Table des matières

1) Mise en œuvre d'une photorésistance

1) Connaissance du capteur

a) J'identifie le composant dans le kit :

Il n'a pas de référence

On a pas beaucoup de données sur la capteur. Il y a une toute petite fiche technique

b)

- obscurité la plus totale : 800 kohms

- obscurité partielle : 60 kohms

- luminosité ambiante : 3 komhs

- éclairé par une lampe : 26 ohms

c) Plus on monte en obscurité et plus la résistance augmente et elle augmente la tension

2) Mise en œuvre

a) U=RI => I=U/R

I=3,3/15000=0,22mA

La valeur choisie pour R permet de limiter le courant à 0,22 mA

b) Je réalise le montage et je mesure la tension U :

- En luminosité ambiante : 0,5V

- Dans l'obscurité : 3,2V

c) Les valeurs étaient prévisibles compte tenu des mesures précédentes de résistances

3) Exploitation

a) Pour procéder je vais utiliser le programme exemple AnalogReadSerial qui va me donner la valeur de N et ensuite je fais une conversion de N en Volts que je vais ensuite utiliser pour définir une seuil entre le jour et la nuit que je vais afficher en fonction de la tension :

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

void loop() {
  int sensorValue = analogRead(A0);  //Je lis la valeur de N
  float tension = sensorValue * (3.3/3600); //Je convertis en V
  if (tension > 1.2){
    Serial.println("Il fait nuit"); //Si V > 1,2 écrire
  }
  else {
    Serial.println("Il fait jour"); //Sinon écrire l'autre
  }
  delay(1000);                      //Attente de 1s
}
```

Voici le résultat :

```
Il fait jour
Il fait jour
Il fait nuit
Il fait nuit
```

b) Pour cette étape je vais rajouter une broche de sortie digitale pour y brancher quelque chose. Ici on mettra un LED pour vérifier qu'elle s'allume quand il fait nuit et qu'elle s'éteint quand il fait jour :

```
const int outpin = 13;
void setup() {
  Serial.begin(9600);
  pinMode(outpin, OUTPUT);
}

void loop() {
  int sensorValue = analogRead(A0);  //Je lis la valeur de N
  float tension = sensorValue * (3.3/3600); //Je convertis en V
  if (tension > 1.2){
    Serial.println("Stores baissés"); //Si V > 1,2 écrire
    digitalWrite(outpin,HIGH);
  }
  else {
    Serial.println("Stores relevés"); //Sinon écrire l'autre
    digitalWrite(outpin,LOW);
  }
  delay(1000);                        //Attente de 1s
}
```

c) Je test sur d'autres ports numériques pour vérifier si je peux ajouter d'autres sorties numériques :

```
const int outpin = 32;
```

En faisant des tests je peux voir que tous les ports ADC peuvent servir de sorties numériques donc les ports GPIO 0, 2, 4, 12, 13, 14, 15, 25, 26, 27, 32, 33, 34, 35, 36 et 39

4) Mise en œuvre d'un écran I2C LCD

a) Notre écran LCD communique via le bus i2c (4 broches) :

- VCC (5V)

- GND

- SCL (Serial Clock Line)

- SDA (Serial Data Line)

On a juste à connecter ces 4 broches sur les bons ports de notre ESP et ensuite tout se passe dans la partie programme

J'installe la librairies donnée dans le sujet du TP (c'est un github).

Pour l'installer je la télécharge puis dans le dossier "libraries" d'arduino je crée un dossier nommé "LiquidCrystal_I2C" et dedans je place tous les fichiers téléchargés

b) Maintenant dans les exemples je trouve la librairie LiquidCrystal_I2C et je prend le programme "Hello World" :

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Set the LCD address to 0x27 for a 16 chars and 2 line display
LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup()
{
                      // initialize the LCD
                          lcd.begin();

          // Turn on the blacklight and print a message.
                         lcd.backlight();
                   lcd.print("Hello, world!");
}

void loop()
{
                      // Do nothing here...
}
```

c) Dans le programme on initialise le LCD sur 16 caractères par ligne et sur 2 lignes avec l'adresse d'affichage sur les ports SCL, SDA (ici 0x27) ce qui correspond aux GPIO 21 et 22

Une fois fait on initialise la connexion et on affiche "Hello World"

d) Si le message dépasse 16 caractère il sera coupé jusqu'à arriver sur la 2ème ligne qui sera ensuite aussi coupé

e) L'adresse en héxadécimal est 0x27. En binaire elle correspond à 0010 0111

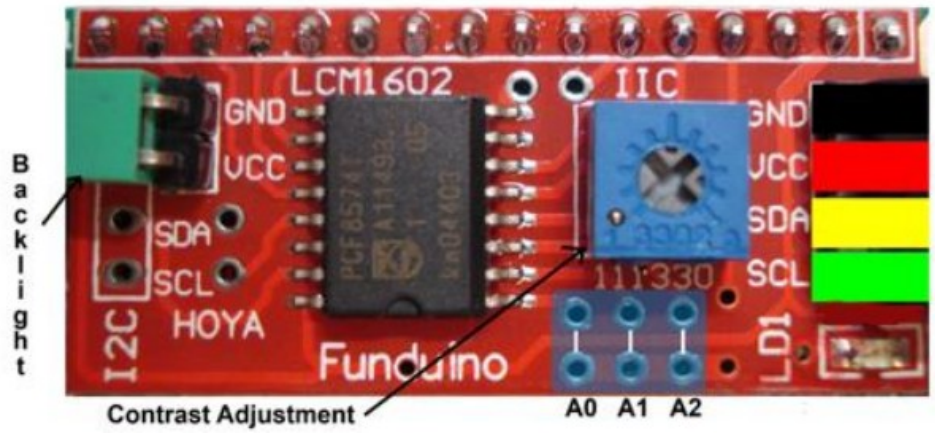
f) Cette adresse peut être modifiée :

Addressing:

A0	A1	A2	Address
Open	Open	Open	0x27
Jumper	Open	Open	0x26
Open	Jumper	Open	0x25
Jumper	Jumper	Open	0x24
Open	Open	Jumper	0x23
Jumper	Open	Jumper	0x22
Open	Jumper	Jumper	0x21
Jumper	Jumper	Jumper	0x20

Pour la modifier il faudrait la changer dans le programme et réaliser la soudure pour les connexions sur les ports A0, A1 ou A2 :

Pinout Diagram:



g) Nous sommes limité à rester sur les ports SCL et SDA de notre ESP

h) Je crée un nouveau programme qui me permet toujours de détecter la luminosité et de l'afficher dans le moniteur série et aussi d'afficher le texte "Il fait jour" et "Il fait sombre" sur mon écran lcd :

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);
const int outpin = 32;

void setup()
{
  lcd.begin();
  lcd.backlight();

  Serial.begin(9600);
  pinMode(outpin, OUTPUT);
}

void loop()
{
  int sensorValue = analogRead(A0);  //Je lis la valeur de N
  float tension = sensorValue * (3.3/3600); //Je convertis en V
  if (tension > 1.2){
    Serial.println("Il fait sombre");
```

```
    lcd.print("Il fait sombre"); //Si V > 1,2 écrire
    digitalWrite(outpin,HIGH);
  }
  else {
    Serial.println("Il fait jour");
    lcd.print("Il fait jour"); //Sinon écrire l'autre
    digitalWrite(outpin,LOW);
  }
  delay(500);
  lcd.clear();
}
```

i) J'améliore mon programme comme demandé dans le TP :

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);
const int outpin = 32;

void setup()
{
  lcd.begin();
  lcd.backlight();

  Serial.begin(9600);
  pinMode(outpin, OUTPUT);
}

void loop()
{
  int sensorValue = analogRead(A0);  //Je lis la valeur de N
  float tension = sensorValue * (3.3/3600); //Je convertis en V
  if (tension > 1.2){
    Serial.println("Il fait sombre");
    lcd.setCursor(0,1);
    lcd.print("Il fait sombre"); //Si V > 1,2 écrire
    digitalWrite(outpin,HIGH);
  }
  else {
    Serial.println("Il fait jour");
    lcd.setCursor(0,0)
    lcd.print("Il fait jour"); //Sinon écrire l'autre
    digitalWrite(outpin,LOW);
  }
  delay(500);
  lcd.clear();
```

```
}
```

2) Mise en œuvre d'un capteur de distance à ultrason

a) Le capteur envoi une onde avec son émetteur (onde généré avec la tension de l'ESP) et attend qu'elle revienne sur son récepteur (onde convertie en tension). Le capteur s'aide du temps et de la vitesse pour trouver la distance entre le capteur et l'obstacle

Fiche technique du capteur : https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf

b) Le capteur connaît déjà la vitesse du son dans l'air : 340m/s

Il se sert de l'aller retour des ondes pour déterminer le temps de parcours et ensuite donner la distance entre le capteur et un obstacle

$$d = \frac{(v \times t)}{2}$$

$$d(cm) = \frac{340 \times \frac{t(us)}{10^6}}{2} \times 100$$

c) Image récupérée de la datasheet :

Working Voltage	**DC 5 V**
Working Current	**15mA**
Working Frequency	**40Hz**
Max Range	**4m**
Min Range	**2cm**
MeasuringAngle	**15 degree**
Trigger Input Signal	**10uS TTL pulse**
Echo Output Signal	**Input TTL lever signal and the range in proportion**
Dimension	**45*20*15mm**

- Plage de mesure de 2cm à 4m

- La durée maximale de l'impulsion envoyée est de $t = \frac{d \times 2}{v} = \frac{4 \times 2}{340} = 23{,}5\,ms$

- Le temps de cycle minimal préconisé est de 60ms car sinon le capteur n'aurait même pas le temps de récupérer les données du premier aller retour d'ondes qu'il recommencerai déjà à en refaire

"we suggest to use over 60ms measurement cycle"

d) Le Trig génère une onde grâce à la tension de l'ESP qui va être envoyé par l'émetteur et le Echo génère une tension grâce à l'onde qu'il reçoit par le récepteur donc le Trig est une sortie et le Echo est une entrée

e) Du point de vue de la carte ESP le Trig est une sortie analogique et le Echo est une entrée numérique

f) Je fais le montage du capteur et je le programme pour mesurer la distance :

```
const int trigPin = 16;
const int echoPin = 17;

long duration;
int distance;

void setup() {
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  Serial.begin(9600); // Starts the serial communication
}

void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance= duration*0.034/2;
  Serial.print("Distance: ");
  Serial.println(distance);
}
```

Résultats :

```
Distance: 5.17
Distance: 99.30
Distance: 100.89
Distance: 102.75
Distance: 100.62
Distance: 104.36
Distance: 30.26
```

g) J'améliore le programme en ajoutant l'utilisation de l'écran LCD pour afficher la distance :

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);

const int trigPin = 16;
const int echoPin = 17;

long duration;
float distance;

void setup() {
  lcd.begin();
  lcd.backlight();

  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  Serial.begin(9600); // Starts the serial communication
}

void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance= duration*0.034/2;
  lcd.print("Distance: ");
  lcd.print(distance);
  delay(1000);
  lcd.clear();
}
```