Proceedings of the 2011 IEEE
International Conference on Mechatronics
April 13-15, 2011, Istanbul, Turkey

IEEE Micromouse for Mechatronics Research and Education

Steven G. Kibler, Andrew E. Hauer, David S. Giessel, Chloe S. Malveaux, and Dejan Raskovic
Electrical and Computer Engineering, University of Alaska Fairbanks
PO Box 755915, Fairbanks, AK 99775-5915, USA
sgkibler@alaska.edu

Abstract — **IEEE Micromouse is a university level competition in which student teams build small robots capable of exploring, mapping, and racing through a small maze. Building a successful micromouse involves the design and construction of the hardware, control, sensor and system integration, microcontroller integration, and development of computer algorithms for mapping and route planning. Beyond the technical skills, team members also learn team work, time management, and communication skills. Micromouse gives students the ability to learn these valuable skills and, more importantly, gain some experience, thus giving them an edge in a very competitive job market. With the motivation we have gained as winners of the IEEE NW Area Micromouse Competition for the past four years in a row, in this paper we aim to describe the important technical aspects and the required continuous improvements to the system in terms of design, controls, and computer algorithms for a better, faster autonomous robot. We also show how a micromouse program can be beneficial, especially for undergraduate research and education in robotics and mechatronics.**

***Keywords*—Micromouse, Mechatronics, Research, Education.**

I. Introduction

ROBOT competitions for students ranging from middle to graduate schools have been growing in quantity, scope, and size over the last two decades. The size and scope of these competitions vary considerably. There are ground based navigation challenges, such as DARPA Grand Challenge, which required autonomous vehicles to traverse several miles of rugged desert terrain. It was followed a couple years later by DARPA's Urban Challenge in which autonomous vehicles were required to traverse an urban setting [1]. While those competitions are over, students may compete in a similar competition known as Intelligent Ground Vehicle Competition sponsored by the AUVSI Foundation [2], in which students must design autonomous vehicles that can complete an obstacle course. The AUVSI Foundation also sponsors air and water vehicle competitions that push the ability of autonomous vehicles in those mediums. There are also competitions for robots that are more humanoid, such as RoboCup, which organizes a competition in robotic soccer [3]. Their goal is to have a robotic soccer team beat a human soccer team by the year 2050. Some competitions, like FIRST robotics (For Inspiration and Recognition of Science and Technology), which focus on middle school and high school students, have different tasks to complete each year, thus removing the ability for future students to repeat successful robot designs [4].

The IEEE Micromouse competition is one of the longest running IEEE student competitions. The competition's greatest features are its relative simplicity and low expense. A micromouse is a small, low-cost system that allows students to build a micromouse for a design class or a competition. Micromouse is also a system that is simple enough to allow a single student or a small group of students to build it fairly quickly, leaving enough time to program it and experiment with it inside a single semester.

The main contribution of this paper is to emphasize the benefits of a micromouse program in universities as a flexible, exciting, and yet extremely economic means to prepare students for many practical aspects of robotics and mechatronics. To this aim, the paper discusses how each technical aspect can be used particularly for undergraduate level research and education, while also providing some technical feedback on developed practices as a result of team research.

The paper is organized as follows: After the introduction in Section I, details about the competition itself will be discussed in Section II, followed by technical design details of the mouse itself in Section III. Finally, Section IV discusses the contributions micromouse makes to education, and Section V is the conclusion.

II. IEEE Micromouse Competition

The IEEE Micromouse competition began in the late 1970's, on 8×8 mazes, with mice achieving speed-run times of around 30 seconds. Since that time competitions have spread around the world, but they are particularly popular in Japan, Taiwan, Indonesia, the U.K. and U.S. Currently the mazes are 16×16 cells, and top mice have race times of less than 7 seconds, covering routes of over 70 cells. This puts current world record mice at a speed of between 2 to 4 meters per second. The competition has gotten so advanced that leading micromouse designers work through the year for increases of hundredths of a second.

For the competition, each mouse begins with some basic knowledge of the maze. Each cell is an 18 cm × 18 cm unit square. In the maze there are 16×16 of these unit squares. The walls of the maze are 5 cm tall and painted white to be reflective to infrared light. The floor of the maze is painted black so that it will not reflect infrared light. The mouse knows that there are walls around the outside perimeter so that

978-1-61284-985-0/11/$26.00 ©2011 IEEE

it cannot fall off the maze; it also knows it starts in one corner and finishes in the center. Beyond that it knows nothing about the configuration of the walls within the maze. Thus, the first job of the micromouse is to explore the maze sufficiently to determine an optimum route from start to finish. As it moves through the maze it tracks its coordinates and maps each cell's walls into its memory. After it finds the center it will continue to explore the maze, eventually exploring it sufficiently and deciding to move back to the start cell. From the start cell it analyzes the map as it knows it and determines the best route to the center. It then races along that optimum route as fast as it can (timed run). The mouse has ten minutes in the maze to accomplish all these tasks. The mouse that has the fastest timed run wins. While this sounds fairly simple and straightforward in the overview, the reality is quite complex. Each individual part alone requires a firm understanding of the fundamentals. Especially among university teams, ourselves included, the project is very often divided into three main systems: mechanical, control, and mapping.

III. Micromouse Design

A successful micromouse needs to be treated as an embedded system and designed using a hardware-software co-design approach [5]. While designing the hardware of the mouse, it is important to consider in what way the software and the hardware will interface.

A. Mechanical

The mechanical design of a micromouse is probably the most important aspect of the overall design yet also the most neglected amongst university teams. The mechanical design influences every other aspect of the mouse design, from how it moves to how it senses walls. Because of the short time-frame involved in the university level competition as well as their other commitments, student teams tend to rush through the mechanical design without giving it much consideration, resulting in micromice that are barely capable of the precise movements required. These mice are usually overly bulky, often top-heavy, and have sensors placed in less-than-ideal locations. For optimum performance though, there are some primary rules-of-thumb that should be followed in the physical design of the mouse.

Generally, the smaller we made the mouse the better it performed (see Fig. 1). This is true because making the mouse smaller typically lowers the center of gravity and the moment of inertia of the mouse, and reduces the friction between the wheels and the floor. As the center of gravity lowers, the tendency for the mouse to "rock" backward and forward upon acceleration and deceleration decreases. These rocking motions create control issues that can push the mouse off course. The lower moment of inertia makes the turning motion more repeatable because it takes less power in the motors to make it move. Less power to the wheels results in less wheel slip and thus more consistent turns. When designing a mouse then, it is important to keep the center of gravity low and the moment of inertia low and centered between the wheels.

1) Encoders allow us to determine how much the motor has actually turned. Type and location of encoders is very important. Also, the higher the encoder count the better the mouse may be controlled. However, it is important to avoid overloading the microcontroller with encoder counts that occur too often. The appropriate number depends highly on the microcontroller chosen and the clock speed at which it is run. Our first mouse had 98 encoder counts per wheel revolution. They were placed in a position to look at the wheel directly, with a pie-like encoder disk placed directly on the wheel [6]. While this was sufficient to get us to the center of the maze, our controls were less than ideal. Also, because the encoders were placed looking at the wheel, they could not be sealed and thus they were more prone to interference from ambient light conditions. Since then, we have modified the original design by cutting a hole in the can of the motors themselves, painting white and black stripes on the rotors, and placing the encoders to look directly into the motors (see Fig. 2). This has given us 12 encoder counts per motor revolution, which when multiplied by the 30:1 gear ratio results in 360 counts per wheel revolution. This quadrupling of the encoders has given us much better control, resulting in the ability to run nearly straight down the corridors of the maze. In this design the encoders are also sealed with the motor and thus immune to ambient light. In future designs we are considering purchasing motors with high count encoders built onto the back end of the motor can, attached to the shaft. These would give us 512 counts per motor revolution, considerably higher than our microcontroller can currently handle, requiring an external counter. Those motors could also significantly increase the cost of the mouse.

2) Infrared (IR) Sensors: Most micromice use IR LEDs and IR light-to-voltage sensors to determine wall locations and its distance to the wall. We have come to find that our mice were

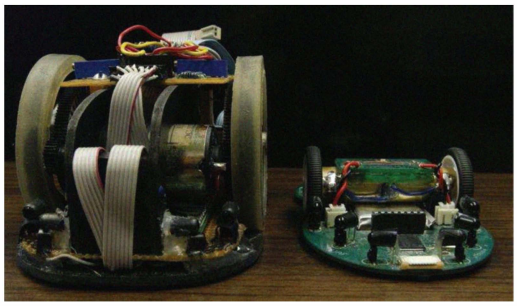


Fig. 1. Our first (left) and our latest (right) mouse (ver. 4).

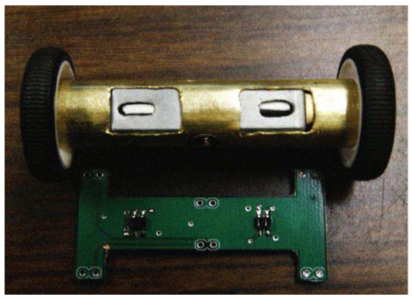


Fig. 2. The motor pack and encoder board. Note the holes in the motor cans with the stripes on the rotors.

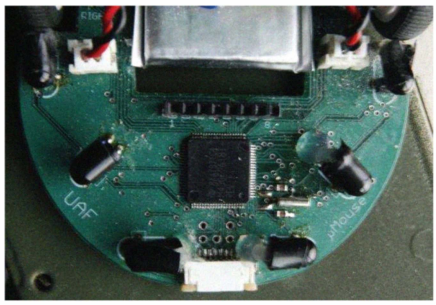


Fig. 3. Infrared Sensor locations on UAF Micromouse.

most successful with these IR sensors placed in six locations: two looking directly forward, two looking diagonally forward at approximately 45 degrees, and one pointed to each side (see Fig. 3). The side looking sensors are used exclusively for mapping walls on the side. The diagonal sensors are used to calibrate the mouse's position in the maze, both laterally and longitudinally, and the front sensors are used for both mapping walls ahead of the mouse as well as keeping it aligned in diagonal corridors. These will all be discussed in the controls section. However, it is important to note in the mechanical section that these sensors all have dead-bands – areas in which the sensor saturates and returns its maximum voltage even though the sensor is not physically as close to the wall as it could be. Because of this dead-band region the placement of the sensors themselves is important. For example, the forward looking sensors are not placed at the front of the mouse; rather they are placed just in front of the wheels, well over an inch from the front of the mouse. This allows us to eliminate most of these dead-band issues. Another consideration is the sensitivity of the sensor. Infrared light exists in most environments, and will interfere with the sensor readings. When buying sensors, it is worth buying a sensor that is less sensitive to the IR light and increasing the output of the LEDs. This allows the mouse to be less sensitive to ambient IR light conditions.

Other issues to consider include the selection of DC motors versus stepper motors, the type and size of the battery, etc.

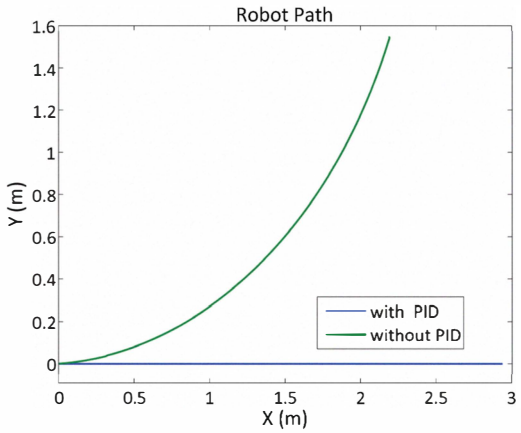


Fig. 4. Mouse Trajectory. Green line is the trajectory without the PID feedback. Blue line is the trajectory with the PID feedback.

B. Controls

1) Simulation: Mathworks' Matlab [7] program is an excellent tool for undergraduate research. This program gives students the opportunity to test different approaches to controls without the need to build the actual hardware. For example, we wanted to investigate how the micromouse would perform if we were to replace our current speed controller of the motors (achieving specific rates of rotation) with a position controller (turning to a specific angle). An undergraduate took up the challenge and created a Simulik model to test this different control paradigm. The model includes a single motor system with its own feedback, which shows that this control model is capable of achieving a specific motor position. The model also includes a simulation of a dual motor system, which incorporates the feedback that includes the error between the two wheels' positions. When both wheels are included in the simulation, it is possible to see that the mouse is capable of moving along a straight trajectory (see Fig. 4).

2) Micromouse Controls: The controls aspect of the micromouse design is where we integrate the hardware and software. This includes running the motors, reading the sensors, and sending data to the mapping algorithm.

The micromouse's movements are broken into smaller, more manageable discrete movements. During the mapping phase there are four main movements: forward one cell, 90° pivot turn right and the same left, and a 180° pivot turn. During the timed run there are about 20 movements that the mouse is capable of making. These extra movements are required so that the mouse may continually be moving forward through the turns rather than stopping and pivoting, as well as allowing it to make movements diagonally in the maze. Each movement requires the accurate reading of all the sensors, as well as closed-loop control of the motors.

Our mice have always used DC motors, which use H-Bridges and Pulse Width Modulated (PWM) control (see Fig. 5). The microcontroller we use, an MSP430 from Texas Instruments [8], has an onboard timer that we may use to create a PWM signal. The important characteristic in the PWM signal is the duty cycle: the percentage of the period of the signal that is at Vcc. This is the amount of time the motor will actually be powered; during the rest of the cycle the motor will be coasting. The larger the duty cycle the motor

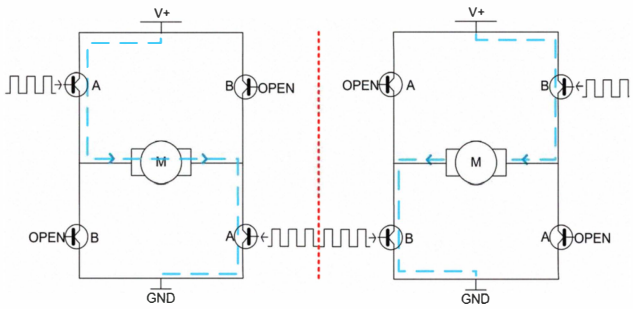


Fig. 5. H-Bridge Control. This controller allows us to control the speed (PWM Duty Cycle, switch set A) and direction (switch set B) of the motors.

receives the more the motor is powered and the less it must coast, and therefore the faster it will rotate. Thus the PWM's duty cycle does the same thing as varying the voltage across the motors - it varies the motor's speed.

Unfortunately, because of minute differences in the motors, in the gears, and in the wheels, applying identical PWM signals to both of the motors on the micromouse will not result in identical motor speeds. For this reason closed loop control is required with micromice. On our mice we have two feedback mechanisms to keep our mouse moving straight and true. Using the data from those two mechanisms we make three comparisons whose outcomes change the PWM duty cycle to each wheel. The two mechanisms are the wheel encoders and the diagonal sensors.

The diagonal sensors are the simplest feedback mechanism. When the mouse is centered in the hallway and aligned, the values the sensors return are identical. If the mouse is not centered or aligned then an error will exist in the sensor values (see Fig. 6). The IR sensor error may be adjusted by a gain value and applied to the motors' PWM signals, adding to one motor and subtracting from the other. However, the astute reader will note that this feedback mechanism only works if walls exist on both sides of the mouse. If one or both walls are absent then the mouse must rely entirely on its internal sensors: the encoders.

The encoders have two functions, first to tell the controller how far the motor has turned, and second to tell it how fast the motor is turning. Determining how fast the motor is turning requires recording the time of each encoder count, and subtracting the previous count time from the current count time. As long as all the edges are equally spaced, the time is an adequate measure of motor speed. There are two types of feedback that can be employed when we know the actual wheel speed. The first feedback is the error between the motor's desired speed and its actual speed. In this case each motor has its own individual feedback. The second feedback is the error between the two motors' speeds. In this case the error is adjusted and added to the slower motor and subtracted from the faster motor. This works to stabilize the wheels regardless of whether they have converged on the desired

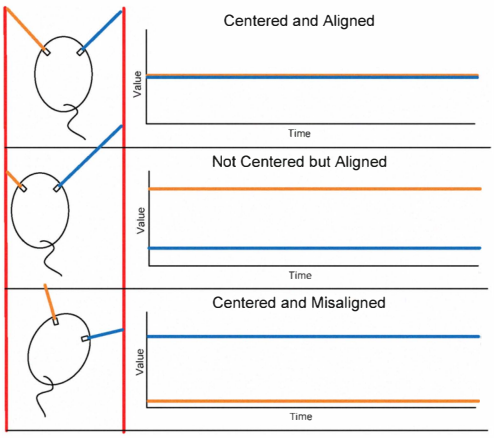


Fig. 6. Diagonal Sensor outputs with the mouse in various states of lateral and angular correctness.

speed. Note that it is possible to apply both types of feedback simultaneously, but under certain circumstances the two will fight each other. This happens whenever one of the motors is below its desired wheel speed but faster than the other motor. The first feedback attempts to push the motor faster while the second feedback attempts to slow it down. The solution is to adjust the gain values of the feedback so that the second feedback has more effect than the first feedback. This ordering of the feedback values is very important, because it is more desirable that the mouse move straight than that it move at a chosen speed.

The infrared sensors are another system that requires some thought about their control. In the mechanical system we considered the sensitivity of the sensors and their ideal locations. In controls it is necessary to consider when to sample them and how to use the data. In our first mouse we discovered that light emitted from one sensor pair will often be detected by the sensor of a different pair. Because of this interference it was necessary to lower the output of each of the LEDs to prevent most of the problems this interference creates. However, in lowering the output of the LEDs, a signal created when a wall was present was sometimes only slightly above the sensor's output noise. Thus the controller would often create false-negatives (not mapping walls that are actually there) because it could not distinguish between the real signal and the noise. This was solved in our second mouse by turning the LEDs on one at a time, and only sampling that light's sensor. This eliminated interference and allowed us to increase the output of each LED to its maximum value, which brought the signal out of the noise and eliminated false-negatives. We rotate them at 2 kHz, which allows the mouse plenty of opportunity to see the walls while not overtaxing the microcontroller.

C. Mapping

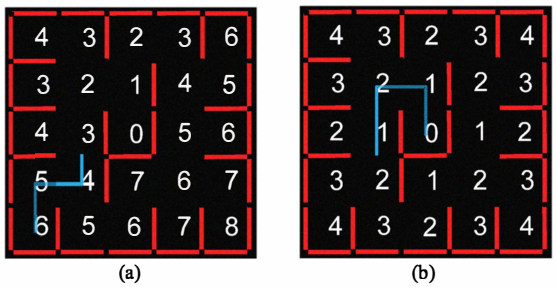
The most commonly used mapping algorithm in micromouse competitions is known as the flood fill algorithm [9]. The flood fill algorithm is a distance-based algorithm in which each cell is assigned a value that represents the number of straight movements the mouse must make to get to the goal square. In each cell the mouse maps the walls in that square and runs the flood-fill algorithm to update all the distance values in all the cells before it can determine which move it must make to get to the center (see Fig. 7). The decision, and subsequent move, is based on which of its neighbors has the lowest value. If there are two neighbors with the lowest value, the mouse will favor the forward direction because it is faster than turning. After a sufficient portion of the maze has been mapped, the mouse can use the distance values of the mapped maze to determine the shortest route to the center. In this case, again, it chooses the route that will have it always moving to the lower valued neighboring cell.

The flood fill algorithm can be processor intensive. On most low power processors, like the MSP430, the flood fill algorithm requires enough time that the mouse either needs to stop in the middle of the cell to run the flood fill and make its decision, or it needs to start making the calculation before the previous movement is completed. This is a price that is paid

for processors such as the MSP430 that have low power consumption and simple design (requiring only a few external capacitors and a crystal). Either way, the mapping algorithm must work together with the control algorithm in deciding when the mouse should sample walls, and when to make the decision for the next move. Excellent animations of the flood fill algorithm in work can be found on the Internet [10].

In our current experiments in the mapping algorithm, we have found that it is not actually necessary to update to maze values in every cell. The mouse may actually run the flood fill one time at the very beginning, and then regardless of where the walls are, the mouse simply chooses to move into the lowest valued neighbor. Because the mouse no longer requires the intense processing to calculate all the distance values, it no longer needs to stop in each cell. The mouse becomes free to move forward continually, which improves the control of the mouse, as stopping always includes a small amount of coasting on each wheel that might misalign the mouse before it actually stops. The only instance that the mouse must recalculate the distance values is if the mouse moves into a dead end. However, the mouse must stop in the dead end to turn around anyway, and therefore, it does not affect the improved control by using this new approach to exploration. Fig. 7b shows the same maze as in Fig. 7a, only without recalculating the distance values before making movements. Notice it still takes the same path.

Depending on the maze configuration, the mouse will not map the entire maze using the flood fill algorithm because it will always try to turn to the center of the maze, rather than branching out to explore. The flood fill algorithm allows the mouse to determine the shortest route only through the portion of the maze that has been mapped. The shortest route, however, may not be the fastest route. If the mouse must make a lot of turns, it will not have the opportunity to accelerate. Long straight stretches do give the mouse that opportunity. Therefore, for the mice that have the ability to accelerate on long straight stretches, it could be advantageous to seek the fastest route rather than the shortest route. Finding the fastest route requires the mouse to map a significant portion of the maze, however. This can be done with several different methods, including a pseudo-random decision algorithm. The algorithm we have begun using is also the flood fill algorithm, but this time we assign multiple goal squares rather than only the center square. Generally, setting the center cell plus about six other uniformly distributed cells as the goal cells is enough



(a) (b)

Fig. 7. A 5x5 maze with the goal square in the center, with (a) and without (b) updated distance values. The mouse's movements follow the blue line.

to map a sufficient portion of the maze to allow the mouse to determine a fast route. Again this depends on the configuration of the maze. Finding the fastest route requires the mouse to make numerous virtual runs through the maze, recording the sequence of random movements. Each movement is assigned a weighting value. The value of the entire run is the summation of all the movements. Each virtual run is analyzed and compared with the run before it. The mouse remembers only the faster of the two runs, and then virtually runs the maze again. After a few hundred runs, the mouse will have a fastest route. Of course this fastest route is compared against the shortest route to ensure it is an option worth taking. Using the option of fastest route instead of shortest route is a technique that should be explored by faster mice.

IV. Contributions to Education

The micromouse, like all robots, is a multi-faceted device with systems covering nearly every major topic in electrical engineering. Following are some examples of how individual systems contribute to university classes. The physical design of the mouse allows students to familiarize themselves with basic physics and mechanics. This is particularly true when selecting the motors, gears, and wheels to provide the correct amount of torque and speed required. Especially for electrical engineering students, this is a topic often guessed at, as it is generally not in the interest of the EE student. As mentioned earlier, however, it is an important system of the micromouse and the more time spent thinking about and truly designing this aspect, the better the micromouse will be. Having an understanding of physics and mechanics is essential to be able to do the design, though.

Selecting the right sensors, designing the circuits to ensure they are working properly, and correctly reading the data is a great review for embedded systems design classes. Sensor integration requires an understanding of the sensor's characteristics, such as accuracy, resolution, precision, drift, sensitivity, linearity, and responsiveness. Also important is the output characteristics, such as the type of output signal, which include Voltage, Serial Peripheral Interface (SPI), Universal Asynchronous Receiver Transmitter (UART), and PWM signals, and any important characteristics of each of these types of signals, such as rise time and fall time for voltages, and baud rate for UART signals.

Most university programs save classes covering microcontrollers until the end of the undergraduate classes. The micromouse, which requires a microcontroller, is an excellent platform for learning the use of a microcontroller. If a student uses micromouse as a class project in a microcontroller based class, then that student will benefit from a project that requires the use of many of the major peripheral functions of the microcontroller, such as clock setup, interrupts, timers, analog-to-digital converter (ADC), general purpose input and output (GPIO), and in some cases the use of communications such as SPI or UART. If however the student participates in micromouse before taking the class, the student will find the class much more understandable and interesting.

Low-voltage power systems are another underappreciated yet highly important system. Generally, every mouse needs two voltage levels: a 3.3 volt rail for the digital electronics, and a 6 to 12 volt rail for the motors. Our first and second generation micromice use a battery pack with two lithium-ion batteries, placed over the motor pack, giving it a total nominal voltage of 7.4 volts. Two buck regulators were used to lower that voltage to the desired 3.3 volts and 6 volts. On the latest mice we used a single 3.7 volt lithium-ion battery and then had a buck regulator to give the 3.3 volts and a boost regulator to give the 6 volts. This allowed us to embed the battery in the bottom board, thus lowering the center of gravity and the moment of inertia. Students in classes such as power electronics could benefit from designing a power system that provides the necessary voltages without being bulky.

Electrical engineers are all required to take at least one class in programming, in which C/C++ and sometimes Matlab are generally covered. Because the language we use to write the programs for the micromouse is C, this experience is a great reinforcement of the concepts of programming.

Obviously the class in which micromouse would be the most useful is control systems. Without understanding the concept of feedback, it is impossible to get the mouse moving straight in the maze using DC motors. Also, control theory can be somewhat abstract, and having a project such as micromouse, in which students can get hands-on experience in controls, strengthens the students' understanding of the theory.

Beyond contributing to class material, the micromouse project helps students pick up some skills that aren't directly taught in class. Teamwork and communication in a team are more important in a project such as micromouse, where each team member is often responsible for different portions of the design and control code. However, because all the systems tie together, each team member must effectively communicate their ideas to one another to ensure fluid merging of the hardware and software systems.

The other skillset is time management. Micromouse is a competition with a fixed competition date. Students must manage their time correctly to ensure that all systems work adequately by the time of the competition. It is not unusual for teams to spend too much time on a single system, leaving another system incomplete by competition time and therefore the mouse fails to complete its task.

Solid time management as well as team organization and communication are essential elements to a successful micromouse, and are important skills for any practicing engineer. Gaining experience with these skill sets early is a great result of working on a micromouse project.

V. Conclusion

The successful micromouse is a merging of many key systems, in both hardware and software. These systems are usually taught, if only briefly, in an undergraduate class. Our school has a long history of providing hands-on experience to students through a large number of undergraduate classes that have a lab component (17 required lab classes in the EE curriculum), and through senior design classes. However, the students that participate in micromouse competitions have a chance to get involved early in their curriculum in an extra-curricular project that combines various aspects of electrical engineering, and to improve their skills if they keep participating throughout their college years. While most students participate in micromouse projects at the undergraduate level (Hauer, Malveaux) some of them continue to use the micromouse platform for their graduate research (Steven Kibler is performing research in the area of self-configuring mobile wireless sensor networks using autonomous micromouse robots). These extra-curricular projects are a valuable experience in any engineering department.

Acknowledgment

Steven Kibler thanks his teammates David Giessel (2007-2008), Devin Boyer (2007-2009), and Andrew Hauer (2009-2010). They all contributed greatly to the pool of micromouse knowledge at our university. D. Raskovic thanks his exchange students from France – David Bory and Bruno Pommier, who created the first micromouse in 2005, as a part of their senior design project in the Embedded Systems Design class.

References

[1] (2010) DARPA Urban Challenge. [Online]. Available: http://ww.darpa.mil/grandchallenge/index.asp
[2] (2010) AUVSI Foundation [Online]. Available: www.auvsi.org
[3] (2010) RoboCup. [Online]. Available: http://www.robocup.org
[4] (2010) U.S. FIRST Robotics. [Online]. Available: http://usfirst.org.
[5] M. Chiodo et al., "Hardware-Software Co-Design of Embedded Systems," *IEEE Micro*, pp. 26-36, August 1995.
[6] Duyao Wang, et al., "A New Method of Infrared Sensor Measurement for Micromouse Control, "*Audio, Language and Image Processing*, pp 784-787, July 2008.
[7] (2010) Mathworks's Matlab. [Online]. Available: http://www.mathworks.com/products/matlab/
[8] (2010) Texas Instruments MSP430 Page. [Online]. Available: http://www.msp430.com.
[9] Mishra, S, et al., "Maze Solving Algorithms for MicroMouse," *Signal Image Technology and Internet Based Systems*, pp 86-93, Nov-Dec 2008.
[10] Steve Benkovic. (2010) Micromouse Info. [Online]. Available: http://www.micromouseinfo.com/intorduction/algorithm.