



**GIẢI PHÁP BÃI GIỮ XE THÔNG MINH
SỬ DỤNG NHẬN DIỆN RFID TẦM XA (UHF)**

Thực hiện: Đồng Quang Quyền
Trịnh Trần Trung
Nguyễn Quang Anh

TP. Hồ Chí Minh, tháng 9 năm 2022

MỤC LỤC

MỤC LỤC	i
MỤC LỤC HÌNH.....	iii
CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI.....	1
1.1. Giới thiệu chung	1
1.2. Thực trạng.....	1
1.3. Mục tiêu	1
1.4. Phương pháp thực hiện.....	2
1.4.1. Đối tượng.....	2
1.4.2. Phạm vi thực hiện.....	2
CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG	3
2.1. Sơ đồ hệ thống.....	3
2.2. Sơ đồ hoạt động của hệ thống	4
2.2.1. Hoạt động quét UHF trên bộ UHF Reader.....	6
2.2.2. Truyền nhận dữ liệu trên các board LoRa	8
2.2.3. Hoạt động nhận và lưu dữ liệu lên Azure Storage trên Raspberry PI	13
2.2.4. Chế độ ngủ.....	16
2.3. Cấu trúc các gói tin truyền nhận.....	17
2.3.1. Gói tin truyền nhận thông qua giao thức LoRa	17
2.3.2. Gói tin lưu trữ lên Server.....	17
2.4. Các tần số và thông số hoạt động trên các thiết bị	18
2.4.1. Các thông số của LoRa.....	18
2.4.2. Anten và tần số quét của UHF.....	19
CHƯƠNG 3. HIỆN THỰC VÀ KẾT QUẢ.....	20
3.1. Hiện thực hệ thống	20
3.1.1. Kết nối hệ thống (nối dây).....	20
3.1.2. Gửi gói tin từ Board UHF đến LoRa Sender.....	21
3.1.3. Gửi gói tin đến LoRa Receiver.....	21
3.1.4. Trường hợp mất gói tin LoRa.....	22
3.1.5. Gửi gói tin đến PI	23
3.1.6. Gửi gói tin lên Server	24

3.2. Kết quả.....	24
CHƯƠNG 4. NHẬN XÉT VÀ HƯỚNG PHÁT TRIỂN.....	26
4.1. Nhận xét.....	26
4.1.1. Ưu điểm	26
4.1.2. Nhược điểm	26
4.2. Hướng phát triển.....	26
TÀI LIỆU THAM KHẢO.....	27

MỤC LỤC HÌNH

Hình 2-1 Sơ đồ tổng quát kiến trúc của hệ thống.....	3
Hình 2-2 Lưu đồ quy trình hoạt động của hệ thống	5
Hình 2-3 Lưu đồ hoạt động chi tiết của UHF Reader	6
Hình 2-4 Lưu đồ hoạt động truyền dữ liệu UHF Reader sang LoRa Sender	7
Hình 2-5 Sơ đồ minh họa các node truyền nhận trong LoRa.....	8
Hình 2-6 Lưu đồ hoạt động của LoRa Sender.....	9
Hình 2-7 Lưu đồ hoạt động của LoRa Receiver.....	10
Hình 2-8 Cách hoạt động của ACK trong hệ thống	11
Hình 2-9 Lưu đồ hoạt động tính toán checksum	12
Hình 2-10 Lưu đồ hoạt động của Raspberry PI.....	13
Hình 2-11 Lưu đồ chi tiết hoạt động tạo container	14
Hình 2-12 Lưu đồ chi tiết hoạt động tạo local folder lưu dữ liệu đệm.....	14
Hình 2-13 Cách chuyển đổi nội dung gói tin tại Raspberry PI	15
Hình 2-14 Lưu đồ hoạt động tạo file JSON chứa dữ liệu.....	15
Hình 2-15 Lưu đồ hoạt động dùng cảm biến chạm làm interrupt	16
Hình 2-16 Bảng các thông số hoạt động của LoRa với các SF khác nhau.....	18
Hình 3-1 Sơ đồ kết nối chân UART của ESP ESP32 và M6e-Nano	20
Hình 3-2 Sơ đồ kết nối chân UART của UHF Reader và UCA Board	20
Hình 3-3 Sơ đồ kết nối UART của UCA Board (LoRa Receiver) và Raspberry PI....	21
Hình 3-4 Mẫu gói tin truyền đi ở LoRa Sender.....	21
Hình 3-5 Hàm thực hiện nhận gói tin LoRa ở LoRa Receiver.....	22
Hình 3-6 Kết quả nhận gói tin LoRa	22
Hình 3-7 Kết quả hiển thị trong trường hợp mất gói tin	22
Hình 3-8 Hàm xử lý tính toán checksum và gửi lại cho LoRa Sender.....	23
Hình 3-9 Kết quả tính toán checksum và thông báo gửi ACK.....	23
Hình 3-10 Hàm truyền UART có tin cậy đến Raspberry PI	23
Hình 3-11 Kết quả nhận gói tin UART trên Raspberry PI.....	24
Hình 3-12 File JSON được lưu trữ trong Azure Blob Storage.....	24
Hình 3-13 Nội dung file JSON được lưu trữ trên Azure Blob Storage.....	24

CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI

1.1. Giới thiệu chung

Ngày nay nhu cầu đi lại với các phương tiện giao thông như xe máy, xe mô tô, ô tô và các phương tiện khác có xu hướng tăng cao dẫn đến các hoạt động bãi đỗ xe trở nên quá tải, khó kiểm soát hơn. Do đó bài toán đặt ra cho các chủ quản lý bãi đỗ xe là làm thế nào để hệ thống giữ xe của mình hoạt động nhanh chóng hơn, kiểm soát ra vào bảo mật hơn, có thể sử dụng các thiết bị công nghệ nào thay thế sức lao động của con người. Hay nói cách khác là tìm giải pháp quản lý bãi giữ xe nào hiệu quả cho thực trạng này.

Giải pháp quản lý bãi giữ xe thông minh bằng công nghệ nhận diện thẻ RFID là một ứng dụng phổ biến đang ngày càng được sử dụng rộng rãi trong cuộc sống. Là giải pháp phần cứng nên nhận diện thẻ RFID có tính hoạt động ổn định, ít những điều kiện nhiễu so với những giải pháp phần mềm.

1.2. Thực trạng

Hiện nay, công nghệ RFID được sử dụng trong nhiều hệ thống quản lý bãi đỗ xe tự động. Việc sử dụng thẻ từ để kích hoạt hệ thống bãi đỗ xe sẽ giúp hạn chế tối đa hiện tượng ùn tắc tại cổng các bãi đỗ xe. Điều này khiến cho người sử dụng bãi đỗ xe thoải mái hơn đồng thời giảm bớt một phần chi phí nhân công cho các bãi đỗ xe. Nhưng vẫn còn mặt hạn chế là phải tốn một khoảng thời gian trong việc phải dừng lại để quét thẻ và đợi mở rào chắn barie. Vì vậy nhóm chúng tôi thực hiện nghiên cứu giải pháp để có thể giải bài toán bỏ qua thời gian đợi quét thẻ và thay vào đó là sử dụng luôn thẻ Tag được dán sẵn trên xe để thời gian vô bãi nhanh hơn.

Đối với bài toán được đưa ra, ứng dụng để tạo nên một bãi đỗ xe thông minh, ta có thể tận dụng được các thẻ xe đã được phát hành và dán sẵn trên các phương tiện xe hơi lưu thông trên khắp cả nước vì mục đích thu phí không dừng trước đó. Ngoài ra, bộ phát hiện thẻ RFID tầm xa còn có khả năng quét thẻ chính xác và nhanh chóng trong phạm vi cho phép.

Với những tính chất đặc trưng như trên, bộ phát hiện thẻ RFID tầm xa rất phù hợp đối với hệ thống bãi giữ xe thông minh với các vai trò trong hệ thống như:

- Xác định được khoảng cách của xe so với điểm đỗ xe.
- Quản lý chính xác thông tin phương tiện đang di chuyển vào điểm đỗ xe.

1.3. Mục tiêu

Mục tiêu của đề tài này nhóm xây dựng một hệ thống quản lý bãi giữ xe thông minh, cải tiến hơn hệ thống bãi giữ xe đã có với giá thành rẻ hơn so với thị trường. Sử dụng RFID (UHF) để nhận diện ra được xe với những thẻ Tag được dán sẵn trên các phương tiện xe hơi lưu thông trên khắp cả nước vì mục đích dùng cho thu phí không dừng trước đó.

Người dùng không cần phải tương tác với hệ thống nhiều như sử dụng quét thẻ RFID như trước và tốn ít thời gian cho việc vào bãi xe.

Hệ thống được thiết kế nhỏ gọn, chi phí cho thiết bị và lắp đặt rẻ hơn so với thị trường để có thể hướng đến xây dựng một hệ thống quy mô lớn của bãi giữ xe cho sân bay, trung tâm thương mại, ... và cũng có thể áp dụng cho quy mô nhỏ như hộ gia đình để dễ dàng cho việc quản lý và giảm thiểu các tình trạng dừng đỗ không đúng quy định.

1.4. Phương pháp thực hiện

1.4.1. Đối tượng

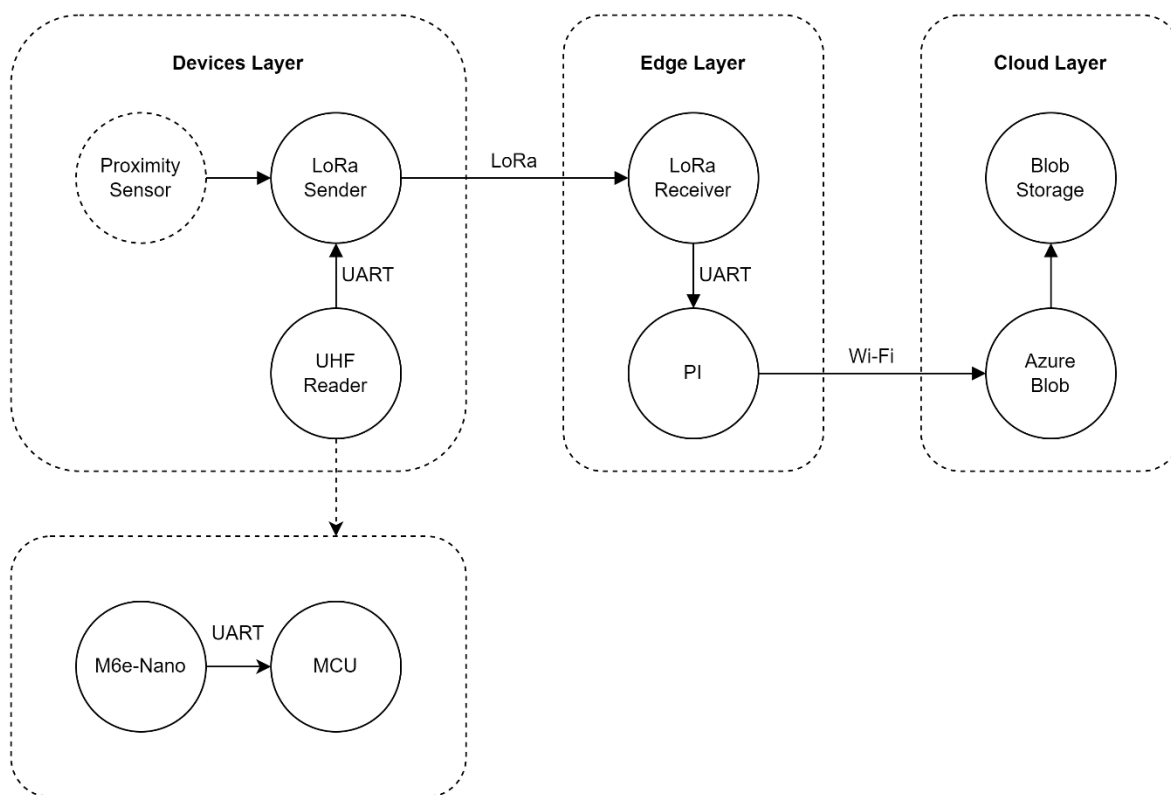
- Raspberry Pi 4 model B.
- Board UHF reader (M6E Nano kết hợp với ESP32) + thư viện nhận diện và đọc được Tag xe + Tag xe (Tag được dán sẵn trên xe).
- Ăng ten RT-CP3-QL Lacuna edition.
- Board UCA_Education (protocol Lora) version 3.9.
- Cảm biến Ultrasonic.
- Azure Blob Storage.

1.4.2. Phạm vi thực hiện

- Tìm hiểu và nghiên cứu cách để đọc được 2 loại Tag xe hiện tại có sẵn trên thị trường bằng board UHF.
- Nghiên cứu sử dụng cảm biến để tiết kiệm năng lượng cho hệ thống.
- Tìm hiểu phương thức giao tiếp Lora để truyền và nhận dữ liệu.
- Sử dụng UART để truyền nhận dữ liệu từ UHF – UCA và UCA – Raspberry Pi.
- Xây dựng và cấu trúc dữ liệu để đẩy lên Sever bằng Azure Blob Storage.

CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

2.1. Sơ đồ hệ thống



Hình 2-1 Sơ đồ tổng quát kiến trúc của hệ thống

Hệ thống được thiết kế dựa trên kiến trúc cơ bản của một hệ thống IoT có đầy đủ các thành phần. Các tầng thiết bị được phân chia rõ ràng như sơ đồ trên:

- Tầng thiết bị (Devices Layer) đại diện cho nhiều node con, mỗi node đóng vai trò thu thập dữ liệu và truyền dữ liệu sang tầng biên (Edge Layer) thông qua giao thức LoRa.
- Tầng biên sẽ thu thập dữ liệu, xử lý cục bộ và gửi dữ liệu đến tầng Cloud (Cloud Layer) để lưu trữ dữ liệu.

Mô tả chi tiết các thành phần được sử dụng trong sơ đồ hệ thống nói trên:

- LoRa Sender và LoRa Receiver là RFThings UCA board có tích hợp LoRa có chức năng truyền và nhận message đọc được từ UHF reader.
- UHF Reader là một board quét RFID tần số cao đọc được Tag dán trên xe, UHF được tạo thành từ một chip M6e-Nano Reader kết hợp ESP32 giao tiếp thông qua UART2.
- Raspberry PI được sử dụng trong hệ thống là Raspberry PI 4 model B là một máy tính nhúng có tích hợp Wi-Fi thích hợp trong việc nhận dữ liệu từ UART và gửi dữ liệu đến Cloud.

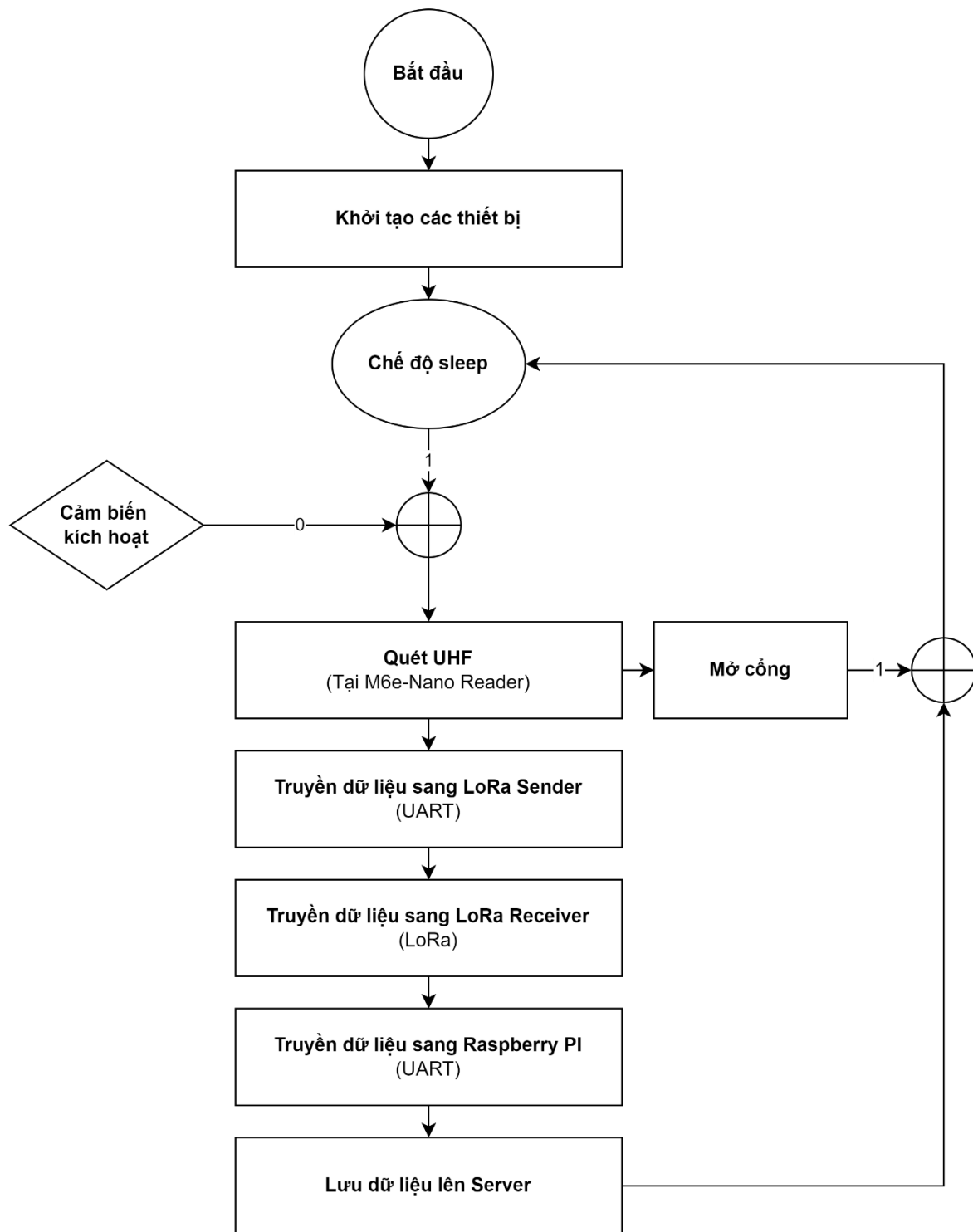
- Cloud được sử dụng là Azure Blob Storage chứa các container con dùng để lưu trữ dữ liệu dưới dạng file JSON hoặc table và đẩy lên server từ PI theo từng ngày cụ thể để quản lý, kiểm soát một cách dễ dàng cho hệ thống.

2.2. Sơ đồ hoạt động của hệ thống

Quy trình hoạt động của hệ thống:

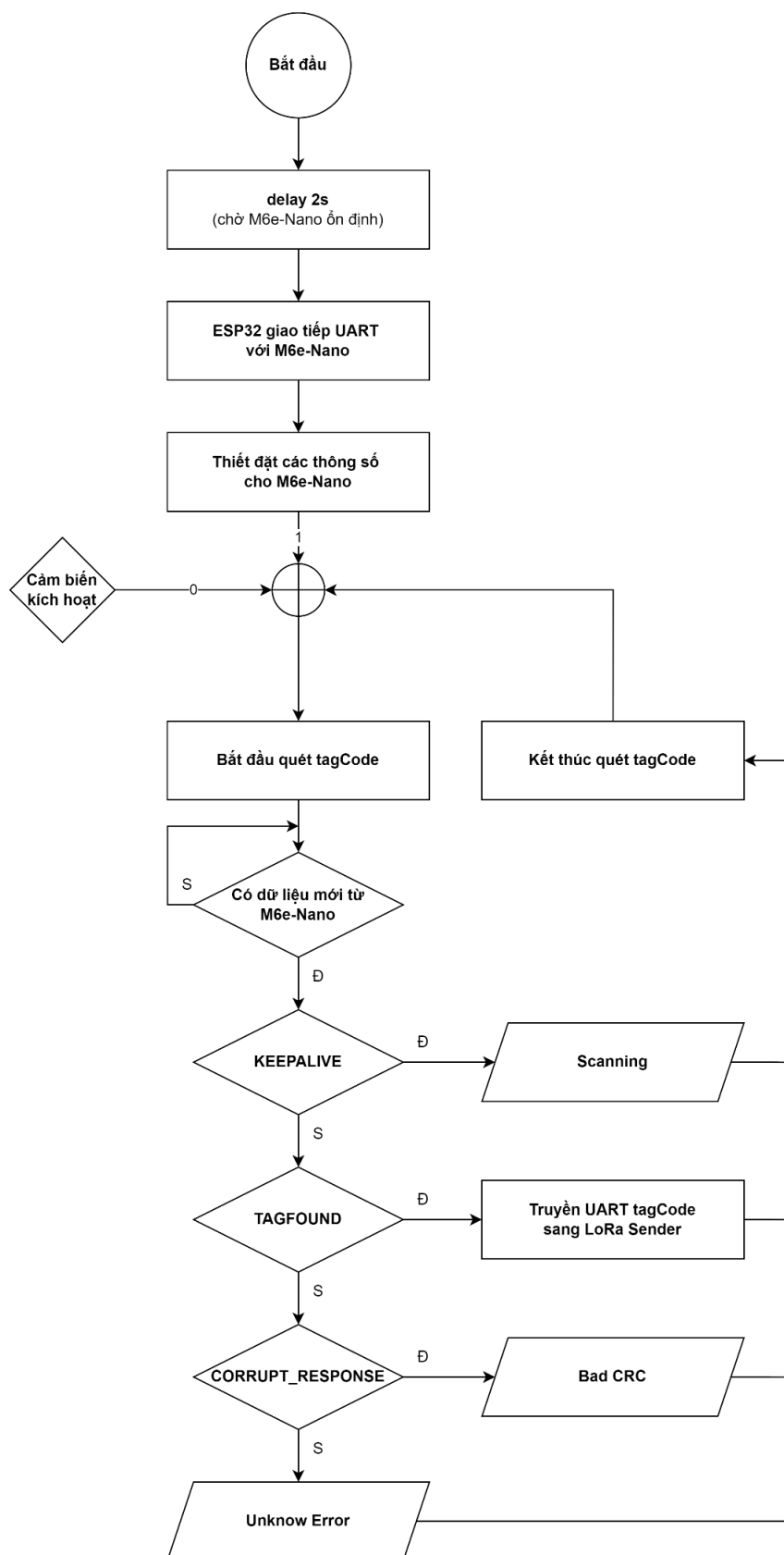
- Bắt đầu được cấp nguồn, hệ thống được khởi động và các chương trình bắt đầu hoạt động.
- Quá trình bắt đầu quét thực hiện nhưng tuy nhiên trong quá trình module UHF không nhận diện được thẻ Tag nào và cảm biến không thấy được vật cản nào đến thì hệ thống sẽ chuyển sang trạng thái sleep để tiết kiệm năng lượng cho hệ thống.
- Khi phát hiện vật thể di chuyển đến từ cảm biến, đầu đọc bắt đầu nhận diện và đọc thẻ. Dữ liệu sẽ được đóng gói và được gửi lên server và cổng bắt đầu mở. Chu kỳ hoạt động lặp đi lặp lại cho đến khi ngắt nguồn.
- Sau những lần truyền và nhận dữ liệu, hệ thống sẽ thực hiện xác thực gói tin để tăng độ tin cậy cũng như tăng khả năng hoạt động ổn định của hệ thống.

Sơ đồ hoạt động chi tiết của hệ thống:



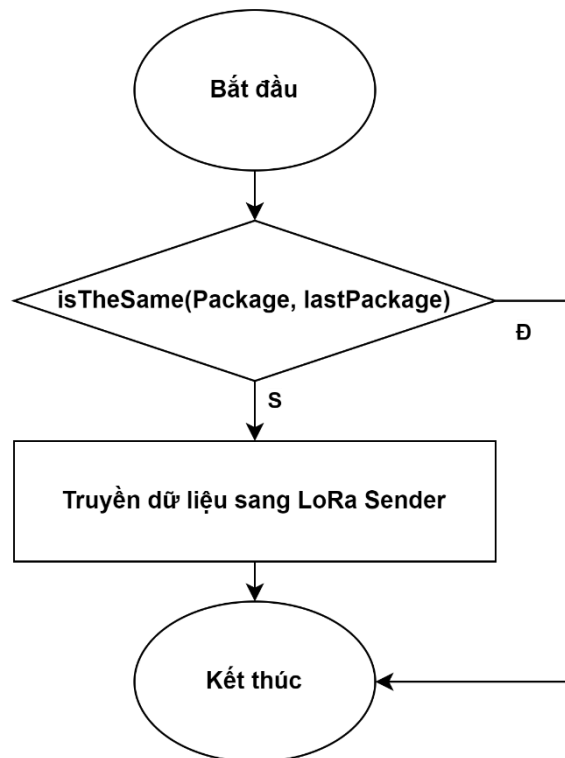
Hình 2-2 Lưu đồ quy trình hoạt động của hệ thống

2.2.1. Hoạt động quét UHF trên bộ UHF Reader



Hình 2-3 Lưu đồ hoạt động chi tiết của UHF Reader

- Đầu tiên, M6e-Nano được cấp nguồn bằng USB do đó, nó cần thời gian 2 giây để ổn định sau khi bật nguồn.
- Tiếp đến, ESP32 sẽ thực hiện giao tiếp với M6e-Nano thông qua UART2 của ESP32 với baudrate là 115200 và thiết đặt các thông số cho M6e-Nano khi đã có kết nối thành công, ngược lại nếu không phát hiện ra M6e-Nano thì báo lỗi không thể kết nối.
- Khi cảm biến phát hiện có xe vào bãi thì có interrupt kích hoạt UHF Reader bắt đầu quét tag. ESP32 bắt đầu nhận dữ liệu liên tục từ M6e-Nano, tùy vào từng responseType của M6e-Nano mà ESP32 có các xử lý phù hợp. Đặc biệt, nếu nhận được các tagCode giống với tagCode đã nhận ngay liền trước nó từ ESP32 tiến hành loại bỏ và đưa M6e-Nano về chế độ stop Reading (hay sleep mode) để tiết kiệm năng lượng và chờ đến lần cảm biến phát hiện có xe tiếp theo.
- Truyền UART tagCode từ UHF Reader sang LoRa Sender:

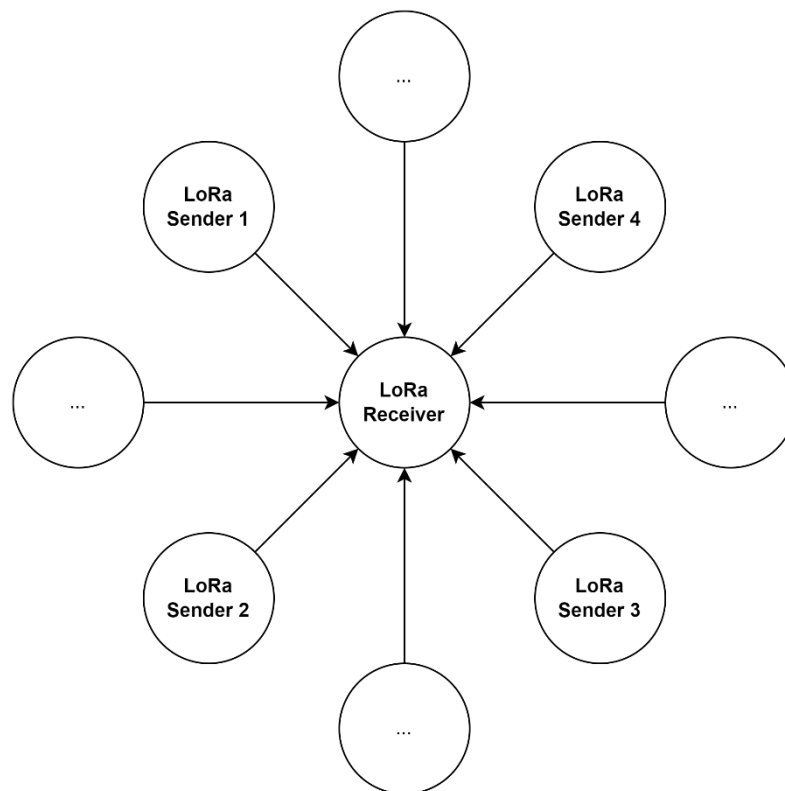


Hình 2-4 Lưu đồ hoạt động truyền dữ liệu UHF Reader sang LoRa Sender

- Rõ hơn về quá trình truyền dữ liệu từ UHF Reader sang LoRa Sender, đầu tiên thực hiện việc so sánh 2 gói tin là gói tin vừa nhận được và gói tin trước đó.
- Nếu chúng giống nhau thì ta kết thúc quá trình truyền dữ liệu sang LoRa Sender, bởi vì nếu giống nhau thì có nghĩa là gói tin này đã được truyền đi và không cần truyền thêm một lần nữa.
- Nếu 2 gói tin khác nhau thì ta thực hiện việc truyền dữ liệu sang LoRa Sender và kết thúc quá trình này.

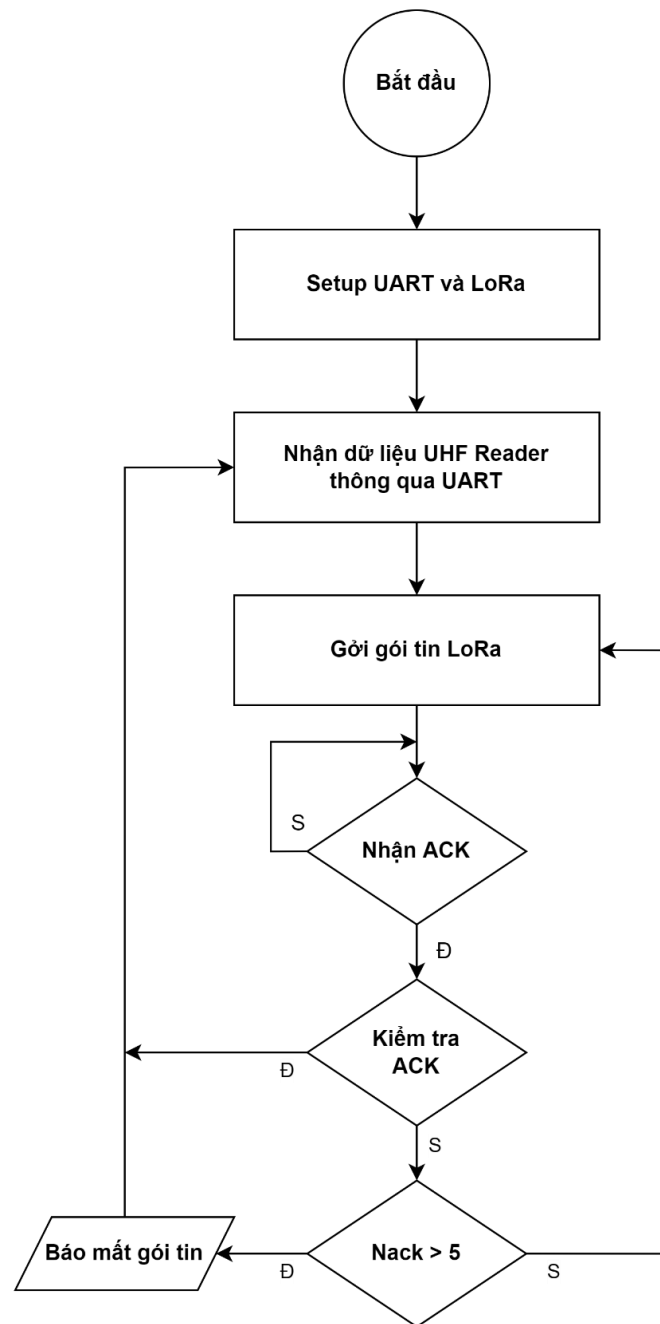
2.2.2. Truyền nhận dữ liệu trên các board LoRa

- Việc gửi dữ liệu thông qua sóng LoRa giúp việc truyền nhận tiết kiệm năng lượng và truyền được đi xa với gói tin nhỏ. Tuy nhiên nhược điểm của việc truyền thông đi xa là rất dễ bị nhận gói tin lỗi hoặc bị bỏ lỡ gói tin. Do đó, phương án ở đây là ta sẽ sử dụng tín hiệu báo nhận (ACK) và ngược lại là (NACK) để xác định gói tin đã mất hay chưa. Bên cạnh đó, gói tin còn được tăng độ tin cậy bằng cách tính toán Checksum trong gói tin ACK.
- Theo đó, ta có nhiều node truyền và 1 node nhận. Các node truyền được gọi tiêu biểu với tên là LoRa Sender và node nhận là LoRa Receiver:



Hình 2-5 Sơ đồ minh họa các node truyền nhận trong LoRa

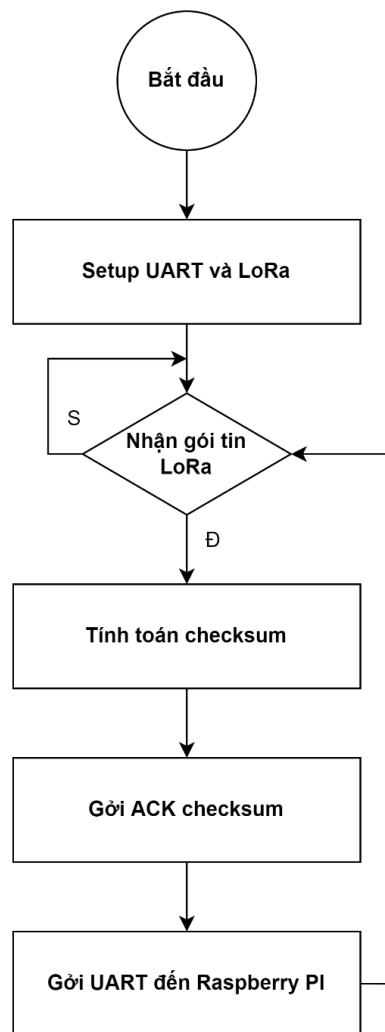
- Lấy ví dụ điển hình về một LoRa Sender và một LoRa Receiver, với LoRa Sender, hoạt động của board như sau:



Hình 2-6 Lưu đồ hoạt động của LoRa Sender

- Đầu tiên, ta cần thiết lập các thông số ban đầu cho các kết nối uart và LoRa. Cụ thể, ta sẽ đặt baudrate để kết nối với board UHF Reader là 9600 và các thông số để thực hiện truyền thông sóng LoRa sẽ như sau: tần số kết nối là 915E6, Spreading Factors là 9 và Bandwidth là 125E3.
- Tiếp theo đó, với baudrate đã thiết lập, LoRa Sender sẽ kết nối thông qua uart nhận được mã thẻ từ UHF Reader. Sau khi có được mã thẻ, tại đây LoRa Sender sẽ tiến hành thêm ID của thiết bị vào gói tin. Lúc này, gói tin đã có được mã thẻ và ID thiết bị. Cuối cùng là tiến hành gửi gói tin thông qua sóng LoRa.

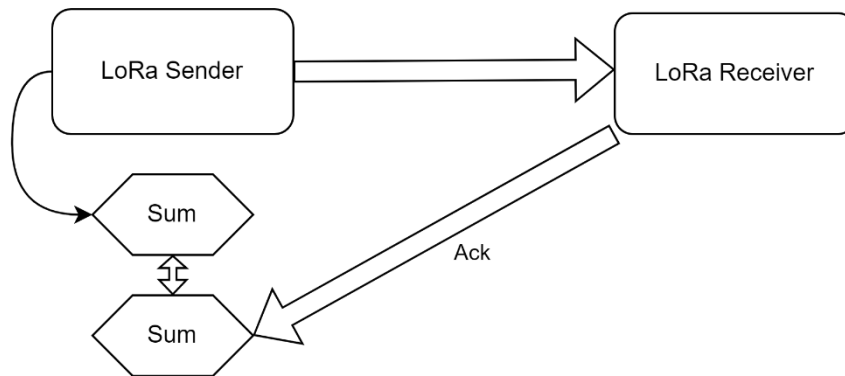
- Sau khi đã gửi gói tin, LoRa Sender sẽ thực hiện một vòng lặp tại chỗ và chờ gói tin ACK được gửi về từ LoRa Receiver. Nếu nhận được gói tin ACK thì sẽ thực hiện việc kiểm tra bằng cách so sánh giá trị gói tin ACK này với giá trị gói tin đã được gửi. Việc tính toán giá trị gói tin như thế nào sẽ được trình bày rõ hơn ở phần sau.
 - Trong trường hợp xác nhận là gói tin đã đúng thì ta quay lại nhận bước nhận dữ liệu từ UHF Reader. Ngược lại thì ta quay lại bước gửi gói tin, và nếu việc mất gói tin vượt quá năm lần thì sẽ thông báo là mất gói tin và quay lại bước nhận dữ liệu từ UHF Reader.
- Với LoRa Receiver, hoạt động của board như sau:



Hình 2-7 Lưu đồ hoạt động của LoRa Receiver

- Đầu tiên sẽ là thiết lập các thông số để kết nối với Raspberry thông qua Uart và LoRa Sender thông qua sóng LoRa. Cụ thể, ta sẽ đặt baudrate để kết nối với Raspberry là 115200 và các thông số để thực hiện truyền thông sóng LoRa sẽ giống như bên LoRa Sender: tần số kết nối là 915E6, Spreading Factors là 9 và Bandwidth là 125E3.

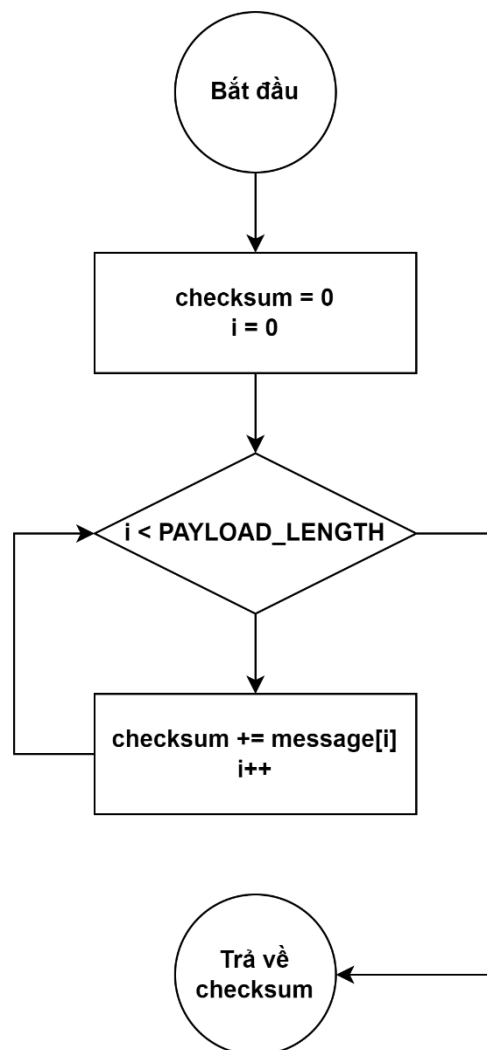
- Tiếp đó, LoRa Receiver sẽ thực hiện một vòng lặp tại chỗ và chờ gói tin được gửi về từ LoRa Receiver. Sau khi nhận được gói tin, sẽ tiến hành tính toán giá trị gói tin và gửi giá trị này ngược về LoRa Sender (giá trị này được gọi là gói tin ACK). Sau đó, LoRa Receiver sẽ gửi gói tin qua cho Raspberry thông qua uart và quay về bước chờ nhận gói tin.
- Chi tiết sử dụng ACK được thể hiện như hình bên dưới:



Hình 2-8 Cách hoạt động của ACK trong hệ thống

- LoRa Sender sẽ gửi gói tin đến LoRa Receiver, tại đây gói tin sẽ được tính tổng giá trị và gửi lại cho LoRa sender.
- Về phía LoRa Sender sẽ tiến hành tính tổng giá trị của gói tin được gửi đi và so sánh với giá trị của gói tin nhận được từ LoRa Receiver. Nếu chúng có giá trị giống nhau thì ta xác nhận rằng gói tin được gửi tại LoRa Sender đã được nhận một cách toàn vẹn tại LoRa Receiver sẽ bằng một và ngược lại thì gói tin được cho là đã bị mất hoặc không còn toàn vẹn.

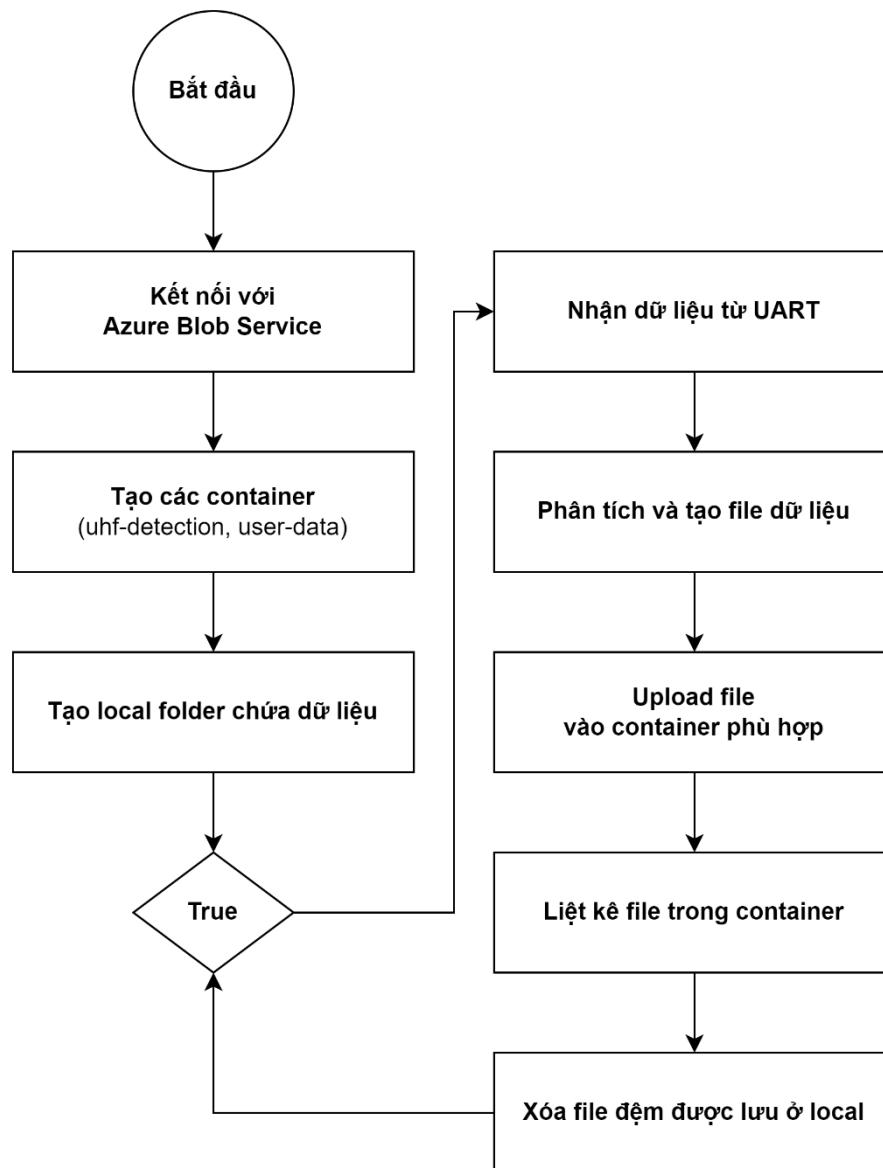
- Trong quá trình truyền nhận các gói tin LoRa, việc tính toán checksum (giá trị của gói tin) được thực hiện ở cả LoRa Sender và LoRa Receiver. Chi tiết hoạt động tính toán checksum được thể hiện ở sơ đồ sau:



Hình 2-9 Lưu đồ hoạt động tính toán checksum

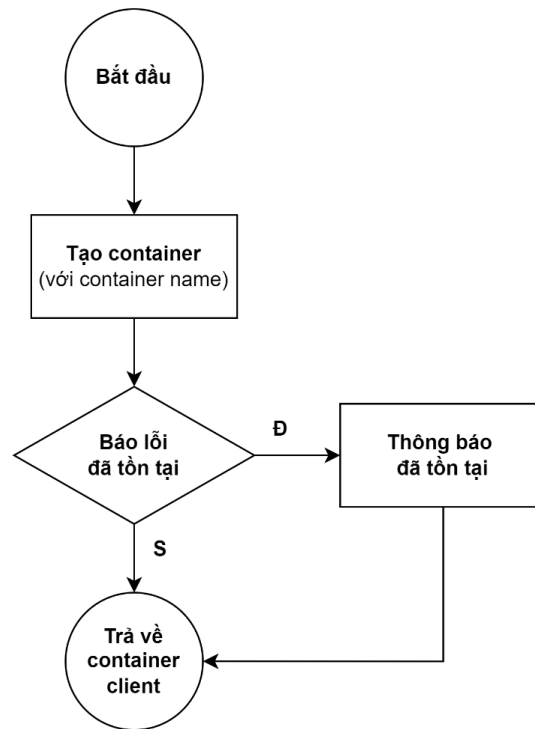
- Ý tưởng của thuật toán này sẽ là tính tổng tất cả các giá trị của phần tử trong gói tin. Để làm được điều này, ta phải quy đổi giá trị Char và Byte sang một kiểu dữ liệu đồng bộ để so sánh, ở đây ta chọn kiểu Integer.
- Đầu tiên, ta sẽ khởi tạo các giá trị checksum = 0. Tiếp đó ta thực hiện vòng lặp cho đến hết gói tin. Cuối cùng ta thu được tổng giá trị của gói tin được lưu vào checksum. Đây là giá trị của gói tin để ta có thể so sánh với nhau.

2.2.3. Hoạt động nhận và lưu dữ liệu lên Azure Storage trên Raspberry PI



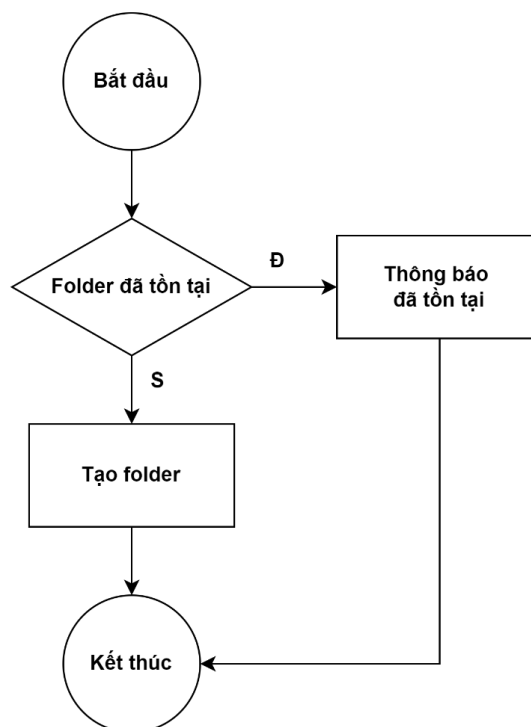
Hình 2-10 Lưu đồ hoạt động của Raspberry PI

- Để lưu trữ dữ liệu lên Azure Storage, server cần kết nối với Azure Storage Service, sau khi kết nối, ta sẽ nhận được một client để thao tác trên Storage Service.
- Tiếp đến là tạo các blob container để lưu trữ dữ liệu. Khi thực hiện tạo blob container, ta sẽ nhận lại được một client để thao tác trên container. Chi tiết hoạt động tạo container như sau:



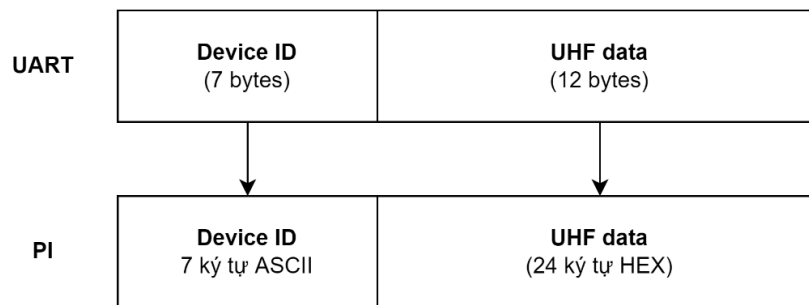
Hình 2-11 Lưu đồ chi tiết hoạt động tạo container

- Sau khi đã tạo container, ta cần có một vị trí để lưu những dữ liệu đệm trước khi được lưu trữ lên Server. Khi đó, ta gọi hàm tạo local folder. Chi tiết hoạt động của hàm như sau:



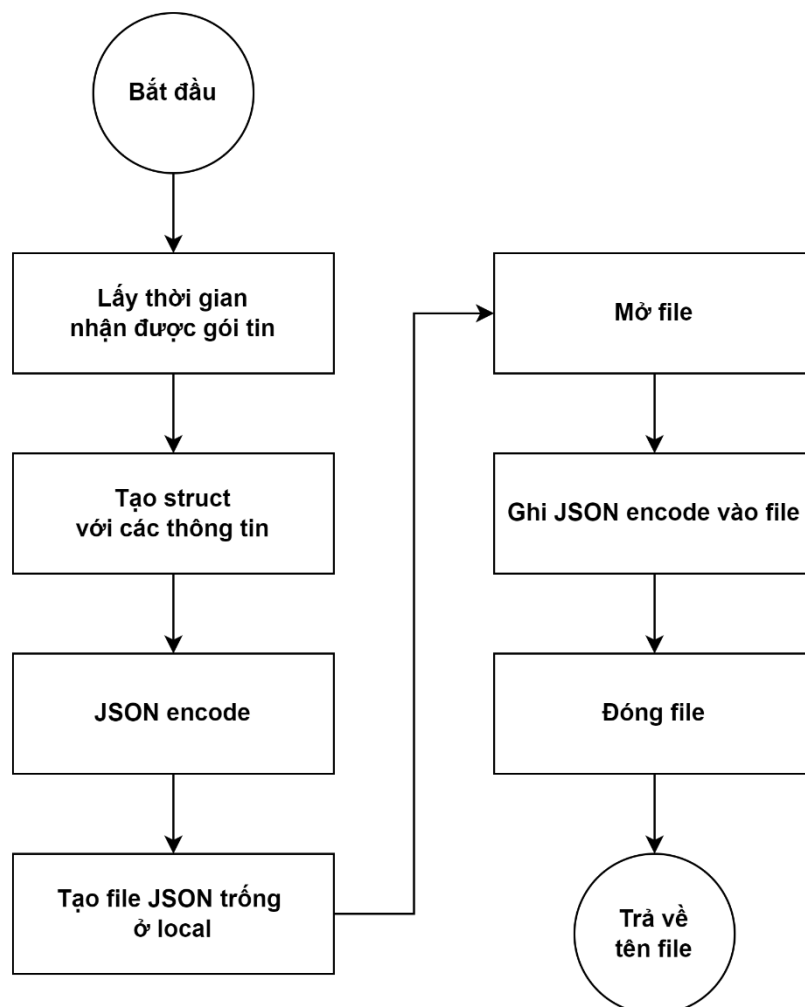
Hình 2-12 Lưu đồ chi tiết hoạt động tạo local folder lưu dữ liệu đệm

- Thư mục local vừa tạo được dùng để lưu trữ file json trước khi upload lên server.
- Tiếp đó, ta tiến hành nhận dữ liệu từ UART liên tục, với mỗi lần nhận UART, ta sẽ phân giải gói tin để nhận được dữ liệu như sau:



Hình 2-13 Cách chuyển đổi nội dung gói tin tại Raspberry PI

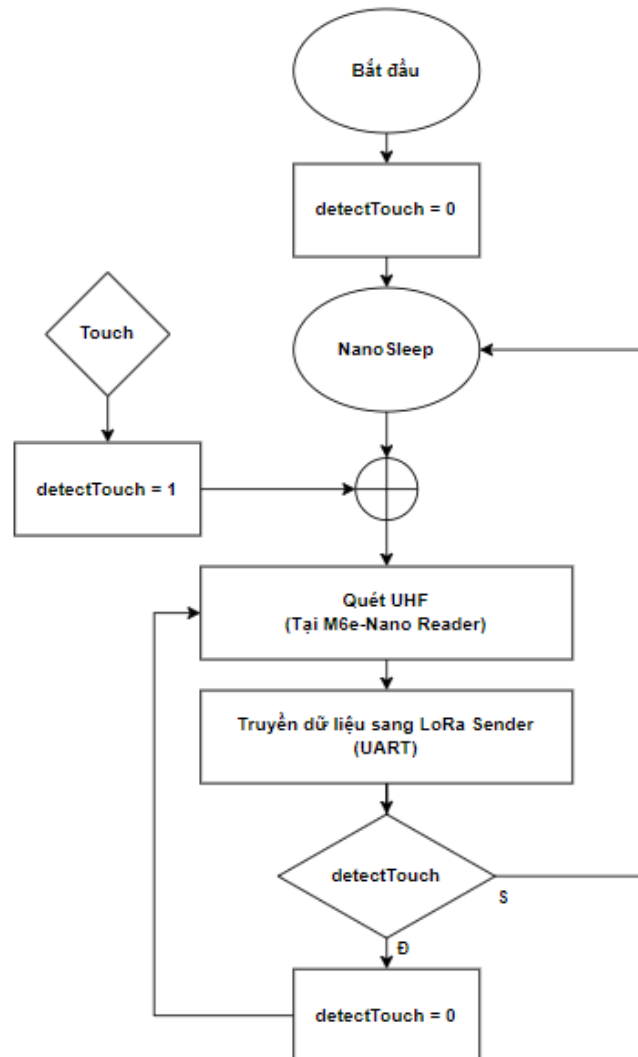
- Sau khi đã nhận được Device ID kèm theo UHF data, ta thực hiện lấy thêm thông tin thời gian và cấu hình gói tin hoàn chỉnh để lưu lên Azure Storage. Chi tiết hoạt động như sau:



Hình 2-14 Lưu đồ hoạt động tạo file JSON chứa dữ liệu

- Sau khi có được file json, ta lưu trữ nó lên Blob container với hàm upload package. Thêm vào đó, để kiểm tra file đã được upload lên container hay chưa ta có thể liệt kê các blobs có trong container.
- Để giải phóng bộ nhớ và quản lý dễ hơn, ta xóa file json ở local được lưu trữ trong local folder và tiếp tục quay lại nhận dữ liệu từ UART.

2.2.4. Chế độ ngủ



Hình 2-15 Lưu đồ hoạt động dùng cảm biến chạm làm interrupt

- Trong tiến độ hiện tại của đề tài, ta vẫn chưa hiện thực được cảm biến dùng để phát hiện xe. Do đó ta sử dụng cảm biến chạm trên board UHF Reader với mục đích mô phỏng cảm biến khi xe chạy đến.
- Đầu tiên giá trị detectTouch sẽ được gán giá trị bằng 0 và lúc này đầu đọc sẽ đi vào chế độ sleep.
- Nếu cảm biến chạm được kích hoạt, giá trị detectTouch sẽ bằng 1 và thực hiện việc quét mã tag và truyền dữ liệu sang LoRa Sender.

- Nếu ta không chạm vào cảm biến chạm thì detectTouch sẽ bằng 0. Do đó, bước tiếp theo ta sẽ so sánh xem detectTouch đã bằng 0 hay chưa. Nếu chưa thì ta sẽ gán giá trị cho detectTouch bằng 0 và quay lại bước quét UHF. Ngược lại, nếu detectTouch bằng 0 thì ta sẽ quay lại chế độ sleep.

2.3. Cấu trúc các gói tin truyền nhận

2.3.1. Gói tin truyền nhận thông qua giao thức LoRa

Gói tin được thiết kế bao gồm 2 phần chính bao gồm header và payload. Cụ thể hơn phần header sẽ bao gồm các thông tin điều khiển như checksum, phiên bản giao thức. Payload là phần sẽ chứa nội dung ta cần truyền tải. Phần payload của gói tin được truyền trong hệ thống này được thiết kế như hình phía bên dưới.

DeviceID (7 bytes)	ECP (12 bytes)
-----------------------	-------------------

Trong đó:

- DeviceID: Là mã định danh của thiết bị tại mỗi cổng nơi quét tag xe. Từ đó ta có thể biết được chính xác thiết bị tại cổng nào đã gửi gói tin và do đó biết được xe đã đi qua những cổng nào. Được quy định là 7 ký tự, tương ứng là 7 byte.
- ECP: Hiểu một cách đơn giản, đây là mã định danh duy nhất của mỗi tag dán trên mỗi chiếc xe. Nhờ vậy ta có thể biết được chiếc xe nào vừa được xác định. Mã ECP trên mỗi tag có độ dài là 12 byte.

2.3.2. Gói tin lưu trữ lên Server

Tại node Receiver, khi nhận được dữ liệu được gửi thông qua giao thức LoRa, thiết bị tiến hành phân giải gói tin, lấy thời gian tại Receiver và lưu giữ liệu lên Server. Các thông tin được lưu trữ lên Server bao gồm:

time (20 bytes)	DeviceID (7 bytes)	ECP HEX String 24 bytes)
--------------------	-----------------------	-----------------------------

Gói tin được lưu trữ lên Server có dạng json, cấu tạo của gói tin như sau:

```
{
  "time": "dd/mm/yyyy, HH:MM:SS",
  "deviceID": "abc1234",
  "data": "xxxxxxxxxxxxxxxxxxxxxxxxxxxx"
}
```

Trong đó:

- Trường “time”: là thời gian nhận được gói tin LoRa ở Receiver được lấy trên Raspberry PI với định dạng ngày giờ đầy đủ như trên.

- Trường “deviceID”: Là mã định danh của thiết bị tại mỗi cổng nơi quét tag xe. Được phân giải từ gói tin LoRa đã nhận được ở LoRa Receiver.
- Trường “data”: Là chuỗi ECP của tag xe, được phân giải từ gói tin LoRa đã nhận được ở LoRa Receiver. Sau khi nhận được 12 bytes dữ liệu, ở Raspberry PI tiến hành chuyển 12 bytes sang chuỗi HEX string 24 ký tự để lưu lên Server. Do đó, độ dài của trường này là 24 bytes (tức 24 ký tự ASCII).

2.4. Các tần số và thông số hoạt động trên các thiết bị

2.4.1. Các thông số của LoRa

- Là một triển khai có thể của mạng diện rộng công suất thấp, công nghệ có phạm vi truyền gửi dữ liệu tầm xa (LoRa) được coi là tiêu chuẩn giao tiếp không dây trong tương lai cho Internet of Things (IoT) vì nó cung cấp các tính năng cạnh tranh, chẳng hạn như phạm vi giao tiếp, chi phí thấp và tiêu thụ điện năng giảm, làm cho nó trở thành một giải pháp thay thế tối ưu.
- Trong đề tài này, chúng tôi đã cài đặt trong board UCA(Lora) với các thông số như sau:
 - BaudRate = 9600 được cài đặt để truyền thông tin trong kênh (UART) với khả năng truyền tối đa 9600-bit mỗi giây.
 - LoRa_freq (Frequency) = 433E6; Sử dụng tần số 433E6 có nghĩa là EU433. Để khớp với tần số hoạt động của Lora có thể truyền được data với khoảng cách xa, chính xác thì chúng tôi đã thay đổi các khoảng tần số khác nhau như 433E6(EU433), 868E6(EU863 – 870), 915E6(US902-928) và chọn ra được tần số phù hợp quá trình truyền là 433E6.
 - LoRa_SF (Spreading Factor) = 9; Với Spreading factor càng thấp thì tốc độ truyền dữ liệu càng cao. Để phù hợp với khoảng cách truyền của board UCA nên nhóm đã chọn SF = 9 với khoảng cách truyền được trong khoảng 4 – 6 km, hiện tại chỉ gửi dữ liệu dạng String nên số lượng gói tin có thể lên tới 318 và tỷ lệ mất gói tin thấp (0 Packet – 0%) đối với khoảng cách tối thiểu.

SF Setting	Distance (km)	Received Image	Number of Transmitted Packets	Number of Received Packets	Packet Loss Ratio	Elapsed Time (Average)
7	1	Y	314	314	0 P (0%)	1 min + 7 s
7	2	Y	315	315	0 P (0%)	1 min + 7 s
7	3	Y	317	317	0 P (0%)	1 min + 7 s
7	4	Y	312	312	0 P (0%)	1 min + 7 s
7	5	N	311	296	15 P (4.82%)	1 min + 7 s
7	6	N	309	0	309 P (100%)	1 min + 49 s
8	4	Y	312	312	0 P (0%)	1 min + 49 s
8	5	N	316	309	7 P (2.21%)	1 min + 49 s
8	6	N	310	0	310 P (100%)	1 min + 49 s
9	4	Y	318	318	0 P (0%)	3 min + 13 s
9	5	N	315	310	5 P (1.58%)	3 min + 13 s
9	6	N	310	0	310 P (100%)	3 min + 13 s
10	4	Y	309	309	0 P (0%)	5 min + 48 s

Hình 2-16 Bảng các thông số hoạt động của LoRa với các SF khác nhau

- LoRa_bw = 125E3 (bandwidth); Tốc độ dữ liệu phụ thuộc vào bandwidth được sử dụng và spreading factor. Để phù hợp với những điều kiện trên thì nhóm đã chọn bandwidth = 125E3 (125kHz) cho board UCA.

2.4.2. Anten và tần số quét của UHF

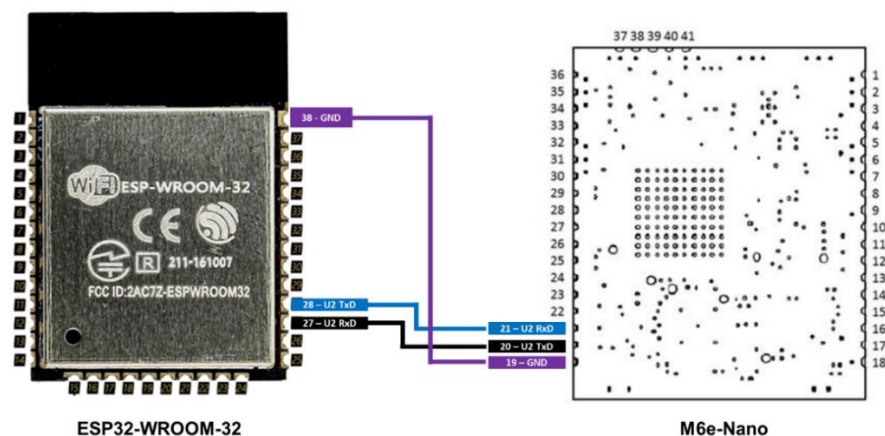
- Antenna là bộ phận cần thiết và đóng vai trò quan trọng đến hiệu năng của đầu đọc UHF RFID. Antenna là bộ phận chuyển đổi tín hiệu đầu ra từ đầu đọc sang sóng điện từ đến các tag xung quanh. Vì vậy, khoảng cách truyền xa hay ngắn nhờ hiệu suất hoạt động của antenna có tốt hay không. Dựa vào hình dạng thì sẽ có rất nhiều loại khác nhau do nguyên lý hoạt động khác nhau. Anten được sử dụng trong bộ UHF Reader là ăng ten được cung cấp bởi RFThings, ăng ten có khả năng quét đẳng hướng.
- Tần số đọc các của các tag đã được gắn trên xe ô tô hiện tại ở Việt Nam nằm trong dải tần số từ 902MHz đến 928 MHz. Do đó ta sẽ phải điều chỉnh dải tần số đọc của UHF Reader bao gồm dải này.
- Bên cạnh đó ta cũng kích hoạt tối đa công suất của đầu đọc để tối ưu được khoảng cách đọc tag. Cụ thể ở đây là 27.00 dBm, tuy nhiên có một nhược điểm là nếu đọc thẻ liên tục trong thời gian dài với công suất lớn thì có thể gây ra tình trạng quá nhiệt và tiêu tốn năng lượng nhiều.

CHƯƠNG 3. HIỆN THỰC VÀ KẾT QUẢ

3.1. Hiện thực hệ thống

3.1.1. Kết nối hệ thống (nối dây)

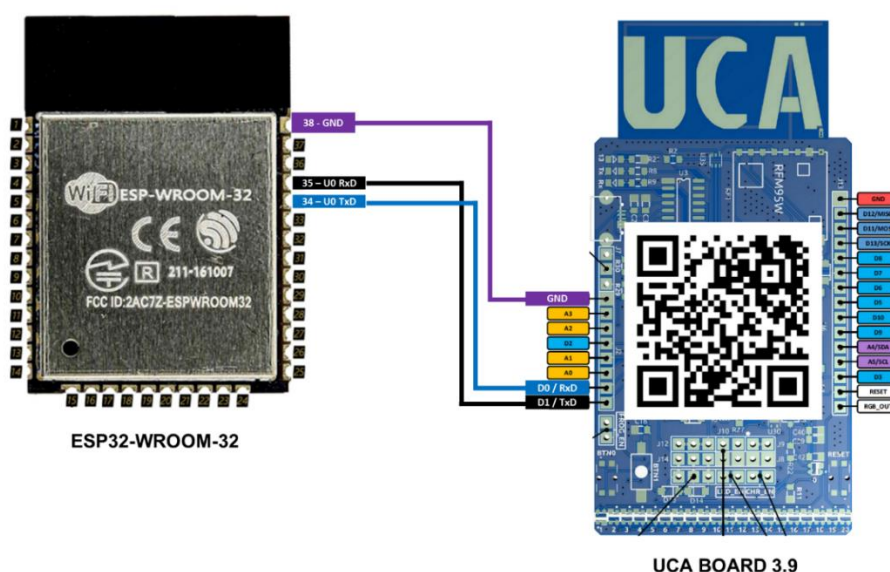
- Kết nối ESP32 và M6e-Nano tạo thành UHF Reader:



Hình 3-1 Sơ đồ kết nối chân UART của ESP ESP32 và M6e-Nano

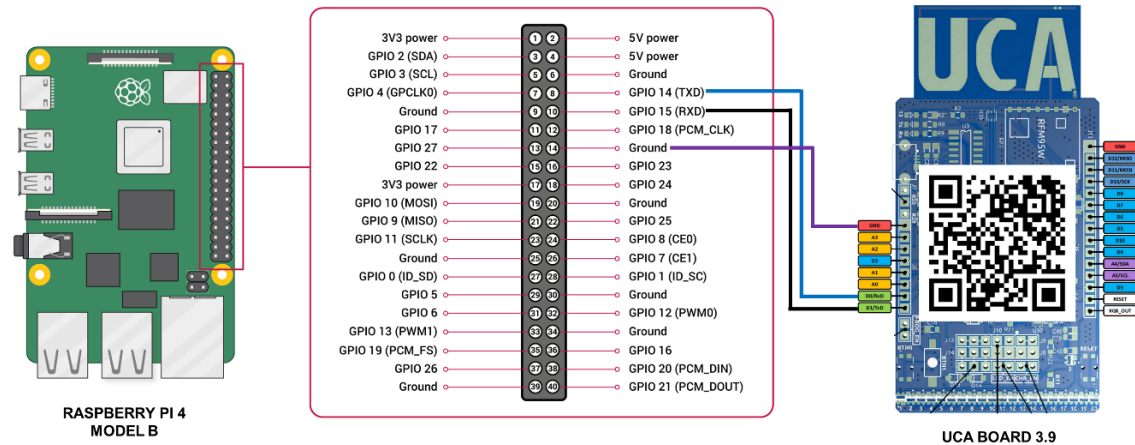
- Mạch M6E-Nano cần nguồn tối thiểu 3.7V để hoạt động. Do đó, nếu cấp nguồn 3.3V VCC cho mạch thì mạch sẽ hoạt động không ổn định.
- Khi kết nối với nguồn 5V từ USB hoặc nguồn ngoài, cần có một bộ chuyển đổi từ 5V sang nguồn thích hợp là 3.7V để mạch hoạt động ổn định.

- Kết nối UHF Reader và UCA Board (LoRa Sender):



Hình 3-2 Sơ đồ kết nối chân UART của UHF Reader và UCA Board

- Kết nối UCA Board (LoRa Receiver) và Raspberry PI:

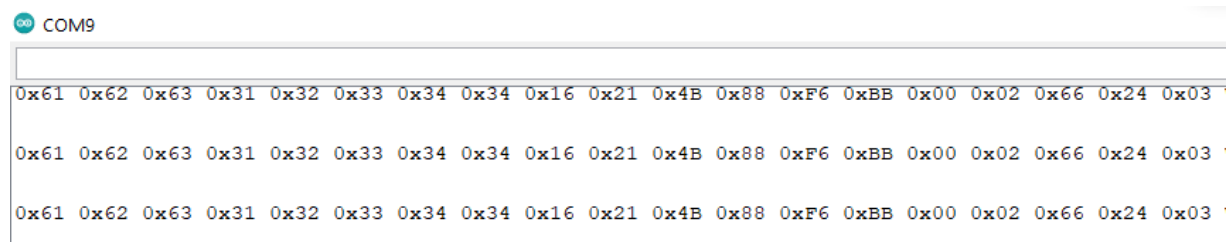


Hình 3-3 Sơ đồ kết nối UART của UCA Board (LoRa Receiver) và Raspberry PI

3.1.2. Gửi gói tin từ Board UHF đến LoRa Sender

Sau khi nhận diện có xe, thì board UHF hoạt động và quét được tag xe. Dữ liệu tag xe đọc được là một chuỗi 12 bytes và được truyền qua UCA Sender bằng UART.

Sau đó gói tin sẽ được cấu trúc lại, thêm tên thiết bị vào để LoRa Receiver có thể nhận được và nhận dạng được khi gửi lên server.



Hình 3-4 Mẫu gói tin truyền đi ở LoRa Sender

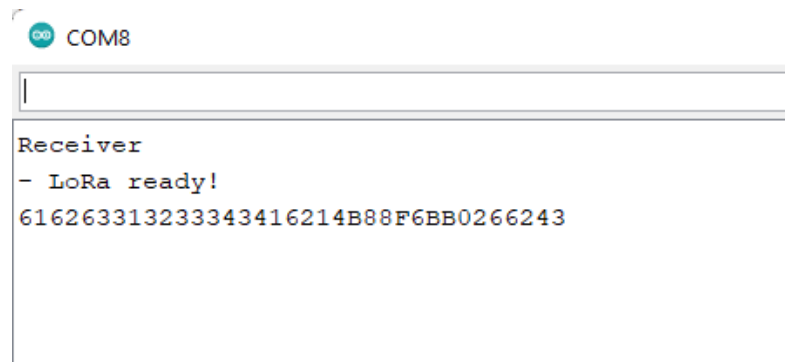
3.1.3. Gửi gói tin đến LoRa Receiver

Sau khi đầu đọc UHF đọc được tag xe và gửi dữ liệu đó qua Lora Sender bằng UART gói tin sẽ được gửi đến Lora Receiver với cấu trúc đầy đủ thông tin như vị trí bãi xe bằng cấu trúc “Device_ID + data” để có thể dễ kiểm soát và đẩy data lên đúng vị trí container ở trên server.

```
//----- Receive LoRa Message -----//

void receiveMessage() {
  int packetSize = LoRa.parsePacket();
  if (packetSize) {
    // received a packet
    byte message[PAYLOAD_LENGTH];
    int i = 0;
    while (LoRa.available()) {
      message[i] = LoRa.read();
      Serial.print(message[i], HEX);
      i++;
    }
    sendACK(message);
    sendUARTMessage(message);
  }
}
```

Hình 3-5 Hàm thực hiện nhận gói tin LoRa ở LoRa Receiver

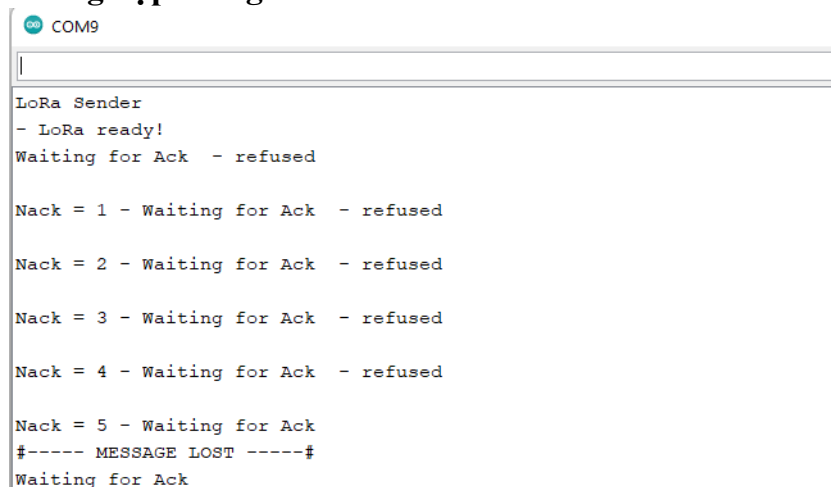


COM8

```
Receiver
- LoRa ready!
616263313233343416214B88F6BB0266243
```

Hình 3-6 Kết quả nhận gói tin LoRa

3.1.4. Trường hợp mất gói tin LoRa



COM9

```
LoRa Sender
- LoRa ready!
Waiting for Ack - refused

Nack = 1 - Waiting for Ack - refused

Nack = 2 - Waiting for Ack - refused

Nack = 3 - Waiting for Ack - refused

Nack = 4 - Waiting for Ack - refused

Nack = 5 - Waiting for Ack
#----- MESSAGE LOST -----#
Waiting for Ack
```

Hình 3-7 Kết quả hiển thị trong trường hợp mất gói tin

Gói tin sẽ được gửi từ LoRa Sender đến LoRa Receiver. Nếu trong trường hợp gói tin bị mất hoặc bị lỗi gói tin. Nack sẽ được tăng lên một giá trị và nếu Nack vượt quá giá trị 5 thì ta sẽ xuất thông báo mất gói tin tại LoRa Sender như hình trên.

3.1.5. Gửi gói tin đến PI

Gói tin sau khi nhận được từ Lora Sender, ngay lập tức Lora Receiver sẽ tính toán và gửi lại một gói tin ACK lại cho Lora Sender để kiểm tra xem gói tin có đúng với gói tin được gửi không và gói tin truyền đi có bị mất không.

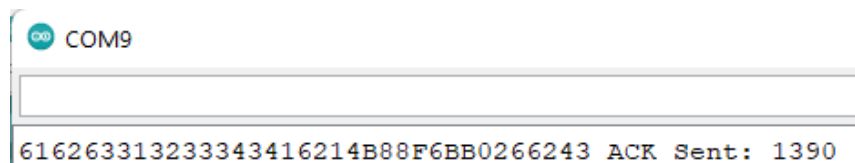
```
//----- Send back LoRa ACK packet -----//

void sendACK(byte *message) {
    int check = 0;
    for (int i = 0; i < PAYLOAD_LENGTH; i++) {
        check += message[i];
    }

    LoRa.beginPacket();
    LoRa.print(String(check));
    LoRa.endPacket();

    Serial.print(" ACK Sent: ");
    Serial.println(check);
}
```

Hình 3-8 Hàm xử lý tính toán checksum và gửi lại cho LoRa Sender



COM9

616263313233343416214B88F6BB0266243 ACK Sent: 1390

Hình 3-9 Kết quả tính toán checksum và thông báo gửi ACK

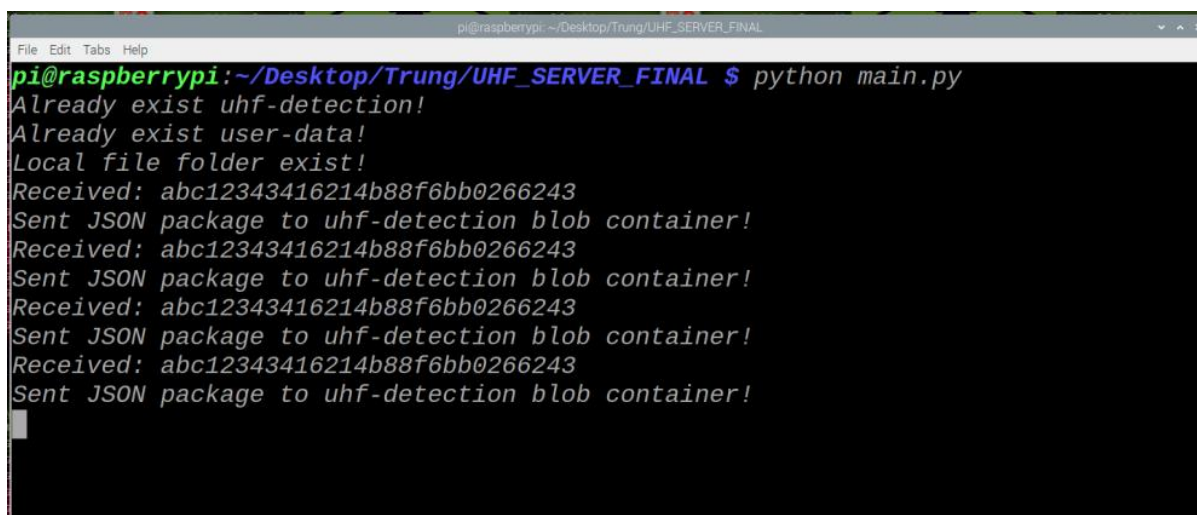
Sau khi kiểm tra gói tin xong, tất cả đều chính xác thì gửi dữ liệu qua Raspberry PI bằng kết nối UART.

```
//----- Send Message to PI via UART -----//

void sendUARTMessage(byte *message) {
    for (int i = 0; i <= PAYLOAD_LENGTH; i++)
        myTransfer.packet.txBuff[i] = message[i];

    myTransfer.sendData(PAYLOAD_LENGTH);
}
```

Hình 3-10 Hàm truyền UART có tin cậy đến Raspberry PI



```

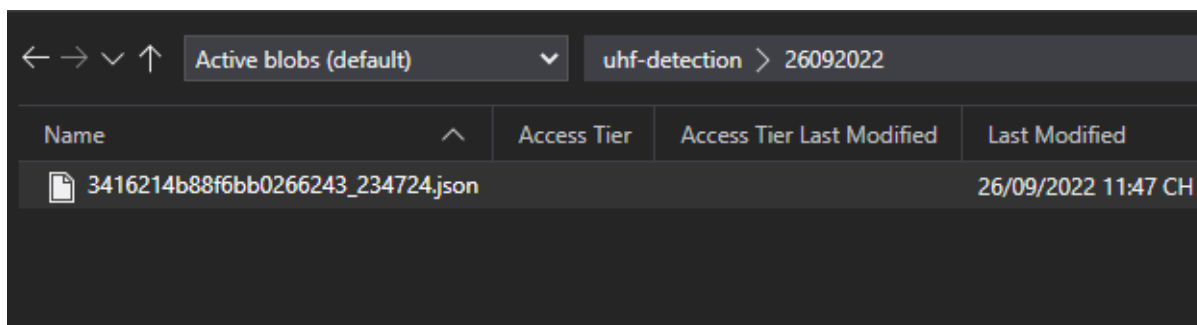
pi@raspberrypi: ~/Desktop/Trung/UHF_SERVER_FINAL
File Edit Tabs Help
pi@raspberrypi:~/Desktop/Trung/UHF_SERVER_FINAL $ python main.py
Already exist uhf-detection!
Already exist user-data!
Local file folder exist!
Received: abc12343416214b88f6bb0266243
Sent JSON package to uhf-detection blob container!
Received: abc12343416214b88f6bb0266243
Sent JSON package to uhf-detection blob container!
Received: abc12343416214b88f6bb0266243
Sent JSON package to uhf-detection blob container!
Received: abc12343416214b88f6bb0266243
Sent JSON package to uhf-detection blob container!


```

Hình 3-11 Kết quả nhận gói tin UART trên Raspberry PI

3.1.6. Gửi gói tin lên Server

Gói tin sau khi gửi lên server sẽ được lưu trữ trong blob container uhf-detection, trong thư mục đặt tên theo ngày nhận được. Tên file là mã EPC + thời gian xe vào bãi như sau:



Name	Access Tier	Access Tier Last Modified	Last Modified
 3416214b88f6bb0266243_234724.json			26/09/2022 11:47 CH

Hình 3-12 File JSON được lưu trữ trong Azure Blob Storage

Mở file được lưu trữ lên Server, ta có thể xem được nội dung file json, với các thông số như đã mô tả trong cấu trúc của gói tin:



```

{} 3416214b88f6bb0266243_234724.json X
C: > Users > Trung > AppData > Local > Temp > 16644359238254 > {} 3416214b88f6bb0266243_234724.json > ...
1 {"time": "26/09/2022, 23:47:24", "deviceID": "abc1234", "data": "3416214b88f6bb0266243"}

```

Hình 3-13 Nội dung file JSON được lưu trữ trên Azure Blob Storage

3.2. Kết quả

Đầu đọc UHF có khả năng nhận diện được các loại tag khác nhau được gắn trên xe ô tô với tầm xa xấp xỉ 3m, mức độ ổn định cao. Nhiệt độ hoạt động của thiết bị cao nếu thiết bị thực hiện quét liên tục các tag.

Truyền nhận gói tin LoRa tầm xa với khoảng cách tối đa mà hoạt động ổn định là dưới 25m.

Xử lý truyền nhận với server thông qua PI và uart với tần suất cao, không bị mất gói tin khi thực hiện các tác vụ xử lý khác.

CHƯƠNG 4. NHẬN XÉT VÀ HƯỚNG PHÁT TRIỂN

4.1. Nhận xét

4.1.1. Ưu điểm

- Mức độ chính xác cao, khó bị ảnh hưởng nhiều từ môi trường

4.1.2. Nhược điểm

- Chi phí đắt để có được thiết bị quét tốt, quét xa hơn.
- Nhiều thẻ khác nhau có tần số khác nhau, cần điều chỉnh tần số quét hợp lý có khả năng thích ứng với nhiều loại thẻ.
- Xe không dán tag hoặc tag được dán ở nhiều chỗ khác nhau.
- Có quá nhiều Sender của LoRa gửi liên tục làm nhiễu LoRa Receiver.

4.2. Hướng phát triển

- Khắc phục các nhược điểm đã nêu ở trên. Bên cạnh đó sẽ kết hợp với ứng dụng Gannha.com để có thể giải quyết nhiều bài toán hơn như định vị bãi đỗ xe, thu phí giữ xe, tính mật độ giao thông, ... để có thể tăng tính ứng dụng của hệ thống.
- Thay thế tầng biên bằng thiết bị Signee đã phát triển bởi công ty TK25.

TÀI LIỆU THAM KHẢO