MicroMouse: An Intelligent Maze Solver Robot

Ahmed N. Mosa **Ayman M. Saad** **Moustafa A. Mohamed**

Supervisor
Samir I. Shaheen

Department of Computer Engineering,
Cairo University,
Giza – Egypt.

Abstract

The Micro-Mouse is an autonomous robot that is capable of moving about within a maze and finding its way to the center in the shortest possible amount of time. The complete robotic system is a synthesis of mechanical design, motor control, sensory input systems, electrical design and software development system.

Keywords: Robotics, Embedded Systems, InfraRed Sensors, Maze Search.

1. INTRODUCTION

The IEEE Spectrum magazine introduced the concept of the micromouse in May 1977, Spectrum announced the 'Amazing Micromouse Competition'[1], which would be held in 1979 in New York. Later on, this competition became an annual competition held by the Institute of Electrical and Electronics Engineers (IEEE). The object of the competition is to build a small, autonomous robot, limited to a 25 cm square footprint, with unlimited height, that is capable of moving about within a maze and finding its way to the center in the shortest possible amount of time, and then return to the start cell for another attempt. The maze is made up of a 16 by 16 grid of cells. Mice must find their way from a predetermined starting position to the central area of the maze unaided. The mouse will need to keep track of where it is, discover walls as it explores, map out the maze and detect when it has reached the goal.

The aim of this paper is to present our micromouse design. It's essentially an integrated system in which a CPU monitors an assortment of sensors, computes the shortest path, and directs the motors. Although a lot of ideas have been proposed for the design, yet none was proven to be ideal which still makes the design a challenging engineering project with plenty of confusing alternatives.

Following a literature survey we describe each part in our robot. A description of the control software and the maze-solving algorithm will then follow. Finally we conclude this paper with key results and conclusions that could improve future designs.

2. LITERATURE SURVEY

The following survey looks upon micromice that performed admirably in the competition or introduced innovative design techniques[2].

Nick Smith's Sterling Mouse was the first micromouse to find the centre of a maze with a run time of 3 minutes and 49 seconds.

In Dave Woodfield's Enterprise, built in 1984 wall sensing was achieved by means of seven top-down reflective infra-red (IR) sensors fixed on wings on both sides that extended over the wall: three sensors on each side and one out front. The main problem with the extended wings has to do with the rotational inertia. However light they're, they have to extend a relatively long way from the center of mass and so contribute a relatively large component of the total inertia. In addition, the arrangement of the sensors on the wings may suffer mutual interference.

MITEE 7 is a four-wheel drive, four-wheel steering mouse. There are a couple of significant advantages to a four-wheel mouse. Chief of these is in going quickly. With four wheels working

Manuscript submitted on September 2005 to IEEE Egypt Section Student Paper Contest. The result of the contest will be announced in mid-January.

together, they all get to do some work whatever the weight distribution. Each motor need only have ½ the torque needed in a two-wheel mouse and can be correspondingly smaller. However, this approach is mechanically and control-wise more complex than two wheel machines.

3. CHASSIS AND MOTORS

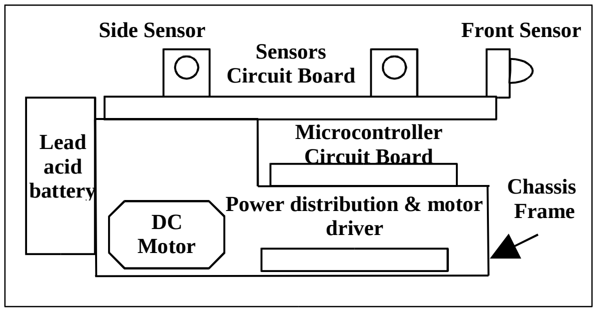


Figure 1: Side-view of the micromouse

The chassis is 25x22x12cm, made of plastic. It moves on four bulky wheels. The diagram above illustrates how different parts of the system are arranged about the chassis. Two DC motors drive the rear-wheels independently. The front-wheels are fixed and cannot be steered. Consequently, the micromouse can make smooth turns on a curved path by rotating the rear-wheels at different speeds. It can also do in place turns**,** about the center of mass, by rotating the rear-wheels in opposite directions at the same speed. The liquid acid battery, which represents a considerable portion of the total mass, is attached to the rear end of the chassis to set the center of mass at a mid-point between the rear wheels. That is necessary to avoid skidding during in place turns.

The DC motor rotates at a speed that's proportional to its operating voltage. The voltage regulators, and bypass capacitors help provide a stable voltage all the time. Also, rapid successive switching of the motors is avoided in the software to evade spikes on the power line. Thus, the motors speed is almost constant.

Knowledge of the constant rotational speed of the motors and the duration they're switched on, gives enough information to track the displacement of the micromouse as it moves.
To drive the DC motors an integrated H-Bridge circuit is used. The IC facilitates independent switching of both motors.

4. PROCESSING UNIT

The microcontroller is the processing unit, which monitors the sensors, executes the search algorithm to change the micromouse's heading, and controls the motors correspondingly. A micro-controller is the most adequate unit to be used compared to other processing units like microprocessors or DSPs. Microcontrollers have built in RAM, Flash ROM, I/O ports, interrupt controllers and counters, which makes them compact, cheap, and simple. The model used is the Atmel 89C52[3]. It has adequate processing power and a variety of useful peripherals. The choice is mainly dependent on availability of the chip and a compatible programmer device.

5. SENSORS

The micromouse, makes use of only a single type of sensors, namely infrared proximity detection sensors. The sensors are used for detecting head-on obstacles and the side-walls. Infrared sensors work by sending out a beam of IR light, and then detecting the reflected signal. The detector's operation is usually hampered by ambient radiations; the solution is to modulate the transmitted IR and then have the receiver circuitry (a filter circuit) only respond to the modulated IR signal. Modulation in this case consists of turning the infra-red light emitting diodes (IR LEDs) on and off rapidly. A digital detector is used, which only triggers when the strength of the reflected signal is above a threshold in contrast to an analog detector whose output corresponds to the strength of the detected signal. Hence, the sensors are used for only proximity detection of the maze walls.

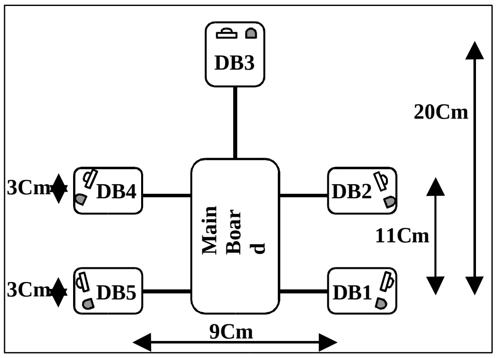
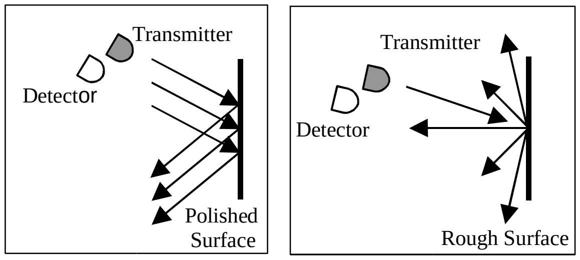


Figure 2: Sensors Layout

Figure 2 above illustrates the layout of the main board and the five **daughter boards (DBs)** on the robot's top-surface. DB2 and DB4 are used for short-range detection of the side-walls; DB5 and DB1 are for long-range detection of the side-walls, while DB3 detects head-on obstacles. Each receiver is placed at a distance from the associated IR LED, away from its irradiance bounds, to avoid detection of the incident signal instead of the reflected one. The front-sensors (DB2 – DB4) are oriented away from the vertical axis and operate at short range, detecting a side-wall when the robot is on a crash course with it. The rear-sensors (DB1-DB5) are high-power sensors operating at long range for wall detection. They're oriented away from the vertical axis to prevent *mutual interference* between them and the front-sensors.

The relative orientation of the sensors with respect to the wall enables it only to detect *diffuse reflection* (Figure 3a) and not *specular reflection* (Figure 3b). Since the maze walls are not highly polished diffuse reflection should be high enough due to its low emissivity factor, which supports proper operation of the sensors. Figure 3 below illustrates the effect of different *surface-emissivity factors*.



a) Specular Reflection **b) Diffuse Reflection**

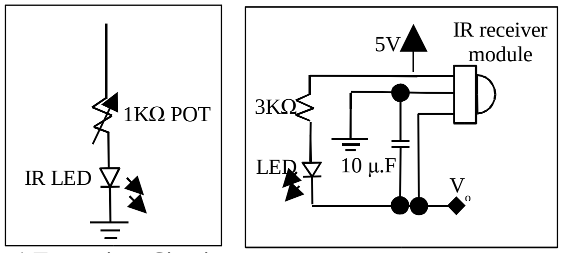
Figure 3: Effect of surface-emissivity factor

In case of an ideal polished surface the reflected signal is traveling away from the detector, but using a rough surface, the practical case, the detector can sense the reflected signal due to diffusion.

The sensors function as follows: short-range sensors are primarily used to correct the robot's heading while the long-range sensors are used to detect walls on the sides of the robot. If a wall is detected an update occurs to the maze-map inside the robot's memory. The update is to a map-cell corresponding to the robot's location. It's also used to synchronize the software displacement value with its real location in the maze, thus avoiding accumulation of error in displacement due to the mechanical inaccuracies. The front sensor serves the same purpose like the long range sensors but for head-on obstacles.

The main board, in Fig. 2, bears the modulation circuit - a 555 timer tuned at 6.4 KHz. The output of the 555 timer is used to switch five similar transistors that drive the IR LEDs. An alternative approach is to use the microcontroller for modulation but that would slow the execution of the algorithm significantly.

Each of the daughter boards holds transmitter and receiver circuits as shown in figure 4.



a) Transmitter Circuit **b) Receiver Circuit**

Figure 4: Transmitter and Receiver circuits

The pickup signal V_o from the receiver, in Figure 4b, switches from 5V to 0.2V upon detection of a properly modulated signal. The signal is fed to a microcontroller port.

The transmitter circuit on a daughter board is simply a potentiometer and a transmitter. The potentiometer is used to adjust the power and therefore the range of the transmitter.

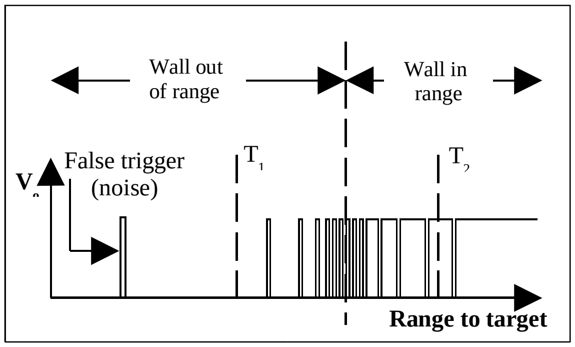


Figure 5: IR Reciever signal

The pickup signal $V_{o,}$ fed to the microcontroller, exhibit glitches at a certain range of the target surface from the detector. The glitches diminish and the signal approaches a constant value as the detector moves out of that range as in Fig. 5. A bypass capacitor is added at the receiver's output pin, in Fig. 4b, to reduce the effect of glitches. The value of the capacitor is determined according to two contradicting constraints - high capacitance delays the receiver from triggering and low capacitance cannot suppress the glitches.

Consequently, V_o signal is sampled by the microcontroller for each detector once every 255 microseconds. That is faster than glitch-transitions. Depending on the sample's value a counter's value is changed between two saturating limits with one counter dedicated for each detector. The counter is incremented when the sample is a logic-0 and it's decremented otherwise. The counter's value is interpreted as follows: if the counter's value exceeds a threshold value (T_1), as shown in Fig. 5, then the "wall-exist" flag is raised, if it drops below another threshold value (T_2), as shown in Fig. 5, such that $T_1 > T_2$, then the "no wall exist" flag is raised. This concept of having two different thresholds is called *hysteresis*, borrowed from the electronic device: The Schmitt Trigger[4]. With only a single threshold the control algorithm would oscillate indefinitely about it while trying to maintain a proper heading for the robot: moving in a zigzag.

Also the time for transition between no-wall to wall-exist state should be chosen such that it is long enough to ensure the fidelity of the sampling process. On the other hand it should be small enough for the micromouse to react when heading towards a wall and before deviating from its original track. Also this period should be adequate for tilting in the other direction with a small angle, after the motor starts responding; thus, avoiding sharp tilts and zigzag response. Hence, even if the mouse started with a large angle to its original track, it will steadily zigzag till it converges to the center-line of the path; its original track.

6. POWER

The power supply block is made up of a 9V dry-battery for the transmitters and a 12V lead-acid battery feeding a power distribution board that provides regulated supplies of 5V for the microcontroller and the IR receivers, 6V for the motors.

To evade false-triggering of the receivers, we've avoided rapid-successive switching of the motors in the software. Also, splitting the power supply and the ground line of the transmitters from the other supply is to avoid ground-line bouncing that would false-trigger the receivers.
The lead-acid battery adds a considerable weight to the mouse but it is the only resort when dry batteries fail to satisfy the power consumption requirements for a reasonable duration during development.

The following analysis shows the power consumption of the system:

Motor: 6V – 300mA.
Transmitters: 9V – 70mA.
Receivers: 5V – 2mA.
Microcontroller: 5V – less than 200 µA
Total Power Consumption = 4.951Watt

7. MAZE SOLVING ALGORITHM

Mazes built for micro-mouse contests are generally not simple ones and are usually built in a challenging way, which is difficult to solve by traditional algorithms directly.

What makes the maze-solving algorithm more complicated is developing one for an embedded system. Embedded systems are ones that impose tight time and memory constraints on the system. These constraints are mainly:

1. Real-time processing.
2. Limited memory.
3. Power limitations.

Our objective is: given a 16x16 grid maze of known cells dimensions, it's required to find a way to the center of the maze in the shortest time and without leaving any marks.

One of the earliest algorithms for maze solving is Lee's algorithm[5]. Lee's algorithm, given a fixed grid maze, can find the shortest path between two points. For a given maze, as in Figure 6a, the algorithm proceeds in 2 phases as follow:

1. *Filling Phase:* As shown in Fig. 6 below, we begin to fill the neighboring cells to the starting point (S) with a one; the next cells are marked with two (these values are the distance of each cell from the source). We repeat this till we reach the target point (T).

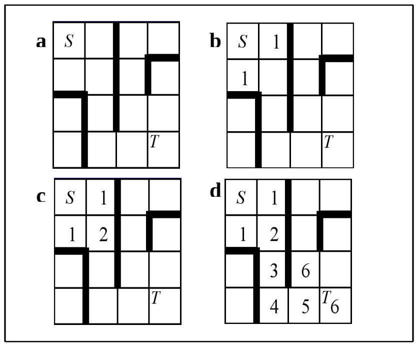


Figure 6: The Filling Phase of Lee's algorithm

2. *Retrace phase:* If the target was reached in step I, we trace our way back by going to the cell with value I-1. We repeat this till we reach the starting point S as shown in Fig. 7.

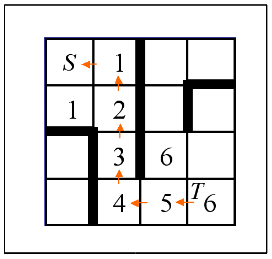


Figure 7: The Re-Trace phase of Lee's algorithm

One main drawback of Lee's algorithm is that it assumes that the maze structure is known, and hence it starts its search procedure. Consequently, maze runners adopt a variant of Lee's algorithm widely known in the Internet community as the Flood Fill algorithm. The Flood Fill algorithm assumes at the start that there are no walls in the maze as in Fig. 8 and then uses Lee's algorithm to find its way to the target. As the mouse proceeds through the maze, it discovers new walls. These walls are added to the mouse's knowledge base and then it applies the Lee's algorithm once more to decide the next cell to move to as shown in Fig. 8.

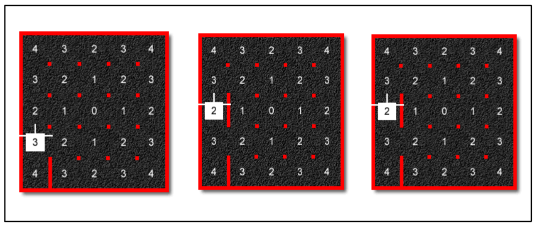


Figure 8: The Flood Fill algorithm

Hence, the algorithm for maze solving is:

Algorithm:

```
Apply Lee's Algorithm
While goal not reached
        Move to next cell closer to goal
        If new wall detected
                Update maze structure
                Apply Lee's algorithm
```

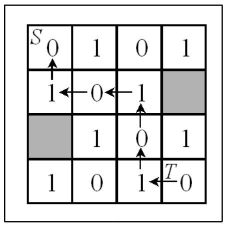
By analyzing Lee's algorithm (the core algorithm) we can see that it's a breadth first search (BFS) algorithm; an application to Dijkarta's algorithm. This is more suitable to our system as it guarantees to find the optimal solution unlike depth first search (DFS) algorithms, which include a lot of search and retreat till it finds the goal coincidently, and of course not necessarily the optimal solution is found. Yet, Lee's algorithm doesn't perform very well in terms of time and memory:

Time complexity = $O(n^2)$
Memory complexity = $O(n^2)$
n is the number of cells in the whole maze.

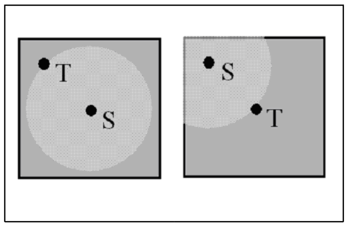
It suffers from the huge memory requirements, and slow performance. This is intolerable in embedded systems due to the tight resources (256 bytes!!) and real time constraints on the system.

Hence, several enhancements and speedup techniques to Lee's algorithm were proposed in the literature. Akers[6] proposed a two bit encoding technique per cell instead of saving the real distance as shown in Fig. 9a. Yet we used another, but similar encoding, where each cell saves a two bit pointer to one of the neighboring cells that is part of the shortest path. This approach will prove to be better in the modified flood fill algorithm, which we will illustrate later on. Furthermore, encoding was applied to other data structures of the algorithm for further compaction in memory usage.

Decreasing the search space is another option. The choice of source or target point to start searching from has a significant impact on the search area as we can see in Fig. 9b. In our case we start the search from the starting point in the corner until we reach the goal.



a) Akers' Encoding



b) Search Space Reduction

Figure 9: Memory optimization Techniques

Performance-wise, a lot of ideas have been proposed to speed up the search procedure as in A* algorithms.

From the algorithm we can see that Lee's algorithm is applied every step from scratch as it discovers a new wall. Hence, it makes no use of the results of its previous search. Consequently, an update procedure is used to modify the last search without re-starting. This technique is known as the Modified Flood Fill algorithm[7], which again is widely used by micromouse contesters.

We also incorporated some heuristics to speed up the search process. Heuristics are usually application specific and to deduce our own set of heuristics we needed to perform some statistics for different mazes from real IEEE contests. We brought a set of 50 mazes and ran a batch program to compute the average, maximum and minimum values of he following parameters:

1. Path length.
2. Number of turns
3. Number of continuous cells

These statistics helps us to take decisions about taking turns or going forward as we head towards the goal, especially when we meet two directions that will lead to the goal in the same number of steps.

Time-wise, moving forward is much faster than turning, which requires the mouse to stop completely for in place turns. Power-wise, continuous uniform movements are much more efficient than repeated unsteadiness in voltage. Finally, from the heuristics, we can deduce the best direction to follow and embed it in the search algorithm as a cost function. The main heuristic we used was to try to keep steady movement in the same direction on the shortest path to the goal.

8. RESULTS

Our micromouse was a prototype that was tested on an 8x8 maze. The mouse was able to solve it in about 84 seconds, finding one of the shortest paths, even though; the algorithm is not guaranteed to do so in all mazes. On the other hand, the maze that we built pretty much complies with the IEEE contests, except for its size, which was too large to build, and material of the walls, yet it reflects IR beams as the IEEE maze walls.

9. CONCLUSION

Interest in robotics and international robot contests has increased a lot recently in Egypt. In this paper, we propose a robot composed of simple components found locally in Egypt and yielding competitive results compared to similar robots. We also have introduced some of the challenging problems, in all fields; one will meet in a similar project. We have presented some innovative ideas in the sensory system, digital control system, and the software. Yet there is a lot of room for more mechanical improvements.

References:

[1]http://www.csuchico.edu/ieee/micromouse.html

[2] http://micromouse.cannock.ac.uk/

[3] Intel MCS 51 Microcontroller family user's manual, 3rd Edition, 1994.

[4] Schmitt, "A Thermionic Trigger," *Journal of Scientific Instruments*, vol. 15, pp. 24-26, 1938.

[5] Lee, C.Y., "An Algorithm for Path Connection and its Applications", *IRE Transactions on Electronics Computers*, 1961.

[6] Akers, S.B., "A Modification of Lee's Path Connection Algorithm", *IEEE Transactions of Electronics Computers*, February 1967.

[7]http://www.micromouseinfo.com/introduction/mfloodfill.html

[8] SHARP GL380/GL381 Datasheet.

[9] SHARP GP1U58X Series Datasheet.

Samir Shaheen received the BSc in Electronics and Communications and the MSc in Computer Engineering from Cairo University, Cairo, Egypt in 1971 and 1974 respectively, and the PhD degree in Computer Engineering from McGill University, McGill, Canada, in 1979.

His research interests include Computer Networks, Image processing, Computer Graphics, and Database systems.

He is currently the Dean of Faculty of Engineering and the Vice Chair of IEEE Egypt Section.

Ayman Saad received the BSc (Distinction with honor) in Computer Engineering from Cairo University, Cairo, Egypt in 2003. He is currently researching character animation using dynamics simulation in his MSc at the Computer Engineering Department, Cairo University. He received the medal of academic distinction from Cairo University in 2003.

His research interests include Computer Graphics, Robotics, and Computer Architecture.

Ahmed Mosa received the BSc in Computer Engineering from Cairo University, Cairo, Egypt in 2003. He is currently working as a Software Developer in Arkan, Cairo, Egypt.

Moustafa Mohamed received the BSc (Distinction with honor) in Computer Engineering from Cairo University, Cairo, Egypt in 2003. He is working on his MSc degree at Applied Mathematics Department, Cairo University. He received the medal of academic distinction from Cairo University in 2003.

His research interests include Computer Aided Design, VLSI Design, Embedded Systems, and Computer Architecture.

He is presently a Teaching and Research Assistant in Mathematics Department, Cairo University and working in collaboration with Mentor Graphics Egypt in Computer Aided Design research and development.