Duinotech WIFI Scanner

This is a simple tool using just four Duinotech parts that can sit on your workbench and quickly tell you what WIFI networks are nearby and their relative strengths. If you’re always working with different WIFI networks like we are, or even have a few XC4614’s running in hotspot mode, it’s handy to know which networks are up and which are not.

Features:

* Displays 5 most powerful WIFI networks nearby.
* Shows relative strength.
* Automatic update every 5 seconds.
* No soldering needed.



## Shopping List:

XC4410 Uno Main Board

XC4614 WIFI Shield

XC4616 84x48 Dot Matrix LCD Display

WC6026 Socket-Socket Jumper Leads

Alternatively, if you don’t mind soldering, and you’d like the display fixed rigidly to the shield, you can replace the jumper leads with a pair of 8-way headers from HM3207 soldered back-to-back.

## Connections:

As with all project involving shields, wiring is straightforward. The shield is plugged into the top of the Uno, and then eight of the jumper leads are run from shield to the display.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Uno | Wifi | LCD | Function | Colour wire |
| 5V | 5V |  | Power |  |
| GND | GND |  | Ground |  |
| D0 (RX) | RX |  | Data from shield |  |
| D1 (TX) | TX |  | Data to shield |  |
| D2 |  | VCC | Power to LCD | Black |
| D3 |  | GND | Ground for LCD | Brown |
| D4 |  | SCE | Serial Chip Enable | Red |
| D5 |  | RST | Reset | Orange |
| D6 |  | D/C | Data/Command | Yellow |
| D7 |  | DN(MOSI) | Serial Data | Green |
| D8 |  | SCLK | Serial Clock | Blue |

The colour codes are not absolutely necessary, but will make it easier to follow the wiring as it is built. They are also easily separated as a group from the pack of jumper leads. By default, the LED backlight is not set to light up (because it needs 3.3V, but the Uno board runs on 5V). If you would like the LED to light up, connect the LED pin on the LCD to one of the 3V3 pins on the ‘ESP13’ side of the board.

## Code Upload:

There are no external libraries to download- everything is included in the Sketch section below (it’s a fairly long sketch because it even includes font bitmap data). Once all the connections are made, ensure that the small switches labelled 1 and 2 on the WIFI shield are switched off by moving them away from the ON label. This frees up the Arduino serial port for uploading. Select the Uno board and correct serial port, load the sketch and press upload. If the upload occurs successfully, turn the small white switches back to on, and reset the Uno. If the screen comes up blank, check the wiring between the screen and the shield. If the screen shows ‘No Networks…’ or gets stuck on’Starting…’, and there are networks in range, then double check the switch positions and try unplugging and replugging the Uno power.

## How it works:

The WIFI shield provides most of the smarts in detecting the WIFI networks, the Uno simply tells the shield what to do and displays the results.

The commands that are sent to the shield are as follows:

### ATE0:

Tells the shield to not echo back commands that are issued to it. This just simplifies the serial communication because there is less data going back and forth.

### AT+CWMODE=1:

This command to tell the shield to only behave as a station, and not an access point. Again, because the shield isn’t busy trying to provide an access point, it has more time to do what we want it to do.

### AT+CWLAP:

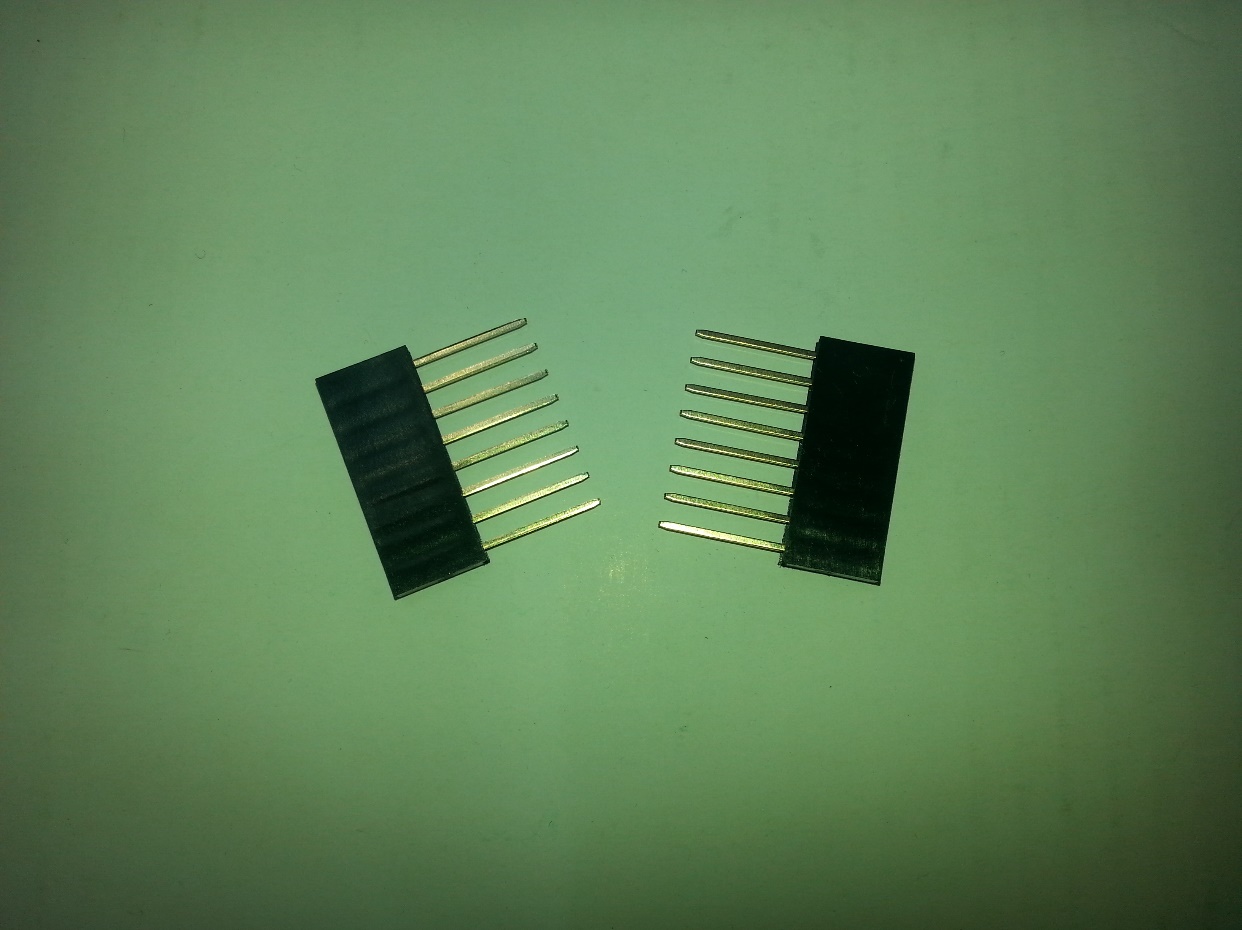
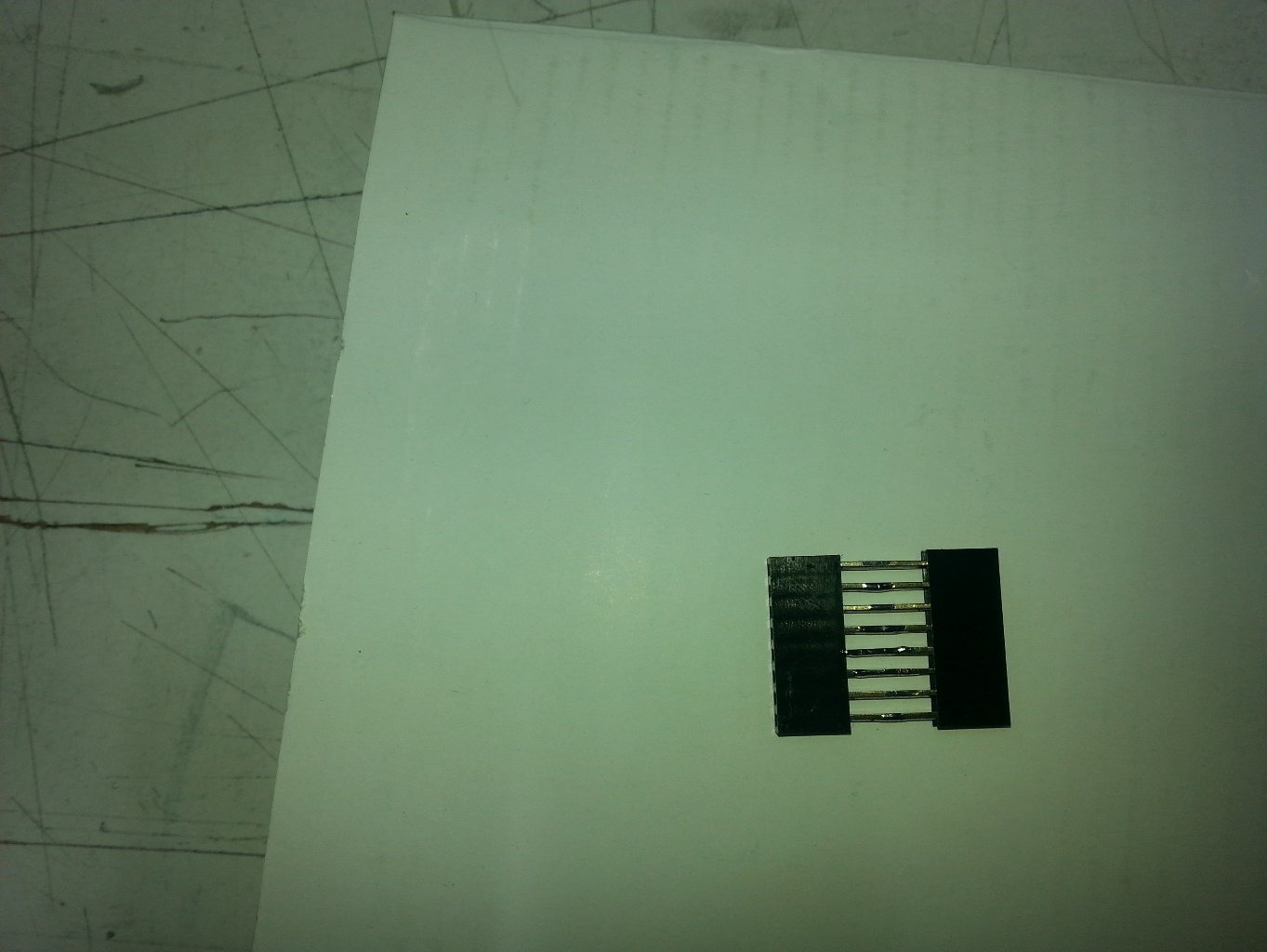
This is the command that does most of the work, and asks the shield to return data about the access points that are nearby. Each line of data looks like below:

+CWLAP:(3,"Arduino",-41,"74:da:38:3c:71:99",1,-47)

The SSID name is that data between the first and second quote marks, and the SSID strength (RSSI) is between the second and third quote marks, which is how the sketch finds it. The sketch waits for five seconds for whatever data comes from the shield, and then processes it. First it separates the SSID name and strength and stores them, then it converts the strength from a negative number between -99 and -1 to a positive number between 1 and 99, and displays the five most powerful networks in order. If no network is found (which is what it will also think if the white switches are turned off), then the ‘No Networks…’ message is displayed.

## Improvements:

To create a more rigid assembly, solder together two of the eight-way headers from the HM3207 pack:

The below photo shows how the WIFI Scanner looks when it is assembled using the HM3207 to give a more compact arrangement. Note how we only use the last seven pins of the header adapter so that the LED pin is not connected.



Note that you still have to run a jumper lead to a 3.3V pin if you want the LED backlight on.

Other information that comes from the +CWLAP command includes:

Security type is the first number after the bracket. The number decodes as 0=open, 1=WEP, 2=WPA\_PSK, 3=WPA2\_PSK, 4=WPA\_WPA2\_PSK.

MAC address of the access point is between the second pair of quote marks. This can be used to distinguish two access points with the same name.

The channel number and frequency offset are the last two numbers, but we find they aren’t really much use in most applications.

You could display more of this information- for example, looking for security=0 so you can sniff out open WIFI hotspots.

This project really only uses the very basic functions of the XC4614 WIFI shield, and we hope to do some more advanced projects soon.

## Sketch:

//Duinotech Wifi Scanner

//XC4614 Wifi Shield - ESP13 module

//XC4616 84x48 Dot Matrix Display

//XC4410 Uno Main board (could be any other that the shield fits on, change Serial to Serial1 on Leonardo)

// LCD VCC GND SCE RST DC MOSI SCLK

// Uno 2 3 4 5 6 7 8

#define PIN\_VCC 2

#define PIN\_GND 3

#define PIN\_SCE 4

#define PIN\_RESET 5

#define PIN\_DC 6

#define PIN\_SDIN 7

#define PIN\_SCLK 8

#define LCD\_C LOW

#define LCD\_D HIGH

#define LCD\_X 84

#define LCD\_Y 48

#define BUFSIZE 700

char buf[BUFSIZE]="";

int bufptr=0;

#define SSIDCOUNT 12

#define SSIDLEN 14

char ssidbuf[SSIDCOUNT][SSIDLEN+1];

int ssidstrength[SSIDCOUNT];

static const byte ASCII[][5] =

{

{0,0,0,0,0}, //

{0,95,95,0,0}, //!

{3,3,0,3,3}, //"

{60,60,36,60,60}, //#

{36,46,127,58,18}, //$

{99,19,8,100,99}, //%

{54,127,73,127,126}, //&

{0,3,3,0,0}, //

{0,28,62,99,65}, //(

{65,99,62,28,0}, //)

{84,56,56,84,0}, //\*

{16,124,124,16,0}, //+

{0,128,224,96,0}, //,

{16,16,16,16,0}, //-

{0,0,96,96,0}, //.

{112,120,12,7,3}, ///

{62,127,65,127,62}, //0

{0,66,127,127,64}, //1

{98,115,89,79,70}, //2

{34,107,73,127,54}, //3

{60,34,127,127,32}, //4

{79,79,73,121,49}, //5

{60,126,75,121,48}, //6

{1,97,121,31,7}, //7

{54,127,73,127,54}, //8

{6,79,105,63,30}, //9

{0,108,108,0,0}, //:

{128,236,108,0,0}, //;

{8,28,54,34,0}, //<

{40,40,40,40,0}, //=

{34,54,28,8,0}, //>

{2,107,105,15,6}, //?

{60,126,66,94,92}, //@

{126,127,17,127,126}, //A

{127,127,73,127,54}, //B

{62,127,65,99,34}, //C

{127,127,65,127,62}, //D

{127,127,73,73,65}, //E

{127,127,9,9,1}, //F

{62,127,65,113,113}, //G

{127,127,8,127,127}, //H

{65,127,127,65,0}, //I

{64,64,64,127,63}, //J

{127,127,28,119,99}, //K

{127,127,64,64,64}, //L

{127,126,12,126,127}, //M

{127,127,30,127,127}, //N

{62,127,65,127,62}, //O

{127,127,9,15,6}, //P

{62,127,97,127,94}, //Q

{127,127,25,127,102}, //R

{38,111,73,123,50}, //S

{1,127,127,1,0}, //T

{63,127,64,127,63}, //U

{31,63,96,63,31}, //V

{127,63,112,63,127}, //W

{119,127,8,127,119}, //X

{71,111,56,31,7}, //Y

{113,121,77,71,67}, //Z

{127,127,65,65,0}, //[

{3,15,60,112,64}, //backslash

{65,65,127,127,0}, //]

{4,6,3,6,4}, //^

{64,64,64,64,64}, //\_

{1,3,2,0,0}, //`

{32,116,84,124,120}, //a

{127,127,72,120,48}, //b

{56,124,68,68,68}, //c

{48,120,72,127,127}, //d

{56,124,84,92,88}, //e

{126,127,17,1,0}, //f

{152,188,164,252,124}, //g

{127,127,8,120,112}, //h

{72,122,122,64,0}, //i

{128,128,128,250,122}, //j

{127,127,56,108,68}, //k

{0,127,127,0,0}, //l

{120,8,120,8,112}, //m

{120,120,8,120,112}, //n

{48,120,72,120,48}, //o

{248,248,72,120,48}, //p

{48,120,72,248,248}, //q

{112,120,8,8,8}, //r

{72,92,84,116,36}, //s

{0,60,124,72,0}, //t

{56,120,64,120,120}, //u

{24,56,96,56,24}, //v

{120,96,16,96,120}, //w

{72,120,48,120,72}, //x

{152,184,160,248,120}, //y

{72,104,120,88,72}, //z

{8,62,119,65,65}, //{

{0,0,127,0,0}, //|

{65,65,119,62,8}, //}

{2,3,1,2,3}, //~

{64,0,64,0,64} //â€¦

};

void setup(void)

{

delay(100);

LcdInitialise();

LcdSetx(0);

LcdSety(0);

LcdString("WIFI SCANNER"); //startup display

LcdSetx(0);

LcdSety(1);

LcdString("Starting...");

Serial.begin(115200); //start wifi shield

Serial.println("ATE0"); //echo off

delay(300); //wait for response

serialpurge(); //ignore response

Serial.println("AT+CWMODE=1"); //Station mode only

serialwaitpurge(100);

}

void loop(void)

{

int d;

Serial.println("AT+CWLAP"); //Get AP's

buf[0]=0; //clear buffers

bufptr=0;

long tmout;

tmout=millis();

while(millis()-tmout<5000){ //go for time before restarting

if(Serial.available()){

d=Serial.read();

if(bufptr<BUFSIZE-2){buf[bufptr]=d;bufptr++;buf[bufptr]=0;} //store in larger buffer

}

}

//clear the buffers

for(int x=0;x<SSIDCOUNT;x++){

for(int y=0;y<SSIDLEN+1;y++){

ssidbuf[x][y]=0;

}

ssidstrength[x]=0;

}

//process into smaller buffers so we can sort by strength order

int ssidno=0;

int ssidptr=0;

int t=0;

for(int i=0;i<bufptr;i++){

if(buf[i]==34){t++;buf[i]=0;} //quote mark > next token found

if(buf[i]=='('){ //open bracket > next line of info

t=0; //reset token

ssidno++; //next ssid

ssidptr=0; //reset to start of char array

}

if(t==1){ //inside first set of quotes

if(buf[i]>31){ //valid character

if(ssidno<SSIDCOUNT){ //check we've got space

if(ssidptr<SSIDLEN-1){ //check we've got space

ssidbuf[ssidno][ssidptr]=buf[i];ssidptr++;ssidbuf[ssidno][ssidptr]=0; //add to current SSID name

}

}

}

}

if(t==2){ //between quotes- SSID strength

if(ssidno<SSIDCOUNT){ //we've got space

if(buf[i]<'9'+1){ //it's a number

if(buf[i]>'0'-1){ //it's a number

ssidstrength[ssidno]=ssidstrength[ssidno]\*10+buf[i]-'0'; //add to current strength

}

}

}

}

}

//all SSID's are in the arrays, do a little bit of postprocessing

for(ssidno=0;ssidno<SSIDCOUNT;ssidno++){

if(ssidbuf[ssidno][12]){ssidbuf[ssidno][11]=127;ssidbuf[ssidno][12]=0;} //if name is too long to display, truncate and end with ...

ssidstrength[ssidno]=100-ssidstrength[ssidno]; //change from -99..0 to 1..100

}

LcdClear();

LcdSetx(0);

LcdSety(0);

LcdString("WIFI SCANNER");

int y=1;

for(int x=99;x>0;x--){ //scan from strongest to weakest

for(ssidno=0;ssidno<SSIDCOUNT;ssidno++){ //scan through ssid's

if(ssidstrength[ssidno]==x){ //if strength == x

if(y<6){ //and there's space on the screen

LcdSetx(0);

LcdSety(y);

LcdString(&ssidbuf[ssidno][0]); //show it

LcdSetx(72);

LcdSety(y);

LcdChar(((ssidstrength[ssidno])/10)%10+'0');

LcdChar(((ssidstrength[ssidno]))%10+'0'); //show strength

y++;

}

}

}

}

if(y==1){

LcdSetx(0);

LcdSety(1);

LcdString("No Networks...");

}

}

void dumpserial(){

while(Serial.available()){

int d;

d=Serial.read();

LcdChar(d);

}

}

void serialpurge(){ //purges incoming serial data

while(Serial.available()){Serial.read();}

}

void serialwaitpurge(long t){ //wait until data appears, then purge

while(!Serial.available()){}

delay(t);

while(Serial.available()){Serial.read();}

}

void LcdCharacter(char character)

{

LcdWrite(LCD\_D, 0x00);

for (int index = 0; index < 5; index++)

{

LcdWrite(LCD\_D, ASCII[character - 0x20][index]);

}

// LcdWrite(LCD\_D, 0x00);

}

void LcdChar(char character) // a more robust version that tests for out of range values

{

if(character>127){return;}

if(character<32){return;}

LcdCharacter(character);

}

void LcdClear(void)

{

for (int index = 0; index < LCD\_X \* LCD\_Y / 8; index++)

{

LcdWrite(LCD\_D, 0x00);

}

}

void LcdInitialise(void)

{

pinMode(PIN\_VCC, OUTPUT); //VCC

digitalWrite(PIN\_VCC,HIGH);

pinMode(PIN\_GND, OUTPUT); //GND

digitalWrite(PIN\_GND,LOW);

pinMode(PIN\_SCE, OUTPUT);

pinMode(PIN\_RESET, OUTPUT);

pinMode(PIN\_DC, OUTPUT);

pinMode(PIN\_SDIN, OUTPUT);

pinMode(PIN\_SCLK, OUTPUT);

digitalWrite(PIN\_RESET, LOW);

delay(100);

digitalWrite(PIN\_RESET, HIGH);

LcdWrite( LCD\_C, 0x21 ); // LCD Extended Commands.

LcdWrite( LCD\_C, 0xB0 ); // Set LCD Vop (Contrast). //A8-B1 TRY ADJUSTING THIS IF IT'S NOT WORKING 80-FF are valid

LcdWrite( LCD\_C, 0x04 ); // Set Temp coefficent. //0x04

LcdWrite( LCD\_C, 0x14 ); // LCD bias mode 1:40. //0x14

LcdWrite(LCD\_C, 0x20);

LcdWrite(LCD\_C, 0x0C); //display on and normal

// LcdWrite(LCD\_C, 0x0D); //display on and inverted

LcdClear();

LcdSetx(0);

LcdSety(0);

}

void LcdString(char \*characters)

{

while (\*characters)

{

LcdChar(\*characters++);

}

}

void LcdWrite(byte dc, byte data)

{

digitalWrite(PIN\_DC, dc);

digitalWrite(PIN\_SCE, LOW);

shiftOut(PIN\_SDIN, PIN\_SCLK, MSBFIRST, data);

digitalWrite(PIN\_SCE, HIGH);

}

void LcdSetx(byte x)

{

if(x<LCD\_X)

{LcdWrite(LCD\_C,0x80 | x);

}

}

void LcdSety(byte y)

{

if(y<(LCD\_Y/8))

{LcdWrite(LCD\_C,0x40 | y);

}

}