Distributed Motion Controllers for a Humanoid Robot

Andrew Hood

ABSTRACT

The challenge of building a fully autonomous humanoid robot was taken up by 12 final year engineering students at the University of Queensland in February 2001. A complete paper design was completed by the end of that year, and by February 2002 the GuRoo was structurally built.

The topic of this thesis was to bring the paper design of the motion controllers into reality and to a functional state. In doing so, conduct a thorough review on the entire operation of the control system. The hardware has been redesigned: improving the old system. Also, a mechanism for initial alignment of joints has been developed.

The 2001 motor control design was realized through a distributed control system – as distinct from a centralised system where control for every joint is coordinated by one processor. The iPAQ pocket PC in GuRoo's chest generates velocity profiles for each joint. These velocities are broadcast every 2ms onto the Controller Area Network(CAN). Each of the 6 joint controllers independently retrieves the appropriate data, and actuates the associated motors.

The distributed control system was found to be very effective. However, the implementation of the local control limited the speed of the control algorithm to 250Hz – the desired speed was 2kHz. This bottleneck has been addressed in the 2002 revision of the controllers. The new controllers will also use optical switches to accurately initialise each joint. The new design is more power efficient, smaller, and reduces the wiring harness complexity.

STATEMENT OF ORIGINALITY

Andrew Hood
230 Pullenvale Rd.
Pullenvale, QLD, 4069

The Dean,
School of Engineering,
The University of Queensland,
St. Lucia, QLD, 4072.

Dear Professor Simmons,

In conforming with the requirements of the degree of Bachelor of Engineering in the division of Electrical Engineering, I present the following thesis entitled 'Distributed Motion Controllers for a Humanoid Robot.'

The work was performed under the supervision of Dr Gordon Wyeth.

I declare that the work presented in this thesis has not been previously submitted for a degree at any institution. To the best of my knowledge, this thesis contains no material published by any other person, except where reference is made in text.

Yours sincerely,

Andrew Hood

ACKNOWLEDGEMENTS

I wish to thank the following people for their contributions to my sanity, and this project:

Gordon Wyeth, for his supervision and assistance throughout the year

Geoff Walker, for his help with the power circuitry

Damo, for his open ear and tremendous help with anything and everything

The other guys working with me on GuRoo, Ian and Adam for their help in making this such a fun project

The rest of the Robotics lab, particularly Rob, Chris and Adam B, for their help, and company through many late nights

My housemates, particularly Nick, for guarding the house during my absence

My other friends, especially Mike, Bec, Rob and Ruth, for their un-faltering support of my work and encouragement to occasionally leave it alone

My family, for their understanding of my lack of presence and unwavering support and interest

TO GUROO

1- INTRODUCTION

- By the year 2050,
develop a team of fully autonomous humanoid robots
that can win against the human world soccer champions. [1]

An ambitious, but attainable goal (pun intended). The task requires much further development in the engineering disciplines of image processing, mechanical design, robot control, and robot intelligence. The RoboCup organization holds annual competitions to develop individual skills necessary to complete the challenge. 2002 was the first year of the humanoid league competition.

In 2001 a fully autonomous humanoid robot, GuRoo (Figure 1) was designed and built at the University of Queensland. This thesis concerns the control of the robot's joints.

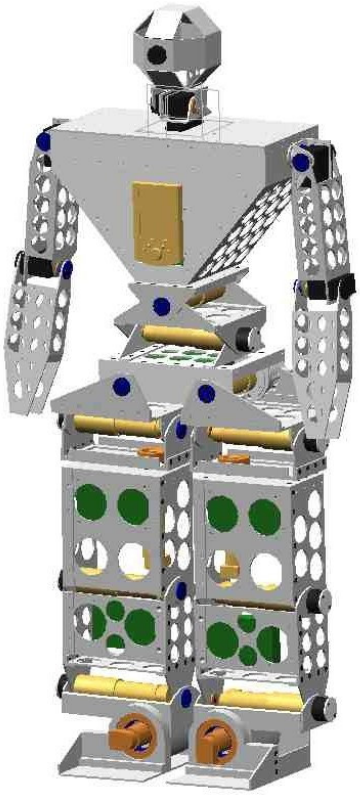


Figure 1-CAD model of GuRoo

1.1- Thesis Overview

The control system for the humanoid was designed and built by two undergraduate students, Jarad Stirzaker and Tim Cartwright, in 2001. Although the hardware design was completed, comprehensive testing was still required. Much of the software also needed writing. A review of the system – which had been developed blind, without a robot to test on – was necessary to measure its effectiveness in the real world. Having learnt from the operation of the 2001 solution, the new design was developed.

This thesis deals only with the low-level control of the robot, much work is yet to be done in higher level programming (gait generation, balance, efficient walking algorithms). This goal of this project is to build a solid base on which those things can be developed.

1.2- Motivation

Apart from amusing the engineers involved with designing and building a humanoid robot, an autonomous machine in the form of a human has positive social ramifications. Its most obvious application would be in rescue operations, and situations that would normally involve endangering human life. For a robot to operate in such situations, it is logical that it would be most effective if it were of the same form and dimensions as the original operator. Wheeled robots already assist humans in dangerous situations: search and rescue robots have been sent into the volatile sites at Chernobyl, Moura Mine, and recently at the World Trade Center in New York. Wheeled robots are very limited in their accessibility on uneven terrain, and legged machines would be much better in uncontrolled environments.

The human form is very top heavy, and unstable. We balance while standing without consciously thinking about it, by constantly flexing and relaxing hundreds of muscles. While running, we maintain dynamic stability (at no point in their trajectory is a runner able to stop moving and still be stable). We can negotiate rough terrain with admirable ease. To complete all three of these tasks, we rely heavily on the precise control of each of our limbs.

To model our hundreds of muscles by 15 heavy motors in the lower body can at best be described as crude. It can also be described as difficult. Precise feedback from each joint is necessary. Currently, the robot is programmed with a specific gait (tested first on the simulator) in which a velocity profile for each motor is generated on a central computer, and broadcast to each board every 2ms. The motors must accurately turn at the speeds they are told to, any error will be carried through and could cause GuRoo to fall over.

To design such a control system in a humanoid robot is of great academic value. By logging data from the joints as they move, a great deal could be learned about human joint control, and further development can be made towards humanoid robotics

1.3- Development so Far – Other humanoids

Figure 2 - Honda's ASIMO

The relatively new field of humanoid robotics includes two main groups of developers. Those robots developed for commercialisation, and those for research.

Two large corporations: Sony and Honda both have humanoid robots. ASIMO is 10th in a line of humanoids built by Honda over the last 17 years. She is 120cm tall, with speech synthesis, ambidextrous hands and stereo vision[2]. The robot walks smoothly and does cute dances for the public, it can also walk up stairs and reports of her successfully running are undocumented though believable.

Sony's SDR-4X is 50cm tall, weighs 6.5kg, and uses a pair of 64-bit processors, it also incorporates stereo vision, speech recognition and speech synthesis to a purchasable product[3]. The robot's adaptability to disturbances is very impressive as it negotiates a rough terrain.

Notable humanoid robots built for research include Waseda'a Wabian [4], and Gifu Industries' Association's NAGARA, both in Japan.

NAGARA, seen here in Figure 3, is only 83cm tall and weighs 15kg – considerably smaller than GuRoo. Like GuRoo, it utilises distributed control to coordinate its joints – which constitute an impressive 28 degrees of freedom. NAGARA contains two main CPUs (one dedicated to walking) and several motor controllers. These communicate with IEEE 1394. NAGARA won the Best Humanoid award at RoboCup 2002, having taken out the humanoid walk and penalty shoot competitions in its class.

Figure 3 ***- NAGARA humanoid - Winner of Best Humanoid award at RoboCup 2002. Seen here with its elated crew from Gifu Industries' Association.***

WABIAN (WAseda BIpedal humANoid) is an especially large humanoid – standing at 166 cm, it weighs 107kg. Despite this, its 43 motors (15 AC) can propel the beast at up to 0.21m/s. WABIAN has 51 degrees of freedom, 43 of which are active.

Waseda University has been instrumental in the development of humanoid robotics technology. The range and success of the humanoid projects conducted at the university is impressive. From the development of musical humanoids to artificial joint activators based on pneumatically controlled pouches, their developments have been ongoing since 1970[4]

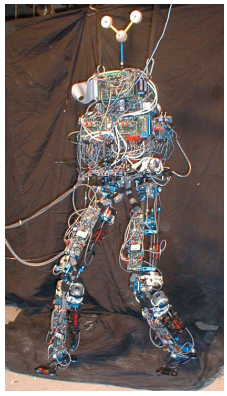


Figure 4 - M2 biped form MIT

MIT's leg lab also have a large scale bi-ped, M2, an exercise in design and manufacturing rather than gait generation (as is the GuRoo at this point). H7 from the University of Tokyo is one of the most stable, well controlled large scale humanoids.[5]

Figure 5 - University of Tokyo's H7

In terms of the control methodologies, many of the very small robots running low power motors are able to have single motor controllers to actuate the machines. Of those of similar size to GuRoo, Waseda's WABIAN, MIT's M2, and The University of Tokyo's H7 all have distributed control systems, though the drive systems being controlled vary from GuRoo's DC motors. The details of the methods used for these systems are not known specifically.

1.4- Development so Far – GuRoo

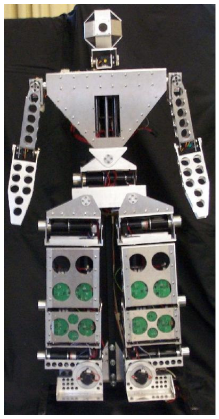


Figure 6 - GuRoo- hanging on his stand

The project at hand, as mentioned previously, is in its second year of development. Last years achievements were the finalisation of a mechanical design, the adaptation of an existing simulator program to calculate relevant forces, and of course simulate, the walking gaits developed. The high level control and gait generation to make the GuRoo walk were also done so some extent, as were the hardware to control the joints. Vision has been developed for him, though not integrated onto the robot. Over the course of this year the GuRoo was mechanically built, the joint controllers were completed and the software was tested and developed in a real environment. Vision has still not been incorporated.

1.5- Other team members

Relevant work by others on GuRoo, both in 2001 and 2002, is outlined in the following undergraduate theses, available from the University of Queensland:

Author	Title	Year
Jarad Stirzaker	Design of DC Motor Controllers for a Humanoid Robot	2001
Adam Drury	Gait Generation & Control Algorithms for a Humanoid Robot	2002
Ian Marshall	Active Balance Control for a Humanoid Robot	2002
Tim Cartwright	Design and Implementation of Small Scale Joint Controllers for a Humanoid Robot	2001
Damien Kee	Design and Simulation of a Humanoid Drive System	2001
Mark Wagstaff	Mechanical Design and Internal Sensors for a Humanoid	2001

There are other papers to do with the GuRoo project, however they are not directly relevant to the topic of this thesis.

1.6- Chapter Outline

This thesis is broken into two distinct parts, though the chapter numbering continues through the document.

Part A is a discussion of the boards designed by Stirzaker and Cartwright in 2001. The hardware design in Chapter 2 is completely their work, though this section also investigates the requirements and background theory of the components of the design. The software to control these boards is the topic of Chapter 3, the skeletons of which were done by Stirzaker. The code was completed as part of this thesis, to have the boards working by the RoboCup competition in July 2002. Chapter 4 is a brief chapter about board placement and layout. A synopsis of the entire control system design forms Chapter 5, a discussion of what was effective and what wasn't, as well as what is necessary to improve.

Part B is about the new design. From what was learnt in first semester, the specifications of a new design were determined, and a new revision of the system made. The hardware design is complete as per Chapter 6, and the software that needs to be written is detailed in Chapter 7. Chapter 8 talks about the layout and positioning of the new boards.

Chapter 9 is a conclusion of the work done this year, and its effectiveness, as well as a summary of the new design

Chapter 10 outlines where this project could go in the future, and specifies work yet to be completed on the new design

PART A

BRINGING THE INCOMPLETE 2001 BOARDS TO FULL FUNCTIONALITY BY THE ROBOCUP 2002 COMPETITION IN JULY, AND A COMPLETE REVIEW OF THAT DESIGN.

2- BREAKING UP THE DESIGN – HARDWARE

2.1- Introduction

This chapter describes the distributed control hardware system of the GuRoo. Both requirements of the system and associated subsystems are discussed, and the design decisions made by Stirzaker[6] and Cartwright[7]. Where appropriate, background theory of a subsystem is explained.

The manner in which the Stirzaker and Cartwright decided to control the joints on the robot was based on the successful implementation of a distributed control system for a Puma robotic arm by Kennedy in 1999 [8]. The design was a step away from the conventional control of that arm, which involves a central box containing analogue and digital control hardware, power equipment and ran sensor and motor power cables to each joint individually. In his distributed control system a central processor exists, though it only generates the movements required by the joints, which it broadcasts onto a data bus. All low-level control is done on local controller boards, which receive data from the bus and actuate those motors connected to it. Such a system greatly reduces wiring harness complexity and requires less powerful processors (though more are needed) to achieve the same accuracy. It also allows modularisation of the intelligence and the control into separate components such that either can be upgraded without affecting the other, as this thesis does indeed do.

The mechanical structure of the humanoid seems an ideal place for such a system, as the majority of 'free' space is close to the motors (in his legs), and not in a central location (chest or head). It was decided that 5 boards would do the control for 3 DC motors each in the lower body of GuRoo. Another single board would control the low-power servo motors in the head and arms. This seemed a good compromise between saving the limited space available and the loss of speed and thus resolution by controlling multiple motors. The rest of this thesis details the design of the DC motor boards in the lower body. The servo motor controller is practically identical, with all motor drive circuitry replaced by a logic tri-state buffer.

The communication standard used is CAN (Controller Area Network) [9]. Developed by Bosch GmbH for the automotive industry, it is a multi-master system, with software identifiable nodes. A simple 2-wire bus is all that is necessary at the physical layer, the

standard includes sophisticated error checking and is capable of data rates greater than 1Mbps.

The controller boards themselves feature a 16-bit DSP (Digital Signal Processor) from Texas Instruments as the onboard processor. This chip contains a CAN module, as well as a range of other features reducing the number of peripherals needed. Some that are necessary include current and motor position sensing, motor drive circuitry, motor protection, and some interfacing chips to the CAN and SCI busses. The board is powered by two supply rails, a 42V unregulated rail which directly powers the motors, and a 7.2V rail which is regulated to 5V on each board and powers all digital circuitry. All of these components of the boards are described in more detail below.

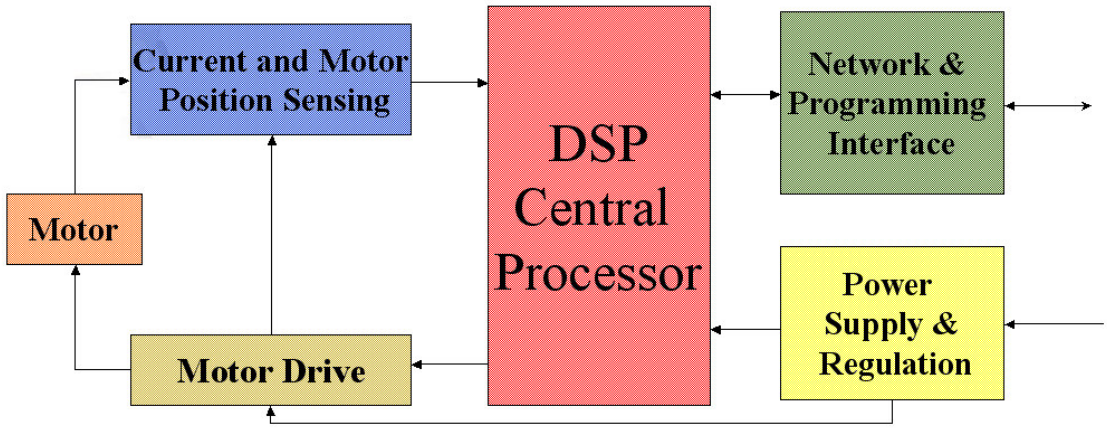


Figure 7-block diagram of controller boards

2.2- DSP

The primary requirement for the processing unit on the Joint Controller was that it must be able to cycle a PI (proportional + integral) control algorithm fast enough to maintain 'good' control of the motor. This limit of what was 'good' was to be based on previous experiments done.

The benchmark was set by Kennedy on the Puma arm. Kennedy chose a 16-bit DSP with an Event Manager containing PWM generation and quadrature decoding done outside the

core of the processor. The encoders in that system had a resolution of 1000 counts per revolution of the motors shaft, which through the 68:1 gearbox resulted in 0.0053º/revolution of the drive shaft. The motors chosen for this robot [10] come with encoders with 2000 counts on the drive shaft and a 156:1 ratio gearbox, and so 0.00115º/rev of resolution. The position therefore of any joint in this system is 4.5 times that of the Puma Arm. The precision obtained in that experiment and more particularly Lillywhite's in 2000 [11] was found to be completely acceptable, and the joints moved at much higher speeds than the GuRoo's ever will. The control algorithm on the Puma arm looped at 20kHz (the same rate as the PWM frequency) though it only controlled 1 motor. It was determined that such precision would not be necessary on the GuRoo. The higher precision of the motor encoders meant that a control loop of 2kHz would be acceptable on a similar processing core (16-bit encoder variables, 16-bit PWM resolution, and low overhead of PWM generation or feedback). It was considered that any processor that could operate with such a refresh rate would be able to sufficiently control the GuRoo's joints.

Other necessary functions of the processor are:

- Analogue to Digital converter
- In-circuit programming capability
- CAN interface module

An 8-bit processor was considered too slow to complete this task. While many exist which have in-chip PWM functions and CAN capability, none are fast enough to consider for the resolution specified.

As mentioned, a great advantage of the distributed control system is that low-end processors can be used and networked to achieve good control. For that reason, no high-end processors (such as those found in modern PC's) were considered seriously.

Which left 16-bit and 32-bit options, of which there are many designs dedicated to motor control. Of the most popular of these – produced by TI, Motorola, and Analog Devices – Stirzaker and Cartwright specified the Motorola 68376 as the most favourable choice in terms of functionality. That chip features 32-bit architecture with onboard TPU (Time Processing Unit) with which 16 pins can be individually programmed to use its capture register (to decode quadrature pulses) or compare register (for PWM generation). The 68376 also has the desired ADC and communication modules. This choice of processor

would have been able to implement the control loop at speeds well beyond the specifications decided upon.

Due to cost and lack of availability, the Motorola 68376 was decided against. Instead, the Texas Instruments TMS320F243 was chosen. It is a 16-bit processor and part of the same family as the chip used on the PUMA arm controller. The difference between the F241 (on the PUMA arm) and the F243 (for the GuRoo) is the inclusion of an external memory interface. This was necessary in order to interface to the external quadrature decoders. The chip contains the following important features:

- 20 MIPS
- CAN module built in
- Serial Communications Interface (SCI) module
- An Event manager which has:
    - 2 x general purpose timers
    - 3 x capture units, 2 with Quadrature decoding
    - 3 x 16 bit full compare units with programmable deadband
- External Memory Interface
- 8 x 10 bit ADC

See Appendix D for further detail.

The event manager which can generate the PWM waveforms and decode a motor rotation was a decisive factor in choosing this DSP.

2.3- Power

2.3.1-The Batteries

There are two power rails supplied to the DC motor boards. These come directly from the two types of batteries used. Firstly, there are large red packs of NiMH cells with a total 42V out, a legacy o the UQ SunShark team. These are connected in parallel to power the motors. There is space in the GuRoo's torso to accommodate 4 of these Battery packs, though only 2 are necessary to supply the power for the period of competition required at the moment, and the saving of 2kg of weight is beneficial. Also are two 7.2 V green RC car batteries. One supplies the power for all digital

chips on all boards. The other supplies the power for the low power servo motors in the GuRoo's upper body.

2.3.2-Power distribution board

All batteries go through a PCB from which all power gets distributed to the boards. The intended setup, and the topic of an undergraduate thesis last year [12] was a converter to minimise losses caused by having a group of mismatched batteries connected in parallel. This intension was never realized, the thesis was inconclusive. The solution was a simple board, with diodes only letting current flow out of them, and 10A fuses running to each output, ensuring that they don't draw too much current.

2.3.3-Local Regulation and protection

The 42V rail is not regulated, it directly powers the high side of the H-bridges. There are fuses on each motor to protect them individually from drawing too much current. The 7.2V rail going to each board is linearly regulated to 5V for all logic chips.

2.3.4-Power Consumption

The DC motor boards each draw about 300mA at 5V. The DC motors are all rated to 4A peak current, 2A continuous consumption, although a typical value is about 1A for the most loaded motors

2.4- H-bridge

2.4.1-Why is an H-bridge necessary?

The only real option for motor drive circuitry is a full bridge switch-mode class D amplifier. The motor power stage must be able to operate in all 4 quadrants of the current-voltage plane, Figure 8, and it must have

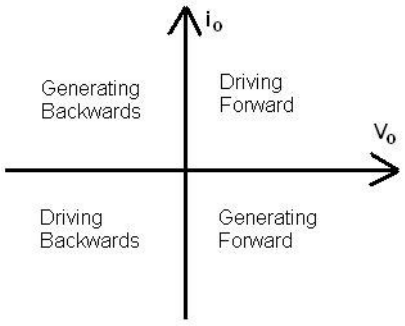


Figure 8-current-voltage plane for h-bridge

a high power output that is precisely controllable. Linear DACs are available, though they are inefficient and bulky.

An H-bridge configuration consists of two legs of two switches in series, from power to ground, with the motor's terminals branching from between the switches (Figure 9). The output (to the motor) is a voltage output which can be controlled in magnitude as well as polarity. Diodes are put in anti-parallel with each switch so that current can constantly flow in the motor, and the power delivered to the motor is directly proportional the voltage applied to it.[13]

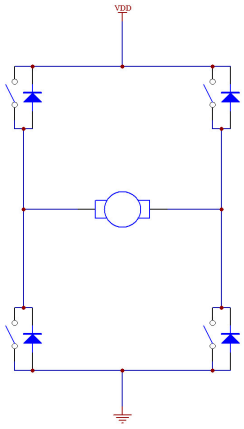


Figure 9-simple H-bridge configuration

The best switches for this type of circuit are Metal Oxide Semiconductor Field Effect Transistors (MOSFETS) used in their saturation region. They turn on and off very quickly and can comfortably handle the power required here.

2.4.2-Switching techniques

Of course it should never occur that both high and low switches on the same leg are on. The four states seen below are all practical states for the switches to be in.

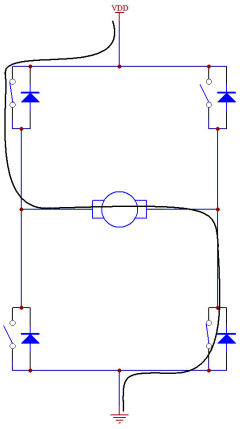


Figure 13 - state 1 – left high and low right on – positive voltage applied across motor

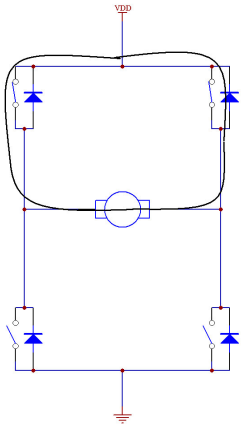


Figure 12 - state 2 – left high and right high on – motor shorted (braked) on high side

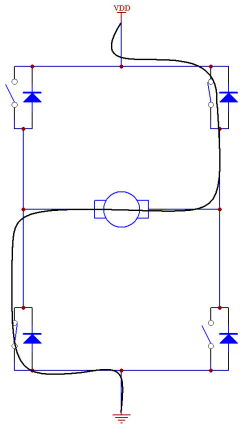


Figure 11 - state 3 – left low and right high on – negative voltage applied across motor

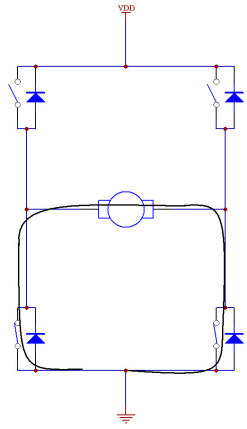


Figure 10 - state 4 – left low and right low on – motor shorted (braked) on low side.

It is also possible to turn off both switches on one leg. Dubbed 'enable switching,' this method destroys the linear relationships that otherwise exist in motor control by taking the motor out of constant conduction mode.

The two most common techniques for accurately controlling the H-bridge are called bi-polar and uni-polar switching, there is a third method discussed called one-phase chopping, it is a variation of unipolar switching.

Bi-polar switching is the easiest to implement (Figure 14), all that is necessary is a control signal and an inverter. State 1 is invoked for the positive time of the duty cycle, and State 3 for the rest of the period. In this method, neither state 2, nor state 4 are ever used. In order to achieve an effective zero voltage applied across the motor, the controller switches with a 50% duty ratio. To drive the motor forward then, the ratio of on time is increased, and vice versa.

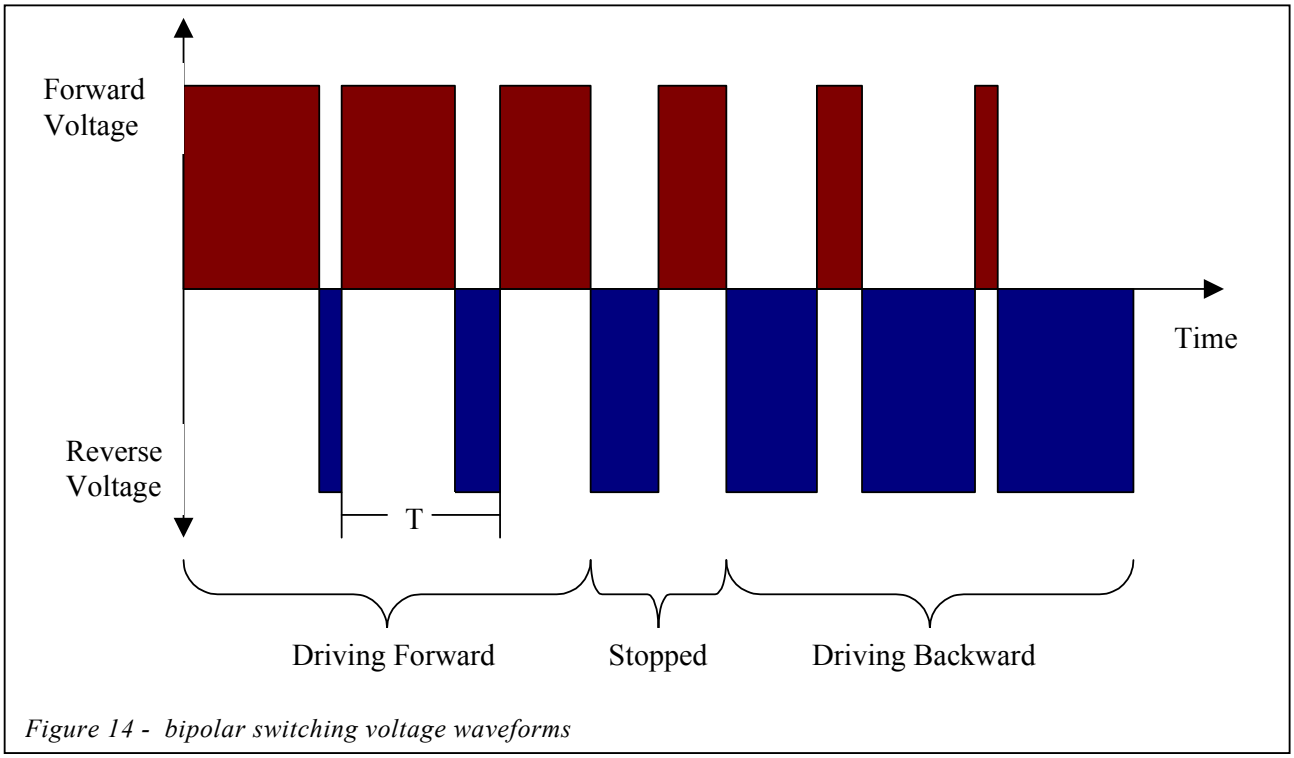


Figure 14 - bipolar switching voltage waveforms

Unipolar switching makes full use of all 4 states above (Figure 15). On one leg of the h-bridge, one switch is kept closed. The other leg then switches between its high and low side with a duty ratio directly proportional to the power to be delivered to the motor. To change the direction of the voltage applied, legs swap roles: the one which was switching now holds one FET on, and the other side starts switching. To continue switching like this is called one-phase chopping, not true unipolar switching, it is the same concept, though slightly less efficient. In one-phase chopping, there is always a MOSFET which is not being used. True unipolar switching involves regularly swapping the legs' roles, while still keeping the same potential across the motor. For example, if the right low switch were held on, and the left leg switched with duty ratio X, then this would regularly be swapped to the top left switch being on, and the right leg switched with ratio 1-X.

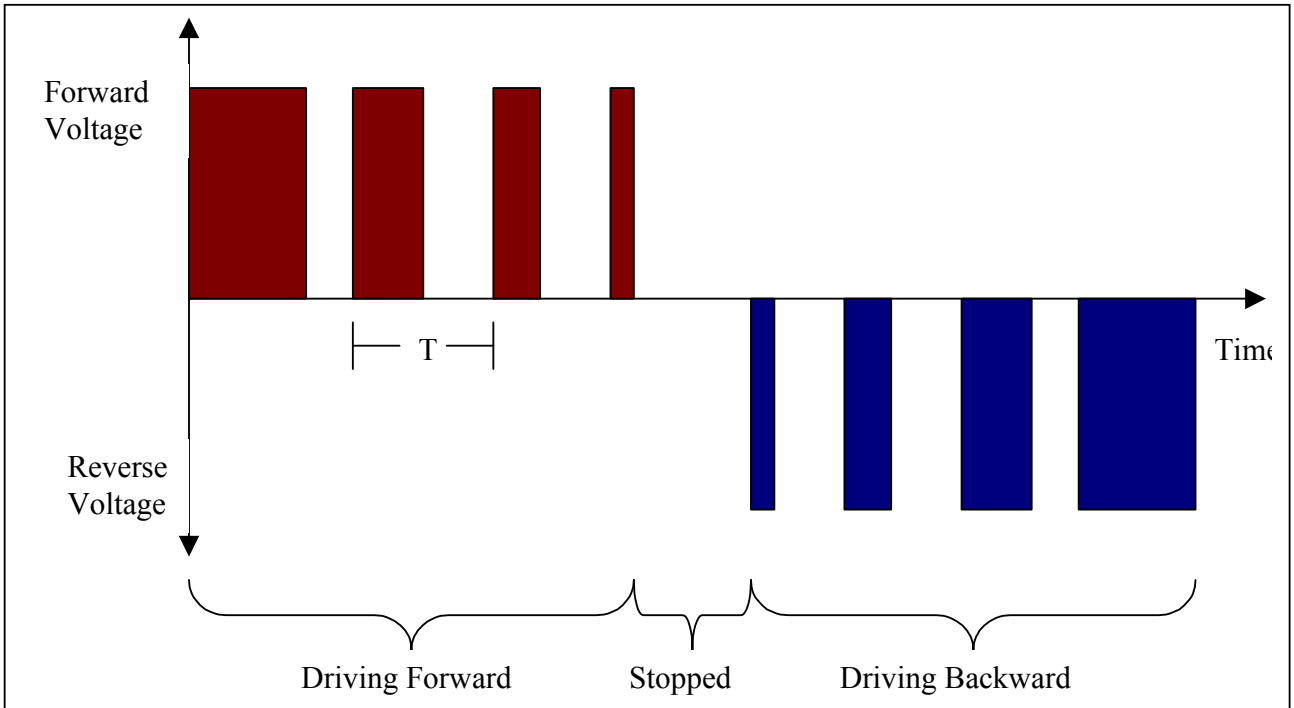


Figure 15 - unipolar switching voltage waveforms

2.4.3-Methods of Implementation

As outlined in [8] and[6], the choice of H-bridge design involves selecting one of:

- Integrated H-bridge package
- Semi-discrete solution
- Fully discrete design

	Integrated	Semi-Discrete	Fully discrete
Involves	• Choosing appropriate chip • Connecting correctly	• Choosing MOSFETS • Choosing MOSFET driver chip • Laying out circuitry conscientiously to optimise efficiency and safety	• Choosing MOSFETS • Designing driver circuitry • Laying out all circuitry efficiently and correctly
Advantages	• Very easy • Very small footprint required	• More efficient FETS can be chosen • Driver chips are common and cheap • Much greater control of switches	• Complete control of all parts of the circuit • Most efficient solution if properly designed
Disadvantages	• Often less efficient than other solutions • Often not able to handle large currents. • Can force Bi-polar switching method (described in section Switching methods)	• Requires more board space than integrated solution • More heat-sinking required for individual MOSFETS • Another current sensory circuit is needed (see section H-bridge design)	• Most space required • Entire driving design required • Same as semi-discrete solution

With effective MOSFET driver chips so cheaply and readily available, the fully discrete option need not be considered for this application.

Stirzaker chose the ST l6203 fully integrated h-bridge solution for the first generation of control boards for the GuRoo. This was because of a predicted lack of space problem. The l6203s can handle up to 4A continuous current, so they just meet specifications required, and they come in a very convenient 11 pin multi-watt package which requires minimal board space, and allows for easy heat-sinking. The chips were mounted along one width of the controller board, and it was initially planned to use the GuRoo's frame itself to dissipate wasted energy (this never eventuated, other heat-sinks were built and mounted). Heat-sinks were required as the losses in the H-bridge were quite high (See section 2.4.4 Power Loss).
Because of the design of this package as well, bipolar switching was required.

2.4.4- Power loss

While MOSFETs are very power efficient, there exists loss in both while the switch is on (conducting), and during switching.

Switching losses can be calculated, defined as:

$$P_{SW} = W_{C(on)} + W_{C(off)} = \tfrac{1}{2} V_{in} I_{out} f_s (t_{C(on)} + t_{C(off)})$$

where:

$P_{SW} = Switching\ power\ loss$

$V_{in} = input\ voltage$

$I_{out} = current\ flowing\ through\ MOSFET$

$f_s = switching\ frequency$

$t_{C(on)} = time\ for\ switch\ to\ fully\ open\ (V_{DS}\ has\ dropped\ to\ lowest\ value)$

$t_{C(off)} = time\ for\ switch\ to\ fully\ close\ (I_{out}\ has\ dropped\ to\ zero)$

Conduction losses are simply

$$P_{ON} = I^2 R * 2$$

Where:

$P_{ON} = Conduction\ \ power\ \ loss$

$I = Current\ \ through\ \ H - Bridge$

$R = On\ \ resista nce\ \ of\ \ MOSFET$

times 2, *the number of MOSFETS conducting at any time*

The power losses in the l6203 are mainly conduction losses, the result of the high R_{DS} (0.3Ω nominal).

These are just the losses in the MOSFETS, there are also losses in the motor itself. These losses are increased by high current ripple from the driving stage. In fact, when GuRoo was first powered, the losses through the motor were so high that within minutes they became too hot to touch.

The power dissipated as heat is found by:

$$P_{loss} = I^2 R$$

where,

$P_{loss} = the\ power\ wasted\ in\ the\ coil\ of\ the\ motor$

$I = the\ current\ ripple$

$R = the\ effective\ impedance\ of\ the\ motor$

the current ripple is found using:

$$V = L\frac{dI}{dT}$$

which, as long as the motor always conducts current (the case here), can be written

$$V = L\frac{\Delta I}{\Delta t}$$

where

$V = effective\ voltage\ across\ motor$

$L = the\ inducta nce\ of\ the\ motor$

$\Delta t = T_{ON}\ or\ T_{OFF}\ of\ duty\ cycle$

$\Delta I = ripple\ current\ amplitude$

As mentioned, the motors in the humanoid were dissipating more heat then was safe for them. Obviously, the impedance of the motor could not be changed, and so the ripple current needed to be minimised. The voltage across the motor was effectively fixed. This was about 80V because of the bipolar switching method used. The value of Δt could not be reduced past 10μs (100kHz), as this was the maximum switching

speed specified for the l6203 h-bridges. The only thing that could be changed was the inductance. Extra inductors were purchased at the suggestion of Dr Wyeth, doubling the motors inductance when in series with it. This halved the current ripple and the motors ran perfectly cold.

2.4.5-Motor protection

There are two cases that are important to consider when designing motor and motor drive circuit protection. The first is when the motor is being over driven, more power is being delivered than the motor's windings were designed to handle. The second is driving circuitry failure, in which a MOSFET, or FET driver begins behaving unexpectedly and could cause damage to the motor or other circuitry.

The first is compensated for in software. Through the current sensing circuitry described below, the amount of power being delivered to the motors is monitored. As the current approaches a dangerous value, the PWM values are limited to limit current to through the motor. In case this software-implemented safety is for any reason insufficient, a hardware mechanism is also in place. The !PDPINT pin of the DSP automatically shuts down all PWM outputs within 12ns of becoming active (low). It is triggered by any of three comparators NORed, the comparators have as their inputs the amplified current sense voltages and a tuneable pot.

The second case is harder to catch. The most likely device to fail is a MOSFET, if its channel is blown open, the switch will be permanently on. Switching the other FET on the same leg as one which is damaged shorts power to ground, and will very quickly damage other components, including batteries. The fuse in series with the power should blow, immediately cutting power. However fuses can be slow to react, and the sensing mechanisms discussed below can be used for fast reactions. Sudden changes in current can be handled in software to disable motor drives (though at the moment they are not) and the hardware interrupt should act quickly enough to avoid excessive damage to the batteries. However, given the integrated package of the current H-bridge, the entire device would need replacing if an internal component fails. Thus far none have.

2.4.6- Current Sensing

It is important for a motor controller to monitor how much power is being put into the motor. The information can be used for safety and component protection (as is the case at the moment), or for global feedback for efficiency monitoring, or even used to help efficiently control the motor. A simple and effective way to measure the current is to place a low-ohmic, high power resistor in between the lower MOSFETS and ground.

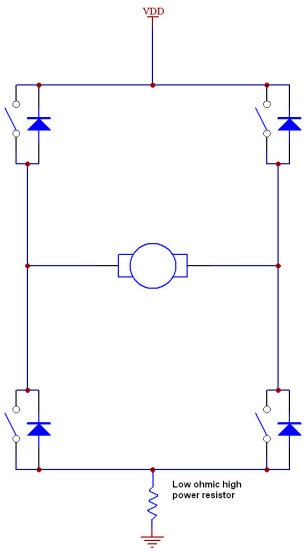


Figure 16 - h-bridge, showing location of sense resistor

The small voltage across the resistor then is amplified, fed into the processor's ADC and appropriately scaled in software to the current being drawn. With this information, the processor can then limit the current going through the motor so that the motor is not always driven at its maximum rating.

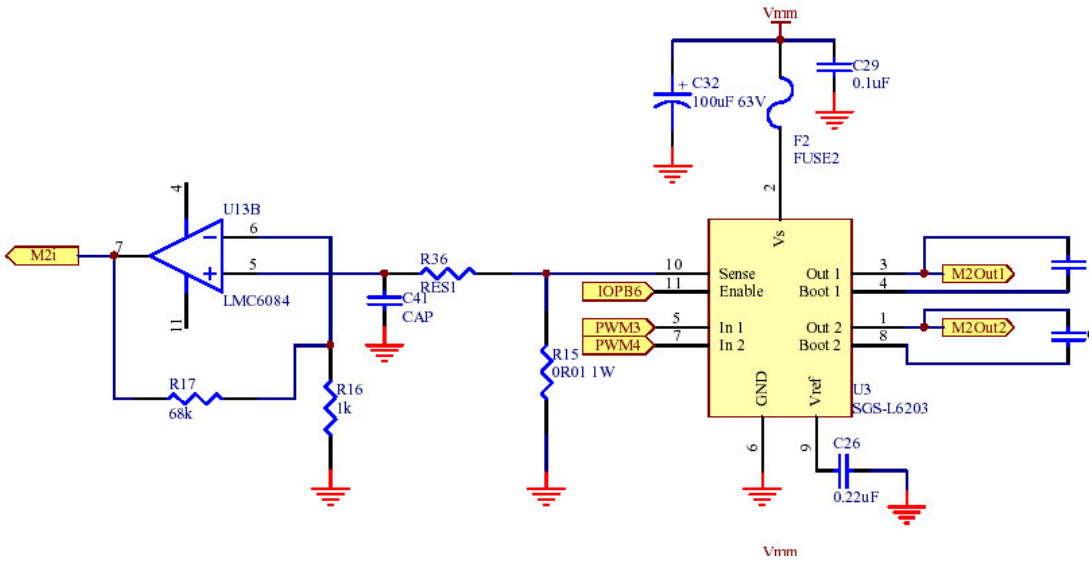


Figure 17 - Motor drive and current sense circuitry on 2001 board design

The TMS chip chosen has an interrupt pin dedicated to shutting down motor power. This is ideal for this sort of situation as analogue to digital conversion can be quite slow. The amplified voltage from the sense resistor is then compared (in the LM2901 comparator) to a voltage set by a potentiometer, and the three results of the comparator are NORed. If the sense voltage (corresponding to the current through

the motor) is ever greater than that set by the pot, the !PDPINT interrupt occurs and shuts down all PWM outputs on the Event manager within 12ns. This feature would be most useful in the case of an H-bridge failure.

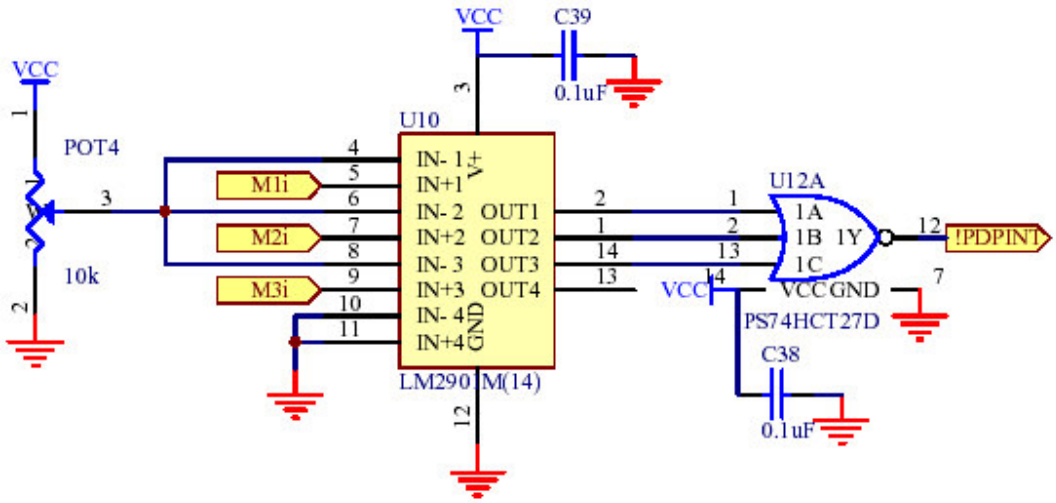


Figure 18 - tuneable limit for hardware interrupt

2.5- External Decoders

In order to accurately monitor the position of each joints position, HEDS5540 quadrature encoders from HP are attached to the shaft of each motor. These encoders output 2 square waves, 90 degrees out of phase and an index which occurs once every revolution. By determining which wave is leading, direction of the motor is determined, and by counting the pulses, relative position is also gained.

The DSPs event manager includes a quadrature decoding module, however with three motors per board, two more decoders are required. The HP HCTL2016 was chosen to do this job. A 16 pin chip, it decodes the 2 pulses into a 16-bit number which it then places on an 8 bit bus. The high byte of data is accessed by asserting the SEL pin. The output can be latched by disserting the !OE pin. This chip must have an external clock to run. The clock has a maximum frequency of 14MHz. As this is slower than the clock frequency of the TMS chip, a T flip-flop was used to halve the output clock speed of the DSP.

The interface to these two decoders is through the external memory bus. The lower byte of the 16-bit data bus on the TMS connects to the quadrature decoder chip for motor 2 of each board, and the high byte connects to the chip for motor 3. The SEL pin is connected to the A0 pin of the external memory address bus, so to access the high bytes of the counter from the decoder, the TMS chip must read from external memory address 0x0001. When an external memory is made, the !IS pin is automatically asserted low, which latches the number in the decoder.

2.6- Networking and Programming Interfaces

There are 3 interfaces from other devices to the TMS DSP on the motor controller board.

The CAN, through which the boards can talk to each other is very simple. The transmit and receive of the CAN module in the TMS connects to the transceiver chip, the PCA82C250 from Philips, which converts the ground referenced signals into a differential 2 wire bus. There are two 2-pin connectors per board, so that the bus might be daisy-chained. There is also a 120Ω resistor and a jumper in series between the CANH and CANL pins of the bus, this is to terminate the bus. Only the last board in the daisy chain will have this jumper shorted.

The SCI module is used to program the DSP, and also help in debugging during development. The onboard interface is simply an 8 pin header, the transceiver and its surrounding circuitry were put on a separate board to save space.

The JTAG (IEEE standard 1149.1) header was included on the controller board, as it was known (from Kennedy's experiences in 1999) that the serial bootloader on the TMS chip was frequently corrupted. The TMS chip can be completely programmed through the JTAGGER, though the peripherals are expensive so usually localised to one computer. The JTAG interface is simply a 14 pin header with appropriate pins connected.

2.7- Other Inclusions on 2001 Controller

Limit switches were envisaged to be put on each joint of the GuRoo so that if confused about where his limbs are, he could never drive two parts of his frame

against each other and damage himself. The six connectors for these are fed into an external interrupt. The use of these was never implemented.

Connectors also for temperature sensors on the motors, and pressure sensors on his feet were included. The pressure sensors run straight into the ADC, while the three temperature sensors are MUXed and amplified before also getting connected to an ADC input.

A reset pushbutton is connected to a MAX811 reset chip, which ensures that the reset line of the TMS chip is asserted for an appropriate length of time.

Another pushbutton is connected to an external interrupt pin. This was very useful in debugging.

LEDs indicate power at both the 42V and the 5V rails as well as three connected to general IO pins, to help with debugging.

3- IMPLEMENTATION OF SOFTWARE FOR 2001 BOARD

From the undertaking of this thesis until the Robocup competition in July, all work was concentrated on bringing the existing boards to a functional state. The boards were designed by Stirzaker[6] and Cartwright[7] in 2001. The capabilities of the boards was not known originally, though the design was sound, having been based on Kennedy's work[8] in 1999 on the puma arm project.

Much of the software, or skeletons of the software to be written had been done last year. The focus of the work to be completed before July was to be able to safely run a control loop. Hardware changes would be unavoidable last resorts to what could not be fixed to a tolerable level in software. To run a control loop the processor had to be able to read accurately read the quadrature encoders on each of the three motors, as well as pulse each of those motors individually to a changing duty ratio. The controller had to be able to cycle through this loop and update the PWM according to the velocity the motor should be turning at several hundred times per second. Another requirement, though not essential for the control loop was the operation of the analogue to digital converter to read the current flowing through each motor, in order to limit the power delivered to it, to fail in a graceful manner.

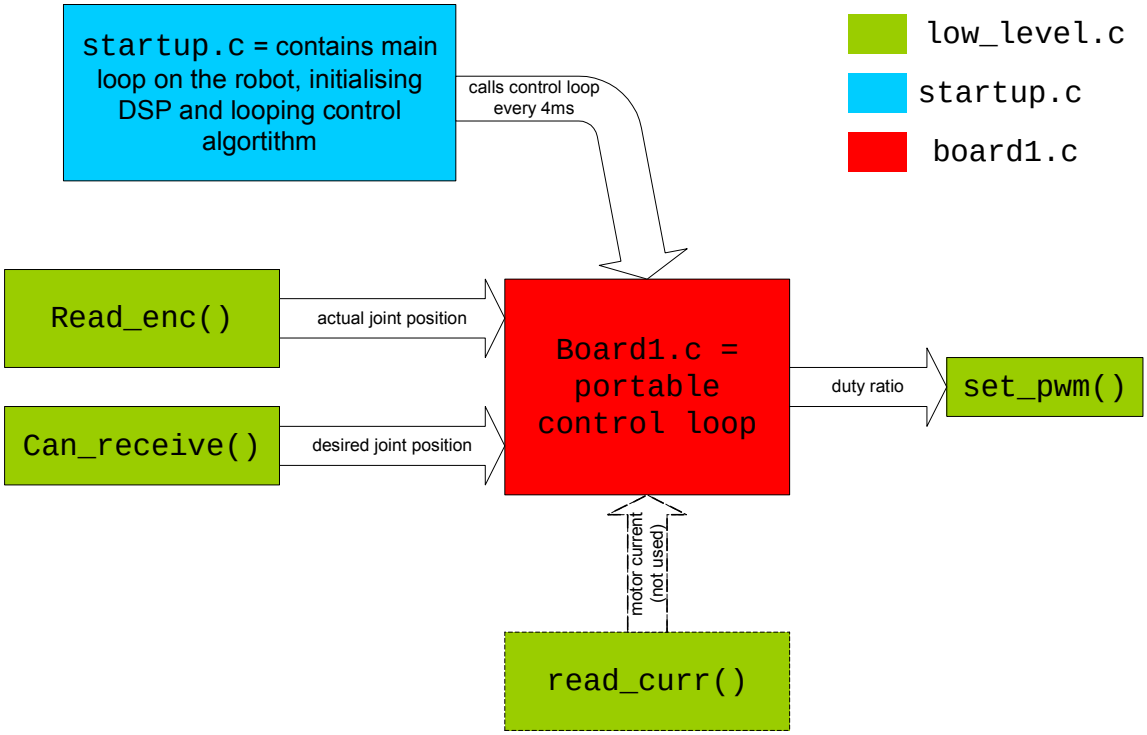
The software that runs on the controller boards for the GuRoo is a PI (Proportional plus Integral) control loop operating on the velocities of each joint. This is the equivalent of PD (Proportional plus Derivative) control on the joint's position, it ensures that the speed that the motor is going is as close as possible to the desired velocity. For more information, see [14].

The control loop itself, and the tuning of the gains for each joint was the responsibility of Drury. The low-level sensor reading and motor actuation was a topic of this thesis.

3.1- Structure

All control software is written to be portable between the robot itself and the simulator used to test on. `boardd1.c` contains the actual control loop, and its associated header files contain the gains associated with each joint. On the robot, `boardd1.c` is called

from `startup.c`, which contains the main loop. The main loop begins by initialising the board by setting the software selectable pins to the appropriate functions, and setting up the internal registers to perform the operations desired of them. It then runs and loops the control algorithm. It is the functions in `boards1.c` that then call back to `low_level.c,` which contains the following methods.



3.2- set_pwm

The PWM is setup to operate at 100KHz by setting the T2PR register to 200 (20Mhz/100kHz = 200). This gives the user a range of 200 values to set the duty ratio. Then it is a simple matter of setting the CMPRx register associated with each motor. This was the original implementation of set_pwm, it was passed a motor number and a duty ratio between –100 and +100, which was then scaled between 0 and 200 and the CMPRx register was set with that.

It was found however, that there was not enough resolution with which to set individual values of the pwm. The solution to this problem of resolution was to 'feather' the pwm values. 'Feathering' was a term coined by Dr Wyeth in the spur of

the moment, though assumed by his dutiful undergraduate students to be the correct word for the concept described below, which was implemented by Adam Drury and Damien Kee.

The control loop (@250Hz) operates on each desired velocity 400 times before the desired velocities are refreshed from the CAN bus. In order to get more than 200 values between 0 and 100%, a number between 0 and 216 is passed to `set_pwm()`, the bottom 4 bits of the 16-bit number are masked off, and for a proportional number of the 400 cycles, the CMPRx register is set to a value 1 greater than its base number. This gives 16 bits more resolution to `set_pwm()`.

3.3- read_curr

A simple matter of correctly configuring the ADC of the DSP correctly, reading converted values and appropriately scaling them. A large amount of time was spent debugging this system before it was found that the VrefHI and VrefLO had been connected backwards on the board. When this hardware problem was fixed, the simple conversion and scaling is indeed all that this function did.

While it worked flawlessly in testing, the function gave some very strange results when used on the actual robot. It was found, though not till after the competition in July, that these strange results were caused by a flaw in the communication method to the computer.

Because of these believed inaccurate readings from the ADC, another method was necessary in order to limit the current delivered to the motors at high load. Dr Wyeth devised a very clever plan. A limit was set on the maximum PWM value that could be used to drive the motors based on the battery voltage, maximum safe current, and back-EMF (based on the current speed of the motor).

$$PWM\_LIMIT = IR - K_V \omega$$

where:

I = maximum safe current

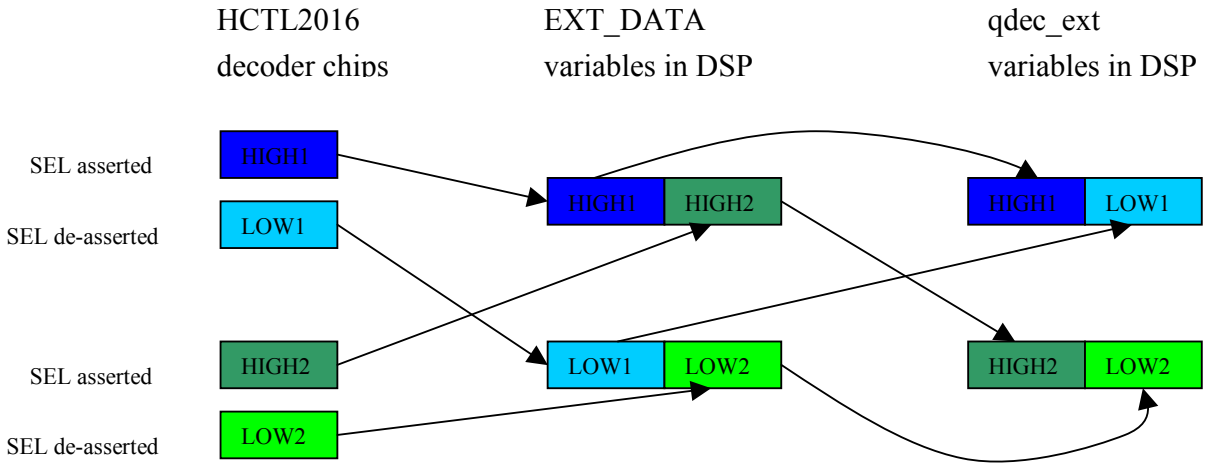
R = Armature resistance of motor

K_V *= back-emf constant*

ω *= measured angular velocity of the motor*

3.4- read_enc

This function simply gets passed an int relating to a motor number, it then reads in the 16-bit numbers of the three quad decoders (one internal and two external) and returns the position of the motor number passed to it. Two intermediate variables are used, into which the 2 external memory reads are loaded (High data bytes when SEL/A0 is asserted, and low data bytes when de-asserted). The data is then split back up and re-arranged back into two 16-bit numbers relating to each motor. A case statement then returns the appropriate number to the calling method.



There was a considerable amount of trouble experienced when trying to access the external memory bus. While the assembler instructions were known, the compiler had an indirect syntax to address the port on this processor, and ignored in-line assembler calls in the C code.

These external memory reads are most certainly the bottleneck of the processor's code execution. While they obviously cannot be eliminated, the code can be further optimized.

A more efficient algorithm was written and tested – though later lost due to an unwittingly reformatted computer – which significantly reduced the delays inherent in the external memory reads. This algorithm eliminated the redundant memory accesses that occur every time the function is called. `Read_enc()` became a simple

case statement returning the now global variables `qdec_ext2, qdec_ext3`, and of course the timer counter associated with the internal decoder. A new function was created which read the data from the memory bus in, and stored them in global variables, this had to be run before `read_enc()`.

The new setup reduced the bottleneck by nearly 2/3ds. Rather than being limited to a maximum speed of 700Hz (actually running at 250Hz), the control loop could now refresh at speeds up to 1.5kHz.

The new revision, as described, is a very simple alteration to the existing code. The control gains were never re-tuned to the new speed because of lack of necessity. There was little point in re-writing the method after it was lost, as all work was concentrated on the hardware re-design, which would not require external memory access.

4- BOARD PLACEMENT AND LAYOUT

4.1- Board locations

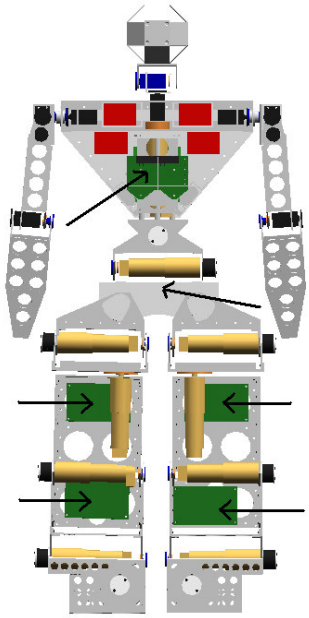


Figure 19-Controller Board locations

- The current design has the 6 controller boards distributed as follows:
- 2 X 1 board in each shin controlling both ankle joints and knee joint
- 2 X 1 board in each thigh controlling all three hip joints
- 1 board in the stomach controlling all three waist joints
- 1 board in the back controlling all 8 servo motors

4.2- Wiring

- The wiring harness consists of:
- A pair of 42V power cables running to each board
- A pair of 7.2V power cables running to each board
- A small CAN pair daisy-chained from board to board
- Short cables running from each board to the motor terminals and encoder outputs

4.3- Board Layout

Figure 20-2001 board populated

The board layout was simple, with power electronics along the left side of Figure 20, the h-bridge packages along the edge of the board, for heat-sinking. Digital circuitry is on the right side of the board. 7.2V power comes into the bottom right of the board, and the same connector type delivers power to each motor on the left. The large, black connector in the center of the board is 42V in. The encoder and CAN headers are at the top of the board, and SCI and JTAG headers on the right.

5- RESULTS AND SYNOPSIS OF OLD SYSTEM

5.1- GuRoo's success

By the beginning of April 2002, the 2001 design of the joint controllers was functional. The control and low-level software had been written and the hardware modification for the ADC had been discovered. By the end of April, all boards had been built, programmed and tested, the GuRoo took his first steps. In June, GuRoo went to RoboCup 2002 to compete in the first humanoid league competition for RoboCup. Unfortunately, he did not win. He placed 7th out of 10 in both the walking competition and the freestyle – in which he stood on one leg and waved to the crowd. Those robots that rated higher were all smaller machines, and the focus of development on them was walking stability, something that was a two-week rush job on this project. GuRoo had some problems in Japan, the source of which was the USB to CAN interface box which had been bought (this has since been fixed by replacing USB with SCI and using a controller board as the interface). The joint controllers worked well.

Figure 21-GuRoo, midstep, coiming towards the photographer

5.2- Board Design

The fundamental idea behind this control system – local controller boards distributed throughout the robot, communicating on a network – worked very well. The network protocol chosen too – CAN – was very successful. The component of the system that required re-working was the actual controller boards.

The control loop was not able to operate at its intended speed. Rather than the 2kHz expected, the control algorithm – which took 1.28ms to execute – was given 4ms (so @250Hz) in which to do so. This allowed room for additional, possibly time-consuming functions to be included (such as serial feedback to a computer).

The feathering function was also a waste of processing time, if this could be eliminated – while maintaining the PWM resolution – the control loop could run much faster.

Apart from the fact that it was not capable of running at the desired speed and it required a slight hardware modification, the 2001 design of the controller board functioned as it was designed to. The joint control was adequate, and the robot was capable of walking.

There exists slackness in the control loop still: this has to do both with the control algorithm (tuning of gains and lack of full PID) and the speed at which the control runs. However, it was discovered that this 'soft' control was in fact fundamental in the success of the gait implementation on the robot (see paper in appendix E). The control at the time of writing operates on the desired velocities of each joint compared to the actual velocities, with no consideration of efficiency of motion. This is somewhat crude control, a much better solution would be to operate on both the velocities and the power used driving the motors. In order to do that, more processing power is required. Whether that extra power be used to increase the speed of the control loop, or to do extra current sensing and related calculations is for future generations of GuRoo developers to decide.

The bulky heatsinks and additional high-current inductors on the motor drive circuitry were an unexpected, though necessary addition to the robot's weight. The efficiency of this system was a large concern and was to be addressed in the new revision of the boards.

5.3- Hole in Current Design

A major problem in testing walking gaits on the GuRoo was that the initial position from which he starts was never set. Each boot-up the power would be applied while on his stand, after which GuRoo was placed on the ground and each joint was adjusted (by eye) to a rough starting position. The original plan during the design was to use the mechanical stops on each joint, either by driving each joint a certain direction slowly until it found a limit switch, or by folding and twisting the robot to its limits before applying power, then driving back to a set relative position. In fact, this second technique is currently used on the ankle joints of the robot. However, to contort the entire robot so that each joint is at a stop is completely impractical, as well as potentially damaging to the robot. A new system is absolutely essential to the progress of the GuRoo's development.

5.4- Implementation of Design

The positioning of components on the 2001 controller boards was quite poor (Figure 20). A large amount of space was wasted, despite the fact that the justifications for many component choices were for a lack of space. There were very few useful test-points, nor a convenient place on which to clip test-leads onto the ground of the board. The DSP was difficult to reach on two sides because of the proximity of high profile components. Also, the motor power headers were not in sequential order, making it easy to connect motors incorrectly (no major mishaps occurred). The lack of debugging LEDs made software development difficult sometimes.

The connectors chosen were another problem. Some connectors chosen were only slightly smaller than standard IDC headers, though they were considerably harder to find, difficult to use because of a locking mechanism, and hard to crimp.

The wiring harness was quite large, requiring two pairs of separate power wires for each board.

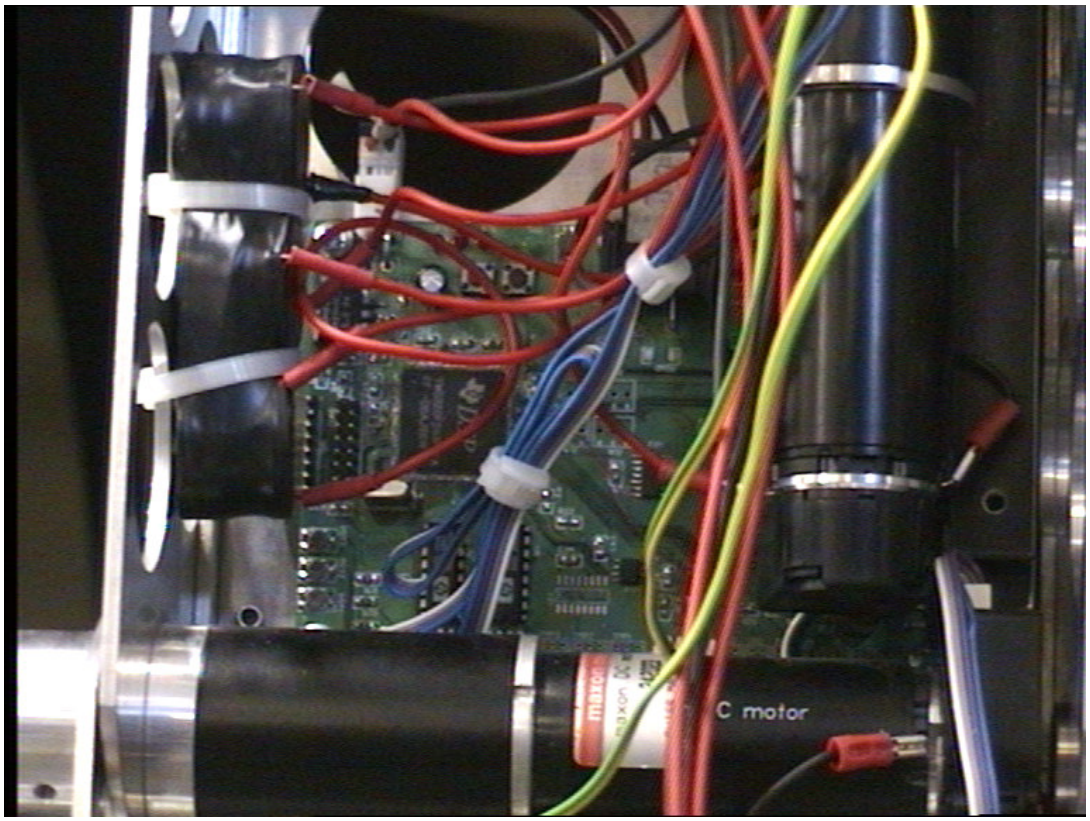


Figure 22 - 2001 board mounted in Left Thigh

PART B

CREATING A NEW DESIGN OF THE CONTROLLER HARDWARE, AND A DISCUSSION OF THE SOFTWARE TO BE WRITTEN TO OPERATE IT.

6- NEW DESIGN

6.1- Design Specifications

In order to correct the faults mentioned previously, a new design of controller boards was necessary. The major issues which needed addressing were:

- The control loop must be able to run faster – at least at the original specification of 2kHz
- A mechanism for accurately aligning each joint must be put in place
- The supply of the motor's power should be made more efficient – efficient enough to do away with the bulky heat-sinks currently required

In order to accomplish these tasks, many options were considered. The second two problems (alignment and motor power supply) could be tackled individually, they both have standard interfaces to any sort of processor. Different processor arrangements could change the arrangement of the entire system though, so this was decided first.

6.2- CPU

As mentioned, the major bottleneck of the system in terms of speed is the external memory accesses require by the DSP at the moment.

e

A simple upgrade of processor (to TI's next generation of 40MHz chips, or to Motorola's or AD's comparable chips) was a possibility, to keep the same configuration, but cycle through instructions faster. This implementation would have met specifications (though hardly exceeded them), in terms of speed, however the bottle neck would remain on the external memory bus. Also, if any more processing power were ever desired of the controller (for more complex control, or to offload some intelligence processing onto these boards), the power would not be available.

A novel idea was to step away from a sequential processor to run the control loop. Programming an FPGA or CPLD using VHDL to decode the quad encoded waves, calculate errors and implement PWM cycles would be a very viable option. The operation of logic devices like an FPGA is very fast as many calculations are

performed in parallel. Such a system is theoretically infinitely expandable to control many motors with large enough gate arrays, and so a good generic design could be used in a number of distributed control environments. A sequential processor would still be required to do the analogue to digital conversion and interpret CAN messages, though a much simpler, cheaper chip could be used. This design was not chosen, mostly because of lack of time and necessity. Unless the distributed control system is abandoned (at this point, there is no point) the joint controllers will never have to operate more than 3 motors, so expandability was not a requirement, nor is the increase in speed justifiable in terms of advantages.

Another option then is to choose a processor with enough dedicated quad decoders such that an external memory interface is not necessary. The Motorola 68376 chosen by Stirzaker and Cartwright is an option, though for the same reasons of cost and availability that it was rejected last year, it was not chosen. Instead, 16-bit DSPs with dedicated quad decoders were focused upon. Of these, there are none that have 3 decoders, many from Motorola and TI have two. To keep the same arrangement of controller boards incorporating these chips, the third quad encoder input could have been connected to 2 external interrupt pins, and the decoding done in an interrupt routine. However this would not have eliminated the overhead of quad decoding, and some doubts were raised about the processors ability to do this decoding without compromising its ability to receive all CAN messages and operate the control loop. The decision then to only operate two motors off of the local control boards was made.

Of those available, the Motorola DSP56F8XX family, the Motorola HC12 microcontrollers and the TI TMS320LF24XX series DSPs made the shortlist. While the Motorola chips were faster and had more flash available, the TI chip was preferred for its higher RAM capacity, easy availability and the design experience with them.

The TMS320LF2406a's full specifications are given in appendices D and F, however some of the most important improvements from the F243 are:

- 40 MIPS operation
- low power 3.3V design
- Same core design, thus fully code compatible with TMS320F243
- 2.5k of 16-bit words of RAM

- 32k of 16-bit words of flash memory
- boot ROM
- 2 event manager modules

6.3- H-bridge

In order to improve efficiency of the Class D motor power stage, it was necessary to eliminate power loss both while ON and during switching.

The only way to reduce power while the switch in ON (in saturation), is by reducing the resistance of the MOSFET.

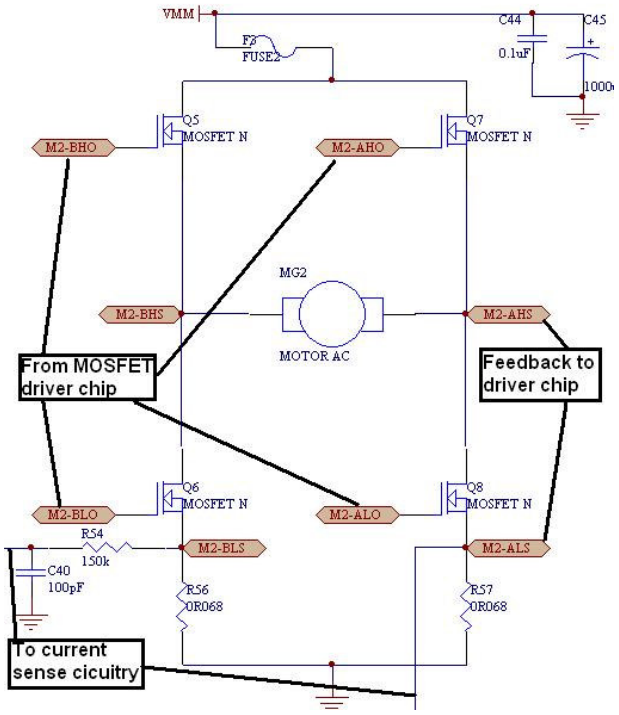


Figure 23-Semi discrete h-bridge design

It is believed that conduction losses account for most of the wastage through the h-bridges. However, switching losses do play a part. The easiest way to reduce switching losses is by reducing the switching frequency. Indeed for software reasons explained later, this would be beneficial in terms of code efficiency. The single disadvantage of reducing switching frequency is if it were to affect the ripple current and thus power loss in the motor windings.

In order to solve the problems of loss through the motors, the ripple current must be decreased.

The solution to all of these improvements is to move from bi-polar switching to uni-polar or a similar method (why it isn't pure unipolar is explained in section 2.4.3 Current sense) and to discard the integrated h-bridge package

6.3.1-MOSFET selection

The main requirement of the MOSFET to be chosen were that it have a very low R_{DS}, and be able to handle up to 60V impulses, and avalanche current. It should also have a method of being heat-sunk onto the PCB, to dissipate what energy is wasted back into the board.

The IRFIZ44VSs a NMOS field effect transistor was chosen from International Rectifier. It has an ON resistance (R_{DS}) of 0.0165Ω, and a reverse breakdown voltage (V_{DSS}) of 60V. It comes in a convenient D2Pak package, which can dissipate 2W of energy.

The matching PMOS transistors for this choice had a considerably higher R_{DS}, it was then desirable (as previous designs indicated) to use NMOS transistors on both the high and low sides of the h-bridge. A MOSFET driver chip was then necessary to driver the high side of the h-bridge.

6.3.2-MOSFET driver selection

NMOS transistors as switches turn on when a positive voltage is applied from the gate to the source of the FET. For the FETs chosen, the threshold of switching is 4V, with a maximum limit of 20V. In order to apply this positive voltage to the high side of the h-bridge, a driver chip provides this bootstrap differential. Often the chips come with additional functionality such as internal inverters and dead-time generators so that only two signals are required (right and left legs) and shoot-through along a leg cannot occur.

The driver chip should be able provide a high enough bootstrap voltage for the FETs chosen, as well as able to control each leg of the bridge independently. It needs to be able to supply enough current to charge the G-S capacitance of the FET at switching frequencies up to 100kHz. No internal inverters are necessary, as the DSP has complementary outputs on its PWM pins, though they may aid in PCB routing, so they are desirable. Nor is dead-band generation necessary as the DSP can do it.

There were three readily available sources of these driver chips. ST (SGS-Thomson), Intersil(previously Harris), and IR (International Rectifier). The majority of these were either built for 20V or 600V systems. Many of those for 600V were appropriate for this situation.

All of those from ST were designed for 5V systems, and required 3.6V as a minimum High Level Input Voltage. They would have required external interface circuitry and were not significantly better than the other options to consider. IR have some very nice small package options, the IR2104S {IR datasheet} in particular. They require very little external circuitry. An 8 pin SOIC packages contains a half bridge (so two would be required per h-bridge) with internal inverter, internal dead-band, and a shutdown pin. Unfortunately, this choice could only source 130mA, which at 100kHz would be only just adequate to drive the gate of the FETs at 100kHz.

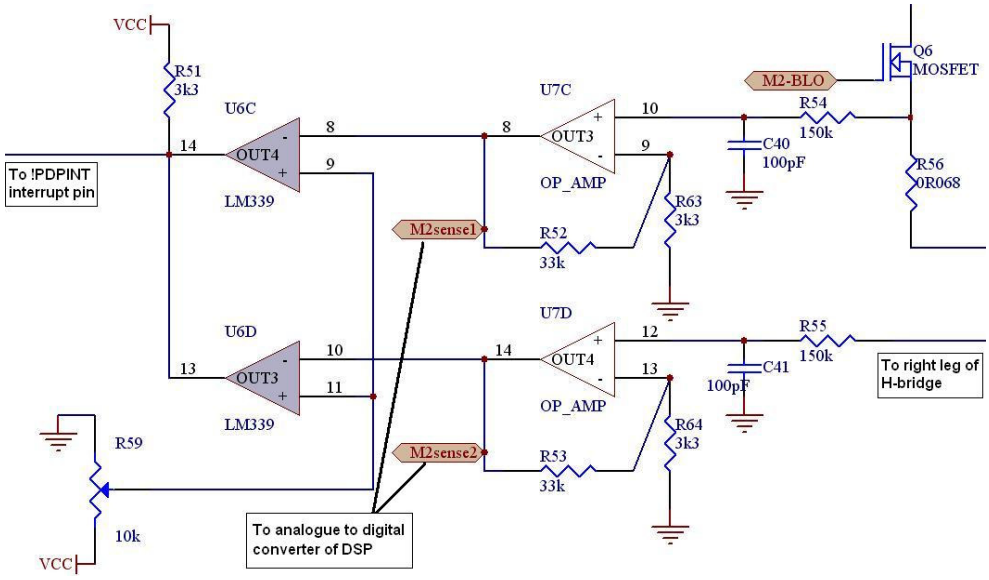
The best option for the MOSFET driver was the HIP4081 by Intersil {hip chip data sheet}. The same chip as used by Kennedy on the Puma arm, it is ideal for Class D amplifiers like this one. It accepts all 4 signals (no internal inverters) though it does have a user settable dead-band. It accepts 3.3V logic signals and is more than capable of driving the MOSFET gates. It also has a disable pin which can be used to float the motors, to move them manually while power is still applied to the robot. Its only disadvantage is the size it requires, both IC footprint and external circuitry requirements.

6.3.3-Current sensing and Motor Protection

The mechanisms in place for motor protection were deemed sufficient. No dangerous case could be thought of which the old design could not handle. As long as the potentiometer to set the level of !PDPINT interrupt is tuned, the batteries, motors and MOSFETs should be safe.

Setup

Current sensing in a semi discrete h-bridge is somewhat more convoluted than the previous design. Whereas previously current always flowed through the h-bridge from top to bottom, the unipolar switching method to be employed allows current to recirculate. When the motor is ‘shorted’ (either both top and both bottom switches are on) the current does not flow out through ground, instead, the current stored in the motor coil loops back through these switches. Therefore the current in each leg is sensed individually. To interpret the power drawn, the current through each leg subtracted from each other. The polarity of that number indicates current direction.



Note that only current sense circuitry is included on the bottom half of the h-bridge. In order to accurately measure power consumption, true unipolar switching cannot be used. State 2 in section 2.4.2 Switching Techniques will not register any current being drawn from the sensors. This does not matter in terms of safety to the motor and circuitry, as that state only occurs briefly and is not dangerous, and any failure will still be recorded. However the use of torque control to efficiently control the motors would be affected. If future development on GuRoo were to include torque control, only one-phase chopping (discussed in section 2.4.2switching Techniques**)** could be used. This is not a bad thing – thus no high-side sensing circuitry – but if a motor were to always be driving in one direction, one FET would be unused.

Component Selection

The circuitry itself involves: a MAX4094 op-amp to scale the voltages read from the sense resistor for the ADC and a LM339 comparator with a tuneable potentiometer to set a current threshold.

The MAX4094 was chosen for the following features, in order of consideration:

- Ability to be powered from a single 3.3V rail
- Low offset voltage – 1.4mV
- Rail to Rail output voltages
- Low input voltage noise – 12nV/rt.Hz
- Low supply current – 660nA

The 3.3V rail was chosen to power the amplifier. While the 12V rail was available, many cheap op-amps can be supplied by a single low voltage rail and as long as they have rail-to-rail outputs, are fine for this application. The low offset voltage was important because if it were too large it could distort the result of the low voltages being amplified. Factors such as high slew rate were not mandatory as it would only be beneficial in the case of a catastrophic failure of a MOSFET, when power will be short circuited to ground, the fuse is the only protection of this case. There are many pin-compatible op-amps with the MAX4094, if a cheaper or more effective alternative is found, it can always replace this choice.

The comparator, the LM339 was chosen primarily for its open collector output. Other features desired were:

- 3.3V operation
- low propagation delay
- low offset voltage
- low power consumption

The open-collector output of the comparator was important to eliminate the logic OR gate necessary on last year's design.

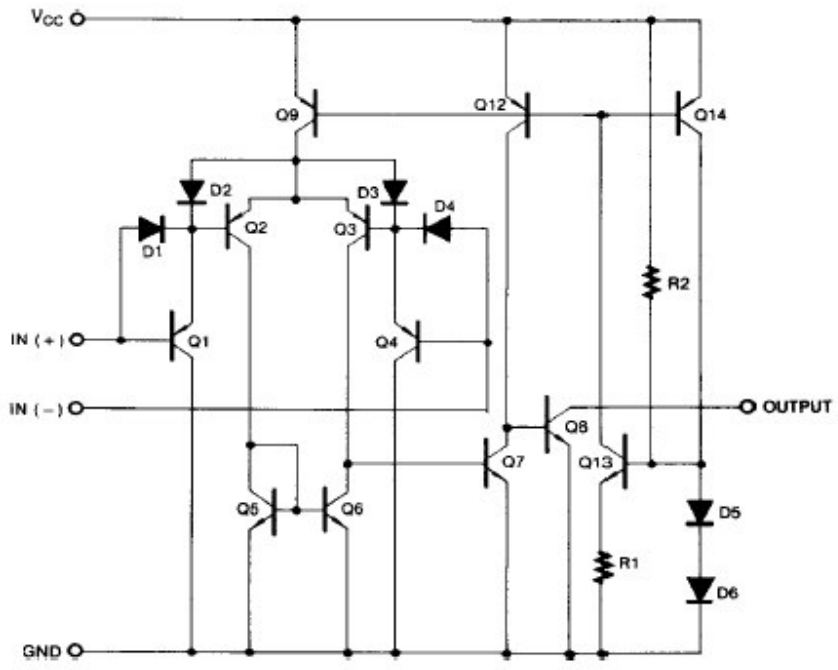


Figure 24- internal diagram of LM339 comparator, note the open collector output

The two comparators are ORed by tying their outputs together, the open-collector output with a pull-up resistors ensures that power and ground are not shorted when either comparator becomes active.

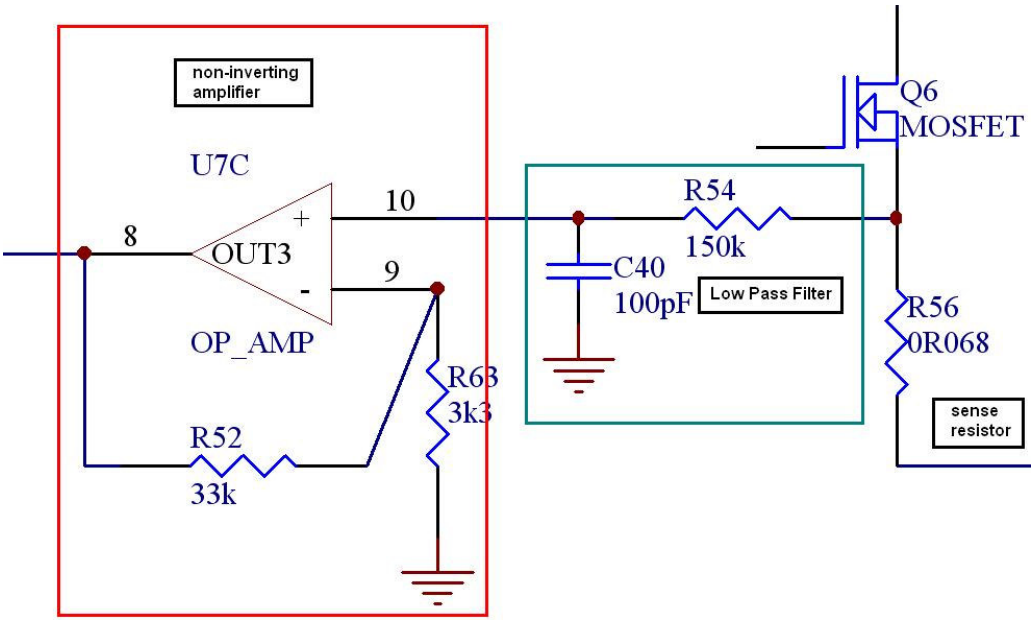
The open-collector output comparators have a much higher propagation delay (500ns versus 4.5ns) however it was calculated that any delay under 1us was acceptable. Given the slew rate of the op amp, the delay inherent in the DSP, and the 500ns delay through this comparator, the current would be able to rise up to 0.14A above the threshold set by the pot. This is well below anything damaging to the batteries or other drive circuitry.

Discrete Device Calculations

The sense resistors should have as large a value as possible, though still able to handle the power flowing through them. 1 Watt resistors are readily available with a large range of low-ohmic values. Kennedy[8] used two 0.047 Ω resistors in series to have a large resistance and high power capability, as he was unsure of the average power through the PUMA's motors, and was worried that the resistors might heat up considerably during normal operation. Knowing a rough average

current of 1A for the most strained motors in this system, any ohmic value which can pass the maximum current of 4Amps should keep sufficiently cool at normal operation. A 0.068Ω, 1W resistor was chosen.

The low pass filter between the sense resistor and the op amp is to ensure that noise associated with the MOSFET switching is not measured. As it is possible that in future, the PWM frequency may be as low as 20kHz, the cutoff should be set at about 10kHz. The break frequency is defined by $f_b = \frac{1}{2\pi RC}$, choosing the common value of 100pF for the capacitor and solving for R gives a good high value of 150kΩ, this high impedance will help block high frequency current flowing out of the through the capacitor.



The op-amps scale the voltage across this small resistor to values between 0 and 3.3V. The voltage across the 0.068Ω resistor with 4A passing through it is calculated by:

$V = I * R$

$V = 4 * 0.068$

$V = 0.272$

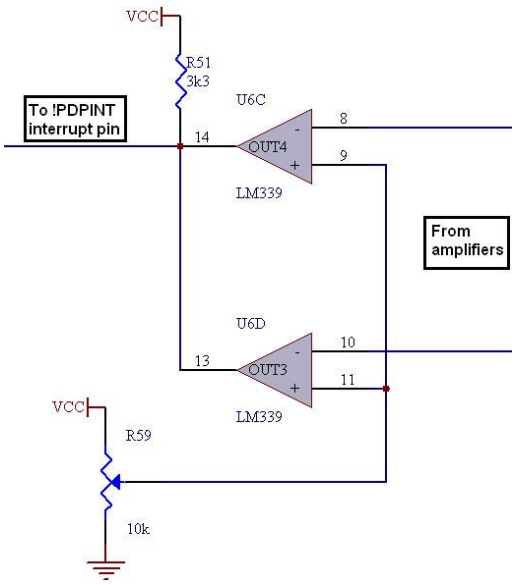
Scaling this to 3.3V requires a gain of ~12.

The resistor values of the amplifier then are calculated by:

$$A_v = 1 + \frac{R_{52}}{R_{63}}$$

$$R_{52} = 11 * R_{63}$$

choosing standard component values from the E24 series, 33kΩ and 3.3kΩ, a gain of 11 is close enough.



Both comparators for each motor are set to the same threshold by tying their V+ input to the swipe of a potentiometer. The V- input is connected to the scaled voltage from the op-amp. When this value is greater than the value on the swiper of the pot, the output of the comparator drops to 0V, and the !PDPINT interrupt is triggered.

6.4- Power

6.4.1- Requirements

The new chosen DSP requires a 3.3V power supply for both the core and the I/O of the chip. The MOSFET driver chip, the hip4081 requires a positive rail between 10 and 20V, nominally 12.

6.4.2- Possible Solutions

Adopting the existing system, and linearly regulating from 7.2V to the now required 3.3V would be a large waste of power.
7.2 - 3.3 = 3.9
3.9V * 0.5A (conceivable current drawn by boards) = 1.95W wasted.

This is considerable waste of power, and a more efficient solution was sought.

The power distribution board in the robot's back could have been modified to regulate power to these fixed voltage levels. The wires running from that board to each joint controller would have been subject to a considerable amount of noise

from the motors, and the inductance in these wires would have generated strange effects which could have been difficult to filter.

Local SMPS (switch-mode power supply) regulators are readily available, from many companies, including maxim, national, and linear technologies (do not be fooled by their name). These converters, in either: buck (regulate down), boost (step up), or buck-boost (either up or down and inverting) configurations commonly have power efficiency close to 95%.

With the 7.2V and 42V rails available, two main options were considered as to where the board's power could be drawn from. The first would be to keep the existing arrangement in which all board power is drawn from the 7.2V batteries. This would involve a buck converter from 7.2V to 3.3V, and a boost to 12V. A much more elegant solution however was to do away with the 7.2V battery packs, thereby saving 0.5kg of weight in the GuRoo. Both 12V and 3.3V would be drawn from the 42V battery packs, as well as the motor power itself. Only 1 power connector is then required, so a smaller board could be made.

To regulate from 42V to 12V and 3.3V, two options were considered. Firstly, a flyback controller topology with two windings on the output coil could do the job. Alternatively, two cascaded buck converters could step from 42V to 12V first, and then from 12V to 3.3V. The second option was favoured as the isolation offered by the first was not needed and the voltage output precision was easier to obtain in the cascaded buck converters.

6.4.3- Chosen

Telecom SMPS chips were found to be ideal for this application. Normally used to supply low-power circuitry off of the 48V telephone lines, they efficiently regulate down to low values with little output ripple.

The LT1676 SMPS chip was chosen from Linear Technologies. It can supply up to 700mA, has the output voltage selectable, and has very little output ripple.

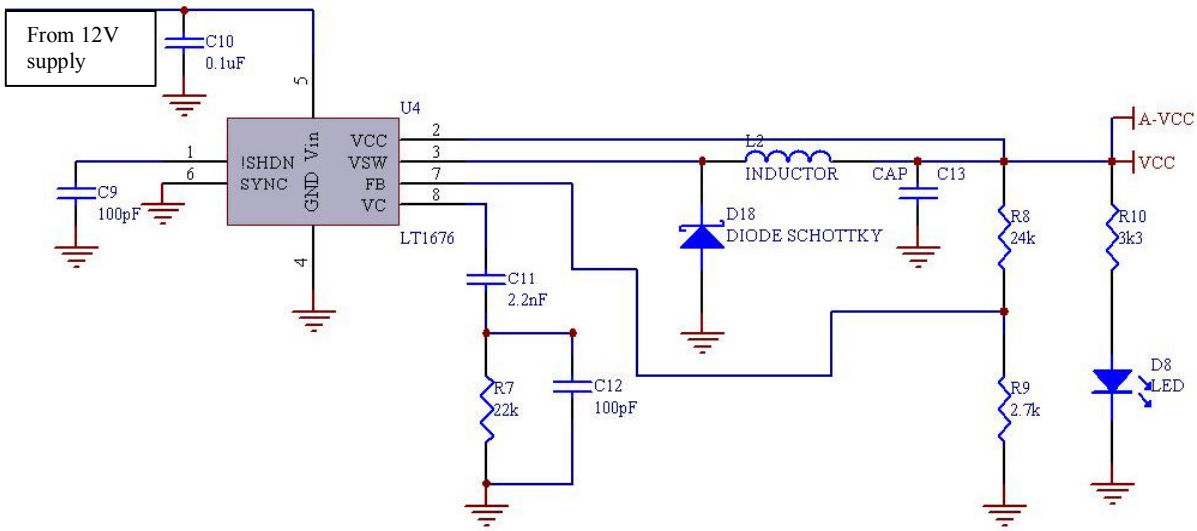


Figure 25- Connections for SMPS regulator, shown is 12V regulated to 3.3V

6.4.4-An Amendment – 5V regulator

After nearly the entire 3.3V and 12V system had been designed, it was discovered that the DSP required a 5V source to power VCCP when programming flash. The pin required a maximum of 15mA so a small package was chosen to do the job. The lp2985 comes in a 5-pin SOP-23 package, and can source up to 150mA.

6.5- Alignment sensors

When the robot is initialised (when power is applied) the joint could be in any position. It is, of course, desirable that absolute position control be achieved. Indeed, it is essential for any calculations of reverse kinematics (which are not used now but will be in future projects) that these values be accurate. It was specified then, as a requirement of the new system that a method of alignment be devised.

A method that required no additional hardware involved keeping the quadrature decoders powered at all times. A small battery mounted on each PCB, and making use of the shutdown feature in the DSP chosen (or similar modes in the FPGAs considered) could have kept a running count for a period of about three

days. A simple recharging circuit and a central plug-pack input could have kept a continuous running count of the joints' positions. The initialisation of this design would have involved locking each joint against a mechanical stop and driving a measured distance to its desired initial placement. While this calibration would need doing only once when the controller boards were in place, each time they were re-programmed, or left un powered for more than 3 days, the joint would need to be reset, such a solution was not practical in a testing environment.

A neat solution would be to mount a potentiometer on each joint. The swiper of the pot would then be fed into the ADC of the DSP. When starting each time, the controller would know how far away, and in which direction, each joint is from its desired initial position. A simple control algorithm could even be run to return the robot to that position in the most efficient and accurate manner. Great resolution from the potentiometer need not be essential if the analogue input is coupled with the index pulse from the quad encoders. This I-pulse occurs once per revolution of the motor, which equates to 2.3 degrees of revolution on the drive shaft. Once the analogue reading was within a certain parameter, the algorithm could drive to the next index pulse. Care would have been necessary to ensure that the thresholds decided upon did not have the error margin associated with the pot across an I-pulse, if so, the robots joints could occasionally be out by 2.3 degrees. The fatal problem with this solution was the mounting of the pot. The physical coupling methods were inconceivable on more than half of the joints in the lower body.

The favoured solution, though still not easy to build, is an array of three optical switches for each joint. One switch would be for the alignment of the joint, the other two would act as limit switches at each extension of the motion.

The switches would involve an Infra-Red LED shining through a tiny (perhaps 0.5mm) hole on one plane of a joint, and a corresponding hole on the other plane, through which is a photosensitive device. The holes must be aligned such that no LED can trigger the wrong sensor. However, for ease of wiring, it may be desirable to keep the trigger holes close together. It is envisaged that the wiring will involve a small PCB – drawing power from the controller board – containing the photodiodes or phototransistors, as well as an amplifier, a comparator and a logical OR gate. From this board the LEDS would be powered through a resistor – either each 3mm LED on its own lead, or a PCB of all three SMD LEDs – depending on each joint's layout.

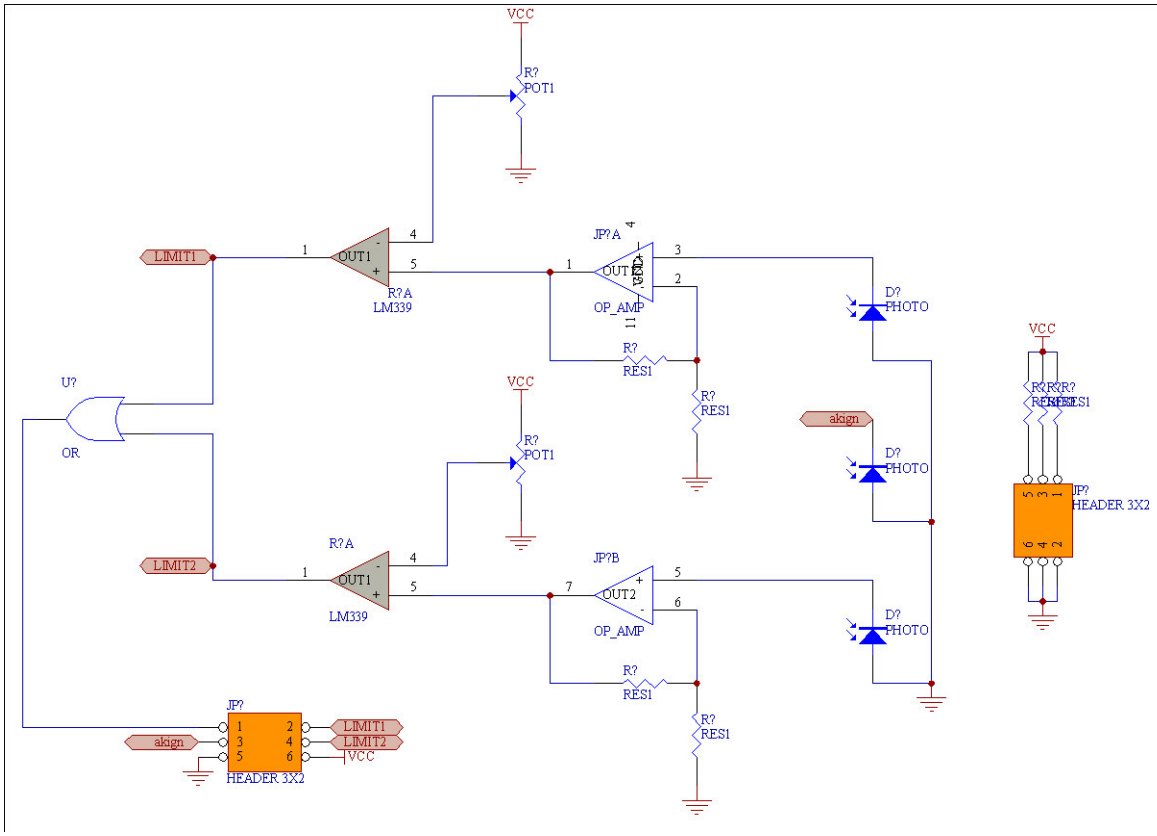


Figure 26 - schematic of propsed opto-switch board

The connection to the controller boards is a 6-pin header including VCC and ground to power the off-board circuitry. The output of the OR gate is connected to the XINT2 pin, to trigger an interrupt in the DSP. The individual outputs of the

comparators are connected to I/O pins so that the interrupt subroutine can determine which switch triggered it, and the output of the alignment sensor is connected through an amplifier to an ADC input.

The reason for the use of an on-board amplifier and the ADC is threefold. Firstly, it solved a problem concerned with a lack of I/O pins. Secondly, much finer tuning of a threshold can be achieved in the DSP than by tweaking a potentiometer. Given the possibility of error due to different ambient light conditions, the use of the index pulse of the quad encoders is used. Thirdly, if a coupling method for a potentiometer is found, it can be integrated into the design of this board without any hardware reworking. The amplifier may be biased to have unity gain, and the potentiometer's readings used to re-align the joints.

6.6- Choice of other chips and their interfaces

6.6.1-CAN

Very few options for 3.3V CAN transceivers are available. The only two found were the MAX3053 and the TI SN65HVD230. The MAX3053 was, at the time of writing, only available from Maxim itself, therefore not considered. The SN65HVD230 is fully compatible with the DSP chosen, and contains some features that the previous transceiver didn't. Most additional functionality is not useful to this system (such as standby mode). The one additional possible use of the device is high-speed mode. The current out of the Rs pin of the transceiver to ground determines the slew rate of the output waveforms, selectable to help overcome electromagnetic interference caused by the harmonics of fast rise times (for more information, see datasheet in appendix D). A 10k resistor has been placed there to achieve a 15V/us slew rate, though this can be by-passed with a jumper to let the transceiver switch as fast as possible, with no internal limitation on the switching rate. This can let the CAN bus operate at speeds up to 2Mbit/second. While at the moment, the bottleneck in the communication is in the interface from CAN to the central computer, it is foreseeable that the interface be upgraded, this high data rate on the CAN bus may then be used.

6.6.2-SCI

Far more options were available on the SCI transceiver market for 3.3V devices. The MAX3221 was chosen for its availability, small size, low price, and features.

The MAX3221 contains only one transmitter and receiver, which is all that is necessary. It also features complex auto-shutdown internal circuitry, and the pins to disable it, they have been connected to give the DSP complete control of the transceiver's state. Another pin (!INVALID) gives feedback to the DSP about the SCI receiver's input condition, whether it is receiving valid voltage levels.

6.6.3-Reset chip

The MAX811 reset chip holds the reset line low for a specified time (150ms) when triggered, it helps to ensure that the pin on the DSP is held long enough for the DSP to react, and filters out de-bounce from the pushbutton. There was no need to change from the cheap and effective MAX811 which has been used on both previous joint controllers.

6.7- Connectors

Standard locking IDC headers are used for all data connections. In cases in which headers of the same size are used for different purposes (the optical switches and the encoders connect to the board the same way) they have been made pin compatible, so that no damage can be done to either circuitry if the wrong connector is plugged into it.

Power headers from JST, the same as those used to drive the motors on the 2001 design have been chosen again, for both 42V power and motor connectors. They are required to be ordered directly from JST, however they are very small, can handle the currents to be drawn by the board and motors, and can be daisy chained. This means that only one pair of power wires and a pair of CAN wires is necessary down each of GuRoo's legs.

6.8- More Debug info

Other, simple additions to the new board design include:

6.8.1-Dip switches

Each board has a 4-bit hardware selectable board ID. Each board must have a unique ID number. This lets the board know which motors it is controlling, and which CAN messages to pay attention to. Rather than the current system in which each board must be given its own ID in software before being programmed, the board number should be fixed to a board so that completely generic code may be uploaded, regardless of the motors to be controlled. It is possible to specify certain pages of the DSPs flash to not be overwritten during programming, and the ID could be stored here. However, it was decided that DIP switches would make board identification easily identifiable to human users of the system, and that this would help with debugging by being able to very quickly 'hot-swap' boards.

6.8.2- Debug info

An entire byte may now be written to the LED display on the user edge of the board. This will help tremendously in debugging and software development, rather than the 3 LEDs available on the old system. Also included is a row of headers, onto which an oscilloscope probe might be clipped. This can help in the debugging of timing issues within the chip.

A ground clip was also supplied such that the user may safely ground the oscilloscope probe.

A connection for a piezo-sounder has been included on a PWM output of the DSP. This will allow the user to help debug information without the board being visible. For example, while testing a walking gate, the controller can sound that it is approaching its PWM limit, and the user can observe the effects of this, knowing what the controller is doing.

7- NEW SOFTWARE TO BE WRITTEN

The code for the new board has not been written. As much care has been taken during the hardware design of the system to keep the software complexity to a minimum, to have optimal speed at the lowest level. The functions, as they are intended to work, are described below. In the methods that simply require correct registry addressing, the registers necessary are described. When more complicated algorithms are required, flow charts have been provided with an explanation.

7.1- Debug LEDS

Unfortunately, do to the configuration of the I/O ports on the TMS320LF2406a, no entire 8-bit port is available to be used onto which a byte of debug LEDS can be placed.

Instead, the first five bits of Port C, and bits 6 and 7 of Port B form the array. A small routine must be written to map a character passed to it to the appropriate pins.

7.2- Board ID

For the same reason as above, the pins routed to the DIP switches from which the board will determine which motors it controls are not from the same place.

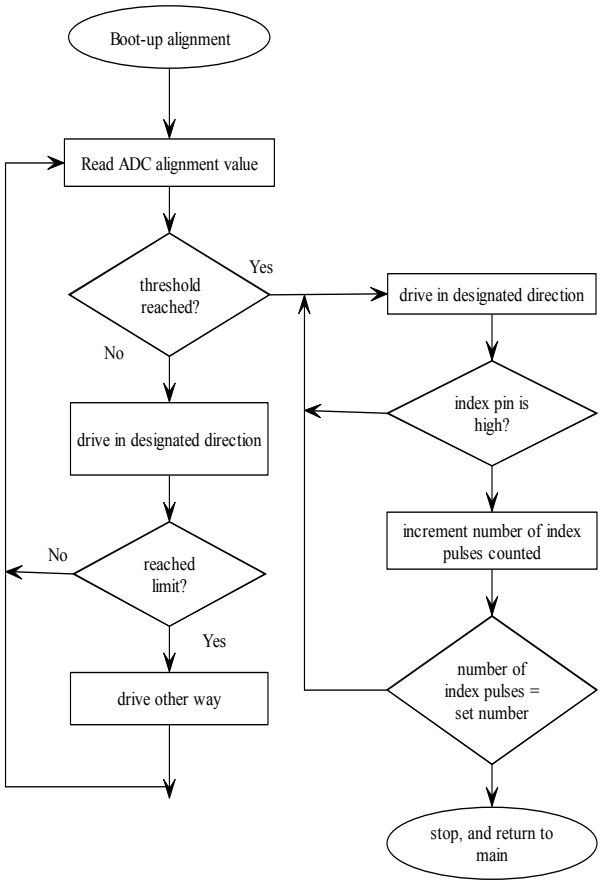
Retrieving the board number need only be a simple mask of those pins on which the dip switches are connected to. This will integrate very simply in to the existing code, by simply #defining the board as the masked variable.

7.3- Alignment

A completely new function must be written which the controller boards run to initialise each joint to a set position each time. The optical switch system described in the hardware section is what is discussed below.

Although the robot boots up from a theoretically unknown position every time generally, he starts from the position in which he hangs naturally on the stand. Care should be taken when deciding on the position of this alignment sensor, that

it is placed a certain direction from where each joint is likely to start, though not too far away. That way, the alignment procedure described below should allow the robot to quickly align itself and be ready to operate.



The algorithm proposed involves first reading the ADC value of the sensor, comparing it to a threshold arbitrarily set for each joint, then running a seek pattern to find the switch if not already aligned. Each joint will have a different seek pattern, dependant on the estimated location of the switch. For example, if it were only possible to place the holes for the light 20 degrees in a certain direction from the robot's hanging position, then that joint would initialise by driving in that direction for, say, 25 degrees before turning back. Another joint may have its switch located very close to the natural position of the joint, and so the switch could be in either direction from where it starts. In that case, the joint should do a brief sweep in both directions, always expanding its scope until the threshold of light is found.

Once the alignment point has been found, it is very unlikely that it itself is the point that the joint should be initialised to, so the motor should drive to that point. It is suggested that the index pulse of the quad encoders be used for this. They are in a fixed position and as they only occur every 2.3 degrees, small errors in the light sensors (due to changing ambient light conditions or small disturbances of the sensor) will not change the alignment position of the joint.

7.4- read_enc() and read_curr()

read_enc() will simply return the TxCNT of the timer associated with that quadrature decoder.

read_curr() will operate exactly as before, simply reading the FIFO registers of the ADC.

7.5- PWM

The new hardware configuration of the motor drive gives the programmer full control of the MOSFETS. The user should be able to switch to any of the 4 states described in section 2.4.2-Switching techniques, as well as be able to float the motors. The full complementary PWM outputs of the DSP are routed to each leg of the h-bridge. The disable pins of both driver chips are connected to an I/O pin of the DSP.

After initial tests have been done to ensure the motors do indeed run cold, the TxPR register should be able to be set to a much larger value, reducing the switching frequency and increasing the resolution of PWM values. This would eliminate the need for the feathering function and further optimise the code.

Implementing one-phase chopping is simple. The CMPRx register associated with Leg B is set to 0, and the Leg A to the duty ratio of the desired power. To drive in the other direction, these roles are swapped.

To do pure unipolar switching however, the duty ratios of the two legs need to be calculated. The desired duty ratio desired should be divided by two, and subtracted from 50% of the TxPR register before being applied to Leg B, then added to 50% of TxPR for Leg A, and applying this number to that CMPRx register. This will result in the same power into the motor as one-phase chopping, but the power losses will be distributed among all of the FETs. Perfectly accurate current sensing is not possible if doing unipolar switching.

If the motors need to be floated, then the pin disabling the MOSFET drivers should be asserted in an external interrupt subroutine. The decoders will continue to count while the motors are moved, which is desirable. The desired joint positions of the motors should be moved with the decoder counts, otherwise when returning from the subroutine the motor will drive back to where it was.

8- BOARD LAYOUT AND PLACEMENT

With 2 motors being controlled per board, there will now be 8 boards to be placed in the GuRoo. The boards were designed with this in mind, Figure 288. Their shape is long, and thin, all short connections (to motors, encoders and opto-switches) come out of the short sides of the board, and all longer connections and interfacing hardware is along the front edge.

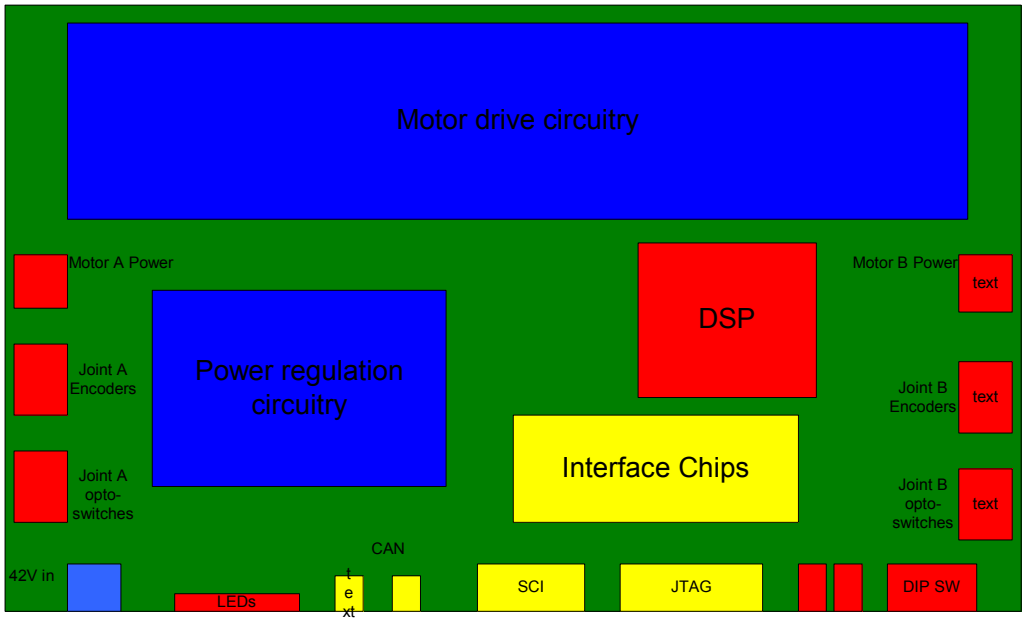


Figure 27-general layout of new controller boards

The intended board locations of the new controllers are:

- Two stacked in the bottom of each shin
- One along the inner thigh
- Two in the stomach
- The servo motor board will remain the same shape, and in the same place

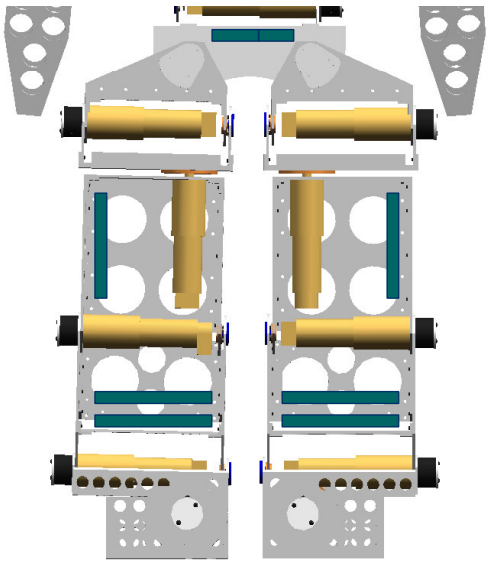


Figure 29-Intended board locations of Controllers

9- CONCLUSION OF PROJECT

The main goal for the humanoid team at the beginning of the year was to get GuRoo to walk. This was achieved by April.

Specific to this thesis, the boards handed down from 2001 in an unknown state of operation were brought to full functionality. In doing so, a complete familiarity with both hardware and software specifics of that design was gained, and used to improve on the current system.

While the new boards are yet to be built, their design is sound, having been based on the old models. Modifications to the design shall make higher level development much more effective, by not being limited by slow and inefficient hardware.

10- FUTURE WORK

This project could go in a number of ways in years to come.

10.1- This design

To finish the implementation of the design in this thesis the following needs to be completed:

- Currently, the PCB has not been completely routed. Components are in place, but the connections between them have not been completed. It is hoped that this will be done before the Innovation Expo, two weeks from the submission of this thesis, so that a complete design may be handed on... to be built in 2003
- The design of a new servo motor controller board has not been completed. There is no need for a new board, as the current model functions perfectly to specifications. A new design could easily be completed, so that all processors in the robot are the same model. The design of this board involves removing all motor drive circuitry from the DC motor board, and replacing it with a 12V buffer chip (HEF40244). 8 independent PWM channels should be routed to the inputs of this buffer. The schematic of the DC motor controllers has been done in such a way that 8 of the 10 independent PWM channels are free for this reason
- The boards must of course all be built and fully tested, and the software described in Chapter 7 written.
- The alignment sensors must be constructed and mounted, as described in section 6.5.

10.2- The Next Generation

- If a new board were to be redesigned in the near future, a strong recommendation is to use the yet-to-be-released TMS320F2810. The latest from TI the 150MHz DSP will have a similar price-tag as the LF2406a's chosen in this thesis. This simple upgrade would simply involve changing the pin connections of the DSP and re-writing the software. Another power supply rail will also be necessary for the 1.8V core. This chip will eliminate any speed concerns as the control loop would most certainly be capable of operating at full 20kHz, the limit if the PWM is cycled at this speed too.

10.3- Another Tack

- An FPGA implementation of the digital distributed control system, as discussed in section 6.2, would be a great new design. If done properly, the same design could be used in many control applications, including the RoboRoos wheeled robot soccer team at UQ.

REFERENCES

1. RoboCup, *RoboCup officail Website*. 2002.
2. honda, *ASIMO page*. 2002, Honda.
3. Sony, *SDR-4X release information*. 2002.
4. Waseda, *Development of Waseda Humanoid*. 2002.
5. JSK Lab, U.o.T., *H7 information page*. 2002.
6. Stirzaker, J., *Design of DC Motor Controllers for a Humanoid Robot*, in *Information Technology and Electrical Engineering*. 2001, University of Queensland: Brisbane.
7. Cartwright, T., *Design and Implementation of Small Scale Joint Controllers for a Humanoid Robot*, in *Information Tecnology and Electrical Engineering*. 2001, University of Queensland: Brisbane.
8. Kennedy, J., *Design and implementation of a Distributed Control System in an Industrial Robot*, in *Computer Science and Electrical Engineering*. 1999, Univeristy of Queensland: Brisbane.
9. GmbH, B., *CAN homepage of Robert Bosch GmbH*.
10. Kee, D., *Design and Simulation of a Humanoid Drive System*, in *Information Technology and Electrical Engineering*. 2001, University of Queensland: Brisbane.
11. Lillywhite, J., *Improving the Performance of the Distributed Digital Control System in an Industrial Robot*, in *Computer science and Electrical engineering*. 2000, University of Queensland: Brisbane.
12. Brewer, N., *Power system for a Humanoid*, in *Information Technology and Electrical Engineering*. 2001, University of Queensland: Brisbane.
13. Mohan, U., Robbins, *Power Electronics*. 2 ed. 1995: Wiley.
14. Drury, A., *Gait Generation & Contorl algorithms for a Humanoid Robot*, in *Informaion Technology and Electrical Engineering*. 2002, University of Queensland: Brisbane.

APPENDIX A – 2001 BOARD SCHEMATIC

Note: The 130R resistor is only for the terminating controller boards. These being boards 1 and 2 - the two boards controlling the ankles and knees.

Title	Joint Controller for 3 motors
Size	A2
Number	A1
Revision	E3
Date:	19-Oct-2001
Sheet	1 of 1
File:	H:\schematics\boards - jetad final.ddb
Drawn By:	

APPENDIX B – 2001 BOARD SOFTWARE

```
/*****************************************************************************
 *     File: startup.c
 *   Author: Adam Drury & Andrew Hood
 *  Project: Robot 1.0
 *  Created: 23 April 2002
 *  Summary: Sets up TMS chip.
 *****************************************************************************/

#include "f24x.h"
#include "startup.h"
#include "ioports.h"
#include "..\Common\board1.h"
#include "can.h"
#include "serial.h"

#define BOARD_ID 3
#define NO_SERIAL

int count=0;
int i = 0;
unsigned int timestamp;

int BOARD;

extern int data[4];

int *p_id ;
int id[4];

/*allows extra resolution for setting PWM values*/
extern int feather_count  = 0;
extern int counter1;
extern int counter2;
extern int counter3;
extern int pwm_high_1 ;
extern int pwm_low_1 ;
extern int pwm_high_2 ;
extern int pwm_low_2 ;
extern int pwm_high_3 ;
extern int pwm_low_3 ;


void main() {

    int count1 = 0;

    /*p_data = &data[0];*/

    get_ID();

    id[0] = BOARD ;
    id[1] = BOARD ;
    id[2] = BOARD ;
    id[3] = BOARD ;
    p_id = &id[0];

    init_board();
    can_transmit(p_id, (unsigned long)((BOARD | 0x20) << 2));    /* transmit the board id numb
er */
    timestamp = 0;

    while(1) {
        if (count >= 400) {
            count = 0;
            feather_count = 0;

            b1_control();

/*          timestamp++;
        /*  data[0] = timestamp;

            /* Put actual velocities in CAN mailboxes */
            /* temp force data */
/*          data[0] = 0x0123 ;
            data[1] = 0x4567 ;
            data[2] = 0x89AB ;
            data[3] = 0xCDEF ;

*/
            CANMCR |= 0x0102 ;      /* Request write access to data field for RTR configed mai
lbox 2*/
            CANMBX2A = data[0];
```

```
            CANMBX2B = data[1];
            CANMBX2C = data[2];
            CANMBX2D = data[3];
            CANMCR &= 0xFEFD ;      /* Clear CDR to allow transmit to occur */
        }
    }
}

void get_ID() {
    /*sets BOARD variable to ID number stored in ?bootloader?*/
    BOARD = BOARD_ID;

    switch (BOARD) {
    case 1:
        LEDON1;
        break;
    case 2:
        LEDON2;
        break;
    case 3:
        LEDON12;
        break;
    case 4:
        LEDON3;
        break;
    case 5:
        LEDON13;
        break;
    default:
        LEDOFF;
        break;
    }
}

void init_board() {
    /*sets timer interrupt to go off every BOARD_DELAY seconds*/
    /*sets up registers and other stuff*/

    WDDISABLE;
    EVIMRA = 0x0080;
    INT_DISABLE;
    IMR = 0x0003;
    PBDATDIR = 0xFFFF;
    OCRA    = 0x1FDF;      /* 00001111 11011011 */
    OCRB    = 0x02FC;       /* setup IO as CAN*/

    scisetup();
    adc_setup();
    qdec_setup();
    pwm_setup();
    cansetup01();
/*  cansetup5(); */
    init_gains();
    init_pos();
    INT_ENABLE;
}

/*sets up the analog to digital converter*/
void adc_setup()
{
    ADCTRL1 = 0xC900;
    ADCTRL2 = 0x4000;
}

/*decoder setup*/
void qdec_setup(){


    /* info   8-66
     * T2CNT  7-19
     * T2PR   7-14
     * T2CMPR 7-15
     * T2CON     8-28
     * CAPCON 8-59
     */

    T2CNT = 0x0000;
    T2PR = 0xFFFF;
    T2CMPR = 0x8FFF;

    /* ++---------------- not affected by emulation suspend
      || ++------------- Directional Up/Down Count mode
```

```
      || ||+++---------- Clock Prescalar = /1
      || ||||| +-------- Use own TENABLE bit
      || ||||| |+------- TENABLE - Enable Timer
       || ||||| ||++----- Clock Source - QEP Circuit
       || ||||| ||||++--- Register reload when counter is 0
       || ||||| ||||||+-- TECMPR - enable timer compare
       || ||||| |||||||+- SELTPR1 - use own period register
       || ||||| ||||||||
      11011000 01110010 */
    T2CON = 0xD872;


    /* +---------------- Clear all capture registers
      |++-------------- Enable QEP Circuit (bits 9, 7-4 ignored - *)
      |||+------------- Disbale CAP3
      |||| +----------- GP Timer selection for cap3   = GP Timer 2
      |||| |+---------- GP Timer selection for cap1/2 = GP Timer 2 - *
      |||| ||+--------- CAP3TOADC = no event
      |||| ||| ++------ CAP1 Edge Detection = Detect both edges - *
      |||| ||| ||++---- CAP2 Edge Detection = Detect both edges - *
      |||| ||| ||||++-- CAP3 Edge Detection = none
      |||| ||| ||||||
      01100000 11110000 */
    CAPCON = 0x60F0;

    outport(WSGR, 0x0BFF);

}

void pwm_setup(void)
{
    /*
     * T1PR             Timer 1 Period Register - 20Mhz/T1PR        - freq
     * CMPRx            Compare Register x                        - duty
     * OCRA  7-4    I/O Mux Control Register A                      - set pwm pins
     * ACTR  8-37   Compare Action Control Register               - active high
     * DBTCON 8-41  Dead Band Timer Control Register                - dead band
     * COMCON 8-36 Compare Control Register                       - pwm
     * T1CON     8-28   GP Timer Control Register (individual)      - up, prescale
     * GPTCON 8-30 GP Timer Control Register                      - up, pol
     */

    /* Set period register for 100kHz = 20MHz/200 */
    T1PR = 200;


    ACTR        = 0x0999;       /* 00001001 10011001 */
    DBTCON  = 0x02E0;           /* 00000010 11100000 */

    CMPR1   = 0x0000;
    CMPR2   = 0x0000;
    CMPR3   = 0x0000;

    /*  +----------------- compare enabled
        |++--------------- reload compare register on underflow or period match
        |||+-------------- space vector not enabled
        ||||++------------ action control register reload on underflow
        ||||||+----------- compare output pins enabled
        |||||||+-++++++++- reserved
        ||||||| ||||||||
        10100010 00000000 */
    COMCON  = 0xA200;

    /*  ++---------------- emulation control = not affected
        || ++------------- continuous up counting
        || ||+++---------- prescaler = 1
        || ||||| +-------- Use own TENABLE bit
        || ||||| |+------- enable T1
        || ||||| ||++----- internal clock as source
        || ||||| ||||++--- reload period on underflow or equal
        || ||||| ||||||+-- enable compare operation
        || ||||| |||||||+- not used in T1
        || ||||| ||||||||
        11010000 01000110 */
    T1CON   = 0xD046;

    /*   +---------------- read only counting status
         |+--------------- read only counting status
         ||  ++----------- no adc event with timer 2
         ||  ||+-+-------- no adc event with timer 1
         ||  ||| |+------- compare output enable - enable
         ||  ||| ||  ++--- polarity timer 2 - active high
```

```
          ||  ||| ||  ||++- polarity timer 1 - active high
          ||  ||| ||  ||||
         00000000 01001010 */
    GPTCON   = 0x004A;
}



/* interrupt service routine
 * calls sepearte function if CANMB, CANER or Phantom error
 */
void c_int1(void){
/*  sciout('I');
*/  if (PIVR == 0x0040) {           /* test for CAN receive interrupt */
        can_receive();
    } else if (PIVR == 0x0041) {    /* Can error */
        CANIFR = 0x0078 ;           /* clear error flag */
        CANESR = 0xFFFF ;           /* clear error register */
    } else if (PIVR == 0x0000) {}   /* phantom interrupt */

    INT_ENABLE;                     /* re enable interrupts */
}

void c_int2 () {
    /*if timer1 period interrupt occurs*/
    if (PIVR == 0x0027) {
        /*clear timer1 period interrupt flag*/
        EVIFRA = 0x0080;
        count++;

        if (feather_count < counter1 )
            CMPR1 = pwm_high_1 ;
        else if (feather_count >= counter1)
            CMPR1 = pwm_low_1 ;

        if (feather_count < counter2 )
            CMPR2 = pwm_high_2 ;
        else if (feather_count >= counter2)
            CMPR2 = pwm_low_2 ;

        if (feather_count < counter3 )
            CMPR3 = pwm_high_3 ;
        else if (feather_count >= counter3)
            CMPR3 = pwm_low_3 ;

        feather_count++;

        if (feather_count  >= 16) {
            feather_count = 0;
        }

    } else if (PIVR == 0x0000) {}   /* phantom interrupt */

    INT_ENABLE;
}
```

```
/*****************************************************************************
 *     File: lowlevel.c
 *   Author: Andrew Hood
 *  Project: Robot 1.0
 *  Created: 03/05/2002
 *  Summary: Robot specific mirror functions of humanoid.cpp
 *****************************************************************************/

#include "f24x.h"
#include "ioports.h"
#include "..\Common\humanoid.h"
#include "..\Robot\lowlevel.h"

int counter1;
int counter2;
int counter3;
int pwm_high_1 ;
int pwm_high_2 ;
int pwm_high_3 ;
int pwm_low_1 ;
int pwm_low_2 ;
int pwm_low_3 ;


extern int values[8];

unsigned int read_enc(int k) {

    /*returns current encoder reading in encoder counts*/
    unsigned int EXT_DATA1;
    unsigned int EXT_DATA2;
    unsigned int qdec_ext2;
    unsigned int qdec_ext3;

    /* read in the high bytes into variable*/
    inport (0, &EXT_DATA1);

    /* read in the low bytes into variable */
    inport (1, &EXT_DATA2);

    /* high byte into 16 bit variables, shift left 8 places for ext2*/
    qdec_ext3 = EXT_DATA1;
    qdec_ext2 = EXT_DATA1 << 8;

    /* low byte into variables, shift right 8 places for ext3.*/
    qdec_ext3 = (qdec_ext3 & 0xFF00) | (EXT_DATA2 >> 8);
    qdec_ext2 = (qdec_ext2 & 0xFF00) | (EXT_DATA2 & 0x00FF);

    /*  return appropriate value, depending on which motor is being driven*/
    switch(k){

    case 0:
        return T2CNT;
        break;
    case 1:
        return qdec_ext2;
        break;
    case 2:
        return qdec_ext3;
        break;
    }
}


/*reads current sensor and returns current in mA*/
unsigned int read_curr(int k)
{
    unsigned int current;
    unsigned int scaled_current;

    switch (k) {
    case 0:
        ADCTRL1 = 0x6900;       /* 6 sets the start conversion now bit, last*/
        break;                  /* number specifies the h-bridge being read */
    case 1:
        ADCTRL1 = 0x6902;
        break;
    case 2:
        ADCTRL1 = 0x6904;
        break;
    }
```

```
    ADCTRL1 = ADCTRL1 | 0x0001 ;            /* Start conversion */

    /* bit seven is high when a conversion is being
    performed, and drops low when it is finished */
    while((ADCTRL1 & 0x0080) == 0x0080);
    current = ADCFIFO1 >> 6;                /* bottom 6 bytes of FIFO1 are reserved */



    scaled_current = (current * 17) + 152;      /* scaled between 0 and 5000mA */

/*  if (k == 2){
        sciout('I');
        sciouthex(scaled_current);
    }

/*  if (scaled_current > CURRENT_LIMIT){        /* if over, disable motor drivers */
/*      PBDATDIR &= 0xFFDF ;
    }else {
        PBDATDIR |= 0x0020 ;
    }
*/
    /* This number determined experimentally result of ADC  / 15.5 under static conditions*/
    return scaled_current;
}


/*
Sets PWM duty cycle for motor k. Value passed ranges between -1600 to 1600
representing 100% backward to 100% foward
*/
void set_PWM(int k, int pwm_duty) {

    switch(k) {
    case 0:
        counter1 = pwm_duty & 0x000F ;
        pwm_high_1 = (pwm_duty >> 4) + 101 ;
        pwm_low_1 = (pwm_duty >> 4) + 100 ;
        break;
    case 1:
        counter2 = pwm_duty & 0x000F ;
        pwm_high_2 = (pwm_duty >> 4) + 101 ;
        pwm_low_2 = (pwm_duty >> 4) + 100 ;
        break;
    case 2:
        counter3 = pwm_duty & 0x000F ;
        pwm_high_3 = (pwm_duty >> 4) + 101 ;
        pwm_low_3 = (pwm_duty >> 4) + 100 ;
        break;
    }


    /*  switch(k){
    case 0:
        CMPR1 = (unsigned int) (pwm_duty + 100);
        break;
    case 1:
        CMPR2 = (unsigned int) (pwm_duty + 100);
        break;
    case 2:
        CMPR3 = (unsigned int) (pwm_duty + 100);
        break;

    } */
}
```

APPENDIX C – 2002 BOARD SCHEMATIC

full rectifier H-bridge configuration, one for each motor

connector for the HEDS5400 encoders on each moor

LT1676: chips step down incoming 42V to 12V then to 3.3V to power all electronics on the control board

HIP4081, h-bridge driver chip. Provides bootstrap voltage for the high side of the h-bridge

TMS320lf2406A 3.3V DSP @ 40MHz. runs control loop

A low current 5 volt regulator VCCP needs 5V to program flash

RESET chip. holds reset line low for 150ms when pushbutton is pressed

JTAG header. if EEPROM needs re-programming, or user has decided to program via JTAG, plug connector into this

CAN Tranceiver chip. provides the interface to the CAN standard differential two wire bus

SERIAL chip. converts logic levels to the RS-232 standard levels for communication (in this case programming and debugging via PC)

debug LEDS and header to clip on oscilloscope

Connector for the optical alignment board design

Title	Joint Controller Board 2002
Size	A2
Number	
Revision	
Date:	17-Oct-2002
Sheet of	
File:	C:\GalRoo\MyDesign.ddb
Drawn By:	

APPENDIX D - DATASHEETS

- **High-Performance Static CMOS Technology**
  - **25-ns Instruction Cycle Time (40 MHz)**
  - **40-MIPS Performance**
  - **Low-Power 3.3-V Design**
- **Based on TMS320C2xx DSP CPU Core**
  - **Code-Compatible With F243/F241/C242**
  - **Instruction Set and Module Compatible With F240/C240**
- **Flash (LF) and ROM (LC) Device Options**
  - **LF240xA: LF2407A, LF2406A, LF2403A, LF2402A**
  - **LC240xA: LC2406A, LC2404A, LC2402A**
- **On-Chip Memory**
  - **Up to 32K Words x 16 Bits of Flash EEPROM (4 Sectors) or ROM**
  - **Programmable "Code-Security" Feature for the On-Chip Flash/ROM**
  - **Up to 2.5K Words x 16 Bits of Data/Program RAM**
    - **544 Words of Dual-Access RAM**
    - **Up to 2K Words of Single-Access RAM**
- **Boot ROM (LF240xA Devices)**
  - **SCI/SPI Bootloader**
- **Up to Two Event-Manager (EV) Modules (EVA and EVB), Each Includes:**
  - **Two 16-Bit General-Purpose Timers**
  - **Eight 16-Bit Pulse-Width Modulation (PWM) Channels Which Enable:**
    - **Three-Phase Inverter Control**
    - **Center- or Edge-Alignment of PWM Channels**
    - **Emergency PWM Channel Shutdown With External $\overline{\text{PDPINTx}}$ Pin**
  - **Programmable Deadband (Deadtime) Prevents Shoot-Through Faults**
  - **Three Capture Units for Time-Stamping of External Events**
  - **Input Qualifier for Select Pins**
  - **On-Chip Position Encoder Interface Circuitry**
  - **Synchronized A-to-D Conversion**
  - **Designed for AC Induction, BLDC, Switched Reluctance, and Stepper Motor Control**
  - **Applicable for Multiple Motor and/or Converter Control**
- **External Memory Interface (LF2407A)**
  - **192K Words x 16 Bits of Total Memory: 64K Program, 64K Data, 64K I/O**
- **Watchdog (WD) Timer Module**
- **10-Bit Analog-to-Digital Converter (ADC)**
  - **8 or 16 Multiplexed Input Channels**
  - **375 ns or 500 ns MIN Conversion Time**
  - **Selectable Twin 8-State Sequencers Triggered by Two Event Managers**
- **Controller Area Network (CAN) 2.0B Module (LF2407A, 2406A, LF2403A)**
- **Serial Communications Interface (SCI)**
- **16-Bit Serial Peripheral Interface (SPI) (LF2407A, 2406A, LC2404A, LF2403A)**
- **Phase-Locked-Loop (PLL)-Based Clock Generation**
- **Up to 40 Individually Programmable, Multiplexed General-Purpose Input/Output (GPIO) Pins**
- **Up to Five External Interrupts (Power Drive Protection, Reset, Two Maskable Interrupts)**
- **Power Management:**
  - **Three Power-Down Modes**
  - **Ability to Power Down Each Peripheral Independently**
- **Real-Time JTAG-Compliant Scan-Based Emulation, IEEE Standard 1149.1† (JTAG)**
- **Development Tools Include:**
  - **Texas Instruments (TI) ANSI C Compiler, Assembler/Linker, and Code Composer Studio™ Debugger**
  - **Evaluation Modules**
  - **Scan-Based Self-Emulation (XDS510™)**
  - **Broad Third-Party Digital Motor Control Support**
- **Package Options**
  - **144-Pin LQFP PGE (LF2407A)**
  - **100-Pin LQFP PZ (2406A, LC2404A)**
  - **64-Pin TQFP PAG (LF2403A, LC2402A)**
  - **64-Pin QFP PG (2402A)**
- **Extended Temperature Options (A and S)**
  - **A: – 40°C to 85°C**
  - **S: – 40°C to 125°C**

Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

Code Composer Studio and XDS510 are trademarks of Texas Instruments.
† IEEE Standard 1149.1–1990, IEEE Standard Test-Access Port

Wide Input Range, High Efficiency, Step-Down Switching Regulator

FEATURES

- **Wide Input Range: 7.4V to 60V**
- 700mA Peak Switch Current Rating
- **Adaptive Switch Drive Maintains Efficiency at High Load Without Pulse Skipping at Light Load**
- True Current Mode Control
- 100kHz Fixed Operating Frequency
- Synchronizable to 250kHz
- Low Supply Current in Shutdown: 30µA
- Available in 8-Pin SO and PDIP Packages

APPLICATIONS

- Automotive DC/DC Converters
- Telecom 48V Step-Down Converters
- Cellular Phone Battery Charger Accessories
- IEEE 1394 Step-Down Converters

DESCRIPTION

The LT®1676 is a wide input range, high efficiency Buck (step-down) switching regulator. The monolithic die includes all oscillator, control and protection circuitry. The part can accept input voltages as high as 60V and contains an output switch rated at 700mA peak current. Current mode control offers excellent dynamic input supply rejection and short-circuit protection.

The LT1676 contains several features to enhance efficiency. The internal control circuitry is normally powered via the V_{CC} pin, thereby minimizing power drawn directly from the V_{IN} supply (see Applications Information). The action of the LT1676 switch circuitry is also load dependent. At medium to high loads, the output switch circuitry maintains high rise time for good efficiency. At light loads, rise time is deliberately reduced to avoid pulse skipping behavior.

The available SO-8 package and 100kHz switching frequency allow for minimal PC board area requirements.

TYPICAL APPLICATION

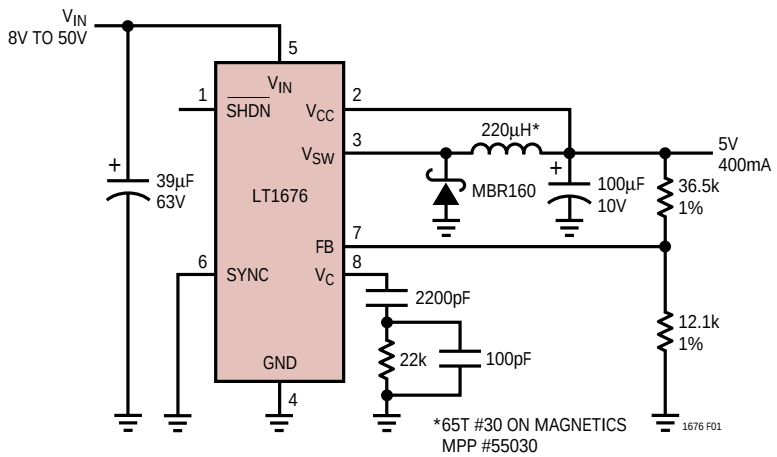
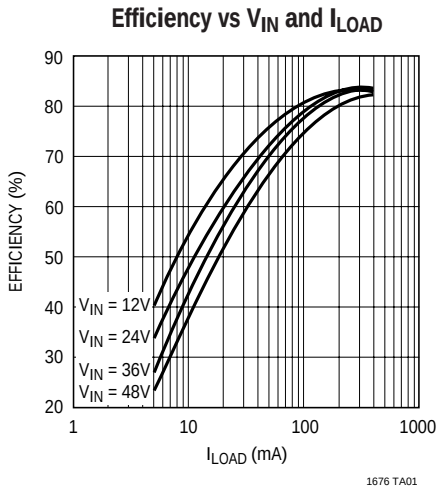


Figure 1



HARRIS SEMICONDUCTOR

HIP4081

80V/2.5A Peak, High Frequency Full Bridge FET Driver

November 1996

Features

- **Independently Drives 4 N-Channel FET in Half Bridge or Full Bridge Configurations**
- **Bootstrap Supply Max Voltage to 95V_{DC}**
- **Drives 1000pF Load at 1MHz in Free Air at 50^{o}C with Rise and Fall Times of Typically 10ns**
- **User-Programmable Dead Time**
- **On-Chip Charge-Pump and Bootstrap Upper Bias Supplies**
- **DIS (Disable) Overrides Input Control**
- **Input Logic Thresholds Compatible with 5V to 15V Logic Levels**
- **Very Low Power Consumption**

Applications

- **Medium/Large Voice Coil Motors**
- **Full Bridge Power Supplies**
- **Class D Audio Power Amplifiers**
- **High Performance Motor Controls**
- **Noise Cancellation Systems**
- **Battery Powered Vehicles**
- **Peripherals**
- **U.P.S.**

Ordering Information

PART NUMBER	TEMP. RANGE (^{o}C)	PACKAGE	PKG. NUMBER
HIP4081IP	-40 to 85	20 Lead Plastic DIP	E20.3
HIP4081IB	-40 to 85	20 Lead Plastic SOIC	M20.3

Description

The HIP4081 is a high frequency, medium voltage Full Bridge N-Channel FET driver IC, available in 20 lead plastic SOIC and DIP packages. The HIP4081 can drive every possible switch combination except those which would cause a shoot-through condition. The HIP4081 can switch at frequencies up to 1MHz andssssss is well suited to driving Voice Coil Motors, high-frequency Class D audio amplifiers, and power supplies.

For example, the HIP4081 can drive medium voltage brush motors, and two HIP4081s can be used to drive high performance stepper motors, since the short minimum “on-time” can provide fine micro-stepping capability.

Short propagation delays of approximately 55ns maximizes control loop crossover frequencies and dead-times which can be adjusted to near zero to minimize distortion, resulting in rapid, precise control of the driven load.

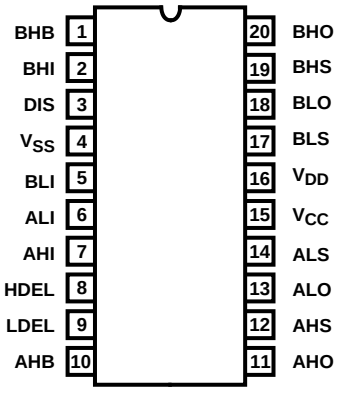
A similar part, the HIP4080, includes an on-chip input comparator to create a PWM signal from an external triangle wave and to facilitate “hysteresis mode” switching.

See Application Note AN9325 for HIP4081, Harris AnswerFAX, (407) 724-7800, document #99325. Harris web home page: http://www.semi.harris.com

Similar part HIP4081A includes undervoltage circuitry which does not require the circuitry shown in Figure 30 of this data sheet.

Pinout

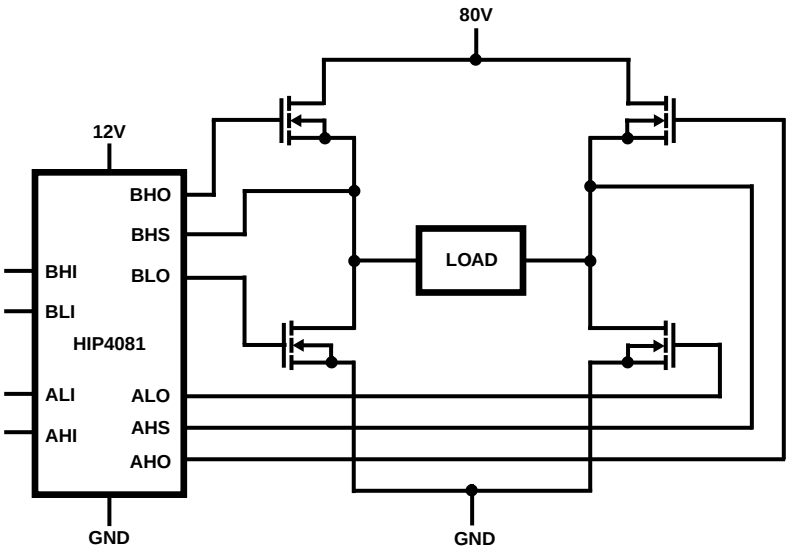
HIP4081
(PDIP, SOIC)
TOP VIEW



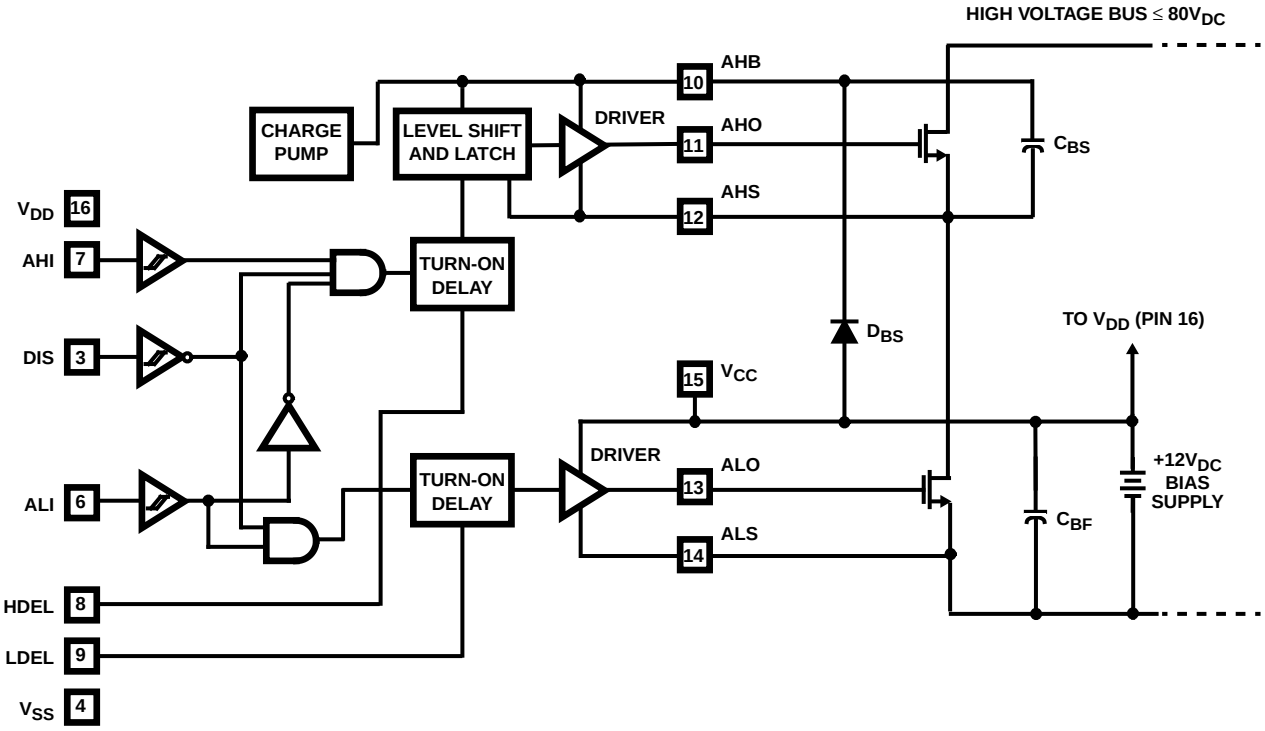
CAUTION: These devices are sensitive to electrostatic discharge. Users should follow proper IC Handling Procedures.

File Number **3556.7**

Application Block Diagram



Functional Block Diagram (1/2 HIP4081)



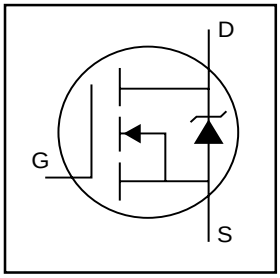
International
IOR Rectifier

PD - 94050A

IRFZ44VS
IRFZ44VL

HEXFET® Power MOSFET

- Advanced Process Technology
- Ultra Low On-Resistance
- Dynamic dv/dt Rating
- 175°C Operating Temperature
- Fast Switching
- Fully Avalanche Rated
- Optimized for SMPS Applications



V_{DSS} = 60V

$R_{DS(on)}$ = 16.5mΩ

I_D = 55A

Description

Advanced HEXFET® Power MOSFETs from International Rectifier utilize advanced processing techniques to achieve extremely low on-resistance per silicon area. This benefit, combined with the fast switching speed and ruggedized device design that HEXFET power MOSFETs are well known for, provides the designer with an extremely efficient and reliable device for use in a wide variety of applications.

The D^2Pak is a surface mount power package capable of accommodating die sizes up to HEX-4. It provides the highest power capability and the lowest possible on-resistance in any existing surface mount package. The D^2Pak is suitable for high current applications because of its low internal connection resistance and can dissipate up to 2.0W in a typical surface mount application.
The through-hole version (IRFZ44VL) is available for low-profile applications.

D^2Pak
IRFZ44VS

TO-262
IRFZ44VL

Absolute Maximum Ratings

	Parameter	Max.	Units
I_D @ T_C = 25°C	Continuous Drain Current, V_{GS} @ 10V	55	A
I_D @ T_C = 100°C	Continuous Drain Current, V_{GS} @ 10V	39	
I_{DM}	Pulsed Drain Current ①	220	
P_D @T_C = 25°C	Power Dissipation	115	W
	Linear Derating Factor	0.77	W/°C
V_{GS}	Gate-to-Source Voltage	± 20	V
E_{AS}	Single Pulse Avalanche Energy②	115	mJ
I_{AR}	Avalanche Current①	55	A
E_{AR}	Repetitive Avalanche Energy①	11	mJ
dv/dt	Peak Diode Recovery dv/dt ③	4.5	V/ns
T_J	Operating Junction and	-55 to + 175	°C
T_{STG}	Storage Temperature Range		
	Soldering Temperature, for 10 seconds	300 (1.6mm from case)	

Thermal Resistance

	Parameter	Typ.	Max.	Units
$R_{\theta JC}$	Junction-to-Case	—	1.3	°C/W
$R_{\theta JA}$	Junction-to-Ambient	—	40	

APPENDIX E – DISTRIBUTED CONTROL GAIT OF A HUMANOID ROBOT

Distributed Control of Gait for a Humanoid Robot

Gordon Wyeth, Damien Kee, Adam Drury, Andrew Hood
School of Information Technology and Electrical Engineering
University of Queensland
St. Lucia, Queensland, 4072
Australia

Abstract

This paper describes a walking gait for a humanoid robot with a completely distributed control system. The motion for the robot is calculated in real time on a central controller, and sent over CAN bus to the distributed control system. The distributed control system loosely follows the motion patterns from the central controller, while also acting to maintain stability and balance. There is no global feedback control system; the system maintains its balance by the interaction between central gait and "soft" control of the actuators. The paper illustrates a straight line walking gait and shows the interaction between gait generation and the control system. The analysis of the data shows that successful walking can be achieved without maintaining strict local joint control, and without explicit global balance coordination.

1 Introduction

Humanoid robots typically require coordinated control of a large number of joints. In most existing implementations of humanoid robots, coordination is achieved by the use of a central control computer that interfaces to all sensors and actuators providing local control of joint positions and torques as well as global control of balance and posture. This paper describes a distributed approach to control and coordination that provides local control of position and torque at each joint in a fashion that maintains global balance and posture.

1.1 Paper Overview

The paper first describes the GuRoo robot that forms the basis for the later experiments. Details of the architecture and design of the robot are followed by a description of the computing system that supports the distributed control system. The paper then describes the approach to distributed control and provides details of gait generation, including results gathered from a straight line walk.

2 The GuRoo Project

GuRoo is a 1.2 m tall, fully autonomous humanoid robot designed and built in the University of Queensland Robotics Laboratory [Wyeth, 2001]. The robot has a total mass of 34 kg, including on-board power and computation. *GuRoo* is currently capable of a number of demonstration tasks including balancing, walking, turning, crouching, shaking hands and waving. The robot has performed live demonstrations of combinations of these tasks at various robot displays.

The intended challenge task for the robot is to play a game of soccer with or against human players or other humanoid robots. To complete this challenge, the robot must be able to move freely on its two legs. It requires a vision sense that can detect the objects in a soccer game, such as the ball, the players from both teams, the goals and the boundaries. It must also be able to manipulate and kick a ball with its feet, and be robust enough to deal with contact with other players. Clearly, the robot must operate in a completely autonomous fashion without support harnesses or wiring tethers. These goals are yet to be realised for the GuRoo project, but serve as inspiration for the current work on the on-board vision system, and the design of balance behaviours and dynamic gait control.

Figure 1: The GuRoo humanoid robot in its current form.

2.1 Architecture

GuRoo has been designed to replicate the human form, to such an extent as conflicting factors of function, power, weight, cost and manufacturability allow. Figure 2 shows the degrees of freedom contained in each joint area of the robot. In the cases where there are multiple degrees of freedom (for example, the hip) the joints are implemented sequentially through short links rather than as spherical joints.

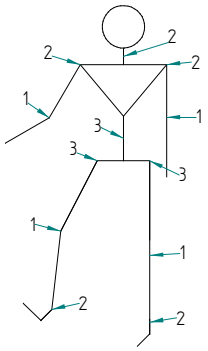


Figure 2: The location of the joints in GuRoo, indicating the degrees of freedom in each joint.

Table 1 shows the mass distribution of GuRoo compared to that of a human. The most notable exception is that the shin and foot are much heavier in GuRoo than the human counterpart, due to the mass of the powerful actuators required in the ankle. The arms are significantly lighter than the human counterpart, as they are significantly inferior in power and do not have hands.

Table 1: Comparison of GuRoo mass distribution with human mass distribution.

Body Component	GuRoo mass (kg)	GuRoo	Human
Head and Upper torso	9.2	27%	31%
Abdomen and Hips	9.8	29%	27%
Thigh	6.4	19%	20%
Shin and Foot	6.8	20%	12%
Arm	1.9	6%	10%
Total	34.1		

The other notable point from Table 1 is the total mass of the robot. A 1.2 m tall human would typically be a child approaching his or her 7th birthday, with a 50th percentile mass of 23 kg. A child with mass of 34 kg at the same age would be in 99th percentile, indicating that GuRoo is somewhat overweight.

2.2 Electro-Mechanical Design

The key element in driving the mechanical design has been the choice of actuator. The robot has 23 joints in total. The legs and abdomen contain 15 joints that are required to produce significant mechanical power, most generally with large torques and relatively low speeds. The other 8 joints drive the head and neck assembly, and the arms. The torque and speed requirements are significantly less.

The 15 high power joints all use the same motor-gearbox combination. The motor is a Maxon RE 36 wound for a nominal voltage of 32 V with a gearbox reduction of 156:1. The maximum continuous generated output torque is 10 Nm, with a maximum output speed of 51 RPM, or 5.3 rad/s. The thermal limits of the motor permit intermittent output torque of up to 19 Nm. Each motor is fitted with an optical encoder for position and velocity feedback. The 8 low power joints are Hi-Tec RC servo motors model HS705-MG. with rated output torque to 1.4 Nm, at speeds of 5.2 rad/s. These have built-in control and power electronics using a pulse width modulated signal to indicate desired position.

The motors that drive the roll axis of the hip joints are supplemented by springs with a spring constant of 1 Nm/degree. These springs serve to counteract the natural tendency of the legs to collide, and help to generate the swaying motion that is critical to the success of the walking gait.

Power is provided by 2 × 1.5Ah 42V NiCd packs for the high power motors, and 2 x 3Ah 7.2 V NiCd battery packs for computing and servo operation. The packs are chosen to give 20 minutes of continuous operation. The drive power electronics is based on a switch mode power stage, requiring only a single supply rail and having an efficiency over 90%.

2.3 Sensing

The position feedback from the encoders on the high power joints provides a count on every edge of both quadrature channels. This provides 867 encoder counts per degree of joint motion. In addition, each DSP can measure the current to each motor, the bus voltage, and the temperatures of the MOSFETs and motors. Provision has also been made for inertial and balance sensors, as well as contact switches in the feet and in the joints.

3 Distributed Control Network

A distributed control network controls the robot, with a central computing hub that sets the goals for the robot, processes the sensor information, and provides coordination targets for the joints. The joints have their own control processors that act in groups to maintain global stability, while also operating individually to provide local motor control. The distributed system is connected by a CAN network. In addition, the robot requires various sensor amplifiers and power conversion circuits.

3.1 Central Control

The central control of the robot derives from two heterogeneous microprocessors that provide coordination between joints, integrate sensor information, and process the vision input. The primary component of the central controller is an iPAQ pocket pc from Compaq. The iPAQ features a 208 MHz StrongARM microcontroller, 32 Mb of RAM and a 320 x 240 colour screen. The screen is touch sensitive allowing stylus input of text and graphics. The iPAQ in the GuRoo operates with Windows CE. As well as the touch screen interface, the iPAQ is equipped with a speaker and microphone, a joypad, and four push-buttons. It has an infra-red interface for external communication.

The second component of the central hub is the vision processing board. This board has been developed for the ViperRoos robot soccer team [Chang, 2001] and features a 200 MHz Hitachi Super-H SH4 microcontroller, an FPGA-based programmable camera and bus adapter, 16 Mb of RAM, 8 Mb of flash ROM, and 512 kb of fast SRAM for video caching. The vision input comes from a custom digital CMOS camera, based around the OV7620 camera chip from OmniVision, which can provide 640 x 480 images at up to 25 fps. The camera can provide data in YUV or RGB formats, and can be programmed to only send data from selected areas of the sense region.

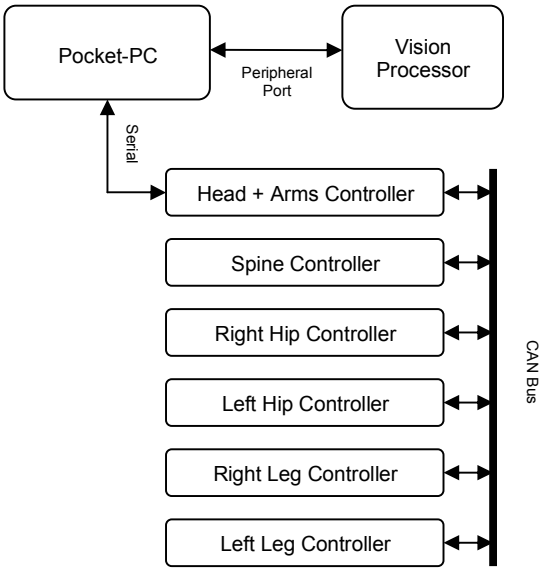


Figure 3: Block diagram of the distributed control system.

The vision board has run video capture from the CMOS camera and colour segmentation of various scenes. It is not yet operational on the robot as the interface to the 100 pin parallel peripheral bus on the iPAQ is yet to be completed.

3.2 Joint Controllers

The TMS320F24x series from Texas Instruments is a 32 bit DSP designed for motor control. It can operates at 20MHz, and can read the A/D converter, calculating a PID control law, current limit, and generate the required PWM output, in under 10 µs [Wyeth, 2000]. The availability of the Control Area Network (CAN) module in this series, along with bootloader programmable internal Flash memory makes the device particularly attractive for this application. In this application, the TMS320F243 is used, which has an external bus that is used for attaching additional sensor interfaces. Five controller boards control the 15 high power motors, each board controlling three motors. A sixth controller board controls the eight RC servo motors.

3.3 Internal Network

The CAN bus is a highly reliable standard developed by Robert Bosch GmbH for use in the automotive environment. It is a multi-master system, with sophisticated error checking and arbitration, so that any high priority message will always get through first without corruption by other messages. All data contained in each packet (up to eight bytes) is also checked with a Cyclic Redundancy Check (CRC) error-checking scheme that can correct up to five random errors, and will be automatically retransmitted if not correct. The network operates at up to 1 Mbit/sec.

4 Software

The software consists of four main entities: the global movement generation code, the local motor control, the low-level code of the robot, and the simulator. The software is organised to provide a standard interface to both the low-level code on the robot and the simulator. This means that the software developed in simulation can be simply re-compiled to operate on the real robot. Consequently, the robot needs a number of standard interface calls that are used for both the robot and the simulator. Figure 4 shows modularisation of the software, and the common interfaces.

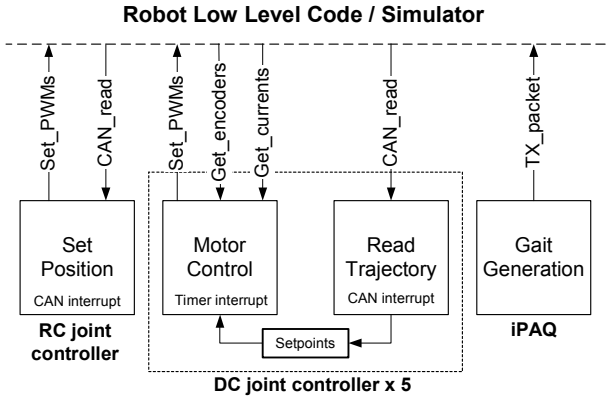


Figure 4: Block diagram of common software modules and the interface used to both the real robot and the simulator.

4.1 Gait Generation

The gait generation module is responsible for producing realisable trajectories for the 23 joints so that the robot can perform basic behaviours, such as standing on one leg, crouching and walking. The most important properties of the trajectories are that they are smooth and that they can be linked together in smooth fashion. Smoothness of motion implies less disturbance to the control of other joints. Based on these criteria, all motor trajectories are generated from a parameterised sinusoidal curve as follows,

$$\omega = \frac{\theta}{T}\left(1 - \cos\left(\frac{2\pi t}{T}\right)\right) \quad (1)$$

where ω is the desired joint velocity, θ is the total joint angle to move and T is the period of that movement. A normalised joint movement is shown in Figure 5.

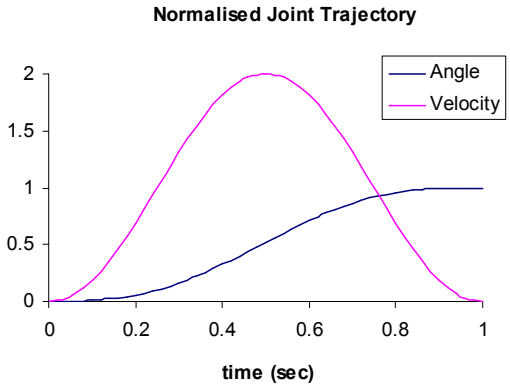


Figure 5: The trajectory used for a total joint movement of 1 radian over a period of 1 second.

This velocity trajectory has the property of being smooth in the sense that the joint acceleration starts and finishes at zero, smoothly increasing to finite peaks. This implies that the jerk (derivative of acceleration) also starts at zero and reaches finite peaks. The trajectory contrasts with typical trajectories generated for robotic manipulators, which typically focus on smoothness of the end effector motion rather than smoothness at the joint.

Trajectories for each of the motors may be coordinated by using the same beginning time for the motion and specifying the same period for the trajectory. Trajectories may be naturally linked as the velocities all reach zero at the beginning and the end of a motion. Section 5 will illustrate how trajectories may be coordinated and linked to perform a walking operation.

4.2 Joint Controller Software

There are two types of joint controller boards used in the robot – five controller boards control the fifteen high power motors and one controller controls the eight low power motors. The controller software for the low power motors is a single interrupt routine that is triggered by the arrival of a CAN packet addressed to the controller's mailbox. The routine reads the CAN mailbox for the change in position sent by the gait generation routine. The PWM duty cycle that controls the position of the RC servos is varied accordingly.

The control loop for the high power controllers has two interrupt routines. As for the low power controller, an interrupt is executed upon receipt of trajectory data in the CAN mailbox. The data is used to set the velocity setpoints for the motor control routine. There is also a periodic interrupt every 500 μs to run the motor control software. The motor control routine compares the error between velocity setpoint and the encoder reading and generates a PWM value for the motor based on a Proportional-Integral control law. The routine also checks the motor current against the current limits, and adjusts the PWM value to prevent over-current situations.

The PI control law on each joint has been hand tuned to provide both good trajectory following for typical velocity input profiles, and spring-damper model impedance to torque disturbances from gravity and the cross-coupling torques from other joints. The "soft" response of this control law to disturbance prevents torques being transmitted throughout the robot and helps to maintain global stability. The disadvantage is that the controller suffers from position error that must be accounted in the gait generation software. Section 5 illustrates how this potential liability is turned to an asset in the generation of a dynamically stable walk.

4.3 Low-level Code

The lowest level of code on the robot provides direct access to the sensors and communication system. The level of abstraction provided by function calls at this level aids in the cross development of code between the simulator and the real robot.

4.4 Simulator

The simulator is based on the *DynaMechs* project [McMillan, 1995], with additions to simulate specific features of the robot such as the DC motors and motor drives, the RC servos, the sensors, the heterogeneous processing environment and the CAN network. These additions provide the same interface for the dynamic graphical simulation as for the joint controller and gait generation code. The parameters for the simulator are derived from the CAD models and the data sheets from known components. These parameters include the modified Denavit-Hartenberg parameters that describe the robot topology, the tensor matrices of the links and the various motor and gearbox characteristics associated with each joint. The surface data from the CAD model is also imported to the simulator for the graphical display.

For the high power DC motor joints, the simulator provides the programmer with readings from the encoders and the current sensors, based on the velocities and torques from the dynamic equations. In the case of the RC servos, the simulator updates the position of the joints based on a PD model with a limited slew rate. The programmer must supply the simulator with PWM values for the motors to provide the control. The simulator provides fake interrupts to simulate the real events that are the basis of the control software.

The simulator uses an integration step size of 500μs and updates the graphical display every 5ms of simulated time. When running on 1.5 GHz Pentium 4 under Windows 2000, the simulation updates all 23 joints at a very useable 40% of real time speed.

5 Walking

The robot can walk with a step rate of 1 Hz using a step length of 100 mm. The walk is open-loop; there is no feedback from the joint controllers to the gait generation software. The lack of global feedback, combined with the absence of a global balance sensor presents a substantial challenge in walking algorithm design. The gait described in this section forms a base for faster and more robust walking with the augmentation of balance control.

5.1 Walking Algorithm

The robot uses a simplified version of a typical human gait. In particular, it limits the swing of the legs to prevent balance disturbance as this cannot be corrected without global balance control. In order to minimise the accelerations of the torso, head and arms (which make up 1/3 of the mass of the robot), the robot maintains a constant relative position of the torso, such that the face of the torso is always normal to the direction of travel. The allowable roll of the torso is also limited. The stabilisation of the torso also reduces disturbances from gravity to the control of the leg joints.

Before walking, the robot loads each motor against gravity by performing a slight squat that introduces a 6 degree ankle pitch, with the knee and hip pitch joints set to keep the torso upright. The initial loading of the joints reduces the likelihood of backlash in the gearheads.

The walking gait commences with a side-to-side sway generated from the roll axes of the ankles and hips. The sway frequency of 0.5 Hz is sympathetic with the spring mass system formed by the ankle controllers with the mass of robot. The sway sets up the pattern of weight transfer from one foot to the other necessary to swing the legs alternately to achieve walking. At each extreme of

the sway, the inertia of the upper body ensures the ZMP (Zero Moment Point) of the robot lies within the support polygon formed by the support foot, even thought the centre of mass may not. This action places requires less torque from the hip and ankle roll actuators, as the motion due to gravity brings the robot away from the extreme of each sway.

Once the sway is sufficient to leave no ground reaction force on the non-supporting foot, the non-supporting leg is lifted using the pitch axes of the hip, knee and ankle. Between the lifting and lowering of the non-supporting leg, each yaw axis motor twists, so that the non-supporting leg swings forward to create the step. When the swing leg contacts the ground, the robot is dynamically stable, with the centre of mass over the supporting foot in the frontal plane, but in front of the toes in the saggital plane. The robot then swings across to the other foot, repeating the sequence and progressing with the walk.

5.2 Analysis of Results

The motion of the robot is best analysed by comparing the desired velocity from the gait generation module to the actual velocity at each joint. Figure 8 shows the comparison for the velocities of the pitch motion of the hip, knee and ankle. Figure 9 shows the roll of the hip and ankle, while Figure 10 illustrates the yaw from the hip. The graphs are initialised in the double support phase, at the point when the robot is vertical, halfway through the movement that transfers the centre of mass from the supporting foot to the swing foot. Both legs are fully extended and are in contact with the ground. The graphs comprise one second of data, describing the right leg as it moves from the double support phase, through the swing phase back to the double support phase.

At the point $t = 0.25$ s, the swing leg starts to lift and loses contact with the ground. The support leg now takes the weight of the whole robot. With the hip roll axis of the swing leg no longer contributing to the support of the robot, the spring located in this axis briefly dominates the actual velocity causing the overdamped oscillation seen at hip joint at this time.

Once the foot leaves the ground, it can no longer be considered a fixed link. At this point, the ankle roll motor switches from driving the leg from the foot, to driving the foot from the leg. This large decrease in relative inertia results in a brief increase in the magnitude of the ankle roll velocity. The foot has a relatively low inertia compared with the rest of the robot, and as such the PI controller has little trouble following the desired velocity until the foot again makes contact with the ground. The robot reaches the extreme of each sway at $t = 0.5$ s, where all motion in the roll plane ceases. The swing leg is now theoretically fully lifted, although Figure 8 indicates the knee and hip pitch do not reach their desired positions until T=0.6s.

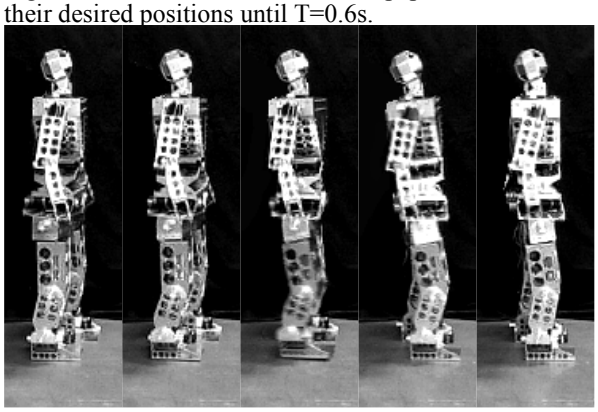


Figure 6: This sequence of images captured 200 ms apart shows the movement of the swing leg through a single step.

The actual joint velocity profile for each pitch motor in the swing leg shows the effect of gravity and leg inertia. The further the leg is actuated away from a vertical position, the greater the influence of gravity. The integral term in the PI controller seeks to eliminate steady state error, and as such, dominates the actual velocity, driving each pitch motor to its desired position. As the motor does not reach the maximum desired velocity, it is forced to lengthen the movement time to ensure the area under each graph is the same. The low proportional term results in the poor tracking of the desired velocity, but allows the joint to better deal with external influences.

The comparison of the actual velocity with the desired velocity for each pitch motor in the leg degrades from the ankle to the knee to the hip. The ankle need only accelerate the foot, whereas the knee must accelerate the foot and lower leg. The hip pitch must accelerate the entire leg during the swing phase.

The motion of the yaw axis as the swing leg is lifted, propels the robot forward. When the yaw motion occurs on the support leg, the momentum of the robot causes the joint to overshoot its position. The swing leg is then lowered placing the robot into a double support phase. The friction of two feet against the ground and the weight of the robot on the support leg, prevents the yaw axis positional error from being resolved. This provides a pre-loading of the joint that supports the motion of the next of yaw swing phase.

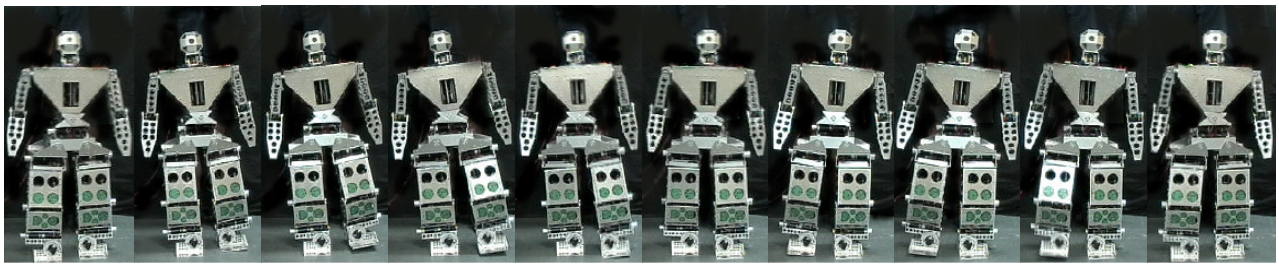


Figure 7: Frontal view of the walking process. The data in figs 9,10,11 start at the image 6 from the left, with the robot in a double support phase, with the left leg becoming the support leg and the right leg the swing leg.

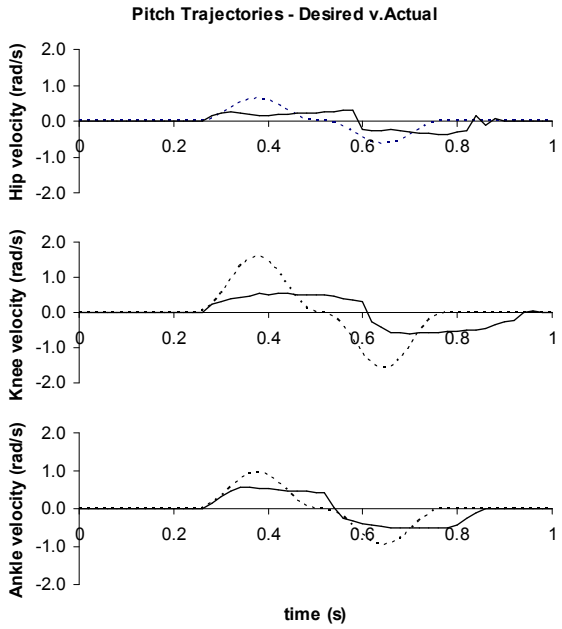


Figure 8: Velocities for the pitch axis of the hip, knee and ankle of the right leg during its swing phase. The dotted line shows the desired velocity, the solid line is the actual velocity.

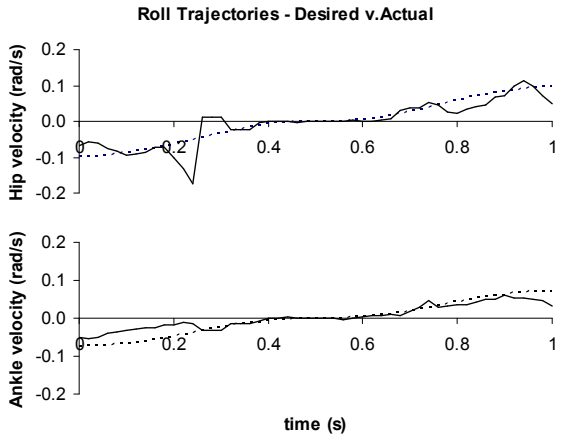


Figure 9: Velocities for the roll axis of the hip and ankle of the right leg during its swing phase.

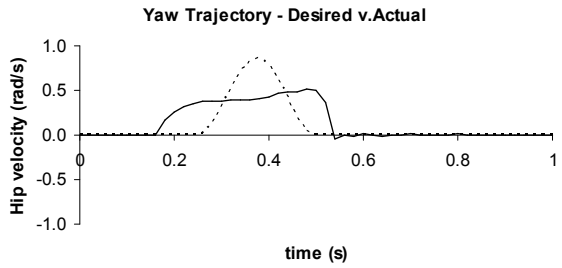


Figure 10: Velocities for the yaw axis of the hip of the right leg during its swing phase.

As the robot sways back to the other side, weight is gradually released from the supporting foot, until the torque acting on the joint overcomes the co-efficient of friction between the foot and the floor. This is not necessarily the point at which the swing leg loses contact with the ground. By time the leg has resolved this error, the joint is experiencing the yaw motion associated with the swing leg twist. As a result, the area under the curve for the hip yaw during the swing phase, is greater than the desired area.

Contact with the ground is achieved at T=0.85s and once made, the robot returns to the double support phase of its gait. Both the hip and ankle of the swing leg now assist the support leg roll motors to sway the robot across to the other foot, and in the process gradually switch the roles of the support and swing leg.

6 Conclusions

This paper has illustrated that a humanoid robot can walk without the need for explicit global feedback, or tightly controlled joint trajectories. By combining a group of loosely coordinated control systems that use "soft" control laws with smooth trajectory generation, the robot can use the natural dynamics of its mechanical structure to move through a gait pattern. The work in this paper shows sound walking performance, that can only improve with the augmentation of global inertial sensors and feedback paths.

References

[Chang, 2001] M. Chang, B. Browning and G. Wyeth. ViperRoos 2000. RoboCup-2000: Robot Soccer World Cup IV. Lecture Notes in Artificial Intelligence 2019. Springer Verlag, Berlin, 2001.

[McMillan, 1995] S. McMillan, Computational Dynamics for Robotic Systems on Land and Underwater, PhD Thesis, Ohio State University, 1995.

[Wyeth, 2000] Wyeth G.F., Kennedy J. and Lillywhite J. (2000) Distributed Digital Control of a Robot Arm, Proceedings of the Australian Conference on Robotics and Automation (ACRA 2000), August 30 - September 1, Melbourne.

[Wyeth, 2001] G. Wyeth, D. Kee, M. Wagstaff, N. Brewer, J. Stirzaker, T. Cartwright, B. Bebel. Design of an Autonomous Humanoid Robot, Proceedings of the Australian Conference on Robotics and Automation (ACRA 2001), 14-15 November 2001,Sydney

APPENDIX F – SOFTWARE COPIES OF ALL DESIGNS AND ASSOCIATED DATASHEETS