**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT**

**THÀNH PHỐ HỒ CHÍ MINH**

**🙟 🕮 🙝**

**HCMUTE**

***Báo cáo môn học:***

**Internet of Things**

**ĐỀ TÀI: HỆ THỐNG BÁO ĐỘNG AN NINH DỰA TRÊN IOT**

**GVHD: ThS. Đinh Công Đoan**

**SVTH: Nguyễn Đức Mạnh 18110155**

**Nguyễn Minh Tuấn 18110228**

**Nguyễn Minh Thuận 18110208**

***Thành phố Hồ Chí Minh****,* ***Tháng*** *5* ***Năm*** *2021*

# MỤC LỤC

[MỤC LỤC 1](#_Toc73226901)

[CHƯƠNG 1: TỔNG QUAN 2](#_Toc73226902)

[1.1. Giới thiệu về đề tài: 2](#_Toc73226903)

[1.2. Phân tích các vấn đề: 2](#_Toc73226904)

[1.3. Mục tiêu và công cụ nghiên cứu: 2](#_Toc73226905)

[1.4. Kết cấu của đề tài: 3](#_Toc73226906)

[CHƯƠNG 2: PHÂN TÍCH ĐỀ TÀI 4](#_Toc73226907)

[2.1. Yêu cầu người dùng: 4](#_Toc73226908)

[2.2. Yêu cầu kĩ thuật: 4](#_Toc73226909)

[2.2.1. Chức năng hệ thống: 4](#_Toc73226910)

[2.2.2. Đặc tả kĩ thuật: 4](#_Toc73226911)

[2.3. Lựa chọn linh kiện: 4](#_Toc73226912)

[2.3.1. Module cảm biến ngọn lửa: 4](#_Toc73226913)

[2.3.2. Module Wi-Fi ESP8266: 5](#_Toc73226914)

[2.4. Sơ đồ khối: 6](#_Toc73226915)

[2.5. Nguyên tắc hoạt động của hệ thống: 7](#_Toc73226916)

[CHƯƠNG 3: THỰC THI THIẾT KẾ 8](#_Toc73226917)

[3.1. Thiết kế phần cứng: 8](#_Toc73226918)

[3.2. Thiết kế phần mềm: 8](#_Toc73226919)

[3.2.1. Lưu đồ thuật toán: 8](#_Toc73226920)

[3.2.2. Server Firebase và cách kết nối phần cứng: 9](#_Toc73226921)

[3.2.3. Tạo ứng dụng kết nối bằng Android Studio: 10](#_Toc73226922)

[CHƯƠNG 4: kẾT QUẢ VÀ NHẬN XÉT 12](#_Toc73226923)

[4.1. Kết quả thu được: 12](#_Toc73226924)

[4.1.1. Kết quả lý thuyết: 12](#_Toc73226925)

[4.1.2. Kết quả thực tế: 12](#_Toc73226926)

[4.2. Nhận xét 13](#_Toc73226927)

[KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 14](#_Toc73226928)

[1. Kết luận: 14](#_Toc73226929)

[2. Hướng phát triển đề tài: 14](#_Toc73226930)

[PHỤ LỤC 15](#_Toc73226931)

[Code phần cứng: 15](#_Toc73226932)

[Code phần mềm Android: 17](#_Toc73226933)

[- File MainActivity.java: 17](#_Toc73226934)

[- File activity\_main.xml: 22](#_Toc73226935)

# CHƯƠNG 1: TỔNG QUAN

* 1. **Giới thiệu về đề tài:**

Trong bối cảnh nền kinh tế thương mai, hàng hoá, dịch vụ ngày càng phát triển, mở rộng. Kết hợp với công nghiệp 4.0, ngành Logistics từ đó cũng phát triển mạnh mẽ, nhiều nhà máy sản xuất, lưu trữ được thành lập với quy mô vô cùng rộng lớn. Đi kèm với sự phát triển nhanh, rộng cũng tồn tại vấn đề an toàn lao động đặc biệt là rủi ro về hoả hoạn vô cùng lớn.

***Hỏa hoạn:*** Do nhiều yếu tố gây ra như chập điện, thời tiết nắng nóng dễ gây bắt lửa các vật liệu dễ cháy như: Vải, bông, mút, xốp, … Sự cố sẽ càng nghiêm trọng hơn nếu không được cảnh báo kịp thời gây thiệt hại lớn đến con người và tài sản.

→ Do các vấn đề xảy ra nêu trên chưa có cách giải quyết triệt để, nên nhóm thực hiện quyết định sẽ thiết kế “**Hệ thống phát hiện cháy thông qua ứng dụng điện thoại**”.

* 1. **Phân tích các vấn đề:**

Tại các nhà máy sản xuất, lưu trữ hiện nay, rất dễ rò rỉ các chất dễ gây cháy nổ như: Khí gas, xăng, dầu, hóa chất,.. hay do chập điện. Khi sự cố xảy ra cần có hệ thống cảnh báo cho người liên quan biết để kịp thời ngăn chặn và giảm tối đa các sự cố nhằm giảm thiểu thiệt hại.

→ Thiết kế “**Hệ thống phát hiện cháy thông qua ứng dụng điện thoại**” sẽ là giải pháp khắc phục cho các vấn đề nêu trên xảy ra với hệ thống nhà kho.

* 1. **Mục tiêu và công cụ nghiên cứu:**

**Mục tiêu đề tài:**

Giới thiệu về công nghệ Internet of Things (IoT): Kiến trúc của IoT, các công nghệ truyền thông trong IoT.

Cảnh báo qua ứng dụng điện thoại cho người dùng biết khi xảy ra hỏa hoạn.

Lữu trữ các sự kiện xảy ra để dễ dàng quản lý.

**Công cụ nghiên cứu:**

Arduino IDE: Phần mềm viết các chương trình cho vi điều khiển.

Tìm hiểu về thành phần cấu tạo của hệ thống thông qua các phương tiện như: Google, Arduino.

Nghiên cứu các tài liệu, datasheet liên quan đến các vi điều khiển, các cảm biến mà hệ thống sử dụng.

Fire base: Tạo cơ sở dữ liệu lưu trữ thông tin.

Androird Studio: Tạo ứng dụng Android để giao tiếp với vi điều khiển.

Ngoài ra còn sử dụng các thư viện hỗ trợ ngoài khác.

* 1. **Kết cấu của đề tài:**

***Chương 1:*** Tổng quan.

***Chương 2***: Phân tích đề tài.

***Chương 3:*** Thực thi thiết kế.

***Chương 4:*** Kết quả và nhận xét.

# CHƯƠNG 2: PHÂN TÍCH ĐỀ TÀI

* 1. **Yêu cầu người dùng:**

Hệ thống thông báo khi phát hiện cháy nổ cho người dùng.

Thông báo cho người dùng nào đang ở gần nơi xảy ra sự cố.

Xác định vị trí chính xác nơi xảy ra sự cố để thông báo cho người dùng.

Thông báo đến cho những người gần nhất để kịp thời xử lí.

* 1. **Yêu cầu kĩ thuật:** 
     1. Chức năng hệ thống:
* Thu thập dữ liệu từ các cảm biến đặt ở mỗi khu vực hoặc những nơi dễ xảy ra sự cố.
* Xử lý dữ liệu, tính toán đưa ra cảnh báo và vị trí phù hợp.
* Thống kê các sự kiện xảy ra theo: Thời gian, loại cảnh báo, vị trí xảy ra.
* Thực thi cảnh báo đến điện thoại người dùng khi xảy ra sự cố.
* Thu thập vị trí người dùng, để gửi cảnh báo đến người dùng ở gần nơi xảy ra sự cố.
  + 1. Đặc tả kĩ thuật:

Dùng vi điều khiển ESP8266 để giao tiếp với cảm biến để lấy dữ liệu và gửi dữ liệu lên Server.

Vi điều khiển sẽ xác nhận thông báo từ cảm biến tại vị trí nào và từ đó gửi cảnh báo thông qua Server cho người dùng một cách chính xác.

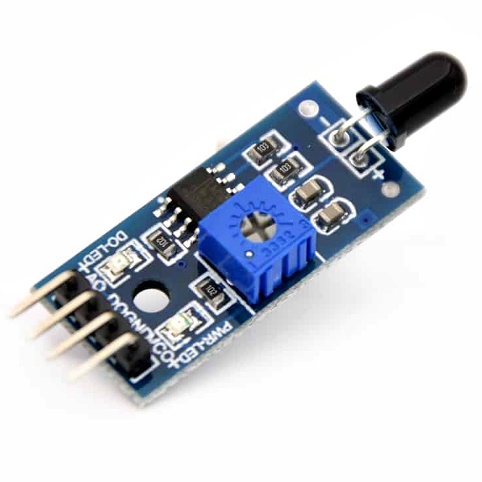
Sử dụng cảm biến hồng ngoại YG1006 tốc độ đáp ứng nhanh, độ nhạy cao, tầm hoạt động 80cm và góc quét 60 độ

Tạo cơ sở dữ liệu trên Fire Base để lưu trữ các thông tin từ vi điều khiển đưa lên: Thời gian cảnh báo, loại cảnh báo, vị trí xảy ra cảnh báo.

* 1. **Lựa chọn linh kiện:**
     1. Module cảm biến ngọn lửa:

Module cảm biến ngọn lửa dùng led hồng ngoại YG1006 có các thông số sau:

* Điện áp hoạt động: 3.3-5 VDC.
* Dòng điện tiêu thụ 15mA
* Tín hiệu ra: Digital 3.3 – 5VDC tùy nguồn cấp hoặc Analog.
* Khoảng cách: 80cm
* Góc quét: 60 độ
* Kích thước: 3.2 x 1.4 cm



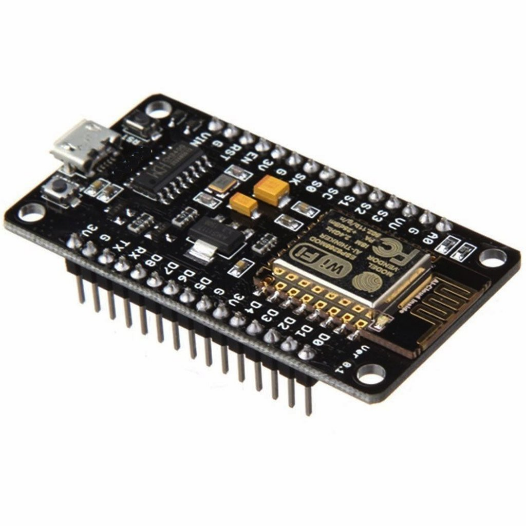
Hình 3.1: Cảm biến nhiệt độ DS18B20.

Chức năng chân:

* VCC: 3.3V ~ 5.3V
* GND : Power supply ground
* AOUT (AO): Analog output
* DOUT (DO): Digital output
  + 1. Module Wi-Fi ESP8266:

NodeMCU V1.0 được phát triển dựa trên Chip WiFi ESP8266EX bên trong Module ESP-12E dễ dàng kết nối WiFi với một vài thao tác.Board còn tích hợp IC CP2102, giúp dễ dàng giao tiếp với máy tính thông qua Micro USB để thao tác với board. Và có sẳn nút nhấn, led để tiện qua quá trình học, nghiên cứu.

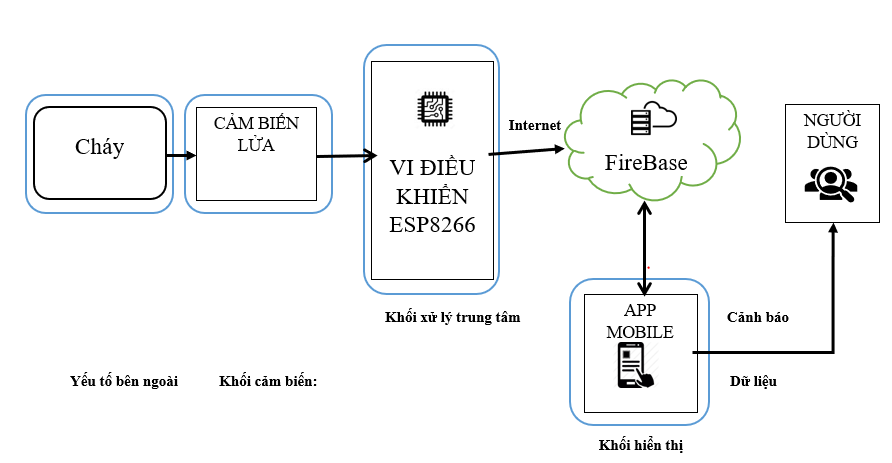
* IC chính: ESP8266 Wifi SoC.
* Phiên bản firmware: Node MCU.
* Chip nạp và giao tiếp UART: CP2102
* Cấp nguồn: 5VDC MicroUSB hoặc Vin.
* GIPO giao tiếp mức: 3.3VDC
* Tích hợp: Led báo trạng thái, nút Reset, Flash.
* Tương thích : Trình biên dịch Arduino.
* Kích thước: 25 x 50 mm



Hình 3.2: Module ESP-12F.

Module được dùng nhiều trong các ứng dụng thu thập dữ liệu và điều khiển thiết bị qua WiFi và phát tiến hiệu qua đèn hiệu.

* 1. **Sơ đồ khối:**



Hình 3.3: Sơ đồ khối toàn hệ thống,

**Khối cảm biến:**

Sử dụng cảm biến ngọn lửa để thu thập dữ liệu từ môi trường bên ngoài. Từ đó gửi dữ liệu liên tục đến vi điều khiển,

**Khối xử lý trung tâm:**

Sử dụng ESP8266 để xử lý dữ liệu từ khối cảm biến gửi về. Sau đó, ESP8266 sẽ gửi dữ liệu lên Server liên tục thông qua mạng Internet.

**Khối hiển thị và cảnh báo:**

Dữ liệu từ khối xử lý trung tâm được gửi lên server thông qua internet, từ đây có thể sử dụng bất kì một máy tính hay các thiết bị có khả năng truy cập vào mạng để truy cập vào server, mọi dữ liệu đều được hiển thị ở đó. Khi nhiệt độ đạt ngưỡng hệ thống sẽ cảnh báo về cho người dùng thông qua một ứng dụng.

* 1. **Nguyên tắc hoạt động của hệ thống:**

Khi ta cấp nguồn cho hệ thống, khối cảm biến với những module cảm biến: ngọn lửa với hồng ngoại YG1006 có chức năng thu thập nhiệt độ và phát hiện chuyển động của người gửi dữ liệu về cho khối xử lý. Khối xử lý với module ESP8266 sẽ gửi dữ liệu lên server thông qua đường truyền Internet. Server sẽ tiếp nhận dữ liệu đồng thời gửi dữ liệu đến điện thoại để hiển thị, nếu dữ liệu nhận được vượt ngưỡng cho phép mà đã được cấu hình từ trước hoặc phát hiện có sự xâm nhâp từ con người thì người dùng sẽ nhận được cảnh báo.

Hệ thống còn có thể sử dụng GPS của smartphone để tính toán khoảng cách đến nơi xảy ra sự cố để thông báo tới người dùng đang ở gần.

# CHƯƠNG 3: THỰC THI THIẾT KẾ

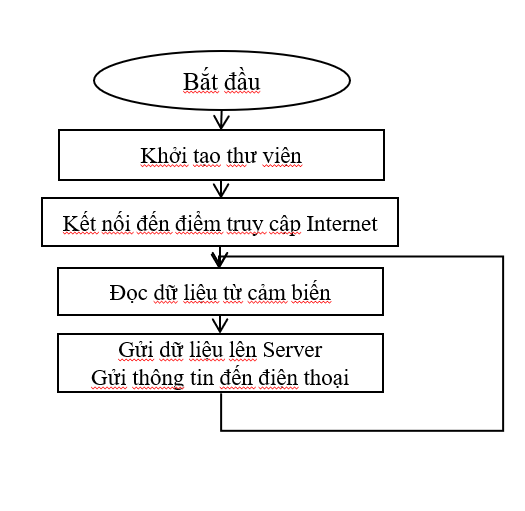
## 3.1. Thiết kế phần cứng:

Trong sơ đồ nguyên lý này gồm 2 linh kiện đó là ESP8266, cảm biến lửa YG1006. Vi điều khiển sẽ nhận dữ liệu từ cảm biến thông qua chân A0. Cảm biến một chân sẽ nối lên nguồn 3V, một chân sẽ nối GND, chân còn lại là chân truyền tín hiệu về cho vi điều khiển ESP8266.

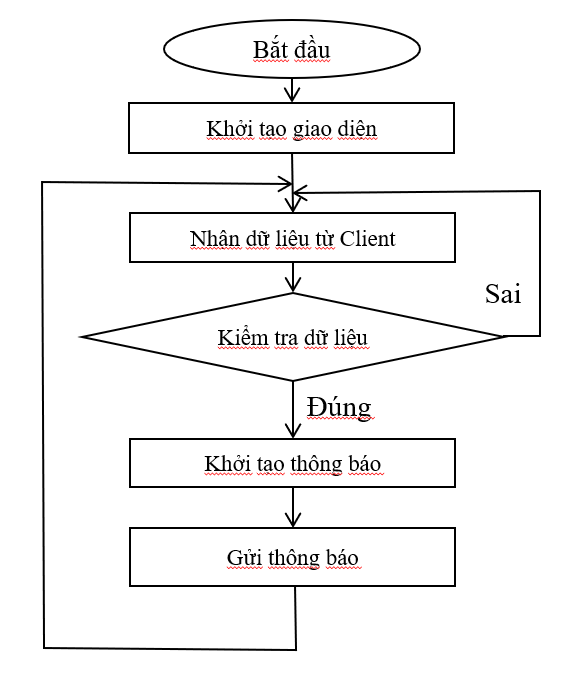
## **3.2. Thiết kế phần mềm:**

(Code chương trình ở phần phụ lục)

### 3.2.1. Lưu đồ thuật toán:



Giải thích lưu đồ phần cứng: Hệ thống khi bắt đầu hoạt động sẽ khởi tạo thư viện cho ESP8266, và ESP8266 thông qua kit WIFI kết nối tới điểm truy cập Internet. Cảm biến sẽ thu thập dữ liệu từ môi trường gửi dữ liệu về ESP8266 và để ESP8266 gửi dữ liệu lên server và gửi cảnh báo đến điện thoại nếu có sự cố xảy ra.



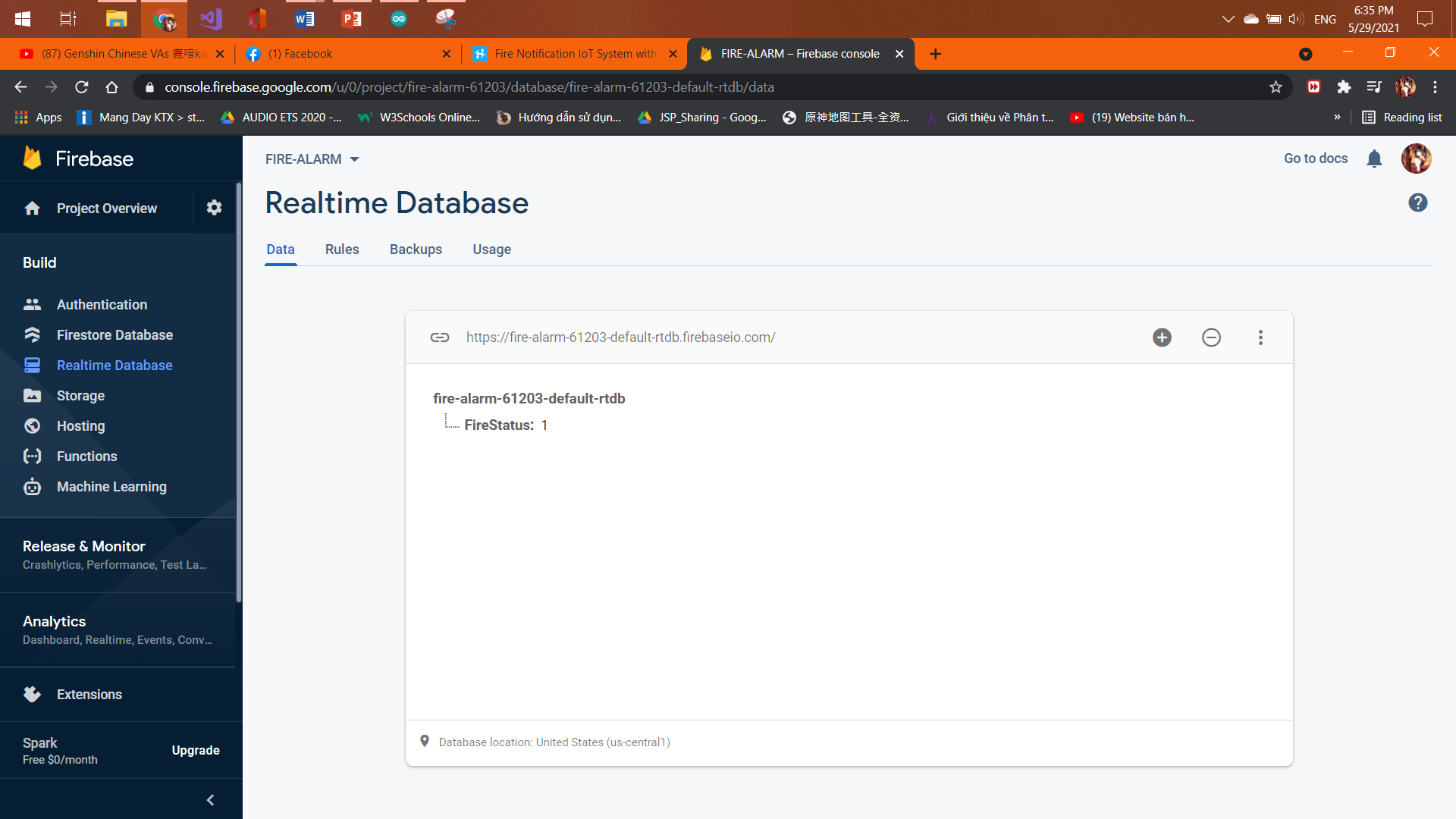
Giải thích lưu đồ phần mềm: Hệ thống khi bắt đầu hoạt động sẽ khởi tạo giao diện làm việc sau đó nhận dữ liệu từ client gửi lên server. Tiến hành hiển thị dữ liệu và kiểm tra xem có đạt điều kiện đã đặt ra không, nếu có sẽ tiến hành khởi tạo thông báo và gửi đến người dùng và quay lại nhận dữ liệu nếu có thay đổi, còn không thì sẽ quay lại nhận dữ liệu.

### 3.2.2. Server Firebase và cách kết nối phần cứng:

Firebase là dịch vụ cơ sở dữ liệu hoạt động trên nền tảng đám mây – cloud. Kèm theo đó là hệ thống máy chủ cực kỳ mạnh mẽ của Google. Chức năng chính là giúp người dùng lập trình ứng dụng bằng cách đơn giản hóa các thao tác với cơ sở dữ liệu. Công cụ của Firebase được sử dụng trong đồ án là Realtime Database.

Để kết nối FireBase với phần cứng ta cần khai báo FIREBASE\_HOST và FIREBASE\_AUTH vào trong code Arduino dưới dạng câu lệnh “Firebase.begin(FIREBASE\_HOST, FIREBASE\_AUTH);

* Cách lấy FIREBASE\_HOST và FIREBASE\_AUTH:
  + Cách lấy FIRE\_HOST:



FIREBASE\_HOST

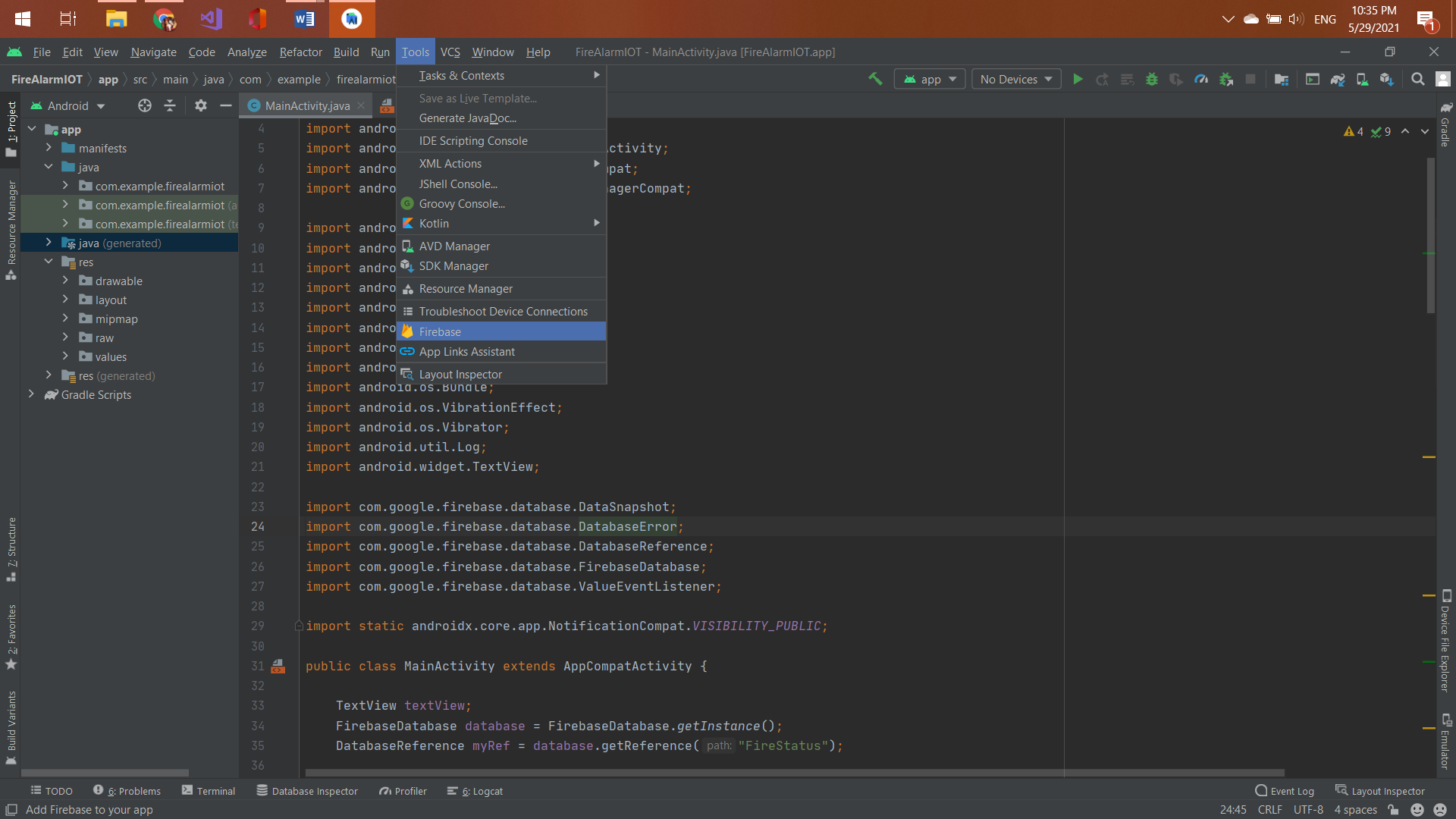
* + Cách lấy FIREBASE\_AUTH: truy cập Project Setting > Service accounts > Database secrets > Copy dòng code phía dưới.

### 3.2.3. Tạo ứng dụng kết nối bằng Android Studio:

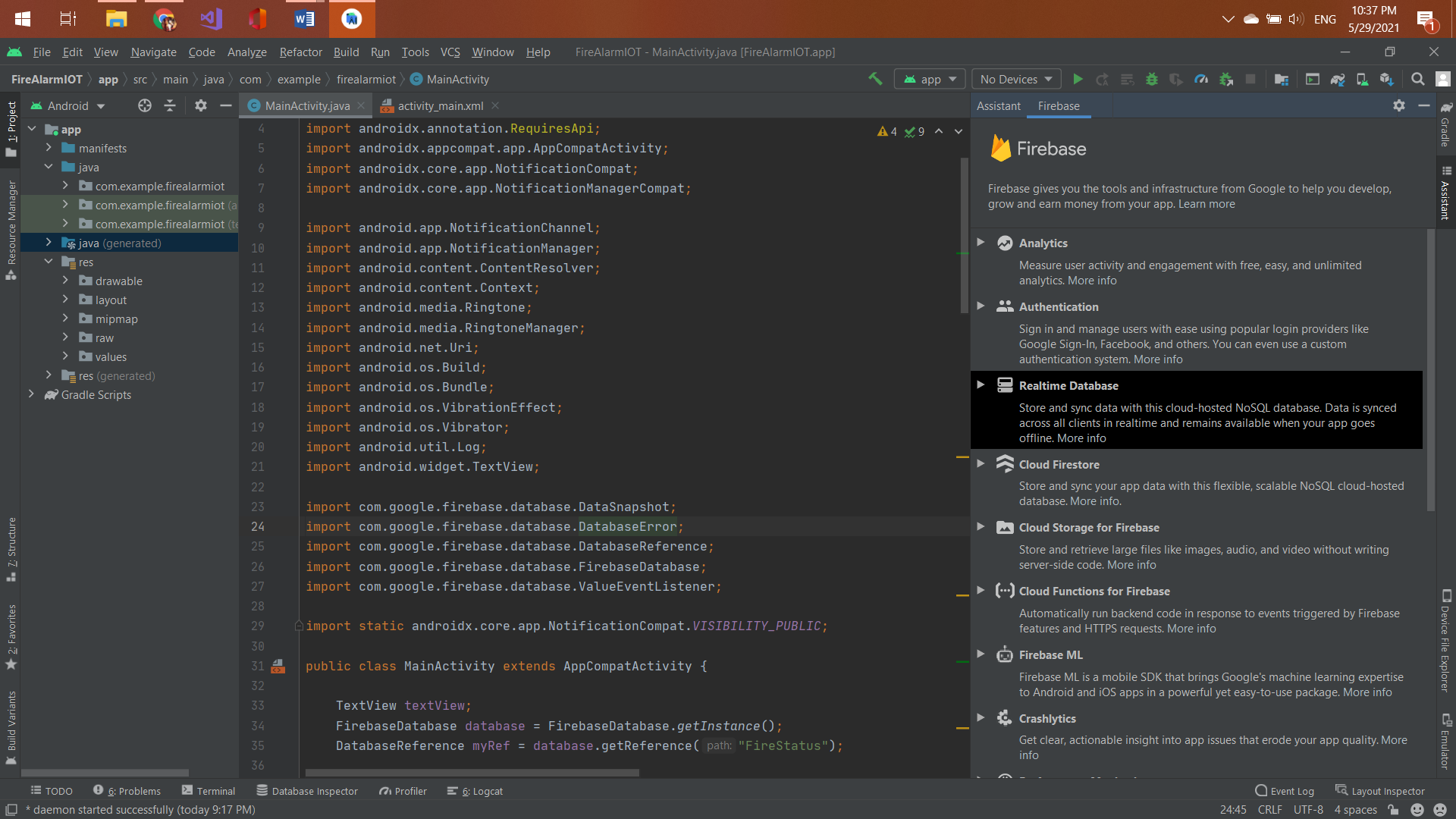
Android Studio là phần mềm giúp tạo ra một ứng dụng android với đầy đủ các công cụ được cung cấp đảm bảo sản phẩm được tạo ra đầy đủ các tính năng và khả năng như các ứng dụng thường thấy trong điện thoại di động.

Để kết nối ứng dụng và firebase ta cần thực hiện như sau:

* Bước 1: Vào Tool > Firebase. Trường hợp không có ô Firebase hiện ra thì cần vào build.gradle và thêm theo hướng dẫn của trang chủ Firebase

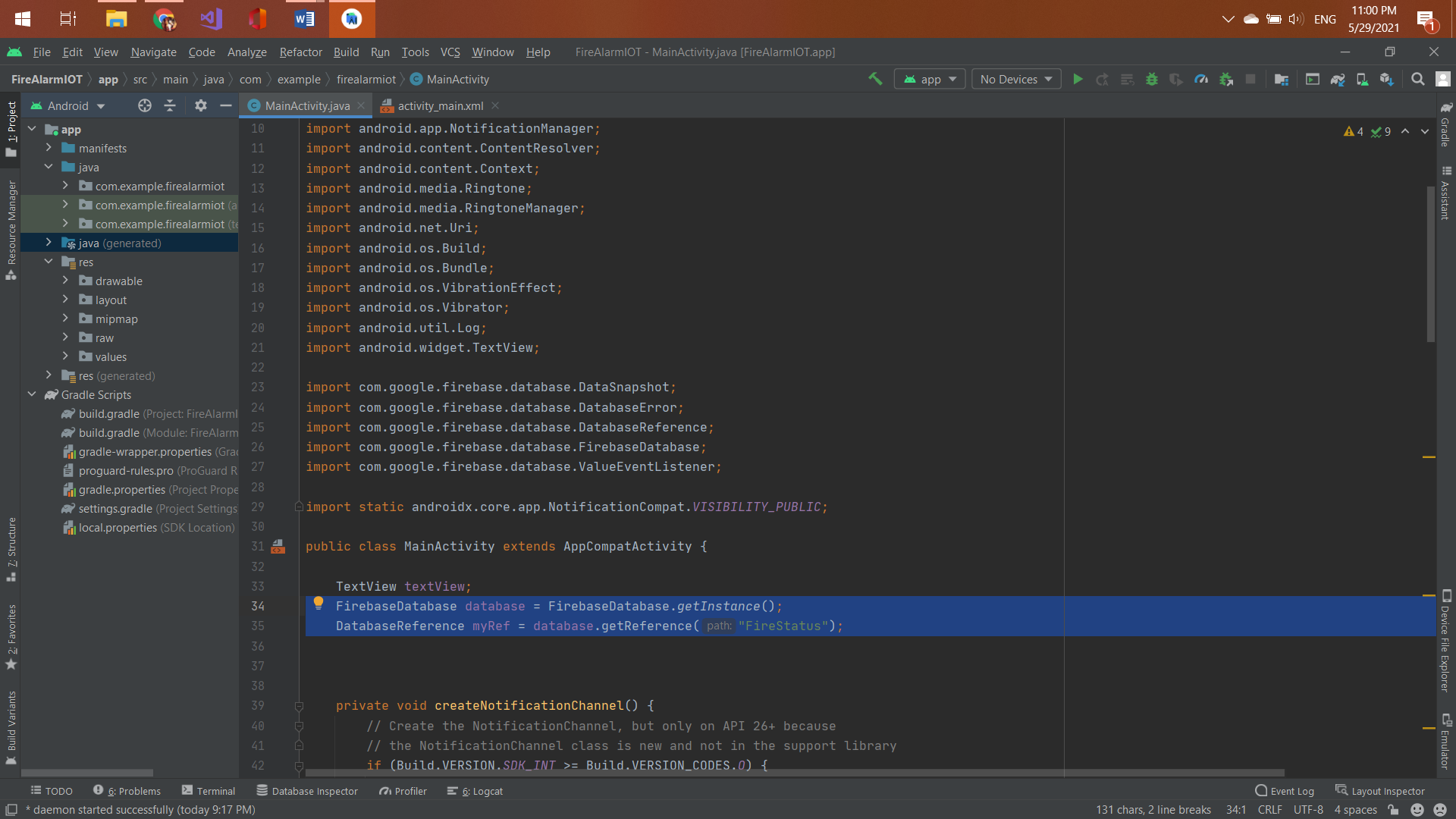


* Bước 2: Vì đồ án này cần cập nhật database liê tục nên ta dùng Realtime Database, sau đó nhấn “*Get started with Realtime Database”*



* Bước 3: Chọn “*Connect to Firebase*” sẽ chuyển sang web Firebase và chọn project để kết nối.

Sau đó trong hàm MainActivity.java ta khai báo:



Như vậy là ta có thể lấy dữ liệu về từ Firebase.

(Code ứng dụng se được để ở phần phụ lục)

# CHƯƠNG 4: kẾT QUẢ VÀ NHẬN XÉT

## 4.1. Kết quả thu được:

### 4.1.1. Kết quả lý thuyết:

Về phần cứng, nhóm tìm hiểu nguyên lý và cách thức hoạt động của các module cảm biến. Khảo sát các khối ngoại vi, thông số kỹ thuật và biết cách giao tiếp của các module.

Về phần mềm, sử dụng được phần mềm Arduino IDE để lập trình trên module ESP8266. Tìm hiểu các bộ thư viện có sẵn, từ đó lập trình cho hệ thống. Hiểu được nguyên lý và cách sử dụng cơ sở dữ liệu của Realtime Database Firebase. Lập trình ứng dụng cảnh báo trên Android qua Android Studio.

### 4.1.2. Kết quả thực tế:

Lần 1: Giá trị được đưa vào Firebase không đổi mặc dù đã đạt điều kiện.

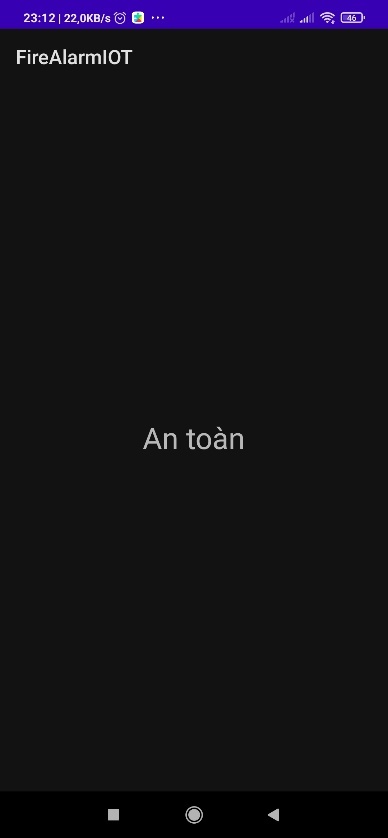
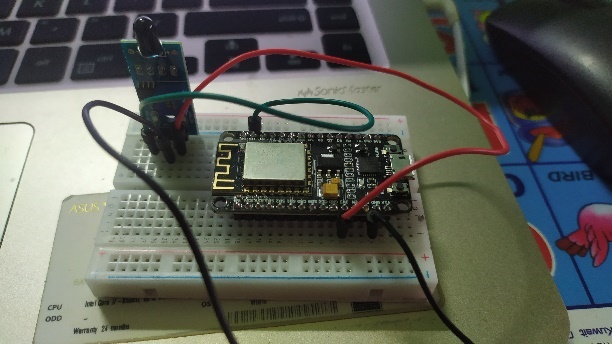
🡪Thay chận D0 thành A0 và chạy ra được kết quả mong muốn.

Lần 2: Giá trị không thể đưa lên Firebase

🡪Kiểm tra code đọc dữ liệu và gởi dữ liệu. Kết quả: đã thành công

Lần 3: Kết quả đã ra được như mong đợi với việc hơ lửa vào module và ứng dụng android hiện thông báo

Hình ảnh thực tế của ứng dụng android và hệ thống phần cứng:

**Ghi chú về ứng dụng**: Người dùng mở ứng dụng trên Smartphone và hiện ra giao diện như trên. Khi xảy ra sự cố như hỏa hoạn, module cảm ứng sẽ gửi dữ liệu về firebase và từ đó về với ứng dụng. Khi đó, thông báo và nhạc chuông cảnh báo sẽ được khởi tạo. Khi nào hết cảm ứng được lửa dữ liệu sẽ được cập nhật lại và nhạc chuông sẽ tắt, nhưng thông báo sẽ vẫn còn để báo là đã có xảy ra hỏa hoạn.

## 4.2. Nhận xét

* Ưu điểm:
  + Cảm biến được lửa tốt, thu thập dữ liệu chính xác
  + Hệ thống hoạt động ổn định, dữ liệu được cập nhật liên tục lên Server và gửi đến ứng dụng di động.
  + Hệ thống tiết kiệm năng lượng
* Nhược điểm:
  + Hệ thống còn khá sơ sài, chưa tối ưu
  + Chỉ kết nối được với mạng đã được lập trình sẵn trong code phần cứng

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. **Kết luận:**

Bài báo cáo thực hiện các yêu cầu đặt ra của đề tài về vấn đề tìm hiểu về công nghệ Interner Of Things, đưa ra được mô hình, kiến trức, với các nội dung như sau:

* Tìm hiểu về cấu trúc, thông số kỹ thuật của các module.
* Tìm hiểu về quá trình xây dựng ứng dụng sử dụng IoT trên FireBase
* Tìm hiểu về cảm biến nhiệt.

1. **Hướng phát triển đề tài:**

Sản phẩm có thể phát triển thêm với cái nội dung sau: giao diện phần cứng thẩm mỹ, tối ưu, nhỏ gọn; Phần mềm tối ưu hơn về mặt giao diện ứng dụng.

Ngoài ra có thể thêm vào các module khác để tăng thêm chức năng cho đồ án, ví dụ cảm biến khói, rung, . . .

# PHỤ LỤC

## Code phần cứng:

#include <FirebaseESP8266.h>

//#include <FirebaseArduino.h>

#include <ESP8266WiFi.h>

#define FIREBASE\_HOST "https://fire-alarm-61203-default-rtdb.firebaseio.com"

#define FIREBASE\_AUTH "QFIpDYBNkgGhxD74kcBJEbPDBCMCFbfD6WiGcLjI"

#define ssid "Sushi"

#define pass "20042020"

FirebaseData firebaseData;

//void notifyOnFire()

//{

// int isButtonPressed=analogRead(D1);

// if(isButtonPressed==1)

// {

// Serial.println("FIRE DETECTED");

// Firebase.setInt("FireStatus",0);

// delay(1000);

// }

// else if (isButtonPressed==0)

// {

// Firebase.setInt("FireStatus",1);

// delay(1000);

// }

//}

void setup()

{

Serial.begin(9600);

pinMode(D1, INPUT\_PULLUP);

while (WiFi.status() != WL\_CONNECTED)

{

Serial.print(".");

delay(500);

}

Serial.println();

Serial.print("connected: ");

Serial.println(WiFi.localIP());

Firebase.begin(FIREBASE\_HOST, FIREBASE\_AUTH);

}

void loop()

{

int sensorVal=analogRead(D1);

if(sensorVal ==1023)

{

Serial.print(".");

Firebase.setInt(firebaseData,"/FireStatus",1);

delay(1000);

}

else if(sensorVal==0)

{

Serial.println("FIRE DETECTED");

Firebase.setInt(firebaseData,"/FireStatus",0);

delay(1000);

}

delay(1000);

}

## Code phần mềm Android:

* **File MainActivity.java:**

package com.example.firealarmiot;

import androidx.annotation.NonNull;

import androidx.annotation.RequiresApi;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.app.NotificationCompat;

import androidx.core.app.NotificationManagerCompat;

import android.app.NotificationChannel;

import android.app.NotificationManager;

import android.content.ContentResolver;

import android.content.Context;

import android.media.Ringtone;

import android.media.RingtoneManager;

import android.net.Uri;

import android.os.Build;

import android.os.Bundle;

import android.os.VibrationEffect;

import android.os.Vibrator;

import android.util.Log;

import android.widget.TextView;

import com.google.firebase.database.DataSnapshot;

import com.google.firebase.database.DatabaseError;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.database.ValueEventListener;

import static androidx.core.app.NotificationCompat.VISIBILITY\_PUBLIC;

public class MainActivity extends AppCompatActivity {

TextView textView;

FirebaseDatabase database = FirebaseDatabase.getInstance();

DatabaseReference myRef = database.getReference("FireStatus");

private void createNotificationChannel() {

// Create the NotificationChannel, but only on API 26+ because

// the NotificationChannel class is new and not in the support library

if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.O) {

CharSequence name = "Trangthai";

String description = "Nope";

int importance = NotificationManager.IMPORTANCE\_DEFAULT;

NotificationChannel channel = new NotificationChannel("CHANNEL\_ID", name, importance);

channel.setDescription(description);

NotificationManager notificationManager = getSystemService(NotificationManager.class);

notificationManager.createNotificationChannel(channel);

}

}

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

textView = findViewById(R.id.text1);

createNotificationChannel();

myRef.addValueEventListener(new ValueEventListener() {

@RequiresApi(api = Build.VERSION\_CODES.LOLLIPOP)

@Override

public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

String trangthai="";

NotificationCompat.Builder mBuilder;

String i = String.valueOf(dataSnapshot.getValue());

if (i.equals("0")) {

trangthai = "Báo động";

mBuilder = new NotificationCompat.Builder(getApplicationContext(), "CHANNEL\_ID")

.setSmallIcon(R.drawable.ic\_launcher\_foreground)

.setContentTitle("Tình trạng")

.setContentText(trangthai)

.setPriority(NotificationCompat.PRIORITY\_DEFAULT)

.setVisibility(VISIBILITY\_PUBLIC)

.setAutoCancel(true);

Vibrator vib = (Vibrator) getSystemService(Context.VIBRATOR\_SERVICE);

if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.O) {

vib.vibrate(VibrationEffect.createOneShot(5000, VibrationEffect.DEFAULT\_AMPLITUDE));

} else {

//deprecated in API 26

vib.vibrate(5000);

}

playNotificationSound();

NotificationManagerCompat notificationManager = NotificationManagerCompat.from(getApplicationContext());

notificationManager.notify(1, mBuilder.build());

}

else {

trangthai = "An toàn";

mBuilder = new NotificationCompat.Builder(getApplicationContext(), "CHANNEL\_ID")

.setSmallIcon(R.drawable.ic\_launcher\_foreground)

.setContentTitle("Tình trạng")

.setContentText(trangthai)

.setPriority(NotificationCompat.PRIORITY\_DEFAULT)

.setVisibility(VISIBILITY\_PUBLIC)

.setAutoCancel(true);}

textView.setText(trangthai);

Log.d("Tag", i);

}

//

@Override

public void onCancelled(@NonNull DatabaseError databaseError) {

}

});

}

public void playNotificationSound() {

try {

Uri soundUri = Uri.parse(ContentResolver.SCHEME\_ANDROID\_RESOURCE + "://"+ getApplicationContext().getPackageName() + "/" + R.raw.disaster\_notification);

Ringtone r = RingtoneManager.getRingtone(getApplicationContext(), soundUri);

r.play();

} catch (Exception e) {

e.printStackTrace();

}

}

public void createNotification(String trangthai)

{

}

}

* **File activity\_main.xml:**

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".MainActivity"

android:background="#673AB7"

>

<ImageView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentEnd="true"

android:layout\_alignParentRight="true"

android:layout\_alignParentBottom="true"

android:layout\_gravity="center"

android:layout\_marginEnd="106dp"

android:layout\_marginRight="106dp"

android:layout\_marginBottom="547dp"

android:src="@drawable/fire\_prevent" />

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_alignParentBottom="true"

android:textColor="#ffffff"

android:layout\_marginBottom="456dp"

android:gravity="center"

android:text="If this set alarm call 114 immediately for the sake of your life."

android:textAlignment="center"

android:textSize="24sp" />

<TextView

android:id="@+id/text1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textColor="#ffffff"

android:text="True"

android:textSize="30dp"

android:layout\_centerInParent="true"

android:layout\_gravity="center\_horizontal"

/>

</RelativeLayout>