Appendix B. Cost of the platform

I've totaled up the approximate cost of constructing this robotics platform in this section. Note this does not include sales tax or shipping where applicable, nor does it cover the $117.00 I spent on the PICSTART PIC programmer.

Cougar	$34.99	$34.99
Battery pack	2@$17.99	$35.98
Charger	$9.99	$9.99
MiniBoard 2.0	$100.00	$100.00
Ultrasonic Board	$79.00	$79.00
Servo	$11.88	$11.88
H-Bridge	$12.00	$12.00
PICProto Board	$9.99	$9.99
IR LEDs	5@$1.69	$8.45
RED LEDs	10/$1.00	$0.80
TI TPIC2701	$5.39	$5.39
PIC16C56/JW	$17.30	$17.30
Sharp IS1U60	3@$4.80	$14.40
Hall Effect Switch	2/$0.55	$1.10
Rare Earth magnets	8/$0.75	$6.00
	Total	$347.27

Appendix A. Suppliers

Polaroid Corporation

Ultrasonics Component Group
119 Windsor St. - 28
Cambridge, MA 02139
(617) 577-4681

Supplies the Ultrasonic transducer/module pair.

All Electronics Corp

P.O. Box 567
Van Nuys, CA 91408
(800) 826-5432 Orders
(818) 904-0524 Information and Customer service

All supplies the Hall effect switches, part #HESW-2

Pure Unobtanium

13109 Old Creedmoor Rd
Raleigh, NC 27613-7421
FAX/Voice (919) 676-4525

Source for motor driver chips, IS1U60 IR detectors, and IR LEDs.

Maxim Integrated Products Inc

120 San Gabriel Dr
Sunnyvale, CA 94086
(408) 737-7600

Source for the MAX620 chip and +5v only RS232 drivers. Ask for the small quantities desk when you call them.

microEngineering Labs

Box 7532
Colorado Springs, CO 80933
(719) 520-5323

Supplies the PICProto 18 and Dual PICProto blank printed circuit boards.

Hobby Services

1610 Interstate Drive
Champaign, IL 61821
(217) 398-0007

Manufacturer of the CS-51 R/C servo used in this project.

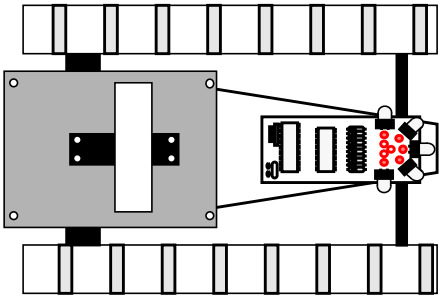
microphone feeds the input of the second microphone. The phase difference gives a general indication of where the tone originated.

Other options for the audio subsystem are an FSK modem chip, a TouchTone[tm] decoder chip, or simply some LM567 tone decoders for single bit output.

6.0 COUGAR PLATFORM EXPERIMENTS

The complete package is shown in Figure 9, “Cougar Systems Layout,” on page 12. We can use this package to do several experiments which are described in this section. Two basic areas of research are being done with this platform, autonomous goal navigation, that is navigating to a goal while avoiding obstacles, and predator/prey/herd experiments where the goal is dynamic rather than static.

FIGURE 9. Cougar Systems Layout



6.1 Find the Beacon

The first experiment is called “Find the Beacon”. This experiment has three simple behaviours; wandering around, identifying and closing on the beacon, and recognizing when the goal has been achieved.

6.2 Navigate a maze

This is the standard “micromouse” sort of application where the robot is responsible for getting from point A to the goal at point B. Given the relative lack of precision this platform has for moving, the challenge in maze and other navigation experiments is to use dynamic goal planning, re-assessing the plan for reaching the goal, based on current sensor input.

6.3 Track the Dino

This is a slightly different problem, instead of tracking a fixed beacon, the Cougar trys to follow the beacon attached to another robot. The primary difference between this code and the code for location and going to a beacon above is that the beacon is constantly relocated rather than assume it sits in place. Note that the Dino also recognizes when it is being followed because it can “see” the IR scanner of the Cougar. When it detects its pursuer it attempts to evade.

6.4 Action Selection Mechanism experiments

One area of research in the artificial life community is the development of action selection mechanism simulators using cellular automata. Using these platforms is much more enlightening because they do not suffer from digital quantization errors that a typical simulator will introduce.

```
                count := count + 1
                if (count > MAXCOUNT) then
                        return MAXCOUNT
        end
        return count
end
```

The count returned by the ping function can be calibrated into inches by multiplying it by a constant. I've found however that simply knowing the relative magnitude of count is sufficient to make navigation decisions on.

When combined with the infrared sensors, the ultrasonic sensor can give the robot an idea of what is around it and what things need to be either gone to or avoided depending on the application at hand. Typical use is to wait for the long range infrared sensors to detect an obstacle and to point the sonar in that direction and take a ping to see how far away the detection occurred. This is particularly useful when the obstacle is a poor IR reflector and is actually closer than the detection range of the IR scanner would indicate.

5.8 Visible light sensors

In addition to the infrared scanner subsystem, several photocells are connected to the Analog to digital converters on the Miniboard. This is easily the simplest sensor package on the Cougar. These sensors look to the sides and forward and are used to track the blinking "watering hole" beacon. They can be used in photovore type applications as well. The photo function is basically as follows:

```
function photo()
do
        if (adc_ch1 < PHOTO_THRESHOLD) then
                result := TO_LEFT
        else if (adc_ch2 < PHOTO_THRESHOLD) then
                result := STRAIGHT_AHEAD
        else if (adc_ch3 < PHOTO_THRESHOLD) then
                result := TO_RIGHT
        return result
end
```

The constant PHOTO_THRESHOLD is set empirically for the light environment at the time. I experimented with setting it dynamically, lowering it until the beacon was detected. However this generally takes too much code and the benefit is marginal. At such time as the Cougar has some form of secondary storage for storing programs this code was considered unneccessary. In our example we use ADC channels 1 through 3 because channel 0 is used to monitor the battery charge.

5.9 Battery level sensor

As was mentioned above, ADC channel 0 is used to measure the state of the battery. This A/D channel is connected directly to the motor power supply through a precision resistor divider (1K/2K @ 1%) This is a standard feature of the Miniboard design so no additional hardware was needed. The state of the battery is kept in a global variable `bat_level`. This variable is used in the speed control routines to figure out how much power will be delivered when the motors are turned on full.

5.10 Audio sensors - simple wireless I/O

The final sensor package on the Cougar is a simple microphone connected to an audio amplifier. The microphone is used to detect whistles that can be used to send signals to the Cougar. Uses include debugging, sending a whistle to the Cougar that tells it to stop or execute some diagnostic, and tracking, using a tone on a target for the Cougar to track when it cannot "see" a target with an IR detector.

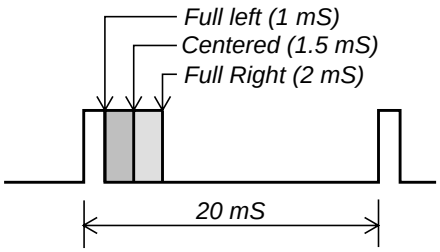
Ideally the cougar should have 'two' ears that help it identify both tone and the direction of the tone. This might be accomplished using a phase locked loop which is set to identify one tone from two inputs. The output of the "left" microphone feeds the source signal of the phase locked loop and the output of the "right"

5.7 Ultrasonic Sensor Software

There are two components to the ultrasonic sensor software, servo control and distance measurement.

Radio control model servos are extremely easy to interface to modern microcontrollers. Servos use a pulse signal that is varied in ‘width’ and repeated at a rate of 50 times per second. When the pulse is “narrow,” (about one millesecond wide) the servo is at one extreme of its travel and when the pulse is “wide,” (about two milleseconds wide) the servo is at the other extreme. The servo manufacturer will tell you the width of the pulse that registers as ‘neutral’ on the servo. In the case of the Hobbico servos this value was 1520 uS. This relationship is shown below in figure 8.

FIGURE 8. Servo Pulse Width and Servo Position



To control the servo requires an interrupt service routine that is run every 20 mS and that service routine sets an output bit on the output compare port. It calculates when the output compare bit should toggle to 0 and writes that into the output compare register. The timer section will automatically toggle the output when the appropriate time is reached.

The interface to control the servos takes a number between 0 and 100 representing a percentage of travel. The value of 50 is the servo “centered” with the Sonar transducer pointed straight ahead. In the actual code you will see that these values are tunable. The requirement for the tuning is due to the nature of the servos. While one millesecond to two milleseconds is the nominal range one can send to the servo, in practice the servos range is not quite so broad. This variation is caused by variations in servo design, both mechanically and electronically. Thus the numbers of interest for each servo are, at what width is the servo at one end of its travel and at what width is the servo at the other end of its travel.

Another area of servo design that we happen to ignore in this case is servo slew rate. This is the rate at which the servo changes to its new position. We *do* wait for the servo to settle into position before using the transducer, but we do not attempt to approach that position at anything less than full speed. To accomodate slewing, our interrupt service routine would have to take into account the “new” desired position versus the current position and make a decsion as to whether or not the next position to send to the servo should be the new position or some fraction of the distance. It would also have to provide a notification mechanism so that the main code could tell when a requested position had been reached. The advantages of slewing are that it can reduce current consumption of the servo since you aren’t driving it as hard, and it smoothes out the motion which can provide a more stable sensor platform.

Unlike the infrared system which is active all the time, the sonar system is only activated when the distance to a wall or target is needed. Consequently it is used by the higher levels of the control program, those involved with achieving the desired goal. For this reason the ranging function, ping, is quite simple. It pulls the INIT line of the interface high, and then counts up until the echo line goes high. The accuracy of this technique is effected by the system being interrupted so the cougar doesn’t move while using sonar. The ranging function is shown in the following psuedo code

```
function ping
do
        init := TRUE
        count := 0
        while (echo != TRUE) do
```

5.5 Ultrasonic hardware - The Polaroid module

The Polaroid Module that came from Circuit Cellar Kits had solder holes for 8 lines. In my experience I was unable to get the multiple echo mode of the module to work so I concentrated on using single echo mode. In this mode only four lines to the module are significant, Vcc, Ground, INIT, and ECHO. This also allowed me to use the otherwise unused SPI port on the 68HC11 because its pins can be used as general purpose I/O when not being used in the SPI function.

5.6 Ultrasonic hardware - Servo Platform

Because the ultrasonic module is expensive, $40 to $80 depending on the source, it seemed impractical to have several modules looking in several directions as I did with the infrared sensors. This limitation of looking in one direction only could be overcome in two ways; I could attach the module to the Cougar and turn the Cougar to face the direction I wished to look or I could attach the sensor to a servo platform.

I chose to mount the sensor to a servo platform for several reasons. The first reason was that given the feedback on the tracks, I am unable to precisely point the Cougar in any direction, nor can I adjust my heading by any small number of degrees. Secondly, using the motors to point the transducer consumes quite a bit of battery reserve and I thought it would be nice to be able to run for a while on a single charge. And finally by fixing the position of the transducer I would be unable to look in a direction that the Cougar wasn't pointed. If I wished to use the sonar for target acquisition or for tracking my progress around obstacles, I needed to be able to turn it.

Creating a servo platform turned out to be quite simple. The basic design is shown below in figure 7. A mounting platform was cut from plexiglass to the same dimension as the Cougar's truck bed. Dowels are glued into the four corners of the bed to provide the supports for the four platform corners. Additionally the dowels are cut to a length such that the platform is level to the ground when it is attached to the Cougar. In this platform a rectangular hole, the size of the servo, is cut and the servo is attached via its mounting bolts to the platform. A circular attachment that came with the servo was glued to the bottom of the Radio Shack project case and the whole thing attached to the servo. Wires from the transducer come out the side of the project box and travel down through a hole drilled into the mounting plate attaching to the Miniboard with is attached to the main Cougar chassis..

FIGURE 7. Ultrasonic Servo Platform

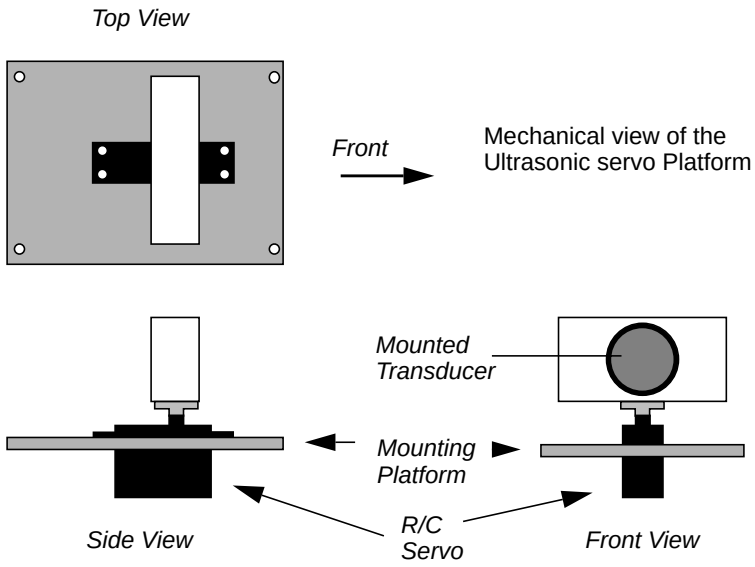
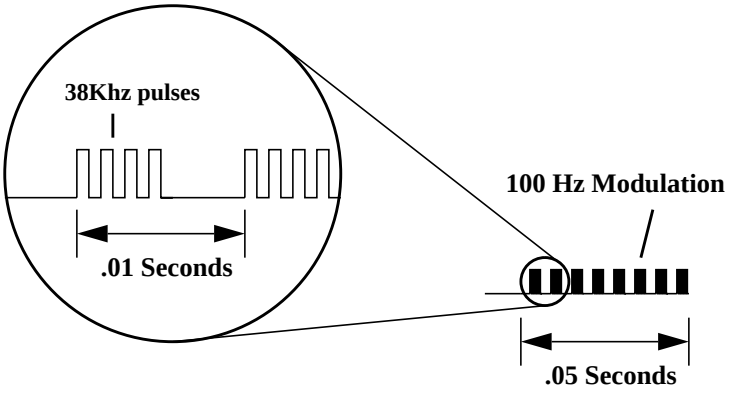
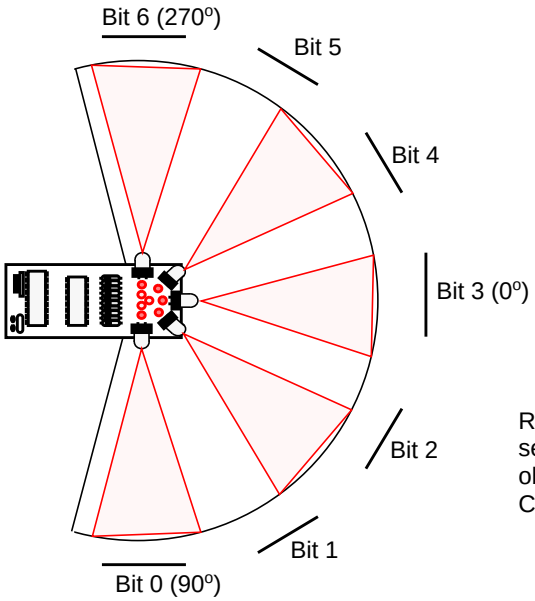


FIGURE 5. Generated Waveforms



The scanner board takes advantage of an additional property of the IR detectors, this is their sensitivity to out of band signals. The IS1U60s that the Cougar uses are most sensitive to IR light modulated at 38Khz. Inside the detector there is a bandpass filter whose 3db points are at + and - 4 Khz. This means that the detector is approximately half as sensitive to 34Khz and 42Khz modulated IR than it is to 38Khz IR.By adding an additional instruction into each half of the IR sending loop of the program we can decrease the frequency by 4Khz! I use this by creating two “on” loops, one designed to produce 38Khz and one designed to produce 34Khz. The IR detectors can detect an obstacle at 48” using the 38Khz loop but at only 18 - 20” using the 34Khz loop. This then gives me three effective “zones” in which I can detect obstacles; Zone A. This zone is < 24” from the Cougar, operation is only allowed at slow speed. Zone B. This zone is obstacles > 24” but < 48”, operation at medium speed is ok. Zone C. Obstacles to 35’ (420”) clearly these things are “far” away. And finally Zone D. Is anything outside of 35’.

FIGURE 6. IR LED Scanning



Relationship between bits set in the collision mask and obstacles in the path of the Cougar.

5.4 Long Range Sensors - Ultrasonics.

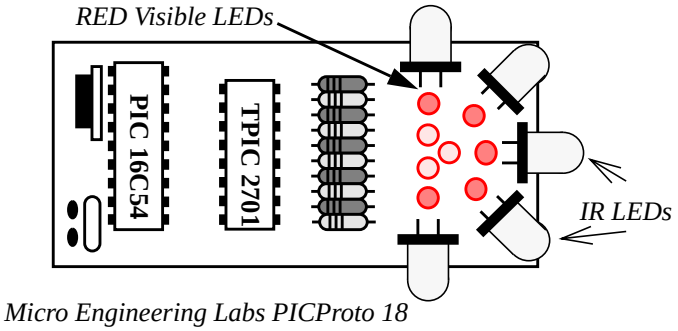
Ultrasonics, in particular a Polaroid T101 Sonar Module, are used for long range sensors. This module offers object detection from 1.5’ to 35’. Accuracy is good to about 1/2”. The sonar module is mounted on a Hobbico Car/Aircraft servo that has 180 degrees of movement. When centered, the servo points the transducer straight ahead. This combination allows the transducer to be pointed fully to the left and fully to the right with any angle in between to an accuracy of slightly better than two degrees.

```
function short_range(led_number)
do
        for iteration := 1 to 192
                LED[led_number] := ON
                delay(13 uS)
                LED[led_number] := OFF
                delay(13 uS)
                adjust_detectors(TRUE, 1)
        end
        return
end

while (TRUE) do
        current_LED := 0
        for cycles := 1 to 5 do
                if (detected == FALSE)
                        long_range(current_LED)
                else
                        short_range(current_LED)
                (* Generate silence for .005 seconds *)
                for iteration := 0 to 192 do
                        LED[current_LED] := OFF
                        adjust_detectors(FALSE, -1)
                end
                (* Check our detectors to see if we have a collision *)
                if (abs(detector_1) > THRESH) then
                        detected := detected + 1
                if (abs(detector_2) > THRESH) then
                        detected := detected + 2
                if (abs(detector_3) > THRESH) then
                        detected := detected + 4
        end     (* .1 seconds for each LED *)
        if (detected != 0) then do
                collision := collision | bit_maps[current_LED, detected]
                detected := 0
        end
        (* return the next LED in sequence, handles bounce back at ends *)
        current_LED := next_LED(current_LED)
        if (((current_LED == 1) OR (current_LED == 5)) AND (collision != 0)) then do
                send_update(collision)
                collision := 0
        end
end
```

All detected reflections are shown in the three inner LEDs on the IR scanner board. When collision information is sent to the Miniboard, it is in the form of an 8 bit byte with the most significant bit being the “short range” (see below) bit. Bit 0 represents the 25.7 degrees to the furthest left of the Cougar. If the cougar was heading at 0 degrees this would be 62.5 degrees through 90 degrees. Bit one would be 38.6 through 64.3 degrees and so on through the half circle until you got to bit 6 which represents 270 through 295.7 degrees. This relationship is shown somewhat more clearly in figure 6.

FIGURE 4. The IR Scanner board



On the IR board, the RED visible LEDs immediately behind the IR LEDs are driven in parallel to the IR LEDs in a pull down configuration. This was done to allow visual inspection of the board operation. The cluster of three RED leds in a triangle behind this set reflect the state of the IR detectors being returned to the PIC. As these are updated in real time, it is possible to “see” that the Cougar has detected an obstacle by observing the associated detector LED being illuminated. The PIC is clocked by a 2.4576 Mhz crystal to give us accurate timing of the LED outputs.

5.3 The IR sensor software

The firmware in the IR sensor board is responsible for generating a 38Khz square wave to feed to the LEDs, monitoring the detection results from the Sharp IS1U60 IR detectors and then sending those results to the miniboard whenever an obstacle is detected. Given that the system is prone to detecting ambient IR energy the LEDs are further modulated using a 100Hz input to distinguish between random IR and generated IR.

The innermost loop of the program generates a 38Khz square wave on one of five I/O lines connected to IR LEDs. There are actually three copies of this loop, one that generates 38Khz, one that generates 34Khz, and one that is quiet. This allows us to generate the 100Hz AM modulation on the output. (See figure 5 on page 8) During both output and quiet stages the detectors are scanned for the “detect” state (which is alternately *true* when 38Khz is being produced and *false* when no signal is being produced) The detectors are repeatedly sampled to eliminate false triggering due to glitches. The detection values are then filtered through a threshold and when we are sure a reflection is valid we combine the information about which detectors “saw” the reflection and which LED was being turned on to calculate where the obstacle is located. This calculation returns a bit representing roughly 25.7 degrees of “space” in front of the Cougar. The program can be expressed in psuedo code as follows:

```
function adjust_detectors(desired_state, adjustment)
do
        for num := 1 to 3 do
                if (detector[num] == desired_state)
                        detector[num] := detector[num] + adjustment
        end
        return
end

function long_range(led_number)
do
        for iteration := 1 to 192
                LED[led_number] := ON
                delay(14.52 uS)
                LED[led_number] := OFF
                delay(14.52 uS)
                adjust_detectors(TRUE, 1)
        end
        return
end
```

of an obstacle by the IR sensors. When either of these events are detected the motors are turned off and the status reported to the main control program. When running, the interrupt routines maintain a count of edges seen from the wheel encoders and this information is stored in global variables `r_encoder` and `l_encoder`. The motor drive function monitors the values in these variables and adjusts the values in the `speed5` and `speed6` to keep the Cougar on track. These variables start out being initialized to 0xFF00 which, when shifted out to the motor bits gives a 50% duty cycle square wave with a frequency of 60Hz. The integration function is shown below.

```
while (distance != requested_distance) do
        if (emergency_stop) then
                return stop_status
        if (r_encoder > l_encoder) then
                adjust_speed(+1)
        if (l_encoder > r_encoder) then
                adjust_speed(-1)
end

function adjust_speed(amount)
do
        on (amount) do
                case +1 :
                        speed5 := speed5 << 1
                        if (speed5 == 0xE000) then do
                                speed5 := (speed5 >> 1) OR 0x8000
                                speed6 := (speed6 << 1)
                                if (speed6 == 0xF000) then
                                        stalled := TRUE
                        end
                        break
                case -1 :
                        speed6 := speed6 << 1
                        if (speed6 == 0xE000) then do
                                speed6 := (speed6 >> 1) OR 0x8000
                                speed5 := (speed5 << 1)
                                if (speed5 == 0xF000) then
                                        stalled := TRUE
                        end
                        break
        end
        return
end
```

5.0 SENSOR PLATFORMS

5.1 Short Range Sensors - IR LEDs

For short range 0” - 24” detection of obstacles the Cougar uses reflected, modulated IR light. This controller has two functions, it provides modulated IR illumination in five directions and IR receivers to detect IR reflected from obstacles in the path of the Cougar. The controller also does filtering on the detected reflections to screen out false positives. When an obstacle is detected, it signals the miniboard using a simple serial protocol on the input capture #0 pin.

5.2 The IR sensor hardware.

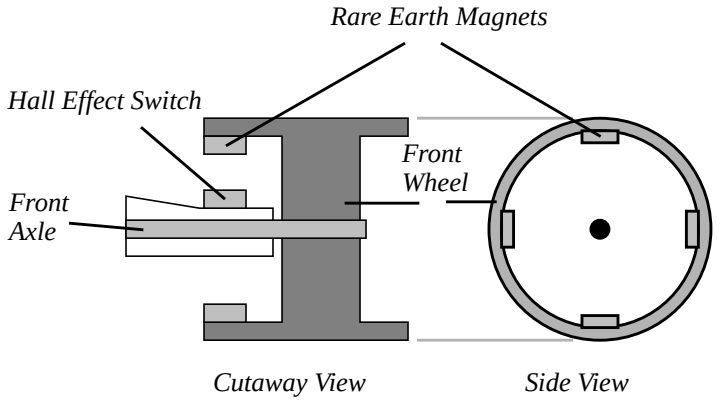
The source for the IR is a set of 5 IR LEDs being driven by a PIC 16C54 microcontroller. This controller is connected to a Texas Instruments driver chip which contains 7 MOSFETs. The MOSFET driver chip was required because the LEDs require a total of 60mA when illuminated. This exceeds the PICs maximum current sink capacity. The controller illuminates each sensor in turn. The board is shown below in figure 4. The circuit was prototyped on a Micro Engineering Labs PICProto18 board.

4.3 Speed Feedback

To effectively control the speed of the motors on the Cougar, feedback was required to ascertain the speed of the vehicle. If I were designing this system from the ground up, I would have included tachometers on both motors to use in controlling their speed. Since I was unable to retrofit tachometers to the motors, I chose to approach the problem by monitoring the turns of the wheels. Note that this solution is not ideal because the wheels provide fewer pulses per revolution than a tachometer would as the wheels come after the gear box. Finally, because the sensors are not on the drive wheels they can't detect the drive wheels slipping inside the treads. In practice, this latter issue has not been a problem.

Configuring the Cougar for feedback was accomplished using hall effect switches and some rare earth magnets (Radio Shack 64-1895) mounted on the front rims of the Cougar's wheels. (See figure 3 on page 4) The switches used are from All Electronics (part #HESW-2) and are bistable. This means they switch 'on' when the north pole of a magnet passes them and 'off' when a south pole passes it. This means that on each revolution of the wheel four edges are clocked. Because the wheel has a diameter of 1.6875", these sensors return one clock edge for every 1.33" of forward travel. This is calculated by taking the circumference and dividing it by 4. The magnets themselves are attached by hot gluing them into the rim.

FIGURE 3. Mounting Magnets and Hall Effect Switches



The outputs from the hall effect switches are wired to port A, the timer port, on the miniboard. The left switch to input capture #1, and the right switch to input capture #2. By wiring them this way it is possible to write an interrupt routine that can count pulses or using the timer values calculate the rotational speed of the wheel associated with that input.

4.4 Motor Driver Software

The motor driver software is encapsulated in an interrupt service routine that gets run 1000 times a second or every millesecond. The job of this software is to take the requested motor speed that is stored in memory, the current speed as indicated by the hall effect sensors, and the relative speed of the two tracks and integrate them into a signal to be sent out to the H bridge.

Driving the motors is done on the Miniboard by an interrupt routine that is part of the system clock interrupt. The original Miniboard libraries setup up the internal OC4 timer of the 68HC11 to interrupt the processor at a rate of 1Khz. At each millisecond the original library implements the PWM algorithim for the motors that had been turned on by calls to the `motor()` function.

I've used a similar techique for the Cougar except that changing the speed it controlled by writing to a global variable rather than making a function call. Using this technique saves space in the generated code.

Speed feedback is collected by the Input Capture 1, and 2 pins (input capture 0 is used in the IR sensor platform below.) The interrupt service routine is also responsible for detecting stalls on either motor or detection

4.0 MOTOR DRIVE

4.1 Overview

Motor control, and especially motor speed control is one of the most challenging aspects of this system. As I mentioned previously the Cougar's motors are quite powerful and capable of moving the Cougar in excess of 20MPH. To give you an idea of how fast this is, at this speed (30 feet per second) the Cougar can cross a good size laboratory in slightly more than one half second! To haul in the speed to a level that the other parts of the system could manage requires active control of the motors speed by the processor.

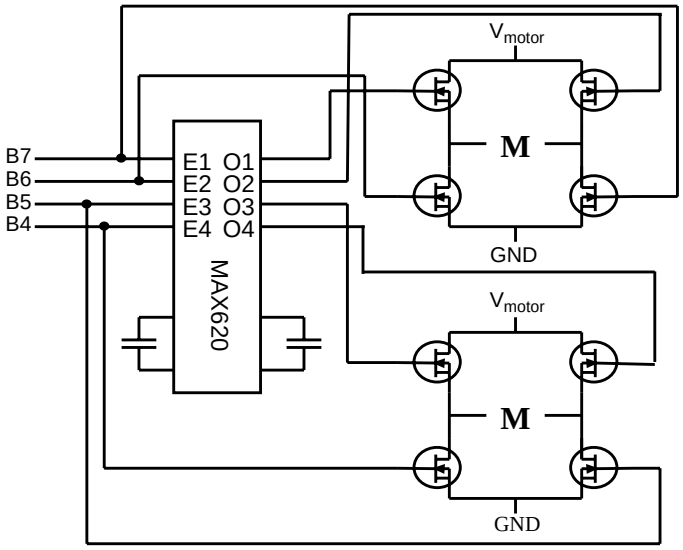
Thus there are two challenges to overcome, the first being how to drive the motors that are capable of using 5 amps when stalled, and secondly how to control the speed such that it would be possible to implement typical robotic behaviours without constantly smashing into things.

4.2 H-bridge Motor Driver Circuitry

The motors are controlled by a Dual MOSFET H-bridge of custom design. The core of the system is a Maxim MAX620 Quad MOSFET driver chip. It utilizes Maxim's charge pump technology to generate a drive voltage that is 10 volts higher than V*motor*. This insures that the MOSFETs that are used on the high side of the bridge are switched fully on. See the schematic in Figure 2, "Schematic of the Cougar's custom H-Bridge," on page 4. The input is driven by 4 bits of the general purpose 8 bit digital I/O port. These are configured as active high inputs with bits 7 and 5 being "drive forward" and bits 6 and 4 being defined to be "drive backward". While this scheme is easy to drive it presents the danger of turning both bits on at the same time. Such a condition will cause the MOSFETs to create a short to ground. This typically burns out the MOSFETs. The traditional way of preventing this situation is to drive the MOSFETs with some logic gates that drive one output on and drives the other output low. The second generation H-bridge design will include this enhancement.

The chip used in this bridge, the MAX620, can be replaced by a MAX621 and the external capacitors eliminated for even fewer parts. The MOSFETs used can be any logic level FETs, I used Phillips BUKV455s which have a current capacity of 38A (Rds of .011 ohms). Another enhancement for this design is to optically isolate the H-bridge from the controller to prevent EMF noise from the motors crossing into the digital supply of the controller.

FIGURE 2. Schematic of the Cougar's custom H-Bridge



platform is that the rims of the wheels overhang the axles. This allowed us to mount magnets on the wheels and hall effect switches on the axles to give us active feedback on the motion of the vehicle.

2.1 Modifications

Modifications to the mechanics of the cougar consisted of replacing the R/C electronics package with a custom H-bridge, installing the hall effect sensors and magnets on the front wheels, and removal of the “truck” plastic. The rear portion of the plastic was retained as a platform for mounting sensor packages.

To this stripped down chassis, sensors were added to detect obstacles and targets. Additionally a CPU was added to control the robot.

3.0 CENTRAL INTELLIGENCE

3.1 Hardware - Miniboard 2.0

The base level intelligence behind the motion and sensors of the Cougar is the Miniboard 2.0.This single board computer was designed by Fred Martin and Randy Sargent at the MIT media lab. This board uses the E2 version of Motorola’s very capable 68HC11 microcontroller. This version of the 68HC11 series has two general purpose 8 bit digital ports, one of which is dedicated to four full H-Bridge high current drivers implemented by two SGS Thompson L293D chips, the other is used for general purpose digital I/O. The chip also has an 8 bit timer interface section which can be used for a variety of purposes from timing events, to generating the PWM necessary for operating R/C servos. Alternatively, this port can also be used for general purpose I/O. Two additional ports complete the picture, 8 channels of single ended analog inputs and 8 bits which form the basis of two serial communication interfaces, one asynchronous with an RS-232C interface and the other synchronous. As with the timer channels these bits can be used as general purpose I/O. I use the 4 bits of the synchronous interface to talk to an ultrasonic transducer system.

The miniboard uses the on chip 2K byte EEPROM of the ‘E2 to store programs. These programs are compiled on a host computer, such as an IBM PC compatible, and are then loaded into the EEPROM using a downloading program developed by Fred. This non-volatile memory is augmented by 256 bytes of RAM. While small, this amount of memory is sufficient for quite a few useful routines. There are alternative versions of the chip available, in particular the MC68HC711E9, with more EPROM for particularly demanding applications.

The development environment consists of cross assemblers and C cross compilers. There are a couple of free C compilers available for this chip, one from Motorola and one a version of the GNU C compiler (gcc) which has been ported by CoActive Asthetics of San Francisco. Neither of these options are particularly useful for people who aren’t comfortable diving in and debugging problems with the compiler since stressing them tends to break things.There are also a variety of commercial compilers available from the complete Archimedes system to the budget conscious $99 version of Micro-C from Dunfield Development. ImageCraft plans to have a compiler on the market soon, a pre-release version is available for free from the ftp host *netcom.com*.

3.2 I/O Utilization

Nearly all of the I/O on the Miniboard is being utilized by the cougar’s motion and sensor packages. The 8 bit I/O port is divided into to two portions. Bits 4 through 7 of this port are dedicated to driving the custom H-bridge (See “MOTOR DRIVE” on page 3.). The timer port, port A, is multipurpose. Bits 1 and 2 are used to field interrupts from the hall effect sensors on the front wheels. Bit 0 is used as a software serial port that receives input from the IR sensor array. and bit 5 is used to control the servo in the ultrasonics system (See “Long Range Sensors - Ultrasonics.” on page 9.). A/D channel 0 is used to monitor the current battery state, A/D channels 1, 2, and 3 are connected to photocells to receive light input, A/D channel 4 is connected to the audio amplifier. As the motor drivers are not being used to drive motors, I’ve used one of them to switch power to the IR sensor platform. This gives me the opportunity to minimize power consumption when the platform is not in motion.

Hacking the *Wild Cougar*

Chuck McManis

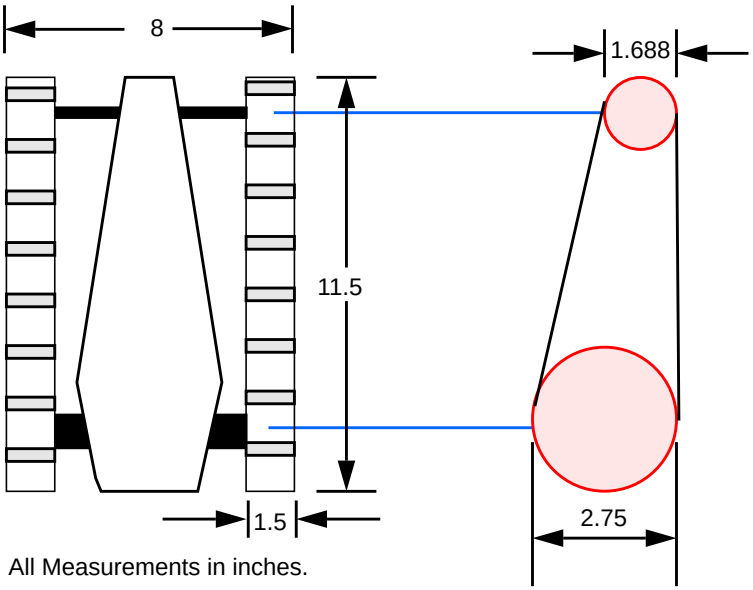
1.0 INTRODUCTION

Thispaper describes changes I've made to a Radio Shack Wild Cougar remote control vehicle. The attributes of this toy that makes it amenable to conversion are its use of track drive, which provides an easy mechanism for steering, rechargable NiCd battery packs which can be swapped easily for a fresh one, and all plastic construction which is easily machined with hand tools. I'll cover the components I've used, both off the shelf and those I've made, and then the integration of the system, and finally some experiments that can be run using the completed system.

2.0 THE PLATFORM

The platform is the Radio Shack *Wild Cougar* remote control toy, Radio Shack part #60-4097. This vehicle has external moldwork to make it appear to be a pickup truck. With the moulding removed, the basic vehicle shape is shown in Figure 1, "The Body Dimensions of the Cougar Vehicle.," on page 1

FIGURE 1. The Body Dimensions of the Cougar Vehicle.



This vehicle is particularly amenable to conversion because its dual track design allows the base to be steered by driving the tracks at different speeds and even different directions for extremely tight turning in place. The rear wheel is the drive wheel and has a diameter of 2.75 inches. The idler wheel is 1.6875 inches in diameter. This particular track design is not very efficient though; it requires the full track to be dragged across the ground during a turn. Modern tracked vehicles often use a more beveled design to minimize surface area while turning.

This deficiency is overcome by a pair of powerful Mabuchi motors. These motors are quite powerful and capable of driving the platform at over 20MPH! They are also something of a problem as they are capable of drawing 5 amps when stalled. This required a custom H-bridge as described later. Another feature of this