VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Tan Xinyu

2.4GHz SENSOR NETWORK: CHIPCON CC2510 SELF-ORGANIZED NETWORK APPLICATION DEVELOPMENT

Technology and Communication

2009

ACKNOWLEDGMENT

The begin with, I have to appreciate my parents. They bring me to this world, taught me the knowledge to be a people, afford me study abroad. I really want to say, all my undertakings from before to the future are all yours. I will be prided you.

Then I have to appreciate my supervisors Mr. Gao Chao and Mr. Jukka Matila. I am so luck to meet these amazing teachers to give me a firm hand of technology. In this project Mr. Gao Chao gave me the topic and concept and Mr. Jukka Matila support me to make it run.

And I also want to say thank you to all of my friends, I am so glad to live with you guys from these years

.

Wednesday March 25 2009, VAASA FINLAND

TAN XINYU

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Degree Programme of Telecommunication Engineering

ABSTRACT

Author	Tan Xinyu
Title	2.4GHz sensor network: CHIPCON CC2006 self-organized network application development
Year	2006-2009
Language	English
Pages	54
Name of Supervisor	Jukka Matila

The abstract is written here using single spacing. The abstract should briefly state
This project is mainly to solve a real problem, which is big area temperature measurement. Short range telecom technology is used to build a self-organized sensor network to support system connections. Chipcon CC2510 RF solution as the main components is used to assemble the devices.

The essence of the project is protocol stack design and implementation. A Self-Adapted Network protocol stack (SAN) for sensor network is built in the system. This protocol stack offers following features:
Self-organization
Self-configuration and Self-reconfiguration
Self-healing
Self-optimization

In the case that the self-organized sensor network protocol is still on the research step around the world, no standardized network topology or protocol is released. This project is a good shot in sensor network area.

Keywords Network, 2.4GHz, Chipcon, Self-organized

ABBREVIATIONS

ACK	Acknowledgement
AP	Access Point
API	Application Program Interface
CCA	Clear Channel Assessment
CRC	Cyclic redundancy check
DK	Development Kit
ED	End Device
EM	Evaluation Module
IDE	Integral Development Environment
ISR	Interrupt Service Routine
LED	Light Emitted Diode
LQI	Link Quality Inductor
LSB	Least Significant Bit
MAC	Medium Access Control
MCU	Micro Controller
MSB	Most Significant Bit
PCB	Printed Circuit Board
RSSI	Received Signal Strength Inductor
SAN	Self-adapted Network
SMPL	Sample Modular Protocol Library
SoC	Software on Chip
SPI	Serial Port Interface
TI	Texas Instrument

CONTENT

1. INTRODUCTION

Nowadays, under the great success of Zigbee, Wi-Fi, Bluetooth, the short range wireless telecommunication in embedded system and network became more and more popular on electrical technology and embedded technology.

The famous electronic chip manufacturer Texas Instrument (TI) released a high performance 2.4GHz short range RF transceiver – CC2510/CC2511, can be used to development many powerful wireless applications.

This project is built for a real sensor network application which can be used for sensing tasks such as farm temperature measurement and forest fire monitoring which covers wide area and many sensor notes should be used. And we also expect longer sensor survives and the notes should be self-organized.

To build a self-organized network, I am using CC2510 to assemble sensor note. In the network, there are two different types of devices: access point and end device. The all the data should be collected by access point from end device. Because of the limited range of access point, the automatic routing algorithm is needed. In this case, every end device can be a router to forward data to the access point.

To implement these devices, three different tasks should be fulfilled:

1. Hardware task: A PCB board to drive RF module and its peripherals.
2. Telecommunication task: A wireless protocol stack for the entire network is indispensable.
3. Embedded system programming task: An embedded software driver

At the end of this project, I successfully built a sensor network contains one access point and seven different end devices which has self-organization, self-adoption features.

2. A PROBLEM IN FOREST TEMPERATURE MEASUREMENT

Think about it, if the scientists want to monitor temperature distribution in a big forest area, what do they need?

First, they need a kind of device can carry sensors and also has wireless data transceiver and power supplier with it. So they can receive data in one machine only. Second, they probably need tens of thousands of this kind of devices to cover this forest. Then they also want this kind of devices can work as long as passable. The most important thing is the devices can not be expensive, because they are all one time use device.

The device should contain following features.

- Low power consumption
- Wireless transceiver included
- Self-Routing protocol
- Cheap

Because of the first feature, low power consumption, the power of RF signal cannot be strong, which means the short rang wireless network is the only choice. So the devices have to forward the data of device behind it to the terminal device. Because of the huge number of devices, it is impassable to configure every device into the network, so the devices should organize themselves into the network. During the work period some of devices might power empty easily then others but the system has to heal itself to maintain the data from the devices still worked can be routed to the terminal machine.

So, my goal is to make a application device can assemble such kind of network and also has the features above.

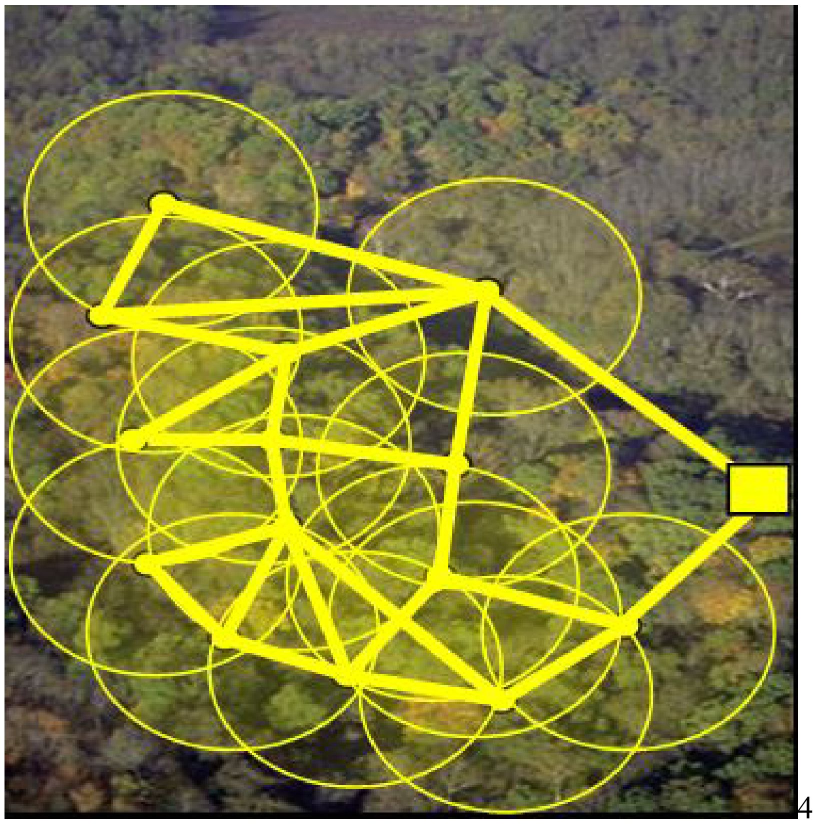
[4]

Figure 1 Forest temperature sensor network

(http://www.dei.unipd.it/~schenato/)

3. SYSTEM OVERVIEW

3.1 Self-organized network

Self-organized network originally is specialized ad hoc network, but some network like p2p show their self-organized character more and more obviously. The concept of self-organized network became wider and wider, not only includes wireless network, but also means p2p network and IP network (IP dynamic routing).

The essential characters of Self-organized network

a) Non-centralization and the equivalence among the notes. Self-organized network is an equivalence network and all the notes in the network are in the same level of hierarchy. It is infrastructureless, all the notes act as terminals and routers, if any note needs to communicate to the note out of its range, and it needs multi-hop distribution from other notes.

b) Self-Discovering, Self-Configuring, Self-Organizing and Self-Healing. The notes in the network can adapt the dynamics of the network, quickly find the other notes and discover the functionality of other notes. The notes correspond their behavior by distributed algorithm, no manually or pre-configure network device is needed. The Self-organized network can be setup quickly in any moment at any place. Because of the distribution of self-organized network, the redundancy of the notes and no single fault spot, any malfunction of note cannot break the network, which means the self-organized network has strong invulnerability and robustness.

3.2 Short range telecommunication system

Along with the development of digital telecommunication and computing technology, some requisites of short range telecommunication are put on the table. Compare with long range telecommunication, the short rang telecommunication has following difference:

a) The TX/RX power is between 1μW to 100μW;
b) The telecommunication range is between cm to hundreds of meters;
c) Mainly be used inside doors;
d) Using Omni directional antenna or PCB antenna;
e) Free charge of signal frequency;
f) High frequency operation;
g) Battery power supply wireless transceiver;

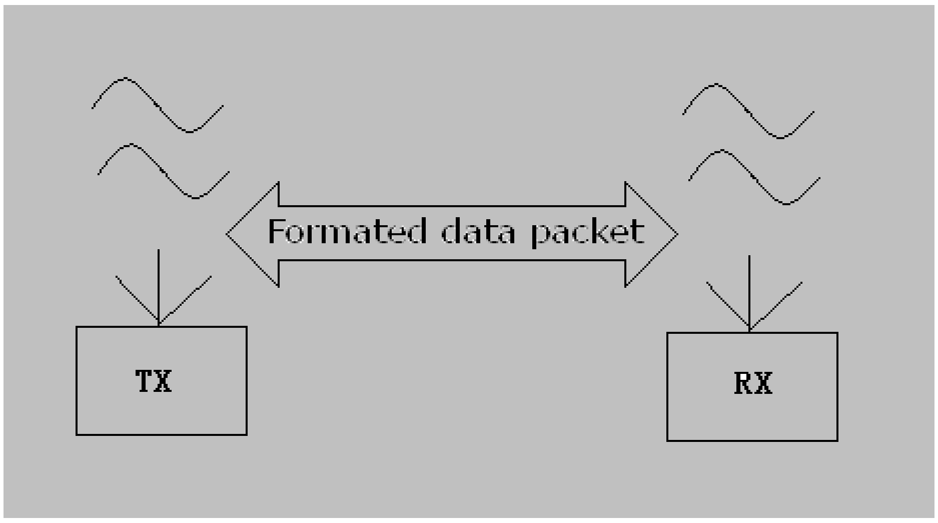


Figure 2 Short range telecommunication system

The transmitter (TX) sends data by wireless; Receiver receives data after process, it will get correct information with correct CRC (Figure 2).

The typical application of short range telecommunication system:

- Radio frequency identifier (RFID) works at 100kHz ~ 2.4GHz
- Wireless local area network (WLAN) works at 900MHz or 2.4GHz

- Wireless bar code reader works at 2.4GHz
- Wireless mouse and keyboard works at 27MHz, 433MHz, 2.4GHz
- Wireless security system works at 300 ~ 500MHz, 800MHz, 900MHz, 2.4GHz

3.3 2.4GHz wireless telecommunication

The unlicensed bandwidth 2.4GHz~2.5GHz used by short range telecom has become very popular in Personal Area Network (PAN) technology. The three remarkable applications worked on 2.4GHz bandwidth are ZigBee (IEEE 802.15.4), WiFi (IEEE 802.11b/g) and Bluetooth (IEEE 802.15.1).

3.4 CC2510

CC2510, a low-cost wireless SoC (System on Chip), is designed for low power consumption wireless application.

CC2510 includes an enhanced 8051 MCU and a wireless transceiver CC2500 with excellent performance. The wireless mainly works on ISM and SRD frequency bandwidth of 2.4GHz, so the frequency of system can be set in 2400~2483.5MHz.

3.4.1 Architecture

A block diagram of CC2510 is shown in Figure 3. The modules can be divided into one out of three categories: CPU related modules, radio-related modules, and modules related to power, test, and clock distribution. In the following subsections, a short description of each module that appears in Figure 3.

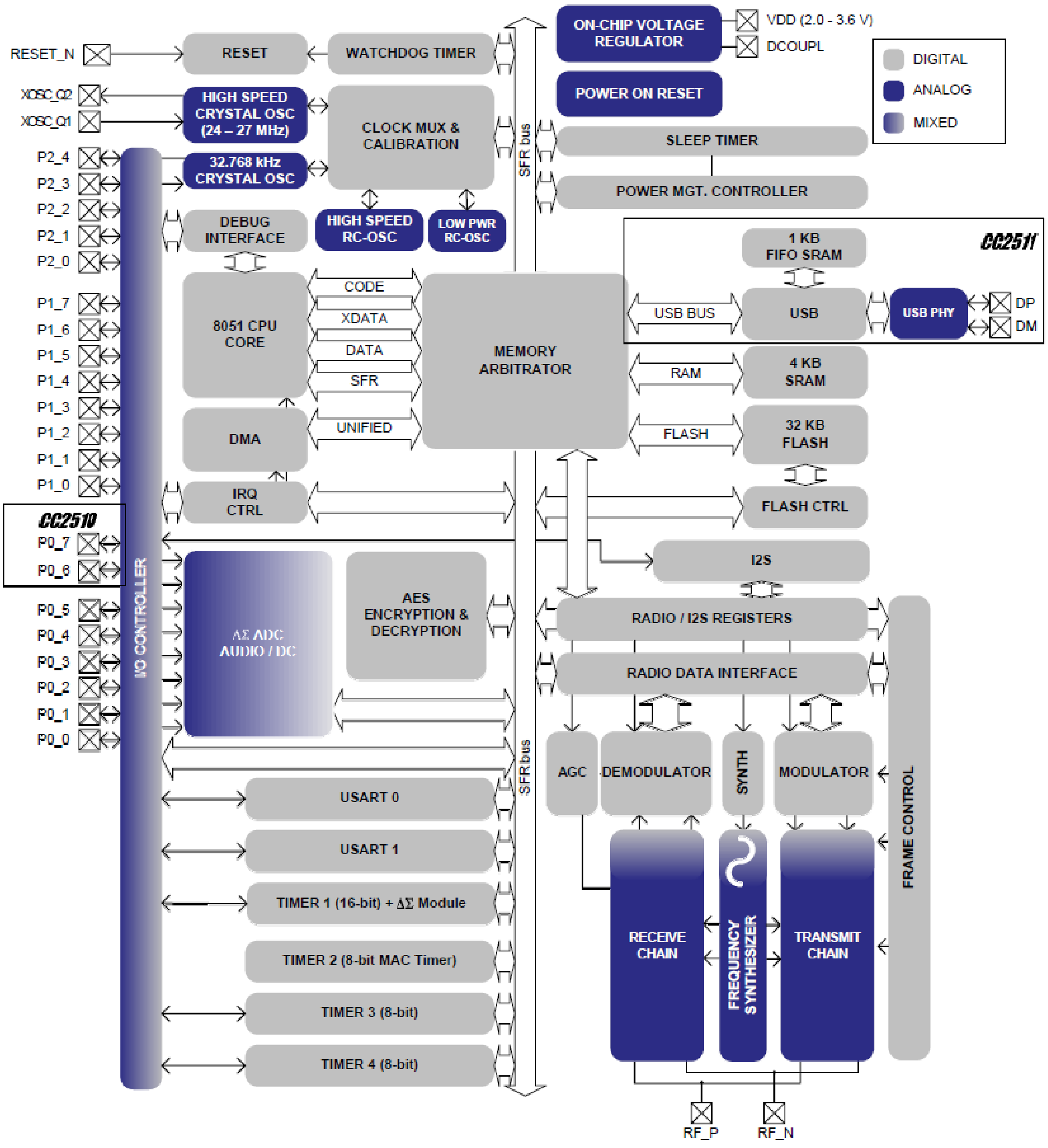


Figure 3: Architecture of CC2510 (Ref. 4)

3.4.2 The pin-out overview

The CC2510 pin-out shows in Figure 4 and Figure 5. There are 21 general IO pins in the CC2510 and low level active manually reset support.

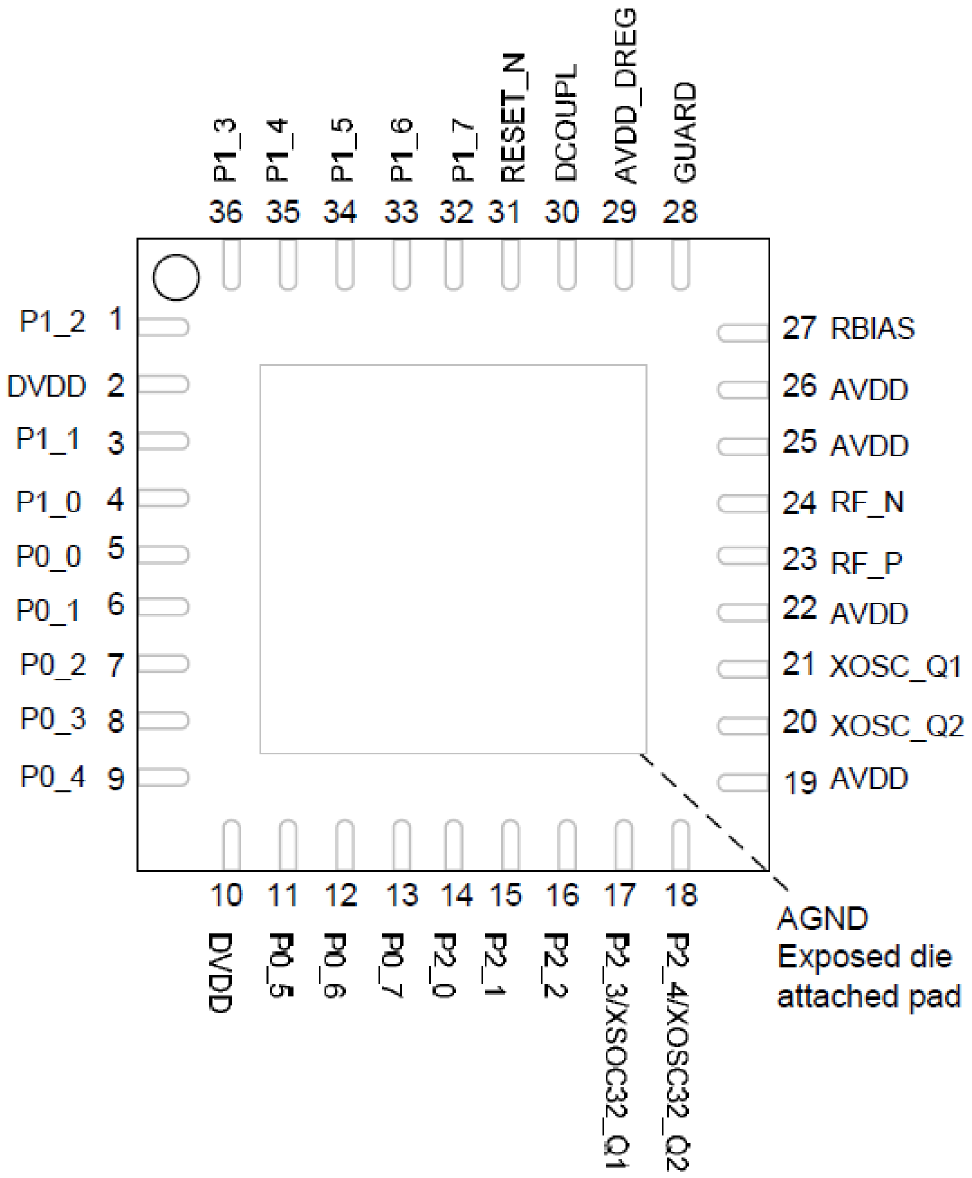
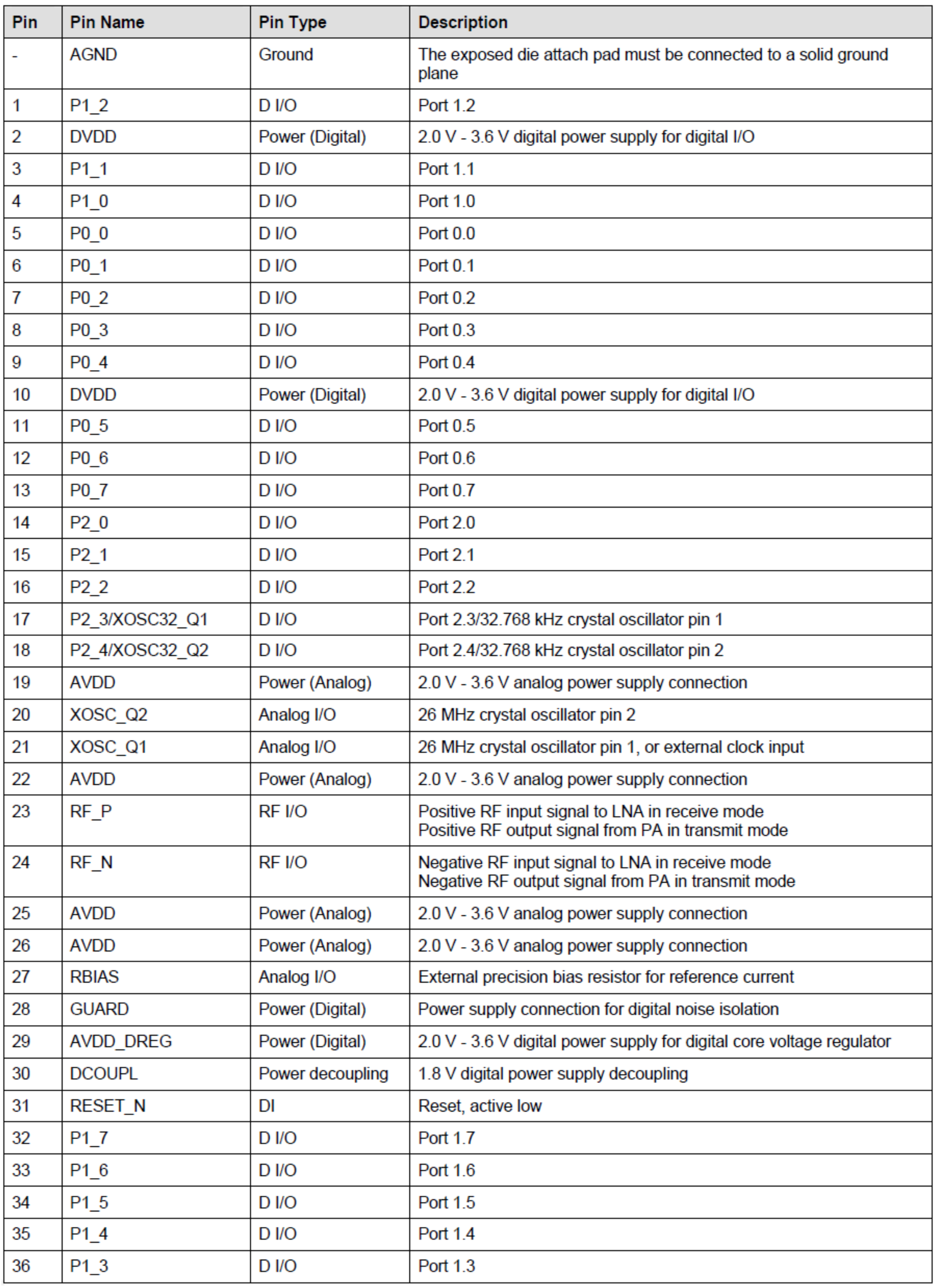


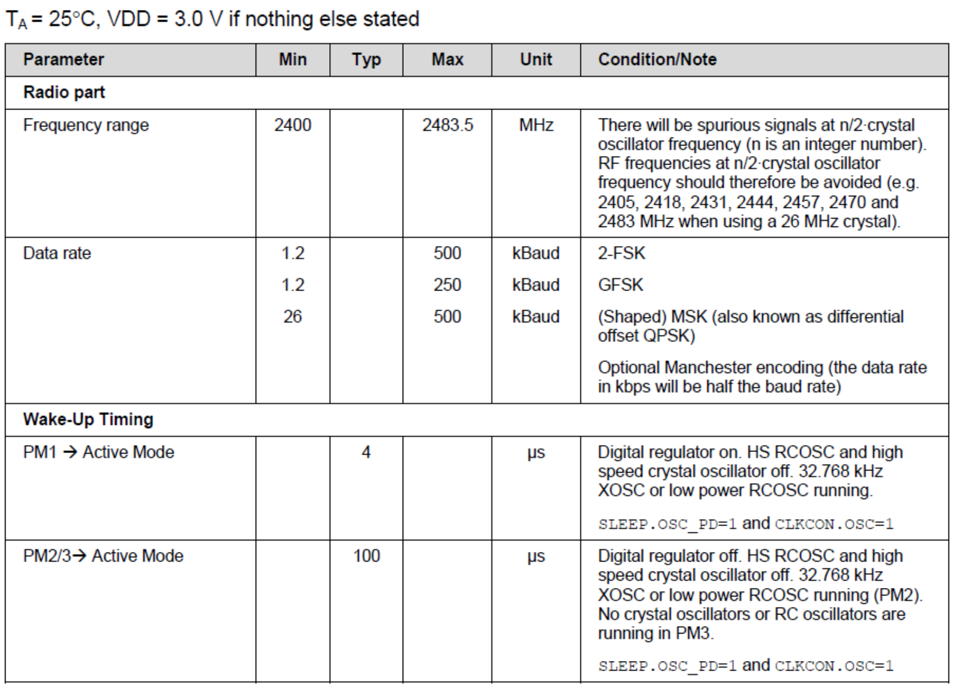
Figure 4: CC2510 Pinout top view (Ref. 4)



Pin	Pin Name	Pin Type	Description
-	AGND	Ground	The exposed die attach pad must be connected to a solid ground plane
1	P1_2	D I/O	Port 1.2
2	DVDD	Power (Digital)	2.0 V - 3.6 V digital power supply for digital I/O
3	P1_1	D I/O	Port 1.1
4	P1_0	D I/O	Port 1.0
5	P0_0	D I/O	Port 0.0
6	P0_1	D I/O	Port 0.1
7	P0_2	D I/O	Port 0.2
8	P0_3	D I/O	Port 0.3
9	P0_4	D I/O	Port 0.4
10	DVDD	Power (Digital)	2.0 V - 3.6 V digital power supply for digital I/O
11	P0_5	D I/O	Port 0.5
12	P0_6	D I/O	Port 0.6
13	P0_7	D I/O	Port 0.7
14	P2_0	D I/O	Port 2.0
15	P2_1	D I/O	Port 2.1
16	P2_2	D I/O	Port 2.2
17	P2_3/XOSC32_Q1	D I/O	Port 2.3/32.768 kHz crystal oscillator pin 1
18	P2_4/XOSC32_Q2	D I/O	Port 2.4/32.768 kHz crystal oscillator pin 2
19	AVDD	Power (Analog)	2.0 V - 3.6 V analog power supply connection
20	XOSC_Q2	Analog I/O	26 MHz crystal oscillator pin 2
21	XOSC_Q1	Analog I/O	26 MHz crystal oscillator pin 1, or external clock input
22	AVDD	Power (Analog)	2.0 V - 3.6 V analog power supply connection
23	RF_P	RF I/O	Positive RF input signal to LNA in receive mode Positive RF output signal from PA in transmit mode
24	RF_N	RF I/O	Negative RF input signal to LNA in receive mode Negative RF output signal from PA in transmit mode
25	AVDD	Power (Analog)	2.0 V - 3.6 V analog power supply connection
26	AVDD	Power (Analog)	2.0 V - 3.6 V analog power supply connection
27	RBIAS	Analog I/O	External precision bias resistor for reference current
28	GUARD	Power (Digital)	Power supply connection for digital noise isolation
29	AVDD_DREG	Power (Digital)	2.0 V - 3.6 V digital power supply for digital core voltage regulator
30	DCOUPL	Power decoupling	1.8 V digital power supply decoupling
31	RESET_N	DI	Reset, active low
32	P1_7	D I/O	Port 1.7
33	P1_6	D I/O	Port 1.6
34	P1_5	D I/O	Port 1.5
35	P1_4	D I/O	Port 1.4
36	P1_3	D I/O	Port 1.3

Figure 5: CC2510 Pinout (Ref. 4)

3.4.3 General Characters



T_A = 25°C, VDD = 3.0 V if nothing else stated

Parameter	Min	Typ	Max	Unit	Condition/Note
Radio part					
Frequency range	2400		2483.5	MHz	There will be spurious signals at n/2·crystal oscillator frequency (n is an integer number). RF frequencies at n/2·crystal oscillator frequency should therefore be avoided (e.g. 2405, 2418, 2431, 2444, 2457, 2470 and 2483 MHz when using a 26 MHz crystal).
Data rate	1.2		500	kBaud	2-FSK
	1.2		250	kBaud	GFSK
	26		500	kBaud	(Shaped) MSK (also known as differential offset QPSK)
					Optional Manchester encoding (the data rate in kbps will be half the baud rate)
Wake-Up Timing					
PM1 → Active Mode		4		μs	Digital regulator on. HS RCOSC and high speed crystal oscillator off. 32.768 kHz XOSC or low power RCOSC running. `SLEEP.OSC_PD=1` and `CLKCON.OSC=1`
PM2/3→ Active Mode		100		μs	Digital regulator off. HS RCOSC and high speed crystal oscillator off. 32.768 kHz XOSC or low power RCOSC running (PM2). No crystal oscillators or RC oscillators are running in PM3. `SLEEP.OSC_PD=1` and `CLKCON.OSC=1`

Figure 6: General Characters (Ref. 4)

3.4.4 Key Features (Ref. 4)

- High performance enhanced 8051 MCU core
- 2.4GHz high-performance RF transceiver CC2500
- High-sensitivity and powerful anti-jamming
- In sleep mode, current is low to 0.5uA, external interrupt or RTC can wake up the system; in standby mode, current is only 0.3uA, external interrupt can wake up the system.
- High to 500kBand programmable band rate
- Hardware support CSMA/CA
- Working voltage is 2.0V~3.6V
- Digital RSSI/LQI and DMA support
- Battery monitoring and on chip temperature sensor\
- 8~14 bit ADC
- Less facility circuitrequirement

3.4.5 CC2510DK (REF. 4)

A CC2510 development kit has a main board (Smart RF 04 EB) and several CC2510EMs.

Evaluation Board is the platform for Evaluation Module (EM) with MMI (Man to Machine Interface), including LCD, LEDs, potential meters and buttons. USB interface and RS232 interface are also attached. (Figure 7)

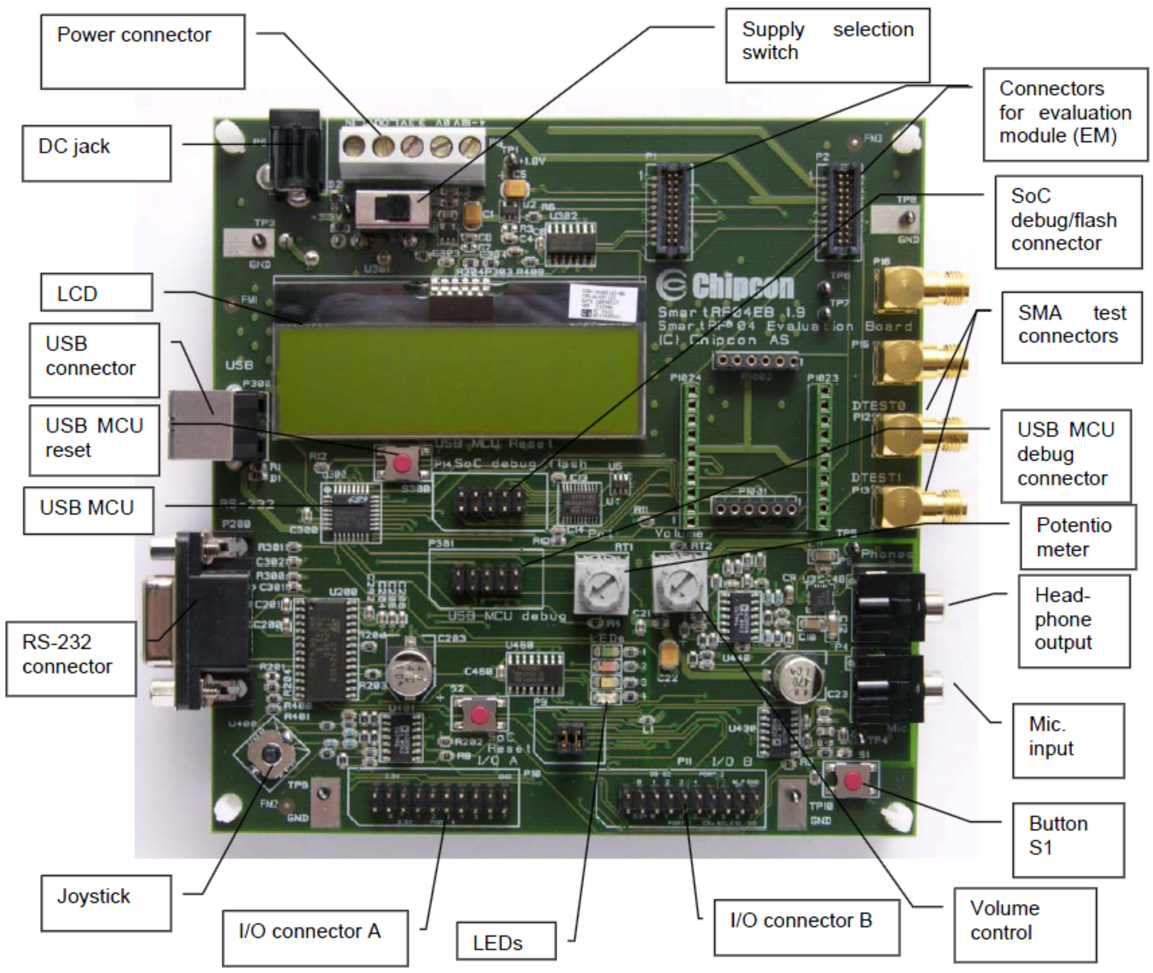


Figure 7: Smart RF 04 EB (Ref. 4)

The SmartRF04DK Development Kit offers quick testing of the RF interface and peripherals with its on-board functions and applications.

- Evaluate the SmartRF®04 products. A RF range testing program has been pre-installed into the kit.

• SmartRF® Studio is a platform can be used to perform RF measurements. The radio function registers can be easily configured for sensitivity measurement, power consumption so on.

• The SmartRF04KD can be used for prototype development. USB interface can be used as emulator interface for CC2510. All the IO ports of CC2510 are can be easily accessed by the connectors on the edge of the board for the external applications.

CC2510 Evaluation Module is a plug-in module for Smart04DK with CC2510 and necessary peripheral circuit.

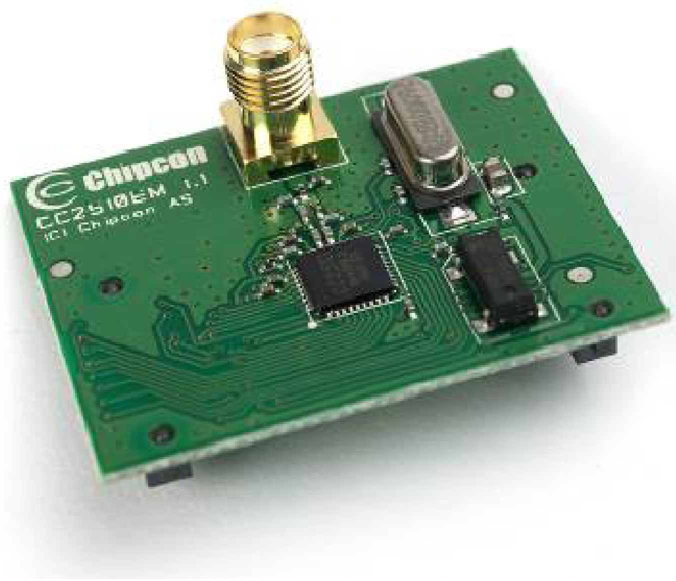
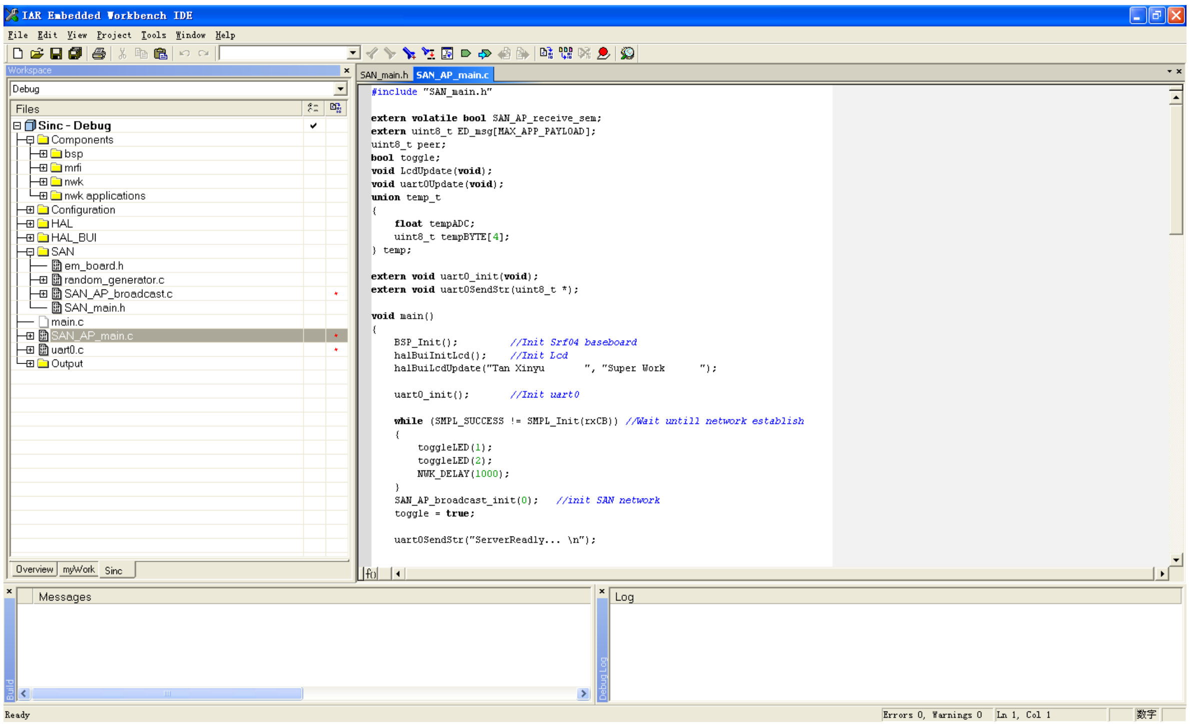


Figure 8: CC2510EM (Ref. 4)

3.5 IAR Embedded Workbench® for 8051 (www.iar.com)

IAR Embedded Workbench is a set of highly sophisticated and easy-to-use development tools for embedded applications. It integrates the IAR C/C++ Compiler™, assembler, linker, librarian, text editor, project manager, and C-SPY® Debugger in an integrated development environment (IDE). With its built-in chip-specific code optimizer, IAR Embedded Workbench generates very efficient and reliable FLASH/ PROMable code for the 8051 microcontroller. In addition to this solid technology, IAR Systems also provides professional worldwide technical support.



```c
#include "SAN_main.h"

extern volatile bool SAN_AP_receive_sem;
extern uint8_t ED_msg[MAX_APP_PAYLOAD];
uint8_t peer;
bool toggle;
void LcdUpdate(void);
void uart0Update(void);
union temp_t
{
    float tempADC;
    uint8_t tempBYTE[4];
} temp;

extern void uart0_init(void);
extern void uart0SendStr(uint8_t *);

void main()
{
    BSP_Init();         //Init Srf04 baseboard
    halBuiInitLcd();    //Init Lcd
    halBuiLcdUpdate("Tan Xinyu       ", "Super Work      ");

    uart0_init();       //Init uart0

    while (SMPL_SUCCESS != SMPL_Init(rxCB)) //Wait untill network establish
    {
        toggleLED(1);
        toggleLED(2);
        NWK_DELAY(1000);
    }
    SAN_AP_broadcast_init(0);   //init SAN network
    toggle = true;

    uart0SendStr("ServerReadly... \n");
```

Figure 9: IAR 8051 WORKBENCH

3.6 SimpliciTI (Ref. 5)

SimpliciTI is a connection-based peer-to-peer protocol. It supports 2 basic topologies: strictly peer-to-peer and a star topology in which the star hub is a peer to every other device. The Access Point is used primarily for network management duties. It supports such features and functions as store-and-forward support for sleeping End Devices, management of network devices in terms of membership permissions, linking permissions, security keys, etc. The Access Point can also support End Device functionality, i.e., it can itself instantiate sensors or actuators in the network. In the star topology the Access Point acts as the hub of the network. The protocol is implemented by a few API functions. These APIs support peer to peer communication on customer application. The APIs can also organized flexible cluster tree topology with access point and extenders.

4. SOLUSION – A SELF-ORGANIZED NETWORK PROTOCOL BASED ON BROARDCASTING

The solution of this project is an extended protocol of SampliciTI. Because SampliciTI support two types of topology only: strict peer to peer topology, and star topology. Both of them are not self-organized. But it still offers some good APIs and that is the reason I use this component.

This solution is specialized on a certain issue – forest temperature measurement sensor network. Two kinds of devices are needed – Access Point (AP) and End Device (ED).

4.1 Access Point is a data collector. Only one AP can be used in a network. It is the center of entire network, all the EDs will try to link to AP and transmit their data to it.

4.2 End Device: is a device carried a temperature sensor and a RF transceiver. All the EDs have the same level in the network hierarchy. They will try to link to AP and transmit their data to it.

4.3 Protocol description

The entire system has hierarchy which depends on the distance from the EDs to AP. The EDs on level 1 means the direct connection to AP. The EDs on level N (*2, 3, 4…*) means there are N – 1 hops between ED and AP.

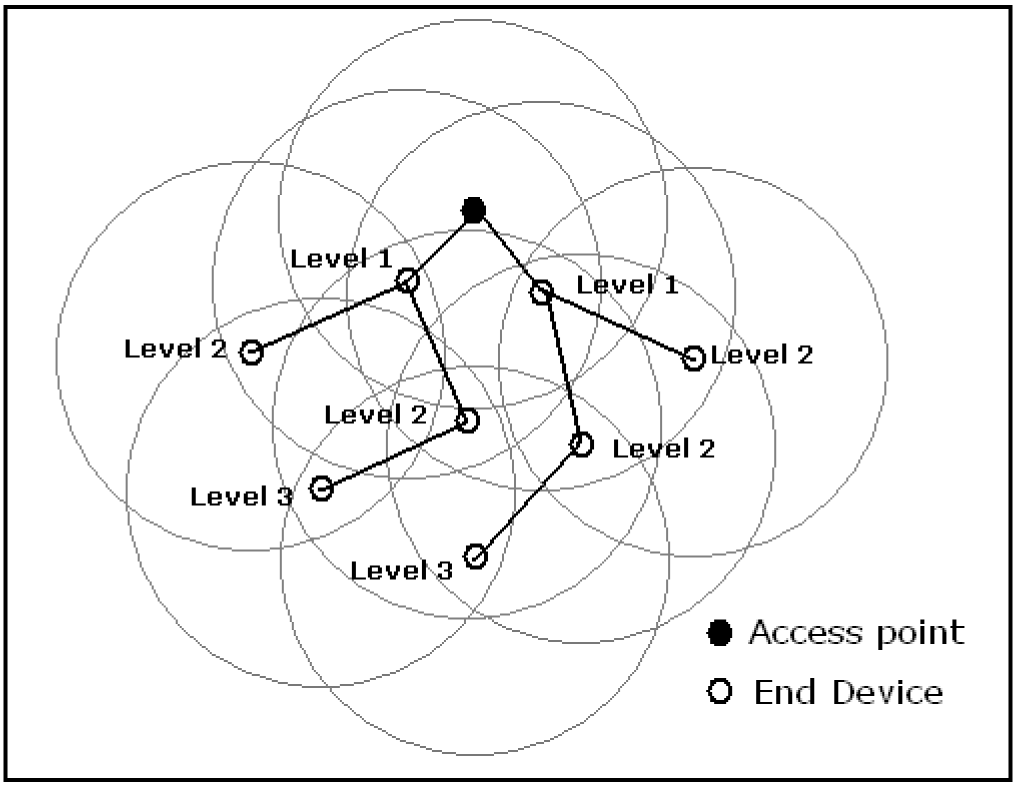


Figure 10: Network overview

4.3.1 AP -- ED

Once the AP setup, will broadcast SAN_AP_BROADCAST every six seconds approximately. Once an unlinked ED hears SAN_AP_BROADCAST, it will wait a random time to hear if any other EDs send SAN_ED_BROADCAST_ACK to AP. If a SAN_ED_BROADCAST_ACK is heard the ED will skip and wait for next SAN_AP_BROADCAST message, otherwise it will broadcast SAN_ED_BROADCAST_ACK message then try to link to AP. Meanwhile, when AP receives SAN_ED_BROADCAST_ACK, it will listening to the link so the ED who sent SAN_ED_BROADCAST_ACK can be linked by the AP. (Figure 11)

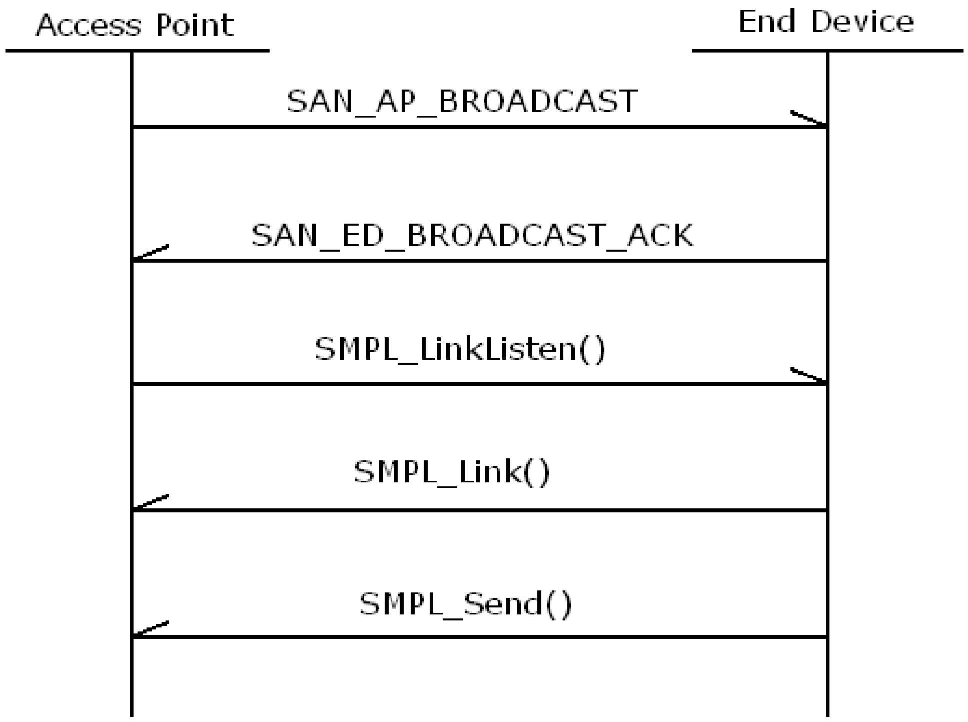


Figure 11: Direct link between AP and ED

4.3.2 ED -- ED

If once an ED has linked to the AP or another ED, it will broadcast SAN_ED_BROADCAST message with the format (Figure 12)

SAN_ED_BROADCAST	Level (hops)	Self ID
one byte	one byte	one byte

Figure 12: Format of SAN_ED_BROADCAST packet

It carries the ED's level (hops to the AP) and its ID. Once a unlinked ED or an ED with the level bigger than level from the SAN_ED_BROADCAST it heard plus one, it will wait a random time to listen if an SAN_ED_ED_BROADCAST_ACK is sent, otherwise it will broadcast SAN_ED_ED_BROADCAST_ACK packet (Figure 13)

SAN_ED_ED_BROADCAST_ACK one byte	Target ID one byte

Figure 13: Format of SAN_ED_ED_BROADCAST_ACK

The packet carries the device ID it want to link. Then the ED with the same ID as Target ID received the packet will listen to the link. Then the ED want to be linked will try to link to the network, in this case this ED can join to the network successfully. Furthermore, each ED knows its next hop to AP and it will send its data message directly to the next hop. Each ED also forwards the data packet it received to its next hop. After all, the entire packet from the EDs can be collected by AP. (Figure 14)

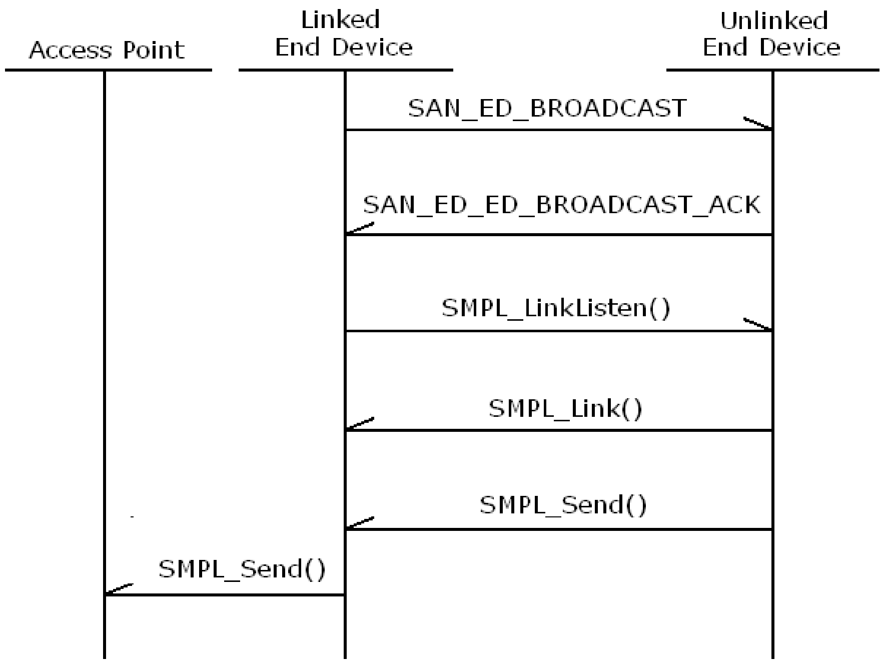


Figure 14: Link to the ED cannot hear AP

4.3.3 Self-Healing

Once an ED is connected to the network, it will be trying to listen to the broadcast message from its next hop. If one ED missed three broadcast of its next hop device, which means his next hop device do not work anymore or it was moved to anther place. Then this ED will try to hear anther device broadcast and link it.

The following picture will show the concept of the Self-Healing.

At the beginningAll the EDs in the network were all connected to the AP. T he ED A and B were connected to AP directly. C and D connected to the ED A, and then A will get the responsibility to forward the message from C and D to the AP. (Figure 15)

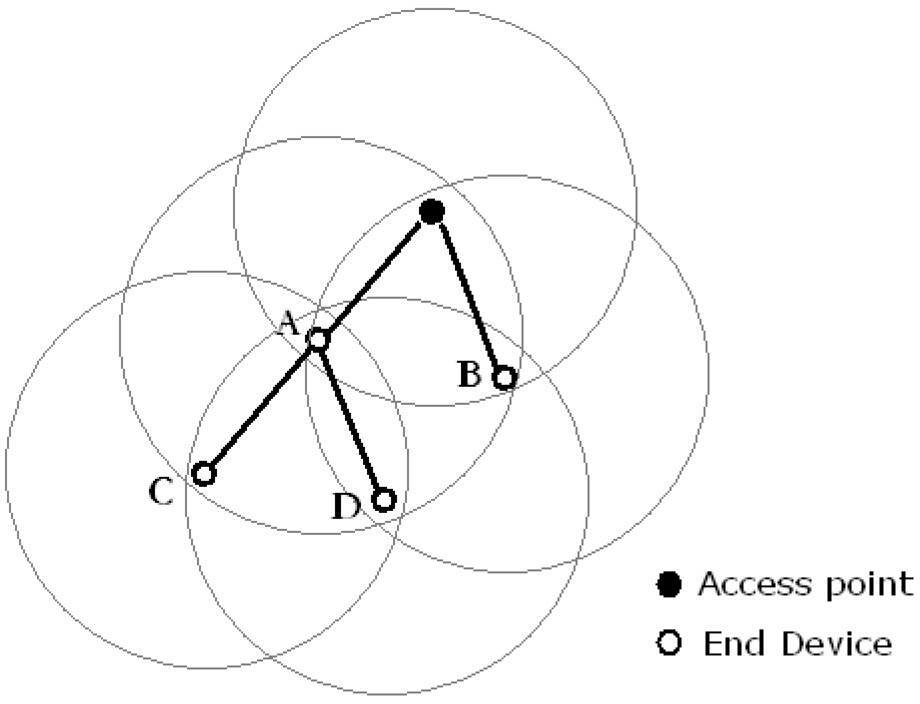


Figure 15: Network was setup successfully

Then ED A is lost its power. At the moment, because of no ACK on directly sending message, the ED C and D have on idea what is happened to the ED A, so they keep sending their message to A. From now, C and D cannot hear the broadcast from A anymore, after awhile, they will notice that the ED A was lost. Then they will be trying to find a new route to the AP. (Figure 16)

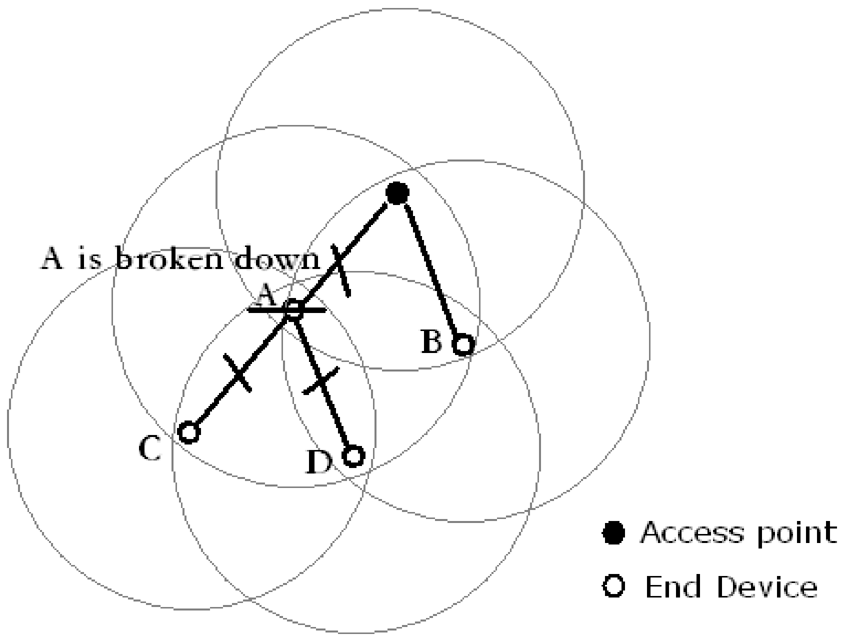


Figure 16: End device A is broken down

The ED D heard the broadcast from the ED B and setup a connection. Meanwhile, the ED C was still out of the network. Once B was connection, it broadcasted the message and C heard it, so C was connected to the system again.

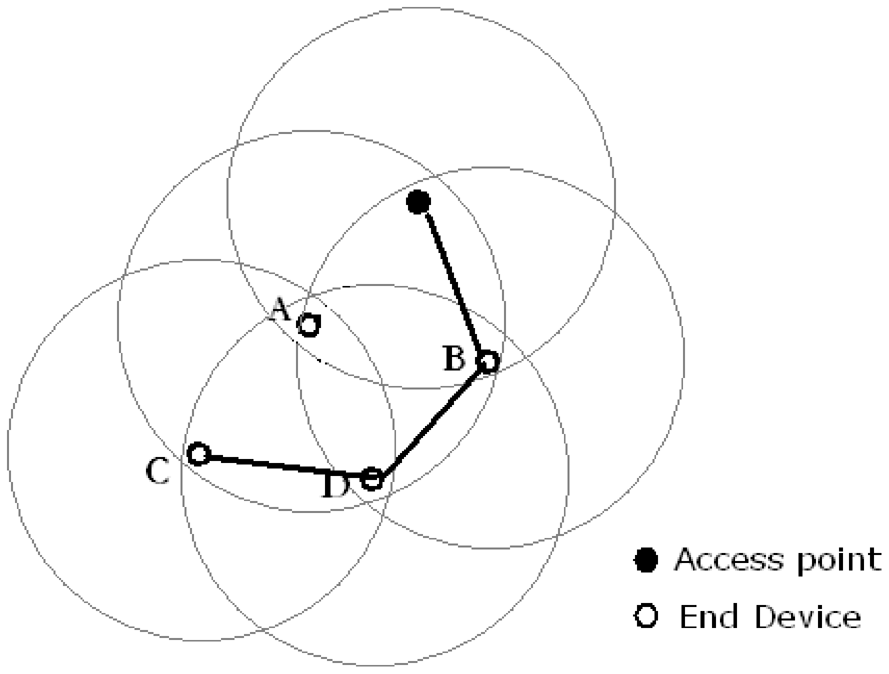


Figure 17: New route is found

5. IMPLEMENTATION

By implementing the system, two main tasks shall be achieved:

➢ **Hardware task:** I am going to make a PCB board can support CC2510EM mount, battery socket and some indispensable circuit.

➢ **Software task:** write an embedded software to implement my protocol stack and some other peripheral functions to drive the device.

5.1 Hardware

There are two types of devices in the project (Access point and End device). I used one of Smart RF 04 BOARD with a CC2510EM as an Access point, only because it has ready made UART – RS232 periphery circuit. In this case, I can transmit the data collected by AP to PC directly through SPI. (Figure 18)

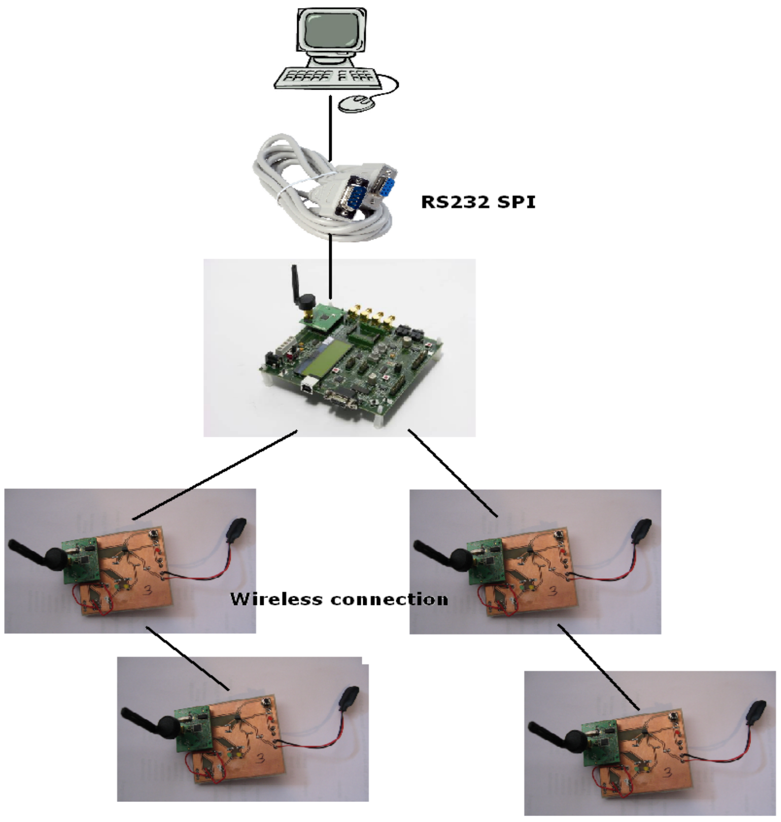


Figure 18: System overview

In the End device base board, there are following components:

- CC2510EM
- 2ROW, 20WAY HEADER * 2
- GREEN LED
- YELLOW LED
- RED LED
- LM3717L positive adjustable voltage regulator
- Button
- Battery socket
- Power socket * 2
- 4 bit digital switch

Figure 19 shows the hardware structure.

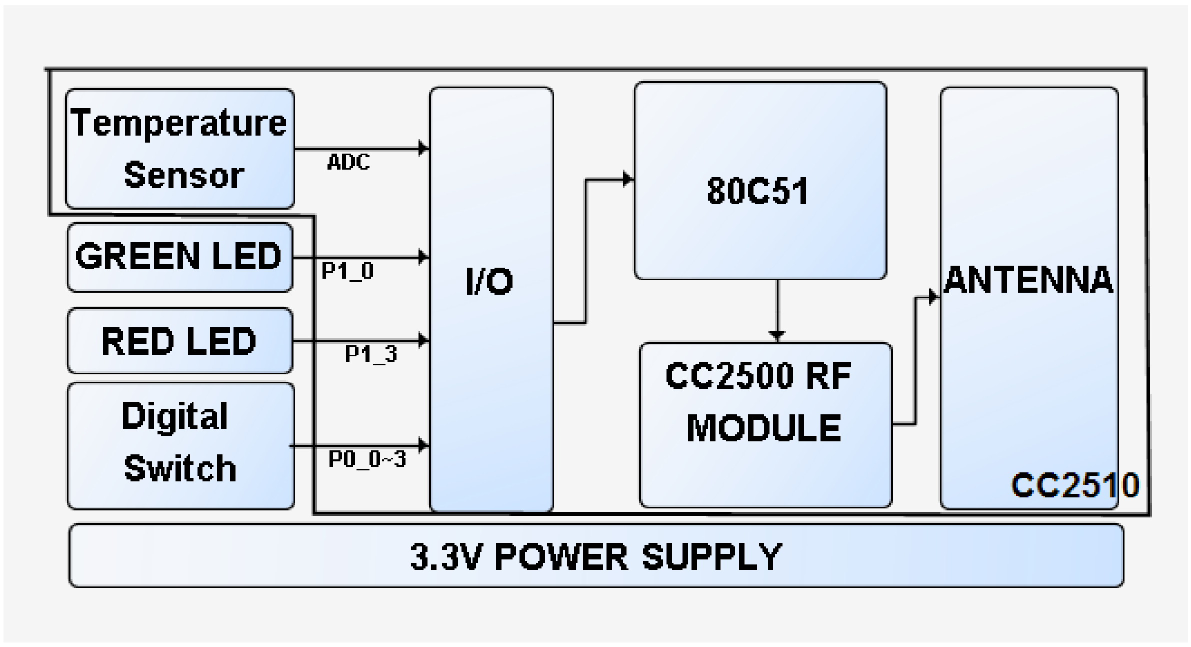


Figure 19: Hardware Structure

3.3V power supply:

LM 317L 0.1A positive adjustable voltage regulator is used.

The LM317L provides an internal reference voltage of 1.25V between the output and adjustments terminals. This is used to set a constant current flow across an external resistor divider (see fig. 4), giving an output voltage VO of:

VO = VREF (1 + R2/R1) + IADJ R2

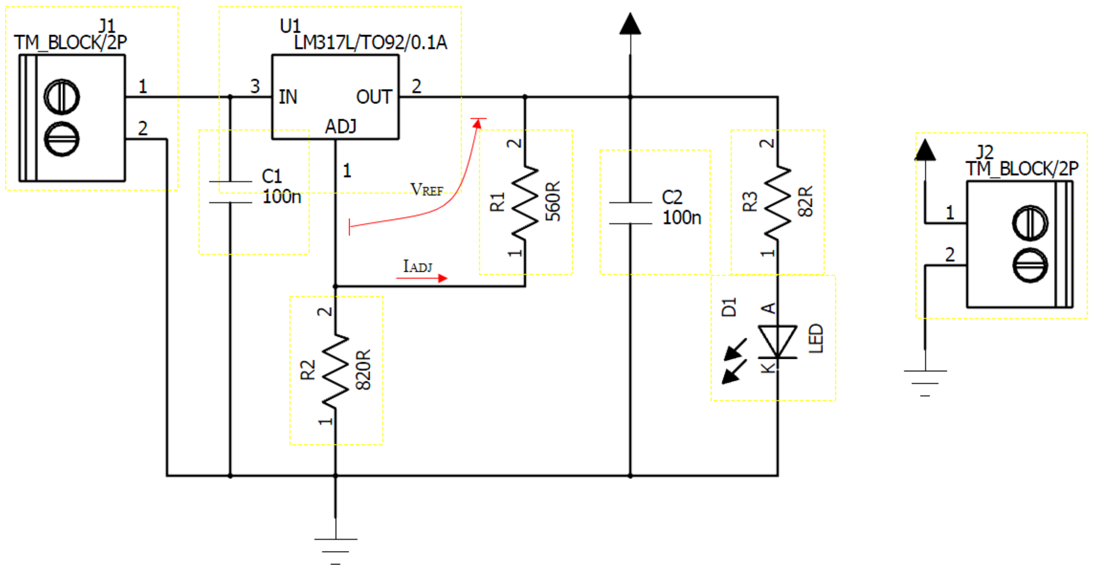


Figure 20: POWER SUPPLY

When R1 is 560 ohm and R2 is 820 ohm, the Vo will be approximately 3.1V. IN case CC2510 can work with Vcc from 2.8V to 3.3V, this power supply will fulfill the system.

Here is the view of ED base board. (Figure 21 and Figure 22)

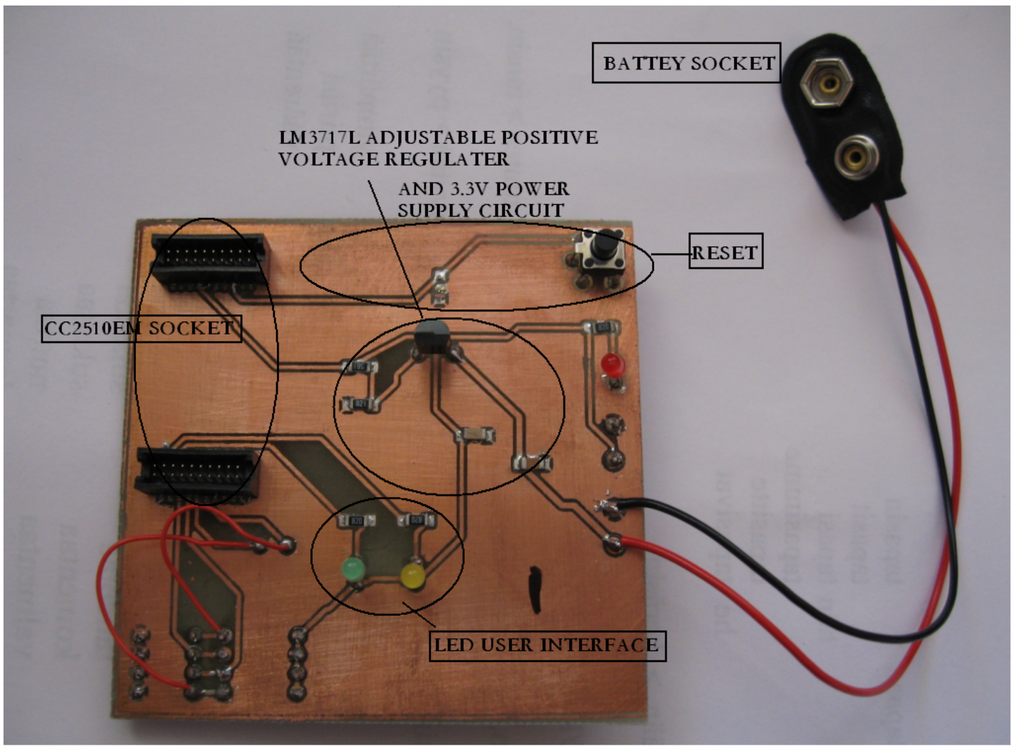

BATTEY SOCKET
LM3717L ADJUSTABLE POSITIVE VOLTAGE REGULATER
AND 3.3V POWER SUPPLY CIRCUIT
RESET
CC2510EM SOCKET
LED USER INTERFACE

Figure 21: ED base board top view

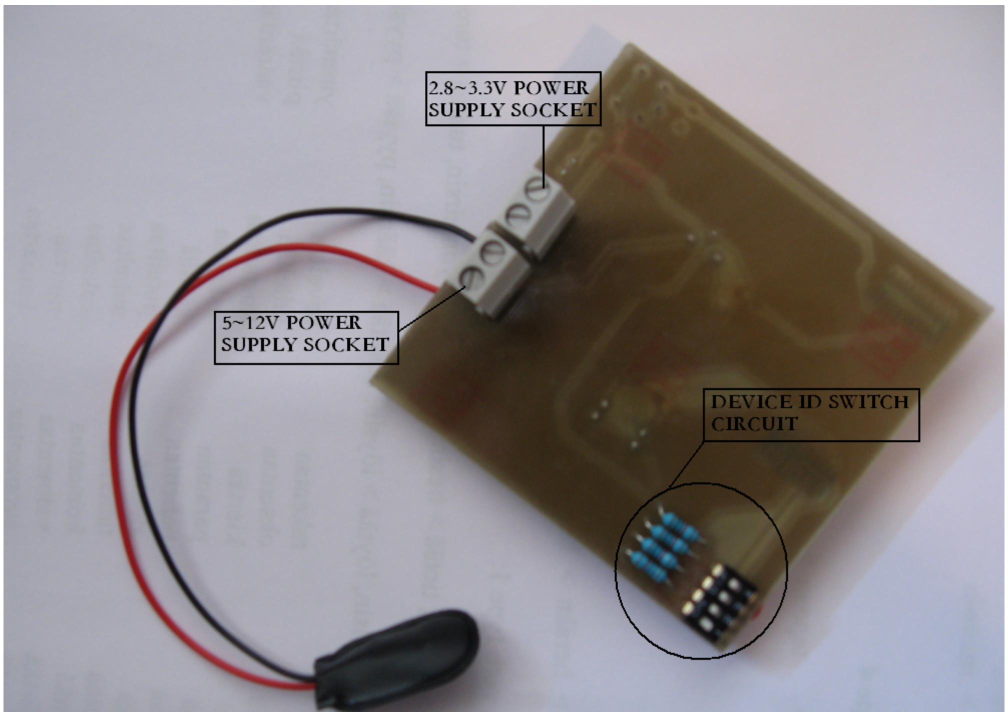


Figure 22: ED base board bottom view

5.2 Application software

The CC2510 includes an 8-bit CPU core which is an enhanced version of the industry standard 8051 core. The enhanced 8051 core used the standard 8051 instruction set. Instructions execute faster than the standard 8051 due to one clock per instruction cycle is used as opposed o 12 clocks per instruction cycle in the standard 8051 and wasted bus states are eliminated.

There are six groups of source and header files inside the program:

- **Components:** the code packet of SimpliciTI
- **Configurations:** the special configuration file of SimpliciTI
- **HAL:**The Hardware analog function management code.
- **HAL_BUI:** the APIs used for Smart RF 04 BOARD peripheries
- **SAN:** Self-organized network management code.
- **Others:** such as UART management code and main code.

5.2.1 SimpliciTI API (Ref. 5)

SimpliciTI software conceptually supports 3 layers as shown in Figure 23. The Application Layer is the only portion that the customer needs to develop. The communication support is provided by a simple set of API symbols used to initialize and configure the network, and read and write messages over air.

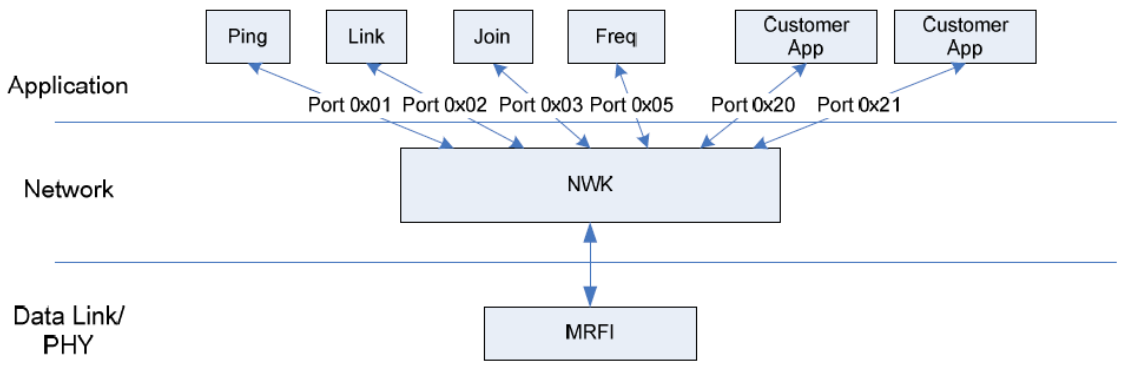


Figure 23: SimpliciTI Architecture

5.2.1.1 Common data types

The following are defined:

typedef signed char int8_t;

typedef signed short int16_t;

typedef signed long int32_t;

typedef unsigned char uint8_t;

typedef unsigned short uint16_t;

typedef unsigned long uint32_t;

typedef unsigned char linkID_t;

typedef enum smplStatus smplStatus_t;

5.2.1.2 APIs

BSP_Init()

Discretion	Not strictly part of the SimpliciTI API this call initializes the specific target hardware. It should be invoked before the SMPL_Init() call.
Prototype	Void BSP_Init(void)
Parameters	None
Return	None

SMPL_Init()

Discretion	This function initializes the radio and the SimpliciTI protocol stack. It must be called once when the software system is started and before any other function in the SimpliciTI API is called.
Prototype	smplStatus_t SMPL__Init(uint8_t (*)(linkID_t))
Parameters	The argument is a pointer to a function and causes the supplied function to be registered as the callback function for the device. Since the initialization is called only once the

	callback serves all logical End Devices on the platform. The function prototype is: uint8_t sCallBack(linkID_t) The function is invoked in the frame-receive ISR thread so it runs in the interrupt context. It is valid for this parameter to be null if no callback is supplied.
Return	SMPL_SUCCESS Initialization successful. SMPL_NO_JOIN No Join reply. Access Point possibly not yet up. Not an error if no Access Point in topology SMPL_NO_CHANNEL Only if Frequency Agility enabled. Channel scan failed. Access Point possibly not yet up.

SMPL_Link()

Discretion	This call sends a broadcast link frame and waits for a reply. Upon receiving a reply a connection is established between the two peers and a Link ID is assigned to be used by the application as a handle to the connection. This call will wait for a reply but will return if it does not receive one within a timeout period so it is not a strictly blocking call. The amount of time it waits is scaled based on frame length and data rate and is automatically determined during initialization. This call can be invoked multiple times to establish multiple logical connections. The peers may be on the same or different devices than previous connections.
Prototype	smplStatus_t SMPL_Link(linkID_t *lid)
Parameters	The parameter is a pointer to a Link ID. If the call succeeds

	the value pointed to will be valid. It is then to be used in subsequent APIs to refer to the specific peer.
Return	SMPL_SUCCESS Link successful. SMPL_NO_LINK No Link reply received during wait window. SMPL_NOMEM No room to allocate local Rx port, no more room in Connection Table, or no room in output frame queue. SMPL_TX_CCA_FAIL Could not send Link frame.

SMPL_LinkListen()

Discretion	This call will listen for a broadcast Link frame. Upon receiving one it will send a reply directly to the sender. This call is a modified blocking call. It will block “for a while” as described by the following constant set in the nwk_api.c source file: CONSTANT DESCRIPTION LINKLISTEN_MILLISECONDS_2_WAIT Number of milliseconds this thread should block to listen for a Link frame. The default is 5000 (5 seconds) The application can implement a recovery strategy if the listen times out. This includes establishing another listen window. Note that there is a race condition in that if the listen call is invoked upon a timeout it is possible that a link frame arrives during the short time the listener is not listening.
Prototype	smplStatus_t SMPL_LinkListen(linkID_T *lid)
Parameters	The parameter is a pointer to a Link ID. If the call succeeds

	the value pointed to will be valid. It is then to be used in subsequent APIs to refer to the specific peer.
Return	SMPL_SUCCESS Link successful. SMPL_TIMEOUT No link frame received during listen interval. Link ID not valid.

SMPL_Send()

Discretion	This function sends application data to a peer. The network code takes care of properly conditioning the radio for the transaction. Upon completion of this call the radio will be in the same state it was before the call was made. The application is under no obligation to condition the radio. By default the transmit attempt always enforces CCA.
Prototype	void SMPL_Send(linkID_t lid, uint8_t *msg, uint8_t len)PA
Parameters	lid Link ID of peer to which to send the message. msg Pointer to message buffer. len Length of message. This can be 0. It is legal to send a frame with no application payload. The ‘lid’ parameter must be one established previously by a successful Link transaction. The exception is the Unconnected User Datagram Link ID. This Link ID is always valid. Since this Link ID is not connection-based a message using this Link ID is effectively a datagram sent to all applications.
Return	SMPL_SUCCESS Transmission successful. SMPL_BAD_PARAM No valid Connection Table entry for Link ID; data in Connection Table entry bad; no message or

	message too long. SMPL_NOMEM No room in output frame queue. SMPL_TX_CCA_FAIL CCA failure. Message not sent.

SMPL_Receive()

Discretion	This function checks the input frame queue for any frames received from a specific peer. Unless the device is a polling device this call does not activate the radio or change the radio's state to receive. It only checks to see if a frame has already been received on the specified connection. If the device is a polling device as specified in the device configuration file the network layer will take care of the radio state to enable the device to send the polling request and receive the reply. In this case conditioning the radio is not the responsibility of the application. If more than one frame is available for the specified peer they are returned in first-in-first-out order. Thus it takes multiple calls to retrieve multiple frames.
Prototype	smplStatus_t SMPL_Receive(linkID_t lid, uint8_t *msg, uint8_t *len)
Parameters	lid Check for messages from the peer specified by this Link ID. msg Pointer to message buffer to populate with received message. len Pointer to location in which to save length of received message.

	The ‘lid’ parameter must be one established previously by a successful Link transaction. The exception is the Unconnected User Datagram Link ID. This Link ID is always valid. The application must ensure that the message buffer is large enough to receive the message. To avoid a buffer overrun the best strategy is to supply a buffer that is as large as the maximum application payload specified in the network configuration file (MAX_APP_PAYLOAD) used during the project build.
Return	SMPL_SUCCESS Frame for the Link ID found. Contents of ’msg‘ and ’len ‘ are valid. SMPL_BAD_PARAM No valid Connection Table entry for Link ID; data in Connection Table entry bad. SMPL_NO_FRAME No frame available. SMPL_NO_PAYLOAD Frame received with no payload. Not necessarily an error and could be deduced by application because the returned length will be 0. SMPL_TIMEOUT Polling Device: No reply from Access Point. SMPL_NO_AP_ADDRESS Polling Device: Access Point address not known. SMPL_TX_CCA_FAIL Polling Device: Could not send data request to Access Point SMPL_NOMEM Polling Device: No memory in output frame queue SMPL_NO_CHANNEL Polling Device: Frequency Agility enabled and could not find channel.

5.2.2 Access point software

An Access point has the tasks of data collection and connection to PC by UART. In another word, an AP will catch the data from the wireless network and forward it to PC by wired network.

5.2.2.1 Protocol stack program

There are three essential functions support AP SAN protocol (SAN_AP_broadcast.c).

SAN_AP_broadcast

Discretion	SAN_AP_broadcast should be executed frequently to make sure all the events generated by the interrupt can be processed on time.
Prototype	bool SAN_AP_broadcast(void)
Parameters	
Return	True Event processed successfully False No event.

rxCB

Discretion	The rxCB will be executed by the RX ISR. Once the AP receives information from wireless then rxCB will be executed. This function is mainly used to de-frame and response.
Prototype	Uint8_t rxCB(uint8_t port)
Parameters	Port the linked of captured frame.
Return	1

t1_int

Discretion	The ISR of timer1, an interrupt is generated every six seconds approximately.
Prototype	void t1_int(void)
Parameters	
Return	

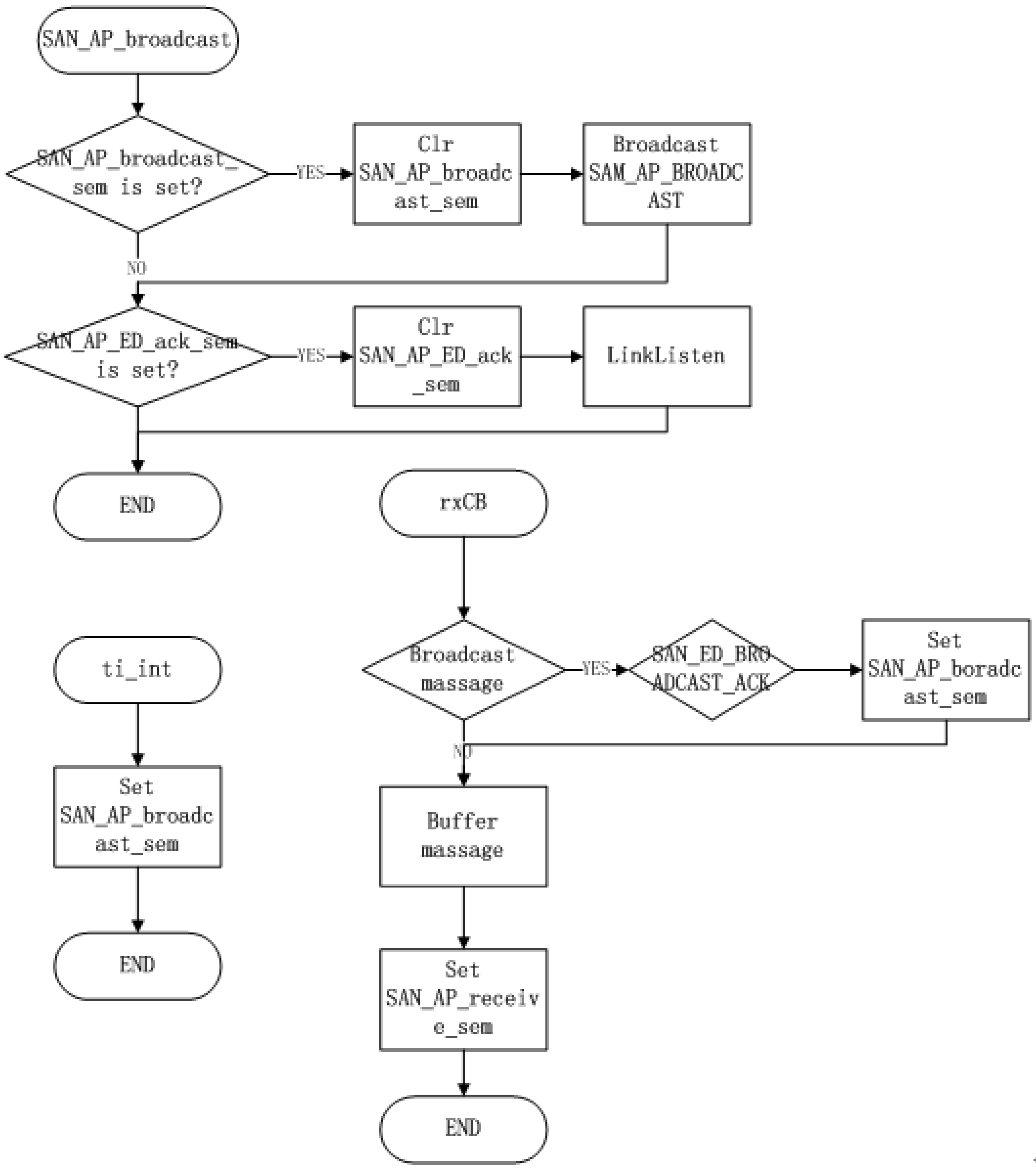


Figure 24: protocol stack flow chart

5.2.2.2 Periphery program

The only periphery need to program is UART0 protocol (9600, 8-N-1)

Uart0Update

Discretion	Should be executed frequently, check if the message receiving semaphore (SAN_AP_receive_sem) is set. Then format the information of the message and send it through uart0.
Prototype	Void uart0Update(void)
Parameters	
Return	

5.2.3 End device software

In the End device software, there is a protocol stack program for the ED and a temperature measurement program by using ADC.

5.2.3.1 Protocol stack program

There are three essential functions support ED SAN protocol and two API functions. (SAN_ED_broadcast.c)

SAN_ED_broadcast

Discretion	SAN_AP_broadcast should be executed frequently to make sure all the events generated by the interrupt can beprocessed on time.
Prototype	bool SAN_AP_broadcast(void)
Parameters	
Return	True Event processed successfully False No event.

rxCB

Discretion	The rxCB will be executed by the RX ISR. Once the AP receives information from wireless then rxCB will be executed. This function is mainly used to de-frame and response.
Prototype	Uint8_t rxCB(uint8_t port)
Parameters	Port the linked of captured frame.
Return	1

t1_int

Discretion	The ISR of timer1, an interrupt is generated every six seconds approximately.
Prototype	void t1_int(void)
Parameters	
Return	

SAN_ED_send

Discretion	An API function to send data to the AP.
Prototype	bool SAN_ED_send(uint8_t *msg)
Parameters	msg the allocation of data buffer
Return	True send success False send fails

SAN_ED_set_addr

Discretion	An API to set device address by using 4-bit digital swtich
Prototype	void SAN_ED_set_addr(void)
Parameters	
Return	

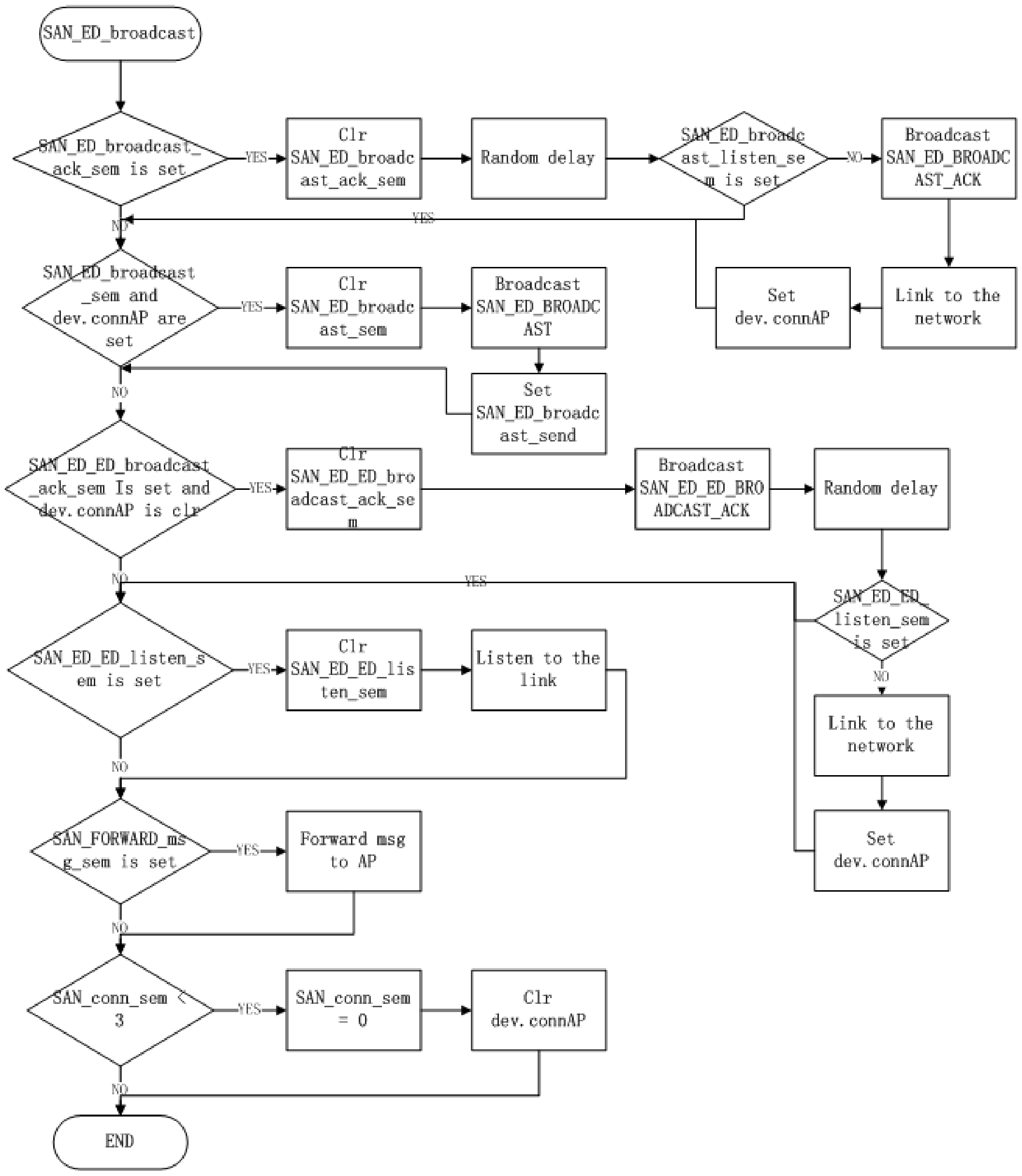

SAN_ED_broadcast
SAN_ED_broadcast_ack_sem is set
YES
Clr SAN_ED_broadcast_ack_sem
Random delay
SAN_ED_broadcast_listen_sem is set
NO
Broadcast SAN_ED_BROADCAST_ACK
YES
NO
Link to the network
Set dev.connAP
SAN_ED_broadcast_sem and dev.connAP are set
YES
Clr SAN_ED_broadcast_sem
Broadcast SAN_ED_BROADCAST
Set SAN_ED_broadcast_send
NO
SAN_ED_ED_broadcast_ack_sem Is set and dev.connAP is clr
YES
Clr SAN_ED_ED_broadcast_ack_sem
Broadcast SAN_ED_ED_BROADCAST_ACK
Random delay
NO
YES
SAN_ED_ED_listen_sem is set
NO
Link to the network
Set dev.connAP
SAN_ED_ED_listen_sem is set
YES
Clr SAN_ED_ED_listen_sem
Listen to the link
NO
SAN_FORWARD_msg_sem is set
YES
Forward msg to AP
NO
SAN_conn_sem < 3
YES
SAN_conn_sem = 0
Clr dev.connAP
NO
END

Figure 25: ED protocol stack flowchart 1

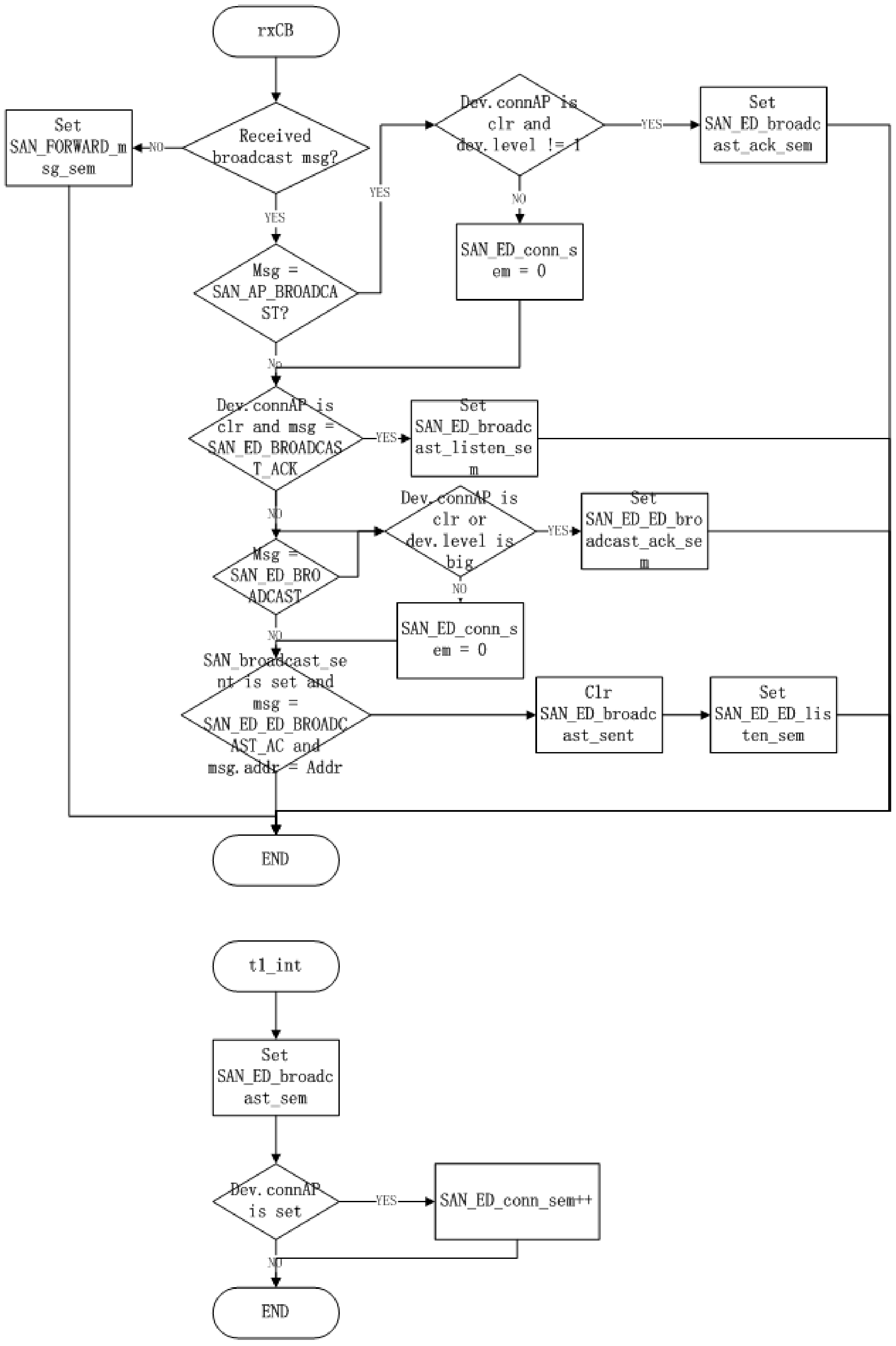
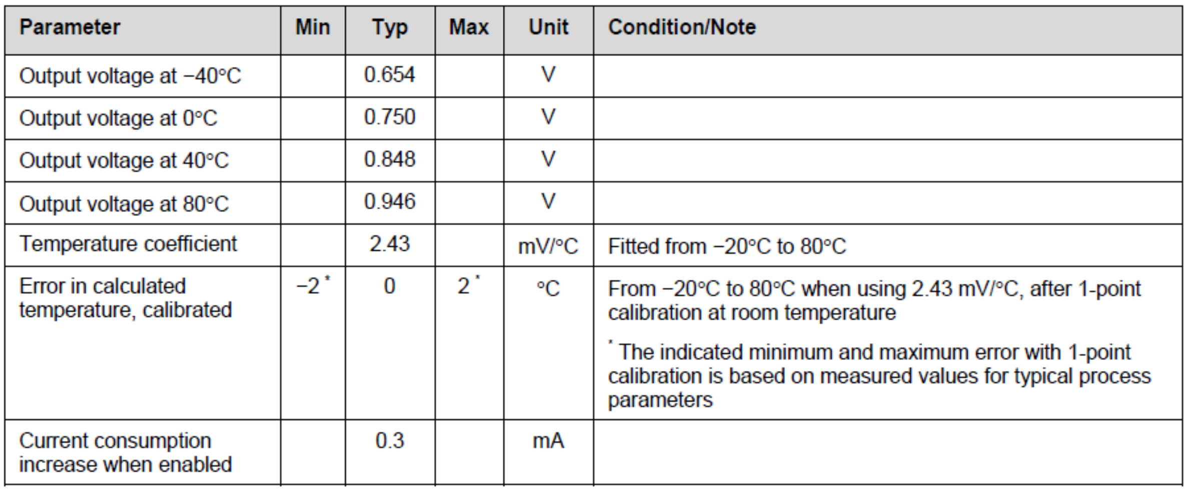

rxCB
Received broadcast msg?
NO
Set SAN_FORWARD_msg_sem
YES
Msg = SAN_AP_BROADCAST?
YES
Dev.connAP is clr and dev.level != 1
YES
Set SAN_ED_broadcast_ack_sem
NO
SAN_ED_conn_sem = 0
No
Dev.connAP is clr and msg = SAN_ED_BROADCAST_ACK
YES
Set SAN_ED_broadcast_listen_sem
NO
Msg = SAN_ED_BROADCAST
Dev.connAP is clr or dev.level is big
YES
Set SAN_ED_ED_broadcast_ack_sem
NO
SAN_ED_conn_sem = 0
NO
SAN_broadcast_sent is set and msg = SAN_ED_ED_BROADCAST_AC and msg.addr = Addr
Clr SAN_ED_broadcast_sent
Set SAN_ED_ED_listen_sem
END
t1_int
Set SAN_ED_broadcast_sem
Dev.connAP is set
YES
SAN_ED_conn_sem++
NO
END

Figure 26: ED protocol stack flowchart 2

5.2.3.2 Periphery program

There is an on-chip analog temperature sensor in CC2510. According to the datasheet (Figure 27), the temperature will be (temperature = Output Voltage / 2.4mV + 20).



Parameter	Min	Typ	Max	Unit	Condition/Note
Output voltage at −40°C		0.654		V	
Output voltage at 0°C		0.750		V	
Output voltage at 40°C		0.848		V	
Output voltage at 80°C		0.946		V	
Temperature coefficient		2.43		mV/°C	Fitted from −20°C to 80°C
Error in calculated temperature, calibrated	−2 *	0	2 *	°C	From −20°C to 80°C when using 2.43 mV/°C, after 1-point calibration at room temperature * The indicated minimum and maximum error with 1-point calibration is based on measured values for typical process parameters
Current consumption increase when enabled		0.3		mA	

Figure 27: On-chip temperature sensor datasheet

An ADC channel is used to input the signal of temperature sensor. The ADC uses 1.25V reference voltage, 10-bit accuracy setting and single sample mode. Then, Input Voltage goes (Input Voltage = 1.25V / 1024 * ADC Value). Because the Input Voltage of ADC is actually the Output Voltage of temperature sensor, the temperature can be calculated by the formula (temperature = ADC Value * 1.22 / 243 + 20). In the case of the temperature sensor is a non-linear analog device, a calibration gene is necessary. According to the experiment result, the calibration gene can be positive one. So, realistic temperature approximately equals to measurement temperature plus one. (Real temperature = ADC Value * 1.22 / 243 + 20 + 1)

In the program, the ISR timer 2 is used to set temperature measurement semaphore. About every seven seconds a temperature will be measured and send to the AP.

During the temperature transmitting, another thing can be mentioned. I used float type for the temperature value, but when you send it through wireless, uint8_t (unsigned char) must be used. In this case, a union is a good choice. It is known that a float type in 8-bit compute system is occupied four bytes. So the union can be organized like:

```
union temp_t
{
    float tempADC;
    uint8_t tempBYTE[4];
} temp;
```

By using above union, it is easy to access the temp allocated memory by different types.

6 System Test

The nominal range of a CC2510 with an Omni-direction antenna is approximately 20m, so it is hard to test the entire network realistically. By test purpose I made a network debug function in the EDs. Because the ED address is defined by a 4-bit digit switch, I made such a rule if the MSB of address is digital '1' then I assumed this ED is out side range of the AP. In another word, when the ED address is greater then '0x8' it is at least level 2 ED in the network.

6.1 Lab Test

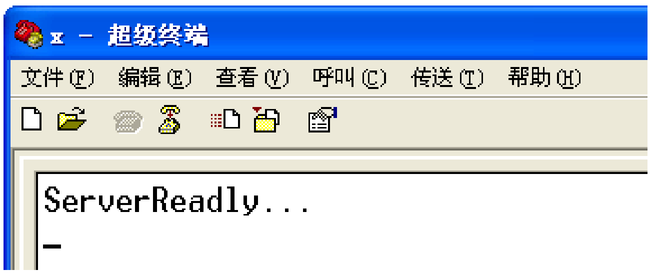
During the test, I set for EDs: 1, 6, C, E. C and E will ignore the SAN_AP_BORADCAST message from the AP, so they simulate the EDs out of the AP range.

A hyper terminal will monitor the message that AP forwarded.

Firstly, the HyperTerminal is set by 9600, 8-N-1 and connected to the AP, and all the EDs are power off.

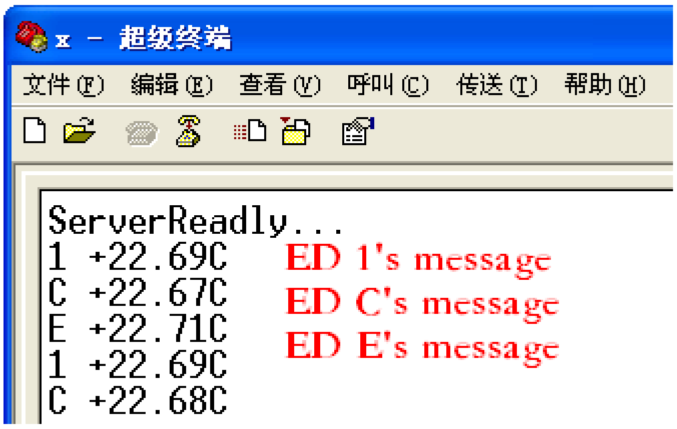
Then I power on C and E. Because of the Address rule, C and E cannot connect to the AP directly. So we got nothing so far. (Figure 28) Then ED 1 is powered on. After ten seconds all these three EDs message are captured. (Figure 29) It means, C and E are connected to the AP by the routing of ED 1. Then ED 6 is powered on. (Figure 30) So far, all the EDs are working well, the network is setup. Now, I powered of ED 1 manually. For a while, no messages from C and E are received. After about ten seconds C and E back to the network automatically. (Figure 31)

In this test, a one-hop network simulation has been proved only. Because the whole system is based on one protocol stack so one-hop network can reflect how the multi-hop network behaves.



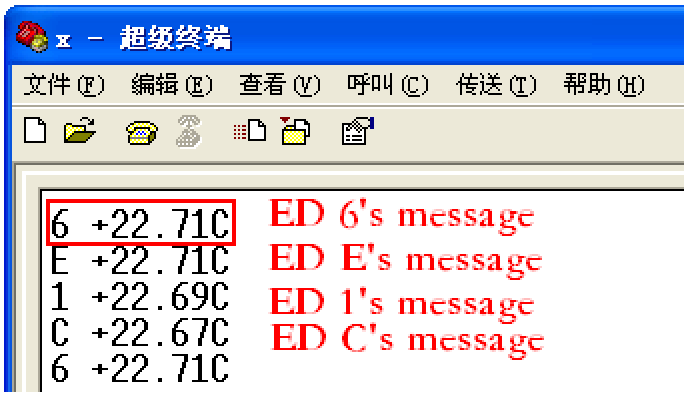
```
ServerReadly...
_
```

Figure 28: Test 1 (when system is initializing, "ServerReadly…" is shown.)



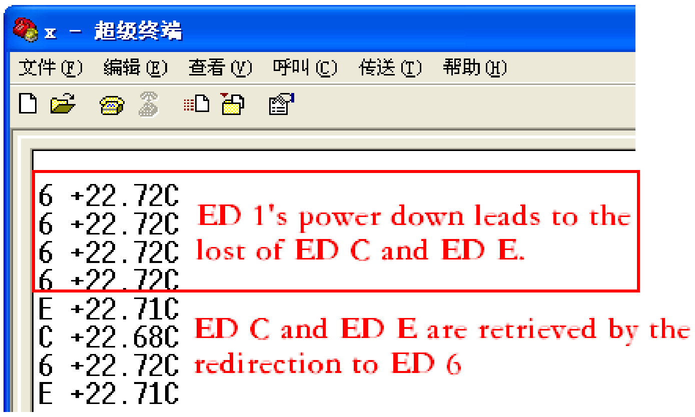
```
ServerReadly...
1 +22.69C
C +22.67C
E +22.71C
1 +22.69C
C +22.68C
```

Figure 29: Test 2 (ED 1 is directly connected to AP. ED C and ED E is one-hop connected to AP with the route of ED 1. From the screen shot, all the nodes are well communicating with AP. The System works fine.)



```
6 +22.71C
E +22.71C
1 +22.69C
C +22.67C
6 +22.71C
```

Figure 30: Test 3 (Now the last node, ED 6 is joined in the group. So far, all the nodes are connected to the AP (ED 1 and ED 6 is directly connect to AP; ED C and ED E is one-hop routed by ED 1 to connect to AP.)


x - 超级终端
文件(F) 编辑(E) 查看(V) 呼叫(C) 传送(T) 帮助(H)
ED 1's power down leads to the lost of ED C and ED E.
ED C and ED E are retrieved by the redirection to ED 6

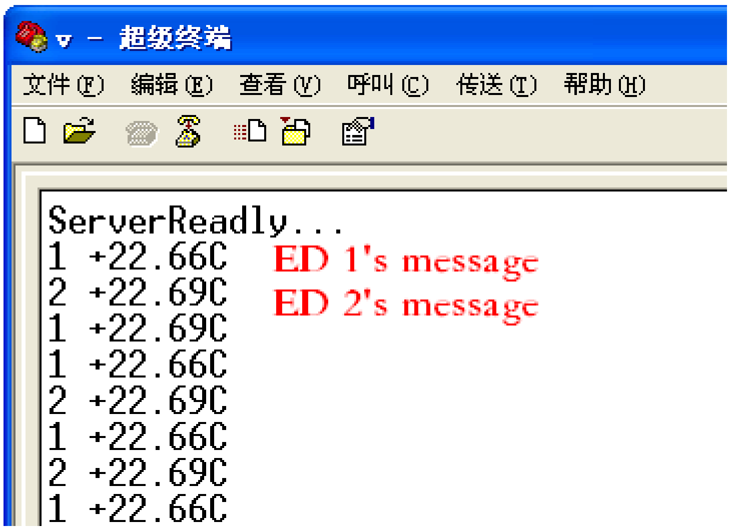
```
6 +22.72C
6 +22.72C
6 +22.72C
6 +22.72C
E +22.71C
C +22.68C
6 +22.72C
E +22.71C
```

Figure 31: Test 4 (Then we shut down ED 1 which is routing ED C and ED E. In this case, ED C and ED E are lost. After awhile, with the self-healing, ED C and ED E are self-reconfigured into system again.)

6.2 Practical Test

Another test goes that, there will be am Access Point and two different End Devices. The first ED with ID 1 is allocated 15 meters away from AP, the other ED with ID 2 will move from AP to the ED 1 and continue move until no data can be captured from ED 2.

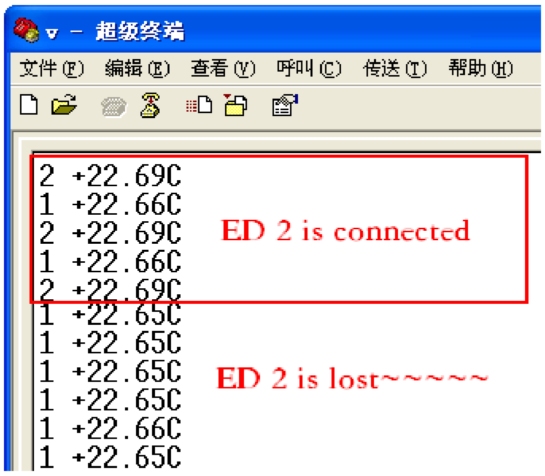
At the beginning two EDs are all connected to the AP (Figure 32)



```
ServerReadly...
1 +22.66C
2 +22.69C
1 +22.69C
1 +22.66C
2 +22.69C
1 +22.66C
2 +22.69C
1 +22.66C
```

Figure 32: Test 5 (System is initialized and ED 1 and ED 2 are directly connected to AP (no hop), because they are all in the range of AP.)

Once we move ED 2 over 20 meters from the AP, the ED 2 is lost (Figure 33)

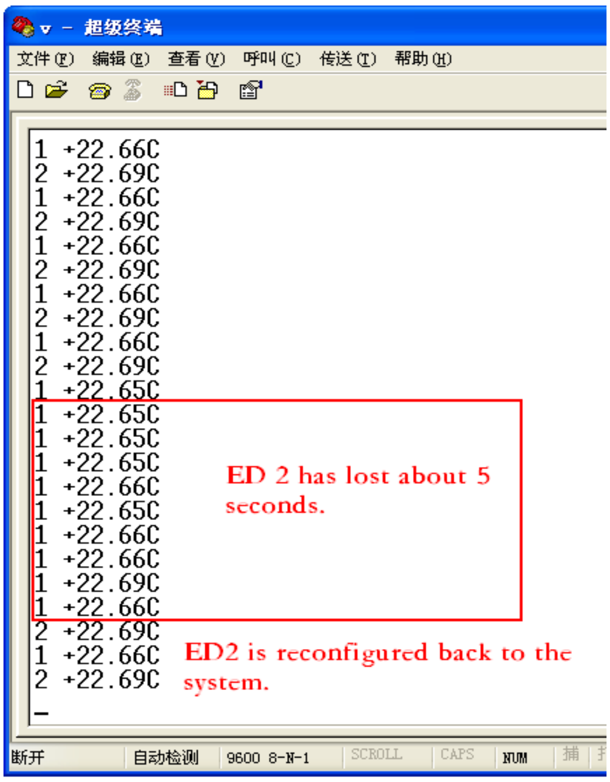


```
2 +22.69C
1 +22.66C
2 +22.69C
1 +22.66C
2 +22.69C
1 +22.65C
1 +22.65C
1 +22.65C
1 +22.65C
1 +22.66C
1 +22.65C
```

ED 2 is connected

ED 2 is lost~~~~~

Figure 33: Test 6 (Once ED 2 goes out of the range of AP (about 20 meters), is lost.)



```
1 +22.66C
2 +22.69C
1 +22.66C
2 +22.69C
1 +22.66C
2 +22.69C
1 +22.66C
2 +22.69C
1 +22.66C
2 +22.69C
1 +22.65C
1 +22.65C
1 +22.65C
1 +22.65C
1 +22.66C
1 +22.65C
1 +22.66C
1 +22.66C
1 +22.69C
1 +22.66C
2 +22.69C
1 +22.66C
2 +22.69C
_
```

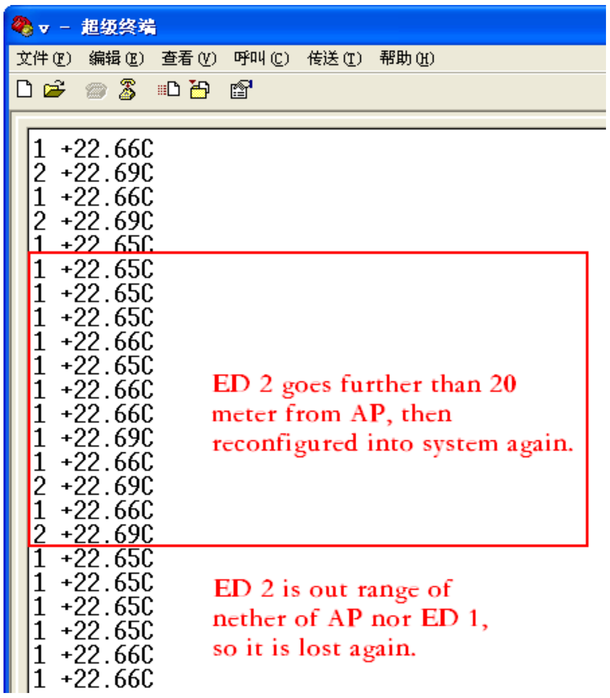
ED 2 has lost about 5 seconds.

ED2 is reconfigured back to the system.

Figure 34: Test 7 (When ED 2 was out range of AP, it cannot establish a directly connection with AP, then it would lost from the system.Because of the self-reconfiguration of the protocol, ED 2 was successfully to found a route from ED 1 to do one-hop connection with AP.)

So stop for awhile about 5 seconds, ED2 is back to the system (Figure 34)

Then ED 2 continue move, about 35 meters from AP and 20 meters from ED1, it is lost again. (Figure 35)

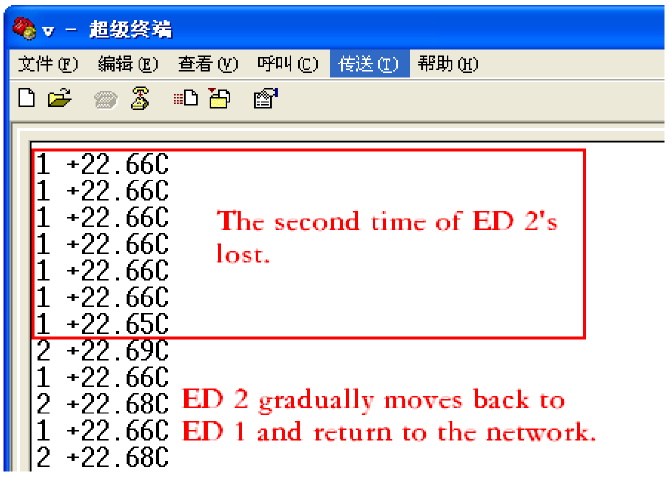


```
1 +22.66C
2 +22.69C
1 +22.66C
2 +22.69C
1 +22.65C
1 +22.65C
1 +22.65C
1 +22.65C
1 +22.66C
1 +22.65C
1 +22.66C
1 +22.66C
1 +22.69C
1 +22.66C
2 +22.69C
1 +22.66C
2 +22.69C
1 +22.65C
1 +22.65C
1 +22.65C
1 +22.65C
1 +22.66C
1 +22.66C
```

ED 2 goes further than 20 meter from AP, then reconfigured into system again.

ED 2 is out range of nether of AP nor ED 1, so it is lost again.

Figure 35: Test 8 (ED 2 is keep move against AP and ED 1, it was lost once more because of the distance between ED 1 and ED 2 is out of bound.)



```
1 +22.66C
1 +22.66C
1 +22.66C
1 +22.66C
1 +22.66C
1 +22.66C
1 +22.65C
2 +22.69C
1 +22.66C
2 +22.68C
1 +22.66C
2 +22.68C
```

The second time of ED 2's lost.

ED 2 gradually moves back to ED 1 and return to the network.

Figure 36: Test 9 (Once ED 2 goes back in to the range of ED 1, it joins into the system automatically.)

Then we move ED 2 back to AP, at 17meters from ED1, ED 2 is found. (Figure 36)
After that, ED 2 is moved back to AP and it is still there. (Figure 36)

So far as I test, I can still the system works nice.

7. FUTURE WORKS

So far, on this project, the basic feature of a self-organized network is done. But there are some known bugs and disadvantages left as future tasks.

7.1 Bidirectional task

A fatal disadvantages of SAN protocol is none bidirectional communication support. When the protocol is on the paper, based on the specialization of the problem mentioned at the beginning (forest temperature measurement), no talk from the AP to the EDs needed. Additionally, think about realistic situation, in a forest temperature measurement sensor network there are thousands of EDs. If we just list a route table in the protocol stack, it will take so much memory of RAM. But it is good to have bidirectional communication feature in a network. So, in the future, it is a good task to find appropriate algorithms to list such a routing table of EDs.

7.2 A reset bug

An already known bug is when a device (AP or ED) is reset, and then the other EDs routed by this device (AP or ED) are lost. That because an ED grantee itself still in the network is to listen the SAN_ED_BROADCAST (or SAN_AP_BROADCAST) of its next-hop. Once an ED is reset, the link between it and the ED behind it is down, but it still keep broadcast itself, so the ED behind still think they are in the network, so it will not recover the link. In this case, it is lost.

A sample solution of this bug is to add a serial number in the frame of device broadcast and after every broadcast it increases. Then the ED behind it checks the number every time. So if this ED reset, the serial number is not correct, so the ED behind it will notice and recover the link.

7.3 A multi link bug

The second already known bug is once two EDs try to link a same device (AP or ED) simultaneously, the network will become unpredictable and unstable. Although, there is a random delay and listening to other broadcast ACK before link to an ED or an AP protocol is implemented. Like Figure 37, B and C want to link A, but B and C cannot hear each other, which means B or C cannot receive the broadcast ACK from C or B. So, they may link to A at the same time to generate unpredictable bug.

A sample solution of this issue is once A received the ACK from B or C, before listen to the link broadcasts one more message with a special device ID of B or C. In this cast both of B and C can hear it can check the device ID of the message to link to A.

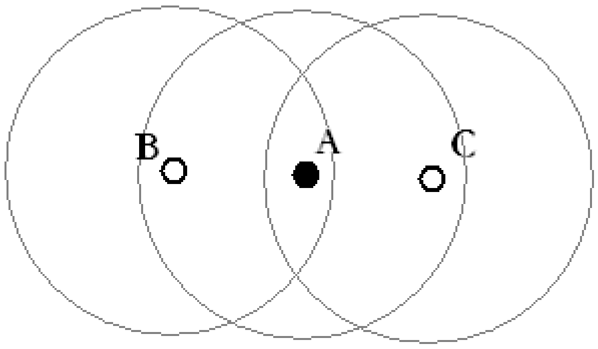


Figure 37: Multi-link bug

7.4 Frequency Agility

Because the system is used of 2.4GHz with the max output power 1dbm, it is easily jammed by other network which works in the same bandwidth like Wi-Fi and Bluetooth. So a feature of frequency agility is necessary. The future version of SAN network will have frequency agility feature.

8. THE CONCLUSION

As a solution of forest temperature measurement, this system fulfilled such features:

- Self-organized network
- Low power consumption (if LED is not used, the working current is less then 1mA.)
- Non-expansive (single CC2510 cost 5 Euros; one ED device, if evaluation module is not used, will less than 10 Euros.)

As protocol research, this protocol stack can be used for the big range sensor network. And this protocol is actually a device base protocol, because it uses the hardware feature of the device (timer). So far, it is impossible to transplant it to other devices (except CC2511, CC1110, and CC1111because they are all use the same kind of MCU) without any changes.

Anyway, the Self-organized network is still on the develop step, it is not standardized so far. So my project is just an attempt in this area.

9. LITERATURE

1 Texas Instrument, CC1110DKCC2510DK -- Development Kit User Manual (Rev. A, 2007).

2 Texas Instrument, CC2510-CC2511DK Quick Start, Reversion 2.1, 14/9/2007.

3 Texas Instrument, cc2511DK_datasheet, 2007.

4 Texas Instrument, CC2510Fx/CC2511Fx, 2007.

5 Texas Instrument, SimpliciTI API, 2008.

6 Texas Instrument, SimpliciTI Developers Notes, 2008.