# Cortex-M4를 이용한 인터럽트 기반 게이트웨이 구현†

성철제, 김창화*
강릉원주대학교 컴퓨터공학과
e-mail : scj0325@hanmail.net, kch@gwnu.ac.kr

# Interrupt-based Gateway Implementation Using Cortex-M4†

Cheol-Je Seong, Changhwa Kim*
Dept. Computer Science and Engineering,
Gangneung-Wonju National University

요 약

센서네트워크에서 가장 큰 화두 중의 하나는 무선 배터리를 사용하는 전력소모문제이다. 배터리 전력소모문제를 해결하기위해 많은 노력을 하고 있다.

본 논문은 Polling 방법보다 배터리 전력소모가 적은 인터럽트 기반의 방법을 선택하여 구현하고, 구현한 알고리즘을 소개한다.

본 논문에서는 소개한 알고리즘은 Sleep상태의 게이트웨이가 데이터를 송수신할 때만 Awake상태로 바뀌어서 배터리 전력소모를 줄이고, 인터럽트 루틴을 통하여 센서노드와 게이트웨이, 게이트웨이와 서버간의 양방향 통신을 제공한다.

## 1. 서론

현대인들은 의식하지 못하지만 많은 곳에서 센서네트워크 기술은 사용되고 있다. 그 예로 센서네트워크는 산불감지나 해양환경, 자동차 위험감지, 병원 등 사람이 접근할 수 없는 위험한 환경에서 주로 사용된다.

요즘 각광받는 분야 중 하나인 IoT(Internet of Things)에서 센서네트워크는 IoT의 한 부분에 포함이 되어있고, 노드간의 의사소통으로 센싱, 통신, 처리를 통하여 상호작용을 한다[1]. 하나의 노드는 센싱한 값을 전송을 통해 다른 노드로 전달하고 전달받은 노드는 전달받은 값을 토대로 상황을 판별하고, 그에 따른 처리를 하거나 또 다른 노드로 전송을 하게 된다. 이 노드들 간의 의사소통 과정에서 노드의 배터리 전력을 소모하게 된다. 배터리 전력소모 문제는 센서네트워크에서 큰 화두 중에 하나이다.

노드의 송수신방법은 프로그래밍 방법으로 두 가지로 나뉘는데 노드가 계속 상태를 체크하여 데이터를 송수신하는 Polling 방법과 Sleep 상태로 있다가 인터럽트가 걸릴 때 깨어나서 송수신하는 방법인 인터럽트 처리 방법이 있다. Polling 방법은 항상 Awake상태를 유지해야하므로 데이터를 송수신할 때 경우에만 Awake상태이고, 평상시에는 Sleep상태를 유지하는 인터럽트 처리 방법에 비해 배터리 전력소모가 심하다[2].

이와 같은 필요에 따라서 본 논문에서는 인터럽트 기반의 처리방법의 알고리즘을 제시한다. 인터럽트 기반의 방법은 Polling 방법에 비해 반응속도가 빨라 실시간성이 높고, 배터리 전력 소모를 줄일 수 있다[2],[3].

본 논문의 구성은 다음과 같다. 2장에서는 관련연구로 인터럽트 기반으로 구현된 센서네트워크와 풀어야할 문제점을 소개한다. 3장에서는 센서네트워크의 구성요소와 본 논문에서 구현에 사용된 구성요소를 소개하고, 4장에서는 본 논문에서 구현한 프로그램의 알고리즘을 제안한다. 5장에서는 본 논문의 의의를 살펴보고 향후 연구방향을 제시한다.

## 2. 관련연구

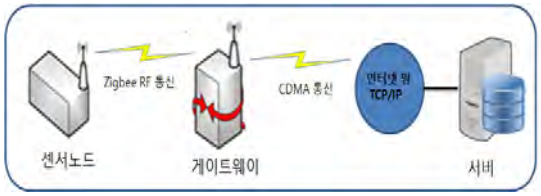
센서네트워크분야에서 인터럽트 처리에 대한 연구가 많이 이루어 졌다. 대표적으로 Tiny OS를 기반으로 NES C를 사용하여 인터럽트기반의 센서네트워크와 Polling방법을 이용한 센서네트워크의 실시간성을 비교한 연구가 있다[3]. 인터럽트기반의 센서네트워크와 Polling기반의 센서네트워크의 실시간성 비교는 두 방법을 사용하여 일정주기로 데이터를 송수신하는 방법으로 연구되었는데 연구결과는 Polling기반의 센서네트워크에 비해 인터럽트기반의 센서네트워크가 송수신하는 통신횟수의 주기가 더 짧다는 결과가 나왔다[3]. 이는 인터럽트기반의 센서네트워크가 실시간성이 높다는 것을 의미한다.

인터럽트 우선순위와 처리시간에 대한 연구도 이루어졌다. 우선순위에 따라 긴급한 우선순위의 처리와 배터리 전력소모문제, 인터럽트의 실시간 처리에 대한 연구의 필요성을 소개하고 있다[4].

† 본 연구는 미래창조과학부 및 정보통신기술진흥센터의 정보통신·방송 연구개발사업의 일환으로 수행하였음. [10041841, 연근해 수산양식(회유성 어종 등) 생장관리 핵심 요소기술 개발]

*교신저자

## 3. 센서네트워크 구성




(그림 1) 센서네트워크구성도

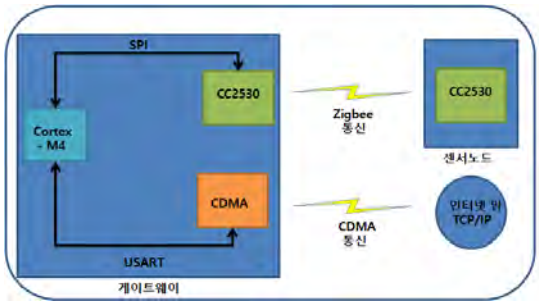
본 논문에서 구현한 센서네트워크의 네트워크 구성은 (그림 1)에서와 같이 다수 혹은 소수의 센서노드와 센서노드에서 센싱하여 얻은 값을 종합하는 싱크노드 즉, 게이트웨이, TCP/IP 서버로 이루어진다[5].

센서노드와 게이트웨이 사이에는 CC2530을 통한 지그비통신을 하고, 게이트웨이와 서버는 CDMA를 통하여 인터넷 망에 연결하여 통신한다. 게이트웨이는 서버로부터 받은 명령을 센서노드로 전하는 기능과 센서노드로부터 받은 데이터를 서버로 전달하는 기능을 모두 갖추고 있다.

### 3.1 게이트웨이 하드웨어 구성

게이트웨이는 Cortex-M4를 기반으로 MCU를 구성하고 있다. Cortex-M4는 32bit의 MCU로 고성능의 MCU이다[6]. 지그비는 CC2530을 사용하고 있으며 CC2530은 2.4-GHz영역에서 사용가능하고, 저전력으로 구동된다[7]. CDMA는 RCU-890을 사용하고 있다. RCU-890은 3G를 지원하는 CDMA로써 양방향 통신을 지원한다[8].

게이트웨이의 하드웨어는 (그림 2)에서와 같이 CC2530과 Cortex-M4 사이에서 SPI통신을 이용하고, CDMA와 Cortex-M4사이에서 USART통신을 이용한다.




(그림 2) 게이트웨이 하드웨어 구성도

### 3.2 게이트웨이 프로그램 구성

게이트웨이 프로그램은 센서노드로부터 데이터를 CC2530을 통한 SPI통신을 이용하여 수신을 한다. 본 논문에서는 외부 핀을 이용한 외부인터럽트를 이용하여 데이터를 수신한다. 또한 RCU-890을 이용한 CDMA통신은 USART 통신을 사용한다. CDMA통신은 USART 수신 인터럽트를 사용하여 데이터를 수신한다. 게이트웨이는 계속 Awake 상태에서 수신메시지를 체크하는 Polling방법과 인터럽트 처리방법이 있다. 센서네트워크의 주요 논제 중 하나는 실시간성과 전력소모를 절약하는 것이다. Polling방법을 이용하게 되면 게이트웨이로 데이터를 송수신하지 않을 경우에도 계속 Awake상태를 유지해야한다. 그러나 인터럽트 처리방법을 사용하면 데이터가 오지 수신하거나 송신하지 않을 경우에는 Sleep상태를 유지하다가 데이터를 송수신 할 때 Awake상태로 되어 통신을 한다. 결과적으로 인터럽트 처리방법은 Polling방법에 비해 배터리 소모전력을 절약할 수 있다. 구현된 인터럽트 처리방법의 프로그램 알고리즘은 4장에서 자세히 논한다.

## 4. 프로그램 구현

3.2절에서 소개한 구성을 바탕으로 구현이 되었다. CC2530의 SPI통신은 CC2530-ZNP의 SREQ를 이용하였다[9]. SREQ방법은 비동기식 방법인 AREQ와는 달리 동기식 방법이다. 즉 SREQ방법을 사용하면 AREQ와는 다르게 Cortex-M4와 CC2530간의 송수신에 대한 응답 즉, SRSP를 Cortex-M4가 수신하게 된다. RCU-890은 TCP/IP 명령을 이용하여 통신을 한다[7]. R-Socket과 M-Socket 중 M-Socket을 사용하였다. R-Socket은 데이터 포맷이 정해져 있어, 제약이 있는 반면에 M-Socket은 데이터 포맷에 대한 제약 없이 서버에서 송신한 데이터 그대로를 센서노드에게 전달할 수 있다. 또한 센서노드에서 송신한 데이터는 송신한 데이터 그대로를 서버에게 전달 할 수 있다.

### 4.1 센서노드로부터 게이트웨이의 데이터 수신

데이터 전송에서 CC2530-ZNP 프로토콜 중 동기식방법인 SREQ 프로토콜을 이용하여 구현하였다. SPI 수신 Handler는 CC2530에서 제공하는 Poll 프로토콜을 이용한다. Poll 프로토콜은 CC2530이 외부로부터 받은 데이터를 Cortex-M4로 전달해주기 위해 사용하는 프로토콜이다[9]. (그림 3)에 표현된 Gateway_SPI_HANDLER 알고리즘은 SPI 수신 Handler이다. 센서노드가 CC2530을 통해 데이터를 전송한다. 게이트웨이의 CC2530은 이 데이터를 수신한다. 데이터를 수신하면 인터럽트가 발생한다. 이 인터럽트를 처리하기 위해 Gateway_SPI_HANDLER를 게이트웨이의 Cortex-M4가 자동으로 수행한다. (그림 3)에 표현된 Gateway_SPI_HANDLER 함수는 다음과 같은 순서로

<표 1> SPI 수신 Handler에서 사용하는 함수

| 함 수 | 설 명 |
|---|---|
| SRDY_Pin_is_Low() | SRDY Pin이 Low상태이면 True를 반환하는 함수 |
| Send_Poll_Command_to_CC2530() | Cortex-M4가 CC2530으로 Poll메시지를 보내는 함수 |
| Wait_SRDY_Pin_High() | SRDY Pin이 High상태일 때까지 기다리는 함수 |
| Data_reception() | CC2530으로부터 Cortex-M4가 데이터를 수신하는 함수 |
| Set_MRDY_Pin_High() | MRDY Pin을 High 상태로 만드는 함수 |
| Send_DATA() | Cortex-M4로부터 게이트웨이가 수신한 데이터를 CDMA로 전송(그림 4) |

```
 1 Function Gateway_SPI_HANDLER()
 2 {
 3   if(SRDY_Pin_is_Low()){
 4   /*prepare Poll_Command*/
 5     Send_Poll_Command_to_CC2530();
 6     /*CC2530에 Poll command를 보낸다*/
 7     Wait_SRDY_Pin_High();
 8     /*Poll명령이 모두 전송되면 SRDY_Pin이 H
 9     igh로 상태가 바뀐다.*/
10     Data_reception();
11     Set_MRDY_Pin_High();
12      /*데이터 수신이 끝나면 MRDY를 High로 설
13     한다*/
14     Send_DATA();
15     /*수신한 데이터를 CDMA를 통해 서버로 송
16     신*/
17   }
18 }
```

(그림 3) SPI 수신 Handler 함수

진행이 되고 (그림 3)에 표현된 알고리즘에서 사용하는 함수는 <표 1>에서 설명한다.

1) Cortex-M4는 먼저 SRDY 핀이 Low로 되는 것을 외부 인터럽트를 통하여 감지한다.(라인 3)
2) Cortex-M4는 Poll_Command를 준비한다.(라인 4)
3) Cortex-M4는 CC2530에게 Poll_Command를 보낸다.(라인 5)
4) Poll_Command를 모두 전송하면, SRDY 핀이 High가 될 때까지 기다린다.(라인 7)
5) SRDY 핀이 High가 되면, Cortex-M4는 CC2530으로부터 데이터를 수신한다.(라인 10)
6) Cortex-M4가 데이터를 모두 수신하면 MRDY 핀을 High로 설정한다.(라인 11)

위의 과정이 완료되면 CC2530으로부터 Cortex-M4가 모두 데이터를 수신한 것이다. 그런데 게이트웨이는 CC2530으로부터 받은 데이터를 서버로 전달을 해주어야 한다.(라인 14)

CC2530으로부터 받은 데이터를 서버로 전달해주기 위해 SPI_HANDLER에서 받은 데이터를 RCU-890을 통하여 서버로 보내는 과정을 추가하였다. RCU-890을 통해 데이터를 서버로 보내주는 (그림 4)의 과정은 다음과 같은 순서로 진행되고, (그림 4)에 표현된 알고리즘에서 사용하는 함수는 <표 2>에서 설명한다.

<표 2> Send_Data 함수

| 함 수 | 설 명 |
|---|---|
| Convert_String_To_HexString() | Char형 문자를 Hex String으로 바꾸는 함수(예 : 'a' 문자를 "61"이라는 스트링으로 표현함) |
| Send_M4_to_CDMA_on_USART() | 변환한 스트링을 CDMA로 전송하는 함수 |

```
 1 Function Send_DATA()
 2 {
 3   Convert_String_To_HexString();
 4   /*문자를 스트링 문자열로 바꾼다.*/
 5   Send_M4_to_CDMA_on_USART();
 6   /*변환한 스트링을 USART통신을 이용하여
 7   CDMA로 전송*/
 8 }
```

(그림 4) Send_Data()함수

1) Cortex-M4는 CC2530으로부터 받은 데이터를 Hex String으로 변환한다.(라인 3)
2) Cortex-M4는 변환한 Hex String을 USART통신을 이용하여 CDMA로 전송한다.(라인 5)

### 4.2 서버로부터 게이트웨이의 데이터 수신

```
 1 Function Gateway_CDMA_HANDLER()
 2 {
 4   while(Data[number]!=EOF){
 5     Data[number]=USART_Receive();
 6     number++;
 7   }
 8   /*데이터가 EOF가 될 때까지 데이터수신*/
 9   if(Data_Message_is_read_Data){
10     Send_M4_to_CC2530();
11     /*USART통신으로 수신한 데이터를 CC2530으
12     로 전송*/
13   }
14 }
```

(그림 5) USART 수신 Handler 함수

<표 3> USART 수신 Handler 자료구조와 함수

| 자료구조와 함수 | 설 명 |
|---|---|
| Data[] | USART통신으로 수신한 데이터를 저장하는 배열 |
| Data_Message_is_read_Data | USART통신으로 데이터를 수신하였다는 조건 |
| Send_M4_to_CC2530() | USART를 통해 수신한 데이터를 CC2530으로 전송(표4) |

Cortex-M4는 CDMA로부터 USART통신을 통하여 데이터를 수신한다. USART 수신 Handler는 (그림 5)에 Gateway_CDMA_HANDLER를 통하여 알고리즘을 표현한다. 서버가 인터넷 망을 통해 데이터를 전송한다. 게이트웨이의 CDMA는 이 데이터를 수신한다. 이 데이터를 수신하면 인터럽트가 발생한다. 이 인터럽트를 처리하기위해 Gateway_CDMA_HANDLER를 자동으로 게이트웨이의 Cortex-M4가 수행한다. Gateway_SPI_HANDLER 알고리즘은 다음과 같은 순서로 진행된다. (그림 5)에 표현된 알고리즘에서 사용하는 자료구조와 함수는 <표 3>에서 설명한다.

Cortex-M4는 CDMA로부터 수신된 Data가 EOF가 될 때까지 Data를 배열에 저장한다. (라인 4 ~ 7)

Data 저장 과정을 마치면 서버로부터 Cortex-M4가 데이터를 모두 수신하게 된다. 그런데 서버로부터 데이터가 오는 경우가 두 가지가 있다.

첫 번째는 Gateway_SPI_HANDLER() 과정을 통하여 Cortex-M4에서 CDMA로 데이터를 전송한 경우에 응답으로 오는 메시지가 있고, 두 번째로는 우리가 원하는 명령으로써 서버에 명령을 내림으로써 오는 데이터이다.

게이트웨이는 서버로부터 온 명령을 센서노드로 전달하여야한다. 그런데 첫 번째 경우의 데이터는 센서노드로 보낼 필요가 없는 데이터이므로 두 번째 Read Data일 경우에만 받은 데이터를 센서노드로 전달한다.(라인 9 ~ 13)

Cortex-M4가 CC2530으로 SPI를 통해 데이터를 보내는 경우 즉, (그림 6)의 과정은 다음과 같은 순서로 진행되고 (그림 6)에서 표현된 알고리즘의 함수는 <표 4>에서 설명한다.

<표 4> Send_M4_to_CC2530 함수

| 함 수 | 설 명 |
|---|---|
| Convert_HexString_To_CharString() | Hex String을 Char String으로 변환하는 함수 |
| Send_M4_to_CC2530_on_SPI() | 변환한 String을 SPI를 통해서 CC2530으로 전송하는 함수 |

```
1 Function Send_M4_to_CC2530()
2 {
3    Convert_HexString_To_CharString();
4    /*HexString을 일반문자열로 바꾼다.*/
5    Send_M4_to_CC2530_on_SPI();
6    /* 변환한 스트링을 SPI통신을 이용하여 CC2530
7    으로 전송한다.*/
8 }
```

(그림 6) Send_M4_to_CC2530()함수

1) Cortex-M4가 CDMA를 통해 받은 HexString을 일반적인 문자열로 바꾼다.(라인 3)
2) Cortex-M4가 CC2530으로 SPI통신을 통하여 데이터를 전송한다.(라인 5)

### 4.3 Main

<표 5> Main 함수

| 함 수 | 설 명 |
|---|---|
| Wait_For_Interrupt() | 인터럽트가 발생할 때까지 Cortex-M4를 sleep모드로 유지시키는 함수 |

```
1 Function main()
2 {
3    while(1){
4       Wait_For_Interrupt();
5    }
6 }
```

(그림 7) main 함수

main함수는 (그림 7)에 표현된 알고리즘 같이 게이트웨이가 처리할 일이 없을 경우 즉, 데이터를 송수신과 데이터 처리를 안 할 경우에는 Sleep상태를 유지한다. (그림 7)에서 표현된 알고리즘의 함수는 <표 5>에서 설명한다.

## 4. 결론

본 논문은 게이트웨이에 대한 연구로써 통신을 Polling 방법이 아닌 인터럽트 기반으로 처리하는 알고리즘을 소개하였다.

본 논문의 의의는 인터럽트 기반으로 처리하는 알고리즘을 통하여 항상 Awake상태를 유지하는 Polling 방법보다 배터리 전력소모를 줄일 것으로 예상되고, 인터럽트 방식을 사용하여 게이트웨이와 센서노드, 게이트웨이와 서버간의 실시간 통신이 가능하다.

향 후 연구로는 본 논문에서 제안한 알고리즘을 적용하여 구현한 게이트웨이에서 Polling방식의 게이트웨이보다 전력소모가 적은 것을 성능평가를 통해 검증하고, 인터럽트 수를 최대한 줄여서 배터리 전력소모를 적게 하는 방법을 연구한다.

## 참고문헌

[1] 민경식, NET TERM, 인터넷 & 시큐리티 이슈, pp31-36, 2013

[2] 신용태 외 7인, 센서네트워크 식별체계 동향 분석 및 관리 방안 연구, 한국진흥원, pp1-151, 2007

[3] 이민구 외 3인, “Interrupt 기반의 반응 속도가 향상된 센서 데이터 전송 방법 구현”, 정보 및 제어 학술대회 ,Vol.2008, No.10, pp546-547, 2008

[4] 안재훈 외 3인, “효율적인 센서 운영체제를 위한 실시간 인터럽트 처리 기법“, 한국정보과학회논문, Vol.16, No.4, pp437-441, 2010

[5] 이준석, 장우진, "무선 센서 네트워크를 위한 클러스터 기반의 라우팅 기법에서의 확률적 클러스터 헤드 결정 방법에 대한 연구“, 대한산업공학회 추계학술대회, Vol.2007, No11, pp206-210, 2007

[6] ST, PM0214 Programming manual, 2014

[7] TEXAS INSTRUMENTS, A True System-on-Chip Solution for 2.4-GHz IEEE 802.15.4 and Zigbee Applications, 2014

[8] WOOJIN Wireless Networks Co, RCU-890 규격 및 프로그램 개발 가이드, 2008

[9] TEXAS INSTRUMENTS, CC2030-ZNP, 2011