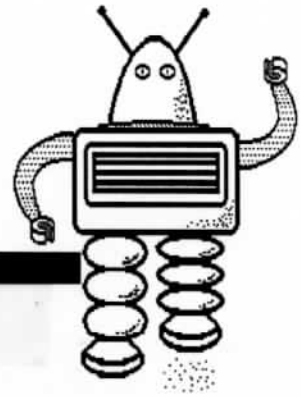

THE ROBOT COMPANION

The Newsletter of the Dallas Personal Robotics Group
July, 1989
Stan Spielbusch, Editor

JULY MEETING AGENDA

The July meeting will be held on July 8th, 2:00 PM at the Dallas Infomart.

Club business and other non-robotics discussions will be kept to a minimum this time.

Brian Vaceluke will give a demo of his latest version of the Home Navigation program.

Stan Spielbusch will give the demo that was rained out last month, which is the optical leash program mentioned later in this issue.

Stan Spielbusch will discuss the maze project which he hopes to get going by August. Anyone interested in homebrew robotics should attend!

JUNE MEETING MINUTES

by David Ratcliff

Ed Rivers gave a recap of the Bill Gates speech in May and the Computer Council meeting.

Stan Spielbusch gave a brief explanation of an "optical leash" program for the HERO 2000, which involves using the head light sensor to locate a light source and follow it. The demo was postponed due to rain.

Dwayne Good and David Ratcliff are building X-Y tables, which can be used for automatic lathes, routers, and plasma cutters.

General discussion on expected and actual past, present, and future military/industrial/consumer user of robotics.

Suggested possibilities of meeting field trips to replace users labs. Suggested trips included the University of Texas at Arlington robotics center, Texas Instruments robotics center, and an eat-in at Showbiz Pizza to see their 'robots'.

Discussed moving the meeting time back to 1:30. This will be voted on at the July meeting.

NOTES FROM THE EDITOR

by Stan Spielbusch

Well, quite a bit has been happening this past month! I've been contacted by several people regarding robotics projects, which is a very exciting sign! The general consensus is that we should continue to put effort into our present projects, as well as come up with more projects. There ARE interested people out there, it's just that we don't hear from them very often! I'm very pleased that I have a good selection of material for this newsletter, and I hope the trend continues. I apologize to anyone who didn't get their article or information printed this month, but I'm saving some in case next month is slower.

NEW PROJECT

Here it comes again -- maze mania! This may seem trivial to the people only interested in "useful" robotics, but believe me, it's not only useful, but more difficult than you might think. A maze-running algorithm is the first big step to autonomous navigation, and designing a robot from scratch is no easy task. Anyway, here's what I want to do, and I hope to find other people willing to participate.

Ed has mentioned that we could have a free table in the Infomart vendor area in the basement. Well, if we choose a table along the side, we could also use the vast area behind it to set up a maze. We could be running robots through the maze (or any other robot activity) in the morning during other meetings. I think this would be an excellent way to give our group exposure.

In addition to exposure, it would hopefully trigger robot projects. I intend to use only homebrew robots for the maze, which will also allow the maze to be smaller (we can put a limit on the robot size). I also think it's a lot more exciting to see 'spare parts' running around a maze than it is to see a 'cute, expensive toy' do it. Several other clubs around the world have an annual maze contest, and they attract a lot of attention. So why not us?

To start things off, I'm in the process of gearing up my homebrew robot (no pun intended) for maze-running, and I should have it working by the August meeting. I know at least two other members who are also planning to participate. Further encouragement came from Karl Lunt, who is designing a maze-running robot with his friends, and may offer the design as a kit-form robot (see his article in this issue). I want to hear from anyone else who would like to participate. Remember, the robot does not have to be a multi-function intelligent robot. It could be a simple motorized base with wall-following sensors and logic. More details later.

BULLETIN

Karl Lunt will be giving a presentation of his maze robot project at the Rocky Mountain SOG. This is a 3-day gathering which includes outdoor activities such as rafting along with computer conferences. It will be held in Gunnison, Colorado from July 27th to July 29th. For more information, call (303) 641-6438.

PRESIDENT'S CORNER

by Ed Rivers

Things are slowly returning to normal. Several members have expressed interest in a group project, and a call from Phoenix may serve to spark interest in maze running robots. If we can come up with a good demo, we can get space in the InfoMart vendor area. This allows a high volume of spectators without allowing them to walk all over the demo.

The July issue of PC Computing has a robot-related article on page 175. The article is titled: "All the Right Moves - Part cop, part maid, all computer: it's RoboPC. PC-powered robots try their luck in the home." The article mentions Heath and the Hero robots.

At last month's meeting, the topic of meeting time came up again. Several members expressed interest in meeting earlier in the afternoon, while one member wished for a later meeting time. At this month's meeting I'd like to hold a vote on moving our meeting time back to 1:30pm.

A MAZE-RUNNING ROBOT PROJECT

by Karl Lunt

I had wanted to build a hobby robot before, but the mechanical details and system design always stopped me. A software engineer by trade, the programming didn't present an obstacle; it was always the nuts and bolts stuff that held me back. Until now. If you don't have all the abilities you need to finish a task, team up with others that do. That is how Norbert Bukowski, Loren Heiny and I all got together to build an autonomous maze-running robot.

Norbert owns Bukowski Robotics, an industrial robotics design shop in Scottsdale, Arizona. Loren develops robotics vision and sensory systems (and has built one or two machines of his own). And I specialize in imbedded assembly language controller software for the Motorola 68xx chips. Between the three of us, we surely had the talent to design and build a little ol' maze-runner!

We got together at Norbert's shop one weekend to hammer out the general concepts of the robot. Each contributed ideas and thoughts from his own background, but the decision to go a certain way had to be agreed upon by all. We laid out the following design goals at that first meeting:

- The robot measures about 11 inches across (height currently unknown), having a round metal base supported by two passive casters and two opposing driven wheels.
- Drive force comes from two independently controlled stepper motors. The steppers are geared down by 60:1 to increase the torque at the wheels.
- The initial sensory system consists of a skirt of bumper switches to detect contact with a wall. World orientation is provided by a Radio Shack flux-gate compass.
- Software runs on a self-contained 68HC11 development board. The board carries 16K of EPROM, 8K of static RAM, 8 channels of 8-bit A/D, 16 I/O lines and a sophisticated counter-timer system.
- The robot must be able to find its own way through a maze built of 2x4's. The boards will be placed on edge so the top of the maze stands about 3.5 inches from the floor. No communication or power transfer between the machine and the outside world will be permitted.

My first task involved writing the stepper motor control software. The stepper interface uses four TTL signals plus ground. MCLK provides a stream of pulses to the drive logic; each high-to-low transition will cause a motor (if enabled) to move one step in the selected direction. DIRA and DIRB each select the rotation direction of stepper A and B. If a direction line is low, that motor (if enabled) moves counter-clockwise; if high, that motor moves clockwise. Finally, MTR* is an active-low signal that enables both motors.

To turn the motors, my software had to perform several functions in a particular sequence. First, the MCLK pulses had to be maintained; I used an interrupt service routine for this. Second, the direction lines had to be monitored and updated as necessary. Third, the pulse rate had to be varied smoothly from 100 to 2000 pulses per second, permitting a ramp-up of the motor speed (the jury is still out on the need to ramp down). Finally, toggling the MTR* line needed to be carefully synchronized with the pulse stream.

It took me several late-night sessions, but by the next weekend I had a first cut of the controller software ready to test. Norbert, Loren, myself and Peter Brown, a top-notch technician, met back at Norbert's shop for the first run of the software. Peter and Norbert hooked up the stepper hardware, I loaded in the controller software and issued a few commands. The thing worked!

By the time you read this, we hope to be testing the robot via an RS-232 umbilical. It should be able to explore a maze and send back information on the map it builds of its world. Hopefully, the machine will be ready for a demo at the Rocky Mountain SOG (a technical conference sponsored by Micro Cornucopia magazine) at the end of July.

We are interested in robotics developments in the DPRG; please write or call us about what you are doing so we can exchange information and ideas. If you find articles such as this interesting, let me know and I will keep you posted on our progress.

Best of luck, Karl Lunt , 4730 W. Menadota Dr., Glendale, AZ 85308 (602) 582-5863 home.

HOME NAVIGATION FOR THE HERO 2000

by Brian Vaceluke

Android Operating System Version 1.0

To use the Android Operating System, you will need a HERO 2000 with a disk drive and at least 128K of memory.

The Android Operating System is a Data Base interpreter that allows the user to schedule the HERO 2000 to remain idle (sleep) until a predetermined time, wake up, navigate to a predetermined room within a house, perform a certain behavior or set of behaviors, (possibly going to other rooms and perform other behaviors as directed by the user's Data Base) and finally remain idle until the next scheduled time.

This operating system has attributes of both Walter and Bev Bryant's Home Navigation program and Joe Roe's Continuous Consciousness program.

There are three parts to the Android Operating System Data Base. Each one is a separate text file that reads like a programing language. There are instructions, parameters, and even comments. The Data Base interpreter does not differentiate between upper and lower case.

The first and lowest level is a description of the house where the HERO 2000 will perform its tasks. It is called the Path file, and it has a default extension ".PTH" .

The second and mid level part to the Data Base is the Behavior file. It has a default extension ".BHV" . This file is a collection of programed behaviors that can be associated with a task name. These behaviors may draw from the path database for "Goto" instructions. These behaviors are similar to subroutines.

The third and highest level of the Data Base is the Schedule file, and it has a default extension ".SCH" . This file draws upon the first two files such that at a predetermined time the HERO 2000 can be instructed to navigate into a certain room via the Path Data Base and perform a certain task or behavior from the Behavior Data Base. This level of data base is the easiest to write in.

This Data Base modularity allows the HERO 2000 to run the same program in any environment provided that only the Path Data Base is changed. Or adding new complex behaviors to the HERO 2000's vocabulary is as easy as updating the Behavior Data Base. Or making a new schedule does not require a new set of Behavior's or Instructions on how to navigate the same house.

New combinations of programs can be can be created by mixing and matching different Path, Behavior, and Schedule Data Bases at initialization of the Android Operating System.

Instructions

There are two levels of instructions. The lowest level can be used by all three Data Bases while the higher level can only be used by the Behavior and Schedule Data Bases. All instructions have exactly one parameter, which follows the instruction name, and it is separated by one or more spaces. This is the instruction parameter pair, designated <ip> in this manual. (ALL instruction parameter pairs are separated by one or more spaces.)

The lowest level of instructions, which may be used by any data base are:

Absolute movement instructions:

LEFT <angle>	Turns the robot left from present position by <angle> degrees
RIGHT <angle>	Turns the robot right from present position by <angle> degrees.

WALK <distance Moves the robot forward <distance> feet if <distance> is greater than zero, or backward <distance> feet if <distance> is less than zero.

Relative movement instructions:

WALKFROM <distance> Moves the robot to a distance <distance> feet from the wall behind the robot.

WALKTO <distance> Moves the robot to a wall in front of it leaving <distance> feet between the wall and the robot.

LEFTPARALLEL <dist> Moves the robot <dist> feet; however, a distance reading is taken to the left of the robot prior to moving and also after moving. An angle relative to the robot and the assumed wall to the left of the robot is calculated. The robot is turned so thats its base is parallel to the wall, if it wasn't to begin with.

RIGHTPARALLEL <dist> Moves the robot <dist> feet; however, a distance reading is taken to the right of the robot prior to moving and also after moving. An angle relative to the robot and the assumed wall to the right of the robot is calculated. The robot is turned so thats its base is parallel to the wall, if it wasn't to begin with.

The higher level of instructions, which can only be used by the Behavior and Schedule data bases are:

GOTO <node> Moves the robot from its present node location to <node> along a logical path which it searches out from intermediate paths described in the Path data base. The physical path is a result of substituting low level movement commands as specified in the intermediate path descriptions in the Path data base.

TASK <behavior> At run time, substitutes a series of low level commands and/or GOTO <node> commands for the TASK <behavior> command as specified in the Behavior data base (like a subroutine.)

The Path Data Base

The Path data base is a series of PATH statements. A PATH statement is as follows:

PATH FROM: <node1> TO: <node2> IS: <ip>

or

PATH FROM: <node1> TO: <node2> IS{ <ip1> <ip2> ... <ipN> }

The series of PATH statements in the Path data base is ended by a THAT'S_ALL statement.

When a GOTO <node> command is encountered in the Behavior or Schedule data bases, the search for a logical path from its present node to <node> will be conducted from a tree described by the <node1> <node2> pairs in all of the PATH statements. The physical path will be the sum of the <ip>'s substituted for each individual partial path in each appropriate PATH statement.

Since the resultant path is one constructed from partial paths, it best that the user keep a reference direction in mind for each node when building a Path data base. The data base should be defined such that the robot always ends up facing ONE direction when it arrives at a particular node, no matter which node it came from. This way, when the robot passes through a node, it will always be facing the same direction when it leaves that node, since all movement commands are relative their starting position. If the robot is passing through a node on its way to a <node> in a GOTO <node> command, it will not make unnecessary turns when it arrives at and leaves from the node that it is passing through because there is a turn cancellation routine built into the GOTO <node> command.

Behavior data base

The Behavior data base is a series of BEHAVIOR statements. A BEHAVIOR statement is as follows:

BEHAVIOR <behavior> IS: <ip>

or

BEHAVIOR <behavior> IS{ <ip1> <ip2> ... <ipN> }

The series of BEHAVIOR statements in the Behavior data base is ended by a THAT'S_ALL statement.

When a TASK <behavior> command is encountered in the Path or Behavior data bases the associated <ip>'s will be executed. Since the TASK command is allowed in the Behavior data base, complicated behaviors can be built from several less complicated behaviors.

Schedule data base

The Schedule data base is a series of SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, or SATURDAY statements as illustrated:

SUNDAY 10:00 AM DO: <ip>

or

SATURDAY 12:00 PM DO{ <ip1> <ip2> ... <ipN> }

The series of statements in the Schedule data base is ended by a THAT'S_ALL statement.

When the Android Operating System is executing, the robot will sleep until the next weekly scheduled event. The associated <ip>'s for that event will then be executed, then the robot will again sleep until the next scheduled event.

FROM THE LIBRARY

by Stan Spielbusch

Sonar Mapper

Loren Heiny has submitted a nice program that will run on a PC, and needs no robotics. What it does is display a graphic map of sonar readings, which are read from a file. The readings may be generated by another program or simply typed in. The program is very flexible, and source code is included (Turbo C). This would make a good PC-end of a robot-to-PC link, which I talked about in a previous newsletter. To order this program, ask for the "Sonar mapper" disk.

Optical Leash

I didn't get to demo it at the last meeting, but I'll put it in the library anyway. This program is for the HERO 2000, and uses the head light sensor to locate a point light source. It filters out slight light-level differences in the room, and only responds to a bright point of light. It then takes a sonar reading and approaches the light in 2-foot steps, taking new light and sonar readings every 2 feet.

The intended use of this program is for guiding the robot around with a flashlight. I have tried it in my home, and it's easier and more fun than using the keypad to direct it. It probably won't work in sunlight unless you have a portable arc-lamp, but it should work well inside any building.

This program is on the HERO 2000 disk.

If you have a program to submit, put it on an MS-DOS format disk (double sided, double-density standard format) and bring it to the meeting or send to:

Stan Spielbusch, 2404 Via Barcelona, Carrollton, TX 75006

****** Please ***** include a description of the program, either as comments in the program or as a separate .DOC file. I don't have the time to study each program to figure out what it does!

When you submit a disk, you receive credit for 1 disk in return. Let us know which one(s) you want, or if you just want your original disk back.

We currently have 5 disks in the library -- a HERO-1 BASIC disk, a HERO-2000 BASIC disk, a HERO-1 Assembler disk, Loren Heiny's EyeSight program, and Loren's Sonar Mapper.

If you want a copy of a disk, the best way is to bring a blank, formatted PC-DOS/MS-DOS disk to the meeting and trade with me there. If you forget to bring a disk, we will have to collect $2.00 per disk. Mail-order -- $3.00 per disk -- no need to include a disk with order. Send orders to Stan (address above).