All-Terrain Metal Detector Bot using ESP32 Microcontroller

Design and implementation of an autonomous robot to navigate various terrains while simultaneously detecting buried metallic objects

Vishwajeith S
Dept. of Electronics and Communication Engineering
College of Engineering, Guindy
Chennai, India
svishwajeith03@gmail.com

John Felix A
Dept. of Electronics and Communication Engineering
College of Engineering, Guindy
Chennai, India
ajohnfelix110304@gmail.com

Pragatheesh A.P
Dept. of Electronics and Communication Engineering
College of Engineering, Guindy
Chennai, India
appragatheesh@gmail.com

Harivaradan V La
Dept. of Electronics and Communication Engineering
College of Engineering, Guindy
Chennai, India

In this paper, the design, implementation, and applications of an All-Terrain Metal detector robot using microcontrollers ESP32 has been explored

I. INTRODUCTION

The combination of advanced sensing technologies and intelligent mobility systems has prepared the stage for game-changing advances in the field of modern robotics. The All-Terrain Metal Detector Bot is one such extraordinary product, an autonomous robotic platform built to explore and navigate varied environments while detecting and identifying buried metallic objects. This cutting-edge robot combines cutting-edge technology with sophisticated software algorithms to provide an unmatched level of adaptability and precision in metal detection applications.

The All-Terrain Metal Detector Bot is envisioned as a game-changing solution for a wide range of industries, including archaeological excavations and geophysical surveys, as well as security and mining operations. Its capacity to navigate difficult terrains, along with its high-sensitivity metal detection skills, enables it to unearth hidden artefacts and identify precious resources, and make substantial contributions to a wide range of scientific and industrial endeavours. This research goes into the intricate design, engineering principles, and operational features of this exceptional robotic system, emphasising its potential to change the way we approach metal detection in a variety of sectors.

II. PROPOSED SYSTEM

The proposed system introduces a cutting-edge all-terrain metal detector rover, meticulously designed with a robust framework using lightweight yet durable 1" PVC components. This agile rover is fortified with six high-performance 12V DC metal gear motors, enabling seamless mobility across various terrains. Its power source is a reliable lead-acid battery, providing the necessary energy to sustain prolonged operations. The rover's intelligence is orchestrated by an ESP32 microcontroller, ensuring precise and efficient control over its movements. The microcontroller is seamlessly interfaced with an L298n motor driver, facilitating seamless coordination between the motors for optimal performance. A pivotal feature of this rover is its integrated metal detection system, enhancing its functionality for diverse applications. The lightweight PVC frame not only ensures structural integrity but also contributes

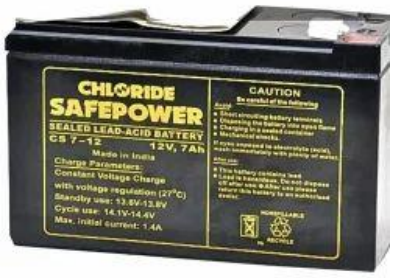
to the rover's overall agility. The six 125mm robot wheels, driven by the potent 12V DC motors, empower the rover to navigate through challenging terrains with ease. This system harmonizes mobility, durability, and advanced control mechanisms, making it an invaluable tool for applications ranging from archaeological exploration to security surveillance, where precise metal detection and agile maneuverability are paramount

III. KEY COMPONENTS

A. ***Figure 1:*** *Lightweight PVC frame for structural integrity enabling the robot to move on various terrains*

B. ***Figure 2:*** *Six 125mm robot wheels powered by 12V DC motors*

C. ***Figure 3:*** *Efficiently driven by a 12V DC lead-acid battery*



D. ***Figure 4:*** *ESP32 microcontroller for precise motor control capable of connecting with a mobile device via Wi-Fi/Bluetooth*

E. ***Figure 5:*** *Integration of a metal detection system which utilizes A88 metal detector. It primarily detects copper and Fe-based metals*

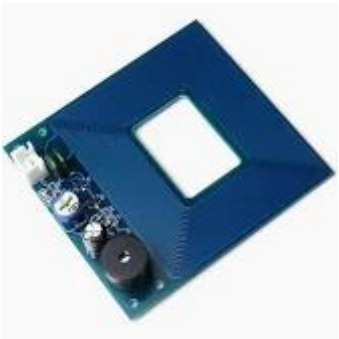
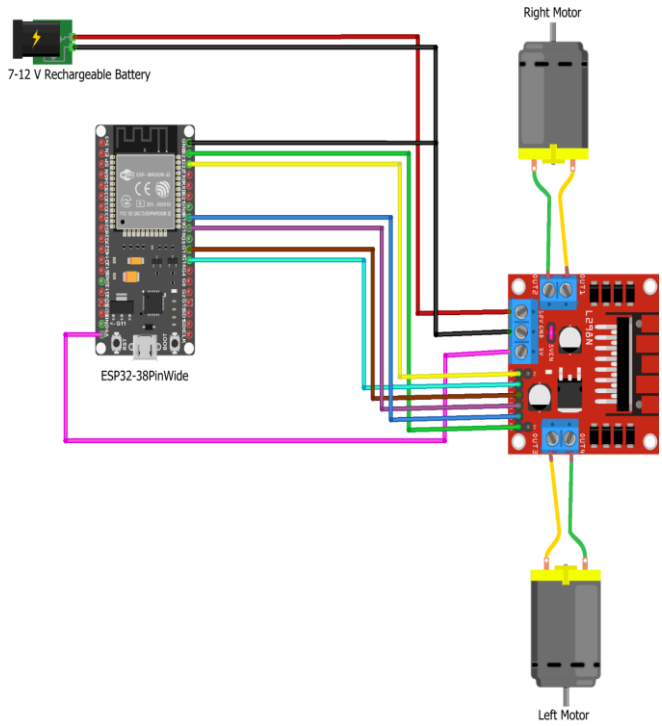


Table 1: Components used in construction

COMPONENTS	NO'S	SPECIFICATIONS
ESP - 32	1	UNO
DC GEAR MOTOR	6	12 VOLT
LEAD ACID BATTERY	1	12 VOLT
METAL DETECTOR	1	A88
ROBO WHEELS	6	
MOTOR DRIVER	1	L298N
PVC ELBOW	6	1 INCH
PVC ELBOW-45 Degree	6	1 INCH
PVC COUPLING	2	1 INCH
PVC CLAMP	6	1 INCH
WIRES		5 M
NUT AND BOLTS	AS REQUIRED	
PVC PIPE-6 FEET	1	1 INCH

FIGURE 6: CIRCUIT SCHEMATIC



IV. IMPLEMENTATION WORK

The design and assembly of the chassis was the first step in the realization of the all-terrain metal detector rover. To build the framework, sturdy 1" PVC components were meticulously cut and drilled, and the assembly was attached using nuts and bolts, ensuring a lightweight yet lasting structure.

The integration of six wheels, each powered by a 12V DC motor, is at the heart of the rover's operation. To accurately coordinate the movements of all six wheels, the ESP32 microcontroller, a strong and adaptable control unit, was interfaced with the L298N motor driver. This integration enabled smooth control and maneuverability.

The software component of the implementation was carried out with the Arduino IDE, and the ESP32 microcontroller was programmed to govern the rover's actions. The code provided provisions for speed control, which is critical for adjusting to various terrains and scenarios.

The rover was outfitted with a Wi-Fi module to improve human control, allowing it to be commanded manually from a mobile device. This function provides a layer of versatility to the rover's operations, allowing users to remotely steer it through a variety of settings.

A metal detector with a buzzer was added to the rover's capabilities as a useful feature. This metal detecting technology improves the rover's capability by allowing it to detect and signal the presence of metal items.

Precision in chassis design, effective motor control via the ESP32 microcontroller, seamless coding using the Arduino IDE, provision for speed control, mobile-based manual control via Wi-Fi, and the integration of a metal detector with a buzzer were all part of the implementation process. This comprehensive approach ensures that the all-terrain metal detector rover is not only tough and agile, but also equipped with features that increase its utility in a variety of applications.

V. WORKING

The all-terrain metal detector rover runs smoothly thanks to a combination of tough hardware and clever software. The implementation makes use of a strong chassis made of 1" PVC components that are precision cut and drilled and secured with nuts and bolts. Six 125mm robot wheels, each powered by a 12V DC metal gear motor, power the core mobility, which is meticulously controlled by an ESP32 microcontroller interfaced with an L298N motor driver.

The ESP32 microcontroller, which was programmed using the Arduino IDE, serves as the rover's brain. The coding gives the rover precise control over its movements, including the critical capability of speed control to adapt to changing terrain. The manual control feature enabled by a Wi-Fi module distinguishes this rover. Users can control the rover wirelessly with a mobile device, providing flexibility and adaptability in navigating different environments.

In addition, the rover is outfitted with a metal detection system that includes a buzzer. This system improves its functionality by detecting and signaling the presence of metal objects in its vicinity. The addition of a metal detector adds an additional layer of utility, making the rover suitable for applications involving the identification of metallic elements.

VI. FINAL OUTPUT

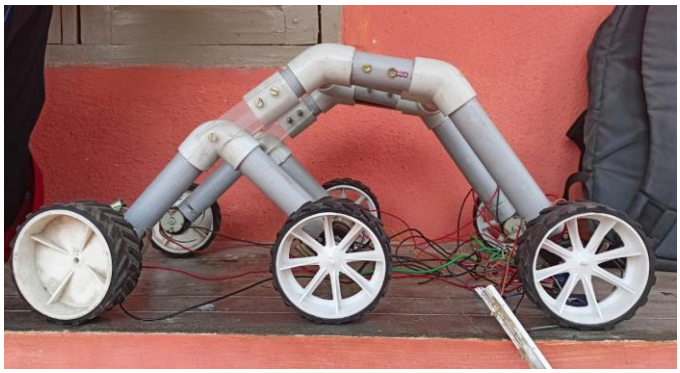


Figure 7: Bot in stationary position

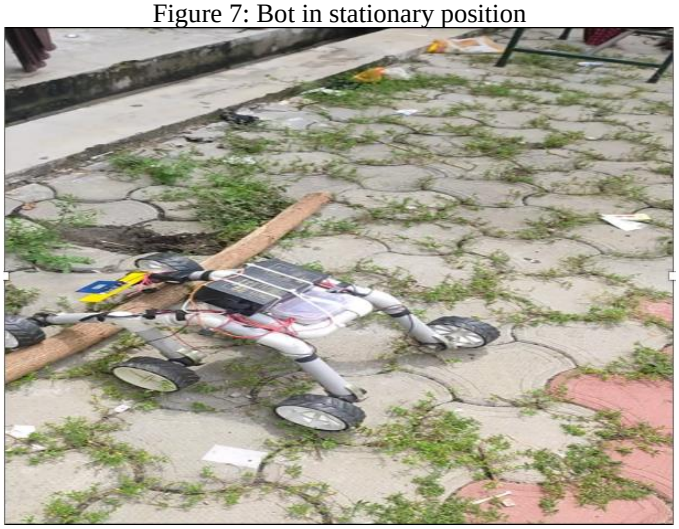


Figure 8: Bot crossing an obstacle (wooden log)

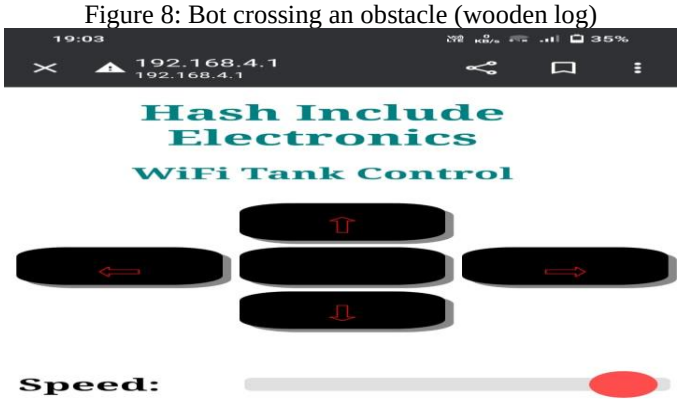


Figure 9: Mobile application (Wi-Fi enabled) to control speed and direction of the robot

VII. APPLICATIONS

*1) **Archaeology and Historical Preservation:** The bot can be used to locate buried artifacts, coins, and historical relics in archaeological sites, helping researchers and archaeologists*

*2) **Search and Rescue Operations**: In disaster-stricken areas, the bot can help locate buried survivors, as well as identify buried objects that may pose a danger to rescuer*

*3) **Geological Exploration:** The bot could be used in geological surveys to identify mineral deposits, ores, or underground resources*

*4) **Mine Clearance:** The bot can play a vital role in demining efforts by detecting buried landmines in post-conflict zones*

*5) **Environmental Monitoring**: It can assist in detecting buried metal pollutants, hazardous materials, or industrial waste that might be affecting soil or water quality*

VIII. CONCLUSION

In conclusion, the All-Terrain Metal Detector Rover exemplifies innovation and engineering excellence. The tough PVC chassis, which is powered by 12V DC motors and controlled by an intelligent ESP32 microcontroller, ensures precision when navigating various terrains.

The addition of manual control via a Wi-Fi module adds a modern touch, increasing adaptability for users who can control the rover seamlessly via mobile devices. The addition of a buzzer to a metal detection system expands its utility, making it suitable for applications ranging from hobbyist metal detection to complex scenarios.

This rover serves as a model for future projects, demonstrating the seamless integration of structural design, intelligent control, and innovative features. It is a testament to a dedication to excellence in all aspects of its conception and execution.

IX. CODE

A. CODE

```
#include <Arduino.h>
#ifdef ESP32
#include <WiFi.h>
#include <AsyncTCP.h>
#elif defined(ESP8266)
#include <ESP8266WiFi.h>
#include <ESPAsyncTCP.h>
#endif
#include <ESPAsyncWebServer.h>

#include <iostream>
#include <sstream>

struct MOTOR_PINS
{
  int pinEn;
  int pinIN1;
  int pinIN2;
};

std::vector<MOTOR_PINS> motorPins =
{
  {22, 16, 17},  //RIGHT_MOTOR Pins (EnA, IN1, IN2)
  {23, 18, 19},  //LEFT_MOTOR  Pins (EnB, IN3, IN4)
};

#define UP 1
#define DOWN 2
#define LEFT 3
#define RIGHT 4
#define STOP 0

#define RIGHT_MOTOR 0
#define LEFT_MOTOR 1

#define FORWARD 1
#define BACKWARD -1

const int PWMFreq = 1000; /* 1 KHz */
const int PWMResolution = 8;
const int PWMSpeedChannel = 4;

const char* ssid     = "MyWiFiCar";
const char* password = "12345678";

AsyncWebServer server(80);
AsyncWebSocket wsCarInput("/CarInput");

const char* htmlHomePage PROGMEM =
R"HTMLHOMEPAGE(
<!DOCTYPE html>
<html>
  <head>
  <meta name="viewport"
content="width=device-width, initial-
scale=1, maximum-scale=1, user-
scalable=no">
    <style>
    .arrows {
      font-size:40px;
      color:red;
    }
    td.button {
      background-color:black;
      border-radius:25%;
      box-shadow: 5px 5px #888888;
    }
    td.button:active {
      transform: translate(5px,5px);
      box-shadow: none;
    }

    .noselect {
      -webkit-touch-callout: none; /* iOS
Safari */
        -webkit-user-select: none; /*
Safari */
         -khtml-user-select: none; /*
Konqueror HTML */
           -moz-user-select: none; /*
Firefox */
            -ms-user-select: none; /*
Internet Explorer/Edge */
                user-select: none; /* Non-
prefixed version, currently
```

```
                                        suppo
rted by Chrome and Opera */
    }

    .slidecontainer {
      width: 100%;
    }

    .slider {
      -webkit-appearance: none;
      width: 100%;
      height: 20px;
      border-radius: 5px;
      background: #d3d3d3;
      outline: none;
      opacity: 0.7;
      -webkit-transition: .2s;
      transition: opacity .2s;
    }

    .slider:hover {
      opacity: 1;
    }

    .slider::-webkit-slider-thumb {
      -webkit-appearance: none;
      appearance: none;
      width: 40px;
      height: 40px;
      border-radius: 50%;
      background: red;
      cursor: pointer;
    }

    .slider::-moz-range-thumb {
      width: 40px;
      height: 40px;
      border-radius: 50%;
      background: red;
      cursor: pointer;
    }

    </style>

  </head>
```

```
  <body class="noselect" align="center"
style="background-color:white">

    <h1 style="color: teal;text-
align:center;">Hash Include
Electronics</h1>
    <h2 style="color: teal;text-
align:center;">WiFi Tank Control</h2>

    <table id="mainTable"
style="width:400px;margin:auto;table-
layout:fixed" CELLSPACING=10>
      <tr>
        <td></td>
        <td class="button"
ontouchstart='sendButtonInput("MoveCar","1"
)'
ontouchend='sendButtonInput("MoveCar","0")'
><span class="arrows" >&#8679;</span></td>
        <td></td>
      </tr>
      <tr>
        <td class="button"
ontouchstart='sendButtonInput("MoveCar","3"
)'
ontouchend='sendButtonInput("MoveCar","0")'
><span class="arrows" >&#8678;</span></td>
        <td class="button"></td>
        <td class="button"
ontouchstart='sendButtonInput("MoveCar","4"
)'
ontouchend='sendButtonInput("MoveCar","0")'
><span class="arrows" >&#8680;</span></td>
      </tr>
      <tr>
        <td></td>
        <td class="button"
ontouchstart='sendButtonInput("MoveCar","2"
)'
ontouchend='sendButtonInput("MoveCar","0")'
><span class="arrows" >&#8681;</span></td>
        <td></td>
      </tr>
      <tr/><tr/>
      <tr/><tr/>
      <tr/><tr/>
```

```
      <tr>
        <td style="text-align:left;font-
size:25px"><b>Speed:</b></td>
        <td colspan=2>
         <div class="slidecontainer">
            <input type="range" min="0"
max="255" value="150" class="slider"
id="Speed"
oninput='sendButtonInput("Speed",value)'>
          </div>
        </td>
      </tr>
    </table>

    <script>
      var webSocketCarInputUrl = "ws:\/\/"
+ window.location.hostname +
"/CarInput";
      var websocketCarInput;

      function initCarInputWebSocket()
      {
        websocketCarInput = new
WebSocket(webSocketCarInputUrl);
        websocketCarInput.onopen    =
function(event)
        {
          var speedButton =
document.getElementById("Speed");
          sendButtonInput("Speed",
speedButton.value);
        };
        websocketCarInput.onclose   =
function(event){setTimeout(initCarInputWebS
ocket, 2000);};
        websocketCarInput.onmessage =
function(event){};
      }

      function sendButtonInput(key, value)
      {
        var data = key + "," + value;
        websocketCarInput.send(data);
      }
      window.onload =
initCarInputWebSocket;
      document.getElementById("mainTable").
addEventListener("touchend",
function(event){
        event.preventDefault()
      });
    </script>
  </body>
</html>
)HTMLHOMEPAGE";


void rotateMotor(int motorNumber, int
motorDirection)
{
  if (motorDirection == FORWARD)
  {
    digitalWrite(motorPins[motorNumber].pin
IN1, HIGH);
    digitalWrite(motorPins[motorNumber].pin
IN2, LOW);
  }
  else if (motorDirection == BACKWARD)
  {
    digitalWrite(motorPins[motorNumber].pin
IN1, LOW);
    digitalWrite(motorPins[motorNumber].pin
IN2, HIGH);
  }
  else
  {
    digitalWrite(motorPins[motorNumber].pin
IN1, LOW);
    digitalWrite(motorPins[motorNumber].pin
IN2, LOW);
  }
}

void moveCar(int inputValue)
{
  Serial.printf("Got value as %d\n",
inputValue);
  switch(inputValue)
  {
```

```
    case UP:
      rotateMotor(RIGHT_MOTOR, FORWARD);
      rotateMotor(LEFT_MOTOR,
FORWARD);
      break;

    case DOWN:
      rotateMotor(RIGHT_MOTOR, BACKWARD);
      rotateMotor(LEFT_MOTOR, BACKWARD);
      break;

    case LEFT:
      rotateMotor(RIGHT_MOTOR, FORWARD);
      rotateMotor(LEFT_MOTOR, BACKWARD);
      break;

    case RIGHT:
      rotateMotor(RIGHT_MOTOR, BACKWARD);
      rotateMotor(LEFT_MOTOR, FORWARD);
      break;

    case STOP:
      rotateMotor(RIGHT_MOTOR, STOP);
      rotateMotor(LEFT_MOTOR, STOP);
      break;

    default:
      rotateMotor(RIGHT_MOTOR, STOP);
      rotateMotor(LEFT_MOTOR, STOP);
      break;
  }
}

void handleRoot(AsyncWebServerRequest
*request)
{
  request->send_P(200, "text/html",
htmlHomePage);
}

void handleNotFound(AsyncWebServerRequest
*request)
{
    request->send(404, "text/plain", "File
Not Found");
}

void
onCarInputWebSocketEvent(AsyncWebSocket
*server,
                         AsyncWebSocketClient
*client,
                         AwsEventType type,
                         void *arg,
                         uint8_t *data,
                         size_t len)
{
  switch (type)
  {
    case WS_EVT_CONNECT:
      Serial.printf("WebSocket client #%u
connected from %s\n", client->id(), client-
>remoteIP().toString().c_str());
      break;
    case WS_EVT_DISCONNECT:
      Serial.printf("WebSocket client #%u
disconnected\n", client->id());
      moveCar(STOP);
      break;
    case WS_EVT_DATA:
      AwsFrameInfo *info;
      info = (AwsFrameInfo*)arg;
      if (info->final && info->index == 0
&& info->len == len && info->opcode ==
WS_TEXT)
      {
        std::string myData = "";
        myData.assign((char *)data, len);
        std::istringstream ss(myData);
        std::string key, value;
        std::getline(ss, key, ',');
        std::getline(ss, value, ',');
        Serial.printf("Key [%s]
Value[%s]\n", key.c_str(), value.c_str());
        int valueInt =
atoi(value.c_str());
        if (key == "MoveCar")
        {
          moveCar(valueInt);
        }
        else if (key == "Speed")
        {
```

```
          ledcWrite(PWMSpeedChannel,
valueInt);
        }
      }
      break;
    case WS_EVT_PONG:
    case WS_EVT_ERROR:
      break;
    default:
      break;
  }
}

void setUpPinModes()
{
  //Set up PWM
  ledcSetup(PWMSpeedChannel, PWMFreq,
PWMResolution);

  for (int i = 0; i < motorPins.size();
i++)
  {
    pinMode(motorPins[i].pinEn,
OUTPUT);
    pinMode(motorPins[i].pinIN1, OUTPUT);
    pinMode(motorPins[i].pinIN2, OUTPUT);

    /* Attach the PWM Channel to the motor
enb Pin */
    ledcAttachPin(motorPins[i].pinEn,
PWMSpeedChannel);
  }
  moveCar(STOP);
}


void setup(void)
{
  setUpPinModes();
  Serial.begin(115200);

  WiFi.softAP(ssid, password);
  IPAddress IP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(IP);
  server.on("/", HTTP_GET, handleRoot);
  server.onNotFound(handleNotFound);

  wsCarInput.onEvent(onCarInputWebSocketEve
nt);
  server.addHandler(&wsCarInput);

  server.begin();
  Serial.println("HTTP server started");

}

void loop()
{
  wsCarInput.cleanupClients();
}
```

X. REFERENCES

[1] J. Li *et al*., "An all-terrain robot based on variable wheelbase design," *2022 41st Chinese Control Conference (CCC)*, Hefei, China, 2022, pp. 2882-2887, doi: 10.23919/CCC55666.2022.9901962.

[2] J. Suthakorn *et al*., "On the design and development of a rough terrain robot for rescue missions," *2008 IEEE International Conference on Robotics and Biomimetics*, Bangkok, Thailand, 2009, pp. 1830-1835, doi: 10.1109/ROBIO.2009.4913280.

[3] M. A. A. B. M. Hata and I. Baharin, "The study of a customisable all terrain mobile robot (ROBUST)," *2014 11th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, Kuala Lumpur, Malaysia, 2014, pp. 232-237, doi: 10.1109/URAI.2014.7057447.

[4] A. Pietikainen, A. Tikanmaki and J. Roning, "Design of the mechanics and sensor system of an autonomous all-terrain robot platform," *2008 IEEE International Conference on Robotics and Automation*, Pasadena, CA, USA, 2008, pp. 2325-2330, doi: 10.1109/ROBOT.2008.4543561.

[5] C. G. C. Carducci, A. Monti, M. H. Schraven, M. Schumacher and D. Mueller, "Enabling ESP32-based IoT Applications in Building Automation Systems," *2019 II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4.0&IoT)*, Naples, Italy, 2019, pp. 306-311, doi: 10.1109/METROI4.2019.8792852.

[6] T. Aoki, H. Yamato, M. Shimaoka and S. Mitsumori, "Study of Omni-directional all terrain mobile robot with globular metal spring wheel," *2013 IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, 2013, pp. 5606-5611, doi: 10.1109/ICRA.2013.6631382.