Walking Robotics

Contents

Introduction

This is the third year I have built a robot.

The first year I built components of a system that were supposed to run with a Commodore VIC20 computer. Unfortunately, the peices didn't work together and the robot had bad mechanisms.

Last year I built an all new mechanical body with better parts. It was also to be controlled by a Commodore. I managed to get the VIC to make the legs move in response to typing, and to display on the screen if a sensor was activated. However, the VIC was too cumbersome to program and I was unable to get a program that coordinated leg movement with sensor information.

This year I built a computer and periferal hardware. I also wrote software for this computer that links the motors to the sensor readings.

Purpose

The purpose of this project is to design and construct a four-legged walking robot and all the computer hardware and software needed to make it walk. I have seen toys that walk unintelligently on four legs. I want to make a robot that is programmed to walk in a fairly versatile fashion. I hope to build on this in the future to make a more intelligent and more capable robot.

Procedure

Soon after the end of last year's science fair season I became acquainted with the 68HC11 microcontroller. I built and tested a computer based on it. Soon after the basic model was working I expanded the 768 byte memory by 32k. Then I interfaced the HC11 to the serial/parallel board from my Commodore interfaces.

The serial/parallel is a slave under control of the computer. It receives bits from the computer serially and outputs them to the motor board. It also gets the input from all of the switch sensors and spools them in to the computer. The motor board uses optoisolators to switch power transistor H-bridges which control the motors. A H-bridge is an arrangement of transistors that allows a simple +/- power supply to reverse the polarity on a motor.

Over the duration of the project I gathered development tools for the

68HC11. I got all of them from various FTP sites on the internet. These tools include an assembler, emulator, downloader and a program to give me “peek” and “poke” access to the computer.

There were many bugs in the computer. At some point everything had a bug. I’m sure that I spent more time debugging than constructing.

Between regional and state science fair, I built new computer. The one that I had previously was based on the MC68HC11A1P. That is a 48 pin DIP case, and it has 4 A/D inputs. The new computer was based on the MC68HC11A1FN. The new chip is a 52 pin PLCC case, the extra four pins are A/D inputs. If I had not made the switch I would have had to build a large expansion board to attach to the old version. I decided that it would be easier and smaller to build the new computer.

I am using the body that I built last year. I changed the leg design from what I was originally using and the back two legs are new. Last year I was using nitinol wire to move the leg forwards and backwards. It was too slow so I replaced the wire with motors.

Software

When I had the majority of the electronic hardware done I began writing software. One of the programs I got off the internet was designed to run a mass produced board based on the same microprocessor. I removed all of the things specific to that board and put in my own programming. The original program was 800 bytes long. After pruning it was 400 bytes. After I added an I/O routine and logic for independant control of four legs, the program was around 1100 bytes.

The program is not especially complex, but it is large. It is based on the “Subsumption Architecture.” The Subsumption Architecture is really nothing more than a large set of conditional branches, written in assembly. An average branch could be described as “IF (leg is forward and on ground) THEN move leg backwards.”

All of the IF-THEN statements have access to the motors, but some come later in the program and can override the earlier decisions. First in the program and of the lowest priority are the rules such as the one above. These rules move a leg through a basic walking cycle and do not deal with actions or non-actions that could cause harm to the robot.

Second come the rules that coordinate the legs. All of the legs have the same program at the lower level, and this makes them work together. This level primarily prevents standard actions on some legs while the normal actions go ahead on other legs.

Next up on the priority ladder is the rules that deal with motors being unable to move in some direction. Sources of these situations come from either bump sensors or limit sensors. These rules primarily have the power to stop motors, but can suggest actions for getting around obstacles.

Above all, there are rules to prevent conflicts between lower levels. If the computer instructs the motor board to go both directions at once, the transistors will short out and damage will most likely result. Just before the end of the control cycle, this level of rules will turn off any motors that have conflicting signals from lower levels.

Conclusion

I have a program that can go through the motions necessary for walking based on input from sensors. It can read sensor info for each leg and make the decisions for all of them within a millisecond.

The whole project cost about $600. I can only estimate that I spent several hundred hours on it.

The complexity of it when compared to other robots is relatively low. However, I will admit that more complexity would be desirable. More degrees of freedom on the legs would allow more agility. A full range of sensors such as sonar and infrared would allow the robot to plan ahead and avoid obstacles.

The software is easily extensible because I can add more rules at any time in a new layer. The Subsumtion Architecture is also easily debuggable. Any level can be changed separate of the others. I did run into many bugs but they only effected two rules and so I just cleaned up those two rules.

Overall, I am satisfied with my product. Of course, there are always features to add.

Applications

A next generation version of this robot could go to the Moon, Mars, and volcanoes where the robot Dante might have tried to go. For $500 each you could get 100 of them for less than 10% the price of one Dante. Also, if half of them get stuck or fall off a cliff, the lost investment is relatively low, because the rest keep going.

Dante 1 failed because it was too complicated and needed additional equipment tacked on at the last moment. Dante 2 was well built, fairly reliable and it cost over $1 million. Dante 2 failed because the environment was chaotic and Dante did not have much hope of surviving what it was put through. That is why many smaller, simpler robots would be better. “Don’t put all of your eggs in one basket.”

The Dante robots are intended to go where it is too dangerous or too costly for humans to go. I think that decendants of my robot could also do this but cheaper. Dante can go on science mission and into space, but cheap robots open up new possibilities. In the $500-$2000 range they enter the same category as personal computers, and could become just as common. I think it could be transition similar to that between PDP-11 and Apple II.

If a gripper were added, they could perform simple chores in more rugged environments where wheeled robots can’t go.

Future

In my present design I am counting on several things to work and I have no way of sensing if they actually are. I would like to sense the actual movement of the motors, and maybe the temperature of the power transistors. I might want to sense the digital level motor control, but if chips are going to fail, it could just as easily be the input.

I would like to build an android. This will require gyros for good balance. The computer could remain the same but I will probably build a multiprocessing computer with one or two HC11’s and maybe a 68000.

Compared to other 8 bit microporcessors, the HC11 is very capable. It is capable of true preemptive multitasking. There are higher languages such as C and Forth. However, it is still an 8 bit machine moving around 8 bits only 2 million times a second and none of this comes easily. A 16/32 bit micro like the 68000 would allow much more freedom.

I would like to explore advanced programming techniques and learning algorithms. All of the learning algorithms I know of need faster computing and more than 64k of addressable memory.

Acknowledgments

I would like to give special thanks to ElRoy Miller who has helped me with technical work for the last three years and has helped me to learn how to actually get things working. He loaned me the use of his home shop tools to work with and an Oscilloscope to help me debug my circuits. Without his help I might not have learned enough to do this project.

Fred Martin for creating great tools for 68HC11.

I would also like to thank my parents who supplied the bulk of the funding for this project in the form of my allowance.