Đồ án tốt nghiệp

Tổng quan về mạng cảm nhận không dây sử dụng CC1010

CHƯƠNG I
TỔNG QUAN VỀ MẠNG CẢM NHẬN KHÔNG DÂY SỬ DỤNG CC1010.

1.1. Giới thiệu về mạng cảm nhận không dây.

Với sự phát triển của công nghệ chế tạo linh kiện điện tử, đặc biệt là công nghệ bán dẫn, các vi điều khiển ngày nay có mật độ tích hợp cao, khả năng xử lý mạnh, kích thước nhỏ, tiêu thụ năng lượng ít, giá thành ngày càng hạ. Khi được cài đặt các phần mềm nhúng, các vi điều khiển này sẽ có khả năng hoạt động độc lập ở các môi trường có vị trí địa lý khác nhau. Nếu kết hợp các vi điều khiển này với các bộ phát sóng vô tuyến và các cảm biến thì chúng có thể trở thành nút mạng trong mạng cảm nhận không dây *(Wireless Sensor Network - WSN).* WSN có thể được tạo ra bằng cách tập hợp nhiều các nút được cấu tạo như vậy. Tại mỗi nút mạng, chúng có thể hoạt động độc lập để tiến hành đo các thông số khác nhau của môi trường như: nhiệt độ, độ ẩm, ánh sáng, áp suất, nồng độ bụi,.... WSN dường như đã trở thành giải pháp hấp dẫn vì mang đến sự tiện lợi về nhiều phương diện, và đặc biệt, trong nhiều trường hợp thậm chí còn hạn chế được sự nguy hiểm cho con người trong những môi trường làm việc khắc nghiệt (nút mạng thay thế cho sự làm việc trực tiếp của con người trong những môi trường có độc tính hay nhiệt độ cao, áp suất cao,...).

Mạng cảm nhận không dây ra đời đáp ứng cho nhu cầu thu thập thông tin về môi trường tại một tập hợp các điểm xác định trong một khoảng thời gian nhất định nhằm phát hiện xu hướng hoặc quy luật vận động của môi trường. Bài toán này được đặc trưng bởi một số lớn các nút mạng, thường xuyên cung cấp thông số môi trường và gửi về một hoặc một

tập trạm gốc có kết nối với trung tâm xử lý (thường là hệ thống máy tính) để phân tích, xử lý, đưa ra các phương án phù hợp hoặc cảnh báo hay đơn thuần chỉ là lưu trữ số liệu.

1.2. Các chỉ tiêu của hệ thống mạng cảm nhận không dây.

Các chỉ tiêu chủ yếu của mạng cảm nhận không dây là: thời gian sống, độ bao phủ, chi phí và dễ triển khai, thời gian đáp ứng, độ chính xác về thời gian, bảo mật, và tốc độ lấy mẫu hiệu quả. Thông thường, khi tăng hiệu quả của chỉ tiêu này lại làm giảm hiệu quả của chỉ tiêu khác. Ví dụ: khi tăng tốc độ lấy mẫu lại làm giảm thời gian sống. Mục đích ở phần này hiểu rõ và cân bằng các chỉ tiêu với khả năng của hệ thống.

1.2.1. Thời gian sống.

Có một giới hạn của mạng cảm nhận không dây đó là thời gian sống. Trong các ứng dụng, các nút mạng thường được đặt ở ngoài môi trường, không có người giám sát theo hàng tháng hay hàng năm. Yếu tố chủ yếu giới hạn thời gian sống của mạng cảm nhận là năng lượng cung cấp. Mỗi nút cần được thiết kế cơ chế quản lý năng lượng nội bộ để tối đa thời gian sống của mạng. Đặc biệt, trong trường hợp mạng an ninh, mỗi nút phải sống trong nhiều năm. Một nút bị lỗi sẽ làm tổn thương hệ thống an ninh.

Trong vài tình huống có thể dùng nguồn năng lượng ngoài. Tuy nhiên, vì ưu điểm chính của mạng không dây là tính linh hoạt dễ triển khai. Yêu cầu nguồn năng lượng ngoài cho tất cả các nút mạng lại mâu thuẫn với ưu điểm này. Một giải pháp được đưa ra là cho một nhóm các nút đặc biệt được cấp nguồn ngoài.

Trong hầu hết các ứng dụng, đặc điểm chính của các nút là tự cấp nguồn. Chúng sẽ có đủ năng lượng cho nhiều năm hoặc có thể lấy năng lượng từ môi trường thông qua thiết bị khác như năng lượng mặt trời,

nguồn áp điện. Cả hai sự lựa chọn đều yêu cầu năng lượng tiêu thụ trung bình của các nút càng ít càng tốt.

Yếu tố quan trọng quyết định thời gian sống là năng lượng tiêu thụ radio. Một nút cảm nhận không dây khi truyền hoặc nhận tín hiệu radio sẽ tiêu thụ năng lượng lớn. Năng lượng tiêu thụ này có thể giảm được bằng cách giảm năng lượng truyền, tức là giảm chu trình làm việc của radio.

1.2.2. Độ bao phủ.

Bên cạnh thời gian sống, độ bao phủ là cũng là tham số đánh giá cho mạng không dây. Nó có thuận lợi là khả năng triển khai một mạng trên một vùng rộng lớn. Điều này làm tăng giá trị hệ thống đối với người dùng cuối. Điều quan trọng là độ bao phủ của mạng không tương đương với khoảng cách kết nối không dây được sử dụng. Kỹ thuật truyền multi-hop có thể mở rộng độ bao phủ của mạng. Về mặt lý thuyết chúng có khả năng mở rộng vô hạn. Tuy nhiên, trong một khoảng cách truyền xác định, giao thức mạng multi-hop làm tăng năng lượng tiêu thụ của các nút, và sẽ làm giảm thời gian sống của mạng. Hơn nữa, chúng đòi hỏi một mật độ tối thiểu, và sẽ làm tăng chi phí triển khai.

Ràng buộc khoảng cách dẫn đến việc mở rộng một số lượng lớn các nút. Giá trị chủ yếu của mạng cảm nhận là khả năng mở rộng. Một người dùng có thể triển khai một mạng nhỏ ban đầu và sau đó tiếp tục thêm các nút. Tăng số lượng các nút trong hệ thống sẽ ảnh hưởng tới thời gian sống. Càng nhiều điểm cảm nhận thì càng có nhiều dữ liệu được truyền và sẽ làm tăng năng lượng tiêu thụ của mạng.

1.2.3. Chi phí và dễ triển khai

Ưu điểm mấu chốt của mạng cảm nhận không dây là dễ triển khai. Người sử dụng không cần phải hiểu về mạng và cơ chế truyền thông khi

làm việc với WSN. Để triển khai hệ thống thành công, WSN cần phải tự cấu hình. Các nút được đặt vào môi trường và có thể hoạt động ngay.

Thêm vào đó, hệ thống cần thích nghi đối với sự thay đổi điều kiện môi trường. Trong suốt thời gian sống, sẽ có thể thay đổi vị trí hay các đối tượng lớn có thể gây nhiễu tới sự truyền thông giữa hai nút. Mạng cần có khả năng tự cấu hình lại để khắc phục những điều này.

Trong thực tế, một phần năng lượng được dành cho kiểm tra và bảo trì hệ thống. Việc tạo ra thông tin chẩn đoán và tái cấu hình sẽ làm giảm thời gian sống của mạng, đồng thời cũng làm giảm tốc độ lấy mẫu.

1.2.4. Thời gian đáp ứng

Trong các ứng dụng cảnh báo, thời gian đáp ứng hệ thống là một thông số quan trọng để đánh giá hệ thống. Một cảnh báo cần được tạo ra ngay lập tức khi nhận thấy có một sự vi phạm. Dù hoạt động năng lượng thấp, các nút cần có khả năng truyền tức thời các thông điệp qua mạng càng nhanh càng tốt. Trong khi những sự kiện như vậy là không thường xuyên, chúng có thể xảy ra tại bất cứ thời điểm nào mà không được báo trước. Thời gian đáp ứng cũng quan trọng khi điều khiển máy móc trong nhà máy. Những hệ thống này chỉ thành hiện thực nếu đảm bảo được thời gian đáp ứng.

Khả năng có thời gian đáp ứng ngắn xung đột với các kỹ thuật làm tăng thời gian sống của mạng. Thời gian sống của mạng có thể tăng bằng cách để các nút chỉ hoạt động radio trong thời gian ngắn. Thời gian đáp ứng có thể cải thiện bằng cách cấp nguồn cho một số nút trong toàn bộ thời gian. Những nút này có thể nghe các thông điệp cảnh báo và chuyển tiếp chúng theo đường khi cần. Tuy nhiên điều này sẽ làm giảm tính dễ triển khai hệ thống.

1.2.5. Độ chính xác về thời gian

Trong ứng dụng theo dõi đối tượng và giám sát môi trường các mẫu từ nhiều nút có liên quan theo thời gian để xác định các hiện tượng khác thường được theo dõi. Tính chính xác của cơ chế tương quan phụ thuộc vào tốc độ lan truyền của hiện tượng được đo. Trong trường hợp xác định nhiệt độ trung bình của một toà nhà, các mẫu chỉ được liên quan với nhau trong vòng cỡ hàng giây. Tuy nhiên, để xác định cách phản ứng của toà nhà đối với một trận động đất thì đòi hỏi độ chính xác cỡ mili giây.

Để đạt được độ chính xác theo thời gian, một mạng cần xây dựng và duy trì một thời gian cơ sở toàn cục có thể được sử dụng để sắp xếp các mẫu và các sự kiện theo thời gian. Trong một hệ phân tán, năng lượng cần được mở rộng để duy trì và phân phát đồng hồ. Thông tin đồng bộ thời gian cần liên tục được truyền giữa các nút. Tần số các thông điệp đồng bộ phụ thuộc vào yêu cầu độ chính xác của đồng hồ thời gian.

1.2.6. Bảo mật

Các thông tin về nhiệt độ đối với ứng dụng giám sát môi trường dường như vô hại nhưng việc giữ bí mật thông tin là rất quan trọng. Các hoạt động của một toà nhà có thể thu thập được dễ dàng bằng cách lấy thông tin về nhiệt độ và ánh sáng của toà nhà đó. Những thông tin này có thể được sử dụng để sắp xếp một kế hoạch tấn công vào một công ty. Do đó, WSN cần có khả năng giữ bí mật các thông tin thu thập được.

Trong các ứng dụng an ninh, dữ liệu bảo mật trở nên rất quan trọng. Không chỉ duy trì tính bí mật, nó còn phải có khả năng xác thực dữ liệu truyền. Sự kết hợp tính bí mật và xác thực là yêu cầu cần thiết của cả ba dạng ứng dụng.

Việc sử dụng mã hoá và giải mã sẽ làm tăng chi phí về năng lượng và băng thông. Dữ liệu mã hoá và giải mã cần được truyền cùng với mỗi gói tin. Điều đó ảnh hưởng tới hiệu suất ứng dụng do giảm số lượng dữ liệu lấy từ mạng và thời gian sống mong đợi.

1.2.7. Tốc độ lấy mẫu hiệu quả

Trong một mạng thu thập dữ liệu. Tốc độ thu thập thông tin hiệu quả là tham số đánh giá hiệu suất của hệ thống. Tốc độ thu thập thông tin hiệu quả là số mẫu lấy được từ mỗi nút riêng lẻ và truyền về điểm thu thập trung tâm. Thông thường, các ứng dụng thu thập dữ liệu chỉ có tốc độ lấy mẫu là 1-2 mẫu trong 1 phút.

Trong một cây thu thập dữ liệu, một nút cần điều khiển dữ liệu của tất cả các con cháu. Nếu mỗi nút con truyền một dữ liệu và một nút có 60 nút con cháu, nó phải truyền 60 lần. Thêm vào đó nó còn phải nhận 60 lần trong một chu kỳ lấy mẫu. Tốc độ và kích thước mạng ảnh hưởng tới tốc độ lấy mẫu hiệu quả.

Một cơ chế để tăng tốc độ lấy thông tin là truyền dữ liệu thô và xử lý dữ liệu nội mạng (innetwork processing). Các dạng nén không gian và thời gian có thể được sử dụng để giảm yêu cầu về băng thông trong khi vẫn duy trì được tốc độ lấy mẫu hiệu quả. Dữ liệu sau đó được truyền qua mạng multi-hop khi băng thông cho phép.

1.3. Các yêu cầu cho nút mạng.

Sau đây là những chỉ tiêu để đánh giá một nút mạng trong WSN. Mục đích là qua các chỉ tiêu đánh giá đó sẽ tạo ra cơ sở để lựa chọn loại VĐK thích hợp và xây dựng hệ thống mạng hiệu quả.

1.3.1. Năng lượng

Để đạt được yêu cầu duy trì năng lượng hoạt động trong nhiều năm, các nút mạng cần phải tiêu thụ năng lượng rất thấp. Việc tiêu thụ năng lượng thấp chỉ đạt được bằng cách kết hợp các thành phần phần cứng năng lượng thấp và chu trình hoạt động ngắn. Trong thời gian hoạt động, truyền thông radio sẽ tiêu thụ một phần năng lượng đáng kể trong tổng mức tiêu thụ năng lượng của nút mạng. Các thuật toán và các giao thức cần được

phát triển để giảm hoạt động truyền nhận radio. Điều này có thể đạt được bằng cách sử dụng sự tính toán cục bộ để giảm luồng dữ liệu nhận được từ cảm biến. Ví dụ, các sự kiện từ nhiều nút cảm biến có thể được kết hợp cùng nhau thành một nhóm các nút trước khi truyền một kết quả đơn lẻ qua mạng cảm nhận.

1.3.2. Tính mềm dẻo

Các nút mạng phải có khả năng thích nghi cao để thích hợp với các ngữ cảnh khác nhau. Mỗi một ứng dụng sẽ yêu cầu về thời gian sống, tốc độ lấy mẫu, thời gian đáp ứng và xử lý nội mạng khác nhau. Một kiến trúc WSN cần phải đủ mềm dẻo để cung cấp một dải rộng các ứng dụng. Thêm vào đó, vì lý do chi phí mỗi thiết bị sẽ chỉ có phần cứng và phần mềm cho một ứng dụng cụ thể. Kiến trúc cần đơn giản để kết hợp giữa phần cứng và phần mềm. Vì vậy, những thiết bị này đòi hỏi một mức độ cao về tính modul của phần cứng và phần mềm trong khi vẫn giữ được tính hiệu quả.

1.3.3. Sức mạnh

Để hỗ trợ cho các yêu cầu về thời gian sống, mỗi nút cần phải càng mạnh càng tốt. Trong sự thực tế, hàng trăm nút mạng sẽ hoạt động trong nhiều năm. Để đạt được điều này, hệ thống cần được xây dựng để vẫn có thể hoạt động khi một nút bị lỗi. Modul hoá hệ thống là một công cụ mạnh để phát triển hệ thống. Bằng cách chia chức năng hệ thống thành các thành phần con độc lập, mỗi chức năng có thể được kiểm tra đầy đủ trước khi kết hợp chúng thành một ứng dụng hoàn chỉnh. Để làm điều này, các thành phần hệ thống phải độc lập đến mức có thể và có giao tiếp chặt chẽ, để ngăn chặn các tương tác không mong đợi. Để tăng sức mạnh hệ thống khi nút bị lỗi, một WSN cũng cần có khả năng đối phó với nhiễu ngoài. Các mạng thường cùng tồn tại cùng với các hệ thống không dây khác, chúng cần có khả năng để thích nghi theo các hành động khác nhau. Nó cũng phải có khả năng hoạt động trong môi trường đã có các thiết bị không dây khác

hoạt động với một hay nhiều tần số. Khả năng tránh tắc nghẽn tần số là điều cốt yếu để đảm bảo một sự triển khai thành công.

1.3.4. Bảo mật

Để đạt được mức độ bảo mật mà ứng dụng yêu cầu, các nút riêng lẻ cần có khả năng thực hiện sự mã hoá phức tạp và thuật toán xác thực. Truyền dữ liệu không dây rất dễ bị chặn. Chỉ có một cách bảo mật dữ liệu là mã hoá toàn bộ dữ liệu truyền. CPU cần có khả năng tự thực hiện các thao tác mật mã. Để bảo mật toàn bộ dữ liệu truyền, các nút cần tự bảo mật dữ liệu của chúng. Trong khi chúng không có lượng lớn dữ liệu lưu bên trong, chúng sẽ phải lưu các khoá mã hoá được sử dụng trên mạng. Nếu những khoá này bị lộ, tính bảo mật của mạng sẽ mất. Để có được tính bảo mật tốt, cần phải rất khó để lấy được khoá mã hóa từ một nút.

1.3.5. Truyền thông

Một chỉ tiêu đánh giá cho bất kỳ WSN là tốc độ truyền, năng lượng tiêu thụ và khoảng cách. Trong khi độ bao phủ của mạng không bị giới hạn bởi khoảng cách truyền của các nút riêng biệt, khoảng cách truyền có một ảnh hưởng quan trọng tới mật độ tối thiểu có thể chấp nhận được. Nếu các nút được đặt rất xa nó không thể tạo được kết nối với mạng liên kết hoặc với một nút dự trữ để có được độ tin cậy cao. Nếu khoảng cách truyền radio thoả mãn một mật độ nút cao, các nút thêm vào sẽ làm tăng mật độ hệ thống tới một mức độ nào đó cho phép. Tốc độ truyền cũng có ảnh hưởng lớn đến hiệu suất của nút mạng. Tốc độ truyền cao hơn làm cho khả năng lấy mẫu hiệu quả hơn và năng lượng tiêu thụ của mạng ít hơn. Khi tốc độ tăng, việc truyền mất ít thời gian hơn và do đó đòi hỏi ít năng lượng hơn. Tuy nhiên, khi tăng tốc độ cũng thường làm tăng năng lượng tiêu thụ radio. Mọi thứ trở nên bằng nhau, một tốc độ cao sẽ tăng hiệu suất hệ thống. Tuy nhiên, tăng tốc độ có ảnh hưởng lớn tới năng lượng tiêu thụ và yêu cầu tính

toán của nút. Tổng thể, lợi ích của việc tăng tốc độ có thể được bù lại bởi các yếu tố khác.

1.3.6. Tính toán

Hai việc tính toán cho nút mạng tập trung chủ yếu vào xử lý dữ liệu nội mạng và quản lý các giao thức truyền thông không dây mức thấp. Có những yêu cầu giới hạn về mặt thời gian thực đối với truyền thông và cảm biến. Khi dữ liệu tới trên mạng, CPU cần điều khiển đồng thời radio và ghi lại/giải mã (record/decode) dữ liệu tới. Tốc độ truyền cao hơn đòi hỏi tính toán nhanh hơn. Điều tương tự cũng đúng đối với xử lý dữ liệu cảm biến. Các cảm biến tương tự có thể phát ra hàng ngàn mẫu trong một giây. Các thao tác xử lý cảm biến nói chung bao gồm lọc số, trung bình hoá, nhận biết ngưỡng, phân tích phổ, … Để tăng khả năng xử lý cục bộ, các nút láng giềng có thể kết hợp dữ liệu với nhau trước khi truyền đi trên mạng. Các kết quả từ nhiều nút mạng có thể được tổng hợp cùng nhau. Xử lý nội mạng này đòi hỏi thêm tài nguyên tính toán. Ngoài ra, ứng dụng xử lý dữ liệu có thể tiêu thụ một lượng tính toán phụ thuộc vào các phép toán được thực hiện.

1.3.7. Đồng bộ thời gian

Để hỗ trợ sự tương quan thời gian đọc cảm biến và chu trình hoạt động ngắn của ứng dụng thu thập thông tin, các nút cần duy trì đồng bộ thời gian chính xác với các thành viên khác trong mạng. Các nút cần ngủ và thức dậy cùng nhau để chúng có thể định kỳ truyền thông cho nhau. Các lỗi trong cơ chế tính thời gian sẽ tạo nên sự không hiệu quả dẫn đến làm tăng chu trình làm việc và làm giảm tuổi thọ của hệ thống mạng.

1.3.8. Kích thước và chi phí

Kích thước vật lý và giá thành của mỗi nút riêng lẻ có ảnh hưởng tới sự dễ dàng và chi phí khi triển khai. Việc giảm giá thành trên mỗi nút sẽ

làm cho có khả năng mua thêm nhiều nút, triển khai một mạng thu thập với mật độ cao hơn, và thu thập được nhiều dữ liệu hơn. Kích thước vật lý cũng ảnh hưởng tới sự dễ dàng khi triển khai mạng. Các nút nhỏ hơn có thể được đặt ở nhiều vị trí hơn và được sử dụng trong nhiều tình huống hơn. Trong tình huống theo dõi nút đối tượng, các nút nhỏ hơn, rẻ hơn sẽ tăng khả năng theo dõi nhiều đối tượng hơn.

1.4. Các đặc điểm giúp CC1010 trở thành nút mạng.

Lựa chọn vi điều khiển nào để xây dựng nút mạng đáp ứng được các yêu cầu về nút mạng và chỉ tiêu của hệ thống mạng đã đưa ra trên đây đã trở thành một vấn đề quan trọng. Vì khi chọn được một vi điều khiển thích hợp sẽ làm cho quá trình xây dựng hệ thống dễ triển khai hơn, mạng hoạt động ổn định trong khoảng thời gian dài hơn và có thể sử dụng trong các ứng dụng mới.

CC1010 là một vi mạch thu phát siêu cao tần rất phù hợp để trở thành nút mạng cảm nhận không dây. Vi điều khiển này có rất nhiều đặc điểm phù hợp để trở thành nút mạng như: được tích hợp ROM, RAM, có bộ chuyển đổi ADC 10 bit, có thể hoạt động ở tần số từ 3MHz đến 24MHz, tiêu thụ năng lượng ít, kích thước nhỏ, có bộ nhớ Flash 32kB, ...Và đặc điểm quan trọng nhất giúp CC1010 được lựa chọn làm nút mạng cảm nhận không dây là nó được tích hợp truyền nhận không dây (300 – 1000MHz) . Vì vậy CC1010 thường được dùng để thiết kế các ứng dụng không dây ít tiêu thụ năng lượng. Khi ghép nối với các đầu đo, không những có khả năng tạo thành các điểm đo thông số môi trường mà còn có thể xây dựng thành một nút mạng trong cấu hình mạng cảm nhận không dây mà không cần đến nhiều thành phần phụ trợ khác.

Ngoài ra, hãng Chipcon còn đưa ra các thư viện để làm việc với CC1010. Do đó, việc viết chương trình sử dụng CC1010 trở nên dễ dàng và thuận tiện hơn.

Hiện nay, hãng Chipcon cung cấp Module CC1010EM (Evaluation Module) để phát triển thêm các ứng dụng của CC1010. Trên Module CC1010EM có tích hợp hầu hết cách linh kiện cần cho một nút mạng như: CC1010, các chân cổng, một cảm biến nhiệt độ đưa vào chân AD1, ăngten, dao động thạch anh. Module CC1010EM nhỏ gọn và đáp ứng được các chức năng của nút mạng là: chức năng mạng và chức năng cảm nhận. Trong khoá luận tốt nghiệp này đã sử dụng Module CC1010EM trong thử nghiệm và dùng ampe kế đo được dòng điện tiêu thụ của nút mạng, qua đó phản ánh hiệu quả mà phần mềm nhúng tiết kiệm tiêu thụ năng lượng đạt được.

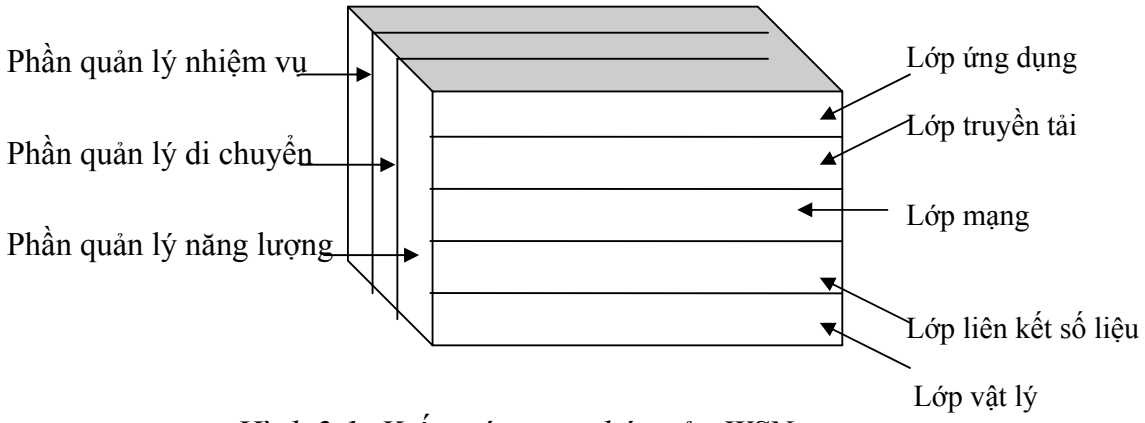
1.5. Kết luận.

Chương 1 đã đưa ra các chỉ tiêu của hệ thống mạng cảm nhận không dây và các yêu cầu đối với nút mạng. Từ đó lựa chọn được vi điều khiển thích hợp để xây dựng nút mạng. Đó là vi điều khiển CC1010 – là vi điều khiển họ 8051, có thể dùng ngôn ngữ lập trình C và chương trình dịch Keil µVision2.0 để viết chương trình ứng dụng nạp cho CC1010.

CHƯƠNG II
VẤN ĐỀ TIẾT KIỆM TIÊU THỤ NĂNG LƯỢNG.

Năng lượng tiêu thụ phụ thuộc vào nhiều yếu tố sử dụng khác nhau như: Kiến trúc giao thức mạng, Giao thức chọn đường, Hoạt động truyền nhận không dây.

2.1. Kiến trúc giao thức mạng.



Hình 2.1: Kiến trúc giao thức của WSN.

Kiến trúc giao thức này kết hợp giữa năng lượng và chọn đường, kết hợp số liệu với các giao thức mạng, sử dụng năng lượng hiệu quả. Kiến trúc giao thức bao gồm: lớp vật lý, lớp liên kết số liệu, lớp mạng, lớp truyền tải, lớp ứng dụng, phần quản lý công suất, phần quản lý di động và phần quản lý nhiệm vụ.

Lớp vật lý cung cấp các kỹ thuật điều chế, phát và thu. Vì môi trường có tạp âm và các nút cảm biến có thể di động, giao thức điều khiển truy cập môi trường (MAC) phải xét đến vấn đề công suất và phải có khả năng tối thiểu hoá việc va chạm với thông tin quản bá của các nút lân cận.

Lớp mạng quan tâm đến việc chọn đường số liệu được cung cấp bởi lớp truyền tải.

Lớp truyền tải giúp duy trì luồng số liệu nếu ứng dụng mạng cảm biến yêu cầu. Tuỳ theo nhiệm vụ cảm biến, các loại phần mềm khác nhau có thể được xây dựng và sử dụng ở lớp ứng dụng. Ngoài ra, các phần quản lý công suất, di chuyển và nhiệm vụ sẽ giám sát việc sử dụng công suất, sự di chuyển và thực hiện nhiệm vụ giữa các nút cảm biến. Những phần này giúp các nút cảm biến phối hợp nhiệm vụ cảm biến và tiêu thụ năng lượng tổng thể thấp hơn.

Phần quản lý năng lượng điều khiển việc sử dụng năng lượng của nút mạng. Ví dụ, nút mạng có thể tắt khối thu của nó sau khi thu được một bản tin từ một nút lân cận. Điều này giúp tránh tạo ra các bản tin giống nhau. Cũng vậy, khi mức năng lượng của nút mạng thấp, nó sẽ phát quảng bá tới các nút lân cận để thông báo nó có mức năng lượng thấp và không thể tham gia vào các bản tin chọn đường. Phần năng lượng còn lại sẽ dành riêng cho nhiệm vụ cảm biến.

Phần quản lý di chuyển phát hiện và ghi lại sự di chuyển của các nút cảm biến để duy trì tuyến tới người sử dụng và các nút cảm biến có thể lưu vết của các nút cảm biến lân cận. Nhờ xác định được các nút cảm biến lân cận, các nút cảm biến có thể cân bằng giữa cân bằng giữa công suất của nó và nhiệm vụ thực hiện.

Phần quản lý nhiệm vụ dùng để làm cân bằng và lên kế hoạch các nhiệm vụ cảm biến trong một vùng xác định. Không phải tất cả các nút cảm biến trong vùng đó đều phải thực hiện nhiệm vụ cảm biến tại cùng một thời điểm. Kết quả là một số nút cảm biến thực hiện nhiều hơn các nút khác tuỳ theo mức công suất của nó. Những phần quản lý này là cần thiết để các nút cảm biến có thể làm việc cùng nhau theo một cách thức sử dụng hiệu quả công suất, chọn đường số liệu trong mạng cảm biến di động và phân chia tài nguyên giữa các nút cảm biến.

Kiến trúc mạng như trên góp phần quản lý năng lượng của các nút mạng đồng thời duy trì hoạt động của toàn mạng trong thời gian dài hơn.

2.2. Giao thức chọn đường.

2.2.1. Khó khăn trong giao thức chọn đường.

Mặc dù các ứng dụng của mạng WSN là rất lớn, tuy nhiên những mạng này có một số hạn chế như: giới hạn về nguồn công suất, khả năng tính toán và độ rộng băng thông. Một số giao thức chọn đường, quản lý công suất và trao đổi số liệu đã được thiết kế cho WSN với yêu cầu quan trọng nhất là tiết kiệm được năng lượng.

Một trong những mục tiêu thiết kế chính của WSN là kéo dài thời gian sống của mạng và tránh suy giảm kết nối nhờ các kỹ thuật quản lý năng lượng. Vấn đề này cần được giải quyết triệt để thì mới đạt được hiệu quả truyền tin trong WSN. Các giao thức chọn đường trong WSN có thể khác nhau tuỳ theo ứng dụng và cấu trúc mạng. Tuy nhiên, việc chọn đường gặp phải những khó khăn như: Phân bố nút, tiêu thụ năng lượng không được làm mất độ chính xác, tính không đồng nhất của nút mạng, tính động của mạng, khả năng định cỡ, khả năng chống lỗi, chất lượng dịch vụ.

Phân bố nút: Việc phân bố nút trong WSN phụ thuộc vào ứng dụng và có thể được thực hiện bằng tay hoặc phân bố ngẫu nhiên. Khi phân bố bằng tay, số liệu được chọn đường thông qua các đường đã định trước. Tuy nhiên, khi phân bố các nút ngẫu nhiên sẽ tạo ra một cấu trúc chọn đường đặc biệt (Ad-hoc).

Tiêu thụ năng lượng không được làm mất độ chính xác: Các nút cảm biến có thể sử dụng quá các giới hạn về công suất để thực hiện tính toán và truyền tin trong môi trường vô tuyến. Thời gian sống của nút mạng cảm biến phụ thuộc rất nhiều vào thời gian sử dụng của pin. Trong WSN đa bước nhảy, mỗi nút đóng hai vai trò là truyền số liệu và chọn đường. Một

số nút cảm biến hoạt động sai chức năng do lỗi nguồn công suất có thể gây ra sự thay đổi cấu hình mạng nghiêm trọng và phải chọn đường lại các gói hoặc phải tổ chức lại mạng.

Tính không đồng nhất của nút mạng: Nghĩa là các nút mạng có khả năng tính toán, khả năng truyền tin và có công suất khác nhau. Khi đó sẽ gây khó khăn cho kỹ thuật chọn đường.

Khả năng chống lỗi: Một số nút cảm biến có thể bị lỗi hoặc bị ngắt do thiếu công suất, hỏng phần cứng hoặc bị nhiễu môi trường. Sự cố của các nút cảm biến không được ảnh hưởng tới nhiệm vụ của toàn mạng cảm biến. Nếu có nhiều nút bị lỗi, các giao thức chọn đường và điều khiển truy cập môi trường (MAC) phải thành lập các tuyến mới tới nút gốc. Việc này có thể cần thiết phải điều chỉnh công suất phát và tốc độ tín hiệu trên các tuyến hiện tại để giảm sự tiêu thụ năng lượng hoặc là các gói phải chọn đường lại qua các các vùng mạng có công suất khả dụng lớn hơn.

Khả năng định cỡ: Số lượng nút mạng có thể là hàng trăm, hàng nghìn hoặc nhiều hơn. Bất kỳ phương pháp chọn đường nào cũng phải có khả năng làm việc với một số lượng lớn các nút cảm biến như vậy.

Tính động của mạng: Trong nhiều nghiên cứu, các nút cảm biến được coi là cố định. Tuy nhiên, trong một số ứng dụng, cả nút gốc và các nút cảm biến có thể di chuyển. Khi đó, các bản tin chọn đường từ hoặc tới các nút di chuyển sẽ gặp phải các vấn đề như: đường liên lạc, cấu hình mạng, năng lượng, độ rộng băng,…

Chất lượng dịch vụ: Khi năng lượng gần hết, các nút mạng có thể yêu cầu giảm chất lượng các kết quả để giảm mức tiêu thụ năng lượng của nút và kéo dài thời gian sống của toàn mạng.

Mặc dù việc chọn đường gặp nhiều khó khăn nhưng khi có một giao thức chọn đường tốt sẽ tiết kiệm thời gian truyền nhận thông tin, giảm năng

lượng tiêu hao lãng phí do tắc nghẽn, lỗi đường truyền,.... Vì vậy, người ta đã đưa ra một số giao thức chọn đường mà dưới đây sẽ trình bầy.

2.2.2. Các giao thức chọn đường.

Phần này sẽ trình bày về phương pháp chọn đường trong WSN. Nói chung, các giao thức chọn đường được chia làm 3 loại dựa vào cấu trúc mạng: ngang hàng, phân cấp hoặc dựa vào vị trí.

Trong chọn đường ngang hàng, tất cả các nút thường có vai trò hoặc chức năng như nhau như: cảm nhận, truyền,... Chúng sẽ thực hiện việc cảm nhận thông tin cần thiết và truyền số liệu về nút gốc.

Trong chọn đường phân cấp, các nút sẽ đóng vai trò khác nhau trong mạng, sẽ có nút chủ đại diện cho từng nhóm các nút cảm biến để nhận số liệu từ các nút cảm biến truyền tới. Các nút chủ cũng tập hợp số liệu từ các nút trong nhóm của nó trước khi gửi số liệu đến nút gốc. Nút chủ sẽ thay đổi khi bắt đầu chu kỳ làm việc mới và sẽ thay bằng nút khác có khả năng đảm nhận chức năng này.

Trong chọn đường dựa theo vị trí thì vị trí của các nút cảm biến sẽ được dùng để chọn đường số liệu. Giao thức chọn đường còn có thể được phân loại dựa theo đa đường, yêu cầu, hỏi/đáp, QoS và liên kết tuỳ thuộc vào chế độ hoạt động. Ngoài ra, cũng có thể chia thành 3 loại: chủ động, tương tác hoặc lai ghép tuỳ thuộc vào cách thức mà nguồn tìm đường tời đích.

Một giao thức chọn đường được coi là thích ứng nếu các tham số của hệ thống có thể điều khiển được để phù hợp với các trạng thái mạng hiện tại và các mức năng lượng khả dụng. Một số giao thức chọn đường đã phát huy hiệu quả tiết kiệm tiêu thụ năng lượng như: LEACH,

TEEN&APTEEN, MECN&SVECN, PEGASIS thuộc chọn đường phân cấp; SPIN, Directed Diffusion, CADR, CUGAR thuộc chọn đường ngang hàng. Trong đó đáng quan tâm nhất là giao thức chọn đường LEACH.

LEACH (Low Energy Adaptive Clustering Hierarchy) – phân cấp nhóm thích ứng công suất thấp, giao thức này cho phép tiết kiệm năng lượng trong mạng WSN. Mục đích của LEACH là lựa chọn ngẫu nhiên các nút cảm biến làm các nút chủ, do đó, việc tiêu hao năng lượng khi liên lạc với nút gốc được trải đều cho tất cả các nút cảm biến trong mạng. Quá trình hoạt động của LEACH được chia thành 2 bước: bước thiết lập và bước ổn định. Thời gian bước ổn định kéo dài hơn so với thời gian của bước thiết lập để giảm thiểu phần điều khiển. Tại bước thiết lập sẽ xác định nút mạng nào sẽ làm chủ. Tại bước ổn định, các nút cảm biến bắt đầu cảm nhận và truyền số liệu về nút chủ, nút chủ cũng tập hợp số liệu từ các nút trong nhóm trước khi gửi các số liệu này về nút gốc.

Theo thử nghiệm mô phỏng giao thức LEACH của mạng WSN có 160 nút, phân bố đều, công suất ban đầu của nút là 3.0. Kết quả thu được là: khi truyền trực tiếp tới nút gốc, sau khoảng 470 chu kỳ thời gian sẽ kết thúc truyền tin; khi sử dụng giao thức LEACH, sau khoảng 685 chu kỳ thời gian mới kết thúc truyền tin. Kết quả này cho thấy đây là một phương pháp chọn đường phân cấp có khả năng tiết kiệm được năng lượng và kéo dài thời gian sống của mạng.

Tuy nhiên, cơ chế hoạt động của LEACH là lựa chọn số liệu được tập trung và thực hiện theo chu kỳ. Do đó, giao thức này chỉ thích hợp với yêu cầu giám sát liên tục bởi mạng cảm biến. Với ứng dụng mà người sử dụng không cần tất cả các số liệu ngay lập tức thì việc truyền số liệu theo chu kỳ là không cần thiết và có thể làm tiêu tốn năng lượng vô ích. Giao thức LEACH cần tiếp tục được cải tiến để khắc phục hạn chế này.

2.3. Hoạt động truyền nhận không dây.

Đối với mạng cảm nhận không dây thì năng lượng trở thành một vấn đề được chú ý để duy trì hoạt động của hệ thống mạng vì các nút mạng độc lập, chỉ được cung cấp năng lượng từ một nguồn cố định (pin). Việc tiêu thụ năng lượng thấp là một nhu cầu quan trọng cho các hệ thống hoạt động bằng pin – không được cung cấp năng lượng thường xuyên. Khi có một dòng tiêu thụ thấp thì thời gian sống của chúng sẽ kéo dài thêm. Hệ thống tiêu thụ năng lượng thấp là mục tiêu cần đạt của mạng cảm nhận không dây sử dụng CC1010. Các vấn đề đưa ra trong phần này là cơ sở cho việc viết phần mềm tiết kiệm tiêu thụ năng lượng cho nút mạng cảm nhận không dây.

2.3.1. Tần số làm việc của CC1010.

Tần số làm việc là một tham số quan trọng vì năng lượng tiêu thụ tăng tuyến tính với tần số làm việc trong hoạt động của mạch CMOS. Vì vậy, vấn đề quan trọng là không sử dụng tần số làm việc cao hơn mức cần thiết cho các ứng dụng.

Sử dụng CC1010 sẽ thuận lợi để tiêu thụ dòng điện thấp bởi vì bộ truyền nhận không dây tiêu thụ năng lượng rất ít. Lõi MCU của CC1010 cũng có nhiều đặc tính tiết kiệm điện năng. Chế độ đồng hồ trong CC1010 rất quan trọng cho tiết kiệm tiêu thụ năng lượng.

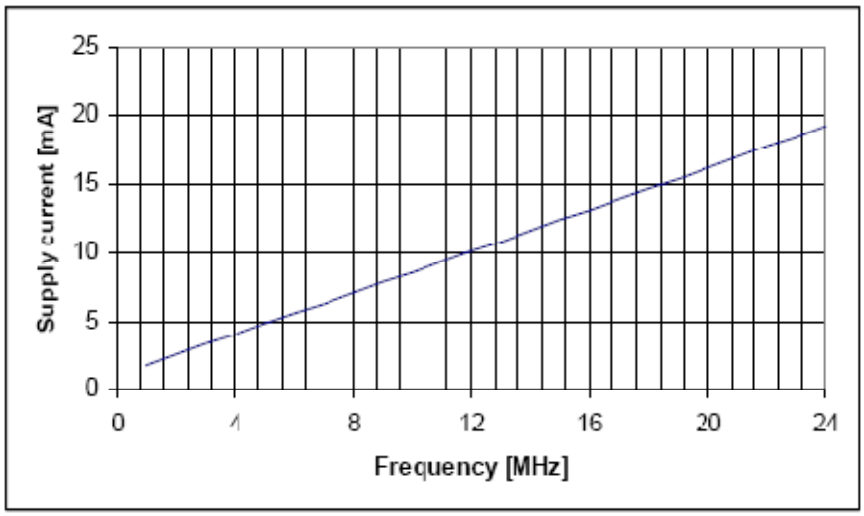
CC1010 với 2 máy tạo dao động tinh thể, một máy tạo dao động tần số cao được sử dụng tinh thể với tần số từ 3MHz đến 24MHz, và máy tạo dao động tần số thấp thiết kế để sử dụng một đồng hồ tinh thể 32KHz. CC1010 có thể chuyển đổi giữa 2 chế độ đồng hồ nguồn bằng cách ghi vào bit CMOS trên thanh ghi X32CON. Có thể mô tả thanh ghi X32CON một cách sơ lược như sau:

7	6	5	4	3	2	1	0
-	-	-	-	-	X32_BYPASS	X32_PD	CMODE

Cần quan tâm đến bit0 – CMODE của thanh ghi này. Khi chạy trên dao động 32kHz sẽ gọi tới mode 1 (CMODE = 1), chạy ở tốc độ dao động cao sẽ gọi tới mode 0 (CMODE = 0). Cả 2 dao động phải được cấp năng lượng và có sự ổn định trước khi chuyển đổi giữa chúng. Khi reset, CC1010 sẽ mặc định chạy ở dao động tần số cao.

Tốc độ truyền nhận dữ liệu có thể là: 0.6, 1.2, 2.4,…,76.8kBaud, tốc độ 76.8kBaud chỉ đạt được khi sử dụng tần số 14.7456MHz.

Hình sau minh hoạ cho quan hệ tăng tuyến tính giữa tần số làm việc và dòng điện tiêu thụ:



Hình 2.2: Mối quan hệ tuyến tính giữa dòng tiêu thụ và tần số.

Từ đồ thị ta thấy tần số làm việc tăng, dòng tiêu thụ cũng sẽ tăng. Do vậy, muốn tiết kiệm năng lượng cần phải chọn tần số hợp lý lúc nút mạng làm việc và không làm việc tương ứng. Việc chọn tần số làm việc hợp lý tức là chuyển đổi về tần số làm việc thấp khi nút mạng nghỉ hay làm việc ở tần số cao khi nút mạng cần truyền nhận thông tin để tránh trình trạng nút mạng làm việc ở tần số cao trong những khoảng thời gian không cần thiết.

2.3.2. Chế độ làm việc của CC1010.

Mạch tổ hợp có thể chuyển đổi giữa chế độ tích cực và chế độ tiết kiệm năng lượng. Dòng tiêu thụ trung bình phụ thuộc vào tỷ lệ giữa thời gian tiêu dùng trong chế độ tích cực và thời gian tiêu dùng trong chế độ tiết kiệm năng lượng.

Các thành phần trong CC1010 như CPU, bộ biến đổi ADC, bộ truyền nhận RF, v.v…trong đa số trường hợp không làm việc đồng thời. Nghĩa là chu kỳ nghỉ và chu kỳ hoạt động được thể hiện bởi tần số nhịp khác nhau. Quản trị chặt chẽ quá trình này sẽ tiết kiệm được năng lượng tiêu thụ của hệ thống. CC1010 có 3 chế độ làm việc:

- Chế độ tích cực (Active Mode).

Trong chế độ này 8051 chạy bình thường ở tần số cao của xung Clock và thực hiện các lệnh từ bộ nhớ Flash. Xung Clock là tần số dao động của tinh thể thạch anh chính. Dòng tiêu thụ phụ thuộc vào tần số thực tế được sử dụng nằm trong khoảng 3MHz đến 24MHz.

- Chế độ nghỉ (Idle Mode).

Chế độ nghỉ sẽ được bắt đầu sau khi thực hiện xong lệnh thiết lập bít PCON.IDLE. Trong chế độ này vi điều khiển 8051 dừng các quá trình xử lý và các thanh ghi sẽ lưu lại dữ liệu hiện tại, còn tất cả các thiết bị ngoại vi khác vẫn chạy bình thường. Có 3 cách để thoát khỏi chế độ này:

+ Kích hoạt ngắt. Việc này có nghĩa là xoá bít IDLE, kết thúc chế độ IDLE và thi hành ngắt.

+ Thiết lập lại: Tất cả các thanh ghi được nạp lại và chương trình sẽ lại tiếp tục từ địa chỉ 0x0000.

+ Tắt/bật nguồn.

- Chế độ tắt nguồn (Power – Down Mode).

Sau khi thực hiện xong lệnh đặt bit PCON.STOP vi điều khiển và các thiết bị ngoại vi sẽ ngừng hoạt động. Trong chế độ này, các thành phần ngoại vi và nhóm các đồng hồ của vi điều khiển 8051 sẽ bị vô hiệu hóa, chỉ có đồng hồ của bộ biến đổi ADC là vẫn hoạt động. Điều này sẽ cho phép bộ ADC có thể sinh ra được tín hiệu reset. Đây chính là chế độ tiết kiệm năng lượng nhất.

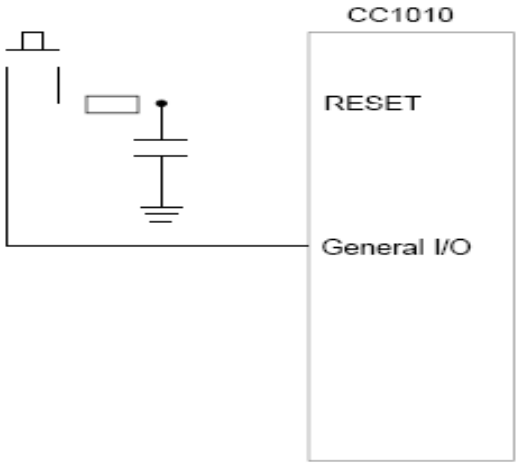
Có hai cách để kết thúc chế độ Power-Down Mode:

+ Thiết lập lại: Tất cả các thanh ghi sẽ được thiết lập lại và chương trình sẽ tiếp tục. Chương trình sẽ bắt đầu lại từ địa chỉ 0x0000.

+ Tắt/ bật nguồn.

Vấn đề quan trọng đặt ra là làm thế nào cho nút mạng có thể trở về chế độ nghỉ và thức dậy sau khi nghỉ một cách linh hoạt để tránh sự lãng phí, tăng thời gian sử dụng nguồn. Dựa vào các cách kết thúc chế độ tiết kiệm năng lượng đưa ra trên đây đã có một số cách đánh thức nút mạng được đưa ra:

- *Thức dậy nhờ việc nhấn nút bấm*, việc này có nghĩa là phải tạo ra một mạch ngoài, có thể mô tả như trong hình vẽ sau:



Hình 2.3 : Sử dụng nút bấm để đánh thức nút mạng.

Giải pháp này sẽ đưa nút mạng trở lại hoạt động nhờ một sự kiện bên ngoài – Đó là sự tác động của con người. Như vậy, sẽ làm mất đi tính độc lập vốn có của nút mạng cảm nhận không dây sử dụng CC1010.

- *Thức dậy theo khoảng thời gian:* Sử dụng đồng hồ thời gian thực (RTC) của CC1010. RTC có thể đánh thức CC1010 từ chế độ ngủ trong khoảng thời gian từ 1 đến 127 giây. Máy tạo dao động 32kHz phải hoạt động để RTC thực hiện chức năng này. Giải pháp này đem lại hiệu quả tiết kiệm tiêu thụ năng lượng cho nút mạng khi nó thức dậy từ chế độ nghỉ.

Với chế độ tắt nguồn, nếu ta đưa nút mạng về chế độ này thì khi muốn nút mạng thức dậy, cần có tác động của con người. Do vậy, trong chương trình sẽ không can thiệp tới chế độ tắt nguồn. Chế độ nghỉ của nút mạng được quan tâm vì có thể không cần đến sự tác động của con người nhưng vẫn tiết kiệm năng lượng nhờ khả năng thức dậy độc lập.

Trong các giải pháp làm cho nút mạng thức dậy sau khi nghỉ, giải pháp bật/tắt nguồn là dễ thực hiện nhất. Tuy nhiên, nó sẽ làm mất đi ưu điểm vốn có của mạng cảm nhận không dây là độc lập, ít cần đến sự tác động của con người. Vì vậy, vấn đề được quan tâm ở đây là chuyển đổi chế độ làm việc của nút mạng WSN thông qua các thanh ghi được lập trình trong thư viện do hãng Chipcon cung cấp cho CC1010.

2.3.3. Thiết lập chế độ làm việc bằng chương trình.

Trong vi điều khiển CC1010, chúng ta quan tâm đến thanh ghi điều khiển năng lượng – PCON (Power Control Rigister). Tìm hiểu về thanh ghi PCON trong tài liệu tham khảo [5]. Bảng dưới đây cho chúng ta biết về PCON với các bit của nó.

PCON (0x87) - Power Control Register

Bit	Name	R/W	Reset value	Description
7	SMOD0	R/W	0	Serial Port 0 baud rate doubler enable. 0 : Serial Port 0 baud rate is not doubled 1 : Serial Port 0 baud rate is doubled
6	-	R/W	0	Reserved
5	-	R1	1	Reserved, read as 1
4	-	R1	1	Reserved, read as 1
3	GF1	R/W	0	General purpose flag 1. Bit-addressable, general purpose flag for software control.
2	GF0	R/W	0	General purpose flag 0. Bit-addressable, general purpose flag for software control.
1	STOP	R/W	0	Power Down (Stop) mode select. Setting the STOP bit places CC1010 core and peripherals in Stop Mode.
0	IDLE	R/W	0	Idle mode select. Setting the IDLE bit places CC1010 in Idle Mode (core is stopped but peripherals are running).

PCON có địa chỉ là 87h và ta có thể biểu diễn thanh ghi này như sau:

7	6	5	4	3	2	1	0
SMOD0	-	-	-	GF1	GF0	STOP	IDLE

Bảng 2.1: Mô tả ý nghĩa các bit.

Bit	Tên bit	Giá trị	Ý nghĩa
7	SMOD0	0	Giá trị sau khi reset là 0, tốc độ baud giữ nguyên giá trị.
		1	Tốc độ baud được nhân đôi.
6	-	0	Bit dự trữ. Giá trị sau khi reset là 0
		1	
5	-	0	Bit dự trữ. Giá trị sau khi reset là 1
		1	
4	-	0	Bit dự trữ. Giá trị sau khi reset là 1
		1	
3	GF1	0	Giá trị sau khi reset là 0. Dùng cho điều khiển phần mềm, bit này được gọi là cờ mục đích chung 1.
		1	
2	GF0	0	Giá trị sau khi reset là 0. Dùng cho điều khiển phần mềm.
		1	
1	STOP	0	Giá trị sau khi reset là 0, nút mạng không ở chế độ dừng.
		1	Nút mạng ở chế độ dừng.
0	IDLE	0	Giá trị sau khi reset là 0, nút mạng ở chế độ tích cực.
		1	Nút mạng ở chế độ im lặng.

Với những đặc điểm của thanh ghi PCON, ta cần nắm bắt cơ chế hoạt động và tiến hành vận dụng được thanh ghi PCON vào việc thiết lập chế độ.

Một số giá trị ban đầu của thanh ghi PCON được đưa ra trong thư viện Hal.h do hãng Chipcon cung cấp như:

PCON|=0x80;

// Giá trị bit 7 là 1 tức tốc độ baud được nhân đôi.

PCON|=0x01;

// Giá trị bit 0 _ bit IDLE là 1 tức có thể chuyển đổi về chế độ nghỉ.

PCON|=0x02;

// Giá trị bit 1_ bit STOP là 1 tức có thể chuyển đổi về chế độ dừng(ngắt điện).

PCON là một thanh ghi quan trọng giúp cho việc đưa phần mềm nhúng tiết kiệm tiêu thụ năng lượng vào chương trình truyền số liệu cảm nhận của nút mạng. Trong đó, bit 0 và bit 1 của PCON là quan trọng nhất đối với khả năng chuyển đổi chế độ làm việc để tiết kiệm năng lượng. Việc chuyển nút mạng về chế độ ngắt điện mặc dù sẽ tiết kiệm năng lượng nhất nhưng khi cần trở về chế độ tích cực lại phụ thuộc vào sự tác động của con người. Do vậy, bit 0 là bit cần quan tâm nhất.

Việc thiết lập chế độ làm việc bằng chương trình sẽ được thực hiện thông qua sự chuyển đổi tần số làm việc và giá trị bit của thanh ghi PCON.

2.4. Kết luận.

Chương 2 đã đưa ra những vấn đề cơ bản liên quan đến tiết kiệm tiêu thụ năng lượng. Qua đó đưa ra những đánh giá về khả năng tiết kiệm năng lượng của một số yếu tố liên quan đến hệ thống mạng. Một trong các yếu tố

đã được lựa chọn để xây dựng phần mềm nhúng tiết kiệm tiêu thụ năng lượng là hoạt động truyền nhận không dây. Trong hoạt động truyền nhận không dây, tần số làm việc cao hay thấp, nút mạng nghỉ hay làm việc sẽ ảnh hưởng tới năng lượng sử dụng của nút mạng. Những nghiên cứu của chương 2 là định hướng cho việc thiết lập chế độ làm việc của nút mạng. Ta dựa vào sự chuyển đổi chế độ làm việc với các tần số làm việc tương ứng của nút mạng không dây sử dụng CC1010 để đưa ra tư tưởng thuật toán.

CHƯƠNG III
PHẦN MỀM NHÚNG.

3.1. Tổng quan về phần mềm nhúng.

Phần mềm nhúng đang có những bước đột phá mới, tạo ra những cuộc cách mạng triệt để trong tương lai nhằm phục vụ nhu cầu ngày càng cao của con người. Cơ sở cho sự phát triển này là những bước tiến mạnh mẽ trong công nghệ phần cứng. Một phần mềm nhúng phải kết hợp chặt chẽ với môi trường của nó bao gồm phần cứng và các hệ thống liên quan. Nó có những ràng buộc về tốc độ xử lý, dung lượng bộ nhớ và mức tiêu thụ điện năng... Một phần mềm nhúng tốt là phần mềm phải đảm bảo các yếu tố trên và đó cũng là hướng phát triển quan trọng của các phần mềm nhúng. Điểm mấu chốt của các phần mềm nhúng ngày nay là việc lựa chọn các phương pháp thực thi của một chức năng giống như một thành phần phần cứng nhưng theo một cách riêng. Vì vậy mà không thể bỏ đi các tính năng "cứng" của phần mềm như các phần mềm truyền thống khác. Một phần mềm nhúng ngày nay được phát triển theo cách sau:

- Liên kết phần mềm nhúng từ dưới lên trên, từ các lớp trừu tượng đến các chức năng hệ thống.
- Liên kết phần mềm nhúng với các nền lập trình được - các nền hỗ trợ nó cung cấp các phương tiện cần thiết để đánh giá xem các ràng buộc đưa ra có thỏa mãn hay không.

Để làm được như vậy thì thực chất cần phải phát triển các kỹ thuật hình thức ở mức trừu tượng để có những đánh giá sớm cùng với các nhóm công cụ và phương pháp đúng đắn. Mặt khác cũng cần phải xem xét phần mềm nhúng và kiến trúc phần cứng của nó trong một tổng thể cân đối. Do phải thỏa mãn nhiều yếu tố khác nhau về phần cứng, môi trường, giá thành, hiệu năng nên tồn tại nhiều thách thức trong phát triển phần mềm nhúng

ngày nay như: Cần có khả năng tái sử dụng cao, có thể đồng thiết kế phần cứng, phần mềm, xây dựng mô hình các thuộc tính phi chức năng, chuyển đổi các phần mềm thành các dịch vụ thông qua các thành phần phần mềm, kiến trúc hệ thống và kiến trúc phần mềm, đánh giá và kiểm định mức hệ thống, xây dựng các hệ thống có khả năng tổ hợp được nhờ các thành phần phần mềm có thể tái sử dụng.

Để viết phần mềm nhúng cho các họ vi xử lý, chúng ta có thể sử dụng các ngôn ngữ khác nhau như C/C++ hoặc Assembler. Tuỳ theo tiêu chí xây dựng hệ thống mà lựa chọn ngôn ngữ thích hợp. Từ đó cũng chọn chương trình dịch thích hợp. Ngày nay, do nhu cầu phát triển hệ thống nhanh, bảo trì dễ dàng nên ngôn ngữ được lựa chọn thường là ngôn ngữ cấp cao như C/C++.

Một số bước cơ bản để xây dựng phần mềm ứng dụng: Trước hết cần tìm hiểu bài toán, phân tích chi tiết để đưa ra hướng đi, tìm hiểu các yêu cầu của bài toán nhằm thiết kế lưu đồ thuật toán, dựa vào phần thiết kế để viết chương trình ứng dụng sau đó đưa vào kiểm thử để xác định hiệu quả mà chương trình đạt được.

Việc xây dựng phần mềm nhúng cũng tuân theo trình tự các bước như trên. Ngoài ra, phần mềm nhúng còn có đặc trưng là làm việc trực tiếp với phần cứng. Do đó để kiểm soát quá trình làm việc với các thành phần chấp hành có đúng đắn hay không là điều đặc biệt quan trọng

3.2. Phần mềm nhúng cho nút mạng WSN sử dụng CC1010.

3.2.1. Công cụ.

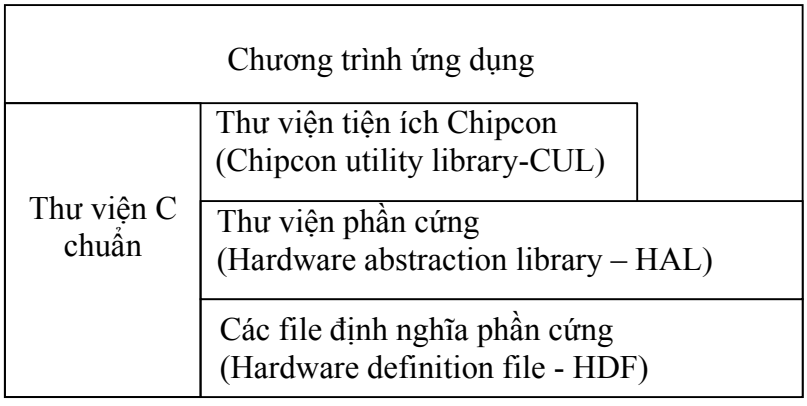
Phần mềm nhúng viết cho CC1010 được viết bằng ngôn ngữ C, sử dụng các thư viện cho CC1010 do hãng Chipcon cung cấp và chương trình biên dịch Keil uVision 2.0.

Chương trình dịch Keil uVision 2.0 do hãng Keil Elektronik GmbH xây dựng là một môi trường phát triển tích hợp (IDE) dùng để xây dựng các chương trình cho các họ VĐK tương thích 8051 của Intel. Đây là bộ chương trình dịch cho phép người viết chương trình soạn thảo chương trình, dịch chương trình và gỡ lỗi trên cùng một môi trường. Chương trình dịch hỗ trợ cho cả ngôn ngữ C và Assembly.

Hãng Chipcon cũng cung cấp bộ thư viện CC1010IDE hỗ trợ cho việc xây dựng phần mềm cho VĐK CC1010. Đây là bộ thư viện giúp cho việc xây dựng chương trình cho CC1010 được dễ dàng và nhanh chóng.

CC1010IDE dựa trên công cụ phát triển "uVision2" của hãng Keil ™ Elektronik GmbH. Công cụ này cung cấp một khung (framework) cho hầu hết các đặc điểm của CC1010IDE và cũng hỗ trợ hầu hết cho các VĐK họ 8051. Trình soạn thảo là một công cụ chủ yếu để soạn thảo các file nguồn và file hợp ngữ. Một điểm đặc biệt của bộ dịch là có thể chuyển các file nguồn được viết bằng C sang dạng hợp ngữ, để sau đó có thể tối ưu hoá mã lệnh. Dạng hợp ngữ sau đó được chuyển thành các file đối tượng (mã máy hoặc dữ liệu nhị phân). Cuối cùng, bộ liên kết đưa ra dạng file thực thi dạng Intel HEX và có thể nạp vào bộ nhớ Flash của VĐK.

Mô hình của một phần mềm nhúng viết cho CC1010 như sau:



Hình 3.1: Mô hình phần mềm nhúng cho CC1010.

Các file định nghĩa phần cứng - Hardware Definition Files (HDF)

Các file định nghĩa phần cứng định nghĩa địa chỉ các thanh ghi, ánh xạ vectơ ngắt và các hằng số phần cứng khác. Chúng cũng thường dùng các macro cho CC1010EB, và các định nghĩa hỗ trợ hợp ngữ và ngôn ngữ C.

Thư viện phần cứng - Hardware Abstraction Library (HAL)

Để hỗ trợ việc phát triển chương trình nhanh chóng và dễ dàng, Chipcon cung cấp thư viện các macro và các hàm truy cập phần cứng C1010 dễ dàng. Những thư viện này nằm trong Thư Viện Phần Cứng (HAL) và thi hành một giao tiếp phần cứng trừu tượng đối với chương trình người dùng. Nhờ đó chương trình người dùng có thể truy cập ngoại vi của vi điều khiển, thông qua các lời gọi hàm/macro, mà không cần hiểu chi tiết về phần cứng. Thư viện HAL hỗ trợ các chức năng sau:

- *Truyền nhận không dây*
- *Đo cường độ RSSI*
- *Truyền nhận RS232*
- *Làm việc với ADC*
- *Xử lý thời gian thực*
- *Mã hoá DES*
- *Thiết lập các bộ định thời*
- *Làm việc với các cổng*

Thư viện tiện ích Chipcon - Chipcon Utility Library (CUL)

Bên cạnh module HAL CC1010IDE cũng cung cấp một thư viện cho truyền thông RF đặt trong Thư Viện Tiện Ích (CUL). Thư viện này thường dùng cho các ứng dụng RF điển hình, cung cấp một giao thức RF đầy đủ. Thư viện CUL hỗ trợ các chức năng sau:

- *Truyền nhận không dây*
- *Tính toán Mã dư vòng (CRC)*
- *Xử lý Thời gian thực (Realtime Clock)*

Cả hai thư viện HAL và CUL đều hỗ trợ Truyền nhận không dây và xử lý thời gian thực. Tuy nhiên, các hàm ở thư viện CUL làm việc ở mức cao hơn, người viết chương trình cũng dễ dàng và tiện lợi hơn, nhưng bù lại cũng kém mềm dẻo hơn so với sử dụng các hàm ở thư viện HAL. Do vậy, đối với những ứng dụng đòi hỏi sự phức tạp thì thường dùng thư viện HAL.

Với những công cụ do Chipcon cung cấp, việc giải quyết các bài toán liên quan đến nút mạng và toàn bộ hệ thống mạng WSN trở nên dễ dàng hơn. Sử dụng các công cụ này linh hoạt sẽ tạo ra các phần mềm có tính linh hoạt và mềm dẻo. Đây là bước mở đầu cho việc triển khai rộng rãi mạng WSN với nhiều ứng dụng có ý nghĩa thiết thực trong tương lai.

3.2.2. Phần mềm nhúng.

a. Nghiên cứu, phân tích phần mềm nhúng truyền nhiệt điển hình của nút mạng WSN.

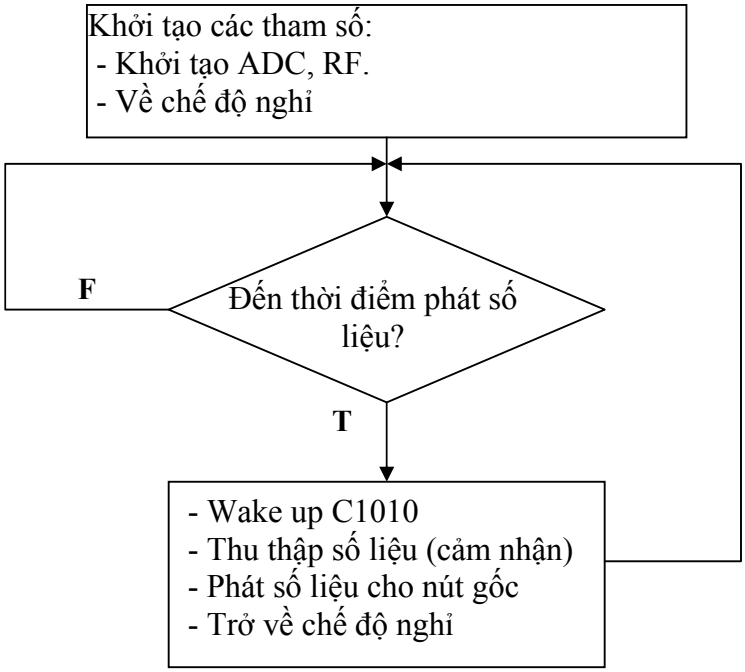
Qua các cách thức tiết kiệm năng lượng đã được tìm hiểu, đề tài đi sâu vào nghiên cứu chương trình truyền nhiệt để tác động vào chế độ làm việc của CC1010 nhằm tạo ra chương trình nhúng tiết kiệm tiêu thụ năng lượng. Chương trình truyền nhiệt được chọn làm cơ sở để thực hiện tiết kiệm năng lượng vì chương trình này đã được hãng Chipcon cung cấp. Nghiên cứu chương trình truyền nhiệt ta nhận thấy, trong đó chưa thực hiện việc chuyển đổi chế độ làm việc cho CC1010. Vì vậy, ta cần đưa thêm hàm thực hiện chức năng này mới có thể tiết kiệm tiêu thụ năng lượng.

Ngoài việc sử dụng chương trình truyền nhiệt, ta có thể sử dụng một số chương trình do Chipcon cung cấp như: powermodes, clockmodes,

xosc_switching để hiểu rõ về các chế độ năng lượng, chế độ đồng hồ, sự chuyển đổi giữa tần số cao (3 – 24MHz) của xung clock và tần số thấp 32kHz của đồng hồ tinh thể. Qua đó thực hiện việc chuyển đổi chế độ làm việc trong chương trình truyền nhiệt tốt hơn.

Tuy nhiên, do vi điều khiển bị ràng buộc về mặt tài nguyên nên đòi hỏi chương trình đưa vào vi điều khiển phải ngắn gọn, tiết kiệm bộ nhớ song vẫn đảm bảo cho việc viết chương trình nhanh, bảo trì và nâng cấp dễ dàng.

Thuật toán của chương trình có thể mô tả như sau:



Hình 3.2: Thuật toán làm việc của nút mạng cảm nhận.

Ý nghĩa của các bước trong lưu đồ thuật toán:

Bước 1: Khởi tạo:

- Khởi tạo ADC:

  + Đặt bộ biến đổi ADC về chế độ đơn (10 bit).

  + Đặt điện áp tham chiếu là 1.25V.

- Khởi tạo RF:

  + Thiết lập một trong các tần số RF: 433MHz, 868MHz, 915MHz.

  + Cách điều chế tín hiệu: mã hoá Manchester

  + Công suất phát: 4 dBm

  + Xác định tốc độ truyền dữ liệu: 2.4kb/s

- Về chế độ nghỉ:

  + Thiết lập giá trị của bit PCON.IDLE = 01h.

Bước 2: Thời điểm phát số liệu là thời điểm xung nhịp của đồng hồ thời gian thực – RTC đã đếm được 15s sau khi nút mạng chuyển về chế độ nghỉ.

+ Nếu True : nút mạng sẽ chuyển sang trạng thái của bước 3.

+ Nếu False : nút mạng quay trở lại trạng thái nghỉ.

Bước 3:

- Wake up C1010: Nút mạng thức dậy và chuyển sang chế độ tích cực, giá trị của bit PCON.IDLE thay đổi.

- Thu thập số liệu (cảm nhận): Cảm biến sẽ thực hiện chức năng cảm nhận, thu thập thông tin, sau đó đưa tín hiệu ở dạng tương tự về cho vi điều khiển. Tại vi điều khiển, ADC sẽ chuyển tín hiệu sang dạng số rồi đọc giá trị vừa chuyển đổi.

- Phát số liệu cho nút gốc: Bộ thu phát RF bật TX để thực hiện truyền số liệu cảm nhận được về nút gốc.

- Trở về chế độ nghỉ: Thiết lập giá trị cho bit PCON.IDLE = 1

b. Phần mềm nhúng thực hiện thuật toán.

Chương trình thực hiện như sau:

- Khởi tạo ADC: dùng hàm halConfigADC() trong thư viện Hal với các giá trị khởi tạo như sau:

```
halConfigADC(ADC_MODE_SINGLE | ADC_REFERENCE_INTERNAL_1_25, CC1010EB_CLKFREQ, 0);
// ADC sẽ chuyển đổi dữ liệu theo chế độ single, không liên tục.
```

- Khởi tạo RF: Các giá trị được thiết lập theo cấu trúc sau:

```
RF_RXTXPAIR_SETTINGS code RF_SETTINGS =
{
   0xA3, 0x2F, 0x15,    // Modem 0, 1 and 2
   0x75, 0xA0, 0x00,    // Freq A
   0x58, 0x32, 0x8D,    // Freq B
   0x01, 0xAB,          // FSEP 1 and 0
   0x40,                // PLL_RX
   0x30,                // PLL_TX
   0x6C,                // CURRENT_RX
   0xF3,                // CURRENT_TX
   0x32,                // FREND
   0xFF,                // PA_POW – Năng lượng đầu ra
   0x00,                // MATCH
   0x00,                // PRESCALER  };
```

Chương trình sử dụng 3 cấu trúc này để thiết lập tần số RF ở 3 giá trị: 433MHz, 868MHz, 915MHz.

Lời gọi hàm: RFSetupTransmit(); để khởi tạo RF. Chương trình sử dụng hàm *halRFCalib(&RF_SETTINGS, &RF_CALDATA)* có trong thư viện Hal để chuẩn hoá RF. Trong hàm này cấu trúc *RF_SETTINGS* hỗ trợ thiết lập các thông số truyền nhận không dây, còn cấu trúc *RF_CALDATA* thể hiện kết quả trả về của lời gọi hàm.

- Cấu hình RTC để đánh thức nút mạng từ chế độ nghỉ:

halConfigRealTimeClock(15); // sau 15s, nút mạng sẽ thức dậy, giá trị này có thể thay đổi tuỳ theo ứng dụng thực tế.

RTC_RUN(TRUE); // Cho phép RTC làm việc để đếm thời gian.

- Sự chuyển đổi chế độ làm việc của nút mạng được thực hiện như sau:

Với *SelectClockMode(0); // nút mạng ở chế độ tích cực.*

XOSC_ENABLE(TRUE); // Làm việc ở tần số cao XOSC.

MAIN_CLOCK_SET_SOURCE(CLOCK_XOSC);// Thiết lập tần số làm việc ở tần số cao của xung clock

X32_ENABLE(FALSE); // Lúc này, nút mạng không làm việc ở tần số 32kHz.

PCON = PCON & 0xfe; // Giá trị PCON.IDLE=0.

Nếu nút mạng ở chế độ tích cực *SelectClockMode(0)*, chương trình thực hiện truyền dữ liệu:

GetParameters(); // Gọi tới hàm thu nhận số liệu cảm biến nhận được // từ môi trường.

halRFSetRxTxOff(RF_TX, &RF_SETTINGS,&RF_CALDATA);

//Bật TX, RF_SETTINGS hỗ trợ thiết lập các thông số truyền nhận không dây, RF_CALDATA thể hiện kết quả trả về của lời gọi hàm.

halRFSendPacket(PREAMBLE_BYTE_COUNT, txDataBuffer, TBC_DATA_LEN);

// Truyền cho nút gốc, dữ liệu lấy từ txDataBuffer – lưu dữ liệu cảm nhận đã chuyển đổi qua ADC.

halRFSetRxTxOff(RF_OFF,&RF_SETTINGS,&RF_CALDATA);//Tắt TX

tbcWait1sec(); // gọi hàm đợi.

Quá trình truyền sẽ tiếp tục, khi gặp: *bSample = 0;* thì mới kết thúc:

bSample = 0;

SelectClockMode(1); // trở về chế độ nghỉ.

Với *SelectClockMode(1); // nút mạng ở chế độ nghỉ.*

X32_INPUT_SOURCE(X32_USING_CRYSTAL);

X32_ENABLE(TRUE); // Đưa về tần số làm việc thấp -32kHz

halWait(250, CC1010EB_CLKFREQ); //chờ để tần số ổn định

halWait(250, CC1010EB_CLKFREQ);

MAIN_CLOCK_SET_SOURCE(CLOCK_X32);// Thiết lập tần số làm việc ở mức thấp của đồng hồ tinh thể 32kHz.

XOSC_ENABLE(FALSE); // Không làm việc ở tần số cao.

PCON = PCON | 0x01; // Giá trị PCON.IDLE=1

Khi nút mạng chuyển sang chế độ nghỉ chương trình sẽ thực hiện ngắt RTC để đếm thời gian nút mạng nghỉ. Sau 15s, chương trình sẽ tự động xoá cờ ngắt RTC để chuyển sang chế độ tích cực:

bSample = 1;

INT_SETFLAG(INUM_RTC, INT_CLR); // Xoá cờ ngắt RTC.

Khi gặp *bSample = 1;* nút mạng sẽ chuyển về chế độ tích cực và lại thực hiện truyền dữ liệu.

3.3. Kết luận.

Chương 3 đã đưa ra những kiến thức sơ lược về phần mềm nhúng và các phương tiện, cách thức thiết kế phần mềm nhúng cho nút mạng sử dụng CC1010 của WSN. Đồng thời cũng đưa ra thuật toán sử dụng trong chương trình nhúng và các hàm thực hiện chức năng cảm nhận, truyền thông của nút mạng. Xin xem phụ lục để biết thêm nội dung chi tiết của chương trình.

CHƯƠNG IV
MỘT SỐ THỬ NGHIỆM VÀ ĐÁNH GIÁ.

4.1. Thiết bị thử nghiệm.

Chương trình thử nghiệm sẽ được tiến hành trên một nút mạng cảm nhận. Nút mạng này sẽ thực hiện chức năng cảm nhận nhờ 1 cảm biến tương tự, sau đó sẽ thực hiện chức năng truyền. Khi nạp chương trình thử nghiệm cho nút mạng nó sẽ có thể chuyển đổi chế độ làm việc. Mục tiêu là đo được cường độ dòng điện tiêu thụ của nút mạng khi nó ở các chế độ khác nhau.

Các thiết bị khác cần thiết dùng trong thử nghiệm: một máy tính được cài đặt chương trình biên dịch và chương trình nạp cho nút mạng, một bản mạch được gắn với máy tính qua cổng nối tiếp, một nguồn pin 3.5V dùng cho nút mạng, một ampe kế và một số dây điện.

4.2. Thử nghiệm.

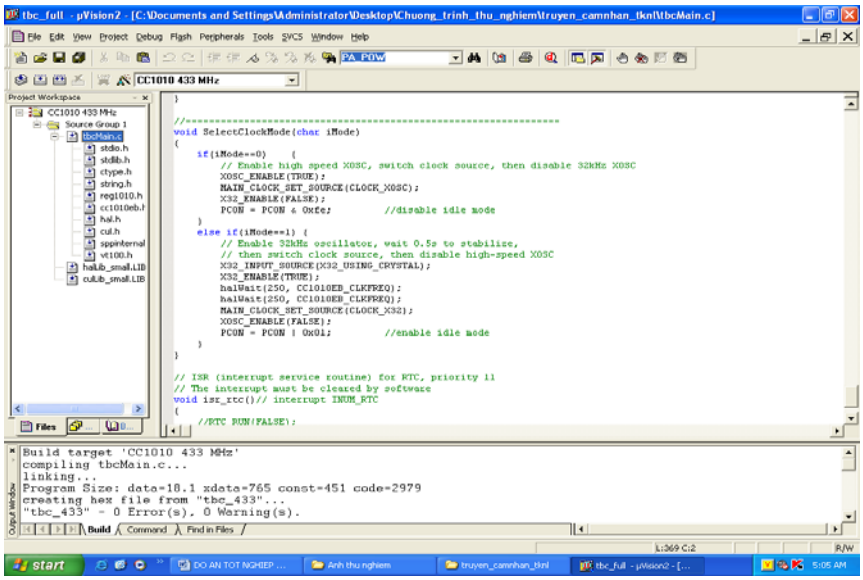
Bước 1: Nối bản mạch với PC. Chương trình nhúng sẽ được nạp cho nút mạng thông qua bản mạch này.

Bước 2: Gắn nút mạng vào bản mạch đã nối với PC. Xem hình 4.1 dưới đây cho các bước 1, 2.

Nút mạng

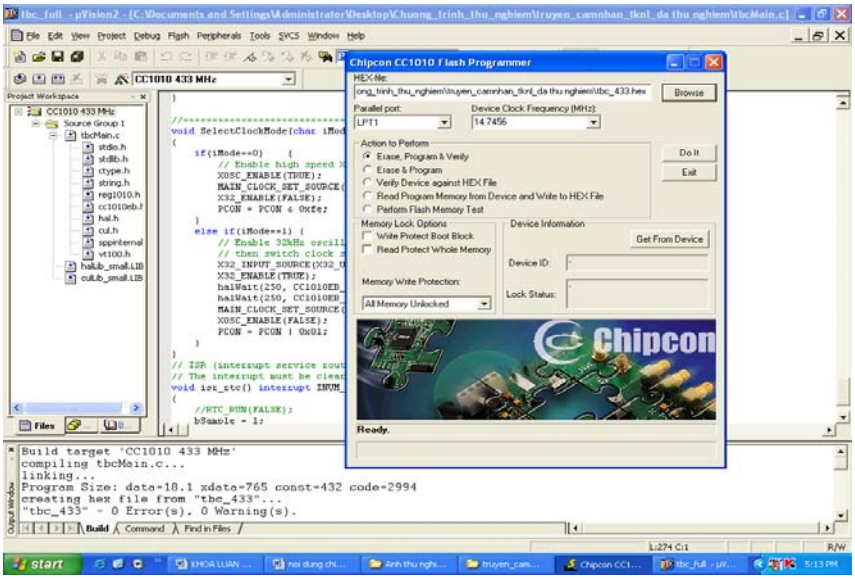
Hình 4.1: Gắn nút mạng vào bản mạch đã nối với hệ thống PC.

Bước 3: Dùng trình biên dịch Keil uVision 2.0 để dịch chương trình thử nghiệm trên PC. Xem hình 4.2 sau đây:



Hình 4.2: Dịch chương trình nhúng bằng Keil uVision 2.0.

Bước 4: Bật nguồn pin của bản mạch vừa gắn nút mạng, mở chương trình Chipcon CC1010 Flash Programmer để nạp tệp *.hex* vừa dịch cho nút mạng. Xem hình 4.3 minh hoạ cho bước này.



Hình 4.3: Nạp chương trình nhúng.

Bước 5: Tháo nút mạng ra khỏi bản mạch, gắn nó với pin 3.5V và tiến hành đo dòng tiêu thụ bằng ampe kế. Xem hình 4.4 và 4.5 để biết kết quả đo được.

Hình 4.4: Đo dòng điện mà nút mạng tiêu thụ trong chế độ nghỉ.

Hình 4.5: Đo dòng điện mà nút mạng tiêu thụ trong chế độ tích cực.

a. Khi nút mạng truyền. *b. Khi nút mạng cảm nhận.*

4.3. Kết quả đo được.

Bảng 1 cho kết quả đo với chương trình có tiết kiệm năng lượng nhờ chuyển đổi chế độ làm việc, tần số RF là 433MHz, kết quả thu được là:

Lần đo	Dòng điện tiêu thụ (mA)		
	Chế độ nghỉ	Cảm nhận	Truyền
1	0.2	23.6	17.9
2	0.2	23.6	17.8
3	0.1	21	18
4	0.2	23.5	17.8
5	0.1	22.8	19
Trung bình	0.16 ± 0.048	22.9± 0.8	18.1± 0.36

Bảng 1: Kết quả thử nghiệm chương trình nhúng.

Đánh giá kết quả:

Từ bảng kết quả trên ta nhận thấy, chương trình đã thực hiện được tiết kiệm năng lượng rất rõ ràng. Dòng tiêu thụ tại chế độ nghỉ chỉ bằng khoảng 1% dòng tiêu thụ tại chế độ tích cực. Vì vậy, nếu thời gian nút mạng ở trong chế độ nghỉ kéo dài sẽ tiết kiệm năng lượng rất nhiều. Trong chương trình này, thời gian nút mạng nghỉ được lấy là 15s. Tuy nhiên, tuỳ theo ứng dụng thực tế yêu cầu thường xuyên hay định kỳ cung cấp thông tin mà giá trị này có thể tăng lên hoặc giảm đi. Ta có thể nhận thấy, với những mạng chỉ cần cung cấp thông tin một cách định kỳ sẽ tốn ít năng lượng hơn. Căn cứ vào nhu cầu thực tế sử dụng ta có thể can thiệp vào thời gian nút mạng nghỉ để có thể tiết kiệm năng lượng nhất.

Với chương trình nhúng tiết kiệm tiêu thụ năng lượng nút mạng sẽ thay đổi chế độ liên tục vì vậy sẽ khó theo dõi kết quả đo. Để có thể thấy rõ hiệu quả tiết kiệm năng lượng, ta bỏ hàm chuyển đổi chế độ làm việc: *void SelectClockMode(char iMode)* và chức năng truyền dữ liệu về nút gốc, kết quả đo được khi mạng chỉ cảm nhận là:

Tần số RF	Dòng điện tiêu thụ (mA)					
	Lần 1	Lần 2	Lần 3	Lần 4	Lần 5	Trung bình
433MHz	21.2	21	21.1	21.2	21	21.1±0.1
915MHz	23.1	23	23.3	23.1	23	23.1±0.1

Bảng 2: Kết quả thử nghiệm khi không có tiết kiệm năng lượng.

Đánh giá kết quả:

Khi tần số truyền nhận tăng lên, dòng điện tiêu thụ cũng lớn hơn. So sánh cột giá trị dòng điện tiêu thụ khi cảm nhận ở bảng 1 với bảng 2 ta nhận thấy, cùng ở tần số 433MHz nhưng khi có chuyển đổi chế độ làm việc dòng điện tiêu thụ sẽ lớn hơn. Như vậy rõ ràng giữa các quá trình chuyển đổi chế độ làm việc cũng tiêu hao 1 phần năng lượng. Tuy nhiên, phần năng lượng do nó tiêu hao là không đáng kể so với phần năng lượng mà nó tiết kiệm được. Vì vậy, giải pháp chuyển đổi chế độ làm việc vẫn được coi là giải pháp tiết kiệm năng lượng.

4.4. Kết luận.

Qua kết quả thực nghiệm đã chứng minh được hiệu quả của việc tiết kiệm năng lượng cho nút mạng sử dụng vi điều khiển CC1010 thông qua chuyển đổi chế độ làm việc của nút mạng một cách hợp lý. Tuy nhiên, khi có điều kiện cần tiến hành truyền nhận trong một hệ thống mạng để có thể tiết kiệm năng lượng ở quy mô lớn hơn.

KẾT LUẬN.

Bản luận văn đã xem xét các đặc trưng chủ yếu của mạng cảm nhận không dây là: thời gian sống, độ bao phủ, chi phí và dễ triển khai, thời gian đáp ứng, độ chính xác về thời gian, bảo mật, và tốc độ lấy mẫu hiệu quả. Trong các đặc điểm đó, luận văn nghiên cứu vấn đề tiết kiệm năng lượng cho nút mạng cảm nhận không dây để kéo dài thời gian sống của nút mạng,

cụ thể là: tiết kiệm năng lượng dựa trên hoạt động truyền nhận không dây bằng phần mềm nhúng.

Việc nghiên cứu các chế độ và cách chuyển đổi giữa các chế độ của CC1010 thông qua tần số đã giúp cho việc thay đổi chế độ làm việc của nút mạng WSN một cách linh hoạt trong phần mềm nhúng. Việc chọn chế độ được thực hiện bằng phần mềm và khi có sự chuyển đổi chế độ làm việc hợp lý sẽ tiết kiệm năng lượng hơn.

Chương trình sau đó được thử nghiệm trên nút mạng cảm nhận và kết quả cho thấy: việc tiêu thụ dòng điện ở ba chế độ khác nhau rất nhiều. Dòng điện tiêu thụ ở chế độ nghỉ chỉ bằng khoảng 1% ở chế độ cảm nhận hoặc chế độ truyền. Nghĩa là năng lượng tiêu thụ trong chế độ nghỉ giảm khoảng 100 lần so với năng lượng tiêu thụ trong chế độ tích cực.

Từ nhận xét đó cho thấy: phân phối hợp lý các chế độ tích cực và nghỉ cho nút mạng sẽ làm tăng tuổi thọ của pin tức tăng tuổi thọ của nút mạng. Điều này có thể thực hiện được bằng phần mềm thông qua các lệnh chương trình mà luận văn đã chỉ ra.

Nếu mỗi nút mạng tiết kiệm năng lượng toàn bộ mạng WSN sẽ tiết kiệm được năng lượng nghĩa là nâng cao thời gian sống cho WSN. Đồng thời, khi năng lượng của nút mạng được duy trì lâu sẽ làm cho việc chọn đường nhanh chóng, dễ dàng hơn, tăng tốc độ của mạng.

Những nghiên cứu chung và kết quả thử nghiệm đã đạt được về tiết kiệm tiêu thụ năng lượng đã khẳng định khả năng tiết kiệm năng lượng nhờ hoạt động truyền nhận không dây mà quan trọng là tần số làm việc và sự chuyển đổi chế độ làm việc của nút mạng. Trong một phạm vi lớn hơn ta vẫn có thể áp dụng nó để tiết kiệm năng lượng cho một hệ thống WSN.

Hướng tiếp theo mà đề tài có thể thực hiện là tiến hành các thử nghiệm về tiết kiệm tiêu thụ năng lượng cho một hệ thống mạng. Mục tiêu này đạt được sẽ giúp WSN có khả năng triển khai rộng rãi với các ứng dụng thiết thực.

LỜI CẢM ƠN

Em xin chân thành cảm ơn Hội đồng quản trị trường Đại Học Dân Lập Hải Phòng đã tạo điều kiện cơ sở vật chất cho em học tập và nghiên

cứu trong những năm học vừa qua và điều kiện thực tập để làm đề tài tốt nghiệp.

Em xin gửi lời cảm ơn tới tất cả các thầy cô giáo trong Bộ Môn Tin Học đã giúp đỡ em trong những năm học vừa qua để có được những kiến thức cần thiết.

Em xin cảm ơn PGS.TS.Vương Đạo Vy là người đã tận tình giúp đỡ em tìm hiểu những vấn đề cơ bản nhất của đề tài.

Em xin chân thành cảm ơn!

Sinh viên

Đàm Thu Phương

TÀI LIỆU THAM KHẢO

[1] "*Mạng truyền số liệu"* – Vương Đạo Vy – Nhà xuất bản ĐHQG – Năm 2006.

[2] "*Cấu trúc dữ liệu và giải thuật"* – Đinh Mạnh Tường – Nhà xuất bản khoa học và kỹ thuật – Năm 2002.

[3] "*Mạng thông tin máy tính"* – Vũ Duy Lợi – Nhà xuất bản Thế giới – Năm 2002

[4] "*Power and Control in Networked Sensors"* – E. Jason Riedy, Robert Szewczyk – Năm 2000.

[5] Chipcon, CC1010_Data_Sheet, www.chipcon.com

[6] Chipcon, CC1010IDE_ User_Manual, www.chipcon.com

[7] Chipcon, CC1010_Examples_Readme, www.chipcon.com

[8]"Application Note AN017" – K.H.Torvmarrk – www.chipcon.com

PHỤ LỤC

Chương trình thử nghiệm vấn đề tiết kiệm năng lượng. Viết cho nút mạng cảm nhận với chức năng cảm nhận và truyền dữ liệu.

#include<stdio.h>

```
#include<stdlib.h>
#include<ctype.h>
#include<string.h>
#include<chipcon/reg1010.h>
#include<chipcon/cc1010eb.h>
#include<chipcon/hal.h>
#include<chipcon/cul.h>
#include<chipcon/vt100.h>

// Temperature packet:
#define TBC_NODE_ID_LENGTH      2      // word
#define TBC_NODE_NAME_LENGTH    5//20
#define  TBC_TEMP_OFFSET                (TBC_NODE_ID_LENGTH  +
TBC_NODE_NAME_LENGTH)
#define TBC_TEMP_LENGTH         2
#define  TBC_TEMP_OFFSET1               (TBC_NODE_ID_LENGTH  +
TBC_NODE_NAME_LENGTH + TBC_TEMP_LENGTH)
#define TBC_TEMP_LENGTH1        2
#define   TBC_TEMP_OFFSET2                (TBC_TEMP_OFFSET1  +
TBC_TEMP_LENGTH1)
#define TBC_TEMP_LENGTH2        2
#define   TBC_TEMP_OFFSET3                (TBC_TEMP_OFFSET2  +
TBC_TEMP_LENGTH2)
#define TBC_TEMP_LENGTH3        2
#define   TBC_TEMP_OFFSET4                (TBC_TEMP_OFFSET3  +
TBC_TEMP_LENGTH3)
#define TBC_TEMP_LENGTH4        2
#define   TBC_TEMP_OFFSET5                (TBC_TEMP_OFFSET4  +
TBC_TEMP_LENGTH4)
```

```
#define TBC_TEMP_LENGTH5       2
#define   TBC_DATA_LEN                    (TBC_TEMP_OFFSET5  +
TBC_TEMP_LENGTH5)
#define PREAMBLE_BYTE_COUNT              18
// Radio related:
#define TBC_MY_SPP_ADDRESS     2
#define TBC_RX_INTERVAL        50

// Node registration
#define TBC_INVALID_NODE_INDEX  255
#define TBC_UNUSED_NODE_ID     0x0000
#define MAJOR_PERIOD       1500//30000:5minutes,   360000:1    hour,
1500: 15sec
#define MINOR_PERIOD       100
#define AVG_COUNT          10

// Calibration data
RF_RXTXPAIR_CALDATA xdata RF_CALDATA;

// Speed related
byte xdata waitMultiplier;

// The temperature "table":
#define TBC_MAX_NODE_COUNT     16
word xdata nodeIDs[TBC_MAX_NODE_COUNT];
byte                                                    xdata
nodeNames[TBC_MAX_NODE_COUNT][TBC_NODE_NAME_LENGT
H];
word xdata nodeLastT[TBC_MAX_NODE_COUNT];
```

```
bit initrunflag = 1;
unsigned long xdata TMajorPeriod;

byte xdata rxDataBuffer[TBC_DATA_LEN];
byte xdata txDataBuffer[TBC_DATA_LEN];
byte xdata avgcnt;
bool xdata bSample;
// Function prototypes
void tbcWait1sec (void);
void GetParameters (void);
void setupTimer0();
void RFSetupTransmit (void);
void SelectClockMode(char iMode);
// Unit name, stored in Flash
byte code flashUnitName[TBC_NODE_NAME_LENGTH];

// RAM buffer for Flash copy
byte xdata ramBufNonAligned[128];
byte xdata received_byte;
word  xdata counter,counter2;

//-------------------------------------------------------------------------
//       MAIN PROGRAM
//-------------------------------------------------------------------------
void main (void)
{
   byte xdata n;

#ifdef FREQ868
```

```
// X-tal frequency: 14.745600 MHz
// RF frequency A: 868.277200 MHz  Rx
// RF frequency B: 868.277200 MHz  Tx
// RF output power: 4 dBm

RF_RXTXPAIR_SETTINGS code RF_SETTINGS =
  {
   0xA3, 0x2F, 0x15,    // Modem 0, 1 and 2
   0x75, 0xA0, 0x00,    // Freq A
   0x58, 0x32, 0x8D,    // Freq B
   0x01, 0xAB,          // FSEP 1 and 0
   0x40,                // PLL_RX
   0x30,                // PLL_TX
   0x6C,                // CURRENT_RX
   0xF3,                // CURRENT_TX
   0x32,                // FREND
   0xFF,                // PA_POW
   0x00,                // MATCH
   0x00,                // PRESCALER
 };

#endif

#ifdef FREQ915

// X-tal frequency: 14.745600 MHz
// RF frequency A: 915.027455 MHz  Rx
// RF frequency B: 915.027455 MHz  Tx
```

```
// RF output power: 4 dBm

RF_RXTXPAIR_SETTINGS code RF_SETTINGS = {
   0xA3, 0x2F, 0x15,    // Modem 0, 1 and 2
   0xAA, 0x80, 0x00,    // Freq A
   0x5C, 0xF4, 0x02,    // Freq B
   0x01, 0xAB,          // FSEP 1 and 0
   0x58,                // PLL_RX
   0x30,                // PLL_TX
   0x6C,                // CURRENT_RX
   0xF3,                // CURRENT_TX
   0x32,                // FREND
   0xFF,                // PA_POW
   0x00,                // MATCH
   0x00,                // PRESCALER
   };
#endif

#ifdef FREQ433

// X-tal frequency: 14.745600 MHz
// RF frequency A: 433.302000 MHz  Rx
// RF frequency B: 433.302000 MHz  Tx
// RF output power: 4 dBm

RF_RXTXPAIR_SETTINGS code RF_SETTINGS = {
   0xA3, 0x2F, 0x0E,    // Modem 0, 1 and 2
   0x58, 0x00, 0x00,    // Freq A
   0x41, 0xFC, 0x9C,    // Freq B
```

```
   0x02, 0x80,        // FSEP 1 and 0
   0x60,              // PLL_RX
   0x48,              // PLL_TX
   0x44,              // CURRENT_RX
   0x81,              // CURRENT_TX
   0x0A,              // FREND
   0xFF,              // PA_POW
   0xC0,              // MATCH
   0x00,              // PRESCALER
   };
#endif

   //=================================================

   // Initialize peripherals
   WDT_ENABLE(FALSE);
   RLED_OE(TRUE);        YLED_OE(TRUE);        GLED_OE(TRUE);
BLED_OE(TRUE);

   // Startup macros for speed and low power consumption
   MEM_NO_WAIT_STATES();
    FLASH_SET_POWER_MODE(FLASH_STANDBY_BETWEEN_RE
ADS);

   // ADC setup
    halConfigADC(ADC_MODE_SINGLE                                |
ADC_REFERENCE_INTERNAL_1_25, CC1010EB_CLKFREQ, 0);

    RFSetupTransmit();
```

```
    // Reset the node IDs
    for (n = 0; n < TBC_MAX_NODE_COUNT; n++)
    {
        nodeIDs[n] = TBC_UNUSED_NODE_ID;
    }

    // Reset our name buffer
    for (n = 0; n < TBC_NODE_NAME_LENGTH; n++)
    {
        nodeNames[0][n] = 0x00;
    }

    // Load name from Flash
    memcpy(&nodeNames[0][0],flashUnitName,TBC_NODE_NAME_LE
NGTH);

    nodeIDs[0] = TBC_MY_SPP_ADDRESS;

    // Prepare the id+name part of the packet
    txDataBuffer[0] = (nodeIDs[0] >> 8) & 0xFF;
    txDataBuffer[1] = nodeIDs[0] & 0xFF;

    for (n = 0; n < TBC_NODE_NAME_LENGTH; n++)
    {
        txDataBuffer[n + TBC_NODE_ID_LENGTH] = nodeNames[0][n];
    }

// Configure realtime clock to support wake-up from IDLE
```

```
        halConfigRealTimeClock(15);

// Enable realtime clock (implicit: interrupt enable)
        RTC_RUN(TRUE);

        SelectClockMode(1);
// Loop forever
   while (TRUE)
     {
      if(bSample)
        {
        SelectClockMode(0);
        BLED = LED_ON;
        GetParameters();
        halRFSetRxTxOff(RF_TX, &RF_SETTINGS, &RF_CALDATA);
        halRFSendPacket(PREAMBLE_BYTE_COUNT,      txDataBuffer,
TBC_DATA_LEN);
        halRFSetRxTxOff(RF_OFF, &RF_SETTINGS, &RF_CALDATA);
        GLED = !GLED;
        tbcWait1sec();

        halRFSetRxTxOff(RF_TX, &RF_SETTINGS, &RF_CALDATA);
        halRFSendPacket(PREAMBLE_BYTE_COUNT,      txDataBuffer,
TBC_DATA_LEN);
        halRFSetRxTxOff(RF_OFF, &RF_SETTINGS, &RF_CALDATA);
        GLED = !GLED;

        bSample = 0;
        BLED = LED_OFF;
```

```
        SelectClockMode(1);
        }
    }
}//main

void tbcWait1sec (void)
{
        halWait (250, CC1010EB_CLKFREQ);
        halWait (250, CC1010EB_CLKFREQ);
        halWait (250, CC1010EB_CLKFREQ);
        halWait (250, CC1010EB_CLKFREQ);
} // tbcWait1sec

void GetParameters(void)
{
    word xdata temp, temp1, temp2, temp3, temp4, p, i;
    temp = 0;temp1 = 0;   temp2 = 0;   temp3 = 0;   temp4 = 0;

// Indicate transmission
        for(i=0;i<4;i++)
        {
                TIMER2_RUN(TRUE);
                halWait (1, CC1010EB_CLKFREQ);
                TIMER2_RUN(FALSE);
                if(i==0) p = temp3;
                else p = (p+temp3)/2;
                tbcWait1sec();
        }
```

```
ADC_POWER(TRUE);
// Power up the ADC and sample the temperature
ADC_SELECT_INPUT(ADC_INPUT_AD0);
ADC_SAMPLE_SINGLE();
temp = ADC_GET_SAMPLE_10BIT();

ADC_SELECT_INPUT(ADC_INPUT_AD1);
ADC_SAMPLE_SINGLE();
temp1 = ADC_GET_SAMPLE_10BIT();

ADC_SELECT_INPUT(ADC_INPUT_AD2);
ADC_SAMPLE_SINGLE();
temp2 = ADC_GET_SAMPLE_10BIT();
ADC_POWER(FALSE);

// Update the TX buffer and the table with the new temperature
txDataBuffer[TBC_TEMP_OFFSET] = (temp >> 8) & 0xFF;
txDataBuffer[TBC_TEMP_OFFSET + 1] = temp & 0xFF;

txDataBuffer[TBC_TEMP_OFFSET1] = (temp1 >> 8) & 0xFF;
txDataBuffer[TBC_TEMP_OFFSET1 + 1] = temp1 & 0xFF;

txDataBuffer[TBC_TEMP_OFFSET2] = (temp2 >> 8) & 0xFF;
txDataBuffer[TBC_TEMP_OFFSET2 + 1] = temp2 & 0xFF;

txDataBuffer[TBC_TEMP_OFFSET3] = (temp3 >> 8) & 0xFF;
txDataBuffer[TBC_TEMP_OFFSET3 + 1] = temp3 & 0xFF;

} // GetParameters
```

```
void setupTimer0()
{
        TMajorPeriod = 0;
        bSample = 0;
        TH0 = 100;
        TL0 = 100;
        TMOD = TMOD | 0x1;    //timer0 mode1-16 bit timer
        INT_ENABLE(INUM_TIMER0, INT_ON);
        INT_GLOBAL_ENABLE (INT_ON);
        CKCON = CKCON & 0xf7;
        IE = IE|0x80;
        TF0 = 1;
}

void FlashIntrHandler(void) interrupt INUM_FLASH
{
        INT_SETFLAG(INUM_FLASH, INT_CLR);
        return;
}

//timer 10ms
void TIMER0_ISR() interrupt INUM_TIMER0
{
        TF0 = 0;
        TH0 = 0xd0;
        TL0 = 0;
        TR0 = 1;
```

```
        if(TMajorPeriod==0)
        {
                bSample = 1;
                TMajorPeriod = MAJOR_PERIOD+1;
        }

        TMajorPeriod--;
}

// Setup RF for RX
void RFSetupTransmit (void)
{
        halRFCalib(&RF_SETTINGS, &RF_CALDATA);

// Turn on RF for TX
        halRFSetRxTxOff(RF_TX, &RF_SETTINGS, &RF_CALDATA);
        INT_ENABLE(INUM_RF, INT_OFF);

// Select RF bytemode
        RFCON |= 0x01;

// Enable RF interrupt based on bytemode
        RF_SET_BYTEMODE();

// Setup preamble configuration
        RF_SET_PREAMBLE_COUNT(16);
        RF_SET_SYNC_BYTE(RF_SUITABLE_SYNC_BYTE);

// Make sure avg filter is free-running + 22 baud settling time
```

```
        MODEM1=(MODEM1&0x03)|0x24;

// Reset preamble detection
        PDET &= ~0x80;
        PDET |= 0x80;
}

//=====================================================

void SelectClockMode(char iMode)
{
        if(iMode==0)
        {
                // Enable high speed XOSC, switch clock source, then disable
32kHz XOSC
                XOSC_ENABLE(TRUE);
                MAIN_CLOCK_SET_SOURCE(CLOCK_XOSC);
                X32_ENABLE(FALSE);
                PCON = PCON & 0xfe;                    //disable idle mode
        }
        else if(iMode==1) {
                // Enable 32kHz oscillator, wait 0.5s to stabilize,
                // then switch clock source, then disable high-speed XOSC
                X32_INPUT_SOURCE(X32_USING_CRYSTAL);
                X32_ENABLE(TRUE);
                halWait(250, CC1010EB_CLKFREQ);
                halWait(250, CC1010EB_CLKFREQ);
                MAIN_CLOCK_SET_SOURCE(CLOCK_X32);
                XOSC_ENABLE(FALSE);
```

```
                PCON = PCON | 0x01;                    //enable idle mode
        }
}

// ISR (interrupt service routine) for RTC,
// The interrupt must be cleared by software
void isr_rtc()// interrupt INUM_RTC
{
        bSample = 1;
        INT_SETFLAG(INUM_RTC, INT_CLR);
        RLED = !RLED;
}
```