Defence Research and Development Canada
Recherche et développement pour la défense Canada

The eXtreme Test Bed vehicle for indoor environments

A first level control

Isabelle Vincent
Defence R&D Canada – Suffield

Technical Memorandum
DRDC Suffield TM 2004-262
December 2004

Report Documentation Page

Form Approved
OMB No. 0704-0188

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE **DEC 2004**	2. REPORT TYPE	3. DATES COVERED **-**

4. TITLE AND SUBTITLE **The eXtreme Test Bed vehicle for indoor environments (U)**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Defence R&D Canada -Suffield,PO Box 4000 Station Main,Medicine Hat, AB,CA,T1A 8K6**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

The original document contains color images.

14. ABSTRACT

The eXtreme Test Bed is a small-scale remotely controlled truck that has been retrofitted with a microprocessor, sensors, servos and actuators to experiment with different concepts and software applications in an indoor environment. Two intelligence levels are exploited. A low level vehicle control is realized by the MPC555 microprocessor for reactive avoidance, wandering state and A-to-B mobility. The second is a navigation control level presently in development. This technical memorandum focuses on the first control level. It details the MPC555 functionalities, program functions, mathematics, algorithms and components used to control the eXtreme Test Bed. The objective is to test technologies and software capabilities before application to more expensive platforms. A second goal is to become familiar with a real-time operating system executing several tasks in parallel.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES **60**	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**			

The eXtreme Test Bed vehicle for indoor environments

A first level control

Isabelle Vincent
Defence R&D Canada – Suffield

Defence R&D Canada – Suffield

Technical Memorandum

DRDC Suffield TM 2004-262

December 2004

Author

Isabelle Vincent

I. Vincent

Approved by

D. Hanna

Head/Tactical Vehicle Systems Section

Approved for release by

Dr. P. D'Agostino

Chairman/Document Review Panel

Abstract

The eXtreme Test Bed is a small-scale remotely controlled truck that has been retrofitted with a microprocessor, sensors, servos and actuators to experiment with different concepts and software applications in an indoor environment. Two intelligence levels are exploited. A low level vehicle control is realized by the MPC555 microprocessor for reactive avoidance, wandering state and A-to-B mobility. The second is a navigation control level presently in development. This technical memorandum focuses on the first control level. It details the MPC555 functionalities, program functions, mathematics, algorithms and components used to control the eXtreme Test Bed. The objective is to test technologies and software capabilities before application to more expensive platforms. A second goal is to become familiar with a real-time operating system executing several tasks in parallel.

Résumé

L'eXtreme Test Bed est un petit véhicule tout-terrain télécommandé qui a été modifié par l'ajout d'un microprocesseur, de capteurs, de servos et d'actuateurs pour expérimenter différents concepts et algorithmes dans un environnement intérieur. Deux niveaux d'intelligence sont exploités. Le contrôle de base du véhicule est réalisé à l'aide du microprocesseur MPC555 pour exécuter des évitements d'obstacles, un état d'errance ou une mobilité d'un point A à un point B. Le second niveau est un module de contrôle de la navigation du robot qui est présentement en développement. Ce mémoire technique traite du premier niveau de contrôle. Il détaille les fonctionnalités du MPC555, les fonctions du programme informatique, les formules mathématiques, les algorithmes de programmation et les composantes du système employées pour contrôler l'eXtreme Test Bed. L'objectif est de tester les capacités de diverses technologies et programmes informatiques avant de les appliquer sur des plate-formes plus dispendieuses. Un second but est de se familiariser avec un système d'opération en temps réel qui exécute plusieurs tâches en parallèle.

This page intentionally left blank.

Executive summary

Background

In future combat fields, unmanned vehicles will play a vital role. Civilian and military communities are developing robotic systems with sufficient autonomy to replace and assist humans or increase their capabilities to achieve specific tasks.

The Autonomous Intelligent Systems project (AIS-Land) at Defence R&D Canada – Suffield, works on conceptualization, invention and application of technologies to develop innovative unmanned ground vehicles.

Results

In Fall 2003, a project using a small-scale prototype, the eXtreme Test Bed (XTB), started. The goal was to test and become familiarized with technologies, sensors, a real-time embedded system, a microprocessor, concepts and algorithms before installing them on bigger and more expensive platforms. It is developed for indoor applications.

This document focusses on the low level control of the robot, i.e. reactive obstacle avoidance, wandering mode and rudimentary A-to-B mobility. The XTB gets data from the different sensors and controls the motor and servos. It details the vehicle components and the way they are controlled. The platform used is a Tamiya TXT-1 Extreme R/C Monster Truck retrofitted to become an autonomous test bed. The sensors allow understanding and interaction with the environment. Odometry and range detection are used to implement the robot servo-control.

An important part of the document is the program algorithm. In the implementation two tools are used: a high-performance microprocessor, the MPC555 from Motorola, and a real-time operating system called RTEMS. The project aims to become familiar with those powerful tools so they can be applied to other systems.

Significance

The first level control is completed. Teleoperation and simple autonomous modes like reactive obstacle avoidance, wandering mode and simple A-to-B mobility are available. This modular first control allows one to add higher control levels sending data to the basic level that controls the vehicle. Thus, it is possible to test different navigational and arbitrational algorithms with very few modifications to the first level control.

Future Results

By adding a camera to the infrared range detection, the system will get a better understanding of the environment. It will involve the implementation of a second control level providing navigation commands to the first control level by using image data to solve navigation algorithms and then verify arbitration rules. This more complex intelligence is under development.

Another eventual work is multi-robot applications. The aim will be to familiarize with communication systems and development of algorithms to realize missions by interaction of several robots.

Vincent, I. 2004. The eXtreme Test Bed vehicle for indoor environments. DRDC Suffield TM 2004-262. Defence R&D Canada – Suffield.

Sommaire

Contexte

Sur les champs de bataille du futur, les véhicules non-pilotés joueront un rôle vital. Les communautés civiles et militaires développent des systèmes robotisés avec suffisamment d'autonomie pour augmenter les capacités des humains, les remplacer ou les assister, afin d'exécuter des tâches spécifiques.

Le projet des Systèmes autonomes intelligents (AIS-Land) à R & D pour la Défense Canada – Suffield, travaille à la conceptualisation, à l'invention et à l'application de technologies pour développer des véhicules terrestres autonomes innovateurs.

Résultats

À l'automne 2003, un projet utilisant un prototype de petite taille, l'eXtreme Test Bed (XTB), vit le jour. Son objectif était de tester et de se familiariser avec des technologies, des capteurs, un système d'opération en temps réel, un puissant microprocesseur, ainsi que des concepts et des algorithmes avant de les installer sur des plate-formes plus grosses et plus dispendieuses. Le véhicule est développé pour des applications d'intérieur.

Ce document traite du premier niveau de contrôle du robot, c'est-à-dire l'évitement d'obstacles, un mode d'errance et une simple mobilité d'un point A à un point B. Il obtient les données de divers capteurs et contrôle le moteur et les servos. Les composantes du véhicules sont détaillées ainsi que la façon dont elles sont contrôlées. La plate-forme utilisée est un camion Tamiya TXT-1 Extreme R/C Monster Truck modifié pour devenir un banc d'essai autonome. Les capteurs permettent de comprendre et d'interagir avec l'environnement. L'odométrie et la détection de la distance des objets sont utilisés pour contrôler le robot.

Une partie importante du document est l'algorithme du programme informatique. Pour exécuter le programme, deux outils sont utilisés: un microprocesseur haute performance, le MPC555 de Motorola, et le système d'opération en temps réel RTEMS. Le projet vise à se familiariser avec ces deux puissants outils pour ensuite pouvoir les employer dans d'autres systèmes.

Portée des résultats

Le premier niveau de contrôle est complété. La téléopération et de simples modes autonomes sont disponibles tels l'évitement d'obstacle par simple réaction, un mode d'errance et une mobilité rudimentaire d'un point A à un point B. Ce premier contrôle modulaire permet d'ajouter des niveaux de contrôle plus élevés qui envoient des données au niveau de base contrôlant le véhicule. Ainsi, il est possible de tester différents algorithmes de navigation et d'arbitration en effectuant peu de modifications au premier niveau de contrôle.

Travaux futurs

En ajoutant une caméra au système en plus de la détection infrarouge, le robot obtiendra une meilleure compréhension de son environnement. Cela impliquera l'application d'un second niveau de contrôle qui fournira les commandes de navigation au premier niveau de contrôle en utilisant les données des images de la caméra et de la détection infrarouge pour résoudre des algorithmes de navigation et vérifier les règles d'arbitration. Cette intelligence plus complexe est présentement en développement.

Éventuellement, des applications de coopération multirobot seront étudiées. Le but sera de se familiariser avec les systèmes de communication et le développement d'algorithmes pour réaliser des missions impliquant l'interaction de plusieurs robots.

Vincent, I. 2004. The eXtreme Test Bed vehicle for indoor environments. DRDC Suffield TM 2004-262. R & D pour la défense Canada – Suffield.

Table of contents

List of figures

List of tables

1. Introduction

The Autonomous Intelligent Systems project (AIS-Land) at Defence R&D Canada - Suffield, works on conceptualization, invention and application of technologies to develop innovative unmanned ground vehicles. The eXtreme Test Bed (XTB) is a small scale unmanned ground vehicle used for indoor applications and, eventually, multi-robot research activities. The prototype being inexpensive, it is a good experimental platform prior to applying technologies to more expensive vehicles. It is suitable to experiment with different concepts, technologies, interfaces and software algorithms.

The XTB is a small RC truck. Specifically, it is a Tamiya TXT-1 Extreme R/C Monster that has been retrofitted with sensors, a MPC555 microprocessor, two scanning bumpers and an active platform. The first control level is realized by the 40MHz MPC555 board which controls every truck component. It reads the encoder, controls the motor speed, the steering angle and the bumper servo angle, maintains the platform balance, monitors the inclinometer and the suspension sensors, and performs the infrared ranging for obstacle avoidance. It executes teleoperation, a wandering state and rudimentary A-to-B mobility.

In future work, a second control level will be provided by a 533MHz PC-104 board. It will run Linux and probably Carmen control software. This software provides mapping, localization, path planning and navigation. The PC104 board will also be in charge of monitoring camera data. Inter Process Communication (IPC) will make the connection between the two CPUs to send the navigation commands from the PC104 to the MPC555, and receive data acquisition and vehicle states transmitted by the MPC555 to the PC104.

The ultimate goal of the project is to produce a small autonomous vehicle to test and become familiarized with technologies, concepts, the MPC555 and the real-time operating system RTEMS, and realizing missions, individually or in multi-robot tasks. In parallel, a teleoperation mode is under development, using a radio frequency remote control to manually direct the prototype.

This technical memorandum focuses on the MPC555 controller for the creation of real world behaviors. The next section presents the robot description, detailing the components and the sensors with their specifications. The following section describes the real-time operating system, the compiler and the debugger. Before presenting the project algorithms from section 7, section 6 details different functions of the program and the control of each component of the vehicle. Finally, Annex A explains the servo's calibration and Annex B lists the acronyms used in the document.

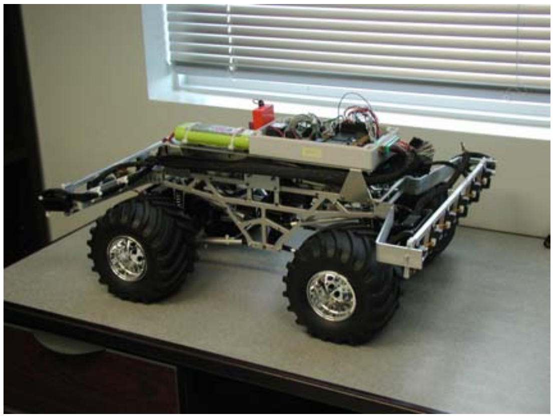

Figure 1: *The eXtreme Test Bed platform.*

2. Vehicle description

This section details the components of the vehicle and the sensors used to control the robot and get information about its environment. To support the description, a picture of the prototype is presented in Figure 1.

2.1 Components

The XTB is a Tamiya TXT-1 Extreme R/C Monster Truck retrofitted to become an autonomous test bed. The light aluminum frame supports four wheels with 4.17” x 6.5” tires. The truck comes with a Tamiya’s Mabuchi RS-540 high torque DC motor providing 540 Watts of power. The dual-motor gearbox generates a 34:1 gear ratio. Two drive axles transmit the power to the wheels, providing a four-wheel drive vehicle. The steering system is composed of independent front and back steering. High torque servos actuate the dual tie-rod to orient the wheels. They are Hobbico HCAMo191 Command CS-70MG Super torque 2BB servos, generating up to 7.7 kg-cm of torque at 4.8 Volts.

The vehicle requires a 7.2 Volt battery to operate. The Venom racing 3000 NIMH power pack delivers 3000 mah to the system. A Novak Super Rooster reversible speed control is used to regulate the power sent to the motor and the servos. Moreover, a DC/DC converter is added. It is a Datel single output UNS series non-isolated, 3.3 and 5 Volt, 3 Amp DC/DC power converter, low cost, high efficiency (90-92%), low output noise and wide range inputs.

The cantilever suspension shown in Figure 2, is composed of oil dampers with 4 inch coils, offering smooth and adjustable damping. Four servos activate the cantilevers to push the arms compressing the shocks. The suspension servos are Hitec HS-5645MG digital ultra torque servos. The speed operation is 0.23sec/60deg at 4.8V with an output torque of 10.3 kg-cm at 4.8V.

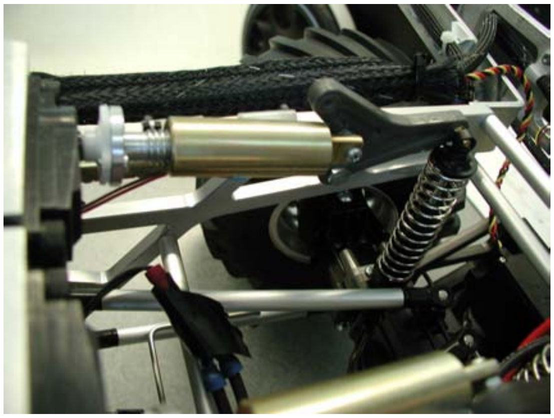

Figure 2: *Cantilever suspension activated by a high torque servo rotating the piston screw.*

The servos send an electrical input determining the position of the motor armature. In fact, the duty cycle of a periodic rectangular pulse train determines this position. The signal period is 22.2 ms. The pulse length varies from 1.0 to 2.0 ms with the pulse being 1.5 ms when the arm is centered. For the active suspension, the position of the servo arm is less useful than the rotation speed. To create this effect, the servo has been modified. The feedback sensor is removed and replaced by an equivalent circuit, making the servo rotate continuously in the appropriate direction as long as the signal remains. This means that a constant pulse length commands the servo to go to a certain position. Being modified, it never reaches this stable position and rotates at a constant speed. Thus, the servo screws or unscrews the piston pushing the suspension cantilever.

The platform is tele-operated with a Fatuba Conquest FP-T5NLP 5 channels FM pulse code modulation radio control. The corresponding receiver is a Fatuba FP-R1051P PCM 72.990 MHz FM 5 channels micro receiver.

Finally, the main processor of the robot is a PowerPC based SS555 from Intec Automation Inc. The 40 MHz MPC555 Motorola embedded microprocessor is a RISC PowerPC core with a floating-point unit. To realize the XTB project, it has 8 pulse width modulation channels, 32 10-bit analog-to-digital converter pins, 2 time processor units with 16 independent channels each and 2 RS-232 serial communication interface ports. It supports the Macraigor Wiggler and P&E ICDPPC In-Circuit Emulator/Debug cable.

2.2 Sensors

The robot needs sensors to sense its environment and adapt its behavior. Twelve infrared object detectors are installed, six on the front bumper and six at the back. They are equally distributed every 7 cm. Figure 3 shows a bumper with the selected sensors: the SHARP IR GP2D12 sensors manufactured by Acroname. These detectors are small, consume very little current and offer good ambient lighting immunity. They use

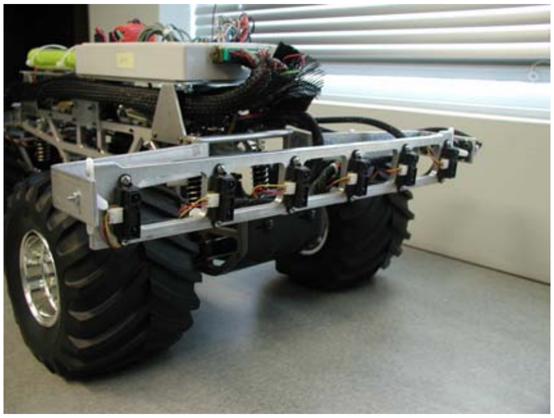

Figure 3: *SHARP GP2D12 Infrared sensors for range detection.*

a triangulation method and a small CCD array to compute the object distance. The non-linear output analog signal varies between 0 to 3 Volt and is updated approximately every 32 ms. The detectors fire continuously and require 25 mA of continuous current. An analog-to-digital converter converts the distance measurements in numeric values. The range detection is 10 to 80 cm, however, with a non-reflective object, this length is reduced. Experimentally, the effective range has been evaluated to be 10 to 25 cm. Over this boundary, the readings are not sufficiently accurate to be useful. Due to the bell shape of the output sensor signal, below 10 cm the voltage appears similar to ranges over 10 cm. To avoid misinterpretation of the output, the range used doesn't cover the 0-10 cm detection range.

The top platform of the truck can be pitched or rolled by controlling the shock damping. By compressing a suspension, the platform tilts. Thus, it is possible to program an active suspension to keep the platform balanced. To determine the tilt angle, a tilt sensor is positioned in the middle of the platform. A dual axis analog tilt sensor from Crossbow, model CXTA02, offers ±75 degree range with ±20 degree linear. It is fully conditioned, its resolution is 0.05 degree and its sensitivity is 35 ±2 mV/degree. Moreover, a linear resistor sensor (35K) is installed on each piston to measure its current position. It allows to do not force the servos when trying to overpass the piston limits, and, for the platform balance controller, to keep the piston as much as possible in its center. This prevents the four suspensions from reaching their maximum or minimum limit.

To measure the vehicle speed, an optical encoder is mounted on the engine shaft. The Servo-Tek PM series face-mount encoder is a small size and low cost solution. It gives a single channel square wave output used for motor speed control and speed indication. The input voltage is 5V and current 16 mA. It produces 60 pulses per revolution for a maximum shaft speed of 12 000 rpm and accepts rotation in either direction.

To get odometry data, a gyroscope is added to the system. The MicroStrain 3DM-G

Gyro Enhanced Orientation Sensor provides orientation and angular velocity. It incorporates 3 triaxial magnetometers, 3 angular rate gyros and 3 orthogonal DC accelerometers for the three axes. A 12 bit analog-to-digital converter converts the output signals. An embedded microcontroller then provides the orientation angles (pitch, roll and yaw) by RS-232 serial communication at 38.4 kbaud to the MPC555. For all axes, the orientation range is 360 degrees, the angular velocity range is ± 300 degrees/second, the accelerometer range is ± 2 G's and the magnetometer is ± 1 Gauss. The orientation angle resolution is less than 0.1 degrees. The output data rate is 100 Hz.

3. Software supports

This section defines the compiler, the debugger and the real-time operating system used in the XTB project.

3.1 Compiler

The GCC is the high quality GNU C compiler. It supports C and C++ language programs. Finally, it is frequently used on machines running Linux.

3.2 Debugger

The powerful open-source GNU program debugger called GDB is utilized for the project.

3.3 Real-time Operating System

RTEMS stands for Real Time Executive for Multiprocessor Systems. It is a free real-time system and it is supported by the MPC555 target. It is based upon the free GNU tools and the open source C Library Newlib. The real-time executive provides a high performance environment for embedded systems.

RTEMS has an object-oriented nature, encouraging modular applications. The real-time operating system is appropriate for applications with rigorous requirements, not just the results of computations, but also critical time constraints to produce the results. Finally, it has multitasking capabilities, so it can manage several concurrent processes.

Table 1 lists the different managers featuring in the RTEMS kernel. More information about RTEMS and details about its functions and managers can be found in [1].

Categories	Managers
Time	Rate monotonic
	Clock
	Timer
Communication and synchronization	Semaphore
	Message queue
	Event
	Signal
Memory	Partition
	Region
	Dual ported memory
Others	Initialization
	Task
	Interrupt
	Input/output
	Fatal error
	User extension
	Multiprocessing

Table 1: *RTEMS managers*

4. Program Functionnalities

This section describes the control of each component of the vehicle. There are also explanations about the arguments chosen for the Motorola functions and the mathematical equations employed.

4.1 Pulse Width Modulation (PWM)

The signal period required by the motor and the servos is 22.2 milliseconds. These components are controlled by PWM command or by modulation of the pulse width. There are eigth PWM channels on the MPC555 and they command the motor speed, the back bumper angles, three suspension actuators and the front steering angle. Due to the lack of channels available, the front bumper and the front-right suspension actuator servos are controlled by two Time Processor Unit (TPU) PWM channels. The PWM pulses sent to the components are obtained by multiplying the duty cycle by the PWM signal period of 22.2 milliseconds. The duty cycle is evaluated by the time processor unit channels receiving signals from the remote control. The duty cycle, proportion of the signal that is high during one period, is constrainted between 4.2% and 9.5%. This is the limits for the servos. The systems have been calibrated to get a relation between the PWM command and the resulting speed or angle. The calibration graphs are presented in Annex A. Variation of the PWM command varies motor or servo speed or angle. The infrared sensor bumpers and the steerings use the PWM command for position control. The motor and the suspension activators use it for speed control.

4.2 Time Processor Unit 3 (TPU3)

The MPC555 comprises an enhanced version of the original TPU designed for timing control. The TPU3 operates simultaneously with the Central Processor Unit (CPU), minimizing the CPU interrupt services. It consists of two time bases, each having 16 independent timer channels. The general functions of the TPU are defined in [2] and a specific reference about the TPU3 is [3, ch.17]. To get more information about the TPU, consult [4].

4.2.1 TPU3 Setup

The TPU3 setup needs to set four registers [3, ch.17]. The TPU configuration register module TPUMCR = 0x0020 means the TPU3 clocks are enabled, no prescaler is used for the time count registers (TCR1 and TCR2), the TPU3 doesn't operate in emulation mode, the assignable registers are accessible from user and supervisor privilege level and the TPU3 mode is selected. The time count register used is the TCR1. The configuration register module 3 TPUMCR3 = 0x005f means that an enhanced 64 prescaler divides the Intermodule Bus (IMB) clock. The prescaler was determined to detect pulses well. Too low frequency would result in undetected edges and too high frequency would involve a measurement window too small to cover the entire signal period. The TPU3 interrupt configuration register TICR requires setting the channel interrupt request level field and the interrupt level byte select field. A priority level of five is selected using the IRQ[0:7] byte.

4.2.2 New Input Transition / Input Capture Function (TPU_NITC)

This function described in [5] can detect rising and/or falling input transitions. It records the TCR value, getting the number of ticks occurring between two edges. It is possible to determine the duty cycle by counting the ticks between two consecutive rising and falling edges. This function is used to measure the duty cycle of the signal sent by remote control to the receiver. The next example shows the TPU and the channel used, an high interrupt priority and a continuous mode on rising and falling edges are chosen, the TCR1 counts the clock ticks, no channel link is utilized, and an interrupt is asseted every two edges.

Example: tpu_nitc_init_tcr_mode(tpua, channel_speed, TPU_PRIORITY_HIGH, TPU_NITC_RISING_FALLING, 2, TPU_NITC_CONTINUOUS,TPU_NITC_TCR1, TPU_NITC_NOLINK, TPU_NITC_START_LINK_CHANNEL_0, TPU_NITC_LINK_ONE, TPU_NITC_INTERRUPT).

This function gets the duty cycle for the servos, the motor and the manual/automatic selector. The 40MHz processor frequency is divided by a 64 prescaler, thus a TCR tick occurs every 1.6 microseconds. The TPU counts these TCR ticks. To get the duty cycle, multiply the tick count by the TCR sample time and divide by the signal period.

$$DC = \frac{TC(P_{tick})}{P}$$

Where

DC = duty cycle

TC = tick count

P = signal period (22.2 ms)

P_{tick}= TCR tick period (1.6 us).

One problem encountered is that the function TPU_NITC doesn't allow the user to select to start on a rising or a falling edge when the mode rising and falling edges is selected. The reading can be the tick count between the rising and the falling edges of the period or between the falling and the rising edges of the period. One is the complement of the other. In term of duty cycle, the complement is 100% minus the result. Another problem is that when using several tpu channels, strange duty cycles are sometimes obtained like 420%. To solve the problem, the defective tpu channel is reset to get a good duty cycle. Also, to avoid overpassing the limitations of the motor and the servos, when the duty cycle is over 9.5%, the previous PWM command is not modified.

The NITC function is used to measure the duty cycle of the remote control signals.

4.2.3 Frequency Measurement Function (TPU_FQM)

This function described in [6, 7] counts completed pulses on a TPU channel within a specified time accumulation window. The options are single time window or continuous mode, and rising or falling edge capture. An interrupt occurs when the time window is complete. For the present case, the next example shows which TPU and channel are used, a continuous mode is chosen to measure frequency of rising edges, the interrupt priority is high and the window size is 8000 (maximum window size).

Example: tpu_fqm_init(tpua, channel_enc, TPU_PRIORITY_HIGH, TPU_FQM_RISE_EDGE_CONT, TPU_FQM_RISE, TPU_FQM_TCR1,0x8000).

This function counts the encoder pulses and determines the wheel speed. Reading the encoder output means counting the number of ticks during the sample time.

$$t_s = \frac{W(64)(4)}{f_{sys}}$$

Where

t_s = sample time [s]

W = window size (8000)

f_{sys} = microprocessor frequency (40 MHz).

The 64 prescaler is set in the tpu setup function. The purpose and origin of the constant 4 is unknown. This prescaler was established by comparing the TPU reading to the expected reading using a 64 prescaler. To get the frequency, divide the tick count by the sample time. This is the engine shaft frequency. To get the wheel frequency, it is divided by the gear ratio (34:1). Then the wheel speed is expressed in rpm.

$$f_w = \frac{TC}{t_s(GR)}$$

$$rpm = \frac{f_w(60)}{f_{sys}}$$

Where

TC = tick count

t_s = sample time [s]

f_w = wheel frequency [Hz]

GR = gear ratio (34/1)

rpm = wheel speed [rpm]

f_{sys} = microprocessor frequency (40 MHz).

The FQM function gives an accurate measure of the wheel speed for this project. Due to the limited time window size, this function wouldn't be appropriate for a low speed system.

4.3 Remote Control

Each TPU channel is configured to measure rising and falling edges in continous mode. Thus, it is possible to know the duty cycle for each signal sent by the remote control and received by the receiver. There are three signals controlled remotely: wheel speed, steering angle, and the manual/automatic selector. Two additionnal channels are available. The manual control allows the user to position the robot and perform tests manually. The automatic mode is used for autonomous control.

4.4 Serial Communication Interface (SCI)

The commands are received and the data transmitted by the serial communication interface. The MPC555 controls two RS-232 ports. The first serial port receives commands to control the vehicle in automatic mode and to send data about sensors and the state of each component. The second serial port communicates with the gyroscope to get orientation and acceleration data, both useful to calculate odometry.

The serial port parameters are a 9600 baud rate for SCI1, and 38400 for SCI2. The gyroscope needs fast communication.

4.5 Infrared range detectors (IR)

To detects obstacles, the XTB has been equipped with two active bumpers, each having six IR sensors equally distanced of 7 cm. Each sensor output is converted by the analog-to-digital converter (ADC) in a binary output . The adc_init function is configured to scan the 16 channels of port B once. The next formula is the conversion in Volt of the 10-bit data.

$$IR_{volt} = \frac{IR_{adc}(V)}{R}$$

Where

IR_{adc} = ADC binary output

IR_{volt} = ADC output converted [V]

V = power supplied to the ADC (5V)

R = 10-bit resolution (1024).

The sensor used a triangulation process to determine the distance. The triangle hypotenuse lentgh is calculated as:

$$\begin{aligned} IR_{hyp-len} &= 0.9195 IR_{volt}^6 + 1.6166 IR_{volt}^5 - 40.426 IR_{volt}^4 + 125.12 IR_{volt}^3 \\ &\quad - 131.76 IR_{volt}^2 + 2.4594 IR_{volt} + 69.999 \end{aligned}$$

Where

IR_{volt} = ADC output converted [V]

$IR_{hyp\text{-}len}$ = hypotenuse length [cm].

The linear distance of an object is finally determined by finding the cosine of the bumper angle.

$$IR_{lin-dist} = IR_{hyp-len} cos(\theta)$$

Where

$IR_{lin\text{-}dist}$ = the linear distance to the object [cm]

$IR_{hyp\text{-}len}$ = hypotenuse length [cm]

θ = bumper angle [rad].

The resulting distance, expressed in centimeters, is the projected distance on the horizontal. A safety process verifies the proximity of the object. If the measured distance is shorter than 25 cm and the obstacle is in the same direction as the vehicle movement, the vehicle is immobilized. The detection accuracy depends on the distance, the reflectivity of the objects and the sensor properties. Consequently, it is better to not use this data to model the environment, but to establish a range of acceptable object proximity. Below 10 cm, the sensor output increases, falsely reporting an increase in distance. Thus, if an object comes too close to the robot, it may be falsely reported as far away. Figure 4 shows the detection range.

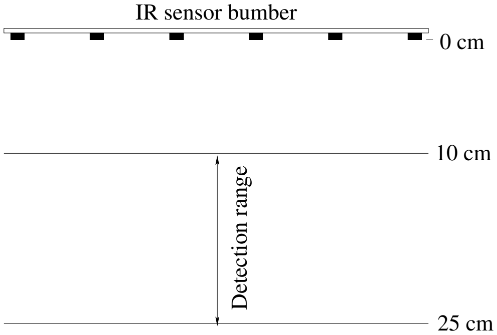

Figure 4: *IR bumper detection range sketch*

4.6 Obstacle avoidance

It is verified that the IR sensor measurements are not below 25 cm. This range was established to give the vehicle enough time to brake. The algorithm notes if the obstacle is at the front, at the back, both, or none. The sensor that detects the object is not important because the curvature radius of the XTB is too large to avoid an obstacle just by turning the steering and the vehicle is also too large to do a quick manoeuver. When an obstacle is seen in front, the best strategy is to reverse and to try another path. Two options are possible. If the obstacle is on the vehicle path, the XTB stops. Otherwise, the presence of the obstacle doesn't influence the robot trajectory and travel can continue. In teleoperation mode, the vehicle doesn't move in the direction of a detected obstacle. In wander mode, the vehicle stops, then turns while driving backward from the obstacle for 6 seconds, then goes straight forward. It's a simple avoidance algorithm. No obstacle detection operates while the vehicle moves backward from the obstacle. If a new one is encountered, it won't stop the movement until the end of the avoidance process. This could be improved. However, it would complicate the algorithm and would not be necessary since the dead period is short and, with the vehicle size, the environment should not be too cluttered. In A-to-B mobility mode, the obstacle avoidance option is not integrated yet. However, to avoid collision the vehicle stops if any obstacle is detected in the driving direction.

4.7 Speed PID Loop

A PID servo-control loop controls smoothly the vehicle speed given a speed reference command. The system was experimentally calibrated through trial and error, to determine the PID proportional, derivative and integrative constants K_P, K_D *and* K_I. Here are the steps to realize the servo-control loop. First, get the error for the n^{th} sample by comparing the desired speed to the current speed.

$$e[n] = rpm_{ref} - rpm_{cur}$$

Where

e[n] = error for the nth sample

rpm_{ref} = reference speed [rpm]

rpm_{cur} = current speed [rpm].

Then evaluate the correction to apply with the next formulas.

$$corr = Ae[n] + Be[n-1] + C\sum e[n]$$

$$A = K_P + 0.5K_I t_s + \frac{K_D}{t_s}$$

$$B = 0.5K_I t_s - \frac{K_D}{t_s}$$

$$C = K_I$$

$$\sum e[n] = \sum e[n-1] + 0.5\,(e[n] + e[n-1])\,t_s$$

Where

corr = correction to apply to the motor command

$\sum e[n]$ = error sum at the nth sample

t_s = TPU sampling time to get the encoder reading [s]

$K_P,\ K_D,\ K_I$ = PID constants determined by trial and error
($K_P = 10,\ K_D = 0.1,\ K_I = 20$).

Finally, the correction is applied to the motor PWM command (Q) by addition to the no movement PWM value ($PWM_{no\text{-}move}$).

$$Q = PWM_{no-move} + corr$$

It has been noticed that the progression is smoother forward than backward due to a smaller PWM value range for backward motion than forward speed.

4.8 Platform tilt and suspension activation

The platform of the robot is mounted on four independant cantilever suspensions that can be controlled to tilt the platform. A dual-tilt sensor indicates the inclination of the platform and a linear sensor is installed on each suspension piston (actuator) to measure its elongation. Figure 5 shows the relation between the components and sensors involved in the servo-control of the platform balance.

The calculation of the tilt sensor angles is presented in subsection 4.8.1, the two controllers balancing the platform are then explained. The first algorithm developed (subsection 4.8.2) uses a PID loop to vary the servo PWM command with regards to the tilt sensor feedback. The second controller was developed to improve the first one by centering the piston position and respecting the piston limits. It is described in 4.8.3.

4.8.1 Tilt sensor

The tilt sensor voltages are converted in angle and then compared to the references ($\theta_{ref-x}, \theta_{ref-y}$) = (0,0) to keep the platform horizontal. A PID loop executes this comparison and modifies the PWM command sent to the suspension actuators to control the suspension height.

First, the sensor sensitivities S is converted in Volt per radian.

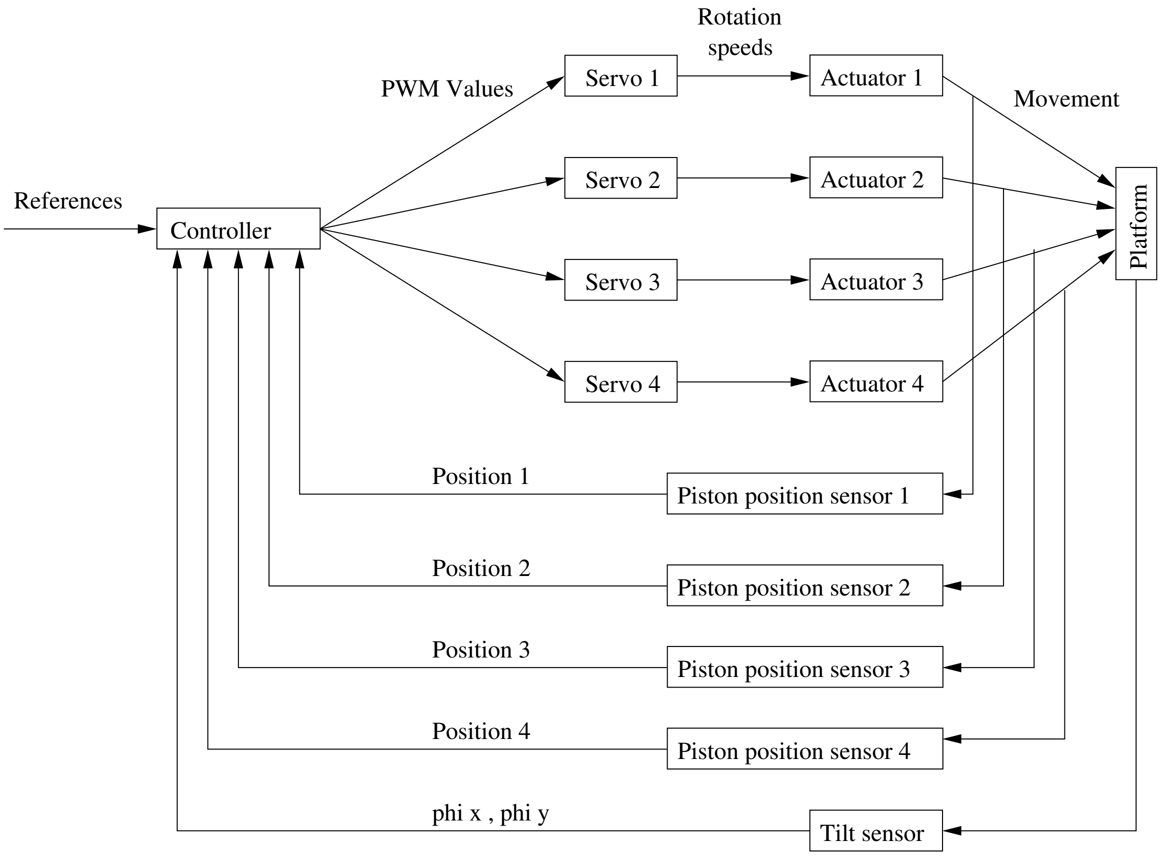

Figure 5: *The eXtreme Test Bed platform balance controler diagram*

$$S_i[V/red] = \frac{180}{\pi} S_i[V/deg]$$

Where

S = sensitivity

i = x or y axis.

The adc_init function is configured to scan the 16 channels of port B once. The adc_get function reads the data. The 10-bit values are then converted to Volt for both axes.

$$V_{out-volt-i} = \frac{V_{out-i}(Pow)}{R}$$

Where

V_{out-i} = ADC binary output

$V_{out-volt-i}$ = ADC output converted [V]

Pow = power supplied to the ADC (5V)

R = 10-bit resolution (1024)

i = x or y axis.

The tilt angles Φ_x and Φ_y are calculated by doing the difference between the output and the reference angle (0,0) voltage Vref for the x and y axes, and then by making the arcsine of the comparison with the sensitivity. The result is then converted to degrees.

$$\Phi_i = \frac{180}{\pi} \arcsin\left(\frac{V_{out-volt-i} - V_{ref-i}}{S_i}\right)$$

Where

$V_{out-volt-i}$ = ADC output converted [V]

Φ_i = tilt angle [deg]

V_{ref-i} = reference voltage for a zero tilt angle [V]

S_i = sensitivity [V/deg]

i = x or y axis.

For the actual tilt sensor, the sensitivities are $S_x = 0.034813V/deg$ and $S_y = 0.033934V/deg$, and the reference voltages for a zero tilt angle are $V_{ref-x} = 2.525V$ and $V_{ref-y} = 2.503V$.

4.8.2 Platform PID loop

The servo-control of the platform stability is done with a PID loop controlling four suspension actuators and two tilt axis measurements.

First, get the error for the n^{th} sample time error[n] in x and y.

$$e[n]_x = \Phi_{ref-x} - \Phi_{cur-x}$$

$$e[n]_y = \Phi_{ref-y} - \Phi_{cur-y}$$

Where

e[n] = error for the nth sample

Φ_{ref} = reference tilt angle [deg]

Φ_{cur} = current tilt angle [deg].

Then, evaluate the corrections to apply with the next formulas.

$$corr_x = A_x e[n]_x + B_x e[n-1]_x + C_x \sum e[n]_x$$

$$corr_y = A_y e[n]_y + B_y e[n-1]_y + C_y \sum e[n]_y$$

$$A_x = K_{P-x} + 0.5 K_{I-x} t_s + \frac{K_{D-x}}{t_s}$$

$$A_y = K_{P-y} + 0.5 K_{I-y} t_s + \frac{K_{D-y}}{t_s}$$

$$B_x = 0.5 K_{I-x} t_s - \frac{K_{D-x}}{t_s}$$

$$B_y = 0.5 K_{I-y} t_s - \frac{K_{D-y}}{t_s}$$

$$C_x = K_{I-x}$$

$$C_y = K_{I-y}$$

$$\sum e[n]_x = \sum e[n-1]_x + 0.5\left(e[n]_x + e[n-1]_x\right) t_s$$

$$\sum e[n]_y = \sum e[n-1]_y + 0.5\left(e[n]_y + e[n-1]_y\right) t_s$$

Where

corr = correction to apply to the motor command

$\sum e[n]$ = error sum at the nth sample

t_s = TPU sampling time to get the encoder reading [s]

$K_P,\ K_D,\ K_I$ = PID constants determined through trial and error ($K_P = 200,\ K_D = 0.1,\ K_I = 10$).

Finally, the corrections are applied to the commands (Q) by addition to the zero movement PWM value (PWM_{zero}). Because there are two corrections applied on each actuator, x and y corrections, and one can be in the opposite direction than the other, the axes orientations have to be carefully studied. The suspension servos are identified by their position: back-right (BR), back-left (BL), front-right (FR) and front-left (FL).

$$Q_{BR} = PWM_{zero-BR} + corr_x + corr_y$$

$$Q_{BL} = PWM_{zero-BL} + corr_x - corr_y$$

$$Q_{FR} = PWM_{zero-FR} - corr_x + corr_y$$

$$Q_{FL} = PWM_{zero-FL} - corr_x - corr_y$$

For instance, when the vehicle tilts on the right side, a correction in x is required. The BR actuator has to increase the suspension height. Similarly for the FR actuator. But the two left ones have to decrease the suspension height. The commands are finally compared to the PWM limits allowed by the servo to operate properly.

The platform movement works well, but is slow. A possible solution would be to increase the screw pitch, which will require enlarging the screw diameter. Another way is to modify the piston mount on the cantilever arm. The length is reduced to get a bigger influence on the compression for a smaller piston deplacement. This modification has been done successfully.

For small inclinations, the PID loop is sufficient to keep the balance, but for important ones, few seconds are required to bring the platform horizontal. Moreover, the continuous loop drains lots of power being always compensating for tilt difference. A tolerance zone should be added to avoid this continuous instability around the reference angles. This instability is related to the tilt sensor accuracy and the ADC resolution. For a constant inclination, the ADC reads a value varying around the correct value.

For each suspension, a sensor reads the position of the piston. This tool is integrated to the PID loop to avoid overshooting the physical limits of the piston and, consequently, damaging the servos. If the maximum or minimum position is reached, the servo stops rotating.

4.8.3 Platform balance controller

Another control algorithm has been developed to avoid draining the battery power. It also maintains the platform as centered as possible to avoid reaching the ends of the suspension movement.

This algorithm needs the tilt angle in both axes and the position of each piston. The ADC scans 64 times each channel on port A to get an average value for each. Then, the height variation required for each suspension is determined with the next formulas. As the tilt angles are very small, the small angle approximation is used ($\sin\theta \cong \theta$).

$$\triangle h_{FR} = \frac{(l\Phi_x - w\Phi_y)}{2}$$

$$\triangle h_{FL} = \frac{(l\Phi_x + w\Phi_y)}{2}$$

$$\triangle h_{BR} = \frac{(-l\Phi_x - w\Phi_y)}{2}$$

$$\triangle h_{BL} = \frac{(-l\Phi_x + w\Phi_y)}{2}$$

where

$\triangle h_i$ = heigth variation for the ith suspension

l = vehicle length [m]

w = vehicle width [m]

Φ_x, Φ_y = tilt angles [deg].

To center the piston, an average value μ is evaluated.

$$\mu = \frac{\triangle h_{BL} + \triangle h_{BR} + \triangle h_{FL} + \triangle h_{FR}}{4}$$

The way the height variations are calculated, the average value is always zero. Thus, it is centered. Then, by comparing $\triangle h_i$ to μ, a new height variation $\triangle h_i'$ is obtained.

$$\triangle h_i' = \triangle h_i - \mu$$

To convert the piston height variation in displacement, a factor K multiplies $\triangle h_i'$. This factor has been determined through trial and error and represents the speed of convergence of the servo. The piston displacement is then evaluated by adding the result to the piston middle position mp_i and by substracting the current position z_i.

$$\triangle z_i = K \triangle h_i' + mp_i - z_i$$

Where

mp_i = middle position of piston i

z_i = piston current position

K = convergence factor (set to for the experiment K = 5)

$\triangle h_i$ = height variation for the ith suspension

$\triangle z_i$ = piston displacement.

The relation between the piston speed and the servo PWM value is displayed graphically in the Annex A. As can be seen, the relation isn't linear. It seems to be more of the 4th order. However, a linear approximation is sufficient for the present application. Knowing the sampling time and the desired displacement, the speed is thus known. Here is the pulse width formula:

$$PWM_i = A_i \frac{\triangle z_i}{\triangle t} + B_i$$

where

PWM_i = pulse width modulation of piston i

$\triangle t$ = sampling time [s]

A, B = linear constants.

To avoid instability of the system around the tilt reference angles, a tolerance area is delimited. The algorithm considers a tilt of $\Phi_x \pm 0.8^\circ$ and $\Phi_y \pm 0.8^\circ$ sufficient. In this zone, the pulse width is set to stop the piston.

Some safety processes are added to the algorithm. First, the piston position is limited to not break the linear sensor and force the servo. When a piston end is reached, the PWM value is set to stop the servo in the end direction. Second, the pulse width is also limited to avoid excessive servo speed.

4.9 Odometry

Odometry is needed to localize the vehicle. This function requires encoder and gyroscope readings. The vehicle position is evaluated every encoder reading. At each iteration, the distance d traveled is calculated.

$$d = V(t)(\pi)(D)$$

where

d = distance [m]

V = current speed [rps]

t = odometry sampling time [s]

D = wheel diameter [m].

The position increments, $\triangle x$ and $\triangle y$, are evaluated using the vehicle yaw angle α. The angle is gyro-stabilized as explained in the Microstrain documentation [8, p.7]. Also, when the autonomous mode is initialized, the angle measured becomes the offset so everytime this mode is selected with the remote control, the angle is set to zero.

$$\triangle x = d \sin \alpha$$

$$\triangle y = d \cos \alpha$$

Then, these increments are added to the previous position to get a global vehicle position.

$$x[n] = x[n-1] + \triangle x[n]$$

$$y[n] = y[n-1] + \triangle y[n]$$

The accuracy of this odometry calculation is sufficient for a short test distance, but the error increased at every iteration. The odometry could be improved by combining gyroscope data with Global Positionning System data. With the MPC555, this combination is not possible since only two serial ports are available: one for multiprocessor communication and one for the gyroscope.

4.10 A-to-B Mobility

The purpose of this function is to move the robot from one point to another. Since the odometry isn't very accurate, the result of the mobility is also not accurate. However, it is reasonable over a short distance.

Position errors in x and y axis are determined by the next formulas.

$$\triangle X = X_B - X$$

$$\triangle Y = Y_B - Y$$

Where

(X_B, Y_B) = arrival position coordinates

(X,Y) = current position coordinates

$(\triangle X, \triangle Y)$ = position errors

The mission is successful if the position respects a tolerance of 20 cm on X and Y.

To move the robot, the idea is to determine where is the arrival point and which side to rotate. First, the final point is positioned in a quadrant around the robot. The quadrant determination is presented in Table 2. Figure 6 sketches the geometry angles of the different configurations to facilitate the comprehension of the next explanations.

△X	△Y	Quadrant
>= 0	>= 0	1
< 0	>=0	2
< 0	< 0	3
>= 0	< 0	4

Table 2: *Quadrant determination*

Then, the absolute heading α from the current position to the target is evaluated with the next formula. One exception is for $\triangle Y$= 0, α = 90 deg.

$$\alpha = \arctan\left[abs\left(\frac{\triangle X}{\triangle Y}\right)\right]$$

The angle α is then expressed in the same reference system as the gyroscope. The result is β. It means that the zero degree angle of the gyroscope is equivalent to β = 0.

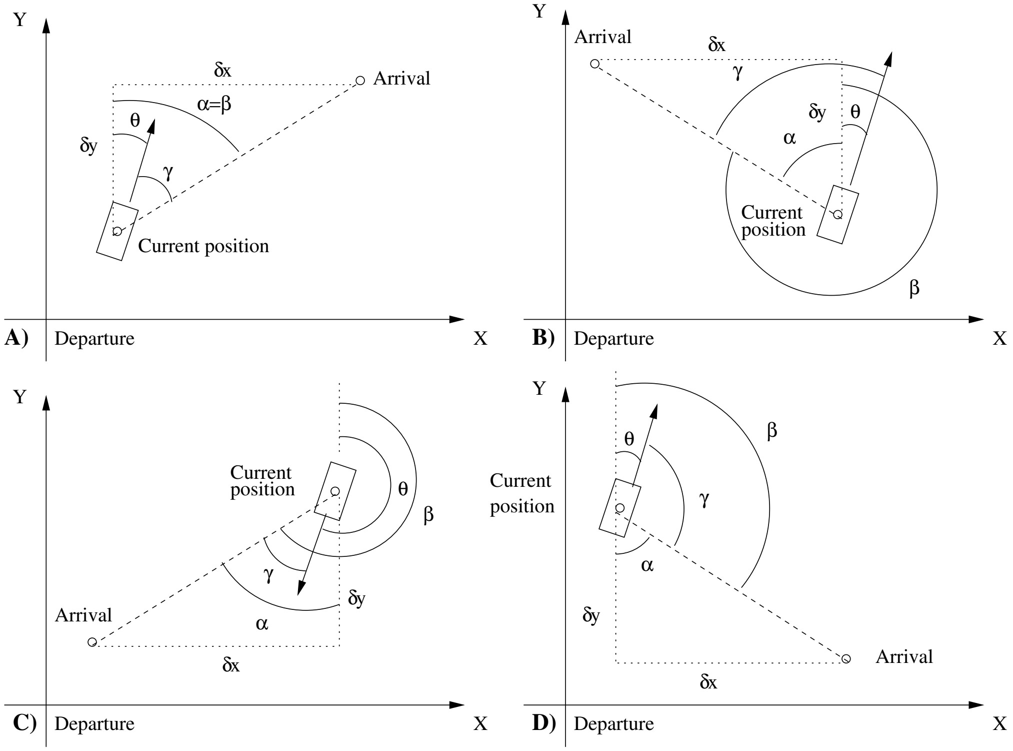

Figure 6: *Representation of the different configurations of the arrival point in accordance with the current position. A) Quadrant 1, B) Quadrant 2, C) Quadrant 3 and D) Quadrant 4.*

Quadrant	Formula to get β
1	$\beta = \alpha$
2	β=360-α
3	$\beta = 180 + \alpha$
4	$\beta = 180 - \alpha$

Table 3: *Equations to evaluate* β *in accordance with the quadrants*

Each quadrant required a different formula to execute this transformation. Table 3 presents the equations to get β.

The useful angle to orient the robot is the angle γ between the current robot orientation θ and the target angle β. γ represents the angle the robot has to rotate. As the algorithm works with positive angles between 0 and 360 degrees, for negative angles $\gamma = \gamma + 360$. Same rule for θ.

$$\gamma = \beta - \theta$$

Now that the rotational movement requirement has been evaluated, a PID servo-control function is used to determine the robot movement. For each $\gamma\,45^\circ$ slice, a different movement is established. It considers that for some angles, it is more practical to go backward than forward and sometimes it is better to go backward a little to reorient the robot and then forward, like driving a car. For instance, if γ is 60 degrees, it is impossible for the vehicle to turn on a short radius, so it necessitates a big space to describe an arc to orient itself in this direction. It is more appropriate to go backward left up to a smaller γ. When γ get between 0-45 degrees, then the robot goes forward right. Thus, the space required to orient the robot is smaller. Table 4 shows the XTB behavior in each $\gamma\,45^\circ$ degree slice.

The Kp constant is a proportional constant $(K_P = 195)$. It has been determined to get a maximum PWM value for a 5 degree γ, where a maximum steering angle is required to turn as quickly as possible. The values 3710 and 4179 are the zero degree steering angle PWM values of the steering servos. V is a speed requested. The value of V has been established to 13 rpm in the experiment. Though this is an arbitrary speed offering good progression, it can be accelerated or decelerated.

Then, the steering PWM values are compared to the servos limits. If they are over the maximum or below the minimum, the values are changed for the limit values. The steering and the motor can now be commanded.

The mobility function is executed continuously every gyroscope reading. When the wander mode program is chosen, there is no A-to-B mobility available. The opposite is true too. They do not have to be integrated together since they shouldn't be used in the same applications.

5. Program algorithms

This section presents simplified diagrams of the algorithms developed to control the XTB. They give a good idea of the software algorithm content and the relations between the different tasks. A short explanation of the RTEMS specific terms used is previously given.

γ [deg]	Behavior	Correction	Front steering PWM	Back steering PWM	Speed
0 and 360	Straight forward	0	3710	4179	V
0<$\gamma \leq$45	Right forward	$K_P \times \gamma$	3710-correction	4179+correction	V
45<$\gamma \leq$90	Left backward	$K_P \times \gamma$	3710+correction	4179-correction	-V
90<$\gamma \leq$135	Left forward	$K_P \times \gamma$	3710+correction	4179-correction	V
135<$\gamma \leq$180	Right backward	$K_P \times \gamma$	3710-correction	4179+correction	-V
180<$\gamma \leq$225	Left backward	$K_P \times (\gamma - 360)$	3710-correction	4179+correction	-V
225<$\gamma \leq$270	Right forward	$K_P \times (\gamma - 360)$	3710+correction	4179-correction	V
270<$\gamma \leq$315	Right backward	$K_P \times (\gamma - 360)$	3710+correction	4179-correction	-V
315<γ<360	Left forward	$K_P \times (\gamma - 360)$	3710-correction	4179+correction	V

Table 4: *A-to-B mobilty PID equations to control the vehicle behavior*

5.1 Terminology

First of all, some real-time system expressions has to be explained.

Task: ”The smallest thread of execution which can compete on its own for system resources.” [1, p.30]

Semaphore: ”Protected variable whose value can be modified only when creating, obtaining or releasing directives. RTEMS supports both binary and counting semaphore.” [1, p.89]

Message-queue: ”Variable length buffer where information can be store to support communication.” [1, p.103]

Event: ”An event flag is used by a task to inform another task of the occurence of a significant situation.” [1, p.119]

5.2 Simplified algorithm diagrams

Due to the complexity of the program software algorithms, simplified diagrams are presented in this section. The first one is the initialization routine. It defines, initializes, starts and deletes tasks, semaphores and message queues. Figure 7 presents the main function of the real-time system that setups all the program.

The second diagram is the TPU setup in Figure 8. It configures the time processor unit. No task using the TPU can run without having received an event from the setup as being completed. Thus, it allows the tasks to proceed.

Follow the tpu reset task in Figure 9. It resets the TPU channels that are not set properly.

The remote control task (Figure 10) reads the signals from the receiver to get their duty cycles. It determines the mode (manual or autonomous), the remote speed and the steering commands. In manual mode, the task commands the motor and the steering servos. It also executes a simple reactive avoidance by forbidding any movement in a detected obstacle direction.

Figure 11 shows the infrared range detection diagram. The sensors detect the presence of obstacles and send warnings to other tasks. It allows the program to realize simple reactive avoidance.

To know the vehicle speed, an encoder reading task is developed, see Figure 12. It counts the encoder pulses and calculates the speed. In autonomous mode, it then compares it to the requested speed, evaluates the correction required to the motor command and commands the engine.

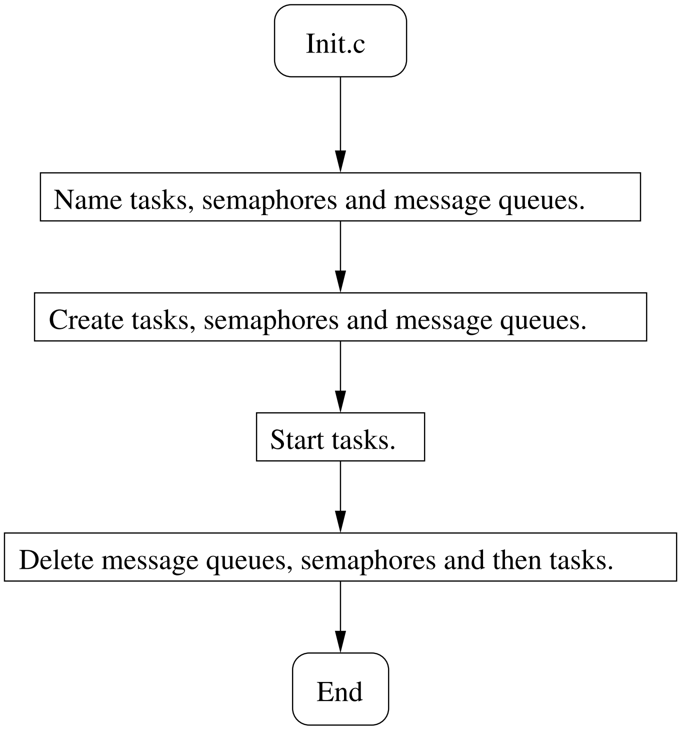
Init.c
Name tasks, semaphores and message queues.
Create tasks, semaphores and message queues.
Start tasks.
Delete message queues, semaphores and then tasks.
End

Figure 7: *Program initialization diagram*

The next task, Figure 13, is a function to wander in an indoor environment with obstacles. It is realized only in autonomous mode. When an object is detected, the robot stops running in the obstacle direction. It reverses while turning for 6 seconds, then stops 2 seconds and continues straight forward.

To control the platform tilt, two different control processes were tried. The first one, Figure 14, uses feedback from a dual-axis tilt sensor and four suspension position sensors to control the platform orientation. It involves a proportional correction and tries to keep centered the pistons. The second one, Figure 15, uses only the tilt sensor with a PID loop. The first control is prefered for the application because it respects the piston limits. Also, it keeps centered the piston as much as possible to optimize the suspension movement.

The last diagram (Figure 16) demonstrated a rudimentary A-to-B mobility algorithm. For a given arrival point, it orients the vehicle in this point direction and then reaches it. The robot analyzes the rotation required and executes backward or forward movements and right or left steering to optimize the space required to orient itself. A 20 cm tolerance is applied in x and y axes to complete a successful mission. Finally, to avoid collision, if an obstacle is detected, no movement is allowed in this direction. When the obstacle is removed, the robot continues. In future work, an obstacle avoidance routine will be integrated to this algorithm.

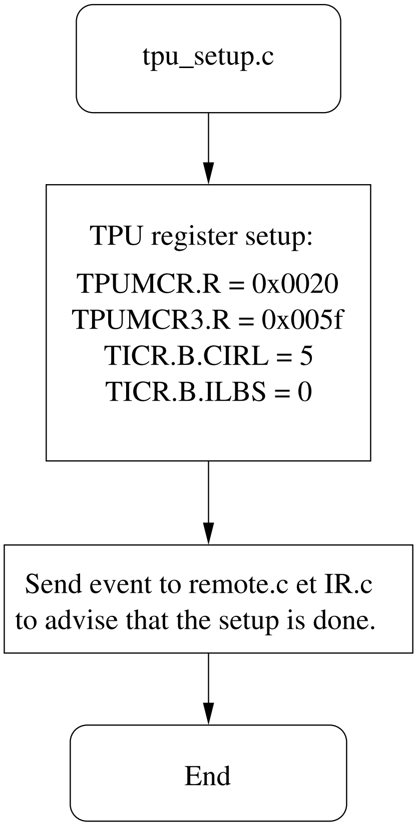

Figure 8: *TPU setup diagram*

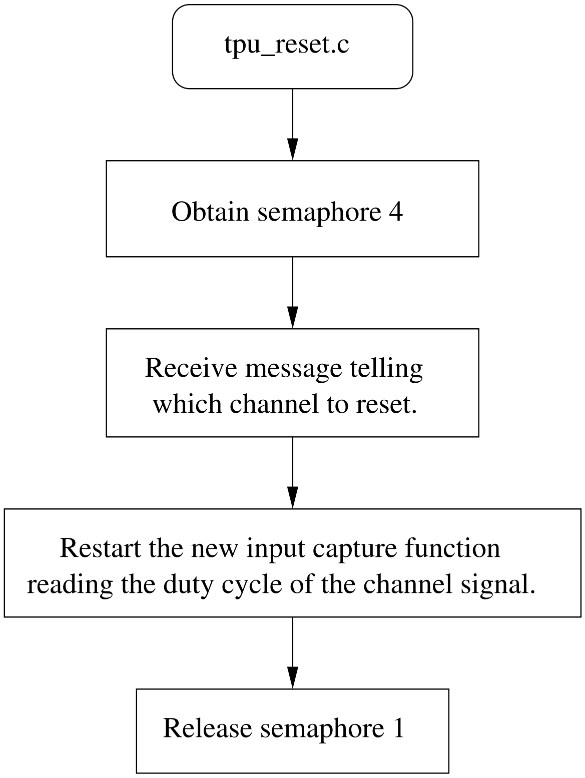
tpu_reset.c
Obtain semaphore 4
Receive message telling which channel to reset.
Restart the new input capture function reading the duty cycle of the channel signal.
Release semaphore 1

Figure 9: *TPU reset diagram*

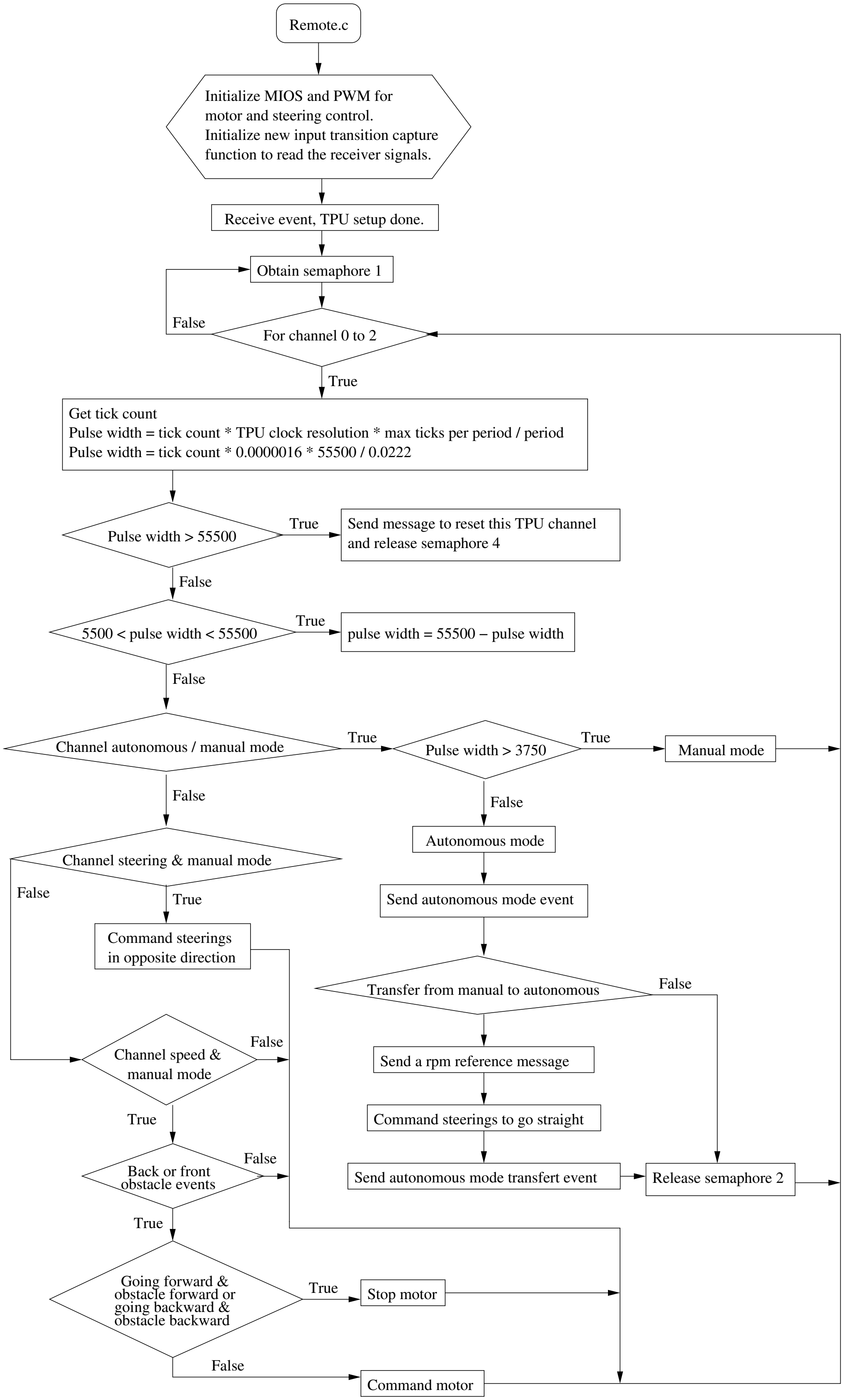
Remote.c
Initialize MIOS and PWM for motor and steering control.
Initialize new input transition capture function to read the receiver signals.
Receive event, TPU setup done.
Obtain semaphore 1
False
For channel 0 to 2
True
Get tick count
Pulse width = tick count * TPU clock resolution * max ticks per period / period
Pulse width = tick count * 0.0000016 * 55500 / 0.0222
Pulse width > 55500
True
Send message to reset this TPU channel and release semaphore 4
False
5500 < pulse width < 55500
True
pulse width = 55500 – pulse width
False
Channel autonomous / manual mode
True
Pulse width > 3750
True
Manual mode
False
False
Autonomous mode
Send autonomous mode event
Channel steering & manual mode
False
True
Command steerings in opposite direction
Transfer from manual to autonomous
False
Send a rpm reference message
Channel speed & manual mode
False
Command steerings to go straight
True
Back or front obstacle events
False
Send autonomous mode transfert event
Release semaphore 2
True
Going forward & obstacle forward or going backward & obstacle backward
True
Stop motor
False
Command motor

Figure 10: *Remote control diagram*

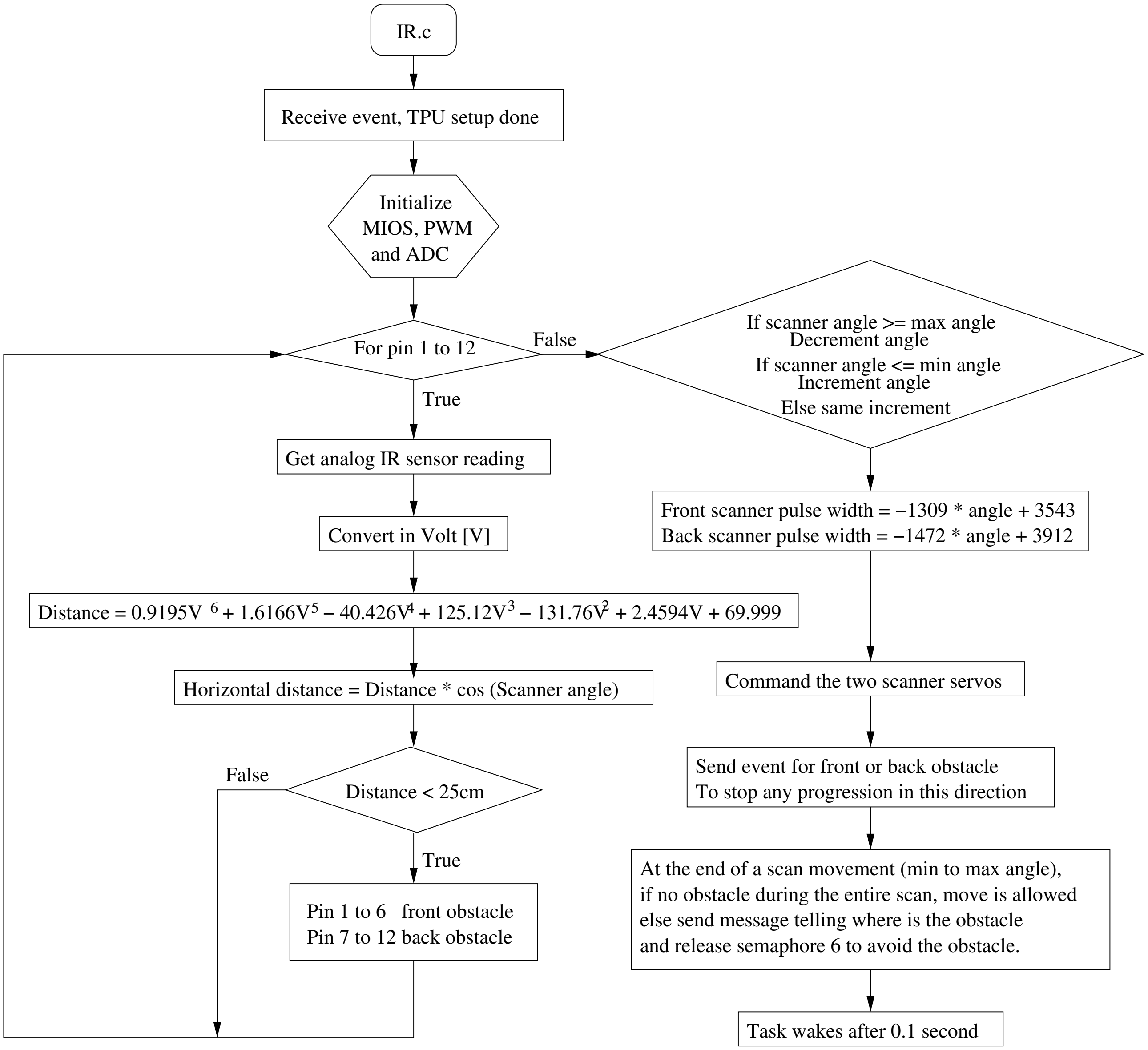
IR.c
Receive event, TPU setup done
Initialize MIOS, PWM and ADC
For pin 1 to 12
False
True
If scanner angle >= max angle
Decrement angle
If scanner angle <= min angle
Increment angle
Else same increment
Get analog IR sensor reading
Convert in Volt [V]
Front scanner pulse width = −1309 * angle + 3543
Back scanner pulse width = −1472 * angle + 3912
Distance = 0.9195V 6 + 1.6166V5 − 40.426V4 + 125.12V3 − 131.76V2 + 2.4594V + 69.999
Horizontal distance = Distance * cos (Scanner angle)
Command the two scanner servos
Send event for front or back obstacle
To stop any progression in this direction
False
Distance < 25cm
True
At the end of a scan movement (min to max angle),
if no obstacle during the entire scan, move is allowed
else send message telling where is the obstacle
and release semaphore 6 to avoid the obstacle.
Pin 1 to 6 front obstacle
Pin 7 to 12 back obstacle
Task wakes after 0.1 second

Figure 11: *IR range detection diagram*

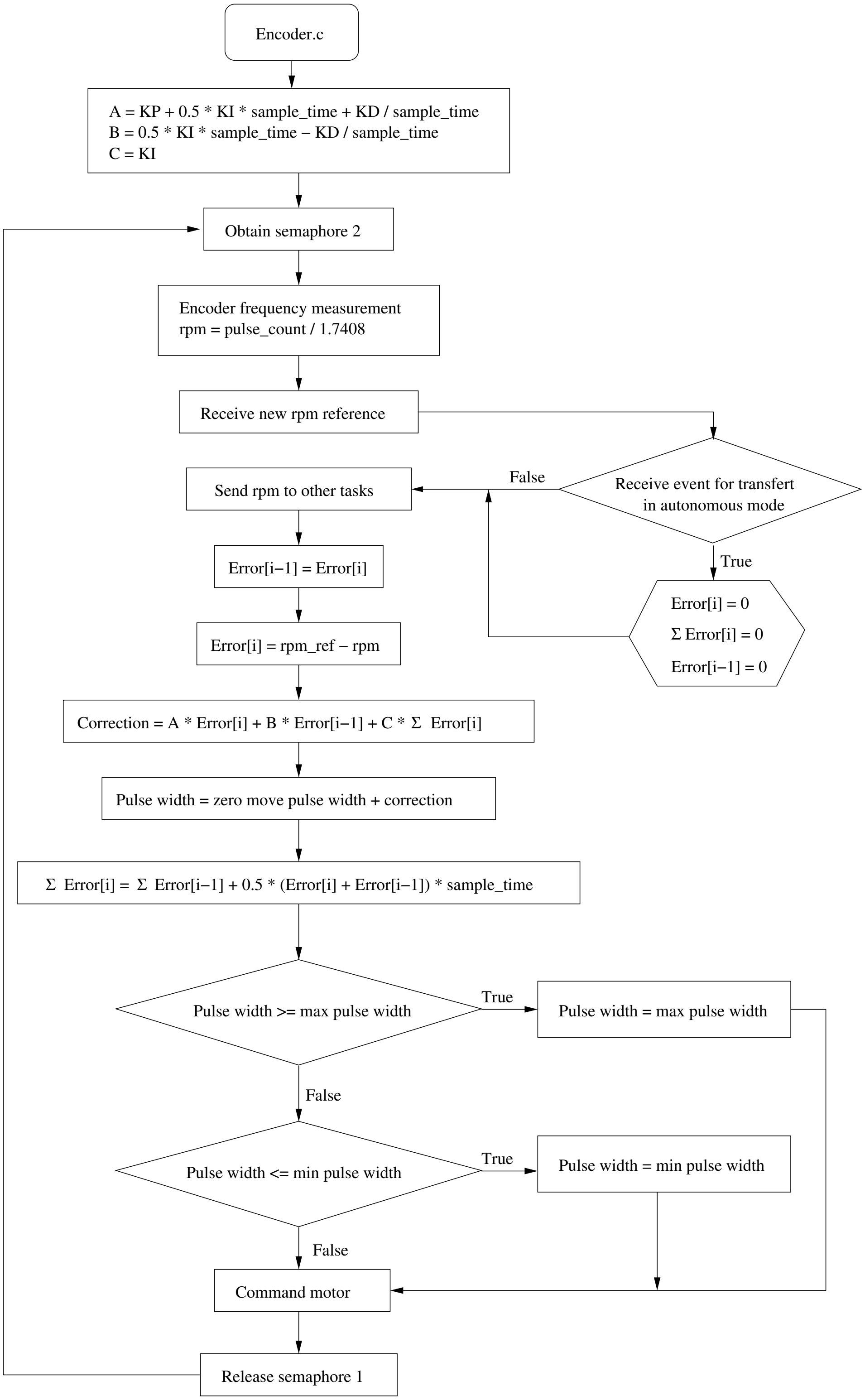
Encoder.c
A = KP + 0.5 * KI * sample_time + KD / sample_time
B = 0.5 * KI * sample_time − KD / sample_time
C = KI
Obtain semaphore 2
Encoder frequency measurement
rpm = pulse_count / 1.7408
Receive new rpm reference
Receive event for transfert in autonomous mode
False
True
Send rpm to other tasks
Error[i] = 0
Σ Error[i] = 0
Error[i−1] = 0
Error[i−1] = Error[i]
Error[i] = rpm_ref − rpm
Correction = A * Error[i] + B * Error[i−1] + C * Σ Error[i]
Pulse width = zero move pulse width + correction
Σ Error[i] = Σ Error[i−1] + 0.5 * (Error[i] + Error[i−1]) * sample_time
Pulse width >= max pulse width
True
Pulse width = max pulse width
False
Pulse width <= min pulse width
True
Pulse width = min pulse width
False
Command motor
Release semaphore 1

Figure 12: *Diagram for the encoder reading and the autonomous mode speed control*

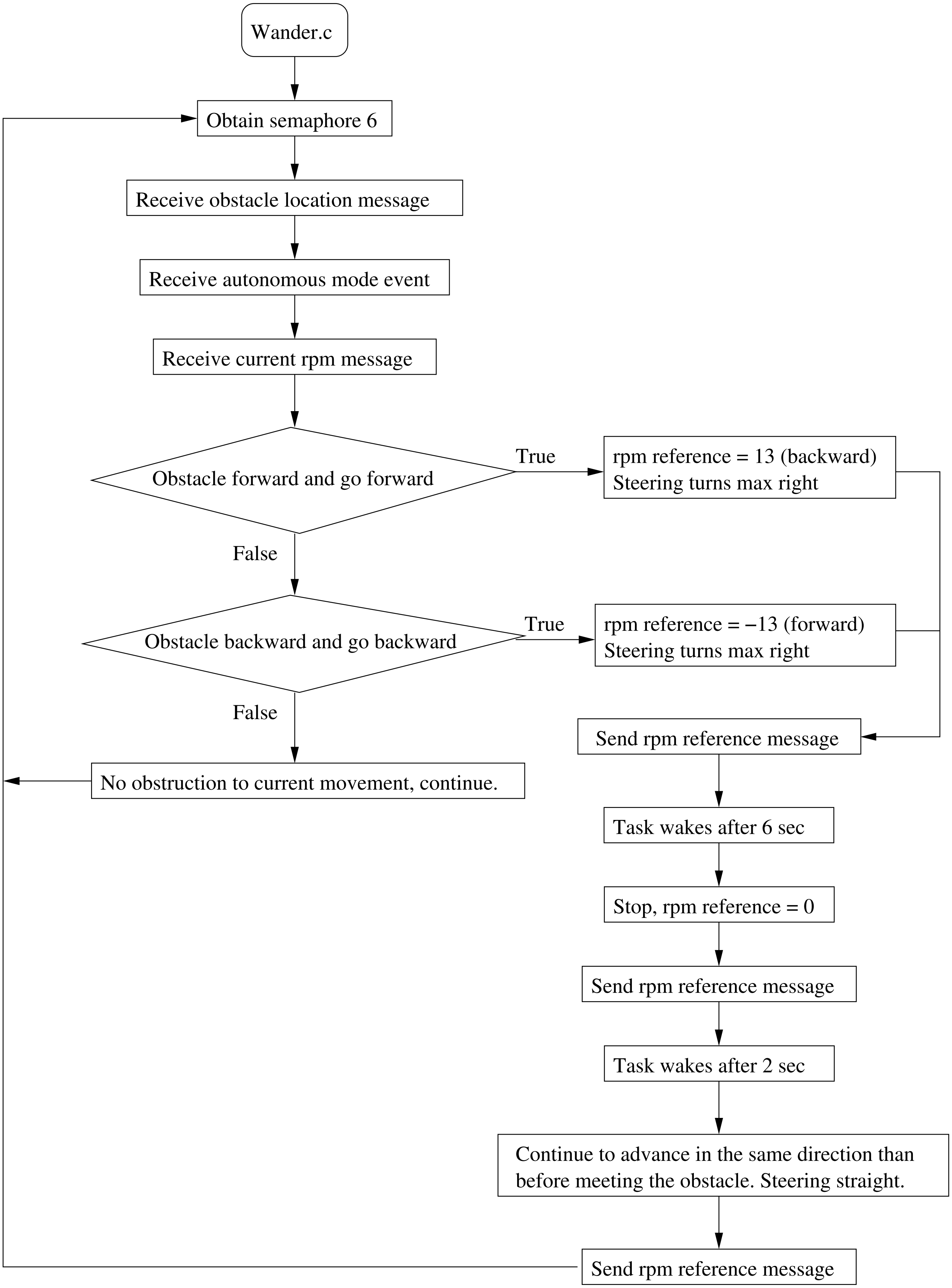
Wander.c
Obtain semaphore 6
Receive obstacle location message
Receive autonomous mode event
Receive current rpm message
Obstacle forward and go forward
True
rpm reference = 13 (backward)
Steering turns max right
False
Obstacle backward and go backward
True
rpm reference = −13 (forward)
Steering turns max right
False
No obstruction to current movement, continue.
Send rpm reference message
Task wakes after 6 sec
Stop, rpm reference = 0
Send rpm reference message
Task wakes after 2 sec
Continue to advance in the same direction than before meeting the obstacle. Steering straight.
Send rpm reference message

Figure 13: *Wander task diagram*

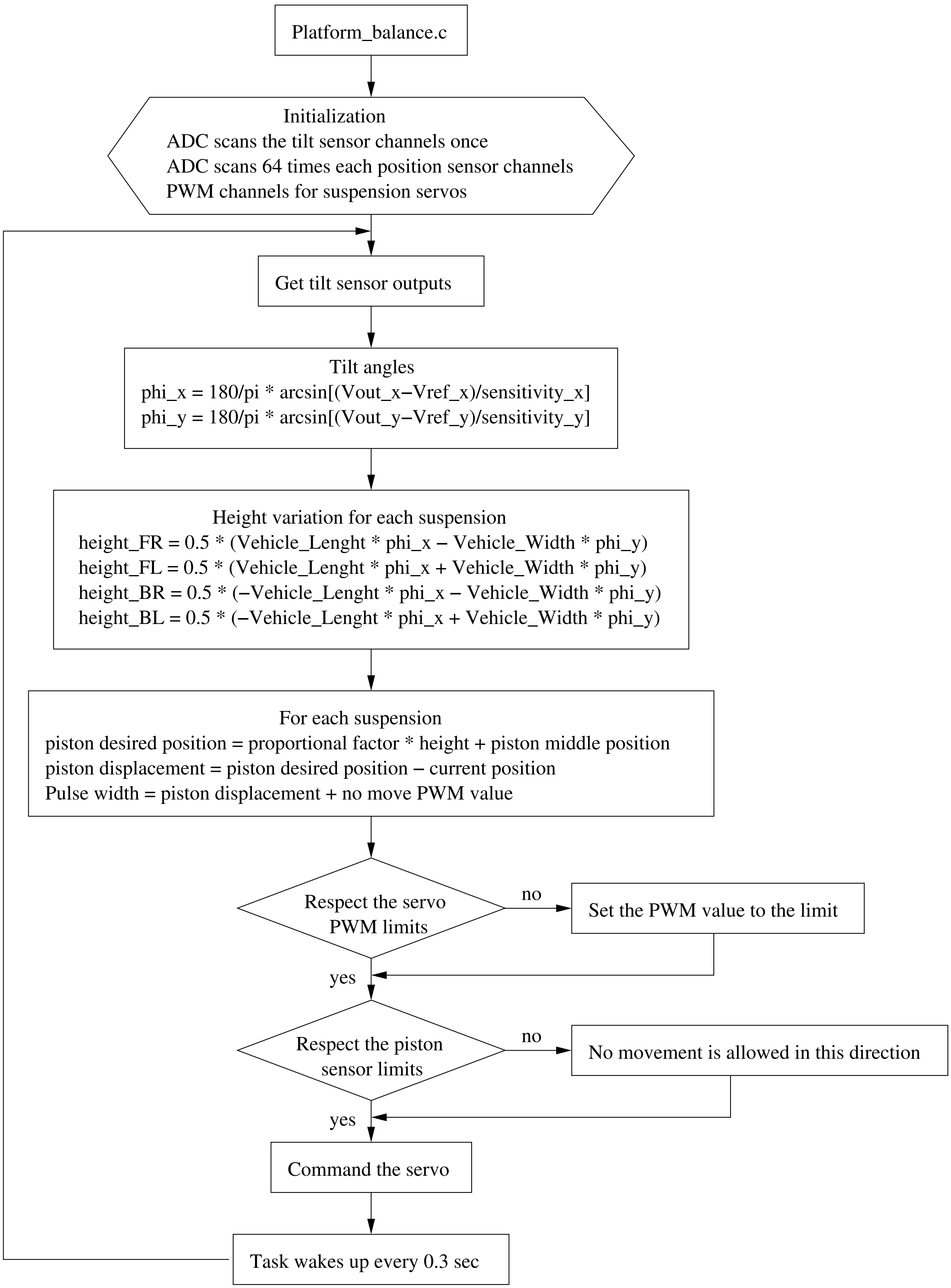
Platform_balance.c
Initialization
ADC scans the tilt sensor channels once
ADC scans 64 times each position sensor channels
PWM channels for suspension servos
Get tilt sensor outputs
Tilt angles
phi_x = 180/pi * arcsin[(Vout_x−Vref_x)/sensitivity_x]
phi_y = 180/pi * arcsin[(Vout_y−Vref_y)/sensitivity_y]
Height variation for each suspension
height_FR = 0.5 * (Vehicle_Lenght * phi_x − Vehicle_Width * phi_y)
height_FL = 0.5 * (Vehicle_Lenght * phi_x + Vehicle_Width * phi_y)
height_BR = 0.5 * (−Vehicle_Lenght * phi_x − Vehicle_Width * phi_y)
height_BL = 0.5 * (−Vehicle_Lenght * phi_x + Vehicle_Width * phi_y)
For each suspension
piston desired position = proportional factor * height + piston middle position
piston displacement = piston desired position − current position
Pulse width = piston displacement + no move PWM value
Respect the servo PWM limits
no
Set the PWM value to the limit
yes
Respect the piston sensor limits
no
No movement is allowed in this direction
yes
Command the servo
Task wakes up every 0.3 sec

Figure 14: *Platform balance diagram*

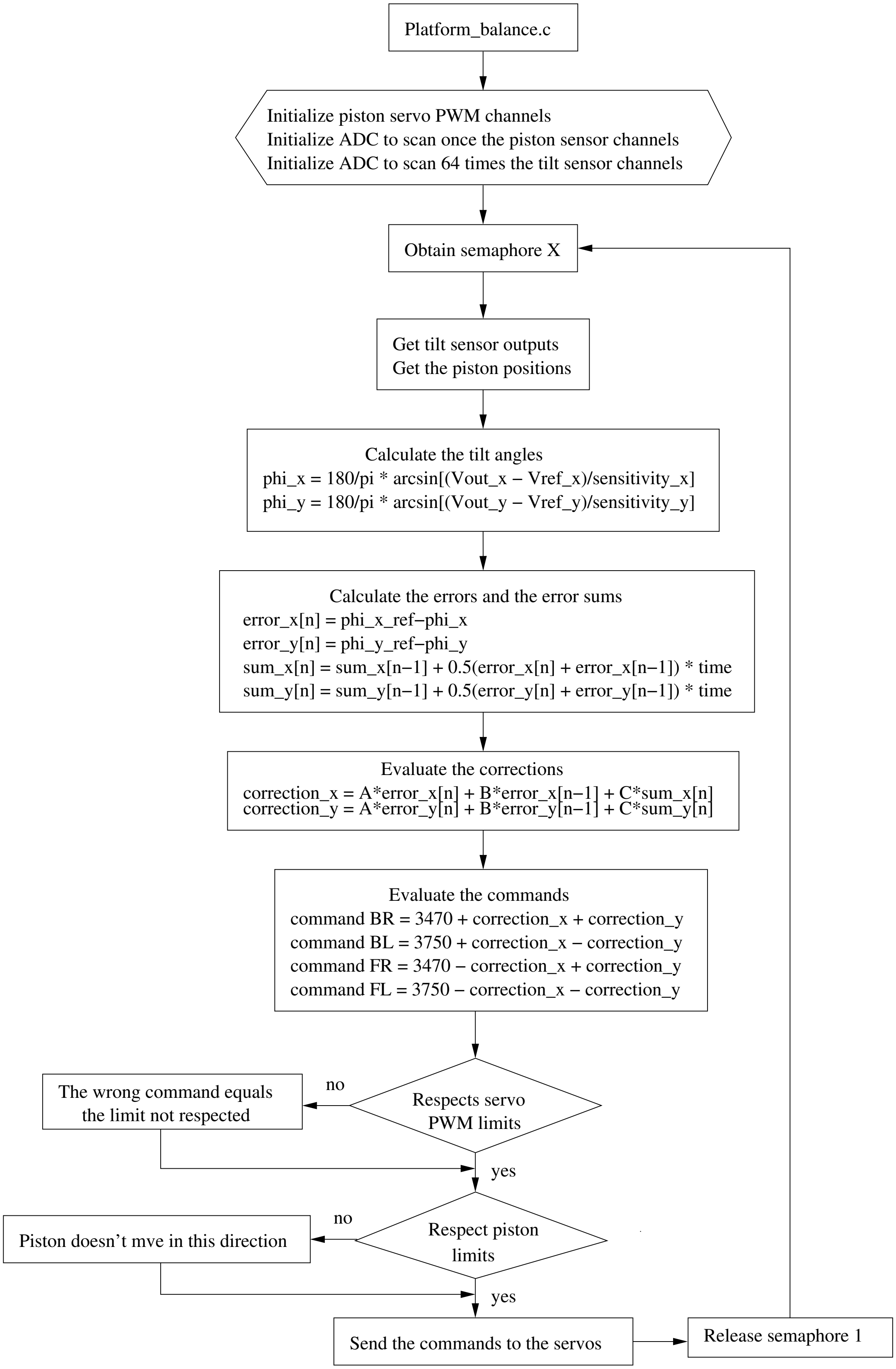
Platform_balance.c
Initialize piston servo PWM channels
Initialize ADC to scan once the piston sensor channels
Initialize ADC to scan 64 times the tilt sensor channels
Obtain semaphore X
Get tilt sensor outputs
Get the piston positions
Calculate the tilt angles
phi_x = 180/pi * arcsin[(Vout_x − Vref_x)/sensitivity_x]
phi_y = 180/pi * arcsin[(Vout_y − Vref_y)/sensitivity_y]
Calculate the errors and the error sums
error_x[n] = phi_x_ref−phi_x
error_y[n] = phi_y_ref−phi_y
sum_x[n] = sum_x[n−1] + 0.5(error_x[n] + error_x[n−1]) * time
sum_y[n] = sum_y[n−1] + 0.5(error_y[n] + error_y[n−1]) * time
Evaluate the corrections
correction_x = A*error_x[n] + B*error_x[n−1] + C*sum_x[n]
correction_y = A*error_y[n] + B*error_y[n−1] + C*sum_y[n]
Evaluate the commands
command BR = 3470 + correction_x + correction_y
command BL = 3750 + correction_x − correction_y
command FR = 3470 − correction_x + correction_y
command FL = 3750 − correction_x − correction_y
The wrong command equals the limit not respected
no
Respects servo PWM limits
yes
Piston doesn’t mve in this direction
no
Respect piston limits
yes
Send the commands to the servos
Release semaphore 1

Figure 15: Platform PID loop diagram

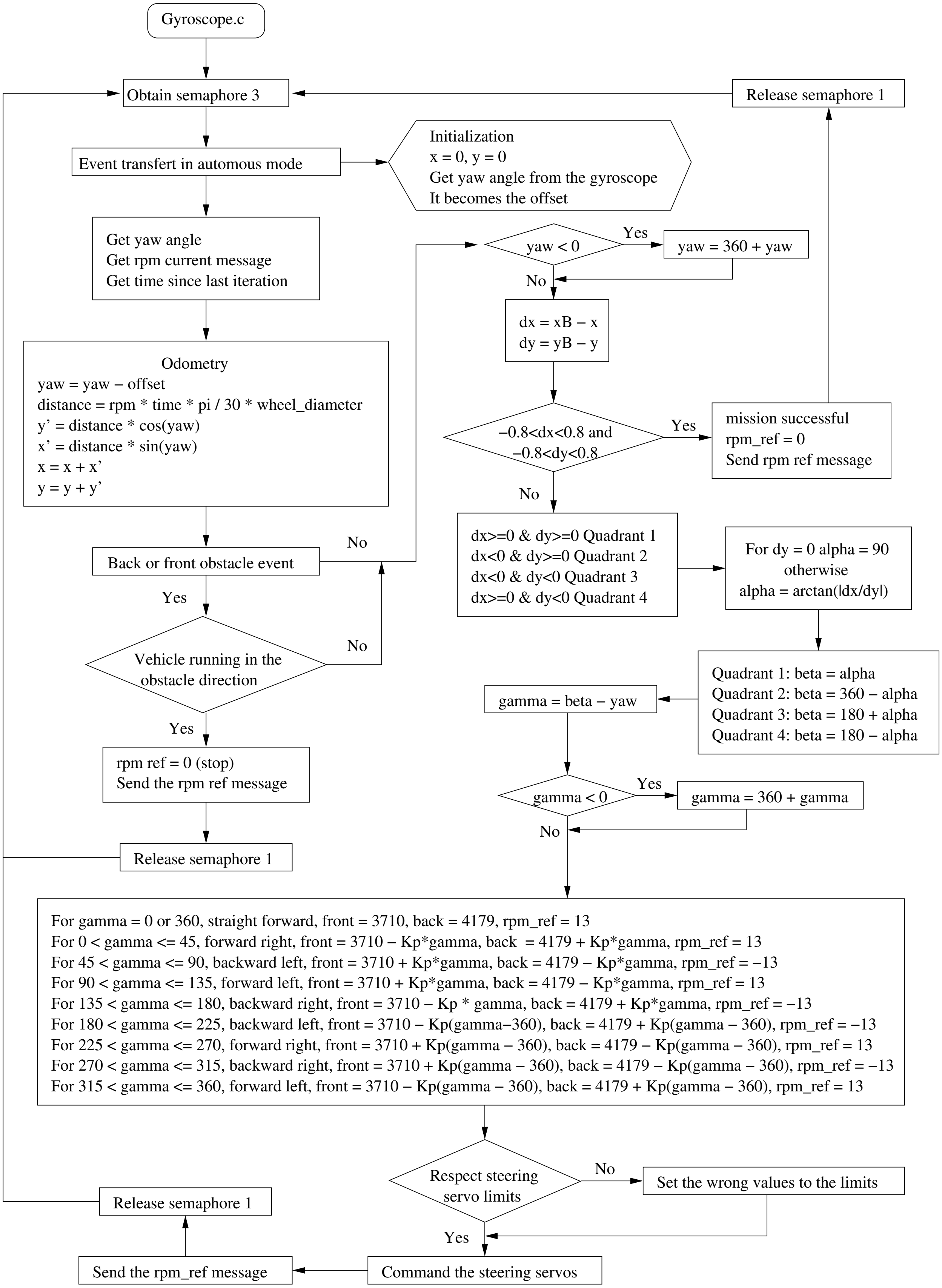
Gyroscope.c
Obtain semaphore 3
Release semaphore 1
Event transfert in automous mode
Initialization
x = 0, y = 0
Get yaw angle from the gyroscope
It becomes the offset
Get yaw angle
Get rpm current message
Get time since last iteration
yaw < 0
Yes
yaw = 360 + yaw
No
dx = xB − x
dy = yB − y
Odometry
yaw = yaw − offset
distance = rpm * time * pi / 30 * wheel_diameter
y' = distance * cos(yaw)
x' = distance * sin(yaw)
x = x + x'
y = y + y'
−0.8<dx<0.8 and
−0.8<dy<0.8
Yes
mission successful
rpm_ref = 0
Send rpm ref message
No
No
Back or front obstacle event
dx>=0 & dy>=0 Quadrant 1
dx<0 & dy>=0 Quadrant 2
dx<0 & dy<0 Quadrant 3
dx>=0 & dy<0 Quadrant 4
For dy = 0 alpha = 90
otherwise
alpha = arctan(|dx/dy|)
Yes
No
Vehicle running in the
obstacle direction
Quadrant 1: beta = alpha
Quadrant 2: beta = 360 − alpha
Quadrant 3: beta = 180 + alpha
Quadrant 4: beta = 180 − alpha
gamma = beta − yaw
Yes
rpm ref = 0 (stop)
Send the rpm ref message
gamma < 0
Yes
gamma = 360 + gamma
No
Release semaphore 1
For gamma = 0 or 360, straight forward, front = 3710, back = 4179, rpm_ref = 13
For 0 < gamma <= 45, forward right, front = 3710 − Kp*gamma, back = 4179 + Kp*gamma, rpm_ref = 13
For 45 < gamma <= 90, backward left, front = 3710 + Kp*gamma, back = 4179 − Kp*gamma, rpm_ref = −13
For 90 < gamma <= 135, forward left, front = 3710 + Kp*gamma, back = 4179 − Kp*gamma, rpm_ref = 13
For 135 < gamma <= 180, backward right, front = 3710 − Kp * gamma, back = 4179 + Kp*gamma, rpm_ref = −13
For 180 < gamma <= 225, backward left, front = 3710 − Kp(gamma−360), back = 4179 + Kp(gamma − 360), rpm_ref = −13
For 225 < gamma <= 270, forward right, front = 3710 + Kp(gamma − 360), back = 4179 − Kp(gamma − 360), rpm_ref = 13
For 270 < gamma <= 315, backward right, front = 3710 + Kp(gamma − 360), back = 4179 − Kp(gamma − 360), rpm_ref = −13
For 315 < gamma <= 360, forward left, front = 3710 − Kp(gamma − 360), back = 4179 + Kp(gamma − 360), rpm_ref = 13
Respect steering
servo limits
No
Set the wrong values to the limits
Release semaphore 1
Yes
Send the rpm_ref message
Command the steering servos

Figure 16: *A-to-B mobility diagram*

6. Conclusion

The document has described the XTB: the mechanical components, the sensors and the microprocessor. Being small, the R/C truck can easily be manipulated by a human. Some other advantages are that it is inexpensive and research using this vehicle can, later, be transfered to bigger and more expensive platforms.

The XTB is useful for testing technologies, concepts and algorithms, and familiarization with the powerful MPC555 microprocessor and the real-time operating system RTEMS. The program algorithms presented in this memorandum are mainly an active platform to keep it horizontal when the vehicle tilts, a teleoperation mode to direct manually the robot, a wander mode to run randomly avoiding the obstacles, and a rudimentary A-to-B mobility to go from point A to point B autonomously. These tasks involve other parts as the IR sensors to detect obstacles, the encoder to get the speed, the tilt sensors to know the platform tilt and the gyroscope to obtain odometry data.

With the first control level complete, a second control level may be added that provide navigation and arbitration. The installation of a camera would be useful for the navigation control level.

References

1. (2003). RTEMS C User's Guide. On-Line Applications Research Corporation (OAR).

2. Dees, R. and Loeliger, J. (2002). General TPU C Functions for the MPC500 Family - AN2360/D. Motorola Inc.

3. (2000). MPC555/MPC556 User's Manual. Motorola Inc.

4. (1996). TPU Time Processor Unit Reference Manual - TPURM. Motorola Inc.

5. Goler, V. (2004). Using the New Input Transition/Input Capture TPU Function (NITC) with the MPC500 Family - AN2366/D. Freescale Semiconductor.

6. Anderson, K. (1997). Frequency Measurement TPU Function (FQM) - TPUPN03/D. Motorola Inc.

7. Dees, R. (2002). Using the Frequency Measurement TPU Function (FQM) with the MPC500 Family - AN2363/D. Motorola Inc.

8. (2003). 3DM-G User Manual Gyro Enhanced Orientation Sensor. Firmware version 1.3.00 ed. MicroStrain Inc.. 310 Hurricane Lane, Williston, VT 05495 USA.

Annex A
Calibration of the servos

Each servo has been calibrated to get a relation between the pulse width and the resulting angle for the steering and the IR sensor bumpers, or the resulting speed for the suspension pistons. The simple calibration method employed has been to send a specific PWM value and to measure the resulting angle or speed.

A.1 IR sensors

To evaluate the infrared sensor angle, a protractor has been fixed to the bumper end so the zero degree indicates the horizontal orientation of the sensor. At this position, the front sensor detects straight in front of the vehicle. A positive angle corresponds to an upper detection than the horizontal. Figure A.2 for the rear bumper and Figure A.1 for the front one, present the regression line obtained for each graphic with its equation. A correlation factor (R) indicates the goodness of the fitting curve. The fit is very good in both cases with R = 0.9982 for the rear sensors and R = 0.9944 for the front ones.

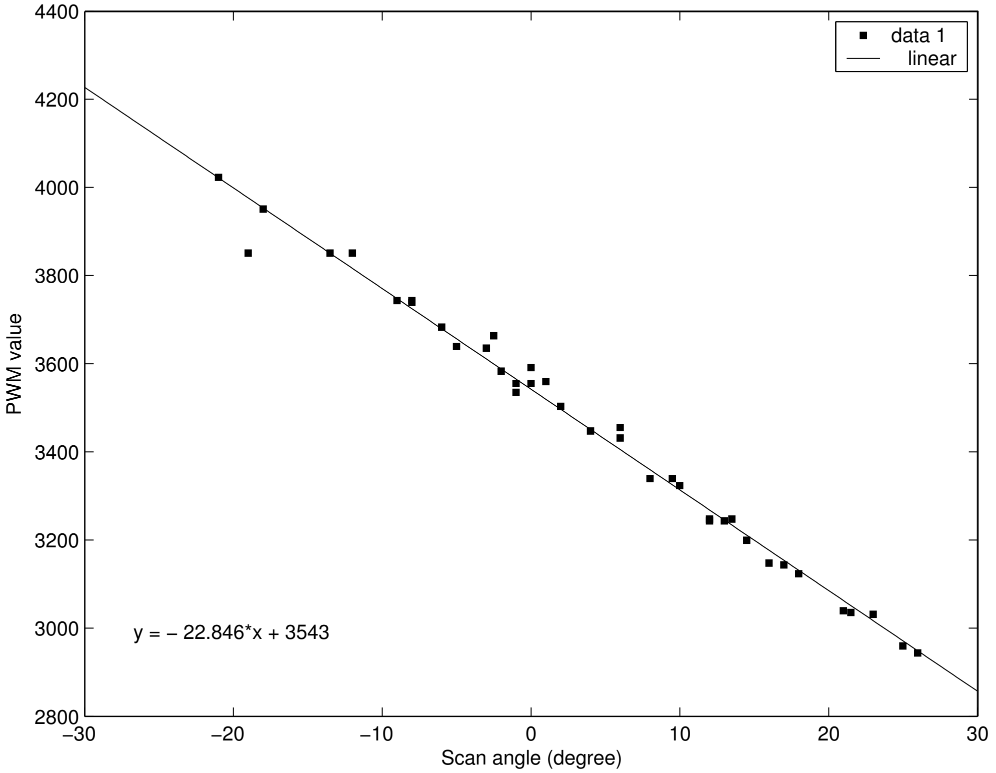

Figure A.1: *Front IR scan servo calibration curve*

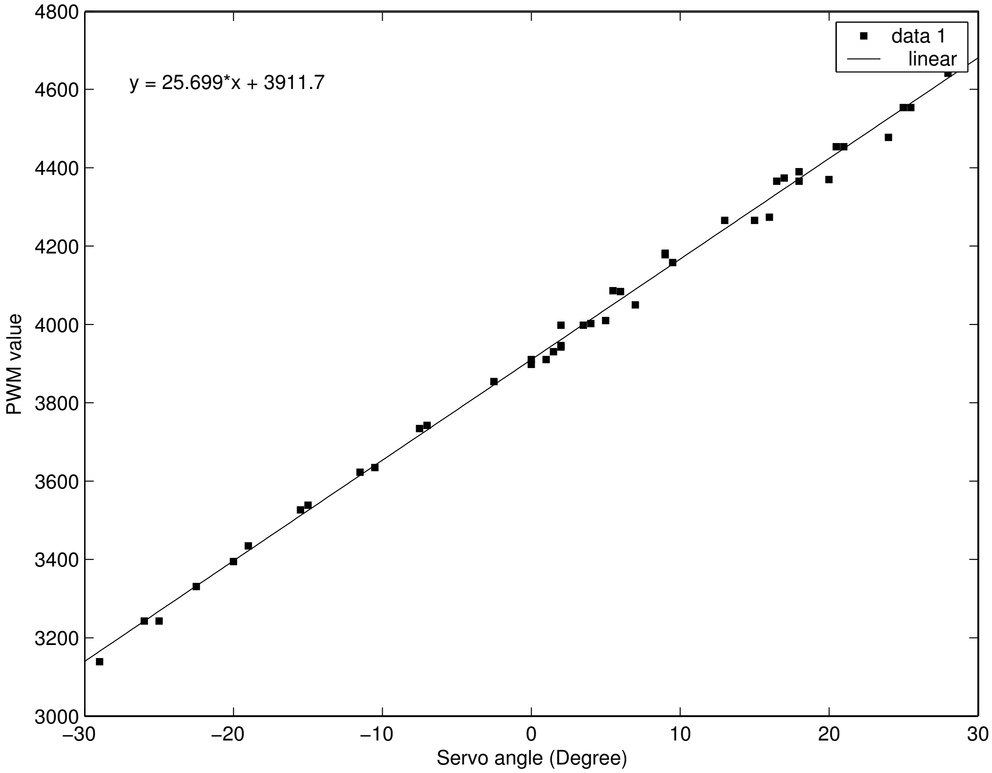

Figure A.2: *Back IR scan servo calibration curve*

A.2 Steering

For the steering servos calibration, the calibration is more complex than for the bumper angles. When the steering moves, the center of the tire tread doesn't stay at the same place. It describes a circular motion around the bracket pivot. Thus, it is impossible to install the protractor above the tire and measure the variation in orientation for different PWM commands. The alternative technique is to mark a line parallel to the tire wall for each PWM command. Then, each line is compared angularly to the line obtained at zero angle. Thus, it is possible to get a relation between the pulse width and the steering angle. Figure A.3 and A.4 present this relation for the front and the back steering respectively. The calibration method is not very accurate and the position of the tire is not exactly the same for a same PWM command. The steering rod clearance induces hysteresys with the floor friction and inaccuracy. This increases the error in odometry data. Nonetheless, the fitting curve for the front steering calibration has a correlation factor of 0.9964 and the back steering calibration of 0.9967. Even if the linear regression fit well, the mechanism and the measurement method are not accurate. Consequently, another way to determine the steering angle is required. It is why a linear sensor should be installed on the steering rod. It would give the exact angle of the steering. By looping a correction function, it would be possible to modify the PWM value and to get the desired steering angle.

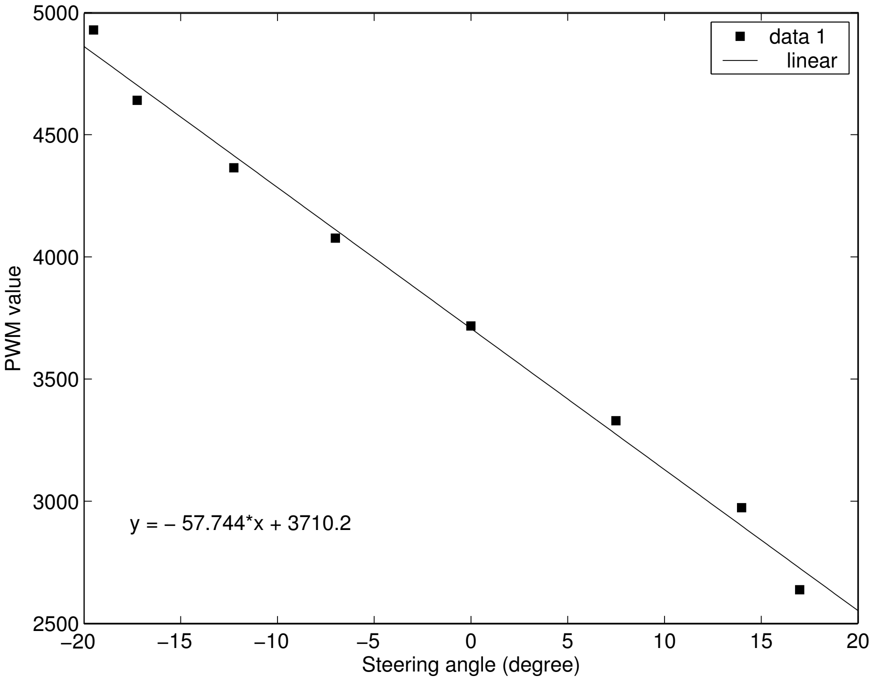

Figure A.3: *Front steering servo calibration curve*

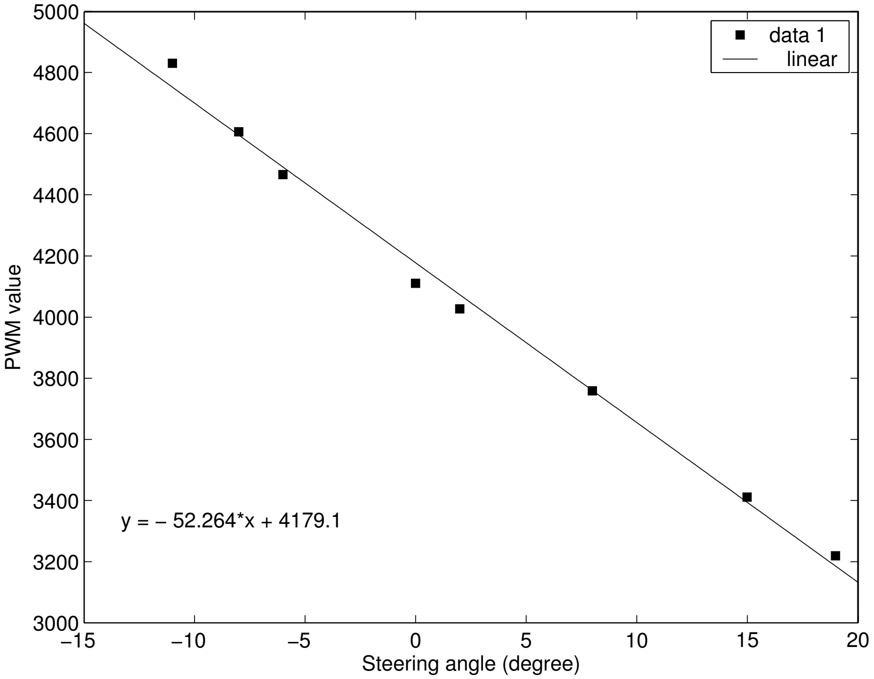

Figure A.4: *Back steering servo calibration curve*

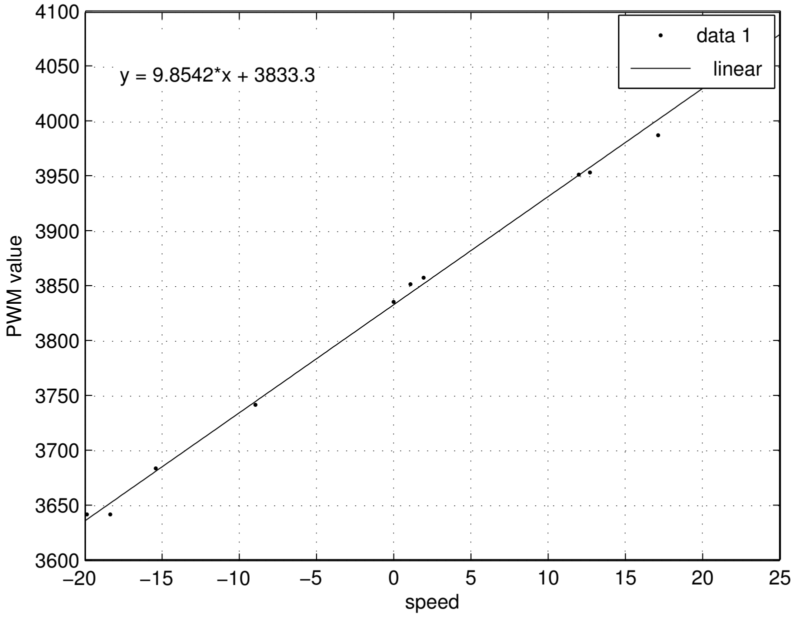

Figure A.5: *Back-left piston relation between speed and PWM*

A.3 Suspension piston

The suspension calibration is easier than for the steering, but not very accurate. A software algorithm samples the time, the piston position sensor output and the servo PWM value. The pulse width stays constant for several samplings before to be changed. After many trials, a graphic of the relation between the piston speed and the servo pulse width is built for each suspension. Figures A.6, A.5, A.7 presents the results. A linear fitting line gives the approximative equation of the system reaction. Even if the system doesn't seem to be linear, this approximation is good enough for the present application. The front-left piston graph has not been done since it was possible to conclude with the others that a linear estimation would be appropriate for the robot activity. The fittness curve data are not utilized in the platform balance algorithm since they are not accurate and change too much from one trial to another. The linear constants used in the program have been set by trials and errors to get a good reaction of the system.

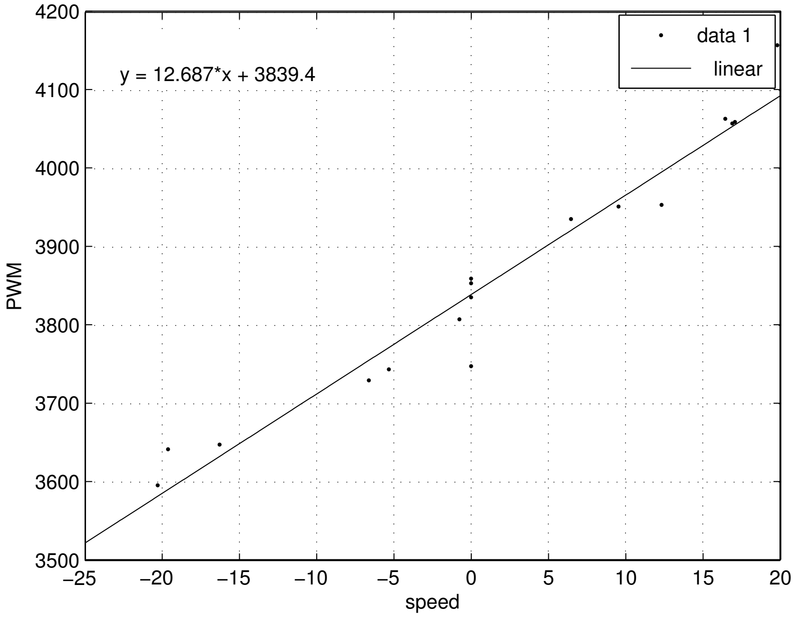

Figure A.6: *Back-right piston relation between speed and PWM*

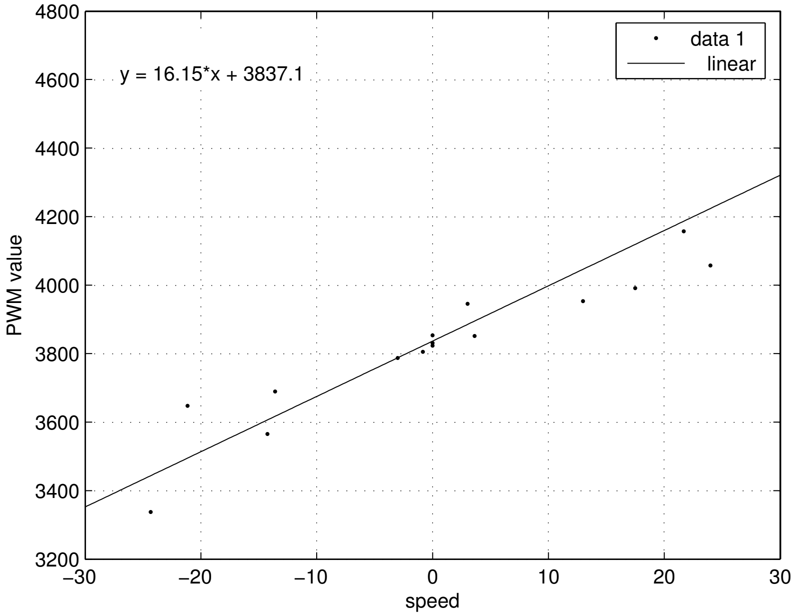

Figure A.7: *Front-right piston relation between speed and PWM*

Annex B
List of Acronyms

The list shown in Table B.1 defined every acronyms used in this document.

Acronym	Definition
ADC	Analog-to-Digital Converter
BL	Back left
BR	Back right
CPU	Central Processor Unit
FQM	Frequency Measurement Function
FL	Front left
FR	Front right
IPC	Inter Process Communication
IR	Infrared sensors
KD	Derivative constant
KI	Integrative constant
KP	Proportional constant
NITC	New Input Transition / Input Capture Function
PWM	Pulse Width Modulation
RTEMS	Real Time Executive for Multiprocessor Systems
SCI	Serial Communication Interface
TCR	Time Count Register
TPU	Time Processor Unit
XTB	EXtreme Test Bed

Table B.1: *Acronym list*

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (the name and address of the organization preparing the document. Organizations for who the document was prepared, e.g. Establishment sponsoring a contractor's report, or tasking agency, are entered in Section 8.)

Defence R&D Canada – Suffield
PO Box 4000, Station Main
Medicine Hat, AB T1A 8K6

2. SECURITY CLASSIFICATION (overall security classification of the document, including special warning terms if applicable)

UNCLASSIFIED

3. TITLE (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title).

The eXtreme Test Bed vehicle for indoor environments (U)

4. AUTHORS (Last name, first name, middle initial. If military, show rank, e.g. Doe, Maj. John E.)

Vincent, Isabelle

5. DATE OF PUBLICATION (month and year of publication of document)

December 2004

6a. NO. OF PAGES (total containing information, include Annexes, Appendices, etc) 55

6b. NO. OF REFS (total cited in document) 8

7. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Technical Memorandum

8. SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include the address.)

Defence R&D Canada – Suffield

9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.)

42zz78

9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.)

10a. ORIGINATOR'S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique to this document.)

DRDC Suffield TM 2004-262

10b. OTHER DOCUMENT NOs. (Any other numbers which may be assigned this document either by the originator or by the sponsor.)

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

(x) Unlimited distribution
( ) Distribution limited to defence departments and defence contractors; further distribution only as approved
( ) Distribution limited to defence departments and Canadian defence contractors; further distribution only as approved
( ) Distribution limited to government departments and agencies; further distribution only as approved
( ) Distribution limited to defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally corresponded to the Document Availability (11). However, where further distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected).

Unlimited

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C) or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

The eXtreme Test Bed is a small-scale remotely controlled truck that has been retrofitted with a microprocessor, sensors, servos and actuators to experiment with different concepts and software applications in an indoor environment. Two intelligence levels are exploited. A low level vehicle control is realized by the MPC555 microprocessor for reactive avoidance, wandering state and A-to-B mobility. The second is a navigation control level presently in development. This technical memorandum focuses on the first control level. It details the MPC555 functionalities, program functions, mathematics, algorithms and components used to control the eXtreme Test Bed. The objective is to test technologies and software capabilities before application to more expensive platforms. A second goal is to become familiar with a real-time operating system executing several tasks in parallel.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifies, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

Robotics, UGV, small platform control, MPC555 microcontroller, RTEMS real-time operating system