

Data Acquisition System Project

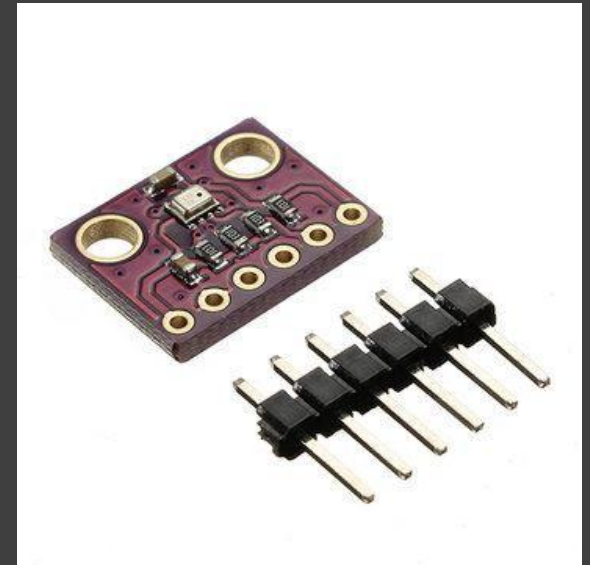
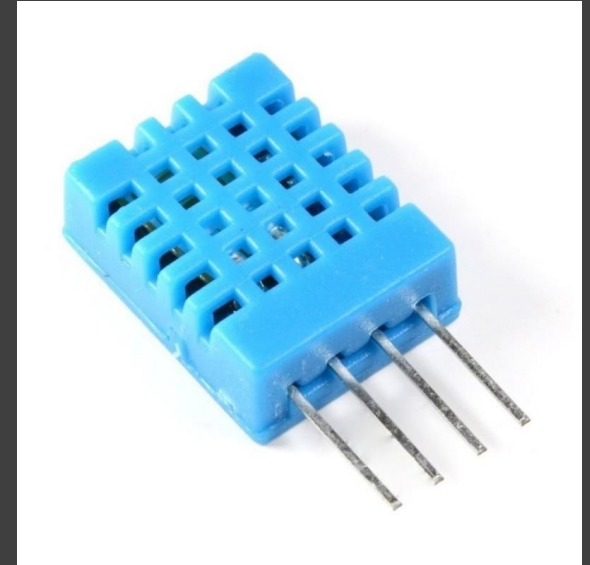
Weather station

Giovanni De Angelis

IST197634

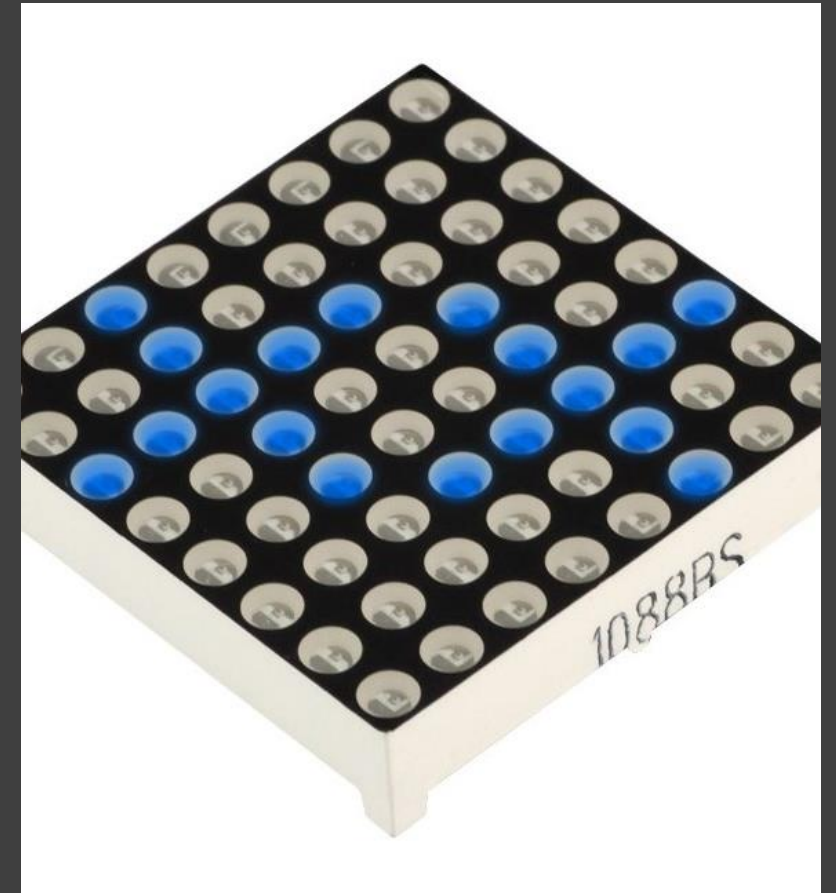
Components: input sensors

- Temperature and humidity sensor DHT11
- Pressure and temperature sensor BMP280



Components: outputs

- Serial monitor
- 8x8 LED matrix

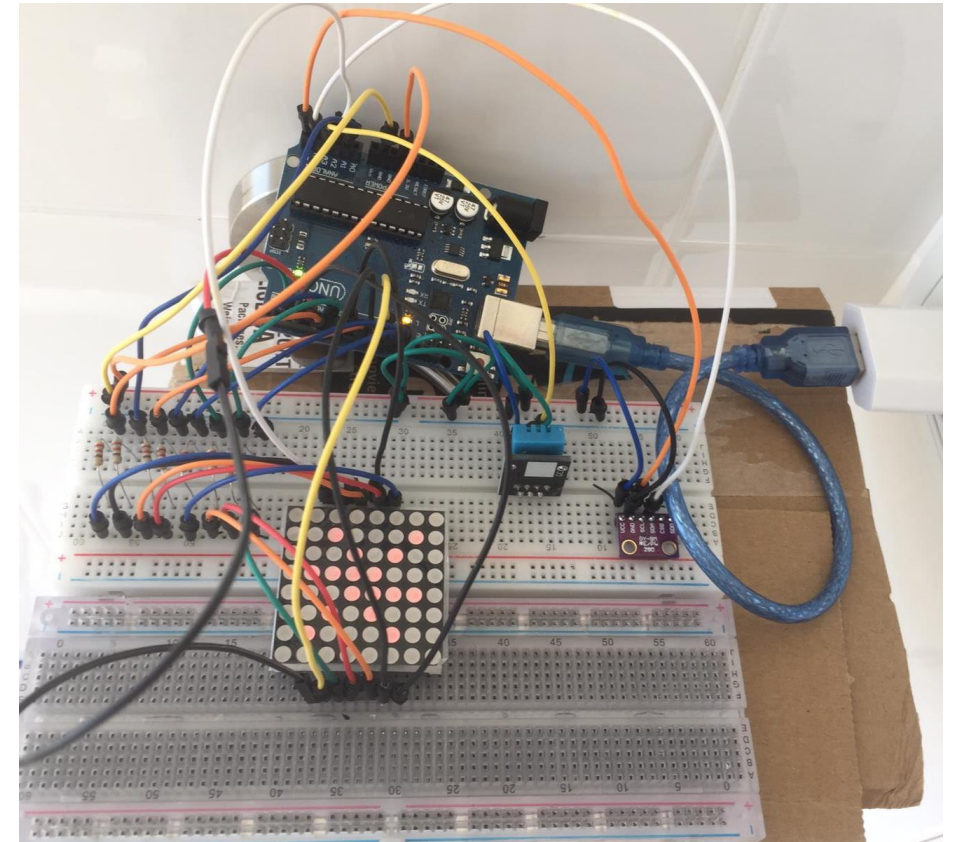


Implementation of the circuit		Jun 8	5
Code acquisition of humidity		Jun 15	2
Code acquisition of Pressure and Temperature		Jun 20	7
Code outputs devices		Jun 22	5
Write down the mathematical model and code it		Jun 27	2
Test the system		Jun 30	9
Fix problems and improve the project			
+ Add			
			30

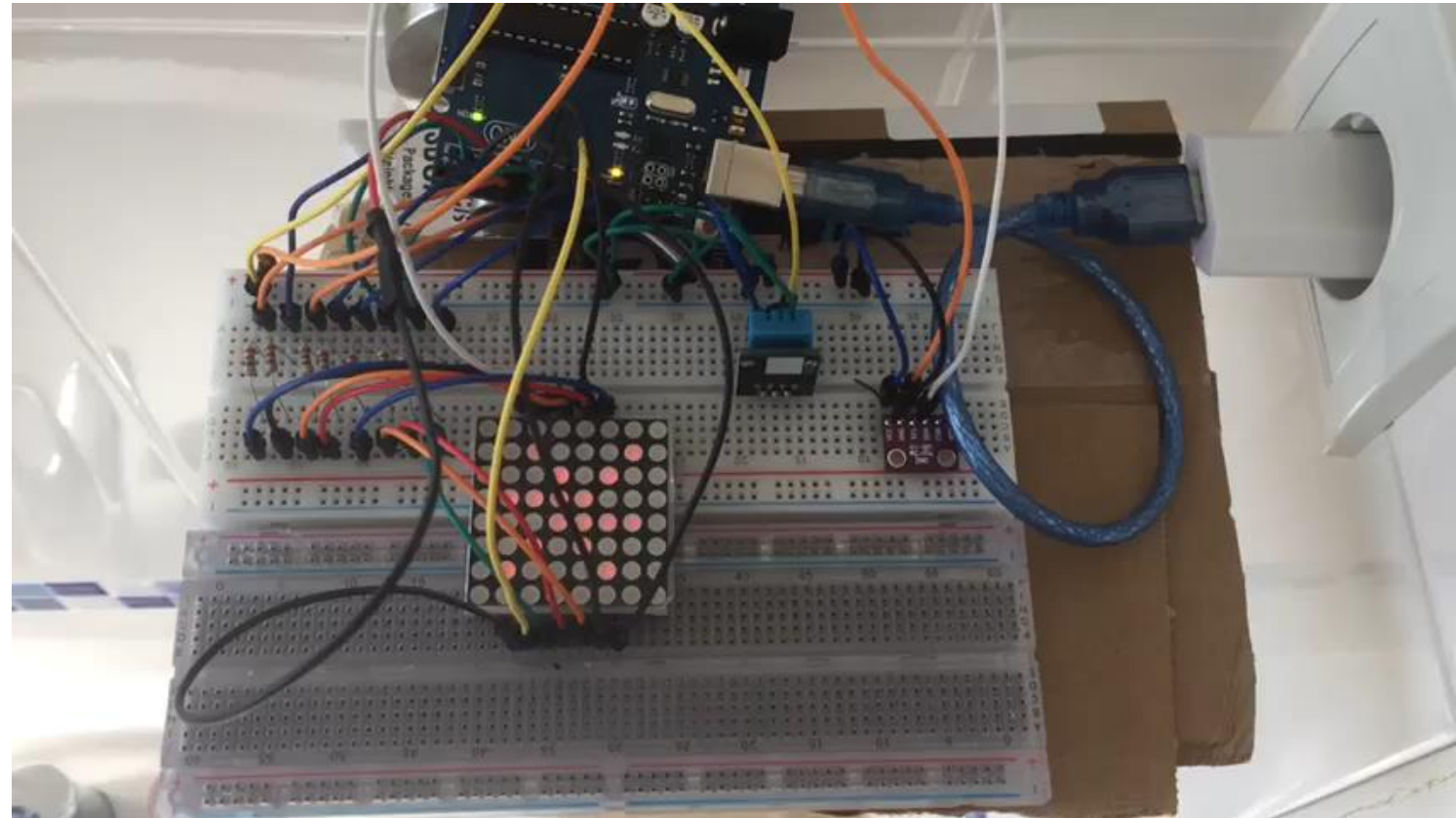
Time schedule

Implementation of the circuit

PIN	goes to
2	> *1st column
3	> *5th row
4	> 3rd column
5	> 4th row
6	> *7th row
7	> *6th row
8	> 3rd row
9	> *4th column
10	> 1st row
11	> 6th column
12	> *7th column
13	> 2nd row
A0	> /
A1	> *2nd column
A2	> 5th column
A3	> DHT
A4	> SCL (BMP)
A5	> SDA(BMP)
* : through a 220 Ω resitor	



Implementation
of the circuitf the
circuit



Model used

The model used to predict the weather is based on humidity(H) and change in pressure(P).

$H > 65\%$ and $\Delta P > 200$ hPa in the last 3 h



probable bad weather in the following 24 hours

$H < 65\%$ regardless the P



probable good weather

Humidity

To do :

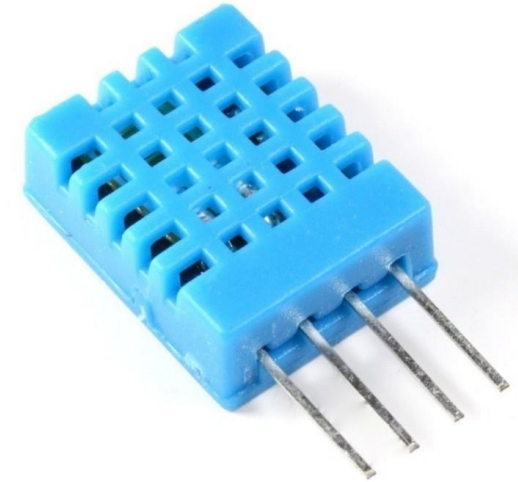
- Code the acquisition
- Test the data acquires

Time schedule :

- 2 hours needed

Problem :

- none



```
#define DHT11_PIN 17

dht DHT;

int chk = DHT.read11(DHT11_PIN);

Serial.print(F("Humidity: "));
Serial.print(DHT.humidity);
Serial.println(" %");|
```


Pressure and Temperature

To do :

- Code the acquisition
- Test the data acquires

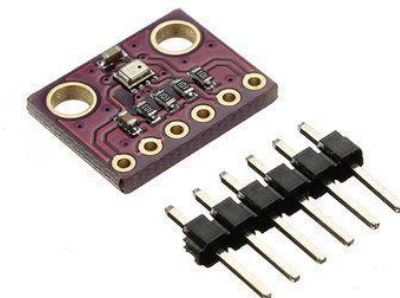
Time schedule :

- 7 hours

Problems :

- It's possible to acquire data only every 2 seconds
- A timer interrupts has been coded in order to get the right acquisition ratio without compromising the other functionality in the loop function
- Total time needed : 10 hours

```
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>
Adafruit_BMP280 bmp;
void setup()
{
  cli(); //stop interrupts
  //set timer0 interrupt at 62Hz
  TCCR0A = 0; // set entire TCCR0A register to 0
  TCCR0B = 0; // same for TCCR0B
  TCNT0 = 0; // initialize counter value to 0
  // set compare match register for 62hz increments
  OCR0A = 252; // = (16*10^6) / (2000*64) - 1 (must be <256)
  // turn on CTC mode
  TCCR0A |= (1 << WGM01);
  // Set CS01 and CS00 bits for 64 prescaler
  TCCR0B |= (1 << CS12) | (1 << CS10);
  // enable timer compare interrupt
  TIMSK0 |= (1 << OCIE0A);
}
ISR(TIMER0_COMPA_vect){
  inc0 = inc0 + 1;
  if (inc0 == 248)
  {
    flagPrint = 1;
    inc0 = 0;
  }
}
if(flagPrint == 1){
  Serial.print(F("Temperature: "));
  Serial.print(bmp.readTemperature());
  Serial.println(" °C ");
  Serial.print(F("Pressure: "));
  Serial.print(bmp.readPressure());
  Serial.println(" Pa ");
}
```



Output devices

To do :

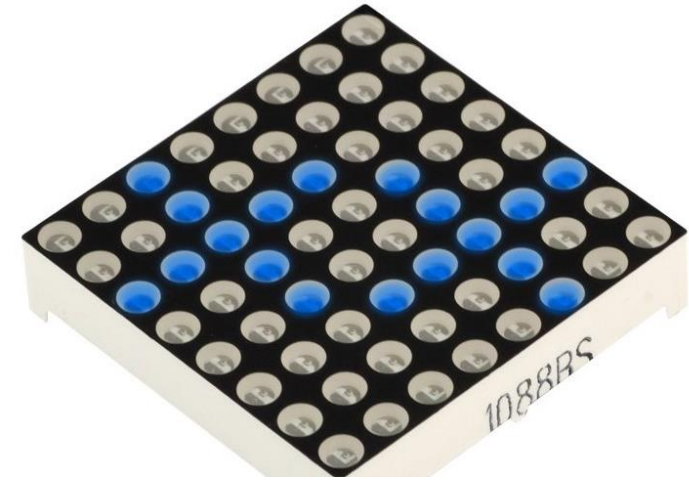
- Code the LED matrix
- Code the serial communications

Time schedule :

- 5 hours

Problems :

- Too many pins requested from the matrix in the board: two pins have been gained making the matrix a 7x7 LED matrix. Fillers number(99) have been used in code in order to keep the same scheme of coding of an 8x8 LED matrix
- It needs a very short time of refresh : time interrupts have been used for the other commands in the loop in order not to waste time in the loop providing the fastest refresh possible to the matrix
- Time needed: 7 hours



```
#define MatrixVector 65
#define Ncolrow 8
#define Npin 17
int pins[Npin]= { 99, 5, 4, 3, 2, 99, 15, 16, 99, 13, 12, 11, 10, 9, 8, 7, 6};
int cols[Ncolrow] = {pins[13], pins[10], pins[15], pins[9], pins[4], pins[16], pins[6], pins[1]
int rows[Ncolrow] = {pins[8], pins[7], pins[3], pins[14], pins[2], pins[12], pins[11], pins[5]}
int vect[MatrixVector];
> int carSun [MatrixVector] = { ...
> int carCloud [MatrixVector] = { // cloud matrix...
void setup(){
    for (int i = 1; i <= (Npin-1); i++){ // I'm defining at output all the ports in the array
        pinMode(pins[i], OUTPUT);}
    for (int i = 1; i <= Ncolrow; i++) { // I'm switching off all the pin
        digitalWrite(cols[i - 1], LOW);
        digitalWrite(rows[i - 1], LOW);} }
void loop(){
    for (r1 = 1; r1 <= Ncolrow; r1++){
        for (c1 = 1; c1 <= Ncolrow; c1++){
            if (vect [(r1-1)*Ncolrow + c1] == 1){ // BYTE = 1 (led to switch on)
                digitalWrite (rows [r1-1], HIGH); // active row (anod)
                digitalWrite (cols [c1-1], LOW);} // disactive (catod) and switch on the led on the
            digitalWrite (rows [r1-1], LOW); // disable the row (anod)
            digitalWrite (cols [c1-1], HIGH); // enable the column (catod) and switch off the led
        }
    }
}
```

Model of prediction

To do :

- Write down the model
- Code it

Time schedule :

- 2 hours

Problems :

- being sure that the code responds in the right way at each iteration taking in account also the previous output (ex. Previous output=bad weather , at the next iteration the difference in pressure is not satisfied any more but this doesn't mean that the weather is going to be good)
- 4 hours

```
if (flagMemory == 1)
{
    hum = DHT.humidity;
    press[add] = bmp.readPressure();
    if (add >= 3)
    {
        if (hum >= HumEdge)
        {
            if ((press[add-3]-press[add]) >= PresDiffEdge || flagBadW == 1 )
            {
                flagMatrix = 1;
                flagBadW = 1;
            }else
            {
                flagMatrix = 0;
                flagBadW = 0;
            }
        }
    }
    if (add == 71)
    {
        add = 0;
    }else
    {
        add = add + 1;
    }
    flagMemory = 0;
}
```

Test of the system

To do :

- test the system in different humidity and pressure situation
- Test the efficiency of the output

Time schedule :

- 9 hours

Problems :

- Time and different atmospheric situation are necessary to make a real test of the project : different humidity situations have been got artificially while the pressure is much more difficult to be handled with
- Time requested : 48 hours

Possible improvement

Code the EEPROM memory in order to save the parameters that can be useful for other analysis:

- Problem : it's possible to save just integers and number smaller the 255
- Possible solution : use a union and save the number in different byte
- Cons : comparison for the estimation of the weather more tricky

Implement a real time clock in order to keep count of the time in which the board is switched off :

- Problem : it's needed a welder to prepare the hardware to use
- Cons : even with a real time clock , if the board has been switched off for a long time comparing the pressure data in the memory with the new ones doesn't make any sense: the board has to start to work as in the case of first use