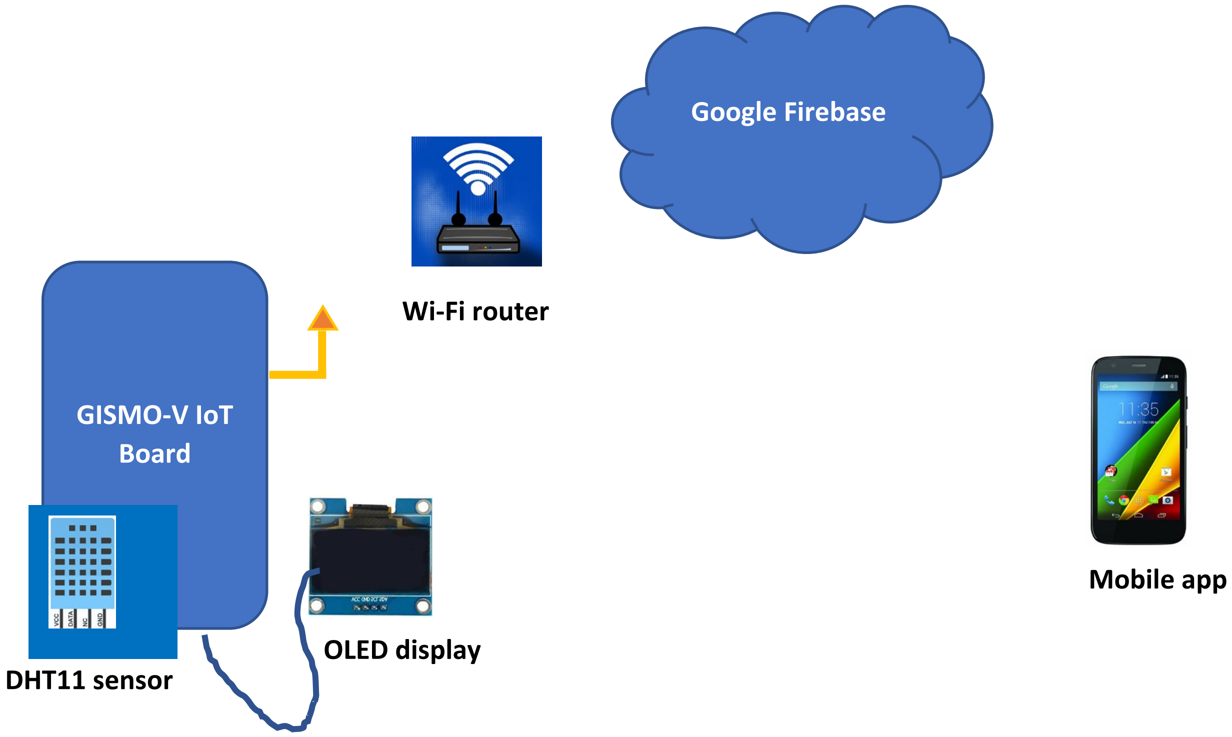
Project: Ambient Parameter Monitoring using IoT

Document: Design Document

Description

The project aims to provide the current ambient parameter values of temperature and relative humidity to the user in an app on his Android mobile phone. The update rate of the parameters will be twos seconds. The values will be read from the DHT11 sensor 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 temperature and humidity values 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 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
    - DHT11 Temperature and humidity sensor
    - 0.96” OLED display with 128x64 resolution

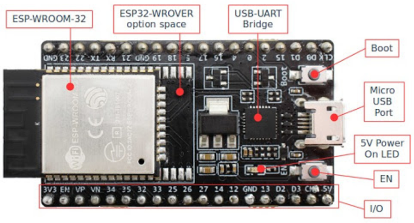
DHT11 has a single-wire serial output and produces 40-bit data output with the following format:

- First 2 bytes: Integer part of humidity in percentage
- Next 2 bytes: Decimal part of humidity
- Next 2 bytes: Integer part of temperature in deg C
- Next 2 bytes: Decimal part of temperature
- Last byte: Checksum

The DATA pin of the DHT11 is pulled up to 3.3V and is connected to GPIO27 of the ESP32 microcontroller. The DHT11 sensor works on 3.3V supply.

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 DHTesp.h library is used for the sensor interface. This library provides a class called DHTesp. An object of that class in instantiated. The class has the following methods:

- setup() : This function takes two parameters – the microcontroller pin to which the DATA pin of DHT11 is connected. In our case it is GPIO27. The second parameter is the type of DHT (DHT11, DHT22,DHT33 etc)
- getStatusString(): This function does not take any parameters. It returns an "OK" if status is OK.
- getHumidity(): No parameters are passed to this function. The return value is a float which gives the relative humidity as a percentage.
- getTemperature(): No parameters are passed to this function. The reurn value is a float which gives the temperature in degrees Celsius

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:

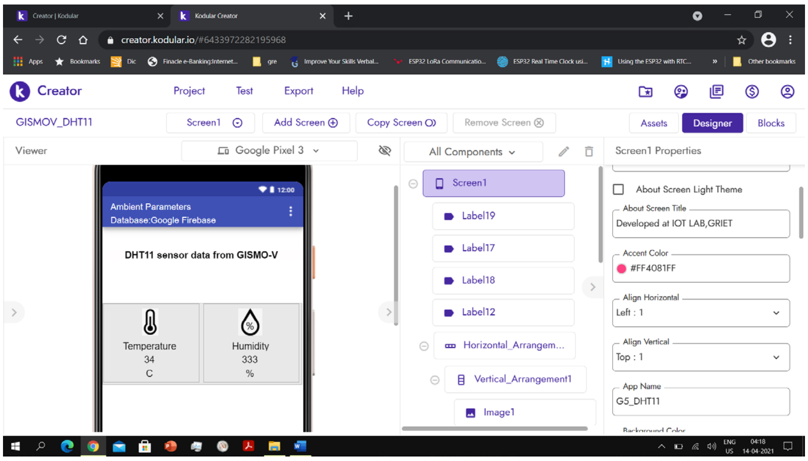
- Host URL
- Database authentication key

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 :



Ambient Parameters
Database:Google Firebase

DHT11 sensor data from GISMO-V

Temperature
34
C

Humidity
333
%

□ About Screen Light Theme

About Screen Title: Developed at IOT LAB,GRIET

Accent Color: #FF4081FF

Align Horizontal: Left : 1

Align Vertical: Top : 1

App Name: G5_DHT11

Background Color

In the Blocks mode, a timer with timing set to 2 seconds will fetch the temperature and humidity values from the Firebase database and fill it in the appropriate place in the UI. To access the Firebase dataset its credentials will be put in the Firebase component