CODEN:LUTEDX/(TEIE-5288)/1-135(2011)

Industrial Electrical Engineering and Automation

Modelling System of "Slide In" Energy transfer Between Road and Vehicle

Fernando Castillo
Gilfredo Remón
Nihad Sinanovic

Division of Industrial Electrical Engineering and Automation
Faculty of Engineering, Lund University

MODELLING SYSTEM OF "*SLIDE IN*" ENERGY TRANSFER BETWEEN ROAD AND VEHICLE

Fernando Castillo
Gilfredo Remón
Nihad Sinanovic

Department of Industrial Electrical Engineering and Automation
Lund University
July 2011

ABSTRACT

As fossil based fuels are ever more proving to be an unsustainable energy-resource, it is of critical importance to find a new way to deliver power to the worlds vehicles in order to anticipate an ever-growing population and sustain economic growth. The task is complicated but very achievable from a technological point of view. It is however greatly overshadowed in complexity when it comes to educating about new technology and making it seem accessible and plausible to members of the general population who are not inclined to embrace radical changes.

This thesis presents a new concept of delivering electric energy to road vehicles. It is in this thesis referred to as "Slide-in method of energy transfer between road and vehicle". The principle of the concept is that direct delivery of energy to vehicles should occur via electrically powered sections in the road.

In order to aid in educating people about the concept a scaled model representing such a system has been built. The model involves autonomously controlled vehicles travelling a course with electrically powered lanes. The vehicles have retractable connectors that connect to the lanes and are used to charge the vehicles' on-board battery. This gives the vehicles the ability to travel on roads with and without powered lanes.

The model was built at LTH in Lund during January through June 2011 and was demonstrated at Volvo Powertrain headquarters in Gothenburg in June 2011. The model got positive response from the attendees and proved to be a valuable asset in visually presenting the concept.

CONTENTS

INDEX OF FIGURES

INDEX OF TABLES

I. INTRODUCTION

Motivation

In the future, fossil fuel resources will not be enough to cover the road transportation energy demand and bio fuels are not enough to compensate this, which leads to a need for using electrical energy. This has been achieved so far by implementing batteries on board vehicles, which are charged to provide them with the energy required. However, battery capacity is not sufficient to cover long distance transportation.

A possible solution to this problem is enabling electrical energy supply to moving vehicles. This is not a completely new idea, since it has been implemented in systems like Trolley Bus supply. Nevertheless, it is not visually acceptable and there is none possibility to use it on small vehicles like cars.

Lately several realistic solutions to electric energy transfer from the road to a moving vehicle are being developed, one of these is known as Slide In technology. This does not suffer from the drawback of e.g. Trolley supply since they have a very low visual impact and work with any type of vehicle. There is however no experience of such system solutions and a pedagogical threshold to overcome in explaining the idea; an idea that may change the world of road transportation.

Problem Definition

Given the lack of knowledge in the general public about this idea; this project is intended to make a “Slide In” demonstration system, with all the basic functionalities of a full scale implementation, such that it can be used to facilitate peoples understanding of this concept.

Goal

The main goal of this project is to rebuild a Scalextric Toy Car system such that it fulfils the following specific goals:

1. The system must have track sections that are electrically isolated from adjacent sections.
2. Each track (2 in parallel) in a section can individually be energized from a DC source on the roadside via a silent relay function.
3. An energized track is visualized with a LED-strip illuminated along the track.

4. Some tracks have not sliding contact surfaces in order to represent a road system without Slide In implementation. Thus, the vehicles must run in electric or hybrid mode.
5. The vehicle has an on board battery to store energy from the track
6. The vehicle has a controllable voltage converter between the track and the battery
7. Every vehicle can identify its position to an external supervisory control computer
8. Vehicles have an on board control system for SOC, Track Power and Motor Voltage.
9. The vehicle must indicate its SOC, Track Power and Motor Power on some kind of display on board.

Conceptual Design

The full project was divided in three subsystems: track, vehicles and supervisory PC software, which interact with each other according to the diagram shown in Figure I.1. The supervisory PC software communicates via serial RS-232 protocol with the Track and via Bluetooth with the vehicles. Also the Track can supply electrical power to the vehicles, and each section emits a unique IR (infrared) signal, which is received by the vehicles in order to identify their position along the track.

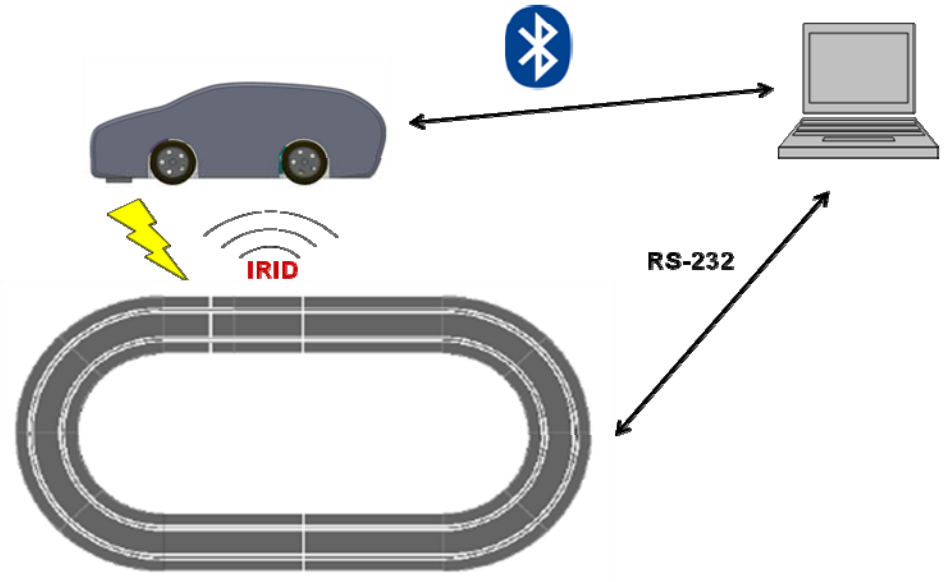


Figure I.1 Subsystems Interaction Diagram

Individual Contributions

The workload of this master thesis project was equally distributed between the three authors. Nihad Sinanovic was on charge of developing the vehicle mechanical implementation and the supervisory PC software; Fernando Castillo and Gilfredo Remón worked on Track Mechanical Implementation, Track Electronic Hardware and Software and Vehicle Electronic Hardware and Software.

II. TRACK MECHANICAL IMPLEMENTATION

Goals

The track represents one of the most notable elements of the overall structure. Due to the necessity of developing an easy-understanding demonstration system, it was important to achieve some specific goals in the mechanical design of the track.

- It needed to have an appropriated size for developing an area with non-slot paths that represented an urban zone and rural paths, surrounded by an electrified highway. This highway must be able to supply power to the vehicles through Slide In technology and it would represent inter-city infrastructures for transportation, which is the main target of Slide In.
- It was necessitated to design a transportable system.
- The electrified highway must be developed based on Scalextric track pieces.
- The system must be built at the IEA Workshop.

Design

For achieving the established goals to the track design, it was decided to set a track size of 3[m] x 4.3[m]. Moreover, it was divided in six different tables of 1[m] x 2.15[m] with independent wiring, all of them with specific track sections and connected each other. This idea was motivated due to the requirement of a transportable track.

The definitive model of the track was designed in AutoCAD; the Figure II.1 shows the mentioned model, it can be seen the main highway built with Scalextric pieces and the non-slot roads that were constructed in the workshop of the IEA department.

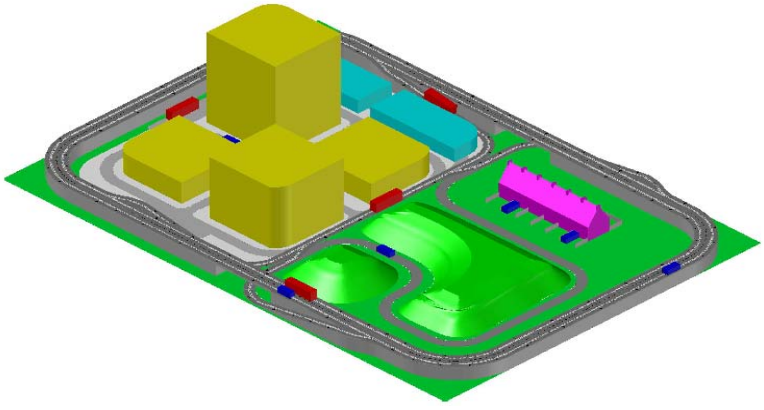


Figure II.1 Track Scale Model

On the other hand, the Figure II.2 illustrates how the design is divided in six subparts (tables).

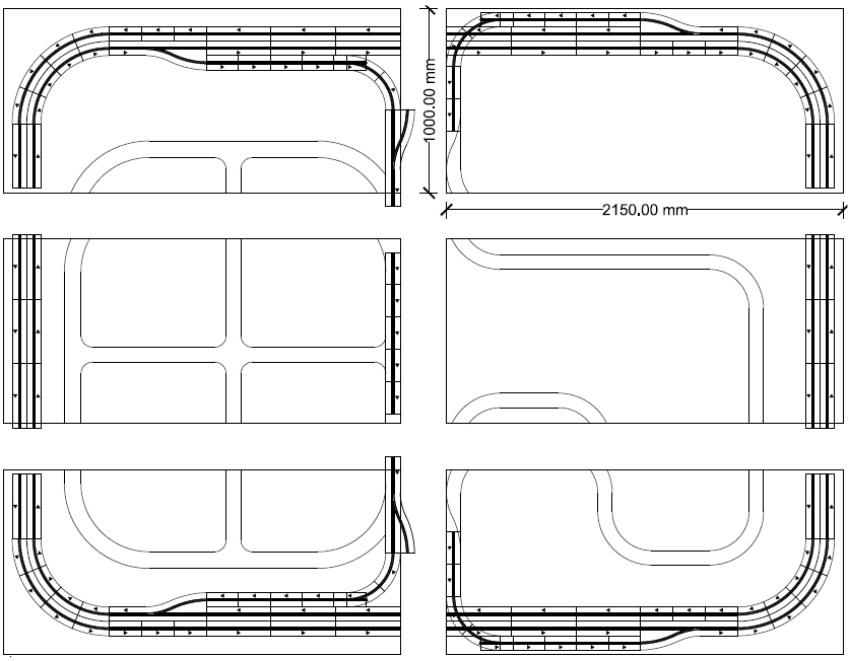


Figure II.2 Track Tables Division

In addition, it was demanded to design a holder structure for the track that was mainly planned and developed by the Engineer Getachew Darge from IEA. It consists in eight adjustable workbenches that were distributed and set in order to support the four (4) main wooden girders, which function is to hold the six independent track sub-tables. The structure is represented in the Figure II.3 and Figure II.4, where the workbenches are drawn in black, the girders in blue and the tables in green.

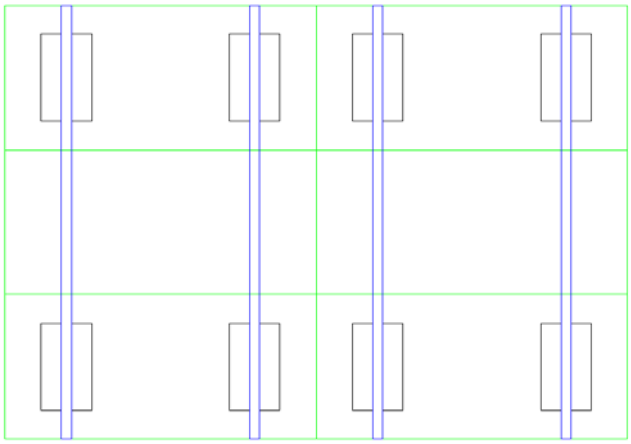


Figure II.3 Top View of the Track Holder Structure

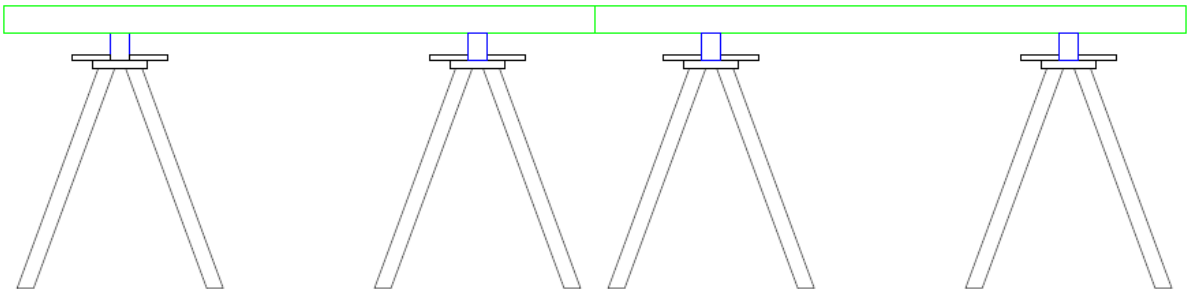


Figure II.4 Lateral View of the Track Holder Structure

Track sections classification

For functionality reasons, it was necessary to implement different types of track sections. There are sections capable of supplying power to the vehicles from the slot when it is demanded; these are called Track Power Sections. Also, the sections unable to supply power are known as Single Track Sections; these can be divided in sections with slot or without it. The assignation of all the track sections according to its type is shown in the Figure II.5. Blue sections correspond to Track Power Sections and the gray and yellow ones are Single Track Sections (with slot and without it respectively).

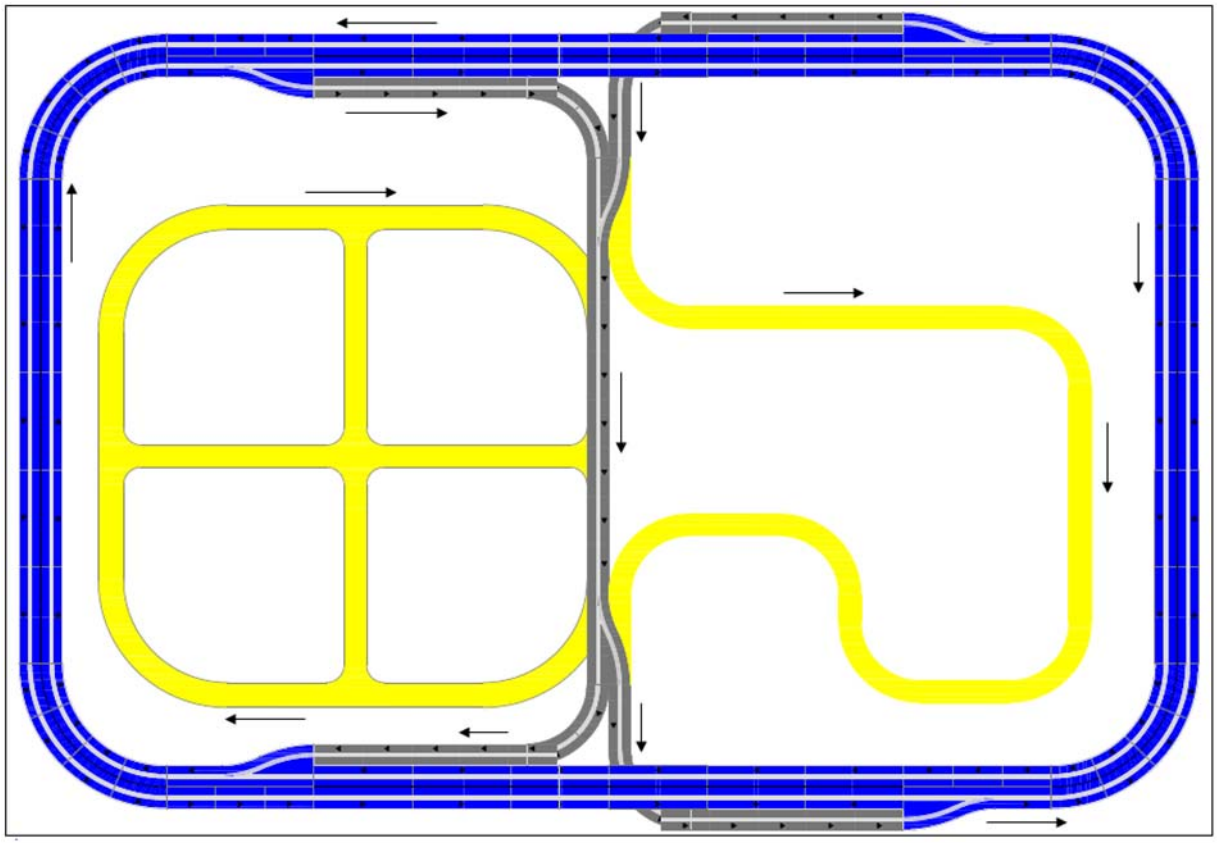


Figure II.5 Distribution of Track Sections

Traffic Direction

It is also important to explain the planned traffic direction on the track; it is based on the right-driving traffic standard, as indicated by the arrows in Figure II.5. This is the used traffic pattern in Sweden.

Finished Track

The constructed track can be seen in Figure II.6. Also, Figure II.7 shows how the tables can be separated in order to transport the whole track.

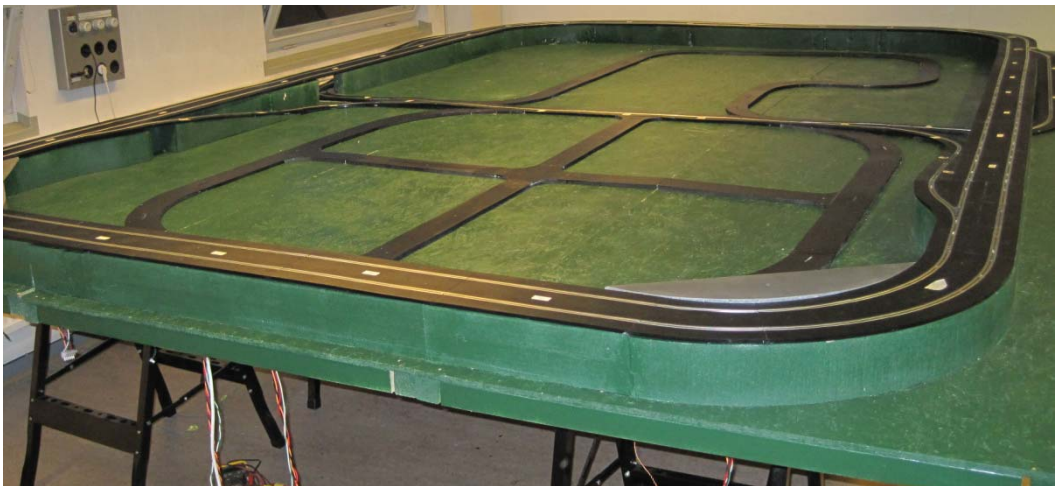


Figure II.6 Finished Track

Figure II.7 Track Disassembly Process

III. TRACK ELECTRONIC HARDWARE

Goals

The Track Electronic Hardware has been designed based on the requirement of several features for the functionality of the whole track system, as mentioned previously. The following objectives can give a main idea about the most important characteristics that the track electronic hardware must fulfil.

- The electronic supply system of the track must be able to supply a maximum power of around 140 Watts, in order to distribute it among all the vehicles (max. 10 units running and charging at same time). In addition, the track supply system was designed for low power consumption while there is no demanding of energy from the vehicles.
- The track was developed to achieve the capability of communication between own designed supervisory PC software and each track power section.
- The track electronic system would be able to switch on/off power of every track power section in a short period of time and in a silent way.
- Each track section should emit a specific IR signal. This feature is required for the developing of the vehicles local position system.
- The track electronic system would have the capability to detect short circuit in every track power section.
- The track electronic system should be able to control the direction of the vehicle traffic in the three intersections on the track.
- It should have an independent hardware for generating the proper signal to the Wire Guide System.

Overview

The development of the track electronic system consisted in the design and implementation of four main elements: the electric power supply, a master board, a slave board and the communication protocol. All of them have specific functions and interrelations with the rest of the elements, in order to set the functionality of the whole track.

The track electronic system structure is shown in the block diagram of Figure III.1.

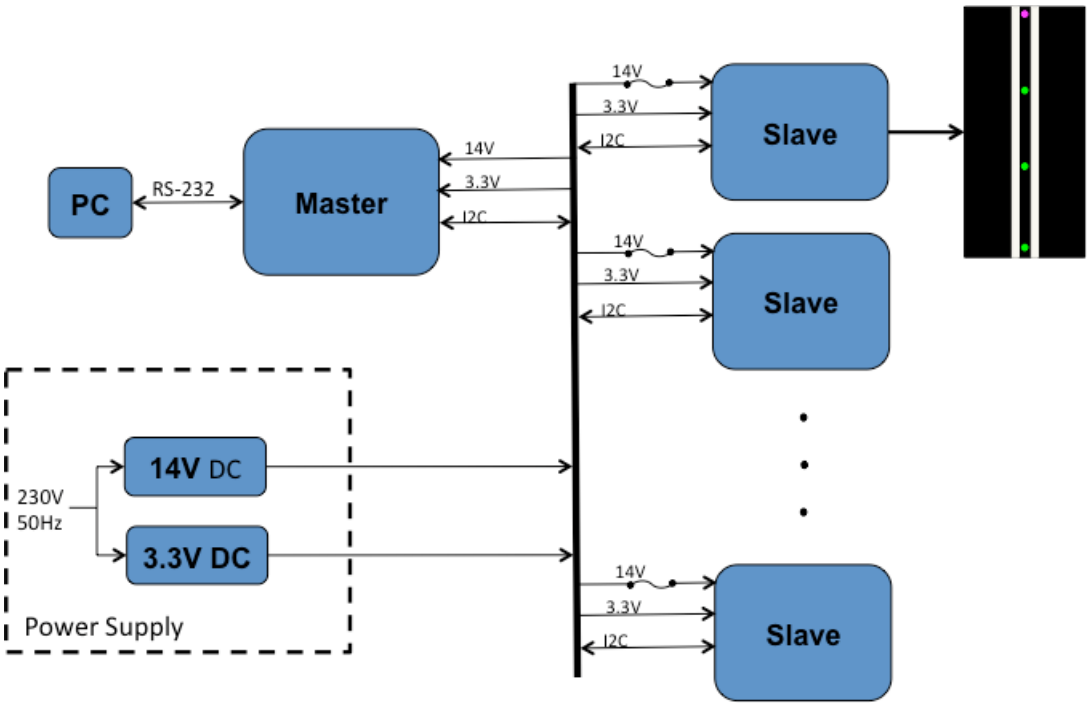


Figure III.1 Track Electronics Block Diagram

The Power Supply consists in two main DC Voltage sources, one of 14VDC and the other of 3.3VDC, with a common ground. These sources supply the power to the rest of the system: the master board and slave boards.

The Master Board is only one board, which main function is creating the communication interface between the PC and the several slave boards that control each track section. It is based on an Atmel ATmega88 8 bits MCU, this unit handles the serial communication via RS232 protocol and 2-wire communication interface for I2C protocol. In addition, the board is also able to control the six (6) electric coils that set the different directions on the track intersections.

On the other hand, the Slave board includes the required hardware for controlling the power, detecting possible short circuits, emitting IR codification and communicating with the PC in order to receive instructions or sent the status of the individual track sections. Besides, the signal demanded by the Wire Guide System is developed in an independent subsystem of the Track Electronic Hardware.

Master Board

This unique main board has the roll of message buffer between the PC and the different slave boards. It receives all the messages from the PC via serial communication with RS232 protocol and converts it to TTL level voltages through the implementation of a

MAX3232 transceiver, and vice versa, i.e. from TTL to RS232. Once the message signals have been transformed to compatible levels with the ATmega88 MCU, the microcontroller process each message and send it through the I2C bus, which is connected to all the slave boards.

Some of the sent messages from the PC to the Master Board are not intended to be delivered toward the slave boards. These messages represent commands to control six (6) different coils from the three intersections on the track. Every two coils allow setting the direction of the vehicles in one intersection through a mechanism called Lane Switching. The Master Board has also all the required hardware for receiving the mentioned commands, interpreting it and controlling the coils in a short time.

The Master Board Hardware Structure is summarized in the scheme of Figure III.2. As it can be seen, there are four main components that involve this Board: MAX3232 transceiver, the ATmega88 MCU, the I2C bus and the Coils Lane Switching Hardware, besides the Power Supply Set.

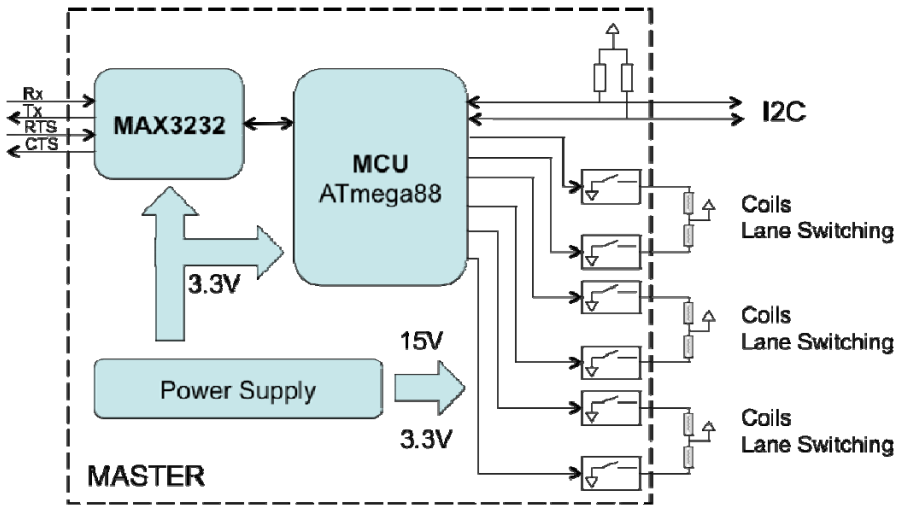


Figure III.2 Master Board Block Diagram

MAX3232 Transceiver:

The ATmega88 includes a Universal Synchronous and Asynchronous Serial Receiver and Transmitter (USART), a very flexible serial communication device (ATmega88 datasheet, 2010). It made possible the alternative of developing a communication bridge between the PC and the microcontroller unit. However, it was required to convert the signals from the RS232 protocol of the PC to compatible TTL level voltages.

The MAXIM MAX3232 transceivers have a proprietary low-dropout transmitter output stage enabling true RS-232 performance from a 3.0V to 5.5V supply. The device requires only four small 0.1µF external capacitors. Moreover, the MAX3232 is guaranteed to run at data rates of 120000 baud while maintaining RS-232 output levels (MAX3232 datasheet, 1999).

The hardware configuration of the MAX3232 was established according to the typical operating circuit, found in the device datasheet. The Figure III.3 presents the corresponded schematic.

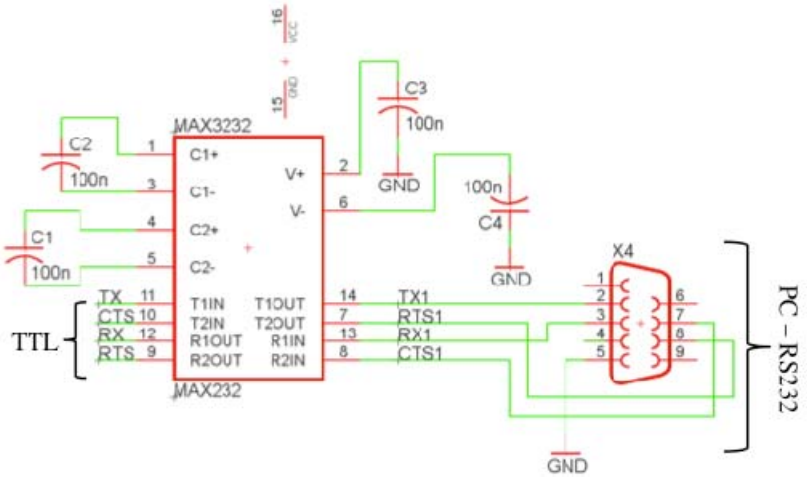


Figure III.3 MAX3232 Schematic

It is important to mention that all the capacitors are plastic of 0.1uF. Besides, the X4 component represents a female DB9 connector, which it is used to connect the PC with the board via serial wire.

I2C Hardware:

All I2C devices are configured as open collector output and because of this, the I2C lines require a pull-up resistor, which value is limited by Equation III.1 (ATmega88 Datasheet, 2010 p307). This equation is valid for I2C clock frequencies lower than 100 [KHz].

$$\frac{V_{cc} - 0.4[V]}{3mA} < R_p < \frac{1000ns}{Cb} \qquad \text{(III.1)}$$

V_{CC} is the supply voltage and Cb is the line capacitance. Using Vcc equal to 3.3 [V] and Cb of 400 [pF], which is its limit according to I2C standard (Atmega88 datasheet, 2010), the following boundaries are obtained.

$$996.66[\Omega] < R_p < 2506[\Omega]$$

Since there was no estimation about whether the lines capacity would be under its limit, it was decided to use 1000 [Ω] pull-up resistors, as shown in Figure III.4. This way, if the lines capacity is bigger than the protocol limit, the chosen resistors would still accomplish with the restriction on their value.

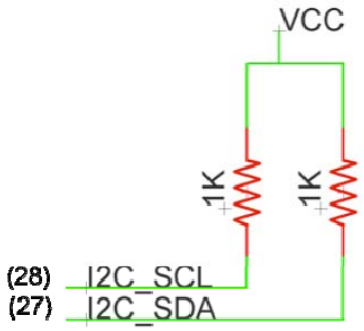


Figure III.4 I2C Hardware Schematic

Coils Lane Switching:

Scalextric® Cars offers a particular track piece which represents a bidirectional and electrically controllable slot intersection. It allows controlling the traffic direction when one of the vehicles goes over those mentioned intersections.

The controller mechanism of the intersection piece consists in two coils that are connected through a mechanical configuration that pairs each coil with a possible direction. Furthermore, setting one of the directions requires the control of the current flow over the corresponded coil throughout time. The Figure III.5 shows the implemented Scalextric piece with the mechanism for changing the direction and the coils.

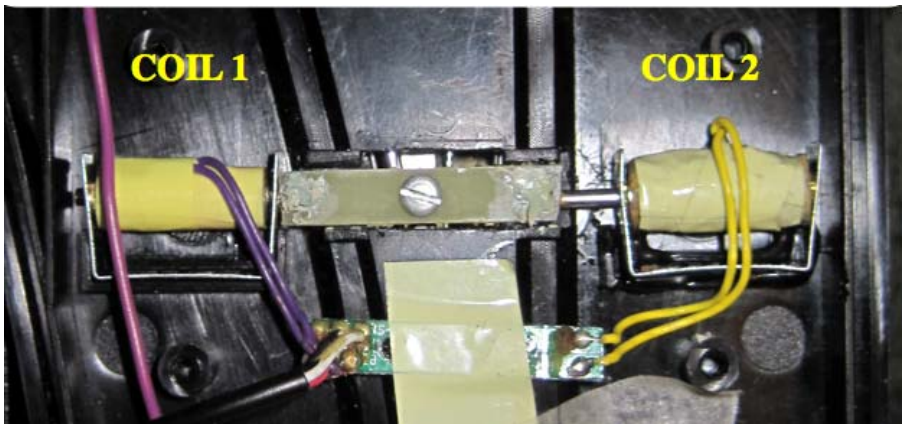


Figure III.5 Coils Lane Switching Mechanism

However, due to lack of time and requirements of designing a simple and compatible system with the electronic hardware, it was decided to implement only the Scalextric coils, discarding the rest of the original electronic piece.

In DC, the real model in stationary state of an inductor consists in a simple resistance with specific value (Sadiku, 2006). The Figure III.6 illustrates this real model of the inductor. The original coils of Scalextric had an internal resistance around 4,7Ω (measured), while the inductances varied. It was not possible to find additional information about these components.

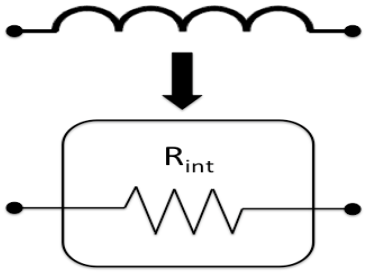


Figure III.6 Real Model in stationary state of a Inductor in DC

As mentioned before, to achieve the operation of the Coils Lane Switching System (CLSS) it was demanded to design a system, which allows controlling the current flow over a coil throughout time with the ATmega88 MCU. The Figure III.7 describes the desired operation of the system: a defined Output MCU port should generate a step of logic high (Vcc) –made by software- for a long enough period of time in order to activate the mechanical mechanism associated to the coil. (Note the transient responses of the coil current around transition times).

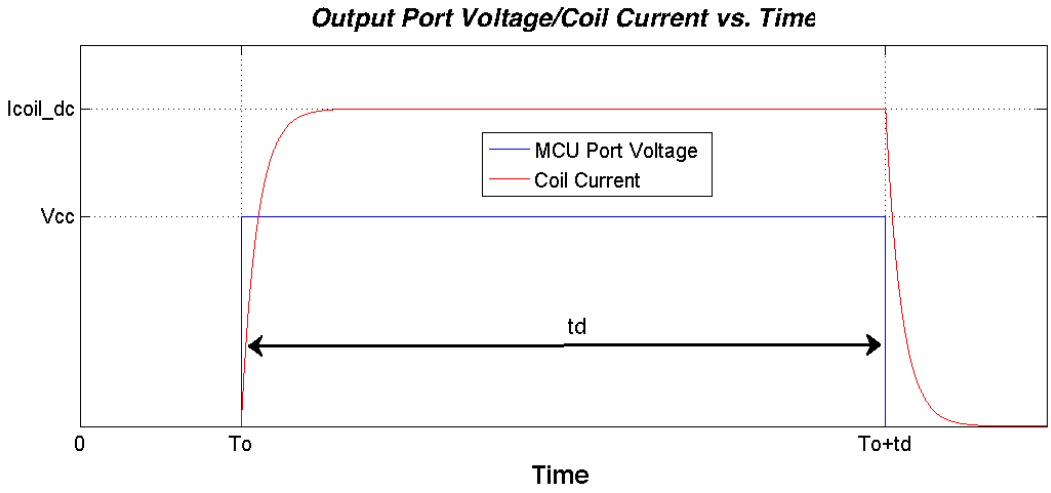


Figure III.7 Output Port Voltage/ Coil Current vs. Time

The internal resistances of the coils are around 4.7 Ω and the applied voltage is close to 14 Vdc (design reasons). According to Ohm's law (Sadiku, 2006) the DC current corresponds to a maximum of 3A, which involves high power requirements. For that reason, it was decided to use step pulses of approx. 200 [ms] (td) for avoiding to break the coils. *Important: This time period was estimate due to lack of additional information.*

Conclusively, according to all the requirements, facts and conditions mentioned before, the resulted circuit of the Coils Lane Switching corresponds to the shown in the Figure III.8. The circuit consists in two main stages: the photo-coupler and a power transistor. All of the components were taken from IEA workshop stock.

- Photo-coupler: this stage allows taking the control signal from the MCU port in order to reflect it over the gate of the power transistor at 14V. This makes possible to isolate the two supplies voltages.
- Power Transistor: it is based in an n-channel mosfet that could handle drain currents over 3A. The model shown in the schematic of the Figure III.8 is not the only implemented unit. The workshop stock had different models of n-channel mosfet, but all of them with very similar characteristics. The 1N4004 diode and the gate resistor have only protection functions.

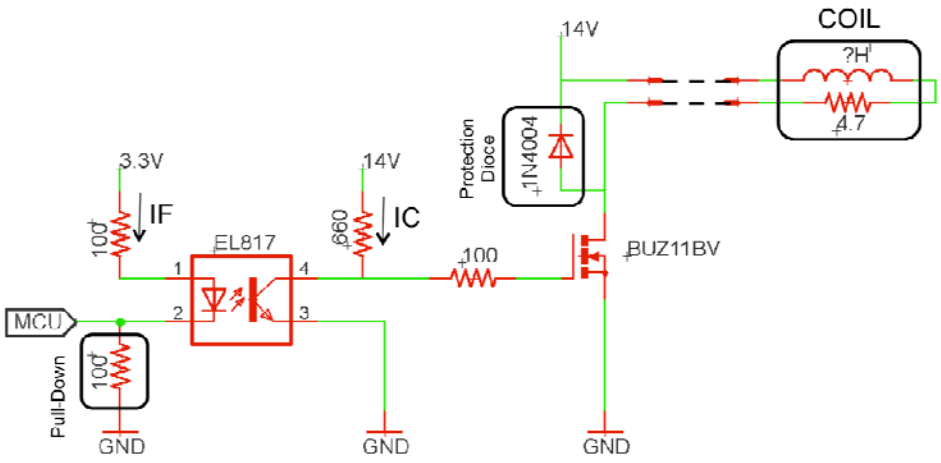


Figure III.8 Coils Lane Switching Schematic Circuit

The circuit operation can be explained in the following Table III.1.

Output port voltage	***If [mA]***	***Vgs [V]***	***Icoil [A]***
High - Vcc	0	14	> 1
Low – 0V	21	0	0
Tri-state (resetting) – Hi Z	10.5	0	0

Table III.1 DC operation of the circuit

The ATmega88 MCU often resets by software due to issues related with I2C; during the resetting, the I/O ports of the microcontroller unit are in Tri-State i.e. High Impedance. For that reason it was mandatory to add a pull down resistor that would avoid activating the coil mechanism during resets.

Finally, the hardware shown for the Coils Lane Switching repeats six (6) times, matching to each coil.

ATmega88 MCU:

The microcontroller unit represents the core of the Master Board. It is in charge of controlling the rest of the features: I2C communication, Coils Lane Switching system and the communication between the PC and the slave boards.

The following list shows all of the MCU resources that were implemented in the Master Board:

- Reset external pin.
- ISP 6 pin Programming Interface, according to AVR Hardware design considerations for 8 bits MCU (AVR042, 2011, p5).

- USART.
- Two Wire Interface for I2C.
- Nine (9) I/O ports, as output: six for CLSS and three for debugging LEDs.

The definitive resource distribution was designed according to the block diagram of Figure III.9.

Power Supply – Master Board:

The Power supply of the Master Board comes from the two main DC Voltage sources that supply the whole track: 3.3V and 14V. However, it is distributed within the board itself. The MAX3232 Transceiver, the I2C hardware and the MCU are supplied only by 3.3VDC (Vcc). On the other hand, the Coils Lane Switching system is supplied by both sources: 14V and 3.3V.

Moreover, additional hardware was placed to keep decoupling in the supply lines. This was achieved with different sets of decoupling capacitors (tantalum, plastic and electrolytic). Protection diodes were also implemented for protecting the whole board.

In despite of the fact that both supply sources have a common ground, the implementation of two independent planes of ground on the Board -corresponded to each source supply- was required in order to avoid resets on the MCU due to high current peaks produced by the Coils Lane Switching.

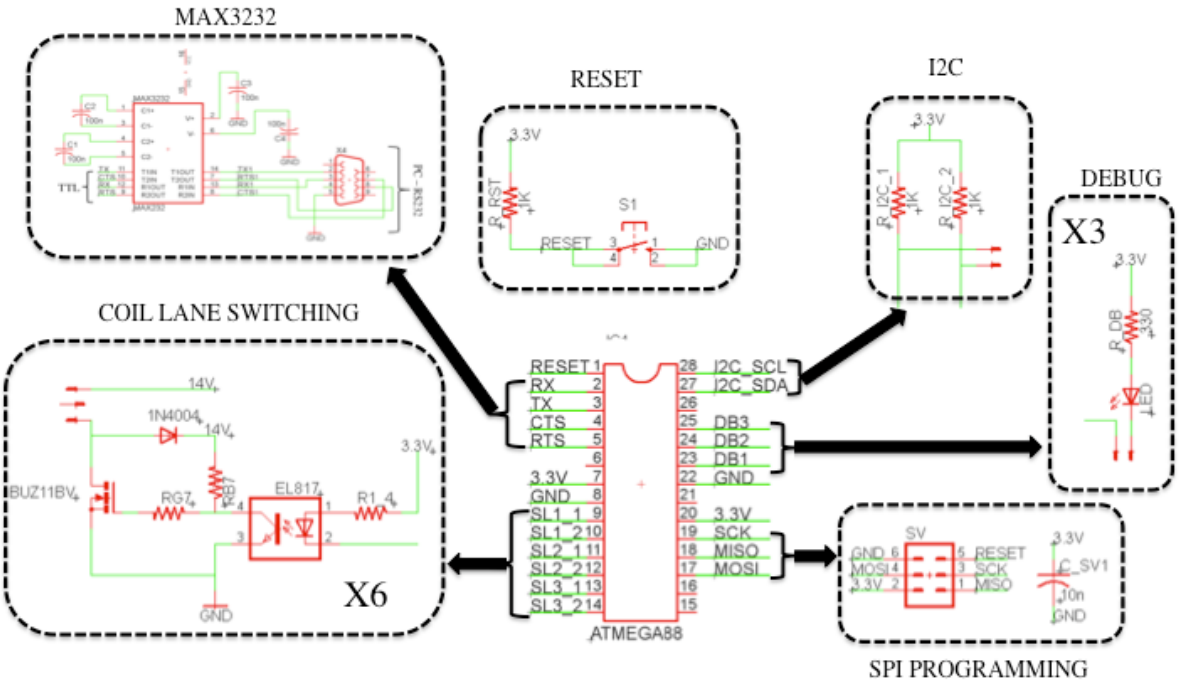


Figure III.9 ATmega88 Resources Distribution

Finally, the Figure III.10 shows the decoupling system through the whole Master Board, with its corresponded component list in Table III.2.

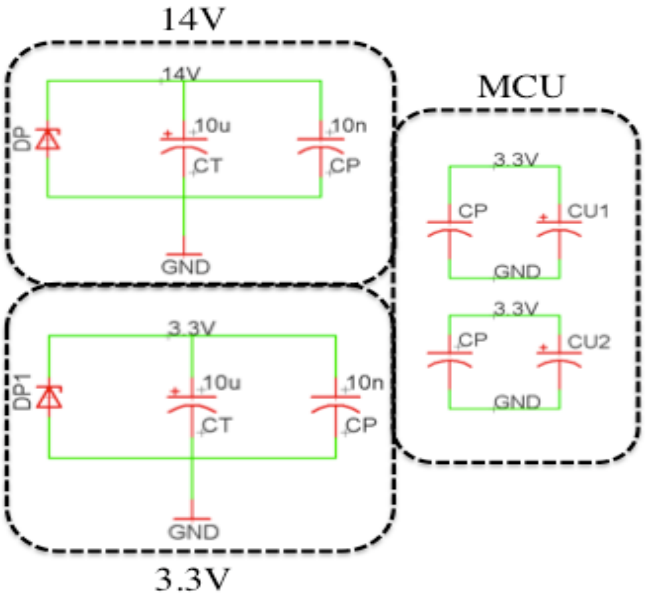


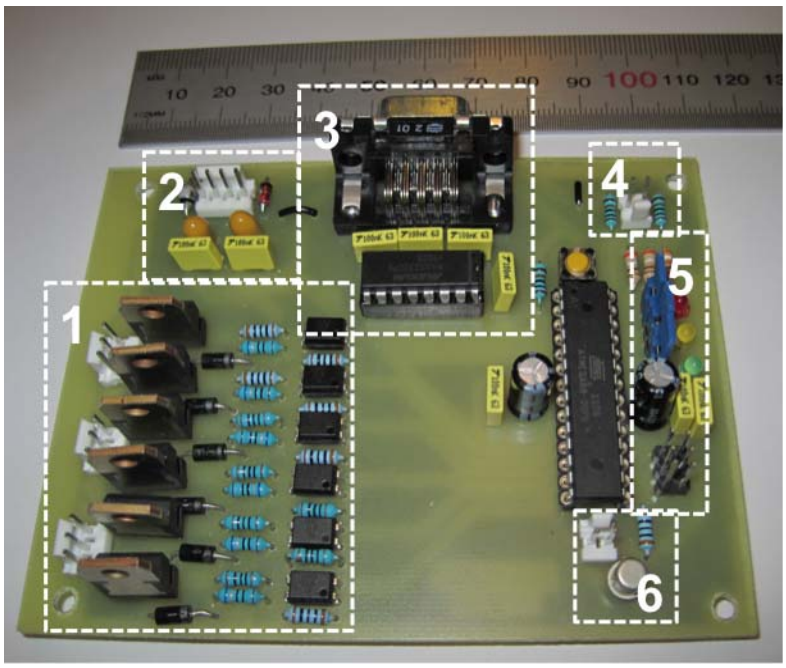
Figure III.10 Master Board Schematic of decoupling system

CU	Electrolytic Capacitor 10uF
CT	Tantalum Capacitor 10uF
CP	Plastic Capacitor 10nF

DP	Zener Diode 15V
DP1	Zener Diode 3.6V

Table III.2 Decoupling Capacitors

The Figure III.11 shows the definitive Master Board with the corresponded distribution of its hardware.



1. - Coil Lane Switching System; 2. – Supply Connector and Decoupling; 3. - MAX3232;

4. – I2C; 5. – ISP Programming Interface & Debugging LEDs; 6. – NOT used.

Figure III.11 Master Board PCB

Slave Board

The Slave Board represents the heart of the track hardware. It involves a group of thirty (30) boards that control all the track sections according to a specific distribution. Each board has the functions of setting on/off the power, detecting short circuits and emitting codified IR signals in four different track sections. The Slave Board Block Diagram is exposed in the Figure III.12.

All of the Slave Boards are connected to the I2C bus, where all the delivered messages from the Master Board go through. The MCU receives the messages, processes it and controls the rest of the board hardware, according to the established instructions coded into the messages.

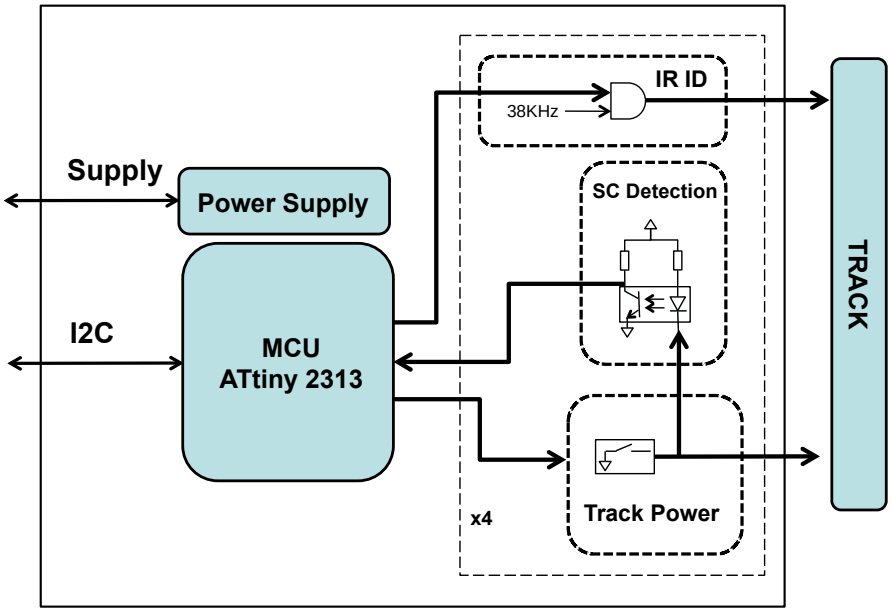


Figure III.12 Slave Board Block Diagram

The hardware of the board is based on an Atmel ATTiny2313 8 bits MCU. Then, it divides in three main subsystems; IR Identification, Track Power and Short Circuit detection; besides the power supply and decoupling systems.

Track Power:

It was required to develop a system that allowed setting independently the power on/off on each track section for a short period of time. The system needed to be controlled from the MCU (at 3.3V ~ 0V levels) and to be able to handle operations around 14V and currents over 1A. For that reason, a photo-coupler was implemented that makes possible to isolate the MCU supply from the supplied power to the track, avoiding damages to the microcontroller unit.

The Figure III.13 shows the design of the definitive Track Power System. The used transistor corresponds to a p-channel Hexfet® power mosfet, able to handle currents up to 11[A] (IRFRU9024 datasheet, 1997). Moreover, the MCU port must be set as output for controlling the power on the Track section.

On the other hand, the Table III.3 presents the DC operation values of the circuit.

The operation of the circuit depends on the two possible values in the output port of the MCU. Low output sets 14V on the track and supplies the currents that it would require, while a High level on the MCU output shuts down the power on the track.

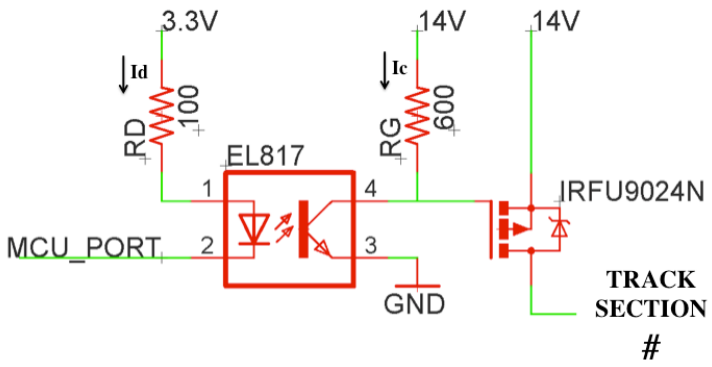


Figure III.13 Track Power System Schematic Circuit

V_{MCU_Port} [V]	Id [mA]	Ic [mA]	V_{Track} [V]
3.3	0	0	2 ~ 3*
0	20.7	21.0	~14

Table III.3 DC operation values of Track Power System

(*): It does not supply current on the drain i.e. the track.

Short Circuit Detection:

For safety reason it was demanded the design of a mechanism for detecting possible short circuits in the slot, i.e. between 14V and GND of a powered track section (see the Figure III.14).

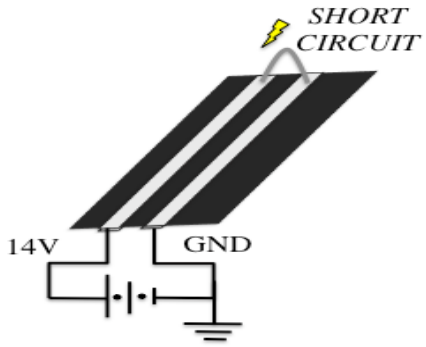


Figure III.14 Short Circuit Demonstration

The system required having a physical connection to the Track Power System, in order to develop a feedback between the short circuit detection and power setting on each track section. It was designed with the same photo-coupler of track power system but with different implementation. The Figure III.15 presents the schematic of the definitive short circuit detection.

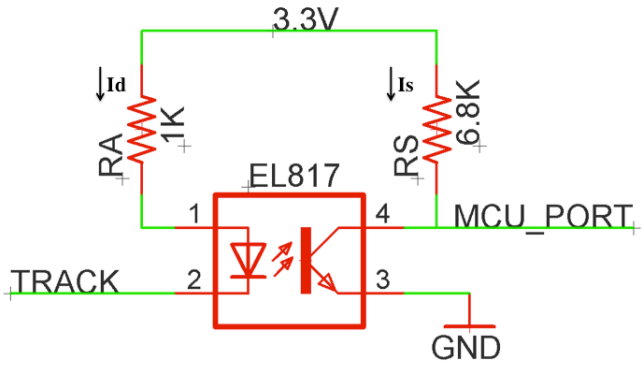


Figure III.15 Short Circuit Schematic circuit

The emitter LED of the photo-coupler emits only if the track voltage is under approx. 2V (EL187 datasheet, 2007, p6). When a short circuit is detected, the voltage on the MCU port (set as input) is at low level. The Table III.4 represents the DC operation of the system.

TRACK	Id [mA]	Is [mA]	V_{MCU_Port} [V]
ON – OK	0	0	3.3
OFF – OK	0	0	3.3
ON - SC	2.1	0.5	0.2
OFF - SC	2.1	0.5	0.2

Table III.4 DC operation of Short Circuit Detection System

The main goal of developing this detection system was establishing a method to avoid setting the power on short-circuited track section. Before powering a determined track section, the assigned short circuit indicator must be examined. It is important to notice that this system is only useful when the short circuit occurs before the track section has been activated.

Infrared Identification (IRID) LEDs:

The IR Identification system corresponds to the track hardware of the local position system for the vehicles. It must be able to set a unique and specific codification signal per track section, which is emitted via an infrared LED.

All of the vehicles have an IR receiver module, which function is detecting the different IR signals emitted by each track section. This device has a high gain for IR continuous or bursts signals at 38 KHz (TSOP58038 datasheet, 2011). For this reason, it was demanded the design of a system with the capability of generating 38 KHz signals with variable codification. The Figure III.16 shows the followed strategy for developing this subsystem.

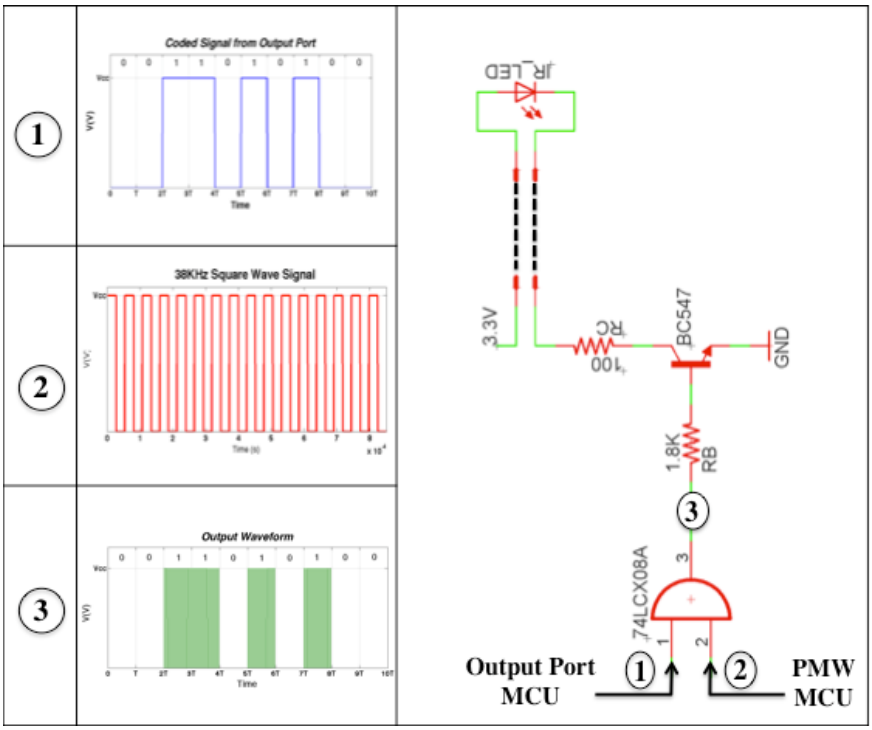


Figure III.16 Diagram of IR Identification system

A Low Voltage Quad 2-Input AND Gate 74LCX08A was implemented, for generating the different coded signals at a frequency of 38 KHz. One of the inputs is connected to an output port of the MCU that sets a specific code based on the track section; while the other input connects to a common square wave of 38 KHz that is generated by the Pulse Width Modulation (PWM) of the ATTiny 2313. Finally, the output signal of the AND gate goes through a simple NPN bipolar transistor for current gain over the corresponded infrared LED.

ATTiny 2313 MCU:

The ATTiny 2313 microcontroller unit is the core of the Slave Board. It is in charge of controlling and inspecting the rest of the board subsystems: I2C communication, IR Identification, Power System and Short Circuit Detection.

The following list shows all of the MCU resources that were implemented in the Master Board:

- Reset external pin.
- ISP 6 pin Programming Interface, according to AVR Hardware design considerations for 8 bits MCU (AVR042, 2011, p5).
- One PWM Channel.
- Two Wire Interface for I2C Communication.

- Twelve (12) I/O ports: 8 as output, four for ON/OFF power and four for IR ID LEDs; 4 as input for SC Detection System.

The definitive resource distribution was designed according to the block diagram of Figure III.17.

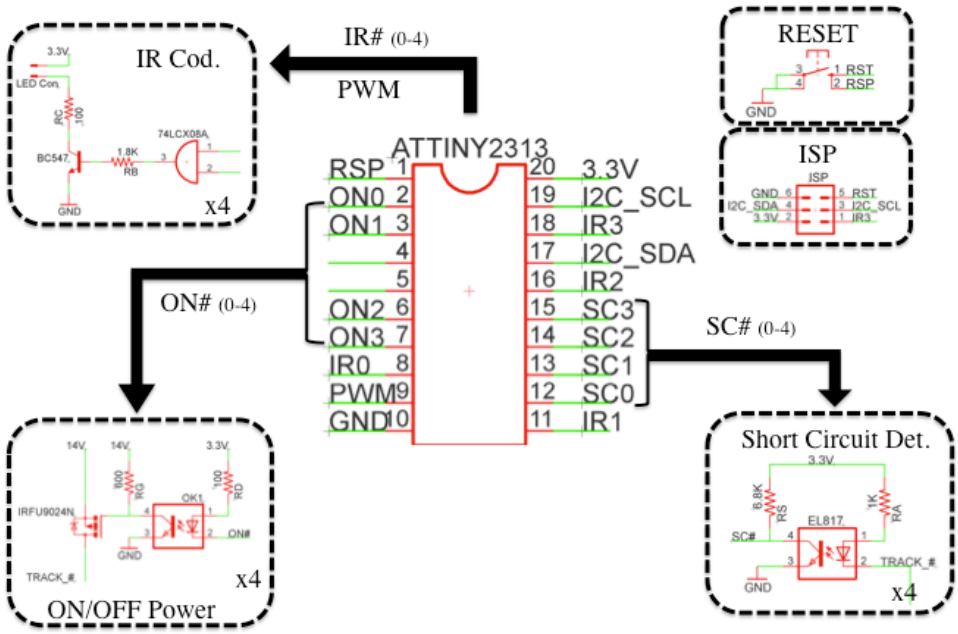


Figure III.17 Slave Board Block Diagram

Power Supply – Slave Board:

This board is supplied by 3.3[V] and 14[V]; however, the Track Power System is the only that requires the 14[V] supply; while the rest of the board was designed with 3.3[V] supply.

In order to improve the operation of the board, additional hardware was placed to keep decoupling in the supply lines:

- Two sets of 10uF tantalum capacitor in parallel with 10nF plastic capacitor. One per each supply and close to the board supplies connector.

- A set of 33uF electrolytic capacitor in parallel with 10nF plastic capacitor close to the 3.3V MCU supply line.

- 10uF tantalum capacitor and 10nF plastic capacitor next to the supply joints of the AND gate 74LCX08A.

Finally, for protection reasons it was decided to set Zener diodes of 5V and 15V on each supply lines.

Additional Information:

As mentioned before (Track sections classification p15), there are two different types of track sections: power track sections and single track sections. The single model requires a Slave board with only the necessary hardware for IR identification (i.e. MCU, 3.3V supply, RESET, ISP and IR Codification system). On the other hand, the power track sections need all the Slave Board features.

However, only one type of board was designed and manufactured, incorporating all the features. This was decided in order to decrease the cost and designing time. The manufactured boards were equipped according to the track requirements. The Figure III.18 and Figure III.19 show the definitive of Slave board.

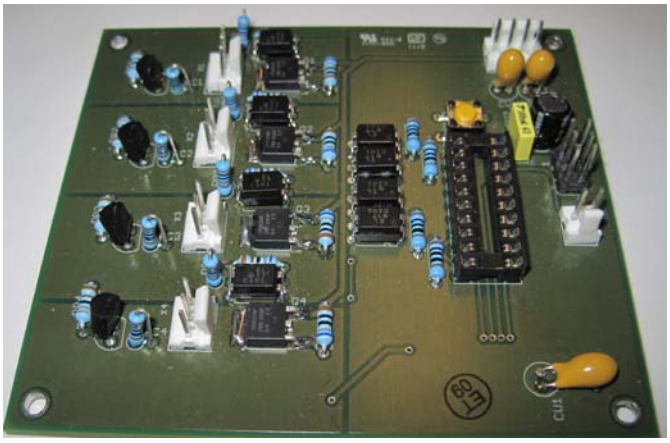


Figure III.18 Slave Board PCB– For power track sections

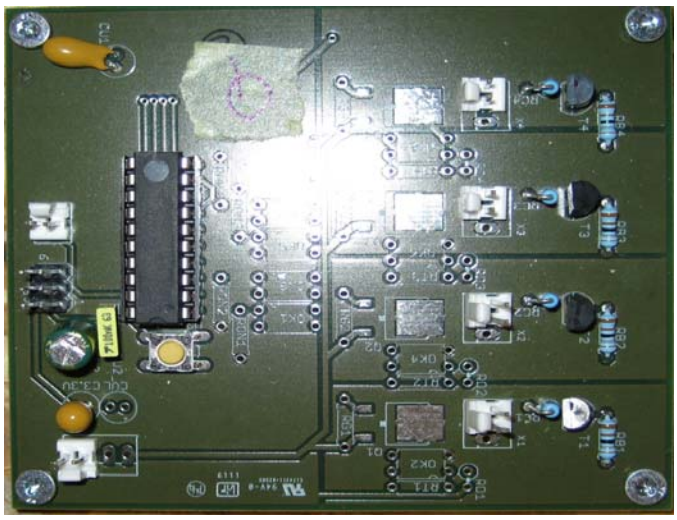


Figure III.19 Slave Board PCB– For single track sections

Wire Guide System

The Wire Guide System is used to run the vehicles through roads without guiding slot, i.e. not the classic Scalextric track parts. This is achieved by placing a hidden wire guide under the no slot-roads. This wire has a special signal that allows the vehicles to follow the wire trajectory through the measurement differences on the magnetic field generated around the wire (Coils Board p80).

The system consists in a long wire that generates a uniform magnetic field nearby; the intensity of this field is increased using high current signals. Moreover, for avoiding interferences with the rest of the track electronic hardware, it was decided to use a sine wave with a fixed high frequency between 140 KHz and 160 KHz.

Two first approaches were implemented into the main board, but unfortunately both failed; this because the generated signal was a square wave, which interfered with the I2C communication. Since there was a lack of time implement a custom sinusoidal wave generator, it was later decided to implement a Power Signal Generator previously built at IEA. The device can generate signals over 1 MHz without high disturbances and it can supply power over 5 Watts without risk of damages. These properties made the implementation of this device a simple and effective solution for developing a dependable hardware. The Figure III.20 shows the structure of the Wire Guide System Hardware on the Track.

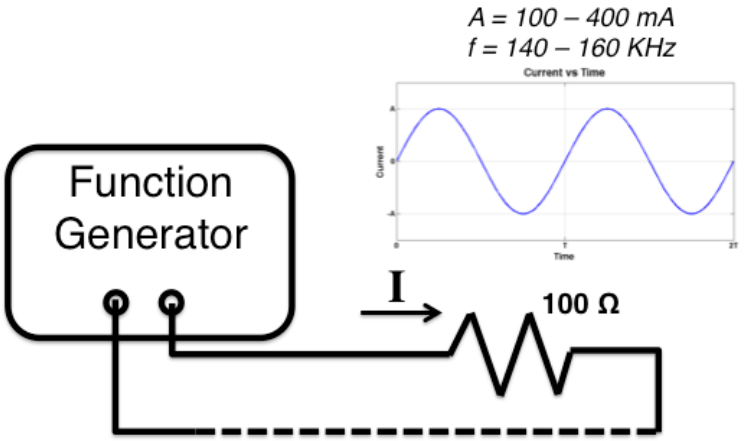


Figure III.20 Wire Guide System Structure

Complementary Hardware

There are two important elements that were considered for a proper design of the track electronic system: the power indicator LEDs and the infrared (IR) LEDs. The first ones required high intensity and low power consumption LEDs that would allow to show the

current ON-OFF state of each section. On the other hand, the IR LED needed to have a specific emitted wavelength (940 [nm]) and a wide viewing angle.

Power Indicator LEDs:

For saving hardware resources, the power indicator LEDs circuit was designed to be connected directly to the power lines of each track section. A proper illumination was an important aspect to be considered when developing of the hardware; for that reason, three high-brightness green LEDs were set per each track section, which were distributed equidistantly along the whole piece. The Figure III.21 shows the definitive circuit for the power indicator LEDs set in each track power section.

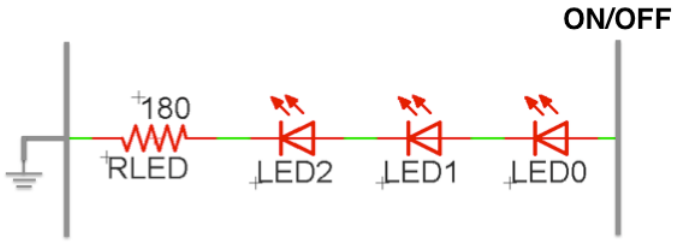
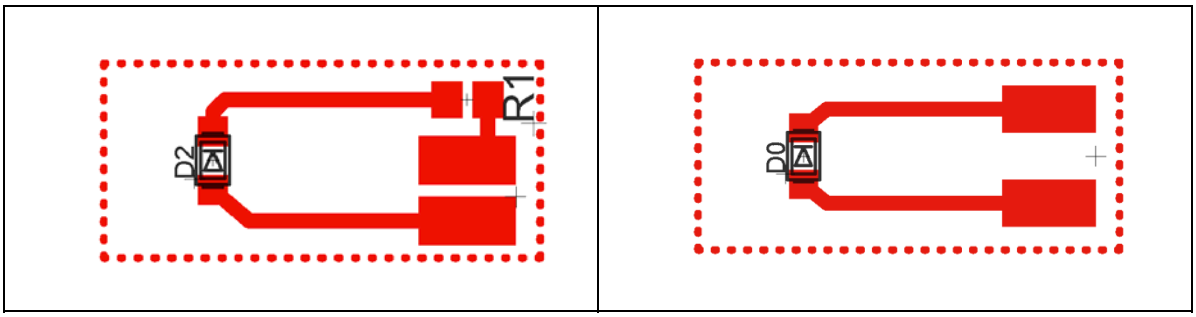


Figure III.21 Power Indicator LEDs Circuit

The three LEDs are on when the track section has been powered from its corresponded Slave board. The implemented LED model has a forward voltage of 3.3V (IR12-21C/ TR8 datasheet, 2011) and an adequate brightness with forward currents around 22 mA (typical operation), this made possible the development of a simple system connected directly to the track power. Besides, when the track section is unpowered, the circuit is not consuming power.

The power indicator LED circuits were manufactured at the IEA Workshop. Two different PCB models were developed: one with a LED and a resistor in series, and the other with a single LED. The Figure III.22 presents the two models designed in EAGLE.



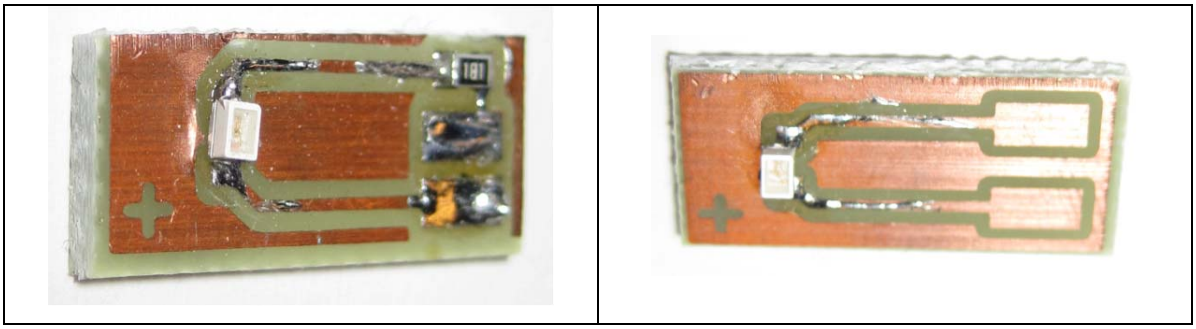


Figure III.22 Power Indicator LED PCB models

Infrared LEDs

The main property that the implemented infrared LED model should have fulfilled was a peak wavelength (λ) of 940 nm in the emitted infrared light. It was mandatory for matching with the operation conditions of the IR Receiver on the vehicles (TSOP58038 datasheet, 2011).

Moreover, the device had to be small enough to fit into the Scalextric slot piece cavities, and it needed to have a wide viewing angle to assure that the vehicles always received the IRID, even when driving at high speeds. For achieving these goals, an Everlight IR 12-21C LED was implemented, which has a low forward voltage (1.2V @ 20 mA) and a view angle of 160°. The definitive PCBs were all manufactured at the IEA workshop; the model is shown in the Figure III.23.

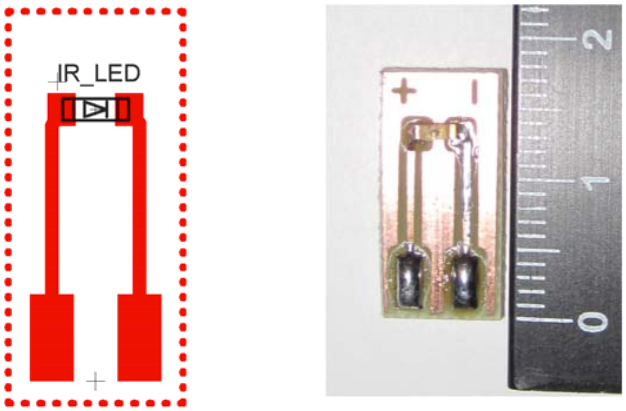


Figure III.23 Infrared LEDs

Main Power Supply:

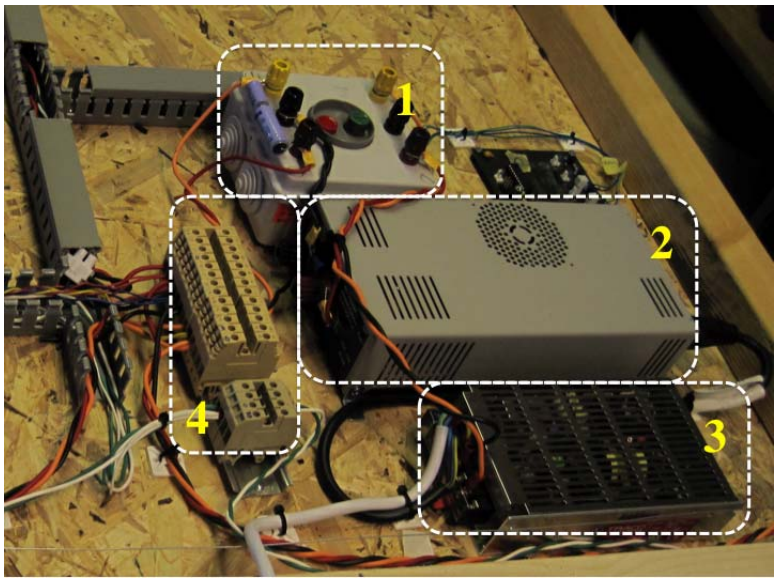
The track electronic system is supplied by two main sources of 3.3VDC and 14VDC. These power supplies were acquired from Farnell. It was important to have sources that would fulfil specific features of voltage, current, power and input voltage from 90 – 240 V AC.

The 14VDC source was in charge of supplying the power to charge the vehicle batteries from the slot. For charging one of the vehicles battery and driving its motors, it was required a current around 800 mA, but this could suffer some variations that suggested a maximum current requirement of 1A and maximum power of 14W per vehicle during charging state. According to this information, to charge 10 vehicles on the track at the same time needs a maximum consumption of 10A, and 140W. Moreover, the Lane Switching Mechanism requires peak of current of maximum 2.5A per change of direction.

Consequently, it was found an EA ELEKTRO-AUTOMATIK power supply that fulfilled all of the requirements: output voltage 14 VDC; output maximum current 21A and power rating of 300W.

On the other hand, the 3.3VDC source has the function of supplying the necessary power to the controller and rest of the track hardware: MCUs, IR LEDs, and Short Circuit Detection. A maximum current consumption of 250 mA was estimated from the 3.3VDC source per each Slave Board and 300 mA from the Master Board. For that reason, a TRACOPOWER Enclosed Power Supply of the TXL Series was implemented; the source has an output voltage of 3.3VDC and maximum current of 6.0 A.

The whole supply system of the track is shown in the Figure III.24. It is possible to see the main power switch and the two power supplies.



1.- Main Power Switch; 2.- 14VDC supply; 3.- 3.3VDC supply; 4.- Power Lines Distributor

Figure III.24 Power Supply system of the Electronic Track.

IV. TRACK SOFTWARE

Communication Protocol

An intermediate MCU (Master) is used to communicate the PC and the MCUs with direct access to the track hardware (Slaves). The communication from the PC to the Master is done via RS-232 and from the Master to the Slaves via I2C.

The Master is used in most of the cases as a repeater, i.e. the messages from the PC will be transmitted to the slaves, and vice versa, but in some cases the Master send messages to the computer, especially to inform communication problems with a specific slave; and the PC can send messages to the master in order to command lane switching.

Message Structure:

According to the I2C protocol (NXP Semiconductors 2007), a message starts with a byte containing the address and the read/write bit in LSB, as shown in Figure IV.1. If the read/write bit is cleared, the master continues sending data bits; and if it is set, the addressed slave sends the data. This structure is also used with certain modifications for flow control in the RS-232 communication.

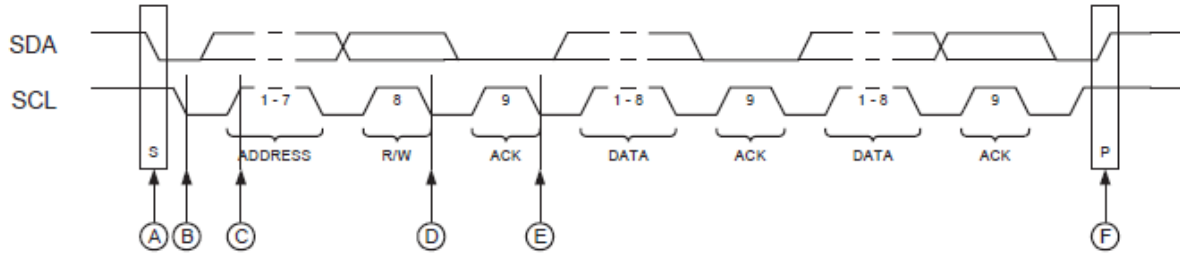


Figure IV.1 I2C messages structure (ATtiny2313 datasheet 2010)

Flow Control:

In the case of messages from the PC to the Master, the RTS (Ready to Send) line is cleared by the PC to notify the Master it wants to transmit a new package; then it waits until the Master is ready to process new data and clears the CTS (Clear to Send) line. After all bytes are sent, the PC sets RTS to notify end of message and the Master sets CTS.

Besides, for messages from the Master to the PC, 0xFF is sent as a start byte to indicate the beginning of a message.

Track Addresses:

The slaves have 7 bit addresses according to the I2C protocol, which go from 0x01 to 0x66 and each of them control five track sections. Address 0x00 is reserved for a general call (i.e. to send a command to all the slaves) and 0x7E is reserved to address the Master in further uses.

Moreover, every track section has a specific 8 bit ID. This ID is related with the slave address in the following way:

Data Bytes Messages:

- **Switch On/Off road power:**

Bit	7	6	5	4	3	2	1	0
	0	0	1	X	a3	a2	a1	a0
Description	Identifier				S3	S2	S1	S0

Table IV.1 Switch on/off road power message format

an = 1 -> Slaves section Sn powers up.

an = 0 -> Slaves section Sn powers down.

- **Switch On/Off IRID LED:**

Bit	7	6	5	4	3	2	1	0
	0	1	0	X	b3	b2	b1	b0
Description	Identifier				S3	S2	S1	S0

Table IV.2 Switch on/off IRID LED message format

bn = 1 -> Slaves section Sn sends IRID.

bn = 0 -> Slaves section Sn doesn't send IRID.

Data Bytes to the PC:

- **Report status:**

Bit	7	6	5	4	3	2	1	0
	0	0	1	0	e	e	e	e
	Identifier							
Bit	7	6	5	4	3	2	1	0
	0	1	0	0	f	f	f	f
	Identifier							
Bit	7	6	5	4	3	2	1	0
	1	1	0	0	g	g	g	G
	Identifier							

Table IV.3 Report status message format

e = 1 -> Slaves section Sn is powered up.

e = 0 -> Slaves section Sn is powered down.

f = 1 -> Slaves section Sn is short circuited.

f = 0 -> Slaves section Sn is OK.

g = 1 -> Slaves section Sn is sending IRID.

g = 0 -> Slaves section Sn is not sending IRID.

- **Communication error:**

Bit	7	6	5	4	3	2	1	0
	0	0	0	1	a	a	a	a
Description	Identifier				Error Code			

Table IV.4 Communication error message format

Possible error codes:

- 0x0 = SLA+W not acknowledged
- 0x5 = SLA+R not acknowledged
- 0xA = Data not acknowledged
- 0xF = Bus error

This message is sent by the Master to the PC after a failure in the I2C communication with a Slave.

Master Software

ATmega88 Modules:

The Master Board has its operational core in the AVR ATTMega88 8 bits microcontroller unit. It was required to program several tasks that the MCU must execute in order to control the two main functions of the board: Hold a communication bridge PC – slaves, i.e. from RS-232 serial protocol with the PC to I2C protocol with the slave units; Control the six different lane switching mechanism from the track that allows to set the traffic direction. The programming was based on C-language and performed with AVR Studio 4.

In order to perform these two systems, the following resources of the MCU were implemented:

- Main clock source: internal RC oscillator at 80 MHz
- USART Module
- One Timer 0 of 8 bits
- A Watchdog Timer

- 10 output ports (PD 07:05, PB 07:06 and PB00 assigned to Lane Switching System; PD2 to CTS of USART communication & PC02:00 to debugging LEDs)
- 1 input pin with external interrupt (PD3) implemented to RTS of USART communication
- 2 wire interface module with I2C protocol implementation, master mode

Code Structure:

The code scheme of the master software was developed in four main libraries, which have the function of handling and controlling the different MCU modules that were implemented to perform the required tasks of the system. Additionally, there is a main function included in an overall library that integrates and manages the whole master program flow. The realized libraries are presented below.

Lane Switching System:

Files: Switch_Lane.h & Switch_Lane.c

Description: this library sets up the Lane Switching ports as outputs. Besides, it is in charge of setting the different flags that indicate the status of each Lane Switching (LS) channel.

Functions:

- void Switch_Lane_I (void): the function is called to configure all the Lane Switching ports as output and clear all the flags that indicates the current status of each LS channel.

- void SW1_Left (void): it belongs to a group of six functions in charge of setting a specific direction in an intersection; this one is used to set to the left the direction of the vehicles passing trough intersection 1. It sets the corresponded LS channel status flag; once the flag has been activated it also sets the pin and starts the activator pulse of the LS mechanism associated to the pin.

- void SW1_ Right(void): this function sets to the right the direction of the vehicles passing trough intersection 1.

- void SW2_Left (void): this function sets to the left the direction of the vehicles passing trough intersection 2.

- void SW2_ Right(void): this function sets to the right the direction of the vehicles passing trough intersection 2.

- void SW3_Left (void): this function sets to the left the direction of the vehicles passing trough intersection 3.

- void SW3_ Right(void): this function sets to the right the direction of the vehicles passing trough intersection 3.

Timer:

Files: Timer.h & Timer.c

Description: all the settings of the Timer0 are set up in this library. The goal of the timer is counting the period of time that an activator pulse has been on in a determined LS channel. As mentioned before in Coils Lane Switching Hardware (see p21), if the length of the pulse is too long it could break the coil due to the high power dissipated in the device.

Functions:

- void Timer_I (void): this function sets the Timer0 in CTC mode, which allows to generate an interruption each time that the counter reach the top value. The interrupt frequency is given by equation IV.1 (ATTmega88 datasheet 2011, p96):

$$f_{int.} = f_{clk}/(2.N.(1 + OCR0A)) \quad \text{(IV.1)}$$

In order to obtain a timer interrupts every 2 ms (f_{int} = 500 [Hz]), the pre-escaler N is set to 64 and the OCR0A register to 125.

Interrupt routine:

The interrupt service routine is executed every 2 ms in order to check which flags are currently activated. All the LS channels that have set flags must check if its corresponded counter have reached the top that indicates if the activator pulse has lasted the requires 200 [ms]. If the top has been reached, the activator pulse is stopped by clearing the corresponding pin.

USART:

Files: USART.h & USART.c

Description: the library includes all the required functions to perform a communication between the PC and Master MCU via serial with the implementation of the USART module. The transfer of data between both elements must be ruled by some facts:

- The master unit sends data to the PC whenever it requires it.
- The PC clears the RTS to request the master permission to send (T1).
- When the master is ready to receive information, it clears CTS. Then, the PC begins to send data bytes (T2).
- When the PC has sent all the data, it sets again RTS. Finally, the master sets CTS and processes the received data (T3).

The steps that must be done in order to send information from the PC to the master are illustrated in time in the Figure IV.2

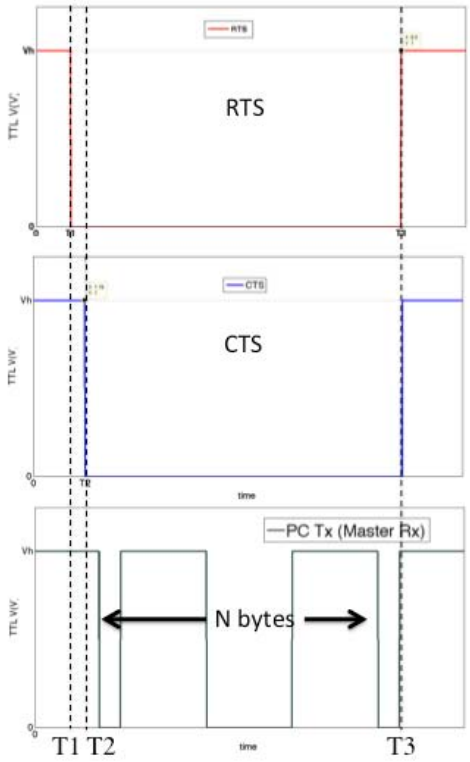


Figure IV.2 Time Diagram of communication from PC to Master unit

Functions:

- void USART_Initialise(uint16_t ubrr): this is an initialization function, it allows to set up the USART module to its desired operation conditions. The system is configured to transfer data at 38000 [bauds/s]; the frame format is 8 data bits, 1 stop bit and no parity bit and CTS and RTS pins are configured as output and external interrupt input respectively.

- void USART_Send(uint8_t data): this function sends the input byte “data” via serial.

- uint8_t USART_GetFromBuffer(void): it allows to read out the received data from the USART Buffer.

- void USART_GetRStatus(void): this function returns the current receiving status of the USART module.

- void USART_SetRStatus(void): it is used to set the USART receiving as ready. This is done to reset the status after a packet has been received and processed.

- uint8_t USART_GetRTS(void): this function reads the current level of RTS flag.

- void USART_ClearCTS(void): the function clears CTS in order to start the communication from the PC to the master MCU; it also enables the USART receiver module.

Interrupt routine:

- SIGNAL (INT1_vect): this routine is executed after any logical change in RTS. In case of a falling edge – i.e. RTS was clear by the PC - the RTS flag is set, which is then detected in the main function and enables the communication. On the other hand, in case of a rising edge, CTS is set, the USART receiver is disabled and the receiving status is updated to data received or data not received.

- SIGNAL (USART_RX_vect): this interrupt is used after a byte is received; it is stored in the USART buffer.

I2C_Master:

Files: I2C_Master.h & I2C_Master.c

Description: the communication between the master unit and the slaves based on I2C protocol is controlled and handled in this library. It also performs the interrupt routine services called when a TWI event has occurred. This library was developed based on the application note “AVR315 - TWI Master Implementation” from Atmel Corporation.

Functions:

- void I2C_Initialise (void): the initial standby state of the I2C module is configured in this function.

- void I2C_Send (uint8_t msg [I2C_BUFFER_SIZE], uint8_t msgSize): this function sends prepared messages via I2C.

- uint8_t I2C_GetFromBuffer (void): it is called when it is demanded to read out the received data from the I2C receiving buffer.

- uint8_t I2C_GetStatus (void): it returns the status of the current operation through the I2C line.

- void I2C_SetStatus (uint8_t newstate): this function is called to set a new I2C status.

Interrupt routine:

- SIGNAL (SIG_TWI): this interrupt is executed after any event on the TWI module. The event information is read from the TWI status register; then, the corresponding instructions are carried on depending on the case. Any received data is stored at the I2C receiving buffer and at the end of the interrupt, the I2C status is updated.

Master Program Flow:

The main program integrates all the processes developed by the master and also directs the program flow. It first initializes the Watch Dog Timer (WDT) with a period of approximately 16 [ms]. The WDT resets the MCU every time it reaches its period; some "check points" are added in the code to reset the WDT and prevent any reset. However, if there is a crash and the program flow is stopped, the WDT resets the MCU and recover from it; this usually happens during the I2C communication.

The next step is to initialize all the libraries - Lane Switching System, Timer, USART and I2C Master - and to enter into the program infinite loop.

The code executed in this loop is described in the flowchart shown in Figure IV.3. It first checks if RTS flag is enabled (i.e. the PC cleared the RTS line), if positive, it clears CTS, starts the communication and resets the WDT; otherwise it only resets the WDT. Once CTS is cleared, it is not possible to enter in this if, until the full communication is carried on.

When the PC sets the RTS line again, the corresponding interrupt updates the USART receiving state. If no data was received, the USART is set again as ready and the WDT is reset. On the other hand, if any data was received, it is retrieved from the USART buffer and depending on the type of instruction, a lane switching would be performed or an I2C communication would be started.

The I2C communication can generate three results: an instruction was transmitted successfully (I2C Data T), a slave status was required and received (I2C Data R) or an error occurred (I2C Error). In any case, the I2C module is set the ready status again and the result of the communication is reported to the PC (only for Data R or Error). After this, the master returns to the idle state, and it waits for the next time the PC clears to RTS line to start a new communication.

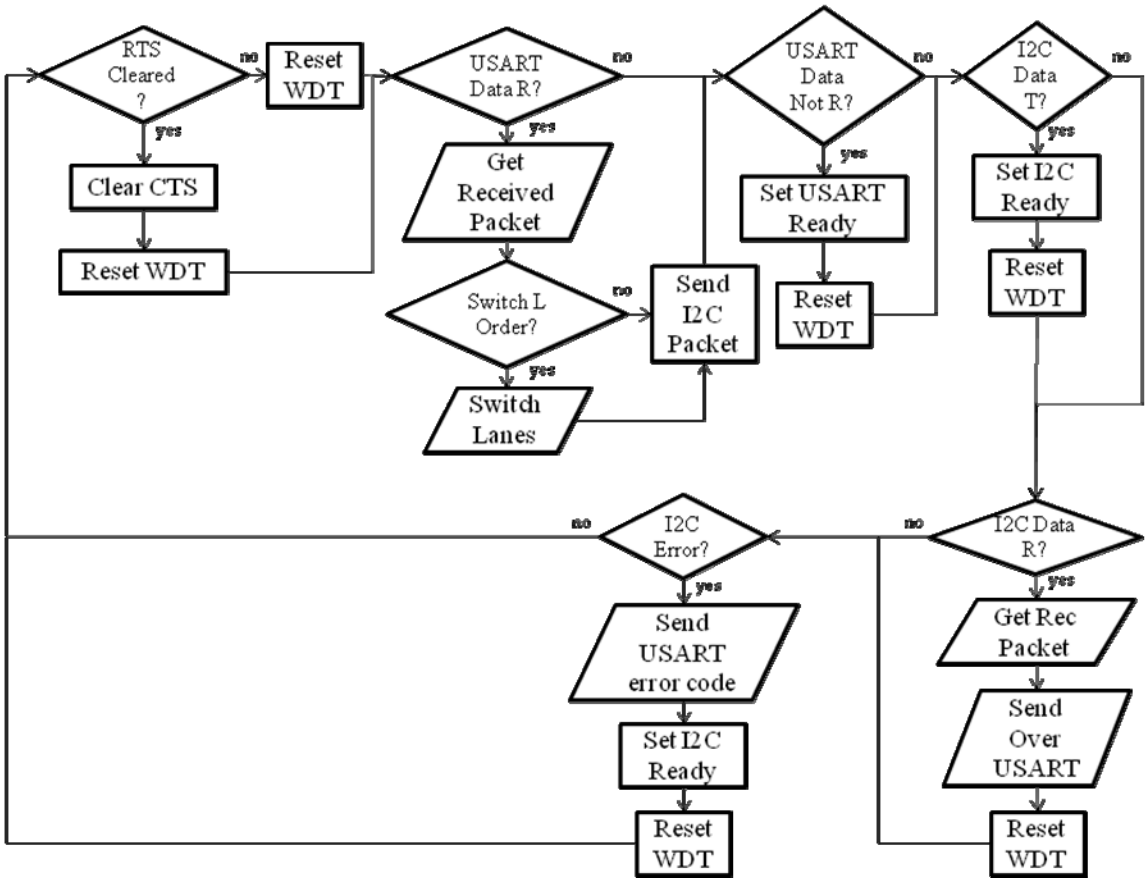


Figure IV.3 Master Main Program Flowchart

Slave Software

The Atmel ATTiny 2313 is in charge of managing all the resources related to the Slave Board. It was mandatory to develop codes based on the MCU modules in order to perform the multiple necessary tasks. Moreover, the codes were also designed and developed with the implementation of Atmel AVR Studio 4 in C language.

ATtiny2313 Modules:

The designed Slave software demanded the implementation of some of the MCU modules:

- Main clock: calibrated internal RC clock of 8 MHz

- One channel of PWM in CTC mode (ATTiny 2313 datasheet 2010, p67), associated to the Timer 0 of 8 bits
- Timer 1 of 16 bits
- 4 Input pins with external interrupts; PINB 3:0
- 9 Output pins; PORTD 6:0, PORTB6, PORTB4.
- 2 Wire Interface with I2C communication protocol

Code Structure:

The slave software was divided in five (5) main groups of tasks, which are directly related to the four key subsystems that conforms the Track Electronic Systems, besides the main class that integrates the first four and performs the program flow.

All the groups of tasks were developed individually, and all of its functions and variables were included into libraries with its code (.c) and header (.h) files. All the developed libraries are described below.

ON/OFF Power Track System:

Files: ON_OFF.h & ON_OFF.c

Description: this library controls the port that is in charge of setting on/off the power in each power track section. It performs tasks that configure and initialize the output pins. Besides it manages the status flags, which indicates the current power status of each track section.

Functions:

- **void** *Set_PowerSys* (**void**): the function sets the power pins (PORTD 3:0) as output and initializes it at high levels, i.e. turns off all the track sections associated to the MCU.

- **void** *Init_PW_Flags* (**void**): it just initializes the power status flags of each power track sections at 1, i.e. turn off all the sections.

Short Circuit Detection System:

Files: SC_Protection.h & SC_Protection.c

Description: it manages the port that monitors the Short Circuit Detection system of each power track section. There are two functions that configure and initialize the input pins. In addition, the library performs an interrupt routine associated to the interrupt vector of pin change interrupt (PCINT).

Functions:

- **void** *Set_SCPin* (**void**): it sets the pins (PINB 3:0) that monitor the short circuit detection system as inputs. Moreover, it enables the external interrupts flags of all the corresponded pins.

- **void** *Init_SC_Flag* (**void**): the function only initializes the short circuit status flags at 1, i.e. it assumes that all the sections have not short circuit when starting the system.

Interrupt routine: it runs after at least one input pin associated to the SC detection changes its voltage level. During the routine, it is determined which sections have short circuit and those that are powered and short-circuited are turned off. Additionally, the SC status flags are set; they store information about the current status of the short circuit detection system.

Infrared Identification System:

Files: IR_ID.h & IR_ID.c

Description: the function of this library is codifying and set the infrared message signals into the corresponded pins. It also implements a timer/counter of 16 bits that controls the data transfer rate of the IR messages. Furthermore, the library configures and sets a channel of the PWM module to generate a square wave signal of 38 KHz that is required for emitting the IR message signals.

Functions:

- void *Set_IR_Sys* (void): initialization function, it sets and configures the IR pins (PD6, PD4, PB6, PB4) and the PWM pin (PD5) as output pins.

- void I*nit_IR_Flags* (void): this subtask sets the initial value of all the infrared identification system flags. The counters of each IR channel starts in zero, and each flag is set in 1, i.e. channel is able to emit its determined IR message.

- void *StartPWM* (void): this function is in charge of starting the timer 0, which is set in CTC mode for generating the square wave of 38 KHz in the output pin PD05.

- void *StartTimer1* (void): the configuration and implementation of the timer 1 of 16 bits is done in this function. The timer is responsible for controlling the period required to set the different bits of each IR messages.

- void *SetIR_Codification* (void): each IR channel has a specific information to identify every track section. This information is saved on the determined data bits (8 bits) sent via infrared. For that reason, this function assigns an individual IR Identification to each IR channel, which is unique in the whole track system.

The codification has been assigned according to Table IV.6 (p54), which shows the definitive IR ID distribution that was implemented.

Interrupt routine: every one millisecond the timer 1 interrupts, it can vary depending on the data transfer rate set by the compare register. During the interrupt routine, the timer 1 is reset for configuring the next interruption and the IR messages are set to its corresponded output pins.

The IR message emitting is based on monitoring an array of counters assigned to each IR channel. The number of the counter indicates the correspondent bit that must be emitted in each IR channel.

All the IR messages have the same structure, which is based on the typical charactering frame of USART messages (Traylor 2009): a start bit, a stop bit, no parity bit and the data bits; the last information is the only unique data assigned to each track section of the whole system. Additionally, 15 bits of low logical level were added to the IR messages in order to avoid that the IR receiver would overlap two consecutive IR messages. According to this information, it was decided to follow a pattern that indicates which kind of data is emitted simultaneously with the current counter of each IR channels, see Table IV.5 and Figure IV.4

ID Counter	**Bits**
0	Start bit
1 – 8	Data bits
9	Stop bit
10 - 25	Safe zone

Table IV.5 Emitted data according to ID counter

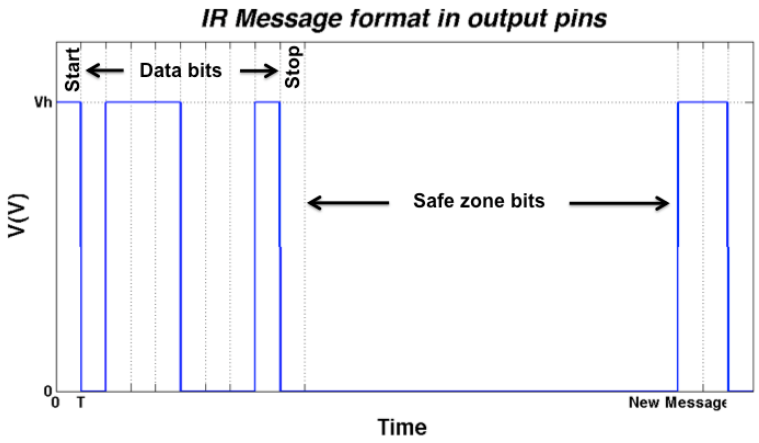


Figure IV.4 IR Message format

I2C Slave:

Files: I2C_slave.h & I2C_slave.c

Description: this library handles all the resources that are necessary to perform the I2C communication between the slave MCUs (ATtiny2313) and the master MCU (ATmega88). It also performs the interrupt routines associated to USI interrupt vectors during different events of to the I2C protocol. This library is based on the example project "example I2C / TWI program for the Attiny 26" developed by Huey Hano.

Functions:

- **void** *I2C_init*(**void**): it sets up the USI module as an I2C slave in its initial standby status. The function configures the corresponded SCL and SDA ports as inputs and it does the required modifications to the USI control register.

- **uint8**_t *I2C_GetFromBuffer* (**void**): this function allows to retrieve the received instructions from the receiving buffer.

- **uint8**_t *I2C_GetStatus* (**void**): it returns the current status of the I2C communication.

- **void** *I2C_SetStatus* (**uint8_t newstate**): it allows setting a new I2C communication status.

Interrupt routines:

- SIGNAL(SIG_USI_START): this interrupt is used after a start condition is detected. It activates the USI module by enabling the counter overflow interrupt.

- SIGNAL(SIG_USI_OVERFLOW): this routine is executed every time there is an overflow in the USI counter and it is used to handle the I2C communication. In order to send or receive a data or address byte, the counter is written to zero (‘0’); and for the acknowledging process it is set to six (‘6’).

At the beginning of the interrupt, the communication status is read and the correspondent instructions are executed; any received data is stored in the receiving buffer and then, the status is updated.

Slave Program Flow:

The main program runs all the processes involved to the slave modules. Firstly, it initializes and configures each associated subsystem: ON/OFF power system, SC Detection system, IR identification system and I2C communication. Then, it is enabled the interruptions, and the two-implemented timers are set: Timer0 is configured in CTC mode for generating the square wave at 38 KHz, while the Timer1 is started with its demanded settings of operation.

Before starting the infinite loop that runs during the rest of the program, it is developed a first loading of the current status of the system on the transmitting buffer of I2C. Then, once it has been started the infinite “for” loop, it is checked the last received status of the I2C. If it is not ready, the information presented in the buffer is gotten and processed, in order to determine the corresponded instructions and theirs IDs.

The information obtained from the instructions allows turning on/off the power in each section, but taking in consideration the presence of possible short circuits in any of these sections. After following the actions received from the instructions, it is reloaded the current status of the system on the I2C transmitting buffer. Finally, at the end of the infinite loop, it is always monitored if any short circuit has been handled and loads it into the status of the system.

The flow of the program is often interrupted by different events related to the managed subsystems. The interruption routines are described previously in the different libraries that involve the code structure of the program.

The definitive Slave Program flow can be summarized in the flowchart illustrated in Figure IV.5.

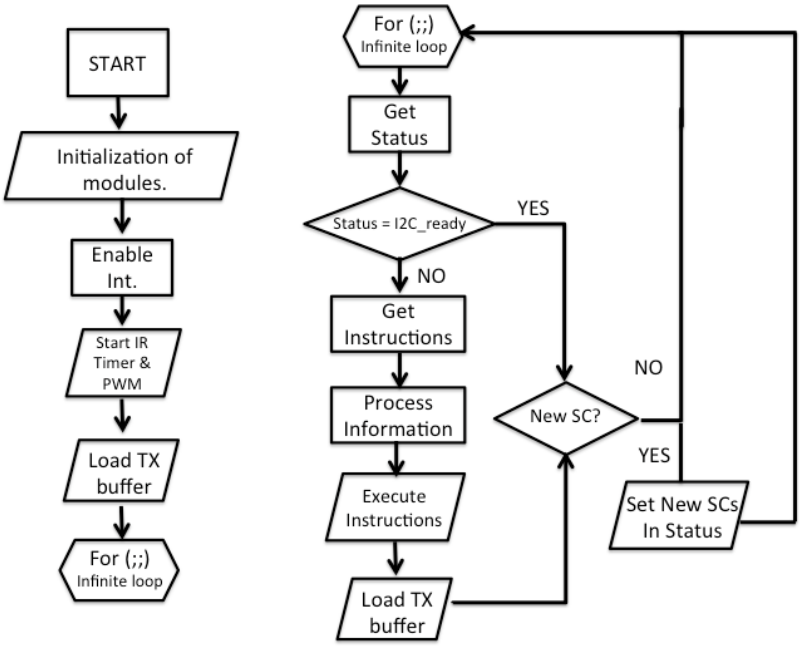


Figure IV.5 Flowchart of the Slave Main Program

Slave Software Considerations:

There are two groups of constants that identify each track section of the whole electronic track system. Firstly, the I2C communication protocol requires a unique identifier number that would represent each slave unit. The high quantity of slaves connected to the common I2C bus of the master unit has forced to take this decision. The constant that identifies each slave belongs to the I2C_slave library and it is called I2C_ID

Second, each track sections must emit a defined infrared message that would also identify its corresponded sections as unique. These identifiers have been storage into an array of integers (*TrackID [Ntrack])* that keeps the ID of each section controlled by the slave. Moreover, the initialization function "void *SetIR_Codification* (void)" from the IR ID library is in charge of defining the IDs.

Finally, the I2C and IR identifiers were defined according to the Table IV.6, which shows the corresponded IDs that were assigned to each slave unit and each track section.

INNER TRACK SECTION SLAVES			**OUTER TRACK SECTION SLAVES**		
Section	**I2C ID**	**IR ID**	**Section**	**I2C ID**	**IR ID**
A	1	1	A	2	5
		2			6
		3			7
		4			8
B	3	9	B	4	13
		10			14

		11			15
		12			16
C	5	17	C	6	21
		18			22
		19			23
		20			24
D	7	25	D	8	29
		26			30
		27			31
		28			32
E	9	33	E	10	37
		34			38
		35			39
		36			40
F	11	41	F	12	45
		42			46
		43			47
		44			48
G	13	49	G	14	53
		50			54
		51			55
		52			56
H	15	57	H	16	61
		58			62
		59			63
		60			64
I	17	65			
		66			
		67			
		68			
J	18	69			
		70			
		71			
		72			
K	19	73			
		74			
		75			
L	20	76			
		77			
		78			
		79			
M	21	80			
		81			
		82			
		83			
N	22	84			

		85
		86
		87

Table IV.6 I2C and IR identifiers table

V. VEHICLE MECHANICAL IMPLEMENTATION

Overview

In order to present the concept of "Slide-in" power-transfer as a viable solution to the energy problem of modern transportation the goal is to develop scaled vehicles that represent a large portion of today's transportation methods. For this purpose, two types of vehicles are developed: a car-model and a truck-model.

The models have to meet the following criteria:

1. Be able to drive in a standard Scalextric-track, guided by a slot without requiring modifications to the dimensions of the slot.
2. Be able to draw power from a standard Scalextric-track without modifications to the standard Scalextric rails.
3. Be able to drive without external power-supply for a limited amount of time, in effect draw power from an internal battery,
4. Be able to drive and steer without a guiding slot.
5. Be able to autonomously control speed
6. Be able to display the current state of charge of the internal battery.
7. The external dimensions of the vehicle may not exceed the maximum dimension allowed for two vehicles to be able to drive simultaneously; each guided by a separate slot of the two adjacent slots on a standard Scalextric track.
8. Be easy to assemble and disassemble in order to facilitate rapid repairs in case of malfunction.

To meet these requirements the design of the car can be broken down in several global subsections, namely:

- Maximum external dimensions and proportioning
- Drivetrain
- Steering
- Guiding mechanism
- Housing of mechanical, electrical and electromechanical components

It is important to note, however, that even though these are separate design tasks; the size of the vehicles requires a highly integrated design-approach. Therefore each aspect of the design has to be derived with the other aspects in mind.

Maximum external dimensions and proportioning

The maximum space the vehicles are allowed to occupy is derived from the criteria of allowing two vehicles to pass each other following two separate adjacent slots on a single track. This holds true for both straight lanes and the curve-shaped track-sections with the smallest radius utilized.

To create a distinct visual differentiation between the car and the truck, the truck is slightly larger in width. Therefore the truck is designed to be as wide as the maximum space allows while the car is designed to be as narrow as possible in comparison to the truck.

The lengths of the vehicles are important in regards to traveling around corners. Mechanical arms positioned close to the front section guide the vehicles. These arms consist out of two parts, which are allowed to rotate freely around a common rotational axle. Because of this, the vehicles travel at an angle negative to the tangent of the curve. This results in the length of the vehicles being a determinative factor for two vehicles being able to pass each other in curves.

Because of the constraint already imposed on size in regards to the width of the vehicles, the lengths of the car and the driving portion of the truck are the same. It is considered that the larger width of the truck and also the added total length when the trailer is attached are sufficient visual cues to differentiate the two types of vehicles.

Drivetrain

The drive consists of 5 main parts:

- Motors
- Transmission
- Wheels
- Bushes
- Axles

In order to simplify development, standard Scalextric-parts were utilized as much as possible. In the truck all parts of the drivetrain are standard Scalextric-parts whereas in the car some custom parts are used in order to allow reduction of the width of the vehicle.

Motors:

The motors used are aftermarket motors obtained from Scalextric and are partly chosen because of the slim design, but also because they were cost-effective and available in

large quantities at the time of purchase. Each vehicle has two motors. The motors are responsible for forward and backward momentum and steering. The motors are mounted in-line with the vehicles in order to conserve lateral space. The specifications for tourque and power output were impossible to obtain before purchase but were considered to be sufficient because of the weight of the vehicles they were originally designed for.

Figure V.1 Scalextric aftermarket motor C8426

Transmission:

The transmission is a 90-degree transmission consisting of a pinion and a contrate gear. The transmission has to be a 90-degree transmission in order to allow the motors to be mounted in-line with the vehicles. The implementations of the transmission in the vehicles utilize a variety of gear ratios. The reason is that gears of a single ratio are unavailable for separate purchase. They only come in bags of five different ratios with two gears of each. The gears were rationed in such a way that the trucks received the greatest gear ratios in order to allow for the greatest possible torque output.

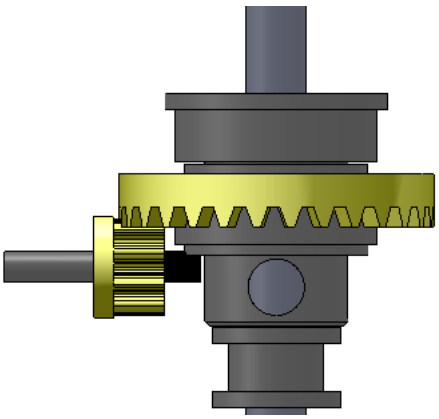


Figure V.2 Transmission

Pinion	Contrate gear	Ratio	Used in
8	29	3.625:1	Truck
9	28	3.111:1	Car
10	27	2.700.1	Not used
11	26	2.364:1	Not used
12	25	2.083:1	Not Used

Table V.1 Available transmission gear ratios

Wheels:

The wheels are standard Scalextric-wheels model C8409, which come in packages including rim, tyre screws and a tool for assembly.

Figure V.3 Wheels, rims, screws and tool. image courtesy of Scalextric Inc.

Axles:

The axels are for the most part standard Scalextric-axles. Some axles are manufactured out of 3 [mm] round-bar because of the lack of axles of sufficient length.

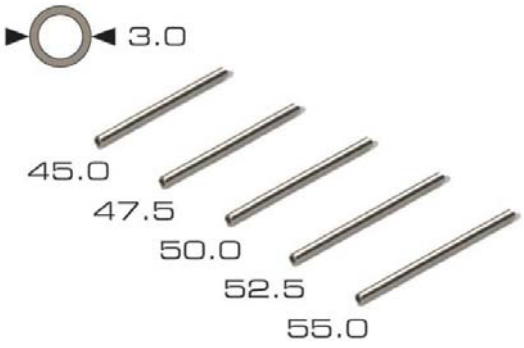


Figure V.4 Scalextric axles of various sizes, images courtesy of Scalextric Inc.

Bushes:

There exist two types of bearings in the vehicles. One is a standard Scalextric type of manufactured out of brass and the other out of SLS – sintered plastic. The plastic bush is used only in the car and is utilized to support one of the axle-attachments directly instead of the axle. This allowed a total of 6 mm of lateral space to be saved.

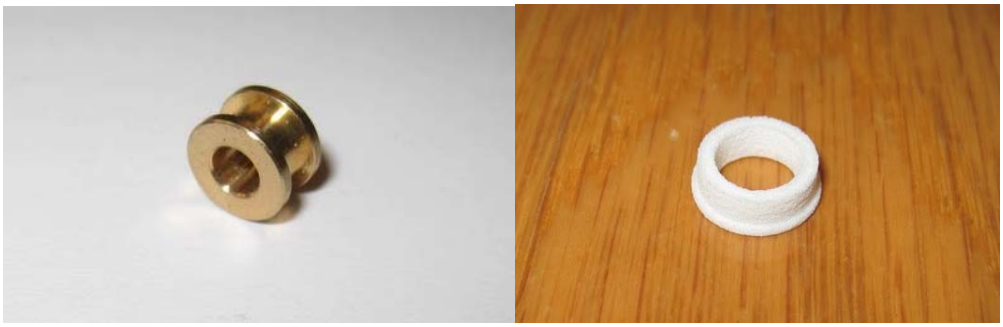


Figure V.5 Scalextric brass bush to the left, SLS-sintered bus to the right

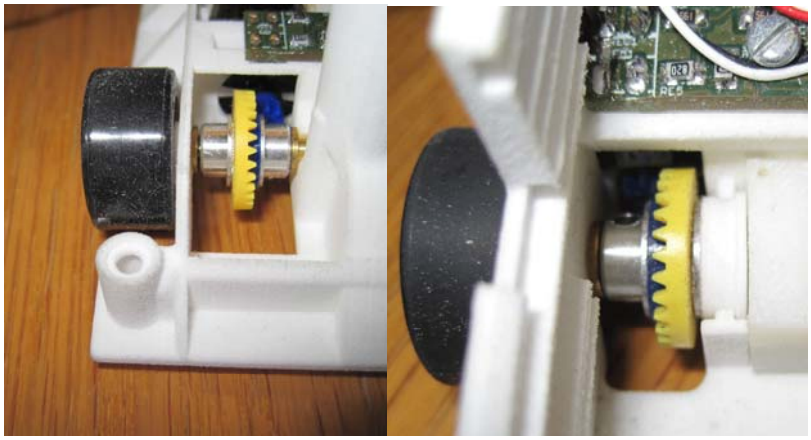


Figure V.6 Assembled: Scalextric brass bush to the left, SLS-sintered bus to the right

Steering

When the vehicle is connected to a slot on the track the vehicle is steered by the slot and has no influence on its direction of movement other than forward or reverse acceleration. When the vehicle is disconnected from the track it utilizes the rear wheels for steering. The principle relies on the fact that as a four-wheel vehicle travels in a corner, the wheels travel along paths that, if measured from the point where the vehicle enters the corner to where it exists, are of different length. As these wheels have to complete their respective paths in roughly the same amount of time the result is that they have to travel at different velocities. If the wheels are induced to travel at different velocities by applying differing amount of torque upon them the result will be that the car is driven to travel in a circular path.

This type of steering is utilized in the car to conserve space. A traditional method of steering, which would involve a linkage system in the front section of the car, would occupy too much space and not leave room for the servo system required to implement the system utilized to connect to and disconnect from the road.

Guiding mechanism

The guiding mechanism serves two purposes. One is to follow the slot of the Scalextric track and transfer lateral force to the vehicle in order to make it change direction as the curvature of the slot varies. The other is to facilitate a method to transfer electric of energy from the metallic lanes on the track to the vehicle. The mechanism consists of three main parts:

- The servo
- The link
- The pickup

Servo

The servo is a small servo of type Hitec HS55. It was the smallest servo readily available for purchase from a retailer at the time of development. The servo comes with an arm, which attaches to the output shaft. The arm has a series of small holes prefabricated that can be used to attach linkages, axles or other mechanical components.

Figure V.7 Servo type Hitec HS55

Link

The link is manufactured from SLS-sintered plastic and has three main features:

1. Cylindrical socket that is fitted on a corresponding axle on the chassis.

2. Slot for transferring the rotating motion of the servo to the linear motion of the link
3. Attachment for the pickup

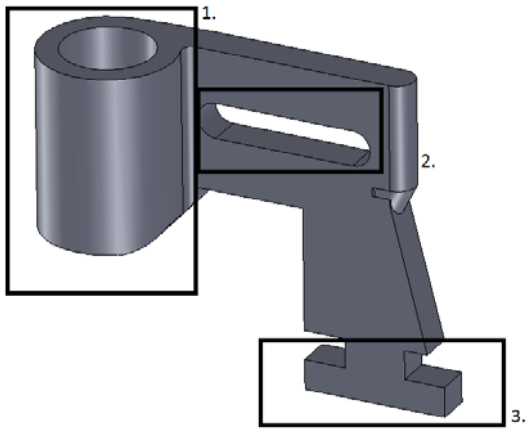


Figure V.8 Linkage assembly

Pickup

The purpose of the pickup is to guide the vehicles through the slot of the track and at the same time hold the special braids that connect to the electrified lanes on the track. The pickup is allowed to rotate around its axis in order to reduce the power required for the vehicles to travel through corners and also minimize the risk of unwanted exiting of lanes by the vehicles. The pickup is manufactured from SLS-sintered plastic.

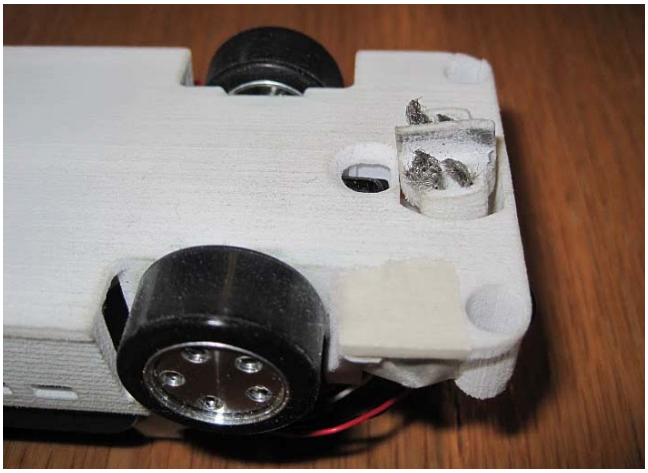


Figure V.9 Pickup as viewed from beneath the car

Figure V.10 Braid

Assembly and function

The pickup attaches to the link by aligning the beam with the rectangular hole on the pickup and then rotating it 90 degrees.

Figure V.11 Linkage assembly

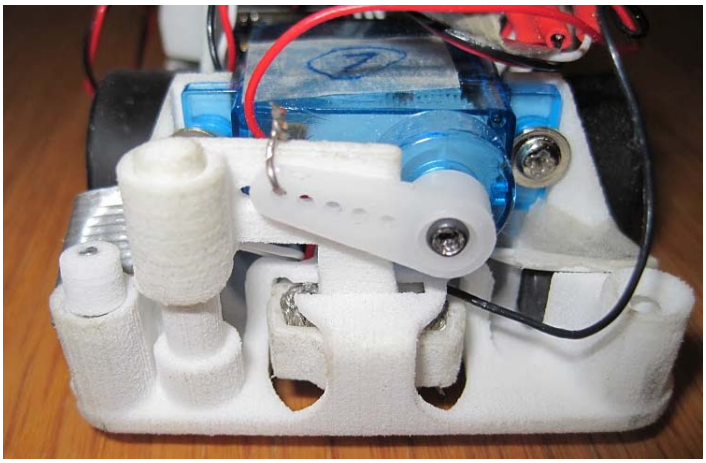


Figure V.12 The mechanism assembled in the car

The mechanism functions in such a way that when the servo rotates, the rotational movement is transformed into a linear vertical motion by the constraints imposed by the vertical cylinder and the horizontal slot. This allows the vehicles to retract the pickup to allow for battery-powered driving without a guiding slot.

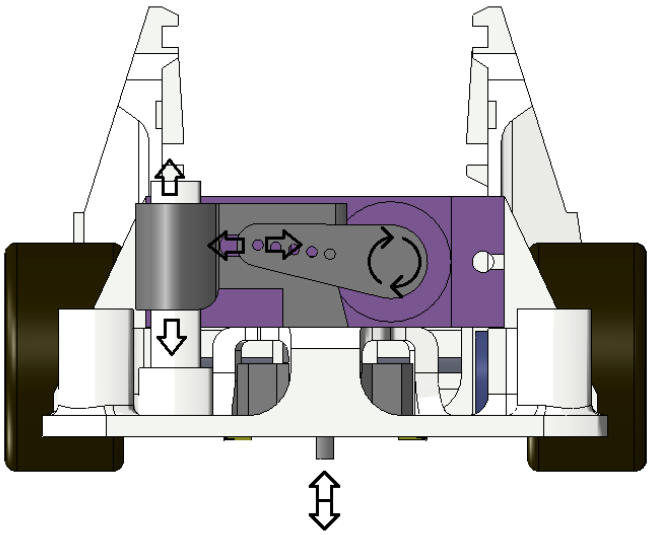
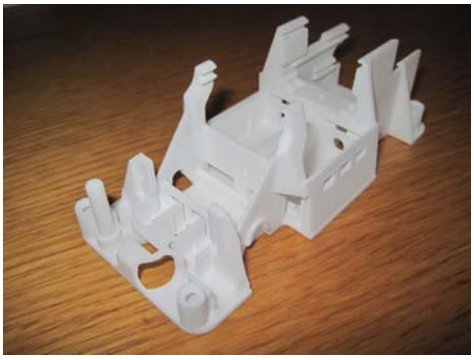


Figure V.13 Function diagram of guiding mechanism

Chassis

The chassis is designed in such a way that a minimum amount of separate pieces are utilized. The chassis for the car model is essentially composed out of a single piece while the chassis for the truck is composed out of two pieces connected at a joint.



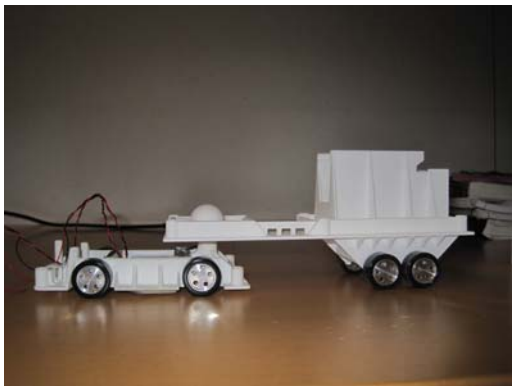


Figure V.14 Car and truck chassis

The chassis were designed iteratively as the internal component selection and design was determined. Initially, the chassis was designed with only the drivetrain, battery and servo in mind. An effort was made to leave as much of a continuous space as possible available for the internal PCBs which would come to be designed in a later stage. This allowed for prototyping and manufacturing selection to be made before the final electronic design was finished. The method used for attaching the components varies from component to component.

Assembly

PCBs

The PCBs are either attached solely by sliding them into slots or attached with fasteners. The slots manufactured in such a way to allow play with an upper tolerance of 0.2mm and a minimum of 0 mm in reference to the thickness of the PCBs. The friction between the SLS-material and the PCB is enough to keep the PCBs in place during normal operation. The PCBs for logic, motor-control and power-conversion are mechanically connected to each other by header-connectors. When they are assembled they can slide in to the slots in a unified manner. There are two PCBs that utilize a fastener: the PCB that holds the IR transmitters and receivers and the PCB for the magnetic field deviation detection circuit. They use M2 machine screws.

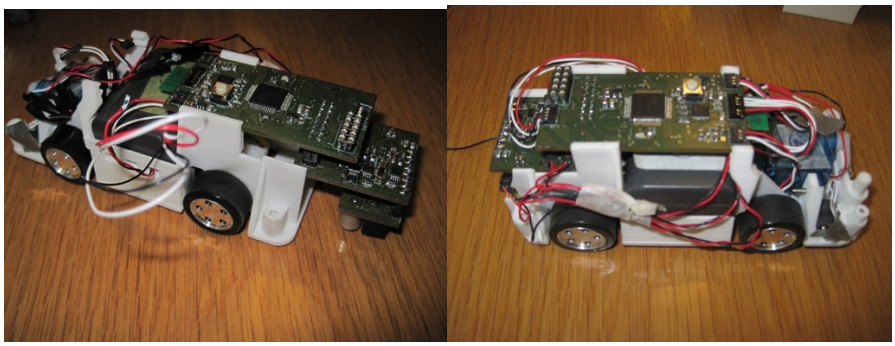


Figure V.15 Main logic, motor-control and power-conversion PCBs

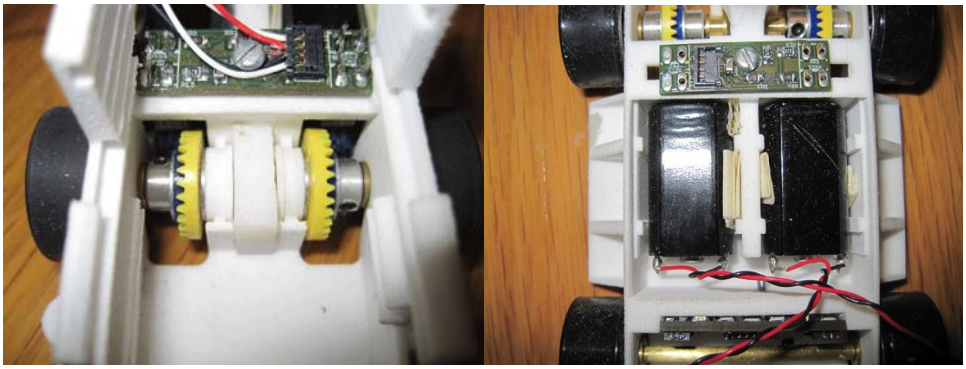


Figure V.16 Speed detection PCB in the car (left) and the truck (right)

Battery:

The battery is attached by using the ledges located at two sides. There is a corresponding slot in the chassis in which the ledges fit with a tight tolerance. The friction attained by this ledge-slot interface is enough to keep the battery in place. On one side of the chassis there are three holes that correspond to the three terminals available on the battery and line up when the battery is inserted.

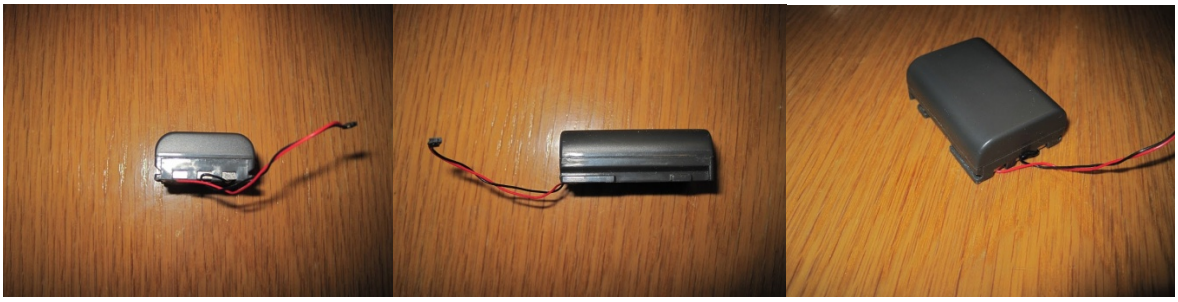


Figure V.17 Battery from various views

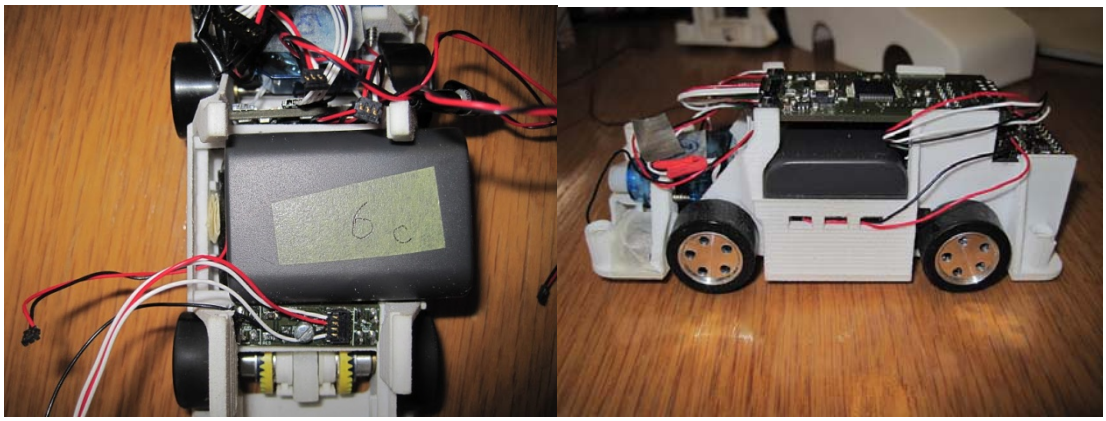


Figure V.18 Battery being assembled (left) and assembled (right) in the car

In the truck model, the battery is located in the trailer because of the lack of space in the truck.

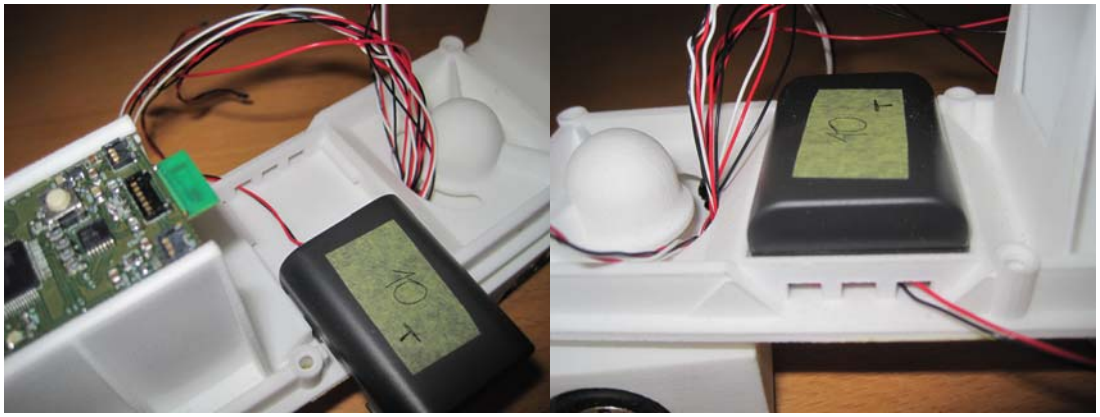


Figure V.19 Battery being assembled (left) and assembled (right) in the truck

Motors:

In the car the motors are located beneath the battery, which serves to keep the motors from moving in the vertical direction. The motors have a small cylindrical protrusion on one side where the axle exits. This protrusion fits into the chassis and serves to define the motors position in two dimensions, and the wall of the motor serves to define the third. On the side where the motors terminals reside it is supported by wedge-shaped protrusions. The shape of the protrusions allow for the motors to be inserted with gradual increase of applied force. In the truck the battery is in the trailer. Because of that the motors in the truck are restrained from moving in the vertical direction by paper-material placed between the shell of the truck and the motors.

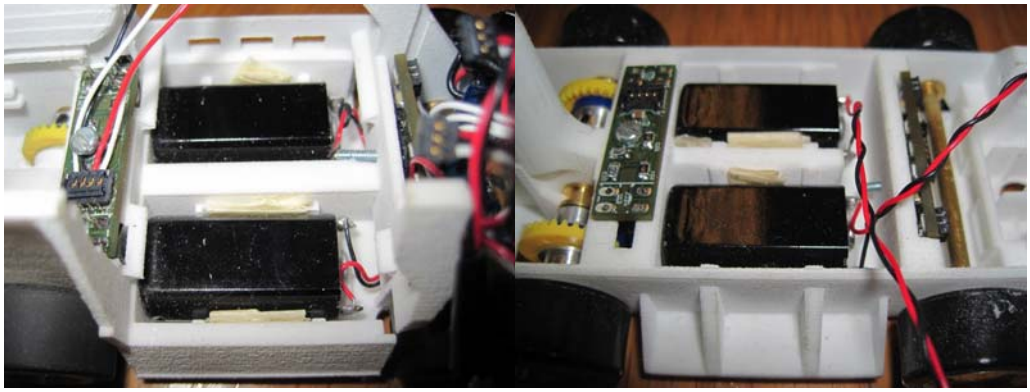


Figure V.20 Motors inserted in the car (left) and the truck (right)

Bushes:

Bushes are the interface between chassis and rotating axles. On the interface facing the chassis there must be maximal possible static force in order to keep the bushes from rotating. On the other side the friction must be kept to a bare minimum in order to give the axle smooth operation and minimize losses. The bushes are placed in sections that have a

circular shape with a sector cut out through which the bush is pushed in place. The sector, which is cut out, is smaller than the part that is solid material. This way the bush, when inserted, “snaps” into place. To minimize the risk of the bush falling out and increase the force required to make the bush rotate in its seat a small amount of Loctite-glue is applied.

Figure V.21 Brass bush used in the vehicles

The plastic bushes that are located in the car are held in place in a special manner. There is a piece between the two bushes that is inserted after the bushes are placed. This piece exerts frictional force on the plastic bushes and keeps them in place. The plastic piece itself is attached by a 2mm machine screw that also holds the speed-detection PCB in place.

Figure V.22 Plastic bush used in car

Servo:

The servo is fastened by two 3mm machine screws. The screws press on flanges on the servo and thread into flanges in the chassis behind the servos’ flanges. To keep the servo from tilting it sits on a support structure located below it.

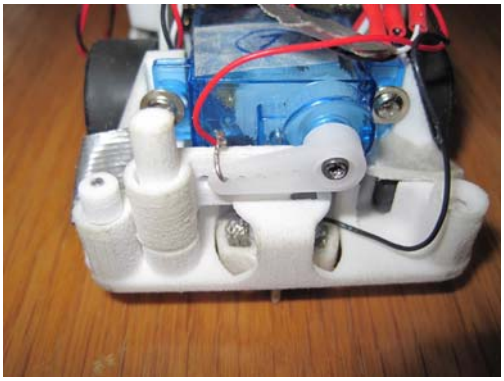


Figure V.23 The servo is attached with two 3 mm screws

Coils:

The coils are placed in small cradles shaped to the exact measurements of the coils. Because of the rubbery surface of the coils, the surface deforms slightly when pushed into the cradle and creates a static force strong enough to keep them in. Small patches of tape are applied as a reassurance to keep them from falling out.

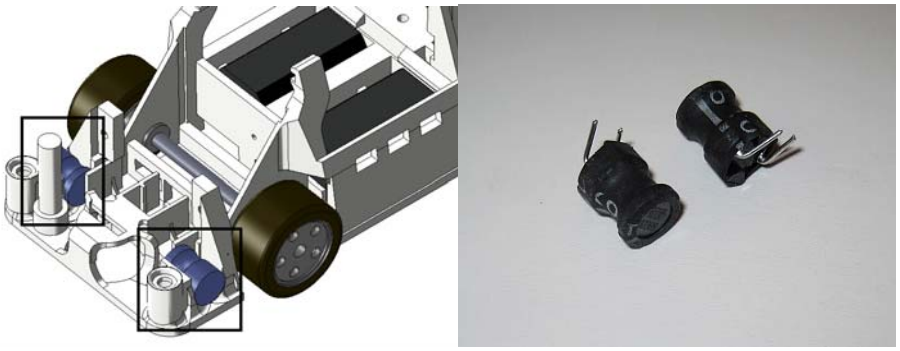


Figure V.24 The coils in a rendered image of the car (left, marked with black rectangles) and a photographed image of the coils before assembly with car (right)

Shells

The body of the vehicles, or the shells as they henceforth will be called, are the only part of the vehicles that are visible to people. Because of that it is very important that they are designed in such a way that they are visually pleasing, but also conform to their surroundings in such a way that they don't deter from the overall look and feel of the model. There is also a requirement that the model should in some way be able to display the battery state of charge in a intuitively visual manner.

Besides the aesthetic part of the design of the shells, they have to be structurally sound. As a person picks up a model there must be no deformation given a reasonable applied

force. The shells also have to withstand a drop from a reasonable altitude and mainly protect the expensive components within

Aesthetic design

It was initially conceived that the shells would resemble some existing model in the line of vehicles that Volvo Trucks and Volvo Cars produce. Due to the time-constraint it became evident that it would be hard to achieve that goal. Instead an alternate route was chosen where a “faceless” design-language was chosen. The vehicles were determined to be white in colour and only symbolically represent cars and trucks in their shape.

In order to show the battery state of charge it was decided that a multi-colour LED was going to be used. In order for the light to be visible a circular hole was created in the shells above the position of the LED. In the car this hole resides on what represents the “roof” of the car and in the truck it’s located in the top of the shell of the trailer.

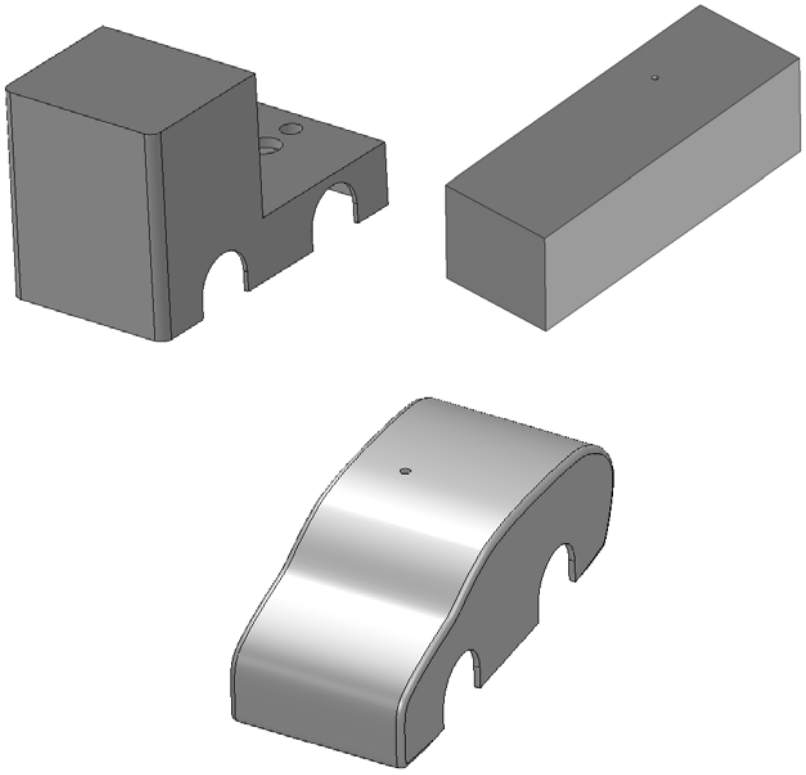


Figure V.25 Computer-generated images of the shells for the truck(top) and car (bottom)

Structural design

The shells of the vehicles are not only the only visible part of the vehicles but also the interface at which human contact will occur. In order for the vehicles not to feel structurally weak when they are picked up the shells are designed in such a way that they do not deform

at normal squeezing pressures. A consideration must also be taken to the weight of the design, so care has been taken to not add too much material at the reinforcements.

The reinforcements are mostly designed as rib-reinforcements. They are intuitively placed at sections where the shells cover relatively large volumes without support. Supports have also been added at sharp turns of the shape in order to relieve the tension build-up at those areas during deformation.

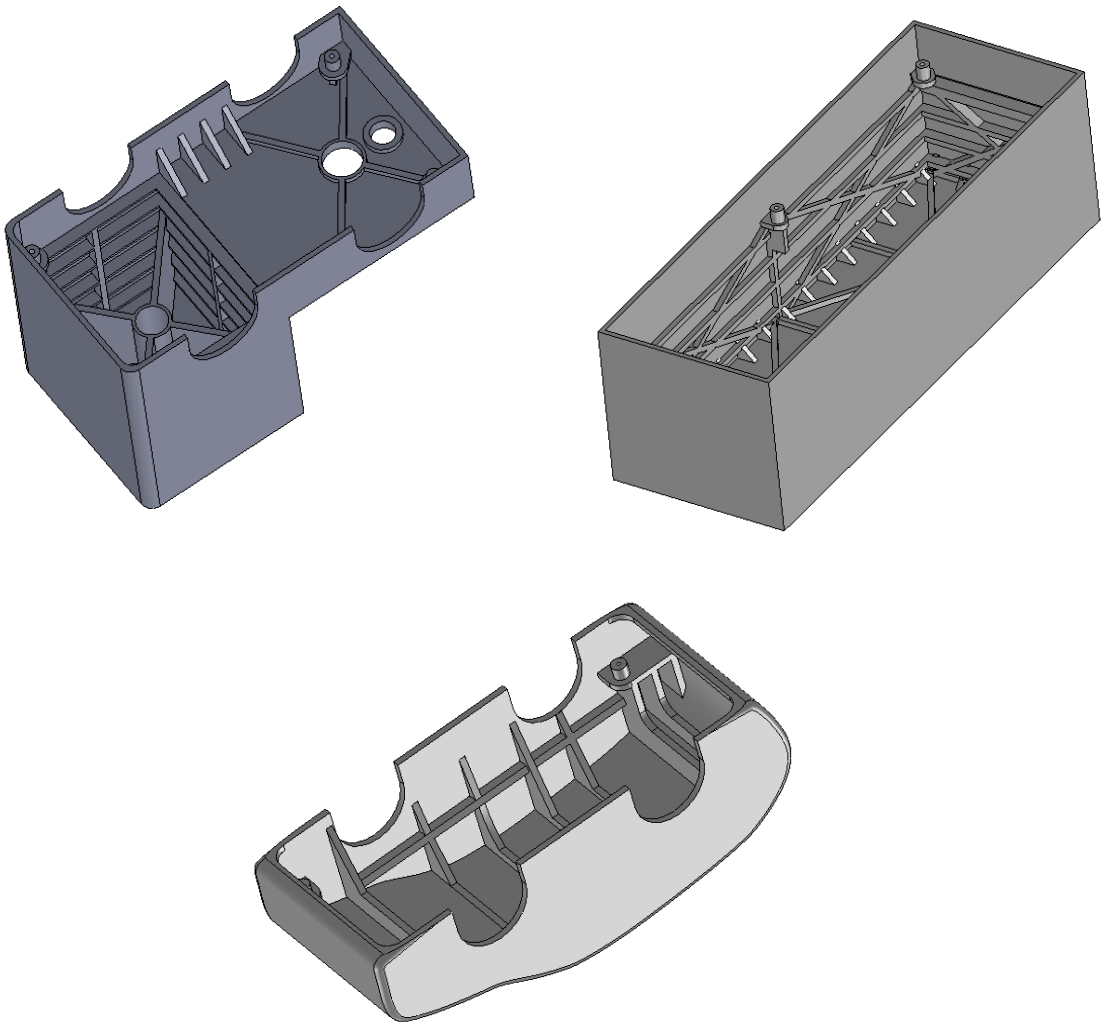


Figure V.26 Computer-generated images of the reinforcements for the truck(top) and car (bottom)

The shells are attached to the chassis using metal screws. For this purpose small cylindrical shapes were created with small indents in the center of the circular caps. The indents were used as a guide when the holes for the attachments were tapped. The cylinders were heavily reinforced because they were going to carry the entire weight of the shell and also absorb any impulses during impacts.

VI. VEHICLE ELECTRONIC HARDWARE

Overview

The vehicles electronic diagram is shown in Figure VI.1. The main board has an Atmel AVR32 UC3B0256 32 bits MCU for running all the processes in the vehicle and is also equipped with a Bluetooth module to allow communication with the PC. This board is directly connected to an infrared receiver, to read the IR identification (IRID) emitted by the track in order to identify the vehicles current position; it is also connected to a servo-motor that takes up/down the electric contacts and the guiding mechanism of the vehicles on the slot tracks.

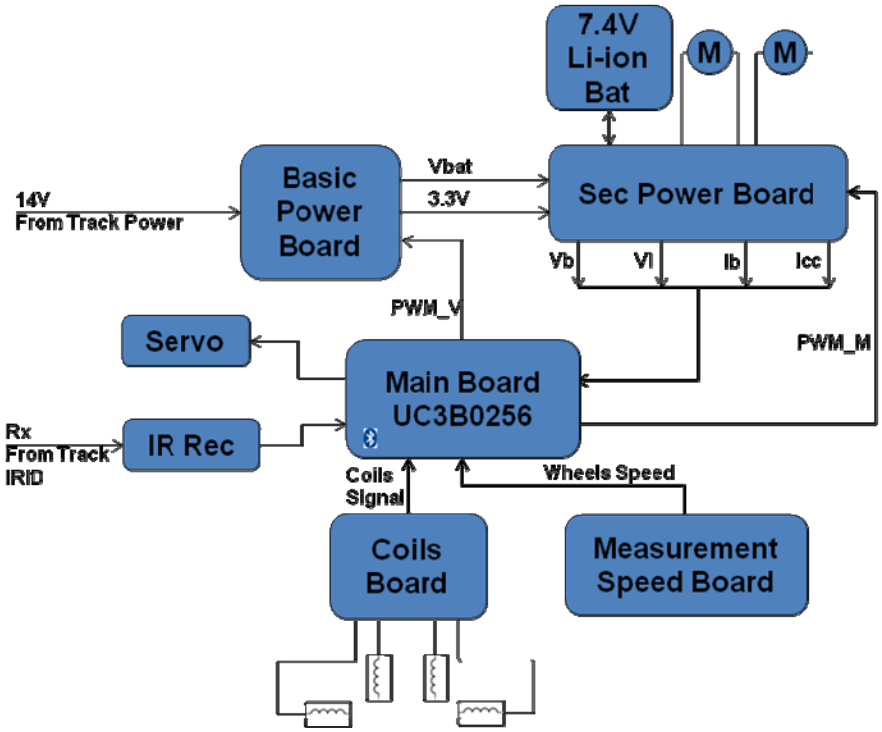


Figure VI.1 Vehicle Electronics Block Diagram

The Basic Power Board consists of two DC-DC converters; the first one was built from discrete components and its output (Vbat) is controlled by a PWM line from the Main Board; it is implemented for charging the vehicles battery. The second converter has a fixed output of 3.3 [V] to source most of the ICs in the vehicle. Even though the 3.3V and Vbat lines are only connected to the Secondary Power Board in Figure VI.1, they feed all the boards in the vehicle.

Moreover, the Secondary Power Board is connected to a 7.4V, two cells, lithium ion battery, which is the vehicles main power supply; this board is also able to measure the current consumed by the battery (Ib) and its voltage (Vb), the line voltage in the track (Vl),

and the current consumed by the rest of the vehicles electronics (Icc); these measurements are sent to the main board. Besides, this board has two full H-bridges that control the two DC-motors on each vehicle, which control input are PWM signals that come from the Main Board.

In order to have a feedback of the motors speed, the Measurement Speed Board was implemented. In addition, to measure the vehicles deviation from the center of the road, when it is driving over non-slot track sections, two pick-up coils are used. Finally, the Coils Board was implemented to amplify and condition the coils signals the MCUs ADC (Analog to Digital Converter) can interpret them.

Battery

A battery is used as the main power supply of the vehicles when there is none electrical connection to the track and it has to accomplish different requirements. First, it has to be small so it can be placed inside the vehicles and it must be available in a store that dispatches to Sweden.

After searching for the batteries available in the market, four types were found: Lead Acid, Nickel Metal-Hydride (Ni-MH), Nickel Cadmium (Ni-Cd) and Lithium Ion (Li-Ion) batteries. The Lead Acid batteries were discarded given that they do not accomplish the size restrictions; the smallest available size was bigger than the vehicle itself.

Moreover, the battery must provide enough current to run the two motors and all the electronic systems. The motors current is limited by the maximum value rated for the H-bridge driver, which is 600 [mA] per channel (L293DD Datasheet 1996), for a total of 1.2 [A]. Regarding the electronics, the highest consumption comes from the RN-41 Bluetooth that, according to its datasheet (2009), sinks up to 100 [mA] when it is transmitting.

For the rest of the circuitry, 300 [mA] was estimated as the top current consumption; this value is considered to be much higher than the actual one, but it was decided to avoid spending unnecessary time in further calculations, simulations or measurements. Adding up the described quantities, the maximum current to be sourced by the battery is estimated in 1600 [mA].

Besides, the vehicle needs autonomy to drive for a certain time without charging, but if there is too much energy stored in the battery, the effect of charging and discharging will not be noted when showing the functional system. The parameter that determines the amount of energy stored by a battery is the Amper-hour rating, which states the maximum current (C)

that can be sourced by the battery in one hour. For a battery of 800 [mAh] sourcing 1600 [mA] (i.e. discharging at 2C) the autonomy is near 30 minutes, depending on its efficiency; which is considered as a proper value to satisfy the specifications stated above.

The final requirement is that the output voltage must be high enough to drive the motors without using any step-up converter. The chosen motors (Scalextric C8426) are rated for 12 [V] and 30,000 [RPM] and are used for racing toy cars, but since high speeds are not needed for this project, 6 [V] was decided to be the minimum voltage.

Ni-MH, Ni-Cd and Li-Ion batteries can satisfy all the requirements stated above, the difference comes in their charging method (Simpson 2007).Ni-MH and Ni-Cd batteries can be charged applying a constant current near 1.2 C, which can fully charge it in one hour; but, an end-of-charge detection circuit must be used to prevent overcharging (Simpson 2007). It is recommended to implement the end-of-charge detection by measuring temperature changes in the battery.

On the other hand, Li-Lion batteries are commonly charged applying a constant voltage of 4.20 ± 0.05 [V] per cell and limiting the current sourced to the battery to 1 C (Simpson 2007). In comparison, Li-Ion batteries are easier to charge and require less hardware (and thus less circuit space) than Ni-MH and Ni-Cd batteries, and for this reason it was decided to use this type of battery.

To accomplish the requirement about the output voltage, a two-cells Li-Ion battery is needed, which has a nominal voltage of 7.4 [V]. Based on market availability, the chosen battery was part number 42223 from Kjell and Company, type Cannon NB-2LH, Li-Ion, 7.4 [V] and 800 [mAh].

Main Power Board

In order to convert the track voltage to the levels handled by the cars electronics, two DC-DC converters were implemented in this board; the first one to provide the voltage to charge the battery and the second to set an output of 3.3 [V] that is required for the electronic circuits. Since the battery's charging process requires a controllable voltage, a buck converter was implemented, which output voltage is controlled by an incoming PWM signal.

The schematic for the buck converter according to Hart (2001) is shown in Figure VI.2.

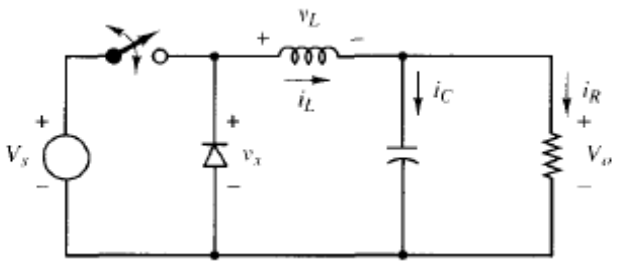


Figure VI.2 Buck Converter Schematic

Taking three assumptions in consideration - an infinite output capacitance, ideals diode and transistor and that the current in the inductance i_L is always positive - the circuit can be described by the following equations (Hart 2001).

$$V_O = V_S . DC \tag{VI.1}$$

$$\Delta i_L = \frac{(V_S - V_O) . DC}{L . f} \tag{VI.2}$$

$$\overline{i_L} = i_R \tag{VI.3}$$

$$i_{Lrms} = \sqrt{\overline{i_L}^2 + \left(\frac{\Delta i_L}{2\sqrt{3}}\right)^2} \tag{VI.4}$$

$$i_{Lmax} = \overline{i_L} + \frac{\Delta i_L}{2} \tag{VI.5}$$

For these equations, V_O and V_S are the output and input voltage respectively; DC is the fraction of time that the switch is closed; L is the inductance value and f is the switching frequency. The highest V_O is the voltage of a fully charged battery, and for a two cells Li-Ion battery it is 8.4 [V]; besides, V_S was decided to be 14 [V], since it is the maximum voltage rated for some of the electronic components. Moreover, the maximum i_R is 2 [A], corresponding to 0.8 [A] to charge the battery and an estimated current consumption of the motorso of 1.2 [A] when they are running.

To accomplish the initial assumption Δi_L must be smaller than $\overline{i_L}$, resulting in the following restriction (Hart 2001).

$$L \geq \frac{(1 - DC) . V_O}{2 . f . i_R} \tag{VI.6}$$

On the other hand, the output capacitor is not really infinite; therefore it generates a ripple in the output voltage, which can be approximated by the following equation (Hart

2001). This ripple must be smaller than 50 [mV], according to the charging instructions of one of the Li-Ion batteries that was considered to be used (Canon NB-2LH datasheet 2010).

$$\Delta V_O = \frac{(1 - DC).V_O}{8.L.C.f^2} \qquad \text{(VI.7)}$$

To decide upon a switching frequency, different values were evaluated to obtain a corresponding minimum inductance according to Equation VI.6. For a switching frequency of 25 KHz the minimum inductance required is 33.6 [µH] and for 40 KHz it is 22.5 [µH]. At high frequencies ΔV_O and Δi_L tend to reduce, hence a switching frequency of 40 [KHz] was selected.

The inductance was chosen from the parts available in the online stores Farnell and ELFA so that it accomplishes the required parameters; the chosen product was the 100 [µH] power choke 7447709101 from Würth Elektronik, (Würth Elektronik 7447709101 choke SMD datasheet 2004) which saturation current is 3.1 [A] and the rated DC current 2.5 [A]. For this inductance value, i_{Lmax} is 2.42 [A], which is smaller than the coil saturation current, while i_{Lrms} is 2.01 [A] that is also smaller than its rated DC current.

Additionally, to choose the capacitor, a simulation was performed in NI Multisim to take into consideration the effect of the capacitor's ESR (Equivalent Series Resistor) over the ripple in V_O; the schematic used for this simulation is shown in Figure VI.3.

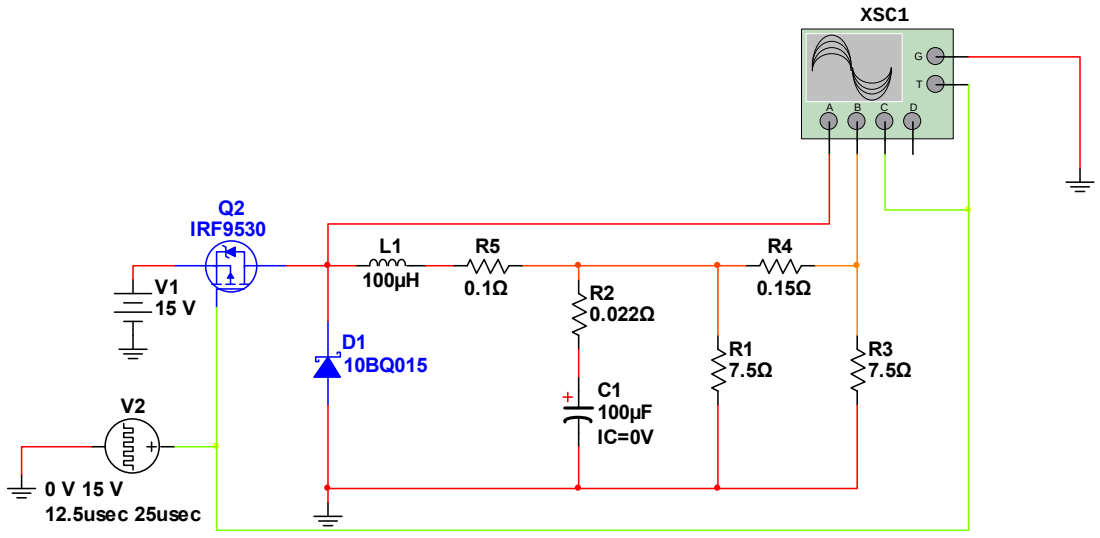


Figure VI.3 Schematic to simulate the effect of capacitors ESR

Different 100 [µF] capacitors were simulated, obtaining the results shown in Table VI.1. The capacitor 1580592 from Farnell was chosen given its performance characteristics and price.

ESR [Ω]	ΔVo [mV]	Price 1 [:-]	Part Nr	Store	Tolerance [%]
Teoretical	11	NA	NA	NA	NA
0,022	15	26,59	1580592	Farnell	20
0,040	20	34,72	1658472	Farnell	10
0,080	38	16,11	1539512	Farnell	20
0,100	45	11,05	1754093	Farnell	10
0,340	140	2,52	67-133-33	ELFA	20

Table VI.1 Simulated ΔV_o for different commercial capacitors

The schematic for the full board is shown in Figure VI.4. To implement the switch, a p-channel MOSFET IRFR-9024NPBF and a MOSFET gate driver LM5112 were used with a 3.3 [Ω] gate series resistance (R1) to limit the gate current. Moreover, for the second supply, the TSR1-2433 was chosen to output 3.3 [V] and up to 1 [A] according to its datasheet (2009).

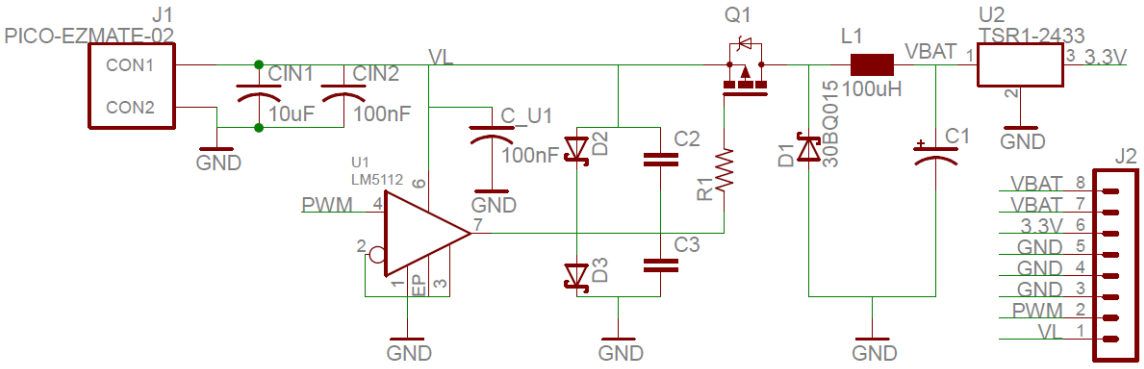
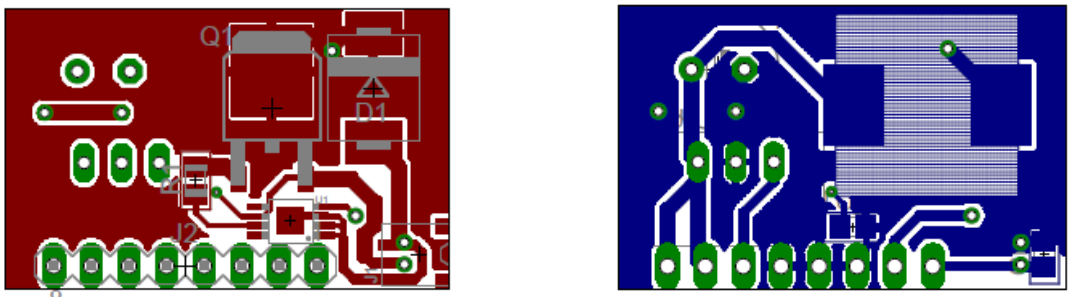


Figure VI.4 Main Power Board Schematic

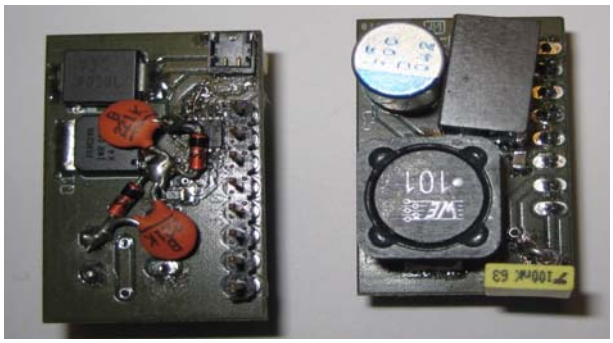
In order to implement this circuit, several PCBs were manufactured at the IEA laboratories to test the different designs and then obtaining a proper performance. The space requirements in the car were also an important aspect to be taken in consideration, so a definitive design was done to accomplish this. This final design, shown in Figure VI.5, was sent to a factory to obtain better qualities PCBs.

Some of the components in Figure VI.4 had to be added after the final PCBs were manufactured; 220 [pF] capacitors C2 and C3 and schottky diodes D2 and D3 were added to protect the LM5112 output from voltage spikes; also, the input decoupling was improved by adding CIN2. Figure VI.6 shows the boards after soldering all the components, including those mentioned above, which were not included in the PCBs design.



a. Top Layer b. Bottom Layer

Figure VI.5 Main Power Board PCB



Top Bottom

Figure VI.6 Final Main Power Board Implementation

Secondary Power Board

The first stage in this board is connected to the Basic Power Board and its function is to measure incoming currents and voltages to use them as feedback for controlling the buck converter and the battery charging process. The schematic for this measuring stage is shown in Figure VI.7; there, connector SV1 goes to the Basic Power Board and J1 is plugged to the battery.

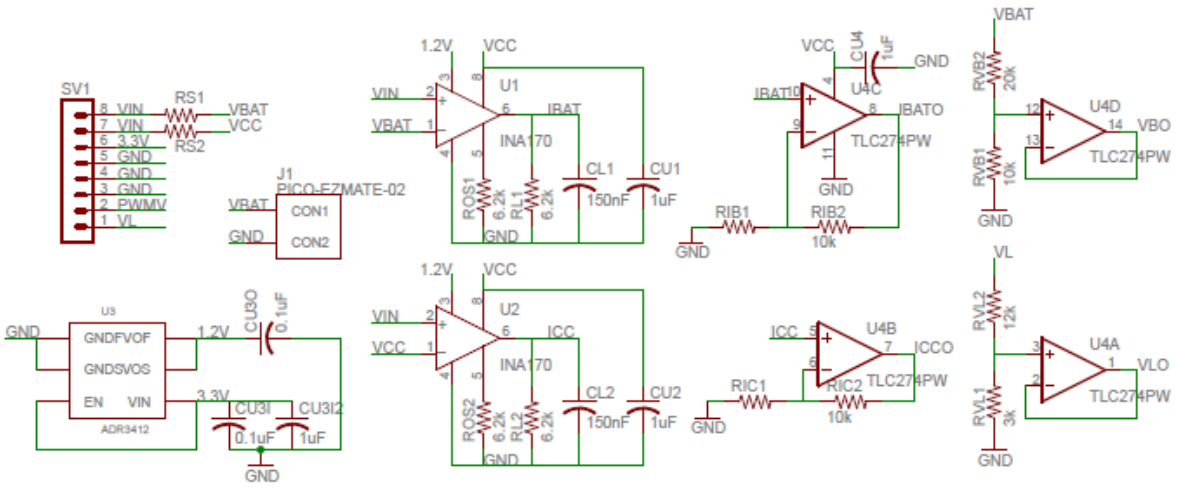


Figure VI.7 Secondary Power Board Measuring Stage Schematic

To achieve current measurement the IC INA170 was implemented, this converts a differential input voltage over a shunt resistor into a current output (INA170 datasheet 2006);

its output is then converted to a voltage using a load resistor. The output voltage is related to the current over the shunt resistor according to following equations.

$$V_O = V_{OS} + \frac{I_S . R_S . R_L}{1[k\Omega]} \quad \text{(VI.8)}$$

$$V_{OS} = \frac{V_{REF} . R_L}{R_{OS}} \quad \text{(VI.9)}$$

In the equations showed above, I_S is the current over the shunt resistor R_S; R_L is the load resistor and R_{OS} can be used to obtain an offset voltage V_{OS} different than the reference V_{REF}. For this application, the maximum current to be measured is 1.2 [A], as it will be explained below, and thus a shunt resistor of 0.15 [Ω] was selected for a maximum voltage drop of ±180 [mV], which is inside the recommended values according to the INA170 datasheet (2006).

To generate V_{REF}, the IC ADR3412 was used, which outputs are 1.2 [V] ± 0.1% (ADR3412 datasheet 2010). Furthermore, R_L was designed to be 6.65 [kΩ] for generating a current that corresponds to an output voltage gain of 0.9975 [V/A]. However, since this value was not available at the moment, 6.2 [kΩ] was used. For these reasons, the equation VI.8 can be expressed as follows, for an output range between 0.084 [V] and 2.316 [V].

$$V_O = 1.2[V] + (I_S \cdot 0.93[V]) \quad \text{(VI.10)}$$

Moreover, the INA170 allows limiting the output's bandwidth by incorporating a load capacitor C_L; this act as a first order, low pass filter with a cut-off frequency according to equation bellow, as expressed in the device datasheet (2006). Finally, 150 [nF] capacitors were used, for a bandwidth of 171.13 [Hz].

$$f_{-3dB} = \frac{1}{2 . \pi . R_L . C_L} \quad \text{(VI.11)}$$

For a proper operation of the mentioned current sensor, there must not be any other component connected to its output that sinks or sources current; according to this information, it was decided to implement an amplifier with almost infinite input impedance. To achieve the mentioned condition, an OPAMP was used in the non-inverter amplifier configuration (Sadiku 2006, p183). This is conformed by the OPAMPs U4C and U4B from Figure VI.7, one for each current channel. Since the amplifier gain was decided to be 1, R_{IB1} and R_{IC1} were not soldered in the final PCB and thus the non-inverter amplifier was

simplified to a buffer; however if a different gain is chosen in the future, it can be adjusted easily by just adding these resistors with specific values.

In order to measure the battery and track voltages, voltage divisors were applied (Sadiku 2006, p43) to decrease the values until valid ADC's range numbers, follow by a buffer configuration to avoid load-effect disturbances on the measurement. The resultant voltages can be calculated with equations below for battery voltage V_{BAT} and track voltage V_L.

$$V_{BO} = \frac{V_{BAT}}{3} \tag{VI.12}$$

$$V_{LO} = \frac{V_L}{5} \tag{VI.13}$$

The next stage in this board consists of the hardware to drive the motors. The two channel full H-bridge L293DD was selected for this purpose due to its small size and good performance (ST L293DD datasheet 1996). The schematic for this stage is shown in Figure VI.8; J2 and J3 are used to connect to the left and right motors respectively. Additionally, SV2 connector goes directly to the Main Board with all the measured signals and supply lines.

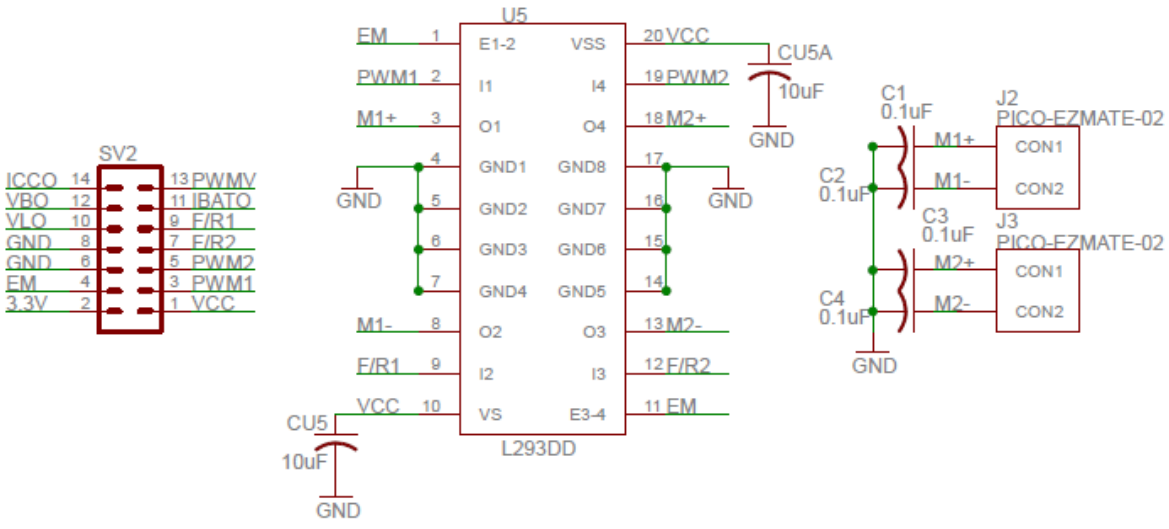
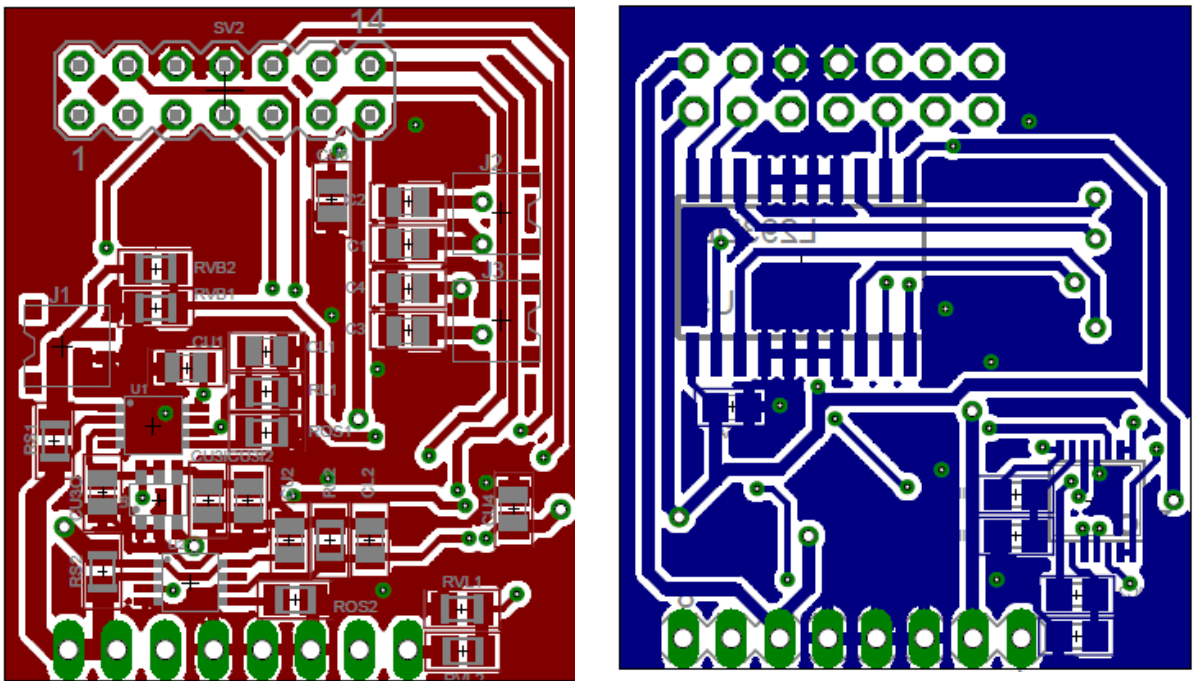


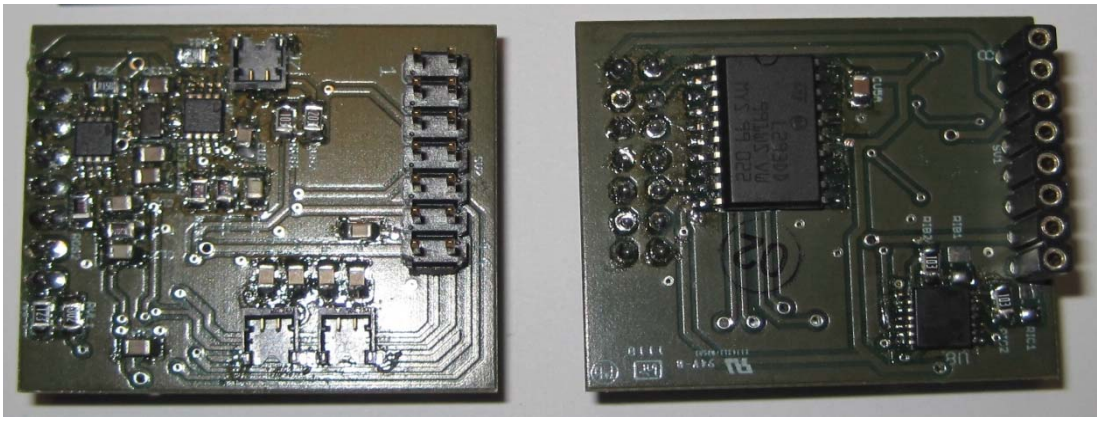
Figure VI.8 Secondary Power Board Driver Stage Schematic

The duty cycles of the lines PWM1 and PWM2 are used to control the motors average voltage and thus the speed; F/R1 and F/R2 controls their spinning direction and EM enables the output to the motors. The final PCB design that was sent to manufacture is shown in Figure VI.9 and the finished board is shown in Figure VI.10.



a. Top Layer b. Bottom Layer

Figure VI.9 Secondary Power Board PCB



Top Bottom

Figure VI.10 Final Secondary Power Board Implementation

Coils Board

The vehicles require a guide system that leads them to follow a desired trajectory when they are not going over slot-track sections. The development of this mechanism was based on Faraday's law of induction and Ampère's law, which state that a changing magnetic field creates an electric field and that a magnetic field is always generated around an electric current respectively (Resnick, Halliday and Krane, 2002).

The system consists in a variable current-carrying wire that produces a determined magnetic field. Furthermore, due to the fact that the current in the wire is variable, an electric field is also induced. According to this phenomenon, the implementation of a coil filter

allows to measure the electric field in form of voltage induced over the coil. It is easier to visualize the structure of this mechanism in Figure VI.11.

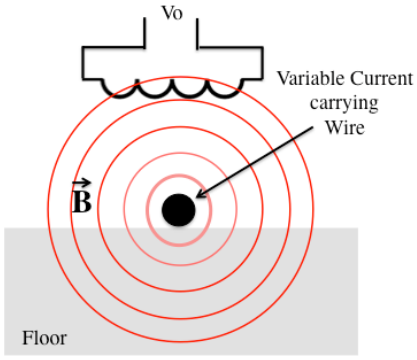


Figure VI.11 Coils and Wire Diagram

Additionally, the voltage on the coil (Vo) can be expressed by the equation below (Laboratorio C USB 2010)

$$Vo = 2\pi fQNAB\cos(\alpha) \qquad \text{(VI.14)}$$

Where f is frequency of the wire's signal; N is the number of turns of the coil; A is the area of the inner circle of the coil and B is the intensity of the magnetic field, which depends on the distance from the wire (Resnick, Halliday and Krane 2002). For that reason, it is possible to set a relation between the distance of the coil from the wire and the measured voltage on the coils.

The current signal in the wire is a sinusoidal wave of variable frequency between 140 and 160 [KHz] (see Wire Guide System p33). The voltage induced over the coils is also a sinusoidal wave of the same frequency and its amplitude has an invert relationship with the distance to the wire. In order to condition this signal, four stages are used as shown in Figure VI.12.

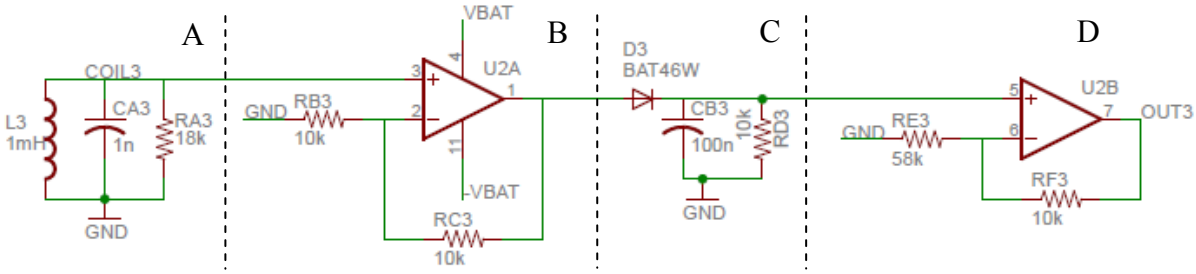


Figure VI.12 Coils Board Schematic for one Channel

The stage A is a RLC resonant circuit, which is excited by the guiding wire. Its frequency response is band pass described by the following equations (Sedra and Smith 2006, p1107).

$$f_O = \frac{1}{2.\pi.\sqrt{L.C}} \tag{VI.15}$$

$$Q = R.\sqrt{\frac{C}{L}} \tag{VI.16}$$

In these equations, f_o is the center resonance frequency and the Q factor is inversely proportional to the bandwidth. Taking the nominal values of the components, f_O is 159.15 [KHz] and Q is 18. However, the used inductors have a tolerance of ±10% and the resonant frequency can change; for this reason, the Q was selected to allow for changes in the wire frequency with respect to f_O without altering the output.

The stage B is a non-inverter amplifier with gain 2, the stage C is an envelope detector with time constant 6.28ms and stage D was used to add another non-inverter amplifier with gain 1.17. All this conditioner stages were necessary to increase as much as possible the analog values of coils signals until the maximum voltage that the MCUs ADC can handle (Vref of 3.0V)

The OPAMP TLV2374 was used for implementing the amplifiers and it was powered with the battery voltage for the positive supply. For the negative one, an ICL7660 was used to provide $-V_{BAT}$. The schematic for the power implementation and the connections from this board to the main board is shown in Figure VI.13.

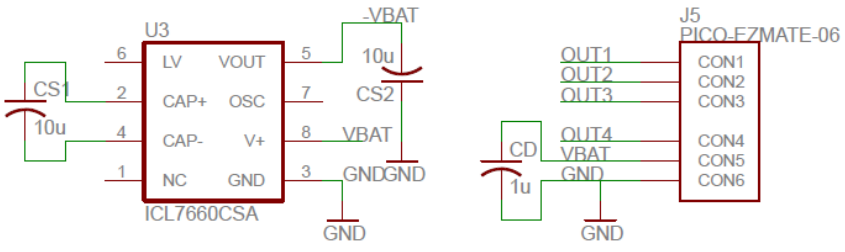


Figure VI.13 Coils Board Schematic for Power and Connections

Moreover, in order to verify the proper behaviour of the design, a simulation was performed in NI Multisim. For this purpose, the stage A was replaced by a sinusoidal voltage source. Also, a low pass filter was added at the output of stage D; this is actually

implemented in the vehicles Main Board. Figure VI.14 shows the schematic used in the simulation and the results can be observed in Figure VI.15 and Figure VI.16. The red signal corresponds to the output of the 160 [KHz] sinusoidal source; the green one is the output of the envelope detector (stage C); and the blue signal is the final output, after the low pass filter.

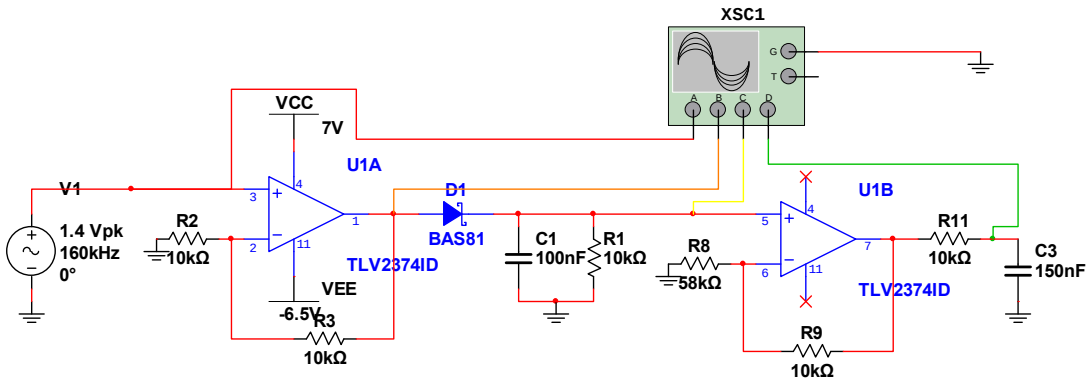


Figure VI.14 Coils Simulation Schematic

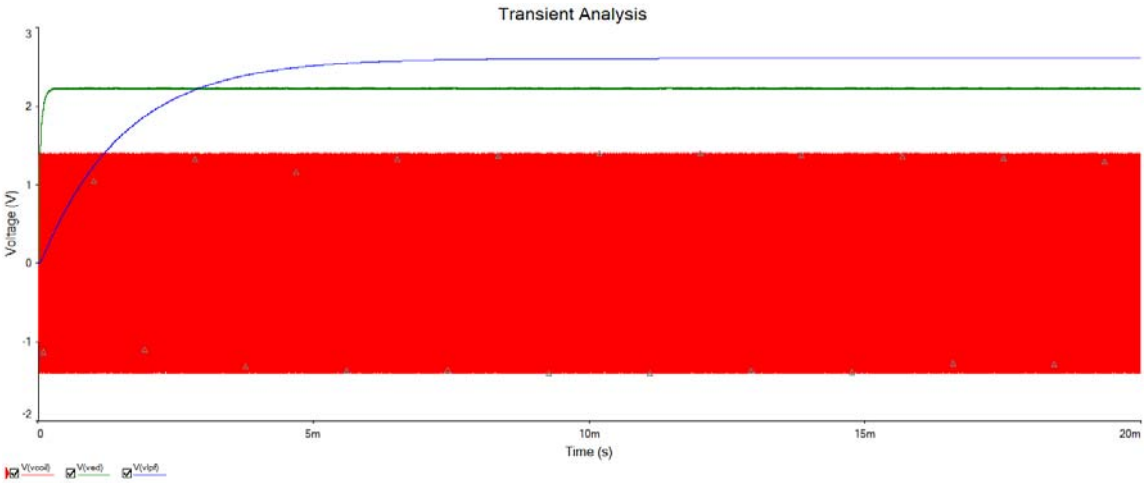


Figure VI.15 Coils Transient Analysis from 0 to 20[ms]

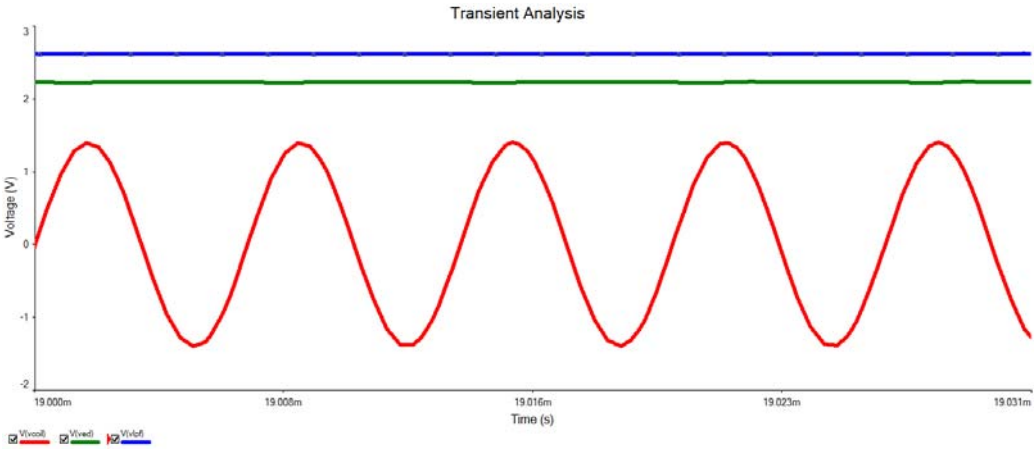
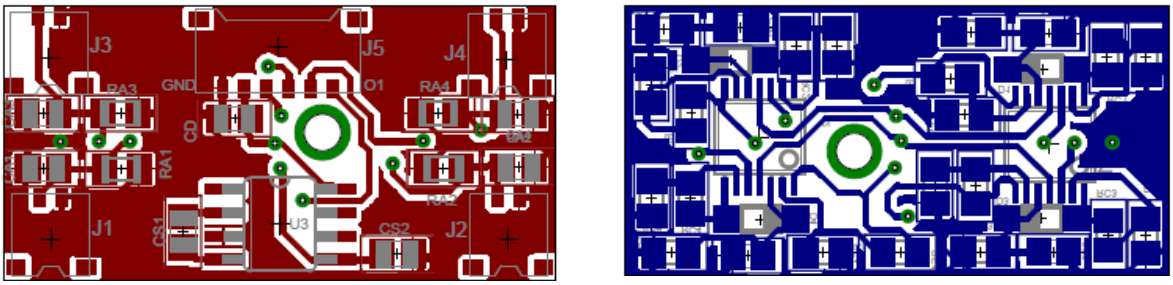


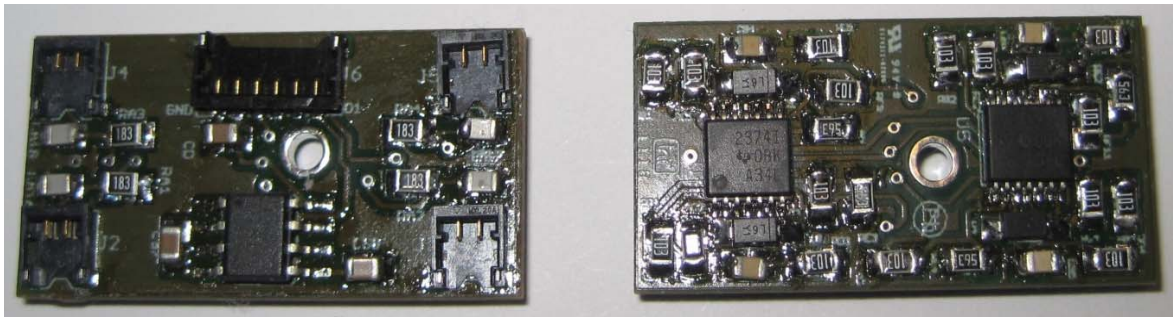
Figure VI.16 Coils Transient Analysis in steady state

For the final PCB, the size was an important constraint since the board had to measure 30 [mm] x 15 [mm]; also the coils were not placed in the board, instead two-pin connectors were used for each one of them. The PCB design is shown in Figure VI.17 and the definitive board is shown in Figure VI.18.



a. Top Layer b. Bottom Layer

Figure VI.17 Coils Board PCB



Top Bottom

Figure VI.18 Final Coils Board Implementation

Measurement Speed Board

The demonstration feature of this project demanded a movement of the vehicles running as natural as possible. For accomplishing this characteristic, it was necessary to control the speed of each unit; and in order to do so, a speed measurement system was required for closing the feedback controller loop.

It was decided to implement the speed measurement only on the vehicles back wheels due to the lack of space on the front part of the units.

The system consists in a set of an Infrared emitting diode type OP245PS and a NPN phototransistor type OP750A with matched wavelength (λ) peaks of spectral responses, at 850 nm, that are optically coupled. However, the fitted light between the set is interrupted nearly periodically according to the speed of the wheel.

The light interruption is triggered by a plastic foil circle of 13 mm diameter attached to the motor axis that is filled with a determined pattern in order to achieve a constant behaviour in the appliance. The Figure VI.19 shows a scheme of the speed measurement mechanism.

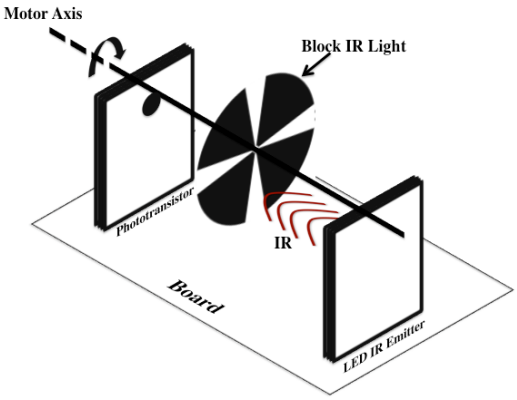


Figure VI.19 Speed Measurement Mechanism.

Circle Design:

The design of the light interrupter circle was done according to physical and functionality reasons of the Speed measurement system. First, the diameter of 13 [mm] was the maximum possible due to lack of space in the vehicle; it was important to maximize it, in order to increase the reliability of the interrupter mechanism. Second, the filling pattern on the circle was set after several experiments with different models. There were tested models with 8, 7, 6 and 4 stripes. Nevertheless, the definitive model of four (4) clear stripes with an angle of 20° was lest affected by the noise, but the lowest resolution.

Circuit Design:

The whole electronic hardware of the speed measurement system is integrated into the same electronic board. According to space requirements inside the vehicles, it was established a specific size for the board and precise physical distribution of the electronic components in order to an adequate performance of the interrupter light mechanism. Then, the IR LED needs to emit light constantly while the NPN phototransistor turns on/off depending on the light incidence over the device. For accomplishing these conditions, it was designed the circuit shown in the Figure VI.20. In addition, the operation points of the circuit are presented in Table VI.2.

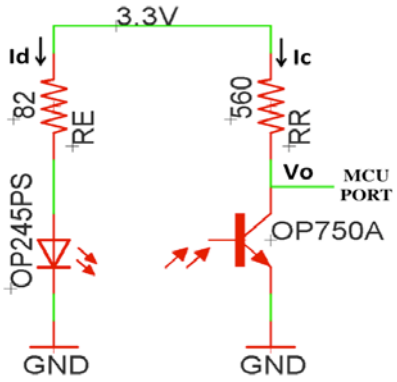


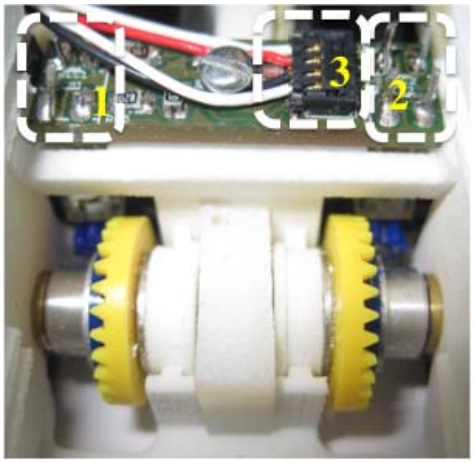
Figure VI.20 Speed Measurement Circuit

IR Light	Id [mA]	Ic [mA]	Vo [V]
Blocked	22	0	3.3
Cleared	22	5	0.2

Table VI.2 Speed Measurement Circuit Operation Points

The IR LED has a forward current of 20 mA –typical test conditions of the datasheet (IR12-21C/TR8 datasheet, rev 1.1, p3)-, which allows the device to radiance highly enough intensity for activating the phototransistor. In addition, the NPN phototransistor has an output voltage that is related directly to the interruption of IR light between the emitter and the receiver.

The speed measurement board includes two pairs of Emitter – Receiver, one per each motor. The Figure VI.21 represents the definitive board and its location inside the cars.



1. - Set LED Emitter & NPN Rec.; 2. – Set LED Emitter & NPN Rec.; 3. – Connector: 3.3V, GND,

Outputs of each set.

Figure VI.21 Speed Measurement Mechanism

On the other hand, it is important to determine the relation between the interruption frequency of the trigger disc and the vehicle speed; this is dependent on the gear ration, the wheel diameter and the disc pattern.

The relation between the motor frequency and the wheel frequency is determined by the gear ratio, as shown in Figure VI.22. This is described by equation bellow, where np and ng are the teeth numbers of the pinion and the contrate gear respectively.

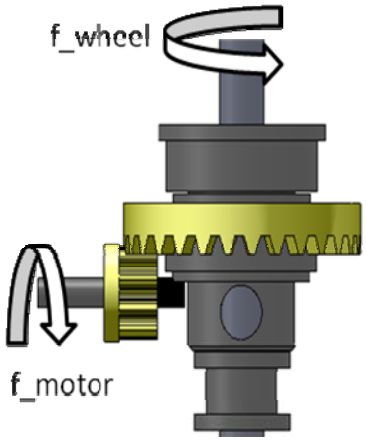


Figure VI.22 Vehicles Transmission

$$f_{wheel} = \frac{np}{ng} \cdot f_{motor} \tag{VI.17}$$

Moreover, assuming that the wheels never slip with respect to the road, the vehicles speed (V) is the same as their tangential speed, which is expressed in equations bellow (Resnick, Halliday and Krane, 2002). In this equation, r represents the wheel radius.

$$V = 2 \cdot \pi \cdot r \cdot f_{wheel} \tag{VI.18}$$

$$V = 2 \cdot \pi \cdot r \cdot \frac{np}{ng} \cdot fmotor \tag{VI.19}$$

Finally, the interrupt frequency is the number of stripes in the trigger disc multiplied by the motor frequency.

$$f_{int} = fmotor \cdot ns \tag{VI.20}$$

Main Board

The whole vehicles electronic system has its operation core in the Main Board; this important circuit has the necessary hardware for a proper performance of the UC3B0256

MCU, for controlling and running the different processes that occurs in the vehicle units. The rest of the electronic boards (Main and Secondary Power Boards, Coils Board and Speed Measurement Board) are all connected to the Main Board through specific connectors. Moreover, additional hardware components are connected directly to this board: the Bluetooth module, the Servomotor, the IR Receiver and the RGB Led. The functions of all this components will be explained with details in this chapter.

Furthermore, all the analog signals coming from the different boards of the vehicle are conditioned and processed in developed filters that were set in the Main Board for an improved data acquisition of the physical measurements of vehicle.

IR Receiver:

The IR Receiver is in charge of detecting and processing the infrared signals emitted by the different track sections IR LEDs. It represents the cars hardware of the Local Position System that is in charge of indicating the current location of each vehicle. The circuit is based on a Vishay Semiconductors IR Receiver Module TSOP58038, a high gain device for continuous or burst IR signals with carrier frequency of 38 [KHz], which is the frequency of the IR signals emitted in all track sections. However, it was mandatory to develop a determined RC filter presented in the unit's datasheet (TSOP58038 datasheet, 2011) for accomplishing an appropriate performance of the device operation. The Figure VI.23 shows the schematic of the application circuit.

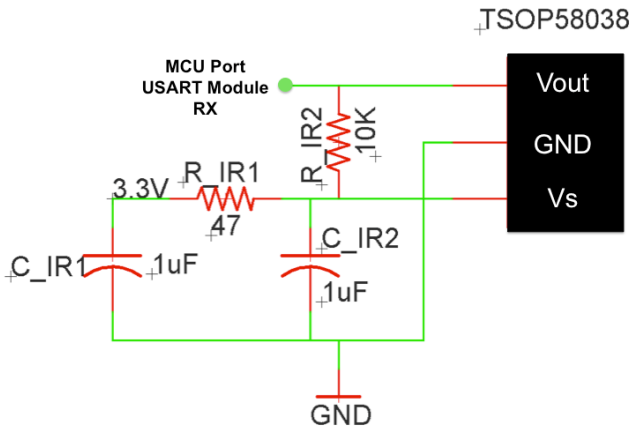


Figure VI.23 IR Receiver Schematic Circuit.

The operation of the IR Receiver is described in the Figure VI.24, it represents the output signal generated for the TSOP58038 when it detects a given IR signals emitted from an external source, from the track IR LEDs in this project.

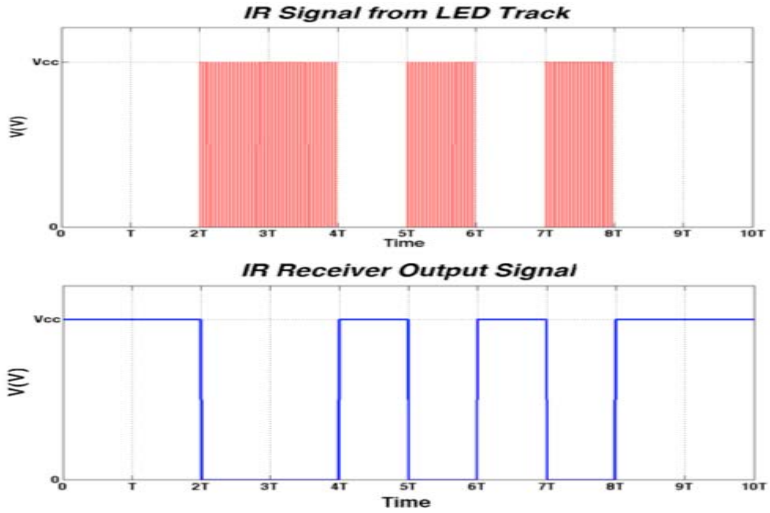


Figure VI.24 IR Receiver Output Signal from the given detected IR signal

According to the Figure VI.24, it is possible to see the inversion of polarity in the output signal versus the IR input signal and the compatibility of the voltage levels with the TTL levels. Additionally, when the IR receiver does not detect any signal, it sets a constant output signal of high voltage level. For that reason, it was possible to set a direct connection between the IR receiver output and one of the USART receiver modules in the main board MCU. The only requirement was setting output signals on the IR receiver that follow the demarked character framing of USART communication: start bit, data bits and stop bit (Traylor 2009). This fact was an important key for developing a simple scheme for decoding the information of Local Position System without the necessity of complicated software.

Bluetooth Module:

A wireless communication between the vehicles and the controller PC was a high priority task in the development of these units. A low power consumption and reliable system was demanded, which would allow communicating the cars and the PC with a highly enough data transfer rate. The Roving Networks Class 1 Bluetooth® module RN-41 was a simple alternative for developing this communication bridge.

This module has several features and tools for implementation. Though, the lack of space in the main board forced to take the minimal required hardware resources for building the communication bridge Bluetooth – PC. The definitive schematic circuit is shown in the Figure VI.25 with the pin out distribution of the Bluetooth according to the information presented in the devices datasheet (RN-41 datasheet 2009).

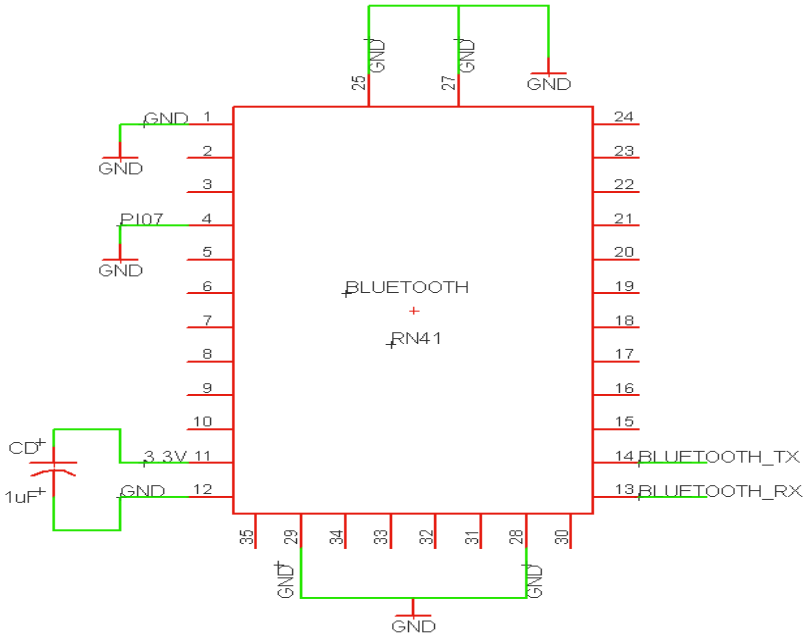


Figure VI.25 Bluetooth Schematic Circuit.

The only implemented pins are the supply pins (Vcc and grounds), the USART Transmitter Pin, the USART Receiver Pin, and the PI07 that sets the baud rate of the transfer data (HIGH: 9600 baud and LOW: 115200 baud) (RN-41 datasheet 2009, p3). CTS or RTS were not implemented since they are not established in the communication protocol.

Servomotor:

All the vehicles needed a mechanism for getting in/out of the guiding slot, because it was demanded to run over the two types of track models: slot track and non-slot track. So, a mechanical-electronic subsystem was designed that would allow to develop this mechanism.

The implementation of this subsystem is based on a servo HS-55, which has an operating voltage between 4.8 [V] 6.0 [V], an operating speed of 0.10 [s/degree] and an output torque of 1.4 [kg/cm] (HS-55 Announced Specifications, 2011). In addition, the most important feature of this device is the simplicity of its requirements for controlling it: only a PWM channel. Finally, the definitive schematic circuit for setting this servo unit is presented in the Figure VI.26.

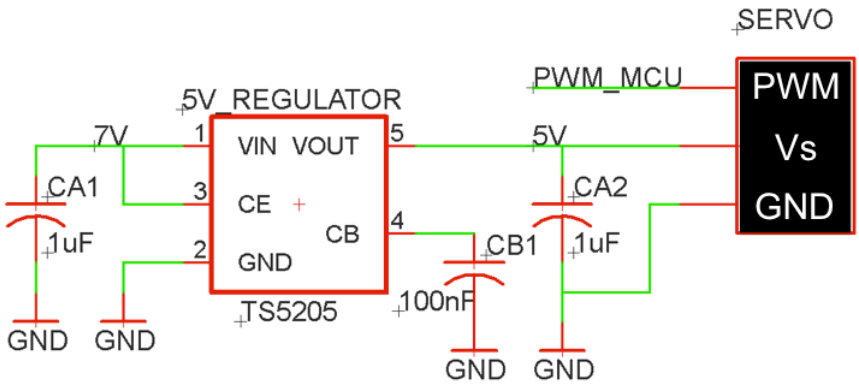


Figure VI.26 Servo Schematic Circuit.

The main board is supplied by two different power sources, one at 3.3 [V] and the other at Vbat (battery voltage), both of them come from the main connector #4 (see Filters and Connectors section below). However, the servo unit requires a supply voltage between 4.8 [V] and 6.0 [V]; therefore, a 5 [V] voltage regulator had to be used. The applied model was a SPX5205M5-L-5-0, with a maximum output current of 150 [mA] and a minimum dropout voltage of 250 [mV] (SPX5205 datasheet 2008). The application circuit for this regulator is show above in Figure VI.26.

As mentioned before, the only requirement for controlling the servomotor was a PWM channel. However, the PWM signal needs to fulfil some properties about the applied duty cycle and frequency. The implemented operating frequency was 50 Hz and the rank of suitable duty cycles is from 5% to 15%, i.e. pulse widths from 1 [ms] to 3 [ms] respectively. It is important to mention that the duty cycle rank has been set according to the rank of suitable angles for the getting on/off mechanism. The mentioned ranks can be represented in Figure VI.27.

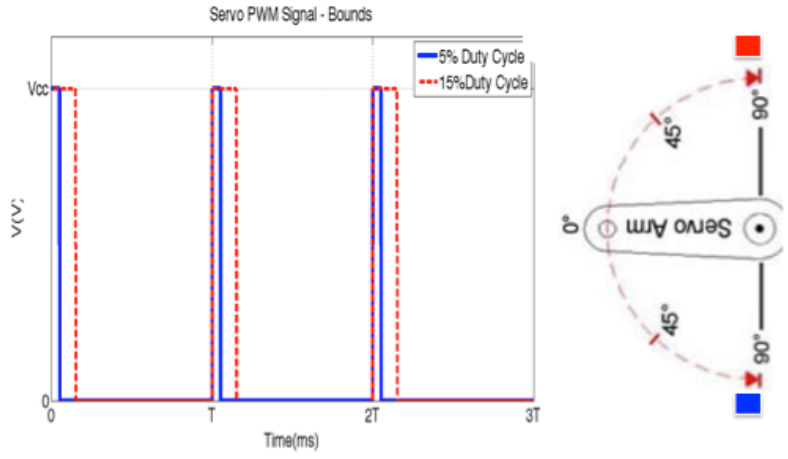


Figure VI.27 Duty cycle rank in the PWM vs. Angle on the Servo Arm.

RG Led:

The battery state of charge (SoC) was a remarkable data that spectators of this demonstration project needed to visualize without the necessity of interpreting a number. Thus, a RG LED was implemented to display the status of the battery in a simple way related to colours.

Nowadays, most of the electronic products that state their batteries SoC associate green colour with full charge, while red is associated with a battery almost discharged. According to this principle, it was decided to follow the same pattern for showing the battery SoC in our vehicles. The vehicles chassis has a particular aperture that allows the pass of light beams that come from the RG LED to the exterior.

The RG LED is controlled by two PWM channels, which frequencies and duty cycles are set by software. The model of the RG LED is a Kingbright 2.0 [mm] x 1.25 [mm] SMD CHIP LED and its operation was designed according to the Figure VI.28, based on the configuration with forward current of 20 mA for both dices. The differences in typical forward voltages – Green: 2.1V and Hyper Red: 1.95V- caused a design with different resistance values (Kingbright RG LED 2007).

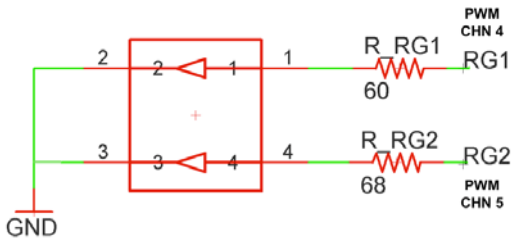
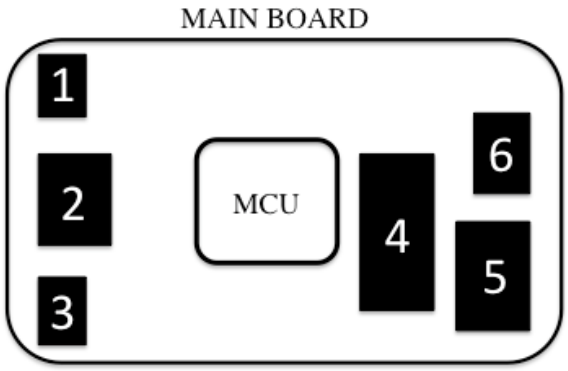


Figure VI.28 RG LED Schematic Circuit.

Connectors and Filters:

The Main board has a total of six connectors, with different models that vary on the number of pins: from 3 to 14 pin connectors. One of the most applied models is the Molex Pico EZ-mate; their small size was the main reason why this model was implemented, in despite of the fact that those were very problematic for manipulation and installation.

Each connector allows creating a specific connection between the Main board and the rest of the vehicle's boards. The Figure VI.29 represents a simple scheme of the distribution and rolls of the connectors in the main board.



# Connector	Hardware Associated	# Pins
1	IR Receiver	3
2	Coils Board	6
3	Servomotor	3
4	Secondary Power Board	14
5	JTAG	10
6	Speed Meas. Board	4

Figure VI.29 Connectors Distribution in the Main Board.

The connectors # 1 and 3 are connected directly to the corresponded electronic devices of each subsystem. The connector 5 (JTAG Interface) corresponds to the programming connector of the MCU. While the rest of the connectors, 2, 4 and 6 handle signals processed previously and supply lines from additional vehicle boards.

Some of the signals coming from connectors 2 and 4 demand an additional filtering to increase the noise rejection and a better performance in the data acquisition of these physical variables. The Table III.1 summarizes the list of those mentioned signals, which corresponds to processed analog signals.

Signal	Connector	Abbreviator
Coils Voltage 1	Coils Board	CO1
Coils Voltage 2	Coils Board	CO2
Coils Voltage 3	Coils Board	CO3 (Auxiliary)
Battery Current	Power Secondary Board	IB
Battery Voltage	Power Secondary Board	VB
Load Current	Power Secondary Board	ICC
Track Voltage	Power Secondary Board	VL

Table VI.3 Analog Signals in the Main Board.

In order to accomplish the filtering of these signals, two very similar models of passive low pass filters were designed; followed by an op-amp in buffer configuration. The definitive model schematic circuits are shown in Figure VI.30. The filter A was implemented for signals from the Secondary Power Board, while model B filters the Coils signals.

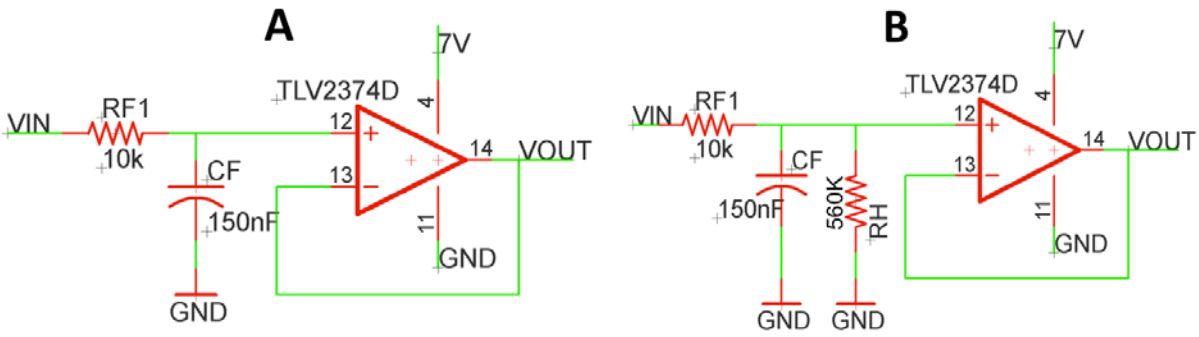


Figure VI.30 Low Pass Filter Models of the Main Board

It was decided to develop passive low pass filters with a buffer configuration due to the small number of components that it involves, taking into consideration the lack of space in the main board. Both circuit models have similar cut-off frequencies (f_c), which are determined by the equation bellow (Sadiku and Alexander, 2006, p638)

$$f_c = \frac{1}{2\pi R_f C_f} \tag{VI.21}$$

In the model B, the resistance RH is 56 times higher than RF1, so the load effect added can be neglected for simplicity. For that reason, it can be considered a cut-off frequency of approximately 106 Hz in both models. In addition, the function of the resistor RH was discharging the capacitor CF when the input pin of the filter disconnected (i.e. open circuit).

UC3B0256 32bits MCU:

All of the subsystems involved in the main board are controlled and monitored by the UC3B0256 32bits MCU. This unit has several features that allowed to develop a high performance system at low power consumption: single 3.3V power supply, capability to run at frequencies up to 60 MHz, seven independent PWM channels, three identical 16 bits Timers/Counters, many communication interfaces, and numerous additional features (UC3B0256 datasheet 2010).

Despite of the fact that the microcontroller has several kind of resources, a large number of different modules were not implemented; only the following modules were used:

- 2 USART Modules.
- 6 independent PWM channels.
- External Clock Input and PLL.
- 7 Channels of Analog-to-Digital Converter.

- 2 External Interrupter Pins.
- 3 I/O Ports.
- JTAG Interface for programming the unit.
- RESET Pin.

These resources were assigned to the different main board processes according to the Table VI.4, which shows the final implemented pin distribution in the microcontroller unit.

Application	PIN	Resource	Abbreviation
JTAG	TCK	-	JTAG_TCK
	PA00	-	JTAG_TDI
	PA01	-	JTAG_TDO
	PA02	-	JTAG_TMS
	PA15	-	EVTO
OSCILLATOR	PA18	XIN0	XIN0
USART	PA24	RX – USART1	BLUETOOTH_TX
	PA23	TX – USART1	BLUETOOTH_RX
	PB10	RX – USART0	IR_RX
PWM	PA11	PWM [CHN 0]	P_MOTOR1
	PA12	PWM [CHN 1]	P_MOTOR2
	PA13	PWM [CHN 2]	P_V (POWER)
	PA22	PWM [CHN 6]	P_SERVO
	PA28	PWM [CHN 4]	RGB1
	PB05	PWM [CHN 5]	RGB2
ADC	PA03	ADC [CHN 0]	ACOIL_1
	PA04	ADC [CHN 1]	ACOIL_2
	PA05	ADC [CHN 2]	ACOIL_3
	PA06	ADC [CHN 3]	A_IB
	PA07	UNIMPLEMENTED	A_IB (DSGN)
	PA08	ADC [CHN 5]	A_VL
	PA30	ADC [CHN 6]	A_VB
	PA31	ADC [CHN 7]	A_IC
EIC	PB02	EXTINT [6]	E_SM1
	PB03	EXINT [7]	E_SM2
I/O	PA16	I/O	ENABLE_MOTOR
	PB08	I/O	Front/Rear2 (F/R2)

	PB09	I/O	Front/Rear1 (F/R1)

Table VI.4 Pin distribution in the Main Board MCU

On the other hand, some of the implemented resources were necessary to improve the performance and the functionalities of the microcontroller unit. The designed of the different applied configurations was based on the MCU datasheet information of the MCU (UC3B0256 datasheet 2010) and additional data related to the rest of devices.

- RESET Pin:

The implementation of the RESET pin circuit was motivated on the EVK1101 schematic circuit (Schematic EVK1101 2007), which is an evaluation kit and development system for Atmel UC3B MCUs. The RESET Pin circuit is shown in the Figure VI.31.

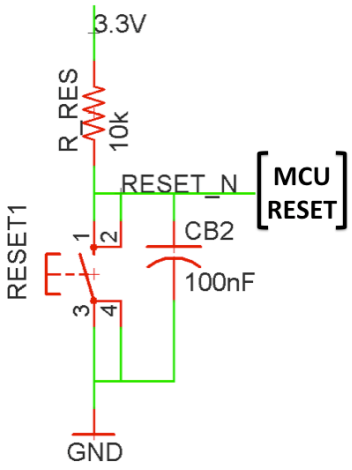


Figure VI.31 RESET Pin Schematic Circuit.

- JTAG Interface:

It was required a programming interface for implementing in the MCU. In order to match with the AVR One! Programmer, it was decided to implement the same interface applied on the EVK1101: JTAG. The Figure VI.32 presents the definitive schematic circuit of the JTAG Interface Connector. It is remarkable to take in consideration that most of the pins are connected to the microcontroller unit, so it is recommended to see the overall schematic circuit of the main board for an easier visualization.

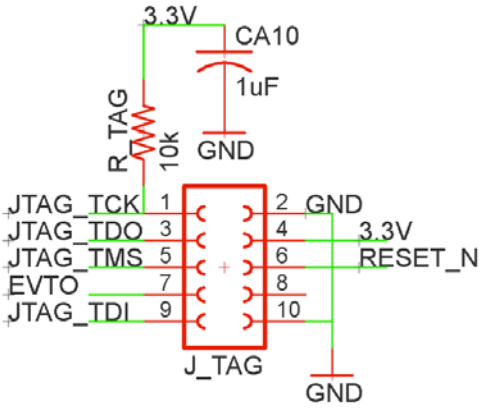


Figure VI.32 JTAG Interface Schematic Circuit.

- External Clock:

The UC3B0256 32bits MCU has only an integrated low power RC oscillator that operates at 115 KHz approximately (UC3B0256 datasheet 2010, p37). However, operating the MCU at a higher frequency would increase the performance. For that reason, it was decided to implement an external clock of 12 MHz that allows the MCU to run at up to 60 MHz through PLL application.

The external clock was generated with a CFPS-9 oscillator; its definitive schematic circuit is presented in the Figure VI.33, which was based on the devices datasheet (2008).

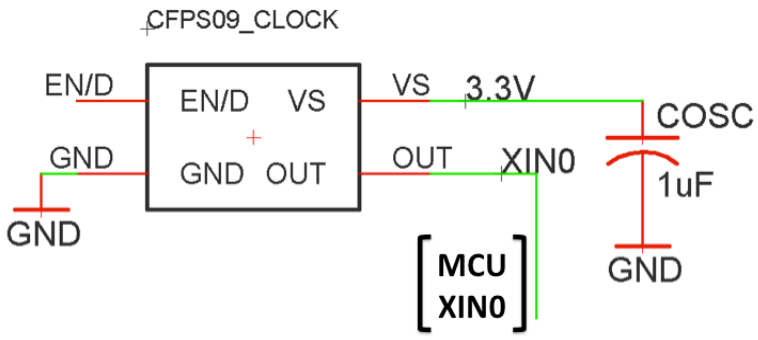


Figure VI.33 External Clock Oscillator Schematic Circuit.

- ADC Reference:

The 3.3V supply line of the main board comes from a DC/DC converter set in the Main Power Board. In consequence, the noise rejection is low in the line, so an ADC voltage reference (ADCVREF) set in the 3.3V does not represent an appropriate design decision. According to this fact, it was decided to establish the ADCVREF at a constant 3VDC, which comes from a Texas Instrument REF3030 voltage reference with low dropout voltage and

minimal input voltage of 3.05V (REF3030 datasheet 2008). Finally, the Figure VI.34 shows the final schematic circuit of the voltage reference.

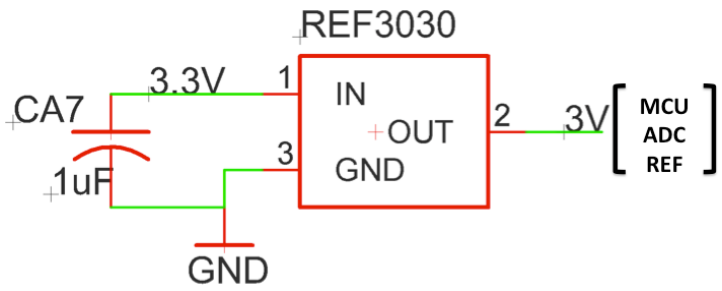


Figure VI.34 ADC Voltage Reference Schematic Circuit.

- Decoupling Capacitors:

The MCU is supplied by the mentioned 3.3V that comes from the Main Power Board, a noisy power source that does not guarantee a stable power supply, a very important requirement for implementing the Atmel UC3B0256.

The evaluation kit EVK1101 implements also several decoupling capacitors close to the different supply pins of the microcontroller unit. The lack of time forced to implement the same design of Atmel in the EVK1101 board. However, there are some differences in some capacitances with respect to the original model due to availability reasons.

The schematic circuit of the final decoupling system implemented in the MCU is shown in the Figure VI.35.

It is possible to see in the Figure VI.35 that the label CS# abbreviates specific capacitors. Those capacitors represent special units that the datasheet information of the MCU (UC3B0256 datasheet 2010, p618) set as required for the UC3B0256 implementation. The Table VI.5 summarizes a list of these devices with its corresponded special technology.

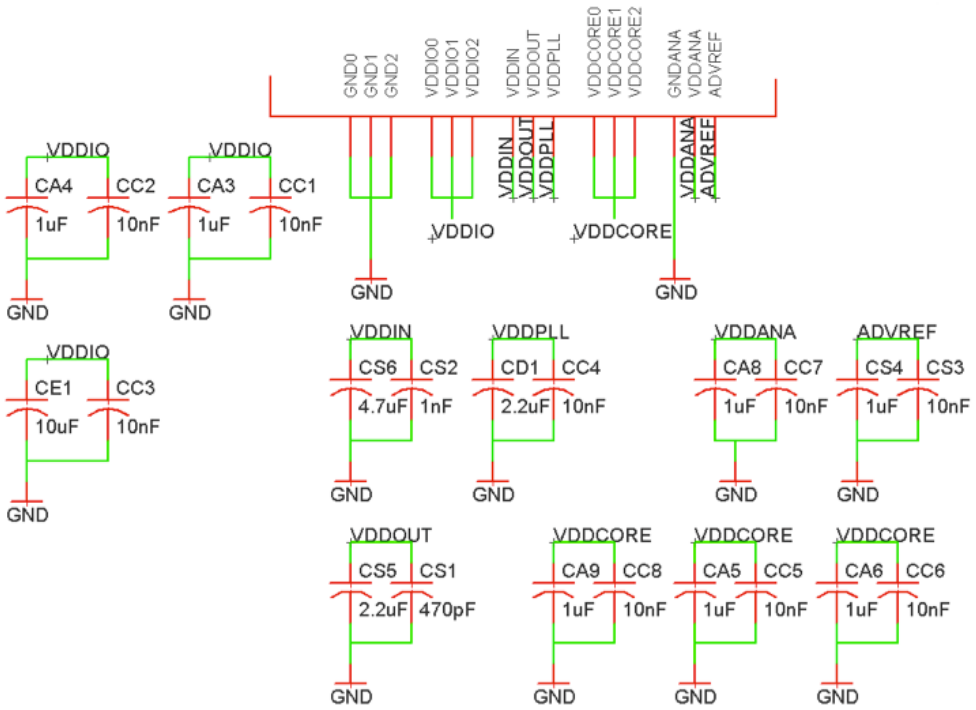


Figure VI.35 MCU Decoupling system.

Capacitor	Technology	Capacitance	Parameter
CS1	NPO	470 pF	Output Regulator Capacitor 1
CS2	NPO	1 nF	Input Regulator Capacitor 2
CS3	NPO	10 nF	Voltage Reference Capacitor 1
CS4	X7R	1 uF	Voltage Reference Capacitor 2
CS5	X7R	2.2 uF	Output Regulator Capacitor 2
CS6	X7R	4.7 uF	Input Regulator Capacitor 2

Table VI.5 Special Decoupling Requirements of the MCU

VII. VEHICLE SOFTWARE

Communication Protocol

The communication from vehicles to the PC is done via Bluetooth, but in both ends it is treated as a standard serial communication, at 115200 bauds/s, using 8 data bits, and none parity bit. There is no need to include address information in the messages, since the PC creates a COM port for each vehicle.

The most important data to be sent from the car is the current position, so that the PC can turn on and off the required sections and control the traffic to avoid crashes. Moreover, the PC needs to be able to control the car speed, and the position of the guiding mechanism (up, down or middle).

Moreover, to avoid bit-operations, all the commands and data are sent using 1 byte, even if it is just a binary instruction.

Message structure:

For all the messages, 11 bytes are sent, including 2 start bytes, a message ID and different data and commands. In total, 4 messages were implemented as shown bellow, but the protocol can be extended to use up to 127 different messages.

0	1	2	3	4	5	6	7	8	9	10
0xFF	0xFF	1	Battery Voltage		Track Voltage		Battery Current		Circuits Current	
Start Bytes		ID	Message							

Table VII.1 Message 1 Format

0	1	2	3	4	5	6	7	8	9	10
0xFF	0xFF	2	Coil 1		Coil 2		Coil 3		Coil 4	
Start Bytes		ID	Message							

Table VII.2 Message 2 Format

0	1	2	3	4	5	6	7	8	9	10
0xFF	0xFF	3	Position		Left W Speed		Right W Speed		Bat SoC	
Start Bytes		ID	Message							

Table VII.3 Message 3 Format

0	1	2	3	4	5	6	7	8	9	10
0xFF	0xFF	4	Left W Speed		Right W Speed		Start/Stop		Guiding Mechanism State	
Start Bytes		ID	Message							

Table VII.4 Message 4 Format

Messages 1, 2 and 3 are used by the vehicles to send their current state to the PC, and message 4 is sent by the PC to command the vehicles.

Data format:

All the quantities in messages 1 and 2 are sent as read from the ADC, so they need to be operated by the PC in order to obtain conventional units (Volts and Amperes). The first operation needed is to convert the digital value to an analog voltage, according to the following equation.

$$V_{ADC} = \frac{N_{ADC}}{2^n} \times V_{REF} \qquad \text{(VII.1)}$$

V_{REF} is the reference voltage, V_{ADC} is the input voltage, N_{ADC} is the converted value and n is resolution. In this case the ADC resolution was configured to 10 bits and 3.0[V] was used as the reference.

For the battery voltage (V_b) and track voltage (V_l), simple voltage divisors were used as explained before (Secondary Power Board p77); equations VI.12 and VI.13 (p79) can be used to convert the ADC voltage to the actual voltage. Also, for the battery current (I_b) and the circuits current (I_{cc}) equation VI.10 (p78) is used to convert from ADC voltage to current value. Since the coils values are only used for calibration purposes, they do not need to be converted.

For the speed, the values that are actually sent correspond to the motor rotating frequency in [Hz] using fixed-point format $Q_{7.8}$, which can be converted to real numbers multiplying by 2^{-8} (Årzén 2009, p207). In order to obtain the vehicle speed, Equation VI.19 (p88) must be used.

The start/stop byte commands the car to run if it is equal 1, otherwise it makes it stop. For the guiding mechanism state, 0 means run straight with the stick down; 1 follow the wire with the stick in the middle, 2 go straight with the stick up, and 3 turn right with the stick up

Controllers Development

Two different controllers had to be developed: one for the Buck Converter in the Main Power Board (Battery Controller) and other for the motors that run the car. The overall process followed to design and implement the controllers was the same for both of them, and thus it is explained as one; however, the results and some other implementation issues are explained in separate sections.

In both cases, it was important to have zero steady state error and hence an Integral Part was needed; but since there was no important constrain on the setting time, a Derivative Part was not necessary (Kuo 1997). Therefore, a PI controller topology was decided to be implemented.

System Identification:

In order to design a controller, it is important to have information regarding the process to be controlled; this can be achieved by developing either a mathematical model or a system identification model based on experimental data. For simplicity reasons, it was decided to record input-output data for both systems – the buck converter and the motors – and then, to use Matlab's System Identification Tool to obtain a process model.

The input-output data was recorded using as input signal multiple steps at random times. Then, this data was imported to Matlab's System Identification Tool as Time domain data, specifying its sampling period and starting time.

The next step was to preprocess the data and there are different options in order to do this, including filtering or merging different experiments. It was decided to use the "quick start" preprocessing, which only eliminates the signals DC content and splits them in two; one half is used to estimate the system model and the other is used to validate it.

Once the data is preprocessed, the System Identification Tool offers many options in order to generate a model. This options are useful when the model structure or some parameters are known; they can be introduced and then, only what remains unknown is estimated.

However, since it was decided not to spend time calculating any information regarding the process, everything was unknown; therefore, the "quick start" estimate was used. It generates different models, and calculates their accuracy; the model with the higher accuracy was chosen for designing the controller.

Controller Design:

After obtaining a process model, the next step is to design a controller. This was achieved using Matlab's SISO design tool; which can be used for different control architectures and the controllers can be either graphically tuned or automatically tuned. Also it offers different analysis plots to see the results of the tuning process.

In order to save time, it was decided to use the "Automated Tuning"; the chosen design method was "PID Tuning", for a PI Controller Type and the used tuning algorithm was "Ziegler-Nichols closed loop", trying its three tuning formulas. Finally, the analysis plots for the three resultant controllers where studied, and the one with the best overall characteristics was used. In the case of the battery controller the important parameter was the

setting time and the disturbance rejection time, and for the motors controllers it was also important that the time response were not oscillatory.

Controllers Programming:

A continuous time PI controller can be programmed in a MCU or PC using the following algorithm (Årzén 2009). It incorporates tracking anti-windup to control the growth of the integral part when the control signal saturates.

```
y = yIn.get();
e = yref - y;
v = K*e + I;
u = sat(v,umax,umin)}
uOut.put(u);
I = I + ie*e + it*(u - v); // ie=(K*h/Ti) and it=(h/Tr)
```

In this example code, K is the continuous time controller gain; Ti and Tr are the integration times for the integral part and the tracking anti-windup respectively, and h is the sampling period.

The drawback of using this algorithm is that it requires the controllers' parameters in continuous time, but they were designed in discrete time; also, since the discrete time difference equations for the controllers are known, they could be directly programmed. However, it was decided to use the mentioned algorithm because it calculates the integral part in the past iteration, and thus it reduces the delay before applying the control signal.

In order to convert the controller to continuous time, Matlab's function "d2c" was used, selecting the "matched" conversion method.

Another important consideration is that controller parameters are real numbers – i.e. not just integers – but the MCU does not have hardware support for floating point arithmetic, which is the method that computers use to handle it. For this reason, it was necessary to implement fixed point arithmetic.

The drawback of using fixed point is that it has a compromise between the resolution and the maximum values that can be stored, and the appropriate format for each number has to be calculated taking in consideration its possible highest value. However, the processing use is much lower than using software emulation for floating point (Årzén 2009).

Battery Controller:

The input output data after preprocessing is shown in Figure VII.1 and different identified systems are shown in Figure VII.2.

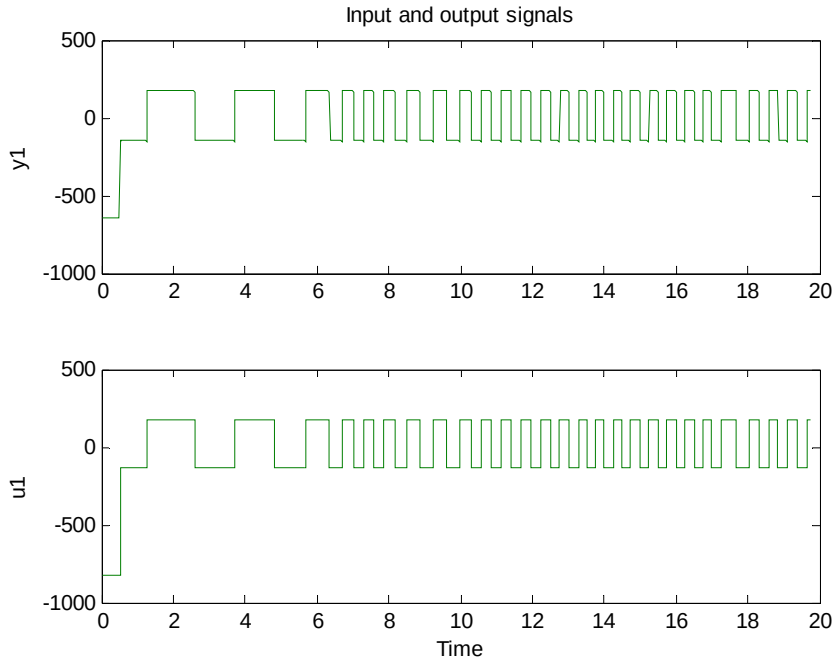


Figure VII.1 Buck Converter Input and Output Data

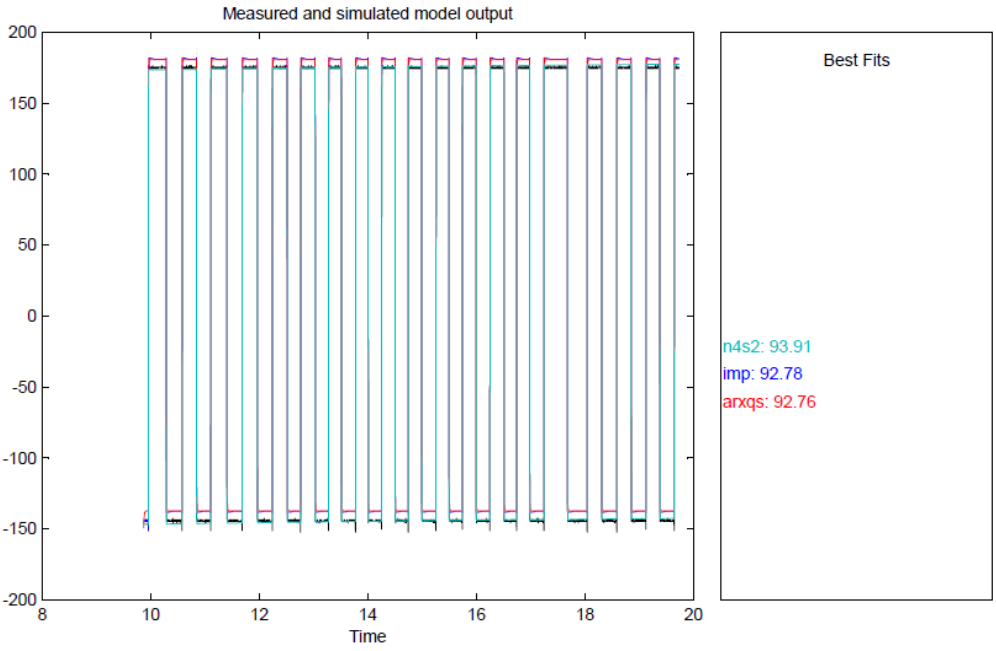


Figure VII.2 Buck Converter Measured and Simulated Model Output

The used model was n4s2:

```
State-space model:  x(t+Ts) = A x(t) + B u(t) + K e(t)
                       y(t) = C x(t) + D u(t) + e(t)
A =
                        x1           x2
           x1       0.10951     0.069958
           x2      0.025413      0.99779
B =
                        u1
           x1    0.00046513
```

```
            x2 -1.3305e-005
C =
                        x1           x2
            y1        2055.1       270.59
D =
                        u1
            y1             0
Estimated using N4SID from data set mydatadv
Loss function 100.315 and FPE 100.721
Sampling interval: 0.0025
```

This model was inputted to the SISO Design Tool, and the following controller was obtained, using Tyreus-Luyben formula. Its step response is shown in Figure VII.3.

$$C(z) = 0.40304 \cdot \frac{z - 0.7967}{z - 1} \rightarrow C(s) = 0.40304 \cdot \frac{s + 81.32}{s}$$

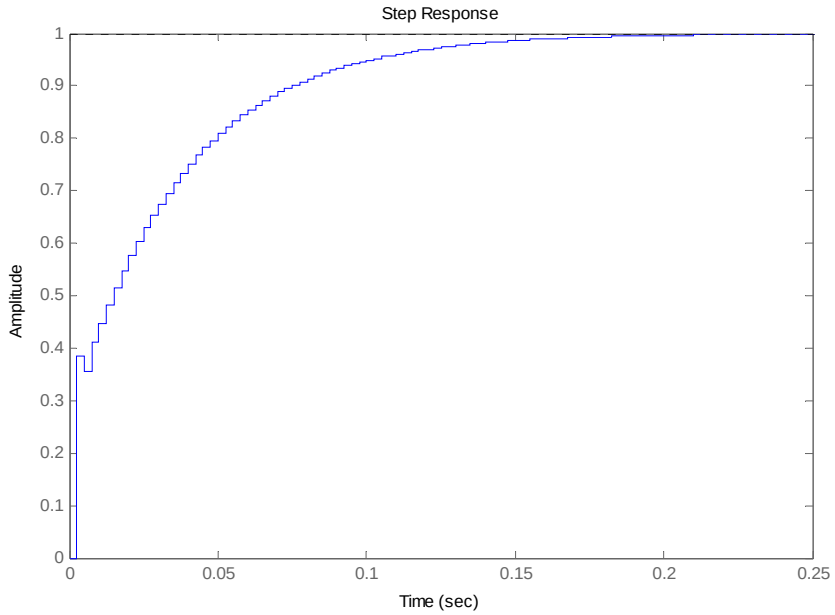


Figure VII.3 Battery Controller Step Response

For this controller K=0.4034, Ti=$(81.32)^{-1}$ and h=0.0025; also, Tr was set equal to Ti. This results in ie=0.08193803 and it=0.2033.

Since all this quantities are smaller than one, it was decided to store them in fixed point format $Q_{1,30}$ which has a resolution r = 9.3132×10^{-10} and can represent real numbers in the range from -2 to (2 - r). The numbers converted to Q_{30} format are: K=432760905, ie= 87980292 and it=218291713.

Given that the controller proved to work, no more data was recorded to check its step response and disturbance rejection; this because saving time had a higher priority in this project than to show controllers results.

Motor Controller:

The input output data after preprocessing is shown in Figure VII.4 and the some identified systems are shown in Figure VII.5; the step responses for these systems are shown in Figure VII.6 where data1 and data correspond to ss2 and ss8 respectively.

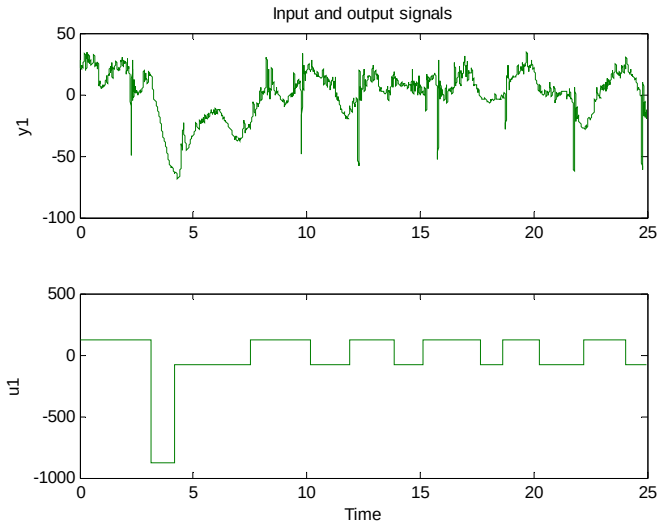


Figure VII.4 Motors Input and Output Data

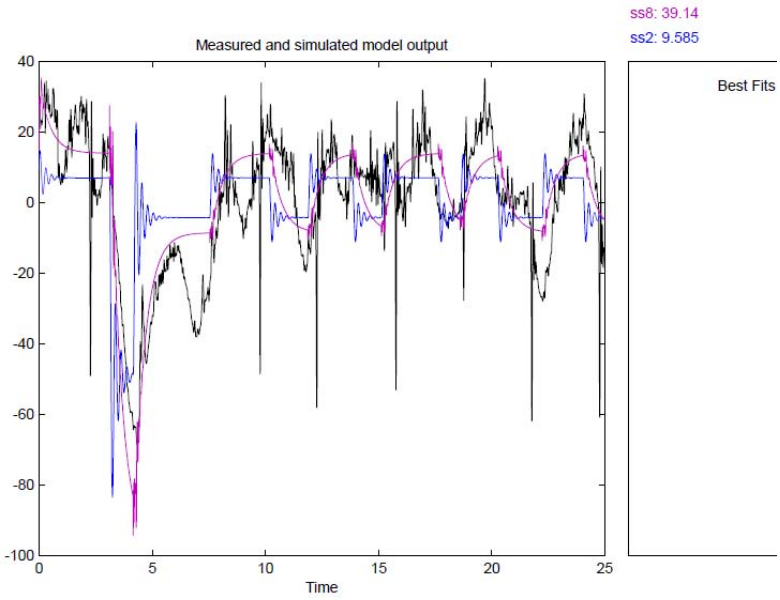


Figure VII.5 Motors Measured and Simulated Model Output

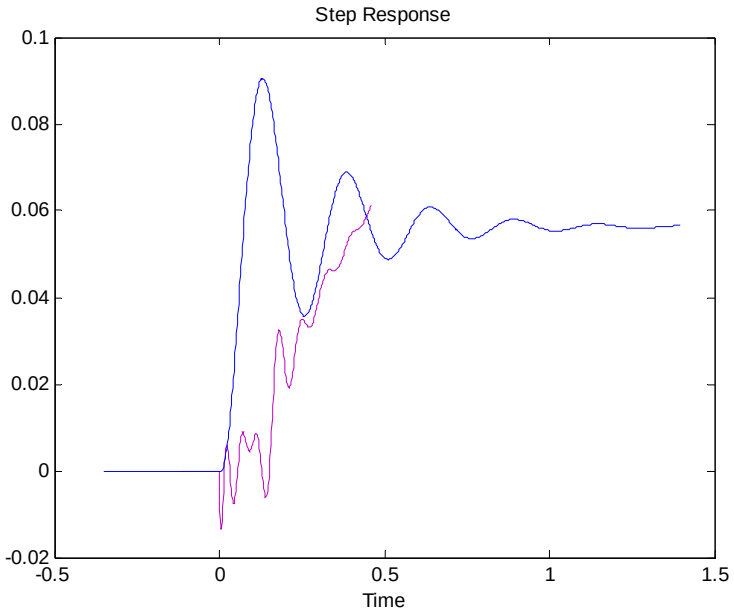


Figure VII.6 Motors Models Step Responses

Even though the model ss8 has the highest accuracy; ss2's step response is more similar to what would be expected from an electric motor, and thus it was decided to use it for designing the controller.

```
State-space model:  x(t+Ts) = A x(t) + B u(t) + K e(t)
                       y(t) = C x(t) + D u(t) + e(t)

A =
                        x1           x2
           x1       0.99803    -0.049786
           x2      0.049293      0.98376
B =
                        u1
           x1 -2.5126e-007
           x2 -2.8808e-006
C =
                        x1           x2
           y1        1003.6      -9.2061
D =
                        u1
           y1             0

Estimated using N4SID from data set mydatadv
Loss function 0.0281729 and FPE 0.028245
Sampling interval: 0.002
```

This model was inputted to the SISO Design Tool, and the following controllers were obtained. Their step responses are shown in Figure VII.7 and Figure VII.8.

$$C1(s) = 12.2401 \cdot \frac{s + 7.983}{s}$$

$$C2(s) = 19.6058 \cdot \frac{s + 2.808}{s}$$

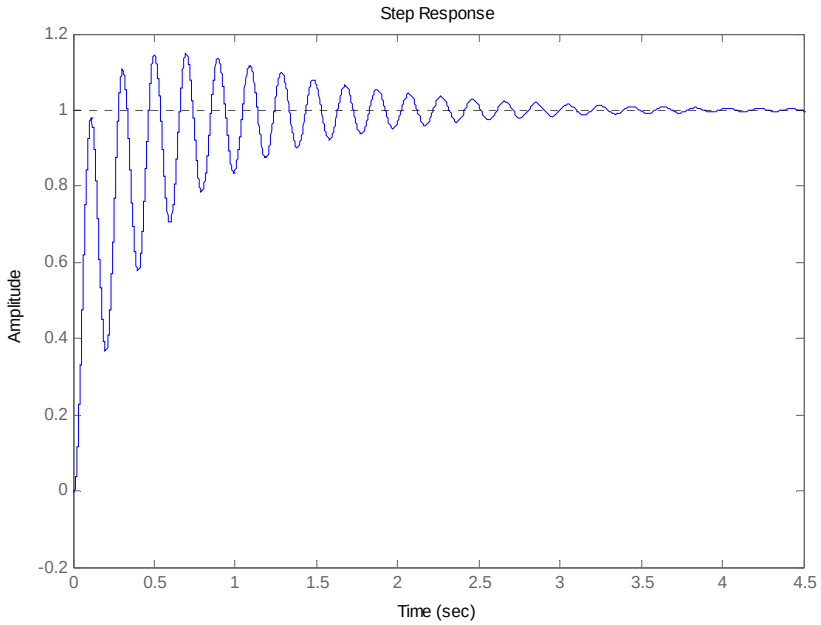


Figure VII.7 Motors Controller 1 Step Response

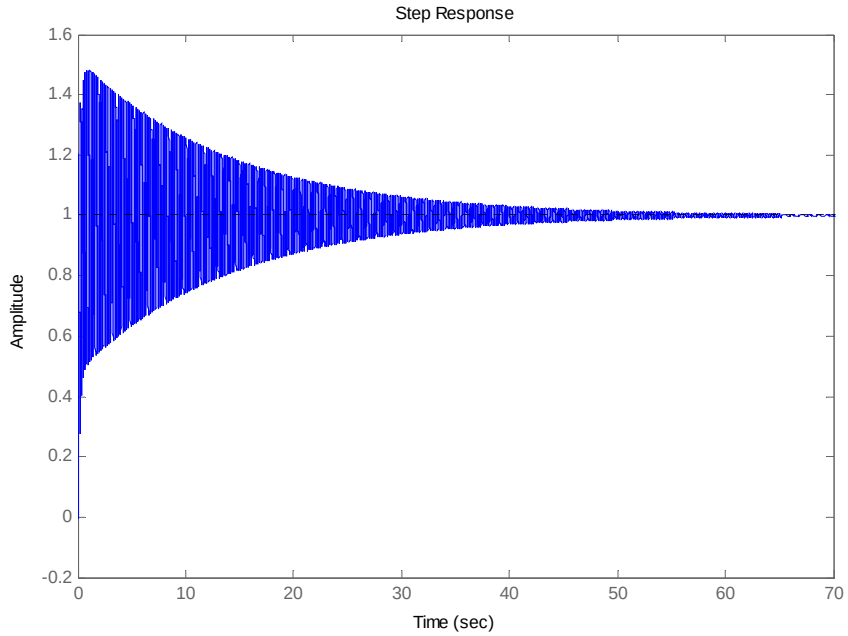


Figure VII.8 Motors Controller 2 Step Response

For these controllers K1=12.2401, Ti1=$(7.983)^{-1}$, K2=19.6085, Ti2=$(2.808)^{-1}$ and h=0.02; also, Tr was set equal to Ti.

These controllers were designed before implementing the PLL and thus the PWM period was 5 times smaller; therefore, to obtain the same duty cycle, the controller gain has to be multiplied by 5. The new quantities are: K1=61.2005, K2=98.0425, ie1=9.77, it1 =0.15966, ie2=5.50 and it2=0.05616.

K1 and K2 were stored in $Q_{7,24}$ format which resolution is r1=5.9604x10^{-8} and its range goes from -128 to (128 – r1); ie1 and ie2 were stored in $Q_{4,27}$ format which resolution is r2=7.4505x10^{-9} and its range goes from -16 to (16 – r2) and finally it1 and it2 were stored in $Q_{1,30}$ format which resolution is r3= 9.3132x10^{-10} and its range goes from -2 to (2 – r3).

Both controllers were tried and C1 response was too oscillatory, thus C2 was decided to be used. However, starting from the stationary position resulted slowly because of the static friction; in order to solve this, a smooth start-up algorithm was used that increases the PWM DC at low motor frequencies. Figure VII.9 shows the increase of PWM DC according to the Motor Frequency.

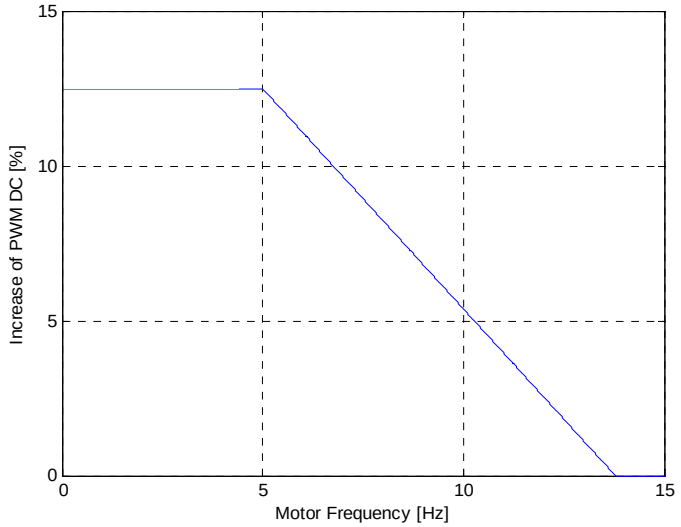


Figure VII.9 Smooth start-up increase of PWM DC vs Motor Frequency

Battery Management

In order to charge the Li-Ion battery, the Constant Voltage Charging method is used. It consists of two steps: the first one where the charging current is limited at 1C (constant current) and the second one where battery is charged at a constant voltage of 8.4 [V] (Simpson 2007). This charging process is shown in Figure VII.10.

The constant voltage and current are set through the Battery Controller, just passing ass reference and output the actual voltage and current values respectively. In order to determine which of the two charging modes has to be used, the state machine shown in Figure VII.11 is applied.

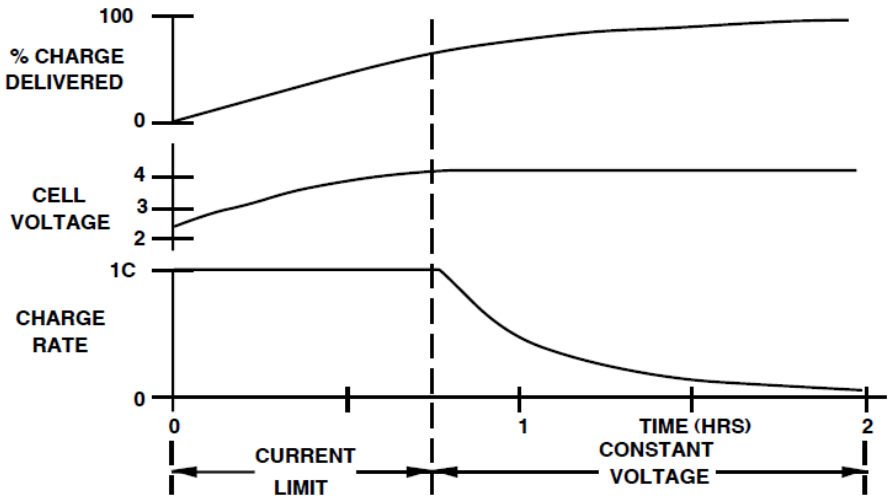


Figure VII.10 Battery Charging Process (Simpson 2007)

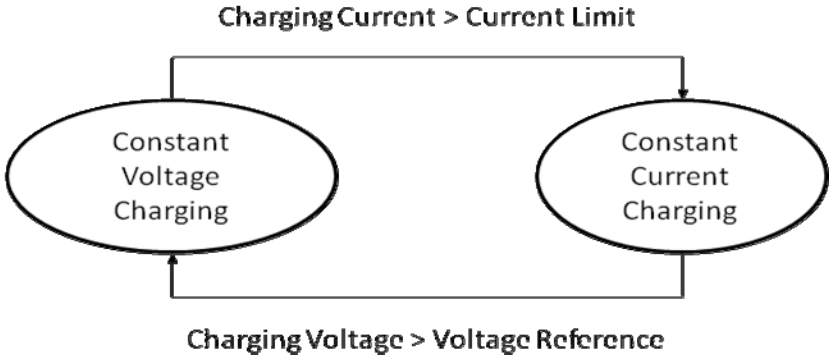


Figure VII.11 Charging State Machine

It is also important to determine the current battery state of charge (SoC); however this is for demonstrative purposes and thus, there is no need for complex systems with high accuracy. Instead, it was decided to use a simple method in order to save developing time.

The first step is to choose the battery model to be used. Generally, the battery is modelled as a voltage supply and different configurations of capacitors and resistors (Hongwen, Rui and Jinxin 2011), from those, the simplest model is the Rint, which is shown in Figure VII.12.

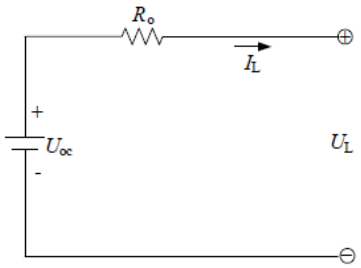


Figure VII.12 Battery Rint Model (Hongwen, Rui and Jinxin 2011)

In this model U_L is the voltage measured at battery terminals, U_{oc} is the open-circuit voltage and R_o is the internal resistance, and it is characterized by the following equation.

$$U_L = U_{OC} + I_L \cdot R_o$$

In order to obtain R_o's value, two different measurements can be subtracted:

$$U_{L2} - U_{L1} = \Delta U_L = \Delta U_{OC} + \Delta I_L \cdot R_o$$

Assuming that the open circuit voltage doesn't change, the last equation can be rewritten as:

$$R_o = \frac{\Delta U_L}{\Delta I_L}$$

In order to calculate R_o a load current step was applied and the battery voltage was recorded, as shown in Figure VII.13, and after that, the past equation was used. The result obtained is R_o=0.63 [Ω], which is an expected value for a two-cell Li-Ion battery.

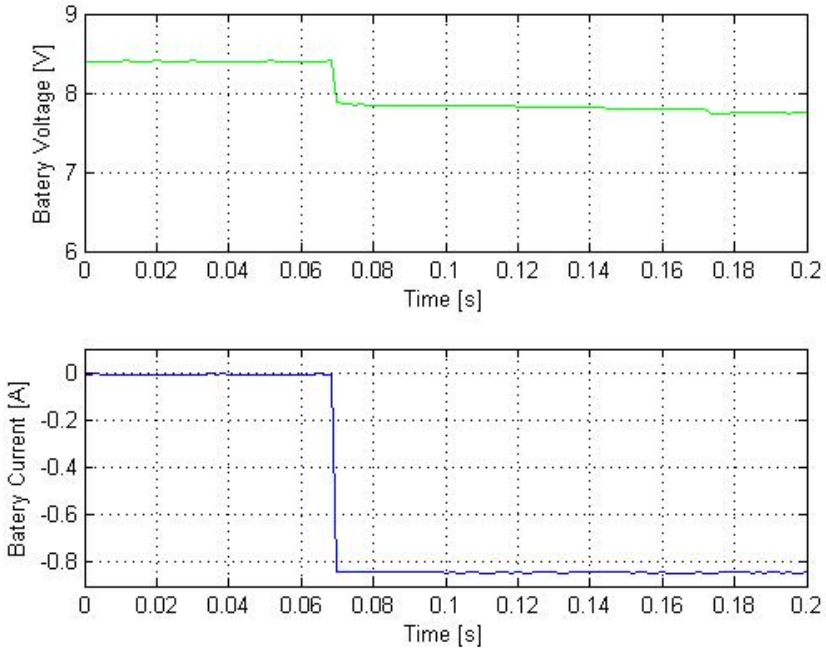


Figure VII.13 Battery voltage measured during a current step

Once the model was characterized, it was decided to use a simple algorithm to determine the SoC. It consists of discharging the battery and calculating at all times the open-circuit voltage and the SoC; then, a look-up table is built with the data of open-circuit voltage and SoC. In order to determine the current SoC it is only necessary to calculate the battery's open-circuit voltage and search in the table for the corresponding SoC (Cadar, Petreus and Orian 2009). Figure VII.14 shows current and voltage data for a full discharge of the battery; it was used then to elaborate the look-up table.

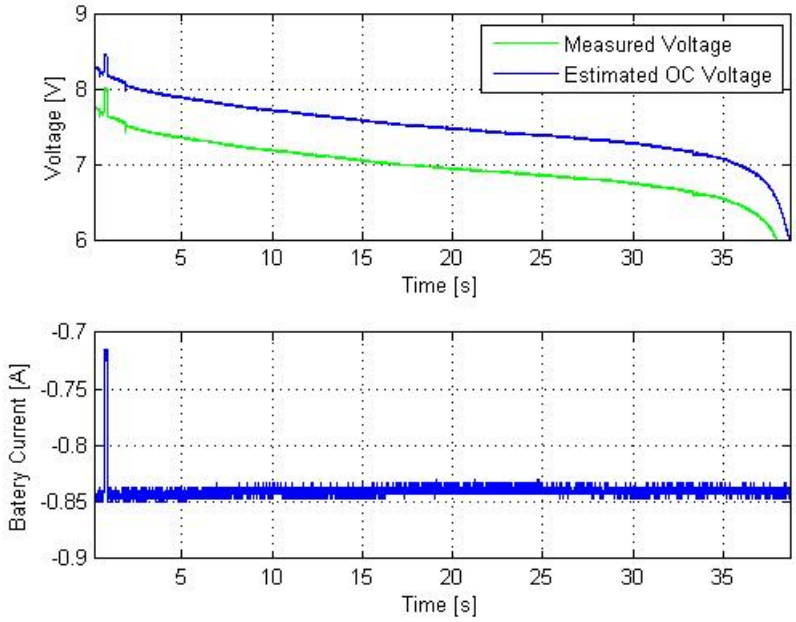


Figure VII.14 Battery Full Discharge at constant current

Code Structure

The MCU performs multiple tasks, including PI controllers, Pulse Width Modulation (PWM), USART communication, analog-to-digital conversions, external and time-triggered interruptions, among others. In order to control the MCU modules needed by these tasks, the following pre-programmed drivers from AVR were used, which include functions to both initialize and use all the peripherals functionalities.

- ANALOG – ADC – Analog to Digital Converter: used to initialize the ADC module and perform convertions.
- CPU – INTC – Interrupt Controller: interfaces the Interrupt Controller to collect the interrupt requests from different modules and send them to the CPU according to a set priority.
- CPU – Interrupt management: to enable and disable global interrupts
- CPU – PM – Power Manager: controls the oscillators and PLLs to generate the clocks for the device.
- GPIO – General-Purpose Input/Output: used to configure the access to the MCU pins.
- MEMORY – EIC – External Interrupt Controller: configures pins that can be used as external interrupt.

- PWM – Pulse Width Modulation: used to generate PWM signals and set their characteristics.
- TIMING – TC – Timer/Counter: configures the three 16-bit timers in the MCU according to the required functionality.
- USART – Univ. Sync/Async Serial Rec/Trans: to configure the USART modules in its available modes.
- UTILITY – Generic board support: includes functions and definitions that are specific for the prototype board used.

In order to simplify the programming and debugging process, each task was developed individually, and all of its functions and variables were included into a library with its code (.c) and header (.h) files. The integration off all the tasks is performed in the Main.c file, which also directs the program flow. All the developed libraries are described below.

ADC Car:

Files: ADC_Car.c and ADC_Car.h

Description: this library is used to initialize the ADC module using its corresponding driver; there are seven signals to be digitalized: battery current (IBAT), battery voltage (VBAT), load current (ICC), line voltage (VL) and three coils voltage inputs. Definitions were made to assign the pin, pin functionality and ADC channel for each signal.

Functions:

- `void ADC_I(void)`: this function is used to initialize the ADC module. It first assigns all the pins needed as ADC inputs, then, since for 10 bit conversions the maximum frequency allowed for the ADC clock is 5 [MHz] (datasheet, p630), a prescaler is set to decrease the 60 [MHz] that inputs the ADC, according to equation 7.X (datasheet, p558).

$$ADC_{Clock} = \frac{MCU_{Clock}}{(PRESCAL + 1) * 2} \qquad \text{VII-1}$$

To accomplish with the previous restriction, the prescaler value is set to 5, for and ADC clock of exactly 5 [MHz]. After setting the prescaler, the ADC is configured and all the channels are enabled.

Battery:

Files: Battery.c and Battery.h

Description: this library is used to manage the battery charging process; it includes a state machine, a PI controller, and PWM generation, all of these to control the Step-Down converter in the Basic Power Board, and thus the charging current and voltage.

First the charging mode is chosen, the voltage or current measurement and their reference are used to calculate a control signal and then generate the PWM that applies it to the Step-Down converter.

Moreover, the following parameters were defined as constants: PWM pin, pin function, channel and period; battery charging voltage and current references and fixed point configuration of the controller variables and gains.

Functions:

- `void Battery_I(void)`: this function initializes the controller gains and the used PWM channel, setting up all its required functionalities.

- `void Set_BatteryDC (int32_t u)`: it applies a new value to the PWM Duty Cycle (DC). The function input is the value to be applied, which must be lower than the period.

- `int32_t Battery_CalcOut (int32_t y_v,int32_t y_i)`: it is used to calculate a new control value, based on the present battery voltage and current measurements that are given as inputs (y_v and y_i respectively). The first step is to check on any transition on the state machine explained above (Figure X.1) and according to the charging mode, it calculates the new control value using the PI discrete controller explained in the Controllers section; this value is the functions output.

- `void Battery_UpdateS (int32_t u)`: this function updates the integration and anti wind-up parts of the PI controller; the input is the saturated control value that was applied to the process.

Fixed Point:

Files: FixedPoint.c and FixedPoint.h

Description: this auxiliary library is used to perform multiplication of two 32-bit numbers in different fixed point formats.

Functions:

- `int32_t FixedP_Mult(int32_t Ain, int32_t Bin, uint8_t n_a, uint8_t n_b)`: this function performs the multiplication of numbers Ain and Bin in fixed point format Q_{n_a} and Q_{n_b} respectively; the result is returned in format Q_0.

- `int32_t FixedP_Mult2(int32_t Ain, int32_t Bin, uint8_t n_a, uint8_t n_b, uint8_t n_c)`: this function performs the multiplication of numbers Ain and Bin in fixed point format Q_{n_a} and Q_{n_b} respectively; the result is returned in format Q_{n_c}.

Follow Wire:

Files: FollowWire.c and FollowWire.h

Description: this library is used to drive the vehicle off the slot, when it follows a reference wire with square current wave that is picked up by the coils placed in the front of the car.

Functions:

- `void FollowW_CalcOut (int32_t C1, int32_t C2, int32_t Speed_Ref)`: this function takes as input the reading of the right and left coils (C1 and C2) and the speed reference at which the vehicle is desired to drive, it then uses a P controller to calculate the amount of control needed to make the reading in both coils equal (this implies that the wire is exactly in the middle between the two coils). According to this control value, speed references for the right and left motors are calculated.

- `int32_t FollowW_GetLS (void)`: this function returns the reference value calculated previously for the left motor.

- `int32_t FollowW_GetRS (void)`: this function returns the reference value calculated previously for the right motor.

Interrupts:

Files: Interrupts.c and Interrupts.h

Description: this library is used to manage all interrupts in the MCU, including the initialization of external interrupts and timers; also the interrupt routines are included here. The parameters defined as constants in this library are the following:

- External interrupt pins and pin functions.
- Timers' configuration constants
- Speed Measuring Algorithm's Constants

Functions:

- `void EI_I(void)`: this function initializes the two pins used for speed measurement. They are configured as external interrupts triggered by a rising edge.

- `void TC_and_Int_I(void)`: this function configures all interrupts. First it initializes the variables to store the motor period – used then to calculate its frequency – and it sets the operation mode for the two timers used: one interrupting at 400 [Hz] with a clock frequency of 7.5 [MHz] to direct the program flow (program timer); and other at 10 [Hz] with clock frequency of 468.75 [KHz] to measure the motors speed (speed timer).

After this, the function register the interrupt routines used for PC communication, timers and external interrupt pins, and initializes all of them.

Interrupt Handlers:

- `static void usart_int_handler(void)`: this interrupt is called after receiving a byte from the PC. It puts the byte into the reception buffer and it checks if the last 11 bytes received correspond to the format of message four (Communication Protocol p100). If so, all the received instructions are stored.

- `static void tc_irq(void)`: this routine executes at a frequency of 400 [Hz] and it is used to enable flags at 400, 50, 25, 16 and 5 [Hz] in order to direct the program flow. It also checks if a message has been received over the IR USART.

- `static void speed_tc_irq(void)`: this interrupt corresponds to the speed timer, used for speed measuring, and actually is the timer count what is used by its algorithm. However, since it is also important to know it there has been an overflow in the counter, this routine updates the number of overflows that had occurred for every motor. Also, if the overflow number is higher than the allowed, the speed measurement algorithm is reset.

- `static void eic_int_wheel1(void)`: this routine is called after a rising edge is detected in the signal for the left motor that comes from the Measurement Speed Board. It stores the speed timer counter value and calculates the difference between the current one and the last one recorded, taking in consideration any counter overflow. The result of the last operation is the left motor period divided by four, and it is used to calculate its frequency.

Moreover, not only the last value calculated is stored, but the last eight values, which are used to apply a Moving Forward Average Filter; this means that the motor frequency is determined using as the period the average of the last eight values.

- `static void eic_int_wheel2(void)`: this interrupt has the same functionality as the last one, but it is used for the right motor.

IRID:

Files: IRID.c and IRID.h

Description: this library initializes the USART module 0, which is used to receive the IRID messages. The address of this USART module, its pins and pins functions, and the clock and baud rate configuration are defined as constants.

Functions:

- `void IRID_I(void)`: it initializes the USART in RS-232 mode with 8-bit, no-parity and 1-stopbit communication. Also, this function assigns the corresponding pin to function as USART Rx input.

Motor:

Files: Motor.c and Motor.h

Description: this library is used to control the speed of the motors running the vehicle; it includes a PI controller with an extra algorithm for smooth start up (as explained in Controllers) and PWM generation for each motor (left and right).

The following parameters were defined as constants:

- PWM pins, pins function and channels.
- PWM period.
- Pins to enable the motors and control their direction.
- Fixed Point Configuration of the PI controllers' variables and gains.
- Smooth start-up parameters.

Functions:

- `void Motor_I( void )`: this function initializes the controller gains and the used PWM channel, setting up all its required functionalities.

- `void Set_MotorRef (int32_t ref_l_n, int32_t ref_r_n)`: this function sets a new speed reference for the left and right motors (ref_l_n and ref_r_n respectively).

- `void Set_MotorDC (int32_t u_l, int32_t u_r)`: it applies a new value to the left and right motors PWM Duty Cycle (DC). The function inputs are the values to be applied, which must be lower than the period.

- `int32_t MotorL_CalcOut (int32_t y_l)`: it is used to calculate a new control value for the left motor, based on based on its current speed (y_l). The first step is to calculate the new control value using a PI discrete controller; then an extra amount is added if required according to the smooth start-up algorithm; all of this is done as it was explained in the Controllers section. The new control value is the functions output.

- `int32_t MotorR_CalcOut (int32_t y_r)`: this function is the same as the past one, but it is used for the right motor.

- `void Motor_UpdateS (int32_t u_l, int32_t u_r)`: this function updates the integration and anti wind-up parts of the PI controllers; the input is the saturated control value that was applied to the left and right motor.

PC Com:

Files: PC_Com.c and PC_Com.h

Description: this library initializes the USART module 1, which is used to communicate with the PC via Bluetooth. The address of this USART module, its pins and pins functions, and the clock and baud rate configuration are defined as constants.

Functions:

- `void PC_Com_I(void)`: it initializes the USART in RS-232 mode with 8-bit, no-parity and 1-stopbit communication. Also, this function assigns the corresponding pins to function as USART Rx input and Tx output.

- `void USART_CHARtoASCII(uint16_t to_send)`: this auxiliary function is used to send an unsigned, 8-bit variable in ASCII format, so it can be visualized in Realterm in the PC.

- `void USART_UINT16toASCII(uint16_t to_send)`: similar to the past function but it sends unsigned, 16-bit variables.

- `void USART_UINT32toASCII(uint32_t to_send)`: similar to the past function but it sends unsigned, 32-bit variables.

PC Protocol:

Files: PC_Protocol.c and PC_Protocol.h

Description: this library is used to send messages to the PC according to the designed Communication Protocol.

Functions:

- `void Send_Message1(uint16_t value_vbat, uint16_t value_vline, uint16_t value_ibat, uint16_t value_icc)`: this function sends the message 1 as described by the protocol; in order to do this, it takes as input the battery voltage (vbat), line voltage (vline), battery current (ibat) and load current (icc).

- `void Send_Message2(uint16_t value_coil1, uint16_t value_coil2, uint16_t value_coil3, uint16_t value_coil4)`: this function sends the message 2 as described by the protocol; in order to do this, it takes as input the measurements of coils 1 to 4.

- `void Send_Message3(uint16_t value_position, uint16_t value_speedl, uint16_t value_speedr, uint16_t value_SoC)`: this function sends the message 3 as described by the protocol; in order to do this, it takes as input the vehicle position, left and tight motor speed and battery State-of-Charge (SoC).

Servo:

Files: Servo.c and Servo.h

Description: this library is used to generate a PWM for controlling the servo that is responsible to take up and down the guiding blade. The configuration parameters defined as constants are the following:

- PWM pin, pin function and channel
- PWM period and maximum and minimum allowed duty cycle
- DC to place the servo up, down or in the middle according to each vehicle.

Functions:

- `void Servo_I( void )`: this initializes the PWM, it assigns the corresponding pin to output the PWM, which is then configured at 50 [Hz]. It also places the guiding blade in the upper position for default.

- `void Servo_GoDown( void )`: this function takes down the guiding blade; this is the position used to drive into the slot.

- `void Servo_GoMidle( void )`: it places the guiding blade in a position that lifts the front wheels; this is used when the car is driving off the slot, to avoid that the friction produced by the front wheels stops the car from turning.

- `void Servo_GoUp( void )`: this function takes up the guiding blade in order to exit the slot.

State of Charge (SoC):

Files: SoC.c and SoC.h

Description: this library is used to calculate the current battery SoC and whether it is charging or not and then it displays them in the RG LED on top of the car. In order to do this, two PWMs are used to control the brightness of the red and green colors of the LED.

The full green color indicates 100% of charge and red 0%, and all the colors in the middle are used to represent the middle values; also, if the battery is discharging the LED blinks.

Functions:

- `void SoC_I( void )`: this function is used to initialize both PWMs and the look-up table used to determine the SoC.

- `int32_t Get_BatterySoC (int32_t vin_bat, int32_t iin_bat)`: it calculates the SoC based on the inputs of battery current and voltage (iin_bat and vin_bat).

- `void TURN_ON_LED(void)`: it turns on the RG LED in the color correspondent to the calculated SoC.

- `void TURN_OFF_LED(void)`: this functions turns off the RG LED and it is used to add the blinking functionality.

Program Flow

The main function integrates all the libraries and directs the program flow. It first initializes oscillator 0 in external crystal mode - driven by a 12 [MHz] external oscillator - and configures the PLL 0 to achieve a clock frequency of 60 [MHz]. Then it calls the initialization functions for all libraries – Motor Controller, Battery Controller, Servo, SoC, ADC, External Interrupt Pins, PC communication, IRID, Timers and Interrupts – and after this, it enters into an infinite loop.

This loop is composed by five different code sections which execute at a specific frequency. Each of these sections is surrounded by an “if” that is conditioned by a flag. This flag is enabled in the program timer interrupt according to the repetition frequency of each code section. The flowchart for the infinite loop is shown in Figure VII.15.

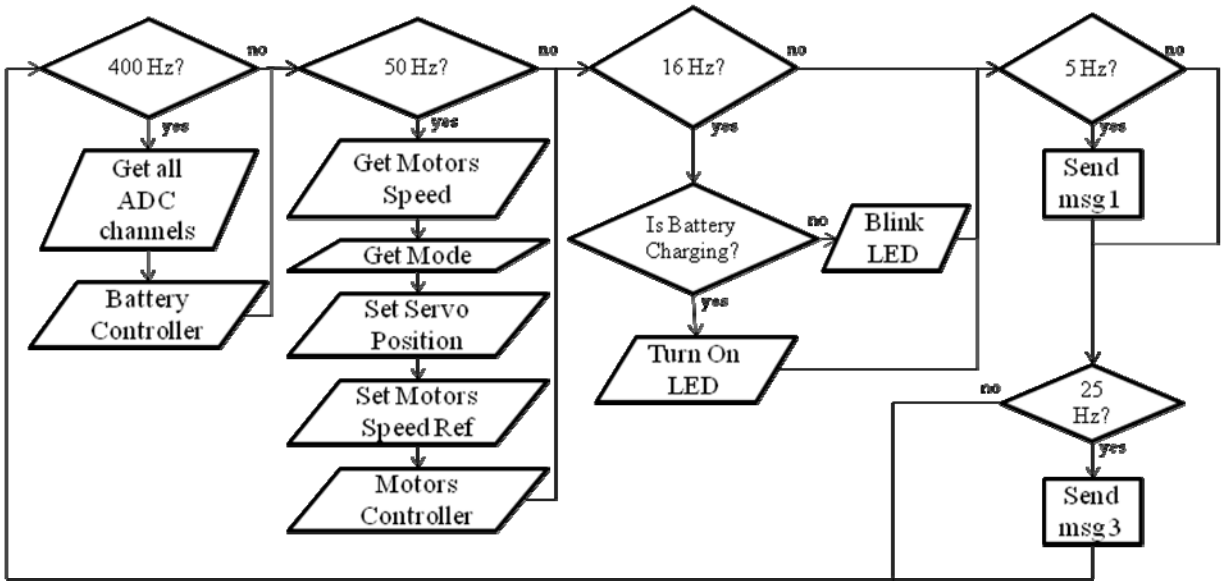


Figure VII.15 Vehicle's Main Program Flowchart

The first section executes at 400 [Hz] and it performs the analog-to-digital conversion of all analog signals and stores its value; then it calculate and apply the control signal correspondent for the battery voltage (or current).

The second code section runs at 50 [Hz]; it first calculates the motors speed based on the period measurement, then, it gets the current driving mode – on or off the slot – and the corresponding servo position; also, based in the driving mode, it calculates the speed references for both motors and performs the controlling action on them.

The next section is in charge of displaying the SoC value on the RG LED. If the battery is charging it just turns on the LED in the corresponding colour; but if its not, it makes the LED blink. The repeating frequency of this section is 16 [Hz].

Finally, there are sections running at 5 and 25 [Hz] in charge of sending the messages 1 and 3, respectively, according to the Communication Protocol (p100) with the PC.

VIII. PC SOFTWARE

Overview

The PC software developed for the project serves three basic purposes:

1. Providing user feedback and control of various states of the track and the vehicles.
2. Issuing commands and handling feedback regarding various states of the vehicles.
3. Issuing commands and handling feedback regarding various states of the track.

It could be argued that the above points could be summed up in a single point stating, “The purpose of the PC software is to provide a user interface with which a human user can control and monitor the states of the vehicles and the track”. However it is desirable to have this division of purposes in order to create a more simplistic per-task approach in describing how the overall system is designed and implemented.

The tasks that needed to be completed in order to cater to the purpose of the software were:

- Selecting a programming-language.
- Develop communications between the PC and the model hardware
- Develop the autonomous control and state handling.
- Enable user interaction.

The tasks are approached in a somewhat concurrent manner because of the intricate dependencies between them.

Language

As the goals of the project were initially formed one of the ideas were to have the users interact with the track and vehicles using a small form factor device such as a tablet or smartphone. It was also decided that a central machine that controls the entire system is required. Because of the limited time-constraint it was deemed impossible to write separate applications for the central system and the handheld devices. Therefore it was decided to write a single cross-platform application that would function properly but could turn features on or off depending upon the device on which it was residing at any given time. The choice of language became Java, which offers the possibility to write code that would be easily

interpreted and run by any given system that supports it, including most of the desktop operating systems and the leading smartphone operating system – Android.

Java executes on a so-called virtual machine. The virtual machine exists in various incarnations depending on the platform upon which it runs. The idea is that hardware implementations are abstracted away from the code the developer writes. It is achieved by compiling the source-code written by the developer into universal byte-code, which the virtual machine understands, and the recompiling it at run-time for the hardware-implementation for which the virtual machine is developed.

Communications

Going back to the points of purposes of the PC – software as described in the overview; the last two points are very similar in nature regarding the type of technical solutions that need to be devised to in order to serve them but they vary greatly in the actual physical representation that they embody. The key differences being that the track is a stationary object whereas the vehicles are constantly in motion and that the track and vehicles have different objectives and therefore different states and operation-flows. It is important however to find similarities and consolidate them in order to facilitate a faster production of code that is less error-prone and simplifies maintenance.

The identified common tasks are (by order of events in which they initially occur):

- Autonomous command issuing
- Communications
- State response and handling

The command issuing and the state response and handling tasks seem to imply that the implementation should vary greatly because of the different nature of the systems. However the only aspect that truly distinguishes these two are the commands and states involved. Therefore the only difference on the PC-side occurs at the stage where the decision what command to issue and how to handle a received state is made. It is then beneficial to consolidate the interface at which these messages cross, which is the communications layer.

The track is a stationary object and the vehicles are constantly in motion. The task of the vehicles is to represent a functioning model that can operate without having to be tethered to an external power-source. This leads the cars requiring a wireless form of communication whereas the track lends itself to utilize a more robust tethered form of communication. Using two different forms of communication implies two separate software-stacks are needed to

accommodate these. However it is not preferable to maintain two separate communication software-stacks from a maintainability point of view.

The choice of using RS232 for the tethered communication and Bluetooth for the wireless alleviates the problem of maintaining two separate software-stacks for communication. This is achieved by utilizing the RFCOMM-component provided by the Bluetooth protocol-stack. RFCOMM provides a way to utilize the same code written for RS232 but over a wireless link. It is because of that RFCOMM is sometimes referred to as "wireless RS232" or "serial port emulation". The idea is that when a Bluetooth – enabled device connects to a PC using the RFCOMM – protocol it is recognized as a RS232 – COM port in the top-level device stack.

The communication interface between the physical world and the virtual world comes down to communicating with RS232-ports. Because the language chosen for the project is java, direct access to hardware such as RS232-ports is not available. It is required to utilize a library which has native code compiled for using the RS232-ports of the specific implementation of hardware the software is to run on. For this project the open-source RXTX library was utilized.

Autonomy and control

Because the vehicles are to be able to drive around the track with minimal user-interaction and the track has to be able to turn sections on or off depending on if a vehicle is residing on it, a system that allows for autonomous behaviour of the track and the vehicles has been devised. The behaviour is defined by a simple set of rules that the system needs to conform to:

Track:

1. If a car is situated on a section, and the section doesn't have an uncleared short-circuit detection-flag, turn it on and the two following it.
2. If a short-circuit is detected, raise a short-circuit flag and turn the power to the section off.
3. If a car is on a section that is a lane-switching section, switch the lane depending upon the state of the car's lane-switching variable.

Vehicle:

1. When reaching a certain section, drive at the speed defined for that section

2. If a vehicle is driving two sections ahead, slow down and resume normal speed when there is no longer a vehicle on the next section or two sections ahead.
3. If a vehicle is driving on the section ahead, come to a full stop and resume normal speed when there is no longer a vehicle on the next section or two sections ahead.
4. If approaching a section which is the last section before a corner section. Stop if there is a car driving on the opposite lane in the same corner.
5. Upon arriving to a section with no slot, raise the guiding mechanism.

The approach used to obtain the desired behaviour is to identify all the properties that require monitoring and divide them into properties of either the vehicles or the track.

The properties are as follows:

Vehicle:

1. Current section
2. Next desired section
3. Guide mechanism up/down

Track

1. Section power on/off
2. Lane switcher directions

Of these properties, only property 1 of the vehicles and the track are monitored by reading back actual values from hardware. The others are properties defined upon initialization and the hardware has no way of reading back these values. There is however no clear distinction between properties from the models point-of-view as these properties are read on a per-needed basis but written to continuously by separate threads and the design lends it self to be implemented in such a way that the remaining properties be defined by values read back from hardware-sensors.

Every object for which action must be taken, be it a vehicle or the track, has it's own thread in which decisions are made based upon the current state of the properties. This of course may lead to question the stability of the system because of the volatile state of each property. It has however been concluded that the system behaves properly with no observable concurrency problems.

Classes description

GraphicalTrackControlApplication:

This is the class containing the main-method, which is the entry point for the software. This class initializes the models and the graphics.

Car/(Track/TrackHandler):

These classes represent the states of the various vehicles and the track. The Car-classes are responsible for their own internal handling of state-change. The Track and TrackHandler have been split up where Track is only a state-container and TrackHandler is responsible for correctly handling the state-change.

SerialCommunicator

This class handles serial communication across various COM-port interfaces attached to the software

SerialEventHandler

This class handles those set of bytes obtained by the SerialCommunicator – class that are considered various events

Interpreter

This class interprets events dispatched by the SerialEventHandler-class and creates different types of objects representing the events.

InterpretedMessageHandler

This class handles messages interpreted by the Interpreter class and dispatches them to the correct objects.

Scene

This class is responsible for drawing the graphics.

Camera

This class handles translation of the viewport depending on mouse input

Hud

This class handles the state of the interface displaying the data of the vehicles.

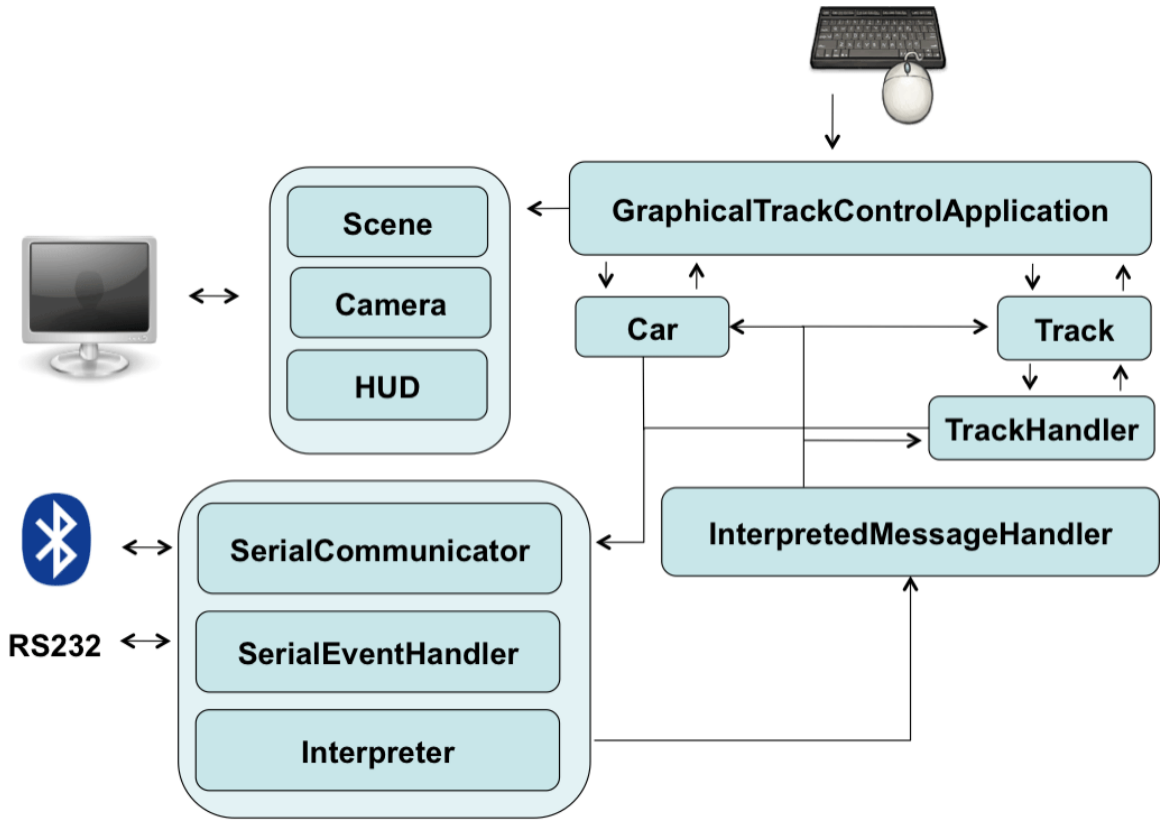


Figure VIII.1 Classes diagram of the pc-control software

User Interaction

In order to control the vehicles and the track, a graphical user interface (henceforth GUI) is devised. Because some level of user interaction by those who come to observe the model is expected it is considered necessary to make the GUI as intuitive as possible. The GUI is therefore developed in such a way that it only displays symbols representing the various parts of the model and interactivity is obtained by clicking on the symbols.

The GUI's parts are the following:

1. Symbols representing the vehicles with some data regarding the current speed and state of charge of the battery.
2. Data for the selected vehicle displaying current consumption, voltage levels and total power consumption.

3. Round, colored symbols representing vehicles with corresponding colors. The position of the symbols change depending on the actual position of the vehicles on the track. The arrow points in the direction of travel of the current section.
4. The track with divisions defining each section. The various parts of the track light up in different colors depending on their current state with following definitions:
    - Green: section is on
    - Red: section is short-circuited
    - Yellow: command to section is being dispatched

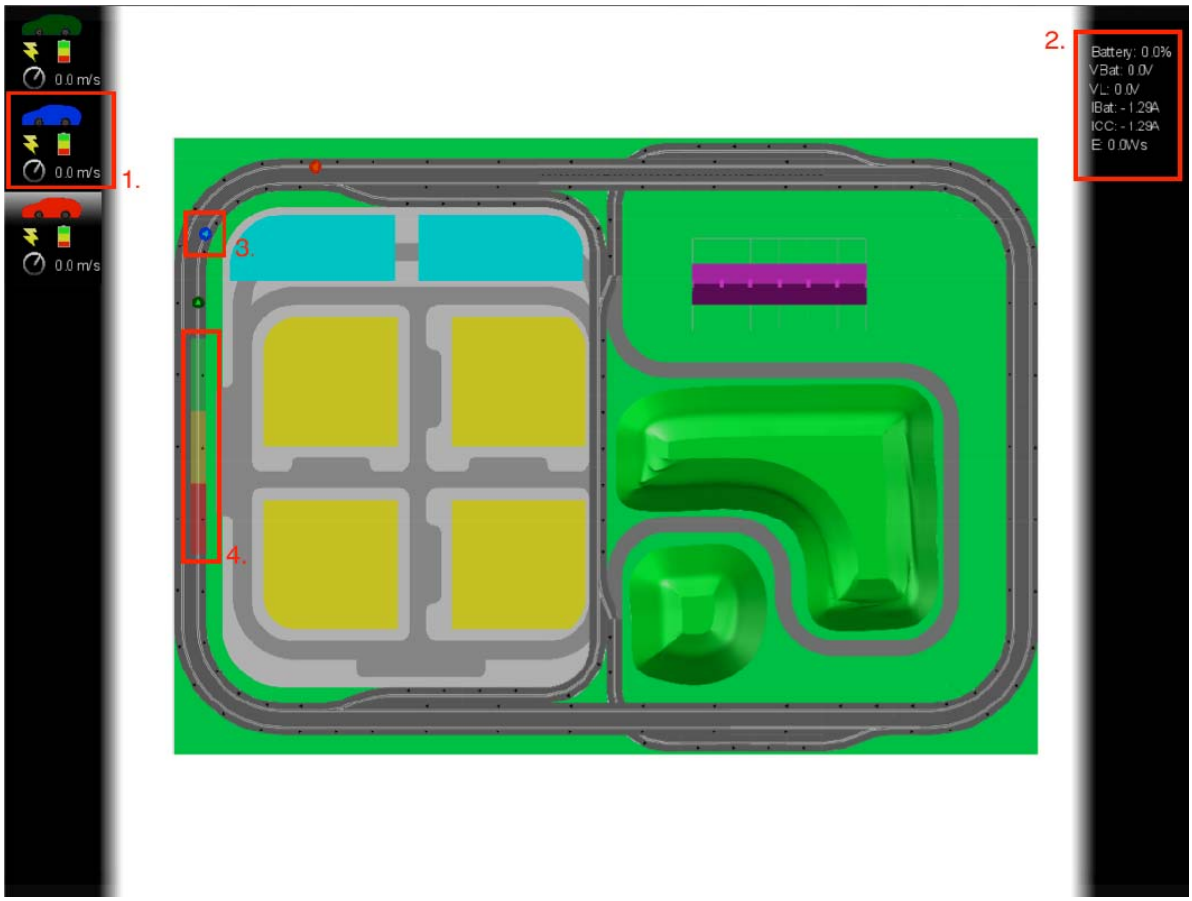


Figure VIII.2 Gui

By clicking on the vehicles on the left pane, the user selects the current user-controlled vehicle. The data on the right pane is representative of the currently selected vehicle. When a vehicle is selected it is possible to:

- Start or stop the vehicle
- Manually increase or decrease the speed
- Raise or lower the guiding mechanism

The sections are manipulated by directly clicking on a section. The user can choose to enable or disable the supplied power and, if raised, clear the short-circuit flag.

By pressing a predefined set of keys on the keyboard, the user can manipulate the track selection arms

Because it is decided that the software should be portable to the Android platform, the GUI was developed using a cross-platform graphics library called lib-gdx. The library runs on any platform that supports Java and OpenGL. OpenGL is an open-source graphics library initially developed by Silicon Graphics Inc. in 1992 and is, because of its hardware-agnostic nature, the most widely used graphics library in the world. Lib-gdx builds on top of OpenGL by wrapping it in Java code and also adds a lot of constructs to simplify the creation of graphical applications.

The graphical elements are created using Adobe Photoshop and placed into so-called "sprite"-objects. Sprites are 2-dimensional graphical elements defined by a bounding box, position and image. By utilizing the possibility of defining an "alpha"-channel in the images as to define which parts of the image are to be considered fully transparent it is possible to create arbitrary shapes.

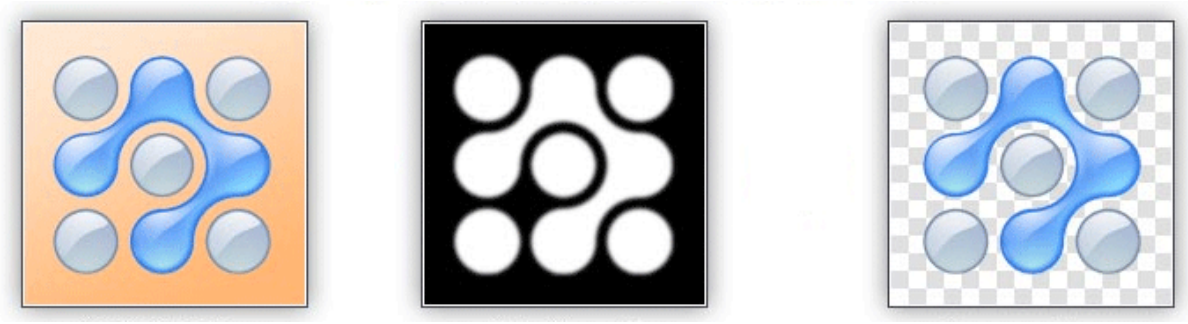


Figure VIII.3 A sprite beginning as an square image (left) then with applied alpha channel (black in center image) results in an arbitrary shape (right)

IX. CONCLUSIONS

This master thesis project consisted in the development of a full-scale demonstration system of Slide In technology. From original Scalextric track pieces, it was possible to design and develop a structure that involves the several features of this mentioned technology. The system development was distributed in the design and implementation of the Track, the Vehicle units and PC software simultaneously.

Scalextric tracks were modified mechanically and electrically in order to build the track system with the addition of electronic devices that monitor and control all track sections.

On the other hand, it was successfully designed cars and trucks models, which included all the desired capabilities. All the vehicles have a set of five custom electronic boards that were designed and manufactured to achieve both functionality and space requirements in the vehicles. The mechanical performance of the moving units was also adapted from original Scalextric cars parts.

Every vehicle has several features such as Bluetooth communication, speed measurement, a rechargeable battery from the road, infrared receiver to Local Position System, a servo mechanism in order to get in/out the road slots, and among others that involves high performance full – scale vehicles with the capability of performing Slide In concept.

Furthermore, it was developed PC software that is in charge of coordinating the track and vehicle unit's interaction plus to perform traffic controller that avoids cars to collide in the track.

Despite of the fact that the requirements to perform a demonstration system of Slide In technology were fulfilled, it appeared difficulties that often impeded the desired performance of the system.

Firstly, the simultaneous development of the vehicles mechanics and electronics obstructed to test the integrated system until the final stage was reached. Therefore, there was not enough time to improve as desired the performance of the vehicles units.

The main problem generated by an uncompleted integration stage of vehicles mechanics and electronics was the Speed Measurement system, which is very sensitive to noise, especially when the cars are charging and the motors are running. This issue was

related to a poor mechanical implementation of this system; vibrations in the motor axis and an unstable position affect the plastic foil disc that interrupts the light beam between infrared receiver and transmitter. The lack of time avoided the development of a more reliable system.

Secondly, one of the main drawbacks in the vehicles was related to the mechanical implementation. Due to the set configuration of the mechanics, the cars were not able to run curve trajectories i.e. the vehicles can not follow the guiding wire in non-slot track sections. The best solution to fix this problematic would be the design of a new mechanics configuration in the vehicles.

Finally, the track performance was one of the most important points in the development of this system because a good performance can facilitate to the public the understanding of Slide In concept. However, the execution of commands that control the track sections depends on the communication bridge PC – slaves, and I2C, which is a notable part of this bridge, fails often so it introduces considerable delays between the PC and the track.

X. REFERENCES

ANALOG DEVICES. *ADR3412/ADR3420/ADR3425/ADR3430/ADR3433/ADR3440/ADR3450 datasheet.* Datasheet, : ANALOG DEVICES, 2010.

Årzén, Karl-Erik. *Real-Time Control Systems.* Lund, 2009.

ATMEL. *32-bit AVR® Microcontroller AT32UC3B0256 Preliminary.* Datasheet, San Jose, CA: Atmel, 2010.

ATMEL. *ATmega48/V ATmega88/V ATmega168/V datasheet.* Datasheet, San Jose, CA: ATMEL, 2011.

ATMEL. *Atmel AVR042: AVR Hardware Design Considerations.* Datasheet, San Jose, CA: ATMEL, 2011.

—. "Atmel's Web Page." *Atmel Product Information.* 2007. http://www.atmel.com/dyn/resources/prod_documents/EVK1101_Schematics_RevB.pdf (accessed 2011).

ATMEL. *ATtiny2313/V preliminary.* Datasheet, San Jose, CA: Atmel, 2010.

BUR-BROWN PRODUCTS. *High-Side, Bidirectional CURRENT SHUNT MONITOR - INA170.* Datasheet, Dallas: Texas Instruments, 2010.

Cadar, Dorin V., Dorin M. Petreus, and Cristian A. Orian. "A method of Determining a Lithium-ion Battery's State of Charge." *15th International Symposium for Design and Technology of Electronics Packages.* Gyula, Hungary, 2009. 257-260.

EVERLIGHT ELECTRONICS CO. *Chip Infrared LED With Right Angle Lens datasheet.* Datasheet, : Everlight Electronics, .

EVERLIGHT ELECTRONICS CO. *Technical Data Sheet Photocoupler-RoHS Compliant.* Datasheet, : Everlight Electronics, 2007.

EXAR CORPORATION. *LOW-NOISE LDO VOLTAGE REGULATOR SPX5205 datasheet.* Datasheet, Fremon, CA: Exar Corp., 2008.

Hart, Daniel W. *Electrónica de Potencia.* Madrid: Pearson Educación, S.A., 2001.

Hongwen, He, Xiong Rui, and Fan Jinxin. “Evaluation of Lithium-Ion Battery Equivalent Circuit Models for State of Charge Estimation by an Experimental Approach.” *Energies*, 2011: 582-598.

International IR Rectifier. *PRELIMINARY IRFR/U9024N* . Datasheet, El Segundo, CA: DatasheetCatalog.com, 1997.

IQD FREQUENCY PRODUCTS. *CFPS-9 SMD CLOCK OSCILLATORS preliminary specifications.* Datasheet, Crewkerne: IQD Frequency P., 2008.

KINGBRIGHT. *SMD CHIP LED LAMP Preliminary Specifications.* Datasheet, Kingbright, 2007.

Kuo, Benjamin C. *Sistemas de Control Automático.* Prentice Hall, 1997.

Laboratorio C USB. “EC3883 - Presentacion Laboratorio de Proyectos.” Caracas: Lab. C USB, 2010. 6-7.

MAXIM. *MAX3222/MAX3232/MAX3237/MAX3241 datasheet.* Datasheet, Sunnyvale: DatasheetCatalog.com, 1999.

NXP Semiconductors. “UM10204 I2C-bus specification and user manual.” 2007 йил 19-June. http://www.nxp.com/documents/user_manual/UM10204.pdf (accessed 2011 йил February).

RCTOYS. *www.rctoys.com.* http://www.rctoys.com/pdf/hitec-servos/HIT-HS55.pdf (accessed 2011).

Resnick, Robert, David Halliday, and Keneth S. Krane. *Física Volumen I.* México D.F.: Editorial Continental, 2002.

Resnick, Robet, David Halliday, and Kenneth S. Krane. *Física, Volumen II.* México D.F.: Editorial Continental, 2002.

ROVING NETWORKS. *Class 1 Bluetooth Module RN-41 Preliminary Specifications.* Datasheet, Los Gatos, CA: Roving Networks, 2009.

Sadiku, Matthew N.O, and Charles K. Alexander. *Fundamentos de circuitos eléctricos.* Mexico, D.F.: McGraw-Hill, 2006.

Sedra, Adel S., and Kenneth C. Smith. *Circuitos Microelectrónicos.* México D.F.: McGrawn-Hill, 2006.

SGS THOMPSON MICROELECTRONICS. *PUSH-PULL FOUR CHANNEL DRIVER WITH DIODES - L293D L293DD datasheet.* Datasheet, ST SGS THOMPSON, 1996.

Simpson, Chester. "Battery Charging." *National Semiconductor Web site.* 21 March 2007. http://www.national.com/assets/en/appnotes/f7.pdf (accessed February 28, 2011).

TEXAS INSTRUMENTS. *CMOS VOLTAGE REFERENCE REF30xx Datasheet.* Datasheet, Dallas: TI, 2008.

TRACO POWER. *DC/DC Converters TSR-1 Series Specifications.* Datasheet, Zurich: TRACO POWER, 2009.

Traylor, Roger L. *USART and Asynchronous Communication.* Corvallis, Oregon: Department of Electric and Computer Engineering, Oregon State University, 2009.

Vishay Semiconductors. *TSOP58038 datasheet - IR Receiver Module for Light Barrier Systems.* Datasheet, : Vishay, 2011.

Würth Elektronik. *Specification for release 7447709101.* Datasheet, : WE, 2011.