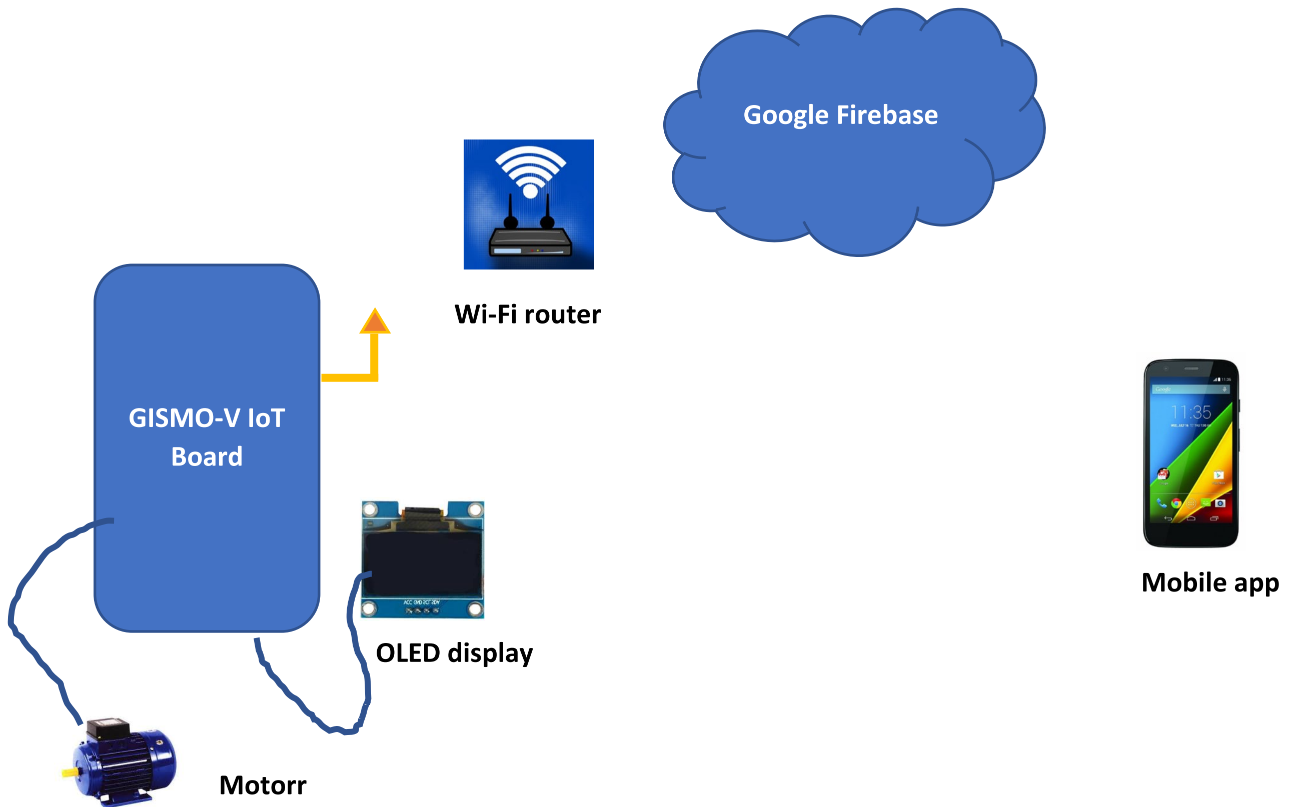
Project: Remote Motor Control using IOT

Document: Design Document

Description

The project aims to provide the user with the facility of turning on/off a motor that is remotely located from his mobile app. The app will have buttons to turn the motor on and off. The specific button press event triggered will be used to send a motor command to the Firebase database in the cloud. This motor command will be read by the GISMO-V board which has an ESP32 microcontroller and can be connected to the Internet and fetch the motor command from the cloud database. Based on the command the ESP32 will operate a relay to appropriately switch the motor. 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 motor commands are pushed to a cloud database – Google's Firebase which is created for the project.. The command is read by the ESP32 and used to switch the motor on or off. 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 commands are accessed

The different components in the project are:



Hardware

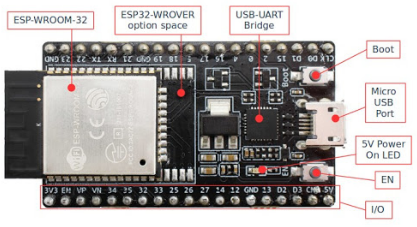
The hardware for the project consists of:

- GISMO-V board with:
    - ESP32 dual-core 32-bit processor with Wi-Fi and BLE
    - Relay for switching the motor
    - 0.96" OLED display with 128x64 resolution

The relay used is a sugar cube relay which operates at 5V. A transistor drive circuit is present on the board for driving the relay. The base of the transistor is connected to GPIO13. By making this GPIO pin high/low the relay can be switched on/off. The Normally Open(NO) contacts of the relay are brought out onto terminal blocks. An AC operated contactor is switched on/off using the relay and the contactor contacts are used to switch on/off the motor.

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:

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

Actuator interface:

GPIO13 is connected to a transistor drive and is used to switch the relay on and off using basic Arduino digital write commands.

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:
    - o ArialMT_Plain_10
    - o ArialMT_Plain_16
    - o 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:
    - o SSID of the WiFi network to which we want to connect
    - o 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

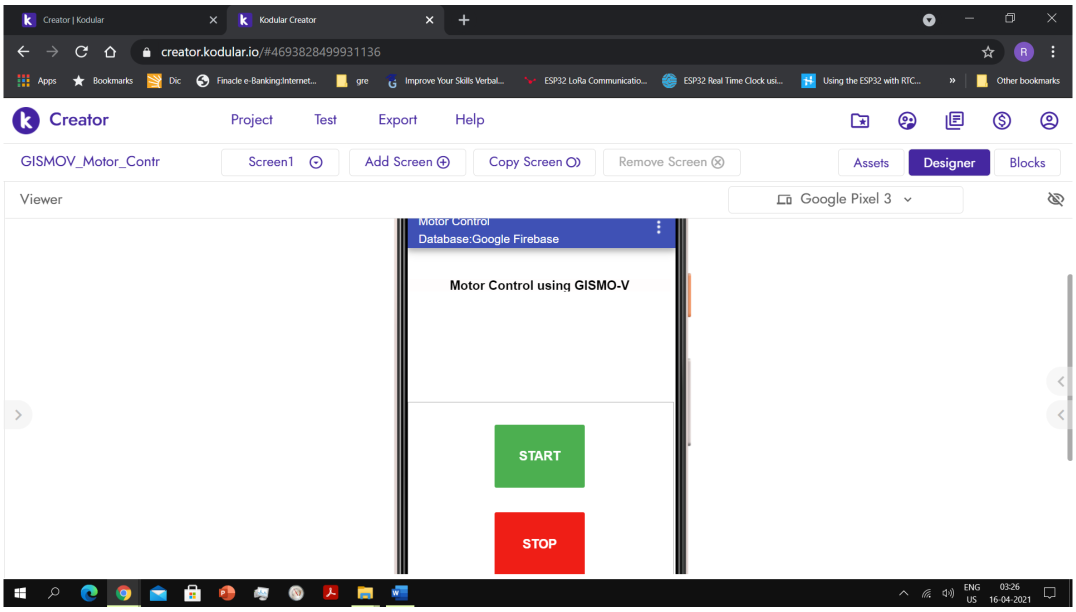
- Database authentication key

These credentials will be put in the ESP32 firmware to access the Google Firebase. The key used to store the motor command values in the project is:

- IOTLAB/Motor_Control/Motor_Cmd

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, depending on which button is pressed, the appropriate motor command value will be stored in the Firebase database. To access the Firebase database, the host URL and the database authentification key should be provided to the Firebsse component