**J.A.R.V.I.S Motor and LED Light Control System**

**1. Introduction**

This experiment is designed to control the two DC motors and an LED by using JARVIS AI voice assistant vis Arduino microcontroller board and L293D motor Driver.

**2.** **Arduino Code**

#define ENA 9

#define A1 6

#define A2 5

#define ENB 10

#define B1 4

#define B2 3

void setup() {

Serial.begin(9600); // Start Serial Communication

pinMode(ENA, OUTPUT);

pinMode(A1, OUTPUT);

pinMode(A2, OUTPUT);

pinMode(ENB, OUTPUT);

pinMode(B1, OUTPUT);

pinMode(B2, OUTPUT);

digitalWrite(A1, LOW);

digitalWrite(A2, LOW);

digitalWrite(B1, LOW);

digitalWrite(B2, LOW);

analogWrite(ENA, 255); // Full speed

analogWrite(ENB, 255); // Full speed

}

void loop() {

if (Serial.available()) {

String command = Serial.readStringUntil('\n'); // Read serial input

command.trim();

if (command == "LED\_ON") {

digitalWrite(7, HIGH);

Serial.println("LED is ON");

}

else if (command == "LED\_OFF")

{

digitalWrite(7, LOW);

Serial.println("LED is OFF");

}

else if (command == "MOTOR\_FORWARD")

{

digitalWrite(A1, HIGH);

digitalWrite(A2, LOW);

digitalWrite(B1, HIGH);

digitalWrite(B2, LOW);

Serial.println("Motors moving FORWARD");

}

else if (command == "MOTOR\_BACKWARD")

{

digitalWrite(A1, LOW);

digitalWrite(A2, HIGH);

digitalWrite(B1, LOW);

digitalWrite(B2, HIGH);

Serial.println("Motors moving BACKWARD");

}

else if (command == "MOTOR\_STOP")

{

digitalWrite(A1, LOW);

digitalWrite(A2, LOW);

digitalWrite(B1, LOW);

digitalWrite(B2, LOW);

Serial.println("Motors STOPPED");

}

}

}

**Explanation:** This Arduino code is designed to control two DC motors and an LED using serial commands sent from a Python-based AI voice assistant.

It utilizes L293D motor driver to control motor movement and Arduino’s serial communication to receive commands from a computer. The program continuously listens for commands via the serial port and executes corresponding actions such as turning the LED on/off, moving the motors forward/backward, or stopping them.

**Working of the program:**

1) Defining Motor Pins:

The code begins by defining Arduino pin assignments for the L293D motor driver. The Enable Pins (ENA and ENB) control the speed of the motors using Pulse Width Modulation (PWM), while direction pins (A1, A2, B1, B2) determine the rotation direction of the motors.

2) Setup Function:

The setup() function initializes the serial communication (Serial.begin(9600)) to allow data transfer between Python and Arduino. It also sets motor control pins as OUTPUT and ensures that all motors and the LED are in an initial OFF state.

3) Loop Function & Serial Communication:

In the loop(), the Arduino continuously checks if any serial data is available (Serial.available()). When a command is received, it is stored in a String, trimmed to remove extra spaces, and processed using if-else conditions.

4) LED Control:

If the received command is "LED\_ON", the LED connected to pin 7 is turned on using digitalWrite(7, HIGH). Similarly, "LED\_OFF" turns the LED off.

5) Motor Control:

"MOTOR\_FORWARD" sets A1 and B1 HIGH, A2 and B2 LOW, causing both motors to rotate forward.

"MOTOR\_BACKWARD" sets A1 and B1 LOW, A2 and B2 HIGH, making both motors rotate backward.

"MOTOR\_STOP" sets all motor direction pins LOW, stopping the motors.

6) Printing Status Messages:

After executing a command, Arduino prints a status message (e.g., "Motors moving FORWARD") back to the serial port, which can be read by Python.

**Concepts used in this code =>**

1) Serial Communication (Serial.begin(9600))

Establishes a data link between Arduino and a computer. Receives voice-based commands from Python over USB.

2) Digital Output (digitalWrite())

Used to control LED and motor direction. Setting HIGH or LOW changes the motor rotation direction.

3) Pulse Width Modulation (PWM) (analogWrite())

analogWrite(ENA, 255); and analogWrite(ENB, 255); set motor speed.

PWM allows variable motor speeds.

4) String Processing (String command = Serial.readStringUntil('\n');)

Reads serial data from Python. trim() function removes unwanted spaces for accurate command processing.

5) Conditional Logic (if-else)

Determines which action to take based on received commands. Also controls motor direction and LED states.

**3. Python program**

import serial

from JARVIS import listen, say

import time

arduino = serial.Serial('COM6', 9600)

time.sleep(2)

def send\_command(command):

    arduino.write((command + "\n").encode())

    say(f"Sent command: {command}")

def main():

    say("Welcome Boss")

    for i in range(3):

        say(f"Listening for command {i+1}")

        task = listen()

        if "light on" in task:

            send\_command("LED\_ON")

        elif "light off" in task:

            send\_command("LED\_OFF")

        elif "move forward" in task:

            send\_command("MOTOR\_FORWARD")

        elif "move backward" in task:

            send\_command("MOTOR\_BACKWARD")

        elif "stop" in task:

            send\_command("MOTOR\_STOP")

        else:

            say("Command not recognized, please try again.")

    say("Session complete. Thank You!")

main()

**Explanation:**

This Python program enables voice-controlled interaction with an Arduino using serial communication. The program listens to voice commands, converts them into text, and sends corresponding instructions to the Arduino over a USB connection. It utilizes the JARVIS voice assistant to recognize speech and respond with spoken feedback. The program allows the user to give three consecutive commands before terminating.

The script starts by establishing a serial connection with the Arduino at a 9600 baud rate using the serial.Serial() function. A 2-second delay (time.sleep(2)) ensures the Arduino is ready to receive commands. The send\_command(command) function encodes and sends the recognized command to the Arduino over the serial interface and provides an audio response using the say() function.

Inside the main() function, the assistant welcomes the user and enters a loop, allowing them to issue three voice commands. The listen() function captures the user’s voice input and converts it into text. The program then compares the spoken command against predefined options:

1) "light on" and "light off" control an LED connected to the Arduino.

2) "move forward", "move backward", and "stop" control two DC motors via the L293D motor driver.

3) If the spoken command matches a predefined instruction, it is sent to the Arduino. If the command is not recognized, the assistant prompts the user to try again. After three commands, the program ends the session, providing a final voice message.

**How this code connects to the Arduino program and enables Arduino?**

This Python script establishes communication with an Arduino microcontroller, allowing it to execute commands received via serial communication. The Arduino program is designed to continuously listen for incoming data, interpret commands, and perform the corresponding actions, such as turning an LED on or off or controlling the movement of DC motors. This seamless interaction between the Python voice assistant and the Arduino hardware enables hands-free control of physical devices.

The connection begins with serial communication setup. In the Python script, the command serial.Serial('COM6', 9600) initializes a USB connection between the computer and the Arduino. This ensures that Python can send data to the microcontroller. On the Arduino side, the function Serial.begin(9600); is executed within the setup() function, enabling the microcontroller to send and receive data at a 9600 baud rate. The baud rate must be the same on both ends to ensure proper communication.

When the user issues a voice command, Python captures the speech input, converts it to text, and determines the appropriate command. The function arduino.write((command + "\n").encode()) then transmits this command to the Arduino. The newline character (\n) is appended at the end of the message to indicate the end of the command, ensuring that the Arduino reads the instruction correctly.

On the Arduino side, the main loop continuously checks for incoming data using Serial.available(). If a command is detected, it is read using Serial.readStringUntil('\n'), which captures the message until the newline character is encountered. The trim() function is then used to remove any trailing spaces, ensuring that the command is processed correctly. Once the Arduino receives the command, it is compared with predefined conditions. For instance, if the command is "LED\_ON", the Arduino executes digitalWrite(7, HIGH), turning on the LED. Similarly, if the command is "MOTOR\_FORWARD", the motors are activated by setting the appropriate control pins (digitalWrite(A1, HIGH); digitalWrite(A2, LOW); for one motor, and digitalWrite(B1, HIGH); digitalWrite(B2, LOW); for the other).

After executing the desired action, the Arduino can also send feedback back to Python. A message such as "Motors moving FORWARD" can be printed on the serial monitor, which Python can read if necessary. This ensures that the system maintains a two-way communication channel, allowing for better monitoring and debugging.

Finally, the Python script allows the user to issue three consecutive commands before automatically ending the session. This prevents the need for manual restarts after every command, making the system more user-friendly. Once the session is complete, the user must restart the Python script to give more commands.

By integrating speech recognition, serial communication, and microcontroller programming, this project enables voice-controlled automation of physical devices. It has potential applications in robotics, home automation, assistive technology, and smart vehicle control, where users can operate devices hands-free using only voice commands.

**4. Hardware Connections**

|  |  |
| --- | --- |
| **L293D Pin** | **Arduino Pin** |
| ENA (Enable A) | Pin 9 (PWM Speed Control for Motor 1) |
| A1 (Input 1) | Pin 6 |
| GND | GND (Common Ground with Arduino and Power Supply) |
| GND | GND (Common Ground with Arduino and Power Supply) |
| A2 (Input 2) | Pin 5 |
| VCC | 5V |
| ENB (Enable B) | Pin 10 (PWM Speed Control for Motor 2) |
| B1 | Pin 4 |
| B2 | Pin 3 |
| Motor 1 Terminal 1 | First DC Motor Terminal 1 |
| Motor 1 Terminal 2 | First DC Motor Terminal 2 |
| Motor 2 Terminal 1 | Second DC Motor Terminal 1 |
| Motor 2 Terminal 2 | Second DC Motor Terminal 2 |
| VCC (Logic Power) | 5V |
|  |  |
|  |  |

**5). Link of the video:**

[**https://youtube.com/@skynet4073?si=BMG42BcJwrxrKG7d**](https://youtube.com/@skynet4073?si=BMG42BcJwrxrKG7d)