# Building a Micromouse for an IEEE Student Activities Conference (SAC) Competition

(/uploads/1/1/0/4/11040693/9682897_orig.png?705)

I have worked with the student chapter of IEEE for several years and was the Vice President in my senior year of my undergraduate degree. Being a graduate student doesn't give me as much time to attend events, but I did offer to help the undergraduate team design and build a micromouse for the SAC competition. If you are unsure what a micromouse is, it is a small robot that can solve a maze using some type of maze solving algorithm. The goal of the competition is to solve a maze as quickly as possible. As a group, we decided to use an Arduino Uno microcontroller to control the robot.


(/uploads/1/1/0/4/11040693/1030726_orig.jpg?269)

http://arduino.cc/en/Main/ArduinoBoardUno

A basic micromouse would use continuous servo motors to drive the robot. Additionally, servo motors are much easier to control and are able to make accurate movements, but they tend to be much slower than DC motors. We decided to focus more on speed. Our design used 2 dc motors that came in a kit. The kit included mounting brackets, wheels, and all mounting screws needed.

(/uploads/1/1/0/4/11040693/7877533_orig.jpg?281)

http://www.parallax.com/

To drive these motors we used a DC motor shield designed to work with an Arduino Uno. It allows us to control speed, direction, and frequency of the motors. With the shield we chose, we can control up to two DC motors at once. Both the Arduino and the motor are powered by a 9v battery and are connected together using 9v battery caps.

(/uploads/1/1/0/4/11040693/8553678_orig.jpg?369)

http://www.adafruit.com/

Ping sensors were used to tell the robot where was and give feedback to control the speed of the motors. The ping sensor sends out a ping and measures how long it takes to come back. Based on the time and the speed of the sound wave, it can determine how much distance there is between the sensor and an object. In our case, it detects the maze walls.


(/uploads/1/1/0/4/11040693/9639628_orig.jpg)

http://www.parallax.com

Conventionally in these designs, a single pin sensor is used and is mounted onto a standard servo. The servo motor would then look left, look right and then look straight ahead and find direction to continue in. This takes a lot of time since the servo motor must rotate in several directions before making a decision to move. By using three sensors, we reduced the time it took to make decisions and to determine which direction to take.

## Constructing the Frame

In order to reduce the cost of the project and to enter into the "custom built" category of the competition, (competitors could also purchase an entire micromouse kit that provided step by step instructions to build it) I custom built a frame for the micromouse.

**Step 1:** I started with some scrap plexiglass I had lying around and cut a rectangle piece from it. I then drilled holes and mounted the two dc motor brackets. From here I started to heat up and shape the frame as I pleased.

The first goal was to create an area where the motors could  be housed.


(/uploads/1/1/0/4/11040693/8554987_orig.jpg?393)

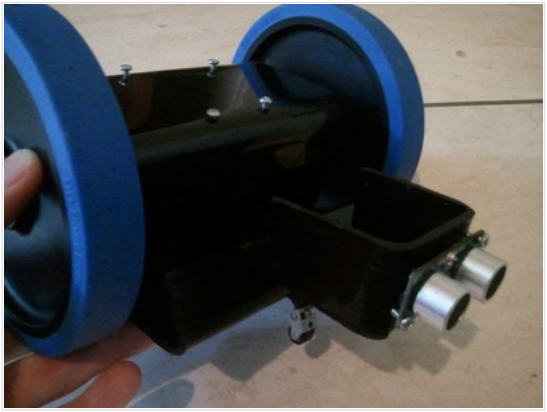
**Step 2:** I then mounted the motors onto the brackets and mounted the wheels onto the motor shafts. I drilled holes and placed screws in the top where the microcontroller and motor shield would be mounted. I curled the front of the frame upward so I had place to mount the ultrasonics and small guiding wheel in the front. The small wheel was taken from a cassette player.


(/uploads/1/1/0/4/11040693/5200135_orig.jpg?403)

Bottom view.

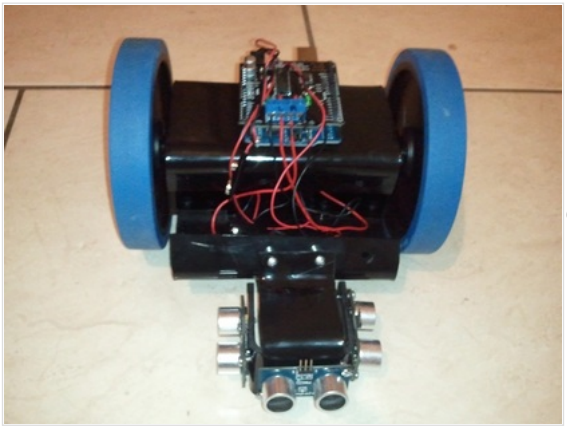

(/uploads/1/1/0/4/11040693/6425019_orig.jpg?406)

**Step 3:** Using the leftover scraps I designed a mounting bracket for the three ultrasonics. I designed it much like a cardboard box is designed keeping in mind how I would bend each side. This first design needed to be modified later due to the ultrasonics being too high and pinging above the mazes wall height limit. You will see in later pictures that the mount was flipped so the sensors were closer to the ground.



(/uploads/1/1/0/4/11040693/268206_orig.jpg?403)

**Step 4:** Once the frame was complete, all of the hardware was mounted and wired. Each dc motor has a positive and negative terminal designated on the dc motor shield. One 9v battery cap was connected to the voltage source input of the motor shield which was used to drive the dc motors. Another 9v battery cap was soldered to a barrel cable that could be plugged into the Arduino Uno for power. We later used replaced the 9v batteries with an RC car battery pack that was capable of recharging to save time and money. Now that the micromouse was build, the next step was to program it.



(/uploads/1/1/0/4/11040693/4743802_orig.jpg?418)

## Programming the Micromouse

The programming was done in the Arduino IDE. I started with some base code provided with the dc motor shield and adapted the code from there. The code was broken up into several portions. The setup portion sets all the pins that are being used for both the dc motor shield and the ping sensors.

```
int leftping = 2;  //left ping pin
int frontping = 3;  //front ping pin
int rightping = 8;  //right ping pin
int M2Dirpin = 4; // Motor 2 Direction control
int M2Spdpin = 5; // Motor 2 PWM control
int M1Dirpin = 6; // Motor 1 PWM control
int M1Spdpin = 7; // Motor 1 Direction control
int M1Speed = 100; // PWM value (0-255)
int M2Speed = 100; // PWM value (0-255)

void setup() {
  int i;
  for(i=5;i<=8;i++) //For Arduino Motor Shield
  pinMode(i, OUTPUT); //set pin 4,5,6,7 to output mode

  Serial.begin(9600); //go to tools/serial moniter
}

void loop()
{
  long leftduration, leftinches, leftcm, fronttduration,
       frontinches, frontcm, rightduration, rightinches, rightcm;

  char state;  //'l' left, 'r' right, 'f' forward, 's' stop
```

(/uploads/1/1/0/4/11040693/2157362_orig.jpg)

The ping sensor code was acquired from the provided examples in the Arduino IDE help file and adapted to work with 3 ping sensors. The sensors ping in a sequence, one after the other, and do this repeatedly.

```
//PING SENSORS
//Left Sensor
pinMode(leftping, OUTPUT);
digitalWrite(leftping, LOW);
delayMicroseconds(2);
digitalWrite(leftping, HIGH);
delayMicroseconds(5);
digitalWrite(leftping, LOW);
pinMode(leftping, INPUT);
leftduration = pulseIn(leftping, HIGH);

//Front Sensor
pinMode(frontping, OUTPUT);
digitalWrite(frontping, LOW);
delayMicroseconds(2);
digitalWrite(frontping, HIGH);
delayMicroseconds(5);
digitalWrite(frontping, LOW);
pinMode(frontping, INPUT);
fronttduration = pulseIn(frontping, HIGH);

//Right Sensor
pinMode(rightping, OUTPUT);
digitalWrite(rightping, LOW);
delayMicroseconds(2);
digitalWrite(rightping, HIGH);
delayMicroseconds(5);
digitalWrite(rightping, LOW);
pinMode(rightping, INPUT);
rightduration = pulseIn(rightping, HIGH);
```

(/uploads/1/1/0/4/11040693/142908_orig.jpg)

The time it takes for the ping to be sent out and bounce off of an object is converted using a function that converts the duration into inches.

```
// Convert the time into a distance
//Left
leftinches = microsecondsToInches(leftduration);
//leftcm = microsecondsToCentimeters(duration);

//Front
frontinches = microsecondsToInches(fronttduration);
//frontcm = microsecondsToCentimeters(fronttduration);

//Right
rightinches = microsecondsToInches(rightduration);
//rightcm = microsecondsToCentimeters(rightduration);
```

(/uploads/1/1/0/4/11040693/4113349_orig.jpg)

For trouble shooting, I wrote a small portion of code that can be uncommented and ran as part of the program. When the code is ran, the built in serial monitor of the Arduino IDE can be used to see the live distances each of the ping sensors is reading. A delay can be used to slow down how fast the returned value is refreshed.

```
// Uncomment to display ping values in serial monitor
// To use serial monitor click tools/serialmonitor
// To uncomment, highlight code below and right click to
// select uncomment

  Serial.print("Left: ");
  Serial.print(leftinches);
  Serial.print(", Front: ");
  Serial.print(frontinches);
  Serial.print(", Right: ");
  Serial.print(rightinches);
  Serial.println();
//  delay(100);
```

(/uploads/1/1/0/4/11040693/8538106_orig.jpg)

This section of code was left for the undergraduate students to complete. I wrote a basic program to confirm that all of the hardware was communicating properly with the software. As you can see, the program is set up so that the current state is updated with the characters "l", "r", "f", or "s" which stand for left, right, front, and stop. These commands can be used to control the robot and tell it which direction to go. It was up to the undergraduate students to determine what type of maze solving algorithm they wanted to use.

```
//MAZE CONTROL PROGRAM HERE

if (leftinches < 2)
  state = 'l';
if (rightinches < 2)
  state = 'r';
if (frontinches < 2)
  state = 'f';
if (leftinches > 2 && rightinches > 2 && frontinches > 2)
  state = 's';
```

(/uploads/1/1/0/4/11040693/4453657_orig.jpg)

The motor control was done using a case statement controlled by the state characters discussed above. This made controlling the robot easy be simply sending the direction they wished the robot to drive in. The speed can also be updated at anytime.

```
//MOTOR CONTROL
//The speed control is achieved through the conventional PWM
//which can be obtained from Arduinoâ€™s PWM output Pins 5 and 6.
//The enable/disable function of the motor control is signalled
//by Arduino Digital Pins 7 and 8.

//Digital 4: Motor 2 Direction control
//Digital 5: Motor 2 PWM control
//Digital 6: Motor 1 PWM control
//Digital 7: Motor 1 Direction control
//set pwm control, 0 for stop, and 255 for maximum speed
  switch(state)
    {
      case 'f': //forward
         Motor1(M1Speed,true);
         Motor2(M2Speed,true);
         break;

      case 'l'://turn left
         Motor1(M1Speed,false);
         Motor2(M2Speed,true);
         break;

      case 'r'://turn right
         Motor1(M1Speed,true);
         Motor2(M2Speed,false);
         break;

      case 's'://stop
         Motor1(0,false);
         Motor2(0,false);
         break;
    }

 //END OF PROGRAM
```

(/uploads/1/1/0/4/11040693/1044866_orig.jpg)

Here are the function that get called on to send the correct control signals to the dc motor shield.

```
void Motor1(int pwm, boolean reverse)
  {
    analogWrite(M1Spdpin,pwm);
    if(reverse)
       {
```

```
        digitalWrite(M1Dirpin,HIGH);
      }
    else
      {
        digitalWrite(M1Dirpin,LOW);
      }
    }

void Motor2(int pwm, boolean reverse)
  {
    analogWrite(M2Spdpin,pwm);
    if(reverse)
      {
        digitalWrite(M2Dirpin,HIGH);
      }
    else
      {
        digitalWrite(M2Dirpin,LOW);
      }
  }

long microsecondsToInches(long microseconds)
{
   return microseconds / 74 / 2;
}

long microsecondsToCentimeters(long microseconds)
{
  return microseconds / 29 / 2;
}
```

(/uploads/1/1/0/4/11040693/6756132_orig.jpg)

The undergraduate team programmed a maze algorithm to the best of their abilities and competed in the competition. They didn't win, but they learned a lot.

### Download Code Here

**micromouse.ino** (/uploads/1/1/0/4/11040693/micromouse.ino)
**Download File (/uploads/1/1/0/4/11040693/micromouse.ino)**