Project Report on

‘MAZE SOLVING BOT’

Submission to **THE ROBOTICS CLUB** as a part of

EMBEDDED SYSTEMS
TECHNICAL ADVISORY BOARD’24

THE ROBOTICS CLUB-SNIST

SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY

(AUTONOMOUS)

(Affiliated to JNTU University, Hyderabad)

Yamnampet, Ghatkesar, Hyderabad – 501301.

2024

CERTIFICATE

This is the project work titled **MAZE SOLVING BOT** by **P.ANIRUDH, K.PAVAN RAJU, V.ANANTH JEETH, J.MANICHAND, MOULI, SUJANA BANDI, MADAR** under the guidance of **MURUGAN SARAVANAN** for the recruitment into the **TECHNICAL ADVISORY BOARD** and is a record of the project work carried out by them during the year 2023-2024 as a part of **TAB** under the supervision of

RAKESH BANGARU	**SAATHVIKA REVALLA**
TAB Chairman	**TAB Vice-Chairman**

Mr. N V V S Narayana

The President of THE ROBOTICS CLUB

Dr. A. PURUSHOTHAM
Technical Advisor
Mechanical Department

DECLARATION

The project work reported in the present thesis titled **“MAZE SOLVING BOT”** is a record of work done by Embedded Systems in **THE ROBOTICS CLUB** as a part of **TECHNICAL ADVISORY BOARD.**

No part of the thesis is copied from books/ journals/ Internet and wherever the portion is taken, the same has been duly referred in the text. The report is based on the project work done entirely by TEAM EMBEDDED SYSTEMS and not copied from any other source.

ACKNOWLEDGMENT

This project report is the outcome of the efforts of many people who have driven our passion to explore into implementation of **'MAZE SOLVING BOT'.** We have received great guidance, encouragement and support from them and have learned a lot because of their willingness to share their knowledge and experience. Primarily, we would like to express our gratitude to our mentor '**MURUGAN SARAVANAN**'. His guidance has been of immense help in surmounting various hurdles along the path of our goal.

We thank our TAB chairman **RAKESH BANGARU,** vice-chairman **SAATHVIKA REVALLA, and** supervisors **GUMMADELLY BHARGAV, DOGIPARTHI DEEKSHITH** for being with us till the end of the project completion.

We also thank our Faculty advisor **Dr. A. PURUSHOTHAM**, Professor Mechanical Department, who encouraged us during this project by rendering his help when needed.

Contents

ABSTRACT
THE ROBOTICS CLUB

EMBEDDED SYSTEMS
(Maze Solving Bot)

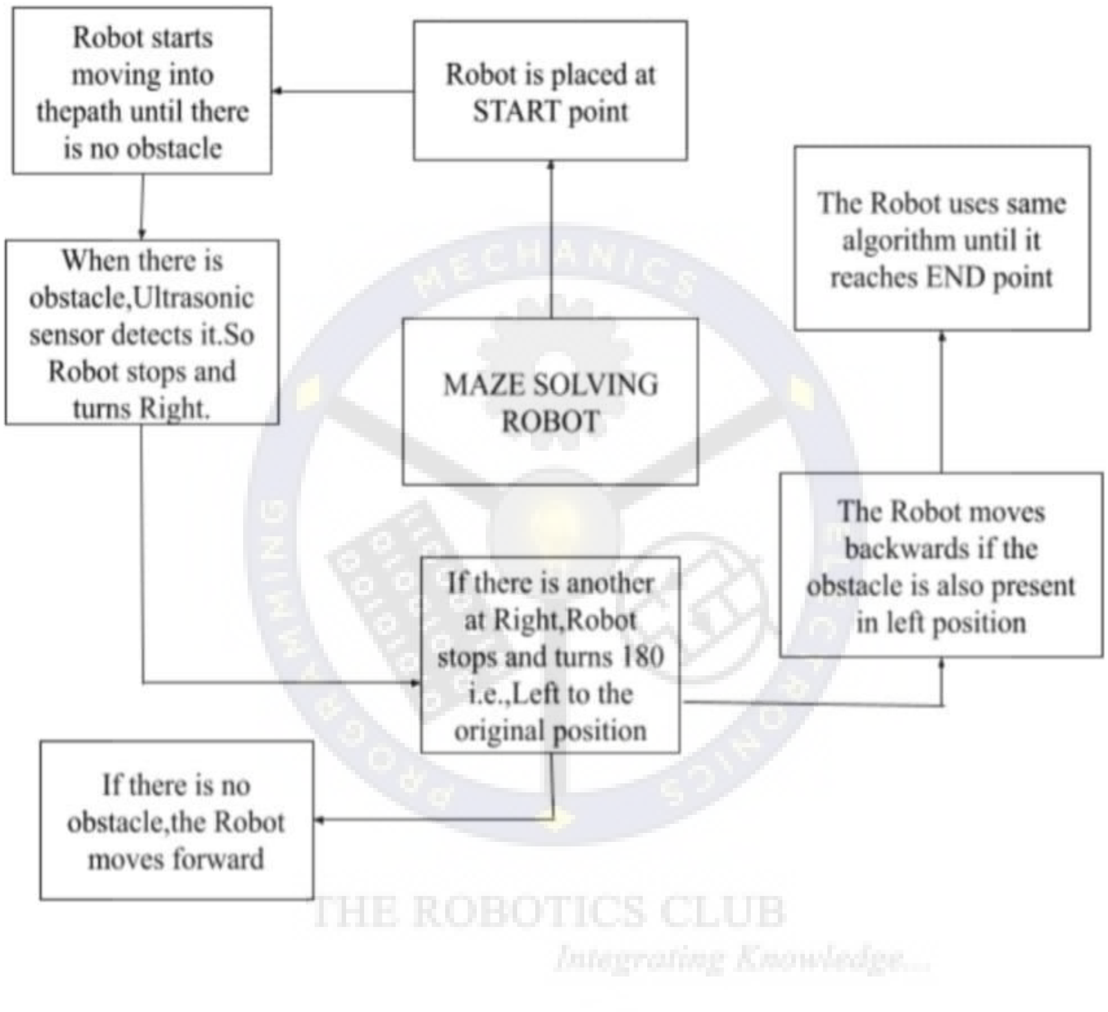
The Problem Statement:

The project's central objective is to design and develop an autonomous robot capable of navigating through the complexities of a maze. Military operations often involve navigating through complex and hazardous environments where the presence of obstacles, such as debris, barriers, and uneven terrain, can pose significant challenges. An autonomous maze-solving robot equipped with the STM32F446RE microcontroller provides a solution by autonomously navigating these environments, avoiding obstacles, and gathering crucial intelligence. Such a robot can be deployed in urban warfare for building clearance, disaster response to locate survivors, and explosive ordnance disposal to identify and map explosive devices.

The Approach:

When placed at the start position, the robot begins moving along the path until it encounters an obstacle. This is detected by the ultrasonic sensor, causing the robot to stop and turn right. If there's another obstacle to the right, the robot stops and turns 180 degrees, reverting to its original position. If the path is clear, the robot advances. However, if an obstacle is also present to the left, the robot moves backward. This algorithm continues until the robot reaches the endpoint.

BLOCK DIAGRAM:


Robot starts moving into thepath until there is no obstacle
Robot is placed at START point
The Robot uses same algorithm until it reaches END point
When there is obstacle,Ultrasonic sensor detects it.So Robot stops and turns Right.
MAZE SOLVING ROBOT
The Robot moves backwards if the obstacle is also present in left position
If there is another at Right,Robot stops and turns 180 i.e.,Left to the original position
If there is no obstacle,the Robot moves forward

Maze Solving Bot

J. Manichand, K. Pavan Raju, P. Anirudh, I.S.S. Mouli, V. Ananth Jeeth, Sujana, Madar

June 28, 2024

Abstract

The project's central objective is to design and develop an autonomous robot capable of navigating through the complexities of a maze. Military operations often involve navigating through complex and hazardous environments where the presence of obstacles, such as debris, barriers, and uneven terrain, can pose significant challenges. An autonomous maze-solving robot equipped with the STM32F446RE microcontroller provides a solution by autonomously navigating these environments, avoiding obstacles, and gathering crucial intelligence. Such a robot can be deployed in urban warfare for building clearance, disaster response to locate survivors, and explosive ordnance disposal to identify and map explosive devices.

1 Introduction

A maze is a network of paths, typically from an entrance to exit. The concept of Maze approximately thousand years old Which was invented in Egypt. From then, many mathematician made various algorithm to solve the maze. Now a days, maze solving problem is an important field of robotics. It is based on one of the most important areas of robot, which is "Decision Making Algorithm". Cause, this robot will be placed in unknown place, and it requires to have a good decision making capability. There are many types of maze solving robot using various type of algorithms. In this project, the system design of Maze solving robot consist obstacle avoidance ultrasonic sensors and then sensors will detect the wall. To solve the maze, this robot will apply wall following algorithms such as left or right hand rule. It will also follow the Flood fill algorithm for finding the shortest path.

1.1 Existing System

Pre-existing mechanisms for autonomous maze-solving robots include Wall-Following, where robots use sensors to maintain a consistent distance to a wall, and the Flood Fill algorithm, which maps the maze with distance values to find the optimal path. The Right-Hand Rule involves keeping the right hand in contact with the wall to ensure finding the exit. Depth-First Search (DFS) explores each branch to its end before backtracking, while Breadth-First Search (BFS) explores all paths level by level for the shortest route. Micromouse competitions demonstrate these techniques, showcasing robots that use sensors, microcontrollers, and motors to navigate mazes autonomously.

1.2 Proposed System

An Autonomous Maze Solving Bot using a Nucleo board, ultrasonic sensors, and BO motors operates by leveraging sensor data, motor control, and pathfinding algorithms. The bot is equipped with ultrasonic sensors that emit sound waves and measure the distance to obstacles based on the echo received. This real-time data helps in detecting walls and open paths in the maze. The Nucleo board processes this sensory input to map the maze environment dynamically. The Nucleo board sends signals to the BO motors, which drive the bot's wheels, enabling precise movements and directional changes. As the bot explores the maze, it continually updates its path based on sensor feedback, ensuring it avoids collisions and adjusts its route as necessary. This integration of sensors, motor control, and intelligent processing enables the bot to navigate and solve the maze autonomously..

2 Architecture

2.1 Hardware

2.1.1 STM32F446RE-Nucleo Board

The STM32F446xC/E devices are based on the high-performance Arm Cortex-M4 32-bit RISC core operating at a frequency of up to 180 MHz. The Cortex-M4 core features a floating point unit (FPU) single precision supporting all Arm® single-precision data-processing instructions and data types. It also implements a full set of DSP instructions and a memory protection unit (MPU) that enhances application security. The STM32F446xC/E devices incorporate high-speed embedded memories (Flash memory up to 512 Kbytes, up to 128 Kbytes of SRAM), up to 4 Kbytes of backup SRAM, and an extensive range of enhanced I/Os and peripherals connected to two APB buses, two AHB buses and a 32-bit multi-AHB bus matrix. All devices offer three 12-bit ADCs, two DACs, a low-power RTC, twelve general-purpose 16-bit timers including two PWM timers for motor control, two general-purpose 32-bit timers. They also feature standard and advanced communication interfaces.

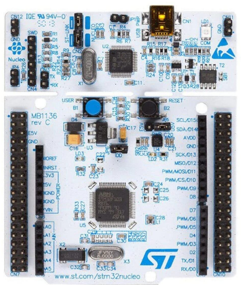


Figure 1: STM32F446RE-Nucleo Board

2.1.2 Bread board

An electronics breadboard (as opposed to the type on which sandwiches are made) is actually referring to a solder-less breadboard. These are great units for making temporary circuits and prototyping, and they require absolutely no soldering.

Figure 2: Bread board

2.1.3 Ultrasonic sensor

An ultrasonic sensor is an instrument that measures the distance to an object using ultrasonic sound waves. An ultrasonic sensor uses a transducer to send and receive ultrasonic pulses that relay back information about an object's proximity.

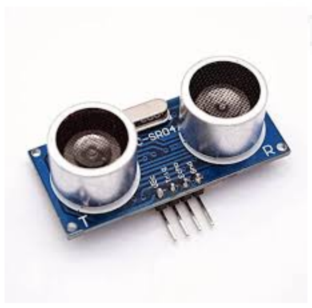


Figure 3: Ultrasonic sensor

2.1.4 BO Motor

The 300 RPM single Shaft BO Motor - Straight motor gives good torque and rpm at lower operating voltages, which is the biggest advantage of these motors.Small shaft with matching wheels gives an optimized design for your application or robot. Mounting holes on the body light weight makes it suitable for in-circuit placement. This motor can be used with 69mm Diameter Wheel for Plastic Gear Motors.

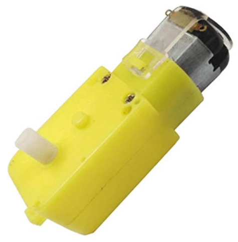


Figure 4: BO Motors

2.1.5 Ball Caster Wheel

A ball caster wheel consists of a rotating ball encased in a housing, enabling omnidirectional movement. Commonly used in robotics, furniture, and transport systems, it allows for smooth, multidirectional maneuverability on flat surfaces. This design reduces friction, enhances mobility, and supports efficient weight distribution, facilitating seamless motion and positioning.

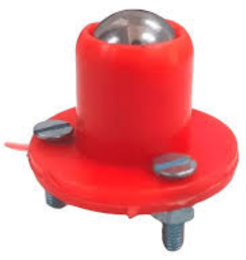


Figure 5: Ball caster Wheel

2.1.6 Wheels

Wheels with good grip are used so that the bot moves freely through the pipes.They reduce friction, making it easier to transport heavy loads and navigate obstacles. We can traverse through rough terrain as they have good grip. The ones we are using are made of plastic and rubber.

Figure 6: Wheels

.

2.1.7 Lead Acid battery

A lead-acid battery is a rechargeable energy storage device using lead plates and sulfuric acid electrolyte. Widely used in automotive and backup power applications, it offers high power output, reliability, and cost-effectiveness. Despite its weight and maintenance needs, it remains popular due to its durability and ability to deliver consistent power.

Figure 7: Lead Acid battery

2.1.8 Motor Driver - L298N

The L298N is a dual H-bridge motor driver integrated circuit (IC) commonly used to control and drive DC motors, particularly in robotics and mechanics applications. It provides a straightforward way to manage motor direction and speed using control signals from micro-controllers like Arduino. The L298N motor driver has a supply range of 5v to 35v and is capable of 2A continuous current per channel, so it works well with most of the DC motors. The L298N is valued for its efficiency, ease of use, and versatility, making it a popular choice for hobbyists and engineers working on projects involving motor control.

Figure 8: Motor Driver - L298N

2.1.9 Jumper Wires

A jumper wire is an electrical wire or group of wires used to connect circuits without soldering. They have connectors or pins at their ends. Depending upon the configuration of end connectors, they are classified into three types: male-to-male, male-to-female and female-to-female.

Figure 9: Jumper Wires

2.2 Software

2.2.1 STM32 CubeIDE

Figure 10: STM32 CubeIDE

2.2.2 Fusion 360

Fusion 360 is a computer-aided designing (CAD) software application for 3-D modelling and simulation. It's other functions include computer-aided manufacturing (CAM) and computer-aided engineering (CAE), as well as designing printed circuit boards.

Fusion 360 provides powerful 3D modeling tools that allow users to create complex 3D models of products and components. It supports parametric modeling, direct modeling, and sculpting.

Figure 11: Fusion 360

2.2.3 Fritzing

Fritzing is an open-source electronic design automation (EDA) software to design electronics hardware, printed circuit boards and schematic circuit diagrams. It is an offshoot of the Processing programming language and the Arduino microcontroller.

Fritzing allows users to create electronic circuits by dragging and dropping various components (e.g., resistors, LEDs, microcontrollers) onto a virtual breadboard. Users can connect components by drawing wires to represent electrical connections.

Figure 12: Fritzing

3 Implementation and working:

3.1 Circuit diagram

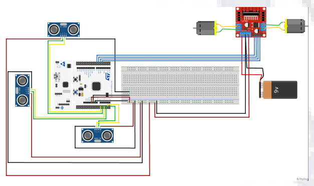
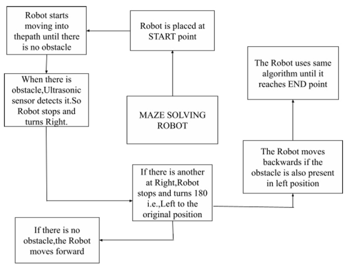


Figure 13: circuit diagram

3.2 Block diagram



Robot is placed at START point
Robot starts moving into thepath until there is no obstacle
When there is obstacle,Ultrasonic sensor detects it.So Robot stops and turns Right.
MAZE SOLVING ROBOT
If there is another at Right,Robot stops and turns 180 i.e.,Left to the original position
If there is no obstacle,the Robot moves forward
The Robot moves backwards if the obstacle is also present in left position
The Robot uses same algorithm until it reaches END point

Figure 14: Block diagram

3.3 Working Mechanism

The three ultrasonic sensors are positioned at the front, left, and right of the robot to detect distances to walls and obstacles. These sensors continuously send distance data to the Nucleo board, which processes the information to map the surroundings. Using algorithms such as Wall-Following or the Right-Hand Rule, the Nucleo board determines the optimal path through the maze. The motor driver receives control signals from the Nucleo board to adjust the speed and direction of the BO motors, enabling precise movements and turns. As the robot navigates, it updates its path in real-time based on the sensor feedback, ensuring it avoids collisions and efficiently finds the maze exit. This integration of ultrasonic sensors, motor driver, and intelligent algorithms allows the robot to autonomously solve the maze.

4 Experimental Results and Conclusions:

4.1 Results

The autonomous maze-solving robot, equipped with three ultrasonic sensors, a Nucleo board, and a motor driver, successfully navigated various maze configurations. The ultrasonic sensors provided accurate and real-time distance measurements, enabling effective wall detection and obstacle avoidance. The Nucleo board processed sensor data efficiently and implemented the Wall-Following algorithm, resulting in reliable pathfinding. The motor driver facilitated smooth and precise control of the BO motors, ensuring accurate movements and turns. The robot consistently reached the maze exit, demonstrating robust performance and adaptability to different maze layouts.

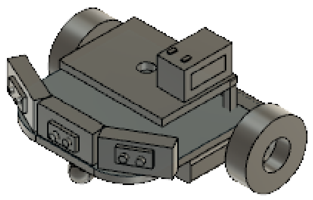


Figure 15: CAD Digram

4.2 Future Enhancements

Future enhancements for an autonomous maze-solving bot include advanced sensor integration (LiDAR, IMUs), machine learning for adaptive behavior, swarm intelligence for collaboration, SLAM for mapping, energy efficiency (battery management, harvesting), communication (Wi-Fi, cloud), user interface upgrades (AR), robust hardware (modular, durable), and safety measures (failsafes, compliance).

4.3 Conclusion

The project successfully demonstrated the capability of an autonomous robot to solve mazes using ultrasonic sensors, a Nucleo board, and a motor driver. The integration of these components allowed for real-time obstacle detection, efficient pathfinding, and precise motor control. The robot's consistent performance across various maze configurations highlights the effectiveness of the chosen algorithms and hardware setup. Future improvements could include enhancing the algorithm for faster solving times and incorporating additional sensors for even more accurate mapping and navigation.

4.4 Source Code

```
#include "main.h"
#include "stdio.h"
TIM_HandleTypeDef htim1;
TIM_HandleTypeDef htim4;
uint32_t readUltrasonic(GPIO_TypeDef* GPIOx,
uint16_t GPIO_Pin_Trigger,
uint16_t GPIO_Pin_Echo) {
  HAL_GPIO_WritePin(GPIOx, GPIO_Pin_Trigger,
  GPIO_PIN_RESET);
  delay_us(2);
  HAL_GPIO_WritePin(GPIOx, GPIO_Pin_Trigger,
  GPIO_PIN_SET);
  delay_us(10);
  HAL_GPIO_WritePin(GPIOx, GPIO_Pin_Trigger,
  GPIO_PIN_RESET);

  while (HAL_GPIO_ReadPin(GPIOx, GPIO_Pin_Echo)
  ==  GPIO_PIN_RESET);
  uint32_t start = __HAL_TIM_GET_COUNTER(&htim1);

  while (HAL_GPIO_ReadPin(GPIOx, GPIO_Pin_Echo)
  == GPIO_PIN_SET);
  uint32_t end = __HAL_TIM_GET_COUNTER(&htim1);

  uint32_t distance = (end - start) * 0.034 / 2;
  return distance;
}

void moveForward(void) {
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0,
  GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1,
  GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_7,
  GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8,
  GPIO_PIN_RESET);


}


void moveBackward() {
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_4,
  GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_5,
  GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_11,
  GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_12,
  GPIO_PIN_SET);



}

void turnLeft() {
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_4, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_5, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_11, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_12, GPIO_PIN_RESET);


}

void turnRight() {
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_4, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_5, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_11, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_12, GPIO_PIN_SET);

}

void stop() {
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_4, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_5, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_11, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_12, GPIO_PIN_RESET);


}

uint32_t distanceFront;
uint32_t distanceLeft;
uint32_t distanceRight;


int main(void)
{
  /* USER CODE BEGIN 1 */
HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  //MX_TIM1_Init();
  MX_TIM1_Init();
  MX_TIM4_Init();
  //MX_I2C1_Init();

  HAL_TIM_Base_Start(&htim1);
  HAL_TIM_Base_Start(&htim4);
```

```
/* USER CODE END 1 */

/* MCU Configuration--------------------
------------------------------------*/

/* Reset of all peripherals, Initializes
the Flash interface and the Systick. */
HAL_Init();

/* USER CODE BEGIN Init */

/* USER CODE END Init */

/* Configure the system clock */
SystemClock_Config();

/* USER CODE BEGIN SysInit */

/* USER CODE END SysInit */

/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_TIM1_Init();
MX_TIM4_Init();
/* USER CODE BEGIN 2 */

/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
  /* USER CODE END WHILE */
/* USER CODE END WHILE */
       uint32_t distanceFront =
  readUltrasonic(GPIOA, GPIO_PIN_8,
  GPIO_PIN_9);
       uint32_t distanceLeft =
  readUltrasonic(GPIOA, GPIO_PIN_10,
  GPIO_PIN_11);
       uint32_t distanceRight =
  readUltrasonic(GPIOB, GPIO_PIN_7,
  GPIO_PIN_6);

      if (distanceFront > 30) {
         moveForward();
    } else if (distanceLeft > distanceRight) {
         turnLeft();
    } else if (distanceLeft < distanceRight){
         turnRight();
    } else {
      moveBackward();
          }

       HAL_Delay(100);
```

4.5 List of Components:

SI NO.	COMPONENT NAME	QUANTITY
1	Ball Caster Wheel	1
2	Ultrasonic sensors	3
3	BO motors (Center Shaft) 300 RPM	2
4	Wheels	2
5	Metal Chassis	1
6	L298N Motor Driver	1
7	STM32F446RE-Nucleo Board	1
8	Jumper Wires	1
9	Bread Board	1
10	12V Lead Acid Battery	1