```
// feather_expander_3
// 30.5.17
// development of wifi control of expander board
// Adafruit Feather M0 plus MCP23017 port expander
// communication via UDP
// Credit: Hobbytronics, Adafruit, Arduino Cookbook
// Development: Julian Rogers
// Assignment of Feather pins:
// No connections for this test except I2C and LED on pin 13

//
// on MCP23017 port expander, ports in use are A0, A2, A4, A6
// a better choice would have been A0, A1, A2, A3
// would have saved a few program lines!

//Libraries:

#include <Adafruit_WINC1500.h>
#include <Adafruit_WINC1500Udp.h>

#include <SPI.h>
#include <Wire.h>

const byte  mcp_address=0x20;      // I2C Address of MCP23017 Chip
const byte  GPIOA=0x12;            // Register Address of Port A
const byte  GPIOB=0x13;            // Register Address of Port B

// Define the WINC1500 board connections below.
// If you're following the Adafruit WINC1500 board
// guide you don't need to modify these:
#define WINC_CS   8     // chip select for wifi
#define WINC_IRQ  7     // irq for wifi
#define WINC_RST  4     // reset pin for wifi - controlled by library
#define WINC_EN   2     // or, tie EN to VCC and comment this out - high to enable wifi
// The SPI pins of the WINC1500 (SCK, MOSI, MISO) should be
// connected to the hardware SPI port of the Arduino.
// On an Uno or compatible these are SCK = #13, MISO = #12, MOSI = #11.
// On an Arduino Zero use the 6-pin ICSP header, see:
//   https://www.arduino.cc/en/Reference/SPI

// Setup the WINC1500 connection with the pins above and the default hardware SPI.
Adafruit_WINC1500 WiFi(WINC_CS, WINC_IRQ, WINC_RST);

// Or just use hardware SPI (SCK/MOSI/MISO) and defaults, SS -> #10, INT -> #7, RST -> #5, EN -> 3-5V
//Adafruit_WINC1500 WiFi;
int status = WL_IDLE_STATUS;
char ssid[] = "******";  //  your network SSID (name)
char pass[] = "******";       // your network password
int keyIndex = 0;            // your network key Index number (needed only for WEP)

unsigned int localPort = 2390;      // local port to listen on

char packetBuffer[255]; //buffer to hold incoming packet
char  ReplyBuffer[255];       // a string to send back

Adafruit_WINC1500UDP Udp;

////////////////////////////////////////////////////////////////////

void setup() {
  pinMode(13, OUTPUT);        // "running" LED
 //Send settings to MCP device
  Wire.begin();              // join i2c bus (address optional for master)

  // IOCON.BANK defaults to 0 which is what we want.
  // So we are using Table 1-4 on page 9 of datasheet

  Wire.beginTransmission(mcp_address);
  Wire.write(0x00); // IODIRA register
  Wire.write(0x00); // set all of bank A to outputs
  Wire.write(0x00); // set all of bank B to outputs
  Wire.endTransmission();

  Wire.beginTransmission(mcp_address);
  Wire.write(0x01); // IODIRB register
  Wire.write(0x00); // set all of bank A to outputs
  Wire.write(0x00); // set all of bank B to outputs
  Wire.endTransmission();
////////////////////////////////////////////////////////////////////////////

  Serial.begin(9600);

//////////////////////////////////////////////////////////////////////

  #ifdef WINC_EN
    pinMode(WINC_EN, OUTPUT);
    digitalWrite(WINC_EN, HIGH);
  #endif

  // attempt to connect to Wifi network:
```

```
  while ( status != WL_CONNECTED) {
    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);

    // wait for connection:

   delay(1000);

  }

  Serial.println("Connected to wifi");
  printWifiStatus();

  Serial.println("\nStarting connection to server...");
    // if you get a connection, report back via serial:
  Udp.begin(localPort);

  ////////////////////////////////////////////////

}

// End of setup()

//////////////////////////////////////////////////////
//Functions start here

void printWifiStatus() {
  // print the SSID of the network you're attached to:
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());

  // print your WiFi shield's IP address:
  IPAddress ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);

  // print the received signal strength:
  long rssi = WiFi.RSSI();
  Serial.print("signal strength (RSSI):");
  Serial.print(rssi);
  Serial.println(" dBm");
}

////////////////////////////////////////////////////////////


void loop() {

  // This is where the program checks to see if data is being sent
  // and sends back status data

  // if there's data available, read a packet
  int packetSize = Udp.parsePacket();
  if (packetSize)
  {
    Serial.print("Received packet of size ");
    Serial.println(packetSize);
    Serial.print("From ");
    IPAddress remoteIp = Udp.remoteIP();
    Serial.print(remoteIp);
    Serial.print(", port ");
    Serial.println(Udp.remotePort());

    // read the packet into packetBufffer
    int len = Udp.read(packetBuffer, 255);
    if (len > 0) packetBuffer[len] = 0;

    Serial.println("Contents:");
    Serial.println(packetBuffer);
    for(int x = 0; x < (len + 1); x++){
      ReplyBuffer[x] = packetBuffer[x];
    }

     //respond to remote command
     //
     if(packetSize == 2){
      byte num = 0;
      byte received = atoi(packetBuffer);
      // these next few lines are needed as MCP chip
      // is not best connected! (Also, there could be a neater way of doing this)
      // scan 4 bit command value and re-assign bits to (effectively) a  7 bit value
      if(received / 8 == 1){
       num = 64;
      }
      received = received % 8;
      if(received / 4 == 1){
        num = num + 16;
      }
      received = received % 4;
```

```
      if(received / 2 == 1){
        num = num + 4;
      }
      received = received % 2;
      if(received == 1){
        num = num + 1;
      }
      Wire.beginTransmission(mcp_address);
      Wire.write(GPIOA);      // address bank A
      Wire.write(num);  // value to send
      Wire.endTransmission();
     }

    Serial.println("sending reply to remote...");
    Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());

    Udp.write("Received ");
    Udp.write(ReplyBuffer);
    //should develop this to report status, errors etc?
    Udp.endPacket();

    }

  // D13 is the 'running' LED
   digitalWrite(13, HIGH);
   delay(250);
   digitalWrite(13, LOW);
   delay(250);
  Serial.println("Running...");
}

//end of loop
//and program!
```