TP M2M

November 12, 2018

Thomas COUCHOUD
thomas.couchoud@etu.univ-tours.fr
Victor COLEAU
victor.coleau@etu.univ-tours.fr

Chapter 1

Prise en main

1.1 Faire clignoter une LED

Controller la LED ne pose pas de grand problème. Il faut juste bien penser à initialiser les différents éléments (Serial, port de la LED).

Le code utilisé est disponible en Appendix A.

1.2 LED RGB

Afin de pouvoir utiliser la LED RGB, nous utilisons la librairie ChainableLED. Cette dernière nous permet de créer des objets ChainableLED. Pour l'initialiser nous donnons la broche de l'horloge, la broche des données puis enfin le nombre de LEDs dans la chaine. Par exemple si nous branchons la LED en D3, nous initialisons avec led(3,4,1).

Une fois l'objet créé, il ne faut pas oublier de l'initialiser dans la méthode init grâce à led.init().

Dans le code nous pouvons ensuite utiliser les méthodes fournies:

- void setColorRGB(led, red, green, blue);
- void setColorHSB(led, hue, saturation, brightness);

Le paramètre led correspond à l'indice de la LED dans la chaine.

Concernant la boucle loop, cette dernière effectue les opérations suivantes:

- Récupère la valeur du potentiomètre (analogRead)
- Si cette valeur est plus grande qu'un seuil, on allume une des LED, sinon on l'éteint
- On map la valeur du potentiomètre entre 0 et 1 puis allumons la LED RGB avec comme intensité la valeur mappé
- On attend 20ms

Le code est disponible en Appendix B.

1.3 Température

De la même manière pour la température, nous utilisons la librairie DHT. Nous pouvons ensuite créer un objet DHT avec comme paramètres le port sur lequel est branché le capteur ainsi que le type du capteur utilisé.

Il fait ensuite initialiser notre objet dans la fonction init grace à dht.begin().

Enfin nous pouvons obtenir la température et l'humidité grâce aux fonction dht.readHumidity() et dht.readTemperature().

Notre code fait exactement les mêmes étapes qu'avec le potentiomètre mais map la température entre 20 et 50 degrés vert une valeur entre 0 et 255 pour controller la LED en RGB. Voir Appendix C.

1.4 LCD

Encore une fois pour utiliser l'écran LCD nous utilisons une librairie: rgb_lcd. L'objet à créer est un bjet rgb_lcd. Nous l'initialisons dans le setup grâce à lcd.begin(nombre de colonnes, nombre de lignes). Ensuite nous définissons une couleur par défaut.

Dans la fonction loop, nous récupérons la température et humidité puis l'affichons sur l'écran. Pour cela:

- On place le curseur en 0,0 grâce à set cursor

- On écrit "T :"
- On place le curseur en 4,0
- On écris la température
- On place le curseur en 0,1
- On écrit "H :"
- On place le curseur en 4,1
- On écris l'humidité
- On place le curseur en 15,1
- On écris "%"

De plus nous changeons la couleur de l'écran en fonction de la température et humidité grâce à lcd.setColor(rouge, vert, bleu).

Code disponible en Appendix D.

Chapter 2

TP1

2.1 Adresse I2C

Afin de récupérer l'adresse I2C du capteur, nous utilisons le I2CScanner proposé ici. Grâce à ce code, nous avons pu identifier que le baromètre à pour adresse 0x76.

2.2 Registre

- 0x0D: ID, contient l'ID du périphérique.
- 0xE0: Reset, si on écrit 0xB6, le périphérique est réinitialisé.
- 0xF2: ctrl_hum, permet de définir les options de mesure de l'humidité. Le registre devient effectif après une écriture dans ctrl_meas.
- 0xF3: status, contient 2 bits indiquant le status du périphérique.
  - Bit 3: mis à 1 quand une conversion est en cours, et 0 quand les résultats ont étés transférés.
  - Bit 0: mis à 1 quand des données NVM sont copiés dans l'image du registre et 0 quand le transfert est fini.
- 0xF4 ctrl_meas: enregistre les données de capture de pression et température.
- 0xF5 config: définis des options supplémentaires.
- 0xF7...0xF9 press (_msb, _lsb, _xlsb): contient les valeurs non modifiées des mesures de pression.
- 0xFA...0xFC temp (_msb, _lsb, _xlsb): pareil mais pour la température.
- 0xFD...0xFE hum (_msb, _lsb): pareil mais pour l'humidité.

2.3 Librairie

La librairie comporte un problème, le port I2C utilisé est 0x77 et non pas 0x76. Nous avons donc du changer ce paramètre.
On commence par déclarer un objet Adafruit_BME280 puis on l'initialise avec bme.begin().
Afin d'obtenir les mesures physiques, nous avons accès à:

- bme.readTemperature()
- bme.readHumidity()
- bme.readPressure()
- bme.readAltitude(SEALEVELPRESSURE_HPA) où le paramètre correspond à la pression à l'altitude 0.

Puis on affiche les données sur l'écran LCD. Le code est disponible en Appendix E.

Chapter 3

TP2

Avec utilisation de la librairie AdafruitBME280, notre programme occupait 12300 octets.
Après n'avoir gardé que l'utile de la librairie nous arrivons à ne plus utiliser que 10616 octets soit un gain de 14%.
Le code est disponible en Appendix F.

Figure 3.1 – Faible température, faible pression

Figure 3.2 – Température élevée, pression élevée

Appendix A

Faire clignoter une LED

Listing A.1 – led.ino

```
#define led 13 // Constante representant le pin de la LED

// Appelée une fois au démarage
void setup() {
  Serial.begin(9600); // Def le débit de transmition de données
  pinMode(led, OUTPUT); // Def que le pin 13 sera une sortie
  digitalWrite(led, LOW); // Ecrit sur un pin digital (0 ou 1)
  Serial.println("Lancement de l'app"); // Log
}

// Exécuter en boucle
void loop() {
  digitalWrite(led, HIGH); // Allume la LED
  Serial.println("LED allumée"); // Log
  delay(1000); // Attend 1s
  digitalWrite(led, LOW); // Eteind la LED
  Serial.println("LED éteinte"); // Log
  delay(800); // Attend 0.8s
}
```

Appendix B

LED RGB

Listing B.1 – led_rgb_pot.ino

```
#include <ChainableLED.h>

#define led1 13
#define led2 3
#define pot 1
#define To 500

int p = 0;
ChainableLED led(3, 4, 1);

void setup() {
  Serial.begin(9600);
  pinMode(led1, OUTPUT);
  pinMode(pot, INPUT);
  led.init();
}

void loop() {
  p = analogRead(pot);
  if(p > To){
    digitalWrite(led1, HIGH);
  }
  else{
    digitalWrite(led1, LOW);
  }
  float f = p/1023.0;
  led2.setColorHSB(0, 0.5, 1, f);
  Serial.println(f);
  delay(100);
}
```

Appendix C

Température

Listing C.1 – led_rgb_temp.ino

```
#include <ChainableLED.h>
#include <DHT.h>

#define led1 13
#define led2 3
#define pot A0
#define To 500

int p = 0;
ChainableLED led(3, 4, 1);
DHT dht(pot, DHT22);

void setup() {
  Serial.begin(9600);
  pinMode(led1, OUTPUT);
  pinMode(pot, INPUT);
  led.init();
  dht.begin();
}

void loop() {
  float humidity = dht.readHumidity();
  float temp = dht.readTemperature();
  p = temp;
  Serial.print("H : ");
  Serial.print(humidity);
  Serial.print("     T : ");
  Serial.println(temp);
  if(p > To){
    digitalWrite(led1, HIGH);
  }
  else{
    digitalWrite(led1, LOW);
  }
  float f = 255*(p-20.0)/(50.0-20.0);

  led.setColorRGB(0, f, 255-f, humidity / 100.0 * 255.0);
  Serial.println(f);
  delay(100);
}
```

Appendix D

LCD

Listing D.1 – lcd.ino

```
#include <DHT.h>
#include <Wire.h>
#include "rgb_lcd.h"

rgb_lcd lcd;

#define pot A0

int p = 0;
DHT dht(pot, DHT22);

void setup() {
  Serial.begin(9600);
  dht.begin();
  lcd.begin(16, 2);
  lcd.setRGB(255, 255, 255);
}

void loop() {
  float humidity = dht.readHumidity();
  float temp = dht.readTemperature();
  p = temp;
  Serial.print("H : ");
  Serial.print(humidity);
  Serial.print("     T : ");
  Serial.println(temp);
  float f = 255*(p-20.0)/(50.0-20.0);

  lcd.setRGB(f, 255-f, humidity / 100.0 * 255.0);

  lcd.setCursor(0, 0);
  lcd.print("T : ");
  lcd.setCursor(4, 0);
  lcd.print(temp);
  lcd.setCursor(0, 1);
  lcd.print("H : ");
  lcd.setCursor(4, 1);
  lcd.print(humidity);
  lcd.setCursor(10, 1);
  lcd.print("%");

  delay(1000);
}
```

Appendix E

Baromètre

Listing E.1 – bme.ino

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#include "rgb_lcd.h"

#define SEALEVELPRESSURE_HPA (1013.25)

Adafruit_BME280 bme; // I2C
rgb_lcd lcd;

unsigned long delayTime;

void setup() {
    Serial.begin(9600);
    bme.begin();
    lcd.begin(16, 2);
    lcd.setRGB(255, 255, 255);
    delayTime = 1000;
}


void loop() {

    float t = bme.readTemperature();
    Serial.print("Temperature = ");
    Serial.print(t);
    Serial.println(" *C");

    float h = bme.readHumidity();
    Serial.print("Humidity = ");
    Serial.print(h);
    Serial.println(" %");

    float p = bme.readPressure() / 100.0F;
    Serial.print("Pressure = ");
    Serial.print(p);
    Serial.println(" hPa");

    float a = bme.readAltitude(SEALEVELPRESSURE_HPA);
    Serial.print("Approx. Altitude = ");
    Serial.print(a);
    Serial.println(" m");

    Serial.println();

    float f = 255*(t-20.0)/(50.0-20.0);

    lcd.setRGB(f, 255-f, h / 100.0 * 255.0);

    lcd.setCursor(0, 0);
```

```
51      lcd.print("T:");
52      lcd.setCursor(2, 0);
53      lcd.print(t);
54
55      lcd.setCursor(8, 0);
56      lcd.print("P:");
57      lcd.setCursor(10, 0);
58      lcd.print(p);
59
60      lcd.setCursor(0, 1);
61      lcd.print("H:");
62      lcd.setCursor(2, 1);
63      lcd.print(h);
64
65      lcd.setCursor(8, 1);
66      lcd.print("A:");
67      lcd.setCursor(10, 1);
68      lcd.print(a);
69
70      delay(delayTime);
71  }
```

Appendix F

Baromètre avec librairie minimale

Listing F.1 – bme.ino

```
#include <Wire.h>
#include "rgb_lcd.h"

rgb_lcd lcd;

#define _SEEED_H_

#include <Arduino.h>

#define ADDRESS 0x76

#define REG_DIG_T1  0x88
#define REG_DIG_T2  0x8A
#define REG_DIG_T3  0x8C

#define REG_DIG_P1  0x8E
#define REG_DIG_P2  0x90
#define REG_DIG_P3  0x92
#define REG_DIG_P4  0x94
#define REG_DIG_P5  0x96
#define REG_DIG_P6  0x98
#define REG_DIG_P7  0x9A
#define REG_DIG_P8  0x9C
#define REG_DIG_P9  0x9E

#define REG_DIG_H1  0xA1
#define REG_DIG_H2  0xE1
#define REG_DIG_H3  0xE3
#define REG_DIG_H4  0xE4
#define REG_DIG_H5  0xE5
#define REG_DIG_H6  0xE7

#define REG_CHIPID       0xD0
#define REG_VERSION      0xD1
#define REG_SOFTRESET    0xE0

#define REG_CAL26        0xE1

#define REG_CONTROLHUMID 0xF2
#define REG_CONTROL      0xF4
#define REG_CONFIG       0xF5
#define REG_PRESSUREDATA 0xF7
#define REG_TEMPDATA     0xFA
#define REG_HUMIDITYDATA 0xFD

#define _INVALID_DATA 0


#define _address 0x76
#define SEALEVELPRESSURE_HPA (1013.25)
```

```

bool isTransport_OK;
uint16_t dig_T1;
int16_t dig_T2;
int16_t dig_T3;

uint16_t dig_P1;
int16_t dig_P2;
int16_t dig_P3;
int16_t dig_P4;
int16_t dig_P5;
int16_t dig_P6;
int16_t dig_P7;
int16_t dig_P8;
int16_t dig_P9;

uint8_t dig_H1;
int16_t dig_H2;
uint8_t dig_H3;
int16_t dig_H4;
int16_t dig_H5;
int8_t dig_H6;

int32_t t_fine;

void init2()
{
  Wire.begin();

  if(Read8(REG_CHIPID) != 0x60)
    return ;

  dig_T1 = Read16LE(REG_DIG_T1);
  dig_T2 = ReadS16LE(REG_DIG_T2);
  dig_T3 = ReadS16LE(REG_DIG_T3);

  dig_P1 = Read16LE(REG_DIG_P1);
  dig_P2 = ReadS16LE(REG_DIG_P2);
  dig_P3 = ReadS16LE(REG_DIG_P3);
  dig_P4 = ReadS16LE(REG_DIG_P4);
  dig_P5 = ReadS16LE(REG_DIG_P5);
  dig_P6 = ReadS16LE(REG_DIG_P6);
  dig_P7 = ReadS16LE(REG_DIG_P7);
  dig_P8 = ReadS16LE(REG_DIG_P8);
  dig_P9 = ReadS16LE(REG_DIG_P9);

  dig_H1 = Read8(REG_DIG_H1);
  dig_H2 = Read16LE(REG_DIG_H2);
  dig_H3 = Read8(REG_DIG_H3);
  dig_H4 = (Read8(REG_DIG_H4) << 4) | (0x0F & Read8(REG_DIG_H4 + 1));
  dig_H5 = (Read8(REG_DIG_H5 + 1) << 4) | (0x0F & Read8(REG_DIG_H5) >> 4);
  dig_H6 = (int8_t)Read8(REG_DIG_H6);

  writeRegister(REG_CONTROLHUMID, 0x05); //Choose 16X oversampling
  writeRegister(REG_CONTROL, 0xB7); //Choose 16X oversampling

}

float getTemperature(void)
{
  int32_t var1, var2;

  int32_t adc_T = Read24(REG_TEMPDATA);

  if(!isTransport_OK) {
    return _INVALID_DATA;
  }
  adc_T >>= 4;
  var1 = (((adc_T >> 3) - ((int32_t)(dig_T1 << 1))) *
```

```
    ((int32_t)dig_T2)) >> 11;

  var2 = (((((adc_T >> 4) - ((int32_t)dig_T1)) *
    ((adc_T >> 4) - ((int32_t)dig_T1))) >> 12) *
    ((int32_t)dig_T3)) >> 14;

  t_fine = var1 + var2;
  float T = (t_fine * 5 + 128) >> 8;
  return T/100;
}

uint32_t getPressure(void)
{
  int64_t var1, var2, p;


  getTemperature();

  if(!isTransport_OK) {
    return _INVALID_DATA;
  }

  int32_t adc_P = Read24(REG_PRESSUREDATA);
  adc_P >>= 4;

  var1 = ((int64_t)t_fine) - 128000;
  var2 = var1 * var1 * (int64_t)dig_P6;
  var2 = var2 + ((var1*(int64_t)dig_P5)<<17);
  var2 = var2 + (((int64_t)dig_P4)<<35);
  var1 = ((var1 * var1 * (int64_t)dig_P3)>>8) + ((var1 * (int64_t)dig_P2)<<12);
  var1 = (((((int64_t)1)<<47)+var1))*((int64_t)dig_P1)>>33;
  if (var1 == 0) return 0;
  p = 1048576-adc_P;
  p = (((p<<31)-var2)*3125)/var1;
  var1 = (((int64_t)dig_P9) * (p>>13) * (p>>13)) >> 25;
  var2 = (((int64_t)dig_P8) * p) >> 19;
  p = ((p + var1 + var2) >> 8) + (((int64_t)dig_P7)<<4);
  return (uint32_t)p/256;
}

uint32_t getHumidity(void)
{
  int32_t v_x1_u32r, adc_H;

  getTemperature();
  if(!isTransport_OK) {
    return _INVALID_DATA;
  }

  adc_H = Read16(REG_HUMIDITYDATA);

  v_x1_u32r = (t_fine - ((int32_t)76800));
  v_x1_u32r = (((((adc_H << 14) - (((int32_t)dig_H4) << 20) - (((int32_t)dig_H5) * v_x1_u32r)) +
      ((int32_t)16384)) >> 15) * (((((((v_x1_u32r * ((int32_t)dig_H6)) >> 10) * (((v_x1_u32r *
      ((int32_t)dig_H3)) >> 11) + ((int32_t)32768))) >> 10) + ((int32_t)2097152)) * ((int32_t)dig_H2) +
      8192) >> 14));
  v_x1_u32r = (v_x1_u32r - (((((v_x1_u32r >> 15) * (v_x1_u32r >> 15)) >> 7) * ((int32_t)dig_H1)) >> 4));
  v_x1_u32r = (v_x1_u32r < 0 ? 0 : v_x1_u32r);
  v_x1_u32r = (v_x1_u32r > 419430400 ? 419430400 : v_x1_u32r);
  return (uint32_t)(v_x1_u32r>>12)/1024.0;
}

float calcAltitude(float pressure)
{
  float A = pressure/SEALEVELPRESSURE_HPA;
  float B = 1/5.25588;
  float C = pow(A,B);
  C = 1.0 - C;
  C = C /0.0000225577;
```

```
  return C;
}

uint8_t Read8(uint8_t reg)
{
  Wire.beginTransmission(_address);
  Wire.write(reg);
  Wire.endTransmission();

  Wire.requestFrom(_address, 1);

  if(Wire.available() < 1) {
    isTransport_OK = false;
    return 0;
  } else {
    isTransport_OK = true;
  }

  return Wire.read();
}

uint16_t Read16(uint8_t reg)
{
  uint8_t msb, lsb;

  Wire.beginTransmission(_address);
  Wire.write(reg);
  Wire.endTransmission();

  Wire.requestFrom(_address, 2);
  // return 0 if slave didn't response
  if(Wire.available() < 2) {
    isTransport_OK = false;
    return 0;
  } else {
    isTransport_OK = true;
  }
  msb = Wire.read();
  lsb = Wire.read();

  return (uint16_t) msb<<8 | lsb;
}

uint16_t Read16LE(uint8_t reg)
{
  uint16_t data = Read16(reg);
  return (data >> 8) | (data << 8);
}

int16_t ReadS16(uint8_t reg)
{
  return (int16_t)Read16(reg);
}

int16_t ReadS16LE(uint8_t reg)
{
  return (int16_t)Read16LE(reg);
}

uint32_t Read24(uint8_t reg)
{
  uint32_t data;

  Wire.beginTransmission(_address);
  Wire.write(reg);
  Wire.endTransmission();

  Wire.requestFrom(_address, 3);
  // return 0 if slave didn't response
```

```
  if(Wire.available() < 3) {
    isTransport_OK = false;
    return 0;
  } else {
    isTransport_OK = true;
  }
  data = Wire.read();
  data <<= 8;
  data |= Wire.read();
  data <<= 8;
  data |= Wire.read();

  return data;
}

void writeRegister(uint8_t reg, uint8_t val)
{
  Wire.beginTransmission(_address);
  Wire.write(reg);
  Wire.write(val);
  Wire.endTransmission();
}

void setup() {
    Serial.begin(9600);

    init2();

    lcd.begin(16, 2);
    lcd.setRGB(255, 255, 255);
}


void loop() {
    float t = getTemperature();
    Serial.print("Temperature = ");
    Serial.print(t);
    Serial.println(" *C");

    float h = getHumidity();
    Serial.print("Humidity = ");
    Serial.print(h);
    Serial.println(" %");

    float p = getPressure() / 100.0;
    Serial.print("Pressure = ");
    Serial.print(p);
    Serial.println(" hPa");

    float a = calcAltitude(p);
    Serial.print("Approx. Altitude = ");
    Serial.print(a);
    Serial.println(" m");

    Serial.println();

    float f = 255*(t-20.0)/(15.0);
    lcd.setRGB(f, 255-f, h / 100.0 * 255.0);

    lcd.setCursor(0, 0);
    lcd.print("T:");
    lcd.setCursor(2, 0);
    lcd.print(t);

    lcd.setCursor(8, 0);
    lcd.print("P:");
    lcd.setCursor(10, 0);
    lcd.print(p);

```

```
    lcd.setCursor(0, 1);
    lcd.print("H:");
    lcd.setCursor(2, 1);
    lcd.print(h);

    lcd.setCursor(8, 1);
    lcd.print("A:");
    lcd.setCursor(10, 1);
    lcd.print(a);

    delay(1000);
}
```