Wroclaw University Of Science And Technology

BATIN TOPBAŞOĞLU 276718

CANER OLCAY 276715

PIOTR KĘDZIA 275540

ADVANCED TOPICS IN ROBOTICS

MILESTONE 1

Task 2: Making the necessary arrangements in the general plan,determining the sensors and devices to be used,presenting the final version of the project plan.

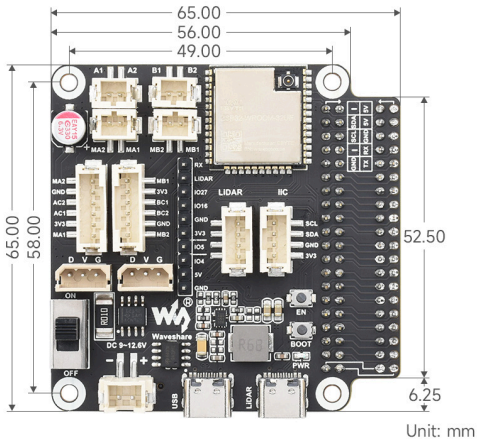
We are going to use an ultrasonic sensor [HC-SR04]. Its purpose is to measure distance to obstacles in the way of driving our robot. The working principle of this sensor is by sending sound waves from the transmitter, which then bounce off of an object and then return to the receiver. You can determine how far away something is by the time it takes for the sound waves to get back to the sensor.

FIGURE 1: HC-SR04 Ultrasonic sensor

Robot's main challenge is driving according to our commands given via a basic phone app. To do so the tracks require some force that will push them forward. In our project, the responsibility for this task falls on two robot chassis motor [33GB-52–18.7]

FIGURE 2: 33GB-520-18.7 robot chassis motor

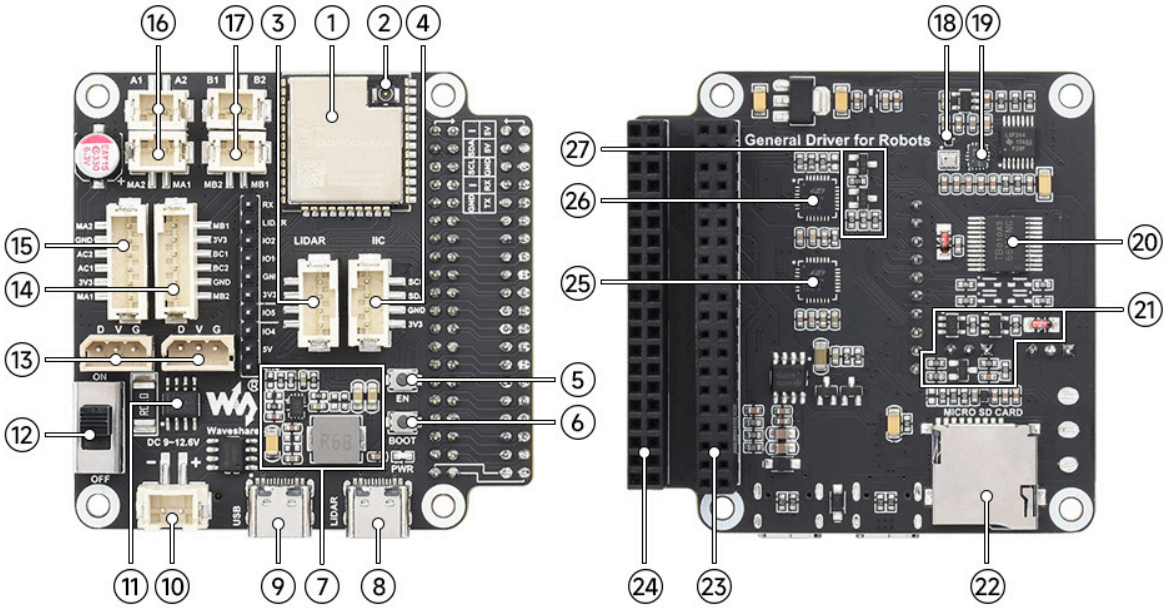
FIGURE 3: ADDITIONAL INFO FROM DATASHEET



If we compare robot motors to a heart pumping blood, the next part would be similar to the human brain. We will use a General Driver For Robots, where all magics occur. This component will be responsible for passing orders and working out all I/O signals. This multifunctional driver board is specially designed for driving robots. Based on the ESP32-WROOM-32 module, Arduino IDE can be used for development, and it supports a variety of wireless communication methods, including WIFI, Bluetooth, and ESP-NOW.

FIGURE 4: Front view of General Driver For Robots

What's On Board



Amid all of possibility this driver board gives up, we will mainly focus on :

- PORT 10 through which we will supply voltage from our battery pack
- PORT 16 through which the microcontroller will supply robot motors

Task 3: Write the first software steps to control platform movement (e.g., speed, direction) using a python programming language. Develop functions for sensor input and decision-making algorithms.

```
#define MOTOR_A_PIN1 16
#define MOTOR_A_PIN2 17
#define MOTOR_B_PIN1 18
#define MOTOR_B_PIN2 19

void setup() {

    Serial.begin(115200);
    Serial.println("Tracked Robot Ready! Use 'w', 's', 'a', 'd', 'x' for control.");


    pinMode(MOTOR_A_PIN1, OUTPUT);
    pinMode(MOTOR_A_PIN2, OUTPUT);
    pinMode(MOTOR_B_PIN1, OUTPUT);
    pinMode(MOTOR_B_PIN2, OUTPUT);


    stopMotors();
}
```

The first part of the software,we defined motor pins since we need to connect our motors to the pins A1,A2 and B1,B2 we want to name it that way.

In the middle `setup()` function initializes the robot by setting up the serial communication at 115200 (it's needed baud for esp32 wroom 32UE) baud and printing an initial message.

It also configures the motor control pins as outputs and stops the motors by calling the `stopMotors()` function.

After all, the control message prompts the user to use the 'w', 's', 'a', 'd', and 'x' keys for movement. We want to make sure that our robot is suitable for our device and before go any further we thought that it will be good idea to check everything

```
void stopMotors() {
    digitalWrite(MOTOR_A_PIN1, LOW);
    digitalWrite(MOTOR_A_PIN2, LOW);
    digitalWrite(MOTOR_B_PIN1, LOW);
    digitalWrite(MOTOR_B_PIN2, LOW);
    Serial.println("Motors stopped.");
}

void moveForward() {
    digitalWrite(MOTOR_A_PIN1, HIGH);
    digitalWrite(MOTOR_A_PIN2, LOW);
    digitalWrite(MOTOR_B_PIN1, HIGH);
    digitalWrite(MOTOR_B_PIN2, LOW);
    Serial.println("Moving forward.");
}

void moveBackward() {
    digitalWrite(MOTOR_A_PIN1, LOW);
    digitalWrite(MOTOR_A_PIN2, HIGH);
    digitalWrite(MOTOR_B_PIN1, LOW);
    digitalWrite(MOTOR_B_PIN2, HIGH);
    Serial.println("Moving backward.");
}

void turnLeft() {
    digitalWrite(MOTOR_A_PIN1, LOW);
    digitalWrite(MOTOR_A_PIN2, HIGH);
    digitalWrite(MOTOR_B_PIN1, HIGH);
    digitalWrite(MOTOR_B_PIN2, LOW);
    Serial.println("Turning left.");
}
```

This part we defined functions to control the movement of a tracked robot:

stopMotors(): Stops both motors by setting all motor pins to LOW.

moveForward(): Moves the robot forward by activating the both motors.

moveBackward(): Moves the robot backward by reversing the motor pin states.

turnLeft(): Turns the robot left by running one motor forward and the other backward.

turnRight(): Turns the robot right by running one motor forward and the other backward.

We used HIGH and LOW refer to the voltage levels sent to the motor control pins;

High means the motor takes the power and moves.

Low means no voltage applied to the selected motor.

```
void loop() {

    if (Serial.available() > 0) {
        char command = Serial.read();
        Serial.print("Received command: ");
        Serial.println(command);

        switch (command) {
            case 'w':
                Serial.println("Command recognized: 'w' - Move Forward");
                Serial.println("Executing: moveForward()");
                moveForward();
                break;
            case 's':
                Serial.println("Command recognized: 's' - Move Backward");
                Serial.println("Executing: moveBackward()");
                moveBackward();
                break;
            case 'a':
                Serial.println("Command recognized: 'a' - Turn Left");
                Serial.println("Executing: turnLeft()");
                turnLeft();
                break;
            case 'd':
                Serial.println("Command recognized: 'd' - Turn Right");
                Serial.println("Executing: turnRight()");
                turnRight();
                break;
            case 'x':
                Serial.println("Command recognized: 'x' - Stop Motors");
                Serial.println("Executing: stopMotors()");
                stopMotors();
                break;
            default:
                Serial.println("Invalid command. Use 'w', 's', 'a', 'd', 'x'.");
                break;
```

For the last part `loop()` function listens for user input through the serial interface and executes commands based on the received character.

`Serial.available() > 0`: Checks if there is data available to read from the serial buffer.

`char command = Serial.read();`: Basically reads the character

Switch-Case Structure: Based on the character received, it shows the output line on the screen.

'w': Call `moveForward()` to move the robot forward.

's': Call `moveBackward()` to move the robot backward.

'a': Call `turnLeft()` to turn the robot left.

'd': Call `turnRight()` to turn the robot right (although you would need to define `turnRight()`).

'x': Call `stopMotors()` to stop the robot.

When the program does not scan any input or doesn't match any of the specified characters, it prints an "Invalid command" message.

Sources:
https://www.waveshare.com/general-driver-for-robots.htm
https://componentslibrary.com/33gb-520-18-7-motor-datasheet.html