International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Pp-92-100 ISSN: 2249-6645

Maze Solving Robot Using Freeduino and LSRB Algorithm

J.Arun Pandian[1], R.Karthick[2], B.Karthikeyan[3]
[1,3]Student, Department of Information Technology, Arunai Engineering College, Anna University, Velu Nagar, Tiruvannamali-606 603, T.N, India
[2]Assistant Professor, Department of Information Technology, Arunai Engineering College, Anna University, Velu Nagar, Tiruvannamali-606 603, T.N, India

ABSTRACT: *This paper "MAZE SOLVING ROBOT USING FREEDUINO AND LSRB ALGORITHM" deals with the development maze robot using simple circuits. A MSR (maze solving robot) is a clever little gadget with a silicon brain that finds its way through an arbitrary maze. It competes against other members of its species, racing as fast as it can. Here MSR is an electro-mechanical device, typically consisting of three main subsystems. They are drive system, an array of sensors, and the control system.*

The drive system consists of a mechanical chassis, a set of motors, gears and wheels. The chassis is usually designed to operate like a wheel-chair, with two large drive wheels on either side of a lightweight aluminum frame. The chassis is built to house the motors, gear system, batteries, and circuitry, and must also be small enough to maneuver within the constraints of the maze.

The control system is series of circuit boards functioning as the brain of the critter. The control system runs a maze solving algorithm (LSRB) based on the information received by the CPU (FREEDUINO BOARD) from the sensors. Typically the first several runs through the maze will be a search sequence, in which the mouse learns the maze and stores it in memory. It must then calculate the fastest possible path, which it will repeatedly run trying to achieve successively faster times.

The final sub-system is the sensors. They report to the CPU the current state of the surroundings where the walls and paths are. These are usually either infrared sensor which picks up light reflected light of the track.
The main objective is to achieve the fastest maze running time and easily find the goal.

I. INTRODUCTION

A **maze** is a tour puzzle in the form of a complex branching passage through which the solver must find a route. In everyday speech, both maze and labyrinth denote a complex and confusing series of pathways, but technically the maze is distinguished from the labyrinth, as the labyrinth has a single through-route with twists and turns but without branches, and is not designed to be as difficult to navigate.

A MSR is placed in the maze below and is programmed to move at random. There are basically 2 steps.

- The first is to drive through the maze and find the end of it.
- The second is to optimize that path so your robot can travel back through the maze, but do it perfectly without going down any dead ends.

The actual final score of the robot is primarily a function of the total time in the maze and the time of the fastest run. We use a technique called the **LSRB**.
This LSRB algorithm can be simplified into these simple conditions:
-If you can turn left then go ahead and turn left, -else if you can continue driving straight then drive straight,
-else if you can turn right then turn right. - If you are at a dead end then turn around.

The expunction of LSRB is shown below:
L- Left
R- Right
S- Straight
B- Turning around (Back).

The robot has to make these decisions when at an intersection. An intersection is any point on the maze where you have the opportunity to turn. If the robot comes across an opportunity to turn and does not turn then this is consider going straight. Each move taken at an intersection or when turning around has to be stored.

II. OVERVIEW OF MAZE SOLVING ROBOT

A maze is a tour puzzle in the form of a complex branching passage through which the solver must find a route. In everyday speech, both maze and labyrinth denote a complex and confusing series of pathways, but technically the maze is distinguished from the labyrinth, as the labyrinth has a single through-route with twists and turns but without branches, and is not designed to be as difficult to navigate. The pathways and walls in a maze or labyrinth are fixed (pre-determined) – puzzles where the walls and paths can change during the game are categorized as *tour puzzles*. The Cretan labyrinth is the oldest known maze.

The robot consists of Analog sensor array (5 Reflection sensors), the freeduino board must stay centered within the base wood to prevent compounding errors in distance and damages with the obstacles.

Controlling the motors is responsible for safely moving the wireless robot through the internet. There are four motors with a RPM of 120 per min, because to carry a weight of about 250gm. (board and battery)

In order to ensure that the robot is not slipping and to avoid collision with the obstacles, the motors are controlled with Reflection sensor. The shaft encoders (built into the motors) read in the speed of the wheels and send this data to the motor control code that adjusts the speed of each wheel. And according to our needs, we can make use of it to control the speed of the motors.

The chases need to carry the board, circuits, sensors and batteries. And the chases are designed to turn 90 and 180.

If the and track branching or ending are available in the routing make the decision based on the algorithms and the detected final path is to be stored in to the registers in the atmega8/168/328. The programs are transfers using uart or usb data cable.

III. COMPONENT DESCRIPTIONS

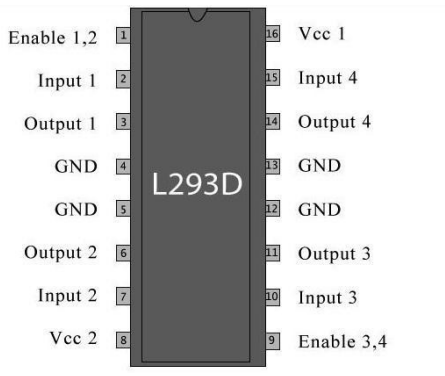
3.1 FREEDUINO BOARD:

The Freeduino is an open source microcontroller board based on the ATmega8/168/328.It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. By plugging in a USB cable into the Freeduino, we can be able to upload the programs to the microcontroller. And also by providing the External adapter to the Freeduino, we can activate the Freeduino board.

3.2 ANALOG SENSOR ARRAY:

The reflectivity of infrared light varies with the color and distance of the reflecting surface. According to this principle, Grove - Infrared Reflective Sensor utilizes a RPR220 reflective photo sensor module to detect color and distance. When a light-colored object approaches, the signal intensity received by infrared reflective sensor increases and the indicator LED on board turns red. When a dark-colored object approaches, the intensity decreases and the LED turns off. This sensor is a basic and widely used part in applications such as line-following cars, rotary speed detection, auto data logging on utility meters or other situations where color or distance contrast is sharp.

3.3 MOTOR DRIVER SYSTEM



The robot uses DC electric motors, and in order to control and drive them, a system incorporating a power converter/regulator is needed. The power from the chassis(External) battery must be translated to the 12V needed by the DC motors, and regulated in such a way as to provide speed, acceleration, and directional control to the robot. L293D is a dual H-Bridge motor driver, so with one IC we can interface two DC motors which can be controlled in both clockwise and counter clockwise direction. L293D has output current of 600mA and peak output current of 1.2A per channel. Moreover for protection of circuit from back EMF output diodes are included within the IC. The output supply (VCC2) has a wide range from 4.5V to 36V, which has made L293D a best choice for DC motor driver.

Specifications

- Supply Voltage Range 12V
- 36V Output current capability per driver

- Separate Input-logic supply
- It can drive small DC-geared motors, bipolar stepper motor.
- Pulsed Current 1.2-A Per Driver
- Thermal Shutdown
- High-Noise-Immunity Inputs

Applications

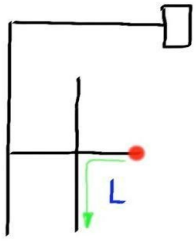
- DC and stepper motor drives
- Position and velocity servomechanisms

IV. LSRB ALGORITHMS:

The LSRB algorithms is processed calculating the shortest path of the robot the given below The LSSRB process the two steps are 1.searching, 2.travelling.

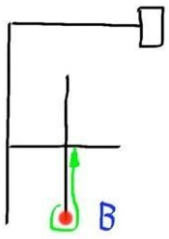
At first we see the searching process of the LSRB Algorithms. the diagrams is the sample of the processing the LSRB.

The first diagram is denotes the starting point and the first decision based on the LSRB priority so the robot is choosing the left side. Second decision Value has been stored in the register. The first decision diagram is shown below,



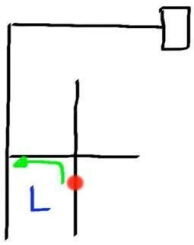
Path = L

In the second diagrams is denoted the after processing of the first decision and the next decision is taken to the back side because there are no more options are available and the second decision Value has been stored in the register. The second decision diagram is shown below,



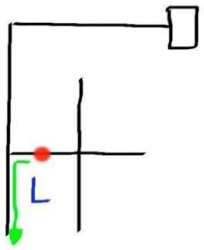
Path = LB

In the Third diagrams is denoted the after processing of the second decision and the next decision is taken to the left side because there are no more options are available and the second decision Value has been stored in the register. The third decision diagram is shown below,



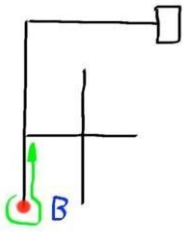
Path = LBL

In the Fourth diagrams is denoted the after processing of the third decision and the next decision is taken to the left side because there are no more options are available and the fourth decision Value has been stored in the register. The fourth decision diagram is shown below,



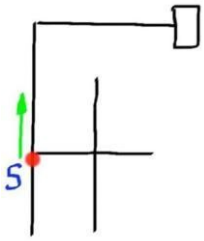
Path= LBLL

In the fifth diagrams is denoted the after processing of the fourth decision and the next decision is taken to the back side because there are no more options are available and the fifth decision Value has been stored in the register. The fifth decision diagram is shown below,



Path = LBLLB

In the sixth diagrams is denoted the after processing of the fifth decision and the next decision is taken to the straight side because there are no more options are available and the sixth decision Value has been stored in the register. The sixth decision diagram is shown below,



Path = LBLLBS

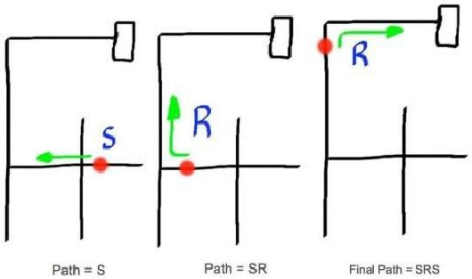
In the seventh diagrams is denoted the after processing of the sixth decision and the next decision is taken to the right side because there are no more options are available and the seventh decision Value has been stored in the register. The seventh decision diagram is shown below, After the seventh step the searching processes will be finished and the finally the "LBLLBSR" value is stored in to the register.

Second process is the travelling processes. This process is performing the robot to go the destination without any searching and using shortest path.

The travelling is simplify the register value using the following equation and shorted and follow the value of the register the equations are shown below,

LBR = B
LBS = R
RBL = B
SBL = R
SBS = B
LBL = S

The register value is shorted from using the eqations and the final register value is "SRR".diagram is shown belows



Path = S Path = SR Final Path = SRS

V. CONNECTIONS AND INTERFACING:

In the Freeduino board solder a 9 pin strip of female headers on the left side of the board from pins labeled "5v" to "A0". This will mate with the sensor plug later. Solder a 4 pin strip of female headers on the right side of the board from pins labeled "D5" to "D8". These pins will be used to control the motor controller. Finally solder a 2 pin strip of female headers on the front right of the board on the GND and 5V pins. These will supply power to the motor controller.

You can ignore the fact that the motor controller and top deck in the photos. We will get to those.

The sensor array bolts onto the front of the robot using #2 bolts. The pin on the far left of the sensor is GND and gets wired to GND on the Freeduino. The second most left pin is Vcc and it gets wired to the 5V pin on the Freeduino. The pins labeled 6-1 on the analog sensor get wired to the Freeduino analog pins from 5-0. So pin 6 on the sensor gets wired to the Freeduino analog pin 5, pin 5 of the sensor gets wired to the Freeduino analog pin 4, etc.

Next wire the mtor controller up. I have a labeled diagram in the photos. Looking at the last photo, I have the motor on the bottom of the picture connected to the what I have labeled as "M1-A" and "M1-B". These are the outputs of the motor controller for the first motor. I have the motor in the top of the last photo connected to what I have labeled as "M2-A" and "M2-B". These are the outputs of the motor controller for the second motor. Now time for the inputs of the motor controller. The Freeduino digital output 7 gets wired to what I have labeled as "In 1A". This is the first input of the first motor. The Freeduino digital output 6 gets wired to what I have labeled as "In 1B". This is the second input of the first motor. The Freeduino digital output 5 gets wired to what I have labeled as "In 2A". This is the first input of the second motor. The Freeduino digital output 8 gets wired to what I have labeled as "In 2B". This is the second input of the second motor.Finally Power and ground get connected to the Freeduino 's 5v and Gnd pins at the front of the board. In the picture I have the wires attached but not plugged in yet.As you can see I also went ahead and used some of the #2 nuts and bolts to bolt down the ball caster and Freeduino.

VI. THE PROGRAM:

The Embedded C language program is used to code the LSRB Algorithms on the FREEDUINO board and the sketch language is also used to develop the Freeduino programming code, compile, executing and transfer the program we use Arduino software application.

Given diagrm is sample programe in Arduino software.

The LSRB program complete code is to be shown below and the code is edit, compile and transfer the program using Arduino application the program sample is shown below,

```
#define
leftCenterSensor        3
#define
leftNearSensor          4
#define leftFarSensor   5

#define
rightCenterSensor       2
#define
rightNearSensor         1
#define rightFarSensor  0
#define
leftMotor1              7
#define
leftMotor2              6

#define rightMotor1 5 #define rightMotor2 8 #define led 13
int leftCenterReading; int leftNearReading; int leftFarReading;
int rightCenterReading; int rightNearReading; int rightFarReading; int leftNudge;
int replaystage; int rightNudge; char path[30] = {}; int pathLength; int readLength; void setup(){
pinMode(leftCenterSensor, INPUT); pinMode(leftNearSensor, INPUT); pinMode(leftFarSensor, INPUT);
pinMode(rightCenterSensor, INPUT); pinMode(rightNearSensor, INPUT); pinMode(rightFarSensor, INPUT);
pinMode(leftMotor1, OUTPUT); pinMode(leftMotor2, OUTPUT); pinMode(rightMotor1, OUTPUT);
pinMode(rightMotor2, OUTPUT);
pinMode(led, OUTPUT); digitalWrite(led, HIGH); delay(1000);
}
void loop(){ readSensors();
if(leftFarReading<200 && rightFarReading<200 && (leftCenterReading>200 || rightCenterReading>200) ){
straight();
}
else{
leftHandWall();
```

```
}
}
void readSensors(){
leftCenterReading = analogRead(leftCenterSensor); leftNearReading = analogRead(leftNearSensor); leftFarReading =
analogRead(leftFarSensor); rightCenterReading = analogRead(rightCenterSensor); rightNearReading =
analogRead(rightNearSensor); rightFarReading = analogRead(rightFarSensor);
}
void leftHandWall(){
if(leftFarReading>200 && rightFarReading>200){ digitalWrite(leftMotor1, HIGH); digitalWrite(leftMotor2, LOW);
digitalWrite(rightMotor1, HIGH); digitalWrite(rightMotor2, LOW);
delay(170);
readSensors();
if(leftFarReading>200 || rightFarReading>200){ done();
}
if(leftFarReading<200 && rightFarReading<200){ turnLeft();
}
}
if(leftFarReading>200){ digitalWrite(leftMotor1, HIGH); digitalWrite(leftMotor2, LOW); digitalWrite(rightMotor1,
HIGH); digitalWrite(rightMotor2, LOW); delay(170);
readSensors();
if(leftFarReading<200 && rightFarReading<200){ turnLeft();
}
else{
done();
}
}
if(rightFarReading>200){
digitalWrite(leftMotor1, HIGH);    delay(2000);
digitalWrite(leftMotor2, LOW);     replay();
digitalWrite(rightMotor1, HIGH); }
digitalWrite(rightMotor2, LOW);
delay(30);
readSensors();    void turnLeft(){
if(leftFarReading>200){   while(analogRead(rightCenterSensor)>200||analogRead(l
delay(140);         eftCenterSensor)>200){
readSensors();    digitalWrite(leftMotor1, LOW);
if(rightFarReading>200 && leftFarReading>200){  digitalWrite(leftMotor2, HIGH);
done();  digitalWrite(rightMotor1, HIGH);
}          digitalWrite(rightMotor2, LOW);
else{      delay(2);
turnLeft();         digitalWrite(leftMotor1, LOW);
return;  digitalWrite(leftMotor2, LOW);
}          digitalWrite(rightMotor1, LOW);
}          digitalWrite(rightMotor2, LOW);
delay(140);         delay(1);
readSensors();    }
if(leftFarReading<200 && leftCenterReading<200 && while(analogRead(rightCenterSensor)<200){
rightCenterReading<200 && rightFarReading<200){digitalWrite(leftMotor1, LOW);
turnRight();        digitalWrite(leftMotor2, HIGH);
return;  digitalWrite(rightMotor1, HIGH);
}          digitalWrite(rightMotor2, LOW);
path[pathLength]='S';      delay(2);
pathLength++;    digitalWrite(leftMotor1, LOW);
if(path[pathLength-2]=='B'){        digitalWrite(leftMotor2, LOW);
shortPath();        digitalWrite(rightMotor1, LOW);
}          digitalWrite(rightMotor2, LOW);
straight();         delay(1);
}          }
readSensors();    if(replaystage==0){
if(leftFarReading<200 && leftCenterReading<200 && path[pathLength]='L';
rightCenterReading<200 && rightFarReading<200 &&pathLength++;
leftNearReading<200 && rightNearReading<200){ if(path[pathLength-2]=='B'){
```

```
turnAround();     shortPath();
}          }
}          }
void done(){       }
digitalWrite(leftMotor1, LOW);     void turnRight(){
digitalWrite(leftMotor2, LOW);     while(analogRead(rightCenterSensor)>200){
digitalWrite(rightMotor1, LOW);    digitalWrite(leftMotor1, HIGH);
digitalWrite(rightMotor2, LOW);    digitalWrite(leftMotor2, LOW);
replaystage=1;     digitalWrite(rightMotor1, LOW);
path[pathLength]='D';     digitalWrite(rightMotor2, HIGH);
pathLength++;    delay(2);
while(analogRead(leftFarSensor)>200){     digitalWrite(leftMotor1, LOW);
digitalWrite(led, LOW);    digitalWrite(leftMotor2, LOW);
delay(150);          digitalWrite(rightMotor1, LOW);
digitalWrite(led, HIGH);  digitalWrite(rightMotor2, LOW);
delay(150);          delay(1);
}          }
while(analogRead(rightCenterSensor)<200){          digitalWrite(leftMotor1, LOW);
digitalWrite(leftMotor1, HIGH);    digitalWrite(leftMotor2, LOW);
digitalWrite(leftMotor2, LOW);     digitalWrite(rightMotor1, HIGH);
digitalWrite(rightMotor1, LOW);    digitalWrite(rightMotor2, LOW);
digitalWrite(rightMotor2, HIGH); delay(7);
delay(2);          return;
digitalWrite(leftMotor1, LOW);     }
digitalWrite(leftMotor2, LOW);     digitalWrite(leftMotor1, HIGH);
digitalWrite(rightMotor1, LOW);    digitalWrite(leftMotor2, LOW);
digitalWrite(rightMotor2, LOW);    digitalWrite(rightMotor1, HIGH);
delay(1);          digitalWrite(rightMotor2, LOW);
}          delay(4);
while(analogRead(leftCenterSensor)<200){digitalWrite(leftMotor1, LOW);
digitalWrite(leftMotor1, HIGH);    digitalWrite(leftMotor2, LOW);
digitalWrite(leftMotor2, LOW);     digitalWrite(rightMotor1, LOW);
digitalWrite(rightMotor1, LOW);    digitalWrite(rightMotor2, LOW);
digitalWrite(rightMotor2, HIGH); delay(1);
delay(2);
digitalWrite(leftMotor1, LOW);     }
digitalWrite(leftMotor2, LOW);
digitalWrite(rightMotor1, LOW);    void turnAround(){
digitalWrite(rightMotor2, LOW);    digitalWrite(leftMotor1, HIGH);
delay(1);          digitalWrite(leftMotor2, LOW);
}          digitalWrite(rightMotor1, HIGH);
if(replaystage==0){          digitalWrite(rightMotor2, LOW);
path[pathLength]='R';     delay(150);
pathLength++;    while(analogRead(leftCenterSensor)<200){
if(path[pathLength-2]=='B'){       digitalWrite(leftMotor1, LOW);
shortPath();       digitalWrite(leftMotor2, HIGH);
}          digitalWrite(rightMotor1, HIGH);
}          digitalWrite(rightMotor2, LOW);
}          delay(2);
void straight(){    digitalWrite(leftMotor1, LOW);
if(analogRead(leftCenterSensor)<200){      digitalWrite(leftMotor2, LOW);
digitalWrite(leftMotor1, HIGH);    digitalWrite(rightMotor1, LOW);
digitalWrite(leftMotor2, LOW);     digitalWrite(rightMotor2, LOW);
digitalWrite(rightMotor1, HIGH); delay(1);
digitalWrite(rightMotor2, LOW);   }
delay(2);          path[pathLength]='B';
digitalWrite(leftMotor1, HIGH);    pathLength++;
digitalWrite(leftMotor2, LOW);     straight();
digitalWrite(rightMotor1, LOW);    }
digitalWrite(rightMotor2, LOW);    void shortPath(){
delay(7);          int shortDone=0;
return;  if(path[pathLength-3]=='L' && path[pathLength-
}          1]=='R'){
```

```
if(analogRead(rightCenterSensor)<200){    pathLength-=3;
digitalWrite(leftMotor1, HIGH);    path[pathLength]='B';
digitalWrite(leftMotor2, LOW);     shortDone=1;
digitalWrite(rightMotor1, HIGH); }
digitalWrite(rightMotor2, LOW);   if(path[pathLength-3]=='L' && path[pathLength-
delay(2);              1]=='S' && shortDone==0){

pathLength-=3;
path[pathLength]='R';
shortDone=1;
}
if(path[pathLength-3]=='R' && path[pathLength-
1]=='L' && shortDone==0){
pathLength-=3;
path[pathLength]='B';
shortDone=1;
}
if(path[pathLength-3]=='S' && path[pathLength-
1]=='L' && shortDone==0){
pathLength-=3;
path[pathLength]='R';
shortDone=1;
}
if(path[pathLength-3]=='S' && path[pathLength-
1]=='S' && shortDone==0){
pathLength-=3;
path[pathLength]='B';
shortDone=1;
}
if(path[pathLength-3]=='L' && path[pathLength-
1]=='L' && shortDone==0){
pathLength-=3;
path[pathLength]='S';
shortDone=1;
}
path[pathLength+1]='D';
path[pathLength+2]='D';
pathLength++;
}
void replay(){
readSensors();
if(leftFarReading<200 && rightFarReading<200){
straight();
}
else{
if(path[readLength]=='D'){
digitalWrite(leftMotor1, HIGH);
digitalWrite(leftMotor2, LOW);
digitalWrite(rightMotor1, HIGH);

digitalWrite(rightMotor2, LOW);

delay(100);
digitalWrite(leftMotor1, LOW);

digitalWrite(leftMotor2, LOW);

digitalWrite(rightMotor1, LOW);

digitalWrite(rightMotor2, LOW);

endMotion();
}
digitalWrite(leftMotor2, LOW);
digitalWrite(rightMotor1, HIGH);
digitalWrite(rightMotor2, LOW);
delay(150);
turnLeft();
}
if(path[readLength]=='R'){
digitalWrite(leftMotor1, HIGH);
digitalWrite(leftMotor2, LOW);
digitalWrite(rightMotor1, HIGH);
digitalWrite(rightMotor2, LOW);
delay(150);
turnRight();
}
if(path[readLength]=='S'){
digitalWrite(leftMotor1, HIGH);
digitalWrite(leftMotor2, LOW);
digitalWrite(rightMotor1, HIGH);
digitalWrite(rightMotor2, LOW);
delay(150);
straight();
}
readLength++;
}
replay();
}
void endMotion(){
digitalWrite(led, LOW);
delay(500);
digitalWrite(led, HIGH);
delay(200);
digitalWrite(led, LOW);
delay(200);
digitalWrite(led, HIGH);
delay(500);
endMotion();
}
```

after the program click compile and if any error occurred just currect it and again compile and if no error then we click the tools menu and select typeof board and port and next to click the upload button.

These are the steps to code compile and transfer the

program from computer to Freeduino board.

if(path[readLength]=='L'){
digitalWrite(leftMotor1, HIGH);

VII. CONCLUSION

Maze-solving involves Control Engineering and Artificial Intelligence. Using a good algorithm can achieve the high efficiency of finding the shortest path. The proposed maze-solving algorithm works better and has short searching time and low spacecomplexity, and it is significant for robot's finding path in some areas like maze-solving.

REFERENCES

[1] http://ewh.ieee.org/reg/1/sac/Main/RegionalEvents /StudentConf/MicromouseRules.pdf.
[2] Kazerouni B.H.Moradi."Variable Priorities in Maze-Solving Algorithms for Robot's Movement",Seville:Proceedings IEEE International Conference on Industrial Informatics. vol. 6,pp181-185,2003.
[3] Tondra De, Dfew Hall. "The Inception of Chedda: A detailed design and analysis of Micromouse",University of Nevada, Las Vegas Senior Design. vol. 5, pp39-44, 2004.
[4] Huo Zhe Fu. "A Design and Realize for Micromouse's Mazesolving", Microcomputer Appfications.vol 9,pp59-62. 2008.
[5] Jin Liang Fang, Yi Jun Zhou. "Micormouse Based on ARM of IEEE Standard", Machine Building and Automation.vol.5,pp99-101,2008.
[6] UK Micromouse contest, held each year in United Kingdom.
[7] Kazuo Sugihara , John Smith. "Genetical algorithms for adaptive motion planning of anautonomous mobile robots" ,Problems IEEE Trans.USA:SIM,19997.
[8] MicroMOuse, California State University at Northridge, http//homepage.mac.com/SBenkovic/MicroMouse/in dex.html.
[9] Dieguez A R,Sanz R,Lopez J. "Deliberative on-line local path plannning for autonomous mobile robot",Journal of Intelligent and Robotic Systems, vol. 37,pp1-19,2003.
[10] http://www.micromouseonline.com.
[11] MicroMousel02 IEEE standard Micromouse data manualm, 2007.