

**Overview:**

Early in the development of the Motor Mind C Solutions Cubed asked Paul Garcia, a member of the Robotics Society of Southern California (RSSC), to do some beta testing of the product. What follows are a brief biography for Paul, and the results of his testing. Paul used the Motor Mind C to add versatility to the club robot for RSSC. The document following this overview was written by Paul, and Solutions Cubed would like to extend our thanks to him for evaluating and documenting his work with our Motor Mind C product.

Paul Garcia has been building models and electromechanical gadgets for the last 30 years, but has only taken up hobby robotics since September 2001 when he joined the Robotics Society of Southern California (RSSC). As a Boeing Project Engineer, he is primarily responsible for directing the Engineering efforts on large commercial aircraft modifications, such as DC-10/MD-11 passenger to freighter Mods. Having to work out the design and installation issues on each robot have given him a greater appreciation for the efforts of the "detail guys".

**Using the Motor Mind C Analog Mode – by Paul Garcia****History:**

In the Fall of 2001, I joined the Robotics Society of Southern California (RSSC). Since I had limited experience with either robots or micro controllers, I started out by building the club robot, the Ebot:



The Ebot is a differentially steered circular platform, with modified R/C servos providing the motive power. While quite functional, many members expressed a desire to have a platform that utilizes geared DC motors instead of servos, and provides room for adding upgrades such as wheel encoders, range sensors, bump skirt switches, etc.

The turning point for me came one day while getting ready for a club line following contest. My Ebot was just not behaving correctly, and I found myself having to repeatedly adjust the speed correction factor for each wheel. After many hours of programming and troubleshooting, I finally determined that the servo stop speed had drifted, causing all my speed commands to be out of whack.

It was time to get started on that "new and improved" robot.

**The Ebot II Project:**

There are widely differing opinions on the best Micro Controller Unit (MCU - Basic STAMP, ATOM, BX-24, PICs, AVRs, HC68s, etc.) and the languages used with them. The Ebot II design focuses on providing a sturdy platform/motor/controller package that can be built by beginner, and used with the builder's choice of MCU.

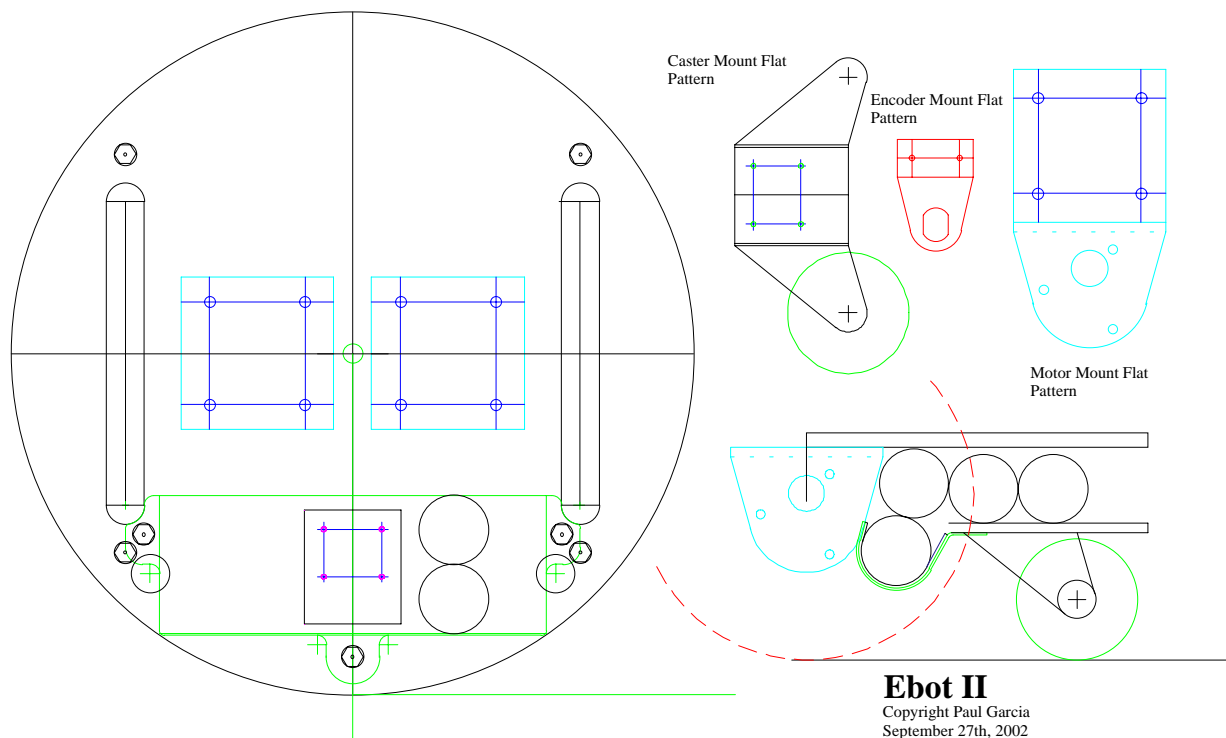
One of the concepts that most club members do agree on, is the desirability of easy to use "robot building blocks". Not too many members enjoy having to develop each hardware/software interface from scratch for their particular combination of parts.

**Design Considerations:**

Center of Gravity (CG) – Robot CG is a design element that is often given full consideration right about the time the robot makes its first rapid deceleration and subsequent nosedive. The Ebot II battery placement is intended to keep the bulk of the robot's weight at, or about axle height, which minimizes pitching during acceleration or deceleration. With both batteries behind the axle, only a single caster has proven necessary to maintain the robot's balance.

Expandability – The design needs to be easily expandable as the builders' skills and requirements grow. The Ebot II allows the user to add platforms as required (and within CG limits).

Parts availability – A project intended to be reproduced by others should use "off the shelf" parts wherever possible. All parts and raw materials used on the Ebot II are readily available from the R/C section of the hobby shop, home improvement center, local electronics store, or mail order sources.

**General layout:**

**Motor controller:**

I considered a number of different controllers. Researching the subject on the web, I found that controllers fell into 3 major groups, Analog, R/C, and Serial. The Pros and Cons for each type of controller will vary with the specific application. Also, the specific capabilities of each MCUs will determine how easy or difficult utilizing a particular mode is.



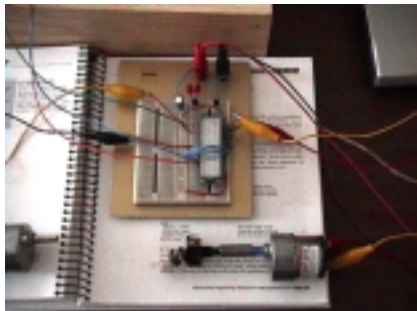
Controllers in each group are available in a wide range of current ratings, features and states of assembly - with prices to match. The only unit I have personally come across which allows the user to select any of the 3 modes noted above, is the MMC. This handy feature provides a great deal of flexibility, particularly when it becomes necessary to change the design approach in the middle of a project. (Yes, that is the voice of experience). Since the MMC was compatible with the current requirement of the geared

motors I was using, I decided to give it a try.

Since R/C mode controllers require refreshing the speed command every 20ms, and the software demands of serial mode are not especially beginner friendly, I chose to concentrate on analog mode.

**Using the MMC:**

After reading the [MMC data sheet](#), I began by bread boarding the MMC for dual motor analog mode per figure 8 of the datasheet.



I used the Parallax BASIC STAMP, and the Basic Micro ATOM for initial testing since I had them on hand. Both the STAMP and ATOM have software PWM (Pulse Width Modulation) commands that allow the user to specify the output Pin, Duty and Cycles. When used in conjunction with a RC filter, the resulting analog output can be connected to the MMC AIN1 & AIN2 pins. I used the 10K resistor and 0.1uf Cap described in the STAMP manual. Different combinations of Duty, Cycle, and repetition were tested with a simple software loop:

Main:

```
PWM 0, 255,40      'Full speed motor 1 (5 Volts)
PWM 1, 255,40      'Full speed motor 2 (5 Volts)
PAUSE X
```

GOTO Main

The value of X was gradually increased until the motor began to “ratchet”. With my setup, I found it was necessary to repeat the PWM command every 20ms or so to maintain smooth operation. Having to update a command every 20ms can be a nuisance, particularly when using a slower MCU. I didn’t see the analog mode with software PWM command as providing any advantage over R/C mode.

Some MCUs (ATOM, AVR, BX-24, HC68s, PICs) have built in hardware pwm capability. With hardware pwm, once the command is given, the pwm/voltage output is maintained until a new command is given. This leaves the MCU free to perform other tasks (such as navigation) without having to repetitively refresh the software PWM command. Like software pwm, hardware pwm also requires the use external RC filter.

Using hardware pwm on the ATOM was a snap. A simple HPWM (Pin, Period, Duty) statement for each motor, and the MMC was up and running.

Main:

```
HPWM 9, 16383, 16383      'Full speed motor 1 (5 Volts)
HPWM 10, 16383, 16383    'Full speed motor 2 (5 Volts)
```

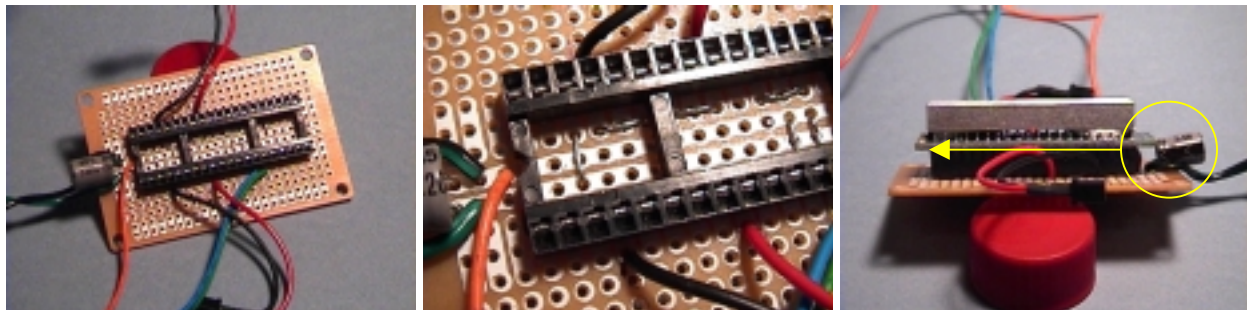
Done:

```
GOTO Done
```

The motors continue to run at the commanded speed while the MCU is in the “Done:” loop. Enclosing the HPWM commands in a FOR..NEXT loop can be used to ramp the motor speeds up and down. For my Ebot II implementation, the decision was easy - I chose the ATOM with HPWM.

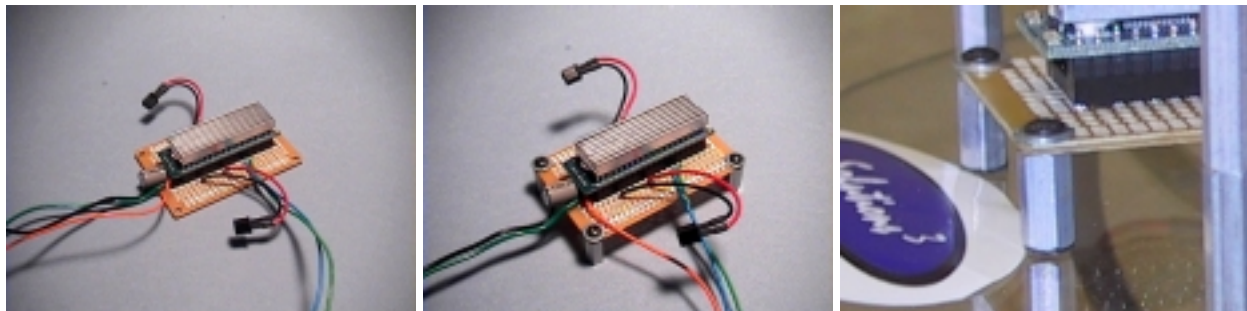
### Building the carrier board:

With testing and mode selection complete, the next step was fabricating a carrier board.



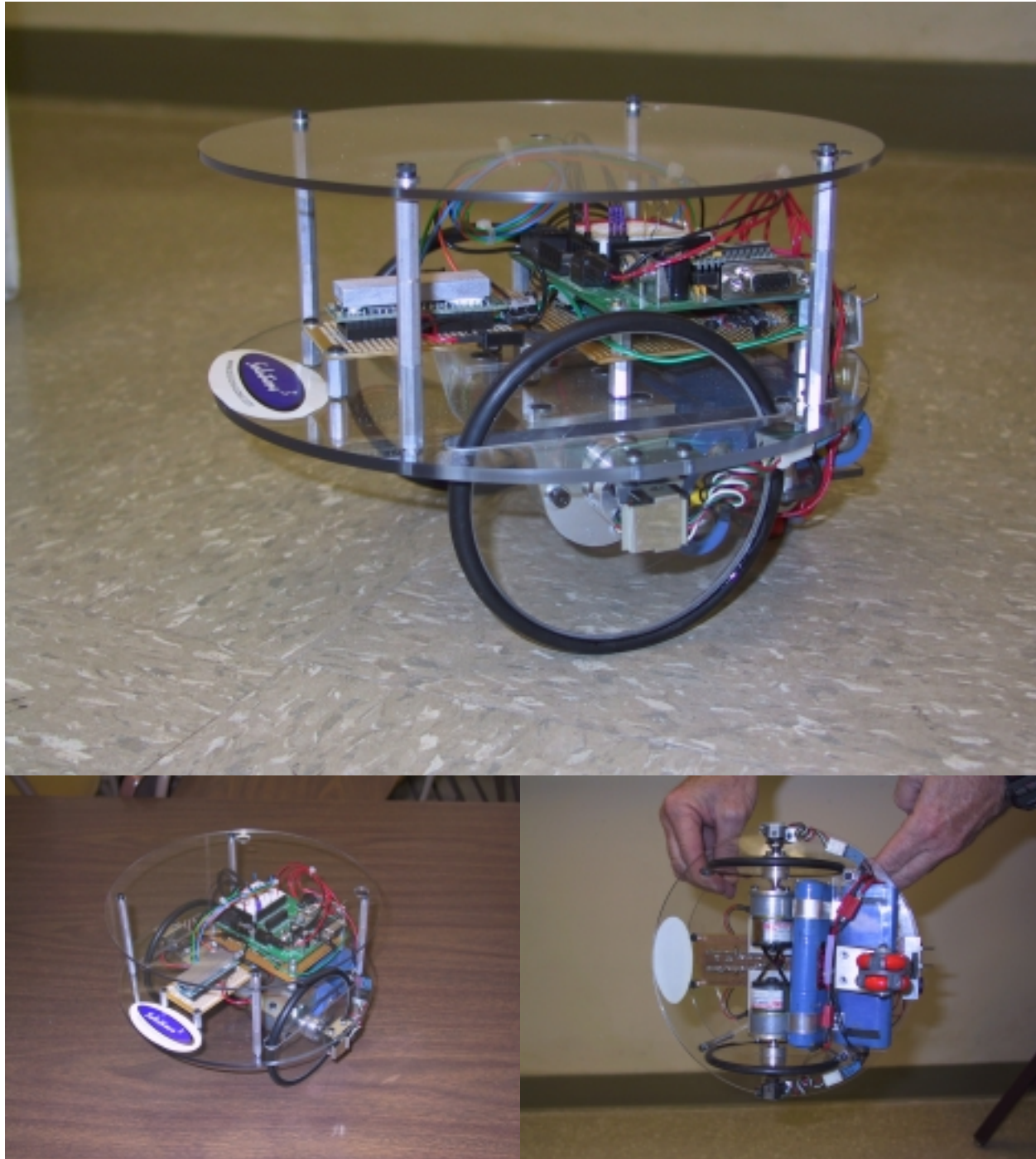
I used a readily available prototype board. The solder pad pattern was just a bit narrow for the 40-pin socket, so most of the jumpers were installed under the socket. (I used leads snipped from a resistor for jumpers; I find that it's quicker than cutting and stripping wires, they bend well, and solder easily). When all the jumpers had been installed and checked against the breadboard and the datasheet, the socket, capacitor, and wire leads were added. Dean's micro connectors were used on the VM and motor leads.

Unfortunately, as the side view shows, I had installed the socket in the center of the board, overlooking the fact that the MMC overhangs the socket. As a result, I was forced to lay the large capacitor on its side, which causes it to stick out beyond the edge of the board. Not a disaster, but it means the MMC board can't be butted up against another board. Mounting the socket and jumpers all the way to one end of the board would alleviate this problem.



After the carrier board was completed and tested, it was mounted on 1/2" standoffs and installed on the robot.

The Assembled Ebot II:



Note: The omniroller tailwheel isn't really crooked – the wide-angle lens has caused some minor distortion.

**Final Testing**

Once assembly was completed, I loaded some simple software that ran both motors forward at the same speed for about 3 seconds before stopping. Of course the resulting path of travel wasn't a straight line. Some iterative testing produced a correction factor that soon had the Ebot II running in a straight line. However, when the robot was stopped, I noticed that the right wheel (motor 2) would occasionally chatter. Swapping the motor leads made no difference as the problem remained on the right (motor 2) side. Reviewing the datasheet, I decided to check the actual pwm output from the MCU. The voltage coming off the #2 pwm output was a little low at 2.46 volts (the deadband is 2.42 to 2.57 volts). The output was adjusted to 2.5 volts, but the wheel continued to chatter while stopped. In the process of swapping the AIN1 and AIN2 leads, I discovered the problem. While experimenting with different component values for the RC filter, I had neglected to change the resistor value when I changed the capacitor back to the original 0.1uf value. The resistor was replaced with the correct value, and the chattering was eliminated.

I presented the completed robot at the September RSSC mtg. I wasn't too sure what type of response I was going to get (seeing as I'm proposing a replacement for the club mascot robot), and was quite pleased with the enthusiastic feedback for the overall platform/controller design.

**Tips:**

The MMC turned out to be very easy to use. I only have two tips specific to analog mode and the MMC –

- 1) Verify the pwm output of your MCU/RC filter. A 50% Duty on each output may not give you exactly 2.5 volts on each out put. In the long run, it's easier to program and maintain speed control if both motors have an equivalent stop point.
- 2) Don't use STOP or END commands in your software when the motors have come to the final stop point. Both the STOP and END commands cause the processor to stop, at which point the voltage bleeds off the RC filter cap, and causes the motors to spin. I barely caught the Ebot II before it launched off the table learning this. With software PWM, it's better to conclude with an endless loop with the stop speed command inside. HPWM can use an endless loop after the stop speed command:

**STAMP Code Snippet**

```
Done:
    PWM 0, 128,40      'stop motor 1
    PWM 1, 128,40      'stop motor 2
GOTO Done:
```

**ATOM Code Snippet**

```
HPWM 9, 16383, 8191      'stop motor 1
HPWM 10, 16383, 8191     'stop motor 2

Done:
GOTO Done
```

**Tips (continued):**

The rest of my tips really pertain to any new product you may be trying to interface – particularly when you think it's not working right:

- 1) Start with a simple setup, and software - I have traced many problems to my trying to write the interface software for a new addition (like wheel encoders) in one whack. Start simple. Make sure the MCU is receiving and sending the right signals. Once you have the components interfacing correctly, then you can start building up.
- 2) Test at each step - While building up a project, I test at each step to make sure everything is still working. It's easier to troubleshoot when you know it's the last addition causing the trouble, instead of trying to figure out which of the last six things you've added is the culprit.
- 3) Sockets and connectors - Although it adds a few dollars to the project, use sockets and connectors between the batteries and individual components. Modular assemblies make trouble shooting easier, and make it convenient to move modules from project to project.
- 4) Hardware mounting – Most of my previous projects are full of misplaced, pulled, and abandoned mounting holes. I've starting using steering servo tape from the R/C car section of the hobby shop. It has thinner foam than regular servo tape, and holds great. Now I can move components around to find the best location, without turning my platform into Swiss cheese.

Paul Garcia  
RSSC  
September, 2002

Schematic:

