**IHU: School of Science and Technology**

**Program**: Mobile and Web Computing: Internet of Things Applications

**Course**: Internet of Things Fundamentals

**Unit**: Coursework

**Title**: Speek2Task IoT Hub

**Instructor**: Pr. Stavros G. Stavrinides

**Student**: Christos Stylidis

**Date**: 26.1.2024

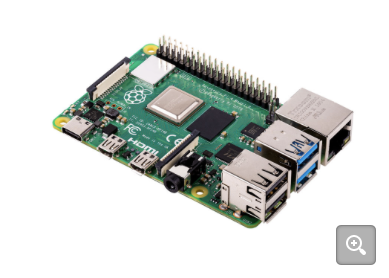
1. **Introduction**

This project, designed for the Internet of Things Fundamentals course, introduces a voice-activated task manager leveraging a Raspberry Pi. The system employs voice commands to add tasks to a Notion database, demonstrating an innovative use of IoT technologies for practical applications. This setup exemplifies the integration of hardware and software to create a smart, interconnected environment.

2. **Hardware and Software Requirements**

**Hardware Components**:

* **Raspberry Pi 4 Model B/4GB**: Serves as the central processing unit, handling all data processing and system control functionalities.



* **1.3 inch IIC SH1106 OLED LCD Display module 3.3v-5v**: Acts as the output device, displaying task statuses and system messages.



* **AirPods Pro**: Used as the voice input device, capturing user commands with clarity.



* **LEDs** (White, Green, Red) for status indication

A red led with a red cap

Description automatically generated

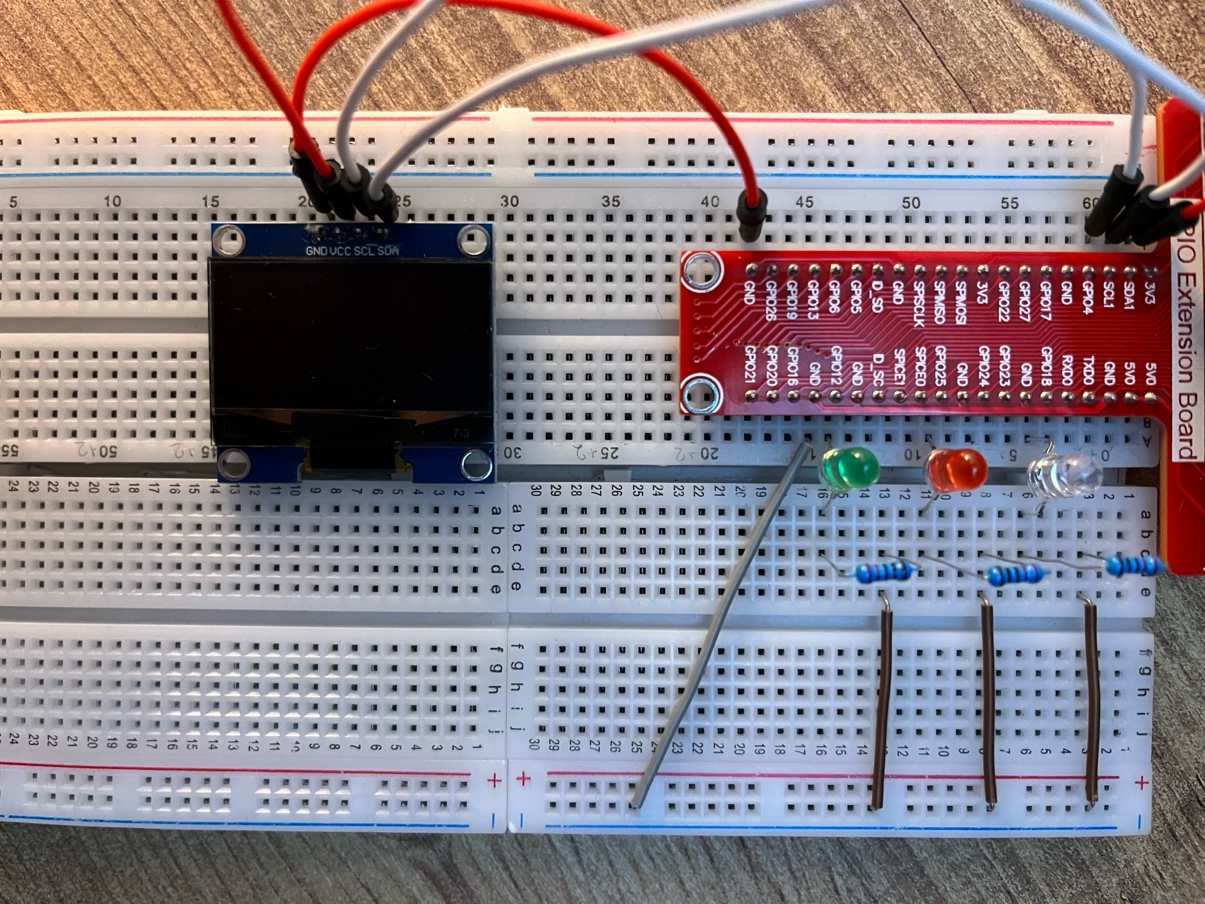
**Software and Platforms:**

* Node.js: Utilized for the main application, orchestrating the interaction between various APIs and system components.
* Python Flask Server: Manages the OLED display, receiving data from the Node.js app and displaying it.
* Google Speech API: Converts voice input to text, enabling the system to understand and process user commands.
* Notion API: Integrates with Notion for task management, adding tasks to a database based on voice commands.
* LED Control Software: Manages the LED indicators (white, green, red) to reflect the system's status.

3. **System Workflow**

This section describes the sequence of operations from the moment a voice command is received until the task is displayed and logged.

* Raspberry Setup



* The system starts with the white LED on, indicating readiness to accept voice commands.

A close up of a computer screen

Description automatically generated

A circuit board with wires and a wire

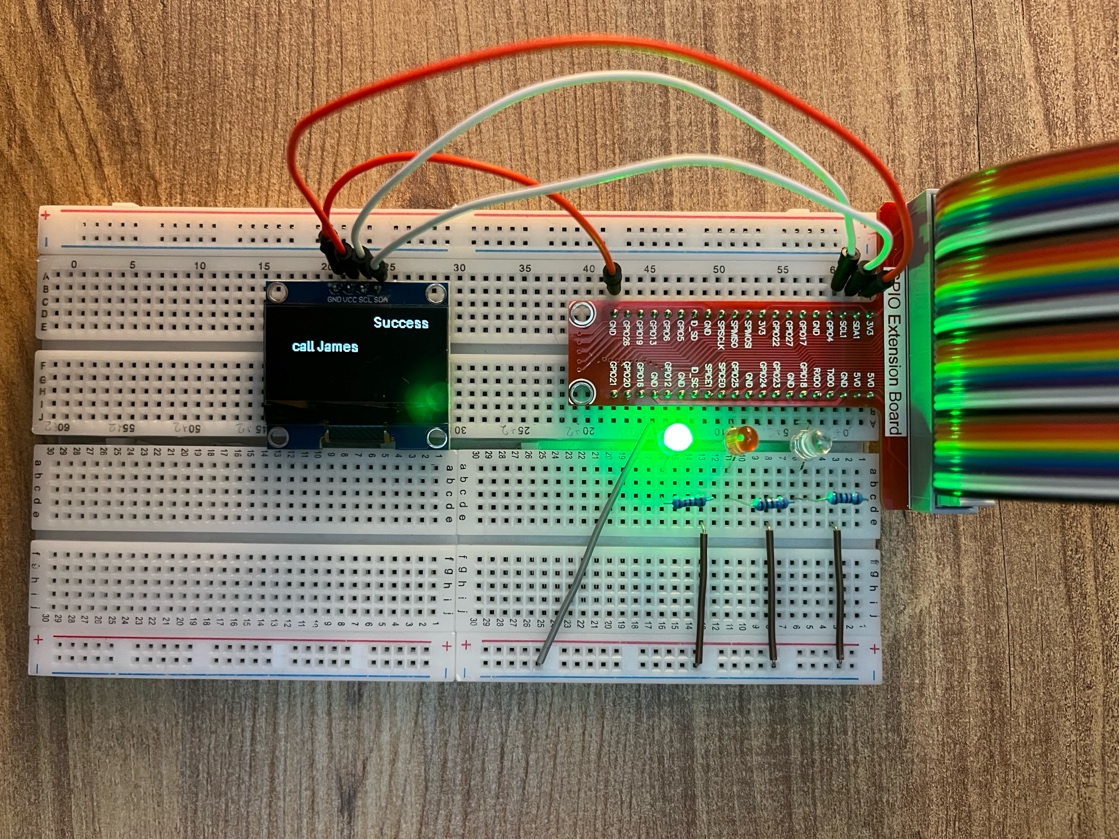
Description automatically generated

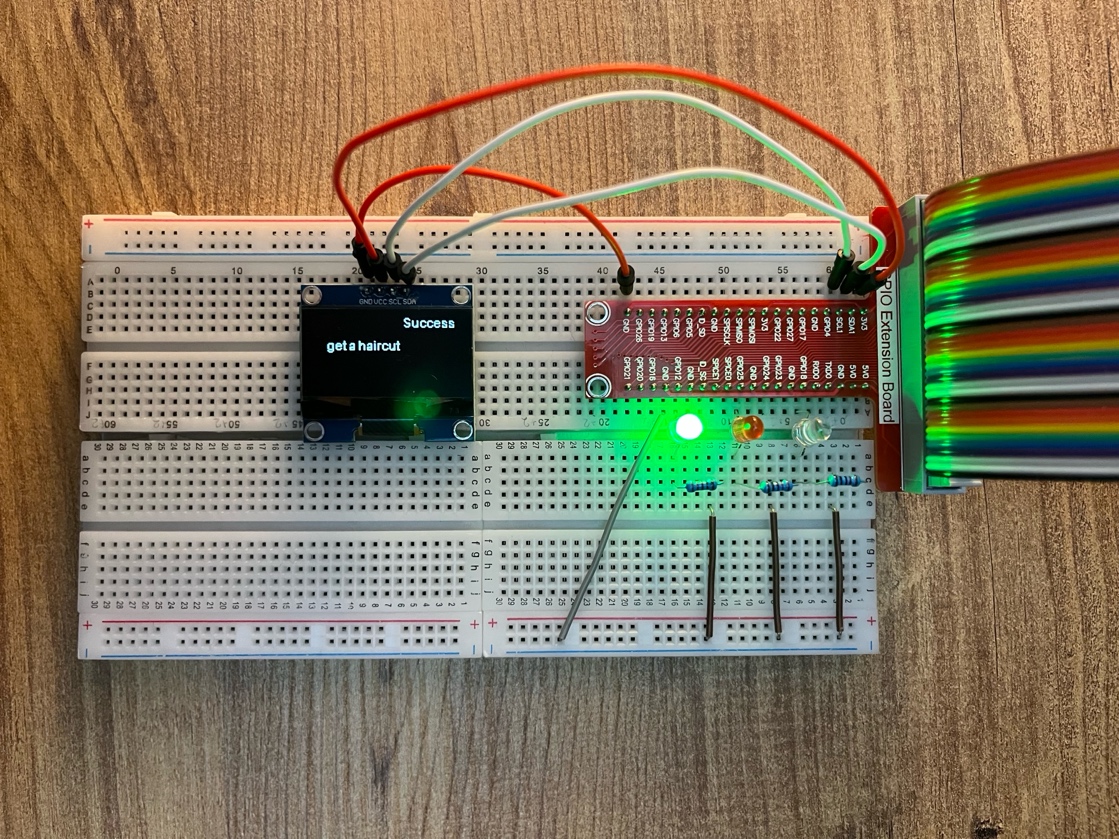
* Upon receiving a voice command, it's processed by Google Speech API and converted to text.
* The text is then sent to the Notion API, where the task is added to the database.
* If successful, a call is made to the Python Flask server, which controls the OLED display to show the task.

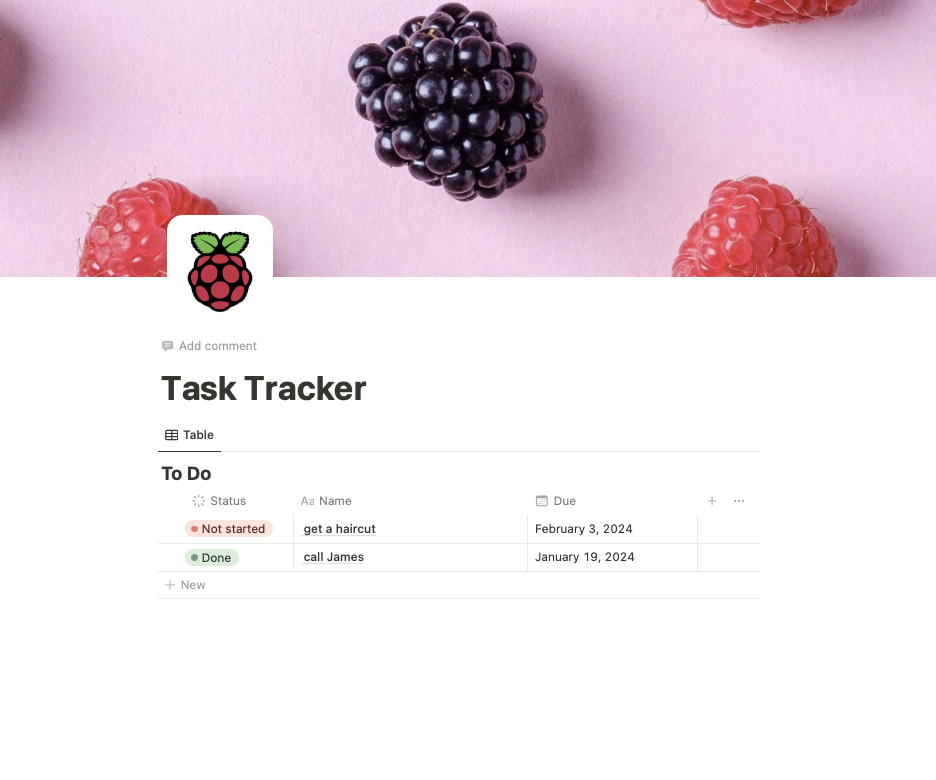
A screen shot of a computer

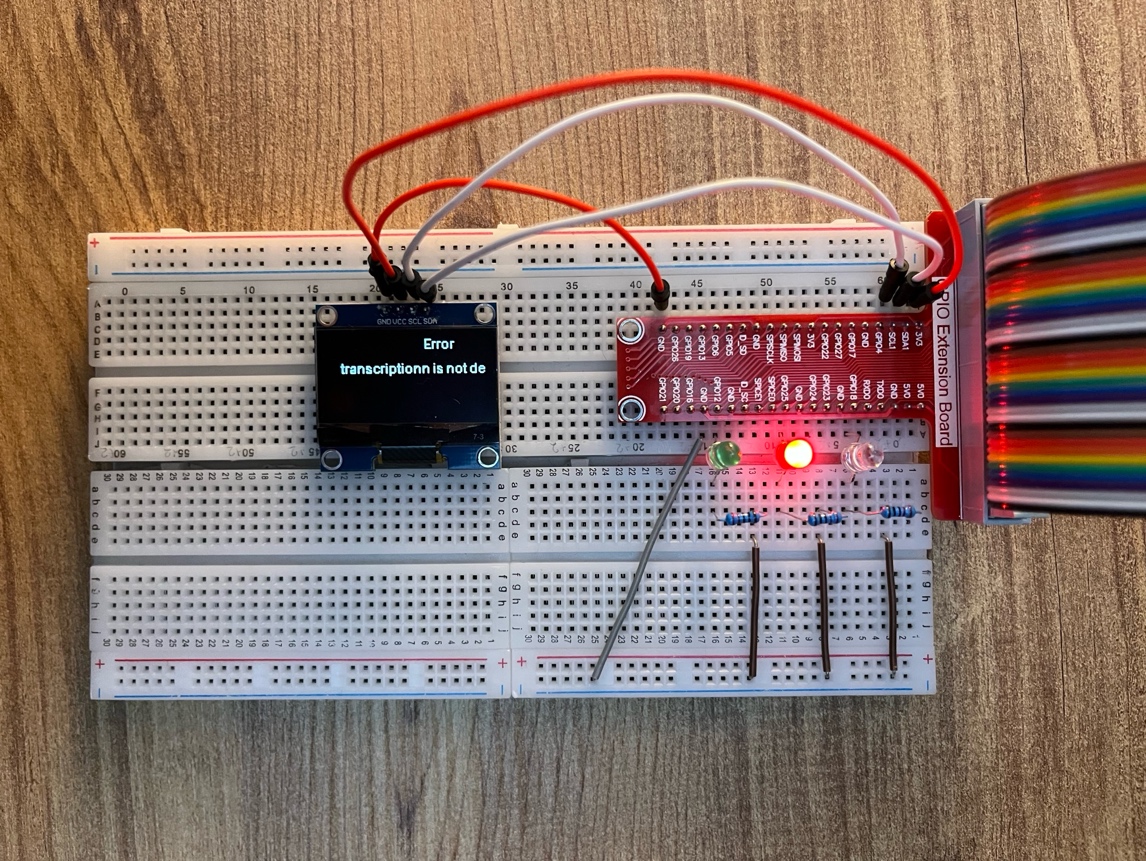
Description automatically generated

* Simultaneously, the white LED turns off, and the green LED lights up to indicate success.





* Notion Task Database
* In case of an error (e.g., a failure in adding the task to Notion), the red LED lights up, and the error message is displayed.



**4. Code Overview**

**Node.js Code**: The Node.js application (main.js) is the core of the system. It initializes and manages the voice input, processes the voice data using Google Speech API, interacts with the Notion API for task management, and communicates with the Python Flask server.

[View Node.js Code](https://github.com/acumenw/IHU/blob/127b1c1405ee6e73c198b33ad3e122e81a690c8e/Master/Internet%20of%20Things%20Fundamental/Coursework/main.js)

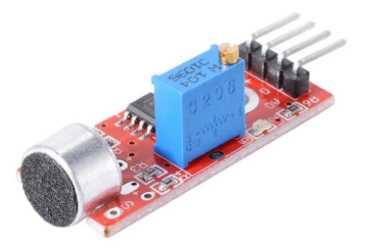
**Python Flask Server Code**: The Flask server (flask\_server.py) listens for data from the Node.js app and controls the OLED display. It receives the task string and displays it, providing a visual confirmation of the task addition.

[View Flask Server Code](https://github.com/acumenw/IHU/blob/127b1c1405ee6e73c198b33ad3e122e81a690c8e/Master/Internet%20of%20Things%20Fundamental/Coursework/flask_server.py)

**5. Challenges and Solutions**

Throughout the project, two significant challenges were encountered:

* Interfacing with the Display: Initially, there was difficulty in controlling the OLED display directly through Node.js. The solution involved implementing a Python Flask server, which offered better support for hardware control. This server listens for data from the Node.js application and manages the display accordingly.
* Voice Input Methods: Various input methods were explored, including different microphones and sound sensors. However, these attempts faced issues like unclear voice capture and complex circuit setups.
* Components Used:
  1. USB Microphone for Raspberry Pi
  2. High Sensitivity Sound Microphone Sensor KY037
  3. Electret Microphone Amplifier - MAX4466 with Adjustable Gain



1 2 3

* The breakthrough came with the use of AirPods Pro connected through Raspberry Bluetooth, which provided clear voice capture, crucial for effective use of the Google Speech API.

**6. Conclusion and Future Work**

Completing this project has been a rewarding experience, demonstrating the potential of IoT in everyday task management. Future enhancements include adding a security feature like voice-activated password protection and improving the display aesthetics for a more engaging user interface alongside with involving more apps integration.

**7. References**

* Flask Server Development: [Raspberry Pi Flask Web Server](https://projects.raspberrypi.org/en/projects/python-web-server-with-flask/0)
* OLED Display Integration: [SSD1306 OLED Display with Raspberry Pi](https://circuitdigest.com/microcontroller-projects/ssd1306-oled-display-with-raspberry-pi)
* LED Control: [Raspberry Pi LED Connection](https://projects.raspberrypi.org/en/projects/rpi-connect-led) and [Node.js LED Control](https://www.w3schools.com/nodejs/nodejs_raspberrypi.asp)
* Google Speech API: [Save speech to text in local using node js #121](https://github.com/googleapis/nodejs-speech/issues/121)
* Notion API: [Notion API Official Documentation](https://developers.notion.com/reference/post-page)