Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

School of Engineering

Division of Mechatronics and Information Technologies
Graduate Programs

Master of Science
Major in Intelligent Systems
Thesis

# CAN-based Network System for Speed Control of an Autonomous Vehicle

by
Jesús Alfredo Del Bosque Garza
786850

Monterrey, N.L., December of 2010

©Jesús Alfredo Del Bosque Garza, 2010.

# CAN-based Network System for Speed Control of an Autonomous Vehicle

By

**Jesús Alfredo Del Bosque Garza**

Thesis

Presented to the Graduate Program in Information Technologies and Electronics

at the

Instituto Tecnológico y de Estudios Superiores de Monterrey

as partial fulfillment of the requirements for the degree of

**Master of Science**

major in

Intelligent Systems

Instituto Tecnológico y de Estudios Superiores de Monterrey

Monterrey, N.L., December 2010

Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

School of Engineering

Division of Mechatronics and Information Technologies
Graduate Programs

The committee members, hereby, recommend that the thesis presented by Jesús Alfredo Del Bosque Garza be accepted as a partial fulfillment of the requirements to be admitted to the degree of **Master of Science**, major in:

**Intelligent Systems**

Thesis Committee:

[illegible]

Dr. José Luis Gordillo Moscoso
Principal Advisor

[illegible]

MsC. Pedro Orta Castañón
Committee Member

[illegible]

Dr. Rogelio Soto Rodríguez
Committee Member

[illegible]

Dr. Ramon F. Brena Pinero
Director of Master of Science in Intelligent Systems
Division of Mechatronics and Information Technologies
December of 2010

# Table of Contents

# Figure List

# Table List

# Abstract

The work hereby presented deals with the partial automation of a utilitarian ground electric vehicle; in particular with the development and implementation of a Network Control System (NCS) using a Controller Area Network (CAN) based bus for speed control. This thesis highlights the use and development of standardized components and protocols in order to provide an easily upgradeable platform for future work, with enough robustness, reliability, and efficiency.

A Programmable Automation Controller (PAC) is used to develop and execute the speed control algorithm, and eventually can act as a human-machine interface via a personal computer. The kinematics involved are those of a rear-wheel differential driven conventional vehicle. An electric power controller is used to manage current and voltage flowing to/from the separately excited electric motor driving the vehicle. To develop the Network Control System based on the CAN protocol, CAN modules and additional specialized interfaces were manufactured, as well as CAN compliant cables and CAN hubs. A wheel speed sensor which functions as an incremental optical encoder was manually assembled in the Robotics Laboratory at the Tecnologico de Monterrey (ITESM). The CAN network is fully operational and has been tested proving to be a reliable channel for critical control information. The speed control algorithm is based on the proportional–integral–derivative (PID) controller model; tuning parameters were calculated and fine tuned via trial and error testing. Also, a fuzzy logic controller was developed to compare its performance against that of the PID.

Three major distinctive phases involve this investigation; starting with the development and testing of the CAN network, followed by the programming of the speed controller algorithms and finally the integration of both into a complete Network Control System.

# Chapter 1 Introduction

Demand for robotic applications has increased in areas that range from exploration to entertainment [Guo, 2008]. The development of mobile robots on the educational, private and military sectors has driven focus for research; modern trends indicate the use of robotic applications for working environments in which the tasks involved are dirty, dull or dangerous [Braybrook, 2004; Álvarez et al., 2006] for operators, making necessary the automation of functions previously done by humans. The development of autonomous mobile robots has demonstrated to be a challenging topic in the fields of computer science, electrical and mechanical engineering.

Automation deals with the detachment of robots from their human operators. Decisional autonomy in robotics is intended to reduce the number of remote operators, change their roles and decrease their workload [Barbier et al., 2009]. During the automation process, an early and crucial stage is the selection and installation of communication channels, sensors and actuators (instrumentation[1]) which will later provide the means for the robot to interact with its surrounding environment.

Autonomous vehicles (A*X*V[2] or AV) are a type of autonomous mobile robots[3] that have gathered significant attention due to characteristics such as precision,

[1] According to the International Society of Automation (ISA) the official definition for instrumentation (ISA standard S51.1) is: A collection of instruments or their application for the purpose of observation, measurement, control, or any combination of these [ISA, 1992].

[2] *X* denotes medium for navigation, e.g. AGV: Autonomous Ground Vehicle; AAV: Autonomous Aerial Vehicle; AUV: Autonomous Underwater Vehicle; ASV Autonomous Surface Vehicle [Barbier et al., 2009].

[3] A distinction is made between these concepts. The term autonomous vehicle usually means the conversion of an existing utilitarian vehicle into an AV; here all or some of the previous knowledge of the vehicles manufacturer is preserved. Whereas a mobile robot usually refers to a robot conceived from scratch, where the design includes the creation of the mobile platform as well as the components needed for automation; highly customized applications can be achieved in this manner [Albores, 2007; Gonzalez, 2004].

repeatability, speed, and most of all, the ability to perform specialized tasks on hazardous environments, where human integrity might be at risk.

A common practice to develop an AV is the instrumentation and automation of utilitarian vehicles, where special care should be taken during the design and implementation of the underlying core information systems of the AV. Reliable protocols and components should be applied to ensure proper and successful operation. Furthermore, as technological advance continues to grow exponentially, to extend the lifespan of applied solutions, widely accepted standardized elements should be employed.

According to DARPA[4] *"An autonomous ground vehicle is a vehicle that navigates and drives entirely on its own with no human driver and no remote control. Through the use of various sensors and positioning systems, the vehicle determines all the characteristics of its environment required to enable it to carry out the task it has been assigned"*.

Over the last two decades one of the main topics of the scientific community, dealing with human-robot interaction environments, has been the relief of disaster scenarios. International associations such as the Institute of Electrical and Electronics Engineers (IEEE) have developed specific research divisions for this area. The term rescue robotics involves systems that are designed to support human first response units during dynamic disaster situations. Applications for rescue robotics include: information gathering of the disaster; hazardous material handling; search, diagnosis and rescue of survivors; quantitative investigation of the damage extent; support for recovery actions and help at evacuation centers [Tadokoro, 2009]; leaving people with the ability to concentrate on high priority actions.

The rescue robotics area requires autonomous vehicles capable of withstanding harsh environmental conditions. High quality standardized components that are able to operate during vibrations, in hot or cold environments, with high electromagnetical noise, humidity, etc., along with complementary software, such as standard communication and navigation protocols, must be used to prototype the AVs acting on these environmental conditions.

Controller Area Network (CAN) is a serial communication protocol that can be used for real-time distributed control. This bus architecture has proven to be a

[4] DARPA is the Defense Advanced Research Projects Agency created in 1958 by the United States of America; although mostly focused on military technological research, its projects have played a fundamental part on the development of new technologies widely implemented outside military applications [Darpa, 2008].

confident medium for modular configurations. With CAN bus, it is possible to implement a Network Controlled System (NCS) or field bus [Davis, 2010] for closed-loop control with relatively low cost and high efficiency. Advantages like less wiring, modularity and interoperability arise from the adoption of this technology.

The automation process can yield as a result functional but complex control architectures. Attempts at the development of an AV, without the application of standardized components and protocols, often result in customized solutions, where only the developers are able to fully understand the know-how involved.

The communication protocol used not only defines software constraints; hardware considerations, like cable types, cable length and network topology, are also dependable on protocol selection. Complex architectures usually require complex data pathways (physical connections), making wiring inside the vehicle a major concern. Maintenance and replacement of customized components, as well as inclusion of new elements may result in a challenging task. The lack of consideration for upgrading capabilities makes these solutions prone to obsolescence.

Therefore, an issue arises during the early development of an AV. When selection of the components for the system's architecture is not based on standards; compatibility, interoperability, robustness, reliability and quality can't be assured; compromising future implementations.

The use of standard certified elements guarantees a general understanding of the applied technology, providing a strong, reliable and common groundwork while maintaining important know-how information for further development and simplifying implementation, maintenance, upgrading times and costs of the system.

The work presented here addresses to prove that a Network Control System based on the CAN standard specification is reliable and robust enough for the development of the control architecture of an autonomous ground vehicle. Using standardized components and protocols that provide minimum cost and a relatively easy implementation, a full control loop can be established.

The objective of this thesis is to successfully develop an in-vehicle network based on the CAN protocol and use it to integrate a closed control system for the speed control module of the vehicles architecture. By developing only the speed control module and demonstrating characteristics such as modularity, interoperability and robustness; scalability can be achieved, thus the general principle behind the development of the complete control architecture can be proved.

In particular the objectives in this research are:

1. Development and implementation of a CAN network for the vehicle.
2. Development of modular CAN interfaces.
3. Subdivision and organization of the system into a distributed network composed of different CAN nodes.
4. Development of a closed-loop NCS with expanding capabilities.
5. Development of a programming solution for the speed control module of the system architecture using a PAC.
6. Establishment of a common ground for future development.

The functions performed by the vehicle constitute different abstractions within the systems architecture; a basic structure contains methods for environmental perception, action planning and control. While these are all tasks performed by humans during a vehicles standard operation, other no so evident functions have to be performed as well. Figure 1-1 shows the basic elements in an AV architecture, where the speed control module is highlighted inside the red square. The complete architecture is composed of different modules, where their interactions are described by information sharing communication channels.

Movement of the vehicle can be traced on a 2-D plane, where localization at any given time can be described by its $\{x, y\}$ coordinates and its orientation $\{\theta\}$. Control is performed in the vehicles trajectory[5] in function of its steering angle $\theta$ and its velocity[6] $\boldsymbol{v}$.

The vehicle used during this research can be modeled beside its Ackerman steering mechanism. Velocity control is crucial for overall operation and efficiency as it plays a major role inside the kinematic model of the vehicle, typically used with dead reckoning techniques for pose estimation and steering control [Borenstein, 1994]. Being the base for speed estimation, the velocity control module constitutes a fundamental element inside the AVs architecture.

[5] A trajectory is defined as an ordered collection of points, composed by positions and orientations [Albores, 2007].

[6] Velocity is defined as the rate of change of displacement over time; it is a vector, which means that both magnitude and direction are needed to define it. Speed is the scalar absolute value of velocity which only deals with the magnitude of the vector.

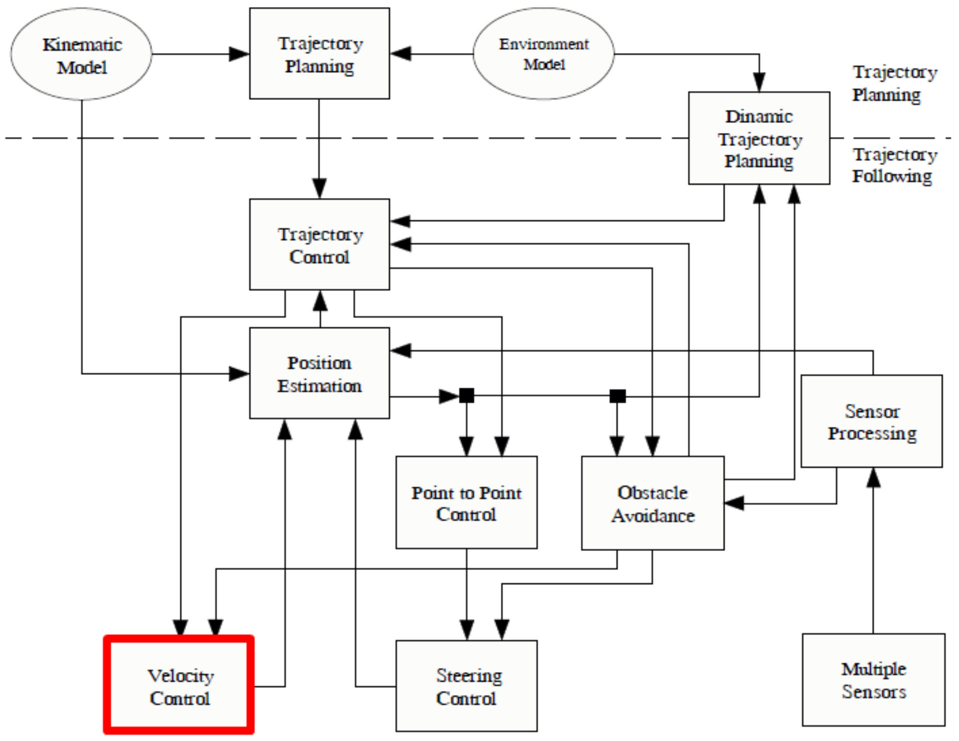


Figure 1-1: Main elements of the control architecture for an AV as proposed by Albores in [Albores, 2007].

Although this thesis deals with speed, it is simple to measure velocity once its magnitude is known, since only direction of the moving vehicle is needed.

The standardized CAN protocol makes possible the integration of information managed by the real-time network with the high computational power of a Programmable Automation Controller (PAC), which offers the industrial ruggedness of a specialized Programmable Logic Controller (PLC) combined with the versatility of a Personal Computer (PC).

The development of a control network using the CAN protocol for the AV to effectively navigate with controlled speed required selection of standard components and design, fabrication and implementation of elements that range from interfaces, to cables and sensors, as well as programming and development of control algorithms within the PAC. The network is also intended to function as a communication channel for future control loops inside the vehicle. An example of a control architecture based on the CAN bus protocol is presented in figure 1-2. Comparing with figure 1-1, it is easy to observe the evident level of simplification achieved by using a bus topology.

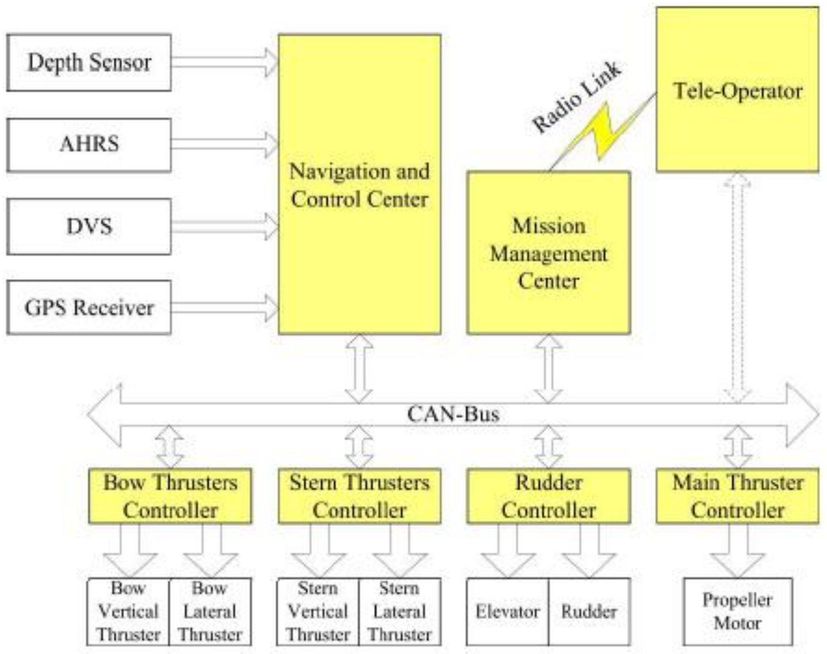


Figure 1-2: CAN control architecture for an AUV presented by [Zhao et al., 2010].

The scope of this thesis covers the implementation of the CAN communication network inside the vehicle for integration of the modules composing the system's architecture. While the complete architecture is integrated by different modules, in this research only the speed control module is developed.

The main contribution achieved is the establishment of a complete closed control loop, composed by three different CAN nodes.

This thesis is divided in 5 chapters and several appendices. Below is brief description of such chapters.

Chapter 2 presents the methodology followed for the successful development and implementation of a Network Control System based on the CAN protocol for speed control of the utilitarian electric vehicle, used in this thesis. A description of the elements in such methodology is presented.

Chapter 3 presents the implementation of the key elements developed for this project. This chapter extends the details of elements generally presented on chapter 2.

Chapter 4 demonstrates the experiments done during this research. Results on these experiments are shown to prove that the main objectives of this thesis were fulfilled.

Finally, Chapter 5 shows the conclusions of this work. Evaluation of the elements developed is presented as well as possible improvements and implementations for future work.

A variety of appendices are also provided to describe auxiliary elements mentioned along this document.

# Chapter 2

# Methodology

The methodology followed is shown in figure 2-1, it is similar to the one presented by Gonzalez in [Gonzalez, 2004]. Some of the steps involved were done separately for the implementation of the CAN network and for the speed controller. This entire research was done with aid from members of the E-Robots research group from the Robotics Laboratory at Tecnologico de Monterrey.

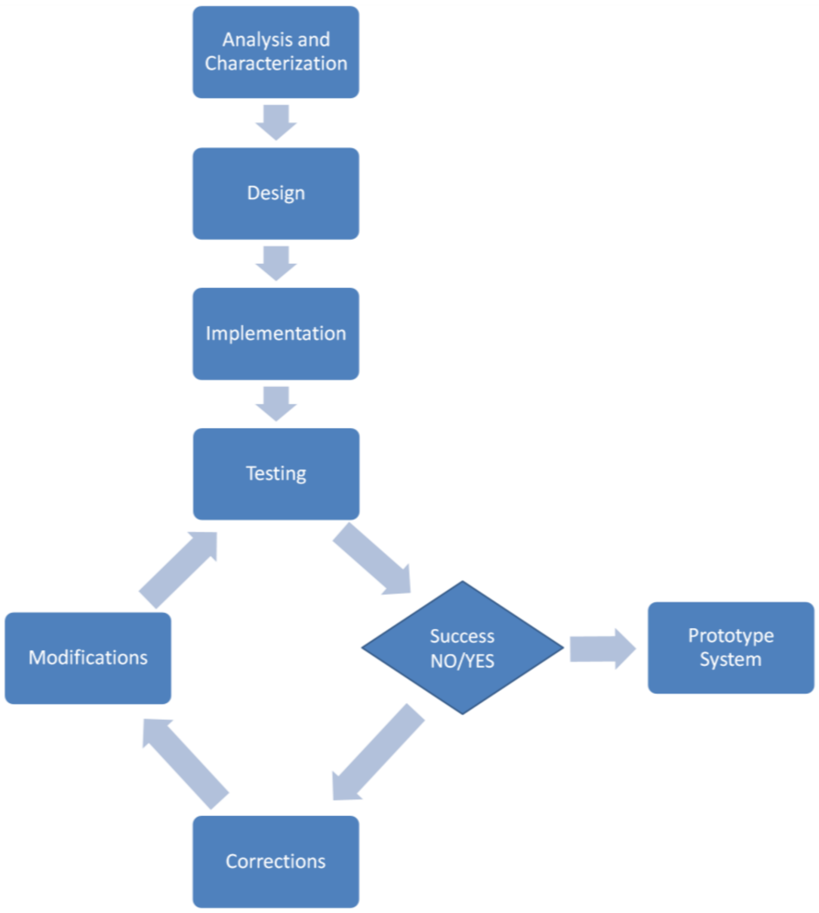


Figure 2-1: Research development methodology.

**Analysis and Characterization:**

A breakdown of the mechanical and electric systems composing the vehicle is made. Documentation provided by different manufacturers is thoroughly examined to fully understand the elements involved and their limitations. Characterization of the vehicle is done; different models are constructed, for instance the electric model, consisting of the schematic diagram of the electric wiring configuration; and the mechanical model, including the kinematic model of the vehicle and the dynamical model of the motor driving the vehicle.

In a similar manner, research is done concerning network controlled systems in order to understand their structure and characteristics. The CAN protocol is revised to understand its performance, restrictions and standard specifications.

**Design:**

Hardware and software solutions for new systems or modifications to existing ones are designed during this phase. At this point, selection of the necessary elements to implement a network based on the CAN protocol for closed loop speed control takes place.

Simple experimental testing is done on these designs in order to avoid malfunctioning during the implementation phase.

**Implementation:**

During the implementation phase, all designs conceived and tested on the previous stage are developed. Manufacturing of the necessary components for automation, modifications required on existing elements, as well as software programming is done. The complete NCS is assembled. After this phase the first prototype emerges and is ready for testing.

**Testing:**

The goal of this stage is to measure the performance obtained by the designed implementation and improve it through several corrections on the system.

Tests are performed in two manners; first, laboratory tests conducted in a controlled environment, where elements such as the CAN network and the speed control may be evaluated independently. Second, field testing, where the interaction of all the elements constituting the control system is evaluated as a whole, acting on the vehicle, providing it with certain degree of autonomy.

**Corrections and Modifications:**

Corrections and modifications help with the improvement of the control system; depending on the magnitude of such actions, the methodology can be reset to the design, implementation or testing phases. Once all tests prove successful, a prototype vehicle with autonomous speed control emerges.

## 2.1 Analysis and Characterization

### 2.1.1 Utilitarian Electric Vehicle

The vehicle used for this research is characterized with non-holonomic constraints due to its Ackerman steering mechanism; this means that rolling exists without lateral slipping between the wheels and the ground [Habumuremyi, 2005]. Non-holonomic constraints restrain the vehicle from moving instantaneously in any direction; they involve a higher number of effective degrees of freedom (DOF) than controllable ones. In order to change orientation, the wheels need to change direction and the vehicle has to move either forward or backward. The maneuverability of the vehicle is described by the degree of mobility ($\delta_m$), which deals with the degrees of freedom that can be manipulated for the vehicles motion; and the degree of steeribility ($\delta_s$), which deals with the number of centered orientable wheels that can be steered independently to steer the vehicle. The degree of maneuverability ($\delta_M$), is the overall degrees of freedom that the vehicle can effectively manipulate, which is simply the sum of the mobility and steeribility measures [Xiao, 2008; Campion et al., 1996]. For the case of an Ackerman steered vehicle the degree of mobility is 1 and the degree of steeribility is also 1 (making the degree of maneuverability, $\delta_M = \delta_m + \delta_s = 2$), this is why car-like vehicles are said to have 2 effective DOF.

The utilitarian vehicle employed is the Super Truck model from Johnson Industries, originally designed to work on mining environments. It is shown in figure 2-2.

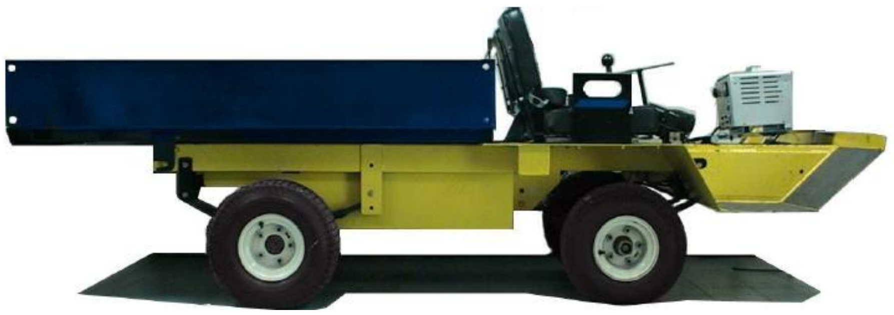


Figure 2-2: The Johnson Industries Super Truck testing platform.

Some characteristics of the Super Truck include:

- 1600 lb payload.
- 17 mph maximum speed (14 mph loaded).
- 5Hp motor.
- 11 gauge steel body.
- 600 ampere Solid State Controller.
- Six deep cycle battery, 36-volt system.

The vehicle follows a mid-engine rear-wheel drive layout; with a differential on the motors output shaft for torque transmission between the rear driving wheels, as shown in figure 2-3.

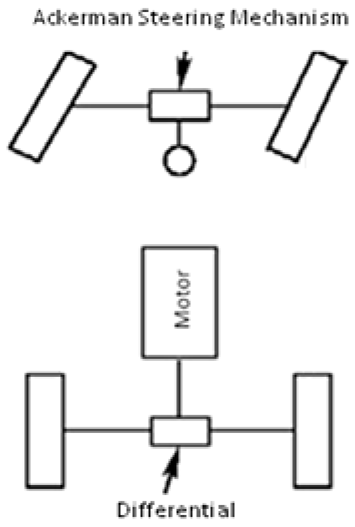


Figure 2-3: General layout of the vehicles mechanical composition. [Habumuremyi, 2005]

## 2.1.2 Ackerman Steering Mechanism

The absence of lateral slipping makes Ackerman steering suitable for low speed vehicles. Wheels on the inside and outside of a turn move at different angles, generating an instantaneous center of rotation (ICR) that travels according to the vehicles position and orientation. The projection of the steering arms for each orientable wheel creates a line that intersects at the middle of the back axis of the vehicle; a car-like vehicle with this geometry reduces its model complexity to that of a bicycle, simplifying the mathematical formulation.

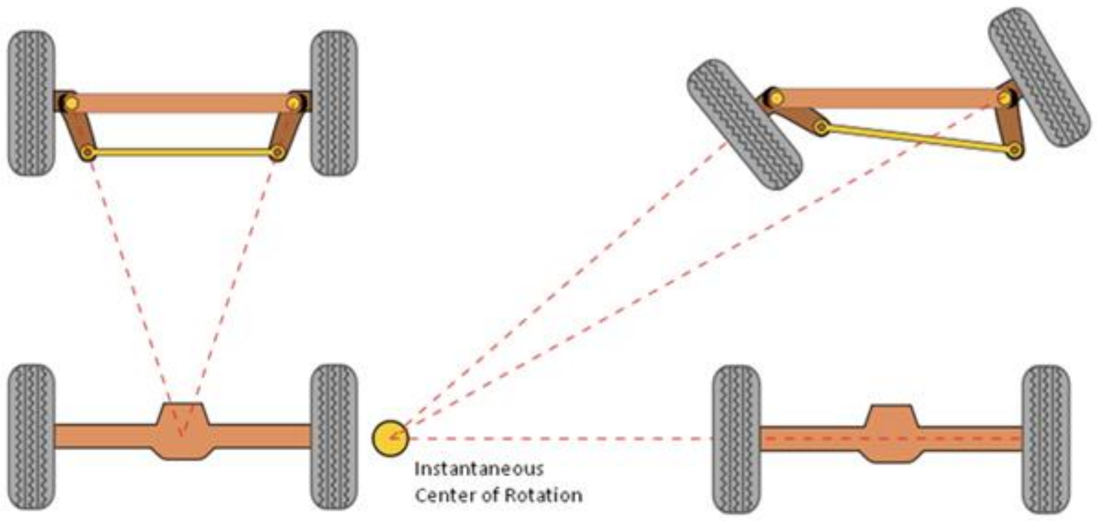


Figure 2-4: The ICR created by the Ackerman mechanism.

### 2.1.3 Kinematic Model

For an $\{x, y\}$ base frame, the mobile platform system can be defined as $\{x_m, y_m\}$ with its origin at the middle of the vehicle's rear axis. The vehicles orientation $\{\theta\}$ is measured with respect to $\{x\}$. Position and orientation may be described at any time by: $[x \quad y \quad \theta]^T$. A simplified bicycle model assumes one mobile centered wheel with an angle $\{\phi\}$ measured with respect to $\{\theta\}$, that is typically taken as the average between the orientation angles of each of the two steering wheels.

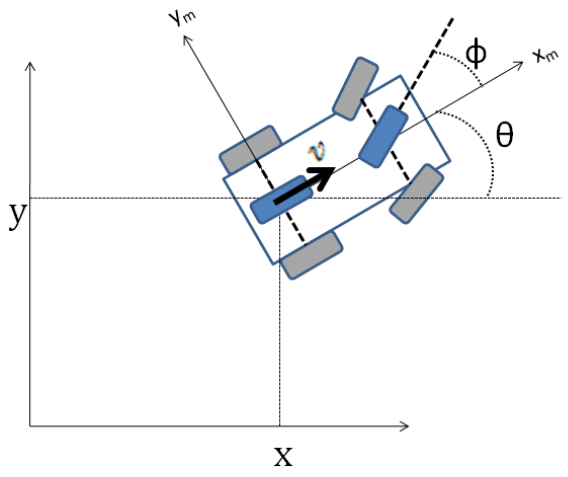


Figure 2-5: Vehicles posture definition.

The kinematic model uses the vehicle's velocity for position estimation, as it can be seen from the model's matrix form. The velocity control module is a critical aspect of the control architecture, since velocity affects proportionally the kinematic model, thus affecting position estimation. Detail on the kinematic model is presented in Appendix A.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi}_2 \end{bmatrix} = \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \\ \frac{\tan \phi_2}{L} \\ 0 \end{bmatrix} V_1 \tag{2.1}$$

### 2.1.4 Motor Model

The vehicle has a 5HP, 36 volt separately excited direct current electric motor as the main driving mechanism. An electric motor takes energy as an input and produces torque (or speed, depending on its operating curve) as an output. Speed and torque depend on the voltage applied on the motor and the current it draws. In a separately excited DC motor, the field current is supplied by a constant voltage supply; the field winding excites the field flux while the rotor's brushes and commutator supply the armature current. When a separately excited motor is excited by a field current and an armature current is flowing in the circuit, a back electromagnetic field ($E_g$) and a torque ($T_d$) are developed at a particular speed [Salam, 2003]. Figure 2-6 shows the separately excited motor's equivalent circuit.

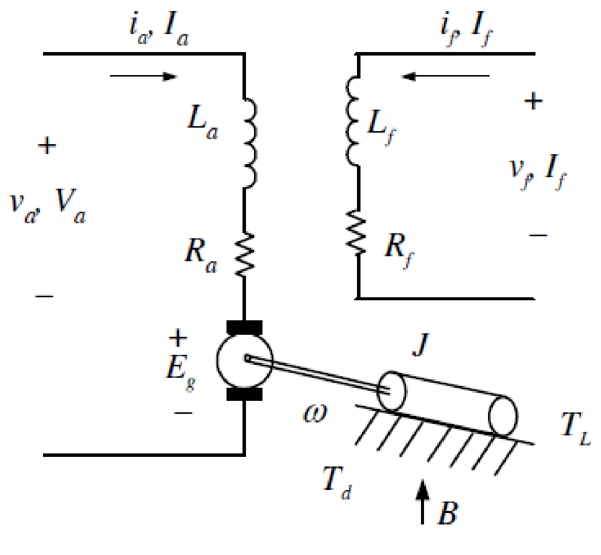


Figure 2-6: Equivalent circuit for the separately excited DC motor. [Salam, 2003]

Detail on the dynamical model for obtaining the motor's transfer function is presented in Appendix B. The transfer function for the electric motor is:

$$\frac{\omega(s)}{V_a(s)} = \frac{K}{T_m s + 1} \tag{2.2}$$

The motor model describes a first order system. A proportional-integral-derivative controller is suitable to control this type of systems without much complexity; hence the selection of such control algorithm for the physical system is theoretically validated by the model.

### 2.1.5 Electric Model

To understand how the electric vehicle works, the electric diagram representing its circuitry was constructed. This electrical model shows the connections between the different electric elements inside the vehicle.

Special interest is taken on the power controller installed on the vehicle. The Millipak controller manufactured by Sevcon, functions as a power interface between the control lines and the motor. A general representation of the main elements in the circuit is shown in figure 2-7.

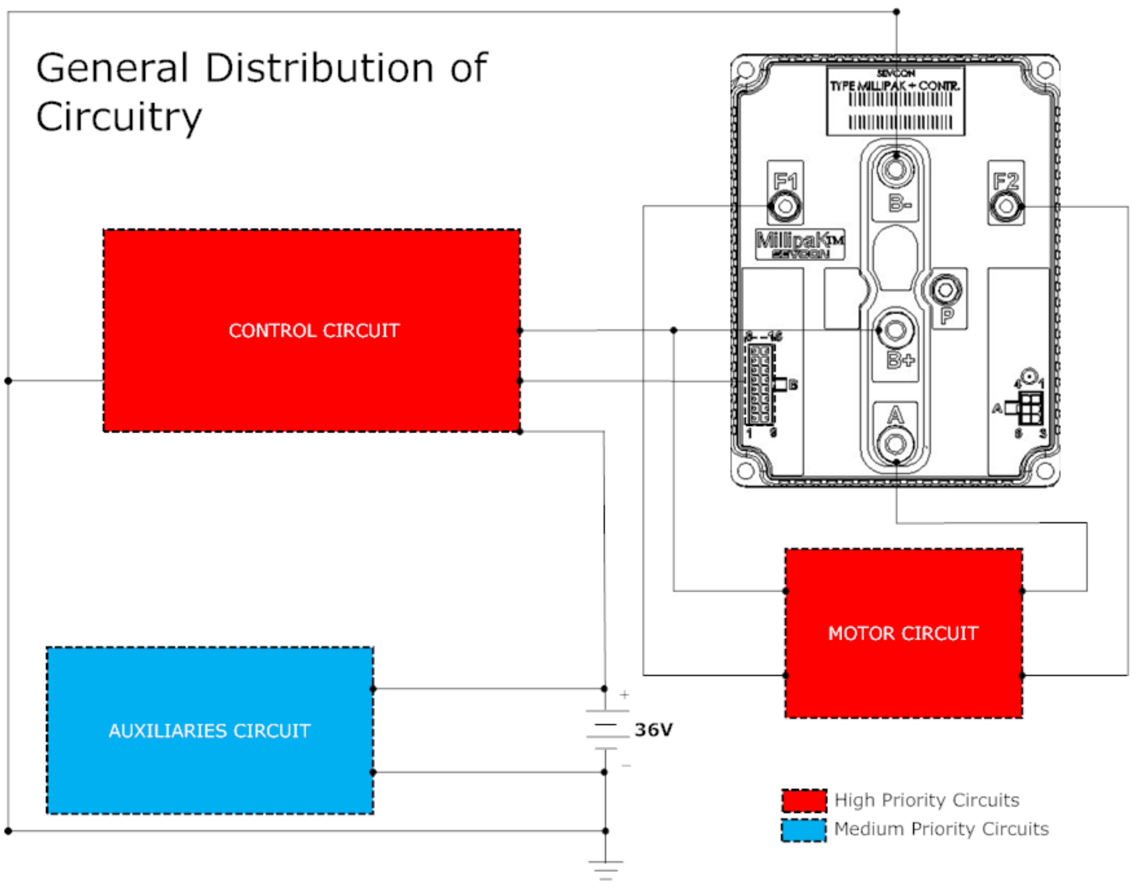


Figure 2-7: General representation of the main electric circuits of the vehicle.

The vehicle's electric circuit can be classified in three distinctive parts; the main or control circuit, the auxiliaries' circuit and the motor's circuit.

The control circuit occupies the major part of the vehicle's wiring. The accelerator signal, fail safe switch in the accelerator pedal and functions located on the manual control panel like the on/off key switching of the system, lights operation, displacement direction selection (forward/reverse) and emergency stop are part of the control circuit.

The auxiliaries' circuit connects secondary elements that are not necessary to drive the vehicle; these include the horn, the battery charge meter and the control mechanism in charge of lowering and raising the vehicle's cargo bed. Devices on this circuit are not interrupted when the emergency stop is pressed.

The connections between the 5HP driving motor and the Millipak power controller are fairly simple. The Millipak controller internally regulates the current and voltage applied to the motor's terminals according to control signals sent through connector B (shown in figure 2-8).

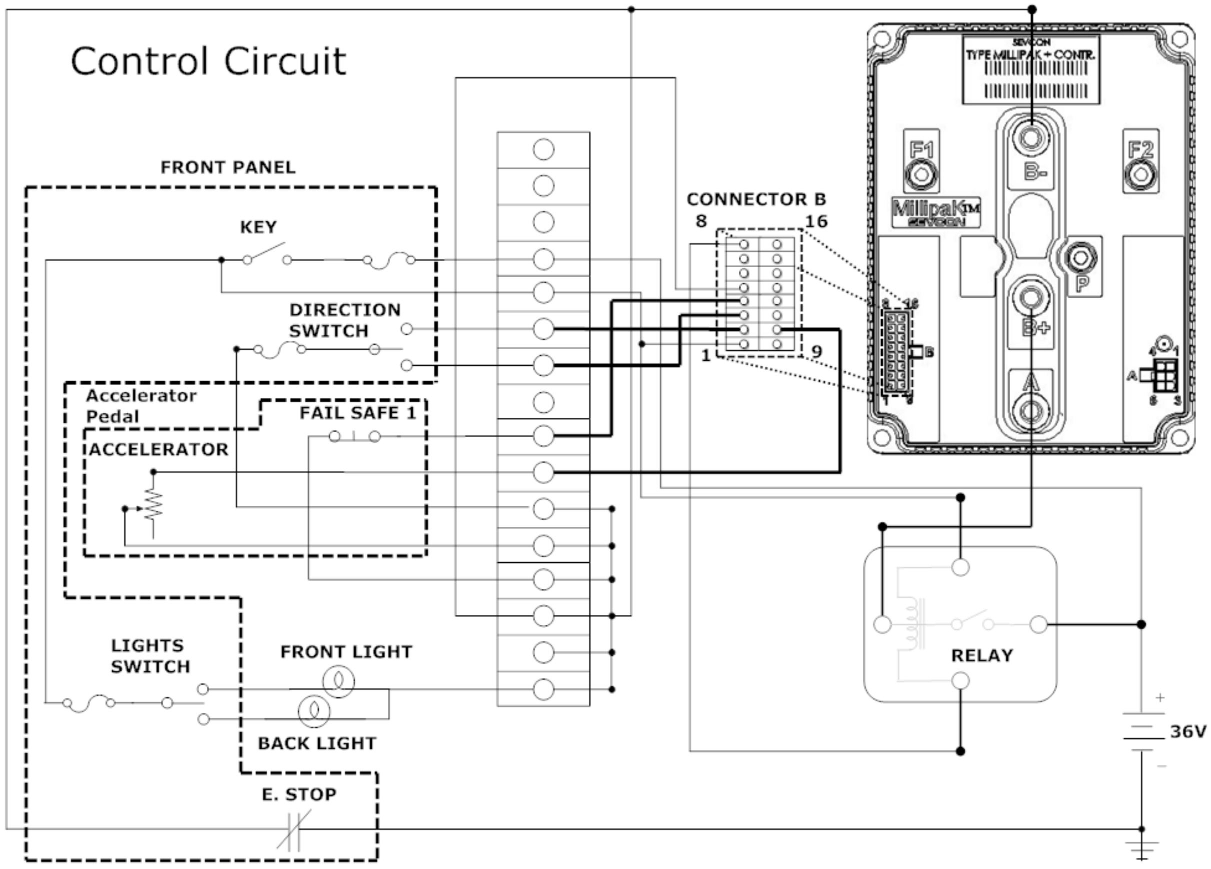


Figure 2-8: Control circuit. Elements contained in the front control panel and accelerator pedal are shown inside the dashed lines. The relay connects the positive terminal of the Millipak controller to the batteries once the key has been switched to the ON position.

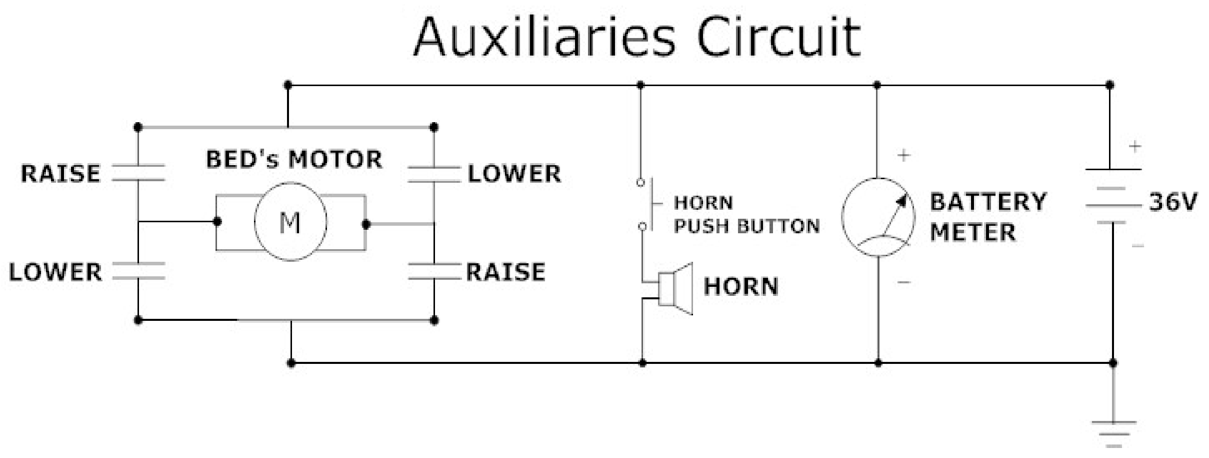
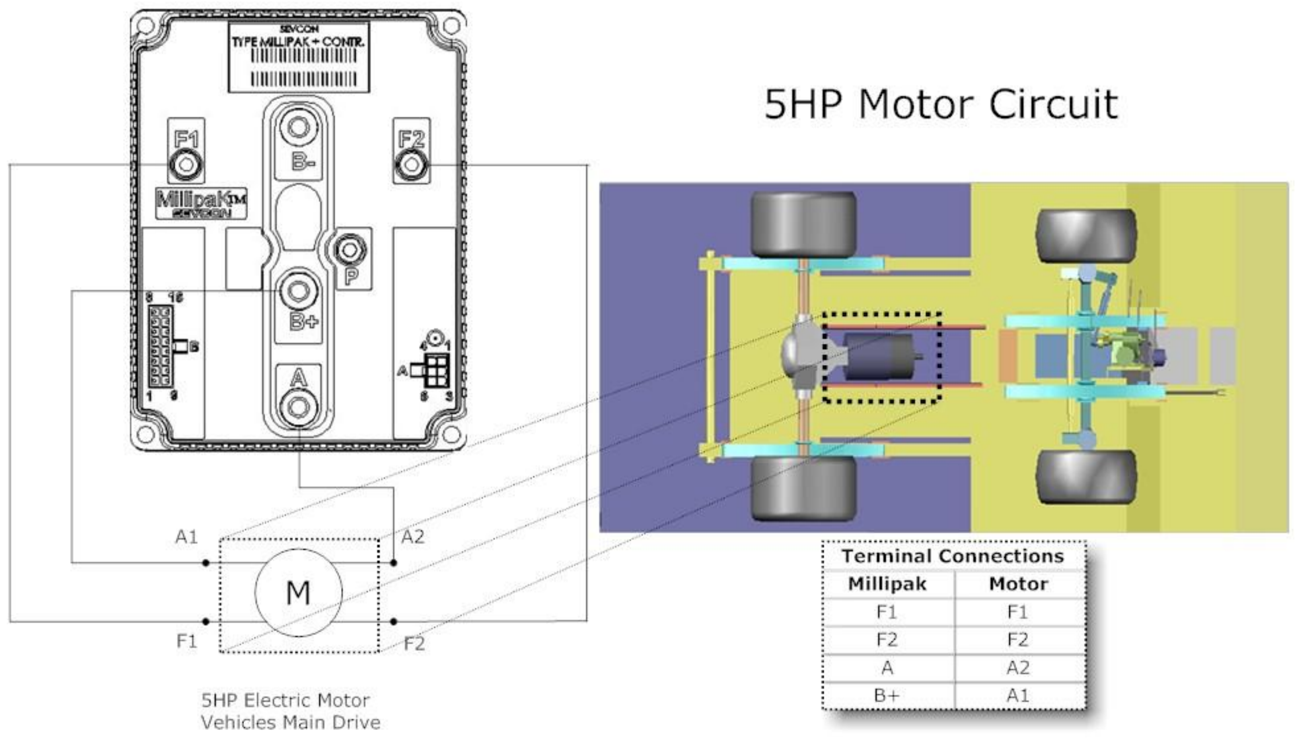


Figure 2-9: Auxiliaries circuit. Elements in the auxiliaries' circuit are not required to drive the vehicle. The horn, battery meter and the motor for the vehicle's cargo bed are independent of the elements in the other circuits.



| Terminal Connections | |
|---|---|
| Millipak | Motor |
| F1 | F1 |
| F2 | F2 |
| A | A2 |
| B+ | A1 |

Figure 2-10: Electric motor circuit. The batteries positive and negative terminals are connected to the respective B terminals of the Millipak controller. Within the figure are shown the connections between Millipak and motor terminals.

### 2.1.6 Millipak Controller

The Millipak controller from Sevcon regulates the power administered to the separately exited electric motor (SEM) on board the Super Truck. All the information presented in this section was taken from [Sevcon, 2004] and through contact with technical assistants at Sevcon. Some of the characteristics of the controller are shown in table 2-1:

Table 2-1: Standard specifications for the Millipak electric controller.

| Characteristic | Rating |
|---|---|
| Battery voltage | 24/48 VDC |
| Peak current | 500 A / 600 A |
| Continuous current | 180 A /200 A |
| Field current | 40 A |
| Max. power | 6.5 kW |
| Operating temperature | -30 to + 40ºC |
| Storage temperature | -40 to + 70ºC |
| Ingress of dust and water | IP66 |
| Humidity | 95% (at 60ºC) |
| Vibration | 6G |
| Switching frequency | 16 kHz |

Two types of connectors are located on the Millipak. First, connector A, for diagnosis and programming via a proprietary tool from Sevcon, denominated "calibrator". Second, connector B, where control signals are received in order to drive the electric motor. Refer to Appendix C for details.

The digital inputs, analogue inputs and contactor drive outputs available on connector B can be configured in a number of ways to suit various applications. Different presets are preprogrammed by Sevcon. To operate the controller one of these presets has to be selected. Once a preset has been selected the pins on connector B are allocated according to the preset's definition and a list of parameters or personalities defining certain characteristics on the controller become available for user modification.

Preset number 7 was used to configure the controller during the development of this research, because its description fits the operating purpose of the vehicle. The value of this preset allocates the different pins on connector B as shown in table 2-3:

Table 2-2: Preset 7 description.

| Digital Function | Description |
|---|---|
| 7 | Ride On vehicle with Speed Cutback 1 and 2 switches and external LED drive. |

Table 2-3: I/O configuration for the Millipak.

| Digital Function | 7 |
|---|---|
| Forward | B2 |
| Reverse | B3 |
| FS1 | B4 |
| Seat | B5 |
| Speed Cutback 1 | B6 |
| Speed Cutback 2 | *Bw* |
| Line Contactor | B8 |
| External LED | B9 |

*Bw* refers to pin B11, which can be used as a digital input if analog input 2 is configured as a digital input also. Analog functions are presented in table 2-4; analog input configuration 3 was used, leaving B10 as the accelerator input and B11 as a digital input.

Table 2-4: Analog functions for the Millipak

| Analog Function | 3 |
|---|---|
| Accelerator | B10 |
| Digital | B11 |
| Footbrake | |

The calibrator tool was used for the system's in-depth analysis and personal configuration. Refer to Appendix C for details on the Millipak configuration information.

### 2.1.7 Network Controlled System

Network Controlled Systems (NCS) are a type of distributed control system in which control loops are closed by means of an information network. The defining feature of an NCS is that information is exchanged among components using the network. This type of setup offers several advantages over traditional point-to-point control solutions, some which are: modularity and flexibility for system design; simple and fast implementation; ease of system diagnosis and maintenance; decentralized control and increased system agility. These advantages are translated into characteristics such as distributed processing, interoperability, reduced wiring, small power requirements and complexity of the physical connections, as well as the possibility of information exchange between different control loops; all which ultimately represent a reduction in costs [Huo et al., 2004; Chow & Tipsuwan, 2001]. In a NCS, sensors and actuators are connected directly to the desired plant and also to the real-time network. The controller, while physically separated from the plant, is also connected to the real-time network, closing the control loop [Lihua et al., 2008]. Proper interconnection of every device on the network depends on the protocol being used. A general representation of the elements included in a NCS is shown in figure 2-11.

Systems controlled by an NCS can be modeled as “discrete-continuous” with time varying elements induced by the network. These induced delays depend on different factors, like network type, cable lengths, and baud rates used. The medium access method affects time variations since every device (referred as a node) competes for broadcasting time on the network, where collisions are meant to happen. Collision handling is an important aspect that depends on the protocol used. Thus there exists a trade-off between the finite bus capacity and the performance of the control loop.

Induced time delays on an NCS can be summarized as the controller-actuator delay ($\tau_{ca}$) and the sensor-controller delay ($\tau_{sc}$), as well as the computational time required by each device performing control operations over the network [Lihua et al., 2008]. The NCS designer should account for synchronization issues; different configurations usually vary depending on whether the nodes activity is event-driven or time-driven.

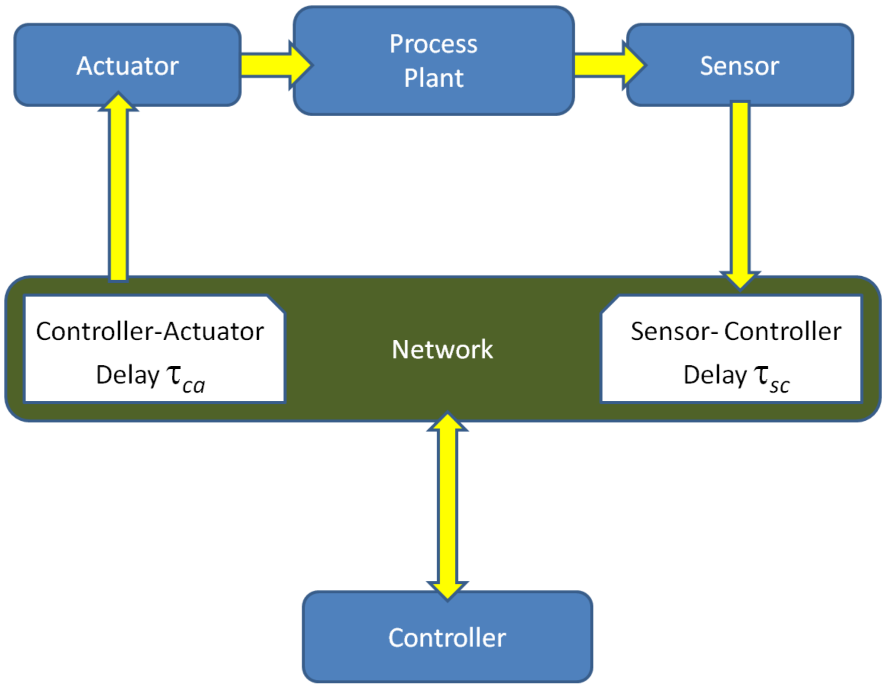

Actuator
Process Plant
Sensor
Controller-Actuator Delay $\tau_{ca}$
Network
Sensor- Controller Delay $\tau_{sc}$
Controller

Figure 2-11: General representation of a Network Control System with induced delays on the sensor and controller. While sensor and actuator are directly attached to the process plant the controller is physically separated but still connected to the other elements through the network.

### 2.1.8 Controller Area Network

Data management and communication systems play an important role within the vehicle's architecture. A reliable communication system improves reaction times and minimizes uncertainty. Controller Area Network (CAN) is a vehicle bus standard for distributed environments originally designed by Robert Bosch [Bosch, 1991] to provide robust serial communication for a vehicle's on board network. The CAN bus protocol offers standard network technology, with high error checking features. Some of the main features of CAN include the ability to use decentralized pre-processing by the inclusion of a microcontroller, which results in less computational work for higher level units. CAN is a broadcast serial bus, based on message exchange between different nodes.

Because of its flexibility it is easy to add and remove CAN nodes from a bus or alter the way a CAN node behaves without disrupting communication on the entire network. The ability to develop and test a few CAN nodes and then scale up to dozens is pragmatically convenient [NI, 2009a].

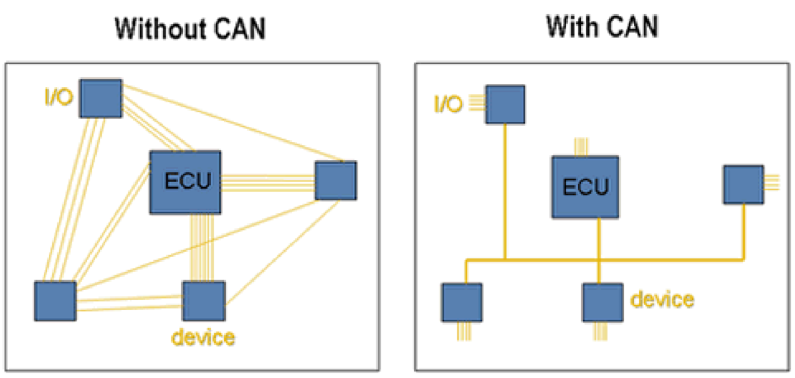


Figure 2-12: Devices, modules and electronic control units without a CAN bus implementation require considerably more wiring in a system (left) [NI, 2009a].

In modern day vehicles several different networks interact with CAN bus. Usually the CAN network controls all real-time critical functions, such as ABS and traction control. Local Interconnect Network (LIN) is used as a sub-bus of CAN, typically for non-safety related critical tasks. The Media Oriented System Transport (MOST), also acting as a sub-bus of CAN, is used to control automotive multimedia. All networks within the vehicle are controlled by a main control system, which also acts as a gateway for all networks to share data [Leen & Heffernan, 2002]. Figure 2-13 shows vehicle network integration, where CAN functions as the main or top network and two different sub-networks operate on less critical functions.

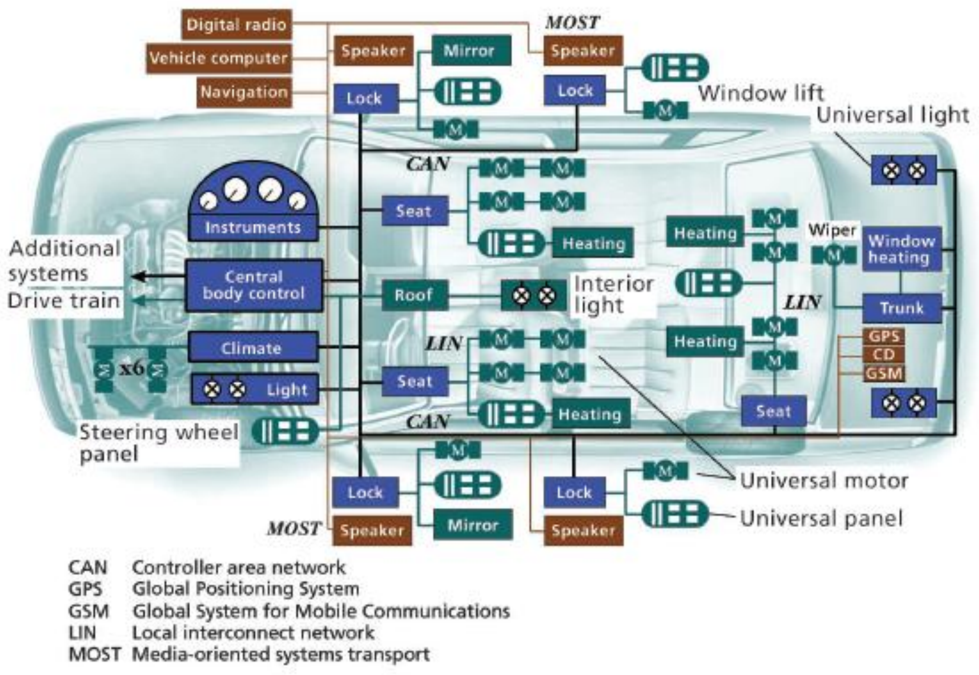


Figure 2-13: Different networks inside a common vehicle [Leen & Heffernan, 2002].

CAN bus technology has made its inclusion not only on automobiles but also in other means of transportation, like motorcycles. BMW uses CAN bus for communication between proprietary control modules on its R1200 and K1200 models. These control modules share a data network through the CAN bus. An example is the interconnection between the Engine management system, called BMS-K (BMW Motor-Steuerung mit Klopfregelung); the Central Chassis Electronics, called ZFE (Zentrale Fahrzeugelektronik) and the Instrument Panel (I-Cluster). Additional modules like the Anti-lock braking system called ABS, and the Alarm System called DWA (Diebstahl Warnanlage) also interact through the CAN bus. The ZFE controls the lights, heated accessories, horn, radio, accessory socket, and cruise control based on inputs from handlebar switches. Control inputs go directly to ZFE, and control outputs go from ZFE to individual components. The BMS-K inputs include the starter button, kill switch, sidestand switch, clutch switch, gear position indicator, and various engine sensors. Outputs control the starter, fuel pump, injectors, ignition coils, and warning lights. Finally, the I-Cluster displays information to the rider; it has only one physical input, the clock setting button, while the outputs are the various instruments and warning lights [Largiader, 2006]. Information is broadcasted to the network by each module, but only modules that require it will act upon it. The I-Cluster is a good example of a CAN control module, because it gets all of its information from other control modules through the CAN bus rather than receiving it directly as an input from sensors and switches. Because of this the I-Cluster panel can be removed and replaced without interrupting any other control module on the network. Figure 2-14 shows a schematic of the CAN network shared by the control modules on a BMW motorcycle.

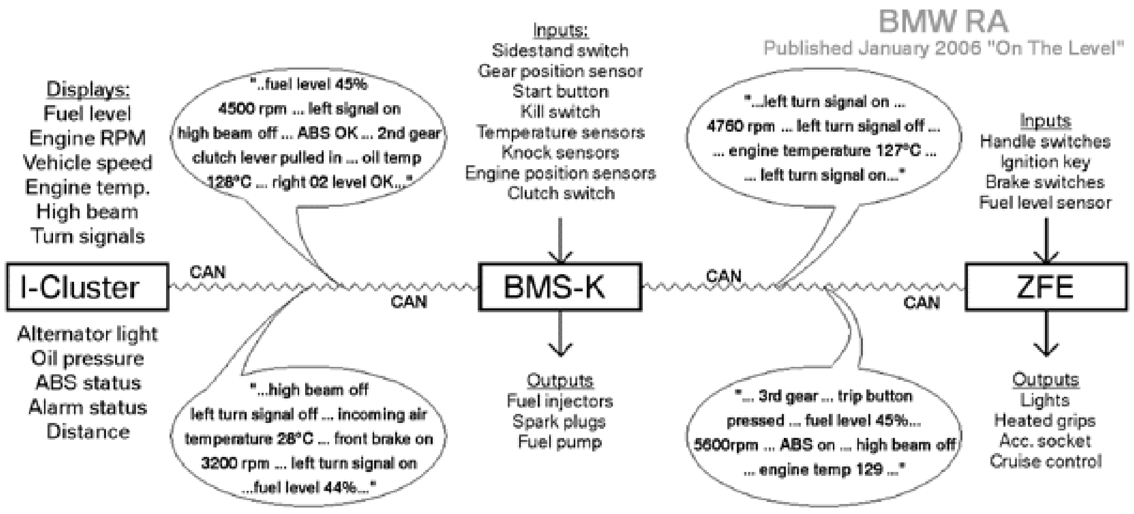

BMW RA
Published January 2006 "On The Level"
Displays: Fuel level, Engine RPM, Vehicle speed, Engine temp., High beam, Turn signals
I-Cluster
Alternator light, Oil pressure, ABS status, Alarm status, Distance
"..fuel level 45% 4500 rpm ... left signal on high beam off ... ABS OK ... 2nd gear clutch lever pulled in ... oil temp 128°C ... right 02 level OK..."
"...high beam off left turn signal off ... incoming air temperature 28°C ... front brake on 3200 rpm ... left turn signal on ...fuel level 44%..."
Inputs: Sidestand switch, Gear position sensor, Start button, Kill switch, Temperature sensors, Knock sensors, Engine position sensors, Clutch switch
BMS-K
Outputs: Fuel injectors, Spark plugs, Fuel pump
"...left turn signal on ... 4760 rpm ... left turn signal off ... ... engine temperature 127°C ... ... left turn signal on..."
"... 3rd gear ... trip button pressed ... fuel level 45%... 5600rpm ... ABS on ... high beam off ... engine temp 129 ..."
Inputs: Handle switches, Ignition key, Brake switches, Fuel level sensor
ZFE
Outputs: Lights, Heated grips, Acc. socket, Cruise control
CAN

Figure 2-14: CAN bus representation in BMWs R1200 and K1200 motorcycle models [Largiader, 2006].

Besides its application on vehicles, the CAN protocol has lately penetrated the field of robotics. CAN bus is popular as a real-time communication network for devices within humanoids. For example, the Rh-1, a 1.3m height 21 degrees of freedom humanoid robot uses CAN bus for communication between some of its electronic components [UC3M]. Toni, another humanoid that was constructed by the University of Freiburg [Behnke, 2006] to participate in the robotic soccer challenge RoboCup; uses CAN bus to establish communication between all of its onboard microcontrollers. Domo, a humanoid intended for research in manipulation by incorporating force sensing modules in its joints, uses CAN bus as a communication channel between different digital signal processing microcontrollers located all over its body and the higher level computational system [Edsinger & Weber, 2004].

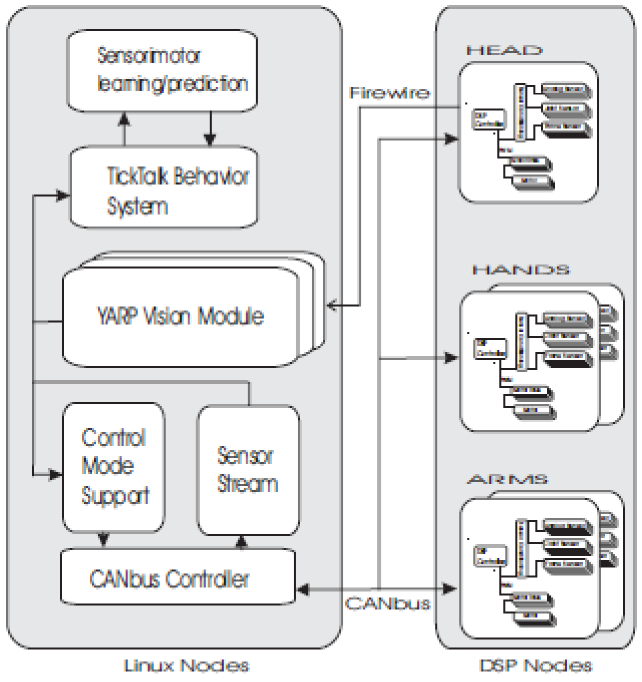


Figure 2-15: Domo's CAN bus node architecture [Edsinger & Weber, 2004].

*CAN Bus Protocol Overview*

CAN is able to provide real-time and fault tolerant features which make it suitable for control applications. Baud rate varies with bus length, where the highest baud rate can be achieved with a maximum length of 40 meters over a twisted pair wire, according to ISO standard specifications. The relationship between transfer rate and bus length is depicted in figure 2-16.

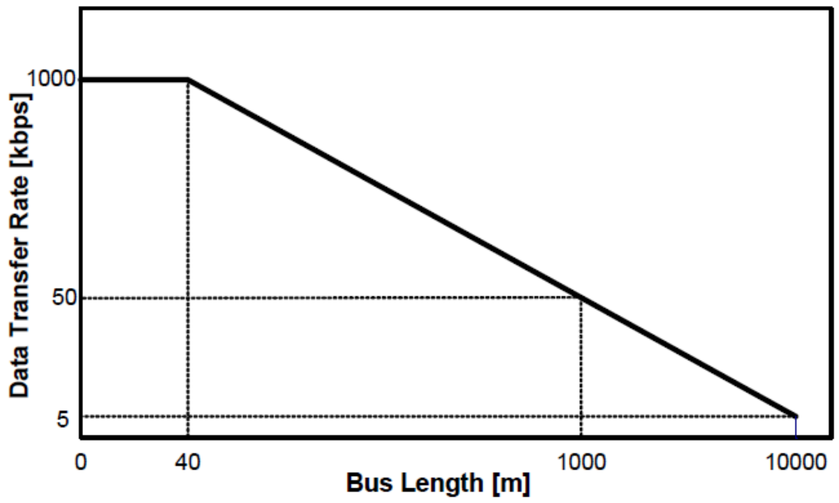


Figure 2-16: Transfer rate versus bus length for CAN bus over a twisted pair wire medium [Azzeh, 2005].

CAN bus is based on the Open System Interconnection (OSI) model created by the International Standards Organization (ISO), meaning it follows a layered approach, which enables interoperability with products from different manufacturers. The CAN protocol implements the two lower layers of the OSI model; data link and physical layers [Pazul, 1999] described next.

*Physical Layer*

The Physical Layer is responsible for interconnection between nodes in the network; it handles the transmission of electrical impulses across the communication medium and deals with timing, encoding and synchronization of the soon to be transferred bit stream.

A bit signal on the bus line can take two possible representations: recessive, which only appears on the bus when the nodes send recessive bits; and dominant, sent only by one node to be heard on the bus. This means that a dominant bit sent by one node can overwrite recessive bits sent by other nodes. This feature is used for bus arbitration.

A differentially driven pair of wires identified as CAN High and CAN Low provide reliable signal transmission despite low power levels; they are usually terminated with 120-ohm resistors to eliminate signal reflections at the end of the bus line and ensure correct voltage levels.

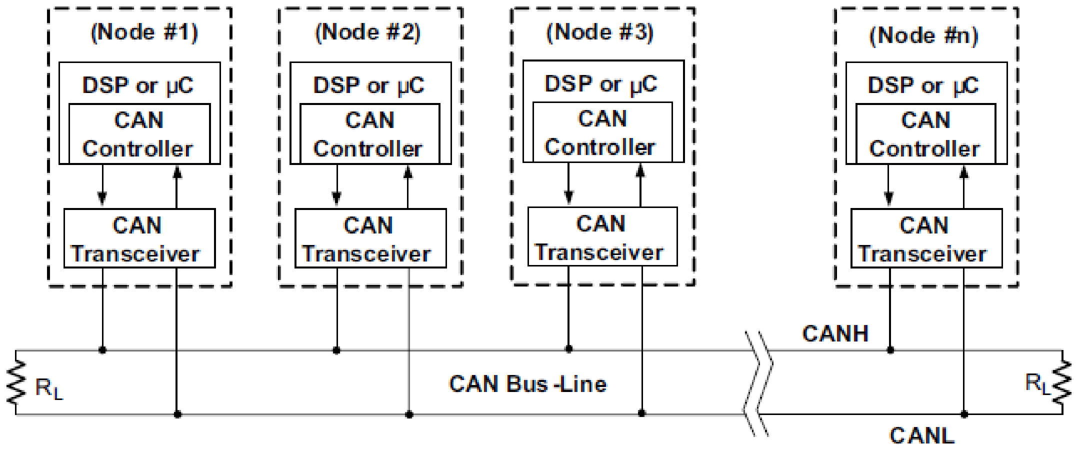


Figure 2-17: Transmission lines CAN High and CAN Low with bus termination resistors [Corrigan, 2008].

For the two-wire bus, the recessive bus state occurs when the CAN Low and CAN High lines are at the same potential (2.5V), and the dominant bus state occurs when there is a voltage difference of ± 1V (CAN L = 1.5V and CAN H = 3.5V). The CAN bus remains in the recessive state while it is idle [Richards, 2005]. Differentially driven signals provide an advantage when dealing with voltage spikes; if a spike is encountered both line conductors are equally affected, maintaining the voltage differential between both wires, providing a certain degree of noise immunity [Azzeh, 2005].

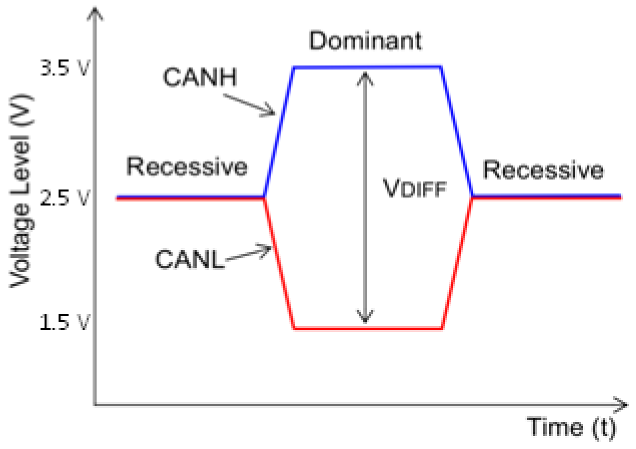


Figure 2-18: Logical levels over CAN bus [Richards, 2005].

### *Data Link Layer*

The Data Link Layer is in charge of building frames that encapsulate the data to be transferred. Each data frame is assigned a unique identifier, which is used to determine bus access and detect errors. Since every frame is unique there is no need for addressing information.

This layer also performs Medium Access Control (MAC) to prevent conflicts when two different nodes try to access the network at the same time. It performs the data encapsulation/decapsulation, error detection and control, bit stuffing/destuffing and the serialization and serialization functions. The use of a MAC method is called bus arbitration. As a consequence of arbitration no bandwidth is wasted; there are methods that a designer can use to predict the longest possible delay before any message is delivered.

The CAN communication protocol uses a Carrier Sense Multiple Access/Collision Detection (CSMA/CD) MAC method. CSMA means that every node on the network must monitor the bus, and only transmit messages when no activity is found. During periods of no activity, every node has an equal opportunity to transmit a message. The Collision Detection portion means that if two nodes start transmitting at the same time, a collision will be detected and the proper corrective action will be taken [Pazul, 1999].

A nondestructive bitwise arbitration method is utilized; with this, even if collisions are detected, messages remain intact after arbitration is completed. Higher priority messages suffer no corruption or delay during the arbitration process. Non-destructive bitwise arbitration makes use of dominant and recessive logic states. The CAN protocol defines a logic bit 0 as dominant and a logic bit 1 as a recessive bit. As the name suggests, a dominant bit state will always win arbitration over a recessive bit state. Since the field used in the message arbitration process is the Message Identifier (ID), the lower its value the higher the priority of the message. A node also monitors the state of the bus while transmitting, to check if the logic state it is trying to send appears on the bus, if the node is transmitting a recessive bit and detects a dominant bit on the bus, it automatically stops transmission and receives the dominant message. After the dominant message has been received and the bus is once again idle, the node is able to try retransmission of its own message [Pazul, 1999].

The CAN protocol is message based, not address based, meaning that messages are not transmitted from one address to another; instead, the message is broadcasted throughout the bus, making it accessible to all nodes. It is up to every node to decide if the message is useful or not. Nodes decide whether to process or discard the information received based on the message identifier (ID) [Pazul, 1999]. Nodes inside the CAN network also have the ability to request for information, by generating a Remote Transmit Request (RTR).

Two standards have been developed by ISO, specifying a 5V differential voltage over the physical layer interface. These are [Pazul, 1999]:

- ISO-11898: CAN High Speed with bit rates between 125kbps and 1Mbps.
- ISO-11519: CAN Low Speed with bit rates up to 125kbps.

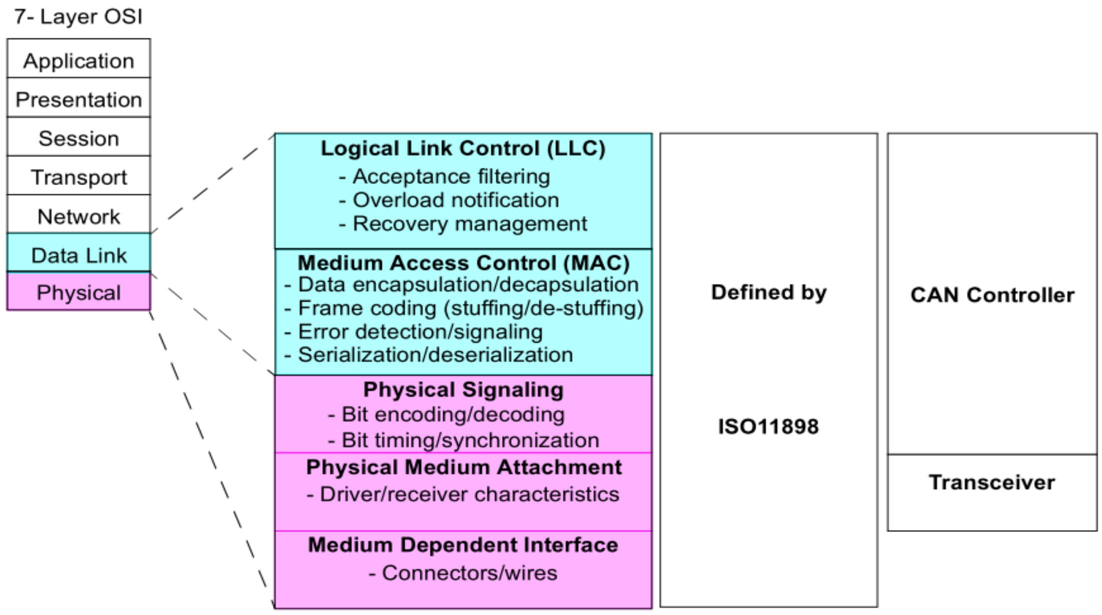

7- Layer OSI
Application
Presentation
Session
Transport
Network
Data Link
Physical
Logical Link Control (LLC)
- Acceptance filtering
- Overload notification
- Recovery management
Medium Access Control (MAC)
- Data encapsulation/decapsulation
- Frame coding (stuffing/de-stuffing)
- Error detection/signaling
- Serialization/deserialization
Physical Signaling
- Bit encoding/decoding
- Bit timing/synchronization
Physical Medium Attachment
- Driver/receiver characteristics
Medium Dependent Interface
- Connectors/wires
Defined by
ISO11898
CAN Controller
Transceiver

Figure 2-19: OSI reference model [Richards, 2005].

*Bit Timing*

Bit rate is characterized as the number of bits transmitted over a certain period of time. The CAN protocol allows bit rate programming, as well as number and location of data samples in a bit period. Optimal selection of these parameters ensures message synchronization and proper error detection. To achieve synchronization all the nodes must have the same nominal bit time. Since no clock is encoded in the data stream, receiving nodes must recover the clock and synchronize it to the transmitter's clock [Microchip, 2007; Jöhnk & Dietmayer, 1997].

The nominal bit rate is the number of bits per second an ideal transmitter will transmit without resynchronization. The CAN nominal bit time is made of non-overlapping segments; each segment is made up of an integer number of the basic time units in a bit period, called time quantas (TQ), each time quanta is one period of the CAN system clock, as shown in figure 2-21. The desired baud rate is calculated by the arithmetical sum of the different segments. With the use of a baud rate prescaler (BRP) custom baud rates can be achieved. The special register called synchronization jump window (SWJ), defines the number of time quantas a segment can be lengthened or shortened for resynchronization. The sampling point defines at which part of the nominal bit time, the signal will be sampled to determine the logical state of the current bit.

Depending on the manufacturer of the hardware used, the segments composing the nominal bit time can be named differently. For example, for Microchip the nominal bit time is composed of 4 segments, as shown in figure 2-20, while for Philips the nominal bit time is composed of 3 segments as depicted on figure 2-21. It is important to refer to the specific manual of the hardware involved for programming details when setting up the network.

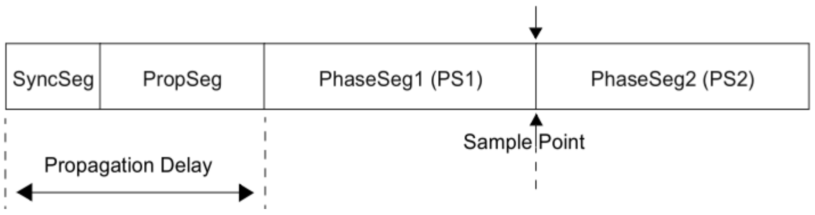


Figure 2-20: Nominal bit time segments as defined by Microchip [Microchip, 2007].

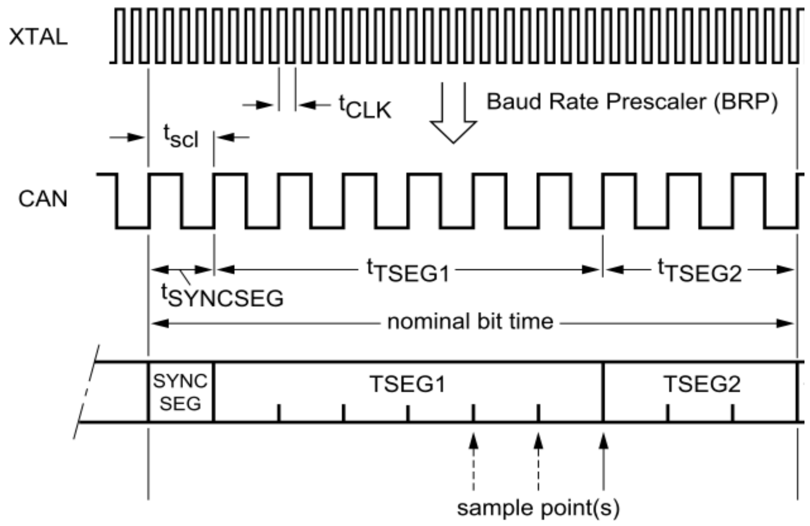


Figure 2-21: Nominal bit time segments defined by Philips [Jöhnk & Dietmayer, 1997].

*CAN Message Frame*

The CAN protocol defines four different types of frames: 1) Data frames, which carry information from one node to the other nodes; 2) Remote frames, which are data frames with the Remote Transmit Request (RTR) bit set, requesting information from other nodes; 3) Error frames, sent by nodes that detect any type of error; and 4) Overload frames, generated by nodes that require more processing time for messages already received.

The CAN protocol defines two data frame formats, the standard format, also known as CAN 2.0A appearing in figure 2-21; and the extended format shown in figure 2-22 known as CAN 2.0B. The difference between formats resides in the number of bits used for the identifier field. CAN 2.0A uses 11 bit identifiers, while CAN 2.0B uses 29 bit identifiers [Pazul, 1999; Azzeh, 2005].

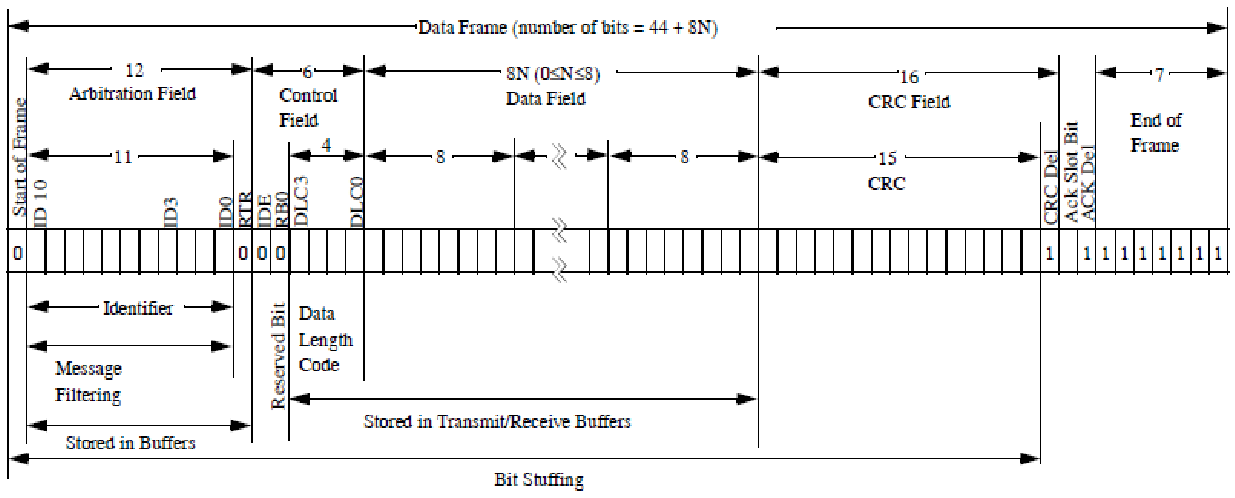


Figure 2-22: Standard message frame format for the CAN protocol [Pazul, 1999].

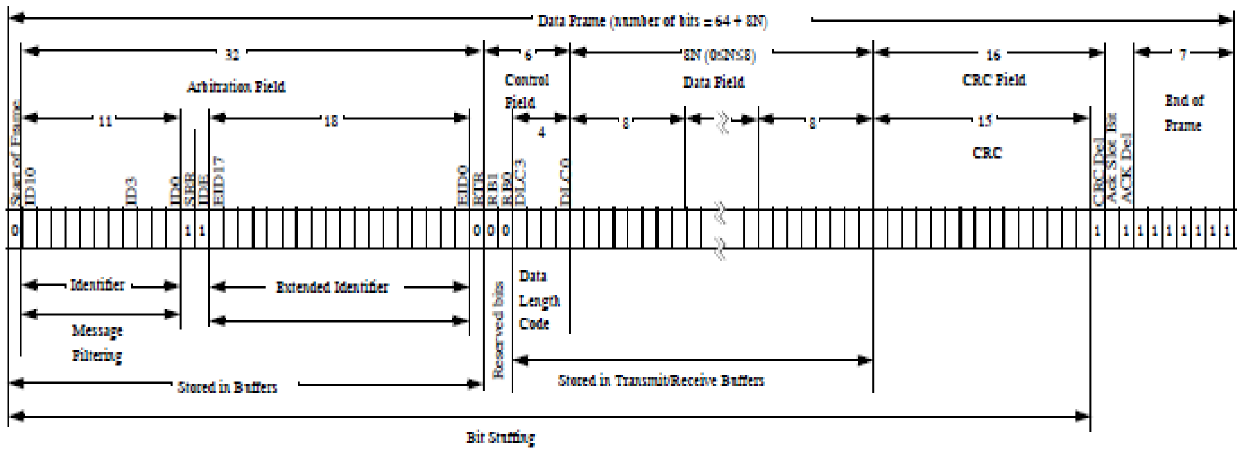


Figure 2-23: Extended message frame format for the CAN protocol [Pazul, 1999].

The fields for the CAN frame format are all described in [Bosch, 1991; Pazul, 1999; Azzeh, 2005] and are as follow:

- **Start of Frame (SOF) Field:** This is a dominant (logic 0) bit that indicates the beginning of a message frame.

- **Arbitration Field:** Containing an 11 bit message Identifier (ID) and the Remote Transmission Request (RTR) bit. A dominant RTR bit indicates that the message is a Data Frame. A recessive value indicates that the message is a Remote Transmission Request and therefore the frame is a Remote Frame. A Remote Frame is a request by one node for data from some other node on the bus. Remote Frames do not contain a Data Field. To accomplish an RTR, a Remote Frame is sent with the identifier of the required Data Frame.
- **Control Field:** Containing six bits; the Identifier Extension (IDE) bit distinguishes between the CAN 2.0A standard frame and the CAN 2.0B extended frame. Followed by the Reserved Bit Zero (RB0) bit, which is a reserved bit and is defined to be a dominant bit by the CAN protocol. The four bit Data Length Code (DLC) indicates the number of bytes in the "Data Field" that follows.
- **Data Field**: Containing from zero to eight bytes of data.
- **Cyclic Redundancy Check (CRC) Field**: Guarantees the frame's integrity; contains a 15 bit cyclic redundancy check code and a recessive delimiter bit.
- **Acknowledge (ACK) Field**: Consisting of two bits; the first is the Slot bit which is transmitted as a recessive bit and is overwritten as a dominant bit by those receivers which have successfully received the message, regardless of whether the node processes or discards the data. The second bit is a recessive delimiter bit.
- **End of Frame (EOF) Field**: Consisting of seven recessive bits.

Following the end of a frame, the Intermission Frame Space (IFS) appears; consisting of three recessive bits. After the three bit intermission period the bus becomes available, and other nodes may begin transmission of their messages.

A message transmitted with an extended frame format is almost the same as one with a standard frame format. By using the IDE bit in dominant form, frames are transmitted in standard format, if the IDE bit is recessive; the CAN extended frame format is used instead. Standard format frames have higher priority than extended frame formats, but by using an extended format higher number of different message identifiers are possible [Pazul, 1999].

### *CAN Error Handling*

CAN nodes have the ability to diagnose fault conditions and operate on different modes based on the severity of the problems encountered. Nodes can detect the difference between short disturbances and permanent failures, modifying their functionality accordingly. CAN nodes can go from normal operation, where they are able to transmit and receive messages, to complete shutdown (or bus-off state) based on the severity of the errors detected. This is called Fault Confinement. No faulty CAN node will be able to monopolize the network's bandwidth because faults will be confined and faulty nodes will shut off before bringing the network down. Correct bandwidth use for critical system information is guaranteed by Fault Confinement.

Five conditions that yield an Error Frame exist; these are [Pazul, 1999]:

- **CRC Error:** The Cyclic Redundancy Check failed (CRC) and an error occurs, an Error Frame is generated. Since at least one node did not properly receive the message, it is resent after a proper intermission time.
- **Acknowledge Error:** The transmitting node checks if the Acknowledge Slot, sent as a recessive bit, contains a dominant bit, with this it would acknowledge that at least one node received the message correctly. If this bit is recessive, then no node received the message properly. An Error Frame is generated and the original message is resent after a proper intermission time.
- **Form Error:** If any node detects a dominant bit in any of the following four segments of the message: End of Frame, Intermission Frame Space, Acknowledge Delimiter or CRC Delimiter; a Form Error is generated. The original message is resent after a proper intermission time.
- **Bit Error:** If a transmitter node sends a dominant bit and detects a recessive bit, or vice versa, when monitoring the actual bus level and comparing it to the bit that it has just sent. In the case where the transmitter sends a recessive bit and a dominant bit is detected during the Arbitration Field or Acknowledge Slot, no Bit Error is generated because normal arbitration or acknowledgment is occurring. If a Bit Error is detected, an Error Frame is generated and the original message is resent after a proper intermission time.
- **Stuff Error:** CAN protocol uses a Non-Return–to-Zero (NRZ) transmission method. This means that the bit level is placed on the bus for the entire bit time. CAN is also asynchronous, and bit stuffing is used to allow receiving nodes to synchronize by recovering clock information from the data stream. Receiving nodes synchronize on recessive to dominant transitions. If there are more than five bits of the same polarity in a row, the protocol dictates to stuff an opposite polarity bit in the data stream. The receiving node(s) will use it for synchronization, but will ignore the stuff bit for data purposes. If, between the Start of Frame and the CRC Delimiter, six consecutive bits with the same polarity are detected, then the bit stuffing rule has been violated. A Stuff Error occurs, an Error Frame is sent, and the message is retransmitted after the intermission time.

Once an error has been detected it is broadcasted via the Error Frame. The erroneous message is aborted and as soon as arbitration allows it, it is retransmitted. Each node is in one of the three possible error states generated by the above conditions, which are shown in figure 2-24 and described as [Pazul, 1999; Azzeh, 2005]:

- **Error-Active:** In this state, the CAN node can actively take part on the bus communication; it can send an Active Error Flag, which consists of six consecutive dominant bits. With this the bit stuffing rule is violated and causes all other nodes to send an Error Flag, called the Error Echo Flag. A node is Error-Active when both the Transmit Error Counter (TEC) and the Receive Error Counter (REC) are below 128. Error-Active is the normal operational mode, allowing the node to transmit and receive without restrictions.
- **Error-Passive:** A node becomes Error-Passive when either the Transmit Error Counter or Receive Error Counter exceeds 127. Error-Passive nodes transmit Passive Error Flags, which consist of six recessive bits. If this node is the only transmitter on the bus, the passive error flag will violate the bit stuffing rule and the receiving node(s) will respond with Error Flags of their own depending on their own error state. If the Error-Passive node in question is not the only transmitter or is a receiver, then the Passive Error Flag will have no effect on the bus due to the recessive nature of the error flag.
- **Bus-off:** A node goes into the Bus-Off state when the Transmit Error Counter is greater than 255. In this mode the node cannot send nor receive messages, achieving Fault Containment.

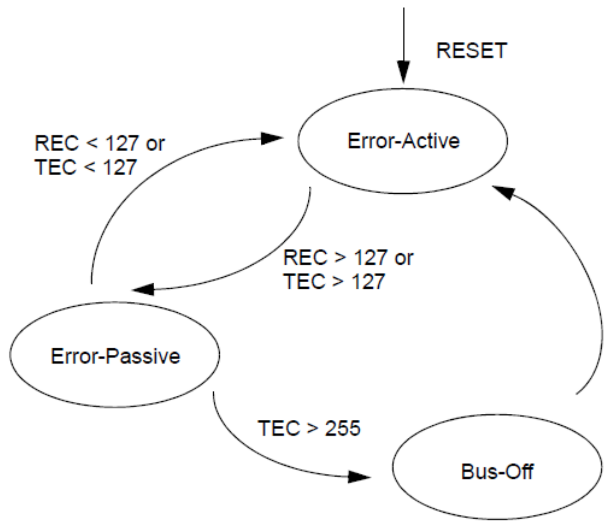


Figure 2-24: State diagram for different error states a CAN node can have [Microchip, 2007].

### 2.1.9 Programmable Automation Controller

A Programmable Automation Controller (PAC) is a compact controller that combines the capabilities of a PC-based control system and a Programmable Logic Controller (PLC) into an open flexible architecture.

PLC users that required highly advanced applications began evaluating PC based solutions. The PC offered a graphical rich programming and user environment, and utilized commercial components like floating point processors and high speed I/O busses, such as PCI and Ethernet. However PC based applications were not designed for rugged environments, system crashes and rebooting, non-industrially hardened components and unfamiliar programming environments for operators proved to be a major challenge [NI, 2007]. This forced the development of products that combined the software capabilities of the PC and the reliability of the PLC. The resulting new controllers combined the best PLC features with the best PC features [NI, 2007]. PACs have evolved incorporating open standard interfaces, multi-domain functionality, modular architectures and modern software integration. [Resnick, 2003].

PACs employ a multi-domain property and are not limited to sequential logic solving. They rely on a single, multi-discipline, integrated development environment that uses common name and process tags as well as a common database for all functions. This integration opens up the possibility of using a single human-machine interface (HMI) for monitoring all functions. The multi-domain capability greatly diminishes total cost by reducing the time needed for design, programming and engineering, while also lowering installation, start-up, training, maintenance and spare parts costs [Iversen, 2008]. Key requirements for any PAC include: open, modular architectures and compliance with industrial standards that range from programming languages (such as the *IEC 61131-3* standard) to communication, networking and interoperability, ranging from Ethernet and its variations, to XML (eXtensible Markup Language) and OPC, an open communication standard [Iversen, 2008].

In summary, Programmable Automation Controllers provide in a single compact controller advanced control features, network connectivity, device interoperability, and enterprise data integration controllers [Iversen, 2008; Resnick, 2003; Opto22, 2008a].

The CompactRIO, an example of this type of technology is presented next. The CompactRIO Programmable Automation Controller, manufactured by National Instruments (NI), is described both in hardware and software structure.

*CompactRIO*

CompactRIO (cRIO) is a Programmable Automation Controller that combines an embedded real-time processor with a field-programmable gate array (FPGA) chip. The FPGA chip is integrated with a reconfigurable chassis that provides the user with the possibility of including swappable I/O modules with built-in signal conditioning for connection of sensors and actuators. A general representation of the cRIO platform is shown in figure 2-25. Modules are connected directly to the FPGA chip on the chassis, which in turn is connected to the real time processor via a high-speed bus following the Peripheral Component Interconnect (PCI) standard.

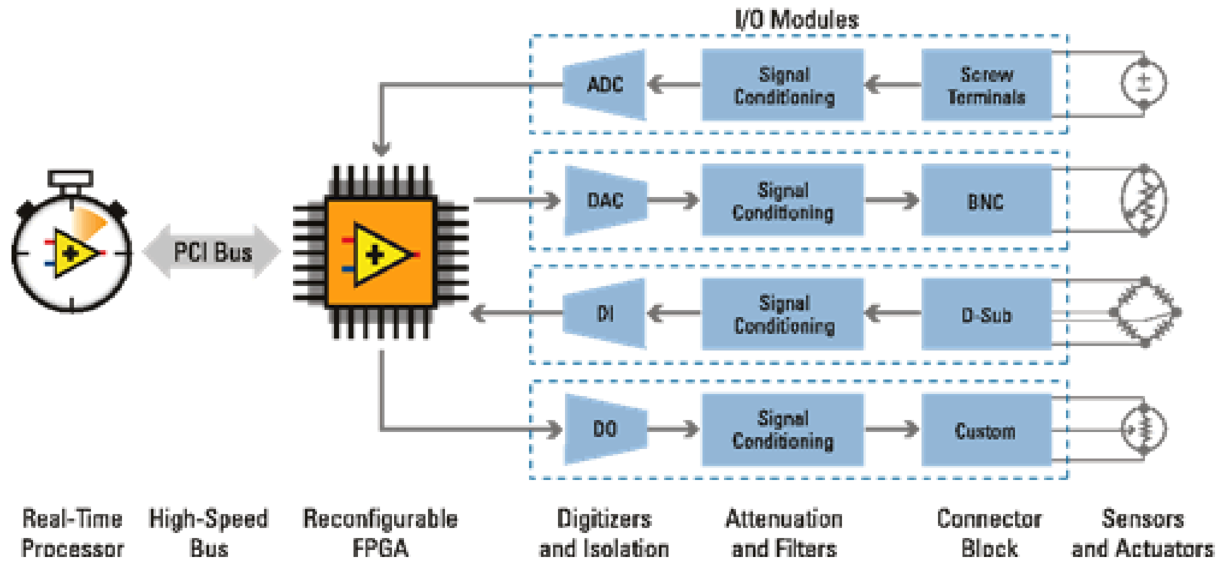


Figure 2-25: Structure of the cRIO embedded system [NI, 2009b].

Custom applications can be achieved using the graphical programming environment LabVIEW and software module add-ons LabVIEW FPGA and LabVIEW Real-Time for programming of the FPGA chip and real-time processor respectively.

LabVIEW FPGA module uses a Xilinx compiler to translate graphical programming into very-high-speed-integrated-circuit hardware description language (VHDL), which is used to program the FPGA chip. The industrial real-time processor deterministically executes applications developed with the LabVIEW Real-Time module. Built-in functions within LabVIEW provide data transfer between the real-time processor and the FPGA chip.

Typical CompactRIO architectures include the following components:

- A reconfigurable input-output (RIO) FPGA core application for communication and control.
- A time-critical loop for floating-point control, signal processing, analysis, and point-by-point decision making.
- Normal-priority loop for embedded data logging, remote panel Web interface, and Ethernet/serial communication.
- Networked host PC for remote graphical user interface, historical data logging, and postprocessing.

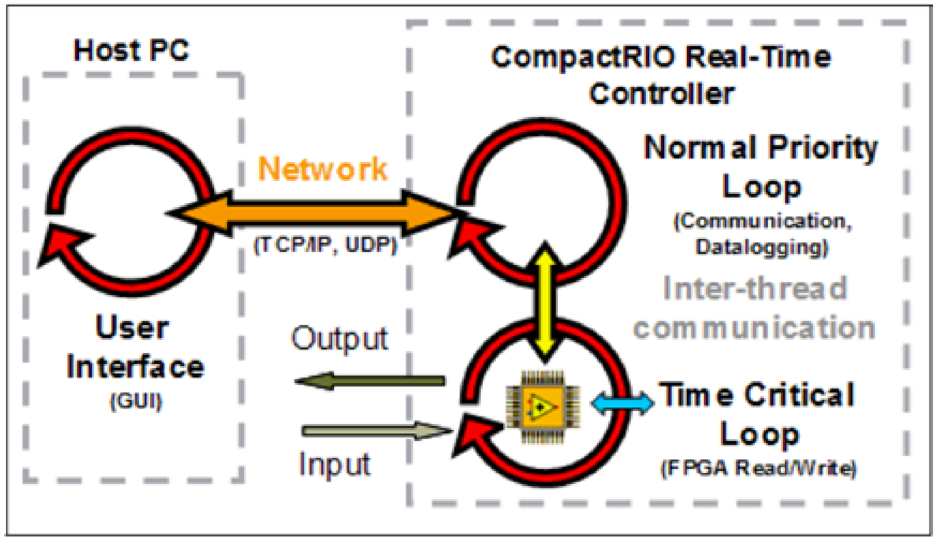


Figure 2-26: Typical architecture for an embedded system using CompactRIO [NI, 2009b].

## 2.2 Design

Design of the CAN network required sub-division of the system and designation of the number of CAN nodes[7]; as well as establishing network attributes such as bandwidth and message properties like data length, type and identifiers. Physical characteristics such as node location, bus termination, cable type, connectors, pin-out, and cable length also required definition.

For devices that are not capable of CAN communication, modules and interfaces had to be designed to integrate them into a CAN node. A careful selection of the elements for each designed CAN node was done; these included microcontrollers

[7] For the purpose of this thesis a node consists of a device or group of devices capable of CAN bus communication.

(MCUs), CAN controllers, CAN transceivers, additional electronic components and plastic containers for each module.

All nodes share the same basic structure, consisting of a microcontroller or central processing unit module, a CAN controller and a CAN transceiver module. Along with these is a custom interface, consisting of input/output channels, relays or signal conditioning elements. Basic structure of the implemented CAN nodes is depicted in figure 2-27.

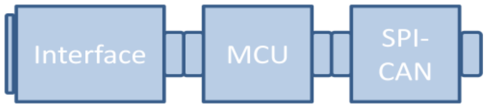


Figure 2-27: CAN Node basic structure.

Because of the nature in which MCUs and CAN controllers can interact, two approaches were evaluated for the development of the modules. Depending on the type of microcontroller, it can either have a built-in CAN controller or the MCU can be interfaced with an external CAN controller via a serial protocol (e.g. Serial Peripheral Interface). Both schemes are shown in figure 2-28.

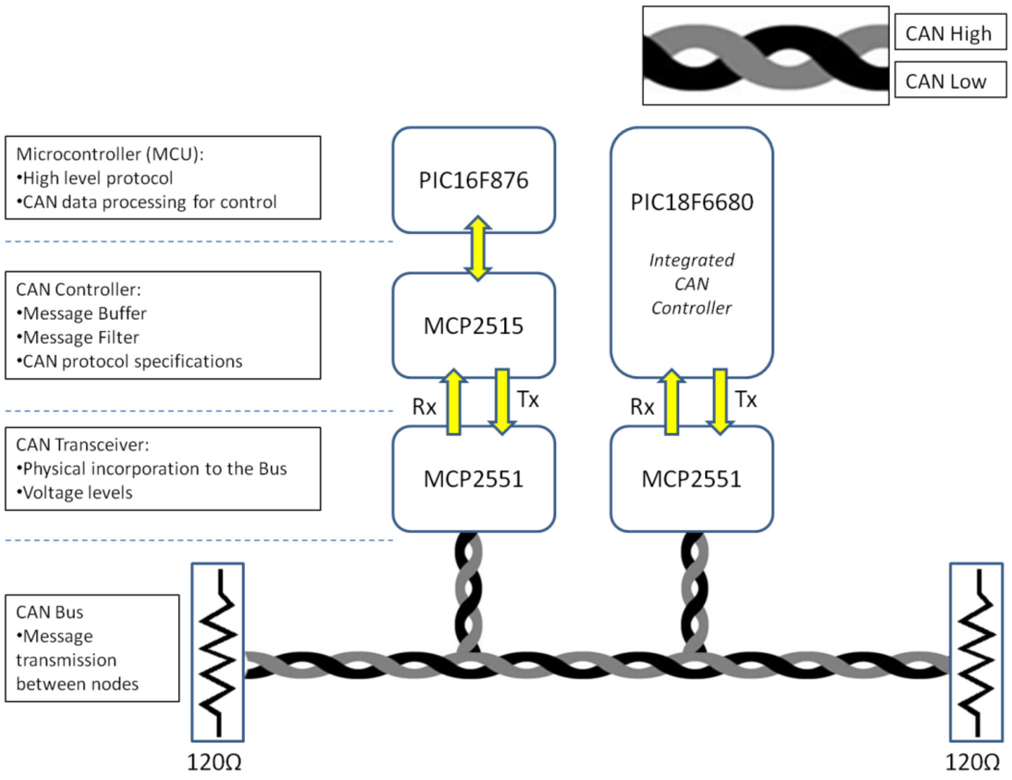


Figure 2-28: Two different but equivalent implementations evaluated for the CAN modules (both PICs are example devices not used during this project).

CAN controllers handle all transmission and reception of CAN messages through the CAN bus, but aren't able to handle the buses physical specifications: CAN High and CAN low voltages used for transmission over the medium. CAN controllers usually have the CAN Tx transmission output and the CAN Rx reception input [Microchip, 2010]; in order to meet the bus specifications, a CAN transceiver is needed.

The advantage of using detached CAN controllers from MCUs is that interfacing with different types of microcontrollers can be done; therefore reusing of programming code from one MCU to another is possible. The downside is that configuration of the CAN controller must be done through the MCU, causing an impact on time efficiency. On the other hand, integrated CAN controllers are more time efficient because of the reduced processing load on the MCU, and physically they occupy less space. A disadvantage is that development done with an integrated architecture may not apply to a second MCU with an on board CAN controller, thus, modularity is sacrificed.

Because of the modularity concerns, the detached communication scheme was preferred over the integrated one. Both CAN transceiver and CAN controller were integrated into a single module. The CAN controllers have a Serial Peripheral Interface (SPI) bus allowing it to communicate with any external device capable of understanding the SPI protocol. In this manner, CAN controller and transceiver are not restricted by design to interact with a certain type of microcontroller. The microcontrollers used in each node were confined to an individual module; this module is named the MCU module. Along with these modules, custom interfaces were designed to grant both the sensor and Millipak power controller, CAN communication.

Throughout the rest of this document, the resulting CAN controller and transceiver module will be referred to as the SPI-CAN module, while the microcontroller module will be named the MCU module.

Electronic design of the circuits required for each module and interface was modeled using the specialized layout editor Eagle, a software tool for designing printed circuit boards (PCBs).

An economic wheel speed sensor was designed because of the short budget available and the deadline established for this project. A commercial solution was not accessible. The sensor functions as an incremental optical encoder. This type of sensor was chosen due to the simplicity with which it can be implemented and its low cost.

Design of the speed control algorithms was done on the graphical programming language LabVIEW.

## 2.3 Implementation

The CompactRIO controller functions as the main CAN node; it is in charge of monitoring the bus activity, receiving information regarding the vehicle's speed and generating control commands to send through the network. It also acts as an interface for the user through a personal computer.

The Millipak controller is in charge of regulating the current and voltage supply to the 5HP electric motor. A CAN node installed with the Millipak receives CAN frames and traduces the information into a speed command value for the controller to drive the motor.

The wheel speed sensor is mounted on the drum brake. The sensor's signal is received by an interface which is part of the sensor's CAN node. The node calculates the speed at which the wheel is turning and sends it through the network.

Location of the nodes is as described next and shown in figure 2-29:

- N1: Interfaces with the Millipak power controller for control of the motor's speed and moving direction.
- N2: Interfaces with the wheel speed sensor for speed data transmission.
- M: The main controller node is composed by the CompactRIO controller; which performs the control algorithm and transmits its output to the network.

These three nodes are the essential minimum nodes required to effectively develop a Network Control System using the CAN protocol for speed control.

Nodes N3 and N4 are under current development, but not included in this thesis. N3 is designed in order to reduce all the non-power wiring for the onboard control panel of the vehicle. N4 is for steering control, this node will be in charge of controlling the actuator that drives the automatic steering mechanism.

Modules, interfaces and CAN hubs were manufactured in the Robotics Laboratory at Tecnologico de Monterrey. To control production, modify designs and generate routing data of the PCB prototypes editing software CircuitCAM from LPKF was employed. Control software BoardMaster was used to operate the c30 series circuit board plotter from LPKF. Soldering of the PCB components as well as assembly of the plastic housing for each of the modules was done manually.

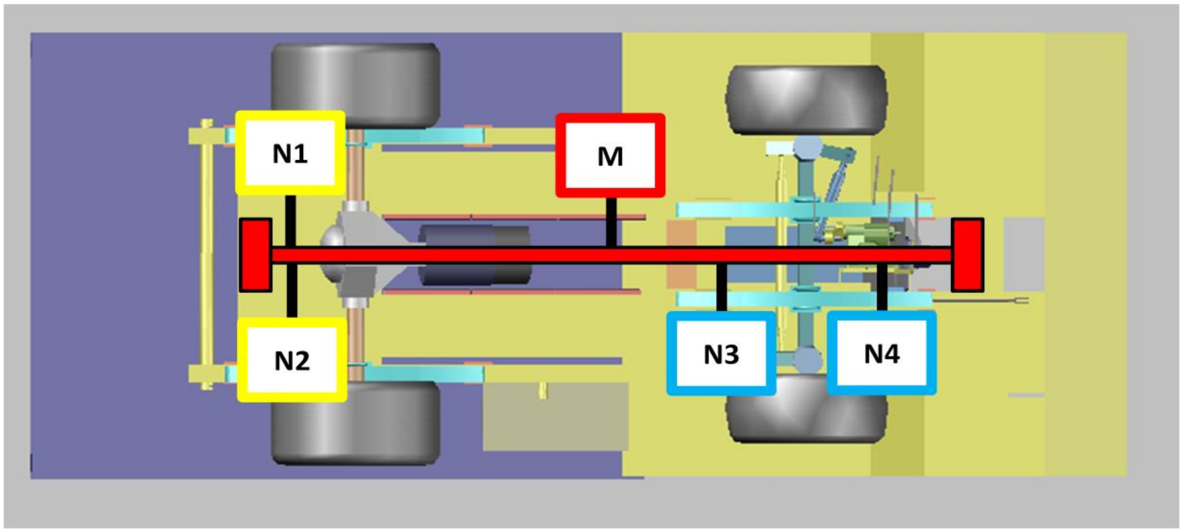


Figure 2-29: CAN nodes physical distribution along the vehicle (bottom view).

Programming of the MCUs on each of the modules was achieved using MikroElektronika's MikroC compiler. Programming of the control algorithms was done on a personal computer using LabVIEW and deployed on the cRIO platform.

Each CAN node, including the main node was mounted onto a DIN rail attached to the vehicle's chassis. Mounting and calibration of the wheel speed sensor was done on one of the rear traction wheels.

Because a trade-off between speed and length exists on the bus, growing capacity is limited by application performance; length and baud rate are parameters that should be carefully selected. Low speed transfer rates mean a greater delay, which can lead to performance bottlenecks; high speed rates increase demand on network interfaces and limit expansion. Real-time control over safety applications requires high speed data rates that range from 250kbps to 1Mbps, other non safety secondary applications can operate safely at 125kps or below (Low Speed CAN) [Leen & Heffernan, 2002; NI, 2009d].

Since the utilitarian vehicle is roughly 2.5 meters long and 1.3 meters wide, taking as reference its perimeter, considerably less than 40 meters of cable is needed to run the wiring around the vehicle; so the maximum baud rate the CAN protocol permits in its specification was chosen, 1Mbit/s, classified as "High Speed CAN". To achieve this transfer rate the physical medium or bus line must meet the requirements specified in ISO 11898. Table 2-5 shows these specifications.

Table 2-5: High Speed CAN cable requirements. [NI, 2008]

| Characteristic | Value |
|---|---|
| Impedance | Minimum: 95Ω<br>Nominal: 120Ω<br>Maximum: 140Ω |
| Length-related resistance | Nominal: 70mΩ/m |
| Specific line delay | Nominal: 5ns/m |

According to the standard, to eliminate reflection effects, terminator resistors should be attached at both ends of the main line. Their value should match the nominal impedance of the cable used; which is typically 120Ω.

CAN cables were manufactured with category 5 unshielded twisted pair (UTP) cable and 120Ω terminators were placed at the end of the main bus line. UTP category 5 cable meets the aforementioned requirements for lengths below 40 meters.

Connection to the main CAN line, for each CAN node, is made through a stub. By specification, the main line should not exceed 40 meters and stubs shouldn't surpass 0.3 meters to satisfy the 1Mbit/s baud rate; although performance of the network can be evaluated to determine the maximum stub length allowed.

Since the CAN protocol works with differential voltages, it is important that all the CAN nodes share a common electrical ground; otherwise, undesired communication errors may arise from voltages changing with respect to different references.

A proportional-integral-derivative (PID) controller was proposed as the speed control algorithm. A fuzzy logic based controller was easily developed and implemented to compare its performance to that of the PID. Both control schemes were implemented within the CompactRIO controller.

Figure 2-30 shows a general conceptualization of the CAN based NCS structure. Nodes M, N1 and N2 are represented with their corresponding modules and interfaces; interaction between elements is represented by the yellow arrows unless a twisted pair connection is depicted in which case it is done through the CAN bus. The control loop is closed with the Millipak controller node (N1) receiving information from the CompactRIO node (M) through the network and driving the motor, which moves the wheels and generates a signal on the sensor node (N2), which measures the wheels speed and sends the data back to the network for the CompactRIO to receive, effectively closing the control loop through the CAN bus.

Details on the implementation of the wheel speed sensor, CAN nodes and CompactRIO controller are presented on the Implementation chapter.

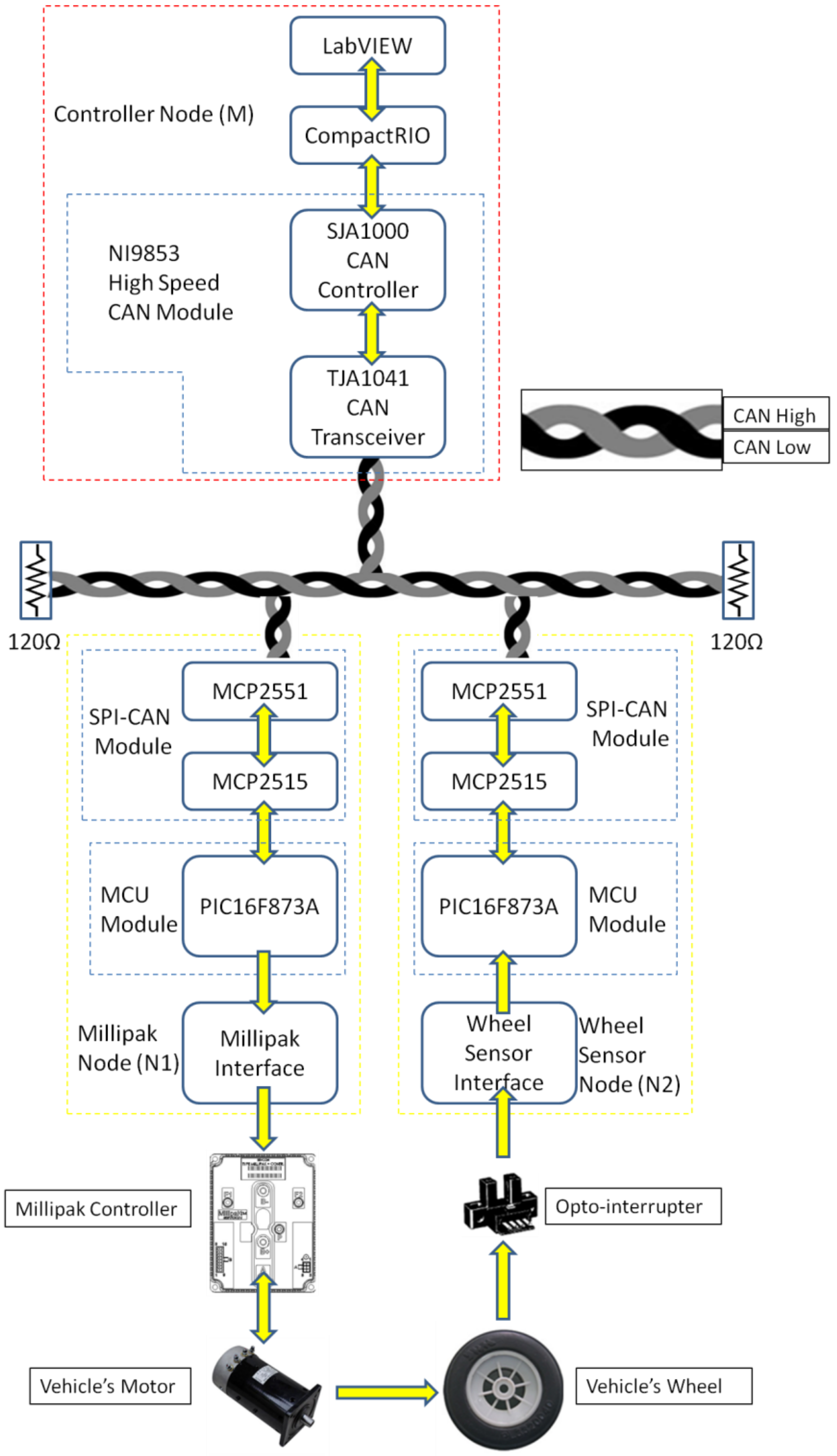


Figure 2-30: Conceptualization for the CAN-based NCS implemented.

## 2.4 Testing

All initial tests were performed indoors at the Robotics Laboratory. The vehicle was suspended from the ceiling and only one wheel was tested at a time. The non-tested rear wheel's drum brake was at its maximum, while the tested wheel was free of any braking, forcing the differential to transmit all torque to the wheel of interest; in this manner speed fluctuations due to the differential transmitting torque from one wheel to another were avoided. The vehicles bed was removed to provide an accessible work zone.

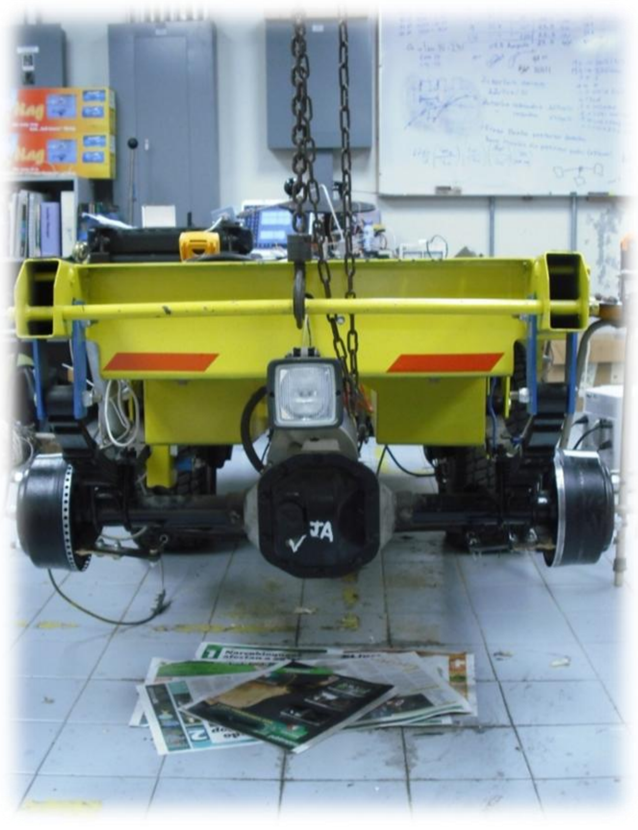


Figure 2-31: Testing conditions for the vehicle.

Performance of the vehicle degrades as battery charge depletes. Speed is not constant since it depends on the batteries current capacity. It is important to mention that batteries on the vehicle are not on the best condition; they are at least 3 years old and were inactive for a long period of time. Reactivation of the chemical components inside the cells had to be done before all testing procedures. In spite of chemical reactivation good performance is still time limited because of these factors.

Tests between the CompactRIO controller and the manufactured CAN nodes were performed to ensure proper communication over the experimental CAN network. To guarantee functionality of the CAN bus, bus load percentage, error count and message frequency were monitored with the USB-MUX-4C2L module with USB connection, which is specialized diagnose equipment from ExxoTest, and a personal computer.

Without a control algorithm, only tele-operation of the vehicle's speed through the CAN bus was possible. With all devices on the network configured to work at a 1Mbit/s baud rate, communication tests consisted in sending a message with the reserved ID for the Millipak node to move the vehicle with a percentage of its max speed. The data contained on byte [0] of the CAN frame sent to the Millipak node is a CAN speed control value (the desired speed information coded as a number on the CAN frame), it ranged from 0 for 0% to approximately 150 for 100% speed.

These tests also involved the monitoring of the messages broadcasted by the sensor node and the transfer and receiving of information through both the cRIO and the diagnostic equipment used. The goal of the tests was achieved without many complications. Figure 2-32 presents the schematic of the setup used for network testing.

Repeatability tests were performed for the Millipak node. Repeatability was monitored to observe the variations on the system's response for the same operating points and determine the fidelity of the developed node.

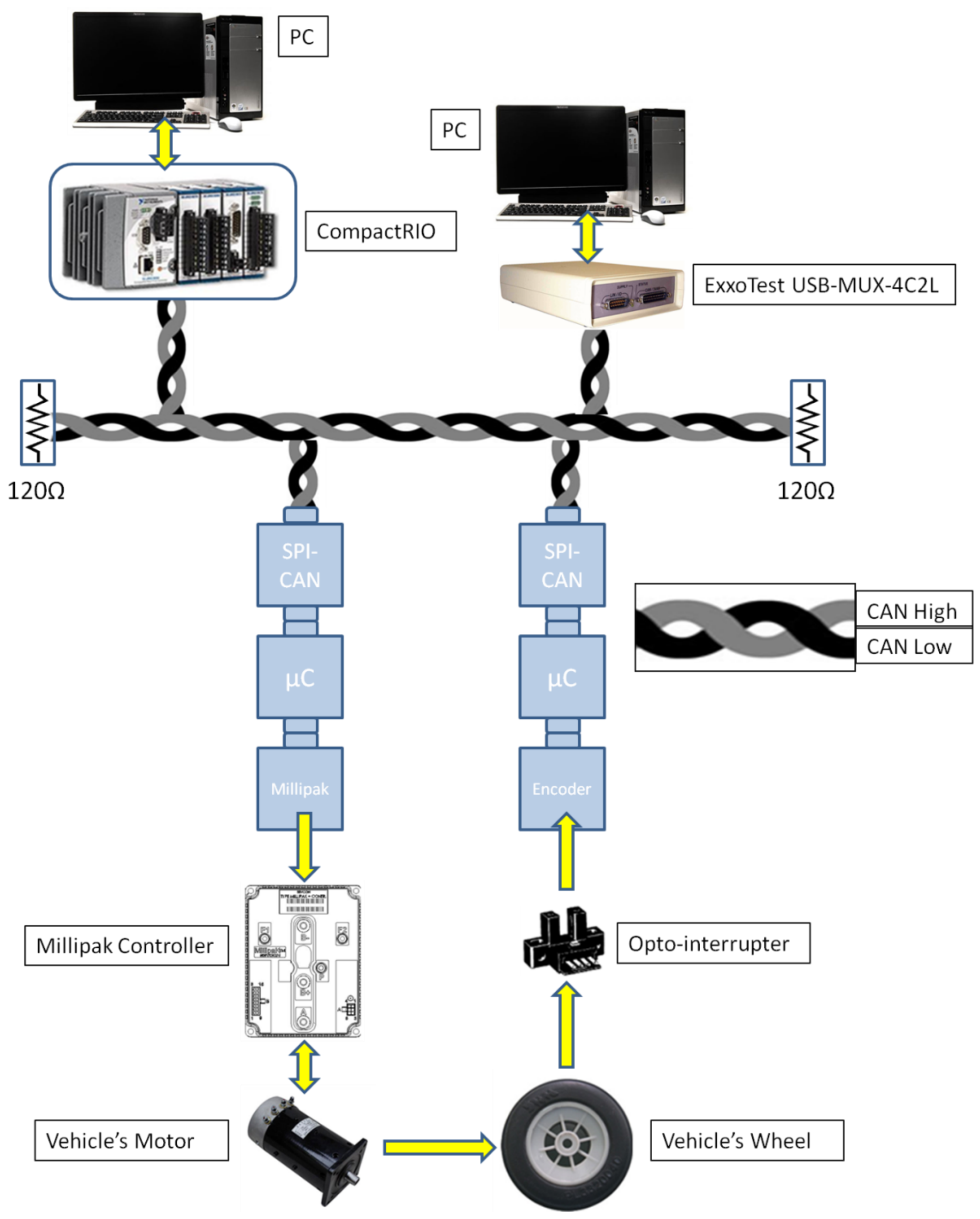


Figure 2-32: Schematic of the developed CAN bus.

### 2.4.1 Repeatability Test

Repeatability of the Millipak CAN node was tested running the vehicle from 0% to 100% and from 0% to 50% full speed without the rear tires. Measuring equipment used for this test included a photo-tachometer from Extech Instruments, model PocketTach 461700. The photo-tachometer was fixed on a stand to maintain a certain distance above the drum and restrict its movement. For the photo-tachometer to sense the revolutions per minute of the wheel, the drum was marked with a white reflecting tape. All runs of the test were performed in a minimum light controlled environment with the vehicle's batteries at full charge. The setup for this test is shown in figure 2-33.

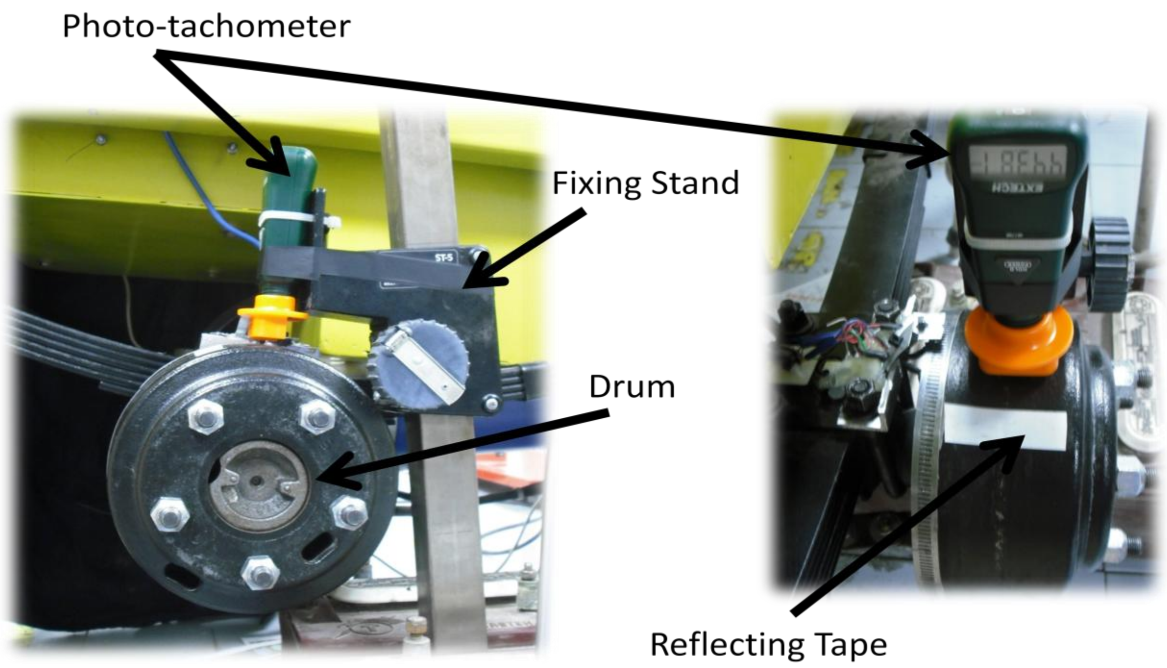


Figure 2-33: Repeatability test setup.

For each of the rear wheels, the 100% and 50% speed tests consisted of 50 samples. First, the 100% test was done for the left[8] wheel, followed by the 50% test. Then the right wheel was tested in the same manner. Time between each sample was random.

The results for the 50% of max speed test are shown in figure 2-34. Both wheels are around the 229 RPM mark. Figure 2-35 shows the results for the 100% of max speed test. It is clear that the right wheel spins faster than the left wheel at maximum speed, but this situation is of no concern, since the differential will distribute the torque between both wheels when the vehicle is rolling on the ground. On both figures the *x* axis represents the sample number.

[8] Looking at the front of the vehicle the left wheel is the one on the passenger's side.

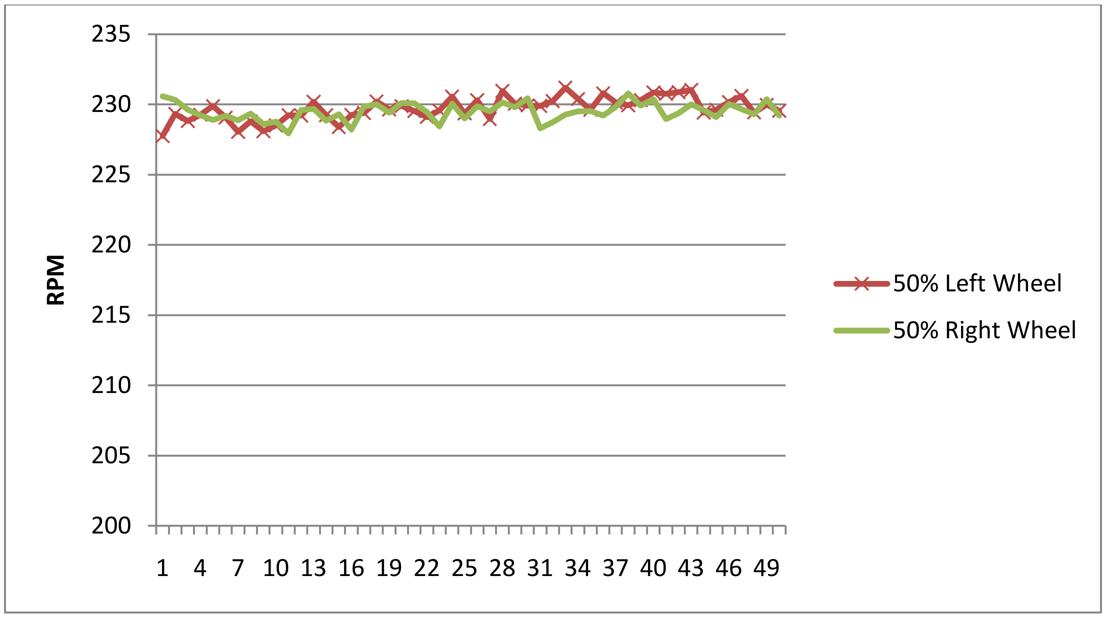


Figure 2-34: RPMs versus sample number are shown as results of the 50% of max speed test.

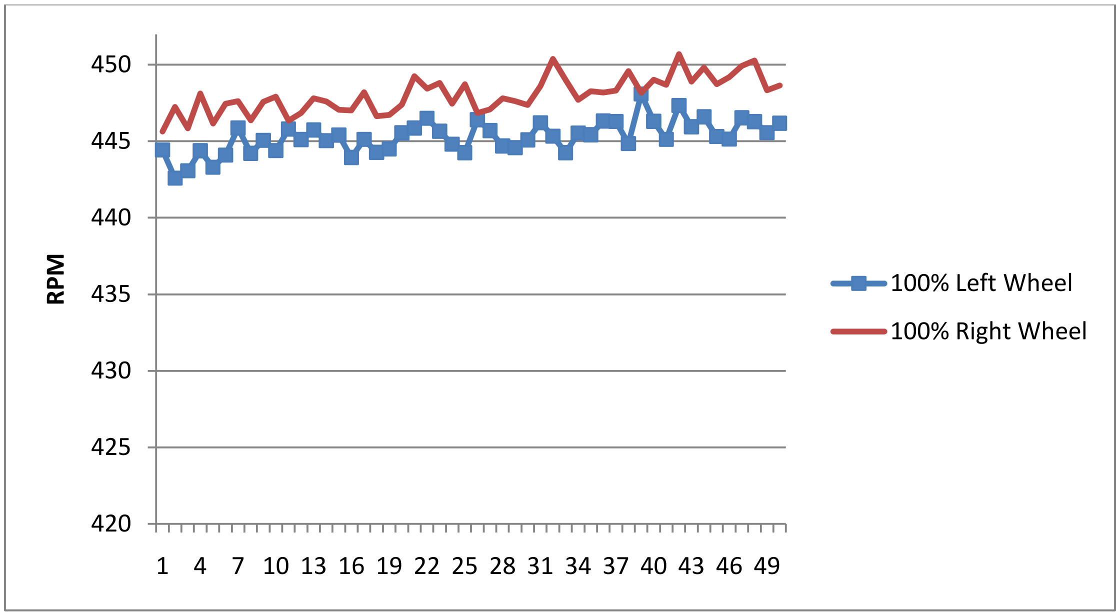


Figure 2-35: RPMs versus sample number are shown as results of the 100% of max speed test.

Speed varies from sample to sample. A maximum difference of 6 RPMs was observed throughout the test. This is considered acceptable since this variations are unnoticeable while driving the vehicle. Tables 2-6 and 2-7 concentrate the results of each wheel (all values are in RPMs).

Table 2-6: Results for the left wheel

| **Left Wheel** | | |
|---|---:|---:|
| **Test Speed** | **50%** | **100%** |
| **Average** | 229.6996 | 445.2814 |
| **Maximum** | 231.18 | 448.09 |
| **Minimum** | 227.75 | 442.59 |
| **Std. Deviation** | 0.80465726 | 1.050471245 |

Table 2-7: Results for the right wheel

| **Right Wheel** | | |
|---|---:|---:|
| **Test Speed** | **50%** | **100%** |
| **Average** | 229.4794 | 448.026 |
| **Maximum** | 230.78 | 450.7 |
| **Minimum** | 227.94 | 445.63 |
| **Std. Deviation** | 0.63737475 | 1.173529475 |

The speed of the vehicle can be approximated by computing the average speed between both wheels, like assumed by the bicycle simplification of the kinematic model. The combined average from both right and left wheels is 446.6537 RPM for 100% speed and 229.5895 RPM for 50% speed.

Although these results were obtained with a suspended wheel, they proved to be significant while testing the control algorithms under the same laboratory conditions. In the case of the PID controller they provided an upper bound reference, required by the algorithm since it calculates the controller's output based on a percentage rather than the whole.

It is important to mention that during these first tests, cables used had no terminator resistors and the main bus line was directly over the batteries, next to the electric motor, extremely susceptible to electromagnetic noise; as the tests proved successful, the robustness of the CAN protocol was also demonstrated.

For subsequent experiments the cables included termination and the main bus line was running through the side of the vehicle.

## 2.5 Corrections and Modifications

During the repeatability test, a first group of 50 samples for the 100% speed measurements was done on each wheel; it was observed that the measured RPM values increased too much over time. For the same CAN control value different speeds were achieved. There was a difference of around 21 RPMs from the first sample to the fiftieth.

After removing the drum, residues of the brake's pressing pads and extreme surface heat were detected; this brought forth the conclusion that friction was slowing down the wheel. With time the pads wore out enough for the wheel to increase its speed, hence the variations. To deal with this issue, adjustment of the pressing pads was done and the insides of the drum cleaned. After these correcting actions, the tests were repeated completely, performing quite well; variance was reduced from nearly 20 to 1.1 and 1.3 for the 100% tests and from almost 10 to 0.64 and 0.40 for the 50% tests on the left and right wheel respectively. Only the test results after these corrections were considered valid.

Past the first testing period of the control network a modification was done to the Millipak interface on the Millipak CAN node. Initially, a 10kΩ digital potentiometer was used to replace the manual throttle of the vehicle, which consists of a 5kΩ analog potentiometer. Since the digital potentiometer has a range of 10kΩ, only half the resolution of the potentiometer was available for control of the throttling system. To double the resolution of the interface, a second digital potentiometer was connected in parallel; allowing more accurate control of the Millipak controller. The PCB of the interface suffered no redesign since it allowed for this type of connection.

The wheel speed sensor developed suffered various modifications. Three different attempts of the sensor were fabricated, each an improvement over the previous version. The first two creations used a metal strip with circular perforations; the last and definitive attempt used a metal strip with rectangular perforations, minimizing a problem caused by asymmetrical periods on the sensors pulse signal.

During the first tests of the sensor node, the cables used to transfer the pulses from the opto-interrupters to the sensor interface were unshielded. The quality of the TTL signal from the sensor was severely affected during operation of the vehicle. Noise levels on the signal varied, going from slightly affecting the pulses to introducing undesired impulses as shown in figure 2-36. To deal with this issue, cables were replaced with twisted pair wire; resulting on an improvement on signal quality as depicted on figure 2-37.

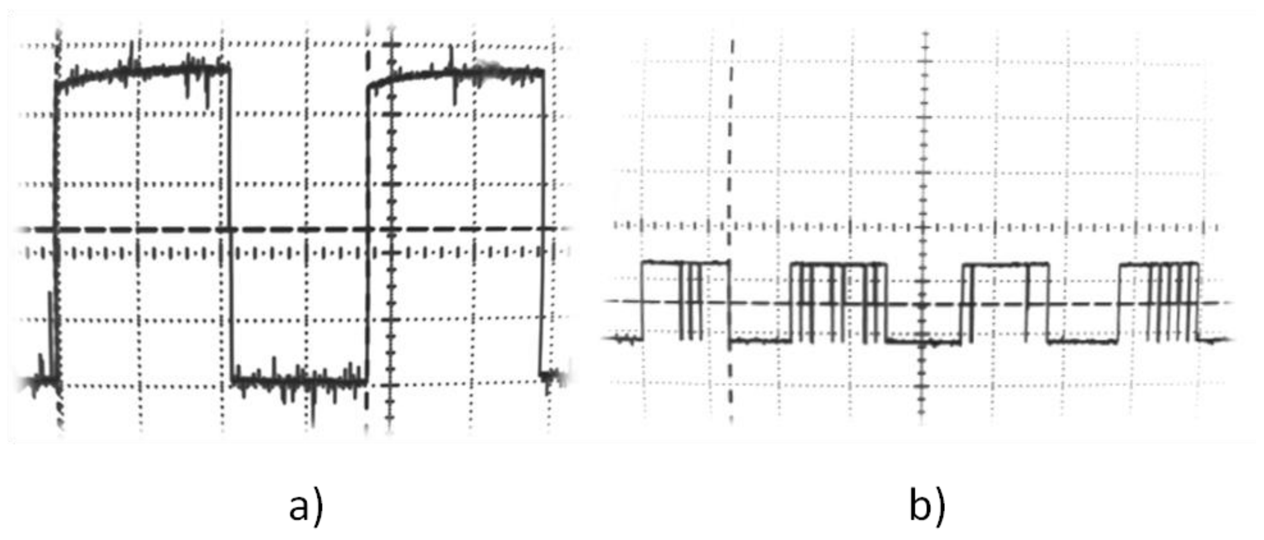


Figure 2-36: a) Slightly affected signal quality. b) Undesired impulses between pulse signals generated during the operation of the vehicle.

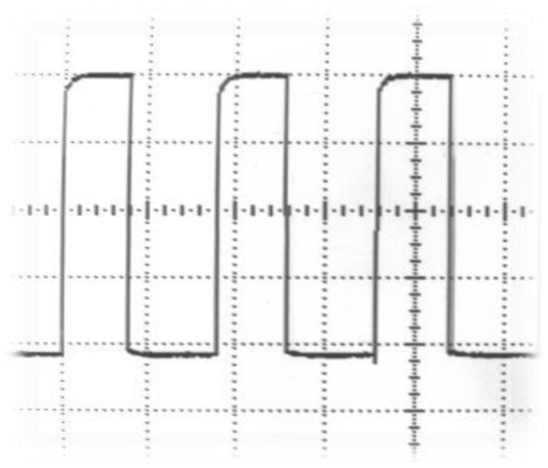


Figure 2-37: Signal quality improvement from the usage of twisted pair wire.

The CAN nodes were initially powered directly by three of the six deep cycle batteries that power up the vehicle; but due to internal circuitry on the Millipak controller, this resulted in unwanted behavior that required the replacing of some of the voltage regulators on the modules. All nodes including the CompactRIO were separated from the main 36 volt power supply and now run on an independent 12 volt battery.

Throughout the whole development of this thesis, different software modifications and improvements were done to the LabVIEW program on the CompactRIO controller.

# Chapter 3

# Implementation

This chapter presents the implementations developed for the creation of the network controlled system based on the CAN protocol for speed control of the electric utilitarian vehicle. Elements previously mentioned on the Methodology chapter, like the manufactured sensor, the modules and interfaces constituting each CAN node and software implementations are shown in detail.

## 3.1 Wheel Speed Sensor

Due to complications an adequate sensor was not accessible during the development of this project. A temporal low cost alternative had to be manufactured in order to perform all tests. A wheel speed sensor had to be improvised using an opto-interrupter; a perforated metal strip and the drum brakes located on the rear wheels.

The sensor consists of an opto-interrupter aligned with a perforated metal strip; it works as a relative incremental encoder. The opto-interrupter has a diode constantly emitting infrared light over a phototransistor, when light emission is interrupted the voltage on the phototransistor's line changes, dropping close to 0 volts and on the contrary, when the light beam passes through, the output signal is close to 5 volts. As the sensor is intermittently interrupted pulses appear; each with a period equal to the time from the start of one pulse to the start of the next; by knowing the number of pulses per revolution and the frequency (or period) of these, the speed at which the wheel is turning can be calculated.

Details on the first two versions of the wheel speed sensor are presented on Appendix D. The third version of the sensor was implemented using a strip with 241 rectangular perforations providing a high resolution for RPM calculations. Compared with the first two versions of the sensor, improvement was noticed on signal quality.

Although not very much industrial, the ITR8102 opto-interrupters were used because of their high sampling frequency.

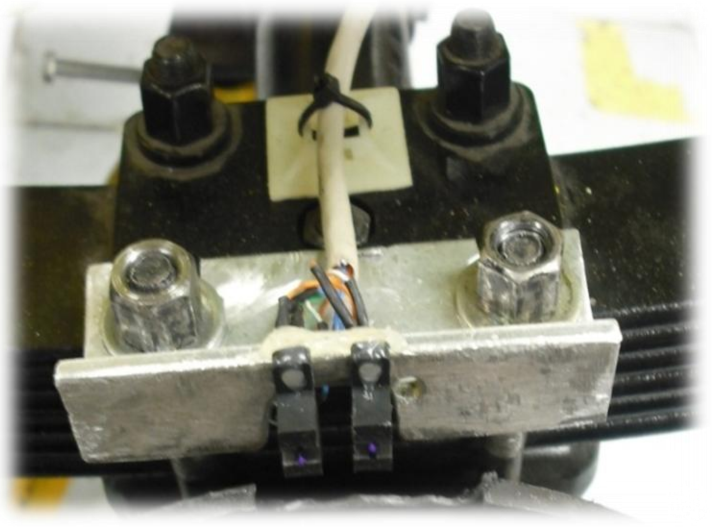


Figure 3-1: ITR8102 opto-interrupters used for the wheel speed sensor.

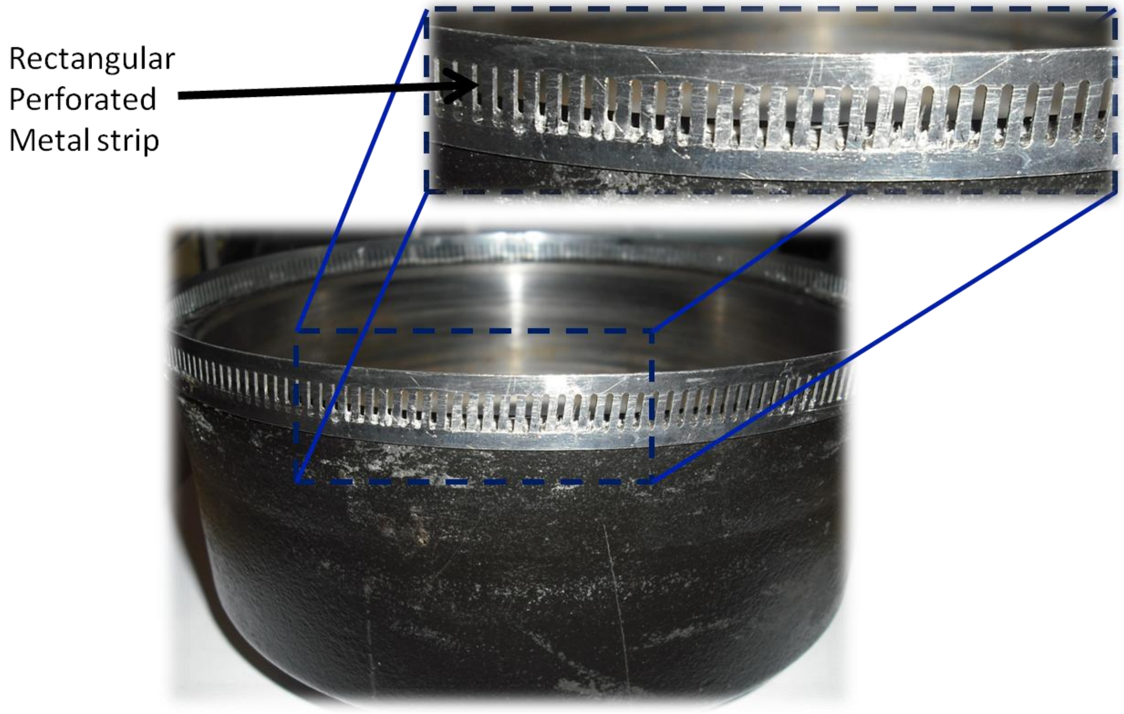


Figure 3-2: Third version of the wheel speed sensor, using the rectangular perforated metal strip.

## 3.2 Modules and Interfaces

All SPI-CAN modules were assembled in the same manner; they contain the same components and their operation is the same for every MCU module they are attached to. All MCU modules share the same PCB design and vary just in the programming of the microcontroller in it. Encoder and Millipak interfaces are unique.

All PCBs were manufactured on single-sided copper clad boards and all components soldered to the boards are of the through-hole type. Every integrated circuit included in the designs has a socket for an easy replacement in case of malfunction.

All modules and interfaces have a 5V voltage regulator for low power consumption and electronic protection for sensible components. Capacitors were included to compensate for any voltage drop and filter noise. Every module has a 20 MHz crystal as its main oscillator. Finally ever module and interface has its own custom plastic case for protection.

### 3.2.1 SPI-CAN Module

The SPI-CAN module is in charge of receiving messages from the bus and transferring them via SPI to an MCU module and vice versa. An MCP2551 CAN transceiver and an MCP2515 CAN controller are used for CAN communication. An MCU module must configure the SPI-CAN module with network parameters through the SPI channel during startup.

SPI-CAN modules are not self-powered; they require a connection through pins on its DB15 HD (high density) connector to a module that can supply power.

The MCP2515 is a CAN stand alone controller capable of implementing the CAN 2.0B specification, which means it can handle standard and extended IDs. This integrated circuit has two acceptance masks and six acceptance filters used to reduce the overhead on the processing device by ignoring unwanted messages. It can interface with any processing device via an industrial Serial Peripheral Interface (SPI) [Microchip, 2007].

The MCP2551 is a high speed CAN transceiver, functioning as an interface between the physical bus and the CAN controller. This integrated circuit provides differential transmit and receive capability by performing conversion of the digital signals generated by the CAN controller to signals suitable for transmission over the medium, also it includes a buffer to account for voltage spikes that may be induced on the bus line by external factors [Microchip, 2010].

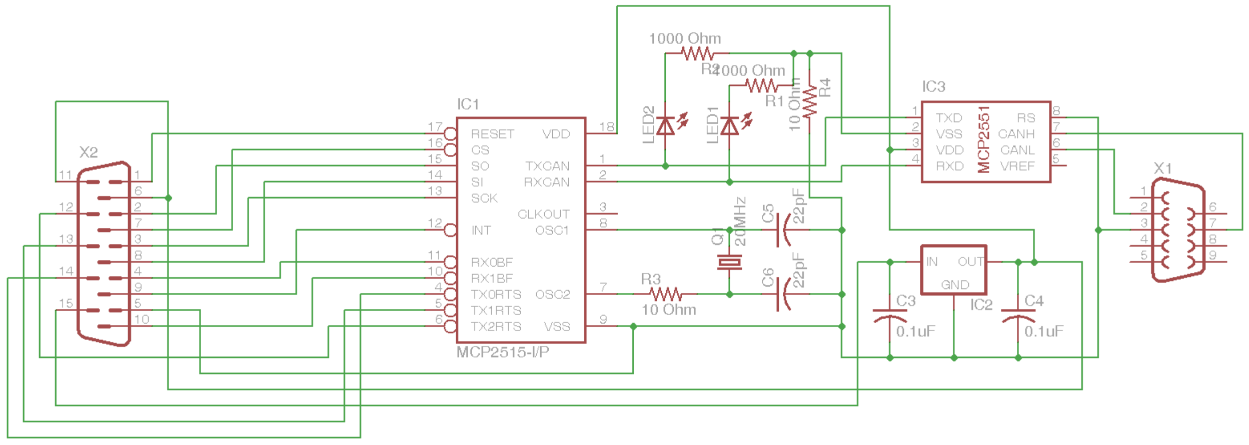


Figure 3-3: SPI-CAN module electric schematic.

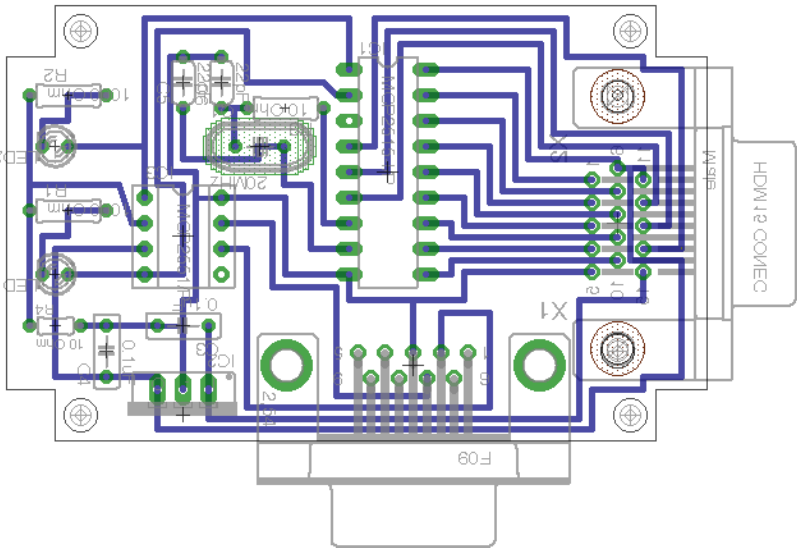


Figure 3-4: SPI-CAN module PCB schematic.

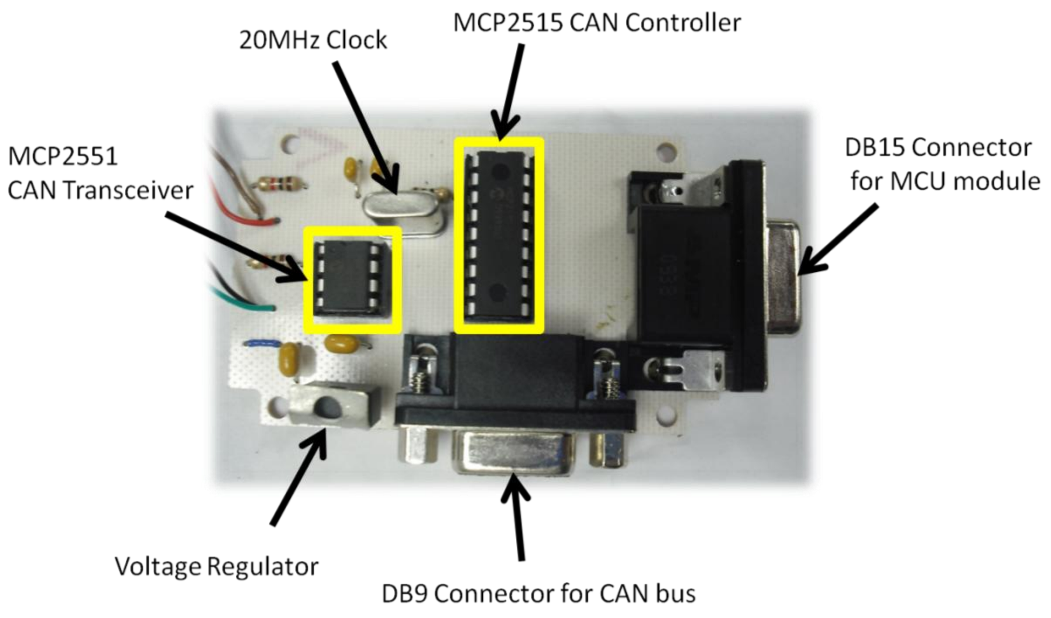


Figure 3-5: SPI-CAN module.

## 3.2.2 MCU Module

The MCU module is programmed through an In-Circuit Serial Programming (ICSP) terminal. Programming of the microcontroller varies according to the function of the CAN node where the module will be located. The MCU module is in charge of processing and interpreting the data contained in a CAN message.

The MCU module included with the Millipak node receives CAN messages through the SPI-CAN module, processes the information for speed control (direction and desired speed) and sends control signals to the Millipak interface. The MCU module on the wheel speed sensor node receives a pulse train signal from the sensor, calculates the corresponding revolutions per hour (RPH) and sends the information to the CAN bus through an SPI-CAN module.

The MCU modules use a PIC16F873A microcontroller as their central processing unit. They have 2 DB15 HD connectors mounted on each side for its input/output channels. The modules also contain their own oscillator and voltage regulator.

The PIC16F873A is an 8-bit microcontroller with 128 bytes of EEPROM and capacity for 3 wire SPI communication. This particular model was chosen because of its small size, speed (200 nanoseconds per instruction), number of inputs and outputs, and the Serial Peripheral Interface communication option [Microchip, 2003].

Like the SPI-CAN modules, MCU modules are not self-powered; they need to be connected to an interface that provides power, although they can share their power connections on one of the DB15 connectors.

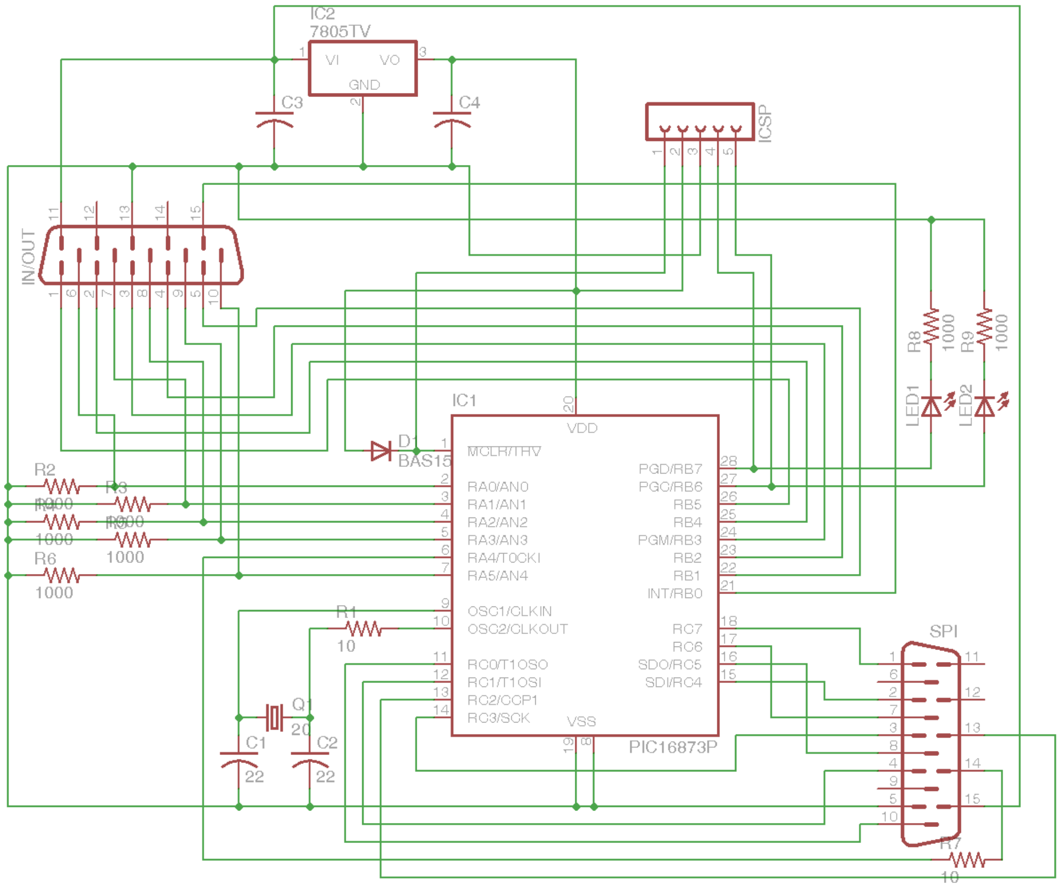


Figure 3-6: MCU module electric schematic.

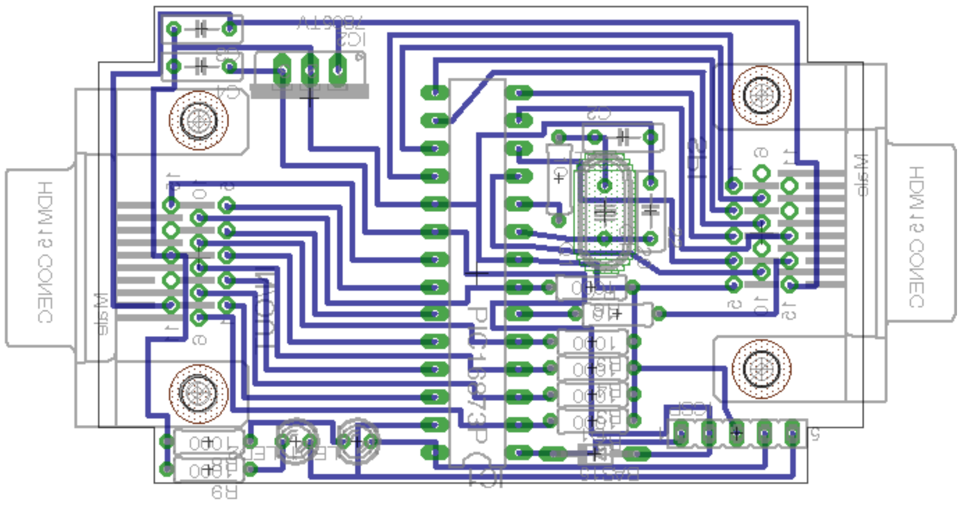


Figure 3-7: MCU module PCB schematic.

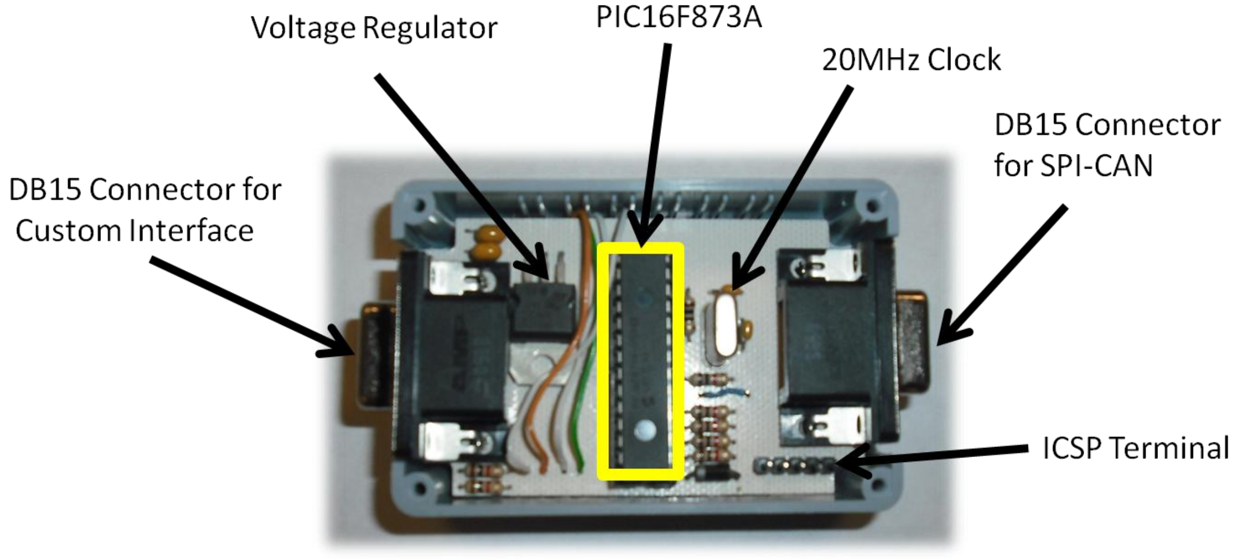

Voltage Regulator
PIC16F873A
20MHz Clock
DB15 Connector for SPI-CAN
DB15 Connector for Custom Interface
ICSP Terminal

Figure 3-8: MCU module.

### 3.2.3 Millipak Interface

The Millipak interface uses an array of three signal relays and two parallel digital potentiometers to control the speed and advancing direction of the vehicle. It has one DB15 HD connector for interfacing with an MCU module and connection terminals for output signals to the Millipak controller.

The only integrated circuit used in this interface is the MCP41010, a digital potentiometer employed to translate the desired CAN speed value into a resistance equivalent. This potentiometer varies the voltage on the accelerator input of the Millipak controller, which varies the accelerator demand on the controller.

The MCP41010 is a 10kΩ digital potentiometer with 256 positions. Its wiper position varies linearly and is controlled through an SPI interface; this made it suitable for interfacing with the PIC16F873A on the MCU module [Microchip, 2004].

This module transmits power to any device connected to the power pins on its DB15 HD connector.

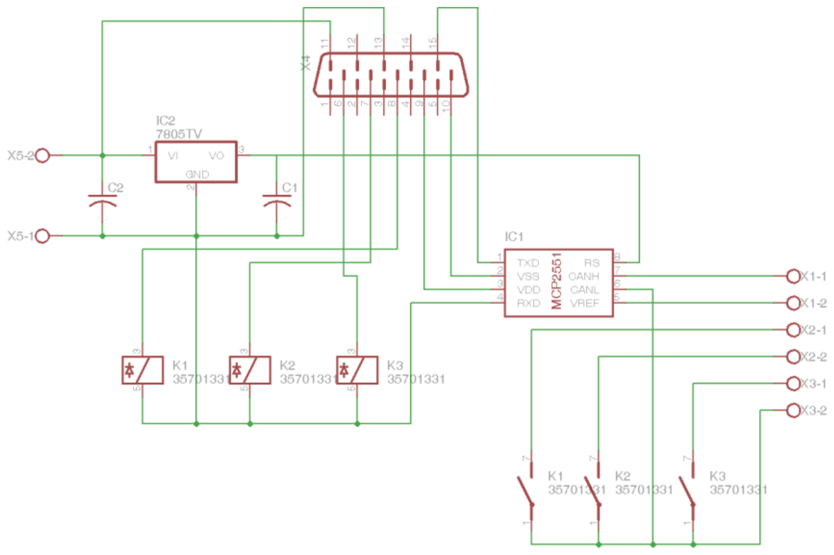


Figure 3-9: Millipak interface electric schematic.

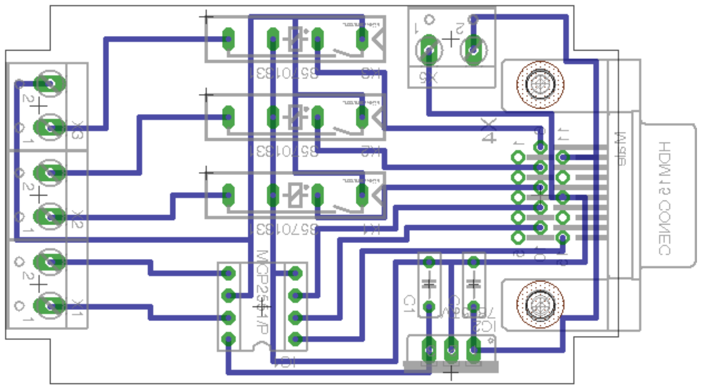


Figure 3-10: Millipak interface PCB schematic.

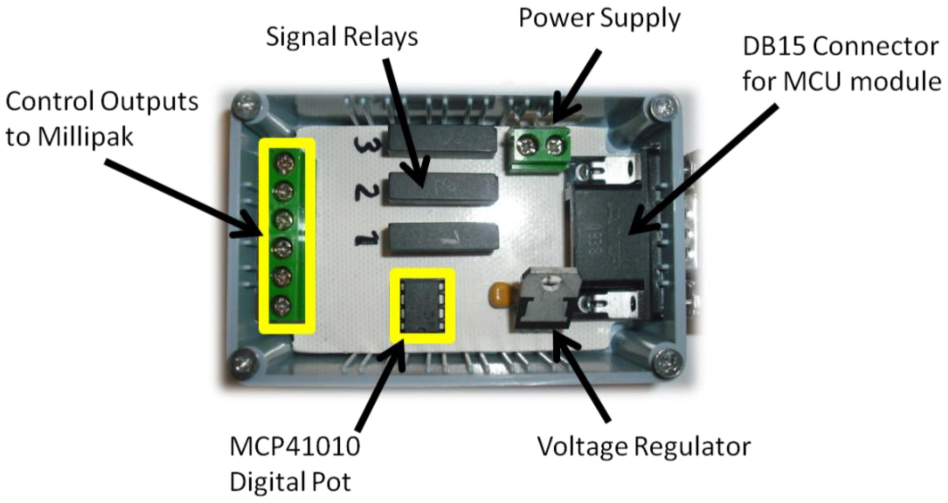


Figure 3-11: Millipak interface.

### 3.2.4 Wheel Speed Sensor Interface

This interface is the simplest of all, it only has connection terminals, power connectors and a voltage regulator. It transmits power to the interfaces attached to it, powers up the wheel speed sensor and yields as an output signal the analog voltage coming from it. Again, on one side a DB15 HD connector connects this interface with the others, while on the other, input terminals for the sensor's signal are available.

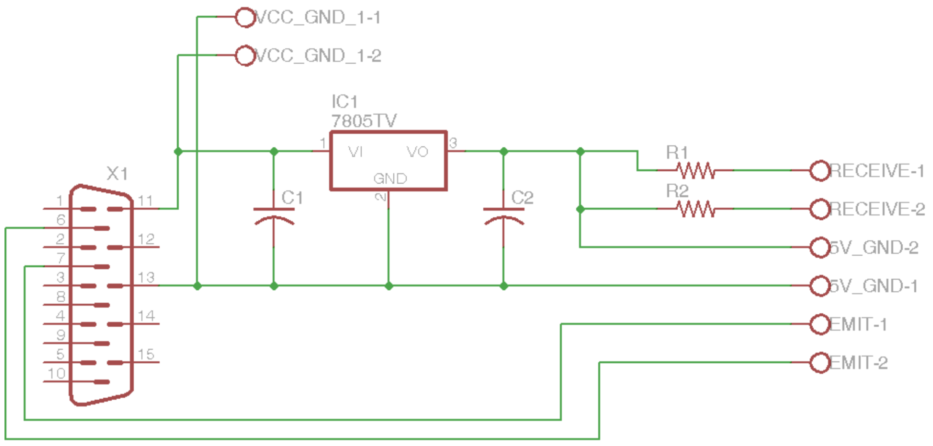


Figure 3-12: Wheel speed sensor interface electric schematic.

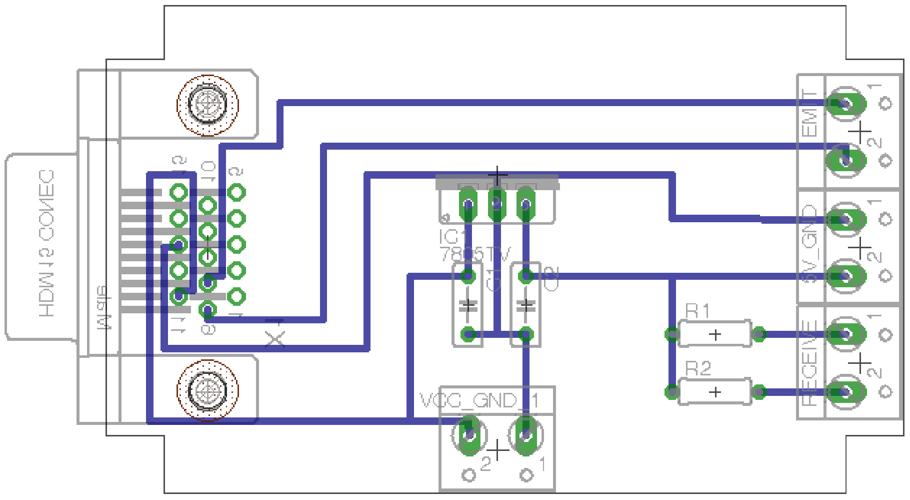


Figure 3-13: Wheel speed sensor interface PCB schematic.

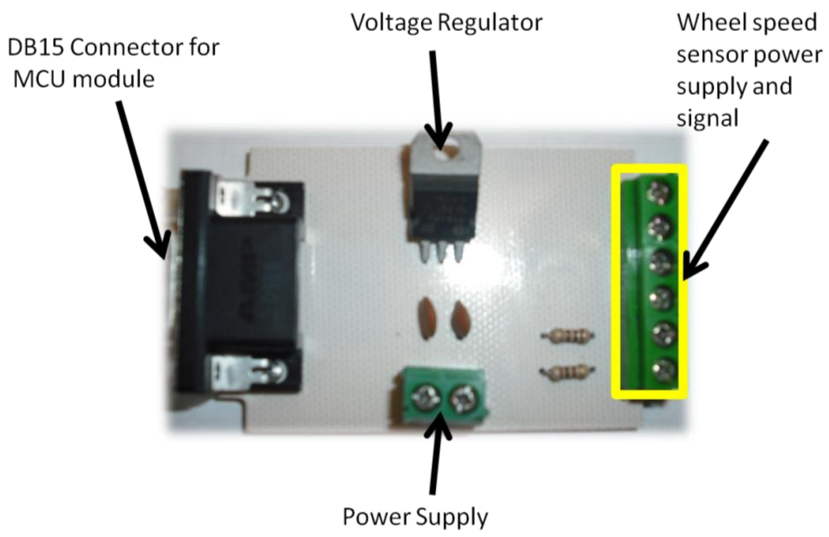

DB15 Connector for MCU module
Voltage Regulator
Wheel speed sensor power supply and signal
Power Supply

Figure 3-14: Wheel speed sensor interface.

## 3.3 CAN Nodes Description

Software elements regarding the CAN nodes will be described in this subsection.

### 3.3.1 Millipak CAN Node

This node only accepts messages with ID 0x300, either extended or standard, and uses the first 3 bytes of data to read the desired speed value, direction, and electronic fail-safe (FS1) contact state respectively. Any other message received is discarded. Messages are accepted from any sender as long as they respect the structure shown in table 3-1.

Table 3-1: Message structure accepted by the Millipak CAN node.

| ID | Byte [n] | Range (Dec) | Description |
|---|---|---|---|
| 0x300 | Data [0] | 0-255 | *Speed Control Value* |
| | Data [1] | | *Moving Direction:* |
| | | 0 | Neutral |
| | | 1 | Forward |
| | | 2 | Reverse |
| | Data [2] | | *Fail-safe contact:* |
| | | 0 | Fail-safe ON |
| | | 1 | Fail-safe OFF |

If the fail-safe contact is on, the node takes no action. The input sequence required for fault-free operation of the vehicle is: Moving Direction → FS1 OFF → Speed Value or within a 2 second time limit, FS1 → Moving Direction→ Speed Value. Once a successful power-up of the Millipak controller is achieved, all data can be sent at the same time on one CAN frame.

Fault-free power-up of the Millipak controller requires the FS1 to be on and the moving direction to be set on neutral. Faults will occur during power-up of the controller if the following events take place:

- A change in the moving direction is selected while the vehicle is speeding in the opposite direction. To change direction an immediate FS1 recycle[9] is necessary.
- The fail-safe is set OFF and the moving direction is not different from neutral (within 2 seconds).
- If the speed command is not zero, a moving direction is selected or the FS1 is OFF during power up.

To clear the faults, FS1 recycle is usually required.

### 3.3.2 Wheel Speed Sensor CAN Node

As soon as the node is powered-up it starts transmitting messages with information regarding the vehicle's wheel speed. CAN frames are broadcasted from this node with ID 0x321. The wheel's speed is coded into the first two bytes of data, using byte [0] as the low part and byte [1] as the high part of the word[10] containing the wheel's revolutions per hour. The structure of the messages sent by the wheel speed sensor CAN node is summarized in table 3-2.

Table 3-2: Message structure broadcasted by the sensor CAN node.

| **ID** | **Byte [n]** | **Range (Dec)** | **Description** |
|---|---|---|---|
| 0x321 | Data [0] | 0-255 | *Speed Value (Low Byte)* |
|  | Data [1] | 0-255 | *Speed Value (High Byte)* |

[9] FS1 recycle means switching the fail-safe to ON and back to OFF.

[10] A word is a natural unit of data used by a particular computer design; in this case it represents a group of 16 bits.

### 3.3.3 CompactRIO CAN Node

The CompactRIO controller performs the control algorithms for the network system based on the information it receives from the wheel speed sensor node. Outputs from the sensor node become CompactRIO node inputs; and outputs from the CompactRIO node become Millipak node inputs. Revolutions per hour sent by the sensor node are converted to revolutions per minute, then they are processed by the control algorithm and based on user specifications, a proper CAN speed control value is sent to the Millipak node.

### 3.3.4 Node Mounting

The CAN bus main line connects to a CAN hub on the back, passes through one side of the vehicle and connects to another hub on the front. Mounting of the nodes was done on a standardized DIN rail. To fix the DIN rail the 11 gauge metal sheet of the vehicle was drilled. Custom clamps are attached to each of the modules and interfaces to fix them to the DIN rail.

Figure 3-15 shows the Millipak node (consisting of the SPI-CAN module, the MCU module and the Millipak interface) mounted on the back part of the vehicle, located under the vehicles bed. Figure 3-16 shows the wheel speed sensor node mounted next to the Millipak controller. Also, on both figures the CAN hub where the main line connects with the nodes stubs is depicted. Figure 3-17 shows a top view of the Millipak and sensor nodes assembly; it corresponds to the network shown on the bottom part of figure 2-30 (without the CompactRIO node).

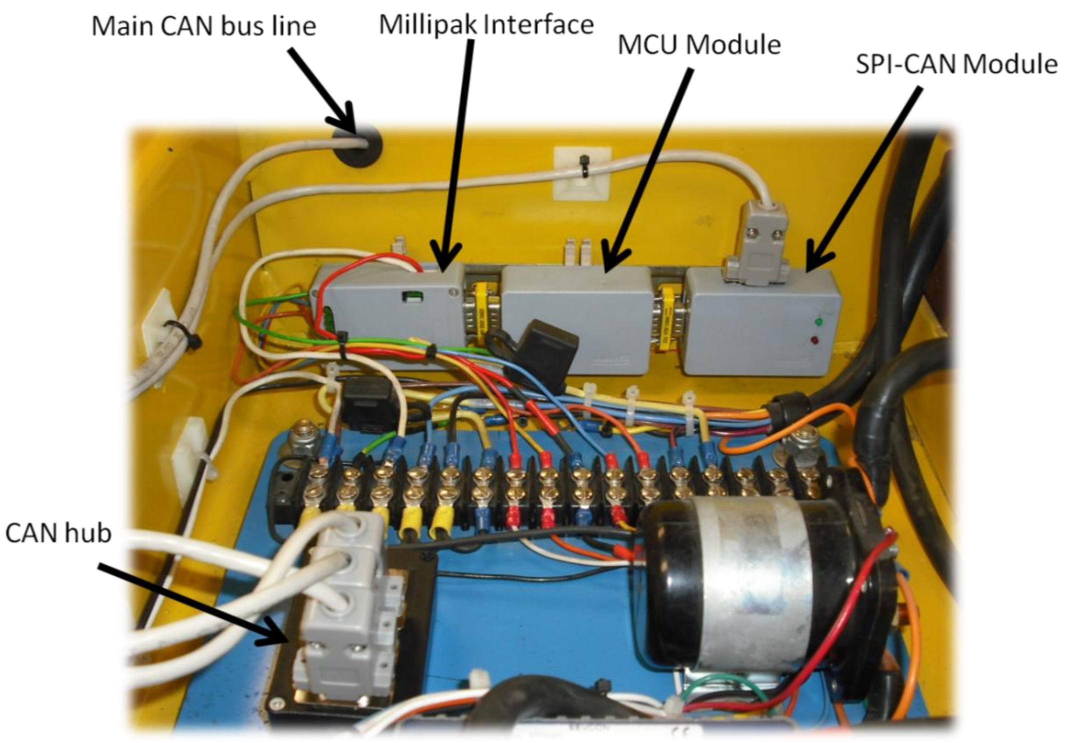


Figure 3-15: Millipak node mounting and main CAN bus line.

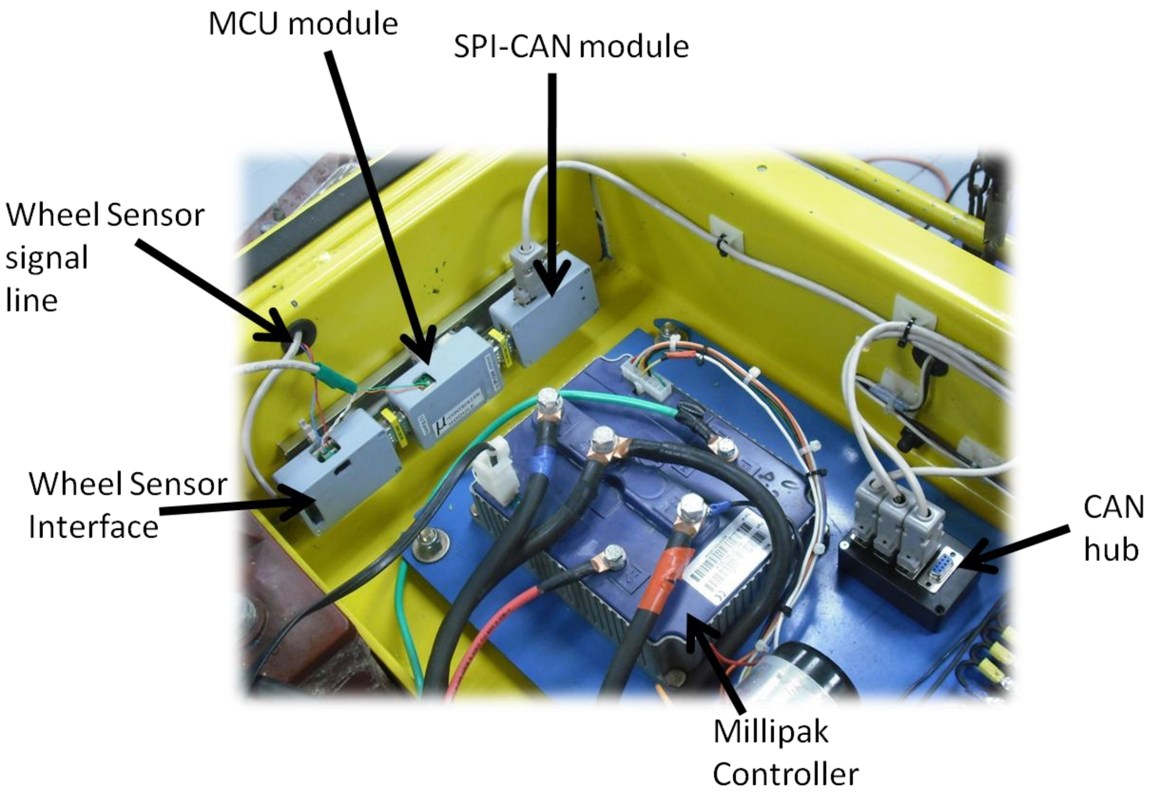


Figure 3-16: Wheel speed sensor node mounting showing Millipak controller.

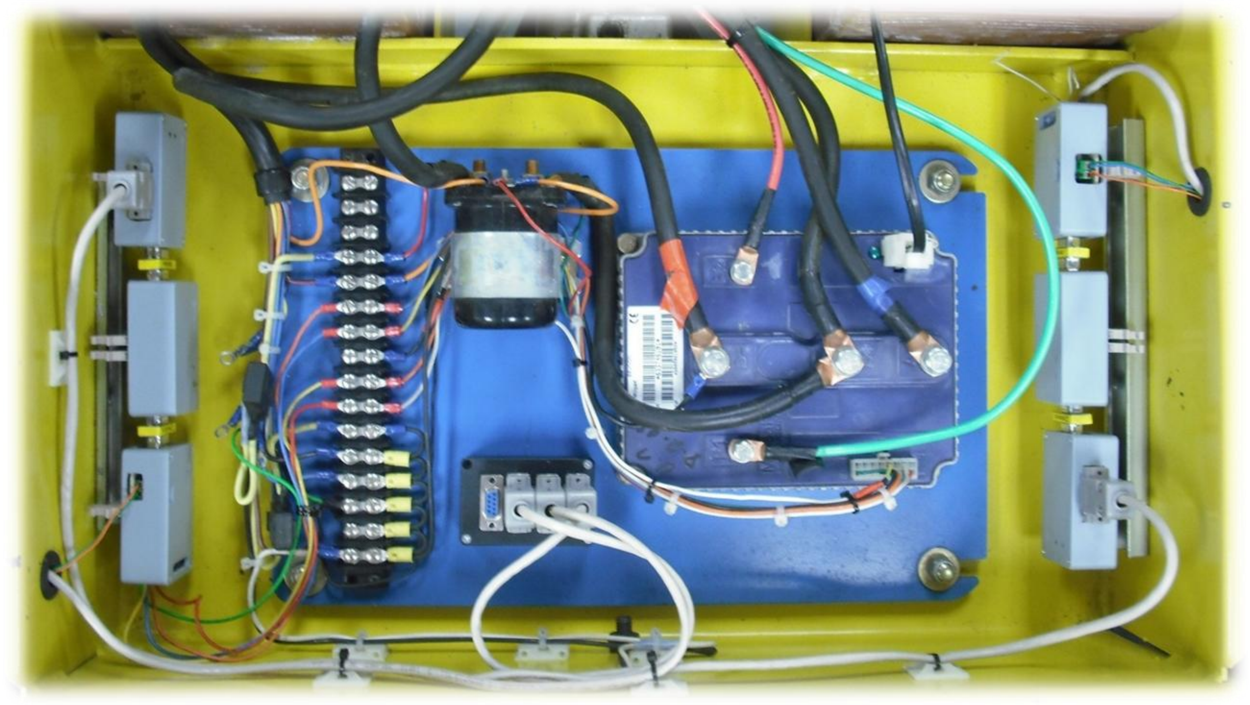


Figure 3-17: Top view of both nodes on the vehicles back part.

## 3.4 CompactRIO

Software applications on the CompactRIO are programmed using LabVIEW and software module add-ons like LabVIEW Real-Time (RT) (for programming applications on the real-time level) and LabVIEW FPGA (for programming on the FPGA level). Programming and configuration of the CompactRIO platform is achieved via the 10/100BASE-T Ethernet port of its on board network interface card.

### 3.4.1 LabVIEW Project Configuration

A project must be created in order to develop an application for the cRIO embedded platform. The project within LabVIEW is structured as shown in figure 3-18.

Three distinct levels can be used for programming. I/O modules are manipulated on the FPGA level and typically high order processes are done on the real-time or PC levels. Communication between the FPGA and RT levels is achieved through direct memory access (DMA) first-in-first-out (FIFO) dedicated channels. Virtual Instruments (VIs) can be created on each of these levels. Depending on the level the VI is placed, it will be handled by LabVIEW, LabVIEW RT or LabVIEW FPGA.

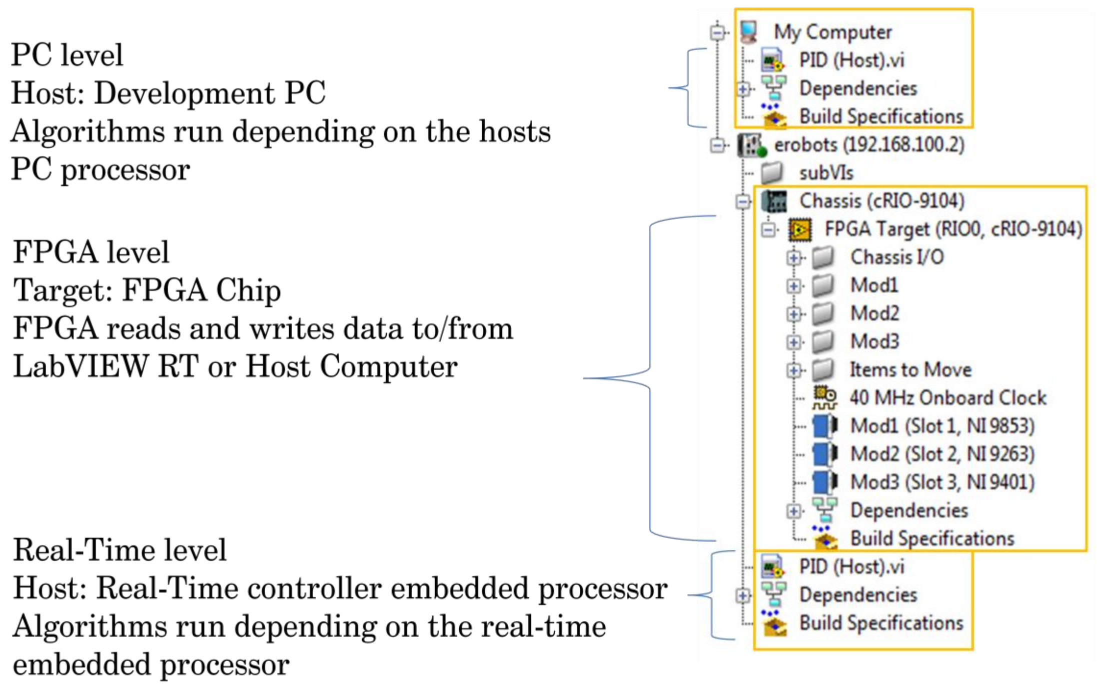


Figure 3-18: Programming levels of a cRIO project.

VIs created on the FPGA level are handled by the LabVIEW FPGA module, in which specialized FPGA blocks become available depending on the FPGA target. Applications created on this level run on the FPGA chip.

VIs created on the RT level are handled by the LabVIEW Real-Time module, it provides communication with the information at the FPGA level, and has special function blocks for optimization of the graphical code for the selected real-time hardware. Applications on this level run directly on the real-time target.

The LabVIEW graphical programming suite running on the host PC can communicate with both the Real-Time and the FPGA software modules. Applications developed on the real-time level can be used on the host PC level to take advantage of features not available on the real-time target. Although these features may prove useful in some cases, reliability and determinism now depend on the host PCs processor capabilities; timing and performance are now subjected to the CPU usage, network errors and operating system crashes on the host PC

The NI 9853 High Speed CAN module of the CompactRIO uses a Philips SJA1000 CAN controller and TJA1041 CAN transceiver on a 16MHz clock. This module provided an easy way to incorporate the capabilities of the embedded platform into the CAN protocol based bus, for fast and reliable communication with devices operating according to this standard. To develop the control model within the

CompactRIO, CAN channels are wired as inputs and outputs to the control model on the FPGA level. A LabVIEW FPGA VI reads and writes CAN frames, and transfers those CAN frames to/from LabVIEW RT as with any other I/O. A LabVIEW RT VI uses virtual interfaces to convert the CAN frames to/from CAN channels and perform the necessary operations on the data included with the frames.

The CompactRIO is mounted on a DIN rail, close to the vehicles seats, for easy connection of a personal computer to act as interface between the controller and an operator.

## 3.5 Timing

Timing throughout this thesis was based on the constraints imposed by the time boundaries for CAN frame transmission. Assuming all nodes are error active and the bus is not idle, an extended frame with 8 bytes of data (the most time consuming data frame possible) takes at least 131 microseconds (with not a single bit stuffed) and at most 155 microseconds (with the maximum bit stuffing possible). For this reason, any process on the network that relies on information traveling through the CAN bus cannot take less than 155 microseconds or it may become compromised.

The tools from ExxoTest were used to determine the network induced delays between CAN nodes. By transmitting messages on a fixed time rate; the period of each transmission can be monitored. The transmission rate was set to 50 milliseconds on both the sensor node and the cRIO. The message period varied at most in 0.003 milliseconds from message to message. This was considered an acceptable delay, since it won't interfere with control actions because of the natural response of the system.

## 3.6 RPM Measurement

The vehicle's speed, measured in revolutions per minute (RPM), is approximated using the traction wheel's rotating speed. Because of the nature of the sensor used, two different techniques for RPM measurements were applied, one resulting in better performance than the other.

RPM measurements were first made using the period method [Opto22, 2008b]. The sensor's signal period is defined as the time elapsed from the start of one pulse to the start of the next pulse. Perforations on the metal strip are neither exactly the same size, nor evenly separated; causing asymmetrical pulses. Under constant speed, this situation provoked RPM measurements to vary from one sampling time to the next one, producing outliers on the signal.

A second measurement method consisted on counting the positive or negative edges over a fixed period of time [NI, 2010]. Although pulse periods were still asymmetrical, by counting edges for a fixed time period, normalization of the pulse distribution over this time span was performed, yielding much better results. Both methods are shown in figure 3-19. The greater the time span defined for edge counting, the less outliers in the signal. But with long time spans uncertainty increases, because during the time defined to count edges, the last speed measurement is held constant; if speed changes too abruptly and the defined time span hasn't elapsed, the sensor won't be able to register such changes. For this reason an appropriate sampling time on the controller and edge counting time span must be defined.

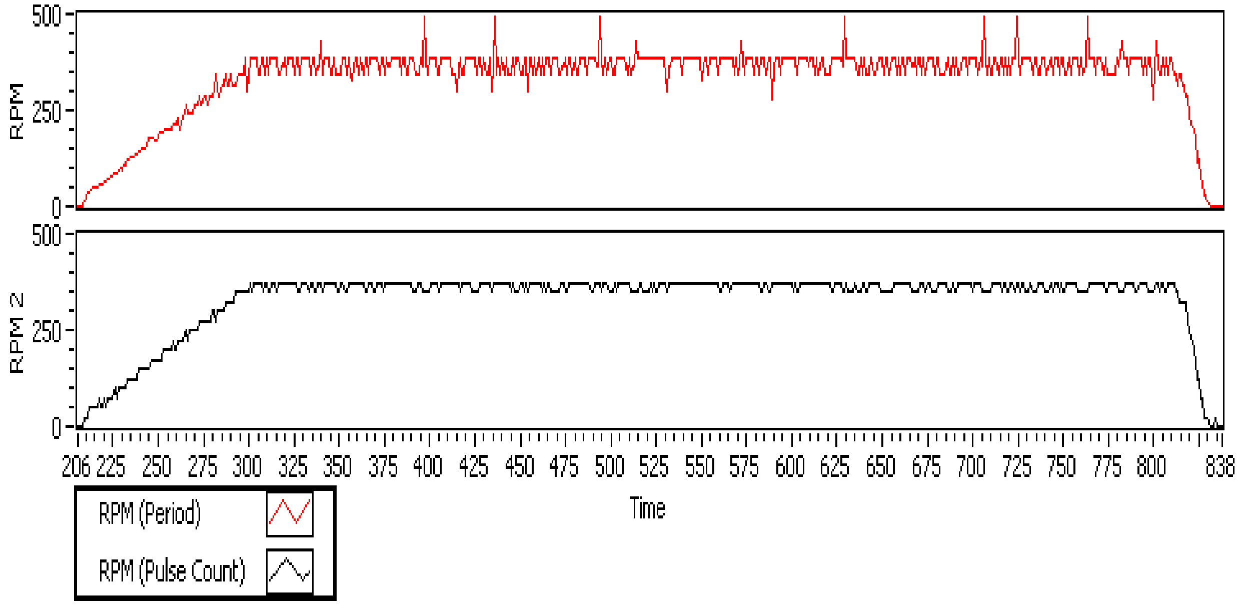


Figure 3-19: On top, RPMs are measured using the period method. Below, measurements are made using the pulse count method.

The microcontroller inside the MCU module on the wheel speed sensor CAN node was programmed to use the edge count method. Revolutions per hour of the traction wheel are sent through the CAN bus every 50 milliseconds.

On the CompactRIO controller, LabVIEW was used to develop a tachometer application to calculate the corresponding RPMs, according to the information provided by the wheel speed sensor node.

Two redundancy measurements were used to ensure the RPMs displayed by the cRIO application were accurate. First, with the setup used for the repeatability tests, the photo-tachometer readings were contrasted to the ones shown by LabVIEW. Then, frequency on the output signal of the sensor was measured using an oscilloscope connected directly to the opto-interrupters cables. Frequency on the output signal was manually converted to RPMs through simple mathematical operations and compared to the results shown by the cRIO.

While holding the speed of the vehicle constant, the redundant measurements from the three sources were within 5 RPMs of each other most of the time, thus concluding that the tachometer implementation in LabVIEW was successful.

Even though the measurements were made using the pulse count method, fluctuations on the signal were still present. To reduce outliers in the signal even further, a mode filter and a median filter were applied through software.

The first filter applied computes the mode of a sample set of user defined size; the filter finds the mode of the set by calculating the histogram of the input data and selecting the center value of the bin with the largest count; these calculations introduce a time delay that depends on the length of the sample set. The filter proved to effectively reduce the number of oscillations present in the signal, but the trade-off between response time and oscillation frequency was increased.

The second filter applied, computes the median of a user defined sample set (called rank on the VI) that is used to define an equal number of elements to the left and to the right of the point of interest, for selection of the median. The time delay of this filter is smaller than the one of the mode filter, presenting equal or better results on outlier removal. As a conclusion, the median was preferred over the mode filter.

A Real-Time VI was used to send the desired speed (RPM setpoint) on a CAN frame to the Millipak CAN node. Also, this VI monitored and logged the RPM measurements obtained.

To compare results for RPM calculation, the period method is plotted along with the pulse count method, and both the mode and median filters. The median filter was also applied to the period method to observe its behavior. Figure 3-20 shows the different methods and filters for the RPM measurements obtained. Outliers are clearly visible for the period method (top plot). Each unit on the time scale represents the Real-Time VI's loop cycle or sampling time. Sampling time was set to 30 milliseconds, which means that from one mark on the *x* axis to the next 30 milliseconds have elapsed.

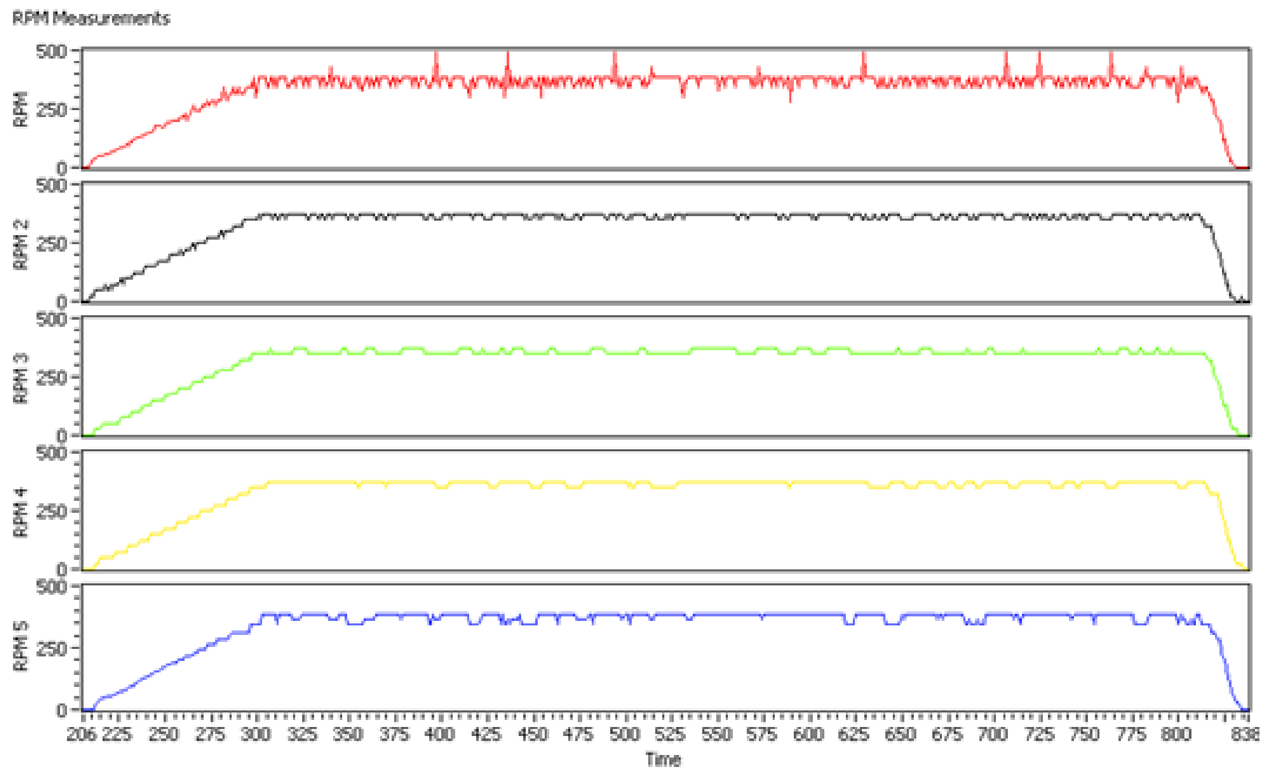


Figure 3-20: Plots from top to bottom; the period method, the pulse count method, mode filtered pulse count method, median filtered pulse count method and median filtered period method.

Figure 3-21 shows a close-up on the "constant" section of figure 3-12. All methods present outliers, but the median filter reduces both frequency and amplitude of these, thus providing a better and more reliable approximation to the actual RPMs of the wheel.

From figures 3-22 and 3-23 it can be observed that the use of a filter induces a delay of approximately 3 time units, which results in 90 milliseconds for a sampling time of 30 milliseconds, independently of the measuring method used. It should be noted that the pulse count method yields a step shaped (discrete) measurement compared to the period method which appears smoother. Also, the application of a filter produces a discrete signal regardless of the measuring method.

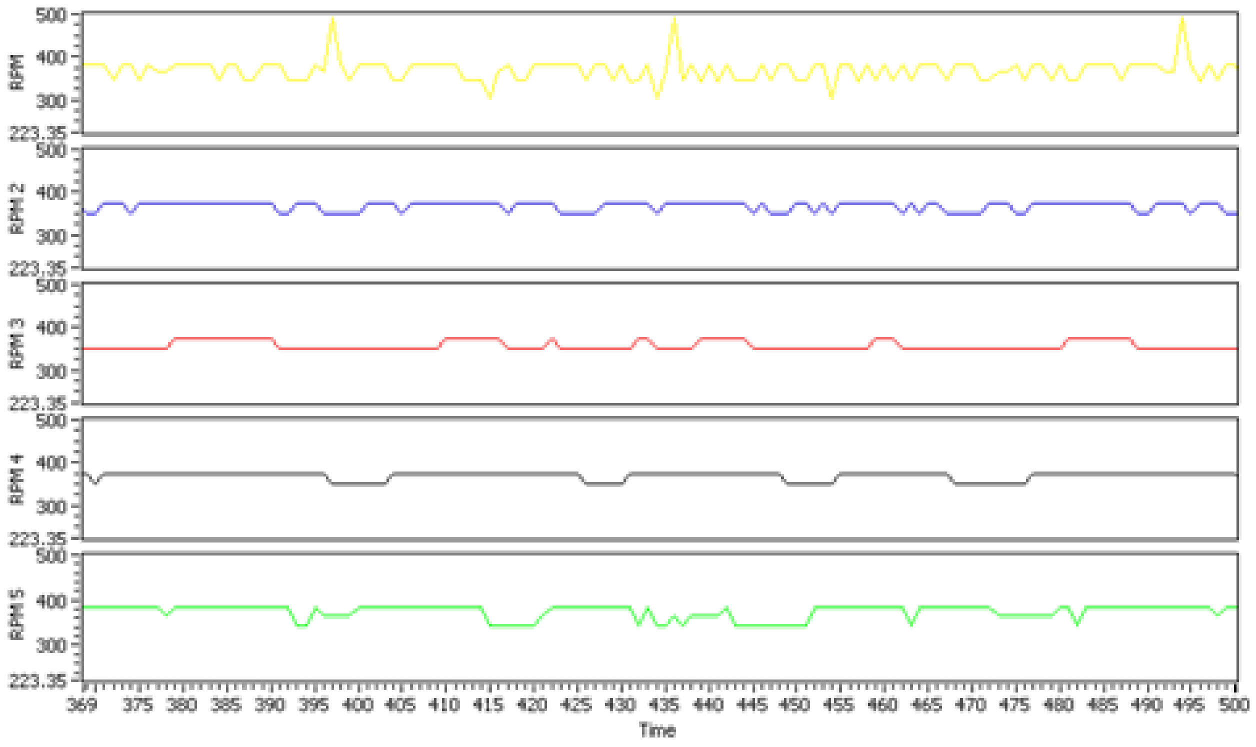


Figure 3-21: Close-up on the constant speed section of figure 3-20. Again, plots from top to bottom; the period method, the pulse count method, mode filtered pulse count method, median filtered pulse count method and median filtered period method.

Figure 3-24 shows the systems response due to a setpoint change. The pulse count method without filtering (middle plot) proves to be an improvement over the period method (bottom plot), which clearly looks messy. The median filtered pulse count method (top plot) demonstrates to be the most stable of all methods. The figure shows that for the acceleration and deceleration portions of the graphics, the three plots share a similar behavior, only during the constant or cruise speed portion of the plot a big difference is remarkable between each method. An improvement in the quality of the graphic is evident from applying a filter to the best of both core methods.

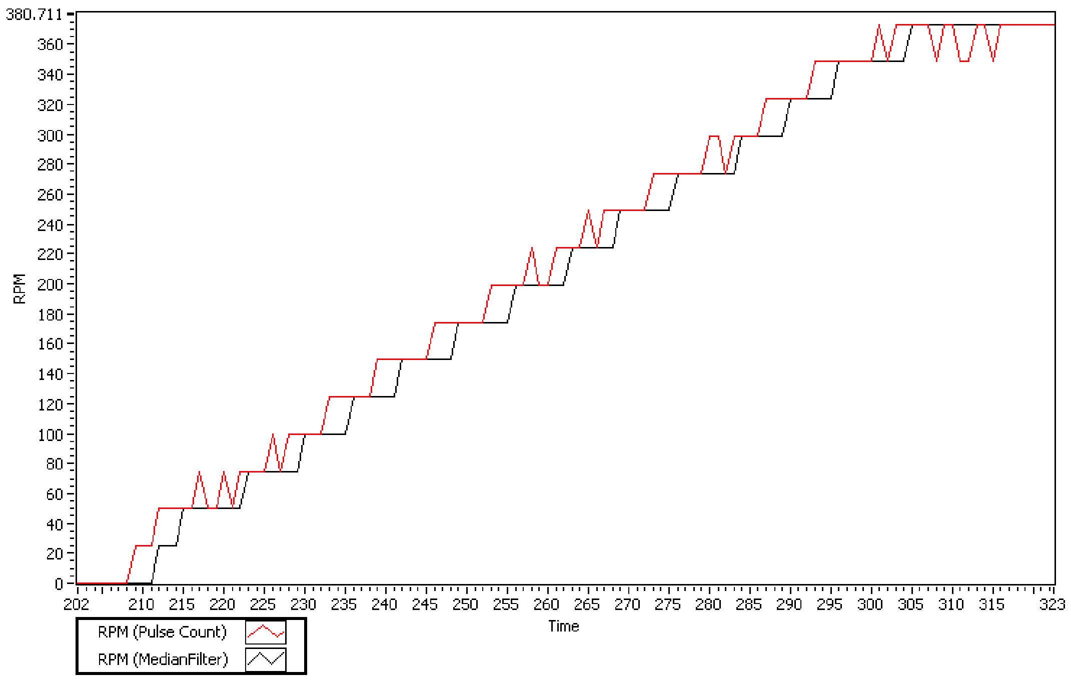


Figure 3-22: Delayed signal induced by the application of the median filter on the pulse count method measurement.

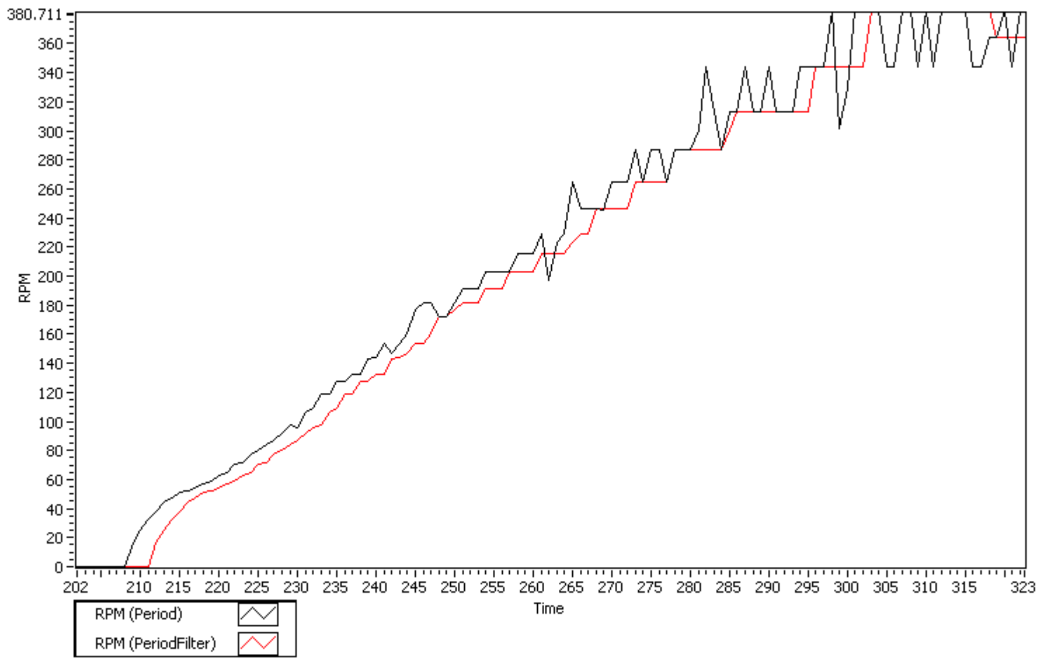


Figure 3-23: Delayed signal induced by the application of the median filter on the period method measurement.

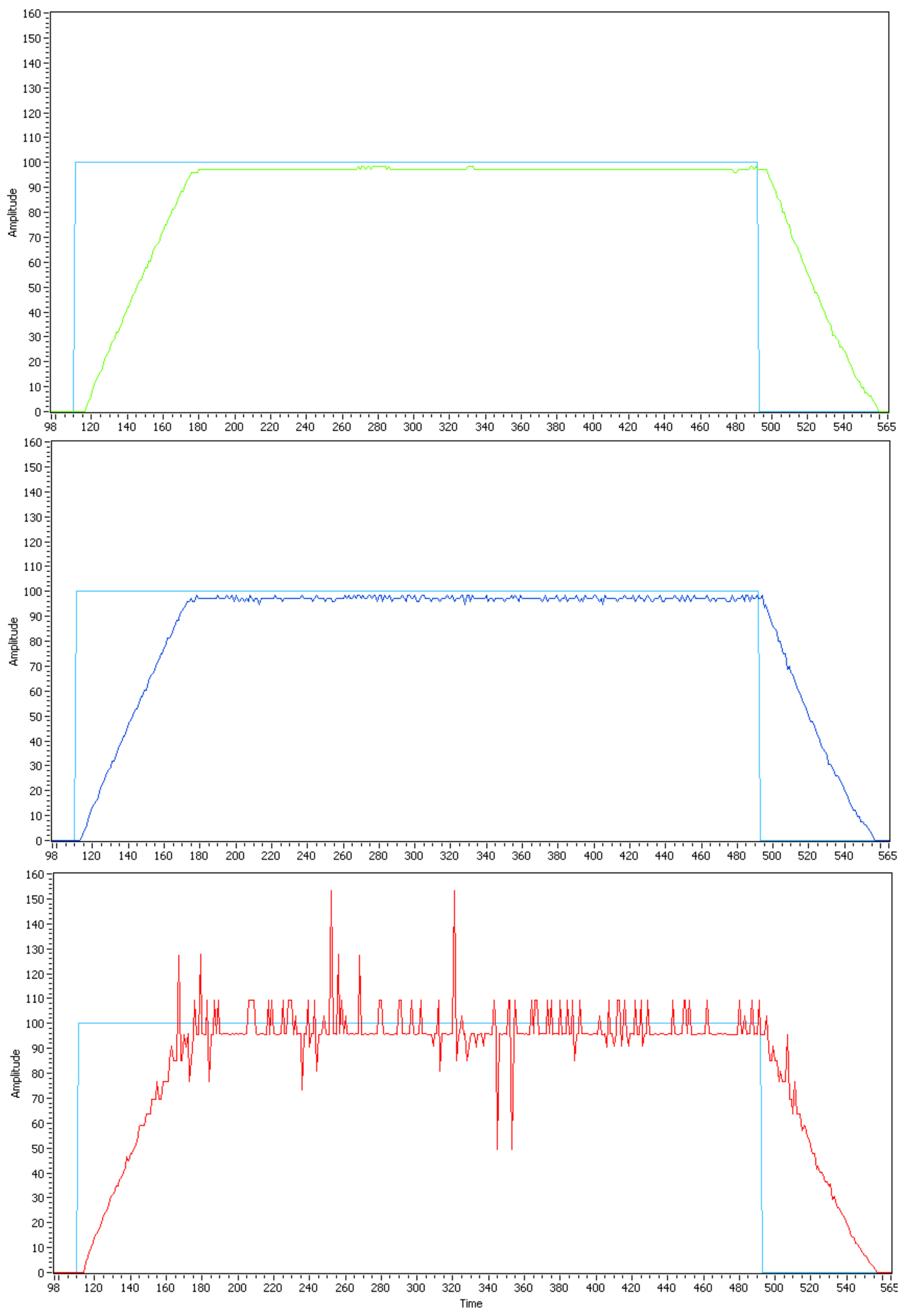


Figure 3-24: Systems response to a 100% change in speed demand. From top to bottom, median filtered pulse count method, pulse count method and period method.

## 3.7 Process Characterization

The parameters that describe a process are: steady state gain (K), which indicates how the process output changes with respect to an input change; the system's time constant ($\tau$), describing the rate of change of the process measurement; and the dead time ($t_0$) of the system, describing how much time is required for the system to start responding to an input change.

Characterization of the process plant was performed with the controller on manual mode (the controller's output didn't depend on feedback from the sensor) resulting in an open loop system.

The system was set to 50% of its maximum speed value; after stabilization, a step change from 50% to 100% was commanded. From this response curve, the process parameters were obtained. Calculations were done with numeric data logged by the cRIO rather than by a graphical method. Figure 3-25 shows the reaction curve.

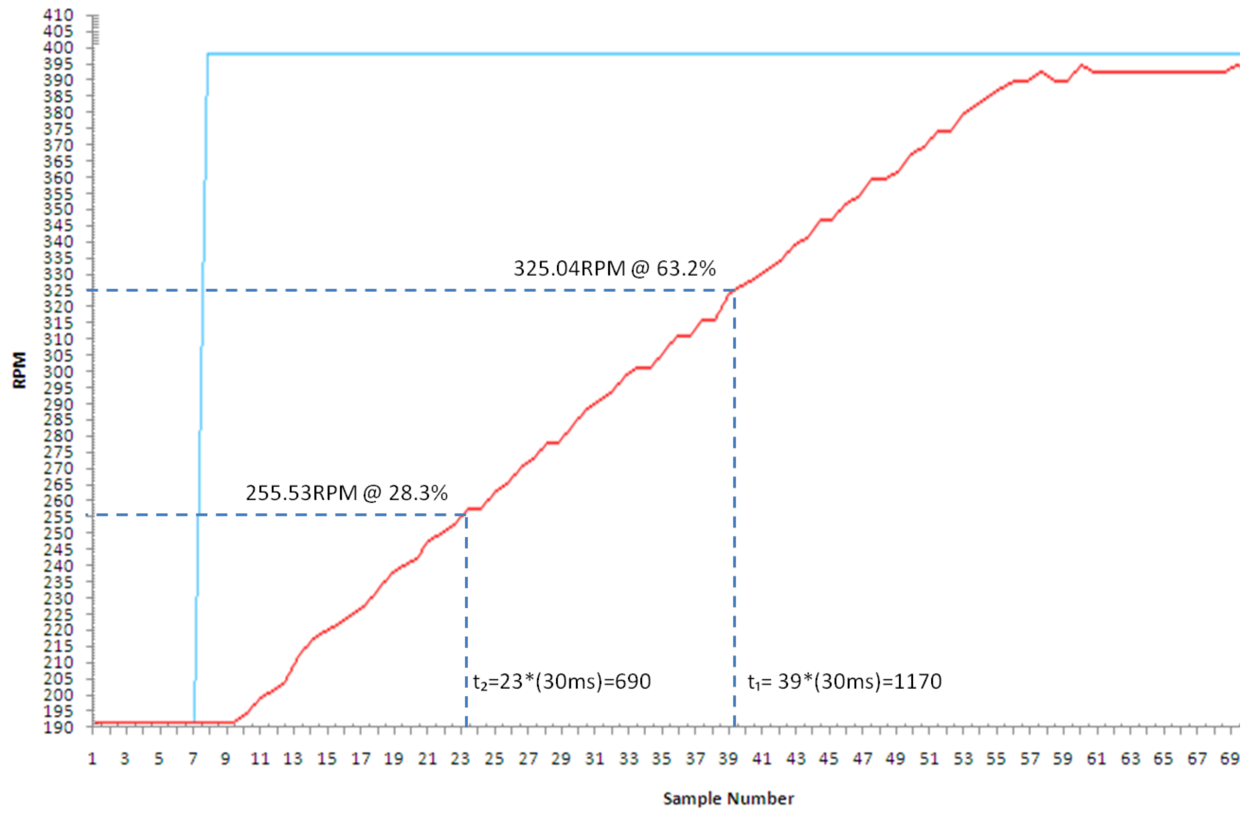


Figure 3-25: Response curve yielded by the system, used for characterization. Each sample marked on the horizontal axis corresponds to a 30 millisecond interval.

The steady state gain of the system is defined as:

$$K = \frac{\Delta Output}{\Delta Input} \tag{3.1}$$

Using the two point method, the time constant and dead time parameters of the process are obtained. In this method, time at which the process reaches 28.3% and 63.2% of its peak value is determined and used to establish the systems parameters. Equation (3.2) shows how the time constant is calculated while equation (3.3) shows the dead time approximation.

$$\tau = 1.5(t_1 - t_2) \tag{3.2}$$

$$t_0 = t_1 - \tau \tag{3.3}$$

From the resulting figure, the system can be characterized as a first order system with a dead time function. Table 3-3 shows the obtained parameters for the process: dead time ($t_0$), time constant ($\tau$) and process gain (K).

Table 3-3: Process parameters: dead time ($t_0$), time constant ($\tau$) and gain (K).

| $t_0$ (ms) | $\tau$ (ms) | $K$ |
|---|---|---|
| 450 | 720 | 2.553463 |
| minutes | | |
| 0.0075 | 0.012 | |

## 3.8 PID Control

A proportional-integral-derivative (PID) controller is implemented to develop a closed loop control system. The PID controller reads data from the process and computes the desired output by calculating and adding proportional, integral, and derivative responses.

A control action is defined based on tuning parameters and the error ($e$) measured between a setpoint ($SP$) and the process variable ($PV$) at a particular time step.

$$e = SP - PV \tag{3.4}$$

The controller's action $u(t)$, is described by:

$$u(t) = K_c\left(e(t) + \frac{1}{T_i}\int_0^t e(t)dt + T_d\frac{de(t)}{dt}\right) \tag{3.5}$$

Where $K_c$ is the proportional gain, $T_i$ is the integral or reset time and $T_d$ is the derivative or rate time. Each of these elements corresponds to a different response.

$$u_p(t) = K_c e(t) \tag{3.6}$$

$$u_I(t) = \frac{K_c}{T_i}\int_0^t edt \tag{3.7}$$

$$u_D(t) = K_c T_d \frac{de}{dt} \tag{3.8}$$

Where $u_p(t)$ is the proportional response, $u_I(t)$ is the integral response and $u_D(t)$ is the derivative response.

To develop a Network Controlled System (NCS) the PID controller is integrated into the CAN network. The system loop is closed by joining the controller, sensor, process plant and actuator through the network. A block diagram representing the resulting NCS is shown in figure 3-26.

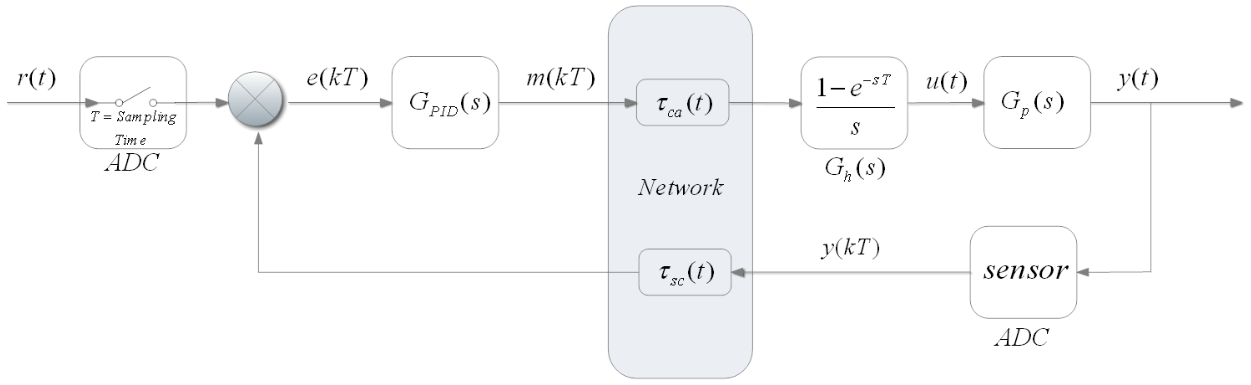


Figure 3-26: Block diagram of a typical NCS with induced delays.

The setpoint or reference ($r(t)$) is received by an analog to digital converter ($ADC$) where sampling occurs according to a time ($T$), then the discrete error ($e(kT)$) is computed and taken as an input on the controller ($G_{PID}(s)$), the controller's output or manipulated variable ($m(kT)$) is sent through the network where an induced controller-actuator delay ($\tau_{ca}(t)$) occurs, a zero order holder ($G_h(s)$) maintains the controller's output while the next output value is generated, finally, the process plant ($G_p(s)$) receives the controller's output as an input. The plant's output ($y(t)$) is registered by a sensor that sends a feedback signal ($y(kT)$) through the network which is affected by an induced sensor-controller delay ($\tau_{sc}(t)$). The error is computed again closing the control loop.

### 3.8.1 PID Tuning

To adjust the PID for correct performance, the controller requires characteristic parameters from the process plant. The controller's performance is directly affected by its tuning parameters. Different methods can be used to tune a controller. Three methods are presented for the PID controller: the Ziegler-Nichols method, the Integral of Time-weighted Absolute Error method, and the ultimate gain method.

The reaction curve method for a step change response described by Ziegler-Nichols (Z-N) [Zhong, 2006; Åström, 2002] was used as a model for obtaining the PID's tuning parameters [Apco]. Table 3-4 shows the controller parameters calculated using the Z-N method.

Table 3-4: PID tuning parameters: Controller gain ($K_c$), integral time ($T_i$) and derivative time $T_d$ for the Ziegler-Nichols response curve method. For P, PI and PID control strategies.

| Z-N | $K_c$ | $T_i$ | $T_d$ |
|---|---|---|---|
| P | 0.6266 | | |
| PI | 0.56394 | 0.025 | |
| PID | 0.676728 | 0.015 | 0.00375 |

Tuning was also done according to the Integral of Time-weighted Absolute Error (ITAE) criteria to compare its performance against the Z-N formulation. This criterion uses the integral of the error over time to penalize persistent errors according to equation (3.9):

$$\int_0^\infty t|e(t)|dt \tag{3.9}$$

ITAE tuning produces more conservative parameters than Z-N. Tuning can be optimized for performance against setpoint changes or against disturbances. Table 3-5 shows the ITAE criteria parameter results.

Table 3-5: ITAE tuning parameters calculated.

| ITAE | Reference Change | | | Disturbances | | |
|---|---|---|---|---|---|---|
| | $K_c$ | $T_i$ | $T_d$ | $K_c$ | $T_i$ | $T_d$ |
| P | | | | 0.40680 | | |
| PI | 0.43302 | 0.017942 | | 0.66217 | 0.01574304 | |
| PID | 0.68356 | 0.023529 | 0.002750 | 1.02453232 | 0.01210533 | 0.00324974 |

As a last implementation, the closed loop ultimate gain tuning method was also computed. This process determines the gain that will cause the system to oscillate at constant amplitude; the ultimate gain (where oscillations stop increasing in amplitude) and period are used to determine the tuning parameters. Table 3-6 presents the results.

Table 3-6: Ultimate gain PID tuning parameters.

| ¼ decay | $K_c$ | $T_i$ | $T_d$ |
|---|---|---|---|
| **P** | 1.25000 | | |
| **PI** | 1.12613 | 0.015493 | |
| **PID** | 1.49701 | 0.009333 | 0.00233 |

Each of the previously discussed methods was used as a guideline. The Z-N response curve method proved to yield the best results, but either way further manual fine tuning was done on the PID controller for this criterion to improve its performance.

## 3.9 Fuzzy Logic Control

Fuzzy systems (FS) are precise, defined systems, even though the processes they model are fuzzy in nature. Fuzzy logic tries to approximate real, complex non-linear real-world processes to estimate reasonable results.

Fuzzy logic is a knowledge-based or ruled-based decision making method. A fuzzy controller uses defined rules to control a process based on the values of its input variables. Linguistic variables (words) represent the input and output variables of the control system; these variables have a range of expected values. Linguistic terms are the different categories each linguistic variable can be classified into. A membership function relates linguistic terms with numerical values; these values represent the degree of membership each linguistic variable has with a linguistic term. Membership ranges from 0 for 0% to 1 for 100% membership. Each of these linguistic variables describes a fuzzy set. The relationship between input and output fuzzy sets can be described with fuzzy IF-THEN rules. A fuzzy system is constructed by a collection of fuzzy rules known as a rule base, which is equivalent to the control strategy of a controller. Basic connectives AND, OR, NOT are used to relate the rule's antecedents and determine the truth value of the aggregated antecedent; the THEN part constitutes the rule's consequent. For example a simple rule can be: IF speed slow AND accelerator not pressed THEN press accelerator.

When a process can be described qualitatively it is possible to use fuzzy logic to design a fuzzy controller. Fuzzification is the first stage of a fuzzy system, the crisp or numerical values of the variables involved are associated with its corresponding linguistic terms. After fuzzification is done the controller uses the input terms and a certain rule base to determine the proper value of the output linguistic variables. A defuzzification method is necessary to translate back the output's membership value into crisp numerical values.

### 3.9.1 Fuzzy Logic System

A Fuzzy Logic controller was developed to compare its performance against the PID. The first step in the development of the Fuzzy System, consisted in choosing input linguistic variables and its corresponding terms. Initially, the system was classified as single-input-single-output (SISO), where the input variable was the process variable (RPMs), and the output was a step-up or step-down on the CAN speed control value depending of the current RPMs. This resulted in a poor implementation; control of the vehicle's speed was easily lost.

A second approach consisted of a multiple-input-single-output (MISO) system with two input variables: first the current error of the system (measured between setpoint and process variable), and second, the error change rate. A step-up or step-down on the CAN speed control value remained as the output.

Initially, three linguistic terms, or classes were defined for each input variable: negative, zero and positive. Five and seven terms were also tested. After experimentation, better results were obtained with five terms. Seven terms didn't improve performance over five, thus, the FS was designed with the following five terms for each input variable: Negative Large, Negative, Zero, Positive, Positive Large.

The error is defined by equation (3.10), while the error change rate is defined as in equation (3.11).

$$e = SP - PV \tag{3.10}$$

$$\Delta e = e(t) - e(t-1) \tag{3.11}$$

Table 3-7 shows the relationship between error, error change rate and process variable; these relationships were used to define the Fuzzy System's rule base.

Table 3-7: Relationship between error, error change rate, and process variable.

| ***Error* $(e)$** | ***Error Change Rate* $(\Delta e)$** | ***Process Variable* $(PV)$** |
|---|---|---|
| **Positive** | Positive | Under the SP and moving away |
| **Positive** | Negative | Under the SP and moving towards |
| **Negative** | Negative | Over the SP and moving away |
| **Negative** | Positive | Over the SP and moving towards |
| **Zero** | Positive | Reached the SP from above |
| **Zero** | Negative | Reached the SP from below |
| **Positive** | Zero | Not moving and under the SP |
| **Negative** | Zero | Not moving and over the SP |
| **Zero** | Zero | Reached the SP and not moving |

The output linguistic variable, as already stated, represents increments on the CAN speed value sent to the Millipak node; seven linguistic terms were used to describe the output variable: Negative Large, Negative Medium, Negative Small, Zero, Positive Small, Positive Medium, Positive Large.

Each term in the input's membership function represents a possible current state of the process variable; for example, if both the error and error change rate are positive large, the process variable is very far below the desired speed and the vehicle is decelerating; the corresponding action by the controller would be to accelerate quickly, which translates to a positive large output, incrementing the CAN speed control value in more than one unit.

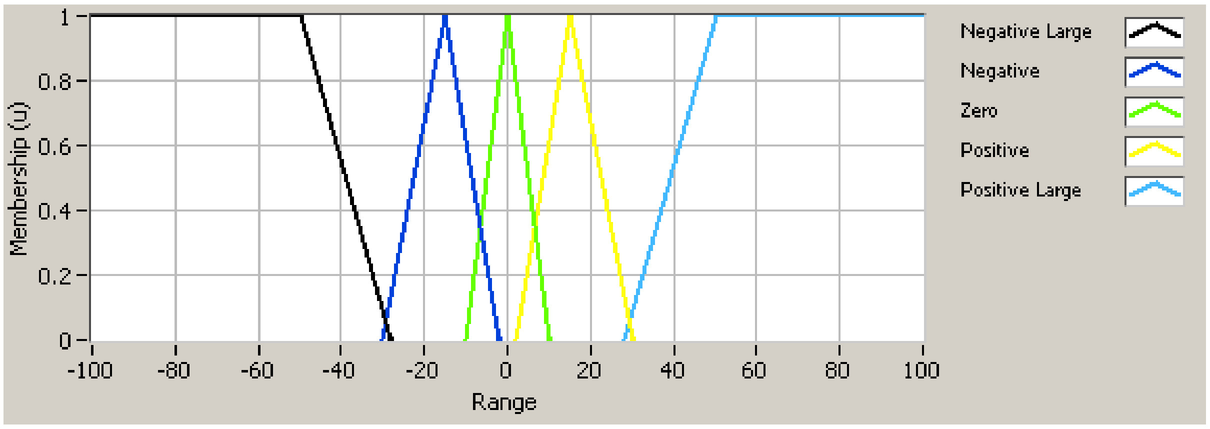


Figure 3-27: Membership function for the input variable Error. Error units are in RPMs. The zero error range is defined in conjunction with the Positive and Negative error range, membership varies depending on the three different linguistic terms.

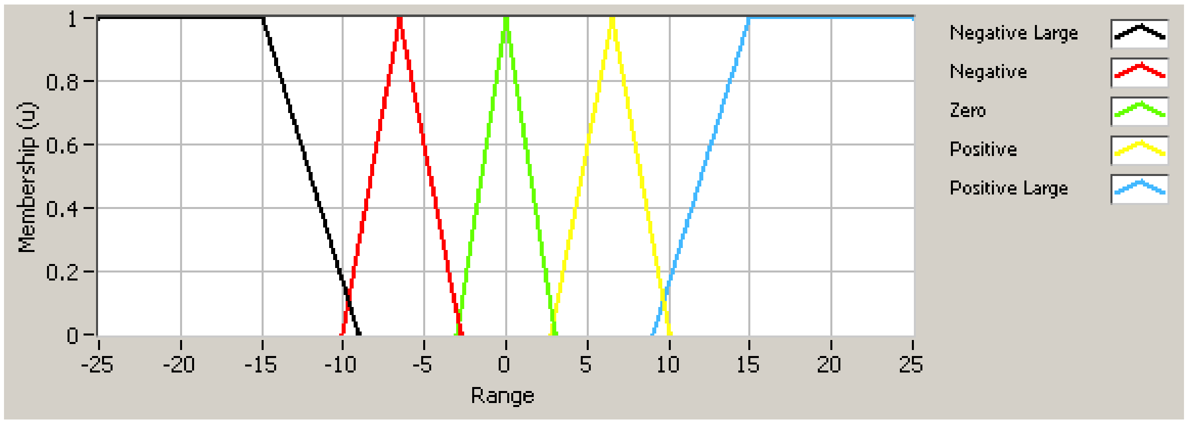


Figure 3-28: Membership function for the input variable Error Change Rate.

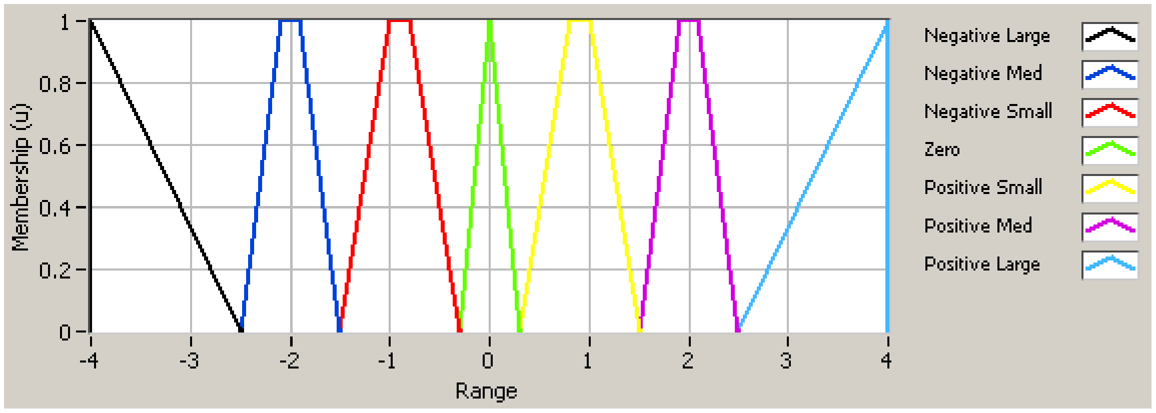


Figure 3-29: Membership function for the output variable CAN Data Changes.

All input antecedents for a given output are connected using the AND operator; the minimum of both membership degrees between the inputs is taken as the truth value for the aggregated rule antecedent.The rule base is composed of 25 different IF-THEN rules and it is summarized on table 3-8.

Table 3-8: Rule base used for the FS.[11]

| $\Delta e$ / *e* | **NL** | **N** | **Z** | **P** | **PL** |
|---|---|---|---|---|---|
| **NL** | NL | NL | NM | NM | Z |
| **N** | NL | NS | NS | NS | Z |
| **Z** | NL | Z | Z | Z | PL |
| **P** | Z | PS | PS | PS | PL |
| **PL** | Z | PM | PM | PL | PL |

[11] Negative Large: NL, Negative Medium: NM, Negative Small: NS, Negative: N, Zero: Z, Positive: P, Positive Small: PS, Positive Medium: PM, Positive Large: PL.

The resulting Fuzzy space is shown in figure 3-30. Input variables, Error and Error Change Rate are plotted versus the CAN speed control value increments which constitute the output variable.

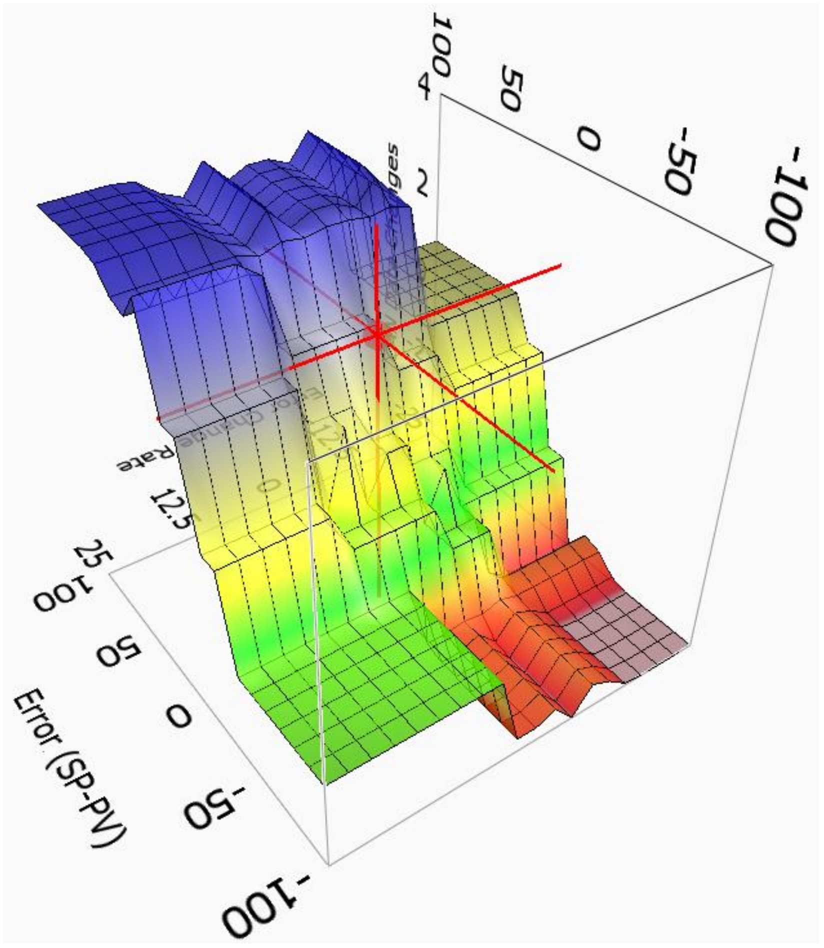


Figure 3-30: Fuzzy Space described by the rule base.

# Chapter 4
# Experiments and Results

This section presents the experiments done on the electric utilitarian vehicle to validate the development of the CAN control network. The tests depicted in this section were done with the complete network control system. This was done by integrating the control algorithms to the CAN network and monitoring the response of the system to different commands.

## 4.1 CompactRIO PID Control

The PID control algorithm implemented on the CompactRIO platform is located at the Real-Time programming level. An FPGA VI runs on the background to transmit/receive CAN messages. Information received through the CAN bus is shared with a Real-Time VI, which requests a subVI to calculate the RPMs of the wheel with the information provided through the network. On the RT VI the software filter is applied to the resulting RPM measurements to reduce outliers in the signal. The filtered signal serves as the PID controller's input, which based on the corresponding tuning parameters generates an output signal. The controller's output is sent through the CAN bus and received by the Millipak node. This node receives the CAN message and generates a control signal for the Millipak controller to drive the motor, trying to reach the desired speed value. The PID control algorithm employed works with percentage values for its input and output. Using the maximum RPMs (or a close approximation) of the system, the PID controller calculates the required output to maintain the speed of the vehicle close to the desired setpoint. A first attempt at controlling the system is shown in figure 4-1. It can be observed that tuning parameters must be adjusted to eliminate overshoot and oscillations in the controllers output.

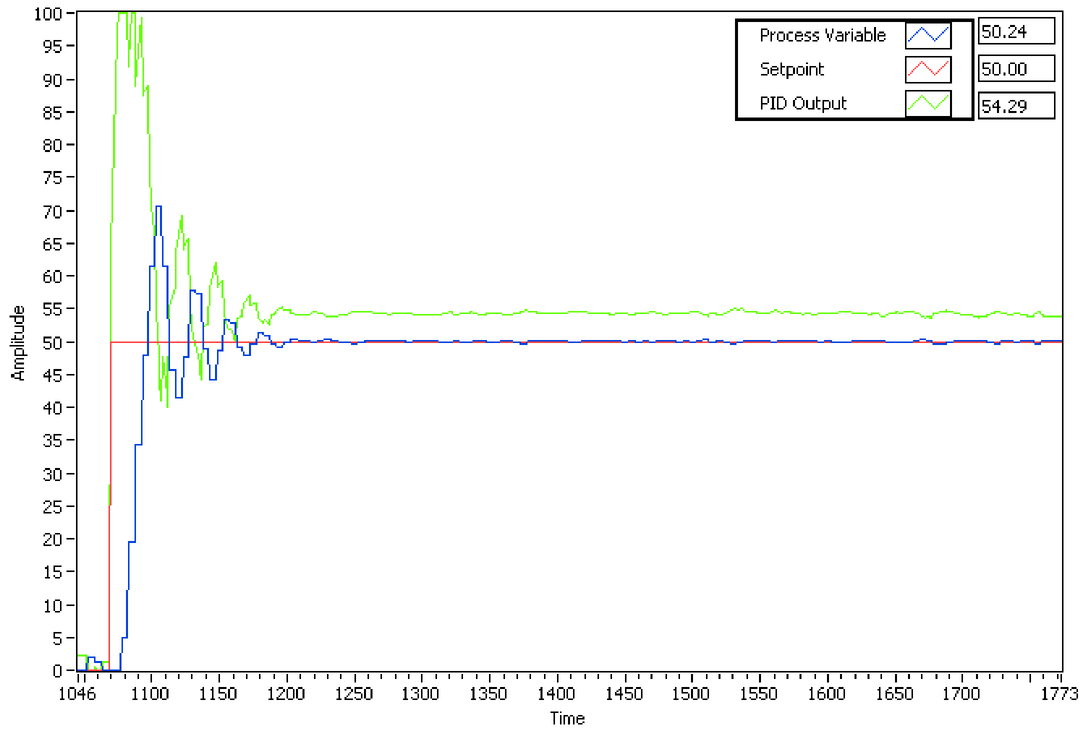


Figure 4-1: Closed Control Loop response with PID controller. Showing sample time (each time sample is equal to 50 milliseconds) on the horizontal axis versus percentage of maximum on the vertical axis.

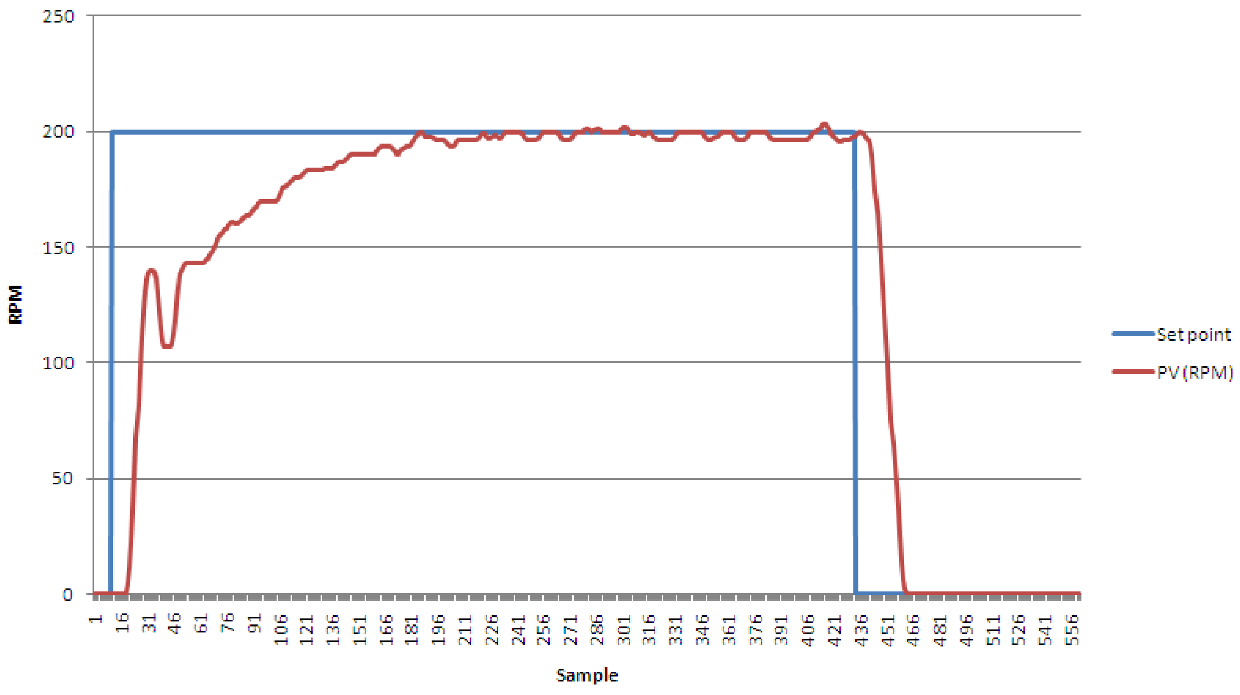


Figure 4-2: Closed Control Loop response for setpoint change with PID controller after Z-N tuning.

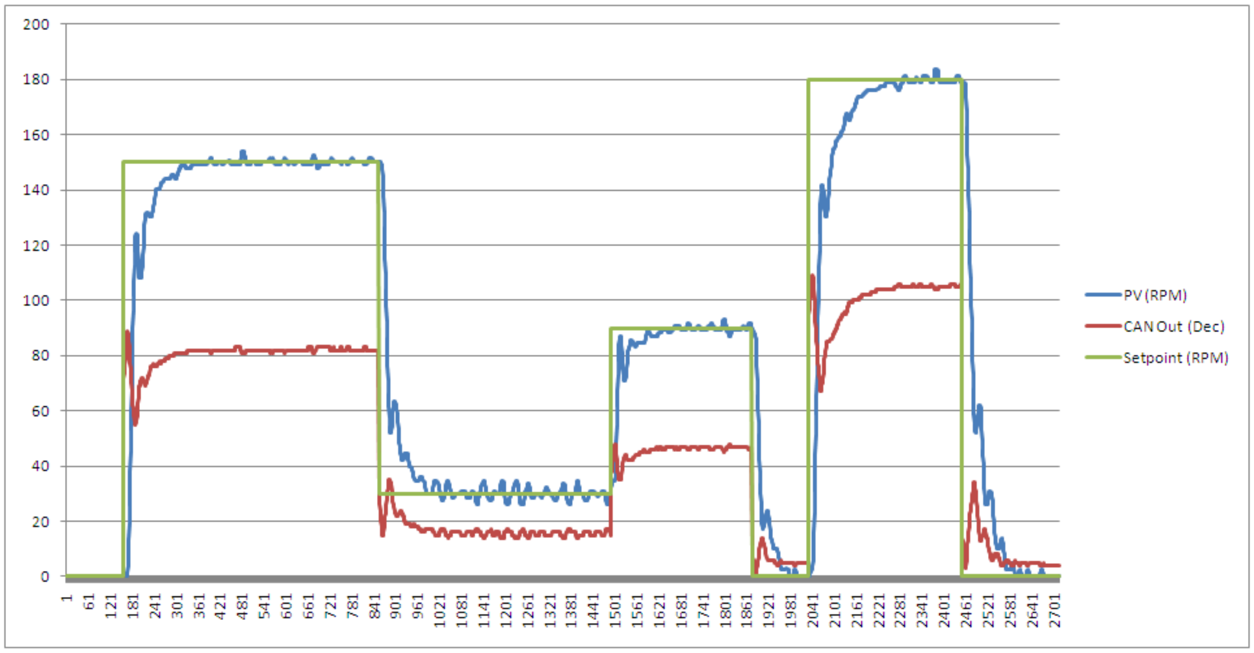


Figure 4-3: Closed Control Loop response for setpoint change with PID controller after fine tuning the Z-N method.

It can be observed that the PID controller indeed follows any setpoint change. Fine tuning helped speed up the response time for the controller to reach the setpoint. Further tuning can be done to shorten this response time span even more, watching carefully where the controller yields an overshoot.

## 4.2 CompactRIO Fuzzy Logic Control

Development of the FS in LabVIEW was done with the aid of the Fuzzy System Designer. The fuzzy system is saved as a *.fs file which is later transferred to the controller and loaded on the RT VI developed for fuzzy control. The *.fs file contains information about input/output variables membership functions and rules. The fuzzy logic controller VI constructed works at the RT level, like the PID controller receives information from the wheel's speed through the CAN bus.

It can be easily seen that the fuzzy controller takes more time to reach the setpoint than the PID, this generates less overshoot on the response as well as providing a smoother acceleration.

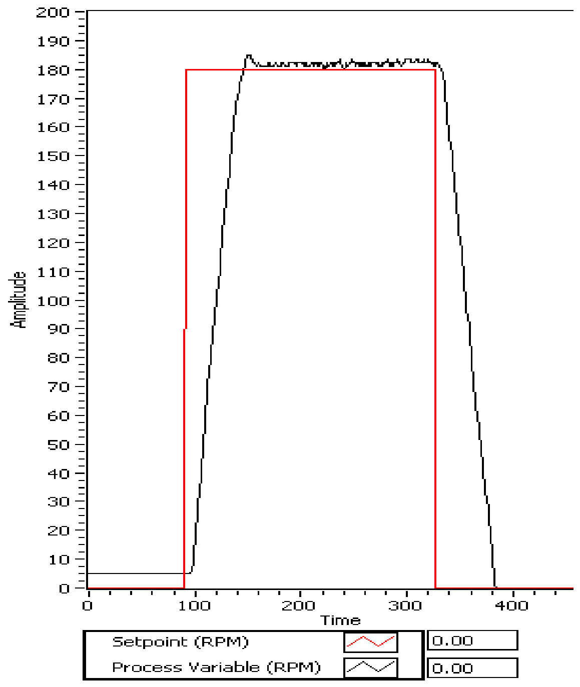


Figure 4-4: Closed-loop response with fuzzy logic controller.

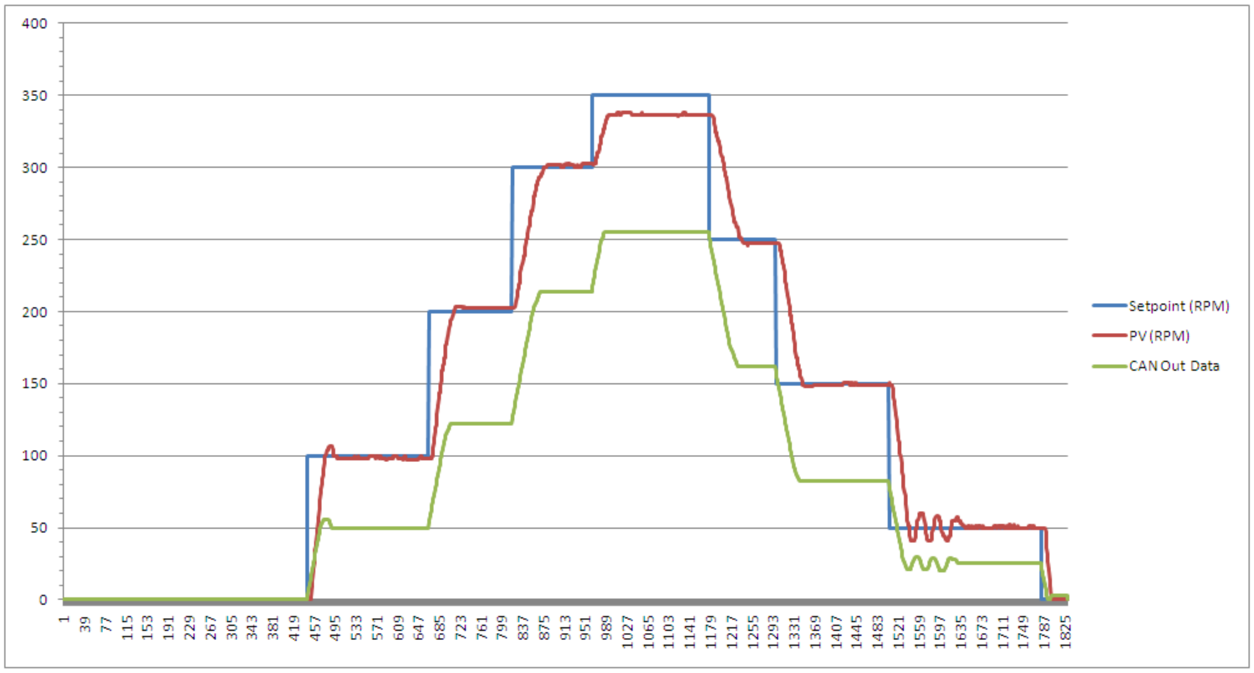


Figure 4-5: Closed-loop response with fuzzy logic for setpoint change. On the upper part of the graph a setpoint beyond the controllers output capacity was desired and thus not met.

# Chapter 5
# Conclusions

This thesis describes the automation of the speed control module for a real-life scale electric mining vehicle. The method for speed automation relies in the development of a Network Control System (NCS) based on the Controller Area Network (CAN) standard specification. This speed controller is proposed as part of a modular control architecture into which other modules can interact by means of the CAN bus.

The vehicle is the Super Truck model from Johnson Industries, whose kinematics are those of a rear-wheel differentially driven conventional vehicle. The driving force is produced by a 5HP electric motor, which is regulated by a power controller from Sevcon, model Millipak.

Since the utilitarian vehicle lacked appropriate preparations to develop an effectively closed control system, instrumentation and automation was required. A wheel speed sensor was developed to estimate the vehicles speed at all time. This information was then processed by the controller to generate an appropriate response, according to parameters defined by the user. The control algorithms reside on a Programmable Automation Controller (PAC); model CompactRIO, manufactured by National Instruments (NI). Both sensor and controller are connected through the CAN bus. On a personal computer, the user can modify controller parameters through a human-machine interface (HMI) developed on the graphical programming environment from NI: LabVIEW. Also, on the HMI, the controllers response and current speed of the vehicle can be monitored, as well as information travelling through the CAN bus. The purpose of the controller is to reach and keep the vehicle at a desired constant speed defined by the user.

The manufactured sensor offers an experimental low-cost solution for the intended application. It acts as an optical encoder to measure the revolutions per minute at which the traction wheels are turning. The sensor's number of pulses per revolution provides a good resolution (1.5° degrees per pulse) for RPM measurement of the vehicle's wheel.

Nevertheless, data quality of the signal produced by the sensor is acceptable; but problems reside on the asymmetry between pulse periods. These readings may be interpreted as false RPM changes. The measuring technique used counts pulses over a defined time span, therefore normalizing the duration of each pulse for such time span. With this technique the resulting signal from the sensor presents fewer variations during constant speed. Besides this, software filtering was applied to reduce outliers in the form of spikes, thus rendering a much smoother signal representing the speed of the wheel.

As a future direction, a commercially state of the art sensor can be used to replace the low-cost alternative used, improving setup times and properties such as robustness and response time frequency. Inclusion of the replacement sensor to the control network is easy, as long as the CAN frame sent by the sensor respects the convention established on this thesis for transmission of speed measurement data, no further modifications have to be made on any other CAN node.

Complementary to this, the establishment of the NCS required the development and implementation of the CAN bus on the vehicle. Specifically, the electronic design and later manufacturing of the modules and interfaces of the CAN nodes used to communicate both the speed sensor and Millipak controller over the network. Since a commercial CAN module from NI was available for the CompactRIO controller, its inclusion to the network was also possible. Programming of the microcontrollers on the manufactured CAN nodes as well as on the CompactRIO was done next. Although complications arose from the lack of information on the documentation provided for the CompactRIO's CAN module, they were ultimately resolved. This resulted in a complete and functional CAN bus.

The manufactured CAN modules and interfaces functioned as intended. CAN communication is effectively established and characteristics such as robustness and reliability of the protocol were verified by operating the network at maximum speed with constant message transfer, in a high electromagnetical environment. Electronic design of the PCBs for the modules and interfaces is simple and reliable, but it can yet be improved, resulting in a smaller size, a better form factor, and an increase in computational capacity. Further electronic protection for the manufactured modules and interfaces can be included to preserve their components in case of an electric malfunction.

Apart from this situation, closed loop control is achieved by connecting sensor, actuator and controller through the network. The sensor is periodically sending the wheels speed through the CAN bus; the controller receives this information and produces a control signal, which is also sent out to the network. The Millipak CAN node receives the information sent by the controller and generates a response that drives the electric motor to the desired speed, which is then again sensed and sent through the bus, closing the control loop. Two different control algorithms were implemented on the CompactRIO; a proportional–integral–derivative (PID) controller and fuzzy logic controller, both developed under LabVIEW. As with most control systems, a trade-off between response time and stability was naturally present.

The PID responded well with short sampling time spans (less than 100 milliseconds) on the controller, but stress applied to the electric motor by the controller's output response was far greater than with the fuzzy logic controller. Since the response time of the system is slow in nature, sampling time spans on the controller can be increased without compromising performance, thus the fuzzy logic controller proves to be in advantage over the PID. With a slower sampling time (130 milliseconds), the fuzzy controller performs as well as the PID, but provides a better alternative in terms of setup time, design and implementation.

Improvements to both control approaches can be made. In the case of the PID, advanced tuning techniques may be applied to provide more accurate control. For the case of the fuzzy logic controller, the number of rules as well as their operating range can be modified to yield better performance. A further approach can be the fusion of both techniques to develop a hybrid fuzzy-PID controller.

During experimental validation, a variety of laboratory exercises with an experimental setup, where the vehicle was suspended on its traction wheels. Experiments were done to determine characteristics such as repeatability of the developed system. Also, several tests were successfully performed outside the laboratory, demonstrating a complete and functional speed control network based on the CAN protocol. With the vehicle rolling on the ground, the CAN network provided a communication channel between the Millipak controller, the wheel speed sensor and the PAC, where both control schemes implemented achieved the desired task of maintaining a constant cruise speed for the vehicle. With this, it was demonstrated that the CAN bus control network developed is suited for management of critical control information, not only limited to the one provided by the speed controller developed on this research.

Unfortunately, the vehicles batteries are not in perfect shape, and usually are drained quickly, affecting performance (e.g. the vehicles maximum speed may be achieved while the batteries are at full charge, but once they deplete below 50% max speed cannot be reached), forcing the controller to yield a response of 100% of its capacity constantly. Higher capacity batteries are required to improve the overall performance of the vehicle and extend testing periods.

Finally, it can be concluded that the methodology applied effectively resulted in a prototype vehicle with autonomous speed control. The resulting Network Control System allows for further inclusion of other control modules. The robustness of the system permits the removal of current modules and insertion of new ones without sacrificing the existing functionality of the control architecture. The use of standardized components grants the system with an interoperability characteristic, thus the system is not brand-specific. In this manner, the proposed CAN network can be expanded to a complete control architecture to successfully build a completely autonomous vehicle. The speed algorithms employed can be modified or even replaced by others, any type of controller can be included without mayor modifications on the system, as long as the controller's output travels through the network with the proper structure. A key advantage of adopting a CAN bus control network is the possible inclusion of nodes without disruption of the whole network; which means that development can grow in complexity without much redesign, thus the network proposed is not limited to the nodes presented.

The vehicle has yet to be equipped with sensors that allow environmental perception, such as sonars, LIDARs (Light Detection and Ranging) or different types of cameras. An improvement to the current implementation would be the use of a quadrature signal, to receive feedback on the moving direction of the vehicle. An actuator mechanism for autonomous steering control is already developed but has not been fully implemented. This elements can be easily adapted to the work completed on this project by means of the CAN bus.

# Appendix A
# Kinematic Model

A simplified bicycle model like the one shown in figure A-1 can be generated from the Ackerman Steering geometry and used to compute the vehicles kinematic model.

Nomenclature is as follows:

$V_{2e}$: Velocity for the frontal external wheel.

$V_2$: Velocity for the frontal virtual bicycle wheel.

$V_{2i}$: Velocity for the frontal internal wheel.

$V_{1e}$: Velocity for the rear external wheel.

$V_1$: Velocity for the rear virtual bicycle wheel.

$V_{1i}$: Velocity for the rear internal wheel.

$\phi_{2e}$: Directional angle for the frontal external wheel.

$\phi_2$: Directional angle for the frontal virtual bicycle wheel.

$\phi_{2i}$: Directional angle for the frontal internal wheel.

L: Length from the origin of $\{x_m, y_m\}$ to the center of the frontal virtual bicycle wheel.

C: Width of the vehicle.

ICR: Instantaneous center of rotation.

R: Distance from the origin of $\{x_m, y_m\}$to the ICR.

$R_m$: Distance from the frontal virtual (middle) bicycle wheel to the ICR.

$R_e$: Distance from the frontal external wheel to the ICR.

$R_i$: Distance from the frontal internal wheel to the ICR.

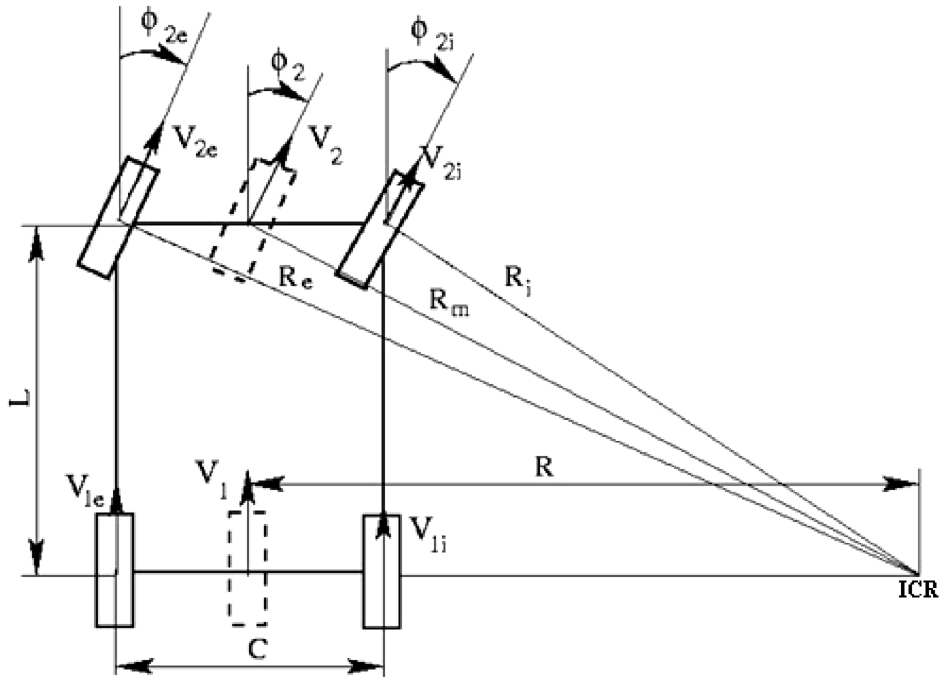


Figure A-1: Bicycle model.

The wheels can be characterized by the constants $\{L, \alpha, r\}$ and the time-varying angles $\phi(t), \varphi(t)$ [Habumuremyi, 2005]. The radius $\{r\}$ of the wheel and its time varying angular position $\varphi(t)$ are used to calculate the velocity $V(t) = r\varphi(t)$ of the wheel.

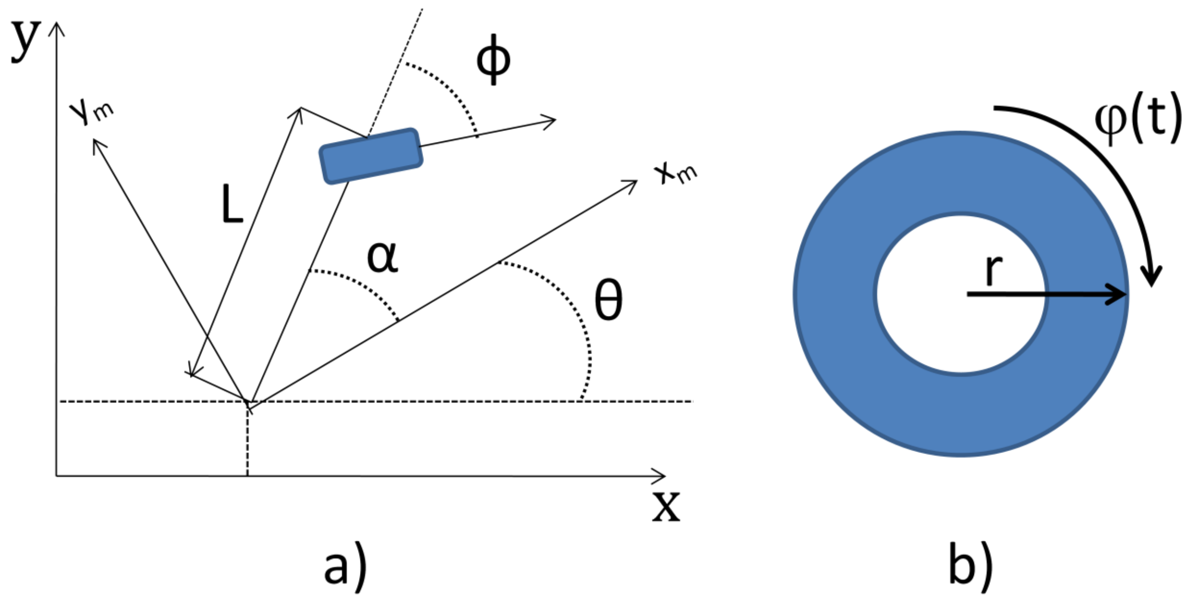


Figure A-2: a) Bicycle model for wheel description b) wheels lateral view.

Two constraints are deducted from these descriptions [Habumuremyi, 2005; Campion et al., 1996]:

Pure rolling constraint:

$$\cos(\alpha+\theta+\phi)\,\dot{x}+\sin(\alpha+\theta+\phi)\dot{y}+l\sin\phi\,\dot{\theta}-V=0 \tag{A.1}$$

Non-slipping constraint:

$$\sin(\alpha+\theta+\phi)\,\dot{x}-\cos(\alpha+\theta+\phi)\dot{y}-l\cos\phi\,\dot{\theta}=0 \tag{A.2}$$

If the wheels are centered with the $x_m$ axis, then $\alpha = 0$ and $\{\phi\}$ is measured with respect to $\{\theta\}$ like stated earlier; so the term can be eliminated from the previous constraints. From the bicycle approach presented in figure A-1 the following equations governing the kinematic model are constructed:

$$\mathrm{V}_{2\mathrm{i}}=\mathrm{V}_2\frac{\cos\phi_2}{\cos\phi_{2i}} \tag{A.3}$$

$$\mathrm{V}_{2\mathrm{e}}=\mathrm{V}_2\frac{\cos\phi_2}{\cos\phi_{2e}} \tag{A.4}$$

$$\mathrm{V}_{1\mathrm{i}}=\mathrm{V}_1\left(1-\frac{\mathrm{C}}{2\mathrm{R}}\right) \tag{A.5}$$

$$\mathrm{V}_{1\mathrm{e}}=\mathrm{V}_1\left(1+\frac{\mathrm{C}}{2\mathrm{R}}\right) \tag{A.6}$$

Finally the Ackerman condition [Habumuremyi, 2005; Xiao, 2008]:

$$\cot\phi_2=\cot\phi_{2e}-\frac{C}{2L}=\cot\phi_{2i}+\frac{C}{2L} \tag{A.7}$$

For a rear traction vehicle like the one on this project an exact relation for the angles and speeds of the real wheels can be obtained from equations (A.3) to (A.7). Based on figure A-1 further constraints can be established from (1) and (2).

Pure rolling constraint of the rear wheel ($l = 0, \alpha = 0, \phi = 0$):

$$\cos(\theta)\,\dot{x}+\sin(\theta)\dot{y}=V_1 \tag{A.8}$$

Non-slipping constraint of the rear wheel ($l = 0, \alpha = 0, \phi = 0$) :

$$\sin(\theta)\,\dot{x}-\cos(\theta)\dot{y}=0 \tag{A.9}$$

Non-slipping constraint of the front wheel ($\alpha = 0$):

$$\sin(\theta+\phi_2)\,\dot{x}-\cos(\theta+\phi_2)\dot{y}-L\cos\phi_2\,\dot{\theta}=0 \tag{A.10}$$

Solving (A.8) to (A.10) we get:

$$\dot{x}=V_1\cos(\theta) \tag{A.11}$$

$$\dot{y} = V_1 \sin(\theta) \tag{A.12}$$

$$\dot{\theta} = V_1 \frac{\tan \phi_2}{L} \tag{A.13}$$

Written as a matrix:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi}_2 \end{bmatrix} = \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \\ \frac{\tan \phi_2}{L} \\ 0 \end{bmatrix} V_1 \tag{A.13}$$

# Appendix B
# Motor Model

In reference to figure 3-6, the torque ($T_d$) produced by the motor is proportional to the product between the armature current ($I_a$) and the flux (which is proportional to the field current) [Zaccarian].

$$T_d = K_2 I_a \tag{B.1}$$

Where $K_2$ is the torque constant of the motor; with constant velocity, $K_2$ is dependent on the flux density of the fixed magnets in the motor, the reluctance of the iron core and number of turns in the armature winding. An induced voltage proportional to the product of the flux and the angular velocity is developed while the armature is rotating; this voltage, called the back emf ($E_g$) or the speed voltage [Salam, 2003; Zaccarian]:

$$E_g = K_3 \omega \tag{B.2}$$

Where $K_3$ is the motor voltage constant (in V/A – rad/s). Using Kirchoffs voltage law on the armature circuit we obtain:

$$R_a I_a + L_a \frac{dI_a}{dt} + E_g = V_a \tag{B.3}$$

Performing an energy balance on the motor, the sum of the torques should also equal cero. That is, the electromagnetic torque of the motor due to the voltages applied on the rotor and stator should compensate the torque due to rotational acceleration of the rotor, characterized by its inertia $J$ and the torque due to the velocity of the rotor characterized by the viscous friction coefficient $B$ as well as the torque $T_L$ developed by the load.

$$T_d = K_2 I_a - T_L = J \frac{d\omega}{dt} + B\omega \tag{B.4}$$

With initial conditions equal to cero and taking the Laplace transforms of the previous equations [Gonzalez, 2004]

$$E_g(s) = K_3 \omega(s) \tag{B.5}$$

$$(R_a + L_a s) I_a(s) + K_3 \omega(s) = V_a(s) \tag{B.6}$$

$$T_d(s) = K_2 I_a(s) \tag{B.7}$$

$$K_2 I_a(s) = (Js + B)\omega(s) \tag{B.8}$$

Substituting $I_a(s)$ of (B.6) with (B.8) we can obtain the transfer function between the angular speed $\{\omega\}$ of the motor's shaft and the input voltage $\{V_a\}$ as shown next:

$$\left[\frac{(R_a + L_a s)(Js + B)}{K_2} + K_3\right]\omega(s) = V_a(s)$$

$$\frac{\omega(s)}{V_a(s)} = \frac{1}{\left[\frac{(R_a + L_a s)(Js + B)}{K_2} + K_3\right]}$$

$$\frac{\omega(s)}{V_a(s)} = \frac{K_2}{[(R_a + L_a s)(Js + B) + K_2 K_3]}$$

$$\frac{\omega(s)}{V_a(s)} = \frac{K_2}{[R_a Js + R_a B + L_a Js^2 + L_a Bs + K_2 K_3]} \tag{B.9}$$

Assuming the armature inductance is very small $\{L_a \approx 0\}$ and simplifying by dividing by $R_a$ we finally obtain:

$$\frac{\omega(s)}{V_a(s)} = \frac{\frac{K_2}{R_a}}{\left[Js + B + \frac{K_2 K_3}{R_a}\right]} \tag{B.10}$$

We can model (B.10) as a first order system by taking [Palacios, 2000]:

$$K = \frac{K_2}{R_a} \ \textit{as the gain of the system}$$

$$T_m = \frac{R_a J}{BR_a + K_2 K_3} \ \textit{as the time constant of the system}$$

The transfer function for the electric motor is:

$$\frac{\omega(s)}{V_a(s)} = \frac{K}{T_m s + 1} \tag{B.11}$$

The differential placed at the end of the motor's shaft alters equation (B.4). The inertia moment, viscous friction and torque provided by the differential should be included in the $J$, $B$ and $T_L$ terms that appear on this equation. But since all the missing elements involved are constants, the magnitude of the order of the system is preserved.

# Appendix C
# Millipak Controller

The Millipak is the electric controller for the vehicle; it serves as a power interface to drive the electric motor. A schematic representation of the Millipak is shown below in figure C-1.

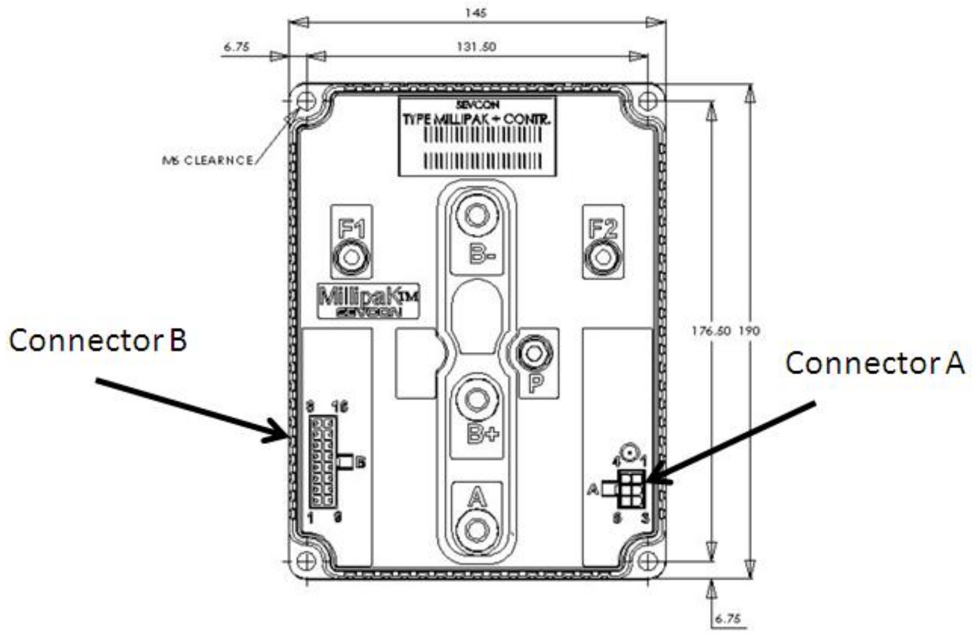


Figure C-1: Millipak controller showing terminals and connectors A and B.

The following figures show the wiring connections required to operate the Millipak controller.

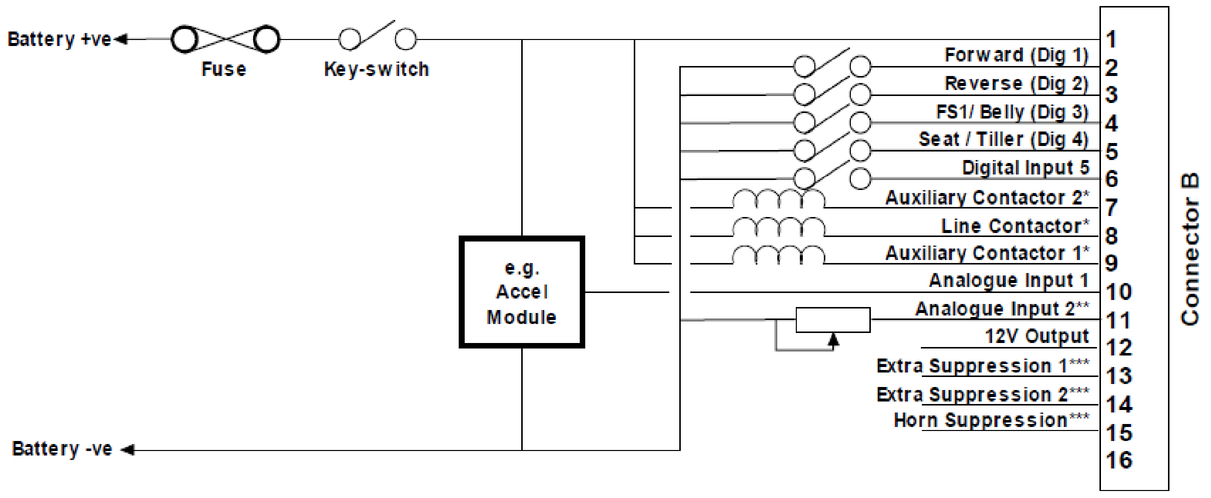


Figure C-2: Pin out for connector B.

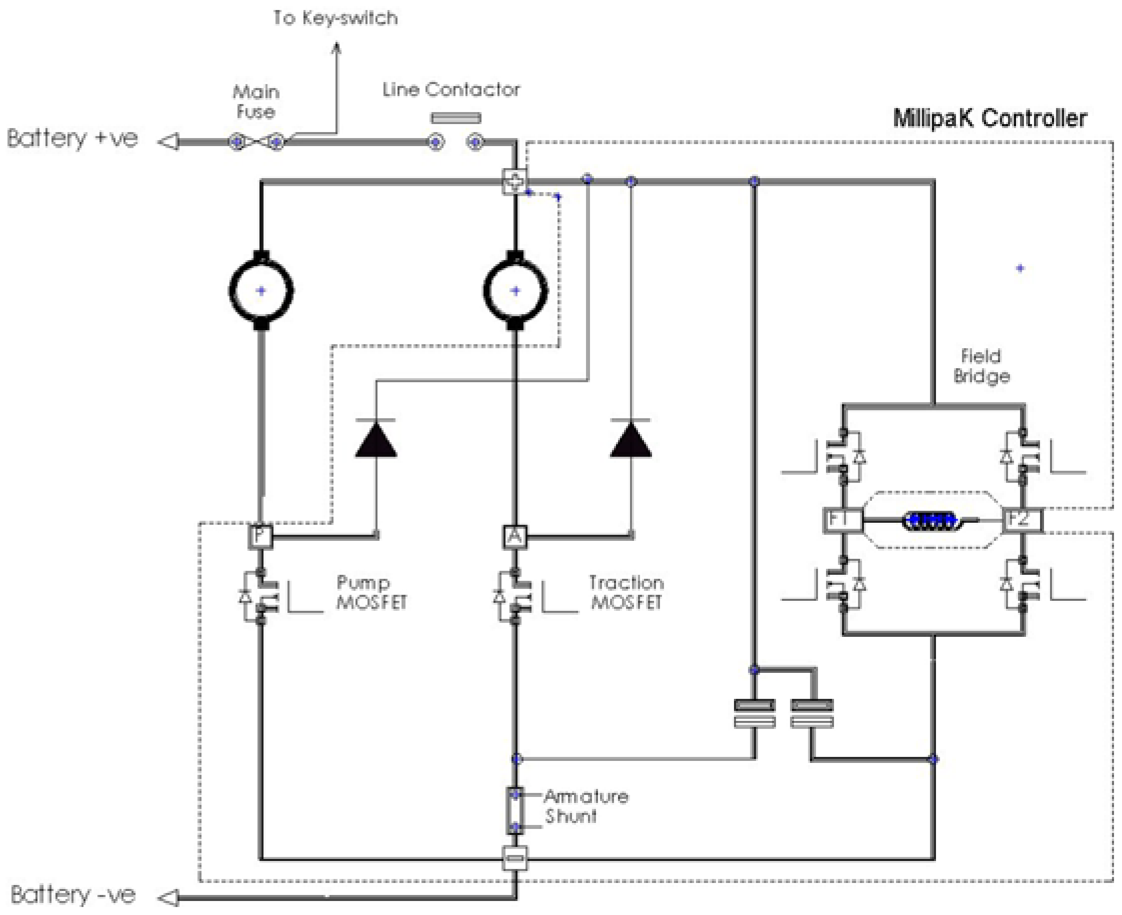


Figure C-3: Millipak controller power wiring.

The calibrator uses a reference shown as xx.xx to display the value of important items. Table C-1 shows the values for every item programmed on the Millipak. For example, calibrator reference 0.01, which refers to the armature current limit, has a value of 600 A, the maximum supported by this model, calibrator reference 2.01 mandates an acceleration delay of 1.5 seconds for the controller to go from 0% to 100% speed.

Table C-1: Calibrator personality list, showing the initial, maximum and minimum values for each property.

| | | Personality | | Actual Value | Range | |
|---|---|---|---|---|---|---|
| | | | | | Minimum | Maximum |
| Current Limits | 0 | 0.01 | Armature Current Limit | 600 | 50A | ABR |
| | | 0.02 | Field Current Limit | 50 | 10A | FBR |
| | | 0.03 | Drive Current Limit Start | 50 | 50A | Armature I Limit |
| | | 0.04 | Drive Current Limit Ramp | 0.25 | 0.00s | 2.50s |
| Braking Levels | 1 | 1.01 | Direction Change Braking | 50 | 5% | 100% |
| | | 1.02 | Neutral Braking Level | 10 | 0% | 100% |
| | | 1.03 | Foot-brake Braking Level | 0 | 0% | 100% |
| | | 1.04 | Dir. Braking Exit Level | 0 | 0% | 100% |
| Accelerator | 2 | 2.01 | Acceleration Delay | 1.5 | 0.1s | 5.0s |
| | | 2.02 | Deceleration Delay | 0.1 | 0.1s | 0.5s |
| | | 2.03 | Accelerator Zero Voltage | 0.2 | 0.00V | 4.50V |
| | | 2.04 | Accelerator Full Voltage | 3.5 | 0.00V | 4.50V |
| Creep Speed | 3 | 3.01 | Creep Speed | 0 | 0% | 25% |
| Max Speed | 5 | 5.01 | Maximum Speed | 100 | 0% | 100% |
| | | 5.02 | Maximum Reverse Speed | 100 | 0% | 100% |
| Cutback 1 Speed | 6 | 6.01 | Cutback Speed 1 | 100 | 0% | 100% |
| Cutback 2 Speed | 7 | 7.01 | Cutback Speed 2 | 100 | 0% | 100% |
| Motor Setup | 8 | 8.01 | Field Current Low | 7 | 2.00A | If Mid or 19.75A |
| | | 8.02 | Arm Current Low | 120 | 10A | Arm I Mid – 10A |
| | | 8.03 | Field Current Mid | 25 | Field I Low | Field I High |
| | | 8.04 | Arm Current Mid | 200 | Arm I Low + 10A | Arm I High – 10A |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | 8.05 | Field Current High | 50 | Field I Mid | FBR |
| | | 8.06 | Arm Current High | 400 | Arm I Mid + 10A | ABR |
| Power Steer Timer | 9 | 9.01 | Power Steer Timer | 2 | 0s | 60s |
| Seat Delay | 10 | 10.01 | Seat Delay | 1.5 | 0.1s | 5.0s |
| Addiotional Personalities | 11 | 11.01 | Roll-Off Speed | 73 | 0% | 100% |
| | | 11.02 | Walk Speed | 0 | 0% | 100% |
| | | 11.03 | Belly Delay | 1 | 0.1s | 5.0s |
| | | 11.04 | PSS Ramp Up Delay | 1 | 0.1s | 1.0s |
| | | 11.05 | PSS Timer | 2 | 0s | 10s |
| | | 11.06 | Line Cont. Dropout Timer | 1 | 1s | 60s |
| | | 11.07 | SRO Delay | 0 | 0s | 5s |
| | | 11.08 | Low Voltage Start | 27 | Low V Cutout | System Voltage |
| | | 11.09 | Low Voltage Cutout | 25 | 14.5V | Low V Start |
| | | 11.1 | High Voltage Start | 45 | System Voltage | High V Cutout |
| | | 11.11 | High Voltage Cutout | 50 | High V Start | 50.0V or 58.0V |
| | | 11.12 | Foot Brake Zero Volts | 0 | 0.00V | 4.50V |
| | | 11.13 | Foot Brake Full Volts | 4.5 | 0.00V | 4.50V |
| System Setup | 12 | 12.01 | Chop Select | OFF | OFF/ON/24V | |
| | | 12.02 | SRO Enable | ON | OFF/ON | |
| | | 12.03 | Control Mode | TORQUE | TORQUE/SPEED | |
| | | 12.04 | Seat Cuts Pump | OFF | OFF/ON | |
| | | 12.05 | Roll-Off Electro brake | OFF | OFF/ON | |
| | | 12.06 | Reverse Speed Limit | OFF | OFF/ON | |
| | | 12.07 | FS1 Recycle on Dir Change | ON | OFF/ON | |
| | | 12.08 | Dir Switch Seq Fault Check | ON | OFF/ON | |
| | | 12.09 | Line Contactor Dropout | OFF | OFF/ON | |
| | | 12.1 | Accelerator Characteristics | LINEAR | LINEAR CURVED 2*SLOPE CRAWL | |
| | | 12.11 | High Speed Mode | NORMAL | NORMAL LATCHED | |
| | | 12.12 | Belly Style | NORMAL | NORMAL CONTINUOUS | |
| | | 12.13 | Power Steer Trigger | 0 | FS1 | |

| | | | | | |
|---|---|---|---|---|---|
| | | | | SEAT DIRECTION | |
| | 12.14 | Foot-brake Priority | DRIVE | DRIVE FOOTBRAKE | |
| | 12.15 | Buzzer Configuration | OFF | OFF REV+ROLL ALL | |
| | 12.16 | Digital I/O | 7 | 1 | 17 |
| | 12.17 | Analogue I/P | 3 | 1 | 4 |
| | 12.18 | System Voltage | 36 | 24V | 48V |

*Acceleration and Braking*

The accelerator/analogue inputs are flexible in the range of the signal sources they can accommodate. Each analogue input has 2 adjustments associated with it to allow the input voltage range to be determined. The 2 adjustments are called the "Accelerator Zero Level" and the "Accelerator Full Level". If these were set to 0.20V and 4.80V then 0% pulsing would start at 0.20V at the input, increasing to 100% pulsing at 4.80V; the inverse occurs with decreasing voltage outputs. As shown in table C-1, the accelerator zero voltage (reference 2.03) is set to 0.2V this means the vehicle will start accelerating when 0.2V are present on the analog input, increasing in step sizes of 0.02V until it reaches its maximum acceleration rate at 4.5V. Also the reader may note that reference 2.01, acceleration delay is set to 1.5 seconds, this is customized for a smooth acceleration in order to ramp the accelerating pulse from 0% to 100%.

The acceleration characteristics personality defines a curve to be followed, either linear, curved, dual slope or crawl. Currently selection is to linear. This function is used to vary how much speed is demanded depending on the accelerator position. Depending on the setting, the controller gives a smaller change in speed for large changes in accelerator position which is useful for low speed maneuvering. The accelerator push refers to how much the operator has the accelerator depressed. The accelerator demand refers to how much accelerator demand is requested after the characteristic function is applied. This accelerator demand is then used along with the creep speed and maximum speed personalities to determine the speed demand for the vehicle. If a valid direction is selected and the accelerator demand is at 0%, the speed demand will be set to the creep speed personality. As the accelerator demand is increased to 100%, the speed demand increases linearly to the maximum speed personality.

Braking can be initiated in one of 3 ways:

- Direction Braking: Initiated when the direction switch inputs are reversed during drive.
- Footbrake Braking: Initiated when the operator depresses the footbrake pedal and a footbrake input is configured.

- Neutral braking: Initiated when the vehicle is put into neutral during drive and the neutral braking level is greater than 0%.

Direction and neutral braking are currently applied; no footbrake preference has been established; although it offers a great possibility for automation because footbrake can be configured to follow one of the next options:

- Via an analogue input configured as a Footbrake Pot. Using a potentiometer allows the operator to vary the amount of braking needed.
- Via a digital input configured as a Footbrake switch. When the switch is active, the system will brake at the footbrake level established on the personality list.

# Appendix D
# Wheel Speed Sensor

Two different models of opto-interrupters were used to construct the sensor; the ITR8102 model from Everlight Electronics and the EE-SX670 model from Omron.

The first version of the speed sensor was mounted on the rear left wheel; it was made using the ITR8102 opto-interrupters and a perforated metal strip with 48 circular holes as shown in figure D-1. Originally the opto-interrupters were detached from their plastic casing because the perforations on the metal strip wouldn't reach the sensing area. The opto-interrupters were mounted on an aluminum L-shaped base attached to a flat support on top of the leaf springs of the vehicle.

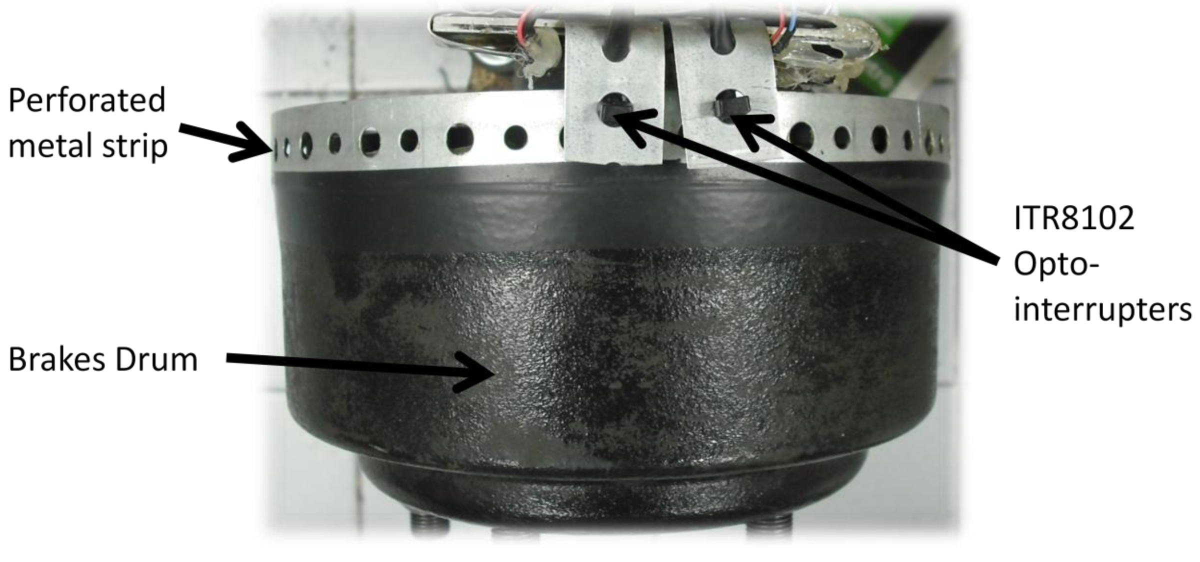


Figure D-1: The first version assembled for the wheel speed sensor.

Problems with the first attempt arose from the fact that the perforations on the metal strip were of different size and that the infrared beam not always crossed the circular hole through its center, instead of sensing along the diameter sometimes it sensed along a chord, taking different time from one pulse to the next one; this resulted in a pulse train with asymmetrical pulse periods, greatly affecting the performance of the sensor. Also, ITR8102 opto-interrupters had to be replaced due to malfunctioning of their phototransistors.

An improvement over the ITR8102 opto-interrupter mounting resulted from using a different L-shaped aluminum base, as shown in figure 3-1. With this arrangement the sensors were kept in their original plastic casing, improving the sensing area, reducing the negative effect of external light and becoming more resistant to vibrations caused by the movement of the vehicle.

The second version of the wheel speed sensor used the EE-SX670 opto-interrupters and attempted to solve previous issues by aligning every center of the perforations. Making them the same size while grinding one edge of the metal strip until it became tangent with the boundary was done to the metal strip. This new sensor had 50 holes opposed to the firsts 48. This second attempt was mounted on the right rear wheel. A slight improvement in performance was noticed but still RPM measurements presented oscillations because of the asymmetrical periods and were not within accepted parameters.

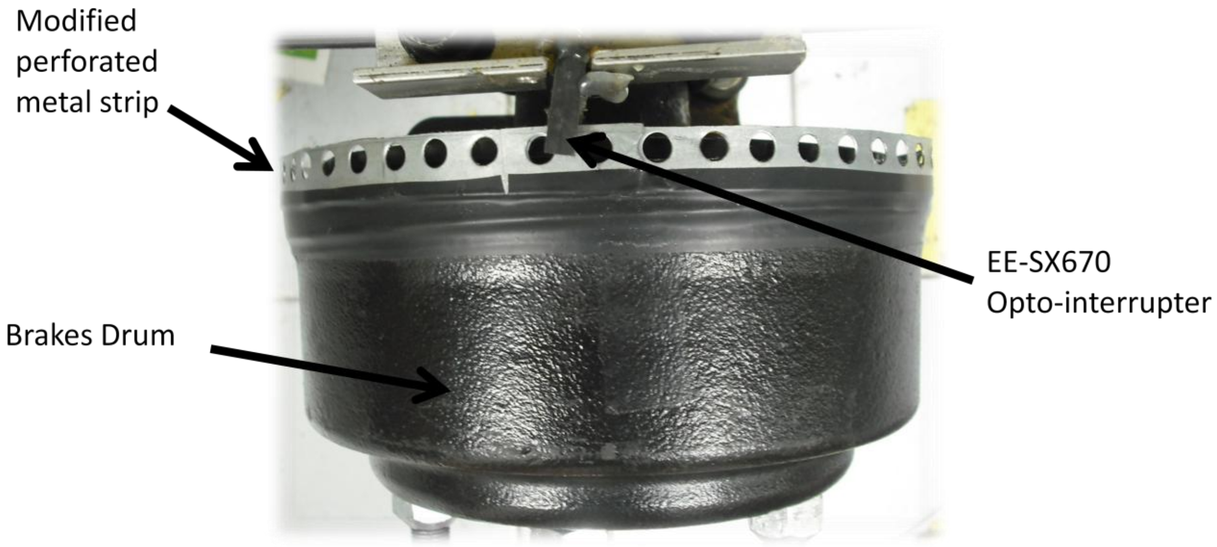


Figure D-2: Second attempt at the wheel speed sensor.

The third and final version of the sensor was implemented using a different perforated metal strip. This new strip had 241 rectangular perforations, solving the alignment of the holes and providing a higher resolution for RPM calculation. Great improvement was noticed on signal quality, but the EE-SX670 opto-interrupter lost track of the pulses when the wheels speed surpassed 400 RPM due to its limiting frequency of 1 KHz. Thus, although less industrial, the ITR8102 opto-interrupters were preferred.

# References

[Albores, 2007] Carlos Albores. *Analysis, Architecture, and Fusion Methods for Vehicle Automation*. ITESM Campus Monterrey, 2007

[Álvarez et al., 2006] Bárbara Álvarez, Pedro Sánchez, Juan A. Pastor, Francisco Ortiz. *An Architectural Framework for Modeling Teleoperated Service Robots*. División de Sistemas e Ingeniería Electrónica (DSIE); Universidad Politécnica de Cartagena (Spain), 2006

[Apco] Apco Inc. *Open Loop Tuning Rules*. Advanced Process Control & Optimization Inc. Apco Website

<http://www.apco-inc.com/articles/pidtune2.pdf>

[Åström, 2002] Karl Johan Åström. *Control System Design*. Department of Mechanical and Environmental Engineering; University of California Santa Barbara, 2002

[Azzeh, 2005] Abdel Azzeh. *CAN Control System for an Electric Vehicle*. University of Canterbury, 2005

[Barbier et al., 2009] M. Barbier, H. Cao, S. Lacroix, C. Lesire, F. Teichteil-Königsbuch and C. Tessier. *Decision issues for multiple heterogeneous vehicles in uncertain environments.* Onera Toulouse4th National Conference on "Control Architectures of Robots" Toulouse, April 23-24, 2009

[Behnke, 2006] Sven Behnke, Jürgen Müller, and Michael Schreiber. *Toni: A Soccer Playing Humanoid Robot*. Albert-Ludwigs-University of Freiburg, Computer Science Institute

| | |
|---|---|
| [Borenstein, 1994] | J. Borenstein. *Internal correction of dead-reckoning errors with the smart encoder trailer.* Intelligent Robots and Systems '94. 'Advanced Robotic Systems and the Real World', IROS '94. Proceedings of the IEEE/RSJ/GI International Conference on , vol.1, no., pp.127-134 vol.1, 12-16 Sep 1994 |
| [Bosch, 1991] | Robert Bosch GMBH. *CAN Specification Version 2.0.* 1991 |
| [Braybrook, 2004] | Ray Braybrook. *Three D missions - - dull, dirty and dangerous*. Armada International, 2004 |
| [Brooks, 1985] | R. A. Brooks. *A robust layered control system for a mobile robot.* Cambridge, MA, USA, Tech. Rep., 1985 |
| [Campion et al., 1996] | G. Campion, G. Bastin and B. d'Andréa Novel. *Structural Properties and classification of kinematic and dynamic models of wheeled mobile robots*. IEEE Trans. On Robotics and Automation, vol. 12, pp. 47-62, 1996. |
| [Chow & Tipsuwan, 2001] | Mo-Yuen Chow, Yodyium Tipsuwan. *Network-based control systems: a tutorial.* Industrial Electronics Society, 2001. IECON '01. The 27th Annual Conference of the IEEE , vol.3, no., pp.1593-1602 vol.3, 2001. |
| [Corrigan, 2008] | Steve Corrigan. *Introduction to the Controller Area Network.* Texas Instruments, 2008. |
| [Coste-Manière & Simmons, 2000] | E. Coste-Manière & R. Simmons. *Architecture: the backbone of robotics systems*. International Conference of Robotics and Automation, ICRA'00, San Francisco, 2000 |
| [Darpa, 2008] | DARPA. *Urban Challenge*. DARPA Urban Challenge Website, 2008<br><http://www.darpa.mil/grandchallenge/index.asp> |
| [Davis, 2010] | Leroy Davis. *Field Buses*. 2010.<br><http://www.interfacebus.com/Design_Connector_Field_Buses.html> |
| [Dowling, 1995] | Kevin Dowling. *Robotics: comp.robotics Frequently Asked Questions*. 1995 |

<http://www.frc.ri.cmu.edu/robotics-faq>

[Edsinger & Weber, 2004] A. Edsinger-Gonzales, J. Weber. *Domo: a force sensing humanoid robot for manipulation research*. Humanoid Robots, 2004 4th IEEE/RAS International Conference on, vol.1, no., pp. 273- 291 Vol. 1, 10-12 Nov. 2004

[Gonzalez, 2004] Gerardo Gonzalez. *Automatización de la Dirección de un Vehículo Autónomo*. ITESM Campus Monterrey, 2004

[Guo, 2008] Yi Guo. *EE631 Cooperating Autonomous Mobile Robots, Lecture 1: Introduction*. Department of Electrical and Computer Engineering; Stevens Institute of Technology, 2008.

<http://personal.stevens.edu/~yguo1/EE631Fall08/Introduction.pdf>

[Habumuremyi, 2005] Jean-Claude Habumuremyi. *Models of a wheeled robot named Robudem and design of a state feedback controller for its posture tracking*. Department of Mechanical Engineering (MSTA), Royal Military Academy; Brussels, Belgium, 2005

[Huo et al., 2004] Zhihong Huo, Huajing Fang, Changlin Ma. *Networked control system: state of the art*. Intelligent Control and Automation, 2004. WCICA 2004. Fifth World Congress on vol.2, no., pp. 1319- 1322 Vol.2, 15-19 June 2004.

[ISA, 1992] International Society of Automation. *ISA-5.1-1984 (R1992) Instrumentation Symbols and Identification*.

[Iversen, 2008] Wes Iversen. *PACs Gain Momentum*. Automation World Website, February 2008 (p. 30)

[Jöhnk & Dietmayer, 1997] Egon Jöhnk & Klaus Dietmayer. *Determination of Bit Timing Parameters for the CAN Controller SJA1000.* Application note 97046. Philips Semiconductors.

[Kaur, 2009] Navneet Kaur. *Development of CompactRIO based PID Temperature Controller*. Thapar University, Patiala, 2009.

[Kelly & Moreno, 2001] R. Kelly & J. Moreno. *Learning PID structures in an introductory course of automatic control.* Education,

| | |
|---|---|
| | IEEE Transactions on , vol.44, no.4, pp.373-376, Nov 2001 |
| [Kopp, 2009] | Emilie Kopp. *Defining a Common Architecture for Robotic Systems*. National Instruments, Webcasts on demand, 2009 |
| [Largiader, 2006] | Anton Largiader. *Single-WHAT-System?* BMW Riders Association International. OTL January 2006. <br><br> <http://www.bmwra.org/otl/canbus/> |
| [Leen & Heffernan, 2002] | G. Leen and D. Heffernan. *Expanding Automotive Electronic Systems*. IEEE, pp. 88-93, 2002. |
| [Lihua et al., 2008] | Zhang Lihua, Wu Yuqiang, Gong Lei, Chen Haojie. *A Novel Research Approach on Network Control Systems*. Internet Computing in Science and Engineering, 2008. ICICSE '08. International Conference on , vol., no., pp.262-265, 28-29 Jan. 2008. |
| [Mahyuddin & Arshad, 2008] | M. N. Mahyuddin and M. R. Arshad. *Classes of Control Architectures for AUV: A brief survey*. USM Robotics Research Group (URRG), School of Electrical and Electronic Engineering; Universiti Sains Malaysia, 2008 |
| [Mataric, 1997] | M. J. Mataric. *Behavior-Based Control: Examples from Navigation, Learning, and Group Behavior*. Journal of Experimental and Theoretical Artificial Intelligence, vol. 9, nos. 2-3, 1997 |
| [Microchip, 2003] | Microchip. *PIC16F87XA, 28/40/44-Pin Enhanced Flash Microcontrollers*. Microchip, 2003 |
| [Microchip, 2004] | Microchip. *MCP41010, Single/Dual Digital Potentiometer with SPI Interface.* Microchip. |
| [Microchip, 2007] | Microchip. *MCP2515, Stand-alone CAN Controller with SPI Interface*. Microchip, 2007 |
| [Microchip, 2010] | Microchip. *MCP2551, High-Speed CAN Transceiver*. Microchip, 2010 |
| [Murphy, 2000] | R. R. Murphy. *An Introduction to AI Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, November 2000 |

[NI, 2007] National Instruments. *PACs for Industrial Control, the Future of Control*. National Instruments White Paper July 30, 2007

[NI, 2008] National Instruments. *NI9853 High Speed CAN Module Operating Instructrions*. National Instruments, 2008

[NI, 2009a] National Instruments. *Controller Area Network (CAN) Overview*. November 7, 2009

[NI, 2009b] National Instruments. *NI CompactRIO – Reconfigurable Control and Acquisition System*. National Instruments White Paper November, 2009

[NI, 2009c] National Instruments. *CompactRIO Developers Guide*. National Instruments, December 2009.

[NI, 2009d] National Instruments. *CAN Physical Layer Standards: High-Speed vs. Low-Speed/Fault-Tolerant CAN*. Knowledge Base. National Instruments, 2009

[NI, 2009e] National Instruments. *PID and Fuzzy Logic Toolkit User Manual.* National Instruments, 2009

[NI, 2010] National Instruments. *Quadrature Encoder Velocity and Acceleration Estimation with CompactRIO and LabVIEW FPGA.* Tutorial. National Instruments, 2010.

[Opto22, 2008a] Opto 22. *Understanding Programmable Automation Controllers (PACs) in Industrial Automation*. Opto 22, PACs in Industrial Automation White Paper

[Opto22, 2008b] Opto 22. *RPM Measurement Techniques*. Technical Note. Opto 22, 2008.

[Palacios, 2000] Gerardo Palacios. *Control De Dirección De Un Vehículo Autónomo.* ITESM Campus Monterrey, 2000

[Pazul, 1999] K. Pazul. *Controller Area Network (CAN) Basics*. Microchip Technology Inc., AN713, May 1999.

[Pohjola et al., 2006] M. Pohjola, L. Eriksson, H. Koivo. *Tuning of PID Controllers for Networked Control Systems.* IEEE Industrial Electronics, IECON 2006 - 32nd Annual Conference on , vol., no., pp.4650-4655, 6-10 Nov. 2006

[Resnick, 2003] Craig Resnick. *Plus-size platform gives controller bigger name*. Automation World Website, June 2003 (p. 68)

[Richards, 2005] Pat Richards. *A CAN Physical Layer Discussion.* Application Note 228. Microchip, 2005

[Rowe & Wagner] Steve Rowe and Christopher R. Wagner. *An Introduction to the Joint Architecture for Unmanned Systems (JAUS).* Cybernet Systems Corporation.

[Salam, 2003] Zainal Salam. *DC motor drives.* Power electronics and drives, 2003

<http://encon.fke.utm.my/courses/see_5433/>

[Sevcon, 2004] Sevcon. *MillipaK SEM Controller Manual For System Version 1.54.01.* 2004

[Tadokoro, 2009] Satoshi Tadokoro. *Rescue Robotics*. Springer, 2009

[Touchton et al., 2006] Bob Touchton, Tom Galluzzo, Danny Kent, and Carl Crane. *Perception and Planning Architecture for Autonomous Ground Vehicles*. University of Florida. 2006

[UC3M, 2009] *UC3M Humanoid*. P3 and ASIMO – The Honda Humanoid Robots

<http://honda-p3.com/robot_worldwide/uc3m-humanoid-2.html>

[Valavanis et al., 1997] K. P. Valavanis, D. Gracanin, M. Matijasevic, R. Kolluru and G.A. Demetriou. *Control architectures for autonomous underwater vehicles*. Control Systems Magazine, IEEE; Volume 17, Issue 6, Dec. 1997; Page(s):48 – 6

[Viswanathan, 2005] V. Sornam Viswanathan. *Embedded Control Using FPGA*. Indian Institute of Technology, Bombay, Mumbai, 2005

[Xiao, 2008] John Xiao. *Mobot: Mobile Robot*. Department of Electrical Engineering; City College of New York. 2008

[Zaccarian] Luca Zaccarian. *DC motors: dynamic model and control techniques.*

[Zhao et al., 2010] Ningning Zhao, Demin Xu, Jian Gao, Weisheng Yan. *Application of CAN-bus to the control system of the TESTBED AUV*. Computer Engineering and Technology (ICCET), 2010 2nd International Conference on , vol.2, no., pp.V2-645-V2-648, 16-18 April 2010

[Zhong, 2006] Jinghua Zhong. *PID Controller Tuning: A Short Tutorial*. Mechanical Engineering, Purdue University, 2006