

ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH
KHOA ĐIỆN - ĐIỆN TỬ
BỘ MÔN ĐIỆN CÔNG NGHIỆP



HCMUTE

ĐỒ ÁN TỐT NGHIỆP

Đề tài:

**ỨNG DỤNG IOT TRONG GIÁM SÁT NĂNG LƯỢNG ĐIỆN
VÀ HỖ TRỢ QUẢN LÝ CHO THUÊ**

GVHD: TS. Nguyễn Phan Thanh

SVTH:	MSSV
Bùi Nguyễn Hiếu Nhân	20342018
Đinh Hoàng Nhật	20342024
Nguyễn Duy Phương	20342019

TP. HỒ CHÍ MINH, THÁNG 7 NĂM 2022

ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH
KHOA ĐIỆN - ĐIỆN TỬ
BỘ MÔN ĐIỆN CÔNG NGHIỆP



HCMUTE

ĐỒ ÁN TỐT NGHIỆP

Đề tài:

**ỨNG DỤNG IOT TRONG GIÁM SÁT NĂNG LƯỢNG ĐIỆN
VÀ HỖ TRỢ QUẢN LÝ CHO THUÊ**

GVHD: TS. Nguyễn Phan Thanh

SVTH:	MSSV
Bùi Nguyễn Hiếu Nhân	20342018
Đinh Hoàng Nhật	20342024
Nguyễn Duy Phương	20342019

TP. HỒ CHÍ MINH, THÁNG 7 NĂM 2022

NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

Sinh viên thực hiện: Bùi Nguyễn Hiếu Nhân

MSSV: 20342018

Đinh Hoàng Nhật

MSSV: 20342024

Nguyễn Duy Phương

MSSV: 20342019

Lớp: 20342A

Khoa Điện – Điện Tử, chuyên ngành Điện Công nghiệp

Tên đề tài: **ỨNG DỤNG IOT TRONG GIÁM SÁT NĂNG LƯỢNG ĐIỆN
VÀ HỖ TRỢ QUẢN LÝ CHO THUẾ**

1. Mục tiêu đề tài:

- Board mạch đo đếm điện năng PZEM-004T thu thập dữ liệu và giao tiếp với IC mạng ESP đồng bộ dữ liệu về Google Cloud Firestore hoạt động chính xác và đảm bảo tính ổn định
- Xây dựng hệ thống sever lưu trữ và giao tiếp dữ liệu broad mạch nhúng và người dùng App (Ios và Android) trên dịch vụ cloud Google Firestore và Google FireBase.
- Phát triển và hoàn thiện Moblie App trên React Native Framework, Cross Platform Moblie, build native cho cả Android và IOS với các chức năng giám sát thu thập năng lượng và quản lý phòng trọ, đăng ký tam trú tạm vắng,....

2. Nội dung các phần thuyết minh:

- Tổng quan về hệ thống
- Thiết kế mạch và App

3. Ngày giao nhiệm vụ: 26/03/2022

4. Ngày hoàn thành nhiệm vụ: 8/7/2022

5. Họ và tên cán bộ hướng dẫn: TS. Nguyễn Phan Thanh

Giáo viên hướng dẫn

Thông qua bộ môn

Tp. HCM, ngày.....tháng.....năm 2022

Chủ nhiệm bộ môn

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

-----***-----

PHIẾU NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

Họ và tên sinh viên: Bùi Nguyễn Hiếu Nhân
Đinh Hoàng Nhật
Nguyễn Duy Phương

MSSV: 20342018
MSSV: 20342024
MSSV: 20342019

Ngành: Công nghệ kỹ thuật Điện – Điện tử

Tên đề tài: **ỨNG DỤNG IOT TRONG GIÁM SÁT NĂNG LƯỢNG ĐIỆN
VÀ HỖ TRỢ QUẢN LÝ CHO THUÊ**

Giáo viên hướng dẫn: TS. Nguyễn Phan Thanh
NHẬN XÉT

1. Về nội dung đề tài & khối lượng thực hiện:

.....
.....

2. Ưu điểm:

.....
.....

3. Nhược điểm:

.....
.....

4. Đề nghị cho bảo vệ hay không?

.....
.....

5. Đánh giá loại:

.....
.....

6. Điểm:(Bằng chữ:)

Tp. Hồ Chí Minh, ngày ... tháng ... năm 2022
Giáo viên hướng dẫn

(Ký & ghi rõ họ tên)

ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM
KHOA ĐIỆN – ĐIỆN TỬ
BỘ MÔN: ĐIỆN CÔNG NGHIỆP

BẢNG NHẬN XÉT VÀ ĐÁNH GIÁ CỦA GIÁO VIÊN PHẢN BIỆN

I. Tiêu chí và điểm đánh giá

Criteria	Grade					Điểm đánh giá
	Very Poor	Poor	Adequate	Very Good	Ideal	
Mục 1: Mức độ thời sự của đề tài, độ khó của đề tài	0-2	3-5	6-7	8 - 9	10	
	Quá dễ thực hiện	Thực hiện được nhưng thực tế không cần	Vân đề vừa sức/Cần phải dành thời gian nghiên cứu	Vân đề khó/Cần nhiều kiến thức tổng hợp đã học	Vân đề rất khó/Cần nhiều kiến thức tổng hợp đã học	
Mục 2: Tính ứng dụng của đề tài vào thực tiễn	0-2	3-5	6-7	8 - 9	10	
	Không có ứng dụng	Thỉnh thoảng có ứng dụng	Có ứng dụng	Thực tế bên ngoài đang cần	Thực tế bên ngoài đang rất cần và cấp thiết	
Mục 3: Tính đúng đắn của đề tài, phương pháp nghiên cứu hợp lý	0-2	3-5	6-7	8 - 9	10	
	Không hợp lý	Có phương pháp nghiên cứu, nhưng chưa rõ ràng	Có phương pháp nghiên cứu, định hướng đúng	Phương pháp nghiên cứu rõ ràng, định hướng đúng	Phương pháp nghiên cứu rõ ràng, khoa học, phù hợp với đề tài, sáng tạo	
Mục 4: Giải pháp & công nghệ, thi công/mô phỏng	0-6	7-15	16-21	22-27	28-30	
	Không có	Giải pháp sơ sài	Giải pháp rõ ràng, có thi công mô hình/mô phỏng	Giải pháp rõ ràng, có quy trình thực hiện thi công/mô phỏng vận hành được	Giải pháp rõ ràng, có quy trình thực hiện thi công/mô phỏng vận hành được, kết quả mô phỏng/vận hành tốt, sáng tạo	

	0-8	9-20	21-28	29-36	37-40	
Mục 5: Xem đĩa CD trình bày báo cáo nội dung LV	Nội dung không phù hợp với mục tiêu	Báo cáo đơn giản, chưa đầy đủ cấu trúc, nội dung như đã đề ra	Có đủ cấu trúc, nội dung	Có đầy đủ cấu trúc nội dung, trình bày hợp lý, khoa học	Có đầy đủ cấu trúc nội dung, trình bày hợp lý, khoa học, logic, rõ ràng, dễ hiểu, đúng quy định về trình bày luận văn, không có lỗi chính tả, sáng tạo	

❖ Các vấn đề cần làm rõ

❖ Các nội dung cần bổ sung hiêu chỉnh

.....
.....
.....

❖ **Ý kiến kết luận: Đồng ý hay không đồng ý cho bảo vệ**

.....

Tp. HCM, ngày....tháng....năm 2022

Giảng viên phản biện

LỜI CẢM ƠN

Sau thời gian học tập tại trường Đại Học Sư Phạm Kỹ Thuật cùng với sự hướng dẫn tận tình của quý Thầy/ Cô, nhóm đồ án đã hoàn thành đồ án tốt nghiệp.

Nhóm xin gửi lời cảm ơn chân thành đến Thầy Nguyễn Phan Thanh đã tận tình góp ý, hướng dẫn nhiệt tình trong quá trình nhóm thực hiện đề tài. Xin được gửi lời cảm ơn đến Thầy được nhiều sức khỏe và thành công nhiều hơn trong quá trình giảng dạy của mình.

Nhóm xin chân thành cảm ơn quý Thầy/ Cô khoa Điện - Điện tử đã cho nhóm kiến thức về lĩnh vực chuyên ngành, những kinh nghiệm về quá trình làm việc nhóm và tạo điều kiện cho nhóm thực hiện đề tài rất hay này, tạo thêm nhiều hiểu biết cho nhóm trước khi ra trường. Do trình độ và kiến thức còn hạn chế nên đề tài có thể còn nhiều sai sót mong Khoa - Bộ môn và quý Thầy/ Cô góp ý để đề tài nhóm thêm hoàn thiện hơn.

Nhóm cũng xin gửi lời cảm ơn đến gia đình và bạn bè đã tạo điều kiện và giúp đỡ nhóm rất nhiều để có được kết quả như hôm nay.

MỤC LỤC

LỜI CẢM ƠN	i
MỤC LỤC	ii
DANH MỤC HÌNH ẢNH	v
DANH MỤC BẢNG BIÊU	viii
DANH MỤC CÁC TỪ VIẾT TẮT	ix
CHƯƠNG 1: TỔNG QUAN	1
I. TÍNH CẦN THIẾT CỦA ĐỀ TÀI	1
1. Khảo sát thực tế.....	1
2. Lợi ích của việc giám sát năng lượng điện và quản lý cho thuê bằng App	3
3. Ưu điểm.....	5
4. Nhược điểm.....	6
5. Nhận xét	7
II. MỤC TIÊU	7
III. NHIỆM VỤ	7
IV. PHƯƠNG PHÁP NGHIÊN CỨU	8
V. PHẠM VI ĐỀ TÀI.....	9
VI. KẾT QUẢ DỰ KIẾN ĐẠT ĐƯỢC	9
VII. BỐ CỤC	10
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT	11
I. GIỚI THIỆU PHẦN CÙNG.....	11
1. Modul nguồn 5V 3W.....	11
2. Relay 5V 10A.....	12
3. Mạch thu phát WiFi ESP8266 ESP-12F	13
4. Mạch chuyển USB UART CP2102	15
5. Màn hình OLED 1.3inch	16
6. Modul đo đếm điện năng AC PZEM-004T	17
II. CHUẨN TRUYỀN GIỮ LIỆU	17
1. Giao tiếp UART.....	17
2. Chuẩn giao tiếp I2C	19
3. Chuẩn giao tiếp WiFi.....	22

III. GIỚI THIỆU FRAMEWORK	23
1. Arduino IDE	23
2. Phần mềm thiết kế mạch Schematic và PCB	24
3. React Native.....	28
4. Redux.....	31
5. React Native Lifecycle	33
6. Xcode	33
7. Android Studio	34
8. NodeJS.....	35
9. ES6.....	36
10. ReactJS.....	36
IV. GOOGLE CLOUD FIREBASE	37
1. Giới thiệu về Firebase	37
2. Firebase Authentication	38
2. Firebase Realtime Database.....	39
3. Cloud Firestore.....	41
CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG.....	45
I. PHẦN NHÚNG	45
1. Cài đặt Arduino IDE	45
2. Các chế độ hoạt động của ESP8266	49
3. WebSocket	52
4. Ghi và đọc giữ liệu bộ nhớ EEPROM	54
5. Đọc dữ liệu từ PZEM-004T về ESP8266.....	57
6. In dữ liệu lên màn hình OLED	58
7. Khởi tạo tập dữ liệu từ ESP8266 và gửi lên Cloud Firestore	59
II. KHỞI TẠO SERVER	62
1. Khởi tạo Google Cloud Firestore.....	62
2. Khởi tạo Server để ESP8266 kết nối tới Cloud Firestore	65
3. Khởi tạo Server để thiết bị IOS kết nối tới Cloud Firestore.....	66
4. Khởi tạo Server để thiết bị Android kết nối tới Cloud Firestore	68
III. SETUP MÔI TRƯỜNG REACT NATIVE TRÊN MACOS BIG SUR.....	71
1. Thiết lập môi trường React Native.....	71
2. Thiết lập React Native Firebase.....	76

3. Thiết lập React Native Cloud Firestore	78
4. Thiết lập React Native Authentication.....	79
5. Thiết lập React Native Vector Icon	80
IV. YÊU CẦU CỦA HỆ THỐNG	82
1. ESP8266 kết nối tới Server.....	82
2. App kết nối đến Server	85
V. THIẾT KẾ PHẦN CỨNG	86
1. Phần mạch nguồn.....	86
2. Phần ESP - 12F.....	87
3. Sơ đồ mạch Schematic và PCB	88
CHƯƠNG 4: VẬN HÀNH HỆ THỐNG	90
I. PHẦN NHÚNG	90
1. Mô tả hoạt động của thiết bị	90
2. Sơ đồ kết nối của thiết bị	91
II. PHẦN GIAO DIỆN APP CHO ANDROID VÀ IOS.....	94
III. PHẦN SERVER GOOGLE CLOUD FIREBASE	104
CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN.....	107
I. KẾT LUẬN.....	107
II. HƯỚNG PHÁT TRIỂN	108
TÀI LIỆU THAM KHẢO.....	109
PHỤ LỤC	110

DANH MỤC HÌNH ẢNH

Hình 1. 1: Biểu đồ thể hiện số lượng nhà ở tại TP.HCM tăng theo từng năm.	1
Hình 1. 2: Biểu đồ thể hiện số lượng phòng trọ ở TP.HCM tăng mạnh.....	1
Hình 1. 3: Biểu đồ khả năng tìm kiếm phòng trọ của sinh viên trường Đại Học Vinh.....	2
Hình 1. 4: Xu hướng biến động FDI vào Việt Nam giai đoạn 1988 – 2019.	2
Hình 2. 1: Modul nguồn Hi-Link AC – DC 5V 3W.....	11
Hình 2. 2: Relay SRD-5VDC-SL-C.	12
Hình 2. 3: Mạch thu phát WiFi ESP8266 ESP-12F.	13
Hình 2. 4: Sơ đồ chân mạch thu phát WiFi ESP8266 ESP-12F.	14
Hình 2. 5: Mạch chuyển USB UART CP2102.	16
Hình 2. 6: Màn hình OLED 1.3in.	16
Hình 2. 7: Modul đo đếm điện năng PZEM-004T.	17
Hình 2. 8: Truyền dữ liệu bằng chuẩn UART.	18
Hình 2. 9: Thiết bị ngoại vi giao tiếp bus I2C.	19
Hình 2. 10: Thiết bị kết nối vào I2C ở chế độ chuẩn và chế độ nhanh.	20
Hình 2. 11: Quá trình giao tiếp giữa thiết bị chủ - tớ.	20
Hình 2. 12: Trình tự truyền dữ liệu.....	21
Hình 2. 13: Giản đồ thời gian điều kiện START và STOP.	21
Hình 2. 14: Mô hình hoạt động của mạng WiFi.	22
Hình 2. 15: Các chuẩn WiFi 802.11.	23
Hình 2. 16: Giao diện phần mềm Arduino IDE.....	24
Hình 2. 17: Tạo dự án đầu tiên để thiết kế mạch.	25
Hình 2. 18: Download những thư viện có sẵn trên mạng EasyEDA.....	26
Hình 2. 19: Chuyển từ Schematic sang PCB trên EasyEDA.	26
Hình 2. 20: Phần mềm Altium Designer.	27
Hình 2. 21: Ví dụ cơ bản sử dụng React Native.	29
Hình 2. 22: Chế độ xem được sử dụng trong các ứng dụng Android và IOS.	30
Hình 2. 23: Components của React Native.....	31
Hình 2. 24: Nguyên lý vận hành của Redux.....	32
Hình 2. 25: Ứng dụng Redux vào logout login.	32
Hình 2. 26: Vòng đời trong React Native.....	33
Hình 2. 27: Giao diện Xcode.	34
Hình 2. 28: Giao diện Android Studio.....	35
Hình 2. 29: Cài đặt ReactJS.....	36
Hình 2. 30: Giao diện Firebase.	37
Hình 2. 31: Thiết lập và cấu hình SDK.	38
Hình 2. 32: Giao diện Firebase Authentication.....	39
Hình 2. 33: Giao diện Firebase Realtime Database.....	40
Hình 2. 34: Giao diện Cloud Firestore.	42

Hình 3. 1: Giao diện tải phần mềm Arduino IDE.....	45
Hình 3. 2: Cấu hình cho Arduino IDE.....	46
Hình 3. 3: Dowload cấu hình của board ESP8266.	46
Hình 3. 4: Sơ đồ phần cứng ESP-12F để nạp code.	47
Hình 3. 5: Add thư viện bằng Manage Libraries và Add .ZIP Library.	48
Hình 3. 6: ESP8266 hoạt động ở chế độ softAP	49
Hình 3. 7: Ví dụ về chế độ softAP của ESP8266.	50
Hình 3. 8: Chế độ WiFi Station.	51
Hình 3. 9: Ví dụ về chế độ STA cho ESP8266.	52
Hình 3. 10: ESP8266 được thiết lập làm Client.	53
Hình 3. 11: ESP8266 được thiết lập làm Server.	54
Hình 3. 12: Chương trình ghi dữ liệu vào bộ nhớ EEPROM.	55
Hình 3. 13: Chương trình đọc dữ liệu từ bộ nhớ EEPROM.	56
Hình 3. 14: Chương trình đọc dữ liệu pzem về esp8266.	57
Hình 3. 15: Chương trình in dữ liệu thời gian thực lên màn hình OLED.	58
Hình 3. 16: Chương trình khởi tạo tập tin từ ESP8266 gửi lên Firestore.	59
Hình 3. 17: Hiển thị trên Firestore khi tạo thành công.	60
Hình 3. 18: Chương trình cập nhập giá trị từ ESP8266 gửi lên Firestore.	61
Hình 3. 19: Chương trình lấy dữ liệu từ Cloud Firestore về ESP8266.	61
Hình 3. 20: Khởi tạo dự án.	62
Hình 3. 21: Khởi tạo cloud Firestore database.	63
Hình 3. 22: Tuỳ chọn bảo mật và kết nối.	64
Hình 3. 23: Chọn vị trí đặt server.	64
Hình 3. 24: Khởi tạo server để esp8266 kết nối với cloud Firestore.	66
Hình 3. 25: Khởi tạo server cho thiết bị IOS kết nối với cloud Firestore.	67
Hình 3. 26: Cấu hình Xcode kết nối tới cloud Firestore.	68
Hình 3. 27: Khởi tạo server cho thiết bị Android kết nối tới cloud Firestore.	69
Hình 3. 28: Cấu hình Android kết nối tới cloud Firestore.	70
Hình 3. 29: Cấu hình thành công.	70
Hình 3. 30: Cài đặt Node và Watchman Android.	71
Hình 3. 31: Cài đặt Java Development Kit.	71
Hình 3. 32: Cài đặt Android Studio.	71
Hình 3. 33: Cài đặt Android SDK.	72
Hình 3. 34: Cấu hình biến ANDROID_SDK_ROOT.	72
Hình 3. 35: Tạo một dự án mới trên Android.	73
Hình 3. 36: Chạy React Native trên Android.	73
Hình 3. 37: Khởi động ứng dụng trên máy ảo Android.	73
Hình 3. 38: Cài đặt Node và Watchman IOS.	74
Hình 3. 39: Cấu hình Xcode.	74
Hình 3. 40: Cài đặt mô phỏng IOS trong Xcode.	74
Hình 3. 41: Tạo một dự án mới trên IOS.	75
Hình 3. 42: Chạy React Native trên IOS.	75
Hình 3. 43: Khởi động ứng dụng trên máy ảo IOS.	75
Hình 3. 44: Cài đặt qua NPM.	76

Hình 3. 45: Tạo thông tin đăng nhập Android.....	76
Hình 3. 46: Cấu hình Firebase bằng thông tin đăng nhập Android.....	76
Hình 3. 47: Tạo thông tin đăng nhập IOS.....	77
Hình 3. 48: Cấu hình Firebase bằng thông tin đăng nhập IOS.....	78
Hình 3. 49: Cài đặt React Native Cloud Firestore.....	78
Hình 3. 50: Truy vấn đến tập tài liệu trên Cloud Firestore.....	78
Hình 3. 51: Bắt đầu và kết thúc truy vấn.....	79
Hình 3. 52: Thêm và cập nhập tài liệu lên Cloud Firestore.....	79
Hình 3. 53: Cài đặt React Native Authentication	79
Hình 3. 54: Đăng nhập bằng email password Authentication.....	80
Hình 3. 55: Đăng xuất Authentication.....	80
Hình 3. 56: Tạo thư mục font trong mục IOS.	81
Hình 3. 57: Thêm font vào dự án.....	81
Hình 3. 58: Lưu đồ giải thuật ESP8266 kết nối đến Server	84
Hình 3. 59: Lưu đồ giải thuật cấu hình mật khẩu wifi cho thiết bị bằng App.....	85
Hình 3. 60: App kết nối Server với người dùng	86
Hình 3. 61: Mạch nguồn cho ESP-12F.....	86
Hình 3. 62: Sơ đồ mạch ESP – 12F kết nối với các phần cứng khác	87
Hình 3. 63: Sơ đồ mạch Schematic	88
Hình 3. 64: Thiết kế mặt trước PCB trên EasyEDA	89
Hình 3. 65: Hình ảnh thực tế mặt trước của PCB.....	89
Hình 3. 66: Thiết kế mặt sau PCB trên EasyEDA.....	90
Hình 3. 67: Hình ảnh thực tế mặt sau của PCB	90
Hình 4. 1: Sơ đồ kết nối của thiết bị	91
Hình 4. 2: Mô hình bên trong của thiết bị	92
Hình 4. 3: Mô hình khi đóng gói thành sản phẩm	93
Hình 4. 4: Giao diện đăng nhập người dùng	94
Hình 4. 5: Giao diện đăng ký người dùng	95
Hình 4. 6: Đăng nhập WiFi cho ESP8266.....	96
Hình 4. 7: Giao diện các thông số điện năng của phòng	97
Hình 4. 8: Giao diện cảnh báo khi sử dụng quá công suất	98
Hình 4. 9: Giao diện bảng tính toán hoá đơn tiền phòng.....	99
Hình 4. 10: Giao diện quét cẩn bước công dân để đăng ký tạm trú tạm vắng.....	100
Hình 4. 11: Giao diện thông tin số điện thoại khách thuê	101
Hình 4. 12: Giao diện giấy đăng ký tạm trú tạm vắng	102
Hình 4. 13: Biểu đồ công suất sử dụng trong ngày	103
Hình 4. 14: Xác thực thông tin người dùng Authentication	104
Hình 4. 15: Thay đổi password cài đặt tài khoản	104
Hình 4. 16: Lựa chọn các phương thức xác thực.....	105
Hình 4. 17: Giám sát dữ liệu Firestore	105
Hình 4. 18: Quản lý lượt đọc và ghi kết nối tới Firestore	106

DANH MỤC BẢNG BIỂU

Bảng 2. 2: So sánh hai phần mềm thiết kế mạch in.....	28
Bảng 2. 3: So sánh hai cơ sở dữ liệu.....	44
Bảng 3. 1: Kết nối ESP-12F với USB-TLL để nạp code.....	47
Bảng 3. 2: Bảng linh kiện sử dụng trong mạch in	88
Bảng 4. 1: Thông số kỹ thuật của thiết bị	92

DANH MỤC CÁC TỪ VIẾT TẮT

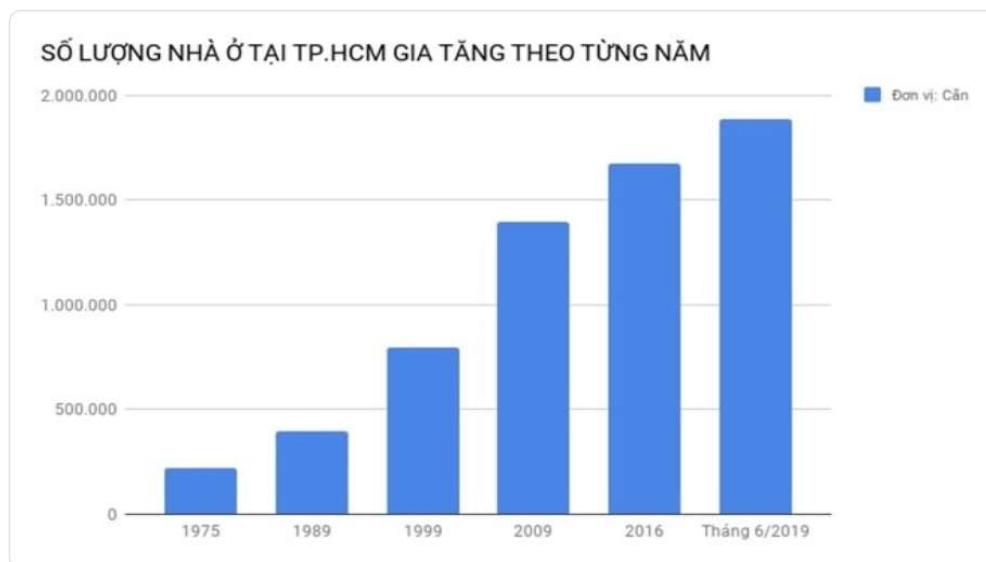
ES6	ECMAScript 6
UI/UX	User Interface và User Experience
MQTT	Message Queuing Telemetry Transport
UART	Universal Asynchronous Receiver/ Transmitter
I2C	Inter – Integrated Circuit
HTML	HyperText Markup Language
API	Application Programming interface
SDK	Software Development Kit
AP	WiFi Access Point
STA	WiFi Station
TCP	Transmission Control Protocol
EEPROM	Electrically Erasable Programmable Read-Only Memory
PCB	Polychlorinated biphenyl
JDK	Java Development Kit

CHƯƠNG 1: TỔNG QUAN

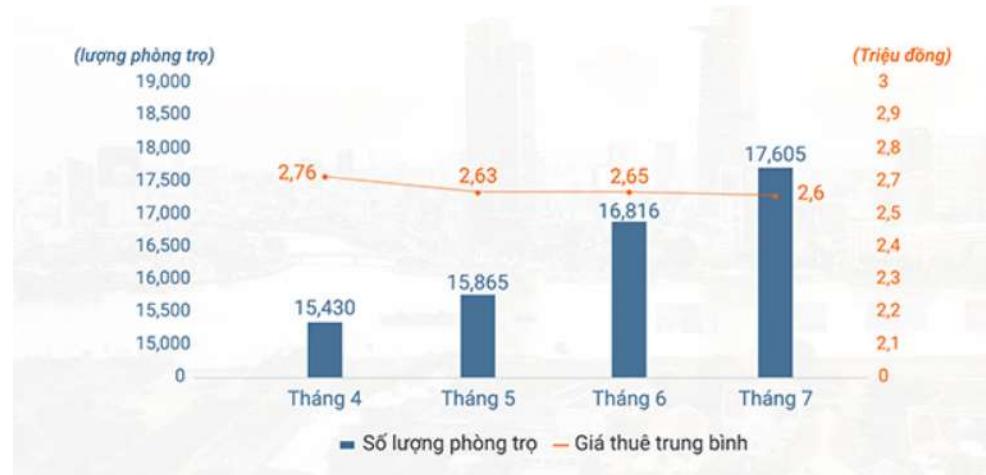
I. TÍNH CẦN THIẾT CỦA ĐỀ TÀI

1. Khảo sát thực tế

- Hiện nay, tỷ lệ gia tăng dân số ở Việt Nam là 0,9% thống kê vào năm 2021 [1] dự báo sẽ tăng trưởng ổn định cho đến năm 2025 vốn đầu tư FDI vào Việt Nam cũng tăng trưởng đều mỗi năm số lượng sinh viên nhập học đại học và cao đẳng cũng duy trì ổn định qua các năm.

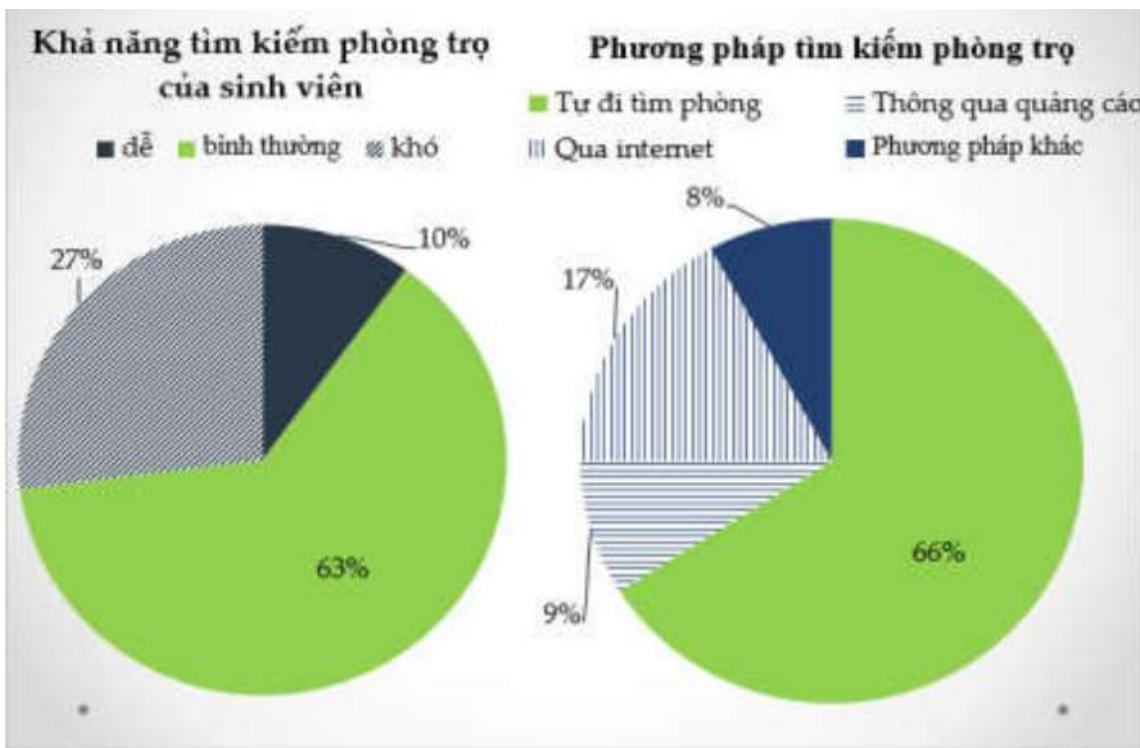


Hình 1. 1: Biểu đồ thể hiện số lượng nhà ở tại TP.HCM tăng theo từng năm

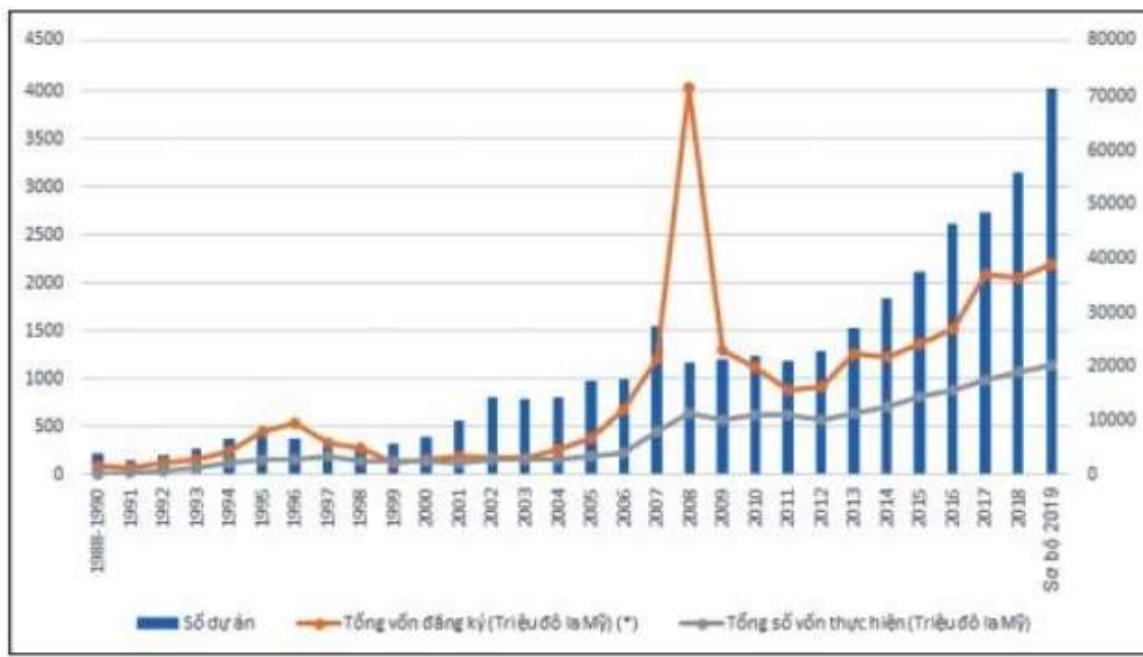


Hình 1. 2: Biểu đồ thể hiện số lượng phòng trọ ở TP.HCM tăng mạnh

CHƯƠNG 1: TỔNG QUAN



Hình 1. 3: Biểu đồ khả năng tìm kiếm phòng trọ của sinh viên trường Đại Học Vinh



Hình 1. 4: Xu hướng biến động FDI vào Việt Nam giai đoạn 1988 – 2019

CHƯƠNG 1: TỔNG QUAN

- Các con số tăng trưởng trên dần đến cầu nhà ở tăng cao thị trường cho thuê phòng trọ nhà ở sẽ tăng trưởng ở các thành phố lớn nói chung và các tỉnh lẻ nói riêng nhu cầu cao dần đến chi phí quản lý cao việc đi ghi điện trực tiếp tại nhà tồn nhiều thời gian và nhân lực để thực hiện. Việc tạo ra ứng dụng giám sát năng lượng sẽ giải quyết được vấn đề đó, giúp cho người chủ dễ dàng quản lý và tiết kiệm chi phí nhân công. Ngoài ra nhóm còn ứng dụng thêm việc đăng kí cho thuê ở khu trọ trực tiếp trên ứng dụng để tiện lợi hơn cho người đăng kí ở trọ tích hợp đăng ký tạm trú tạm vắng online giúp việc quản lý trở nên thông minh tiết kiệm thời gian.
- Từ những nhu cầu trên mang internet of Things (IOT) ra đời là một liên mạng, trong đó các trang thiết bị được nhúng với các bộ phận điện tử, cảm biến, cơ cấu chấp hành cùng với khả năng kết nối mạng máy tính giúp cho các thiết bị này có thể thu thập và truyền tải dữ liệu. IoT đang được sử dụng rộng rãi trong cuộc sống giúp mọi người sống và làm việc thông minh hơn.
- Ứng dụng giám sát năng lượng điện là ứng dụng thu thập dữ liệu từ thiết bị nhúng bao gồm các thông số như: dòng điện, điện áp, công suất tiêu thụ,... và gửi lên App đã được lập trình giúp cho người quản lý dễ dàng thấy được các thông số về điện trực tiếp trên App và quản lý nó một cách dễ dàng.

2. Lợi ích của việc giám sát năng lượng điện và quản lý cho thuê bằng App

Phương pháp quản lý nhà trọ thủ công ngày nay không thực sự mang lại hiệu quả cao, độ chính xác và nhanh chóng. Đặc biệt khi quy mô kinh doanh nhà trọ ở khắp nơi ngày càng mở rộng. Trong trường hợp như thế phần mềm quản lý cho thuê phòng trọ sẽ giúp mọi người xử lý được nhiều số liệu phức tạp một cách nhanh chóng và chính xác hơn.

a. Tiết kiệm thời gian và chi phí

- Tiết kiệm thời gian và chi phí quản lý các phòng trọ là điều mong muốn nhất của các chủ nhà trọ. Việc sử dụng phần mềm quản lý nhà trọ sẽ đáp ứng được mong muốn này. Phần mềm sẽ hỗ trợ cho chủ nhà trọ cách quản lý nhanh chóng và hiệu quả. Không cần phải quản lý thủ công như trước nữa.

CHƯƠNG 1: TỔNG QUAN

- Sử dụng phần mềm quản lý còn giúp chủ nhà trọ tiết kiệm chi phí. Nhà trọ quy mô lớn sẽ phải thuê đi thuê nhân công để làm việc, mỗi người phụ trách một công việc sẽ tốn nhiều nhân công. Với việc quản lý bằng phần mềm, mọi hoạt động của khu trọ sẽ được quản lý trên cùng một hệ thống. Vì vậy chỉ cần người chủ quản lý phần mềm là đủ, giúp chủ nhà trọ sẽ tiết kiệm được chi phí thuê nhân công. Không cần phải tốn thêm chi phí nhân công như trước. Người chủ hoàn toàn có thể quản lý hiệu quả bằng việc sử dụng App.

b. *Chủ động sử dụng mọi lúc khi cần*

- Với việc quản lý dãy trọ của mình mà không cần phải có mặt tại đó là lợi ích quan trọng mà phần mềm quản lý mang lại. Bởi vì phần mềm có tính năng quản lý ở mọi nơi. Chỉ cần một thiết bị có kết nối internet, chủ nhà trọ sẽ quản lý được tất cả các hoạt động trực tiếp trên App mà không cần phải đến tận nơi. Ngoài ra còn giúp người thuê trọ chủ động hơn trong việc giám sát năng lượng điện của phòng mình để sử dụng điện một cách hợp lý.
- Phần mềm còn giúp cho người thuê trọ chủ động đăng ký tạm trú tạm vắng trực tiếp trên App mà không cần phải tới tận nơi để đăng kí.

c. *Giám sát năng lượng điện chuẩn xác*

- Phần mềm sẽ giảm thiểu những sai sót trong việc giám sát năng lượng điện. Hàng tháng, phần mềm sẽ liên tục cập nhập chỉ số điện năng theo thời gian thực và tính toán chính xác số tiền điện sử dụng của người thuê trọ. Đồng thời, chỉ số điện năng của tất cả các tháng sẽ được lưu trữ trên đám mây. Khi thấy sự bất thường, người chủ có thể kiểm tra lại một cách dễ dàng. Điều này giúp quản lý nhà trọ dễ dàng và khoa học hơn.

d. *Quản lý toàn bộ thông tin khách thuê*

- Thông tin người thuê trọ là một dữ liệu quan trọng. Với việc có được thông tin này giúp người chủ có thể kiểm soát tốt hơn về vấn đề an ninh phức tạp đang diễn ra ở các khu trọ đông người. Ngoài ra việc quản lý các hoạt động khác cũng dễ dàng hơn như là : liên lạc được với khách thuê khi cần thiết, gửi thông báo tiền thuê hàng tháng,...

e. *Gửi thông báo đến khách cho thuê*

- Chủ nhà trọ có thể kết nối dễ dàng với người thuê thông qua App. Gửi những thông báo cần thiết đến từng khách thuê trọ. Bao gồm: thông báo thu tiền, thông báo giảm tiền điện, thông báo phòng tránh dịch bệnh,... Khách thuê nhanh chóng tiếp nhận được các thông báo. Kịp thời thực hiện các yêu cầu, quy định của nhà trọ trong việc đóng tiền thuê nhà và các chi phí khác đúng hạn cho chủ nhà trọ.

f. *Tránh được các rủi ro*

- Phần mềm sẽ giúp nhà trọ tránh được những rủi ro đáng tiếc vì phần mềm đã lưu trữ toàn bộ hợp đồng, các hóa đơn thanh toán trên App nên việc xảy ra mâu thuẫn, tranh cãi giữa chủ nhà và khách thuê được hạn chế.

3. Ưu điểm

a. *Quản lý tự động, tiết kiệm nhân lực*

- Phần mềm sẽ thu thập dữ liệu từ đồng hồ đo điện của phòng trọ như: điện áp, dòng điện, công suất tiêu thụ,... Sau đó sẽ cho hiện thị trên App, App sẽ tự động tính toán số tiền điện hàng ngày, hàng tháng của khách thuê. Khách thuê chỉ việc tải ứng dụng và đăng nhập sẽ xem được tất cả thông số về năng lượng điện, tiền phòng,... của phòng mình thuê. Mọi thứ đều minh bạch, rõ ràng, người chủ không cần phải đi ghi tiền điện nước hàng tháng trực tiếp tại phòng mà có thể theo dõi trực tiếp trên App. Nhờ vào việc này, người chủ trọ có thể tiết kiệm được các chi phí về nhân lực,...
- Ngoài ra, khách đi thuê có thể đăng ký tạm trú tạm vắng trực tiếp trên App mà không cần phải đến tận nơi. Việc này giúp tiết kiệm thời gian và chi phí cho cả người chủ trọ lẫn người thuê trọ.

b. *Sử dụng được trên nhiều thiết bị*

- App được thiết kế có thể sử dụng cho nhiều thiết bị kể cả hệ điều hành Android và IOS thông dụng nhất hiện nay. Vì vậy chỉ cần một chiếc điện thoại thông minh kết nối internet, ta có thể xem được các thông số đã thiết lập trên App mọi lúc mọi nơi giúp cho người chủ trọ và người thuê chủ động hơn trong việc nắm bắt thông tin.

CHƯƠNG 1: TỔNG QUAN

c. Lưu trữ thông tin

- Những thông tin của khách hàng bao gồm những thông tin cá nhân, tiền điện, tiền phòng,... sẽ được lưu trữ tất cả trên App, những thông tin này người chủ và người thuê đều xem được điều này giúp cho người chủ và người thuê nắm bắt được những thông tin cần thiết.

4. Nhược điểm

Việc quản lý cho thuê phòng trọ cách thủ công hiện nay đang thực sự không hiệu quả nhất là trong thời đại 4.0 hiện nay, việc này làm giảm hiệu quả kinh doanh của người chủ. Sau đây là một số nhược điểm của việc quản lý cho thuê cách thủ công.

a. Khó khăn trong việc quản lý

- Với việc quản lý thủ công, người chủ phải cho nhân viên đi đến từng phòng ghi chép số liệu như tiền điện nước và những chi phí phát sinh khác,... bằng giấy bút hàng tháng. Việc này có thể dẫn đến sai sót không đáng có và có thể xảy ra mâu thuẫn giữa người chủ và người cho thuê, làm giảm uy tín của người chủ. Ngoài ra việc đến tận nơi để đăng ký tạm trú tạm vắng sẽ rất bất tiện cho người đi thuê trong trường hợp như người chủ trợ không có nhà hoặc là hết phòng.

b. Lưu trữ thông tin kém hiệu quả

- Việc lưu trữ thông tin khách thuê và một số hoá đơn thanh toán như tiền điện, tiền phòng,... trên sổ sách sẽ làm tăng nguy cơ làm mất thông tin, chưa kể còn tốn thời gian cho việc tìm kiếm lại thông tin nếu có xảy ra mâu thuẫn giữa người chủ và người thuê. Việc này dẫn đến sự thiếu minh bạch trong thông tin và có thể xảy ra mâu thuẫn giữa hai bên.

c. Tốn thời gian và chi phí

- Việc đi đến từng phòng để ghi tiền điện nước, tiền phòng và những thông báo hàng tháng sẽ tốn nhiều thời gian, chưa kể ở một số dãy trọ lớn người chủ có thể thuê nhiều người để làm những công việc này hàng tháng. Việc này sẽ tốn thêm những chi phí phát sinh cho nhân công.

CHƯƠNG 1: TỔNG QUAN

d. Hiệu quả kinh doanh thấp

- Trong quản lý thủ công ở những dãy trọ có quy mô lớn, việc ghi chép, lưu trữ thông tin khách thuê, quản lý những số liệu thủ công sẽ gây ra nhiều rủi ro như những tình trạng ghi sai số tiền điện hàng tháng làm tính tiền sai. Việc không có những thông tin để đối chứng sẽ làm phát sinh những mâu thuẫn không đáng có giữa hai bên.
- Ngoài ra việc đi đến từng phòng để thông báo những chi phí sinh hoạt cho người thuê cũng sẽ tốn nhân công để thực hiện. Những việc này làm sẽ làm giảm hiệu quả kinh doanh của người chủ dẫn đến làm giảm lợi nhuận.

5. Nhận xét

Nhận thấy được ưu điểm lớn của việc giám sát năng lượng điện và quản lý cho thuê bằng App và nhược điểm của việc quản lý thủ công. Nhóm đã quyết định thực hiện đề tài này nhằm khắc phục những khó khăn mà người chủ trọ và người thuê gặp phải. Với việc ứng dụng IOT trong đề tài này cũng là phù hợp với xu hướng phát triển 4.0 ở Việt Nam hiện nay.

II. MỤC TIÊU

- Board mạch đo đếm điện năng PZEM-004T thu thập dữ liệu và giao tiếp với IC mạng ESP đồng bộ dữ liệu về Google Cloud Firestore hoạt động chính xác và đảm bảo tính ổn định.
- Xây dựng hệ thống sever lưu trữ và giao tiếp dữ liệu broad mạch nhúng và người dùng App (IOS và Android) trên dịch vụ Google Cloud Firestore.
- Phát triển và hoàn thiện Moblie App trên React Native Framework, Cross Platform Moblie, build native cho cả Android và IOS với các chức năng giám sát thu thập năng lượng và quản lý phòng trọ, đăng ký tạm trú tạm vắng,...

III. NHIỆM VỤ

- Tìm hiểu về phần mềm EasyEDA Designer thiết kế mạch Schematic và PCB đảm bảo các tiêu chuẩn trong thiết kế và gia công mạch in, đọc hiểu Datasheet, sơ đồ kết nối của ESP và modul PZEM-004T.

CHƯƠNG 1: TỔNG QUAN

- Hoàn thiện thiết kế lập trình nhúng trên arduino giao tiếp thu thập liệu tiến hành kiểm thử dữ liệu các thông số nhận về với đồng hồ vạn năng để so sánh và đánh giá mức độ hoàn thiện độ chính xác của thiết bị đã thiết kế. Sau đó tổng hợp và xử lý làm nhỏ data trước khi gửi về Google Firestore
- Xây dựng hệ thống sever giao tiếp và lưu trữ dữ liệu trên cloud Google Firestore bao gồm:
 - Tìm hiểu về dịch vụ Google Cloud Firestore cấu hình cài đặt và chính sách bảo mật.
 - Tìm hiểu về hệ điều hành linux các lệnh teminal cơ bản và ngôn ngữ lập trình nodejs.
 - Xây dựng mosquito mqtt bao MQTT client (Publisher/Subscriber) và MQTT broker (máy chủ mô giới) dữ liệu sẽ được truyền nhận realtime.
 - Xây dựng mongodatabase cho việc lưu trữ dữ liệu.
 - Tìm hiểu về authenticator firebase connection react native (Google FireBase) để xác thực người dùng.
- Xây dựng App người dùng (IOS và Android)
 - Tìm hiểu về ngôn ngữ lập trình Javascript ES6 và Framework React-Native, Xcode (IOS) và Android Studio (Android) để cấu hình cài đặt chạy ứng dụng đầu tiên.
 - Xây dựng giao diện đăng nhập và đăng ký giao diện giám sát,... đảm bảo các tiêu chuẩn cơ bản của ux/ui.

IV. PHƯƠNG PHÁP NGHIÊN CỨU

- Sử dụng phương pháp nguyên cứu vật lý:
 - Hệ thống nhúng: tìm hiểu về datasheet, cấu tạo, nguyên lý hoạt động, modul PZEM-004T, ESP8266 so sánh dữ liệu thu thập được với đồng hồ vạn năng và trong datasheet nhà sản xuất để đánh giá chi tiết kết quả đạt được sai số và mức độ hoàn thiện sản phẩm.

CHƯƠNG 1: TỔNG QUAN

- Hệ thống sever: chạy và kiểm tra thử hệ thống tốc độ truyền dữ liệu độ trễ và tính đồng nhất dữ liệu giữa hệ thống nhúng với sever và app mức độ ổn định của hệ thống các lỗi hệ thống có thể xảy ra và đưa ra hướng khắc phục sửa chữa.
- App người dùng: kiểm tra tính đồng nhất dữ liệu giữ hệ thống nhúng sever và app độ trễ dữ liệu mức độ hoàn thiện về giao diện tính toán của app đảm bảo các tiêu chuẩn ix/iu cơ bản.
 - Xây dựng một cách cụ thể như mục tiêu đã đặt ra. Chạy thử nghiệm trên 2 nền tảng Android và IOS.

V. PHẠM VI ĐỀ TÀI

- Tìm hiểu nghiêm cứu cơ bản về hệ thống nhúng và ngôn ngữ lập trình nhúng micropython với arduino ứng dụng vào việc đọc dữ liệu modul PZEM-004T trả về giao tiếp với IC mạng ESP sau đó đưa lên Internet.
- Thiết kế mạch Schematic và PCB trên phần mềm EasyEDA chạy thử và kiểm tra lỗi đi dây.
- Tìm hiểu về các phương thức giao tiếp và truyền nhận dữ liệu realtime trong hệ thống IOT từ đó đánh giá và chọn ra phương thức phù hợp với ứng dụng MQTT.
- Hiểu về ngôn ngữ lập trình Nodejs và Google Cloud Firestore từ đó xây dựng hệ thống sever giao tiếp dữ liệu và tìm hiểu về cấu trúc dữ liệu mongodb và giải thuật sắp xếp để truy vấn dữ liệu hợp lý.
- Hiểu về ngôn ngữ lập trình Javascript ES6 sử dụng framework React-Native để code giao diện App cho người dùng giám sát năng lượng điện quản lý phòng trọ ở mọi nơi.

VI. KẾT QUẢ DỰ KIẾN ĐẠT ĐƯỢC

- Thu thập dữ liệu điện năng từ PZEM-004T và giao tiếp với IC mạng ESP đưa lên Internet hệ thống Google Cloud Firestore.
- Xây dựng hệ thống sever lưu trữ và truyền nhận dữ liệu hoạt động ổn định trên Google Cloud Firestore.
- Thiết kế thành công App để giám sát năng lượng điện cho nhà ở.

CHƯƠNG 1: TỔNG QUAN

- Kết quả thu được các thông số như: dòng điện, điện áp, công suất tiêu thụ,... Từ mạch đo đếm điện năng không chênh lệch quá nhiều so với đồng hồ đo đếm điện năng.
- Việc sử dụng App để quản lý năng lượng điện giúp cho người chủ giảm bớt chi phí nhân công, không cần phải đi ghi điện ở từng hộ và tránh những sai sót không đáng có.
- Việc thiết kế thêm phần đăng ký tạm trú tạm vắng online trên App giúp cho người đi thuê trợ chủ động hơn trong việc đi thuê. Không cần phải đến tận nơi mà có thể đăng ký trực tiếp trên App.

VII. BÓ CỤC

Chương 1: Tổng quan.

Chương 2: Cơ sở lý thuyết .

Chương 3: Phân tích thiết kế hệ thống.

Chương 4: Vận hành mô hình.

Chương 5: Kết luận và hướng phát triển đè tài.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

I. GIỚI THIỆU PHẦN CỨNG

1. Modul nguồn 5V 3W

- Module Nguồn AC-DC 5V 3W là module chuyển đổi điện áp AC - DC với điện áp đầu vào 220V AC thành đầu ra 5V DC – 3W.
- Module có thiết kế nhỏ gọn, cứng cáp thích hợp với các mạch cần nguồn nhỏ gọn.
- Module này được xem như một nguồn điện chuyển đổi điện áp AC sang DC.
- Có tính chống nhiễu cao giúp cho điện áp cung cấp ổn định.



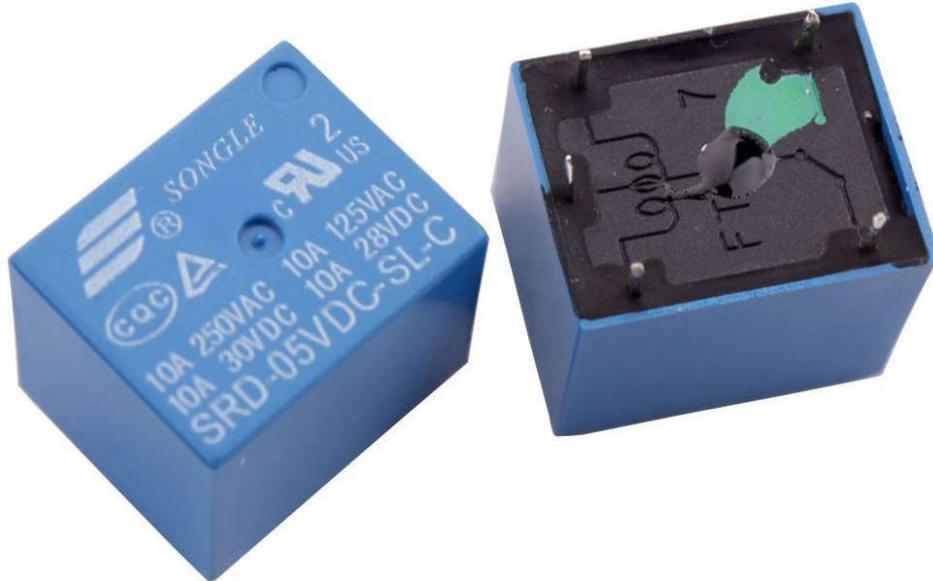
Hình 2. 1 Modul nguồn Hi-Link AC – DC 5V 3W

❖ Thông số kỹ thuật:

- Điện áp đầu vào: 100V – 240VAC.
- Tần số đầu vào: 50-60Hz.
- Dòng điện đầu vào tối đa: < 0.2A.
- Điện áp đầu ra: 5VDC.
- Dòng điện đầu ra: 600mA.
- Công suất đầu ra: 3W.
- Nhiệt độ hoạt động: -20°C ~ +60°C.
- Gồm 4 chân: AC , AC , -Vo , +Vo.

2. Relay 5V 10A

Relay 5V là một công tắc chuyển đổi hoạt động bằng điện. Relay có 2 trạng thái ON và OFF, relay ở trạng thái ON hay OFF phụ thuộc vào có dòng điện chạy qua nó hay không.



Hình 2. 2: Relay SRD-5VDC-SL-C

❖ Nguyên lý hoạt động:

- Khi có dòng điện chạy qua relay, dòng điện này sẽ chạy qua cuộn dây bên trong và tạo ra một từ trường hút. Từ trường hút này tác động lên một đòn bẩy bên trong làm đóng hoặc mở các tiếp điểm và sẽ làm thay đổi trạng thái của relay. Số tiếp điểm điện bị thay đổi có thể là 1 hoặc nhiều, tùy vào thiết kế.
- Relay có 2 mạch độc lập nhau hoạt động. Một mạch là để điều khiển cuộn dây của relay: Cho dòng chạy qua cuộn dây hay không, hay có nghĩa là điều khiển relay ở trạng thái ON hay OFF. Một mạch điều khiển dòng điện ta cần kiểm soát có qua được relay hay không dựa vào trạng thái ON hay OFF của relay.
- Chân 3: đặt điện áp 5V.
- Chân 1 và chân 2 được nối vào cuộn hút, khi có điện vào cuộn hút sẽ hút tiếp điểm từ vị trí 4 xuống tiếp điểm 5.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

❖ *Thông số kỹ thuật:*

- Dòng AC max: 10A.
- Dòng AC min: 6A.
- Diameter, PCB hole: 1.3mm.
- Length / Height, external: 22mm.
- Nhiệt độ hoạt động: - 45°C to 75°C.
- Công suất cuộn dây (coil) DC: 360mW.
- Thời gian tác động: 10ms.
- Thời gian nhả hẫm: 5ms.
- Điện áp điều khiển cuộn dây (coil): 5V.

3. Mạch thu phát WiFi ESP8266 ESP-12F

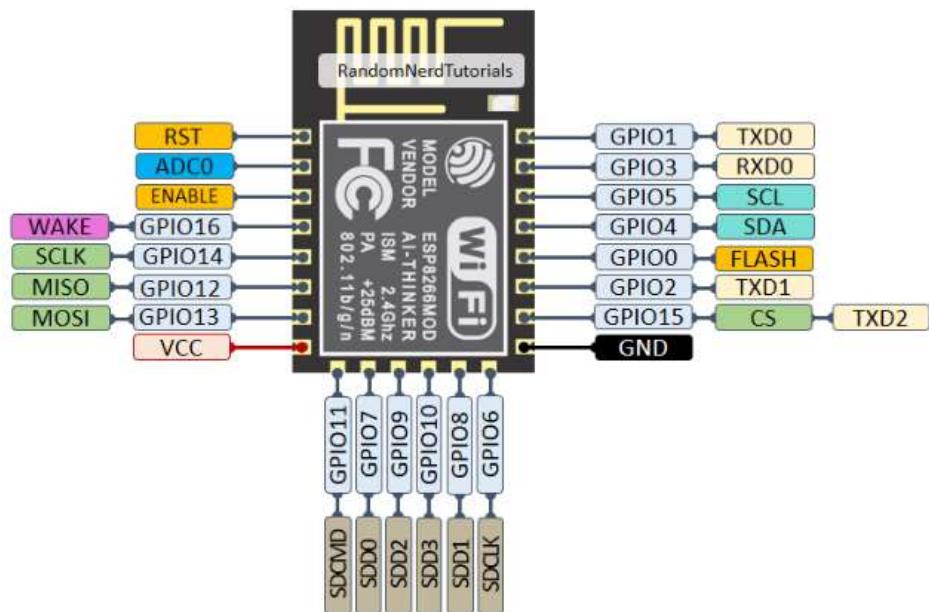
ESP-12F là một mô-đun WiFi được thiết kế nhỏ gọn, tích hợp bộ xử lý lõi ESP8266 MCU 32-bit công suất cực thấp, tốc độ hỗ trợ 80 MHz và 160 MHz, hỗ trợ RTOS và tích hợp WEP, TKIP, AES và WAPI. Modul WiFi này hỗ trợ giao thức IEEE802.11 b/g/n tiêu chuẩn và tích hợp ngăn xếp giao thức TCP/IP hoàn chỉnh. Người dùng có thể sử dụng modul này để bổ sung khả năng kết nối mạng cho các thiết bị hiện có hoặc xây dựng các bộ điều khiển mạng riêng biệt.



Hình 2. 3: Mạch thu phát WiFi ESP8266 ESP-12F

❖ Thông số kỹ thuật

- Điện áp sử dụng: 3.0V – 3.6V (Tối ưu ở mức điện áp 3.3V).
- Dòng điện làm việc: ~ 70mA (tối đa 170mA).
- Dòng điện giữ ở chế độ ngủ: 10uA.
- Nhiệt độ làm việc: -40 °C ~ 125°C.
- Tần số MCU: 80 – 160 MHz.
- SRAM: 36 KB.
- ROM: 4MB (SPI External Flash).
- Tốc độ truyền nhận giữ liệu: 110 – 460800 bps.
- ADC độ phân giải 10bit (0 ~ 1V).
- Hỗ trợ WiFi @2.4GHz, bảo mật WPA/WPA2, hỗ trợ chuẩn 802.11 b/g/n.
- Giao thức mạng: IPv4, TCP/UDP/HTTP/FTP.
- Chế độ hoạt động: Station/Access Point.
- Số ngõ ra giao tiếp: 30 pins (10 GPIO đều có chức năng PWM, I2C, onewire).
- Tích hợp sẵn bộ nhớ Flash 4Mb, các chân từ 9 -> 14 được sử dụng để giao tiếp với IC Flash nên không thể dùng cho bất kì mục đích nào khác.



Hình 2. 4: Sơ đồ chân mạch thu phát WiFi ESP8266 ESP-12F

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

❖ **Các chân được sử dụng trong khi khởi động:** ESP8266 – ESP12F có thể bị ngăn nếu một số chân được kéo mức thấp hoặc cao, sau đây là các chân được sử dụng khi khởi động.

- GPIO16: chân ở mức cao khi khởi động.
- GPIO0: lõi khởi động nếu kéo mức thấp.
- GPIO2: chân ở mức cao khi khởi động, không khởi động được nếu kéo mức thấp.
- GPIO15: lõi khởi động nếu kéo mức cao.
- GPIO3: chân ở mức cao khi khởi động.
- GPIO1: chân ở mức cao khi khởi động, không khởi động được nếu kéo mức thấp.
- GPIO10: chân ở mức cao khi khởi động.
- GPIO9: chân ở mức cao khi khởi động.

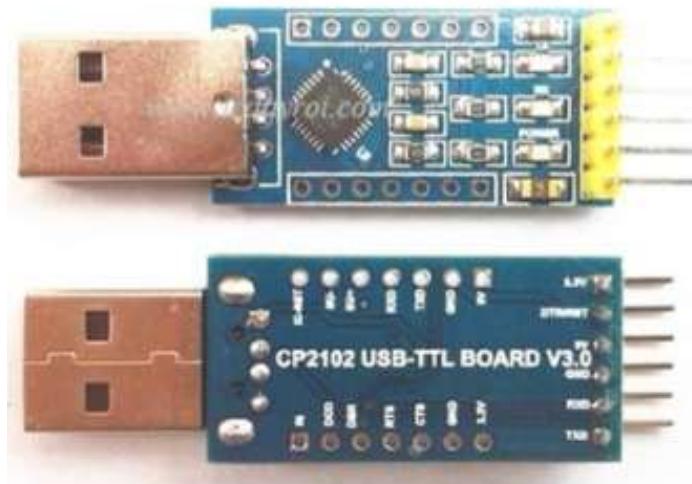
4. Mạch Chuyển USB UART CP2102

Mạch chuyển USB UART CP2102 sử dụng IC CP2102 được dùng để chuyển giao tiếp từ USB sang UART TTL và ngược lại với tốc độ và độ ổn định cao, Driver của mạch có thể nhận trên tất cả các hệ điều hành hiện nay: Windows, MacOS, Linux, Android,...

Tốc độ tối đa 115200 bps. Module có sẵn LED nguồn sáng khi gắn vô máy tính và LED báo hiệu Tx/Rx sẽ sáng khi module nhận/gửi dữ liệu. [3] Link tải Drive từ hãng.

❖ **Thông số kỹ thuật:**

- TXD: chân truyền dữ liệu UART TTL (3.3VDC), dùng kết nối đến chân nhận RX của các module sử dụng mức tín hiệu TTL 3.3~5VDC.
- RXD: chân nhận dữ liệu UART TTL (3.3VDC), dùng kết nối đến chân nhận TX của các module sử dụng mức tín hiệu TTL 3.3~5VDC.
- GND: chân mass hoặc nối đất.
- 5V: chân cấp nguồn 5VDC từ cổng USB, tối đa 500mA.
- DTR: chân tín hiệu DTR, thường được dùng để cấp tín hiệu Reset nạp chương trình cho mạch Arduino.



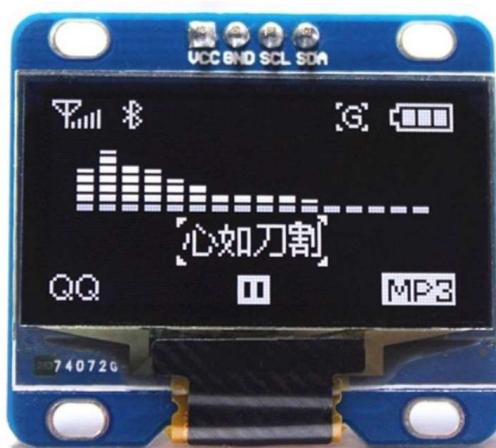
Hình 2. 5: Mạch chuyển USB UART CP2102

5. Màn hình OLED 1.3inch

Màn hình OLED với kích thước 1.3 inch, cho khả năng hiển thị hình ảnh tốt với khung hình 128×64 pixel. Ngoài ra, màn hình còn tương thích với hầu hết các vi điều khiển có giao tiếp I₂C hiện có ngày nay.

❖ *Thông số kỹ thuật:*

- Điện áp sử dụng: 2.2~5.5VDC.
- Công suất tiêu thụ: 0.04w.
- Số điểm hiển thị: 128×64 điểm.
- Giao tiếp: I₂C.
- Driver: SH1106.



Hình 2. 6: Màn hình OLED 1.3in

6. Modul đo đếm điện năng AC PZEM-004T

Modul đo đếm điện năng AC được sử dụng để đo và theo dõi các thông số như điện áp, dòng điện, tần số, cos phi, công suất và công suất tiêu thụ. Modul giao tiếp qua UART dễ dàng kết nối với các vi điều khiển để theo dõi các thông số về điện năng qua UART.



Hình 2. 7: Modul đo đếm điện năng PZEM-004T

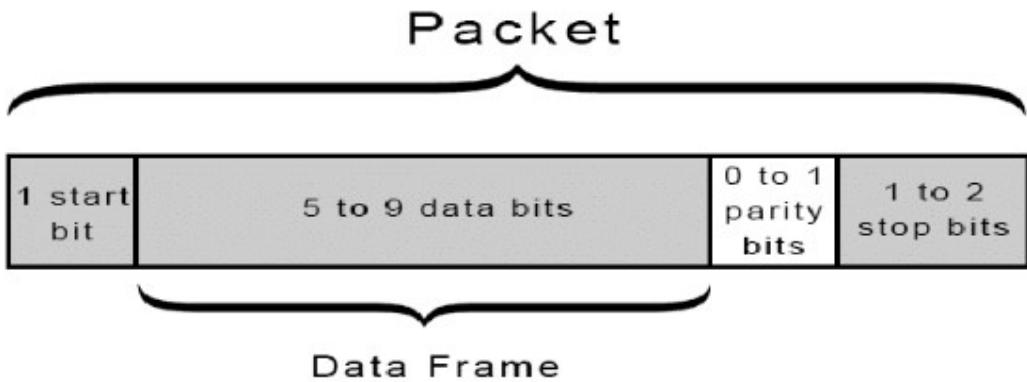
❖ Thông số kỹ thuật:

- Điện áp đo và hoạt động: 80 ~ 260VAC / 50 – 60Hz, sai số 1%.
- Dòng điện đo: 0 ~ 100A, sai số 0.01.
- Công suất đo: 0 ~ 26000W.
- Công suất tiêu thụ: 0 ~ 9999kWh.
- Có opto cách ly an toàn giữa mạch đo và mạch nhận tín hiệu UART.

II. CHUẨN TRUYỀN GIỮ LIỆU

1. Giao tiếp UART

Hiện nay, chuẩn UART (Universal Asynchronous Receiver – Transmitter) được sử dụng rất nhiều trong các board mạch điều khiển để truyền nhận dữ liệu giữ các thiết bị với nhau. Để thực hiện công việc truyền dữ liệu, bắt đầu bằng việc gửi đi một bit START, tiếp theo là các bit dữ liệu và cuối cùng là bit STOP để kết thúc.



Hình 2. 8: Truyền dữ liệu bằng chuẩn UART

Khi chưa truyền dữ liệu thì ban đầu điện áp ở mức logic 1 (mức cao). Khi bắt đầu truyền dữ liệu, bit START chuyển từ mức logic 1 về mức logic 0, báo cho bộ nhận là việc truyền dữ liệu bắt đầu được thực hiện. Tiếp theo là truyền đi các bit dữ liệu (có thể là logic 1 hoặc 0). Bộ nhận sẽ kiểm tra tính đúng đắn của dữ liệu truyền đi dựa theo bit PARITY (kiểm tra chẵn/lẻ). Cuối cùng bit STOP sẽ báo cho thiết bị rằng dữ liệu đã được gửi đi.

Tất cả các board ESP8266, Arduino đều có ít nhất 1 cổng UART hoặc USART. Cổng giao tiếp UART trên chân TX/RX được sử dụng mức logic TTL (5V hoặc 3.3V) để giao tiếp với máy tính hay các thiết bị khác. Nếu đã sử dụng 2 chân TX/RX này thì không thể dùng với mục đích input/output của Arduino nữa.

❖ **Các thông số trong truyền nhận UART:**

- **Baud rate:** hay còn gọi là tốc độ Baud, đây là khoảng thời gian của 1 bit được truyền đi. Lưu ý là phải được cài đặt giống nhau ở thiết bị gửi và nhận.
- **Frame:** khung truyền quy định về số bit trong mỗi lần truyền.
- **Bit start:** đây là bit đầu tiên truyền đi trong một Frame để báo hiệu cho thiết bị nhận sẽ có dữ liệu sắp được truyền đến.
- **Data:** đây là dữ liệu cần gửi, bit trong số nhỏ nhất (LSB) được truyền đi trước và cuối cùng là bit MSB.
- **Parity bit:** kiểm tra tính chẵn/lẻ của dữ liệu được truyền đi.
- **Stop bit:** đây là bit báo cho thiết bị nhận rằng việc gửi dữ liệu đã hoàn tất. Thiết bị nhận sẽ kiểm tra khung truyền nhằm đảm bảo tính đúng đắn của dữ liệu.

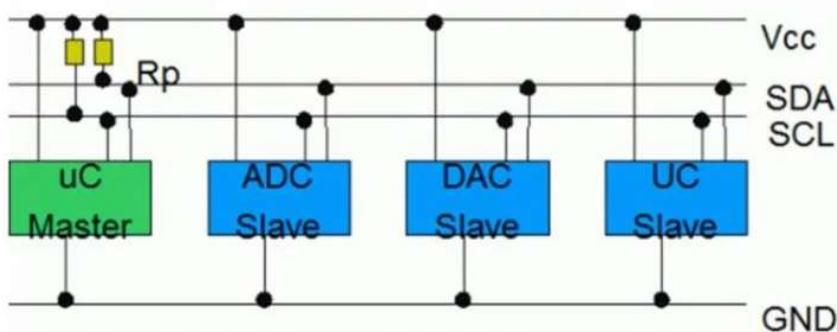
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

- ❖ **Các ứng dụng của UART:** UART thường được sử dụng trong các bộ vi điều khiển cho các yêu cầu chính xác và chúng cũng có sẵn trong các thiết bị liên lạc khác nhau như giao tiếp không dây, thiết bị GPS, Modul Bluetooth, modul WiFi như ESP8266, Arduino và nhiều ứng dụng khác.
- ❖ **Ưu điểm:** chỉ sử dụng hai dây, không cần tín hiệu đồng hồ, có một bit chẵn lẻ để cho phép kiểm tra lỗi, cấu trúc của gói giữ liệu có thể thay đổi, phương pháp được ghi chép rõ ràng và được sử dụng rộng rãi.
- ❖ **Nhược điểm:** kích thước của khung dữ liệu được giới hạn tối đa là 9 bit, không hỗ trợ nhiều hệ thống phụ hoặc nhiều hệ thống chính, tốc độ truyền của mỗi UART phải nằm trong khoảng 10% của nhau.

2. Chuẩn giao tiếp I2C

a. Giới thiệu

I2C là tên viết tắt của cụm từ Inter – Intergrated Circuit. Đây là đường Bus giao tiếp giữa các IC với nhau. Chuẩn giao tiếp I2C đã được rất nhiều nhà sản xuất IC trên thế giới sử dụng. I2C dần trở thành một chuẩn công nghiệp cho các giao tiếp điều khiển. Bus I2C được sử dụng làm bus giao tiếp ngoại vi cho rất nhiều loại IC khác nhau như các loại vi điều khiển 8051, PIC, AVR, ARM,... chip nhớ như: RAM, EEPROM, bộ chuyển đổi tương tự số (ADC), số tương tự (DAC), IC điều khiển LCD, OLED,...

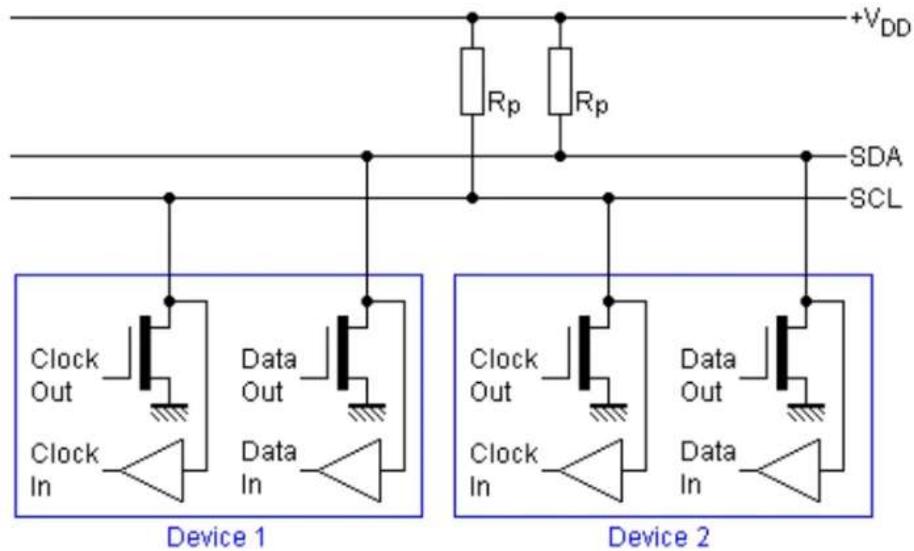


Hình 2. 9: Thiết bị ngoại vi giao tiếp bus I2C

b. Đặc điểm của giao tiếp I2C

Một giao tiếp I2C gồm có 2 dây là Serial Data (SDA) và Serial Clock (SCL). SDA truyền dữ liệu theo 2 hướng, còn SCL là truyền một hướng để truyền xung clock đồng bộ.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

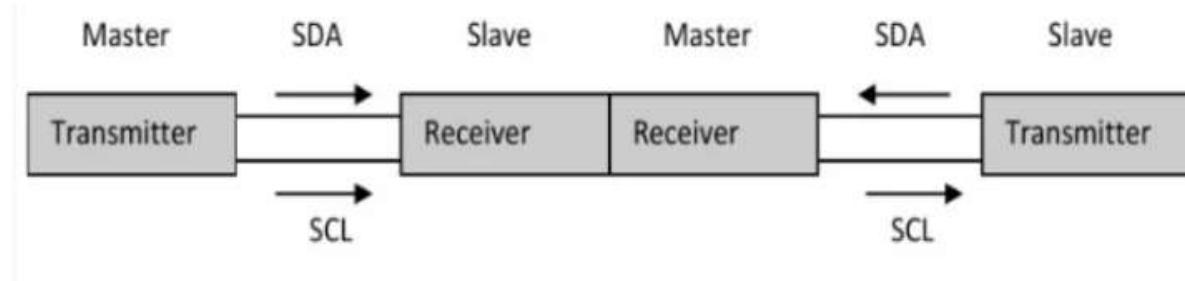


Hình 2. 10: Thiết bị kết nối vào I2C ở chế độ chuẩn và chế độ nhanh

Mỗi dây SDA và SCL điều nối với điện áp dương của nguồn thông qua một điện trở kéo lên. Giá trị của các điện trở này khác nhau tuỳ thuộc vào từng thiết bị và chuẩn giao tiếp.

Ở hình 2.8 bên trên, có rất nhiều thiết bị cùng kết nối vào một bus, tuy nhiên sẽ không xảy ra trường hợp nhầm lẫn giữa các thiết bị, vì mỗi thiết bị sẽ nhận ra bởi một địa chỉ duy nhất, với một quan hệ chủ/tớ tồn tại trong suốt quá trình kết nối. Một thiết bị có thể hoạt động như một thiết bị truyền hay nhận dữ liệu hoặc vừa truyền vừa nhận.

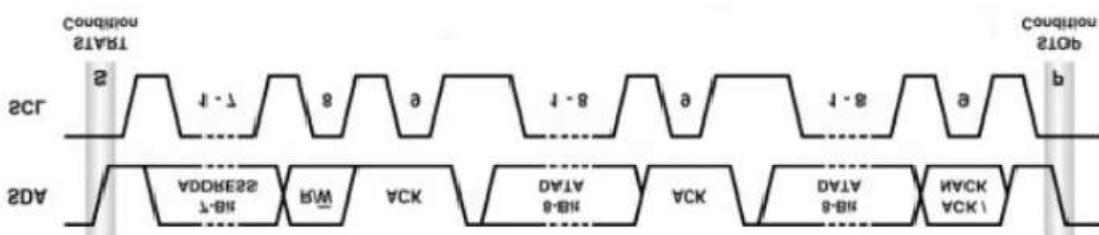
Một thiết bị hay IC khi kết nối với I2C, ngoài địa chỉ duy nhất để phân biệt, nó còn được cấu hình là một thiết bị chủ tớ - với quyền điều khiển thuộc về thiết bị chủ. Khi giữa 2 thiết bị chủ - tớ giao tiếp, thì thiết bị chủ có vai trò tạo xung clock và quản lý địa chỉ của thiết bị tớ trong suốt quá trình giao tiếp.



Hình 2. 11: Quá trình giao tiếp giữa thiết bị chủ - tớ

c. Trình tự truyền bit trên đường truyền

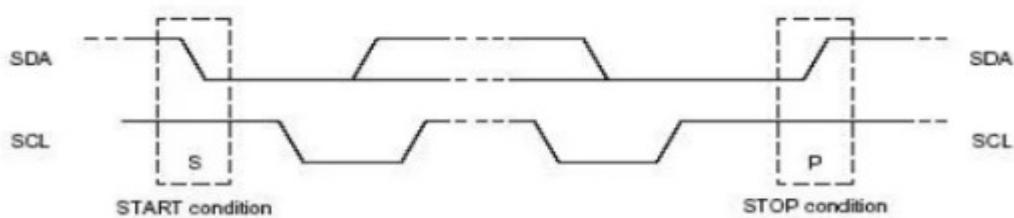
Thiết bị chủ tạo một điều kiện start. Điều kiện này thông báo cho tất cả các thiết bị tớ lắng nghe dữ liệu trên đường truyền. Sau đó, thiết bị chủ sẽ gửi đi một địa chỉ của thiết bị tớ mà thiết bị chủ muốn giao tiếp và đọc/ghi dữ liệu. Thiết bị tớ mang địa chỉ đó trên bus I2C sẽ phản hồi lại bằng một xung ACK. Khi đó việc giao tiếp giữa thiết bị chủ tớ bắt đầu. Bộ truyền gửi 8 bit dữ liệu đến bộ nhận, bộ nhận trả lời với 1 bit ACK. Để kết thúc, thiết bị chủ tạo ra một điều kiện STOP.



Hình 2. 12: Trình tự truyền dữ liệu

d. Điều kiện START và STOP

START là điều kiện khởi đầu, báo hiệu bắt đầu của giao tiếp và ngược lại, STOP là báo hiệu kết thúc.



Hình 2. 13: Giản đồ thời gian điều kiện START và STOP

Khi chưa thực hiện giao tiếp, cả SDA và SCL đều ở mức cao.

- Điều kiện START: sự chuyển đổi từ mức logic cao xuống thấp trên đường SDA trong khi SCL vẫn đang ở mức cao báo hiệu một điều kiện START.
- Điều kiện STOP: sự chuyển đổi trạng thái từ mức logic thấp lên cao trên đường SDA trong khi đường SCL đang ở mức cao. Cả hai điều kiện START, STOP đều được tạo ra bởi thiết bị chủ.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

Trong quá trình giao tiếp, khi có một tín hiệu START được lặp lại thay vì một tín hiệu STOP thì bus I2C vẫn tiếp tục trong trạng thái bận. Tín hiệu START và lặp lại START (Repeated START) đều có chức năng giống nhau là khởi tạo một giao tiếp.

3. Chuẩn giao tiếp WiFi

a. Giới thiệu

WiFi là viết tắt của Wireless Fidelity, được gọi chung là mạng dây sử dụng sóng vô tuyến, loại sóng vô tuyến này tương tự như sóng truyền hình, điện thoại và radio. WiFi phát sóng trong phạm vi nhất định, các thiết bị điện tử tiêu dùng ngày nay như laptop, smartphone hoặc máy tính bảng có thể kết nối và truy cập Internet trong tầm phủ sóng.

b. Nguyên tắc hoạt động

Để tạo được kết nối WiFi nhất thiết phải có Router (bộ thu phát), Router này lấy thông tin từ mạng Internet qua kết nối hữu tuyến rồi chuyển nó sang tín hiệu vô tuyến và gửi đi, bộ chuyển tín hiệu không dây (Adapter) trên các thiết bị di động thu nhận tín hiệu này rồi giải mã nó sang những dữ liệu cần thiết. Quá trình này có thể thực hiện ngược lại, Router nhận tín hiệu vô tuyến từ Adapter rồi giải mã chúng rồi gửi qua Internet.



Hình 2. 14: Mô hình hoạt động của mạng WiFi

c. Một số chuẩn kết nối WiFi

Tuy nói WiFi tương tự như sóng vô tuyến truyền hình, radio hay điện thoại nhưng nó vẫn khác các loại sóng kia ở mức độ tần số hoạt động.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

Sóng WiFi truyền nhận dữ liệu ở tần số 2,5Ghz đến 5Ghz. Tần số cao này cho phép nó mang nhiều dữ liệu hơn phạm vi truyền của nó bị giới hạn; còn các loại sóng khác, tuy tần số thấp nhưng có thể truyền đi rất xa.

Kết nối WiFi sử dụng chuẩn kết nối 802.11 trong thư viện IEEE (Institute of Electrical and Electronics Engineers), chuẩn này bao gồm 4 chuẩn nhỏ a/b/g/n.

CÁC CHUẨN WIFI 802.11					
Chuẩn IEEE	802.11a	802.11b	802.11g	802.11n	802.11ac
Năm phát hành	1999	1999	2003	2009	2013
Tần số	5 GHz	2.4 GHz	2.4 GHz	2.4/5 GHz	5 GHz
Tốc độ tối đa	54 Mbps	11 Mbps	54 Mbps	600 Mbps	1 Gbps
Phạm vi trong nhà	100 ft.	100 ft.	125 ft.	225 ft.	90 ft.
Phạm vi ngoài trời	400 ft.	450 ft.	450 ft.	825 ft.	1,000 ft.

Hình 2. 15: Các chuẩn WiFi 802.11

III. GIỚI THIỆU FRAMEWORK

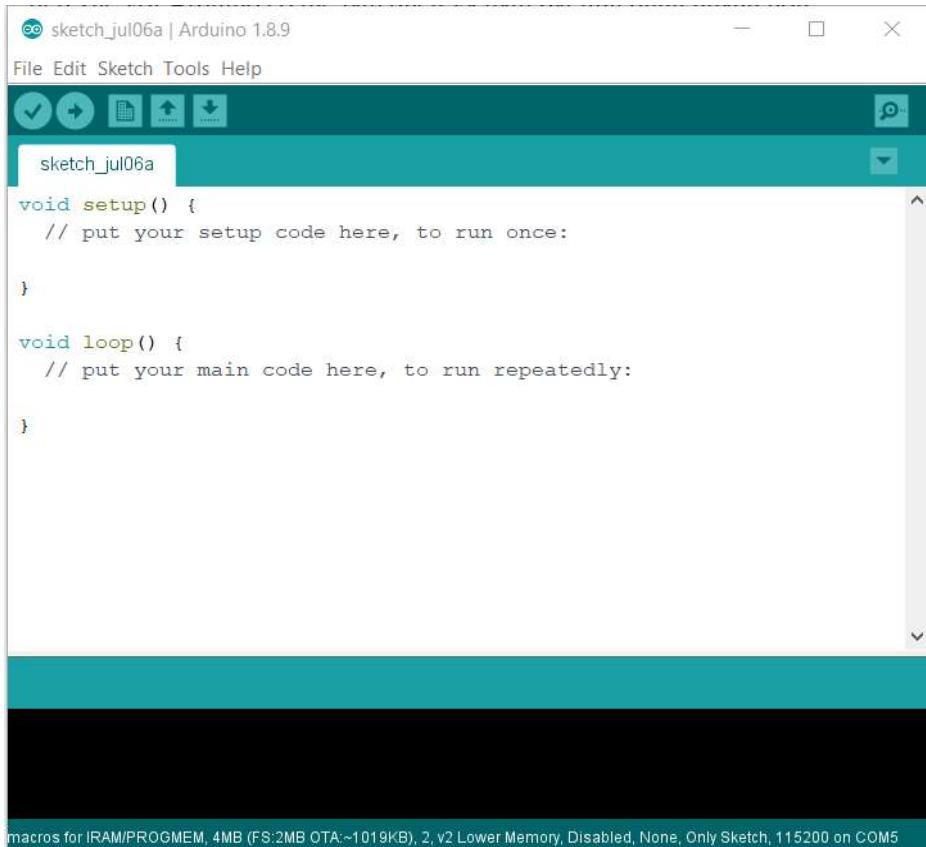
1. Arduino IDE

Arduino là một môi trường phát triển tích hợp đa nền tảng, làm việc cùng với một bộ điều khiển Arduino để viết, biên dịch và tải code lên bo mạch. Phần mềm này cung cấp sự hỗ trợ cho một loạt các bo mạch nhúng như Arduino, ESP8266,...

Ngôn ngữ phổ biến cho Arduino là C và C++, do đó phần mềm phù hợp cho những lập trình viên đã quen thuộc với cả 2 ngôn ngữ này. Các tính năng như làm nổi bật cú pháp, thụt đầu dòng tự động,... làm cho nó trở thành một sự thay thế hiện đại cho các IDE khác.

Trong trường hợp đã kết nối bo mạch nhúng với máy tính và cài đặt các driver cần thiết, bạn sẽ được lựa chọn các ví dụ để làm việc dựa vào đường dẫn File > Example của ứng dụng. Sau đó, có thể bắt đầu viết chương trình bằng cách sử dụng những ví dụ có sẵn để làm việc thoái mái mà phần mềm Arduino IDE cung cấp. Bạn có thể bắt đầu với các ví dụ cơ bản từ 1 đến 10 có sẵn trong phần mềm Arduino để tìm hiểu.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT



Hình 2. 16: Giao diện phần mềm Arduino IDE

2. Phần mềm thiết kế mạch Schematic và PCB

a. EasyEDA

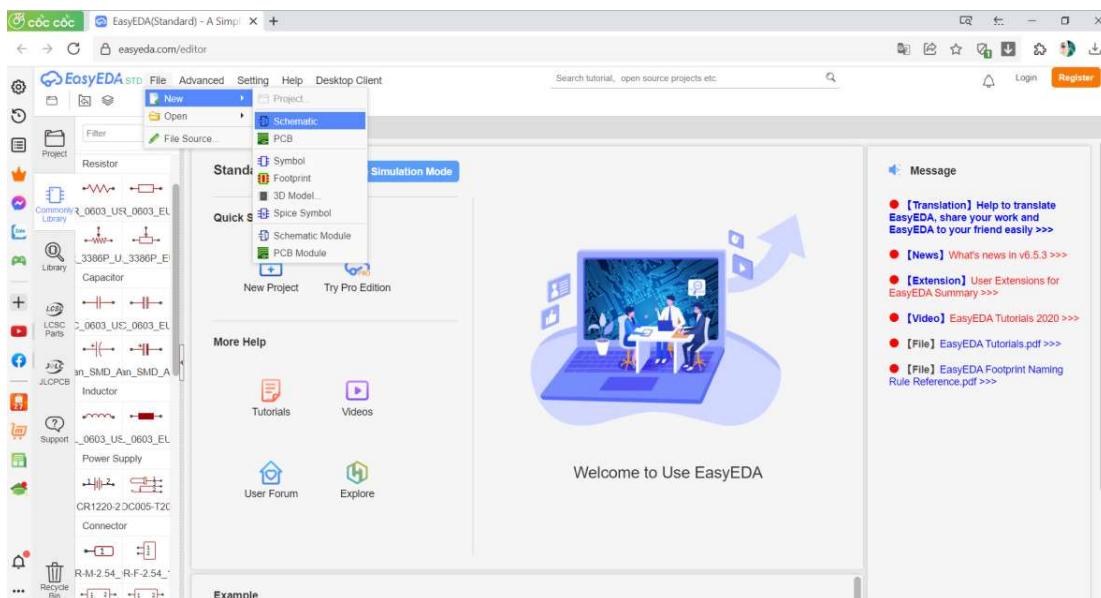
EasyEDA (Easy Electronics Design Automation) là một công cụ thiết kế vi mạch (EDA) miễn phí, không cần cài đặt, những dự án được thiết kế sẽ được lưu trữ trên nền tảng điện toán đám mây, phần mềm được thiết kế để mang đến cho kỹ sư điện, giảng viên, sinh viên kỹ thuật và những người yêu thích điện tử một trải nghiệm thiết kế mạch dễ dàng hơn.

Đây là phần mềm miễn phí mà bất kỳ ai cũng có thể sử dụng mà không cần phải mua bản quyền. Bạn chỉ cần thực hiện quá trình đăng ký và đăng nhập là đã sẵn sàng tạo một dự án cho riêng mình. Các hệ điều hành được hỗ trợ là Windows, Mac và Linux. Trình duyệt được hỗ trợ là Chrome, Internet Explorer, Safari và Firefox.

Chương trình có hơn 70.000 sơ đồ sẵn có trong cơ sở dữ liệu web gồm hơn 15.000 thư viện. Vì ứng dụng được xây dựng trên đám mây nên nó mang lại sự tiện lợi. Một ưu điểm khác đi kèm với ứng dụng đám mây là khả năng tương thích giữa các thiết bị.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

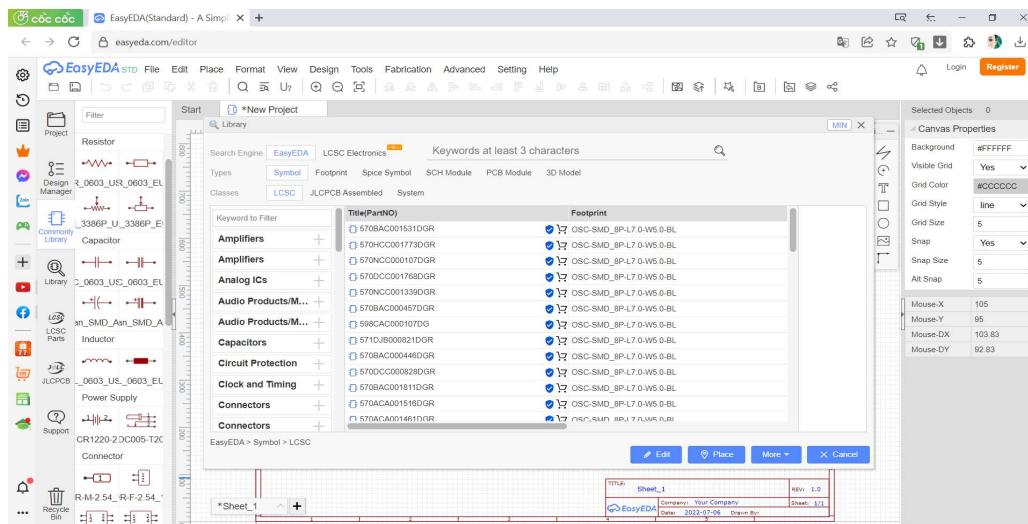
- ❖ Những tính năng nổi bật của phần mềm EasyEDA bao gồm:
 - Có thể chia sẻ dự án của mình lên web EasyEDA.
 - Có thể Download những dự án có sẵn trên web EasyEDA.
 - Hàng triệu thư viện có sẵn miễn phí trên phần mềm.
 - Đọc được các file Altium/Kicad/Eagle, PNG, DXF.
 - Xuất file Gerber chuẩn quốc tế để đặt mạch in.
 - Xuất tài liệu (PDF, PNG, SVG), xuất file nguồn EasyEDA (json), xuất định dạng Altium Designer,...
- ❖ Để tạo một dự án đầu tiên cho riêng mình, ta bằng cách truy cập vào đường link <https://easyeda.com/editor>, sau đó chọn Register để đăng ký tài khoản.
 - Sau khi tạo tài khoản thành công, ta chọn File > New > Schematic.



Hình 2. 17: Tạo dự án đầu tiên để thiết kế mạch

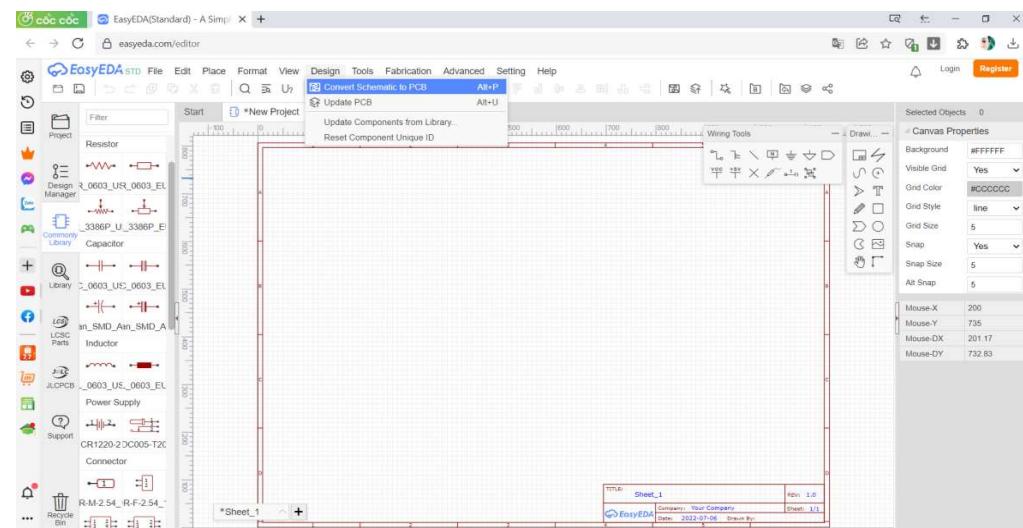
- Sau khi tạo thành công, ta có thể sử dụng thư viện có sẵn của phần mềm trong Commonly Library. Hoặc có thể vào mục Library và bấm vào Keywords để tìm kiếm những thư viện có sẵn trên mạng và download về.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT



Hình 2. 18: Download những thư viện có sẵn trên mạng EasyEDA

- Sau khi thiết kế xong mạch Schematic, ta có thể chuyển qua PCB bằng cách vào Design > Convert Schematic to PCB để thiết kế mạch PCB.



Hình 2. 19: Chuyển từ Schematic sang PCB trên EasyEDA

- Khi đã thiết kế thành công mạch Schematic và PCB, ta có thể xuất file Gerber chuẩn quốc tế để đem ra những nơi nhận làm mạch và thiết kế.

b. Altium Designer

Altium Designer ngày nay đang là một trong những phần mềm vẽ mạch điện tử được ưa chuộng ở Việt Nam. Ngoài việc hỗ trợ tốt cho hoạt động vẽ mạch, Altium còn hỗ trợ tốt trong việc quản lý mạch, trích xuất file ,...

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

Altium Designer cung cấp một ứng dụng kết hợp tất cả công nghệ và chức năng cần thiết cho việc phát triển sản phẩm điện tử hoàn chỉnh, như thiết kế hệ thống ở mức bo mạch và FPGA, phát triển phần mềm nhúng cho FPGA và các bộ xử lý rời rạc, bố trí mạch in (PCB)... Altium Designer thông nhất toàn bộ các quá trình lại và cho phép bạn quản lý được mọi mặt quá trình phát triển hệ thống trong môi trường tích hợp duy nhất. Khả năng đó kết hợp với khả năng quản lý dữ liệu thiết kế hiện đại cho phép người sử dụng Altium Designer tạo ra nhiều hơn những sản phẩm điện tử thông minh, với chi phí sản phẩm thấp hơn và thời gian phát triển ngắn hơn.

❖ Tính năng của phần mềm Altium Designer

- Giao diện quản lý và chỉnh sửa dễ sử dụng, dễ dàng biên dịch và quản lý file.
- Hỗ trợ mạnh mẽ cho việc thiết kế tự động, đi dây tự động theo thuật toán tối ưu.
- Hệ thống các thư viện phong phú, chi tiết bao gồm các linh kiện nhúng, số, tương tự,...
- Mô phỏng mạch PCB 3D đem lại hình ảnh trung thực trong không gian 3 chiều.
- Hỗ trợ thiết kế PCB sang FPGA và ngược lại.



Hình 2. 20: Phần mềm Altium Designer

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

So sánh ưu điểm nhược điểm của hai phần mềm	
Altium Designer	EasyEDA
<p>Ưu điểm:</p> <ul style="list-style-type: none">- Bộ công cụ đa dạng phù hợp với đa số các yêu cầu về mạch in trên thị trường.- Altium hỗ trợ việc sử dụng file 3D của các phần mềm khác để mô phỏng khiến việc thiết kế trở nên trực quan hơn.- Có lượng cộng đồng đông đảo, bộ thư viện việc đầy đủ. <p>Nhược điểm:</p> <ul style="list-style-type: none">- Cấu hình phức tạp, khó cho người mới sử dụng.- Phần mềm nặng, máy yếu dùng rất khó.- Tính năng mô phỏng yếu.	<p>Ưu điểm:</p> <ul style="list-style-type: none">- Có sẵn trực tiếp trên web chỉ cần đăng nhập và tạo một dự án mới.- Lưu trữ trực tiếp trên web mà không phải lo mất file dự án.- Có liên kết với các nhà phân phối với các nhà phân phối linh kiện và nhà sản xuất mạch PCB giúp chúng ta đặt mạch gia công một cách dễ dàng.- Cho phép chia sẻ dự án với mọi người với nhau trên trang trực tuyến. Ta có thể truy cập tải về và sửa đổi nó.- Phần mềm nhẹ, phù hợp cho các máy có cấu hình yếu. <p>Nhược điểm:</p> <ul style="list-style-type: none">- Không được tối ưu khi sử dụng phần mềm Offline.

Bảng 2. 1: So sánh hai phần mềm thiết kế mạch in

➔ Với những tính năng và ưu điểm vượt trội của phần mềm EasyEDA, nhóm đã quyết định tìm hiểu và sử dụng phần mềm EasyEDA để thiết kế mạch Schematic và PCB cho đồ án của mình.

3. React Native

React Native là một framework với mã nguồn mở được phát triển bởi Facebook. React Native được sử dụng để phát triển các ứng dụng di động của Android, IOS và Web bằng cách cho phép các nhà phát triển sử dụng React cùng với môi trường ứng dụng gốc (native).

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

Các nguyên tắc hoạt động của React Native gần giống như React. React Native chạy một quá trình xử lý nền (phiên dịch JavaScript viết bởi các nhà phát triển) trực tiếp trên thiết bị đầu cuối và giao tiếp với nền tảng gốc qua trung gian. Các thành phần React được bao bọc bởi mã gốc và tương tác với API gốc qua mô hình UI và JavaScript của React. Điều này giúp việc phát triển ứng dụng cho nhiều nền tảng trở nên nhanh hơn.

Một ví dụ cơ bản về chương trình Hello World:

```
1 import React, { Component } from 'react';
2 import { AppRegistry, Text } from 'react-native';
3
4 export default class HelloWorldApp extends Component {
5   render() {
6     return (
7       <Text>Hello world!</Text>
8     );
9   }
10 }
11
12 // Bỏ qua dòng này nếu bạn sử dụng công cụ Create React Native App
13 AppRegistry.registerComponent('HelloWorld', () => HelloWorldApp);
14
15 // Code ReactJS có thể được sử dụng từ một thành phần khác bằng câu lệnh sau:
16
17 import HelloWorldApp from './HelloWorldApp';
```

Hình 2. 21: Ví dụ cơ bản sử dụng React Native

Ưu điểm: React Native cùng với Flutter đang là xu hướng lập trình di động hiện nay bởi tính đa nền tảng cũng như tiết kiệm thời gian triển khai dự án. Sau đây là những lợi ích cho việc triển khai dự án như khả năng tái sử dụng code, cộng đồng lớn, mã nguồn mở,...

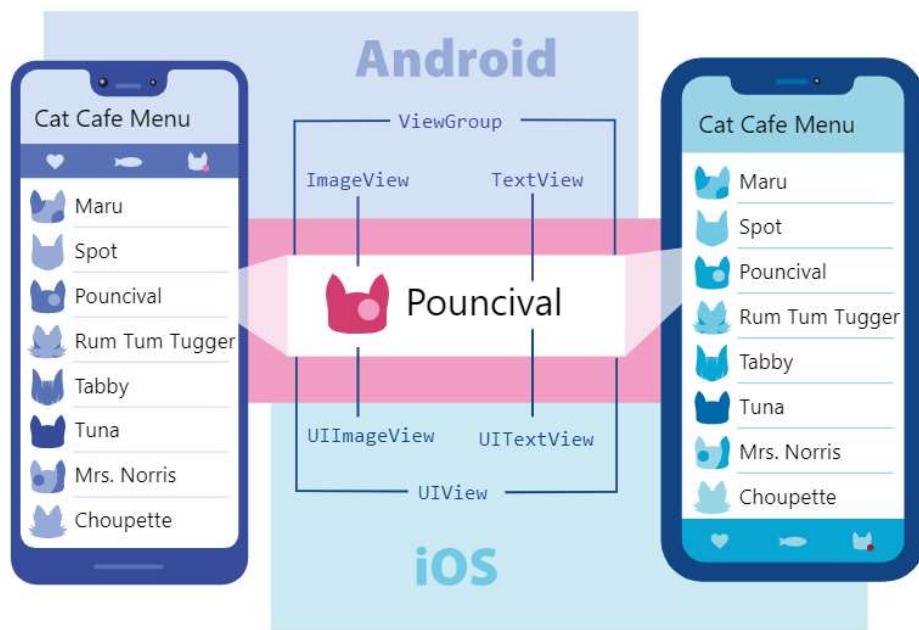
Nhược điểm: Còn thiếu các component quan trọng nhưng dần dần cũng đang có thêm nhiều cập nhật mới, Không build được ứng dụng IOS trên Window và Linux: do yêu cầu từ Apple, mọi ứng dụng IOS cần được sử dụng nhiều native libs, cert...từ Xcode, không dùng để viết game có tính đồ họa và cách chơi phức tạp,...

Lộ trình học React Native:

- Kiến thức lập trình web căn bản : kỹ thuật lập trình, Cơ sở dữ liệu, HTML, CSS, JavaScript...
- ES6 và Javascript nâng cao.
- Nodejs: nhiều khái niệm cần biết như: NPM, các lệnh như npm install, npm install – save –dev, npm start,... Promises / Callbacks / Async Await.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

- React cần biết các khái niệm: Components (Class với Functional), Kiểm soát các thành phần, Handlers, this.setState và this.props trong React , Life cycle methods (Một chuỗi các sự kiện xảy ra từ khi thành phần React ra đời cho đến khi nó chết.), Fetch/Axios để gọi APIs.
 - Redux với React.
 - Flexbox: Flexbox rất hữu ích trong việc thiết kế giao diện người dùng và thành phần quan trọng của Lộ trình học React Native.
 - Một số kiến thức khác nên biết: redux-thunk, redux-saga, LESS, SASS, React hooks, TypeScript, Proptypes. Bất kỳ cơ sở dữ liệu nào để kết nối ứng dụng của bạn, mới học thì tốt nhất là Firebase (đây là một cloud service của Google để xác thực, cơ sở dữ liệu, lưu trữ...).
- ❖ Core components and Native Components.



Hình 2. 22: Chế độ xem được sử dụng trong các ứng dụng Android và IOS

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

THÀNH PHẦN GIAO DIỆN NGƯỜI DÙNG REACT NATIVE	CHẾ ĐỘ XEM ANDROID	CHẾ ĐỘ XEM IOS	WEB ANALOG	SỰ MÔ TẢ
<View>	<ViewGroup>	<UIView>	Không cuộn <div>	Một vùng chứa hỗ trợ bối cảnh với flexbox, style, một số điều khiển cảm ứng và điều khiển trợ năng
<Text>	<TextView>	<UITextView>	<p>	Hiển thị, kiểu và lồng các chuỗi văn bản và thậm chí xử lý các sự kiện chạm
<Image>	<ImageView>	<UIImageView>		Hiển thị các loại hình ảnh khác nhau
<ScrollView>	<ScrollView>	<UIScrollView>	<div>	Một vùng chứa cuộn chung có thể chứa nhiều thành phần và chế độ xem
<TextInput>	<EditText>	<UITextField>	<input type="text">	Cho phép người dùng nhập văn bản

Hình 2. 23: Components của React Native

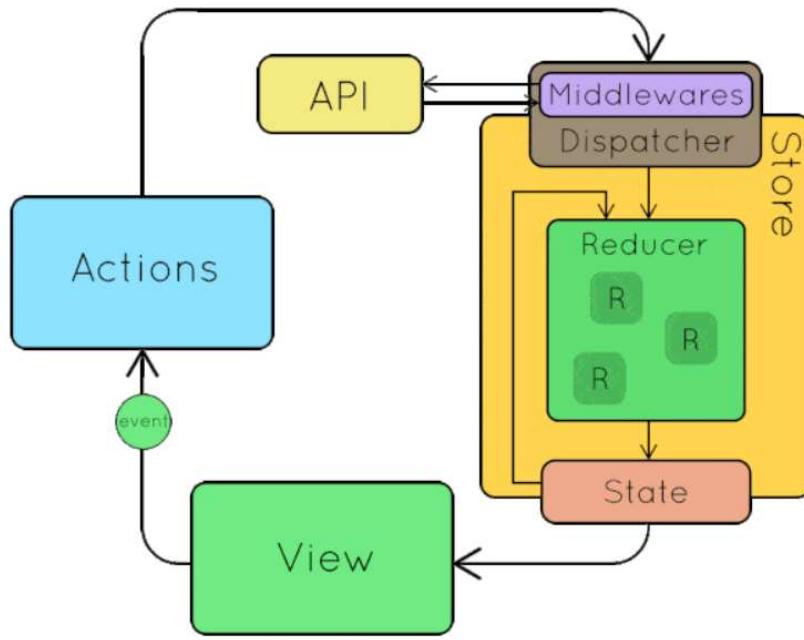
4. Redux

Redux là một predictable state management tool cho các ứng dụng Javascript. Nó giúp bạn viết các ứng dụng hoạt động một cách nhất quán, chạy trong các môi trường khác nhau (client, server, and native) và dễ dàng để test. Redux ra đời lấy cảm hứng từ tư tưởng của ngôn ngữ Elm và kiến trúc Flux của Facebook. Do vậy Redux thường dùng kết hợp với React.

Hầu hết các lib như React, Angular, etc được built theo một cách sao cho các components đến việc quản lý nội bộ các state của chúng mà không cần bất kỳ một thư viện hoặc tool nào từ bên ngoài.

Nó sẽ hoạt động tốt với các ứng dụng có ít components nhưng khi ứng dụng trở lên lớn hơn thì việc quản lý states được chia sẻ qua các components sẽ biến thành các công việc lặt nhặt.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT



Hình 2. 24: Nguyên lý vận hành của Redux

The screenshot shows the Xcode interface with the following details:

- EXPLORER**: Shows the project structure under `SOCIAL-APP-MASTER`, including files like `actions.js`, `index.js`, `reducers.js`, `styles.ts`, `@types.ts`, `.buckconfig`, `.editorconfig`, `.eslintrc.js`, `.flowconfig`, `.gitattributes`, `.gitignore`, `prettierrc.js`, `.watchmanconfig`, `App.js`, `app.json`, `babel.config.js`, `index.js`, `metro.config.js`, `package-lock.json`, `package.json`, `react-native.config.js`, `README.md`, `yarn-error.log`, and `yarn.lock`.
- actions.js** is selected in the Explorer.
- EDITOR**: Displays the content of `actions.js` (shown below).
- TERMINAL**: Shows the output of the command `yarn run v1.22.18` and the message "Building the app.....".

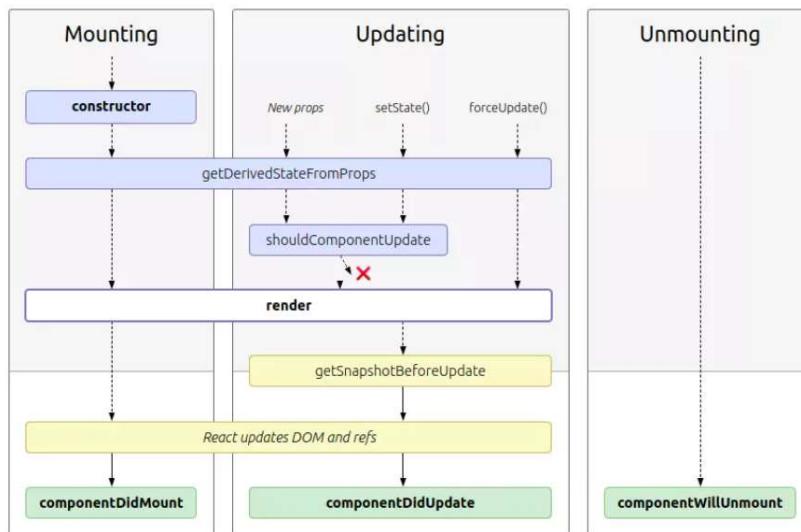
```
src > store > JS actions.js > @@ init > ⓘ <function>
1 import AsyncStorage from "@react-native-async-storage/async-storage";
2 export const Init = () => {
3   return async dispatch => {
4     let token = await AsyncStorage.getItem('token');
5     if (token !== null) {
6       console.log('token fetched');
7       dispatch({
8         type: 'LOGIN',
9         payload: token,
10      })
11    }
12  }
13
14
15 export const Login = (email, password, authen) => {
16   return async dispatch => {
17     let token = null;
18     if (email === '' && password === '') {
19       token = authen;
20       // here we can use login api to get token and then store it
21       await AsyncStorage.setItem('token', token);
22       //console.log('token stored');
23     }
24     dispatch({
25       type: 'LOGIN',
26       payload: token,
27     })
28   }
29 }
30
31
```

PROBLEMS	OUTPUT	DEBUG CONSOLE	TERMINAL
			Installing React-callInvoker 0.64.0 (was 0.64.2) Installing React-cxxreact 0.64.0 (was 0.64.2) Installing React-jsi 0.64.0 (was 0.64.2) Installing React-jsserverexec 0.64.0 (was 0.64.2) Installing React-mtexec 0.64.0 (was 0.64.2) Installing React-perflogger 0.64.0 (was 0.64.2) Installing React-runtimeexecutor 0.64.0 (was 0.64.2) Installing ReactCommon 0.64.0 (was 0.64.2) Installing SwiftlyJSON 5.0.0 Installing Yoga 1.14.0 Installing react-native-svg-charts-wrapper (0.5.8) Generating Pods project Integrating client project Pod installation complete! There are 69 dependencies from the Podfile and 76 total pods installed. USER->no-MacBook-Pro:ios user\$ cd .. USER->no-MacBook-Pro:social-app-master user\$ yarn ios yarn run v1.22.18 \$ tsc --noEmit --force info Found Xcode workspace "FeeuleeCM.xcworkspace" info Launching iPhone 12 (iOS 14.5) info Building (using "xcodebuild -workspace FeeuleeCM.xcworkspace -configuration Debug -scheme

Hình 2. 25: Ứng dụng Redux vào logout login

5. React Native Lifecycle

Đối với Mobile Developer, chắc hẳn khái niệm lifecycle hay còn gọi là vòng đời của ứng dụng không còn xa lạ với các bạn nữa. Thông thường, vòng đời ứng dụng Android bắt đầu từ khi ứng dụng được hiển thị lên màn hình cho đến khi thoát khỏi ứng dụng.



Hình 2. 26: Vòng đời trong React Native

6. Xcode

Xcode là công cụ chính để lập trình Iphone/Ipad. Tất cả những gì mới nhất của Apple đều được tích hợp trong Xcode, IOS 6 hay những gì mới nhất của hệ điều hành. Apple rất ưu ái cho các lập trình viên những người đã làm cho hệ sinh thái của Apple trở nên đa dạng.

Xcode còn tích hợp cả máy ảo (Simulation) để cho các lập trình viên thử nghiệm các sản phẩm không cần thiết bị thật và môi trường SandBox để kiểm tra các mua bán (in game purchase). Xcode là công cụ dễ dàng và đơn giản nhất để người sử dụng phát triển một phần mềm Iphone hay Ipad. [5] link tải Xcode.

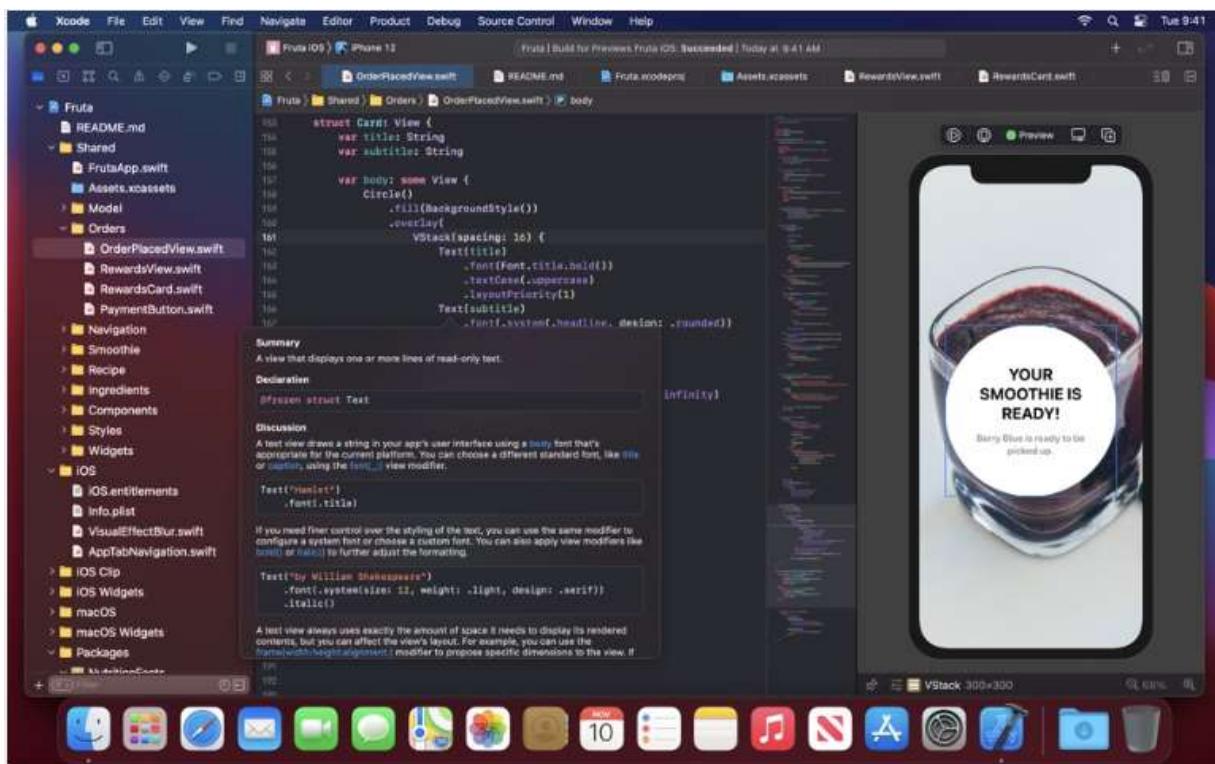
Ưu điểm:

- Sử dụng IB (interface builer) chính là cách ngắn nhất và đơn giản nhất để chương trình không chứa những lỗi sai và nhanh chóng thuận lợi hơn rất nhiều so với việc viết Code bằng tay.
- Nhanh chóng nếu làm form với số lượng ít.
- Hạn chế tối đa số lượng code phải đánh vào.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

Nhược điểm:

- Không sử dụng được ở những form động (dynamicform) những form mà sự biến đổi dựa trên những thông số truyền vào.
- Chậm nếu phải làm rất nhiều form.



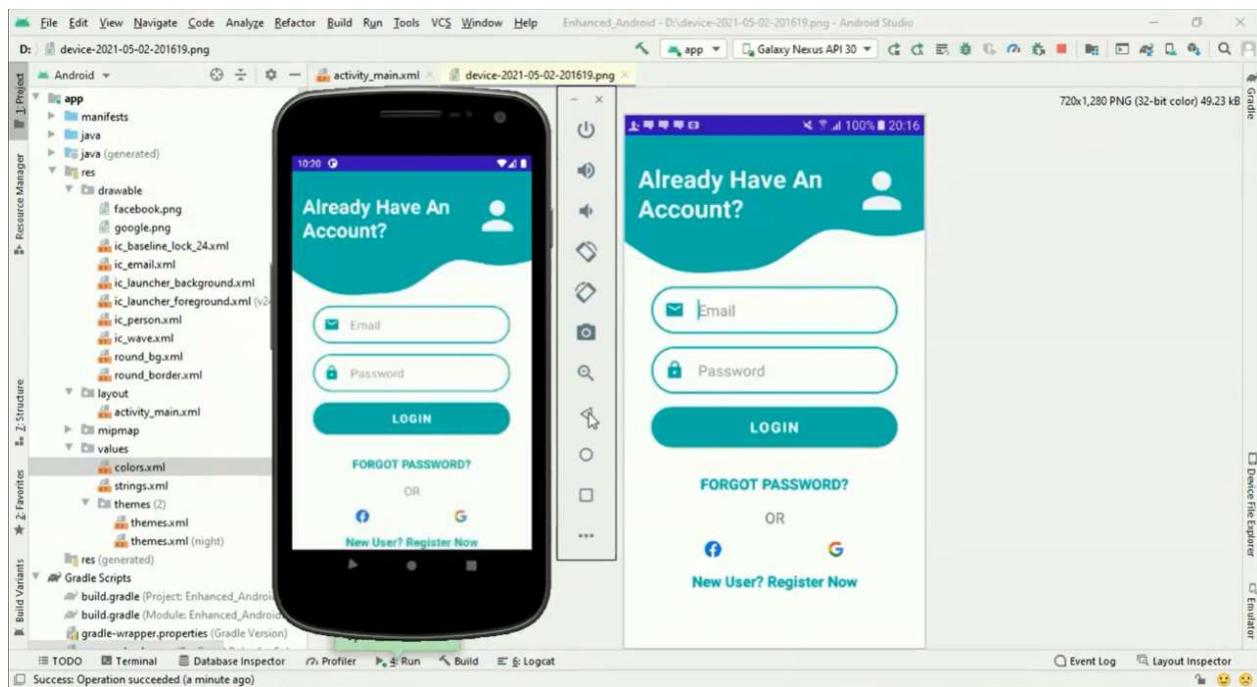
Hình 2. 27: Giao diện Xcode

7. Android Studio

Android Studio là IDE được sử dụng trong phát triển ứng dụng Android dựa trên IntelliJ IDEA. Chức năng chính của Android Studio là cung cấp các giao diện giúp người dùng có thể tạo các ứng dụng và xử lý các công cụ file phức tạp. Ngôn ngữ lập trình được sử dụng trong Android Studio là Java và nó sẽ được cài đặt sẵn trên thiết bị của người sử dụng.

Khi sử dụng Android Studio thì người sử dụng chỉ cần viết, chỉnh sửa và lưu trữ chúng trên các dự án của mình và các file nằm trong dự án đó. Đồng thời, Android Studio còn cung cấp quyền truy cập vào Android SDK. [6] link tải Android Studio.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT



Hình 2. 28: Giao diện Android Studio

8. NodeJS

NodeJS Là một môi trường runtime chạy Javacripts đa nền tảng và có mã nguồn mở, được sử dụng để chạy các ứng dụng web bên ngoài trình duyệt của client. Nền tảng này được phát triển bởi Ryan Dahl vào năm 2009, được xem là một giải pháp hoàn hảo cho các ứng dụng sử dụng nhiều dữ liệu nhờ vào mô hình hướng sự kiện (event-driven) không đồng bộ.

Ưu điểm:

- IO hướng sự kiện không đồng bộ, cho phép xử lý nhiều yêu cầu đồng thời.
- Chia sẻ cùng code ở cả phía client và server.
- NPM(Node Package Manager) và module Node đang ngày càng phát triển mạnh mẽ.

Nhược điểm:

- Không có khả năng mở rộng, vì vậy không thể tận dụng lợi thế mô hình đa lõi trong các phần cứng cấp server hiện nay.
- Khó thao tác với cơ sở dữ liệu quan hệ.
- Cần có kiến thức tốt về JavaScript.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

9. ES6

ES6 là chữ viết tắt của ECMAScript 6 được làm tiêu chuẩn của ngôn ngữ Javascript. khá nhiều trình duyệt Browser ra đời và nếu mỗi Browser lại có cách chạy Javascript khác nhau thì các trang web không thể hoạt động trên tất cả các trình duyệt đó được, vì vậy cần có một chuẩn chung để bắt buộc các browser phải phát triển dựa theo chuẩn đó.

Các chức năng của ES6:

- Arrow function: Bạn có thể tạo hàm bằng cách sử dụng dấu mũi tên =>.
- Block Scoped: Định nghĩa biến với từ khóa let, cách định nghĩa này thì biến chỉ tồn tại trong phạm vi khối của nó (Block Scope).
- Destructuring Assignments: Bạn có thể khởi tạo các biến từ một mảng bằng một dòng code đơn giản.
- Default Parameters: Bạn có thể gán giá trị mặc định cho các tham số.

10. ReactJS

ReactJS là một thư viện Javascript dùng để xây dựng giao diện người dùng, React hỗ trợ việc xây dựng những thành phần (components) UI có tính tương tác cao, có trạng thái và có thể sử dụng lại được, React được xây dựng xung quanh các component.

```
npm install -g create-react-app  
create-react-app my-app  
  
cd my-app  
npm start
```

**kết quả **



Hình 2. 29: Cài đặt ReactJS

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

IV. GOOGLE CLOUD FIREBASE

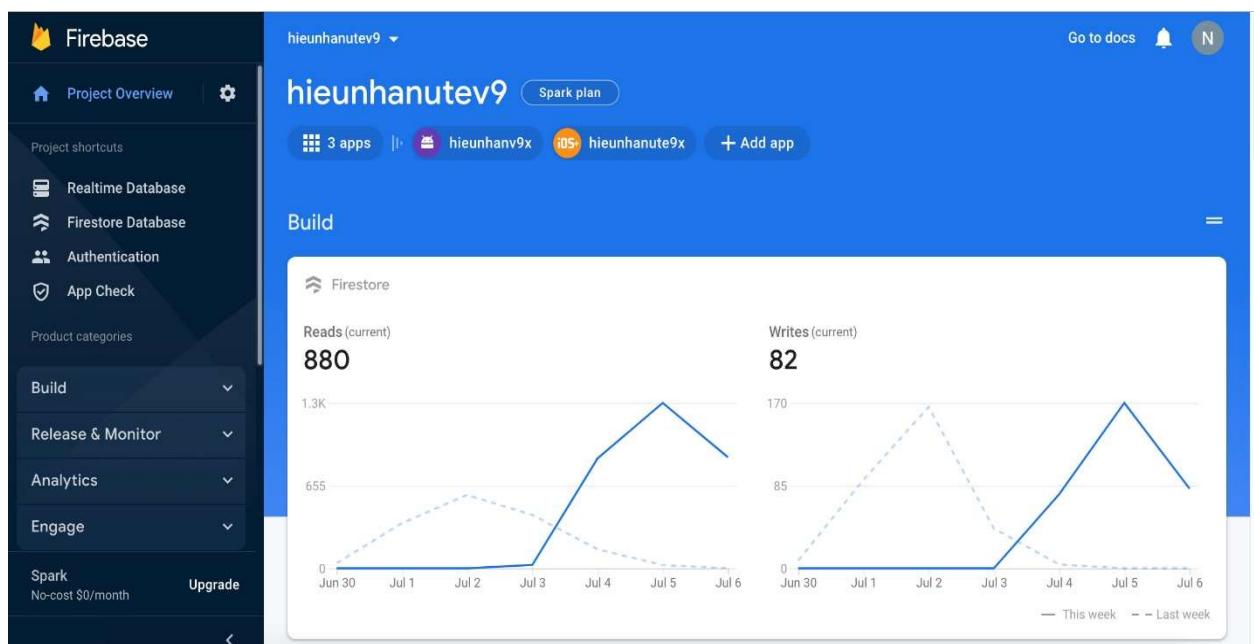
1. Giới thiệu về Firebase

Firebase là một nền tảng được sở hữu bởi Google giúp chúng ta phát triển các ứng dụng như ứng dụng di động và web. Firebase cung cấp rất nhiều công cụ và dịch vụ tiện ích để phát triển và tạo nên một ứng dụng chất lượng. Điều đó rút ngắn thời gian phát triển và giúp ứng dụng sớm được hoàn thành.

Firebase cung cấp cho người sử dụng các dịch vụ cơ sở dữ liệu hoạt động trên nền tảng đám mây với hệ thống máy chủ cực kỳ mạnh mẽ và ổn định của Google. Chức năng chính của Firebase là giúp người dùng lập trình các ứng dụng, phần mềm trên các nền tảng web hoặc di động bằng cách đơn giản hóa các thao tác với cơ sở dữ liệu.

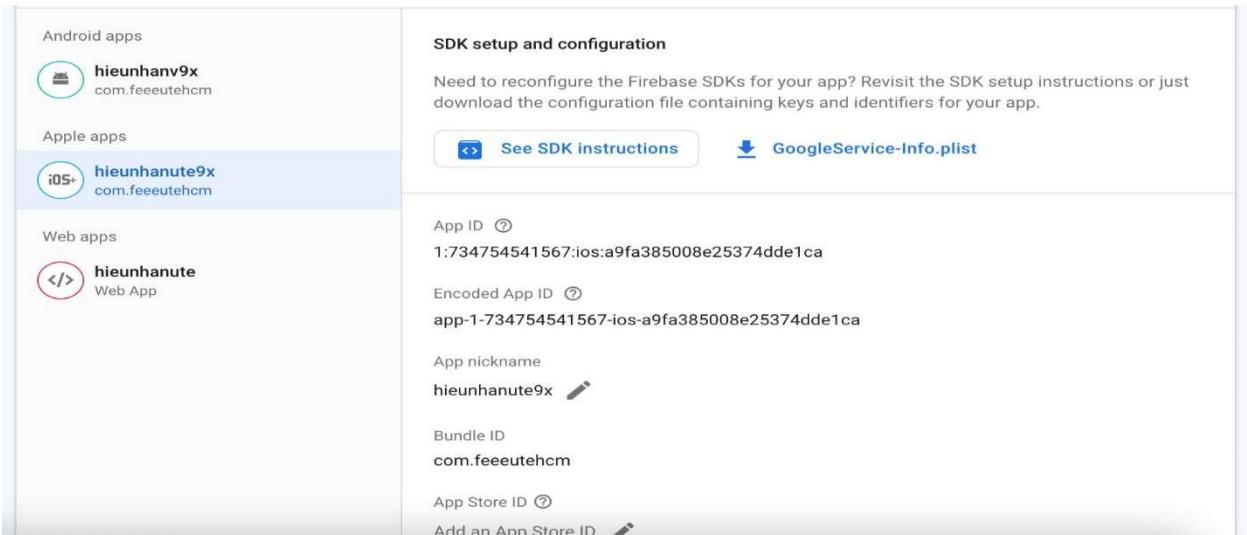
Người có thể tạo ra những ứng dụng real-time như ứng dụng chat và cùng nhiều tính năng như xác thực người dùng,... Người dùng có thể dùng Firebase giống như phần backend của app.

Các dịch vụ của Firebase hoàn toàn được sử dụng miễn phí, tuy nhiên người sử dụng cần phải trả thêm tiền nếu muốn nâng cấp lên. Điều này nên cân nhắc nếu muốn xây dựng một ứng dụng lớn sử dụng phần backend là Firebase, vì giá thành khi nâng cấp còn khá đắt đỏ so với việc xây dựng backend truyền thống.



Hình 2. 30: Giao diện Firebase

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT



Hình 2. 31: Thiết lập và cấu hình SDK

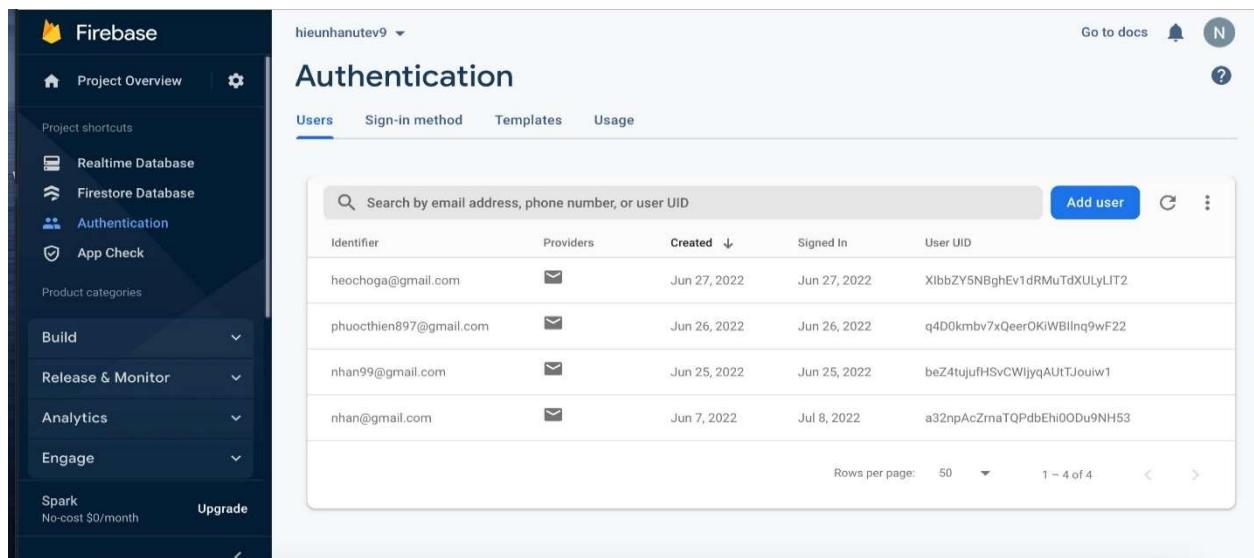
2. Firebase Authentication

Hầu hết các ứng dụng cần biết danh tính của người dùng. Việc biết danh tính của người dùng cho phép ứng dụng lưu trữ dữ liệu người dùng một cách an toàn trên đám mây và cung cấp trải nghiệm được cá nhân hóa giống nhau trên tất cả các thiết bị của người dùng.

Firebase Authentication cung cấp các dịch vụ phụ trợ, SDK dễ sử dụng và thư viện giao diện người dùng được tạo sẵn để xác thực người dùng với ứng dụng của người sử dụng. Nó hỗ trợ xác thực bằng mật khẩu, số điện thoại, các nhà cung cấp danh tính liên hợp phổ biến như Google, Facebook và Twitter,...

Để đăng nhập một người dùng vào ứng dụng của người sử dụng, trước tiên người sử dụng nhận được thông tin xác thực từ người dùng đó. Các thông tin xác thực này có thể là địa chỉ email và mật khẩu của người dùng hoặc mã thông báo OAuth từ nhà cung cấp danh tính được liên kết. Sau đó, người sử dụng sẽ chuyển các thông tin xác thực này cho Firebase Authentication SDK. Các dịch vụ backend của Firebase sau đó sẽ xác minh các thông tin đăng nhập đó và trả lại phản hồi cho khách hàng. Sau khi đăng nhập thành công, người sử dụng có thể truy cập thông tin hồ sơ cơ bản của người dùng và có thể kiểm soát quyền truy cập của người dùng vào dữ liệu được lưu trữ trong các sản phẩm Firebase khác. Người sử dụng cũng có thể sử dụng mã xác thực được cung cấp để xác minh danh tính của người dùng trong các dịch vụ phụ trợ.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT



The screenshot shows the Firebase console's Authentication section. On the left, there's a sidebar with project settings like Project Overview, Realtime Database, Firestore Database, Authentication (which is selected), App Check, and various build and release monitoring options. The main area is titled 'Authentication' and shows a table of users. The columns are 'Identifier', 'Providers', 'Created', 'Signed In', and 'User UID'. There are four entries:

Identifier	Providers	Created	Signed In	User UID
heochoga@gmail.com	✉️	Jun 27, 2022	Jun 27, 2022	XlibbZY5NBghEv1dRMuTdXULyLIT2
phuocthien897@gmail.com	✉️	Jun 26, 2022	Jun 26, 2022	q4D0kmbv7xQeerOKIWBIlnq9wF22
nhan99@gmail.com	✉️	Jun 25, 2022	Jun 25, 2022	beZ4tujufHSvCWljyqAUTJouiw1
nhan@gmail.com	✉️	Jun 7, 2022	Jul 8, 2022	a32npAcZrmaTQPdbEhi0ODu9NH53

At the bottom right of the table, there are buttons for 'Rows per page' (set to 50) and navigation arrows. The top right of the main area has links for 'Go to docs', a bell icon, and a question mark.

Hình 2. 32: Giao diện Firebase Authentication

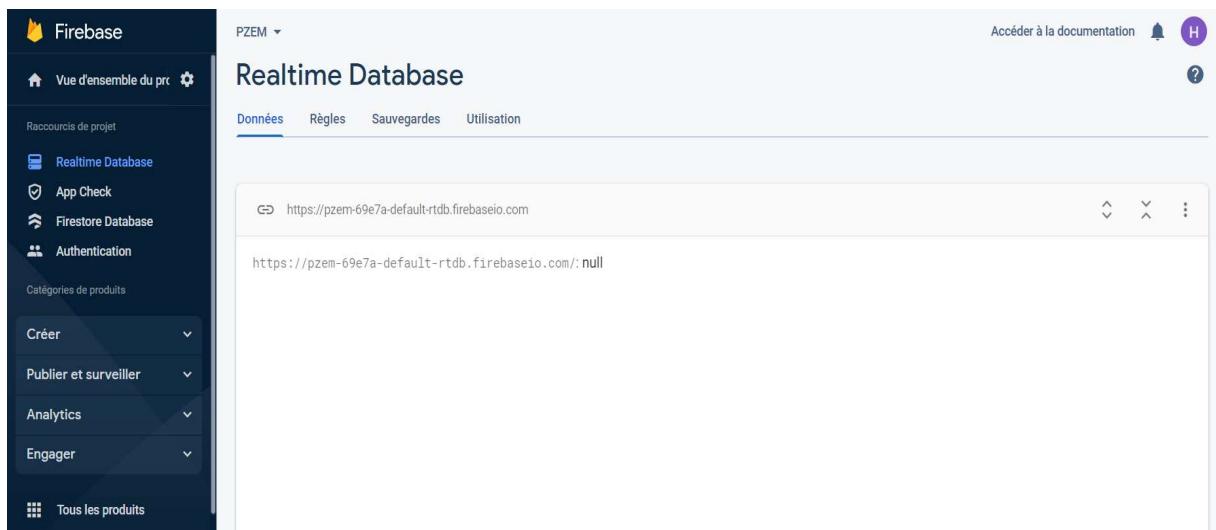
Các tính năng: Firebase SDK Authentication:

- Xác thực dựa trên email và mật khẩu.
- Xác thực người dùng bằng cách tích hợp với các nhà cung cấp danh tính được liên kết. Firebase SDK Authentication cung cấp các phương pháp cho phép người dùng đăng nhập bằng tài khoản Google, Facebook, Twitter và GitHub của họ.
- Xác thực người dùng bằng cách gửi tin nhắn SMS đến điện thoại của họ.
- Kết nối hệ thống đăng nhập hiện có của ứng dụng người sử dụng với Firebase SDK Authentication và có quyền truy cập vào Firebase Realtime Database và các dịch vụ Firebase khác.
- Sử dụng các tính năng yêu cầu xác thực mà không yêu cầu người dùng đăng nhập trước bằng cách tạo tài khoản ẩn danh tạm thời. Nếu sau đó người dùng chọn đăng ký, người sử dụng có thể nâng cấp tài khoản ẩn danh lên tài khoản thông thường.

2. Firebase Realtime Database

Firebase Realtime database là cơ sở dữ liệu được lưu trữ trên đám mây. Dữ liệu được lưu trữ dưới dạng JSON và được đồng bộ hóa trong thời gian thực cho mọi máy khách được kết nối. Khi bạn tạo các ứng dụng đa nền tảng với các ứng dụng Apple, Android và JavaScript SDK của Firebase, tất cả khách hàng của người sử dụng đều chia sẻ một cơ sở dữ liệu thời gian thực và tự động nhận các bản cập nhật với dữ liệu mới nhất.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT



Hình 2. 33: Giao diện Firebase Realtime Database

Các tính năng:

- Realtime: Cơ sở dữ liệu thời gian thực của Firebase sử dụng đồng bộ hóa dữ liệu, mỗi khi dữ liệu thay đổi, mọi thiết bị được kết nối sẽ nhận được bản cập nhật đó trong vòng mili giây.
- Offline: Các ứng dụng Firebase vẫn phản hồi ngay cả khi ngoại tuyến vì Firebase Realtime Database SDK sẽ duy trì dữ liệu của người sử dụng. Khi kết nối được thiết lập lại, thiết bị khách sẽ nhận được bất kỳ thay đổi nào mà nó đã bỏ qua, đồng bộ hóa nó với trạng thái máy chủ hiện tại.
- Có thể truy cập từ thiết bị khách: Firebase Realtime database có thể được truy cập trực tiếp từ thiết bị di động hoặc trình duyệt web.
- Quy mô trên nhiều cơ sở dữ liệu: Người sử dụng có thể hỗ trợ cung cấp dữ liệu của ứng dụng trên quy mô lớn bằng cách tách dữ liệu của bạn trên nhiều phiên bản cơ sở dữ liệu trong cùng một dự án Firebase. Hợp lý hóa việc xác thực với Firebase trong dự án của bạn và xác thực người dùng trên các phiên bản cơ sở dữ liệu của bạn. Kiểm soát quyền truy cập vào dữ liệu trong từng cơ sở dữ liệu bằng quy tắc cơ sở dữ liệu thời gian thực của Firebase tùy chỉnh cho từng phiên bản cơ sở dữ liệu.

3. Cloud Firestore

Cloud Firestore là một cơ sở dữ liệu linh hoạt, có thể mở rộng để phát triển thiết bị di động, web và máy chủ từ Firebase và Google Cloud. Giống như Cơ sở dữ liệu thời gian thực của Firebase, nó giữ cho dữ liệu của bạn được đồng bộ hóa trên các ứng dụng khách thông qua trình xử lý thời gian thực và cung cấp hỗ trợ ngoại tuyến cho thiết bị di động và web để bạn có thể tạo các ứng dụng đáp ứng hoạt động bất kể độ trễ mạng hoặc kết nối Internet. Cloud Firestore cũng cung cấp khả năng tích hợp liền mạch với các sản phẩm Firebase và Google Cloud khác, bao gồm cả chức năng Cloud.

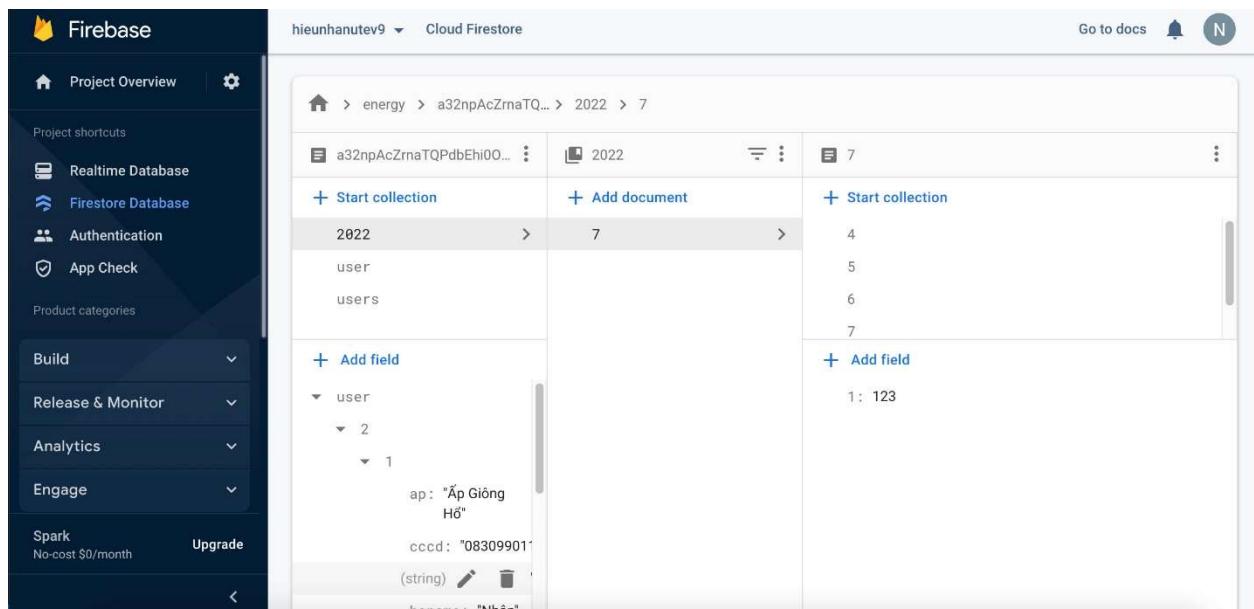
Cloud Firestore là cơ sở dữ liệu NoSQL, được lưu trữ trên đám mây mà các ứng dụng web của Apple, Android và web của người sử dụng có thể truy cập trực tiếp thông qua SDK gốc. Cloud Firestore cũng có sẵn trong các SDK Node.js, Java, Python, Unity, C ++ và Go, ngoài ra còn có API REST và RPC.

Theo mô hình dữ liệu NoSQL của Cloud Firestore, bạn lưu trữ dữ liệu trong các tài liệu có chứa các trường tới các giá trị. Các tài liệu này được lưu trữ trong các bộ sưu tập, là các vùng chứa tài liệu của bạn mà bạn có thể sử dụng để sắp xếp dữ liệu và xây dựng các truy vấn. Tài liệu hỗ trợ nhiều kiểu dữ liệu khác nhau, từ chuỗi và số đơn giản, đến các đối tượng lồng nhau, phức tạp. Bạn cũng có thể tạo các bộ sưu tập con trong tài liệu và xây dựng cấu trúc dữ liệu phân cấp mở rộng quy mô khi cơ sở dữ liệu của bạn phát triển. Mô hình dữ liệu Cloud Firestore hỗ trợ bất kỳ cấu trúc dữ liệu nào hoạt động tốt nhất cho ứng dụng của bạn.

Ngoài ra, truy vấn trong Cloud Firestore rất rõ ràng, hiệu quả và linh hoạt. Tạo các truy vấn ngắn để truy xuất dữ liệu ở cấp độ tài liệu mà không cần truy xuất toàn bộ bộ sưu tập hoặc bất kỳ bộ sưu tập con lồng nhau nào. Thêm sắp xếp, lọc và giới hạn cho các truy vấn của bạn để phân kết quả. Để giữ cho dữ liệu trong các ứng dụng của bạn luôn cập nhật mà không cần truy xuất toàn bộ cơ sở dữ liệu mỗi khi cập nhật xảy ra.

Bảo vệ quyền truy cập vào dữ liệu của bạn trong Cloud Firestore với Xác thực Firebase và Quy tắc bảo mật trên Cloud Firestore cho các nền tảng Android, Apple và JavaScript hoặc quản lý danh tính và truy cập (IAM) cho các ngôn ngữ phía máy chủ.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT



The screenshot shows the Firebase Cloud Firestore interface. On the left, there's a sidebar with project settings like Project Overview, Realtime Database, Firestore Database, Authentication, App Check, and various build and release monitoring options. The main area displays a hierarchical document structure under the path energy > a32npAcZrnaTQ... > 2022 > 7. The document '7' contains a collection named 'user' which has two documents, '2' and '1'. Document '2' has fields 'ap' (value: "Áp Giồng Hồ") and 'cccd' (value: "083099011"). Document '1' has a field '(string)' with value '123'. There are also buttons for 'Start collection', 'Add document', and 'Add field'.

Hình 2. 34: Giao diện Cloud Firestore

Các tính năng:

- Tính linh hoạt: mô hình dữ liệu Cloud Firestore hỗ trợ cấu trúc dữ liệu phân cấp, linh hoạt. Lưu trữ dữ liệu của bạn trong các tài liệu, được sắp xếp thành các bộ sưu tập.
- Truy vấn: Trong Cloud Firestore, bạn có thể sử dụng các truy vấn để truy xuất các tài liệu riêng lẻ, cụ thể hoặc để truy xuất tất cả các tài liệu trong một bộ sưu tập phù hợp với các tham số truy vấn của bạn. Các truy vấn của bạn có thể bao gồm nhiều bộ lọc theo chuỗi và kết hợp lọc và sắp xếp. Chúng cũng được lập chỉ mục theo mặc định, vì vậy hiệu suất truy vấn tỷ lệ thuận với kích thước của tập kết quả, không phải tập dữ liệu của bạn.
- Hỗ trợ ngoại tuyến: Cloud Firestore lưu trữ dữ liệu mà ứng dụng của bạn đang sử dụng, vì vậy ứng dụng có thể viết, đọc, nghe và truy vấn dữ liệu ngay cả khi thiết bị ngoại tuyến. Khi thiết bị trực tuyến trở lại, Cloud Firestore sẽ đồng bộ hóa mọi thay đổi cục bộ trở lại Cloud Firestore.
- Được thiết kế để mở rộng quy mô: Cloud Firestore mang đến cho bạn cơ sở hạ tầng tốt nhất của Google Cloud: sao chép dữ liệu đa vùng tự động, đảm bảo tính nhất quán,...

4. So sánh Firebase Realtime Database và Cloud Firestore

Firebase Realtime Database	Cloud Firestore
<p>Mô hình giữ liệu: Lưu trữ dữ liệu dưới dạng chuỗi JSON lớn.</p> <ul style="list-style-type: none"> - Dữ liệu đơn giản rất dễ lưu trữ. - Dữ liệu phân cấp, phức tạp khó tổ chức trên quy mô lớn. 	<p>Mô hình giữ liệu: Lưu trữ dữ liệu dưới dạng bộ sưu tập tài liệu.</p> <ul style="list-style-type: none"> - Dữ liệu đơn giản dễ lưu trữ trong các tài liệu. - Dữ liệu phân cấp, phức tạp dễ tổ chức hơn trên quy mô lớn, bằng cách sử dụng các bộ sưu tập con. - Yêu cầu ít chuẩn hóa và làm mượt dữ liệu hơn.
<p>Độ tin cậy và hiệu suất: Firebase Realtime Database là một giải pháp cho một tệp dữ liệu.</p> <ul style="list-style-type: none"> - Cơ sở dữ liệu được giới hạn trong phạm vi khả dụng của một tệp dữ liệu - Độ trễ thấp, tùy chọn để đồng bộ hóa trạng thái liên tục. 	<p>Độ tin cậy và hiệu suất: Cloud Firestore là một giải pháp cho nhiều tệp dữ liệu, có thể mở rộng quy mô tự động.</p> <p>Lưu trữ dữ liệu của bạn trên nhiều trung tâm dữ liệu ở các tệp riêng biệt, đảm bảo khả năng mở rộng với quy mô lớn và độ tin cậy cao.</p>
<p>Truy vấn:</p> <p>Các truy vấn sâu theo mặc định: chúng luôn trả về toàn bộ dữ liệu con.</p> <p>Các truy vấn có thể truy cập dữ liệu ở bất kỳ mức độ chi tiết nào, tùy thuộc vào các giá trị riêng lẻ trong chuỗi JSON.</p> <p>Truy vấn không yêu cầu chỉ mục; tuy nhiên, hiệu suất của các truy vấn nhất định sẽ giảm khi tập dữ liệu của bạn phát triển.</p>	<p>Truy vấn:</p> <p>Các truy vấn ngắn: chúng chỉ trả về tài liệu trong một bộ sưu tập hoặc nhóm bộ sưu tập cụ thể và không trả về dữ liệu bộ sưu tập con. Các truy vấn phải luôn trả về toàn bộ tài liệu.</p> <p>Các truy vấn được lập chỉ mục theo mặc định: Hiệu suất truy vấn tỷ lệ thuận với kích thước của tập kết quả.</p>

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

Hỗ trợ thời gian thực và ngoại tuyến: cho ứng dụng Apple và Android.	Hỗ trợ thời gian thực và ngoại tuyến: cho ứng dụng khách, Apple, Android và web.
--	--

Bảng 2. 2: So sánh hai cơ sở dữ liệu

➔ Với việc sử dụng nhiều tập tài liệu khác nhau theo như ngày, tháng, năm để tổng hợp các dữ liệu điện áp, dòng điện, công suất, công suất tiêu thụ,... để đưa lên App thì việc sử dụng cơ sở dữ liệu Cloud Firestore là hợp lý nhất cho đồ án này.

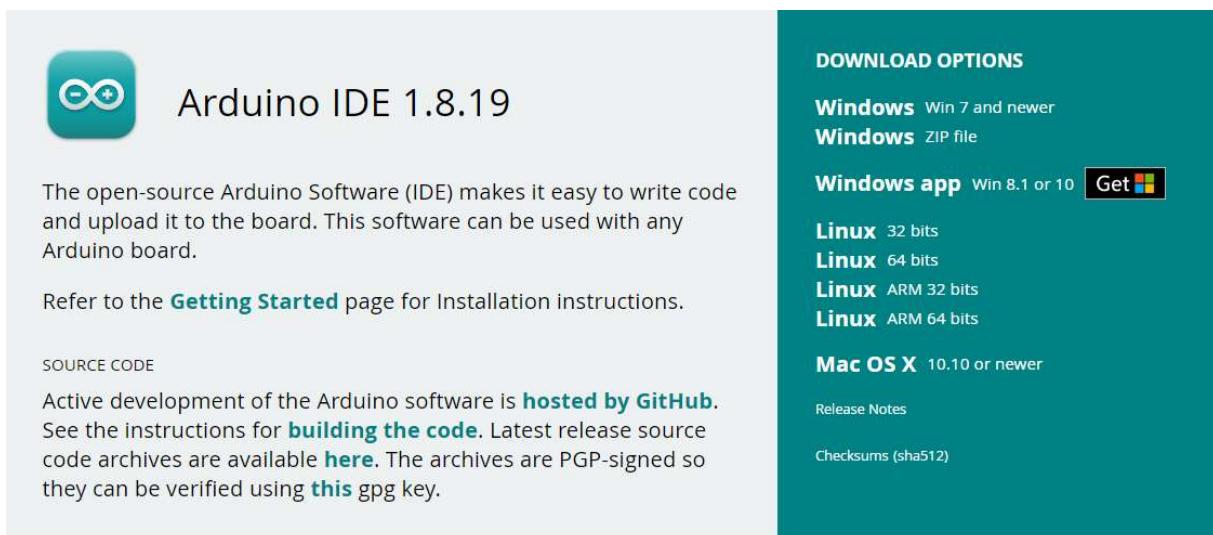
CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

I. PHẦN NHÚNG

1. Cài đặt Arduino IDE

a. Cài đặt phần mềm

Để cài đặt phần mềm Arduino IDE, ta truy cập vào đường link [4] để tải, ở đây luôn được update các phiên bản mới nhất và là trang chủ chính thức của phần mềm Arduino IDE. Phần mềm hỗ trợ các hệ điều hành mới nhất hiện nay như Windows, MacOS, Linux,...



Hình 3. 1: Giao diện tải phần mềm Arduino IDE

Khi tải xong thì việc cài đặt rất đơn giản, ta chỉ cần ấn next là sẽ cài đặt thành công.

b. Hướng dẫn nạp code cho ESP8266 ESP-12F

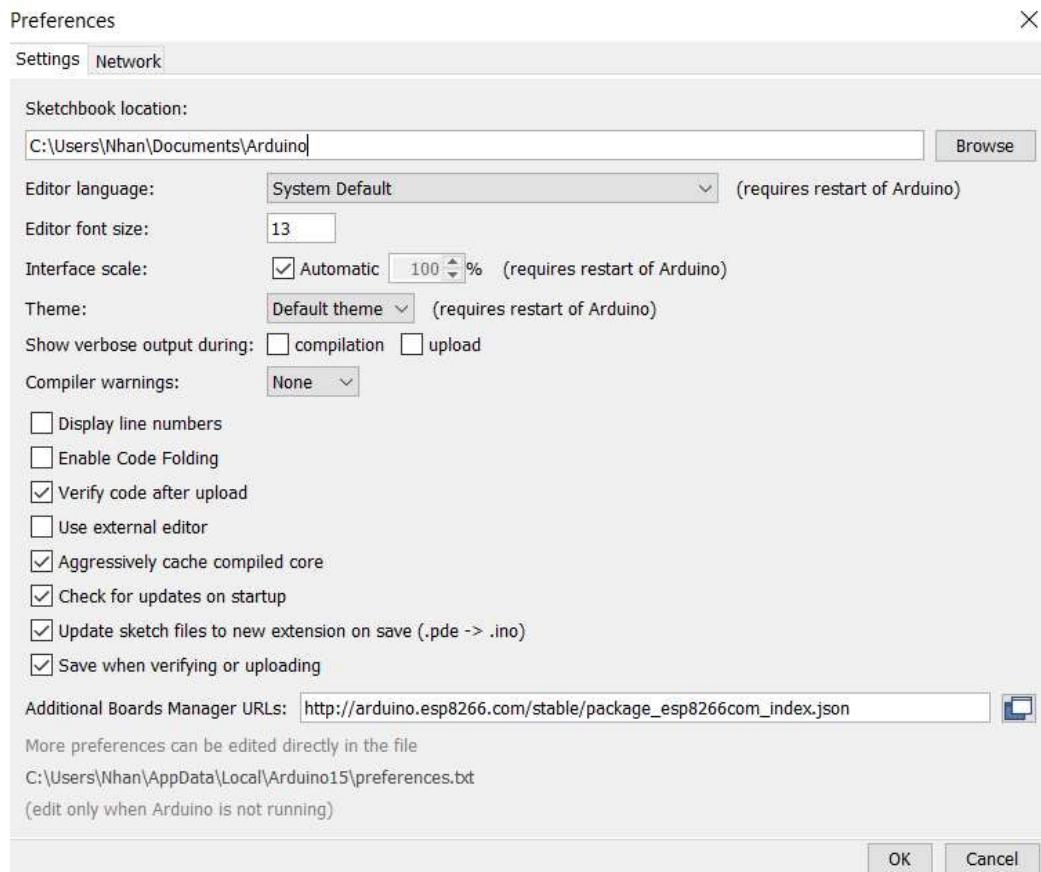
Để nạp code ESP-12F ta phải sử dụng USB-TLL như đã giới thiệu ở trên.

❖ Cấu hình Arduino IDE

Để cấu hình cho ESP-12F ta phải sử dụng Arduino IDE phiên bản 1.6.5 trở lên.

- Đầu tiên, vào File > Preferences sau đó copy đường link vào text box Additional Board Manager URLs sau đó bấm OK.
- Tiếp theo vào Tool > Board > Board Manager, ta nhập vào ô tìm kiếm là ESP8266 và chọn ESP8266 by ESP8266 Community sau đó nhấn install và chờ phần mềm tự động download và cài đặt.

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG



Hình 3. 2: Cấu hình cho Arduino IDE



Hình 3. 3: Download cấu hình của board ESP8266

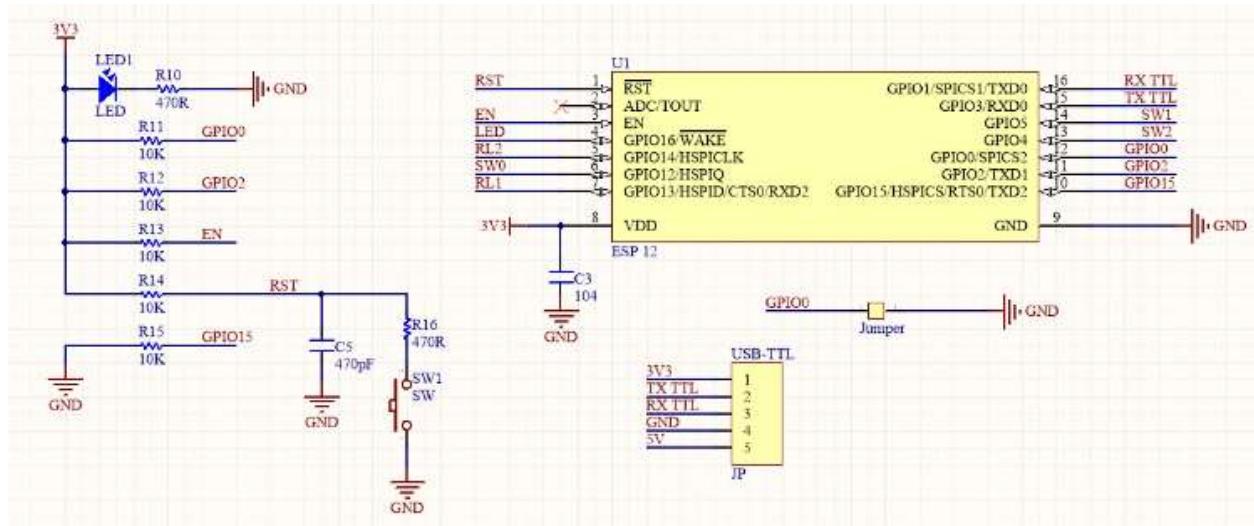
CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

❖ Cấu hình phần cứng và nạp code

Khi thiết kế board cần đảm bảo nó sẽ hoạt động tốt ở chế độ Run mode (load code từ bộ nhớ flash khi reset hoặc cấp nguồn) và Flash mode (nạp code mới).

Trước khi nạp code cần cấu hình mức logic các chân của ESP8266 như sau:

- GPIO0 ở mức LOW.
- GPIO2 ở mức HIGH.
- GPIO15 ở mức LOW.



Hình 3. 4: Sơ đồ phần cứng ESP-12F để nạp code

ESP-12F	USB - TLL
VCC	3V3
GND	GND
TX	RX
RX	TX

Bảng 3. 1: Kết nối ESP-12F với USB-TLL để nạp code

Khi đã kết nối, ta cắm USB-TLL vào máy tính và chọn Tool > Board > NodeMCU 1.0. Chọn cổng COM tương ứng tại Tool > Port và sau đó nạp code bình thường (lưu ý: ESP-12F sử dụng điện áp 3.3V nên không được kết nối chân 5V trên USB-TLL sẽ làm hỏng vi điều khiển).

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

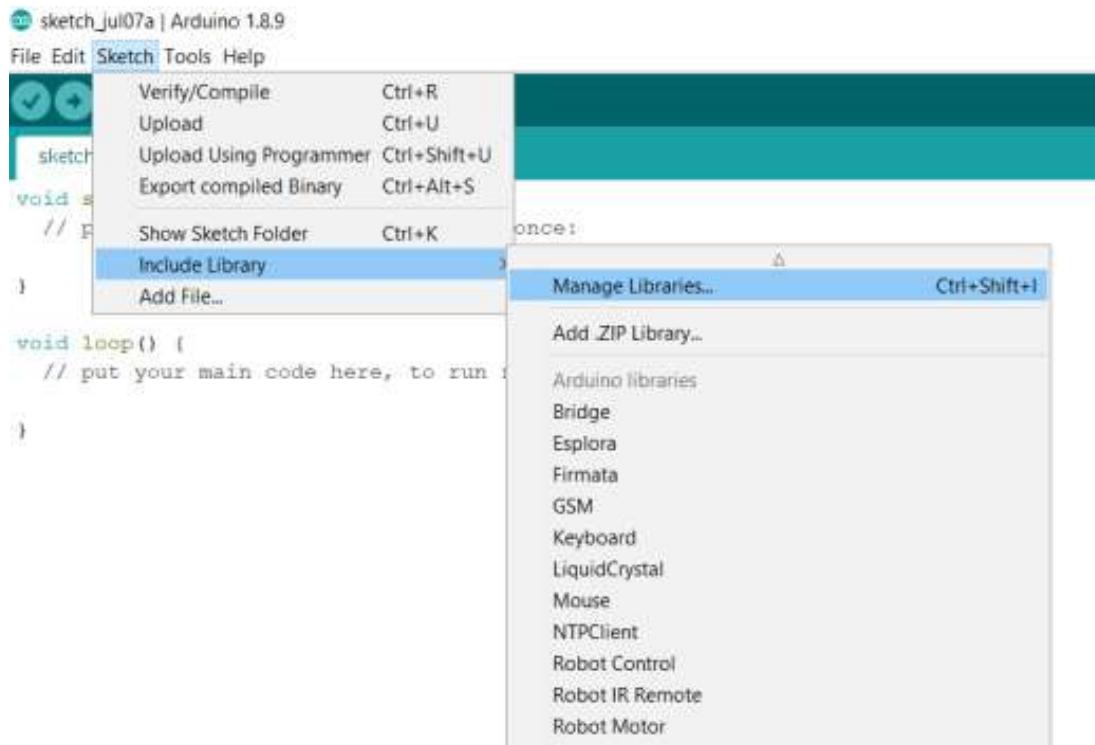
Khi nạp code xong, để thoát khỏi chế độ flash mode và chuyển về chế độ run mode ta cần giữ các chân của ESP-12F ở mức điện áp như sau:

- GPIO0 ở mức HIGH.
- GPIO2 ở mức HIGH.
- GPIO15 ở mức LOW.

❖ Add thư viện cho Arduino IDE

Để Add thư viện có sẵn vào Arduino IDE ta có 2 cách như sau:

- Cách 1: Ta chọn Sketch > include Library > Manager Libraries, trong đây có các thư viện được phần mềm Arduino IDE update lên. Ta có thể tìm kiếm trên thanh công cụ và nhấn install để dowload về. Thư viện Manage Libraries không được đa dạng, nhiều thư viện không có sẵn trong này.
- Cách 2: Ta chọn Sketch > include Library > Add.ZIP library, ở đây ta có thể add thư viện vào bằng tệp .ZIP. Với cách add thư viện .ZIP này ta có thể tìm kiếm các thư viện ở các trang nguồn ở trên mạng như githup,... để dowload về add vào Arduino IDE.



Hình 3. 5: Add thư viện bằng Manage Libraries và Add .ZIP Library

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

- Ở đồ án này, nhóm đã sử dụng các thư viện bao gồm: ESP8266WiFi.h, EEPROM.h, WebSocketsServer.h, Firebase_ESP_Client.h, time.h, Wire.h PZEM004Tv30.h và OLED.h. [14] Link download các thư viện, mọi người có thể copy đường link, download và add vào Arduino IDE.

2. Các chế độ hoạt động của ESP8266

a. Chế độ WiFi Access Point

Chế độ WiFi Access Point (AP - Điểm truy cập) cung cấp quyền truy cập mạng WiFi cho các thiết bị khác (Station). Khi ở chế độ softAP, ESP8266 sẽ phát ra một WiFi với ssid của bạn cài đặt và địa chỉ để kết nối với ESP, có thể cài thêm password để tăng tính bảo mật. Số lượng thiết bị tối đa để kết nối với soft-AP là 5.



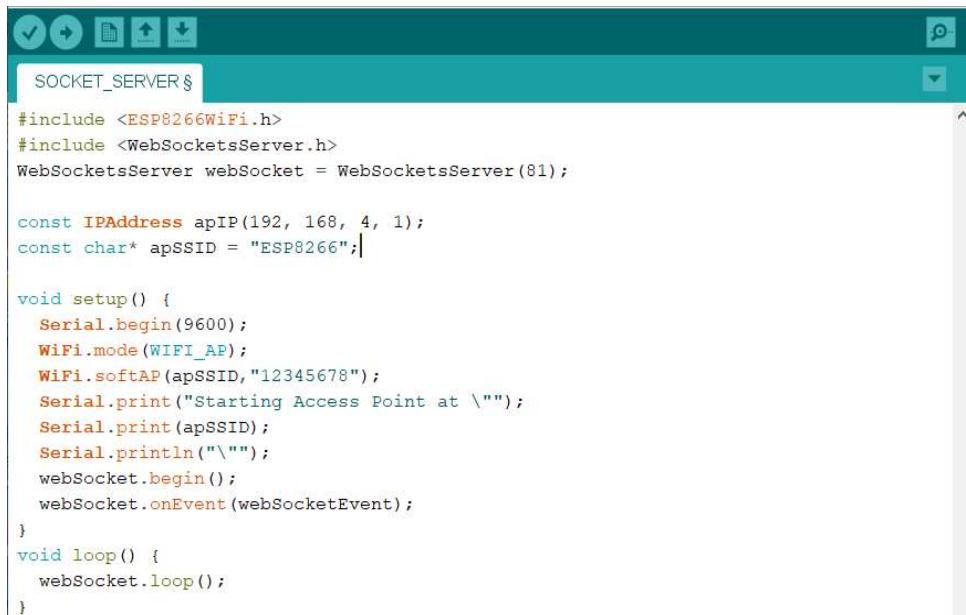
Hình 3. 6: ESP8266 hoạt động ở chế độ softAP

❖ Thiết lập chế độ softAP cho ESP8266 với các lệnh cơ bản:

- Thiết lập chế độ softAP với một lệnh duy nhất: WiFi.softAP (ssid).
- Nếu muốn cài đặt thêm cấu hình thông số mạng bổ sung, ta thực hiện lệnh như sau:
WiFi.softAP (ssid, password, channel, hidden).
 - ssid: là chuỗi ký tự ssid mạng (tối đa 63 ký tự).
 - password: là chuỗi ký tự tùy chọn với mật khẩu. Đối với mạng WPA2-PSK, phải có ít nhất 8 ký tự. Nếu không cài đặt mật khẩu, thì đây sẽ là một mạng WiFi mở.
 - channel: là tham số tùy chọn để thiết lập kênh Wi-Fi, từ 1 đến 13. Kênh mặc định = 1.
 - hidden: là tham số tùy chọn, thiết lập là true để ẩn SSID.

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

- WiFi.softAPgetStationNum(): lệnh để lấy số lượng các thiết bị station kết nối với softAP.
 - WiFi.mode(WIFI_AP): bật chế độ AP cho ESP8266.
 - WiFi.softAPdisconnect(wifioff): chức năng này sẽ thiết lập ssid và password của softAP thành giá trị null. Wifioff là tùy chọn, nếu thiết lập là true thì nó sẽ tắt chế độ softAP.
 - WiFi.softAPIP(): lấy IP của mạng AP.
 - WiFi.softAPmacAddress(): lấy địa chỉ mac của AP.
- ❖ Ví dụ về chương trình cơ bản ở chế độ softAP cho ESP8266: Trong chương trình này, ESP8266 sẽ phát wifi với ssid là ESP8266 và password là 12345678. Cùng với đó là địa chỉ IPAddress và port để các máy trạm kết nối tới.



The screenshot shows a code editor window titled "SOCKET_SERVER". The code is written in C++ for an Arduino. It includes headers for WiFi and WebSocketsServer, initializes a WebSocketsServer on port 81, and sets up the WiFi mode to AP with SSID "ESP8266" and password "12345678". It then prints a message to the serial monitor and starts the webSocket loop.

```
#include <ESP8266WiFi.h>
#include <WebSocketsServer.h>
WebSocketsServer webSocket = WebSocketsServer(81);

const IPAddress apIP(192, 168, 4, 1);
const char* apSSID = "ESP8266";| 

void setup() {
    Serial.begin(9600);
    WiFi.mode(WIFI_AP);
    WiFi.softAP(apSSID, "12345678");
    Serial.print("Starting Access Point at \\"");
    Serial.print(apSSID);
    Serial.println("\\"");
    webSocket.begin();
    webSocket.onEvent(webSocketEvent);
}
void loop() {
    webSocket.loop();
}
```

Hình 3. 7: Ví dụ về chế độ softAP của ESP8266

b. Chế độ WiFi Station

Với việc kết nối mạng WiFi cho EPS8266, lúc này ESP8266 được gọi là station (trạm).

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG



Hình 3. 8: Chế độ WiFi Station

❖ Thiết lập chế độ WiFi Station cho ESP8266:

- WiFi.mode(WIFI_STA): bật chế độ STA cho ESP8266.
- WiFi.begin(ssid, password): kết nối ESP8266 vào một điểm truy cập WiFi.
- WiFi.reconnect(): kết nối lại WiFi.
- WiFi.disconnect(WiFiOff): thiết lập ssid và password thành null để ngắt kết nối internet, với WiFiOff là tham số kiểu boolean, nếu là true thì chế độ STA sẽ bị tắt.
- WiFi.isConnected(): kiểm tra kết nối WiFi.
- WiFi.status(): chuẩn đoán WiFi.
 - WL_CONNECTED: được chỉ định khi được kết nối với mạng WiFi .
 - WL_NO_SHIELD: được chỉ định khi không có lá chắn WiFi.
 - WL_IDLE_STATUS: đó là trạng thái tạm thời được chỉ định khi WiFi.begin () được gọi và vẫn hoạt động cho đến khi số lần thử hết hạn (kết quả là WL_CONNECT_FAILED) hoặc kết nối được thiết lập (kết quả là WL_CONNECTED).
 - WL_NO_SSID_AVAIL: được chỉ định khi không có SSID.
 - WL_SCAN_COMPLETED: được chỉ định khi hoàn thành các mạng quét.
 - WL_CONNECT_FAILED: được chỉ định khi kết nối không thành công cho tất cả các lần thử.
 - WL_CONNECTION_LOST: được chỉ định khi mất kết nối.
 - WL_DISCONNECTED: được chỉ định khi ngắt kết nối mạng.

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

❖ Ví dụ về chương trình cơ bản ở chế độ STA cho ESP8266: Trong chương trình này ESP8266 được thiết lập để kết nối với ESP8266 ở chế độ softAP như hình 2.21. Khi đã kết nối thành công, ESP8266 ở chế độ STA sẽ gửi đi một chuỗi dữ liệu bằng lệnh webSocket.SendTXT. Trong trường hợp này 2 con ESP8266 có thể gửi dữ liệu qua lại như một ứng dụng chat, được gọi là websocket Server và websocket Client.

```
#include <ESP8266WiFi.h>
#include <WebSocketsClient.h>
WebSocketsClient webSocket;

const char* ip_host = "192.168.4.1";
const uint16_t port = 81;
const char* ssid = "ESP8266";
const char* pass = "12345678";
void setup() {
    Serial.begin(9600);
    WiFi.begin(ssid, pass);
    Serial.printf("Connecting to %s \n", ssid);
    while (WiFi.status() != WL_CONNECTED)
    {
        delay(500);
        Serial.print(".");
    }
    Serial.println();
    Serial.print("Connected, IP address: ");
    Serial.println(WiFi.localIP());
    webSocket.begin(ip_host, port);
}
void loop() {
    webSocket.loop();
    webSocket.sendTXT("/HEHEHE=0776537277&43hjkFKDSOK9u787234klj@12");
    delay(200);
}
```

Hình 3. 9: Ví dụ về chế độ STA cho ESP8266

3. WebSocket

Websocket là công nghệ hỗ trợ giao tiếp hai chiều giữa client và server bằng cách sử dụng một TCP socket để tạo một kết nối hiệu quả và ít tốn kém.

Ưu điểm của nó là cung cấp khả năng giao tiếp nhanh chóng, có độ trễ thấp và dễ xử lý lỗi thường được sử dụng trong các ứng dụng chat.

Sau đây là ví dụ cơ bản về việc giao tiếp 2 ESP với nhau bằng websocket: trước khi nạp code, ta cần phải add 2 thư viện là WebSocketsClient.h và WebSocketsServer.h để khi build không bị lỗi thiếu thư viện.

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

```
#include <ESP8266WiFi.h>
#include <WebSocketsClient.h>
WebSocketsClient webSocket;

const char* ip_host = "192.168.4.1";
const uint16_t port = 81;
const char* ssid = "ESP8266";
const char* pass = "12345678";
void setup() {
    Serial.begin(9600);
    WiFi.begin(ssid,pass);
    Serial.printf("Connecting to %s \n", ssid);
    while (WiFi.status() != WL_CONNECTED)
    {
        delay(500);
        Serial.print(".");
    }
    Serial.println();
    Serial.print("Connected, IP address: ");
    Serial.println(WiFi.localIP());
    webSocket.begin(ip_host, port);
}

void loop() {
    webSocket.loop();
    webSocket.sendTXT("/HEHEHE=0776537277&43hjkFKDSOK9u787234klj@12");
}
```

Hình 3. 10: ESP8266 được thiết lập làm Client

Trong trường hợp này, ESP8266 được thiết lập làm Client để kết nối tới ESP8266 làm Server. Khi kết nối thành công, Client sẽ gửi một chuỗi dữ liệu qua Server và ngược lại server cũng gửi lại một chuỗi dữ liệu xuống client. Để nhận được dữ liệu, ESP8266 client bắt buộc phải có chương trình void WebSocketEvent.

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

```
#include <ESP8266WiFi.h>
#include <WebSocketsServer.h>
WebSocketsServer webSocket = WebSocketsServer(81);

const IPAddress apIP(192, 168, 4, 1);
const char* apSSID = "ESP8266";

void setup() {
    Serial.begin(9600);
    WiFi.mode(WIFI_AP);
    WiFi.softAP(apSSID,"12345678");
    Serial.print("Starting Access Point at \\"");
    Serial.print(apSSID);
    Serial.println("\\"");
    webSocket.begin();
    webSocket.onEvent(webSocketEvent);
}

void loop() {
    webSocket.loop();
    webSocket.broadcastTXT("@C201=C201nltao&xyzGHYKLM123x");
}

void webSocketEvent(uint8_t num, WStype_t type,
                    uint8_t * payload,
                    size_t length)
{
    String payloadString = (const char *)payload;
    Serial.print(payloadString);
}
```

Hình 3. 11: ESP8266 được thiết lập làm Server

Trong trường hợp này, ESP8266 được thiết lập làm Server để phát Wifi. Khi WiFi được phát, ESP8266 Client sẽ kết nối tới và gửi một chuỗi dữ liệu qua client và ngược lại client cũng gửi lại một chuỗi dữ liệu xuống Server. Để nhận được dữ liệu, ESP8266 Server cũng bắt buộc phải có chương trình void WebSocketEvent.

4. Ghi và đọc dữ liệu bộ nhớ EEPROM

a. Ghi dữ liệu vào bộ nhớ EEPROM

Khi nhận được dữ liệu từ server gửi xuống, ta phải lưu vào bộ nhớ EEPROM để không bị mất dữ liệu. Vì khi mất điện, chương trình sẽ chạy lại từ đầu khi bắt đầu có điện, điều này sẽ làm mất dữ liệu đã lưu từ trước nên bắt buộc ta phải lưu vào bộ nhớ EEPROM.

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

Sau đây là ví dụ về việc ghi dữ liệu vào bộ nhớ EEPROM. Bộ nhớ EEPROM có tất cả là 512 ô nhớ, vì vậy dữ liệu gửi sẽ phải nhỏ hơn 512, nếu lớn hơn thì sẽ không lưu vào được bộ nhớ EEPROM. Vì thư viện EEPROM đã có sẵn lúc ta cài đặt thư viện ESP8266 nên chỉ cần khai báo và sử dụng.

```
#include <ESP8266WiFi.h>
#include <EEPROM.h>
String username;
String password;
String Token;
String Phong;
void setup() {
    Serial.begin(9600);
    EEPROM.begin(512);
}
void loop()
{
    writeStringToEEPROM();
}
// ghi dữ liệu vào bộ nhớ EEPROM
void writeStringToEEPROM()
{
    for (int i = 0; i < 512; ++i) {
        EEPROM.write(i, 0); //xoa bo nho EEPROM
        delay(10);
    }
    Serial.println("Writing username to EEPROM...");
    for (int i = 0; i < username.length(); ++i) {
        EEPROM.write(i, username[i]);
    }
    Serial.println(username);
    Serial.println("Writing Password to EEPROM...");
    for (int i = 0; i < password.length(); ++i) {
        EEPROM.write(32 + i, password[i]);
    }
    Serial.println(password);
    Serial.println("Writing ToKen to EEPROM...");
    for (int i = 0; i < Token.length(); ++i) {
        EEPROM.write(96 + i, Token[i]);
    }
    Serial.println(Token);
    Serial.println("Writing ToKen to EEPROM...");
    for (int i = 0; i < Phong.length(); ++i) {
        EEPROM.write(500 + i, Phong[i]);
    }
    Serial.println(Phong);
}
```

Hình 3. 12: Chương trình ghi dữ liệu vào bộ nhớ EEPROM

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

b. Đọc dữ liệu từ bộ nhớ EEPROM

Khi đã ghi dữ liệu vào bộ nhớ EEPROM, thì khi cúp điện hoặc khởi động lại chương trình ta chỉ cần đọc dữ liệu đã lưu trước đó và thực hiện chương trình.

Sau đây là ví dụ về đọc dữ liệu từ bộ nhớ EEPROM. Khi có dữ liệu thì sẽ đọc bộ nhớ EEPROM, khi không có dữ liệu chương trình sẽ in ra (config not found) tức là không có dữ liệu để đọc.

```
#include <ESP8266WiFi.h>
#include <EEPROM.h>
String username;
String password;
String Token;
String Phong;
void setup() {
    Serial.begin(9600);
    EEPROM.begin(512);
}
void loop()
{
    readStringToEEPROM();
}
// ghi dữ liệu vào bộ nhớ EEPROM
void readStringToEEPROM()
{
    if (EEPROM.read(0) != 0) {
        for (int i = 0; i < 31; ++i) {
            ssid += char(EEPROM.read(i));
        }
        Serial.println("Reading username to EEPROM: ");
        Serial.println(username);
        for (int i = 32; i < 95; ++i) {
            pass += char(EEPROM.read(i));
        }
        Serial.println("Reading Password to EEPROM: ");
        Serial.println(password);
        for (int i = 96; i < 499; ++i) {
            Token1 += char(EEPROM.read(i));
        }
        Serial.println("Reading Token to EEPROM: ");
        Serial.println(Token);
        for (int i = 500; i < 512; ++i) {
            Phong1 += char(EEPROM.read(i));
        }
        Serial.println("Reading Token to EEPROM: ");
        Serial.println(Phong);
    }
    else{
        Serial.println("Config not found.");
    }
}
```

Hình 3. 13: Chương trình đọc dữ liệu từ bộ nhớ EEPROM

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

5. Đọc dữ liệu từ PZEM-004T về ESP8266

Để đọc được dữ liệu từ pzem, ta phải giao tiếp truyền thông UART cho esp8266 với pzem bằng cách nối chân TX của pzem với chân RX (GPIO3) của esp8266 và nối chân RX của pzem với chân TX (GPIO1) của esp8266. Nguồn điện 5v cũng được lấy từ esp8266 để kết nối với pzem.

Sau đây là ví dụ về việc đọc dữ liệu của pzem về esp8266 thông qua cổng serial. Trước khi biên dịch chương trình bạn cần phải cài đặt thư viện PZEM004Tv30.h để không bị lỗi.

```
#include <PZEM004Tv30.h>
PZEM004Tv30 pzem(&Serial);
void setup() {
    Serial.begin(9600);
}
void loop() {
    // dien ap
    float voltage = pzem.voltage();
    if( !isnan(voltage) ){
        Serial.print("Voltage: "); Serial.print(voltage); Serial.println("V");
    } else {
        Serial.println("Error reading voltage");
    }
    // dong dien
    float current = pzem.current();
    if( !isnan(current) ){
        Serial.print("Current: "); Serial.print(current); Serial.println("A");
    } else {
        Serial.println("Error reading current");
    }
    // cong suat
    float power = pzem.power();
    if( !isnan(power) ){
        Serial.print("Power: "); Serial.print(power); Serial.println("W");
    } else {
        Serial.println("Error reading power");
    }
    // cong suat tieu thu
    float energy = pzem.energy();
    if( !isnan(energy) ){
        Serial.print("Energy: "); Serial.print(energy,3); Serial.println("kWh");
    } else {
        Serial.println("Error reading energy");
    }
    // tan so
    float frequency = pzem.frequency();
    if( !isnan(frequency) ){
        Serial.print("Frequency: "); Serial.print(frequency, 1); Serial.println("Hz");
    } else {
        Serial.println("Error reading frequency");
    }
    // cos phi
    float pf = pzem.pf();
    if( !isnan(pf) ){
        Serial.print("PF: "); Serial.println(pf);
    } else {
        Serial.println("Error reading power factor");
    }
    delay(2000);
}
```

Hình 3. 14: Chương trình đọc dữ liệu pzem về esp8266

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

6. In dữ liệu lên màn hình OLED

Mục đích sử dụng màn hình oled trong đồ án này là in dữ liệu các thông số như: điện áp, dòng điện, công suất, công suất tiêu thụ,... lên trên màn hình để thuận tiện theo dõi vì các thông số này sẽ được thay đổi liên tục. Ngoài ra còn sử dụng với mục đích in dữ liệu thời gian thực bao gồm: ngày, tháng, năm, giờ, phút lên màn hình.

Sau đây là ví dụ in dữ liệu thời gian thực lên màn hình oled. Đầu tiên ta kết nối đến server thời gian thực sau đó lấy dữ liệu và in lên màn hình oled.

```
#include <Wire.h>
#include "OLED.h"
#include <time.h>
OLED display(4, 5);
void setup() {
    Serial.begin(9600);
    display.begin();
    configTime(timezone, dst, "pool.ntp.org", "time.nist.gov");
    Serial.println("Witting Connect Timezone");
    while(!time(nullptr)){
        Serial.print("*");
        delay(500);
    }
}
void loop() {
    time_t now=time(nullptr);
    struct tm* p_tm=localtime(&now);
    Serial.print(p_tm ->tm_mday);
    Serial.print("/");
    Serial.print(p_tm ->tm_mon + 1);
    Serial.print("/");
    Serial.print(p_tm ->tm_year + 1900);
    Serial.print(" ");
    Serial.print(p_tm ->tm_hour);
    Serial.print(":");
    Serial.print(p_tm ->tm_min);
    Serial.print(":");
    Serial.print(p_tm ->tm_sec);
    Serial.print("\n");
    display.print(tm_day, 1, 3);
    display.print("/", 1, 5);
    display.print(tm_mon, 1, 6);
    display.print("/", 1, 8);
    display.print(tm_year, 1, 9);
    display.print(tm_hour, 2, 5);
    display.print(tm_min, 2, 8);
}
```

Hình 3. 15: Chương trình in dữ liệu thời gian thực lên màn hình OLED

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

7. Khởi tạo tập dữ liệu từ ESP8266 và gửi lên Cloud Firestore

a. Tạo tập dữ liệu trên Cloud Firestore

Để khởi tạo được tập dữ liệu từ esp8266 và gửi lên cloud Firestore, ta cần phải kết nối Internet cho esp8266, sau đó kết nối đến server của cloud Firestore. Để kết nối được với cloud Firestore của mình đã tạo, ta cần phải biết được các mã như: API_KEY, FIREBASE_PROJECT_ID,... USER_NAME, USER_PASSWORD của Authentication, để đảm bảo kết nối đúng tới server mình đã tạo.

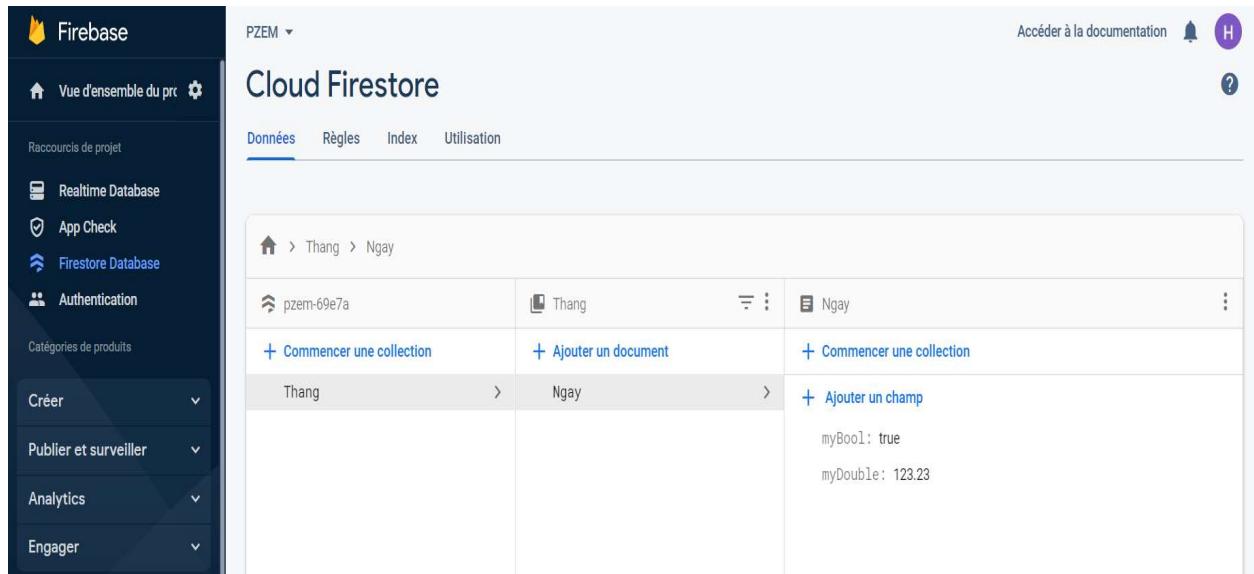
```
#include <Firebase_ESP_Client.h>
#include "addons/TokenHelper.h"

#define WIFI_SSID "WIFI_AP"
#define WIFI_PASSWORD "WIFI_PASSWORD"
#define API_KEY "API_KEY"
#define FIREBASE_PROJECT_ID "PROJECT_ID"
#define USER_EMAIL "USER_EMAIL"
#define USER_PASSWORD "USER_PASSWORD"
FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;
unsigned long dataMillis = 0;
int count = 0;
void setup()
{
    Serial.begin(115200);
    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
    Serial.print("Connecting to Wi-Fi");
    while (WiFi.status() != WL_CONNECTED)
    {
        Serial.print(".");
        delay(300);
    }
    Serial.println();
    Serial.print("Connected with IP: ");
    Serial.println(WiFi.localIP());
    Serial.println();
    Serial.printf("Firebase Client v%s\n\n", FIREBASE_CLIENT_VERSION);
    config.api_key = API_KEY;
    auth.user.email = USER_EMAIL;
    auth.user.password = USER_PASSWORD;
    config.token_status_callback = tokenStatusCallback; //see addons/TokenHelper.h
    Firebase.begin(&config, &auth);
    Firebase.reconnectWiFi(true);
    createdocument();
}
void loop()
{
}
void createdocument(){
    String content;
    FirebaseJson js;
    String documentPath = "Nam/Thang/Ngay";
    js.set("fields/myDouble/doubleValue", 123.23);
    js.set("fields/myBool/booleanValue", true);
    js.toString(content);
    Serial.print("Create a document... ");
    if (Firebase.Firestore.createDocument(&fbdo, FIREBASE_PROJECT_ID, ""+ documentPath.c_str(), content.c_str()))
        Serial.printf("ok\n%s\n\n", fbdo.payload().c_str());
    else
        Serial.println(fbdo.errorReason());
}
```

Hình 3.16: Chương trình khởi tạo tập tin từ ESP8266 gửi lên Firestore

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

Sau đây là ví dụ về việc tạo một tập tin từ esp8266 và gửi lên cloud Firebase. Tập dữ liệu bao gồm Nam > Thang > Ngay. Các biến dữ liệu được khởi tạo bao gồm myBool và myDouble sẽ lưu vào tập tin cuối cùng đó là ngày.



The screenshot shows the Firebase Cloud Firestore interface. On the left, there's a sidebar with project settings like Realtime Database, App Check, and Firestore Database. The main area is titled "Cloud Firestore" and has tabs for "Données", "Règles", "Index", and "Utilisation". Below the tabs, it shows a hierarchical path: Home > Thang > Ngay. A table lists a single document named "Ngay" under the collection "Thang". The document contains two fields: "myBool" with the value "true" and "myDouble" with the value "123.23". There are also buttons to "Commencer une collection" and "Ajouter un champ".

Hình 3. 17: Hiển thị trên Firestore khi tạo thành công

b. Cập nhập dữ liệu từ ESP8266 và gửi lên Cloud Firestore

Sau khi đã tạo được tệp dữ liệu thì việc cập nhập khá dễ dàng. Việc cập nhập dữ liệu này liên quan đến việc cập nhập các thông số bao gồm điện áp, dòng điện, công suất tiêu thụ,... trong đồ án này vì những giá trị này luôn được thay đổi. Sau đây là ví dụ về cập nhật dữ liệu lên cloud Firestore, biến dữ liệu myDouble có thể thay đổi giá trị khi ta thay đổi giá trị trong esp8266. Ta có thể sử dụng biến dữ liệu như điện áp, dòng điện,... để bỏ vào giá trị 123.23 đã được cài mặc định từ trước. Các biến dữ liệu điện áp dòng điện khi được thay đổi sẽ làm giá trị của biến myDouble thay đổi theo.

Chương trình sau đây được cài đặt 60 giây sẽ cập nhập giá trị một lần. Phần void updatedocument() sẽ được thay vào phần void createdocument ở chương trình trước.

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

```
void updatedocument() {
    if (Firebase.ready() && (millis() - dataMillis > 60000 || dataMillis == 0))
    {
        dataMillis = millis();
        String content;
        FirebaseJson js;
        String documentPath = "Nam/Thang/Ngay";
        js.set("fields/myDoubledoubleValue", 123.23);
        js.set("fields/myBool/booleanValue", true);
        js.toString(content);
        Serial.print("Update a document... ");
        if (Firebase.Firestore.patchDocument(&fbdo, FIREBASE_PROJECT_ID, "", documentPath.c_str(), content.c_str(), "myDouble"))
            Serial.printf("ok\n%s\n\n", fbdo.payload().c_str());
        else
            Serial.println(fbdo.errorReason());
    }
}
```

Hình 3. 18: Chương trình cập nhập giá trị từ ESP8266 gửi lên Firestore

c. Nhận dữ liệu Cloud Firestore về ESP8266

Mục đích của việc nhận dữ liệu từ cloud Firestore về esp8266 là dùng để điều khiển các biến giá trị bool như bật tắt relay để điều khiển tải và reset năng lượng cho pzem được sử dụng trong đồ án này.

Sau đây là ví dụ về việc lấy biến giá trị kiểu bool về esp8266. Khi dữ liệu trên Firestore thay đổi, esp8266 sẽ lấy giá trị đó về và dùng để điều khiển. Cũng giống như tạo và cập nhập dữ liệu, phần void getdocument() sẽ được thay vào phần với updatedocument() ở phần trước. Khi dữ liệu myBool hoặc myDouble thay đổi trên Firestore thì esp8266 sẽ nhận về giá trị đã thay đổi đó. Chương trình được cài đặt 60 giây nhận về một lần.

```
void getdocument() {
    if (Firebase.ready() && (millis() - dataMillis > 60000 || dataMillis == 0))
    {
        dataMillis = millis();
        String content;
        FirebaseJson js;
        String documentPath = "Nam/Thang/Ngay";
        js.set("fields/myDoubledoubleValue", 123.23);
        js.set("fields/myBool/booleanValue", true);
        js.toString(content);
        Serial.print("Get a document... ");
        if (Firebase.Firestore.getDocument(&fbdo, FIREBASE_PROJECT_ID, "", documentPath.c_str(), mask.c_str()))
            Serial.printf("ok\n%s\n\n", fbdo.payload().c_str());
        else
            Serial.println(fbdo.errorReason());
    }
}
```

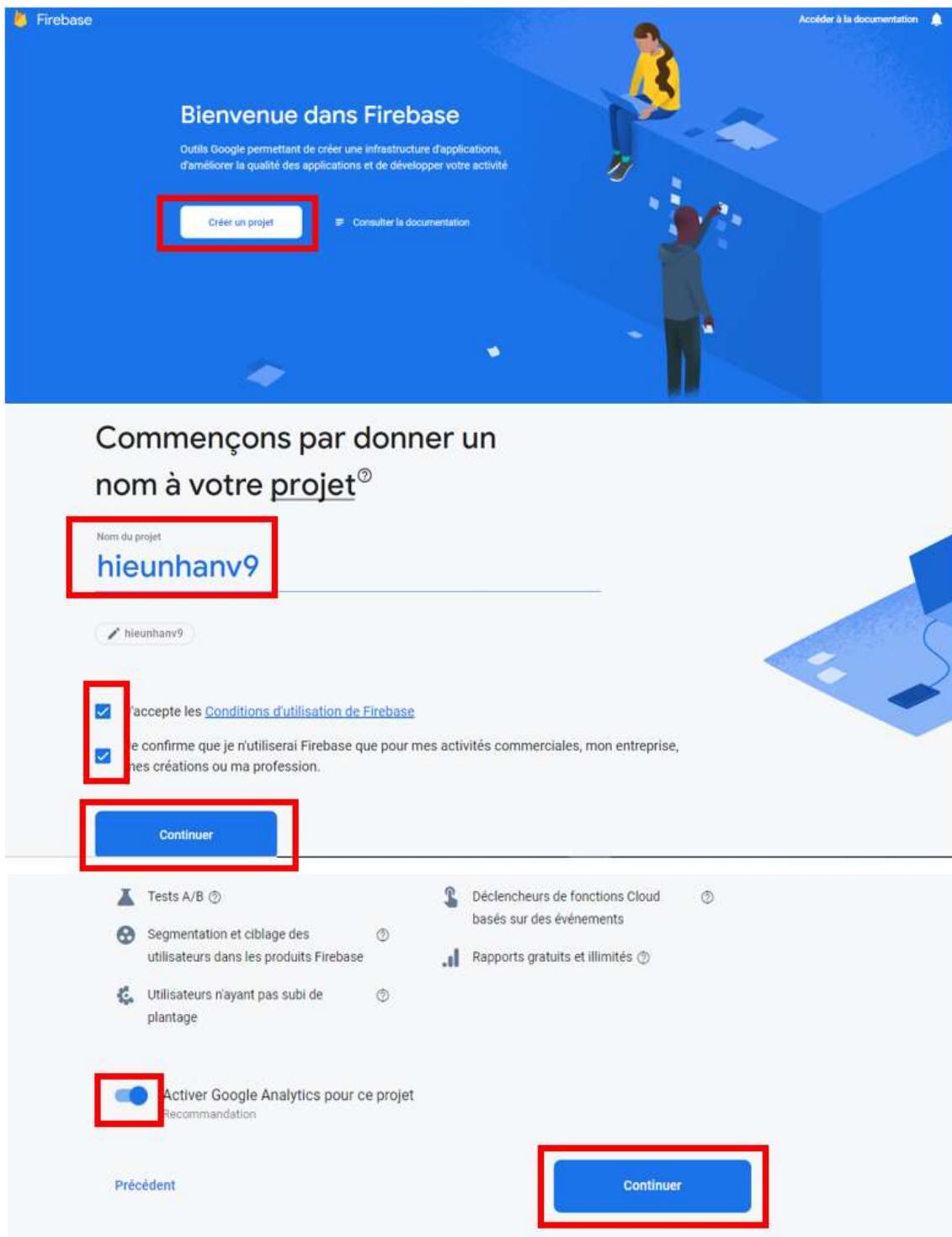
Hình 3. 19: Chương trình lấy dữ liệu từ Cloud Firestore về ESP8266

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

II. KHỞI TẠO SERVER

1. Khởi tạo Google Cloud Firestore

Truy cập vào đường link [7] và khởi tạo dự án.



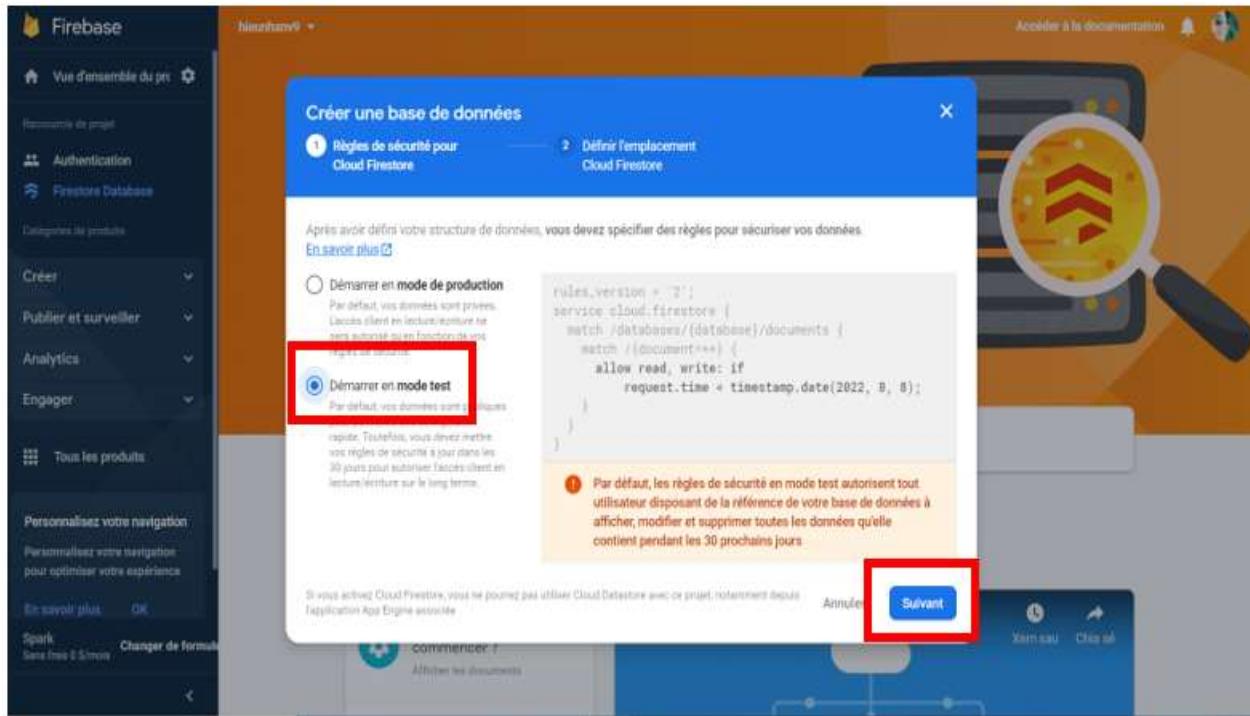
Hình 3. 20: Khởi tạo dự án

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

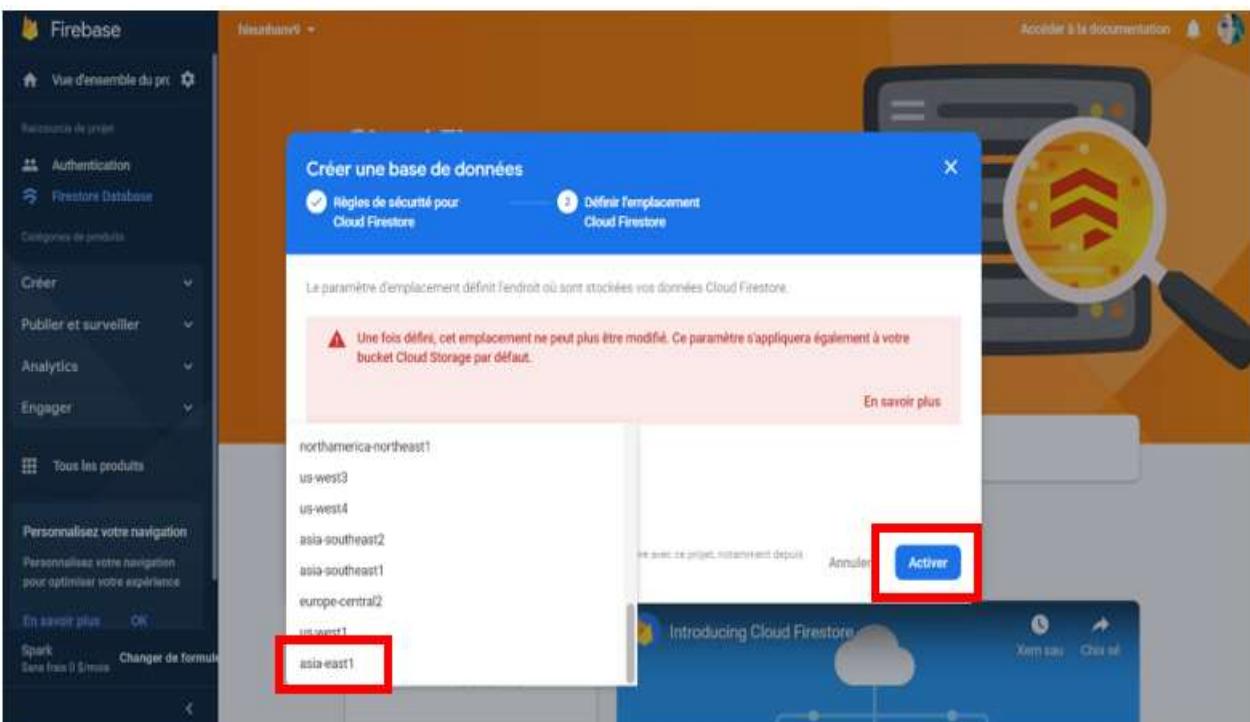
The screenshot shows the Firebase console interface. On the left, a sidebar menu includes 'Vue d'ensemble du proj' (selected), 'Utilisateurs et autorisations', 'Créer', 'Publier et surveiller', 'Analytics', 'Engager', and 'Tous les produits'. Below this is a section for 'Personnalisez votre navigation' and a plan summary: 'Spark' (Sans frais 0 \$/mois) with a 'Changer de formule' link. On the right, the main area is titled 'du projet' with tabs for 'Paramètres du projet', 'Cloud Messaging', 'Intégrations', 'Comptes de service', 'Confidentialité des données', 'Utilisateurs et autorisations', and 'App Check'. The 'Paramètres du projet' tab is active, showing fields for 'Nom du projet' (redacted), 'ID du projet' (redacted), 'Numéro du projet' (redacted), and 'Emplacement des ressources GCP par défaut' (redacted). A red box highlights these four fields. Below this is the 'Environnement' section with 'Type d'environnement' set to 'Non spécifié'. The 'Paramètres publics' section shows 'Nom public' (redacted) and 'project-389685835643' (with a pencil icon). A second screenshot below shows the 'Cloud Firestore' section of the console, with 'Authentication' selected in the sidebar. It features a large orange background with the 'Cloud Firestore' logo and text: 'Mises à jour en temps réel, requêtes performantes et scaling automatique'. A red box highlights the 'Créer une base de données' button. At the bottom, there's a 'Cloud Firestore vous convient-il le mieux?' button and a 'Comparer les bases de données' link. The 'En savoir plus' section includes links for 'Que dois-je faire pour commencer?' and 'Afficher les documents'.

Hình 3. 21: Khởi tạo cloud Firestore database

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG



Hình 3.22: Tuỳ chọn bảo mật và kết nối



Hình 3.23: Chọn vị trí đặt server

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

2. Khởi tạo Server để ESP8266 kết nối tới Cloud Firestore

The screenshot shows the Firebase console interface. On the left, there's a sidebar with various services like Authentication, Firestore Database, and Analytics. The main area is titled "Paramètres du projet" (Project Settings) for a project named "hieunhanut". It includes sections for "Paramètres publics" (Public settings), "Vos applications" (Your apps), and a "Supprimer le projet" (Delete project) button. A red box highlights the "Paramètres du projet" header and the "Vos applications" section. Another red box highlights the "Web" icon in the "Vos applications" row.

Ajouter Firebase à votre application Web

The screenshot shows the "Ajouter Firebase à votre application Web" (Add Firebase to your web application) process. It consists of two main steps:

- 1 Enregister l'application**:
A form to register a new application. The "Pseudo de l'application" field contains "hieunhanut" (highlighted by a red box). There's a checkbox for "Also set up Firebase Hosting for this app." and a "Enregistrer l'application" button (highlighted by a red box).
- 2 Ajouter le SDK Firebase**:
A section to add the Firebase SDK. It shows two options: "Utiliser npm" (radio button) and "Utiliser une balise <script>" (radio button, highlighted by a red box). Below is a code snippet for initializing the Firebase SDK in an HTML file:

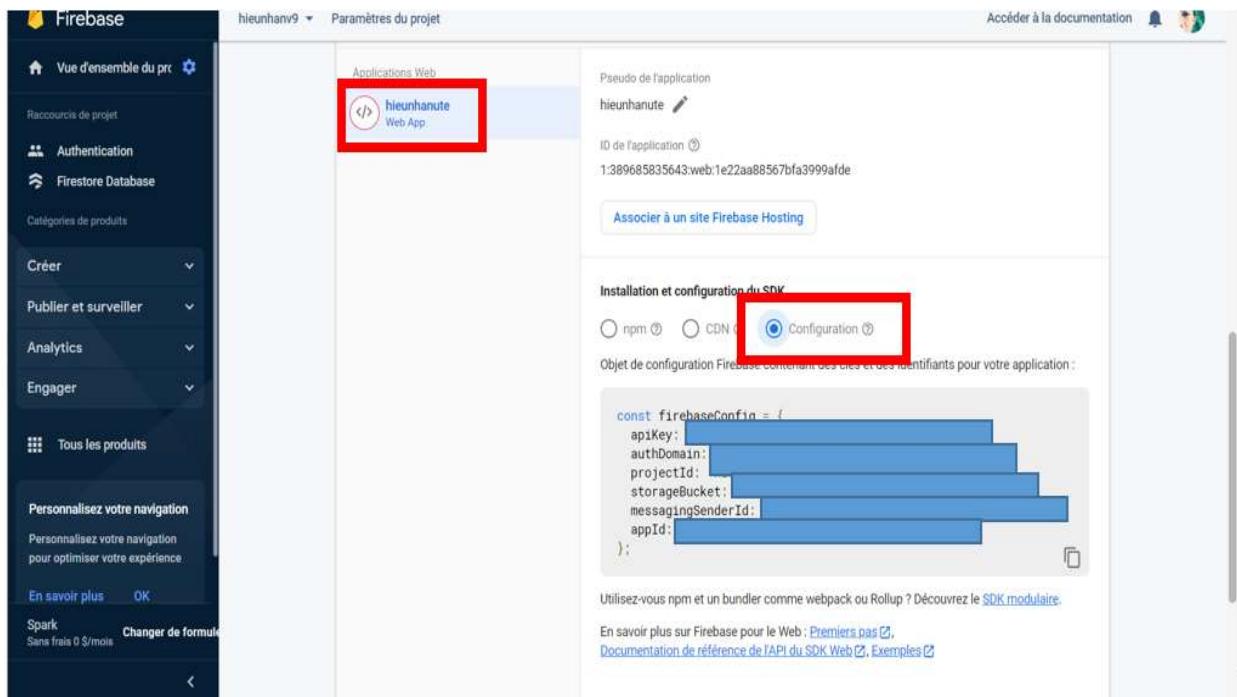
```
<script type="module">
// Import the functions you need from the SDKs you need
import { initializeApp } from "https://www.gstatic.com/firebasejs/9.9.0.firebaseio.js"
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
const firebaseConfig = {
  apiKey: "REDACTED",
  authDomain: "REDACTED",
  projectId: "REDACTED",
  storageBucket: "REDACTED",
  messagingSenderId: "REDACTED",
  appId: "REDACTED"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
</script>
```

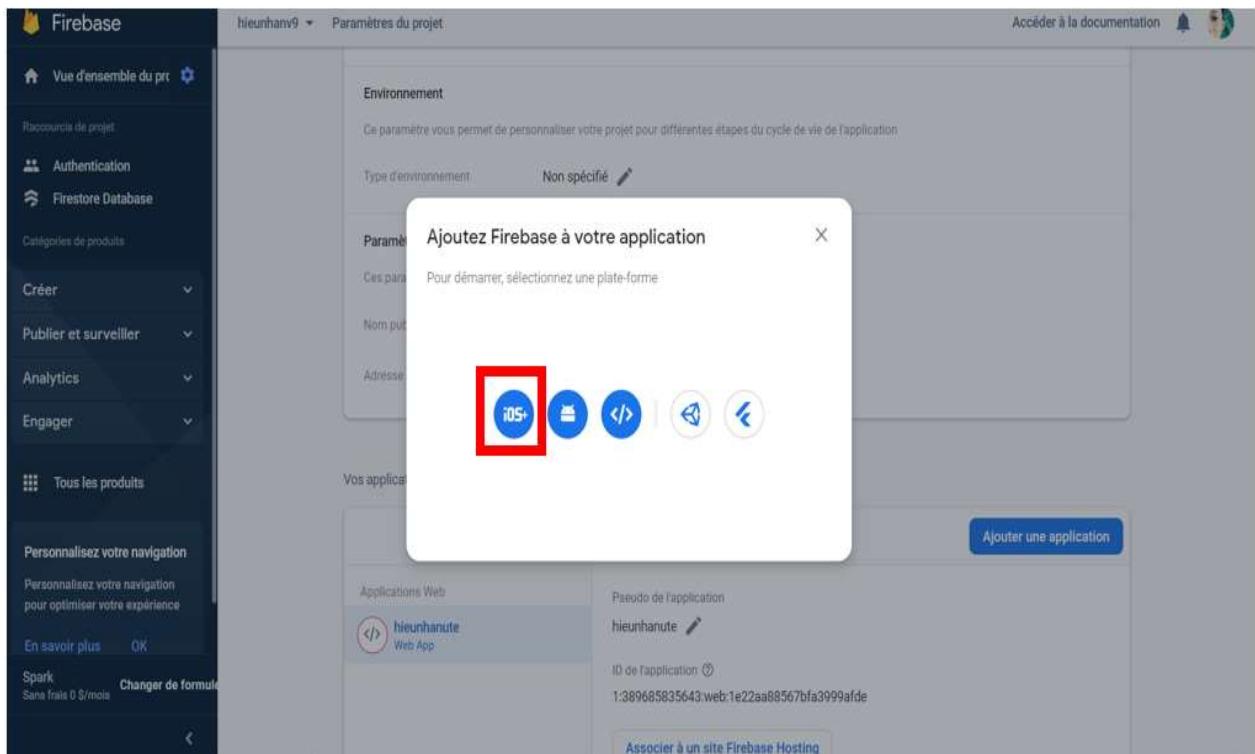
On the right, there's a blue background illustration featuring a smartphone and a laptop connected by a wire, symbolizing connectivity.

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG



Hình 3. 24: Khởi tạo server để esp8266 kết nối với cloud Firestore

3. Khởi tạo Server để thiết bị IOS kết nối tới Cloud Firestore



CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

1 Enregistrer l'application

ID du bundle Apple: com.feeehcmute

Pseudo de l'application (facultatif): Mon application Apple

ID App Store (facultatif): 1123456789

Enregistrer l'application

2 Télécharger le fichier de configuration

Télécharger GoogleService-Info.plist

Déplacez le fichier GoogleService-Info.plist que vous venez de télécharger à la racine de votre projet Xcode, puis ajoutez-le à toutes les cibles.

Suivant

3 Ajouter le SDK Firebase

4 Ajouter le code d'initialisation

5 Étapes suivantes

Ajouter Firebase à votre application Apple

Enregistrer l'application
Télécharger le fichier de configuration
Ajouter le SDK Firebase
Ajouter le code d'initialisation
Étapes suivantes

C'est terminé !

Consultez la [documentation](#) pour découvrir comment vous lancer avec les différents produits Firebase que vous souhaitez utiliser dans votre application.

Vous pouvez également parcourir les [exemples d'applications Firebase](#).

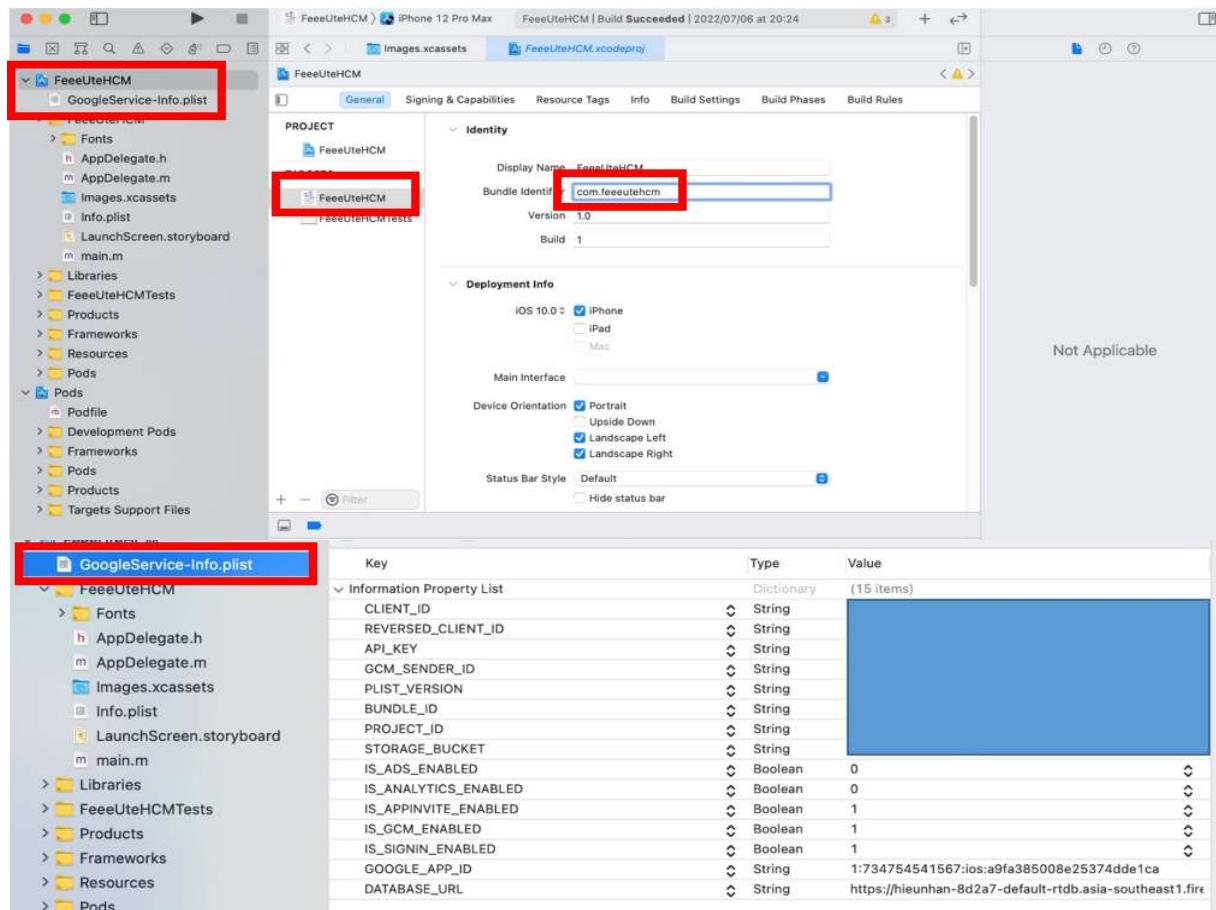
Vous pouvez également accéder à la console pour explorer Firebase.

Accéder à la console

Hình 3. 25: Khởi tạo server cho thiết bị IOS kết nối với cloud Firestore

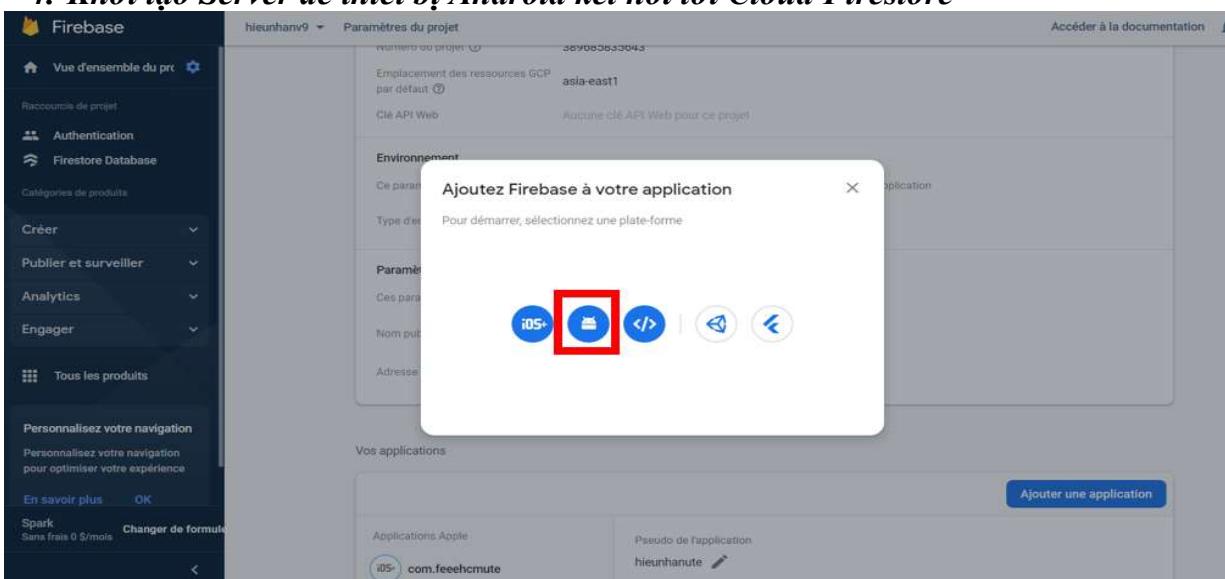
CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

Sau khi tạo xong, bạn cần dowload thư mục GoogleService-info.plist, mở Xcode và cấu hình theo nhu file đã dowload.

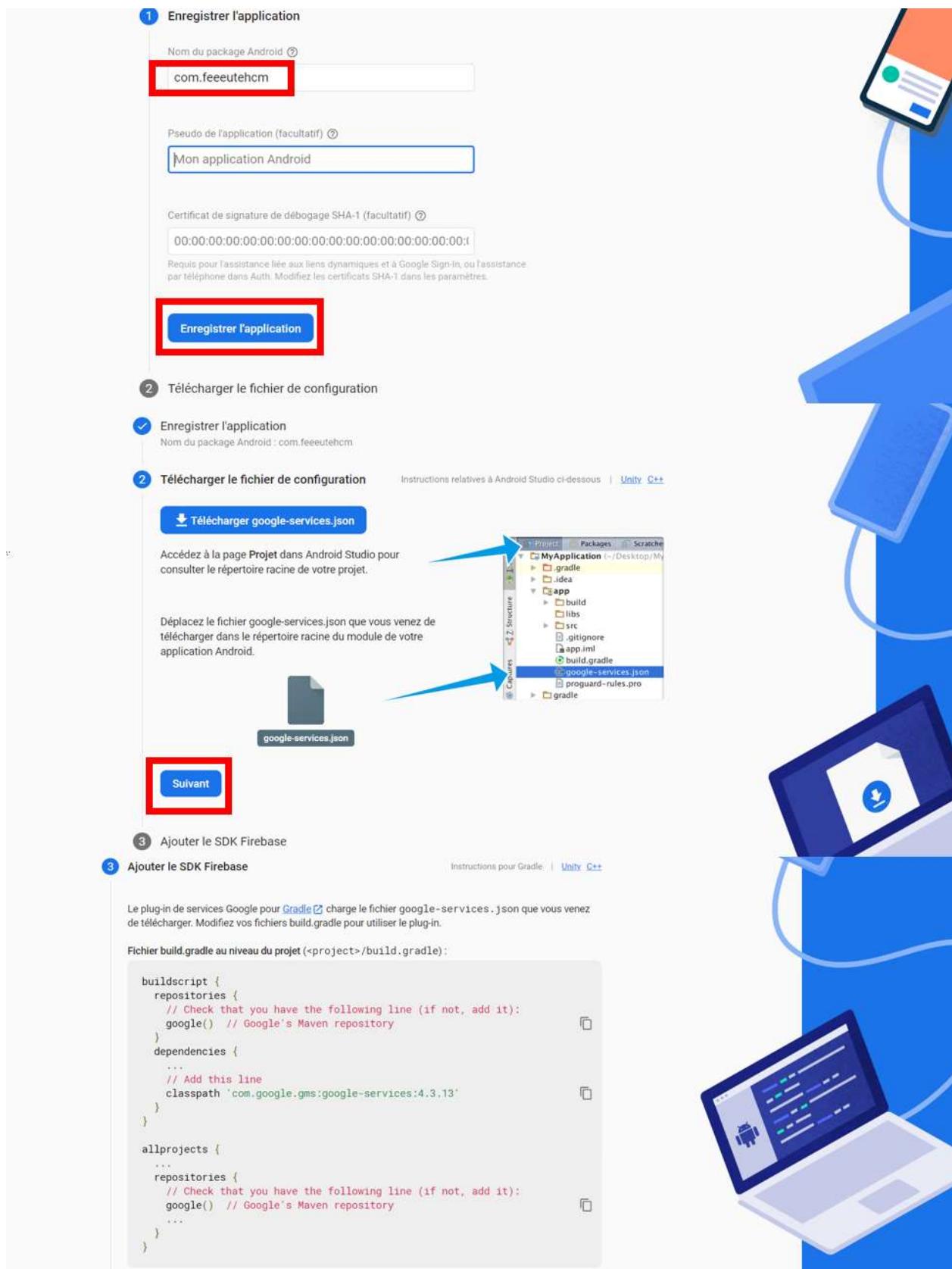


Hình 3. 26: Cấu hình Xcode kết nối tới cloud Firestore

4. Khởi tạo Server để thiết bị Android kết nối tới Cloud Firestore

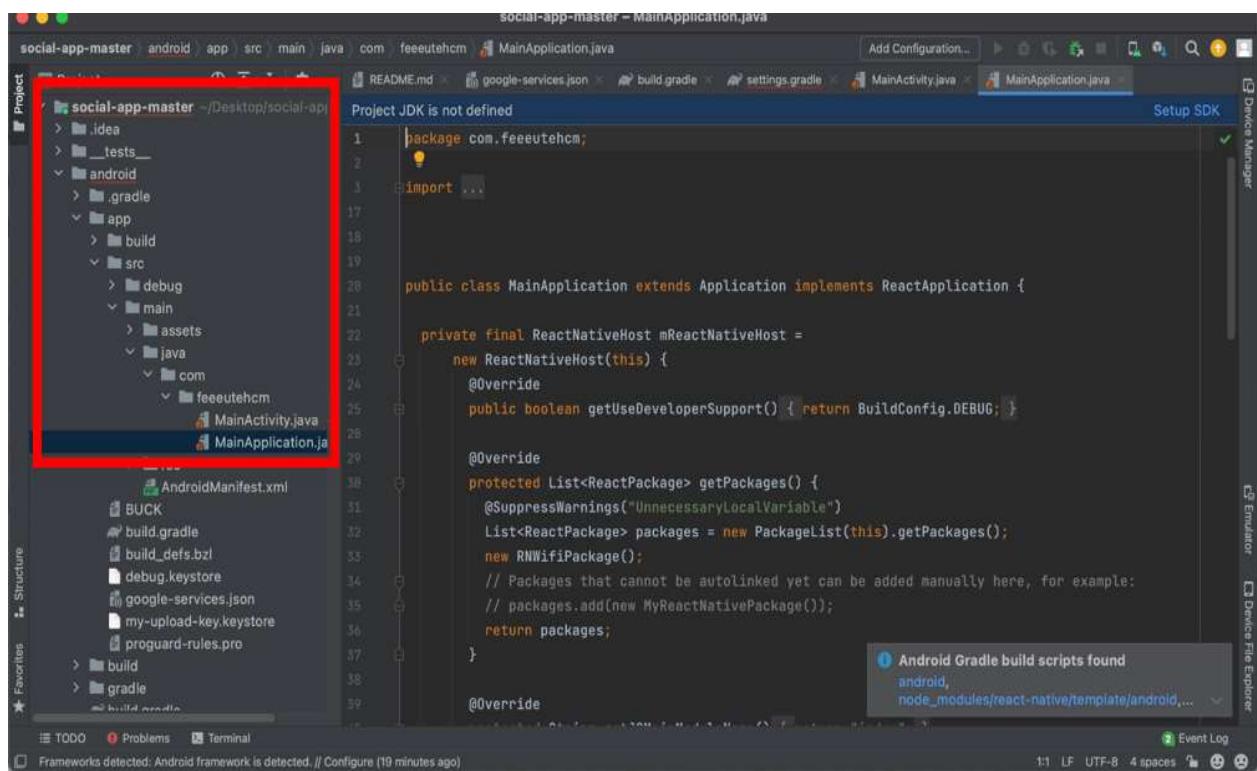


CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

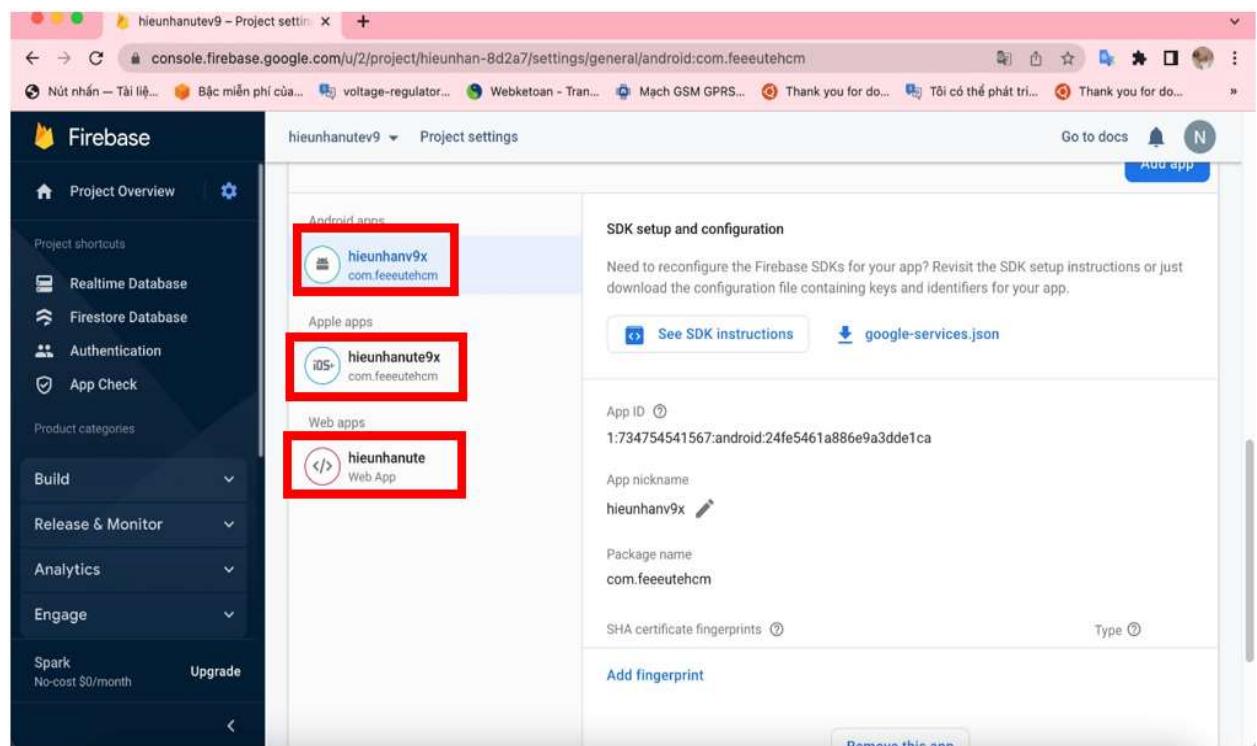


Hình 3.27: Khởi tạo server cho thiết bị Android kết nối tới cloud Firestore

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG



Hình 3. 28: Cấu hình Android kết nối tới cloud Firestore



Hình 3. 29: Cấu hình thành công

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

III. SETUP MÔI TRƯỜNG REACT NATIVE TRÊN MACOS BIG SUR

1. Thiết lập môi trường React Native

a. *Android*

Node & Watchman

We recommend installing Node and Watchman using Homebrew. Run the following commands in a Terminal after installing Homebrew:

```
brew install node  
brew install watchman
```

Hình 3. 30: Thiết lập Node và Watchman Android

Java Development Kit

We recommend installing the OpenJDK distribution called Azul Zulu using Homebrew. Run the following commands in a Terminal after installing Homebrew:

```
brew tap homebrew/cask-versions  
brew install --cask zulu11
```

Hình 3. 31: Thiết lập Java Development Kit

1. Install Android Studio

Download and install Android Studio. While on Android Studio installation wizard, make sure the boxes next to all of the following items are checked:

- Android SDK
- Android SDK Platform
- Android Virtual Device

Then, click "Next" to install all of these components.

If the checkboxes are grayed out, you will have a chance to install these components later on.

Hình 3. 32: Thiết lập Android Studio

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

2. Install the Android SDK

Android Studio installs the latest Android SDK by default. Building a React Native app with native code, however, requires the `Android 12 (S)` SDK in particular. Additional Android SDKs can be installed through the SDK Manager in Android Studio.

To do that, open Android Studio, click on "Configure" button and select "SDK Manager".



The SDK Manager can also be found within the Android Studio "Preferences" dialog, under **Appearance & Behavior → System Settings → Android SDK**.

Select the "SDK Platforms" tab from within the SDK Manager, then check the box next to "Show Package Details" in the bottom right corner. Look for and expand the `Android 12 (S)` entry, then make sure the following items are checked:

- `Android SDK Platform 31`
- `Intel x86 Atom_64 System Image` or `Google APIs Intel x86 Atom System Image` or (for Apple M1 Silicon) `Google APIs ARM 64 v8a System Image`

Next, select the "SDK Tools" tab and check the box next to "Show Package Details" here as well. Look for and expand the "Android SDK Build-Tools" entry, then make sure that `31.0.0` is selected.

Hình 3. 33: Thiết lập Android SDK

3. Configure the ANDROID_SDK_ROOT environment variable

The React Native tools require some environment variables to be set up in order to build apps with native code.

Add the following lines to your `$HOME/.bash_profile` or `$HOME/.bashrc` (if you are using `zsh` then `~/.zprofile` or `~/.zshrc`) config file:

```
export ANDROID_SDK_ROOT=$HOME/Library/Android/sdk
export PATH=$PATH:$ANDROID_SDK_ROOT/emulator
export PATH=$PATH:$ANDROID_SDK_ROOT/platform-tools
```

`.bash_profile` is specific to `bash`. If you're using another shell, you will need to edit the appropriate shell-specific config file.

Type `source $HOME/.bash_profile` for `bash` or `source $HOME/.zprofile` to load the config into your current shell. Verify that `ANDROID_SDK_ROOT` has been set by running `echo $ANDROID_SDK_ROOT` and the appropriate directories have been added to your path by running `echo $PATH`.

Please make sure you use the correct Android SDK path. You can find the actual location of the SDK in the Android Studio "Preferences" dialog, under **Appearance & Behavior → System Settings → Android SDK**.

Hình 3. 34: Thiết lập biến ANDROID_SDK_ROOT

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

Creating a new application

If you previously installed a global `react-native-cli` package, please remove it as it may cause unexpected issues.

React Native has a built-in command line interface, which you can use to generate a new project. You can access it without installing anything globally using `npx`, which ships with Node.js. Let's create a new React Native project called "AwesomeProject":

```
npx react-native init AwesomeProject
```

Hình 3. 35: Tạo một dự án mới trên Android

Running your React Native application

Step 1: Start Metro

First, you will need to start Metro, the JavaScript bundler that ships with React Native. Metro "takes in an entry file and various options, and returns a single JavaScript file that includes all your code and its dependencies."—[Metro Docs](#)

To start Metro, run `npx react-native start` inside your React Native project folder:

```
npx react-native start
```

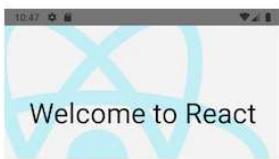
Hình 3. 36: Chạy React Native trên Android

Step 2: Start your application

Let Metro Bundler run in its own terminal. Open a new terminal inside your React Native project folder. Run the following:

```
npx react-native run-android
```

If everything is set up correctly, you should see your new app running in your Android emulator shortly.



Step One

Edit `App.js` to change this screen and then come back to see your edits.

See Your Changes

Double tap `R` on your keyboard to reload your app's code.

Debug

Press `menu button` or `Shake` your device to open the React Native debug menu.

Learn More

Read the docs to discover what to do next:

Explains a Hello World

Hình 3. 37: Khởi động ứng dụng trên máy ảo Android

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

b. iOS

Node & Watchman

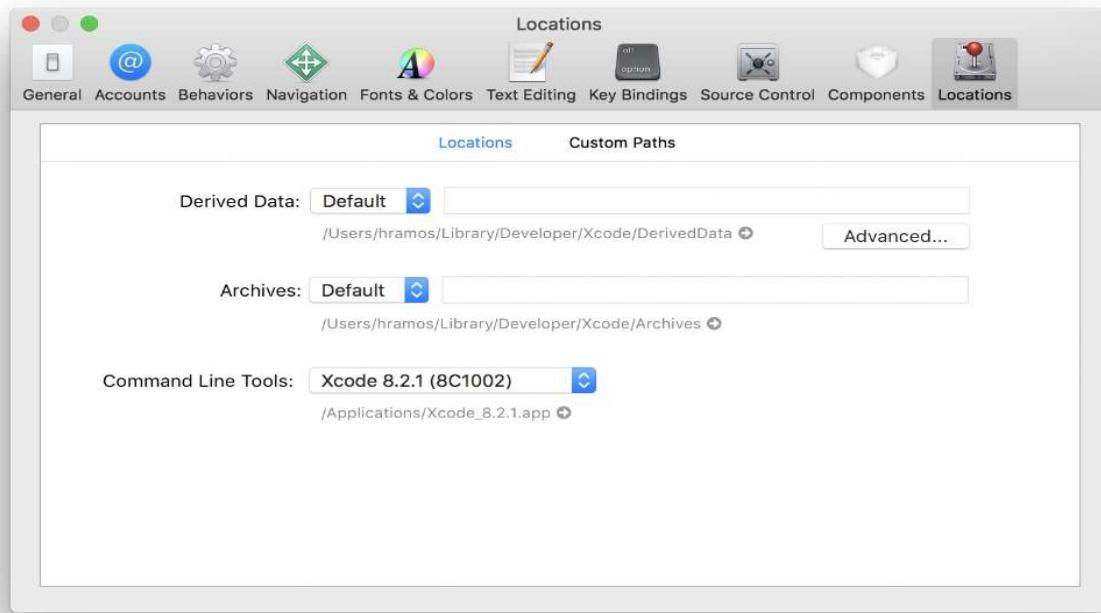
We recommend installing Node and Watchman using [Homebrew](#). Run the following commands in a Terminal after installing Homebrew:

```
brew install node  
brew install watchman
```

Hình 3. 38: Thiết lập Node và Watchman IOS

Command Line Tools

You will also need to install the Xcode Command Line Tools. Open Xcode, then choose "Preferences..." from the Xcode menu. Go to the Locations panel and install the tools by selecting the most recent version in the Command Line Tools dropdown.



Hình 3. 39: Cấu hình Xcode

Installing an iOS Simulator in Xcode

To install a simulator, open **Xcode > Preferences...** and select the **Components** tab. Select a simulator with the corresponding version of iOS you wish to use.

CocoaPods

[CocoaPods](#) is built with Ruby and it will be installable with the default Ruby available on macOS.

Using the default Ruby available on macOS will require you to use `sudo` when installing gems. (This is only an issue for the duration of the gem installation, though.)

```
sudo gem install cocoapods
```

Hình 3. 40: Thiết lập mô phỏng IOS trong Xcode

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

Creating a new application

If you previously installed a global `react-native-cli` package, please remove it as it may cause unexpected issues.

You can use React Native's built-in command line interface to generate a new project. Let's create a new React Native project called "AwesomeProject":

```
npx react-native init AwesomeProject
```

Hình 3.41: Tạo một dự án mới trên IOS

Running your React Native application

Step 1: Start Metro

First, you will need to start Metro, the JavaScript bundler that ships with React Native. Metro "takes in an entry file and various options, and returns a single JavaScript file that includes all your code and its dependencies."—[Metro Docs](#)

To start Metro, run `npx react-native start` inside your React Native project folder:

```
npx react-native start
```

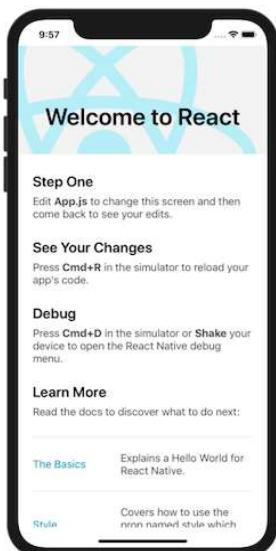
Hình 3.42: Chạy React Native trên IOS

Step 2: Start your application

Let Metro Bundler run in its own terminal. Open a new terminal inside your React Native project folder. Run the following:

```
npx react-native run-ios
```

You should see your new app running in the iOS Simulator shortly.



Hình 3.43: Khởi động ứng dụng trên máy ảo IOS

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

2. Thiết lập React Native Firebase

```
# Using npm  
npm install --save @react-native-firebase/app
```

```
# Using Yarn  
yarn add @react-native-firebase/app
```

Hình 3. 44: Cài đặt qua NPM

a. Thiết lập kết nối Android

The debug signing certificate is optional to use Firebase with your app, but is required for Dynamic Links, Invites and Phone Authentication. To generate a certificate run `cd android && ./gradlew signingReport`. This generates two variant keys. You have to copy both 'SHA1' and 'SHA-256' keys that belong to the 'debugAndroidTest' variant key option. Then, you can add those keys to the 'SHA certificate fingerprints' on your app in Firebase console.

Hình 3. 45: Tạo thông tin đăng nhập Android

First, add the `google-services` plugin as a dependency inside of your `/android/build.gradle` file:

```
buildscript {  
    dependencies {  
        // ... other dependencies  
        classpath 'com.google.gms:google-services:4.3.12'  
        // Add me --- /\  
    }  
}
```

Lastly, execute the plugin by adding the following to your `/android/app/build.gradle` file:

```
apply plugin: 'com.android.application'  
apply plugin: 'com.google.gms.google-services' // <- Add this line
```

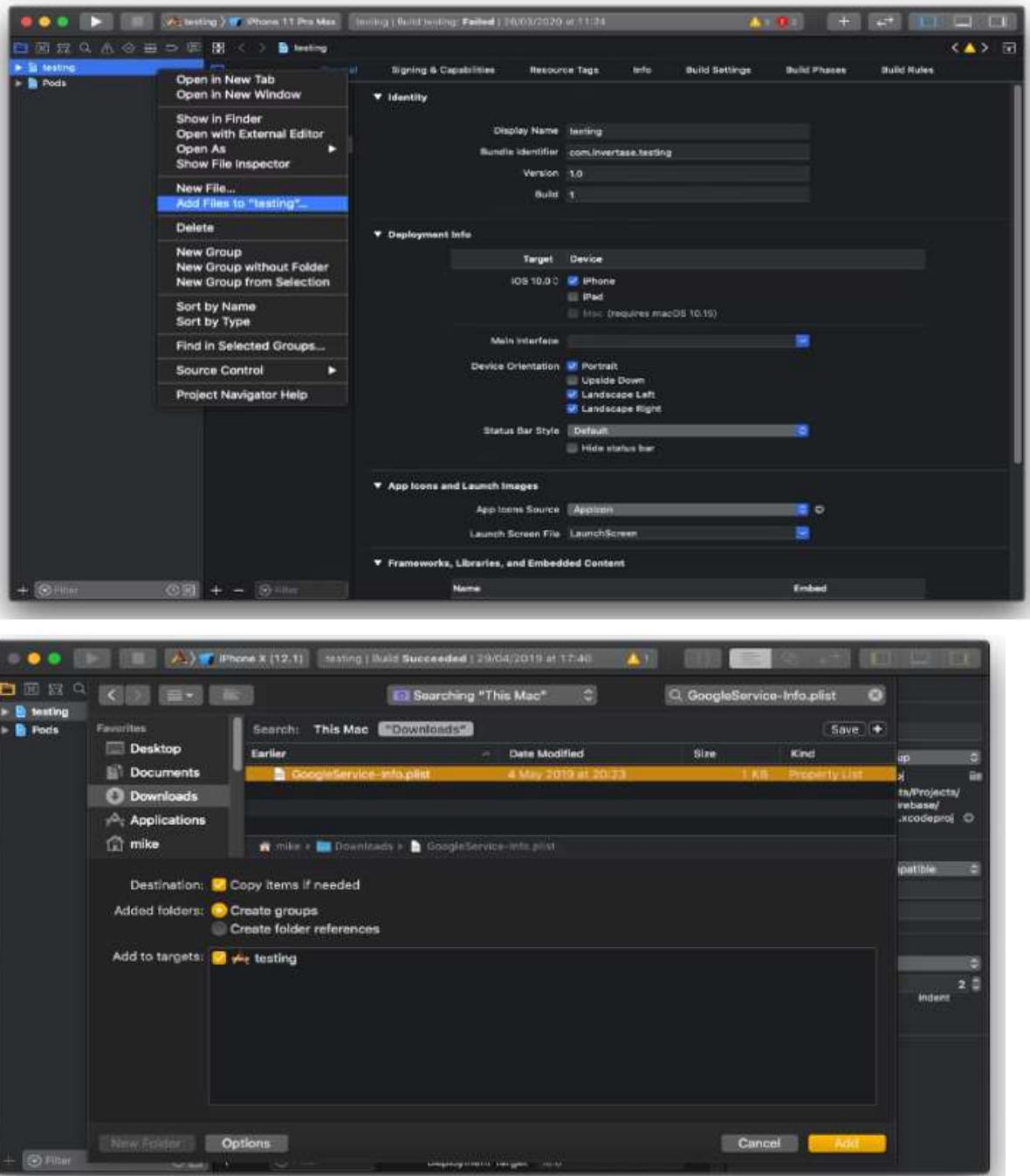
Hình 3. 46: Cấu hình Firebase bằng thông tin đăng nhập Android

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

b. Thiết lập kết nối iOS

Using Xcode, open the projects `/ios/{ projectName }.xcodeproj` file (or `/ios/{ projectName }.xcworkspace` if using Pods).

Right click on the project name and "Add files" to the project, as demonstrated below:



Hình 3.47: Tạo thông tin đăng nhập iOS

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

At the top of the file, import the Firebase SDK:

```
#import <Firebase.h>
```

Within your existing `didFinishLaunchingWithOptions` method, add the following to the top of the method:

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *  
    // Add me --- /\  
    [FIRApp configure];  
    // Add me --- /\  
    // ...  
}
```

Hình 3. 48: Cấu hình Firebase bằng thông tin đăng nhập IOS

3. Thiết lập React Native Cloud Firestore

a. Thiết lập kết nối

```
# Install & setup the app module  
yarn add @react-native-firebase/app  
  
# Install the firestore module  
yarn add @react-native-firebase/firestore  
  
# If you're developing your app using iOS, run this command  
cd ios/ && pod install
```

Hình 3. 49: Cài đặt React Native Cloud Firestore

b. Cách sử dụng

To access the documents within a `QuerySnapshot`, call the `forEach` method:

```
import firestore from '@react-native-firebase/firestore';  
  
firestore()  
  .collection('Users')  
  .get()  
  .then(querySnapshot => {  
    console.log('Total users: ', querySnapshot.size);  
  
    querySnapshot.forEach(documentSnapshot => {  
      console.log('User ID: ', documentSnapshot.id, documentSnapshot.data());  
    });  
  });
```

Hình 3. 50: Truy vấn đến tập tài liệu trên Cloud Firestore

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

```
firestore()
  .collection('Users')
  .orderBy('age', 'desc')
  .startAt(18)
  .endAt(30)
  .get()
  .then(querySnapshot => {
    /* ... */
  });
});
```

Hình 3. 51: Bắt đầu và kết thúc truy vấn

```
import firestore from '@react-native-firebase/firestore';

firestore()
  .collection('Users')
  .add({
    name: 'Ada Lovelace',
    age: 30,
  })
  .then(() => {
    console.log('User added!');
  });
});
```

```
import firestore from '@react-native-firebase/firestore';

firestore()
  .collection('Users')
  .doc('ABC')
  .set({
    name: 'Ada Lovelace',
    age: 30,
  })
  .then(() => {
    console.log('User added!');
  });
});
```

Hình 3. 52: Thêm và cập nhập tài liệu lên Cloud Firestore

4. Thiết lập React Native Authentication

a. Thiết lập kết nối

```
# Install & setup the app module
yarn add @react-native-firebase/app

# Install the authentication module
yarn add @react-native-firebase/auth

# If you're developing your app using iOS, run this command
cd ios/ && pod install
```

Hình 3. 53: Cài đặt React Native Authentication

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

b. Cách sử dụng

```
import auth from '@react-native-firebase/auth';

auth()
  .createUserWithEmailAndPassword('jane.doe@example.com', 'SuperSecretPassword!')
  .then(() => {
    console.log('User account created & signed in!');
  })
  .catch(error => {
    if (error.code === 'auth/email-already-in-use') {
      console.log('That email address is already in use!');
    }

    if (error.code === 'auth/invalid-email') {
      console.log('That email address is invalid!');
    }

    console.error(error);
  });
});
```

Hình 3. 54: Đăng nhập bằng email password Authentication

```
import auth from '@react-native-firebase/auth';

auth()
  .signOut()
  .then(() => console.log('User signed out!'));
```

Hình 3. 55: Đăng xuất Authentication

5. Thiết lập React Native Vector Icon

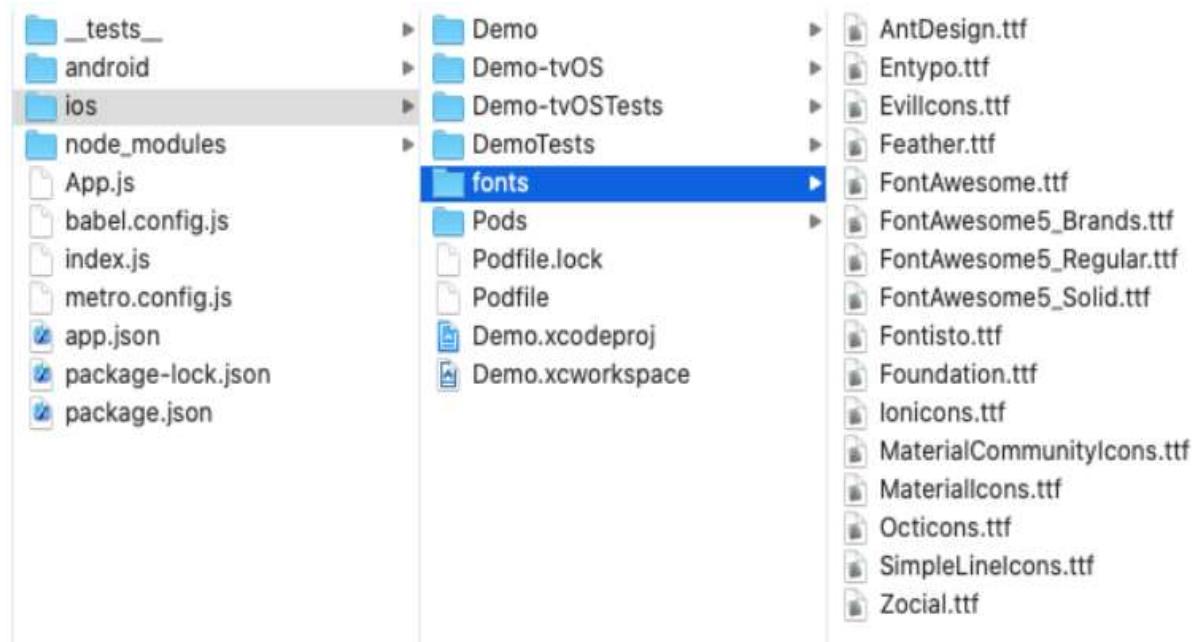
a. Thiết lập kết nối

- Chạy dòng lệnh: `$ npm install --save react-native-vector-icons` trên terminal
- Đối với mỗi nền tảng (iOS / Android / Windows) bạn định sử dụng, hãy làm theo một trong các tùy chọn cho nền tảng tương ứng.

b. Cách sử dụng

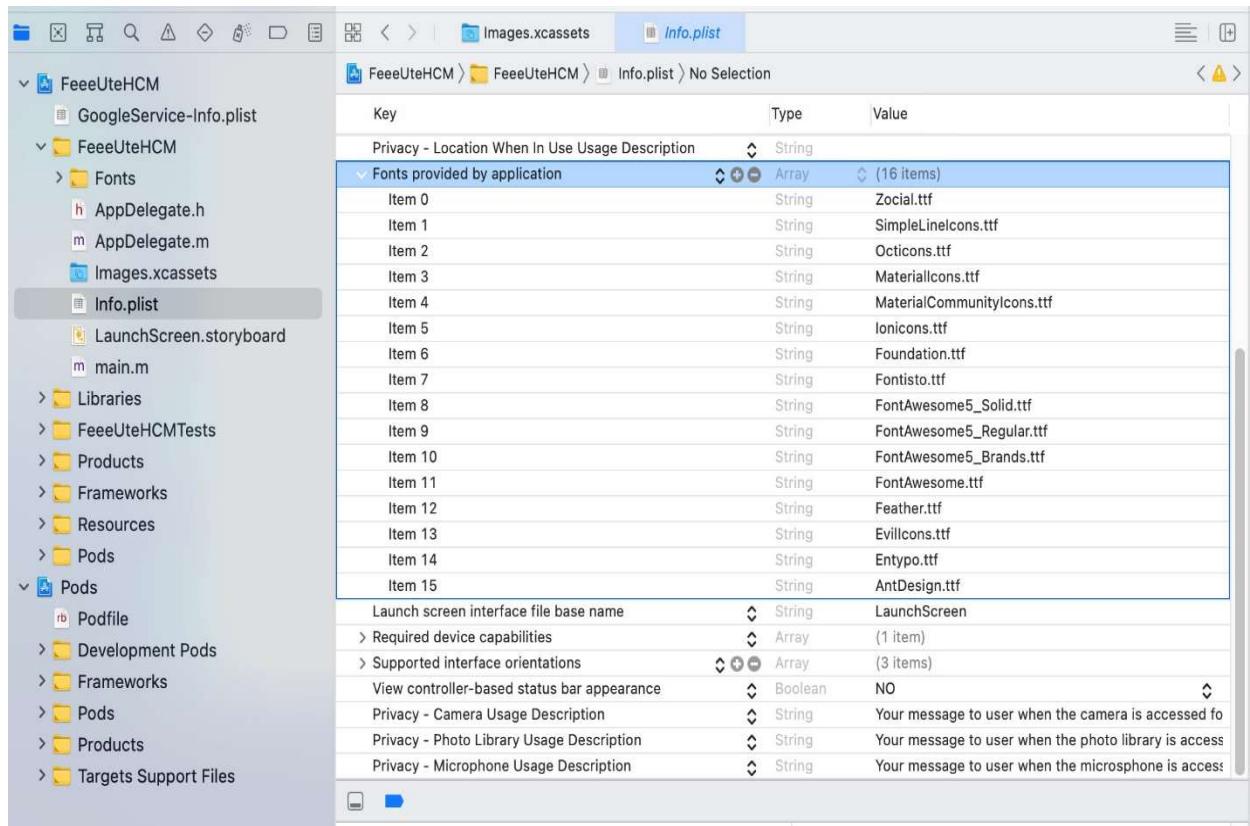
Tạo một thư mục fonts trong IOS và sao chép tất cả phông chữ từ node_modules/react-native-vector-icon/Fonts vào đó.

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG



Hình 3. 56: Tạo thư mục font trong mục IOS

Khi đã tạo xong, ta truy cập vào file dự án và thêm những font file sử dụng vào



Hình 3. 57: Thêm font vào dự án

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

6. Cấu trúc thư mục trong Framework React Native

The screenshot shows a code editor interface with the following details:

- EXPLORER:** Shows the project structure:
 - SOCIAL-APP-MASTER 2
 - _tests
 - > android (highlighted)
 - > ios
 - > node_modules (highlighted)
 - src
 - common
 - components
 - constants
 - data
 - firebase
 - image
 - screens
 - store
 - styles
 - buckconfig
 - editorconfig
 - .eslintrc.js
 - flowconfig
 - gitattributes
 - gignore
 - prettierrc.js
 - watchmanconfig
 - App.js
 - app.json
 - babel.config.js
 - index.js
 - metro.config.js
 - package-lock.json
 - package.json (highlighted)
 - react-native.config.js
 - README.md
 - yarn-error.log (highlighted)
 - yarn.lock
- JSS Feeds.js**: A JavaScript file containing code related to feeds.
- package.json**: The package configuration file.
- yarn-error.log**: A log file showing an error message about a missing Firebase dependency.

Hình 3. 58: Cấu trúc thư mục trong Framework React Native

IV. YÊU CẦU CỦA HỆ THỐNG

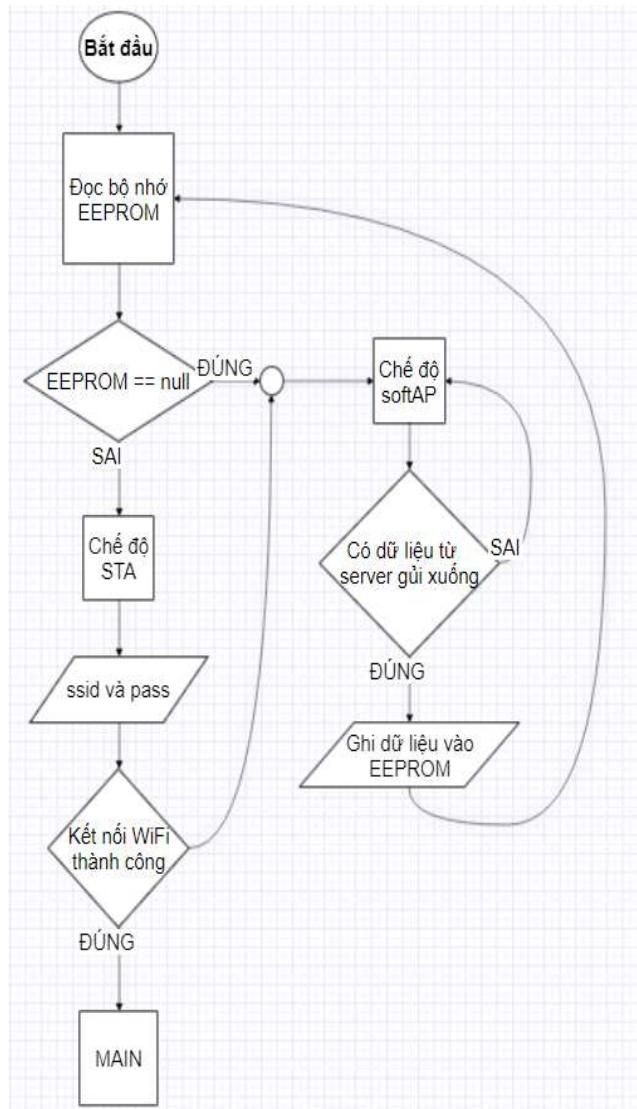
1. ESP8266 kết nối tới Server

Khi bắt đầu hệ thống, ESP8266 sẽ đọc dữ liệu từ bộ nhớ EEPROM:

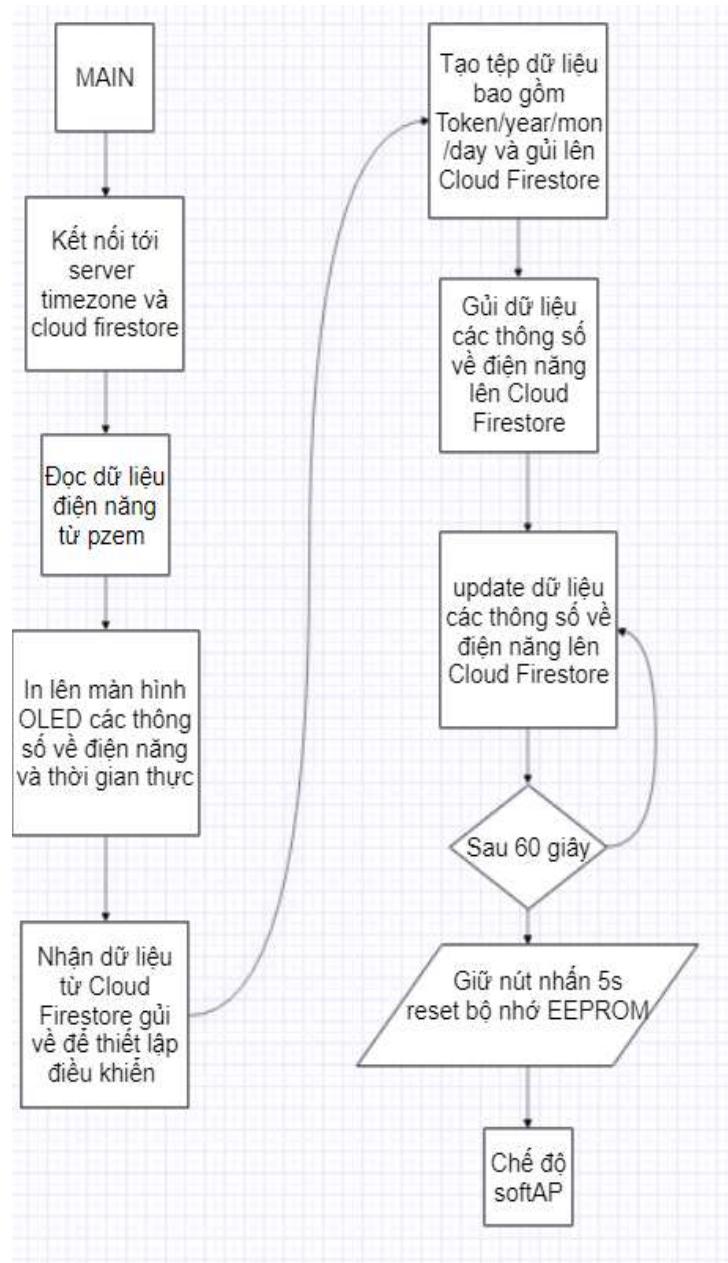
- Khi có dữ liệu từ bộ nhớ EEPROM, ESP8266 sẽ đọc dữ liệu từ EEPROM ra. Lúc này ESP8266 sẽ ở chế độ STA lấy dữ liệu từ EEPROM để kết nối tới WiFi.
 - Nếu kết nối WiFi thất bại, ESP8266 sẽ chuyển sang chế độ softAP
 - Khi kết nối thành công, ESP8266 sẽ đọc các dữ liệu điện năng từ modul PZEM-004T sau đó kết nối tới Cloud Firestore để khởi tạo một tập tin bao gồm: Token/năm/ tháng/ngày. Mọi biến dữ liệu sẽ được gửi vào tệp ngày và lưu trữ trên Firestore.

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

- Khi không có dữ liệu từ bộ nhớ EEPROM, ESP8266 sẽ ở chế độ softAP, để server có thể kết nối tới và gửi dữ liệu bao gồm: Token, ssid và password WiFi xung quanh, số phòng. Khi đã nhận được dữ liệu từ server, ESP8266 sẽ lưu lại những dữ liệu đó trong EEPROM và lại quay lại đọc dữ liệu từ EEPROM và thực hiện lại từ đầu.



CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

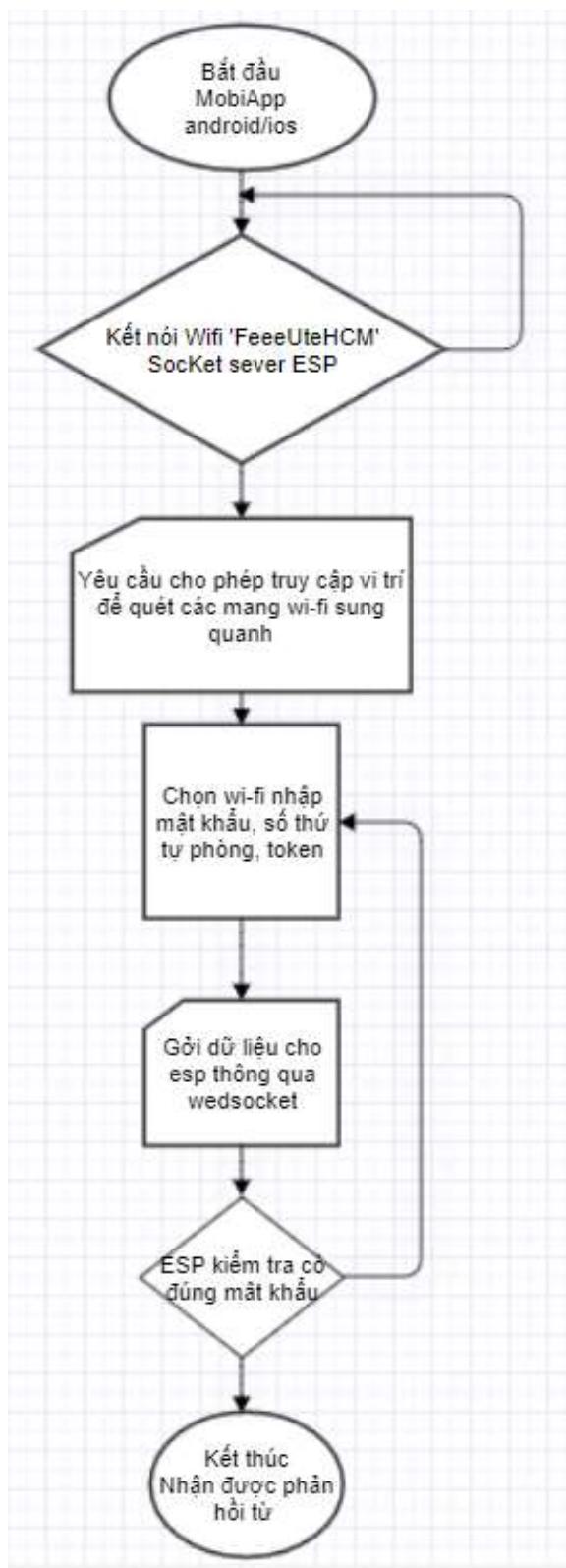


Hình 3. 59: Lưu đồ giải thuật ESP8266 kết nối đến Server

Việc lưu lại những dữ liệu trong bộ nhớ EEPROM sẽ khắc phục được việc mất điện đột ngột từ điện lực. Vì khi mất điện, ESP sẽ tắt việc này sẽ làm mất dữ liệu từ server gửi xuống, khi có điện trở lại ESP sẽ thực hiện lại chương trình từ đầu nên việc gửi dữ liệu lại cho ESP từ App là không hiệu quả. Vì vậy việc lưu trữ dữ liệu vào bộ nhớ EEPROM là cần thiết trong lúc này.

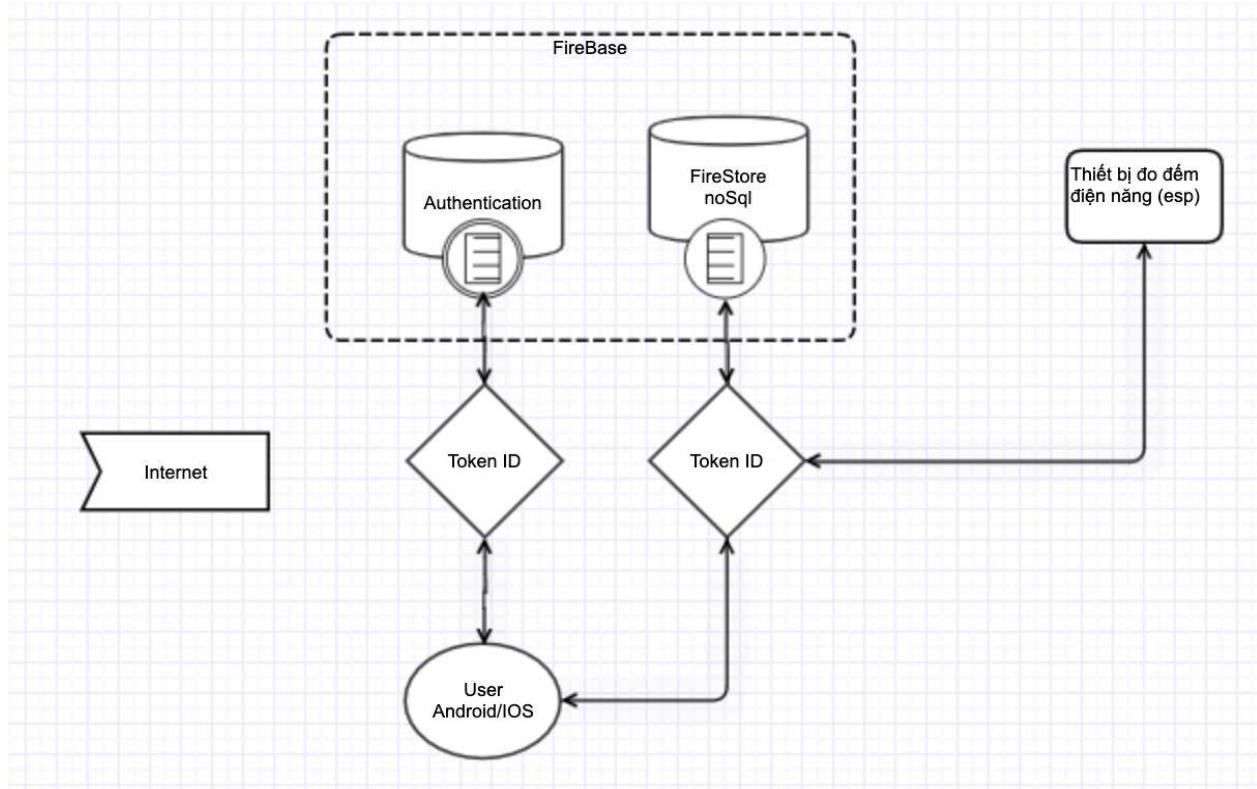
CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

2. App kết nối đến Server



Hình 3. 60: Lưu đồ giải thuật cấu hình mật khẩu WiFi cho thiết bị bằng App

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

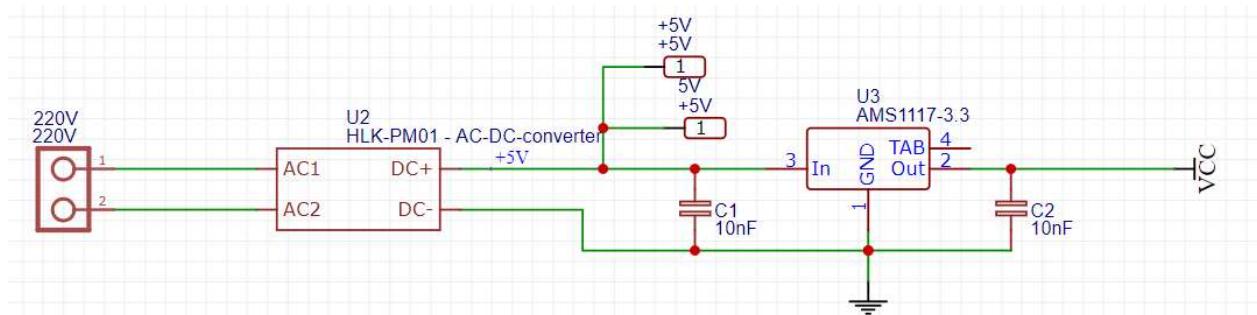


Hình 3. 61: App kết nối Server với người dùng

V. THIẾT KẾ PHẦN CỨNG

1. Phần mạch nguồn

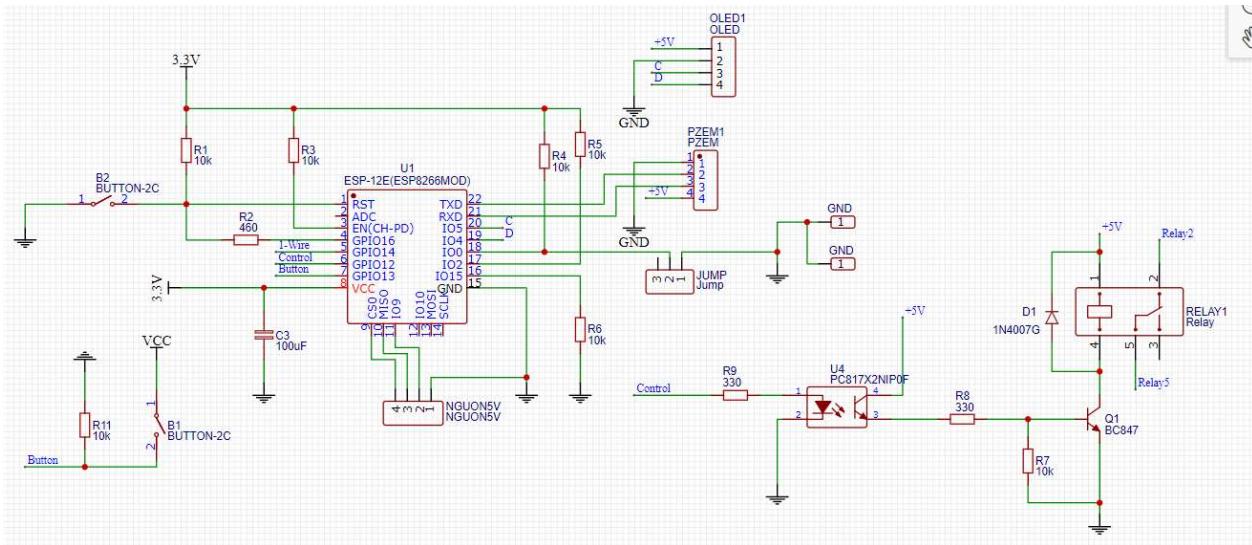
- Mạch sẽ cấp nguồn 220V để cung cấp điện năng cho PZEM-004T hoạt động sau đó từ nguồn 220V kết nối tới Hilink để giảm điện áp từ 220V xuống 5V để cấp nguồn cho quạt tản nhiệt, Relay ngắt tải và màn hình OLED.
- Sử dụng IC ổn áp AMS1117 giảm điện áp từ 5V xuống 3.3 để cấp nguồn cho ESP-12F



Hình 3. 62: Mạch nguồn cho ESP-12F

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

2. Phần ESP - 12F



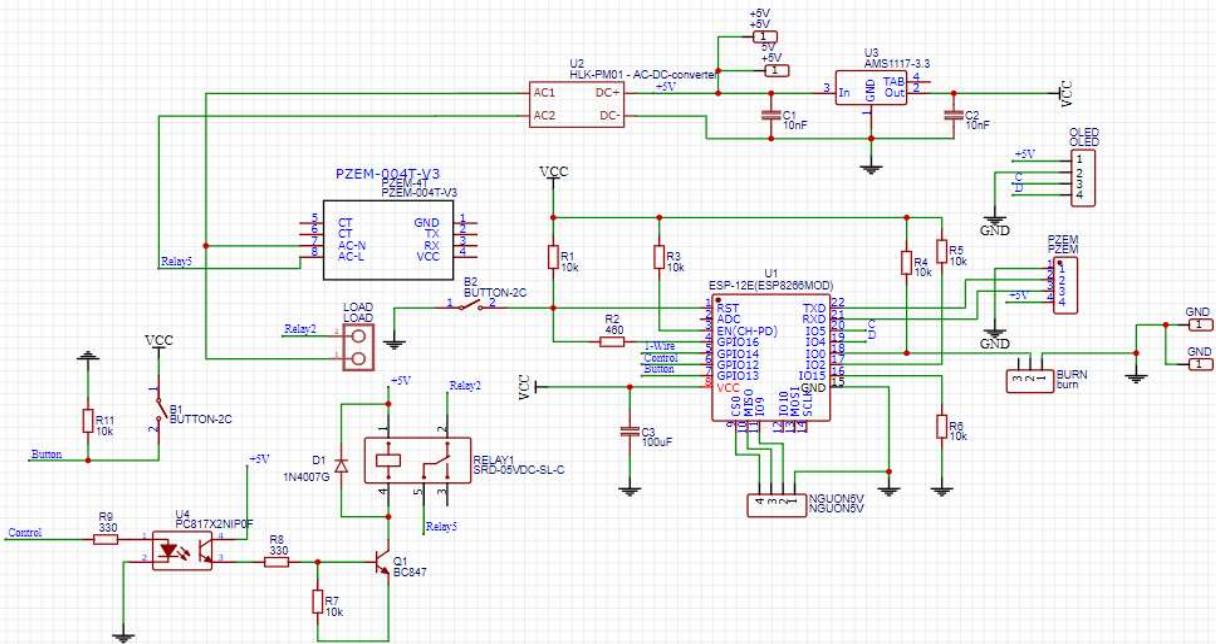
Hình 3.63: Sơ đồ mạch ESP – 12F kết nối với các phần cứng khác

- Ở chân RST được nối với B2 là nút nhấn là reset của ESP8266
 - Ở IO0 được nối với một jump, khi nạp code cho ESP8266 ta cần nối lại để đưa chân IO0 xuống mức thấp. Khi nạp xong ta tháo jump ra để ESP trở lại trạng thái hoạt động bình thường.
 - Nguồn 5V được kết nối để báo nguồn điện khi sử dụng thiết bị.
 - Chân IO4 và IO5 tương ứng với SCL và SDL được sử dụng để kết nối đến màn hình OLED. Màn hình OLED sẽ hiển thị các thông số như ngày, tháng, năm, giờ, phút, giây và chế độ WiFi on hay off. Ngoài ra còn hiển thị các thông số điện năng lên trên màn hình.
 - Chân TX và RX được kéo ra jump với 2 mục đích sử dụng:
 - 1 là sử dụng cho việc nạp code cho ESP – 12F.
 - 2 là sử dụng để đọc dữ liệu từ Modul đo điện năng PZEM-004T về ESP8266.
 - Chân GPIO13 kết nối với nút nhấn với mục đích sử dụng để Reset giữ liệu trong bộ nhớ EEPROM mà mình đã lưu từ trước.
 - Chân GPIO12 được kết nối với Relay với mục đích sử dụng là để ngắt tải khi cần thiết.
 - Các chân còn lại được cấu hình theo đúng chức năng nhà sản xuất đưa ra.

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

3. Sơ đồ mạch Schematic và PCB

a. Sơ đồ mạch schematic



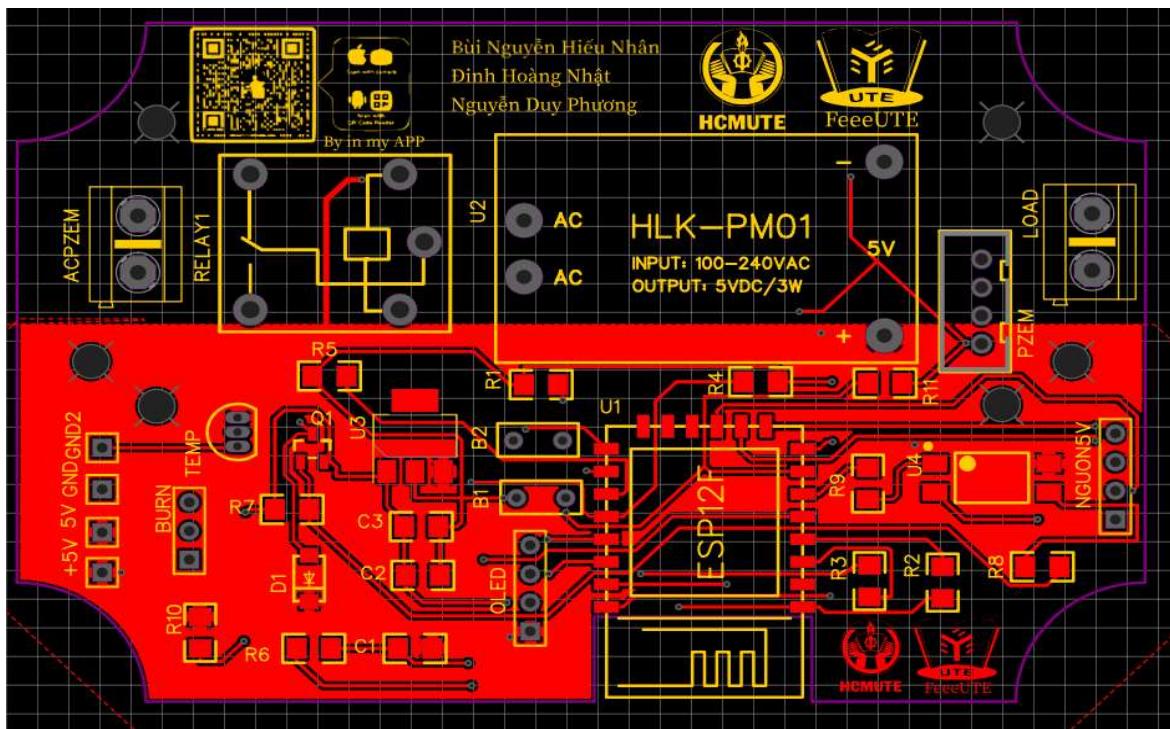
Hình 3. 64: Sơ đồ mạch Schematic

1	HILINK 5V 3W
2	PZEM – 004T
3	Relay
4	OLED 1.3in
5	ESP-12F
6	LED báo nguồn
7	AMSG1117 – 3.3V
8	PC817
9	BC847
10	Diode 1N4007
11	Tụ gián 1206 10nF
12	Tụ gián 1206 100uF
13	Nút nhấn 2 chân
14	Điện trở gián 10k 1206
15	Điện trở gián 330R 1206
16	Điện trở gián 4.7k 1206
17	Điện trở gián 470R 1206
18	Domino 5mm

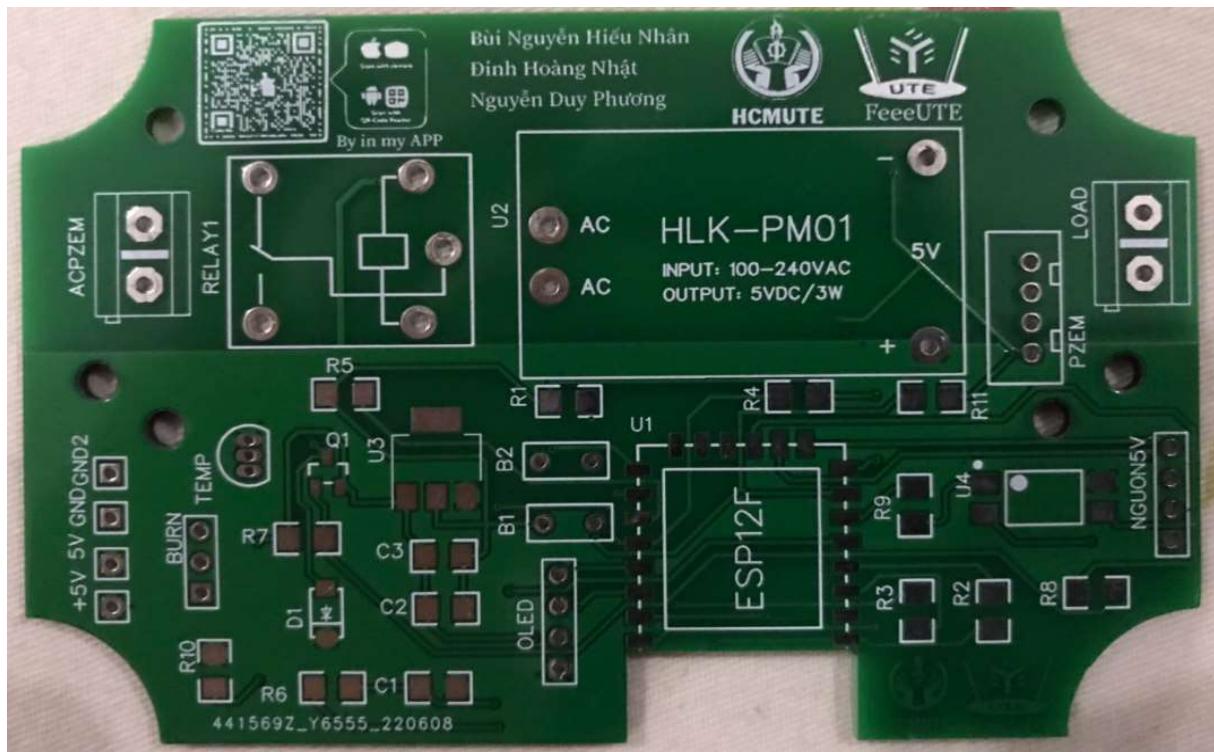
Bảng 3. 2: Bảng linh kiện sử dụng trong mạch in

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

d. Sơ đồ mạch PCB và hình ảnh thực tế

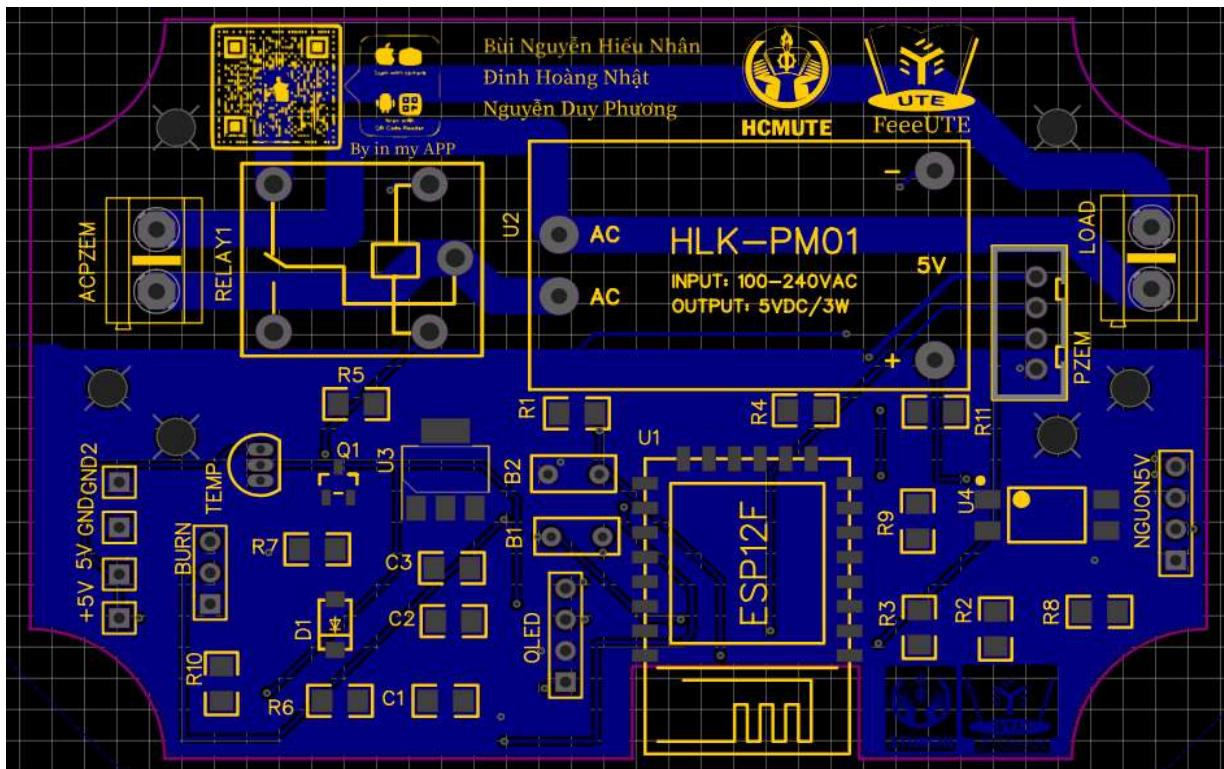


Hình 3. 65: Thiết kế mặt trước PCB trên EasyEDA

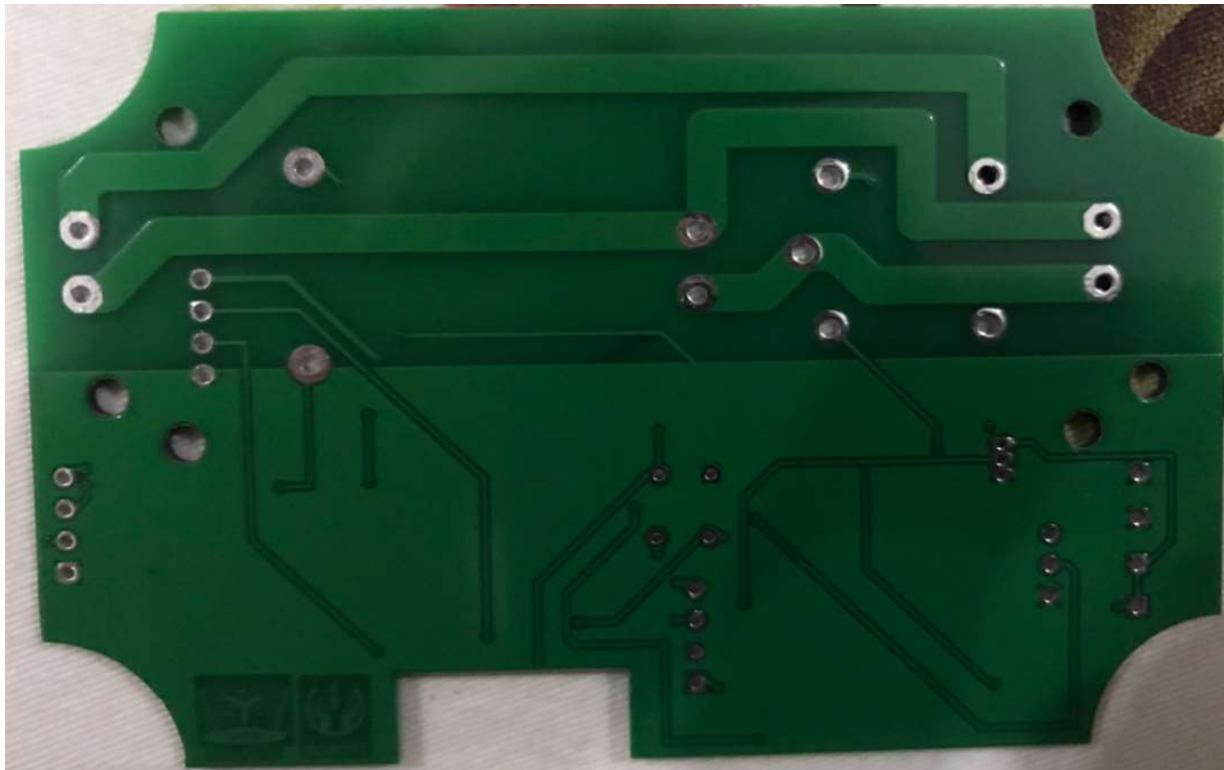


Hình 3. 66: Hình ảnh thực tế mặt trước của PCB

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG



Hình 3. 67: Thiết kế mặt sau PCB trên EasyEDA



Hình 3. 68: Hình ảnh thực tế mặt sau của PCB

CHƯƠNG 4: VẬN HÀNH HỆ THỐNG

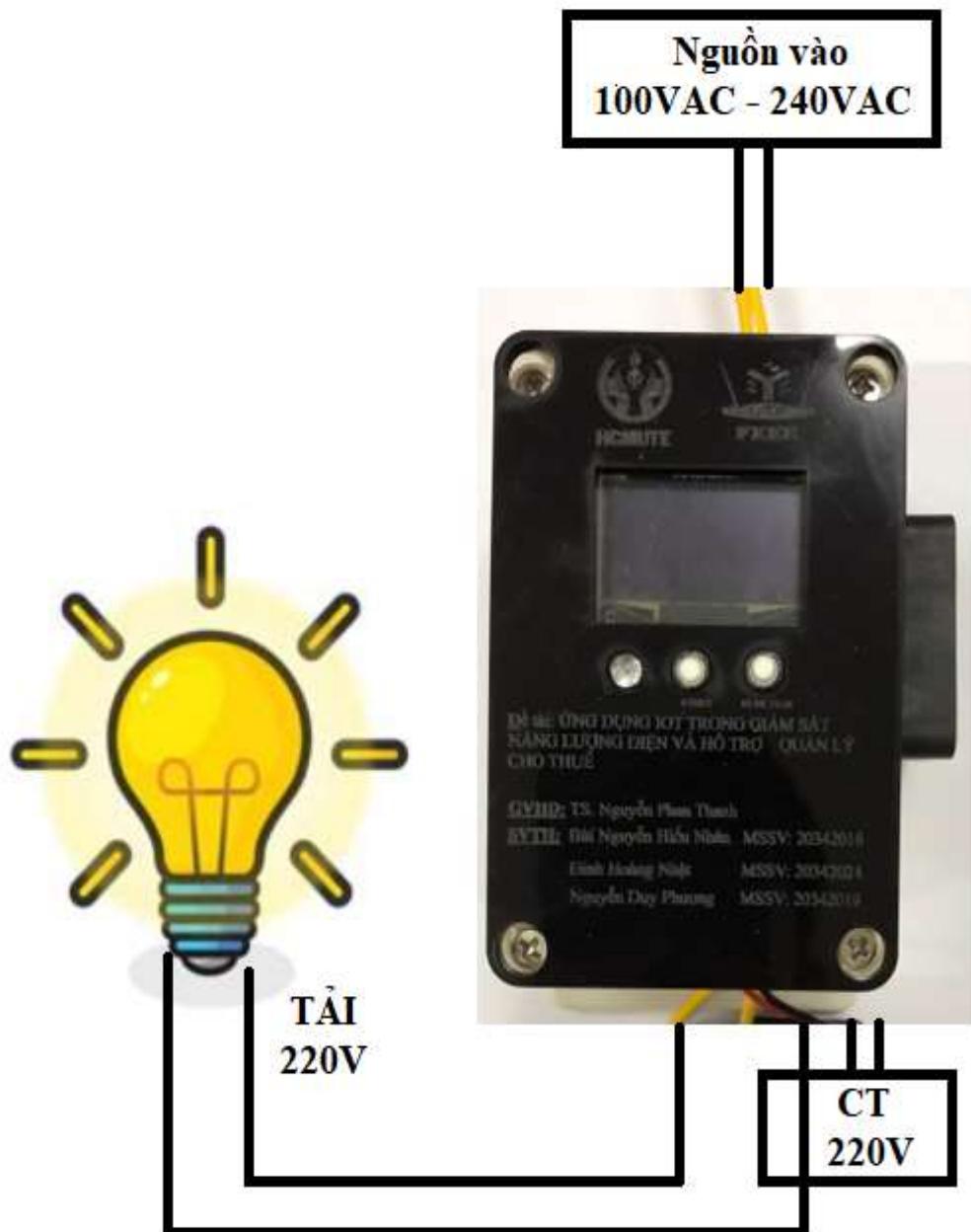
I. PHẦN NHÚNG

1. Mô tả hoạt động của thiết bị

Khi khởi động mô hình phần cứng, ESP8266 sẽ đọc dữ liệu từ trong bộ nhớ EEPROM:

- Nếu có dữ liệu từ bộ nhớ EEPROM, ESP8266 sẽ đọc dữ liệu từ EEPROM ra. Lúc này ESP8266 sẽ ở chế độ STA lấy dữ liệu từ EEPROM bao gồm ssid và password đã cài đặt từ trước để kết nối tới WiFi.
 - Nếu kết nối WiFi thất bại, ESP8266 sẽ thoát chế độ STA và chuyển sang chế độ softAP để lấy dữ liệu từ App gửi xuống.
 - Nếu kết nối WiFi thành công, ESP8266 sẽ kết nối đến server thời gian thực và Cloud Firestore. Sau đó ESP8266 sẽ đọc các dữ liệu điện năng từ modul PZEM-004T và khởi tạo một tập dữ liệu bao gồm: Token/năm/tháng/ngày trên Cloud Firestore. Khi tạo tập dữ liệu thành công, mọi biến dữ liệu điện năng sẽ được lưu vào tệp ngày và lưu trữ trên Cloud Firestore. Sau 60 giây mọi biến dữ liệu điện năng sẽ được cập nhập 1 lần. Ngoài ra ESP8266 cũng sẽ lấy dữ liệu từ Cloud Firestore về để điều khiển ngắt tải và reset năng lượng trên pzem nếu như biến dữ liệu trên Cloud Firestore thay đổi.
- Khi không có dữ liệu từ bộ nhớ EEPROM, ESP8266 sẽ chuyển sang chế độ softAP, để server có thể kết nối tới và gửi dữ liệu bao gồm mã Token, ssid và password WiFi xung quanh. Khi đã nhận được dữ liệu từ server, ESP sẽ lưu lại những dữ liệu đó trong EEPROM và lại quay lại đọc dữ liệu từ EEPROM và thực hiện lại từ đầu.

2. Sơ đồ kết nối của thiết bị



Hình 4. 1: Sơ đồ kết nối của thiết bị

CHƯƠNG 4: VẬN HÀNH HỆ THỐNG

Điện áp	80 – 260VAC, sai số 0.01
Dòng điện đo và hoạt động	0 – 100A, sai số 0.01
Công suất đo và hoạt động	0 ~ 23kW
Tần số đo và hoạt động	45 – 65Hz
Chuẩn giao tiếp	UART
Chuẩn WiFi	2.4GHz
Dải đo công suất tiêu thụ	0 - 9999.99kWh
Kích thước	68 x 100 x 50mm

Bảng 4. 1: Thông số kỹ thuật của thiết bị



Hình 4. 2: Mô hình bên trong của thiết bị

CHƯƠNG 4: VẬN HÀNH HỆ THỐNG

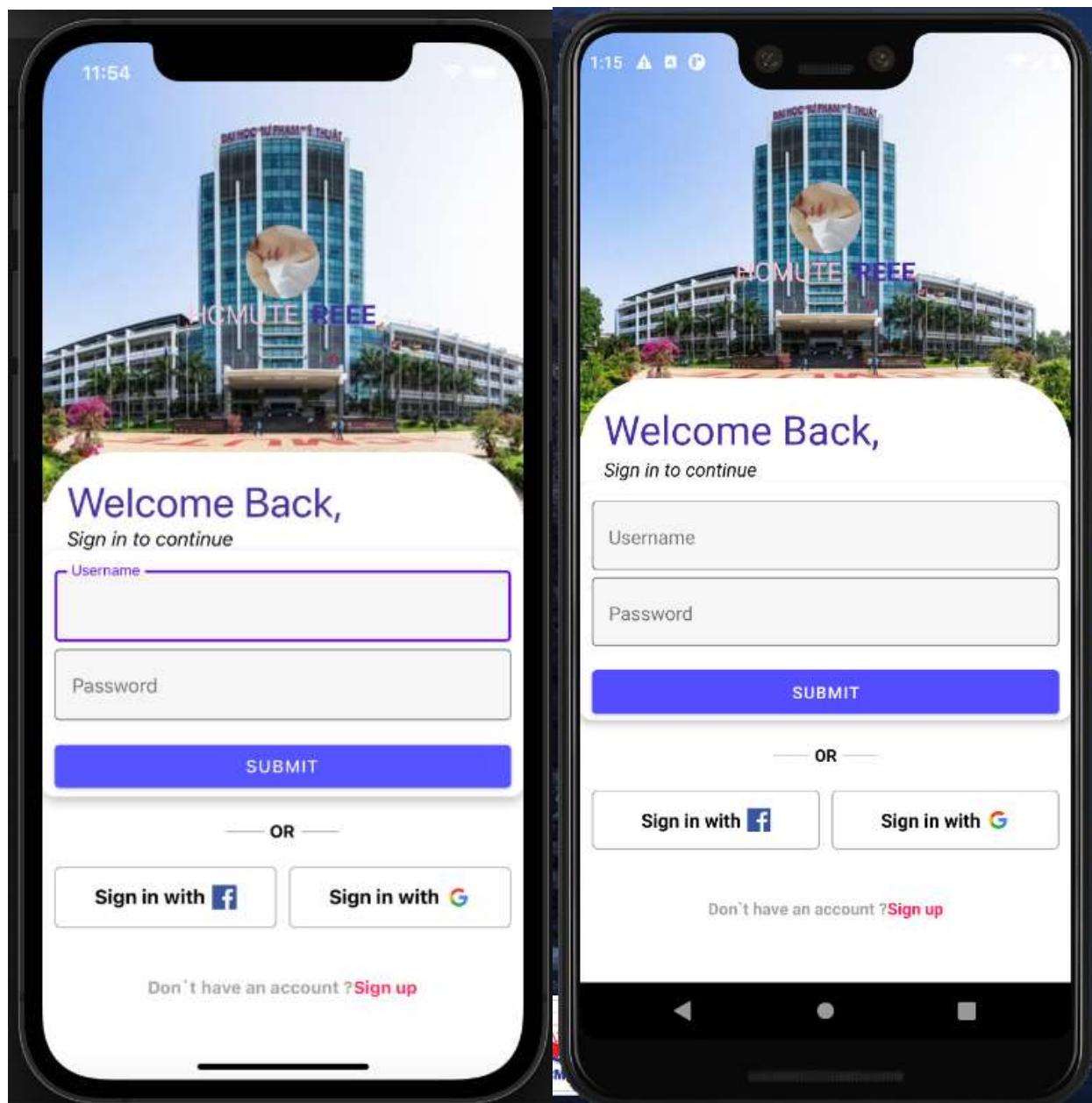


Hình 4. 3: Mô hình khi đóng gói thành sản phẩm

CHƯƠNG 4: VẬN HÀNH HỆ THỐNG

II. PHẦN GIAO DIỆN APP CHO ANDROID VÀ IOS

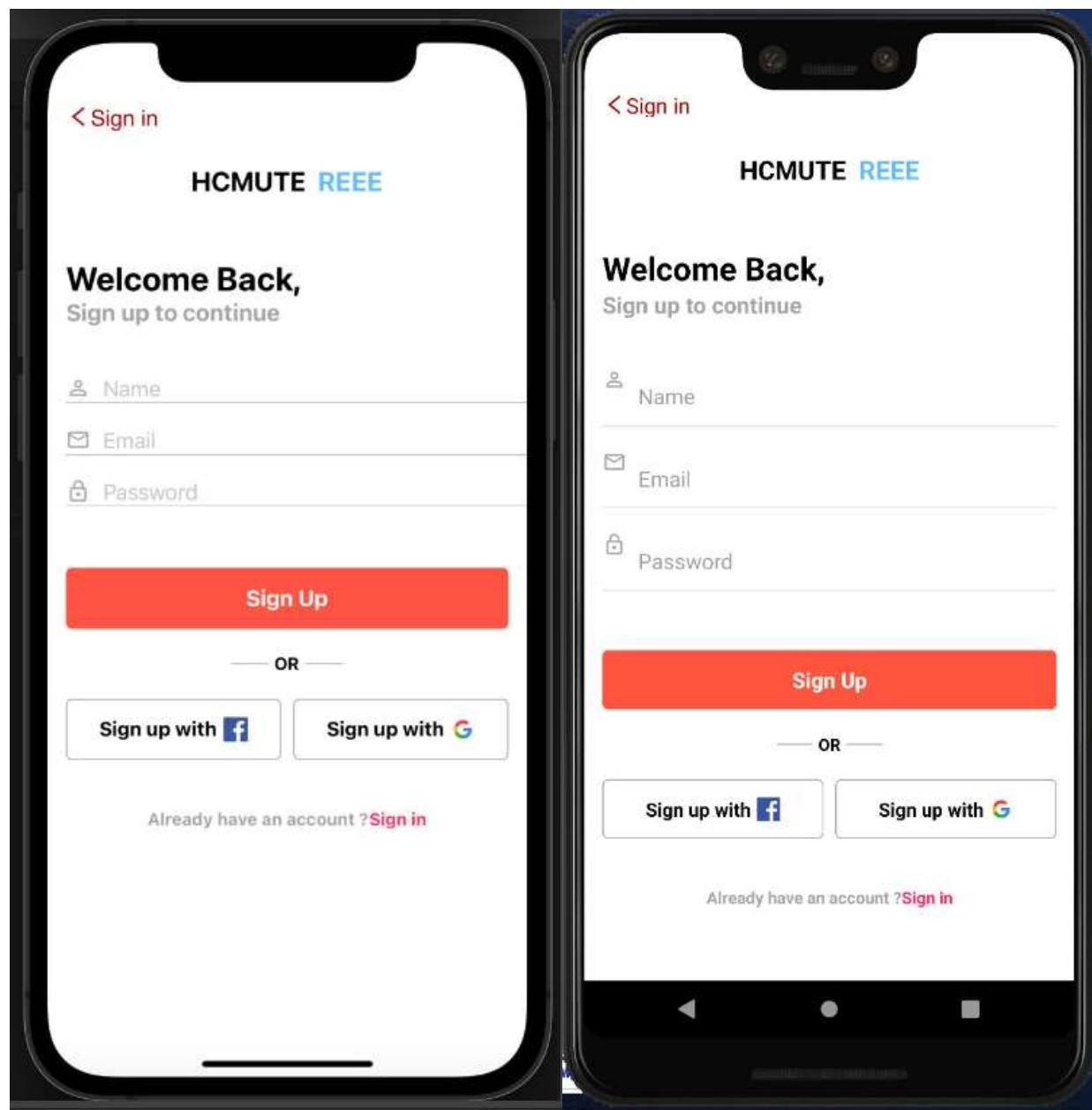
Đăng nhập xác thực qua Google Authentication sử dụng Gmail



Hình 4. 4: Giao diện đăng nhập người dùng

CHƯƠNG 4: VẬN HÀNH HỆ THỐNG

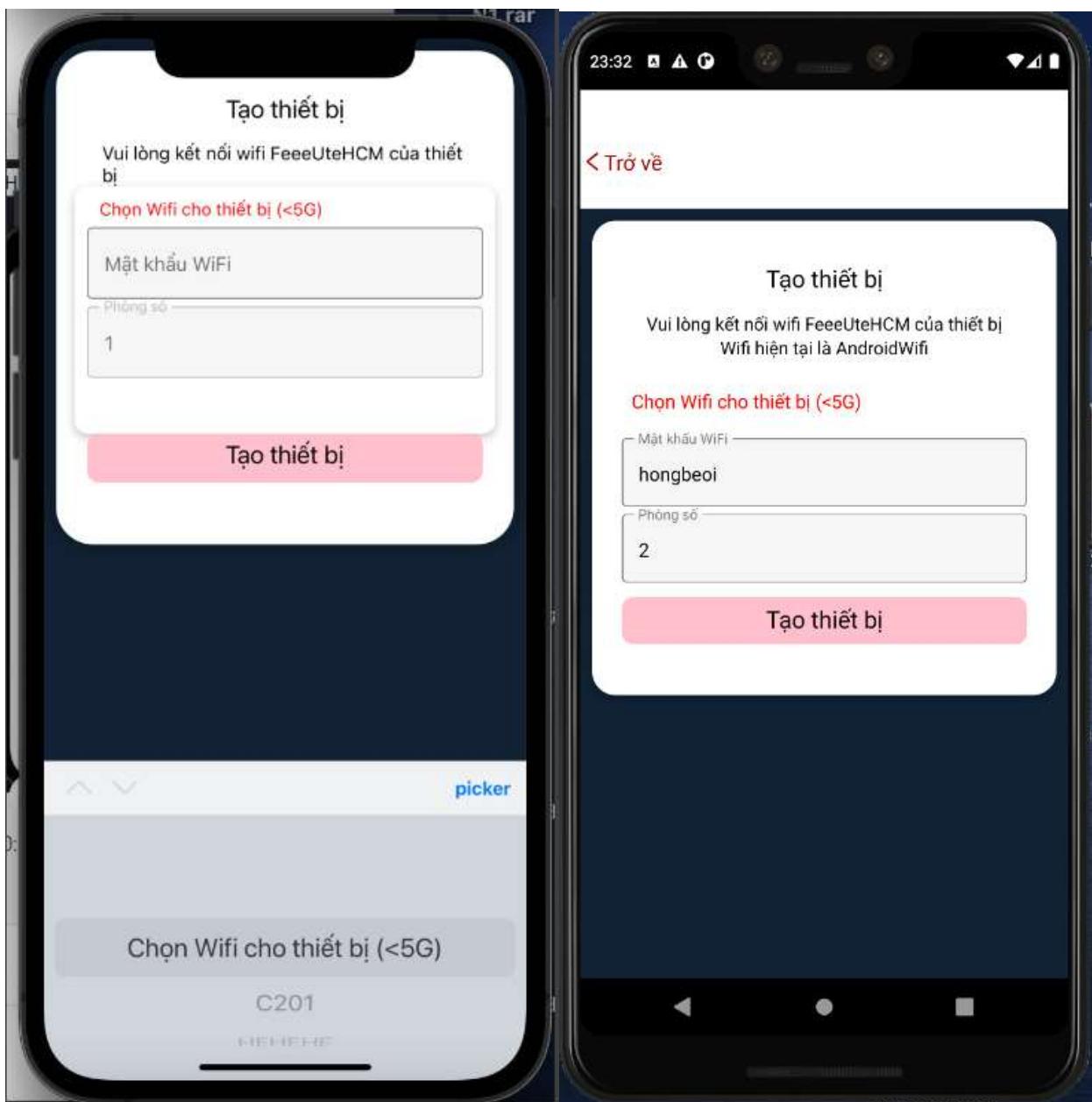
Đăng ký xác thực qua Google Authentication sử dụng Gmail



Hình 4.5: Giao diện đăng ký người dùng

CHƯƠNG 4: VẬN HÀNH HỆ THỐNG

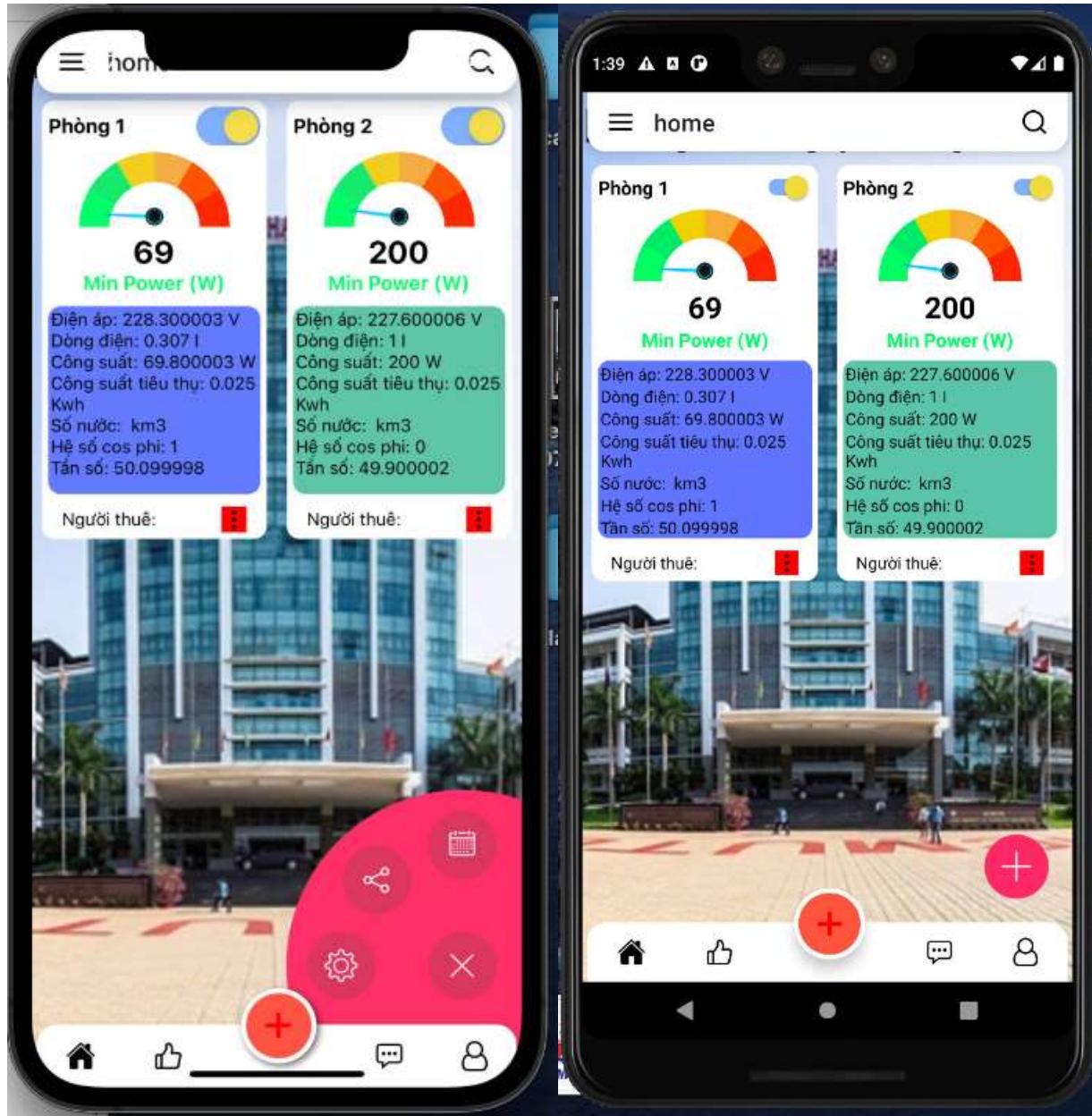
Cấu hình tự động WiFi và Token cho ESP8266 thông qua giao thức WebSocket



Hình 4. 6: Đăng nhập WiFi cho ESP8266

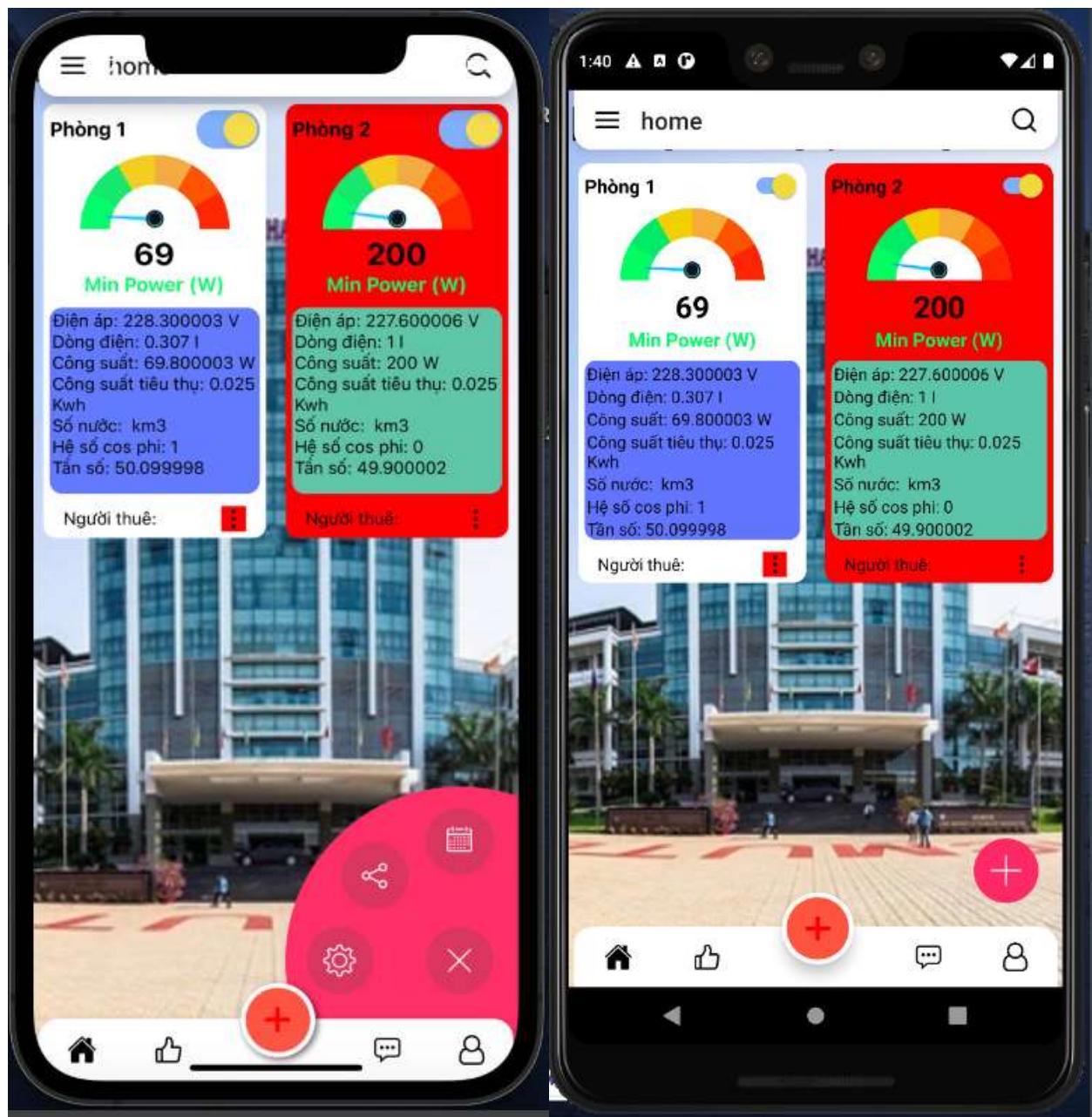
CHƯƠNG 4: VẬN HÀNH HỆ THỐNG

Giao diện giám sát năng lượng điện có cảnh báo công suất và bật tắt tải. Các thông số giám sát bao gồm: điện áp, dòng điện, công suất, công suất tiêu thụ, tần số, cos phi và số nước.



Hình 4. 7: Giao diện các thông số điện năng của phòng

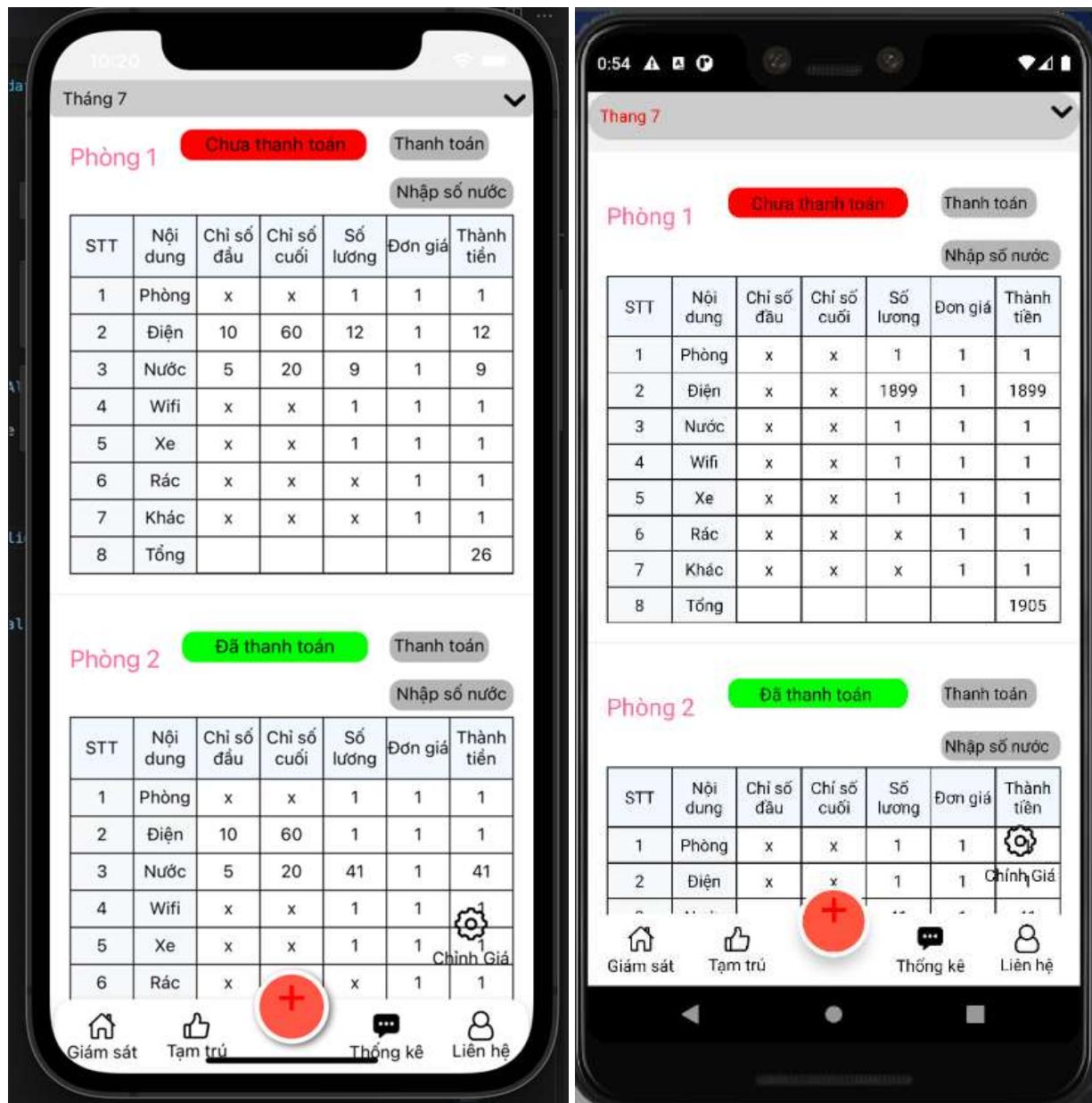
CHƯƠNG 4: VẬN HÀNH HỆ THỐNG



Hình 4. 8: Giao diện cảnh báo khi sử dụng quá công suất

CHƯƠNG 4: VẬN HÀNH HỆ THỐNG

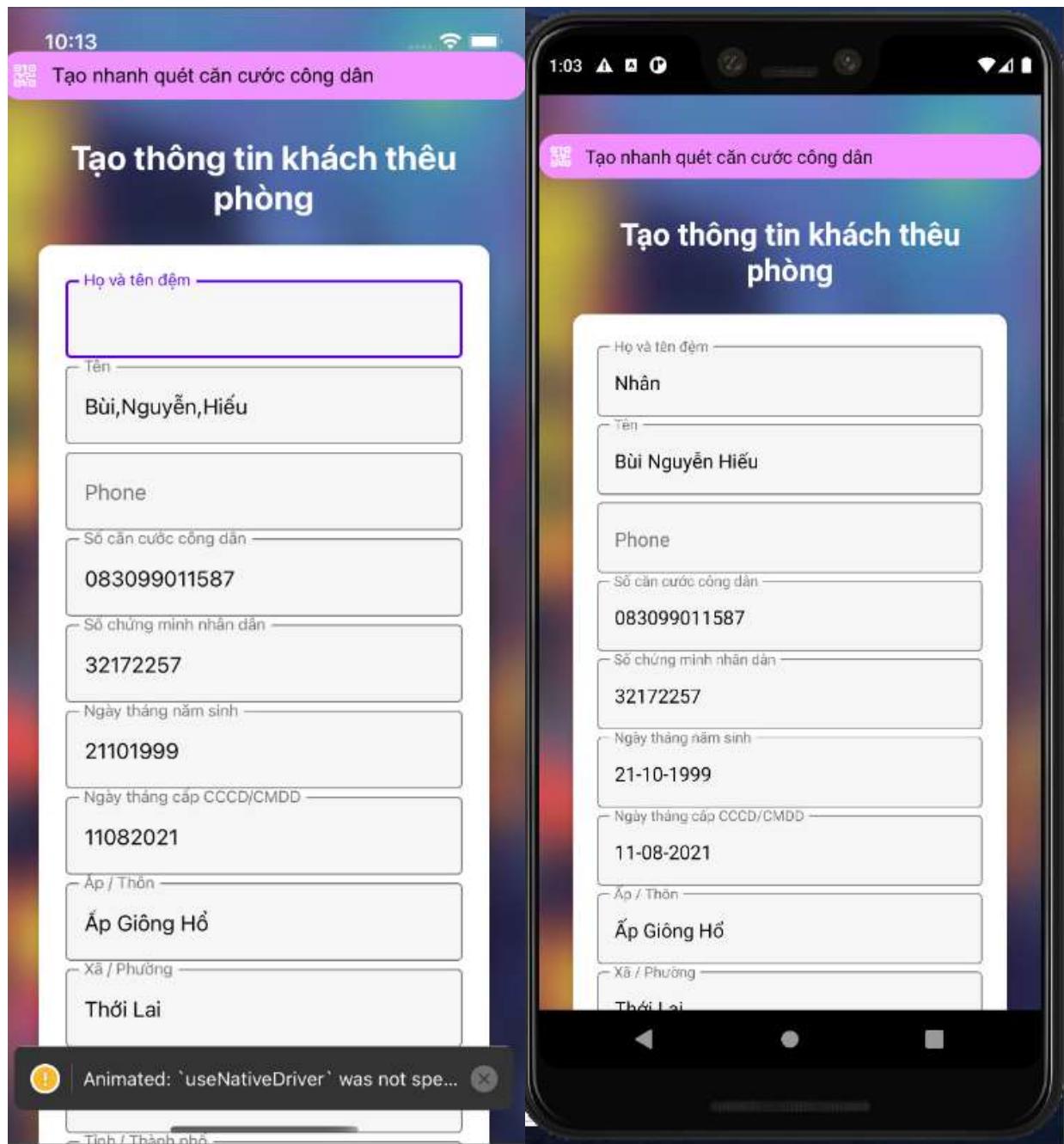
Giao diện thanh toán hóa đơn tiền phòng tiền phòng bao gồm tiền điện, nước, Wifi, rác, xe,... Đơn giá và số lượng.



Hình 4. 9: Giao diện bảng tính hoá đơn tiền phòng

CHƯƠNG 4: VẬN HÀNH HỆ THỐNG

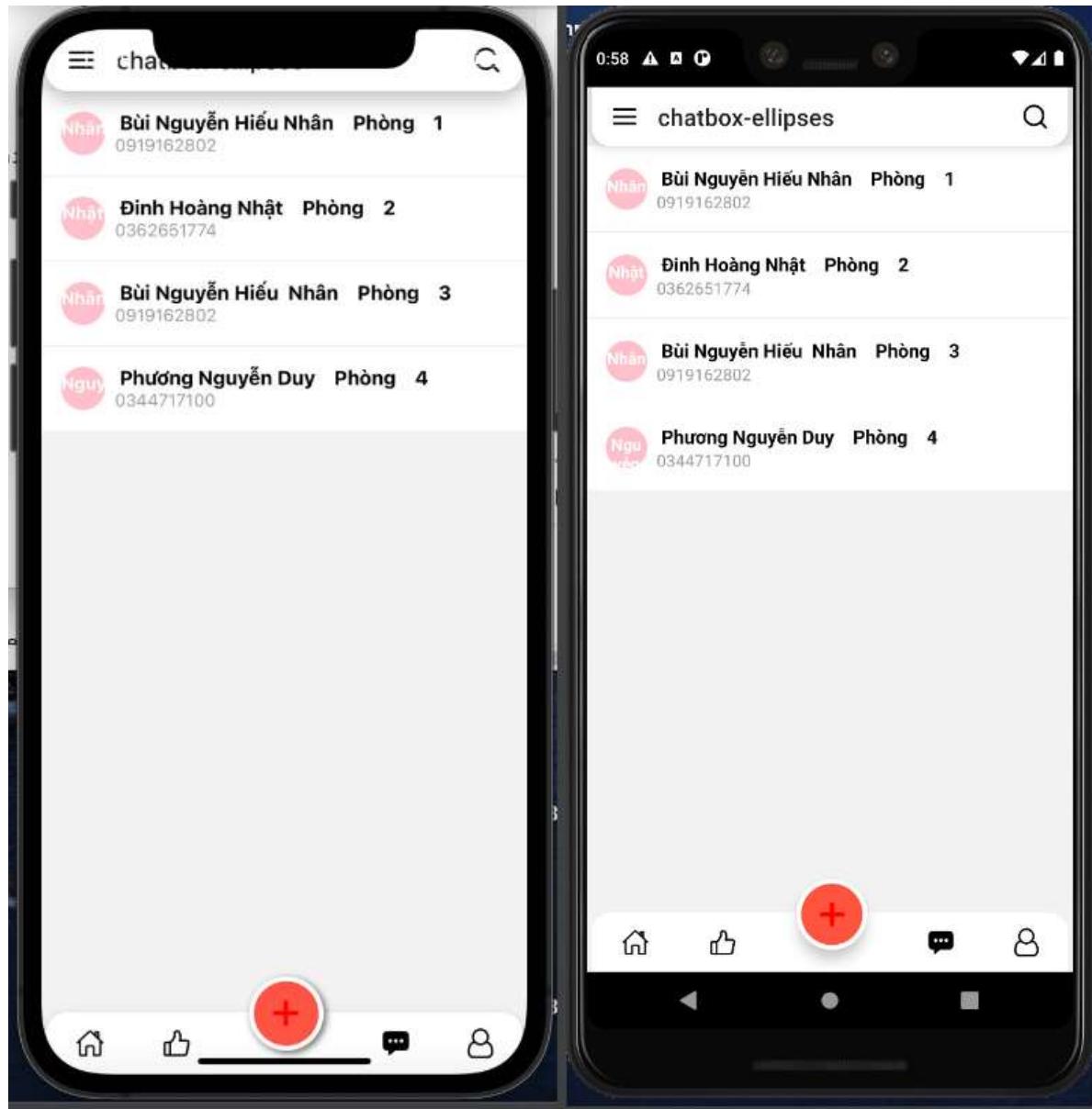
Tạo thông tin khách hàng thuê phòng bằng phương pháp quét mã QR hoặc nhập tay.



Hình 4. 10: Giao diện quét căn cước công dân để đăng ký tạm trú tạm vắng

CHƯƠNG 4: VẬN HÀNH HỆ THỐNG

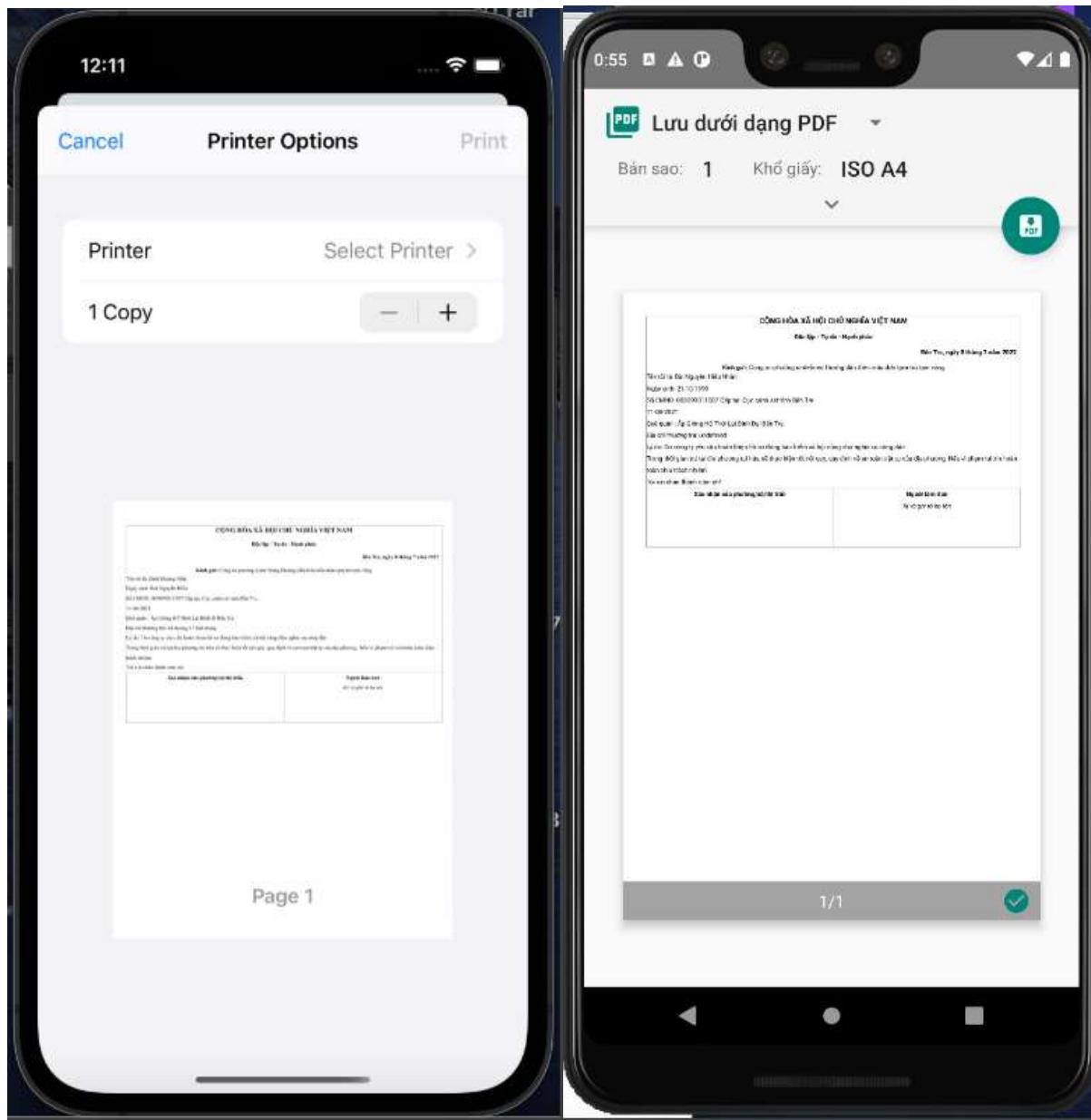
Thông tin liên hệ khách thuê phòng qua số điện thoại, chuyển sang gọi tự động nếu được yêu cầu.



Hình 4. 11: Giao diện thông tin số điện thoại khách thuê

CHƯƠNG 4: VẬN HÀNH HỆ THỐNG

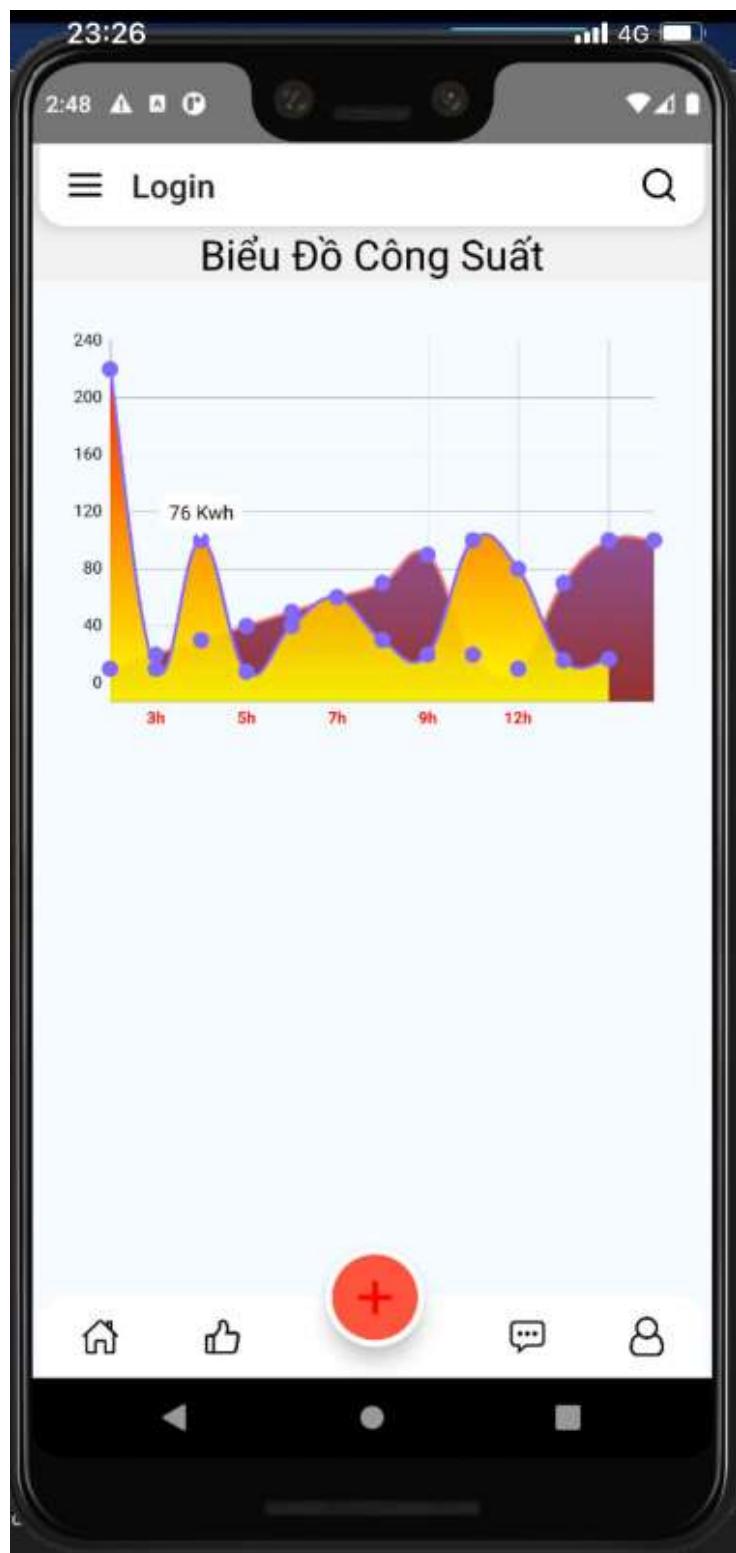
Thực hiện giấy đăng ký tạm trú tạm vắng theo thông tin của khách thuê phòng và chuyển sang PDF.



Hình 4. 12: Giao diện giấy đăng ký tạm trú tạm vắng

CHƯƠNG 4: VẬN HÀNH HỆ THỐNG

Biểu đồ hiện thị giá trị công suất trong ngày của phòng.



Hình 4. 13: Biểu đồ công suất sử dụng trong ngày

CHƯƠNG 4: VẬN HÀNH HỆ THỐNG

III. PHẦN SERVER GOOGLE CLOUD FIREBASE

1. Xác thực thông tin người dùng Authentication

Xác thực thông tin người dùng Authentication bao gồm: thời gian khởi tạo, thời gian đăng nhập, User ID.

The screenshot shows the Firebase console's Authentication section. On the left sidebar, 'Authentication' is selected under 'Project shortcuts'. The main area displays a table of users with the following columns: Identifier, Providers, Created, Signed In, and User UID. There are four entries:

Identifier	Providers	Created	Signed In	User UID
heochoga@gmail.com	✉️	Jun 27, 2022	Jun 27, 2022	XlbbZY5NBghEv1dRMuTdXULyLIT2
phuocthien897@gmail.com	✉️	Jun 26, 2022	Jun 26, 2022	q4D0kmbv7xQeerOKiWBInq9wF22
nhan99@gmail.com	✉️	Jun 25, 2022	Jun 25, 2022	beZ4tujufHSvCWljyqAUTJouiw1
nhan@gmail.com	✉️	Jun 7, 2022	Jul 10, 2022	a32npAcZmaTQPdbEhi0ODu9NH53

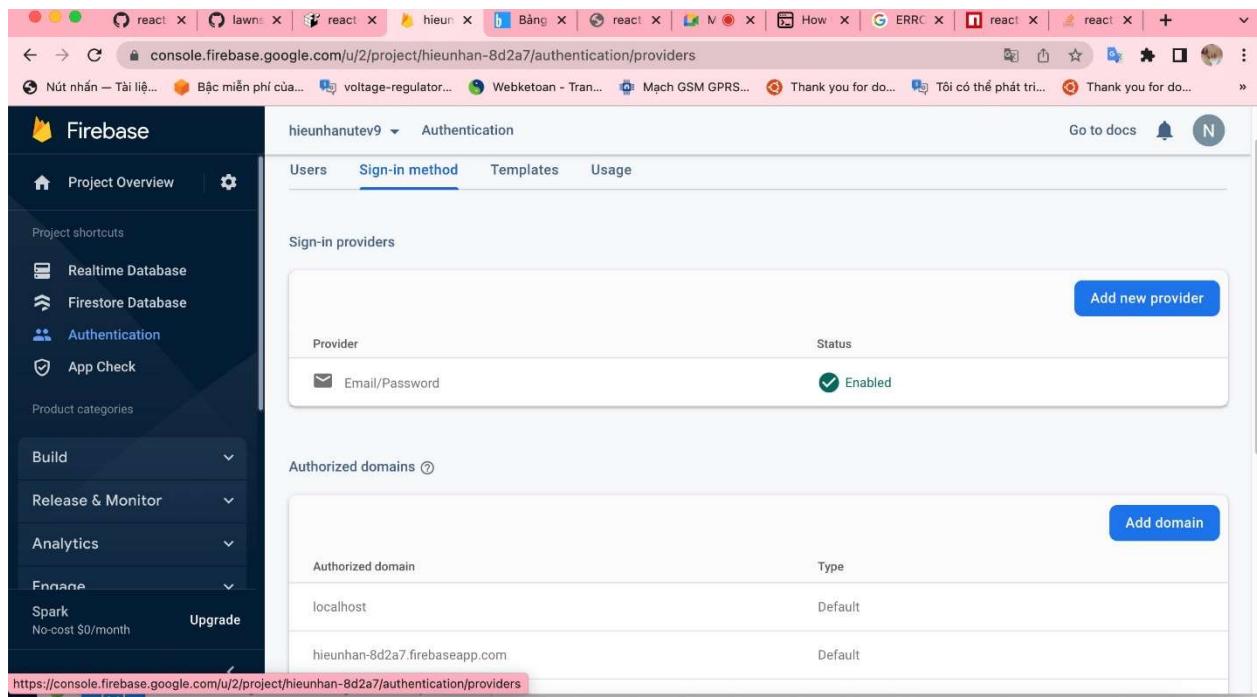
Hình 4. 14: Xác thực thông tin người dùng Authentication

The screenshot shows the Firebase console's Authentication Templates section. On the left sidebar, 'Authentication' is selected under 'Project shortcuts'. The main area displays a table of templates with the following columns: Email, Email address verification, Password reset, Email address change, SMTP settings, SMS, and SMS verification. The 'Email address verification' template is selected, showing its configuration details:

Email	Email address verification
✉️	When a user signs up using an email address and password, you can send them a confirmation email to verify their registered email address. Learn more
✉️	Sender name: not provided, From: noreply@hieunhan-8d2a7.firebaseioapp.com
✉️	Reply to: noreply
✉️	Subject: Verify your email for %APP_NAME%
SMS	Message: Hello %DISPLAY_NAME%, Follow this link to verify your email address.
SMS	Verification code: %VERIFICATION_CODE%

Hình 4. 15: Thay đổi password cài đặt tài khoản

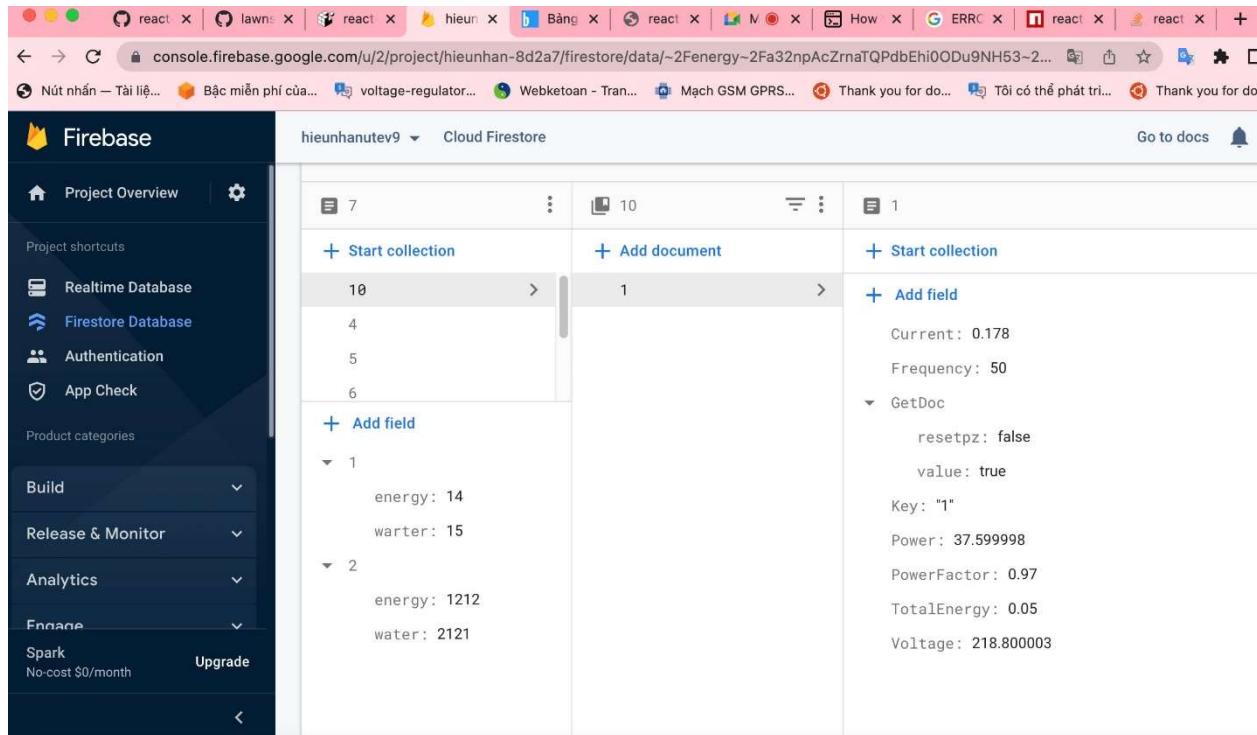
CHƯƠNG 4: VẬN HÀNH HỆ THỐNG



The screenshot shows the Firebase Authentication providers page. On the left sidebar, 'Authentication' is selected under 'Product categories'. The main area displays the 'Sign-in providers' section, which lists 'Email/Password' as an enabled provider. Below this, the 'Authorized domains' section shows two entries: 'localhost' and 'hieunhan-8d2a7.firebaseioapp.com', both marked as 'Default' type.

Hình 4. 16: Lựa chọn các phương thức xác thực

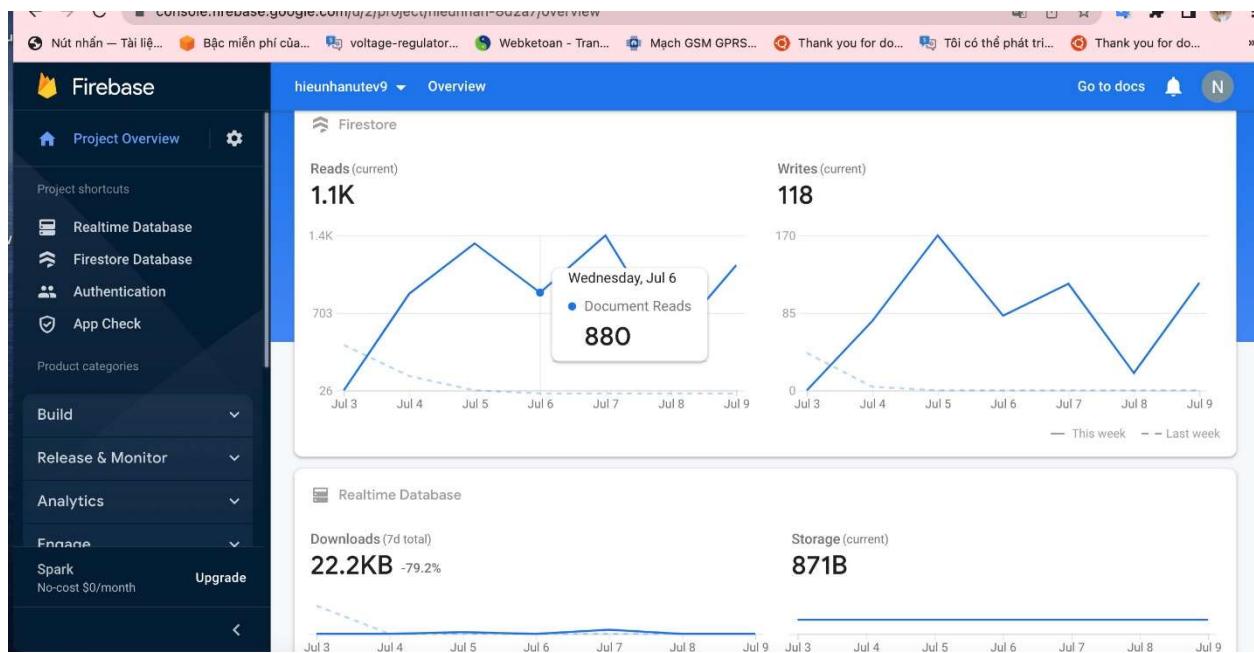
2. Cloud Firestore



The screenshot shows the Cloud Firestore data page. On the left sidebar, 'Cloud Firestore' is selected under 'Product categories'. The main area displays a table with two collections: '10' and '2'. Collection '10' contains documents '4', '5', and '6'. Collection '2' contains documents '1' and '2'. To the right of the table, detailed information for document '1' is shown, including fields like 'Current: 0.178', 'Frequency: 50', 'Power: 37.599998', 'PowerFactor: 0.97', 'TotalEnergy: 0.05', and 'Voltage: 218.800003'. There are also buttons for 'Start collection', 'Add field', and 'GetDoc'.

Hình 4. 17: Giám sát dữ liệu Firestore

CHƯƠNG 4: VẬN HÀNH HỆ THỐNG



Hình 4.18: Quản lý lượt đọc và ghi kết nối tới Firestore

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

I. KẾT LUẬN

- Thiết kế thành công thiết bị đo đếm đo đếm điện năng và đưa lên Server Google Cloud Firestore để việc giám sát diễn ra mọi nơi có kết nối internet.
- Xây dựng được hệ thống sever lưu trữ và truyền nhận dữ liệu hoạt động ổn định trên Google Cloud Firestore.
- Thiết kế thành công App để giám sát năng lượng điện cho phòng trọ với giao diện thân thiện người dùng, giám sát các thông số của điện năng (Realtime), thông tin khách hàng kết hợp tính toán các chi phí thuê phòng với việc quản lý một cách dễ dàng hơn, tiết kiệm thời gian, độ chính xác cao.
- Kết quả thu được các thông số như: dòng điện, điện áp, công suất tiêu thụ,... Từ mạch đo đếm điện năng không chênh lệch quá nhiều so với đồng hồ đo đếm điện năng.
- Việc sử dụng App để quản lý năng lượng điện giúp cho người chủ giảm bớt chi phí nhân công, không cần phải đi ghi điện ở từng hộ và tránh những sai sót không đáng có.
- Việc thiết kế thêm phần đăng ký tạm trú tạm vắng online trên App giúp cho người đi thuê đăng ký tạm trú tạm vắng trở nên dễ dàng hơn, ít tốn thời gian hơn. Không cần phải đến tận nơi mà có thể đăng ký trực tiếp trên App.

II. HƯỚNG PHÁT TRIỂN

- Tích hợp thêm các chức năng thanh toán online qua Vnpay, Momo,...
- Tích hợp thêm của hàng mua bán nước, gas, gạo,... cho khách thuê có thể đặt online.
- Tích hợp thêm chức năng tìm người ở cùng để giám bớt chi phí thuê phòng.
- Tích hợp thêm chức năng cho thuê phòng, đăng ký cho thuê phòng trọ online.

TÀI LIỆU THAM KHẢO

- [1] <https://danso.org/viet-nam/>
- [2] <https://www.24h.com.vn/kinh-doanh/thue-phong-tro-vua-y-voi-gia-hop-ly-o-dau-cho-tan-sinh-vien-tp-hcm-c161a1075139.html>
- [3] <https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers>
- [4] <https://www.arduino.cc/en/software>
- [5] <https://apps.apple.com/vn/app/xcode/id497799835?l=vi&mt=12>
- [6] <https://developer.android.com/studio>
- [7] <https://console.firebaseio.google.com>
- [8] <https://reactnavigation.org/docs/getting-started/>
- [9] <https://firebase.google.com/docs/auth/web/password-auth#web-version-9>
- [10] <https://www.npmjs.com/package/react-native-picker-select>
- [11] <https://redux.js.org/tutorials/fundamentals/part-3-state-actions-reducers>
- [12] <https://rnfirebase.io/>
- [13] <https://stackoverflow.com/questions/63715979/cannot-find-module-react-native-reanimated-plugin-from-projectfolder>
- [14] Thư viện Arduino
 - <https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WiFi>
 - <https://github.com/PaulStoffregen/EEPROM>
 - <https://github.com/Links2004/arduinoWebSockets>
 - <https://github.com/mobizt/Firebase-ESP-Client>
 - <https://github.com/mandulaj/PZEM-004T-v30>
 - <https://github.com/klarsys/esp8266-OLED>

PHỤ LỤC

Code phần nhúng

```
#include <ESP8266WiFi.h>
#include <EEPROM.h>
#include <WebSocketsServer.h>
#include <Firebase_ESP_Client.h>
#include <PZEM004Tv30.h>
#include <Wire.h>
#include "OLED.h"
#include <time.h>

int relay = 12;
PZEM004Tv30 pzem(&Serial);
OLED display(4, 5);

int timezone = 7*3600;
int dst=0;

WebSocketsServer webSocket = WebSocketsServer(81);
const IPAddress apIP(192, 168, 4, 1);
const char* apSSID = "FeeeUteHCM";

int indexhang;
int indexbang;
int indexva;
int indexacong;
String inputString = "";
String username = "";
String password = "";
String Token = "";
String Phong = "";
String Phong1 = "";
String Phong3 = "";
String Phong4 = "";
String Token1 = "";
String Token3 = "";
String Token4 = "";
bool stringComplete = false;
boolean receivedata;

// button
static const int buttonPin = 13;
```

```
int buttonStatePrevious = LOW;
unsigned long minButtonLongPressDuration = 5000;
unsigned long buttonLongPressMillis;
bool buttonStateLongPress = false;
const int intervalButton = 50;
unsigned long previousButtonMillis;
unsigned long buttonPressDuration;
unsigned long currentMillis;

// Firesotre
#include "addons/TokenHelper.h"

#define API_KEY "AIzaSyDBmkNyA8910b5P-546IkjW0Wkj_-Ntu9w"

#define FIREBASE_PROJECT_ID "hieunhan-8d2a7"

#define USER_EMAIL "nhan@gmail.com"
#define USER_PASSWORD "123456"

FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;
unsigned long dataMillis = 0;
int count = 0;
bool taskcomplete = false;
boolean sendfirestore;
boolean connectfiretore;
boolean sendok;
bool value = false;
bool resetpz = false;

void setup() {
  pinMode(relay, OUTPUT);
  digitalWrite(relay, HIGH);
  pinMode(buttonPin, INPUT);
  Serial.begin(9600);
  EEPROM.begin(512);
  display.begin();
  display.clear();
  delay(100);
  if (restoreConfig()) {
    checkConnection();
  }
  webSocket.begin();
  webSocket.onEvent(webSocketEvent);
```

```

// firestore

}

void loop() {
    readButtonState();
    webSocket.loop();
    if(sendfirestore){
        if(connectfiretore){
            configTime(timezone, dst, "pool.ntp.org","time.nist.gov");
            Serial.println("Witting Connect Timezone");
            while(!time(nullptr)){
                Serial.print("*");
                delay(500);
            }
            Serial.println("OK!");
            Serial.printf("Firebase Client v%os\n\n", FIREBASE_CLIENT_VERSION);
            config.api_key = API_KEY;
            auth.user.email = USER_EMAIL;
            auth.user.password = USER_PASSWORD;
            config.token_status_callback = tokenStatusCallback; //see addons/TokenHelper.h
            Firebase.begin(&config, &auth);
            display.clear();
            Firebase.reconnectWiFi(true);
            connectfiretore = false;
        }
        firestorepzem();
    }
}

// read EEPROM
boolean restoreConfig() {
    String ssid = "";
    String pass = "";
    if (EEPROM.read(0) != 0) {
        for (int i = 0; i < 31; ++i) {
            ssid += char(EEPROM.read(i));
        }
        Serial.println("Reading username to EEPROM: ");
        Serial.println(ssid);
        for (int i = 32; i < 95; ++i) {
            pass += char(EEPROM.read(i));
        }
        Serial.println("Reading Password to EEPROM: ");
        Serial.println(pass);
    }
}

```

```

for (int i = 96; i < 499; ++i) {
    Token1 += char(EEPROM.read(i));
}
Serial.println("Reading Token to EEPROM: ");
Serial.println(Token1);
delay(100);
String Token2 = "%" + Token1 + "$" ;
int indextien;
int indexphantram;
indexphantram = Token2.indexOf("%");
indextien = Token2.indexOf("$");
Token3 = Token2.substring(indexphantram+1, indextien);
Token4 = Token2.substring(indextien+1);
for (int i = 500; i < 512; ++i) {
    Phong1 += char(EEPROM.read(i));
}
Serial.println("Reading Token to EEPROM: ");
Serial.println(Phong1);
delay(100);
String Phong2 = "^" + Phong1 + "*" ;
int indexmu;
int indexsao;
indexmu = Phong2.indexOf("^");
indexsao = Phong2.indexOf("*");
Phong3 = Phong2.substring(indexmu+1, indexsao);
Phong4 = Phong2.substring(indexsao+1);
display.clear();
WiFi.begin(ssid.c_str(), pass.c_str());
return true;
}
else {
    Serial.println("Config not found.");
    setupMode();
    return false;
}
}
// check connect WiFi
boolean checkConnection() {
    int count = 0;
    Serial.print("Waiting for Wi-Fi connection");
    while ( count < 30 ) {
        if (WiFi.status() == WL_CONNECTED) {
            Serial.println();
            Serial.println("Connected!");
            Serial.println(WiFi.localIP());

```

```

delay(1000);
sendfirestore = true;
connectfiretore = true;
return (true);
}
delay(500);
Serial.print(".");
count++;
}
Serial.println("Timed out.");
setupMode();
return false;
}
// che do Access point
void setupMode(){
  WiFi.disconnect();
delay(100);
WiFi.mode(WIFI_AP);
WiFi.softAP(apSSID);
Serial.print("Starting Access Point at \\"");
Serial.print(apSSID);
Serial.println("\\"");
display.clear();
display.print("WiFi: OFF", 0, 3);
receivedata = true ;
}
// nhan giu lieu tu Client
void webSocketEvent(uint8_t num, WStype_t type,
                     uint8_t * payload,
                     size_t length)
{
String payloadString = "";
payloadString = (const char *)payload;
sendok = true;
if(sendok){
  webSocket.broadcastTXT("OK");
  sendok = false;
}
inputString = payloadString;
if(receivedata){
  for(int n = 10; n < inputString.length() ; n++ ){
    if(n >= ( inputString.length()-1 )){stringComplete = true;}
  }
  if (stringComplete) {
    indexthang = inputString.indexOf("/");
}

```

```
indexbang = inputString.indexOf("=");
indexva = inputString.indexOf("&");
indexacong = inputString.indexOf("@");
username = inputString.substring(indexthang+1,indexbang);
password = inputString.substring(indexbang+1,indexva);
Token = inputString.substring(indexva+1, indexacong);
Phong = inputString.substring(indexacong+1);
Serial.println(username);
Serial.println(password);
Serial.println(Token);
Serial.println(Phong);
inputString = "";
display.clear();
display.print("Waiting Connect", 3, 0);
display.print("WiFi..... ", 4, 3);
WiFi.mode(WIFI_STA);
for (int i = 0; i < 512; ++i) {
    EEPROM.write(i, 0);
    delay(10);
}
Serial.println("Writing username to EEPROM...");
for (int i = 0; i < username.length(); ++i) {
    EEPROM.write(i, username[i]);
}
Serial.println(username);
Serial.println("Writing Password to EEPROM...");
for (int i = 0; i < password.length(); ++i) {
    EEPROM.write(32 + i, password[i]);
}
Serial.println(password);
Serial.println("Writing ToKen to EEPROM...");
for (int i = 0; i < Token.length(); ++i) {
    EEPROM.write(96 + i, Token[i]);
}
Serial.println(Token);
Serial.println("Writing ToKen to EEPROM...");
for (int i = 0; i < Phong.length(); ++i) {
    EEPROM.write(500 + i, Phong[i]);
}
Serial.println(Phong);
EEPROM.commit();
stringComplete = false;
receivedata = false ;
if (restoreConfig()) {
    checkConnection();
```

```

        }
    }
}
}

// Button reset EEPROM 5s
void readButtonState() {
    currentMillis = millis();
    if(currentMillis - previousButtonMillis > intervalButton) {
        int buttonState = digitalRead(buttonPin);
        if (buttonState == HIGH && buttonStatePrevious == LOW &&
!buttonStateLongPress) {
            buttonLongPressMillis = currentMillis;
            buttonStatePrevious = HIGH;
        }
        buttonPressDuration = currentMillis - buttonLongPressMillis;
        if (buttonState == HIGH && !buttonStateLongPress && buttonPressDuration >=
minButtonLongPressDuration) {
            buttonStateLongPress = true;
            for (int i = 0; i < 512; ++i) {
                EEPROM.write(i, 0);
            }
            EEPROM.commit();
            Serial.println("Reset EEPROM");
            display.clear();
            WiFi.disconnect();
            Serial.println("Disconnect WiFi");
            sendfirestore = false;
            delay(500);
            setupMode();
        }
        if (buttonState == LOW && buttonStatePrevious == HIGH) {
            buttonStatePrevious = LOW;
            buttonStateLongPress = false;
        }
        previousButtonMillis = currentMillis;
    }
}

// doc giu lieu tu pzem gui len firestore
void firestoredpzem(){
    time_t now=time(nullptr);
    struct tm* p_tm=localtime(&now);
    Serial.print(p_tm ->tm_mday);
    Serial.print("/");
    Serial.print(p_tm ->tm_mon + 1);
    Serial.print("/");
}

```

```

Serial.print(p_tm ->tm_year + 1900);
Serial.print(" ");
Serial.print(p_tm ->tm_hour);
Serial.print(":");
Serial.print(p_tm ->tm_min);
Serial.print(":");
Serial.print(p_tm ->tm_sec);
Serial.print("\n");
display.print("WiFi: ON", 0, 3);
// ngay
char day[3];
String strday;
strday += p_tm ->tm_mday;
strday.toCharArray(day, 3);
display.print(day, 1, 3);
// thang
char mon[2];
String strmon;
strmon += p_tm ->tm_mon + 1;
strmon.toCharArray(mon, 2);
display.print("/", 1, 5);
display.print(mon, 1, 6);
// nam
char year[5];
String stryear;
stryear += p_tm ->tm_year + 1900;
stryear.toCharArray(year, 5);
display.print("/", 1, 8);
display.print(year, 1, 9);
// gio
char hour[3];
String strhour;
strhour += p_tm ->tm_hour;
strhour.toCharArray(hour, 3);
display.print(hour, 2, 5);
// phut
char min[3];
String strmin;
strmin += p_tm ->tm_min;
strmin.toCharArray(min, 3);
display.print(":", 2, 7);
display.print(min, 2, 8);
// // giay
// char sec[3];
// String strsec;

```

```

// strsec += p_tm ->tm_sec;
// strsec.toCharArray(sec, 3);
// display.print(":", 2, 9);
// display.print(sec, 2, 10);
// dien ap
float voltage = pzem.voltage();
char charvol[7];
String strvol;
strvol += voltage;
strvol.toCharArray(charvol, 7);
display.print("V:", 3, 1);
display.print(charvol, 3, 4);
display.print("V", 3, 10);
// dong dien
float current = pzem.current();
char charcur[7];
String strcur;
strcur += current;
strcur.toCharArray(charcur, 7);
display.print("I:", 4, 1);
display.print(charcur, 4, 4);
display.print("A", 4, 9);
// cong suat
float power = pzem.power();
char charpow[6];
String strpow;
strpow += power;
strpow.toCharArray(charpow, 6);
display.print("P:", 5, 1);
display.print(charpow, 5, 5);
display.print("W", 5, 9);
// cong suat tieu thu
float energy = pzem.energy();
char charene[7];
String strene;
strene += energy;
strene.toCharArray(charene, 7);
display.print("E:", 6, 1);
display.print(charene, 6, 5);
display.print("kWh", 6, 10);
// tan so
float frequency = pzem.frequency();
char charfre[5];
String strfre;
strfre += frequency;

```

```

strfre.toCharArray(charfre, 5);
display.print("F:", 7, 1);
display.print(charfre, 7, 4);
display.print("Hz", 7, 8);
display.on();
delay(500);
String firestoretoken = "energy/" + Token3 + "/" + stryear + "/" + strmon + "/" + strday
+ "/" + Phong3 ;
// firestore
if (Firebase.ready() && (millis() - dataMillis > 15000 || dataMillis == 0))
{
    dataMillis = millis();
    if (!taskcomplete)
    {
        taskcomplete = true;
        String content;
        FirebaseJson js;

        String documentPath = firestoretoken;

        float current = pzem.current();
        float power = pzem.power();
        float pf = pzem.pf();
        float frequency = pzem.frequency();
        float energy = pzem.energy();
        float voltage = pzem.voltage();
        if( !isnan(current) && !isnan(power) && !isnan(pf) && !isnan(frequency) &&
        !isnan(energy) && !isnan(voltage)){
            js.set("fields/CurrentdoubleValue", current);
            js.set("fields/PowerdoubleValue", power);
            js.set("fields/PowerFactordoubleValue", pf);
            js.set("fields/FrequencydoubleValue", frequency);
            js.set("fields/TotalEnergydoubleValue", energy);
            js.set("fields/VoltagedoubleValue", voltage);
            js.set("fields/Key/stringValue", Phong3);
            js.set("fields/GetDoc/mapValue/fields/value/booleanValue", true);
            js.set("fields/GetDoc/mapValue/fields/resetpz/booleanValue", false);
            js.toString(content);
            delay(200);
            Serial.print("Create a document... ");
            if (Firebase.Firestore.createDocument(&fbdo, FIREBASE_PROJECT_ID, /*
databaseId can be (default) or empty */ , documentPath.c_str(), content.c_str()))
                Serial.printf("ok\n%s\n\n", fbdo.payload().c_str());
            else

```

```

        Serial.println(fbdo.errorReason());
    }
else{
    Serial.println("Not create firebase");
    js.set("fields/GetDoc/mapValue/fields/value/booleanValue", true);
    js.set("fields/GetDoc/mapValue/fields/resetpz/booleanValue", false);
    js.set("fields/Key/stringValue", Phong3);
    js.toString(content);
    delay(200);
    Serial.print("Create a document... ");

    if (Firebase.Firestore.createDocument(&fbdo, FIREBASE_PROJECT_ID, """
/* databaseId can be (default) or empty */, documentPath.c_str(), content.c_str()))
        Serial.printf("ok\n%s\n\n", fbdo.payload().c_str());
    else
        Serial.println(fbdo.errorReason());
}
}

String content;
FirebaseJson js;

String documentPath = firestoretoken;

float current = pzem.current();
float power = pzem.power();
float pf = pzem(pf);
float frequency = pzem.frequency();
float energy = pzem.energy();
float voltage = pzem.voltage();

if( !isnan(current) && !isnan(power) && !isnan(pf) && !isnan(frequency) &&
!isnan(energy) && !isnan(voltage)){
    js.set("fields/CurrentdoubleValue", current);
    js.set("fields/PowerdoubleValue", power);
    js.set("fields/PowerFactordoubleValue", pf);
    js.set("fields/FrequencydoubleValue", frequency);
    js.set("fields/TotalEnergydoubleValue", energy);
    js.set("fields/VoltagedoubleValue", voltage);
    js.toString(content);
    delay(200);
    Serial.print("Update a document... ");
    if (Firebase.Firestore.patchDocument(&fbdo, FIREBASE_PROJECT_ID, "" /*
databaseId can be (default) or empty */, documentPath.c_str(), content.c_str(),
"Current,Power,PowerFactor,Frequency,TotalEnergy,Voltage" /* updateMask */))
        Serial.printf("ok\n%s\n\n", fbdo.payload().c_str());
}

```

```

        else
            Serial.println(fbdo.errorReason());
    } else {
        Serial.println("Not update firebase");
    }
}
Serial.print("Get a document... ");
String documentPath = firestoretoken;
String mask = "GetDoc";
if (Firebase.Firestore.getDocument(&fbdo, FIREBASE_PROJECT_ID, "", documentPath.c_str(), mask.c_str())) {
    Serial.printf("ok\n%s\n\n", fbdo.payload().c_str());
    // Create a FirebaseJson object and set content with received payload
    FirebaseJson payload;
    payload.setJsonData(fbdo.payload().c_str());
    FirebaseJsonData jsonData;
    payload.get(jsonData, "fields/GetDoc/mapValue/fields/value/booleanValue", true);
    Serial.println(jsonData.boolValue);
    if(jsonData.boolValue == true){
        digitalWrite(relay, HIGH);
    }
    if(jsonData.boolValue == false){
        digitalWrite(relay, LOW);
    }
    payload.get(jsonData, "fields/GetDoc/mapValue/fields/resetpz/booleanValue", true);
    Serial.println(jsonData.boolValue);
    if(jsonData.boolValue == true){
        delay(1000);
        pzem.resetEnergy();
        delay(1000);
    }
}
}
}

```

Code phần App

- ❖ Login

```

import React, { useState } from 'react'
import {
    Image, StyleSheet,
    View, ImageBackground,
    Dimensions, SafeAreaView,
    ScrollView, TouchableOpacity,
    StatusBar, Alert, Keyboard, KeyboardAvoidingView
}
from 'react-native'

```

```
import { Button, Surface, Title, Text, TextInput } from 'react-native-paper'
import { useDispatch } from 'react-redux';
import { Login } from '../store/actions';
//import Colors from '../constants/Colors';
//import { loginBg } from '../constants/raw';
import COLORS from '../constants/Colors';
import STYLES from '../styles/index';

import { authentication } from '../firebase/Fire';
import { signInWithEmailAndPassword } from "firebase/auth";

const LoginScreen = ({navigation}) => {
  let heightscreen = Dimensions.get('window').height/2.2;
  let widthscreen = Dimensions.get('window').width;
  const [email, setUsername] = useState("");
  const [password, setPassword] = useState("");
  console.log('aaa')
  const dispatch = useDispatch();
  const submit = () => {
    signInWithEmailAndPassword(authentication, email, password)
    .then(()=>{
      const authen = authentication.currentUser.uid.toString();
      Alert.alert(
        'Alert Title',
        'Đăng nhập thành công: ' + email,
        [
          {text: 'Cancel', onPress: () => console.log('Cancel Pressed'), style:'cancel'},
          {text: 'OK', onPress: () => dispatch(Login(email, password, authen))}]
      ),
      {cancelable: false}
    );
    //console.log(authentication.currentUser.uid)
  })
  .catch(function(error) {
    Alert.alert(
      'Alert Title',
      'Đăng nhập thất bại vui long kiểm tra lại !',
      [
        {text: 'Cancel', onPress: () => console.log('Cancel Pressed')},
        {text: 'OK', onPress: () => console.log('OK Pressed') }
      ],
      {cancelable: false}
    );
  });
}
```

```

return (
  <ScrollView
    style = {{flex: 1, backgroundColor: '#ffffff'}}
    showsVerticalScrollIndicator = {false}>
  <StatusBar translucent backgroundColor={COLORS.transparent} />
  <KeyboardAvoidingView
    behavior={Platform.OS === "ios" ? "padding" : "height"}
    style={styles.container}
    enabled={false}>
  >
  <ImageBackground
    source={require('../image/spkt.jpeg')}
    style={{height: heightscreen, width: widthscreen}}>
    <View style={styles.brandView}>
      <Image
        source={require('../image/hieunhan.jpeg')}
        style={{height:60, width:60, borderRadius: 30}}>
      </Image>
      <Text style={{fontSize: 22, color: 'pink'}}>
        HCMUTE {' '}
        <Text style={{fontWeight: 'bold', fontSize: 22, color: '#4632A1', marginTop: 17}}>
          REEE
          </Text>
        </Text>
      </View>
    </ImageBackground>
    <View style={styles.bottomView}>
      <View style={{marginTop: 20, marginLeft: 20}}>
        <Text style={{color: '#4632A1', fontSize: 34}}>Welcome Back,</Text>
        <Text style={{fontStyle: 'italic', fontSize: 16,}}>Sign in to continue</Text>
      </View >
      <Surface style={styles.box}>
        <View>
          <TextInput
            label="Username"
            mode="outlined"
            value={email}
            onChangeText={(text) => setUsername(text)}
            selectionColor='pink'
            autoFocus={false}
            enablesReturnKeyAutomatically = {true}
          />
          <TextInput

```

```
        label="Password"
        mode="outlined"
        secureTextEntry
        value={password}
        //error={true}
        onChangeText={(text) => setPassword(text)}
    />
</View>
<Button
    mode="contained"
    color={COLORS.blue}
    style={{ marginTop: 20 }}
    onPress={submit}>
    Submit
</Button>
</Surface>
<View
    style={{
        marginVertical: 20,
        flexDirection: 'row',
        justifyContent: 'center',
        alignItems: 'center',
    }}>
<View style={STYLES.line}></View>
    <Text style={{marginHorizontal: 5, fontWeight: 'bold'}}>OR</Text>
    <View style={STYLES.line}></View>
</View>
<View
    style={{
        flexDirection: 'row',
        justifyContent: 'space-between',
        marginLeft: 10,
        marginRight: 10,
    }}>
<View style={STYLES.btnSecondary}>
    <Text style={{fontWeight: 'bold', fontSize: 16}}>
        Sign in with
    </Text>
    <Image
        style={STYLES.btnImage}
        source={require('../image/facebook.jpeg')}
    />
</View>
<View style={{width: 10}}></View>
<View style={STYLES.btnSecondary}>
```

```
<Text style={{fontWeight: 'bold', fontSize: 16}}>
  Sign in with
</Text>
<Image
  style={STYLES.btnExit}
  source={require('../image/google.jpeg')}
/>
</View>
</View>
<View
  style={{{
    flexDirection: 'row',
    alignItems: 'flex-end',
    justifyContent: 'center',
    marginTop: 40,
    marginBottom: 20,
  }}}>
  <Text style={{color: COLORS.light, fontWeight: 'bold'}}>
    Don't have an account ?
  </Text>
  <TouchableOpacity onPress={() => navigation.navigate('SignUp')}>
    <Text style={{color: COLORS.pink, fontWeight: 'bold'}}>
      Sign up
    </Text>
  </TouchableOpacity>
</View>
</View>
</KeyboardAvoidingView>
</ScrollView>
)
}

export default LoginScreen;
const styles = StyleSheet.create({
  brandView: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
  },
  textWelcome: {
    fontSize: 25,
    color: '#F781BE',
    fontWeight: 'bold',
    textTransform: 'uppercase',
    marginTop: 5,
  },
})
```

```

bottomView: {
  flex: 1.5,
  backgroundColor: '#ffffff',
  bottom: 50,
  borderTopStartRadius: 60,
  borderTopEndRadius: 60,
},
laBle: {
  marginVertical: 5,
  fontSize: 14,
  color: '#D8D8D8',
},
box: {
  borderRadius: 10,
  elevation: 5,
  padding: 10,
  height: 200,
},
);

```

❖ Register

```

import React, { useState } from 'react';
import {
  Text, View,
  ScrollView ,
  Dimensions,
  Image,
  TextInput,
  TouchableOpacity,
  Alert,
  StatusBar,
}
from 'react-native';
import Icon from 'react-native-vector-icons/MaterialIcons';
import IconAntDesign from 'react-native-vector-icons/AntDesign';
import COLORS from '../constants/Colors';
import STYLES from '../styles/index';
import { authentication } from '../firebase/Fire';
import { createUserWithEmailAndPassword } from "firebase/auth";

```

```

const SignUpScreen = ({navigation}) => {
  const [email, setIsEmail] = useState("")
  const [password, setIsPassword] = useState("")
  const RegisterInUser = async() => {

```

```

        await createUserWithEmailAndPassword(authentication,email, password)
        .then((userCredential) => {
            Alert.alert(
                'Alert Title',
                'Đăng ký thành công ! ' + email,
                [
                    {text: 'Cancel', onPress: () => console.log('thanhcong')},
                    {text: 'OK', onPress: () => navigation.navigate('Login') }
                ],
                { cancelable: false }
            );
        })
        .catch((error) => {
            Alert.alert(
                'Alert Title',
                'Đăng ký thất bại: '
                [
                    {text: 'Cancel', onPress: () => console.log('Cancel Pressed'), style:'cancel'},
                    {text: 'OK', onPress: () => console.log('Cancel Pressed')}
                ],
                { cancelable: false }
            );
            console.log(error)
        });
        //this.props.navigation.navigate('Home');
        //alert('Đăng nhập thành công')
    }
    return (
        <ScrollView
            style={{paddingHorizontal: 15, flex: 1, backgroundColor: COLORS.white,}}
            showsVerticalScrollIndicator={false}>
            <View style={{height: 100, width:Dimensions.get('window').width}}>
                <TouchableOpacity onPress={() => navigation.goBack()}>
                    <IconAntDesign
                        name="left"
                        size={20}
                        style={{color: '#900', fontWeight: 'bold', marginTop:50}}>
                        <Text style={{fontSize: 18,}}>Sign in</Text>
                    </IconAntDesign>
                </TouchableOpacity>
            </View>
            <View style={{flexDirection: 'row', alignItems:'center', flexDirection:'row', alignItems:'center', justifyContent:'center'}}>
                <Text style={{fontWeight: 'bold', fontSize: 22, color: COLORS.dark}}>

```

```
    HCMUTE
  </Text>
  <Text
    style={{fontWeight: 'bold', fontSize: 22, color: COLORS.secondary}}>
    {' '} REEE
  </Text>
</View>
<View style={{marginTop: 50}}>
  <Text style={{fontSize: 27, fontWeight: 'bold', color: COLORS.dark}}>
    Welcome Back,
  </Text>
  <Text style={{fontSize: 19, fontWeight: 'bold', color: COLORS.light}}>
    Sign up to continue
  </Text>
</View>
<View style={{marginTop: 20}}>
  <View style={STYLES.inputContainer}>
    <Icon
      name="person-outline"
      color={COLORS.light}
      size={20}
      style={STYLES.inputIcon}
    />
    <TextInput placeholder="Name" style={STYLES.input} />
  </View>
  <View style={STYLES.inputContainer}>
    <Icon
      name="mail-outline"
      color={COLORS.light}
      size={20}
      style={STYLES.inputIcon}
    />
    <TextInput
      placeholder="Email" style={STYLES.input}
      value = {email}
      onChangeText={(email) => setIsEmail(email)}
    />
  </View>
  <View style={STYLES.inputContainer}>
    <Icon
      name="lock-outline"
      color={COLORS.light}
      size={20}
      style={STYLES.inputIcon}
    />
```

```
<TextInput
    placeholder="Password"
    style={STYLES.input}
    secureTextEntry
    value = {password}
    onChangeText={(password) => setIsPasswork(password)}
/>
</View>
<TouchableOpacity style={STYLES.btnExitPrimary} onPress={RegisterInUser}>
    <Text style={{color: '#fff', fontWeight: 'bold', fontSize: 18}}>
        Sign Up
    </Text>
</TouchableOpacity>
<View
    style={{
        marginVertical: 20,
        flexDirection: 'row',
        justifyContent: 'center',
        alignItems: 'center',
    }}>
    <View style={STYLES.line}></View>
    <Text style={{marginHorizontal: 5, fontWeight: 'bold'}}>OR</Text>
    <View style={STYLES.line}></View>
</View>
<View
    style={{
        flexDirection: 'row',
        justifyContent: 'space-between',
    }}>
    <View style={STYLES.btnExitSecondary}>
        <Text style={{fontWeight: 'bold', fontSize: 16}}>
            Sign up with
        </Text>
        <Image
            style={STYLES.btnExitImage}
            source={require('../image/facebook.jpeg')}
        />
    </View>
    <View style={{width: 10}}></View>
    <View style={STYLES.btnExitSecondary}>
        <Text style={{fontWeight: 'bold', fontSize: 16}}>
            Sign up with
        </Text>
        <Image
            style={STYLES.btnExitImage}
```

```

        source={require('../image/google.jpeg')}
      />
    </View>
  </View>
</View>

<View
  style={{{
    flexDirection: 'row',
    alignItems: 'flex-end',
    justifyContent: 'center',
    marginTop: 40,
    marginBottom: 20,
  }}>
  <Text style={{color: COLORS.light, fontWeight: 'bold'}}>
    Already have an account ?
  </Text>
  <TouchableOpacity onPress={() => navigation.navigate('Login')}>
    <Text style={{color: COLORS.pink, fontWeight: 'bold'}}>
      Sign in
    </Text>
  </TouchableOpacity>
</View>
<ScrollView>
);
};

export default SignUpScreen;

```

❖ Ghi nhớ đăng nhập

```

import AsyncStorage from "@react-native-async-storage/async-storage";
export const Init = () => {
  return async dispatch => {
    let token = await AsyncStorage.getItem('token');
    if (token !== null) {
      console.log('token fetched');
      dispatch({
        type: 'LOGIN',
        payload: token,
      })
    }
  }
}

export const Login = (email, password, authen) => {

```

```
return async dispatch => {
  let token = null;
  if (email != "" && password != "") {
    token = authen;
    // here we can use login api to get token and then store it
    await AsyncStorage.setItem('token', token);
    //console.log('token stored');
  }
  dispatch({
    type: 'LOGIN',
    payload: token,
  })
}
}
```

```
export const Logout = () => {
  return async dispatch => {
    await AsyncStorage.clear();
    dispatch({
      type: 'LOGOUT'
    })
  }
}
```

```
import { createStore, combineReducers, applyMiddleware } from "redux";
import thunk from 'redux-thunk';
import Reducers from "./reducers";
const RootReducers = combineReducers({
  // reducers
  Reducers,
});

```

```
export const store = createStore(RootReducers, applyMiddleware(thunk));
```

```
const initialState = {
  authToken: null,
}

export default (state = initialState, action) => {
  switch(action.type) {
    case 'LOGIN':
```

```

    return {
      ...state, //copy all previous states
      authToken: action.payload,
    }
  case 'LOGOUT':
    return {
      authToken: null,
    }
  default:
    return state;
}
}

```

❖ Giao diện giám sát năng lượng

```

import React, { useEffect, useRef, useState, memo } from 'react'
import { Animated, Switch, FlatList, Image, StyleSheet, TouchableOpacity, View, Dimensions, ToastAndroid, ImageBackground } from 'react-native'
import { Surface, Text, Button, List } from 'react-native-paper'
import MyHeader from './components/MyHeader'
import BottomTab from './components/BottomTab'
import FabScreen from './FabScreen';
import PowerSpeedometer from './PowerSpeedometer';

import '@react-native-firebase/app';
import firestore from '@react-native-firebase/firestore';
import Styles from './common/Styles';
import Colors from './constants/Colorss';
import * as Animatable from 'react-native-animatable'
import Icon, { Icons } from './components/Iconss';
import { Animations } from './constants/Animations';
import { set } from 'react-native-reanimated'
import { useSelector } from 'react-redux';
const colorAr = [
  '#637aff',
  '#60c5a8',
  '#CCCCCC',
  '#ff5454',
  '#039a83',
  '#dcb834',
  '#8f06e4',
  'skyblue',
  '#ff4c98',
]
let dataNo = true;
let date = new Date().getDate(); //To get the Current Date

```

```

let month = new Date().getMonth() + 1; //To get the Current Month
let year = new Date().getFullYear(); //To get the Current Year
const bgColor = (i) => colorAr[i % colorAr.length];
const ListItem = ({ item, index, animation, navigation }) => {
  const [isEnabled, setIsEnabled] = useState('white');
  const [warningPower, setWarningPW] = useState(70);
  setTimeout(()=>{
    if (item.Power){
      if(isEnabled === 'white')
      {
        setIsEnabled('red')
      } else {
        setIsEnabled('white')
      }
    }
  }, 1500)

  const token = useSelector(state => state.Reducers.authToken).toString();
  const toggleSwitch = async() => {
    await firestore()
      .collection(`energy/${token}/${year}/${month}/${date}`)
      .doc(item.Key.toString())
      .update({
        GetDoc:
        {
          value: !item.GetDoc.value,
          resetpz: true,
          //resetpz: item.GetDoc.resetpz,
        }
      })
      .then(() => {
        //console.log('User updated!');
        //setIsEnabled(!item.value);
      });
  }
  //console.log(animation)
  return (
    <Animatable.View
      animation={ dataNo == true ? animation : Animations[0]}
      duration={1000}
      delay={index * 300}
    >
    <View style={{
      height: 350, // chieu cao khung
      width: Dimensions.get('window').width / 2 - 16,

```

```

        backgroundColor: item.Power <= warningPower ? 'white' : isEnabled,
        margin: 8,
        borderRadius: 10,
    }}>

<View style={{flexDirection: 'row', alignItems: 'center', justifyContent: 'space-between', marginLeft: 5, marginTop: 5, marginRight: 5}}>
    <View>
        <Text style={styles.name}>Phòng {item.Key}</Text>
    </View>
    <View style={{justifyContent: 'flex-end'}}>
        <Switch
            trackColor={{ false: "#767577", true: "#81b0ff" }}
            thumbColor={item.GetDoc.value ? "#f5dd4b" : "#f4f3f4" }
            ios_backgroundColor="#3e3e3e"
            onChange={toggleSwitch}
            value={item.GetDoc.value} />
    </View>
</View>

<PowerSpeedometer datapower={item.Power}></PowerSpeedometer>
<TouchableOpacity
    activeOpacity={0.7}
    onPress={() => navigation.navigate('Screen')}>
    <View style={[styles.image, { backgroundColor: bgColor(index) }]}>
        <Text>Điện áp: {item.Voltage} V</Text>
        <Text>Dòng điện: {item.Current} V</Text>
        <Text>Công suất: {item.Power} W</Text>
        <Text>Công suất tiêu thụ: {item.TotalEnergy} Kwh</Text>
        <Text>Số nước: {item.ReactivePower} km3</Text>
        <Text>Hệ số cos phi: {item.PowerFactor} </Text>
    </View>
    <View>
    </View>
</TouchableOpacity>
<View style={styles.detailsContainer}>
    <Text>Người thuê: {item.name}</Text>
    <TouchableOpacity
        style={{ backgroundColor: 'red' }}
        onPress={() => navigation.navigate('List')}>
        <Icon type={Icons.Feather} name="more-vertical" size={20} />
    </TouchableOpacity>
</View>

```

```

        </View>
    </View>

    </Animatable.View>
)
}

const CONTAINER_HEIGHT = 50;
function HomeEnergy({ route, navigation }) {
    const [dataone, setDataone] = useState([]);
    const token = useSelector(state => state.Reducers.authToken).toString();

    useEffect(() => {
        async function fetchData() {
            try {
                await
                firestore().collection(`energy/${token}/${year}/${month}/${date}`).onSnapshot(onResult
=> {
                    let user = []
                    onResult.forEach(documentSnapshot => {
                        user.push(documentSnapshot.data())
                    });

                    user.length >= 1 ? setDataone(user) : setDataone([]);
                    //console.log('documentSnapshot', dataone)
                    //data.current = true;z
                    //data = false;
                });
                setTimeout(() => {
                    dataNo = false;
                    //console.log(data)
                },100)
            } catch (error) {
                console.log('Failed to fetch posts: ', error.message);
            }
        }
        fetchData();
    }
    , [setDataone]);
    const scrollY = useRef(new Animated.Value(0)).current;
    const offsetAnim = useRef(new Animated.Value(0)).current;
    const clampedScroll = Animated.diffClamp(
        Animated.add(
            scrollY.interpolate({

```

```

    inputRange: [0, 1],
    outputRange: [0, 1],
    extrapolateLeft: 'clamp',
  )),
  offsetAnim,
),
0,
CONTAINER_HEIGHT
)
var _clampedScrollValue = 0;
var _offsetValue = 0;
var _scrollValue = 0;
useEffect(() => {
  scrollY.addListener(({ value }) => {
    const diff = value - _scrollValue;
    _scrollValue = value;
    _clampedScrollValue = Math.min(
      Math.max(_clampedScrollValue + diff, 0),
      CONTAINER_HEIGHT,
    )
  });
  offsetAnim.addListener(({ value }) => {
    _offsetValue = value;
  })
}, []);
}

var scrollEndTimer = null;
const onMomentumScrollBegin = () => {
  clearTimeout(scrollEndTimer)
}
const onMomentumScrollEnd = () => {
  const toValue = _scrollValue > CONTAINER_HEIGHT &&
    _clampedScrollValue > (CONTAINER_HEIGHT) / 2
    ? _offsetValue + CONTAINER_HEIGHT : _offsetValue - CONTAINER_HEIGHT;

  Animated.timing(offsetAnim, {
    toValue,
    duration: 500,
    useNativeDriver: true, // <-- Add this
  }).start();
}
const onScrollEndDrag = () => {
  scrollEndTimer = setTimeout(onMomentumScrollEnd, 250);
}

```

```

const headerTranslate = clampedScroll.interpolate({
  inputRange: [0, CONTAINER_HEIGHT],
  outputRange: [0, -CONTAINER_HEIGHT],
  extrapolate: 'clamp',
})
const opacity = clampedScroll.interpolate({
  inputRange: [0, CONTAINER_HEIGHT - 20, CONTAINER_HEIGHT],
  outputRange: [1, 0.05, 0],
  extrapolate: 'clamp',
})
const bottomTabTranslate = clampedScroll.interpolate({
  inputRange: [0, CONTAINER_HEIGHT],
  outputRange: [0, CONTAINER_HEIGHT * 2],
  extrapolate: 'clamp',
})
const bottomTabTranslateFab = clampedScroll.interpolate({
  inputRange: [0, CONTAINER_HEIGHT],
  outputRange: [0, CONTAINER_HEIGHT * 60],
  extrapolate: 'clamp',
})
const viewRef = useRef(null);
const animation = Animations[Math.floor(Math.random() * Animations.length)]
//console.log('=====');
//console.log(Math.floor(Math.random() * Animations.length), Math.random() *
Animations.length, Animations.length);
//console.log('=====');

const renderItem = ({ item, index }) => (
  <ListItem item={item} index={index} animation={animation}
navigation={navigation} />
)

const ListEmptyComponent = () => {
  const anim = {
    0: { translateY: 0 },
    0.5: { translateY: 50 },
    1: { translateY: 0 },
  }
  return (
    <View style={styles.listEmpty}>
      <Animatable.Text
        animation={anim}
        easing="ease-in-out"
        duration={3000}
        style={{ fontSize: 24 }}>

```

```

        iterationCount="infinite">
        Mang yeu
    </Animatable.Text>
</View>
)
}
useEffect(() => {
    const unsubscribe = navigation.addListener('focus', () => {
        viewRef.current.animate({ 0: { opacity: 0.5, }, 1: { opacity: 1 } });
    })
    // ToastAndroid.show(animation+ ' Animation', ToastAndroid.SHORT);
    return () => unsubscribe;
}, [navigation])
return (
    <View style={[Styles.container]}>
        <ImageBackground source={require('../image/spkt.jpeg')}
            style={{ height:Dimensions.get('window').height, width:
Dimensions.get('window').width }}>
            <Animatable.View
                ref={viewRef}s
                easing={'ease-in-out'}
                duration={500}
                style={styles.container}>
                <Text style={{fontSize: 35}}>Nang luong phong tro</Text>
                <Animated.FlatList
                    onScroll={Animated.event(
                        [{ nativeEvent: { contentOffset: { y: scrollY } } }],
                        { useNativeDriver: true }
                    )}
                    data={dataone}
                    keyExtractor={(_, i) => String(i)}
                    numColumns={2}
                    renderItem={renderItem}
                    showsVerticalScrollIndicator={false}
                    contentContainerStyle={{ paddingBottom: 100 }}
                    //contentContainerStyle={styles.contentContainerStyle}
                    onMomentumScrollBegin={onMomentumScrollBegin}
                    onMomentumScrollEnd={onMomentumScrollEnd}
                    onScrollEndDrag={onScrollEndDrag}
                    scrollEventThrottle={1}
                    ListEmptyComponent={ListEmptyComponent}
                >
                </Animated.FlatList>
            </Animatable.View>
        </ImageBackground>

```

```

    <Animated.View style={[styles.view, { top: 0, transform: [{ translateY: headerTranslate }] }]}>
      <MyHeader
        menu
        title={route.name}
        right="search"
        style={[styles.header, { opacity }]}>
      />
    </Animated.View>

    <Animated.View style={[styles.viewtab, { bottom: 40, transform: [{ translateY: bottomTabTranslateFab }] }]}>
      <FabScreen dataflist={dataone} ></FabScreen>
    </Animated.View>

    <Animated.View style={[styles.view, { bottom: 0, transform: [{ translateY: bottomTabTranslate }] }]}>
      <Surface style={[styles.rowContainer, styles.bottomBar]}>
        <BottomTab navigation={navigation} route ={route.name} />
      </Surface>
    </Animated.View>
  </View>
}

export default HomeEnergy;
const styles = StyleSheet.create({
  name: {
    fontWeight: 'bold',
    fontSize: 16,
    color: 'black',
  },
  separator: {
    height: StyleSheet.hairlineWidth,
    backgroundColor: 'rgba(0, 0, 0, .08)',
  },
  listEmpty: {
    height: Dimensions.get('window').height,
    alignItems: 'center',
    justifyContent: 'center',
  },
  listItem: {
    height: 350, // chieu cao khung
    width: Dimensions.get('window').width / 2 - 16,
    backgroundColor: 'white',
  }
});

```

```
margin: 8,  
borderRadius: 10,  
},  
image: {  
height: 150,  
margin: 5,  
borderRadius: 10,  
backgroundColor: Colors.primary,  
},  
detailsContainer: {  
paddingHorizontal: 16,  
paddingVertical: 5,  
flexDirection: 'row',  
justifyContent: 'space-between',  
alignItems: 'center',  
},  
container: {  
flex: 1,  
},  
listScreen:{  
flex: 1,  
},  
viewtab: {  
position: 'absolute',  
left: 0,  
right: 0,  
height: CONTAINER_HEIGHT + 50,  
},  
view: {  
position: 'absolute',  
left: 0,  
right: 0,  
height: CONTAINER_HEIGHT,  
},  
header: {  
borderBottomRightRadius: 16,  
borderBottomLeftRadius: 16,  
marginHorizontal: 4,  
},  
bottomBar: {  
borderTopRightRadius: 16,  
borderTopLeftRadius: 16,  
marginHorizontal: 4,  
},
```

```
contentContainerStyle: {
  paddingTop: CONTAINER_HEIGHT,
  marginTop: 8,
},
rowContainer: {
  flex: 1,
  flexDirection: 'row',
  alignItems: 'center',
  justifyContent: 'space-around',
},
item: {
  marginHorizontal: 10,
  marginBottom: 12,
  elevation: 6,
  borderRadius: 16,
},
title: {
  fontSize: 16,
  fontWeight: '700',
},
caption: {
  color: Colors.darkGray,
},
bottomView: {
  alignItems: 'center',
  flexDirection: 'row',
  padding: 16
},
content: {
  alignItems: 'center',
  flexDirection: 'row',
  marginHorizontal: 16,
  paddingVertical: 8,
},
textContainer: {
  marginHorizontal: 16,
},
avatar: {
  height: 35,
  width: 35,
  borderRadius: 20,
  backgroundColor: Colors.primary,
},
rowView: {
  alignItems: 'center',
```

```

flexDirection: 'row',
justifyContent: 'space-around',
},
icon: {
  marginHorizontal: 10,
},
})

/*
<Image source={require('../image/arrow_down.gif')} style={{width: 10, height:200
}}/>
const [postList, setPostList] = useState([]);
useEffect(() => {
  async function fetchData() {
    try {
      firestore()
        .collection('a32npAcZrnaTQPdbEhi0ODu9NH53')
        .get()
        .then(querySnapshot => {
          querySnapshot.forEach(documentSnapshot => {
            setPostList(documentSnapshot.data());
            console.log(postList);
          });
        });
    } catch (error) {
      console.log('Failed to fetch posts: ', error.message);
    }
  }
  fetchData();
}, []);

```

```

<Animated.View style={[styles.viewtab, { bottom: 40, transform: [{ translateY:
bottomTabTranslateFab }] }]}>
  <FabScreen datalist={dataone} ></FabScreen>
</Animated.View>

```

```

const dataFlastlist = firestore()
  .collection('a32npAcZrnaTQPdbEhi0ODu9NH53')
  .get()
  .then(querySnapshot => {
    querySnapshot.forEach(documentSnapshot => {
      setData(documentSnapshot.data());
      console.log(data);
    });
  });

```

```
    });
    });
*/
```

Vì Code quá nhiều nên mọi người có thể truy cập đường link để dowload