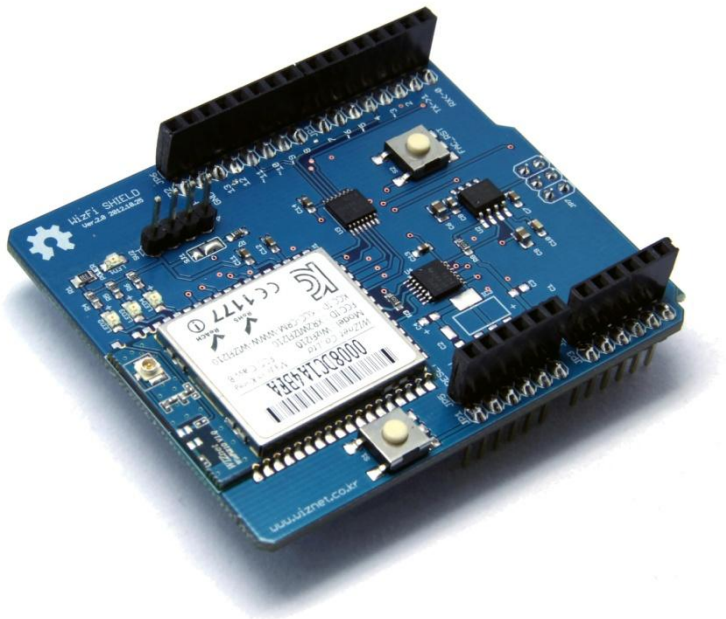
WIZnet, Inc.

WizFi Shield Users' Manual Rev1.0

For Arduino users



목 차

그림 목차

1. Overview

WizFi Shield는 위즈네트의 무선랜 모듈인 WizFi210을 사용하여 아두이노 환경에서 무선랜 통신을 쉽게 구현할 수 있도록 만든 Open Source Hardware 기반의 Wi-Fi 쉴드입니다. 본 제품에 사용한 WizFi210 모듈의 저전력 특성과 안정적인 무선랜 기능을 바탕으로 아두이노 보드에서의 제어에 필요한 기능들을 라이브러리화 하여 제공하기 때문에 아두이노 개발 환경에서 누구나 쉽게 사용할 수 있습니다. 뿐만 아니라 Open Source Hardware 정책을 준수하기 때문에 모든 설계 자료를 누구나 공유하고 개선할 수 있습니다.

WizFi Shield의 주요 기능은 다음과 같습니다.

- 아두이노 보드와 연결 가능한 하드웨어 설계 및 핀 배치
- 아두이노 보드와 연결을 위한 SPI 인터페이스 제공
- 다중 TCP/IP 통신 소켓 지원(최대 16 소켓)
- 아두이노 IDE용 라이브러리 제공(아두이노 Ethernet Shield 라이브러리와 유사함.)

2. 하드웨어 구성

2.1. 개요

WizFi Shield는 무선랜 처리를 위해서 사용하는 WizFi210을 SPI 버스를 이용해서 아두이노 보드와 연결합니다. 통신을 위한 SPI 버스 외에도 WizFi210을 제어하기 위한 제어신호를 아두이노 보드의 GPIO 핀을 이용해서 연결합니다.

2.2. 아두이노 보드와의 연결

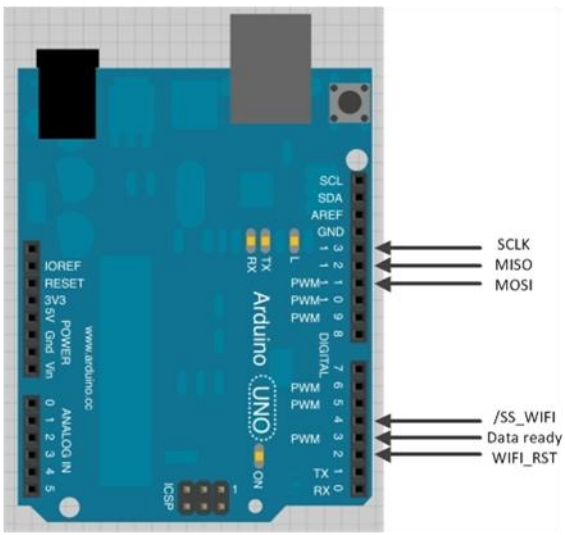


그림 1 아두이노 보드와의 연결 핀 맵

2.2.1. 전원 및 리셋용 GPIO 핀

WizFi Shield는 아두이노 보드로 부터 5V 입력을 받으며, 쉴드 내부 레귤레이터를 통하여 WizFi210 모듈에 3.3V 전원을 공급합니다. 그리고 이 레귤레이터의 Enable 신호를 Digital Pin #2로 제어하여 WizFi210 모듈을 리셋합니다.. 이 리셋 신호에는 풀업(Pull up)이 걸려 있으며, 이 신호를 Low로 하면 모듈이 리셋됩니다.

```
#define WizFi2x0_RST      2
pinMode(WizFi2x0_RST, OUTPUT);
```

2.2.2. SPI 버스용 GPIO 핀

WizFi Shield는 아두이노 보드와 SPI 버스를 통해서 통신합니다.

SPI slave enable(chip select) 신호는 Digital Pin #4를 사용하고 WizFi210 모듈로부터의 출력 신호인 Data ready 신호는 Digital Pin #3을 사용합니다. 이들 신호들은 회로도에서 각각 /SS_Wifi와 GPIO19로 표시되어 있습니다.

Data ready신호는 WizFi210 모듈이 아두이노 보드에 전달할 데이터가 있는지 여부를 알려주는 신호이며, 데이터가 있을 경우에 enable(HIGH) 됩니다. 이 모든 신호들은 3.3V 레벨이므로 5V I/O 인터페이스를 위해 Voltage level transceiver가 사용되었습니다.

```
#define WizFi2x0_DataReady      3
#define WizFi2x0_CS             4
 pinMode(WizFi2x0_DataReady, INPUT);
 pinMode(WizFi2x0_CS, OUTPUT);
```

2.2.3. 모드 설정 및 UART

WizFi210 모듈의 디폴트 동작 모드는 실행모드(run mode)이며, 만약 모듈의 펌웨어를 업데이트해야 할 경우에는 프로그램 모드(program mode)로 변경을 해야 합니다. 프로그램 모드로 변경을 하기 위해서는 SJ1의 1, 2번을 short 시키고, 2, 3번의 패턴을 끊어야 합니다. 프로그램 모드에서 실행 모드로 되돌리기 위해서는 SJ1의 1, 2번 연결을 끊고 2, 3번을 연결해야 합니다.

WizFi210의 UART핀이 WizFi Shield 보드에 있는 4핀 핀헤더 SV2에 연결되어 있습니다. 이 UART핀은 WizFi210 모듈의 펌웨어를 업데이트 할 필요가 있을 때 사용하기 위한 단자로, 동작모드를 프로그램 모드로 변경하고 이 단자를 이용해서 새 펌웨어를 다운로드 합니다.

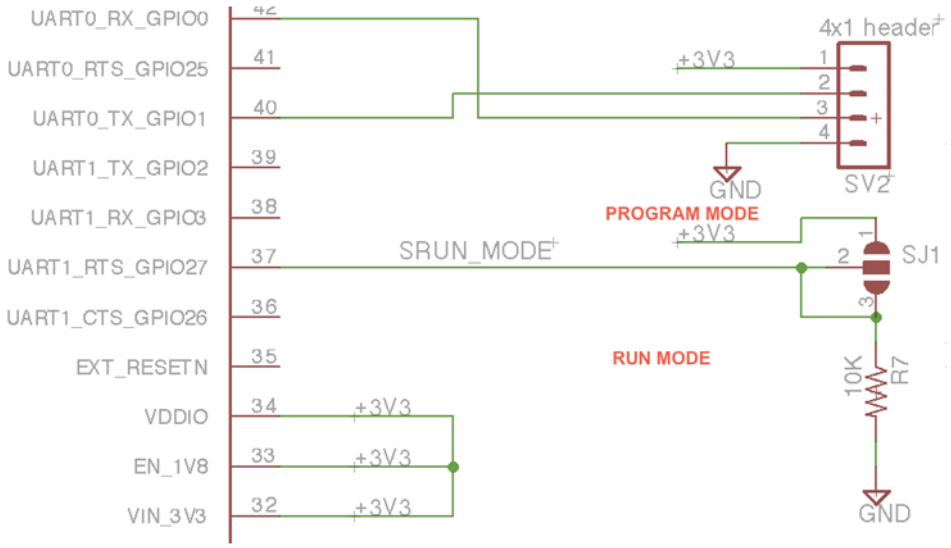


그림 2 모드 설정 및 UART 연결 관련 WizFi Shield 회로 구성

3. 라이브러리

3.1. 라이브러리 설치

WizFi Shield를 사용해서 무선랜 통신 기능을 구현하기 위한 개발 환경 설정 방법입니다. 개발 환경의 설치부터 라이브러리 설치까지 아래의 단계에 따라서 개발 환경을 설정할 수 있습니다.

3.1.1. 아두이노 IDE 설치

아두이노 IDE를 설치하지 않은 분은 http://arduino.cc/en/Main/Software 에서 프로그램을 다운로드 해서 설치합니다. (이미 아두이노 IDE를 설치하신 분은 skip하셔도 됩니다.)

3.1.2. WizFi 관련 라이브러리 설치

WizFi2x0.zip 과 TimeOne.zip 파일을 Arduino IDE의 libraries 디렉터리에 복사합니다.

두 zip 파일의 압축을 해제하면 아래 그림과 같이 WizFi2x0 디렉터리와 TimerOne 디렉터리가 생성됩니다.

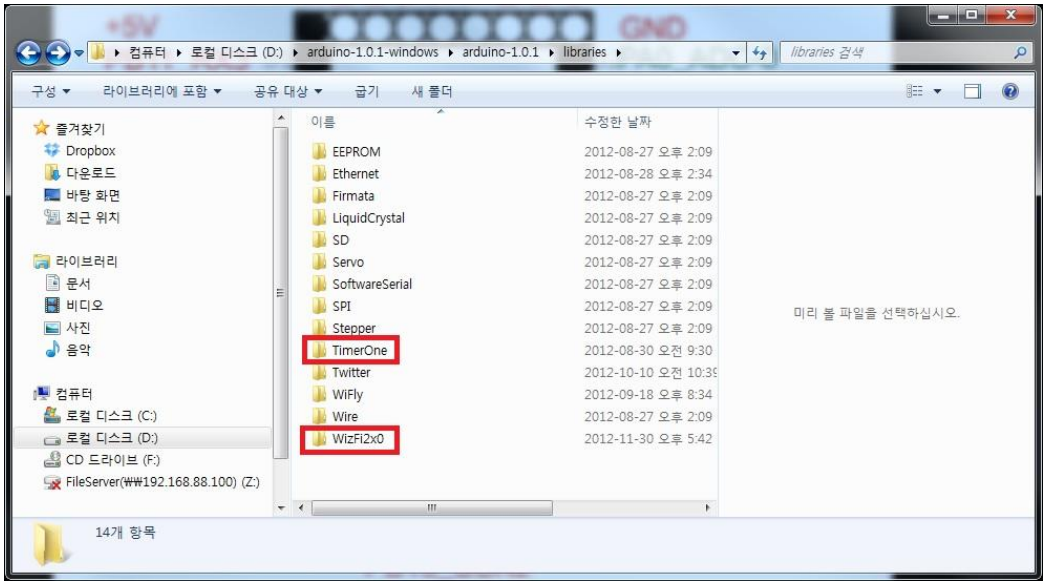


그림 3 WizFi2x0 관련 라이브러리 디렉토리 화면

3.2. 라이브러리 구성

WizFi 쉴드의 라이브러리는 WizFi2x0 디렉터리와 TimerOne 디렉터리로 구성됩니다.

WizFi2x0 라이브러리는 WizFi 쉴드에 탑재된 위즈네트의 WizFi210 모듈을 제어하는 함수와 WizFi210 모듈로부터의 응답을 처리하는 함수들을 포함하고 있습니다.

TimerOne 디렉터리는 Arduino 보드의 MCU에서 제공하는 Timer1 인터럽트를 사용하기 위한 함수입니다. WizFi2x0 라이브러리 내의 함수들은 이 Timer1 인터럽트를 이용해서 일정시간이 경과했는지를 확인합니다. 따라서 TimerOne 디렉터리도 반드시 설치해야 합니다.

3.3. WizFi2x0 라이브러리의 구성

3.3.1. WizFi2x0 라이브러리의 파일 구성

파일명	설명
Keywords.txt	Arduino 라이브러리 규격과 동일 Sketch에서 접근할 수 있는 함수 명, 상수 등 Keyword를 명기
WizFi2x0.cpp	WizFi2x0 모듈을 직접 제어하기 위한 함수들의 구현
WizFi2x0.h	WizFi2x0 모듈을 직접 제어하기 위한 함수들의 정의
WizFiClient.cpp	WizFi2x0 모듈을 통해 TCP Client 소켓을 사용하기 위해 필요한 함수들의 구현
WizFiClient.h	WizFi2x0 모듈을 통해 TCP Client 소켓을 사용하기 위해 필요한 함수들의 정의
WizFiServer.cpp	WizFi2x0 모듈을 통해 TCP Server 소켓을 사용하기 위해 필요한 함수들의 구현
WizFiServer.h	WizFi2x0 모듈을 통해 TCP Server 소켓을 사용하기 위해 필요한 함수들의 정의
WizFiUDP.cpp	WizFi2x0 모듈을 통해 UDP 소켓을 사용하기 위해 필요한 함수들의 구현
WizFiUDP.h	WizFi2x0 모듈을 통해 UDP 소켓을 사용하기 위해 필요한 함수들의 정의

그림 4 WizFi2x0 라이브러리의 구성 파일 설명

3.3.2. WizFi2x0 라이브러리의 Class 구성

Class	주요 기능	비고
WizFi2x0Class (in WizFi2x0.cpp, WizFi2x0.h)	- WizFi2x0 초기화 - AP 접속 및 해제 - WizFi2x0 모듈에 대한 각종 제어	
TimeoutClass (in WizFi2x0.cpp, WizFi2x0.h)	- WizFi2x0 모듈의 응답 대기 시간 지정 - 특정 시간 단위의 Event 처리 등의 목적	
WizFiClient (in WizFiClient.cpp, WizFiClient.h)	- TCP 클라이언트 소켓 생성 - 서버에 대한 접속 및 해제 - 데이터 송수신	- 함수 수행과정에서 WizFi210 모듈을 제어하기 위해서 WizFi2x0Class 이용
WizFiServer (in WizFiServer.cpp, WizFiServer.h)	- TCP 서버(Listen) 소켓 생성 **주의)** **WizFiServer 객체는 데이터 송수신 함수가 없음. 상대방과 연결이 되고 나면 데이터 송수신은 WizFiClient 객체를 통해서 할 수 있음. 따라서 WizFiServer 객체를 사용하는 경우에는 필요한 소켓의 개수에 따라서 WizFiClient 객체를 선언해서 사용해야 함.**	- 함수 수행과정에서 WizFi210 모듈을 제어하기 위해서 WizFi2x0Class 이용
WizFiUDP (in WizFiUDP.cpp, WizFiUDP.h)	- UDP 소켓 생성 - 데이터 송수신 - 상대방 IP 주소 및 포트번호 지정	- 함수 수행과정에서 WizFi210 모듈을 제어하기 위해서 WizFi2x0Class 이용

그림 5 WizFi2x0 라이브러리의 Class 구성

3.3.3. WizFi2x0 라이브러리의 Class 별 제공 Functions

3.3.3.1. WizFi2x0Class

Function Name	**WizFi2x0Class(void)**
Return value	**None**
Parameter	**None**
Description	- WizFi2x0Class 객체의 생성자

Function Name	**void begin(void)**
Return value	**None**
Parameter	**None**
Description	- WizFi2x0 모듈과 연결된 핀들을 설정 - SPI 속도 지정 - WizFi2x0 모듈의 초기화 수행(HW Reset) - HW Reset핀을 High로 올리고 4초간 wait 함.(모듈의 초기화를 기다림) - ReplyCheckTimer 초기화 - 모든 State Diagram의 Current State 초기화

Function Name	**uint8_t associate(void)**
Return value	**1 : success, 0 : fail**
Parameter	**None**
Description	- AP에 대한 접속 시도 - WizFi2x0Class 객체에 AP의 SSID 등 필요한 정보를 설정하여 이미 AP 접속이 있었던 경우에 다시 접속을 시도할 때 이 함수를 호출합니다.

Function Name	**uint8_t associate(ssid, passphrase, EncryptType, isDHCP)**
Return value	**1 : success, 0 : fail**
Parameter	**ssid : AP의 SSID 값, string type** **passphrase: Security Key 값(HEX format)** **EncryptType: Encryption type에 따라 Enumeration 값 지정** **isDHCP: boolean type, DHCP 모드 사용 여부를 지정**
Description	- AP에 대한 접속 시도.

	- 최초로 AP 접속을 시도하거나 파라미터의 값이 변경된 경우에 AP 접속을 하려면 이 함수를 호출합니다.. - EncryptType TYPE / 의미 NO_SECURITY / 보안 없음 WEP_SECURITY / WEP 보안 WPA_SECURITY / WPA 보안 WPA2PSK_SECURITY / WPA2-PSK 보안 **주의)** **Parameter 중 EncryptType 필드에 위 4개중 하나를 지정해야 합니다.**

Function Name	**uint8_t disassociate(void)**
Return value	**1 : success, 0: fail**
Parameter	**None**
Description	- AP 접속을 해제합니다.. **주의)** **많은 무선장치가 존재할 때, AP 주파수를 점유하게 되면 다른 장치의 통신에 방해가 될 수 있기 때문에 데이터 송수신이 끝나면 AP 접속을 해제하는 것이 좋습니다.**

Function Name	**boolean IsAssociated(void)**
Return value	**true: associated, false: not associated**
Parameter	**None**
Description	- 현재 AP에 접속한 상태인지를 확인할 수 있는 함수입니다. - associated 상태이면 true, disassociated 상태이면 false를 반환합니다.

Function Name	**void SetOperatingMode[1](mode)**

[1] Limited AP 기능 사용을 위해 추가된 함수.
WizFi210 모듈은 자신이 AP 처럼 동작할 수 있음. 단, 프로그램 메모리 크기 등의 이유로 일반적인 AP의 기능 중 꼭 필요한 것만 제공하는데 이것을 WizFi210 모듈의 Limited AP 모드라고 함. 별도의 AP없이 스마트폰과 직접 Wi-Fi 통신을 해야하는 경우에 유용함.

Return value	**None**
Parameter	**Mode: OPMODE Type의 Enumeration 값 지정**
Description	- WizFi2x0 모듈의 동작 모드를 지정하는 함수입니다. - 동작 모드는 다음과 같은 종류가 있습니다. **주의)** **WizFi2x0 모듈의 default 동작 모드는 INFRA_MODE입니다. 만약 동작 모드를 변경하고자 한다면 associate(…) 함수를 호출하기 전에 이 함수를 호출해서 동작 모드를 변경해야 합니다.**

동작 모드	의미
INFRA_MODE	Infrastructure mode
ADHOC_MODE	Adhoc mode
LIMITEDAP_MODE	AP mode

Function Name	**void SendSync(void)**
Return value	**None**
Parameter	**None**
Description	- Arduino 보드와 WizFi Shield간에 SPI 통신이 정상적인지를 확인하기 위해 Sync 패킷을 전송합니다. - Sync 패킷의 값은 0xF5 입니다.

Function Name	**uint8_t CheckSyncReply(void)**
Return value	**None**
Parameter	**None**
Description	- Sync 패킷에 대해 WizFi Shield로부터 정상적인 Reply를 보냈는지를 확인합니다. - 정상적인 Reply는 0xF5, 0xFF 그리고 0x00 중에 하나입니다.

Function Name	**uint8_t SetTxPower[2](uint8_t power_level)**
Return value	**1: success, 0: fail**
Parameter	**Power_level: 변경할 송신 출력 레벨.**

[2] WizFi2x0 모듈의 RF 신호 송신 출력을 변경하기 위해 추가되었습니다.

Description	- WizFi2x0 모듈의 RF 송신 출력을 변경하기 위한 함수입니다. - WizFi210의 경우는 0 ~ 7, WizFi220의 경우는 2 ~ 15 사이의 값을 지정할 수 있습니다. 그 외의 경우에는 [ERROR]가 발생합니다.

Function Name	**uint8_t GetTxPower[3](void)**
Return value	**Power_level: 현재 설정된 송신 출력 레벨**
Parameter	**None**
Description	- WizFi2x0 모듈의 현재 설정된 RF 송신 출력 레벨을 확인하기 위한 함수입니다.

Function Name	**uint8_t wifiScan (void)**
Return value	**1: success, 0: fail**
Parameter	**None**
Description	- WizFi2x0 모듈 주변의 모든 AP를 스캔하는 함수입니다. - 현재의 라이브러리에서는 AP들의 SSID 값만 화면에 출력합니다. 그러나 사용자가 라이브러리의 출력문을 수정해서 다른 정보도 출력하도록 수정할 수 있습니다.

Function Name	**uint8_t wifiScan (uint8_t channel)**
Return value	**1: success, 0: fail**
Parameter	**channel: 특정 주파수 채널을 선별하기 위한 파라미터**
Description	- WizFi2x0 모듈 주변의 모든 AP를 스캔하여 지정된 채널로 열려있는 AP의 SSID만 출력하는 함수입니다. 채널에 '0'을 지정하면 채널에 따른 검색을 하지 않습니다. - 현재의 라이브러리에서는 AP들의 SSID 값만 화면에 출력합니다. 그러나 사용자가 라이브러리의 출력문을 수정해서 다른 정보도 출력하도록 수정할 수 있습니다.

Function Name	**uint8_t wifiScan (uint8_t channel, uint8_t RSSI)**
Return value	**1: success, 0: fail**

[3] 위와 같습니다.

Parameter	**channel: 특정 주파수 채널을 선별하기 위한 파라미터** **RSSI: 신호의 세기를 지정하기 위한 파라미터(양수로 표현)**
Description	- WizFi2x0 모듈 주변의 모든 AP를 스캔하여 지정된 채널로 열려있으며 신호의 세기가 지정된 것보다 강한 AP의 SSID만 출력하는 함수입니다. 채널에 '0'을 지정하면 채널에 따른 검색을 하지 않습니다. - 현재의 라이브러리에서는 AP들의 SSID 값만 화면에 출력합니다. 그러나 사용자가 라이브러리의 출력문을 수정해서 다른 정보도 출력하도록 수정할 수 있습니다.

Function Name	**void SetSrcIPAddr(byte * buf)**
Return value	**None**
Parameter	**buf: Source IP address를 string 형태로 가지고 있는 배열**
Description	- WizFi2x0 모듈의 IP 주소 등을 고정적으로 지정할 때 사용하는 함수 중 하나로써 인자로 받은 buf의 값을 Source IP Address 로 지정하는 함수입니다.

Function Name	**void GetSrcIPAddr(byte * buf)**
Return value	**None**
Parameter	**buf: Source IP address를 string 형태로 반환 받을 배열**
Description	- WizFi2x0 모듈의 현재 설정된 Source IP 주소를 buf 변수에 담아서 반환합니다.

Function Name	**void SetSrcSubnet(byte * buf)**
Return value	**None**
Parameter	**buf: Source Subnet Mask를 string 형태로 가지고 있는 배열**
Description	- WizFi2x0 모듈의 IP 주소 등을 고정적으로 지정할 때 사용하는 함수 중 하나로써 인자로 받은 buf의 값을 Source Subnet Mask 로 지정하는 함수입니다.

Function Name	**void GetSrcSubnet(byte * buf)**
Return value	**None**
Parameter	**buf: Source Subnet Mask를 string 형태로 반환 받을 배열**

Description	- WizFi2x0 모듈의 현재 설정된 Source Subnet Mask를 buf 변수에 담아서 반환합니다.

Function Name	**void SetSrcGateway(byte * buf)**
Return value	**None**
Parameter	**buf: Source Gateway address를 string 형태로 가지고 있는 배열**
Description	- WizFi2x0 모듈의 IP 주소 등을 고정적으로 지정할 때 사용하는 함수 중 하나로써 인자로 받은 buf의 값을 Source Gateway Address로 지정하는 함수입니다.

Function Name	**void GetSrcGateway(byte * buf)**
Return value	**None**
Parameter	**buf: Source Gateway address를 string 형태로 반환 받을 배열**
Description	- WizFi2x0 모듈의 현재 설정된 Source Gateway Address를 buf 변수에 담아서 반환합니다.

Function Name	**void SetSrcPortnum(uint16_t portnum)**
Return value	**None**
Parameter	**portnum: Server 소켓에서 사용할 port number**
Description	- WizFi2x0 모듈의 Server 소켓에서 참조할 WizFi2xoClass 객체 내의 SrcPortNum 멤버변수에 인자 portnum의 값을 지정하는 함수입니다. - WizFiServer 객체가 멤버함수 begin()내에서 이 함수를 참조합니다.

Function Name	**uint16_t GetSrcPortnum(void)**
Return value	**None**
Parameter	**None**
Description	- WizFi2x0 모듈의 SrcPortNum 변수에 저장된 현재 값을 반환합니다.

Function Name	**void RcvPacket(void)**
Return value	**None**

Parameter	**None**
Description	- WizFi2x0 모듈의 SPI 포트를 통해서 나오는 데이터(수신 데이터, 모듈에서 발생한 이벤트에 대한 통지 메시지 등)를 loss없이 읽어들이기 위한 함수입니다. **주의)** **WizFi2x0 모듈은 SPI 포트를 통해서 사용자의 통신 데이터나 통지 메시지를 지속적으로 전송하기 때문에 데이터나 메시지의 loss가 없도록 하기 위해서는 이 함수를 지속적으로 호출하도록 조치하여야 합니다.**

3.3.3.2. WizFiClient

Function Name	**WizFiClient(void)**
Return value	**None**
Parameter	**None**
Description	- WizFiClient의 기본 생성자

Function Name	**WizFiClient(uint8_t ip, uint16_t port)**
Return value	**None**
Parameter	- **ip : 연결할 서버의 IP address. ex) ip[4] = {192,168,0,100}** - **port : 연결할 서버의 Listen port number**
Description	- Server의 IP 주소를 알고 있을 때의 WizFiClient 생성자입니다.

Function Name	**WizFiClient(const char* domain, uint16_t port)**
Return value	**None**
Parameter	- **domain : 연결할 서버의 domain name. ex)www.google.com** - **port : 연결할 서버의 Listen port number**
Description	- Server의 domain 주소를 알고 있을 때의 WizFiClient 생성자입니다.

Function Name	**uint8_t connect()**
Return value	**1: success, 0: fail**
Parameter	**None**
Description	- Server와의 접속을 시도합니다. - 접속할 서버의 IP 주소(또는 도메인)와 서버의 대기 포트번호

	는 객체를 생성할 때 부여한 것을 (함수 내부적으로) 사용합니다. - 접속이 성공하면 1 을, 실패하면 0 을 반환합니다.

Function Name	**uint8_t disconnect()**
Return value	**1: success, 0: fail**
Parameter	**None**
Description	- Server와의 접속을 해제합니다. - 접속 해제가 성공하면 1 을, 실패하면 0 을 반환합니다.

Function Name	**boolean available()**
Return value	**true: connected, false: disconnected**
Parameter	**None**
Description	- 현재 소켓이 상대방과 연결된 상태인지 여부를 알기 위한 함수로써 연결 상태이면 true 를, 연결 해제 상태이면 false 를 반환합니다. - WizFi2x0 모듈이 모듈 내부의 이벤트(특정 소켓의 연결, 연결 해제, 에러 발생 등) 발생을 알리기 위해 전송한 통지 메시지가 자신에게 해당하는 지를 확인해서 소켓의 상태를 그에 맞도록 변경하고, 소켓의 연결상태를 반환합니다.

Function Name	**void write(byte value)**
Return value	**None**
Parameter	**value: 소켓을 통해서 전송할 한 바이트 데이터**
Description	- 한 바이트의 데이터(value)를 서버로 전송합니다. - 이 함수를 사용하기 위해서 소켓은 연결상태여야 합니다.

Function Name	**void write(byte *buf)**
Return value	**None**
Parameter	**buf: 소켓을 통해서 전송할 바이트 스트림 데이터**
Description	- 두 바이트 이상의 데이터를 서버로 전송할 때 사용하는 함수입니다. 유효한 데이터의 맨 마지막 바이트 다음 바이트는 널 문자(NULL character)로 채워야 합니다.

	- **만약, 널 문자가 포함된 데이터를 전송하고자 할 때, 이 함수를 사용하면 데이터가 소실될 수 있습니다. 그런 경우에는 다음의 write(byte *buf, size_t size) 함수를 사용해야합니다.**
	- 예) "abcde"를 전송하고자 할 때,
	Byte buf[10];
	memset(buf, 0, 6);
	memcpy(buf, "abcde", 5);
	myClient.write(buf); //myClient는 WizFiClient type의 객체

Function Name	**void write(byte *buf, size_t size)**
Return value	**None**
Parameter	**buf: 소켓을 통해서 전송할 바이트 스트림 데이터** **size: 바이트 스트림 중, 실제로 전송할 데이터 사이즈**
Description	- 두 바이트 이상의 데이터를 서버로 전송할 때 사용하는 함수로써 전송할 데이터의 길이를 알고 있거나 일부 데이터만 전송하는 경우에 이 함수를 사용합니다. - **널 문자가 포함되었거나 포함될 가능성이 있는 데이터를 전송하는 경우에는 반드시 이 함수를 사용해야 합니다.**

Function Name	**uint8_t read()**
Return value	**수신한 데이터**
Parameter	**None**
Description	- 서버로부터 수신한 데이터를 읽어옵니다. - 데이터는 반환값으로 전달됩니다.

Function Name	**uint8_t read(byte *buf)**
Return value	**수신한 데이터의 바이트 수**
Parameter	**buf: 서버로부터 수신한 데이터를 buf에 채워서 반환.**
Description	- 서버로부터 수신한 데이터를 배열로 읽어옵니다.

Function Name	**uint8_t read(byte *buf, size_t size)**
Return value	**수신한 데이터의 바이트 수**
Parameter	**buf: 서버로부터 수신한 데이터를 buf에 채워서 반환.**

	size: 수신할 데이터의 바이트 수를 지정.
Description	- 서버로부터 수신한 데이터를 배열로 읽어옵니다. - 인자로 전달한 buf에 size보다 적거나 같은 바이트 수만큼 수신 데이터를 채워서 반환 받습니다.

Function Name	**void GetCIDstr(byte *strCID)**
Return value	**None**
Parameter	**strCID: CID 값을 문자열로 변환한 배열의 포인터.**
Description	- 본 함수는 CID값을 Str으로 반환 받기 위한 함수입니다. - 접속 성공한 경우, WizFiClient는 별도의 CID 값을 할당 받습니다.(0 ~ 15)

Function Name	**uint8_t GetCID(void)**
Return value	**CID값을 정수값으로 반환.**
Parameter	**None**
Description	- 본 함수는 CID값을 integer값으로 반환 받기 위한 함수입니다.

3.3.3.3. WizFiServer

Function Name	**WizFiServer(Serverport)**
Return value	**None**
Parameter	**None**
Description	- WizFiServer 객체의 생성자

Function Name	**uint8_t begin(void)**
Return value	**1: success, 0: fail**
Parameter	**None**
Description	- 본 함수는 Server 소켓이 생성자에서 지정한 포트 값으로 Listen 하도록 합니다.

Function Name	**void GetCIDstr(byte *strCID)**
Return value	**None**
Parameter	**strCID: CID 값을 문자열로 변환한 배열의 포인터.**
Description	- 본 함수는 CID값을 문자열로 반환 받기 위한 함수입니다.

Function Name	**uint8_t GetCID(void)**
Return value	**None**
Parameter	**None**
Description	- 본 함수는 CID값을 정수값으로 반환 받기 위한 함수입니다.

3.3.3.4. WizFiUDP

Function Name	**WizFiUDP(void)**
Return value	**None**
Parameter	**None**
Description	- WizFiUDP의 기본 생성자

Function Name	**WizFiUDP (uint8_t ip, uint16_t port, uint16_6 src_port)**
Return value	**None**
Parameter	- **ip : 데이터를 보낼 상대방의 IP 주소** - **port : 데이터를 보낼 상대방의 포트 번호** - **src_port: UDP 소켓 자신의 포트 번호**
Description	- 상대방의 IP 주소를 알고 있을 때 사용하는 WizFiUDP 생성자입니다. - 상대방의 IP 주소나 포트넘버는 데이터를 보낼 때 마다 변경해서 보낼 수 있기 때문에, ip와 port 파라미터는 지정하지 않아도 괜찮지만, src_port 는 반드시 지정해두어야 합니다.

Function Name	**WizFiUDP (const char* domain, uint16_t port, uint16_t src_port)**
Return value	**None**
Parameter	- **domain : 데이터를 보낼 상대방의 도메인 네임** - **port : 데이터를 보낼 상대방의 포트 넘버** - **src_port: UDP 소켓 자신의 포트 번호**
Description	- 상대방의 domain 주소를 알고 있을 때 사용하는 WizFiUDP 생성자입니다. - 상대방의 IP 주소나 포트넘버는 데이터를 보낼 때 마다 변경해서 보낼 수 있기 때문에, domain과 port 파라미터는 지정하지 않아도 괜찮지만, src_port 는 반드시 지정해두어야 합니다.

Function Name	**uint8_t open(void)**

Return value	**1: success, 0: fail**
Parameter	**None**
Description	- WizFiUDP 생성자를 통해 지정한 정보를 토대로 WizFi2x0 모듈이 실제로 UDP 소켓을 생성하도록 하는 함수입니다. - UDP 소켓이 정상적으로 생성되면 '1'을 리턴하고 실패하였으면 '0'을 리턴합니다.

Function Name	**uint8_t close(void)**
Return value	**1: success, 0: fail**
Parameter	**None**
Description	- 생성된 UDP 소켓을 닫기 위해서 사용하는 함수입니다. - UDP 소켓이 정상적으로 종료되면 '1'을 리턴하고 이미 소켓이 종료된 경우 등 여러 이유로 종료 명령이 실패한 경우에는 '0'을 리턴합니다.

Function Name	**void write(byte value)**
Return value	**None**
Parameter	**value: 소켓을 통해서 전송할 한 바이트 데이터**
Description	- 한 바이트의 데이터(value)를 상대에게 전송합니다. - 이 함수를 사용하기 위해서 소켓은 **생성**상태여야 합니다.

Function Name	**void write(byte *buf)**
Return value	**None**
Parameter	**buf: 소켓을 통해서 전송할 바이트 스트림 데이터**

Description	- 두 바이트 이상의 데이터를 상대방에게 전송할 때 사용하는 함수로써 유효한 데이터의 맨 마지막 바이트 다음 바이트는 널 문자(NULL character)로 채워야 합니다.
	- **만약, 널 문자가 포함된 데이터를 전송하고자 할 때, 이 함수를 사용하면 데이터가 소실될 수 있습니다. 그런 경우에는 다음의 write(byte *buf, size_t size) 함수를 사용해야합니다.**
	- 예) "abcde"를 전송하고자 할 때,
	Byte buf[10];
	memset(buf, 0, 6);
	memcpy(buf, "abcde", 5);

	myUDP.write(buf); //myUDP는 WizFiUDP 타입의 객체

Function Name	**void write(byte *buf, size_t size)**
Return value	**None**
Parameter	**buf: 소켓을 통해서 전송할 바이트 스트림 데이터** **size: 바이트 스트림 중, 실제로 전송할 데이터 사이즈**
Description	- 두 바이트 이상의 데이터를 상대방에게 전송할 때 사용하는 함수로써 전송할 데이터의 길이를 알고 있거나 일부 데이터만 전송하는 경우에 이 함수를 사용합니다. - **널 문자가 포함되었거나 포함될 가능성이 있는 데이터를 전송하는 경우에는 반드시 이 함수를 사용해야 합니다.**

Function Name	**uint8_t read()**
Return value	**수신한 데이터**
Parameter	**None**
Description	- 상대방으로부터 수신한 데이터를 읽어옵니다. - 데이터는 반환값으로 전달됩니다.

Function Name	**uint8_t read(byte *buf)**
Return value	**수신한 데이터의 바이트 수**
Parameter	**buf: 상대방으로부터 수신한 데이터를 buf에 채워서 반환.**
Description	- 상대방으로부터 수신한 데이터를 배열로 읽어옵니다.

Function Name	**uint8_t read(byte *buf, size_t size)**
Return value	**수신한 데이터의 바이트 수**
Parameter	**buf: 상대방으로부터 수신한 데이터를 buf에 채워서 반환.** **size: 수신할 데이터의 바이트 수를 지정.**
Description	- 상대방으로부터 수신한 데이터를 배열로 읽어옵니다. - 인자로 전달한 buf에 size보다 적거나 같은 바이트 수만큼 수신 데이터를 채워서 반환 받습니다.

Function Name	**void GetCIDstr(byte *strCID)**
Return value	**None**

Parameter	**strCID: CID 값을 문자열로 변환한 배열의 포인터.**
Description	- 본 함수는 CID값을 Str으로 반환 받기 위한 함수입니다. - 소켓 생성에 성공한 경우, WizFiUDP는 별도의 CID 값을 할당 받습니다.(0 ~ 15)

Function Name	**uint8_t GetCID(void)**
Return value	**CID값을 정수값으로 반환.**
Parameter	**None**
Description	- 본 함수는 CID값을 integer값으로 반환 받기 위한 함수입니다.

Function Name	**void GetCurrentDestInfo(byte *ipaddr, uint16_t *portnum)**
Return value	**None**
Parameter	**ipaddr: WizFiUDP 객체가 가지고 있는 상대 IP를 받는 문자열** **portnum: WizFiUDP 객체가 가지고 있는 상대 포트번호를 저장할 변수의 주소**
Description	- 본 함수는 WizFiUDP 객체가 현재 가지고 있는 상대방의 IP 주소와 포트 번호를 읽어오기 위한 함수입니다. - 상대방의 IP 주소와 포트 번호는 UDP 패킷을 수신할 때마다 수신된 패킷으로부터 상대방의 IP 주소와 포트 번호를 추출하여 자동 업데이트 됩니다. 또, SetCurrentDestInfo(…) 함수를 통해 사용자가 변경할 수도 있습니다.

Function Name	**void SetCurrentDestInfo(byte *ipaddr, uint16_t portnum)**
Return value	**None**
Parameter	**ipaddr: WizFiUDP 객체의 상대 IP를 갱신할 IP 주소값의 문자열** **portnum: WizFiUDP 객체의 상대 포트번호를 갱신할 포트번호**
Description	- 본 함수는 WizFiUDP 객체내의 상대방의 IP 주소와 포트 번호를 변경하기 위한 함수입니다.

3.3.3.5. TimeoutClass

Function Name	**void init(void)**
Return value	**None**
Parameter	**None**
Description	- Timeout을 체크하기 위한 변수들을 초기화합니다.

Function Name	**void TimerStart(void)**
Return value	**None**
Parameter	**None**
Description	- Timeout 체크를 개시합니다. - TimeoutClass 내부 변수인 TimerValue를 현재 설정된 값 그대로 사용합니다.

Function Name	**void TimerStart(uint16_t timevalue)**
Return value	**None**
Parameter	**None**
Description	- Timeout 체크를 개시합니다. - 인자로 지정한 timevalue를 TimeoutClass 내부 변수인 TimerValue에 할당합니다. - 인자인 timevalue의 단위는 1ms입니다. 예를 들어, timevalue값이 5000이면 5000ms, 즉 5초가 경과하면 Timeout 조건에 도달하게 되고 IsTimeout이 true가 됩니다.

Function Name	**void TimerStop(void)**
Return value	**None**
Parameter	**None**
Description	- Timeout 체크를 중지합니다.

Function Name	**boolean GetIsTimeout(void)**
Return value	**None**
Parameter	**None**
Description	- TimeoutClass 의 내부 변수인 IsTimeout 플래그가 설정되었는지를 확인합니다. - Timeout이 되었으면 true를 아니면 false를 반환합니다.

Function Name	**void CheckIsTimeout()**
Return value	**None**
Parameter	**None**
Description	- 현재의 TimerCount가 TimerValue와 같아졌는지를 확인한 후,

	TimerCount를 1 증가시킵니다. - TimerCount와 TimerValue가 같아지면 IsTimeout 플래그를 true로 변경하고 TimerStop() 을 호출합니다. - **이 함수는 Timer1의 callback 함수에서 호출해주어야 합니다. (Timer1은 1ms마다 Timer interrupt가 발생하도록 설정되어 있습니다.)**

Function Name	**void SetIsTimeout(boolean flag);**
Return value	**None**
Parameter	**flag: true or false**
Description	- IsTimeout 값을 인자인 flag의 값으로 변경합니다.

4. 스케치 프로그래밍 가이드

WizFi Shield를 사용하는 스케치 프로그램은 선언부, setup() 함수, 그리고 loop() 함수로 구성된 아두이노 프로그래밍 구조를 기반으로 합니다. 따라서, 본 절에서는 WizFi Shield를 사용하기 위해서 선언부, setup() 함수, 그리고 loop() 함수에서 반드시 해야 할 작업에는 어떤 것이 있고 어떤 절차에 따라서 그 작업을 수행해야 하는 지를 설명합니다.

4.1. 선언부

선언부는 include 부, 변수 선언부 그리고 Timer1의 Callback 함수부로 구성됩니다. 각 부분에 대해서는 각 절에서 상세히 살펴보겠습니다.

4.1.1. include 부

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <SPI.h>
#include <WizFi2x0.h>
#include <WizFiClient.h>
#include <TimerOne.h>
```

그림 6 스케치 프로그램의 include 부 예제

include 부에서는 WizFi Shield를 사용하기 위해 필요한 라이브러리의 헤더 파일들을 포함합니다. 위에서 빨간색으로 표시한 헤더 파일은 반드시 포함해야 하는 헤더 파일들 입니다..

SPI.h 는 arduino 보드와 WizFi Shield간의 SPI 통신을 위해서 필요한 함수들을 포함합니다. WizFi2x0.h 는 WizFi2x0 모듈을 제어하기 위해 필요한 함수들을 포함합니다. WizFiClient.h 는 클라이언트 소켓을 사용하기 위해 필요한 함수들을 포함합니다. 만약, 서버 소켓을 사용한다면 WizFiServer.h 헤더 파일도 반드시 포함해 주어야 합니다.

4.1.2. 변수 선언

```
#define SSID            "AP_SSID"     // SSID of your AP
#define Key             "1234567890"  // Key or Passphrase
#define ServerName      "google.com" // Server Domain Name

unsigned char SIP[4] = {192, 168, 123, 119};
unsigned int ServerPort = 5000;

WizFi2x0Class myWizFi;
WizFiClient myClient;
WizFiServer myServer;
TimeoutClass ConnectInterval;
```

그림 7 스케치 프로그램의 변수 선언부 예제

SSID는 WizFi210 모듈이 접속할 AP의 SSID 값을 정의한 것입니다. 사용자가 접속할 AP의 SSID를 확인해서 이중 따옴표내의 값을 변경해야 합니다.

Key는 보안이 설정된 AP에 접속할 때, 접속을 위해 필요한 보안키입니다. **이중따옴표내의 값을 실제 AP의 구성에 맞게 변경해야 합니다.**

ServerName은 서버의 Domain name만으로 접속할 때 사용하기 위한 값입니다.

(ServerName이 NULL이 아니면 Domain name으로 접속을 시도합니다. 서버의 IP 주소를 정확하게 알고 있는 경우라면 ServerName 선언부를 삭제하시는 것이 좋습니다. DNS Query와 같은 불필요한 과정을 줄일 수 있습니다.)

SIP는 서버의 IP 주소를 알고 있을 때, 사용하는 변수로 서버의 IP 주소를 직접 적어줍니다. ServerPort는 서버에서 접속을 기다리고 있는 포트 넘버입니다. 해당하는 값으로 변경해야 합니다. WizFi2x0Class 타입의 myWizFi 객체는 WizFi210 모듈에 대한 제어를 위해 사용하게 될 객체입니다. **(myWizFi 객체는 WizFiClient 타입의 객체나 WizFiServer 타입의 객체에서 참조하기 때문에 반드시 myWizFi 라는 이름으로 선언해 주세요.)**

WizFiClient 타입의 myClient는 Client 소켓 처리를 위한 객체입니다. WizFiServer 타입의 myServer는 Server 소켓 처리를 위한 객체입니다. **WizFiClient 객체는 연결시도, 연결해제, 데이터 송수신 관련된 멤버 함수를 가지고 있지만, WizFiServer 객체는 연결대기(Listen) 멤버 함수 만을 가지고 있습니다. WizFi210 모듈이 서버 소켓으로써 다른 장치로부터의 연결을 기다리는 응용의 경우에 연결대기를 위해서는 WizFiServer 객체를 사용하고 연결된 이후에 데이터 송수신과 연결해제를 위해서는 별도로 생성한 WizFiClient 객체를 사용해야 합니다. 실제 사용 예는 "Example" 중 "WizFiBasicServerTest" 를 참조하세요.**

TimeoutClass 타입의 ConnectionInterval 객체는 스케치 프로그램에서 특정한 시간이 경과되었는 지를 확인하기 위해서 사용하는 객체입니다. 이 객체를 사용하지 않는다면 선언할 필요는 없습니다. **(하지만, TimeoutClass 객체를 전혀 사용하지 않는다 하더라도 TimerOne.h 는 반드시 include 부에서 포함하셔야 합니다. WizFi2x0Class 내에서 TimerOne.h를 사용해야 하기 때문입니다.)**

4.1.3. Timer1_ISR callback 함수

```
void Timer1_ISR()
{
    uint8_t i;
    myWizFi.ReplyCheckTimer.CheckIsTimeout();
}
```

그림 8 스케치 프로그램의 Timer1 Callback 함수 구성 예

Timer1_ISR callback 함수는 Timer1 인터럽트가 발생하면 호출되는 인터럽트 서비스 루틴입니다. 반드시 포함해야 합니다.

Timer1_ISR() 함수내에 있는 myWizFi.ReplyCheckTimer.CheckIsTimeout(); 구문이 myWizFi 객체에서 필요한 부분입니다. myWizFi 객체는 WizFi210 모듈에 명령을 내리고 일정 시간 내에 응답이 없으면 타임아웃 처리를 합니다. 응답이 없는 상태로 정해진 시간이 경과했는지를 확인하는 부분이 위 구문입니다.

4.2. setup() 함수 구성

setup() 함수 내에서는 한번만 수행하면 될 작업을 처리합니다. 다음과 같은 작업들은 setup() 함수 내에서 처리하는 것이 좋은 작업들입니다.

- 디버그 메시지 출력을 위한 시리얼 포트 초기화
- Client 소켓 객체 초기화
- Server 소켓 객체 초기화
- myWizFi 객체 초기화
- Timer1 초기화
- WizFi Shield와 SPI 통신 Sync 확인
- AP association 수행

(이 작업은 사용자의 필요에 따라서 loop() 함수에서 수행할 수도 있습니다.)

각 작업 수행을 어떻게 하는 지는 아래의 example 설명 부분을 참고하세요.

4.3. loop() 함수 구성

loop() 함수는 함수내의 코드가 무한 반복해서 호출됩니다. 따라서, 이 함수 내에서는 무한 반복해서 수행할 작업들을 기재해야 합니다. 다음과 같은 작업들을 이 함수 내에서 처리해야 합니다.

- myWizFi.RcvPacket() 호출
- Client 소켓을 서버에 연결시키는 작업(myClient.connect() 호출)
- 서버와의 접속 상태를 확인 하는 작업(myClient.available() 호출)
- 서버로부터 수신한 데이터가 있으면 읽어오는 작업(myClient.read() 호출)
- 서버에 송신할 데이터가 있으면 쓰는 작업(myClient.write() 호출)
- Client 소켓을 서버에서 연결 해제시키는 작업(myClient.disconnect() 호출)
- 그 외 기타 작업
  - 수신한 데이터를 시리얼 포트를 통해 디버그 창에 표시하기
  - 수신한 데이터를 분석해서 그에 맞는 후속 조치 하기

- 그 외 다수

각 작업 수행을 어떻게 하는 지는 아래의 example 설명 부분을 참고하세요.

5. Examples

5.1. WizFiBasicTest

5.1.1. WizFiBasicTest 의 개요

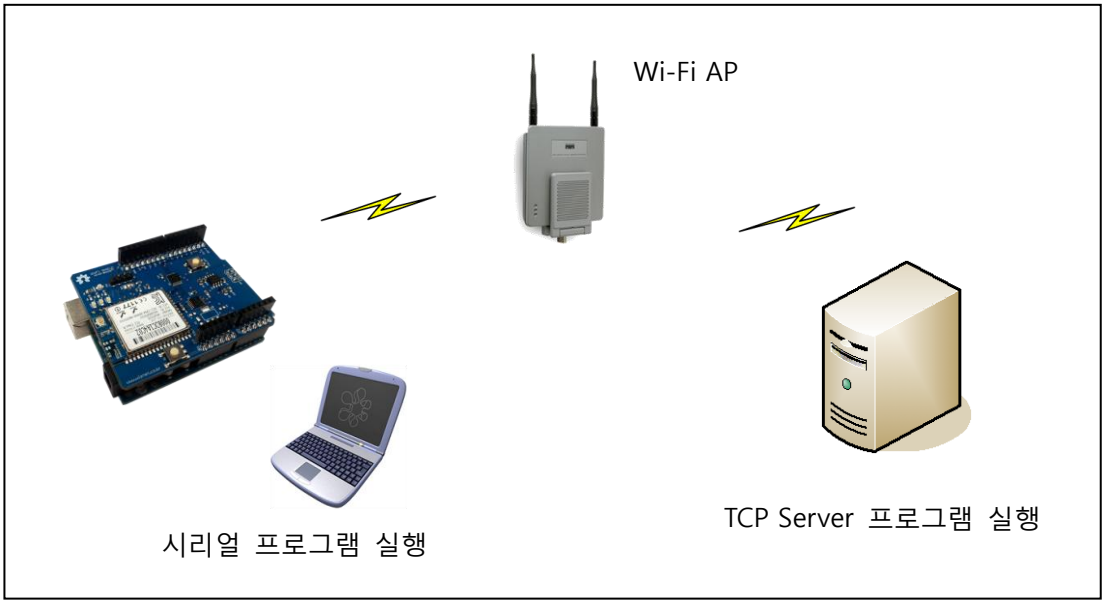


그림 9 WizFiBasicTest를 위한 네트워크 구성도

WizFiBasicTest 스케치는 다음과 같은 순서로 작업을 수행하는 예제입니다.

① WizFiBasicTest는 WizFiClient 소켓 하나를 만들어서 지정된 AP에 접속한 후, 사용자의 명령을 기다립니다.

② 시리얼 콘솔로 사용자의 '접속' 명령(영문 문자 'c' 또는 'C')이 들어오면, WizFiClient 소켓은 지정된 서버에 연결합니다.

③ 서버에 연결되었다면, 서버로부터 수신한 데이터가 있는 지를 확인합니다.

④ 수신한 데이터가 있으면 해당 데이터를 시리얼 콘솔 창에 display한 후, 그 데이터를 서버에 그대로 재전송합니다.

⑤ 시리얼 콘솔로 사용자의 '접속 해제' 명령(영문 문자 'd' 또는 'D')이 들어오면, WizFiClient 소켓의 연결을 종료합니다.

아래는 본 예제가 정상적으로 수행되어 사용자의 명령 입력을 기다리는 상태의 시리얼 창의 예 입니다.

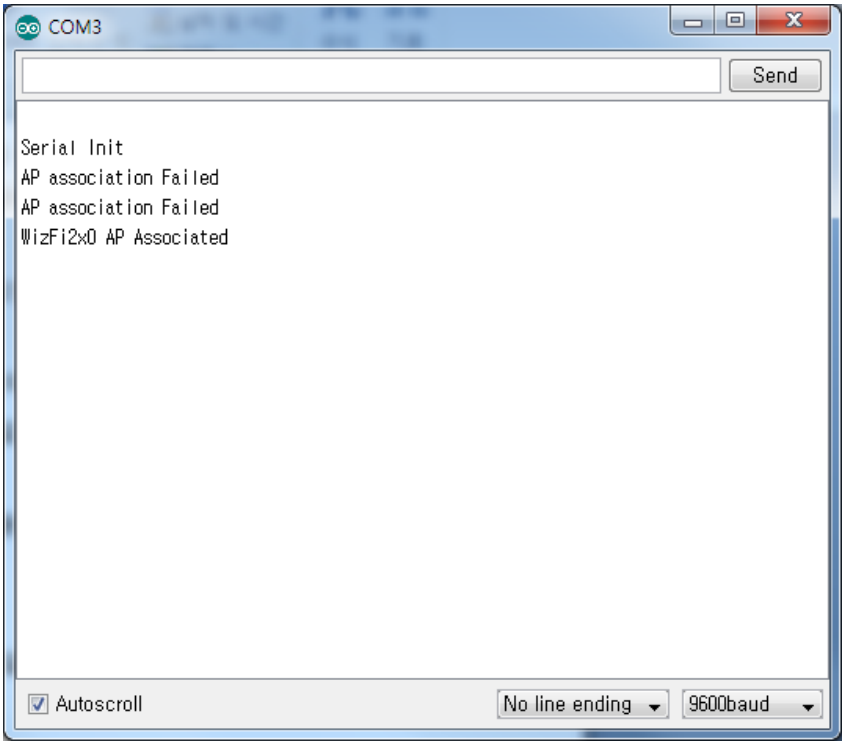


그림 10 WizFiBasicTest 예제의 정상 실행 화면

5.1.2. 선언부

5.1.2.1. include 부

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <SPI.h>
#include <WizFi2x0.h>
#include <WizFiClient.h>
#include <TimerOne.h>
```

그림 11 WizFiBasicTest 예제의 include 부 구성

5.1.2.2. 변수 선언

```
#define SSID     "AP_SSID"     // SSID of your AP
#define Key      "1234567890"  // Key or Passphrase
unsigned char SIP[4] = {192, 168, 123, 119};
unsigned int ServerPort = 5000;
WizFi2x0Class myWizFi;
WizFiClient myClient;
TimeoutClass ConnectInterval;
```

그림 12 WizFiBasicTest 예제의 변수 선언부 구성

5.1.2.3. Timer1_ISR callback 함수

```
void Timer1_ISR()
{
    uint8_t i;
    myWizFi.ReplyCheckTimer.CheckIsTimeout();
}
```

그림 13 WizFiBasicTest 예제의 Timer1 Callback 함수 구성

5.1.3. setup() 함수 구성

다음은 setup() 함수내의 구성입니다.

```
void setup() {
  byte key, retval, i;

  Serial.begin(9600);
  Serial.println("\r\nSerial Init");

  myClient =  WizFiClient(SIP, ServerPort);

  // initalize WizFi2x0 module:
  myWizFi.begin();

  // Timer1 Initialize
  Timer1.initialize(1000); // 1msec
  Timer1.attachInterrupt(Timer1_ISR);

  myWizFi.SendSync();
  myWizFi.ReplyCheckTimer.TimerStart(1000);

  Serial.println("Send Sync data");

  while(1)
  {
    if(myWizFi.CheckSyncReply())
    {
      myWizFi.ReplyCheckTimer.TimerStop();
      Serial.println("Rcvd Sync data");
      break;
    }
    if(myWizFi.ReplyCheckTimer.GetIsTimeout())
    {
      Serial.println("Rcving Sync Timeout!!");
      return;
    }
  }

  ///////////////
  // AP association
  while(1)
  {
    retval = myWizFi.associate(SSID, Key, WEP_SECURITY, true);

    if(retval == 1){
      Serial.println("WizFi2xo AP Associated");
      Wifi_setup = true;
      break;
```

```
    }else{
      Serial.println("AP association Failed");
    }
  }

}
```

그림 14 WizFiBasicTest 예제의 setup() 함수 구성

```
    Serial.begin(9600);
```

그림 15 WizFiBasicTest 예제의 시리얼 초기화 구문

시리얼 포트는 baud rate 9600bps로 초기화합니다.

```
    myClient = WizFiClient(SIP, ServerPort);

    // initalize WizFi2x0 module:
    myWizFi.begin();
```

그림 16 WizFiBasicTest 예제의 객체 초기화 부분

Client 소켓을 SIP와 ServerPort를 이용해서 생성하고, WizFi2x0Class 객체를 생성하고 초기화합니다.

```
    // Timer1 Initialize
    Timer1.initialize(1000); // 1msec
    Timer1.attachInterrupt(Timer1_ISR);
```

그림 17 WizFiBasicTest 예제의 Timer1 초기화 부분

Timer1 인터럽트의 주기를 1msec로 설정하고 Timer1 인터럽트의 callback 함수로 Timer1_ISR을 등록합니다.

```
      myWizFi.SendSync();
      myWizFi.ReplyCheckTimer.TimerStart(1000);
      while(1)
      {
              if(myWizFi.CheckSyncReply())
             {
                     myWizFi.ReplyCheckTimer.TimerStop();
                     break;
              }
             if(myWizFi.ReplyCheckTimer.GetIsTimeout())
            {
                     return;
              }
      }
```

그림 18 WizFiBasicTest 예제의 WizFi2x0 모듈과의 Sync 체크 부분

arduino 보드와 WizFi2x0 모듈간의 통신 여부를 check하기위해서 Sync를 주고 받습니다.

```
    retval = myWizFi.associate(SSID, Key, WEP_SECURITY, true);
```

그림 19 WizFiBasicTest 예제의 AP association 관련 구문

AP에 접속을 시도합니다. **AP에 설정된 보안 방식에 따라서 빨간색으로 표시한**

EncryptionType을 변경해야 합니다.

5.1.4. loop() 함수 구성

loop() 함수는 함수내의 코드가 무한 반복해서 호출되는 함수로 WizFiBasicTest 스케치의 loop() 함수 전체 코드는 다음과 같습니다.

```
void loop()
{
    uint8_t retval, i;
    byte rcvdBuf[129];
    byte cmd;
    byte TxBuf[100];

    memset(rcvdBuf, 0, 129);
    if(Wifi_setup){
        myWizFi.RcvPacket();
        if(myClient.available()){
            if(myClient.read(rcvdBuf)){
              Serial.print("CID[");
              Serial.print((char)myClient.GetCID());
              Serial.print("]");
              Serial.println((char *)rcvdBuf);
              myClient.write(rcvdBuf);
            }
            if(Disconnect_flag){
              retval = myClient.disconnect();
              Disconnect_flag = false;
              if(retval == 1)
              Serial.println("Disconnected! ");
              else
              Serial.println("Disconnection Failed");
            }
        }else{
          if(Connect_flag){
            retval = myClient.connect();
            Connect_flag = false;
            if(retval == 1)
                   Serial.println("Connected! ");
            else
                   Serial.println("Connection Failed");
          }
        }
        CheckConsoleInput();
    }

}
```

그림 20 WizFiBasicTest 예제의 loop() 함수 구성

```
myWizFi.RcvPacket();
```

그림 21 RcvPacket() 함수 호출 부분

WizFi2x0Class 객체의 RcvPacket() 함수는 WizFi2x0 모듈에서 나오는 데이터를 읽어 들여서 사용자의 데이터와 시스템 통지 메시지를 구분해주는 함수입니다.

주의) loop()내에서 반드시 호출되도록 해야합니다.

```
if(myClient.available()){
  if(myClient.read(rcvdBuf)){
    Serial.print("CID[");
    Serial.print((char)myClient.GetCID());
    Serial.print("]");
    Serial.println((char *)rcvdBuf);
    myClient.write(rcvdBuf);
  }
}
```

그림 22 Client 소켓 연결 상태 확인 및 데이터 송수신 처리 예제 구문

클라이언트 소켓이 서버에 연결되었고 수신한 데이터가 있다면 수신한 데이터를 시리얼 콘솔 창에 출력하고 그 데이터를 그대로 서버로 재전송하는 샘플입니다. 서버에 연결되었는지를 확인하는 부분이 myClient.available() 입니다. 수신한 데이터가 있는지를 확인하는 부분이 myClient.read(rcvdBuf) 입니다. 수신한 데이터들은 rcvdBuf에 복사하고 수신 바이트 수를 반환합니다. myClient.write(rcvdBuf) 함수 호출을 통해서 수신한 데이터를 서버로 재전송합니다.

```
retval = myClient.connect();
```

그림 23 Client 소켓의 연결 시도 부분

클라이언트 소켓이 서버에 접속하도록 요청하는 부분입니다. 본 예제에서는 사용자의 명령을 받아서 수행합니다.

```
retval = myClient.disconnect();
```

그림 24 Client 소켓의 연결 해제 시도 부분

클라이언트 소켓의 연결해제를 요청하는 부분입니다. 본 예제에서는 사용자의 명령을 받아서 수행합니다.

```
CheckConsoleInput();
```

그림 25 콘솔을 통한 사용자 입력 처리 함수 호출 부분

사용자의 명령에 대한 입력을 받아서 해당 명령에 대한 Flag를 setting하는 부분입니다.

이상으로 WizFiBasicTest 예제를 이용해서 WizFi Shield에서 프로그래밍을 하기 위해 필요한 절차와 방법에 대해서 살펴보았습니다.

5.2. WizFiBasicServerTest

5.2.1. WizFiBasicServerTest 의 개요

WizFiBasicServerTest 스케치는 다음과 같은 순서로 작업을 수행하는 예제입니다.

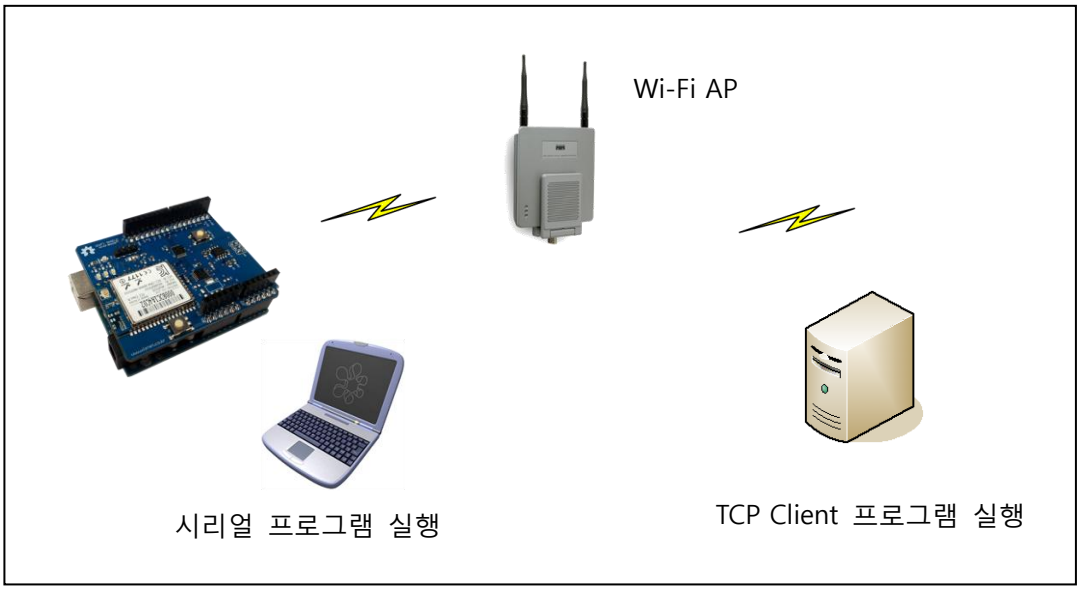


그림 26 WizFiBasicServerTest를 위한 네트워크 구성도

① WizFiBasicServerTest는 지정된 AP에 접속합니다.

② 클라이언트로부터의 TCP 연결이 이루어졌을 때, 해당 클라이언트 소켓과 데이터 통신을 할 WizFiClient 소켓을 4개 선언합니다.

③ WizFiServer 소켓 하나를 만들어서 지정한 포트번호로 클라이언트 소켓으로부터의 연결 시도를 기다립니다.

④ 각각의 WizFiClient 소켓은 상대방 시스템과 연결 된 경우, 수신한 데이터가 있는지 확인해서, 수신 데이터가 있으면 시리얼 포트로 그 값을 출력하고, 수신한 데이터를 상대방 시스템으로 재전송합니다. 이 동작을 계속적으로 수행합니다.

아래는 본 예제의 ③번까지 정상적으로 수행한 상태의 시리얼 창의 예 입니다.

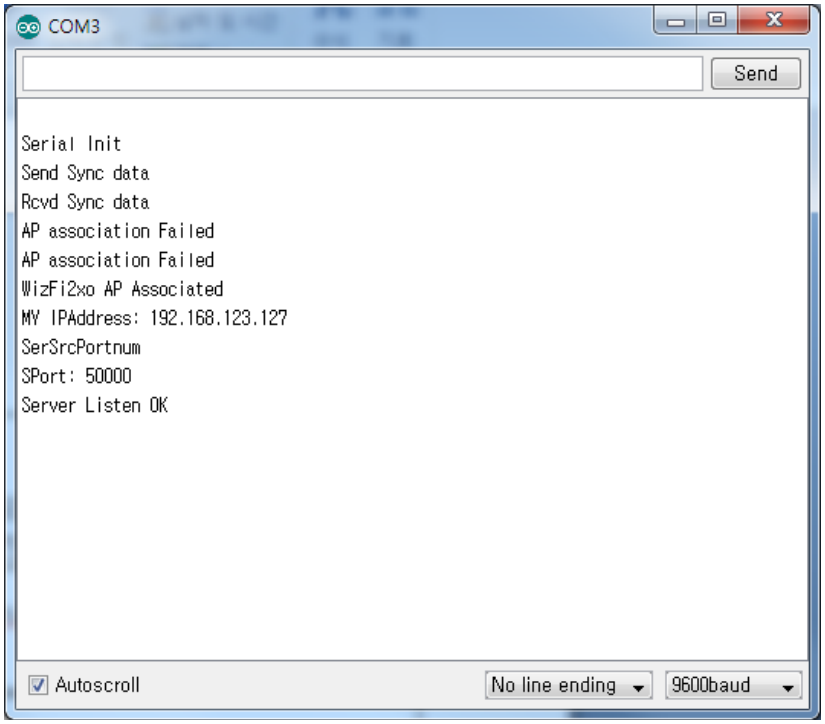


그림 27 WizFiBasicServerTest 예제의 정상 실행 결과 화면

5.2.2. 선언부

5.2.2.1. include 부

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <SPI.h>
#include <WizFi2x0.h>
#include <WizFiClient.h>
#include <WizFiServer.h>
#include <TimerOne.h>
```

그림 28 WizFiBasicServerTet 예제의 include 부

5.2.2.2. 변수 선언

```
#define SSID           "AP_SSID"     // SSID of your AP
#define Key            "1234567890" // Key or Passphrase
unsigned char SIP[4] = {192, 168, 123, 119};
unsigned int ServerPort = 5000;
unsigned int SrcPort = 5000; //WizFiServer Listen port #

WizFi2x0Class myWizFi;
WizFiClient myClient[4]; //the number of Client : 4
WizFiServer myServer(SrcPort);
TimeoutClass ConnectInterval;
```

그림 29 WizFiBasicServerTest 예제의 변수 선언 부

5.2.2.3. Timer1_ISR callback 함수

```
void Timer1_ISR()
{
    uint8_t i;
    myWizFi.ReplyCheckTimer.CheckIsTimeout();
}
```

그림 30 WizFiBasicServerTest 예제의 Timer1 Callback 함수 구성

5.2.3. setup() 함수의 구성

setup() 함수의 전체 구성은 다음과 같습니다.

```
void setup() {
  byte key, retval, i;
  byte retry_count = 0;
  byte tmpstr[64];

  Serial.begin(9600);
  Serial.println("\r\nSerial Init");

  for(i=0; i<4; i++)
    myClient[i] =  WizFiClient();

  // initalize WizFi2x0 module:
  myWizFi.begin();

  ConnectInterval.init();
  // Timer1 Initialize
  Timer1.initialize(1000); // 1msec
  Timer1.attachInterrupt(Timer1_ISR);

  myWizFi.SendSync();
  myWizFi.ReplyCheckTimer.TimerStart(1000);

  Serial.println("Send Sync data");
  while(1)
  {
    if(myWizFi.CheckSyncReply())
    {
      myWizFi.ReplyCheckTimer.TimerStop();
      Serial.println("Rcvd Sync data");
      break;
    }
    if(myWizFi.ReplyCheckTimer.GetIsTimeout())
    {
      Serial.println("Rcving Sync Timeout!!");
      return;
    }
  }

  ////////////////////////////////////////////////////
  // AP association
  while(1){
    byte tmpstr[32];

    retval = myWizFi.associate(SSID, Key, NO_SECURITY, true);

    if(retval == 1){
      myWizFi.GetSrcIPAddr(tmpstr);
      Serial.println("WizFi2xo AP Associated");
      Serial.print("MY IPAddress: ");
      Serial.println((char *)tmpstr);
      Wifi_setup = true;
      break;
    }else{
      Serial.println("AP association Failed");
    }
  }
```

```
  if(myServer.begin())
    Serial.println("Server Listen OK");
  else
    Serial.println("Server Listen Failed");
}
```

그림 31 WizFiBasicServerTest 예제의 setup() 함수

```
    for(i=0; i<4; i++)
       myClient[i] =  WizFiClient();

    // initalize WizFi2x0 module:
    myWizFi.begin();
```

그림 32 WizFiBasicServerTest 예제의 Client 소켓들 생성 구문 예

본 예제에서는 4개의 클라이언트와 연결을 허용하기 때문에 상대방 클라이언트 객체와 통신을 전담할 WizFiClient 객체 4개를 미리 생성합니다. 그리고, myWizFi.begin()을 호출하여 WizFi2x0 모듈 제어를 담당하는 객체인 myWizFi를 초기화합니다. 나머지 부분은 WizFiBasicTest 예제와 동일하고 다른 점은 다음과 같습니다.

```
  if(myServer.begin())
    Serial.println("Server Listen OK");
  else
    Serial.println("Server Listen Failed");
```

그림 33 WizFiBasicServerTest 예제의 Server 소켓 시작 예

위 코드는 SrcPort 변수에서 지정한 포트 번호로 연결 대기할 서버 소켓을 생성하는 부분입니다. 서버 소켓 생성이 성공하면 '1'을 반환하고, 실패하면 '0'을 반환합니다.

5.2.4. loop() 함수의 구성

loop() 함수의 전체적인 구성은 다음과 같습니다.

```
    void loop()
    {
       uint8_t retval, i;
       byte rcvdBuf[129];
       byte cmd;
       byte TxBuf[100];

       memset(rcvdBuf, 0, 129);
       if(Wifi_setup){
          myWizFi.RcvPacket();
          for(i=0; i<4; i++){
             if(myClient[i].available()){
                if(myClient[i].read(rcvdBuf)){
                   Serial.print("CID[");
           Serial.print((char)myClient[i].GetCID());
                   Serial.print("]");
                   Serial.println((char *)rcvdBuf);
                   myClient[i].write(rcvdBuf);
                }
```

```
                }
            }
        }
    }
```

그림 34 WizFiBasicServerTest 예제의 loop() 함수 내부 구성

loop() 함수의 동작은 WizFiBasicTest와 동일한 기능을 합니다.

단지, for 루프 구문을 통해서 소켓의 연결상태와 데이터 수신 여부를 4개의 myClient 객체별로 각각 확인해서 소켓별로 수신한 데이트를 시리얼 포트로 출력하고, 소켓에 대응하는 상대방 시스템에 수신한 데이터를 재전송하도록 하는 것만 다를 뿐입니다..

5.3. WizFiLimitedAPTest

5.3.1. WizFiLimitedAPTest 의 개요

LimitedAP 모드

스마트폰과 같은 Wi-Fi 장치를 이용해서 WizFi210 모듈과 통신을 하고자 할 때, 별도의 AP를 사용하지 않고 WizFi210 모듈을 AP로 설정해서 사용할 수 있습니다. 하지만, WizFi210 모듈은 Embedded 용으로 만들어진 모듈이라서 프로그램 메모리가 충분히 크지 않기 때문에 일반적인 AP에서 제공하는 기능들을 모두 제공하기에는 어려움이 있습니다. 그래서 스마트폰과 같은 Wi-Fi 장치와 연결하기 위해서 필수적으로 필요한 기능들만을 뽑아서 제한적인 기능을 가진 AP 처럼 동작하도록 구성하였습니다.

이 동작모드를 LimitedAP 모드라고 합니다.

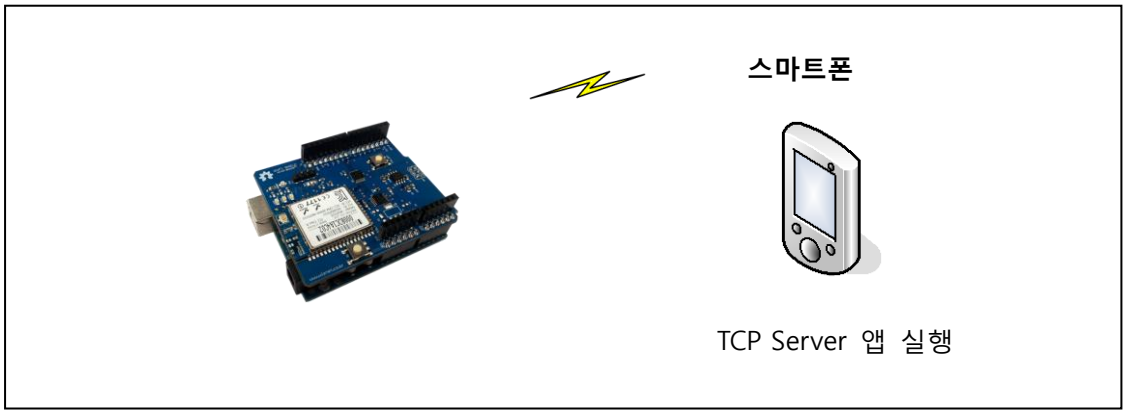


그림 35 WizFiLimitedAPTest 용 네트워크 구성도

WizFiLimitedAPTest 스케치는 다음과 같은 순서로 작업을 수행하는 예제입니다.

① WizFiLimitedAPTest는 SetOperatingMode() 함수를 호출하여 동작 모드를 "LIMITED_AP" 모드로 지정합니다.

② associate(…) 함수를 호출하여 WizFi210모듈을 AP로 구동합니다.

③ AP로 정상적인 초기화가 이루어지면 WizFiClient 소켓을 하나 만들어서 사용자의 명령을 기다립니다.

④ 시리얼 콘솔을 통해 서버 접속 명령(문자 'c' 또는 'C')을 받으면 WizFiClient 소켓은 서버에 연결을 시도합니다.

⑤ WizFiClient 소켓은 Peer 시스템과의 연결 된 경우, 수신한 데이터가 있는지 확인해서, 수신 데이터가 있으면 시리얼 포트로 출력하고, Peer 시스템에 수신한 데이터를 재전송합니다. 이 동작을 계속적으로 수행합니다.

아래는 본 예제가 정상적으로 수행되어 사용자의 서버 접속 명령을 기다리는 상태의 시리얼 창의 예 입니다.

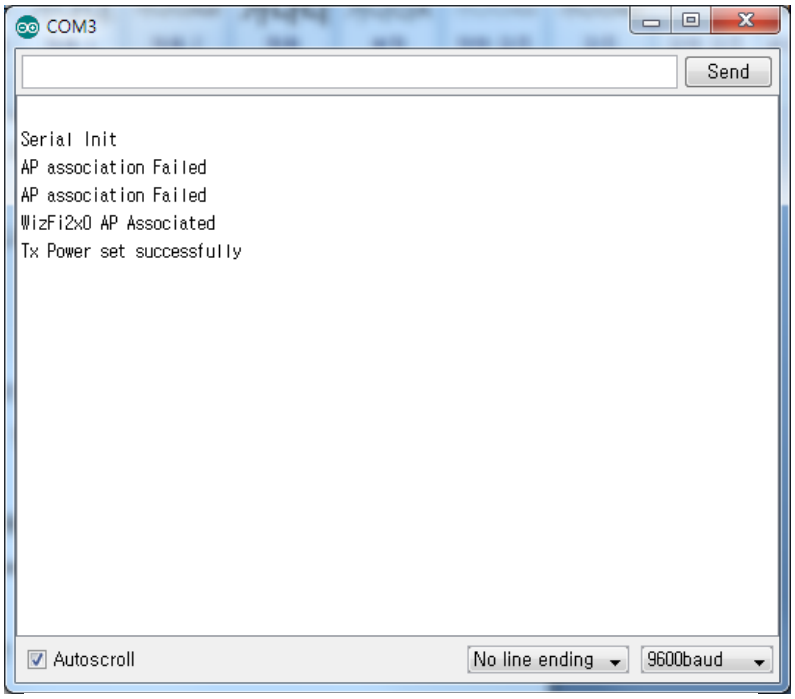


그림 36 WizFiLimitedAPTest 예제의 정상 실행 화면

5.3.2. 선언부

5.3.2.1. include 부

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <SPI.h>
#include <WizFi2x0.h>
#include <WizFiClient.h>
#include <TimerOne.h>
```

그림 37 WizFiLimitedAPTest 예제의 include 부

5.3.2.2. 변수 선언

```
#define SSID     "LimitedAp"   // replace with SSID of your AP
#define Key      "1234567890" // Key or Passphrase
unsigned char SIP[4] = {192, 168, 123, 119};
unsigned int ServerPort = 5000;
unsigned int SrcPort = 5000; //WizFiServer Listen port #

WizFi2x0Class myWizFi;
WizFiClient myClient;
TimeoutClass ConnectInterval;
```

그림 38 WizFiLimitedAPTest 예제의 변수 선언 부

SSID 문자열에 입력하는 값은 LimitedAP 모드로 동작할 때, Wi-Fi 단말 장치에서 보이는

SSID 값이 됩니다. 또 Key 문자열에 있는 값은 Wi-Fi 단말 장치들이 LimitedAP로 설정된 WizFi210 모듈에 접속할 때 사용해야하는 보안 키 값입니다.

5.3.2.3. Timer1_ISR callback 함수

```
    void Timer1_ISR()
    {
        uint8_t i;
        myWizFi.ReplyCheckTimer.CheckIsTimeout();
    }
```

그림 39 WizFiLimitedAPTest 예제의 Timer1 Callback 함수

5.3.3. setup() 함수의 구성

setup() 함수의 전체 구성은 다음과 같습니다.

```
void setup() {
  byte key, retval, i;
  byte retry_count = 0;
  byte tmpstr[64];

  Serial.begin(57600);
  Serial.println("\r\nSerial Init");

  // initalize WizFi2x0 module:
  myWizFi.begin();
  myClient =  WizFiClient(SIP, ServerPort);

  // Timer1 Initialize
  Timer1.initialize(1000); // 1msec
  Timer1.attachInterrupt(Timer1_ISR);

  myWizFi.SetSrcIPAddr((byte *)"192.168.55.1");
  myWizFi.SetSrcSubnet((byte *)"255.255.255.0");
  myWizFi.SetSrcGateway((byte *)"192.168.55.1");

  myWizFi.SetOperatingMode(LIMITEDAP_MODE);

  myWizFi.SendSync();
  myWizFi.ReplyCheckTimer.TimerStart(1000);

  Serial.println("Send Sync data");

  while(1)
  {
    if(myWizFi.CheckSyncReply()){
      myWizFi.ReplyCheckTimer.TimerStop();
      Serial.println("Rcvd Sync data");
      break;
    }
    if(myWizFi.ReplyCheckTimer.GetIsTimeout()){
      Serial.println("Rcving Sync Timeout!!");
      return;
    }
  }
  ////////////////////////////////////////////////////////
  // AP association
  while(1)
  {
    retval = myWizFi.associate(SSID, Key, WEP_SECURITY, false);

    if(retval == 1){
      Serial.println("WizFi2x0 AP Associated");
      Wifi_setup = true;
      break;
    }else{
      Serial.println("AP association Failed");
    }
  }
  retval = myWizFi.SetTxPower(5);
  if(retval == 1){ Serial.println("Tx Power set successfully"); }
```

```
  else{ Serial.println("Tx Power parameter is invalid"); }
  delay(1000);
}
```

그림 40 WizFiLimitedAPTest 예제의 setup() 함수 구성

Limited AP로 동작하면 WizFi2x0는 Gateway로 동작하게 됩니다. 따라서 WizFi2x0 모듈의 IP 주소, Subnet mask, Gateway 주소를 직접 지정해주어야 합니다.

```
  myWizFi.SetSrcIPAddr((byte *)"192.168.55.1");
  myWizFi.SetSrcSubnet((byte *)"255.255.255.0");
  myWizFi.SetSrcGateway((byte *)"192.168.55.1");
```

그림 41 WizFiLimtedAPTest 예제의 AP로써의 네트워크 정보 초기화 부분

본 예제에서는 위와 같이 지정하였습니다. Source IP Address와 Source Gateway Address가 "192.168.55.1"로 동일함을 확인할 수 있습니다.

```
myWizFi.SetOperatingMode(LIMITEDAP_MODE);
```

그림 42 WizFiLimitedAPTest 예제에서 WizFi2x0 모듈의 동작 모드 설정 부분

IP 주소 등의 설정을 마쳤으면 WizFi2x0 모듈의 동작 모드를 위와 같이 지정합니다.

```
retval = myWizFi.associate(SSID, Key, WEP_SECURITY, false);
```

그림 43 WizFi2x0 모듈이 AP로 동작하도록 하는 명령 부분

동작 모드 지정이 끝났으면 associate(…) 함수를 호출하여 WizFi2x0 모듈이 "Limited AP"로 초기화 되도록 합니다.

```
retval = myWizFi.SetTxPower(5);
```

그림 44 WizFi2x0 모듈의 출력 파워를 설정하는 코드

위 코드는 WizFi2x0 모듈의 RF 출력 레벨을 '5'로 지정하는 부분입니다.

5.3.4. loop() 함수의 구성

loop() 함수의 전체적인 구성은 다음과 같습니다.

```
void loop(){
  uint8_t retval, i;
  byte rcvdBuf[129];
  byte TxBuf[100];

  memset(rcvdBuf, 0, 129);
  if(Wifi_setup){
     myWizFi.RcvPacket();
     if(myClient.available()){
      if(myClient.read(rcvdBuf)){
```

```
        Serial.print("CID[");
        Serial.print((char)myClient.GetCID());
        Serial.print("]");
        Serial.println((char *)rcvdBuf);
        Serial.flush();
        myClient.write(rcvdBuf);
      }
      if(Disconnect_flag){
        retval = myClient.disconnect();
        Disconnect_flag = false;
      }
    }else{
      if(Connect_flag){
        retval = myClient.connect();
        Connect_flag = false;
      }
    }
    CheckConsoleInput();
  }
}
```

그림 45 WizFiLimitedAPTest 예제의 loop() 함수 구성

loop() 함수의 동작은 WizFiBasicTest와 동일한 기능을 합니다.

5.4. WizFiUDPClientTest

5.4.1. WizFiUDPClientTest의 개요

> WizFiUDPClientTest 예제는 데이터를 송수신할 상대방의 IP 주소와 포트 넘버를 알고 있는 경우에 WizFiUDP 객체를 생성하는 방법과 데이터 송수신 방법을 설명하는 예제입니다. 만약 상대방의 IP 주소와 포트 넘버를 모르는 상태에서 자기 자신의 포트넘버만 알고 있는 응용을 만드는 경우에는 5.5 절의 WizFiUDPServerTest 예제를 참조하세요.

WizFiUDPClientTest 스케치는 다음과 같은 순서로 작업을 수행하는 예제입니다.

① WizFiUDP 객체를 하나 생성합니다.

② associate(…) 함수를 호출하여 AP에 연결합니다.

③ AP에 연결이 되면 WizFiUDP 객체에 지정된 정보를 토대로 UDP 소켓을 생성합니다.

④ 100ms 단위로 동작하는 Timer(ConnectInterval)의 counter를 시작합니다.

⑤ ConnectInterval 이 Timeout이 되면(100ms가 경과되었다는 의미임) "Time elapsed..I'm alive"라는 문자열을 UDP로 전송합니다.

⑥ ④와 ⑤의 동작을 반복합니다.

아래는 본 예제가 정상적으로 수행된 경우의 시리얼 창의 예 입니다.

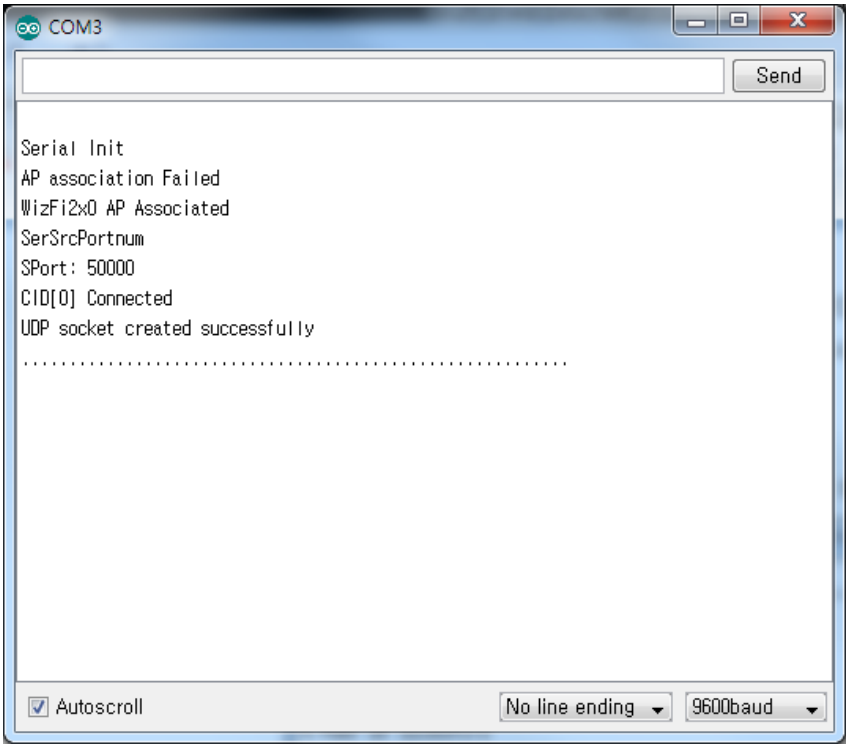


그림 46 WizFiUDPClientTest의 실행 화면 예

5.4.2. 선언부

5.4.2.1. include 부

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <SPI.h>
#include <WizFi2x0.h>
#include <WizFiUDP.h>
#include <TimerOne.h>
```

그림 47 WizFiUDPClientTest 예제의 include 부

UDP 소켓을 사용해야 하기 때문에 WizFiUDP.h 헤더파일을 포함합니다.

5.4.2.2. 변수 선언

```
#define SSID        "AP_SSID"       // SSID of your AP
#define Key         "1234567890"    // Key or Passphrase

unsigned char  SIP[4]             = {192, 168, 123, 170};
unsigned int ServerPort = 50000;

WizFi2x0Class myWizFi;
WizFiUDP myUDP;
TimeoutClass ConnectInterval;
```

그림 48 WizFiUDPClientTest 예제의 변수 선언부

WizFiUDP 타입의 객체 myUDP를 선언합니다.

5.4.2.3. Timer1_ISR Callback 함수

```
void Timer1_ISR()
{
  uint8_t i;

  myWizFi.ReplyCheckTimer.CheckIsTimeout();
  ConnectInterval.CheckIsTimeout();
}
```

그림 49 WizFiUDPClientTest 예제의 Timer1_ISR callback 함수 부분

100ms 단위로 UDP 패킷을 전송할 것이기 때문에 100ms 단위로 시간 경과를 체크할 변수인 ConnectInterval의 시간 경과를 업데이트할 CheckIsTimeout() 멤버 함수를 Timer1_ISR Callback() 함수 내에서 호출합니다.

Timer1_ISR() 은 1ms가 경과할 때마다 호출되는 Callback 함수로 이 안에서

ConnectInterval.CheckIsTimeout()를 호출하면 ConnectInterval 객체의 지정시간이 경과되었는지를 정확하게 업데이트할 수 있습니다.

5.4.3. setup() 함수의 구성

setup() 함수의 전체 구성은 다음과 같습니다.

```
void setup()
{
  byte key, retval, i;
  byte retry_count = 0;
  byte tmpstr[64];

  Serial.begin(9600);
  Serial.println("\r\nSerial Init");

  // initalize WizFi2x0 module:
  myWizFi.begin();
  myUDP =  WizFiUDP(SIP, ServerPort, 50000);

  ConnectInterval.init();

  // Timer1 Initialize
  Timer1.initialize(1000); // 1msec
  Timer1.attachInterrupt(Timer1_ISR);


  myWizFi.SendSync();
  myWizFi.ReplyCheckTimer.TimerStart(1000);

  //Serial.println("Send Sync data");

  while(1)
  {
    if(myWizFi.CheckSyncReply())
    {
      myWizFi.ReplyCheckTimer.TimerStop();
      //Serial.println("Rcvd Sync data");
      break;
    }
    if(myWizFi.ReplyCheckTimer.GetIsTimeout())
    {
      //Serial.println("Rcving Sync Timeout!!");
      return;
    }
  }

  ////////////////////////////////////////////////////////////
  // AP association
  while(1)
```

```
        {
          retval = myWizFi.associate(SSID, Key, WEP_SECURITY, true);

          if(retval == 1){
            Serial.println("WizFi2x0 AP Associated");
            Wifi_setup = true;
            break;
          }else{
            Serial.println("AP association Failed");
          }
        }

        if(myUDP.open())
          Serial.println("UDP socket created successfully");
        else
          Serial.println("UDP socket creation failed");

        ConnectInterval.TimerStart(100);
    }
```

그림 50 WizFiUDPClientTest 예제의 setup() 함수 구성

```
      myUDP =  WizFiUDP(SIP, ServerPort, 50000);
```

그림 51 WizFiUDP 객체 생성 코드 예

상대방의 IP 주소, 포트 넘버와 자신의 포트넘버를 지정해서 WizFiUDP 객체인 myUDP를 생성합니다.

```
      if(myUDP.open())
```

그림 52 WizFiUDP 객체를 통해 UDP 소켓 생성 코드 예

WizFiUDP 클래스의 멤버함수인 open()을 호출해서 실제로 UDP 소켓을 생성합니다.

5.4.4. loop() 함수의 구성

loop() 함수의 전체적인 구성은 다음과 같습니다.

```
      void loop()
      {
        uint8_t retval, i;
        byte rcvdBuf[129];
        byte cmd;
        byte TxBuf[100];

        memset(rcvdBuf, 0, 129);

        if(Wifi_setup)
        {
          myWizFi.RcvPacket();
```

```
        if(myUDP.available())
        {
          if(ConnectInterval.GetIsTimeout())
          {
            myUDP.write((byte *)"Time elapsed..I'm alive");
            ConnectInterval.TimerStart();
            Serial.println("Time elapsed..I'm alive ");
          }
        }
      }
    }
```

그림 53 WizFiUDPClientTest 예제의 loop() 함수 구성

```
myUDP.write((byte *)"Time elapsed..I'm alive");
```

그림 54 WizFiUDP 객체를 통해 UDP 데이터를 전송하는 코드

WizFiUDP 클래스의 멤버 함수 write()를 호출해서 UDP 패킷을 전송합니다.

5.5. WizFiUDPServerTest

WizFiUDPServerTest는 WizFiUDP 객체를 생성할 때, 상대방의 IP 주소, 포트넘버 없이 자신의 포트넘버 만을 이용합니다. 나머지는 WizFiUDPClientTest와 동일하지만 다른 예를 위해서 임의의 데이터를 수신하면 UDP 데이터를 송신한 상대방의 IP 주소와 포트넘버를 시리얼 창에 출력하고 그 데이터를 상대방에게 되돌려 주고 특정한 IP 주소와 포트 넘버에게도 전송하는 예제를 구성하였습니다.

아래는 WizFiUDP 객체를 생성할 때의 코드 예입니다.

```
myUDP = WizFiUDP((uint8_t *)NULL, 0, SrcPort);
```

그림 55 자기 포트 넘버만을 이용해서 WizFiUDP 객체를 생성하는 코드 예

```
void loop()
{
  uint8_t retval, i;
  byte rcvdBuf[129];
  byte cmd;
  byte TxBuf[100];
  byte tmpIP[16];
  uint16_t tmpPort;

  memset(rcvdBuf, 0, 129);

  if(Wifi_setup)
  {
    myWizFi.RcvPacket();
    if(myUDP.available())
    {
      if(myUDP.read(rcvdBuf))
      {
          Serial.print("CID[");
          Serial.print((char)myUDP.GetCID());
          Serial.print("]");
          Serial.println((char *)rcvdBuf);
          myUDP.GetCurrentDestInfo(tmpIP, &tmpPort);
          Serial.println((char *)tmpIP);
          sprintf((char *)tmpIP, "Portnum: %u", tmpPort);
          Serial.println((char *)tmpIP);
          myUDP.write(rcvdBuf);
          myUDP.SetCurrentDestInfo((byte *)"192.168.123.170", 5500);
          myUDP.write(rcvdBuf);
      }
    }
  }
}
```

그림 56 WizFiUDPServerTest 예제의 loop() 함수 구성 예