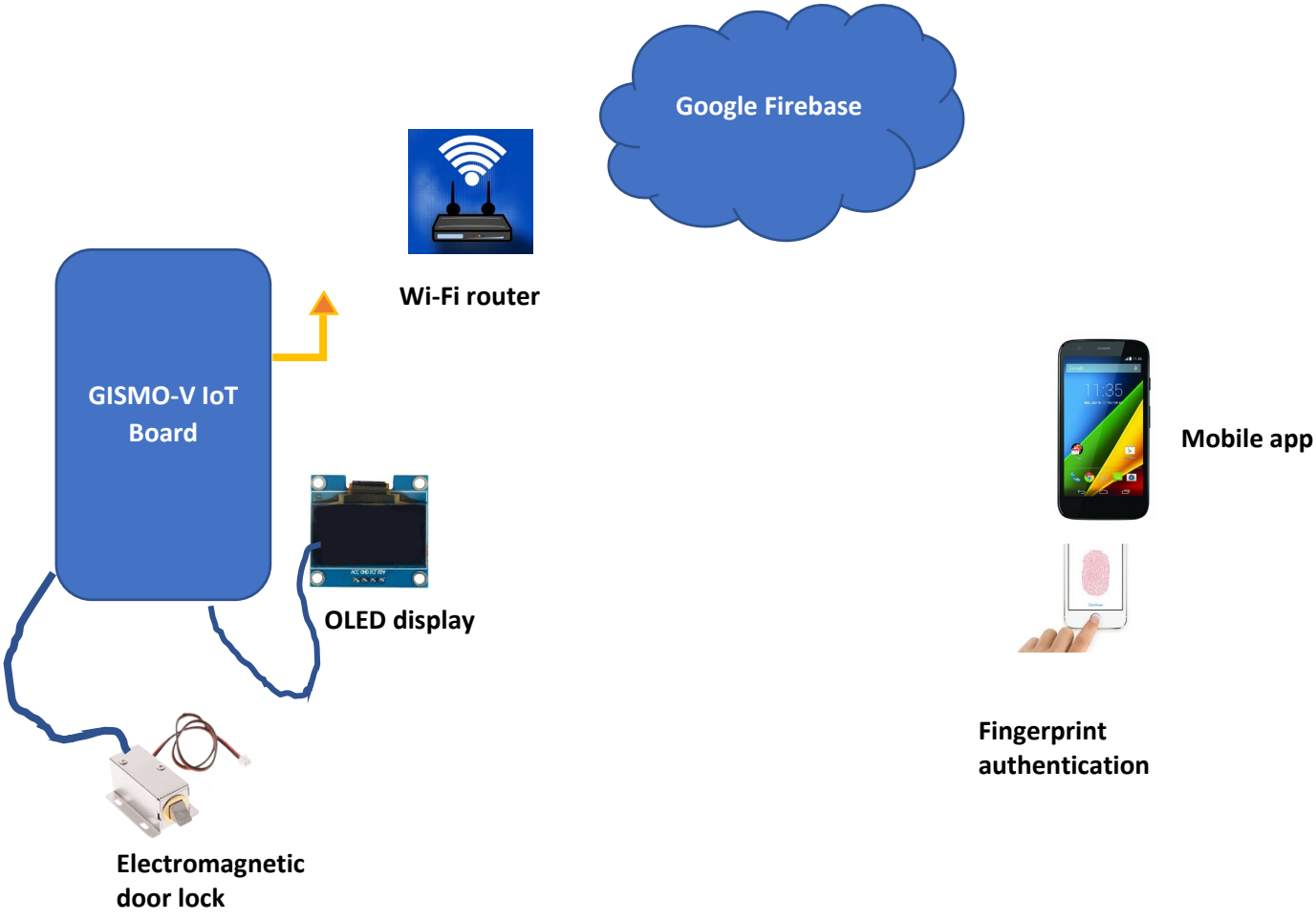
Project: Biometric Access Control using IOT

Document: Design Document

Description

The aim of the project is to provide access control by opening and closing a lock remotely using the fingerprint of the user as the key .The user operates a mobile app which will prompt him for providing his fingerprint. The fingerprint validation is done in the mobile app itself and the validation status – verified or failed is updated in a cloud database. This validation status is read by the GISMO-V board with a ESP32 microcontroller. If the validation is successful the ESP32 will operate a relay which will in turn open a electromagnetic door lock to open the door. 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 validation status is 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 the fingerprint validation status is accessed by the ES{32

The different components in the project are:



Hardware

The hardware for the project consists of:

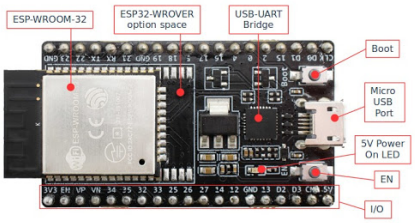
- GISMO-V board with:
    - ESP32 dual-core 32-bit processor with Wi-Fi and BLE
    - 12V solenoid electromagnetic door lock
    - 0.96” OLED display with 128x64 resolution

The electromagnetic door lock works on 12V DC. When the supply is on, the lever of the solenoid is retracted and the door can be opened. When the suppy is cut off, the lever moves forward into the slot in the frame of the door and the door gets locked.

The supply to the electromagnetic lock is switched on and off by a 5V sugar cube relay that is there on the GISMO-V board. A transistor-based drive circuit switches the relay on and off. The base of the transistor is driven by 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:

In this project, the biometric fingerprint sensor of the mobile phone is used as a sensor. The Fingerprint component in Kodular has the following methods:

- Authenticate : Used to authenticate a fingerprint. This will be an asynchronous process and the result will be the firing of either of two events:
    - OnAuthenticationSucceeded
    - OnAuthenticationFailed
- In the blocks for these two events the authentication status : success or failure will be stored in the Firebase database
- In case of success, a timer is started and on expiry of the timer, the status is changed to failed again so that the door will be kept open for a short while after the user's fingerprint is successfully authenticated

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:

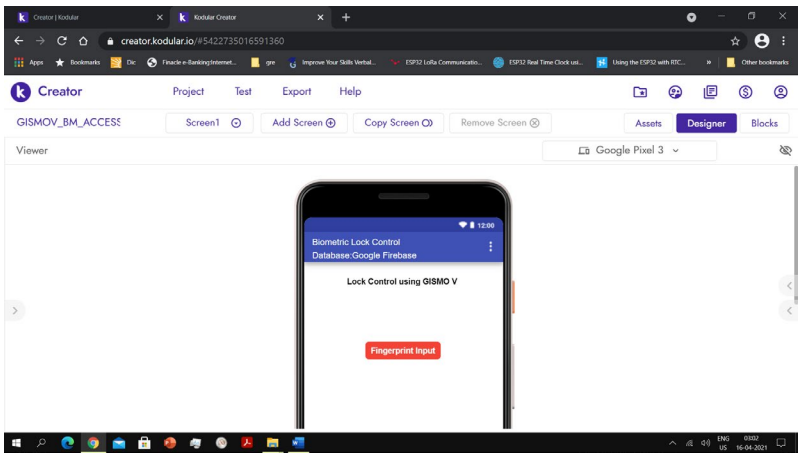
- Host URL
- Database authentication key

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

- IOTLAB/Ambient_Parameters/Temperature
- IOTLAB/Ambient_Parameters/Humidity

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, the AuthenticationSucceeded and AuthenticationFailed events will be used to push the authentication status to the Google Firebase database. This status will then be periodically accessed bt the ESP32 using the Firebase credentials to activate or deactivate the solenoid electromagnetic lock