Voice Controlled Robot Documentation

Robotics and Mechatronics (EEE469 – Section 1)

Spring 2025

Abyaz Karim – ID: 2110366
A F M Afnan Uzzaman Sheikh – ID: 2130635

Course Instructor: Dr. Kh Shahriya Zaman

Date of Submission: 03 May 2025

1 Introduction

The project focuses on the construction of a voice-controlled robot under limited budget. A large portion of robotics projects utilise built-in voice recognition which is then translated to logic within the machine's microcontroller and used to actuate movement according to command. Due to a limited budget and time, the robot here utilizes a less powerful microcontroller. However, is able to produce simple movement instructions according to commands given. For example, saying "forward, backward, spin" can then let the robot follow these instructions in sequence. Though the project was implemented in a basic form, more sophisticated voice controlled robots can be used in automation mainly. For example, household tasks or chores, otherwise cleaning or serving tasks within a restaurant. For this project, the assigned design requirements were:

1. Design a mobile autonomous robot (take instruction from voice, no remote controller)
2. Implement voice recognition algorithm/services
3. Implement some predefined instructions for custom actions.

The voice-controlled robot made has achieved the above requirements during demonstration.

2 Design Overview

For the control unit, the system utilizes an ESP8266 microcontroller (MCU) on a NodeMCU development board. This choice was made due to the ESP8266's inbuilt Wi-Fi module, which allows seamless internet connectivity. This feature is crucial for the project, as we rely on an external laptop to handle voice processing and send control instructions via webhooks. The webhooks are managed using Blynk, a low-cost Internet of Things (IoT) cloud platform that simplifies remote device control.

On the hardware side, an L298N motor driver was selected to ensure the motors receive sufficient current (0.5A per motor) for optimal operation. The motors that are being drived are DC motors that are geared. The gearbox ensures the motor outputs a high enough torque.

Blynk's virtual pins feature plays a central role in this system. Once the ESP8266 connects to the Blynk cloud, it can monitor virtual pins for triggers. These pins can be activated manually and via web requests, enabling the system to execute specific commands remotely.

Furthermore, the voice control is facilitated by Python, using the Google Speech Recognition API. The computer's built-in microphone records the user's speech, which is then processed and converted into a string. Using logic operations, we can associate voice commands, such as "forward," with a specific web requests. These requests trigger corresponding virtual pins on the Blynk platform.

When a virtual pin is triggered, the MCU receives the signal and performs the necessary action. For instance, if the "forward" command is recognized, the ESP8266 sends a signal

to the motor driver, which controls the motors to rotate clockwise or counterclockwise based on the direction indicated by the command. This ensures that the car responds accordingly to the user's voice commands.

3 Bill of Materials (BOM)

Material	**Price / BDT**
L298N H-Bridge DualMotor Driver, Stepper Motor Driver	183
Kit consisting of: Chassis 2 wheels 2 gear motors Battery holder Swivel type castor wheel	650
ESP8266 CH340 NodeMCUWifi Module Lua V3	390
MAX7219 Dot LedMatrix LED Display Module	250
Total	1473

The above materials and the prices show that the project was affordable given that the total is under 1500 BDT (12.36 USD or 10.88 EUR). The NodeMCU ESP8266 is also a viable and cheap option that allows for WiFi capabilities compared to alternatives such as the Arduino UNO R3 with a Bluetooth module or a Raspberry Pi Pico. It should be mentioned that the voice input to the robot is supplied with through any machine that can locally run the Python script made for this project.

4 Assembly Instructions

The assembly of the robot is fairly simple. The aim is to keep the power supply and microcontroller on top and the motor driver and motors on the bottom. The connections can be made through a breadboard, Veroboard or PCB as shown in the circuit connection diagram. It is advisable to use header pins to connect the ESP8266 microcontroller to the rest of the circuit with female pin headers (2.54mm) when using a Veroboard or PCB. After the circuit has been built the power supply is connected to the circuit and a switch is kept between the positive terminal and the positive supply to the circuit to allow the circuit to be easily turned on and off.

The motor driver attached to the underside of the chassis however can be placed on top as well. As was the case for this project, the motor driver was attached to the underside of the chassis using hot glue and electrical tape. The outputs from the motor driver are then connected to the motors to allow necessary actuation.

After all the connections have been made, the code for the ESP8266 is required to be uploaded into the microcontroller. The necessary modifications in brief are the Blynk device

authorization token and the WiFi network name and password. After the modifications are made, the code can be verified and uploaded. The authorization token is generated from the Blynk device setup, which then allows for the creation of "virtual pins". Each virtual pin corresponds to a different action and is denoted in the ESP8266 code provided. The setup of the Blynk device and virtual pins are described more in detail in the software and programming section.

The final part of the process involves the Python script provided. This script needs to be modified according to the Blynk device created. This process is simple because the script operates through webhooks which activate the virtual pins according to a voice command. It is suggested that the script is run locally on a personal computer and the necessary packages are installed (elaborated in the Software and Programming section). It is also recommended that the Spyder IDE be used to run the script to observe live feedback of the script's operation. The script is then run and voice commands can then operate the robot.

The assembly steps can be summarized as below:

1. Connect circuits as shown in diagram.
2. Connect circuits to power and the motors on the chassis.
3. Edit in personal Blynk authorization token and Wifi network ID and password. Then verify and upload code to the ESP8266 board.
4. Setup the Blynk device board and virtual pins according to the instructions in the ESP8266 code.
5. Open and run Python webhook script provided, use voice commands into microphone to move the robot.

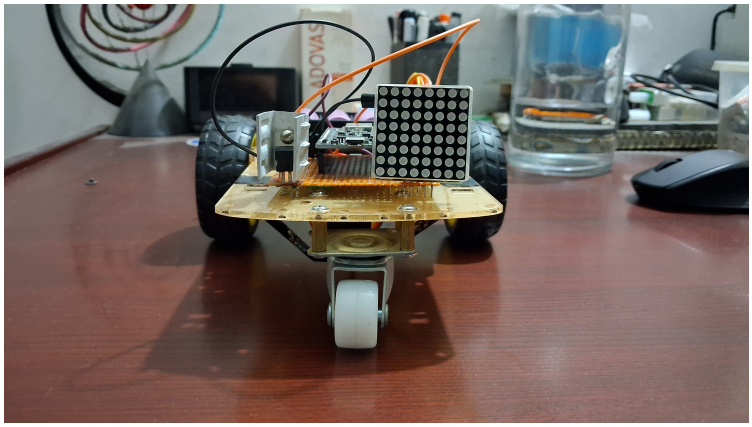

Figure 1: Robot front view

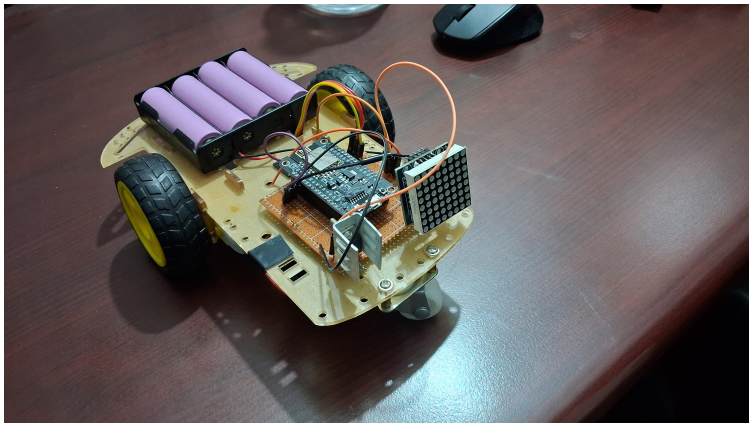

Figure 2: Robot isometric view

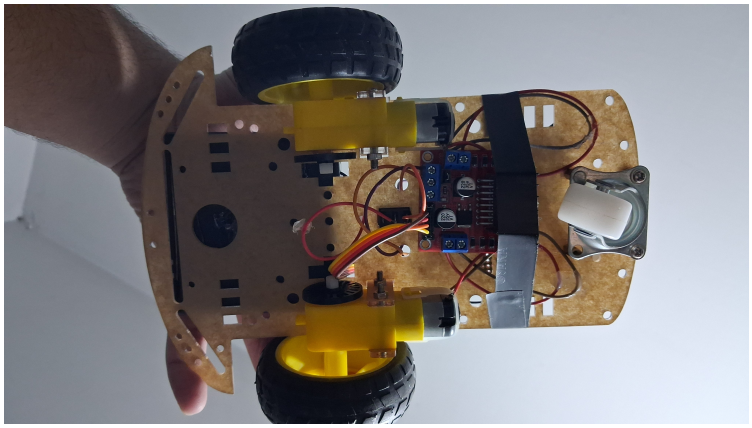

Figure 3: Robot bottom view

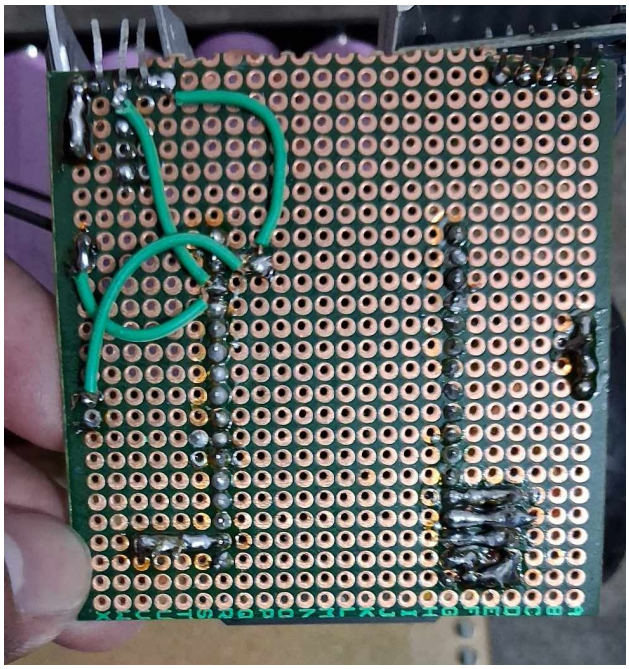

Figure 4: Veroboard Soldering

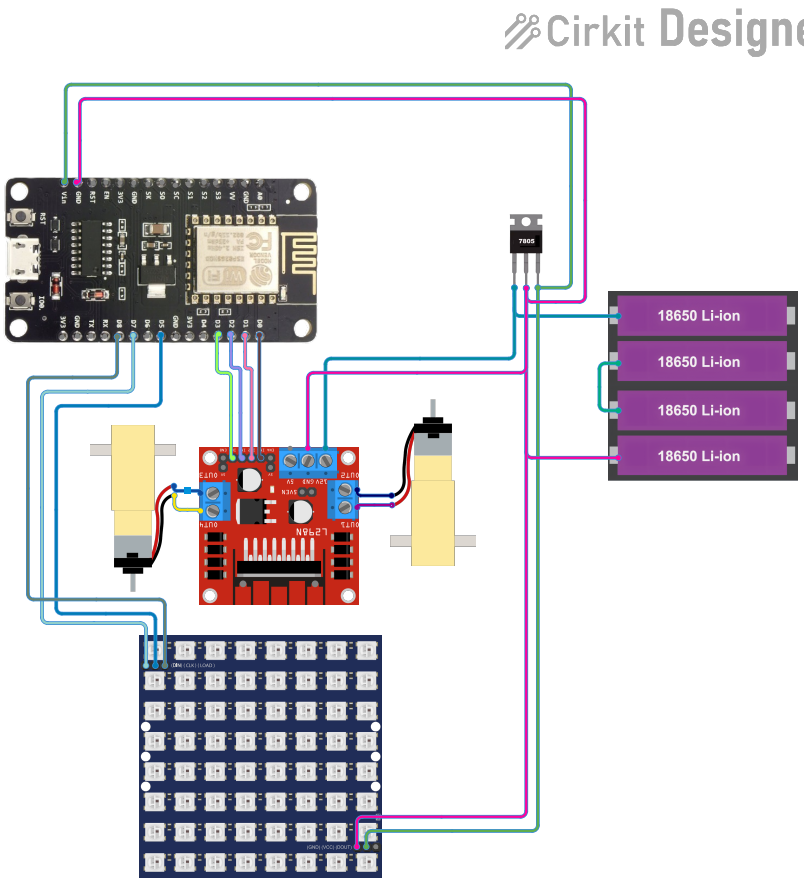

Figure 5: Circuit diagram of our project

5 Circuit Diagram and Wiring

According to the circuit diagram, our power supply was 4×18650 rechargeable batteries. Each battery supplies ≈ 3.7 V. Therefore, the total voltage received can range from $12 - 16$ V.

The voltage level on these batteries drains over time. On a full charge, the voltage might be as high as 16, but it will decrease over time. We call this an unregulated voltage that fluctuates over time.

It is good practice to supply a regulated voltage, a constant, unchanging voltage supply to the MCU. Therefore, we used the LM7805 voltage regulator. It takes our unregulated voltage and converts it into a steady supply of 5V. Additionally, this voltage regulation is susceptible to heating; therefore, a heatsink was placed to aid cooling.

The 5V from the regulator output goes into the V_{IN} pin of the NodeMCU board and is also supplied to the V_{CC} pin of the MAX7219 LED matrix.

In addition, the battery voltage was supplied to the L298N motor driver. There was a 5V voltage input option as well. However, we stuck with the 12V input, reasoning being that

we need to supply a good amount of power for our high-torque geared motors.

Because we supplied the (fully charged) battery voltage of 16V to the 12V input of the motor driver, the motors had more power than necessary and would drive at very high speeds. Therefore, for proper operation, a pulse width modulation (PWM) speed control should be introduced. Due to time constraints, speed control was not achieved.

With the power situation sorted, let us proceed with the control and peripherals. The NodeMCU is connected to the motor driver, the D0, D1, D2, D4 pins from the NodeMCU are connected to the IN1, IN2, IN3, IN4 pins on the motor driver respectively.

Furthermore, the D5, D7, and D8 pins are connected to the DIN, CLK, and the LOAD pin of the LED Matrix module respectively.

A crucial thing to note is that all the ground connections from all the components (MCU, motor driver, LED Matrix, voltage regulator, and the negative terminal from the battery should **all** be shorted with eachother. This node becomes a common reference point for all the sources. And as such, all voltage readings will be with respect to this node.

6 Software and Programming

To program the MCU, we used the arduino IDE. In the preferences settings for the Arduino IDE, the additional board URL should be set with this link to be able to download the supported board for ESP8266.

Afterward, in the board manager we can download the necessary NodeMCU board, and similarly, we can download the necessary libraries from the library manager.

Two scripts need to be prepared. One is for the MCU, coded in C++ and the other is coded in Python. The link to a GitHub repository for this project is found here.

6.1 The INO file

Many may not be familiar as the sketch does not follow the traditional arduino sketch format. We followed the Blynk documentation inorder to set up our Device. The auth code, device ID, and device name were all copied from the Blynk dashboard. The authors invite everyone to make the same changes to their own scripts.

The pins were defined and the logic was set. The logic worked upon the trigger of virtual pins from the Blynk cloud. These virtual pin triggers work similarly to how an interrupt is executed.

Below shows one such function being called on its own when a virtual pin, V0, is raised.

```
1  BLYNK_WRITE(V0)
2  {
3    // Set incoming value from pin V0 to a variable
4    int temp = param.asInt();
5    if (temp == 1) {
6      Blynk.virtualWrite(V1, 0);  // Turning off other virtual pins
7      Blynk.virtualWrite(V2, 0);
8      Blynk.virtualWrite(V3, 0);
9
10     allStop();        // Reset motor from any transient state
11
12     digitalWrite(IN1, LOW);    // Motor A anti-clockwise
13     digitalWrite(IN2, HIGH);
14
15     digitalWrite(IN3, LOW);    // Motor B anti-clockwise
16     digitalWrite(IN4, HIGH);
17
18     timer.setTimeout(5000L, allStop);   // Turns off after 5s
19
20   }
21 }
```

Lines 12 - 16 are the control signals for the motor driver following the below "truth table". The "x" represents dont care conditions. ↻ represents clockwise rotation of the motor. and similarly, ↺ means anti-clockwise rotation of the motors.

IN1	IN2	IN3	IN4	Motor A	Motor B
1	1	x	x	Active	Not active
1	0	x	x	↻	-
0	1	x	x	↺	-
x	x	1	1	Not active	Active
x	x	1	0	-	↻
x	x	0	1	-	↺

From the table we can refer how a motors direction can be changed from clockwise and anticlockwise just by changing the signals provided to the motor driver via the IN1, IN2, IN3, and IN4 pins.

Additionally, the code also sets up the dot LED matrix which can easily be coded to light up any specific LED in an arrangement. We used the matrix to create a smiley face for the robot.

But the critical part of the code is the following.

```
1  #include <ESP8266WiFi.h>
2  #include <BlynkSimpleEsp8266.h>
3  #include <Wire.h>
4
5  BlynkTimer timer;
6
7  char ssid[] = "you looked here";
8  char pass[] = "";    // Change to current wifi credentials
9
10 void setup() {
11   Serial.begin(115200);
12   pinMode(IN1, OUTPUT);
13   pinMode(IN2, OUTPUT);
14   pinMode(IN3, OUTPUT);
15   pinMode(IN4, OUTPUT);
16
17   Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
18 }
19
20 void loop() {
21   Blynk.run();
22   timer.run();
23 }
```

Lines 7-8 need to be adjusted to reflect the WiFi the ESP8266 has to connect inorder to access the internet.

Line 17 is responsible for connecting the ESP8266 to the Blynk Server. The serial monitor also shows more info about connection status, if necessary for debugging and troubleshooting.

Lines 21 and 22 are essential to keep in the `loop()` function. Additionally, it is not recommended and at all advised to keep anything else in said function, also the use of `delay()` is strictly prohibited from the Blynk documentation as the functions kills/ pauses all instruction set in the MCU. This in turn would cause our MCU to get disconnected from the Blynk server. To migitate this issue, Blynk has their own timer object that can be used to track the time and manage delays without disrupting the overall system.

6.2 The Python file

Webhooks

Luckily for our case, the virtual pins in Blynk can be triggered via a web request. Python handles this by sending a GET request to the webhook URL, it tells Blynk to update a virtual pin with a specific value, in our case, change its value to 1.

```
webhook_url = "https://sgp1.blynk.cloud/external/api/update?token="+auth+"&v0=1"

    try:
        response = requests.get(webhook_url)

        if response.status_code == 200:
            print("Webhook triggered successfully.")
        else:
            print(f"Failed to trigger webhook. Status code:
    ↪{response.status_code}")

    except requests.exceptions.RequestException as e:
        print(f"Error triggering webhook: {e}")
```

Above is the function responsible for executing the web request. Essentially it is the same thing as copy pasting the link `webhook_url` in your browser and going to that address. The entire code can be found in the same GitHub repository as referred to above. Please be mindful to change the auth token to the token used by your Blynk device.

Here is a breakdown of the link:

https://[region].blynk.cloud/external/api/update?token=[authToken]]&[virtualPin]=[value]

Where,

1. `[region]` - Can be found in the bottom right on your Blynk dashboard.
2. `[authToken]` - Unique for everyone, Your personalized "security key".
3. `[virtualPin]` - The virtual pins from Blynk for example: v0, v1, v2, ...
4. `[value]` - The virtual pin value, for example: '1' or '0'

Voice Recognition

The voice recognition is done by the `speech_recognition` library. It uses googles speech recognition API to convert the speech detected from the computer microphone into a string. Based on the value of the string, different webhooks get triggered.

Below is a snippet of the code:

```
recognizer = sr.Recognizer()

# Use the microphone as the audio source
with sr.Microphone() as source:
    print("Adjusting for ambient noise...")
    recognizer.adjust_for_ambient_noise(source)  # Adjusting for ambient noise
    print("Listening for speech...")

    while True:  # Keep listening indefinitely
        audio = recognizer.listen(source)  # Capture the speech

        try:
            print("Recognizing...")
            text = recognizer.recognize_google(audio)  # Use Google's speech
    ↪recognition
            print("You said: " + text)  # Output the recognized text

            if (len(text.split(sep=" ")) < 1):
                voiceCommands(text)
            else:                # a bunch of commands said in succession like,
    ↪"forward backward spin spin stop left"
                for words in text.split(sep=" "):
                    voiceCommands(words.lower())
                    time.sleep(5)
        except sr.UnknownValueError:
            print("Sorry, I couldn't understand the speech.")
        except sr.RequestError:
            print("Sorry, there was an error with the speech service.")
```

This script uses the computers default microphone as the input and after adjusting for the ambient noise, it starts to record, listening for the voice. Using Google's speech recognition, it converts the recorded voice and stores it as a string. This will be printed in the terminal as well, for the convinence of the User.

There is an additional functionality that was coded in Python to be able to say composite innstructions. Such that the car could execute them one by one from just one voice command. That is shown from Line 20 onwards.

7 Testing and Troubleshooting

A few issues were encountered for this project; they are listed as follows:

1. The use of IoT services like Blynk or IFTTT can be slow and produce very noticeable delays between giving the command the actuation in the robot.
2. The voltage provided by the batteries need to be at constant 12V otherwise the motor driver cannot let the motors spin.
3. Due to time constraints and a learning curve, speed control was not implemented.
4. Robot is prone to crashing into obstacles in smaller spaces.
5. Voice may not be picked up or be misinterpreted by the speech recognition API.
6. Robot sometimes not turning on because of poor contact in battery case
7. The device may not appear “Online” on the Blynk dashboard.
8. The device was subject to rate limiting from Blynk if many web requests were sent over a very short time. The paid version will solve this issue.

Preventative measures and debugging for the robot are limited as the circuitry is simple. However, some cautionary measures can be making sure the batteries are charged and supplying the necessary voltage; use the NodeMCU ESP8266 to let the motor driver supply PWM to the motors; making sure the microphone on the device is working/connected; making sure the batteries in the case are making contact in the terminals; making sure that the Wifi / WLAN network ID and password are correct.

8 Conclusion

This project successfully implements a basic voice-controlled robot that can perform simple movements, including "forward," "backward," "left," "right," "spin," and "stop," based on spoken commands. By using the ESP8266 microcontroller and the Blynk platform, the robot responds to voice commands processed through a Python script that utilizes Google’s speech recognition API. The design, built on an affordable budget, includes a motor driver for control and a Wi-Fi connection to send instructions via webhooks. Despite some challenges, such as occasional delays due to IoT services and power supply issues, the project meets its design goals. This prototype paves the way for more advanced voice-controlled robots, with potential applications in home automation or service industries.

Looking ahead, this project can be improved by using a Raspberry Pi pico W (W includes WiFi) running MicroPython. If the script can run directly on the Pico, then we can remove the need for webhooks. And the small time delays associated with it, and the possibility of getting rate limited.

9 References

1. GitHub Project Repository
2. LM7805 voltage regulator data sheet
3. MAX7219 LED matrix
4. Arduino IDE Additional board link for ESP 8266
5. Blynk Documentation
6. Python Library: Speech Recognition
7. Python Library: Requests
8. Zaman sir slides