**PATH BUILDER ROBOT**

**“Automated and Map Creation by a 3 Wheeled Robotic System”**

## Team Members

* Pramit Majumder(2326006, B.Tech in Mechatronics Engineering, KIIT UNIVERSITY)
* Elina Yasmin(2330158, B.Tech in Electronics & Computer Science Engineering, KIIT UNIVERSITY)
* Varunsai Sindri(2426306, Diploma in Mechatronics Engineering, KIIT UNIVERSITY)

## **Abstract**

Our Project mainly is a 3 wheeled robot consisting of Motor Drivers, Wheels, Castor Wheel, ESP32 Devboard and more. The project mainly aims to solve the famous Kidnapped Robot Problem, in which a robot is taken to an unknown environment where it has to localize and create a map of the environment. This map helps the robot create a pathway for other robots to use it according to their own will. The Kidnapped robot problem has been a problem statement the time, since robots have become autonomous. It is also known as the mobile robot localisation. Although the problem has been solved using various sensors over time, LIDAR Sensors are mostly used for this problem. Whereas for the Algorithm, SLAM has been used to solve problems. Another unique algorithm is the Monte Carlo Algorithm. The outcome usually relies that a resolute map has been generated by the robot and will be able to navigate the environment with ease and with points where it can go from one position to another position without crashing.

## **Introduction**

The Significance of the problems statement lies within the name of the problem in itself, While we are solving the Kidnapped Robot Problem, this mainly a case of path detection and planning. A robot that can plan for itself will be much more self reliant, rather than a robot that cannot. Let there be 2 robots who clean rooms being named, Robot A who can solve the problem, and robot B who can’t. While both can avoid obstacles. Robot A can overtime notice the concentration where the dust and particulates settle overtime and thus are more efficient. Robot B however has to clean the whole floor over and over again thus using more battery power and more time.

The objective of the project is to make the robot create a map of the platform it is running on.

## **Problem Statement**

The problem statement states that a robot has to localize itself and learn the unknown environment they have been kept in.

**Constraints**

* There should be definite boundary for the environment
* The boundary cannot change overtime

## **Literature Review**

**Existing Solutions**

* Using the Monte Carlo Algorithm and LiDar in ROOMBA(room cleaning robots)
* LiDar and Particle Swarm Optimisation Technique in GPS Denied Drones

## **System** **Design**

**HARDWARE** **OVERVIEW**

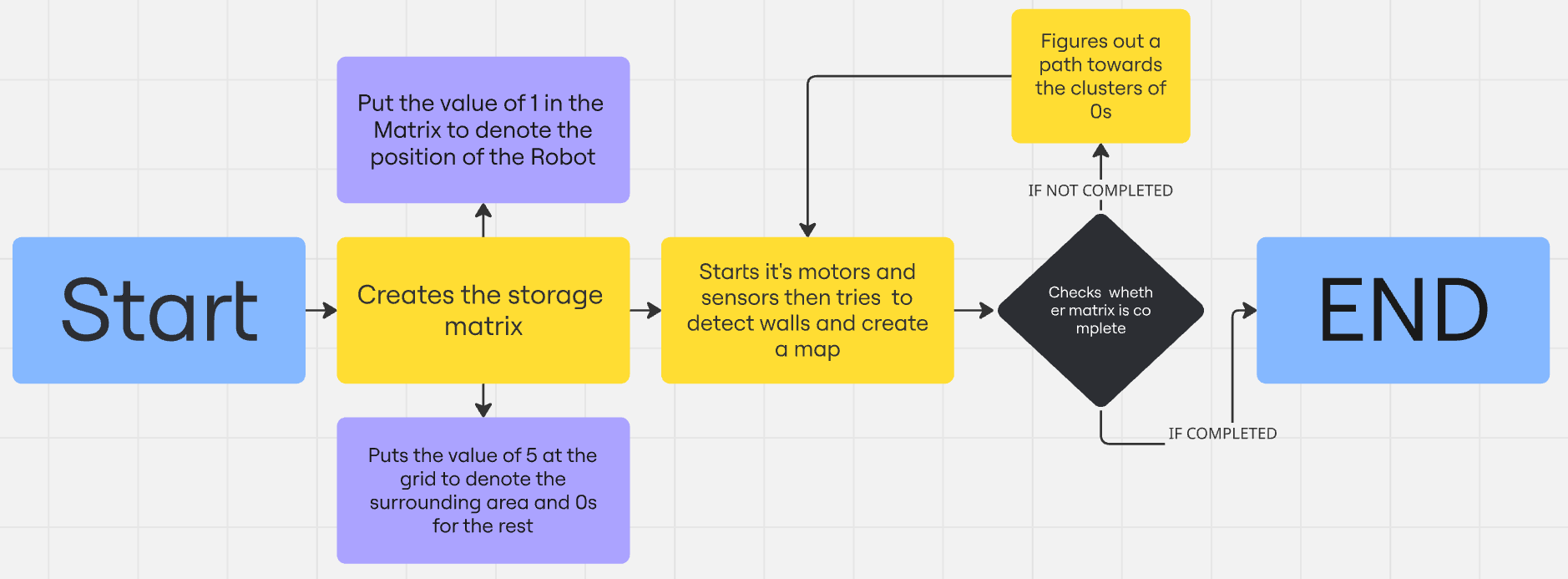
* Wheels
* Castor Wheel
* 3D Printed Chassis
* L298N Motor Driver
* ESP32 Devboard
* LiPo Battery Pack
* Breadboard
* BO Motors
* Ultrasonic Sensors

**MECHANICAL OVERVIEW**

* The chassis was ordered from Robu.in, so it was assembled with the main chassis and the castor wheel and the BO motors were added later with compatible Wheels.
* It has ESP32 and a L298N mounted on top of the chassis secured using super glue.
* The battery consists of 3 Lithium Polymer Battery secured using super glue underneath the chassis

**SOFTWARE OVERVIEW**

* To solve the Kidnapped Robot Problem on our way, Our team thought of using the matrix method to solve it and using our laptop to host a standard HTML Website and send the necessary matrix data and upload it to the website. The website would show how the map made by the robot is looking.
* The robot would start off mapping it’s boundaries then go forward to create the appropriate length of map for the matrix then it would slowly send the data to the website and hence would create a low resolution map.



## **METHODOLOGY**

* Development Process
  + The chassis was ordered from Robu.in, so it was assembled with the main chassis and the castor wheel and the BO motors were added later.

All the electronics are held together by zip ties

* Algorithm Explanation
  + So we have the robot loiter and create a necessary boundary while each data point is sent to the web server.
  + After the boundary is measured, we have our robot loiter inside of the boundary to detect more ranges and barriers within the surroundings
* Testing and Calibration
  + The robot has to be set in a proper boundary and have enough space for it to move around by itself.

## **IMPLEMENTATION**

## 

## 

## 

## 

// #include <stdio.h>

// #include <math.h>

// #include <stdlib.h>

const int Y = 7;

const int X = 7;

int final\_x = 1;

int final\_y = 1;

int target\_x = 0;

int target\_y = 0;

int da = 0, i = 0, j = 0;

int dest\_x = 0, dest\_y = 0;

bool left, right, front, leftM, rightM, frontM, completed;

int rowise = 1;

int colwise = -1;

int x\_way[] = {0,1,0,-1}, y\_way[] = {1,0,-1,0};

int x\_way\_n = 0, y\_way\_n = 0;

int Matrix[X][Y], row[Y], col[X];

void print\_Matrix(){

  for (i = 0; i < Y; i++) {

    for ( j =0; j < X; j++) {

        Serial.print(Matrix[i][j]);

        Serial.print(" ");

      }

      Serial.println("");

  }

}

//MOTOR FUNCTIONS

void turn90(bool direction){

  if(!direction){//RIGHT

      digitalWrite(5, HIGH);

      digitalWrite(18, LOW);

      digitalWrite(19, LOW);

      digitalWrite(21, HIGH);

      delay(1000/2);

      digitalWrite(5, LOW);

      digitalWrite(18, LOW);

      digitalWrite(19, LOW);

      digitalWrite(21, LOW);

  }

  else{

      digitalWrite(5, LOW);

      digitalWrite(18, HIGH);

      digitalWrite(19, HIGH);

      digitalWrite(21, LOW);

      delay(1000/2);

      digitalWrite(5, LOW);

      digitalWrite(18, LOW);

      digitalWrite(19, LOW);

      digitalWrite(21, LOW);

      delay(2000);

  }

}

void change\_right(){

  x\_way\_n = check(x\_way\_n + 1);

  y\_way\_n = check(y\_way\_n + 1);

  rowise = -rowise;

  colwise = -colwise;

  turn90(false);

}

void change\_left(){

  x\_way\_n = check(x\_way\_n - 1);

  y\_way\_n = check(y\_way\_n - 1);

  rowise = -rowise;

  colwise = -colwise;

  turn90(true);

}

void change\_forward(){

  digitalWrite(5, HIGH);

  digitalWrite(18, LOW);

  digitalWrite(19, HIGH);

  digitalWrite(21, LOW);

  delay(500);

  digitalWrite(5, LOW);

  digitalWrite(18, LOW);

  digitalWrite(19, LOW);

  digitalWrite(21, LOW);

  Matrix[final\_x][final\_y] = 0;

  // Serial.print("X: ");

  // Serial.println(final\_x + x\_way[x\_way\_n]);

  // Serial.print("Y: ");

  // Serial.println(final\_y + y\_way[y\_way\_n]);

  // Serial.println();

  // Serial.println(final\_x + x\_way[x\_way\_n] != 0);

  // Serial.println(final\_x + x\_way[x\_way\_n] != X - 1);

  // Serial.println(final\_y + y\_way[y\_way\_n] != 0);

  // Serial.println(final\_y + y\_way[y\_way\_n] != Y - 1);

  Matrix[final\_x][final\_y] = 7;

  if(final\_x + x\_way[x\_way\_n] != 0 && final\_y + y\_way[y\_way\_n] != 0 && final\_x + x\_way[x\_way\_n] != X - 1  && final\_y + y\_way[y\_way\_n] != Y - 1){

    final\_x += x\_way[x\_way\_n];

    final\_y += y\_way[y\_way\_n];

    Matrix[final\_x][final\_y] = 1;

  }

  else{

    Serial.println("Cannot Move sir");

    Matrix[final\_x][final\_y] = 1;

  }

}

//LOOKERS

bool look\_right(int x\_, int y\_){

  int x\_right = (x\_ + x\_way[check(x\_way\_n + 1)]);

  int y\_right = (y\_ + y\_way[check(y\_way\_n + 1)]);

  Serial.print("Right map: ");

  Serial.println(Matrix[x\_right][y\_right]);

  if(right == false && Matrix[x\_right][y\_right] == 0){

    Matrix[x\_right][y\_right] = 5;

    right = true;

  }

  if((Matrix[x\_right][y\_right] == 0) /\*|| Matrix[x\_right][y\_right] != 5 || Matrix[x\_right][y\_right] != 7\*/){

    return true;

  }

  else{

    return false;

  }

}

bool look\_left(int x\_, int y\_){

  int x\_left = (x\_ + x\_way[check(x\_way\_n - 1)]);

  int y\_left = (y\_ + y\_way[check(y\_way\_n - 1)]);

  Serial.print("Left map: ");

  Serial.println(Matrix[x\_left][y\_left]);

  if(left == false && Matrix[x\_left][y\_left] == 0){

    Matrix[x\_left][y\_left] = 5;

    left = true;

  }

  if((Matrix[x\_left][y\_left] == 0/\*|| Matrix[x\_left][y\_left]\*/)){

    return true;

  }

  else{

    return false;

  }

}

bool look\_forward(int x\_, int y\_){

  int x\_front = (x\_ + x\_way[check(x\_way\_n)]);

  int y\_front = (y\_ + y\_way[check(y\_way\_n)]);

  Serial.print("Front map: ");

  Serial.println(Matrix[x\_front][y\_front]);

  if(front == false && Matrix[x\_front][y\_front] == 0){

    Matrix[x\_front][y\_front] = 5;

    front = true;

  }

  if(Matrix[x\_front][y\_front] == 0  /\*|| Matrix[x\_front][y\_front] != 5 || Matrix[x\_front][y\_front] != 7\*/){

    return true;

  }

  else{

    return false;

  }

}

int find\_zeroes(){

  Serial.println("asdas");

  int occurance = find\_occurance();

  int targets[occurance][2];

  int counter = 0;

  for (i = 0; i < X; i++) {

    for (j = 0; j < Y; j++) {

      if (Matrix[i][j] == 0) {

        targets[counter][0] = i;

        targets[counter][1] = j;

        counter++;

        Serial.print("Found Targets");Serial.println(X);Serial.println(Y);

      }

    }

  }

  if(occurance > 0){

    int x\_ = targets[0][0], y\_ = targets[0][1];

    double d = distance(targets[0][0], targets[0][1], final\_x, final\_y);

    for(int i = 1; i < occurance; i++){

      Serial.println(d);

      Serial.println(distance(targets[i][0], targets[i][1], final\_x, final\_y));

      if(0 < (d - distance(targets[i][0], targets[i][1], final\_x, final\_y))){

        d = distance(targets[i][0], targets[i][1], final\_x, final\_y);

        x\_ = targets[i][0];

        y\_ = targets[i][1];

      }

    }

    dest\_x = x\_;

    dest\_y = y\_;

    da = 0;

    find\_pathway();

  }

}

int find\_occurance(){

  int count = 0;

  for (i = 1; i < X - 1; i++) {

    for (j = 1; j < Y - 1; j++) {

      if (Matrix[i][j] == 0) {

        count++;

      }

    }

  }

  return count;

}

double distance(int x1, int y1, int x2, int y2){

  double d = ((x2 - x1)\* (x2 - x1)) + ((y2 - y1)\*(y2 - y1));

  return sqrt(d);

}

void find\_pathway(){

  da == 0;

  if(final\_x < dest\_x){

    int difx = abs(final\_x - dest\_x);

    int dify = abs(final\_y - dest\_y);

    for(int i = 1; i <= difx; i++){

      if(!closed\_space(final\_x+i, final\_y)){

        Matrix[final\_x + i][final\_y] = 0;

      }

    }

    print\_Matrix();

  }

  Serial.println("FOUND");

}

//checkers

bool closed\_space(int x, int y){

  return !(look\_right(x,y) || look\_left(x,y) || look\_forward(x,y));

}

int check(int x){

  if(x == 4){

    return 0;

  }

  else if(x == -1){

    return 3;

  }

  return x;

}

long measureDistance(int trigPin, int echoPin) {

  digitalWrite(trigPin, LOW);

  delayMicroseconds(2);

  digitalWrite(trigPin, HIGH);

  delayMicroseconds(10);

  digitalWrite(trigPin, LOW);

  long duration = pulseIn(echoPin, HIGH);

  return (duration \* 0.034 / 2);

}

bool dataPoint() {

    // Iterate over each row

    for (i = 0; i < X; i++) {

        // Iterate over each column

        for (j = 0; j < Y; j++) {

            if (Matrix[i][j] == 0) {

                return true; // Exit once the element is found

            }

        }

    }

    return false;

}

void setup() {

  pinMode(13, OUTPUT);

  pinMode(12, INPUT);

  pinMode(14, OUTPUT);

  pinMode(27, INPUT);

  pinMode(26, OUTPUT);

  pinMode(25, INPUT);

  pinMode(5, OUTPUT);

  pinMode(18, OUTPUT);

  pinMode(19, OUTPUT);

  pinMode(21, OUTPUT);

  pinMode(22, OUTPUT);

  pinMode(23, OUTPUT);

  analogWrite(22, 155);

  analogWrite(23, 155);

  int y = 1;

  for (i = 0; i < Y; i++) {

      for (j =0; j < X; j++) {

            if(j == 0 || j == X - 1 || i == 0 || i == Y - 1){

                Matrix[i][j] = 5;

            }

            else{

                Matrix[i][j] = 0;

                y++;

            }

      }

  }

  Matrix[final\_x][final\_y] = 1;

  Serial.begin(115200);

}

//OPERATIONS

void check\_next\_step(){

  // if(Matrix[final\_x + x\_way[x\_way\_n]][final\_y + y\_way[y\_way\_n]] == 5){

    leftM = look\_left(final\_x, final\_y);

    frontM = look\_forward(final\_x, final\_y);

    rightM = look\_right(final\_x, final\_y);

    if(leftM == false){

        if(frontM == false){

            if(rightM == false){

                Serial.println("REVERSE");//if LEFT RIGHT AND FRONT ARE FALSE

                change\_right();

                change\_right();

                if(da == 2){

                  find\_zeroes();

                }

                da++;

                delay(2000);

            }

            else{

                Serial.println("TURN RIGHT");// IF LEFT AND FRONT ARE FALSE

                change\_right();

            }

        }

        else{

            Serial.println("GO FRONT");// IF RIGHT AND LEFT ARE FALSE

        }

    }

    else{

        if(frontM == false){

            if(rightM == false){

                Serial.println("TURN LEFT");//IF RIGHT AND FRONT ARE FALSE

                change\_left();

            }

            else{

                Serial.println("THE OPTIMAL WAY");// IF ONLY FRONT IS FALSE

                change\_left();

            }

        }

        else{

            Serial.println("CONTINUE AS NORMAL");// IF FRONT IS NOT FALSE

        }

    }

}

void get\_row(){

  for(i = 0;i < Y; i++){

    row[i] = Matrix[final\_x][i];

  }

}

void get\_column(){

  for(i = 0;i < X; i++){

    col[i] = Matrix[i][final\_y];

  }

}

void loop() {

  check\_next\_step();

  change\_forward();

  print\_Matrix();

    // Serial.println("Recording data");

  long leftd = measureDistance(26, 25);

  if((leftd <= 10)){

    Serial.println("LEFT BLOCKED");

    left = false;

  }

  long rightd = measureDistance(14, 27);

  long frontd = measureDistance(13, 12);

  if((rightd <= 10)){

    Serial.println("RIGHT BLOCKED");

    right = false;

  }

  if((frontd <= 20)){

    Serial.println("FRONT BLOCKED");

    front = false;

  }

  Serial.println("\nRECORDED");

  delay(1000);

  Serial.println(leftd);

  Serial.println(rightd);

  Serial.println(frontd );

}

***Challenges***

* The initial plan was to do it all on the esp32 board, The only problem was that we would not be able to see the data. But the new and revised code now presents the data on a locally hosted website

## **Experimental Results**

Performance Metrics

* The Ultrasonic Sensors were able to handle the sensor data and were able to kind of complete a small data map.

Comparison To Expectations

* We had thought that the project would’ve been easier than the project presented to be but with a bit more time, we are positive to complete it.

## **Analysis and Discussion**

* The systems is only able note the ground positions, as in only the wall positions, It cannot measure anything elevated
* The motors have worked well, But we would’ve improved upon the vibration caused by the motors while operating.

## **Challenges and Limitations**

* The only limitations are the sensors, which sometimes considered unstable. While we could’ve used a Time of Flight sensor for better range and results, we chose to use Ultrasonic sensors since we were more comfortable with it.

## **Future Work and Enhancements**

* The sensors could be upgraded to a time of flight sensor for a small scale robot. It would be even better if we were able to use a LiDar sensor.
* If instead of Zip ties we could’ve 3D printed the cases for the battery and motor holder the vibration would've been much less.

## **Conclusion**

The project’s main achievement has been the way of figuring out a path for such a small scale robot.

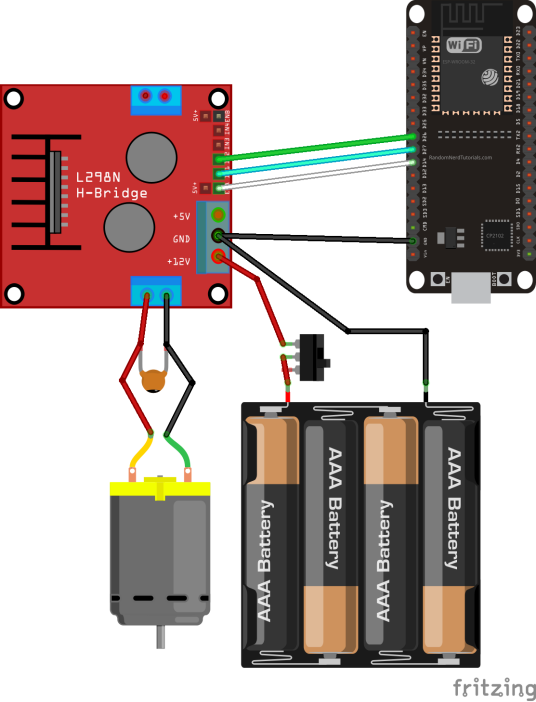
The system could prove useful for more small scale robotics or swarm robotics.

## **References**

* + <https://www.sciencedirect.com/topics/engineering/path-planning#:~:text=The%20aim%20of%20path%20planning,time%20to%20the%20final%20destination>.
  + <https://randomnerdtutorials.com/esp32-web-server-arduino-ide/>
  + <https://randomnerdtutorials.com/esp32-client-server-wi-fi/>
  + <https://www.w3schools.com/ai/ai_matrices.asp>
  + <https://stackoverflow.com/questions/7652342/how-to-print-the-following-multi-dimensional-array-in-java-script>
  + <https://randomnerdtutorials.com/esp32-access-point-ap-web-server/#:~:text=To%20connect%20to%20the%20access,IP%20address%20on%20your%20browser>.
  + https://www.youtube.com/watch?v=8a3KX2InUqY

## **Appendices**

Appendix A

* 
* 