Politecnico di Torino

Polytechnic of Turin

College of Engineering

Master of Science in Mechatronic Engineering

Master Degree Thesis

Development of an interface for an electric motor speed control

Supervisors:

Prof. Andrea Mura

Prof. Luigi Mazza

Dott. Edoardo Goti

Candidate: Alessandra Lonigro

Academic year [2021-2022]

Abstract

The electric motor has seen an important expansion in the last decade, in multiple applications such as: automotive, e-mobility, industrial automation, agriculture and several industrial sectors, in order to make the industry more sustainable.

In conjunction with the electric motor evolution, it has been necessary to provide increasingly efficient motor control systems. In this optic, it has been realized a motor control system to facilitate and improve the analysis and study of an asynchronous motor.

The development system is composed of an electrical cabinet with all the most important safety components, a three-phase inverter and a communication method to make simple the interface between the final user and the machine. In particular, the electrical panel design has been developed in order to realize a safety structure able to accommodate the inverter and the relative components for controlling the asynchronous motor. The adopted inverter is the ES350-F0-4K0G-3B model provided by the Cumark company. The employed inverter implements the overall drive for the motor control, starting from the AC-DC power supply stage converter, passing through the logic control, to end with the inverter power module to drive the motor.

Finally, to realize a user-friendly interface with the motor, a graphic user interface (GUI) has been designed. The GUI has been developed with MATLAB environment, and it implements the serial communication between the inverter and the PC, through Modbus RTU protocol, common industrial robust system. This solution allows to realize a speed control system to manage the motor in remote mode, in particular the GUI control panel has been focused on the setting of the speed reference and of the acceleration and deceleration time for adjusting the speed variations, moreover it provides the monitoring of the voltage and the frequency behaviour, and the main drive information.

INDEX

1. Asynchronous motor

In this chapter there is a brief theoretic background about the asynchronous motor functioning, the constructive aspects and mainly the methods for managing the rotor velocity, that is the aim of this work. In particular, the asynchronous motor used in the laboratory, has been designed by Lafert Group, suitable mainly for industrial application, thanks to its low energy consumption and its high efficiency.[10]

1.1. Construction

The asynchronous motor consists of a fixed part, the stator, and a rotating part, the rotor. The former is composed of a reed valve pack with the shape of a circular crown in ferromagnetic material, in order to reduce the iron losses, caused by parasite currents. It is placed inside a cast iron casing (in our case it is in aluminum), to protect the internal part of the machine from the external agents. On the internal surface of reed valve pack there are quarries, in which the stator windings can be accommodated and linked whit star or triangle connection, according to the design requirements and needs.

The rotor is realized similarly to the stator, but it is located inside the stator gage and mechanically connects to the machine shaft. On its external surface it presents the rotor windings.

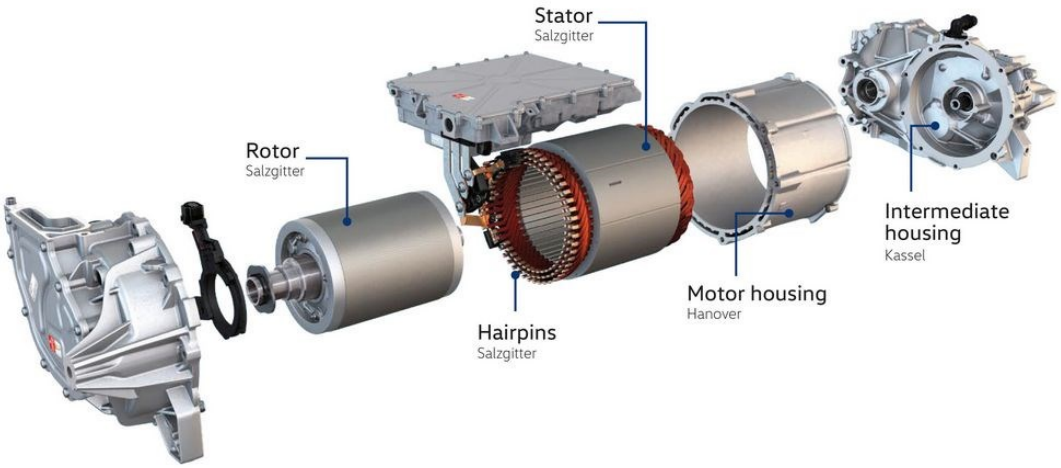

Figura 1.1.1 Asynchronous motor main components

1.2. Principle of functioning

The functioning of three-phase asynchronous machine is based on the magnetic rotational field concept, conceived by Galileo Ferraris.

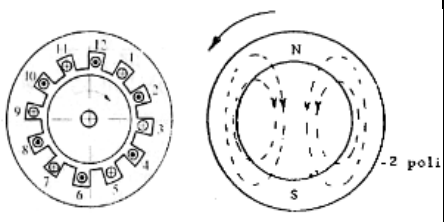

Figura 1.2.1a-Magnetic rotating field generated by statoric currents

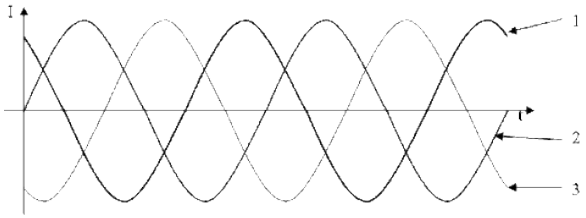

Figura 1.2.1b -Statoric currents behaviour

The three-phase power supply generates three out-of-phase voltages on the windings, consequently three equal and phase shifted currents start to flow in the stator winding.

$$v_A = V_M \cos(\omega_s t + \psi) \qquad (1.2.1)$$

$$v_B = V_M \cos\left(\omega_s t + \psi - \frac{2}{3}\pi\right) \quad (1.2.2)$$

$$v_C = V_M \cos\left(\omega_s t + \psi - \frac{4}{3}\pi\right) \quad (1.2.3)$$

The sum of three currents, equations (1.2.4-1.2.6), provides the total stator current with the amplitude value I_M, and the phase φ given by the angle between the rotating vector and each phase. Mathematically, the current is described by means the rotating vector I_S with a speed ω_s, equal to the power supply pulsation (1.2.7).

$$i_A = I_M \cos(\omega_s t + \psi - \varphi) \quad (1.2.4)$$

$$i_B = I_M \cos\left(\omega_s t + \psi - \varphi - \frac{2}{3}\pi\right) \quad (1.2.5)$$

$$i_C = I_M \cos\left(\omega_s t + \psi - \varphi - \frac{2}{3}\pi\right) \quad (1.2.6)$$

By composing the three current, the rotating vector Is is achieved.

$$I_S = I_M e^{j(\omega_s t + \psi - \varphi)} \quad (1.2.7)$$

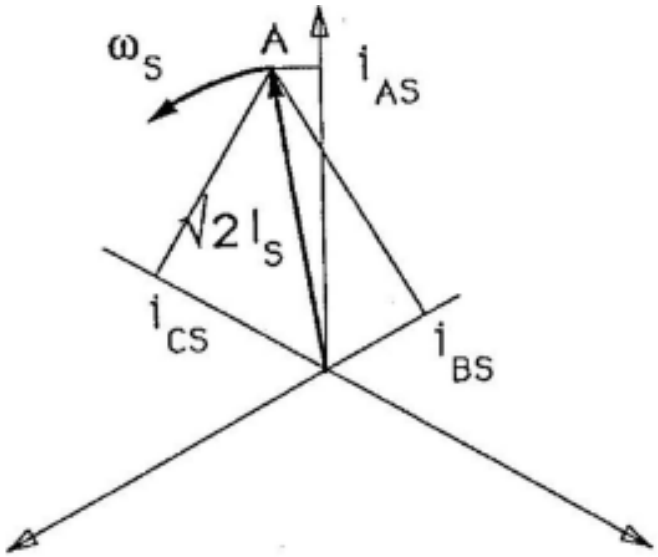

Figura 1.2.2-Representation of rotating current vector

In each line, the flowing current generates a magnetic field in the air gap. The magnetic fields for each phase are:

$$H_A = \frac{4}{\pi}\frac{K_S\ N_S}{2\partial K_{SAT}} i_A \cos\theta \tag{1.2.8}$$

$$H_B = \frac{4}{\pi}\frac{K_S\ N_S}{2\partial K_{SAT}} i_B \cos(\theta - \frac{2\pi}{3}) \tag{1.2.9}$$

$$H_C = \frac{4}{\pi}\frac{K_S\ N_S}{2\partial K_{SAT}} i_C \cos(\theta - \frac{4\pi}{3}) \tag{1.210}$$

It is possible to notice that the contribute of cosine indicates that the field has a cosinusoidal waveform static in the space, but time variable, this concept is expressed as pulsating magnetic field (∂ is the airgap width, K_{SAT} is the iron saturation level of the machine, $K_S N_S$ is the stator winding distribution coefficient).

By considering that the windings are crossed by currents with the same maximum value I_M, it follows that the maximum value of the airgap magnetic field is:

$$H_M = \frac{4}{\pi}\frac{K_S\ N_S}{2\partial K_{SAT}} I_M \tag{1.2.11}$$

Summarizing the fields of each phase, the final rotating magnetic field is achieved, formula 1.2.12.

$$H_S = \frac{3}{2} I_M \cos(\omega_s t + \psi - \varphi - \theta) \tag{1.2.12}$$

To better analyze the formulation for the magnetic field, it is possible to consider a polyphase statoric winding with a number m_s of phases supplied by a symmetric current system. This last induces the e.m.f with sinusoidal behaviour and a rotating speed equal to the power supply pulsation ω_S, the formulation of these forces is expressed in the following equation ($K_S N_S$, $K_R N_R$ are respectively winding distribution factor and total number of coils for stator and rotor).

$$F_s = \frac{m_s}{2} F_{SM} = \frac{\frac{m_s}{2}4}{\pi} K_S N_S I_{MS} = \frac{\frac{m_s}{2}4}{\pi} K_S N_S \sqrt{2} I_S \tag{1.2.13}$$

Where F_{SM} is the maximum amplitude of the sine wave, while the last mathematical step is obtained by considering the effective value of $I_S = \frac{I_{MS}}{\sqrt{2}}$. The same reasoning is exploited for the rotor winding, in this case the spatial sine wave rotates with a $\omega_S - \omega_R$ speed, because it's considered in a reference system integral with the rotor.

$$F_R = \frac{m_R}{2} F_{RM} = \frac{\frac{m_R 4}{2}}{\pi} K_R N_R I_{MR} = \frac{\frac{m_R 4}{2}}{\pi} K_R N_R \sqrt{2} I_R \tag{1.2.14}$$

The resulting wave is made by the sum of the two forces contributions:

$$F = F_S + F_R \tag{1.2.15}$$

In a reference system integral with the stator, it's possible to represent this force as a time rotating vector with speed ω_s, and amplitude $F_M = F_S + F_R$, the head and direction are obtained with the vectorial sum as presented in the figure 1.2.3

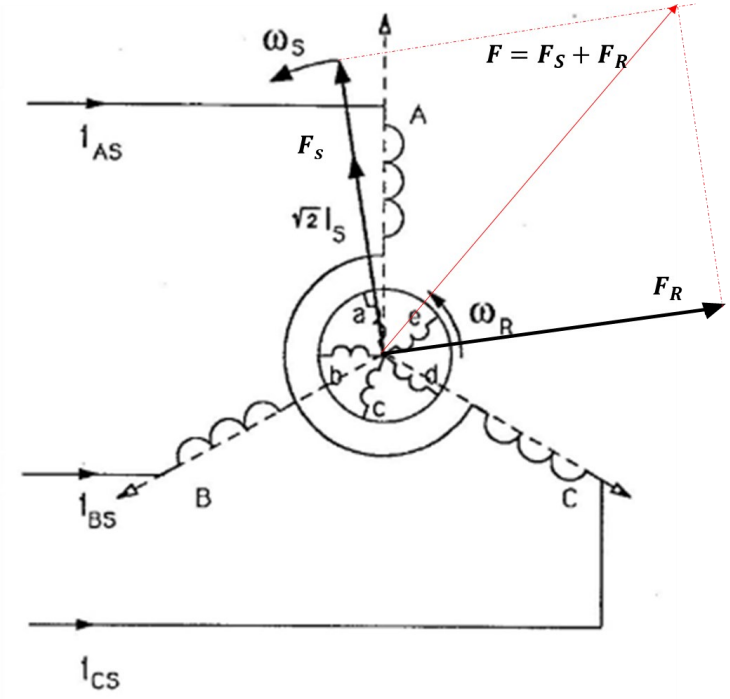

Figure 1.2.3-Vectorial representation of e.m.f on a three-phase winding

This force is relating to the magnetic field on the air gap H with amplitude $H_M = \frac{F_M}{2\alpha}$, with α the gap depth, and the correspondent resultant magnetic induction B with amplitude $B_M = \mu_0 H_M$. The spatial wave associated to the magnetic induction gives rise to the air gap flux Φ, its amplitude Φ_M is computed as below:

$$\Phi_M = \mathrm{B_{med}} S = \frac{2}{\pi} B_M S = \frac{2}{\pi} B_M \frac{\pi D l}{2} \tag{1.2.16}$$

Where B_{med} is the media of the induction and *S* is the crossed surface.

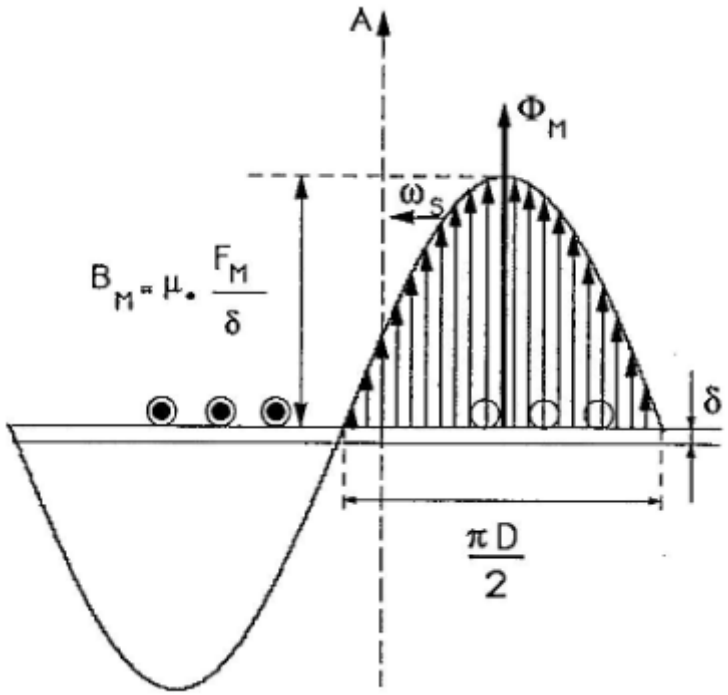

Figure 1.2.4-Rapresentation of air gap magnetic induction

As consequence of the rotation of the spatial flux wave on the air gap, the flux, which is concatenated with the each statoric phase, varies over the time with sinusoidal law. The expressions of the time vector representative the concantenated flux for the stator and the rotor are remarked in equations (1.2.17), (1.2.18).

$$\lambda_{ms} = K_S N_S \Phi \tag{1.2.17}$$

$$\lambda_{mR} = K_R N_R \Phi \tag{1.2.18}$$

Due to the flux time variation, the e.m.f are generated for the statoric and rotor windings and described as:

$$E_{MS} = j\omega_s \lambda_{ms} \tag{1.2.19}$$

$$E_{MR} = j(\omega_s - \omega_R)\lambda_{mR} \tag{1.2.20}$$

1.3. Equations and equivalent circuit of the asynchronous machine

The flux Φ_M, as previously explained, is generated by both current of stator I_S and rotor I_R. Actually, a part of this flux fails to cross the air gap, this is the dissipated flux and it's possible to distinguisch two contributions for the stator and the rotor, respectively through these relations:

$$\lambda_{ds} = I_S L_{dS} \tag{1.3.1}$$

$$\lambda_{dR} = I_R L_{dR} \tag{1.3.2}$$

The quantities L_{dS}, L_{dR}, are the dissipated inductance of the stator and the rotor. These fluxes are the responsible of the dissipated e.m.f.

$$E_{dS} = j\omega_s L_{dS} I_S \tag{1.3.3}$$

$$E_{dR} = j(\omega_S - \omega_R) L_{dR} I_R \tag{1.3.4}$$

By introducing a rotating reference system that has a speed ω_s, the rotating vector becomes phasor, thus it is possible to define the motor's equations, by recalling the formula in (1.2.19), (1.2.20).

$$\begin{cases} V_s = R_S I_S + j\omega_s L_{dS} I_S + E_{MS} \\ E_{MR} = R_R I_R + j(\omega_s - \omega_R) L_{dR} I_R \end{cases} \tag{1.3.5}$$

The following picture summarized the aforementioned explanation under form of equivalent circuit, where are reported the series winding equivalent resistance, the parrassitic series inductance for the dissipated flux and the equivalent inductance for the effective e.m.f., rispectvely for the stator and the rotor.

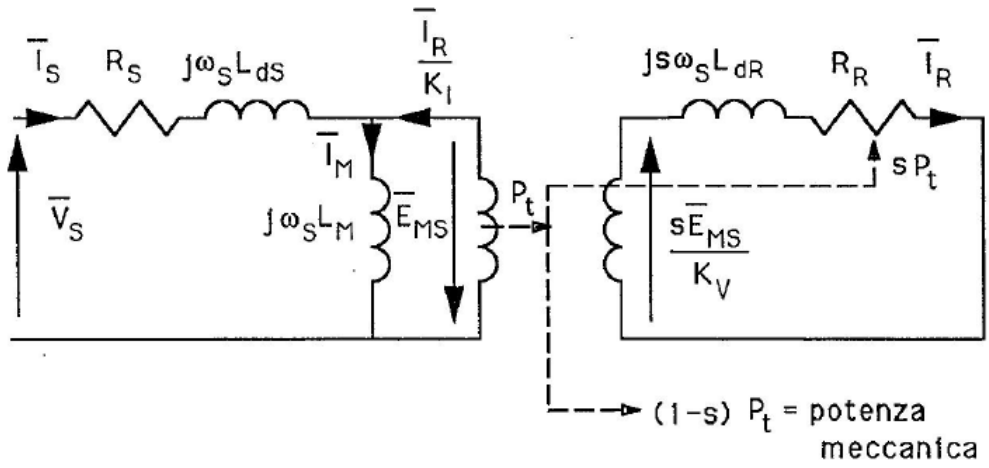

Figure 1.3.1-Equivalent circuit of asynchronous machine.

1.4. Synchronism speed and slip

The synchronism speed is a master concept to distinguish an asynchronous machine from a synchronous one, moreover is a primary element to describe the mechanic characteristics and to evaluate the speed regulation.

By supposing that the stator of an asynchronous machine with n_p couples of poles is supplied by a three-phase symmetric system of sinusoidal voltages with pulsation ω_s, the resulting magnetic field rotates respect to the stator with a mechanic angular speed:

$$\Omega_S = \frac{\omega_s}{n_p}\left[\frac{rad}{sec}\right] \qquad (n_{S_{RPM}} = \frac{60 f_s}{n_P}\left[\frac{giri}{min}\right]) \tag{1.4.1}$$

This field induces in the rotoric windings a symmetric system of e.m.f (electromotive force) with a pulsation $\omega_R = n_p(\Omega_S - \Omega_m)$, it depends on the

relative speed of rotating stator magnetic field w.r.t. rotor and on its angular mechanic speed Ω_m. Consequently, the rotoric rotating magnetic field rotates with:

$$\Omega_R = \frac{\omega_R}{n_p}\left[\frac{rad}{sec}\right] \qquad (n_{R_{RPM}} = (n_{S_{RPM}} - n) = \frac{60 f_R}{n_P}\left[\frac{giri}{min}\right]) \qquad (1.4.2)$$

From the interaction of these two rotating magnetic fields with a synchronous speed, a motrix torque is generated, this torque leads the rotation of rotor with a speed ω_R close to ω_S. The rotor never can reach the synchronism speed, because if the condition $\Omega_m = \Omega_S$ is achieved, the relative motion among the rotor and the stator magnetic field does not exist, and the inductive effect stops. Therefore, for the absence of the induced e.m.f and of the rotoric rotating magnetic field, the torque will be null, and the only possible torque will be the breaking torque caused by friction and ventilation.

The rotor can only follow the inductive statoric field, and the difference between the speed of rotating field and the statoric winding $(\Omega_m - \Omega_S)$ is able to produce rotoric currents that bring the necessary torque. As conclusion the rotor rotates only in asynchronous modes, hence the name asynchronous machine.

Starting from the previous considerations, it is possible to introduce the concept of split. The split is the relative difference between the speed rotating magnetic field on air gap ω_S and the electric speed of rotor ω_R.

$$s = \frac{\omega_S - \omega_R}{\omega_S} \qquad (1.4.3)$$

Particular cases:

- $s = 1$, the motor results stopped, it occurs $\omega_R = 0$ and the motor is considered in “blocked rotor”.
- $s = 0$, in this case, it is possible to anayze the induced e.m.f and the machine equations, by recalling $s\omega_S = \omega_S - \omega_R$ the relations are:

$$E_{MR} = j(\omega_S - \omega_R)K_R N_R \Phi_M = js\omega_s K_R N_R \Phi_M \qquad (1.4.4)$$

$$E_{dR} = j(\omega_S - \omega_R)L_{dR} I_R = js\omega_s L_{dR} I_R \qquad (1.4.5)$$

The previous machine equations (1.3.5) become:

$$\begin{cases} V_s = R_S I_S + j\omega_s L_{dS} I_S + j\omega_s K_S N_S \Phi_M \\ E_{MR} = js\omega_s L_{dR} I_R = R_R I_R + j\omega_s L_{dR} I_R \end{cases} \quad (1.4.6)$$

For a null value of the slip, the induced e.m.f is zero, so the torque is not developed with $\omega_S = \omega_R$. For there to be torque in the motor, the split must have always a value different from zero.

1.5. Mechanical characteristics

The mechanical characteristics represents the torque as a function of the speed $C_e = f(\omega_R)$, it quantifies how the rotor velocity incides on the electromagnetic torque, moreover, it is fundamental to understand the functioning region of the motor according to the speed.

The electromagnetic torque is written as:

$$C_e = \frac{P_m}{\omega_R} = \frac{(1-s)P_t}{(1-s)\omega_s} = \frac{P_t}{\omega_s} \quad (1.5.1)$$

For the development of this formula, it is necessary to represent the equivalent circuit from stator point of view, figure 1.5.1. in this way it is possible to compute the transmitting power from rotor to stator.

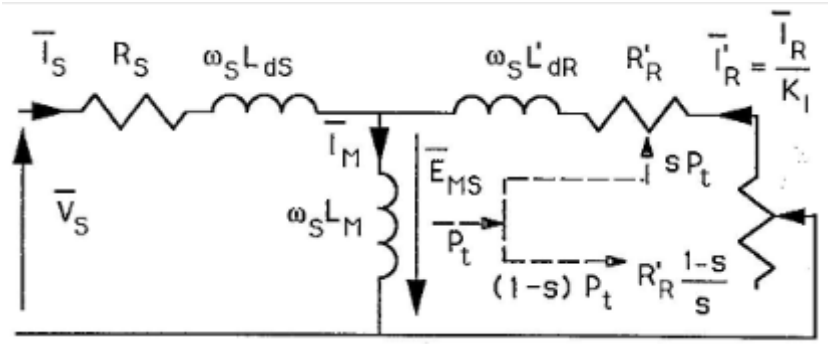

Figura 1.5.1-Equivalent circuit of asynchronous machine from stator side

The side stator circuit is obtained by:

- multiplying the rotor e.m.f $\frac{E_{MS}}{K_V}$ for K_V
- dividing the rotor current I_R for K_I

All the impedence are multiplied for $K_I K_V$, that are respectively the carry current factor of the stator to rotor, and the curry voltage factor of the stator to rotor, in particular K_V represent the ratio between the effective coils of rotor and stator, K_I take into account also the ratio between the stator and rotor phases.

$$K_I = \frac{m_s}{m_R} K_V \tag{1.5.2}$$

$$K_V = \frac{K_S N_S}{K_R N_R} \tag{1.5.3}$$

By recalling the formulation of P_t, transmitting power from rotor to stator is:

$$P_t = m_s \frac{R_R}{s} \left(\frac{I_R}{K_I}\right)^2 = m_s E_{MS} I'_R cos\varphi \tag{1.5.4}$$

The contribution I'_R is the rotor current on the side stator circuit, from now on all the contributes with the pedix are referred to the side stator circuit.

It is possible to explicitate the contributon I'_R and $cos\varphi$, equation 1.5.5, 1.5.6.

$$I'_R = \frac{E_{MS}}{\sqrt{\left(\frac{R'_R}{s}\right)^2 + (\omega_S L'_{dR})^2}} \tag{1.5.5}$$

$$cos\varphi = \frac{\frac{R'_R}{s}}{\sqrt{\left(\frac{R'_R}{s}\right)^2 + (\omega_S L'_{dR})^2}} \tag{1.5.6}$$

By substituting these relations in formula (1.5.4) it is obtained the transmitting power equation:

$$P_t = \frac{m_s (E_{MS})^2 \frac{R'_R}{s}}{\left(\frac{R'_R}{s}\right)^2 + (\omega_S L'_{dR})^2} \tag{1.5.7}$$

After some transformation, by multiplying and dividing by $\left(\frac{s}{R'_R}\right)^2$, and by defining as rotor *transient time constant* the variable $T_{TR} = \frac{L'_{dR}}{R'_R}$, the formula for Pt is:

$$P_t = \frac{m_s (E_{MS})^2 \frac{R'_R}{s}}{1 + (s\omega_s T_{TR})^2} \tag{1.5.8}$$

Finally, by multiplying and divinding by $2\omega_S L'_{dR}$, the final formula for the transmitting power results:

$$P_t = \frac{m_s (E_{MS})^2 2s\omega_s T_{TR}}{2\omega_S L'_{dR}[1 + (s\omega_s T_{TR})^2]} \tag{1.5.9}$$

By considering the formula 1.5.1 it is possible to obtain the formulation for the electromagnetic torque:

$$C_e = \frac{P_t}{\omega_S} = \frac{m_s (E_{MS})^2 2s\omega_s T_{TR}}{2\omega_S{}^2 L'_{dR}[1 + (s\omega_s T_{TR})^2]} = \frac{m_s E_{MS}^2}{2L'_{dR}\omega_S^2} \frac{2x}{(1 + x^2)} \tag{1.5.10}$$

Where $x = s\omega_S T_{TR}$.

For $x = 1$, it's possible to define the maximum value of slip for achieving the maximum torque.

$$s_{emax} = \frac{1}{\omega_s \frac{L'_{dR}}{R'_R}} = \frac{R'_R}{\omega_s L'_{dR}} = \frac{R'_R}{X'_{dR}} \tag{1.5.11}$$

By observing this equation, a variation of rotor resistance could help to adjust the slip. Usually, in practice, the value of s is around a range of $0{,}01 \div 0{,}02$

The first step, to reach the final trend of C_e is represented in the below figure, the two curves are obtained by 1.5.10 equation by considering two distint cases:

- $\omega_R \cong \omega_S$, thus, for low values of s, in the equation (1.5.10) it is possible to neglect the quadratic term x^2, so the torque became $C_e = C_{e\_max} \cdot 2x$, that represents the equations of a straight line.
- $\omega_R \cong 0$, thus, for high values of s, the relation to be considered is 1.5. and it is evident the hyperbolic behaviour.

$$C_e = C_{e\_max} \cdot \frac{2x}{x^2} = C_{e\_max} \cdot \frac{2}{x} \qquad (1.5.12)$$

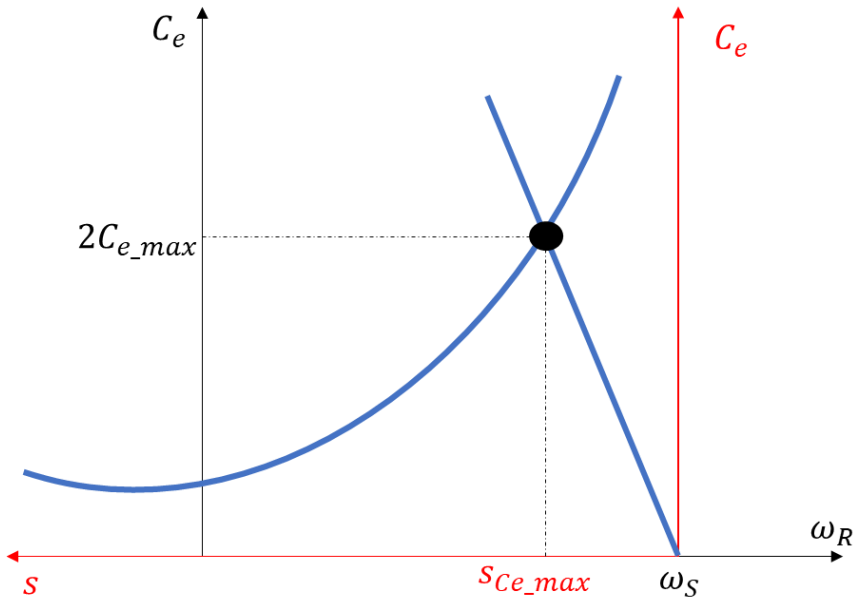

Figure 1.5.2- Electromagnetic torque behaviour

In practice the value of $2C_{e\_max}$ is not achieved (contrary to the ideal computations), and for values close to s_{emax}, it is possible to approximate the

behaviour of the two curves as in the following picture, limiting the effective max value, and “melting” in a single final curve (the red curve in picture 1.5.3)

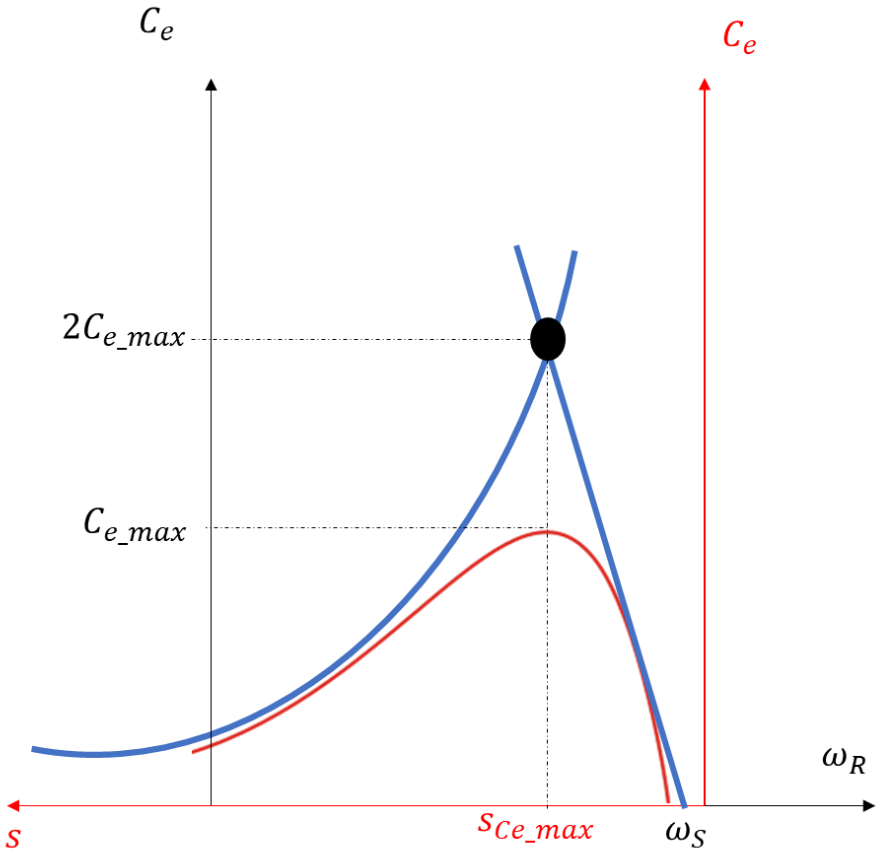

Figure 1.5.3-Electromagnetic torque approximate behaviour

In conclusion the mechanical characteristic of an asynchronous machine is figured in figure 1.5.4, as it seen, it is possible to identify three operating regions.

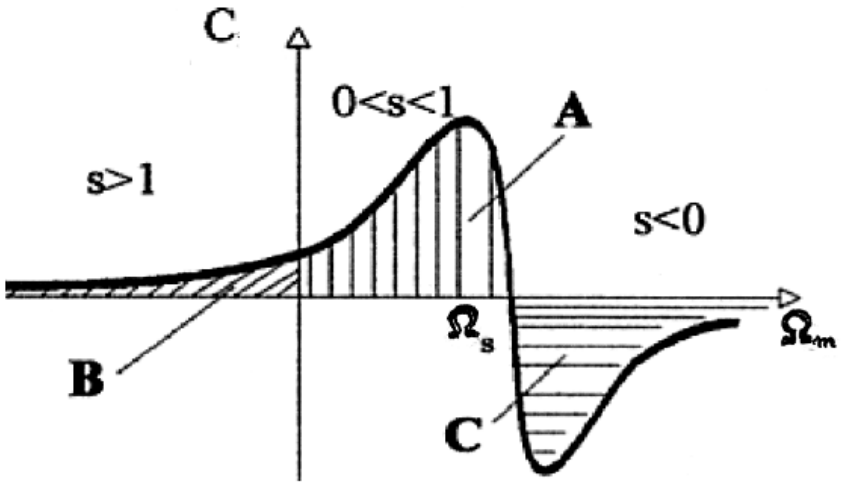

Figure 1.5.4-Mechanical characteristics of asynchronous motor

Operating regions	Description
A=MOTOR	($0 < \Omega_m < \Omega_S$ and therefore $0 < s < 1$) The dissipated energy rapresents the mechanical energy delivered by the machine.
B=BRAKE	Ω_m is opposite to Ω_S, the machine abosrbs mechanical energy and electrical energy from the stator terminals, and dissipates it for Joule effect
C=GENERATOR	($\Omega_m > \Omega_S$, thus $s < 0$); the dissipated energy is equal to the total absorbed mechanical energy.

Table 1.5.1-Operating regions of an asynchronous machine

1.6. Starting of the asynchronous machine

The starting of the machine consists of taking the speed from zero to a certain value.

As broadly said, at the starting the motor presents two main problems:

- High absorption current I_{abs}
- Low starting torque C_{avv}

To face up with these inconvenients and to improve the starting, two parallel ways are proposed:

- Reducing the starting current I_{avv}, that flows when the line connection switch is closed. This reduction can be obtained by executing the starting with a low voltage, the reduction will be proportional to the dropping of the voltage.
- Increasing the starting torque.

For what concern the first option, the starting with a low voltage can be implemented with three different techniques:

1. Insterting of impedences on the power supply line
2. Star-triangle starting
3. Starting with autotransformer

We will focus the most used of these methods, namely star-triangle starting.

1.6.1 Star (Y)-Delta (Δ) starting

The machine is prepared to operate in nominal contidions with a star connection of the stator phases, in such way, each phase is powered by line voltage $V_l/\sqrt{3}$.

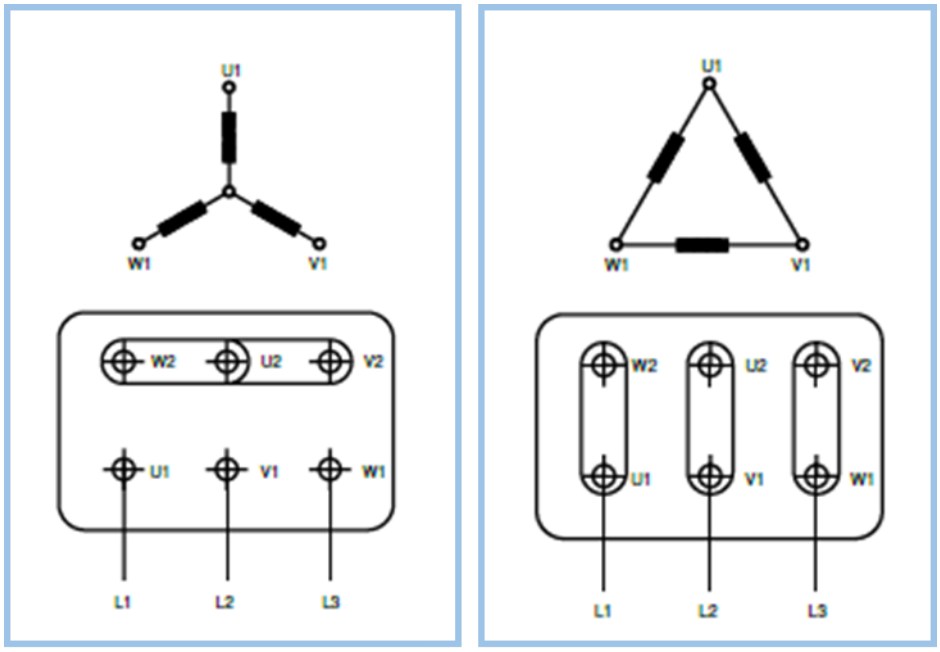

Figure 1.6.1.1- Star connection on the right, delta connection on the left.

Start connection: The star connection is achieved by connecting terminals W2, U2, V2, and by suppling the terminals U1, V1, W1. Phase current and voltage are respectively: $I_{ph} = I_l$; $U_{ph} = V_l / \sqrt{3}$, where I_l is the line current e V_l the line voltage relative to the start connection.

Delta conection: This type of connection is obtained by connecting the final part of a phase with the start of the successive phase. Phase current and voltage are respectively: $I_{ph} = I_l / \sqrt{3}$; $U_{ph} = V_l$.[10]

At the starting the statoric phases will be provisionally connected with a star configuration, in this manner each phase will result to powered by a voltage $V_l/\sqrt{3}$, that it's less than the nominal one.

Finally, after the start up of the machine occurs, the phases are connected to delta for having a functioning with a rated voltage. The conversion of the connections is achieved through a commutator like in figure 1.6.1.1.

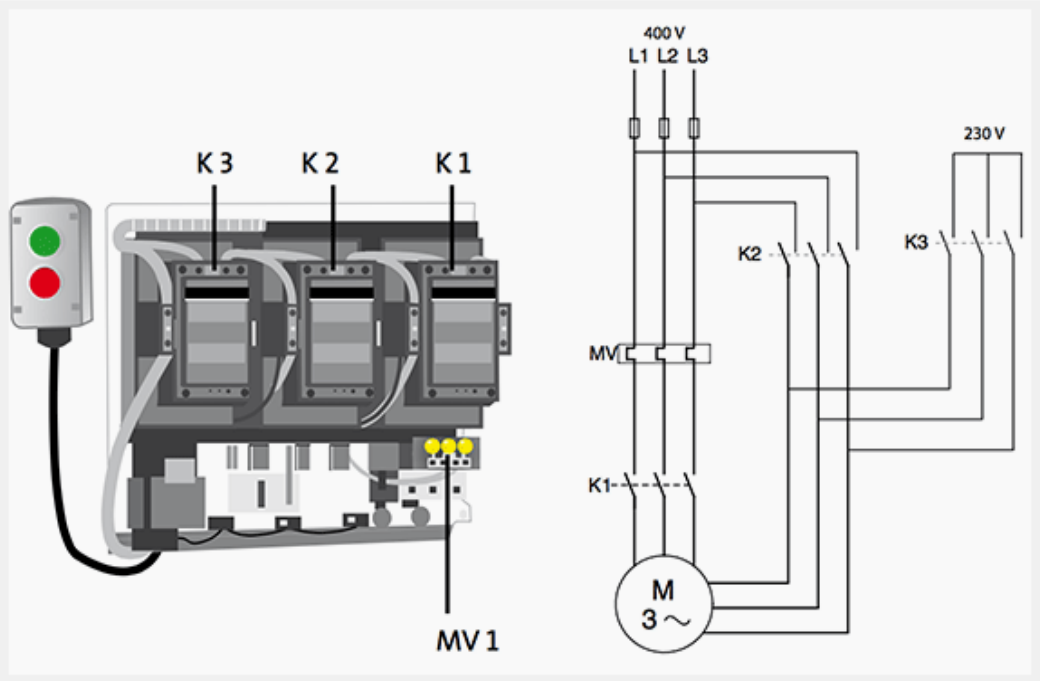

Figure 1.6.1.2-Scheme of star-delta starting for an AC motor

It is possible to highlight the advantages and drawback of this starting method.

Advantages: greater construction simplicity and lower cost.

Disadvantages: possible problems during the star-delta commutations, in terms of currents peacks and short circuit, impossibility of choosing the starting characteristics, because they are predefined with this method. This starting requires an elevate number of contactor as it visible in figure. [2]

For this reason, it is preferable to start the machine directly, according to the selected type of connection.

1.7. Speed regulation

In this section, we will go to elaborate some of the techniques to manage the speed in an asynchronous motor. They are recognized in:

- Supply voltage variation
- Supply frequency variation
- Volt/Hertz control

1.7.1 Supply-voltage variation

This method is valid by considering a supply voltage V_s less than the nominal one, and by mantaining the supply frequency to a constant value (formula 1.7.1.1). The synchronism speed is changeless since its dependency with the frequency. On the other side, the torque is subjected to a reduction (formula 1.7.1.2), as consequence of the supply voltage falling approximable as $V_s \cong E_{MS}$.

$$\frac{\omega_s}{n_p} = \frac{2\pi f_s}{n_p} \tag{1.7.1.1}$$

$$C_e = \left(\frac{1}{2L'_{dR}} \frac{E_{MS}^2}{\omega_s^2}\right) \frac{2x}{(1+x^2)} \tag{1.7.1.2}$$

Con $x = s\omega_s T_{TR} = (\omega_s - \omega_R) T_{TR}$.

The characatheristcs has always the same intersection point with abscissa axes for all voltage values, the only mutation is about the maximum value of the torque, that decreases according to the reduction of the supply voltage.

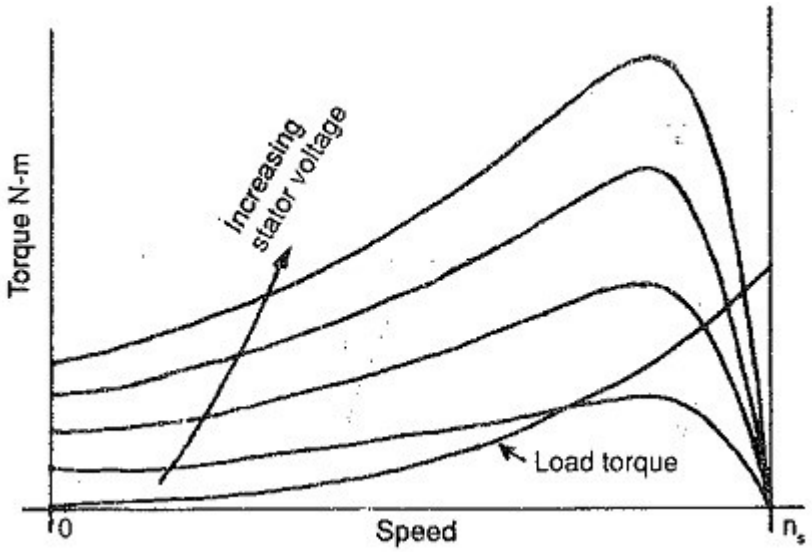

Figure 1.7.1-Torque family due to the voltage variation

By observing the picture (1.7.1), if a load torque $C_r < C_{avv}$ were applied for each curve, it would be possible to notice how the working point moves to the left and therefore how the rotor speed decreases in relation to the variation of V_{sn}.

1.7.2 Supply frequency variation

On the contrary, by keeping constant the supply voltage and varying the frequency, especially by reducing it with respect to the rated frequency, it is possible to carry out another type of speed regulation.

In this case the mechanical rotation speed of the rotating magnetic field varies according to the following relation:

$$\Omega_m = \Omega_S = \frac{\omega_s}{n_p} = \frac{2\pi f_s}{n_p} \tag{1.7.2.1}$$

The consequence is the changing of the rotor speed:

$$\Omega_R = \Omega_S(1 - s) \tag{1.7.2.2}$$

For what concern the torque, keeping constant the value of the supply voltage ($V_s \cong E_{MS}$), a reduction in the frequency corresponds to the increasing of the maximum torque and to the shift to the left of the curve that identify the mechanical characteristic of the asynchronous machine.

$$C_e = \left(\frac{1}{2L'_{dR}} \frac{E_{MS}^2}{\omega_s^2}\right) \frac{2x}{(1+x^2)} \tag{1.7.2.3}$$

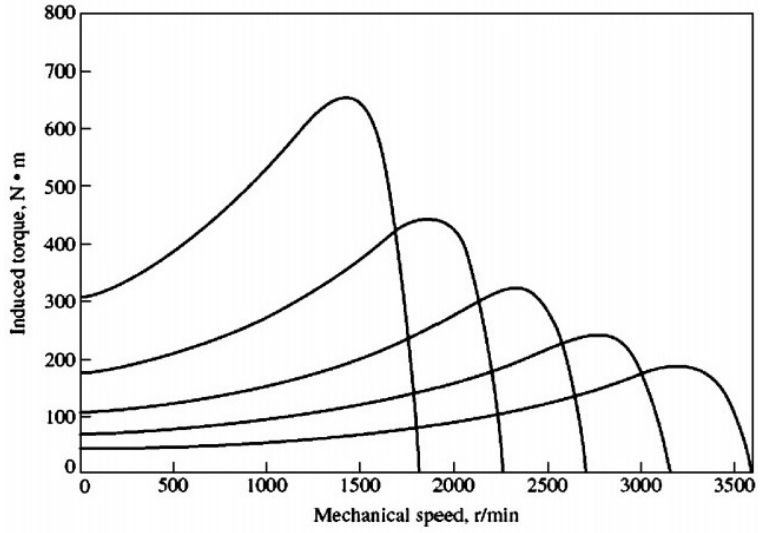

Figure 1.7.2.1[19]-Torque family due to the frequency variation

The rise of the torque, due to the frequency reduction, (with costant voltage) is not convenient, because it yields to a noticeable increase in flux as described in equation 1.7.2.4.

$$\frac{E_{MS}}{\omega_s} = K_S N_S \varphi_M \tag{1.7.2.4}$$

The flux increasing would create problems such as: high current absorbtion and the rise of iron losses. The solution to avoid these inconveniences would be to keep the ratio $\frac{E_{MS}}{\omega_s}$ constant. The third method born as the best trade-off among the previous methods.

1.7.3 Volt/Hertz control

The most efficient way to regulate the speed consists in properly changing the ratio voltage-frequency, in order to preserve the flux value constant. In this way, according to the relation (1.7.3.1) it is possible to highlight that the maximum torque value remains unchanged.

$$C_{e\_max} = \frac{1}{2L'_{dR}} \frac{E_{MS}^2}{\omega_s^2} \tag{1.7.3.1}$$

The most important analyzed behaviours are the following:

- Frequency reduction

  By dropping the supply frequency, the mechanical characteristics moves parallel to itself (to the left). In other words, the maximum torque (equation (1.7.3.1)) depends only on the pulsation of the rotor and not on the power pulsation, whereas torque trend, represented in the equation (1.7.3.2), moves according to difference $\omega_s - \omega_R$. This yields the behaviour described in the graph ().

  $$C_e = C_{e\_max} \frac{2((\omega_s - \omega_R)T_{TR})}{(1+((\omega_s - \omega_R)T_{TR})^2)} \tag{1.7.3.2}$$

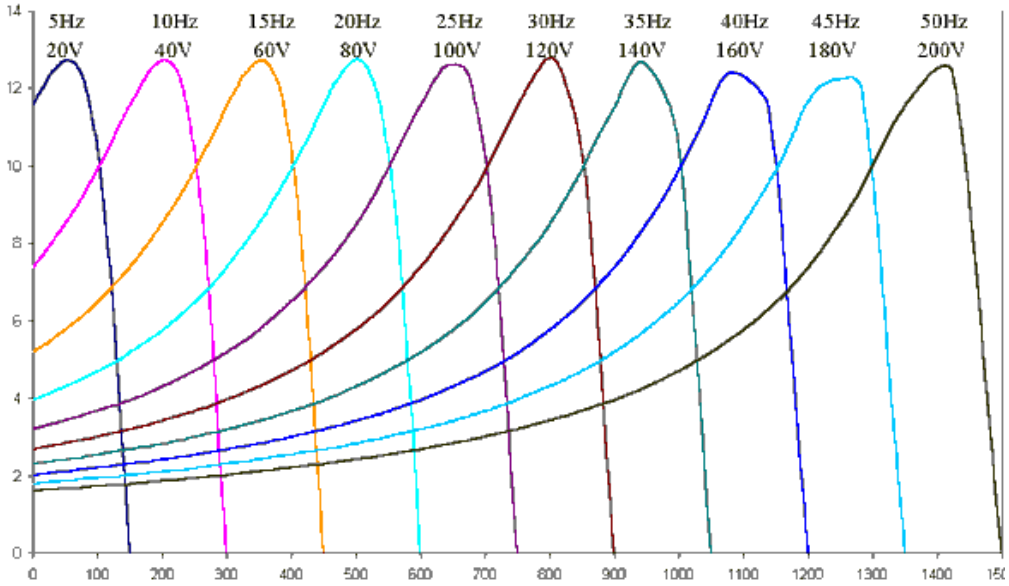

Figure 1.7.3.1-Electromagnetic torque behaviour according to the frequency reduction w.r.t the rated value.

- Frequency increasing

  By increasing the frequency increasing, it is not possible to increase, at the same time, the supply voltage. If we had the supply voltage greater than the nominal one $V_s > V_{sn}$, the insulating material present in the machine would puncture instantly, therefore with the frequency increasing, the supply voltage must be constant and equal to the nominal one to avoid system collaps. The enhance of the frequency produces a flux drop, the machine works in so called “weakned flux”, hence the torque maximum value is lowered with a relative shift towards the right, as showen in the figure (1.7.3.1).

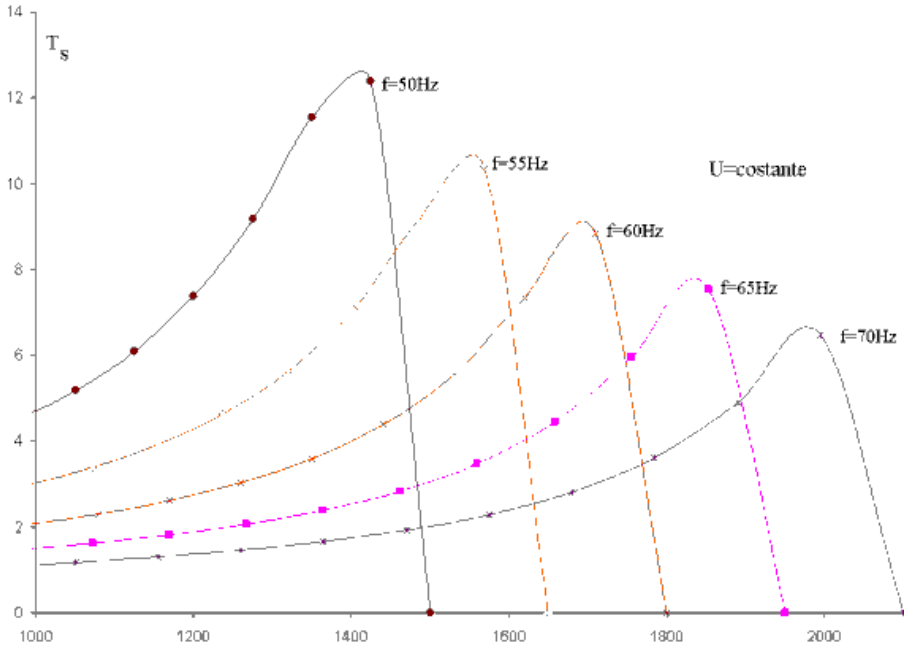

Figure 1.7.3.2-Electromagnetic torque behaviour according to the frequency increasing w.r.t the rated value.

The same behaviour described above could be also analyzed under another point of view. In particolar, in the following graphs, the voltage supply and the flux will be highlighted in relation with the frequency variation.

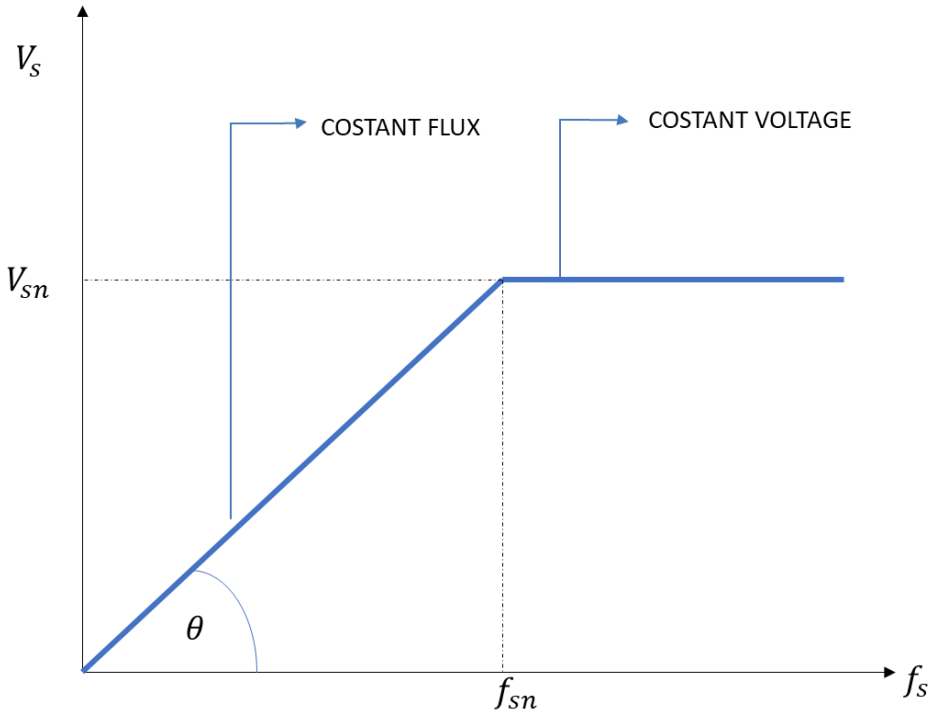

Figure 1.7.3.3-V/F regulation

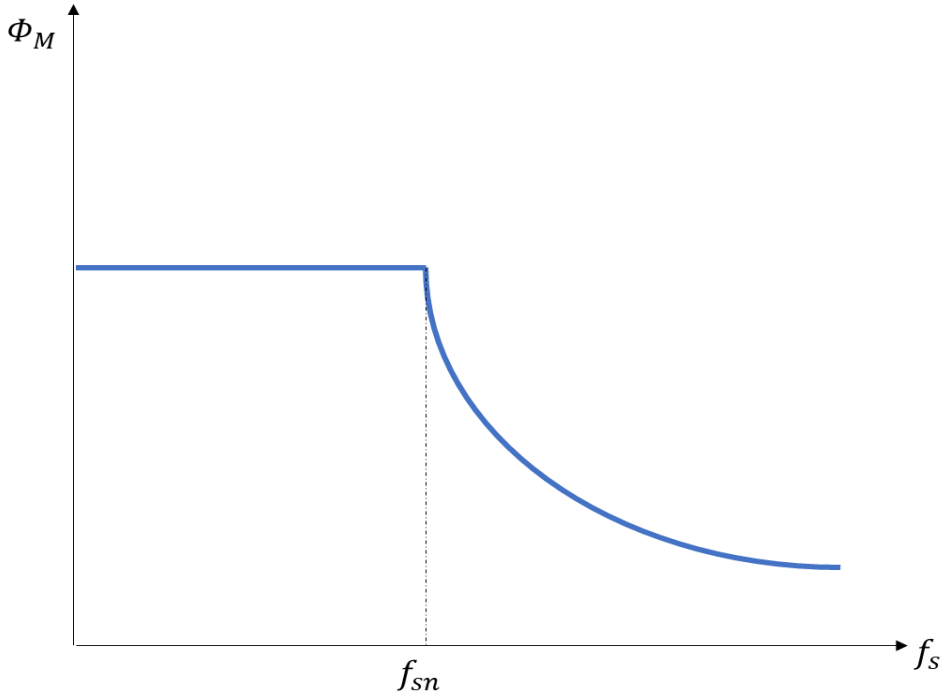

Figure 1.7.3.4-Flux variation according to V/Hz control

In the figure 1.7.3.3, it is possible focus the attention on two different behaviors:

- COSTANT FLUX REGOLATION, when the supply frequency is less than the nominal one $f_s < f_{sn}$ the angular coefficient of the linear tract of $V_s(f_s)$ depends on the flux, according to the relation $tg\theta = K_S N_S \Phi_M$.

- COSTANT VOLTAGE REGOLATION, for $f_s > f_{sn}$ the voltage, as previously explained, must be equal to the rated one. The Φ_M decreases according to the rise of the frequency with an iperbolic trend, respecting the equation (1.7.2.4).

In conclusion, after the speed regulation dissertation, it is possible to support that the more performant technique is the Volt/Hertz regulation. This latter allows to avoid all the inconveniences linked to the other to solutions, as for example the increasing of the flux or the absorption current. For all these reasons the Volt/Hertz control is the one chosen to carry on the design of this thesis.

2. Speed control design

In this chapter have been analyzed and described the requirements needed to properly design the complete speed control system, starting from the asynchronous motor to conclude with the inverter dedicated to the management of the motor control.

Firstly, it is crucial to define the model and the characteristics of the motor and the inverter that are considered for this work.

2.1. Aynchronous motor model

The asynchronous motor for the test bench is a type of machine used for industrial application, for the studied system, the model 'AMPE 90L DA' of Lafert motors Group, figure 2.1.1, is employed. According with the manual [10], the motor lies in the class of motors with axis height 90 mm or higher in standard configuration, thus suitable for operating with an inverter controller, and considering the following requirements:

- "The maximum applicable voltage is 500 V, with peak voltages $\hat{U} \leq 1460$ V and $\frac{du}{dt} \leq 13 \frac{kV}{us}$. For higher inverter output voltages, or for voltage values peak, or $\frac{du}{dt}$ higher, special isolation systems are required." [10]
- "In applications where the load torque curve has a quadratic trend at varying the speed, the motors can operate by delivering their nominal torque." [10]
- "For constant torque applications, the nominal torque of self-ventilated motors, at low speeds, it must be reduced due to lower ventilation. Depending on of the adjustment range, it may be advisable to use servo ventilation." [10]
- "Motors with shaft heights from 90 to 112 are suitable for operation with a frequency maximum inverter output of 60 Hz (eg, applications with

quadratic torque, adjustment range 1:10, such as pumps or fans). As regards higher frequencies, a special series with name is available on request.” [10]

Figure 2.1.1- Asynchrnous motor for Inverter

The plate data of the asynchronous motor are described in figure 2.1.2, it is evident that the rated values depend on the star or delta configuration. It has been decided to connect the phases with star connection and a rated frequency of 50 Hz, thus the plate data to be considered are the following:

- Model: AMPE 90L DA 2
- Number of poles: 2
- Rated frequency: 50 Hz
- Rated power: 3kW
- Rated voltage: 400V
- Rated Current: 6.3A
- Rated speed: 2865rpm
- Power factor: 0.80
- Efficiency class: IE3

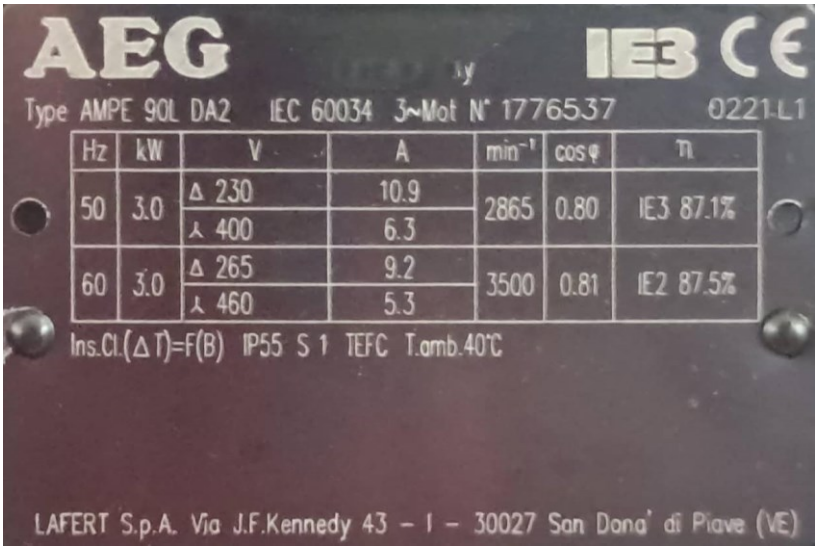

Figure 2.1.2. Plane data of asynchronous motor

2.2. Inverter model

The Inverter used in this project is the model ES-350-F0-4K0-3B of Cumark company, figure 2.2.1, it is a type of driver usually exploited in combination with

asynchronous or synchronous machine. “The drive has the characteristics of stable, reliable, intelligent, easy usage” [9]. The plate data are explained in figure 2.2.2 and in the below relative legend.

Figure 2.2.1- External Layout Inverter ES-350-F0-4K0-3B

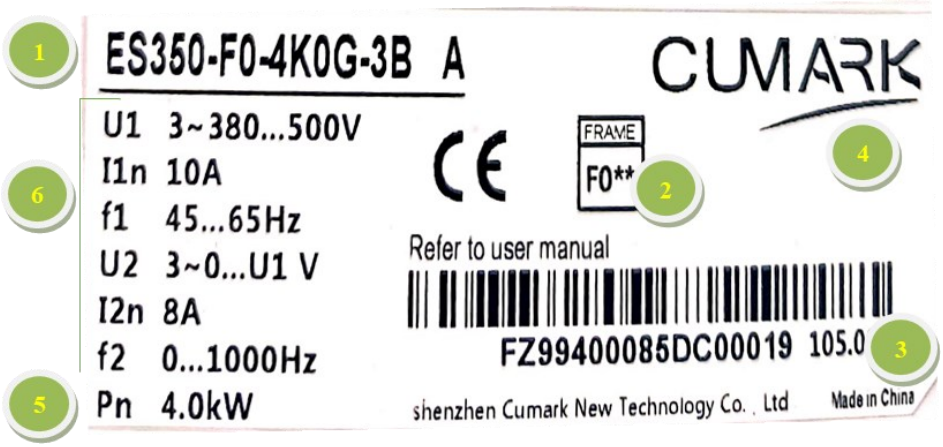

Figure 2.2.2-Inverter type plate description

Legend:

1=Model number, 2=Shapes and volumns, 3= Serial number, 4= Enterprise name, 5=Power, 6=Voltage/Current/Frequency

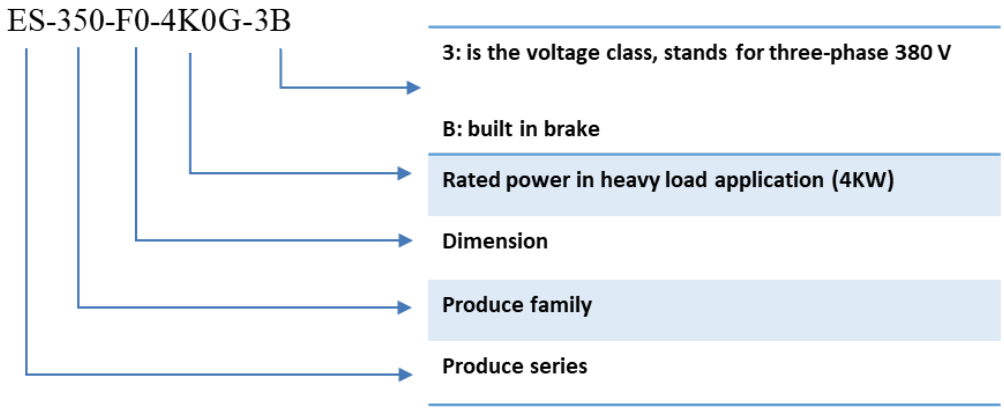

Figure 2.2.3 Model code legend

The inverter represents a central role in the control motor system. Once analayzed in detail all the specification reported in the datasheet, it has been necessary to define the best strategy to control the motor, among the possible modes supported by the inverter:

- Open-loop V/F (voltage vs. frequency) control
- Open-loop sensorless vector control (SVC)
- Close-loop vector control

The absence of the encoder for the exploited machine reflects the impossibilty of take advantage of close-loop control design. Assumed the impossibility to realize a closed loop control, the choice is fell on the open-loop speed regulation, and in particular it has been implemented the V/F control, as explained in the chapter 1.7.3.

2.3. Speed drive with Inverter

The speed control drive consists of the three-phase inverter, it is a device able to manage the possible load variations, with a variable frequency, by adjusting in a precisely manner the speed or the torque.

The internal scheme of an inverter, showed in the figure 2.3.1, provides an essential description of the main components that allow the motor control.

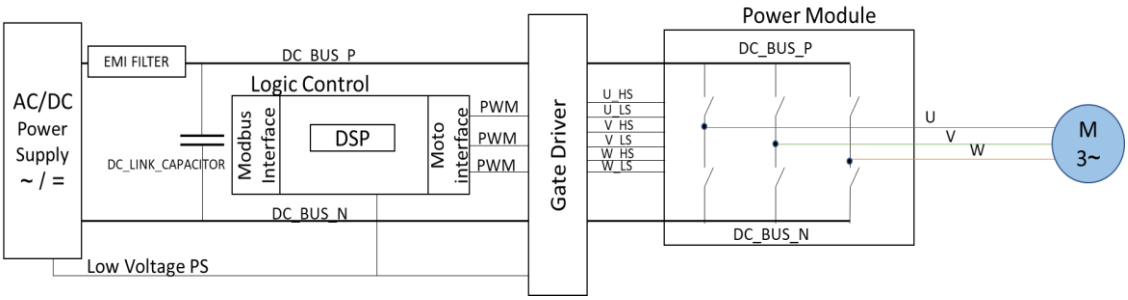

Figure 2.3.1-Block scheme for a three-phase inverter

The main parts of the general scheme are:

1. AC-DC converter
2. EMI Filter
3. DC-Link Capacitor
4. Logic Control
5. Gate Driver
6. Inverter Power module

AC-DC Converter

The first stage consists of a circuit able to convert the AC three-phase power supply in the DC voltage signal. With the block AC-DC in the block scheme is included both the High voltage DC bus reference to provide to the motor block, and the low DC voltage to supply all the block in the inverter system. In particular, the detailed architecture and design of the conversion circuit are not available in the datasheet provided by the Cumark ES350-F0, the inverter exploited in the thesis work.

EMI FILTER

The electric magnetic interference filter (EMI filter) is a circuital block, fundamental to avoid damages to the inverter system but also to reduce the noise level. In particular, the filter circuit is needed to reduce the conducted emission towards the electrical network but also to cut-off the noise coming from the power supply source and to stabilize and to make robust the inverter high voltage power reference. In particular, the detailed architecture and design of the filter strategy are not available in the datasheet provided by the Cumark ES350-F0, the inverter exploited in the thesis work, and not further filter circuits have been added.

DC-LINK CAPACITOR

The DC link capacitor is placed on the High voltage power reference, and it is needed to mantain the as constant as possible voltage on the DC, reducing the ripple on the power line and providing the high current density required by the motor. In particular, the dimension and the technology exploited for the capacitor are not available in the datasheet provided by the Cumark ES350-F0, the inverter exploited in the thesis work.

LOGIC CONTROL

The logic control circuit is acted to manage the states of the motor control, the sensor monitoring and the communication interface. The inverter system adopted in the thesis work is based on the DSP controller and on the Mobus protocol communication, based on RS485 bus. On the other side, the logic controller is needed to generate the PWM signal to properly control the motor, according to the speed, voltage, and current regulation.

GATE DRIVER

The PWM signal generated by the logic block are not directly connected to the gate of the power module. To increase the robustness of the system and amplify the power of the PWM signal, the Gate Driver components are fundamental. The gate driver as said are components that provides high current for the gate of the high-power devices of the power module. Furthermore, they are acted to manage the whole inverter bridge system. Their main duty consists in avoiding situation that could be disastrous for the inverter. For example, they must control and avoid the leg-short condition by controlling step-by-step the PWM signal on the two gates on the same leg. Other possible features linked to the gate drive could be the DESAT base overcurrent protection, overtemperature protection, faults detection and other important functions to preserve the motor control. In particular, the detailed architecture and design of the gate driver circuit are not available in the datasheet provided by the Cumark ES350-F0, the inverter exploited in the thesis work.

INVERTER POWER MODULE

The inverter power module is the block composed by the switching devices exploited for the power conversion from the DC-BUS voltage to the three phase

AC signal for the motor control. The circuit as reported in 2.3.2 is an inverter bridge composed by the Inuslated Gate Bipolar Transistor.

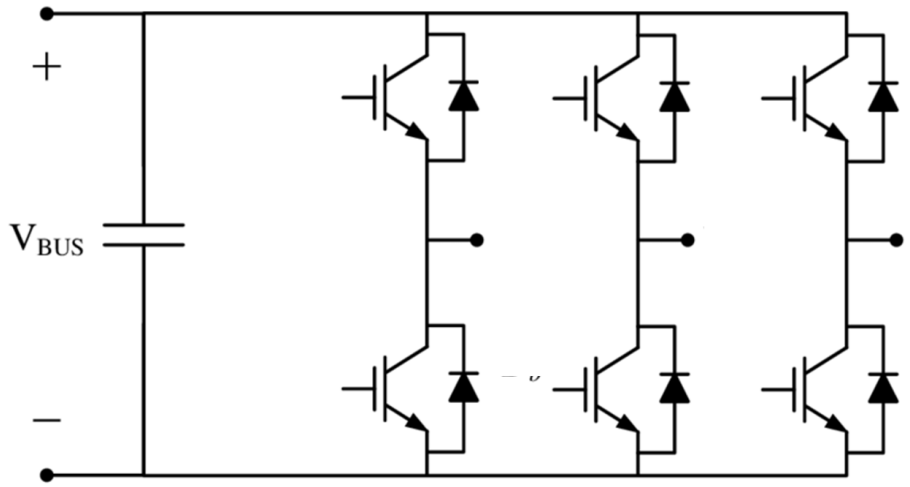

Figure 2.3.2- Scheme of electrical driving that involves inverter

The IGBT is a semiconductor device which presents the BJT and the MOSFET advantages. The power device is suitable for the high current, and it combine the advantages of the high input impedance (MOS) with low saturation current (BJT). The input is composed by a low power MOS to drive the output power BJT.[18]

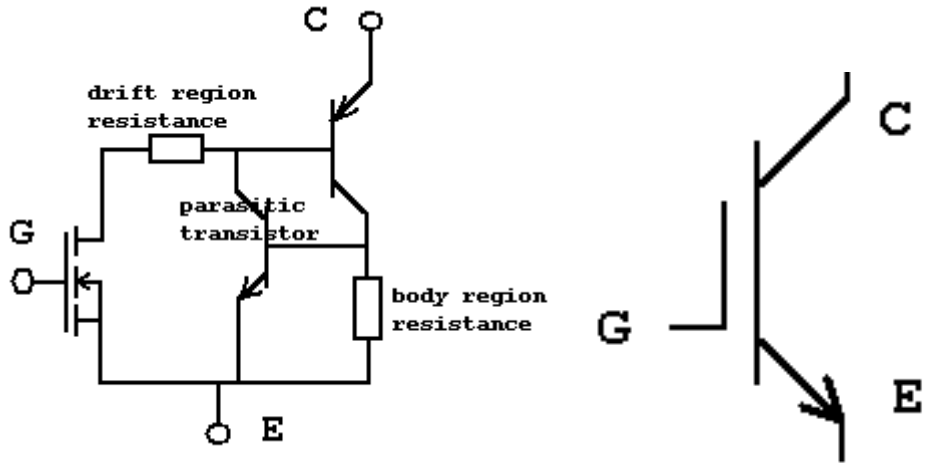

Figure 2.3.3- IGBT connection scheme [18]

Two power devices are placed along three parallel legs, one for each motor phase. The IGBTs directly connected to the DC_BUS positive reference are called “high

side" ones, whereas the devices with the emitter (E) connected to the negative reference of the DC-BUS are named "low side". Every IGBT could be described as simple switch with its on and off state. By properly drive the gate of the IGBT in on and off condition throught the PWM technique is possible generate three phase shifted sinusoidal signals for the phase U, V and W of the motor. The rotation per minute (RPM) of the motor is strictly linked to the switching frequency of the IGBTs, and as it the voltage level provided is proportional to the PWM maximum duty cycle, that define the time on and the off time of the IGBT.

2.4. Asynchronous machine speed regulation with open-loop V/Hz control

The considered control mode to realize the speed regulation is the open-loop V/Hz control. The figure 2.4.1 represents a brief description of this control, in which the inverter implements the dashed part.

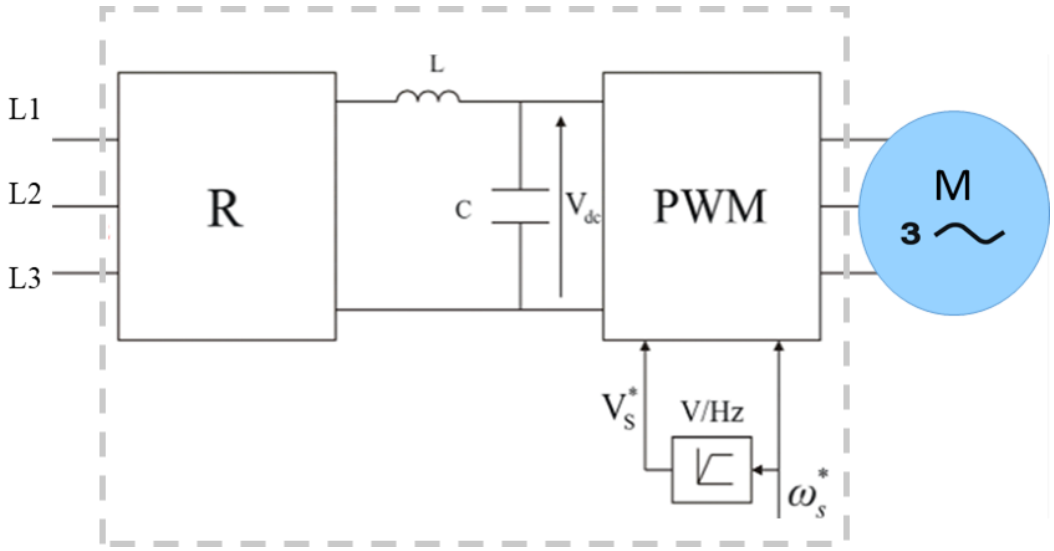

Figure 2.4.1-Open-loop control with V/Hz regulation and PWM

In the open-loop control the desired speed ω_S^* is set by the user and the inverter, by means of V/Hz regulation, is able to maintain the stator magnetic flux constant

without any feedback control loop, by holding the supply voltage amplitude proportional to its frequency.

The variation Voltage/Frequency is achieved by driving these devices with the PWM technique managed by the on-board microprocessor and gate driver amplifier. [8] These steps are integrated in the inverter, that manages the V/F curve with an intelligent adaptive mode, block V/Hz in figure 2.4.1.

Inverters based on PWM technology have MOSFETs in the switching phase of the output. Nowadays most of the inverter provide this mode and are able to produce AC voltage for different magnitudes and frequencies. There are multiple protection and control circuits in these types of inverters. An important aspect, that take advantage of the PWM technique, is their application in different contests, because it makes them suitable and ideal for different connected loads.

The inverter control logic, based on the control signal, will increase or decrease the output voltage according to the explained curve in figure 1.7.3.3, thus the variation of output voltage will be proportional to the frequency, this statement is valid for value of frequency less than the nominal one, $f < f_{sn}$, when the frequency exceeds the nominal value, the voltage must be equal to the rated voltage in order to avoid inconvenient rise of flux.[8]

It is possible to highlight the advantages and disadvantages of using the open-loop control technique in the following table.

ADVANTAGES	• The absence of a direct measurement of the quantity to be controlled has a positive impact on the costs and implementation times of the controller, as well as on the weight. • The possibility of avoiding sensors for measuring the quantity to be controlled entails an advantage in terms of reliability, as the correct operation of the control system will not depend on the operation of the sensor.

	or on the system for acquiring the data read by the sensor.[14]
DRAWBACKS	• The need to develop an accurate mathematical model for experimental tests on the system leads a consequent increase in development costs. • Weak robustness to parametric variations in the model, due to component aging. • Poor robustness in the presence of disturbances acting on the system. [14]

Table 2.4.1- Adavantages and Disadvantages for open-loop control

2.4.1 PWM technique

The PWM (Pulse width modulation) method is a technique of electric power regulation, based on the pulse width modulation, as suggests the name.

This technique is used for speed motor regulation systems, by operating on the voltage supply. It is possible to consider that if a motor is supplied by a pulse sequence, thanks to the mechanic motor inertia, the motor rotates according to a continuous motion, proportional to the medium value of these pulses.

The main important concept to develop this theory is the duty cycle (DC), that is the ration between the time interval in which a signal assumes a high value T_{ON} and the total period $T = T_{ON} + T_{OFF}$, figure 2.4.1.1.

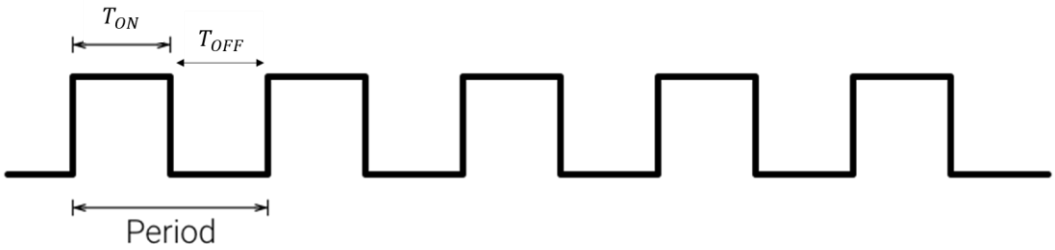

Figure 2.4.1.1-Square wave

The PWM signal is a square wave of variable duty cycle which allows to control the absorbed power of an electrical load varying the duty cycle. The sequence of pulse is repeated according with a certain frequency. The modulation of DC allows to obtain an average voltage variation and consequently a speed regulation.[13] It is possible to explain the voltage variation by analyzing some case of duty cycle.

Duty cycle=25%, the pulse width is minimal, thus they have a short Ton. The resulting voltage is the 25% of the supply voltage, the absorbed power results to be minimal.

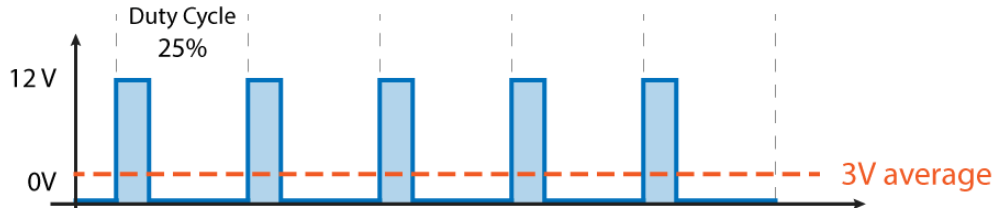

Figure 2.4.1.2-Square wave with 25% of Duty-Cycle

Duty cycle 50%,$T_{ON} = T_{OFF}$, The pulses have a width equal to half of the total period, in this case the voltage has an average value.

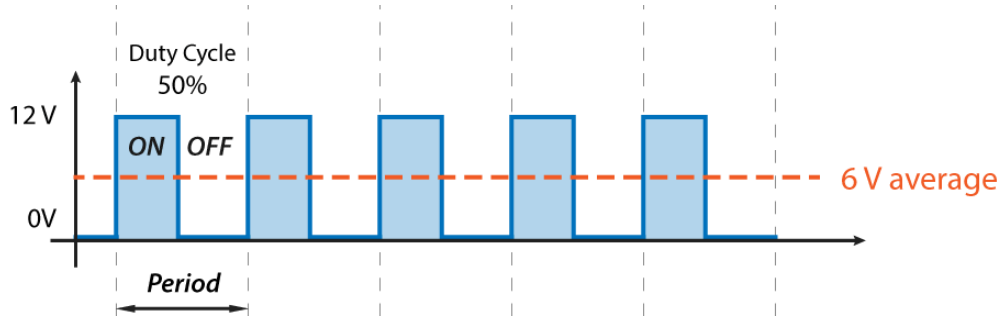

Figure 2.4.1.3-Square wave with 50% of Duty-Cycle

Duty cycle 90%, The pulses almos occupy the allowed time duration; the power reaches practically the maximum value, the same happens to the average voltage.

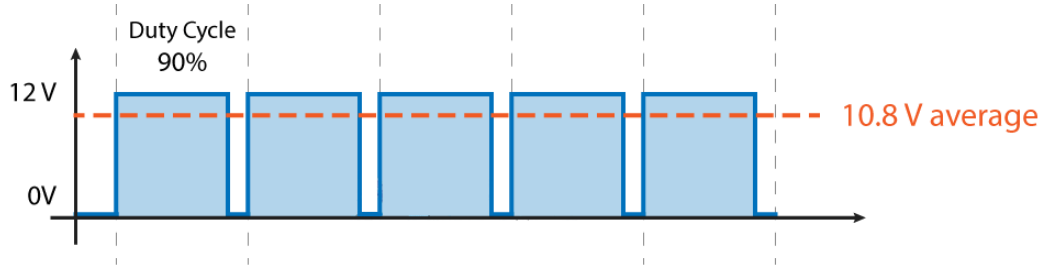

Figure 2.4.1.4-Square wave with 90% of Duty-Cycle

Assumed all the previous consideration about the PWM, the technique can be applied to the motor control, driving the bridge inverter circuit.

In the three-phase motor control, the PWM signals, with its duty-cycle and frequency is applied to the gate (G) of the power device (see the chapter 2.3. and the figure 2.3.2). Each device is switched on state or off state with a certain time (duty-cycle) and a certain frequency, in order to generate a sinusoidal signal for the motor. The amplitude and frequency of the sinusoidal shape is linked to the PWM frequency and duty-cycle as described below.

1) The PWM GENERATION

The PWM is generated by the microcontroller by comparing a sinusoidal wave of frequency f_{el}, with a triangular wave of frequency f_{sw}. The frequency of the the sine signal represents the fundamental frequency, whereas the frequency f_{sw} the switching frequency applied to the IGBT gate.

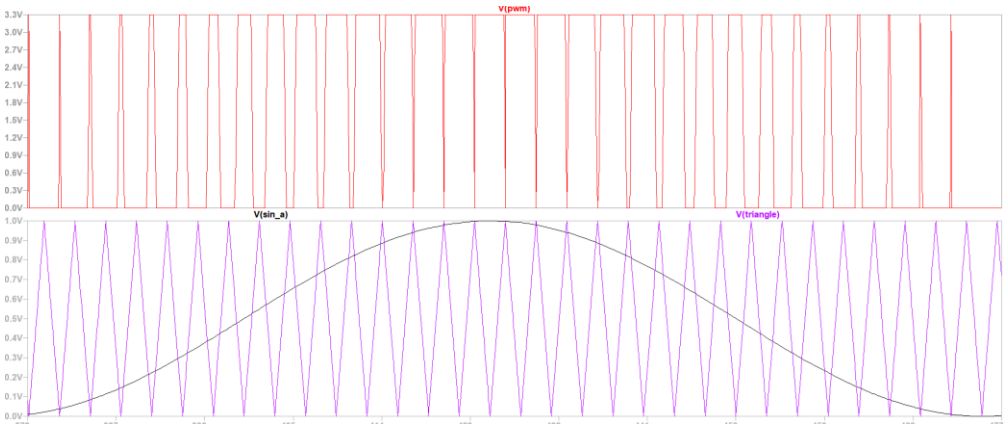

Figure 2.4.1.5-LTspice simulation, Triangle and sine comparation to generate PWM fsw = 320kHz, fel = 10kHz

2) Bridge inverter control

The PWM signal, through the gate drive control is sent to the IGBT gate, on the same leg the two PWM signals must be one the opposite of the other in order to avoid the fault short-leg condition.

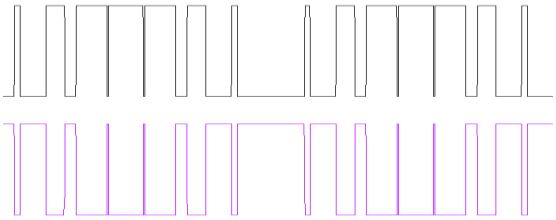

Figure 2.4.1.6-LTspice simulation, two inverted PWM

The three PWM signals (without consider the inverted one) are in phase of shift of $1/3\ \pi$ and they drive the Gate of the IGBTs, the result is a sequence as shown below (eache IGBT is called Sx):

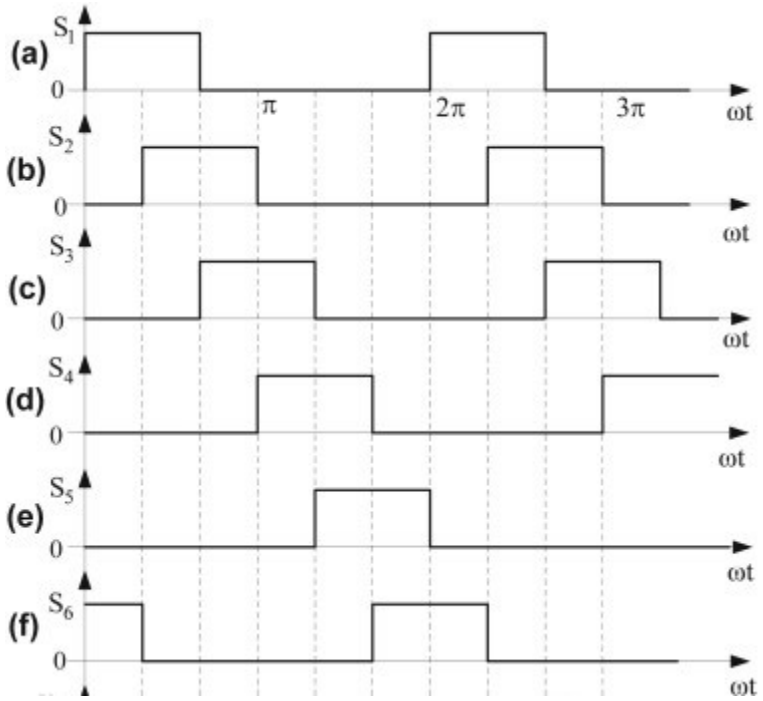

Figure 2.4.1.7-PWM to the six gates of the Bridge

At each high level corresponds a closed switch, whereas at each low voltage levele corresponds an open switch. Respectively, S1, S2 and S3 are never in the same state of S4, S5, S6 (short leg removed).

3. Electrical cabinet design

The primary step to realize an electrical cabinet has regarded the recovery and restoration of an old electrical cabinet to develop a structure able to accommodate the inverter and the relative components for controlling the asynchronous motor.

The disused electrical cabinet, figure 3.1, was destinated to the control of a DC motor, most of the components have been removed and tested to analyze which of them could be reusable for the new configuration.

Figure 3.1- Eletrcical panel to be dismantled

The recovered components have been:

- General contactor, model LC1-D66 Telemecanique
- Rotary switch
- DIN guide
- Relays
- Feed-through modular terminal block

The new design is based on the wiring diagram provided by the hardware manual of the inverter, figure 3.2. Firstly, the usefulness of the contactor MMCB and MCB have been analyzed, both are automatic contactors, the MCCB, “miniature case circuit breaker” and the MCB “miniature circuit breaker”, with thermal and thermo-magnetic protection, with the only difference for the maximum current rate and for the tripping current. The recomanded circuit breaker MCCB is avoidable, because the actual electrical sockets already include a safety system against current overloads, therefore it has been considered only the installation of the general contactor MC. The expected AC reactor and the noise filter were regarded negligible for F0 model inverter, as described in the manual, their inclusion could be necessary only in case of electromagnetic disturbances.

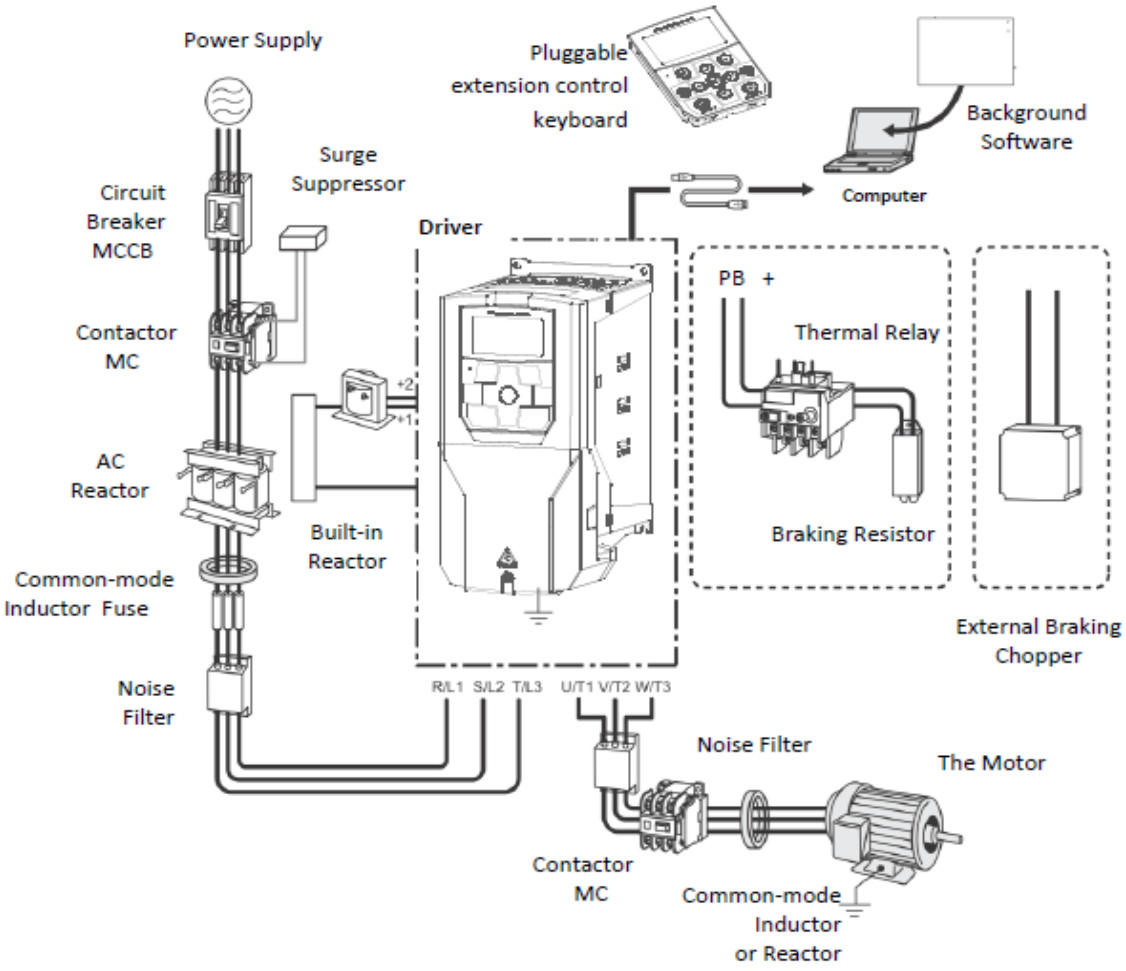

Figure 3.2- Standard connection diagram of the Drive and its periphey

Other components in the following list are added in order to provide the correct functioning of the entire structure, in particular the 400V/12-24V transformer for the power supply of the contactor coil.

In in conclusion, the final list of internal components of the electrical cabinet is the following:

1. General Contactor- LC1-D25 10
2. Transformer 400V\12-24V
3. Rotary switch
4. Inverter ES350-F0-4K0G-3B
5. Relè
6. Din Guide
7. Earthing busbar

8. Feed-through modular terminal block.

The front external part of the panel is dedicated to the start/stop buttons, the emergengy button, a three positions commutator interruptor (0-1-2) that is able to exclude the inverter and to deviate the power supply directly to the motor, in order to permit the managing of the motor also without the inverter, on the back of the panel have been drill holes for the three-phase supply and motor connections.

1) Start button, Tiass/AC-660V-10 model, it is a switch normally opened composed by two poles and a push button able to support up to 10 A and guarantee an insulation of 660 V.
2) Stop button, Tiass/AC-660V-10 model, it is a switch normally closed composed by two poles a push button able to support up to 10 A and guarantee an insulation of 660 V.
3) Emergency button, ZB2-BE102C model, it is a switch normally closed composed by two poles, a push button able to support up to 10 A and guarantee an insulation of 415V.
4) Commutator Switch, SZW26-20/D303.3 model, it provides 3 positions, 3 phases and 12 terminals, it is able to support up to 20 A and guarantee an insultion of 660 V.

3.1. Electrical connections

This chapter illustrates all the plans relating to the electrical connections, which are essential for the overall operation of the panel.

3.1.1 Scheme 1- Command connections

The scheme 1, figure 3.1.1.1, represents the connections between the general contactor, the power supply and the control panel, that includes start, stop and emergency buttons.

The general contactor is the device necessary to guarantee the correct and safety insulation during the open circuit state and, it must support a certain amount of the electrical current that enters a circuit without any fault. The control of the general contactor is managed by the on/off switches on the electrical panel and the emergency button that act on its coil. Our model has also auxiliar contacts normally open (NO), pin13 and 14.

The contactor, denominated as KML, has three inputs, named with the odd numbers 1-3-5, and three outputs on 2-4-6, in the same way the auxiliary contacts placed on it are distinguished in input 13, output 14. The total four breakers can be nominated as K1(1-2), K2 (3-4), K3(5-6), K4 (13-14). The KML coil is characterized by pin A1-A2, and it requires a power supply of 24VAC, supplied by the transformer 400/12-24 V as can be seen in the scheme 2, figure 3.1.2.1.

The start button is a NO contact, while the stop and the emergency are a NC.

The general contactor exictasion occurs when the start button is pushed, the L1 to L3 is closed, thus the coil is powered, in order to maintain the exictation it is useful a parallel with the auxiliar pin 13-14 throught the normally close buttons Stop and Emergenc, realizing the self-holding circuit.

Then, the emergency button was inserted according to the retaining circuit principle, for which if the emergency button is pressed, the contactor contacts open, the contactor is de-excitated. Once it is disarmed, it will be necessary to press the start button again to restart the power supply.

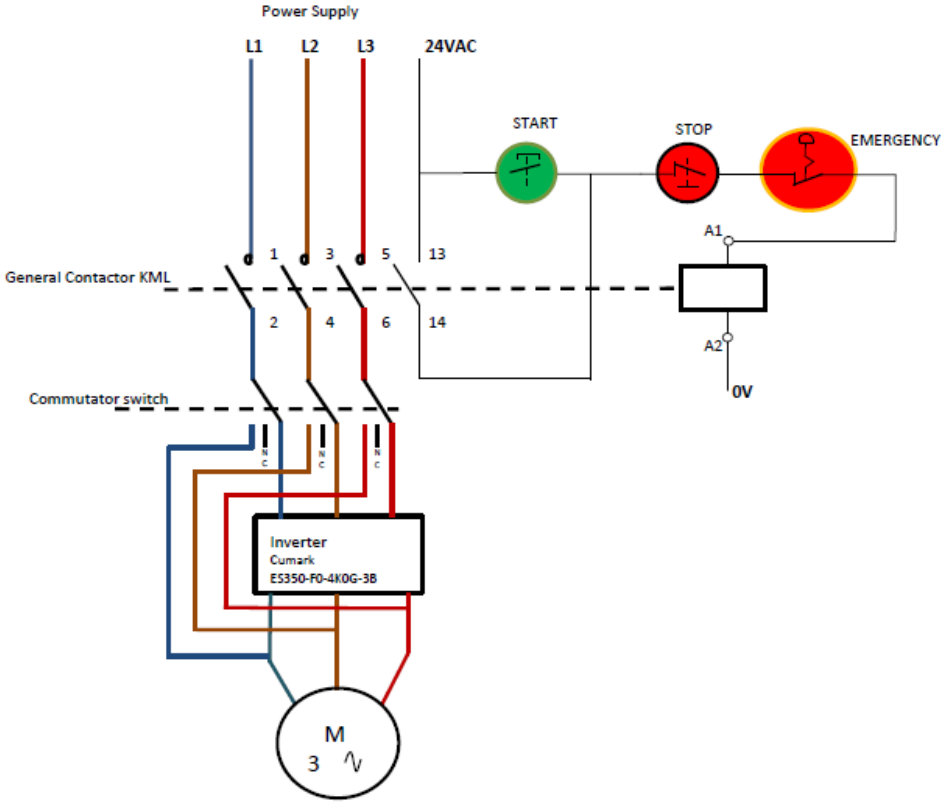

Figure 3.1.1.1 Scheme1-Command connections between inverter, motor, and power supply

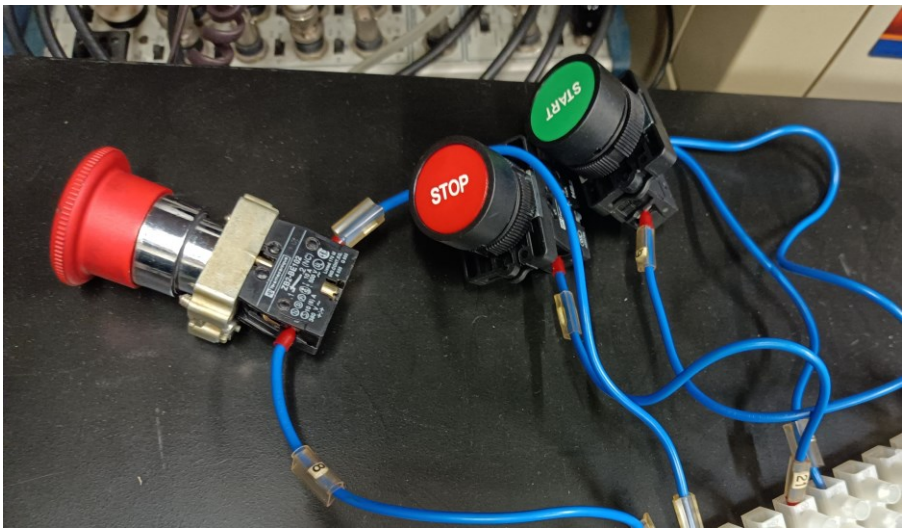

Figure 3.1.1.2-Connections of Emergency, Start and Stop buttons.

It is important to report a scheme to describe the different routes of the power supply according to the commutator switch selection. The model of the commutator switch is SZW26-20/D303.3-660V-20A, it provides 3 positions, 3 phases and 12 terminals, that are connected according to the figure 3.1.1.3.

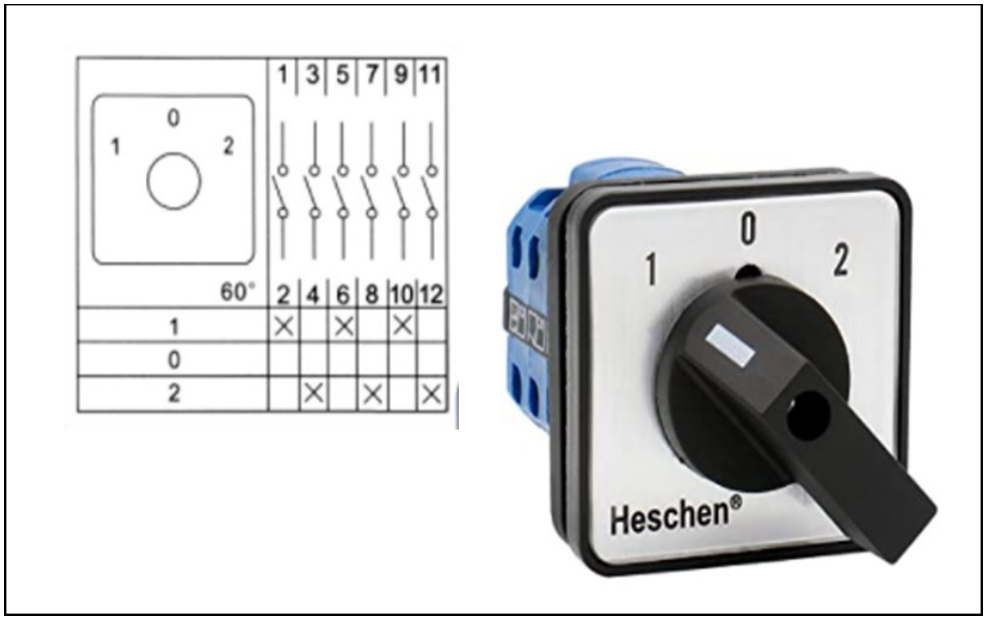

Figure 3.1.1.3-Commutator switch terminal connection.

In this work the following positions describes two specific voltage routes:

- position 1 was selected to describe the route: Contactor KML-Inverter-Motor.
- position 2 was related to the route: Contactor KML-Motor.

3.1.2 Scheme 2- Power Supply circuit for KML coil and panel fans

The second design has been the purpose to realize a power supply circuit for the general contactor coil. The coil, for our model of contactor, needs a 24V power supply, at this aim it has been used a transformer (TF1) with the primary supply equal to 400VAC, and two secondary windings at 12VAC and 24VAC.

The scheme 2, in figure 3.1.2.1, reports the appropriate connections for the secondary windings to extract the 24VAC power supply for the coil, more precisely the pin A1 was connected to S2(24VAC), while the pin A2 was connected to E1(0V).

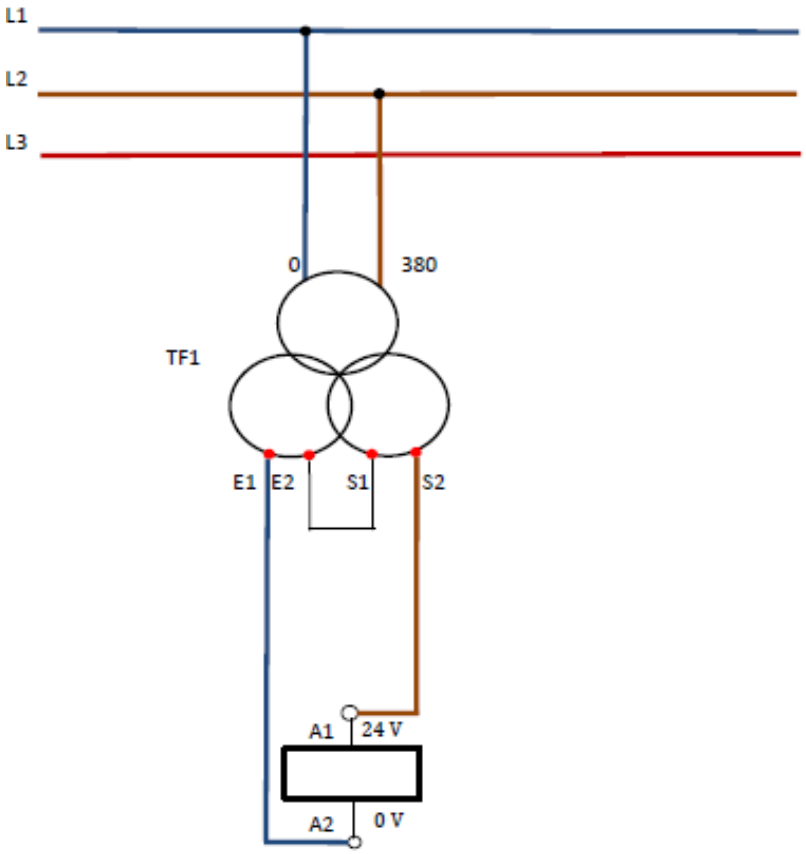

Figure 3.1.2.1-Scheme 2-Power supply for contactor coil by means of transformer 1

For what concern the cooling circuit for the cabinet, from the old electrical panel two fans have been recycled and inserted in the new panel configuration. They expect a 220V power supply, thus they were connected directly to three-phase line.

3.1.3 Final implementation

The final implementation of the electrical cabinet was showed in figure 3.1.3.1, more specifically it presents the electrical installation of the internal components according to the above explained schemes.

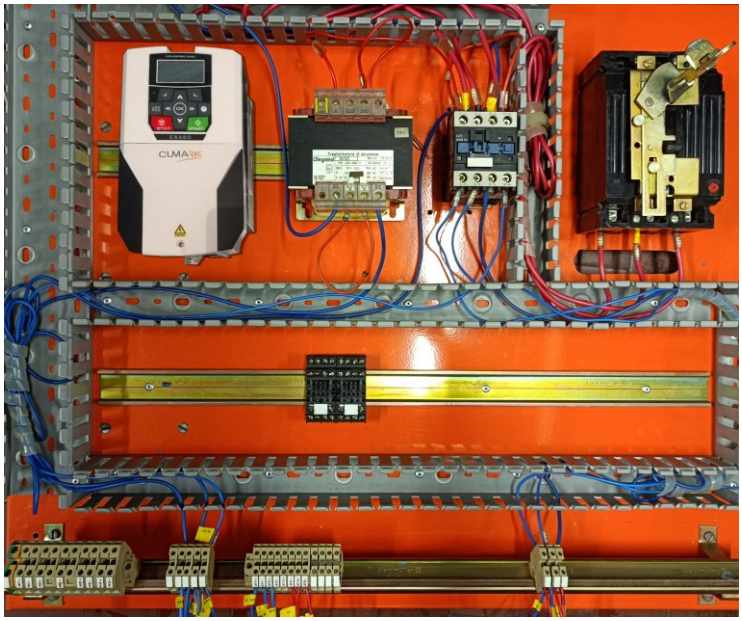

Figure 3.1.3.1-Internal electrical connections of electrical panel

4. Remote Control

The main aim of this work has been the realization of a graphical interface to control in remote mode the asynchronous motor, by means of the inverter. At this purpose it has been necessary to define the modality of the remote control.

Firstly, the configuration of Inverter parameters is fundamental to manage an external control. The analyzed solutions to implement this type of control can be summarized in two ways:

- Firstly, configure the inverter parameters by the LCD and the front panel, and drive the inverter by means of the connections between digital inputs of Inverter and Arduino pin, and realize the front-end circuit to interface the Arduino signals (5V) with the control logic pin of the inverter (24V). Design a proper firmware for Arduino to drive the GPIO and read the inverter response. Finally, to realize a graphic interface to manage via PC the control.
- Through Modbus protocol, the parameters configuration is necessary to select the external control function source as fieldbus, then need to generate a code which to communicate the master device (PC) with the slave (Inverter) through the connection of the Inverter pin (A+/485+, A-/485-) dedicated to the serial communication. This connection requires the using of a USB connector RS485. In conclusion, the serial communication is managed by a custom user interface that permits to drive in remote mode the Inverter.

The second solution has been adopted. The reasons are multiple: in this way it is avoided to add further wiring in the electrical cabinet, thanks to remote control it is possible not to be dependent on the front panel configuration, and finally everything is customizable with remote method from the configuration of the inverter to end with the post measurements analysis. Last but not least, the Modbus protocol allows

for a more robust interface against noise, and it allows the access to all the parameter registers.

4.1. Modbus Protocol

Modbus is an application layer messaging protocol, which provides client/server communication between different devices on different types of networks. [5]

There existis two versions of Modbus protocol:

1. On serial port (RS485 or RS/322). The Serial port protocol presents two variant of Modbus protocol:
    - Modbus RTU, the data representation is compact of hexadecimal type; It exploites the CRC (cyclic redundancy check) for the error check. .[11] [12]
    - Modbus ASCII, the codification is in ASCII, it results easly readable and redundant. It exploites the LRC (Longitudinal redundancy check).
2. On Ethernet. This version provides the following type of Mobus protocol:
    - Modbus/TCP, it is similar to RTU version, but it transmits protocol packets into TCP / IP data format via ethernet connection.[11] [12]

Modbus is a type of communication with a request/reply between master and slave, the master manage the line trasmission by sending a specific message whose format is defined by the protocol. The reply format has a certain structure according to the type of the request.

4.1.1. General Modbus frame

"The Modbus communication occurs by means of a frame defined as application data unit (ADU), figure 4.1.1, that includes a simple protocol data unit (PDU) independent of the underlying communication layers." [5]

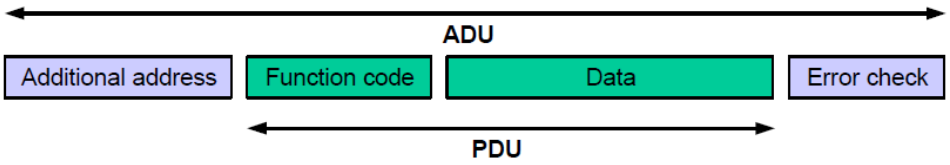

Figure 4.1.1.1-General modbus frame

The MODBUS application protocol establishes the format of a request initiated by a client; the request is divided in four fields:

- Additional address or Slave address
- Function code
- Data
- Error Check

The data field of messages sent from a client to server devices contains additional information that the server uses to take the action defined by the function code. This can include items like discrete and register addresses, the quantity of items to be handled, and the count of actual data bytes in the field.

Additional Address

It represents the slave address; it is useful in case of multiple number of slaves. In this specific case, when the communication is between the master (PC) and the slave (Inverter), it is irrelevant because there is only one slave identified with the address 1. Furthermore, it is important to specify that 0 address is reserved for the broadcast.

Function Code and Data (PDU)

The function code indicates to the server what kind of action to perform.

“The function code field of a MODBUS data unit is coded in one byte. Valid codes are in the range of (1-255) decimal (the range 128 – 255 is reserved and used for exception responses). When a message is sent from a Client to a Server device the function code field tells the server what kind of action to perform. Function code "0" is not valid”.[5]. Some function codes are defined in associaton with sub-function codes to define multiple actions.

The most relevant function codes are summarized in the following table, and they are distinguished for the type of action to be performed (read, write or diagnostics).

Type of Action	**Description**	**Function codes**		
		Code	Sub-code	Hexadecimal
READ	Read coils	01		01
	Read discrete inputs	02		02
	Read input registers	04		04
	Read holding register	03		04
WRITE	Write single coils	05		05
	Write Multiple Coils	15		0F
	Write Single Holding Register	06		06
	Write Multiple Holding Registers	16		10
DIAGNOSTICS	Diagnostic	08	00-18,20	08

Table:4.1.1.1- List of most used function code with Modbus protocol

In the following chapter the functions code exploited in the remote-control design were explained in detail.

03(0x03) Read Holding Registers

This function code is used to read the contents of a contiguous block of holding registers in a remote device. “The Request PDU specifies the starting register address and the number of registers to be read. In the PDU register numbering starts from address 0 while the number of registers starts from 1, thus registers numbered 1-16 are addressed as 0-15.” [5]

0x03 Request frame		
Slave address	1 Byte	
Function code	1 Byte	0x03
Register start address	2 Bytes	0x0000 to 0xFFFF
Number of registers	N x 2 Bytes	0x0001 to 0xFFFF

0x03 Reply frame		
Slave address	1 Byte	
Function code	1 Byte	0x03
Byte count	1 Byte	2*N
Register data 1	2 Bytes	0x0000 to 0xFFFF
Register data 2	2 Bytes	0x0000 to 0xFFFF

Table 4.1.1.2 -Request and reply frame for 0x03

“The register data in the reply message are packed as two bytes per register, and they are addressed respect the reading.” [5]

06(0x06) Write Single Holding Register

This function code is used to write a single holding register in a remote device. The Request PDU specifies the address of the register and the data to be written. The normal response reflects the request, returned after the rgister have been written.[5]

0x06 Request frame		
Slave address	1 Byte	
Function code	1 Byte	0x06
Register address	2 Bytes	0x0000 to 0xFFFF
Register value	2 Bytes	0x0000 to 0xFFFF

0x06 Reply frame		
Slave address	1 Byte	
Function code	1 Byte	0x06
Register address	2 Bytes	0x0000 to 0xFFFF
Register value	2 Bytes	0x0000 to 0xFFFF

Table 4.1.1.3-Request and raframe for 0x06

08(0x08) Diagnostics

“MODBUS function code 08 is aimed at checking the communication system between a master device and a server, or for verifying the precense of server internal error conditions”.[5] “The function code 08 exploits some sub-function code to characterized specific task to be performed. A server device can, for example, be forced into ‘Listen Only Mode’, with subfunction code 0x04, in which it will monitor the messages on the communications system but not respond to them”.[5]

The request frame format follows the scheme in figure 4.1.1.3. while the reply frame is different according to the selected sub-function.

0x08 Request frame		
Slave address	1 Byte	
Function code	1 Byte	0x08
Sub-function	2 Bytes	0x0000 to 0xFFFF
Data	N x 2 Bytes	0x0000 to 0xFFFF

Table 4.1.1.4-Request and raframe for 0x06

16(0x10) Write Multiple Holding Registers

Function code 10 provides the writing continuity the current value of the N parameter. The request and reply frame are visible in table 4.1.5. The number of bytes is equal to 2 times the number of registers. This PDU is similar to the 0x06 request, but it permits to write multiple registers, so it is necessary to specify the register data for each holding registers. The response frame returns the unction code, starting address, and quantity of written registers.[5]

0x10 Request frame		
Slave address	1 Byte	
Function code	1 Byte	0x10
Register start address	2 Bytes	0x0000 to 0xFFFF
Number of registers	2 Bytes	0x0001 to 0xFFFF
Number of bytes	1 Byte	2*N
Register data 1	2 Bytes	0x0000 to 0xFFFF
…		

0x10 Reply frame		
Slave address	1 Byte	
Function code	1 Byte	0x10
Register start address	2 Bytes	0x0000 to 0xFFFF
Number of registers	2 Bytes	0x0001 to 0xFFFF

Table 4.1.1.5-Request and reply frame for 0x10

CRC (Cyclic redundancy check)

The ADU expects 2 bytes of CRC (cyclic redundandacy check) code, it is used to ensure message data integrity and for checking framing errors. The CRC is computed by using a determinated alghoritm called CRC16, by using every byte in the message frame except for the last two bytes dedicated to the CRC itself. The computation alghorithm was described by means of a flowchart in figure 4.1.1.2.

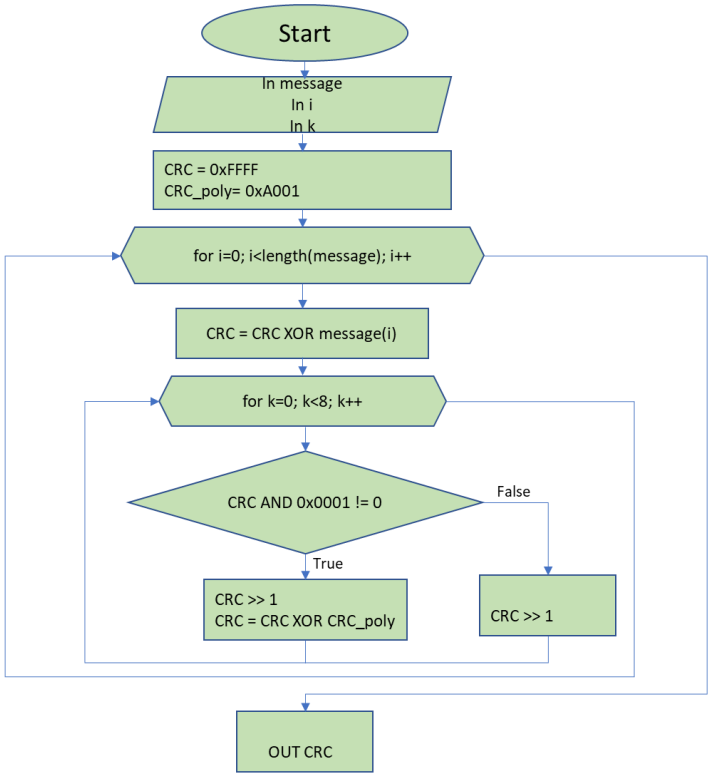

Figura 4.1.1.2-Flow chart for the CRC16 computation

The CRC computation is based on the division between binary polynomials. Firstly, it is necessary to define the initial value of the divider, *0xFFFF*, in the hexadecimal system, and the initial polynomial value CRC_poly = 0xA001. After that, the XOR operation is repeated *i* times, with *i* equal to the number of bits of the message for which the CRC is required. For each i_{th}-iteration, the logic AND among the CRC and the hexadecimal 0x0001 is carried out. If the result is equal to 0, only a right shift is performed on the CRC, otherwise the CRC is shifted to the right and then a new CRC is computed as the XOR between the previous one and the CRC_poly.

4.1.2. Modbus protocol for the Inverter

The considered inverter for this thesis work allows the serial communication through modbus protocol. The serial connection follows the scheme in figure 4.1.2.1 that represents the wiring diagram of the inverter model F0, in particular the dedicated pins for the serial RS485 are A+ and A-.

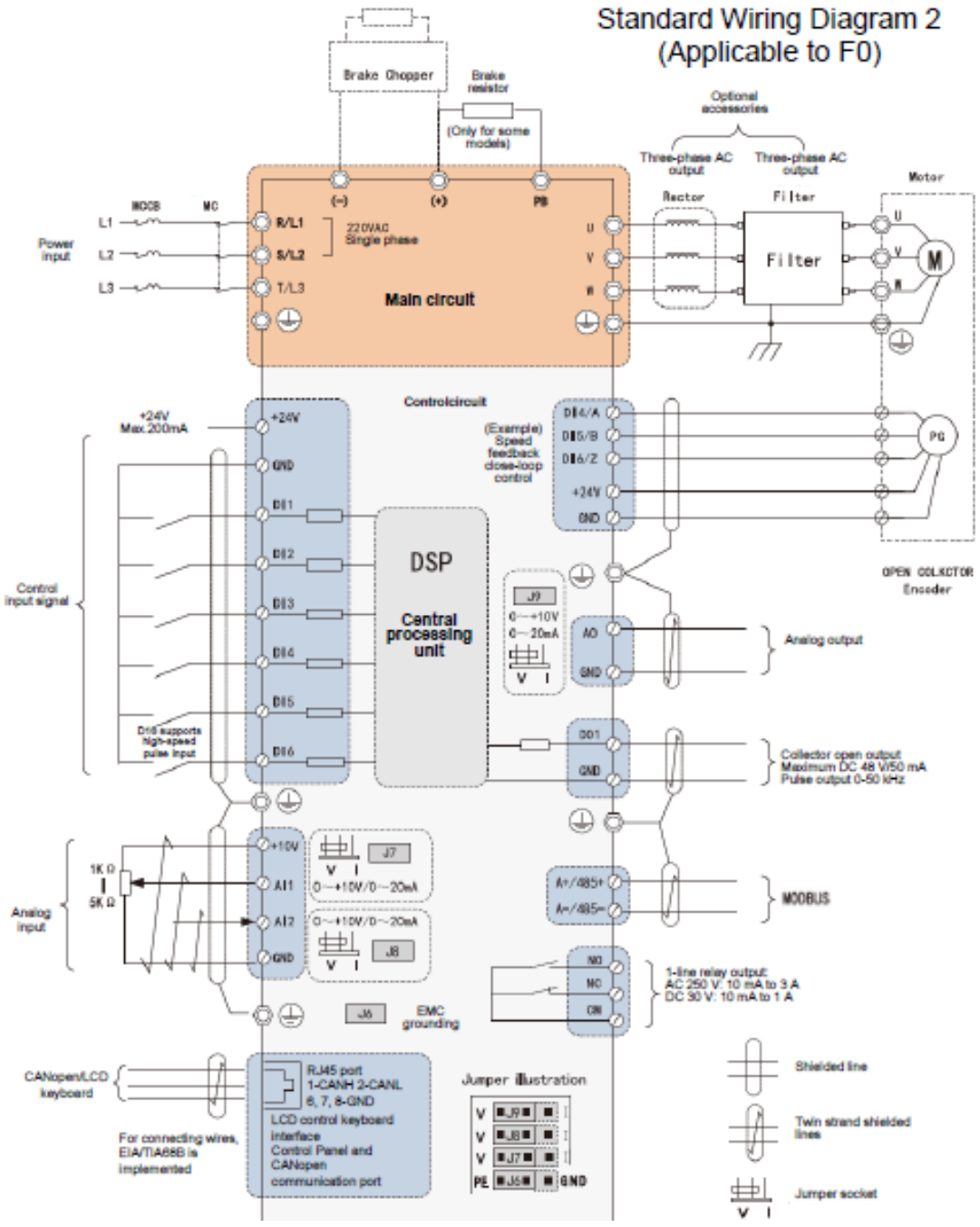

Figure 4.1.2.1- Wiring diagram of Cumark Inverter ES350 model F0

According to the firmware manual, the inverter contains 63 groups of parameters, based on this consideration it is possible to derive the specific register address for each group, table 4.1.2.1.

GROUP	INDEX	Address	
		Hexadecimal	Deciamal system
00Communication data	01-30Data set	0001-001E	0001-0030
01 Parameter group01	00-255Parameter 01.00-01.255	0100-01FF	256-511
02Parameter group02	00-255Parameter 02.00-02.255	0200-02FF	512-767
...	...		
63 Parameter group63	00-255Parameter 63.00-63.255	3F00-3FFF	16128-16383

Table 4.1.2.1-Address for each inverter parameter group

The group 00 of communication data is allocated to the data set, listed in table 4.1.2.2, they are destinated for monitoring specific paramaters.

DATA SET	
Address	Name
0001	Fieldbus control word (corresponding to monitoring parameter address 6.05)
0002	Field bus given 1(corresponding to monitoring parameter address02.15)
0003	Field bus given 2(corresponding tomonitoring parameter address02.16)
0004	Field bus status word
0005	Field bus actual value1
0006	Field bus actual value2
0007-0018	Field bus module input1-12(parameter50.05-50.16)
0019-0030	Field bus module output1-12(parameter50.17-50.28)

Table 4.1.2.2- Data set address and description

Another fundamental aspect to perform the remote control through the serial communication RS485 was the using of a specific interface circuit in order to convert the USB signal in Serial RS485 one. For the system comuunication the chosen is fallen on the commercial and cheap solution DSD TECH pictured in figure 4.1.2.2 with FTDI Chip FT232 on board. Finally, the connection between master (PC) and slave (Inverter) has been realized by following the diagram in the figure 4.1.2.3, where 485+(1) and 485-(2) corresponds respectively to pin A+ and A- on the Inverter.

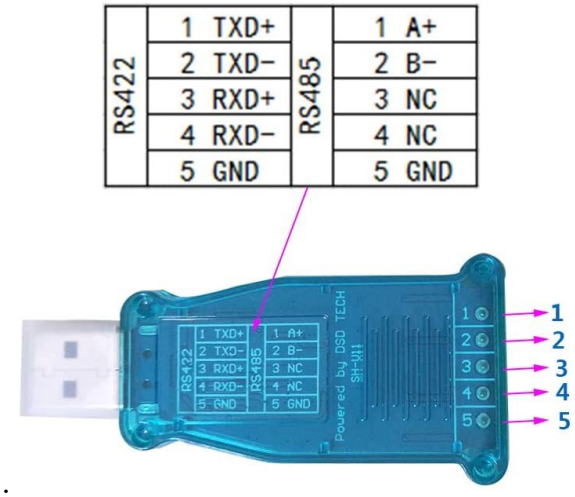

Figure 4.1.2.2- DSD TECH converter RS422 USB/RS485 with Chip FTDI FT232

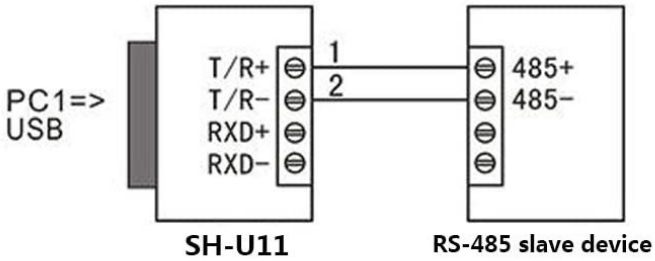

Figura4.1.2.3-USB to RS485 wiring connection

4.2. Inverter Parameters Configuration

As extensively stated in the previous chapters, the inverter allows remote control throught modbus communication, for which, for the first time, it is necessary to previously configure certain parameters; all the changes have been listed, in execution order, in the following table.

RELATED PARAMTERS

Parameter 63-MOTOR SET UP Insertion of plate data of the associated motor and driving control mode: V / F open loop.
Paramteter 10- EXTERNAL CONTROL 10.00 Ext1 control function: Fiedlbus; in order to define the type of external control.
Paramter 11 -Start stop settings 11.00 Stop mode: RAMP; to obtain a deceleration stop of the motor. 11.02-11.03 Ext1 cntrlmode: speed; to perform a speed regulation.
Parameter 20-LIMITS 20.00 Maximum speed:4000 rpm; to define the maximum allowed speed. 20.01 Minimum speed -4000 rpm; to define the minimum allowed speed
PARAMETRO 21-SPEED REFERENCE 21.00 Speed ref1 src: Fieldbus ref1, to set that the speed reference was controlled by fieldbus(modbus) communication.

21.02 Speed ref function: ref1; the signal selected by the parameter 21.00 Speed ref1src (Signal source for speed given 1) is used as the speed given value1.

PARAMETRO 22- RAMP GENERATOR

Definition of ramp signal for having a deceleration in stop mode and acceleration in start mode.

22.00 ACC TIME1=10s

22.01 DEC TIME1=10s

22.04 EM stop time =1s

PARAMETRO 51- MODBUS

51.00 Modbus enable: enable

51.01 node address:1

51.02 Baudrate:9600

51.03 Format:0, 8, N, 1 (8-bit data, No verification,1stop bit)

51.04 Master mode: disable; the inverter is the slave

51.05 Reg data: P03.04 (Ramp and forming speed given)

51.06 Reg add:2

51.07 Comm cycle:100ms

Table 4.2.1-Parameters configuration

Once the parameters were set, it was possible to implement the graphic interface relating to the remote control. The parameters setting has been performed only one time and the configuration was stored in inverter memory.

5. Graphic Interface

The following chapter describes the graphic interface design for the speed control of an asynchronous motor. The development environment used was *MATLAB* and in particular *MATLAB App Designer*, used for creating the *GUI* (graphic user interface). It contains a large library of components, such as buttons, checkboxes, trees and drop-down lists, which allow to reproduce the operations of the instrument panels.[15]

The interaction among the components is allowed by customized *callback functions*, that are executed when the user interfaces with the application.

The graphic interface design has been developed in three phases:

- Serial Port Settings
- Control panel
- Monitoring of parameters

Before creating the command interface, *App Designer* generates a *classdef,* in which it defines the *app properties* where all the components present in the application are declared, then it is possible to add a *methods* section or a furthermore *properties* section, for defining respectively all public or private function and all the the variables of public use (global variable), to be able to use them in different callback functions.

All the global variables, defined in *properties(access=public)* section, can be used within all the functions of the application by writing *app.nameofvariable.*

In the following sub-chapters, the various phases that led to the creation of the GUI for motor control are described.

5.1. Serial Port Settings

Firstly, it was necessary to create the port on which the two devices start the communication, for this purpose the function *serialport* on *Matlab* has been used, it is a type of function dedicated to the creation of the serial port. To make customize the serial settings, the user was given the possibility to select the various features of the serial port from the interface, in particular the COM through which the transmission occurs, BaudRate, DataBits, Parity and StopBits. In this application the settings for serial communication must match those set in the parameter configuration.

The code related to the creation, open and close of the serial port has been shown in the following table. An error management, *Open Port Error*, has been added in the *OpenportButtonPushed* function, in the event that the opening is unsuccessfulI, furthermore the error produces to the user the advice to check for possible errors that prevent serial communication, figure 5.1.2., with this solution even if an user error occurs, the code keeps run, improving the robustness of the interface.

The serial port is closed by means of the *CloseportButton*, associated to a function in which the value "*serial*" is set to zero.

```
% Button pushed function: OpenportButton
 function OpenportButtonPushed(app, event)
     try

         app.serial=serialport(app.SelectedPortDropDown.Value,..
         str2double(app.BaudRateDropDown.Value)'DataBits',…
         str2double(app.DataBitDropDown.Value),'StopBits',…
         str2double(app.StopBitDropDown.Value),'Parity',..
         app.ParityDropDown.Value);

         app.TextArea.Value='Connected';
         app.TextArea.BackgroundColor='g';

     catch
         errordlg('Please check cable connection and the…
         communication parameter andretry','Open Port Error');
     end
```

```
  end

% Button pushed function: CloseportButton
  function CloseportButtonPushed(app, event)

          app.serial=0;
          app.TextArea.Value='Disconnected';
          app.TextArea.BackgroundColor='w';

  end
```

Table 5.1.1-Code for the creation of serial port and relative callback functions

The final implementation of the serial port settings in the GUI is shown in figure 5.1.1, where the TextArea *Serial Connection* changes according to the successful of the port connection, if the opening was successful, it shows *Connected*, otherwhile compares the *Disconneted* message.

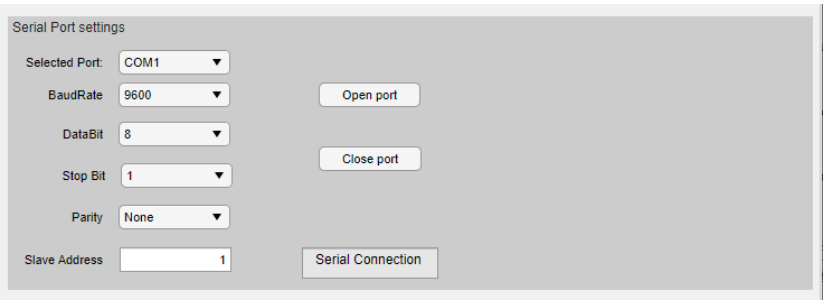

Figure 5.1.1-Serial port settings panel implementation in the GUI of App Designer

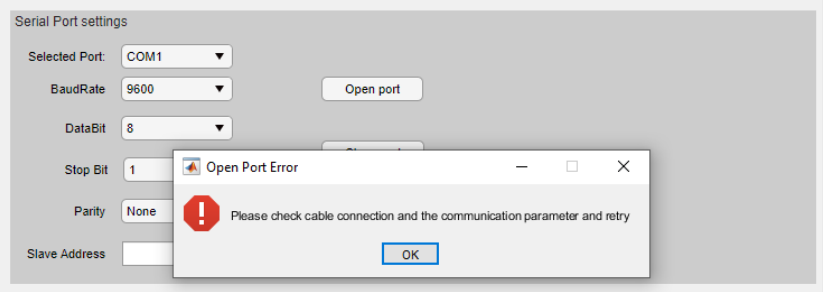

Figure 5.1.2-Open Port Error in the serial port settings panel

5.2. Control panel

The second phase of this project has seen the implementation of the open-loop speed control for the asynchronous machine by means of the Modbus protocol.

The primary step has been the desired speed setting, once the choices speed denominated as "*speed_ref*" has been set, through the *SetValueButon* a write frame is sent to the inverter through which it is possible to modify the speed value.

The function code for the writing of a holding register is 0x06, while the address for allocating the reference speed is 0002, from data set table 4.1.2.2. Starting from the frame *"01 06 00 02"*, the next two bytes will be destinated to the value of *speed_ref*. For the correctness of the frame the *speed_ref* value was converted from decimal to hexadecimal.

The last two byte of the frame are occupied by CRC code, as it has been explained in chapter 4.1.1, CRC has been computed by means of *Modbus_CRC16* [21] function described in table 5.2.2, it follows the alghoritm elaborated in figure 4.1.1.2.

To each writing it has been chosen to associate the corresponding reading, in such way as to flush the residual answer bit on the serial port and avoiding wrong reading.

```
%Button pushed function: SetvalueButton
 function SetvalueButtonPushed(app, event)
     app.STARTButton.Enable = 'on';

     %% WRITE FOR DESIRED SPEED SETTING
     speed_ref = dec2hex(app.speedvalueEditField.Value,4
);
     frame = ['01'; '06'; '00'; '02'];
     frame(5,:) = [speed_ref(1) speed_ref(2)];
     frame(6,:) = [speed_ref(3) speed_ref(4)];
     % CRC coomputation
     message_crc=uint16(hex2dec(frame));
     crc_code=Modbus_CRC16(message_crc);
     CRC=dec2hex(crc_code,4);
     frame(7,:) = [CRC(3) CRC(4)];
     frame(8,:) = [CRC(1) CRC(2)];
     request = uint8(hex2dec(frame));
     write(app.serial, request,"uint8"); %mod speed in dec
     data=read(app.serial,8,"uint8");    %reply frame

  end
```

Table 5.2.1- Code part relating to the speed setting

```
function [ CRC ] = Modbus_CRC16( message )
%message is the frame to Generate the CRC,it is 8 bit type
CRC = uint16(hex2dec('FFFF')); %0xFFFF, initial value
CRC_poly = uint16(hex2dec('A001')); %the Polynomial 0xA001
for i = 1:length(message)
    CRC = bitxor(CRC,uint16(message(i)));
    for k = 1:8
        if(bitand(CRC,hex2dec('0001')))
            CRC = bitsrl(CRC,1);
            CRC = bitxor(CRC, CRC_poly);
        else
            CRC = bitsrl(CRC,1);
        end
    end

end
```

Table 5.2.2-Code relating to the CRC computation function

Once the desired speed has been set, it is possible to use the start and stop button to manage the running of the motor.

Both the operations have been implemented through the function code 0x06, for writing the register 0001, this register is associated to the field bus control word, that monitoring parameter *06.05-Fieldbus Control word*. In figure 5.2.1 it has been shown the structure of parameter 06.05 to better understand what kind of writing to perform according to the start and stop operations.

Field bus control word

Position	Name	Information
0	Stop	1=Drive stop.
		0=Maintain current status.
1	Start	1=Drive start.
		0= Maintain current status.
2	StopMode OFF2	1 =Mandatory for emergency shutdown mode.
3	StopMode OFF3	1 =Mandatory for coast stop mode.
4	Local ctrl	1 =Request for local control.
5	StopMode ramp	1 =Mandatory for deceleration stop mode.
6	StopMode coast	1 =Mandatory for coast stop mode.
7	Run enable	1 = Run enable. 0 = Run inhibit.
8	Reset	0->1 Reset drive fault.
9	Jog1	1 = Jog 1 start.
10	Jog2	1 = Jog 2 start.
11	Remote	1 = Request for remote control.
12	Ramp in 0	1 =Force the input of the given ramp generator as 0.
13	Ramp hold	1 = Force the output of the given ramp generator to remain constant.
14	Ramp out 0	1 = Force the output of the given ramp generator as 0.
15	Ext2 sel	1 = Select external control place2.

Figure 5.2.1-Register structure for parameter 06.05(fieldbus control word)

According to the above figure, the control word for having the motor in start mode is: '0000100010000010', where the least significant bit corresponds to position 0, to generate this request with a type of format suitable to the modbus protocol, it has

been converterd from binary to hexadecimal. For the remote control the bit in position 7 and 11 must be always equal to 1.

BINARY	HEXADECIMAL
0000100010000010	0882

The same reasoning was applied to the stop request, for which the control word is '0000100010000001', obviously in this case the bit in position 0 is put to 1; the conversion from the binary value into a hexadecimal value it has been necessary for the same reason above explained.

BINARY	HEXADECIMAL
0000100010000001	0881

Also in this case, the writing operations are followed by the correspondant reading for avoiding the clogging of the serial port.

```
% Button pushed function: STARTButton
function STARTButtonPushed(app, event)

%% WRITE FOR START
 request = uint8(hex2dec(['01'; '06'; '00'; '01'; '08';
'82'; '5F'; 'AB']));%address 0001: monitoring Fieldbus
control word
 write(app.serial, request,"uint8"); %start in dec
 data=read(app.serial,8,"uint8");  %reply
```

Table 5.2.3- Code for the start request

```
% Button pushed function: STOPButton
function STOPButtonPushed(app, event)

 %% WRITE FOR STOP
```

```
 request = uint8(hex2dec(['01'; '06'; '00'; '01'; '08';
'81'; '1F'; 'AA']));
 write(app.serial, request,"uint8"); %stop in dec
 data=read(app.serial,8,"uint8");    %reply

end
```

Table 5.2.4- Code for the stop request

In order to regulate the motor speed as precisely as possible it was thought to add a section to set the acceleration and deceleration time of the *speed_ref* from the interface. They refer to the group parameter of inverter *22.00-Speed given ramp generator*.

Ramp generator is used for speed control, by selectiong appropiartely the acceleration and deceleration time it defines the S-curve time to produce ramp speed given signal and take this as speed input for the speed regulator. The S-Curve approximates the trapezoidal profile generated by the ramp signal.

The drive provides two groups of sets of the acclerations and deceleration time in case of multiple speed. In our case it will be always select the first group described by parameter *22.00-Acceleration time1* and *22.01-Deceleration time1*. [16]

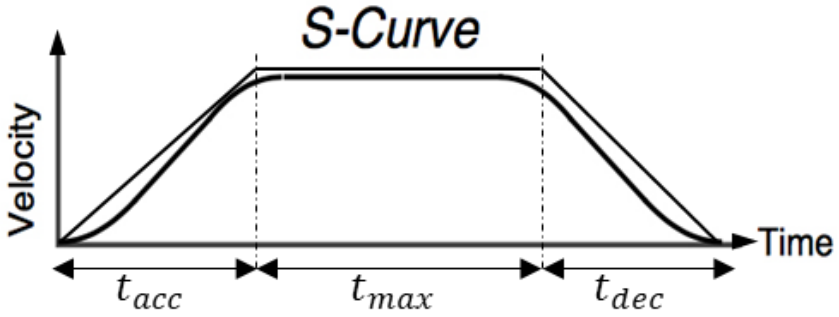

Figure 5.2.2-Ramp generator profile

The GUI provides two "*Spinner*" components for managing the calibration of acceleration and deceleration time, and a button *SetRamptimeButton* for setting up the parameters value. The *SetRamptimeButton* is associated to a callback function

that transmits the suitable message to the Inverter for writing the registers correspondent to parameters *22.00-Acceleration time1* and *22.01-Deceleration time1*.

As for the speed *SetvalueButton*, also the *SetRamptimeButton* corresponds to a writing hold register operation, so the function code to be used is 0x06. Then the register address is necessary to create the frame, in this case the register address for Acceleraation time1 and Deceleration time1 are respectively 22.00 and 22.01, that in the hexadecimal correspondence result to be 16.00 and 16.01, table.5.2.5.

The following two bytes for both operations are dedicated to the value set by means of *AccTimesSpinner* and *DecTimesSpinner*, the choices values in the interface are expressed in the measure unit of second, then for the settings of inverter parameters it is necessary to multiply it for 100, due to the unit measure in the Inverter (0,01s), moreover for a correct conversion from decimal to hexadecimal it is necessary represent the values with 4 elements.

For completing the structure of the frame to be transmitted, it will be necessary the CRC code, computed as always through the *Modbus_CRC16* function.[21]

It is important to evidence that for the *acc_time* and *dec_time* values were defined some limits in the *Spinner* component design, in order to avoid the possibility that the user can insert inappropriate values and can provoke the motor winding damage. For the *Spinner* the limit interval, in seconds, is [0,30].

```
% Button pushed function: SetRamptimeButton
        function SetRamptimeButtonPushed(app, event)

            %WRITE FOR SPEED RAMP GENERATOR
            %ACCELERATION TIME
            acc_time=dec2hex(app.AccTimesSpinner.Value*100,4)
            frame = ['01'; '06'; '16'; '00'];%2200:address for
monitoring acc time 1
                 frame(5,:) = [acc_time(1) acc_time(2)];
                 frame(6,:) = [acc_time(3) acc_time(4)];
                 % CRC coomputation
                 message_crc=uint16(hex2dec(frame));
                 crc_code=Modbus_CRC16(message_crc);
                 CRC=dec2hex(crc_code,4);
                 frame(7,:) = [CRC(3) CRC(4)];
                 frame(8,:) = [CRC(1) CRC(2)];
```

```
                request = uint8(hex2dec(frame));
                write(app.serial, request,"uint8"); %modify
acceleration time in dec
                data=read(app.serial,8,"uint8");  %reply frame
            %DECELERATION TIME
            dec_time=dec2hex(app.DecTimesSpinner.Value*100,4)
            frame = ['01'; '06'; '16'; '01'];% 2201 address for
monitoring dec time 1
                frame(5,:) = [dec_time(1) dec_time(2)];
                frame(6,:) = [dec_time(3) dec_time(4)];
                % CRC coomputation
                message_crc=uint16(hex2dec(frame));
                crc_code=Modbus_CRC16(message_crc);
                CRC=dec2hex(crc_code,4);
                frame(7,:) = [CRC(3) CRC(4)];
                frame(8,:) = [CRC(1) CRC(2)];
                request = uint8(hex2dec(frame));
                write(app.serial, request,"uint8"); %modify
acceleration time in dec
                data=read(app.serial,8,"uint8");  %reply frame

        end
    end
```

Table 5.2.5-Code relative to the SetRampTimeButtonPushed callback function

The last important aspect to be considered in the *Control Panel* configuration is the option of an Emergency button, at this purpose it has been added the section *Emergency* in which two buttons were provided: *EM stop* and *Disarm EM stop.*

The first allows the immediate stop of the motor, according to the configuration of parameter *22.04-EM stop time = 1sec*, the latter deal with the disarm of the emergency stop, both are referred to the parameter *10.13-Emergency stop*, it describes the signal source of the emergency stop command and it is defined as follow:

10.13-Emergency stop	CONSTANT FALSE [0] = Emergency stop mode
	CONSTANT TRUE [1] = Keep the current state

Table 5.2.6- Parameter 10.13-Emergency stop description

Another fundamental configuration to be considered is the emergency stop mode determined by the parameter *10.14-EM stop mode*, that specifies the deceleration procedure, in the event of emergency button is pushed. From the configuration this parameter has ben set to *OFF 3,* thus it considers the emergency stop time previously defined in parameter *22.04 EM stop time.*

10.14-EM stop mode	OFF 1 [0] = Deceleration stop, Deceleration time is the acceleration and deceleration time1.
	OFF 2 [1] = Coast stop
	OFF 3 [2] = Deceleration stop, Deceleration time is the emergency stop time.

Table 5.2.7- Parameter 10.14-EM stop mode description

Once the parameters to be managed were identified, it was possible to write the relative frames to be sent to the inverter for implementing the Emergency stop and Disarm Emergency stop buttons. The tables 5.2.8 and 5.2.9 rapresent the code relating respectively to the *EMstop Button* and *DisarmEMstop Button.*

For safety reasons, in remote control it is necessary to send a standard stop message after disarming the emergency stop.

```
% Button pushed function: EMstopButton
  function EMstopButtonPushed(app, event)

    %0A0Dhex-> 10.13 dec: address of emergency stop parameter
    %0000hex-> CONSTANT FALSE[0]bin, for having an emergency stop

     request1 = uint8(hex2dec(['01'; '06'; '0A'; '0D'; '00'; '00';...
                    '1B'; 'D1']));
     write(app.serial, request1,"uint8"); %Emergency stop
     data=read(app.serial,8,"uint8");  %reply frame
  end
```

Table 5.2.8-Code relating to the *EMstopButton* function callback

```
% Button pushed function: DisarmEMstopButton
  function DisarmEMstopButtonPushed(app, event)
      %0A0Dhex->10.13 dec: address of emergency stop parameter
      %0001hex->CONSTANT TRUE [1] bin, for disarming the emergency

      request = uint8(hex2dec(['01'; '06'; '0A'; '0D'; '00'; '01';...
                'DA'; '11']));
      write(app.serial, request,"uint8"); %disarm Emergency stop
      data=read(app.serial,8,"uint8");  %reply frame

      request = uint8(hex2dec(['01'; '06'; '00'; '01'; '08'; '81';...
              '1F'; 'AA'])); %0881hex->binary
      write(app.serial, request,"uint8"); %stop in dec
      data=read(app.serial,8,"uint8");
   end
```

Table 5.2.9-Code relating to the *DisarmEMstopButton* function callback

The final implementation of the *Control Panel* of the GUI has been shown in figure 5.2.3.

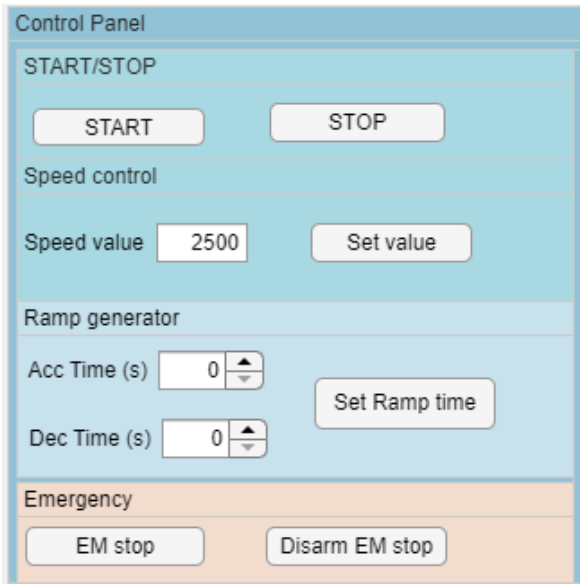

Figure 5.2.3-Control panel implementation in the GUI App Designer

5.3. Monitoring of parameters

The third step of this design regards the monitoring of parameters, it is necessary to monitor all the parameters of interest, according to the start or stop mode of the machine.

Firstly, the timer object has been defined for reading the parameters in real time. Its functioning is explained in figure 5.3.1. The timer starts by means of a *StartFcn callback* or through the *start()* function, the *TimerFcn callback* represents the queue events that the timer manages according with specific properties:

- *The Period*, time with which the events are repeated.
- the *ExectionMode,* that describes the timer function callback scheduling.
- the *TasksToExecute*, that is the duration of the timer counts.

The StopFcn callback is used to stop the timer exeectuion, it can be substituted by the *stop()* command.

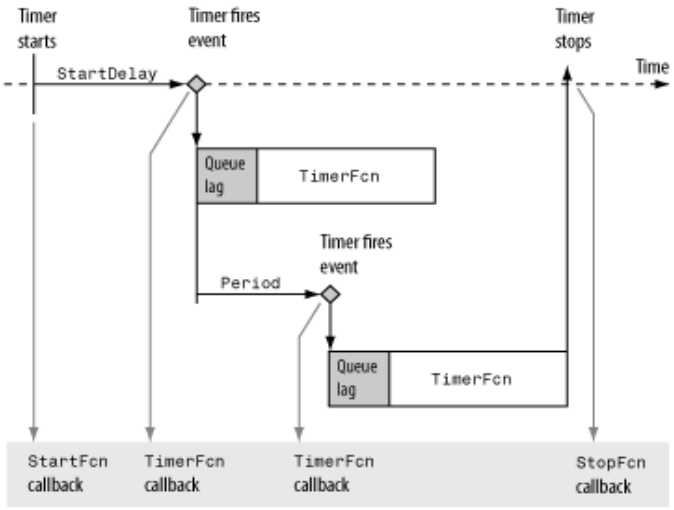

Figure 5.3.1- Timer Object Events and Related Callback Function

Firstly, it has been necessary to define the timer object with function *timer* and all of its properties, in our case it has been selected a *Period* of 2 seconds, the *Executionmode* as 'fixedRate', for which the timer scheduling provides that the timer starts immediately after the timer callback function is added to the *MATLAB* execution queue, and *TasksToExecute* equal to infinity.[17]

It has been chosen to program the timer start every time the start button is pushed and to reset it at every start request, as it visible in figure 5.3.1.

```
% Button pushed function: STARTButton
  function STARTButtonPushed(app, event)

  %% WRITE FOR START
  request = uint8(hex2dec(['01'; '06'; '00'; '01'; '08'; '82'; '5F';
'AB']));%address 0001: monitoring Fieldbus control word
  write(app.serial, request,"uint8"); %start in dec
  data=read(app.serial,8,"uint8");  %reply frame of inverter

% Timer object creation and start

  app.myTimer=0;
  app.myTimer = timer('Period',2,'ExecutionMode', 'fixedRate',…
                …'TasksToExecute', inf);
  app.myTimer.TimerFcn = @(varargin) app.ReadParameters
  start(app.myTimer);

  end
```

Table 5.3.1-Code relating to the creation and definition of the timer object

The *TimerFcn callback* is associated to the function *ReadParameters*, that will be explained in below, it provides all the readings for monitoring specific parameters, or some drive info in order to control the motor behaviour.

Actual values monitoring

The *ReadParameters* function contains the evaluation of the main important parameters to control the motor: speed, current, voltage, frequency and torque.

All the operations of reading are described by function code 0x03 (Read Hold Register) explained in chapter 4.1.1.

The actual values of the inverter are stored in group 01, in particular *01.00-Motor Speed* with unit of measurement 1RPM, *01.01-Output Frequency* with unit of measurement 0.1Hz, *01.03-Motor Current* with unit of Measurement 0.1A, *01.21-Output Voltage* with unit of measurement 0.1V, *01.22-Motor Torque* evaluated in percentage, with unit of measurement 0.1%.

The frame structure to be sent for reading the actual values has been shown in table 5.3.2. The message describes the request to read the register address 0100, that represent the address of group 01 of the inverter, which contains the actual values, and it specifies the number of values to be read in the register, in this case it is equal to 36, in decimal, (0024 in hexadecimal), that represent the total number of values in that specific group.

01	03	01	00	00	24	CRC
Slave address	Function code	Register address		Number of registers to be read		Check redundancy code

Table 5.3.2-Request frame for reading all the parameters of group 01

Any reading is achieved by sending a request through the *write()* function, once the request is sent to the Inverter, the response containing the parameter values is read using the *read ()* function, in which it has been need to specifiy the number of values to read through the reply.

01	03	02	05	DB	00	FD	…
Slave Address	Function code	Number of byte for values	01.00-Motor Speed		01.01-Output Frequency		…
1	2	3	4	5	6	7	…

Table 5.3.3-Reply frame for reading parameters of group 01

The received message has the decimal structure, thus the number of values to be read in the function *read()* has evaluated by considering that the reading of 36 parameters was requested, and each of them is expressed in 2 byte, the number of values dedicated to the actual parameters is 72, by adding 2byte of CRC, 1byte for Slave address, 1byte for function code, 1byte for number of byte for the data, in conclusion the count for the *read()* function is 77.

According to the actual values order into the register it is possible to read certain parameters. The relative code for the reading of output parameters inside the function *ReadParameters* has been shown in table 5.3.3. The argument *varargin* is an input variable in a function definition statement that enables the function to accept any number of input arguments, it has been used for associating without any problem of input, this function to the *TimerFcn* function.

The real time updating of parameters are managed in the GUI by the *EditField* component for current, voltage, torque, frequency, and by means of the *Gauge* component for the speed, as can be seen in the figure 5.3.2 there is the *ExportDataButton* that allows to save in a *'Exportdata.mat'* file the current values, for eventual analysis by the user.

```
methods (Access = private)

function ReadParameters(app,varargin)

%% READ OF ACTUAL PARAMETERS
```

```
%Group 01: Actual Values -> address 0100
%0024 hex-> 36 dec: number of parameters can be read in register
0100


      %01.00 Group of Actual paramters
      frame1_03=['01'; '03'; '01'; '00';'00';'24'];
      message1_crc=uint16(hex2dec(frame1_03));
      crc_code=Modbus_CRC16(message1_crc);
      CRC=dec2hex(crc_code,4);
      frame1_03(7,:) = [CRC(3) CRC(4)];
      frame1_03(8,:) = [CRC(1) CRC(2)];
      request1 = uint8(hex2dec(frame1_03));
      write(app.serial,request1,"uint8");
      data=read(app.serial,77,"uint8");
      x=dec2hex(data);

       %01.00 Motor Speed
       x1=[x(4,:) x(5,:)];
       app.speed_read=hex2dec(x1);
       app.MotorspeedRPMGauge.Value = app.speed_read;

       %01.01 Motor Frequency
       x2=[x(6,:) x(7,:)];
       app.freq_read=hex2dec(x2)*0.1;
       app.MotorFrequencyHzEditField.Value = app.freq_read;

       %01.03 Motor Current
       x3=[x(10,:) x(11,:)];
       app.current_read=hex2dec(x3)*0.1;
       app.MotorCurrentAEditField.Value=app.current_read;


       %01.21 Motor Voltage
       x4=[x(46,:) x(47,:)];
       app.voltage_read=hex2dec(x4)*0.1;
       app.MotorVoltageVEditField.Value = app.voltage_read;

       %01.22 Motor Torque
       x5=[x(48,:) x(49,:)];
       app.torque_read=hex2dec(x5)*0.1;
       app.MotorTorqueEditField.Value = app.torque_read;
```

Table 5.3.4-Code relating to the implemention of the actual values reading

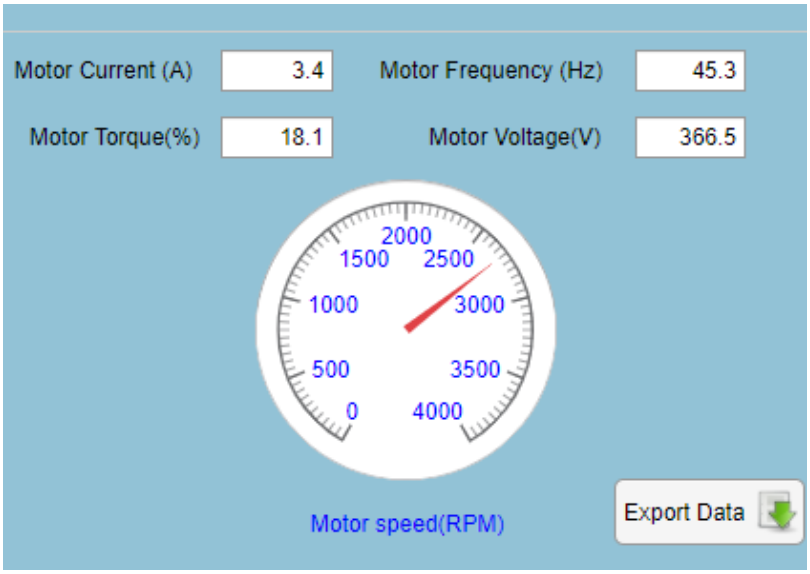

Figure 5.3.2-Reading parameters implementation in the GUI App Designer

Drive information

In the *ReadParameters* function there is a code section intended for acquiring certain info about the inverter and the motor. In particular it has been realized to obtain information on the motor status, run or not, on the presence of fault and on the control mode to ascertain an appropriate inverter using.

All these information are contained in the inverter group *06-Drive status,* in particular in the parameter *06.00-Status word1*, described in table 5.3.5.

06.00 Status word1	Drive status word1 .		
	position	Name	Information
	0	Ready	1=Drive ready to receive start command.
			0=Drive not ready.
	1	Fault	1=Drive fault.
			0=Drive no fault.
	2	Alarm	1=Drive warning.
			0=Drive no warning.
	3	Limiting	1=Drive limited.
			0= Drive unlimited.
	4	Running	1= Drive running.
			0=Drive not running.
	5	Rev req	1=Drive starting reversal.
			0=Drive starting forward.
	6	Start req	1=Driver received Start request.
			0=Drive not received Start request.
	7	Stop req	1=Drive received shutdown request
			0=Drive not received shutdown request.
	8	JOG active	1=Drive jog operation.
			0=Drive jog function not activated.
	9	Int stop req	1=Drive internal forced shutdown activated.
			0=Drive forced shutdown function not activated.
	10	Ext run enable	1=Drive external operation enabled.
			0=Drive external operation not enabled.
	11	JOG2	1=Drive JOG2 activated.
			0= Drive JOG1 activated.
	12	DC charged	1=DC high voltage capacitor charging completed.
			0=DC high voltage capacitor charging not completed.
	13	Chg rly closed	1=Soft start relay closure.
			0=Soft start relay disconnect.
	14	Ext2	1=Control place2 activated.
			0=Control place1activated.
	15	Loc ctrl	1=Drive operates in remote control mode.
			0=Drive operates in the local control mode.

Table 5.3.5-Structure of Drive status word 1, parameter 06.00

The previous table has been extrapolated from the inverter firmware manual, after several reading tests of the drive status it has been possible to notice an error in the description of position 15, relating to the local control, for which the bit 0 is for the Remote control while the Local is described by bit1.

According with the 06.00 Status word1 structure it has been possible to evaluate the specific request info that are: Fault in position 1, Drive Running in position 4, Local control in position 15. The bit in posion 0 is the least significant bit (LSB),

the position 15 represents the most significant bit (MSB), so it is possible to considerer the following identification, table 5.3.6, for extracting the needed drive information and to facilitate the reading.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1 MSB	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16 LSB

Table 5.3.6-Bit position correspondence

For the above consideration the status word will be interpreted in such a way as to identify the highlighted bit in table 5.3.6.

The request to read the parameters 06.00, will have the structure of request frame for function code 0x03, thus it has been necessary to use *write()* function, to specify the register address 0600, and the number of registers to read, in this case the only register of interest is 06.00-StatusWord1 thus the number of registers to read is 1 (0001 in hexadecimal). The complete structure of the message is described in table 5.3.7. The reading is achieved by specifying the number of values to read, in this case the total number of bytes of the frame is 7 (1 for the Slave address, 1 for the function code, 1 for the number of bytes for data, 2bytes for the *status_word*, 2 bytes for the CRC code).

```
% READ OF DRIVE STATUS
 request = uint8(hex2dec(['01'; '03'; '06'; '00'; '00'; '01'; '84';
'82']));
 write(app.serial, request,"uint8"); %start in dec
 data_status=read(app.serial,7,"uint8");
 x=dec2hex(data_status);
 x1_status=[x(4,:) x(5,:)];
 x2_status=hex2dec(x1_status);
 status_word=dec2bin(x2_status,16)

   if (status_word(12)=='1')  %bit 1=DRIVE RUNNING/0=NOT RUNNING
       app.RunorStop.Text='RUN';
       app.RunorStop.FontColor='g';
   else
       app.RunorStop.Text='STOP';
       app.RunorStop.FontColor='r';
```

```
end

if(status_word(15)=='1') %bit 1=DRIVE FAULT/0=NO FAULT
   app.Faultornot.Text='FAULT';
   app.Faultornot.FontColor='r';
else
   app.Faultornot.Text=' NO FAULT';
   app.Faultornot.FontColor='k';
end

 if (status_word(1)=='0') %bit 1=Local control/0=Remote control
   app.LocalorRemote.Text='REMOTE';
 else
   app.LocalorRemote.Text='LOCAL';
 end
```

Table 5.3.7- Code relating to the status word reading

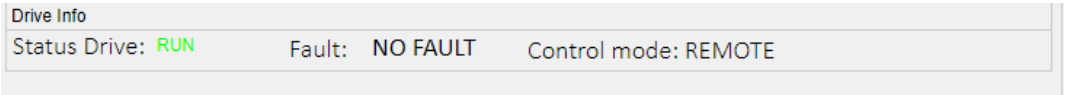

Figure 5.3.3 –Drive info implementation on the GUI of App Designer

Plot of actual parameters

The final step of the GUI design has been the implementation of graphs able to reproduce the real-time actual parameters behaviour, for this reason it has been used a different panel in the interface, denominated as "*Monitoring*", to visualize the plots.

The plots are managed by the *ReadParameters* function because they are dependent from the timer. The time axis is adjusted by means of the variable *time_plot,* that follows the timer configuration, by updating its value according to the timer *Period* set to one.

In order to appreciate the chosen type of speed control, with V/Hz regulation, it has been chosen to plot the voltage and frequency vs. time, and the time varying graph of the V/ F ratio, which represents the magnetic flux trend.

In the *Monitoring* panel it has been inserted a *StopRecordButton* able to stop the plotting, for allowing to the users a graph analysis, after it is pushed, according to the *flag_stoprecord* variable*, as* it has been explained in table 5.3.9*,* it became a *StartRecordButton* in order to consent the continuous control of the plots display.

```
%% PLOT FOR MONITORING PARAMETERS

 %Vector creation
 app.voltage_plot(1,app.index) = app.voltage_read;      %voltage
vector
 app.freq_plot(1,app.index) = app.freq_read;            %freq vector
 app.voltagefreq_plot = app.voltage_plot./app.freq_plot;%V/F vector

 app.time_plot(1,app.index) = app.time_var;

 if(app.flag_stoprecord)

   plot(app.UIAxes,app.time_plot,app.voltage_plot);     %voltage graph
   plot(app.UIAxes2,app.time_plot,app.freq_plot)        %freq graph
   plot(app.UIAxes3,app.time_plot,app.voltagefreq_plot) %V/F graph

 end

  app.time_var = 1+ app.time_var;
  app.index = app.index+1;
  hold on
```

Table 5.3.8-Code relating to the plot of actual parameters

```
% Button pushed function: StopRecordButton
    function StopRecordButtonPushed(app, event)
        app.flag_stoprecord = not(app.flag_stoprecord);
        if(app.flag_stoprecord)
        app.StopRecordButton.Text = 'Stop Record';
        else
        app.StopRecordButton.Text = 'Start Record';
        end

    end
```

Table 5.3.9-Code relating to the StopRecordButton

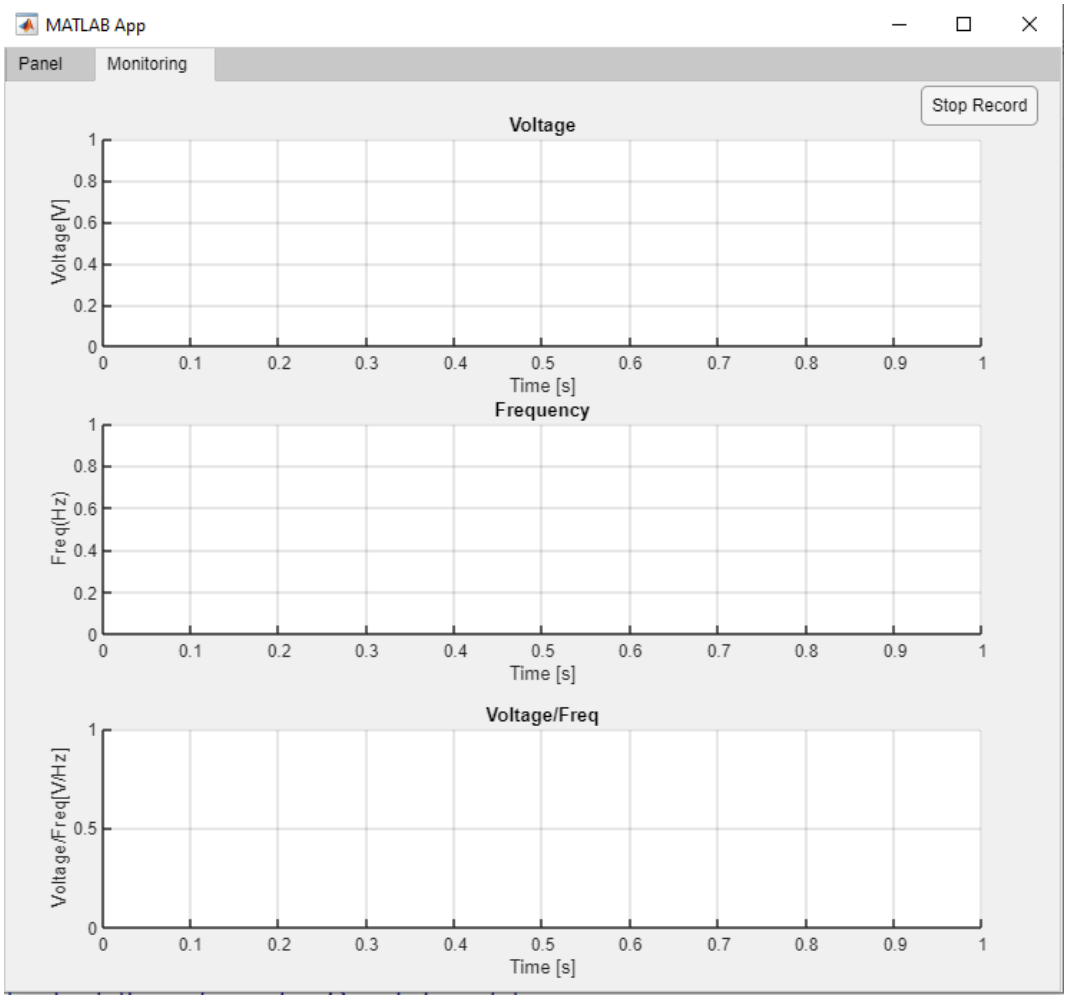

Figure 5.3.4-Monitoring panel implementation in the GUI of App Designer

The final design of the GUI has been shown in figure 5.3.5; it has been chosen to disable all the buttons in the *startup* function, table 5.3.10, exept for *Open Port* and *Close Port*, to avoid possible crash of the application if a user can use the control panel before opening the serial port. The buttons will be enabled after the correct opening of the serial port.

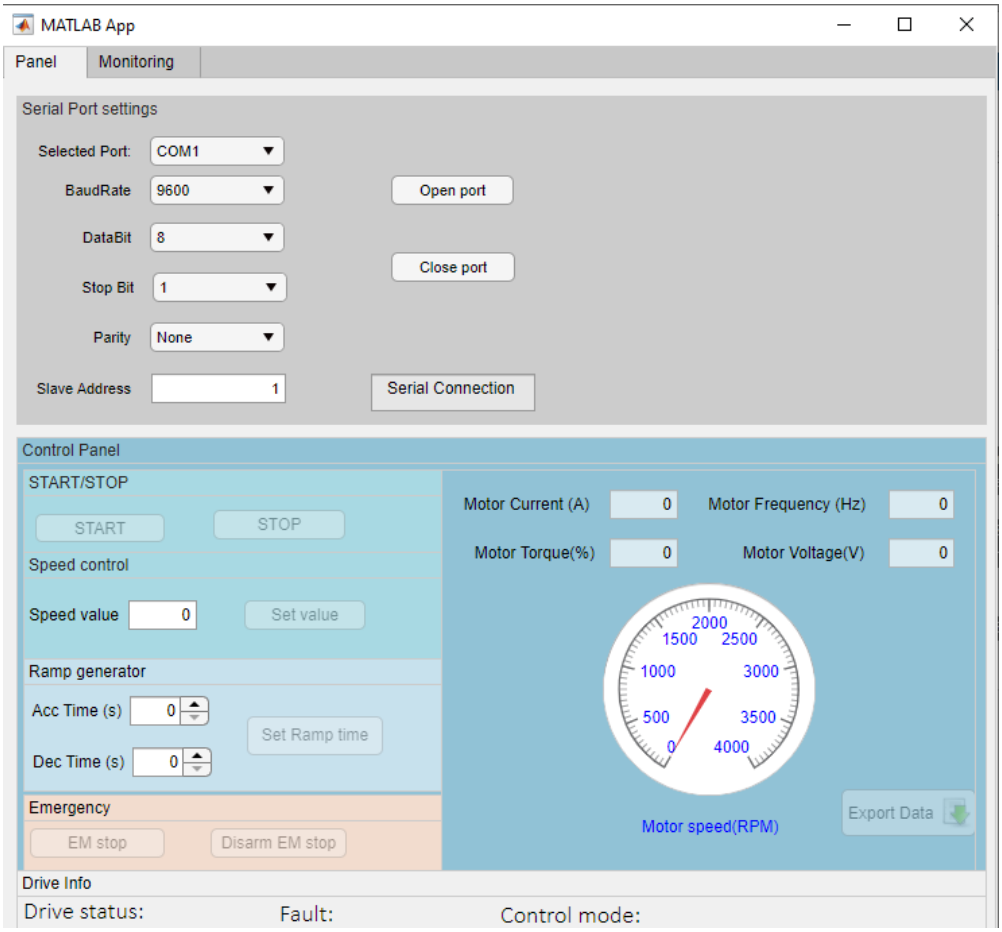

Figure 5.3.5-Final implementation of the GUI for the motor speed control

Once the entire code was created to implement the graphical interface, it was possible to finalize the project by creating a standalone application for desktop denominated as "*Inverter_cmk*", figure 5.3.6.

Figure 5.3.6- Desktop connection of *Inverter_cmk* application

The figure 5.3.7 shows the using of the graphic interface for the motor control, in the case of a speed reference setting equal to 2500 rpm.

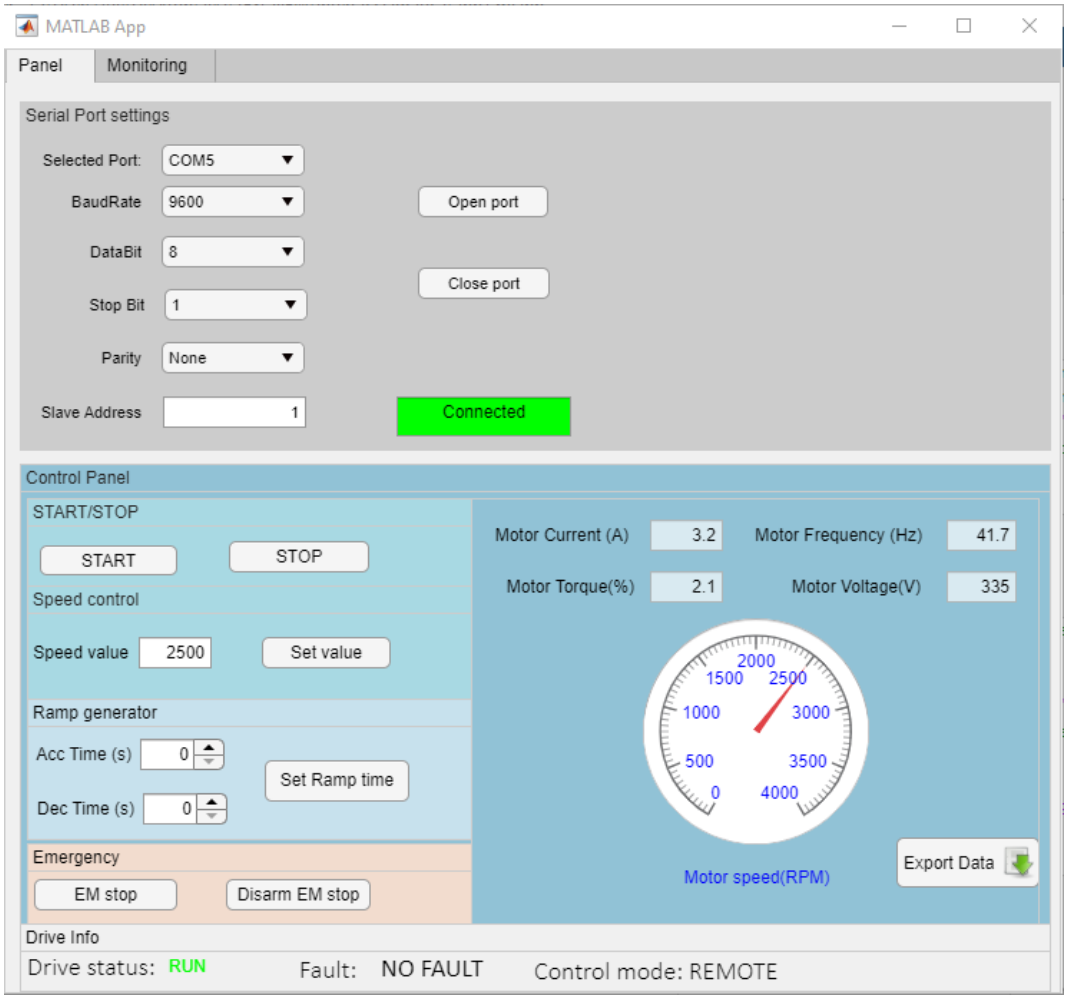

Figure 5.3.7-Test with GUI for the motor speed control, by using a speed reference equal to 2500 rpm

6. Conclusion and results

In conclusion, the application was used to monitor the set motor control and to evaluate the overall functioning. In the figure 6.1 there is the practical connection among the PC and the Inverter for exploiting the graphic interface.

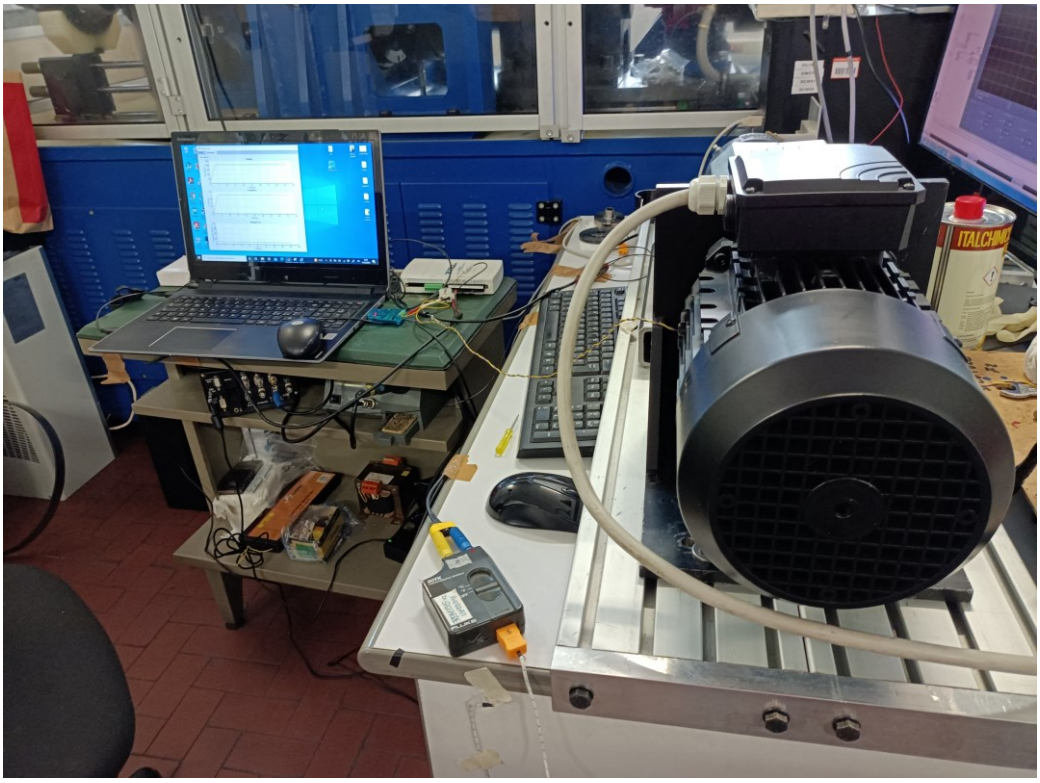

Figure 6.1-Serial communication master (PC)-slave (Inverter), for the GUI using

The following figures show five different analyzed cases during the motor operation.

- Case 1. Speed reference set to 2500 rpm

The open port has been correctly opened, thus the serial connection text shows "Connected". Once the reference speed is set, through the *Set value button*, it is possible start the motor and the value of the parameters begins to grow until they settle in the steady state. The drive info panel shows the main device feautures

during the functioning, and the appropriate applied control mode. The *gauge* component illustrates the achievement of the desired motor speed.

Figure 6.2-Test with GUI for a motor speed control by applyng 2500 rpm

In the monitoring panel it is possible to appreciate the selected acceleration time equal to 6 second, highlighted through the ramp profile of the voltage and frequency graphs, figure 6.3.

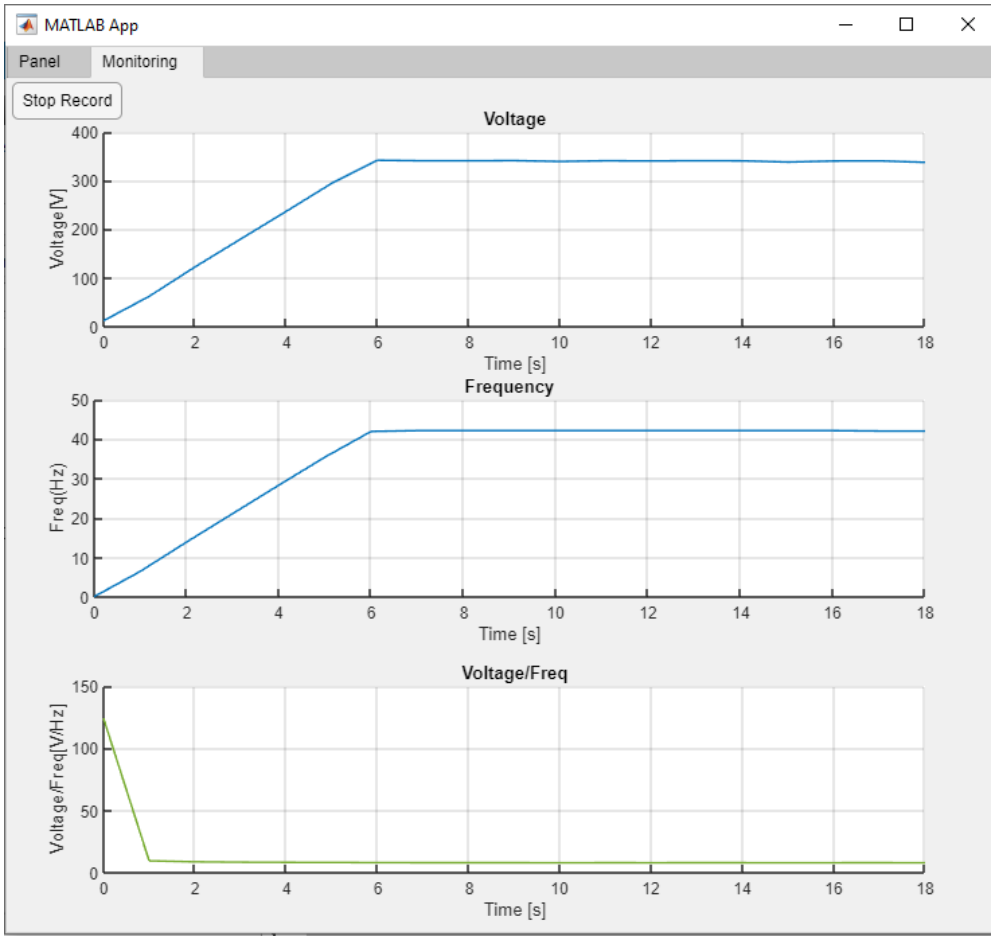

Figure 6.3- Monitoring panel for a test with the speed set to 2500rpm

- Case 2. Speed variations (0, 2500rpm, 2800rpm)

The second test has been performed to evaluate the speed variations according to the set acceleration time. It has been chosen to realize the first acceleration from 0 to 2500 rpm in 5 seconds, then with an acceleration time equal to 2 seconds the final speed of 2800 rpm has been achieved. In the figure 6.5 the monitoring panel illustrates the variations of voltage and frequency, consistent with the values shown in the main panel (figure 6.4), indeed, at the value of 2500 rpm speed corresponds the $V_s = 339{,}1V \; and \; f_s = 42{,}1 \; Hz$, whereas, the final value of 2800 rpm coincides with the $V_s = 377{,}9V \; and \; f_s = 46{,}9 \; Hz$.

Figure 6.4-Main panel, test with GUI for a motor speed control by applying 2800rpm

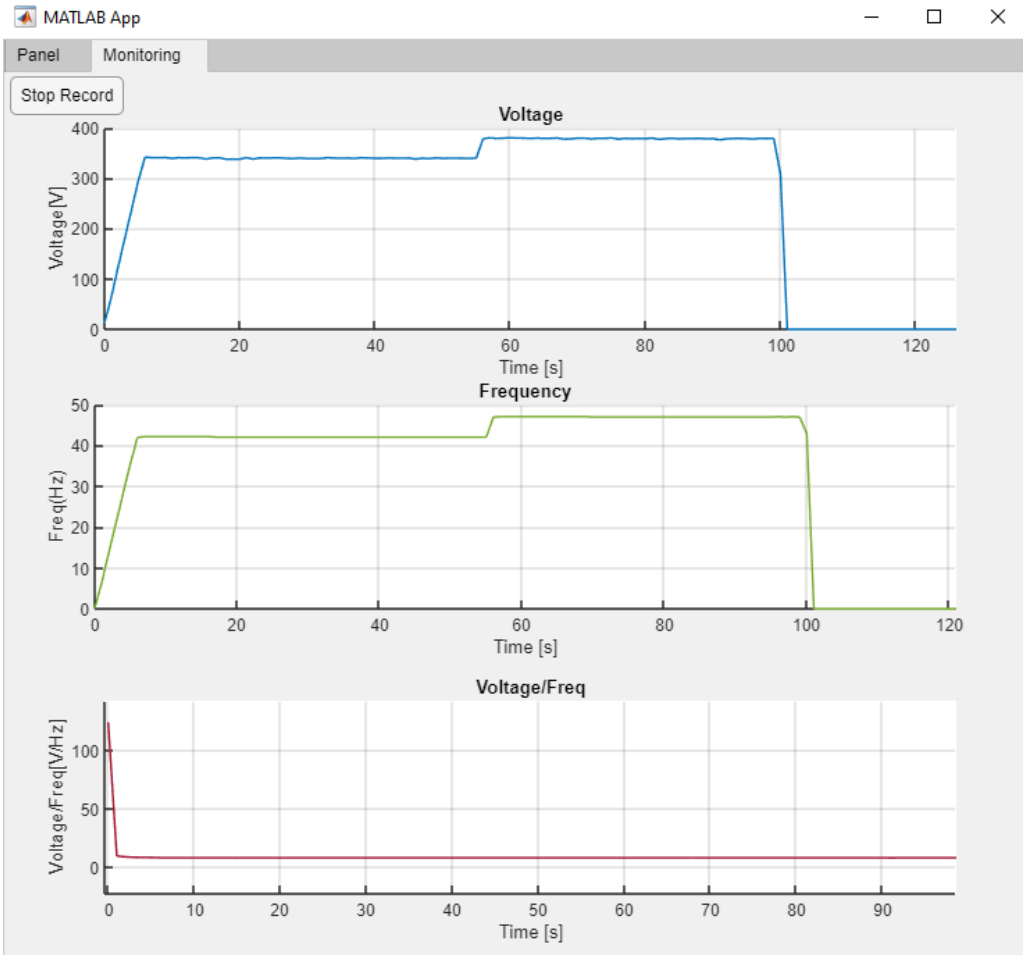

Figure 6.5-Monitoring panel for a test with speed variations from 0 to 2500rpm, from 2500rpm to 2800rpm

The final ramp, at second 100s, has been described in the following case.

- Case 3. Emergency Stop mode

After 100 seconds the motor has been stopped with an emergency stop. Once the *EM Stop* button is pushed, the motor instantly arrests. It is evident in the figure 6.6 that *Status drive* in *Drive info* highlitghts the operating mode of the motor, and consequently to the stop of the motor, all the parameters properly decrease to the value 0. Only after the *Disarm EM stop* button is pushed will be possible to restart the motor.

Figure 6.6-Main panel, during a test with GUI, after the complete stop of the motor

The figure 6.7 highlights the emergency stop mode through the steep slope for both voltage and frequency behaviour.

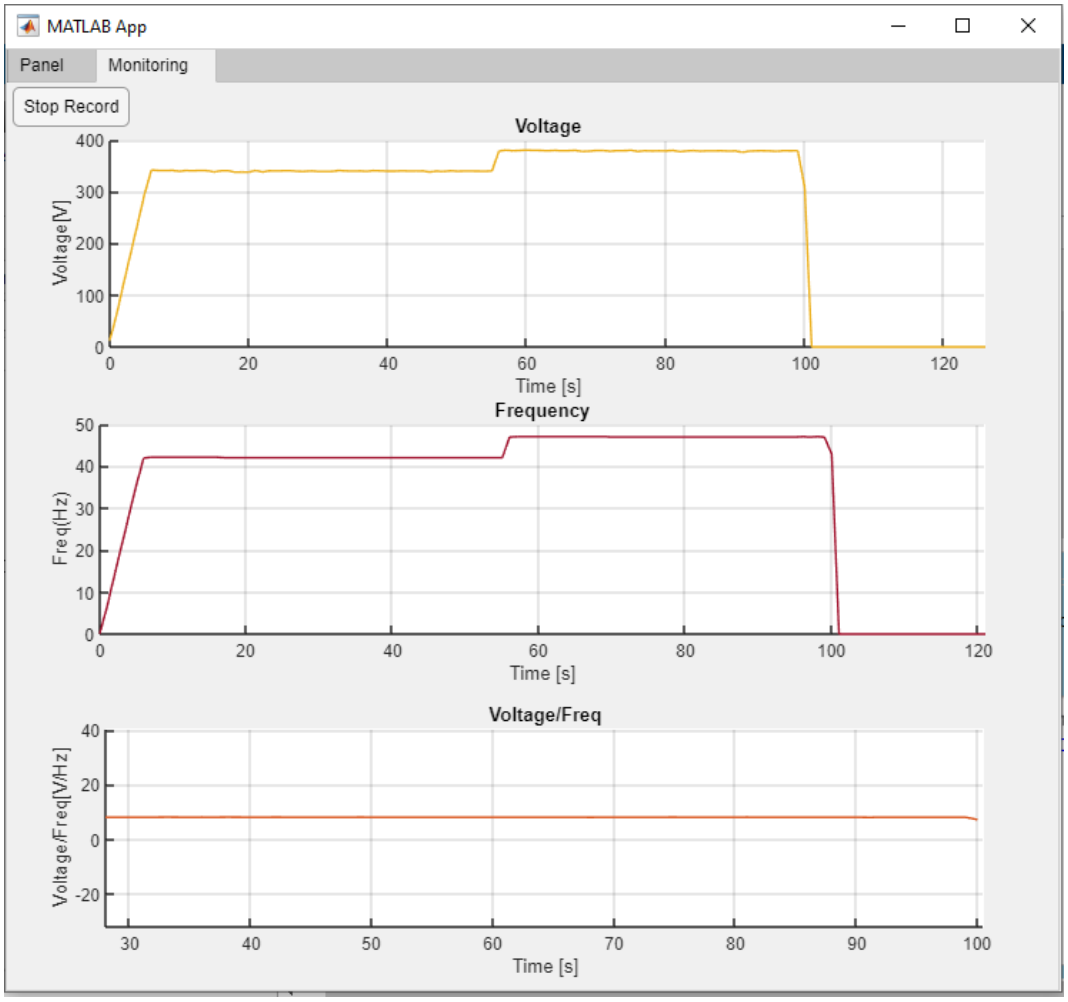

Figure 6.7-Monitoring panel, during a test with GUI, after the emergency stop button is pushed

- Case 4. Stop mode with deceleration time set to 8s

On the contrary of the emergency stop, the normal stop is controlled by the deceleration time imposed by the *Set RampTime* button. In the figure 6.8 is possible to appreciate the deceleration time for which voltage and frequency pass from a certain value, at time 255s, to 0, after 8 seconds, at time 263s. At the same time the ratio V/F increases according to the voltage and frequency drop, until to the complete stop of the motor. The unexpected increasing is due to the faster frequency decreasing than the voltage fall, when the frequency reaches the 0 value the plot stops to show the curve due to the division $V/0$ and 0/0 conditions.

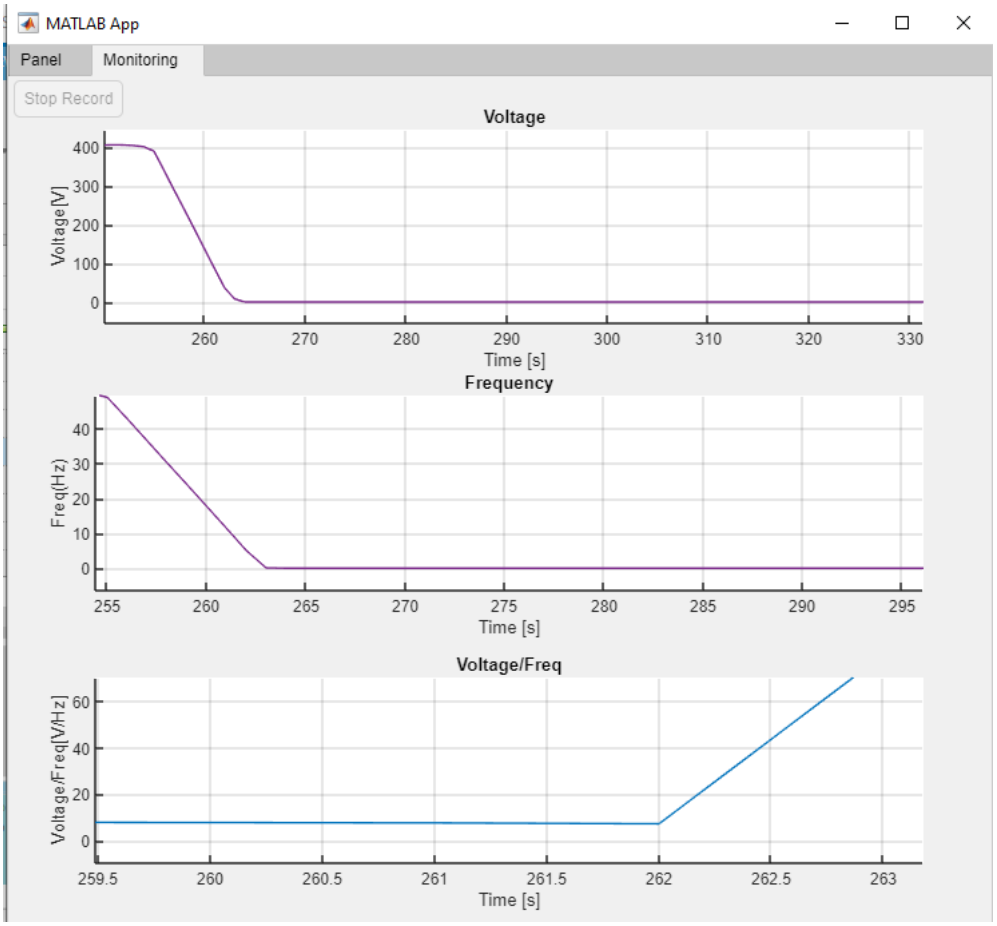

Figure 6.8- Monitoring panel, during a test with GUI, after the stop button is pushed and the deceleration time is set to 8s

- Case 5. Speed reference greater than 3000 rpm

To appreciate the V/Hz regulation, the speed reference has been set equal to 3000 rpm, approximatively equal to the motor rated speed, thus the frequency is close to the rated value $f_{sn} = 50Hz$, while the voltage must maintain a value equal to the rated voltage $V_{sn} = 400V$.

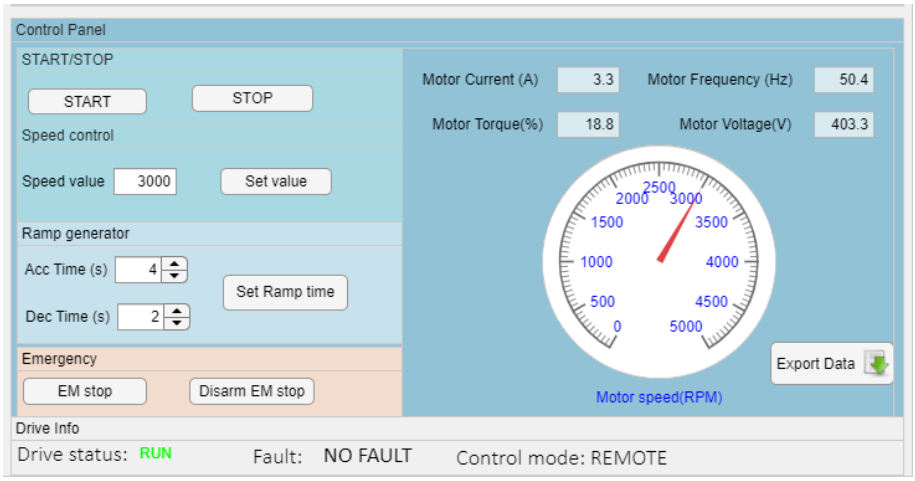

Figure 6.9- GUI control panel during a test of speed control with 3000 rpm

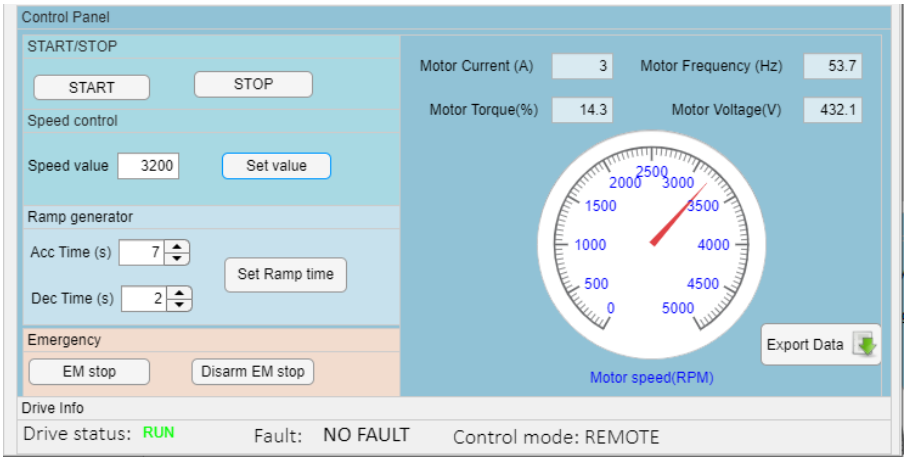

Figure 6.10- GUI control panel during a test of speed control with 3200 rpm

By consulting the graphs in the monitoring panel and the updating parameters in the control panel, figures 6.9-6.11, it is possible to assert that once the motor speed exceeds the rated speed of 2865 rpm, the inverter performs a proper motor control with V/Hz regulation, in fact with 3000 rpm result $Vs = 403{,}3\ V, fs = 53.7$, thus the voltage try to maintain a value possibly approximable to the rated voltage V_{sn}.

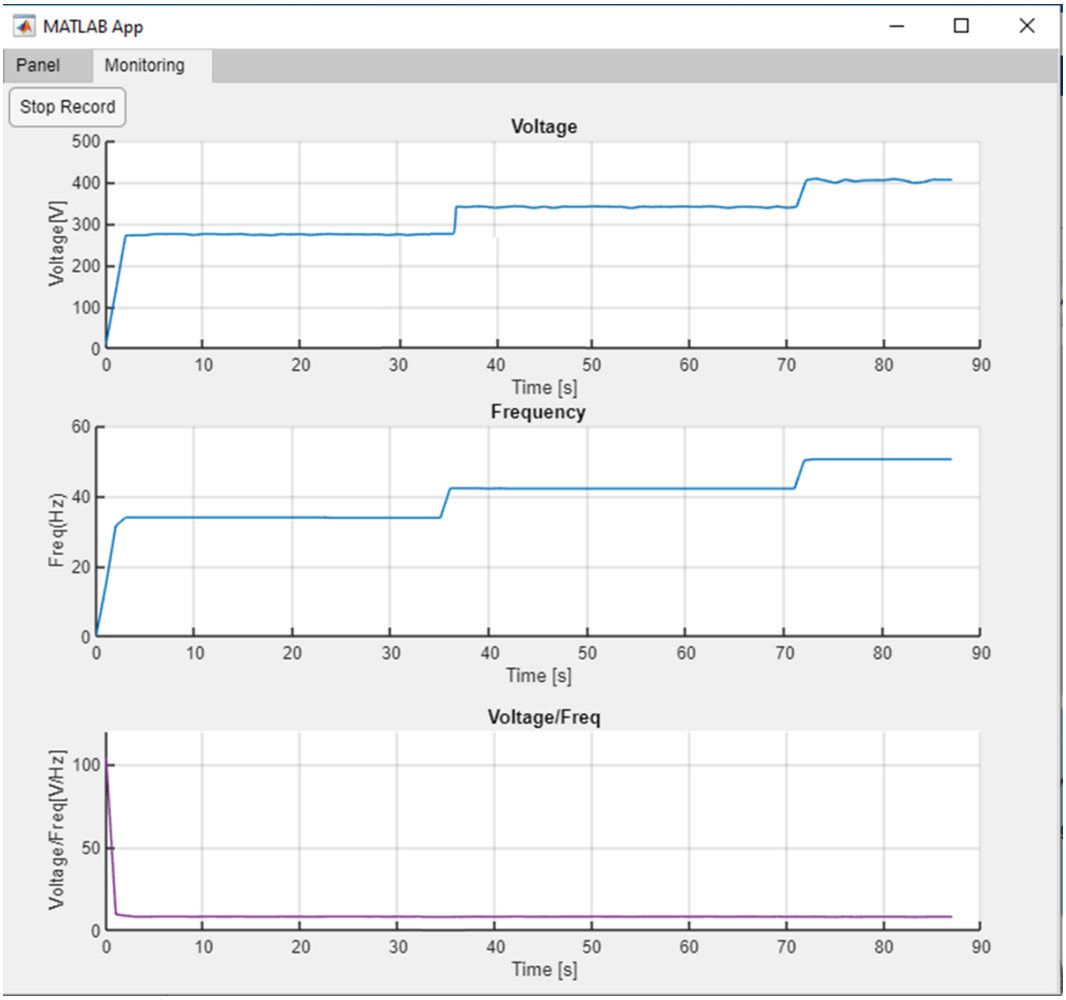

Figure 6.11- Monitoring panel in the GUI, during a test of speed control with 3000 rpm

In the monitoring panel it has been evaluated the time behaviour of the motor voltage and frequency, while the last graph demonstrates the proper waveform of the V/F ratio that approximate the flux trend, it results costant in the interval $[0, f_{sn}]$, while according with the frequency increasing w.r.t the rated value, there is a flux weakening.

By using increasingly high speed, the output voltage presents some ripples at the settling time, this behaviour is emphasized in figure 6.12.

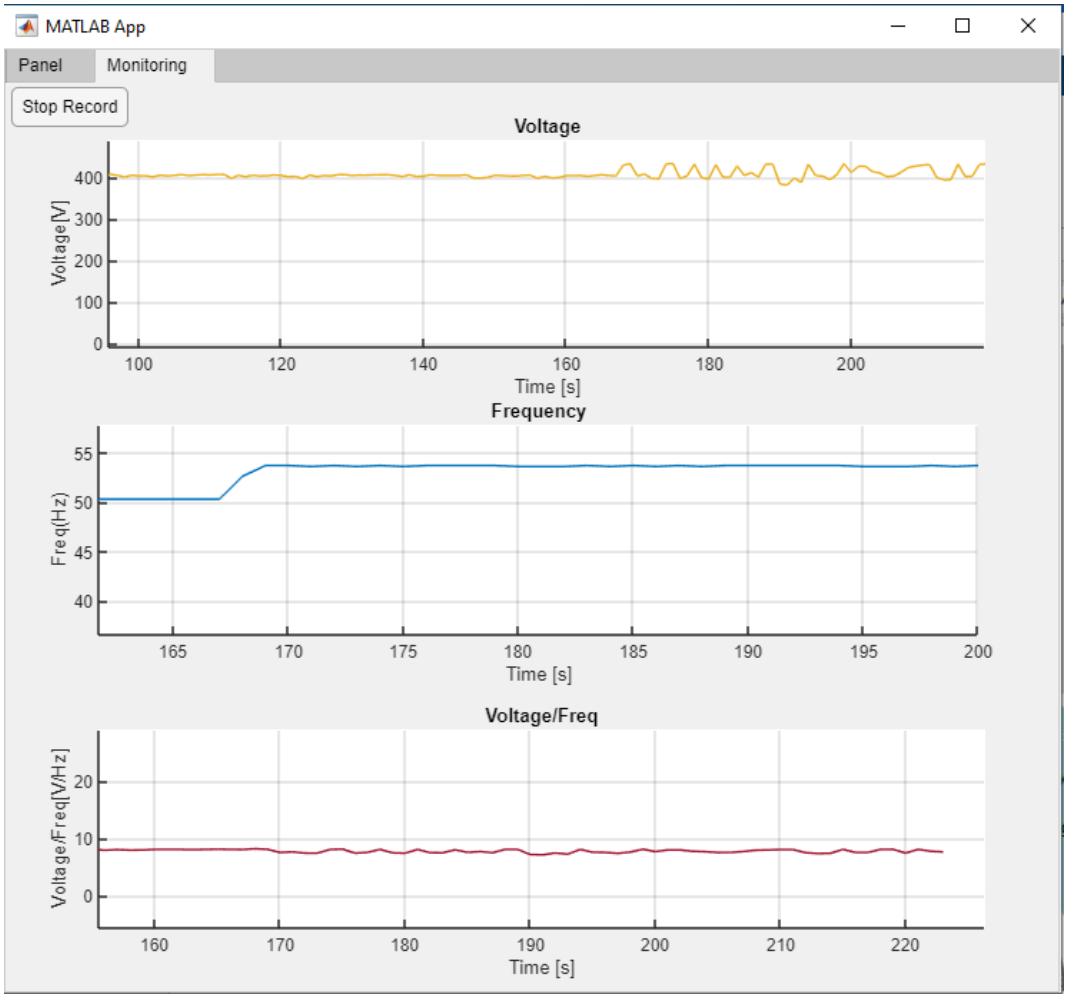

Figure 6.12 – Monitoring panel, in the GUI, during speed increasing from 3000 rpm to 3200 rpm

By applying a set speed of 3200 rpm, the voltage and frequency behaviors allow us to state that a proper V/Hz regulation has been implemented, nervethless it is possible to observe significant ripples around the final value, due to some controller inaccuracies the voltage oscillates in an interval [399,9-435,3V], close to the rated value $V_{sn} = 400\ V$.

In conclusion, the final test bench will be used to test graphene greases in thrust bearings, exploiting as load for the motor.

The electrical cabinet has been designed for managing the motor in combination with the inverter, in particular, it has been chosen to add a *three-positions commutator* for permitting the exclusion of the inverter.

Finally, it has been realized an application with *App Designer* tool of *MATLAB*, employing Modbus RTU protocol, to control the system in remote mode, all the inserted buttons in the GUI are necessary for implementing a complete and effective speed control system.

The speed control has a 5000 rpm limit for the motor, this allows to reach high performances. The speed control test has been performed on load and off load, and the results are very similar, thus the load doesn’t affect the performance.

7. Bibliography

[1]https://crushtymks.com/it/electric-motor/277-comparision-of-direct-on-line-dol-and-star-delta-motor-starting.html

[2]https://elettronicasemplice.weebly.com/sistema-trifase-stella-triangolo.html#:~:text=%E2%80%8BA%20STELLA%20%2D%20si%20collegano,una%20tensione%20di%20230%20volt%20.&text=A%20TRIANGOLO%20%2D%20si%20collegano%20i,una%20tensione%20di%20400%20volt.

[3]https://gallery.proficad.com/pages/Results.aspx?kw=&grp=150&ctl00%24ctl00%24main%24main%24buttonSubmit=%C2%BB%C2%BB%C2%BB

[4]https://campoelettrico.it/blog/p-teleruttore-contattore-a-cosa-serve-funzionamento

[5] https://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf

[6]https://www.inftub.com/economia/finanze/Azionamenti-a-velocit-variabil93546.php

[7]https://politecnicobarimy.sharepoint.com/personal/studentidemocratici_poliba_it/_layouts/15/onedrive.aspx?id=%2Fpersonal%2Fstudentidemocratici%5Fpoliba%5Fit%2FDocuments%2FSTUDENTI%20DEMOCRATICI%2FDEI%2FMagistrale%2FIng%2E%20Automazione%2Fsecondo%20anno%2Fazionamenti%20elettrici%2FSlide%20%2D%20Appunti%20Docente%2FMotore%20ad%20induzione%5Fcap1%2Epdf&parent=%2Fpersonal%2Fstudentidemocratici%5Fpoliba%5Fit%2FDocuments%2FSTUDENTI%20DEMOCRATICI%2FDEI%2FMagistrale%2FIng%2E%20Automazione%2Fsecondo%20anno%2Fazionamenti%20elettrici%2FSlide%20%2D%20Appunti%20Docente

[8]https://studylibit.com/doc/4141207/controllo-della-velocit%C3%A0-di-un-motore-in-ac

[9] ES series cluster driver Hardware manual.pdf

[10] Catalogue LAFERT AC_Motors_MKTG-10-127_16_IT.pdf

[11]https://www.virtual-serial-port.org/it/articles/modbus-rtu-guide/#:~:text=tentativo%20di%20comunicazione.-,RTU%20Modbus%20vs%20TCP,Modbus%20su%20reti%20TCP%20%2F%20IP.

[12]https://it.wikipedia.org/wiki/Modbus

[13]http://www.robertopasini.com/index.php/tips-italian/2-uncategorised/647-hw-pwm#:~:text=Un%20segnale%20PWM%20(Pulse%20Width,(modulando)%20il%20duty%20cycle

[14]https://it.wikipedia.org/wiki/Controllo_in_anello_aperto

[15] https://it.mathworks.com/products/matlab/app-designer.html

[16]ES350_580_850 firmware manual.pdf

[17]https://it.mathworks.com/help/matlab/ref/timer.html

[18]https://it.wikipedia.org/wiki/Transistor_bipolare_a_gate_isolato

[19]https://www.theengineeringknowledge.com/what-are-the-speed-control-method-of-induction-motors/

[20] http://www.sunshine2k.de/articles/coding/crc/understanding_crc.html

[21] https://it.mathworks.com/matlabcentral/fileexchange/69606-modbus_crc16

[22]https://ozeki.hu/p_5881-mobdbus-function-code-6-write-single-holding-register.html#:~:text=This%20function%20code%20is%20used%20to%20write%20a,Therefore%20register%20numbered%201%20is%20addressed%20as%200.

[23] https://realpars.com/modbus-protocol/

8. Acknowledgements

First of all, I would like to express my gratitude to my thesis supervisors Prof. Andrea Mura and Prof. Luigi Mazza, for their willingness and kindness shown me during all the work time. I would like to thank Dr Edoardo Goti for all the support provided. I would like to express my gratitude to my family to be always on my side. Thanks to Davide for the infinite patience and support. Finally, I would like to all my friends to be a costant of leisure.