



Module SCI

Ecole Nationale Supérieure d'Informatique d'Alger (Ex. ini)

Compte rendu TP4

14 Avril 2022

Introduction

Les plateformes électroniques telles que Arduino et raspberry Pi sont communément utilisées sous forme de modules embarqués visant à accomplir une ou plusieurs fonctionnalités d'un système plus large, de ce fait il est important de comprendre comment pouvoir exploiter les différentes interfaces et protocoles de communication.

Objectifs

1. **Partie théorique :** Compréhension des différents modes de communication entre Arduino et des modules externes (I2C, SPI, UART)
2. **Partie pratique :** Application des protocoles de communication séries avec ADS1115 (pour I2C) et HC-05 (pour UART)

Partie théorique

A. Explication des protocoles de communication I2C SPI et UART

Interface UART

Qu'est-ce que l'UART ?

- Signifie "Réception et transmission asynchrones universelles" (UART).
- Un protocole de communication série simple qui permet à l'hôte de communiquer avec le dispositif auxiliaire.
- L'UART prend en charge la transmission bidirectionnelle, asynchrone et en série des données.
- Il possède deux lignes de données, une pour transmettre (TX) et une autre pour recevoir (RX), qui sont utilisées pour communiquer par l'intermédiaire de la broche numérique 0, la broche numérique 1.
- TX et RX sont connectés entre deux dispositifs. (par exemple, USB et ordinateur)
- L'UART peut également gérer les problèmes de gestion de la synchronisation entre les ordinateurs et les périphériques série externes.

Comment fonctionne-t-il ?

Il peut fonctionner entre les dispositifs de 3 façons :

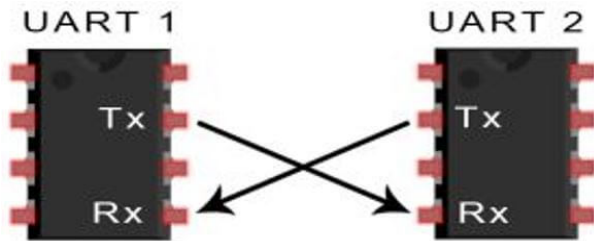
- Simplex = transmission de données dans une seule direction
- Half-duplex = transmission de données dans les deux sens mais pas simultanément
- Duplex intégral = transmission de données dans les deux sens simultanément.

Une fois connecté, les données circulent de TX de l'UART émetteur à RX de l'UART récepteur. L'UART étant un protocole de transmission série asynchrone, il n'a pas d'horloge. L'UART émetteur convertit les données parallèles du dispositif maître (ex. CPU) en données série et les transmet en série à l'UART récepteur. Il convertit ensuite les données série en données parallèles pour le dispositif récepteur. Comme l'UART n'a pas d'horloge, il ajoute les bits de début et d'arrêt qui sont transférés pour représenter le début et la fin d'un message. Cela aide l'UART récepteur à savoir quand commencer et arrêter la lecture des bits. Lorsque l'UART récepteur détecte un bit de début, il lit les bits à la fréquence BAUD définie. La vitesse de transmission des données de l'UART est appelée taux BAUD et est définie à 115 200 par défaut (le taux BAUD est basé sur le taux de transmission des symboles, mais est similaire au taux de bits).

Les deux UART doivent fonctionner à peu près au même débit en bauds. Si la différence de débit en bauds est supérieure à 10 %, la synchronisation des bits peut être décalée et rendre les données inutilisables. L'utilisateur doit s'assurer que les UART sont configurés pour transmettre et recevoir le même paquet de données.

Protocole de travail de l'UART

- Un UART qui transmet des données reçoit d'abord des données d'un bus de données envoyé par un autre composant (par exemple, l'unité centrale).
- Après avoir reçu les données du bus de données, il ajoute un bit de départ, un bit de parité et un bit d'arrêt pour créer le paquet de données.
- Le paquet de données est ensuite transmis sur la broche TX, où l'UART récepteur lira le paquet de données sur sa broche RX. Les données sont envoyées jusqu'à ce qu'il n'y ait plus de données dans l'UART émetteur.



Avantages de l'utilisation de l'UART

- Simple à utiliser, bien documenté car il s'agit d'une méthode largement utilisée avec de nombreuses ressources en ligne.
- Aucune horloge n'est nécessaire
- Bit de parité pour permettre la vérification des erreurs

Inconvénients de l'utilisation de l'UART

- La taille de la trame de données est limitée à 9 bits seulement.
- Impossibilité d'utiliser plusieurs systèmes maîtres et esclaves
- Les débits en bauds de chaque UART doivent se situer dans une fourchette de 10 % l'un par rapport à l'autre pour éviter toute perte de données.
- Faible vitesse de transmission des données

Interface I2C

Qu'est-ce que l'I2C ?

- L'acronyme I2C signifie Inter-integrated-circuit (circuit intégré).
- Il s'agit d'un protocole de communication série similaire à l'UART. Cependant, il n'est pas utilisé pour la communication entre un PC et un appareil, mais plutôt avec des modules et des capteurs.
- Il s'agit d'un simple bus série synchrone bidirectionnel à deux fils qui ne nécessite que deux fils pour transmettre des informations entre les dispositifs connectés au bus.
- Ils sont utiles pour les projets qui nécessitent de nombreuses pièces différentes (par exemple, des capteurs, des broches, des extensions et des pilotes) travaillant ensemble car ils peuvent connecter jusqu'à 128 dispositifs à la carte mère tout en maintenant une voie de communication claire !

Comment fonctionne-t-elle ?

- Il a 2 lignes qui sont SCL (ligne d'horloge série) et SDA (port d'acceptation de la ligne de données série)
- SCL est la ligne d'horloge pour synchroniser la transmission. SDA est la ligne de données par laquelle les bits de données sont envoyés ou reçus.
- Le dispositif maître initie le transfert de données par le bus et génère une horloge pour ouvrir le dispositif transféré et tout dispositif adressé est considéré comme un dispositif esclave.


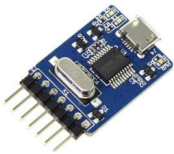
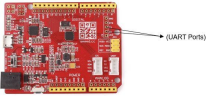



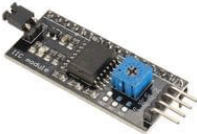
Avantages de l'utilisation de l'I2C

- Faible nombre de broches et de signaux, même avec de nombreux dispositifs sur le bus.
- Flexible, car il prend en charge la communication multi-maître et multi-esclave.
- Simple, car il n'utilise que 2 fils bidirectionnels pour établir la communication entre plusieurs dispositifs.
- Adaptable car il peut s'adapter aux besoins de divers dispositifs esclaves.
- Supporte plusieurs maîtres.

Inconvénients de l'utilisation de l'I2C

- Vitesse plus lente car elle nécessite des résistances pull-up plutôt que des résistances push-pull utilisées par SPI. Il a également une conception à drain ouvert = vitesse limitée.
- Nécessite plus d'espace car les résistances consomment de l'espace précieux sur le PCB.
- Peut devenir complexe lorsque le nombre de dispositifs augmente.

Exemples de composants

Standard de communication	Modèle	Image
UART	USB CP2102 Serial Converter	
	FT232r USB UART / USB to UART 5V	
	UART Seeeduino V4.2	
	Base Shield V2	
I2C	Grove – I2C Hub (6 Port)	
	4-Channel 16-Bit ADC for Raspberry Pi (ADS1115)	
	PCF 8574	

B. Explication et description ADS1115

Description du module ADS1115 avec PGA

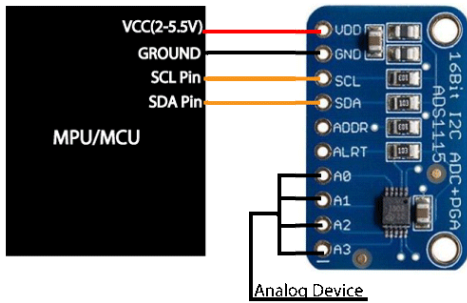
La partie pratique tourne autour de cette petite carte. En bref, il s'agit d'un convertisseur analogique-numérique (CAN) de 16 bits de résolution avec un amplificateur de gain programmable (PGA). On se dit déjà "mais l'Arduino n'a-t-il pas déjà un ADC intégré ?" et la réponse est "oui", mais il n'a que 10 bits de résolution. Alors quel est le problème de cette "résolution" dont on parle ? lorsque vous voulez apporter des informations du monde extérieur à un micro-contrôleur, vous utilisez un capteur. Le problème est que, surtout dans le cas de la luminosité, le signal est sous la forme d'une tension dont la valeur est comprise entre une valeur minimale et une valeur maximale. Dans le cas d'un Arduino et des périphériques, cette plage va de la masse (0 Volts) à Vcc (de l'ordre de 5 Volts). D'autres plages de tension peuvent être utilisées avec un Arduino, mais un réseau de diviseurs de tension devra être utilisé pour empêcher la tension d'entrée maximale de dépasser Vcc, les micro-contrôleurs traitent les informations sous la forme de signaux numériques qui n'ont que deux états, HIGH et LOW ou 1 et 0. Par conséquent, pour intégrer les deux, il faut un ADC ! Cet ADC convertit une tension en un signal numérique en prenant une tension et en la faisant correspondre à une valeur comprise entre un minimum et un maximum définis. C'est là que la résolution de l'ADC devient importante pour obtenir une mesure précise et fiable. Selon la formule ci-dessous on obtient selon le nombre de bits, pour les 10-bit de l'Arduino prend des pas de 4.9mV alors que l'ADS1115 prend des pas de 76.3 uV et comme vous le verrez plus tard, plus le pas de l'ADC est petit, mieux c'est !

Caractéristiques et spécifications du module ADS1115 avec PGA

- Plage de tension d'alimentation : 2-5.5V
- CAN 16 bits
- Consommation de courant continu : 150uA
- Interface I2C
- Oscillateur interne
- Débit de données programmable : 8SPS à 860SPS
- 860 échantillons/seconde sur I2C

Interfaces et connection des pins

L'interfaçage d'un module ADC ADS1115 à un MCU/MPU est facile. Comme indiqué ci-dessus, le module ADC communique via la communication I2C.



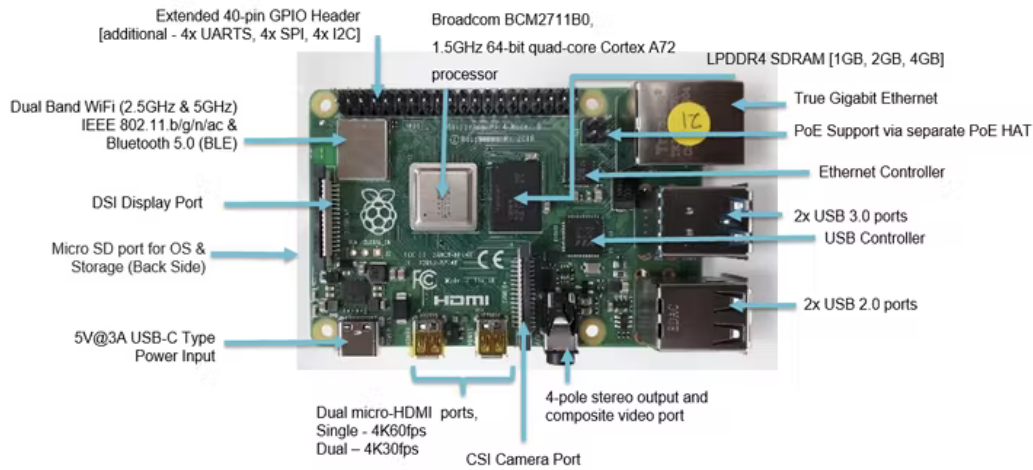
SCL (Serial Clock) et SDA (Serial Data) sur le module doivent être connectés à la broche SCL et SDA sur le MCU, respectivement. 2 broches sont destinées à l'alimentation ; VDD et la masse qui peuvent être connectées à la broche 5V et à la masse d'une MCU, respectivement. A0, A1, A2, A3 sont quatre broches d'entrée analogique, qui doivent être connectées à une source analogique (potentiomètre, etc).

C. Raspberry Pi 4 Model B

Le Raspberry Pi est une série de petits ordinateurs monocartes développés au Royaume-Uni par la Fondation Raspberry Pi pour promouvoir l'enseignement de l'informatique de base dans les écoles et les pays en développement. Le modèle original est devenu beaucoup plus populaire que prévu, se vendant en dehors de son marché cible pour des utilisations telles que la robotique. Il ne comprend pas les périphériques (tels que les claviers et les souris) et les étuis. Cependant, certains accessoires sont emballés dans plusieurs packs officiels et non officiels.

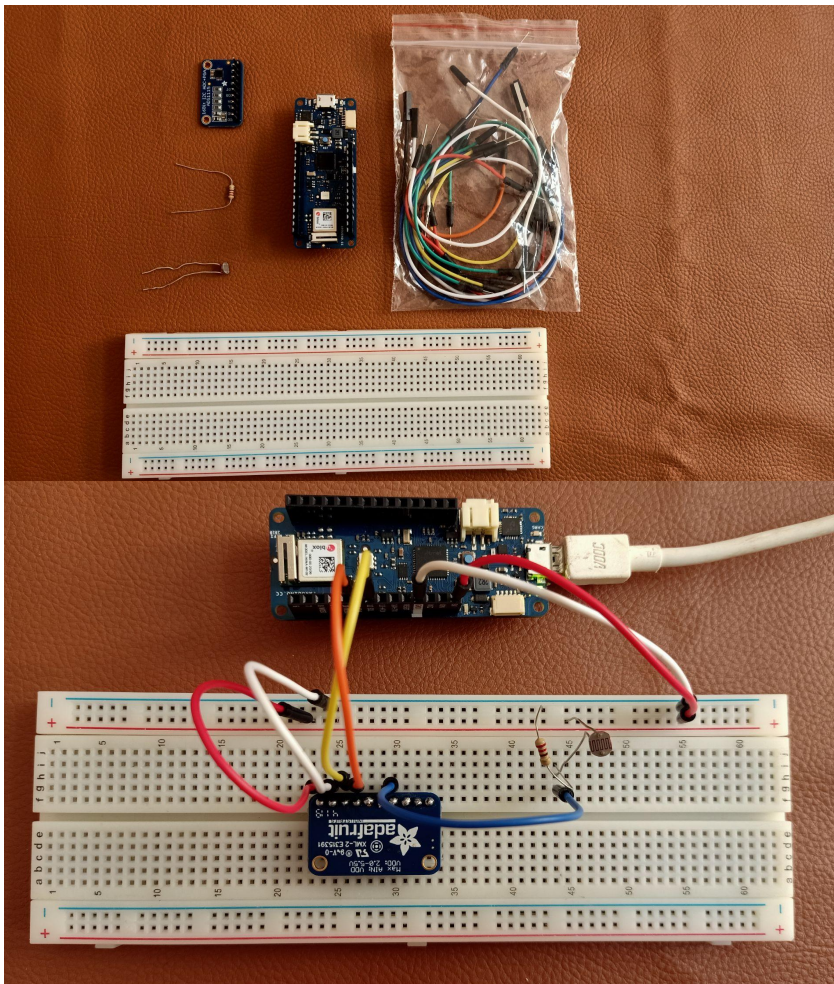
Configuration du Raspberry Pi 4

- Une alimentation électrique : Un port USB de type C est inclus dans l'appareil. Vous avez besoin d'une alimentation électrique d'au moins 3,0 A.
- Une carte Micro-SD : Vous en aurez besoin pour stocker ses fichiers et le système d'exploitation Raspbian. La capacité de stockage minimale requise est de 8 Go. De nombreux vendeurs fournissent des cartes micro-SD avec le système d'exploitation Raspbian préinstallé, vous êtes donc prêt à partir.
- Clavier et souris
- Écran de télévision/ordinateur



Partie pratique

A. Communication I2C



Les composants :

- ADS1115 Adafruit
- Arduino MKR 1010
- Câbles
- Breadboard
- Photorésistance
- Résistance 220 ohm

Le câblage :

VCC <-> 5V

GND <-> GND

SCL <-> SCL ET SDA <-> SDA

Formule :

$$\frac{\text{Résolution ADC}}{V_{cc}} = \frac{\text{Lecture ADC}}{\text{voltage en entrée}}$$

précision de lecture ADC = Tension en entrée \times $\frac{\text{Résolution ADC}}{V_{cc}}$

Exemple :

Arduino unifié :

$$\text{ADC}_{10 \text{ bits}} = 3 \text{ V} \times \frac{1024}{5 \text{ V}} = 614,4$$

Avec ADS1115 :

$$\text{ADC}_{16 \text{ bits}} = 3 \text{ V} \times \frac{65536}{5 \text{ V}} = 39321,6$$

Formule de calcul de la
résolution de l'ADC

```
#include <Wire.h>
#include "Adafruit_ADS1115.h"

Adafruit_ADS1115 ads(0x48);

float Voltage = 0.0;

int ADC_REF_VOLTAGE = 5;
int MAX_ADC_READING = 65536;
int REF_RESISTANCE = 220;

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

void loop(void) {
  int16_t adc0; // Lecture à partir de l'ADC

  float ldrRawData = ads.readADC_SingleEnded(0);

  // MAX_ADC_READING est 65536 (utilisation de 16 bits) and ADC_REF_VOLTAGE is 5 V
  float resistorVoltage = (float)ldrRawData / MAX_ADC_READING * ADC_REF_VOLTAGE;

  float ldrVoltage = ADC_REF_VOLTAGE - resistorVoltage;

  float ldrResistance = ldrVoltage/resistorVoltage * REF_RESISTANCE; // REF_RESISTANCE est 220 ohm

  Serial.print("AIN0: ");
  Serial.print(adc0);
  Serial.print("\tVoltage: ");
  Serial.println(ldrResistance, 7);
  Serial.println();
  delay(1000);
}
```

COM5

```
AIN0: 0 Voltage: 371.9322510
AIN0: 0 Voltage: 377.1672363
AIN0: 0 Voltage: 382.8840332
AIN0: 0 Voltage: 387.1350708
AIN0: 0 Voltage: 391.3577881
AIN0: 0 Voltage: 394.3418884
AIN0: 0 Voltage: 396.2744446
AIN0: 0 Voltage: 397.8578796
```

Code utilisé

Résultat obtenu

Remarque : La librairie Adafruit_ADS1X15 sert de pilote pour les convertisseurs analogique numérique Adafruit 12 et 16 bits (1015 et 1115) L'interfaçage se fait via I2C. L'adresse peut être changée de sorte que l'on peut avoir jusqu'à 4, ADS1x15 connectés sur un seul bus I2C à 2 fils pour 16 entrées simples.

B. Communication I2C

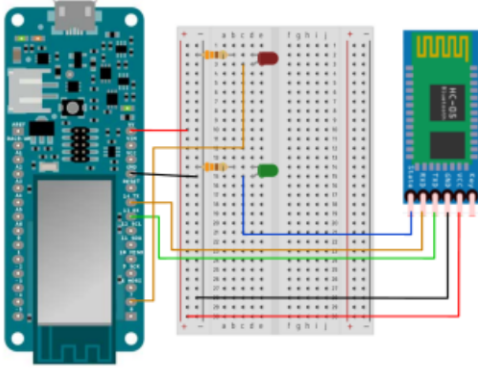
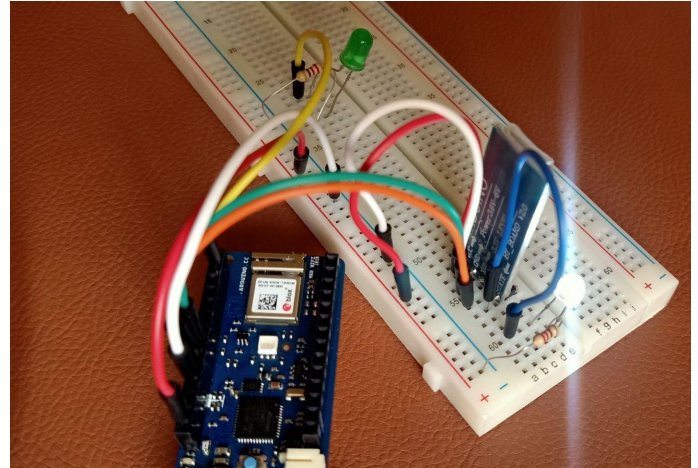


Schéma de branchement



Montage réel

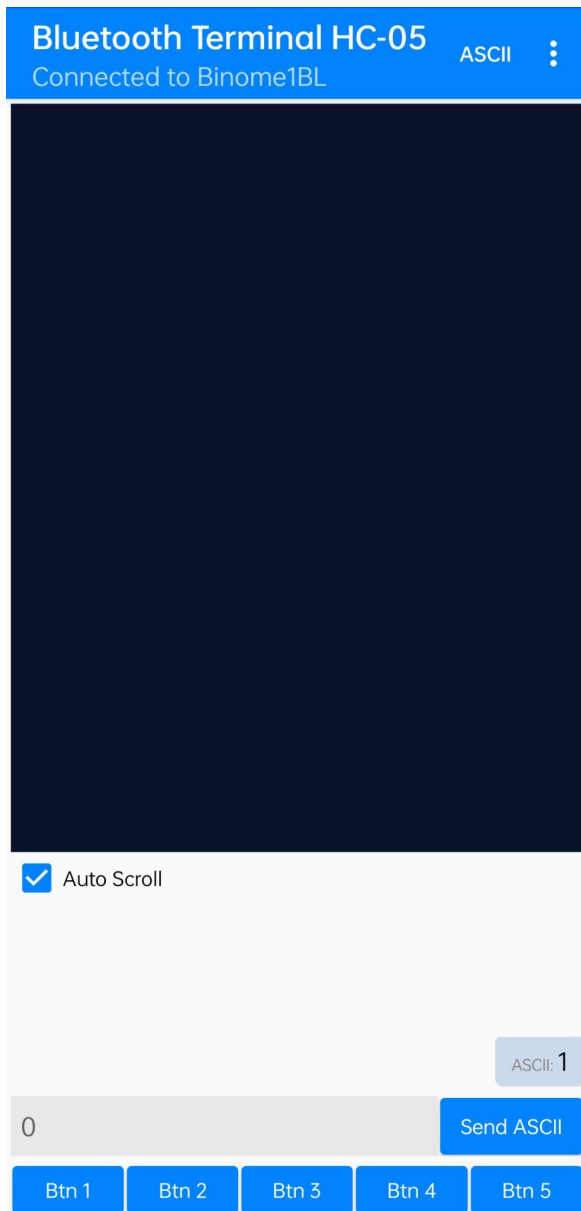
```
char flag = '0';
int LED = 7;
void setup()
{
  Serial.begin(9600);
  Serial1.begin(9600);
  pinMode(LED, OUTPUT);
  Serial.println("Ready to connect\nDefault password is 1234 or 000");
}
void loop()
{
  if (Serial1.available())
  {
    flag = Serial1.read();
    if (flag == '1')
    {
      digitalWrite(LED, HIGH);
      Serial.println("LED On");
    }
    else if (flag == '0')
    {
      digitalWrite(LED, LOW);
      Serial.println("LED Off");
    }
  }
}
```

Code permettant de contrôler la LED

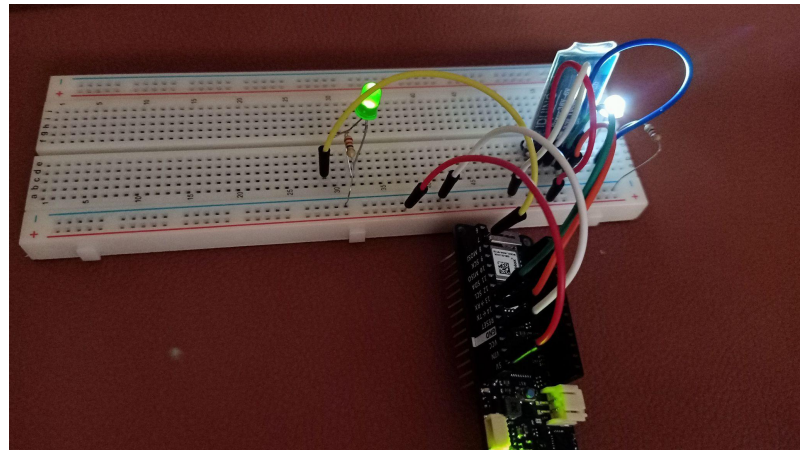
```
COM5
15:21:47.060 -> ERROR: (0)
15:21:55.594 -> OK
15:22:38.170 -> +NAME:HC-05
15:22:38.170 -> OK
15:22:43.343 -> +ADDR:98d3:c1:fd8275
15:22:43.343 -> OK
15:22:49.682 -> +PSWD:1234
15:22:49.682 -> OK
15:23:19.813 -> ERROR: (0)
15:23:35.657 -> OK
15:23:42.388 -> +NAME:Binome1BL
15:23:42.388 -> OK
15:23:55.927 -> OK
15:24:03.107 -> +PSWD:Password
15:24:03.107 -> OK
15:24:31.361 -> OK
```

Résultat de la configuration

Pour tester le programme : Nous utilisons un terminal sur le smartphone qui sera manipulé grâce à l'instruction Serial1 et ses fonctions fournies soit : available(); read(); ...etc



Terminal mobile connecté par Bluetooth



Résultat de la commande à distance