

BÁO CÁO

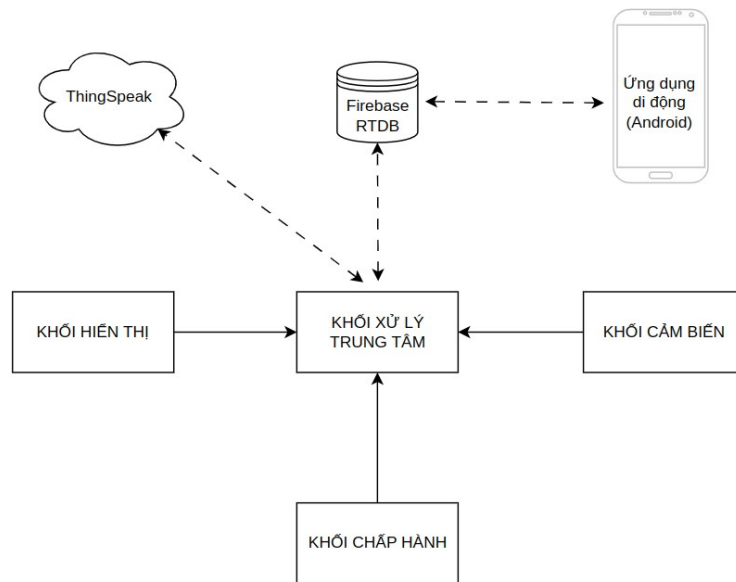
**HỆ THỐNG GIÁM SÁT NHIỆT ĐỘ, ĐỘ ẨM
VÀ
ĐIỀU KHIỂN THIẾT BỊ THÔNG QUA ĐIỆN THOẠI THÔNG MINH**

| | | |
|-------------------|--|--|
| Trường | ĐH SPKT Tp HCM | |
| Năm học | 2024-2025 | |
| Học phần | Cơ sở ứng dụng & IoT | |
| GVHD | Trương Quang Phúc | |
| Nhóm SV thực hiện | Nhóm MEOW: 1. Nguyễn Thanh Phú 2. Vũ Mai Liên 3. Trần Đức Tài 4. Trần Thủy Tiên 5. Nguyễn Hữu Trí | 22119211 22119194 22119226 22119238 22119244 |

MỤC TIÊU

Mục tiêu chính của hệ thống này là giám sát và kiểm soát môi trường một cách hiệu quả và tiện lợi, giúp người dùng luôn nắm bắt được tình hình nhiệt độ và độ ẩm trong không gian sống hay làm việc của mình. Bên cạnh đó, hệ thống còn hỗ trợ điều khiển các thiết bị điện tử xa, giúp tiết kiệm năng lượng và chi phí một cách tối ưu.

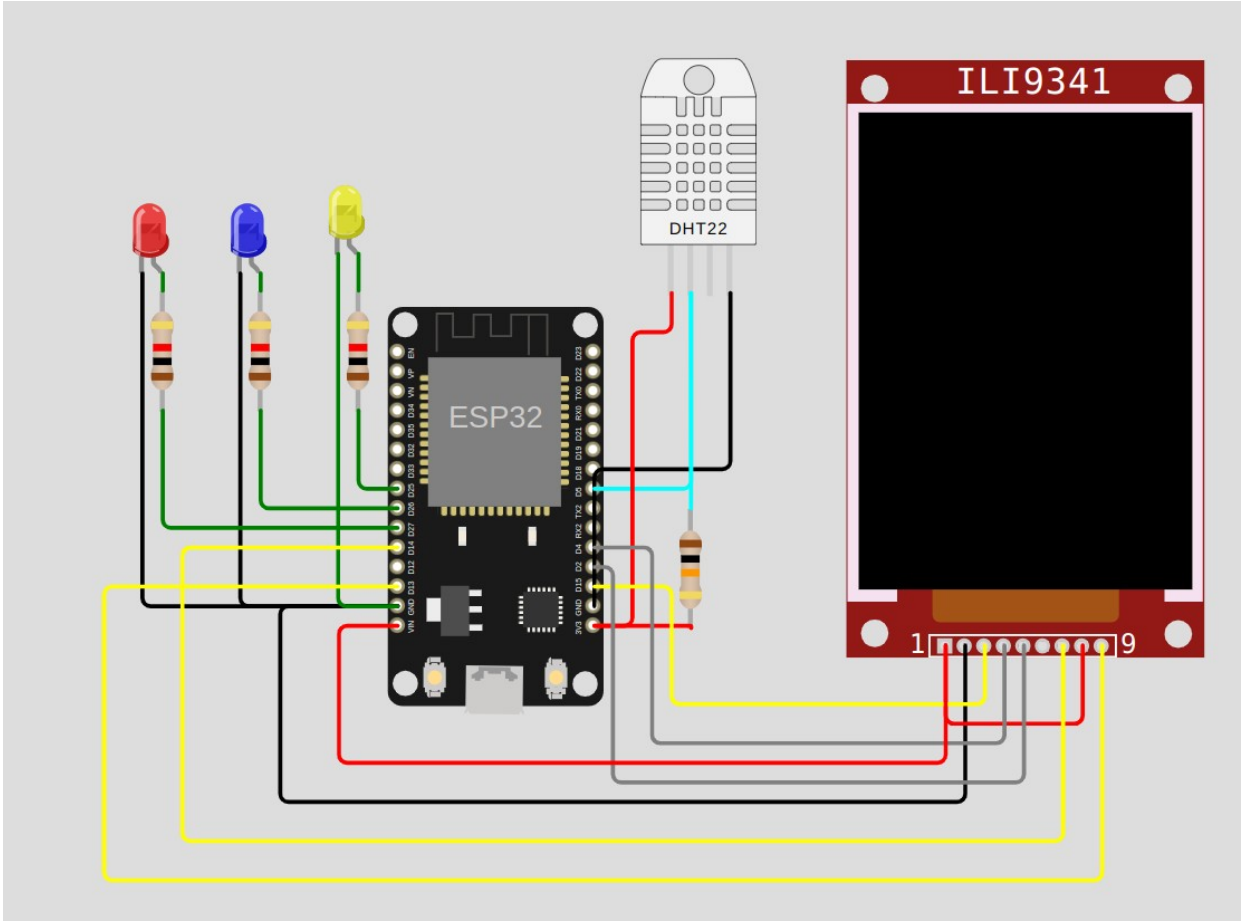
SƠ ĐỒ KHỐI HỆ THỐNG



Hệ thống gồm các khối:

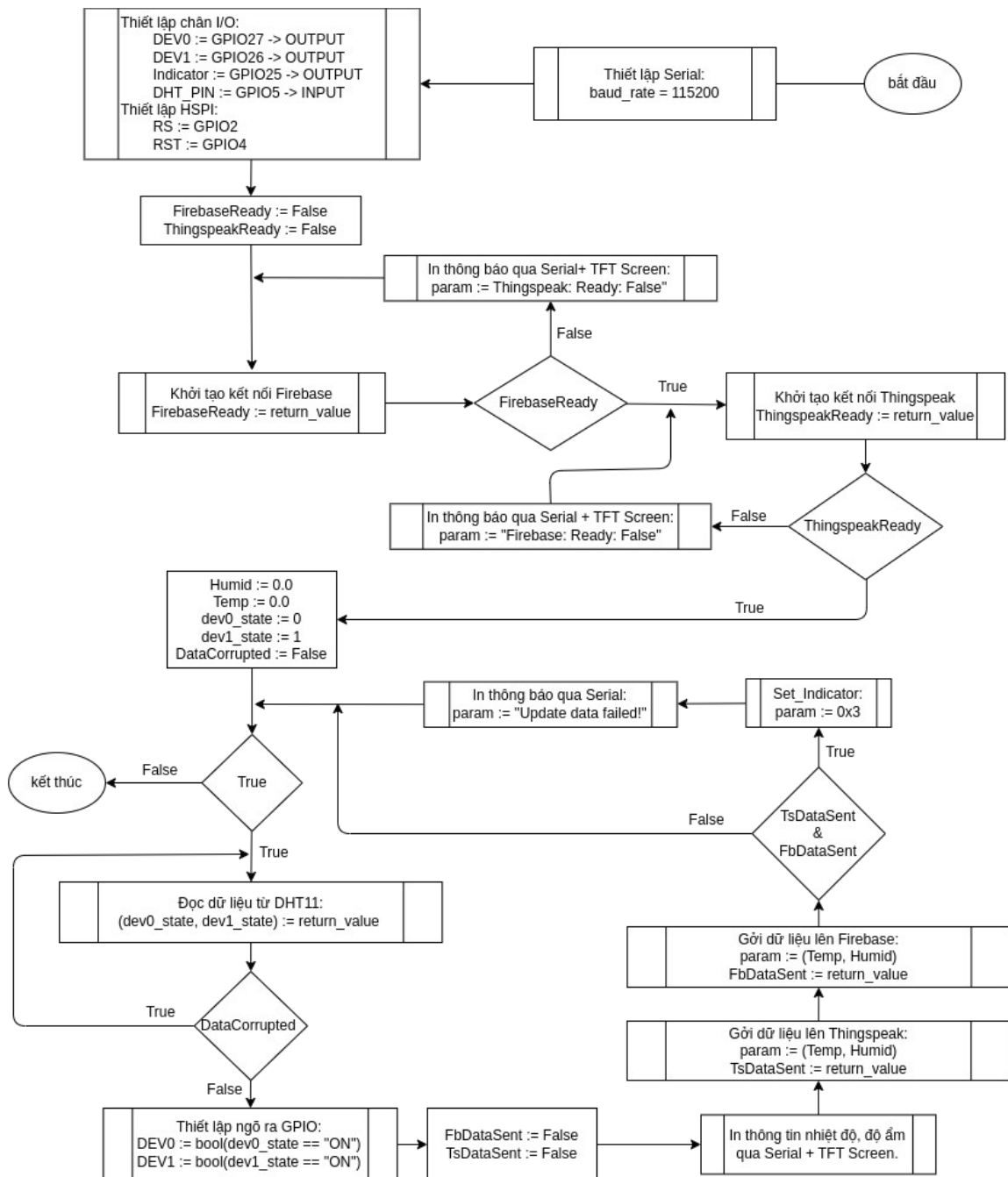
1. Khối xử lý trung tâm: Gồm 01 bảng mạch ESP32 để xử gởi/nhận dữ liệu đến cơ sở dữ liệu; điều khiển các thiết bị ở khối chấp hành, khối hiển thị; đọc giá trị ghi nhận được từ cảm biến.
2. Khối hiển thị: Gồm 01 màn hình TFT 2.2in, và 01 đèn LED. Màn hình hiển thị dữ liệu nhiệt độ và trạng thái hoạt động của hệ thống, đèn LED nhấp nháy mỗi chu kỳ hoạt động để thông báo thiết bị vẫn đang hoạt động.
3. Khối chấp hành: Gồm 02 đèn LED đại diện cho hai thiết bị được điều khiển.
4. Khối cảm biến: Gồm 01 cảm biến DHT11 để thu thập dữ liệu về nhiệt độ, độ ẩm từ môi trường.
5. Firebase RTDB: Cơ sở dữ liệu thời gian thực được lưu trữ trên đám mây, kết nối với ứng dụng di động Android.
6. Ứng dụng di động (Android): Ứng dụng này tương tác với Firebase RTDB, cung cấp khả năng điều khiển từ xa trên điện thoại di động (Android).

SƠ ĐỒ NGUYÊN LÝ

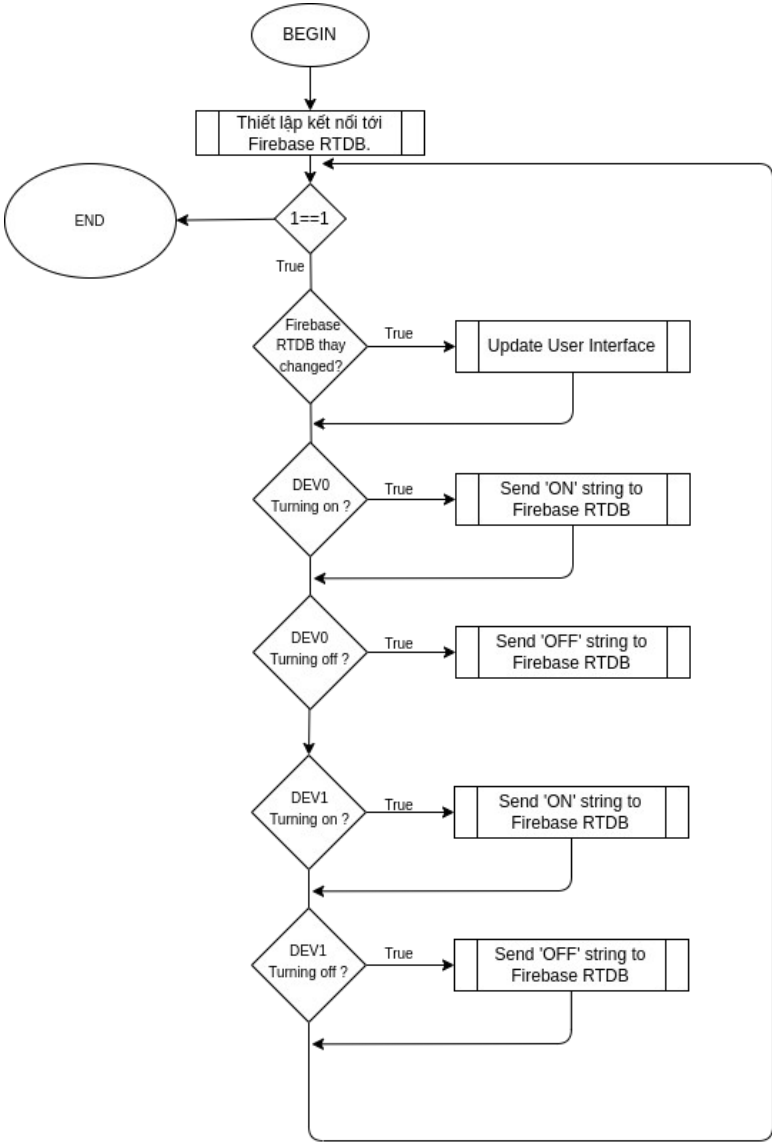


LƯU ĐỒ GIẢI THUẬT

Lưu đồ của hệ thống chính (không bao gồm lưu đồ giải thuật của ứng dụng Android)



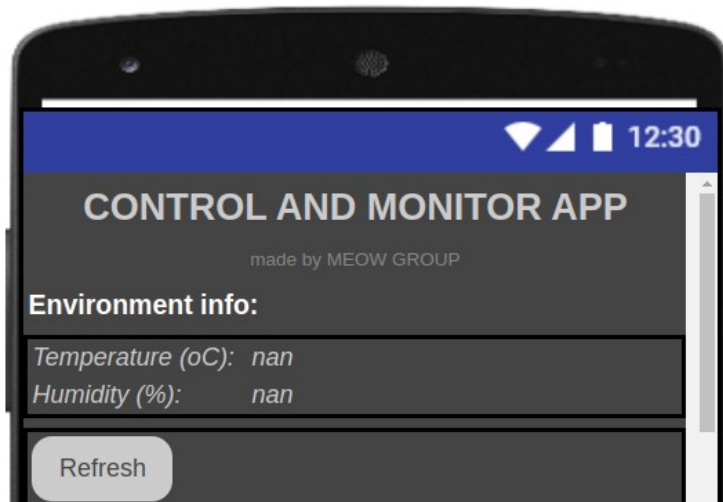
Lưu đồ giải thuật của ứng dụng Android:



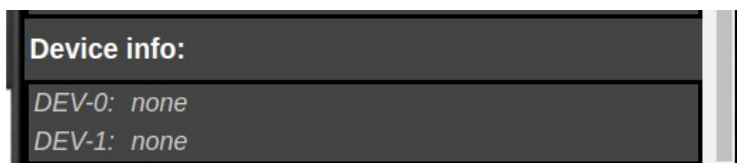
THIẾT KẾ GIAO DIỆN ỨNG DỤNG DI ĐỘNG

Ứng dụng di động (Android) được thiết kế trên nền tảng App Inventor 2, sau đó xuất bản thành tập tin *.APK và cài vào điện thoại Android.

1. Thông tin về môi trường:

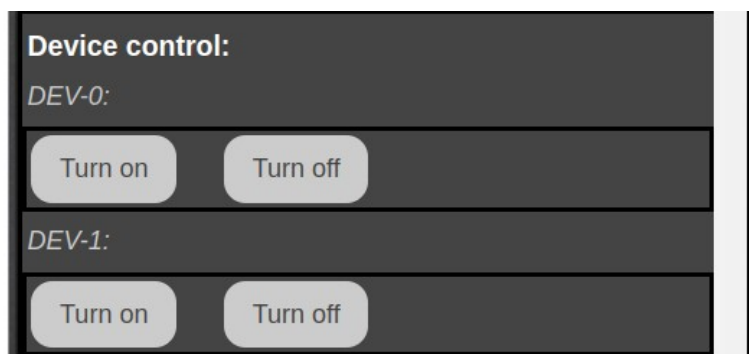


2. Thông tin về trạng thái thiết bị:



Giá trị mặc định của hai DEV-0 và DEV-1 là “none” khi chưa có dữ liệu nào được lấy về từ Firebase hoặc truy vấn không thành công.

3. Điều khiển thiết bị:



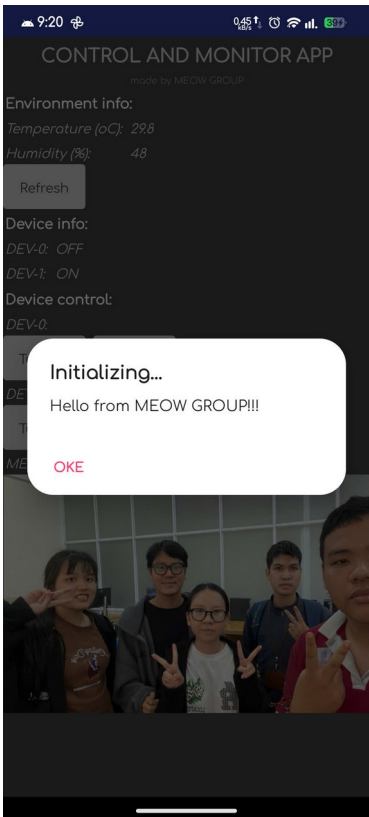
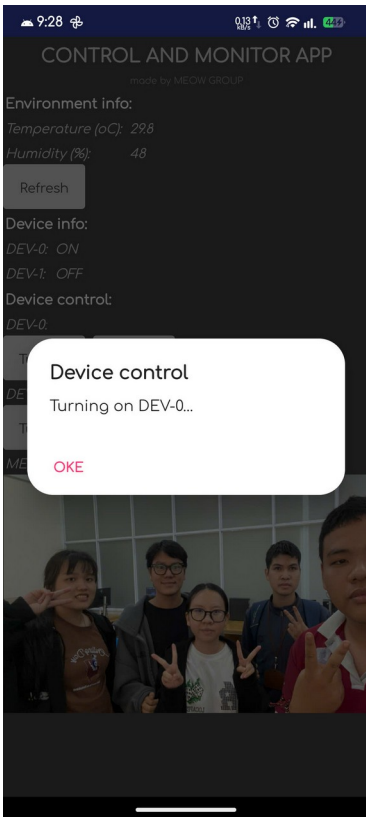
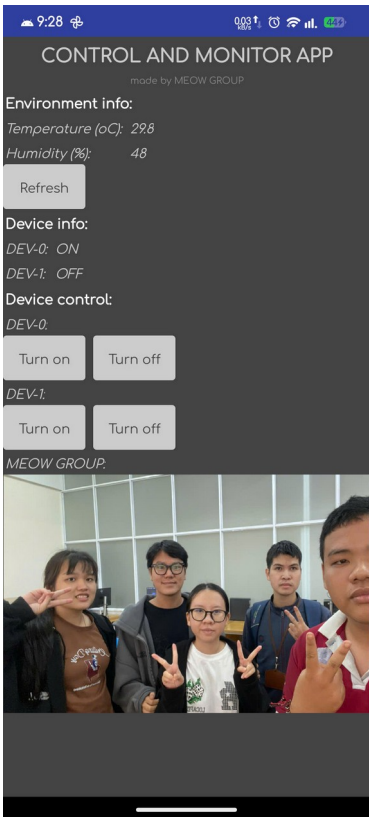
Gồm 04 nút nhấn cho việc bật/tắt hai thiết bị.

THIẾT KẾ GIAO DIỆN ỨNG MÀN HÌNH TFT

Giao diện trên màn hình TFT chỉ bao gồm các dòng thông báo trạng thái về nhiệt độ, độ ẩm theo thời gian thực; Hiển thị thông tin về quá trình gửi dữ liệu lên Firebase RTDB và ThingSpeak; Hiển thị thông tin quá trình truy vấn dữ liệu từ Firebase RTDB.

KẾT QUẢ

Ứng dụng di động (Android):

| | | |
|--|--|--|
|  |  |  |
| Giao diện lúc vừa khởi động ứng dụng. | Giao diện chính của ứng dụng khi nhấn nút “Turn on” của thiết bị DEV0. | Sau khi chọn “OKE” thì sẽ quay về giao diện chính của ứng dụng. |

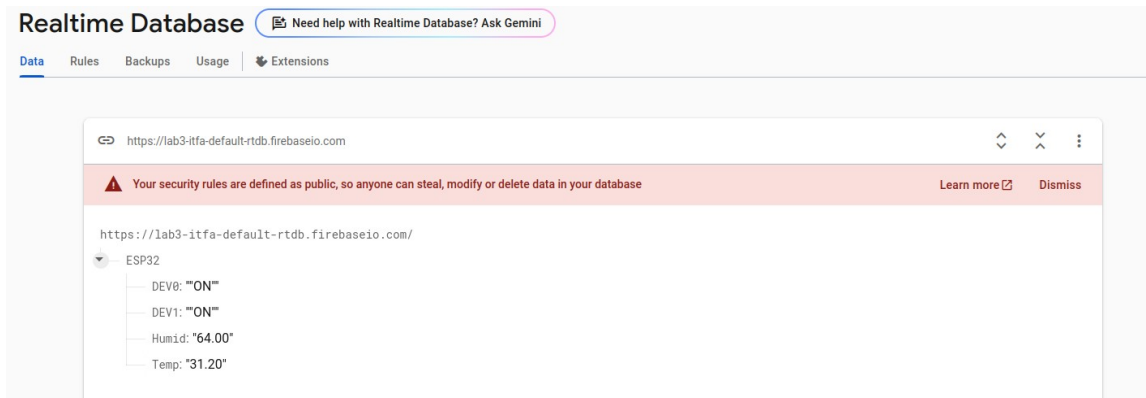
Màn hình TFT:

| | | |
|---|---|--|
|  |  |  |
| Khi vừa bật nguồn ESP32 | Hoàn thành một chu kỳ đọc giá trị cảm biến, gửi/ nhận dữ liệu lên các nền tảng số thành công. | Hoàn thành một chu kỳ đọc giá trị cảm biến, nhận dữ liệu lên các nền tảng số thành công. (Do giới hạn về số lần gửi dữ liệu của Thingspeak là 1 lần/15s) |

Chú thích:

- + “OK” : Thành công.
- + “:<” : Không thành công.
- + “x” : Bị bỏ qua, không thực hiện.

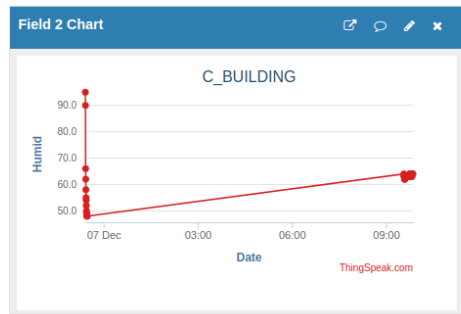
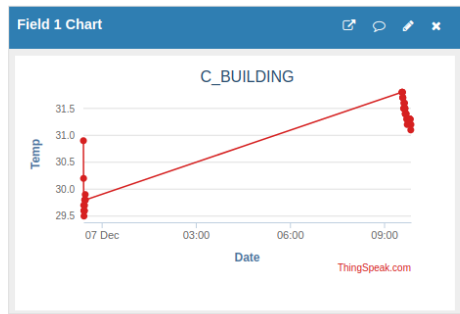
Firestore RTDB:



ThingSpeak:

Channel Stats

Created: 7 days ago
Last entry: less than a minute ago
Entries: 1132



Serial Monitor:

```
Executing task: platformio device monitor --environment esp32doit-devkit-v1 --port /dev/ttyUSB0

--- Terminal on /dev/ttyUSB0 | 115200 8-N-1
--- Available filters and text transformations: colorize, debug, default, direct, esp32_exception_decoder, hexlify, log2file, nocontrol, printable, send_on_enter, time
--- More details at https://bit.ly/pio-monitor-filters
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H
ets Jul 29 2019 12:21:46

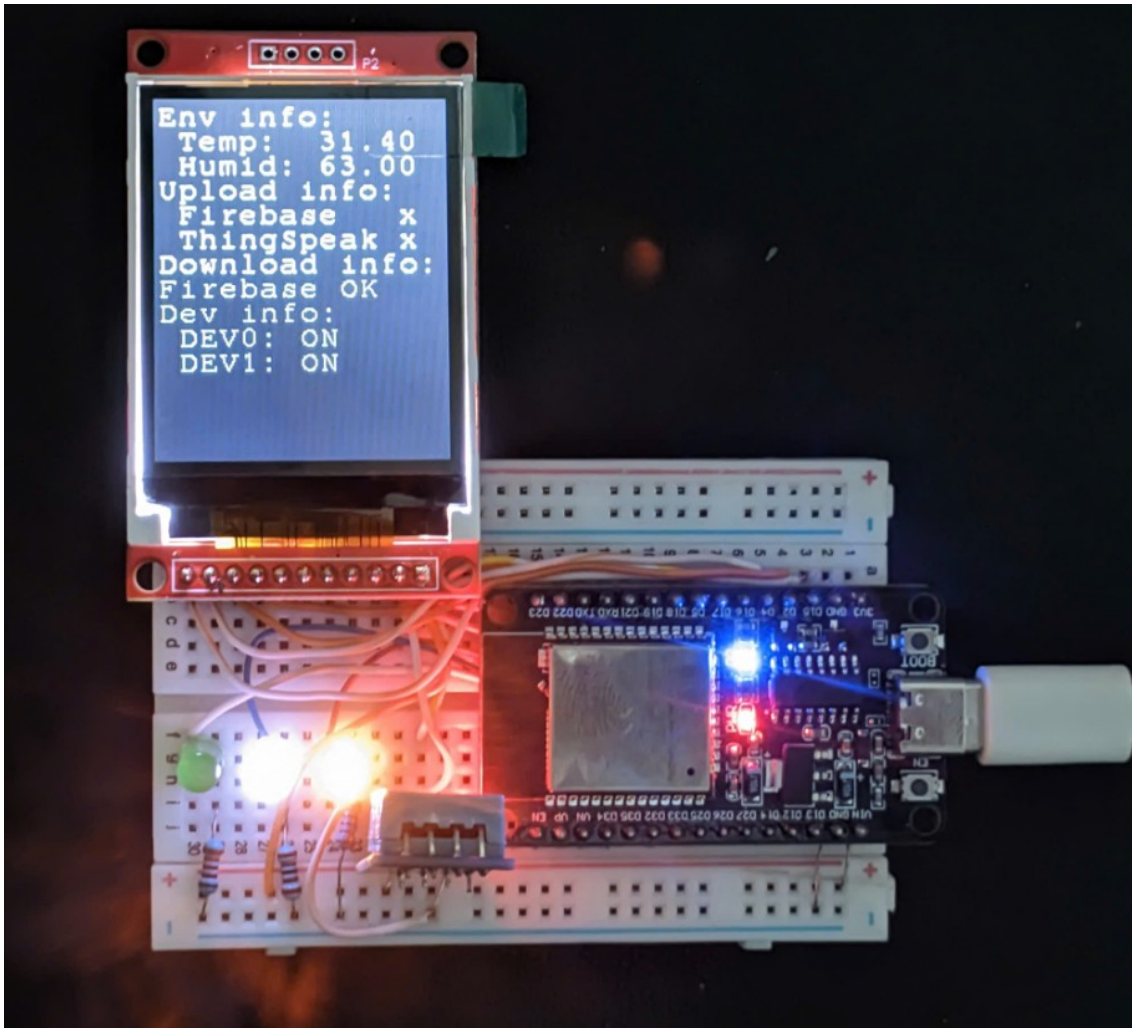
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:1184
load:0x40078000,len:13232
load:0x40080400,len:3028
entry 0x400805e4
ets Jul 29 2019 12:21:46

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:1184
load:0x40078000,len:13232
load:0x40080400,len:3028
entry 0x400805e4

Hello!
From MEOW GROUP!
No Wi-Fi connection!
Connecting to Wi-fi...
Connected to Wi-fi!
Connecting to Firebase...
Connected to Firebase!
Connecting to Thingspeak...
Connected to Thingspeak!
Temp :31.30
Humid :64.00
Temp :31.30
Humid :64.00
Temp :31.30
Humid :64.00

```

Các thiết bị được điều khiển, màn hình:



KẾT LUẬN

Nhận xét:

- + Hệ thống hoạt động đúng với thiết kế.
- + Thời gian thiết bị thay đổi trạng thái (ON - OFF) kể từ lúc nhấn nút <4 giây.
- + Chu kỳ cập nhật dữ liệu về nhiệt độ và độ ẩm ~15 giây.
- + Giao diện thiết bị di động (Android) và màn hình TFT còn sơ xài, chưa được chỉnh chu.

Hướng phát triển trong tương lai:

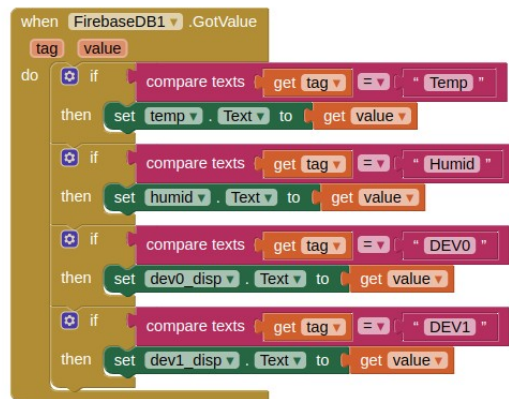
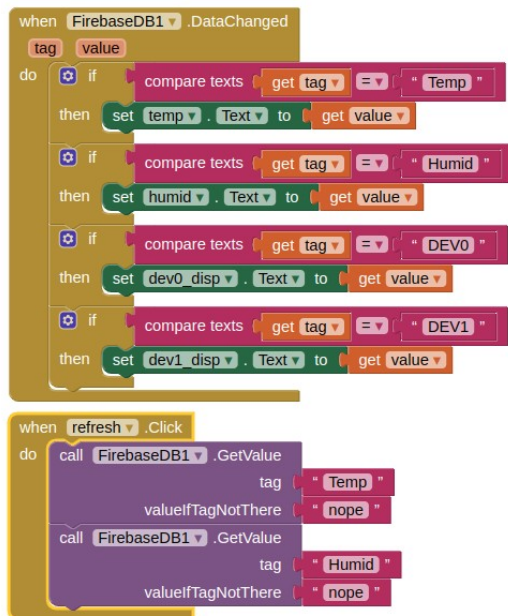
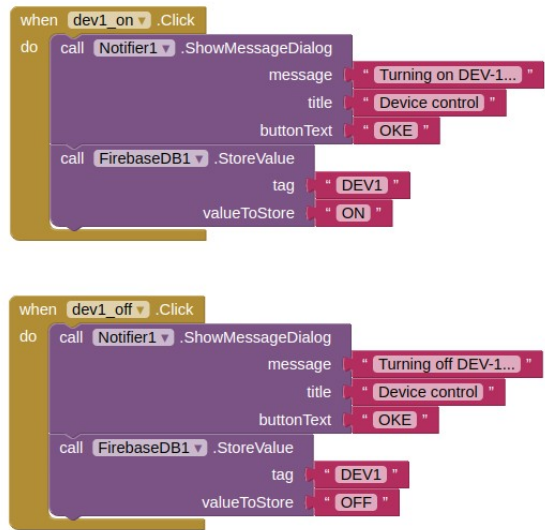
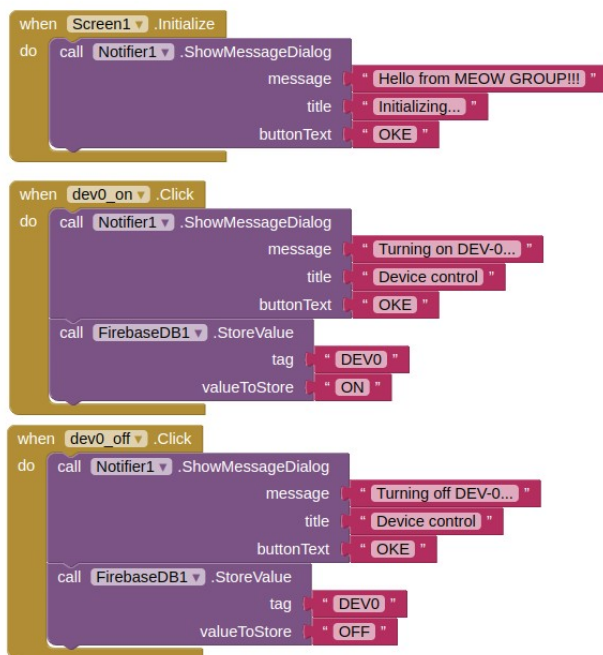
- + Thiết kế giao diện thiết bị di động (Android) và màn hình TFT chỉnh chu và trực quan hơn.
- + Mở rộng nhiều thiết bị có thể điều khiển và nhiều cảm biến giám sát.
- + Tích hợp các tính năng cảnh báo dựa trên các cài đặt đặt trước và các cảnh báo thông minh.

HÌNH ẢNH LÀM VIỆC NHÓM:



MÃ NGUỒN:

Ứng dụng Android:



ESP32-WROOM:

dht_def.h

```
#ifndef _DHT_DEF_H_
#define _DHT_DEF_H_

#include <Arduino.h>
#include "pin_def.h"

// Thư viện DHT dùng để đọc cảm biến
#include <DHT.h>
// Định nghĩa macro cho chân nhận dữ liệu từ cảm biến
#define DHT_DAT_PIN 5
// Định nghĩa macro cho kiểu DHT
#define DHTTYPE DHT11

// Tạo đối tượng DHT
DHT dht(DHT_DAT_PIN, DHTTYPE);

// Định nghĩa STRUCT sensor_data để đóng gói dữ liệu trả về
struct SENSOR_DATA{
float temp, humid;
};

// Định nghĩa hàm kiểm tra dữ liệu lỗi
bool is_data_corrupted(SENSOR_DATA data){
// Nếu dữ liệu is nan (Not a Number) thì trả về True
if( isnan(data.temp) || isnan(data.humid)){
set_indicator(0x1);
return true;
}
return false;
}

// Định nghĩa hàm đọc dữ liệu từ cảm biến DHT
// Trả về kiểu dữ liệu nhận được từ DHT được đóng gói
SENSOR_DATA dht_read(uint16_t delay_ms = 2000){
// Chờ một khoảng thời gian để cảm biến đo
delay(delay_ms);
// Nháy LED để báo hiệu quá trình đọc dữ liệu từ cảm biến DHT
set_indicator(0x0);
// Trả về kiểu dữ liệu được đóng gói dạng SENSOR_DATA
return {dht.readTemperature(), dht.readHumidity()};
}

#endif
```

pin_def.h

```
#ifndef _PIN_DEF_H_
#define _PIN_DEF_H_

#include <Arduino.h>

// Định nghĩa các macro cho các chân LED, DEV0, DEV1
#define INDICATOR_LED_PIN 25
#define DEV_0_PIN 26
#define DEV_1_PIN 27

// Định nghĩa STRUCT sensor_data để
// đóng gói dữ liệu trả về từ Firebase
struct DEV{
bool __0, __1;
};

// Hàm nháy LED theo mẫu để thông báo
// Codes:
// 0x0: Read data from sensor
// 0x1: DHT Received data corrupted
// 0x2: WiFi error
```

```

// 0x3: Database connection error
// 0x4: Speed direction error
void set_indicator(int err_code = 0){
// Blink led to noti error
switch (err_code){
case 0x0: err_code = 1; break;
case 0x1: err_code = 2; break;
case 0x2: err_code = 3; break;
case 0x3: err_code = 4; break;
case 0x4: err_code = 5; break;
default: err_code = 0;
}
while(err_code--){
digitalWrite(INDICATOR_LED_PIN, LOW); delay(50);
digitalWrite(INDICATOR_LED_PIN, HIGH); delay(200);
digitalWrite(INDICATOR_LED_PIN, LOW); delay(50);
}
}

// Hàm khởi tạo các chân GPIO cho LED, DEV0, DEV1
void gpio_init(){
pinMode(INDICATOR_LED_PIN, OUTPUT);
pinMode(DEV_0_PIN, OUTPUT);
pinMode(DEV_1_PIN, OUTPUT);
}

#endif

```

serial_def.h

```

#ifndef _SERIAL_DEF_H_
#define _SERIAL_DEF_H_
#include <Arduino.h>

// Hàm khởi tạo giao tiếp Serial
void serial_init(uint32_t baud_rate = 9600){
Serial.begin(baud_rate);
Serial.println("\n\nHello!\nFrom MEOW GROUP!");
}

// Định nghĩa phương thức gửi dữ liệu qua Serial
void msg2ser(String msg = ""){
Serial.println(msg);
}

// Nạp chồng hàm
void msg2ser(String msg0, String msg1){
Serial.print(msg0);
Serial.println(msg1);
}

// Nạp chồng hàm
void msg2ser(String msg0, String msg1, String msg2){
Serial.print(msg0);
Serial.print(msg1);
Serial.println(msg2);
}

#endif

```

TFT_176x220.h

```
#ifndef _LCD_176X220_H_
#define _LCD_176X220_H_

#pragma message(" \n\
Connections:\n\
LCD-TFT (slave) ESP32 (master)\n\
VCC VIN (5V)\n\
GND GND\n\
CLK HSPI-CLK (GPIO14)\n\
SDA HSPI-MOSI (GPIO13)\n\
RS (any GPIO)\n\
RST (any GPIO)\n\
CS HSPI-CS (GPIO15)\n\
")

#define drawline(Line, Text, Color) tft.drawGFXText(3, Line, (Text), Color)

enum lines_enum{
line0 = 15,
line1 = 30,
line2 = 45,
line3 = 60,
line4 = 75,
line5 = 90,
line6 = 105,
line7 = 120,
line8 = 135,
line9 = 150,
line10 = 165,
line11 = 180,
};

#include "SPI.h"
#include "TFT_22_ILI9225.h"
#include <../fonts/FreeSans12pt7b.h>
#include <../fonts/FreeSans24pt7b.h>
#include <../fonts/FreeMono9pt7b.h>

#define TFT_RST 4
#define TFT_RS 2
#define TFT_CS 15
#define TFT_LED 0
#define TFT_BRIGHTNESS 200
SPIClass hspi(HSPI);
TFT_22_ILI9225 tft = TFT_22_ILI9225(
TFT_RST,
TFT_RS,
TFT_CS,
TFT_LED,
TFT_BRIGHTNESS
);

int16_t w = 5, h = 9;

void tft_initial(){
hspi.begin();
tft.begin(hspi);
tft.setGFXFont(&FreeMono9pt7b);
tft.setBackgroundColor(0);
tft.clear();
}

#endif
```


firebase_thingspeak_def.h

```
#ifndef _FIREBASE_THINGSPEAK_DEF_H_
#define _FIREBASE_THINGSPEAK_DEF_H_

#include <Arduino.h>
#include "pin_def.h"
#include "dht_def.h"
#include "serial_def.h"

// Định nghĩa các macro cho SSID và PW của Wi-Fi
#define WIFI_SSID "rm.note.11"
#define WIFI_PWD "nGXXFUS@3204"
// Thư viện Wi-Fi
#include "WiFi.h"

// Định nghĩa các macro để kết nối tới Firebase
#define DATABASE_URL "https://lab3-itfa-default-rtdb.firebaseio.com/"
#define API_KEY "AIzaSyBMq4aT1Jx6nICW4yuYtTSIU9QSzIa1vmk"
#define EMAIL "ngxxfus@lab3-itfa.iam.gserviceaccount.com"
#define PASSWORD "nGXXFUS@3204"

//Thư viện dùng để kết nối tới Firebase
#include "FirebaseESP32.h"

// Tạo các đối tượng để thiết lập, xác thực, là truy xuất dữ liệu
// từ Firebase
FirebaseData firebaseData;
FirebaseAuth auth;
FirebaseConfig config;

// Định nghĩa các macro để kết nối với Thingspeak
#define CHANNEL_ID 2768618
#define WRITE_API_KEY "VOICB5QB0T4RPFY6"
#define READ_API_KEY "MPFD1MXPB0B5051W"
// Tạo đối tượng kiểu WifiClient để gửi nhận dữ liệu đến ThingSpeak
WifiClient client;
// Thư viện kết nối tới ThingSpeak
#include "ThingSpeak.h"

//----- Firebase -----//
// Định nghĩa hàm kiểm tra trạng thái kết nối Wi-Fi
bool is_connected_to_wifi(){
    if(WiFi.status() != WL_CONNECTED){
        // Nếu không có kết nối Wi-Fi thì thông báo lỗi
        set_indicator(0x2);
        // và in ra serial lỗi
        msg2ser("No Wi-Fi connection!");
        return false;
    }
    return true;
}

// Định nghĩa hàm kết nối Wi-Fi
void wifi_init(){
    if(is_connected_to_wifi()) return;
    // Thiết lập Wi-Fi ở chế độ tiêu chuẩn
    WiFi.mode(WIFI_STA);
    // Kết nối Wi-Fi với macro đã định nghĩa từ trước
    WiFi.begin(WIFI_SSID, WIFI_PWD);
    do{
        msg2ser("Connecting to Wi-fi... ");
        delay(5000);
    }while(!is_connected_to_wifi());
    msg2ser("Connected to Wi-fi!");
}
```



```

// Định nghĩa hàm thiết lập kết nối với Firebase
void firebase_init() {
  msg2ser("Connecting to Firebase ... ");
  config.api_key = API_KEY;
  config.database_url = DATABASE_URL;
  auth.user.email = EMAIL;
  auth.user.password = PASSWORD;
  // Thực hiện việc kết nối đến khi thành công
  do Firebase.begin(&config, &auth);
  while(!Firebase.ready());
  // Thông báo đã kết nối Wi-Fi thành công
  msg2ser("Connected to Firebase!");
  // Thiết lập tự động kết nối lại Wi-Fi
  Firebase.reconnectWiFi(true);
}

// Hàm gửi dữ liệu đến Firebase
bool firebase_upload(SENSOR_DATA data){
  // Nếu không có kết nối Wi-Fi, trả về 0 - Không gửi được dữ liệu
  if(is_connected_to_wifi() == false) return false;
  // Tạo biến STATUS để lưu trạng thái gửi dữ liệu
  uint8_t STATUS = 0;
  // Nếu gửi dữ liệu thành công, phương thức setString() của Firebase sẽ trả
  // về true, ngược lại false
  STATUS += Firebase.setString(firebaseData, "ESP32/Temp", String(data.temp)) * 0x1;
  STATUS += Firebase.setString(firebaseData, "ESP32/Humid", String(data.humid)) * 0x2;
  // Thông báo mã lỗi nếu có
  if(STATUS&0x1 == 0) msg2ser("Thingspeak: Upload Temp feiled!");
  if(STATUS&0x2 == 0) msg2ser("Thingspeak: Upload Humud feiled!");
  return bool(STATUS==3);
}

bool firebase_download(bool* dev0, bool* dev1){
  uint8_t success = 0;
  static char strON[] = {'\\', '\'', 'O', 'N', '\\', '\'', '\0'};
  if(Firebase.getString(firebaseData, "ESP32/DEV0")){
    if(firebaseData.dataType() == "string"){
      success += 1;
      firebaseData.stringData() = String(strON)?
      *dev0 = 1: *dev0 = 0;
    }else
      msg2ser("Firebase: Get DEV0: Error type!");
    }else
      msg2ser("Firebase: Get DEV0: Failed!");
    if(Firebase.getString(firebaseData, "ESP32/DEV1")){
      if(firebaseData.dataType() == "string"){
        success += 2;
        firebaseData.stringData() = String(strON)?
        *dev1 = 1:*dev1 = 0;
      }
      else
        msg2ser("Firebase: Get DEV1: Error type!");
      }else
        msg2ser("Firebase: Get DEV1: Failed!");
    return (success == 3);
  }

//----- Thingspeak -----//
void thingspeak_init(){
  msg2ser("Connecting to Thingspeak ... ");
  ThingSpeak.begin(client);
  ThingSpeak.readLongField(CHANNEL_ID, 1, READ_API_KEY);
  ThingSpeak.getLastReadStatus();
  while(ThingSpeak.getLastReadStatus() != 200){
    msg2ser("Connecting to Thingspeak ... ");
    delay(2000);
  }
  msg2ser("Connected to Thingspeak!");
}

bool thingspeak_upload(SENSOR_DATA data){
  bool STATUS = true;
  ThingSpeak.setField(1, data.temp);
  ThingSpeak.setField(2, data.humid);
  auto rcv_code = ThingSpeak.writeFields(CHANNEL_ID, WRITE_API_KEY);
  if(bool(rcv_code != 200)){
    msg2ser("Thingspeak: Upload: Failed! ", "CODE: ", String(rcv_code));
    STATUS = false;
  }
}

```

```

}
return STATUS;
}
#endif

```

main.cpp

```

#include "Arduino.h"
#include "pin_def.h"
#include "serial_def.h"
#include "dht_def.h"
#include "firebase_thingspeak_def.h"
#include "TFT_176x220.h"

unsigned long lastcheck = 0;

void setup(){
  tft_initial();
  drawline(line0, String("Connections: "), 0x46f0);
  serial_init(115200);
  drawline(line1, String(" Serial: OK"), 0x46f0);
  gpio_init();
  drawline(line2, String(" GPIO: OK"), 0x46f0);
  wifi_init();
  drawline(line3, String(" Wi-Fi: OK"), 0x46f0);
  firebase_init();
  drawline(line4, String(" Firebase: OK"), 0x46f0);
  thingspeak_init();
  drawline(line5, String(" ThingSpeak:OK"), 0x46f0);
}

void loop(){
  auto data = dht_read(2000);
  msg2ser("Temp :", (String)data.temp);
  msg2ser("Humid :", (String)data.humid);
  tft.clear();
  if(is_data_corrupted(data)){
    set_indicator(0x1);
    return;
  }
  drawline(line0, String("Env info:"), 0x46f0);
  drawline(line1, String(" Temp: ") + String(data.temp), 0x46f0);
  drawline(line2, String(" Humid: ") + String(data.humid), 0x46f0);

  drawline(line3, String("Upload info:"), 0x46f0);
  if( millis() - lastcheck > 15000UL){
    lastcheck = millis();
    firebase_upload(data)?
    drawline(line4, String(" Firebase OK"), 0x46f0):
    drawline(line4, String(" Firebase :<"), 0x46f0);

    thingspeak_upload(data)?
    drawline(line5, String(" ThingSpeak OK"), 0x46f0):
    drawline(line5, String(" ThingSpeak :<"), 0x46f0);
  }else{
    drawline(line4, String(" Firebase x"), 0x46f0);
    drawline(line5, String(" ThingSpeak x"), 0x46f0);
  }

  drawline(line6, String("Download info:"), 0x46f0);
  bool dev0_state, dev1_state;
  firebase_download(&dev0_state, &dev1_state)?
  drawline(line7, String("Firebase OK"), 0x46f0):
  drawline(line7, String("Firebase :<"), 0x46f0);

  drawline(line8, String("Dev info:"), 0x46f0);
  drawline(line9, String(" DEV0: ") + String(dev0_state?"ON":"OFF"), 0x46f0);
  drawline(line10, String(" DEV1: ") + String(dev1_state?"ON":"OFF"), 0x46f0);

  digitalWrite(DEV_0_PIN, dev0_state);
  digitalWrite(DEV_1_PIN, dev1_state);
}

```

platformio.ini

```
[env:esp32doit-devkit-v1]
platform = espressif32
board = esp32doit-devkit-v1
framework = arduino
lib_deps =
adafruit/DHT sensor library@^1.4.6
adafruit/Adafruit Unified Sensor@^1.1.14
mobizt/Firebase ESP32 Client@^4.4.14
mathworks/ThingSpeak@^2.0.0
SPI
nkawu/TFT 22 ILI9225@^1.4.5

monitor_speed = 115200
upload_speed = 921600
```