|  |  |
| --- | --- |
| **Thành viên** | **Nội dung thực hiện** |
| Trần Thanh Hùng | Câu 1, 2 (Station node - công việc - 100%) |
| Nguyễn Trung Hiếu | Câu 1, 2 (Light node - công việc - 100%) |
| Trần Nguyễn Việt Hoàng | Câu 2 (Temp node - công việc - 100%) |

**BẢNG PHÂN CHIA CÔNG VIỆC:**

**ĐÁNH GIÁ KHÁC (\*):**

|  |  |
| --- | --- |
| **Nội dung** | **Kết quả** |
| Tổng thời gian thực hiện bài thực hành trung bình (1) | 2 tuần |
| Link Video thực hiện (2)  *(nếu có)* | <https://drive.google.com/drive/folders/1GxItUcATf3GERZNuozgNdd0HpHUnk_QV?usp=drive_link> |
| Ý kiến (3) *(nếu có)*  + Khó khăn  + Đề xuất … |  |
| Điểm tự đánh giá (4) | 9/10 |
| (\*): phần (1) và (4) bắt buộc thực hiện. | |

GVHD: **Phan Trung Phát**

Nhóm: **TH2.02**

|  |  |
| --- | --- |
| **Họ và tên** | **MSSV** |
| Trần Nguyễn Việt Hoàng | 23520541 |
| Trần Thanh Hùng | 23520580 |
| Nguyễn Trung Hiếu | 23520487 |

**Wireless sensor network**

6

1. Giao thức truyền thông AM (Active Message):

## Định nghĩa:

Active Message (AM) là một giao thức truyền thông nhẹ, cho phép các node trong mạng cảm biến không dây giao tiếp với nhau thông qua việc gửi và nhận các thông điệp (message) nhỏ gọn

AM được thiết kế để tối ưu hóa hiệu suất trong các hệ thống nhúng với tài nguyên hạn chế (cpu, ram, bộ nhớ)

## Đặc điểm chính:

* Gói tin nhỏ gọn: các thông điệp trong AM thưởng chỉ chứa dữ liệu cảm biến và một số thông tin điều khiển, giúp giảm thiểu băng thông và tiết kiệm năng lượng
* Truyền thông không đồng bộ: AM hỗ trợ giao tiếp bất đồng bộ, tức là các node có thể gửi và nhận thông điệp mà không cần đồng bộ hóa thời gian với nhau
* Định dạng thông điệp: mỗi loại thông điệp được gán một AM ID (ví dụ AM\_BLINKTORADIO = 6) giúp các node phân biệt và xử lí đúng loại thông điệp yêu cầu
* Địa chỉ đích: AM hỗ trợ gửi thông điệp đến một node cụ thể hoặc broadcast (AM\_BROADCAST\_ADDR)

## Cách hoạt động:

Gửi thông điệp:

* Một node sử dụng hàm AMSenderC(AM\_ID) để gửi thông điệp với mã định danh AM ID của nó
* Dữ liệu sẽ được đóng gói vào một cấu trúc (ví dụ SenseToRadioMsg) và gửi qua giao thức không dây (Wifi)

Nhận thông điệp:

* Node nhận sử dụng hàm AMReceiverC(AM\_ID) để chỉ nhận các thông điệp của các Node khác có AM ID khớp với nó.
* Khi nhận được dữ liệu, thông điệp sẽ được xử lí trong hàm Receive.receive

1. Thực hành:

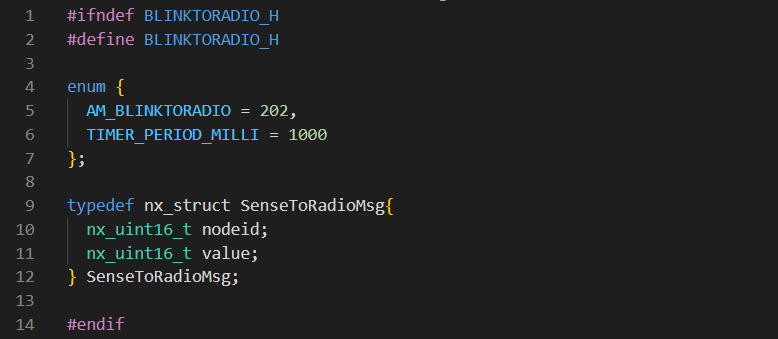
Câu 1: Hoàn thành chương trình gửi dữ liệu cảm biến không dây.

## LightNode

**BlinkToRadio.h**

Định nghĩa mã định danh AM ID của cảm biến và đặt giá trị cho chu kỳ của việc đọc và gửi dữ liệu

Thiết lập cấu trúc gói tin được gửi với các thành phần là ID của node cảm biến (nodeid) và giá trị ánh sáng đo được (value)



1. Định nghĩa hằng số và cấu trúc dữ liệu cho giao tiếp

**Makefile**

A black background with white text

AI-generated content may be incorrect.

1. Cấu hình trình biên dịch

**LightNodeAppC.nc**

Khởi tạo các thành phần chính:

* MainC: cung cấp giao diện Boot để khởi động ứng dụng
* TimerMilliC as Timer0: bộ đếm thời gian
* ActiveMessageC: cung cấp giao diện để bật/tắt giao tiếp không dây (AM)
* AMSenderC(AM\_BLINKTORADIO): thành phần gửi thông điệp AM với ID AM\_BLINKTORADIO
* HamamatsuS10871TsrC as LightSensor: thành phần điều khiển cảm biến ánh sáng

Kết nối giao diện:

* App.Boot -> MainC: liên kết sự kiện khởi động của ứng dụng
* App.Timer0 -> Timer0: kích hoạt việc đọc cảm biến định kỳ
* App.Packet -> AMSenderC và App.AMPacket -> AMSenderC: cung cấp khả năng đóng gói và gửi thông điệp AM
* App.AMControl -> ActiveMessageC: điều khiển bật/tắt giao tiếp không dây
* App.AMSend -> AMSenderC: gửi thông điệp AM
* App.LightRead -> LightSensor: đọc dữ liệu từ cảm biến ánh sáng

A computer screen shot of white text

AI-generated content may be incorrect.

1. Khai báo các thành phần và kết nối giao diện

**LightNodeC.nc**

Gọi thư viện và các thành phần giao diện cần sử dụng

* Boot: để khởi động ứng dụng
* Timer<TMilli>: bộ đếm thời gian định kỳ
* Packet và AMPacket: quản lý việc đóng gói và gửi thông điệp AM
* AMSend: gửi thông điệp AM
* Read<uint16\_t>: đọc dữ liệu từ cảm biến ánh sáng
* SplitControl: điều khiển bật/tắt giao tiếp không dây

A computer screen shot of white text

AI-generated content may be incorrect.

1. Thư viện và giao diện sử dụng

Biến:

* pkt là đối tượng thông diệp (message\_t) dùng để gửi dữ liệu
* busy là cờ để kiểm tra xem node có bận gửi thông điệp hay không
* lux dùng để lưu giá trị ánh sáng sau khi chuyển đổi từ giá trị thô

Khi thiết bị khởi động, thực hiện gọi hàm AMControl.start() để kích hoạt giao tiếp không dây

Hàm AMControl.startDone được dùng để kiểm tra trạng thái giao tiếp

* Kích hoạt bộ đếm thời gian định kì nếu thành công (err == SUCCESS), không thì thử khởi động lại

Mỗi khi bộ đếm thời gian kích hoạt (mỗi 1 giây), nếu nút không bận (!busy), gọi LightRead.read() để đọc dữ liệu từ cảm biến ánh sáng

A screenshot of a computer program

AI-generated content may be incorrect.

1. Khởi tạo giao tiếp không dây và kích hoạt bộ đếm thời gian

Khi cảm biến hoàn tất việc đọc dữ liệu (LightRead.readDone)

* Xử lí chỉ khi trạng thái đọc là SUCCESS và xem node không bận (!busy)
* Chuyển đổi giá trị cảm biến (val) thành giá trị ánh sáng lux bằng công thức: lux = 2.5 \* (val / 4096.0) \* 6250.0
* Tạo gói tin SenseToRadioMsg với ID của node (nodeid) và giá trị ánh sáng đã được xử lí (value)
* Gửi gói tin đến tất cả các Node trong mạng (AM\_BROADCAST\_ADDR) bằng AMSend.send
* Đặt busy = TRUE để tránh việc gửi lại gói tin

Sau khi gửi thông điệp hoàn tất, đặt lại busy = FALSE để cho phép gửi gói tin tiếp theo

A computer screen with white text

AI-generated content may be incorrect.

1. Đọc và gửi dữ liệu ánh sáng

## LightStation

**BlinktoRadio.h**

Định nghĩa AM ID dùng để nhận các thông điệp từ LightNode

Định nghĩa cấu trúc cho dữ liệu nhận được bao gồm ID của node gửi (nodeid) và giá trị ánh sáng nhận được (value)

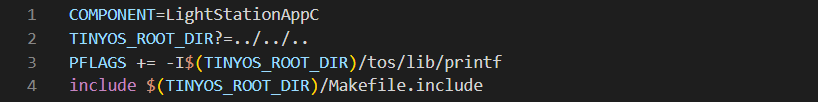
A computer screen with white text

AI-generated content may be incorrect.

1. Định nghĩa hằng số và cấu trúc để nhận dữ liệu

**Makefile**

PFLAGS += -I$(TINYOS\_ROOT\_DIR)/tos/lib/printf: thêm đường dẫn đến thư viện printf để hỗ trợ in dữ liệu qua giao diện nối tiếp



1. Cấu hình trình biên dịch ứng dụng

**LightStationAppC.nc**

Khai báo các thành phần:

* AMReceiverC(AM\_BLINKTORADIO): thành phần nhận thông điệp AM với ID AM\_BLINKTORADIO (202)
* SerialPrintfC: cung cấp chức năng in dữ liệu qua giao diện nối tiếp bằng printf
* SerialStartC: khởi động giao tiếp nối tiếp (serial) trên TelosB

Kết nối giao diện:

* App.Receive -> AMReceiverC: nhận thông điệp AM từ Sensor Node

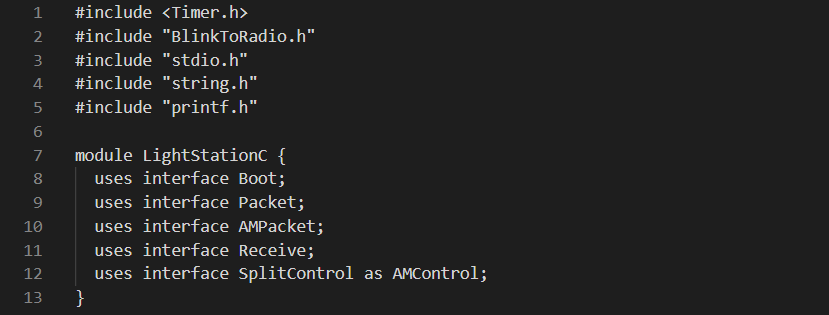
****

1. Khai báo và kết nối các thành phần

**LightStationC.nc**

Giao diện sử dụng

* Boot: khởi động ứng dụng
* Packet và AMPacket: quản lý thông điệp AM nhận được
* Receive: nhận thông điệp không dây
* SplitControl: điều khiển bật/tắt giao tiếp không dây



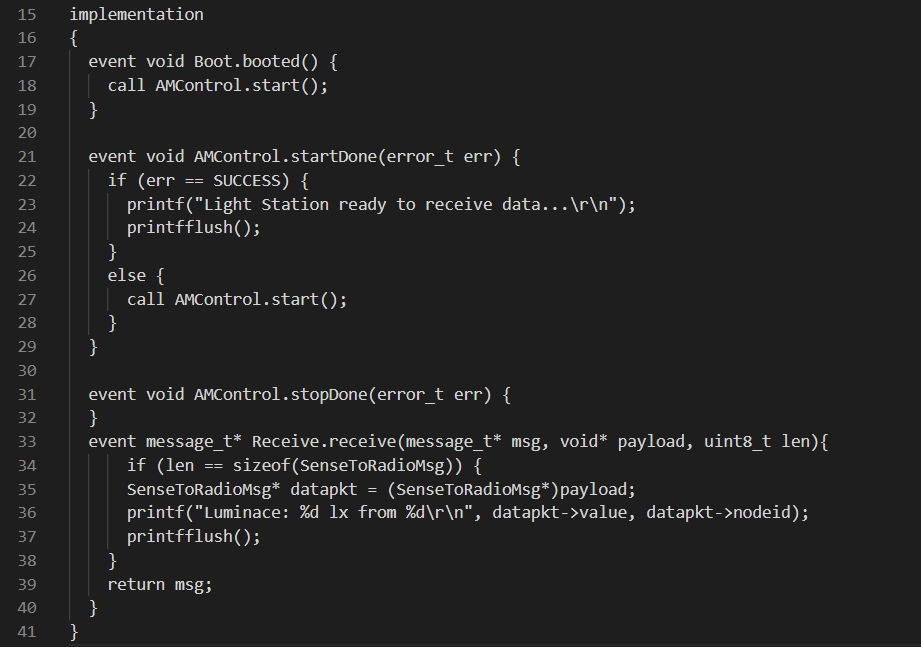
1. Gọi thư viện và giao diện cần dùng

Khi thiết bị khởi động, gọi AMControl.start() để kích hoạt giao tiếp không dây

Nếu giao tiếp không dây khởi động thành công (err == SUCCESS), in thông báo qua giao diện nối tiếp (printf), còn không thì khởi động lại

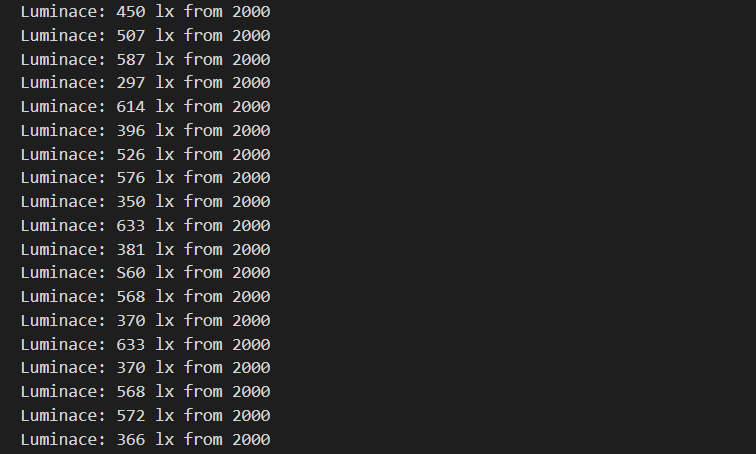
Khi nhận được thông điệp (Receive.receive)

* Xử lí khi độ dài thông điệp (len) khớp với kích thước của SenseToRadioMsg
* Ép kiểu payload thành con trỏ SenseToRadioMsg\* để truy cập dữ liệu
* In dữ liệu ra giao diện nối tiếp "Luminace: <giá trị ánh sáng> lx from <ID nút>"
* Gọi printfflush() để đảm bảo dữ liệu được gửi ngay
* Trả về msg (bộ đệm thông điệp) để TinyOS tái sử dụng



1. Thực hiện khởi động nhận dữ liệu và xử lí

## Kết quả



1. LightNode (Sensor Node) đã thành công lấy được dữ liệu từ cảm biến ánh sáng và gửi dữ liệu đó đến LightStation (Base Station)

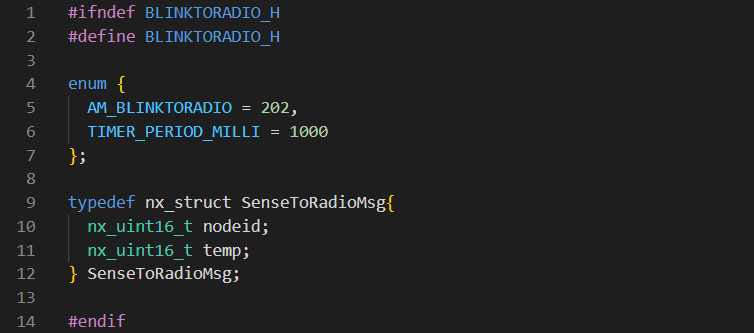
Câu 2: Mở rộng câu 1 thêm một node thu thập cảm biến nhiệt độ.

## TempNode

**BlinkToRadio.h**

Định nghĩa mã định danh cho cảm biến nhiệt độ và đặt chu kì đọc cảm biến là 1s

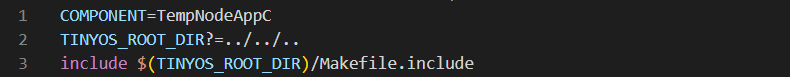
Thiết lập cấu trúc gói tin được gửi với các thành phần là ID của node cảm biến (nodeid) và giá trị nhiệt độ đo được (value)



1. Định nghĩa giá trị và cấu trúc nhận dữ liệu

**Makefile**

Khai báo file chính sẽ được biên dịch là TempNodeAppC.



1. File biên dịch chương trình

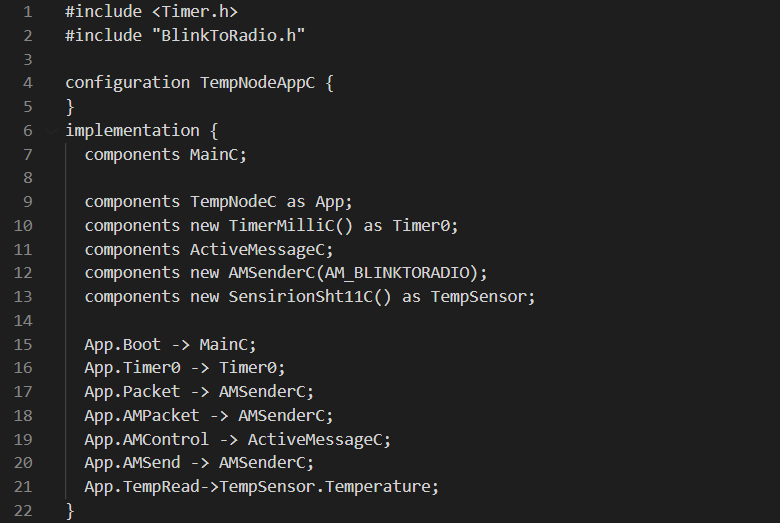
**TempNodeAppC.nc**

Khởi tạo các thành phần chính:

* MainC: cung cấp giao diện Boot để khởi động ứng dụng
* TimerMilliC as Timer0: bộ đếm thời gian
* ActiveMessageC: cung cấp giao diện để bật/tắt giao tiếp không dây (AM)
* AMSenderC(AM\_BLINKTORADIO): thành phần gửi thông điệp AM với ID AM\_BLINKTORADIO
* SensirionSht11c as LightSensor: thành phần điều khiển cảm biến nhiệt độ

Kết nối giao diện:

* App.Boot -> MainC: liên kết sự kiện khởi động của ứng dụng
* App.Timer0 -> Timer0: kích hoạt việc đọc cảm biến định kỳ
* App.Packet -> AMSenderC và App.AMPacket -> AMSenderC: cung cấp khả năng đóng gói và gửi thông điệp AM
* App.AMControl -> ActiveMessageC: điều khiển bật/tắt giao tiếp không dây
* App.AMSend -> AMSenderC: gửi thông điệp AM
* App.TempRead -> TempSensor: đọc dữ liệu từ cảm biến nhiệt độ

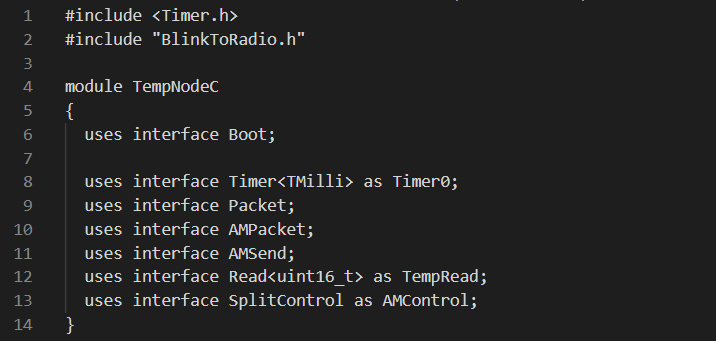


1. File cấu hình

**TempNodeC.nc**

Gọi thư viện và các thành phần giao diện cần sử dụng

* Boot: để khởi động ứng dụng
* Timer<TMilli>: bộ đếm thời gian định kỳ
* Packet và AMPacket: quản lý việc đóng gói và gửi thông điệp AM
* AMSend: gửi thông điệp AM
* Read<uint16\_t>: đọc dữ liệu từ cảm biến nhiệt độ
* SplitControl: điều khiển bật/tắt giao tiếp không dây



1. Khai báo các thành phần sử dụng

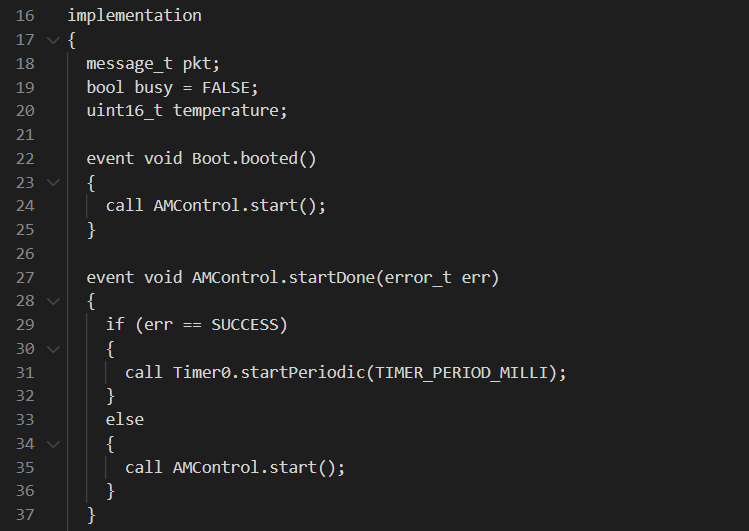
Biến:

* pkt là đối tượng thông diệp (message\_t) dùng để gửi dữ liệu
* busy là cờ để kiểm tra xem node có bận gửi thông điệp hay không
* Temperature dùng để lưu giá trị nhiệt độ sau khi chuyển đổi từ giá trị thô

Khi thiết bị khởi động, thực hiện gọi hàm AMControl.start() để kích hoạt giao tiếp không dây

Hàm AMControl.startDone được dùng để kiểm tra trạng thái giao tiếp

* Nếu thành công (err == SUCCESS) thực hiện kích hoạt bộ đếm thời gian định kì. Ngược lại gọi lại hàm AMControl.start().

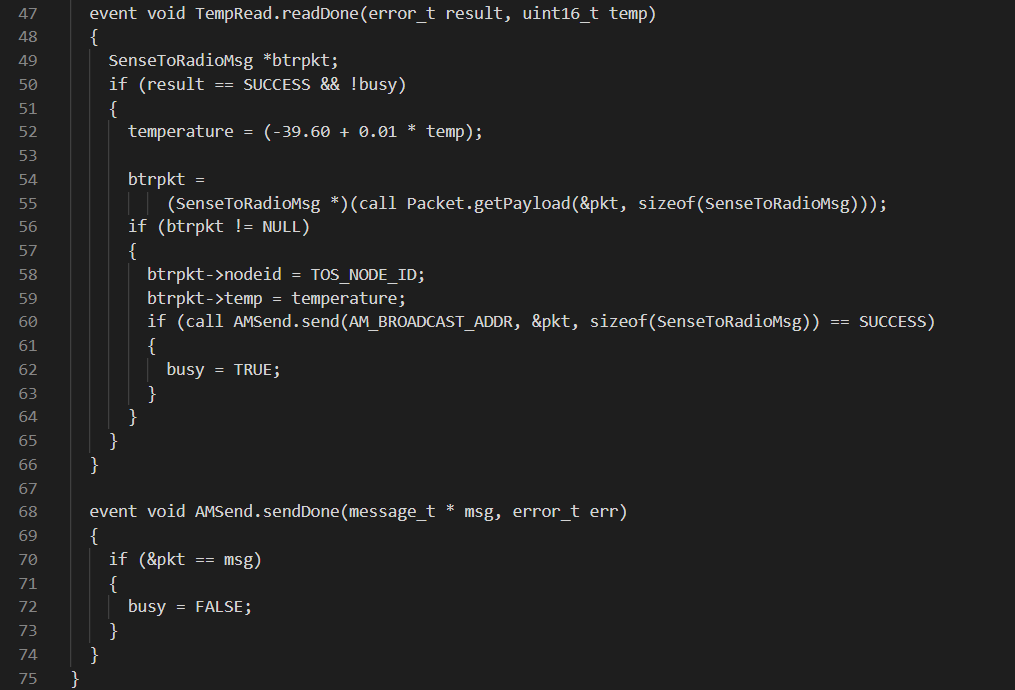


1. Khởi động AMControl và bộ đếm thời gian

Khi cảm biến hoàn tất việc đọc dữ liệu (TempRead.readDone)

* Xử lí gửi giá trị thu được chỉ khi trạng thái đọc là SUCCESS và node không bận (!busy)
* Chuyển đổi giá trị cảm biến (val) thành giá trị nhiệt độ temperature bằng công thức: temperature = -39.60 + 0.1\*temp
* Tạo gói tin SenseToRadioMsg với ID của node (nodeid) và giá trị nhiệt độ đã được xử lí (value)
* Gửi gói tin đến tất cả các Node trong mạng (AM\_BROADCAST\_ADDR) bằng AMSend.send
* Đặt busy = TRUE để tránh việc gửi lại gói tin

Sau khi gửi thông điệp hoàn tất, đặt lại busy = FALSE để cho phép gửi gói tin tiếp theo



1. Hàm gửi dữ liệu nhiệt độ

## LightTempStation

**BlinkToRadio.h**

Định nghĩa AM ID dùng để nhận các thông điệp từ TempNode

Định nghĩa cấu trúc cho dữ liệu nhận được bao gồm ID của node gửi (nodeid) và giá trị nhiệt độ nhận được (value)

A computer screen with white text

AI-generated content may be incorrect.

1. Định nghĩa AM ID và dữ liệu nhận được

**Makefile**

Khai báo file chính sẽ được biên dịch là LightTempStationAppC

A screen shot of a computer

AI-generated content may be incorrect.

1. File biên dịch

**LightTempStationAppC.nc**

Khai báo các thành phần:

* AMReceiverC(AM\_BLINKTORADIO): thành phần nhận thông điệp AM với ID AM\_BLINKTORADIO (202)
* SerialPrintfC: cung cấp chức năng in dữ liệu qua giao diện nối tiếp bằng printf
* SerialStartC: khởi động giao tiếp nối tiếp (serial) trên TelosB

Kết nối giao diện:

* App.Receive -> AMReceiverC: nhận thông điệp AM từ Sensor Node

A computer screen shot of white text

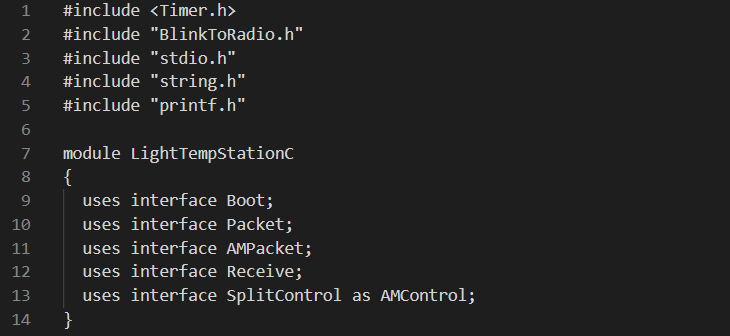
AI-generated content may be incorrect.

1. File cấu hình

**LightTempStationC.nc**

Giao diện sử dụng

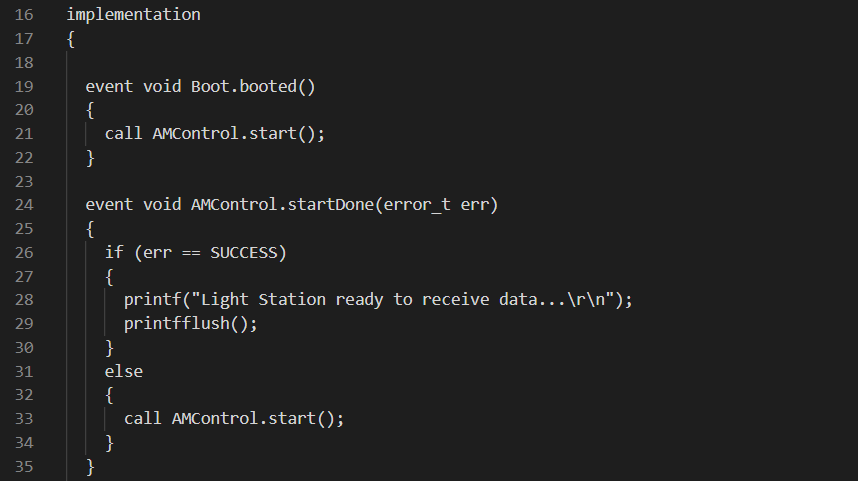
* Boot: khởi động ứng dụng
* Packet và AMPacket: quản lý thông điệp AM nhận được
* Receive: nhận thông điệp không dây
* SplitControl: điều khiển bật/tắt giao tiếp không dây



1. Khai báo thư viện và giao diện cần dùng

Khi thiết bị khởi động, gọi AMControl.start() để kích hoạt giao tiếp không dây

Nếu giao tiếp không dây khởi động thành công (err == SUCCESS), in thông báo qua giao diện nối tiếp (printf), còn không thì khởi động lại



1. Bắt đầu quá trình nhận dữ liệu

Khi nhận được thông điệp (Receive.receive)

* Xử lí khi độ dài thông điệp (len) khớp với kích thước của SenseToRadioMsg
* Ép kiểu payload thành con trỏ SenseToRadioMsg\* để truy cập dữ liệu
* Nếu Nodeid == 2000 (cảm biến ánh sáng) thực hiện in dữ liệu ra giao diện nối tiếp "Luminace: <giá trị ánh sáng> lx from <ID nút>"
* Nếu Nodeid == 3000 (cảm biến nhiệt độ) thực hiện in dữ liệu ra giao diện nối tiếp "Temperature: <giá trị nhiệt độ> C from <ID nút>"
* Gọi printfflush() để đảm bảo dữ liệu được gửi ngay
* Trả về msg (bộ đệm thông điệp) để TinyOS tái sử dụng

A computer screen with white text

AI-generated content may be incorrect.

1. Nhận giá trị cảm biến và in ra màn hình

## Kết quả

A screenshot of a computer

AI-generated content may be incorrect.

1. Kết quả hiển thị

**YÊU CẦU CHUNG**

1. Đánh giá

* Chuẩn bị tốt các yêu cầu đặt ra trong bài thực hành.
* Sinh viên hiểu và tự thực hiện được bài thực hành, trả lời đầy đủ các yêu cầu đặt ra.
* Nộp báo cáo kết quả chi tiết những đã thực hiện, quan sát thấy và kèm ảnh chụp màn hình kết quả *(nếu có)*; giải thích cho quan sát *(nếu có)*.
* Sinh viên báo cáo kết quả thực hiện và nộp bài.

1. Báo cáo

* Nộp file .docx. Tập trung vào nội dung, giải thích.
* Nội dung trình bày bằng Font chữ Cambria hoặc Times New Roman (*tuy nhiên, phải chuyển đổi hết báo cáo này sang 1 font chữ thống nhất*) – cỡ chữ 13. Canh đều (Justify) cho văn bản. Canh giữa (Center) cho ảnh chụp.
* Đặt tên theo định dạng: STTNhom-LabX\_MSSV1\_MSSV2. (trong đó X là Thứ tự buổi Thực hành).

Ví dụ: TH1.01-Lab1\_25520001\_25520002

* Nếu báo cáo có nhiều file, nén tất cả file vào file .ZIP với cùng tên file báo cáo.
* Không đặt tên đúng định dạng – yêu cầu, sẽ KHÔNG chấm điểm bài thực hành.
* Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

Bài sao chép, trễ, … sẽ được xử lý tùy mức độ vi phạm.

**HẾT**