TECHNICAL REPORT MICROMOUSE

Link: https://github.com/emstef/Micromouse

EKA OKTAVIANI
1103194044

SINOPSIS

Proyek ini merupakan sebuah kompetisi yang terdapat robot kecil menyerupai tikus pada simulator 3D webots ini. Robot kecil yang menyerupai tikus ini dapat meyelesaikan labirin 16x16 blok.

Untuk mencapai tujuan ini, robot menggunakan empat prinsip dasar: lokalisasi, pemetaan, perencanaan jalur, dan kontrol gerak. Saat robot bergerak di dalam labirin, dan menggunakan serangkaian sensor untuk menghindari rintangan dan merekam posisi mereka di labirin menggunakan referensi posisi awalnya. Secara bersamaan, peta labirin yang direkam digunakan untuk menentukan kemungkinan jalur ke pusat setiap kali masuk ke sel berikutnya.

APA ITU MICROMOUSE?

Micromouse adalah kompetisi di mana robot kecil akan berlomba ke tengah labirin. Lari pertama adalah lari pencarian dan sisanya lari lari. Mouse tidak memiliki informasi lain selain posisi awalnya (yang selalu berada di sudut labirin dengan sisi kirinya menghadap bingkai labirin), ukuran labirin, dan harus mencapai pusatnya. Untuk menghitung jalur terbaik ke pusat, proses pertama digunakan untuk memetakan labirin dan ini disebut pencarian. Saat tikus berlomba lari, ia mungkin mencoba menjelajahi labirin dalam perjalanan kembali ke sel awal. Labirin dirancang dengan cermat untuk menonjolkan agen otonom yang lebih canggih yang mungkin memanfaatkannya untuk meminimalkan waktu kerja mereka. Perhatikan bahwa jalur terbaik ke tengah belum tentu yang terpendek karena mouse dapat melaju lebih cepat saat tidak harus berbelok. Setiap acara Micromouse kurang lebih memiliki seperangkat aturan yang sama. Perbedaan ditemukan dalam sistem penilaian, terutama untuk mempromosikan perilaku otonom yang lebih maju.

MATERIAL

WEBOTS

Webots merupakan perangkat lunak (software) yang digunakan sebagai model, program dan simulasi suatu robot bergerak (mobile robot).
Webots digunakan untuk membuat sebuah perancangan robot yang berbentuk sebuah software simulator. Webots memiliki antarmuka yang sederhana, mendukung bahasa : C / C ++, Java, Python, Urbi, MATLAB dan adanya dukungan interface untuk perangkat lunak pihak ketiga melalui TCP/IP. Webots salah satu platform simulasi yang dukungan komponen yang besar yang mana dapat digunakan untuk simulasi dan kemungkinan untuk penambahan komponen lain.

Webots

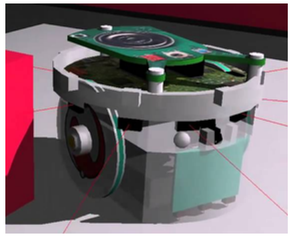
robot simulation

MATERIAL

E-Puck Robot

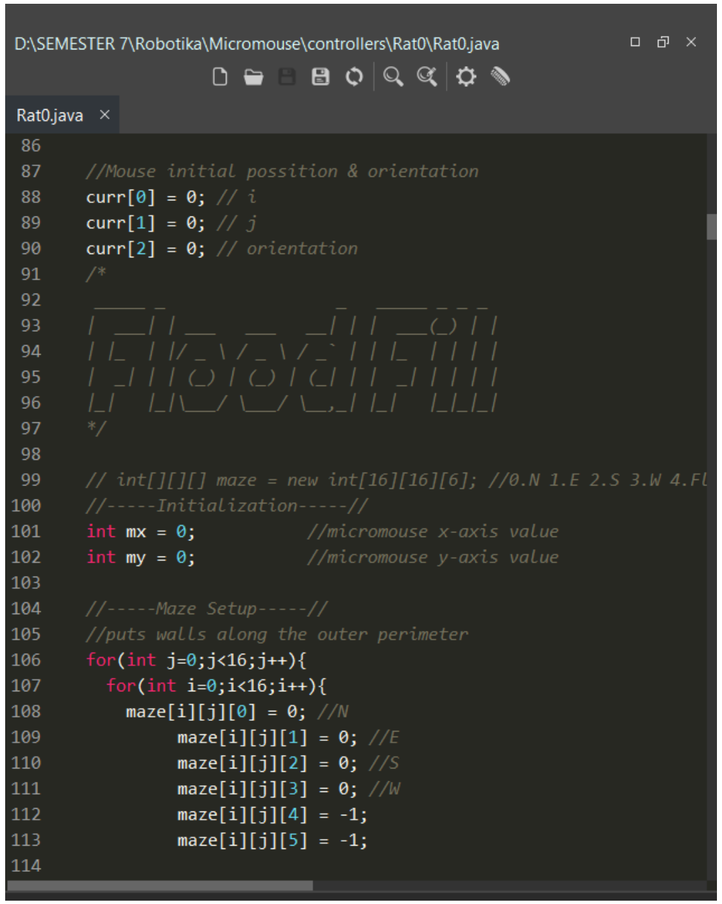
E-Puck Robot merupakan salah satu contoh robot yang dibuat melalui software Webots yaitu robot miniatur bergerak yang awalnya dikembangkan di EPFL untuk tujuan pengajaran oleh para perancang robot Khepera yang sukses.
E-puck dirancang untuk memenuhi persyaratan berikut:
Desain yang elegan, fleksibelitas, software simulasi, pengguna yang ramah, perawatan yang mudah, dan terjangkau.
Pada percobaan kali ini menggunakan E-Puck Robot ditekankan pada:

- Motor, diperlukan untuk gerakan lurus dan rotasi robot.
- Sensor Posisi, pada masing - masing roda, diperlukan untuk mencatat kecepatan rotasinya untuk perkiraan jarak yang ditempuh.
- LIDAR (Light Detection And Ranging): mendeteksi lingkungan sekitar sesuai kapasitas jangkauannya melalui pemindaian laser
- Kamera: memberikan umpan balik video secara real-time dari perspektif frontalrobot; dapat memungkinkan untuk memperkirakan jarak dari posisi dalam gambar yang akan diambil.



ALGORITMA

Algoritma Flood-Fill merupakan metode yang digunakan untuk menyelesaikan masalah pencarian rute terpendek pada labirin 5x5 dalam penelitian ini. Algoritma Flood-Fill melibatkan proses penomoran pada setiap sel dalam labirin dimana nomor-nomor ini merepresentasikan jarak setiap sel dengan sel tujuan. Sel tujuan yang ingin dicapai diberi nomor 0 dan sel–sel pada labirin yang memungkinkan untuk mencapai posisi tujuan, ditandai dengan cara n+1. Dengan algoritma ini, awalnya robot melakukan eksplorasi pada setiap sel pada labirin sehingga dihasilkan peta dari labirin tersebut, selanjutnya robot menelusuri sel-sel dengan nilai terkecil sesuai dengan peta yang telah dibuat dengan waktu yang lebih cepat dari waktu eksplorasi.



Dari hasil pengujian eksplorasi pada beberapa lapangan, robot selalu berhasil menemukan GOAL dengan metode algoritma flood-fill ini, tetapi tidak semua lapangan yang peta jalur terpendeknya berhasil dibuat oleh robot yang mungkin disebabkan adanya informasi yang belum dimiliki robot ketika robot menemui beberapa kondisi baru di beberapa lapangan yang memerlukan tindakan khusus.

SOURCE CODE

Rat0 - Notepad

File Edit Format View Help

```
import com.cyberbotics.webots.controller.Accelerometer;
// import com.cyberbotics.webots.controller.Camera;
import com.cyberbotics.webots.controller.DistanceSensor;
import com.cyberbotics.webots.controller.LED;
import com.cyberbotics.webots.controller.LightSensor;
import com.cyberbotics.webots.controller.Motor;
import com.cyberbotics.webots.controller.Robot;

import com.cyberbotics.webots.controller.PositionSensor; //ADDED

import java.util.*; //ADDED

public class Rat0 extends Robot {

  protected final int timeStep = 32;
  protected final double maxSpeed = 800;
  // protected final double[] collisionAvoidanceWeights = {0.06,0.03,0.015,0.0,0.0,-0.015,-0.03,-0.06};
  //                                                      0    1      2      3    4   5     6   7
  // protected final double[] collisionAvoidanceWeights = {-0.002,-0.005,-0.015,-0.6,0.3,0.005,0.0,0.0};{-0.002,-0.005,-0.015,-0.6,0.3,0.005,0.0,0.0};
  //                                                      0   1    2      3   4   5    6     7
  protected final double[] collisionAvoidanceWeights = {0.075,0.05,0.02,0.0,0.0,-0.02,-0.05,-0.075};

  protected final double[] slowMotionWeights = {0.0125,0.00625,0.0,0.0,0.0,0.0,0.0,0.00625,0.0125};
  // protected final double[] slowMotionWeights = {0.4,0.1,0.01,0.0,0.0,0.0,0.01,0.1,0.4};

  //ADDED
  protected final double wheelRadius = 0.02;
  protected final double axleLength = 0.052;

  protected Accelerometer accelerometer;
  // protected Camera camera;
  protected int cameraWidth, cameraHeight;
  protected Motor leftMotor, rightMotor;
  protected DistanceSensor[] distanceSensors = new DistanceSensor[8];
  protected LightSensor[] lightSensors = new LightSensor[8];
  protected LED[] leds = new LED[10];

  //ADDED
  //for distance sensor
  protected PositionSensor lps;
  protected PositionSensor rps;
```

Ln 10, Col 1

Rat0 - Notepad

File Edit Format View Help

```
  protected double ldis = 0;
  protected double rdis = 0;
  protected double dori = 0;
  //for turning
  protected double startori = 0;
  protected double oldori = 0;
  //for cell counting
  protected double[] oldpos = new double[2];
  protected boolean step = false;
  protected int counter = 0;
  protected int count = 0;
  protected int countpos = 0;
  protected int[] curr = new int[3]; //i, j ,orientation
  protected int[][] nextc = new int[3][3]; //i, j ,orientation x3

  protected boolean overR = false;
  protected boolean overL = false;
  protected boolean overU = false;

  protected int[][][] maze = new int[16][16][6]; //0.N 1.E 2.S 3.W 4.Flood 5.Orientation if visited

  public Rat0() {
    accelerometer = getAccelerometer("accelerometer");
    leftMotor = getMotor("left wheel motor");
    rightMotor = getMotor("right wheel motor");
    leftMotor.setPosition(Double.POSITIVE_INFINITY);
    rightMotor.setPosition(Double.POSITIVE_INFINITY);
    leftMotor.setVelocity(0.0);
    rightMotor.setVelocity(0.0);
    for (int i=0;i<10;i++) {
      leds[i]=getLED("led"+i);
    };
    for (int i=0;i<8;i++) {
      distanceSensors[i] = getDistanceSensor("ps"+i);
      distanceSensors[i].enable(timeStep);
    }

    //ADDED
    lps = leftMotor.getPositionSensor(); //left_position_sensor
    rps = rightMotor.getPositionSensor(); //right_position_sensor
    lps.enable(64);
```

SOURCE CODE

Rat0 - Notepad

File Edit Format View Help

```
    rps.enable(64);


    //Mouse initial possition & orientation
    curr[0] = 0; // i
    curr[1] = 0; // j
    curr[2] = 0; // orientation
    /*
     _____ _                   _   _____ _ _ _
    |  ___| | ___   ___   __| | |  ___(_) | |
    | |_  | |/ _ \ / _ \ / _` | | |_  | | | |
    |  _| | | (_) | (_) | (_| | |  _| | | | |
    |_|   |_|\___/ \___/ \__,_| |_|   |_|_|_|
    */

    // int[][][] maze = new int[16][16][6]; //0.N 1.E 2.S 3.W 4.Flood 5.Visited
    //-----Initialization-----//
    int mx = 0;           //micromouse x-axis value
    int my = 0;           //micromouse y-axis value

    //-----Maze Setup-----//
    //puts walls along the outer perimeter
    for(int j=0;j<16;j++){
      for(int i=0;i<16;i++){
        maze[i][j][0] = 0; //N
          maze[i][j][1] = 0; //E
          maze[i][j][2] = 0; //S
          maze[i][j][3] = 0; //W
          maze[i][j][4] = -1;
          maze[i][j][5] = -1;

        maze[i][15][0] = 1; // j==15 North
        maze[i][0][2] = 1; // j==0 South
        maze[0][j][3] = 1; // i==0 West
        maze[15][j][1] = 1; // i==15 East
      }
    }
    //-----Flood-----//
    //fills all flood array spaces with -1
    for(int i=0;i<16;i++){
      for(int j=0;j<16;j++){
        maze[i][j][4] = -1;
```

Rat0 - Notepad

File Edit Format View Help

```
    }
    //fills the four goal flood array spaces with 0
    maze[7][7][4] = 0;
    maze[7][8][4] = 0;
    maze[8][7][4] = 0;
    maze[8][8][4] = 0;

    // maze[7][7][2] = 1;
    // maze[7][6][0] = 1;

    // maze[8][7][2] = 1;
    // maze[8][6][0] = 1;

    // maze[7][7][3] = 1;
    // maze[6][7][1] = 1;

    // maze[7][8][3] = 1;
    // maze[6][8][1] = 1;


    maze[0][0][1] = 1;
    maze[1][0][3] = 1;
    // //North-West
    // maze[0][15][2] = 1;
    // maze[0][14][0] = 1;
    // //South-East
    // maze[15][0][0] = 1;
    // maze[15][1][2] = 1;
    // //North-East
    // maze[14][15][1] = 1;
    // maze[15][15][3] = 1;
    // maze[15][14][0] = 1;
    // maze[15][15][2] = 1;

    //Orientation
    maze[0][0][5] = 0;


    //fills the flood array with values using flood fill logic
    int k=0;
    while(maze[mx][my][4]==-1){  //stops filling when the flood fill reaches the micromouse's position
  //      System.out.print("OK: %d\n",k);
```

Rat0 - Notepad

File Edit Format View Help

```
    int k=0;
    while(maze[mx][my][4]==-1){  //stops filling when the flood fill reaches the micromouse's position
  //      System.out.print("OK: %d\n",k);
          for(int i=15;i>=0;i--){
        for(int j=15;j>=0;j--){
          if(maze[i][j][4]==k){  //if the flood array space equals k (starting at 0), place k+1 in adjacent flood array spaces
            if(j+1<16){
              if(maze[i][j+1][2]==0 && (maze[i][j+1][4]==-1)){  //North
                maze[i][j+1][4] = maze[i][j][4] + 1;
              }
            }
            if(j-1>=0){
              if(maze[i][j-1][0]==0 && (maze[i][j-1][4]==-1)){  //South
                maze[i][j-1][4] = maze[i][j][4] + 1;
              }
            }
            if(i+1<16){
              if(maze[i+1][j][3]==0 && (maze[i+1][j][4]==-1)){  //West
                maze[i+1][j][4] = maze[i][j][4] + 1;
              }
            }
            if(i-1>=0){
              if(maze[i-1][j][1]==0 && (maze[i-1][j][4]==-1)){  //East
                maze[i-1][j][4] = maze[i][j][4] + 1;
              }
            }
          }
        }
      }
      k++;
  //    if(k>50)
  //            break;
    }
    // print(maze);
  }
  public void run() {
    // char message[128];

    int blink = 0;
    int oldDx = 0;
    // Random r = new Random();
    boolean turn = false;
```

SOURCE CODE

Rat0 - Notepad

File Edit Format View Help

```
boolean turn = false;
boolean seeFeeder = false;
double battery;
double oldBattery = -1.0;
// int image[];
double distance[] = new double[8];
int ledValue[] = new int[10];
double leftSpeed, rightSpeed;

//ADDED
int timer = 0;
boolean rturn = false;
boolean lturn = false;
boolean uturn = false;
boolean front = false;
boolean right = false;
boolean left = false;

 for (int i = 0; i < 3; i++) {
   for (int j = 0; j < 3; j++) {
     nextc[i][j] = -1;
   }
 }

while (step(timeStep) != -1) {

  // read sensor information
  for(int i=0;i<8;i++) distance[i] = distanceSensors[i].getValue();
  // battery = batterySensorGetValue();
  for(int i=0;i<10;i++) ledValue[i] = 0;

  // obstacle avoidance behavior
  leftSpeed  = maxSpeed;
  rightSpeed = maxSpeed;

  for (int i=0;i<8;i++) {
    leftSpeed  -= (slowMotionWeights[i]+collisionAvoidanceWeights[i])*distance[i];
    rightSpeed -= (slowMotionWeights[i]-collisionAvoidanceWeights[i])*distance[i];
  }
  // return either to left or to right when there is an obstacle
  // if (distance[6]+distance[7] > 1800 || distance[0]+distance[1] > 1800) {
```

Rat0 - Notepad

File Edit Format View Help

```
  /*
   ___      _                      _
  / _ \  __| | ___  _ __ ___   ___| |_ _ __ _   _
 | | | |/ _` |/ _ \| '_ ` _ \ / _ \ __| '__| | | |
 | |_| | (_| | (_) | | | | | |  __/ |_| |  | |_| |
  \___/ \__,_|\___/|_| |_| |_|\___|\__|_|   \__, |
                                            |___/
  */

 double l = lps.getValue();//wb_position_sensor_get_value(left_position_sensor);
 double r = rps.getValue();//wb_position_sensor_get_value(right_position_sensor);
 ldis = l * wheelRadius;        // distance covered by left wheel in meter
 rdis = r * wheelRadius;        // distance covered by right wheel in meter
 dori = (rdis - ldis) / axleLength;  // delta orientation
 // System.out.print("estimated distance covered by left wheel: "+ldis+" m.\n");
 // System.out.print("estimated distance covered by right wheel: "+rdis+" m.\n");
 // System.out.print("estimated change of orientation: "+dori+" rad.\n");


 // System.out.println("rdiff: "+rdiff+" ldiff: "+ldiff);
if(!step && ldis >0 && rdis > 0){
  // System.out.println("STEP");
  oldpos[0] = ldis;
  oldpos[1] = rdis;
  // System.out.println("Saved:oldpos0: "+ldis+" oldpos1: "+rdis);
  step = true;
 }
```

Rat0 - Notepad

File Edit Format View Help

```
//step while turning
double rdiff = ldis-oldpos[0];
double ldiff = rdis-oldpos[1];

if(rdiff < 0){
  rdiff = 0;
}
if(ldiff < 0){
  ldiff = 0;
}
if(step && (uturn || rturn || lturn)){
  // System.out.println("STEP+TURN: left= "+rdiff+" right= "+rdiff+"rturn: "+rturn+" lturn: "+lturn+" uturn: "+uturn);
  if(rdiff > 0.08 && ldiff > 0.08){
    counter++;
    System.out.println("Counter++: "+counter);
    rdiff = 0;
    ldiff = 0;
  }
  step = false;

}


//strait step
if(step && rdiff > 0.108 && ldiff > 0.108){
  // System.out.println("oldpos0: "+oldpos[0]+" oldpos1: "+oldpos[1]);
  // System.out.println("rdiff: "+rdiff+" ldiff: "+ldiff);
  counter++;
  System.out.println("Counter: "+counter);
  rdiff = 0;
  ldiff = 0;
  step = false;
}
  // System.out.println("step: "+step);

/*
```

Rat0 - Notepad

File Edit Format View Help

```
/*
              _ _       _      _            _   _
__      ____ _| | |   __| | ___| |_ ___  ___| |_(_) ___  _ __
\ \ /\ / / _` | | |  / _` |/ _ \ __/ _ \/ __| __| |/ _ \| '_ \
 \ V  V / (_| | | | | (_| |  __/ ||  __/ (__| |_| | (_) | | | |
  \_/\_/ \__,_|_|_|___\__,_|\___|\__\___|\___|\__|_|\___/|_| |_|
                 |_____|
*/

if(counter > count){
  //new possition
  switch(curr[2]){
    case 0:
      curr[1]++;
      break;
    case 1:
      curr[0]++;
      break;
    case 2:
      curr[1]--;
      break;
    case 3:
      curr[0]--;
      break;
  }
```

Rat0 - Notepad

File Edit Format View Help

```
      break;
  }

  //wall detection
  if(distance[0] > 150 && distance[7] > 150){
    System.out.println("WALL: FRONT");
    front = true;
  }
  if(distance[2] > 150){ //distance[0]+distance[7] > 900 && distance[5] > distance[2]
    System.out.println("WALL: RIGHT");
    right = true;
  }
  if(distance[5] > 150){
    System.out.println("WALL: LEFT");
    left = true;
  }

  count++; // update to new position

  //update maze array
  switch(curr[2]){
    case 0:
      // System.out.println("North");
      if(front){
        maze[curr[0]][curr[1]][0] = 1;
        if(curr[1] > 0 && curr[1] < 15) maze[curr[0]][curr[1]+1][2] = 1;
      }
      else{
        nextc[0][0] = curr[0];
        nextc[1][0] = curr[1]+1;
        nextc[2][0] = 0;
      }
      if(right){
        maze[curr[0]][curr[1]][1] = 1;
        if(curr[0] > 0 && curr[0] < 15) maze[curr[0]+1][curr[1]][3] = 1;
      }
      else{
        nextc[0][1] = curr[0]+1;
        nextc[1][1] = curr[1];
        nextc[2][1] = 1;
      }
      if(left){
```

```
        maze[curr[0]][curr[1]][3] = 1;
        if(curr[0] > 0 && curr[0] < 15) maze[curr[0]-1][curr[1]][1] = 1;
      }
      else{
        nextc[0][2] = curr[0]-1;
        nextc[1][2] = curr[1];
        nextc[2][2] = 3;
      }
      break;
    case 1:
      if(front){
        maze[curr[0]][curr[1]][1] = 1;
        if(curr[0] > 0 && curr[0] < 15) maze[curr[0]+1][curr[1]][3] = 1;
      }
      else{
        nextc[0][0] = curr[0]+1;
        nextc[1][0] = curr[1];
        nextc[2][0] = 1;
      }
      if(right){
        maze[curr[0]][curr[1]][2] = 1;
        if(curr[1] > 0 && curr[1] < 15) maze[curr[0]][curr[1]-1][0] = 1;
      }
      else{
        nextc[0][1] = curr[0];
        nextc[1][1] = curr[1]-1;
        nextc[2][1] = 2;
      }
      if(left){
         maze[curr[0]][curr[1]][0] = 1;
        if(curr[1] > 0 && curr[1] < 15) maze[curr[0]][curr[1]+1][2] = 1;
      }
      else{
        nextc[0][2] = curr[0];
        nextc[1][2] = curr[1]+1;
        nextc[2][2] = 0;
      }
      break;
    case 2:
      if(front){
```

SOURCE CODE

Rat0 - Notepad

File Edit Format View Help

```
        break;
      case 2:
        if(front){
          maze[curr[0]][curr[1]][2] = 1;
          if(curr[1] > 0 && curr[1] < 15) maze[curr[0]][curr[1]-1][0] = 1;
        }
        else{
          nextc[0][0] = curr[0];
          nextc[1][0] = curr[1]-1;
          nextc[2][0] = 2;
        }
        if(right){
          maze[curr[0]][curr[1]][3] = 1;
          if(curr[0] > 0 && curr[0] < 15) maze[curr[0]-1][curr[1]][1] = 1;
        }
        else{
          nextc[0][1] = curr[0]-1;
          nextc[1][1] = curr[1];
          nextc[2][1] = 3;
        }
        if(left){
          maze[curr[0]][curr[1]][1] = 1;
          if(curr[0] > 0 && curr[0] < 15) maze[curr[0]+1][curr[1]][3] = 1;
        }
        else{
          nextc[0][2] = curr[0]+1;
          nextc[1][2] = curr[1];
          nextc[2][2] = 1;
        }
        break;
      case 3:
        if(front){
          maze[curr[0]][curr[1]][3] = 1;
          if(curr[0] > 0 && curr[0] < 15) maze[curr[0]-1][curr[1]][1] = 1;
        }
        else{
           nextc[0][0] = curr[0]+1;
          nextc[1][0] = curr[1];
          nextc[2][0] = 3;
        }
        if(right){
          maze[curr[0]][curr[1]][0] = 1;
        }
        else{
          nextc[0][1] = curr[0];
          nextc[1][1] = curr[1]+1;
          nextc[2][1] = 0;
        }
        if(left){
          maze[curr[0]][curr[1]][2] = 1;
          if(curr[1] > 0 && curr[1] < 15) maze[curr[0]][curr[1]-1][0] = 1;
        }
        else{
          nextc[0][2] = curr[0];
          nextc[1][2] = curr[1]-1;
          nextc[2][2] = 2;
        }
        break;
    }

    // Flood Fill values regeneration
    for (int i = 0; i < 16; i++) {
      for (int j = 0; j < 16; j++) {
        maze[i][j][4] = -1;
      }
    }
    maze[7][7][4] = 0;
    maze[7][8][4] = 0;
    maze[8][7][4] = 0;
    maze[8][8][4] = 0;

    for (int k = 0; k < 256; k++) {
      for (int i = 0; i < 16; i++) {
        for (int j = 0; j < 16; j++) {
          if (maze[i][j][4] == k) {  //if the flood array space equals k (starting at 0), place k+1 in adjacent flood array sp
            if (i < 15) {
              if (maze[i + 1][j][3] == 0 && (maze[i + 1][j][4] == -1)) {  //North
                maze[i + 1][j][4] = maze[i][j][4] + 1;
              }
            }
```

Rat0 - Notepad

File Edit Format View Help

```
            }
            if (i > 0) {
              if (maze[i - 1][j][1] == 0 && (maze[i - 1][j][4] == -1)) {  //South
                maze[i - 1][j][4] = maze[i][j][4] + 1;
              }
            }
            if (j < 15) {
              if (maze[i][j + 1][2] == 0 && (maze[i][j + 1][4] == -1)) {  //East
                maze[i][j + 1][4] = maze[i][j][4] + 1;
              }
            }
            if (j > 0) {
              if(maze[i][j - 1][0] == 0 && (maze[i][j - 1][4] == -1)) {  //West
                maze[i][j - 1][4] = maze[i][j][4] + 1;
              }
            }
          }
        }
      }
    }
    //neihbour selection


    //turn selection
    if(front && right && left){
      overU = true;
      System.out.println("TURN: UTURN");
      curr[2] -= 2;
    }
    else if(front && right){
      overL = true;
      System.out.println("TURN: LEFT");
      curr[2] -= 1;
    }
    else if(front && left){
      overR = true;
      System.out.println("TURN: RIGHT");
      curr[2] += 1;
    }
    else if(front){
      if(maze[nextc[0][1]][nextc[1][1]][4] < maze[nextc[0][2]][nextc[1][2]][4]){ // R>L
        overR = true;
        overR = true;
        System.out.println("**TURN: RIGHT");
        System.out.println("maze["+nextc[0][1]+"]["+nextc[1][1]+"][4]="+maze[nextc[0][1]][nextc[1][1]][4]+
          " < maze["+nextc[0][2]+"]["+nextc[1][2]+"][4]="+maze[nextc[0][2]][nextc[1][2]][4]);
        curr[2] += 1;
      }
      else{
        overL = true;
        System.out.println("**TURN: LEFT");
        System.out.println("maze["+nextc[0][1]+"]["+nextc[1][1]+"][4]="+maze[nextc[0][1]][nextc[1][1]][4]+
          " >= maze["+nextc[0][2]+"]["+nextc[1][2]+"][4]="+maze[nextc[0][2]][nextc[1][2]][4]);
        curr[2] -= 1;
      }
    }
    else if (left && !right) {
      if(maze[nextc[0][1]][nextc[1][1]][4] < maze[nextc[0][0]][nextc[1][0]][4]){ // N>L
        overR = true;
        System.out.println("**TURN: RIGHT");
        System.out.println("maze["+nextc[0][1]+"]["+nextc[1][1]+"][4]="+maze[nextc[0][1]][nextc[1][1]][4]+
          " < maze["+nextc[0][2]+"]["+nextc[1][2]+"][4]="+maze[nextc[0][2]][nextc[1][2]][4]);
        curr[2] += 1;
      }
    }
    else if (right && !left) {
      if(maze[nextc[0][0]][nextc[1][0]][4] > maze[nextc[0][2]][nextc[1][2]][4]){ // L>N
        overL = true;
        System.out.println("**TURN: LEFT");
        System.out.println("maze["+nextc[0][1]+"]["+nextc[1][1]+"][4]="+maze[nextc[0][1]][nextc[1][1]][4]+
          " >= maze["+nextc[0][2]+"]["+nextc[1][2]+"][4]="+maze[nextc[0][2]][nextc[1][2]][4]);
        curr[2] -= 1;
      }
    }
    else
      System.out.println("going straight");

    //orientation in bounds
    if(curr[2] == 4)
      curr[2] = 0;
    else if(curr[2] == -1)
      curr[2] = 3;
    else if(curr[2] == -2)
      curr[2] = 2;
```

```
    //reset
    front = false;
    right = false;
    left = false;

    maze[curr[0]][curr[1]][5] = curr[2];

    print_maze(maze);

    System.out.println("POSITION: ["+curr[0]+","+curr[1]+"] Orientation: "+curr[2]);
  }

  /*
    ____
   |_   _|
     | || | | | '__| '_ \
     | || |_| | |  | | | |
     |_| \__,_|_|  |_| |_|

  */
  // if (distance[0]+distance[7] > 900 && (distance[2] >= distance[5] - 100 && distance[2] <= distance[5] + 100) || uturn) {
  if ((distance[0]+distance[7] > 600 && (distance[2] >= 200 && distance[5] >= 200) || uturn) || overU){
    // System.out.println("U-Turn"+uturn+":"+dori);
    if(uturn == false){
      startori = dori;
      // System.out.println("startori: "+dori);
    }
    uturn = true;
    overU = false;


    ledValue[8] = 1;

    leftSpeed  = -maxSpeed;
    rightSpeed = maxSpeed;

    if(startori + 3.1 < dori){
      uturn = false;
      ledValue[8] = 0;


      ledValue[8] = 1;

      leftSpeed  = -maxSpeed;
      rightSpeed = maxSpeed;

      if(startori + 3.1 < dori){
        uturn = false;
        ledValue[8] = 0;

        leftSpeed  = maxSpeed;
        rightSpeed = maxSpeed;
      }

    }
    else if (rturn || overR){ //distance[0]+distance[7] > 900 && distance[5] > distance[2] ||
      // System.out.println("Turning: RIGHT>>");
      if(rturn == false){
        startori = dori;
        // System.out.println("startori: "+dori);
      }
      // uturn = true;
      lturn = false;
      rturn = true;
      overR = false;


      ledValue[8] = 1;

      leftSpeed  = maxSpeed;
      rightSpeed = -maxSpeed;

      if(startori - 1.4 > dori){
        rturn = false;
        ledValue[8] = 0;

        leftSpeed  = maxSpeed;
        rightSpeed = maxSpeed;
      }
    }
    else if (lturn || overL){ //distance[0]+distance[7] > 900 && distance[5] < distance[2] ||
```

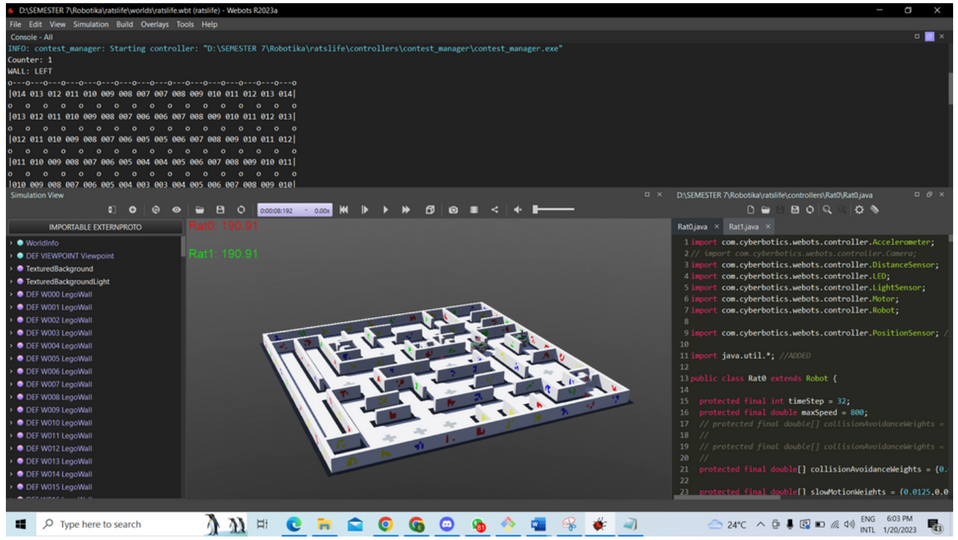
SOURCE CODE

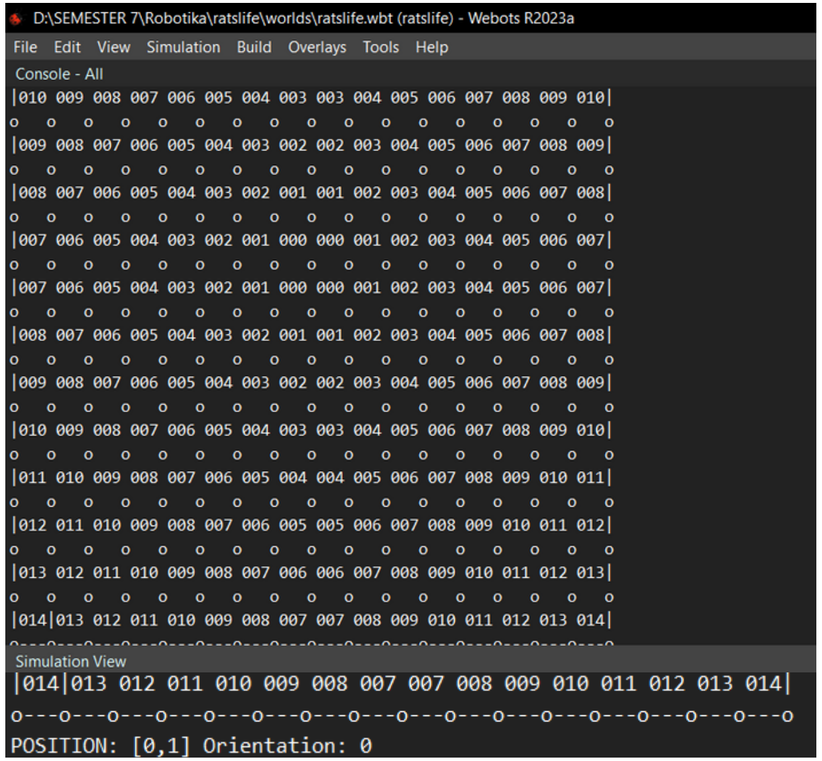
Rat0 - Notepad

File Edit Format View Help

```
    }
  }
  else if (lturn || overL){ //distance[0]+distance[7] > 900 && distance[5] < distance[2] ||
    // System.out.println("Turning: LEFT>>");
    if(lturn == false){
      startori = dori;
      // System.out.println("startori: "+dori);
    }
    // uturn = true;
    lturn = true;
    // rturn = true;
    overL = false;


    ledValue[8] = 1;

    leftSpeed  = -maxSpeed;
    rightSpeed = maxSpeed;

    if(startori + 1.4 < dori){
      lturn = false;
      ledValue[8] = 0;

      leftSpeed  = maxSpeed;
      rightSpeed = maxSpeed;
    }
  }

    if (blink++ >= 5) { // blink the back LEDs
      ledValue[6] = 0;
      if (blink == 20) blink = 0;
    }

    //LED mapping 8:body-G led0' to 'led7' (e-puck ring), 'led8' (body-G) and 'led9' (front)

    // set actuators
    for(int i=0; i<10; i++) {
      leds[i].set(ledValue[i]);
    }
    leftMotor.setVelocity(0.00628 * leftSpeed);
    rightMotor.setVelocity(0.00628 * rightSpeed);
  }

  }
  // Enter here exit cleanup code
}

public void print_maze(int[][][] maze){
  /*
             _                 _
 _ __  _ __(_)_ __ | |_    _ __ ___   __ _ _______
| '_ \| '__| | '_ \| __|  | '_ ` _ \ / _` |_  / _ \
| |_) | |  | | | | | |_   | | | | | | (_| |/ /  __/
| .__/|_|  |_|_| |_|\__|__|_| |_| |_|\__,_/___\___|
|_|                  |_____|
  */

  int w = 16;
  int h = 16;
      // int dim = w*h;
      int i, j;


      // System.out.print("print\n");

      // for(j=0; j<h*2+1; j++){ //Left POV
      for(j=h*2; j>=0; j--){ //Left POV
        for(i=0; i<w*2+1; i++){ //Right POC

              if(j%2==0){ //j -> joint & h-wall value

                if(i%2==0){ // i -> joint value
                      System.out.print("o");
                }else{//j -> h-wall value
                        switch(j){
                        case 32:
                                if(maze[i/2][15][0]==1){ //North
                                        System.out.print("---");
                                }else
                                        System.out.print("   ");
            break;
          case 0:
                                if(maze[i/2][0][2]==1){ //North
                                        System.out.print("---");
                                }else
                                        System.out.print("   ");
            break;
          default:
            // if(j<2) break;
            if(maze[i/2][j/2][2]==1 || maze[i/2][j/2-1][0]==1){ //South
                                        System.out.print("---");
                                        // if(maze[i/2][j/2][2] != maze[i/2][(j-2)/2][0] && j!=0){
                                              //  System.out.print("ERROR: Neibhour N&S dont match!");
                                        // }
                                }else
                                        System.out.print("   ");
                        }
                  }
                }
                else{//j -> v-wall&cell value

                  if(i%2==0){ // i -> v-wall value

                        switch(i){
                        case 0:
                                if(maze[0][j/2][3]==1){ //West
                                        System.out.print("|");
                                }else
                                        System.out.print(" ");
            break;
          default:
                                if(maze[i/2-1][j/2][1]==1 || maze[i/2][j/2][3]==1){ //East
                                        System.out.print("|");
                                        // if(maze[(i-2)/2][j/2][1] != maze[i/2][j/2][3] && i!=h*2){
                                              //   System.out.print("ERROR: Neibhour E&W dont match!");
                                        // }
                                }else
                                        System.out.print(" ");
                        }

                    }else{ // i -> cell value

                         // System.out.print("%.3d",i/2+(j/2)*h);
                         // System.out.print("%.3d",i/2+(j/2)*h);
                         // System.out.print("%.2d,%.2d",i/2,j/2);
                         // System.out.printf("%03d",maze[i/2][j/2][5]);
                         // System.out.print("    ");
                           if(maze[i/2][j/2][4]>=0){
                                   System.out.printf("%03d",maze[i/2][j/2][4]);
                           }else{
                                   System.out.printf("%03d",maze[i/2][j/2][4]);
                           }
                      }
                }
        }
        System.out.print("\n");
      }

  }

  public static void main(String[] args) {
    Rat0 rat0 = new Rat0();
    rat0.run();

  }
}
```

HASIL dan ANALISA





Pada percobaan ini, selain robot bisa melewati rintangan juga sekaligus bisa merekam untuk mendeteksi jalur mana yang lebih cepat dan ringkas untuk dilalui. Terdapat juga sensor agar robot bisa mendeteksi rintangan sehingga robot bisa menghindari rintangan tersebut.