

Xilinx Zynq FPGA, TI DSP, MCU
기반의 프로그래밍 및 회로 설계
전문가 과정

ESP8266 Connect WIFI

학생 : 김시윤
강사 : Innova Lee(이 상훈)

- 서론 -

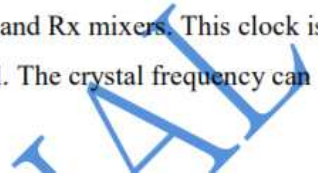
이 절차를 실행 하기 전 '오덴님'의 ESP8266 펌웨어 업로드 절차를 완료 후 진행하도록 한다. 여기서는 TMS57LC4357으로 ESP8266에 업로드 된 펌웨어로 UART를 하도록 한다.

1. ESP8266 UART

ESP8266이 지원하는 UART통신 주파수부터 알아보고자 한다.
ESP8266의 데이터 시트를 보면 다음과 같이 나와있다.

11.1 High Frequency Clock

The high frequency clock on ESP8266 is used to drive both the Tx and Rx mixers. This clock is generated from the internal crystal oscillator and an external crystal. The crystal frequency can range from 26MHz to 52MHz.



11.1 고주파 클럭

ESP8266의 고주파 클럭은 Tx 및 Rx 믹서를 구동하는 데 사용됩니다. 이 클럭은 내부 크리스탈 발진기와 외부 크리스탈로 생성됩니다. 크리스탈 주파수는 26MHz ~ 52MHz 범위 사이입니다.

이와 같이 나와있다. ESP8266 의 UART Interface user guide 에는 이 클럭 주파수는 보우 (baud rate)를 결정한다.

During this period, the baud rate of print information is related to the frequency of the external crystal applied. When the crystal frequency is 40MHz, the baud rate of the print information will be 115200, while when the crystal frequency is 26MHz, the baud rate will be 74880

앞단계에서 펌웨어를 업그레이드 시킨 이유는 Baud Rate를 낮추기 위함이다.
우리가 앞단계에서 업로드한 펌웨어는 9600 Baud rate를 지원하는 펌웨어이다.

우리는 UART를 이용하여 ESP8266을 AT명령어로 제어해야한다.

2. AT 명령어

AT 명령어 정리표

<http://room-15.github.io/blog/2015/03/26/esp8266-at-command-reference/#AT>

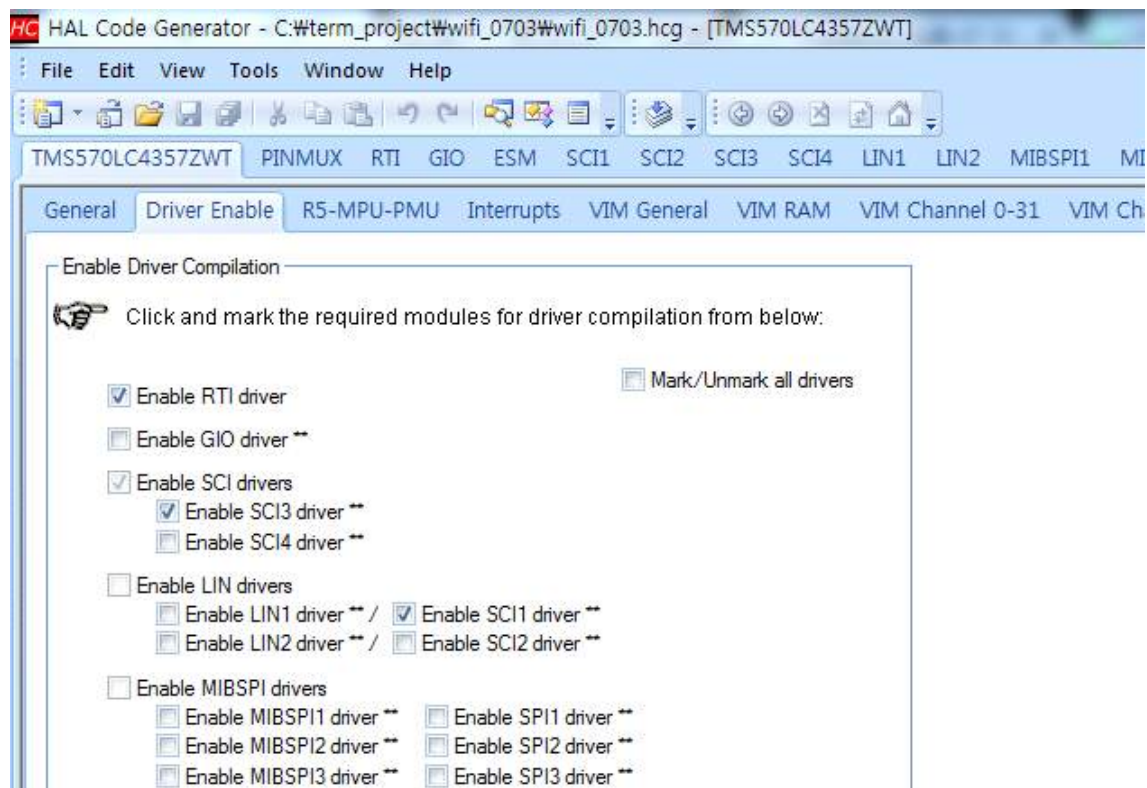
이미 펌웨어를 업그레이드를 했다면 아두이노로 AT명령어를 해보았을 것이다.

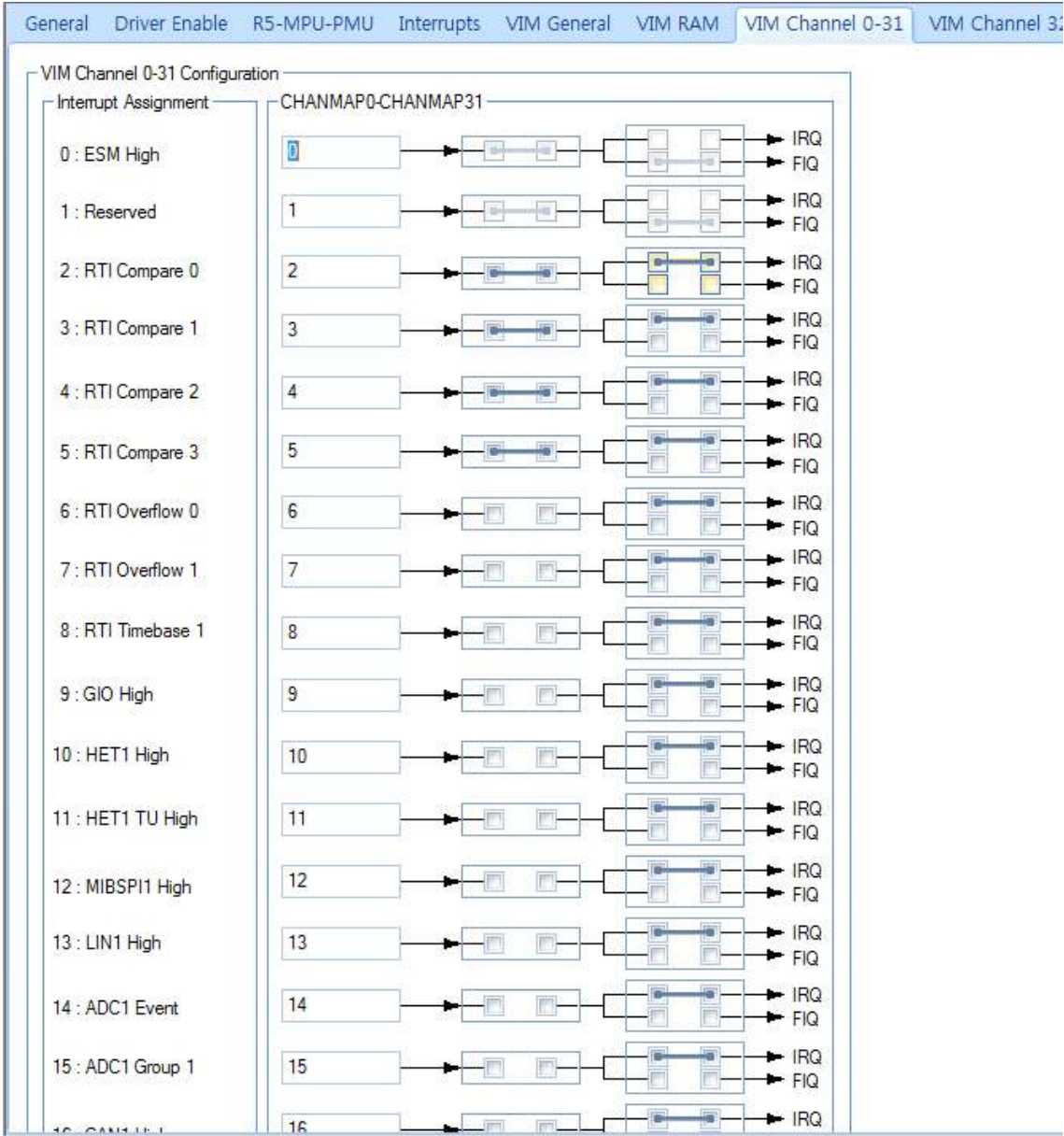
여기서는 AT명령어를 통해 Wi-Fi에 연결하고 데이터를 송수신 해야하기 때문에 위 AT명령어를 정리한 사이트를 참고하고 다음으로 넘어가도록 하자.

3. TMS57LC4357 & ESP8266 UART

소스코드 작성에 앞서 HALCOGEN 설정을 먼저 하도록 한다.

열심히 삽질 결과, 소스코드 작성시 delay를 주지 않으면 ESP8266에서 인식을 못하는 것을 확인 하였다. Delay가 필요하지만 정확한 Delay를 주기위해 RTI 인터럽트를 사용하도록 하고, ESP8266 <-> MCU UART와 MCU <-> USB UART 총 두 개가 필요하므로, SCI1 과 SCI3을 사용하도록 하고 HALCOGEN 설정에 들어간다.





TMS570LC4357ZWT PINMUX RTI GIO ESM SCI1 SCI2 SCI3 SCI4 LIN1 LIN2

Pin Muxing Input Pin Muxing Special Pin Muxing

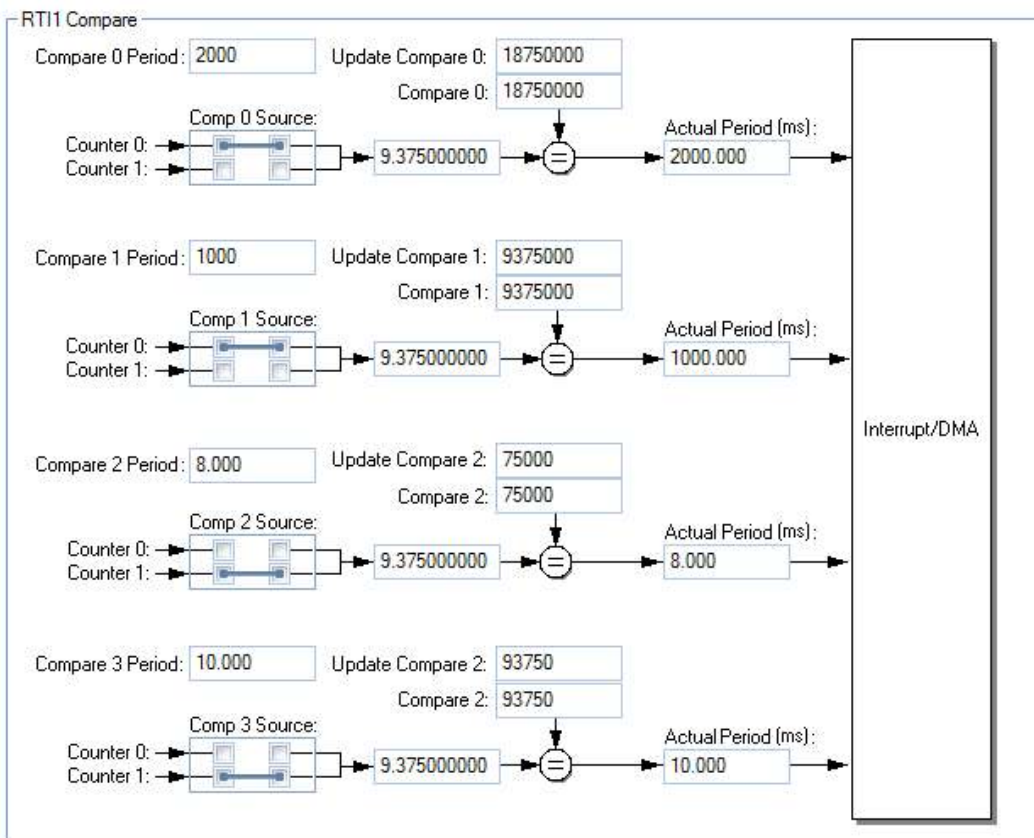
Enable / Disable Peripherals

<input type="checkbox"/> HET1	<input type="checkbox"/> GIOA	<input type="checkbox"/> MIBSPI2	<input type="checkbox"/> MIBSPI1	<input checked="" type="checkbox"/> SCI3	<input type="checkbox"/> RMI
<input type="checkbox"/> HET2	<input type="checkbox"/> GIOB	<input type="checkbox"/> MIBSPI4	<input type="checkbox"/> MIBSPI3	<input type="checkbox"/> SCI4	<input type="checkbox"/> MII
<input type="checkbox"/> EMIF	<input type="checkbox"/> EQEP	<input type="checkbox"/> AD1EVT	<input type="checkbox"/> MIBSPI5	<input type="checkbox"/> LIN2/SCI2	<input type="checkbox"/> CAN4
<input type="checkbox"/> ETPWM	<input type="checkbox"/> ECAP	<input type="checkbox"/> AD2EVT	<input type="checkbox"/> I2C1	<input type="checkbox"/> I2C2	

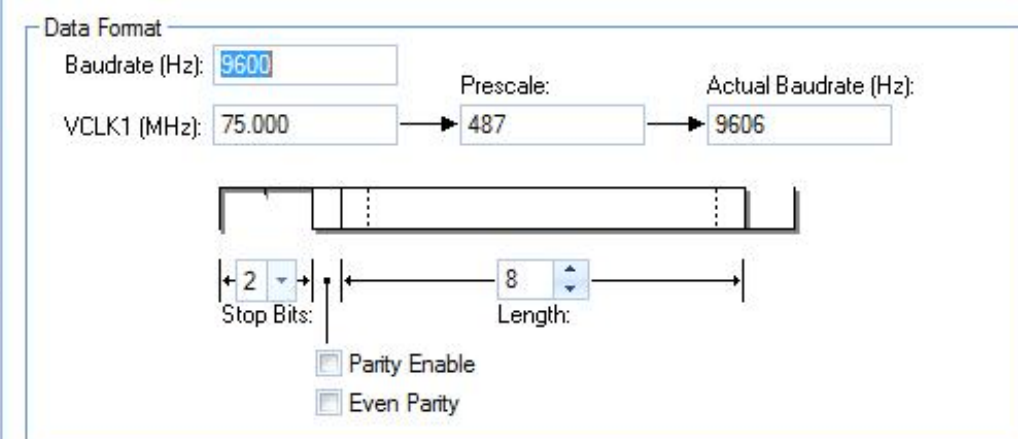
Note: GIO and MII are not available in RMI and Spec.

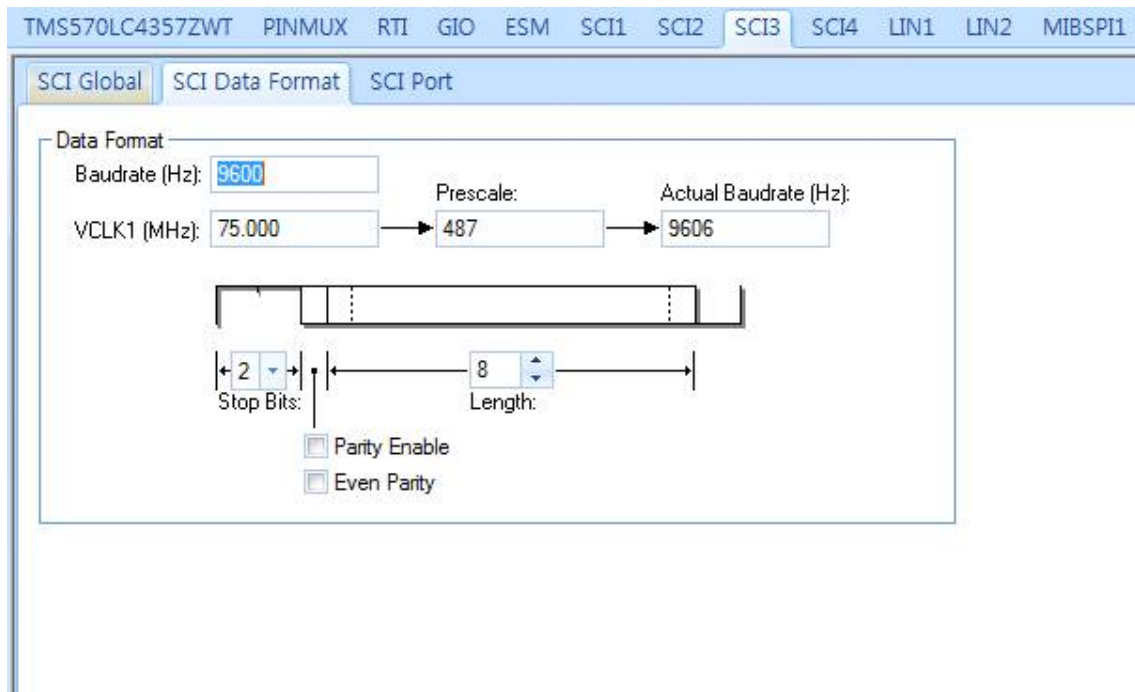
Roll Default Mux Mux Option 1 Mux Option 2 Mux Option 3

RTI1 General RTI1 Counter 0 RTI1 Counter 1 RTI1 Compare



SCI Global SCI Data Format SCI Port





위의 설정을 맞추었다면 코드 제너레이트를 하도록 한다.

CCS ESP8266 UART CODE

```
#include <HL_reg_sci.h>
#include <HL_rti.h>
#include <HL_sci.h>
#include <HL_sys_core.h>

uint8 i = 0;
uint8 flag = 0;
uint8 time=0;
uint8 data[100]= {0};

void connect_WIFI()
{
    sciSend(sciREG3, 40, "AT+CWJAP=\"Android2573\", \"00000000\"\\r\\n");
}

void rtiNotification(rtiBASE_t *rtiREG, uint32 notification)
{
    switch(flag)
```

```

    {
        case 0:
            sciSend(sciREG3, 4, "AT\r\n");
            flag++;
            break;

        case 1:
            sciSend(sciREG3, 8, "AT+GMR\r\n");
            flag++;
            break;

        case 2:
            sciSend(sciREG3, 13, "AT+CWMODE=1\r\n");
            flag++;
            break;

        case 3:
            sciSend(sciREG3, 12, "AT+CWMODE?\r\n");
            flag++;
            break;

        case 4:
            connect_WIFI();
            flag++;
            break;

        default :
            rtiDisableNotification(rtiREG1, 1);
            break;
    }
}

```

```

void main(void)
{
    sciInit();
    rtiInit();
}

```

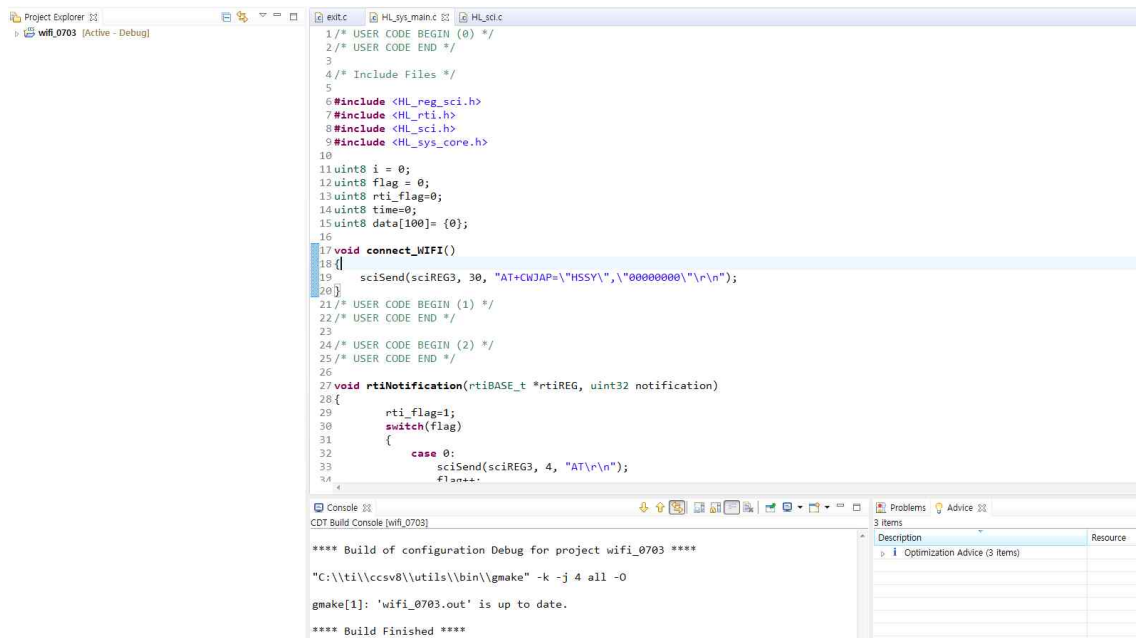
```

rtiEnableNotification(rtiREG1, 1);
rtiStartCounter(rtiREG1, rtiCOUNTER_BLOCK0);

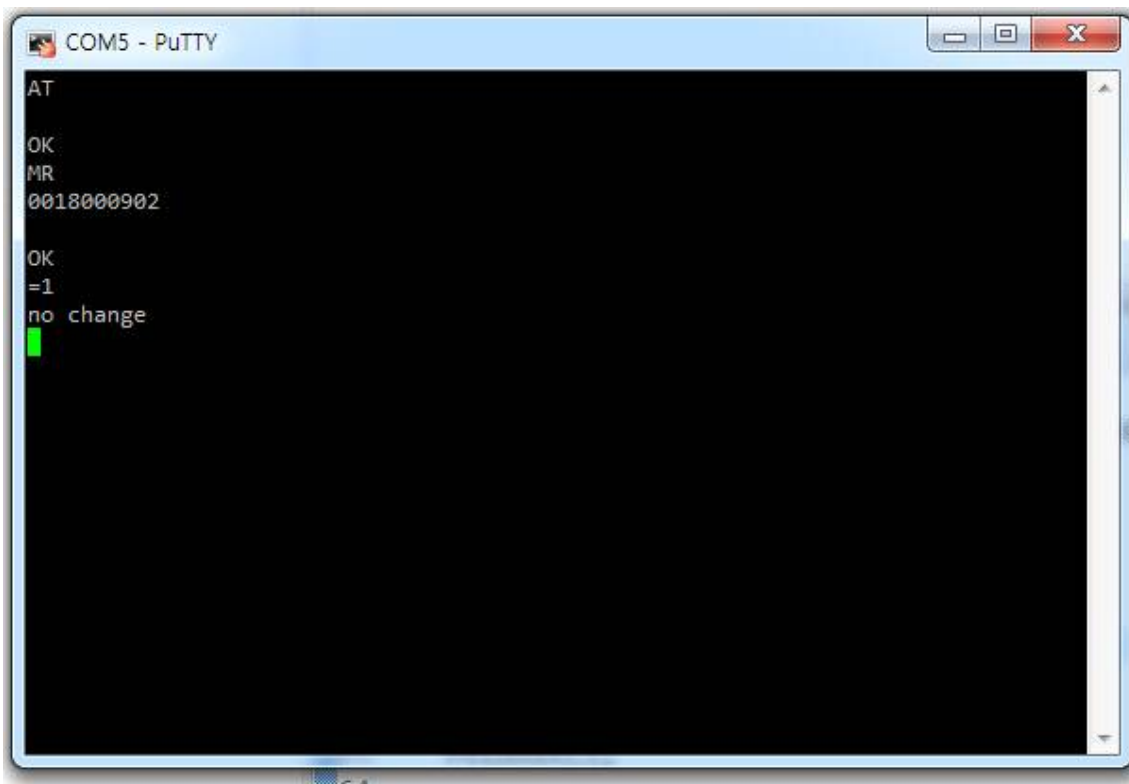
_enable_IRQ_interrupt_();

while(1)
{
    sciSendByte(sciREG1, sciReceiveByte(sciREG3));
}
}

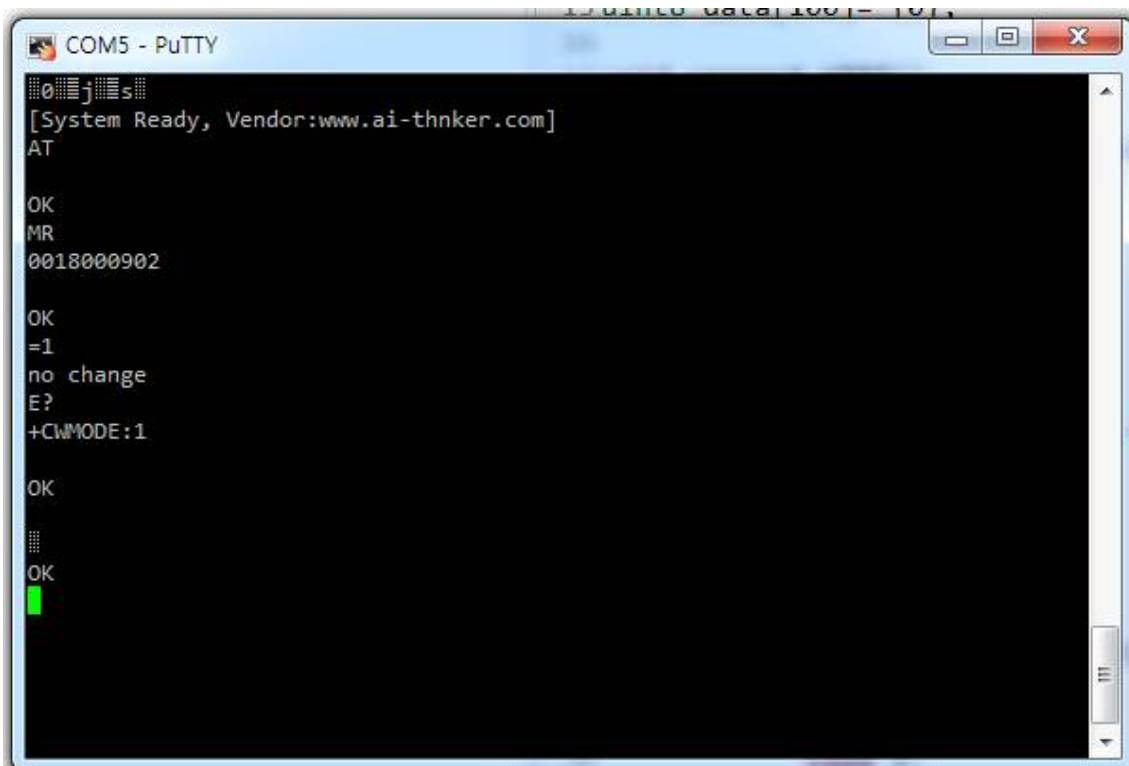
```



위의 코드를 작성하였으면 디버깅 모드로 업로드를 진행하거나 flash를 하여 바로 보드에 업로드 하고, PUTTY를 실행시키도록 한다.



```
COM5 - PuTTY
AT
OK
MR
0018000902
OK
=1
no change
█
```



```
COM5 - PuTTY
[System Ready, Vendor:www.ai-thnker.com]
AT
OK
MR
0018000902
OK
=1
no change
E?
+CWMODE:1
OK
█
```

마지막이 Wi-Fi 연결하는 AT COMMAND이다 휴대폰 HotSpot HOST 이름과 비밀번호 또는 공유가 HOST이름과 비밀번호 입력후 마지막 OK가 뜨고 연결된걸 확인하면 성공이다. 나머지는 AT 명령어 매뉴얼을 보며 공부해보도록 한다.

< 설정

개인용 핫스팟

개인용 핫스팟



이제 인식 가능합니다.

다른 사람들이 Wi-Fi 및 Bluetooth를 통해 'HSSY'(이)라는 이름으로 사용자의 공유 네트워크를 찾을 수 있습니다.

Wi-Fi 암호

00000000 >



Wi-Fi(으)로 연결하려면

- 1 컴퓨터 또는 기타 기기의 Wi-Fi 설정에서 'HSSY'을(를) 선택하십시오.
- 2 요청을 받으면 암호를 입력합니다.



BLUETOOTH로 연결하려면

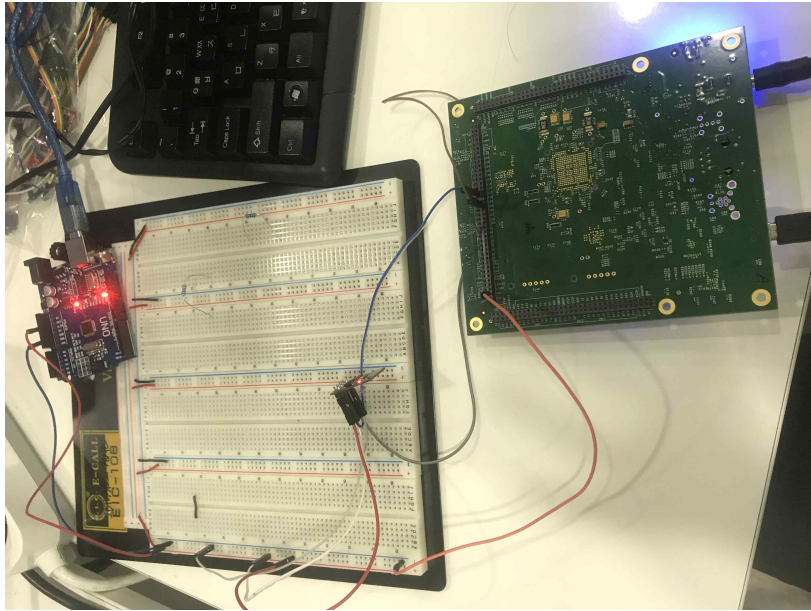
- 1 iPhone을 컴퓨터와 쌍으로 연결합니다.
- 2 iPhone에서 쌍으로 연결을 탭하거나 컴퓨터에 표시된 코드를 입력합니다.
- 3 컴퓨터에서 iPhone으로 연결합니다.



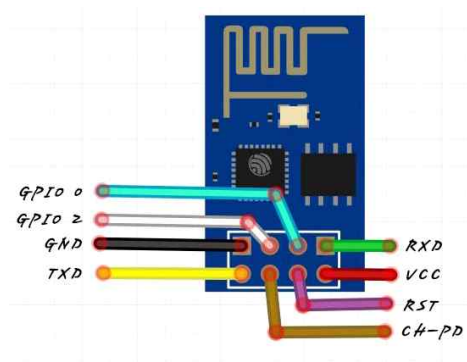
USB로 연결하려면

- 1 iPhone을 컴퓨터에 연결합니다.
- 2 설정에 있는 네트워크 서비스 목록에서 iPhone을 선택합니다.

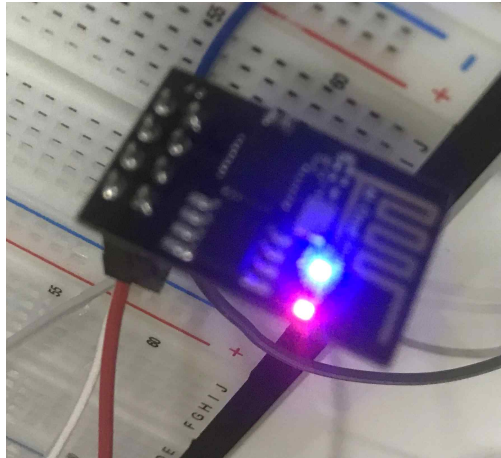
최종 연결성공 확인사진



----- 회로구성 -----



ESP8266	MCU
RXD	SCI3 TX(N2)
TXD	SCI3 RX(W3)
GND	GND(아두이노 & MCU)
VCC	3.3V(아두이노 or MCU GIO 출력 *아두이노 추천)
GPIO 0	X
GPIO 2	X
CH-PD	3.3V
RST	X



UART 통신을 성공하면 파란색 LED에 불이 깜박거린다.
실패하면 빨간불만 들어온다.

ESP8266 Open TCP/IP SERVER

앞장에서 ESP8266을 이용하여 공유기 Wi-Fi에 연결을 하였다.

이번에는 ESP8266으로 TCP/IP 서버를 열어 Client를 접속시켜 데이터를 받아보도록 한다.

일단 들어가기 앞서 AT-Comaand 명령어를 먼저 정리하고 넘어가도록 한다.

명령어	동작
AT	연결 테스트. 연결 성공 시 응답 = OK
AT+RST	모듈 리셋 명령어 리셋 성공 시 응답 = OK
AT+GMR	모듈 버전정보 응답 = 버전정보\n OK
AT+CIPMODE=0	모듈 전송모드 설정 0은 노말모드 바뀌었으면 응답 OK, 변동없으면 no change
AT+CWJAP="ssid", "pw"	공유기 AP에 연결 연결 시 응답 OK
AT+CIPCLOSE	열려있는 서버가 있으면 닫아주는 명령어
AT+CIPMUX=1	단일연결인지 멀티 연결인지 정해줌 1은 멀티연결 응답: OK
AT+CIPSERVER=1,777	서버 create 명령어 1은 create 0은 delete , 포트번호
AT+CIFSR	ESP8266의 로컬 IP와 공유기에서 할당받은 IP출력
AT+CIPSTO=1000	서버 타임아웃 설정

할코젠 설정은 GIO를 추가하여 다시 설정 해 주도록 한다.

GIOA의 0을 출력으로 설정하고 핀믹스에 GIOA를 활성화 시킨 후 코드 제너레이트를 하도록 한다.

위의 AT명령어를 통해 서버 오픈하는 소스코드부터 먼저 짜보면 다음과 같다.

```
#include "HL_reg_sci.h"
#include "HL_rti.h"
#include "HL_sci.h"
#include "HL_sys_core.h"
#include "HL_gio.h"
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

uint8 flag = 1; //진행절차 상태 변수

void rtiNotification(rtiBASE_t *rtiREG, uint32 notification)
{
    switch(flag) {
        case 0:
```

```

        sciSend(sciREG3,8 ,"AT+RST \r \n");
        flag++;
        break;
    case 1:
        sciSend(sciREG3, 4, "AT \r \n");
        flag++;
        break;
    case 2:
        sciSend(sciREG3,9 , "AT+GMR \r \n");
        flag++;
        break;
    case 3:
        sciSend(sciREG3,14,"AT+CIPMODE=0 \r \n");
        flag++;
        break;
    case 4:
        sciSend(sciREG3,
        34,"AT+CWJAP= \"josephahn\", \"00000000 \" \r \n");
        flag++;
        break;
    case 5:
        sciSend(sciREG3, 13, "AT+CIPCLOSE \r \n");
        flag++;
        break;
    case 6:
        sciSend(sciREG3,13,"AT+CIPMUX=1 \r \n");
        flag++;
        break;
    case 7:
        sciSend(sciREG3,25 ,"AT+CIPSERVER=1,777 \r \n");
        flag++;
        break;
    case 8:
        sciSend(sciREG3,10 , "AT+CIFSR \r \n");
        flag++;
        break;
    case 9:
        sciSend(sciREG3, 16, "AT+CIPSTO=1000 \r \n");
        flag++;
        break;
    default :
        rtiDisableNotification(rtiREG1, 1);
        break;
    }
}

void main(void)
{
    sciInit();
    rtiInit();
    rtiEnableNotification(rtiREG1, 1);
    rtiStartCounter(rtiREG1, rtiCOUNTER_BLOCK0);
    _enable_IRQ_interrupt_();
    while(1)
    {
        sciSendByte(sciREG1,
        sciReceiveByte(sciREG3));
    }
}

```

서버에 접속 시 밑에 할당받은 ip로 접속해야 접속이 가능하다.

```

SR
192.168.4.1
172.20.10.11
OK
00

OK
ND

ERROR

+IPD,0,2:g
OK

+IPD,0,2:g
OK

+IPD,0,21:ffffggsshdhdjdjfjfjf
OK

+IPD,0,9:i love u
OK
Unlink

```

데이터가 두 개인것은 g_n 이 보내졌기 때문이다.

이제 클라이언트로 LED를 제어해보도록한다.

아까 GIO를 활성화 시켰기 때문에 GIO를 이용하여 LED를 제어하는 코드를 작성하면 다음과 같다.

```
#include "HL_reg_sci.h"
#include "HL_rti.h"
#include "HL_sci.h"
#include "HL_sys_core.h"
#include "HL_gio.h"

uint8 flag = 1;
uint8 setting_flag=0;

void rtiNotification(rtiBASE_t *rtiREG, uint32 notification)
{
    switch(flag){
        case 0:
            sciSend(sciREG3,8 ,"AT+RST\r\n");
            flag++;
            break;
        case 1:
            sciSend(sciREG3, 4, "AT\r\n");
            flag++;
            break;
        case 2:
            sciSend(sciREG3,9 , "AT+GMR\r\n");
            flag++;
            break;
        case 3:
            sciSend(sciREG3,14,"AT+CIPMODE=0\r\n");
            flag++;
            break;
        case 4:
            sciSend(sciREG3, 34, "AT+CWJAP=\"josephahn\", \"00000000\" \r\n");
            flag++;
            break;
        case 5:
            sciSend(sciREG3, 13, "AT+CIPCLOSE\r\n");
            flag++;
```



```

        break;
    case 6:
        sciSend(sciREG3,13,"AT+CIPMUX=1\r\n");
        flag++;
        break;
    case 7:
        sciSend(sciREG3,25 , "AT+CIPSERVER=1,777\r\n");
        flag++;
        break;
    case 8:
        sciSend(sciREG3,10 , "AT+CIFSR\r\n");
        flag++;
        break;
    case 9:
        sciSend(sciREG3, 16, "AT+CIPSTO=1000\r\n");
        flag++;
        break;
    default :
        setting_flag = 1;
        rtiDisableNotification(rtiREG1, 1);
        break;
    }
}

void main(void)
{
    gioInit();
    sciInit();
    rtiInit();

    rtiEnableNotification(rtiREG1, 1);
    rtiStartCounter(rtiREG1, rtiCOUNTER_BLOCK0);

    _enable_IRQ_interrupt_();

    while(1)
    {
        if(setting_flag == 0)
            sciSendByte(sciREG1, sciReceiveByte(sciREG3));
        else if(setting_flag == 1)

```

```

    {
        if(sciReceiveByte(sciREG3) == 'a')
            gpioSetBit(gioPORTB, 0, 1);

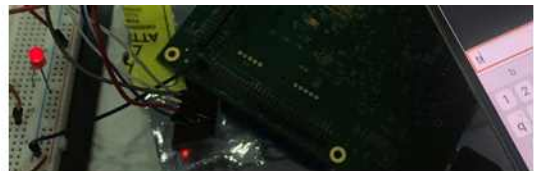
        sciREG3->RD = 0;

        if(sciReceiveByte(sciREG3) == 's')
            gpioSetBit(gioPORTB, 0, 0);
    }
}

```

여기서 sciREG3->RD = 0;는 Receive에서 데이터를 받아 저장하는 버퍼이다.
버퍼를 0으로 초기화 시킨 것이다.

0으로 초기화 하지 않을 경우 데이터를 두 번 보내야 인식을 하는 오류가 생겨 0으로 초기화
시켜 주었다.



실행결과는 위의 사진과 같다.

a를 보내면 LED가 켜지고 b를 보내면 LED가 꺼진다.