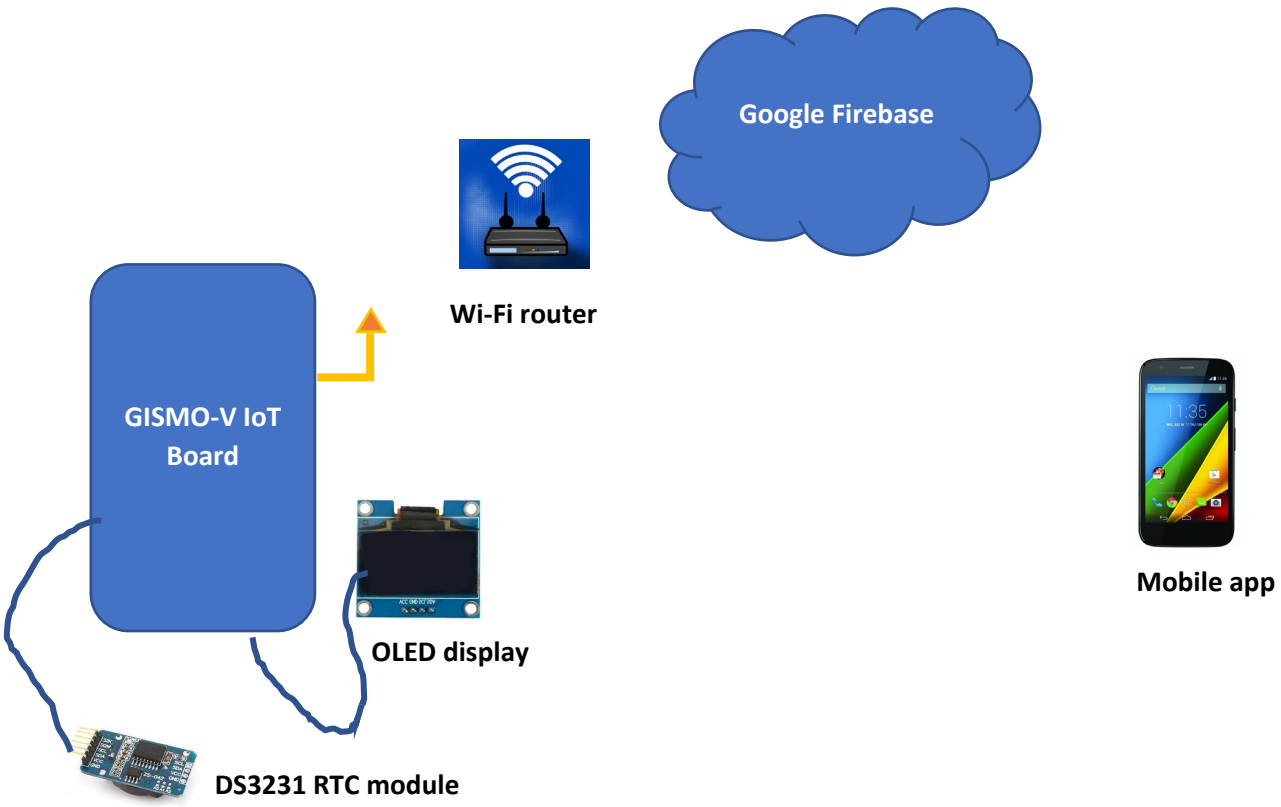
Project: Appliance based control using RTC

Document: Design Document

Description

The aim of the project is to switch on an appliance during a prescribed slot time. The start time and the end time of the slot will be settable from a mobile app. These values will be stored in a cloud database and will be read by the ESP32 processor on the GISMO V board. A Real Time Clock module DS3231 with battery backup is connected to the GISMO V board. The interface between the RTC module and the ESP32 is I2C. 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 time slot parameters are pushed to a cloud database – Google's Firebase which is created for the project. 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 slot values are accessed and compared with real time read from the RTC module

The different components in the project are:



Hardware

The hardware for the project consists of:

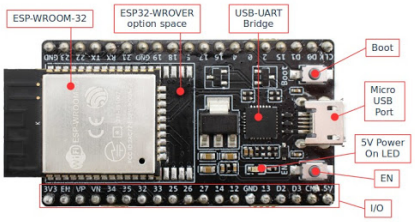
- GISMO-V board with:
    - ESP32 dual-core 32-bit processor with Wi-Fi and BLE
    - DS3231 RTC module
    - Relay
    - 0.96” OLED display with 128x64 resolution

The DS3231 RTC module is interfaced with the ESP32 using the I2C interface. The I2C interface has two signals – SCL and SDA. These pins on the BME280 are connected to I2C pins on ESP32 – GPIO22 and GPIO21. The DS3231 works on 3.3V supply and is provided with a battery backup

The relay on the GISMO V board is used to turn on the appliance when the actual time is with the strat and end times of the slot. There is a transistor based drive circuit for the realy operation. This drive circuit is connected to GPIO13 of the ESP32

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:

The two libraries used for interfacing the BME280 sensor board are:

- Wire.h
- RTClib.h

The RTClib library provides a class called RTCDS3231. An instance of the class is created. The class supports the following methods:

- begin(): This method takes no parameters and is used for initialization. The function returns true is the initialization is successful
- adjust(): This method is for setting the time of the RTC. The current date and time is passed to this function
- now();The return value of this function is an object that belongs to the DateTime class. Ths class has the following members:
    - year
    - month
    - day
    - hour
    - minute
    - second
- The start time and end time of the slot are defined in the HH:MM format and the real time in the HH:MM format is read from the RTC module and if the real time is within the slot time, the appliance is switched on using the relay on the GISMO V board

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:

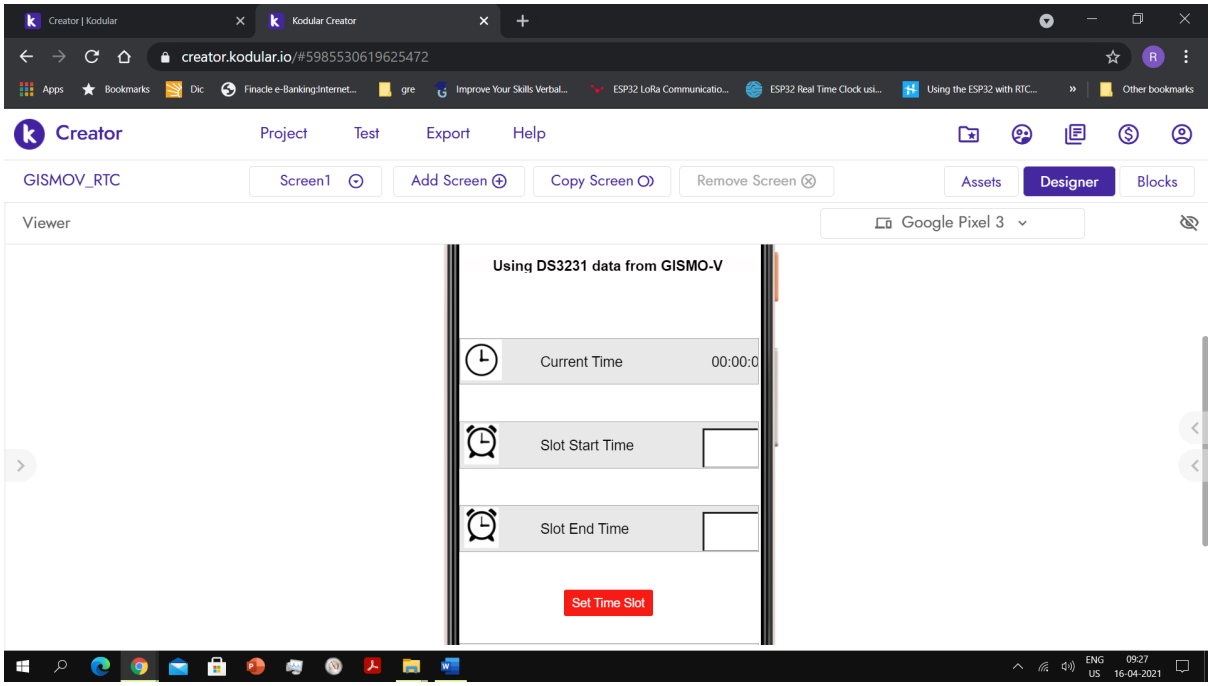
- Host URL
- Database authentication key

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

- IOTLAB/Appliance_Control/Appliance_Status
- IOTLAB/ Appliance_Control /Current_Time
- IOTLAB/ Appliance_Control /Start_Time
- IOTLAB/ Appliance_Control /End_Time

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, when the user sets the start time and the stop time and presses the Set Time Slot button, these times will be stored in the Firebase database against the respective keys. The GISMO V board will update the real time value as well as the status of the appliance (ON/OFF) periodically