Faculty of Engineering, Computer
& Mathematical Sciences

School of Mechanical Engineering

THE UNIVERSITY OF ADELAIDE
AUSTRALIA

EDGAR, A Self-Balancing Scooter

Final Report

October 27, 2005

Authors

M. A. Clark
J. B. Field
S. G. McMahon
P. S. Philps

Supervisor

Dr. B. S. Cazzolato


EDGAR

Executive Summary

This report covers the design and testing of a two wheeled self-balancing vehicle (EDGAR) capable of carrying a human by maintaining the wheels underneath the rider's centre of gravity.

The aim of the project was to design and build a self-balancing scooter that functions similarly to the Segway Human Transporter (HT) (the first and only self-balancing vehicle to be commercially available). EDGAR's design draws upon the successes and failures of the Segway HT and other attempts at producing self-balancing scooters which utilise various automatic control methods. Angular feedback from a gyroscopic sensor and PWM output to motors are used in a control system to achieve balance of EDGAR.

The process EDGAR goes through to self balance is similar to how a human balances. The human brain recognises the force due to gravity on the vestibular system and is able to discern the direction it is coming from. The brain then sends impulses to the muscles in the limbs to help provide balance. Similarly, the microcontroller receives information from sensors, interprets the information and then sends commands to the drive system to maintain balance.

EDGAR was designed to be robust and easy to use whilst not compromising on strength. The design included special attention to aesthetics of EDGAR and the ergonomics of the rider/vehicle interface. After some small changes to the initial design, the completion of a fully functioning prototype was achieved. EDGAR satisfies all the basic specifications and project goals and is enjoyable to ride.

Disclaimer

This report is the work of the four authors. Any information that has been obtained from other sources has been referenced where used.

.....

Miguel Clark

Date:

.....

James Field

Date:

.....

Paul Philps

Date:

.....

Simon McMahon

Date:

Acknowledgements

The authors would like to express their sincere thanks to the many people who have contributed to the success of EDGAR. In particular the project supervisor, Dr Benjamin Cazzolato, for his guidance and assistance throughout the year. The authors appreciate the work of Dr Frank Wornle on the real-time MC9S12 target and his help with the implementation of the MiniDRAGON+ on-board EDGAR.

The authors would like to thank all the technicians from the Mechanical Workshop, including Bill Finch and Richard Pateman. A special thanks must go to Steve "ey Steve!" Kloeden whose dedicated efforts to make us laugh, as well as supply, improve and fix the mechanical parts, was much appreciated.

The Instrumentation Workshop has also played a large part in the completion of EDGAR. The authors would like to thank Mr Derek Franklin and, in particular, Mr Silvio De Ieso whose supreme knowledge and skills in all things electronic saved the group many headaches.

And finally, the authors would like to thank all their families and friends for their support throughout the year.

Contents

List of Figures

Chapter 1

Introduction

> An idea that is not dangerous is unworthy of being called an idea at all.
>
> *Oscar Wilde (1854-1900)*

Dean Kamen, inventor and entrepreneur, first penned the idea of a revolutionary new type of personal transportation during the mid 1990's. Nowadays, his invention, the Segway Human Transporter (HT), is a common sight in America and is sold around the world. As of early 2005, it is available for purchase in Australia. The Segway HT is a vehicle which has two coaxial wheels driven independently by a controller that balances the vehicle both with and without a rider. The balancing is regulated by feedback from an array of tilt sensors and gyroscopes. The controller uses advanced State Space (SS) control making the system very robust and responsive. It is robust enough to accept riders of different weights and responsive enough to provide adequate balancing for different riders and riding styles.

The aim of the project was to investigate, research, design, and build a self-balancing, coaxial scooter loosely based around the commercially available Segway HT. From this short design brief, 'EDGAR - a self-balancing scooter' was born. EDGAR is an acronym which literally stands for Electro-Drive Grav-Aware Ride. It was important that EDGAR be of easy manufacture, use off-the-shelf parts where possible, provide adequate balancing and be aesthetically pleasing. A person of average weight and height must be able to ride EDGAR for at least one hour at half of the peak load. It should also provide adequate safety measures to ensure the safety of the rider.

The method that the group used to approach the project was to first read and undertake a critical review of literature relevant to the project. Material covered includes mobile inverted pendulums, the Segway HT, Segway HT clones, self-balancing robots, and new ideas in Personal Electric Vehicles (PEV). Once the thorough literature review had been completed, the group began conceptualising the hardware and software design. In particular, a mathematical model of the system was developed and implemented into MATLAB's Simulink component (Mathworks Inc 2005). Using Simulink, the group was able to prototype the control system completely in Virtual Reality (VR). The hardware components were interfaced to Simulink to allow rapid prototyping of the control system with the hardware connected. This was achieved through dSPACE ControlDesk (dSPACE Inc 2005) on the computer and a dSPACE external interface (otherwise known as a Breakout Box (BB)). The control system developed was a classical Proportional Derivative (PD) control system.

Once a working control system was achieved in software, EDGAR was assembled and the hardware attached. The final goal of this project was to have EDGAR working, in accordance with the group's specifications, tethered to the dSPACE Breakout Box and a power supply. Extension goals for the project included cross-compiling the Simulink model onto a MC9S12 based MiniDRAGON+ microcontroller development board and including an on-board battery pack to provide power. It was also desired that the tethered model be controlled with a SS control system and then the system implemented untethered.

The research the group undertook is significant because it is believed that an attempt to build a full scale self-balancing scooter has not been undertaken by an Australian university before. It was also significant because it provides The University of Adelaide and its personnel in the Mechanical Engineering Department with a way of presenting classical control to students and how they can be applied to real-world applications.

This report begins by covering background information pertaining to the project, followed by a review of information relevant to the project, where a critical analysis of recent literature is conducted. The development of the project's goals and specifications are explained in Chapter 3. Chapter 4 explains the development of the control system from beginning to the end system implemented on EDGAR. It is then followed by a detailed explanation of the

different components used on EDGAR in Chapter 5. Chapter 6 describes the implementation of the control system and all the required logic systems in software and the subsequent programming platforms. The integration of all aspects of the hardware is made clear in Chapter 7 where the structural design is justified. Chapter 8 explains the phase of the design process where the different components were assembled for the first time and tested. This chapter also includes the findings of the testing and the actions taken to remedy any adverse results. Chapter 9 includes an in depth analysis of the final design of the vehicle regarding the satisfaction of the project goals and specifications, and also a costing analysis. Chapter 10 finishes the body of the report with the conclusion and future work the authors believe is warranted.

Chapter 2

Background

A literature review was undertaken, focussing on the two integral parts of the project, the mathematical and control theory behind self-balancing scooters as well as the mechanical and aesthetic design of developed models. The review examined, in detail, previous projects related to mobile inverted pendulums, self-balancing vehicles and personal electric vehicles in both research and commercial spheres. Section 2.1 discusses the background associated with balancing control and in particular, the mobile inverted pendulum. Section 2.2 analyses the literature reviewed by the group.

2.1 Fundamental Control Principles

The development of automatic control began in 1769 by James Watt when he invented the flyball governor (Dorf & Bishop 2001). From that time many different systems have emerged as being fundamentally control based, something that has a primary function which demonstrates the concept of automatic control. One of these systems is the mobile inverted pendulum.

Initially thought of as a mass on the end of a rod balanced in one's hand, as shown in Figure 2.1, it has since been mathematically represented as a moving cart of a mass to more accurately measure and derive appropriate equations of motion as shown in Figure 2.2.

In order to begin to understand how a self-balancing scooter would operate, it was apparent that a brief analysis of a mobile inverted pendulum system was required. A simple mobile inverted pendulum model involves a cart of certain mass and a post with a mass at the end. The post is fixed to the cart through

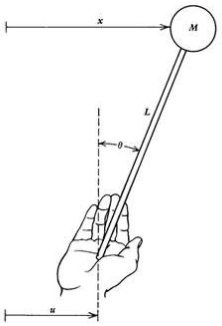


Figure 2.1: Free Body Diagram of inverted pendulum (Sophia University Japan 2005)

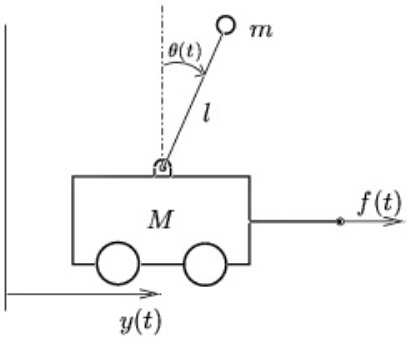


Figure 2.2: Free body diagram of mathematical representation of mobile inverted pendulum (University of Newcastle 2005)

a pivot joint just above the axle, and is free to rotate in one plane only. The only force that is applied is a horizontal force on the cart, which results in subsequent movement of the cart and post. The equations of motion for the system are derived from a force balance in the horizontal plane and torque balance about the pivot point. In order to simplify the analysis of the system, most texts then linearise the equations about the upright position. This is valid providing all movements of the post are small enough to result in a very small angle of rotation. Equations 2.1 and 2.2 show these simplifications.

$$M\ddot{y} + mL\ddot{\theta} - U(t) = 0 \tag{2.1}$$

$$mL\ddot{y} + mL^2\ddot{\theta} - mLg\theta = 0 \tag{2.2}$$

$$G(s) = \frac{-1/LM}{s^2 - g/L} \tag{2.3}$$

The equations are combined to remove the horizontal acceleration component, rearranged and then further simplified by assuming that the mass of the base is much greater than that at end of the post. This results in a transfer function relating the input force to the angular acceleration of the post, shown in Equation 2.3. This enabled a Simulink model of the system to be built and tested, as shown in Figure 2.3.

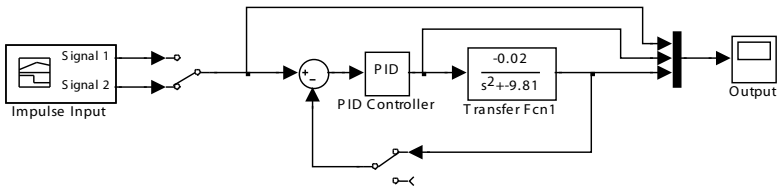


Figure 2.3: Simulink model of mobile inverted pendulum

Figure 2.4 shows the response to a unit impulse input, with the post angle output increasing exponentially as gravity pulls the post to the ground. It was assumed that this response would be similar to the response of a self-balancing scooter without a rider receiving a push or attempting to move.

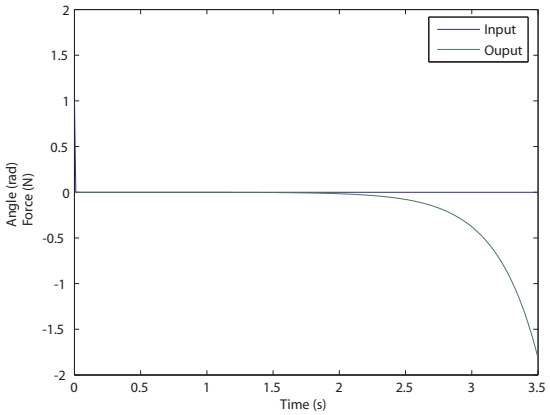


Figure 2.4: MATLAB plot of unit step response

Through the use of a suitable controller, it is possible to apply forces to the cart in order to compensate for a disturbance input, thus keeping the post

upright. The response of a model with feedback PID control, discussed further in Chapter 4, is shown in Figure 2.5.

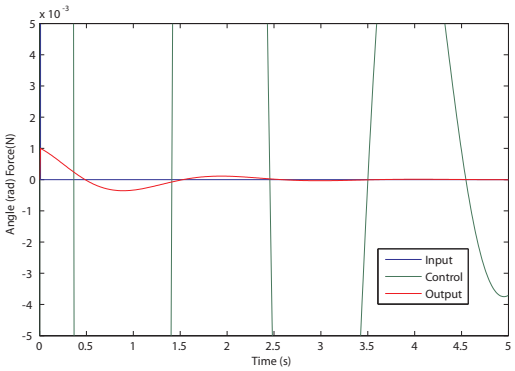


Figure 2.5: MATLAB plot of PID controller response

This simple mobile inverted pendulum model proved to be a useful starting point for the design of a controller for EDGAR.

2.2 Recent Research and Development

This section provides an overview of previous efforts to construct self-balancing devices. The knowledge gained in reviewing these endeavours was extremely useful in the preliminary design of EDGAR.

2.2.1 The Segway Model

The Segway model, as shown in Figure 2.6, has been reviewed and analysed from two different sources, the first being the Segway website (Segway Inc 2005) and the second, the How Stuff Works website dedicated to the Segway and how it functions (How Stuff Works Inc 2005).

The Segway HT is the only commercially available self-balancing vehicle in the world to date. According to How Stuff Works Inc (2005), the Segway HT began it's life when entrepreneur and engineer, Dean Kamen, slipped while exiting the shower and his body threw himself backwards to try and counteract the slip. Even though he crashed to the floor, he began thinking afterwards that if the human body can respond that quickly, a machine should also be able to. Having previously patented a portable kidney dialysis machine, Kamen went about brainstorming ideas to help people with this new idea of

Figure 2.6: Photographs of the Segway i180 (Segway Inc 2005)

machine motion. It is not widely known that Kamen's first venture into self-balancing vehicles resulted in the iBot, a motorized wheel chair that is able to climb stairs by rotating two sets of coaxial wheels and using the self-balancing stabilization when only on one set of wheels. During the development of the iBot, Kamen saw the possibility of using the self-balancing principle for an everyday transportation vehicle. The Segway HT was developed by Kamen's own research company, DEKA Research and Development Corporation. In 2001, the first Segway was shown on public television and was made available for purchase in 2002.

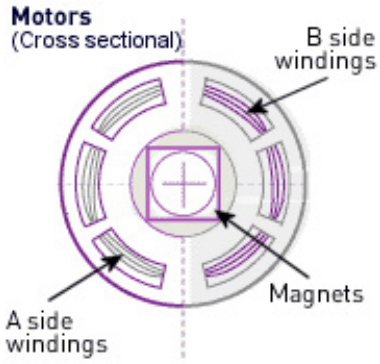


Figure 2.7: Section view of Segway motor (Segway Inc 2005)

How Stuff Works Inc (2005) states that the motors of the Segway are 1.88kW 8000RPM brushless servo electric motors with neodymium-iron-boron perma-

nent magnets driven by a set of twelve high-power, high-voltage field-effect transistors (FET). According to Segway Inc (2005) the motors are the highest power motors for their size and weight ever put into mass production. They are high torque, maintenance free and electrically redundant as each of the motors is monitored by a separate circuit board. The motors are built with two independent sets of windings as shown in Figure 2.7. When the Segway HT functions normally, both sets of windings evenly share the load. Should any of the windings falter, the Segway HT will instantly disable the offending side and use only the other winding to power itself to a safe, controlled stop.

How Stuff Works Inc (2005) explains that the two-stage transmission has a compact 24:1 gear ratio for maximum torque transmission efficiency. It uses a helical gear assembly that significantly reduces noise. The team developing the Segway HT configured the two gear mesh frequencies in the gearbox to make the tones exactly two octaves apart, as shown below in Figure 2.8. The gears are also designed to have non-integer gear ratios, so the gear teeth mesh at different points from revolution to revolution which minimises wear on the teeth.

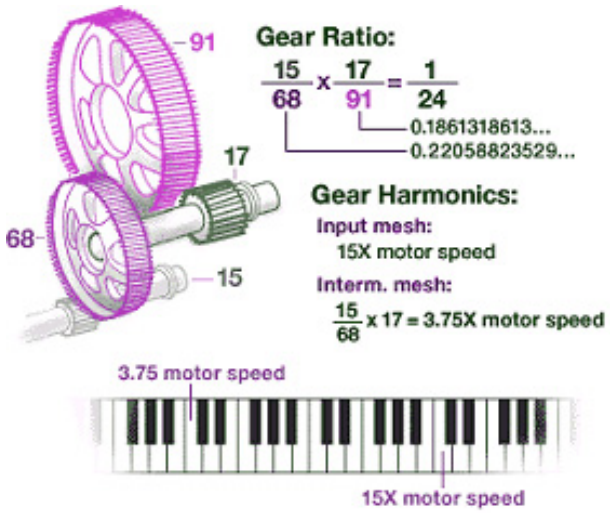


Figure 2.8: Diagram explaining gear mesh frequencies of Segway HT power transmission (Segway Inc 2005)

According to How Stuff Works Inc (2005), the wheels consist of a forged steel wheel hub with a glass-reinforced engineering-grade thermoplastic rim as shown in Figure 2.9. Each wheel is secured to the geared drive shaft with a single nut. The Michelin tyres are made of a silica compound, which provides

good traction even on wet surfaces. The turning system is governed by a throttle type hand grip on the left end of the handlebar. Since the Segway HT only has two wheels, the Segway can rotate on the spot, both wheels going different directions while the control system balances.

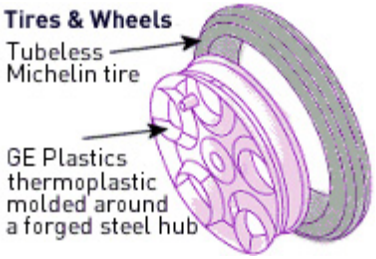


Figure 2.9: Diagram of Segway wheel and tyre (Segway Inc 2005)

How Stuff Works Inc (2005) describes the two rechargeable battery packs that power the Segway HT as shown in Figure 2.10. The original design used Nickel Cadmium (Ni-Cd) battery packs but now they are sold with either 60 cell Nickel Metal Hydride (Ni-MH) packs or 92 cell Lithium-Ion (Li-Ion) packs both designed to output a voltage of 72VDC (Segway Inc 2005). The batteries are constantly monitored by a redundancy system circuit board and there is circuitry on-board allowing them to be recharged with mains electricity. The types of battery influence the range of the Segway, with the Ni-MH packs having a range of 19km while the Li-Ion packs have a range of 39km.

Figure 2.10: Photograph of Segway Ni-MH battery packs (Segway Inc 2005)

How Stuff Works Inc (2005) states that the Segway control and processor system is made up of two circuit boards, housed in the vehicle's chassis as

shown in Figure 2.11. The circuit boards can each function independently in the event of a problem and the system is redundant so if one circuit board fails, the other will bring the vehicle to a stop. The controller boards sample the balance sensor assembly at 100Hz and output commands to the motors at 1000Hz with each board being responsible to one of the two windings in the motors. The Segway HT uses the Texas Instruments TMS320LF2406A Digital Signal Processor (DSP) which runs at 40 millions of instructions per second (MIPS), has 32 kilobytes (kB) of Flash memory and many peripheral communication ports implemented on-board the chip. There is not much literature on how the Segway balancing system works as it is a patented system, however the Segway HT uses the DSPs "to implement digital closed loop motor control and balance computation" Segway Inc (2005).

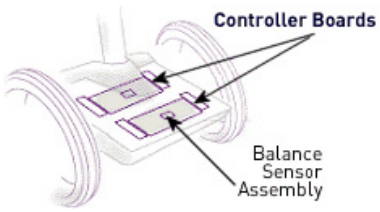


Figure 2.11: Diagram of location of controller boards on the Segway HT (Segway Inc 2005)

The Segway uses five solid-state, vibrating-ring, angular-rate sensors (gyroscopes) and two liquid-filled tilt sensors on the same circuit boards in its balance sensor assembly (BSA), as shown in Figures 2.12 and 2.13, to keep it upright (How Stuff Works Inc 2005). The "gyroscopes" use the Coriolis effect to measure rotation speed. These tiny rings are electromechanically vibrated in such a way that when they are rotated, a small force is generated that can be detected in the internal electronics of the sensor. According to Segway Inc (2005), only three gyroscopes are needed (one on each axes), the extra sensors are included as yet another redundancy system. The balance sensor assembly is mounted below the rider and in between the axle as shown in Figure 2.14. It measures the angle of pitch rotation.

The Segway has four weight sensors built into its platform to tell the computer when a rider has stepped on.

Segway Inc (2005) explains that the Segway HT uses an electronic key system, shown in Figure 2.15, which looks something like a car lighter and

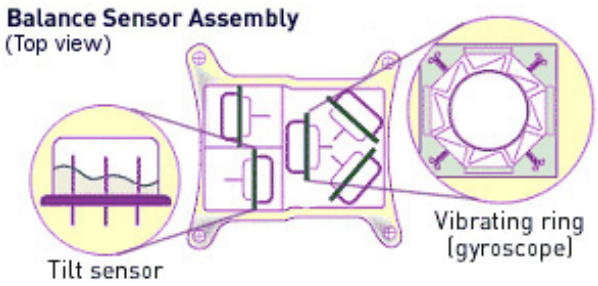


Figure 2.12: Diagram of Segway Balance Sensor Assembly (Segway Inc 2005)

Figure 2.13: Photograph of Segway Balance Sensor Assembly (How Stuff Works Inc 2005)

stores a 64-bit encrypted digital code. The Segway won't start unless the key is plugged into its port. The key can also store settings for vehicle operation like maximum speed limit, power usage, etc. The Segway HT has a small Liquid Crystal Display (LCD) screen, also shown in Figure 2.15, that tells the driver how much battery power is left and how well the vehicle is functioning (How Stuff Works Inc 2005).

The Segway HT's sensitive electronic equipment is housed in a sturdy die-cast aluminium chassis with plastic fairing. According to Segway Inc (2005), the chassis can withstand 7 tons of force. The fairing has a sleek, low-profile design and a scratch-resistant construction an example is shown in Figure 2.16.

Figure 2.14: Diagram of location of balance sensors assembly on the Segway (Segway Inc 2005)

Figure 2.15: Photograph of Segway handlebar, showing key system and LCD (Segway Inc 2005)

Figure 2.16: Photograph of Segway fender (Segway Inc 2005)

The Segway HT is an evolving commercial product and thus has had a few variations of the vehicle as shown in Figure 2.17.

Figure 2.17: Photograph of current versions of Segway available for purchase (Left to Right:i180, p133, XT, GT) (Segway Inc 2005)

According to Segway Inc (2005) the different models have the following different specifications.

Model	i series	p series	XT series	GT series
Speed(km/h)	20	16	20	20
Range(km)	19 Ni-MH /39 Li-Ion	16 Ni-MH	16 Li-Ion	35 Li-Ion
Terrain	Variable	Even	Rugged	Grass
Payload(kg)	113	95.3	118	118
Footprint(cm)	48 x 64	41 x 55	54 x 78	48 x 64
Weight(kg)	38	32	45.4	43

Table 2.1: Table of Segway specifications (Segway Inc 2005).

As can be seen from the Table 2.1, the current models are varied in their design and specifications. The everyday i180 performs equally as well to the GT which is specifically designed for use on golf courses. There is a huge difference between the XT, which is a dedicated off road Segway, and the p133 model, which is a light, small Segway with a tiny footprint. They weight of the models varies from 32kg to 45.4kg. The difference between the Ni-MH battery packs and the Li-Ion battery packs is pronounced if the specifications for the i180 are examined in detail, with the Ni-MH packs giving the i180 a range of 19km while the Li-Ion packs give a range of 39km. An exploded view of the Segway is shown in Figure 2.18.

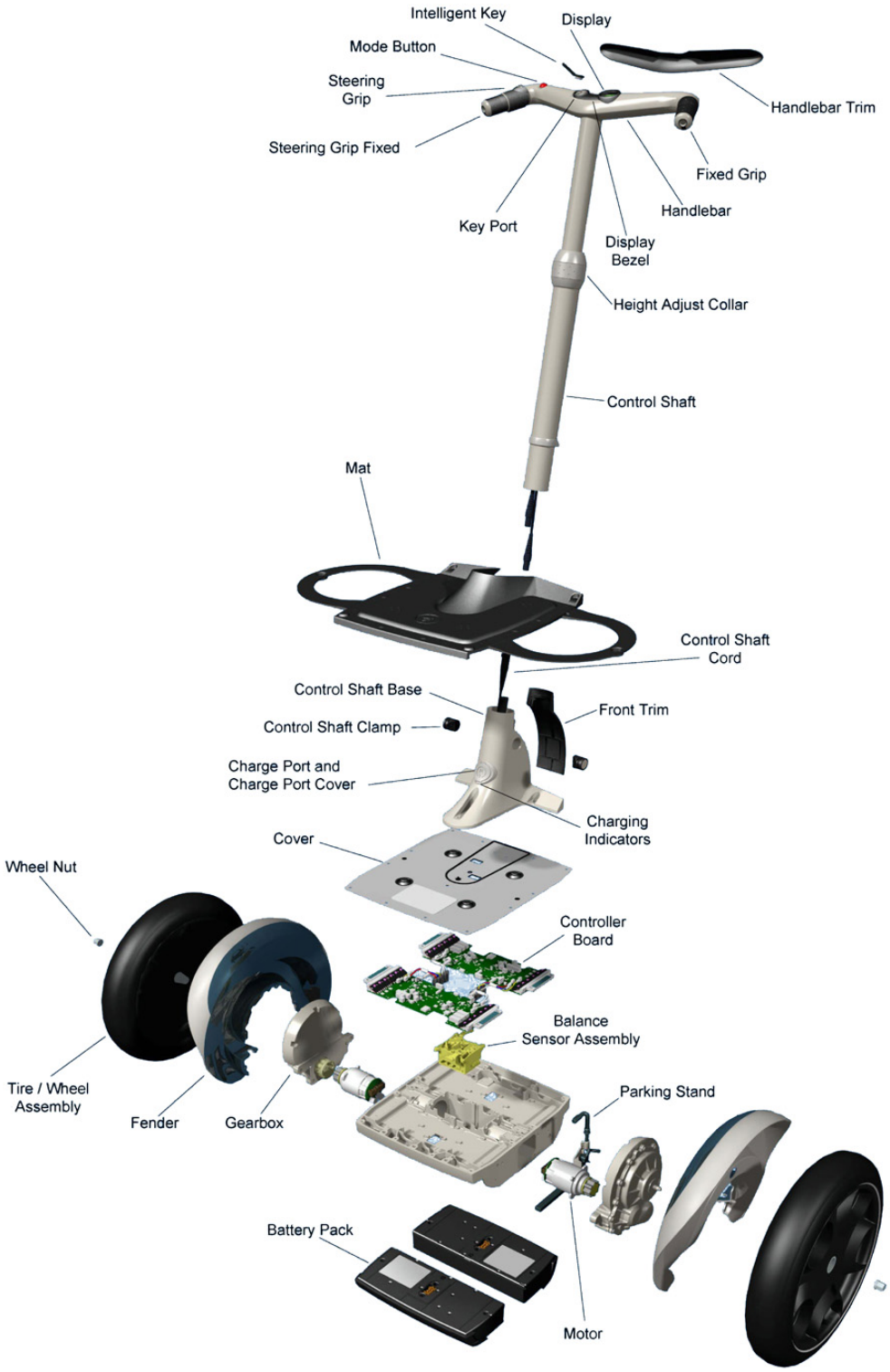


Figure 2.18: Exploded view of the Segway HT (How Stuff Works Inc 2005)

2.2.2 The Grasser Model

In the article, JOE: A Mobile, Inverted Pendulum, Grasser et al. (2002) present an overview of the project with the desired outcome of building a vehicle that balances its driver on two coaxial wheels using automatic control. JOE: A Mobile, Inverted Pendulum is shown in Figure 2.19.

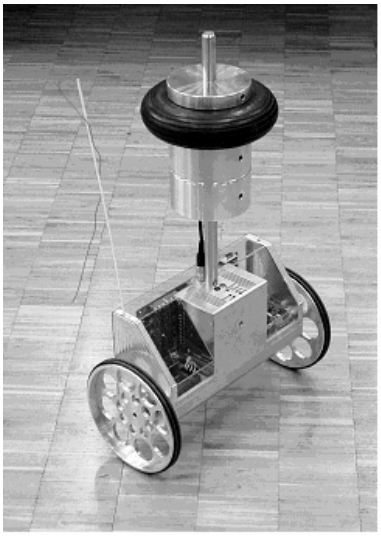


Figure 2.19: Photograph of JOE: A Mobile, Inverted Pendulum (Grasser et al. 2002)

The scale prototype was designed to carry a weight to simulate a human, rather than an actual human. This was done for two reasons, "In order to reduce cost as well as danger for the test pilots" (Grasser et al. 2002). Another advantage of a scaled model is the exactness of states and thus, the measurement of those states is easier and does not vary. There are also disadvantages in using a scaled down prototype, particularly the scaled-down prototype does not provide an exact replica of the final vehicle that was originally envisaged. Although the states are easier to measure, it will not have the variable driver weights to take into consideration and thus may not be able to prove its robustness. It also does not take into account the fact that humans are dynamic and will thus apply dynamic loads when they shuffle their feet on the platform or lean into a corner.

The vehicle was modelled using State Space (SS) theory, specifically to achieve a desired linear speed, as well as a desired turning rate. The modelling also included disturbance rejection for both wheels and rejection for angular

disturbances as external forces from the driver's mass. It is seen that the modelling for this particular project is one of its strengths, as it has enabled the authors to develop a highly accurate model of the system as shown in Figure 2.20.

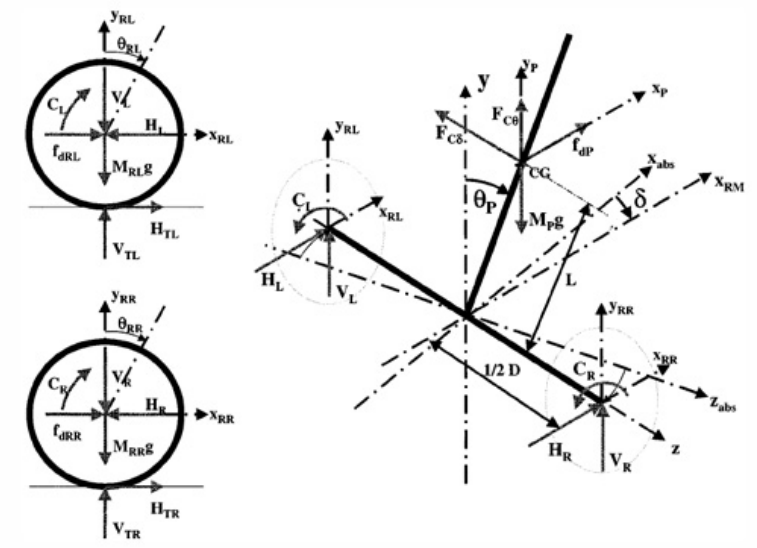


Figure 2.20: Free body diagram of JOE: A Mobile, Inverted Pendulum (Grasser et al. 2002)

Control system development is the next topic discussed in the article. The control of the model was devised with two separate State Space control systems for pitch and yaw respectively. They are decoupled with certain weighting factors to provide torque and therefore voltage outputs to the motors. These are advantageous because they allow the pitch control and yaw control to operate independently when attempting to reach a desired linear speed, or turning rate, or both. The main reason to decouple the systems is to firstly make it easier to prototype for troubleshooting two separate parts of the system rather than one whole system woven together. The second reason stems from the fact that there is only one output system (the motors) and there are two systems (balancing and steering) polling for control of the motor. The decoupling enables a weighting factor to be applied to the two systems independently, in this case, the balancing system is much more critical than the steering system and thus a higher weighting factor is applied.

The control system on the chosen hardware includes a controller which is "composed of a Sharc floating-point DSP, a XILINX field-programmable

gate array (FPGA), four 10-bit D/A converters, as well as fourteen 12-bit A/D converters" (Grasser et al. 2002). It also explains the hardware used for on-board state measurement, the backlash and subsequent noise emitted by the planetary gearbox. The main strength of this implementation is the use of encoders to measure linear displacement and velocity, instead of using accelerometers to integrate for velocity and displacement which would introduce drift. Another strength of the design is the pitch rate limiting built into the controller to simulate safety systems for a real vehicle with a human driver. If the pitch rate exceeds a certain level, the controller kills the control system. The planetary gearbox introduces backlash, which is a weakness because the encoders are mounted on the motors and not the wheels themselves. The encoders do not take into account backlash associated with the gearbox.

Overall, the article describes a process that was well planned and computed to produce a model that is very accurate, fast and efficient.

2.2.3 The Blackwell Model

2.2.3.1 Model I

The article, Building a Balancing Scooter (Blackwell 2005), is an overview of a successful attempt to build a self-balancing scooter similar to the Segway HT that was completed in 2002. The author provides a comparison between his attempt and the Segway HT. Detail about the construction of the vehicle is included, therefore the review will be based on the discussion of the components and systems used in the design and implementation.

The mechanical construction of the vehicle is simple and made from only a few off-the-shelf parts. This is advantageous as it makes the construction inexpensive and easy to both manufacture and assemble, according to Blackwell (2005) "it's mechanically much simpler than any other kind of vehicle". However this can be a downfall as well, making the vehicle not aesthetically pleasing and providing little regard to ergonomics as can be seen in Figure 2.21.

The dashboard is a hobby electronics box attached to the post containing dials for steering and closed loop gain control as well as a 'dead man's' safety switch. The setup that is shown in Figure 2.22 is ungainly and not very practical as these systems could be built into other parts of the vehicle whilst not making construction overcomplicated.

Figure 2.21: Photograph of Blackwell vehicle (Blackwell 2005)

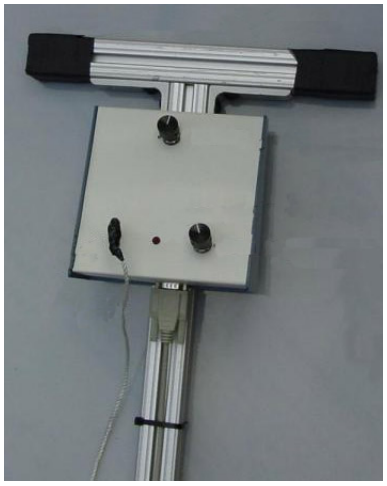


Figure 2.22: Photograph of steering and gain control box on Blackwell vehicle (Blackwell 2005)

The power system consists of Remote Control (RC) car battery packs. They are in parallel with accompanying bridge rectifiers for each pack to prevent current flowing unnecessarily from one battery to another. The Nickel Metal Hydride (Ni-MH) packs provide a power source that can support the high discharge rates demanded by the motors. The weakness that was discovered in this article was that poor battery mounting on the undercarriage

(shown in Figure 2.23) may result in disconnection of the packs and a subsequent loss of power. Another disadvantage in this instance was the relatively small current capacity of the batteries (2500mAh), even though the battery packs were easy to mount and connect, they were only AA batteries inside these packs which deliver a relatively small capacity compared to larger batteries.

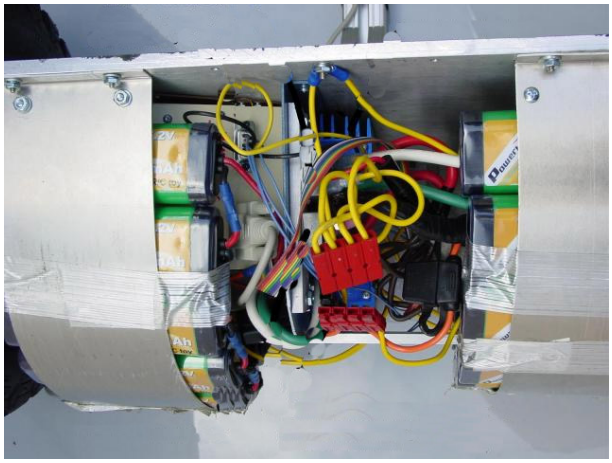


Figure 2.23: Photograph of undercarriage of Blackwell vehicle (Blackwell 2005)

The control system of the vehicle is run from an 8-bit Atmel microcontroller using Proportional Derivative (PD) control with feedback from a piezoelectric rate gyroscope. The weakness of this setup is the need to tune by hand the proportional and derivative gains whilst actually using the vehicle. It is safer to tune the control system in a virtual simulation or with a rapid prototyping tool.

The steering system adds and subtracts a small percentage of power from the motors depending on the current speed of the vehicle. The motors are controlled by Pulse Width Modulation (PWM) signals from the motor controller. PWM allows motors to drive at sub-full speeds whilst still delivering all available torque. It does this by pulsing the rated full voltage to the motors and varying the duty cycle of these pulses to vary the speed. A weighting factor was applied to the steering system to ensure the balancing system maintained priority over the steering system.

Overall, the article is similar to a construction manual but contains information pertinent to the design of EDGAR. The vehicle has been built cheaply

and demonstrates clearly that it is not difficult to achieve something similar to the Segway HT for well under the commercial price.

2.2.3.2 Model II

The article, Balancing Scooter Version Two, is an overview of the second version of a home-made self-balancing scooter as shown in Figure 2.24. It is the second iteration of the vehicle and has many improvements from the first version. It is also designed to have certain abilities and specifications as good as or better than the Segway HT. The review of this model occurred very close to the end of the project due to the information only being released on the author's website in October 2005 and thus the review did not have a great impact on EDGAR but is included as extra information.

Figure 2.24: Photograph of second version of Blackwell vehicle (Blackwell 2005)

The new model was designed to be smoother, lighter, faster and to have a further range than the first model. It was designed to fit through a doorway, have high ground clearance and employ a better steering system than the previous iteration. All of these changes were in direct response to the author

being able to use a Segway HT and see how deficient the first model was in comparison.

The author has moved from 14inch foam filled trailer tyres to 20inch bicycle wheels with fixed hubs and reinforced spoke system. This reduced both weight, rolling resistance and increases the speed at the expense of torsional stiffness, inertia and any vibration isolation properties of the foam filled tyres. The author has also changed the Roboteq dual channel motor controller for a OSMC (Open Source Motor Controller) motor controller driving each wheel independently. The OSMC can supply from 13V to 50V at 160A continuous and 400A peak while the major advantage stems from the processing time which is in the order of one or two milliseconds compared to tens of milliseconds with the previously used Roboteq motor controller.

The author also replaced the gyroscope system used in the first model with a gyroscope/accelerometer assembly that has significatnly less noise and is less susceptable to vibrations. The batteries were also changed on the second model, the battery packs were changed to consist of 60 D cell 6500mAh Ni-MH batteries supplying 36V at 200A peak draw. The author has also included a bluetooth connection and written a program to enable the scooter to be driven remotely whilst balancing. The steering has changed from the touch pads in the first model to a potentiometer system similar to a bicycle with the rotation of the handlebars with reference to the stationary upright post. There has also been some updating of the chassis and undercarriage to make it stronger and to protect the inner components.

2.2.4 The Beckwith Model

In the report for the Human Transport Vehicle (HTV) Project prepared by final year students at Camosun College, Canada, their aim was to "construct a two-wheeled balancing vehicle to explore the electronic fundamentals behind an inverted pendulum as well as solutions to modern day transportation problems" (Beckwith et al. 2004). This is a review of literature in a similar report format to the requirements of the EDGAR project.

The vehicle is a two-wheeled coaxial scooter based around the Segway HT design. It uses data from a gyroscope and an accelerometer directed to a PIC microcontroller in order to balance. Modern fuzzy logic control techniques were utilised on the microcontroller to provide balance for the system. The main strengths from this type of design are the relative simplicity of the control

Figure 2.25: Photograph of Human Transport Vehicle project (Beckwith et al. 2004)

system, the inexpensiveness of the controller, and the use of a gyroscope for angular rate measurements. The main weakness with the design would be the use of an accelerometer uncorrected to give position and velocity data as drift occurs when integrating acceleration data.

The steering system utilised a joystick giving a voltage divider type input to the controller. The controller adds or subtracts a percentage of the speed from each wheel, the amount depends on the current speed. This is similar to both the Blackwell (2005) and Chudleigh et al. (2005) models. The motors, which are the same used in the Blackwell (2005) model, are driven by Pulse Width Modulation (PWM) from a purpose built motor driver circuit fed by the microcontroller. The system uses four 12V Sealed Lead Acid (SLA) batteries to power the device, which are strapped to the undercarriage for easy access, as shown in Figure 2.26. The advantage of this system is the PWM provided by the motor driver circuit; it enables the motors to retain all of the rated full-speed torque when traveling at sub-full speeds. The disadvantages of the setup is the SLA batteries; they are large, heavy and slow to recharge.

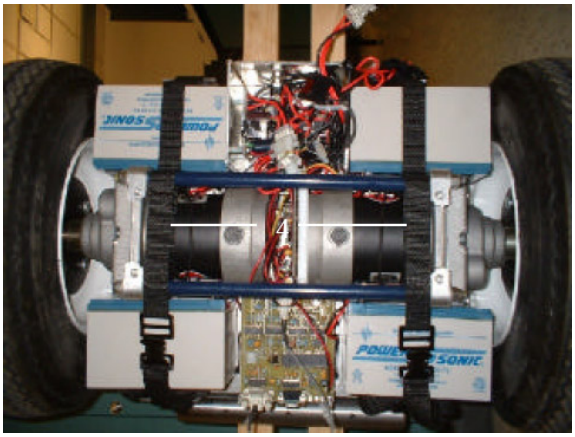


Figure 2.26: Photograph of undercarriage of Human Transport Vehicle project (Beckwith et al. 2004)

2.2.5 The Chudleigh Model

In this article, Project Emanual: The Almost Self-Balancing 2 Wheeled Electric Skateboard (Chudleigh et al. 2005), the authors present an overview of a project that involved retrofitting the skateboard with two wheels and motors, and using practical automatic control to make a skateboard balance on the two coaxial wheels positioned under the centre of the skateboard as shown in Figure 2.27. The authors also provide significant insight into their views on the future of the class of PEVs.

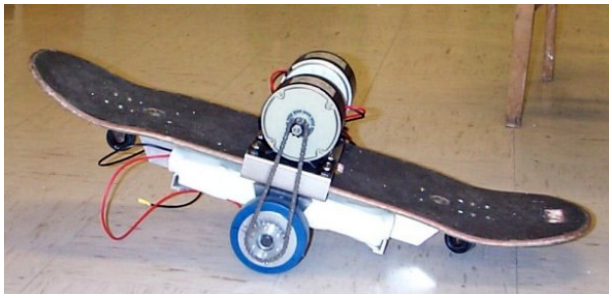


Figure 2.27: Photograph of Project Emanual skateboard (Chudleigh et al. 2005)

The article describes the components used and the problems which arose during the construction of the vehicle. The retrofit consists of two 300W electric motors, rated at 24V, which each drive their corresponding wheel

by chain reduction, as shown in Figure 2.28. The design used two 14.4V Lithium Ion (Li-Ion) battery packs in series. The strengths of this design are the battery packs, not only lightweight, but they can also provide high peak current outputs often greater than average rated current by a factor of ten.

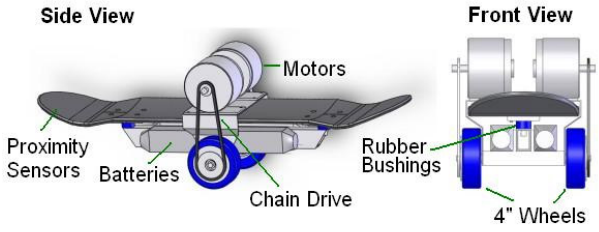


Figure 2.28: CAD pictures of Project Emanual skateboard (Chudleigh et al. 2005)

As opposed to the Segway HT's inertial sensors, the vehicle uses proximity sensors on the undercarriage to determine how far the end of the board is from the ground. The board is mounted on rubber bushings which enable it to roll a few degrees either side of horizontal and underneath the centre of the board on either side are Infra-Red (IR) sensors which enables the direction and amount of roll to be quantified for steering. Both of these sensors are shown in Figure 2.29. The data from these sensors is fed to the PIC (Programmable Integrated Circuit) microcontroller for output to the motors with a percentage added or subtracted for steering, similar to the Blackwell (2005) model. The strengths of such a design are the relative inexpensiveness of the parts required; the microcontroller, proximity sensors and IR sensors cost very little. The major weakness of the design is that both the IR and proximity sensors introduce significant measurement noise into the system and therefore the control system can not be designed to be as robust as desired.

Chudleigh et al. (2005) used a classical Proportional Derivative (PD) control algorithm, which was first implemented in MATLAB's Simulink toolbox and tested in Virtual Reality (VR). The authors possessed no cross-compiler and had to code the entire algorithm manually. Coding was time consuming and proved to be the main weakness of this project.

Overall, the article is a brief summary of the project with a more electrical focus on the content. The design is cost-effective, compact, and innovative. The main weaknesses are the use of sensors that introduce measurement noise

Figure 2.29: Photograph of proximity and Infra-Red sensors on Project Emanual skateboard (Chudleigh et al. 2005)

to the system and the lack of a prototyping tool which made the coding process a long and tediously iterative process.

2.2.6 The Larson Model

In the article, Balancing Robot Project - "Bender" (Larson 2005), the author presents a detailed description of the process used to build an autonomous, self-balancing robot.

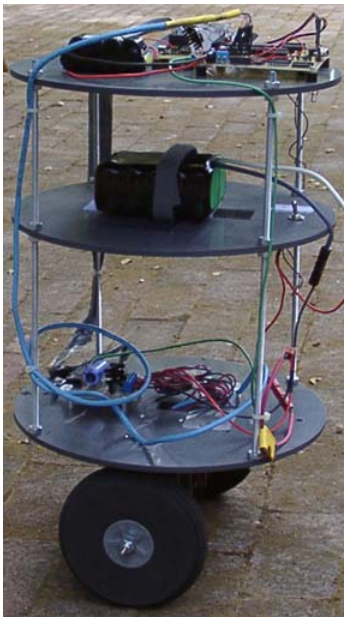


Figure 2.30: Photograph of complete "Bender" the Balancing Robot (Larson 2005)

Initially the robot was built on a large scale. It was a simple mechanical design that incorporated an aluminium frame, which housed the motors and drive systems and PVC plastic tiers to provide height and stability. This made the design not only strong, but lightweight as well. The motors used in this design incorporated fine-grain shaft encoders pre-mounted on them. This provided detailed and accurate position and velocity data.

It was soon discovered that the test model "...quickly proved [to be] too large to be a viable test platform" as "...early testing with the big bot, damaged a couple of walls." (Larson 2005). The size and problems with control were put down to tuning of the PID algorithm. It was then decided that the scale size of the robot should be reduced for a more manageable robot that could be "super-sized" (Larson 2005) at a later date. A CAD program was used to scale down the circuitry. Throughout the tuning process the robot was tethered to a power source and a computer. During this time caster wheels were also installed for prototyping.

The main CPU used in this design was a Microchip PIC18F452 which includes sub-processing units that take the shaft encoder input from the motors. Thus the main CPU does not have to interpret raw data from the encoders, lessening processing load. When the robot was put together the CPU and batteries were all mounted up high to aid in balancing. Balancing was achieved through the use of a "small piezo-electric gyro (Tonkin CG16DO) combined with the output of a two-axis accelerometer (Memsic MXD2125-GW)." (Larson 2005). The use of components such as these is effective because they are relatively cheap and easy to access.

The strengths of this design are that it is easy to assemble and modify the components, and the combination of the accelerometer with the rate gyroscope to provide data.

Chapter 3

Project Goals and Specification Development

In this chapter of the report the goals of the project are outlined including primary and extension goals in Section 3.1. The development of the basic specifications will be explained in Section 3.2, which leads on to Section 3.3 where the component specifications are listed.

3.1 Project Goals

As part of the initial stages of the project a number of goals were chosen to measure the project's success. They were divided into essential and extension goals, of which the essential goals are imperative to the project being considered a success. These goals are:

- To develop an accurate mathematical model and control system for EDGAR
- To reproduce and analyse the model in MATLAB and Simulink
- To develop a Virtual Reality (VR) model of the prototype for the tuning of the control system
- To design and build a physical prototype of EDGAR
- To run the prototype with classical Proportional Derivative (PD) control tethered to a computer using the dSPACE rapid prototyping system

In addition, a number of goals were added to extend the project beyond initial expectations. These were considered to be challenging but achievable if the project proceeded ahead of schedule. The extension goals are:

- To run EDGAR untethered using a classical PD controller
- To determine an accurate State Space (SS) model and controller for EDGAR
- To run EDGAR tethered with SS control using dSPACE
- To run EDGAR untethered with SS control

3.2 Specification Development

The device behaviour was the first specification that was formalised and the subject of most debate. Whilst the vehicle was to be based on the Segway HT, the project was not to merely to produce a duplicate of it. However, it should be noted that the Segway HT was heavily engineered and therefore some of the safety measures were essential to include on EDGAR.

When getting on the device it was desired that the rider would turn a master switch from off to on. This would be in the form of a pole-throw switch or a key type switch. EDGAR would power up but the control system would not engage. The microcontroller would go through any self checks necessary. After the self checks, the microcontroller would operate a display alerting the driver to the readiness of the vehicle. In this state the microcontroller awaits the activation of a foot sensor which signifies a rider's intention to get on the platform and thus the need for the balancing system to start. It was decided to implement the system after reviewing the Blackwell (2005) model which needed the driver to plug in a flimsy 'dead man's' switch whilst placing both feet on the platform. The group felt that this was too difficult to coordinate for a driver who would not have ridden a device like it before.

A major part in the design of EDGAR was to ensure it was safe to operate, since the dynamics involved with the project involve large changes in kinetic and potential energies in relatively short periods of time. In these circumstances it was desired for there to be an emergency stop on EDGAR. This so called 'dead man's' switch was implemented as an open circuit in the power system in the Blackwell (2005) and Beckwith et al. (2004) models. In

other words, there was a connector rigged to the rider's body or clothes, which completed the power circuit from the battery. If this connector was removed, the power circuit would become an open circuit and power to the whole device would cease. In the case of EDGAR, it was decided that the previous methods were clumsy and fallible, therefore the method the Segway HT uses should be integrated but in a different fashion. Sensors were placed beneath the feet to ascertain if the rider had been thrown from the platform, and tolerate the rider to shuffle their feet or take one foot off the platform.

The motion of EDGAR will be similar to that of the Segway HT, as it is the same principle of a mobile inverted pendulum with coaxial wheels. Forward and reverse motion is achieved by leaning forward or backward respectively with a fairly rigid connection to the post through the rider's arms. The vehicle senses a change in the pitch from its set point arising from motion of the rider, and tries to keep the wheels of the vehicle under the centre of gravity of the driver. It is this action and consequent velocity that keeps the user from falling over.

It was desired to run the device both tethered and untethered. In its tethered state, EDGAR uses a hard-wired 24V power supply for the motors, with power provided over a long wiring loom, delivered from above the vehicle. When running untethered 24V power is supplied from a set of rechargeable batteries. A regulated power supply for the microcontroller, IMU and other sensors isolate them from the varying voltage of the batteries.

The steering has been implemented in a similar way to the Chudleigh et al. (2005) model when using the classical PD control system. It may be changed if a future group attempts to move from a classical control model to a modern State Space control model and use a decoupling system as in JOE: a mobile, inverted pendulum (Grasser et al. 2002). For the initial iteration of EDGAR, a percentage of current power is deducted from one motor and the same percentage added to the other motor. One wheel spins faster than the other and thus the vehicle turns. The percentage is inversely proportional to the current speed of the two motors, so that as the speed increases, the device turns in a wider arc to avoid cornering accidents. Another important specification in regard to steering is the need of the device to be able to turn on the spot whilst not moving forward or backward.

The vehicle had some performance guidelines set in terms of capacities and abilities. EDGAR must be able to travel on smooth and rough terrain.

It needed to be able to traverse small inclines of up to ten degrees. It was not however designed to drive down steps or over large rocks. The intended maximum linear speed was ten to fifteen kilometres per hour and the device must run for a minimum of one hour at fifty percent load. It was to carry a human of average weight (70-90kg) and average height (160-190cm), and also possess inherent robustness to counter deviations from these parameters. It should have a height adjustable handle so that different drivers can use the device. To make EDGAR as useful as possible, it should be able to travel indoors as well as outdoors and thus must be able to fit through a standard sized door frame (W820xH2040mm). The device should possess enough ground clearance to avoid general bumps, small rocks, and speed humps. The device must have adequate coverage of moving parts so that the driver could not become entangled. This required fenders for the front and rear of the platform as well as wheel covers.

3.3 Basic Component Specifications

The following list of component specifications was derived from Section 3.2 and details requirements of major components. These items are covered in detail along with other components in Chapter 5.

Motors

- Bidirectional
- Rotate at low speeds
- Low backlash
- Mounting points present
- Both motors need similar operating specifications
- Provide sufficient torque at 50 percent of maximum speed

Wheels

- Single side-supported axle
- Be able to support half person's bodyweight

- Preferably pneumatic tires
- Non-marking tires
- Ability to withstand rough/bumpy surfaces
- High moment of inertia

Motor controller

- Either be one controller per motor or one controller that controls both motors
- Be able to drive motors bi-directionally
- PWM preferred over current or voltage limiting drive modes
- Accept inputs from microcontroller and dSPACE

IMU

- Easily interfaced to microcontroller and dSPACE
- Detect angle and angular rate on at least one axes
- Low power consumption desirable

Motor Controller

- Desirable to accept compiled code from Simulink
- Have multiple I/Os
- Have multiple ADC/DAC
- Able to accept encoder and IMU inputs
- Low power consumption desirable

Power Source

- Rechargeable
- Provide adequate current to motors
- Provide power to components as needed

- Enough capacity to run EDGAR for one hour at 50 percent of maximum speed

Chassis

- Support 90kg person
- Height adjustable handlebars
- Ergonomic and aesthetically pleasing
- Secure connection between structural plate and upright post
- Fit through door frame
- Non-slip surface on platform for feet

Chapter 4

Control System Design

This chapter discusses the design of the control system implemented on EDGAR. In Section 4.1, the methodology of the control system design is examined along with a short explanation of the choice of the controller. Section 4.2 includes the derivation of the equations of motion for EDGAR from the free bodies initially discussed. The iterations of the control system design is discussed in Section 4.3 and the final control system design is explained in Section 4.4.

4.1 Introduction

As discussed in Chapter 1, a self-balancing scooter is inherently unstable, and as such needs a control system to be able to keep it upright. This control system must take measurements of the scooter's current state and determine what signals to apply to the motors in order to achieve a desired response.

Generally a control system is designed using the following stages:

- First a mathematical model of the system is derived; this needs to be as accurate as possible whilst trading off against complexity
- The system is then identified by analysing the model and its response to various inputs
- From the results of the previous step, a suitable control system can then be designed

- The control system is then tuned to achieve a desired response using the mathematical model; often graphical tools such as Virtual Reality are used to help with visualisation
- Finally, when the real device is available, the tuned control system is then implemented into hardware
- Ideally the process would stop here, but realistically the controller would then need to be retuned with focus given on making sure that the control system is robust to changes in the device, such as extra weight and different wheels.

This was the basic process taken in order to design the control system for EDGAR, however before going into detail, a few controllers will be discussed with relevance to the project. The most common type of controller used today is the classical PID (Proportional Integral Derivative) controller. This type of controller is used most frequently in control applications due to its simplistic design and ability to be tuned easily to control a variety of systems. A PID controller is generally used to regulate an output to a setpoint value, or track a changing input reference signal. The control signal is generated by applying three gains to the input error and summing the results. There is a proportional part which is simply proportional to the magnitude of the error, an integral part that takes into account how long there has been an error and a derivative part which changes according on how quickly the error is changing.

$$\begin{aligned} K(s) &= K_p + K_d s + \frac{K_i}{s} \\ &= \frac{K_d s^2 + K_p s + K_i}{s} \\ &= \frac{K_p T_d s^2 + K_p s + T_i K_p}{s} \\ &= K_p \left(\frac{T_d s^2 + s + T_i}{s} \right) \end{aligned} \tag{4.1}$$

The transfer function of a PID controller is shown in Equation 4.1, with K_p the proportional gain, K_i the integral gain and K_d the derivative gain. This can also be written in terms of a single gain, K and time constants, T_i and T_d, see Equation 4.1. These time constants are indicative of how quickly the parts of the control system will settle to a value. The transfer function of

the PID controller has one pole at zero and 2 conjugate-pair zeros that can be placed anywhere depending on the chosen parameters. Figure 4.1 shows a pole-zero map for an arbitrary PID controller with changing gains. The placement of the zeros changes the overall system response and are used to help reduce the effects of any undesirable poles in the system that is being controlled.

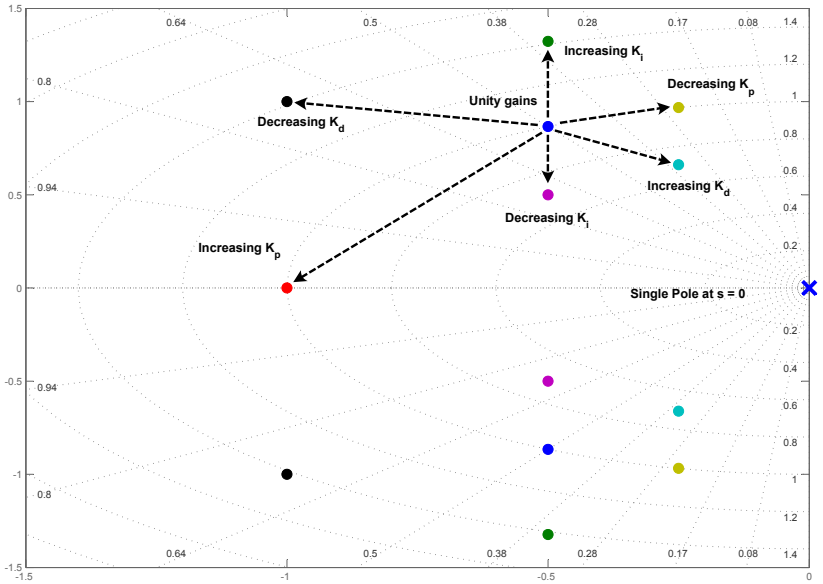


Figure 4.1: MATLAB pole-zero map plot for differing gains

Although this controller works well in simulation and numerical analysis, in a discrete system it could not be implemented correctly. The main problems implementing this type of controller in a discrete format are:

- A finite sampling frequency and resolution means that signals are made up of discrete steps rather than the ideal continuously smooth signal. As a result, the derivative part of the controller will either be zero or infinite, which cannot be practically implemented.
- Step changes in setpoint signal will also result in unwanted derivative spikes.
- A limited range in the control signal due to limits on the actuator inputs. This means that the controller cannot simply produce as large a signal as it wants and can result in a situation know as ‘wind-up’ as the integral

part of the controller keeps increasing while the actual control signal saturates at a limit.

The high frequency derivative spikes are removed using a lowpass filter in the derivative part of the controller. This adds a pole to the controller that slows down its response to the high frequency signals that make up the step changes. The selection of this filter is usually chosen through testing to find a suitable value that removes unwanted derivative spikes, without slowing down the controller too much.

The derivative spikes that result from input step changes can be avoided by using a setpoint weighting that still applies the derivative part to the feedback signal but not to the setpoint input.

Actuator windup, which only occurs with the integral part in the controller, can be avoided by limiting the control signal, or by 'leaking' off the integral part when the actuator reaches saturation.

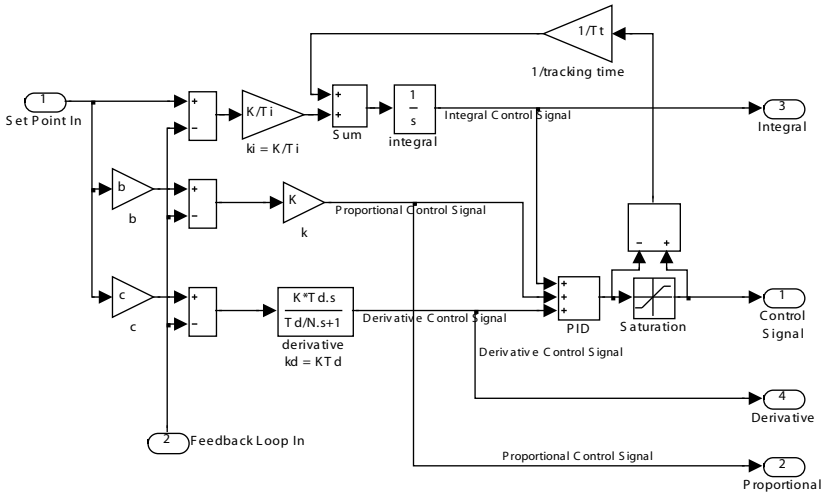


Figure 4.2: Simulink model of generalised PID block (Cazzolato 2005)

In Figure 4.2, a Simulink block diagram of a generalised PID controller is shown. It includes the features mentioned previously; a lowpass filter on derivative part, setpoint weighting and actuator wind up compensation. Cazzolato (2005) specifies the following table and outlines the effect that each part of the control system has on the overall system response.

Closed Loop Response	**Rise Time**	**Overshoot**	**Settling Time**	**Steady State Error**
K_p	Decrease	Increase	Small Change	Decrease
$K_i = \frac{K_p}{T_i}$	Decrease	Increase	Increase	Eliminate
$K_d = K_p * T_d$	Small Change	Decrease	Decrease	Small Change

Recently, a newer method of control has become popular, the State Space (SS) method. The basis of this method is that every system can be represented, not just by a single transfer function from one parameter to another, but by a group of state variables, that may be independent but more often than not depend on each other. A SS model is based around the state equation and the output equation. The state equation, as shown in Equation 4.2, describes how the states respond to an input and each other, resulting in the derivatives of the states that are used to predict future states. The output equation, as shown in Equation 4.3, chooses a specific state, or multiple states, as the output variable of the model. This allows one or more states to be fed back into the control system for a more accurate and robust response.

$$\dot{x} = Ax + Bu \tag{4.2}$$

$$y = Cx + Du \tag{4.3}$$

In an ideal real world situation, a system would have multiple sensors measuring all possible states with as best accuracy as possible. These become the inputs to the controller which uses the SS model to control the desired states. If accurate measurements of a state cannot be made, a system known as an observer can be created which uses the known measured states to try and predict the unknown value. This predicted state can then be used as if it was a measured value. The main drawback of SS control is that an accurate model is required, since this is used within the controller and also to predict unknown states. However, due to the feedback of more than one state, it introduces the ability to give more accurate control of the desired variables.

As mentioned previously in Chapter 2, EDGAR was to be designed initially with a classical PD controller. Using classical control, it was decided that the

tilt angle of the post would be used as the control variable. In order to keep the scooter upright, this variable was required to be forced back to vertical. A turning adjustment signal with no feedback was used to offset the main drive signal for each wheel. If a SS control system was to be used, then measurements of both wheel speeds, the tilt angle, velocity and acceleration would likely have been also used as inputs. Outputs would have been separate drive signals to each motor.

4.2 Kinematic Analysis

Returning to the design methodology, the first step required a mathematical model of EDGAR. By looking at previous attempts of self-balancing scooters and mobile inverted pendulums, a suitable free body diagram was found. The model chosen was based on the Grasser model (Grasser et al. 2002), but with some simplification of the equations.

Three bodies were used, the two rotating masses of the wheels and the chassis/person combined body, which was represented using a single point mass a certain distance from the axle. The latter was possible as it was anticipated that the person would be significantly heavier than the base itself.

Diagrams of the free bodies and resulting forces are shown in Figures 4.3 and 4.4. The forces acting on the wheels include forces due to contact with the ground, horizontal disturbances, weight, reactions with the chassis and torques applied by the motors. The forces on the chassis included a force that a person would apply to the post when tilting forward or backward, weight and reactions from the base. By balancing these forces and torques, the following equations of motions were derived.

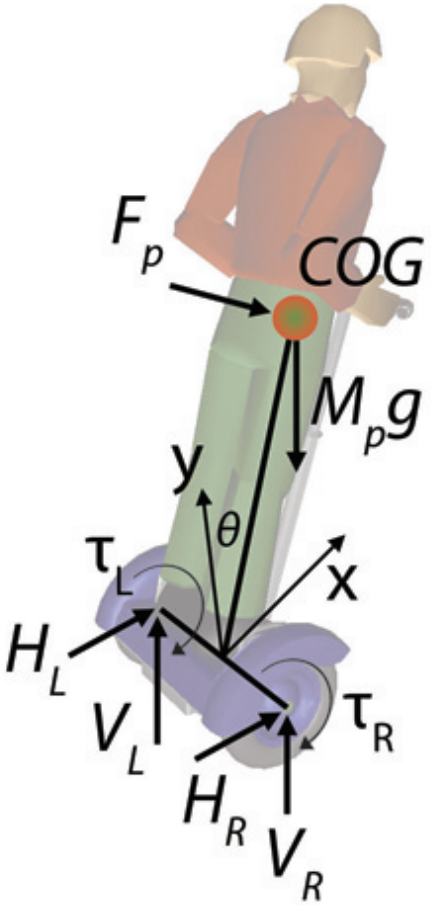


Figure 4.3: Free body diagram of entire system

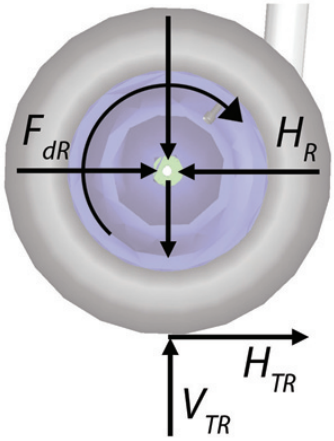


Figure 4.4: Free body diagram of wheel (applies to both left and right wheels)

Sum of Moments on Chassis:

$$J_{P\theta}\ddot{\theta}_P = F_P\frac{L}{2} + M_P g\sin(\theta_P)\frac{L}{2} + (V_R + V_L)\sin(\theta_P)\frac{L}{2} - (H_R + H_L)\cos(\theta_P)\frac{L}{2} - (\tau_R + \tau_L) \tag{4.4}$$

Sum of Moments on Wheels:

$$J_{R\phi}\ddot{\phi}_R = \tau_R - H_{TR} \qquad J_{L\phi}\ddot{\phi}_L = \tau_L - H_{TL}R \tag{4.5}$$

Sum of Forces on Wheels:

$$\begin{aligned} M_R\ddot{x}_R &= f_{dR} + H_{TR} - H_R & M_L\ddot{x}_L &= f_{dL} + H_{TL} - H_L \\ M_R\ddot{y}_R &= V_{TR} - M_R g - V_R & M_L\ddot{y}_L &= V_{TL} - M_L g - V_L \end{aligned} \tag{4.6}$$

Which are rearranged to:

$$H_R = f_{dR} + H_{TR} - M_R\ddot{x}_R \qquad H_L = f_{dL} + H_{TL} - M_L\ddot{x}_L \tag{4.7}$$

$$V_R = V_{TR} - M_R g - M_R\ddot{y}_R \qquad V_L = V_{TL} - M_L g - M_L\ddot{y}_L \tag{4.8}$$

Substituting into Equation 4.4:

$$\begin{aligned} J_{P\theta}\ddot{\theta}_P \;=\; & F_P\frac{L}{2} + M_P g\sin(\theta_P)\frac{L}{2} \\ & +(V_{TR} - M_R g - M_R\ddot{y}_R + V_{TL} - M_L g - M_L\ddot{y}_L)\sin(\theta_P)\frac{L}{2} \\ & -(f_{dR} + H_{TR} - M_R\ddot{x}_R + f_{dL} + H_{TL} - M_L\ddot{x}_L)\cos(\theta_P)\frac{L}{2} \\ & -(\tau_R + \tau_L) \end{aligned} \tag{4.9}$$

Assume that the reaction with ground is due to total weight:

$$V_{TR} = V_{TL} = \frac{1}{2}(M_R + M_L + M_P)g \tag{4.10}$$

Assume friction with the ground is significant enough that there is no wheel slip:

$$H_{TR} = -\frac{\tau_R}{R} \quad \& \quad H_{TL} = -\frac{\tau_L}{R} \tag{4.11}$$

Assume no vertical motion of wheels on flat ground:

$$\ddot{y}_R = \ddot{y}_L = 0 \tag{4.12}$$

Horizontal motion provided by rotation of the wheels with H_{TL} and H_{TR} as defined previously and therefore:

$$\ddot{x}_R = R\ddot{\phi}_R = \frac{R}{J_{R\phi}}(\tau_R - H_{TR}R) \tag{4.13}$$

$$\ddot{x}_L = R\ddot{\phi}_L = \frac{R}{J_{L\phi}}(\tau_L - H_{TL}R) \tag{4.14}$$

With moments of inertia defined as:

$$J_{P\theta} = M_P(\frac{L}{2})^2 \tag{4.15}$$

$$J_{R\phi} = \frac{1}{2}M_R R^2 \tag{4.16}$$

$$J_{L\phi} = \frac{1}{2}M_L R^2 \tag{4.17}$$

The new moment equation becomes:

$$\begin{aligned} M_P(\frac{L}{2})^2\ddot{\theta}_P \quad &= F_P\frac{L}{2} + M_P g\sin(\theta_P) \\ &-[f_{dR} - \frac{\tau_R}{R} - M_R\frac{R}{J_{R\phi}}(\tau_R + \frac{\tau_R}{R}R)]\cos(\theta_P)\frac{L}{2} \\ &+[f_{dL} - \frac{\tau_L}{R} - M_L\frac{R}{J_{L\phi}}(\tau_L + \frac{\tau_L}{R}R)]\cos(\theta_P)\frac{L}{2} - (\tau_R + \tau_L) \end{aligned} \tag{4.18}$$

Which simplifies to:

$$\begin{aligned} \ddot{\theta}_P \quad = \quad & \frac{F_P + \frac{2M_P g\sin(\theta_P)}{L}}{M_P\frac{L}{2}} \\ & -\frac{[f_{dR} + f_{dL} - (\frac{1}{R} + M_R\frac{2R}{J_{R\phi}})\tau_R - (\frac{1}{R} + M_L\frac{2R}{J_{L\phi}})\tau_L]\cos(\theta_P) + \frac{2(\tau_R+\tau_L)}{L}}{M_P\frac{L}{2}} \end{aligned} \tag{4.19}$$

Assuming wheels have same mass and thus same moment of inertia:

$$M_R = M_L = M_W \tag{4.20}$$

$$J_{R\phi} = J_{R\phi} = \frac{1}{2}M_W R^2 \tag{4.21}$$

$$\ddot{\theta}_P = \frac{F_P + \frac{2M_P g \sin(\theta_P)}{L} - [f_{dR} + f_{dL} - \frac{5}{R}(\tau_R + \tau_L)]\cos(\theta_P) - \frac{2(\tau_R+\tau_L)}{L}}{M_P \frac{L}{2}} \tag{4.22}$$

This differential equation then gives the angular acceleration of the post, $\ddot{\theta}_P$, as a function of force provided by the person, F_P, and torque provided by the motors, τ_R and τ_L.

4.3 Controller Development

The differential equation derived from the mathematical model of EDGAR is non-linear and as such cannot be simply converted to a transfer function using the Laplace transform. As it was desired to try and control the non-linear system, this equation was implemented into a MATLAB and Simulink model. The simplest way to do this was to to create the function and use the integrated angular acceleration for the angle measurement. The angle was fed through the controller block to create the individual torque signals, a force input was used and the model parameters (wheel size, weight and location of centre of mass) were specified as required.

Even though a controller for a non-linear system was to be designed, it was still desirable to find the poles of the system. This was done by linearising the equations about the upright position, or $\theta_P = 0$. Disturbances at the base were assumed to be zero and the torque values combined into one, $\tau_R + \tau_L = T$. This allowed a Laplace transform to be performed resulting in a transfer function from T and F_P to θ_P, as shown in Equation 4.24.

The trigonometric identities once linearised about the upright position ($\theta_P = 0$):

$$\sin(\theta_P) = \theta_P, \quad \cos(\theta_P) = 1 \tag{4.23}$$

Assuming that disturbances are zero to begin with and combining wheel torques:

$$\ddot{\theta}_P - \frac{4RM_P g\theta_P}{M_P RL^2} = \frac{2LRF_P}{M_P RL^2} + \frac{(10L - 4R)(T)}{M_P RL^2} \tag{4.24}$$

Apply Laplace Transform:

$$\theta_P(s)s^2 - \frac{4RM_Pg\theta_P(s)}{M_PRL^2} = \frac{2LRF_P(s)}{M_PRL^2} + \frac{(10L-4R)T(s)}{M_PRL^2} \tag{4.25}$$

$$\theta_P(s)(s^2 - \frac{4RM_Pg}{M_PRL^2}) = \frac{2LRF_P(s)}{M_PRL^2} + \frac{(10L-4R)T(s)}{M_PRL^2} \tag{4.26}$$

$$\theta_P(s) = \frac{2LR}{(M_PRL^2s^2 - 4RM_Pg)}F_P(s) + \frac{(10L-4R)}{(M_PRL^2s^2 - 4RM_Pg)}T(s) \tag{4.27}$$

$$= \frac{\frac{2R}{M_PL}}{(s+\frac{2\sqrt{g}}{L})(s-\frac{2\sqrt{g}}{L})}F_P(s) + \frac{\frac{10L-4R}{M_PRL^2}}{(s+\frac{2\sqrt{g}}{L})(s-\frac{2\sqrt{g}}{L})}T(s) \tag{4.28}$$

The two transfer functions, $\frac{\theta_P(s)}{F_P(s)}$ and $\frac{\theta_P(s)}{T(s)}$, are very similar and each have two purely real poles. Their position depends only on the value of L, which will change with the height and weight of the rider. The smaller L is, and thus the lower the centre of gravity, the further the poles move from the imaginary axis, which results in a harder to control system, requiring more effort to regulate.

As can be seen, the system appears to have two inputs and one output. However, the open loop transfer function is simply the transfer function from F_p to θ, since the control signal to be fed to the torque system is generated by the feedback controller and thus is not included in the open loop configuration. This reduces the system to a simple single input, single output system when a controller, C, is included and the loop is closed. The controller acts as a regulator, attempting to force the output signal, θ, back to zero while the input comes from the disturbance, F_p. The closed loop flow diagram and transfer function are shown in Figure 4.5.

It was then possible to start designing a suitable controller for the system. The process described here is an example of pole placement, where the poles of the controller are placed specifically by specifying a desired closed loop system natural frequency, damping and in a 3rd order system, where to place the lowpass filter pole.

A Proportional Derivative (PD) controller was decided on since the system response would benefit from a small steady state error in the tilt angle. This steady state error provides the forward movement required by the scooter. Proportional gain is used to provide the main power to the motors to drive the scooter. The derivative part predominately acts when changes in angle

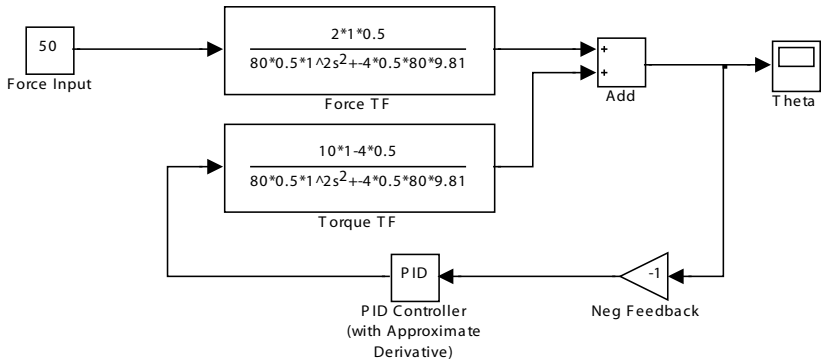


Figure 4.5: Simulink model of linearised model of control system

are detected but also damps out any oscillations.

First a theoretical PD controller was designed. This has the transfer functions shown below:

$$C = K_p + K_p T_d s \tag{4.29}$$

It is not practically implementable due to the stepwise nature of discrete signals, as mentioned previously, but is useful for initial analysis. The controller was included into the closed loop transfer function, and rearranged to the standard 2nd order form. The denominator was then compared to a desired 2nd order characteristic equation which specified the desired natural frequency, ω_n, and damping coefficient, ζ, of the closed loop system. The proportional gain, K_p, and derivative time constant, T_d, was then specified using these parameters.

Closed Loop Transfer Function with theoretical PD control:

$$CL = \frac{\frac{2R}{M_P L}}{s^2 + \frac{10L-4R}{M_P R L^2} K_d s + \frac{10L-4R}{M_P R L^2} K_p - \frac{4g}{L^2}} \tag{4.30}$$

Compare to denominator to the desired 2nd order characteristic:

$$s^2 + \frac{10L-4R}{M_P R L^2} K_p T_d s + \frac{10L-4R}{M_P R L^2} K_p - \frac{4g}{L^2} = s^2 + 2\zeta\omega_n s + \omega_n^2 \tag{4.31}$$

$$\Rightarrow \quad K_p = \frac{\omega_n^2 + \frac{4g}{L^2}}{\frac{10L-4R}{M_P R L^2}} \quad , \quad T_d = \frac{2\zeta\omega_n}{\frac{10L-4R}{M_P R L^2} K_p} \tag{4.32}$$

The practical PD controller must include an approximate derivative, which is implemented using a lowpass filter with time constant $\frac{T_d}{N}$. The transfer function for the practical controller is shown below.

$$C = K_p + \frac{K_p T_d s}{1 + s\frac{T_d}{N}} \tag{4.33}$$

Again, this controller was included in the system transfer function, but this time resulted in a 3rd order system. The desired 3rd order characteristic is shown below, which also includes a term for the single real pole that defines the lowpass filter properties. After comparing the characteristic with the denominator of the closed loop transfer function, K_p, N and T_d can all be specified.

Closed Loop Transfer function with practical PD control:

$$CL = \frac{\frac{2R}{M_P L}(1 + s\frac{T_d}{N})}{s^3\frac{T_d}{N} + s^2 + (\frac{T_d}{N}\frac{4g}{L^2} + \frac{T_d}{N}\frac{10L-4R}{M_P R L^2}K_p + K_p T_d)s + \frac{10L-4R}{M_P R L^2}K_p - \frac{4g}{L^2}} \tag{4.34}$$

$$CL = \frac{\frac{2R}{M_P L}(\frac{N}{T_d} + s)}{s^3 + \frac{N}{T_d}s^2 + (\frac{4g}{L^2} + \frac{10L-4R}{M_P R L^2}K_p + NK_p)s + \frac{N}{T_d}\frac{10L-4R}{M_P R L^2}K_p - \frac{N}{T_d}\frac{4g}{L^2}} \tag{4.35}$$

Compare the denominator to the desired 3nd order characteristic:

$$s^3 + \frac{N}{T_d}s^2 + (\frac{4g}{L^2} + \frac{10L-4R}{M_P R L^2}K_p + NK_p)s + \frac{N}{T_d}\frac{10L-4R}{M_P R L^2}K_p - \frac{N}{T_d}\frac{4g}{L^2} \tag{4.36}$$

$$= s^3 + (\alpha\omega_n + 2\zeta\omega_n)s^2 + (\omega_n^2 + 2\zeta\omega_n^2)s + \alpha\omega_n^3$$

$$\Rightarrow \qquad K_p = \frac{\frac{\alpha\omega_n^2}{\alpha+2\zeta} + \frac{4g}{L^2}}{\frac{10L-4R}{M_P R L^2}} \tag{4.37}$$

$$\Rightarrow \quad N = \frac{\omega_n^2 + 2\zeta\omega_n^2 - \frac{4g}{L^2} - \frac{10L-4R}{M_P R L^2}K_p}{K_p} \tag{4.38}$$

$$\Rightarrow \qquad T_d = \frac{N}{\alpha\omega_n + 2\zeta\omega_n} \tag{4.39}$$

The pole-placement method is suitable for designing controllers to achieve a specific system response. As a desired response was not known initially, the

first controller was designed using trial and error.

This was done by using the implementation of the non-linear mathematical model of EDGAR in Simulink. A Virtual Reality (VR) model was also created to aid with visualising the response of the model. This was imported into the MATLAB VR simulator from the design modelled in SolidWorks. Because the VR simulator required a specific format to be able to manipulate the model pieces correctly, the output of the model was fed through another system to convert it to the correct values. Part of this was to convert the torque applied to each wheel to an acceleration and then combine to two accelerations to give a forward acceleration and turning rate for EDGAR. The differential turning was implemented by added and subtracting a small value from each wheel control signal. There was no feedback to get a specific turning rate, but the amount of offset was determined during testing and observation of the response.

It was also required to model the forces that the person was going to apply during leaning and the resistance effects on the movement of the base. A joystick was used as the input for the person, simulating a lean on the post. The input from the joystick was then multiplied to give a value for a force that was thought to be able to be exerted without too much effort on the post. Based on an average rider weight of approximately 80kg, and estimating that one would not push with more than 5-10 percent of their bodyweight, a maximum input force of 50N was used. The resistance effects were applied by feeding back the calculated ground speed of the model and applying a proportion of friction to act against the direction of motion. This was difficult to estimate but through observing the response of the model once a controller was implemented, a suitable value was chosen. This is probably the most inaccurate part of the model due to the estimated applied force.

The system response was then analysed with varying inputs, both constant and also varying quickly to simulate rocking and sudden changes. The controller gains were found by first increasing the proportional gain until the model was able to stay upright to simple impulse inputs. This was acceptable, however when an oscillating input was used, the system did not settle and gave an unstable response. The increase of derivative gain resulted in not only damping out the oscillations, but also a fast response to quick changes.

Because of the non-linear nature of the model, and the inability to realistically model the friction and resistance effects that different surfaces would

provide, it was difficult to achieve a perfectly realistic response. When a response was produced with minimal oscillations and a stable steady state value, it was decided that the control system could then be implemented on EDGAR.

4.4 Final Controller Design

Once the controller had been designed from within MATLAB and Simulink, it was required to be implemented on the real system. This was initially implemented through the dSPACE platform, which is described in more detail in Chapter 6. The controller was tuned using trial and error. It was found that the proportional gain provided the power to keep the base underneath the rider. With higher gains the response become oscillatory and was sensitive to very small movements. The addition of derivative gain resulted in the damping of some of the oscillatory behaviour and a smoother response. This was the same behaviour as observed within Simulink, confirming that the model was indeed accurate. It was found that a proportional gain of 21 and derivative gain of 0.5, implemented with a lowpass filter time constant of 1/100, produced the most stable response. It was sensitive enough that it did not require the rider to lean far to move forwards, but provided a smooth enough change that felt controllable.

After successful tethered tuning, the focus shifted to implementing the control system on-board EDGAR. Further discussion on generating the code required to integrate all the sensors and motor control can be found in Chapter 6. The same controller that was used in the dSPACE implementation was used on the microcontroller, but this control system did not behave as was expected on the first attempt. This was found to be due to the mistaken implementation of discrete transfer functions which need to be written in alternative forms to the classical continuous functions. Moving back to continuous transfer functions resulted in the behaviour that was expected. Using trial and error again, the gains were modified to $K_p = 15$ and $K_d = 0.14$ with the same lowpass filter time constant of 1/100. The difference in the gains from the initial implementation may have been due to changes in the weight distribution of the system, which has included the completed handlebars, on-board batteries and the faring and wheel covers. It could also have been due to the much slower sample rate of the microcontroller, which was sampling the sensors and calculating values at 200Hz, as compared to the core speed of

250MHz on the dSPACE platform.

The similarity between the tuning parameters when EDGAR was tethered to dSPACE and later tuned untethered reinforced the belief that the final control system had been implemented correctly.

Chapter 5

Component Selection

The following chapter addresses the issues of designing and selecting hardware and structural related components, detailing hardware considerations including sensors and the drive system. Refer to Section 7.1 for factors affecting the physical size and appearance of EDGAR.

The selection of EDGAR's components was completed gradually to meet the project demands including budget constraints, manufacturing complexity, elegance of solution, and specification satisfaction. Refer to Chapter 3 for the specifications used to design EDGAR.

Since EDGAR has been designed and constructed from the initial idea, many systems had to be designed and integrated concurrently. The lack of existing mechanical or electrical systems made selecting multiple components that depended on each other to satisfy specifications challenging. Whilst calculations were performed to check the expected performance of systems, it was not always possible to perform these calculations before component selection. Estimates of the capability of a system were made as required.

5.1 Motors & Gearboxes

An integrated DC electric geared motor simplified the mechanical design considerations, providing a lighter and more compact assembly than a separate motor and gearbox arrangement. Two 250W geared electric motors from Unite Motors were chosen to drive EDGAR, shown in Figure 5.1. The specifications as provided by the vendor, Oatley Electronics, for the SC250G motors are listed below:

- 24VDC operation
- Power: 250W
- Rated speed: 320 RPM
- Nominal output torque: 7.46 Nm
- Measures: 110mm Diameter x 115mm Length (+ output shaft)
- Weight: 2.2kg

The geared motors provide enough torque at the wheels to keep the platform under the user, as the torque is similar to the torque rating of the motors used in the Blackwell model, as discussed in Section 2.2.3.

For safety reasons the motors are limited to about fifty percent of their total capacity (refer to Section 3). Mounting the motors was possible by attaching angle aluminium to the pre-tapped holes on the front surface of the motor body, and fixing the angle aluminium to the main platform of EDGAR. Limiting the capacity of the motors provided a maximum speed of 150RPM. This allowed the wheel selection to follow, which defined the maximum speed of EDGAR.

5.2 Wheels

As discussed in Section 3, the maximum speed of EDGAR was limited to around 15km/h. EDGAR has been designed to carry a user weighing 90kg, and some suspension in the tyres would help smooth the ride for such a large mass. Aesthetic considerations required the diameter of the wheels to be at least as large as the front-back dimension of the platform, and thus the diameter of the wheels needed to be at least 400mm. A high inertia was a desirable property of the wheels as it allows torque from the motors at low speeds to act on the angle of the platform as well as the speed of the wheels. This means that at low speeds or when stationary, small changes in angle from upright results in the motors correcting this angle directly, rather than having to spin the wheels to move them underneath the platform.

A number of wheels were considered in the selection process including power wheelchair, golf caddy, wheelbarrow, and bicycle wheels. Golf-caddy

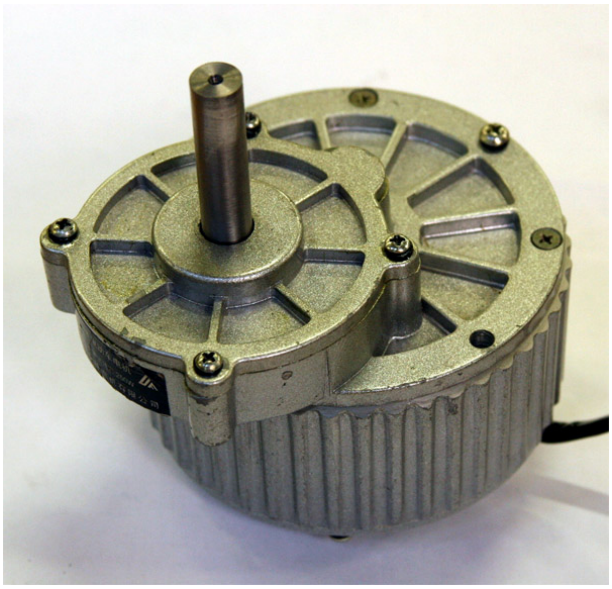


Figure 5.1: Photograph of the SC250G geared motor

wheels were eliminated quickly as their tyre surface was not suited to bitumen or abrasive surfaces and possessed poor suspension properties. Bicycle wheels have relatively little inertia and their tread was considered too narrow for aesthetic looks. Power wheelchair wheels had a number of advantages over golf caddy and bicycle wheels including high inertia, wide treads and appealing looks, though the cost of such wheels was too high. Wheelbarrow wheels were found to have reasonable inertia, wide treads, interchangeable tyres, good suspension characteristics, and the wheel hubs were readily modifiable.

Suitable wheelbarrow wheels were sourced from Super Cheap Auto, an automobile supply store in Adelaide. They were of low cost and reasonable construction. The hubs were rebuilt to convert the axles from a double-side supported rolling axle, to a single-side supported fixed axle arrangement with a keyed sleeve. Refer to Figure 5.2 for a photograph of the wheels.

5.3 Motor Controller

Motor controllers were required to turn the control signal from the microcontroller into an appropriate varying power level to drive the motors. Pulse Width Modulation (PWM) is one method of communicating between a microcontroller and a motor controller. By sending a train of pulses at regular

Figure 5.2: Photograph of the 400mm wheel

intervals and varying the width of the pulses, the motor controller is able to interpret this pulse width as a requested motor duty level.

Two PWM motor controllers were supplied by The University of Adelaide Instrumentation Workshop to drive the motors. These controllers used reasonably low capacity components to provide the output current, and were destroyed during initial testing of the motors when current draw up to 10A was observed. Peak current requirements for the motors were found to be upon rapid reversal of direction and the workshop built controllers were incapable of supplying large currents.

A dual channel Roboteq motor controller was then selected and purchased to drive the motors. The AX2850 motor controller has two independent output channels each supplying 24V at up to 140A. This is considerably more capacity than EDGAR is expected to require, though the extra capacity of the controllers allows for expansion of EDGAR's capabilities in the future. A photograph of the motor controller is shown in Figure 5.3. The specifications of the AX2850 are attached as Appendix A.

Some main features of the AX2850 include:

- Independent output channels at 24V and up to 140A
- Input options including RS232 Serial, PWM, and Analogue control
- Built-in temperature, voltage, and current monitoring
- Dead-man and emergency stop switch inputs

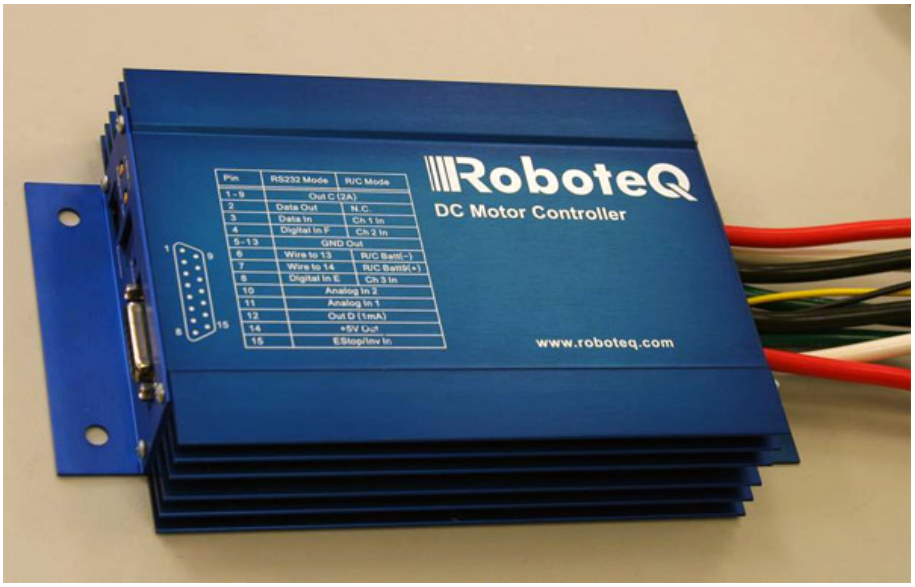


Figure 5.3: Photograph of the AX2850 motor controller

5.4 Microcontroller

Initially, EDGAR was tethered to a host computer running Simulink and dSPACE Control Desk to provide the balancing control system described in Section 4. After the testing period, the control system was implemented on-board EDGAR. A development board was chosen to fill this requirement as they provide ample processing and memory storage capability. The Wytec MiniDRAGON+ development board was selected and provided by The University of Adelaide.

Some features of the Wytec MiniDRAGON+ board include:

- 16 channel 10bit analogue to digital (A/D) converter
- 256KB flash memory
- Many digital input output (I/O) pins (up to 89)
- Programmable in C (allowing cross-compilation from Simulink)

The numerous A/D converters allow for multiple inputs/outputs as required. Two serial ports are supported by the MiniDRAGON+. The first serial port is used to connect the controller to the computer via a tether during testing, whilst the second serial port allows the Inertial Measurement Unit (IMU) to be connected to the controller. This arrangement changed during testing as described in Section 8.1.5.

A real-time target for Simulink was made available by Dr Frank Wornle. It allowed cross-compilation of Simulink models directly onto this board. Rapid migration of models from Simulink onto the development board saved time that would otherwise have been spent translating the Simulink models into code that the MiniDRAGON+ understands.

A picture of a MiniDRAGON+ development board is shown in Figure 5.4.

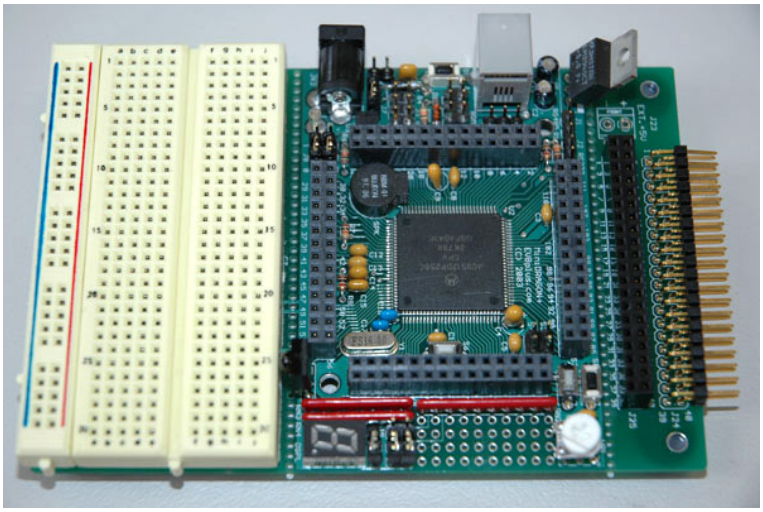


Figure 5.4: Photograph of the MiniDRAGON+ development board

5.5 Axles

Axles were required to connect the motors to the wheels. The size, shape, and material of the axles depended on a number of factors. The axles were machined from mild steel, and the length of axles was defined by the combined length of other components.

The profiles of the axles were chosen by strength requirements under design loads, and the size of related components including couplings, bearings, and the wheel hubs. Two diagrams showing the main parts of the axle assembled and unassembled are shown in Figure 5.5 and Figure 5.6.

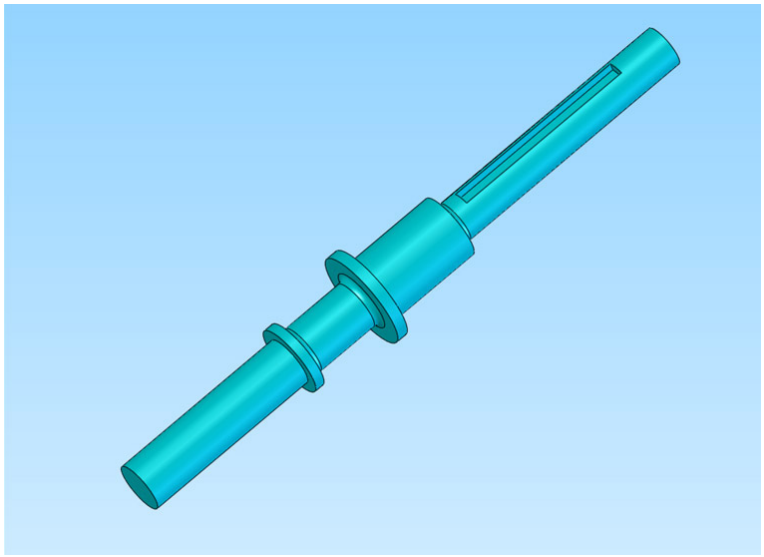


Figure 5.5: SolidWorks model of the axle design

The wheel has a keyed sleeve welded to its hub. The keyed section of axle slides into the sleeve and the wheel is secured with a nut threaded onto the end of the axle. The wheel presses against a shoulder on the axle to eliminate sideways movement. Two roller bearings are mounted between the coupling and the wheel. A rigid coupling connects the main axle to the shaft that exits the gearbox.

A simple analysis was used to determine the force acting on each wheel and axle assembly, and FEA simulation in the CAD modelling software SolidWorks followed to verify strength requirements were met.

Assuming that the mass of EDGAR is 40kg and the mass of the user is 90kg, the force applied to each axle at the wheels is:

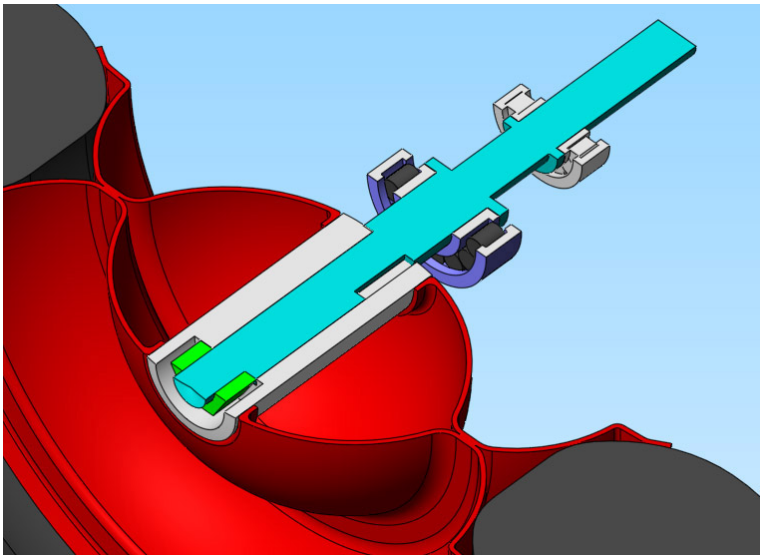


Figure 5.6: SolidWorks model of the axle showing cross section of assembly

$$F = 9.81 \times \frac{40 + 90}{2} = 650N \tag{5.1}$$

The result was then roughly doubled to account for bumps and jolts in the surfaces EDGAR travels on. The design load applied to the axle for stress simulation was therefore 1500N.

After the axle was modelled constraints were placed at bearing locations, and the design load was applied. Results of this analysis showed that the minimum factor of safety with this load was 1.3, which translates to an overall factor of safety under normal use of 3.9 (as normally there would be 650N load applied to each axle). The final axles were machined from mild steel instead of 4340 steel as the above analysis showed that high tensile steel was not necessary. Mild steel has a yield strength roughly half the 710MPa of 4340 steel. This results in a factor of safety of 2 which is acceptable for normal running conditions.

The purple arrows indicate an applied distributed load in Figure 5.7.

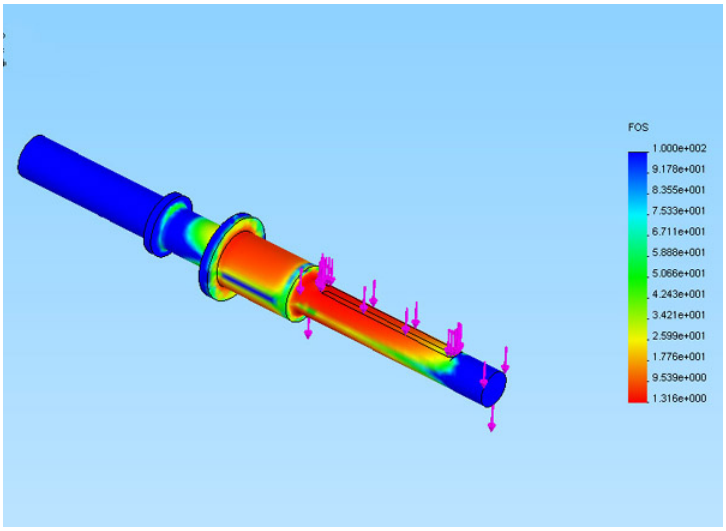


Figure 5.7: Factor of safety distribution over 4340 steel axle with 1500N applied load

A plot of the stress distribution across the axle is shown in Figure 5.8 under the same test conditions. The result showed that the axle design was sufficiently strong under normal operating conditions, and included a satisfactory safety factor to account for expected extra loads.

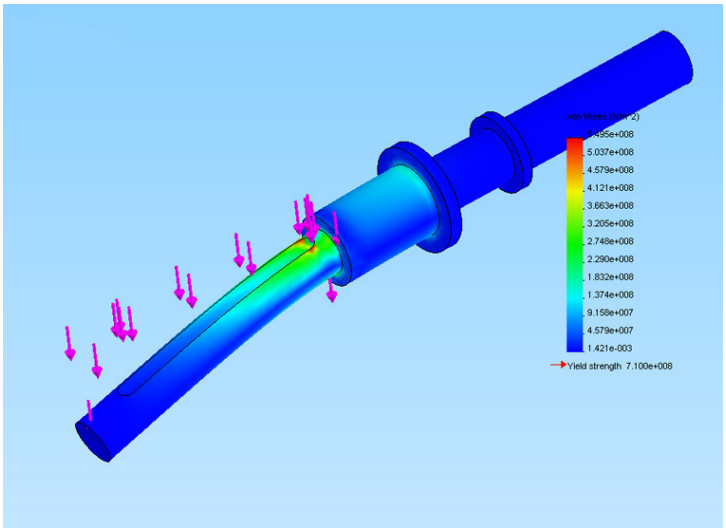


Figure 5.8: SolidWorks stress distribution on AISI 4340 steel axle with 1500N applied load

The output shaft on the motors was also redesigned as the motors were supplied with a short keyed and threaded axle. The supplied shaft was not long enough to fit the coupling so a new shaft with the same main dimension (12mm OD) was machined and mounted in the gearbox.

5.6 Couplings

As EDGAR will be travelling around corners at speed, hitting bumps, and generally being used to its full capacity, the concern of axial shocks being transmitted along the axles and damaging the gearbox/motor assembly became an issue. Isolating the gearbox shaft from axial and radial shocks travelling along the main axles whilst maintaining high torsional stiffness was to be addressed using a flexible coupling.

The Lovejoy L-075 elastomeric jaw type flexible coupling as shown in Figure 5.9 was chosen to provide torsional stiffness and axial shock absorbing capabilities. The coupling has two jaws that mesh between a flexible elastomeric material. The two sides of the coupling connect to either axle. This type of flexible coupling does not require lubrication, and will continue operating once the elastomeric cushion wears out.

For sizing purposes, the maximum power transmitted through the coupling was assumed to be the full 250W power capacity of the motor at 320RPM. The L-075 flexible coupling has a power capacity of 310W at 300RPM which was ample for this design.

Upon installing the flexible couplings on EDGAR, it was found that there was backlash in the flexible couplings. Although a very small amount, this backlash allowed the wheels to rotate with respect to the output shaft of the motors. This small amount of uncontrolled rotation proved to make EDGAR quite uncomfortable to ride because changing the direction of a motor resulted in a jolt as the flexible coupling introduced slack to the system. This jolt was also accompanied with a loud ‘clunk’ which did not reinforce the rider’s faith in the vehicle.

Following from the realisation that backlash was a large problem, the flexible couplings were replaced with rigid couplings. The tight tolerances designed in the drive system proved to be useful in protecting the motors from axial shocks as the order in which the axle and bearings assemble naturally block loads from entering the motors. A photograph of a rigid coupling installed on

an axle is shown as Figure 5.10. See Section 8.2.2 for more information on the replacement of the flexible couplings with rigid couplings.

Figure 5.9: Photograph of a Lovejoy L-075 flexible coupling

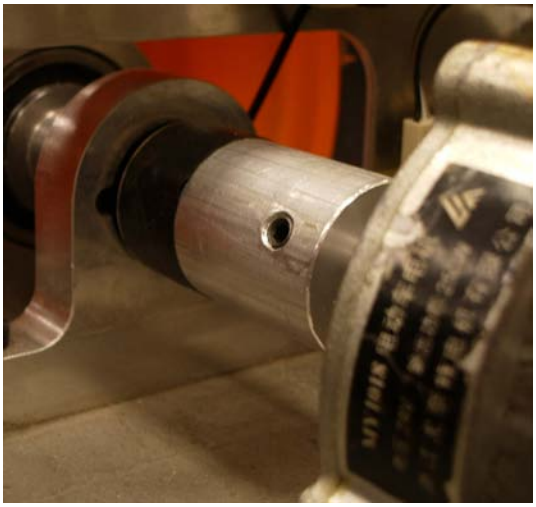


Figure 5.10: Photograph of rigid coupling installed on axle

5.7 Bearings

Two cylindrical roller bearings have been used on each axle to provide both thrust support against the axle moving relative to the gearbox enclosure, and to transmit vertical loads placed on the platform through to the wheels.

The following is a brief calculation used to verify the required capacity of the bearings:

$$\begin{aligned} M_{person} &= 90kg \\ M_{EDGAR} &= 40kg \\ M_{total} &= 130kg \\ F_{total} &= 9.81 \times 130 \\ F_{total} &= 1.3kN \end{aligned} \tag{5.2}$$

Assuming a safety factor of 2, the maximum load applied to the axles would be 2.6kN. Four bearings are used to support the axles, therefore the maximum load applied to each bearing is 0.65kN.

The FAG NJ203 cylindrical roller bearing was appropriate for use as the outer (wheel side) bearing. For assembly purposes, the inner diameter needed to allow it to be fitted over the keyway along the axle. The NJ203 has a dynamic load rating of 17.6kN which is far greater than the expected load of 0.65kN.

The inboard (motor side) bearing was chosen as the FAG NJ202 cylindrical roller bearing, which was the size down from the NJ203. This bearing could be assembled onto the axle from the motor side and therefore did not need to have as large internal diameter. The dynamic load rating of 12.7kN was still far larger than the required 0.65kN.

The thrust capacity of each bearing was not required to be large as the maximum thrust load down the axles was envisaged to be due to bumps or small shocks. The nature of both the NJ202 and NJ203 allow for some protection of the motor gearbox against thrust loads.

Figure 5.11 shows a model of the roller bearings. The inner face can be seen to sit against the edge of the rolling cylinders in Figure 5.12, thereby providing the thrust capability of this type of bearing. A shoulder on the axle

would then need to press against the inner face of the bearing.

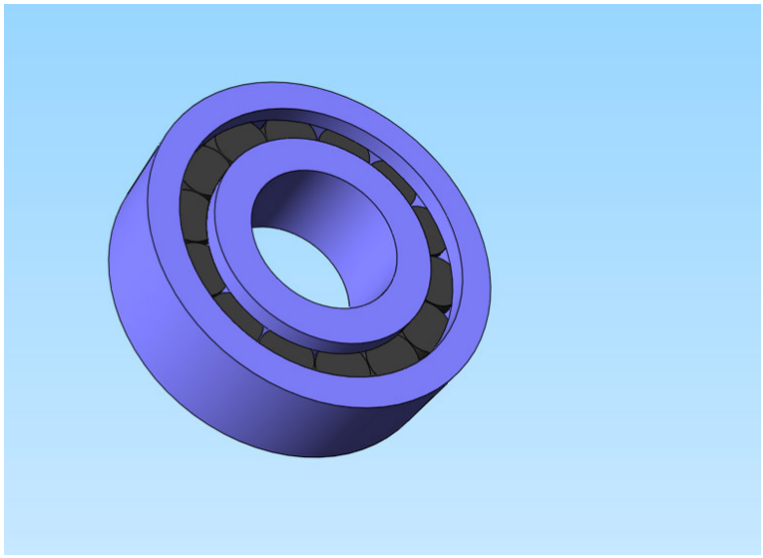


Figure 5.11: SolidWorks model of a FAG cylindrical roller bearing

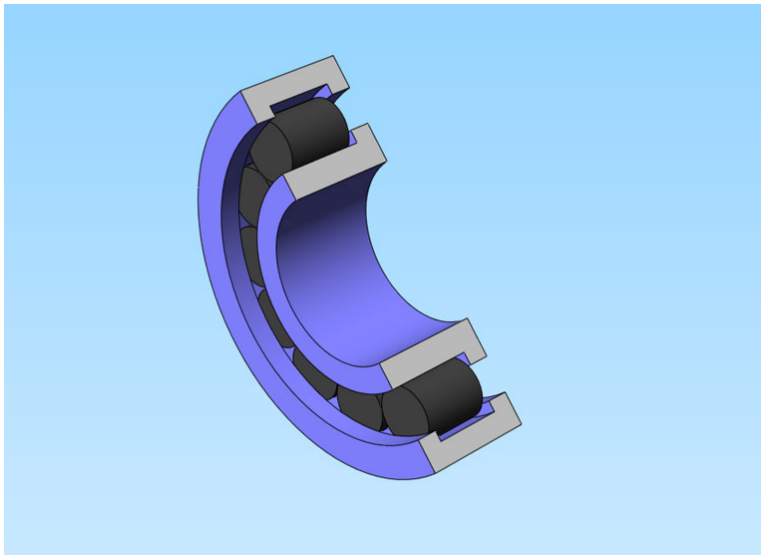


Figure 5.12: SolidWorks model of a FAG cylindrical roller bearing in cross section

5.8 Batteries

The runtime of EDGAR is dependent on the capacity of the on-board power supply. Batteries were quickly determined to be the simplest way to provide on-board power.

The power supply needed to meet the following requirements:

- As compact as possible
- As light as practical
- Fast charge rate (minimum recharging time)
- High discharge capacity to cope with high current draw by motor controller
- Low cost
- Provide sufficient power to EDGAR for 1 hour

Different types of rechargeable batteries include NiCad, SLA, NiMH, and Li-ion. Nickel Cadmium (NiCad) and Sealed Lead Acid (SLA) have both been around for many years and are well suited to low current drawing applications. Nickel Metal Hydride (Ni-MH) and Lithium Ion (Li-ion) batteries are relatively newer and the performance of these types has improved to surpass both NiCad and SLA batteries in high current applications such as for powering EDGAR.

Jaycar D-cell NiMH rechargeable batteries were purchased having the following specifications:

- 1.2V cell charge
- Capacity = 9000mAh
- Power capacity = 10.8Wh
- 5 hour charge rate = 3A
- 1.5 hour charge rate = 9A
- High discharge current capacity

Twenty D-cell batteries in a serial configuration provide 24V for the motor controller and motors. DC/DC regulated step-down circuits reduce the 24V supply to 12V for the IMU and capacitive sensors, 9V for the MiniDRAGON+, and 5V for the steering potentiometer.

Figure 5.13: Photograph of the battery pack for EDGAR

An estimate of the runtime that the batteries can provide to EDGAR was completed assuming that the motors were running at 25 percent of their rated speed of 320RPM. Halving the speed from 160RPM to 80RPM (6km/h) means that the motors each would be consuming 25 percent of their 250W rated capacity. It should also be noted that the power consumption of the motors is significantly more than the other components, and thus they have been omitted from this calculation.

$$P_{EDGAR} = 2 \times 250W = 500W \tag{5.3}$$

$$25\%\, of\ P_{EDGAR} = 0.25 \times 500 = 125W \tag{5.4}$$

As the batteries supply 24V, the current draw for 125W would be:

$$I_{EDGAR} = \frac{P_{EDGAR}}{V_{EDGAR}} = \frac{125}{24} = 5.2A \tag{5.5}$$

Therefore the estimated running time of EDGAR at 50 percent of its permissable speed (80RPM or 6km/h) is:

$$Runtime_{EDGAR} = \frac{9Ah}{I_{EDGAR}} = \frac{9}{5.2} = 1.7hours \tag{5.6}$$

The estimated runtime of EDGAR is around 1.5 hours at 6km/h. This runtime however does not take into account disturbances in the system that would be introduced whilst operating. Actual runtime is expected to be less than 1.5 hours which will be acceptable providing that EDGAR is not driven hard. A photograph of the 20 D-cell NiMH batteries is shown as Figure 5.13.

5.9 Inertial Measurement Unit (IMU)

The system to balance EDGAR required measurements of the angle between the direction of gravity and the platform for PD control. See Chapter 4 for more information on the balance control of EDGAR.

Figure 5.14: Photograph of 3DM-G IMU

The University of Adelaide supplied an Inertial Measurement Unit (IMU) for use with this project. The MicroStrain 3DM-G utilises 9 sensors to provide data about both static and dynamic orientation. The following components make up the IMU:

- Three orthogonal magnetometers
- Three orthogonal angular rate gyroscopes
- Three orthogonal accelerometers
- 16bit A/D converter
- On-board microcontroller

The IMU is able to communicate to the microcontroller through an RS232 serial connection, and provides the angular position of EDGAR for the control system. A datasheet detailing the specifications of the 3DM-G is attached in Appendix A. A photograph of the IMU is shown in Figure 5.14.

5.10 Steering Mechanism

The steering mechanism is critical in providing the user with a simple, smooth, and intuitive way to control the direction that EDGAR moves. Steering configurations including turning the whole top handlebar, tilting the platform from side to side by adjusting the rider's weight distribution, and a single handed twist grip were all considered. It was decided that a twist grip would be most appropriate for EDGAR as discussed in Section 7.3.3.

A self centring grip has been used that twists in either direction to turn EDGAR, and provides the control system with an input that reflects this motion is shown in Figure 5.15 and Figure 5.16. A potentiometer mounted in the twist grip is connected to the MiniDRAGON+ as an analog input, and a dead-zone around the neutral position of the twisting mechanism eliminates erroneous steering.

The assembly of the steering mechanism is as follows:

1. Spacing washer is attached to face of potentiometer using nut
2. Extension rod is fixed to potentiometer using 4mm grub screw

3. Potentiometer as assembled is attached to the inner handlebar tube with three 4mm grub screws which screw into the side of the spacing washer

4. End of outer handlebar tube is attached to inner tube with short allen key bolt screwed into slot of inner tube

5. Spring is inserted to hole in edge of inner tube

6. Potentiometer is centred and end cap finishes the assembly of the twisting mechanism with four more grub screws

7. Twisting mechanism is inserted to main handlebar tube and attached with three grub screws

Figure 5.15: Photograph of handlebar assembly

5.11 Main Structural Materials

The main structural plate supports the weight of a person whilst providing a rigid structure for other components to be attached to. The main structural plate is located underneath the motor and bearing assembly. A fairing is located above the motor and bearing assembly which provides a platform for the user to stand on. The dimensions of the fairing is discussed in Section 7.3.2. The main structural plate is 8mm stainless steel after the first 5mm plate aluminium structural plate was found to be too flexible. See Section 8.2.1 for

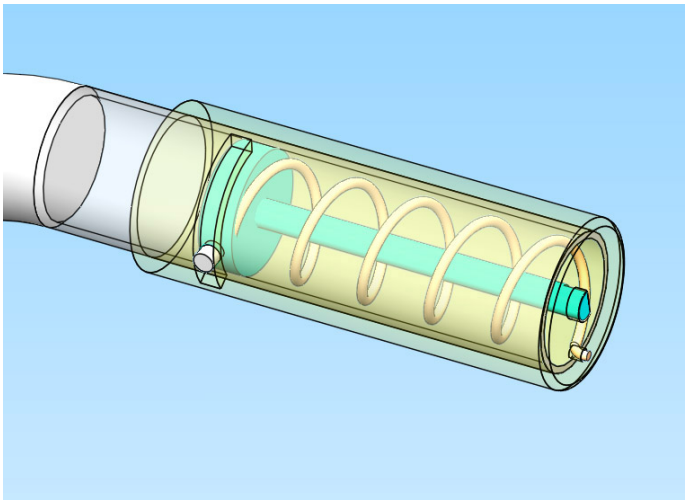


Figure 5.16: SolidWorks model of EDGAR's steering mechanism

more information on the flexibility of the main structural plate. The fairing is made of Medium Density Fibreboard (MDF).

It was decided that EDGAR needed a cover to hide the components on and under the platform, and to give the effect of EDGAR being one assembly instead of a series of individual parts. Thus, the idea of the faring was formed. The fairing wheel covers were vacuum moulded from high impact polystyrene (HIPS). The design and construction of the fairing is discussed in more detail in Section 7.1.3. Photographs of the fairing both unpainted and painted are shown in Figure 5.17 and Figure 5.18.

Kick guards to protect the fairing from the rider's feet were fabricated from aluminium rod of diameter 12mm, with aluminium plate welded in between as discussed in Section 7.4.

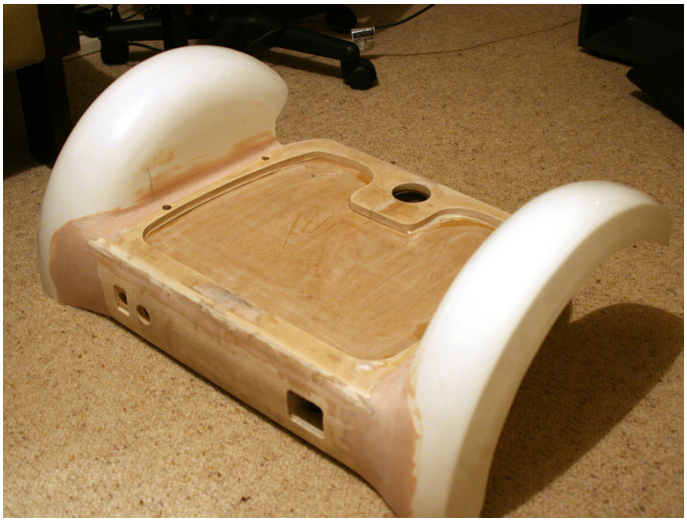


Figure 5.17: Photograph of unpainted fairing showing HIPS and MDF

Figure 5.18: Photograph of painted fairing

5.12 Upright Post and Handlebar

The upright post and handlebar are made of tube aluminium. The shape and ergonomics of the upright post are discussed in Section 7.4 and the mechanical design is discussed in Section 7.1.4. A nylon sleeve is used at the connection point between the upper and lower tubes to ensure wear is kept to a minimum. A quick-release clamp mechanism provides a positive locking connection in the same fashion as a bicycle seat post clamp. The post clamp is shown in Figure 7.9 and a photograph of the nylon sleeve is shown in Figure 5.19.

Figure 5.19: Photograph of nylon sleeve used in post clamp assembly

5.13 Switches and Displays

The user is able to control the behaviour of EDGAR using a number of switches. EDGAR's behaviour was determined during specification development as discussed in Section 3.3. These switches were incorporated into EDGAR's design to provide adequate safety for the rider.

To complement the switches, visual feedback enables the user to easily determine which state EDGAR is in. Visual feedback of EDGAR is comprised of three LEDs with the following functions:

- Green LED to indicate that power is on and EDGAR is ready to balance
- Blue LED to indicate that EDGAR is balancing
- Red LED to indicate low battery level

All three LEDs are connected to digital output pins on the microcontroller. A resistor in series with each LED reduces the voltage from 5V to the LED's required 3.6V. The 'power on' and 'balancing' LEDs states are determined through the software on the microcontroller. The 'low battery level' LEDs state is determined using a voltage divider to interpret the 0-26V (completely flat to fully charged) battery level to the 0-5V range that the analog inputs of the microcontroller need. Using a voltage divider and the microcontroller is more desirable than a hard wired circuit as it allows adjustment of the low battery voltage threshold during testing.

These displays are mounted together on the upper side of the handlebar so that the rider can interpret the current status of EDGAR with nothing more than a glance.

Arising from the aforementioned behaviour, the following switches have been included in EDGAR's design:

- On/Off switch (doubling as a circuit breaker)
- Capacitive foot sensors to detect when a rider has a foot on the platform
- On/Charge switch (to ensure that EDGAR does not power up when the batteries are being charged)

5.13.1 On/Off Switch

The on/off switch has been recessed into the rear of the fairing to prevent accidental power loss to EDGAR. This switch incorporates a 70A circuit breaker to ensure EDGAR is switched off in the event of an electrical fault. All power that EDGAR uses flows through this circuit breaker.

5.13.2 Capacitive Foot Sensors

Two capacitive proximity sensors are mounted in the platform that the rider stands on. They are located where the rider places their feet to detect when either or both feet are on the platform as discussed in Section 3. These sensors are normally high (12V) but switch to 0V when mass is placed above their front surfaces. These signals are reduced to 5V through voltage dividers before being fed into analog inputs on the microcontroller.

5.13.3 On/Charge Switch

A second switch in addition to the on/off switch enables EDGAR to be placed in a 'charge' mode. When in 'charge' mode, a charging lead may be plugged into the back of EDGAR's fairing which allows the on-board batteries to be recharged. This switch drives a relay that severs power between the batteries and the rest of the power distribution board.

5.14 Power Distribution, Cabling and Connectors

Cabling and connectors join the various electronic components in EDGAR. A set of multi-core cables were used to run EDGAR tethered before being tested isolated from a host computer. The tether provided data to and from EDGAR and also power prior to the control system being implemented on-board.

A power distribution board was manufactured by The University of Adelaide Instrumentation Workshop to manage the power needs of EDGAR. The board takes the 24V nominal input from the batteries and distributes it to the various electronic components. Three DC/DC converters provide 5V, 9V, and 12V to the steering potentiometer (5V), microcontroller (9V), IMU (9V), and capacitive sensors (12V). The remaining power flows unregulated to the motor controller which itself regulates 24V power for the motors.

Chapter 6

Software Implementation

In this chapter the implementation of the software used to control EDGAR is discussed. Initially an overview of the software and its implementation is discussed in Section 6.1. In Section 6.2, the communication of data between the components of EDGAR is examined. In Sections 6.3 and 6.4, the dSPACE system and the MiniDRAGON+ Development Board respectively are discussed and their use during the prototyping and final design of EDGAR. The final software implementation for EDGAR is discussed in Section 6.5.

6.1 Overview

The software used in the development of EDGAR integrates the control system to the hardware. Since EDGAR is not driven by an open loop throttle, but by a closed loop system that always returns the pitch to vertical, the software provides the efficiency to implement this system on a microprocessor. There are sufficient safety measures in the software that allow the rider to be safe on and off the vehicle. The software also governs the display system that informs the rider of the current status of the vehicle.

The software has been designed in Simulink, a part of the MATLAB software (Mathworks Inc 2005). Simulink is a powerful Graphical User Interface (GUI) tool which allows complex problems to be broken down into many stages to produce a solution. Blocks are used in Simulink to represent sources, mathematical operations, signal routing, and outputs while lines are used to represent the signals travelling from one block to the next. A simple example of the Simulink block and line structure is shown in Figure 6.1 where the

sine wave and constant are added together then output to a scope. Another feature of Simulink and MATLAB is the Real Time Workshop which allows the interfacing of real world components to Simulink and to display variable parameters on screen in real time. The Real Time Workshop can also generate C code which can then be downloaded onto microprocessors.

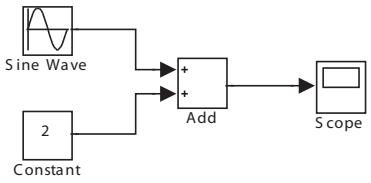


Figure 6.1: A simple Simulink block diagram (Mathworks Inc 2005)

The choice of two platforms for the development of EDGAR was based on availability and support rather than any researched strengths or weaknesses. There are several dSPACE systems available in the School of Mechanical Engineering at the University of Adelaide. The dSPACE rapid prototyping system interfaces with MATLAB's Simulink tool. The dSPACE hardware consists of a PCI card with a PowerPC RISC-based processor, a Breakout Box (BB) and the toolbox for MATLAB and Simulink. The PCI card acts as the interface between the real world objects connected to the ports on the BB, as shown in Figure 6.2, and the dedicated dSPACE blocks that exist in Simulink.

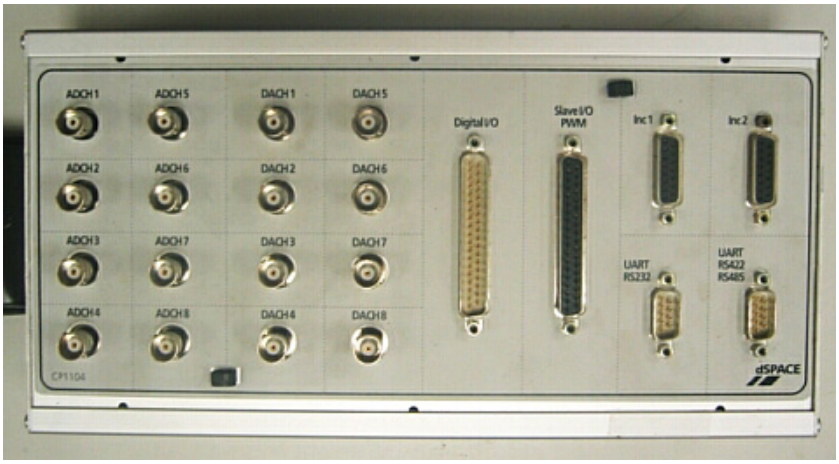


Figure 6.2: Photograph of dSPACE Breakout Box (dSPACE Inc 2005)

The MiniDRAGON+ Development Board, as shown in Figure 5.4 was made available for use by the School of Mechanical Engineering. It is based around a Motorola MC9S12 microprocessor chip which is in turn itself a derivative of the original HC12 chipset. The MiniDRAGON+ is programmed in assembly language which can be generated by the Metrowerks CodeWarrior Software Development Suite for HC12 (FreeScale Semiconductor Co 2005) from code written in C. Dr Frank Wornle of the University of Adelaide School of Mechanical Engineering has kindly provided a Simulink Real Time target for the MC9S12 chipset which allows Simulink blocks representing ports on the MiniDRAGON+ to be interfaced to Simulink. This allows pertinent values of these blocks to be changed in real time. This rapid prototyping approach is similar to dSPACE in that regard.

EDGAR has been tested using software rapid prototyping. Not to be confused with regular hardware rapid prototyping where a Computer Aided Design (CAD) is made into an actual object quickly using inexpensive materials, software rapid prototyping uses an interface to the actual hardware and componentry to enable algorithms and values to be changed in real time while being able to watch the results on the hardware. This saves time spent changing small parts of software and recompiling and downloading, to try and eliminate problems that are present. It is also useful in finding problems with the hardware or component interfacing early on in the prototyping stage. Both the dSPACE system and the MiniDRAGON+ development board were used to rapid prototype EDGAR in the tethered and untethered states, and this rapid prototyping was one of the main reasons EDGAR was able to be completed within the scope of one year.

6.2 Data Input and Manipulation

The connections between the components and the rapid prototyping tools are many and varied in both protocol and electronics. As part of the development of the software for EDGAR each connection had to be examined to make sure it would have coherent and cogent data passing through it.

The motor controller has three different types of inputs to control the signals sent to the motors. Either RS232 communication, remote control (RC) radio pulse width modulation (PWM) signals or analogue voltage signal.

The Inertial Measurement Unit (IMU) has only two types of communica-

tion protocols that it uses, namely RS232 and RS485. Both are serial protocols allowing data to be sent at greater distances while RS485 allows the connection of multiple devices on the same data lines. The RS232 protocol was used for EDGAR and both dSPACE and the MiniDRAGON+ have serial communication lines. As the IMU delivers two signed 8 bit integers for each orientation value to the rapid prototyping systems, a conversion from the 8 bit signed integers to useful angular data was necessary as shown in Figure 6.4.

A potentiometer has been implemented for the turning system of EDGAR, it requires that a voltage range of 0 to +5V be able to be read in by the prototyping system. In both the dSPACE system and the MiniDRAGON+, sufficiently resolute analogue-to-digital (A/D) converters are available.

The capacitive proximity sensors are run from a 12VDC supply and feature a normally closed type of contact. When a foot is present, the contact goes to open circuit levels. This drop from +12V to 0V is necessary to capture and is done by using generic single bit input ports on the dSPACE system and saturating the signal at 10V while the MiniDRAGON+ also uses a single bit input for each sensor with the signal saturating at 5V in this case. See Section 8.1.4 for more information on the integration of the capacitive sensors.

The user display LEDs required only a digital output of 5V with a resistor in series; this was accommodated by both the dSPACE and the MiniDRAGON+ board.

6.3 dSPACE DS1104 R&D Controller Board

The dSPACE DS1104 R&D Controller Board along with the dSPACE BB have been developed for rapid research and development prototyping in the areas of control and software interfacing. The controller board has been designed to be user friendly for university use in engineering related fields whilst still economical for the industrial environment, where saving money whilst prototyping is paramount. It has a full graphical user interface in the form of ControlDesk and possesses a Simulink tool box which makes the system easy to interface, program and use. It is a real-time interface to Simulink which provides A/D, D/A, digital I/O lines, incremental encoder interface and PWM generation, all functioning from a PC through a PCI card.

While the group was prototyping a tethered EDGAR, the dSPACE system was connected to the chassis of EDGAR by a 10m long tether. The tether

included the necessary communication lines between the structure of EDGAR and the processing centre. The ports used on the dSPACE BB were:

- RS232 port (for the IMU)
- 1 16-bit A/D converter channel (for the potentiometer)
- 2 16-bit D/A converter channels (for the motor controller)
- 6 parallel single bit digital input/output channels (for the foot sensors, LEDs and ground)

The development of the software for EDGAR in the tethered configuration began with the developed control system and continued with the implementation of the real world componentry that were necessary to satisfy the specifications. Initially it was necessary to get the IMU working from within Simulink. The group had been given a Simulink system from a previous project that obtained IMU data through the dSPACE system and converted it from its signed 8-bit integer format to useable angular measurements. Figure 6.3 shows the subsystem created to receive the input data from the IMU and convert it to a useful form. It should be noted that due to the orientation of the IMU within the chassis of EDGAR, only the roll of the IMU is of interest as this actually translates into the pitch of EDGAR. There has also been included an optional lowpass filter that is used to smooth the IMU output signal. Figure 6.4 shows in detail the conversion from the bytes received from the IMU to an angle in degrees.

After testing the IMU in software, the focus shifted to the motor controller. There were three ways to control the motor controller:

- RS232 control
- RC Radio PWM control
- Analogue voltage control

It was known beforehand that the MiniDRAGON+ only has two Serial Communication Interfaces (SCI) and one would be used for external connection to MATLAB and Simulink. Therefore only one RS232 enabled device would be able to be used on the untethered version of EDGAR. That device was to be the IMU as it had no other way of communicating with the rapid

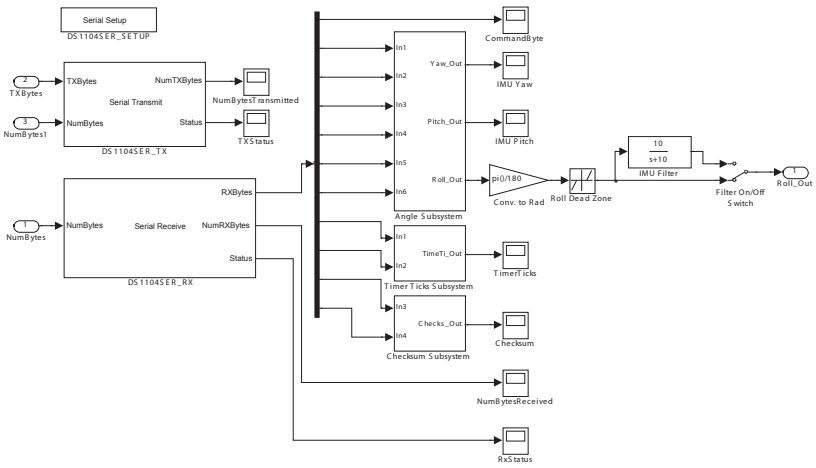


Figure 6.3: Simulink IMU Subsystem

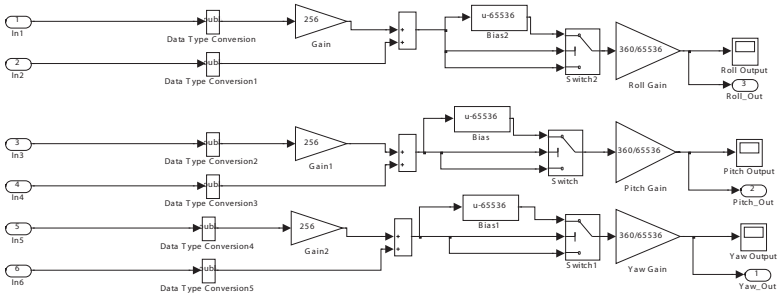


Figure 6.4: Simulink IMU angle conversion Subsystem

prototyping systems. It was decided to use the RC radio PWM signal. Pulse Width Modulation (PWM) is the variation of duty cycle of a pulse train in order to control the speed of an actuator whilst still using the entire rated torque of the actuator. Remote Control (RC) radio PWM signals are a special form of these signals, set up to be used as a common standard for all remote control vehicles, the communication signal is shown in Figure 6.5.

Unfortunately, there were complications with the PWM signal and the interpretation of it in the motor controller. The attempts to rectify the complications are documented in Chapter 8. Therefore it was chosen to use analogue voltage control. This control used two D/A ports on the dSPACE BB, as shown in Figure 6.6, with 0V corresponding to full reverse, +2.5V being stationary, and +5V being full forward. Initial concerns were that this varying

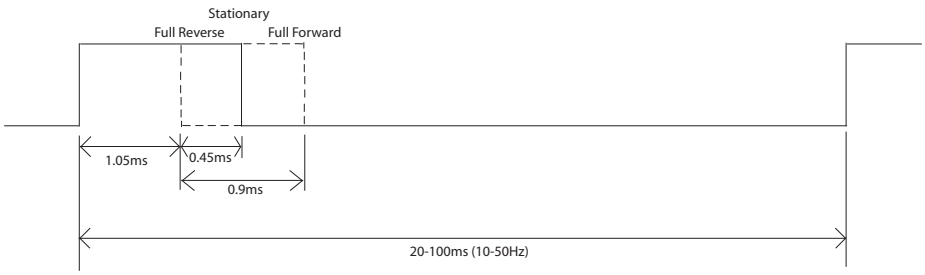


Figure 6.5: RC PWM signal diagram

voltage control method would not have enough resolution for the application but initial testing showed that this was not the case. Once these two steps were completed, open loop testing of the controller was initiated.

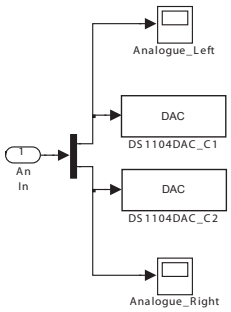


Figure 6.6: Simulink DAC outputs

The pitch angle from the IMU was able to drive the motors in unison. From this point the turning potentiometer was implemented via an A/D port, as shown in Figure 6.7. In order to satisfy the basic specifications it was necessary to implement turning by subtracting a value from the speed of one wheel while adding the same value to the other. In order to satisfy the safety specifications it was desired that the turning value should decrease as the speed of the vehicle increased to avoid turning a corner too sharply.

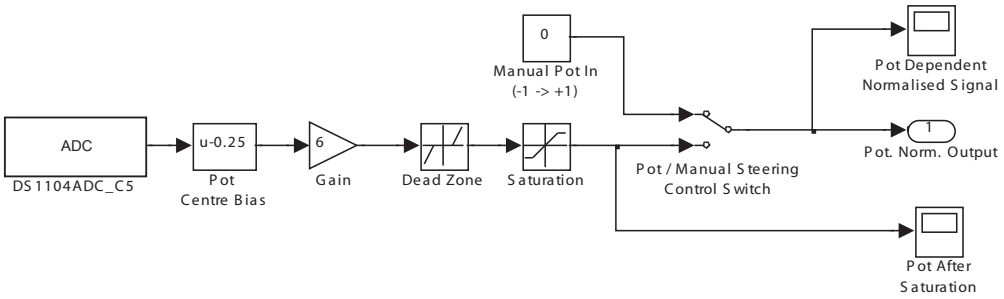


Figure 6.7: Simulink model of potentiometer based turning subsystem

After the turning system was modelled, safety features and user display hardware were added to the Simulink model. The foot sensors each used a digital input, as shown in Figure 6.8, and were arranged in Simulink using a truth table which used XOR logic to inform EDGAR when a rider was no longer on the platform.

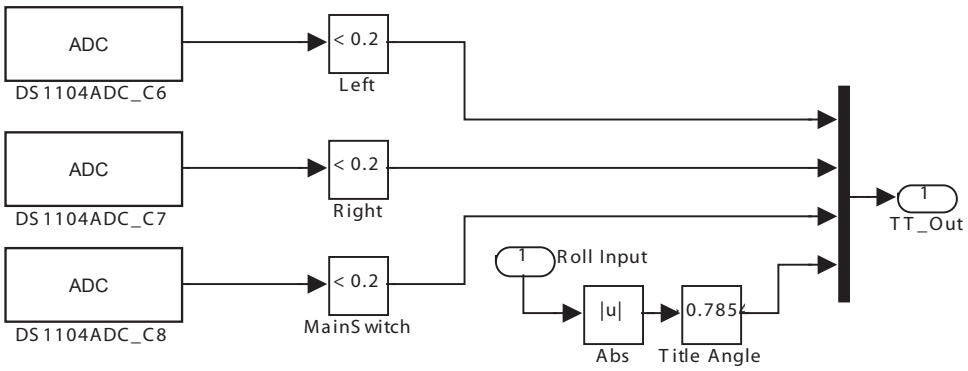


Figure 6.8: Simulink model of truth table preparation subsystem

After the main hardware components had been successfully included in the Simulink model, the classical PD control system was added. ControlDesk enabled values in the model to be changed 'on the fly' and the response observed in real time. In Figure 6.9 the control system is shown with both proportional and derivative control signals flowing into the turning subsystem in the model, Figure 6.10.

The complete Simulink model for EDGAR when tethered is shown in Figure 6.11.

The main use for ControlDesk with the tethered set up was not only to see the effect of changing the proportional and derivative gains but also to see the effect of filtering various signals in the control system. The ControlDesk layout for EDGAR when tethered is shown in Figure 6.12.

Once EDGAR was balancing tethered to dSPACE, the main project goals had been satisfied and focus shifted to the first extension goal of implementing EDGAR's control system on-board. Untethered control required the control system to be implemented on the MiniDRAGON+.

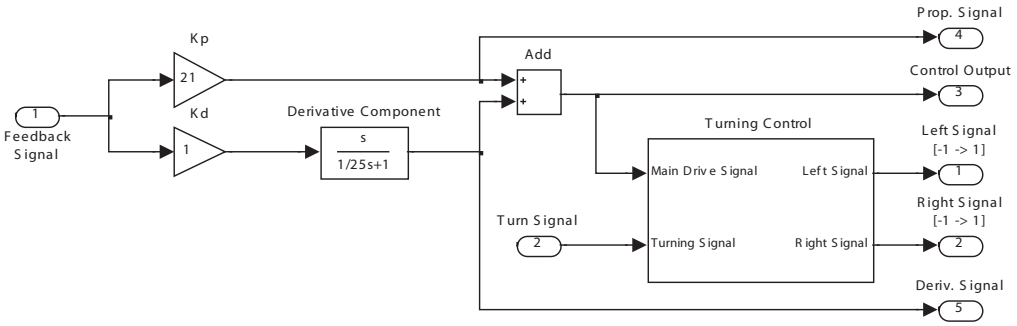


Figure 6.9: Simulink model of control system

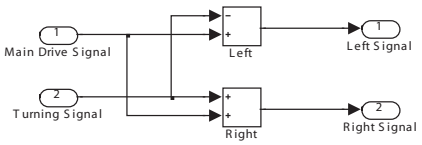


Figure 6.10: Simulink model of the turning part of the control system

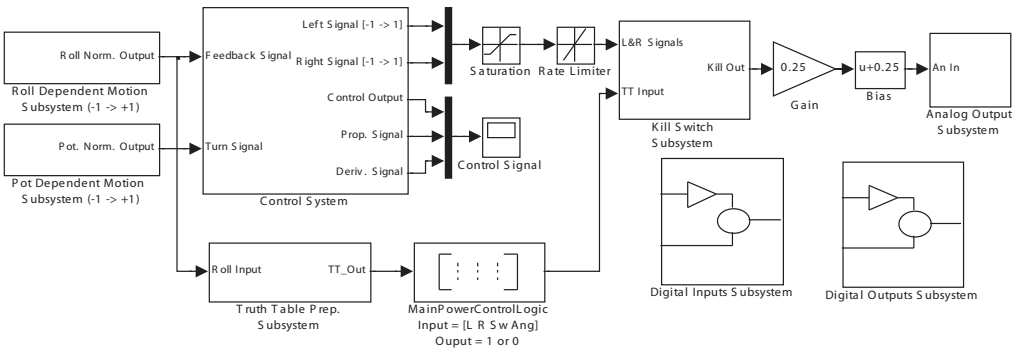


Figure 6.11: Final Simulink model of dSPACE-tethered EDGAR

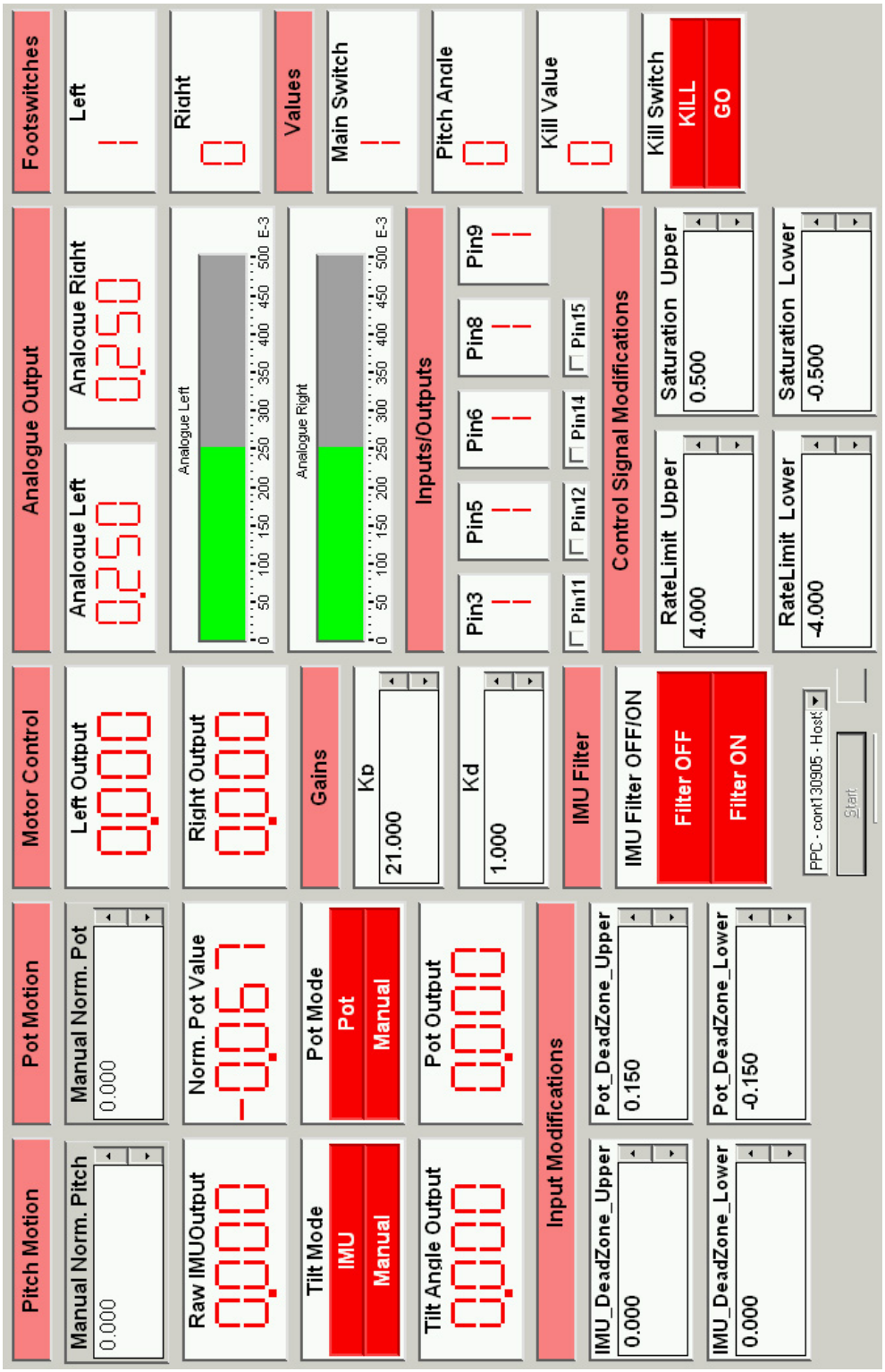


Figure 6.12: ControlDesk setup for dSPACE-tethered EDGAR

6.4 MiniDRAGON+ Compact Development Board

The MiniDRAGON+ Compact MC9S12DP256 Development Board is an inexpensive development board for the Motorola MC9S12 chipset made by Wytec Ltd. in the United States of America. It has been designed to be user friendly and economical for students and robotic enthusiasts. It has many features considering its small size and enough I/O ports for most robotics applications. The MC9S12 chipset is based around the HC12 chipset often used in many embedded systems throughout the commercial and industrial worlds.

The MiniDRAGON+ is programmed in assembly language which was generated by the Metrowerks CodeWarrior Software (FreeScale Semiconductor Co 2005) from code written in C. Although it was possible for the group to implement the untethered software in C code, it was decided to continue using Simulink as had been used for control system development and tethered control implementation. This was made possible by Dr Frank Wornle of the University of Adelaide School of Mechanical Engineering who provided a Simulink real-time target for the MC9S12 chipset. This allows Simulink blocks representing ports on the MiniDRAGON+ to be interfaced to Simulink and act in a similar way to the dSPACE rapid prototyping system. Dr Wornle also included the cross compiler from Simulink to appropriate C code for Metrowerks CodeWarrior for downloading onto the MiniDRAGON+. Using Simulink's Real Time Target enabled the continued use of Simulink to implement EDGAR's control system.

When running EDGAR untethered, the MiniDRAGON+ was mounted to the chassis and connected to the computer via a serial cable for programming and external interfacing. The ports of the microcontroller used in the final untethered iteration of EDGAR were:

- 1 Serial Communication Interface - SCI0 (RS232 port for the IMU and when programming)
- 2 16-bit A/D converter channel (for the pot and batteries)
- 2 16-bit PWM ports each running at 50Hz for motor control
- 6 digital input/output ports (for the foot sensors, LEDs and ground)

The MiniDRAGON+ was programmed in Simulink to communicate with the IMU as shown in Figure 6.13.

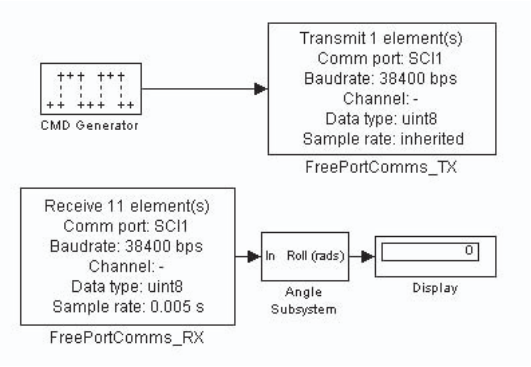


Figure 6.13: Simulink model for IMU communications with MiniDRAGON+

As with the tethered order of connecting and interfacing components, the analogue voltage control of the motor controller was connected after the IMU was communicating correctly. The MiniDRAGON+ is a smaller version of the Dragon12 Development Board made by Wytec Ltd. On the Dragon12 the I^2C bus is connected to two Analog Devices AD5311 I^2C D/A converters, on the MiniDRAGON+ however this is removed to save space but was built into Dr Wornle's real time target as two separate D/A channels. Therefore the breadboard on the MiniDRAGON+ was removed and D/A chips were soldered onto the circuit board.

The D/A chips were connected to the I^2C bus which required two lines, the SDA (Serial Data Line) and the SCL (Serial Clock Line) to operate the D/A converters. After pull up resistors were placed on the SDA and SCL lines, the voltage output, V_{out}, was connected to the motor controller channel inputs and worked successfully. However during the testing of EDGAR, the D/A converters were damaged and rendered inoperable (see Section 8) and thus the PWM ports were used. A Simulink model of the servo PWM ports is shown in Figure 6.14.

The turning system was nearly exactly the same as the tethered configuration except the voltage passing through the pot for simplicity's sake was 12V. This voltage was too high for the MiniDRAGON+ to properly read each varying level of the analogue signal, thus a voltage divider circuit was implemented to convert the variable input between 0 to +5V. It is shown in Figure 6.15.

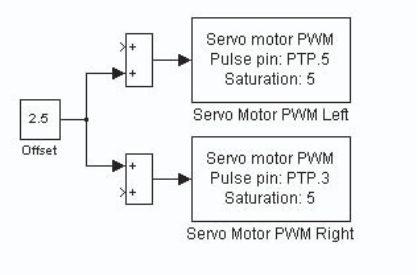


Figure 6.14: Simulink model of Servo PWM outputs

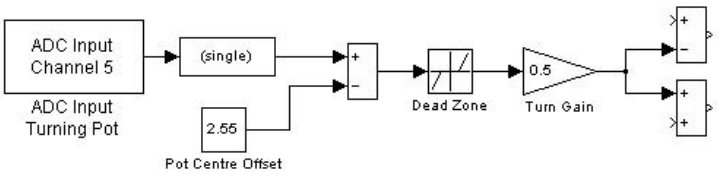


Figure 6.15: Simulink model of A/D potentiometer input

During testing of EDGAR, the D/A converters were damaged and rendered inoperable. See Section 8.1.2 for more information on other problems communicating between the MiniDRAGON+ and the motor controller. The PWM generation problems were resolved following the loss of the D/A converters, and communication with the motor controller was reverted back to PWM.

The LED outputs and foot sensor inputs used single digital I/O ports which saturate at 5V so the foot sensors registered the normally open and normally closed contacts with ease. The implementation of these systems can be seen in Figure 6.16.

The control system implemented on the MiniDRAGON+ Development Board is shown in Figure 6.17. The main difference between the dSPACE implementation is the discretisation of the entire system for it to be able to run on the MiniDRAGON+ board. Note the gains which are able to be adjusted from within Simulink for rapid prototyping in an untethered configuration.

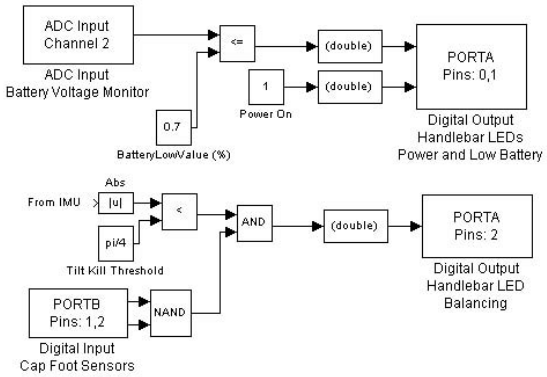


Figure 6.16: Simulink model of single bit inputs and outputs for LEDs and capacitive foot sensors

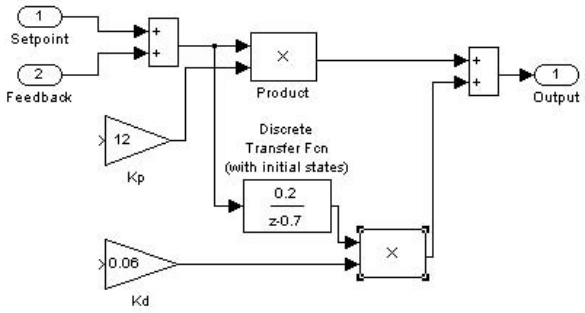


Figure 6.17: Simulink model of discrete control system with adjustable gains

6.5 Final Software Implementation

The final design of the untethered, fully-operational EDGAR in the Simulink blockset is shown below in Figure 6.18 with the MiniDRAGON+ real-time inputs and outputs in yellow and blue respectively.

As can be seen in Figure 6.18, there are many parts to this control system. The IMU command byte is sent out via the FreeCommPort_Tx block to the IMU. This output is driven by a pulse generator which sends the command byte periodically. The desired command byte is 0x14 which returns the Gyro-Stabilised Euler Angles, that is pitch, roll, and yaw of the IMU, through the FreeCommPort_Rx block which passes through the angle subsystem similarly to the dSPACE angle subsystem (Figure 6.4) and yields a roll value in radians. This value is fed into the controller as the feedback value and also feeds onto

the angular tilt cutout switch which is set at $\pi/4$ radians from vertical. Once in the controller, the IMU roll angle is compared against the desired setpoint (also adjustable) and the error between the actual angle and the setpoint is passed through a proportional gain, and in parallel the derivative function. The turning signal is introduced to the system from the potentiometer through an A/D converter. This value is conditioned to be similar to the controller signal and weighted in terms of the importance (aggressiveness) of the desired turning against the balance system. This value is then added to the wheel which corresponds to the desired turning direction of the pot while the value is then subtracted from the wheel opposite to the turning direction of the potentiometer steering system. This value goes through a saturation which makes sure the outputs to the PWM ports are from -2.5V to 2.5V. Both signals then pass through a switching system which enables the kill switch and then are offset by 2.5V to make them between 0V and +5V which drive the motor controller via the Servo PWM ports. The Servo PWM ports take the range of 0V (full reverse) to +5V (full forward) and converts it to a servo PWM signal where 1.05ms represents full reverse and 1.95ms represents full forward.

The kill switch system is determined by 'AND' logic between the tilt threshold of EDGAR and the two capacitive proximity sensors acting as foot sensors. The proximity sensors are set to be normally closed and thus a foot on the sensor is a logic zero level. Thus when both sensors are applied to 'NAND' logic the only time a logic high level is registered is when both feet are off the platform. This value yields a non-zero output of a high logic level from the'AND' block. This non-zero level moves the toggle in the switches to the second input which is the kill value of 0V, offset to +2.5V which corresponds to stationary.

The last system implemented on the MiniDRAGON+ is the 'power on' LED and the 'low battery' LED. Whenever the program is on and running the green power LED is on. The battery voltage levels is brought into the system through a voltage divider which inputs into a A/D converter. This level is set to 80% of the maximum voltage levels of the batteries which corresponds to 20V.

The completed Simulink block diagram shown in Figure 6.18 was then compiled into C-code and finally downloaded through Metrowerks CodeWarrior Software (FreeScale Semiconductor Co 2005) onto the MiniDRAGON+. The C-code generated is shown in Appendix B.

As can be seen, the software implementation on EDGAR experienced many issues (refer to Chapter 8 for more information). The desired outcome of the software implementation of the behaviour of EDGAR to be a self-balancing scooter has been successful.

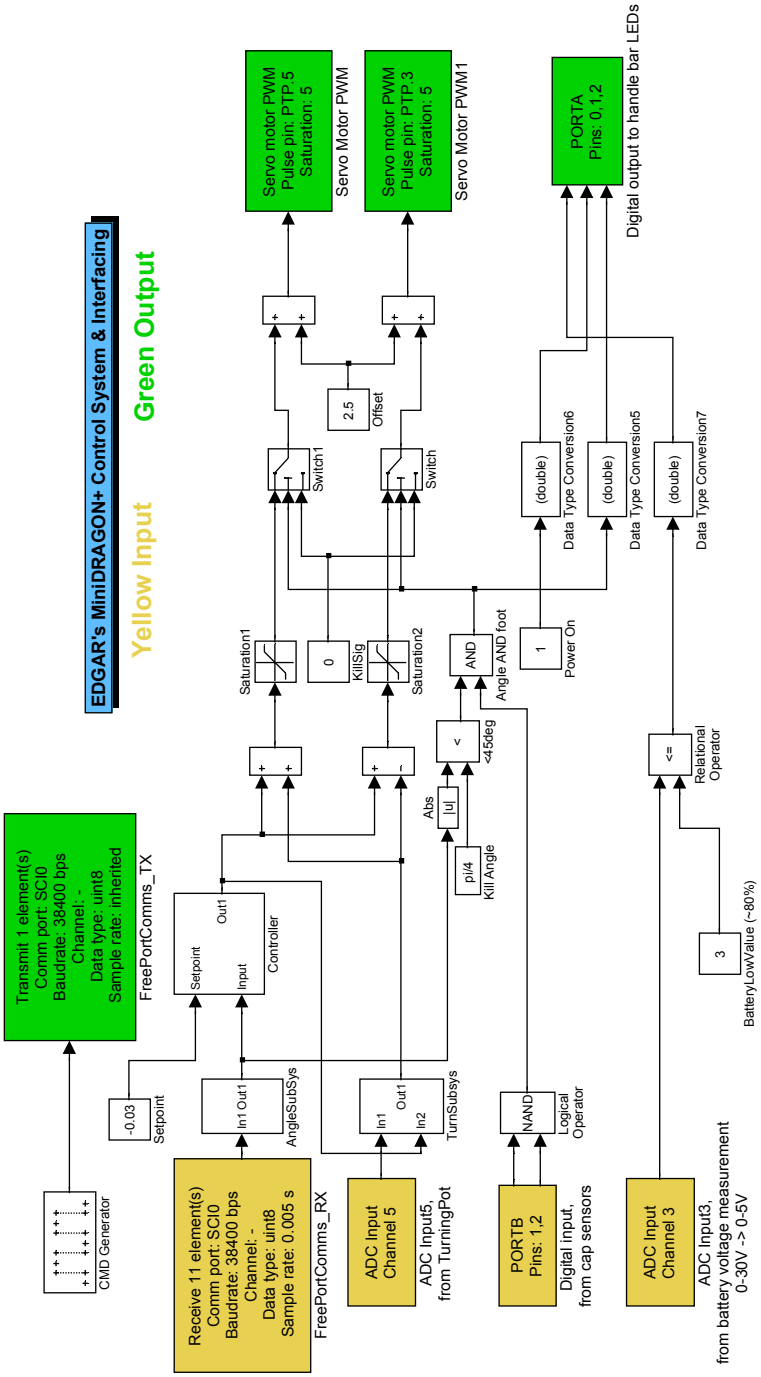


Figure 6.18: Simulink model of final MiniDRAGON+-based untethered software implementation on EDGAR

Chapter 7

Hardware Integration

This chapter covers the structural design and integration of the individual hardware components of EDGAR. It also explains the ergonomic and aesthetic aspects and considerations.

7.1 Structural Design

The design of the structure and hardware of EDGAR was an organic process throughout the planning and fabrication stages. The initial designs were primarily based on the basic specifications that had been decided on early in the year. The hardware and structural designs evolved to better satisfy these specifications. These were further updated as members of the workshop were consulted and suggested changes to the structure to improve the ease of manufacture.

7.1.1 Evolution of Hardware

The overall concept of the initial structure remained similar from the beginning of planning to the end of production; the design evolution was more concerned with the individual structural components.

It was realised that a structure underneath EDGAR was needed to protect the components bolted to the underside from hitting large protrusions when EDGAR was being ridden. The design for a bash guard started as a series of welded bars designed to withstand the force of impact. After considering this more, a folded aluminium plate was decided on, primarily for its ability to shield components from a reasonable amount of water, mud and dust etc.

Also, the folded shape would remain much easier to manufacture. Two views of the bash guard are shown in Figures 7.1 and 7.2.

Figure 7.1: Photograph of the bash guard

Figure 7.2: Photograph showing the location of the bash guard

To support the weight of the rider on EDGAR, it was firstly decided that four angle aluminium sections would be welded to the base plate protruding upwards (one in each corner) which the fairing would rest on and transfer the weight through the fairing to the base plate. Similarly to the bash guard, it was decided to change the design of this component and fabricate all four supports into a single box structure that would be made from sheet aluminium,

as shown in Figure 7.3. This shielded the inside components more effectively from the elements, and again, would also make the process of assembling EDGAR much easier. As the final design of the box was bolted and not welded to the main base plate, it gives much more flexibility to the design, if ever someone wanted to change EDGAR to fit different or more components. As it was necessary to change the plate from the original aluminium to a thicker steel plate (refer to Chapter 8), it would have meant re-fabricating the support posts and attaching them to the new base plate. The other main problem with welding components to the main plate is due to the extreme temperatures involved in welding, the plate would have most likely undergone warping. In the case of EDGAR, this would have lead to unsatisfactory results, due to the very small tolerances that were present concerning the drive train.

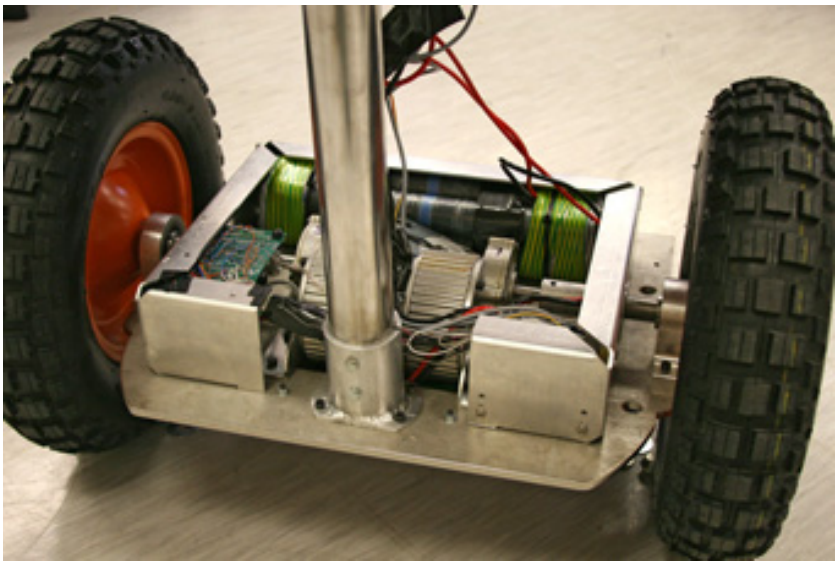


Figure 7.3: Photograph of the final assembly

The axles were one other main component that took many iterations to achieve a satisfactory result. The main obstacle to overcome was that the wheels and drivetrain of EDGAR needed to be able to sustain not only forces in a radial direction, but also axial forces. Two thrust bearings were used, one for each direction. A series of shoulders were needed to support forces in different directions for the wheels and bearings. Four is the minimum number of shoulders for each axle, two for the bearings and two for the wheel. In this configuration, the bearings have to be assembled onto the axles from different

directions. The other main constraint is that the outside bearing has to be large enough to fit over the shoulder required for the wheel hub. The final CAD design of the axle is shown in Figure 7.4.

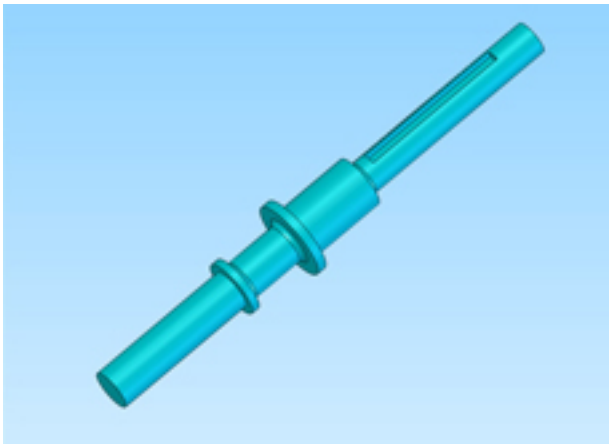


Figure 7.4: SolidWorks rendering of axles.

7.1.2 Drivetrain

The most important hardware system on EDGAR is the drivetrain. The mounting of the drivetrain components was of great importance. Because of the large forces that act on the drivetrain, a very high degree of accuracy is needed to achieve smooth and quiet operation of EDGAR. Individual components had to be attached to EDGAR securely, but needed also to be easily removable when disassembling the scooter for maintenance.

7.1.2.1 Bearings

Due to the use of the rigid couplings instead of flexible ones, the mounting of the bearings was rather critical. As EDGAR would be subjected to heavy loads, it was imperative that the forces would be transferred through the bearings to the base plate, and not to the motor. This is because the shaft of the motor is not strong enough to support dynamic impulse loads as experienced when driving over bumps.

Apart from the radial loads experienced by the bearings under normal conditions, the bearings are subjected to axial loads as EDGAR turns, or is ridden across inclines. These forces either pull or push the axle into the

motor, decreasing the efficiency of the drivetrain, and very possibly damaging the motor as well. For this reason, the bearing housing had to be designed to bear radial and axial loads from the bearing. Radial loads can be accounted for more readily than axial loads. The FAG NJ type bearings that have been chosen are manufactured with a shoulder on the inner and outer part of the bearing. The inner shoulder is flush against a corresponding shoulder on the axle. The shoulder on the outer section of the bearing is flush against the bearing housing, as shown in Figure 7.5.

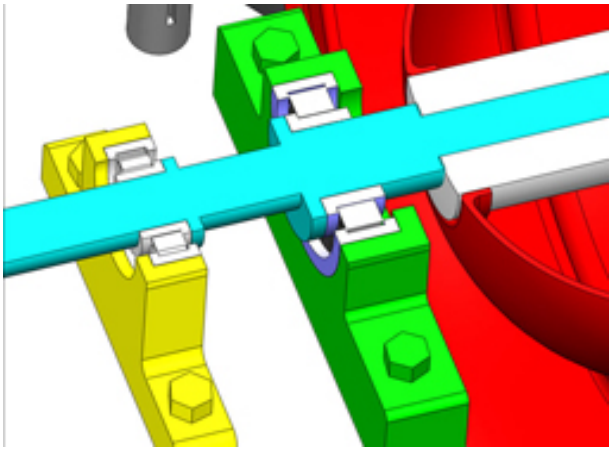


Figure 7.5: SolidWorks cutout of bearings and drivetrain

To support thrust in both directions, two bearings, one a mirror of the other, must be used. The housings will both be bolted onto the main structural platform. They are placed as far apart as space permits (approximately 70mm) as this increases the rigidity of the axles. The housings are 85mm from the centre of the bearings to the platform, and are attached with 2 bolts each. The housings were milled from plate aluminium 15mm and 20mm thick respectively. Two bearings and their corresponding housings are shown in Figure 7.6.

7.1.2.2 Motors

The motors, for the same reasons as the bearings, required accurate mounting on the main structural plate. Because of the relatively large torque of the motors, the mounting method had to withstand large forces. It was necessary to mount the motors as low as possible to increase the stability of EDGAR. There was a trade off between this necessity, the ease of manufacture, and the

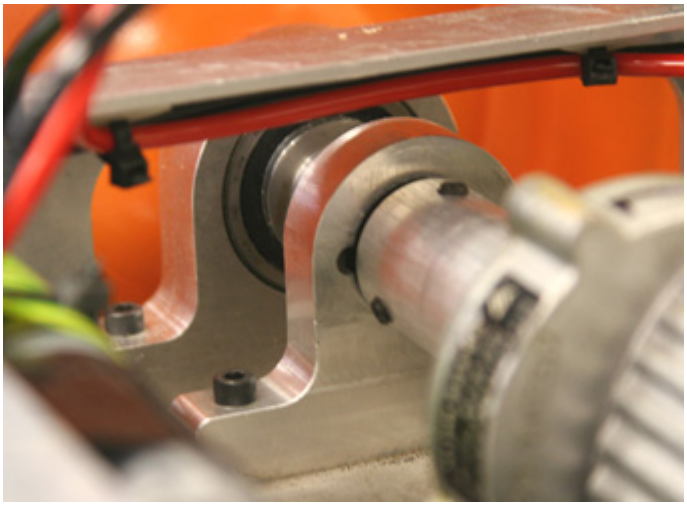


Figure 7.6: Photograph of bearings and bearing housings

complexity of the drivetrain design. It was decided that the motors would be mounted in such a way that the protruding gearbox would sit almost touching the base plate. The motors could have been mounted lower, but the problems that would have arisen from having to cut irregular shapes in the base plate and running out of room to mount bearings would have outweighed the benefits from the extra stability arising from the lower motors and drivetrain assembly. The motors were mounted from above and below the main structural plate by two pieces of angle aluminium which were bolted by three bolts together through the plate, and had one bolt each attaching the motor as shown in Figures 7.7 and 7.8.

It was found that the angle aluminium attached very securely to the structural base plate, but after extended use of EDGAR, the two bolts that attached the motors to the angle aluminium came loose. This caused very unpleasant sounds, and also added extra disturbances in the form of backlash into the system which made balancing more difficult. After a generous amount of Loctite was added to the holes, there has been no problem with these bolts becoming loose.

7.1.2.3 Wheels and Axles

The wheels of EDGAR are securely attached to the axles using a recessed nut. The two axles transfer the radial and axial forces to the bearings which are

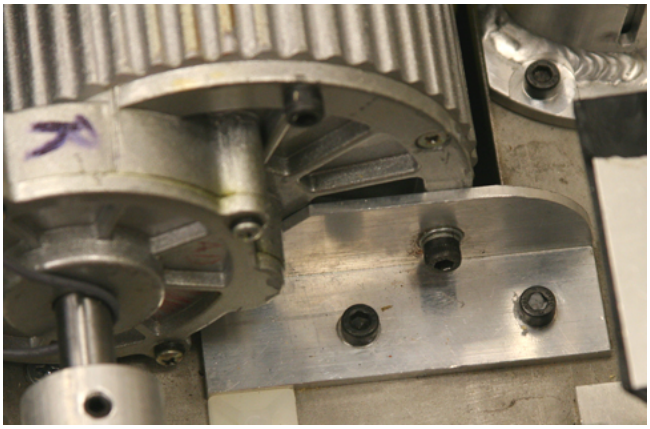


Figure 7.7: Photograph of top motor bracket

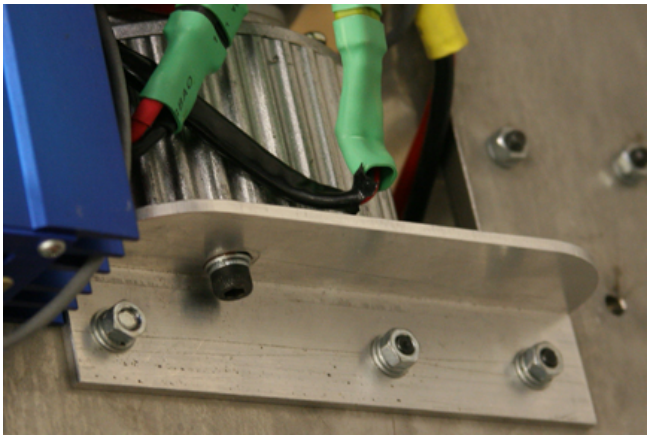


Figure 7.8: Photograph of bottom motor bracket

then transferred to the main structural platform through the bearing housings. It was originally decided that axial misalignment would be adsorbed by flexible couplings, while still transferring the torque along the axle to the wheel assembly from the motor. After implementation of the flexible couplings, it was seen that the backlash due to compression of the rubber element was too great, and a rigid coupling was needed. The rigid couplings were designed, fabricated and fitted. Due mainly to the rigidity of the 8mm stainless plate and the accuracy with which all other components had been manufactured, there were only negligible misalignments between the axles and motors. Surprisingly, the wheels with the rigid couplings spun more freely than they had previously done with flexible couplings.

7.1.3 Fairing and Kick Guards

The purpose of the fairing is two fold. Firstly, it protects the components from the environment and users. It also keeps the user's feet and clothes away from the moving parts, primarily the wheels and drivetrain. The second use of the fairing is that it is one of the main components that communicate the aesthetics of EDGAR, which can be read below in Section 7.3.

7.1.3.1 Structural Design

The weight of users standing on the fairing is transferred to the structure through an aluminium box bolted to the main structural plate. The main section of the fairing was made from MDF (Medium Density Fibreboard). This gives the fairing adequate strength and stiffness while still being easy to work with, as it is a fairly soft material to sand and file. The fairing is attached to the base plate purely by the kick guards. The end of the kick guards protrude through the bottom of the base plate. They have been threaded, which means that it is a simple task of fastening a nut to each 'leg' of the kick guards to completely secure the fairing to the base plate.

7.1.3.2 Construction

The wheel covers were manufactured from High Impact Polystyrene (HIPS). HIPS is a thermoplastic, similar to acrylic. Being a thermoplastic means that the HIPS will soften and flow when it is heated, making it appropriate for vacuum forming. HIPS has high impact resistance, making it a good choice for use as the wheel covers of EDGAR. The HIPS was vacuum formed over a positive made of MDF. Because of the relatively expensive cost of thick MDF, many thinner sheets were laminated together to create a solid block, from which the mould was carved. The main tool used to shape the block of MDF into a wheel cover shape was a large disk sander while a band saw, a belt sander and a planer were also used. Once the shape was arrived at, it was polished to make sure the MDF mould wouldn't stick in the HIPS after vacuum forming. To vacuum form the HIPS, it was warmed under a large heater bank, to a point where it was quite soft. The HIPS was lowered onto the mould, and the vacuum was turned on. After the HIPS was cooled, the positive MDF mould was removed and reused to make the cover for the other wheel.

The main section of the fairing was made from MDF that was bonded using PVC glue, and in a similar way to the mould for the wheel covers, it was formed mostly on a large disk sander, to achieve the correct radiuses and then finished by hand. The HIPS wheel covers were cut to a very rough shape, and glued to the MDF using a two part epoxy, similar to Araldite. The wheel covers were screwed onto the MDF to ensure they did not move during the drying of the glue, and to permanently add a mechanical join between the two parts. The radiuses were inserted between the MDF and the HIPS using multipurpose car bog. This was sanded to achieve a smooth uniform finish over the assembly and to eliminate all joining lines.

The MDF was sealed using Shellac to stop the primer seeping into the wood. After the Shellac coat dried, the fairing was painted with an alcohol based automotive primer. After three coats of primer, the parts of the fairing where the racing stripes would be were painted with three coats of orange. After the orange had dried, the stripes were masked using tape, and the brown colour was sprayed on. Before the brown paint had a chance to cure, the tape was removed. This ensured that the tape didn't remove large flakes of paint which would have happened if it was to be removed later on. The orange 'EDGAR' letters were attached, and four coats of clear lacquer were sprayed on to enhance the glossy finish of the fairing.

7.1.4 Post assembly

The dimensions of the upright pole were determined from the ergonomics and aesthetics as explained in Section 7.3. The upright pole and handle assembly pass through a hole in the fairing after the fairing has been attached, and slot into a sleeve on the steel platform where it is bolted into place. There are many advantages of using a non-permanent method of attaching the upright assembly. Primarily, it makes the process of maintenance much easier. A hole at the bottom of the upright pole was drilled to ensure that the wires for the LEDs and steering potentiometer could exit the pole. There are three holes drilled into the top of the handle for the three indicator LEDs to sit in their bezels.

The height adjustment mechanism of the pole was one section that presented a few problems. A nylon bush was inserted between the two poles to ensure they slid smoothly with respect to one another and did not scratch the outside of the smaller pole. The sliding action achieved between the poles was

very smooth and successful until it was realised that the poles did not clamp tight enough. This had an adverse effect when a rider was turning corners sharply while on EDGAR. The rider would shift their weight to one side of the handle, and the upper section of the pole and handle assembly would twist in the lower section. A number of ideas were suggested, but it was decided that the upper pole would be roughened with course sandpaper to keep it from twisting. This solved the problem to a sufficient degree. The grain of the abrasion traveled along the length of the tube, not radially. This meant the pole would still slide in and out smoothly, but it would not twist back and forward like it had previously. A photograph showing the completed height adjustment mechanism is shown in Figure 7.9.

Figure 7.9: Photograph of the height adjustment mechanism

7.1.5 Structural Considerations of other Components

There were other structural considerations that applied to different components including the IMU, power distribution board, MiniDRAGON+ Development Board, motor controller and batteries.

7.1.5.1 IMU

The IMU had to be very securely attached to the structure, as any movement of the IMU with respect to the structure would introduce severe disturbances into the control system. It was determined that the IMU would fit well into the upper left section of the aluminium box attached to the base plate. A block had to be made to raise the mounting position, so the base of the IMU was connected to the steel plate, and not the base of the aluminium box. The IMU was mounted on a block of timber using wood screws, and then the whole IMU and mounting block assembly was bolted on from underneath, using nuts that were pressed into the timber base.

Because the IMU was in a difficult place to reach, being able to attach it to the structure from underneath the base plate saved a lot of time. The other arrangement that was considered for mounting the IMU was to mount it underneath the overhanging section of the aluminium box. This was decided against as this area was a larger space that could be more economically used for larger components (such as the power distribution board or the MiniDRAGON+ Development Board). The IMU mounting assembly can be seen in Figure 7.10 and can be seen in place in Figure 7.11.

7.1.5.2 Power Distribution Board

The power distribution board was mounted via two screws from the front of the aluminium box, as shown in Figure 7.12, which kept it insulated from beneath, and also gave enough room above it to make it fairly simple to attach all relevant connectors and power cables, as shown in Figure 7.13.

7.1.5.3 MiniDRAGON+ Development Board

The microcontroller was mounted on the opposite side to the power distribution board, above the IMU. The microcontroller was turned upside down, and

Figure 7.10: Photograph of the IMU mounting assembly

attached to the aluminium box using three countersunk screws, as shown in Figure 7.14.

This was the simplest mounting solution, but did create a few problems when trying to reattach cables when the microcontroller was attached to EDGAR.

7.1.5.4 Motor Controller

From the first initial designs, the motor controller has been designed to sit upside-down attached to the underneath of the main structural plate. As no issues arose that prohibited the motor controller from being mounted in that particular orientation, so it remained there through to completion, as can be seen in Figure 7.15.

The motor controller cables were passed from the underside of the plate to the top side through the large hole that was cut for the motors to sit in.

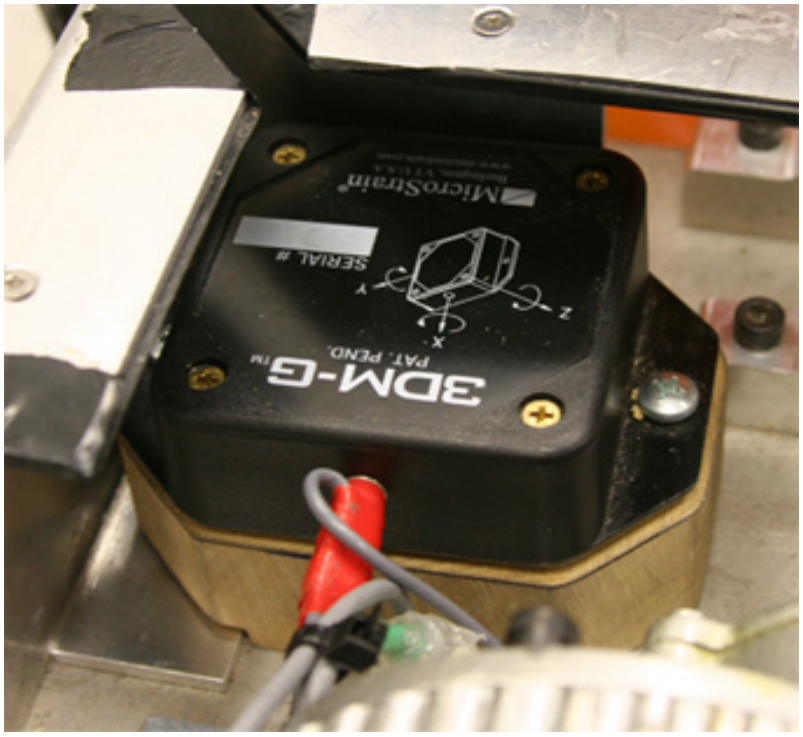


Figure 7.11: Photograph of the IMU mounted on EDGAR

7.1.5.5 Batteries

Throughout the year, there was significant discussion on how the batteries were going to be mounted in EDGAR. The mounting of the batteries seemed like a fairly difficult task, mainly due to the irregular shape and large weight of the batteries, but also due to the fact that they had to be very secure but also easily removable. Approaching the end of the project, it was realised that as EDGAR had only one set of batteries, and they could be charged in situ, it was not necessary to come up with a way of easily removing the batteries and changing them over. This proved very useful as a simple and easy solution was promptly reached. The batteries could have been mounted underneath EDGAR next to the motor controller, but since there was a large amount of space towards the rear of EDGAR behind the drivetrain, it was decided that this would be the most appropriate place to mount them. Placing the batteries on the top instead of the bottom of the plate meant they were more accessible, and had more airflow in case they were to get hot during charging and discharge. The batteries were grouped together in series in four banks

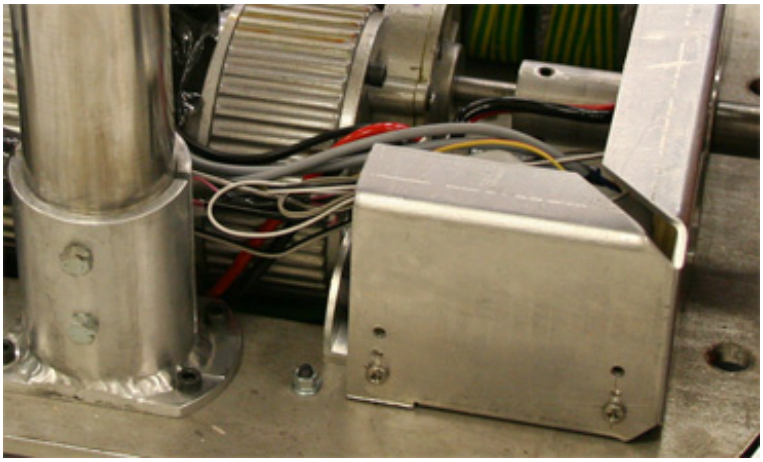


Figure 7.12: Photograph showing the mounting location of the power distribution board

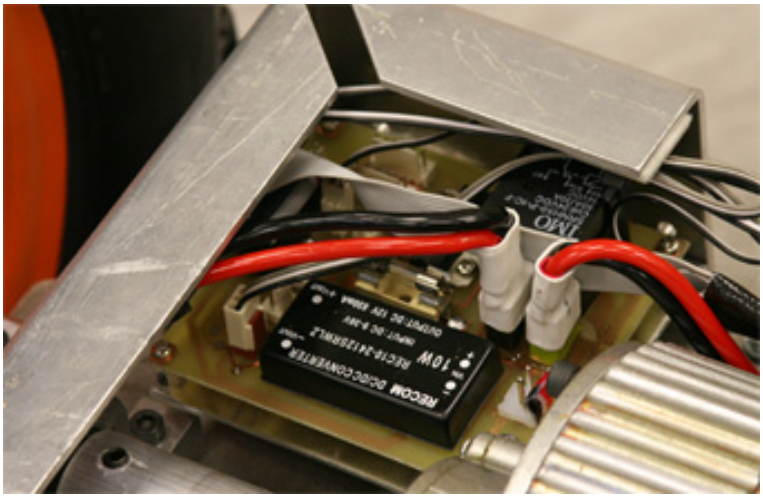


Figure 7.13: Photograph of the power distribution board

of five batteries and then heat-shrinked together. The battery assembly was cable tied to the inside back of the aluminium box. This proved to be a very simple and also rather successful solution to the problem. The mounting of the batteries is shown in Figure 7.16.

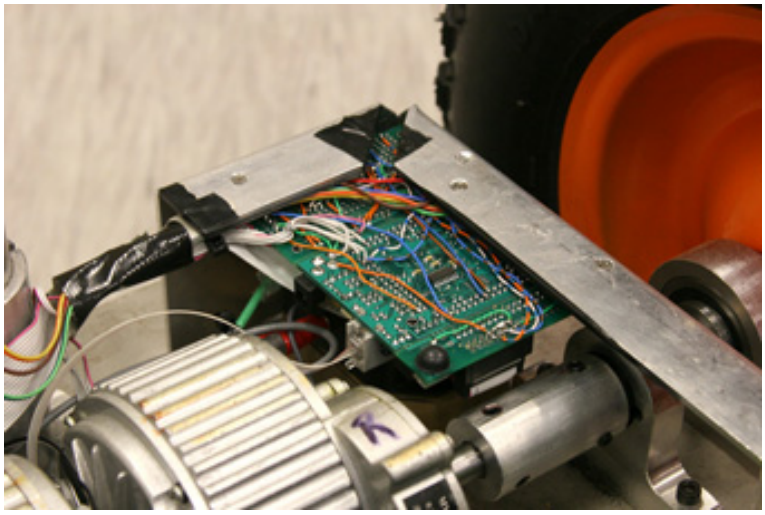


Figure 7.14: Photograph showing position of MiniDRAGON+

Figure 7.15: Photograph showing the motor controller mounted on the main structural plate

7.2 CAD Modelling Of Design

The Computer Aided Design (CAD) of EDGAR had four main purposes; to produce engineering drawings that the workshop could use to fabricate components for EDGAR. Secondly, to confirm of the integrity of structural components. The third reason was to make sure that the designs were aesthetically pleasing, and that the individual styles of the components complement

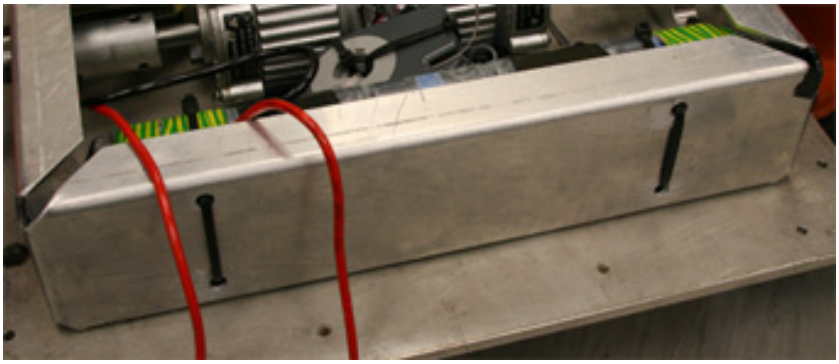


Figure 7.16: Photograph showing the location of the batteries

each other in the whole assembly of EDGAR. Finally, it was used to render photo-realistic images and videos that aided the seminar and exhibition. The third and fourth reasons are atypical of a final year engineering project.

EDGAR was primarily an educational engineering experience for the group members. The fact remains though, that a large proportion of people who have experienced (and are yet to experience) EDGAR are not going to have solid engineering backgrounds. As this is the case, it prompted the group into making EDGAR a project that not only the engineering community can assimilate with, but also a project that the friends and family of the group can appreciate.

There are many CAD packages available for use on the market today, after much deliberation it was decided that SolidWorks would be used as the CAD package for EDGAR. SolidWorks is a completely parametric program. This is of much use when making parts that share relationships with other parts. The relations between parts can be reasonably complex or the relationships can be relatively simple using Boolean statements. Using these tools, one can create smart models. In the initial stages of design, where dimensions are changing often, these smart models can save a great deal of time.

There are many add-ons to the SolidWorks package. Two of the most useful ones are CosmosWorks and PhotoWorks. These are an FEA package and photorealistic image and movie rendering package respectively. CosmosWorks easily validated the design analysis of EDGAR, and gave assurance that the designs are well within safety limits. Using CosmosWorks, the design of important structural members was re-engineered to make them safer, or produced more economically and reduce weight if over engineered. The realistic render-

ing engine, PhotoWorks, was also important to the project.

Throughout the entire project the aforementioned software packages were used to produce a 3D solid model of EDGAR. An assembly view of the final model with fairing is shown in Figure 7.17. A view of the model without the fairing showing internal components is shown in Figure 7.18.

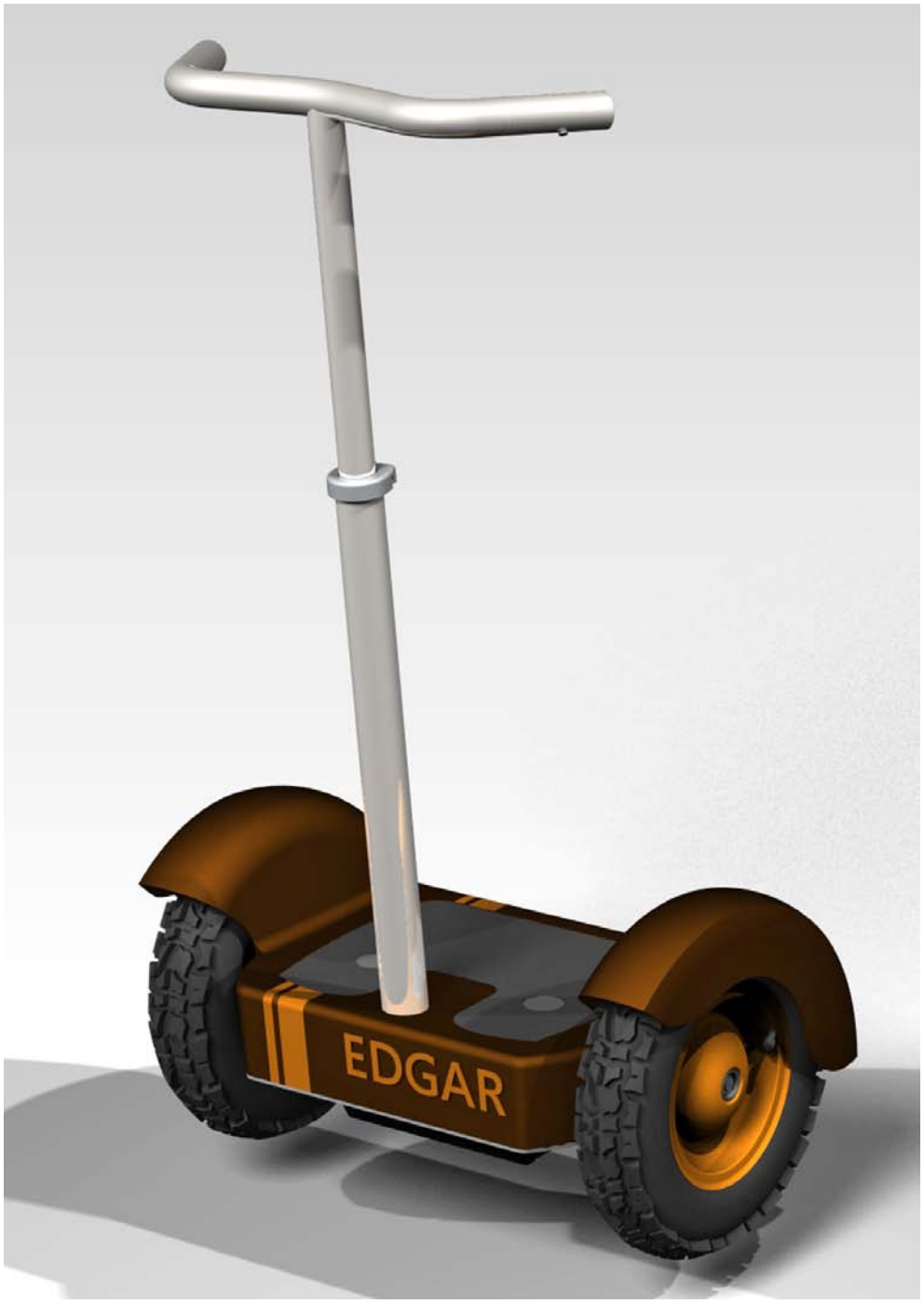


Figure 7.17: SolidWorks assembly of EDGAR

Figure 7.18: SolidWorks assembly of EDGAR without fairing

7.3 Ergonomics

A product's usability and acceptance are often dependent on the user feeling that it is easy to learn and use. Consequently, the ergonomics of EDGAR and interface between EDGAR and the user were details that were decided to be of reasonable importance. One of the main concerns the author of the Blackwell model (refer to Section 2.2) had when testing the scooter, was the effort in which the rider had to put into riding it, as quoted below. This is unacceptable since one of the main attractions of a self-balancing scooter is its ability to save time and effort.

> "It's fairly tiring to ride. Standing still on a hard, bouncing platform makes my feet tired. Not as bad as rollerblading, but a somewhat similar feeling. The body is really evolved to be in constant motion, and the combination of static posture (even more static than standing normally, since you try to keep your weight centred) and being jolted by bumps is probably bad for your spine." (Blackwell 2005).

Also, the look and control of the Blackwell model is conducive to thinking that the machine was built solely considering functionality and ease of manufacture, without paying regard to the usability or ergonomics of the device as shown in Figure 2.22. The ergonomics, in regard to a comfortable operating position and ease of use of the steering mechanism and other controls, have been considered in sufficient depth for EDGAR.

There were some dimensions and attributes that were considered when designing EDGAR. These included the positions and angles of limbs, distribution of weight, riding positions, and also the degrees of human joint movement.

7.3.1 Height Measurements

To find a reasonable standing position, the group carried out its first experiment which involved individual members of the group standing on a hand cart. This gave some indication of the angles that were comfortable to stand and which height would be suitable for a handle.

The second experiment was carried out at Glenunga International High School (GIHS) where a prototype shell was manufactured. With this prototype the appropriate height for the handle was measured more accurately. The

height was found to be suitable at a distance of 1000mm above the platform for all members of the group. Since all the group members are taller than the average build, this measurement was unsuitable if shorter people were to ride EDGAR. It was decided after taking a short informal survey of the students and lecturers at the university that the maximum height of the upright would not need to be taller than 1150mm. The upright, at its shortest position was decided to be 800mm.

The handle fabricated at GIHS was produced with a reasonable amount of guesswork. Upon the first construction, the handle was much wider and further away from the body than was comfortable. It was decided on the second iteration of the handle, that a width of 400mm from the inside of the user's hands was a comfortable distance for a broad cross section of people. Unlike the height of the upright, this distance could not be adjusted per user. Optimum angles and curves were found experimentally by printing 1:1 scale drawings from CAD onto heavy card and cutting them out. There were approximately five iterations until one was uniformly agreed upon. It was decided that an upright angle of 7 degrees from the horizontal towards the user would be appropriate. At this angle, the handle was not so close that it felt constricting, and not so far away that it felt unsafe.

7.3.2 Footprint and Platform

The dimensions of the base and footprint of EDGAR were derived from aesthetic, ergonomic and functional considerations. As per the specifications, EDGAR had to fit through a doorway; this limited the maximum width of the footprint to 820mm. A comfortable platform size was found by laying different sizes of paper on the ground and having group members stand on top of them while shifting their weight around. After a while, the individual would come to a comfortable position, while feeling secure and stable. This gave a good indication of how much room was needed on the platform for comfort, and to feel stable enough while traveling at reasonable speeds through tight corners and manoeuvres.

It was decided that dimensions of 450mm wide (not including wheels) and 300mm deep would give enough room to shuffle about and feel comfortable while riding EDGAR. After the first generation of CAD was produced, the platform looked too skinny and disproportional when the wheels were added

(the wheels increased the overall width to 700mm) so an extra 100mm was added to the depth of the platform.

To reduce shock and vibration experienced by the user while riding EDGAR, it was decided that the platform should have a rubber surface to increase grip and comfort. A suitable rubber doormat was found and cut to size. The edges of the platform were raised so the rubber mat sat flush against the rest of the platform. Similarly, the capacitive foot sensors were mounted flush with the rubber as well. This would ensure the foot sensors were neither mounted too high so that they protruded into the riders foot, or were mounted too low, that the rider felt a hole in the platform surface. This is illustrated in Figure 7.19.

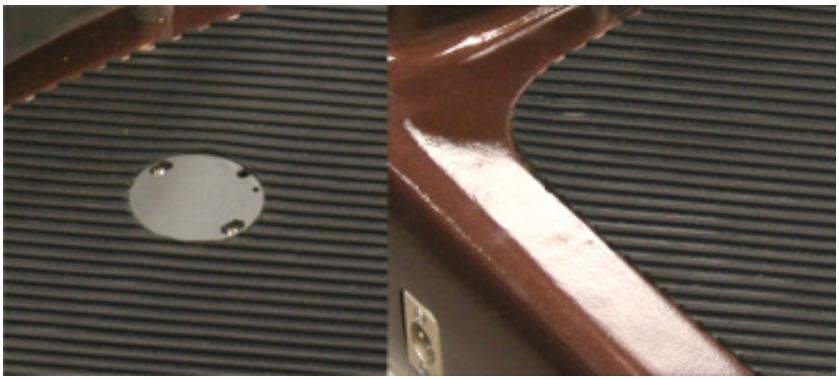


Figure 7.19: Photographs of the rubber mat showing a capacitive sensor

7.3.3 Steering and Other Controls

To be able to adequately control and feel comfortable using EDGAR, the steering mechanism had to have a simple and intuitive way of changing the direction of EDGAR's motion. There were many options brainstormed to be used as the turning mechanism for EDGAR. Two of the less popular ideas were; a steering wheel mounted on the upright post, or a simple a knob on the post. These ideas were deemed to be inconvenient, as it would be impossible to hold on with two hands to the handlebar while steering. Another idea was to steer by the user moving their weight from side to side, while measuring the small changes in displacement of the sides of the platform. This idea was considered inappropriate as it would be unreasonable to expect a rider to not shift their weight around and inadvertently steer EDGAR.

There were two ideas that were more favoured; a twisting sleeve on the handle, (not unlike a motorbike throttle) and installing an axial pivot on the upright post. The latter idea would have the effect of the whole handlebar assembly twisting left and right to steer. It was originally decided that a pivoting post would be an appropriate method, but after the group used a Segway HT whilst visiting the 2005 Adelaide Motor Show, it was agreed that a throttle type steering mechanism would provide an intuitive way to turn EDGAR. The main benefit of this type of control is that the upright pole and handlebar can be completely solid, and affixed to the base. This helps users feel more secure when EDGAR turns at reasonable speeds. One negative point concerning the first chosen method of steering is as follows: because there is no suspension, and as the platform will not lean into corners, it is up to the rider to intuitively shift their weight appropriately, to be able to sustain a centre of gravity in between the wheels consequently keeping EDGAR upright and stable. If the whole handle assembly was to rotate left and right as the steering mechanism when users turned left and right at speed, it may mean that as they were to shift their weight, pulling the handle further around, it would increasing the turning rate, and possibly unbalancing (or at least unsettling the rider). A photograph of the complete steering assembly is shown in Figure 7.20.

The rest of the user interface of EDGAR will consist of an on/off switch, the two aforementioned foot sensors, and a circuit breaker. These, with the addition of a series of LEDs and the steering potentiometer will give the user both control and knowledge of the current state of EDGAR.

7.4 Aesthetics

The overall appearance and impression of EDGAR was not a trivial matter in the design and construction. Two main ideas were suggested as possibilities as styles of design; a ‘retro 60’s style’ and a modern, more contemporary style. The main components that communicate the chosen style include the upper pole and handle bar assembly, and also the fairing.

The contemplated styles were sketched in ideation, as shown in Figure 7.21, and continued as CAD models, as shown in Figure 7.22. It was decided that the retro style would be better suited to EDGAR’s utility and users. It was agreed that to make sure EDGAR’s appearance was to work overall, each

Figure 7.20: Photograph of assembled steering mechanism

visible component needed to share the same aesthetic.

It was decided that because of the time constraints of the project, using anything but a bent metal tube for the handle would not be viable. If there had been more time, a few other manufacturing options would have been explored. The most probable alternative was to shape it from blue closed cell foam and strengthen it with a few layers of fibreglass. Manufacturing the handle this way would have meant a larger range of shapes were possible to achieve. It was decided that the most trouble free method to produce the handle would be as a metal tube, similar to the upright pole. The logic followed that using a modern, streamlined 'faux aerodynamic' look would not be successful. This was due to the fact that the overall appearance of EDGAR would not be coherent.

The fairing incorporates concentric circles, repeated curves and straight lines which result in a pleasing design. The retro look is achieved through both the pure, clean shape of the fairing and also choice of colour. The repeated curves are not something that will be obvious to an observer at first glance,

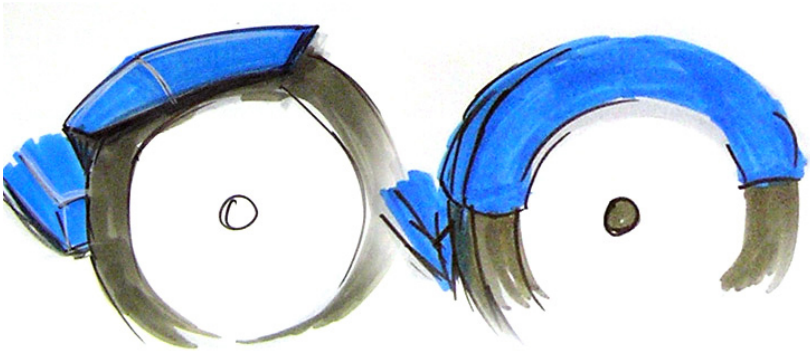


Figure 7.21: Sketch of fairing design

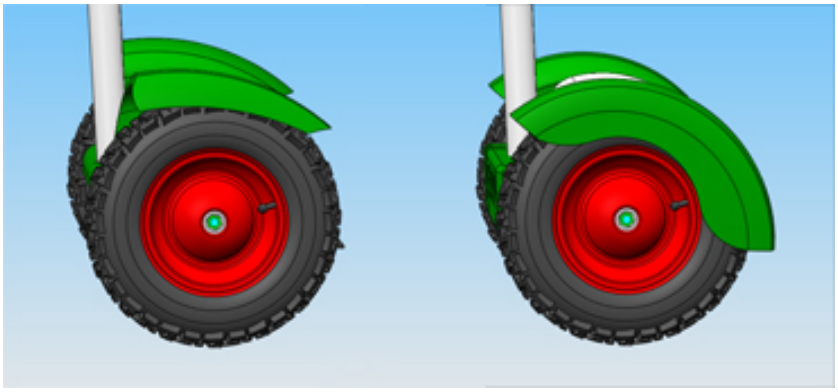


Figure 7.22: Initial SolidWorks rendering of two types of fairing designs

but subconsciously they produce a sense of harmony within the design. The curves of the cross section of the wheel covers are repeated in the cutout for the rubber footplate as seen in Figure 7.23. In Figure 7.24, it can be seen that the back face of the fairing is consistent with the curve of the lower section of the wheel covers. The front face of the fairing is at a five degree draft. This is at the same angle as the upright pole, as shown in Figure 7.25 and further solidifies the overall appearance of EDGAR.

The handle was made from 28mm outside diameter tube aluminium, while the upright pole was manufactured from both 32mm and 48mm tube aluminium. These dimensions were chosen carefully, such that they do not look too skinny and out of proportion with the rest of EDGAR. They are also not too thick, such that they would make EDGAR look top heavy. The upright pole has been designed to be very simplistic and clean. Apart from the small

Figure 7.23: Photograph of wheel cover curve repeated in foot mat

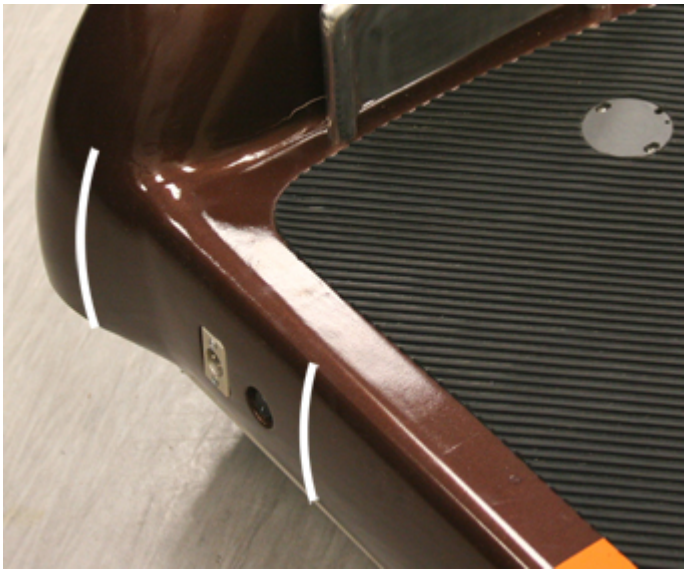


Figure 7.24: Photograph of wheel cover curve repeated in fairing

unobtrusive grub screws that comprise the turning mechanism, the only visible distractions to the lines of the handles are three LEDs and a bolt that forms part of the turning mechanism. The turning twist grip is completely flush with the main section of the handle to promote the clean curves of the handle.

The colours that were chosen for EDGAR were 'Satellite Brown' and 'Venus Orange.' These further add to the retro aesthetic of EDGAR, as these

Figure 7.25: Photograph of angle of upright post repeated in fairing

colours are respective of retro 'Puma' clothing, footwear and the like, which are becoming very popular these days. The speed-stripes that run front to back around EDGAR further emphasise this chosen aesthetic.

The fairing of EDGAR is one of the main components that communicate the aesthetic. Currently, with the large wheels, 'knobbly' tyres and retro fairing, EDGAR appears to be suited to a large range of terrains. EDGAR would not look out of place indoors, but is also appears capable of taking to off-road dirt tracks or grass.

Kick guards were installed to stop feet coming into regular contact with the wheel covers and fairing. The kick guards possess an aesthetic component as well as a functional one though, they will add to the retro look of EDGAR, as they incorporate sweeping curves and are also a simple design. A number of designs were considered for the kick guards in the end, a design was settled on which consists of 12mm aluminium bent in a upside down 'U' shape (with one corner), with 3mm plate aluminium welded in between the verticals of the kick guard. These were polished to match the upright tube and handle.

From all the positive feedback that has been given about the looks of EDGAR, it seems that the group has been successful in producing a coherent, deliberate retro aesthetic.

Chapter 8

Implementation and Testing

This chapter aims to convey some of the solutions to difficulties faced whilst testing EDGAR. Both hardware and software related problems were encountered, and each were individually overcome with research and persistent troubleshooting.

8.1 Software Based Problems

8.1.1 Simulink to C

As previously mentioned in Chapter 6, the real-time target developed by Dr Frank Wornle was used to streamline moving the control system from Simulink to code for the MiniDRAGON+. Unfortunately this was not as simple as first thought, as some of the blocks initially used in Simulink were not available in the real-time target toolbox. The model then had to be recreated using compatible blocks. More serious problems occurred when the model was expanded to include steering logic from the potentiometer mounted in the handlebars. This input came through the analog to digital (A/D) converter built into the microcontroller, which worked correctly when tested isolated from the rest of the control system. When linked to the main control system to add and subtract a percentage of the control signal from each motor, the program stopped functioning correctly. After consultation with Dr Wornle, it was found that the program was exiting with a serial input buffer overrun error. After comparing the generated code for the models that worked previously and the code that started to crash, it was found that the order that the Simulink blocks were run in the compiled code had changed significantly. Instead of the serial

transmit blocks being run first, they were being left to the end of each sample period. It was believed that this resulted in the serial receive blocks attempting to read from an empty buffer and producing a buffer overflow error. This problem was overcome by prioritising the block compilation order from within Simulink, through block properties. By forcing the transmit blocks to be run first and then attempting to delay the running of the receive blocks for as long as possible meant that the IMU, which was attached to the serial port, had time to receive the transmitted command byte and send its response in time. Once a delay was included the program ran without further timing problems.

8.1.2 PWM Generation

One of the first problems experienced relating to the software and hardware integration was when attempting to drive the motors through the motor controller using a PWM signal. It was found when a PWM signal was used, generated from either a Simulink block model or a signal generator, the motors twitched violently. Using various techniques, including using different PWM sources, capacitors as lowpass filters and changing the sampling frequency of the signal did not resolve the problem. It was decided to then use the analog input for the motor controller.

Testing with the analog signals proved successful and simple to achieve through dSPACE in the initial implementation of the software. However, when moving the code to the MiniDRAGON+, it was found that there was no onboard digital to analog (D/A) converters and these had to be made and connected to the I^2C bus of the microcontroller. While these were being built, another attempt at controlling the motors with PWM was made. This was marginally successful when using the 8-bit PWM signal generators but would only work with one channel when using the 16-bit generators. The 8-bit resolution proved to be inadequate with the format of the signal that was required. The R/C format only uses a small percentage of the sample time as the actual control signal, refer to Section 6.2 for more detail pertaining the generation of the required PWM signal. Again PWM was avoided and the analog format was firmly decided upon. Once the D/A converters were mounted, the control of the motors proved easy and suitable. Unfortunately, when there was trouble getting the input signals from the capacitive sensors into the MiniDRAGON+, detailed further below, it is believed that an oscilloscope was incorrectly grounded on one of the D/A units, burning out either

the D/A itself or the I^2C bus on the microcontroller. From this point on there was nothing that could be done to get any signal from the D/A units to the motor controller.

Again PWM signals were investigated. Dr Wornle was consulted and the problem with the 16-bit generators that restricted the output to only a single channel was resolved. It was also found that by using Servo PWM generators rather than standard PWM generators in Simulink, higher resolution could be used in the small control section of the signal. Finally, a smooth response was achieved from the motor controller using the PWM signals. It was believed after conferring with the Instrumentation Workshop that the original twitching was due to the motor controller and PWM generators not having a common ground. Once the MiniDRAGON+ and motor controller were attached to the on-board power supply with a common ground, no more problems were encountered with the PWM signal.

8.1.3 IMU Initialisation Time

When power to all components was supplied from the on-board batteries the program initially behaved in a very unusual way. When code was downloaded onto the MiniDRAGON+'s memory and then executed it worked correctly, but if power was cycled, the program would not behave as expected. The motors would respond correctly to steering signals but not to pitch changes of EDGAR. The program would then return to normal behaviour if the microcontroller reset button was used without power being cycled. An attempt to come up with an easy way to reset the microcontroller shortly after the power was initially cycled did not solve the problem. It was suggested that because all components were getting power at the same time, the IMU was not initialising correctly before a serial communication request was made. Using an external power source for the IMU the program operated correctly, even when power to all other components was cycled using the main switch. It was then decided that a pause should be included at the beginning of the compiled code to delay the main program from starting, allowing the IMU time to initialise. This delay was implemented by flashing the handle bar LED's in sequence, with the rider instructed only to mount EDGAR when a solid green LED indicated that the program had started. The additional code that was added to perform this delay is shown in Figure 8.1.

```
/* Start for root system: '<Root>' */
void MdlStart(void)
{
  /* Initialize digital outputs for port PORTA */
  DDRA |= 7;

  /* Blink LEDs on handle bars before starting main program */
  PORTA |= 0x01;
  blinky(100000);
  PORTA |= 0x04;
  blinky(100000);
  PORTA |= 0x02;
  blinky(100000);
  PORTA &= 0xFE;
  blinky(100000);
  PORTA &= 0xFB;
  blinky(100000);

  /* DiscretePulseGenerator Block: <Root>/CMD Generator */
  {
    int_T Ns;
    real_T tFirst = rtmGetTStart(rtM_md_Final);
    Ns = (int_T)floor(tFirst / 0.005 + 0.5);
    if (Ns <= 0) {
      rtDWork.CMD_Generator_IWORK.ClockTicksCounter = Ns;
```

Figure 8.1: Segment of the C-code that was modified to add a startup delay

8.1.4 Capacitive Sensor Problems

Complications arose when integrating the capacitive sensors into the program. The specifications of the sensors required a supply voltage of at least 12V, which as a result meant that the signal they produced would also be 12V. The digital inputs of the MiniDRAGON+ were TTL compliant, which mean that they normally operated at 5V high and 0V low. The problem was solved by using a voltage divider to bring the sensor signals down to approximately 5V which could then be used as a digital input. A voltage divider comprising of two resistors in series connects the sensor to ground. The input signal is taken from between the resistors. The formula for the resulting voltage is given by the following:

$$V_{out} = \frac{R_1}{R_1 + R_2} V_{in} \tag{8.1}$$

The choice of resistor value needed to be high enough so that the current draw would be minimal, but not too high as that may produce a noisy signal. The choice of resistors of $4.7k\Omega$ and $5.6k\Omega$ proved to be adequate. A similar voltage divider, though this time using different resistor values, was used to convert the battery voltage, of up to a possible 26V, to approximately 0-5V analog input. Resistors of value $10k\Omega$ and $51k\Omega$ were used to give a relationship of:

$$V_{out} = \frac{1}{6.8} V_{in} \tag{8.2}$$

8.1.5 Damage to Second Serial Port

During tuning of the control system on EDGAR one of the serial ports was damaged. A loose foot managed to short out the second communications port (SCI1) that the IMU was connected to, resulting in that port being inoperable. The MiniDRAGON+ has two Serial Communication Interfaces (SCI) and SCI0 was only being used for downloading program code. The IMU could be connected to SCI0 once the program was loaded into memory and the programming cable removed. A new cable was made to connect the IMU to SCI0 and the problem fixed within a couple of hours. Testing was then able to resume, though with measures taken to reduce the risk of physically damaging the MiniDRAGON+ again.

8.2 Hardware Based Problems

During the construction and testing of EDGAR, a number of changes were made to the designs as assembly and operational problems were identified. These included flexibility in the main structural plate, backlash in the drive system, and changes to the design of the steering mechanism.

8.2.1 Flexibility of Main Structural Plate

The main structural plate was initially laser cut from 5mm plate aluminium. Upon assembling EDGAR for the first time, it was found that the particular directions that forces act on the main structural platform produced two main flexural behaviours; whole plate bending due to the rider's weight and vertical movement of the motors due to forces on the upright post.

The weight of the rider being transferred through the bearing housings resulted in the entire plate flexing. The location of the bearing housings on the main structural plate was in line with a number of holes that were used to attach other components. This produced a line of weakness in the plate which flexed when a rider stood on EDGAR. This movement was partially plastic, and as a result the plate was permanently bent during testing. This movement is illustrated through the first set of arrows in Figure 8.2.

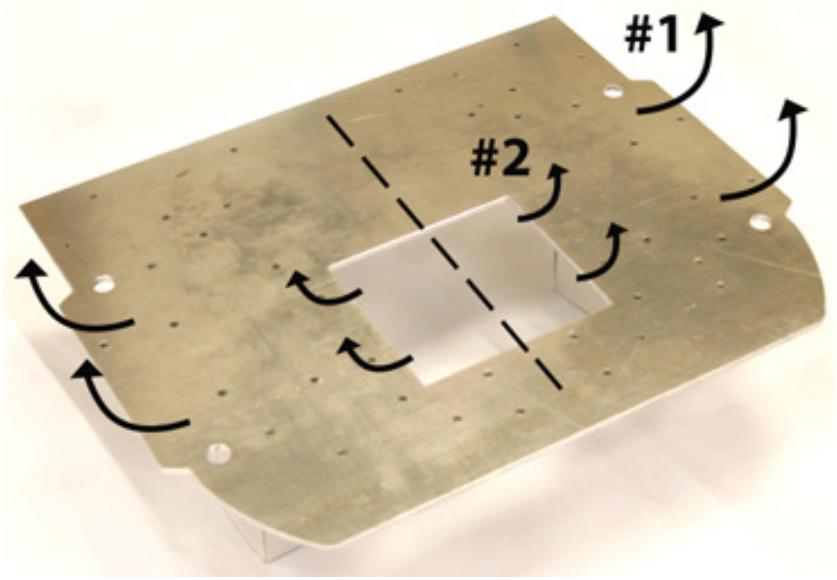


Figure 8.2: Photograph showing flexion in main structural plate

Forces originating from the rider holding onto the upright post produced vertical movement of the motors with respect to other drive system components. As the rider shifted weight from one side of the handlebars to the other, the main structural plate flexed where the post was attached to the base plate. This movement did not result in plastic deformation of the structural plate. This movement is illustrated through the second set of arrows in Figure 8.2.

A second plate was manufactured in the same fashion as the first but from 8mm plate stainless steel. This plate proved much more rigid than the first and no plastic deformation or movement in the motors has been observed.

8.2.2 Backlash in Drive System

As discussed in Section 5.6, flexible couplings were used initially to connect the motor shafts to the main axles. The very small amount of backlash that existed in the jaw-type flexible couplings chosen resulted in a mechanically introduced deadzone whenever the motors applied a torque to the wheels. This meant that when the control system tried to speed up or slow down the motors, the change in power to the motors was not apparent at the wheels until the deadzone had been overcome. Consequently backlash was felt by the user whenever the motors changed directions. After riding EDGAR during initial testing, it was decided to remove the flexible couplings and replace them with rigid couplings manufactured by the Mechanical Workshop. The rigid couplings were fixed to the motor shafts and main axles by 6 grub screws. Shallow holes were drilled into the axles to positively locate the grub screws. Upon installation of rigid couplings, the level of backlash was reduced significantly. Minimal backlash exists only due to the gear meshing in the motors.

8.2.3 Steering Mechanism Design Changes

Whilst manufacturing the steering mechanism it was noted that more friction existed between the metal on metal interfaces than was expected. As the original design for the steering mechanism relied on the return force of the spring being transferred through the potentiometer casing, a new design was produced that eliminated the potentiometer from bearing the force of the spring. This allowed the gauge of the spring to be increased, resulting in a stronger return force which overcame the friction in the steering mechanism. The second design also allowed the return point of the mechanism to be ad-

justed during assembly whereas the first design permanently fixed the return point during construction which ran the risk of manufacturing an off-centre steering mechanism.

The final steering mechanism works to self-centre the handle grip after twisting whilst measuring the instantaneous position of the potentiometer as required.

8.3 Modifications to Controller During Testing

To assist with fine tuning the controller while riding EDGAR, a set of push-buttons was used to alter the proportional and derivative gains. In order to implement this, the input from the buttons needed to be registers on the digital input ports of the MiniDRAGON+. This signal would then trigger an addition or subtraction of a set amount from the current gain value. The 7-segment LED display on the MiniDRAGON+ was used to indicate the number of times the buttons had been pressed. By knowing the increments being used, it was then easy to know what the current gains were. The Simulink block used to generate the code for this testing procedure is in Figure 8.3 below.

While riding EDGAR it was found that while attempting to stay still, there was a tendency to keep wanting to lean backwards. It was believed that this was due to the uneven weight distribution within the base. Without power applied to the system, EDGAR was weighted heavily backwards, and even when a rider was present, the centre of gravity when the base was horizontal was not directly vertical from the axle. By changing the angle setpoint from zero to a small forwards angle, this resulted in the centre of gravity at the 'upright' position to now be much closer to vertical. This improved the ride-ability of EDGAR dramatically, though was a bit disconcerting to have to lean forwards a bit to stay still. A redistribution of the weight in the base would help stability of the system. This setpoint change is shown in Figure 8.3. It must be noted however that the setpoint is negative because the angle that is read from the IMU is actually a negative value. This was a minor issue that came about when the IMU was mounted into the base for the first time; it would only fit in one way and that measured the forward tilt with negative angles.

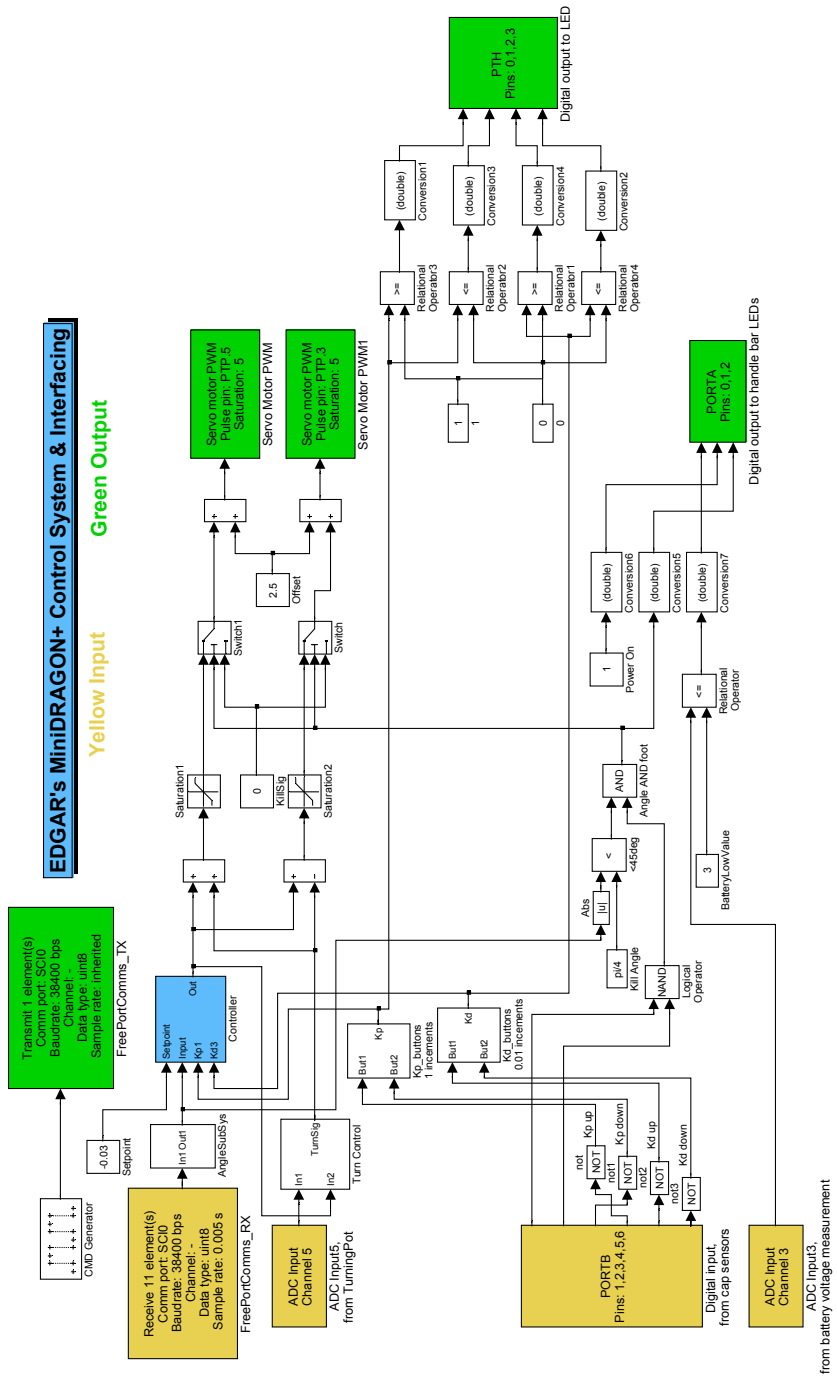


Figure 8.3: Simulink model of program used during testing

Chapter 9

Final Design Analysis

This chapter analyses the outcomes of the project. This includes summarising the extent to which the primary and extension goals of the project were satisfied and providing an assessment of the overall cost of EDGAR.

9.1 Satisfaction of Project Goals and Basic Specifications

As aforementioned in Chapter 3, the goals of this project were as follows:

- To develop an accurate mathematical model and control system for EDGAR
- To reproduce and analyse the model in MATLAB and Simulink
- To develop a Virtual Reality (VR) model of the prototype for the tuning of the control system
- To design and build a physical prototype of EDGAR
- To run the prototype with classical Proportional Derivative (PD) control tethered to a computer using dSPACE equipment

All five primary project goals have been satisfied during the development of EDGAR. A theoretical model of the system was developed early on during the project and enabled the generation of a Proportional Derivative (PD) controller for the system, as described in Chapter 4. This project goal was satisfied and is verified because EDGAR balances. The goal of a mathematical

model in Simulink was achieved which allowed simulation of the system and analysis of its performance as discussed in Chapter 6. Prior to the construction of EDGAR, a Virtual Reality (VR) model was developed. This enabled the mathematical model and PD controller to be tested before the physical model was ready for use. This goal was successfully achieved. The physical model was designed and built incorporating adequate safety measures. The success of this goal was verified as EDGAR could support and balance a person.

A number of extension goals were also defined with the intention to attempt them if time permitted:

- To remove the need for tethering the physical prototype of EDGAR by putting all processing and power onboard,
- To determine an accurate State Space (SS) model and controller for EDGAR,
- To run EDGAR tethered with SS control using dSPACE,
- To run EDGAR untethered with SS control implemented on-board.

One of the four extension goals has been met at the conclusion of this project. Following the testing of EDGAR tethered to a computer using dSPACE and Simulink software, work was completed in transferring the control system onto a MiniDRAGON+ for testing EDGAR untethered. Some difficulty was encountered during the cross-compilation stage from Simulink to C, however these problems were resolved and the PD control system has been successfully implemented onboard EDGAR. In addition to implementing an on-board controller, an on-board power system was installed comprising rechargeable batteries, see Section 5.8, and power distribution circuitry, see Section 5.14, which allowed EDGAR to run independent of tethered power and control. This extension goal has been completed successfully.

The three remaining extension goals all relate to the implementation and testing of SS instead of PD control of EDGAR. This was not attempted by the group.

EDGAR was designed and constructed with a set of basic specifications in mind. These are discussed in Section 3.3 and outline the main features

and behaviour of EDGAR that were desired. Almost all of the specifications listed have been satisfied in the final model of EDGAR. The only specification which has not been entirely met is that EDGAR should fit through a doorway. EDGAR by itself fits through a doorway, but a rider atop EDGAR is liable to knock their head on the doorframe.

9.2 Cost Analysis

This section provides a summary of the costs associated with EDGAR. The table below shows a list of the costs of major components.

ITEM	**COST**
Oatley DC Motor (2x)	$160
Wheels & Tyres (2x)	$122
MiniDRAGON+ Microcontroller Board	$100
Bearings (4x)	$100
Batteries	$280
Turning Mechanism	$50
Structural Materials	$350
Fairing	$200
Power Distribution Board	$200
Rubber Foot Mat	$11
Battery Charger	$118
Miscellaneous	$100
TOTAL	**$1791**
Omitted	
MicroStrain IMU	*$2000*
Roboteq Motor Controller	*$850*

The two items in italics underneath the total have not been included in project budget but were made available by the department. These items could be replaced with lower cost alternatives if EDGAR was to be made again or work was done to reduce costs.

Labour costs of this project would have been significantly higher than the equipment cost. As four students have completed the project, there is no real

cost associated to their work. The labour costs from the Mechanical Engineering and Instrumentation Workshops at the University of Adelaide have, however, incurred real costs toward this project. The number of workshop hours spent working on this project has been estimated roughly at one full day of work in the Instrumentation Workshop and two to three days in the Mechanical Workshop.

ITEM	**HOURS**
Mechanical Workshop	24
Instrumentation Workshop	8
TOTAL	**32**

The higher labour cost than equipment cost is typical of development projects where work is being completed for the first time where most time is spent designing and troubleshooting issues that arise during construction and testing.

Chapter 10

Summary

10.1 Future Work

There are several changes that could be made to increase the functionality and performance of EDGAR. As this was the first version of EDGAR, it was considered infeasible to address these issues in the allowable time frame.

- The implementation of State Space control on EDGAR could significantly improve the robustness of the control system. This would require the introduction of new sensors such as encoders to measure appropriate states as required.
- Reducing the cost of EDGAR could be achieved by implementing solid-state rate gyroscopes instead of the Inertial Measurement Unit, using a Digital Signal Processor to implement the control system, and installing cheaper motor controllers (perhaps one for each channel).
- The motors used on EDGAR are incapable of handling large peak currents. Replacing these with more powerful motors would improve EDGAR's performance at higher speeds.
- It would also be beneficial to implement a bank of ultra-capacitors to help protect the batteries from the peak current draw of the motor controllers by providing a short supply of high current when needed.
- Dynamic variation of the control system based on tilt angle and current battery capacity would provide a safer and more consistent ride over the entire battery charge.

- Steering has been implemented open loop and the current control system does not measure the turning rate. Closed loop steering would improve EDGAR's ride quality.
- Implementing a wireless communication link between the MiniDRAGON+ and Simulink would enable tuning parameters to be changed whilst EDGAR is running and provide the ability to capture the control system's response for later analysis.
- Modifying the fairing such that the switches currently mounted in the rear of EDGAR be permanently attached to the main structure is recommended. This would eliminate the need to position multiple electrical cables each time the fairing is attached.

10.2 Conclusion

The aim of this project was to design and build a coaxial, rideable, self-balancing scooter. This aim has been achieved and an untethered scooter with on-board power and control has been manufactured and successfully tested.

A comprehensive literature review was conducted, covering technical information relevant to the project. The mathematical model of EDGAR was developed and then implemented along with a classical Proportional Derivative control system in Simulink. The Virtual Reality toolbox allowed the system to be simulated and the controller tuned prior to manufacture.

A formulated design approach was used to create the most efficient and robust configuration of EDGAR to satisfy all the project goals and one extension goal. The structural design was considered concurrently with component selection, aesthetics, and ergonomics to minimise mechanical, electrical and rider integration problems. The use of rapid control prototyping systems reduced software development time during EDGAR's tethered and untethered stages.

The outcome of this project has been a mechanically sound, aesthetically pleasing and easy to ride self-balancing scooter.

References

Beckwith, B., Desjardins, E., Howard, C., Murphy, J., Uganecz, M. & Woolley, J. (2004), *HTV Project Final Report*, Camosun College, Canada.

Blackwell, T. (2005), *How To Build A Self Balancing Scooter*, Last accessed October 2005.
URL: *http://tlb.org/scooter.html*

Cazzolato, B. S. (2005), *MECH ENG 5962 Advanced Automatic Control Course Notes*, The University of Adelaide, Australia.

Chudleigh, M., Clarke, D. & Lemire-Elmore, J. (2005), *Project Emanual: The Almost Self Balancing Two Wheeled Electric Skateboard*, Last accessed October 2005.
URL: *http://www.ebikes.ca/Projects/Emanual/Index.html*

Dorf, R. & Bishop, R. (2001), *Modern Control Systems*, Prentice Hall International, New Jersey, USA.

dSPACE Inc (2005), *ControlDesk 2.3 (Release 3.5) Software.*

FreeScale Semiconductor Co (2005), *Metrowerks CodeWarrior for Freescale HC12*, Last accessed October 2005.
URL: *http://www.metrowerks.com/MW/Develop/Embedded/HC12/Default.htm*

Grasser, F., D'Arrigo, A., Colombi, S. & Rufer, A. (2002), *JOE: A Mobile, Inverted Pendulum*, Vol. 49, IEEE Transactions on Industrial Electronics.

How Stuff Works Inc (2005), *How Segways Work Website*, Last accessed October 2005.
URL: *http://travel.howstuffworks.com/ginger.htm*

Larson, T. (2005), *Balancing Robot Project - "Bender"*, Last accessed October 2005.
URL: *http://www.tedlarson.com/robots/balancingbot.htm*

Mathworks Inc (2005), *MATLAB R14 7.0.1 Software*.

Segway Inc (2005), *Segway Website*, Last accessed October 2005.
URL: *http://www.segway.com*

Sophia University Japan (2005), *Inverted Pendulum Research*, Last accessed May 2005.
URL: *http://www.me.sophia.ac.jp/ctrl/ri/pendulum.html*

University of Newcastle (2005), *Inverted Pendulum Tutorial*, Last accessed May 2005.
URL: *http://www.eng.newcastle.edu.au/eecs/cdsc/books/csd/simulations/pend_sim.html*

Appendix A

Component Datasheets

AX2550/2850

Dual Channel Forward/Reverse Digital Robot Controller with Optical Encoder Inputs

up to 2 x 140 SmartAmps

for Computer Guided and Remote Controlled Robotic Vehicles

Roboteq's AX2550 controller and its 2850 variant are designed to convert commands received from a R/C radio, Analog Joystick, wireless modem, or microcomputer into high voltage and high current output for driving one or two DC motors. Designed for maximal ease-of-useby professionals and hobbyist alike, it is delivered with all necessary cables and hardware and is ready to use in minutes.

The controller's two channels can either be operated independently or mixed to set the direction and rotation of a vehicle by coordinating the motion on each side of the vehicle. The motors may be operated in open or closed loop speed mode. Using low-cost position sensors, they may also be set to operate as heavy-duty position servos.

The AX2850 version is equipped with quadrature optical encoders inputs for precision speed or position operation.

Numerous safety features are incorporated into the controller to ensure reliable and safe operation. A high efficiency version is also available for higher current operation in extended temperature environment.

The controller can be reprogrammed in the field with the latest features by downloading new operating software from Roboteq.

Applications

- Heavyweight, heavy duty robots
- Terrestrial and Underwater Robotic Vehicles
- Automatic Guided Vehicles
- Electric vehicles
- Police and Military Robots
- Hazardous Material Handling Robots
- Telepresence Systems

Key Features	Benefits
Dual MCU digital design	Accurate, reliable, and fully programmable operation. Advanced algorithms
R/C mode support	Connects directly to simple, low cost R/C radios
RS232 Serial mode support	Connects directly to computers for autonomous operation or to wireless modem for two-way remote control
Analog mode support	Connects directly to analog joystick
Optional Optical encoder in (AX2850)	Stable speed regardless of load. Accurate measurement of travelled distance
Built-in power drivers for two motors	Supports all common robot drive methods
Up to 140A output per channel	Gives robot strongest lifting or pushing power
Programmable current limitation	Protects controller, motors, wiring and battery.
Open loop or closed loop speed control	Low cost or higher accuracy speed control
Closed loop position control	Create low cost, ultra-high torque jumbo servos
Data Logging Output	Capture operating parameters in PC or PDA for analysis
Built-in DC/DC converter	Operates from a single 12V-40V battery
Extruded aluminum, heat sinking enclosure	Operates in the harshest shock and temperature environment
Field upgradable software	Never obsolete. Add features via the internet

Technical Features

Microcomputer-based Digital Design

- Multiple operating modes
- Fully programmable using either built-in switches and 7 segment LED display or through connection to a PC
- Non-volatile storage of user configurable settings. No jumpers needed
- Simple operation
- Software upgradable with new features

Multiple Command Modes

- Serial port (RS-232) input
- Radio-Control Pulse-Width input
- 0-5V Analog Voltage input

Multiple Motor Control modes

- Independent channel operation
- Mixed control (sum and difference) for tank-like steering
- Open Loop or Closed Loop Speed mode
- Position control mode for building high power position servos
- Modes can be set independently for each channel

Optical Encoder Inputs (AX2850)

- Two Quadrature Optical Encoders inputs
- 250kHz max. frequency per channel
- 32-bit up-down counters
- Inputs may be shared with four optional limit switches

Automatic Command Corrections

- Joystick min, max and center calibration
- Selectable deadband width
- Selectable exponentiation factors for each command inputs
- 3rd R/C channel input for accessory output activation

Special Function Inputs/Outputs

- 2 Analog inputs. Used as
  - Tachometer inputs for closed loop speed control
  - Potentiometer input for position (servo mode)
  - External temperature sensor inputs
  - User defined purpose (RS232 mode only)
- One Switch input configurable as
  - Emergency stop command
  - Reversing commands when running vehicle inverted
- Up to 2 general purpose outputs for accessories or weapon
  - One 24V, 2A output
  - One low-level digital output
- Up to 2 digital input signals

Built-in Sensors

- Voltage sensor for monitoring the main 12 to 40V battery
- Voltage monitoring of internal 12V
- Temperature sensors near each Power Transistor bridge

Advanced Data Logging Capabilities

- 12 internal parameters, including battery voltage, captured R/C command, temperature and Amps accessible via RS232 port
- Data may be logged in a PC or microcomputer
- Data Logging Software supplied for PC

Low Power Consumption

- On board DC/DC converter for single 12 to 40V battery system operation
- Optional 12V backup power input for powering safely the controller if the main motor batteries are discharged
- 200mA at 12V or 100mA at 24V idle current consumption
- Power Control wire for turning On or Off the controller from external microcomputer or switch
- No consumption by output stage when motors stopped
- Regulated 5V output for powering R/C radio. Eliminates the need for separate R/C battery.

High Efficiency Motor Power Outputs

- Two independent power output stages
- Dual H bridge for full forward/reverse operation
- Ultra-efficient 2.5 mOhm (1.25mOhm HE version) ON resistance MOSFETs
- Four quadrant operation. Supports regeneration
- 12 to 40 V operation
- User programmable current limit up to 140A depending on controller version and heatsink arrangement
- Standard Fast-on connectors for power supply and motors
- 16 kHz Pulse Width Modulation (PWM) output
- Aluminum heat sink. Optional conduction cooling plate

Advanced Safety Features

- Safe power on mode
- Optical isolation on R/C control inputs
- Automatic Power stage off in case of electrically or software induced program failure
- Overvoltage and Undervoltage protection
- Watchdog for automatic motor shutdown in case of command loss (R/C and RS232 modes)
- Large and bright run/failure diagnostics on 7 segment LED display
- Programmable motors acceleration
- Built-in controller overheat sensors
- "Dead-man" switch input
- Emergency Stop input signal and button

Compact Design

- All-in-one design. Built from aluminum heat sink extrusion with mount brackets
- Efficient heat sinking. Operates without a fan in most applications.
- 9" (228.5mm) L, 5.5" W (140mm), 1.8" (40mm) H
- -20o to +70o C (-40 to +85o C, HE version) operating environment
- 3 lbs (1,350g)

Ordering Information

Model	Description
AX2550	Dual Channel DC Motor controller up to 120 SmartAmps per channel
AX2550HE	Dual Channel High-Efficiency, Ext Temperature, DC Motor controller up to 140 SmartAmps per channel
AX2850	Dual Ch. Forward/Reverse DC Motor controller up 120 SmartAmps per ch. with Optical Encoder inputs
AX2850HE	Dual Ch., High-Efficiency, Ext.Temperature, DC motor controller up to 140 SmartAmps per ch. with Optical Encoder inputs

RoboteQ

www.roboteq.com

8180 E.Del Plomo Dr.
Scottsdale, AZ 85258 - USA
602-617-3931

3DM-GX1™

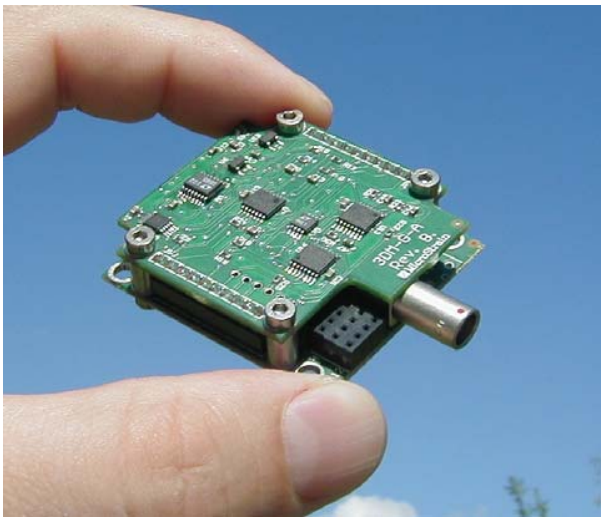
GYRO ENHANCED ORIENTATION SENSOR

3DM-GX1 combines three angular rate gyros with three orthogonal DC accelerometers, three orthogonal magnetometers, muliplexer, 16 bit A/D converter, and embedded microcontroller, to output its orientation in dynamic and static environments.

Operating over the full 360 degrees of angular motion on all three axes, 3DM-GX1 provides orientation in matrix, quaternion, and Euler formats. The digital serial output can also provide temperature compensated, calibrated data from all nine orthogonal sensors at update rates of 350 Hz.

APPLICATIONS

- unmanned aerial /underwater vehicles, robotics
  navigation, artificial horizon
- computer science, biomedical
  animation, linkage free tracking/control
- mobile cameras, sonar scanners
  image reconstruction
- mobile radio antennas
  aiming optimization, dynamic correction, antenna shaping
- manufacturing
  container handling, hydraulic lift systems, machine tools



Networks of 3DM-GX1 nodes can be deployed by using the built-in RS-485 network protocol. Embedded microcontrollers relieve the host system from the burden of orientation calculations, allowing deployment of dozens of 3DM-GX1 nodes with no significant decrease in system throughput.

Output modes and software filter parameters are user programmable. Programmed parameters and calibration data are stored in nonvolatile memory.

As with all MicroStrain products, every module is carefully tested prior to shipment, and calibration data are included with each order.

To place an order, or for more information, call us today at 800-449-3878.

BEST OF SENSORS EXPO Spring 2002 Gold

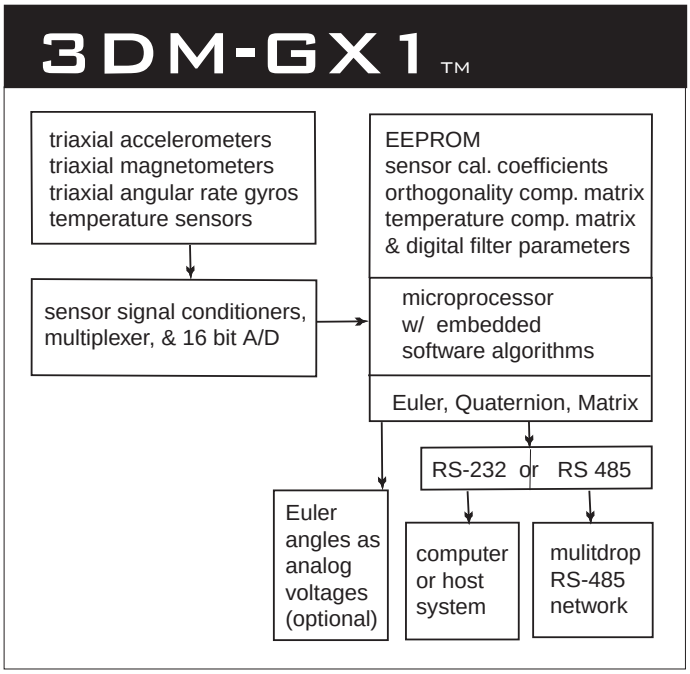
MicroStrain®

How it works

3DM-GX1 utilizes the triaxial gyros to track dynamic orientation and the triaxial DC accelerometers along with the triaxial magnetometers to track static orientation. The embedded microprocessor contains a unique programmable filter algorithm, which blends these static & dynamic responses in real-time.

This provides a fast response in the face of vibration and quick movements, while eliminating drift. The stabilized output is provided in an easy to use digital format. Analog output voltages proportional to the Euler angles can be ordered as an option.

Full temperature compensation is provided for all nine orthogonal sensors to insure performance over a wide operating temperature range.



Patents Pending

SPECIFICATIONS

Orientation Range	*360 deg. full scale (FS), all axes (Matrix, Quaternion modes)*
Sensor Range	*gyros: +/- 300 deg./sec FS* *accelerometers: +/- 5 G's FS* *magnetometers: +/- 1.2 Gauss FS*
A/D Resolution	*16 bits*
Accel. Nonlinearity	*0.2%*
Accel. Bias Stability*	*0.010 G's*
Gyro Nonlinearity	*0.2%*
Gyro Bias Stability*	*0.7 degrees/sec*
Magnetom. Nonlinearity	*0.4%*
Magnetom. Bias Stability*	*.010 Gauss*
Orientation Resolution	*< 0.1 degrees minimum*
Repeatability	*0.20 degrees*
Accuracy*	*+/- 0.5 degrees typical for static test conditions, ±2 degrees typical for dynamic (cyclic) test conditions & for arbitrary orientation angles*

** Accuracy & stability specs. obtained over operating temperatures of -40 to 70 degrees C with known sine and step inputs, including angular rates of +/- 300 degrees per second.*

Output Modes	*matrix, quaternion, Euler angles, & nine scaled sensors w/ temperature*
Digital Outputs	*serial RS-232 & RS-485 optional with software programming*
Analog Output Option	*0-5 volts full scale for Euler angles (pitch +/-90 , roll +/- 180, yaw 360 deg.)*
Digital Output Rates	*100 Hz for Euler, Matrix, Quaternion* *350 Hz for nine orthogonal sensors only*
Serial Data Rate	*19.2/38.4/115.2 kbaud, software prog.*
Supply Voltage	*5.2 VDC min., 12 VDC max.*
Supply Current	*65 milliamps*
Connectors	*one keyed LEMO, two for RS-485 option*
Operating Temp.	*- 40 to +70 deg C with enclosure* *-40 to +85 deg C w/o enclosure*
Enclosure (w/tabs)	*64 mm by 90 mm by 25 mm*
Weight	*75 gr. with encl., 30 gr. without*
Shock limit	*1000g (unpowered), 500g (powered)*

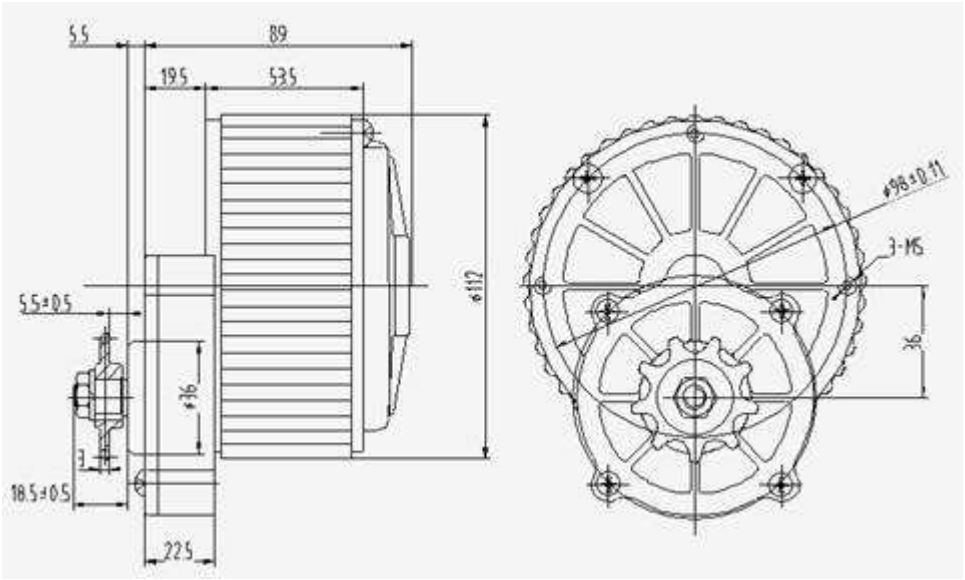
MicroStrain®

310 Hurricane Lane, Unit 4 Williston, VT 05495 USA
Phone: **800-449-3878** Fax : 802-863-4093
sales@microstrain.com www.microstrain.com

浙江尤奈特电机有限公司
UNITE ELECTRIC MOTOR CO.,LTD.

MY1018-250W

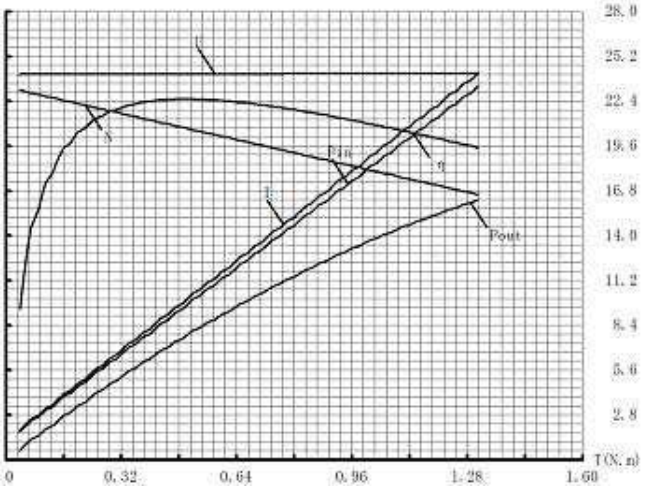
Dimensions



Power Curves & Specifications

单位:浙江尤奈特电机有限公司　　日期: 2005年

N(rpm)	Pout(W)	U(V)	I(A)	Pin(W
5000	700	28.0	28.0	700
4500	630	25.2	25.2	630
4000	560	22.4	22.4	560
3500	490	19.6	19.6	490
3000	420	16.8	16.8	420
2500	350	14.0	14.0	350
2000	280	11.2	11.2	280
1500	210	8.4	8.4	210
1000	140	5.6	5.6	140
500	70	2.8	2.8	70



REFERENCE	Torque (N.m)	Speed (RPM)	Output Power (W)	Voltage (V)	Current (A)	Input Power (W)	Efficiency (%)
No Load	1.24	344	5.41	24.12	1.12	27.01	20.03
Max. Efficiency Point	6.64	299	177.15	24.16	8.73	211.00	83.96
Rate Load	7.46	273	250.84	24.19	13.01	314.78	79.69
Max. Torque Point	7.82	227	400.74	24.27	24.17	585.19	68.48

Technical Specifications

Model	Spec	Voltage (V)	No Load		Rate Load				
			Speed (RPM)	Current (A)	Torque (N.m)	Speed (RPM)	Current (A)	Output Power (W)	Efficiency (%)
1018	250W24V	24	335±5%	≤1.8	7.46±5%	275±5%	≤13.3	250	≥78%
1018	250W36V	36	335±5%	≤1.8	7.46±5%	275±5%	≤8.9	250	≥78%

Physical Packing Specifications

Model	Weight (kg)	Quantity per box	Box gross weight (kg)	Box dimensions (cm)	Packing type
MY1018-250W24V	2.165	12	26.8	37x32x26	General carton
MY1018-250W36V	2.165	12	26.8	37x32x26	General carton

Proximity sensors capacitive series SK1	DC	Ø22/4	Sensing distance Sr: 6 mm ± 10%	Ø30/4	Sensing distance Sr: 2...15 mm adjustable

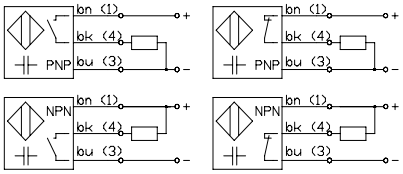
SENSORS

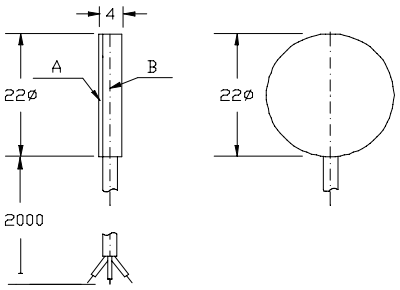
Order key i.e.:

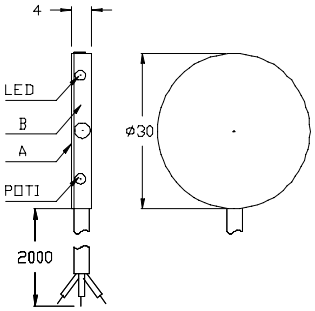
SK1-25-50/10-X-b-X

SK1:	Sensor capacitive, amplifier integrated
25:	Max. sensing distance (mm)
50/10:	Housing diameter/height (mm)
X:	Function switchable P = PNP, N = NPN
b:	Flush mounting
X:	Function switchable S = N.O.; Ö = N.C.

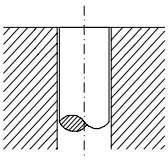
Connection:





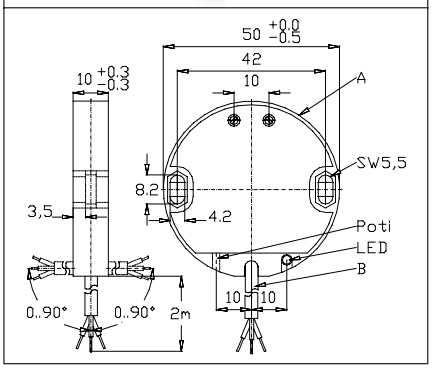


Mounting: flush



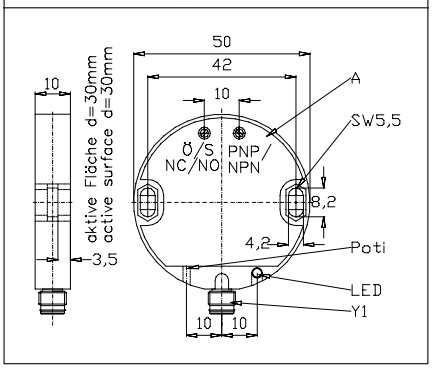
Type	Article No.	**SK1-6F-22/4-P-b-S**	06085	**SK1-15-30/4-P-b-S**	06081
				SK1-15-30/4-P-b-Ö	06082
		SK1-6F-22/4-N-b-S	06087	**SK1-15-30/4-N-b-S**	06083
				SK1-15-30/4-N-b-Ö	06084
Mounting		flush		flush	
Switching hysteresis		≤ 15% of Sr		≤ 15% of Sr	
Switching frequency		100 Hz		100 Hz	
Repeatability (23 ± 5 °C)		≤ 2% of Sr		≤ 2% of Sr	
Ambient temperature		-30...+70 °C		-30...+70 °C	
Temperature drift		≤ 10% of Sr		≤ 10% of Sr	
Protection class IEC 60529		IP 67		IP 67	
Housing material		A: PTFE; B: V2A (stainless steel)		A: PTFE; B: V2A (stainless steel)	
Connection		cable 2 m / 3 x 0.14 mm² PUR		cable 2 m / 3 x 0.14 mm² PUR	
Supply voltage U_B		10...35 VDC		10...35 VDC	
No-load current		approx. 10 mA		approx. 10 mA	
Output stage		semiconductor, I_A max = 300 mA		semiconductor, I_A max = 300 mA	
		U_d (I_e = 50 mA) = 0.8 VDC		U_d (I_e = 50 mA) = 0.8 VDC	
Integrated protective functions		short circuit protection		short circuit protection	
		overload protection		overload protection	
		EMC protective circuit		EMC protective circuit	
		reverse polarity protection		reverse polarity protection	
Conformity		EMC as per IEC 60947-5-2		EMC as per IEC 60947-5-2	

Ø50/10	Sensing distance Sr: 2...25 mm adjustable



SK1-25-50/10-X-b-X	06089
flush	
≤ 20% of Sr	
50 Hz	
≤ 2% of Sr	
-30...+60 °C	
≤ 10% of Sr	
IP 67	
POM	
cable 2 m / 3 x 0.25 mm² PVC	
10...30 VDC	
approx. 15 mA	
semiconductor, I_A max = 150 mA	
U_d (I_e = 50 mA) ≤ 2 VDC	
short circuit protection	
overload protection	
EMC protective circuit	
reverse polarity protection	
EMC as per IEC 60947-5-2	

Ø50/10	Sensing distance Sr: 2...25 mm adjustable



SK1-25-50/10-X-b-X-Y1	06090
flush	
≤ 20% of Sr	
50 Hz	
≤ 2% of Sr	
-30...+60 °C	
≤ 10% of Sr	
IP 65	
POM	
connector Z10, Z11; page 13.03	
quick-disconnect see p. 1.07	
10...35 VDC	
approx. 10 mA	
semiconductor, I_A max = 150 mA	
U_d (I_e = 50 mA) = 1.5 VDC	
short circuit protection	
overload protection	
EMC protective circuit	
reverse polarity protection	
EMC as per IEC 60947-5-2	

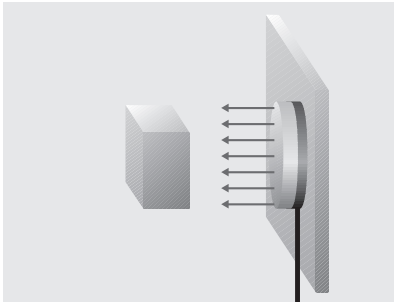
Capacitive disk sensor

As a distance sensor with an adjustable sensing distance of up to 25 mm, the compact-size disk sensor, 50 mm diameter, is suitable for scanning plastics, glass, ceramics, wood, etc. This sensor's large active surface also enables it to be used as a fill level indicator, since it can detect liquids and granules through plastic or glass walls.

Applications

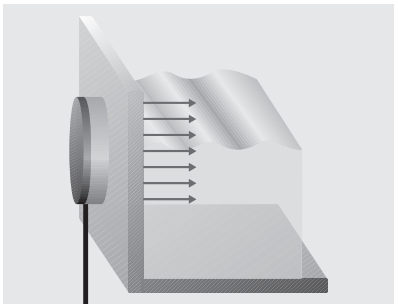
1. Object detection

Detecting plastics, glass, ceramics, wood and all metals, etc.



2. Fill level monitoring

Level control of liquids and granules like water, blood, cereals, coffee beans, etc.



Glass- or plastic wall max. 4 mm, depending on the material of the container wall

Adjustment

The sensor is adjusted using a flush-mounted potentiometer. The adjustment instructions for flush sensors on page 1.09 also apply.

Signal evaluation

The SNG series (page 12.01-12.03) of sensor devices is available for signal evaluation.

Depending on the application involved, you can choose between a power supply, a power supply with timer function, or a MinMax control unit. The sensor can also be run from a PLC.

Appendix B

C-code Compiled from Simulink

```
/*
 * Real-Time Workshop code generation for Simulink model "md_Final.mdl".
 *
 * Model Version                        : 1.1005
 * Real-Time Workshop file version      : 5.1 $Date: 2003/08/08 18:37:24 $
 * Real-Time Workshop file generated on : Wed Oct 26 18:49:09 2005
 * TLC version                          : 5.1 (Aug  8 2003)
 * C source code generated on           : Wed Oct 26 18:49:10 2005
 */

#include <math.h>
#include <string.h>
#include "md_Final.h"
#include "md_Final_private.h"

/* Block signals (auto storage) */
BlockIO rtB;

/* Continuous states */
ContinuousStates rtX;

/* Block states (auto storage) */
D_Work rtDWork;

/* Parent Simstruct */
static rtModel md_Final model_S;
rtModel_md_Final *const rtM_md_Final = &model_S;

/* Initial conditions for root system: '<Root>' */
void MdlInitialize(void)
{

  /* ZeroPole Block: <S4>/Zero-Pole */
  rtX.Zero_Pole_CSTATE = 0.0;
}

/* Start for root system: '<Root>' */
void MdlStart(void)
{
  /* Initialize digital outputs for port PORTA */
  DDRA |= 7;

  /* Blink LEDs on handle bars before starting main program */
  PORTA |= 0x01;
  blinky(100000);
  PORTA |= 0x04;
  blinky(100000);
  PORTA |= 0x02;
  blinky(100000);
  PORTA &= 0xFE;
  blinky(100000);
  PORTA &= 0xFB;
  blinky(100000);

  /* DiscretePulseGenerator Block: <Root>/CMD Generator */
  {
    int_T Ns;
    real_T tFirst = rtmGetTStart(rtM_md_Final);
    Ns = (int_T)floor(tFirst / 0.005 + 0.5);
    if (Ns <= 0) {
      rtDWork.CMD_Generator_IWORK.ClockTicksCounter = Ns;
    } else {
      rtDWork.CMD_Generator_IWORK.ClockTicksCounter = Ns -
        (int_T)(rtP.CMD_Generator_Period*floor((real_T)Ns /
        rtP.CMD_Generator_Period));
    }
  }

  /* S-Function "freePortComms_rxd" initialization Block: <S8>/SEND_VIA_FREE_PORT */
  {

    myUsrBuf *admin;

    /* allocate memory for buffer (1 bytes) and its admin structure, returns the access pointer */
    if((admin = AllocateUserBuffer(0, 1, 1)) == NULL) abort_LED(34);

    /* store the access pointer in the workspace variable */
    freecomTelBuf[0] = admin;

    /* retain a local copy for fast access */
    rtDWork.SEND_VIA_FREE_PORT_PWORK = admin;

    /* open free serial port (SCI0), initialize reception ring buffers */
    FreePort_Init(0, BAUD_38400);
```

```
}

/* S-Function "adc_sfcn_9S12" initialization Block: <S2>/Analog Input SFcn */

/* ATDxCTL2 register bits:
 * [ ADPU AFFC AWAI ETRIGLE ETRGP ETRGE ASCIE ASCIF ]
 *     1 Normal
 *          0  Any access to result reg will clears all flags
 *               0 Power down during wait mode
 *                     0  High level
 *                           0 Rising edge
 *                                 0 Disable external trigger
 *                                      0 Disable sequence complete interrupts
 *                                            0 (This bit read only)
 *  Example: ATD0CTL2 = 0x80;
 */

ATD0CTL2 = 0x80;

/* ATD0CTL3 register bits:
 * [ b7  S8C  S4C  S2C  S1C  FIFO  FRZ1  FRZ0 ]
 *    0 (this bit read only)
 *        0  see Table 3-3 ATD 10B16C Block User Guide
 *             0   see Table 3-3
 *                  0  see Table 3-3
 *                       1  see Table 3-3
 *                             0 non-Fifo mode
 *                                  0  Finish conversion, then freeze
 *                                        0 (combined with above line)
 *  Example: ATD0CTL3 = 0x00;
 */

ATD0CTL3 = 8;

/* ATD0CTL4 register bits:
 * [ SRES8  SMP1  SMP0  PRS4  PRS3  PRS2  PRS1  PRS0 ]
 *    0 ATD resolution select set to 8-bits (0 : 10-bit)
 *           0  Sample time select for conversions
 *                 0 (combined with above)
 *                       0  Default divide by 12
 *                             0 (combined with above)
 *                                   1  (combined with above)
 *                                         0  (combined with above)
 *                                               1 (combined with above)
 *  Example: ATD0CTL4 = 0x05;
 */

ATD0CTL4 = 0x05 | 0;

/* ATD0CTL5 register bits:
 * [ DJM  DSGN  SCAN  MULT  CD  CC  CB  CA ]
 *    1 Right justified data in result registers
 *          0 Signed data representation in result registers
 *               1 Use continuous conversion
 *                     0 Sample multiple channels at a time
 *                          x  For channel selection
 *                              x  same as above
 *                                  x  same as above
 *                                     x  same as above
 *
 * Example: ATD0CTL5 = 0xA0     initiates a right justified
 *                              conversion for channel 0
 *
 * Initialization of this register is done in mdlOutputs
 */

/* Initialize digital inputs for port PORTB */
DDRB &= ~6;

/* S-Function "adc_sfcn_9S12" initialization Block: <S1>/Analog Input SFcn */

/* ATDxCTL2 register bits:
 * [ ADPU AFFC AWAI ETRIGLE ETRGP ETRGE ASCIE ASCIF ]
 *     1 Normal
 *          0  Any access to result reg will clears all flags
 *               0 Power down during wait mode
 *                     0  High level
 *                           0 Rising edge
 *                                 0 Disable external trigger
 *                                      0 Disable sequence complete interrupts
 *                                            0 (This bit read only)
 *  Example: ATD0CTL2 = 0x80;
 */
```

```
  ATD0CTL2 = 0x80;

  /* ATD0CTL3 register bits:
   * [ b7  S8C  S4C  S2C  S1C  FIFO  FRZ1  FRZ0 ]
   *    0 (this bit read only)
   *        0  see Table 3-3 ATD 10B16C Block User Guide
   *             0   see Table 3-3
   *                  0  see Table 3-3
   *                       1  see Table 3-3
   *                             0 non-Fifo mode
   *                                  0  Finish conversion, then freeze
   *                                        0 (combined with above line)
   *  Example: ATD0CTL3 = 0x00;
   */

  ATD0CTL3 = 8;

  /* ATD0CTL4 register bits:
   * [ SRES8  SMP1  SMP0  PRS4  PRS3  PRS2  PRS1  PRS0 ]
   *    0 ATD resolution select set to 8-bits (0 : 10-bit)
   *           0  Sample time select for conversions
   *                 0 (combined with above)
   *                       0  Default divide by 12
   *                             0 (combined with above)
   *                                   1  (combined with above)
   *                                         0  (combined with above)
   *                                               1 (combined with above)
   *  Example: ATD0CTL4 = 0x05;
   */

  ATD0CTL4 = 0x05 | 0;

  /* ATD0CTL5 register bits:
   * [ DJM  DSGN  SCAN  MULT  CD  CC  CB  CA ]
   *    1 Right justified data in result registers
   *          0 Signed data representation in result registers
   *                1 Use continuous conversion
   *                      0 Sample multiple channels at a time
   *                           x  For channel selection
   *                              x  same as above
   *                                 x  same as above
   *                                    x  same as above
   *
   * Example: ATD0CTL5 = 0xA0     initiates a right justified
   *                              conversion for channel 0
   *
   * Initialization of this register is done in mdlOutputs
   */

  /* S-Function "freePortComms_rxd" initialization Block: <S7>/RECEIVE_FROM_FREE_PORT */
  {

    myUsrBuf *admin;

    /* allocate memory for buffer (11 bytes) and its admin structure, returns the access pointer */
    if((admin = AllocateUserBuffer(0, 11, 1)) == NULL) abort_LED(34);

    /* store the access pointer in the workspace variable */
    freecomTelBuf[0] = admin;

    /* retain a local copy for fast access */
    rtDWork.RECEIVE_FROM_FREE_PORT_PWORK = admin;

    /* open free serial port (SCI0), initialize reception ring buffers */
    FreePort_Init(0, BAUD_38400);
  }

  /* Initialize digital outputs for port PORTA */
  DDRA |= 7;

  /* S-Function "servo_pwm_sfcn_9S12" Block: <S9>/servo pwm */

  /* Set PWM initial polarity bit to '1' */
  PWMPOL |= 32;

  /* Set PWM 'scaled clock' bit (SA) */
  PWMCLK |= 32;
  PWMSCLA = 4;

  /* Set PWM clock B and clock A prescalers */
  PWMPRCLK &= 0xF0;
  PWMPRCLK |= 0;

  /* cascading channels 4 & 5 */
```

```
  PWMCTL |= 0x48;

  /* set period register (0.02 seconds) */
  PWMPER45 = 60000;

  /* initialize RWork with an 'impossible' input value -- this enables mdlOutput to run */
  rtDWork.servo_pwm_a_RWORK = -1234.5678;

  /* S-Function "servo_pwm_sfcn_9S12" Block: <S10>/servo pwm */

  /* Set PWM initial polarity bit to '1' */
  PWMPOL |= 8;

  /* Set PWM 'scaled clock' bit (SB) */
  PWMCLK |= 8;
  PWMSCLB = 4;

  /* Set PWM clock B and clock A prescalers */
  PWMPRCLK &= 0x0F;
  PWMPRCLK |= 0;

  /* cascading channels 2 & 3 */
  PWMCTL |= 0x28;

  /* set period register (0.02 seconds) */
  PWMPER23 = 60000;

  /* initialize RWork with an 'impossible' input value -- this enables mdlOutput to run */
  rtDWork.servo_pwm_b_RWORK = -1234.5678;

  MdlInitialize();
}

/* Outputs for root system: '<Root>' */
void MdlOutputs(int_T tid)
{
  /* local block i/o variables */
  real_T rtb_Zero_Pole;
  real_T rtb_Abs_b;
  real_T rtb_Sum1_c;
  real_T rtb_Sum3;
  real_T rtb_Sum;
  real_T rtb_temp24;
  real_T rtb_temp25;
  real_T rtb_temp26;
  real_T rtb_temp27;
  real32_T rtb_Data_Type_Conversion_b[11];
  real32_T rtb_Sum5;
  real32_T rtb_Sum6;
  real32_T rtb_Abs_a;
  real32_T rtb_temp34;
  boolean_T rtb_deg;

  if (rtmIsSampleHit(rtM_md_Final, 1, tid)) { /* Sample time: [0.005, 0.0] */

    /* DiscretePulseGenerator: '<Root>/CMD Generator' */
    rtb_temp24 =
      (rtDWork.CMD_Generator_IWORK.ClockTicksCounter < rtP.CMD_Generator_Duty &&
      rtDWork.CMD_Generator_IWORK.ClockTicksCounter >= 0) ?
     rtP.CMD_Generator_Amp :
      0.0;
    if (rtDWork.CMD_Generator_IWORK.ClockTicksCounter >=
     rtP.CMD_Generator_Period-1) {
      rtDWork.CMD_Generator_IWORK.ClockTicksCounter = 0;
    } else {
      (rtDWork.CMD_Generator_IWORK.ClockTicksCounter)++;
    }

    /* DataTypeConversion: '<S8>/Data Type Conversion' */
    if (rtb_temp24 < 0.0) {
      rtB.Data_Type_Conversion_a = MIN_uint8_T;
    } else if (rtb_temp24 >= MAX_uint8_T) {
      rtB.Data_Type_Conversion_a = MAX_uint8_T;
    } else {
      rtB.Data_Type_Conversion_a = (uint8_T) rtb_temp24;
    }

    /* S-Function "freePortComms_txd" Block: <S8>/SEND_VIA_FREE_PORT */
    {

      myUsrBuf *admin = rtDWork.SEND_VIA_FREE_PORT_PWORK;

      uint16_T size = (uint16_T)admin->buf_size;
      uint8_T *buf = (uint8_T *)admin->buf;
```

```
    uint8_T *blockInput = (uint8_T *)&rtB.Data_Type_Conversion_a;

    /* buffer empty -> check if new user data has arrived */
    if(memcmp(buf, blockInput, size) != 0) {

      /* new block input data available -> copy to the data buffer */
      memcpy(buf, blockInput, size);

      /* send data */
      put_fpdata_SCI0(0, (uint8_T)size, 1, buf, 1);
    }
  }

  /* S-Function "adc_sfcn_9S12" Block: <S2>/Analog Input SFcn */

  /* start single-channel conversion, channel: 5 */
  ATD0CTL5 = 0x80|5;

  /* wait for conversion complete flag CCF0 (last in sequence) */
  while (ATD0STAT1_CCF0 == 0);

  /* 10-bit resolution */
  /* Normalize to a maxium block output value of 5 */
  rtB.Analog_Input_SFcn_a = (real_T)ATD0DR0 / 1023 * 5;

  /* read port data register and return (PORTB) */
  {
    uint8_T value = PORTB;

    {
      int_T i1;

      real_T *y0 = &rtB.Digital_Input_SFcn[0];

      const real_T *p_Digital_Input_SFcn_P3 = &rtP.Digital_Input_SFcn_P3[0];

      for (i1=0; i1 < 2; i1++) {

        if((value & (uint8_T)(1 << (uint8_T)(p_Digital_Input_SFcn_P3[i1]))) >
         0) y0[i1] = 1.0;
        else y0[i1] = 0.0;
      }
    }
  }

  /* S-Function "adc_sfcn_9S12" Block: <S1>/Analog Input SFcn */

  /* start single-channel conversion, channel: 3 */
  ATD0CTL5 = 0x80|3;

  /* wait for conversion complete flag CCF0 (last in sequence) */
  while (ATD0STAT1_CCF0 == 0);

  /* 10-bit resolution */
  /* Normalize to a maxium block output value of 5 */
  rtB.Analog_Input_SFcn_b = (real_T)ATD0DR0 / 1023 * 5;

  /* S-Function "freePortComms_rxd" Block: <S7>/RECEIVE_FROM_FREE_PORT */
  {

    myUsrBuf *admin = freecomTelBuf[0];

    /* attempt to fetch a FreePort telegram from the freePort ring buffer */
    process_fpdata_SCI0(1);

    /* check if output needs updated... */
    if(admin->buffer_full) {

      uint8_T *blockOutput = (uint8_T *)&rtB.RECEIVE_FROM_FREE_PORT[0];

      /* new data available for this instance -> update output */
      (void)memcpy(blockOutput, admin->buf, (uint16_T)admin->buf_size);

      /* clear buffer full flag */
      admin->buffer_full = 0;
    }
  }

  /* DataTypeConversion: '<S3>/Data Type Conversion' */
  {
    int_T i1;

    const uint8_T *u0 = &rtB.RECEIVE_FROM_FREE_PORT[0];
    real32_T *y0 = &rtb_Data_Type_Conversion_b[0];
```

```
      for (i1=0; i1 < 11; i1++) {
        y0[i1] = (real32_T)u0[i1];
      }
    }

    /* Sum: '<S3>/Sum5' incorporates:
     *   Gain: '<S3>/Gain'
     *
     * Regarding '<S3>/Gain':
     *   Gain value: rtP.Gain_Gain
     */
    rtb_Sum5 = (rtb_Data Type Conversion_b[1] * rtP.Gain_Gain)
      + rtb_Data_Type_Conversion_b[2];
  }

  if (rtmIsSampleHit(rtM_md_Final, 1, tid)) { /* Sample time: [0.005, 0.0] */

    /* Switch: '<S3>/Switch2'
     *
     * Regarding '<S3>/Switch2':
     * Switch Block: '<S3>/Switch2'
     * Input0  Data Type:  Floating Point real32_T
     * Input1  Data Type:  Floating Point real32_T
     * Input2  Data Type:  Floating Point real32_T
     * Output0 Data Type:  Floating Point real32_T
     * Round Mode: Floor
     * Saturation Mode: Wrap
     *
     * Threshold parameter uses the same data type and scaling as Input1
     *
     */
    if ( rtb_Sum5 > rtP.Switch2_Threshold )
    {

      /* Sum: '<S3>/Sum6' incorporates:
       *   Constant: '<S3>/Constant'
       *
       * Regarding '<S3>/Sum6':
       * Sum Block: '<S3>/Sum6'
       *
       *  y = - u0 + u1
       *
       * Input0  Data Type:  Floating Point real_T
       * Input1  Data Type:  Floating Point real32_T
       * Output0 Data Type:  Floating Point real32_T
       * Round Mode: Floor
       * Saturation Mode: Saturate
       */
      rtb_Sum6 = rtb_Sum5;
      rtb_Sum6 -= ((float)rtP.Constant_Value);

      rtb_temp34 = rtb_Sum6;
    }
    else
    {
      rtb_temp34 = rtb_Sum5;
    }

    /* Gain: '<S3>/Roll Gain'
     *
     * Regarding '<S3>/Roll Gain':
     *   Gain value: rtP.Roll_Gain_Gain
     */
    rtb_temp34 *= rtP.Roll_Gain_Gain;

    /* Abs: '<Root>/Abs' */
    rtb_Abs_a = rt_ABS(rtb_temp34);

    /* RelationalOperator: '<Root>/<45deg' incorporates:
     *   Constant: '<Root>/Kill Angle'
     *
     * Regarding '<Root>/<45deg':
     * RelationalOperator Block: '<Root>/<45deg'
     * <
     * Input0  Data Type:  Floating Point real32_T
     * Input1  Data Type:  Floating Point real_T
     * Output0 Data Type:  Pure Integer   U8
     */
    rtb_deg = (((real_T)rtb_Abs_a)) < (rtP.Kill_Angle_Value);

    /* Logic: '<Root>/Angle AND foot' incorporates:
     *   Logic: '<Root>/Logical Operator'
     */
```

```
    rtB.Angle_AND_foot = rtb_deg && (!(rtB.Digital_Input_SFcn[0] &&
      rtB.Digital_Input_SFcn[1]));

    /* DataTypeConversion: '<Root>/Data Type Conversion5' */
    rtb_temp24 = (real_T)rtB.Angle_AND_foot;

    /* DataTypeConversion: '<Root>/Data Type Conversion7' incorporates:
     *   RelationalOperator: '<Root>/Relational Operator'
     *   Constant: '<Root>/BatteryLowValue (~80%)'
     */
    rtb_temp25 = (real_T)(rtB.Analog_Input_SFcn_b <=
      rtP.BatteryLowValue_80_Value);

    /* HiddenBuffer */
    rtB.TmpHiddenBuffer_Feeding_Digital[0] = rtb_temp25;
    rtB.TmpHiddenBuffer_Feeding_Digital[1] = rtP.Power_On_Value;
    rtB.TmpHiddenBuffer_Feeding_Digital[2] = rtb_temp24;

    /* assemble port value and set port (PORTA) */
    {
      uint8_T value = PORTA;

      {
        int_T i1;

        const real_T *u0 = &rtB.TmpHiddenBuffer_Feeding_Digital[0];

        const real_T *p_Digital_Output_SFcn_P3 = &rtP.Digital_Output_SFcn_P3[0];

        for (i1=0; i1 < 3; i1++) {

          if(u0[i1] >= rtP.Digital_Output_SFcn_P4) value |= (1 <<
            (uint8_T)p_Digital_Output_SFcn_P3[i1]);
          if(u0[i1] < rtP.Digital_Output_SFcn_P5) value &= ~(1 <<
            (uint8_T)p_Digital_Output_SFcn_P3[i1]);
        }
      }

      /* set port value */
      PORTA = value;
    }

    /* Sum: '<S4>/Sum1' incorporates:
     *   Constant: '<Root>/Setpoint'
     *
     * Regarding '<S4>/Sum1':
     * Sum Block: '<S4>/Sum1'
     *
     *  y =  u0 + u1
     *
     * Input0  Data Type:  Floating Point real_T
     * Input1  Data Type:  Floating Point real32_T
     * Output0 Data Type:  Floating Point real_T
     * Round Mode: Floor
     * Saturation Mode: Saturate
     */
    rtB.Sum1_a = rtP.Setpoint_Value;
    rtB.Sum1_a += ((real_T)rtb_temp34);

    /* Gain: '<S4>/Kp2' incorporates:
     *   Product: '<S4>/Kp'
     *   Constant: '<S4>/Kp1'
     *
     * Regarding '<S4>/Kp2':
     *   Gain value: rtP.Kp2_Gain
     */
    rtB.Kp2 = (rtB.Sum1_a * rtP.Kp1_Value) * rtP.Kp2_Gain;
  }

  if (rtmIsContinuousTask(rtM_md_Final, tid)) { /* Sample time: [0.0, 0.0] */

    /* ZeroPole Block: <S4>/Zero-Pole */
    {
      rtb_Zero_Pole = rtP.Zero_Pole_D*rtB.Sum1_a;
      rtb_Zero_Pole += rtP.Zero_Pole_C*rtX.Zero_Pole_CSTATE;
    }

    /* Sum: '<S4>/Sum' incorporates:
     *   Gain: '<S4>/Kd2'
     *   Product: '<S4>/Kd'
     *   Constant: '<S4>/Kd1'
     *
     * Regarding '<S4>/Kd2':
     *   Gain value: rtP.Kd2_Gain
```

```
     */
    rtb_Sum = rtB.Kp2 + ((rtb_Zero_Pole * rtP.Kd1_Value) * rtP.Kd2_Gain);
  }

  if (rtmIsSampleHit(rtM_md_Final, 1, tid)) { /* Sample time: [0.005, 0.0] */

    /* Sum: '<S11>/Sum1' incorporates:
     *   DataTypeConversion: '<S11>/Data Type  Conversion4'
     *   Constant: '<S11>/2.55'
     *
     * Regarding '<S11>/Sum1':
     * Sum Block: '<S11>/Sum1'
     *
     *  y =  u0 - u1
     *
     * Input0  Data Type:  Floating Point real32_T
     * Input1  Data Type:  Floating Point real_T
     * Output0 Data Type:  Floating Point real_T
     * Round Mode: Floor
     * Saturation Mode: Saturate
     */
    rtb_temp25 = ((real_T)((real32_T)rtB.Analog_Input_SFcn_a));
    rtb_temp25 -= rtP.id_Value;

    /* DeadZone: '<S11>/Dead Zone' */
    if (rtb_temp25 >= rtP.Dead_Zone_Start) {
      rtB.Dead_Zone = rtb_temp25 - rtP.Dead_Zone_Start;
    } else if (rtb_temp25 <= rtP.Dead_Zone_End) {
      rtB.Dead_Zone = rtb_temp25 - rtP.Dead_Zone_End;
    } else {
      rtB.Dead_Zone = 0.0;
    }
  }

  if (rtmIsContinuousTask(rtM_md_Final, tid)) { /* Sample time: [0.0, 0.0] */

    /* Abs: '<S11>/Abs' */
    rtb_Abs_b = fabs(rtb_Sum);

    /* MultiPortSwitch: '<S11>/Multiport Switch' incorporates:
     *   Rounding: '<S11>/Rounding Function1'
     *   Saturate: '<S11>/Saturation'
     *   Constant: '<S11>/FastTurn'
     *   Constant: '<S11>/MedTurn'
     *   Constant: '<S11>/SlowTurn'
     *   Constant: '<S11>/SlowestTurn'
     *
     * Regarding '<S11>/Saturation':
     *   Lower limit: rtP.Saturation_LowerSat
     *   Upper limit: rtP.Saturation_UpperSat
     */
    switch
    ((int_T)(ceil(rt_SATURATE(rtb_Abs_b,rtP.Saturation_LowerSat,rtP.Saturation_UpperSat))))
    {
     case 1:

      rtb_temp26 = rtP.FastTurn_Value;
      break;
     case 2:

      rtb_temp26 = rtP.FastTurn_Value;
      break;
     case 3:

      rtb_temp26 = rtP.MedTurn_Value;
      break;
     case 4:

      rtb_temp26 = rtP.SlowTurn_Value;
      break;
     case 5:

      rtb_temp26 = rtP.SlowestTurn_Value;
      break;
     default:
      /* Result undefined */
#if defined(MATLAB_MEX_FILE)
      (void)mexPrintf("Error: Invalid control input for block:"
       "md_Final/TurnSubsys/Multiport Switch\n"
       "Result is undefined.");
#endif
      break;
    }
```

```
    /* Product: '<S11>/Product' */
    rtb_temp26 = rtB.Dead_Zone * rtb_temp26;

    /* Sum: '<Root>/Sum1' */
    rtb_Sum1_c = rtb_Sum + rtb_temp26;
  }

  if (rtmIsContinuousTask(rtM_md_Final, tid)) { /* Sample time: [0.0, 0.0] */

    /* Switch: '<Root>/Switch1' incorporates:
     *   Saturate: '<Root>/Saturation1'
     *   Constant: '<Root>/KillSig'
     *
     * Regarding '<Root>/Saturation1':
     *   Lower limit: rtP.Saturation1_LowerSat
     *   Upper limit: rtP.Saturation1_UpperSat
     */
    if (rtB.Angle_AND_foot) {
      rtb temp27 =
        rt SATURATE(rtb_Sum1_c,rtP.Saturation1_LowerSat,rtP.Saturation1_UpperSat);
    } else {
      rtb_temp27 = rtP.KillSig_Value;
    }

    /* Sum: '<Root>/Sum2' incorporates:
     *   Constant: '<Root>/Offset'
     */
    rtB.Sum2 = rtb_temp27 + rtP.Offset_Value;
  }

  if (rtmIsSampleHit(rtM_md_Final, 1, tid)) { /* Sample time: [0.005, 0.0] */

    /* S-Function "servo_pwm_sfcn_9S12" Block: <S9>/servo pwm
     *
     * Set PWM duty cycle whenever the input signal "u" changes
     */
    {

      real_T u;

      /* get input value */
      u = rtB.Sum2;

      /* only update unit if the input voltage 'u' has changed */
      if(u != rtDWork.servo_pwm_a_RWORK) {

        /* retain current input value for the next time round... */
        rtDWork.servo_pwm_a_RWORK = u;

        /* check polarity of input voltage */
        if (u < 0) {

          /* negative inputs : reset to '0' */
          u = 0;
        }

        /* limit input voltage to the specified maximum (Vsat) */
        if(u > rtP.servo_pwm_a_P4) {

          u = rtP.servo_pwm_a_P4;
        }

        /* new duty cycle (store in IWork[0]) */
        rtDWork.servo_pwm_a_IWORK = (uint_T)(u/rtP.servo_pwm_a_P4 * 2700 + 3120);

        /* duty cycle channels 4 & 5 */
        PWMDTY45 = rtDWork.servo_pwm_a_IWORK;
      }                                 /* if(u !=  u) */
    }                                   /* S-Function "servo_pwm_sfcn_9S12" Block: <S9>/servo pwm */

    /* enable PWM channel 5 */
    PWME |= 32;
  }

  if (rtmIsContinuousTask(rtM_md_Final, tid)) { /* Sample time: [0.0, 0.0] */

    /* Sum: '<Root>/Sum3' */
    rtb_Sum3 = rtb_Sum - rtb_temp26;
  }

  if (rtmIsContinuousTask(rtM_md_Final, tid)) { /* Sample time: [0.0, 0.0] */

    /* Switch: '<Root>/Switch' incorporates:
     *   Saturate: '<Root>/Saturation2'
```

```
     *   Constant: '<Root>/KillSig'
     *
     * Regarding '<Root>/Saturation2':
     *   Lower limit: rtP.Saturation2_LowerSat
     *   Upper limit: rtP.Saturation2_UpperSat
     */
    if (rtB.Angle AND_foot) {
      rtb_temp27 =
        rt_SATURATE(rtb_Sum3,rtP.Saturation2_LowerSat,rtP.Saturation2_UpperSat);
    } else {
      rtb_temp27 = rtP.KillSig_Value;
    }

    /* Sum: '<Root>/Sum4' incorporates:
     *   Constant: '<Root>/Offset'
     */
    rtB.Sum4 = rtP.Offset_Value + rtb_temp27;
  }

  if (rtmIsSampleHit(rtM_md_Final, 1, tid)) { /* Sample time: [0.005, 0.0] */

    /* S-Function "servo_pwm_sfcn_9S12" Block: <S10>/servo pwm
     *
     * Set PWM duty cycle whenever the input signal "u" changes
     */
    {

      real_T u;

      /* get input value */
      u = rtB.Sum4;

      /* only update unit if the input voltage 'u' has changed */
      if(u != rtDWork.servo_pwm_b_RWORK) {

        /* retain current input value for the next time round... */
        rtDWork.servo_pwm_b_RWORK = u;

        /* check polarity of input voltage */
        if (u < 0) {

          /* negative inputs : reset to '0' */
          u = 0;
        }

        /* limit input voltage to the specified maximum (Vsat) */
        if(u > rtP.servo_pwm_b_P4) {

          u = rtP.servo_pwm_b_P4;
        }

        /* new duty cycle (store in IWork[0]) */
        rtDWork.servo_pwm_b_IWORK = (uint_T)(u/rtP.servo_pwm_b_P4 * 2700 + 3120);

        /* duty cycle channels 2 & 3 */
        PWMDTY23 = rtDWork.servo_pwm_b_IWORK;
      }                                 /* if(u != _u) */
    }                                   /* S-Function "servo_pwm_sfcn_9S12" Block: <S10>/servo pwm */

    /* enable PWM channel 3 */
    PWME |= 8;
  }
}

/* Update for root system: '<Root>' */
void MdlUpdate(int_T tid)
{
}

/* Derivatives for root system: '<Root>' */
void MdlDerivatives(void)
{
  /* simstruct variables */
  StateDerivatives *rtXdot = (StateDerivatives*) rtM_md_Final->ModelData.derivs;

  /* ZeroPole Block: <S4>/Zero-Pole */
  {

    rtXdot->Zero_Pole_CSTATE = rtB.Sum1_a;
    rtXdot->Zero_Pole_CSTATE += (rtP.Zero_Pole_A)*rtX.Zero_Pole_CSTATE;
  }
}

/* Projection for root system: '<Root>' */
```

```
void MdlProjection(void)
{
}

/* Terminate for root system: '<Root>' */
void MdlTerminate(void)
{
  if(rtM_md_Final != NULL) {

    /* free instance local data buffer */
    (void)free(((myUsrBuf *)rtDWork.SEND_VIA_FREE_PORT_PWORK)->buf);
    (void)free((myUsrBuf *)rtDWork.SEND_VIA_FREE_PORT_PWORK);

    /* reset global buffer access pointer to NULL */
    freecomTelBuf[0] = NULL;

    /* switch ATD unit 0 off again */
    ATD0CTL2 &= ~0x80;

    /* switch ATD unit 0 off again */
    ATD0CTL2 &= ~0x80;

    /* free instance local data buffer */
    (void)free(((myUsrBuf *)rtDWork.RECEIVE_FROM_FREE_PORT_PWORK)->buf);
    (void)free((myUsrBuf *)rtDWork.RECEIVE_FROM_FREE_PORT_PWORK);

    /* reset global buffer access pointer to NULL */
    freecomTelBuf[0] = NULL;

    /* disable Servo Motor PWM channel 5 */
    PWME &= ~32;

    /* disable Servo Motor PWM channel 3 */
    PWME &= ~8;
  }
}

/* Function to initialize sizes */
void MdlInitializeSizes(void)
{
  rtM_md_Final->Sizes.numContStates = (1); /* Number of continuous states */
  rtM_md_Final->Sizes.numY = (0);       /* Number of model outputs */
  rtM_md_Final->Sizes.numU = (0);       /* Number of model inputs */
  rtM_md_Final->Sizes.sysDirFeedThru = (0); /* The model is not direct feedthrough */
  rtM_md_Final->Sizes.numSampTimes = (2); /* Number of sample times */
  rtM_md_Final->Sizes.numBlocks = (64); /* Number of blocks */
  rtM_md_Final->Sizes.numBlockIO = (12); /* Number of block outputs */
  rtM_md_Final->Sizes.numBlockPrms = (78); /* Sum of parameter "widths" */
}

/* Function to initialize sample times */
void MdlInitializeSampleTimes(void)
{
  /* task periods */
  rtM_md_Final->Timing.sampleTimes[0] = (0.0);
  rtM_md_Final->Timing.sampleTimes[1] = (0.005);

  /* task offsets */
  rtM_md_Final->Timing.offsetTimes[0] = (0.0);
  rtM_md_Final->Timing.offsetTimes[1] = (0.0);
}

/* Function to register the model */
rtModel_md_Final *md_Final(void)
{
  (void)memset((char *)rtM_md_Final, 0, sizeof(rtModel_md_Final));

  {
    /* Setup solver object */
    static RTWSolverInfo rt_SolverInfo;
    rtM_md_Final->solverInfo = (&rt_SolverInfo);

    rtsiSetSimTimeStepPtr(rtM_md_Final->solverInfo,
     &rtM_md_Final->Timing.simTimeStep);
    rtsiSetTPtr(rtM_md_Final->solverInfo, &rtmGetTPtr(rtM_md_Final));
    rtsiSetStepSizePtr(rtM_md_Final->solverInfo, &rtM_md_Final->Timing.stepSize);
    rtsiSetdXPtr(rtM_md_Final->solverInfo, &rtM_md_Final->ModelData.derivs);
    rtsiSetContStatesPtr(rtM_md_Final->solverInfo,
     &rtM_md_Final->ModelData.contStates);
    rtsiSetNumContStatesPtr(rtM_md_Final->solverInfo,
     &rtM_md_Final->Sizes.numContStates);
    rtsiSetErrorStatusPtr(rtM_md_Final->solverInfo,
     &rtmGetErrorStatus(rtM_md_Final));
```

```
    rtsiSetRTModelPtr(rtM_md_Final->solverInfo, rtM_md_Final);
  }

  /* timing info */
  {
    static time_T mdlPeriod[NSAMPLE_TIMES];
    static time_T mdlOffset[NSAMPLE_TIMES];
    static time_T mdlTaskTimes[NSAMPLE_TIMES];
    static int_T mdlTsMap[NSAMPLE_TIMES];
    static int_T mdlSampleHits[NSAMPLE_TIMES];

    {
      int_T i;

      for(i = 0; i < NSAMPLE_TIMES; i++) {
        mdlPeriod[i] = 0.0;
        mdlOffset[i] = 0.0;
        mdlTaskTimes[i] = 0.0;
      }
    }
    (void)memset((char_T *)&mdlTsMap[0], 0, 2 * sizeof(int_T));
    (void)memset((char_T *)&mdlSampleHits[0], 0, 2 * sizeof(int_T));

    rtM_md_Final->Timing.sampleTimes = (&mdlPeriod[0]);
    rtM_md_Final->Timing.offsetTimes = (&mdlOffset[0]);
    rtM_md_Final->Timing.sampleTimeTaskIDPtr = (&mdlTsMap[0]);
    rtmSetTPtr(rtM_md_Final, &mdlTaskTimes[0]);
    rtM_md_Final->Timing.sampleHits = (&mdlSampleHits[0]);
  }
  rtsiSetSolverMode(rtM_md_Final->solverInfo, SOLVER_MODE_SINGLETASKING);

  /*
   * initialize model vectors and cache them in SimStruct
   */

  /* block I/O */
  {
    void *b = (void *) &rtB;
    rtM_md_Final->ModelData.blockIO = (b);

    (void)memset(b, 0, sizeof(BlockIO));

    {
      int_T i;

      b =&rtB.Analog Input SFcn a;
      for (i = 0; i < 12; i++) {
        ((real_T*)b)[i] = 0.0;
      }
    }
  }

  /* parameters */
  rtM_md_Final->ModelData.defaultParam = ((real_T *) &rtP);

  /* states */
  {
    int_T i;
    real_T *x = (real_T *) &rtX;
    rtM_md_Final->ModelData.contStates = (x);
    for(i = 0; i < (int_T)(sizeof(ContinuousStates)/sizeof(real_T)); i++) {
      x[i] = 0.0;
    }
  }

  /* data type work */
  {
    void *dwork = (void *) &rtDWork;
    rtM_md_Final->Work.dwork = (dwork);
    (void)memset((char_T *) dwork, 0, sizeof(D_Work));
    {
      int_T i;
      real_T *dwork_ptr = (real_T *) &rtDWork.servo_pwm_a_RWORK;

      for (i = 0; i < 2; i++) {
        dwork_ptr[i] = 0.0;
      }
    }
  }

  /* Model specific registration */

  rtM_md_Final->modelName = ("md_Final");
  rtM_md_Final->path = ("md_Final");
```

```
  rtmSetTStart(rtM_md_Final, 0.0);
  rtM_md_Final->Timing.tFinal = (-1);
  rtM_md_Final->Timing.stepSize = (0.005);
  rtsiSetFixedStepSize(rtM_md_Final->solverInfo, 0.005);

  rtM_md_Final->Sizes.checksums[0] = (279584964U);
  rtM_md_Final->Sizes.checksums[1] = (2244223745U);
  rtM_md_Final->Sizes.checksums[2] = (3259867442U);
  rtM_md_Final->Sizes.checksums[3] = (4169898757U);

  return rtM_md_Final;
}
```