UNIVERSITY OF SOUTHERN DENMARK

Faculty of Engineering
Institute of Technology and Innovation

Diego FraguasTejero

CRUISE CONTROL ON AN ELECTRIC BIKE

Report

Report regarding the Electrical Power Engineering Project, from the corresponding course, at the Southern Denmark University, during the Autumn semester.

Supervisor: Heiko Steenbuch Vester

ODENSE

2011

Índice

1. SYNOPSIS

The main goal of this project is apply the learnt knowledge during this semester at the SDU in the courses of PEM and CES.

The project that we have chosen consists on control an electric bike motor, so it can assist the ride. So, if the speed is lower than the desired speed, the motor must speed up, and vice versa.

In order to achieve this goal, we need to deal with three defined subjects: the motor, the controller and the power electronics.

2. PREFACE

The first two wheel vehicle with steering system was designed in 1816 by a German scientist. Although, there are old testimonies about this vehicle. It is said that Leonardo Da Vinci painted a cycle in one of his works.

In 1839, pedals were included in this machine with the purpose of riding it without touching the ground with the feet.

In the following years, the experts dedicated the time to improve the previous models and designs.

As an anecdote, we want to comment that in 1887, Thomas Stevens, origin from the USA, was the first man who went around the world riding a cycle. He took 3 years to complete the whole trip.

In 1889, the first official cycling competition appears. From then onwards, there is more curiosity in improving the characteristics and the performance of the cycle.

In the last decade, the electrical bicycles have begun to market. They have an engine to help the cyclist in his ride. The usually bikes don´t need energy therefore its used is increasing, but although this is a disadvantage for the electrical bikes, they are very useful in places with large slopes, and factors like the wind will be eliminated. On the other hand, you can ride bigger distances with much less effort.

3. INTRODUCTION

For the realization of this project we have to take into account some aspects related with the idea of our project: cruise control on an electric bike.

First of all, we must have a clear idea about what an electrical bike consist on: is a machine designed to move through the effort of the cyclist using an electrical motor in order to help the cyclist, and not to replace. In case there is no effort of the cyclist, the vehicle will be considered as a motorbike.

There are some aspects that determine the cycling: the way to drive, the state of the bicycle (for example the tire pressure), the state of the battery, ambient temperature, the wind, topography, the weight, the age, the rider´s physical condition…

On the other hand, a cruise control is an electronic system that automatically controls the speed of a vehicle motor and maintains it steady as set by the driver without use the accelerator or break.

We think that our cruise control must satisfy some conditions:

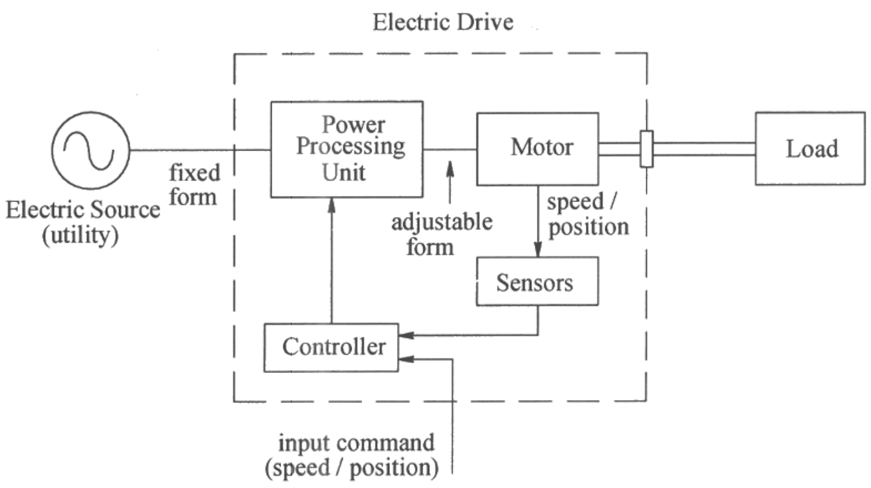
-The motor can also be turned on or off using a switch, in order to let the rider choose use it or not.

-The speed must be steady without taking care about the required torque.

-As a safety condition, the motor has to stop when the cyclist breaks.

-The energy dissipated during the break, will be used to charge the battery.

The system can be described in the image shown below:



As we can see, our system is divided in four parts: the motor, the sensor, the controller and the power electronics. As well, we'll need a power supply to feed our system and a load (bicycle torque).

The electrical drive is a system which performs the conversion of electrical energy to mechanical energy.

Now we are going to describe briefly the components of the electrical drive:

The motor: The motor obtain power from electrical sources. It converts electrical energy into mechanical energy. The motors can be classified in AC or DC motors.

Power processor: With controllable sources, the motor can be reversed, bake or can be operated with variable speed. The characteristic of the motors can be changed at will. Power electronic converters have several advantages such as more efficiency or flexibility

Controller: The complexity of it depends on the desired drive performance and the type of motor used. The main controllers can be: analog ,digital, DSP/microprocessor

Source: Electrical sources or power supplies provide the energy to the electrical motor. For high efficiency operation, the power obtained from the electrical sources need to be regulated using power electronic converters.
Power sources can be of AC or DC. There can be several factors that affect the selection of different configuration of electrical drive system such as: torque and

speed, capital and running cost or space and weight restrictions environment and location

With the provided information in this introduction, we can conclude that the purpose of our project is to create an electronic system, which control a motor. It’s formed but two quite differenced parts: controller or logical part and driver or power electronics part.

To achieve this goal, we need to study the different parts described before, in order to know the specifications of our system.

MAIN REPORT

4. THE BATTERY

As we have a DC motor, we will need to feed it with some source, as DC power can be stored, we are going to use a battery.

The battery that we was given is a LiFePO4 one, and it fits perfectly with our motor.

The main advantages of this kind of batteries are, as the conventional lithium-ion battery:

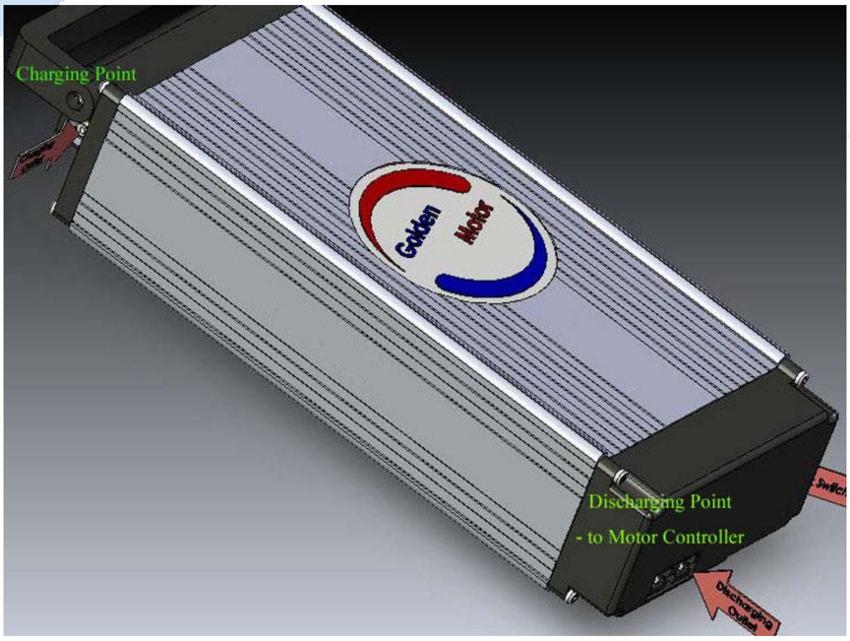
-Light weight that fits with the power it gives

- No memory effect (effect that cause the battery to hold less charge)

-Self-discharge rate approx. 5-10% per month

-Environmental friendly

-But it also has higher current and more peak power current.



Some of the specifications are:

- 5.26 kg of weigh
- 361x149x83mm
- 36 V
- 10 AH

We must say that the specifications are rated values. When the battery is completely charged is expected to have higher values of voltage and maximum current than the rated values. To make sure that our electronic devices can handle this peaks, we decided to test the battery.

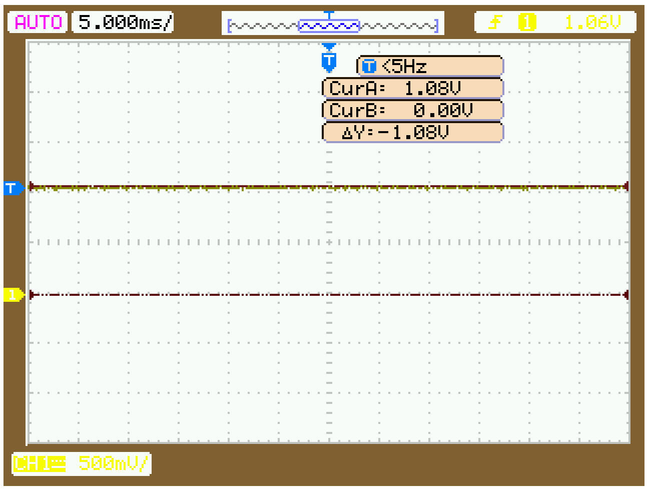
We measured the voltage of the battery fully charged without load; we measured a value of 41 V. Known the power of the motor: 500 W, we can calculate a load value in order to get the current.

$$R = \frac{U^2}{P} = 3.362\Omega$$

In the laboratory there were three variable resistances, able to handle this power, which minimum value was 10 ohms, so we connected these resistances with their minimum value in parallel in order to get a value closed to the desired resistance.

$$R_{eq} = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3}} = 3.33\Omega$$

To measure the current we couldn't use an ammeter because its fuse just can handle 10A as maximum value, so we used a peg plugged to the oscilloscope, getting the following value:



The peg has a gain of 100mV/A) so the current is of 10.8 A

5. THE MOTOR

5.1. INTRODUCTION

Due to the DC power supply, our motor must be a DC one. We can find two kind of DC motor: brushed and brushless motor. Due to the improvement that the brushless motor has (less maintenance, more efficiency...) we choose it.

5.2. WORKING PRINCIPLE

In a DC brushed motor the torque is obtained by the interaction of the magnetic field induction and the current of the rotating armature winding.

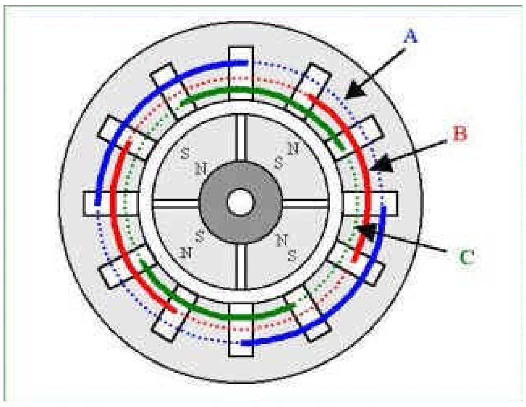
The DC brushless motor is very similar to the convectional DC motor but they are built in different way:

-The switching is done electronically rather than mechanically.

-Permanent magnets are in the rotor instead of the stator.

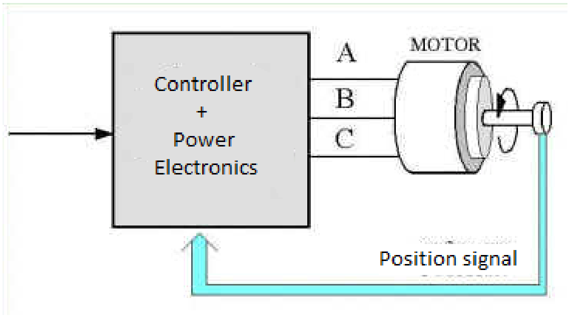
-The coils are located in the stator, forming a single-phase winding.

Brushless dc motors are typically configured as permanent magnet synchronous motors.. Its performance is based on sequential feeding of each stator phases in synchronism with the movement of the rotor. So the permanent magnets follow the movement of the magnetic field.



4 poles DC brushless motor winding

The advantage of brushless dc motors lies in the absence of rotating contacts such as slip ring, brushes in the case of conventional wound-field synchronous motors, or commutating brushes in the case of conventional commutator-type dc motors. The commutation is made electronically. So the electronic has a very important role on this kind of motors.



DC brushless motor schematic

Also its have high maximum torques, more reliability, more acceleration at start and it’s less noisy.

This type of motors, however, requires rotor position sensing in order to get a proper performance. Other disadvantages are that they are more expensive than the conventional DC motors.

In spite of these disadvantages, the brushless permanent magnet motor use is increasing due to its low maintenance and cheap power electronics.

5.3. KIND OF DC BRUSHLESS MOTOR

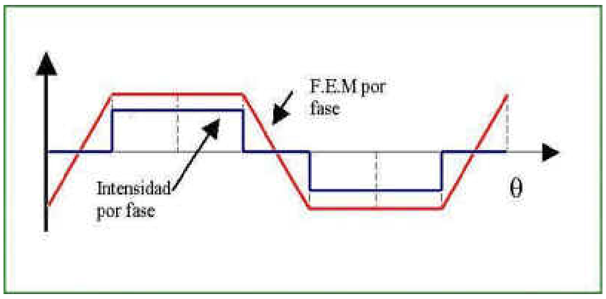
According to the wave form of induced voltage, we can classify them in:

-Sinusoidal wave:

This kind of motor must be fed with systems of sinusoidal voltage and current. Sinusoidal voltage provides a smooth motor rotation and fewer ripples. Control this type of motors is difficult.

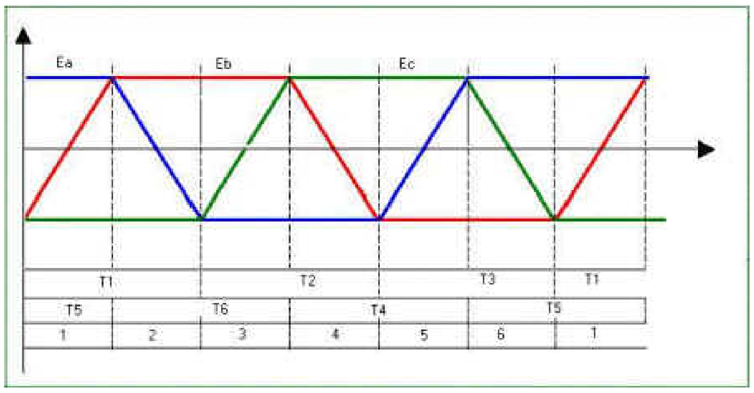
-Trapezoidal or square wave:

In this kind of motor the current is applied to two phases at a time and the third phase is left idle. Trapezoidal voltage is simpler to implement and less complex.



Basic form of voltage and current of a

In the picture above we can see the wave of current and voltage per phase.



In this picture, there are drawn 3 voltages, one of each windings, and the transistors which are divided in two groups: top (T1, T2 y T3) and lower (T4, T5 y T6). We can see that the conduction always is through two transistors, for this reason always one of the windings is off.

We also can classify this motor according to the technique used to determine the rotor position.

-Hall Sensor-based commutation: The rotor position is determined using three hall sensor set inside the rotor.

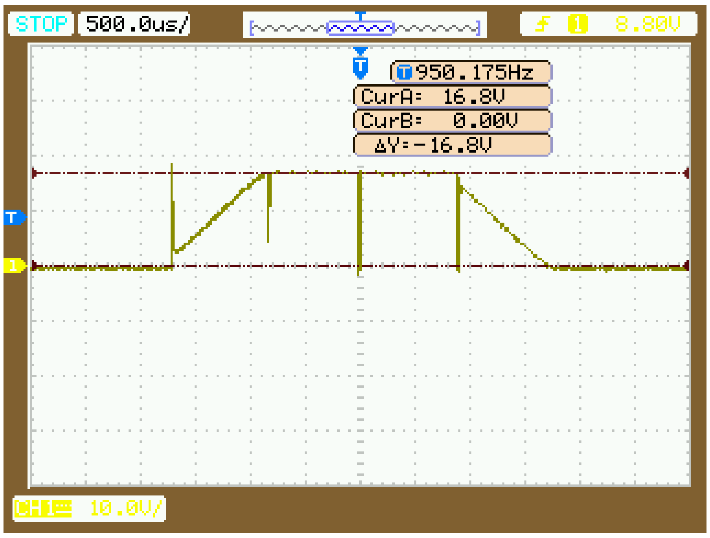
-Sensor less commutation: In the sensorless commutation technique, the back-EMF induced in the idle phase is used to determine the moment of commutation. When the induced idle-phase back-EMF equals one-half of the DC bus voltage, commutation is complete.

5.4. OUR MOTOR

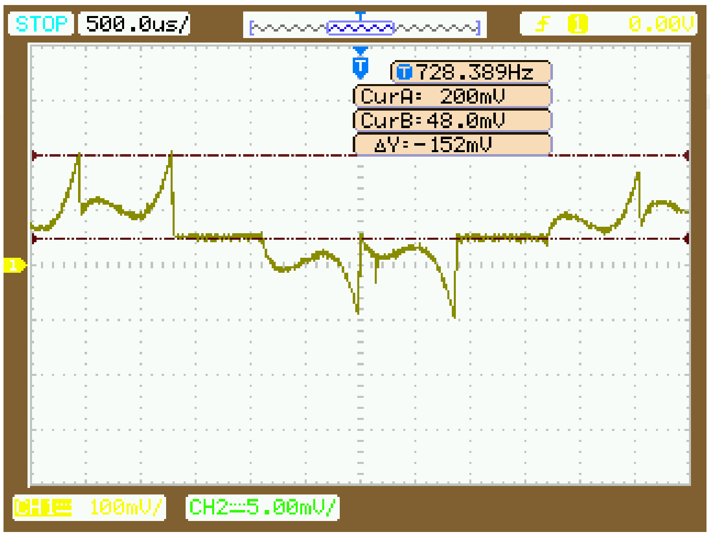
Our motor is a HBS36F sensored brushless from www.goldenmotor.com. Some specifications of the motor are: 36V, 500W and 5kg.

The motor was provided with a datasheet (*see annex), but we performed the motor in order to know a bit more about it. To do this we fed it with a DC power supply and no load. We took the measures with a digital oscilloscope.

The voltage between one phase and ground was measured using a differential prove with an attenuation of 1/20. It's shown on the image below:



The current in one phase was measured with a peg with a gain of 10mV/A and it's shown on the image below:



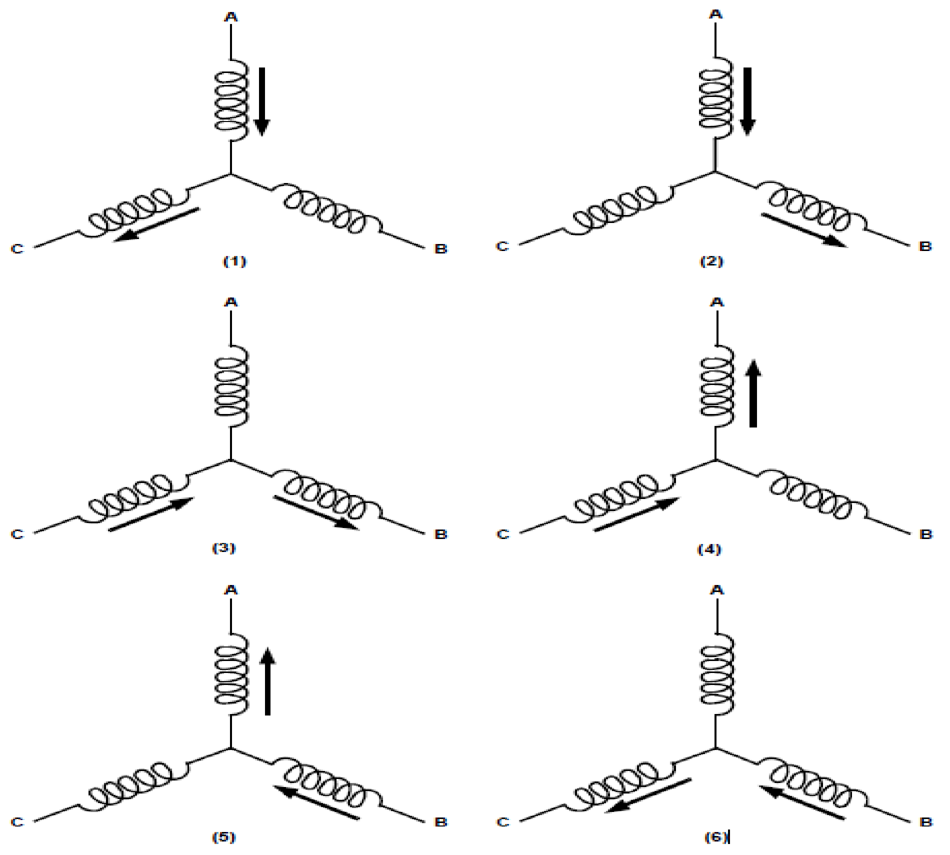
Now we know that our motor is a trapezoidal one.

5.5. FEEDING SEQUENCE

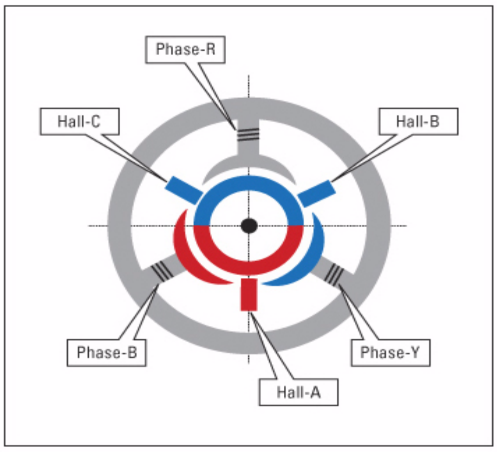
As we said before the motor need a correct feeding sequence in order to make the motor spin. Now we know our motor is a sensored one it's time to explain more in detail how it works.

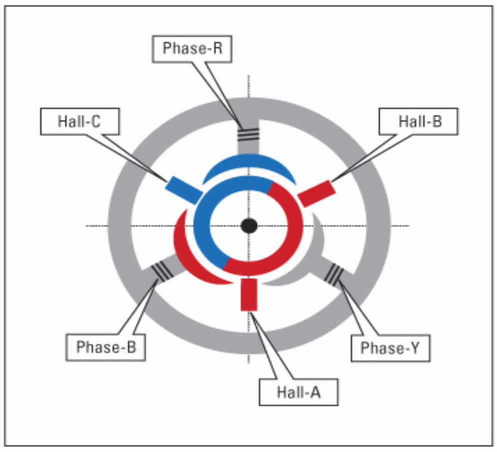
In the Hall sensor technique, there are three Hall sensors inside the motor with 120 degrees apart. Each Hall sensor provides either a high or low output (5, 0 V) based on the polarity of magnetic pole close to it. Rotor position is determined by analyzing the outputs of all three Hall sensors. Based on the output from the hall sensors, the voltages to the motor's three phases are switched.

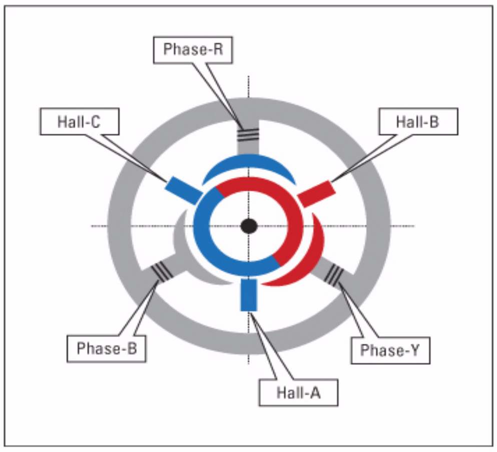
To explain this, we are going to show a picture for a 4 pole motor: Every 60 electrical degrees of rotation, one of the Hall sensors change the state. So we need six steps to complete an electrical cycle. However, one electrical cycle don't correspond to a complete mechanical revolution of the rotor. One rotor pole pairs is one electrical cycle. So, the relation between a electrical and mechanical revolution is the number of poles.

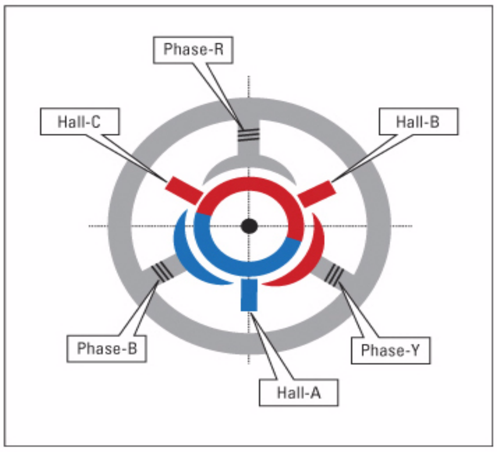


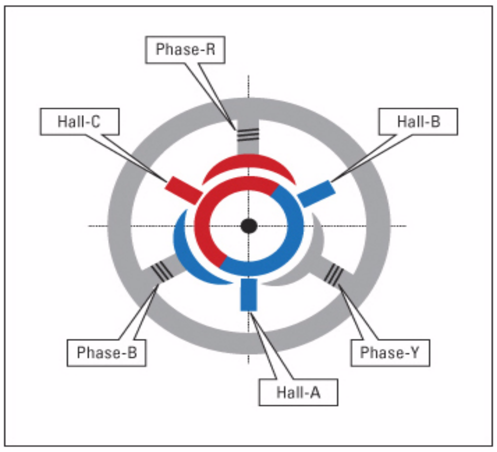
WINDING ENERGIZING SEQUENCE WITH RESPECT TO THE HALL SENSOR

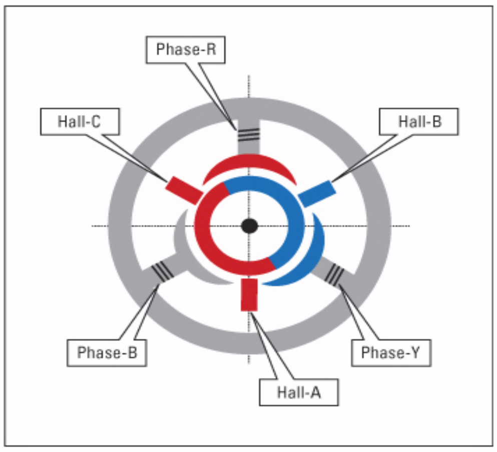












🟦 North Pole, Hall Sensor = HIGH

🟥 South Pole, Hall Sensor = LOW

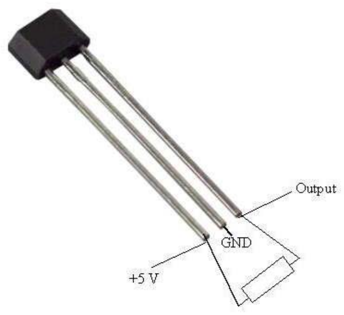
6. SENSOR

The Hall sensor uses the Hall effect to measure magnetic fields or currents in order to determine the rotor position.

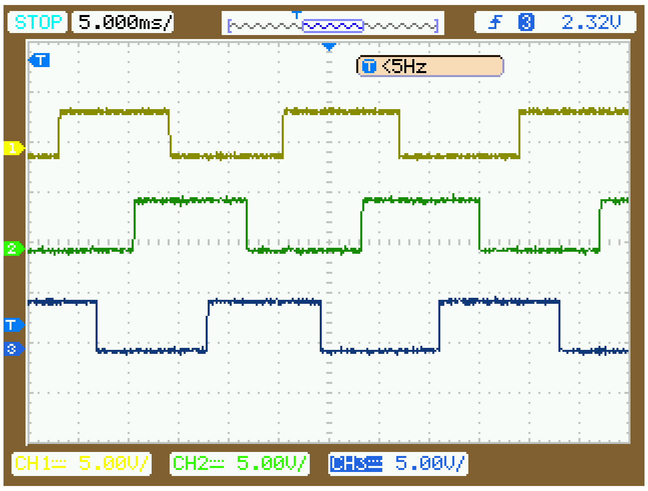
If current flows through a Hall sensor and it’s close to a magnetic field which flows perpendicular to the sensor, the sensor creates a voltage proportional to the product projection of the magnetic field and current. Then with the current value, we can calculate the magnetic field force. If the magnetic field created by current flowing through a winding, then you can measure the value of the current in the conductor or coil.

In order to determine the motor speed we need a sensor, our motor is provided with three hall sensors, due to the fact that is a brushless DC motor.

The Hall sensor detects the direction and the magnetic field from the motor. The output of these sensors is 5V when the rotor is close to it.



Using the digital oscilloscope we can know the sequence that follow our motor, it's shown in the following image:



The yellow wave is the hall sensor A, the green one is the hall sensor B and the blue one is the hall sensor C.

6.1 NUMBER OF POLES.

Now we are going to calculate the number of poles of our machine.

How can we do this?

We performed the motor with no load and the rated values. We can find the speed for that case in the datasheet, n = 343.4 rpm, so we need to know the value of the physical frequency or period:

1º) Pass the rpm to revolution per second.

$$n = \frac{343.5}{60} = 5.75\ rev/second$$

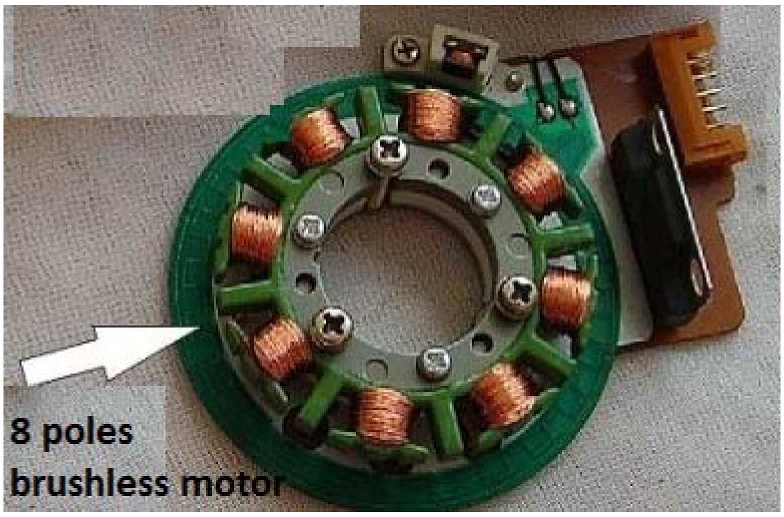
2º) We have 5.75 revolutions per second, so we can know how many time spend in one revolution, the inverse is the period of a physical revolution. Tp = 174,67 ms.

3º) In the picture of the hall sensors test, we can measure the period of a electrical cycle, if each square is 5 m/s, and one cycle spend 4 squares and a quarter of other square, the period of the cycle is Te= 21.5 ms

4º) The ratio between the physical period and the electrical period is the number of poles.

$$n^{\underline{o}}\,poles = \frac{174{,}67}{21{,}5} = 8{,}12$$

So we can conclude that our motor has 4 pair of poles.



7. CONTROLLER AND POWER ELECTRONIC

7.1. CONTROLLER

7.1.1. Introduction

The controller is one of the most important things in our project. It's a device which doesn't handle high values of voltages and current, but has to process a lot of information encoded in signals, in order to get the desired behaviour of our motor.

The controller must have the following features:

-Furnish the correct voltage according to desired speed

-Feed the windings in the correct sequence

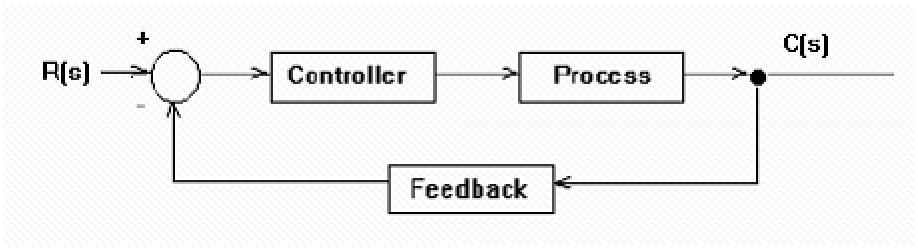
-Acquire the measured speed

After discussing about which kind of controller was more suitable for our project, we decided to implement a digital controller. In order to show the improvements that this kind of controller provides us, we are going to show a bit of information about analogic controllers as well.

7.1.2. Analogic

- ***Closed loop control***

As we can see in the following image the main characteristic is that a feedback is used to control the process. The feedback is measured by a sensor which send a signal to the controller.



The systems are shown as black boxes where you can check the input and the output, that's called transfer function. Working in the Laplace domain you can make a mathematical model of any process.

- PID controller

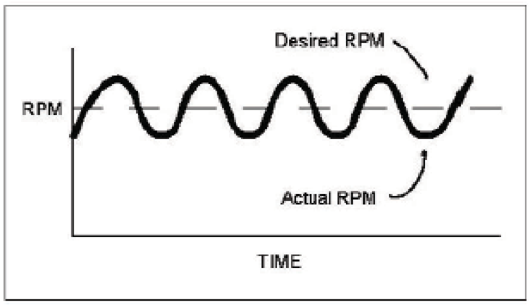
The PID controller is the most extended one in the industrial sector. The controller transfer function is shown below:

$$R(s) = k_p + \frac{k_i}{s} + k_d \cdot s$$

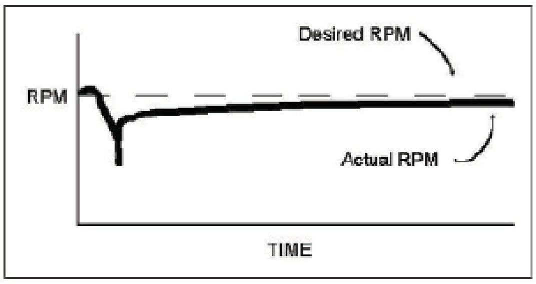
According to this parametres, we can make that our system get a desired behavior. The proportional part use to make around the 80% of the task, the integral part make around the 15% of the task and the derivative part the rest.

The proportional part use to fix the error between the desired value and the measured value (feedback) while the integral part takes care of the desired response time. To fix this values we always have to take care about the stability of the system. If we need to make the system stable we should increase the derivative part.

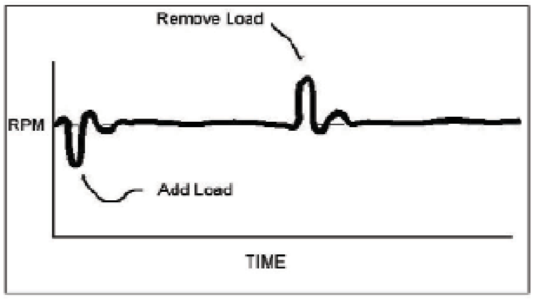
Now, we show some images of the responses that we can get from a regulator according to the parameters described before:



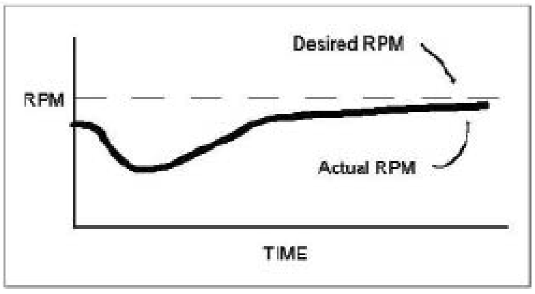
Very high PID gains



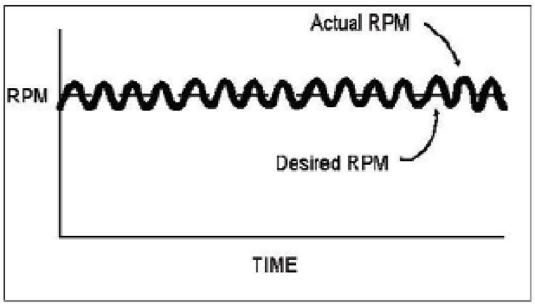
Low integral gain



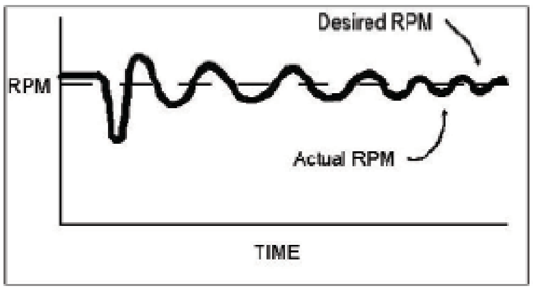
Desired response



Very low PID gains



Very high derivative gain



Very high integral gain and very low

As we can see in the images, the kind of signal that we can get is changing with the gains, so is important to choose a suitable PID gains.

There are many ways to implement this kind of controllers;

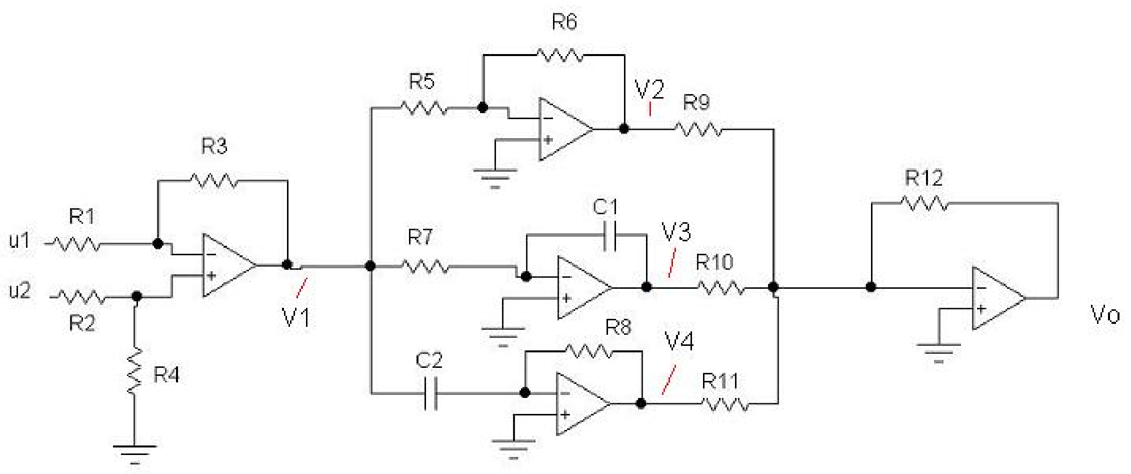
We can clasify them according to the kind of electronics used: analogic and digital controllers.

In the next sub-chapters, we are going to see what they consist on, in order to decide which is the more suitable for this project.

Now, we explain the analogic controller.

- ***PID analogic circuit***

The PID transfer function and the difference between the two outputs can be represented with the following electronic circuit:



The two inputs (u1 and u2) are voltage proportional to the speed, according to the used sensor gain.

The first operational amplifier works as a substractor, which represent the substractor drawn in the transfer function.

$$V_1 = U_2 \cdot \frac{R_4}{R_2 + R_4} \cdot \frac{R_3 + R_1}{R_1} - U_1 \cdot \frac{R_3}{R_1}$$

This voltage is connected to three operational amplifiers which work as an inverter, integrator and derivator, respectively:

$$V_2 = -\frac{R_6}{R_5} \cdot V_1$$

$$V_3 = -\frac{1}{R_7 \cdot C_1} \cdot \int_0^t V_1(t) \cdot dt$$

$$V_4 = -R_8 \cdot C_2 \cdot \frac{dV_1(t)}{dt}$$

The last operational amplifier works as an adder.

$$V_o = -R_{12} \cdot \left(\frac{V_2}{R_9} + \frac{V_3}{R_{10}} + \frac{V_4}{R_{11}}\right)$$

So:

$$V_o = R_{12}\left(\frac{R_6}{R_7} \cdot V_1(t) + \frac{1}{R_{10} \cdot R_7 \cdot C_1} \cdot \int_0^t V_1(t)dt + \frac{R_8 \cdot C_2}{R_{10}} \cdot \frac{dV_1(t)}{dt}\right.$$

$$V_o = R_{12} \cdot \left(\frac{R_6}{R_5 \cdot R_9} + \frac{1}{C_1 \cdot R_7 \cdot R_{10}} + \frac{C_2 \cdot R_8}{R_{11}}\right) \cdot \left(V_2 \cdot \frac{R_4}{R_2 + R_4} \cdot \frac{R_3 + R_1}{R_1} - V_1 \cdot \frac{R_3}{R_1}\right)$$

As we can see, the circuit represent the behaviour of a PID, where its constants can be adjusted with capacitors and resistances.

$$K_p = \frac{R_6}{R_5 \cdot R_9}$$

$$K_i = \frac{1}{C_1 \cdot R_7 \cdot R_{10}}$$

$$K_d = \frac{C_2 \cdot R_8}{R_{11}}$$

The analogic output have a voltage signal, so in order to produce a PWM we need to convert the output, using a divice like LM331 or AD650.

Finally, with this kind controller we can't use the hall sensor signal in a simple way, looking a sensorless driver a easier way, but we aren't going to get such a smooth performance as we can get using the hall sensor. Also another analogic sensor we'll be needed.

On the other hand, we have the digital controller, which offer us a simpler circuit, with the disadvantage of a complex programming. It has enough input to handle the signals from the three hall sensor and the throttle, and can measure the speed without any additional sensor, so the budget is also something that bet for the analogic way.

7.1.3 Digital

Digital control is a very different approach from the analogue way, because it allows you to actively manage the system. Also, it obtains more flexibility, more opportunity to manage the system and more safety in case of some fault of the system, like overcurrent or overvoltage.

The purposes of the digital controller are:

-Closed-loop speed control for precise speed regulation

-Read set speed (Speed cruise control)

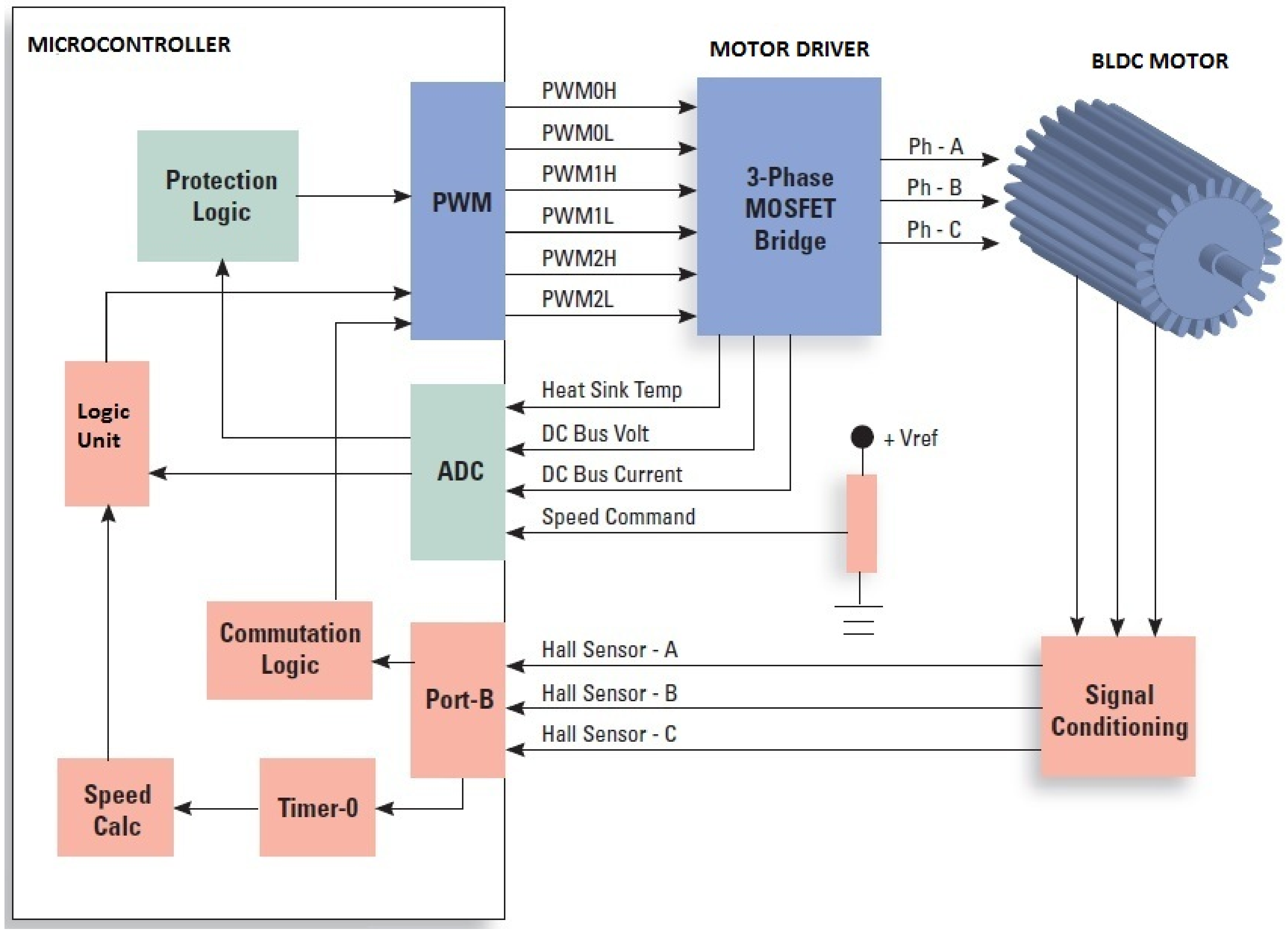
-Implement the cruise control

-Read the position of rotor with the hall sensor and measure the speed

-Logic protection for over-voltage, over-current and thermal protection

To realize this digital controller we need a microcontroller with a driver, that it's performed to control a brushless DC motor (BLDC).

The following figure represents the diagram block of the Digital system:



The diagram block is composed by a microprocessor, the 3-phase Mosfet Bridge (driver), signal conditioning for the hall sensor and the throttle.

The design involves running the BLDC motor in a closed loop, with the speed set by the throttle, that it's realized with a potentiometer that give an output of 0-5V. This signal is read by the Analog-Digital Converter (ADC) of the micro that converts the input voltage in a digital value that represents the set speed.

This value is read by the logic unit that converts this value in a proportional signal that is sent to the PWM module.

The PWM (pulse width modulation) is a type of digital modulation, which allows for a medium voltage variable dependent on the relationship between the duration of the positive and 0V. The duration of each pulse can be expressed in relation to the period between two successive pulses, implying the concept of duty cycle. In this module, will be set a duty cycle proportional to the desirable speed.

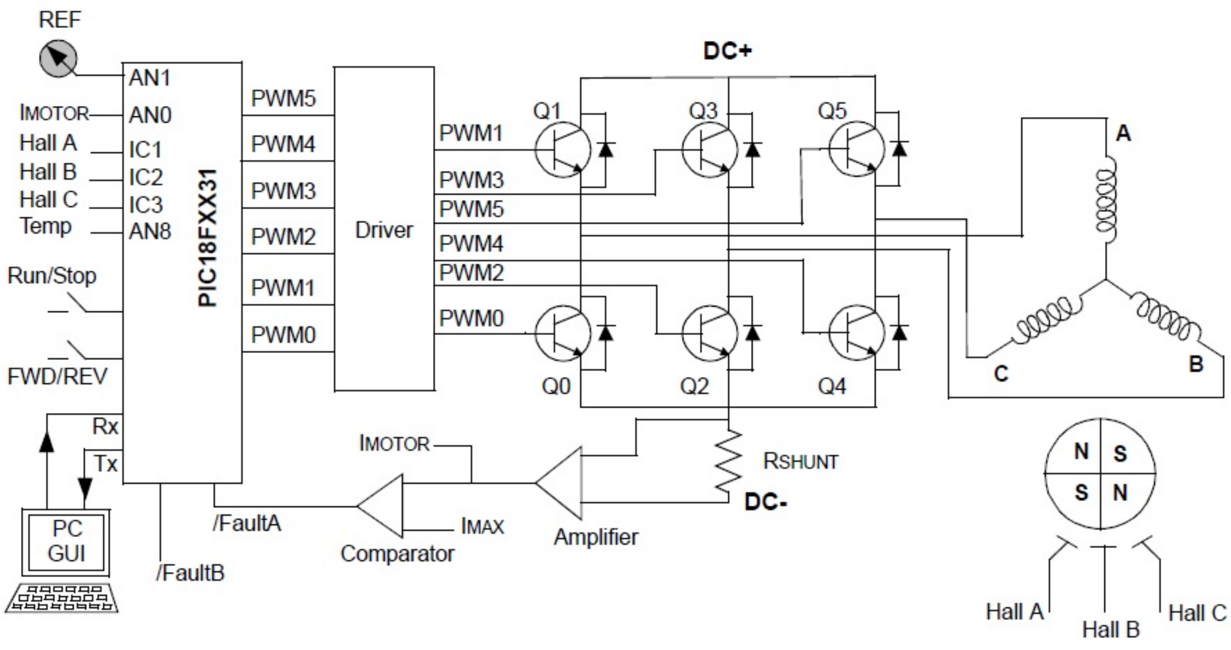
The output signals of the PWM module, controls the motor driver that has to energize the correct winding of the BLDC motor. The BLDC motor have 3 Hall sensors embedded that provides either a High or Low output based on the polarity of magnetic pole close to it. Rotor position is determined by analyzing the outputs of all three Hall sensors that generate in the micro an interrupt. Based on the output from hall sensors, the voltages to the motor's three phases are switched with the commutation logic unit.

Also, this interrupts are counted to measure the motor speed. Other peripheral functions (heat sink temperature and DC bus current) are used to protect the system in case of overload, under-voltage, and over-temperature.

For this application, we choose the PIC18F2431 microcontroller of the Microchip because it has peripherals that are suitable for motor control applications.

About the schematic, we are headed to the application note (*) of the Microchip to create the system, because it furnish a lot of information about how to develop the system with the hardware and firmware.

This is the control block diagram for controlling a BLDC motor with the controller PIC18F2431.



Control block diagram for controlling a BLDC motor

In the input of the micro controller, there are the Hall sensor input, the ref (potentiometer), input for run and stop the motor and for the forward/reverse direction of the motor. Also, we have to opportunity to send or receive some data to the PC with USART (serial port) module.

The Fault pin is linked to a circuit that in case of Overcurrent, Overvoltage and Over temperature, furnishes a voltage that indicates to the micro that one of these and disables automatically all the PWM outputs.

The PWM outputs from the PIC control the power switches Q0 to Q5. A matching driver circuit should be used for supplying the required gate current drive for the power switches.

A sequence table is entered in the program memory according to the type of Hall Sensor placement. The sequence may be different for clockwise and counterclockwise rotations. Each commutation sequence has two of three phases connected across the power supply and the third phase is left open. Using PWMs, the average voltage supplied across the windings can be controlled, thus controlling the speed.

In table 1 and 2, it’s shown the sequence of the brushless DC Hall sensor and which of all sensors have to be activated.

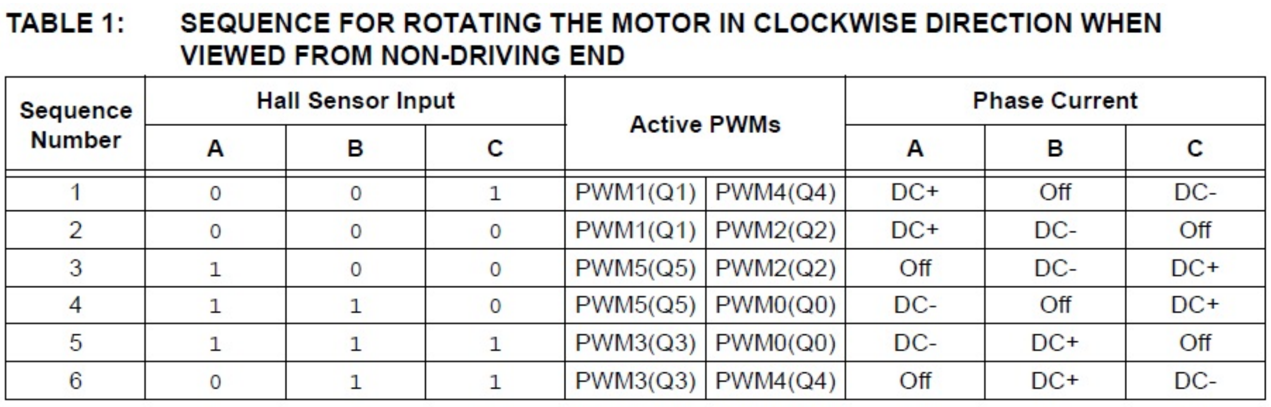


TABLE 1: SEQUENCE FOR ROTATING THE MOTOR IN CLOCKWISE DIRECTION WHEN VIEWED FROM NON-DRIVING END

Sequence Number	Hall Sensor Input			Active PWMs		Phase Current		
	A	B	C			A	B	C
1	0	0	1	PWM1(Q1)	PWM4(Q4)	DC+	Off	DC-
2	0	0	0	PWM1(Q1)	PWM2(Q2)	DC+	DC-	Off
3	1	0	0	PWM5(Q5)	PWM2(Q2)	Off	DC-	DC+
4	1	1	0	PWM5(Q5)	PWM0(Q0)	DC-	Off	DC+
5	1	1	1	PWM3(Q3)	PWM0(Q0)	DC-	DC+	Off
6	0	1	1	PWM3(Q3)	PWM4(Q4)	Off	DC+	DC-

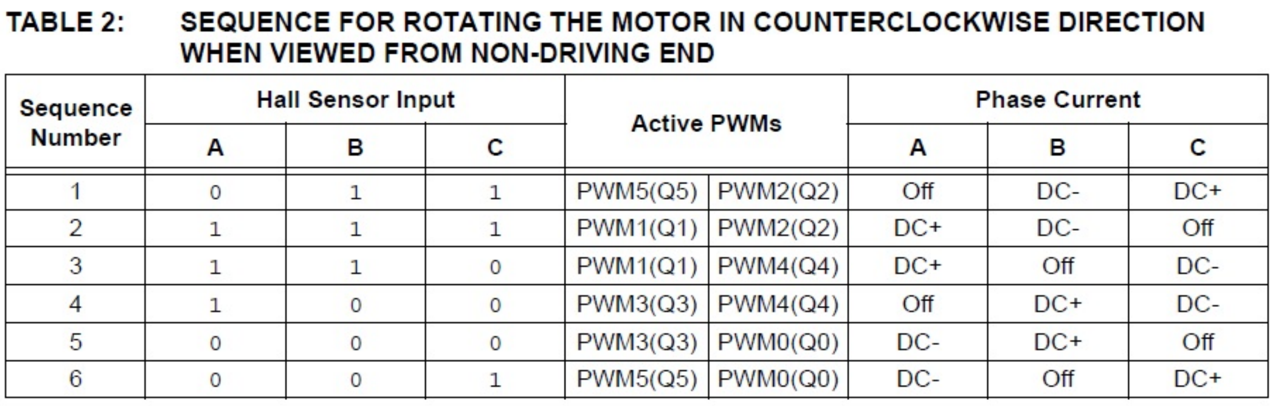


TABLE 2: SEQUENCE FOR ROTATING THE MOTOR IN COUNTERCLOCKWISE DIRECTION WHEN VIEWED FROM NON-DRIVING END

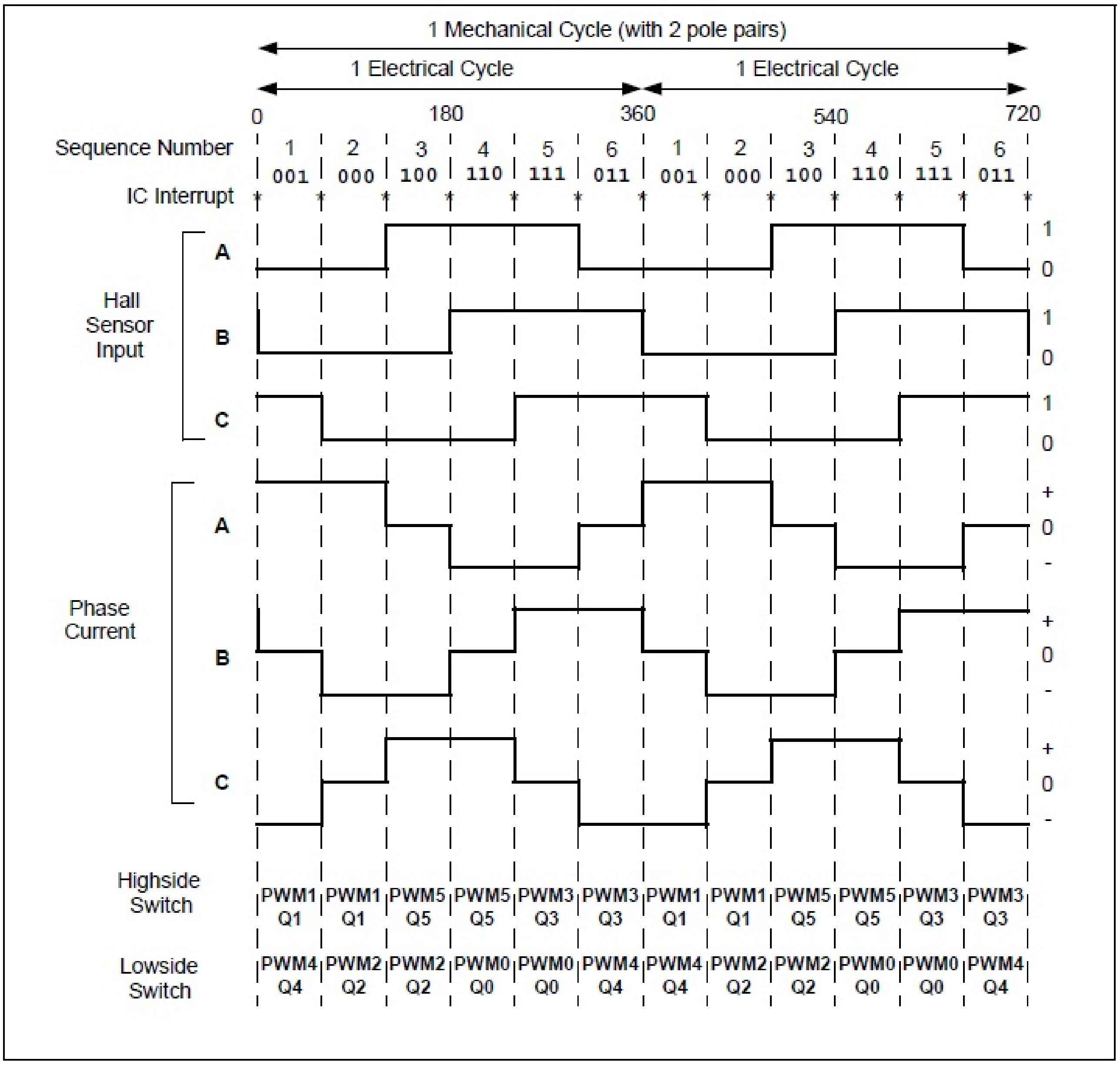
Sequence Number	Hall Sensor Input			Active PWMs		Phase Current		
	A	B	C			A	B	C
1	0	1	1	PWM5(Q5)	PWM2(Q2)	Off	DC-	DC+
2	1	1	1	PWM1(Q1)	PWM2(Q2)	DC+	DC-	Off
3	1	1	0	PWM1(Q1)	PWM4(Q4)	DC+	Off	DC-
4	1	0	0	PWM3(Q3)	PWM4(Q4)	Off	DC+	DC-
5	0	0	0	PWM3(Q3)	PWM0(Q0)	DC-	DC+	Off
6	0	0	1	PWM5(Q5)	PWM0(Q0)	DC-	Off	DC+

The Table 1 shows a typical switching sequence used to run the motor in the clockwise direction and Table 2 shows the counterclockwise sequence.

In one of the tests of the motor, we verified that this motor runs in the counterclockwise sequence.

The following graph shows the draw of the sequence in the clockwise direction with its phase current:

HALL SENSOR INPUT VERSUS PHASE CURRENT



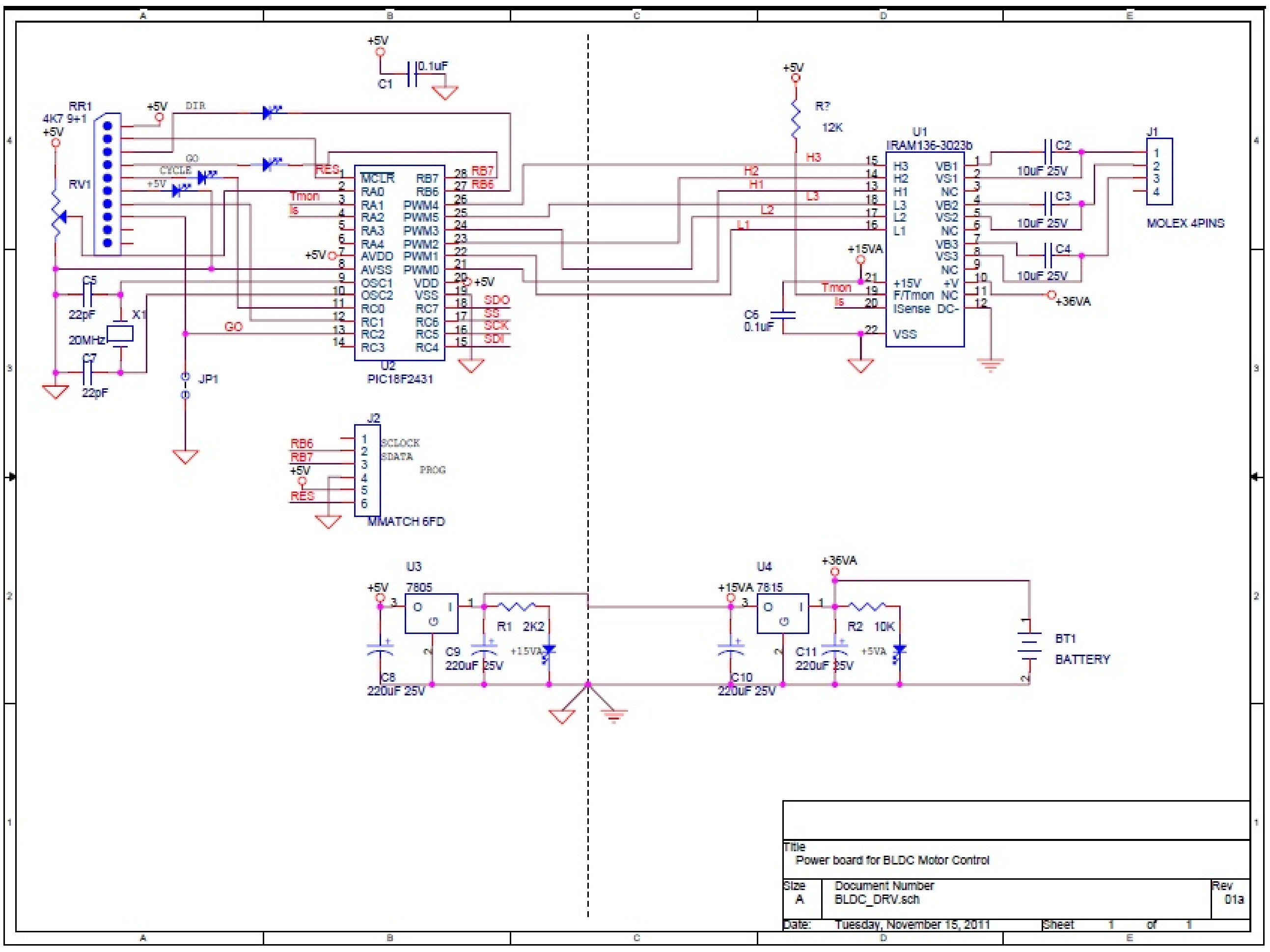
With all this information, we design an electrical schematic for this application, like we can see in the next figure.

The schematic is composed by three parts:

-Logical

-Power supply

-Motor driver



Electrical schematic “Cruise control”

Logical circuit

The logical circuit is composed by the microcontroller PIC18F2431, a crystal, the interface for the potentiometer, the interface for programming and some test led.

This microcontroller provides the features described in the following table:

Characteristics	PIC 18F2431
Operating frequencies	DC-40MHz
Flash Memory(Byte)	16K
EEPROM Memory(Byte)	256
Timer	4
PWM	3 PWM channels with complementary outputs
Converter A/D	10bit 200Ksps 5 channels
Motion feedback module	Input Capture Channels
Serial comunication	USART e SPI

Table: PIC18F2431 features

The microcontroller has to perform all the purpose of the controller, which will be implementing in the firmware.

The firmware is written in C language that is made with the CCS compiler. The code is transferred to the micro with the MPLAB-ICD2 programmer of Microchip, that is plugged with a cable to the interface of the programming on the board. The interface for programming is realized with the specification of the MPLAB datasheet. After the programming, the code is stored in the micro and when the supply voltage is furnished to the micro, the firmware starts to run.

The micro works at the frequency of 20Mhz, that is provided by a crystal.

The throttle is a variable resistance that furnishes a voltage signal proportional to the set resistance.

The interface of the throttle is composed by a connector that furnishes 5v, middle signal and gnd. The middle signal is an input to the microcontroller that is inserted in one of the pins that implements the ADC converter.

The read voltage (0-5V) by the ADC is converted in a digital value that is proportional to the speed for the Cruise Control.

This value is important because it sets the relative average current that flow in the motor. So, this is realized with the PWM signal of the microcontroller.

This micro has 3 PWM modules, where every module is compose by 2 PWM signal (one direct and one in complementary mode). In total, the micro has 6 PWM signal.

The micro uses this signal to control the driver of the power electronics and finally the motor.

Then, the micro has to turn on or off the correct PWM signal according to the relative sequence of the hall sensor. This PWM signal is a signal with a variable average voltage, dependent on the relationship between the duration of the high logic level (5V) and low logic level (0V). The switching frequency is set to 20 KHz.

The duty cycle of the PWM is directly proportional to the throttle potentiometer input. The changes in the duty cycle control the current through the motor winding, thereby controlling motor torque.

This PWM signal is activated in a sequence that is given by the Hall sensors signal, which indicate the position of the rotor. An interrupt is generated when the input state on any pin changes. This signal is plugged in the board into a connector that provides 3 hall sensors signal (Ha, Hb and Hc), 5 volts and ground. Also, the hall sensor signal needs an interface of pull-up resistor to work.

So, when the motor is running, the state of the three Hall sensors changes according to the rotor position. The supply of each motor phase is switched based on the state of the sensors (commutation). Also, Hall sensors interrupts are counted to measure the motor speed that is calculated every second with an internal timer of the micro. This value represents the RPS (Revolution Per Second), that it's obtained with this formula:

$$\text{Speed in RPM} = \left(\frac{\text{Operating Frequency}/4}{\text{Timer5 Count x Timer5 Prescale x Number of Pole Pairs x 6}} \right) \text{x } 60$$

If it's multiply by 60, it's obtained in RPM (Round per Minute).

Similarly, the speed reference input is translated into a speed value in order to have both reference and feedback in the same platform. Next equation shows converting speed reference from a potentiometer setting read through an AD channel.

$$\text{Speed Reference} = \text{Rated Motor Speed x } \frac{\text{ADC Value}}{\text{Maximum ADC Value}}$$

In the next section, the calculation error is described in order to control the speed.

Also, this board manages the system in case of overload, under-voltage, and over-temperature with some signal taken from the driver.

Firmware implementation

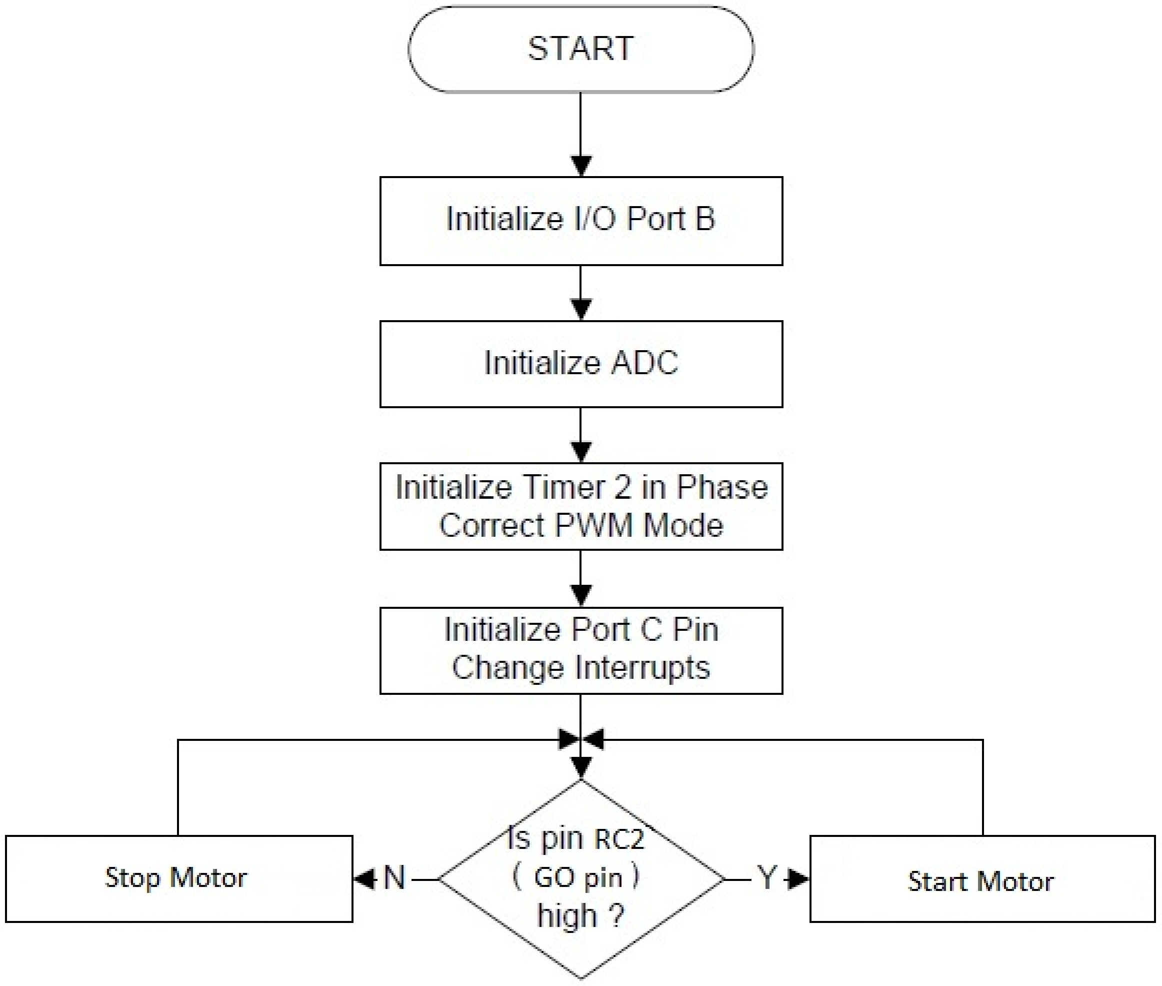
Generally, the firmware is a program that internally controls various electronic devices.

In this case, the firmware is inside the micro and controls all the peripherals of the micro and of the system. Then, it's the main part of the controller, because it describes how the system is working.

Inside the firmware, the closed-loop speed control is implemented, reducing the error between the speed set by the potentiometer and the actual motor speed. The output of the loop changes the duty cycle of the PWM module, thereby changing the average voltage to the motor.

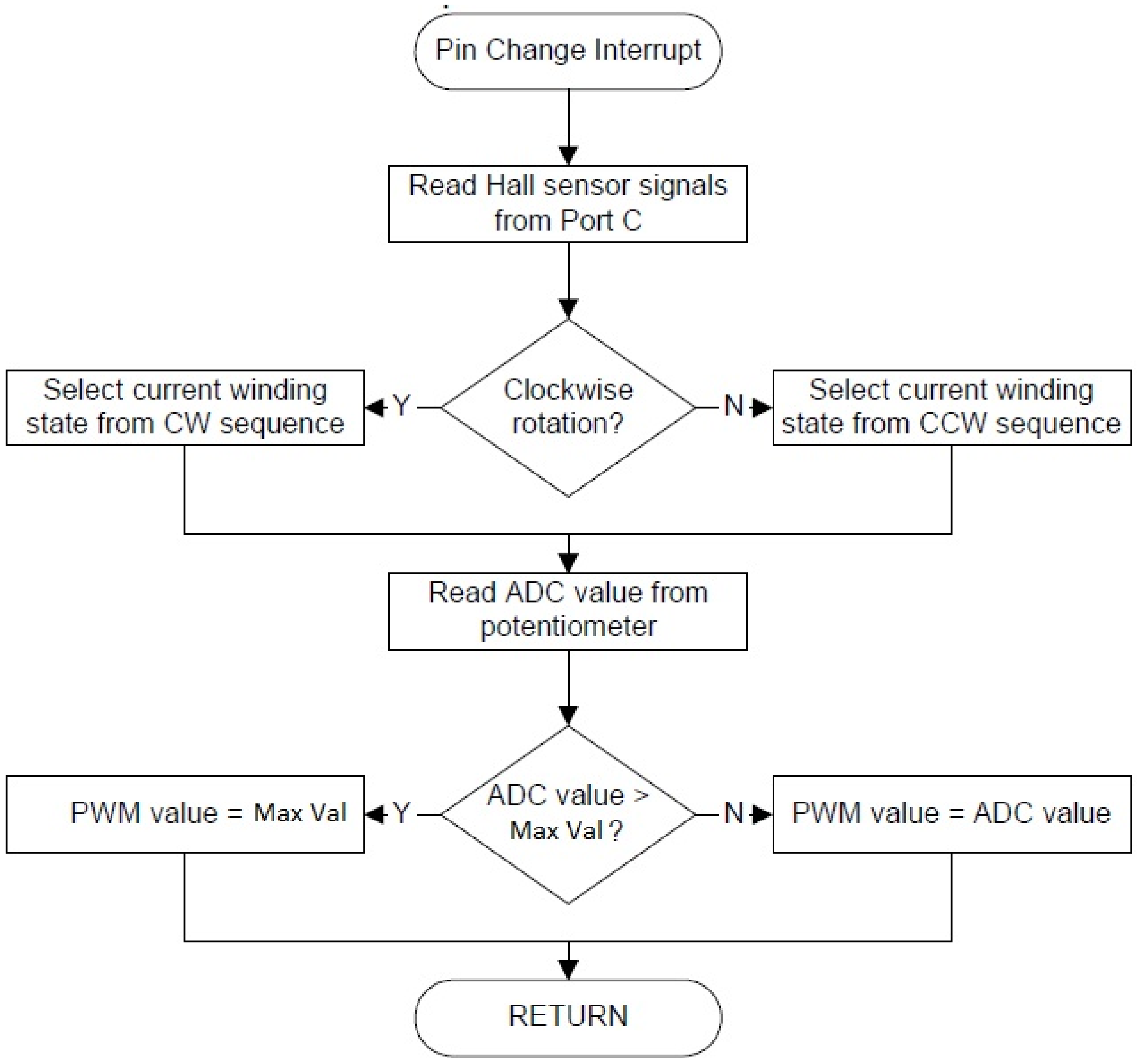
The firmware is composed by three parts:

- Main loop
- Interrupt Handler of Hall sensor
- Speed error calculation(Cruise control)


START
Initialize I/O Port B
Initialize ADC
Initialize Timer 2 in Phase Correct PWM Mode
Initialize Port C Pin Change Interrupts
Is pin RC2 (GO pin) high ?
N
Y
Stop Motor
Start Motor

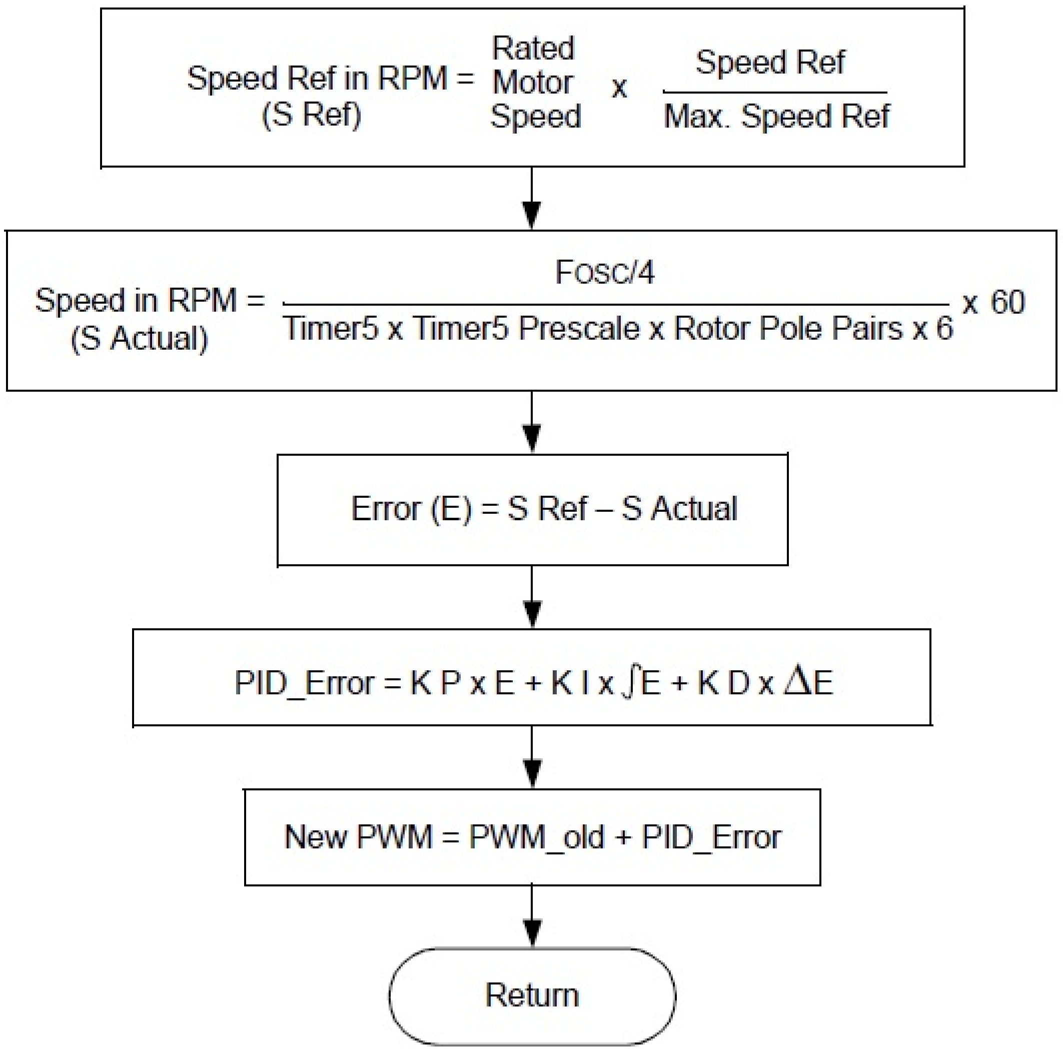
Flowchart of Main loop

The main loop flowchart describes the main function of the firmware, that execute every cycle of the code. The initial part is for the initialization of the micro. One of the most important initializations regards the routine of the interrupt that detects the change of the state of the hall sensor signal. The next step in the loop is checking the state of the pin RC2, that it's linked to the jumper or a switch. When the jumper or the switch is closed, the motor start to run at the set speed that is read from the ADC.


Pin Change Interrupt
Read Hall sensor signals from Port C
Clockwise rotation?
Y
Select current winding state from CW sequence
N
Select current winding state from CCW sequence
Read ADC value from potentiometer
ADC value > Max Val ?
Y
PWM value = Max Val
N
PWM value = ADC value
RETURN

Flowchart of Interrupt Handler of Hall sensor

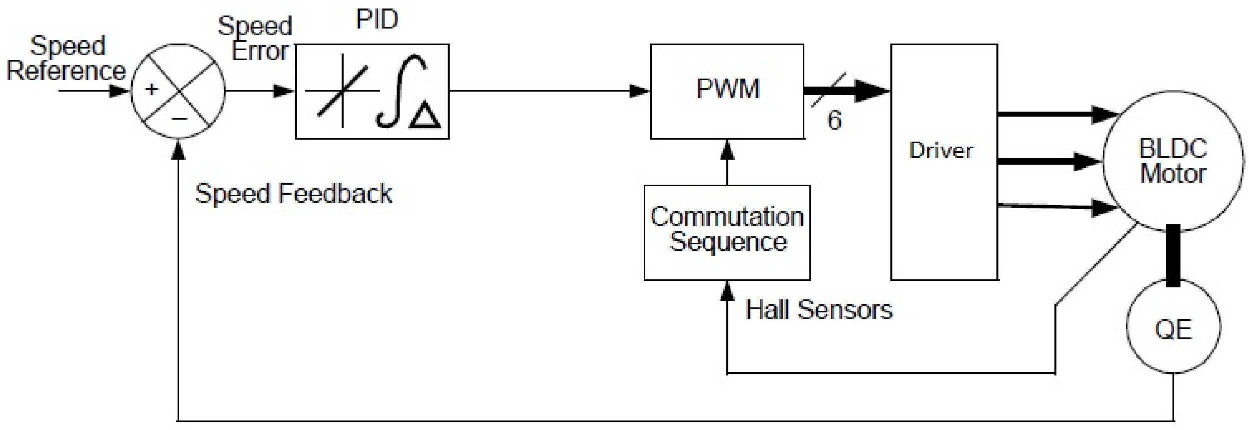
This flowchart explains how the Interrupt of the Hall sensor works. The interrupt has a special function that it’s executing (in this case), when one signal of the three Hall sensor changes its state. This function is very important because, it’s possible to energize the correct winding, according to the actual state of the Hall sensor (position of the rotor). Also, this routine sets the actual speed that is read by the ADC.


Speed Ref in RPM = Rated Motor Speed x Speed Ref / Max. Speed Ref
(S Ref)
Speed in RPM = Fosc/4 / Timer5 x Timer5 Prescale x Rotor Pole Pairs x 6 x 60
(S Actual)
Error (E) = S Ref – S Actual
PID_Error = K P x E + K I x ∫E + K D x ΔE
New PWM = PWM_old + PID_Error
Return

Flowchart of Speed error calculation

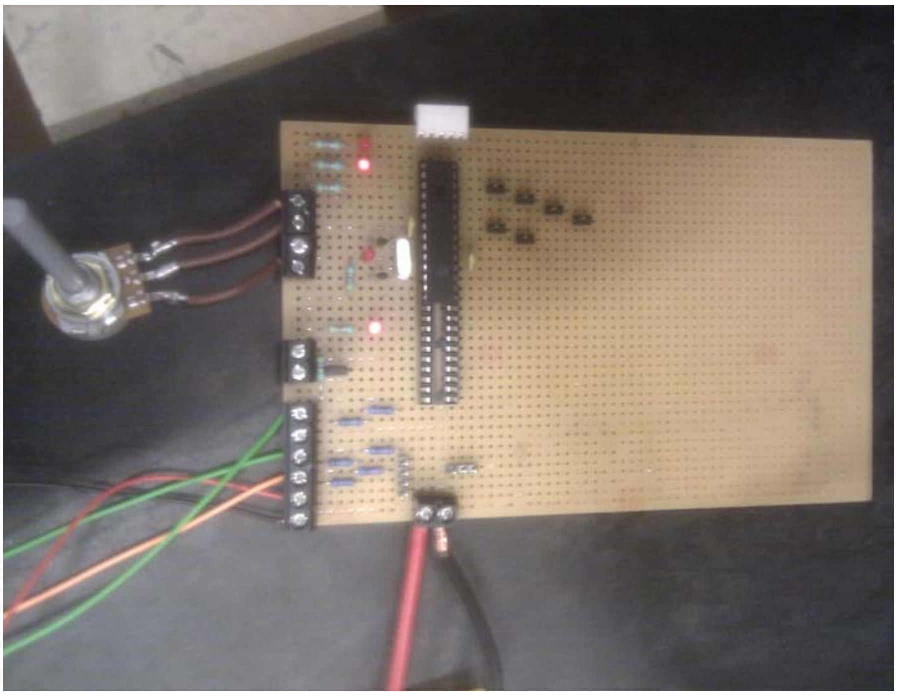
The Speed error calculation is the function that implements the Cruise Control. This function calculates the error from the set speed of the throttle and the actual speed. The error may be positive or negative, indicating if the speed is more or less than the set speed. This error is passed through a PID algorithm to amplify the error. The amplified error is used to readjust the PWM duty cycles originally calculated.

Then, now it's possible to describe the system with a block diagram of a control loop with a PID algorithm.



Block diagram of a control loop with a PID algorithm

Realization and test

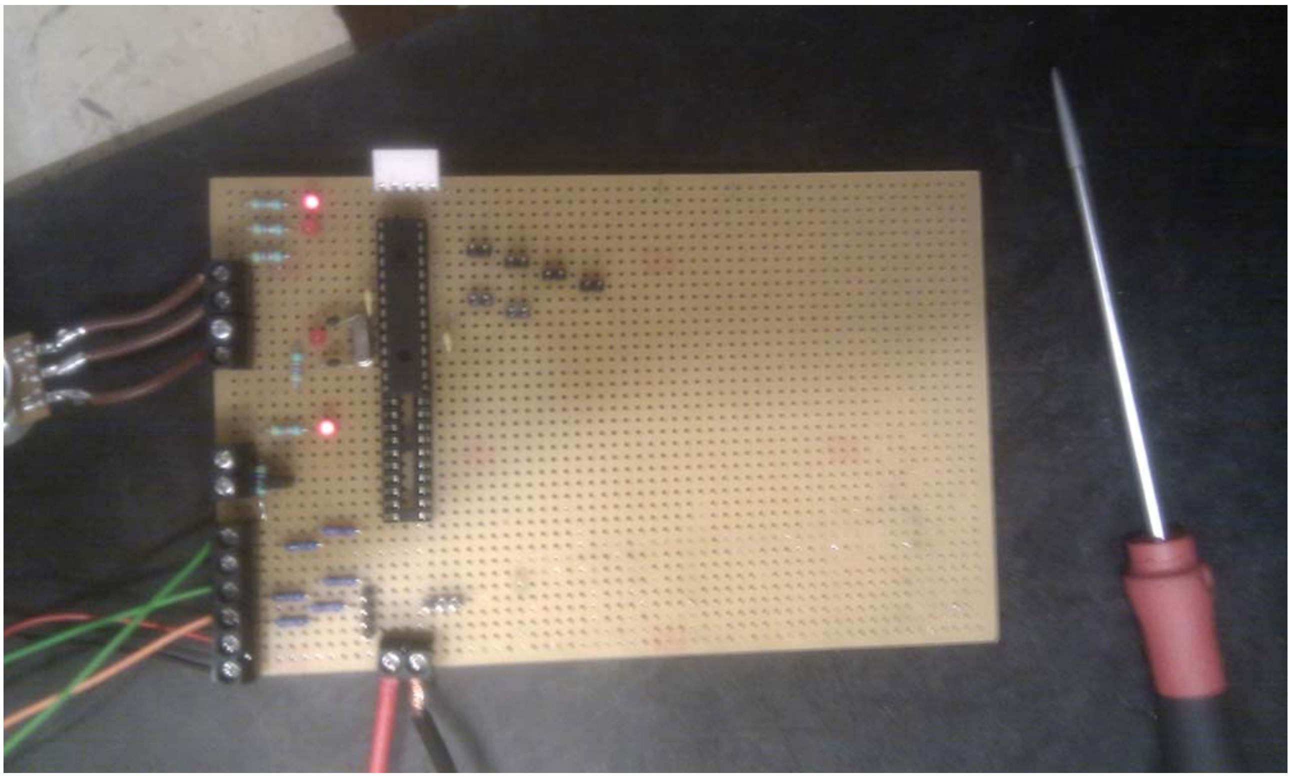


The Logical circuit is realized in a prototype board, that it's supplied by the power supply part with 5V. The microcontroller is programmed with the firmware, described in the flowchart of the main loop and of the Interrupt Handler of Hall sensor, but not about Flowchart of Speed error calculation. It means that, the microcontroller executes the part of initialization and acquires the change of the state of the Hall sensor, with the activation of the relative PWM.

The Speed error calculation is not implemented because we didn´t have enough time to make this part. So, that means that the Cruise Control is not implemented. But, we will explain how to implement this part.

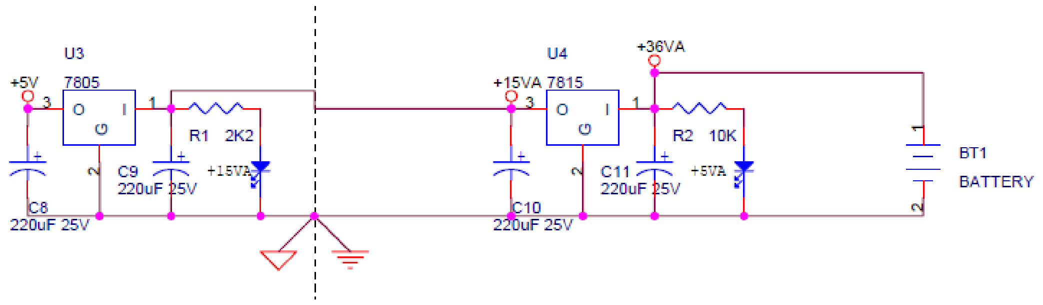
Before activating the board, we realized the test of isolation and continuity to ensure that there isn't any short circuit or any wrong connection.

In the next photo, it's possible to see how the Hall sensor is working. The led indicates the actual sequence of the hall sensor, where the first led in the top is Hall sensor A, the second in the middle is B, the third is Hall sensor is C and the last indicates the 5V in the board. In this case, there is the sequence 101.



7.2 POWER ELECTRONICS

7.2.1 Power supply



This is the electrical circuit of the power source. This circuit is very important because without this, all the electronic part would not work correctly. The main goal of this circuit is to supply the correct voltage to the rest of the electronic board.

As we know, our board is divided in 3 differentiated parts (the power supply, the logical part, and the driver for the motor).

All these 3 parts work at different voltages and we don´t have in the circuit, 3 different variable power sources to supply each of the circuits. This is the reason for what we need this power supply circuit.

We only dispose one battery. This one will work as our power supply and it will give us 36 Volts. With this reference voltage we should get another 2 different voltages.

The logical part where the microcontroller is connected is going to work with 5 volts. We have to be very careful with these values because a high voltage can burn and destroy all the electronic part. The driver section will need a voltage of 15 volts

To get these voltages we are going to use 2 regulators. The regulator has 3 pins, Input, Ground and output. With these regulators, whichever voltage enters from the input, the regulator processes it and takes out through the output, the desire voltage. In our case, how we mentioned before, we need 15 V and 5V.

The LED diodes are on the circuit just to indicate us that there is current and voltage flowing through the circuit.

This circuit also needs some electrolytic capacitors. Their mission is to stabilize the voltage that goes to the regulator. We need to maintain the voltage that enters into the regulator to a fix values and sometimes the source (battery, common electric network) has voltage peaks that can burn the regulator and the capacitors maintain the level of the voltage in a stabilized value.

Realization:

To implement this part of the circuit we have to be very attentive to all the components that we use and the values and the directions that we introduce them in the board

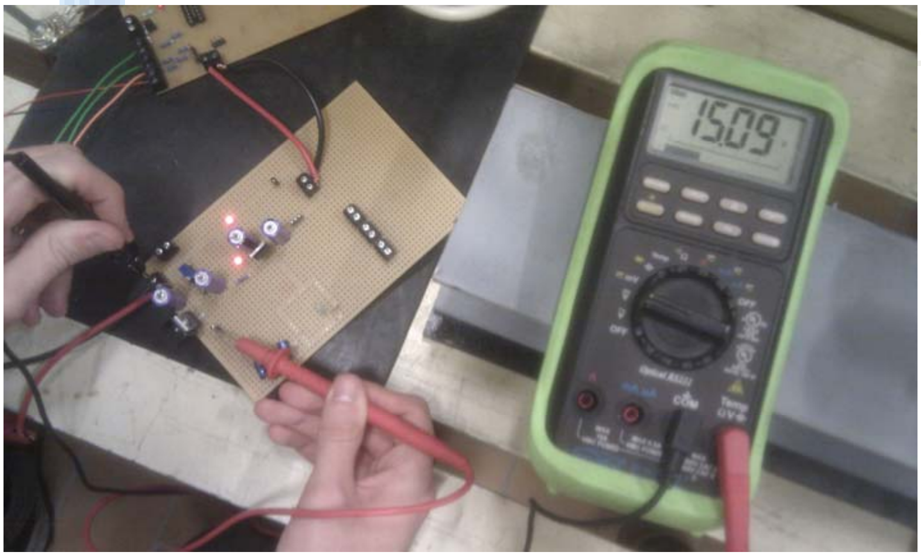
About the regulators, we had some problems fixing and deciding them. Concretely the one which supplies the 15 voltage at the output has a problem because the 36 volts that come straight from the battery will arrive directly to the regulator.

Looking at the datasheet of the first regulator that was chosen (7815) we realized that this one was not going to be correct because the maximum allowance voltage of this regulator is 35 V and just the battery gives minimum 36 V and it can give more when is fully charged.

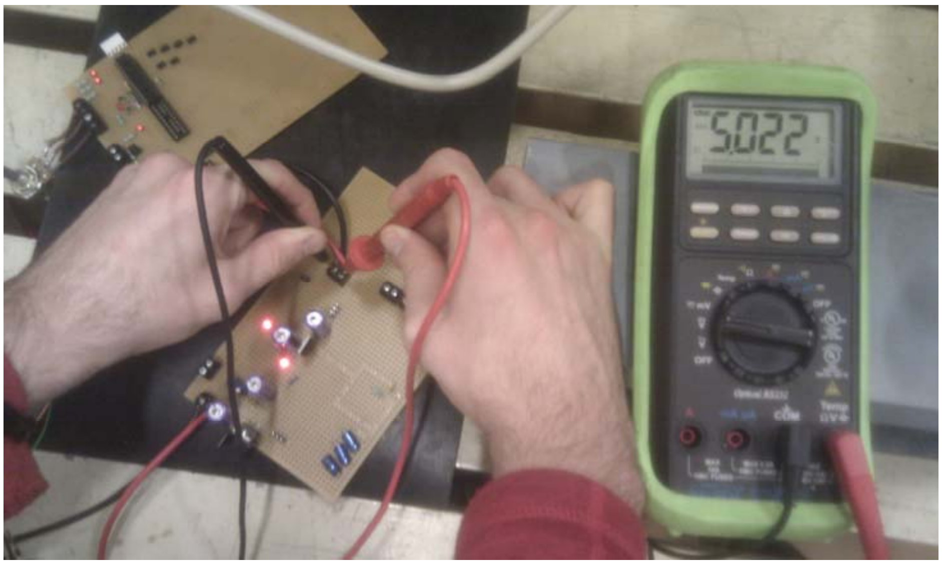
To solve this we have to look for another regulator that bears high voltage. We found the regulator LM317HV that has the capacity of handling high voltage (60 volts).

Another factor to take into account is the polarity of the electrolytic capacitor. It is very important to put the capacitor in a right way because a problem with this, can automatically burn the components.

In this image we can see the voltage that the battery gives us. How we said before it is not exactly 36 Volts. Normally this should not be a problem, but we had it because our regulator accepted maximum 36 Volts. Because of this reason we had to change it and put another one that could bear more voltage.



In this picture we are testing the output of the first regulator when we put the voltage of the battery into the power circuit (40 volts). We can see that is practically 15 V. How we mentioned, this voltage will go straight to the driver.



In this picture we can see the output of the other regulator. It is practically 5 Volts. After getting this voltage, we will pass it straight to the logical part.

7.2.2. Driver

There are two basic types of Brushless DC motors, sensored and sensorless. It is important to know the position of the rotor in order to energize the correct winding, so some method of detecting the motor position is required. A sensored motor will directly report to the controller, the actual position of the motor. Driving a sensored motor requires that the induced voltage in the un-driven winding be sensed and used to determinate the actual speed of the motor. Then, the next commutation pattern can be determined by a time delay from the previous pattern.

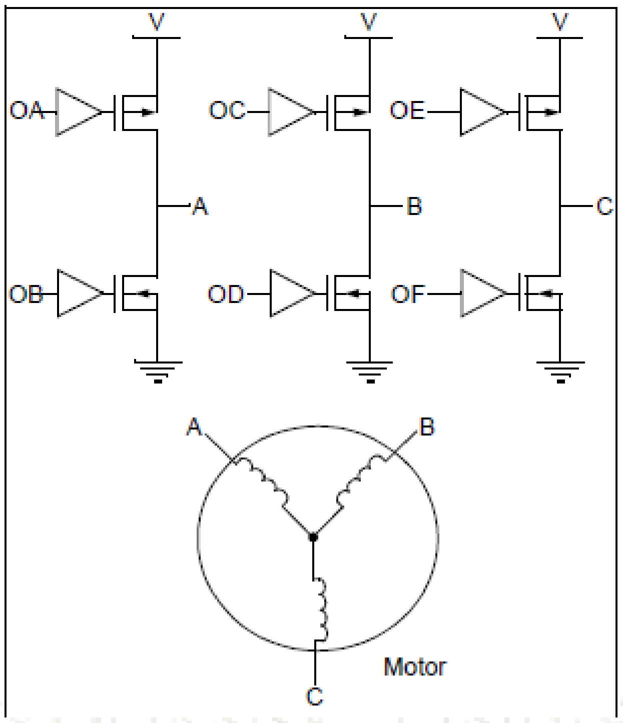
Sensorless motors are simpler to build due to the lack of the sensors, but they are more complicated to drive. A sensorless motor performs very well in applications that don´t require the motor to start and stop. A sensor motor would be a better choice in applications that must periodically stop the motor.

The motor cannot commutate the windings (switch the current flow), so the control circuit and software must control the current flow correctly to keep the motor turning smoothly.

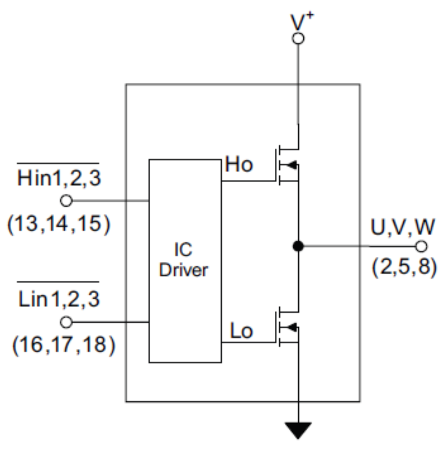
The driver circuit will be in charge of controlling and feeding the necessary windings of the motor. It will be the last processing step before the motor.

The driver receives a PWM signal from the micro and transfers it to the correct winding of the motor.

This driver is composed of a simple half-bridge of mosfet on each of the three motor windings. Here is a schematic of the circuit which is inside this driver.



We have chosen the driver model IRAM136-3023B, because this device has all the necessary circuit integrated in it. This device is composed by three half bridge circuit, that works with a input output logic level table like this:



I_{TRIP}	$\overline{\text{HIN1,2,3}}$	$\overline{\text{LIN1,2,3}}$	U,V,W
0	0	1	V^+
0	1	0	0
0	1	1	X
1	X	X	X

Here we can see that the input of the PWM has two types of signals, high and low. Beside is the table that tells us how it works. The output will be the signal that feeds the motor.

The signals are active low (they get activated when they get a 0). In the table we can see that if the H signal is 0 and the L signal is 1, the output will be V+ (36V).

If the H signal is 1 and the L signal is 0, then the driver will deliver an output where the current will flow to ground and the associated winding will not be energized.

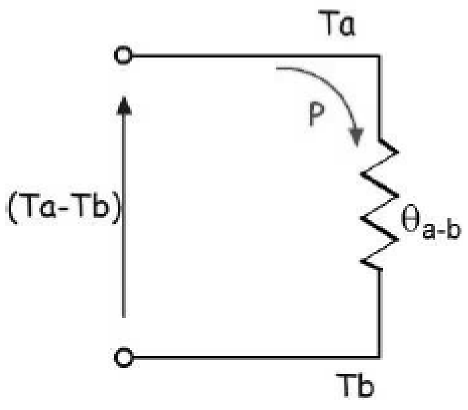
If both the signals are 0 (activated) then there is a mismatch because we would close all the circuit and the output would not have any sense.

Due to the high current and voltage that is going to flow through the driver, we will need to dissipate the heat in a way so our driver does not have any problems of overheating that could burn it.

To choose the heat sink which is suitable for our application we had to do some calculations.

The heat sink ability to dissipate heat to the ambient depends on many physical, geometrical and power factors and its determination is very complex. One technic is with a representation of the thermal resistance of the system.

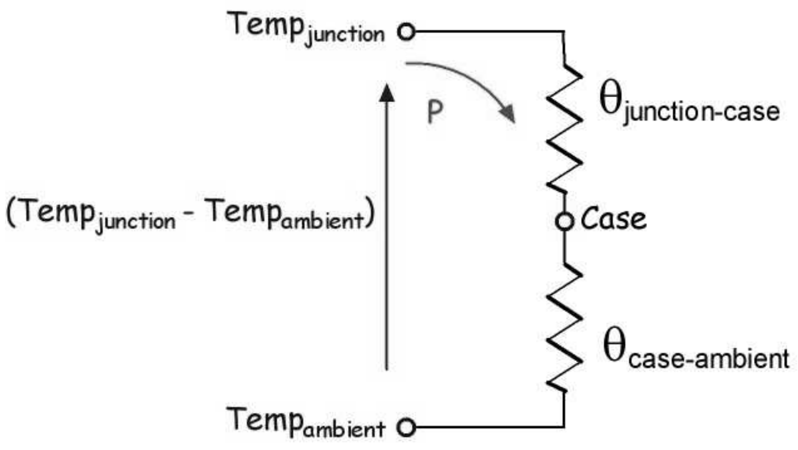
To dissipate the heat, we have to calculate the thermal resistance from point "a" to a point "b":



$$Temp_b - Temp_a = Power \bullet \theta_{a-b}$$

Where Θ is the terminal resistance from point to a to b, that indicates how many centigrade the temperature on point "a" will rise up over point "b" for each watt of power dissipated.

In this case, the equivalent circuit is compose to:



To dissipate the heat, we have to calculate the $+\Theta_{C-DIS}$ that represent the thermical resistance of the heat sink.

Below, there is calculation of the main parameters of the heat sink:

INPUT VOLTAGE = 36-41 V

MAXIMUM OUTPUT CURRENT= 11 A

P_D (Power dissipation mosfet)= 89W

$T_{junction}$ =150 ^oC

$R_{th(j-c)}$ =1.2 $^oC/W$

$R_{th(c-s)}$=0.1 $^oC/W$

$\Theta_{J-A} = \Theta_{J-C} + \Theta_{C-DIS}$

$T_J \geq T_a + P^* \Theta_{J-A}$

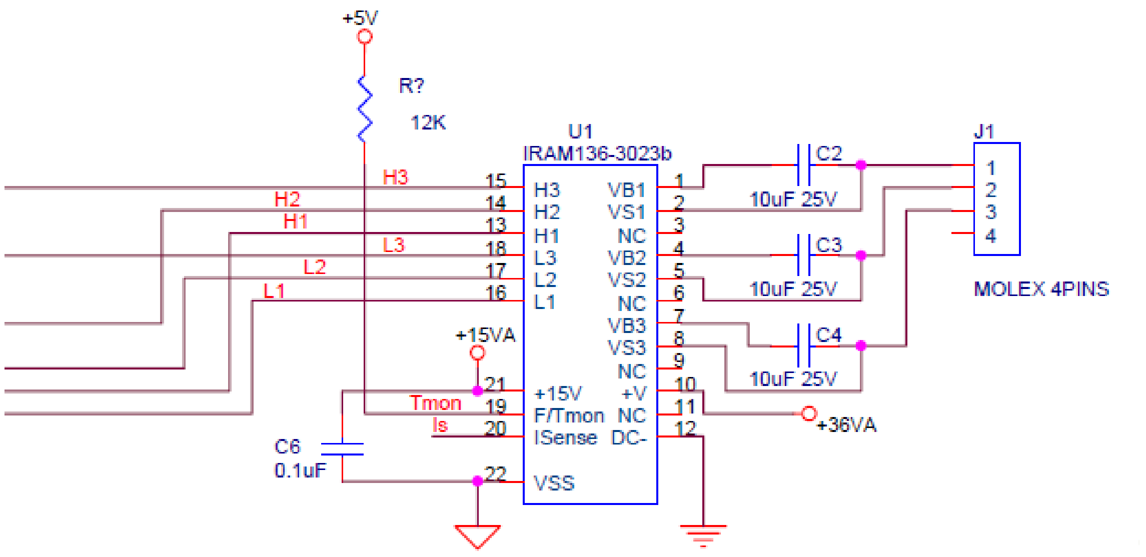
$150 \geq 25 + P * \Theta_{J-C} + P * \Theta_{C-DIS}$

$$\Theta_{C\text{-}DIS} \leq \frac{150 - 25 - P * \Theta_{JC}}{P} = \frac{150 - 89 * 1,2}{89}$$

$\Theta_{C\text{-}DIS} \leq 0,2\ ^{o}C/W$

In the image on the right side, we can see the heat dissipator that we should have attach to our driver.

Realization:



This is the circuit of the driver that we implemented. The pins from the 13 – 18 are the PWM signals that come from the microcontroller, high and low signals. Depending on this signal we feed the correspondent winding of the motor.

The driver we feed it with a voltage of 15 Volts that comes from the power supply. This voltage we put it in the pin 21.

The pin number 10 is connected directly to the battery (36 V). This voltage will directly go to the drain of the Mosfets and the output of these will be connected straight to the windings of the motor. The output pins are the first three.

Here we see the final mounting of the driver.

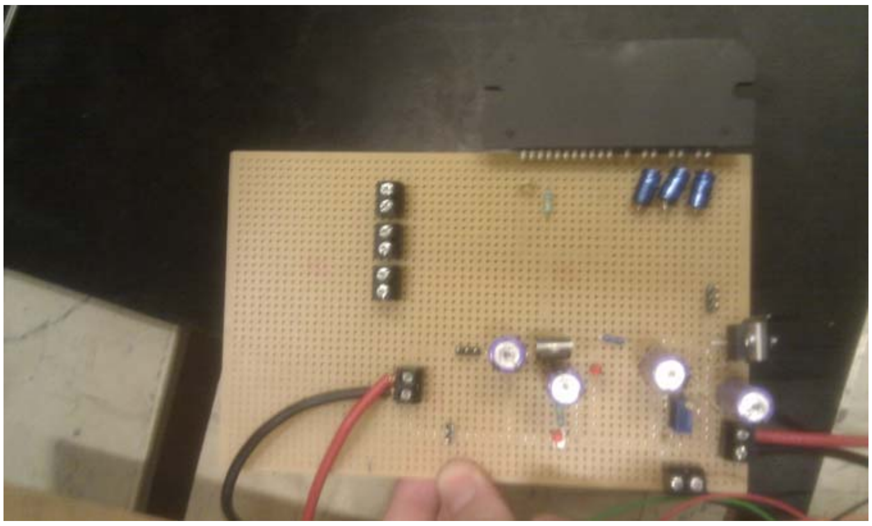


Fig. Picture of Power electronic board including driver

8. CONCLUSIONS

After explaining our project, it's time to take a look on our work: the achieved goals, working method and critical view.

Being able to take part in this project has allowed us to learn a lot about the topics involved in our task, completing the theoretical knowledge got in the lessons. Also how to work in group: having our own project room to deal about the topics, access to the lab and the component room and also can deal with the people from the workshop, who lead us in our work and got the needed components. All of this followed by being in contact with real life problems, such as the time, agree on the decisions and choices and so on.

As we saw along the report, the parts were well defined, so in order to get success in our task, we needed to deal with all those topics. The study of the motor help us to understand not only how this could work but also how to control it, and change the following electronic and controlling studying part before wasting time in a wrong idea.

Due to the size of the project we had some time problems, because we couldn't work on the realization until we got the whole study, devices information, and tests of the motor, and consequently, we couldn't build our electronic board as early as we would like.

The goals we have achieved are: understand the motor performance, how the hall sensor sequence works, get the desired and variable output PWM signal and feed correctly the windings of the motor according to the sequence with our own electronic board.

Also we have to take into account the goals that we couldn't achieve: test the driver, install the driver heat sink, program a generative break in order to make the motor work as a generator while it's breaking and charge our battery and build our circuit in a PCB board after testing the proto-board. Finally, in order to avoid noise from the power part in the logical part, we should have separated optically the two parts (we show the schematic in the annex).

Although we get along very well with the group, we can make a critical view about our working method, the way we focused the project wasn't the optimal one. In our opinion we should have made a better planning and tasks distribution so everybody would work not only the same, but faster and easier. If we recognize ourselves that despite the final work is done we made these mistakes, next time we may work so much better.

9. BIBLIOGRAPHY

AN857_Brushless DC Motor Control Made Easy

AN898_Determining MOSFET Driver Needs for Motor Drive Applications

AN899_Brushless DC Motor Control Using PIC18FXX31 MCUs

AN885_ Brushless DC (BLDC) Motor Fundamentals

DC Motor Control Tips 'n Tricks

Electric Bike BLDC Hub Motor Control Using the Z8FMC1600 MCU

AVR448: Control of High Voltage 3-Phase BLDC Motor

Electronic Circuits Desing For Beginners – Chapter 5

IRAM136-3023B_ Integrated Power Hybrid IC for Low Voltage Motor Applications

PIC18F2331/2431/4331/4431_ Microcontrollers with nanoWatt Technology, High-Performance PWM and A/D

BOOKS:

First course on Power electronics_ Ned Mohan, year 2003 Edition

Principles of Electrics Machines and Power Electronics_ P.C.Sen, Second Edition

Electrical Machines, Drives, and Power Systems_ Theodore Wildi, Fifth Edition

WEBSITES:

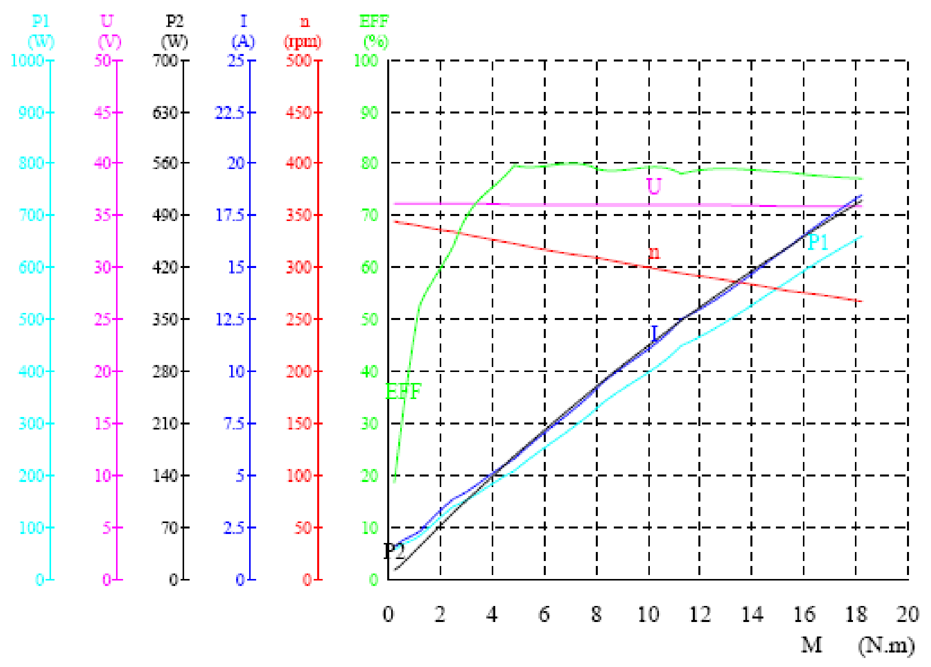
http://www.goldenmotor.com/hubmotors/HBS-36V500W%20Performance.pdf

http://bicicletaselectricas.wordpress.com/electronica-de-una-bicicleta-electrica/

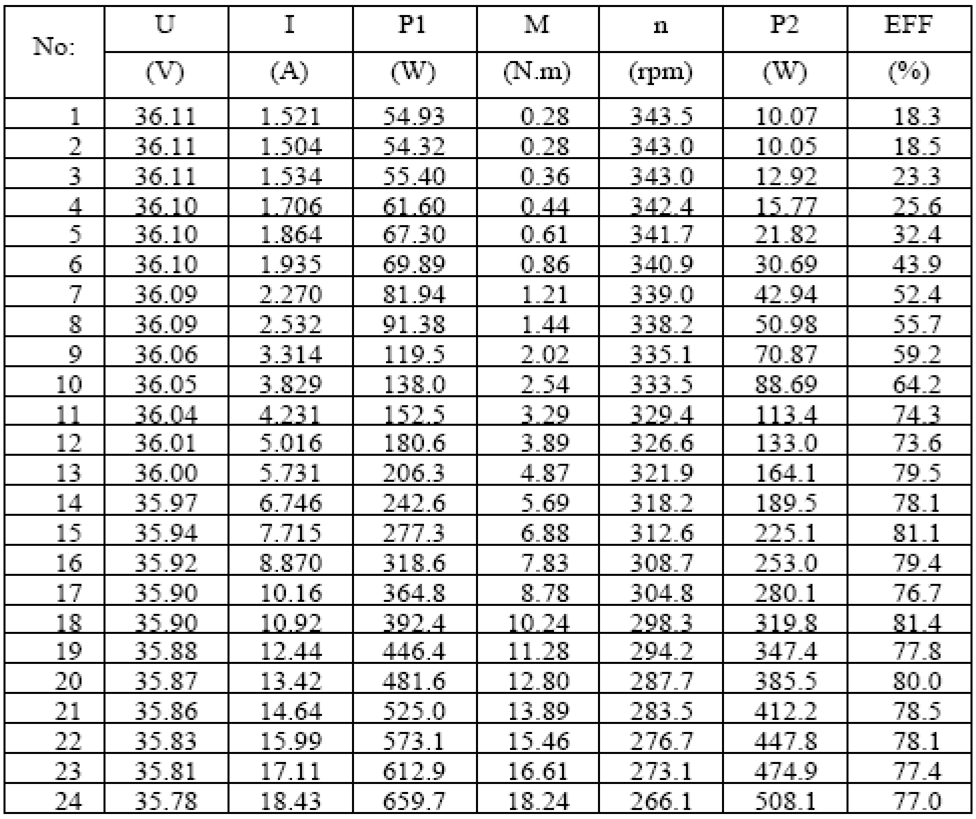
http://ww1.microchip.com/downloads/en/AppNotes/00857a.pdf

http://www.brushlessmotor.com.ar/Caracteristicas.html

ANNEX A – MOTOR DATASHEET



项目(Descripton)	U	I	P1	M	n	P2	Eff
	(V)	(A)	(W)	(N.m)	(rpm)	(W)	(%)
空载点(No_Load)	36.11	1.521	54.93	0.28	343.5	10.07	18.3
最高效率点(Max_Eff)	35.90	10.92	392.4	10.24	298.3	319.8	81.4
最大输出功率点(Max_Pout)	35.78	18.43	659.7	18.24	266.1	508.1	77.0
最大转矩点(Max_Torque)	35.78	18.43	659.7	18.24	266.1	508.1	77.0
结束(End)	35.78	18.43	659.7	18.24	266.1	508.1	77.0



No:	U	I	P1	M	n	P2	EFF
	(V)	(A)	(W)	(N.m)	(rpm)	(W)	(%)
1	36.11	1.521	54.93	0.28	343.5	10.07	18.3
2	36.11	1.504	54.32	0.28	343.0	10.05	18.5
3	36.11	1.534	55.40	0.36	343.0	12.92	23.3
4	36.10	1.706	61.60	0.44	342.4	15.77	25.6
5	36.10	1.864	67.30	0.61	341.7	21.82	32.4
6	36.10	1.935	69.89	0.86	340.9	30.69	43.9
7	36.09	2.270	81.94	1.21	339.0	42.94	52.4
8	36.09	2.532	91.38	1.44	338.2	50.98	55.7
9	36.06	3.314	119.5	2.02	335.1	70.87	59.2
10	36.05	3.829	138.0	2.54	333.5	88.69	64.2
11	36.04	4.231	152.5	3.29	329.4	113.4	74.3
12	36.01	5.016	180.6	3.89	326.6	133.0	73.6
13	36.00	5.731	206.3	4.87	321.9	164.1	79.5
14	35.97	6.746	242.6	5.69	318.2	189.5	78.1
15	35.94	7.715	277.3	6.88	312.6	225.1	81.1
16	35.92	8.870	318.6	7.83	308.7	253.0	79.4
17	35.90	10.16	364.8	8.78	304.8	280.1	76.7
18	35.90	10.92	392.4	10.24	298.3	319.8	81.4
19	35.88	12.44	446.4	11.28	294.2	347.4	77.8
20	35.87	13.42	481.6	12.80	287.7	385.5	80.0
21	35.86	14.64	525.0	13.89	283.5	412.2	78.5
22	35.83	15.99	573.1	15.46	276.7	447.8	78.1
23	35.81	17.11	612.9	16.61	273.1	474.9	77.4
24	35.78	18.43	659.7	18.24	266.1	508.1	77.0

ANNEX B – FIRMWARE

```
#include <18F2431.h> //header file for this PIC

#device ADC=10 //initialize for ADC

#fuses
HS,NOFCMEN,NOIESO,PUT,BROWNOUT,BORV45,NOWDT,NOPWMPIN,LPOL_LOW,HPO
L_LOW,NOLVP,NOPROTECT

#use delay(clock=20000000)//set clock of the micro

#use fast_io (A) //initialize INPUT-OUTPUT

#use fast_io (B)

#use fast_io (C)


#byte OVDCOND   = 0x0F6B //initialize register of the micro

#byte OVDCONS   = 0x0F6A

#byte FLTCONFIG = 0x0F6C

#byte PWMCON1   = 0x0F6E

#byte PDC0L     = 0x0F79

#byte PDC1L     = 0x0F78

#byte PDC2L     = 0x0F77

#byte PDC0H     = 0x0F76

#byte PDC1H     = 0x0F75

#byte PDC2H     = 0x0F74

#byte PORTA     = 0x0F82

#byte PORTB     = 0x0F81
```

```
#byte PORTC     = 0x0F80

#byte SSPBUF    = 0x0FC9

#byte SSPSTAT   = 0x0FC7

#byte SSPCON    = 0x0FC6

// encoder:

#byte CAP1CON   = 0x0F63

#byte CAP2CON   = 0x0F62

#byte CAP3CON   = 0x0F61

#byte DFLTCON   = 0x0F60

#byte MAXCNTL   = 0x0F64

#byte MAXCNTH   = 0x0F65

#byte POSCNTL   = 0x0F66

#byte POSCNTH   = 0x0F67

//#byte QEICON    = 0x0FB6

#byte VELRH     = 0x0F69

#byte VELRL     = 0x0F68


#include <stdio.h>

#define  FW_VERSION  13

char buildVers[10] = {__DATE__};


//---------------------------------------------------------------------------

//      DEFINES PER I/O:

//---------------------------------------------------------------------------
```

```
// #define  DEBUG  1

#define  LED_MOTOR_ON   output_low (PIN_B7)

#define  LED_MOTOR_OFF  output_high(PIN_B7)

#define  LED_DIR_ON     output_low (PIN_B6)

#define  LED_DIR_OFF    output_high(PIN_B6)

#define  LED_CYCLE_ON   output_low (PIN_C0)

#define  LED_CYCLE_OFF  output_high(PIN_C0)


#define  OFF          0x00  // motor OFF

#define  ON           0x01  // motor ON


#define  BACKWARD     0x00  // direction backward

#define  FORWARD      0x01  // direction forward


#define  MAX_SPEED     80

#define  DEFAULT_SPEED  72

#define  MID_SPEED     8

#define  MIN_SPEED     1


// Duty cycle limit definition, for 20KHz @20MHz, 2uS deadtime

// minimum duty cycle corresponds to 3 x deadtime = 3 x 2uS = 6uS   PDC =
6uS/(4/Fosc)

#define  MIN_DUTY_CYCLE     0x003C   // try doubling from 0x1E   ;
```

```
#define  MAX_DUTY_CYCLE      0x03E0   //maximum duty cycle is 4 x PTPER


#define GO_MIN                  0x44

#define TYPE_COMMAND            0x4C

enum STATUS_ENUM

{

  STATUS_OFF = 0,

  STATUS_RUN_FW,

  STATUS_RUN_BK,

  STATUS_BRAKE,

  STATUS_BRAKE_IN_POSITION,

  STATUS_GOTO_IN_POSITION

};


//variabili globali

unsigned int8 _direction = 0;           // flag direction

unsigned int8 _status = STATUS_OFF;


unsigned int8  _phaseOffset1;  // Phase1 offset to the Sine table(0)

unsigned int8  _phaseOffset2;  // Phase2 offset to the Sine table(120)

unsigned int8  _phaseOffset3;  // Phase3 offset to the Sine table(240)

void init_PCPWM( void )

{

  unsigned char portbStatus;
```

```
// setup_power_pwm(modes, postscale, time_base, period, compare, compare_postscale, dead_time);

// clock x PWM diviso 4,free run, clock per dead time clock diviso 2,postscale 1:1,time base=0,periodo=F9.

// compare e compare postscale=0,dead time=18(1sec secondo il source assembly)

  setup_power_pwm( PWM_CLOCK_DIV_4 | PWM_FREE_RUN |PWM_DEAD_CLOCK_DIV_2, 1, 0, 0xF9, 0, 0, 20 );

  set_power_pwm0_duty( 0 );

  set_power_pwm2_duty( 0 );

  set_power_pwm4_duty( 0 );

  set_power_pwm6_duty( 0 );


  if( _status == STATUS_BRAKE  )

  {

    setup_power_pwm_pins( PWM_ODD_ON, PWM_OFF, PWM_OFF, PWM_OFF );

    portbStatus = PORTB & 0xC0;

    PORTB = portbStatus | 0x1B;

  }

  else

  {

    portbStatus = PORTB & 0xC0;

    PORTB = portbStatus | 0x3F;

    setup_power_pwm_pins( PWM_COMPLEMENTARY,

                  PWM_COMPLEMENTARY,
                        PWM_COMPLEMENTARY,

                        PWM_OFF );

  }


  OVDCOND = 0xFF;


  PWMCON1 = 1;

  OVDCONS = 0;     // OVDCONS is configured such that all PWM outputs are 0 upon
power-up.

  // bsf      TRISC, 1        ;Ensure that RC1 is an input

  FLTCONFIG = 0;

}


void initMotorStart(void)

{

  _phaseOffset1 = 0;


  if( _direction == BACKWARD )

  {

    _phaseOffset2=12;

    _phaseOffset3=24;

  }

  else if( _direction == FORWARD )

  {
    _phaseOffset2=24;

    _phaseOffset3=12;

  }


  set_timer0( 62758 );

}

#INT_TIMER0

void update(void)

{

  unsigned int8 power;


  power = _speed / 16;


  if( ( power > 2 ) && (_period > 2000) )

    power = 2;

  else if( power > 3 )

    power = 3;


  set_timer0( t0Set[_speed] );


  if( _status != STATUS_BRAKE  )

  {

    if( ++_phaseOffset1 > 35 )

      _phaseOffset1 = 0;
    if( ++_phaseOffset2 > 35 )
      _phaseOffset2 = 0;

    if( ++_phaseOffset3 > 35 )
      _phaseOffset3 = 0;
  }

  set_power_pwm6_duty( 0 );
  set_power_pwm0_duty( sine_table[power][_phaseOffset1]);
  set_power_pwm2_duty( sine_table[power][_phaseOffset1]);
  set_power_pwm4_duty( sine_table[power][_phaseOffset1]);
}
//----------------------------------------------------------------------------------

short CheckInput( unsigned int16 pin )
{
  short status;
  unsigned int8 i;

  for( i=0; i < 10; i++ )
  {
    status = input( pin );
    delay_ms( 1 );
    if( status != input( pin ) )
      i=0;
  }

  return ~status;
}

//---------------------------------------------------------------------------

void stopMotor(void)
{
  disable_interrupts( INT_TIMER0 );

// disable PWM
  OVDCOND = 0;
  setup_power_pwm_pins(PWM_OFF, PWM_OFF, PWM_OFF, PWM_OFF);

  _phaseOffset1 = _phaseOffset2 = _phaseOffset3 = 0;

  if( _status < STATUS_BRAKE )
  {
    _status = STATUS_OFF;
    //LED_MOTOR_OFF;
    //LED_DIR_OFF;
  }
}
//---------------------------------------------------------------------------

void startMotor( short newStatus )
{
  _status = newStatus;

  if( _status == STATUS_RUN_FW )
  {
    _direction = FORWARD;
  }

  if( _status == STATUS_RUN_BK )
  {
    _direction = BACKWARD;   }
  OVDCOND = 0xFF; // accende PWM
  init_PCPWM();
  initMotorStart();

  enable_interrupts( INT_TIMER0 );
  enable_interrupts( GLOBAL );
}
```

```
//--------------------------------------------------------------------------

unsigned int8 misura( void )
{
  unsigned int16 mis = 0;
  unsigned int8 i;

  for( i=0; i<16; i++ )
    mis += READ_ADC();

  mis >>= 10;
  return( (unsigned int8) mis );
}

//------------
#INT_TIMER5
void Timer5( void )
{
  if( _countPeriod <= 0xFFFF0000 )
    _countPeriod += 0xFFFF;
}

//--------------------------------------------------------------------------
```

```
//   Input capture interrupt

//-----------------------------------------------------------------------------


#INT_IC1

void Hall_A()//input capture IC1

{


if( !CheckInput( PIN_A2 ) )&& ( !CheckInput( PIN_A3 ) )//SENSOR ABC=110

  {

    LED_MOTOR_ON;

    set_power_pwm_override( 1, 0, 0);

  }else if ( CheckInput( PIN_A2 ) )&& ( !CheckInput( PIN_A3 ){

    LED_MOTOR_OFF;

    set_power_pwm_override( 2, 0, 0);

   else if ( !CheckInput( PIN_A2 ) )&& ( CheckInput( PIN_A3 ){

    LED_MOTOR_OFF;

    set_power_pwm_override( 3, 0, 0);

  }*/




}


#INT_IC2QEI

void Hall_B()//input capture IC2
```

```
{
  if( !CheckInput( PIN_A3 ) )
  {
    LED_DIR_ON;
  }else{
    LED_DIR_OFF;
  }
}


#INT_IC3DR
void Hall_C()//input capture IC3
{
  //LED_CYCLE_ON;
  if( !CheckInput( PIN_A4 ) )
  {
    LED_CYCLE_ON;
  }else{
    LED_CYCLE_OFF;
  }
}
void main( void )
{
  unsigned int8 i;
```

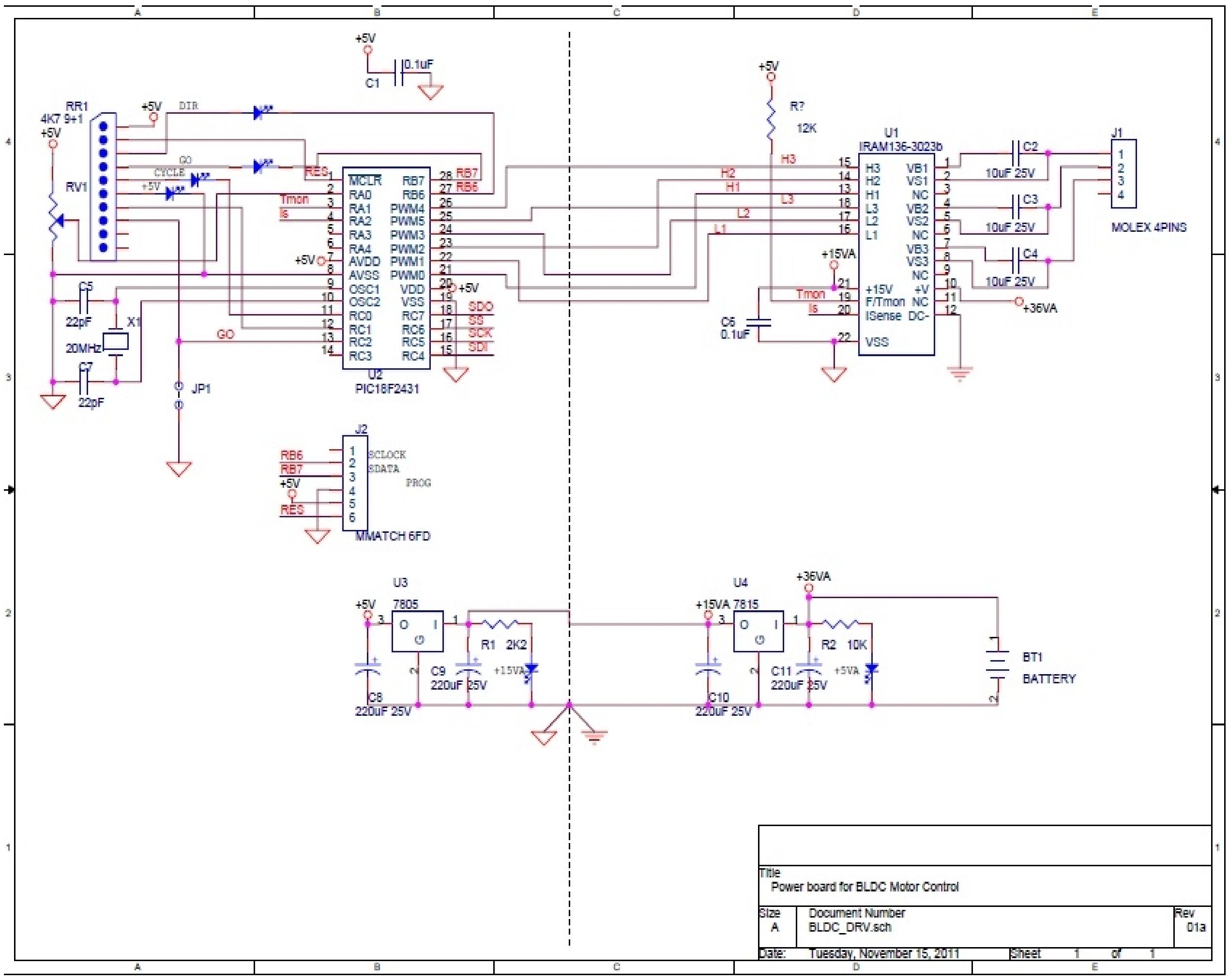
```
// Inizializzazioni
   set_tris_a( 0xFD );   // all input except RA1
   set_tris_b( 0x00 );   // all output
   set_tris_c( 0x7E );
   output_a( 0xFD );
   output_b( 0xFF );
   output_c( 0xFF );

   setup_wdt( WDT_OFF );
   setup_timer_0( RTCC_DIV_1 | RTCC_INTERNAL );
   setup_timer_1( T1_DIV_BY_1 |T1_INTERNAL );
   setup_adc_ports( sAN0 );
   set_adc_channel( 0 ); // initialize ADC
// initialize ADC
   setup_adc( ADC_CLOCK_DIV_8 );

// Set encoder
   DFLTCON = 0x7E;
   MAXCNTH = 0x00;
   MAXCNTL = 0xFF;

   CAP1CON|= 0b01001000;
   CAP2CON|= 0b01001000;
   CAP3CON|= 0b01001000;
```

```
// ------------------------------------------------------

// inizializzazione per pilotaggio diodi IRED:

// Timer = 4 * Tx_DIV_x) / clock

// Timer = (4 * 8) / 20000000 = 0.0000016

// ------------------------------------------------------

  setup_timer_5( T5_INTERNAL | T5_DIV_BY_1 );

  enable_interrupts( INT_TIMER5 );

  enable_interrupts(INT_IC1);//INTERRUPT INPUT CAPTURE

  enable_interrupts(INT_IC2QEI);

  enable_interrupts(INT_IC3DR);


  _speed = DEFAULT_SPEED;

  _status = STATUS_OFF;


  enable_interrupts( INT_SSP );

  enable_interrupts( INT_TIMER1 );

  enable_interrupts( GLOBAL );


    for(;;)

    {


      LED_CYCLE_ON;

      delay_ms( 200 );
```

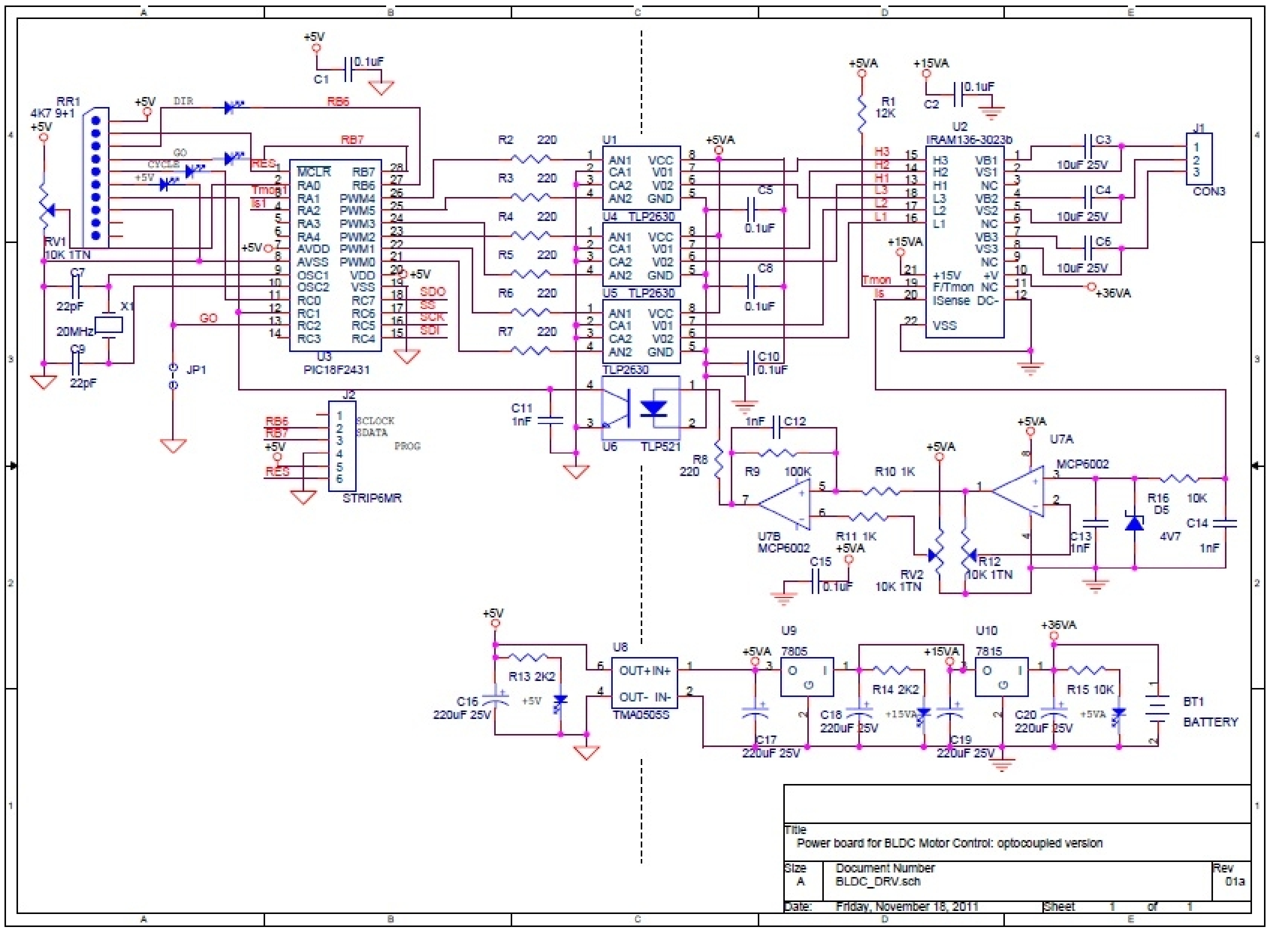
```
LED_CYCLE_OFF;

delay_ms( 200 );


if( !CheckInput( PIN_A4 ) )

{

  LED_CYCLE_ON;

}else{

  LED_CYCLE_OFF;

}


_speed = misura() & 0x7F;  // numero che varia da 0 a 15


if( CheckInput( PIN_C2 ) )

{

   _status = STATUS_RUN_FW;


   startMotor( _status );

   LED_MOTOR_ON;

}

else

{

   stopMotor();

   LED_MOTOR_OFF;

}
```

UNIVERSITY OF SOUTHERN DENMARK

ANNEX B -CIRCUIT



ANNEX C- CIRCUIT WITH OPTOCOUPLER


U3 PIC18F2431
U1 TLP2630
U4 TLP2630
U5 TLP2630
U6 TLP521
U2 IRAM136-3023b
J1 CON3
J2 STRIP6MR
U7A MCP6002
U7B MCP6002
U8 TMA0505S
U9 7805
U10 7815
BT1 BATTERY
RR1 4K7 9+1
RV1 10K 1TN
X1 20MHz
JP1
Title
Power board for BLDC Motor Control: optocoupled version
Size A
Document Number BLDC_DRV.sch
Rev 01a
Date: Friday, November 18, 2011
Sheet 1 of 1