



Arduino 原始碼讀書會(V) : Arduino Standard Libraries 重點解析 (下)



DMP Electronics Inc. (瞻營全電子)
robotics@dmp.com.tw

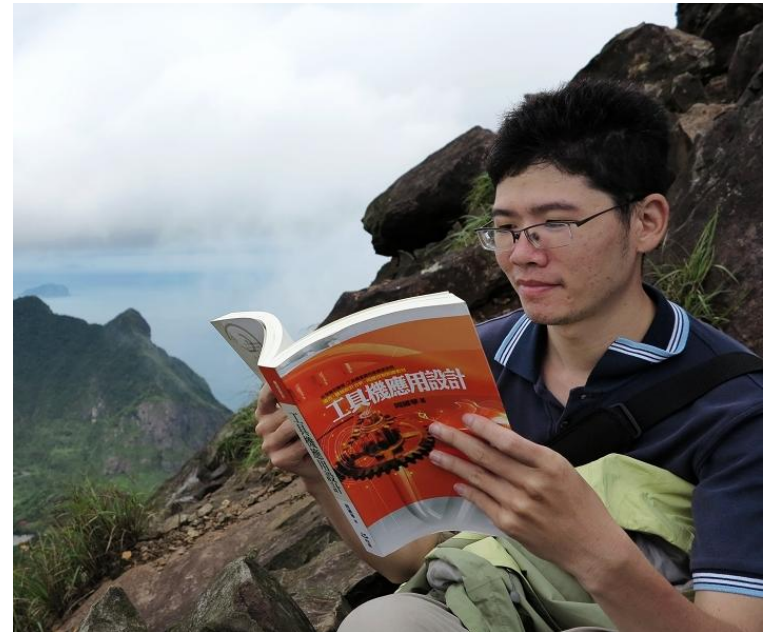
講者介紹

- 郭宏澤
- 瞻營全 RD
 - 等級2
 - 打怪練級中
- 技能: 86Duino 未來軟體維護與擴展



講者介紹

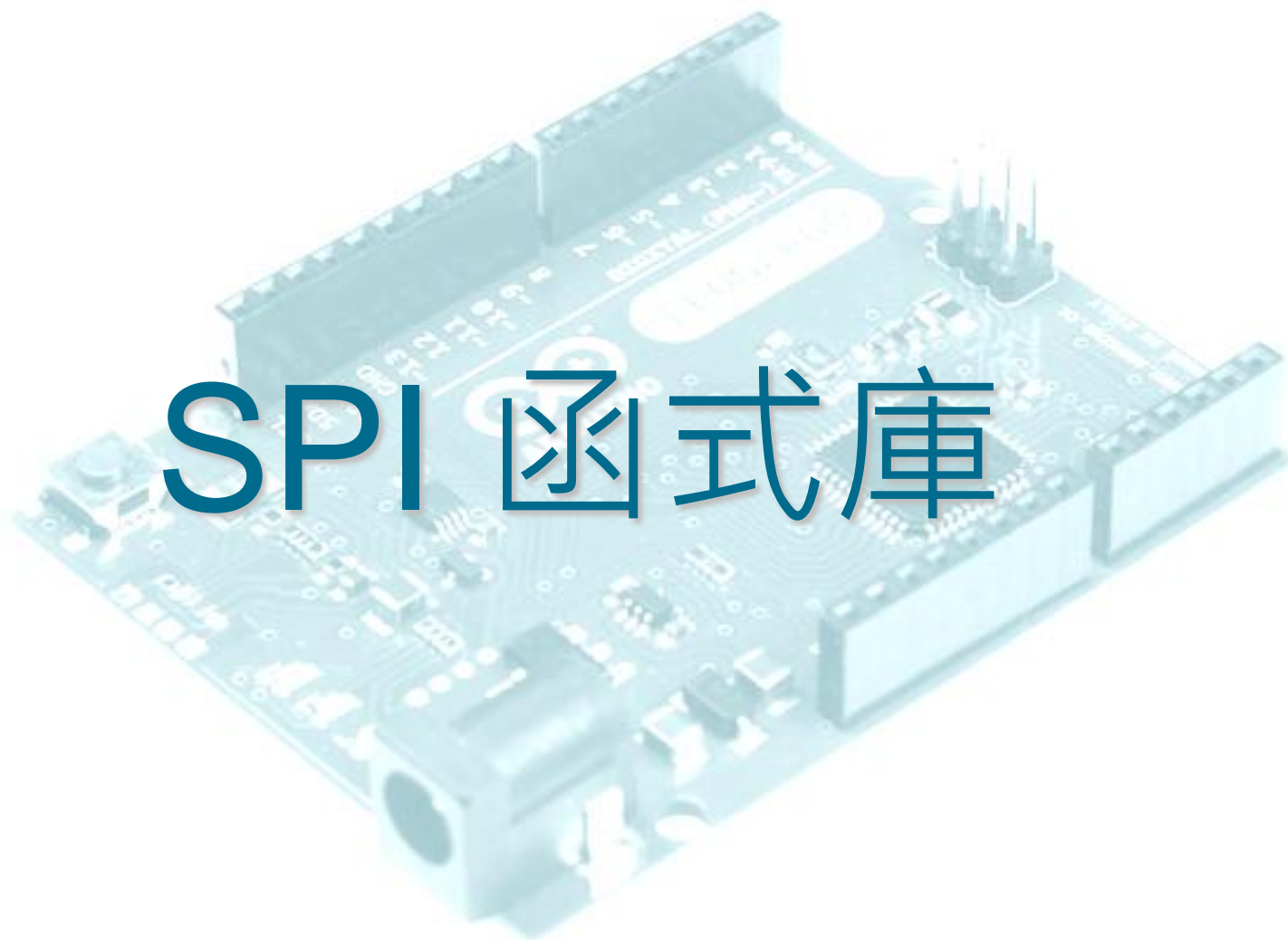
- 林恩州
- 瞻營全 RD
 - 等級4
 - 目前與郭宏澤一起組隊打怪
- 技能:
86Duino 基礎軟體開發



Arduino 原始碼讀書會大綱

- SPI 函式庫
- Ethernet 函式庫
- WiFi 函式庫
- GSM 函式庫
- SoftwareSerial 函式庫
- TFT 函式庫

SPI 函式庫



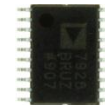
SPI

- 全名：Serial Peripheral Interface
- 許多電子裝置都有用到它，例如：

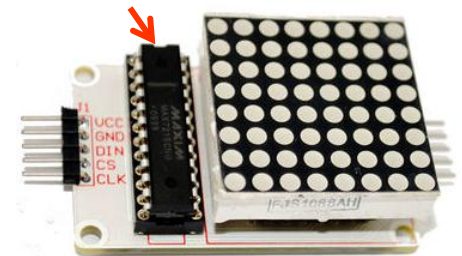
- SD 記憶卡



- 數位/類比轉換 IC (例如 AD7928)



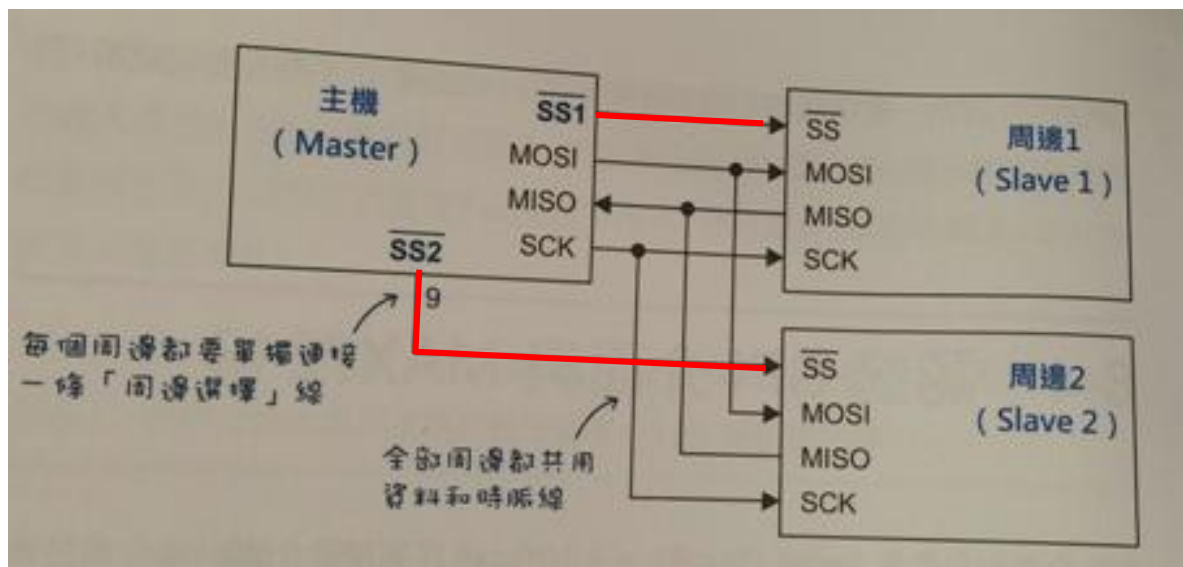
- LED 控制晶片 (例如 MAX7219)



- 還有很多
這裡無法一一列出...

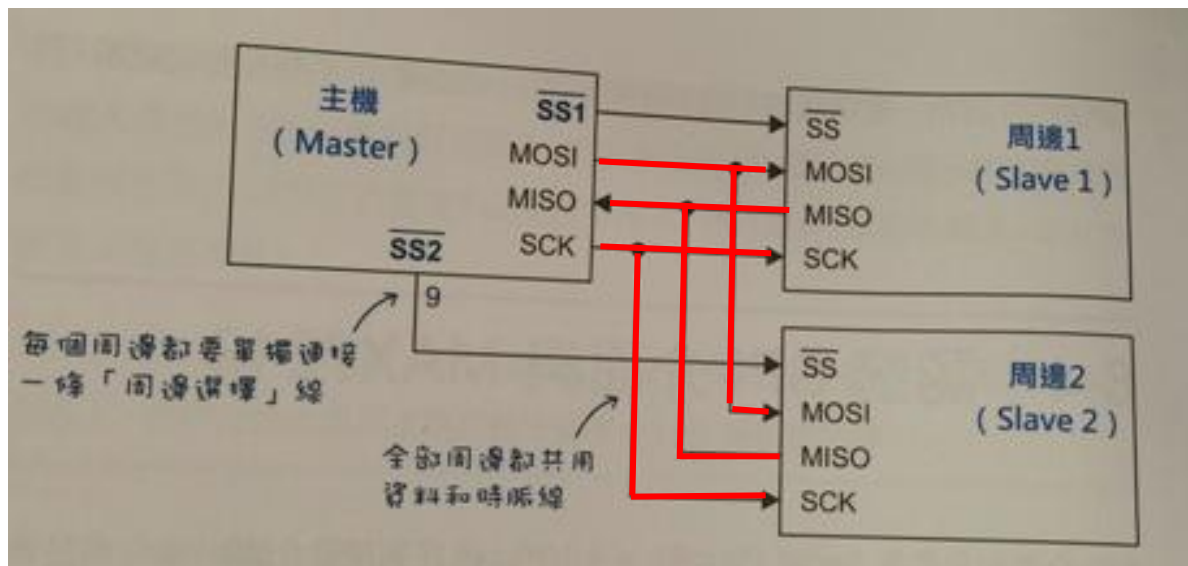
SPI 介面

- SPI 採用四條線連接主機和周邊設備, 這四條線的名稱和用途如下：
 - SS：周邊選擇線(**S**lave **S**elect)，指定要連接哪一個周邊設備。這條線也稱為 CS (Chip Select 晶片選擇線)



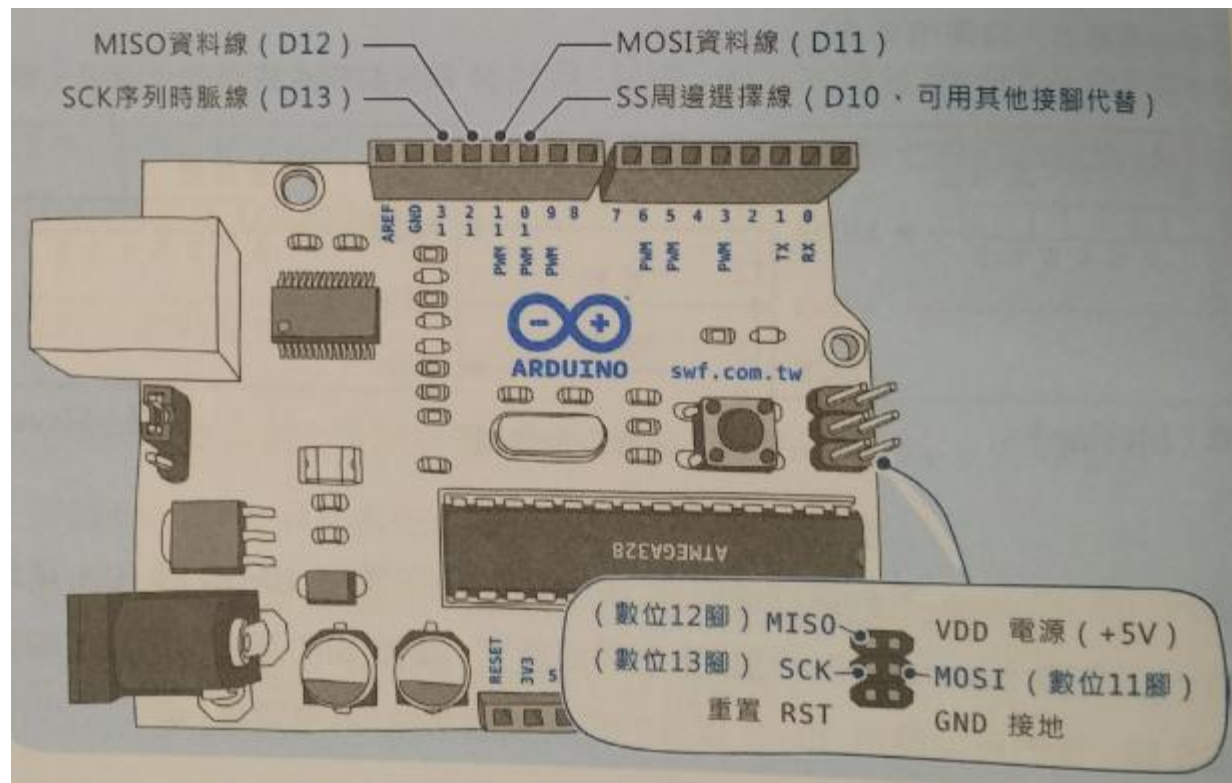
SPI 介面 (續)

- MOSI：從主機(master)送往周邊(slave)的資料線，
(Master Output Slave Input)
- MISO：從周邊(slave)送往主機(master)的資料線，
(Master Input Slave Output)
- SCK：序列時脈線

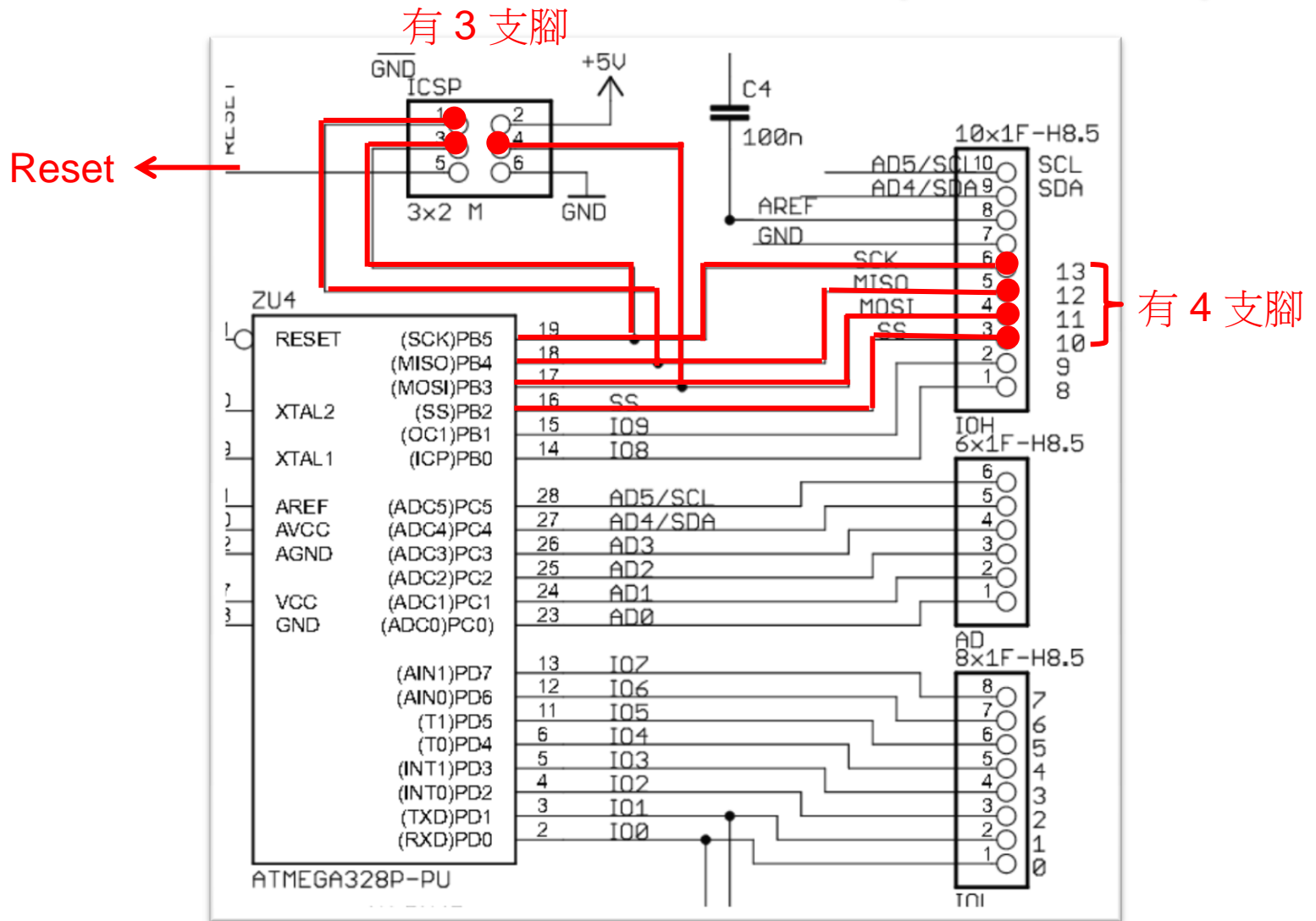


Arduino UNO 上的 SPI 腳位

有用 UNO 燒錄過 Arduino bootloader 的人，對 SPI 腳位應該不陌生。如下圖所示：



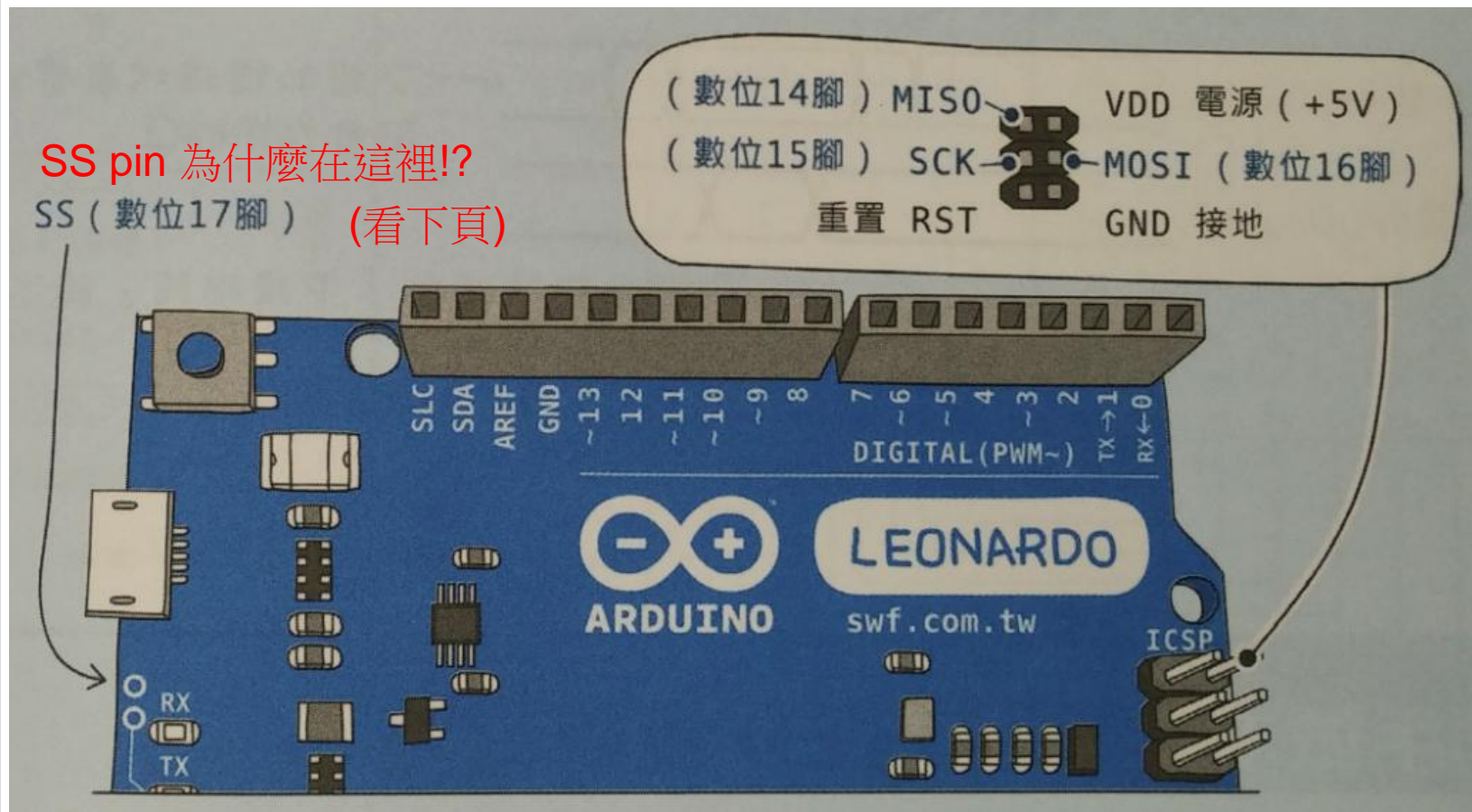
UNO 上的 SPI 腳位 (電路圖)



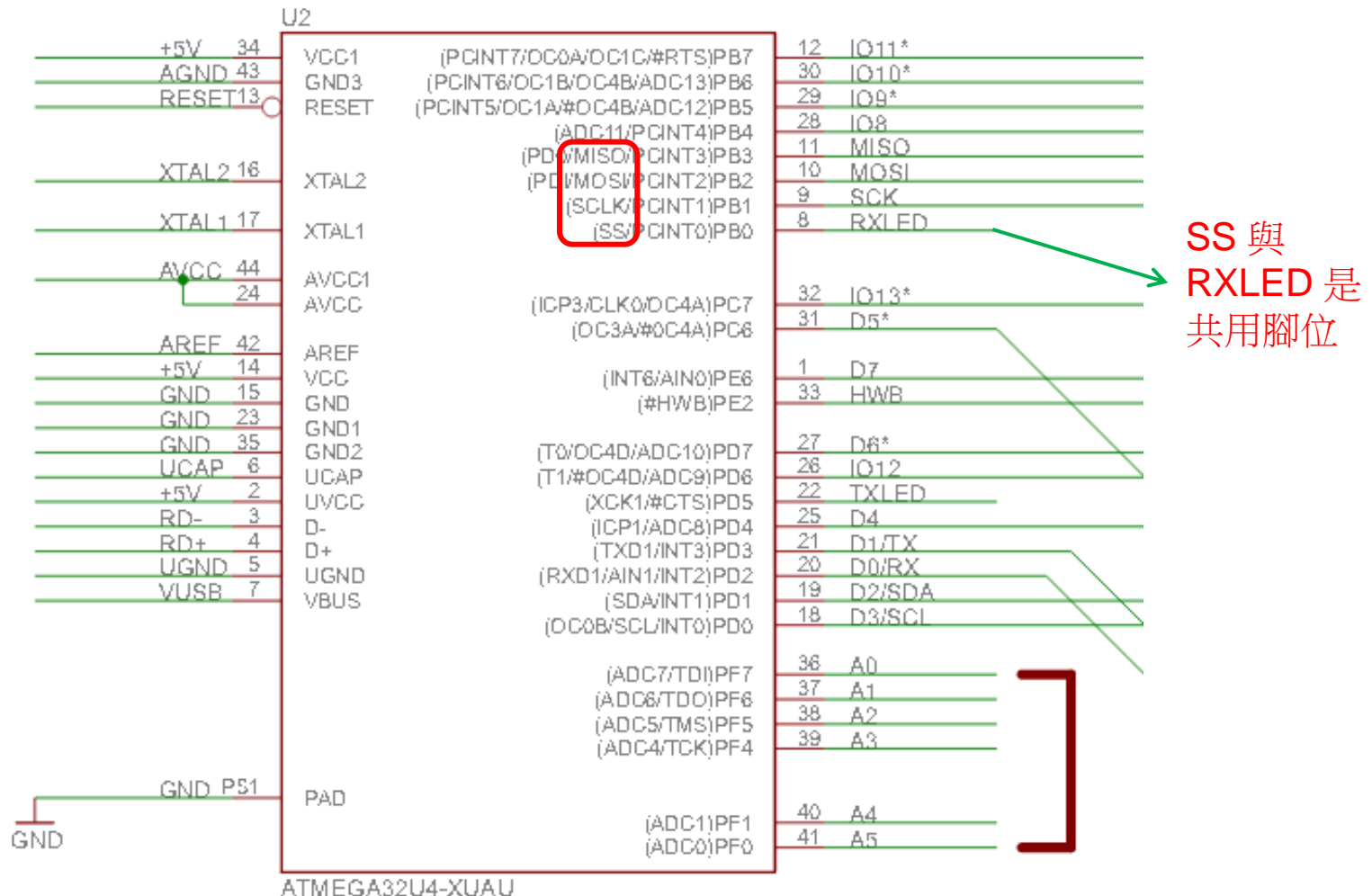
Arduno Leonardo 上的 SPI 腳位

SS pin 為什麼在這裡!?

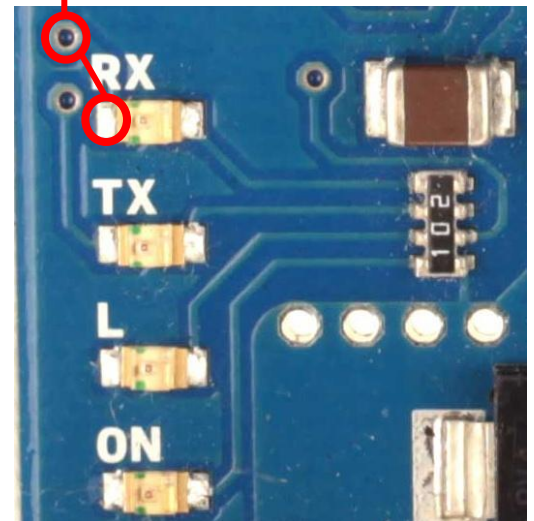
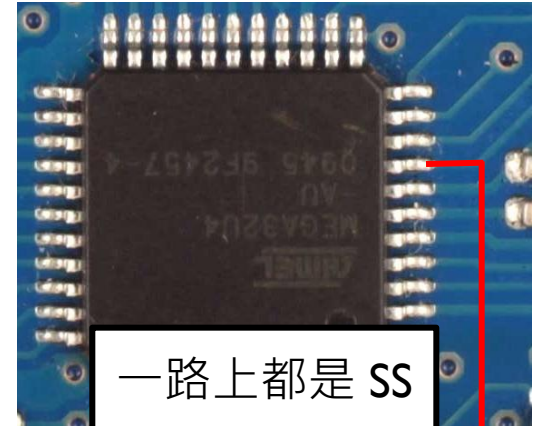
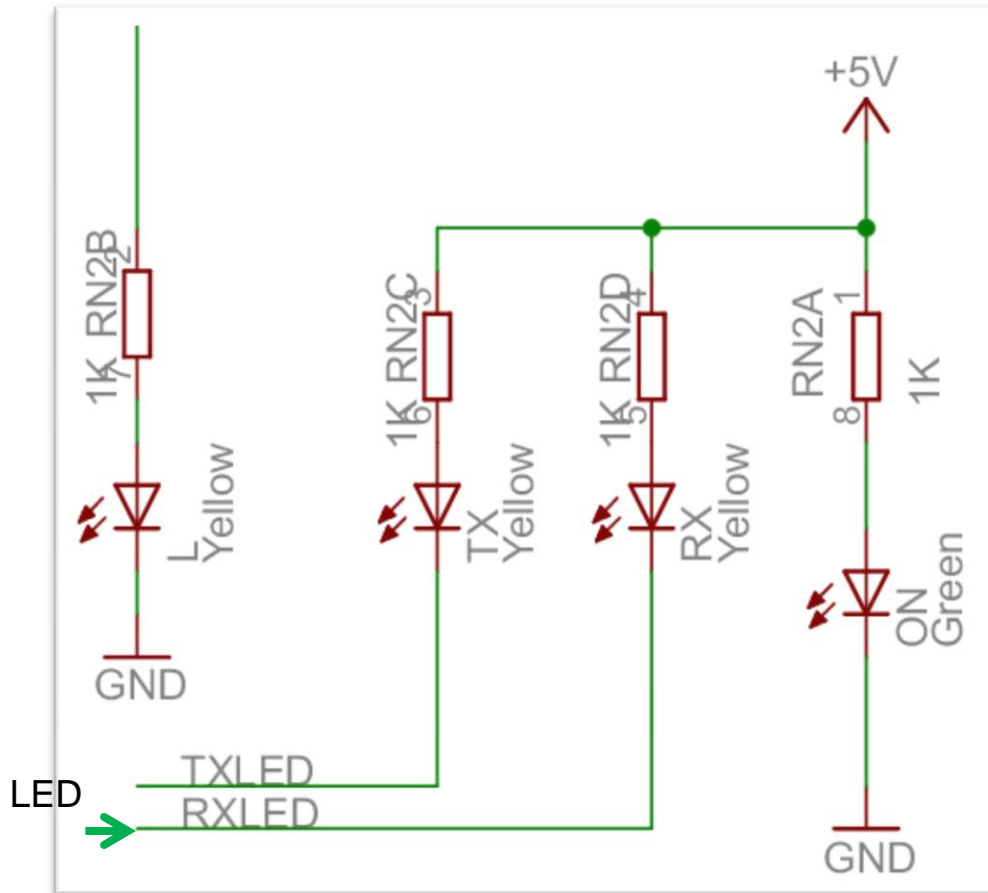
SS (數位17腳) (看下頁)



Leonardo 上的 SPI 腳位 (電路圖)



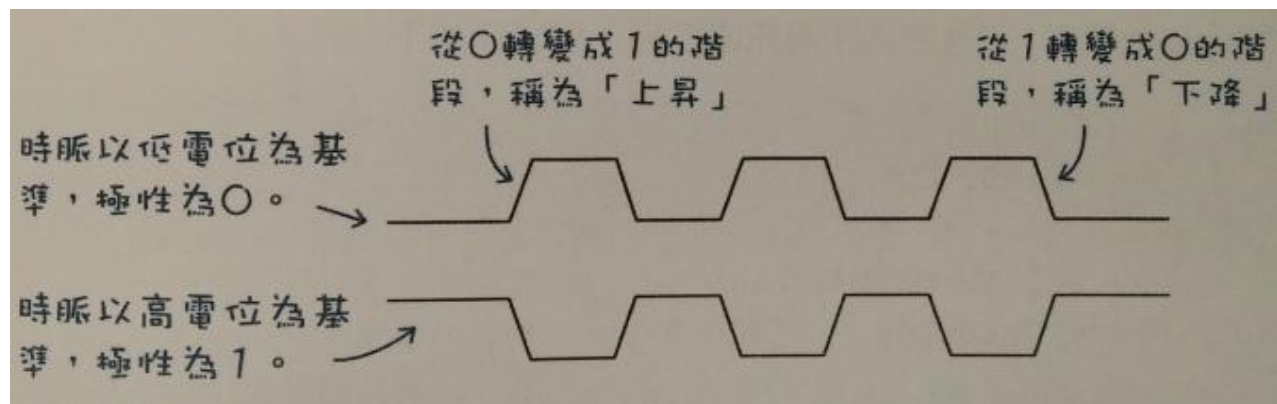
電路圖中搜尋 RXLED 關鍵字



心中 OS：或許選一支 GPIO 來當 SS 會比較方便

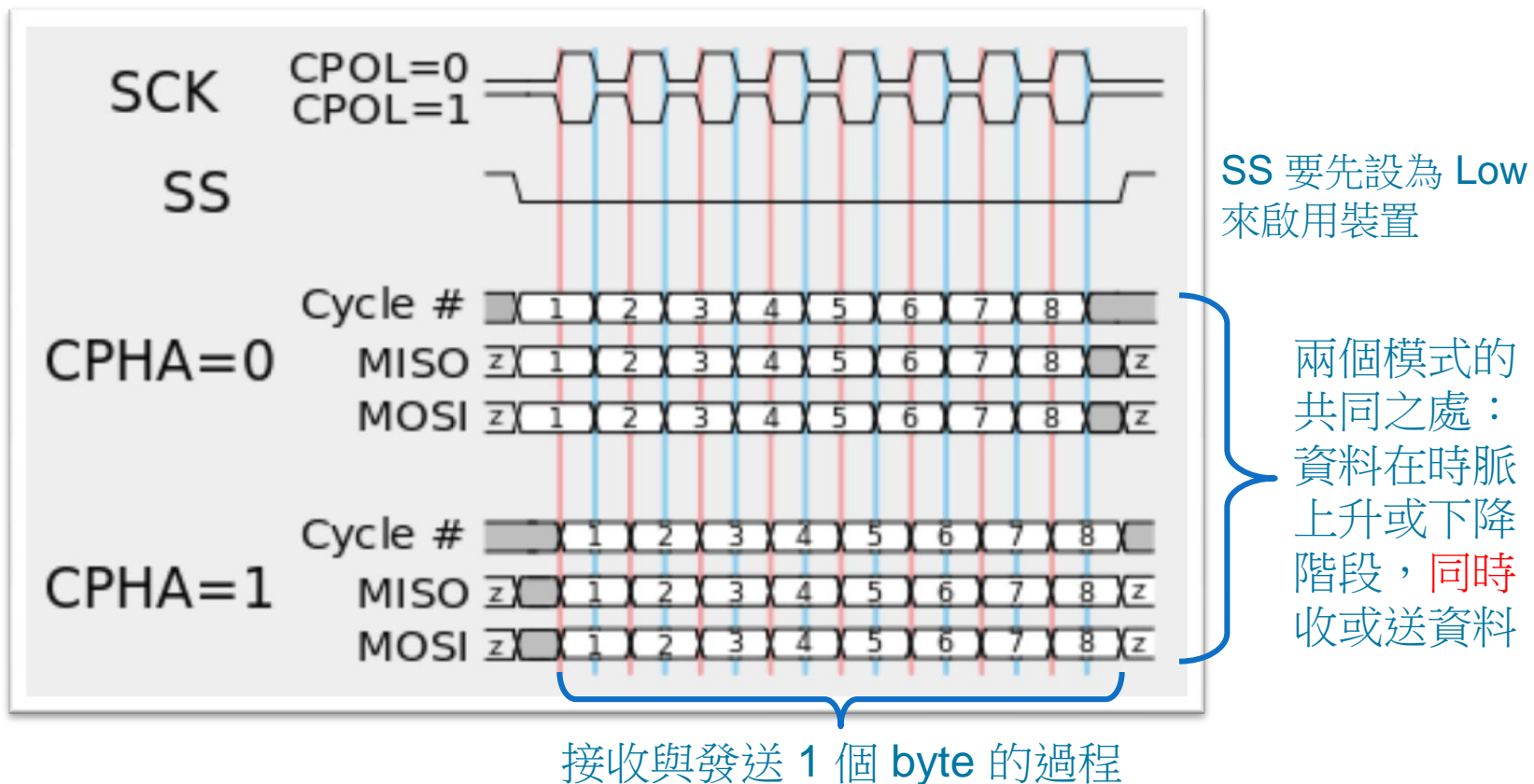
SPI 的通訊格式

- SPI 是一種同步**全雙工**的序列埠，主機和周邊之間的資料傳遞，都要跟著時脈的 **High**、**Low** 一同進行
- SPI 介面沒有強制規範時脈訊號的標準，大部分是由 SPI 介面晶片來決定使用哪一種時脈訊號格式
- 一般來說，SPI 可以由 **CPOL** 和 **CPHA** 來組成四種不同的格式：
 - **CPOL**：時脈極性，時脈信號的電位基準



SPI 的通訊格式 (續)

- CPHA：時脈相位，資料在時脈的上升階段或者下降階段被讀取/送出



SPI 的通訊格式 (續)

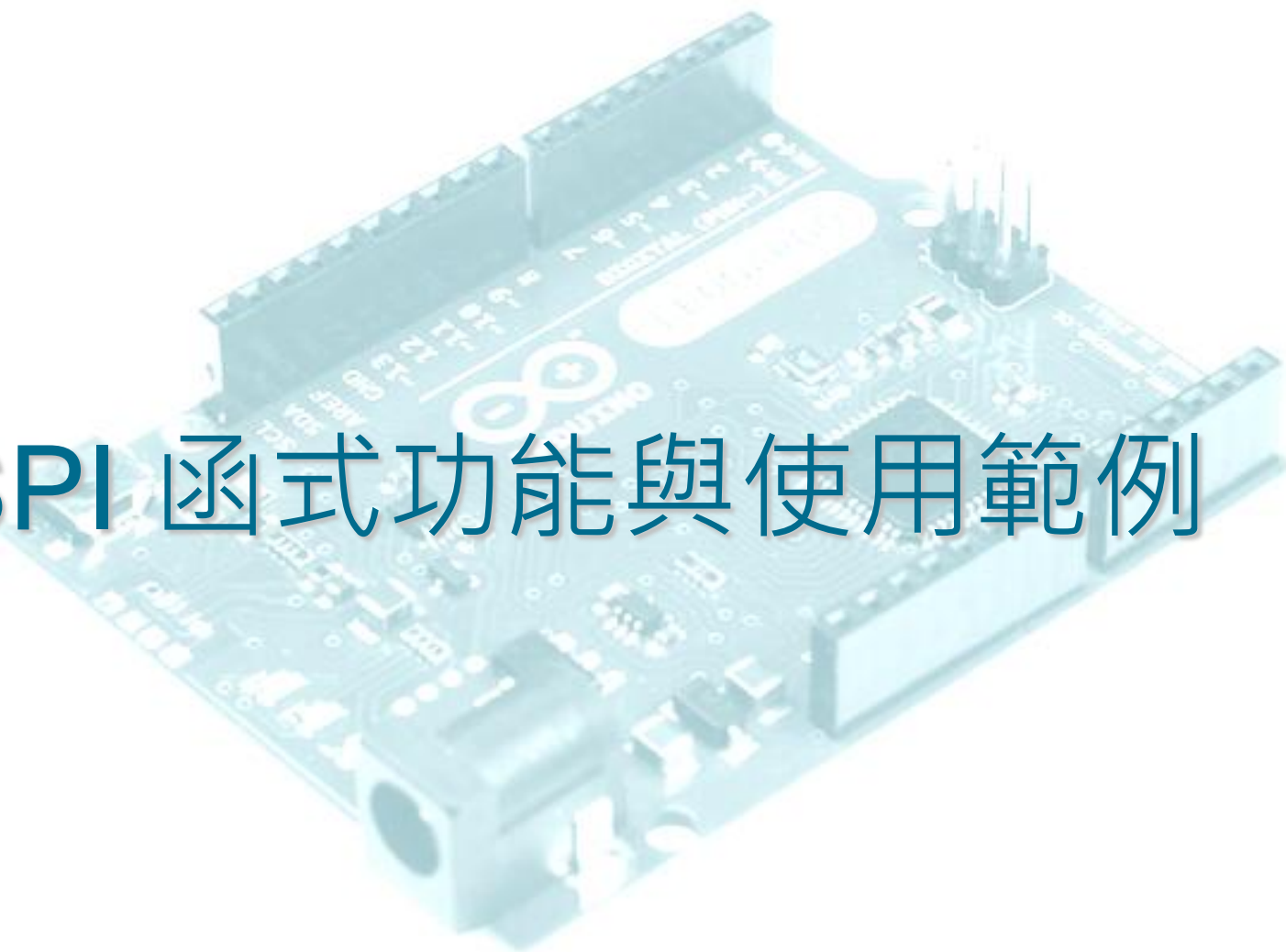
- CPOL 和 CPHA 配合後，四種組合如下：

Arduino 函
式庫中的
命名

資料模式名稱	時脈極性 (CPOL)	時脈相位 (CPHA)
SPI_MODE0	0	0 (上昇階段)
SPI_MODE1	0	1 (下降階段)
SPI_MODE2	1	0
SPI_MODE3	1	1

- 與裝置通訊前的確認事項
 - 資料傳遞的格式
 - 訊號傳遞位元順序 (bit order)：分成高位元先傳 (MSBFIRST) 和低位元先傳 (LSBFIRST) 兩種
 - 裝置所能接收的最高時脈頻率

SPI 函式功能與使用範例



SPI.begin()

- 函式功能：
 - 初始化 SPI 硬體
 - 設定為 Master 端
 - 預設資料格式是 $\text{CPOL} = 0$, $\text{CPHA} = 0$
 - 預設傳輸速度是 4MBps
 - 預設傳遞位元順序是高位元先傳 (MSBFIRST)

SPI.end()

- 函式功能：
 - 關閉 SPI 硬體功能

SPI. setBitOrder()

- 函式功能：
 - 設定傳遞位元順序，可輸入兩種參數：
 - MSBFIRST
 - LSBFIRST

SPI.setDataMode()

- 函式功能：
 - 設定資料格式，可輸入四種參數：
 - SPI_MODE0
 - SPI_MODE1
 - SPI_MODE2
 - SPI_MODE3

SPI.setClockDivider()

- 函式功能：
 - 設定傳輸速度，可輸入七種參數：
 - SPI_CLOCK_DIV2 : 8MBps
 - SPI_CLOCK_DIV4 : 4MBps
 - SPI_CLOCK_DIV8 : 2MBps
 - SPI_CLOCK_DIV16 : 1MBps
 - SPI_CLOCK_DIV32 : 500KBps
 - SPI_CLOCK_DIV64 : 250KBps
 - SPI_CLOCK_DIV128 : 125KBps

SPI.transfer()

- 函式功能：
 - 傳送並同時接收資料

SPI 使用範例

- 傳送/接收資料

```
Int data;
```

```
void setup() {  
    SPI.begin();  
}
```

```
void loop() {  
    digitalWrite(SS, LOW); // 把 SS 設定為 LOW，開始傳送資料  
    SPI.transfer(0x01); // 送 0x01 給 slave  
    data = SPI.transfer(0x00); // 讀取 slave 回傳的值  
    digitalWrite(SS, HIGH); // 把 SS 設定為 HIGH，結束資料傳送  
    Serial.println(data);  
    delay(10);  
}
```

這裡傳送的資料是隨便給的
實際要看 **slave** 晶片而定

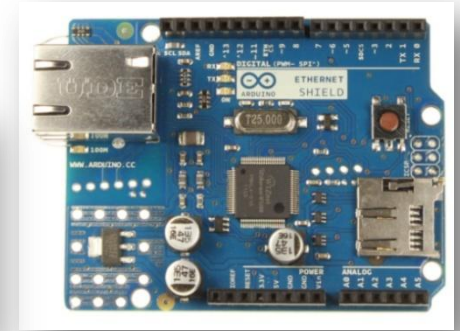
SPI Library 常常被引用

- 它們都是 SPI 介面：

- Ethernet Library
- TFT Library
- SD Library
- Wifi Library
- SpiderL3S Library



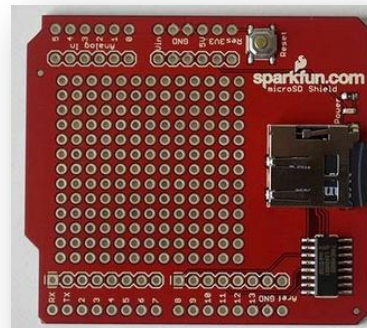
Arduino TFT



Arduino Ethernet shield



SpiderL3S
(CC3000 Wifi module)

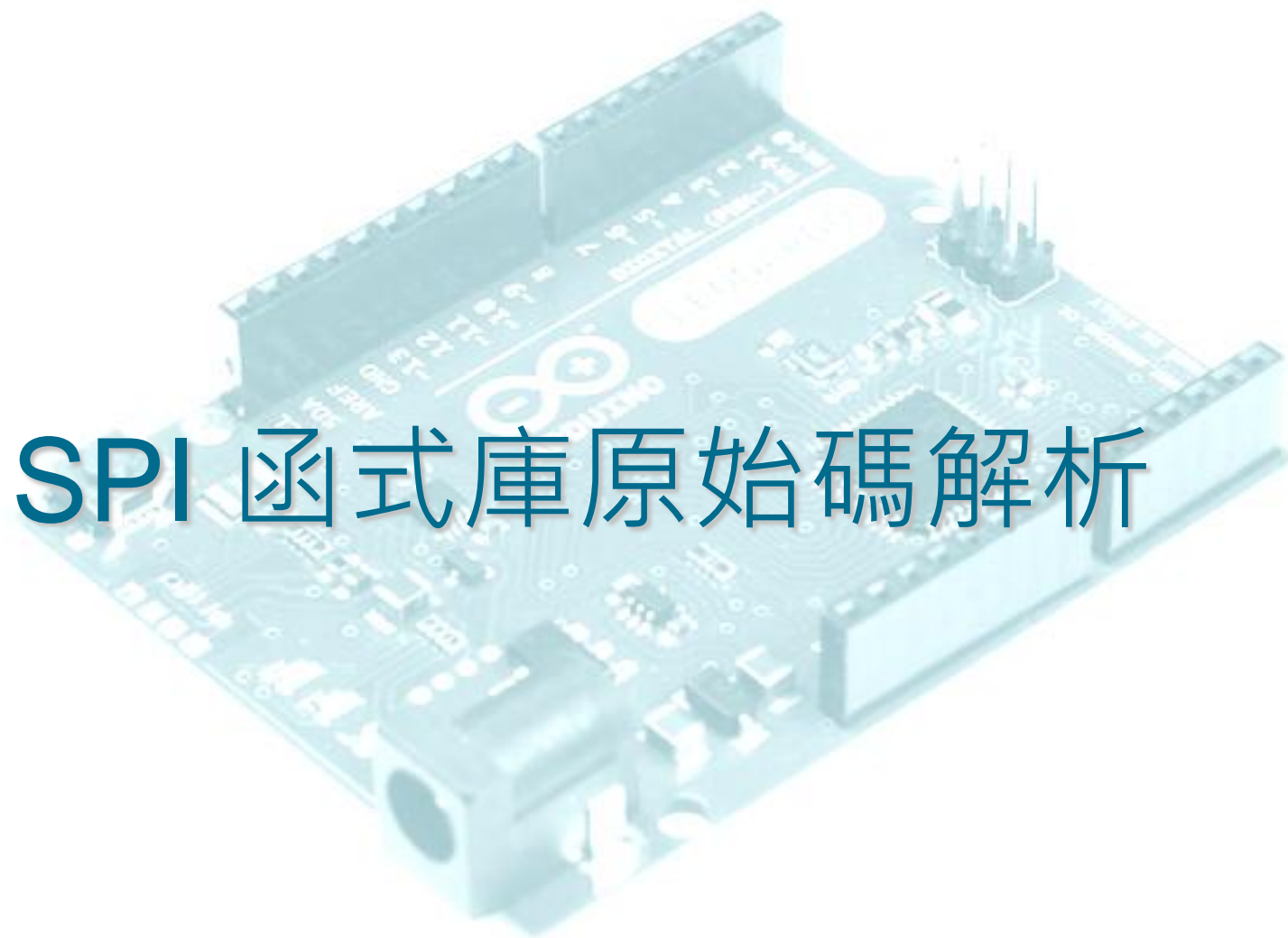


SD shield



Arduino Wifi shield

SPI 函式庫原始碼解析



SPI 函式庫資料夾內容

<https://github.com/arduino/Arduino/tree/master/libraries/SPI>

The screenshot shows the GitHub repository for the Arduino SPI library. At the top, it indicates the current branch is 'master'. Below this, a commit by 'pmjdebruijn' from July 19th is shown, with the message 'SPI: remove duplicate BarometricPressureSensor example'. The file list on the left includes 'examples', 'SPI.cpp', 'SPI.h', and 'keywords.txt'. The 'SPI.cpp' and 'SPI.h' files are highlighted with a red rectangle. The commit messages for these files are 'just change a comment to make more sense' and 'Fixing warnings in Ethernet library (Paul Stoffregen)' respectively.

File	Commit Message
examples	SPI: remove duplicate BarometricPressureSensor example
SPI.cpp	just change a comment to make more sense
SPI.h	Fixing warnings in Ethernet library (Paul Stoffregen).
keywords.txt	Renaming SPI constants to better match general style. Renaming setDat...

SPI 函式庫原始碼重要細節

SPI.cpp : begin()

```
void SPIClass::begin() {  
  
    // Set SS to high so a connected chip will be "deselected" by default  
    digitalWrite(SS, HIGH);  
  
    // When the SS pin is set as OUTPUT, it is no longer a general purpose  
    // output port, and will only work for SPI operations).  
    pinMode(SS, OUTPUT);  
  
    // Warning: if the SS pin ever becomes a LOW INPUT then SPI  
    // automatically switches to Slave, so the data direction of  
    // the SS pin MUST be kept as OUTPUT.  
    SPCR |= _BV(MSTR);  
    SPCR |= _BV(SPE);  
  
    // Set direction register for SCK and MOSI pin.  
    // MISO pin automatically overrides to INPUT.  
    // By doing this AFTER enabling SPI, we avoid accidentally  
    // clocking in a single bit since the lines go directly  
    // from "input" to SPI control.  
    // http://code.google.com/p/arduino/issues/detail?id=888  
    pinMode(SCK, OUTPUT);  
    pinMode(MOSI, OUTPUT);  
}
```

將 SS pin 設定為 output HIGH
確保等一下 enable SPI 後會是
Master 狀態

將 SPI 設定為 Master 然後 enable

SPI enable 後，SCK 和 MOSI pin
的方向需自行定義

SPI 函式庫原始碼重要細節

SPI.cpp : end()

```
void SPIClass::end() {  
    SPCR &= ~_BV(SPE);  
}
```

SPI.cpp : setBitOrder()

```
void SPIClass::setBitOrder(uint8_t bitOrder)  
{  
    if(bitOrder == LSBFIRST) {  
        SPCR |= _BV(DORD); → LSBFIRST  
    } else {  
        SPCR &= ~(_BV(DORD)); → MSBFIRST  
    }  
}
```

SPI 函式庫原始碼重要細節

SPI.cpp : setDataMode()

```
void SPIClass::setDataMode(uint8_t mode)
{
    SPCR = (SPCR & ~SPI_MODE_MASK) | mode;
}
```

SPI.cpp : setClockDivider()

```
void SPIClass::setClockDivider(uint8_t rate)
{
    SPCR = (SPCR & ~SPI_CLOCK_MASK) | (rate & SPI_CLOCK_MASK);
    SPSR = (SPSR & ~SPI_2XCLOCK_MASK) | ((rate >> 2) & SPI_2XCLOCK_MASK);
}
```

設定 divider 暫存器

SPI 函式庫原始碼重要細節

SPI.h : transfer()

```
byte SPIClass::transfer(byte _data) {  
    SPDR = _data;  
    while (!(SPSR & _BV(SPIF)) )  
        ;  
    return SPDR;  
}
```

→ 將要送的資料填入 SPDR 暫存器

→ 資料送完的時候，SPIF 會設 1
(設 1 後第一次對它讀取會清 0)

→ 回傳收到的值

An 86Duino board, a single-board computer based on the Intel 86 microprocessor. It features a USB port, a DC-IN jack, a reset button, and various digital and analog pins. The board is labeled with 'SOM1', 'A2', 'A1', 'B64', 'B63', 'DIGITAL (PWR)', 'ANALOG IN', 'POWER', 'ESET', 'DC-IN', 'USB', 'RESET', 'DM222', 'REV: 1.0', and '2013/08/14'.

SPI 函式庫的移植: 以 86Duino 為例

SPI 在 86Duino 上的移植

- 移植重點：
 - 將填 ATmega CPU 內的 SPI 暫存器的行為，改成填 86Duino CPU 內的 SPI 暫存器
 - 其它與硬體無關的程式碼幾乎無需改動，可直接延用

86Duino spi.cpp : begin()

```
void SPIClass::begin() {  
    void *pciDev = NULL;
```

在 86Duino 中，SPI 是一個 PCI 裝置，
所以需要先取得 I/O address

```
// Get SPI device base address
```

```
pciDev = pci_Alloc(0x00, 0x10, 0x01);  
if(pciDev == NULL) {printf("SPI device don't exist\n"); return;}  
SPI_IOaddr = (unsigned) (pci_In16(pciDev, 0x10) & 0xFFFFFFFF0L);  
pci_Free(pciDev);
```

```
io_outpb(SPI_IOaddr + 7, FULLDUPLEX); 設定 SPI 為全雙工
```

```
io_outpb(SPI_IOaddr + 7, io_inpb(SPI_IOaddr + 7) & 0xF1 | SPI_MODE0);
```

```
io_outpb(SPI_IOaddr + 0x0b, 0x08);
```

預設傳輸格式為
SPI_MODE0

```
setClockDivider(13); 預設速度為 4MHz
```

```
useFIFO(); 開啟 FIFO 功能
```

```
detachInterrupt();
```

```
setSS(HIGH); 將 SS (實際上是 SPICS) 設定為 HIGH
```

```
}
```

86Duino spi.cpp : end()

```
void SPIClass::end() {  
    if(SPI_IOaddr == 0) return;  
}
```

86Duino spi.cpp : setBitOrder()

```
void SPIClass::setBitOrder(uint8_t bitOrder)  
{  
    if(SPI_IOaddr == 0) return;  
    if(bitOrder == LSBFIRST)          設定為 LSBFIRST  
        io_outpb(SPI_IOaddr + 7, io_inpb(SPI_IOaddr + 7) | LSBSHIFT);  
    else                               設定為 MSBFIRST  
        io_outpb(SPI_IOaddr + 7, io_inpb(SPI_IOaddr + 7) & ~LSBSHIFT);  
}
```

86Duino spi.cpp : setDataMode()

```
void SPIClass::setDataMode(uint8_t mode) {  
    if(SPI_IOaddr == 0) return;  
    io_outpb(SPI_IOaddr + 7, (io_inpb(SPI_IOaddr + 7) & 0xF1) | mode);  
}
```

設定 SPI 傳輸模式

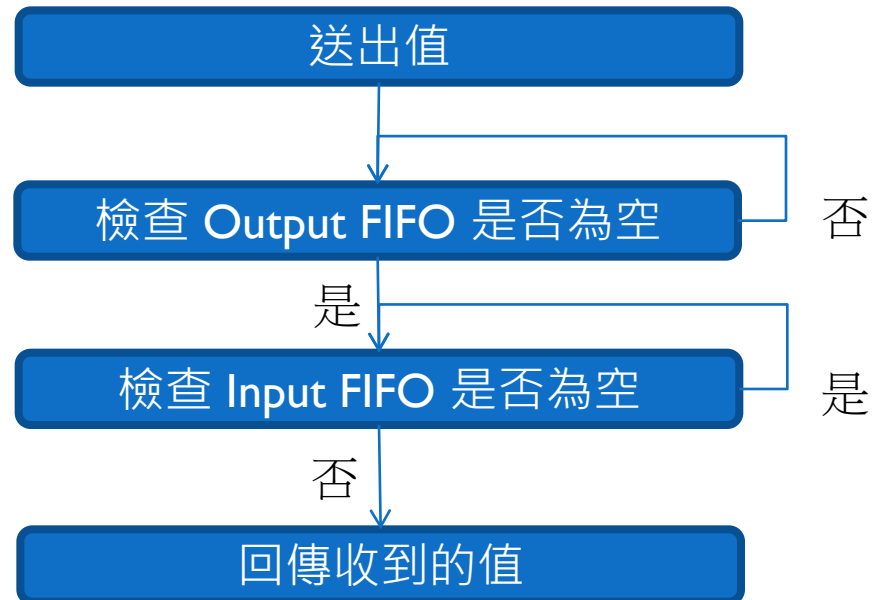
86Duino spi.cpp : setClockDivider()

```
void SPIClass::setClockDivider(uint16_t rate)  
{  
    if(rate == 0 || rate > 4095) return;  
    if(SPI_IOaddr == 0) return;  
    if(rate > 15)  
        io_outpb(SPI_IOaddr + 6, (rate&0x0ff0)>>4);  
  
    io_outpb(SPI_IOaddr + 2, (io_inpb(SPI_IOaddr + 2) & 0xF0) | (rate%16));  
}
```

限制值的範圍在 1 ~ 4095，因為 divider register 只有 12 bit

86Duino spi.cpp : transfer()

```
byte SPIClass::transfer(byte data) {  
    if(SPI_IOaddr == 0) return 0;  
    io_outpb(SPI_IOaddr, data);  
    while((io_inpb(SPI_IOaddr + 3) & 0x08) == 0);  
    while((io_inpb(SPI_IOaddr + 3) & 0x20) == 0);  
    return io_inpb(SPI_IOaddr + 1);  
}
```



不同 Arduino 板子的 SPI 速度差異

- 在 Arduino UNO 上，SPI 速度只有固定那七種
- 在 Arduino Due 及 86Duino 上，可允許更快的 SPI 速度
 - 在新版的 Arduino IDE 1.5.x 新增了 `beginTransaction()` 來使用更快的 SPI 速度
 - 但目前 Arduino 網站尚未提供關於此函式的使用文件~冏rz
 - 另一種使用更快 SPI 速度的方式是直接對 `setClockDivider()` 輸入對應的 `divider` 數值 (而不是像 `SPI_CLOCK_DIV2` 這樣的常數)
 - 但使用此法需要先知道不同 Arduino 板子上 SPI 速度的計算方式

86Duino SPI clock 的計算方式

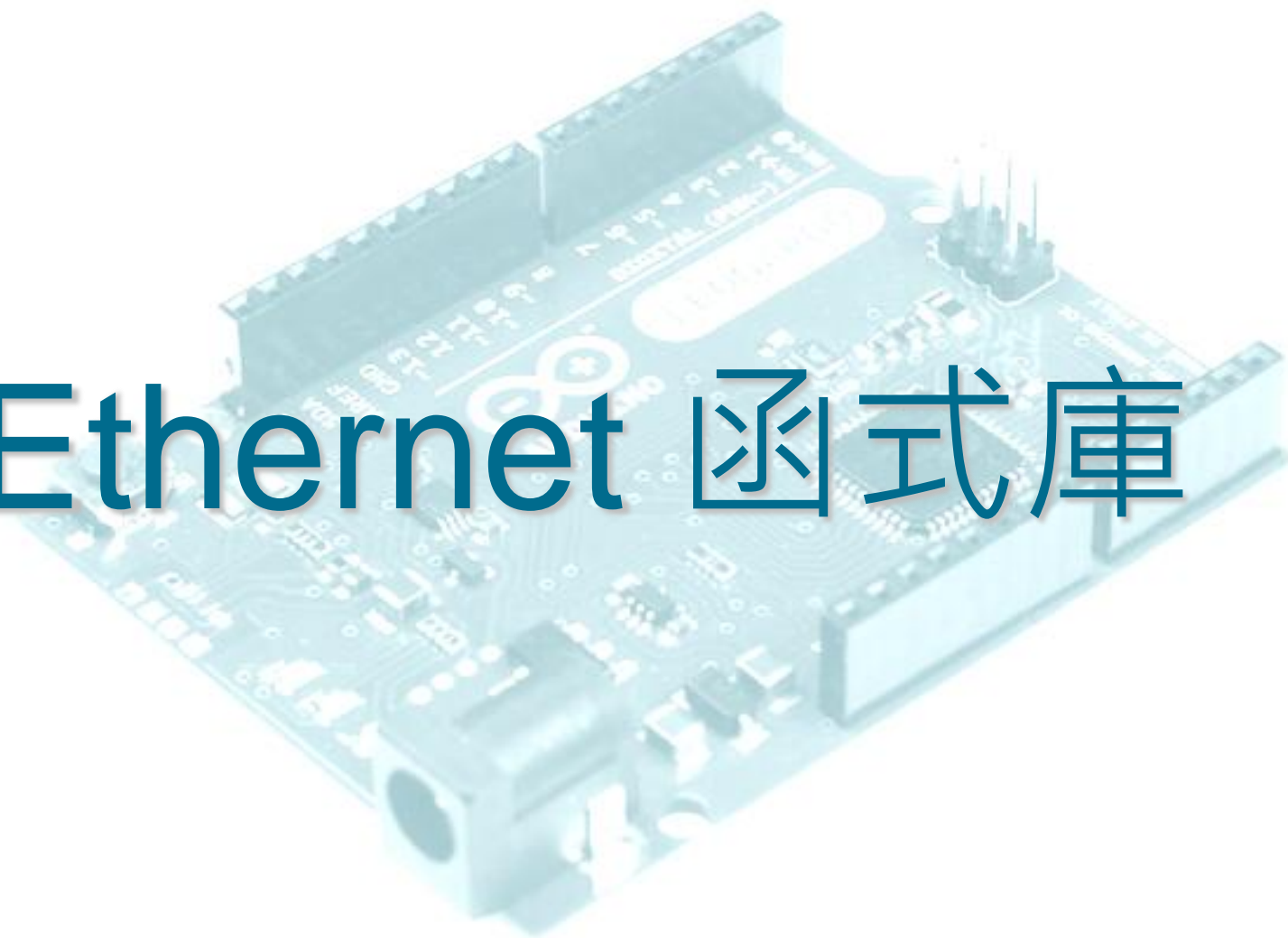
- 在 86Duino 上，計算 SPI 速度的公式如下：

$$100\text{MHz} / (2 \times \text{div})$$

div 是 divider 值，範圍：1 ~ 4095

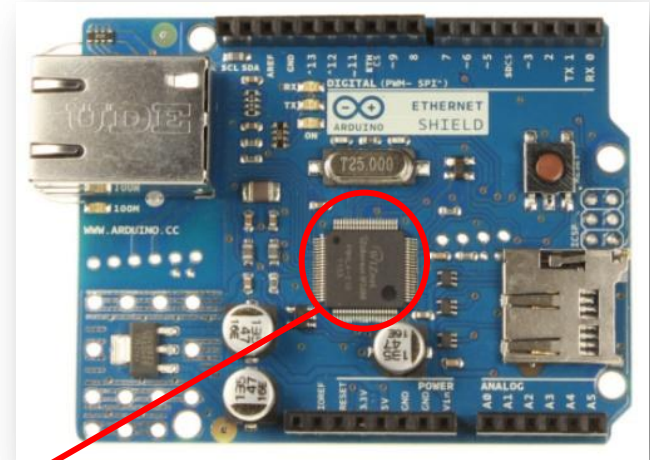
- 當 div 設成 1 時，86Duino 最快可輸出的 SPI clock 速度：50MHz

Ethernet 函式庫



Arduino Ethernet shield 簡介

- 操作電壓：5V
- 控制晶片：W5100
- 速度：10/100 Mbps
- 通訊介面：SPI



Ethernet shield 外觀



WIZnet W5100

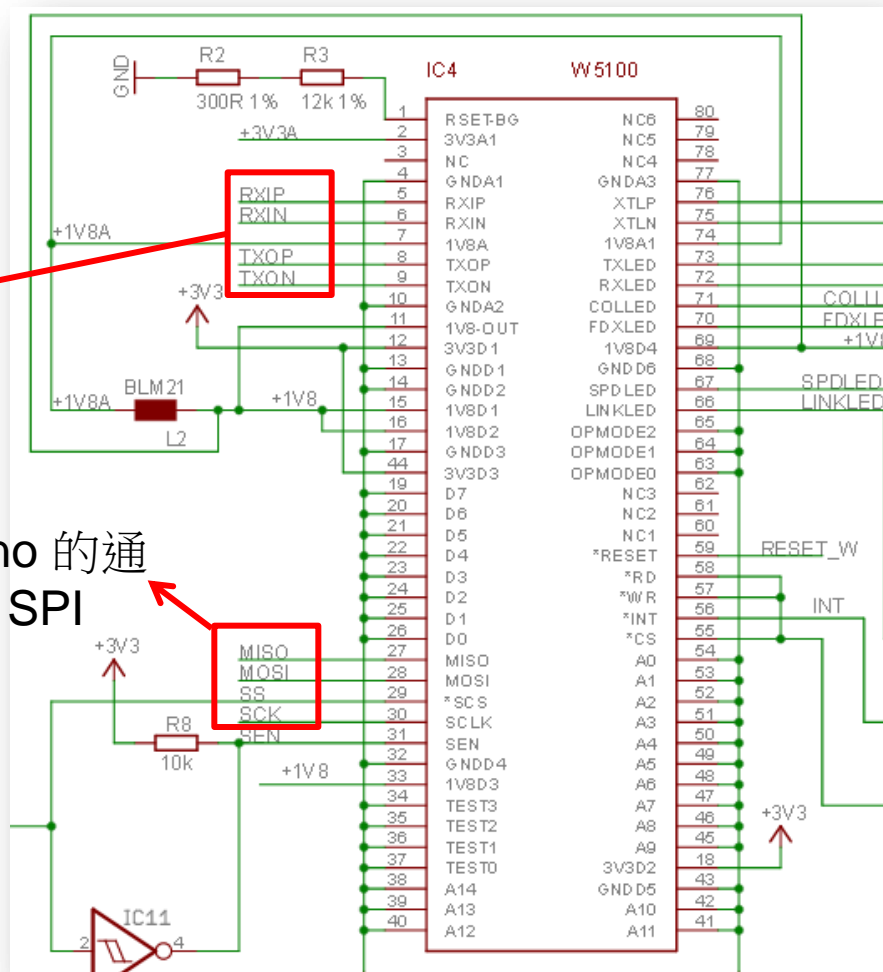
WIZnet W5100 datasheet :

http://www.wiznet.co.kr/Upload_Files/ReferenceFiles/W5100_Datasheet_v1.2.2.pdf

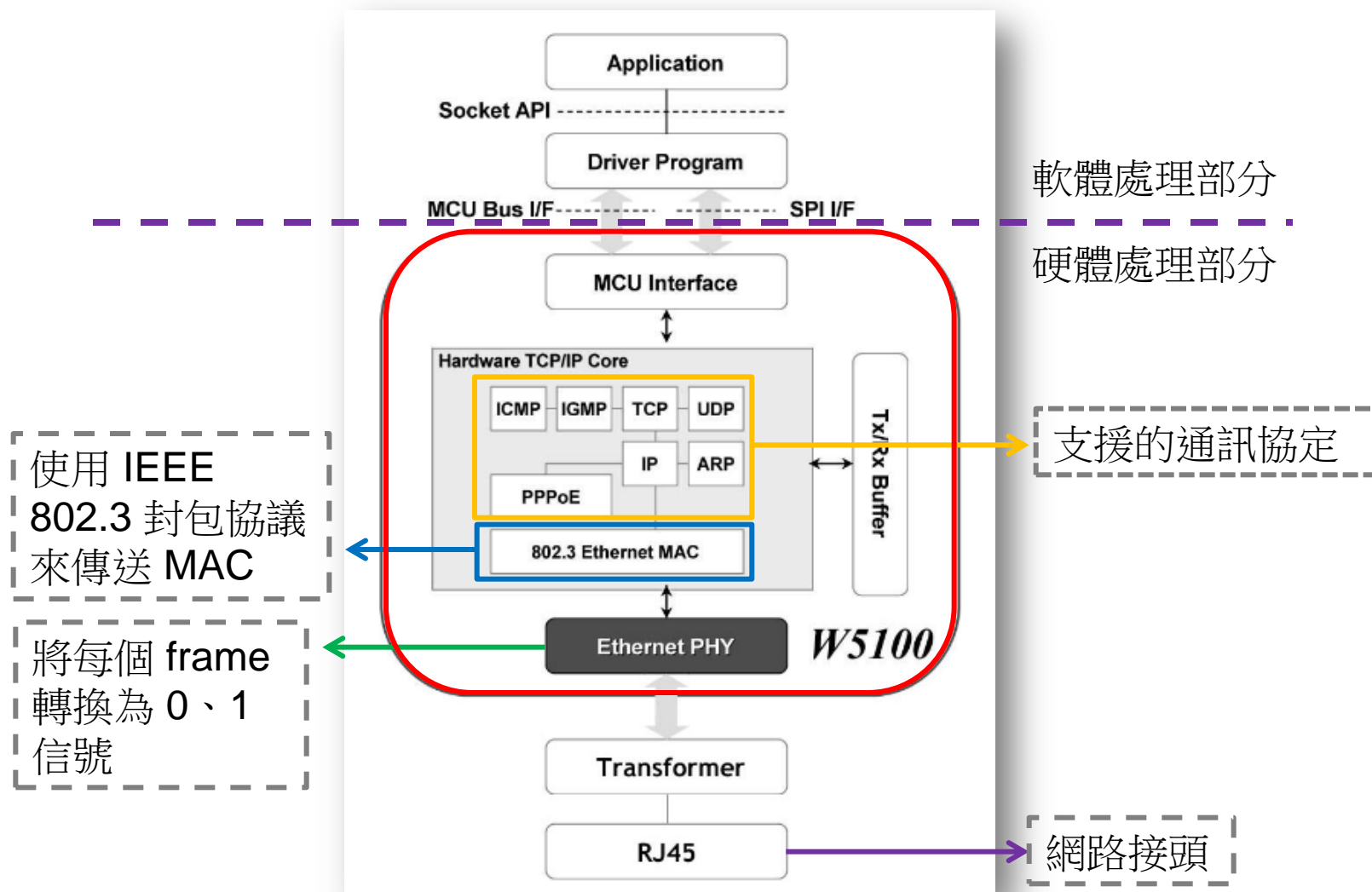
Arduino Ethernet shield 電路

輸出：LAN 的
差動信號

對 Arduino 的通
訊介面：SPI



Arduino Ethernet shield 工作原理



Ethernet 使用範例：ChatServer

- 把 Arduino 當作 WebServer，等待 client 連線

```
byte mac[] = {  
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };  
IPAddress ip(192,168,1, 177);  
IPAddress dnsserver(192,168,1, 1);  
IPAddress gateway(192,168,1, 1);  
IPAddress subnet(255, 255, 0, 0);
```

設定網路卡實體位址、IP位址、子網路遮罩和閘道器位址

```
EthernetServer server(23);  
boolean alreadyConnected = false;
```

設定伺服器的 port 為 23，預設使用 Telnet 服務

```
void setup() {  
  Ethernet.begin(mac, ip, dnsserver, gateway, subnet);  
  server.begin();  
  Serial.begin(9600);  
  while (!Serial) {  
    ; // wait for serial port to connect. Needed for Leonardo only  
  }  
  Serial.print("Chat server address:");  
  Serial.println(Ethernet.localIP());  
}
```

啟動乙太網路連線

啟動伺服器

ChatServer : loop()

```
void loop() {  
  // wait for a new client:  
  EthernetClient client = server.available();  
  
  // when the client sends the first byte, say hello:  
  if (client) {  
    if (!alreadyConnected) {  
      // clear out the input buffer:  
      client.flush();  
      Serial.println("We have a new client");  
      client.println("Hello, client!");  
      alreadyConnected = true;  
    }  
  
    if (client.available() > 0) {  
      // read the bytes incoming from the client:  
      char thisChar = client.read();  
      // echo the bytes back to the client:  
      server.write(thisChar);  
      // echo the bytes to the server as well:  
      Serial.write(thisChar);  
    }  
  }  
}
```

聆聽用戶的連線請求

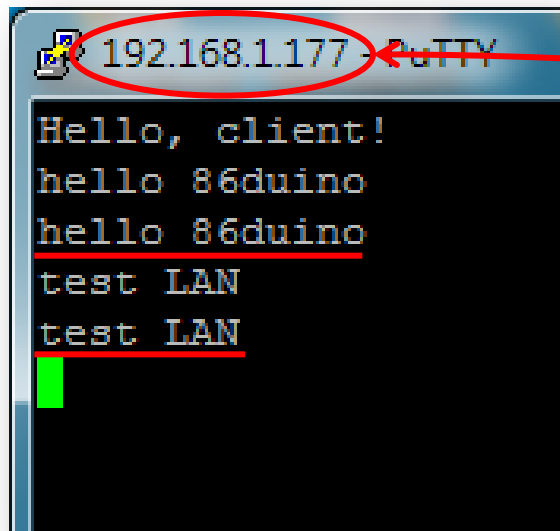
如果收到用戶的連線請求

第一個用戶連線，送出Hello 字串

檢查用戶是否有送字元過來，如果有，則返回相同的字元給所有已連結上的用戶

ChatServer 執行結果

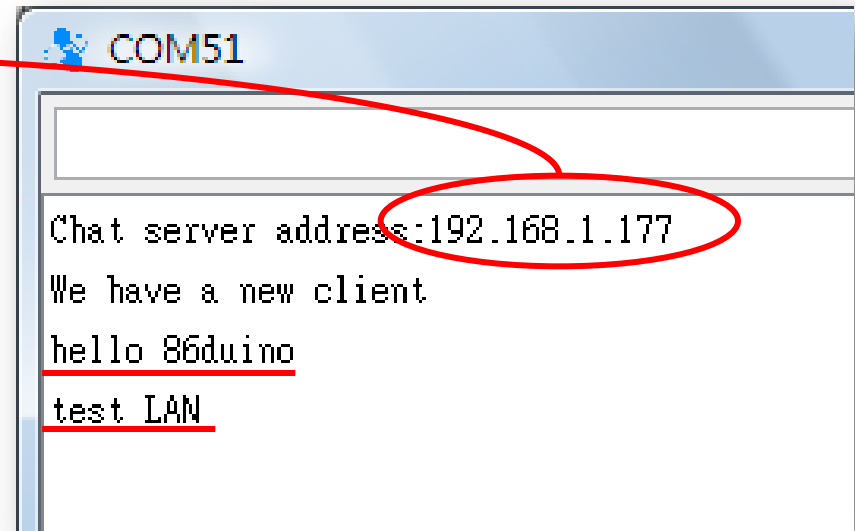
PC : putty.exe



```
192.168.1.177 - PuTTY
Hello, client!
hello 86duino
hello 86duino
test LAN
test LAN
```

Client 端

86Duino : Serial Monitor




```
COM51
Chat server address:192.168.1.177
We have a new client
hello 86duino
test LAN
```

Server 端



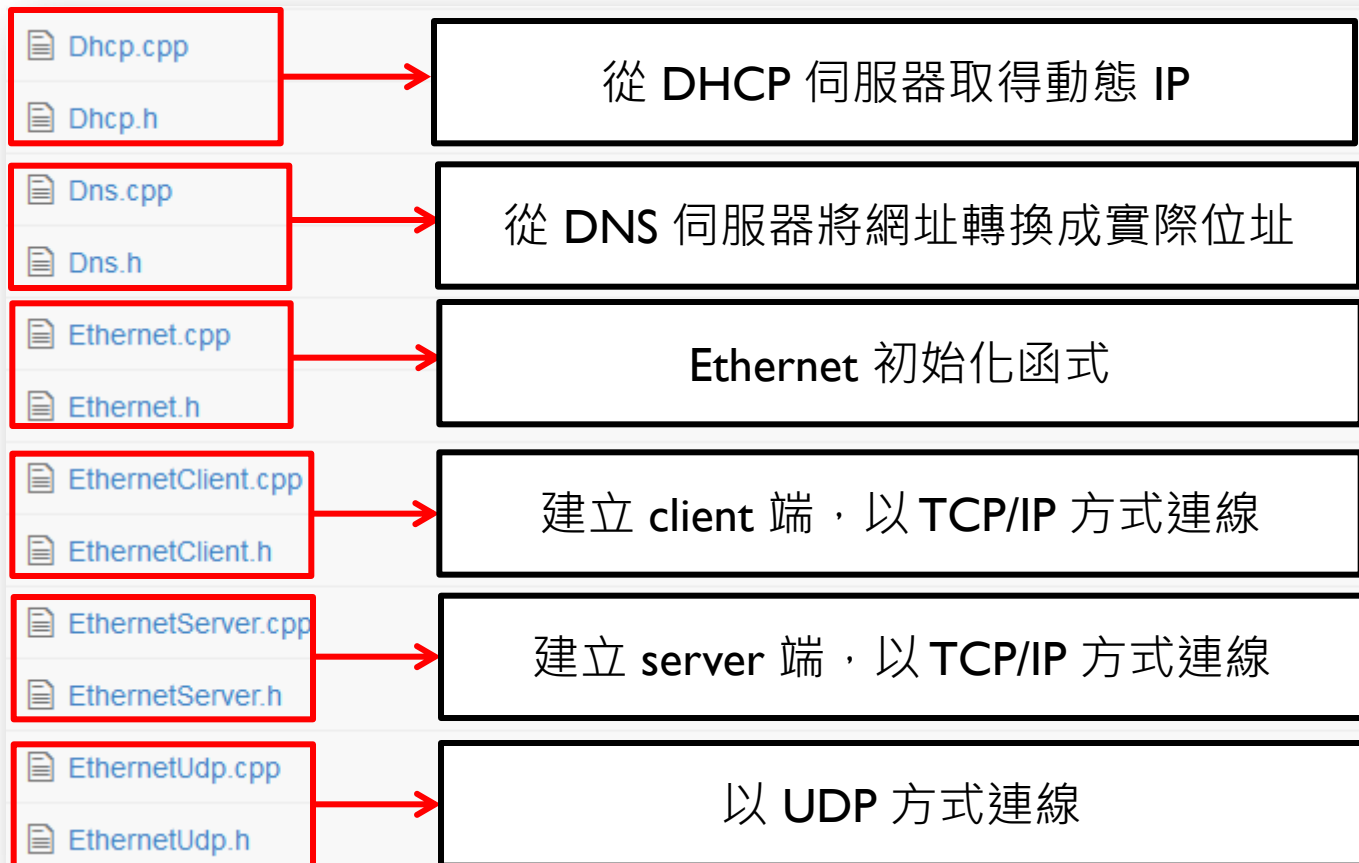
Ethernet Library 原始碼概觀



- 
- 由於 **Ethernet Library** 相當龐大，這裡不深入討論上面每個 **.cpp** 裡的函式細節
 - 只討論和移植有關的大架構

Ethernet 資料夾

<https://github.com/arduino/Arduino/tree/master/libraries/Ethernet>



Ethernet / utility 資料夾

<https://github.com/arduino/Arduino/tree/master/libraries/Ethernet/utility>

socket.cpp

socket.h



Arduino 提供的底層 API
(部分遵循 BSD Socket 標準，但沒有完全相容)

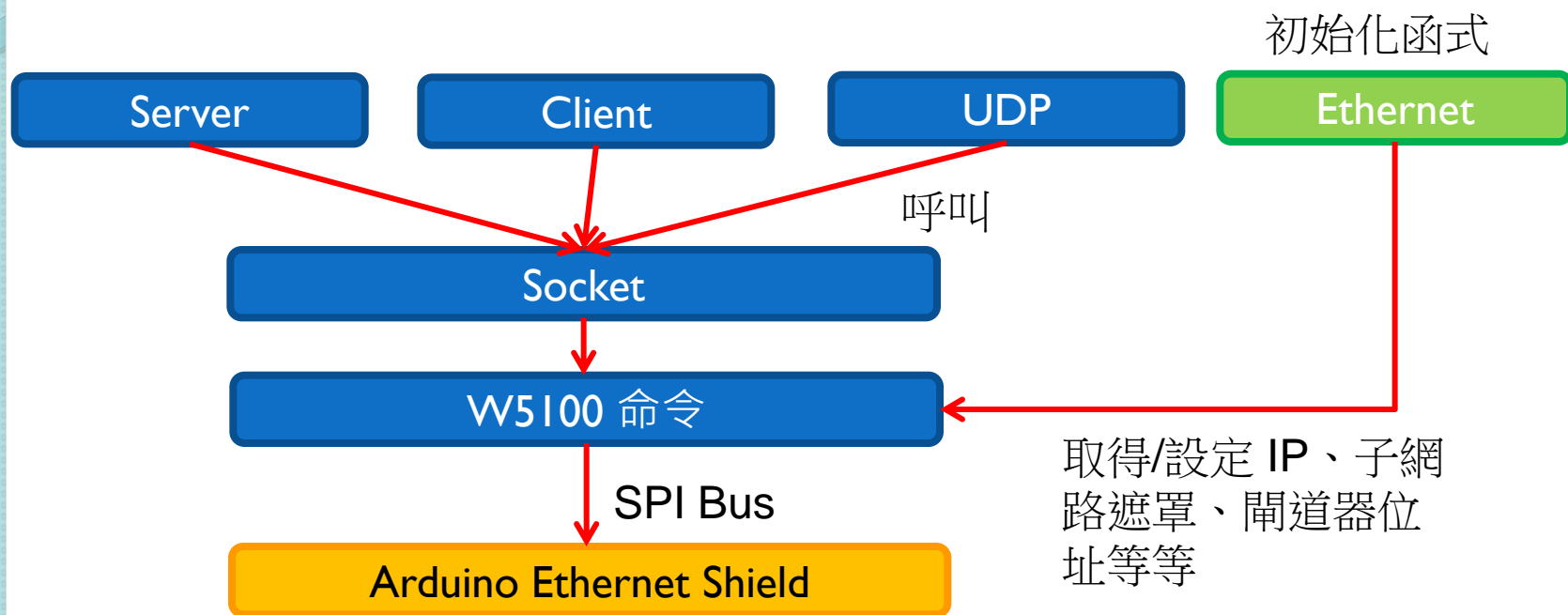
w5100.cpp

w5100.h



透過 SPI 介面將命令送給 W5100 晶片

Arduino Ethernet Library 的架構



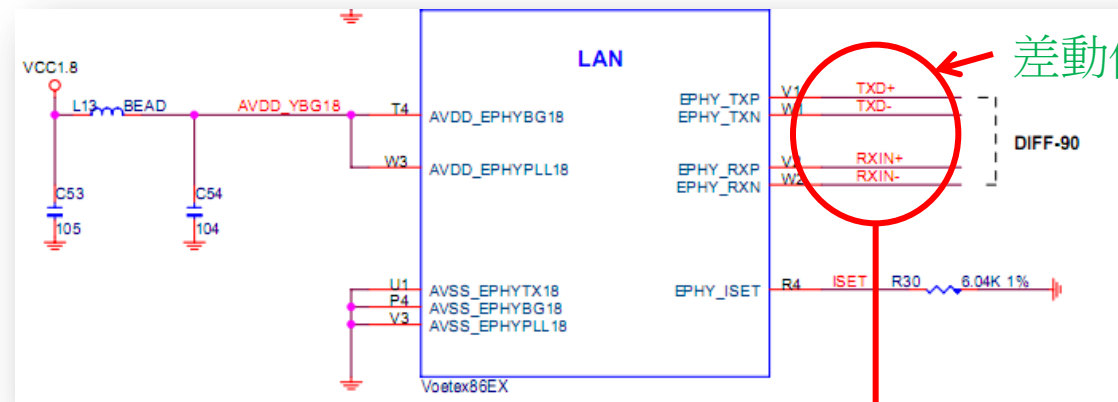
The background of the slide features a blue 86Duino circuit board. The board is populated with various electronic components, including a central microcontroller, memory chips, and connectors. A large, semi-transparent watermark of a padlock with the word "86DUINO" inside is centered over the board. The text "Ethernet Library" is written in a large, blue, sans-serif font, and "在 86Duino 上的移植" is written in a slightly smaller, blue, sans-serif font below it. The entire text is centered horizontally and partially overlaid by the padlock watermark.

Ethernet Library 在 86Duino 上的移植

86Duino 內建的 Ethernet 電路

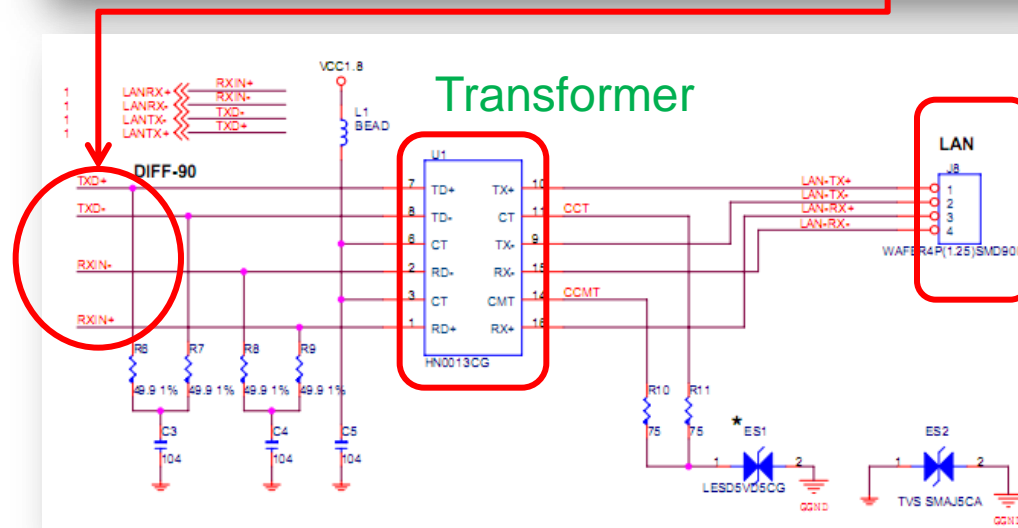
- 網路卡內建在 CPU 中 (含 PHY)

Vortex86EX
CPU LAN
(PCI 裝置)



差動信號輸出

86Duino
底板



Transformer

網路接頭

DOS 下的網路驅動程式

- 內建在 86Duino 韌體裡的 NDIS driver

R6040	2013/12/25 上午 10...	檔案資料夾
dis_pkt.dos	2007/8/29 下午 03:48	DOS 檔案
NETBIND.COM	1994/8/31 上午 12:00	MS-DOS 應用程式
PROTMAN.DOS	1994/8/31 上午 12:00	DOS 檔案
PROTMAN.EXE	1994/8/31 上午 12:00	應用程式

- NDIS : Network Driver Interface Specification
， 參考資料：
 - http://en.wikipedia.org/wiki/Network_Driver_Interface_Specification

DOS 下的 Socket Library

- BSD Socket 規範：
<http://web.mit.edu/macdev/Development/MITSupportLib/SocketsLib/Documentation/sockets.html>
- DOS 下常用的 Socket Library：
 - Watt32：<http://www.watt-32.net/>
 - SwsSock (86Duino 使用的 Socket Library)：
<http://www.softsystem.co.uk/products/swssock.htm>
 - ...

Ethernet Library 移植方式

86Duino Ethernet Library

沿用大部分 Arduino 的 API 名稱，但內容全部用標準 Socket API (SwsSock library 提供) 來實作

移植

Arduino Ethernet Library

Server

Ethernet

Client

UDP

Socket

虛擬層 (NDIS to Package)

NDIS driver

86Duino LAN

替換

W5100 命令

Ethernet Shield

移植後的差異： 以 Ethernet Server : begin() 為例

86Duino

```
void EthernetServer::begin()
{
    int idx;
    struct sockaddr_in sin;
    u_long lArg = 1;
    int yes = 1, no = 0;

    _sock = socket(AF_INET, SOCK_STREAM, 0);
    if (_sock == INVALID_SOCKET) {
        shutdown(_sock, SD_BOTH);
        closesocket(_sock);
        return;
    }

    bzero(&sin, sizeof(sin));
    sin.sin_family = AF_INET;
    sin.sin_addr.s_addr = SWSOCK.getULLocalIp();
    sin.sin_port = htons(_port);

    if (bind(_sock, (struct sockaddr *)&sin, sizeof(sin)) != 0) {
        shutdown(_sock, SD_BOTH);
        closesocket(_sock);
        _sock = INVALID_SOCKET;
        return;
    }
}
```

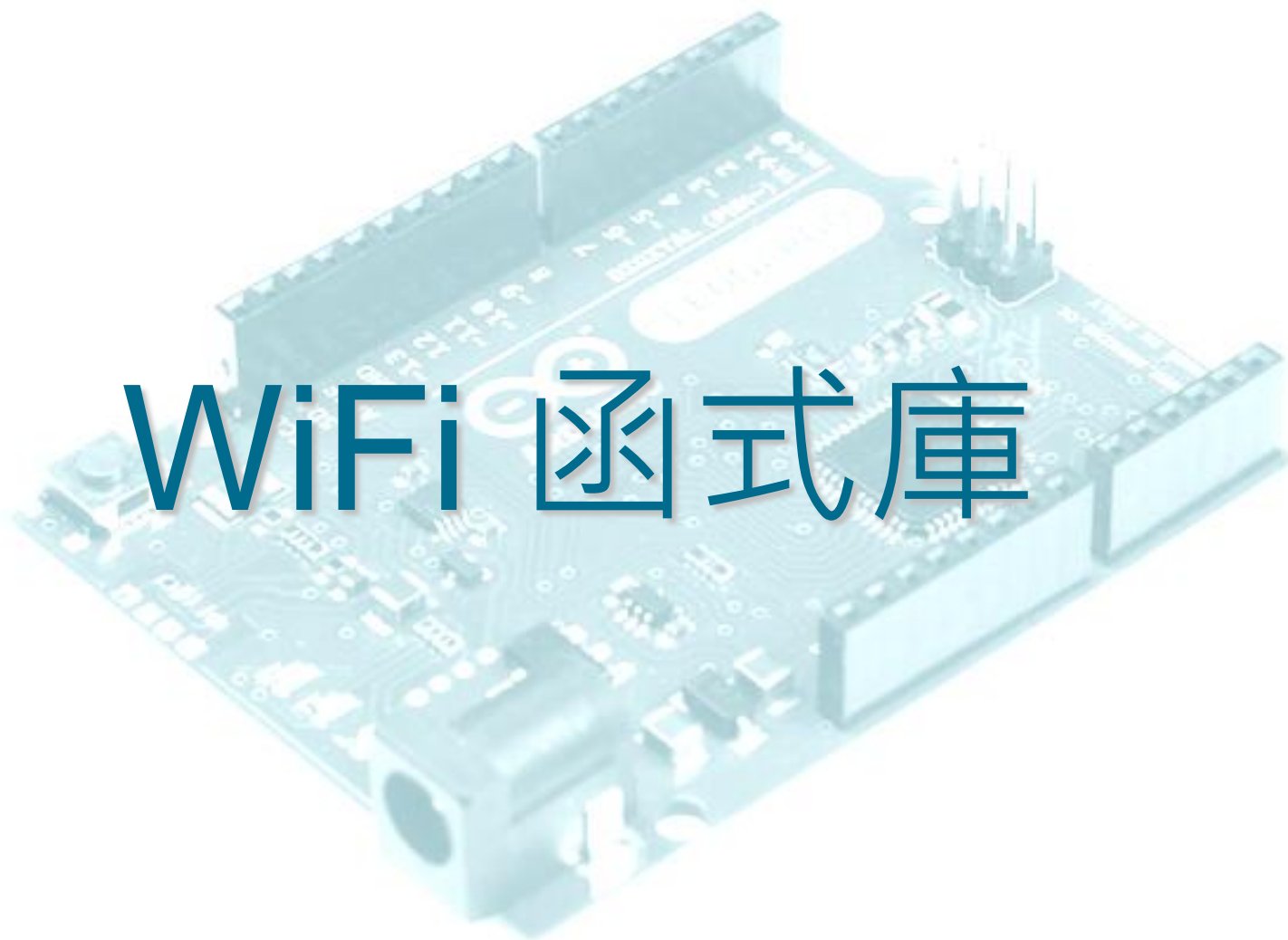
呼叫 Socket API



Arduino

```
void EthernetServer::begin()
{
    for (int sock = 0; sock < MAX_SOCK_NUM; sock++) {
        EthernetClient client(sock);
        if (client.status() == SnSR::CLOSED) {
            socket(sock, SnMR::TCP, _port, 0);
            listen(sock);
            EthernetClass::_server_port[sock] = _port;
            break;
        }
    }
}
```

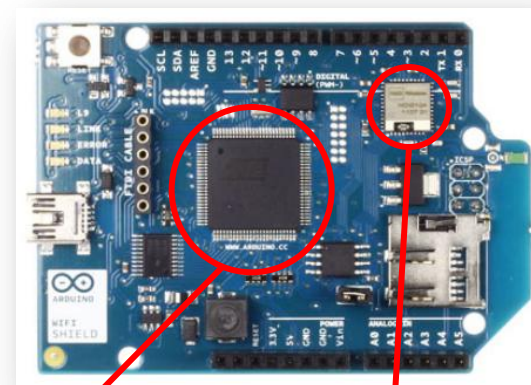

WiFi 函式庫



WiFi shield 簡介

- 工作電壓：3.3V
- 控制晶片：AT32UC3A1256
- 無線模組：HDG104
- 無線網路通訊標準：802.11/g
- 資料加密方式：WEP 或 WPA2
- 與 Arduino 的通訊介面：SPI

WiFi shield 外觀



AT32UC3A1256



HDG104

AT32UC3A1256 datasheet :

http://www.gaw.ru/pdf/Atmel/AVR_32/AT32UC3A0512_0256_0128_1512_1256_1128s.pdf

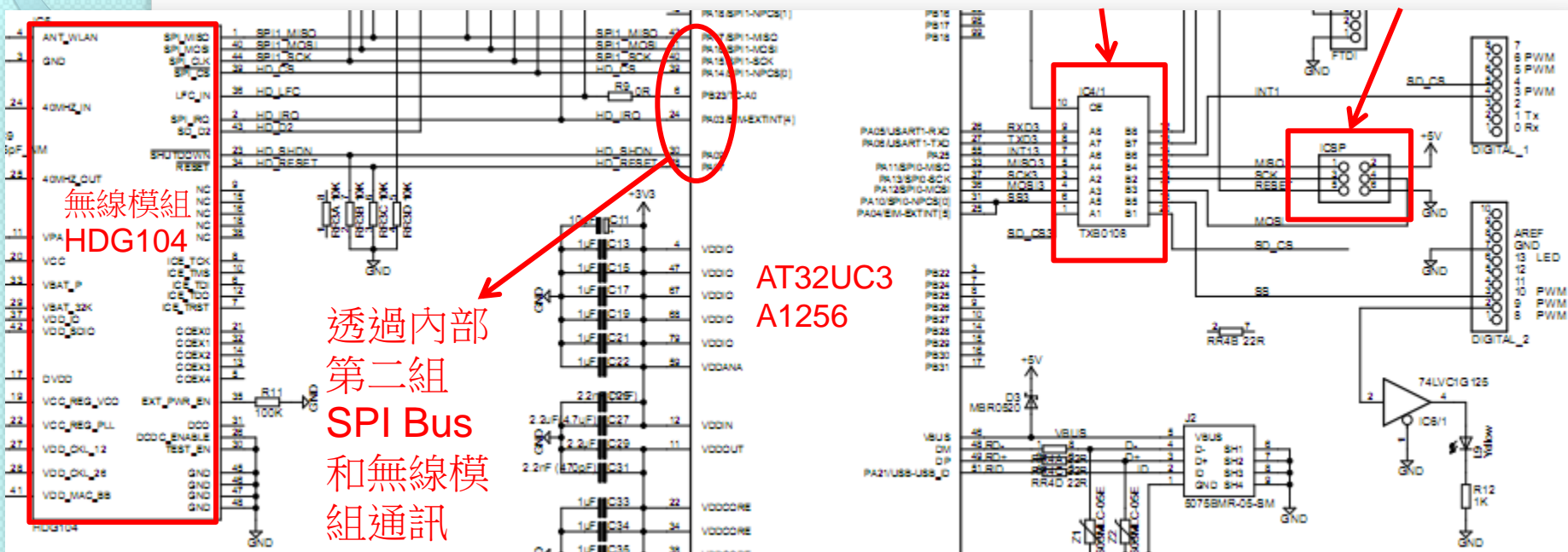
HDG104 datasheet :

<http://datasheet.octopart.com/HDG104-DN-3-H%26D-Wireless-datasheet-11793609.pdf>

Arduino WiFi shield 電路

電壓準位從
5V 轉成 3.3V

對 Arduino 的通訊介
面：SPI



WiFi 使用範例：WifiCharSever

```
#include <SPI.h>
#include <WiFi.h>

char ssid[] = "yourNetwork";
char pass[] = "secretPassword";

int keyIndex = 0;          // your network key Index number (needed only for WEP)
int status = WL_IDLE_STATUS;
WiFiServer server(23);
boolean alreadyConnected = false;

void setup() {
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }

  if (WiFi.status() == WL_NO_SHIELD) {
    Serial.println("WiFi shield not present");
    // don't continue:
    while(true);
  }
}
```

輸入你的 SSID

輸入 WEP/WAP2 形式的密碼

檢查 WiFi shield 是否存在

WifiCharSever : setup()

```
while ( status != WL_CONNECTED ) { ← 假如尚未連線成功
    Serial.print("Attempting to connect to SSID: ");
    Serial.println(ssid);
    // Connect to WPA/WPA2 network. Change this line if using open or WEP network:
    status = WiFi.begin(ssid, pass); ← begin() 會不斷嘗試連線

    // wait 10 seconds for connection:
    delay(10000);
}
// start the server:
server.begin(); ← 連線成功後，啟動伺服器
// you're connected now, so print out the status:
printWifiStatus(); ← 印出目前的 SSID，取得的 IP，以及訊號強度
```

印出目前的 SSID，取得的 IP，以及訊號強度

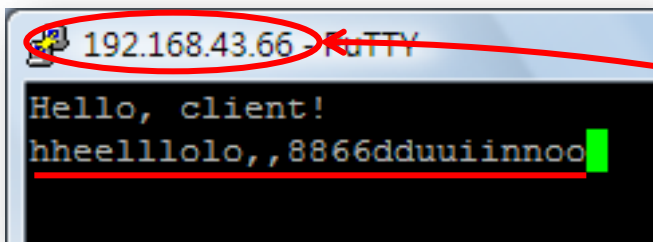
WifiCharSever : loop()

```
void loop() {  
  // wait for a new client:  
  WiFiClient client = server.available();  
  
  // when the client sends the first byte, say hello:  
  if (client) { ← 聆聽用戶的連線請求  
    if (!alreadyConnected) {  
      // clead out the input buffer:  
      client.flush();  
      Serial.println("We have a new client");  
      client.println("Hello, client!"); ← 第一個用戶連線，送出 Hello 字串  
      alreadyConnected = true;  
    }  
  
    if (client.available() > 0) {  
      // read the bytes incoming from the client:  
      char thisChar = client.read();  
      // echo the bytes back to the client:  
      server.write(thisChar);  
      // echo the bytes to the server as well:  
      Serial.write(thisChar);  
    }  
  }  
}
```

檢查用戶是否有送字元過來，如果有，則返回相同的字元給所有已連結上的用戶

WifiChatServer 執行結果

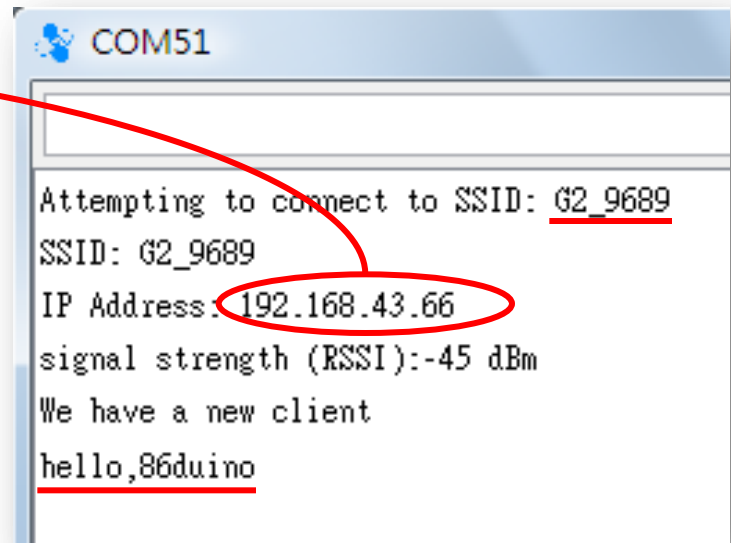
PC : putty.exe



```
192.168.43.66 - PuTTY
Hello, client!
hheelllolo,,8866dduuiinnoo
```

Client 端

86Duino : Serial Monitor



```
COM51
Attempting to connect to SSID: G2_9689
SSID: G2_9689
IP Address: 192.168.43.66
signal strength (RSSI):-45 dBm
We have a new client
hello,86duino
```

Server 端

WiFi 原始碼概觀

<https://github.com/arduino/Arduino/tree/master/libraries/WiFi>

WiFi.cpp	Added copyright licens
WiFi.h	Added copyright licens
WiFiClient.cpp	Added copyright licens
WiFiClient.h	Added copyright licens
WiFiServer.cpp	Added copyright licens
WiFiServer.h	Added copyright licens
WiFiUdp.cpp	Added copyright licens
WiFiUdp.h	Added copyright licens
keywords.txt	updated config() metho

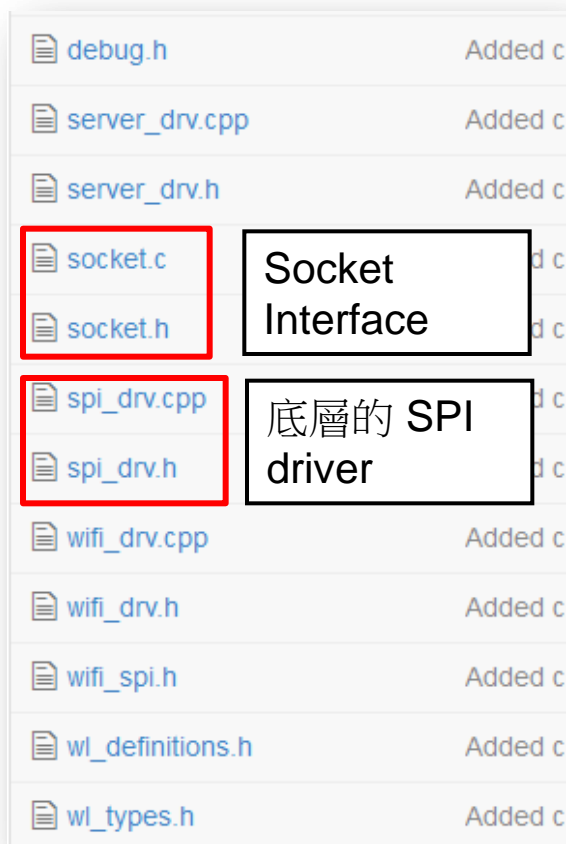
功能皆類似
Ethernet Library

Ethernet

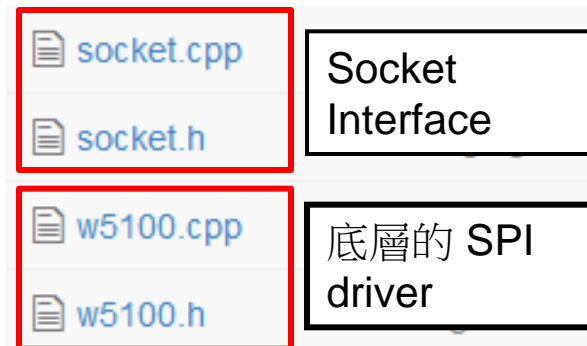
Dhcp.cpp	* Fixed memory
Dhcp.h	Adding Ethernet
Dns.cpp	Fixing warnings
Dns.h	Fixing warnings
Ethernet.cpp	* Fixed memory
Ethernet.h	Adding Ethernet
EthernetClient.cpp	remove all Char
EthernetClient.h	update Advance
EthernetServer.cpp	Fixing warnings
EthernetServer.h	Making Print::wr
EthernetUdp.cpp	Fixing (maybe) a
EthernetUdp.h	Fixing (maybe) a

WiFi 原始碼概觀

<https://github.com/arduino/Arduino/tree/master/libraries/WiFi/utility>



Ethernet/utility



WiFi Library 在 86Duino 上的移植

- 移植重點：WiFi Library 與硬體相關的只有 SPI 的部分，其它部分程式碼與硬體無關，可以在 86Duino 上直接延用
 - 將 spi_drv.cpp 存取方式換成 86Duino SPI 的存取方式
 - 上層的 wifi_drv、server_drv、WIFIClient 等等都不用動

移植後的 spi_drv.cpp 差異

86Duino

```
void SpiDrv::begin()
{
    pinMode(SCK, OUTPUT);
    pinMode(MOSI, OUTPUT);
    pinMode(SS, OUTPUT);
    pinMode(SLAVESELECT, OUTPUT);
    pinMode(SLAVEREADY, INPUT);
    pinMode(WIFILED, OUTPUT);

    digitalWrite(SCK, LOW);
    digitalWrite(MOSI, LOW);
    digitalWrite(SS, HIGH);
    digitalWrite(SLAVESELECT, HIGH);
    digitalWrite(WIFILED, LOW);

#ifdef _DEBUG_
    INIT_TRIGGER()
#endif

    wSPI.begin();
}
```

替換

```
char SpiDrv::spiTransfer(volatile char data)
{
    return (char)wSPI.transfer((uint8_t)data);
}
```

替換

Arduino

```
void SpiDrv::begin()
{
    pinMode(SCK, OUTPUT);
    pinMode(MOSI, OUTPUT);
    pinMode(SS, OUTPUT);
    pinMode(SLAVESELECT, OUTPUT);
    pinMode(SLAVEREADY, INPUT);
    pinMode(WIFILED, OUTPUT);

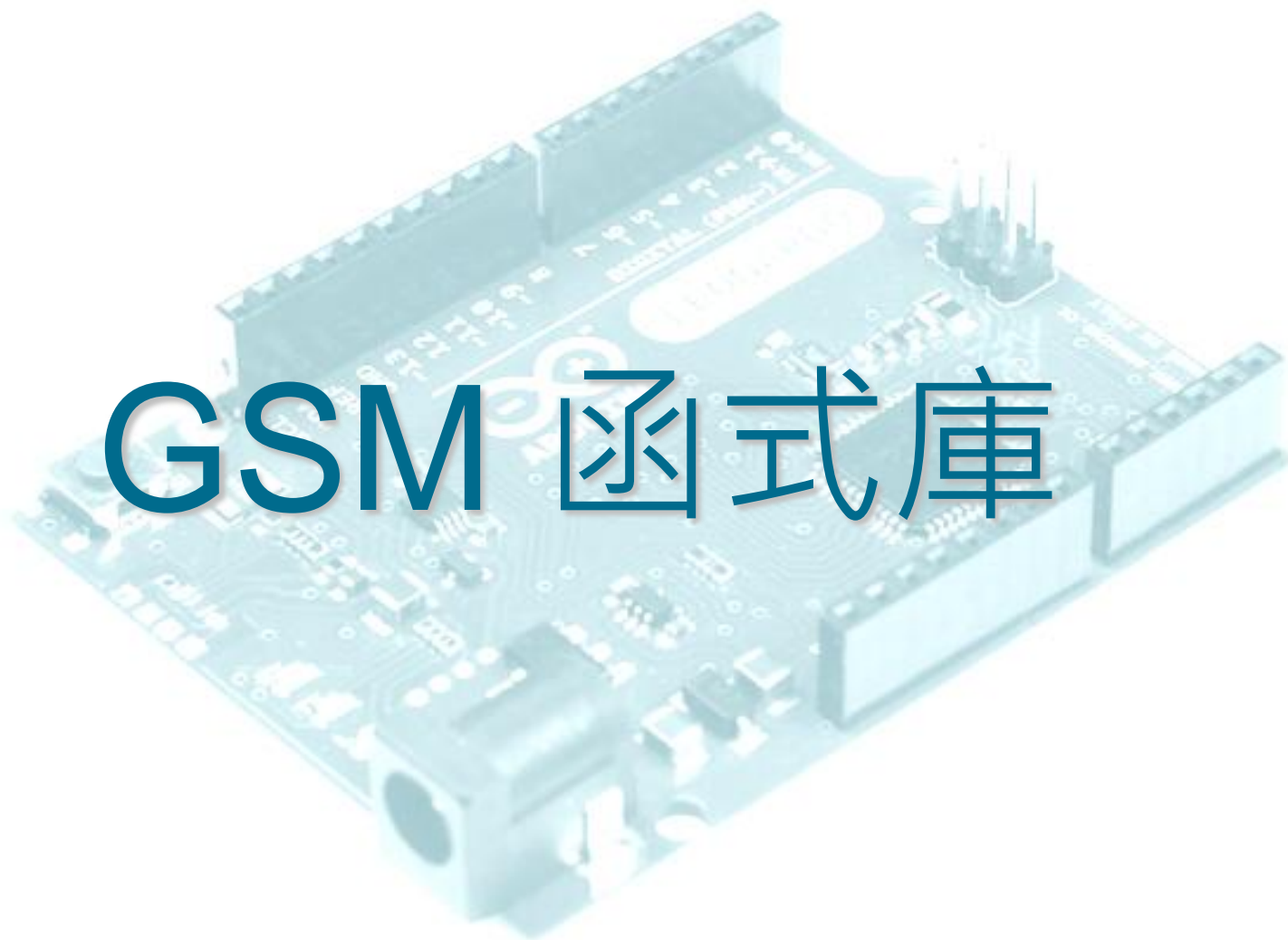
    digitalWrite(SCK, LOW);
    digitalWrite(MOSI, LOW);
    digitalWrite(SS, HIGH);
    digitalWrite(SLAVESELECT, HIGH);
    digitalWrite(WIFILED, LOW);

#ifdef _DEBUG_
    INIT_TRIGGER()
#endif

    SPCR |= _BV(MSTR);
    SPCR |= _BV(SPE);
}
```

```
char SpiDrv::spiTransfer(volatile char data)
{
    SPDR = data; // Start
    while (!(SPSR & (1<<SPIF))) // Wait
    {
    };
    char result = SPDR;
    DELAY_TRANSFER();
    return result; // retu
}
```

GSM 函式庫



GSM 簡介

- GSM 全名 **G**lobal **S**ystem for **M**obile Communications (全球行動通訊系統)
- 在台灣，**GSM** 的頻段是 **900**、**1800MHz**，由於在同一頻段內要讓多人使用又要抗干擾，所以實際上传送數據的速度只達到 **9.6Kbps** (相當於看一個 **200KB** 的網頁要需要等 **20** 幾秒)

GPRS 簡介

- GPRS 全名 **G**eneral **P**acket **R**adio **S**ervice，在現有 **GSM** 技術上，加上數據交換節點（具有處理封包的能力），配合動態分配頻段來增加使用率，在連線人數不多的情況下，速度可達 **56Kbps ~ 100多Kbps**，用來看網頁和圖片，已經綽綽有餘。

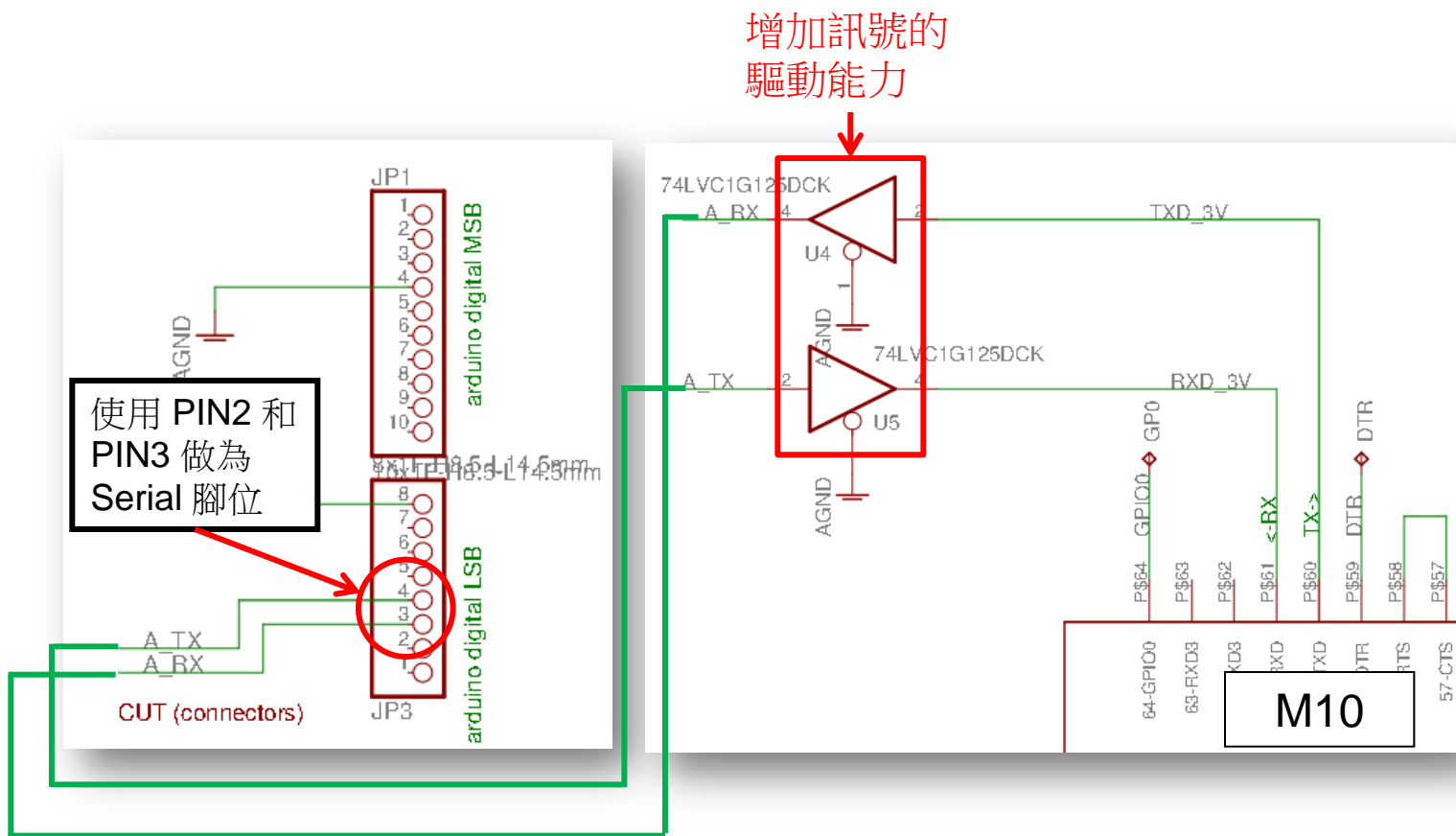
Arduino GSM shield 簡介

- 使用 M10 晶片：
有 GSM + GPRS 功能
- 支援 4 種頻段：
GSM850MHz
GSM900MHz
DCSI800MHz
DCSI900MHz
- 支援 TCP/UDP 和
HTTP 網路通訊協定
上傳和下載的速度
最高可達 85.6Kbps
- Arduino 透過 AT command
控制 GSM shield

GSM shield 外觀



Arduino GSM shield 電路

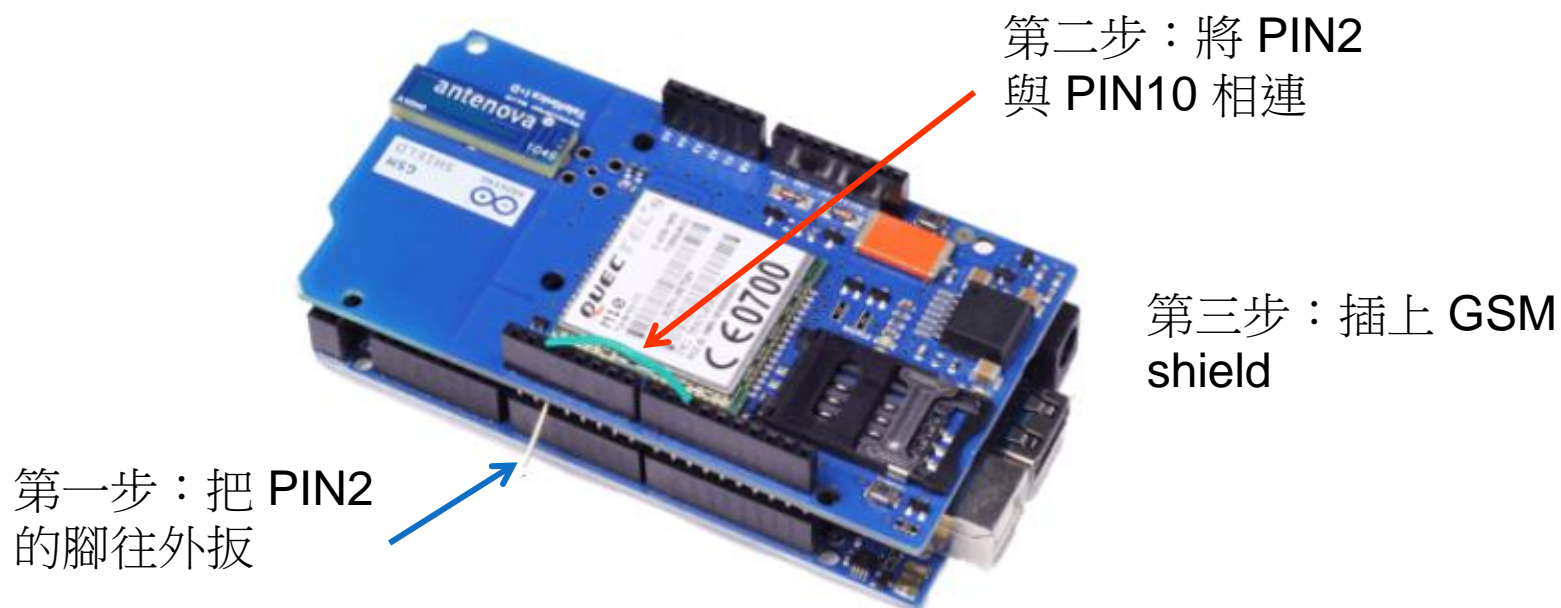


使用 GSM shield 前應注意的事項

- GSM Library 中預設使用 **software** serial 來傳送 AT command：
 - M10 晶片支援的 AT command：
http://arduino.cc/en/uploads/Main/Quectel_M10_AT_commands.pdf
 - GSM shield 使用 PIN2 和 PIN3 做為預設的 software serial 輸出腳位，不使用 hardware serial (PIN0 和 PIN 1) 來傳輸資料，因為這會與 Arduino 燒錄程式的腳位衝突。
- GSM shield 在使用 modem 傳送資料的時候用電量稍大，建議接上外部電源 (700mA ~ 1A)，而不要只使用 USB 供電。
- 需要一張 SIM 卡才能撥打電話、發簡訊以及上網（必須先向電信業者開通上網功能）

使用 GSM shield 前應注意的事項(續)

- 在 Atmega 2560 上，GSM shield 需要額外的跳線：



這是因為具有 toggle trigger 的中斷腳位，在 UNO 上是 PIN2，在 2560 上是 PIN10 的緣故。(在 Leonardo 上則要換成 PIN8)



用 Arduino + GSM shield 打電話和接電話

準備

- 首先，拿一張可用的 SIM 卡，插入 GSM shield

用的是 mini
SIM 卡，手機
用的是 micro
SIM 卡，需要
使用轉卡才不
會掉出來



MakeVoiceCall 範例程式

- 使用者可透過 Serial monitor 輸入電話號碼來撥打給對方

```
#include <GSM.h>
#define PINNUMBER ""
GSM gsmAccess;
GSMVoiceCall vcs;

String remoteNumber = "";
char charbuffer[20];
```

```
void setup()
{
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }
}
```

如果已經在手機上已取消 PIN 碼，這裡就不用輸入 PIN 碼

初始化 GSMAccessProvider class

初始化 GSMVoiceProvider class

MakeVoiceCall : setup()

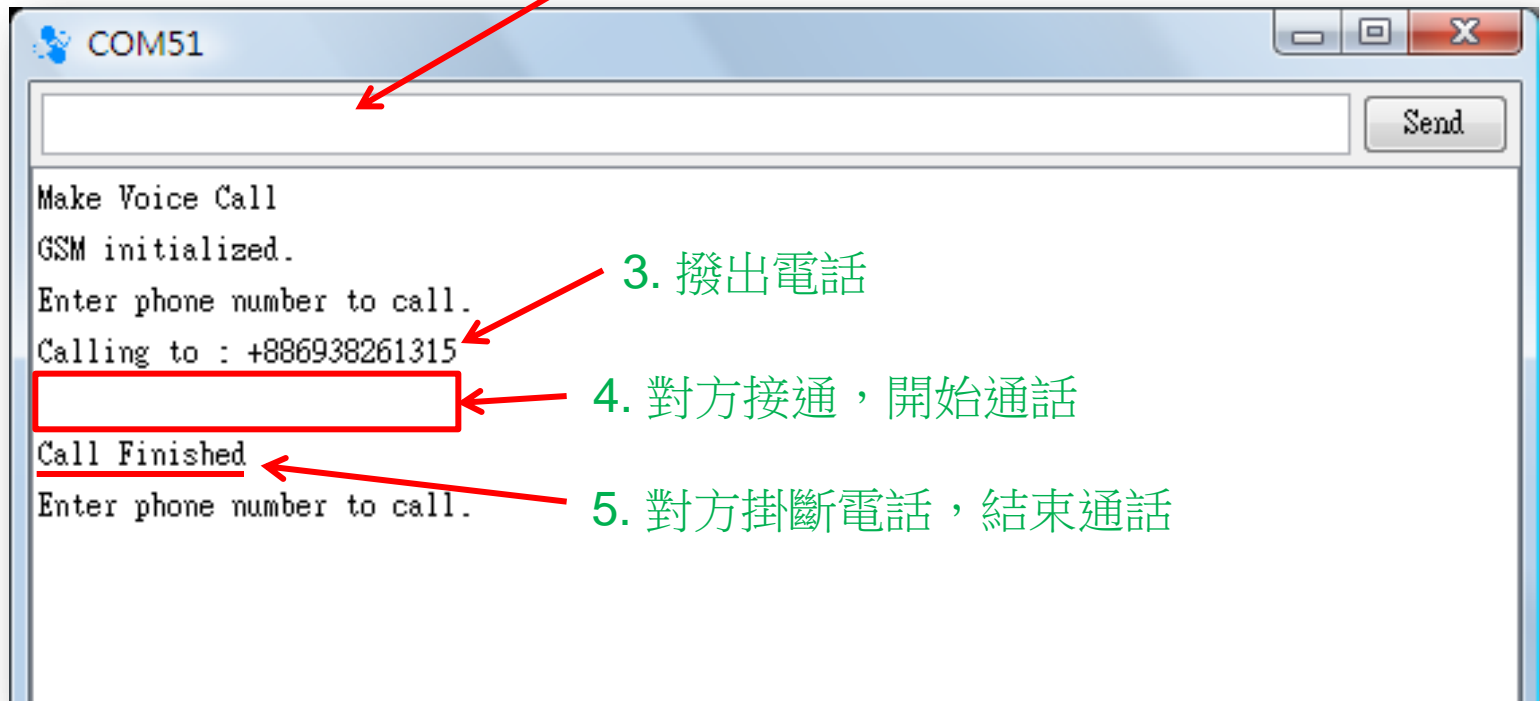
```
...  
// Start GSM shield  
// If your SIM has PIN, pass it as a parameter of begin() in quotes  
while(notConnected)  
{  
  if(gsmAccess.begin(PINNUMBER)==GSM_READY)  
    notConnected = false;  
  else  
  {  
    Serial.println("Not connected");  
    delay(1000);  
  }  
}  
  
Serial.println("GSM initialized.");  
Serial.println("Enter phone number to call.");  
...
```

begin() 完成基本的初始化

回傳結果都沒問題，就可以開始打電話了

上傳 MakeVoiceCall 之後...

1. 打開 serial monitor
2. 輸入要撥打的電話 (前面需要加上台灣區碼：+886)，輸入完後按 Send



3. 撥出電話

4. 對方接通，開始通話

5. 對方掛斷電話，結束通話

用 GSM shield 接電話

- ReceiveVoiceCall 範例程式 (內容與 MakeVoiceCall 類似，所以略過)
- 上傳後打開 serial monitor：

```
COM51  
  
Receive Voice Call  
Waiting for a call  
RECEIVING CALL  
Number:0934351333  
TALKING. Press enter to hang up.  
Hanging up and waiting for the next call.
```

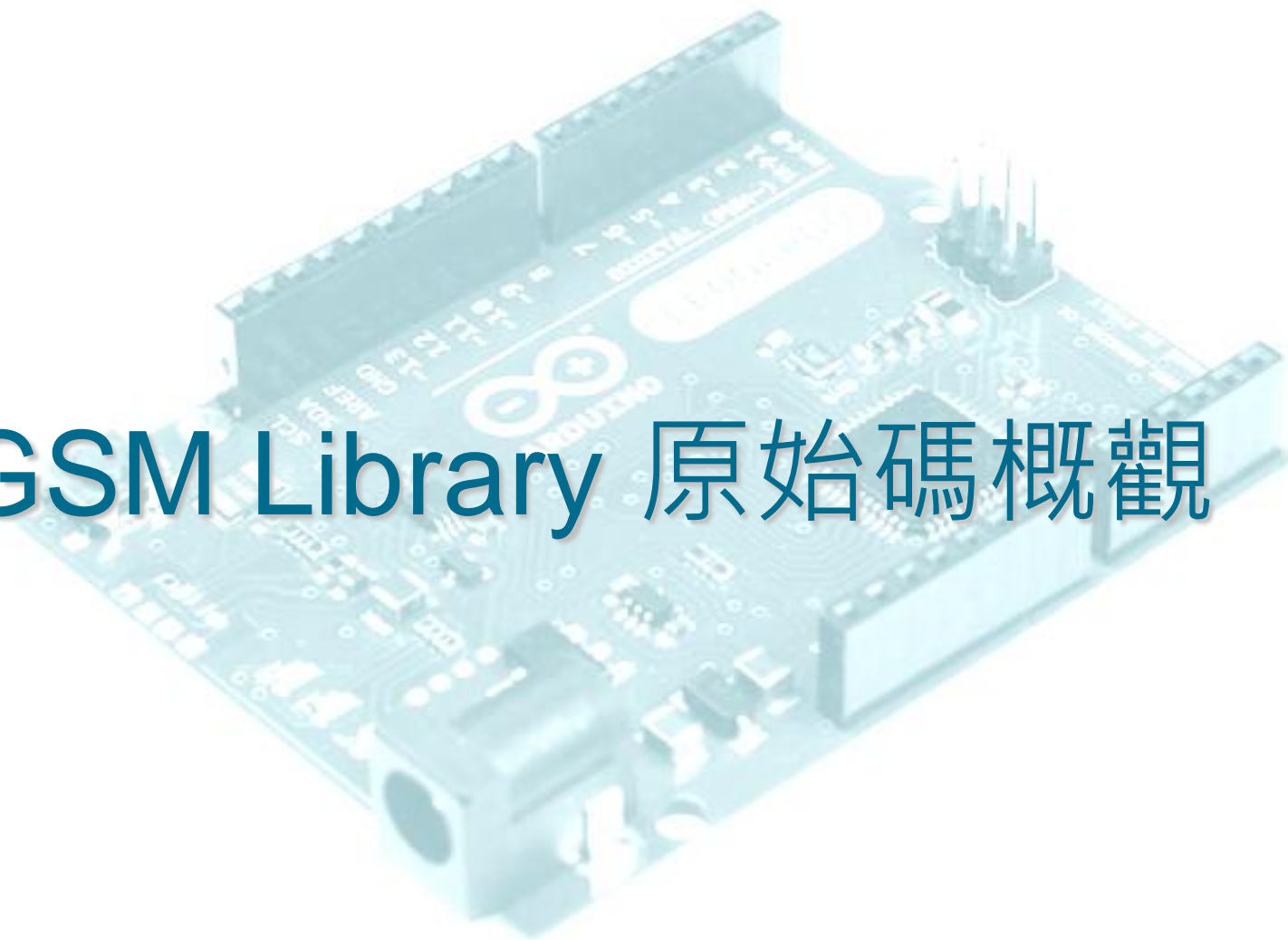
初始化完成，等待別人撥電話進來

接到一通電話，並顯示電話號碼

雙方通話中

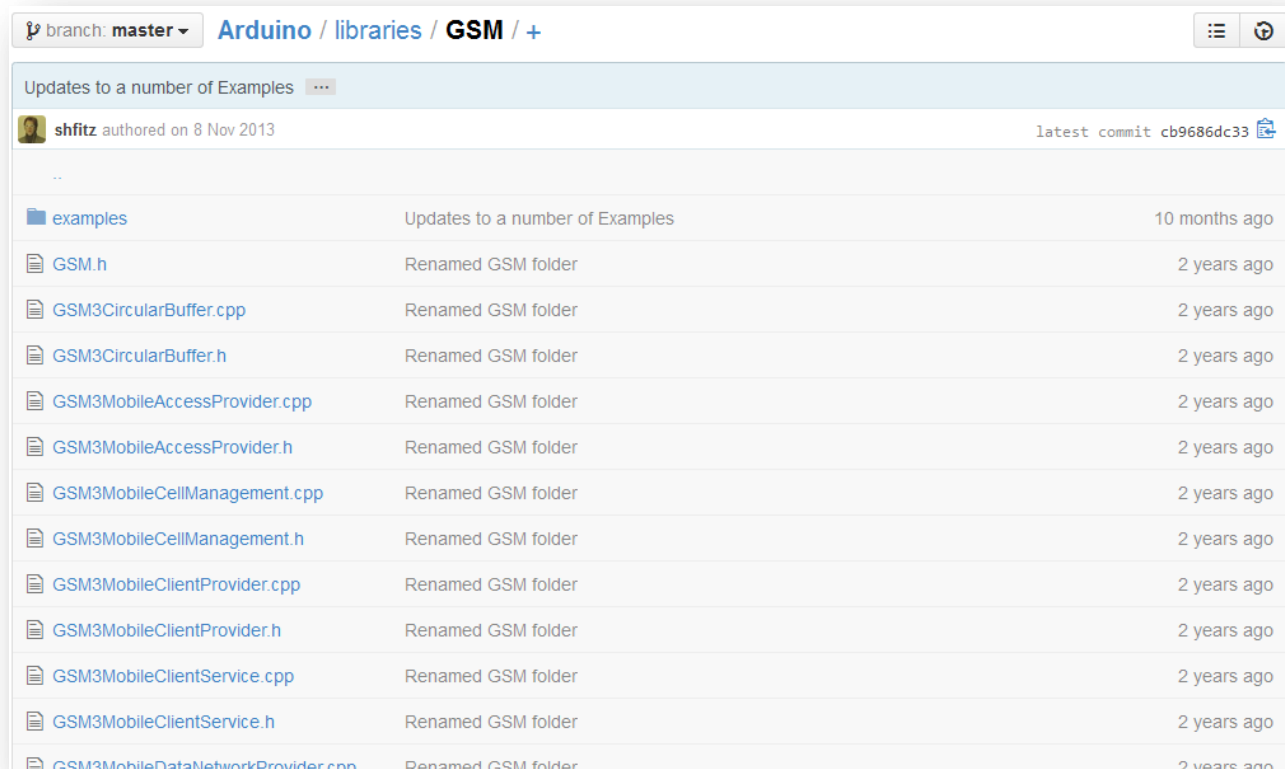
Send '\n' 後結束通話

GSM Library 原始碼概觀



GSM Library

- 到 Arduino 的 GitHub 網站，看看 GSM 資料夾內的檔案：
 - <https://github.com/arduino/Arduino/blob/master/libraries/GSM>



GSM Library 檔案

- GSM Library 裡面的檔案很多，我們可以大致分類如下：

 [GSM3CircularBuffer.cpp](#)

 [GSM3CircularBuffer.h](#)

負責管理 SoftSerial
使用的 buffer

 [GSM3ShieldV1AccessProvider.cpp](#)

 [GSM3ShieldV1AccessProvider.h](#)

負責送出 PIN 碼、偵
測 GPRS 網路等等

 [GSM3ShieldV1VoiceProvider.cpp](#)

 [GSM3ShieldV1VoiceProvider.h](#)


負責撥號和建立語音連
線

GSM Library 檔案 (續)

 GSM3ShieldV1SMSProvider.cpp

 GSM3ShieldV1SMSProvider.h

負責收發簡訊

 GSM3ShieldV1MultiClientProvider.cpp

 GSM3ShieldV1MultiClientProvider.h

 GSM3ShieldV1MultiServerProvider.cpp

 GSM3ShieldV1MultiServerProvider.h

在 GPRS 網路上，建立 server/client 端

 GSM3SoftSerial.cpp

 GSM3SoftSerial.h

以 Software Serial 的方式收送 AT command

GSM Library 檔案 (續)

 GSM3ShieldV1PinManagement.cpp

 GSM3ShieldV1PinManagement.h

PIN 碼管理工具

 GSM3ShieldV1DirectModemProvider.cpp

 GSM3ShieldV1DirectModemProvider.h

 GSM3ShieldV1ModemCore.cpp

 GSM3ShieldV1ModemCore.h

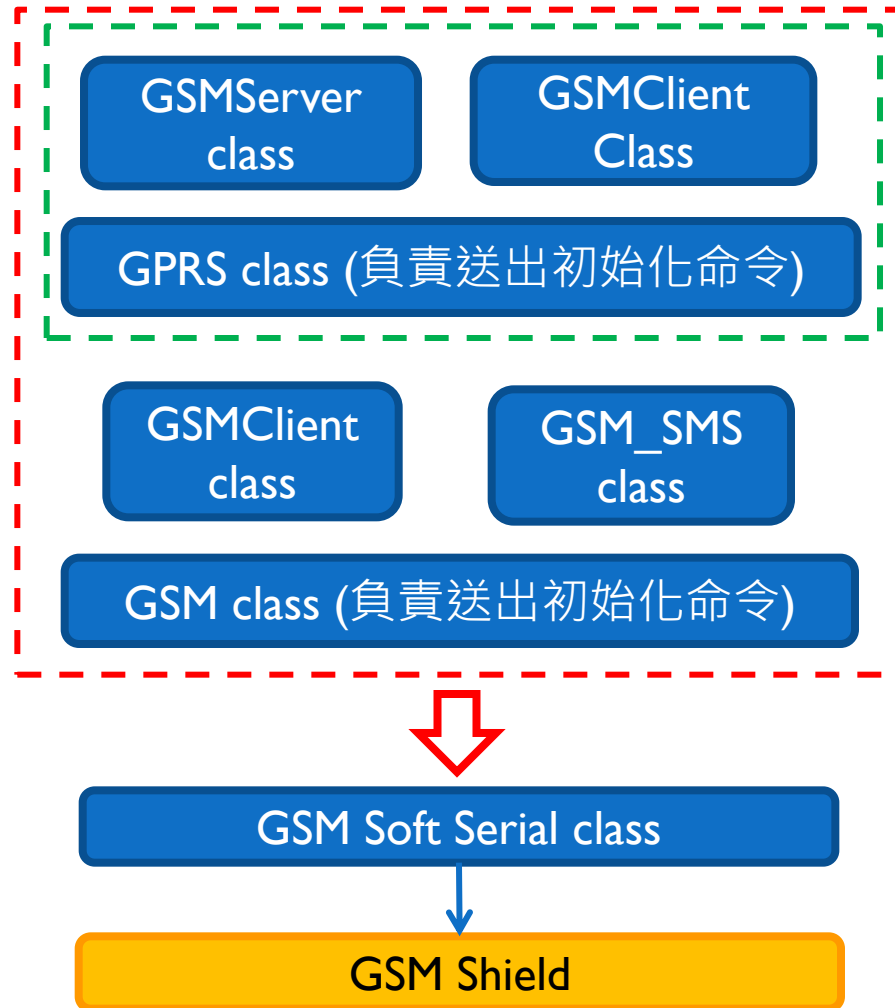
Modem 測試工具

 GSM3ShieldV1ScanNetworks.cpp

 GSM3ShieldV1ScanNetworks.h

GPRS 網路掃描工具

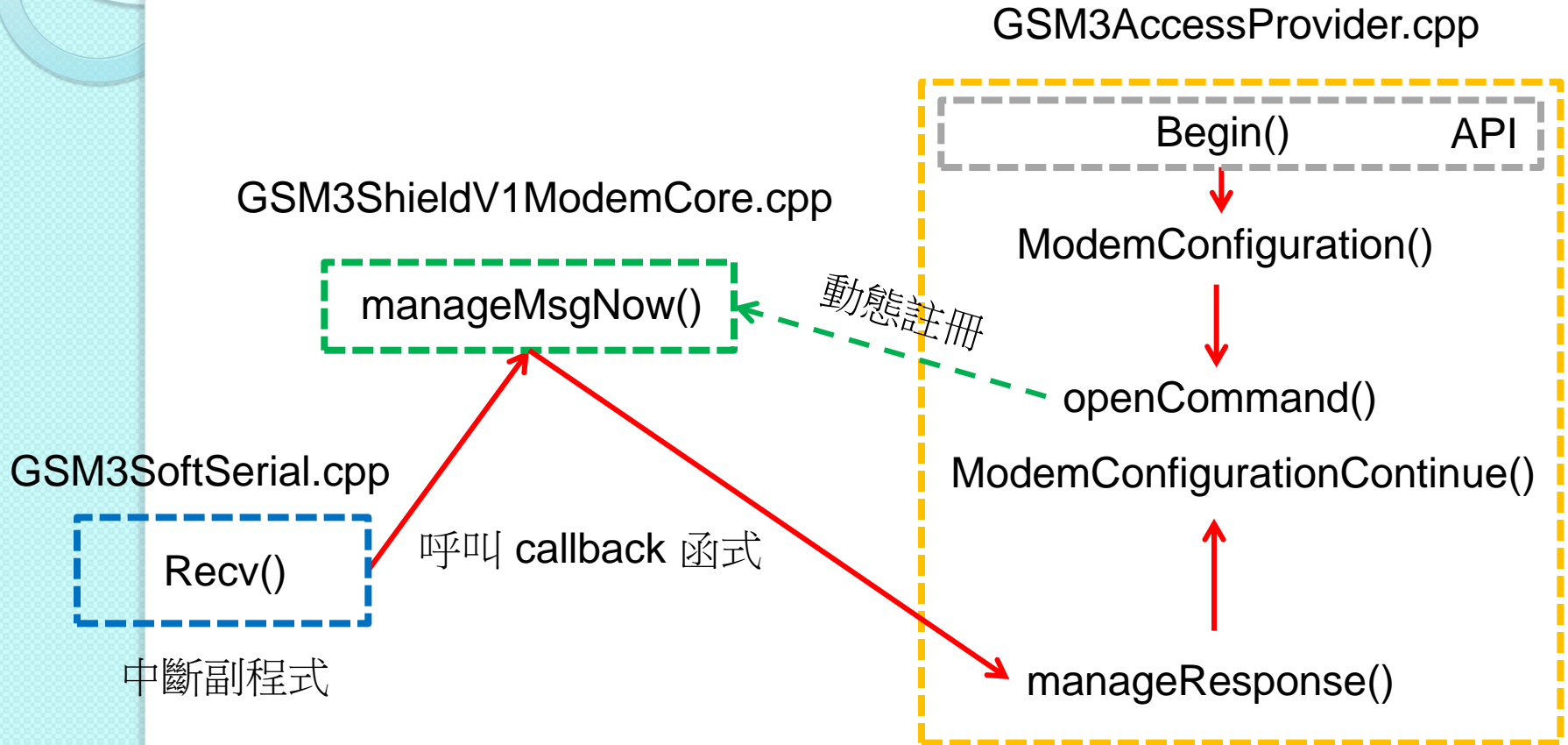
GSM Library 架構



1. 每個 class 包含自己專用的 AT command

2. 當使用者呼叫 GSM API 時，該 API 會把自己的處理程序註冊到一個特殊列表中，讓 SoftSerial 中斷副程式以 callback 的方式來呼叫

GSM Library 運作方式： 以 GSM.begin() 為例



其他功能的運作方式皆大同小異

ModemConfigurationContinue()

```
if(ct==1)
{
    // Launch AT
    theGSM3ShieldV1ModemCore.setCommandCounter(2);
    theGSM3ShieldV1ModemCore.genericCommand_rq("AT");
}
else if(ct==2)
{
    // Wait for AT - OK.
    if(theGSM3ShieldV1ModemCore.genericParse_rsp(resp))
    {
        if(resp)
        {
            // OK received
            if(theGSM3ShieldV1ModemCore.getPhoneNumber() && (theGSM3ShieldV1ModemCore.getPhoneNumber() != ""))
            {
                theGSM3ShieldV1ModemCore.genericCommand_rq("AT+CPIN=", false);
                theGSM3ShieldV1ModemCore.setCommandCounter(3);
                theGSM3ShieldV1ModemCore.genericCommand_rqc(theGSM3ShieldV1ModemCore.getPhoneNumber());
            }
        }
    }
}
```

第一個 AT 指令

檢查回傳結果

第二個 AT 指令

此函式內容可以說是 AT command 的腳本。
它最後會被 SoftSerial 的中斷副程式呼叫。

在其他功能的 .cpp 中，看到以 Continue 結尾的函式，函式內容都差不多，只差在 AT command 不同

```
if(resp)
{
    theGSM3ShieldV1ModemCore.setCommandCounter(4);
    theGSM3ShieldV1ModemCore.takeMilliseconds();
    theGSM3ShieldV1ModemCore.delayInsideInterrupt(2000);
    theGSM3ShieldV1ModemCore.genericCommand_rq("AT+CGREG?");
}
else if(hwcomIsUsed == false) theGSM3ShieldV1ModemCore.closeCommand(3);
}
else if(ct==4)
{
    char auxLocate1 [12];
    char auxLocate2 [12];
    prepareAuxLocate("+CGREG: 0,1", auxLocate1);
    prepareAuxLocate("+CGREG: 0,5", auxLocate2);
    if(theGSM3ShieldV1ModemCore.genericParse_rsp(resp, auxLocate1, auxLocate2))
    {
        if(resp)
        {
            theGSM3ShieldV1ModemCore.setCommandCounter(5);
            theGSM3ShieldV1ModemCore.genericCommand_rq("AT+IFC=1,1");
        }
    }
}
```

檢查回傳結果

第三個指令

檢查回傳結果

第四個指令

GSM Soft Serial 運作方式：recv()

- GSM 以既有的 lib 為基礎，再寫一個新的 SoftSerial

```
do
{
    ...

    for (uint8_t i=0x1; i; i <= 1) {
        tunedDelay(_rx_delay_intrabit);
        uint8_t noti = ~i;
        if (rx_pin_read())
            d |= i;
        else // else clause added to ensure function timing is ~balanced
            d &= noti;

        if(fullbuffer&&!capturado_fullbuffer)) {
            if((uint8_t)__XOFF__ & i)
                tx_pin_write(HIGH);
            else
                tx_pin_write(LOW);
        }

        ...

        if(keepThisChar(&d)) {
            cb.write(d);
            if(firstByte)
            {
                firstByte=false;
                thisHead=cb.getTail();
            }
        }
        ...
    }while(morebytes);
```

用 `digitalRead()` + `delay()` 讀取 8bit 值

假如 buffer 滿了
就送 XOFF 回去
請對方暫緩傳送

檢查是不是特殊字元 'w'

假如不是，就放入 buffer 中

這部分是新增的

GSM Soft Serial 運作方式 (續)

```
}while (morebytes);  
// If we find a line feed, we are at the end of a paragraph  
// check!  
  
if (fullbuffer)  
{  
    // And... go handle it!  
    if (mgr)  
        mgr->manageMsg(thisHead, cb.getTail());  
}  
  
else if (d==10)  
{  
    // And... go handle it!  
    if (mgr)  
        mgr->manageMsg(thisHead, cb.getTail());  
}  
  
else if (d==32)  
{  
    // And... go handle it!  
    if (mgr)  
        mgr->manageMsg(thisHead, cb.getTail());  
}  
}
```

buffer 滿了，則呼叫
callback 處理函式

收到 LF 符號，則呼叫
callback 處理函式

收到 space 符號，則呼叫
callback 處理函式

這部分是
新增的

An 86Duino circuit board, which is a compact single-board computer. It features a central microcontroller, various peripheral components like capacitors and resistors, and multiple connectors including a USB port, a DC power input, and a digital pin header. The board is populated with several integrated circuits, including a large black chip labeled 'SOM28EX' and various smaller chips labeled with codes like 'A1', 'A2', 'A64', 'A63', 'B64', and 'B63'. The text '86DUINO' is printed on the board. The background of the slide is a light blue gradient with a decorative vertical bar on the left side containing overlapping circles and a small blue sphere.

GSM Library 在 86Duino 上的移植

GSM Library 在 86Duino 上的移植

- 移植重點：
GSM Library 與硬體相關的只有
SoftSerial 的部分，其它部分程式碼與硬
體無關，可以在 86Duino 上直接延用

GSM Library 在 86Duino 上的移植

- GSM3SoftSerial.cpp
 - 直接套用 86Duino softserial Library 的實作方式，然後再加入 GSM SoftSerial 新增的那些 code
 - 預設的腳位更動，RX 是用 PIN42 (有 attachInterrupt 功能的)，TX 是用 PIN3
- 小改進：
 - 新增 Hardware Serial 的通訊方式，可以選用 COM1 ~ COM3 其中一組來通訊。
 - 使用 Hardware Serial 可以保證 GSM Library 工作更穩定

加入 Hardware Serial 功能

- 不使用中斷
 - 因為 `SoftSerial` 已經與各項功能的 `callback function` 和軟體 `flow control` 寫在一起，為了實現相同的功能，可能要大改既有的 `Hardware Serial library`。
- 改用 `polling`
 - 經過觀察，主要 `class` 在運行過程中，都會持續呼叫名為 `ready` 的函式，用來得到送出 `AT command` 的回傳結果
 - 因此就把 `hardware serial` 函式加入 `ready()`，實現原來 `SoftSerial` 會做的事

GSM3ShieldV1BaseProvider.cpp : ready()

```
int GSM3ShieldV1BaseProvider::ready()  
{  
    theGSM3ShieldV1ModemCore.manageReceivedData();  
    return theGSM3ShieldV1ModemCore.getCommandError();  
};
```

呼叫

GSM3SoftSerial.cpp : recv()

```
do  
{  
    ...  
    fullbuffer=(cb.availableBytes()<6);  
    // Receive Data and handle soft flow control event  
    ...  
    if(keepThisChar(&d))  
    {  
        cb.write(d);  
        if(firstByte)  
        {  
            firstByte=false;  
            thisHead=cb.getTail();  
        }  
    }  
    ...  
}while (morebytes);  
if (fullbuffer)  
    if(mgr) mgr->manageMsg(thisHead, cb.getTail());  
else if(d==10)  
    if(mgr) mgr->manageMsg(thisHead, cb.getTail());  
else if (d==32)  
    if(mgr) mgr->manageMsg(thisHead, cb.getTail());
```

原來的 Software
Serial code

行為相同

GSM3ShieldV1ModemCore.cpp : manageReceivedData()

```
else if(hwcomIsUsed == true)  
{  
    // simulate the functions in original soft-Serial  
    if(HWSerial[hwcomport]->available() > 0)  
    {  
        fullbuffer = (gss.cb.availableBytes()<6);  
        if(!fullbuffer)  
        {  
            d = HWSerial[hwcomport]->read();  
            if(gss.keepThisChar(&d))  
            {  
                gss.cb.write(d);  
                if(firstByte)  
                {  
                    firstByte=false;  
                    thisHead=gss.cb.getTail();  
                }  
            }  
            if(d==10)  
                manageMsg(thisHead, gss.cb.getTail());  
            else if (d==32)  
                manageMsg(thisHead, gss.cb.getTail());  
        }  
        else  
            manageMsg(thisHead, gss.cb.getTail());  
    }  
}
```

添加的
Hardware
Serial code

小插曲 - 遇到 Arduino GSM Library 的 Bug

GSM3VoiceCallService.cpp

```
#include <GSM3VCS.h>
#include <Arduino.h>

#define GSM3VOICECALLSERVICE_SYNCH 0x01 // 1: synchronous
#define __TOUT__ 10000

GSM3ShieldV1VoiceProvider theGSM3ShieldV1VoiceProvider;

GSM3VoiceCallService::GSM3VoiceCallService(bool synch)
{
    if(synch)
        flags |= GSM3VOICECALLSERVICE_SYNCH;
    theGSM3MobileVoiceProvider->initialize();
}
```

呼叫建構子

```
#include <GSM3SVP.h>
#include <Arduino.h>

extern bool hwcomIsUsed;

GSM3ShieldV1VoiceProvider::GSM3ShieldV1VoiceProvider()
{
    phonelength=0;
    theGSM3MobileVoiceProvider=this;
}

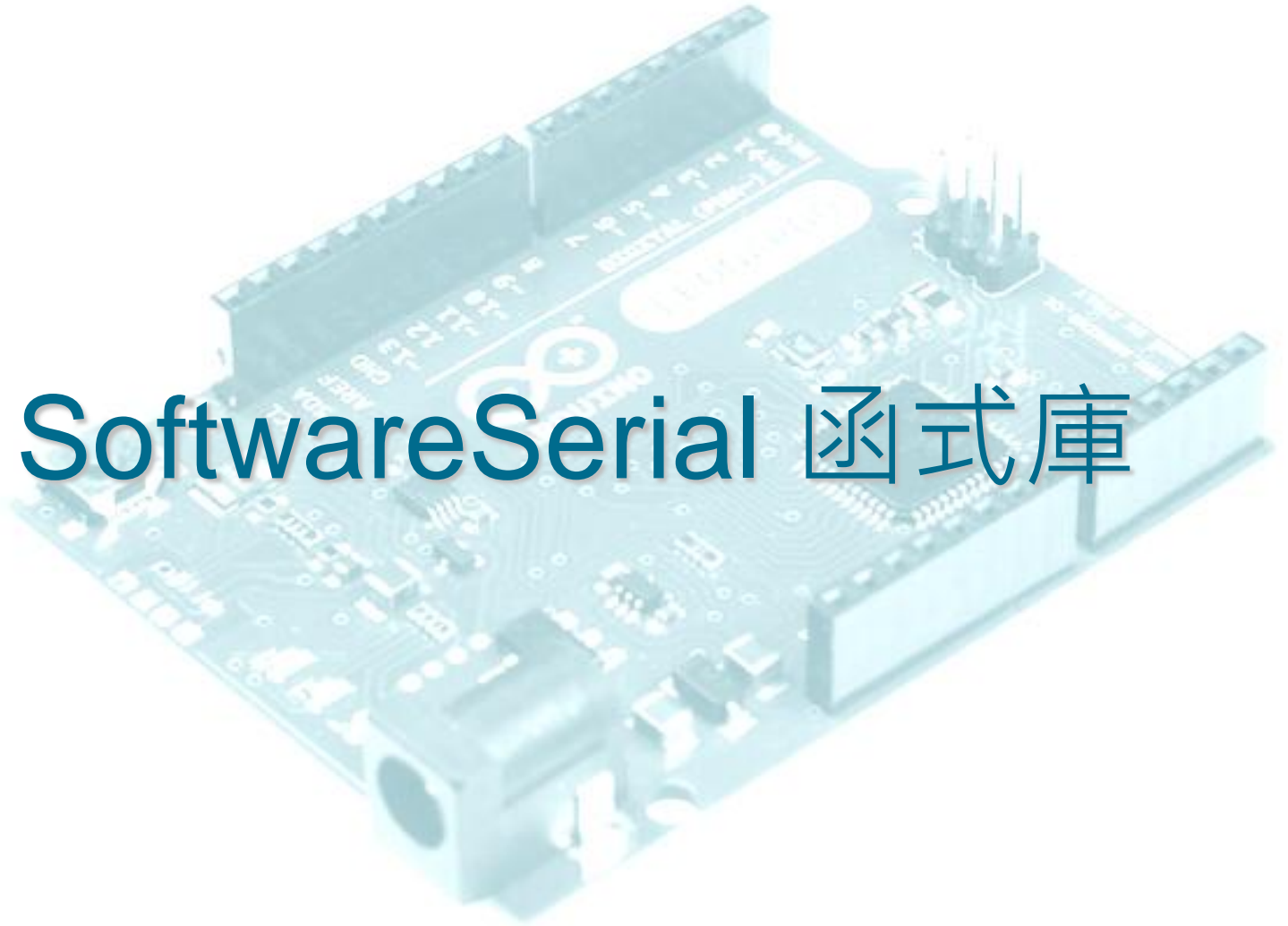
void GSM3ShieldV1VoiceProvider::initialize()
{
    theGSM3ShieldV1ModemCore.registerUMProvider(this);
}
```

GSM3ShieldV1VoiceProvider.cpp

小插曲 - 遇到 Arduino GSM Library 的 Bug

- Bug 來由：
 - 靜態物件 **GSM3VoiceCallService** 的實作方式要求另一個靜態物件 **GSM3MobileVoiceProvider** 先被初始化
 - 但這兩個物件的初始化順序與編譯器有關，不同平台的編譯器會得到不一樣的結果
 - 錯誤的初始化順序會造成 **GSM3VoiceCallService** 使用到尚未給定初值的 **theGSM3MobileVoiceProvider** 變數而當機

SoftwareSerial 函式庫



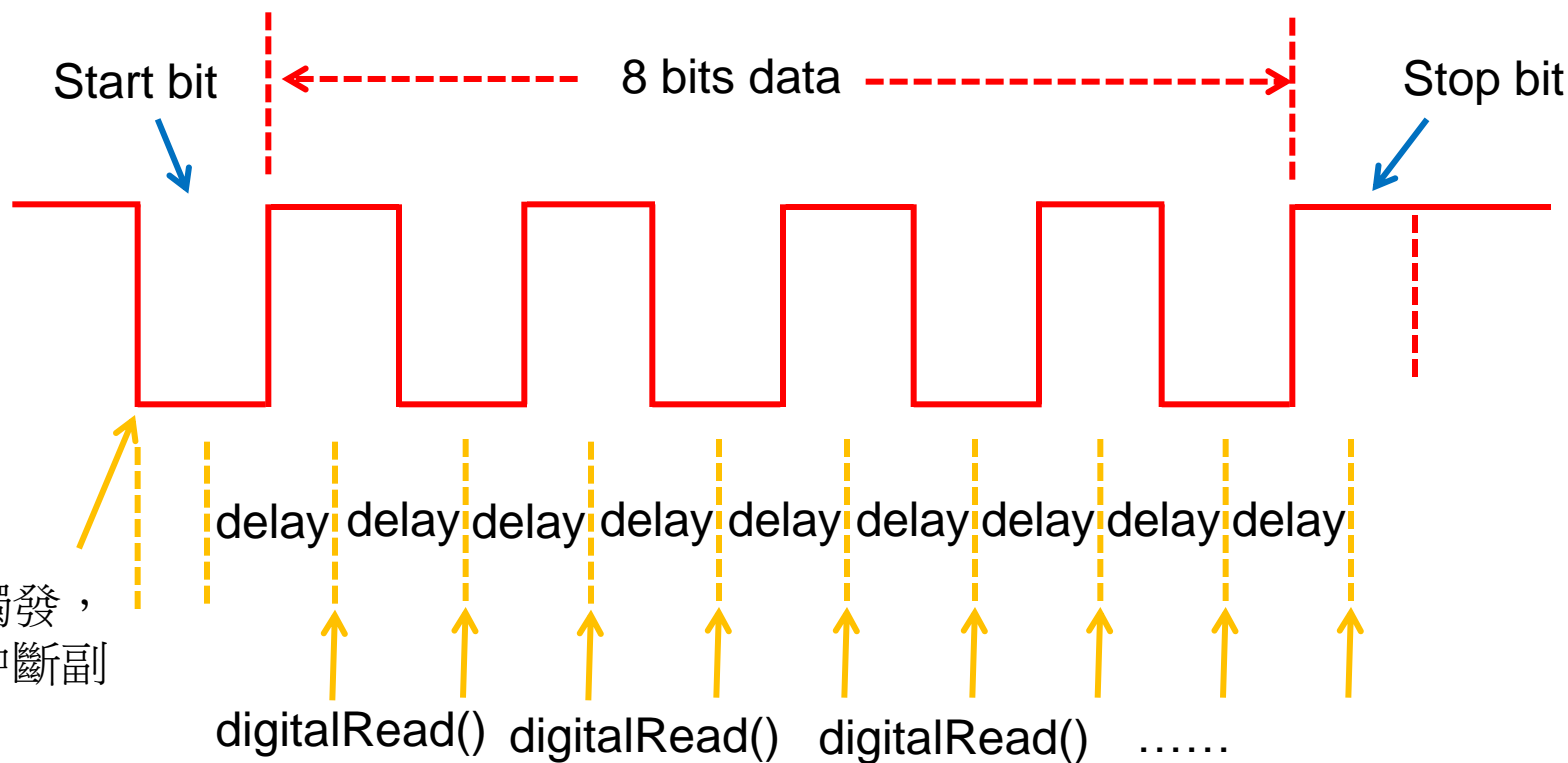
SoftwareSerial 簡介

- 簡單的說，就是用 **GPIO** 來模擬硬體 **UART** 的 **TX**、**RX** 行為
- 在 **Arduino UNO** 上，預設使用 **PIN2** 和 **PIN3** 做為 **RX** 和 **TX**。
- **SoftwareSerial** 函式庫不允許隨意指定腳位，因為 **RX** 的行為需要由具有 **toggle** 中斷的 **PIN** 腳來模擬。

具有 toggle 中斷的 PIN 腳

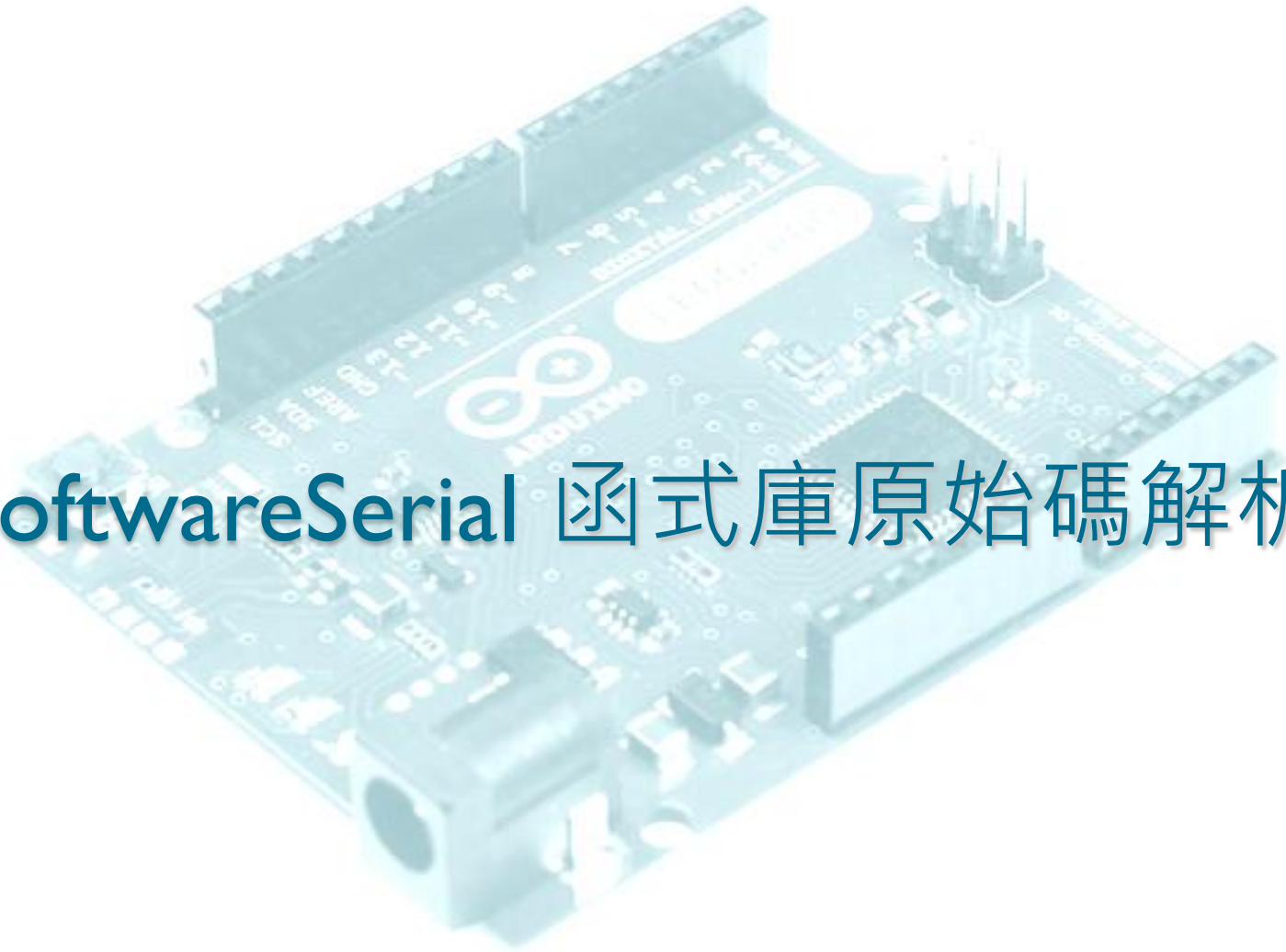
- Arduino UNO
 - PIN2
- Arduino Mega2560 :
 - PIN10 ~ 15 , PIN50 ~ 53 , A8 ~ A15
- Leonardo :
 - PIN8 ~ 11 , PIN14 ~ 16

Arduino 上，RX 行為的模擬方法：





SoftwareSerial 函式庫原始碼解析




SoftwareSerial 函式庫資料夾內容





<https://github.com/arduino/Arduino/tree/master/libraries/SoftwareSerial>

branch: master ▾ **Arduino / libraries / SoftwareSerial / +**

Fix idle level when initializing a inverted SoftwareSerial ...

 **jenscski** authored on 23 May

..

 examples	Updated SoftwareSerial examples so that they work easier with Leonard...
 SoftwareSerial.cpp	Fix idle level when initializing a inverted SoftwareSerial
 SoftwareSerial.h	Reverted begin(long speed) to its original prototype (i.e.
 keywords.txt	Minor typo correction

鮑率延遲時間表

SoftwareSerial.cpp : SoftwareSerial()

```
static const DELAY TABLE PROGMEM table[] = TX 的 delay
{
  // baud      Start bit 的 delay 資料和 Stop bit 的 delay
  // baud      rxcenter  rxintra  rxstop  tx
  { 115200,    1,        17,        17,        12,    },
  { 57600,     10,       37,        37,        33,    },
  { 38400,     25,       57,        57,        54,    },
  { 31250,     31,       70,        70,        68,    },
  { 28800,     34,       77,        77,        74,    },
  { 19200,     54,      117,       117,       114,    },
  { 14400,     74,      156,       156,       153,    },
  { 9600,      114,     236,       236,       233,    },
  { 4800,      233,     474,       474,       471,    },
  { 2400,      471,     950,       950,       947,    },
  { 1200,      947,    1902,      1902,      1899,    },
  { 600,      1902,   3804,       3804,      3800,    },
  { 300,      3804,   7617,       7617,      7614,    },
};
```

這些數值在不同的 CPU 都不一樣，
需要實際 tune 過才能確定

SoftwareSerial 函式庫原始碼重要細節

SoftwareSerial.cpp : SoftwareSerial()

```
SoftwareSerial::SoftwareSerial(uint8_t receivePin, uint8_t transmitPin,  
                               初始化數值    bool inverse_logic /* = false */) :  
{  
    _rx_delay_centering(0),  
    _rx_delay_intrabit(0),  
    _rx_delay_stopbit(0),  
    _tx_delay(0),  
    _buffer_overflow(false),  
    _inverse_logic(inverse_logic)  
    設定 Tx, Rx 腳位  
    setTX(transmitPin);  
    setRX(receivePin);  
}
```

SoftwareSerial 函式庫原始碼重要細節

SoftwareSerial.cpp : begin()

```
void SoftwareSerial::begin(long speed)
{
    _rx_delay_centering = _rx_delay_intrabit = _rx_delay_stopbit = _tx_delay = 0;

    for (unsigned i=0; i<sizeof(table)/sizeof(table[0]); ++i)
    {
        long baud = pgm_read_dword(&table[i].baud);
        if (baud == speed)
        {
            _rx_delay_centering = pgm_read_word(&table[i].rx_delay_centering);
            _rx_delay_intrabit = pgm_read_word(&table[i].rx_delay_intrabit);
            _rx_delay_stopbit = pgm_read_word(&table[i].rx_delay_stopbit);
            tx_delay = pgm_read_word(&table[i].tx_delay);
            break;
        }
    }
    // Set up RX interrupts, but only if we have a valid RX baud rate
    if (_rx_delay_stopbit)
    {
        pinMode(_receivePin, INPUT_PULLUP);
        tunedDelay(_tx_delay); // if we were low this establishes the end
    }
    listen();
}
```

設定鮑率

設定傳輸, 接收 延遲時間

listen(); 等待中斷發生 開始接收資料

SoftwareSerial 函式庫原始碼重要細節

SoftwareSerial.cpp : ISR()

```
#if defined(PCINT0_vect)
ISR(PCINT0_vect) ← 等待接受訊號觸發中斷
{
    SoftwareSerial::handle_interrupt();
}
#endif
```

SoftwareSerial.cpp : handle_interrupt()

```
inline void SoftwareSerial::handle_interrupt()
{
    if (active_object)
    {
        active_object->recv(); ← 觸發中斷執行 recv()
    }
}
```

SoftwareSerial 函式庫原始碼重要細節

SoftwareSerial.cpp : available()

```
int SoftwareSerial::available()
{
    if (!isListening())
        return 0;

    return (receive_buffer_tail + SS_MAX_RX_BUFF - receive_buffer_head) % SS_MAX_RX_BUFF;
```

回傳 buffer 資料數量

SoftwareSerial.cpp : read()

```
int SoftwareSerial::read()
{
    if (!isListening())
        return -1;

    // Empty buffer?
    if (_receive_buffer_head == _receive_buffer_tail)
        return -1;

    // Read from "head"
    uint8_t d = receive_buffer[receive_buffer_head]; // grab next byte
    _receive_buffer_head = (_receive_buffer_head + 1) % _SS_MAX_RX_BUFF;
    return d;
```

接收資料

SoftwareSerial 函式庫原始碼重要細節

SoftwareSerial.cpp : write()

```
size_t SoftwareSerial::write(uint8_t b)
{
    if (_tx_delay == 0) {
        setWriteError();
        return 0;
    }
    uint8_t oldSREG = SREG;
    cli(); // turn off interrupts for a clean txmit
    // Write the start bit
    tx_pin_write(_inverse_logic ? HIGH : LOW);
    tunedDelay(_tx_delay + XMIT_START_ADJUSTMENT); ← 傳送延遲
    // Write each of the 8 bits
    if (_inverse_logic){
    }
    else{
        for (byte mask = 0x01; mask; mask <<= 1){
            if (b & mask) // choose bit
                tx_pin_write(HIGH); // send 1
            else
                tx_pin_write(LOW); // send 0

            tunedDelay(_tx_delay); ← 傳送延遲
        }
        tx_pin_write(HIGH); // restore pin to natural state
    }
    SREG = oldSREG; // turn interrupts back on
    tunedDelay(_tx_delay); ← 傳送延遲
```

SoftwareSerial 函式庫原始碼重要細節

SoftwareSerial.cpp : recv ()

```
void SoftwareSerial::recv()
{
    uint8_t d = 0;

    // If RX line is high, then we don't see any start bit
    // so interrupt is probably not for us
    if (_inverse_logic ? rx_pin_read() : !rx_pin_read())
    {
        // Wait approximately 1/2 of a bit width to "center" the sample
        tunedDelay( rx delay centering); ← Start bit 延遲
        DebugPulse(_DEBUG_PIN2, 1);

        // Read each of the 8 bits
        for (uint8_t i=0x1; i; i <<= 1)
        {
            tunedDelay(_rx_delay_intrabit);
            DebugPulse(_DEBUG_PIN2, 1);
            uint8_t noti = ~i;
            if (rx_pin_read())
                d |= i;
            else // else clause added to ensure function timing is ~balanced
                d &= noti;
        }

        // skip the stop bit
        tunedDelay( rx delay stopbit); ← Stop bit 延遲
        DebugPulse(_DEBUG_PIN2, 1);
    }
}
```

讀取 8bit 資料

SoftwareSerial 函式庫原始碼重要細節

SoftwareSerial.cpp : recv ()

```
if (_inverse_logic)
    d = ~d;

// if buffer full, set the overflow flag and return 判斷 buffer 是否滿了
if ((_receive_buffer_tail + 1) % _SS_MAX_RX_BUFF != _receive_buffer_head)
{
    // save new data in buffer: tail points to where byte goes
    _receive_buffer[_receive_buffer_tail] = d; // save new byte
    _receive_buffer_tail = (_receive_buffer_tail + 1) % _SS_MAX_RX_BUFF;
}
else
{
    #if _DEBUG // for scope: pulse pin as overflow indictator
        DebugPulse(_DEBUG_PIN1, 1);
    #endif
    _buffer_overflow = true;
}
}
```

將資料丟進 buffer

An 86Duino board, a small single-board computer, is shown at an angle. It features a USB port, a DC-IN jack, and various digital and analog pins. The board is populated with various components including a microcontroller, memory, and passive components. The text "SoftwareSerial 函式庫的移植: 以 86Duino 為例" is overlaid on the board.

SoftwareSerial 函式庫的移植: 以 86Duino 為例

SoftwareSerial在 86Duino 上的移植

- 移植重點：
 - 在 86Duino 上，要使用具有 `attachInterrupt()` 功能的 **PIN** 腳來模擬 **RX**（**TX** 與原來一樣），掛中斷的方式與 **Arduino** 不同
 - 鮑率延遲的時間表需要實際 **tune** 過一次
 - 其餘部分不用動到

86Duino 的鮑率延遲時間表

```
static const DELAY_TABLE table[] =  
{  
    //  baud      rxcenter  rxintra  rxstop   tx  
    { 19200,      0,        50,      50,       49,    },  
    { 14400,      9,        67,      67,       66,    },  
    { 9600,       26,       100,     100,      100,    },  
    { 4800,       78,       204,     204,      204,    },  
    { 2400,      182,      410,     410,      412,    },  
    { 1200,      390,      826,     826,      828,    },  
    { 600,       806,     1654,    1654,     1660,   },  
    { 300,      1638,     3308,    3308,     3324,   },  
};
```

SoftwareSerial函式庫原始碼重要細節

SoftwareSerial.cpp : listen()

```
// This function sets the current object as the "listening"
// one and returns true if it replaces another
bool SoftwareSerial::listen()
{
    int i;
    if (active_object != this)
    {
        _buffer_overflow = false;
        io_DisableINT();
        _receive_buffer_head = _receive_buffer_tail = 0;
        active_object = this;
        io_RestoreINT();
        for(i=0; i<EXTERNAL_NUM_INTERRUPTS; i++)
            if(pinMap2[i] == _receivePin) break;
        if(i == EXTERNAL_NUM_INTERRUPTS) i = 0;
        attachInterrupt(i, SoftwareSerial::handle_interrupt, FALLING);
        return true;
    }

    return false;
}
```

判斷中斷觸發腳位

中斷觸發, 執行 handle_interrupt

SoftwareSerial函式庫原始碼重要細節

SoftwareSerial.cpp : write()

```
size_t SoftwareSerial::write(uint8_t b) {
    if (_tx_delay == 0) {
        setWriteError();
        return 0;
    }
    io_DisableINT();
    // Write the start bit
    tx_pin_write(_inverse_logic ? HIGH : LOW);
    tunedDelay( tx delay); ← 傳送延遲
    // Write each of the 8 bits
    if (_inverse_logic) {
        .
        .
        .
    }
    else ← 傳送資料
    {
        for (byte mask = 0x01; mask; mask <<= 1) {
            if (b & mask) // choose bit
                tx_pin_write(HIGH); // send 1
            else
                tx_pin_write(LOW); // send 0

            tunedDelay( tx delay); ← 傳送延遲
        }
        tx_pin_write(HIGH); // restore pin to natural state
    }
    io_RestoreINT();
    tunedDelay( tx delay); ← 傳送延遲
}
```

SoftwareSerial函式庫原始碼重要細節

SoftwareSerial.cpp : recv ()

```
// skip the stop bit
tunedDelay(_rx_delay_stopbit);
//DebugPulse(_DEBUG_PIN2, 1);
```

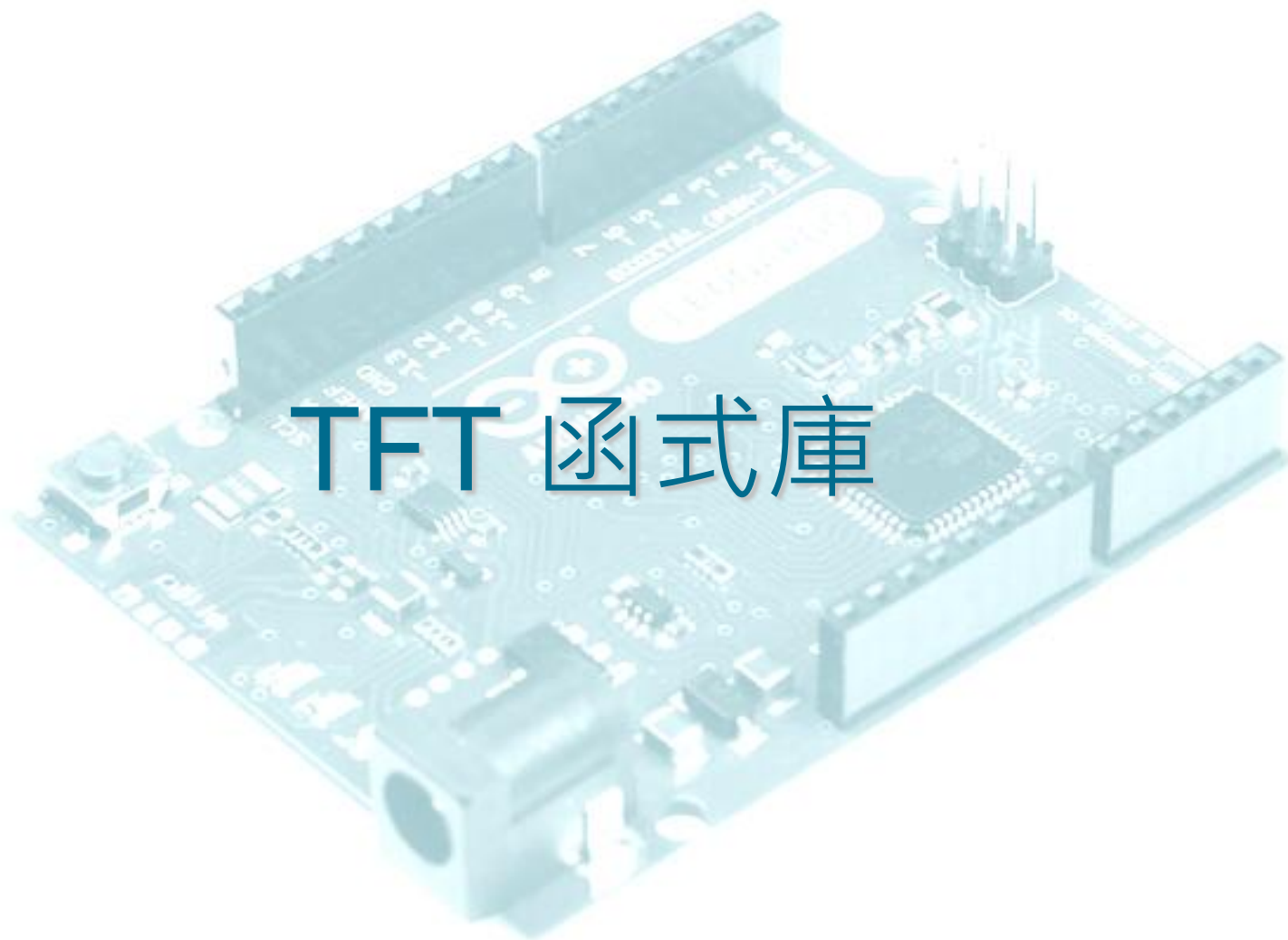
```
if (_inverse_logic)
    d = ~d;
```

```
// if buffer full, set the overflow flag and return 判斷 buffer 是否滿了
if ((receive_buffer_tail + 1) % SS_MAX_RX_BUFF != receive_buffer_head)
```

```
{
    // save new data in buffer: tail points to where byte goes 將資料丟進 buffer
    _receive_buffer[_receive_buffer_tail] = d; // save new byte
    _receive_buffer_tail = (_receive_buffer_tail + 1) % _SS_MAX_RX_BUFF;
}
```

```
else
{
    _buffer_overflow = true;
}
```

```
}
io_outpb(CROSSBARBASE + 0x90 + pinMap[_receivePin], 0x08); // switch to Encoder 將腳位切換成Encoder
}
```



TFT 函式庫

TFT shield 簡介

- 工作電壓：5V
- 螢幕解析度：160 x 128 pixels
- 色彩深度：16bit (R-5bit, G-5bit, B-6bit)
- 通訊介面：SPI
- 附帶 microSD 插槽



TFT 外觀

TFT 函式庫原始碼概觀

- <https://github.com/arduino/Arduino/tree/master/libraries/TFT>



TFT函式庫原始碼重要細節

TFT.h

```
#include "Arduino.h"
#include "utility/Adafruit_GFX.h" ← 高階的畫圖函式
#include "utility/Adafruit_ST7735.h" ← 低階的繪圖函式 (含初始化命令)

class TFT : public Adafruit_ST7735 {
public:
    TFT(uint8_t CS, uint8_t RS, uint8_t RST); ← 初始化自定義腳位

    void begin(); ← 初始化 TFT 螢幕
};
```

TFT 函式庫原始碼重要細節

TFT.cpp

```
TFT::TFT(uint8_t CS, uint8_t RS, uint8_t RST)
: Adafruit_ST7735(CS, RS, RST) ← 初始化自定義腳位
{
    // as we already know the orientation (landscape,
    // set default width and height without need to c
    _width = ST7735_TFTHEIGHT; ← 設定螢幕寬
    _height = ST7735_TFTWIDTH; ← 設定螢幕高
}

void TFT::begin() {
    initR(INITR_REDTAB); ← 初始化 TFT (送出連續的初始化命令)
    setRotation(1); ← 設定畫面旋轉方向
}
```

TFT 函式庫原始碼重要細節

- 以 drawFastVLine() 為例：

utility / Adafruit_ST7735.cpp

```
void Adafruit_ST7735::drawFastVLine(int16_t x, int16_t y, int16_t h,
uint16_t color) {

    // Rudimentary clipping
    if((x >= _width) || (y >= _height)) return;
    if((y+h-1) >= _height) h = _height-y;
    setAddrWindow(x, y, x, y+h-1);

    if (tabcolor == INITR_BLACKTAB)    color = swapcolor(color);

    uint8_t hi = color >> 8, lo = color;
    *rsport |= rspinmask; ← CS 設為 LOW，RS 設為 HIGH
    *csport &= ~cspinmask;
    while (h-->0) {
        spiwrite(hi); ← 透過 SPI 函式送命令，其
        spiwrite(lo); ← 它函式也是一樣
    }
    *csport |= cspinmask; ← CS 設為 HIGH
}
```

TFT Library 在 86Duino 上的移植

- 移植重點：
 - 將直接存取 ATmega 暫存器的程式，用 `digitalWrite()`、`digitalRead()` 函式替換
 - 延用 86Duino 既有的 SPI 函式庫，相關的函式如 `SPI.write()` 不需要改動

TFT Library 在 86Duino 上的移植

86Duino : Ada_ST77.cpp

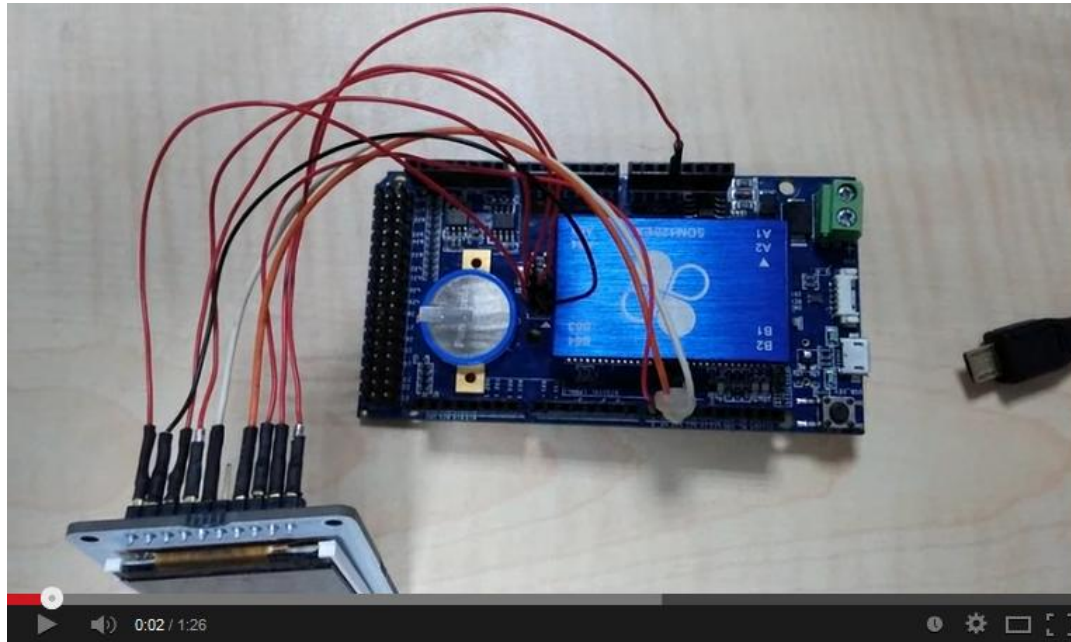
```
void Adafruit_ST7735::writecommand(uint8_t c) {  
    digitalWrite(_rs, LOW);  
    digitalWrite(_cs, LOW);  
  
    spiwrite(c);  
  
    digitalWrite(_cs, HIGH);  
}  
  
void Adafruit_ST7735::writedata(uint8_t c) {  
    digitalWrite(_rs, HIGH);  
    digitalWrite(_cs, LOW);  
  
    spiwrite(c);  
  
    digitalWrite(_cs, HIGH);  
}
```

Arduino : Adafruit_ST7735.cpp

```
void Adafruit_ST7735::writecommand(uint8_t c) {  
    *rsport &= ~rspinmask;  
    *csport &= ~cspinmask;  
  
    //Serial.print("C ");  
    spiwrite(c);  
  
    *csport |= cspinmask;  
}  
  
void Adafruit_ST7735::writedata(uint8_t c) {  
    *rsport |= rspinmask;  
    *csport &= ~cspinmask;  
  
    //Serial.print("D ");  
    spiwrite(c);  
  
    *csport |= cspinmask;  
}
```

移植

TFT demo 影片



- <https://www.youtube.com/watch?v=eZefqL6FtOE>



Thank You

感謝大家來參與 **Arduino** 原始碼讀書會

接下來問題討論時間

DMP Electronics Inc. (瞻營全電子)
robotics@dmp.com.tw