**Project**: Sound Level Indicator

**Document**: Design Document

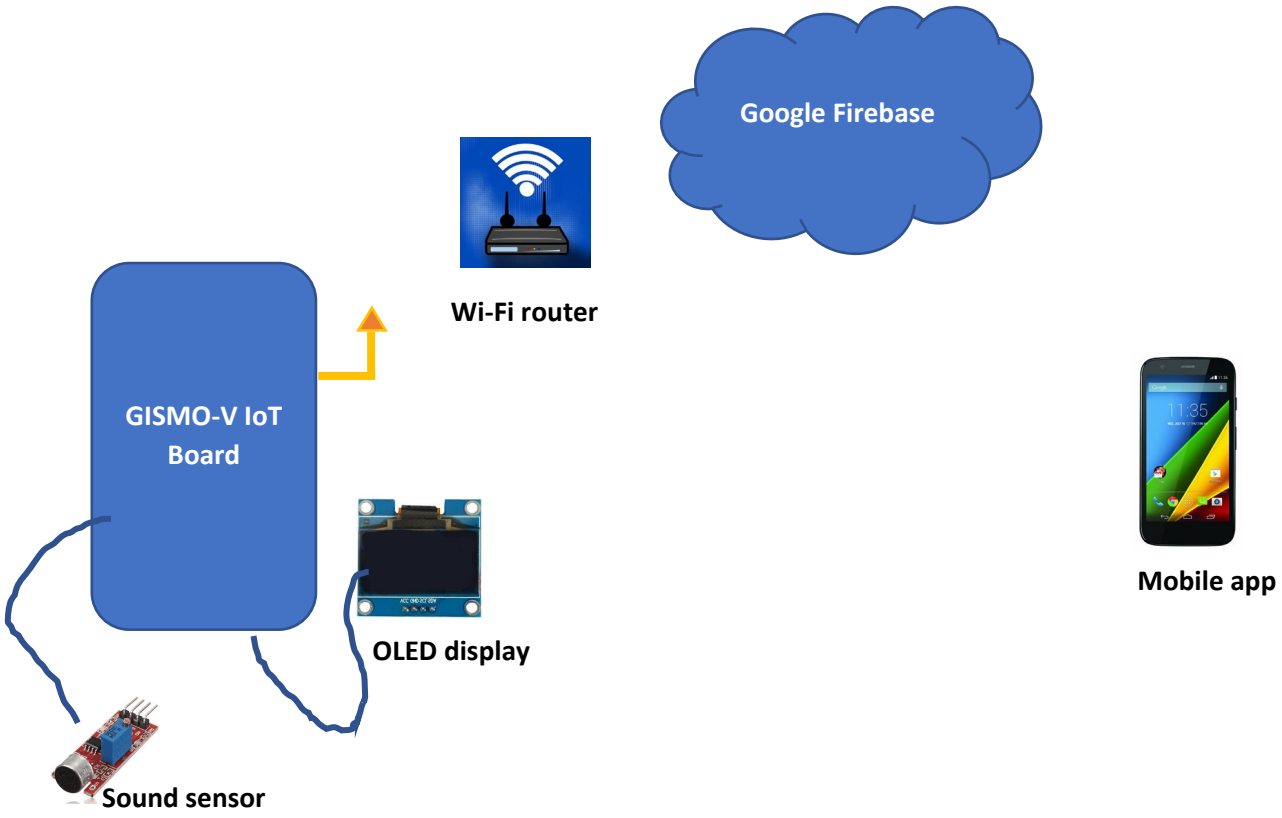
**Description**

The project aims to design and implement a sound level indicator with the following features:

- Monitoring of sound level using a sound sensor module
- Conversion of sound level into db
- Classification of ambient sound level as noisy or silent based on a threshold level
- Display of sound level in db and ambient level classification as well as facility for changing threshold level in mobile app

The sound sensor module contains a microphone which will convert sound signals to electrical signals. A comparator with a potentiometer for sensitivity setting is used to produce a digital output. The board also gives an analog output which is used in the project.. The analog output will be read by an ESP32 controller. The ESP32 controller is a 32-bit dual core processor with Wi-Fi and Bluetooth capabilities built on-chip. The Wi-Fi capability is used to join a Wi-Fi network and connect to the Internet. The soil moisture values are pushed to a cloud database – Google's Firebase which is created for the project. The soil moisture limit is also stored in the database and can be set from the user's mobile app. The limit is read by the ESP32 and used to switch the pump on or off. Ths witch status also is transferred to the cloud database. The database credentials – the host URL and the database authentication key are to be fed into the firmware. The Firebase database is a key-value based database and using the Firebase credentials and the key the values are accessed and displayed in the mobile app of the user.

The different components in the project are:



**Hardware**

The hardware for the project consists of:

- GISMO-V board with:
    - ESP32 dual-core 32-bit processor with Wi-Fi and BLE
    - Soil moisture sensor module
    - Relay for switching the pump
    - 0.96” OLED display with 128x64 resolution

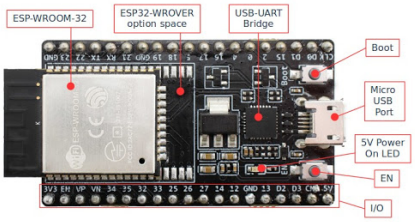
The sound sensor module contains a microphone which will convert sound signals to electrical signals. A comparator with a potentiometer for sensitivity setting is used to produce a digital output. The board also gives an analog output which is used in the project.

- VCC connected to 3.3V
- GND
- DO: Digital output
- AO: Analog output

The analog output of the sensor module is used and is connected to GPIO36 of ESP32, which is an  analog input pin. The ADC in the ESP32 is 12-bit and the ADC output will vary from 0 to 4095.

The OLED display is a graphic display with I2C interface to the microcontroller. The SCL and SDA lines of the OLED display are connected to GPIO22 and GPIO21 pins of the ESP32. These are the I2C pins of the ESP32. The supply voltage for the OLED display is 3.3V

The ESP32 development board used is the ESP32 Dev Kit. The ESP32 board has a micro USB for programming, powering up and for transfer of data on serialline to and from the laptop. It has an on-board LED connected to GPIO2 which can be used for debugging purposes. It has a RESET and a BOOT button. The BOOT button needs to be kept pressed while downloading the program to the board. The Dev Kit also has 4 MB of external flash memory.



**Firmware**

The Arduino IDE is used to develop the firmware. The ESP32 board support package is downloaded and added to the existing boards in the IDE. The particular board to be selected is the ESP32 Dev Kit. The firmware can be divided into the following blocks:

- Sensor interface
- OLED display interface
- Internet connectivity
- Cloud database interface

**Sensor interface**:

No separate library is used to interface the sound sensor module. The sensor analog output is connected to GPIO36, an analog input pin of the ESP32. The ADC is 12-bit and will give a digital value in the range 0 to 4095.

A timing window is defined and within the timing window the max anc min values for the ADC values are found out. Using these two values, the peak-to-peak value is computed. The db level of the sound is calculated using this peak-to-peak value.

A threshold level is used to classify the ambient sound level as silent or talking. This threshold sound level can be set from the mobile app

**OLED display interface**:

The two libraries used for the OLED display interface are:

Wire,h : For basic I2C interface

SSD1306.h: For display specific functions

The SSD1306 library provides a SSD1306 class. An object belonging to that class is created with the following initialization parameters:

- 7-bit I2C address of the display (0x3c)
- SCL pin of microcontroller (22)
- SDA pin of microcontroller (21)

The functions supported by the SSD1306 class are:

- init(): For initialization
- setFont(); Takes the font as a parameter. The font will decide the size of the text that will be displayed. The fonts supported are:
    - ArialMT_Plain_10
    - ArialMT_Plain_16
    - ArialMT_Plain_24
- clear(): For clearing the display
- drawSring(0,0,"fgfhgfhfhfh"): Takes three parameters – the x, y co-ordinates of the strat of the string and the string to be displayed
- display(): No parameters. The memory buffer is transferred to display

**Internet connectivity**

The in-built WiFi object is used to connect to the WiFi network. The WiFi object has the following functions:

- begin(): Takes two parameters:
  - SSID of the WiFi network to which we want to connect
  - Password of the WiFi network
- status(): Tells us whether the ESP32 is connected to the WiFi network or not.
- localIP(): The IP address dynamically assigned to the ESP32 by the access point (router)

**Cloud database access**

The cloud database used in the project is Google's Firebase. It is a No-SQL database in which data is stored as Key-Value pairs. The following are the credentials of the database required for access:

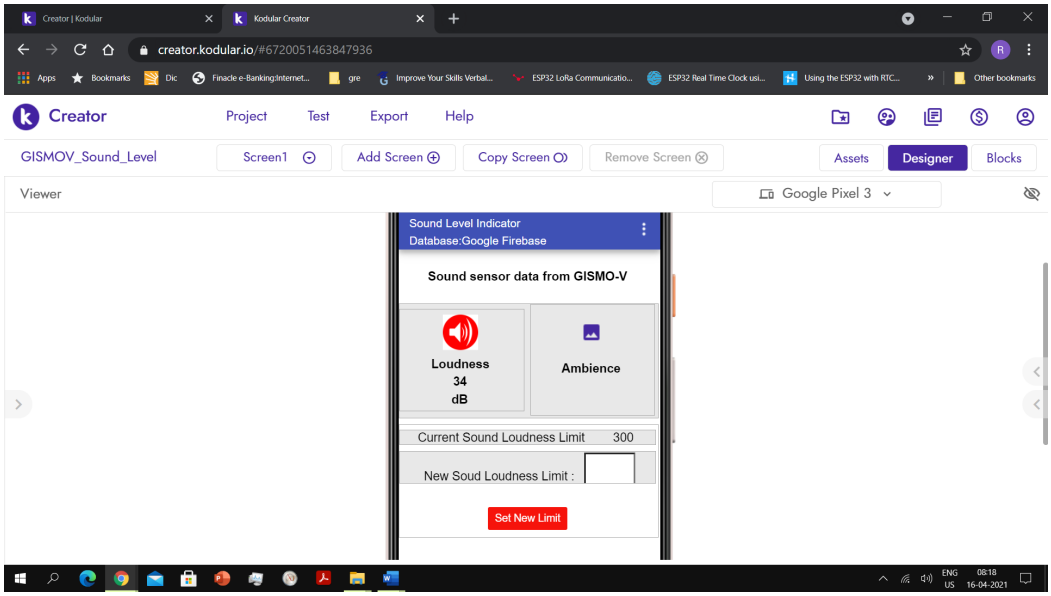
- Host URL
- Database authentication key

These credentials will be put in the ESP32 firmware to access the Google Firebase. The keys used to store the sound level parameter values in the project are:

- IOTLAB/Sound_Level_Indicator/Sound_Level_Class
- IOTLAB/ Sound_Level_Indicator /Sound_Level_Limit
- IOTLAB/Sound_Level_Indicator/Sound_Level_Value

**Mobile App development**

The Kodular rapid mobile app development utility is used for mobile app development. The utility has a Developer mode and a Blocks mode. In the Developer mode, there are different functional blocks available – User Interface, Sensors, Connectivity, Firebase, and so on. The UI gets defined in the Developer mode as under :



In the Blocks mode, a timer with timing set to 2 seconds will fetch the sound level and class values from the Firebase database and fill it in the appropriate place in the UI. To access the Firebase datase its credentials will be put in the Firebase component of the Kodular app