HO CHI MINH CITY, UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEER

ĐỒ ÁN TỔNG HỢP ĐA NGÀNH

Báo cáo

SMART GARDEN

GVHD:	Trần Thanh Bình	
Nhóm sinh viên thực hiện:	Lê Hoàng Nam	1914220
	Trần Đình Nghĩa	1914325
	Lê Thành Long	1913991
	Nguyễn Văn Khoa	1913817
	Hồ Vũ Đại Hải	1913241

HỒ CHÍ MINH CITY

Content

1 Giới thiệu đề tài

1.1 Ý tưởng

Đề tài của nhóm là về hiện thực hệ thống theo dõi tình trạng cây trồng tự động trong môi trường xác định với hỗ trợ tự động hóa trong việc lưu trữ và thực thi trên dữ liệu về cây trồng. Với mô hình thực hiện gồm các hệ thống và thiết bị đèn, máy bơm để người dùng có thể theo dõi và chắm sóc có giới hạn cây trồng trong phạm vi không giới hạn với điều kiện kết nối internet được đảm bảo. Về cơ bản là một mô hình vườn thông minh cơ bản.

1.2 Các yêu cầu về hệ thống

1.2.1 Các yêu cầu về chức năng

1. Cập nhật thông tin: nhiệt độ, độ ẩm, nồng độ CO2, độ pH được ghi nhận thông qua các cảm biến (đã được lắp đặt), được cung cấp cho người dùng tại thời điểm truy cập app
2. Hệ thống tự động tưới: thông qua các thiết bị đầu cuối của người dùng (web app hay mobile app) có thể tùy chỉnh để bật tắt chế độ tự động tưới cho cây.
3. Các tính năng thủ công: người dùng có thể tự điều chỉnh việc tưới tiêu và bật tắt đèn tại khu vực cây trồng không qua tự động.

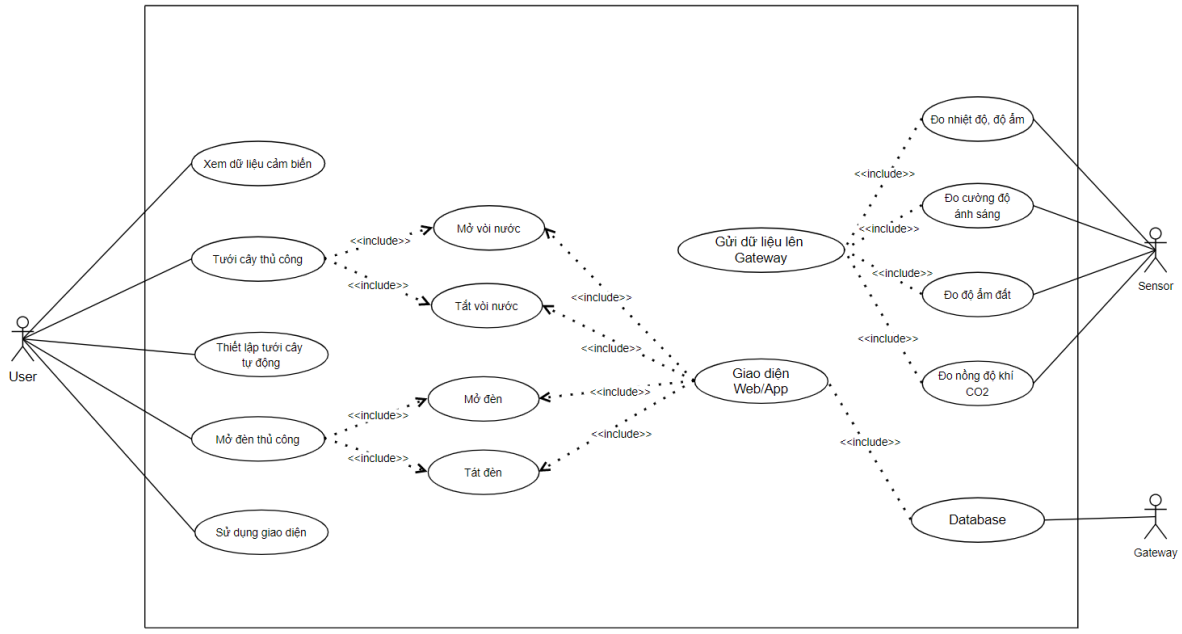
1.2.2 Các yêu câu phi chức năng

1. Các yêu cầu chung:
   - Chỉ có 1 khu vườn (1 bộ hệ thống theo dõi cố định)
   - Thể hiện các thông số dưới dạng đường với tổng dữ liệu thể hiện trên 1 biểu đồ là 20 phần tử mới nhất
   - Chỉ có 1 kho dữ liệu lưu trữ được truy xuất, tuy nhiên lưu trữ trên 2 nên tảng khác nhau là Adafruit và Firebase
   - Các dữ liệu trên các kho dữ liệu được đồng bộ ở gateway
2. Web:
   - Thời gian truy xuất nhanh (không quá 10s)
   - Chạy được trên các browser phổ biến Firefox, Chrome, Safari,...
3. Mobile app:
   - Có thể được đưa lên Google play
   - Chạy trên Android (có thể được mở rộng ra iOS)

1.3 Usecase của hệ thống

Chỗ này thêm vào cái diagram với cái mô tả

1.3.1 Usecase Diagram



Hình 1: Usecase Diagram

1.3.2 Mô tả

1	Use-case name	Xem dữ liệu cảm biến
	Actor	User
	Description	Người dùng xem dữ liệu đo được từ các cảm biến đã lắp đặt
	Preconditions	Giao diện phần mềm chạy tốt
	Normal Flows	Chọn mục Statistics để xem dữ liệu
	Exceptions	Biểu đồ hiển thị dữ liệu chính xác
	Alternative-Flows	Không có

2	Use-case name	Tưới cây thủ công
	Actor	User
	Description	Điều khiển máy bơm nước để tưới cây
	Preconditions	Giao diện phần mềm và máy bơm hoạt động tốt
	Normal Flows	Vào mục Device controller để điều khiển máy bơm
	Exceptions	Máy bơm hoạt động theo điều khiển
	Alternative-Flows	Không có

3

Use-case name	Mở vòi nước
Actor	User
Description	Mở vòi nước bằng nút bấm trên giao diện
Preconditions	Giao diện phần mềm và máy bơm hoạt động tốt
Normal Flows	(1) Vào mục Device controller để điều khiển máy bơm (2) Điều chỉnh tín hiệu nút Pump sang ON
Exceptions	Máy bơm hoạt động
Alternative-Flows	Không có

4

Use-case name	Tắt vòi nước
Actor	User
Description	Tắt vòi nước bằng nút bấm trên giao diện
Preconditions	Giao diện phần mềm và máy bơm hoạt động tốt
Normal Flows	(1) Vào mục Device controller để điều khiển máy bơm (2) Điều chỉnh tín hiệu nút Pump sang OFF
Exceptions	Máy bơm ngừng hoạt động
Alternative-Flows	Không có

5

Use-case name	Thiết lập tưới cây tự động
Actor	User
Description	Tưới cây tự động dựa theo độ ẩm đất
Preconditions	Giao diện phần mềm và máy bơm hoạt động tốt
Normal Flows	
Exceptions	Cây tự động tưới khi độ ẩm đất thấp và ngừng tưới khi độ ẩm đất đạt ngưỡng
Alternative-Flows	Không có

6

Use-case name	Mở đèn thủ công
Actor	User
Description	Bật/tắt đèn bằng nút bấm trên giao diện
Preconditions	Giao diện phần mềm và đèn hoạt động tốt
Normal Flows	Vào mục Device controller để điều khiển đèn led
Exceptions	Đèn bật/tắt theo điều khiển
Alternative-Flows	Không có

7

Use-case name	Mở đèn
Actor	User
Description	Bật đèn bằng nút bấm trên giao diện
Preconditions	Giao diện phần mềm và đèn hoạt động tốt
Normal Flows	(1) Vào mục Device controller để điều khiển đèn led (2) Điều chỉnh tín hiệu nút LED sang ON
Exceptions	Đèn bật
Alternative-Flows	Không có

8

Use-case name	Tắt đèn
Actor	User
Description	Tắt đèn bằng nút bấm trên giao diện
Preconditions	Giao diện phần mềm và đèn hoạt động tốt
Normal Flows	(1) Vào mục Device controller để điều khiển đèn led (2) Điều chỉnh tín hiệu nút LED sang ON
Exceptions	Đèn tắt
Alternative-Flows	Không có

9

Use-case name	Sử dụng giao diện
Actor	User
Description	Người dùng thao tác trên giao diện Web/App
Preconditions	Giao diện hoạt động tốt
Normal Flows	Tải App và sử dụng trên điện thoại
Exceptions	Người dùng có thể thao tác với thiết bị qua giao diện
Alternative-Flows	Truy cập Website để sử dụng online

10

Use-case name	Đo nhiệt độ, độ ẩm
Actor	Sensor
Description	Đo nhiệt độ và độ ẩm của môi trường
Preconditions	Cảm biến đang bật
Normal Flows	(1) Lắp đặt cảm biến vào mạch Microbit (2) Nạp chương trình
Exceptions	Cảm biến đo được nhiệt độ và độ ẩm
Alternative-Flows	Không có

11

Use-case name	Đo cường độ ánh sáng
Actor	Sensor
Description	Đo cường độ ánh sáng của môi trường
Preconditions	Cảm biến đang bật
Normal Flows	(1) Lắp đặt cảm biến vào mạch Microbit (2) Nạp chương trình
Exceptions	Cảm biến đo được cường độ ánh sáng
Alternative-Flows	Không có

12

Use-case name	Đo độ ẩm đất
Actor	Sensor
Description	Đo độ ẩm đất của môi trường
Preconditions	Cảm biến đang bật
Normal Flows	(1) Lắp đặt cảm biến vào mạch Microbit (2) Nạp chương trình
Exceptions	Cảm biến đo được độ ẩm đất
Alternative-Flows	Không có

13

Use-case name	Đo nồng độ khí CO2
Actor	Sensor
Description	Đo nồng độ khí CO2 của môi trường
Preconditions	Cảm biến đang bật
Normal Flows	(1) Lắp đặt cảm biến vào mạch Microbit (2) Nạp chương trình
Exceptions	Cảm biến đo được nồng độ CO2
Alternative-Flows	Không có

14

Use-case name	Gửi dữ liệu lên Gateway
Actor	Sensor
Description	Gửi dữ liệu cảm biến lên Gateway
Preconditions	Cảm biến đang bật
Normal Flows	(1) Lắp đặt cảm biến vào mạch Microbit (2) Nạp chương trình (3) Chạy code Gateway
Exceptions	Dữ liệu được gửi thành công
Alternative-Flows	Không có

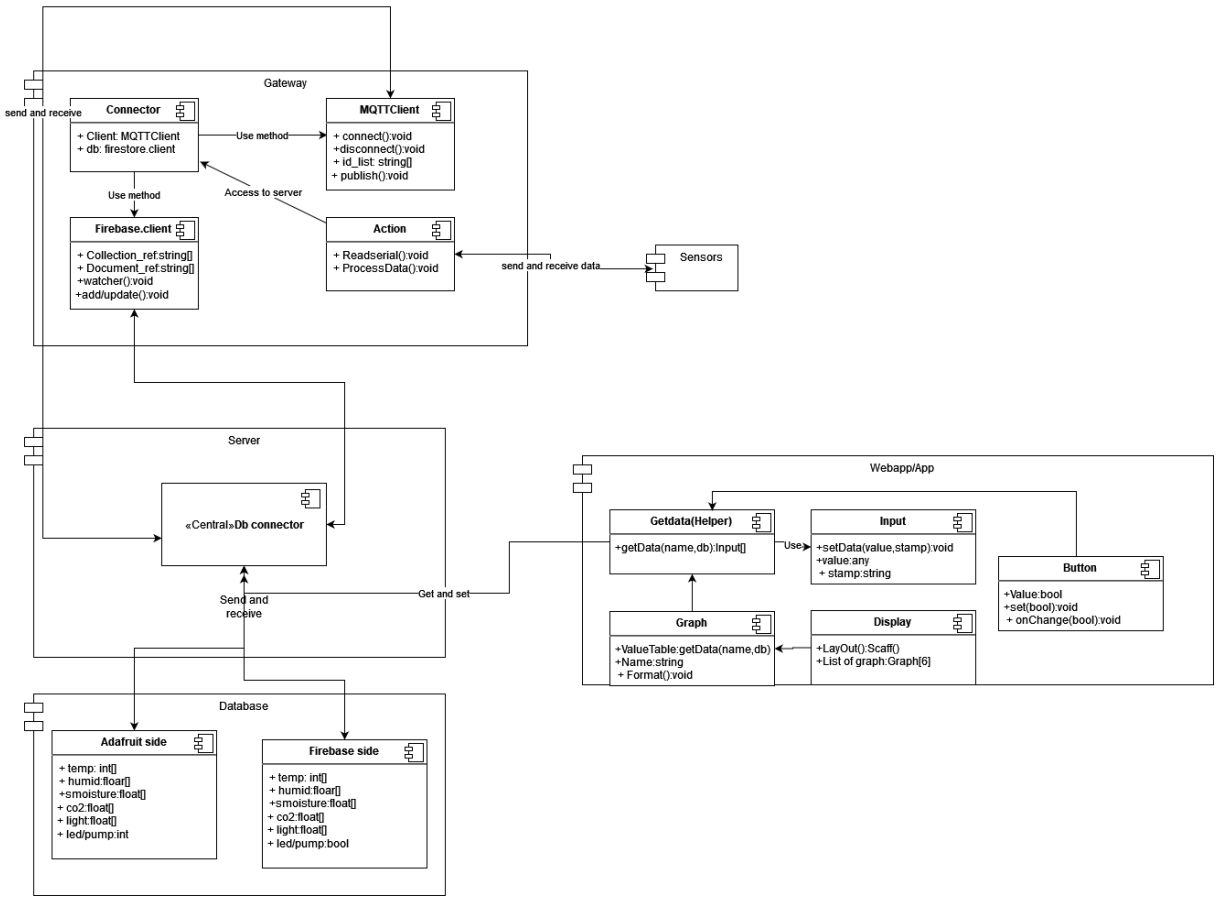
15

Use-case name	Database
Actor	Gateway
Description	Dữ liệu từ Gateway được đẩy lên Database
Preconditions	Sever của database hoạt động bình thường
Normal Flows	Gửi dữ liệu từ Gateway lên Database
Exceptions	Đẩy dữ liệu lên database thành công
Alternative-Flows	Không có

16

Use-case name	Giao diện Web/App
Actor	Gateway
Description	Giao diện hiển thị các thông số do cảm biến đo được và các nút bấm điều khiển động cơ
Preconditions	Có dữ liệu từ Gateway truyền lên Database
Normal Flows	
Exceptions	Người dùng có thể theo dõi và thao tác với khu vườn thông qua giao diện
Alternative-Flows	Không có

1.4 Component Diagram



Hình 2: Component/Class Diagram

Qua sơ đồ trên, có thể thấy cấu trúc hệ thống có thể được chia thành 4 khối lớn là Gateway, Server, App và Database. Các khối này gồm các chức năng:

- Gateway: trực tiếp giao tiếp dữ liệu từ mạch microbit
- Server: host DB và Webapp
- App: Endpoint cho người dùng để tương tác với hệ thống
- Database: nơi lưu trữ các thông tin từ Gateway

2 Sensor implementation

2.1 Microbit

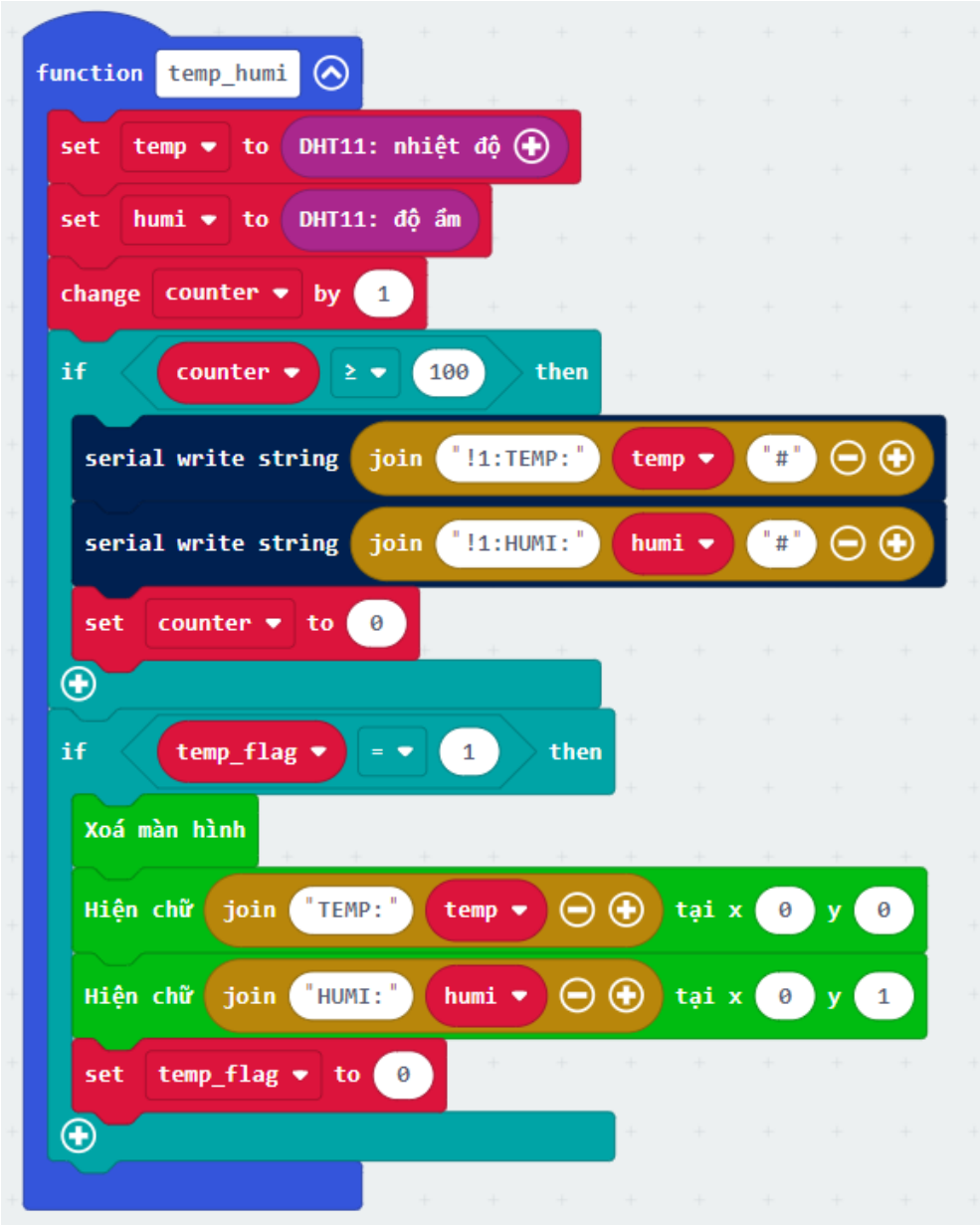
Đây là bộ kiểm soát trung tâm, có tác dụng điều khiển, tương tác với quá trình nhận và gửi dữ liệu sensor qua server

2.2 Adapter 5V

Có tác dụng cung cấp thêm điện năng cho hệ thống

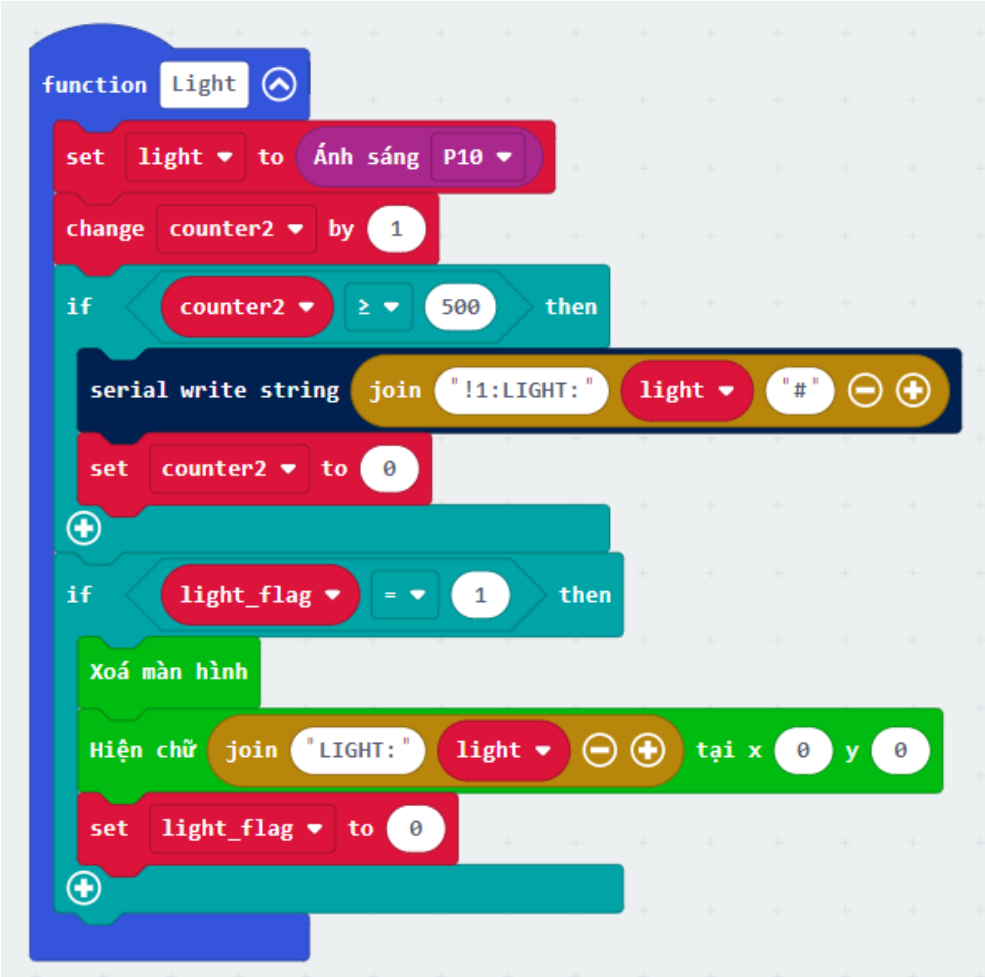
2.3 DHT11

- **Function:** Đo nhiệt độ và để thông báo lên máy chủ IoT (Adafruit). Sensor này có thể giúp chung ta kiểm tra và kiểm soát nhiệt độ, độ ẩm sao cho phù hợp với cây
- **The format:** !1:TEMP:xx
- **The format:** !1:HUMI:xx



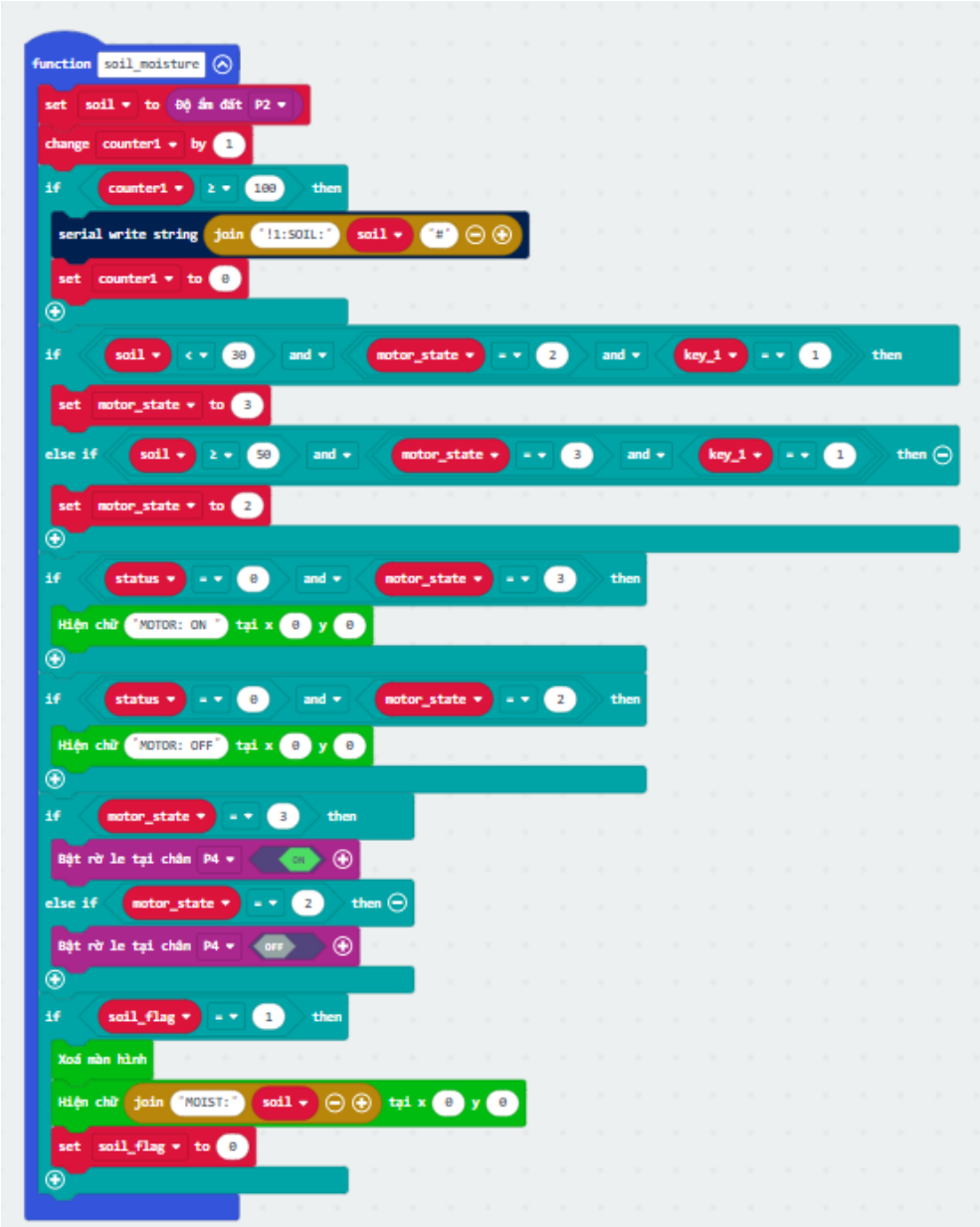
2.4 Light sensor and 2-color single LED

- **Function:** Bật/Tắt đèn thủ công, phụ thuộc vào điều kiện ánh sáng mà ta có thể bật lên để tạo điều kiện cho cây quang hợp tốt hơn
- **The format:** !1:LIGHT:xx



2.5 Soil moisture

- **Function:** Đo độ ẩm của đất, từ đó có thể biết được lượng nước mà cây đang cần thêm
- **The format:** !1:SOIL:xx

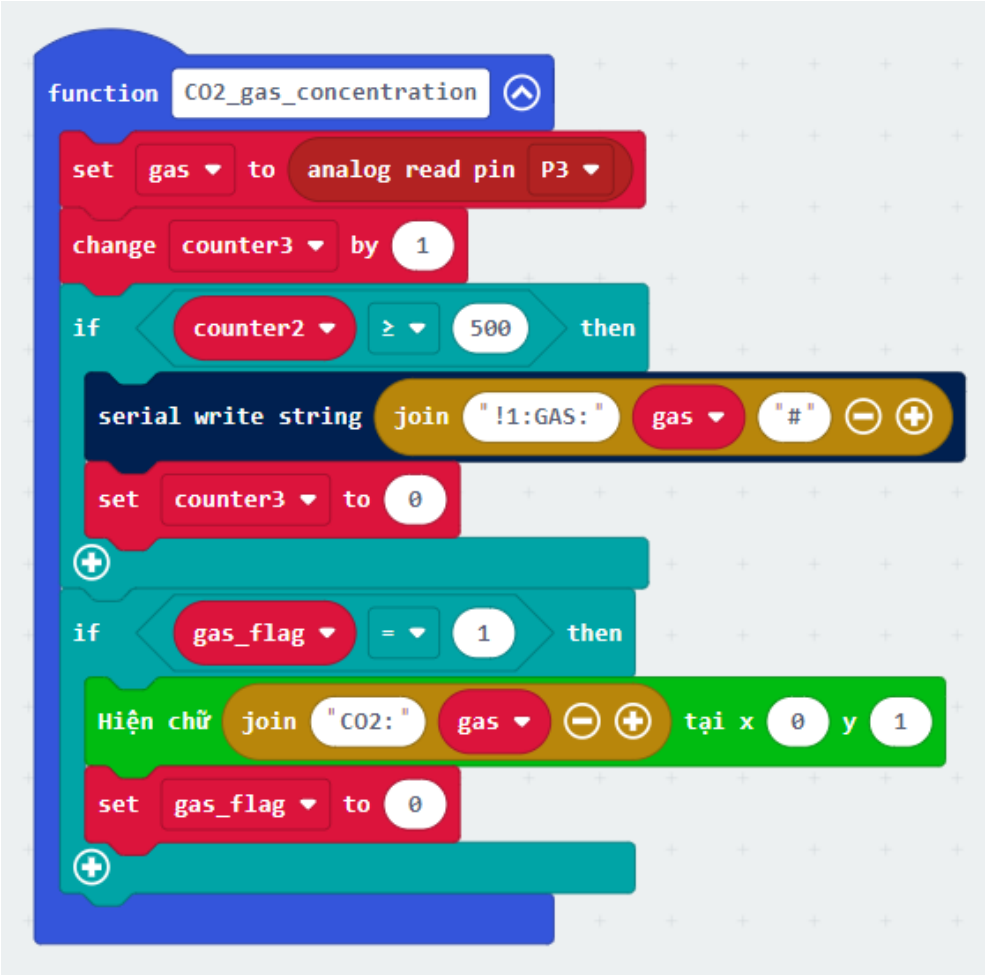


2.6 Relay circuit and Mini pump

- **Function:** Để bật/tắt rơ-le, từ đó điều khiển bật/tắt máy bơm để bơm nước vào đất
- **The format:** !1:PUMP:xx

2.7 Gas sensor

- **Function:** Do nồng độ CO2 trong không khí, từ đó đánh giá được sức khoẻ của cây
- **The format:** !1:CO2:xx



3 Gateway implementation

3.1 Input

3.1.1 Nhiệt độ - độ ẩm

```
if splitData[1] == "TEMP":
    client.publish("microbit-temp", splitData[2])
    collection_ref['temperature_data'].add({
        'value': push_data
        })
elif splitData[1] == "HUMI":
    client.publish("microbit-humid", splitData[2])
    collection_ref['humid_data'].add({
        'value': push_data
        })
```

3.1.2 Độ ẩm đất

```
elif splitData[1] == "SOIL":
    client.publish("microbit-soil-moisture", splitData[2])
    collection_ref['soil_moisture_data'].add({
        'value': push_data
        })
```

3.1.3 Nồng độ CO2

```
elif splitData[1] == "CO2":
    client.publish("microbit-co2", splitData[2])
    collection_ref['co2_data'].add({
        'value': push_data
        })
```

3.1.4 Ánh sáng

```
elif splitData[1] == "LIGHT":
    client.publish("microbit-light", splitData[2])
    collection_ref['light_data'].add({
        'value': push_data
        })
```

3.2 Output

3.2.1 Led

```
def on_snapshot_led(doc_snapshot, changes, read_time):
    global ser
    state = 'ON'
    for doc in doc_snapshot:
        state = doc.to_dict().get('state')
    try:
        ser.write(f"{1 if state == 'ON' else 0}#".encode())
    except Exception as e:
        print(e)
```

3.2.2 Máy bơm

```
def on_snapshot_pump(doc_snapshot, changes, read_time):
    state = 'ON'
    for doc in doc_snapshot:
        state = doc.to_dict().get('state')
    try:
        ser.write(f"{3 if state == 'ON' else 2}#".encode())
    except Exception as e:
        print(e)
```

3.3 UART received data

Việc trích xuất dữ liệu từ micro được hiện thực thông qua giao thức sau đây:

Listing 1: Reading/decoding from Serial port

```
def readSerial():
    bytesToRead = ser.inWaiting()
    if (bytesToRead > 0):
        global mess
        mess = mess + ser.read(bytesToRead).decode("UTF-8")
        # print(mess)
        while ("#" in mess) and ("!" in mess):
            start = mess.find("!")
            end = mess.find("#")
            processData(mess[start:end + 1])
            if (end == len(mess)):
                mess = ""
            else:
                mess = mess[end+1:]
```

Trước tiên, microbit sẽ truyền vào gateway một chuỗi byte liên tục. Tuy nhiên, chuỗi này có thể chứa nhiều thành phần dữ liệu của các sensor khác nhau, như vậy cần tách chuỗi dài này thành các chuỗi riêng biệt ứng với từng sensor để xử lý. Sử dụng hàm find để tìm đầu cuối của 1 chuỗi lẻ, từ đó trích xuất gửi vào hàm xử lý chuyên biệt. Tuy nhiên do việc nhận các dữ có thể truyền thành các gói không đầy đủ, có thể chứa các chuỗi lẻ không hoàn hảo, do đó cần kiểm tra sự tồn tại của các kí hiệu đầu cuối trước trích xuất. Đồng thời cần loại các chuỗi đã trích ra khỏi chuỗi nhận vào để tránh tình trạng lặp dữ liệu.

Listing 2: Extracting core data

```
def processData(data):
    data = data.replace("!", "")
    data = data.replace("#", "")
```

```
    splitData = data.split(":")
    print(splitData)
    push_data = splitData[2]
```

Do data đầu vào của gateway là 1 chuỗi chứa các đối số liên tục, được bắt đầu bằng 1 dấu ! và kết thúc bằng 1 dấu # nên chương trình muốn trích xuất dữ liệu bên trong cần lọc bỏ 2 kí hiệu này đi. Thêm vào đó dữ liệu gồm 2 phần là tên sensor và giá trị của nó vì vậy, cần tách chuỗi 1 lần nữa để lấy được giá trị và tên cụ thể.

3.4 Data uploading process

Listing 3: Example of uploading data

```
        if splitData[1] == "TEMP":
            client.publish("microbit-temp", splitData[2])
            collection_ref['temperature_data'].add({
                'createAt': datetime.now(),
                'value': push_data,
            })
```

Để truyền dữ liệu, do có 2 server db khác nhau nên gateway sẽ thực hiện 2 câu lệnh publish và add lên Firebase và Adafruit. Với Fb, dữ liệu truyền lên dạng dictionary, với Ada chỉ chuyền lên giá trị thuần trên feed. Như code ở trên, để biết truyền lên feed nào và collection nào, dựa vào tên sensor được lọc ra trong chuỗi

3.5 Data received process

Listing 4: Receive module

```
def message(client , feed_id , payload):
    print("Receive Data: " + payload)
    try:
        if isMicrobitConnected:
            ser.write((str(payload) + "#").encode())
    except Exception as e:
        print(e)
def on_snapshot_led(doc_snapshot, changes, read_time):
    global ser
    state = 'ON'
    for doc in doc_snapshot:
        state = doc.to_dict().get('state')
    try:
        post(
```

```
15              f'https://io.adafruit.com/api/v2/{AIO_USERNAME}/feeds/{↩
                    AIO_FEED_IDS[6]}/data',
16              headers={ "X-AIO-Key": AIO_KEY, },
17              data={ "value": "2" if state == "OFF" else "3" }
18          )
19          # ser.write(f"{1 if state == 'ON' else 0}#".encode())
20      except Exception as e:
21          print(e)
22
23  flag = False
24  def on_snapshot_pump(doc_snapshot, changes, read_time):
25      global flag
26      state = 'ON'
27      for doc in doc_snapshot:
28          state = doc.to_dict().get('state')
29      try:
30          flag = True
31          post(
32              f'https://io.adafruit.com/api/v2/{AIO_USERNAME}/feeds/{↩
                    AIO_FEED_IDS[3]}/data',
33              headers={ "X-AIO-Key": AIO_KEY, },
34              data={ "value": "2" if state == "OFF" else "3" }
35          )
36      except Exception as e:
37          print(e)
38      finally:
39          flag = False
40
41  def pump_feed_watcher():
42      current_value = "0"
43      while True:
44          sleep(2)
45          while flag:
46              pass
47          try:
48              respone = get(f'https://io.adafruit.com/api/v2/{↩
                    AIO_USERNAME}/feeds/{AIO_FEED_IDS[3]}/data?limit=1', ↩
                    headers={
49                   "X-AIO-Key": AIO_KEY
50              })
51              value = loads(respone.text)[0]['value']
52              if current_value == value:
53                  continue
54              current_value = value
```

```
55              document_ref['pump'].update({ 'state': "ON" if value == "3↩
                    " else "OFF" })
56              print(value)
57          except:
58              pass
```

Sử dùng hàm message để liên tục gửi dữ liệu về microbit cho mỗi lần cập nhật trên adafruit. Ở đây có 2 hàm nhận dữ liệu để truyền về Adafruit sau khi cập nhật ở firebase là on_snapshot_led, và on_snapshot_pump.

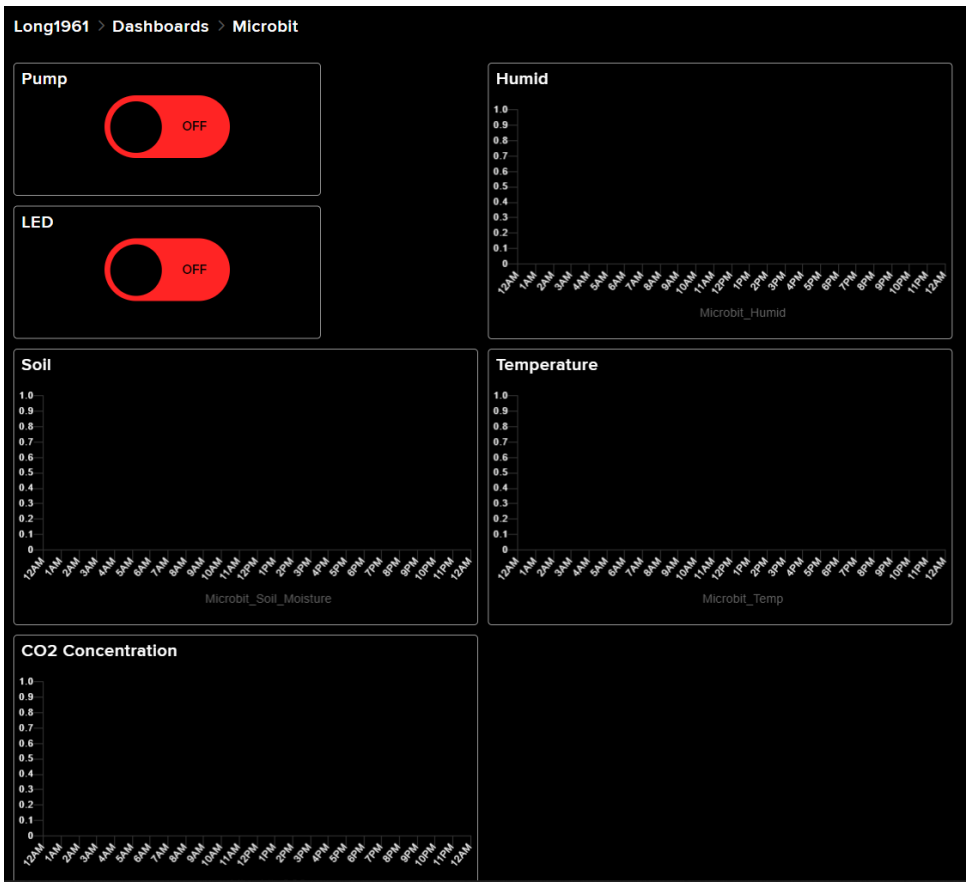
4 AIO Feed and Dashboard

Switch:

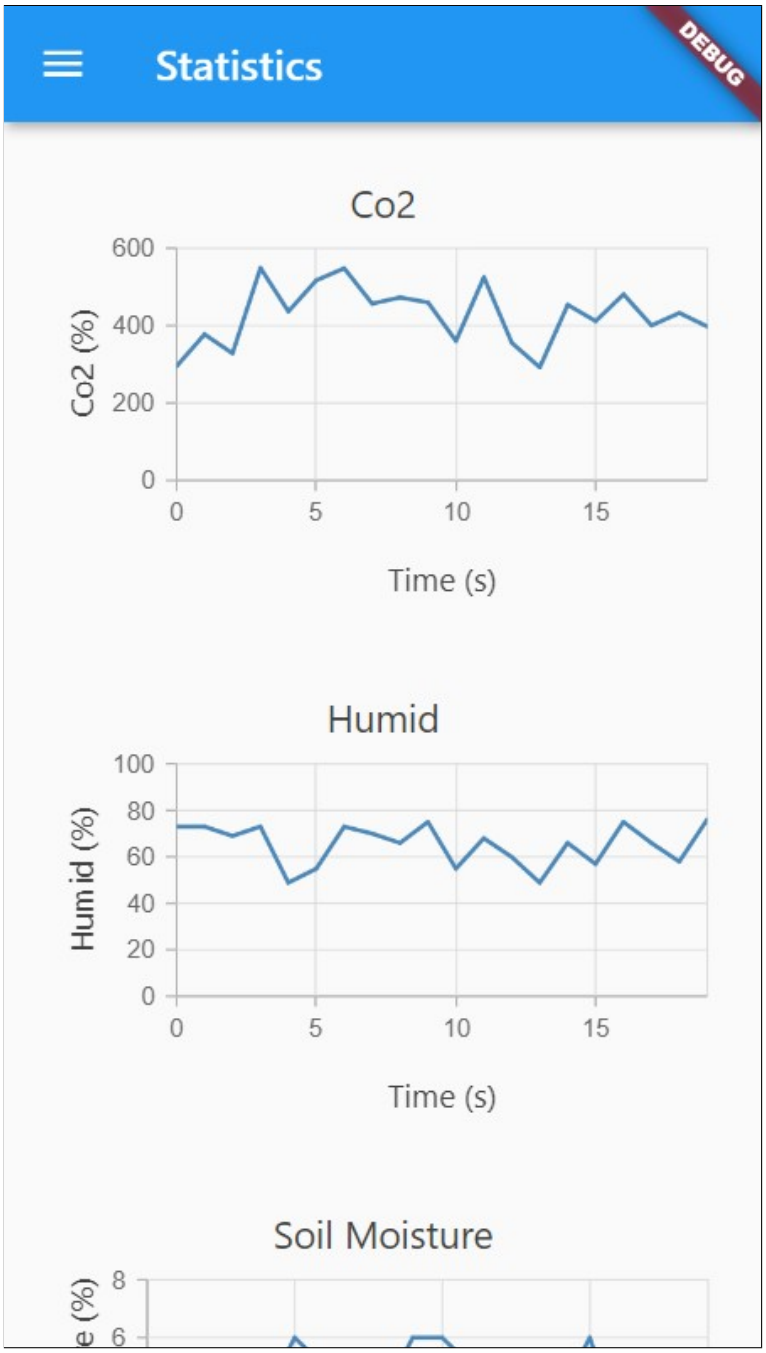
- **LED**: Dùng để bật/tắt đèn led
- **Pump**: Dùng để bật/tắt pump

Graph:

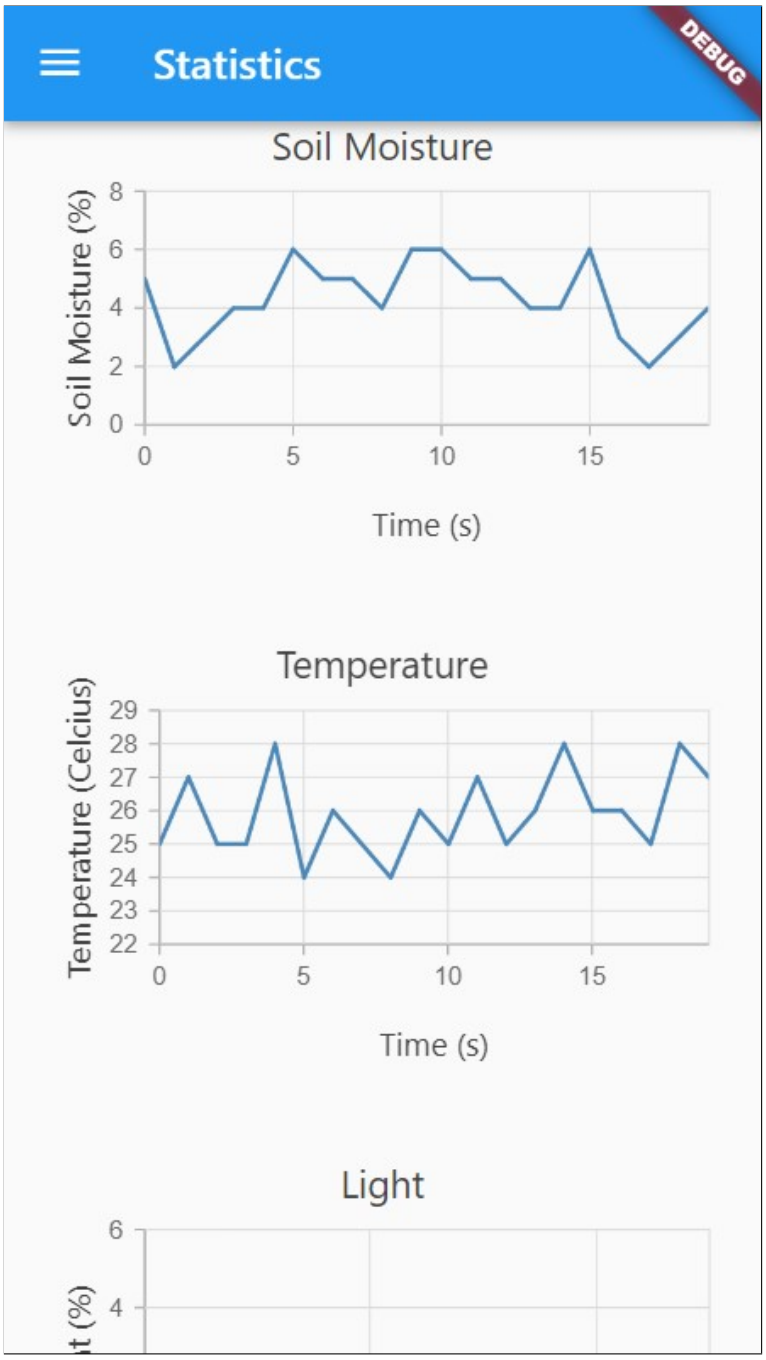
- **Soil**: Cập nhật dữ liệu của feed độ ẩm đất
- **Templ**: Cập nhật dữ liệu của feed nhiệt độ
- **Humid**: Cập nhật dữ liệu của feed ánh sáng
- **CO2 Concentration**: Cập nhật dữ liệu của feed nồng độ CO_2

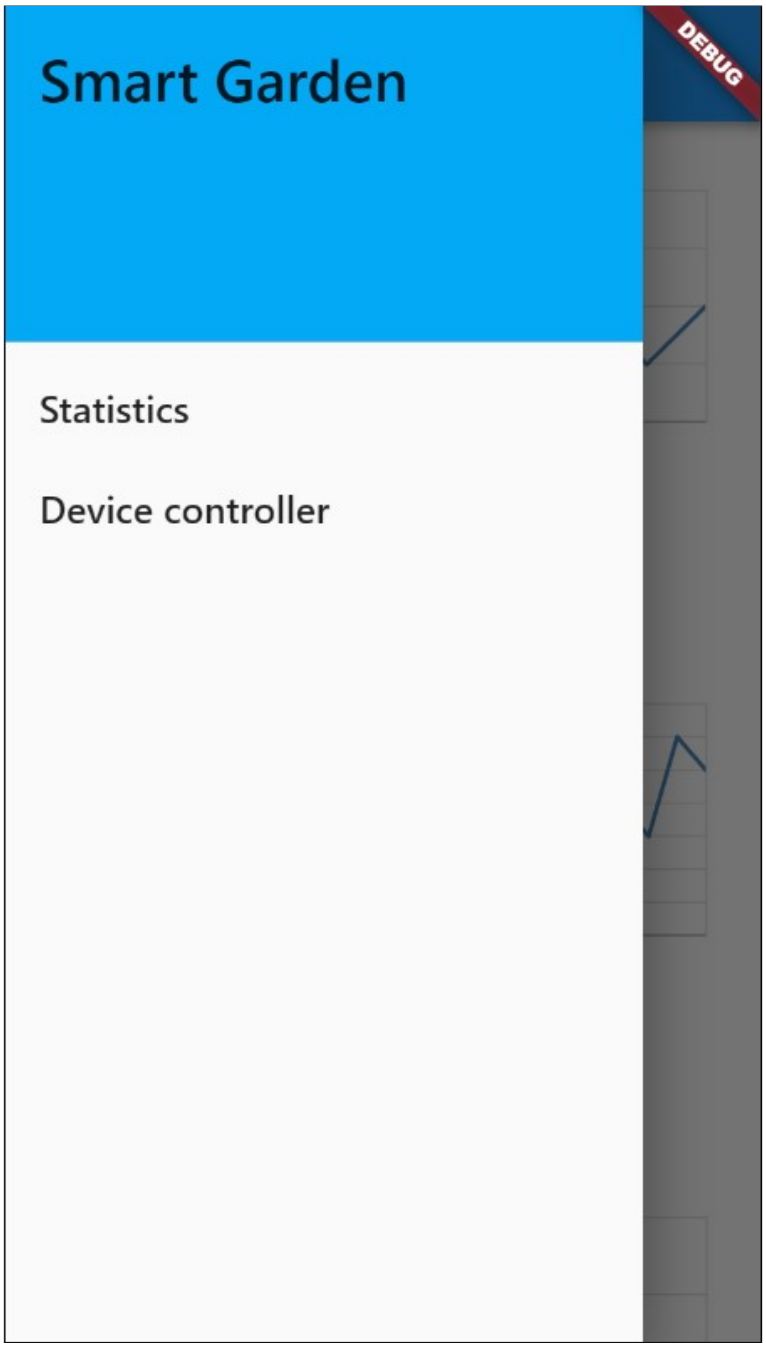

Long1961 > Dashboards > Microbit
Pump
OFF
LED
OFF
Humid
Microbit_Humid
Soil
Microbit_Soil_Moisture
Temperature
Microbit_Temp
CO2 Concentration

5 Giao diện sản phẩm

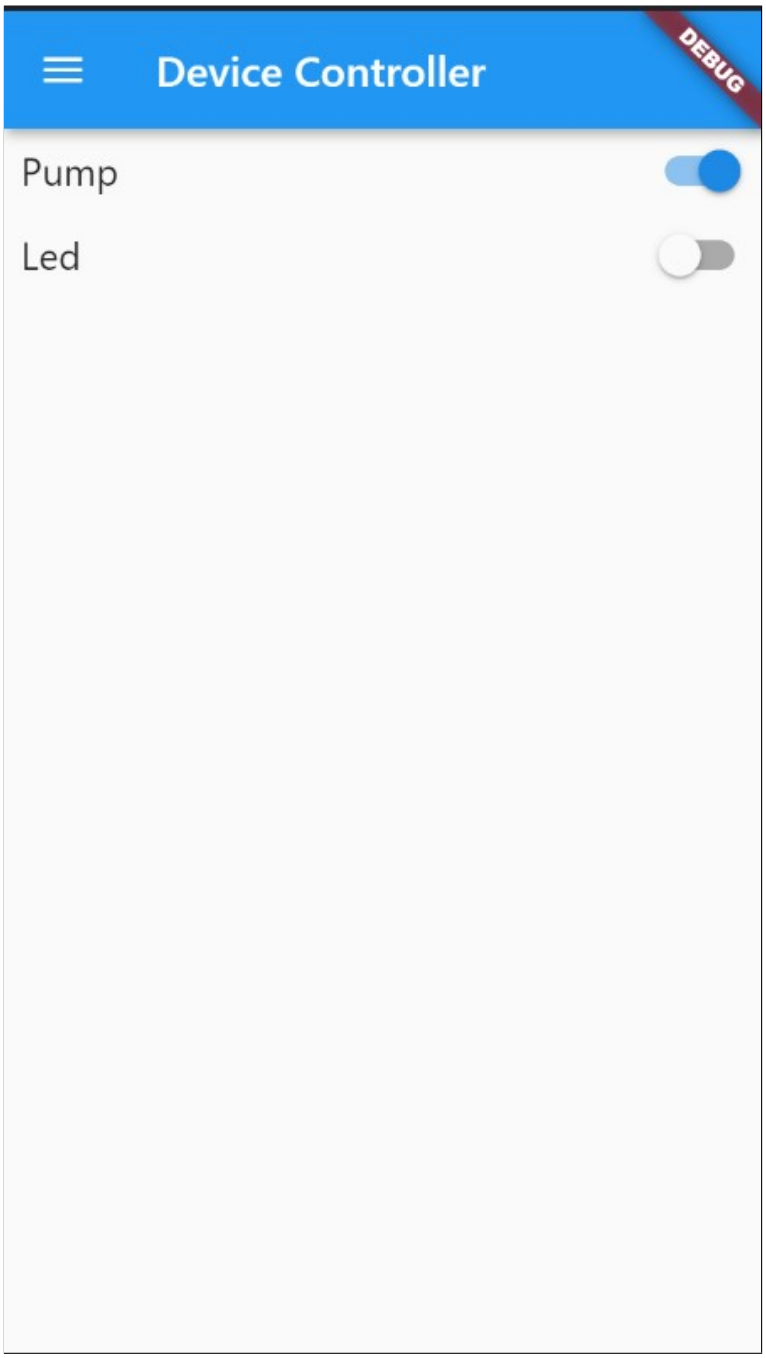


Giao diện chính(staticstic) chứa các biểu đồ theo thời gian của các yếu tố như: nồng độ Co2, Độ ẩm, Độ ẩm đất, nhiệt độ


Statistics
DEBUG
Soil Moisture
Soil Moisture (%)
0
2
4
6
8
0
5
10
15
Time (s)
Temperature
Temperature (Celcius)
22
23
24
25
26
27
28
29
0
5
10
15
Time (s)
Light
6
4

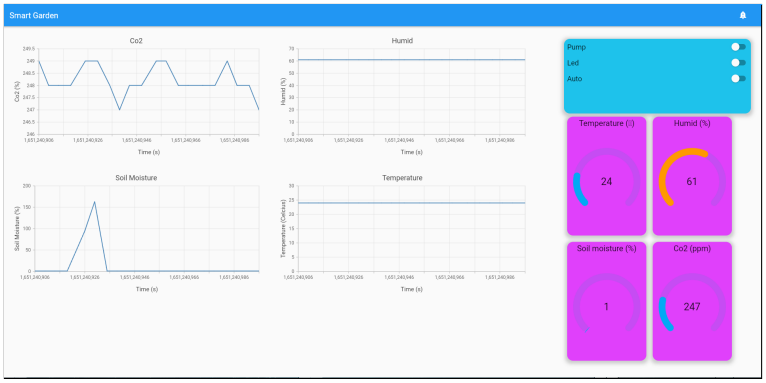


Giạo diện khi ấn vào nút bên góc trái. Ta có thể chọn "Device controller" để vào giao diện điều chỉnh các thiết bị



Giao diện công tắc

Host web: https://smartgarden-b8fcb.web.app/#/
Giao diện Web của App



6 Tổng kết

6.1 Tính hiệu quả trong mô hình

Mô hình demo được hiện thực thông qua các mạch Microbit để thể hiện đại diện khả năng của hệ thống cho thấy trên qui mô tạm thời hệ thống cung cấp được các tính năng cần có của 1 khu vườn thông minh cơ bản. Việc liên kết các online database của nhóm đã giúp tối ưu hóa mô hình iot thông qua việc tạo ra 2 db giống nhau nhằm tránh việc xuất hiện lỗi về kết nối đến 1 trong 2 kho lưu nhằm đảm bảo tính liên tục của việc lưu trữ và tương tác với các thiết bị trong trường hợp xuất hiện lỗi về kết nối với 1 trong 2 server. Các vấn đề trong hiện thực mạch microbit cũng như việc thiết kế trong môi trường thực tế cũng gặp không ít khó khắn, cần tới sự hỗ trợ từ phía bên giáo viên cũng như các nhóm khác. Nhóm nhận thấy sản phẩm thực hiện được đã đạt được những yêu cầu tối thiểu về iot, trở thành 1 trải nghiệm, kinh nghiệm thực tiễn cho tiếp tục phát huy kiến thức về iot và các công nghệ khác mai sau.

6.2 Đề xuất mở rộng mô hình

Dựa trên mô hình hiện tại của nhóm, hệ thống hiện chỉ đang xây dựng 1 môi trường xử lý cho 1 bộ iot của 1 khu vườn (1 cây) cụ thể. Vì vậy nhóm đề cử 2 hướng phát triển sau:

- Phát triển theo hướng làm hệ thống quản lý trung gian cho các mô hình iot và ứng dụng liên kết
- Phát triển theo hướng tối ưu việc nuôi dưỡng, chăm sóc cây trồng cho khu vườn hiện tại

6.2.1 Hệ thống quản lý

Do gateway và app được hiện thực theo hướng cho phép thay đổi database cần liên kết và lưu trữ ở dạng file rời. Như vậy hoàn toàn có thể phát triển theo hướng đa người dùng với hệ thống iot cụ thể liên kết qua server xử lý hiện tại của nhóm để nhanh chóng có được app liên kết với các thiết bị thông qua nền tảng sẵn có của nhóm. Như vậy sẽ cần thêm lưu trữ và tăng tính bảo

mật các thông tin về user, các thông tin thiết yếu khác như thông tin database cũng như quyền truy xuất.

6.2.2 Chăm sóc cây trồng

Hiện thực hệ thống tự động chăm sóc cây trồng thông qua theo dõi trạng thái cụ thể của môi trường. Gán các dữ liệu cụ thể cho cây, không chỉ môi trường. Có thể hiện thực AI để theo dõi cũng như thực hiện đề xuất để cải thiện tình trạng cây trồng.

Tài liệu

[1] Flutter documentation for building stateful widget,https://docs.flutter.dev/development/data-and-backend

[2] Real time chart for flutter, https://www.youtube.com/watch?v=t8Tyj9Sw_iM

[3] Flutter turorial app, https://docs.flutter.dev/get-started/codelab

[4] GiaoTrinh - IoT - Python - Final_2

[5] GiaoTrinh - IoT - Python - Final