A logo for a university

Description automatically generated  
**Autonomous Path Finding Vehicle**

Internet of Things Project Proposal

Pero Schwitalla

BEng(H) in Software & Electronic Engineering 2nd Year

Atlantic Technological University

2023/2024

# Autonomous Vehicle Poster

A screenshot of a computer screen

Description automatically generated

# Project Hardware

A robot on a wood floor

Description automatically generated

# Declaration

This Project is presented is partial fulfilment of the requirements for the degree of Bachelor of Engineering is Software & Electronic Engineering at the Atlantic Technical University, Galway.

This project is my own work, except where otherwise accredited. Where the work of others has been used or incorporated during this project, this is acknowledged and referenced.

# Acknowledgments

I wish to acknowledge and thanks the people who have helped me with this project.

Nathasha Rohan

Brian O’Shea

Brian Costello

Ben Kinsella

Pat Hurney

Contents

[Autonomous Vehicle Poster 2](#_Toc176202281)

[Project Hardware 3](#_Toc176202282)

[Declaration 4](#_Toc176202283)

[Acknowledgments 5](#_Toc176202284)

[1. Summary 8](#_Toc176202285)

[2. Project Plan 9](#_Toc176202286)

[2.1 Semester 1 Plan 9](#_Toc176202287)

[2.2 Semester 2 Plan 9](#_Toc176202288)

[2.3 Changes to the plan 10](#_Toc176202289)

[3. Project Architecture 11](#_Toc176202290)

[3.1Architecture Diagram 11](#_Toc176202291)

[4. Development Platform and Tools 12](#_Toc176202292)

[4.1 Development Platform 12](#_Toc176202293)

[4.2 Development Platform (IDE) 12](#_Toc176202294)

[4.3 Web Development Platform 12](#_Toc176202295)

[4.4 Organising and Planning 12](#_Toc176202296)

[5. Sensors 13](#_Toc176202297)

[5.1 HSR-04 Ultrasonic Distance Sensors 13](#_Toc176202298)

[5.1.1 Ultrasonic Sensor code 14](#_Toc176202299)

[5.2 DHT-11 Temperature Sensor 14](#_Toc176202300)

[5.2.1 DHT11 Code 15](#_Toc176202301)

[5.3 QMC5883L Magnetometer 15](#_Toc176202302)

[5.3.1 QMC5883L Code 15](#_Toc176202303)

[6. Motors 16](#_Toc176202304)

[6.1 Servo Motors 16](#_Toc176202305)

[6.1.1 Servo Code 17](#_Toc176202306)

[7. Bluetooth Control 17](#_Toc176202307)

[8. Web Server 19](#_Toc176202308)

[8.1 Web Server Code 19](#_Toc176202309)

[9. Problem Solving 20](#_Toc176202310)

[9.1 Magnetometer Interference 20](#_Toc176202311)

[9.2 Ultrasonic Location System 21](#_Toc176202312)

[9.3 Vehicle Veering to one side 21](#_Toc176202313)

[9.4 ESP32 not connecting to PC 22](#_Toc176202314)

[10. Impact of Project on Sustainability 22](#_Toc176202315)

[11. Conclusion 23](#_Toc176202316)

[1. References 24](#_Toc176202317)

[Appendix 1: Code 25](#_Toc176202318)

[Appendix 2: Bill of Materials 47](#_Toc176202319)

[Appendix 3: Schematic 48](#_Toc176202320)

# Summary

The cleaning and maintenance of lawns, pitches, basketball courts etc are very repetitive tasks which could be automated with the right technology. With an interest in automation and robotics, I am developing an autonomous vehicle platform, which could complete these tasks to a high standard. This will be achieved by using an array of sensors for obstacle detection, environmental monitoring, and navigation. The vehicle will be connected to a webserver, which can used to manually control the vehicle, select the operation mode, and monitor the vehicles status. Telemetry data, such as temperature and humidity can also be viewed on the webpage. This will make it possible to automatically do a wide variety of tasks such as cleaning, cutting grass and ice resurfacing.

The scope of this project involved the integration of hardware and software components to enable the collection and processing of data to autonomously navigate through unknown environments. It also includes the design and implantation of user-friendly interfaces and networking to create a friendly and pleasant user experience.

This project will work towards sustainable development goals 9 and 13. The project would improve the efficiency of maintaining lawns, playing fields, basketball courts etc. It is also fully electric. The project will also reuse parts of previous projects which will further reduce its environmental impact and promote sustainable industrialisation.

1. **Sustainable development goal 13 – take urgent action to combat climate change and its impacts.**
2. **Sustainable development goal 9 – Build resistant infrastructure, promote inclusive and sustainable industrialisation and foster innovation.**

I approached this project by breaking it up into separate sections. I wanted to get the sensors working individually before putting it all together. I also created functions to easily use the sensors and control the motors efficiently. Once all the basic functions of the project where individually implemented, I started to build the webpage.

From this project I learned how to deal with noisy data from sensors. I used a magnetometer to create a compass for navigation, but this proved very difficult as signal is very noisy. By calibrating the compass and averaging the values over time I managed to get reliable readings. I also learned that testing and prototyping is very important and can avoid large problems further down the line.

The main technologies used are C/C++ programming, data analyses and processing, noise reduction/blocking, website design, user interface design, Html, CSS, JavaScript fetch API.

I achieved my goal of creating an autonomous vehicle that can be used as a platform for automating certain tasks, however I ran into many problems along the way and had to come up with creative solutions. To achieve this, I had to change the design of the project in major ways.

# 2. Project Plan

I planned the project during the research phase. The plan was altered over time as the project changed during development. However, I stuck to the original plan as closely as I could.

## Semester 1 Plan

**Week 1 and 2**

* In depth research for the project. Acquire components and start testing.

**Week 2 and 4**

* I will build the ultrasonic location system. I will also research and write the code to find the coordinates of the ultrasonic receivers.

**Week 4 and 5**

* I will test the accuracy of the ultrasonic location and fix any bugs. I will also begin building the vehicle. If possible, I will use the robocar that I built as my first-year project.

**Week 6 and 7**

* I will research and code the path finding. I will also write all the functions I need to control the vehicle.

**Week 8-10**

* I will create a basic implementation of the website. This should include basic features such as manual control, telemetry and have the basic layout of the final webpage. Do a lot of testing and bug fixing during this time to ensure the project is in good shape for the initial project demonstration.

**Weeks 11-12**

* Finalise the project demonstration and start filming for the demonstration video. Create and edit the demonstration video.

## Semester 2 Plan

**Week 1**

* Order new magnetometer and calibrate it. Test and debug the driveTowardsHeading() function.

**Week 2**

* Unifi all the drive functions into a single versatile function.
* Continue debugging the compass.

**Week 3-5**

* add servo for the front ultrasonic sensor. Write to detect obstacles using the ultrasonic sensor.

**Week 5-7**

* Rebuild the website using flexbox, implement fetch API to send commands from the website back to the ESP32.
* Implement the website controls on the ESP32.
* Send telemetry to the website.

**Week 7-9**

* Start work on the project poster.

**Week 9-12**

* Implement obstacle avoidance function.
* Finalise project features.
* Bug testing and tidy up code and website.
* Write project report and film and edit the demonstration video.

## 2.3 Changes to the plan

During development of the project, significant changes were made to the design. As I result the project plan also changed. The removal of the ultrasonic location system was the biggest change. During development of this system, it became apparent that it would be very difficult to implement, and the accuracy was limited to approximately within 5-10 cm during testing.

The alteration to the plan came because of issues with the accuracy and reliability of the magnetometer. I had to order a different sensor and rewrite all the code as result. To avoid this impacting and delaying the rest of the project, I ended putting this part of the project on hold and working on other aspects first.

# Project Architecture

## 3.1Architecture Diagram

A diagram of a computer

Description automatically generated

# Development Platform and Tools

## Development Platform

The ESP32 is a low power microcontroller with built it Wi-Fi and Bluetooth capability. It runs at 240Mhz compared to 16Mhz clock speed of the Arduino Uno. These features make it a perfect choice for an IoT project. It allows a responsive web server to run while running the other code simultaneously. It has 4Mb of flash memory allowing large programs and websites to be stored. [1]

## Development Platform (IDE)

The ESP32 can be programmed using the Arduino ide, making it easy to develop and build projects with. It also supports many of the same libraries as the Arduino. The Arduino IDE has a versatile serial monitor and code debugger to aid development and testing. The Arduino IDE also allows for easy management of libraries and has support for multiple files, making it easy to write and organise custom functions.

## Web Development Platform

To develop the webpage, I used Visual studio code as my code editor. Offers great debugging and supports an extension for a live webpage view, making it easy and quick to make changes the webpage. I also used the Microsoft edge 3D debugger to debug the webpage. This feature makes it easy to debug layering issues Z-index.

Once I finished the webpage, I used the Arduino IDE to store it in literal strings, which can be stored on the ESP32.

## Organising and Planning

I used OneNote to keep a development log of my project. I used Trello to make rough long-term plans. This made it easy to get an overview of the whole project at a glance.

# Sensors

I used 3 ultrasonic sensors to detect obstacles in front and to the side of the vehicle. I used a DHT11 to measure temperature and humidity to make distance measurements more accurate and to display the temperature and humidity on the webpage for the user. I also used a magnetometer to detect which way the vehicle is facing.

The sensors have objects initialised as follows:

A screen shot of a computer code

Description automatically generated

## HSR-04 Ultrasonic Distance Sensors

The HSR-04 provides distance measurement with an accuracy of +-3mm and a field of view of 15 degrees. It operates at 40KHz making it far outside the range of most sounds that could interfere with the measurement. The sensor works by sending out a 40KHz sound in 8 bursts and measuring how long it takes for the signal to return. The output on the Echo pin is in proportion to the time it takes for the signal to return. [2]

To use the ultrasonic sensor a HIGH signal must be applied to the trigger pin for 10us.The Arduin pulseIn() function can then be used to measure the length of the pulse at the Echo pin. The length of the pulse is multiplied by the speed of sound and divided by two to account for the fact the sound has to travel to the object and then back again. The final distance can be calculated using the following formula: distance = (pulseIn × (340M/S) / 2.

pulseIn is a blocking function. This means that while it is waiting for a pulse, it will effectively stop code execution. To optimize my final code, I adjusted the maximum wait time of pulseIn() to 3000us, which is how long it takes for sound to travel 50cm and back. This means that the maximum amount of time spent waiting for all three sensors to measure a distance will be close to 1ms. Since I don’t need the sensors to measure anything beyond 30cm, it will not have any effect on the functionality of my project. To detect if pulseIn timed out or an obstacle is close, I check if pulseIn returned a 0 or a non-zero value.

## Ultrasonic Sensor code

Set pins for ultrasonic sensors

A screen shot of a computer

Description automatically generated

Function to check for obstacles for each ultrasonic sensor. This function is called periodically in all modes. The speed of sound is adjusted by using the temperature reading from the DHT11.

A computer screen shot of a program code

Description automatically generated

This function is used to stop the vehicle if an obstacle is detected in any mode.

A screen shot of a computer code

Description automatically generated

## DHT-11 Temperature Sensor

The DHT-11 is a temperature and humidity sensor with an accuracy of +-2 degrees Celsius and +- 5%RH. The DHT-11 uses a thermistor and a capacitive humidity sensor for measuring. The output is a digital signal using a proprietary communication protocol [3]. It is used in this project to adjust the distance calculations for the ultrasonic sensors as the speed of sound is dependent on the temperature and humidity of the air it is traveling through [4]. The DHT11 can operate at both 3.3 volt and at 5v. I choose to operate it at 5v as the ESP32 input pin work between 0 and 3.3v.

## DHT11 Code

A screen shot of a computer code

Description automatically generated

## QMC5883L Magnetometer

The QMC5883L is a three-axis magnetic sensor. It is a surface mount device that is mounted on a breakout board for easy installation. It is targeted for high precision applications such as virtual reality gaming, drones and robot use. It supports the I2C interface for data transfer. The sensor operates between 2.16V and 3.6V with a low power consumption of 75uA. [5]

In this project it is used to determine the direction the vehicle is heading. This allows me to keep the vehicle driving in a specific direction and to make precise turns, removing the need to precisely calibrate the motor speed to turn a specific direction. The main downside to using a magnetometer to determine the vehicle heading is that it is extremely sensitive to magnetic and electromagnetic fields. To work around this issue, the sensor is mounted on a pole on the back of the vehicle to keep it away from the motors and any other electro magnetic interference. Using magnetic shielding was considered, but it has limited effectiveness and is expensive.

## QMC5883L Code

Initialise I2C interface and setup compass

A screen shot of a computer code

Description automatically generated

Sets the minimum and maximum heading before the vehicle corrects itself. Code also scales values to a 0 to 360 degree rage.

A computer screen shot of a program code

Description automatically generated

Read heading



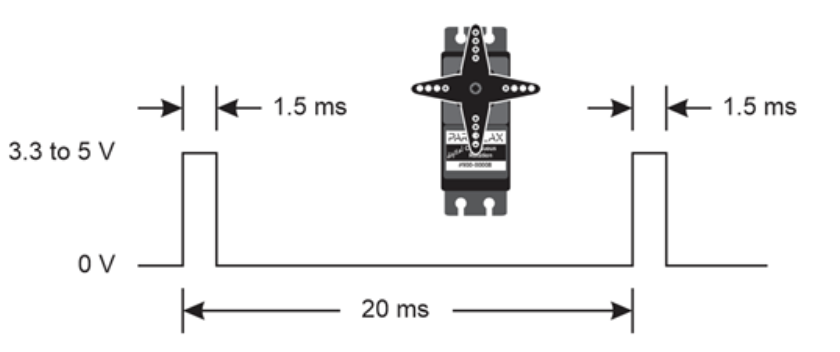
# Motors

I used two 360-degree servo motors as propulsion for my vehicle. Originally, I used two dc motors with a dual H-bridge, but this turned out to be difficult to control precisely. The main issue was that, when running the dc motors at a low duty rate it was very difficult to keep them both moving at the same speed leading to the vehicle veering to one side. The relatively large dc motors also caused large current spikes which could shut of the ESP32 momentarily, causing the program to reset. Using 360 servo motors allows for easy and accurate control of the vehicle while preventing any large current spike due to the small motors and high gear ratio. This allows me to use a smaller battery and simply the code and hardware.

## Servo Motors

The parallax continues rotation servo (#900-00008) is a non-encoded 360-degree servo motor. It can operate between 0 and 50 rpm clockwise and anticlockwise. It operates between 4V and 6V DC. To control the speed of the servo, a PWM signal is applied to the signal pin. A HIGH pulse of 1.3ms turns the servo at full speed clockwise. A High Pulse of 1.7ms turns the servo at full speed counterclockwise. To stop the servo, a 1.5ms HIGH pulse is applied to the signal pin. [6]

To use the parallax servo in my project, I made use of a modified Servo library [7]. This allows me to use the servoWrite() function to control the speed and direction of the servo motor. Since the library is originally designed to be used with regular 180 degree servos, an angle between 0 and 180 degrees is used as the argument is the servoWrite() function. To stop the servo, the argument is 90 degrees. For full speed clockwise rotation, the argument is 0 degrees and for full speed anti-clockwise rotation, the argument is 180 degrees.



## Servo Code

Attach servos to objects

A black rectangular object with white text

Description automatically generated

Keep vehicle pointing in set direction using compass and servos

A computer screen shot of white text

Description automatically generated

# Bluetooth Control

The vehicle can also be manually controlled by a Bluetooth device such as a phone or a laptop.

Bluetooth setup code. Gives the ESP32 a name and connects to device

A close up of a screen

Description automatically generated with medium confidence

Main Bluetooth function to receive data from connected device and control the vehicle accordingly. This function gets called when the vehicle is in Bluetooth mode.

A screen shot of a computer program

Description automatically generated

# Web Server

## Web Server Code

This is the setup code for the webserver. It configures the Wi-Fi network and password. handleKeyPress and handleSlider are used to receive data from the website when a button is pressed or when the slider is used. The switch statement is used to change the vehicle mode when a button is pressed on the website. The code for the website is stored in literal strings which are appended to each other in handle root. Every variable that is displayed on the website need to be appended between two strings with the html code on either side. This is used to display telemetry and to change the colour of the obstacle indicators on the website.

Code for the buttons and slider which send data to the ESP32

A screen shot of a computer

Description automatically generated

A black background with a black border

Description automatically generated with medium confidence

A screen shot of a computer

Description automatically generated

# Problem Solving

While working on the project, I encountered many problems and complications. A few weeks into the project I realized that the scope of the project is too big and that I would not be able to successfully implement every feature at a high quality. After testing and prototyping the ultrasonic location system, it became apparent that it was a very complex system to get right. I decided to focus on other aspects of the project first and later decided to remove the location system all together.

To solve problems in my project, I worked with other students and with lecturers. I also used numerus online resources such as the Arduino forums, datasheets, and guides to help diagnose and fix problems. Below I outlined the major problems I encountered while working on the project.

## Magnetometer Interference

The QMC5883 was giving very unreliable readings when using the example provided by the library I was using. I found out that the specific gy271 breakout board that I sourced was using the cheaper QMC5883L sensor instead of the HMC5883L sensor. I sourced a library designed for this sensor and found it more accurate but still unreliable.

After some experimentation I found that placing the sensor on a pillar on the back of the vehicle away from the magnets in the motor fixed this issue and I got reliable and accurate readings from the compass.

I considered using magnetic shielding foil, but shielding is expensive and less reliable than just placing the sensor away from any interference. The sensor can’t be entirely shielded either because it needs to be able to read earth’s magnetic field to be used as a compass.

## Ultrasonic Location System

Splitting the ultrasonic sensors into transmitter and receiver pairs was effective when the sensors were pointed directly are each other. When the sensors were pointed more than 5 or 10 degrees away from each other, they gave inaccurate readings by about +- 2cm per degree. I modified one of the transmitters and receivers to have a wider field of view. This resulted in slightly more accurate readings but still not good enough for my application. Since I was unable to find a solution that was within the scope of this project, I had to cut the location and path finding aspect from my project. Below are two images of the modified sensors.



## Vehicle Veering to one side

The vehicle was always veering to the right and then snapped back to the original heading. Initially I thought this was an issue with the compass drifting, or interference. After investigating the problem further, I realised that the left DC motor was slightly slower than the right motor causing the vehicle to always veer slightly to the right. Initially to solve this problem, I ran the left motor with a slightly higher duty cycle by about 10%. This improved the problem, but it still wasn’t perfect, and it wasn’t a clean solution and made the code more complicated and messier.

I decided to use continues rotation servos instead of the DC motors for my final hardware. Servo motors are extremely precise and easy to code. Servo motors are also easy to control at low speeds while DC motors are very unreliable at a duty cycle of less than 30%, which is still faster than I want the project to drive at.

Bottom view


Figure 1 Project with DC motors and H-Bridge

A small vehicle with wheels and wires

Description automatically generated

Figure 2 Project with Servo Motors

## ESP32 not connecting to PC

Near the end of the project, the ESP32 stopped connecting to the pc or laptop. I tried using multiple USB cables, updating the Arduino IDE and using different computer. I then tried the legacy version of the Arduino IDE, which gave me an error code relating to the bios mode when trying to upload my code. I locked up the error and found a forum post with a solution. Holding the boot button on the ESP32 while uploading code solved the issue.

# Impact of Project on Sustainability

I considered the impact of my project on the united nationals sustainability (SDGs)goals. I also considered the impact my project would have sustainability during the conception and design of my project to ensure to ensure that my project works towards the SDGs as much as possible.

I concluded that my project directly works towards two of the SDGs

1. **Sustainable development goal 13 – take urgent action to combat climate change and its Sustainable development goal 13 – take urgent action to combat climate change and its impacts.**

I re-used components from previous projects to build my autonomous vehicle. This reduces toxic e-waste and reduces the carbon footprint of my project. My project is also fully electric, leading to close to zero emissions if the electricity is generated using renewable sources.

1. **Sustainable development goal 9 – Build resistant infrastructure, promote inclusive and sustainable industrialisation and foster innovation.**

My project is made mostly from re-used components leading to sustainable industrialisation.

# Conclusion

In conclusion, I learned a lot from designing and building this project. I think that overall it went well.

The outcome of this project is:

A website with live telemetry and controls, which is designed to be accessible.

An autonomous cleaning robot with ultrasonic sensors for obstacle detection and temperature and humidity sensors for environmental monitoring.

Manual Bluetooth control of the vehicle using a switch on the vehicle.

While I had to make some modifications to the project due to time constraints, I think my project still achieves the functionality outlined in the project proposal but using a different method. In the future, I would like to further develop the project to be capable of line paint and following specific pre-programmed paths using a local location system.

# References

|  |  |
| --- | --- |
| [1] | DOIT, “Esp32 Datasheet,” [Online]. Available: https://roboeq.ir/files/id/4034/name/ESP32%20MODULE.pdf/. [Accessed 30 08 2024]. |
| [2] | SparkFun, “HCSSR04 Datasheet,” sparkfun, [Online]. Available: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf. [Accessed 08 01 2024]. |
| [3] | MOUSER, “DHT11 datasheet,” [Online]. Available: https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf. [Accessed 08 01 2024]. |
| [4] | W. &. G. M. &. W. T. &. v. d. G. C. van Schaik, “High Accuracy Acoustic Relative Humidity Measurement in Duct Flow with Air. Sensors,” 08 2010. [Online]. Available: https://www.researchgate.net/figure/Speed-of-sound-vs-temperature-and-relative-humidity-according-to-2-p-1013-kPa-314\_fig1\_51873087. [Accessed 10 01 2024]. |
| [5] | “3-Axis Magnetic senosr QMC5883L datasheet,” [Online]. Available: https://github.com/e-Gizmo/QMC5883L-GY-271-Compass-module/blob/master/QMC5883L%20Datasheet%201.0%20.pdf. [Accessed 15 01 2024]. |
| [6] | Parrallax, “900-00008-Continuous-Rotation-Servo-Documentation-v2.2.pdf,” Parrallax, [Online]. Available: https://www.parallax.com/package/parallax-continuous-rotation-servo-downloads/. [Accessed 10 01 2024]. |
| [7] | madhephaestus, “ESP32 servo library,” [Online]. Available: https://github.com/madhephaestus/ESP32Servo. |

# Appendix 1: Code

Github link: https://github.com/Pero0123/IOT-Project

#include <WiFi.h>

#include <WiFiClient.h>

#include <WebServer.h>

#include <ESPmDNS.h>

#include "homepage.h"

#include <Servo.h>

#include <BluetoothSerial.h>

#include <Wire.h>

#include <QMC5883L.h>

#include <DFRobot\_DHT11.h>

//dht variables and pins

#define DHT11\_PIN 15

int temperature;

int humidity;

//switch pin

#define sw 34

int vehicleHeading = 140;  //stores vehicle heading from compass//

int targetHeading = 140;   //the heading the vehicle should drive towards

int maxHeading, minHeading;

int operationMode = 4;  //0:Manual, 1:Autonomous, 2:Drive to Heading

unsigned long startMillis;

unsigned long currentMillis;

const unsigned long period = 50;  //the value is a number of milliseconds, ie 1 second

/////Ultrasonic pins and variables/////

//right sensor

#define Trig1 12

#define Echo1 25

//front sensor

#define Trig2 13

#define Echo2 26

//left sensor

#define Trig3 14

#define Echo3 27

// ultrasonic distance variable

float frontDistance;

float rightDistance;

float leftDistance;

bool obstacleFront = false;

bool obstacleLeft = false;

bool obstacleRight = false;

#define collisionDistance 25

#define headingRange 14

//website collision indicators//

String CollisionLeft = "Green";

String CollisionCenter = "Red";

String CollisionRight = "Green";

String vehicleMode = "Select Mode";

//bluetooth stuff here

char BTmessage;

//Sensor objects

BluetoothSerial SerialBT;

QMC5883L compass;

DFRobot\_DHT11 DHT;

Servo servoRight;

Servo servoLeft;

const char\* ssid = "Pero";

const char\* password = "pero1234";

WebServer server(80);

void handleKeyPress() {

  String receivedData = server.arg("button");

  operationMode = receivedData.toInt();

  switch (operationMode) {

    case 0:

      vehicleMode = "Bluetooth";

      break;

    case 1:

      vehicleMode = "Autonomous";

      break;

    case 2:

      vehicleMode = "Drive Toward Heading";

      break;

    default:

      vehicleMode = "Select Mode";

      break;

  }

  //Serial.println(receivedData);

  server.send(200);

}

void handleHeading() {

  String receivedHeading = server.arg("heading");

  if (operationMode == 2) {

    targetHeading = receivedHeading.toInt();

    configureVehicleHeading();

  }

  server.send(200);

}

void handleRoot() {

  String message =

    homePagePart1 + vehicleHeading + homePagePart2 + temperature + homePagePart3 + humidity + homePagePart4 + vehicleMode + homePagePart5 + CollisionLeft + homePagePart6 + CollisionCenter + homePagePart7 + CollisionRight + homePagePart8;

  server.send(200, "text/html", message);

}

void handleNotFound() {

  String message = "File Not Found\n\n";

  message += "URI: ";

  message += server.uri();

  message += "\nMethod: ";

  message += (server.method() == HTTP\_GET) ? "GET" : "POST";

  message += "\nArguments: ";

  message += server.args();

  message += "\n";

  for (uint8\_t i = 0; i < server.args(); i++) {

    message += " " + server.argName(i) + ": " + server.arg(i) + "\n";

  }

  server.send(404, "text/plain", message);

}

void setup(void) {

  Serial.begin(115200);

  WiFi.mode(WIFI\_STA);

  WiFi.begin(ssid, password);

  Serial.println("");

  // Wait for connection

  while (WiFi.status() != WL\_CONNECTED) {

    delay(500);

    Serial.print(".");

  }

  Serial.println("");

  Serial.print("Connected to ");

  Serial.println(ssid);

  Serial.print("IP address: ");

  Serial.println(WiFi.localIP());

  if (MDNS.begin("esp32")) {

    Serial.println("MDNS responder started");

  }

  server.on("/", handleRoot);

  server.on("/sentKeyPressToWebServer", handleKeyPress);

  server.on("/sentHeadingToWebServer", handleHeading);

  server.on("/inline", []() {

    server.send(200, "text/plain", "this works as well");

  });

  server.onNotFound(handleNotFound);

  server.begin();

  Serial.println("HTTP server started");

  bluetoothSetup();

  //Compass setup Code//

  Wire.begin();

  compass.init();

  compass.setSamplingRate(200);

  Serial.println("QMC5883L Compass Demo");

  Serial.println("Turn compass in all directions to calibrate....");

  //Servo Setup Code//

  servoLeft.attach(2);

  servoRight.attach(4);

  startMillis = millis();  //initial start time for millis timing

  //Configure the pins for ultrasonic sensors

  pinMode(Echo1, INPUT);

  pinMode(Echo2, INPUT);

  pinMode(Echo3, INPUT);

  pinMode(Trig1, OUTPUT);

  pinMode(Trig2, OUTPUT);

  pinMode(Trig3, OUTPUT);

  //switch setup//

  pinMode(sw, INPUT);

  delay(3000);  //delay to allow user to switch on vheicle safely

//stop the servos

  servoRight.write(90);

  servoLeft.write(90);

  Serial.print("Pleas connect to the webserver at ");

  Serial.print(WiFi.localIP());

  Serial.print(" to access the Dashboard");

  Serial.println("");

  delay(100);

}

void loop(void) {

  currentMillis = millis();  //current time passed since start in milliseconds

  if (currentMillis - startMillis >= period) {  //check if the period has elapsed

  server.handleClient();

  startMillis = currentMillis;

  delay(1);  //allow the cpu to switch to other tasks

  }

  checkForObstacles();

  vehicleHeading = compass.readHeading();

  //check if bluetooth switch is on. switch to bluetooth mode if it is

  if (digitalRead(sw) == HIGH) {

    operationMode = 0;

  }

    //check which mode the vehicle should run in

    switch (operationMode) {

      case 0:  //Bluetooth Mode

        Serial.println("Bluetooth Mode");

        stopForObstacle();

        configureVehicleHeading();

        bluetoothControl();

        break;

      case 1:                      //Autonomous Mode

        Serial.println("Autonomous mode");

        Serial.println(vehicleHeading);

        driveTowardHeading();

          if (obstacleFront || obstacleRight || obstacleLeft) {

          servoRight.write(0);

          servoLeft.write(180);

          delay(1500);

          servoRight.write(90);

          servoLeft.write(90);

          //pick a new direction

          int newHeading;

          if (obstacleLeft == false) {

            newHeading = targetHeading - 100;

          }

          else if (obstacleRight == false) {

            newHeading = targetHeading + 100;

          }

          else {

            newHeading = targetHeading + 180;

          }

          //correct heading to be between 0 and 360

          if (newHeading > 360) {

            newHeading = newHeading - 360;

          }

          else if (newHeading < 0) {

            newHeading = newHeading + 360;

          }

          targetHeading = newHeading;

        }

        break;

      case 2:  //Drive towards heading Mode

        driveTowardHeading();

        stopForObstacle();

        break;

      default:

        server.handleClient();

        Serial.print("Pleas connect to the webserver at ");

        Serial.print(WiFi.localIP());

        Serial.print(" to access the Dashboard");

        Serial.println("");

        delay(100);

        servoRight.write(90);

        servoLeft.write(90);

        Serial.println("defualt mode");

        break;

    }

    //update temperature and humidity

    DHT.read(DHT11\_PIN);

    temperature = DHT.temperature;

    humidity = DHT.humidity;

//    Serial.println("mode =");

//    Serial.print(operationMode);

//    Serial.println("targetHeading =");

//    Serial.print(targetHeading);

//    Serial.println("vehicleHeading =");

//    Serial.print(vehicleHeading);

    delay(5);

}

void bluetoothSetup() {

  String device\_name = "AutonomousVehicle";  //bluetooth name of the esp32

  //defintions for bluetooth setup

#if !defined(CONFIG\_BT\_ENABLED) || !defined(CONFIG\_BLUEDROID\_ENABLED)

#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it

#endif

#if !defined(CONFIG\_BT\_SPP\_ENABLED)

#error Serial Bluetooth not available or not enabled. It is only available for the ESP32 chip.

#endif

  //bluetooth setup code

  SerialBT.begin(device\_name);  //Bluetooth device name

  Serial.printf("The device with name \"%s\" is started.\nNow you can pair it with Bluetooth!\n", device\_name.c\_str());

  //Serial.printf("The device with name \"%s\" and MAC address %s is started.\nNow you can pair it with Bluetooth!\n", device\_name.c\_str(), SerialBT.getMacString()); // Use this after the MAC method is implemented

#ifdef USE\_PIN

  SerialBT.setPin(pin);

  Serial.println("Using PIN");

#endif

}

void bluetoothControl() {

  char message;

  //reads and sends information between bluetooth app and serial console

  if (SerialBT.available()) {

    message = SerialBT.read();

  }

  switch (message) {

    case 'w':

    servoRight.write(180);

    servoLeft.write(0);  //go forwards

      break;

    case 's':

    servoRight.write(0);

    servoLeft.write(180);  //reverse obvisoulsy

      break;

    case 'a':

    servoRight.write(45);

    servoLeft.write(45);  //turn left

      break;

    case 'd':

    servoRight.write(135);

    servoLeft.write(135);  //turn right

      break;

    case 'g':

    servoRight.write(90);

    servoLeft.write(90);  //stop.

      break;

  }

}

void configureVehicleHeading() {

  //configurs the max and min headings before the vehicle corrects itself

  //ie. if target = 355 and range is +- 6. max will be 355+6= 361 which is actually 1 degrees.

  if ((targetHeading - headingRange) < 0) {

    maxHeading = (targetHeading - headingRange) + 360;

    minHeading = targetHeading + headingRange;

  } else if ((targetHeading + headingRange) > 360) {

    minHeading = (targetHeading + headingRange) - 360;

    maxHeading = targetHeading - headingRange;

  } else {

    minHeading = targetHeading - headingRange;

    maxHeading = targetHeading + headingRange;

  }

}

void driveTowardHeading() {

  configureVehicleHeading();

  if (vehicleHeading < maxHeading && vehicleHeading > minHeading) {

    servoRight.write(180);

    servoLeft.write(0);

  } else if (vehicleHeading > maxHeading) {

    servoRight.write(87);

    servoLeft.write(87);  //facing to the right. steer left

  } else if (vehicleHeading < minHeading) {

    servoRight.write(93);

    servoLeft.write(93);  //facing to the left. steer right

  }

}

void checkForObstacles() {

  float duration;

  float distance;

  float speedOfSound;

  speedOfSound = 0.0331 + ( 0.61\*temperature );

  //check for collision on the Right

  digitalWrite(Trig1, LOW);

  delayMicroseconds(2);

  digitalWrite(Trig1, HIGH);

  delayMicroseconds(10);

  digitalWrite(Trig1, LOW);

  duration = pulseIn(Echo1, HIGH, 3000);

  rightDistance = (duration \* speedOfSound / 2);

  if (rightDistance <= collisionDistance && duration != 0) {

    obstacleRight = true;

    CollisionRight = "Red";

  } else {

    obstacleRight = false;

    CollisionRight = "Green";

  }

  delayMicroseconds(100);

  //check for collision on the front

  digitalWrite(Trig2, LOW);

  delayMicroseconds(2);

  digitalWrite(Trig2, HIGH);

  delayMicroseconds(10);

  digitalWrite(Trig2, LOW);

  duration = pulseIn(Echo2, HIGH, 3000);

  frontDistance = (duration \* speedOfSound / 2);

  if (frontDistance <= collisionDistance && duration != 0) {

    obstacleFront = true;

    CollisionCenter = "Red";

  } else {

    obstacleFront = false;

    CollisionCenter = "Green";

  }

  delayMicroseconds(100);

  //check for collision on the Left

  digitalWrite(Trig3, LOW);

  delayMicroseconds(2);

  digitalWrite(Trig3, HIGH);

  delayMicroseconds(10);

  digitalWrite(Trig3, LOW);

  duration = pulseIn(Echo3, HIGH, 3000);

  leftDistance = (duration \* speedOfSound / 2);

  if (leftDistance <= collisionDistance & duration != 0) {

    obstacleLeft = true;

    CollisionLeft = "Red";

  } else {

    obstacleLeft = false;

    CollisionLeft = "Green";

  }

}

void stopForObstacle()

{

  if (obstacleFront || obstacleRight || obstacleLeft) {

    servoRight.write(90);

    servoLeft.write(90);

  }

}

// To create literal strings stored in flash memory enclose your HTML code between

// F(R"=====( HTML code here )=====");

// If you have 1 reading then you probably have 2 literal strings

// If you have 2 readings then you probably have 3 literal strings etc.

String homePagePart1 = F(R"=====(

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta http-equiv="refresh" content="3">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Autonomous Vehicle</title>

</head>

<body>

  <style>

\*{

    background-color: rgb(4, 75, 138);

    font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif;

    text-align: center;

    margin: 5px

}

p{

  background-color: transparent;

  font-size: 20px;

}

th{

  text-align: left;

  font-size: 20px;

}

.flex-container {

  display: flex;

  flex-direction: row;

  justify-content: space-around;

}

h1{

  height: 15%;

  width: 100%;

  color: rgb(255, 255, 255);

  font-size: 60px;

}

h2{

  width: 100%;

  color: rgb(255, 145, 0);

  font-size: 25px;

  padding-bottom: 60px;

}

#Dashboard{

  display: flex;

  flex-direction: row;

  justify-content: space-around;

}

#Telemetry{

 width: 20%;

}

#CarView{

  display: flex;

  flex-direction: column;

  justify-content: center;

  width: 20%;

}

#Controls{

  width: 20%;

}

.slidecontainer {

  width: 80%;

  justify-content: center;

}

.slider {

  -webkit-appearance: none;

  width: 100%;

  height: 25px;

  border-radius: 20px;

  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: 25px;

  height: 25px;

  background: orange;

  cursor: pointer;

}

.slider::-moz-range-thumb {

  width: 25px;

  height: 25px;

  background: orange;

  cursor: pointer;

}

.collision\_object{

  width: 20%;

  padding: 2%;

  min-height: 10px;

  text-align: center;

  border-radius: 20px;

}

.footer {

  display: flex;

  flex-direction: row;

  justify-content: left;

  margin-top: 20px;

  margin-left: 7%;

}

.pageLink{

  background-color: orange;

  border-radius: 20px;

}

@media screen and (max-width: 700px) {

  #Dashboard {

    flex-direction: column;

  }

  h1{

    font-size: 30px;

  }

  h2{

    font-size: 15px;

  }

  th{

  font-size: 10px;

}

p{

  font-size: 10px;

}

}

@media screen and (max-width: 1000px) {

  h1{

    font-size: 30px;

  }

  h2{

    font-size: 15px;

  }

  th{

  font-size: 10px;

}

p{

  font-size: 10px;

}

}

  </style>

<!-- Title -->

<div class = "flex-container" style="flex-direction: column;">

<h1>Autonomous Vehicle</h1>

<h2>Welcome to the Autonomous Vehicle Dashboard</h2>

</div>

<!-- Main content -->

<div id = "Dashboard">

  <div id="Telemetry" style="text-align: left;">

    <table>

      <tr>

        <th>Heading:</th><th>

)=====");

String homePagePart2 = F(R"=====(

</th>

      </tr>

      <tr>

        <th>Tempreture:</th><th>

)=====");

String homePagePart3 = F(R"=====(

  Celsius</th>

      </tr>

      <tr>

        <th>Humidity:</th><th>

)=====");

String homePagePart4 = F(R"=====(

%</th>

      </tr>

      <tr>

        <th>Mode:</th><th>

)=====");

String homePagePart5 = F(R"=====(

</th>

      </tr>

    </table>

  </div>

  <div id="CarView">

    <div class="flex-container">

    <div class="collision\_object" style="background-color:

)=====");

String homePagePart6 = F(R"=====(

     ;"></div>

    <div class="collision\_object" style="background-color:

)=====");

String homePagePart7 = F(R"=====(

;"></div>

    <div class="collision\_object" style="background-color:

)=====");

String homePagePart8 = F(R"=====(

;"></div>

  </div>

    <img src="https://freesvg.org/img/car\_topview.png" alt="Vehicle">

  </div>

  <div class = "flex-container" id="Controls">

    <div class="slidecontainer">

      <h2 style="padding-bottom: 5px;">Set Heading</h2>

      <input type="range" min="1" max="360" value="90" class="slider" id="myRange" oninput="fetch('sentHeadingToWebServer?button=T')">

      <h2><span id="demo"></span></h2>

      <div class="flex-container">

        <button class="button" onclick="fetch('/sentKeyPressToWebServer?button=0')">Bluetooth</button>

        <button class="button" onclick="fetch('/sentKeyPressToWebServer?button=1')">Autonomous</button>

        <button class="button" onclick="fetch('/sentKeyPressToWebServer?button=2')">Heading</button>

      </div>

    </div>

  </div>

</div>

<script>

  var slider = document.getElementById("myRange");

  var output = document.getElementById("demo");

  output.innerHTML = slider.value;

  slider.oninput = function() {

    var heading = slider.value;

    output.innerHTML = this.value;

    fetch(`sentHeadingToWebServer?heading=${heading}`);

  }

  </script>

</body>

</html>

)=====");

# Appendix 2: Bill of Materials

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Component Name | Quantity | Manufacture | ATU Stores | Sourced From | Order No | Cost Euros |
| ESP32 | 1 | DOIT | y | ATU stores |  |  |
| GY-271\* | 1 | osoyoo | n | Amazon |  | 5.60 |
| 10k\* | 1 |  | y |  |  |  |
| 3K3 \* | 3 |  | y |  |  |  |
| 1K8\* | 3 |  | y |  |  |  |
| HC-SR04\* | 3 | Kuongshun Electronic | n | Amazon |  | 9.80 |
| Battery\* | 1 | Turnigy | n | RC ireland |  | 15.99 |
| Chassis | 1 |  | y |  |  |  |
| Parallax Servo | 2 | Parallax | y |  |  |  |
| DHT-11 | 1 | DF Robot | y | Radionics |  | 5.45 |

# Appendix 3: Schematic

A paper with a diagram

Description automatically generated