Τελικό Project

Αφού είδαμε τη λειτουργία κάθε μιας διάταξης ξεχωριστά, ήρθε η στιγμή να συνδέσουμε όλα τα επιμέρους τμήματα σε ένα ενιαίο σύνολο.

Η Τελική Διάταξη

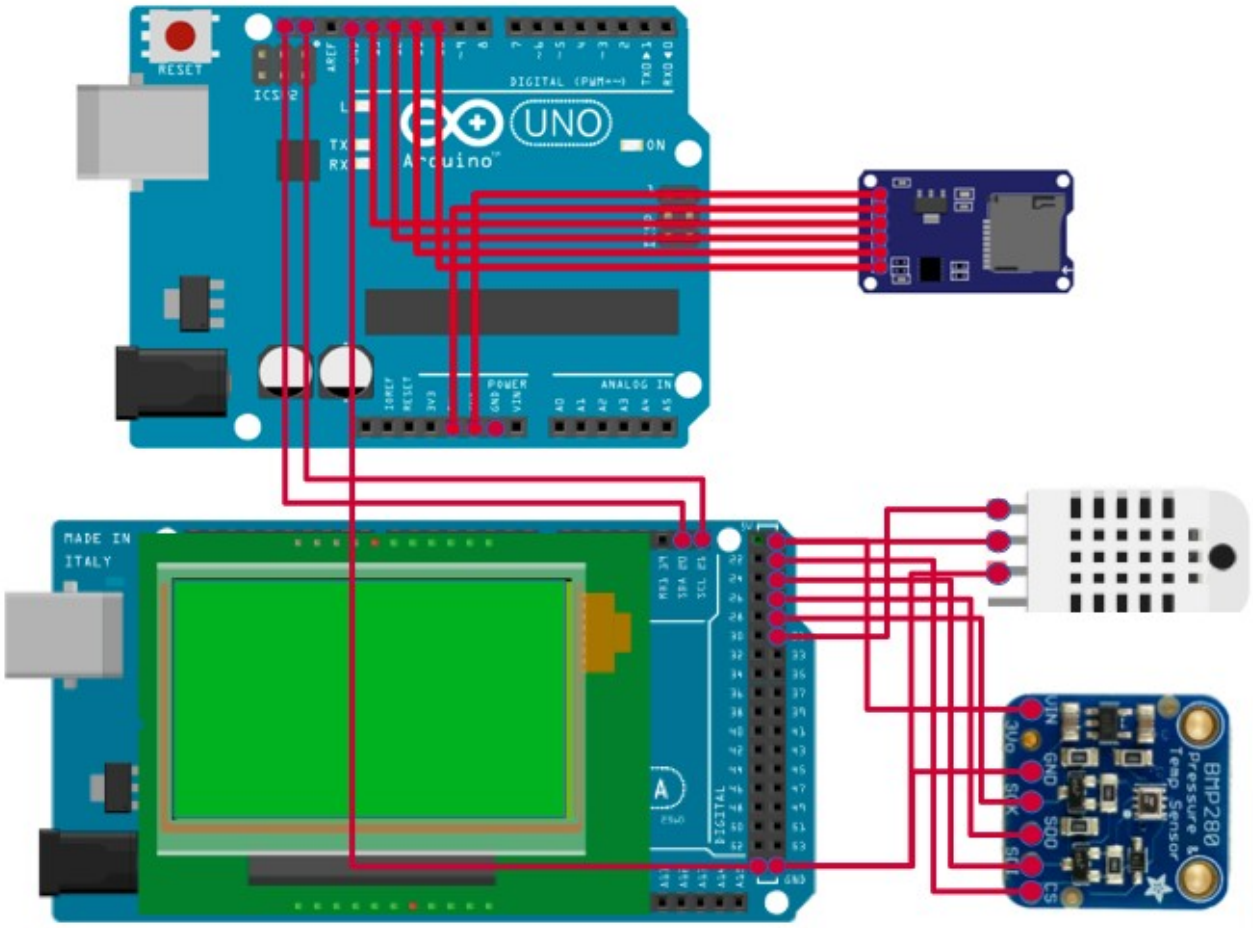
Η τελική διάταξη περιλαμβάνει

- Τη μονάδα **Master**. Είναι η μονάδα που
  - παίρνει τις μετρήσεις,
  - τις εμφανίζει στην οθόνη
  - τις μεταδίδει στη μονάδα Slave.

  Αποτελείται από
  - Arduino Mega
  - Αισθητήρας DHT11
  - Αισθητήτρας BMP280
  - Οθόνη Αφής TFT LCD 2,8”
  - Παρελκόμενα
- Τη μονάδα **Slave**. Είναι η μονάδα που
  - δέχεται τις μετρήσεις από τη μονάδα Master,
  - τις αποθηκεύει στην μονάδα αποθήκευσης,
  - τις εμφανίζει με τη μορφή ιστοσελίδας στο τοπικό δίκτυο
  - τις μεταδίδει στο Twitter.

  Αποτελείται από
  - Arduino Uno WiFi
  - SD Card Module
  - Παρελκόμενα

Η τελική διάταξη απεικονίζεται στο ακόλουθο σχήμα.



Στο σχήμα αντί για Arduino WiFi χρησιμοποιήθηκε "απλό" Arduino μιας και δεν βρέθηκε αντίστοιχη διάταξη.

Σύγχρονη Σύνδεση i2c

Μία από τις κύριες δυσκολίες είναι η επικονωνία μεταξύ Arduino Mega και Arduino Uno WiFi.

Γίνεται σύγχρονα χρησιμοποιώντας το πρωτόκολλο **i2c**. Για να λειτουργήσει το πρωτόκολλο αυτό απαιτείται η βιβλιοθήκη Wire.h και στις δύο διατάξεις αλλά και φυσική σύνδεση μεταξύ των Arduino ως εξής

- SDA – SDA
- SCL – SCL
- Κοινή γείωση

Για καλύτερο συντονισμό, προτιμότερο είναι στις παραπάνω συνδέσεις να χρησιμοποιήσουμε τα εξειδικευμένα pin που έχει κάθε μονάδα.

Μιας και η τελική διάταξη αποτελείται από δύο μονάδες, υπάρχει κώδικας που τρέχει στο Master και διαφορετικός κώδικας που τρέχει στο Slave.

Αναλυτικά παρουσιάζονται παρακάτω

Κώδικας Master

```
#include <Adafruit_GFX.h>
#include <MCUFRIEND_kbv.h>
#include <TouchScreen.h>
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>
#include "DHT.h"
#include <Wire.h>

#define BMP_SCK 29  // Έξοδος SCK του αισθητήρα στο Pin 13 του Arduino
#define BMP_MISO 27 // Έξοδος SDO του αισθητήρα στο Pin 12 του Arduino
#define BMP_MOSI 25 // Έξοδος SDI του αισθητήρα στο Pin 11 του Arduino
#define BMP_CS 23   // Έξοδος CS του αισθητήρα στο Pin 10 του Arduino

#define MINPRESSURE 200
#define MAXPRESSURE 1000

#define DHTPIN 31  // Το Ψηφιακό Pin 6 θα χρησιμοποιηθεί για διάβασμα δεδομένων
#define DHTTYPE DHT11   // DHT 11

DHT dht(DHTPIN, DHTTYPE); // Βασικό Αντικείμενο για αποθήκευση Μετρήσεων

Adafruit_BMP280 bmp(BMP_CS, BMP_MOSI, BMP_MISO,  BMP_SCK); // Βασικό Αντικείμενο Μετρήσεων
float temperature;
float pressure;
float hum;

String msg = "";
char buff[20];
int counter = 0;

const int XP=8,XM=A2,YP=A3,YM=9; //240x320 ID=0x9341
const int TS_LEFT=878,TS_RT=119,TS_TOP=107,TS_BOT=896;

MCUFRIEND_kbv tft;

TouchScreen ts = TouchScreen(XP, YP, XM, YM, 300);

Adafruit_GFX_Button t_btn, h_btn, p_btn, all_btn;

int pixel_x, pixel_y;     //Touch_getXY() updates global vars

#define BLACK   0x0000
#define BLUE    0x001F
#define RED     0xF800
#define GREEN   0x07E0
#define CYAN    0x07FF
#define MAGENTA 0xF81F
```

```
#define YELLOW  0xFFE0
#define WHITE   0xFFFF

void setup(void){

 counter += 1;

 Serial.begin(9600);
 Wire.begin();
 bmp.begin();
 dht.begin();

 uint16_t ID = tft.readID();
 if (ID == 0xD3D3) ID = 0x9486;

 tft.begin(ID);
 tft.setRotation(0);
 tft.fillScreen(BLACK);
 tft.setFont(NULL);

 t_btn.initButton(&tft,  120, 60, 200, 50, WHITE, RED, WHITE, "TEMP(C)", 2);
 h_btn.initButton(&tft, 120, 130, 200, 50, WHITE, RED, WHITE, "HUMM(%)", 2);
 p_btn.initButton(&tft, 120, 200, 200, 50, WHITE, RED, WHITE, "PRESS(mb)", 2);
 all_btn.initButton(&tft, 120, 270, 200, 50, WHITE, RED, WHITE, "ALL", 2);

 t_btn.drawButton(false);
 h_btn.drawButton(false);
 p_btn.drawButton(false);
 all_btn.drawButton(false);

}


void loop(void){

 Serial.println(counter);

 pressure = bmp.readPressure();  // Αποθήκευση τιμής Ατμοσφαιρικής Πίεσης
 temperature = bmp.readTemperature(); // Αποθήκευση τιμής Θερμοκρασίας
 hum = dht.readHumidity();          // Αποθήκευση τιμής Υγρασίας (DHT.humidity) στη μεταβλητή hum

 bool down = Touch_getXY();
 t_btn.press(down && t_btn.contains(pixel_x, pixel_y));
 h_btn.press(down && h_btn.contains(pixel_x, pixel_y));
 p_btn.press(down && p_btn.contains(pixel_x, pixel_y));
 all_btn.press(down && all_btn.contains(pixel_x, pixel_y));

  if (t_btn.justReleased() || h_btn.justReleased() || p_btn.justReleased() || all_btn.justReleased()){
     t_btn.drawButton();
     h_btn.drawButton();
     p_btn.drawButton();
```

```
      all_btn.drawButton();
    }
    if (t_btn.justPressed()) {
      msg = String(temperature) + " C";
      tft.fillScreen(BLACK);
      msg.toCharArray(buff, 50);
      showmsgXY(10, 100, 3, NULL, buff);
      delay(5000);
      tft.fillScreen(BLACK);
    }
    if (h_btn.justPressed()) {
      msg = String(hum) + " %";
      tft.fillScreen(BLACK);
      msg.toCharArray(buff, 50);
      showmsgXY(10, 100, 3, NULL, buff);
      delay(5000);
      tft.fillScreen(BLACK);
    }
    if (p_btn.justPressed()) {
      msg = String(pressure/100) + " mbar";
      tft.fillScreen(BLACK);
      msg.toCharArray(buff, 50);
      showmsgXY(10, 100, 3, NULL, buff);
      delay(5000);
      tft.fillScreen(BLACK);
    }
    if (all_btn.justPressed()) {
      msg = String(temperature) + " C\n\n" + String(hum) + " %\n\n" + String(pressure/100) + " mbar";
      tft.fillScreen(BLACK);
      msg.toCharArray(buff, 50);
      showmsgXY(0, 100, 3, NULL, buff);
      delay(5000);
      tft.fillScreen(BLACK);
    }
    // Δημιουργία String τιμών
    msg = String(temperature);
    msg += ",";
    msg += String(hum);
    msg += ",";
    msg += String(pressure/100);
    msg += "\n";
    msg.toCharArray(buff,20); // Μετατροπή String σε πίνακα Χαρακτήρων για μετάδοση
    for(int i=0;i<20;i++){  // Μετάδοση 20 χαρακτήρων
      Wire.beginTransmission(9); // Εκκίνηση μετάδοσης στη διεύθυνση 9 (dummy τιμή)
      Wire.write(buff[i]);              // Μετάδοση ενός Χαρακτήρα
      Wire.endTransmission();     // Τέλος Μετάδοσης
    }

    delay(500);    // Αναμονή 0,5 sec
}
```

```
bool Touch_getXY(void){
   TSPoint p = ts.getPoint();
   pinMode(YP, OUTPUT);
   pinMode(XM, OUTPUT);
   digitalWrite(YP, HIGH);
   digitalWrite(XM, HIGH);
   bool pressed = (p.z > MINPRESSURE && p.z < MAXPRESSURE);
   if (pressed) {
      pixel_x = map(p.x, TS_LEFT, TS_RT, 0, tft.width());
      pixel_y = map(p.y, TS_TOP, TS_BOT, 0, tft.height());
   }
   return pressed;
}

void showmsgXY(int x, int y, int sz, const GFXfont *f, const char *msg){
   //int16_t x1, y1;
   //uint16_t wid, ht;
   tft.setFont(f);
   tft.setCursor(x, y);
   tft.setTextColor(RED);
   tft.setTextSize(sz);
   tft.print(msg);
}
```

Κώδικας Slave

```
#include <SPI.h>
#include <WiFiNINA.h>
#include <Wire.h>
#include "Twitter.h"
#include <SD.h>

const int chipSelect = 10;

char ssid[] = SECRET_SSID;
char pass[] = SECRET_PASS;
int keyIndex = 0;

int status = WL_IDLE_STATUS;
WiFiServer server(80);
Twitter twitter("MY_SECRET_KEY");

char msg[20] = "";
char x = 0;
String message = "";
String rest = "";
String temperature = "";
String humidity = "";
String pressure = "";
int index = 0;
int counter = 0;

void setup(){

 Serial.begin(9600);

 if (!SD.begin(chipSelect)) { // Αν η ενεργοποίηση της κάρτας ΔΕΝ πετύχει
    Serial.println("Card failed, or not present"); // Τύπωσε μήνυμα αποτυχίας
    while (1); // Μπες σε ατέρμονα βρόχο
 }else{
  Serial.println("card initialized."); // Αλλιώς εκτύπωσε μήνυμα επιτυχίας
 }

 Wire.begin(9);              // Επικοινωνία στη διεύθυνση 9 (dummy τιμή)
 Wire.onReceive(receiveEvent); // Όταν δέχεσαι είσοδο εκτέλεσε τη συνάρτηση διαχείρισης receiveEvent

 if (WiFi.status() == WL_NO_MODULE) {
  Serial.println("Communication with WiFi module failed!");
  // don't continue
  while (true);
 }

 String fv = WiFi.firmwareVersion();
 if (fv < WIFI_FIRMWARE_LATEST_VERSION) {
```

```
   Serial.println("Please upgrade the firmware");
  }

  // 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 10 seconds for connection:
   delay(1000);

  server.begin();
  Serial.print("server is at ");
  Serial.println(WiFi.localIP());
}

  printWifiStatus();

}


void loop(){

  counter += 1;

  index = message.indexOf(',');                    // Βρες που είναι το κόμμα στο μήνυμα που έλαβες
  temperature = message.substring(0,index); // από την αρχή μέχρι το κόμμα είναι η θερμοκρασία
  rest = message.substring(index+1);               // μετά το κόμμα μέχρι το τέλος είναι το υπόλοιπο μήνυμα

  index = rest.indexOf(',');                    // που είναι το καινούριο κόμμα
  humidity = rest.substring(0,index);  // από την αρχή μέχρι το κόμμα είναι η υγρασία
  pressure = rest.substring(index+1); // μετά το κόμμα μέχρι το τέλος είναι η πίεση
  if(counter == 30){  // αν ο μετρητής είναι 30 (=30 * 0,5 = 15 sec)
   // Αποθήκευσε τις μετρήσεις στην Κάρτα SD
   String dataString = temperature + "," + humidity + "," + pressure + "\n"; // String μετρήσεων
   File dataFile = SD.open("datalog.txt", FILE_WRITE);
   if (dataFile) {
     dataFile.println(dataString);
     dataFile.close();
     Serial.println("Data Saved to Disk!");
   }else{
     Serial.println("error opening datalog.txt");
   }
  }

if(counter == 120){// αν ο μετρητής είναι 120 (=120 * 0,5 = 60 sec)
// Στείλε το μήνυμα Μετρήσεων στο Twitter
  Serial.println("---------- Trying to tweet ----------");
  String myData = "";
  myData = "Θερμοκρασία: " + temperature + " C\n";
```

```
  Serial.println(myData);
  myData += "Υγρασία: " + humidity + " % \n";
  myData += "Ατμοσφαιρική Πίεση: " + pressure + " mbar";
  myData.toCharArray(msg,140);

  if (twitter.post(msg)) {
    int status = twitter.wait();
    if (status == 200) {
      Serial.println("---------------------------------- OK.");
    } else {
      Serial.print("---------------------------------- TWEET failed : code ");
      Serial.println(status);
    }
  } else {
    Serial.println("---------------------------------- connection failed.");
  }

  counter = 0;
}
// Υποστήριξη Ιστοσελίδας Μετρήσεων
  WiFiClient client = server.available();
  if (client) {
    Serial.println("new client");
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        Serial.write(c);
        if (c == '\n' && currentLineIsBlank) {
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println("Connection: close");
          client.println("Refresh: 5");
          client.println();
          client.println("<!DOCTYPE HTML>");
          client.println("<html>");

          client.println("<body>");

          client.println("<h1>IoT Meteorologic Hub @ 1st Lyceum of Serres</h1>");
          client.print("<h3>Temprature: ");
          client.print(temperature);
          client.print(" C");
          client.println("<br /><br />");

          client.print("Humidity: ");
          client.print(humidity);
          client.print(" %");
          client.println("<br /><br />");

          client.print("Pressure: ");
```

```
        client.print(pressure);
        client.print(" mbar</h3>");
        client.println("<br />");

        client.println("</body>");

        client.println("</html>");
        break;
       }
       if (c == '\n') {
        currentLineIsBlank = true;
       } else if (c != '\r') {
        currentLineIsBlank = false;
       }
     }
   }
   delay(1);
   client.stop();
   Serial.println("client disconnected");
 }

 message = "";
 delay(500);
}

// Συνάρτηση Χειρισμού εισόδου από την i2c
void receiveEvent(int bytes){
 x = Wire.read();    // Αποθήκευση της εισόδου με τη μορφή int
 message += x;      // Σύνθεση μηνύματος Εισόδου
}

void printWifiStatus() {
 Serial.print("SSID: ");
 Serial.println(WiFi.SSID());

 IPAddress ip = WiFi.localIP();
 Serial.print("IP Address: ");
 Serial.println(ip);

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


Meteors