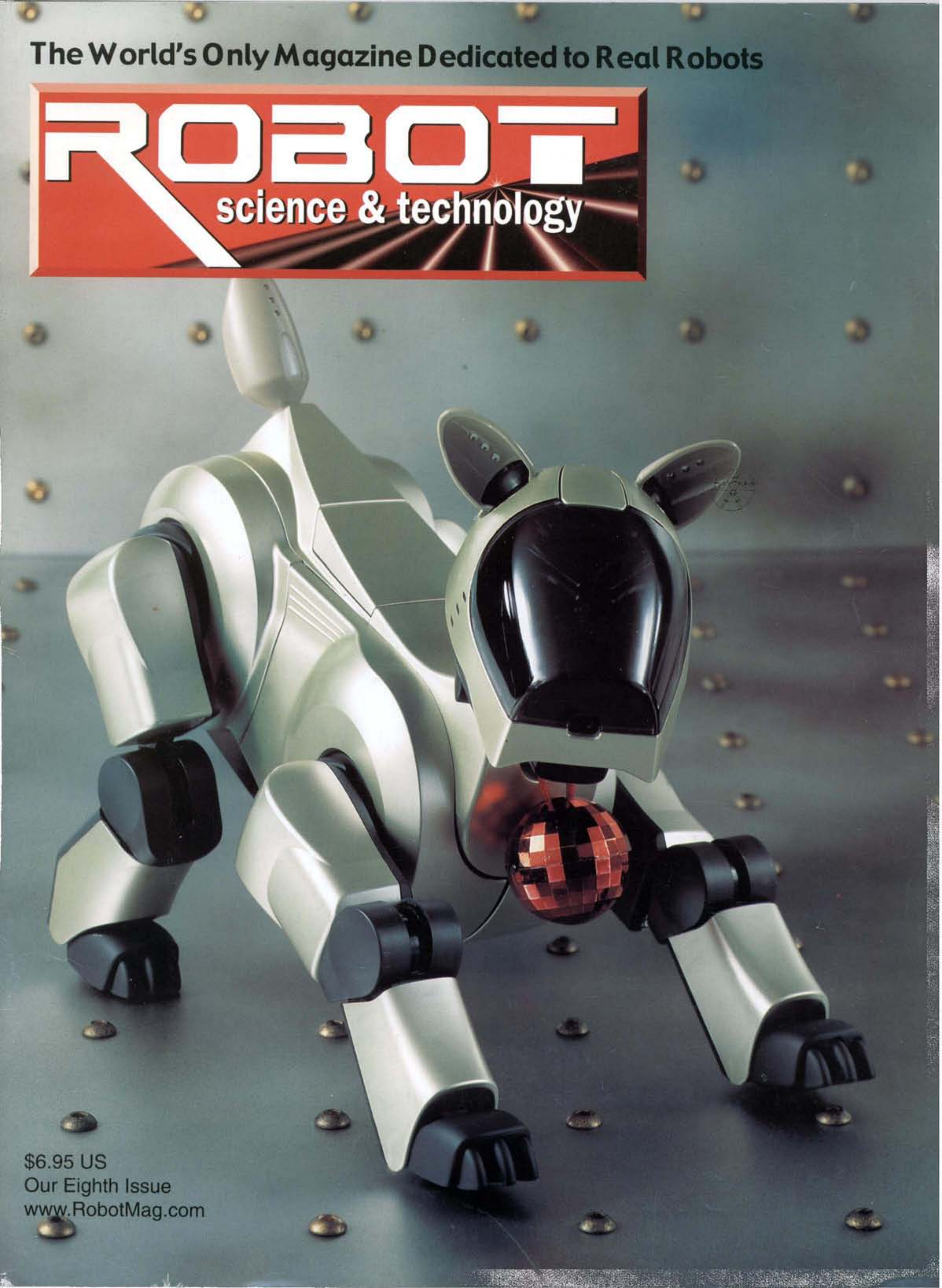
The World's Only Magazine Dedicated to Real Robots

ROBOT
science & technology



$6.95 US
Our Eighth Issue
www.RobotMag.com

ROBOT science & technology

PROGRAMMING

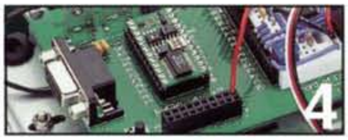
CONSTRUCTION

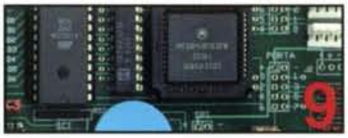
EXPERIMENTS

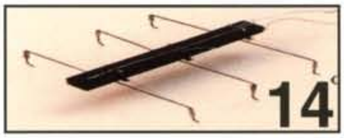
FABRICATION

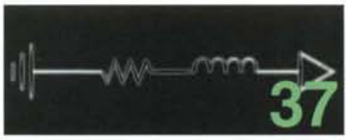
BRICKS & CLICKS

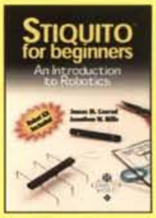
Departments





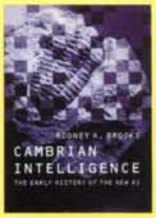






Stiquito for Beginners: An Introduction to Robotics
by James M. Conrad, Jonathan W. Mills.

Build Your Own Robot!
by Karl Lunt



Cambrian Intelligence: The Early History of the New AI
by Rodney Brooks

The Microcontroller Application Cookbook
by Matt Gilliland

Definitive Guide to Lego Mindstorms
by Dave Baum

Our Mission

To immerse readers in 21st century robotics technology with in-depth reports on real robots, and through hands-on adventures with home, classroom, and sport robotics.

Our Eighth Issue

Volume 2, Issue 2

Robot Science & Technology

3875 Taylor Road Suite B
Loomis California USA 95650

Publisher

Michael A. Greene
publisher@RobotMag.com

Editor

Mike Greene
editor@RobotMag.com

Creative Director

Richard Colby
artist@RobotMag.com

Contributing Editors

John Piccirillo, PhD

Tak Auyeung, PhD

Kristine Wills, MD

CUSTOMER CARE

Suzzette Clothier
service@RobotMag.com
toll-free 888.510.7728

ADVERTISING

Territories available, experience required. Email publisher. To advertise: email publisher or call 916.632.1000. Media kit available for Internet or print advertising.

MARKETING

Positions available, experience required. Email resume to publisher.

WE NEED HELP

Needed: Freelance artists, photographers, illustrators needed, editorial subject experts, online editors, and contributors. Link samples of your work as attachments to your email and send to publisher.

Exciting Times

It's an exciting time to be in robotics. Pioneers like Christopher Columbus, John Glenn, and Bill Gates placed themselves in unpredictable adventures, cutting paths for the next generation to settle and prosper. So do you, our readers.

Readers will blaze torches in garages, bending plastic for their creations; others will follow an adventure reading Rod Brooks' research papers in *Cambrian Intelligence*, and more will tie knots in *Stiquito's* shape-memory wires. Gearheads at keyboards will pound out code to multi-task microcontrollers, coaxing reactive behavior from embodied intelligences.

RS&T's contributing editors and freelance artists are crunching words for Issue 9, weaving together interviews with NASA researchers, FIRST graduates, SETI scientists, Poul Anderson and Marvin Minsky. How to submerge robots into Europa's ice? How can a home hobbyist get into that action? How can a teacher launch an experiment? We'll help.

As Robin Murphy wrote in her textbook *Introduction to AI Robotics* (review in Issue 9), we at *RS&T* are attempting to bridge the gap between hobby bots and research bots. Yes, we discovered that many of our readers held advanced degrees, and most of the PhD's described themselves as *hobbyists*.

Robotics, more than any other field, gives us an incredibly fun wild ride, re-learning and re-building. No wonder Lego bricks are taken seriously in the premier halls of science. *Extreme MindStorms* (review in Issue 9) co-author Michael Gasperi found Lego's vision brick to be a friendly introduction to AI's most perplexing questions (see page 51).

In fact, the word "friendly" keeps popping up in robot labs. ActivMedia's new AmigoBot (page 58) has the word built into its name. This is a trend: roboticists make friends when they gather together at events (see page 45). It's a good thing, too: To carve out the path for mankind's robotic future, we'll all need a little help from our friends.

Robot Science & Technology** (ISSN 1096-4754) is published bi-monthly at: 3875 Taylor Road, Suite B, Loomis, CA 95650; 916.660.0480; Fax: 916.660.0730; info@RobotMag.com. **Subscription** inquiries: U.S. and Canada 888.510.7728; all other countries 916.660.0480. Subscription rates: 6 issues $24 U.S., $29 Canada or Mexico (U.S. funds). Other countries $49. Periodicals postage paid in Loomis, CA and additional mailing offices. **Postmaster: Send change of address to: Robot Science & Technology, 3875 Taylor Road, Suite B, Loomis, CA 95650.

BASIC

Branching and Servo Control

2d article in a hands-on educational series

by Chuck Schoeffler and Al Williams

In this article we'll introduce BASIC Stamp branching and EEPROM access commands used to make the Boe-Bot follow a pre-determined path. A piezospeaker will provide a feedback mechanism to identify the location within your program.

Movement is one of the most distinctive features of robots, and it is also an ideal way to learn how to structure and write a simple PBASIC program. This experiment is all about BASIC programming as it pertains to Boe-Bot movement without sensor input. Structuring your program so the Boe-Bot moves as you intend requires an understanding of how to call subroutines, read movement patterns from an EEPROM, know how far to travel using a for..next loop, and how to get back to your starting point (physically, and within your source code). The parts we will be using are listed in Figure 1. The complete schematic for programs used in this article is shown in Figure 2.

Figure 1: Parts Required

1	*Complete Boe-Bot (robot built around Board of Education)*
1	*Piezospeaker*
1	*3300 uF capacitor*
2	*10 k ohm resistors (optional in circuit)*

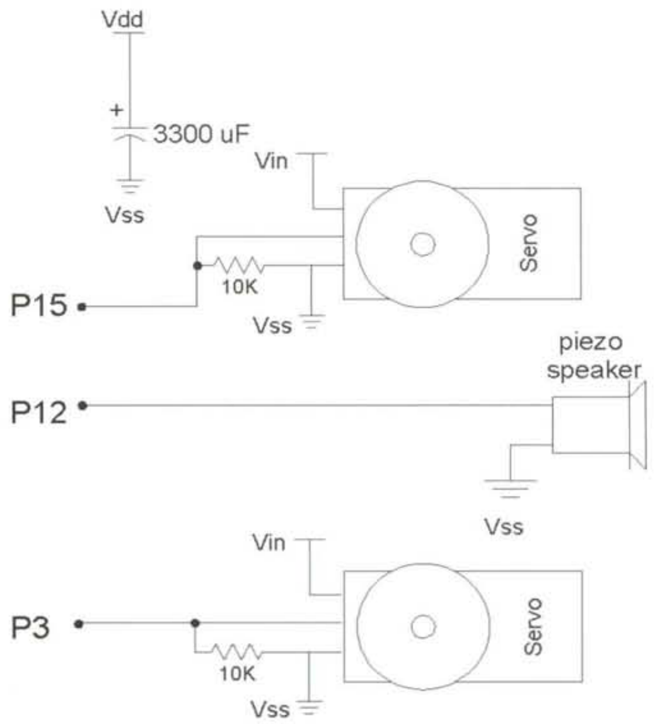
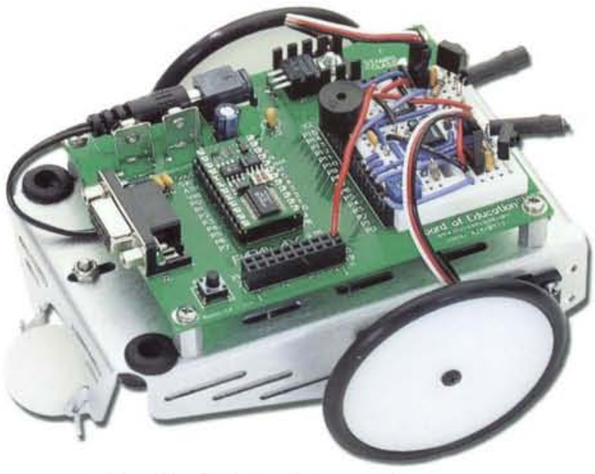


Figure 2: Basic Control and Feedback Schematic



Your Boe-Bot chassis may differ.
This design is described in Issue 7. RS&T

Figure 3: Servo Control Using Pulse Width Modification (not to scale)

Review of Servo Control

Servos are closed loop devices, and are constantly comparing their commanded position (from the BASIC Stamp's pulsout command) to their actual position (proportional to the resistance of a potentiometer mechanically linked to the shaft). If there is more than a small difference between the two, the servo's electronics will turn the motor to eliminate the error.

We modified the servo's potentiometer shaft until the gears stopped moving when the BASIC Stamp sent a 1500 µs pulse. A pulsout value of 750 is equal to 1500 µs (the command operates in units of two microseconds). A value larger than 750 will turn the servo clockwise, and a value less than 750 will turn it counter-clockwise. A value very close to 750, like 760, will cause the servo to turn very slowly. Figure 3, on the previous page, is a timing diagram of the pulse width modulation.

A for-next loop can be used to see how different pulse widths affect the servo's speed. Stand your Boe-Bot on it's front end or place an object underneath it to keep it from rolling away. Download Program Listing 1 to your BASIC Stamp. Figure 4 is a graph of pulse width compared to revolutions per minute using the Futaba FP-S148 servo.

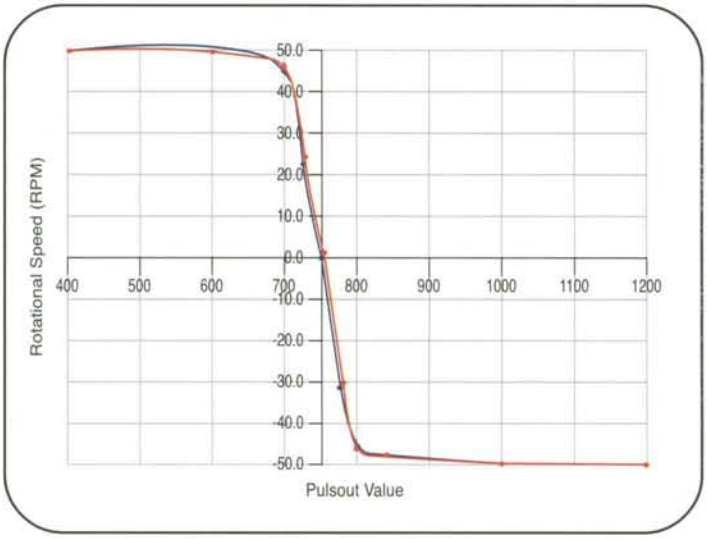


Figure 4: Pulse Width versus RPM Using the Futaba FP-S148

```
'Program Listing 1

left_servo              con     15
right_servo             con      3
x                       var     word

pause 2000
start:

for x = 650 to 850
 pulsout left_servo,x         'pulse width of 1500 us
 pulsout right_servo,1500-x   'pulse width of 1500 us
 pause 20                     'pause for 20 ms
next
```

Sound Feedback

The BASIC Stamp's freqout command can be used to add sound feedback to your Boe-Bot. Like all PBASIC commands, it has a particular syntax that must be followed to make it work. To hear the speaker download the following code to your BASIC Stamp:

```
freqout 12,750,2000
'750 ms 2000 Hz tone on P12
```

For a more "robotic" sound try this:

```
'Program Listing 2

Hz    var    word

for Hz = 1 to 4000 step 1000
 freqout 12,70,Hz,4000-Hz
 'generate two 70 ms tones on P12
next
```

This routine begins by declaring Hz as a word variable, a number between 0 and 65,536. The loop executes a total of four times ((4000-1)/1000), generating two frequencies at once on P12. The first frequency is increasing from 1 to 4000 Hz while the second frequency is decreasing from 4000 to 1 Hz. Sounds like this could be added throughout your program.

Goto Statement

Normally, PBASIC programs execute instructions line by line. The goto command causes the BASIC Stamp to jump to a named place somewhere else in the program. It can be either forward or backward in the program. The syntax is quite simple.

```
goto forward
'jump to the forward routine
```

Approximating Distance of Travel

It's easy to approximate distance of travel by calculating the circumference of a wheel and estimating the pulse width timing loops executed by your PBASIC code.

circumference =
π x wheel diameter

circumference =
3.14159 x 6.67 cm = 21 cm

Knowing the rotational speed of different pulse widths wi allow you to determine a specific travel distance. For e: ample, a *pulsout* command of 850 causes the servo to turn about 50 revolutions per minute (RPM) or 0.83 revolutior per second. Therefore the speed of the robot will be abou 21 cm/revolution x 0.083 revolutions/s = 17.5 cm/s.

To travel 100 cm, the Boe-Bot would have to travel for:
100 cm/17.5 cm/s = about 5.7 seconds

Since each servo pulse sequential and each pulse takes abo 1.5 ms and there is a 20 ms pause in the loop, each loop wi take about 23 ms (1.5 + 1.5 + 20), or 0.023 seconds to e: ecute. A total of 247 loops is required to travel 100 cm. Yc may have to adjust the for..next loop for your particul: servo.

```
5.7 sec/0.023 sec/loop = 247 loops
.
.
forward:
for x=1 to 247
 pulsout left_servo,650
 pulsout right_servo,850
 pause 20
next
```

Gosub is a Close Relative of Goto

The gosub (Goto Subroutine) statement also causes the pr gram execution to jump somewhere else, but the line aft the gosub is remembered so that the program can automa cally go back and continue where it left off.

This lets us easily reuse sections of the program. The fc lowing example illustrates the gosub command.

```
gosub right
pause 1000
gosub right
return

right:
for x=1 to 18
   pulsout left_servo,650
   pulsout right_servo,650

pause 20
```

This example shows the right routine being executed twi with a one-second pause.

The gosub commands may also be nested up to four dee so that each return takes the program back to the instru tion after the most recent gosub.

Using the Data Statement and EEPROM to Store Movements

The BASIC Stamp has a 2K EEPROM that is used for program storage (which builds from address 2047 toward 0) and data storage (stores in the opposite direction–from address 0 toward 2047). If the data collides with your program the source code won't execute properly. Each location is a byte. This isn't enough memory to build a complex environmental data logger, but it's certainly enough space to store bytes of information you'd like to use in a program.

The BASIC Stamp's EEPROM is different from RAM variable storage in several aspects:

- EEPROM takes more time to store a value, sometimes up to several milliseconds.

- EEPROM can accept a finite number of write cycles, around 10 million, (RAM has unlimited read/write capabilities).

- Primary function of the EEPROM is to store programs; data is stored in leftover space.

Three commands are used to access the EEPROM: data, read, and write. The data stored in EEPROM builds from the upper left-hand corner (position 0,0) and fills downward in a left to right fashion, by row. The source code builds from lower right-hand corner (position 16,128) and builds upward by row, right to left. If the two collide the BASIC Stamp won't execute the program properly. This is shown in Figure 5. The EEPROM memory map is accessed within the BASIC Stamp Windows editor at: *www.stampsinclass.com* under Run/Memory Map.

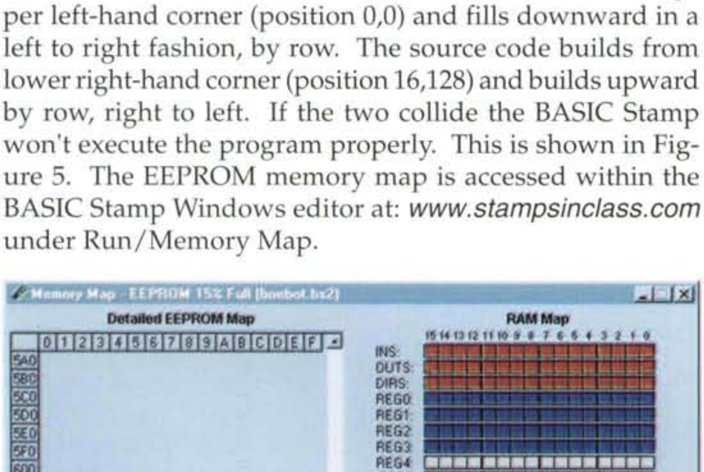


Figure 5: EEPROM Memory Map

The syntax for the read command is shown below. The write command uses the same syntax.

```
write 0,100
'write 100 into EEPROM byte 0
read 0,x
'read EEPROM byte 0 and store value in x
debug dec ? x
'display value on PC screen
```

Combine the Concepts

Program Listing 3 brings all of these concepts together: sound, movement, and speed. In order to make effective use of the speed, you will need to identify the exact center position of your servo. Chances are although we started with 750 (1500 ms) the servo may have wandered to a slightly different value. Program Listing 1 may be run to identify these values. At higher speeds (speed constant around 40) this is not as visible, but at slower speeds the Boe-Bot will slowly move sideways if the center position is not exactly 750. Change values, movement patterns, and place the sound routines in different sections of the program. This program is also available from the downloads section in: *http://www.stampsinclass.com.*

```
'Program Listing 3

'Boe-Bot Program for Roaming, Light, and Sound
'Define Variables and Constants
'----------------------------------
x             var     word     'loop counter for
                               pulsout
position      var     word     'EEPROM address counter
direction     var     word     'value stored in EEPROM
Hz            var     word     'frequency variable
right_servo   con     3        'right servo on P3
left_servo    con     15       'left servo on P15
speed         con     40       'added or subtracted
value
'----------------------------------
'Programmed Movement Patterns
data "FRFRFRBBTFE"             'store movements
'Main Program
position=0                     'start at EEPROM cell 0
move:                          'main loop
read position,direction        'read direction command
  if direction="E" then quit
                               'Decide which action to
                               take
  if direction="F" then forward'by matching command
                               letter
  if direction="R" then right
  if direction="L" then left
  if direction="B" then backward
  if direction="T" then turn_around
pause 250                      'pause 250 ms
position=position+1            'increment to next
                               cell
goto move                      'repeat until E is
                               seen
'----------------------------------
'Sound Routines
'----------------------------------
forward_sound:
for Hz = 1 to 4000 step 1000
  freqout 12,70,Hz,4000-Hz
next
return

back_sound:
for Hz = 4000 to 6000 step 1000
  freqout 12,70,Hz,Hz-400
next
return

right_sound:
  freqout 8,800,2500
return

left_sound:
  freqout 8,800,4500
return

'-----------------------------------------
'Movements
'-----------------------------------------
forward:
gosub forward_sound
for x=1 to 60pulsout left_servo,750-speed
  pulsout right_servo,750+speed
pause 20
next
goto move

backward:
gosub back_sound
for x=1 to 60
  pulsout left_servo,750+speed
  pulsout right_servo,750-speed
pause 20
next
goto move

right:
high 0
gosub right_sound
for x=1 to 18
  pulsout left_servo,750-speed
  pulsout right_servo,750-speed
pause 20
next
low 0
goto move

left:
high 14
gosub left_sound
for x=1 to 18
  pulsout left_servo,750-speed
  pulsout right_servo,750-speed
pause 20
next
low 14
goto move

turn_around:
for x=1 to 30
  pulsout left_servo,850
  pulsout right_servo,850
pause 20
next
goto move

quit:
end
'-----------------------------------------
```

LIKE TO PROGRAM ROBOT BRAINS?
WANT TO MAKE YOUR ROBOT JUMP THROUGH HOOPS?
So do our readers!

Experienced robot programmers and microcontroller engineers are cordially invited to email Editor@RobotMag.com for our FREE writer's guide.

YOU'RE A GENIUS – WE CAN TYPE.

Multitasking

68HC11 Robot Brain

by Jim Salvino

After working with the 68HC11 microcontroller for a while, I thought it would be interesting and useful to be able to run more than one program at a time. So I decided to take up the challenge of writing a simple multitasking operating system for the 68HC11.

As implied above, multitasking is defined as the ability to run many tasks (programs) "simultaneously." But a CPU like the 68HC11 can fetch and execute only one instruction at a time. So how could you ever get more than one task to run at a time? Answer: implement a mechanism (called the "scheduler") where each task is allowed to use the CPU for a very short period of time. This will give the illusion of running multiple tasks simultaneously.

There are two varieties of multitasking – cooperative and preemptive. They differ in the complexity of their scheduler logic. Cooperative multitasking uses a simple scheduler. However, the task logic needs to be more complex. Each task decides for itself when to return control to the scheduler and is responsible for saving its "environment" before giving control back to the scheduler. In preemptive multitasking, a complex scheduler gives each task a small amount of time in which to execute (called its "time slice").

At the end of a task's time slice, the scheduler interrupts the task, saves the task's "environment," loads in the next task's "environment," and finally transfers control to the next task. Writing task code for a preemptive multitasker is straightforward because there is no need for the "bookkeeping" logic used in the cooperative model.

This article will present a preemptive multitasker for the 68HC811E2. You may be asking yourself at this point, why bother with this stuff anyway? Why can't I run my robot from a well-structured C or Basic program? To answer these questions I will use an example. Say you want to tether your robot to a desktop PC via a serial connection. Now it would be useful to have the robot continually scan its sensors at the same time as it waits for commands from the serial port. Using a traditional top down program, you could read the sensors and possibly miss some serial data or you could wait for the serial data and crash into an obstacle because you weren't polling the sensors. However, by using preemptive multitasking, you can do both at the same time.

An excellent book got me started thinking about this. It was: *Mobile Robots - Inspiration to Implementation*, by Joseph L. Jones and Anita M. Flynn, published by AK Peters.

The Hardware

The hardware of choice for this project is the Motorola 68HC811E2 microcontroller. For those not familiar with this device, it has 2K bytes of on-chip EEPROM and 256 bytes of on-chip RAM. There are a number of I/O ports and even eight channels of A/D.

If you want to sacrifice a couple of the I/O ports, you can even add external RAM. For this project we will fit the entire multitasker into the on-chip memory and still have memory to spare for the tasks. The microcontroller itself can be found on Kevin Ross's web page at: *www.nwlink.com/~kevinro*

He sells a complete kit of parts for a very reasonable price.

The Software and Tools

Hats off to Karl Lunt. He has written a BASIC variant for the 68HC11 called SBASIC and has made it available free of charge from his web site at: www.seanet.com/~karllunt.

SBASIC has all the structures that one would expect in any well written BASIC: do..loop, while..wend, if..then, select..case, etc., and many instructions that facilitate programming the 68HC11 such as peek, poke, waitwhile, and interrupt.

Another nice feature of SBASIC used in this project is the ability to mix 68HC11 assembler code and SBASIC code in the same program.

In addition to a copy of SBASIC you will need to get a copy of PCBUG11 (the code downloader/debug utility) and the Motorola 68HC11 assembler program. To acquire and use these tools, please refer to the premier and second issues of RS&T, where Karl Lunt presents an excellent two part article entitled "Basics of a Digital Brain."

In these articles, Karl lays an excellent foundation to enable the reader to begin programming the 68HC11. Another helpful source is Karl's book *"Build Your Own Robot!,"* available from AK Peters. (See review on page 47)

The Scheduler & the Real Time Interrupt (RTI)

Essentially we are going to make use of a hardware function built into the 68HC11 called the real time interrupt (RTI). This feature, once the appropriate flag and mask bits are set, will trigger a jump to the RTI service routine every 4.1 ms. This is where the scheduler code will be put. Every 4.1 ms the scheduler will save what is on the hardware stack.

And what is on the hardware stack upon entry to the RTI routine–the condition code register, accumulators B and A, index registers X and Y, and the program counter from the task that has been running for the past 4.1 ms (our time slice). In essence, everything needed to pick up the current task at the point from which it was rudely interrupted by the RTI.

After saving the current task in memory, the scheduler then reads a previously saved task's program counter and registers and pushes them on the hardware stack in place of the first task's registers. It's the old switcheroo. And that's it.

The last instruction that is executed in the scheduler is a return from interrupt.

This instruction will restore the program counter, index, re isters etc., from what is on the hardware stack - everythi needed to restart. Not the task that was running at the tin of the RTI, but the next task that we want to dispatch.

The code for multitsk.bas is shown in Listing 1. This pr gram is a skeleton, and it's only meant to illustrate the tec nique of preemptive multitasking. It is up to the reader replace the logic in the sample tasks with something mo useful.

The code is well documented so the reader will not be su jected to a line by line explanation, but a few importa points should be made.

Variables / Constants

The most important data structure is something called t "task save area" or TSA for short. It is an area in RAM o size equal to the total number of tasks (as defined by t constant total_tasks) times 18. The TSA is where the sche uler saves a task's registers.

Each task occupies an 18 byte slot within the TSA. The va able TSAA is used as the pointer into the TSA. The da table 'modeltsa' is used to initialize each TSA slot. The va able task_ptr either points to the task which the schedul is saving or to the next task which it is going to dispatch

Multitasking Initialization Logic

The purpose of this code is to build the TSA before the R routine can be activated. Each task must start with a li label. In the initialization logic, each line label is convert to a two byte address which equates to the starting addre of the task. The task start addresses are then put in a tab the TSAAT, such that the table index corresponds to the o der of the task in the program. For example the code:

```
tsaat(1) = addr(countup),
```

says that the address of the line label "countup:" will stored at index 1 in the table tsaat. Moreover, the address stored at index 1 because task "countup:" is the first task the program. This procedure is followed for each task to run under the multitasker. The for..next loop then copi the 18 byte model TSA slot for each task to the TSA a then seeds the bytes corresponding to the program count (PCL, PCH) with the addresses that we just tabularized. Tl takes care of the "first-time" condition when the schedul needs to dispatch a task which it never saved. After t TSA is built, all we do is set the task pointer to task 1 a turn on the RTI.

Task Construction

Constructing a task is easy. As stated earlier, it must start with a line label. Also, each task must be thought of as a self contained program running on its own. So to accomplish this, a do..loop surrounds the task code such that it is not possible to fall through to the next task. As shown in Listing 1, following the line label, variables can be initialized before we enter the infinite do..loop, and once inside the loop you are free to code anything you like. Also, because all variables are global, any task can read/write to any other task's variables.

Running the Code

You may copy Listing 1 on page 12, or download it from: www.ctrobots.org/jims/listing1.html

After downloading it into your 68HC11 using the PCBUG11 utility, issue the "term" command to put your PC into special terminal mode. Set the BotBoard into single-ship mode by removing the jumper from pin2 (MODB) of the 68HC11 to ground. This maked pin2 "high" and the chip will go into single-chip mode following a reset.Then restrap the BotBoard for single-chip mode and press the reset button. You should see a stream of "var_1 = ..." and var_2 = ..." being displayed on your PC. You will notice that the values for var_1 count up from 1 to 30000, and values for var_2 count down from 30000 to 1. What this all proves is that the three tasks are being dispatched one after the other by the scheduler with each task getting a time slice of about 4 ms.

Some Caveats

There are a few caveats that need to be mentioned. First, because there are only 256 bytes of RAM on a 68HC811E2 there is a natural limitation as to how many tasks the program can handle. If you try to create too many, the hardware stack (which grows down from location $00FF) will collide with the variables and the TSA (which grows up from location $0000) with nasty results.

The next caveat is having too many nested subroutines in any one task. Each TSA slot is 18 bytes. Long enough to hold all the 68HC11 registers and up to 4 levels of nested subroutines. If you try to nest more GOSUBs than this, the scheduler code will overlay the TSA slot of the next task to be dispatched with disastrous results

The last caveat has to do with the SBASIC data stack. SBASIC has a data stack that is separate from the hardware stack but nonetheless shares the 256 byte RAM with everything else. No effort was made to save this area from task to task so only one task may make use of the SBASIC data stack.

Conclusion

Well that's it for now. What we have accomplished on a very small scale here with our little 68HC11 is what happens on millions of Windows 95 PCs every day - multitasking. And we didn't need megabytes of RAM to implement it.

Give it a try for yourself. I really think this can add a new dimension to your robot programming.

So have fun ...
And get those robots running.

Jim Salvino has been an avid robotics, computer, and electronics hobbyist for over 30 years. He is a Senior Systems Programmer and an active member of the Connecticut Robotics Society. Jim lives in Scotia, New York with his wife Carol and his three robots.

RS&T

Like to program robot brains?
Want to make your robot jump through hoops?
So do our readers!

Experienced robot programmers and microcontroller engineers are cordially invited to email Editor@RobotMag.com for our FREE writer's guide.

YOU'RE A GENIUS – WE CAN TYPE.

Listing 1 - Multitsk.bas

```
include "regs11.lib"         ' *********************************
                             ' Task Related Variables / Constants
                             ' *********************************
declare var_1                ' work variable used in task 1
declare var_2                ' work variable used in task 2
                             ' *********************************
                             ' Multitasking Variables / Constants
                             ' *********************************
const total_tasks = 3        ' This is the number of started tasks
declare task_ptr             ' task pointer
declare tsaa                 ' task save area address
const tsaat_space = total_tasks + 1
                             ' task save area address table
declare tsaat(tsaat_space)       ' **** Task Save Area ****
                    ' Note: variables in SBASIC are 2 bytes long
const tsa_space = total_tasks * 9
declare tsa(tsa_space)
        ' **** Model Task Save Area in eeprom
        ' used to initialize the tsa ****
        ' Each slot in the task save area is 18
        ' bytes long and has the following format:
        ' byte 0 - # of items in this task's tsa stack
        ' byte 1 - (CCR); byte 2 - (ACCB); byte 3 - (ACCA)
        ' byte 4 - (IXH); byte 5 - (IXL); byte 6 - (IYH)
        ' byte 7 - (IYL); byte 8 - (PCH); byte 9 - (PCL)
        ' bytes 10 thru 17 - variable (whatever the task had on
        ' the stack at the time of the RTI service routine)

modeltsa:                   ' Model Task_Save_Area Slot
datab $09,$00,$00,$00,$00,$00,$00,$00,$00
datab $00,$00,$00,$00,$00,$00,$00,$00,$00

                             '***********************************
                             ' Real Time Interrupt Service Routine
                             ' Multitasking Scheduler Logic
                             '***********************************
interrupt $fff0              ' RTI service routine
                             ' Pick up the task save area address
                             ' pointer for the current task
tsaa = tsaat(task_ptr)
                             ' Save the machine state for the
                             ' current task
asm
        ldd _tsaa            ' Get tsaa pointer and ..
        xgdx                 ' .. put in x index reg
        sts _tsaa            ' Use tsaa as a work var ..
        ldd #$00ff           ' .. and load with h/w stack pointer
        subd _tsaa           ' Subtract current h/w stack pointer
                             ' from begin address of h/w stack ..
        stab 0,x             ' .. and save the # of items on h/w
                             ' stack in the 1st byte of the tsa
        xgdy      ' Put # of items on h/w stack in the y index
                             ' reg
pulloop equ *                ' Bump the tsaa pointer past the
         inx                 ' number of items
        pula                 ' Pull task regs from the h/w stack
                             ' and ..
        staa 0,x             ' .. save in the tsa stack
        dey                  ' Decrement y index reg loop counter
        bne pulloop          ' loop until finished (zero)
endasm

if task_ptr < total_tasks     ' Bump task_ptr to the next task or
                              ' reset to 1 if at last task

task_ptr = task_ptr + 1       ' Bump the task pointer to the next
                              ' task
else
        task_ptr = 1          ' Reset the task pointer to the
                              ' first task
endif

tsaa = tsaat(task_ptr)        ' Pick up the Task Save Area
                              ' Address pointer for the next task
                              ' to be dispatched


asm             ' Restore the machine state for the next task
                ' to be dispatched
        ldd _tsaa ' Get tsaa pointer and ..
        xgdx      ' .. put in x index reg
        ldab 0,x ' Get # of items on tsa stack
        abx       ' Bump tsaa pointer by # of items on the tsa
                  ' stack
        pshloop equ *
        ldaa 0,x              ' Get task regs from tsa stack and ..
        psha                  ' .. push on machine stack
        dex                   ' Decrement tsaa pointer by 1
        cpx _tsaa             ' All done?
        bne pshloop           ' No, loop
endasm

pokeb tflg2, %01000000     ' rearm the RTI interrupt

end
```

```
                          ' *****************************
                          '       Main Line Logic
                          ' *****************************

main:

                          ' Activate sci so we can display some
                          ' results
pokeb baud, $30           ' 9600 baud
pokeb sccr2, $0c          ' turn on rcvr and xmtr

                          ' *********************************
                          ' Multitasking Initialization Logic
                          ' *********************************

                          ' Load task save area address table
                          ' with the starting address
                          ' of each task
tsaat(1) = addr(countup)  ' Add one entry for each started task
tsaat(2) = addr(countdn)
tsaat(3) = addr(printem)

                          ' For each task ..
                          ' .. calculate its task save area
                          ' address pointer into the tsa ..
                          ' .. initialize its task save area
                          ' slot by copying in the model
                          ' tsa slot ..
                          ' .. seed the tsa stack bytes
                          ' reserved for PCL, PCH for
                          ' that task ..
                          ' .. then save the tsaa in the tsaa
                          ' table

for task_ptr = 1 to total_tasks
       tsaa = ((task_ptr - 1) * 18) + addr(tsa)
                          ' calculate the TSA address fr task x
 copy addr(modeltsa), tsaa, 18
                          ' copy in model tsa slot
 pokeb tsaa+9, tsaat(task_ptr)
                          ' poke PCL in tsa slot for task x
 pokeb tsaa+8, swapb(tsaat(task_ptr))
                          ' poke PCH in tsa slot for task x
 tsaat(task_ptr) = tsaa   ' now use tsaat(x) to save the TSA
                          ' address for task x
next

task_ptr = 1              ' start at task #1
pokeb tflg2, %01000000    ' arm RTI interrupt
pokeb tmsk2, %01000000    ' enable RTI interrupt
interrupts on             ' allow interrupts

                          ' ******************************
                          '        Start of Task #1
                          ' ******************************

countup:
var_1 = 0
do
       var_1 = var_1 + 1
       if var_1 = 30000
               var_1 = 1
       endif
loop

                          ' *****************************
                          '        Start of Task #2
                          ' *****************************

countdn:
var_2 = 30000
do
       var_2 = var_2 - 1
       if var_2 = 1
               var_2 = 30000
       endif
loop

                          ' *****************************
                          '        Start of Task #3
                          ' ******************************

printem:
do
       print "var_1 = "; var_1
       print "var_2 = "; var_2
loop
```

I am a third grade student at Stevens Forest Elementary School, located in Columbia, Maryland. I learned about your company from your website.

I know some elements important to all robots, such as: gears to help move parts, energy (electricity, magnetic, hydroelectric, wind sun and hot air), a microchip to tell it what to do, sensors to send data to the microchip, a skeleton to hold it up and together, joints so it can move in different directions, motors to make it move, and software to control the robot's computer.

I would appreciate it very much if you could send me more information that I can use to build a robot.

Thank you very much,

Melanie

Melanie

Dear Melanie,

Thanks for writing for information about building a robot. Our magazine, Robot Science & Technology tries to help everyone build the best robots they can, at home or in school.

Choosing the best robots for a bright and experienced third or fourth grader depends on how much experience and interest you have in each of three areas: soldering, assembling small parts, and programming.

Look at the Wall Hugging Mouse and Hyper Peppy sold at RobotStore.com. Be sure to get their catalog by calling 800-374-5764.

The most versatile programmable sets are Lego-based kits, which you probably have seen already.

If you can solder electronic components to a circuit board, you might want to try building the Cybug series of kits available from HVWTech.com. Several Cybugs can interact as prey/predator. No programming is required, as the Cybugs' behaviors are hardwired, modified by adding circuits. Please ask your parents to help you with soldering kits.

If you want to program the robots' microchips directly, the OWI company at owirobot.com sells Navius and the WAO-G, which don't require soldering but are good for learning binary concepts and programming.

Certainly tell your teachers to look into robot classes taught by the KISS Institute for Practical Robotics at kipr.org. Teachers should also read the book, "Robots for Kids," by Allison Druin and James Hendler. The next issue of our Robot Science & Technology magazine will have a review of this book.

Parents may want to read Richard Raucci's book, "Personal Robotics," which we reviewed in RS&T Issue 7.

Enjoy building your robots, Melanie. I would be interested in hearing which kits you choose and how you like them.

Michael A Greene

Mike Greene
Editor and Publisher
Robot Science & Technology

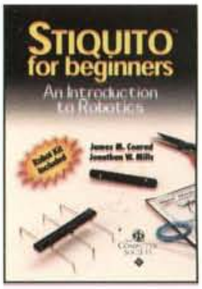
A Simple Circuit to Make Stiquito Walk on Its Own Effectively

by James M. Conrad and Serge Caron

Stiquito is a small, inexpensive hexapod (i.e., six-legged) robot. Universities, high schools, and hobbyists have used it since 1992. Stiquito is unique not only because it is so inexpensive but also because its applications are almost limitless. The propulsion in these robots is nitinol, an alloy actuator wire that expands and contracts, roughly emulating the operation of a muscle. The application of heat causes a crystalline structure change in the wire. Nitinol contracts when heated and returns to its original size and shape when cooled (See RS&T Issue 4).[Conrad and Brown 1999].

Stiquito was developed by Jonathan Mills of Indiana University as an inexpensive vehicle for his research. He soon found its applications extended to educational uses. It has been used to introduce students to the concepts of analog electronics, digital electronics, computer control, and robotics. It has also been used for advanced topics such as subsumption architectures, artificial intelligence, and advanced computer architecture.

The IEEE Computer Society Press has published two books, *Stiquito: Advanced Experiments with a Simple and Inexpensive Robot* [Conrad and Mills 1997] *and Stiquito for Beginners: An Introduction to Robotics* [Conrad and Mills 1999].



These books contain instructions for building the Stiquito robot, instructions for designing and building control circuits, and examples of student projects that use Stiquito. Most importantly the books contain all the supplies needed to build the robot. (See the books reviewed on page 17 of this issue.) Only a few common tools are needed to assemble a complete Stiquito.

One way to control the walking gait of a Stiquito robot is to build a circuit and program it using the parallel port of a PC [Conrad and Mills 1999a].

You can also build a circuit board that mounts on top of a already built and tested Stiquito robot and directs the rob to walk in a tripod gait. This circuit feeds current to th nitinol on a periodic basis, which you can adjust. This setu allows Stiquito to be an autonomous robot. The book *Stiqui for Beginners: An Introduction to Robotics* [Conrad and Mil 1999] includes a schematic for such a circuit.

The circuit generates enough current to alternately contra three nitinol actuators at a time. It uses a popular tim circuit to generate a pulse to contract the legs. The two ma parts of the circuit are the actuator components (LEDs, tra sistors, and nitinol, for user feedback and motion) and th pulsing components (555 timer, capacitors, potentiomete and resistors for generating the pulses to the nitinol legs)

The potentiometer will allow you to "fine tune" the gait work at an optimal speed. To fine-tune your robot, adju the potentiometer so that the leg actuators contract for or and one-half seconds and rest for one and one-half second

The 555 Timer Circuit

The 555 timer is a class of timing circuits used to generate known, periodic pulse. With the addition of a capacitor an two external resistors the 555 can be configured to produ this pulse without any external triggering pulse. An examp of this "astable multivibrator" is shown in Figure 1, and wi be used in this article.

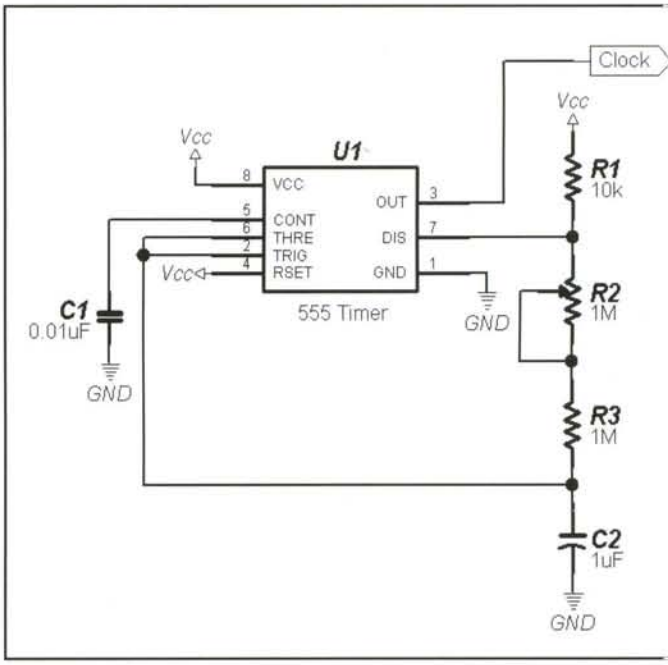


Figure 1: 555 Timer Circuit

The circuitry inside the 555 works based on the voltage sensed at pin 6 and 7. Suppose at time t=0 the output of pin 3 is high (supply voltage = Vcc) and pin 6 is low (ground voltage). The external circuit shown in Figure 1 will charge the 1.0 uf capacitor. When the voltage of pin 6 reaches 2/3 x Vcc, the output voltage of pin 3 will switch to 0 volts, and the 555 internal circuitry will start to discharge the 1.0 uf capacitor. When the voltage of pin 6 falls to 1/3 x Vcc, the cycle starts all over again. A timing diagram of the stable operation of this circuit is shown in Figure 2.

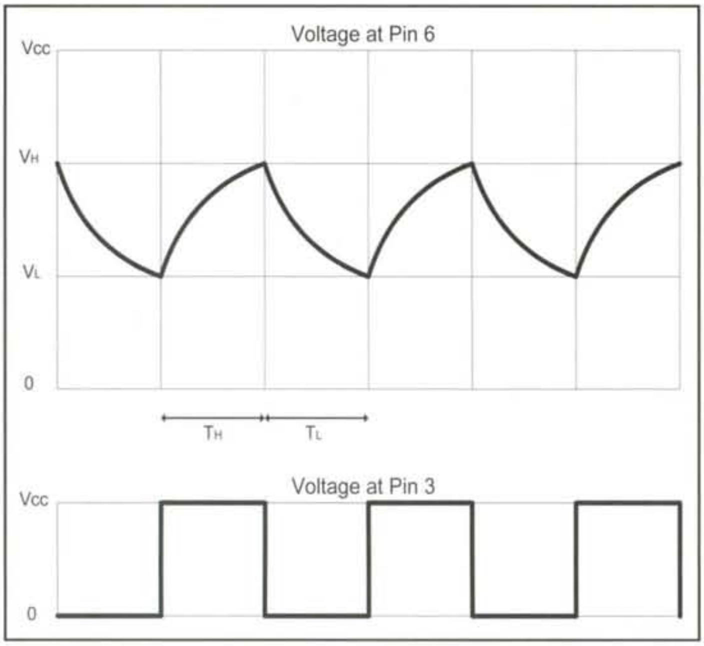


Figure 2: Timing of the Astable Multivibrator, 555 timer circuit

The times of the high and low outputs on pin 3 can be determined by using common RC (Resistor/Capacitor) network formulas. We define the following symbols as:

TL = The time of the 0 volts part of the pulse (low voltage on pin 3)
TH = The time of the Vcc volts part of the pulse (high voltage on pin 3)
P1 = the value, in ohms, of the Potentiometer (we will assume 0.5 MΩ)
R1 = the value, in ohms, of the R1 resistor (2.0 MΩ)
R2 = the value, in ohms, of the R2 resistor (10 KΩ = 0.01 MΩ)
C1 = the value, in microfarads, of the capacitor connected to pin 6 (1.0 uf)
ln2 = Natural logarithm of 2 = 0.693

The formulas are:

$$\begin{aligned} TL &= (P1 + R1) \times C1 \times \ln 2 \\ &= (0.5 + 2.0\ M\Omega) \times 1.0\ uf \times 0.693 \\ &= 1.733\ \text{seconds} \end{aligned}$$

$$\begin{aligned} TH &= (P1 + R1 + R2) \times C1 \times \ln 2 \\ &= (0.5 + 2.0 + 0.01\ M\Omega) \times 1.0\ uf \times 0.693 \\ &= 1.739\ \text{seconds} \end{aligned}$$

The obvious observation from these equations is that: if R1 >> R2, then **TL** will be close to T**H**.

The Transistor Driver Circuit

The ULN2003A is a 7-transistor Darlington Array. Pins 1, 2, 3, 4, 5, 6 and 7 are the bases of the Darlington transistors. Pins 10, 11, 12, 13, 14, 15 and 16 are the collectors of the transistors. All the transistors' emitters are tied to pin 8, which is connected to the ground in this design. When you apply +5 to +9 volts (logical 1) on a base, you turn its transistor ON and allows the current to flow from the collector to the emitter. By connecting one end of a nitinol wire to Vcc and the other end to the collector, you can control the current flow into the wire by applying proper voltage on the base. This circuit is shown in Figure 3.

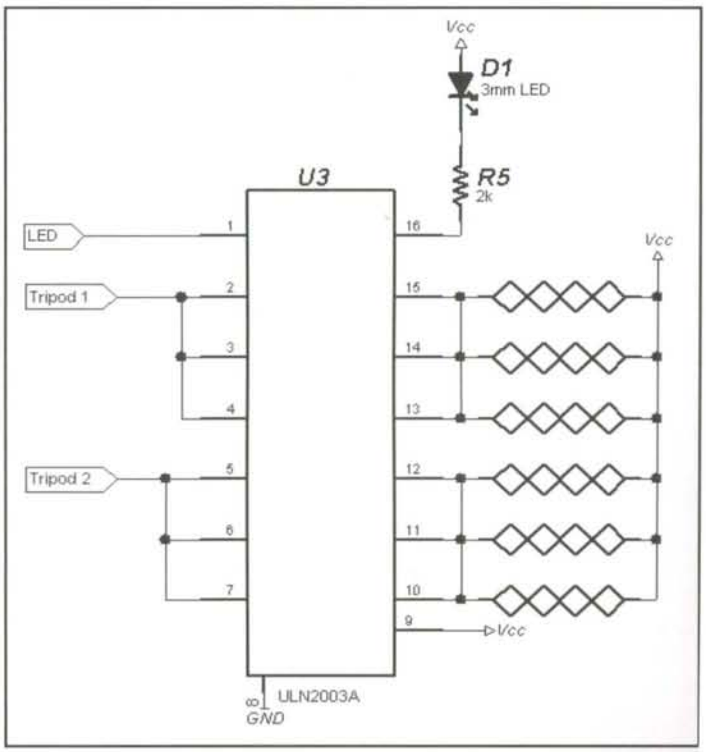


Figure 3: Nitinol Driver Circuit (Transistor Array)

Description of the Gait Generator using a Shift Register (CD4035)

The circuit schematics shown in Figures 2 and 4 are two of the three parts needed for controlling Stiquito's gaits. The final section of the circuit is placed between these two parts, and provides an optimal gait controller better than the one described in the book *Stiquito for Beginners: An Introduction to Robotics* [Conrad and Mills 1999]. The main purpose of this section of the circuit, shown in Figure 4, is to allow a tripod to relax before the other tripod starts contracting.

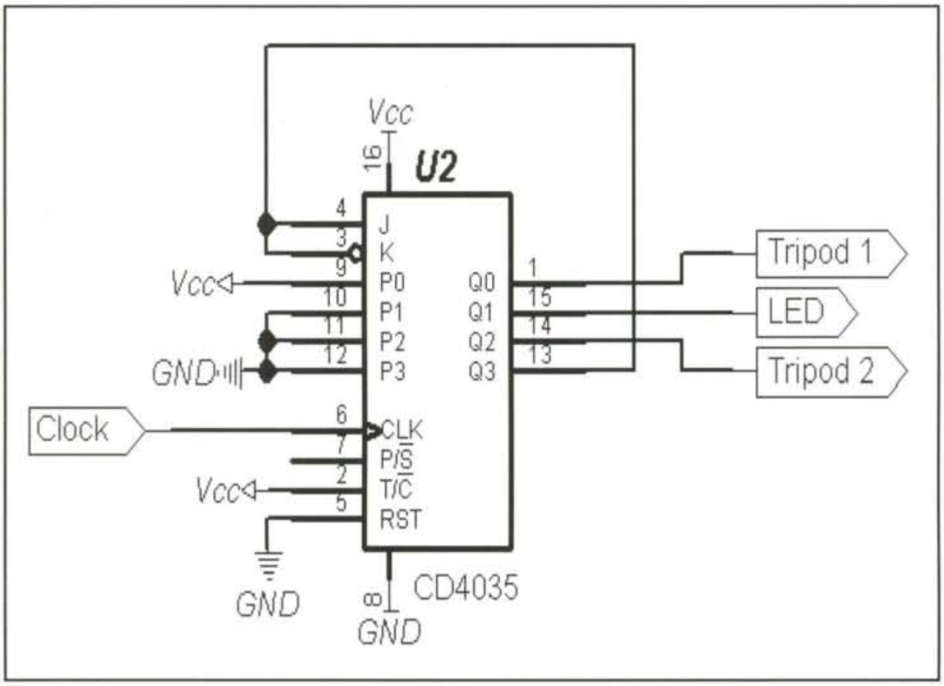


Figure 4: Shift-register Circuit

The CD4035 is a CMOS 4 bits shift register that can operate within 3 to 15 volts. This device is interesting because it executes parallel and serial loads. The parallel load will allow you to initialize the register with a specific value before the 555 timer starts sending pulses. The serial mode will be used to make Stiquito walk.

In order to make the CD4035 shift data correctly, you must connect OUT (pin 3) of the 555 to the Clock (pin 6) of the CD4035.

Pin P/S (pin 7) is the mode selector. When it is set to 0, the CD4035 is put into parallel load mode. When P/S is set to 1, the CD4035 is put into serial shift mode. By connecting an RC network to this pin, we can start the circuit in one mode and make it switch into another mode once the RC network stabilizes. In this case, P/S will be set to 0 (parallel inpu when you connect your battery to the circuit. Immediatel afterward, the capacitor C3 starts charging.

When the voltage on pin P/S reaches about 6.3 Volts, th chip considers it a 1 and switches into serial shift mod The voltage on pin P/S will stabilize to +9 volts (logical 1 well before the 555 timer starts pulsing since the time con stant (R x C = 2kΩ x 0.1uF = 0.2 millisecond) is very short

Pins P0, P1, P2 and P3 (pins 9, 10, 11, 12) are the paralle inputs. Their content is copied to the corre sponding Q outputs when the CD4035 i in parallel load mode (pin 7 P/S set to 0) By connecting P0 to 1 and the others to (only one Q ouput will be set to 1 at an given time.

Pins Q0, Q1, Q2 and Q3 (pins 1, 15, 14, 13 are the 4 outputs of the shift register. In se rial shift mode (pin 7 P/S set to 1), the con tent of each line will be transferred to th next one each time Clock (pin 6) goes fron 0 to 1 : the content of Q0 will be moved int Q1, while the content of Q1 will be move to Q2 and so on.

Pins K and J (pins 4 and 3) are the seria inputs of the shift register. Each time th Clock (pin 6) goes from 0 to 1, the conten of these pins is copied in Q0. By connect ing Q3 to J and K, we make sure the data i Q3 is not lost but returns instead in Q0.

Pin T/C (pin 2) is the True/Complement Output selecto You can use it to output on the Q pins the complement o the shift register's content (1 for zeros and zeros for 1). Sinc we do not use this feature, we connect T/C to Vcc to selec true outputs mode.

In short, the circuit works like this

- Upon powerup, the shift register loads a logical 1 on Q0 and zeros on Q1, Q2 and Q3.
- It then switches in serial shift mode and starts shifting its bits when the 555 timer starts pulsing.
- At this moment, the 1 present in the shift register will move from one Q output to another, endlessly.
- Each time a Q outputs a logical 1 to its transistor bases, the transistors are turned ON and the nitinol that is connected to it contracts. Otherwise, the nitinol relaxes.

(Continued on page 18

FOR MORE ADVENTURES WITH STIQUITO, READ THE BOOKS

Here we present our review of "Stiquito for Beginners." In the next issue, you'll read more about Stiquito in Contrad and Mills' "Stiquito: Advanced Experiments."

Stiquito for Beginners: An Introduction to Robotics

by James M. Conrad and Jonathan W. Mills

Published by The Institute of Electrical and Electronics Engineers Computer Society

The first time he saw a Stiquito walking under manual control, an observer asked the same question many non-robot builders ask: What is it good for? To find a cogent answer, build it yourself!

Stiquito for Beginners is a comprehensive book that introduces, teaches, explains, demonstrates and exercises the reader in many required skills needed for designing and building small robots. A half-dozen major contributors filled this book with clear schematics, simple diagrams, informational tables, close-up photos and explanatory drawings. There are many references; all easily available, some rather academic.

There are six quick and easy experiments designed to illustrate basic electrical principles, and three more experiments to demonstrate the basic mechanical abilities of Flexinol, a refined nitinol shape-memory alloy (SMA) actuator wire, manufactured and trademarked by Dynalloy, Inc.

This book is designed primarily to teach. Each chapter ends with a variety of exercises listed in order of difficulty. A solutions manual is available from the publisher, and a teacher's guide is available online, including more experiments, benchmarks, and standards.

Unique among robot books, Stiquito books provide both the skills and the *supplies* to build a small robot. You see, this is not a snap-together kit. Building Stiquito requires discrete component soldering, SMA wire fastening with knots, crimps and screws, and circuit board assembly, all of which are illustrated in the text.

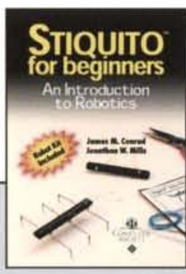
Conrad and Mills included a short overview of the history of robotics, contrasting descriptions of different fields in engineering, and details of the engineering design process, giving the reader an introduction to problem solving and production planning.

The reader needs basic math skills, precision hobby model building skills, curiosity and patience. Because any really good journey involves missteps, the authors include a troubleshooting guide and describe the principles behind logical problem solving.

Along the way, Conrad and Mills show us, step-by-step, how to build a tiny insectoid hexapod robot, a manual switch, a single board timer-based controller, and a PC-based parallel port controller using QBASIC and hexadecimal representation of digital signals.

Mills and Conrad answer our original question: "We build them because they are simple, they are fun, and they let us investigate robotics without getting a million-dollar grant."

But this understates the value of their endeavor. A bright teenager can follow this book, a teacher can learn from it, an entrepreneur can get help with his own projects. After experiencing the challenges of building Stiquito, you'll know the true reward of the science and craft of robotics.



Stiquito for beginners: an introduction to robotics
James M. Conrad, Jonathan W. Mills
1999, IEEE Computer Society Press
ISBN 0-8186-7514-4

Since only one Q output can be set to 1 at any time, the sequence is as follows :

Q0 is 1 = tripod 1 contracts and tripod 2 relaxes
Q1 is 1 = tripod 1 relaxes and tripod 2 relaxes
Q2 is 1 = tripod 1 relaxes and tripod 2 contracts
Q3 is 1 = tripod 1 relaxes and tripod 2 relaxes

and then repeats. Therefore, each tripod has time to relax before the other starts contracting.

Replacing the CD4035 with two CD4013

Even though the CD4035 shift register is a very convenient all-in-one chip, it is a little difficult to obtain. It is easier to make a shift register out of common D-flip-flops. The CD4013, a dual D-flip-flop, can be obtained much more easily. You will need two CD4013 chips to replace a single CD4035 shift register.

The operation of a D-flip-flop is very simple. When the clock input goes from 0 to 1 (rising edge of the clock), the value present on the D input is copied on the Q output. Afterward, Q will remain unchanged even if the D input changes. The output Q will only change when the clock goes again from rises from 0 to 1. Q prime always contain the complement of Q.

The S input is the Master Set. When this pin is set to 1, the Q output is unconditionally set to 1. The R input is the Master Reset. When this pin is set to 1, the Q output is unconditionally set to 0. We can use these pins to initialize our shift register by connecting the proper S and R pins to the RC network (C3 and R4) described in the previous circuit. The S and R pins that are not used are connected to the ground.

By connecting all the clocks together to the 555 timer, we make the D flip flop update their Q in unison. By connecting the Q output of the previous flip flop to the D input of the next, we allow the data to move from one flip flop to the next, just like a shift register would do.

All we have to do afterward is to connect the first and thir Q output to the ULN2003A to generate the proper gait se quence. This circuit is shown in Figure 6. Again, with thi circuit, we use the same 555 and ULN2003A circuit show in Figures 1 and 3, respectively.

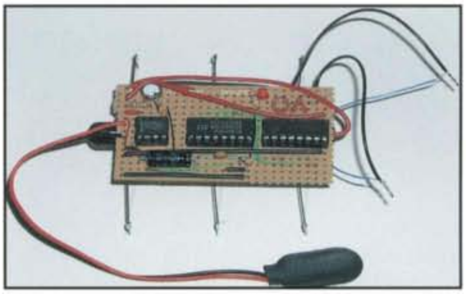


Figure 6: Shift-register Circuit Attached to Stiquito

So what can Stiquito be used for in the future? Legged ro bots have been shown in futuristic movies as a valuable fo certain applications. Surely Stiquito could have some ap plications, even though it is small. Some ideas included:

- Biodegradable insect robots for pest control on farms [Conrad and Mills 1997].
- Instead of sending one, very expensive wheeled vehicle to Mars, why not send thousands of small Stiquitos?
- In buildings of the future, "scavenger Stiquitos" would leave their home base in the walls and wander the floor in search of crumbs . . . and other bugs to eat!
- More nimble variations of Stiquito, equipped with microphones and cameras, could be sent into the rubble of a building to look for survivors.

All of these examples require the first step . . . a step!
The future of Stiquito is only limited by your imagination.

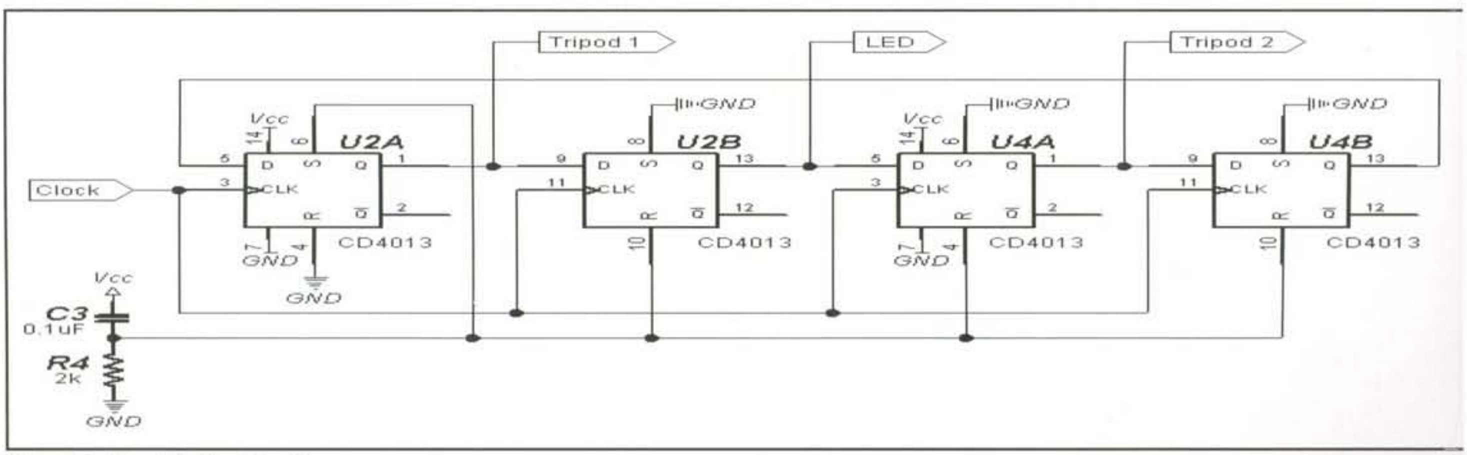


Figure 5: Four Flip-flop Circuit

Bibliography

[Conrad and Brown 1999] Conrad, James M. and Wayne Brown, *"Building an Inexpensive Insectoid Robot: Propulsion using Nitinol Wire,"* Robot Science & Technology, no. 4, pp. 12-19, January 1999.

[Conrad and Mills 1997] Conrad, James M. and Jonathan W. Mills, *Stiquito: Advanced Experiments with a Simple and Inexpensive Robot*, IEEE Computer Society Press: Los Alamitos, CA, 1997.

[Conrad and Mills 1999] Conrad, James M. and Jonathan W. Mills, *Stiquito for Beginners: An Introduction to Robotics*, IEEE Computer Society Press: Los Alamitos, CA, 1999.

[Conrad and Mills 1999a] Conrad, James M. and Jonathan W. Mills, *"A PC Based Controller for the Stiquito Robot,"* Circuit Cellar Ink, no. 108, pp. 18-22, July 1999.

Biographies

James M. Conrad received his bachelor's degree in computer science from the University of Illinois, Urbana, and his master's and doctorate degrees in computer engineering from North Carolina State University. He is currently an engineer at Ericsson, Inc., and an adjunct professor at North Carolina State University.

He is the author of numerous book chapters, journal articles, and conference papers in the areas of robotics, parallel processing, artificial intelligence, and engineering education. Conrad can be reached at: jconrad@stiquito.com.

Serge Caron received his technical formation at the College of Sherbrooke, Québec, Canada, where he got his diploma in digital systems. He is currently a technician at the University of Sherbrooke, where his hobby, robotics, became the biggest part of his job.

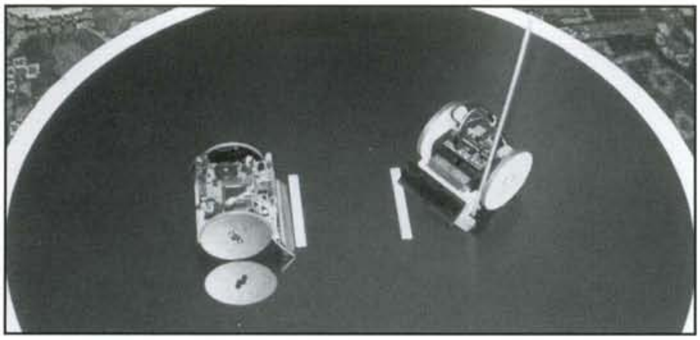
He is also a close collaborator of LABORIUS, the new robotic lab at the University of Sherbrooke. Caron can be reached at: serge.caron@gel.usherb.ca.

Customer Service

Submit Manuscripts, diagrams, glossy photos and disk (with article, code listing, and letter to the editor). Include SASE for return of materials. Writer Guidelines and an Editorial Calendar are available by sending a SASE to the editor. Do not embed art in your word documents. **Order New Subscriptions** for $24 for 6 issues in the US, $29 in Canada or Mexico. $49 for 6 issues in all other countries. In the US and Canada, call toll-free 888-510-RSAT (7728) with your Visa or MasterCard. All other countries must prepay by check or money order in US funds mailed to the address above. For faster service, subscribe via our secure server on our website at www.RobotMag.com. **Back Issues** are available and are black-and-white copies of the original magazine. Back Issue orders are $10 each in US funds. Please send check or money order payable to Robot Science & Technology at 3875 Taylor Road, Suite B, Loomis, California USA 95650. For faster service, call 888.510.7728 with Visa/MC or email service@RobotMag.com. **Questions About Status** of subscription and changes of address, email your complete name, new and old addresses to service@RobotMag.com or call toll free 888.510.7728 10:00a.m. to 4:00p.m., Monday through Friday, California time. **Letters and Feedback** are welcome. RS&T may edit letters for length or content. Email editor@RobotMag.com or snail mail Robot Science & Technology. **Reprints and Permissions**: fax purpose of your request to 916.660.0730. Reproduction of cartoons and artwork requires permission from the artist. Don't quote our articles on the Internet, please, it ultimately raises costs for all readers and steals from the writers. When copying for schools, we'd like to know, and we can even help.

***Warning!** We make reasonable efforts to ensure the circuits and code work, but we can't guarantee it. Our tutorials are meant for adult supervision. Make sure your adult is not accident prone. We cannot guarantee you won't hurt yourself while building stuff. All robot tests are conducted by a professional driver on a closed track.*

Advertisers Index



Build Your Own Robot Mini-Sumo Marvin Slyder Style

by Bill Harrison

If you've watched a robot mini-sumo contest, probably you've seen at least one robot built out of one of Sine Robotics' "Marvin Slyder" kits. These kits are very popular because they are so inexpensive and easy to build, yet make competitive Robot Mini-Sumos.

Why robot mini-sumo? Well, it provides an exciting competitive sport, using abilities to design and build robots. I get a kick out of seeing something that I created compete against another robot on its own.

Robot mini-sumo is a spectator sport in which two 4-inch square robots try to push each other off of a 2-1/2 foot diameter black disk. While competing in the autonomous division, the robot must do so without any help outside the robot.

The robots are made from scratch, parts of discarded gadgets, toys, and/or kits. They can be very simple to very high tech. All ages and types of people build them, and all have a chance to win. But what most competitors come out with, besides having a great time, is a better understanding of robotics.

The Marvin Slyder is easy to build since it's only made of a few simple, easy to get parts. The base is simply a small square of material, the wheels are just two disks of material with something on the rim to give some traction, and the scoop is just a angled part attached to the base. The motors, gears, driver electronics, wheels hubs, and housings are just easy to get remote control servos, modified to rotate continuously. A small simple controller, such as the "Basic Stamp" can be used. Sensors for the opponent and edge can be as simple as a couple of switches.

If you have some scraps of thin plywood laying around and are a little bit handy with hand tools, you can build this type of robot from scratch in a day or so. This article describes how I built just such a robot: "Old Tech." As well as the plywood, I also used a knitting needle for a sensor boom.

Old Tech is just one way to build a robot mini-sumo. I encourage builders to use this information as a starter, and try their own hand at designing a robot. I will not give complete exact specifications, because I want to encourage you to try your own ideas out. If you want to make a robot like Old Tech, there is enough information here for you to figure it out.

Designing Old Tech

The first step to building the robot mini-sumo was to figure out in my head what I wanted. I had some experience at this, from building many robots, so I just "picked through" tid-bits of ideas I had in my head. I wanted Old Tech to be inexpensive, made out of common materials, not require any special tools or abilities, a top notch competitor, and most of all fun to build. I used my Marvin Slyder kit as a model since it is not only very popular, but the design had been proven to work well.

I then drew down some parts of the design, using a soft pencil, paper and my easy chair. With a little erasing and redrawing, I set about making some hand drawings of the robot I envisioned. The drawing didn't look very "professional," but I could see if it looked like it would work.

There are a lot of little things that have to be figured out like: where would the batteries go (so they can be replaced easily), where would the servo motors mount, and how much room do I have to fit a sensor mechanism? With this first drawing I can work this out. It's easy to erase and redraw at this stage.

When I got to a design I was happy with, I sketched each part separately using the drawing of the whole robot as a reference. I then used these drawings to make the parts and assemble my robot.

Finding Materials and Tools

With a set of part sketches in hand, I set about gathering materials. Often I will find that I don't have all the materials laying around. Sometimes I go out and buy them, yet other times I just redesign around stuff I can find. I found all I needed laying around my garage.

The "core" material was a small piece of 1/4-inch thick Birch plywood, that was a scrap from some previous project. This type of high grade plywood can be purchased in small pieces from many hobby shops that sell raw materials for flying model airplanes. However, there are many types of materials that can be used, plastic is one example.

I also located an old "Basic Stamp" clone controller board. It's pretty simple and limited, but was all I needed for monitoring the two switches that I was to use for Old Tech's sensors.

I found two used Airtronics 94102 standard R/C servos for the motor/gearbox. Most servos can be used, as long as they are less than 3-15/16 inch wide when stacked back to back, with wheels attached (I'm going to use 1/4-inch thick wheels).

I set about to find a spot to work and collected the tools I needed. I also used safety glasses, a spot light, and a comfortable chair.

I now had a pile of all the things I needed to build Old Tech.

Building Old Tech

First I made the base. I marked out a square 3-1/8 by 3-1/8 inches on the inch thick plywood. I cut out the square with the coping saw. I used a course file to smooth it up a bit, and finished smoothing with some fine sand paper.

I located the two scoop mount holes. I made dents to help the 1/8-inch drill bit get started in the right spot (a nail can be use to make the dents).

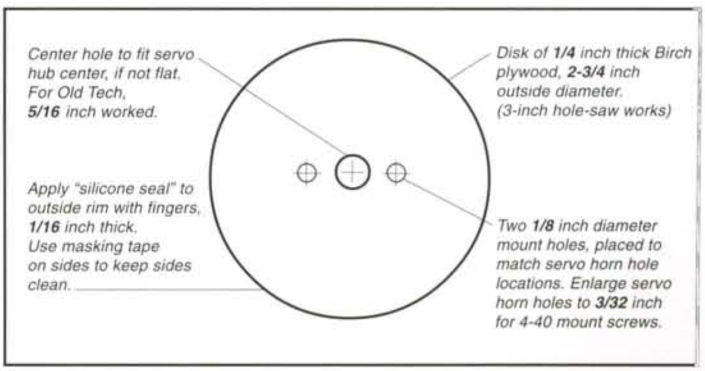
The cutout for the needle was made at a later stage of construction. I first built the robot, without the needle sensor. Then on figuring out where the needle was to mount, I cut away the scoop and base to fit. If you make yours the same way I did, your cutout will look like shown in the drawing.

Next I made a couple of wheels. I started by cutting out two disks, 2-3/4 inches in diameter, by using a 3-inch hole saw. You can carefully cut out the disks with the coping saw, using a compass to mark it out.

I enlarged the center hole to fit the raised center part of the servo hub. You would do this too, if your servo hub calls for it. I found two holes per servo hub, that looked good for mounting the wheel. I marked the location on the wheel, by holding the hub against the wheel and using a pencil through the holes I wanted. I drilled the holes in the wheels with a 1/8 -inch drill bit. I also enlarged the servo horn holes with a 3/32-inch drill bit.

I applied masking tape to the two sides of the wheel disks, to keep them clean. I used pieces of tape that overlapped the edges and trimmed to the edge with a sharp knife.

I wanted the wheels to have more traction in moving the robot around than just the wood. After all, robot mini-sumo is a pushing contest. So I used a type of silicone seal used as gasket material in cars, since it was a pretty blue color. But ordinary clear or white silicone seal in a tube, from most hardware stores works just fine. I applied a thin layer with my fingers, about 1/16-inch thick all the way around the outside rim of the wheel disk. This is what the masking tape is for, to keep the seal off the sides of the disk. You want it fairly round, but little bumps and imperfections are not a problem.



Old Tech Wheel Fabrication

After the seal hardened, I removed the making tape. Small bits of seal that stick out can be trimmed with a knife. I stretch the piece a bit, so it cuts easily.

I also countersunk the wheel mount screw holes, so that I could use flat-head screws. I needed to use these, as my robot would be too wide for the rules if I used regular screws.

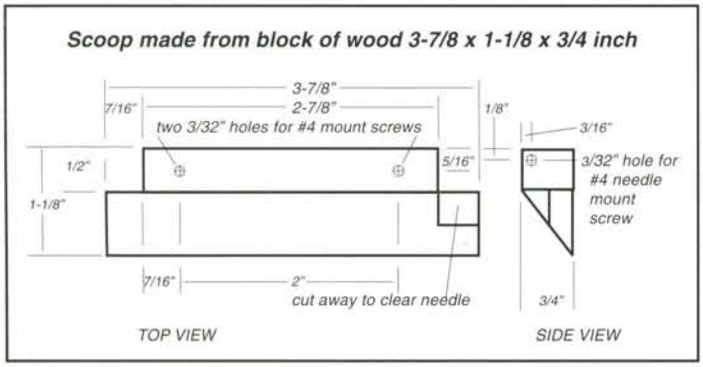
I modified my two Airtronics servos for continuous rotation. I took off the top gear housing, removed the little plastic part between the pot shaft and the output gear, centered the pot in its travel, cut off the stop on the output shaft, and reassembled the servo. It's now ready to use as the motor/gearbox for the robot.

I used 5/16-inch long 4-40 machine screws to mount the wheels to the servo horns. I used machine screws so that I wouldn't have the sharp tip of a wood or sheet metal screw sticking through. They just thread themselves into the 3/32-inch holes you've made in the servo horns, with a push as they are screwed in. I then attached the wheel and hub to the servo. Now the motor/wheel assembly is ready to mount on the robot.

Next I made a scoop. I started out with a piece of hard wood, 3-7/8 by 1-1/8 by 1-inch.

I made a mark an 1/2-inch from the back edge and cut the angle carefully. Holding the piece in a small vice helps. Use cardboard in the vice jaws so you don't make marks on the wood with the jaws. I cut it out with a coping saw and used a file to get the final shape, then did the final smoothing with sand paper. The front scoop edge doesn't need to be sharp, as we will deal with the scoop edge later.

I cut out the side wheel clearance notches. The left side (as a driver sitting in the robot would see it, on the right in the drawing) has a larger notch to also clear the needle sensor boom.



Old Tech Scoop Fabrication

Next comes the two scoop mount holes. They are **3/32**-incl so that a typical #4 wood or sheet metal screw will worl Note the needle mount hole on the larger notch side. It also **3/32**, but I used a longer machine 4-40 screw for thi Probably a wood or sheet metal screw would work as wel I mounted this to the base.

I made some marks on the base to act as guides for moun ing the servos. I made a line **1/2**-inch back from the front t guide the front sides of the servo. I made two lines **3/16** inc apart, in the center, perpendicular to the first line to guid the backs of the servos. I used two strips of "servo tape **1/4**-inch wide to stick each servo to the top side of the bas

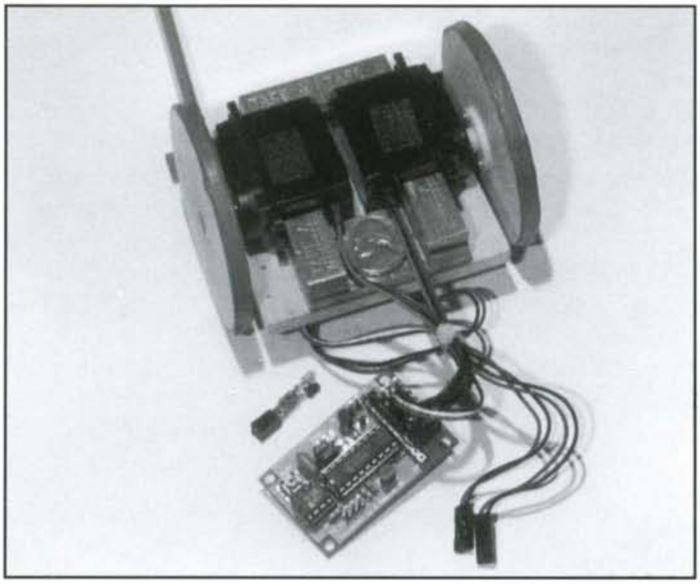
Servo tape is a double sided foam tape, used to mount R/ servos in remote control planes and cars. It is very stron much stronger that typical double sided foam tape. You ca get it at most hobby shops that deals with remote contr cars or planes.

I started taping the servos down by tilting the front insid corner of each into the marked corners I've made. Then tilted down one edge along it's line, then laid the whol servo flat lightly. I checked its position. If it's in the wron position, I lift it up and try again (if it picks up wood fiber start with new tape), if it's in the right spot, I pressed th servo down hard to make a strong tape bond. Now yo should have the basic robot, without sensors or electronic

I next mounted the controller. I just used two more strips c servo tape, about **1/4**-inch wide, to tape the controller to th top side of the servo motors.

Since this robot is to use a lot of power (relatively) to pus the opponent, I used two separate 9-volt batteries. One ba tery to power the controller, with a 5-volt regulator to lim the voltage. The other battery is to power the servos. I ra the full 9-volts to the servos, which is more than recor mended, and it shortens their life. This does make them g faster and stronger, which I want in competition. Using to high of a voltage can damage the servos quickly, but the seem to last long enough using a 9-volt battery.

With the clone controller I had, it was easy to cut a trace an plug the second power supply into the servo connection You can also make a second board and wire this up at ar other places on the robot. You can also put in switches an lights. I didn't use any power switches, as I just unplug th battery when I want to turn it off. Just be sure that the nega tive side of both batteries are connected, so all signals wi be in reference to the same "ground." Keep the positive side separate, however.



Controller

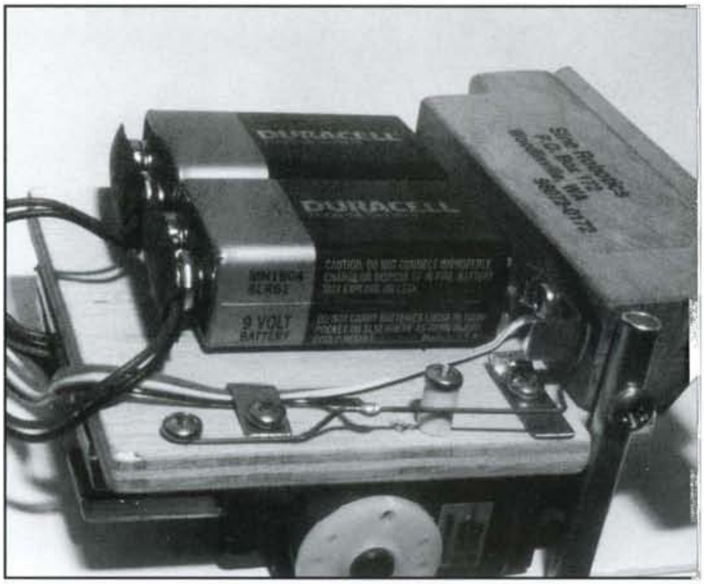
Next I mounted the batteries. I used 9-volt batteries because they are small and two of them mount, side by side, very easily. You can use four-cell, 6-volt battery packs too, but you may need to mount one on the bottom of the robot and the other on the top. I use a couple of **1/4**-inch wide strips of that servo tape to mount the batteries.

I bring a couple of extra batteries, with tape already on them, to competitions to aid in battery changes. I just rip off the old ones, and peal the protective tape cover off and slap them down and I'm ready to go again.

The controller I used came with a 9-volt battery connector that just plugged into a power header on the controller. I got a second one of the same type to plug the servo power in.

It's pretty easy to make your own connector out of a female header strip. Just cut off two pins worth and solder it to the battery lead wires. Be sure to get the positive and negative battery leads on the right pins when plugging the power into the controller. Plugging it in backwards can damage things by overheating. If you get it wrong, unplug it as quickly as you can to minimize the damage. This is one advantage of using switches, once plugged in, you just turn it on and off, without worrying each time of getting it wrong. You can also use polarized connectors which can't be plugged in backwards, and/or reverse polarity protection (such as using a diode) which prevents power to the controller if it's backward.

Battery placement is partly determined by where you want the weight. You want the weight forward enough so the front scoop doesn't hop too much. At the same time. You want it back enough to put as much weight on the wheels, to get maximum traction. With the Marvin Slyder design, you pretty much want the weight forward, since it uses such large wheels.

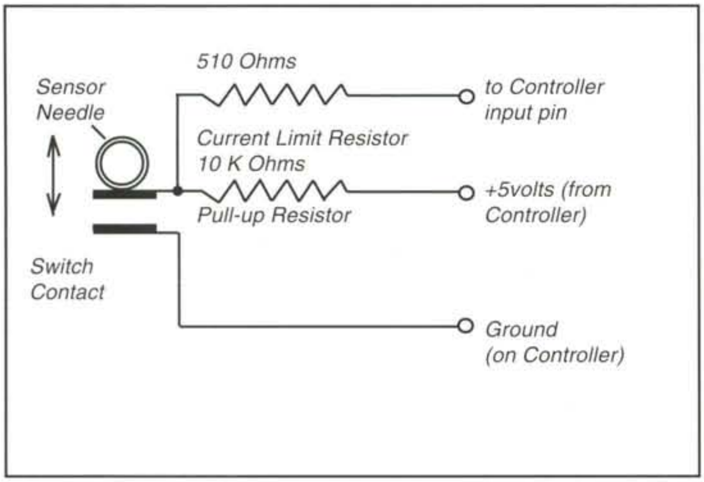


Batteries

Old Tech uses large wheels to get maximum speed with the servos. Changing wheel size is a tricky way to change the gearing to the wheels. I chose large wheels to give fast speed, since it still gets enough torque with the servos I used. Another advantage of the larger wheels is that the batteries mount conveniently underneath the robot. The batteries are easy to change out when needed, and yet all the electronics on top are easy to get to.

The Airtronics servos just plugged into my clone controller. All I needed to do was use three pin sections of a female header strip to make new connectors with and a slight modification to use a separate power supply for the servo power. I needed to make new connectors, since the original ones were too thick to mount side-by-side on my controller. This modification isn't always needed, and yet other times you'll need to do the connector on a second little board to make it work right.

Note that with the standard Airtronics connector, the ground is the center wire, but with others such as Futaba, the ground is at the end. Be sure you get this right when you power it up, or you can cause overheat damage pretty fast. Red is usually power and black is usually ground. Signal wires could be anything (take a guess what that blue wire might be). The signal has always been on the edge with all the servos I've seen.



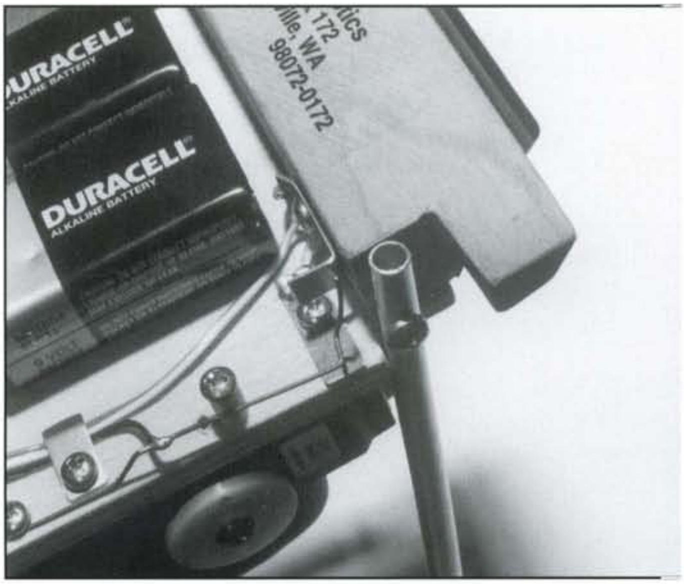
Old Tech Sensor Contact Wiring

While on the subject of controllers, it's a good idea to make up a "debug LED." It's handy to trace down problems when programming your robot. I use a two-pin section of a female header strip, a 330 ohm resistor, and an LED to make a debug LED. This can be plugged into the controller or the second little board if the controller doesn't have a ground or power pin next to a signal pin. Then when writing the software to control the robot, you can always turn the LED on or off when the program gets to a point of interest. You can watch for this to see if the program is at the point you want it to be, which is a big help in finding problems. For example, I used a debug LED to light up when my sensor detected the edge or an object. This way I could leave the power off on the motors and move the robot by hand to see if the sensor worked all right. If the LED doesn't light up when it should, try turning it around. It doesn't hurt it to be backwards, but it won't light up.

Next I built the sensor boom. I cut a length of knitting needle and drilled a cross hole **1/2**-inch from the back end. Mine is 10-inches long, but something like 7 or 8 would work better (10-inches hangs too far over the edge and can lose by touching the floor outside the ring). Also, a solid rod would be easier to use. Crochet needles are solid, so these can be used, but it doesn't sound a cute as "I used a knitting needle." The knitting needle gets caught in the mount screw threads (since it's a tube), so you'd have to use shrink tubing or something to smooth the threads out. Also I had to add little "BB" balls to the inside of my needle to get enough weight to activate the switches. With a solid needle, this wouldn't be necessary. I made the cross hole in the needle a little larger than its mount screw, about 1/2-inch from the back end, so that the needle can pivot sideways as well as up and down.

This is important as that's how we use one needle to dete both the edge or an object.

If you can find very small switches, then just mount the about 3/8-inch away from the pivot (either in front or b hind to suit). You would connect one side of the switch to controller pin through a 510 ohm current limiting resistc to protect the controller pin just in case raw power is shorte to the wire. This side of the switch is also tied, very weakl to the positive controller power through a 10 Kohm hol up resistor. The other side of the switch in connected to th negative side of the controller power, called "ground." Th way the controller input "sees" positive if the switch is n "hit," but it sees negative if it is hit. You can thus check th sensor boom position, in software, by checking this pin.



Close-up of Sensor Switch

My switches were just too big for the space I gave. You mig leave more room for switches. Make a test setup to test th out, before building the robot, so you can consider th needed space when designing it in the first place. I solve the problem by making my own home-brew switches. The were pretty easy to fabricate, though they took a little fi dling to adjust. I started by making a spring from some mus wire I got from a local hardware store (available in mus stores for guitars and pianos).

I fashioned one end in a loop so I could screw it down to th base. I fashioned the other end to hook around the end the needle. It had to allow the needle to point up at the sta of the contest and fall down to it's active state when cor peting (required by contest rules). It also had to move th spring up and down, as well as side to side, since I used th as one of my "switch contacts."

```
rem Program Listing

rem   otekmain.bas for Basic Stamp 1 controlled
                                           mini-sumo.
rem   connect second power to servo power
rem   tie negative sides together on two battery
                                           packs
rem   input pins:
rem   pin3   edge detect switch, 0=edge
rem   pin4   object detect switch, 0=object
rem   output pins:
rem   pin0   left servo motor, 0.7 msec = fwd
rem   pin1   right servo motor, 1.7 msec = rev
rem          pin6   edge detect LED, 1=on
rem          pin7   object detect LED, 1=on
rem   variables:
rem          b0     reserved for bit variables
rem          b1     reserved for bit variables
rem          w1 (b2 & b3) random number
rem          b4     random number capture
rem          b6     pivot counter timer
rem          b7     short pulse code width
rem          b8     long pulse code width
rem          b9     loop counter
rem          b10    not used
rem          b11    not used
rem          b12    reserved for gosub pointers
rem          b13    reserved for gosub pointers

dirs=%11000011               ' set port dir
pins=0                       ' clear output pins


b6=0                         ' initialize pivot
timer
b7=70 : b8=170               ' set pulse width code
pause 5000                   ' wait 5 seconds

for b9=1 to 10               ' drop boom
     pulsout 0,b8            ' left motor rev
     pulsout 1,b7            ' right motor rev
     pause 20                ' pulse code cycle time
next
     pin0=0 : pin1=0         ' stop both motors
for b9=1 to 10               ' let boom fall
     pause 20
next

for b9=1 to 20               ' move to opponent's
                                           side
     pulsout 0,b7            ' don't look at edge
     pulsout 1,b8            ' due to false detect
     pause 20
next

for b9=1 to 20               ' move to opponent's
                                           side
     pulsout 0,b7            ' left motor fwd
     pulsout 1,b8            ' right motor fwd
     pause 20
     if pin5=0 then loop     ' drop out if at edge
next
```

```
random w1: b4=b2/4+64        ' random 64 to 127
for b9=1 to 20               ' back away from edge
     pulsout 0,b8
     pulsout 1,b7
     pause 20
next

pin6=0                       ' edge detect LED off
goto loop
rem  main program loop
loop:
rem  check edge and object detect boom sensors
rem  risky attack =>  check object first
rem  safe attack  =>  check edge first

if pin4=0 then object        ' is opponent detected?
if pin5=0 then edge          ' is robot at edge?
if b6 > b4 then wander       ' is pivot timed out?
rem  pivot to search for opponent with boom sensor
     for b9=1 to 10
       pulsout 0,b7          ' pivot right
       pulsout 1,b7          ' to seek opponent
       b6=b6+1               ' increment pivot timer
       pause 20              ' pulse code cycle time
next


goto loop
edge:pin6=1                  ' edge detect LED on
     b6=0                    ' reset pivot timer
for b9=1 to 30' go forward
       pulsout 0,b7
       pulsout 1,b8
       pause 20
next
pin7=0                       ' object detect LED off
goto loop
wander:      b6=0            ' reset pivot timer
     for b9=1 to 20          ' wander forward
       pulsout 0,b7          ' don't check for edge
       pulsout 1,b8          ' as it willx xxxx be
                             false
       pause 20              ' due to tip-up on
                             start
next

for b9=1 to 40               ' wander forward
       pulsout 0,b7          ' move to a new spot
       pulsout 1,b8
     pause 20
     if pin5=0 then loop     ' drop out if at edge
     next
random w1: b4=b2/4+64        ' new random 64 to 127
goto loop

end
```

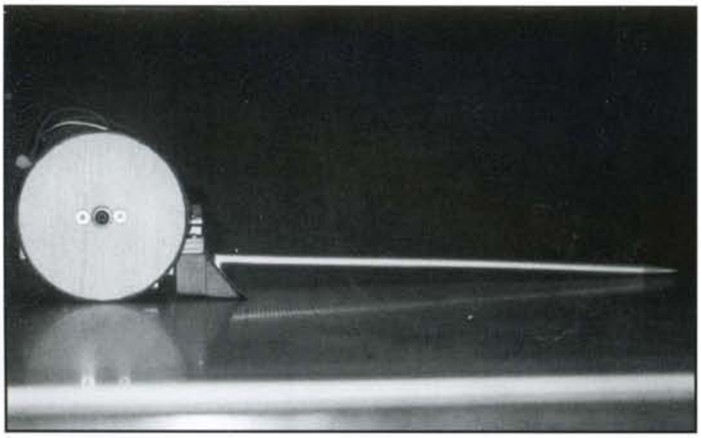
Regular solder doesn't stick to this type of wire, so I used silver solder. It used a special flux paste, and solder wire. It worked the same as regular solder, but stuck to the steel wire. I got mine at my hardware store. You can also just wrap the wire around the spring mount screw and tighten it down. This way you can avoid soldering to the spring wire all together.

I used bits of brass shim stock for the second contacts of both "switches." But brass or copper wire would work just as well. I mounted them down with more screws. I connected the controller ground to the spring, and each brass contact to separate pins on the controller (with resistors wired as described above). I tested the setup with a multi-meter set to "Ohms."

I next added a steel blade for a scoop. I used a small piece of five thousandths thick steel shim stock for the blade. I cut it to the size of the wood scoop front, leaving an area for the needle. I made it about 1/8-inch longer in front, so the leading edge would slide directly on the surface of the ring (raising the wood off the ring a bit).

I was very careful to round all the edges and corners, so that my scoop would not be a hazard for cuts to fingers and such. With a five thousandths thick shim you want the edge to be a full round, to keep it safe. Even at full round, it'll still be real close to "shaving" the ring, since this shim stock is so thin. I snipped a tid-bit off the corners and sanded them smooth, so that the corners wouldn't be a hazard either.

I used servo tape to attach the scoop blade. I had to tape it down a couple of times to get it in the right position, then I did the final adjustments by slight bends in the leading edge. I ended up with a scoop that rides on the rings surface from side to side. I'll bring a couple of backup scoops with me to competitions, so I can change them out if they get damaged.



Side view of Scoop

After I added the scoop blade, I could do the final "tweak ing" of the sensor switches. I just bent the spring and con tacts to activate where I wanted them to. This is a good plac to use those "debug LEDs," or you can use a multi-mete During these tests, I found that on my robot, the object de tect had have a stronger spring than the edge detect. Since was using just one spring for both, I found an alternative: mounted a screw and a plastic spacer half way down th spring to block its sideways movement. This made objec detect much more "stiff,"and yet let edge detect remai "soft." This tweaking was an advantage of making my ow switches. With regular switches, you might have to play wit where they are mounted, which can complicate adjustmen On my larger robot sumos, with this set up, I used switche with a bendable metal tab.

Programming Old Tech

I intended Old Tech to locate its opponent by spinnin around until the sensor boom bumped into the opponen Then Old Tech would push forward to push the opponen off the ring.

There is a rule that the robot can't start for 5-seconds afte the start of the match, to allow the operators to get out o the way. This is easy to program into the software.

There is a maximum start size. The needle laying down o the platform would be too large, so it has to start in the "up position. To get it down, Old Tech backs up after the 5-sec ond wait, then goes after the opponent.

So that Old Tech is more "aggressive," I programmed it t rush at the opponent right after the needle falls to the ring' surface, and before it starts pivoting. Old Tech doesn't re ally know when the needle is down, I just put a time dela in the software.

There is a rule that a robot can't stay in one spot for longe than 5-seconds. This is true even if it's pivoting, so Old Tec has to move to a new spot from time to time, if it doesn' locate the opponent. I added a: "time out," so the robo would wander to a new spot if it didn't locate an opponen in less than 5-seconds.

The trouble with moving, is that the robot can drive righ off the edge and lose. Now do you see why we needed tha edge detect capability? With an edge detector, Old Tech ca back away from an edge and not drive off.

Summarizing Old Tech's behavior: First the operator plug in the two power supplies, sets the robot down, holds dow the reset button till the start of the contest, releases the but ton, Old Tech waits five seconds after that, backs up, drop boom, lurches forward, and goes into a pivot to try to locate the opponent. Either Old tech finds the opponent, in which Old Tech pushes it off, or it times out, in which it wanders to a new spot. Old Tech will either time out on wander, in which it'll go back to pivot searching, or find the edge, in which it will back away from the edge and go back to pivot search. Old Tech will continue this until the operator either pushes the reset button or pulls the power connectors.

Study the sample software I wrote for my Old Tech. This is written in PBASIC for Basic Stamp I type of controllers. You can either use this as is, or use it as a model for your own software.

Testing Old Tech

Find a practice robot mini-sumo ring to test Old Tech's function. I started by just seeing if it worked OK without anything else on the ring. This is the best way to check out the edge detect.

Next I put a small wood block on the ring, about the size of a robot mini-sumo. I found that I had to fix a couple of things at this point. My main problem was that every time I saw an opponent, my sensor also touched the edge switch. Since I programmed Old Tech to be "careful," it always responded to edges first. Thus my robot backed away from opponents. This is not exactly what I wanted.

The solution was simple. I just changed to an "aggressive" strategy: I programmed Old Tech to always respond to objects first. This is sometimes called the "kamikaze" attack strategy. This isn't always good, but with Old Tech, it seems to work just fine. The final test was against my old champion robot mini-sumo: "RA." With several dozen rounds, Old Tech always won! So Old Tech is very competitive, one of my design wishes.

Preliminary operation testing can be done without a practice ring. Since the switches are the only sensor, the color of the surface doesn't matter. If you don't need to check the edge detect, the robot can do the object detect on any hard floor. To check the edge detect, use a table top. Hold your hands under the robot, so it doesn't fall if it doesn't work right (which does happen at times).

The last step before entering Old Tech into robot mini-sumo competitions, is to get it's mass up the maximum allowed, 500-grams (just over a pound). I used those tape-on lead weights for balancing car mag wheels. Most auto parts stores carry them. You'll need to find a scale to weigh the robot. I just snipped off what I needed and taped them down. Anything can be used to bring the mass up. I've seen robots with nails hot-glued on, to add mass.

Old Tech proved to be very inexpensive, easy to build, and very competitive in robot mini-sumo. But most importantly I learned a lot and had a great time doing it. I feel that this is what you will find too, if you build a similar robot.

The design and building of Old Tech applies to general small mobile robots as well, not just Robot Mini-Sumo. How about rigging it up with a slider tail and whiskers, and have it wander while avoiding obstacles? Old Tech will also teach you a lot of valuable lessons that will be useful in future robotics projects.

For more information on Robot Mini-Sumo, you can check out sinerobotics.com/sumo or you can e-mail me at: bill@sinerobotics.com.

May your robotics projects warm your spirit.

RS&T

'Mister Sumo America' Bill Harrison is an avid professional and hobby robot builder, using his mechanical engineering degree to invent small personal mobile robots. He is also experienced in robotics electronic and software and specializes in machine design. He is a member of the Seattle Robotics Society, heads the Eastside Robotics group, and has hosted the annual Northwest Robot Sumo Tournament for 8 years. Readers can reach him at his R&D company, Sine Robotics, at www.sinerobotics.com.

AIBO Second Generation: *New & Improved*

by Dean Creehan

Sony's AIBO ERS-110/111 has a new sibling who just might overshadow the original. The new model goes by the nondescript name, ERS-210, and is almost completely redesigned. At first glance, it looks similar to the original except in the ears and tail. In reality, very little of the original hardware remains. Even the battery is different. Let's look at some of the differences between the two breeds.

External Features

Each of the hardware modules as well as the main trunk of the robot has been redesigned. The overall appearance is based off of a lion cub, which is rumored to have been designed by Hajime Sorayama, who designed the original. After watching how current owners petted their AIBOs, Sony added two new touch sensors: one on the back and one under the chin. The one on the back was made possible by the elimination of the cooling fan. This removal makes the robot quieter and allows designers more flexibility. It also allows the Japanese to dress their AIBOs without worry of blocking the vent. In addition to being shortened, the ears were each given a motor to add to the robot's expressiveness. This somewhat makes up for the loss of the floppy tail. The new tail has lights to aid in showing mood and mode. Additional LEDs were also added to the eyes as well as a mode light. The openings for the battery and Memory Stick have been moved from the tail-end of the robot to the belly. The battery is new and the memory stick now has a locking mechanism to prevent removal while writing. After using the supplied tool to remove the head, legs, and tail, all that remains is a processing brick.

Internal Hardware Features

The ERS-210 comes with more horsepower under the hood. There, reportedly, is a 192 MHz CPU with 32 MB of internal RAM verses 100 MHz and 16MB for the ERS-110/111. The camera in the new robot uses a CMOS image sensor instead of a CCD. This should improve battery life since the new camera operates on about one-tenth of the power neede by the CCD. The motors are reported to be a bit stronge and quieter than the previous model. The addition of a Wire less LAN may be the most notable internal feature of th ERS-210. Removal of the external appendages is necessar to install the LAN card, which uses the IEEE 802.11b stan dard. The LAN will have a nominal working range of 3 meters. Initially, it will only work in conjunction with th $500 AIBO Master Studio for transfer of behaviors. Source have indicated that other uses will come in the future in cluding remote viewing. Unfortunately, the LAN and AIB Master Studio software will not be ready until next yea Also, the owners must supply their own wireless LAN car on their PC.

Main Software – AIBO Life

Something unique to the new AIBO is that its necessary soft ware is sold separately. There are several different Memor Sticks containing software to choose from, but AIBO Life i the "must have." This program corresponds to the softwar that came with the 110/111 models. It allows autonomou behavior by giving an AIBO instincts, emotions, and th ability to learn and mature. AIBO Life also enables th robot's voice recognition and mimic features. The ability t use the voice features is tied to the maturity model so tha more features are enabled as the robot "grows up." Usin the "Record a name" command, AIBO will learn its nam and will react in the future when the person who recorde it calls him/her. Since most owners named their ERS-110/ 111s, this is a natural progression. Some examples of th other 40 or so voice commands are "Hello, Good bye, I'r home, Good night, Good, Bad, Say your name," (says it ir tonal language), "How old are you?" (will indicate his age) "Sit," and "Take a picture." The 210's voice recognition i reported to be over 90 per cent accurate. The Mimic Mod is a unique feature of the new AIBO. The robot will imitat the sounds of words you say using a tonal language which matches the length, pitch, and inflection of phrases you say to it. If the owner says, "Goodbye," the AIBO will mimic the words with something like "Beep-Bop."

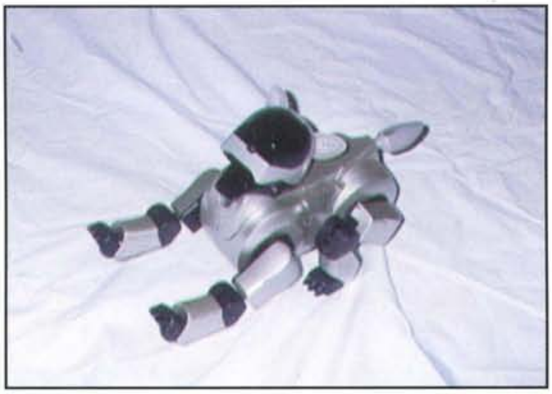
In addition to the Autonomous Mode, some new modes were added to AIBO. Since there is no Sound Controller (remote) with the new model, either voice or use of the touch sensors is used to enter or exit a mode. There are now Rest and Held modes, which would have been nice to have in the first generation AIBO. They allow movement of the head and tail while the legs are either limp or put into a sitting or lying position. That allows the AIBO to sit on your lap while you and your mechanical pet watch Battlestar Gallactica together.

The ERS-210's emotional model appears to be similar to the original AIBO, but a new instinct was added. The Sleep instinct makes the AIBO want to sleep. If an AIBO had just returned to Autonomous Mode from Sleep Mode, the Sleep instinct would be still strong, and the AIBO might seem a little groggy and absent-minded. The growth model is also very similar to the ERS-110/111s. One difference rumored at some Japanese sites is that an adult AIBO can switch between types. In the original AIBO, once the adult stage was reached, no further "growth" would occur.

The AIBO Life software also takes snapshots using the robot's color camera. There are two picture-taking methods. First, an owner can use voice command. When the command is given, AIBO counts down using six LED segments in the eyes, turning them off one by one. After six seconds a picture is taken. It is rumored that AIBO can hold seven pictures at a resolution of 180x140. It's a Pet, it's a Robot, it's a Digital Camera! The second type of pictures is the one AIBO takes each day to create a diary. The AIBO Fun Pack PC software is needed to view the pictures.

Other Software – AIBO Fun Pack

The AIBO Fun Pack runs on a PC and is used in conjunction with the AIBO Life Memory Stick. The owner can check on AIBO's level of maturity or use the calendar function to plan special events and set AIBO's wake-up time. You can also view AIBO's diary, which includes a daily photograph along with a phrase on how it was feeling each day. This could be something like "I was happy today," "I played with my pink ball all day," or maybe "Today I studied the papers of Werner Heisenberg and believe I found a flaw in his work in quantum mechanics." Well, maybe not that last one, but you get the idea. AIBO retains the last seven days worth of its diary. The pictures taken via the voice command are also viewable using the AIBO Fun Pack. Along with the software, the pack comes with AIBO wallpaper and icons for the PC.



Hello AIBO

There is once again a Hello AIBO Memory Stick, which performs the same function as the one for the ERS-110/111. If it is designed like the original software, it is a youth based AIBO which has all the functionality of the AIBO Life software sans the growth routines and AIBO Fun Pack interfaces. The biggest difference between the old and new versions is that people outside of Japan can purchase the ERS-210 version. The old version was only available for sale in Japan or as a promotional item in the U.S.

Party Mascot

When you want to have some fun with your new robot at, say, a Christmas or New Years party, put in the Party Mascot Memory Stick. This software has four gaming categories to choose from. One is its namesake "Party Mascot" where you can play interactive games. There is "Athlete AIBO" where the robot can compete in sporting event with other AIBOs. Another category is "Entertainer AIBO" where the ERS-210 sings, dances, and becomes the life of the party. Unfortunately, this makes your guests forget you are even there, and you will find your AIBO getting invited to more parties than you. The last category is "You and Your AIBO" for head to head competition. Actual detail on any of these categories is unavailable at this time.

AIBO Master Studio

The ERS-110/111's $450 AIBO Performer Kit has been replaced by the $500 AIBO Master Studio for the ERS-210s. This software is not available yet, so details are sketchy. The Performer Kit only allowed the editing of performances. The new software will add LED control, sound coordination, and actual behavior modifications. This latter capability might open up interesting possibilities because it adds conditional logic. It appears to be similar to a very high-level programming language where individual actions or motions become subroutines that can be selected based on external stimuli. For instance, the AIBO could be instructed to head-butt its pink ball. What is not known is the level of detail allowed in the conditional clauses. The Wireless LAN Card, sold separately, will initially only work with the AIBO Master Studio. It will be used to wirelessly transfer new behaviors to the robot. This will make programming less tedious. Without this feature, the current programming of motions is an endless cycle of editing, downloading to a Memory Stick, transfer of the stick from PC to AIBO, rebooting of AIBO, setting of AIBO into Performance Mode, executing performance, noting where problems are, shutting down AIBO, transfer of stick back to PC, reloading Memory stick, back to step one. There is a 3-D AIBO that can be used to view general movements, but you need to run it on the real thing to see the problems.

Extras

The charging stand, now called the Energy Station, has been completely redesigned for the ERS-210. It is a wedge shaped device with an LCD display on the front that can be used for checking/setting AIBO's time, date, and volume. It also allows monitoring of the battery condition. Since the "X" shaped base from the original charger is missing, the new station allows for the future addition of self-charging. Unlike the 110/111's charger, AIBO's feet are always on the ground, so mounting and dismounting are plausible.

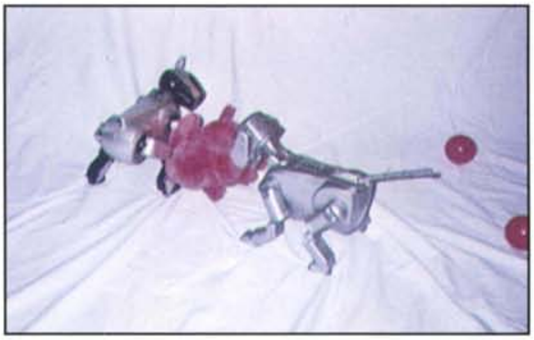
One big negative change in the ERS-210 for hackers is the introduction of the red Memory Stick. These Memory Sticks are set up to work with only one kind of software. This way, an owner can only load an update, a back-up, or a hacked version of software if he already owns the appropriate Memory Stick. For example, the Hello AIBO software can only be restored to a Hello AIBO Memory Stick. This is a compromise to protect Sony's software while maintaining the ability to do back-ups and upgrades.

A special stick called the AIBO Programming Memory Stick is needed to save programs created with the AIBO Master Studio software. This would appear to limit the scope of modifications from the AIBO Master Studio software, since neither AIBO Life, Party Mascot, nor Hello AIBO can be changed.

ERS-210 Demo Model

Sony graciously supplied *Robot Science & Technology* with a pre-production demo model of the ERS-210. Though not finished, the model gave valuable insight into the next generation AIBO. One of the more visible features is the extensive use of LEDs. For happy emotions, the eyes use the newer blue-green LEDs, which are more brilliant.

Also, to get more expression, Sony utilizes the ability to independently illuminate individual eye segments. This robot likes to wink. There is also a solid bar at the top of the faceplate, which uses a yellow-green LED. It only came on



only twice, when it appeared the AIBO was doing some so
of reset. It was noted at those instances that the new AIB
performs a graceful fall and self-reset instead of collapsin
into a heap and requiring a manual reset. The tail assembl
uses blue and orange LEDs feeding into a single fiber opti
line to the tail. It was off or blue most of the time, probabl
indicating happiness.

The demo software was a representation of AIBO Life bro
ken into two pieces on two separate Memory Sticks. On
stick was for autonomous activity, and one was to demon
strate the voice capability. The voice recognition had prob
lems with children's voices, but always seemed to work wit
an adult male. This, of course, directly matches the predomi
nant demographics for the buyer of this product. The mimi
mode could be enjoyed by everyone because the robot woul
imitate anyone's voice regardless of pitch. The autonomou
stick was similar to the first generation AIBO.

One thing new was the inter-breed communications. Th
ERS-210 would listen for certain tunes from a nearby ERS
110/111, e.g., the sound after praising. After recognizing
tonal sequence, the ERS-210 would sit down and procee
to emit Sound Controller commands. It tried out at leas
four "Hello" commands: DAF#, ADF#, EGC, and GCE. Th
first two appeared to cause the ERS-110/111 to blink its gree
eyes and make a happy sound. When the ERS-110/111 hear
one of the second pair of commands, the robot sat down
waved with a paw, and played a happy tune. This prompte
the ERS-210 to wave a paw while playing a similar tune
Also, on two occasions, while the 210 was trying to get th
110's attention and the 110 played an angry sound (becaus
it missed the ball), the 210 played the identical angry tune
It appeared to be mocking the 110. Disassembly of the nev
robot was a snap with the supplied "pin" tool. The littl
display on the internal processing brick can show an
change the time, date, volume (0-3), and UTC (-12 to +12)
whatever that is. A simple indication of remaining batter
life is also displayed. Another interesting feature of the bric
is an extra on/off button and status light on the top. Thi
was obviously placed to maximize future flexibility. Wit
it, someone could design an upright AIBO.

Some other miscellaneous observations are noted here. Th
back and chin sensors reacted to being stroked. The me
chanical pet stopped what it was doing, lay down, made

clicking noise, and showed star-burst eyes. Upon stopping the stroking, the robot lit its lower red (sad) eyes and began emitting a pining sound.

The chin sensor is a type of membrane switch that reacts to a very light touch. On further investigation, the "clicking" sound was traced to a problem with the ears that did not move on the demo model. After watching the new robot for several days, it appeared that the main contributor to ear damage was falling. Because of the way they stick out, the ears often take a solid hit when the ERS-210 tips over.

Also, several times the 210 exhibited head vibration. The new robot does snap its head quickly left and right when navigating a room. The 110/111 would turn its head much more slowly. Another noted difference in the new AIBO is that it uses a circular scan pattern with its head when searching for the ball. The previous robot used a linear pattern. This gives the new robot an eerie "Exorcist" look about it. The infrared distance sensor in the head is now aiming out of the nose along with the camera. This should allow for better correlation between vision and distance. The ERS-210 is much quieter than the original, mainly because of the missing cooling fan. For charging a battery without the station, the battery must be in the robot. This is a bit inconvenient when working with multiple batteries.

Selling Differences

One of the main reasons the new AIBO's price seems cheaper is because all of AIBO's accessories are sold separately. When you add back the pieces that came with the original robot, the price gap narrows. Some cost is removed because there is no sound controller needed. The ERS-110/111 was an Internet and phone item only. This generation AIBO is available via the Internet, phone, and at retail outlets including the three Sony stores (New York, San Francisco, and Chicago). The black version of the robot can be purchased at all The Sharper Image stores. RS&T

Dean Creehan has been tinkering with robotics and other inventions since childhood. He received a BSEE at Carnegie Mellon and an MSEE at USC, where he studied Robotics, AI, Machine Vision, and DSP. Dean is currently a Systems Engineer at Lockheed Martin Global Telecommunication.

CRASH -N- BURN SUMO ROBOT

by Jim Frye

Autonomous Sumo is gaining popularity across the country. If you want to build a winning robot, build the strongest, fastest machine possible. The robot must be able to detect the edge of the ring to avoid running out. Search out the opponent if possible, but use of a random or repetitive motion will result in contact well within the three-minute time limit. Crash-n-Burn was built using a relatively low-tech approach, but the overall design is sound and has features that make it well suited for Sumo. At the time of this writing Crash-n-Burn has been undefeated in competition.

COMPONENTS

The robot is made up of several important components. The motors, tires and wheels, chassis, batteries, microcontroller, and sensors all fit together in a way that complement each other. I decided to use Hitec HS-700BB hobby servos as the drive motors. They are relatively cheap and, like all hobby servos, they are very easy to control with a microcontroller. The following math was used for a simplistic ballpark analysis of the servos and the 3-inch diameter wheels used on the robot. The servos are rated at 161-oz./in. of torque each. Thus, at one inch from the center of rotation of the horn, a force of 161-ounces is generated. At the circumference of the 3 inch diameter wheels used (1.5-inches from the center of rotation) the force is reduced to 107.3-ounces. Dividing this result by 16 for the force in pounds and doubling it because there are two servos results is a total of 13.4-pounds of pushing force (disregarding loss of traction). That is plenty of power for the application. They do require modification for continuous rotation, but this is a straightforward operation and doesn't even require soldering. Follow this step by step procedure to do the modification.

PROCEDURE

1) The output shaft is limited to about 180-degrees of rotation. Rotate the output shaft by hand from stop to stop to find the center of rotation. Keep the output shaft in this cer ter position and remove the round servo horn for the fo lowing steps.

2) Remove the four long screws from the bottom of the cas and disassemble the housing. Take out the screw that hold the feedback potentiometer in place as shown in Photo and carefully remove the pot from the case. This part is sti required for the servo t operate, so don't cut th wires.

Photo 1

3) Remove the ball bea ing from the output sha of the final gear. Cu away the mechanica stop on the final gear us ing a razor saw, as illus trated in Photo 2. Clea all of the debris left fror the saw, as small pa ticles will gur up the gea train.

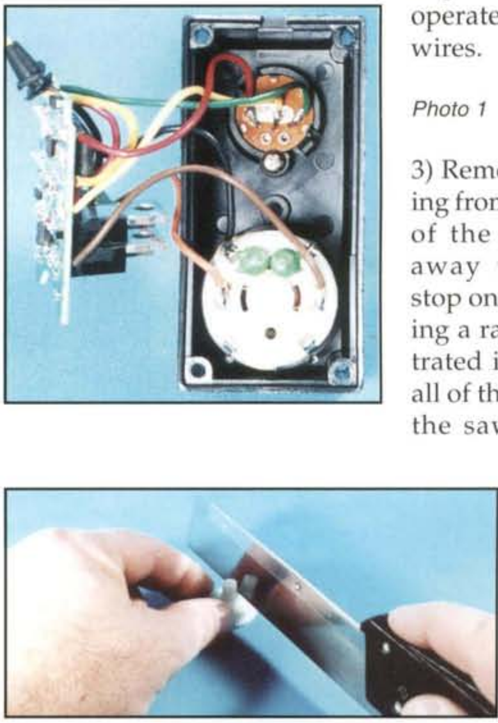


Photo 2

4) Cover th pot with small piece c electrical tap to insulate it from the electronic components. There i enough room for the potentiometer to be left inside the cas Just push it into the corner away from the motor and rea semble the bottom of the housing.

5) Replace the gears as shown in Photo 3, and reassemble the rest of the housing. The servo can now be controlled by sending a 1-ms or 2-ms pulse. 1-ms will rotate the shaft clockwise at full speed, and 2-ms will rotate the shaft counter clockwise at full speed. Simply stop sending pulses to stop rotation.

Photo 3

The tires and wheels are the 1061 Road Hawgs made by Pro-line. They should be available at your local RC hobby shop. They fit on a 2-inch rim, are just under 3-inches in diameter, and have an overall width of just under 1.75-inches. They are a soft rubber compound with an aggressive tire tread. The nylon rims mount directly to the round servo horns after they are drilled for common 4-40 hardware. Photo 4 , shows the finished product ready for installation.

Photo 4

The dimensions for the chassis and structural components are shown in Figure 1. These parts were cut from 6-mm thick foam PVC panel using a band saw. The servos mount back to back on the main chassis, as shown in Photo 5. The second smaller panel holds them securely in place as illustrated in Photo 6. Double sided tape was added to hold the servos in securely. A small 1.125-inch spacer is used along with #4 self tapping hardware to hold everything together. The small panel also makes a nice platform for mounting the control electronics.

The scoops are mounted at approximately 70-degrees. There is nothing magic about this angle, it just happens to be how far the scoop could lean back and still allow room for the battery packs. The scoops are reinforced with supports. The distance between these supports is just enough to fit a standard 7.2-volt battery pack. I use a medium cyanoacrylate (super glue equivalent) called Pro CA+ sold by Great Planes. This glue bonds in 10 to 15-seconds and, as long as there is sufficient surface contact, the resulting joint is actually stronger than the parent PVC material. Two reinforcing beams were added to the bottom of the robot that span from one scoop to the other. This really stiffened up the entire chassis, see Photo 7.

Two weights were also added for mass and balance. The bottoms of the scoops are sanded flat so they would ride evenly across the surface of the ring. A competition of otherwise evenly matched robots can be decided by which robot gets its scoop under their competitors.

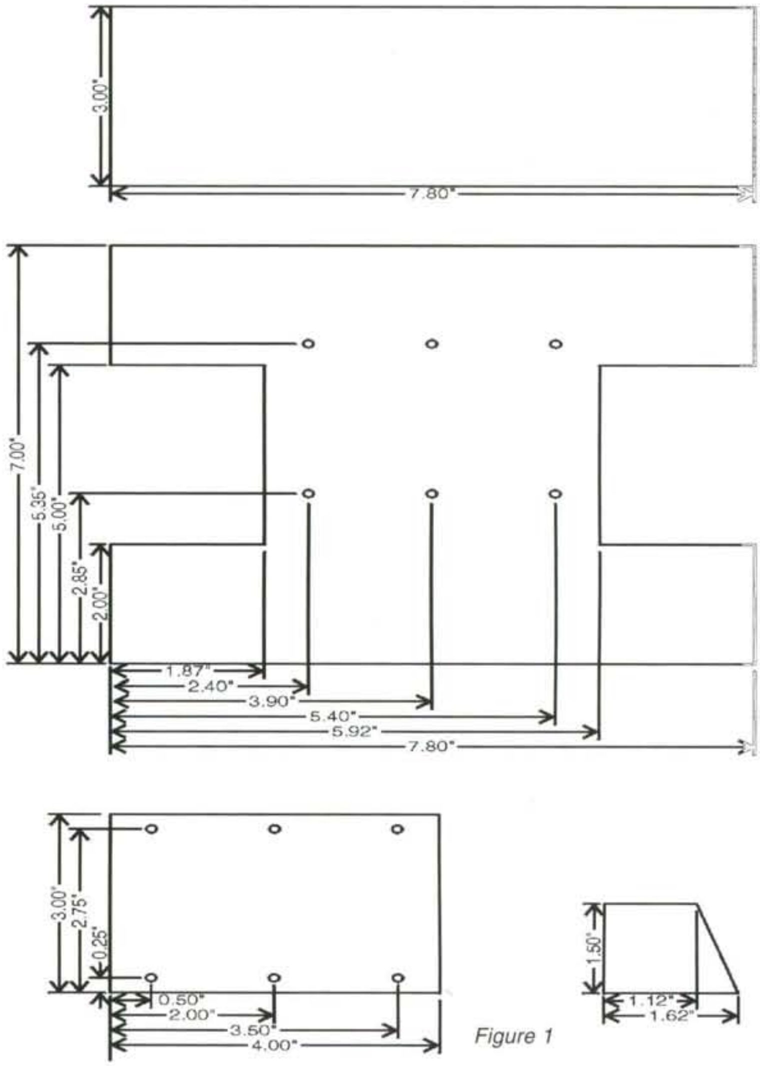


Figure 1

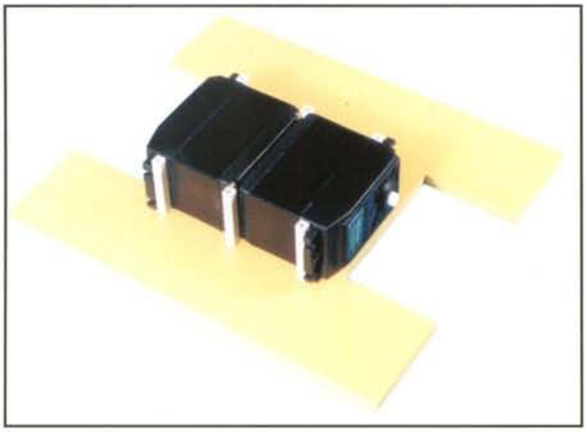


Photo 5

Photo 6

Photo 7

At this point, the basic chassis is complete and ready for paint. Many robots use infrared reflective sensors to search out the opponent, therefore I chose flat black paint for Crash-n-Burn. This makes the robot very difficult to detect. Although it may have compromised the stealthy nature of the robot a bit, I opted to add some cool decals to the scoops. CorelDRAW was used to design the decals and they were printed on adhesive backed paper.

A color laser was used, but I imagine any color printer woul suffice. After application, I gave them a coat of clear ename to prevent damage during competition. Photo 8, shows th finished robot.

A separate 7.2-volt battery was used for each servomoto The weight of the extra battery pack isn't a problem and i allows a much longer run time between charging, or bat tery replacement. Don't underestimate the importance o run time. Depending on the number of robots entered, an the type of elimination trials employed, the robot may nee to compete up to 20 times in a two-hour span. Run time ca make or break a robot's chances for success. A separate 9 volt battery is mounted for the microcontroller and senso

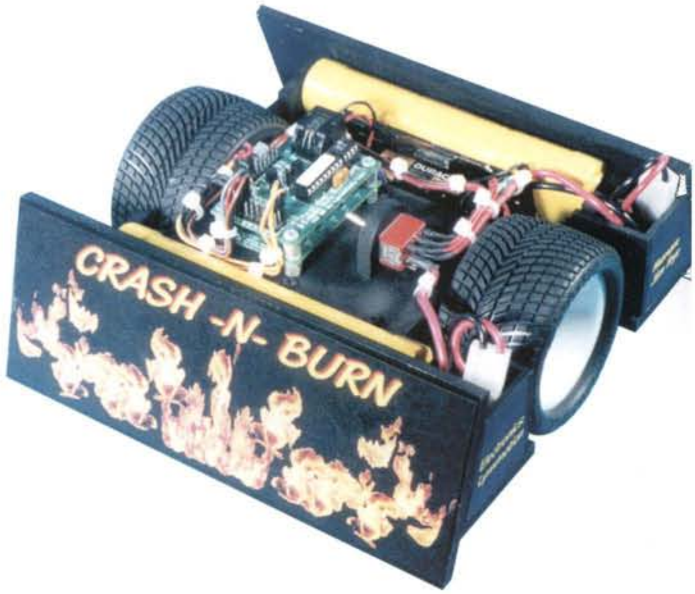


Photo 8

electronics. This provides isolation from voltage spikes tha are present on the higher powered servo motor supply line Note that there is an edge detector mounted just behind eac scoop. A robot with only one scoop would need to backu and turn when it reaches the edge, leaving it vulnerable t attack. Crash-n-Burn does not need to backup and turn whe it reaches the edge. It only needs to reverse direction an look for the edge with the opposite sensor.

The schematic diagram can be seen in Figure 2. The robot' control electronics are actually off the shelf componen available from Lynxmotion, with a couple of minor modifi cations.

The First Step (BASIC Stamp 1 based microcontroller) ha provisions for plugging servos directly onto the PC boar However, since I chose to control these servos with sepa rate battery packs, I had to breadboard the servo conne tions off the board.

The other modification is to the Tracker. The Tracker was designed to be mounted to the front of a robot and allow easy line following. These infrared sensors just happen to work exceptionally well at detecting the white ring at the Sumo ring edge. Simply wire the sensors off the PC board using standard 24-AWG wire, and heat shrink to protect against shorts. They can then be mounted exactly where they are needed, see Photo 8. The sensors have a wide operating range, and will work when spaced from between 0.0625 and 0.5-inches from the floor. The sensors used are Kodenshi SG-2BC reflective sensors, but most any reflective IR sensor should work well. You may need to experiment with the pull-up resistor value when substituting. The Schmitt Trigger inputs of the 74HC14 use hysterysis to clean up the signal. The input voltage must go to 1.7-Vdc before the output will go low, and must drop to 0.9-Vdc before the output will go high. This helps prevent erratic signals from reaching the microcontroller. The wiring of the robot isn't extremely critical, just try to keep runs as short as possible. Liberal use of tie wraps will ensure everything is secure. All of the servo power distribution connections are soldered, but removable jumpers were relied on for the signal interconnections.

The software that controls Crash-n-Burn is extremely simple. It consists of two loops, a forward loop and a reverse loop. When starting all autonomous Sumo competitions require a five second delay before any motion is allowed. The pause 5000 takes care of that requirement. The program immediately drops into the forward loop. In this loop, the robot moves in the forward direction indefinitely until the forward edge detector takes the First Steps pin seven high. When it detects the edge, the program drops into the reverse loop. In this loop, the robot moves in the reverse direction indefinitely until the rear edge detector takes the First Steps pin 6 high. When this condition occurs it jumps back to the forward loop. Simple right?

```
symbol f_sens = pin7
symbol r_sens = pin6

pause 5000

forward:
pulsout 0, 100              'rotate the right wheel CW
pulsout 1, 200              'rotate the left wheel CCW
pause 10                    'wait 10mS
if f_sens = 1 then backup   'check sensor and branch
goto forward

backup:
pulsout 0, 200              'rotate the right wheel CCW
pulsout 1, 100              'rotate the left wheel CW

pause 10                    'wait 10mS
if r_sens = 1 then forward  'check sensor and branch
goto backup
```


Crash-n-Burn was constructed for a Sumo competition that offered a middle weight class. The rules were based on the standard Japanese rules, but the weight was limited to 1.5-kg (3.3 lb.). I believe the same concept would be successful with larger servos for the standard 3-kg (6.6 lb.) class. The robot performed flawlessly winning most matches within 30-seconds. Due to a limited amount of wheel traction slippage, the robot exhibited an interesting emergent behavior. If the robot contacted an opponent with the edge of its scoop, the robot would automatically steer into the rival, resulting in a self centering behavior.

I still consider myself a beginner in robot Sumo, having only attended three competitions so far. Every new event we attend has a little tougher competition, resulting in my having to try a little harder as the bar gets raised.

It is my hope that this article will help you to be more successful in your Sumo robot building.

Good luck, and Happy Roboting!

RS&T

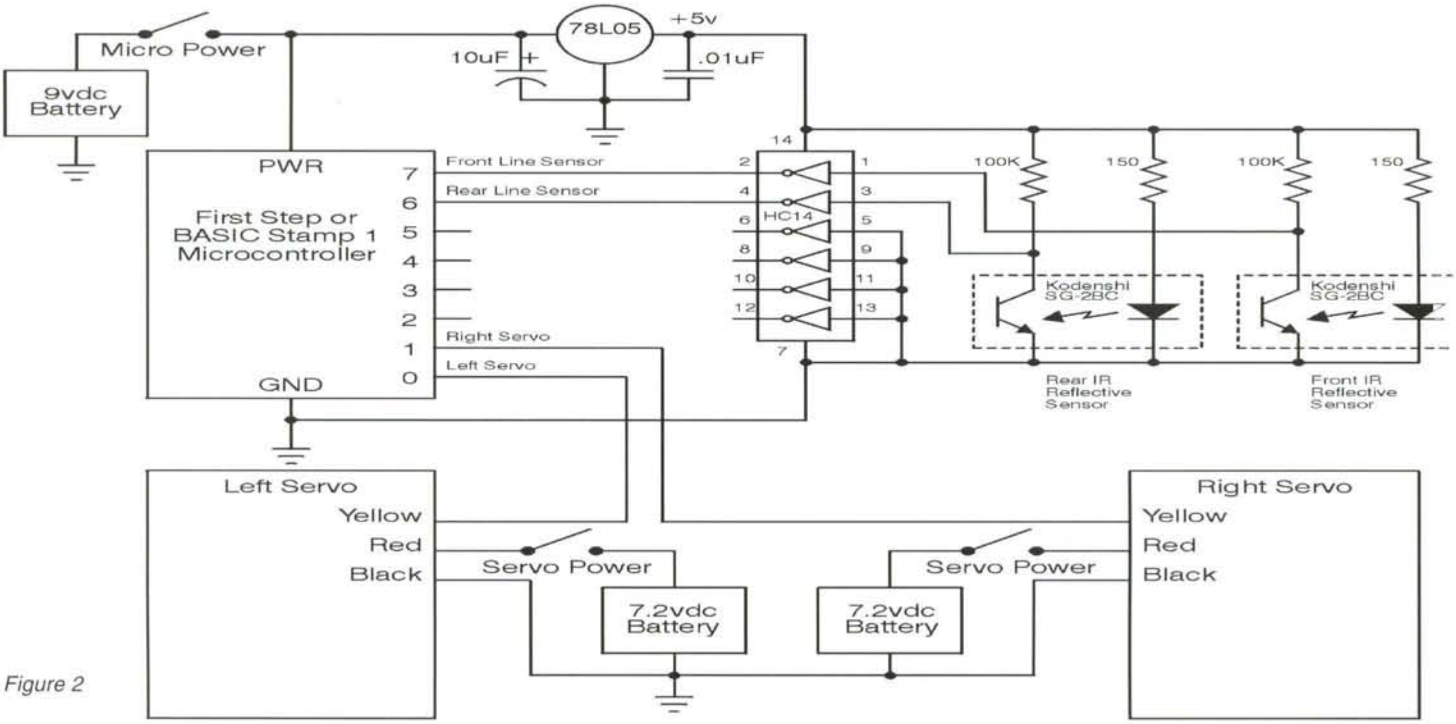

Micro Power
9vdc
Battery
78L05
+5v
10uF
.01uF
14
PWR
First Step or
BASIC Stamp 1
Microcontroller
GND
7
6
5
4
3
2
1
0
Front Line Sensor
Rear Line Sensor
Right Servo
Left Servo
HC14
100K
150
Kodenshi
SG-2BC
Rear IR
Reflective
Sensor
Front IR
Reflective
Sensor
Left Servo
Yellow
Red
Black
Servo Power
7.2vdc
Battery
Right Servo

Figure 2

Driving a DC Motor with an H-Bridge

by Tak Auyeung

My previous articles in Issue 6 (RS&T Feb/Mar) discussed pulse-width modulation (PWM) and driving a DC motor with a transistor. While these articles illustrate two fundamental concepts in controlling a DC motor, they do not address the control of direction. Most mobile robots require the control of motor speed as well as direction. This article discusses the concept and implementation of an H-bridge for directional DC motor drive. We will also take a close look at some H-bridge ICs and discuss the criteria for choosing an H-bridge. The last part of the article presents a prototype for use in experimenting with an H-bridge.

The Concept

Rotational direction of a DC motor can be controlled using the direction of current flowing through it. How can the direction of a direct current flowing through a DC motor be changed? Let us first look at the scenarios. Figure 1 illustrates the path of current flowing from the left-hand side of the motor to the right. Figure 2 illustrates the reversed path of current.

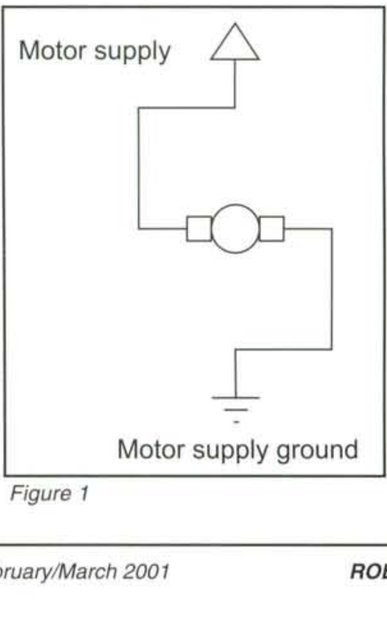


Figure 1

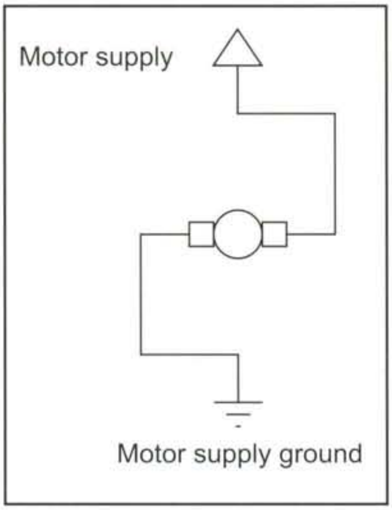


Figure 2

Figure 3 is a circuit of four switches that can control the flow of current as shown in Figures 1 and 2. When we close switches A and D and open switches B and C, the current flows from left to right through the motor. When we open switches A and B and close B and C, the current flows in the opposite direction. Consequently, the configuration in Figure 3 accomplishes the objective of controlling the direction of a DC motor. Indeed, the basic concept of an H-Bridge is this simple!
The inquisitive will ponder what the effects are for other switch configurations. Obviously, if switches A and C are both closed, the current goes directly from the positive supply to ground. Closing switches B and D has the same effects. These two combinations are not only useless, they could cause fires or even explosions! What happens if we close only switches C and D? This combination, while seemingly harmless, does have an effect.

Closing switches C and D causes braking of a rotating DC motor. When a DC motor rotates, the magnetic field of the permanent magnets generates electrical potential in the moving coil. This happens regardless of whether external electrical potential is applied. An open rotating DC motor is like any free rotor because the generated electrical potential is not pushing any current.

The motor will eventually stop due to friction. However, a shorted rotating DC motor (when switch C and D are closed) behaves differently.

The closed circuit allows the generated electrical potential to push current through the coils. As current flows through a coil, a magnetic field is developed. This developed magnetic field opposes the magnetic field of the permanent magnets in the motor. Consequently, the motor coils develop torque opposing the rotation of the motor. This torque helps to slow the motor. This technique of using generated current to brake a motor is sometimes known as passive braking.

A Relay Implementation

The implementation of an H-Bridge deserves some discussion. Figure 3 illustrates the positions of the switches with respect to nodes in the circuit, but it does not indicate the physical implementation of the switches. Figure 4 illustrates a double-pole double-throw (DPDT) switch (A and D on one side, and B and C on the other side) that makes a simple manual H-bridge. Of course, a computer cannot control such a manual switch. A computer, on the other hand, can control a relay with minimal additional hardware. How can we use the DPDT relay arrangement for automated control? Combining pulse-width modulation with a drive circuit using a single transistor, a computer can control both the speed and direction of a DC motor. Figure 5 illustrates such a circuit.

Indeed, such a design is possible and inexpensive. However, most relay coils require at least 40-mA at 5-V to actuate. This current requirement is not transistor transistor logic (TTL) com patible, and many microcontrollers and logic ICs cannot directly drive such a relay. A small transistor and a fly-back diode are often needed to interface a logic IC to a relay. For comparison purposes, a relay-based H-bridge that has a logic IC interface for PWM must have the following components:

- DPDT relay
- Small transistor to drive the relay
- Small flyback diode to protect the relay transistor
- Large transistor to control PWM
- Large flyback diode to protect the PWM transistor

Note that this design does not include the circuit to perform passive braking discussed later in this article. Passive braking requires an additional double-pole single-throw relay and its own transistor and flyback diode.

A Semiconductor Implementation

Alternatively, one can use semiconductor components to make an H-bridge. Bipolar junction transistors (BJTs) and field-effect transistors (FETs) can replace contacts in the relay to switch the current. The control voltage (pushing current through the relay coil) of a relay is isolated from the switched current. This is not the case for either BJTs or FETs.

Because switches A and B connect to the positive supply voltage, we call them the "high-side" switches. Similarly, we call switches C and D the "low-side" switches. Because the high-side switches connect to the positive supply voltage, designers use PNP BJTs or P-type MOSFETs to implement them. The low-side switches connect to ground, which makes it convenient to use NPN BJTs or N-type MOSFETs as components.

Given 5-V TTL logic control signals, it is easy to construct a NPN Darlington transistor or select an N-type MOSFET with a low on-threshold voltage. However, the high-side switches (PNP BJTs or P-type MOSFETs) require the control voltage to be between the supply voltage and some threshold offset below the supply voltage. This poses a problem for high-voltage motors (more than 5 or 6-V). If the supply voltage is 12-V and a PNP BJT is used, the control voltage (at the base) must be 12-V to turn off the BJT, and some offset below 12-V to turn on the BJT. A similar scenario occurs if a P-type MOSFET replaces the PNP BJT. The TTL logic control signal is between 0-V and 5-V, which does not meet the requirements.

Designers use two common techniques to overcome this problem. The first technique is to use a charge-pump to generate the high voltage required to control the high-side switches. The second technique uses another transistor to switch the motor supply voltage to control the high-side switches. Both techniques, unfortunately, increases the number of components significantly. Consequently, it requires many semiconductor components to implement a high voltage capable H-bridge with TTL logic interface.

Unlike relays, however, semiconductor components can be densely packed in ICs. Because H-bridges are very useful circuits, semiconductor manufacturers design and mass-produce H-bridge ICs. The L293D (available from Unitrode and SGS-Thompson) and SN754410 (equivalent to the L293D from Texas Instruments) are two cost-effective H-bridge ICs. Both are quad half-H bridges and cost less than $3 US, even in small quantities. Both of these ICs are BJT-based, which dissipate (waste) a bit of energy due to the voltage drop across the collectors and emitters. For higher efficiency at a higher cost, you can also use MOSFET-based H-bridges. Such H-bridges include the Vishay-Siliconix Si9987 and Harris HIP4020.

Even with an H-bridge IC, a motor con troller circuit sometimes requires fou additional diodes for back Electro-mag netic force (EMF) protection as well a passive braking. Even so, we only nee five small components to implemen one H-bridge. Certain H-bridge ICs such as the L293D (D for diode), hav internal diodes for protection agains the back EMF of an inductive load.

The internal diodes reduce the numbe of components significantly. Compare to the relay approach, H-bridge ICs ar cost-effective as well as compact fo current switching up to about 3-A ci cuit can use H-bridges that are con nected in parallel for additional switch ing capacity, but this approach require special considerations to synchroniz the parallel H-bridges.

For applications that switch more tha 10-A of current, the IC approach be comes infeasible. Designers often us parallel banks of discrete powe MOSFETs (in TO-220 packages) t implement H-bridges.

The relay approach also becomes rela tively cost-effective in such heavy-dut applications because P-type powe MOSFETs are expensive.

Experimenting with the L293D H-Bridge IC

The L293D H-bridge is a "quad half-H IC. Each half-H is also known as a push pull driver. Four (quad) half-Hs make two full-Hs. In other words, a singl L293D H-bridge IC can implement tw full-H bridges, driving two separate DC motors.

The L293D has one CE (chip enable) fo each pair of half-H circuits (physicall on the same side on the IC). The CE sig nal is active-high. In other words, whe the chip enable signal is high, the cor responding half-H circuits are active An active half-H circuit connects th output to either the motor supply volt age or the ground. An inactive half-H circuit does not connect the output to any signals. Each half-H circuit has a single control signal and a single output. The input signal controls whether the output connects to the motor supply voltage or to ground.

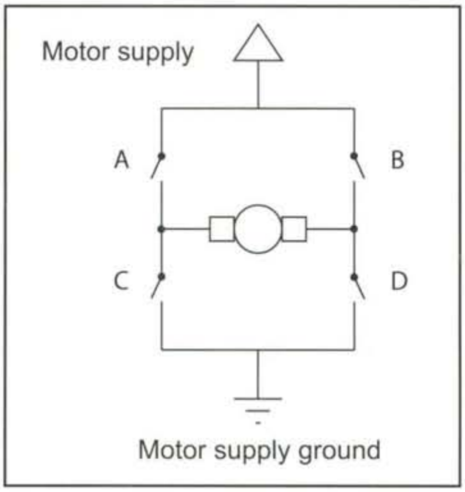


Figure 3

A high input signal connects the output to the motor supply voltage.

The following table indicates how to use the H-bridge to control a DC motor. 'H' means a high voltage (5-V) and 'L' means a low voltage (0-V) at the respective pins on the IC.

Chip enable	Input1	Input2	Meaning
H	L	L	Passive Braking
H	H	L	Driving in one direction
H	L	H	Driving in the other direction
L	X	X	Not driving nor braking

Table 1. Logic table for H-bridge

To experiment with the H-bridge, we will use normally open momentary switches to control the signals. The signals are "pulled" to a default state via 4.7-kΩ resistors. The CE (chip enable) signal connects to a 4.7-kΩ resistor, which then connects to ground (0-V). Because the CE pin is a high-impedance input pin, the voltage drop across the 4.7-kΩ resistor is insignificant, and the CE is driven low. Similarly, we connect each input pin via a 4.7-kΩ resistor to ground. With this set up, the H-bridge is disabled by default, an inherently safe state.

In order to change the voltage at the CE and input pins, each pin is connected to a normally open (NO) momentary single-pole-single-throw (SPST) switch, which then connects to 5-V. With this arrangement, when a switch is closed, the corresponding pin connects directly to 5-V and senses a high voltage. What about the 4.7-kΩ resistor that connects to ground? A current of 5-V/4.7-kΩ (approximately 1-mA) passes through the resistor. With this set up, the circuit does not enable the H-bridge to drive current by default. The operator must push the momentary switch in order to enable the H-bridge, minimizing chances of leaving the H-bridge enabled and heating up.

With such precautions in place, we can start to conduct experiments with the L293D.

Let all switches remain released. The motor should be easily rotated by external torque because only friction is slowing it down.

Let's try passive braking first.

Press the switch for the CE signal, but leave the other two switches released. This configures the H-bridge for passive braking by shorting out the terminals of the motor without connecting the external power. Therefore, the motor will not move by itself. Turning the motor with external physical force generates a current in the armature coil due to the coil moving through the magnetic field of the permanent magnets. This current in the armature generates a magnetic field that opposes the field in the motor's magnets. This results in a braking force, which decelerates the rotation faster than just friction forces.

Press the switch for the CE signal, then press one of the two input switches. The motor should turn one way or the other. As soon as you release the switch, the motor should start to slow. For passive braking, keep pressing the CE switch while releasing the input switch.

Press the CE switch, then press the input switch other than the one you chose in the previous experiment. The motor should turn in the opposite direction. You can also try active braking. Active braking applies external voltage to the motor to turn it in the opposite direction. Note that active braking can draw twice the stall current through the H-bridge into the motor! In other words, if your motor is rated at 0.5-A stalled, it will draw up to 1.0-A when you apply active braking. The L293D is rated for 0.6-A given sufficient heatsink. If the stall current of your motor is more than 0.5-A, you are advised not to perform the following experiment.

Press the CE switch and hold on to it, then press one of the other two input switches. As the motor turns in one direction, suddenly release the input switch and press the other input switch. The motor should decelerate very quickly, come to a stop, and then immediately start to rotate in the opposite direction. Doing this repetitively will heat up the H-bridge and potentially damage it. You should provide sufficient heatsink, and use a relatively small motor to preclude this occurrence.

Adding Pulse-width Modulation

At this point, the H-bridge can drive or brake a motor at 100% capacity. You can vary the speed of the motor simply by connecting the CE signal to a pulse-width-modulated signal (such as the output of timer M of the 555 PWM circuit, see Issue 6). It is advisable to connect the CE signal through a normally open momentary SPST switch to the output of timer M of the PWM circuit. In other words, you should redirect the original CE switch from 5-V to pin 3 of the timer M 555 IC. This ensures the H-bridge only drives when you press CE switch.

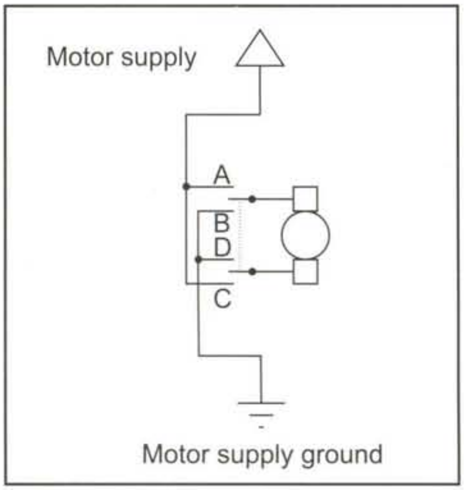


Figure 4

You can repeat the previous experiments while varying the duty cycle with the PWM circuit (by adjusting the variable resistor of timer M). Indeed, with PWM duty cycle control and an H-bridge for directional and braking control, you have a lot of flexibility driving the motors of a robot. Most of the circuits we have built so far are suitable for experiments and maybe a wired remotely controlled robot.

The Next Project

We have covered the basic concepts of driving a motor. However, we have not touched on the issues of controlling a DC motor. While the duty cycle is related to the speed of the motor, it is still an open loop. That is, we can only tell that a larger duty cycle results in a faster speed, but we cannot control the DC motor to rotate at a known speed within known tolerances.

DC motor control requires two additional components. The first component is a mechanism to ascertain the current speed of the motor. The second component is a method to adjust the duty cycle in order to make the motor rotate at a known speed. The next project in this series discusses the encoder, a component that can read back the rotation of a DC motor.

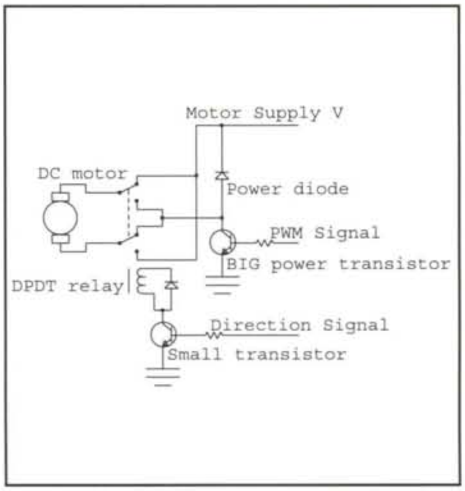


Figure 5

RS&T

Editor/Engineer/Educator Tak Auyeung, PhD, was the software development group leader for embedded controllers at Zworld before jumping ship to RS&T. He teaches the micromouse lab at UC, Davis, and is best known to our readers for his search algorithm series in RS&T issues 2 - 5.

FEEDBACK LOOP

"A new magazine, *Robot Science & Technology*, appears to be bridging the gap between hobby 'bots and research 'bots." – from *Introduction to AI Robotics*, by Robin Murphy (review to come in Issue 9).

Rescued by Mac MindStorms

I can't believe the people who dismiss Lego Mindstorms as just "a toy." Either they are mechanical geniuses who never need to modify their designs or they have access to a complete machine shop.

Lego is an ideal tool for trying out ideas, prototyping concepts and developing systems. If you create a claw hand and it doesn't work, you can repair or redesign it in a matter of minutes without the soldering, welding, cutting, gear box fabrication and wiring that makes redesigning an all-day job with conventional metal and wires.

I was unable to use MindStorms on my Mac until recently, thanks to Marc Jensen's letter in Issue 7. While trying out a terrain-conforming crawler idea, I was constantly modifying, altering and fine tuning the beast. As a result of Lego flexibility it operated better than I expected, and rebuilding it with metal was easy since I already worked everything out in a Lego Technics prototype. Now I'll be able to experiment with much more complex ideas and concepts with spur-of-the-moment alterations and quick fixes that make Lego worthy of consideration.

Keep up the good work, RS&T. You're doing a great job offering a wide range of information to a wide range of hobbyists, from the hard-core gear heads to the interested beginner. As a suggestion, I'd like to see a few how-to-build BEAM robots articles in the coming year along with your more robust robot building articles.

Paul Lenoue

Dear Paul, I'm glad we could help our Mac buddies. Roboticists (whether in theoretical AI research or hands-on robot builders), are all incredibly creative. And creative types, like artists, tend to use a lot of Macs. So we'll continue to increase our support for non-Wintel systems, while still serving the "mainstream" DOS posse.

Your interest in BEAM reflects a wide-spread and encouraging evolutionary leap of faith. Beamers are increasing in number, discovering that educators can teach math, science and biology with non-programmable bugs. We at RS&T continually ask for BEAM stories, but all the potential writers are too busy; they're building bots! BEAMers, unite and write!

Supply and Demand Rule Robotics

Thanks for your hard work. Roboticists across the country appreciate your efforts. I for one read RS&T cover to cover, every word.

Paul A. Jacobs

I am grateful for your response to my e-mail. I really enjoy the magazines and look forward to receiving the rest. I know that starting a business is difficult and I try to be patient. It's the only robot magazine out there and I really hope it will be around for a while.

Thomas Bock

Dear Paul and Thomas, letters like yours keep us going. We know the demand for robots and robot-related stuff comes from a huge, rapidly growing, affluent and educated market. Fortunately, suppliers and sellers are increasingly supplying our demand. 2001 is an exciting year to be in robotics.

Your care in supporting your subscribers and your wonderful magazine are a winning combination. I look forward to issue 8 and all the future issues to come.

Robert Martin

Here it is Robert, enjoy! Thanks for your compliment. We try to care for customers as best we can. Although, I must admit that we are a busy shop with very few people, so we sometimes lose track of things.

For folks who need to reach us, the fastest, most reliable way to get a response is to email service@robotmag.com. May all our future issues speedily reach your door.

Advancing Amateur Robotics Through Robot Contests

by John Piccirillo and Dave Everett

Mobile robotics is a fast growing area not only for both the professional and industrial researcher, but also for the hobbyist. New books, magazines, web sites, and college courses are added every year, and robots are receiving more media attention. While mobile robots have a long way to go before catching up with the movie versions, now is a good time to pursue this interesting and future technology. Often it is difficult for an amateur to decide what direction to pursue. One can see this clearly in newsgroups posts by self-declared newbies, where the enthusiasm is clearly out of synch with reality. What is needed is a focused challenge, a manageable task with opportunity for creative solutions, i.e. a well-designed contest.

Contests are particularly valuable for amateurs, who need the direction and association that these events provide. By the way, by amateur we mean someone with resource ceilings: limited funding, no support staff, limited tools and test equipment, and only discretionary time in which to work on projects. On the other hand, amateurs have certain advantages that are worth pointing out, namely, vision, enthusiasm, and freedom. Vision: many times the established workers in a field can't see the forest for the trees. This was certainly true of the personal computer revolution. Enthusiasm: early powered-flight development benefited greatly from the boundless enthusiasm of amateurs. Freedom: individuals employed by companies, the government, or universities don't set their own agenda, but amateurs can.

Why contests? Contests offer several worthwhile advantages and opportunities. Contests

- Focus on Integration – To satisfy the contest goal one must put together several technologies and a control scheme. This prevents one from getting stuck on just one aspect of a robot.
- Let one Compare Approaches – After working intensively on a solution for a specific problem, it's very instructive to see what approaches others have pursued.
- Involve Real Hardware vs. Simulations – Most contests force one to put up the real thing.
- Provide an Assessment of Current Limits – A broadly attended contest offers a good cross-section of the available technology and integration techniques.

In our view, not all contests are equally efficacious. There are several different types of competitions. The most popular involve:

- Radio-Controlled Machines
- Single-Behavior Performance
- Micromouse Maze Navigation
- Free-Form Rules
- Task-Oriented Objective

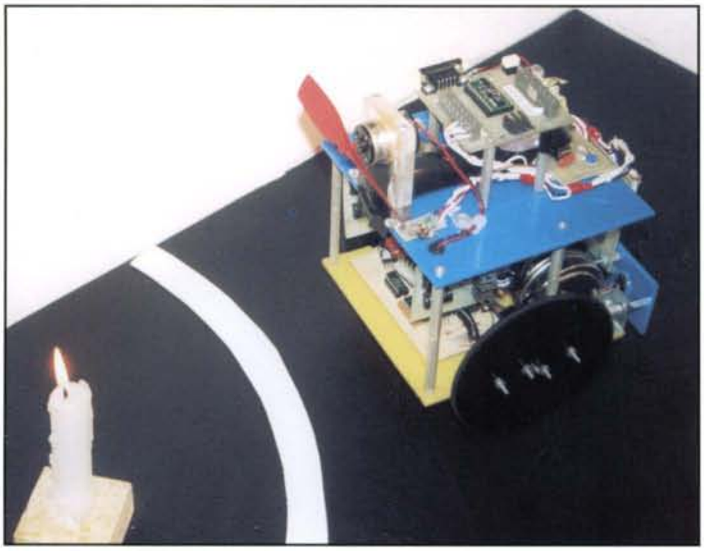
We argue that only the last category is useful for advancing the state of amateur robotics, the theme of this article. The first category, radio-controlled machines, is excluded as advancing anything because the intelligence is on the wrong side of the link. For our present purpose a robot is, a) autonomous, b) mobile, c) interacts actively with its environment, and d) has an embedded task as a goal. The second contest category includes things like line-following robots that exercise only one capability of a robot instead of a complex, task-achieving goal. Those contests are fine for entry level skills but don't advance the state-of-the-art. The same goes for micromouse competitions. Although they are challenging, they are best suited for stimulating interest in robotics, but are too artificial. Robots are allowed to look over the wall and to have multiple runs – a solution (maze solving algorithm) looking for a problem. Free form rule competitions allow one to do whatever one likes, which is certainly not bad, but doesn't set the bar high enough nor provide the focus necessary to promote amateur robotics to the next step.

Technology has advanced to where amateurs can begin to break into building more sophisticated robots, ones that can achieve scaled-down, real-world tasks in a natural environment – the next step. Before enumerating contest design criteria that promotes this next step, we point out what we consider misdirections, that is, robot adventures that seem cool but either don't advance next step skills or simply become black holes for time and money. Efforts we consider fruitless are:

- Building Anthropomorphic Robots
- Making a Computer-On-Wheels Robot
- Building Science Fiction Imitations
- Pursuing An Immature Technology
- Image Processing Vision System

There is nothing wrong with any of these approaches in the appropriate context, but they don't lead to viable robots. Without the underlying capabilities, an anthropomorphic robot is just a mannequin. As for science fiction imitations, the movie versions cheat anyway, using radio control and film special effects.

Sometimes one aspect of a robot can become so dominating that the robot never gets built. Vision is one of those tempting capabilities that we'd all like our robot to have, and it can in limited ways.



At the Trinity College Fire-Fighting Home Robot Contest in 1999, Ikea-Bot by Arun Patel of Ontario's Grand River Collegiate took 2d place, and was the only junior division robot to complete all three runs. Parallax BS2 brains on top, O-rings for traction, red DC motorized fan to extinguish the flame.

However, getting involved in milking the vision system to get high level performance will take all your energy and resources, leaving nothing left for the rest of your robot.

That leaves our pet peeve misdirection, the robot as a computer on wheels. Many amateurs consider the mechanical aspects of a robot as a bother to get over with in a hurry. This may be because many people come to robotics from computer programming. The robot as merely a computer on wheels has been previously termed the "software viagra" approach, the implicit view that software will make up for mechanical and sensor deficiencies. Cars don't fly and vacuum cleaners don't mow the grass – a robot's mechanical design should be custom made for its primary task.

So how do we move forward to fulfill our ambitions to build interesting robots and overcome our limitations as hobbyists? Here are a few suggestions:

• Join A Group – Isolation breeds stagnation. The group can be a local club or an Internet special interest group. Ask questions, answer questions, follow discussions, and visit web sites. Keep an open dialog with a peer group.

• Demonstrate Your Robot – Show your robot to a peer group. It may be a scout troop, a science class, or some friends. Self-evaluation isn't enough, and not nearly as much fun.

• Complete or scale back lingering projects. It's encouraging to have something to show for all your effort.

Don't let one component dominate. That laser scanner project getting out of hand? You may have to substitute another item or technique or get help from someone else.

Thankfully there are many enabling resources for hobbyists. Thanks to the Internet, one can locate and purchase a great variety of components; there are several worthwhile books on mobile robotics one can purchase; there's a new crop of easy-to-use single-board computers suitable for small robots; and there are some really good sensor modules at reasonable prices. As your skill increases and your desire to build more sophisticated robots becomes a reality you should consider these next design steps:

• Networked Microcontrollers – the advantage here is to build workable modules that are competent in themselves. A modification of the robot's capability should not mean having to completely re-do the control program or the construction.

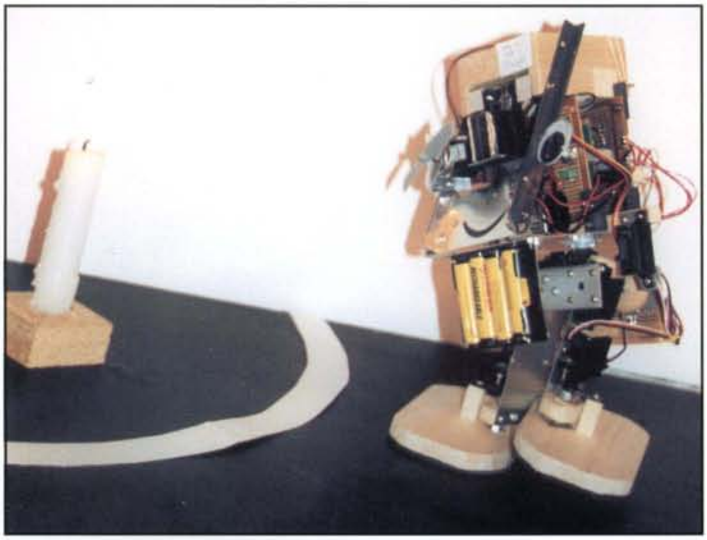
• Solid Mechanical Design – unfortunately this is still a do-it-yourself area, but don't hesitate to have local machine shops work on specialized parts.

• Behavior-Based Mobility Control – This is an outgrowth of the work originally done at MIT by Rodney Brooks and his students. There are many behavior-based variants. We strongly recommend reading his original papers (see page 54) and the excellent *Behavior-Based Robotics* (MIT Press, 1998, to be reviewed in the next issue) by Ronald Arkin.

We have advanced the thesis that robot competitions can advance amateur robots and we've given examples of how it does this. However, not all contests are equally effective in advancing the state of robotics. In fact, it's not easy to design a good competition. It is not our intention to stifle or limit the kinds of robots that are built or the types of robot competitions that are held. Rather we would like those so inclined to give thought to introducing competitions that have the goal of advancing amateur robotics. We offer these contest design criteria:

• Relax Constrained Environments – the contest should offer an operating arena that is in between that of real, dynamic environments and the completely artificial environment of uniform surfaces with high contrast divisions between floor, walls, and objects.

Patel, Ikea-Bot and teacher Carl Chcsanits at the Trinity College fire-fighting maze.



In 1999, Mark Whitney demonstrated Stampy, the wooden walking robot. In 2000, Mark returned with Mrs Stampy, having bigger feet and a faster Parallax BS2-SX, which completed the course and blew out the flame. Waddling along at 10 fpm, Mrs Stampy took 5 minutes to look into every room, backing up and turning around. "Just phenomenal," organizer Jake Mendelssohn said, giving Mark the award for Most Unique Design. Look for a vision system and two processors in 2001. Mark now works at Acroname, Inc.

• Use Multiple Success Criteria – options or bonus situations allow robot builders with different skill levels to enter the same contest, and allows competitors to push their creations.

• Design for Multiple Robots or Competitors on the Same Field – this adds real-world complexity and makes for a much more interesting contest.

• Move to Real World Situations – as a contest matures in years it should also evolve to accommodate products, techniques, and lessons learned from previous competitions. Contests that don't evolve become stagnant.

• Use a Meaningful Task for the Objective – tasks that are scaled from real-world situations are more interesting and allow for natural extensions as the contest matures and as the entries grow in capability.

Robot contests can be entertaining, instructive, and cha lenging. Currently there are not many challenging contes suitable for amateurs at all levels. Although no amateu contest meets all of the above criteria, the contest that be meets them is the Fire Fighting Home Robot Contest hel annually at Trinity College, CT. We hope that others wi sponsor challenging contests that allow us to improve ou robot building skills, and bring us together to enjoy and lear from each others creations.

RS&T

Dave Everett runs Deetron Services hardware/software consultancy in Sydn Australia. Dave has been involved in robotics for more than years. Dave was RS&T's official correspondent and photograp at the world championship robotic soccer games, RoboCup 200

Contributing editor John Piccirillo, Ph has degrees in physics and astronomy a has worked on missile defense systems for twenty years. He teac a senior design at the class University of Alabama in Huntsv in which engineering students design and build a mobile rob and he also consults for Army defense contractors. John's been co peting and writing about robot contests for five years. His detai articles are published at RobotMag.com and in RS&T since Premiere Issue. See his MiniFAQ for Beginners www.RobotMag.com.

The opinions expressed in these pages are not necessarily endors by the staff of Robot Science & Technology, except for this articl which I think is AOK. —Mike

Calendar of Events

5th Annual Atlanta Robot Rally
Feb 3, 2001 at SciTrek in Atlanta, Georgia
Home-brew robots from around the Southeast.
botlanta.org


See Clint Bridges' Spherion at SciTrek in Atlanta

Botball Regional Tournaments
Feb 24 – Apr 21, 2001; nine regionals around the USA
3-day teacher tutorial. Students design, build, and program in C. Robots compete on a 4 X 8 board.
kipr.org

FIRST Regional Competitions
March 1 – 24; thirteen regionals around the USA
April 5 – 7, 2001, National Championship, EPCOT, Orlando
usfirst.org

Acroname Robotics Expo
March 3, 2001 in Boulder, Colorado
University level. Also, presentations about NAS Robonaut and robot mission to Pluto.
aroname.com

8th Annual CANADA FIRST Robotic Games
March 1 - 3, 2001 in Toronto, Ontario
canada1st.org

Western Canadian Robot Games
April 6 - 7, 2001 in Calgary, Alberta
Southern Alberta Institute of Technology, Calgary
robotgames.com

15th Student Robotic Engineering Challenge
April 7 - 8, 2001 at Robert Morris College in Pittsburgh, Pennsylvania
Robotics International of the Society of Manufacturing Engineers.
17 events, 800 students. sme.org

9th Annual Northwest Robot Sumo Tournament
April 21, 2001 in the Edmunds Community College Gym, Lynnwood, WA
Autonomous and RC sumo, Japanese class and Mini-Sumo class.
sinerobotics.com

Trinity College Fire-Fighting Home Robot Contest
April 21 – 22, 2001 in West Hartford, Connecticut
Extinguish a candle in an 8 x 8 maze. A truly international event. Added divisions, bigger prizes, exhibits, activities, seminars. Video of previous year available.
trincoll.edu/events/robot/

Robothon 2K+1
April 28, 2001 in the Seattle Center
Seattle Robotics Society mazes, fire-fighting, sumo, plus lectures.
seattlerobotics.org

14th International FLAIRS Conference
May 21 - 23, 2001 in Key West, Florida
Special Track on Niche Autonomous Robots. How can mobile robots be brought into the real world? In cooperation with the American Association for Artificial Intelligence.
flairs.com

IEEE ICRA-2001, Frontiers of Robotics & Automation
May 21 – 26, 2001 Seoul, Korea
Sponsored by the IEEE Robotics & Automation Society
Including an International Advanced Robotics Program on the Technological Challenge of Dependable Robots in Human Environments
ircra2001.org

AUVSI's 28th Annual Symposium and Exhibition
July 31 - August 2, 2001 in Baltimore, Maryland
Association for Unmanned Vehicle Systems International
auvsi.org

PROPERTIES OF THERMOPLASTICS

Pick a Plastic for your Robot

by Conrad Hopkins

Plastics have changed our world almost as much as semiconductors have. And, like semi-conductors, they are impacting virtually every part of our lives, in most cases for the better. Plastics lend themselves to robot technology because they are easy to work with, light, strong, clear, colored, flexible, rigid, formable, and reasonably priced.

Though there are hundreds of plastics in existence, they fall into two basic categories: Thermoplastics, and Thermosets. This article will deal with thermoplastics.

The difference between thermosets and thermoplastics has to do with their basic properties.

Thermosets typically are purchased in liquid resin form and include products such as epoxy, polyester, and urethane. Like concrete, the user gets one chance to mold or shape the liquid form before it permanently hardens to the desired state. These products are often combined with fiber reinforcements to produce composites.

Thermoplastics are more like wax in that they can be melted and reshaped over and over. Thermoplastics are typically sold to the consumer in the form of sheets, rods, and tubes, which can be cut and formed for specific uses.

Knowing which plastic meets the properties needed for a specific project is often the first challenge a robot builder faces.

There is no such thing as a handy reference list that identifies which plastic is best for which robot part, though that would sure make things easy. For example, no single plastic is best for the outside body of a robot. As a builder I must ask, "Do I want the case to be clear so people can see my interior workmanship? Or, do I want it to be impact resistant to handle attacks from other robots? Or, does it need to filter UV light, or perhaps resist to chemicals?"

All these questions lead to different plastics. This article wi describe various properties of plastics and identify the pe formance of each plastic in relation to each property. Th will allow you to pick out the plastic that best meets you specific need. Since there are literally hundreds of plastic with more being created every day, this article will be lin ited to the most common and readily available plastics. E gineering and specialty plastics can be prohibitively expe sive and may not offer a significant advantage over a mo economical alternative. The plastics included here will be

- ABS
- Acrylic
- Impact modified acrylic
- Polycarbonate
- Styrene
- Polyethylene
- Delrin
- PVC

ABS

ABS is a blend of three different plastics into a sort of pla tic alloy. It is most readily available in opaque black an white, with one side smooth and one side textured. ABS available in thickness down to 0.040". ABS is hard, rigi and very tough. It can be glued and bent as well as vacuu formed. It is more economical than acrylic, but lacks th transparent color options. ABS is used for a variety of par such as football helmets, camper tops, housings for sma appliances, communications equipment, etc. It can b chrome plated for a completely convincing metal look. has a lower melting point than acrylic, so it lends itself heat forming and vacuum molding. Ideal for prototyping

ACRYLIC

Acrylic is used for everything from department store di plays to aircraft windows. Sometimes known as Plexigl (a trade name), acrylic is lighter, clearer, and more impa resistant than glass.

Transparency is one of the big advantages of plastic Though there are several transparent plastics, the greate clarity comes from acrylic. With a light transmission of 92% acrylic is clearer than glass. (A typical car windshield tran mits about 70%). This clarity, when combined with polishe edges, can produce a spectacular appearance. Frosted acryl sheet (also known as P-95) seems to glow when it is lit fro behind.

Since acrylic comes in a variety of shapes, there are endle possibilities to create special effects. Acrylic fiber optic cab is readily available and can be used to direct or 'pipe' ligh rays. Narrow diameter rod can be used to 'pipe' light, and if the sides of the rod have a sanded finish, the whole rod will glow with an 'other-worldly' appearance. Acrylic spheres (used to make the eyes in the movie *Bicentennial Man*) and cabochons (highly polished half-sphere beads) also create special effects. Florescent acrylic produces effects that defy description in print. It is a 'must have' for visual impact. Try engraving on it for an even more dazzling effect.

Another advantage of acrylic is its ability to resist the effects of UV. It is virtually unaffected by UV and is ideal for exterior applications. Speaking of UV, regular acrylic filters UV light up to about 360-nm. A special acrylic called OP-3 filters up to 98% of UV below 400-nm. This is ideal for protecting valuable objects from damaging UV light.

Besides being clearer than glass, acrylic has significant impact resistance. It can withstand impact many times greater than glass, and when it does break, the pieces are large and much less sharp than glass. The one down side of plastic is that it scratches. However, it is now possible to purchase acrylic that has been treated with an abrasion resistant coating. This coating can stand up to steel wool, and is thus ideal for high contact applications.

Acrylic is also available in a rainbow of colors, ranging from transparent to opaque. It can easily be machined, bent, and glued. More details on that are below.

POLYCARBONATE

Polycarbonate (also known by its trade name, Lexan) is readily available in clear, smoke, and white and in thickness down to 0.005-inches. It is an extremely impact resistant material that is designed for applications requiring durability and high service temperatures. While not quite as clear as acrylic (approx. 80-89% transmission), it offers superior impact properties. To illustrate this, TAP Plastics has a piece of 1/8-inch thick polycarbonate mounted in a frame that is taken around to various trade shows and demonstrations. People are invited to use a hammer to break the material. After three years of serious pounding, no one has managed to break it! So, if you are looking for a plastic that can take a beating, this is the choice.

Polycarbonate is also able to handle higher temperatures before softening and it has superior chemical resistance, especially to acids. Polycarbonate can be heat bent at a temperature of about 300°F. While glue joints are possible with polycarbonate, they are not as strong as acrylic joints. Thin gauge polycarbonate is commonly used for the body shells on radio controlled cars. In thin gauge, it is incredibly tough. Polycarbonate can be sawed, drilled, routed and tapped.

IMPACT MODIFIED ACRYLIC

Impact modified acrylic, also known as HP at TAP stores, offers the aesthetic advantages of acrylic and has impact properties approaching polycarbonate in a single product. It fabricates more easily than polycarbonate and produces better glue joints. The down side is that it is an interior-only product. It is rapidly affected by UV, causing it to haze quickly. It is also sensitive to heat, which will cause it to turn milky. Even with these limitations, HP offers a great combination of looks and performance.

STYRENE

This sheet material, most commonly available in white, also offers excellent impact resistance. Its chief advantage is its thermoformability. In thin gauge, it is ideal for vacuum forming or heat bending. It glues well, but does not have good chemical resistance, especially to aromatic and chlorinated solvents. The other big advantage of styrene is its price. It is approximately one-third less in cost than acrylic, and about half the price of polycarbonate. This is a versatile and economical product for experimentation.

POLYETHYLENE

Polyethylene is perhaps the largest volume manufactured thermoplastic. It is used for everything from sandwich bags to cutting boards to gallon milk bottles. It is characterized by toughness, outstanding chemical resistance, low coefficient of friction, and ease of machining. The most readily available versions of polyethylene are low density and high density. Low density is used for bags, packaging, and plastic sheeting often used for a moisture barrier. High density polyethylene (HDPE) is more durable and used for cutting boards and containers such as juice bottles. Polyethylene is generally a white translucent color and is readily available in thickness ranging from 0.060" to 1", in both smooth and textured finishes.

For robot applications, the big advantage of polyethylene is its easy machinability and impact resistance. It can be milled into low friction gears and bushings, which do not need to be lubricated and have high resistance to wear yet are light weight. It is the only one of the plastics mentioned here that will actually float in water.

There is one other polyethylene that is often mentioned, called UHMW, which stands for ultrahigh molecular weight polyethylene. This engineering plastic is costly and is seldom required for most applications. HDPE (high-density polyethylene mentioned above) will usually meet the UHMW applications.

PVC

The initials PVC stands for Polyvinylchloride. In its flexible form it is called vinyl and comes in a range of thicknesses and is very clear. PVC is most commonly known as the material plastic sprinkler pipe is made of. It is also available in sheet form. Smooth PVC sheet is glossy, comes in several thicknesses, and can be glued, formed, and machined like the other plastics. It has good chemical and impact resistance, but is one of the heavier plastics available. PVC sheet also comes in a textured foam version. It is very light, economical, and available in a variety of colors. Usually used as a backing for signs, it too can be bent, glued, and machined to form light weight parts. However, since it has a rigid cell structure, it will exhibit very different properties than its solid counterpart.

DELRIN

Delrin is a brand name for Acetal plastic. It is an engineering plastic that is commonly used for gears, window cranks, housings, seat-belt components, cams, and bushings. It has excellent toughness but is very machinable and has a low friction self-lubricating property. It is supplied in rod or sheet form, and in many ways is the most versatile and durable of the above plastics. It is available in black or natural (translucent white). One of Delrin's most outstanding characteristics is its ability to resist fatigue, even after repeated bending. Unlike metals which will eventually lose strength when repeatedly bent back and forth, Delrin seems almost indestructible. Delrin's properties remain virtually unchanged over a wide temperature range, even down to –40F. Machined parts are hard, smooth and glossy, creating a slippery feel and low coefficient of friction, even without a lubricant. Finally, Delrin has outstanding abrasion resistance, For making mechanical parts, it is probably the best choice

WORKING WITH PLASTICS

Plastic is in many ways easier to work with than wood. Be low are a few brief suggestions. Your local TAP Plastics stor has much more detailed information available.

BENDING

All of the above plastics can be bent with the use of heat Caution needs to be used, as these are combustible materi als, so heating in your home oven or with a torch is to b avoided. A heating element in a channel or nichrome wir can be used to make a line bender, which will heat the ma terial to about 300°F for bending.

POLISHING

Acrylic and HP can be easily edge-polished with the use o a torch. The easiest method is using MAPP gas. propan and oxygen in combination can also be used. Propane alon does not provide enough heat for a good finish. In the ab sence of a torch, a buffing wheel can also be used with polishing compound.

Polycarbonate cannot be flame polished and must be buffed The other plastics do not edge-polish very well.

CUTTING

Typical home shop tools can be used with modest success Very thin material (up to 0.125") can be scored or even cu with a matte knife and then either snapped or bent to fa tigue. Thicker materials should usually be cut with a saw.

A saber saw can be used, but it will leave a rough edge When using a saber saw, the biggest problem is vibration between the saw and the plastic. Laying the plastic on top of a two-inch thick piece of Styrofoam and holding the plasti down while cutting through both the Styrofoam and plasti can solve this. The Styrofoam provides a good solid base Another problem with a saber saw is that the blade heats up and melts the plastic. This can be minimized by blowing compressed air on the blade while cutting. The blade itsel should have about 10 teeth per inch and should be used to cut only plastic. A blade that has cut wood will be too dul for a good plastic cut.

A band saw works fairly well, but again compressed ai might help to minimize melting. The blade should have 14 teeth per inch. A band saw will not produce a good finished edge, but can be used for rough cutting.

A table saw is the best method for cutting plastic, but the proper blade should be used. A new plywood blade can do an adequate job. But for better edge quality, a carbide tipped blade with a triple-chip tooth design is best. A 10″ blade should have about 80 teeth.

DRILLING

All these plastics drill very easily. The secret is to use the proper bit. A regular twist drill bit will chip the plastic. There are economical drill bits available which will produce very clean cuts. Trying to use typical home bits will only produce frustration and wasted plastic. Drill speed should range from a slow 400-rpm for large holes (1″), up to 3500-rpm for 1/8″ holes. For larger holes a Forstner bit works well.

FASTENING

Mechanical fasteners work with all the above plastics. But because acrylic can expand or contract with temperature, do not use countersunk flathead screws. Use a standard round-head screw with a smooth shank. The guidelines for this process are extensive and are available in any TAP Plastics store. All of the plastics except polyethylene can be glued. The glues actually solvent-weld the parts so that in most cases the joint is as strong as the sides. Again, extensive details are available in any TAP store.

RS&T

Plastic FABRICATION

SHOPPING FOR PLASTIC

When shopping for plastic, there are some key questions to ask in order to get correct pricing, avoid unexpected delays, an make sure you are purchasing the correct product.

When requesting the price of a particular plastic over the phone, it is essential that you ask for the price of a specific siz piece. When someone quotes you a price "per square foot," you may not be getting the full picture. Some companies wil charge a four-square-foot minimum, regardless of how small a piece you need. Other companies have cutting charges an other costs which may not be quoted over the phone. So for accurate pricing, ask, "How much is the price for a such and suc plastic, 11" X 13" (or whatever your size is)." This should reveal any hidden charges that don't show up until you purchas the piece. If you have a shape other than a square or rectangle, this should be mentioned too, as that may incur special labo charges.

Ask about turn-around time. For simple shaped pieces (squares, rectangles, circles, etc.), the order should be done while yo wait, unless you need a very large quantity. "While you wait" should not mean 30 minutes or more. It should mean 5 to 1 minutes. Some industrial suppliers may have a great price, but you may grow old waiting for a small order. Obviously, i your piece(s) have holes to be drilled, edges to polish, etc., the waiting period should be a bit longer.

Finally, ask the sales person what he or she thinks is the best material for your project. A knowledgeable sales person can hel prevent problems and can often suggest more economical ways of doing your project.

As can be seen by this very brief introduction, the possibilities are endless. Plastics are versatile, economical, and practica This article just scratches the surface, so to speak. Any TAP Plastics store can provide you with extensive and practica information.

Below is a chart with some comparative information. The numbers for things like tensile strength are helpful in comparin one plastic with another. For example, if I wanted the stiffest material below, I would pick the highest flexural modulus which is acrylic.

	Acrylic	ABS	Poly	Hi-imp.	HP	HDPE	PVC	Delrin
Cost[1]	Med.	Med.	High	Low	High	Low	Med.	High
Clarity	Best	Opaque	Good	Opaque	Good	Opaque	Opaque[7]	Opaque
Impact Resistance	Good	Exc.	Exc.	Exc.	V.Good	Exc.	Exc.	Exc.
Specific Gravity[2]	1.19	1.05	1.20	1.05	1.11	0.96	1.3-1.6	1.41
Flex. Modulus[3]	480,000	300,000	340,000	280,000	440,000	220,000	3-800,000	410,000
Hardness[4]	Best	Fair	V.Good	Fair	Good	Fair	Fair	Good
Heat distortion[5]	195	190	270	190	195	170	140-170	230
Tensile Strength[6]	10,000	6,000	9,400	2,600	7,800	4,400	6-8000	8,800

1. Relative to others in list
2. Measure of relative weight. Waters specific gravity = 1g/cc. The higher the specific gravity, the more dense the material.
3. Measure of 'stiffness' measured in PSI
4. Resistance to scratching. Note: Acrylic and Polycarbonate are available in a highly scratch resistant finish.
5. Temperature at which material begins to deflect under a load, measured in degrees Fahrenheit.
6. The strength needed to pull a piece apart as if pulling on the ends. Measured in PSI.
7. Flexible vinyl sheet is available in clear.

GET STARTED IN ROBOTIC VISION

BY MICHAEL GASPERI

If your robot can't see, consider an affordable entry-level vision recognition system that you can buy off the shelf. Robot Science & Technology asked Extreme MindStormer, engineer and author Michael Gasperi to introduce RS&T readers to Lego's.

MINDSTORMS VISION COMMAND

Vision Command is the latest add-on for Lego's popular Robotics Invention System. It combines a PC based color video camera with interactive software to enable your robotic creations to see. The package includes a special Lego camera, CD-ROM, 140 building elements and instructions for several projects. It retails for about $100.

You don't need to own a Robotics Invention System to use Vision Command, because it can be used standalone. Its recognition software can be used to trigger your PC to produce a variety of sounds, capture still images, and even record videos in reaction to programmed events. The software provided lets you use it as a regular digital camera to take pictures, create animations, make time-lapse movies, and record videos up to 640x480 resolution. The package comes with enough parts to build several camera mounts and adjustable stands.

The Lego camera is really a Logitech QuickCam that has been repackaged into a special translucent brick. It has regular Lego studs on the top and bottom as well as Technic style mounting holes on the sides. These make integrating the camera into your Lego creations easy. It has a built-in microphone for recording sound, a push button for taking still photos, and a ring for manually adjusting the focus. It also includes a 15-foot (5-Meter) USB cable that is permanently attached. Because the cable is rather stiff, tracked vehicles are recommended for pulling it around.

Software installation takes about 10 minutes from start to finish. Virtually all the documentation for the product is on the CD-ROM. The introductory video and tutorials are excellent, requiring over two hours to work through all of them. The tutorials depend on assembling some projects, with their step-by-step instructions, are provided in the printed documentation or Constructopedia.

After you learn how to use Vision Command there are six challenge exercises to master.

The recognition software works fairly simply. The incoming video image is divided into a predefined pattern of regions. There are 20 predefined patterns that can have up to 8 regions. In each of the regions you select a condition that will trigger an event. The conditions include movement, light level, or the presence of a color. When an event is triggered, a stack of instructions executes to make sounds, capture images or send motor commands to your RCX. There are over 25 different instructions you can put into the stack.

Programming is reminiscent of stripped down RCX Code. There are no variables, subroutines, or conditional statements. Everything executes inline with the exception of a loop type instruction. Once an event is triggered no other processing occurs until that stack of instructions completes. There are provisions for a single background activity like simply scanning the camera back and forth. Even with these limitations, some interesting behaviors, like motion tracking, can be easily programmed.

Vision Command interacts with your RCX through the IR tower. It can send commands to the RCX one-at-a-time to turn on motors, set power levels or play sounds. There is even a little control panel built into the Vision Command interface that lets you drive your robot around without any programming at all.

When running a Vision Command program, a small control program is downloaded to the RCX that contains all the instructions for each event. From then on it only sends a single number to indicate which event has occurred. The sensor inputs on the RCX are not used by Vision Command and the RCX can't be easily programmed independently. This seriously limits the complexity of projects you can build with it.

Although the hacker community hasn't jumped on Vision Command like they did with the RCX, there are a few people discovering ways to work around its apparent shortcomings. LUGNET has an interest group dedicated to Vision Command at http://news.lugnet.com/robotics/vc/. Also, because the camera is actually a Logitech at heart, some people are downloading the Logitech Software Development Kit from http://developer.logitech.com/sdk/ . This allows them to interface the camera using other computer languages.

Vision Command is recommended for ages 12+ years old and as such it represents an excellent introduction to robot vision. You are not going to be able to create an autonomous house-cleaning robot with it, but Vision Command does demonstrate the basic principles without becoming overly complex. Just being able to hitch a ride on a Lego robot and operate it by remote control is an interesting experience all by itself. It comes with a useful set of building elements and an added bonus is the software that allows you to use the Lego camera for general imaging.

Minimum System Requirements:

- USB Port
- Windows 98 (Windows ME is expected to work, but was not confirmed as of the time of this writing)
- Pentium II 233 MHz processor or higher
- 32MB RAM, 64 MB RAM recommended
- 200 MB available hard disk space
- Windows compatible mouse
- Windows compatible sound device
- A DirectX 6.1 compatible sound card
- 4MB (8MB recommended) DirectX 6.1 or higher compatible graphics card
- 4x CD-ROM drive, 8x recommended
- 800 x 600 SVGA Display in High Color (16-bit) mode

Michael Gasperi works as a principa engineer with the Advanced Technolog division of Rockwell Automation and is a expert in building custom sensors for Leg MindStorms. He is a co-author of th new book "Extreme MINDSTORMS An Advanced Guide to LEG MINDSTORMS" from Apress.

His Web page: www.plazaearth.com/usr/gasperi/lego.htm is th central depository for homebrew RCX sensor information.

FEEDFORWARD: SELECTED REVIEWS

Principia Robotica

a book review by Mike Greene

Cambrian Intelligence: The Early History of the New AI

by Rodney A. Brooks, PhD

Published by MIT Press, Cambridge, Massachusetts

"I wish to build completely autonomous mobile agents that co-exist in the world with humans, and are seen by those humans as intelligent beings in their own right."

Deciding that computationally intensive artificial intelligence research was an unneeded and self-deluding direction for robot builders, Rod Brooks invented a new and controversial approach, sometimes called subsumption architecture or behavior-based robotics. *Cambrian Intelligence* is the world's most authoritative book on the creation and evolution of his new AI.

Eagerly anticipated by our readers, this is a series of complementary scientific papers linking Brooks' personal philosophy to behavior-based principles, to the actual construction of intelligent robots. *Cambrian Intelligence* generously informs, forcefully persuades and masterfully educates.

The technical first section is well illustrated, propping up those of us who stumble over big words. The philosophical second section is easier to read and intellectually entertaining.

For a quick start, read chapters 4, 5 and 6 before tackling the more difficult sections.

Brooks elucidates the requirements, constraints, challenges, and principles that enabled his lab to turnout physically grounded robots as diverse as Squirt (50 grams) and the six-legged Genghis (now at the Smithsonian Air and Space Museum).

Brooks explicitly says what works and what doesn't: "Traditional systems... will never be transportable to the real worlds that the situated behavior-based robots already inhabit." He acknowledges that studies of human processes provide useful insights, but insists our machines should not try to imitate humans completely. Why build something that already exists? Besides, much of what humans think and do is not optimal by any measure!

Those of us who enjoy a fast-paced and tenacious quarrel will enjoy Chapter 8, Brooks' 1991 "Intelligence Without Reason." Here, Brooks exposes weaknesses in every approach to AI, and often depicts "successful" outcomes as artificially predetermined or just plain unscientific. His sharp critiques provide powerful lessons for students and hobbyists alike. The detailed reports of the sinners' actions imply precautions for all robot builders.

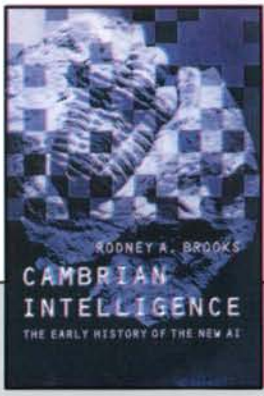
Brooks fuses each chapter together at the front and back and in between, showing the relations between his ideas and others. His thoughts are expressed clearly, not hiding behind the dry academic smoke that so often blocks readers from seeing inside the ivory towers and ivy walls of science.

But there are challenges for the reader. We need a glossary and an index. Thankfully, Brooks uses easily concretized abstractions so most of us can keep up. And because the papers span eight years of research, readers benefit by watching ideas evolve. Thus, the text uses solid teaching techniques: repetition, analogy, moving from known to the new. It is accessible to those intelligent enough to want to read it. Brooks spares us from Greek gobbledygook, even though "to this day many people still find the contents of this paper entirely disreputable because it is not filled with mathematical equations." Mostly, he relies on *existence* proofs. Test your stuff in the real world, he says.

He argues well. He compares th worldviews of scientists to theists, atheist and agnostics. He includes two Fables c Fortran and a Parable of Artificial Flight t speed our conversion to his philosophy.

Finally, Brooks contrasts his behavior-base robotics to other approaches, and lists wha subsumption architecture is not. Whil subsumption involves many finite stat machines closely linking sensors to actua tors, it isn't connectionism. It's not neura nets. It certainly isn't heuristic search algo rithms. In some ways, planning gets in th way of effective action.

Our readers will enjoy having the entire te of Brooks' early major papers. His persona introductions and footnotes, combined wit concise abstracts and bulleted conclusion leave reader satisfied, armed with well-sup ported and useful knowledge, and fille with fresh perspectives to view intelligenc ourselves, and our machines.



Cambrian Intelligence:
The Early History of the New AI
by Rodney Allen Brooks
MIT Press, Cambridge, Massachusetts
1999; ISBN 0-262-02468-3 (hc) or 0-262-52263-2 (pbk)
187pp, extensive bibliography

Quick Start Guide for the Imagination

a book review by Mike Greene

Dave Baum's Definitive Guide to Lego Mindstorms

by Dave Baum Published by Apress

Dave Baum's Definitive Guide to Lego MindStorms is more than an introduction to a toy robot system of programmable bricks, it is an inviting, encouraging, easy-to-read, fact-filled guide to two languages and 800 colorful bricks.

This book amply illustrates that while Lego bricks have made building small robots quite an easy task, the ease of construction broadens the challenge of your potential designs. Fortunately, the Definitive Guide presents 19 discrete chapters with quickly completed goals. Here are 14 robots to build, exercises in practical and conceptual problems every designer faces. How to lengthen battery-life, how to strengthen rigid links, increase work space around a joint. How to augment firmware.

Use the Definitive Guide as a handy reference while actually building bots. It's written by a Motorola engineer who graduated from MIT, the birthplace of the programmable brick. Baum, holding a degree in mathematics and applied computer science, places only two equations in the whole book. Frankly, even hobby robot designers need more math, not less.

Every concept seems to come in two's, inextricably coupled like the magnetic poles. For every problem, a solution. For every hardware solution, a software counterpart. For every learning challenge, at least one exercise. The book presents two programming environments, the graphical environment Lego provided, and a traditional text-based freeware language called NQC. Not Quite C, written by Baum.

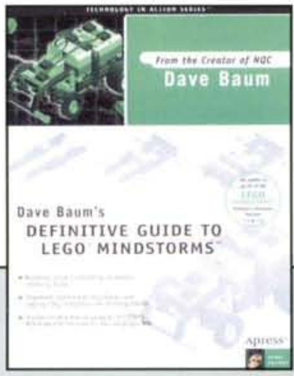
Two levels of readers can learn from this book. Newbies will find simple explanations that experienced roboticists will skip over: the definition of firmware, how to figure a gear ratio, a diagram illustrating rack and pinion steering, an illustration that demystifies a differential gearbox. Gearheads will find improved designs that maneuver gracefully, escape corners effectively, and use data logging.

Baum's writing strikes an efficient balance between reader-friendly and data-filled. The wide-open typesetting and picture-per-page layout makes information leap off the page. Skip over or delve into debouncing, hysteresis, argument passing or ratchet splitting. I wish more publishers would follow Apress' example.

Baum successfully addresses two diverse audiences; the code-makes-my-eyes-glaze-over mechanically inclined gearheads and the dangerous-with-a-screwdriver software geeks. Each community finds "common" knowledge, and each discovers insights into the other world. Rarely would a single author hope to approach two audiences at once, but Baum succeeds easily. He couldn't have done this without skilled illustration by Rodd Zurcher. Zurcher's crisp 3D renderings clarify complexity. The companion CD has PDF documentation, ready to run programs and useful AV clips.

It doesn't make sense that in the first sentence and in the last line is the word 'toy.' Whether to use that diminutive term should be left to the discretion of the experimenters, roboticists, and educators who use this book.

Dave Baum's Definitive Guide to Lego MindStorms is more than a description of practical hardware and software concepts, it is a quick start guide for your imagination.



Dave Baum's Definitive Guide to Lego Mindstorms
by Dave Baum
Apress
6400 Hollis St., Ste. 9, Emeryville, CA 94608
2000; ISBN 1-893115-097, pbk, 379 pp with 5 appendices, index, and CD

FeedForward: Selected Reviews

Everyone's BASIC Cookbook

a book review by Gene Burbeck

The Microcontroller Application Cookbook

by Matt Gilliland
Published by Woodglen Press, California

The Microcontroller Application Cookbook by Matt Gilliland does an excellent job of explaining in detail a fundamental aspect of robot design: using microcontrollers to enable your robot to respond to changing conditions in the real world

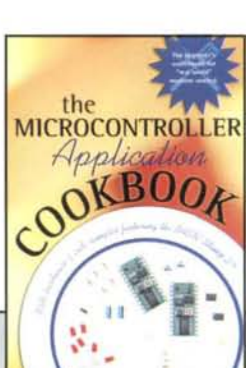
Unlike many other treatments of robot microcontroller input and output, this book leaves theory behind and gets down to the practicalities of hooking up input and output circuits. Circuits are presented in a simple, straightforward manner; the book assumes little prior knowledge of the subject.

The Microcontroller Application Cookbook would also be quite useful to people who are further along in robotics. It is a good review and adds to one's understanding of how microcontrollers should and shouldn't be hooked up. The book often confirms what was learned through experience, and will speed the learning process for those in their first several years of robotics.

For someone with his or her first BASIC Stamp, this book is a must-read. This is especially true if he or she plans to use a BASIC Stamp to control a robot. The *Cookbook* could give a beginner a great head start on learning to use microcontrollers, teaching how to hook up input and output circuits right the first time.

Nevertheless, the *Microcontroller Application Cookbook* is not directed only at BASIC Stamp users. It explains ways of hooking up input and output devices so they can be applied to any microcontroller. The BASIC Stamp II is used for examples because it is a common choice for beginners and circuits that are compatible with it are compatible with most other micros. The *Cookbook* clearly states when a circuit is specific to the BASIC Stamp II and tells how to make it work with microcontrollers that have different input/output characteristics.

The book is arranged into five chapters. It opens with an introduction, followed by a basic description of the dos and don'ts of microcontroller hookup. The meat of the book describes specific circuits categorized into a chapter for input and a chapter for output circuits. The final chapter shows ways of getting many inputs and many outputs with just a few pins and doing analog input and output. The appendix is a useful list of sources for parts and information.



The Microcontroller Application Cookbook
by Matt Gilliland
Woodglen Press, California
2000; ISBN 0-615-11552-7, (pbk)
Available now at www.parallaxinc.com
Soon in major bookstores

The input chapter describes how to i terface a number of devices includi switches, phototransistors, temperatu sensors, pots and microphones, to nan just a few.

The output chapter covers controlli things with a microcontroller. LED output buffering using transistor buffer chips, and MOSFETs, motor co trol using an H-bridge, solid state r lays for controlling AC devices, ar controlling hobby servos with har ware are some of the output circui covered.

The pages are nicely laid out so tl book doesn't look intimidating. Tl text is broken up by schematics ar code snippets, making it pleasant read.

There are some formulas, but nothir daunting. *The Microcontroller Applic tion Cookbook* is clearly organized; it easy to find what you're looking for. has a table of contents and an index th helps one quickly find topics of inte est.

The Microcontroller Application Cookbo is a great book for beginner and inte mediate robot builders alike. Beginne can use this book as a guide for hool ing up input and output devices. Inte mediate builders can learn better way of hooking up circuits and solidify the understanding of things they hav learned through trial and error.

I highly recommend the book for thos needing solid, easy to understand ir formation about trouble-free circuits fc interfacing most input and output de vices that robot builders would fin useful.

FEEDFORWARD: SELECTED REVIEWS

Brainwaves in a Time Capsule

a book review by Tak Auyeung

Build Your Own Robot!

by Karl Lunt
Published by A K PETERS, LTD.

Long-time readers of Robot Science & Technology remember Karl Lunt's important contribution to starting our little magazine back when it was still a rambling robot website with a meaningless name.

Then, when RS&T evolved into the printed magazine you hold in your hands, Karl was gracious enough to contribute to our first three issues with his practical experience, prompting internet rumors that he had "jumped ship" from his regular column at another magazine. In fact, he was, by popular demand, working on a book.

Build Your Own Robot! is based on his favorite columns. Although the Acknowledgement section indicates the articles were converted for book form, they retain the distinct taste of column articles. For instance, references to articles from previous issues are scattered throughout. Additionally, many articles contain multiple subjects. Karl's high-bandwidth brainwaves move rapidly from subject to subject.

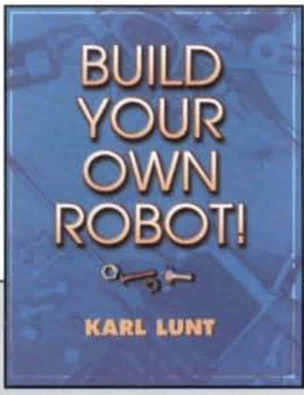
Karl's book is really packed with information and ideas. There are many projects in power electronics, microcontroller and digital electronics, batteries, mechanical design, software tools and software development techniques. An experienced reader will find this information useful and insightful. However, a beginner with little background in electronics, software and physics may find the articles difficult to digest.

Build Your Own Robot! is very "hands-on," with little treatment of theories and principles. It's not a textbook of neatly arranged and developed concepts. Despite an attempt to organize articles by categories, the book's structure is far from that needed for a beginner to read from cover to cover. As such, this book is more suitable for advanced readers who already understand the basics of mechanical design, electronics and software.

Most advanced readers will browse through this book easily to pick up new ideas, read about some new technologies and learn new techniques.

This eagerly anticipated book is indeed fun to read and packed with good ideas. Those two facts make any book a good buy.

We at *Robot Science & Technology* recommend the book for also a third, more unusual reason: because Karl was, for years, the unofficial king of amateur robot-builders, *Build Your Own Robot!* is a time capsule to be treasured by long-time Karl Lunt fans.



Build Your Own Robot!
by Karl Lunt
A K PETERS, LTD. Publishers of Science and Technology
63 South Ave., Natick, MA 01760
2000; ISBN 1-56881-102-0, paperback; 592 pp; $34.00
Available from: *akpeters.com*

Introducing: Mi Amigo y La Internet

Remember Carnegie Mellon's Xavier? This trash-can-sized robot would roam hallways snapping photos of unsuspecting profs. Internet visitors commanded Xavier to snoop around and send snapshots through email.

Now you can have the same ability, and more, in your home, school or office, using a smart new machine from long-time robot maker ActivMedia. Dr Kurt Konolige from SRI International came to *Robot Science & Technology* to demonstrate his shiny red machine, its interface and programming software.

AmigoBot uses a high-speed 16-bit microcontroller, eight Polaroid sonars, and two 12 Vdc Pittman motors with a 19.5:1 gear ratio and built-in hi-resolution quadrature encoders. It's light-weight, bigger than a bread box, and scoots along on fat rubber tires at about 2 meters per second.

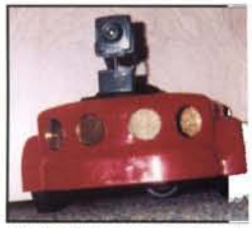
AmigoBot avoids obstacles, patrols pre-set search patterns, and tours hallways using a pre-built map of your building. You can, of course, simply point and click to any destination you like; but it avoids walls and walking people faster than a human tele-operator could.

Using the popular Saphira Robot Control Operating System, AmigoBot continually calculates the "least cost path" to its destination, using the most current knowledge of its environment, which is updated 10 times each second. This allows AmigoBot to rapidly change directions.

Kurt, who teaches at Stanford, says the AmigoBot uses a probabilistic "wavefront" algorithm to localize itself. It assigns probabilities to every possible locus, then uses sensors to award probability points to each locus that seems to match its current worldview. You can track real-time results on your computer, as some red dots disappear and others coalesce into blobs, indicating high-probability locations. Within seconds, there is only one big red blob on the screen: AmigoBot is pretty sure it has found itself.

AmigoBot showed us four ready-to-run behaviors: Wander, Avoid, Forward and StallRecover, which is executed whenever something rams the robot. The interface looks easy to learn, although our prototype didn't yet run on Windows or Mac. Instead, ActivMedia chose a more stable software platform preferred by developers: Linux.

To program your own behaviors into your AmigoBot, you would use the interactive (near-real-time), interpreted C-like language called Colbert. This language is optimized to allow synchronous activities for real-time control of robots, and it allows Internet access. Programmers familiar with C++ or Java should be able to learn Colbert easily. Colbert lets you call functions written in other languages, and its interpreted interactive environment facilitates debugging your programmed beha iors without needing to re-compile your new code. This an open architecture that allows users to program beha iors in a common language, then mix-and-match progra calls, all on the fly. ActivMedia's AmigoColbert interfa allows the user to customize behaviors simply by changi text files.



AmigoBot comes in the do Video cam on top, Gold Polaroid sonars surround body.

Stanford's Kurt Konelige is having fun programming ActivMedia's AmigoBot via Innomedia's radio link, but Parallax prez Chip Gracey looks a little concerned about having a robot roaming around his building.

Definitely the best looking robot ActivMedia has ev fielded, its plastic shell is attractively curved. The inside spacious and sturdy, with plenty of room for add-on boar and accessory mounts. Our model used a reliable, off-th shelf 418 MHz data radio from Innomedia (who, quite coi cidently, also advertises in this issue).

We tested the prototype AmigoBot in a beautiful new glas brick and metal facility owned by Basic Stamp maker Pa allax. Our test areas were a flat carpeted floor between c fices and a polished wood basketball court that is (belie this) the office of Parallax president Chip Gracey.

AmigoBot and its big brother Pioneer roamed through c bicles stuffed with computers, devilish pranksters and te equipment. When RS&T's own Dayne VanPelt sneaked u to block the Pioneer's path with a trash basket, the unfla pable bot swiftly recalculated its new path around the u expected obstacle.

Output? The video relayed from AmigoBot's on-board col camera to our monitor provided a clear, colorful cat's-e view of the path ahead. You can spy on co-workers, vie intruders in your warehouse, or keep a virtual eye on t kids through the Internet. I expect hobbyists will tinker, ad telescope-mounted m crophones, speaker smoke and heat dete tors, lasers and mo cameras. Hey, let's ad speech recognition an computer-generate voices! Finally we hav a true surrogate to inte act with burglars an in-laws. RS&T

RS&T's Dayne VanPelt sneaks up on an ActivMedia Pioneer with laser rangefinde on top. It was no problem for this robot to get around the trash can that suddenly appeared.

Robot Kits

Spider Walking IR Avoider Robot #3-299 $69.95

Futuristic & maneuverable 6-legged walker with IR for object avoidance. Dual motor drive, front mounted sensor. 13 cm L x 13 W x 10 H. No soldering required, needs 2 AA & one 9V cell.

Cybug Robot Kit #3-466 $49.95

Cybug has touch & light sensors to exhibit active & interesting behaviors. Printed circuit board & 2 DC motors! 12 cm L x 10 W x 4 H. Soldering and one 9V cell required. Machine or living organism?

Cybug Predator/Prey Add-on #3-501 $32.95

Build & attach the predator & prey circuits to two of your Cybugs & then let them roam wild! The Predator will sense the I.R. signal from the Prey & hunt it down!

Cybug Hunger Instinct Add-on #3-503 $29.95

The hunger circuit board lets your Cybug sense its battery level and seek light when "hungry". Includes notes for making a recharging station (requires other parts) so your Cybug can recharge itself!

SolarFly Robot Kit #3-529 $59.95

Powered by the Sun! The SolarFly continually seeks the brightest object it sees. Twin feelers help navigate around obstacles. It even has short-term memory about objects it contacts! The long vertical antennae can draw energy from other sources such as the Sunflower Power Plant or your feeding stations described in the kit. Assembles with basic tools, requires soldering. Add a swarm to your CYBUG farm!

LEGO® MindStorms™

LEGO® Robotics Invention System V1.5* #3-699 $199.99

New version 1.5 Explore your creativity with the RIS, featuring the revolutionary RCX computer brain with 3 inputs and 3 outputs. Includes 2 motors, 2 touch sensors, 1 light sensor, infrared transceiver, cables, software (Win 95 / 98) and detailed English instructions for creating your own Lego robots! Over 700 pieces!

MindStorms RCX "Brain" Brick	#3-560	$119.99
MindStorms RCX IR Remote Control	#3-546	$24.99
MindStorms Motor 3-pack	#3-552	$49.99
MindStorms Touch Sensor	#3-347	$17.99
MindStorms Light Sensor	#3-329	$29.99
MindStorms Rotation Sensor	#3-544	$24.99
MindStorms Temperature Sensor	#3-516	$29.99

LEGO Robotics Discovery System* #3-521 $149.99

Bring your Lego creations to life! Scout computer is fully programmable from the on-board LCD command center. Create different behaviors with the touch of a button - no external programming from a computer required! With built-in light sensor, two inputs, two outputs, LED feedback lights, and IR communications plus 2 motors, 2 touch sensors, manual and over 380 LEGO pieces.

LEGO Droid Development Kit* #3-589 $99.99

The Droid Developer Kit is your introduction to robot building. Make your own R2-D2, battle droids and other robots from Star Wars, then experiment with different configurations to create your own droid series. Includes the Micro-Scout Lego computer with built-in light sensor, built-in motor and seven built-in behaviors plus over 650 Lego pieces.

MONDO-TRONICS ROBOT STORE

WAO G Programmable Robot #3-477 $84.95

A new edition of a sophisticated, full function robot kit. 2 motors, programmable "fuzzy logic" brain, 26 button keypad, sensor arms for collision or edge detection. Moves in all directions, performs if-then decisions based on sensor inputs and "fuzzy" membership rules, even do-loops! Requires 3 AA & one 9V battery. No soldering required. Great intro to robot programming.

Descartes Robot Kit #3-710 $229.00

Descartes is based on a dual processor architecture. Features four true 8-bit analog to digital (A/D) converters, four CdS light sensors, LEDs, shaft encoders, thermistor, touch switches and a speaker. Demonstrate nocturnal/diurnal behavior, phototropism/photophobia, non-contact object detection, motion detection, dead reckoning and more! ***Programmable from your PC.***

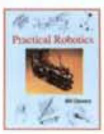
Rug Warrior Pro Deluxe Kit #3-406 $595.00

A complete, serious research robot platform from MIT. With all hardware, C software, 68HC11 CPU, memory, sensors, wheels, motors, chassis, instructions & more. 18 cm dia. The ideal companion kit to the Mobile Robots book (3-613, below). *Mac & PC compatible!*

Mobile Robots 2nd Ed. #3-613 $39.95

From MIT, a complete guide to designing & building robots: hardware, software, sensors, motors, "subsumption" software. 336 pages, illust. A must for all roboticists.

Practical Robotics #3-506 $39.95



If you ask "Where do I start?" the answer is "HERE!" Bill Davies shares years of robotics experience, covering all aspects: tools, techniques, components, sensors, soldering, etc... 330 pages with a fine index.

Build Your Own Underwater Robot #3-418 $29.95

Dive in to underwater robotics! The authors provide dozens of hands-on experiments using readily available parts. 148 pages. Great illustrations and a lot of fun!

Toll Free Order Line

800-374-5764

Mondo-tronics Inc.

PMB-N 4286 Redwood Hwy. Dept 167
San Rafael, CA 94903

Questions: 415-491-4600
Fax: 415-491-4696
Email: info@mondo.com

Send check, MO or credit card info. All orders add shipping:

Subtotal	
$25	$6.00
$25 to $50	$7.00
$50 to $75	$9.00
$75 to $100	$10.00
$100 to $250	$12.00
Over $250	$20.00

Calif. orders also add 7.25% sales tax

www.RobotStore.com

International Orders Welcome! See website for details.

BASIC Stamp®

Stamp One Starter Pack #3-444 $99.00

A complete starter set: Stamp Software, manual, cable, a Stamp 1 (see right), and a BS-1 Carrier Board with prototyping area & connector for the programming cable. A great value!

Stamp Two Starter Pack #3-439 $148.00

Everything needed to move with the Stamp 2: All of the items in the Stamp One Starter but with a Stamp 2 and BS-2 Carrier instead. Monstrous power!

Stamp Two SX Starter Pack #3-647 $159.00

New! Powered by Scenix microchip. Runs at 50 MHz (2.5 times faster than the standard Stamp 2), and has eight 2KB program spaces. A powerful brain for your next robot!

Mini SSC 2 & Servo Bonus Pack #3-436 $129.95

Control up to 8 servos via one serial line. Talk to it from any PC, Mac, Stamp, etc... Plus, daisy chain up to 32 together to control up to 255 servos. Includes six Hitec-300B servos!

Robot Legs, Robot Arms

5 Axis Robotic Arm Kit #3-366 $195.00

Fast, repeatable, and controlled from your computer serial port! Can lift over 160 grams. Control swivel, shoulder, elbow, wrist, and gripper. Mini SSC 2 board & DOS software included!

Hexapod Walker Kit #3-434 $150.00

Three servos provide full directional walking. Program in Basic from your PC. All hardware, electronics, software & instructions included. Over 20 cm long. Requires soldering.

Quadrapod Walker Kit #3-670 $295.00

Walks forward, backwards and turns left or right. Includes precut body and leg panels, mechanical hardware, 8 servos and assembly manual. Add your own servo controller (Mini SSC, ASC-16, etc.), microcomputer and sensors for full autonomous operation.

Mega Hexapod Walker Kit #3-423 $375.00

A heavy duty walking platform – over 33 cm (13 in) long! 12 servos give full control & major strength. Add your own servo controller (Mini SSC, ASC-16, etc...), a microcomputer (Stamp2, BOTBoard, etc.) & sensors for advanced projects!

Muscle Wires®

Muscle Wires Book & Deluxe Kit #3-168 $59.95

Create direct linear action with Muscle Wires. They contract up to 5% when powered! Use them in robots, models and anything needing strong all-electric *motion*. 128 page book, full plans for BORIS, a motorless walking robot & 14 others. Includes one meter each 50, 100 & 150 µm dia Muscle Wires. 128 pgs, illust. *Get moving today!*

Muscle Wires Project Book only #3-133 $17.95

Get the book now, order wires later!

Space Wings Kit #3-001 $19.95

Sleek silver wings flap using only 6 cm of Muscle Wire! Variable speed up to 36 flaps/minute. 15 cm tall. With circuit board, all parts & instructions. Requires soldering & 2 AA cells.

An ideal introduction to Muscle Wires.

Robot Kits

5 Axis Arm Kit — *5AA-KT* **$195.00**
- Preassembled electronics.
- Easy to assemble mechanics.
- Basic controllable, uses the SSC.

3 Axis Arm Kit — *3AA-KT* **$155.00**
- Preassembled electronics.
- Simplified Design.
- Basic controllable, uses the SSC.

Big Grip Kit — *BG-KT* **$75.00**
- Mechanics and servos only.
- Lifts up to one pound.
- 3" Gripper opening.

Mobile 5 Axis Arm — *M5AA-KT* **$250.00**
- Jumpstart your advanced experiments.
- Make a remote piloted arm.
- Includes the SSC servo controller.

Mobile 3 Axis Arm — *M3AA-KT* **$210.00**
- Jumpstart your advanced experiments.
- Make a remote piloted arm.
- Includes the SSC servo controller.

Mobile Big Grip Kit — *MBG-KT* **$174.00**
- Jumpstart your advanced experiments.
- Make a remote piloted arm.
- Includes the SSC servo controller.

Hexapod 2 Kit — *H2-KT* **$375.00**
- Large 12" x 12" walking robot.
- 3.5" of ground clearance.
- Walks with several different gaits.

Quadrapod Kit — *QP-KT* **$295.00**
- Large 7" x 12" walking robot.
- 3.5" of ground clearance.
- Walks with several different gaits.

Hexapod Walker Kit — *H1-KT* **$150.00**
- Simple 3 servo walking robot.
- Add IRPD for autonomous operation.
- Includes the First/Next Step Micro.

Remote Piloted H1 — *RPH-KT* **$210.00**
- Robot walks under remote control.
- Includes pan / tilt for CCD camera.
- Includes the First Step Micro.

Carpet Rover II Explorer — *CR2E-KT* **$225.00**
- Next Step equipped.
- Includes all sensors and nine projects.
- 8" x 8" acrylic available in five colors.

NEW!

Carpet Rover II Kit — *CR2-KT* **$115.00**
- Next Step equipped.
- Add IRPD for autonomous operation.
- 8" x 8" acrylic base, in five colors.

NEW!

CR Boe-bot Retrofit Kit — *CRBB-KT* **$45.00**
- Add your Boe-bot electronics.
- Improves performance and appearance.
- 8" x 8" acrylic available in five colors.

Carpet Rover I Kit — *CR1-KT* **$105.00**
- First Step equipped.
- Add IRPD for autonomous operation.
- 6" x 6" acrylic available in five colors.

NEW!

Rugged Terrain Platform — *TRP-KT* **$155.00**
- Next Step equipped.
- Add IRPD for autonomous operation.
- 5.5" x 9" acrylic available in five colors.

Lynxmotion
Got Robots?

Lynxmotion, Inc.
PO Box 818
Pekin, IL 61555-0818
www.lynxmotion.com

Tel: 309-382-181
Fax: 309-382-125
sales@lynxmotion.cor
tech@lynxmotion.cor

Electronics

First Step Micro Kit — *FS-01* **$40.0**
- Robotics microcontroller (Stamp1).
- Easy to assemble circuit board.
- Programmable in Parallax Basic.

Next Step Micro Kit — *NS-01* **$40.0**
- Robotics microcontroller. **BS2-01 $49.0**
- Uses the BASIC Stamp 2. **BS2E-01 $54.0**
- 2 LEDs and Push Buttons. **BSSX-01 $59.0**

SSC Servo Controller — *SSC-01* **$44.0**
- Preassembled electronics.
- Control up to 8 servos per SSC.
- Addressable as 0-7 or 8-15.

NEW!

12 Ch. Servo Controller — *SSC-12* **$75.0**
- Preassembled electronics.
- Control up to 12 servos.
- Now with variable speed!

IR Proximity Detector Kit — *IRPD-01* **$30.0**
- No contact obstacle detection.
- First/Next Step compatible.
- Adjustable range.

Line Tracker Kit — *TRA-01* **$20.0**
- Reliable line tracking made easy.
- First/Next Step compatible.
- Sumo robot edge detector!

Dual H-Bridge Kit — *DHB-01* **$30.0**
- Speed and direction is controllable.
- First/Next Step Compatible.
- 2 amps per channel at 12 volts.

NEW!

H-Bridge — *HBR-01* **$15.0**
- Powerfull single H-bridge circuit.
- Easy to control with one I/O pin.
- Control a DC motor as a modified servo.

Serial LCD Module — *SLM-01* **$49.0**
- Serial LCD Display.
- First Step / Basic stamp compatible.
- Bright LED backlit supertwist display.

HS-300 6 Pack — *S6P-01* **$80.0**
- Powerfull, compact and durable.
- Easy to control with one I/O pin.
- Many other servos to choose from.

Battery Charger — *SBB-01* **$46.0**
- Peak detecting quick charger.
- Auto stop and auto trickle charge.
- Selectable 2 or 4 amp charge rate.

Nicad Battery Pack — *BAT-01* **$20.0**
- Powerful 7.2 volt 1500mAh output.
- Quick connect plug.
- Powers H2-KT for 1 hour continuos.


JAMECO®
Since 1974
ELECTRONIC COMPONENTS
COMPUTER PRODUCTS
1•800•831•4242
Free Shipping on Your First Order! Just Mention VIP# RT2
Call for details – Certain restrictions apply
www.jameco.com/rst
Click on this URL to receive your free shipping and special product pricing!
Full-Service Electronics Distribution