

# A dynamically-Balanced Walking Biped

Graham Mann<sup>†</sup>, Bruce Armstrong<sup>†</sup>,

Phil Preston<sup>‡</sup>, Barry Drake<sup>‡</sup>

<sup>†</sup>School of Information Technology  
Murdoch University  
Australia

{g.mann, b.armstrong}@murdoch.edu.au

<sup>‡</sup>School of Computer Science and Engineering  
University of New South Wales  
Australia

{philp, bdrake}@cse.unsw.edu.au

UNSW-CSE-TR-0110

September 2001

THE UNIVERSITY OF  
NEW SOUTH WALES



School of Computer Science and Engineering  
University of New South Wales  
Sydney 2052, Australia

## Abstract

Describes the mechanical, electronic and software design of a 10-DOF bipedal robot which has been constructed to study control, parameterisation and automatic expansion of the stability envelope of a complex real-time behaviour, namely, dynamically-balanced two-legged walking. The machine is physically complete and demonstrates reasonable reliability in movement control including dynamically-balanced standing. High-level reinforcement learning code is being developed to extend this to walking. The machine offers a challenging problem domain to the flourishing machine learning community and represents a shift in emphasis, away from learning algorithms that work on simplified, preprocessed, artificial and static data sets to learning heuristics which deal with noisy, real-time data collected from sensors on a dynamic, real-world system.

# A Dynamically-Balanced Walking Biped

Graham Mann<sup>†</sup> Bruce Armstrong<sup>†</sup>  
Phil Preston<sup>‡</sup> Barry Drake<sup>‡</sup>

**Abstract.** Describes the mechanical, electronic and software design of a 10-DOF bipedal robot which has been constructed to study control, parameterisation and automatic expansion of the stability envelope of a complex real-time behaviour, namely, dynamically-balanced two-legged walking. The machine is physically complete and demonstrates reasonable reliability in movement control including dynamically-balanced standing. High-level reinforcement learning code is being developed to extend this to walking. The machine offers a challenging problem domain to the flourishing machine learning community and represents a shift in emphasis, away from learning algorithms that work on simplified, preprocessed, artificial and static data sets to learning heuristics which deal with noisy, real-time data collected from sensors on a dynamic, real-world system.

## 1 Introduction

After a long period of slow progress, bipedal walking has within the past few years enjoyed a number of notable successes, demonstrating that this means of robot locomotion is a basically practical, if difficult, proposition. In particular, dynamically-balanced walking, in which a vertical line through the centre of balance of the walking system is permitted to stray outside the polygon formed by the feet has been demonstrated in a number of machines (Yamaguchi et.al., 1996; Kun & Millar, 1996). Yet further technical development will be required before robots can show the kind of reliability, dexterity, speed and power which characterises ordinary human performance of this skill. This work details the design of an active-balance biped called TarBaby (Mann, 2000) which serves as a test bed for experiments contributing to this end. Once the basic walking behaviour has been demonstrated, planned experiments aim to create a new kind of algorithm that systematically explores the continuous sensorimotor control space of the robot system outside an initially learned envelope of stability, then reports the system's functional limits of variation for specified parameters; and to discover methods by which this information can be used to automatically generalise the robot's behaviour so that the envelope of stability may be safely enlarged.

---

<sup>†</sup> School of Information Technology, Murdoch University, South Street Murdoch WA 6050  
{g.mann, b.armstrong}@murdoch.edu.au

<sup>‡</sup> School of Computer Science & Engineering, University of New South Wales, Sydney NSW 2052.  
{philp, bdrake}@cse.unsw.edu.au

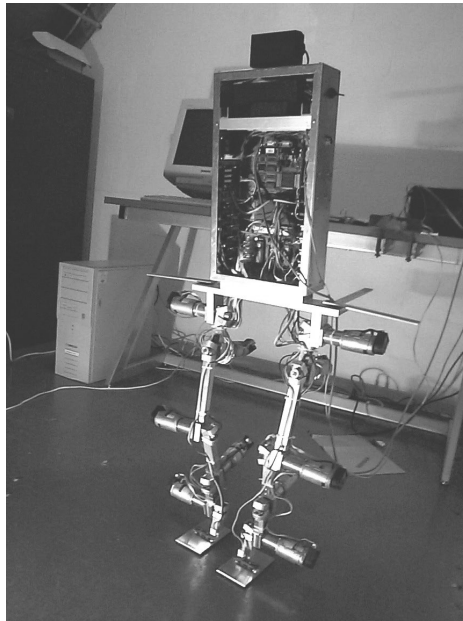


Figure 1. TarBaby robot assembly and external computer

## 2 Design and Construction

### 2.1 Mechanical Structure

Physically, the robot consists of a pair of jointed legs linked by a hip bar supporting a torso consisting of metal case (Figure 1). The torso contains most of the control circuitry and a small lead-acid battery, which powers the main microcontroller. A 3-D motion tracker providing positional feedback is mounted atop the torso. The ten motors actuating the leg joints are powered via cable from a 30Ah laboratory power supply. Control signals from an off-board computer are also supplied via cable.

The legs are essentially scaled-down, simplified versions of human legs. Each leg possesses all but one of the 6 degrees of freedom in a human leg: that enabling yawing motion of the foot. Pairs of single axis joints at the hip and ankle joints approximate the flex-extend and adduct-abduct axes of their human counterparts. The upper member of each joint supports a gearhead-motor/shaft-encoder assembly, with the output shaft coupled to the lower member by a load-spreading flange. The motors are 12 volt DC permanent magnet types that draw 2-3 Amps under normal loads. All gear ratios are 180:1, except the knees, which have a 256:1 ratio for extra torque. The foot is a 12 x 7 cm metal plate with four force sensing resistor disks coupled to floor pads of neoprene rubber. Cabling for motor power, shaft encoder data, and force sensor data run up the legs

and enter the torso through bilateral holes in the hip bar. The whole assembly is 107cm in height and weighs 12kg.

## 2.2 Electronic Systems

The circuitry of the robot is organised as shown the block diagram of Figure 2. The system is designed around a 40MHz 16-bit 80C167-based industrial microcontroller evaluation board manufactured by Phytec. Except for this microcontroller and the motion tracking sensor, all circuits are duplicated bilaterally.

### 2.2.1 Motor Driver Circuits

The principle processing load on the system is continuous monitoring of the ten shaft encoders, which might generate pulses at frequencies as high as 6kHz. To avoid the possibility of miscounting pulses during peak loads, and to simplify the problem of keeping track of shaft positions, a dual subprocessor handles the incoming encoder pulses. The shaft encoder divider board consists of a pair of flash-programmable 8-bit AT90S8515 microcontrollers, each responsible for the shaft encoders of one leg. When the 80C167 microcontroller supplies a 5-bit address specifying one encoder, the divider board returns a 16-bit number representing the current position of the corresponding shaft relative to the zero position at power on. The zero position can also be reset by a command from the main controller.

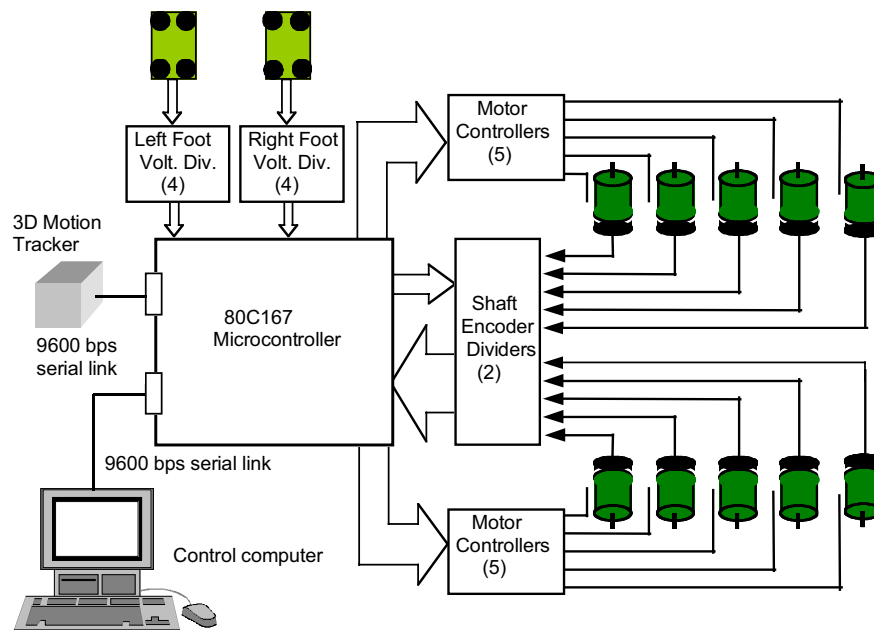


Figure 2. Block diagram of TarBaby circuits

### 2.2.1 Motor Driver Circuits

The motors of each leg are driven by a motor driver board, consisting of five A3952SW full-bridge pulse width modulation (PWM) motor drivers. These drivers accept enable and direction signals, with the controlling PWM signal from the 80C167 imposed on the

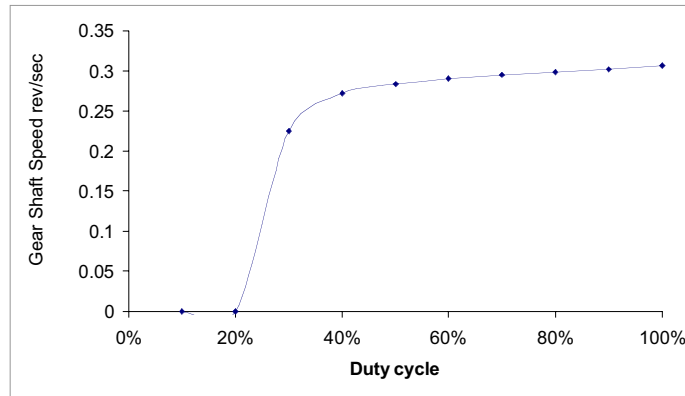


Figure 3. Range of unloaded shaft speeds available from motor driver circuits.

enable line. Figure 3 shows that an unloaded 180:1 gearhead motor can be driven at duty cycles of 20-100%. Under load, the range of usefully different speeds is considerably less but difficult to characterise, because during walking motions, the joint loads vary over a wide range. Power for the main microcontroller is taken from a 3Ah 12V lead acid gel battery located near the top of the torso for a high centre of gravity of the machine, which helps the balance problem. All other circuits are powered by a 12V laboratory power supply via a 3m cable. 5V signal levels are supplied by individual voltage regulators on each board. The 80C167 communicates with the system's Pentium control computer via a 9600 bps RS232 serial link at port P1. A second RS232 serial link at port P2 is used to transfer data from the motion tracker unit.

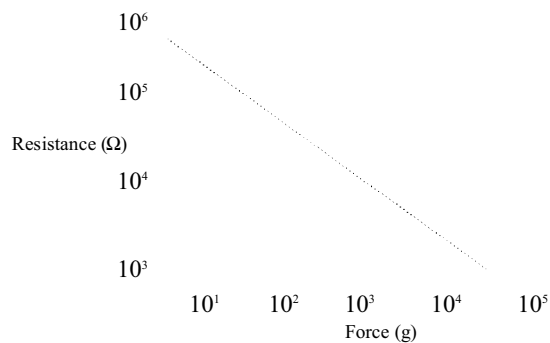


Figure 4. Characteristic performance of force sensing resistors

### 2.2.2 Force Sensor Circuits

Foot placement, and in particular a good contact with the supporting surface, is essential for stable walking. To measure pressures operating on the feet, semiconductive polymer force sensing resistors are attached at each of the four corners of the two feet. These are disks approximately 2.5cm in diameter, and have the property that the resistance across them varies in direct proportion as the force applied to them increases. Figure 4 shows how the relationship between force and resistance is linear within the range of forces experienced during ordinary walking motion. Using a simple voltage divider connected to a 10-bit A/D converter in the 80C167 microcontroller, a reliable indication of foot forces can be recorded.

The force sensors drive eight 10-bit A/D converters on the main microcontroller via two small voltage divider boards. Each board consists of four voltage dividers featuring multiturn potentiometers for signal level adjustment of each channel. With the weight on the feet, forces on a given force sensor is approximately within the range of  $10^3$  to  $10^4$  g. The multiturn potentiometers enable the output voltages to be shifted into the centre of the working range of the A/D converters.

### 2.2.3 Body Attitude Sensor Circuits

Orientation of the robot's torso is measured by a CyberTrackII motion tracking module. This is a self-contained, sourceless, hybrid device combining a flux-gate compass with electrolytic fluid tilt sensors. It generates packets of data representing three angles - pitch, roll and yaw on demand. Yaw is measured over  $360^\circ$  using a 3-axis magnetometer. Pitch and roll are measured over a range of  $\pm 45^\circ$  using 2-axis inclinometer. A built-in processor performs the necessary transformations to return valid yaw values given pitch and roll inputs. Data packets are delivered on request via a second 9600 bps serial link to the microcontroller at port P2.

## 2.3 Software Systems

The speed and memory capacity of the 80C167 microcontroller was a limiting factor. The 80C167 is not fast enough to run the learning algorithms required by the project. This constraint mandated that processing be split between an external controlling computer and the on-board microcontroller, with high-level commands and learning calculations computed externally and low-level functions and servo loops computed on-board. The two machines communicate by a serial communications link. The link is asynchronous and may be operated at speeds as high as 19,200 bps. This constraint required that the controlling computer and on-board processor functionality be partitioned in order to allow the data rate to be low enough to be accommodated on the serial link. To summarise, the on-board 80C167 microcontroller performs the following tasks:

- providing enable and direction signals to the motor controllers
- collecting sensory and shaft encoder data
- communicating with the control program on the control computer.

while the Pentium external controlling computer is assigned the following tasks:

- commanding the service program on the on-board microcontroller.
- generating high-level motions from the approximate walking heuristic
- performing learning calculations

The software was written in C using the Keil  $\mu$ Vision development environment for the 80C167 and Microsoft Visual C Studio for the Pentium machine. The main() function in the 80C167 code initialises the motor control subsystem (which includes starting a regular interrupt routine) and then executes the server loop which provides RS-232 communications via the P1 serial port.

### **2.3.1 Motor Control Processes**

The low-level motor control software provides functions for shaft position monitoring, PWM generation and motor servo control. Shaft position monitoring consists of a set of low-level service functions returning information from the shaft encoder divider board such as the angle of a specified joint from zero point in 0.xx degree units. Each of the ten servo channels runs a PID servo loop, ensuring that each motor reaches its goal smoothly with a minimum of overshoot. In practice, the goal velocity is regulated by the control host's geometry generator (see below) since the servo controller receives new goals 50 times per second.

The motor PWM output is generated by a 10 kHz software interrupt. This allows the power to each motor to be varied in one percent steps with an update rate of 100 pulses per second. The allowable pulse width varies from 0% (off) to 100% (on). The motors are controlled by a PWM signal, delivered to the motors via the H-Bridge circuits. This arrangement allows the motor torque/speed and direction to be controlled by the microprocessor. The PWM signal is looked up from a bit table of 100 rows, which allows each interrupt to be kept short by requiring only a simple logical operation on the motor control output port, according to the state of a bit in the table.

### **2.3.2 Sensory Loop**

Sensory data is read directly from [serial] data ports P on demand, with the exception of shaft encoder values and derived values. The serial ports are scanned as part of the 10 kHz servo PWM interrupt. An efficient state machine controls the setting up and acquiring of channel data in sequence.

Because the 80C167 has only a single serial port, used by the serial link to the external computer, a second port P2 was implemented in software using an interrupt-driven state machine to encode and decode the asynchronous serial data. This was needed to set up and read data packets from the CyberTrackII motion tracking module.



### 2.3.3 Communications Link

The serial interface has been designed so that when the TarBaby is started, the serial interface initially uses a line-oriented ASCII protocol that allows commands to be sent to the robot by typing them on the external control computer, with the responses being echoed in ASCII. This is used to perform basic health checks on the machine before the external control program is started. Then a binary protocol is used for subsequent real-time control of the Tarbaby from the external program. A pair of mode-changing commands switch between the ASCII and binary protocols.

The Tarbaby responds to simple goals for each motor channel and responds with the current sensor state of the machine. A complete exchange on the serial link is completed 50 times per second. This response time requirement made the binary protocol necessary. When operating in real-time mode, sending of commands and receiving of state from the Tarbaby is fully overlapped and synchronised with the 50 Hz loop. Typically, when a response is received, the external controller can process the basic geometry quickly enough to send the updated targets down to the robot on the next or second next poll cycle. Setup and configuration commands from the external controller interrupt the real-time operation so that such tasks as zeroing the leg positions can be accomplished.

The subsystem handling communications with the 80C167 microcontroller is based on a client/server model. The client is the external control computer and the server is the microcontroller. The server software consists of a loop which reads an input line from the RS-232 port, P1. Every request is exactly one line of ASCII characters. The requests are semi-human-readable in that the characters are all printable and numbers are represented in the normal fashion. However, commands are parsimoniously-chosen single characters to reduce communication time. Every request elicits a single line response. If the request is an action, the response is simply, 'ok', otherwise the requested data are returned on a single line.

The requests available can be summarised as:

- get the version of the server software
- get motor shaft encoder values
- set/get motor torque values
- set/get PID desired shaft values
- get PID error values
- set/get PID controller constants
- set/get/use a configuration register.

The server routine is suspended every time the motor subsystem interrupt routine is executed. As the motor subsystem is given more tasks to perform (eg. more PID controllers activated), less computational power is available for the server. This affects the server response times, but may be ameliorated by judicious selection of the M and T parameters, which set the granularity of the PWM.

### 3 Dynamic Walking Behaviour

A good solution to the problem of active-balanced walking on two legs with small feet could eventually lead to robots capable of greater speed, agility and energy efficiency than are available today. Machines with legs could potentially access locations beyond the reach of any machine depending on wheels for locomotion. Roughly human-sized and shaped robots that walk on two legs may be the optimal shape for moving inside buildings designed for humans, and potentially more acceptable to their human occupants. Dynamic bipedal walking is also an interesting problem from a purely control engineering point of view. For these and other reasons, research on bipedal robots continues.

An ideal solution to bipedal walking problem should satisfy the following criteria:

*Simplicity.* It should not unduly complicate to the mechanical or computational complexity of the whole machine.

*Efficiency.* Animal locomotion systems use legs for efficient energy usage. A solution that optimised power consumption would be attractive for self-contained robots.

*Robustness.* Not only should falls be minimised, but the solution should afford the best possible performance on uneven slopes, irregular floor surfaces and carrying different loads.

*Parameterisability.* Offered as a service to higher level control functions, the solution should offer a range of speeds, directions and possibly gait styles. Walking should be able to be stopped and started at arbitrary times without compromising stability.

So far, bipedal robots have only partially achieved these goals. Fair control of some active-balance bipeds has been accomplished by building a detailed dynamic model of the moving linkages, then designing a controller that generates the joint angle vector sequences needed to regulate stability and optimise forward velocity as derived indirectly from inertial sensors in the torso and contact and/or orientation sensors on the feet. For example, Raibert tackled the problem by first developing a single-legged hopper (Raibert, 1986), then generalising that to an agile hopping biped (Playter & Raibert, 1992).

Furushu and Sano used two independent dynamic models for lateral and sagittal plane motion control in their BLR -G2 and -G3 bipeds (Furushu & Sano, 1990; Sano et.al., 1992). Yamaguchi and coworkers (1996) designed specially-instrumented feet with clinometers, gyroscopes and linear potentiometers for their WL-12RVII biped, which enabled walking speeds of up to 0.38m/sec on floor surfaces with vertical irregularities of  $\pm 16$ mm and foot tilt disturbances of  $\pm 3^\circ$ .

Honda's P2 and P3 humanoids, have produced impressive results. Though not strictly dynamically-balanced bipeds by the above definition (they keep their centre of mass directly above their supporting foot at all times), they do depend on a complex dynamic model based on the Zero Moment Point algorithm (Takanishi, et. al., 1990). They are claimed to walk at speeds of up to 2km/hr, carry loads of 2 to 5kg and use staircases, and gather information from a head-mounted vision system (Honda Motor Corp, 1996). Smaller versions of this type of humanoid are also under development by Sony Corp.

However, this dynamic modelling approach has disadvantages. First, detailed dynamic models are generally complicated to design, build and tune (violates simplicity). Specifying optimal leg vectors for all possible combinations of commanded input parameters, current postures and desired motions is certainly non-trivial. (For example, the Honda robots use a completely pre-programmed sequence of steps in order to move.) Second, if something in the physical system changes, the model may no longer generate suitable movement. Linkages and motors deteriorate, circuits behave differently with variations in temperature or battery voltage, and environmental conditions of a robot could be subject to uncontrollable change (violates robustness). Third, a solution developed for a given machine could be difficult to adapt to a biped of a new and different design.

Adaptive control methods, in particular those that can rapidly learn control functions online from error signals fed back during practice offer hope of dealing with these problems. In the past decade, reinforcement learning paradigms such as actual return (Jaakola, et.al, 1995), temporal differences (Sutton, 1988) and model-based learning (Zazala, 1996) have been combined with such adaptive mechanisms as memory-based lookup (Schaal & Atkeson, 1994), neural networks (Warwick, 1996) and fuzzy logic controllers (Verbrugen, et. al., 1999) to tackle the problems of robot control, which are characterised by noisy or incomplete sensor information, non-Markovian state dependencies and high-speed learning/performance requirements.

Neural network controllers, in particular, have contributed to robust dynamic walking in bipedal robots. Salatian and Zheng simulated off-line and on-line neural network adaptation of bipedal gaits used by their experimental SD-2 8-link biped (Zheng & Shen, 1990) on flat surfaces into gaits suitable for slopes (Salatian & Zheng 1992). Inspired by that work, Kun and Miller achieved respectable parameterisability and robustness at walking speeds of up to 10cm/s on flat surfaces with a small biped that used three Cerebellar Model Articulation Controllers (CMAC) to learn key parameters of a side-to-side rocking gait. Importantly, no detailed engineering models of the machine was needed and the training times required for stable walking were short, in the order of 60 minutes (Miller, 1994; Kun & Miller, 1996). The TarBaby biped is in turn inspired by the work of Kun and Miller.

From a machine learning perspective, the problem requires a vector of 10 2000-level motor position updates approximately every 20 milliseconds as a nearly continuous learned function of 10 counts from the shaft encoders, 8 force sensors in the feet, and 3 raw attitude values from the motion tracker. A problem space of that magnitude is probably beyond the capability of known reinforcement learning algorithms, because i) training visits to the problem space are necessarily sparsely distributed, so generalisation is difficult ii) memory demands in such systems suffer a combinatorial explosion (Sehad, 1998) and iii) on-line learning imposes serious computational demands on computation per update cycle (but see Schaal & Atkeson, 1994).

Another challenge is thrown down by the passive dynamic systems engineers, who argue that much of the power efficiency and adeptness of natural bipedal walking can be explained in terms of tuned mechanical oscillators formed by the limbs and their springy joints. To support this contention, they have designed mechanical models which achieve limited forms of bipedal motion down slopes with no sensors, no actuators and no electronic control (McGeer, 1990; Coleman & Ruina, 1998). This suggests that with

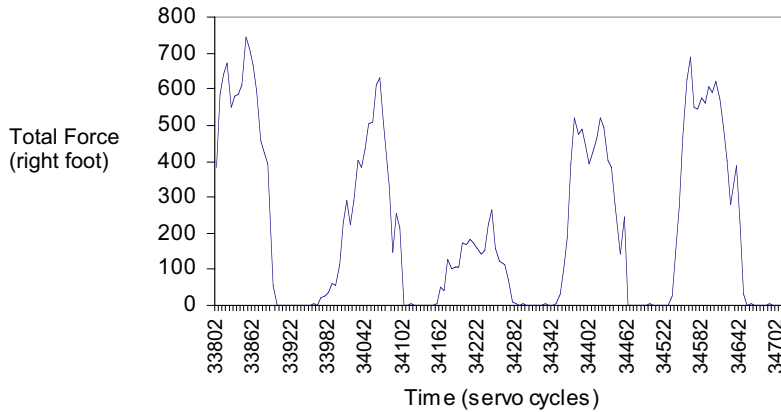


Figure 5. Sum of all four FSR signals on one foot during 5 steps generated by a rudimentary inverse kinematic model of walking that was not phase-synchronised to the natural dynamics of the robot and was not tuned by adaptive learning. Servo cycle labels are 1.08 seconds apart.

better mechanical design powered, controlled bipeds could be simplified. How can these developments be accommodated in TarBaby's design?

Rather than trying to develop a control algorithm that learns the entire transformation from raw sensory data to motor commands, the approach favoured here aims to use machine learning processes to modulate a simple gait generator. In the TarBaby design, a basic generator cycles through eight phases of walking motion while respecting the natural inverted-pendulum dynamics of the robot. Transitions from one phase to the next in sequence are to be triggered when learned sensory thresholds are exceeded. Instead of learning to control the joints then, the function approximators will learn how to control a program which controls the joints.

Figure 6 shows the basic design. At the time of writing, software development has only completely developed and tested the lowest three layers. The only fixed reflex currently in use is a feedback loop which reads foot forces from the eight FSR sensors on the feet and generates ankle pitch and yaw commands in order to minimise the centres of force acting on the feet. This is done because when the robot has both feet on the floor, it will not be maximally stable unless the ankles are adjusted so that the soles of both feet are flat on the floor. Centre of force is defined as follows: let the force on the single right-front sensor of foot  $i$  be represented by the expression  $F_{i, rf}$ . A convenient lateral plane center of force in the range  $[-32, 32]$  may now be represented as

$$\text{LatCoF}_i = \frac{F_{i, rf} + F_{i, rr} - F_{i, lf} - F_{i, lr}}{(\text{ForceF}_i + 32)/32}$$

This will be a maximum when most of the force is on the right side of the foot and minimum when it is on the left side. Another measure for the sagittal plane center of force may be similarly defined such that it reaches a maximum when most of the force is on the front of the foot.

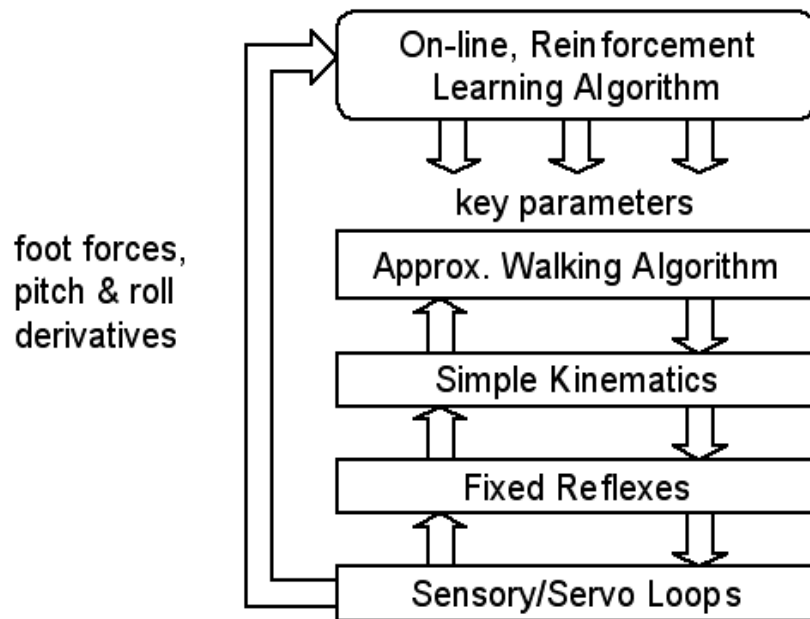


Figure 6. Software architecture consists of an approximate walking algorithm built out of kinematic functions, which in turn are built out of calls to the sensory/servo loop and fixed reflex motor actions. An on-line learning process tunes the approximate walking algorithm via parameters such as lean magnitude. Reinforcement feedback is taken from foot force distribution and body attitude sensor data.

Interestingly, with hip and knee positions set within a range of reasonable values for stationary standing, the ankle motions generated by the reflex are enough to allow the robot to balance, even if gently pushed. This bodes well for the ankle reflex loop, which may be able to simplify the Kun & Millar (1996) design by reducing to two the number of learning modules required for walking. These would use on-line reinforcement learning to predict the lateral and sagittal lean magnitude (ie. side to side rocking and back-to-forward rocking) of the machine required to permit forward steps of a given height and reach.

#### 4 Research Issues

What kind of adaptive function approximation can be used to tune the approximate walking algorithm? Ideally, such a learning mechanism would be able to avoid serious errors such as becoming trapped in local minima, deal with the (reduced) problem space presented by the approximate walking algorithm, quickly converge on a solution during on-line training and output updates of the learned function at high frequency once learning achieved a satisfactory level. Cerebellar Model Articulation Controllers (Glanz, et.al., 1991) are artificial neural networks inspired by studies of the function of human cerebellum, a brain structure implicated in the fine-tuning of muscle activity. According to that review, a CMAC has a number of useful characteristics, including:

- Accepts multiple integer-valued input vectors (typically with 5 to 20 elements valued at one of 200 levels) and returns real-valued vectors (typically 1 to 10 components).
- Generalises by returning approximately correct outputs even to input patterns it has not seen.
- May be capable of learning any well-behaved function to some degree of accuracy.
- Even large CMACs can be trained in reasonable length of time compared to other kinds of neural networks.
- Function computation time is short as it requires little calculation.
- With standard LMS weight change equations, cannot become caught by local minima.

We have been able to reproduce the results of a simple CMAC experiment described in the Glanz review paper, using code obtained from the University of New Hampshire's Electrical Engineering Department. The experiment demonstrates the power of a CMAC to generalise and rapidly learn a non-linear function. An input vector of 1 element is randomly assigned to an integer between 1 and 100. The desired output is one if the input is in the range 30-70, zero otherwise. The threshold on the real-valued output is set at 0.5, above which the actual output will be taken to signify one and below which it will be taken to signify zero. Figure 7 shows the result of one such learning trail. A CMAC with 10,000 receptive fields and moderate generalisation rapidly duplicates the desired function, including, at the lower numbers of iterations, predictions for values which had not yet appeared. In less than 500 trials the function had been faithfully reproduced.

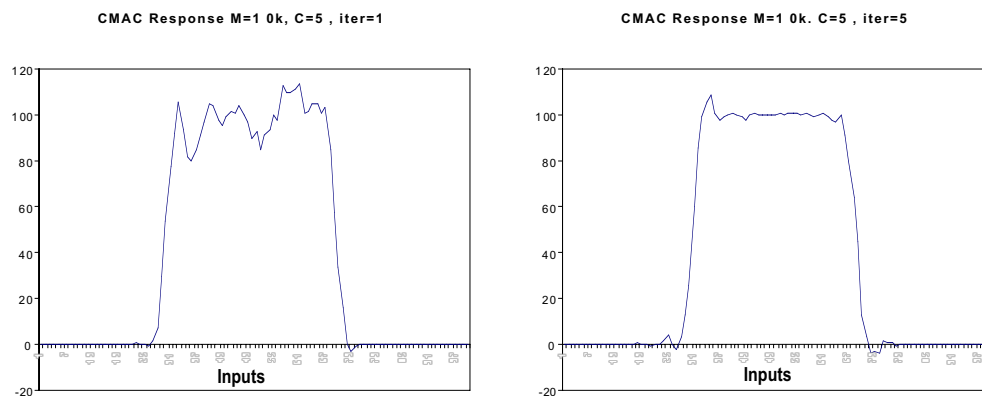


Figure 7. Learning of a square wave function by test CMAC with 10,000 receptive fields, generalisation parameter of 5 after 100 random input values (left) and 500 inputs (right). Scale of the vertical axes is 100x output values. Further inputs did not significantly improve the fidelity of the learned function.

Experiments designed to apply the learning potential of such devices to the task of . Early experiments have suggested that acceleration data from the CyberTrack II attitude sensor may be confounded by its constantly changing acceleration as the torso moves, and it may be necessary to gyroscopically stabilise the sensor to remove this effect. This is done reluctantly, since it tends to complicate the robot's mechanical design which is undesirable with respect to the first principle outlined in 3. Possibly extra accelerometers could be added to allow compensation to be made in software.

Two function approximators will be used to simultaneously generate parameters which the approximate walking algorithm will use to set lateral and forward-backward rocking amplitude for steps of a specified length and rate. Sense data representing both body attitude and forces operating on the feet will be used as input. Ankle reflexes will be in operation but slightly weakened to allow flexibility of the feet during stepping. Quality of walking will be assessed by periodic smoothness of body attitude signals, difference between commanded and actual stepping parameters, and fall rate.

## References

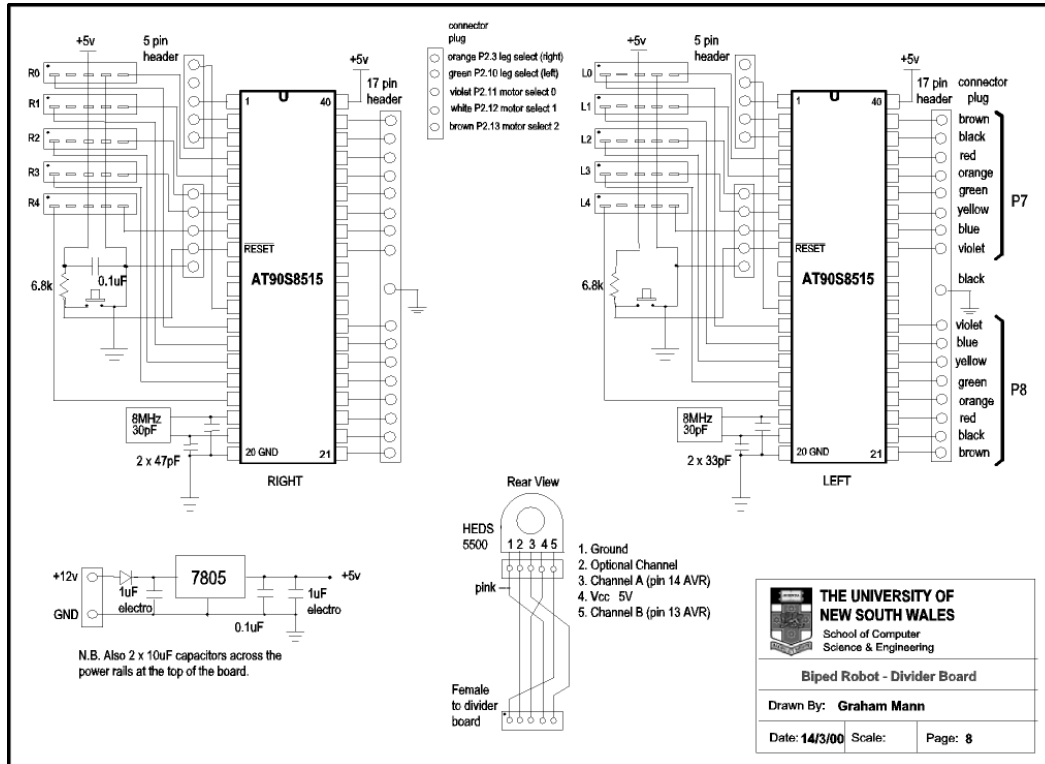
- Coleman, M.J. & Ruina, A. An uncontrolled walking toy that cannot stand still. *Physical Review Letters*, 1998, 80:3658-3661.
- Junji Furusho, J. & Sano, A. Sensor-based control of a nine-link biped. *International Journal of Robot Research*, April 1990, 9(2):83-98.
- Glanz, F.H., Miller, W.T. & Kraft, L.G. An overview of the CMAC neural network. *IEEE Conference on Neural Networks for Ocean Engineering*, pages 301—308. Washington D.C., 1991.
- Honda Motor Corp. Unpublished notes from 15<sup>th</sup> *International Joint Conference on Artificial Intelligence*, Nagoya, Japan, 1997 and Honda website <http://www.honda.co.jp/english/technology/robot/index.html>
- T. Jaakola, J., Singh, S.P. & Jordan, M.I. Reinforcement learning algorithm for partially observable Markov decision processes. In *Advances in Neural Information Processing Systems 7*. Morgan Kaufmann, 1995.
- Kun, A. & Miller, W.T. Adaptive balance of a biped robot using neural networks. *Proceedings of the 1996 IEEE International Conference on Robotics & Automation*, pages 240-245, Minneapolis, Minnesota, April, 1996.
- Mann, G. Stability Control in an Actively Balanced Bipedal Robot. *Proceedings of the Australian Conference on Robotics and Automation*, 191-195. ARAA/ Monash University, Melbourne Australia, August, 2000.
- McGeer, T. Passive Dynamic Walking. *International Journal of Robot Research*, April, 1990, 9(2):62-82.
- Miller, W.T. Real-time neural network control of a biped walking robot. *IEEE Control Systems*, February, 1994, 40(1):41-48.
- Playter, R.R. & Raibert, M.H. Control of a biped somersault in 3D. *International Symposium on Theory of Machines and Mechanisms*, pages 669-674, Nagoya, Japan, September, 1992.
- Raibert, M.H. *Legged Robots that Balance*. Cambridge, MA. MIT Press, 1986.

- Salatian, A. W. & Zheng, Y.F. Gait synthesis for a biped climbing sloping surfaces using neural networks -II. Dynamic Learning. *Proceedings of the 1992 IEEE Conference on Robotics and Automation*, pages 2607-2611, Nice, France, 1992.
- Sano, A., Furusho, J. & Yabuki, W. Biped walking on floor and stairs based on torque distribution. *International Symposium on Theory of Machines and Mechanisms*, pages 663-668, Nagoya, Japan, September, 1992.
- Schaal, S. & Atkeson, C. Robot juggling: An implementation of memory-based learning. *Control Systems Magazine*, 14(1):57-71, 1994.
- Schad, S. Neural reinforcement learning for robot navigation. In Lakhmi. C. Jain & Toshio Fukuda (Eds.) *Soft Computing for Intelligent Robotic Systems*. Physica-Verlag, Heidelberg, 1998, pages 159--184.
- Sutton, R.S. Learning to predict by the methods of temporal differences. *Machine Learning*, 1988, 3:9-44.
- Takanishi, A., Lim, H., Tsuda, M. & Kato, I. Realization of dynamic biped walking stabilized by trunk motion on a sagittally uneven surface. *IEEE International Workshop on Intelligent Robots and Systems, IROS*, 1990.
- Warwick, K. An overview of neural networks in control applications. In A.M.S. Zalzal and A.S. Morris.(Eds.) *Neural Networks for Robotic Control*. Ellis-Horwood, London, 1996, pages 1--
- Verbruggen, H.B. & Babu ka, R. (Eds.) *Fuzzy logic control : advances in applications*. River Edge, New Jersey: World Scientific, 1999.
- Yamaguchi, J., Kinoshita, N., Takanishi, A & Kato, I. Development of a biped walking robot adapting to the human's living floor. *Proceedings of the 1996 IEEE International Conference on Robotics & Automation*, pages 232--239, Minneapolis, Minnesota, April, 1996.
- Zalzal, A.M.S. Model based adaptive neural structures for robotic control. In A.M.S. Zalzal and A.S. Morris.(Eds.) *Neural Networks for Robotic Control*. Ellis-Horwood, London, 1996, pages 80--105.

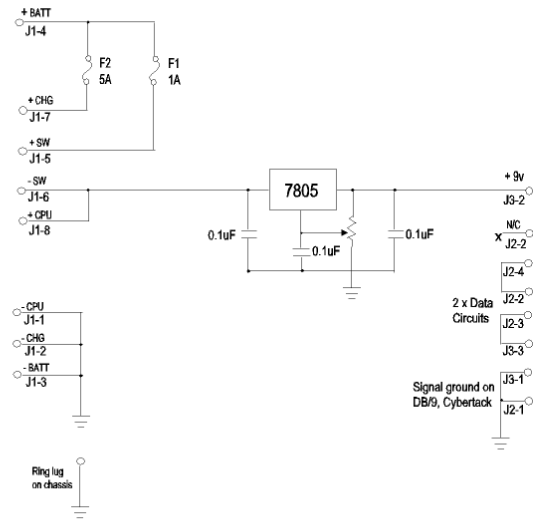



# Appendix A Circuit Diagrams

## Encoder Divider Board

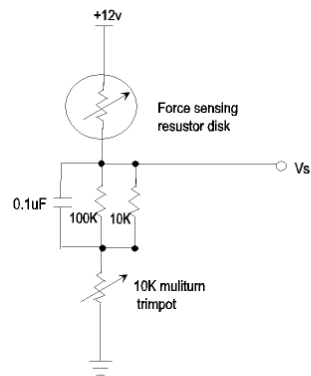


## Power Distribution Board




 <b>THE UNIVERSITY OF NEW SOUTH WALES</b> School of Computer Science & Engineering		
Biped Robot - Power Distribution		
Drawn By: <b>Graham Mann</b>		
Date: 23/12/00	Scale:	Page: 9

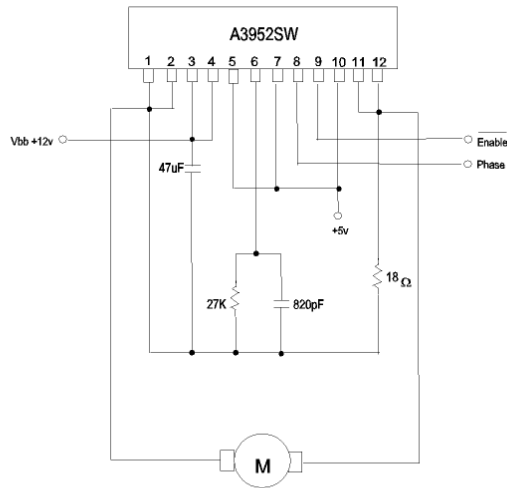
## Force Sensor Boards




N.B. Circuit is duplicated in 4 channels per board.  
Boards are bilaterally duplicated.

 <b>THE UNIVERSITY OF NEW SOUTH WALES</b> School of Computer Science & Engineering		
Biped Robot - Force Sensor Circuits		
Drawn By: <b>Graham Mann</b>		
Date: 23/12/00	Scale:	Page: 10

## Motor Driver Boards



N.B. Circuit is duplicated in 5 channels per board.  
Boards are bilaterally duplicated.

	<b>THE UNIVERSITY OF NEW SOUTH WALES</b> School of Computer Science & Engineering
Biped Robot - Motor Drivers	
Drawn By: <b>Graham Mann</b>	
Date: 23/12/00	Scale: Page: 11