# Robot Architecture

by
Jerry Burton

## 1.0 Introduction

What is meant by Robot Architecture ? The definition of 'Architecture' that applies in this context is "Construction or structure generally; any ordered arrangement of the parts of a system" (Funk & Wagnalls). Now this applies to the shape of the robot, what sensors or types of motors are used, etc. In this paper I wish to limit the discussion to the way the internal structures are organized. In particuliar how many processors are used and if more than one, how are are they connected.

The structure of processor architecture does not affect what sensors are used or even what method of locomotion is used. These are lower level decisions that are beyond the scope of this paper.

## 2.0 Self Contained

A self contained robot as I define it is a robot with a single CPU that must process information from ALL sensors as well as control ALL motors. This is the architecture that was used on ALL commercially available personal robots with the exception of the Newton by Synpet.

This included the RB5X which used a National Semiconductor INS8073 an 8 bit microprocessor. This chip had a built-in Tiny Basic for ease of programming.

The TOPO series used a Motorola 6502 processor and was actually remotely controlled from a main computer via either an RF or IR link.

The HUBOT was a Z80 based CPM machine.

The HERO I and jr series by Heath used a Motorola 6808 CPU, whereas the Hero 2000 used an Intel 8088.

The Artec machine was Motorola 68000 based and was designed for using multiple processors using an internal bus.

The primary drawback with a single processor architecture is that the robot is basically under-powered, since the single processor had to do everything it was limited in what could be added. The limited amount of I/O lines meant you could not add additional sensors or other experimental subsystems very easily or not at all.

Todays microprocessors on the other hand have a lot more capability built-in and offer a lot more processing power. The 68HC11 has 40 I/O lines, with counter-timers built-in, plus

serial I/O ports. The Signetics 8x552 has built-in pulse width modulators, serial and parallel I/O, and timers - all built-in. Either of these processors provide enough power to build a self-contained robot that could perform quite well.

3.0 Central + Subsystems

This architecture uses a Master processor that communicates with various subsystems, via a bus structure, that provide lower level processes. The Newton robot uses this architecture. It is essentially a PC on wheels. It uses 2 subsystems, one is the HPC which controls the main motors and interfaces all sensors. The other subsystem is the TI speech subsystem for recognition and speech and special effects sound generation.

It is possible to add additional subsystems by merely plugging them into the PC bus and adding the appropriate driver software. For example, an arm subsystem or a video subsystem could be added to provide additional capability.

The master processor talks to the subsystem via special driver software which is normally designed as either TSRs or linked into the main program at compile and link time.

The main advantage of this type of architecture is that the subsytems can do the low level processing and the master processor can give high level commands that can be carried out by the subsystem independantly. For example, the main processor can tell the HPC subsystem to move forward 10 feet, and command the TI speech subsystem to talk while the move is in progress. This allows a certain amount of parallel operation to occur which is a definite improvemnet over the self-contained approach.

The major disadvantage of this approach is that each subsystem must have the support circuitry to interface to the PC bus. Another disadvantage is that the only way to communicate with a subsystem is from the master processor. There is no means of allowing subsystems to communicate with one another directly.

4.0 Network

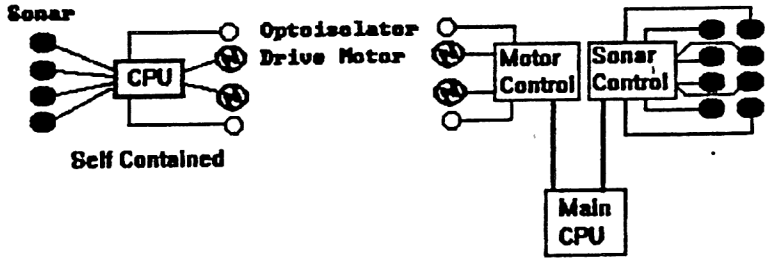
In a network architecture each subsystem is independant of the others and are linked to the others via a serial link. There are several alternatives as to how this link can be implemented.

Figure 1 shows a simple daisy chain approach. Each subsystem has a unique address and messages are passed from one to other in turn. When a message for a particuliar subsystem is received it is taken off the link and messages that don't belong to a particuliar subsystem are merely passed on to the next subsystem. In this way, any number of messages could be sent, with new ones being added on to the end and old ones taken off by the appropriate subsystem. Each message should have some sort of

block check method so that if an error is encountered a retransmission can be done.

The advantage of this architecture is that any subsystem can talk directly to ANY OTHER subsystem. This means the sonar subsystem can send a message directly to the locomotion subsystem to inform it that a collision is eminent. A disadvantage of this architecture is that if ANY subsystem fails the entire network fails.

Figure 2 shows a star network, in this configuration there is one link to each subsystem. This network is in fact the same as a bus oriented architecture, but does not require the bus electronics. It has the drawback that the central subsystem, i.e. the main processor, is the only one that can talk to the other subsystems. Subsystems can not talk to each other directly. The advantage is that the network continues to function even if one of the subsystems malfunctions. The only way the network can fail is if the main systems fails.



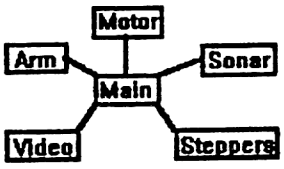




Figure 1 - Star Network

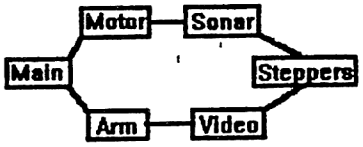



Figure 2 - Daisy Network

## 5.0 The $I^2C$ Serial I/O Bus

This is a fairly common network protocol which defines how different devices on the network will communicate. There are a number of processors that provide hardware to support this protocol. The $I^2C$ bus uses two wires to transfer information between devices connected on the bus. The main features of the bus are :

- Bidirectional data transfer between masters and slaves
- Multimaster bus (no central master)

- Arbitration between simultaneously transmitting masters without corruption of serial data on the bus.
- Serial clock synchronization can be used as a handshake mechanism to suspend and resume serial transfer.

Each device on the bus can assume any one of four modes :

- Master Transmitter
- Master Receiver
- Slave Receiver
- Slave Transmitter

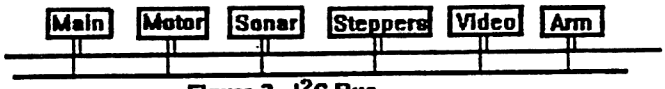



Figure 3 - I2C Bus