Electronic Control of a Two-Dimensional, Knee-less, Bipedal Robot

Final Report for MAE 490
under Professor Andy Ruina
June 22, 2005

Matt Haberland
401 Thurston Ave.
Ithaca, NY 14850
(607) 257-6576
(through August 2005)

1291 Shaker Woods Rd.
Herndon, VA 20170
(703) 904-8566
(after August 2005)

Abstract

Representing advances in dynamic understanding and mechanical recreation of the human body, the Marathon Walking Robot successfully demonstrates a stable walking cycle using the same control components as machines created in high school competitions and code simple enough to be understood by participating students. Encouragingly, further steps to be taken towards more robust and efficient locomotion involve little more than planned mechanical improvements, more sophisticated but readily available feedback, and more intelligent control afforded by existing algorithms. In addition to and perhaps even more significant than contributions to machine walking, further efforts will pioneer new techniques for uniting simulation with the physical world to aid in the design of future machines.

Introduction

From prehistory to present, humans have always been fascinated by themselves. Motivations range from vanity to applied science to the purest curiosity, but the interest remains and humans are a prime subject of research. One area of investigation that is beginning to make significant progress is that of human locomotion – how people move, and specifically how the body moves itself from place to place. Books and movies reveal that popular culture would consider it desirable to use the human body as a model for machines, the dream being that the machine would be as versatile a living person. One of the most notable attempts toward this end is Honda’s Asimo humanoid robot, a machine that can walk, manipulate objects, recognize people, and perform other basic human-like functions. However, a machine based on the technology of Asimo is unlikely to fulfill this vision because Asimo merely mimics human motions with precise angular control of every joint. As a result, Asimo is versatile and robust but expensive and tremendously inefficient; such a machine can barely function for half an hour between recharges – let alone on three meals a day. It is possible that scientists will discover revolutionary ways of producing and storing energy such that battery life is no longer a concern, but in the meantime it is useful to explore more efficient ways of replicating human movement, both as a step towards the aforementioned grand vision and for the better understanding of the human body itself, which evolved under tight efficiency constraints. Other machines, such as the unpowered “passive dynamic” walkers pioneered by Tad McGeer, are so well suited to walking that the only energy they consume is provided by gravity and no electronics are required for control; the motions are inherent to positions of the joints and mass distributions of the limbs. The limitation in this case is that the machines are only capable of a single particular motion, they cannot be controlled, they are not very robust, and they can only move downhill. The Human Power and Robotics lab at Cornell University works to merge these advantages of both classes of machines to create a robot that is efficient, versatile, and robust. The robotic subject of the following paper represents an advancement toward this end, specifically in the robustness and consistency of an efficient powered walker and the agreement between robotic walking simulation and experiment.

Methods

The mechanical structure of the two dimensional, knee-less, bipedal walker was originally created by Gosse [1] in 1998. Energy was to be added to the system by actuation of the ankles alone. A control system was implemented in 2000 by Yevmenenko [2], and again in 2003 by Yeshua [3], but the robot never achieved a walking cycle. As part of the Marathon Walking Robot project in 2005, a motor was added for hip actuation by Seidel and Strasberg [4] and the feet were replaced by Harry [5]. In addition, the entire electronic control system, including all hardware and software, was replaced, hence the need for documentation provided in this paper.

Hardware

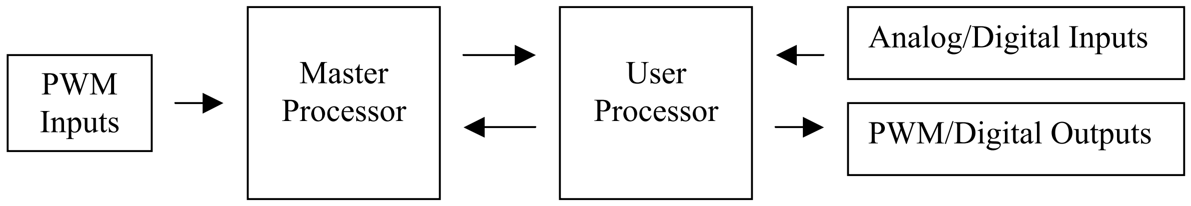
Robot Controller

The Innovation FIRST Mini Robot Controller was chosen to control the robot because it provides a user-friendly interface between sensors and a microprocessor. Eliminating the need to order and prepare dozens of discrete components, the Robot Controller combines a programmable chip, programming port, power terminals, and a wide variety of pins for signal input and output in a single, convenient package. Prewritten functions, instructive documentation, and useful software facilitate generation of custom programs.

Feature:	**Details:**	**Use:**
Speed	10MIPS	Sufficiently fast processor means precise and accurate robot control
Power	Approx. 1W	Low power consumption means greater robot efficiency and longer walking time
Size	3.4” X 4.6” X ¾”	Small size means low weight and more space for other robot parts
Language	C	Powerful, familiar language means flexibility and easy use
Digital Inputs	22 Max	Used to take input from limit switches
Analog Inputs	16 max, 10-bit	Used to take input from potentiometers
PWM Outputs	8	Used to control motors

Internal Processors

The robot controller actually contains two Microchip 18F8520 PICmicro microcontrollers – one programmable User processor that runs the code for the robot, and one Master processor that relays radio control signals to and monitors the operations of the User processor. The Master processor and User processor only communicate once

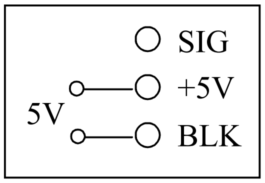


every 17ms, which limits response to radio input. However, all digital/analog in and PWM ports are wired directly to the User processor, providing for immediate communication between the computer and onboard electronics. Other than this, the workings of the Master processor are irrelevant to the user of the system and thus are not disclosed by Innovation FIRST; in practice it is only necessary to understand the operations of the User processor.

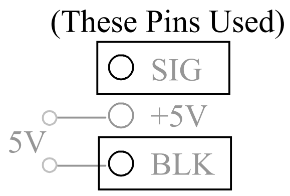
Ports

Each of the 16 Digital In/Out – Analog In ports can be configured in the code as either a digital in, digital out, or analog in. Each of the ports labeled "Interrupt" is permanently configured as a digital in, but when triggered can be used to fire interrupt conditions in the code. The PWM outputs can be used to send PWM signals to motor controllers using prewritten functions. The robot controller also features solenoid outputs and PWM inputs, but they are not used on the robot.

Digital In/Out – Analog In

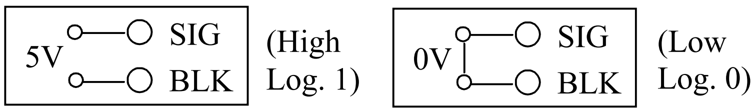


The Digital In/Out – Analog In ports have three pins each – one "SIG" or signal pin, one "+5V" or "high" pin, and one "BLK", black, or "low" pin. A 5V potential difference is always maintained between the "+5V" pin and "BLK" pin; if these pins are shorted the mini robot controller automatically shuts down. Depending on the state of the digital/analog input device, however, the potential difference between the "SIG" pin and the "BLK" pin may vary. The voltage between these pins is measured by the robot controller, converted to either a 1-bit or 10-bit value, and can be referenced in the code as input from the sensors.



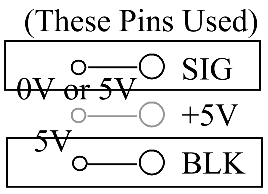
Digital Input

When configured as a digital input, only the "SIG" and "BLK" pins are used. The robot controller measures the voltage between the pins as either high (5V) when the circuit between them is open or low (0V) when the circuit is closed. The high measurement is converted to a logical 1 and the low measurement is converted to a logical 0 for use in robot code. It has been found experimentally that the controller can only detect a change in the digital input value a few hundred times per second.



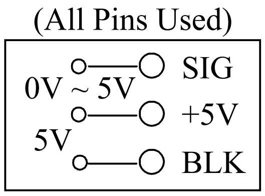
Digital Output

When configured as a digital output, only the "SIG" and "BLK" pins are used. The robot controller outputs either a logical "high" (5V) voltage between the pins or a logical "low" (0V) voltage between the pins as specified in the code. The robot can change the signal at a maximum rate of about 1 kHz.



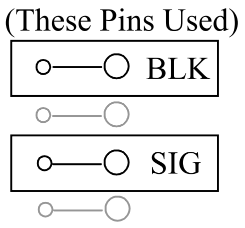
Analog Input

When configured as an analog input, all three pins are used. The "+5V" and "BLK" pins provide a constant reference voltage for the sensor, while the voltage between "SIG" and "BLK" varies depending on the state of the sensor. The robot controller measures the voltage between these pins (0V-5V) and converts this to a 10-bit value (an integer from 0-1023) for use in the code.



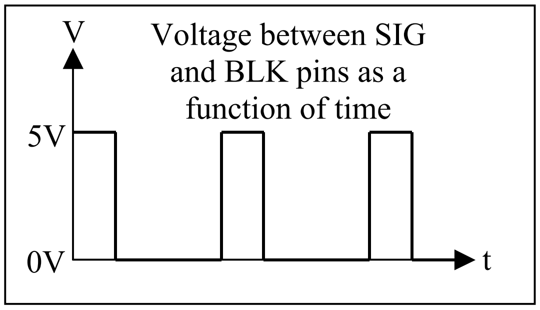
Interrupt Digital Inputs

The "interrupts" are ports that are permanently configured as digital inputs, but can be used to fire an "interrupt" condition in the code when they are triggered. When the logical state of these ports changes, the processor immediately jumps to a specific portion of code, the "interrupt handler". Currently, the ports are being used as regular digital inputs; the interrupt feature is not being used.



PWM Output

A PWM (Pulse Width Modulation) output sends signals to a voltage controller which in turn sends a specified voltage to a motor. A PWM signal is a series of digital pulses which vary in length depending on the desired motor voltage. The voltage controller (which can be a simple H-Bridge or hobby speed controller) amplifies the voltage of these signals and provides current to the device. This pulsing voltage simulates a constant voltage proportional to the percentage of digital "high" time, the "duty cycle". The PWM ports have four pins – one BLK pin, two supply voltage pins, and one SIG pin. Only the SIG pin and BLK pin are used in this robot, in much the same way as a simple digital output. The purpose of using a PWM port instead of a regular digital out is that routines are already programmed into the microprocessor for generating proper PWM signals corresponding to an 8-bit integer specified in the code.

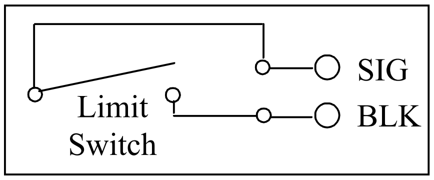


Radio PWM Input

Radio control is not critical for this robot; the only control an operator has is turning. The current radio control system completely bypasses the IFI controller, so the radio PWM inputs are not being used. However, the IFI Radio PWM inputs can receive signals from a hobby radio receiver through standard PWM cables. When the radio receiver is plugged into this port and a signal is received from the radio transmitter, the Master processor converts the signal into an 8-bit value which is relayed to the User processor every 17ms for use in the code.

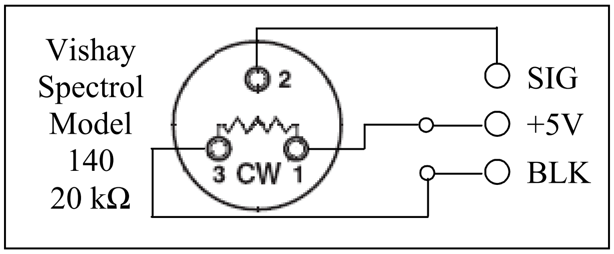
Sensors

Limit Switches



There are two limit switches, one at the heel of each foot. Each limit switch is simply a contact switch that is pressed when its respective heel is supported by the ground. There are two wires leading into the limit switch – one "signal" wire, and one "black" wire. The signal wire plugs into the "SIG" pin of a digital in on the robot controller, the black wire plugs into the "BLK" pin of the digital input. When the limit switch is not pressed ("open"), the voltage between the two pins is 5V and the robot controller reads this as a single bit – "high" or "1". When the limit switch is pressed ("closed"), an electrical connection is made between the two wires inside the switch, creating a short circuit. This forces the voltage between the two pins to jump to zero, and the robot controller reads this as "low" or "0". So when the heel of the robot is in the air, the limit switch is not pressed and there is a 5V potential difference between its lead wires, and thus the robot controller reads a "1" for that digital input. When the heel of the robot is in contact with the ground, the limit switch is pressed, shorting the lead wires, so there is 0V between them and thus the robot controller reads a "0". In this way the limit switches tell the robot how it is supported by the ground.

Potentiometers



There are three potentiometers, one at each ankle and one at the hip. Potentiometers are variable resistors with three pins. The right "reference" pin is connected to the "5V" pin of the robot controller, the left "black" pin is connected to the "BLK" pin, and the center "signal" pin is connected to the "SIG" pin. There is always a fixed 20 kΩ between the "reference" pin and the "black" pin, but turning the knob of the potentiometer changes the voltage between the "signal" pin and the "black" pin. When connected properly, the robot controller measures the voltage between the "signal" pin and the "black" pin and converts the measurement to a 10-bit value (an integer between 0 and 1023). For example, when the foot is in the air and is rotated to a fully retracted position, the voltage between the signal and black pin is .5V, corresponding to a value of 100. In this way the potentiometer tells the robot the position of the ankle.

Encoder

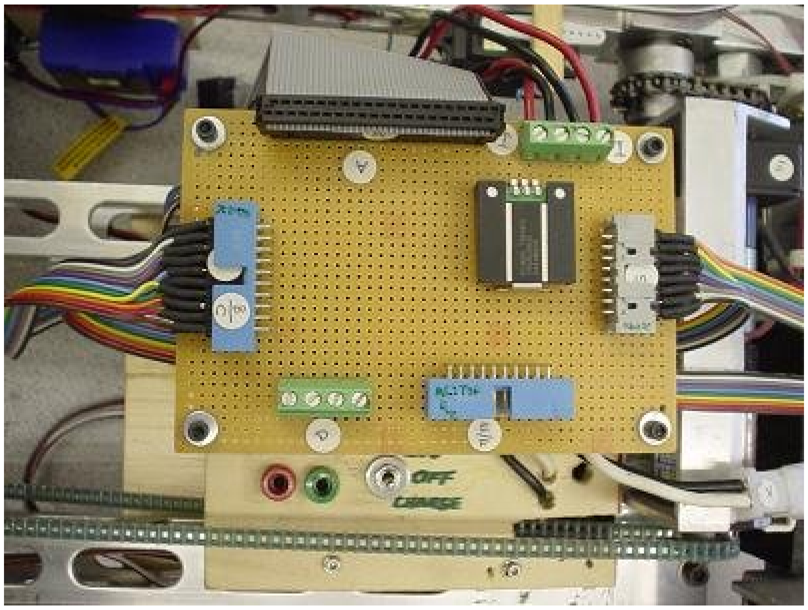
Encoders are mounted on each of the motors for measuring angular changes, but none of them are in use. An encoder uses a beam of light and a slotted wheel coupled to the motor shaft to detect changes in angular displacement. When the beam of light passes through a slot in the wheel, the encoder sends a pulse to the robot controller indicating a change in motor shaft angle. In practice the pulses came too rapidly to be processed by the robot controller.

Voltage Controller

The PWM output was originally connected to an H-Bridge Voltage Controller created by former lab consultant Mike Sherback. A PWM output sends square pulses with variable lengths corresponding to an 8-bit value (an integer between 0 and 255) at a fixed frequency. The H-Bridge boosts the voltage of the signal and backs it up with sufficient current to power the motor. By switching the power on an off in pulses, the H-Bridge simulates a controlled, constant voltage due to the inductive properties of the motor. The simulated voltage is proportional to the length of a pulse over the period of the signal – the "duty cycle" percentage. A separate two-bit signal (generated using digital outputs) controlled the function of the motor – forward, reverse, coast, or brake. However, the microprocessor routine that controls the PWM output only provides for signals between 2 and 40 kHz. Such frequencies proved too high for the inductive motor, resulting in low mechanical power output. The most recent voltage controllers are Innovation First Victor 884 Speed Controllers. A function written by Innovation First generates a PWM signal at a frequency of 120 Hz with a duty cycle measured to vary between 10% and 20%. Unknown internal circuitry of the speed controller does not merely relay an amplified signal to the motors, rather it converts this signal to voltage between –12V and 12V.

Interface Board

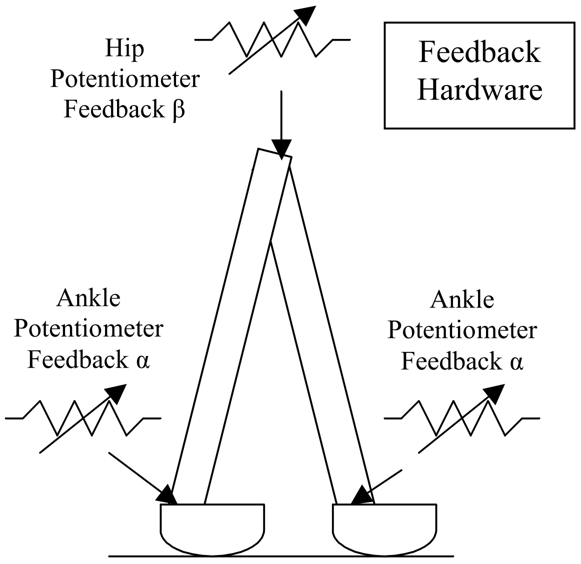
The interface board carries all signals to and from the robot controller via a 40-pin IDE cable and routes them to the proper sensors via smaller ribbon cables. Also included is an IC DC/DC converter that regulates voltage supplied from the battery down to 6V for the robot controller and radio receiver. Although the function of the interface board is simple, the detailed design is rather complicated and its fabrication using a breadboard, wire, headers, and solder was very time-consuming, so complete technical drawings are included as an appendix.



Software

Algorithm

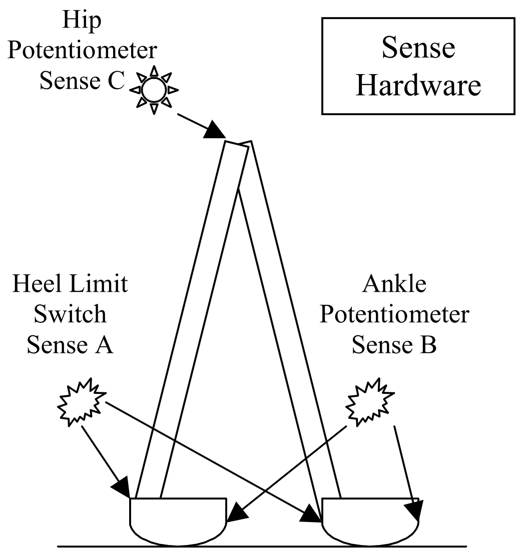
The most difficult part of programming the software was finding a simple but flexible and efficient algorithm. The robot is modeled as a state machine – at any given point in time the robot is in a particular state in which it only performs a certain subset of its possible actions based on the relevant feedbacks, and only switches to the next state when given the cue from one of its senses.


Hip Potentiometer Feedback β
Feedback Hardware
Ankle Potentiometer Feedback α
Ankle Potentiometer Feedback α

Feedbacks:

Feedback α: Ankle Angle
A potentiometer senses the angle of the ankle.

Feedback β: Hip Angle
A potentiometer senses the angle of the hip.


Hip Potentiometer Sense C
Sense Hardware
Heel Limit Switch Sense A
Ankle Potentiometer Sense B

Senses:

Sense A – Heel-Strike
The robot senses front heel-strike with a contact limit switch on the heel. When the robot's front heel strikes the ground at the end of a step, the limit switch is pressed and heel strike has occurred.

Sense B – Toe-Lift
The robot senses rear toe-lift with a potentiometer on the robot's ankle. When the ankle angle (Feedback α) reaches a certain value, the robot registers toe-lift.

Sense C – Hip-Switch
The robot senses hip-switch with a potentiometer at the hips. When the hip angle (Feedback β) reaches a certain value corresponding with the inner and outer legs being in line, hip-switch has occurred.

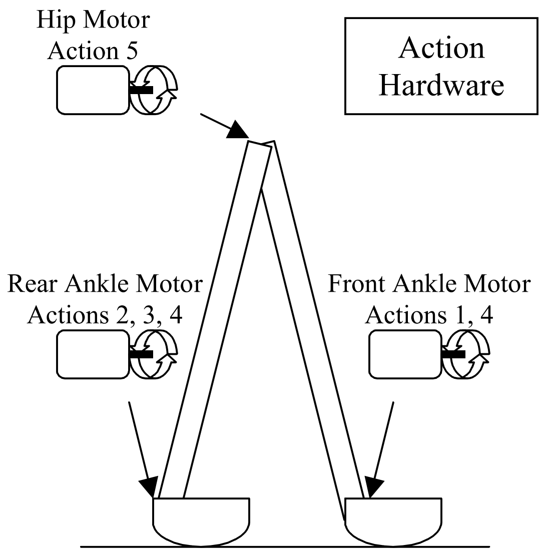
Actions:

Action 1: Ankle Preparation

The robot performs front ankle preparation, rotating foot to the proper position for heel-strike (Sense A). Position is controlled with a "P-loop" using information about current ankle angle (Feedback α).

Action 2: Ankle Push

Robot performs rear ankle push-off. The voltage applied to the ankle motor is constant, as specified in the code.



Action 3: Ankle Retraction

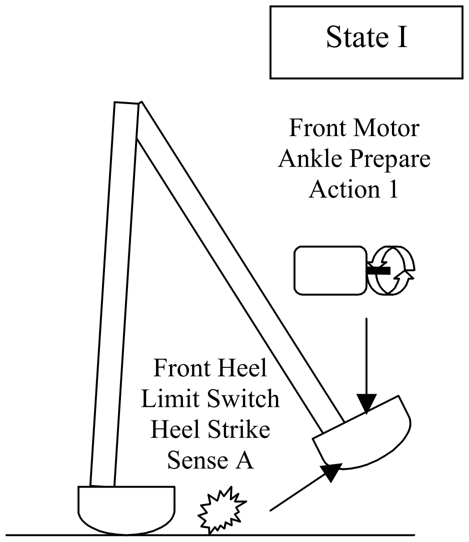
Robot performs rear ankle retraction, rotating foot to prescribed position to provide clearance with the ground at hip switch (Sense C). Position is controlled with a "P-loop" using information about current ankle angle (Feedback α).

Action 4: Ankle Lock

Robot performs ankle lock, holding the foot at a prescribed position. Position is controlled with a "P-loop" using information about current ankle angle (Feedback α).

Action 5: Hip Swing

Robot performs hip swing. The voltage applied to the hip motor is a function of hip angle (Feedback β).



States

Initial State

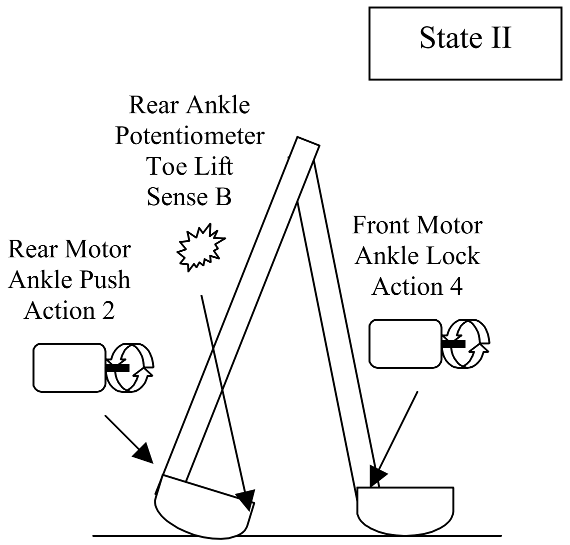
State I

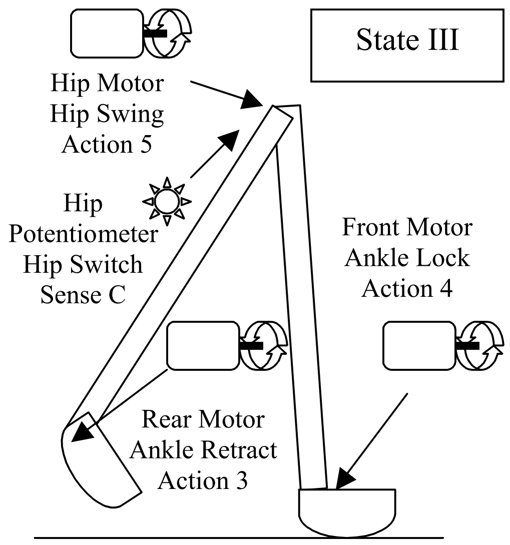
The robot has been set up manually by the user with its rear foot on the ground and front heel in the air. The robot prepares front foot for heel strike (Action 1). When the operator releases the robot, the robot passively falls as a double pendulum with a rolling contact pivot. When heel-strike (Sense A) occurs, the robot enters State II.

Walking Cycle

State II

The robot performs rear ankle push (Action 2) and front ankle lock (Action 4). When rear toe lift (Sense B) occurs, the robot enters State III.


State II
Rear Ankle Potentiometer Toe Lift Sense B
Rear Motor Ankle Push Action 2
Front Motor Ankle Lock Action 4

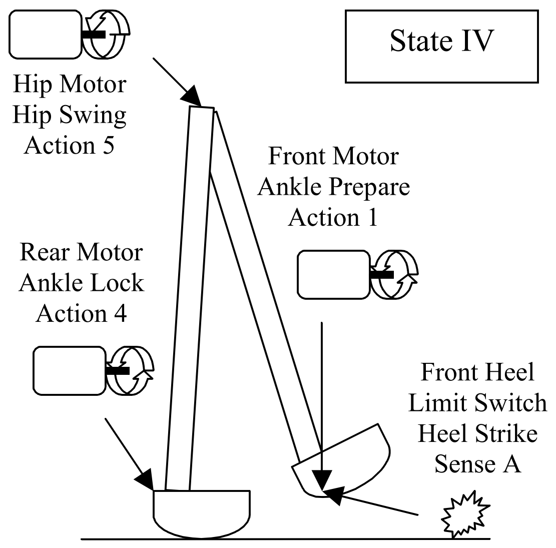

State III
Hip Motor Hip Swing Action 5
Hip Potentiometer Hip Switch Sense C
Front Motor Ankle Lock Action 4
Rear Motor Ankle Retract Action 3

State III

The robot performs rear ankle retraction (Action 3), front ankle lock (Action 4), and hip swing (Action 5). When hip switch (Sense C) occurs, robot enters State IV.

State IV

The robot performs front ankle preparation (Action 1), rear ankle lock (Action 4), and hip swing (Action 5). When front heel-strike (Sense A) occurs, robot returns to State II.


State IV
Hip Motor Hip Swing Action 5
Front Motor Ankle Prepare Action 1
Rear Motor Ankle Lock Action 4
Front Heel Limit Switch Heel Strike Sense A

Code

The software is written in C with MPLAB IDE, compiled into a single .hex file by the MPLAB C18 Compiler, and downloaded to the mini robot controller via serial cable using IFI Loader. Three files of the default code included with the robot controller were heavily modified to achieve desired performance.

user_routines_fast.c

The algorithm detailed above is implemented using a single variable to keep track of the robot state, one function for each sense to return the logical state of the sense, one function for each robot action to cause the hardware to perform the action, a single function to return continuous feedback from a specified potentiometer, and various processor functions to assist the rest of the functions with their tasks. The Process_Data_from_Local_IO() function contains the state machine itself. A "switch-case" statement tests the "State" variable - for each state, certain action function are called and a conditional statement calls a sense function. If the sense function in the conditional statement returns true, the state variable changes to its next value. When the code reaches the end of the switch statement, the loop repeats.

user_routines.c

The User_Initialization() routine is used to set up ports on the robot controller for use in the code.

user_routines.h

This is the header file where custom functions are prototyped and user-defined macros are declared. Once the robot began executing the four states successfully, this file became the most important in debugging because it is where the values of all experimentally determined program constants could be changed.

Adjustments

After all hardware was wired and code was loaded on the robot, a tremendous amount of both mechanical and electrical "tweaking" was necessary before a stable walking cycle was achieved.

Electrical

After finalizing the code structure, the most important changes were the adjustment of constants that controlled the magnitude of motor actuation. In general, the scientific method was followed – the robot would be set in its initial state and allowed to follow its instructions. Instead of walking, the robot would crash. The cause of the problem would be determined and a single constant would be adjusted based on experience with passive walkers, insight from simulations, and physical intuition in an attempt to remedy the problem. This basic cycle of testing and tweaking was executed until a walking cycle was realized. Due to machine asymmetry, separate constants were often needed for inner and outer legs and for front and rear positions. When this was found to be the case, slight modifications to the action functions allowed for the specification of additional constants. Hardware changes include relocation of all control components from the top of the robot to the legs, and replacement of Mike Sherback's H-bridge units with Innovation First Victor 884 Speed Controllers.

Mechanical

Many mechanical adjustments were made to the robot in the debugging process, but the volume of material that would be required to properly document each was prohibitive. Some examples of this work include complete replacement of the hip potentiometer mounting bracket and the coupling between the hip potentiometer and hip motor, straightening of the hip motor bracket, notching the hip motor bracket to prevent collision with battery terminals, replacing the mis-cut carbon fiber tube on the chain tensioner, drilling out unnecessarily and detrimentally tapped holes, addition of ankle drive gear set screw holes, and addition of a bumper to prevent overextension of the hip.

Results

Results are evident in video footage of the robot walking. The robot successfully executes a walking cycle, recovers from minor conditional imperfections, fails when faced with significant adverse effects, and consumes energy less efficiently than the average car air conditioning system.

Discussions

The initial goal of this project was accomplished in full – the existing robot has been modified such that it is capable of executing a stable walking cycle. The next goal of the project, robust walking, has been partially satisfied – the robot has shown the ability to recover from slight perturbations, but frequently encounters challenges posed by the environment and internal mechanical difficulties that it cannot overcome. The final, long term goal of this project - efficient, long distance walking – is only beginning to be tackled. The current walking cycle of robot wastes a tremendous amount of energy locking the feet in specified positions and moving the hip to an angle it should ultimately swing to on its own. There is a tremendous amount of proposed modification that can be made to make the robot both more efficient and robust.

First, the robot must be improved mechanically. The current feet should be replaced with versions that are slightly shorter to compensate for insufficient motor torque, but stiffer to prevent deflection, a suspected cause of failure. The new feet must be aligned properly so that they provide symmetric support on the ground. New limit switches should be mounted more robustly so that they are triggered whenever any part of the heel is in contact with the ground, and not triggered when the heel is off the ground, as there have been problems with the limit switches sticking, breaking, and not triggering. The leg chain tensioner slots should be lengthened to provide for further tensioning as the chains are rather loose and unexpectedly skip links. The motor mounting bracket should be replaced so that it does not require a spacing shim. A new tensioning system that relies on an idler sprocket should be created, as the current setup introduces high levels of friction. Although the new "homemade" potentiometer coupling is sufficient, a standard coupling should be purchased and installed or the potentiometer should be remounted and turned by its own sprocket. While the robot works as it is and these changes are not all necessary, it is hoped that making them would save time performing urgent maintenance, which has slowed down the debugging process considerably in the past. Also, many aspects of the current code were written to reflect imperfections in the robot's mechanical functionality, and relaxing this constraint would offer more freedom for success.

The next task would be to make the robot more symmetric by providing for easy repositioning of control system components to adjust weight distribution. A battery box should be made and an adjustable mounting system should be devised for easy battery removal, installation, and repositioning. The robot controller, interface board, and speed controllers should be remounted on thin aluminum plates for durability, as they will need to be repositioned frequently until desired symmetry is achieved.

Once the robot is symmetric and mechanically reliable, the speed controllers should be replaced with more efficient and precisely controlled H-bridges, and a torque control system should be devised. A strategy designed but never implemented was to sense the voltage drop across a small resistor in series with the motor; from this the current could be inferred. Using this information, the current through, and thus the torque produced by the motor could be controlled. Another suggested strategy was to implement a current mirror circuit. In either case, a function must be written to generate PWM signals for any arbitrary desired frequency and duty cycle in order to send signals to the motor control hardware.

Data from the accelerometer already installed on the robot should be used in the code to determine the overall angle of the robot with respect to the vertical. Information from a gyro could be used to help distinguish between the different components of the overall acceleration vector. A method for using the encoders already mounted on the motors should be devised – perhaps by using a counter chip and a latch or even by replacing the robot controller with a faster model more suitable for interpreting the quadrature signal. Encoders would provide much more precise positioning information and, more importantly, angular velocity data.

Once more relevant control over actuators and better feedback data are available, the code should be changed to reflect the fact that robot performance can more closely match simulation. The current asymmetric state machine architecture is necessary for the robot in its present state, but it should be replaced by a more elegant structure equivalent to that in simulation code. This would allow for most of the performance optimization to occur in simulation rather than by physical testing and tweaking. If developed, such an agreement between simulation and the actual machine would be a true milestone in robotic walking not only because it would become relatively easy to optimize this system for efficiency and long distance walking, but also because it would aid in the generation of new and improved physical robot designs.

Conclusion

The Marathon Walking Robot project is in the process of succeeding in its goals. The robot has already achieved a walking cycle and has the ability to recover from mild disturbances. It is known that the robot must become even more stable and much more efficient for it to cover distances on the order of kilometers, but efforts to the present have provided a firm foundation and great insight on what must be done. It is within realistic hope that further time and effort invested in this machine will yield significant results, not only accomplishing the goals of the specific project, but also contributing to the realization of the culture's vision of humanoids. Perhaps even more importantly than providing a basis for future walking locomotion, the project will yield a model for the use of simulation in the design and improvement of even more advanced systems.

Acknowledgements

I have found it funny how "university research" works. The professor organizes a graduate student project. Graduate students lead a team of undergraduates. I, an undergraduate, have high school students to thank for helping me get started. I would like to thanks Lucas Waye of FIRST Team #639 at Ithaca High School for helping me get comfortable with robot controller programming, and Kyle Witte of FIRST Team #116 at Herndon High School for providing recommendations about electronics. Thank you to Gregg Stiesberg for assistance with virtually every aspect of the project. Thanks to Professor Andy Ruina for providing many of the pieces for the robot control algorithm, and to Mike Sherback for his technical expertise.

Citations

[1] Gosse, L. 1998. *Minimally Power Walker Based on Passive Models*. Master of Engineering Project Report.
[2] Yevmenenko, Y. 2000. *Powered Straight Legged Walker with Circular Feet*. Master of Engineering Thesis.
[3] Yeshua, O. 2003. *Power & Control of a 2D Passive-Dynamic Walking Robot*. MAE 490 Final Paper.
[4] Seidel, W. and Strasberg, M. 2005. *Mechanical Hip Actuation of a 2-D Passive Dynamics Based Walker*. T&AM 492 Final Paper.
[5] Harry, Z. 2005. *2-Dimensional Bipedal Passive-Dynamics Based Walker*. M&AE 491 Senior Design Project Paper.
[6] Innovation FIRST. *2004 EDU Robot Controller Reference Guide*. Available at: http://www.ifirobotics.com/docs/legacy/edu-rc-2004_ref_guide_2004-jan-14a.pdf
[7] Innovation FIRST. *2004 Programming Reference Guide*. Available at: http://www.ifirobotics.com/docs/legacy/2004-programming-reference-guide-12-apr-2004.pdf
[8] Innovation FIRST. *EDU Default Software Reference Guide*. Available at: http://www.ifirobotics.com/docs/legacy/edu-default-software-guide_10-15-2003.pdf
[9] Innovation FIRST. *12V Victor 884 Users Manual*. Available at: http://www.ifirobotics.com/docs/ifi-v884-users-manual-1-26-05.pdf
[10] Vishay Spectrol. *Vishay Spectrol Model 140*. Available at: http://www.vishay.com/docs/57039/140142.pdf