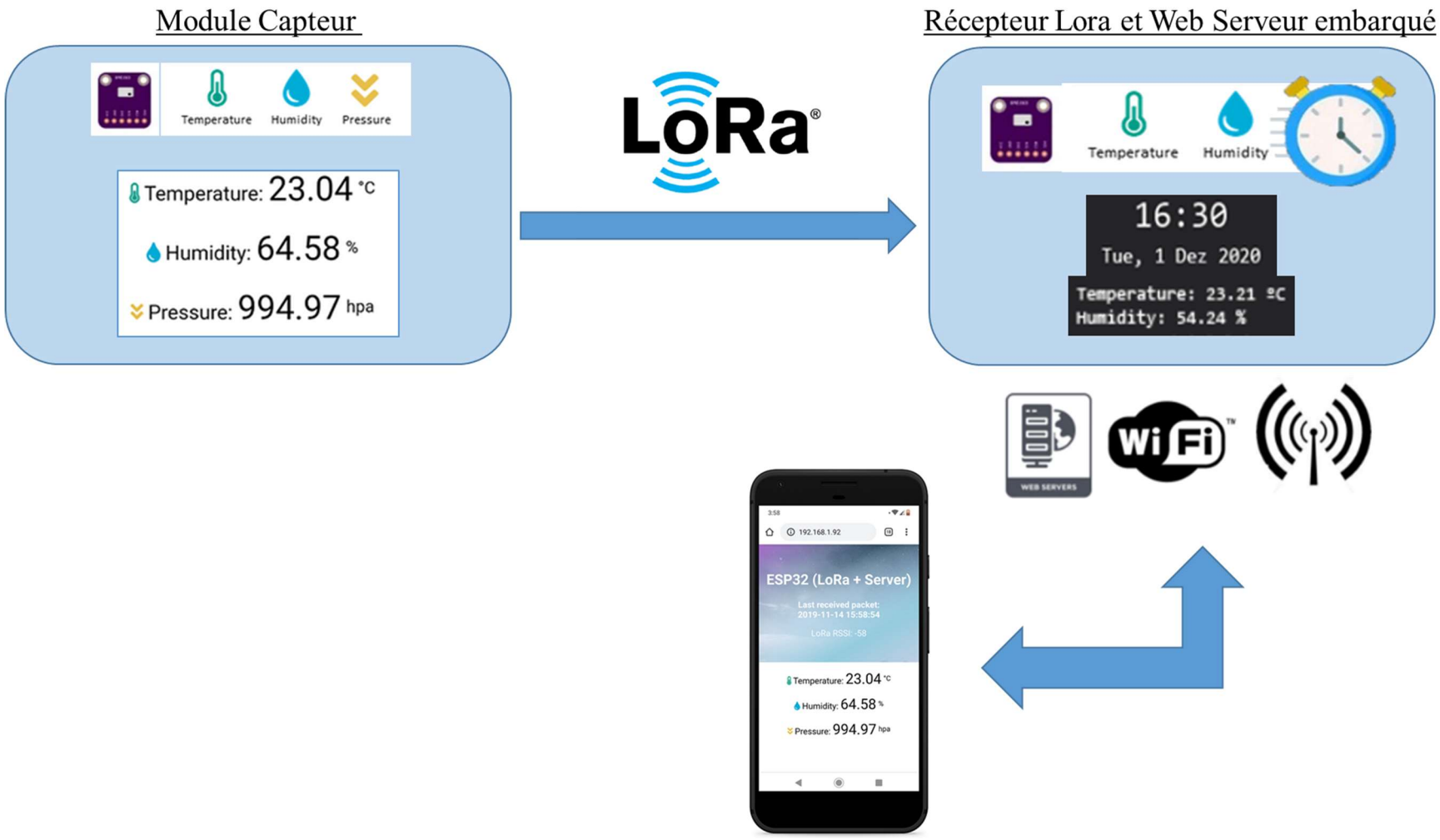
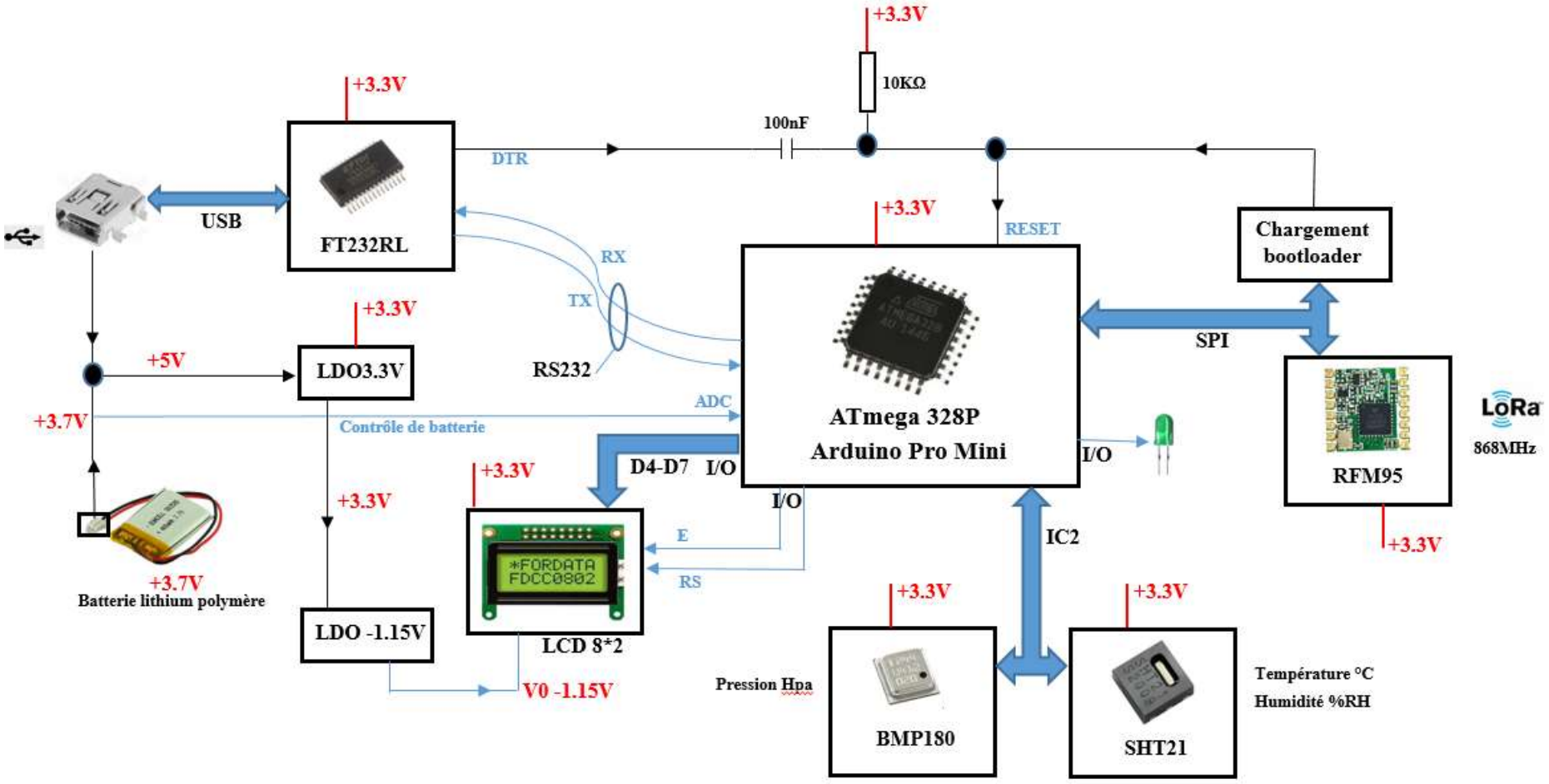
SA2- Prototypage électronique

Station météo connectée



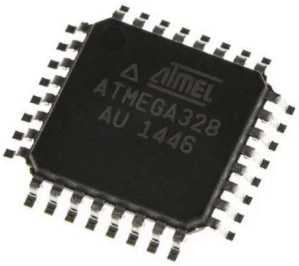
Synoptique du module Capteur :



Module Capteur

Le module a pour but d'acquérir la température, l'humidité (capteur SHT21) et la pression atmosphérique (BMP180), ces valeurs sont affichées sur un afficheur LCD (2*8 caractères) et transmises à la base en LoRaWAN (module RFM95). Le cœur du système du module capteur est basé sur un microcontrôleur 8bit (ATmega328) directement programmable sur IDE ARDUINO (via un module FTDI USB/RS232). Le module est alimenté à partir d'une batterie lithium polymère de 3.7V, permettant de générer la tension d'alimentation générale de 3.3V. Il doit être intégré dans un boîtier.

1. ATmega328P :



https://www.microchip.com/wwwproducts/en/ATmega328p#datasheet-toggle

Le cœur du système est basé sur le microcontrôleur ATmega328 de chez Microchip (anciennement Atmel). Tous les capteurs du module ont une alimentation maxi de 3.3V, pour éviter les circuits d'interface (SPI, I/O, I2C) entre le microcontrôleur et ces capteurs (et ainsi limiter le nombre de composants) et réduire la consommation du système. L'ATmega328 étant alimenté en 3.3V, sa fréquence max d'utilisation sera donc limitée à 13.33MHz (voir figure 29-1 page 312 de la DATASHEET).

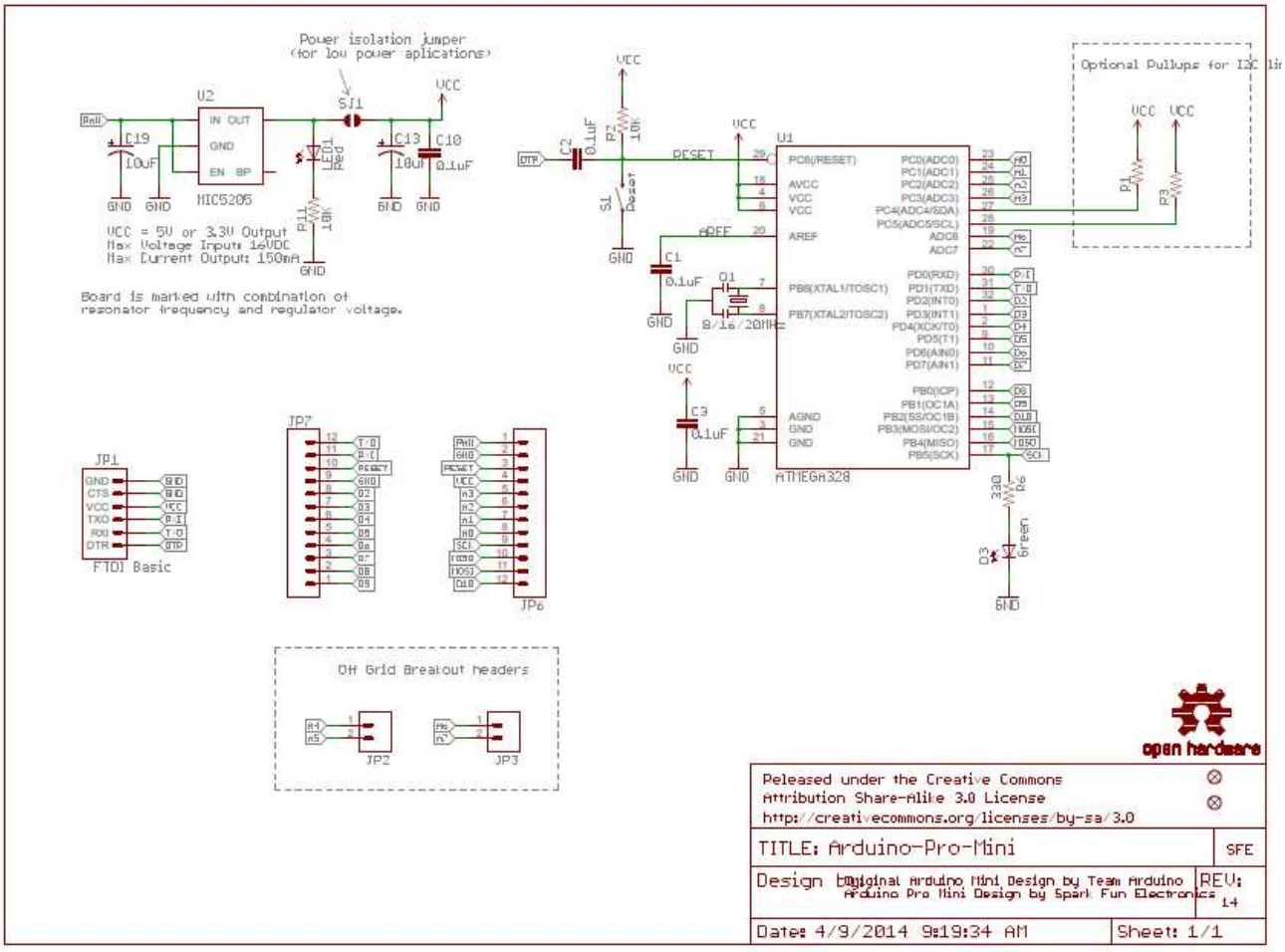
Nous choisirons finalement une source d'horloge de 8MHz pour notre microcontrôleur. La carte Arduino Pro Mini avec un quartz de 8MHz correspond à notre besoin, de plus cette carte est directement programmable via l'IDE ARDUINO.

Notre microcontrôleur doit être directement programmable via l'IDE ARDUINO, nous devons donc pouvoir le précharger avec un bootloader via l'interface ISP (In Site Programming, sur le bus SPI pour notre version). Ce préchargement sera réalisé à l'aide d'une autre carte ARDUINO.

https://www.arduino.cc/en/Tutorial/BuiltInExamples/ArduinoISP

http://riton-duino.blogspot.com/2020/01/arduino-le-bootloader.html

Une fois le bootloader chargé nous pourrons programmer notre microcontrôleur ATmega328 directement sous l'IDE ARDUINO sur le port série via une interface USB/RS232.


Power isolation jumper
(for low power aplications)
Optional Pullups for I2C lir
VCC = 5V or 3.3V Output
Max Voltage Input: 16VDC
Max Current Output: 150mA
Board is marked with combination of
resonator frequency and regulator voltage.
MIC5205
ATMEGA328
8/16/20MHz
FTDI Basic
0H Grid Breakout headers
open hardware
Released under the Creative Commons
Attribution Share-Alike 3.0 License
http://creativecommons.org/licenses/by-sa/3.0
TITLE: Arduino-Pro-Mini
SFE
Design by Original Arduino Mini Design by Team Arduino
Arduino Pro Mini Design by Spark Fun Electronics
REV: 14
Date: 4/9/2014 9:19:34 AM
Sheet: 1/1

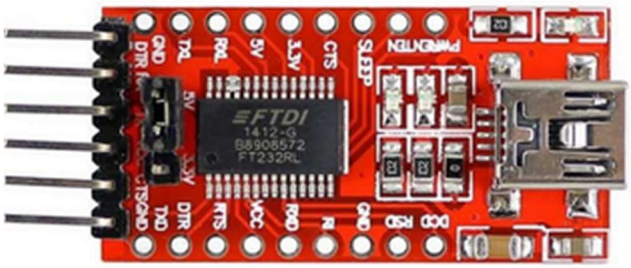
2. Interface USB/RS232 :

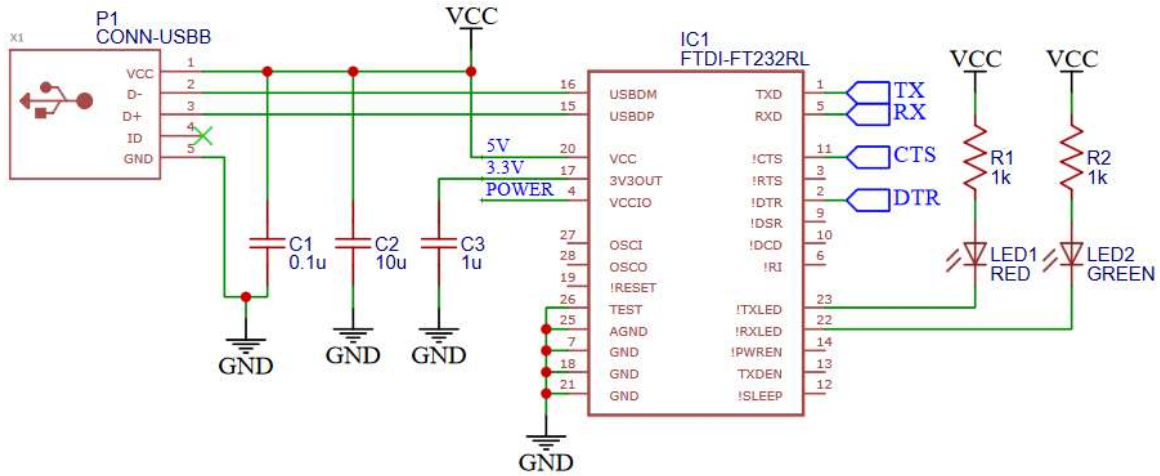
https://ftdichip.com/products/ft232rl/

Notre convertisseur USB/RS232 est basé sur le circuit FT232RL de chez FTDIChip. Ce circuit est un convertisseur USB2.0 UART série TTL, il va nous permettre de connecter notre module sur un port USB de PC et de programmer notre microcontrôleur directement sous l'IDE ARDUINO ou d'utiliser le moniteur série pour le débogage du système.

La carte FTDI 232RL nous servira de base pour la réalisation de notre schéma, le connecteur USB sera du type Micro USB 2.0.

https://fr.rs-online.com/web/p/connecteurs-usb/1362267





3. Capteur de température SHT21 :

https://fr.rs-online.com/web/p/capteurs-de-temperature-et-d-humidite/7865554/

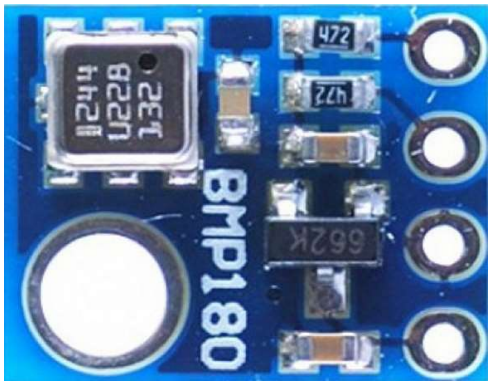
Le capteur SHT21 est un capteur d'humidité et de température numériques de Sensirion. C'est un capteur basse consommation (plage d'alimentation de 2.1V à 3.6V), en boitier CMS DFN (3*3*1.1 mm), disposant d'une interface série I2C. Sa plage de fonctionnement de -40°C à +125°C, avec une précision de +-0.3°C et +-2%RH (résolution de 8 ou 14 bits).

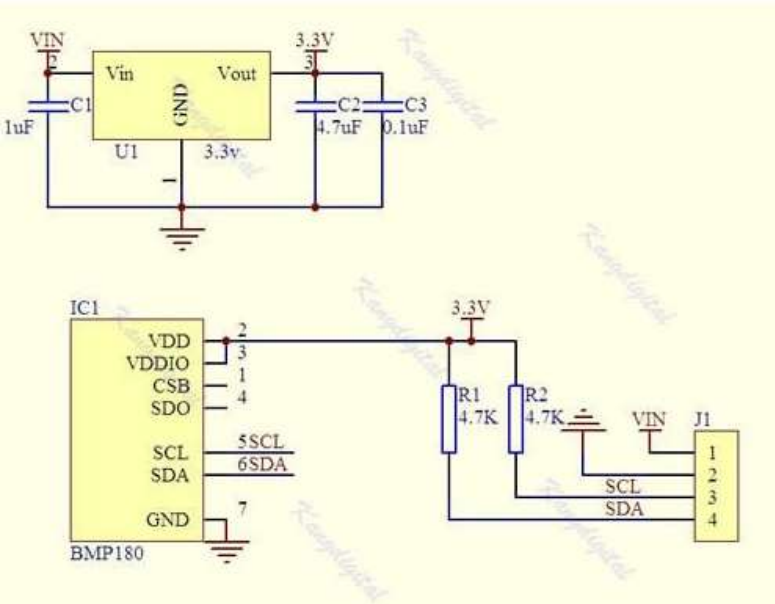
4. Capteur de pression :

Le circuit BMP180 est un circuit intégré mis au point par Bosch Sensortec conçu pour mesurer la pression atmosphérique avec précision et la température. En convertissant la pression mesurée en altitude, on peut obtenir un altimètre de grande précision. Le capteur est dans un boitier CMS LGA (3.8*3.6*0.93 mm) très difficile à souder, nous utiliserons donc une carte breakout.

https://www.amazon.in/Digital-Barometric-Pressure-Compatible-Arduino/dp/B00TMJP4JC

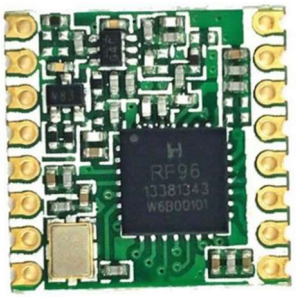
Le BMP180 est un capteur basse consommation (plage d'alimentation de 1.62V à 3.6V, 32uA) disposant d'une interface série I2C. Sa plage de fonctionnement de -40°C à +85°C, une mesure de pression de 300 hPa à 1100 hPa (résolution de 16 bits).





La carte breakout peut s'alimenter de 3 à 5VDC (un régulateur LDO est intégré à la carte), de plus des résistances de tirage sur le bus I2C (deux résistances de 4.7KΩ) sont intégrés.

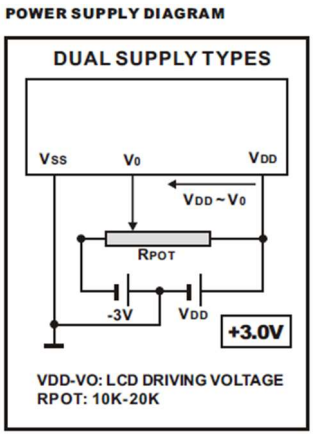
5. Module LoRa :



Nous utiliserons le même module LoRa dans le module capteur et dans le module serveur. Le module de transmission LoRa que nous allons utiliser est un module intégré de la famille RFM9xW (voir fonctionnement et caractéristiques détaillées ci-dessous, dans la partie module serveur).

6. Afficheur LCD :

Un afficheur LCD (Liquid Crystal Display) doit permettre d'afficher les différentes valeurs d'humidité, de température et de pression. Cet afficheur doit pouvoir s'alimenter en 3.3V. Nous prendrons le modèle FC0802E00 de chez FORDATA, un LCD alphanumérique de 2 lignes de 8 caractères, avec interface parallèle. Nous utiliserons cet afficheur en mode 4 bits (pour limiter le nombre d'I/O utilisées sur l'ATmega328). La tension d'alimentation est en +3.3V, par contre il faut créer une tension négative pour la tension V0 de contraste de l'afficheur. Cette tension négative pourra être ajustée.



https://fr.farnell.com/fordata/fc0802c00-fhyybh-91-e/afficheur-alphanumerique-8x2-jaune/dp/2674125?ost=2674125

http://fractale.gecif.net/si/logiciels/proteus/lcd/#11

7. Gestion des alimentions :

L'alimentation de base de notre module est un accu de 3.7V, de type lithium polymère (LiPo).

Estimation de la consommation totale :

Module LORA	120mA
Microcontrôleur	5mA
Capteur de température	330uA
Capteur de pression	1mA
Afficheur LCD	1.5mA
Total	127.8mA

(1) Alimentation 3.3v

Cette estimation est une consommation maxi (LoRa en mode émission, et les capteurs en mode calcul) en pointe. Notre module doit faire une acquisition des capteurs et une transmission LORA toute les secondes, en mode standby la consommation du module LORA est de l'ordre de 2mA, on peut estimer la consommation en mode de non émission de l'ordre de 15mA.

Nous choisirons une batterie lithium polymère de 3.7V pour 980mAh avec une platine chargeur intégré (environ 15€ pièce). Cette batterie pourra être directement rechargée en USB (via un connecteur et un adaptateur secteur 220VAC/5DCV). Le contrôle de la tension de batterie sera réalisé par l'ATmega328 sur une entrée ADC (via un pont diviseur de tension), on estimera que la batterie sera vide pour une tension de 3.6V (on fera clignoter la led, pour une tension faible). L'information de tension sera transmise au module serveur.

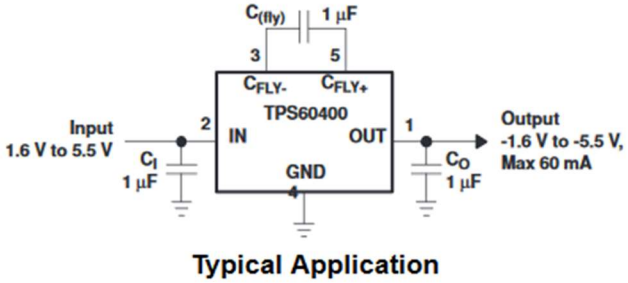
Nous allons donc réguler la tension batterie de 3.7V en 3.3V via un régulateur linéaire à faible DropOUT (écart entre la tension d'entrée et de sortie) et devant fournir un courant minimal de 130 mA. Notre choix s'est porté sur le modèle TC1262 de chez Microchip. Ce modèle est un régulateur à faible chute (LDO pour Low DropOut) LDO CMOS haute précision à sortie fixe, pouvant fournir 500 mA.

https://fr.rs-online.com/web/p/regulateurs-ldo/5339577/

Nous allons donc réguler la tension batterie de 3.7V en 3.3V via un régulateur linéaire à faible DropOUT (écart entre la tension d'entrée et de sortie) et devant fournir un courant minimal de 130 mA. Notre choix s'est porté sur le modèle TC1262 de chez Microchip. Ce modèle est un régulateur à faible chute (LDO pour Low DropOut) LDO CMOS haute précision à sortie fixe, pouvant fournir 500 mA.

(2) Alimentation négative

Un deuxième circuit permettra de générer une tension négative afin de pouvoir piloter le contraste de l'afficheur LCD 2x8. La tension de sortie sera ajustée par une résistance série en sortie et la valeur de la tension devra être entre de -1.2V à -2V (réglage contraste).

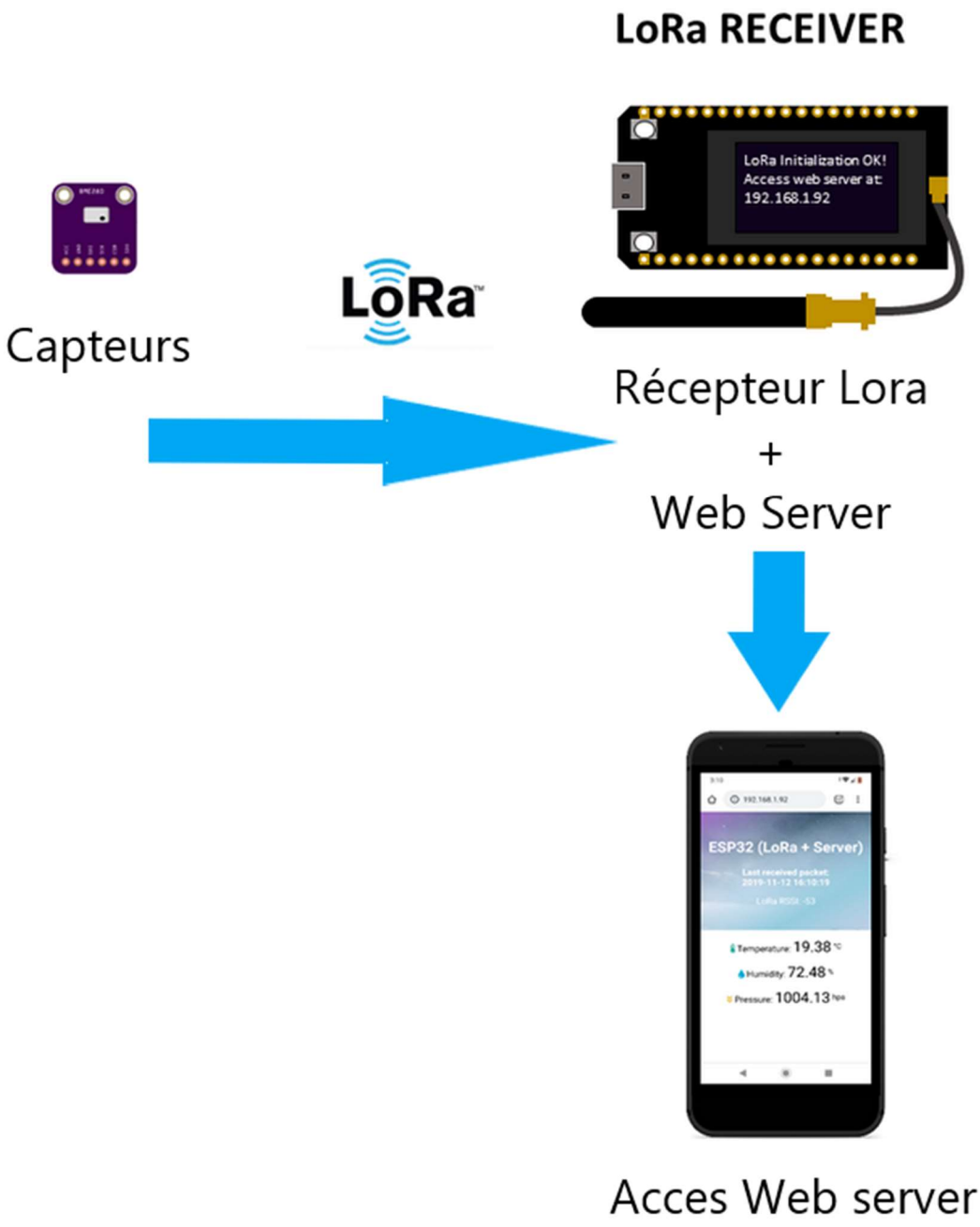


Typical Application

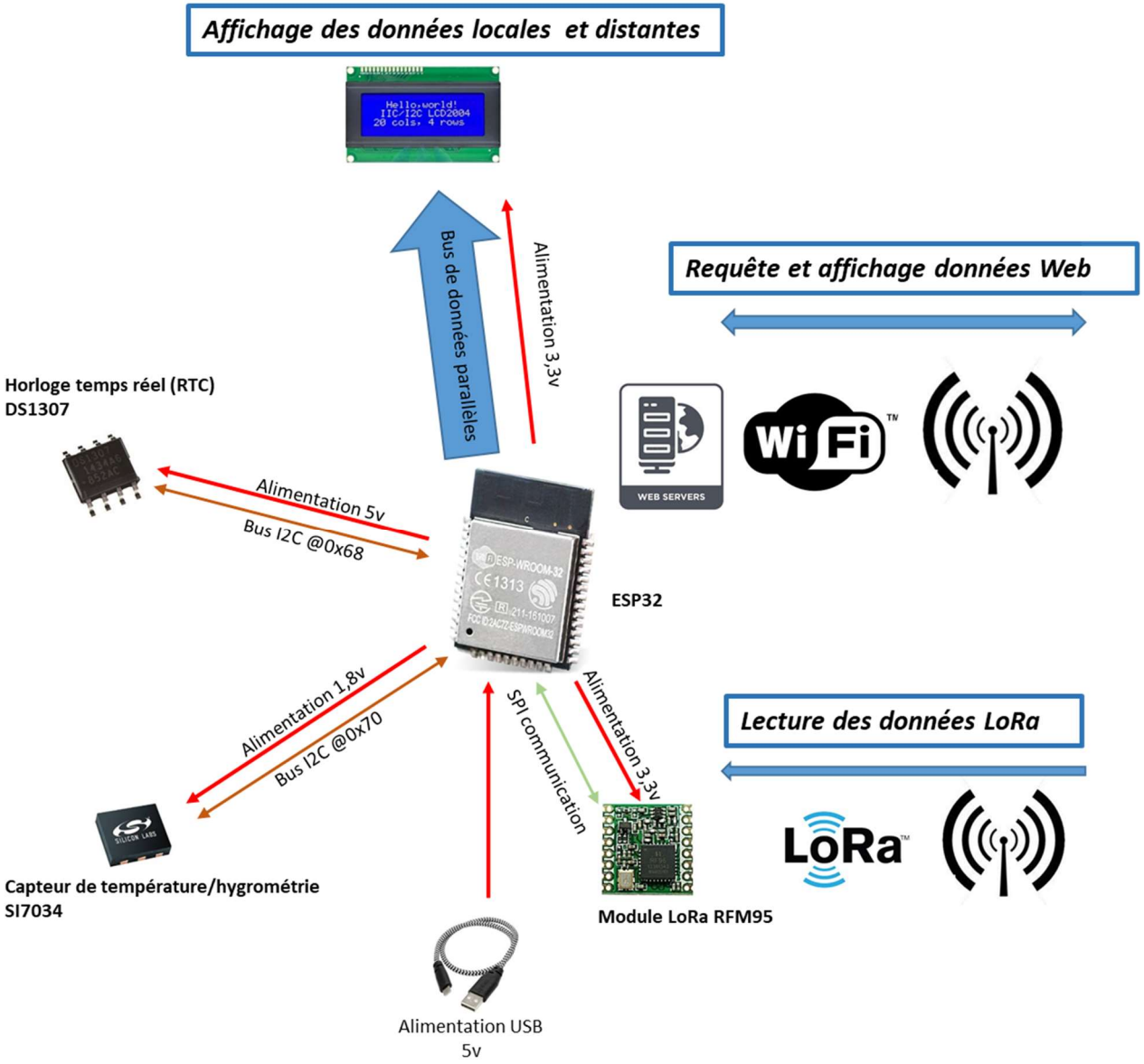
https://fr.rs-online.com/web/p/regulateurs-de-tension/0526975/

SA2- Prototypage électronique

Récepteur Lora et Web Serveur embarqué



Synoptique général



Le récepteur LoRa, le Web server embarqué et l'internet des objects

Nous allons réaliser une carte électronique composée des éléments permettant de recevoir des données de température et hygrométrie d'ensembles distants et de mettre ces informations à disposition sur internet. Cet Internet des Objects (IoT internet of things) sera composé d'un récepteur LoRa dans son mode point à point et d'un serveur Web embarqué accessible via le wifi.

Afin d'étoffer le dispositif et d'en faire une véritable station de météo, un afficheur local de 4 lignes par 20 caractères nous permettra d'afficher directement les données. Ce qui nous sera bien utile en phase de test.

Pour compléter et finir cette station, un capteur de température et d'hygrométrie est ajouté ainsi qu'une horloge temps réelle (RTC). L'heure sera rendue disponible après y avoir été stockée et synchronisé via le protocole NTP d'internet.

8. Internet des Objects (IoT) :

Tout en restant modeste dans cette configuration, la connexion d'éléments physiques (des capteurs) à un monde de l'internet (adresse http) sera rendu possible par une chaine de traitement et des éléments correctement choisis.

Dans un premier temps, cela passera par la prise en compte de l'information capteur dans sa forme souhaitée : synchronisation et mise en forme de la donnée.

Ensuite transmission de cette donnée sur un réseau ouvert puis relecture de ces données et présentations des différentes informations sur le monde internet.

9. Les éléments sélectionnés

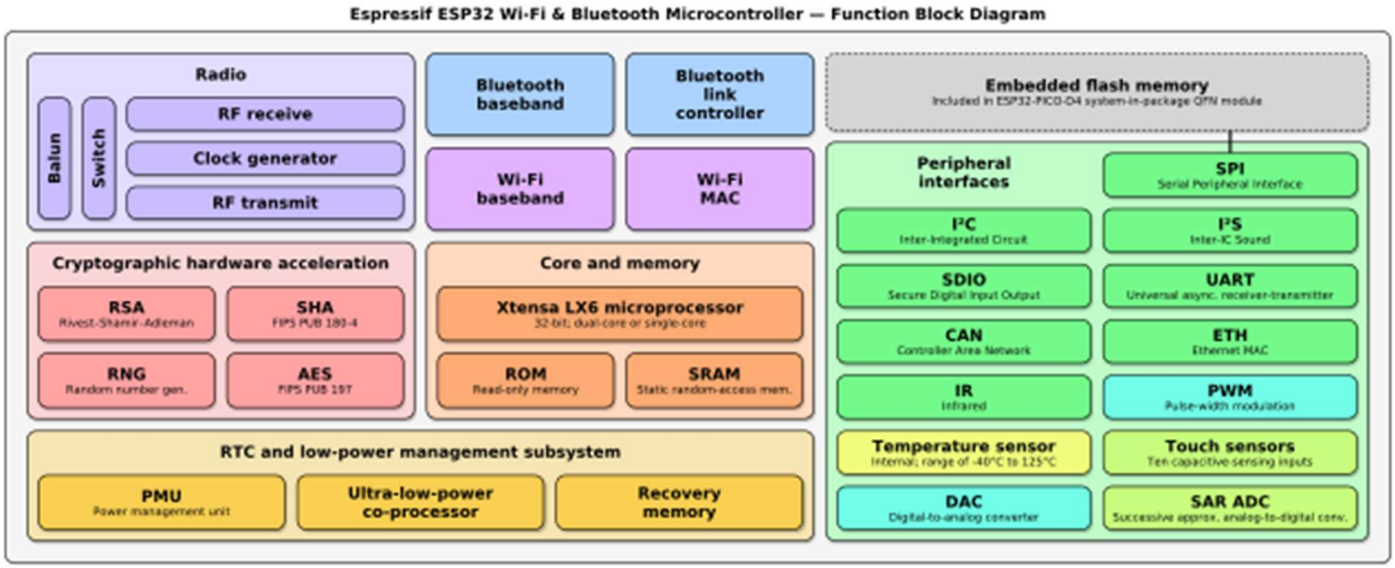
(3) ESP32 :

Le cœur du système, coté réception, est basé sur un microcontrôleur ESP32.

L'ESP32 est une évolution de l'ESP8266. Il s'agit en fait d'une famille de microcontrôleurs de type système sur une puce (SoC-Système on Chip) d'Espressif Systems intégrant la gestion du Wi-Fi, du Bluetooth et un DSP.

Très puissant, ce module est très apprécié dans le monde de l'IoT car il permet plusieurs modes de connexion mais aussi parce qu'il possède un mode de veille et une gestion de la consommation qui le rende très peu gourmand en énergie (gestion de fréquence d'horloge, mode Sleep, Bluetooth BLE).

Schéma fonctionnel

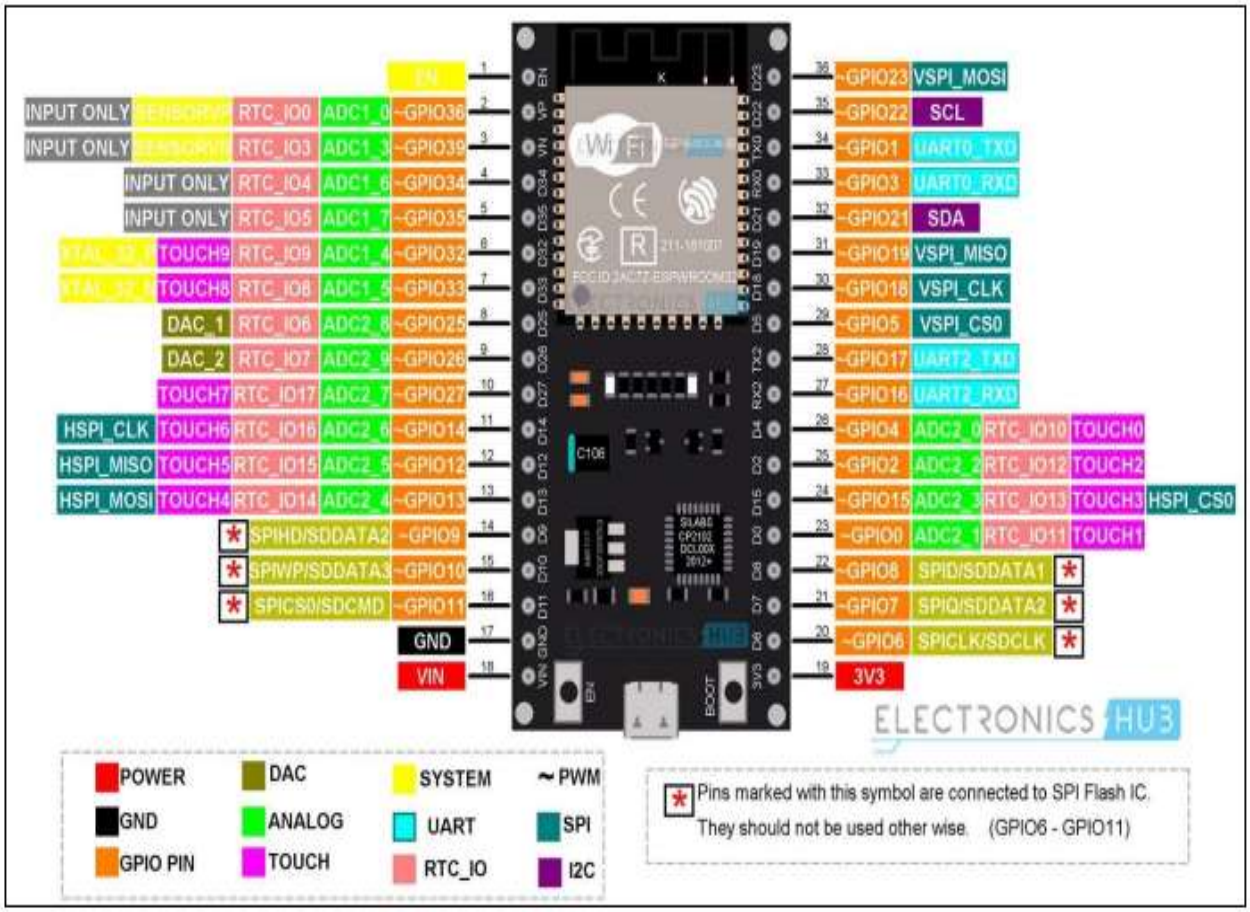


Les ESP32 comprennent notamment les caractéristiques techniques suivantes:

- Processeurs :
  - CPU : Xtensa double-cœur (ou simple-cœur), microprocesseur LX 32 bits, fonctionnant à 160 ou 240 MHz et fournissant jusqu'à 600 DMIPS ;
  - coprocesseur ultra basse consommation (ULP) ;
- Mémoire : 520 KiO SRAM ;
- Connectivité sans-fil :
  - Wi-Fi : 802.11 b/g/n ;
  - Bluetooth : v 4.2 BR/EDR et BLE jusqu'à v 5.0 et v 5.1 ;
- Interfaces de périphériques :
  - 12-bit Segmentation sur les ADC (SAR ADC) jusqu'à 18 canaux ;
  - 2 × 8 bit DAC ;
  - 10 × capteurs de touché (GPIO de capteur capacitif (en)) ;
  - 4 × SPI ;
  - 2 × interfacs I²S ;
  - 2 × interfaces I²C ;
  - 3 × UART ;
  - contrôleur hôte SD/SDIO/CE-ATA (en)/MMC/eMMC ;
  - contrôleur esclave SDIO/SPI ;

- interface MAC Ethernet avec DMA dédié et support du protocole de temps précis IEEE 1588 ;
- Bus de données CAN 2.0 ;
- contrôleur infrarouge distant (TX/RX, jusqu'à 8 canaux) ;
- Moteur PWM ;
- LED PWM (jusqu'à 16 canaux) ;
- Capteur à effet Hall ;
- pré-amplificateur analogique ultra-basse consommation ;

- Sécurité :
  - Standard de sécurité supportant complètement IEEE 802.11,incluant WPA/WPA2 et WAPI de WFA ;
  - Secure boot (démarrage sécurisé) ;
  - Chiffrement de la Flash ;
  - 1024-bit OTP, jusqu'à 768 bit pour les clients ;
  - Accélération matérielle du chiffrement : AES, SHA-2, RSA, Cryptographie sur les courbes elliptiques (ECC), Générateur de nombres aléatoires (RNG) ;
- Gestion de l'énergie :
  - low-dropout regulator (en) interne.
  - Domaines d'alimentation individuels pour le RTC
  - Alimentation en sommeil profond de 5 µA ;
  - Réveil depuis des interruption GPIO, timer, mesure ADC, interruption du capteur de touché capacitif.



Celui qui va nous intéresser tout particulièrement est l'ESP32-WROOM-32 en version DEVKITC:

https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/get-started-devkitc.html
https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf
https://fr.aliexpress.com/item/4000373163955.html?spm=a2g0s.9042311.0.0.53cf6c37F2VPxQ

CPU :

- ESP32-D0WD-V3 embedded, Xtensa®dual-core32-bit LX6 microprocessor, up to 240 MHz
- 448 KB ROM for booting and core functions
- 520 KB SRAM for data and instructions
- 16 KB SRAM in RTC

WiFi :

- 802.11b/g/n

Bluetooth :

- Bluetooth V4.2 BR/EDR and Bluetooth LE

Hardware :

- Interfaces: SD card, UART, SPI, SDIO, I2C, LEDPWM, Motor PWM, I2S, IR, pulse counter, GPIO,capacitive touch sensor, ADC, DAC
- 40 MHz crystal oscillator•4 MB SPI flash
- Operating voltage/Power supply: 3.0~3.6 V

- Operating temperature range: –40~85 °C

Remarques :

L'entrée Vin de l'«ESP32 DEVKIT DOIT» fonctionne entre 4,5 et 12 volts. Un régulateur (NCP1117) sur le circuit abaisse la tension à 3,3 volts et fournit un courant maximum de 800mA.

Les broches d'entrées/sortie de « ESP32 DEVKIT DOIT» ne supportent pas le 5 volts. Le voltage à l'entrée ne doit pas dépasser 3,6 volts.

Les GPIO 36, 39, 34 et 35 sont des broches d'entrée de signal seulement (I). Elles ne possèdent pas de résistances internes de pull-up et pull-down.

Les broches GPIO < 34 peuvent fonctionner en PWM.

Le courant maximal absolu par GPIO est de 40 mA.

Programmations de l'ESP32

L'ESP32 peut être programmé en : Espressif IDF (IoT Development Framework) qui est le langage officiel, Arduino IDE, LUA FreeRTOS, Micropython, Mrubis, JavaScript, …là encore, l'on peut voir l'intérêt de ce microcontrôleur par la diversité d'environnement de travail disponible.

Ici, nous utiliserons le logiciel de programmation ARDUINO.

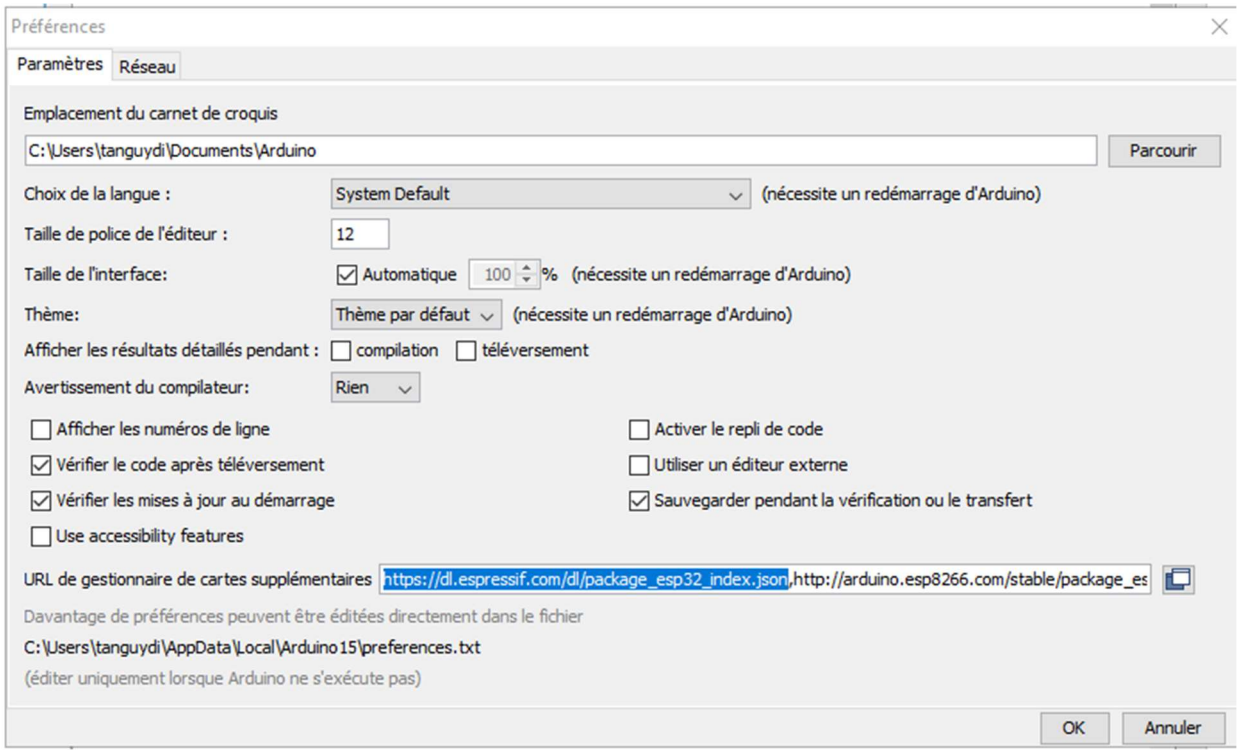
Préparation de l'environnement

Le logiciel Arduino IDE est téléchargeable : https://www.arduino.cc/en/software

Ensuite, pour l'utilisation de ESP32 DEVKIT, il faut renseigner dans :

Fichier>Préférences>URL de gestionnaire de cartes supplémentaires

Rajouter une virgule à la suite des adresses déjà existantes et rajouter l'adresse : **https://dl.espressif.com/dl/package_esp32_index.json** > OK

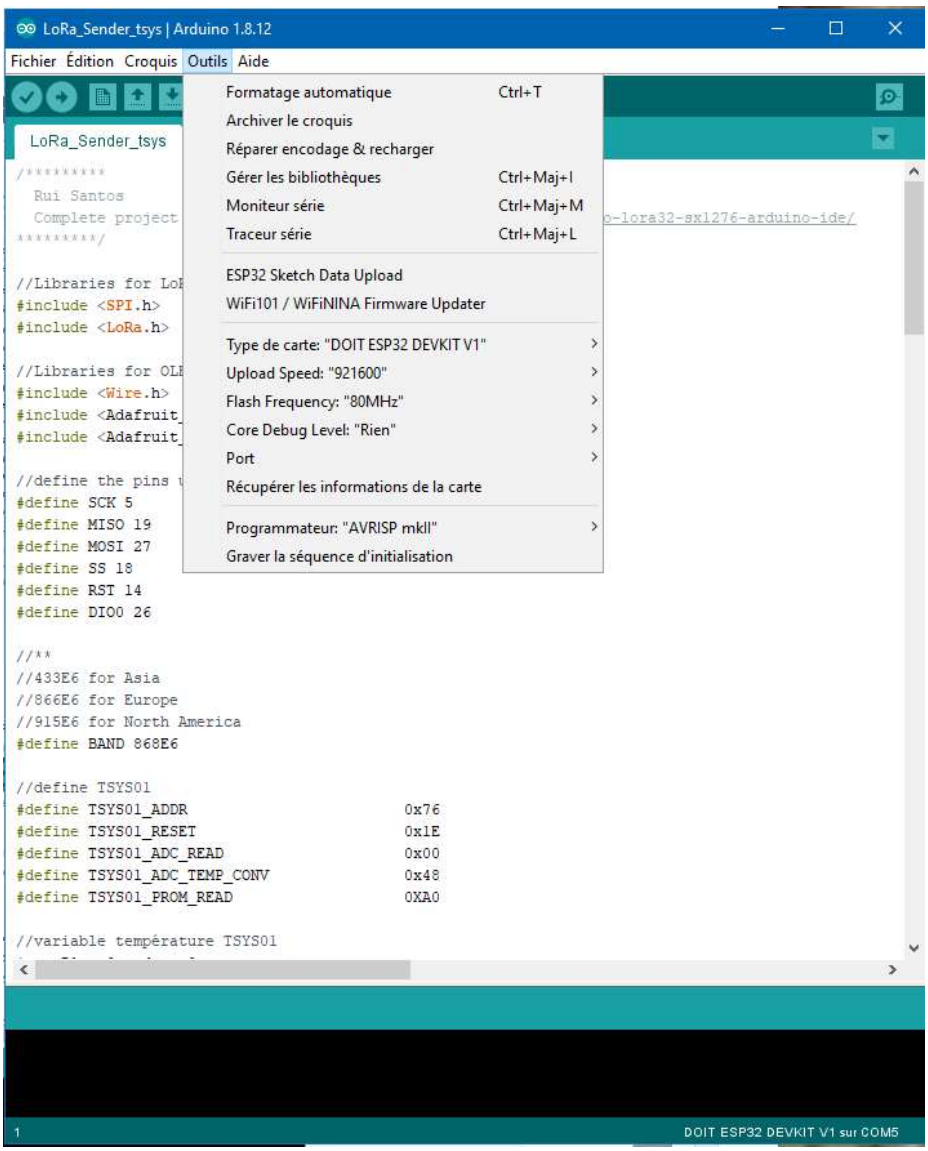


Ensuite,

Outils>Type de carte> Gestionnaire de cartes>Type>search « esp32 »>installer

Pour vérifier si la carte est bien prise en compte,

Outils>Type de carte>DOIT ESP32 DEVKIT V1 doit apparaitre



Et tester votre carte, connecter le port USB sur la carte,

Outils>Port>COM5 ensuite

Fichier>Example>WIFI>WiFiScan

Un code apparait, presser le bouton Téléverser « Compilation du croquis » apparait.

Une action du bouton « Boot» permet de lancer le téléchargement lorsqu'indiqué.

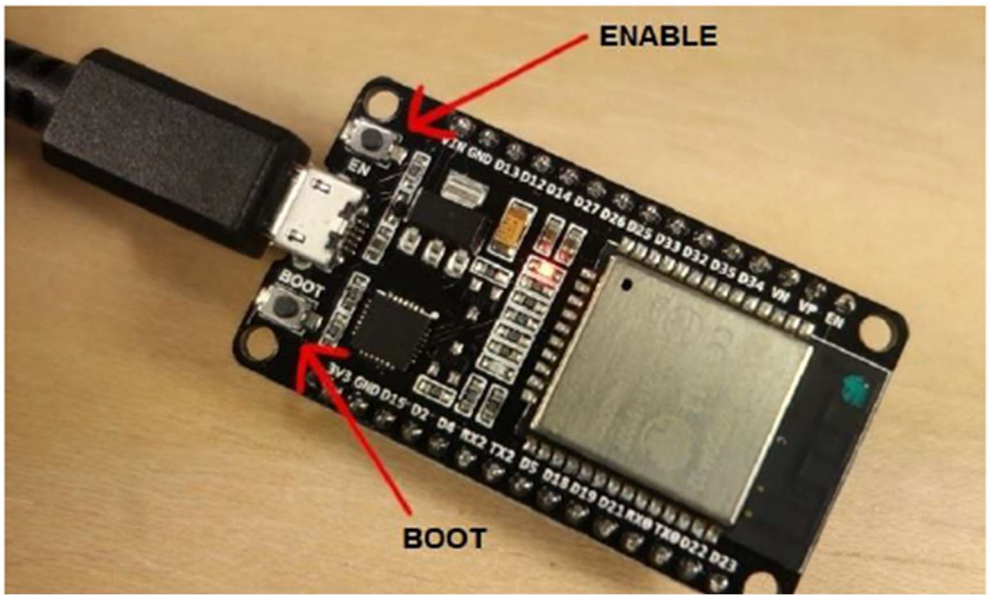
Maintenant le code est exécuté sur l'ESP32. Cliquer sur le bouton «Moniteur série ». Vérifier que la vitesse du terminal est sur « 115200 baud ».

Les réseaux wifi disponibles sont scannés par l'ESP32 et les noms apparaissent sur le moniteur série.

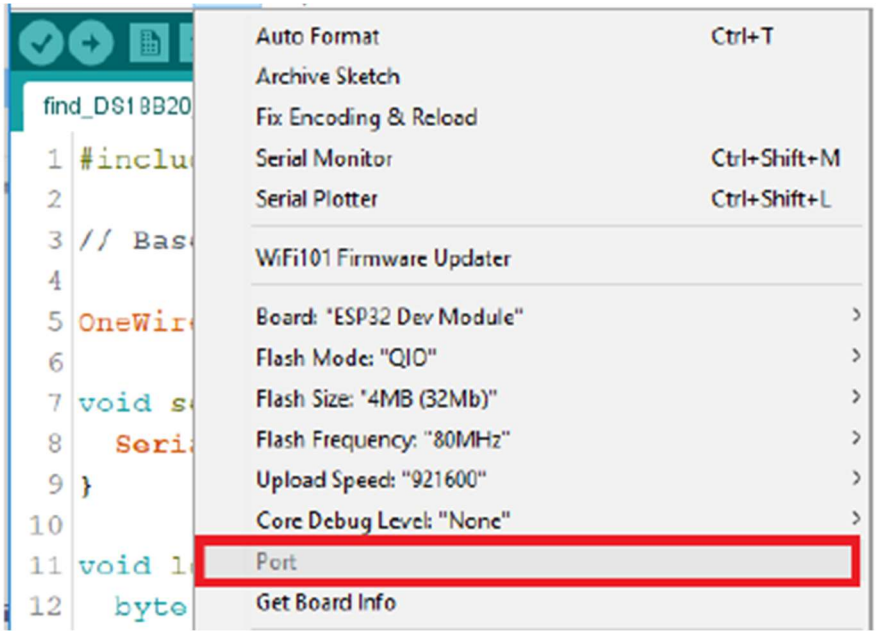
Problème éventuel

"Failed to connect to ESP32: Timed out… Connecting…"

Lorsque vous essayez de « téléverser », la connexion à la carte ne s'effectue pas correctement. Vérifier la connexion au port COM, maintenir le bouton BOOT et relancer le téléchargement.



« COM Port not found/not available »



Cela peut dû à 1 : le driver de l'USB n'est pas présent sur votre machine 2 : le câble USB est défectueux.

L'ESP32 DEVKIT utilise un contrôleur USB CP2102. Vous pouvez télécharger sur internet et installer le driver sur votre machine en effectuant une recherche sur « CP2102 driver download». Redémarrer ensuite ARDUINO IDE.

Pour une utilisation d'un serveur web asynchrone, vous devez télécharger

(download ESPAsyncWebServer library) et (download AsyncTCP library)

Pour l'utilisation de NTP, vous devez télécharger et renommer le répertoire en « NTPClient »

(NTPClient library forked by Taranais)(Click here to download the NTPClient library)

Ces bibliothèques ne sont pas disponibles et ne peuvent pas être installées via le gestionnaire de bibliothèques. Vous pouvez y accéder par,

Croquis> Inclure la bibliothèque> Ajouter une bibliothèque .ZIP

… et sélectionner les bibliothèques que vous venez de télécharger.

Les bibliothèques LoRa <LoRa.h>, SPI<SPI.h> et I2C <Wire.h> sont disponibles via le gestionnaire de bibliothèques.

Outils>Gérer les bibliothèques

(4) Le LoRa et RFM95

Actuellement, de très nombreux acteurs se lancent dans l'Internet des Objets (IdO):

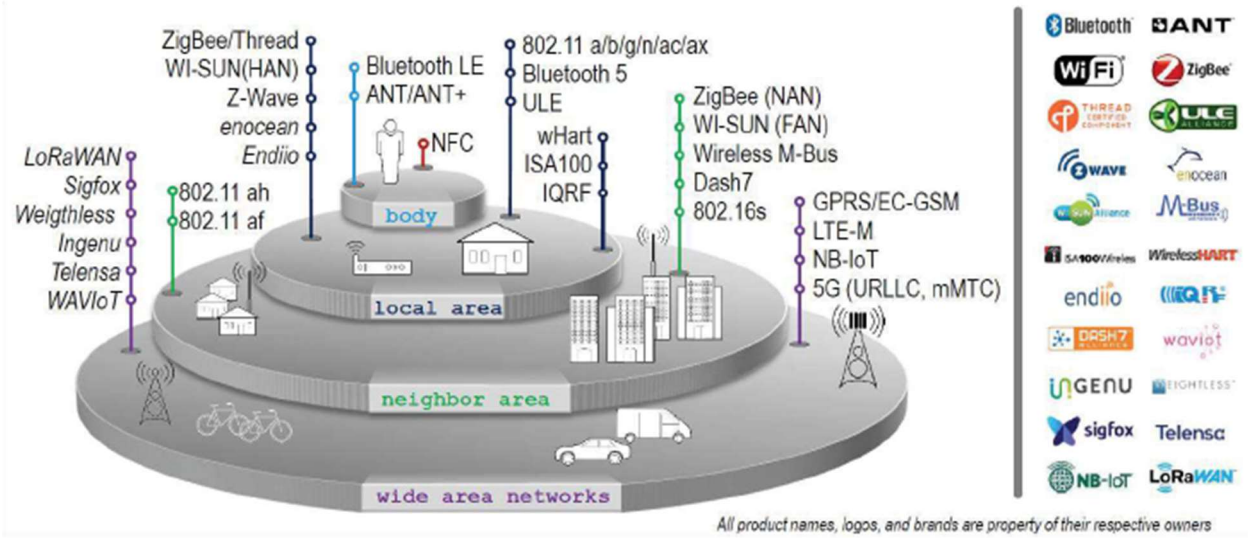
· les opérateurs de télécommunications ;

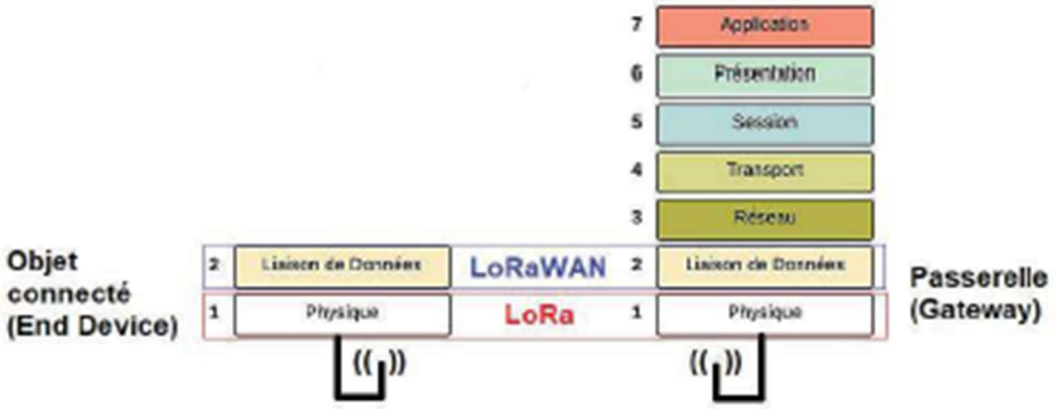
· les consortiums institutionnels tels que ceux à l'origine des normes standards :

- WiFi (IEEE 802.11),
- Bluetooth (IEEE 802.15.1),
- Zigbee (IEEE 802.15.4) ;

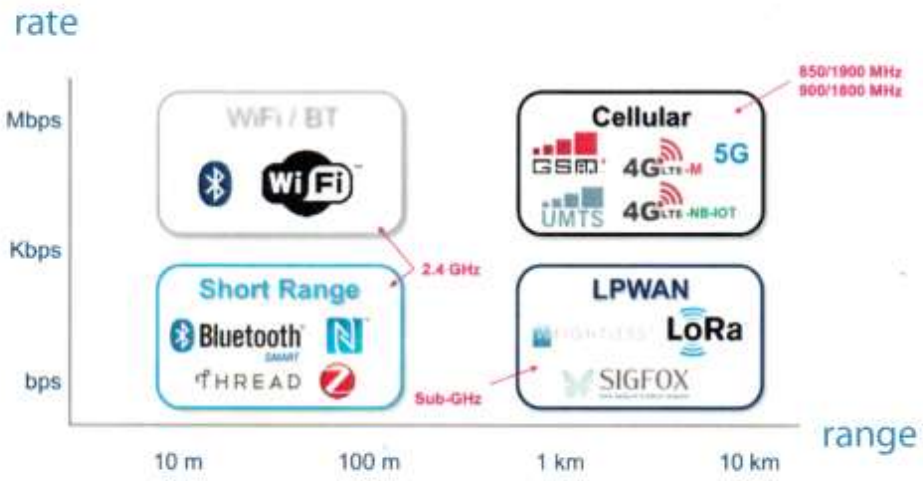
· un certain nombre de start-up tentant d'imposer leur standard.

Les deux premiers acteurs proposent une évolution de leurs produits alors que les derniers proposent une technologie dédiée à l'IdO en natif.





L'internet des objets a mis au-devant de la scène une catégorie que l'on classe dans les LPWAN (Low Power WAN) dont les caractéristiques sont une grande portée, une consommation faible et un débit moindre.



LoRa versus LoRaWAN

La technologie LoRaWAN a été développée par la start-up grenobloise Cycléo. Cette dernière a été rachetée en 2012 par l'entreprise californienne Semtech qui détient désormais les droits exclusifs sur la technologie LoRa. La technologie LoRa, aussi parfois appelée LoRa RF, désigne l'interface radio du réseau de communications sans fil (couche physique), alors que l'appellation LoRaWAN, parfois désignée également par le terme LoRa MAC désigne le protocole de communication. D'où la modélisation du réseau de communications LoRaWAN associant un objet communicant au serveur de réseau :

* LoRa, qui est l'abréviation de Long Range, désigne l'interface radio « longue portée » assurant le transit des informations bidirectionnelles entre les objets connectés et la ou les passerelle(s) ;

* LoRaWAN, qui signifie Long Range Wide Area Network, désigne l'ensemble du réseau de communications, depuis chacun des objets connectés jusqu'au serveur.

Signalons que la technologie liée à l'interface radio LoRa est propriétaire (brevet déposé par la société Semtech), alors que la couche protocole LoRaWAN est ouverte et accessible à tous.

Pour notre besoin, nous utiliserons uniquement le transport de l'information en interface radio LoRa.

Principe

La capacité maximum d'un canal de transmission a supporté un débit d'information en présence de bruit peut s'exprimer par le théorème de ShanonHartley :

$$C = BW \log_2 (1+S/N)$$

avec C la capacité du canal en bits/s, BW la bande passante, S la puissance moyenne du signal reçu et N la puissance moyenne du bruit du canal de réception.

Pour améliorer le débit à S/N constant (sans augmentation de la puissance émise)

→augmenter la bande passante

Pour compenser un S/N faible (augmentation du bruit)

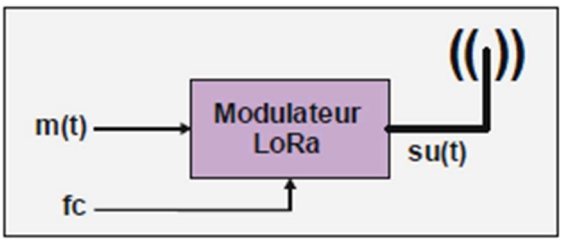
→augmenter la bande passante

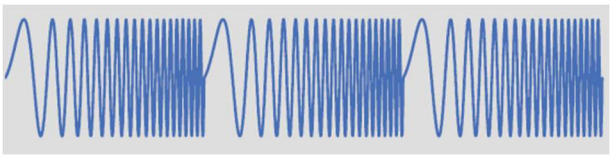
C'est ce que fait la modulation LoRa en reprenant une technique radar appelée CHIRP (Compressed High Intensity Radar Pulse) ou « étalement de spectre ».

Le LoRa utilise une méthode d'étalement de spectre qui permet de transmettre en même temps, sur le même canal.

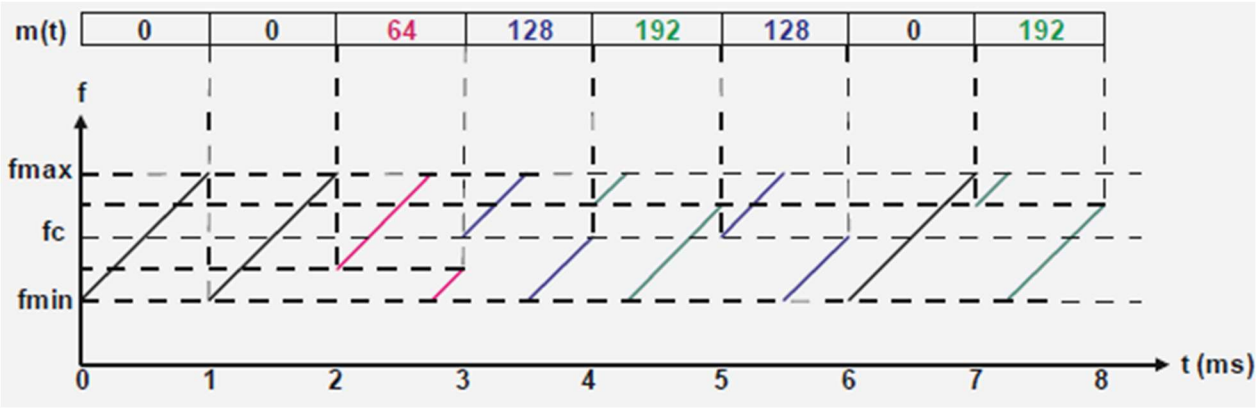
Le protocole LoRa utilise sept « codes d'étalement » appelés Spreading Factor [SF6, SF7, SF8, SF9, SF10, SF11 et SF12] qui lui permet d'avoir sept transmissions simultanées sur un même canal de transmission.

Le signal émit est donc une modulation fonction du symbole à émettre





Le codage de l'information consiste, à l'instant de transmission d'un nouveau symbole, à faire commencer le chirp avec une fréquence f dont la valeur est fonction de la valeur du symbole n. Soit pour un codage sur 8 bits (0 à 255) à 1 bits par millisecondes (1Kbauds).



L'on voit ici que la modulation LoRa dépend d'au moins 4 facteurs qui sont :

- La fréquence d'émission : en Europe, on utilise une partie de la bande libre ISM (Industrielle, Science, Médicale) soit le 868Mhz
- La largeur de bande ou bande passante du signal étalé, qui pour l'Europe peuvent être 125 ou 250khz
- Le facteur d'étalement ou Spreading Factor (SF) qui à 6 valeurs de 7 à 12 correspondant au nombre de bits transmit (SF8=8bits=0 à 255)
- Et le temps d'un symbole ou vitesse de modulation Ts tel que Ts = 2SF/BW

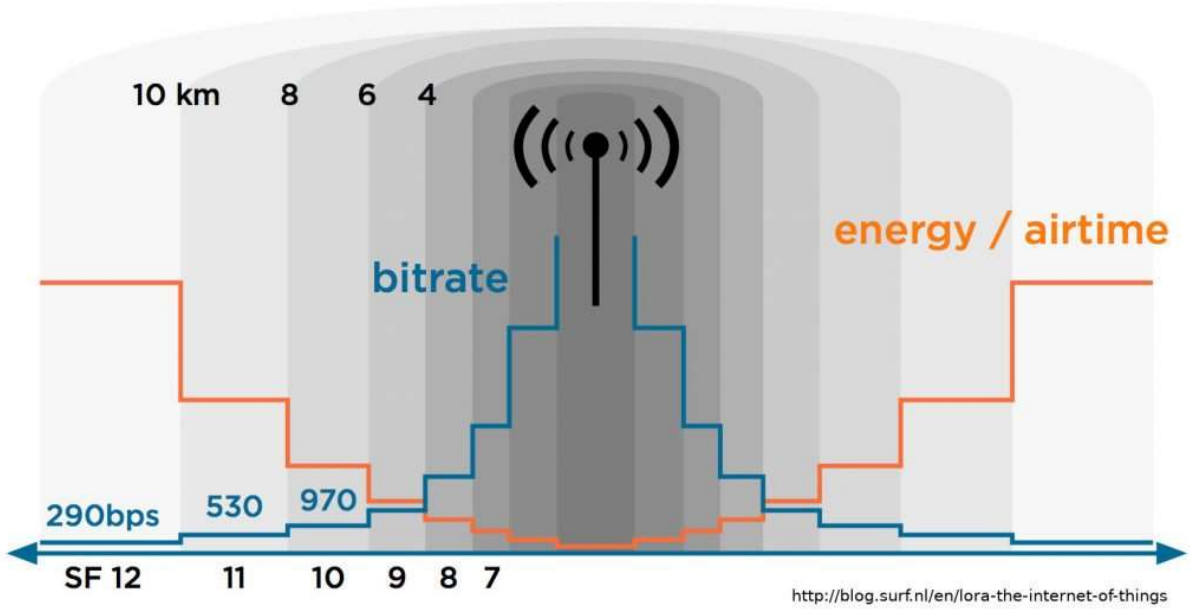
On obtient donc un débit binaire D_b= SF x BW/2^{SF} (en bits/s)

A cela le constructeur rajoute des bits supplémentaires qui serviront à détecter et corriger d'éventuelles erreurs appelés CR (Code Rate) sur le principe du FEC (Forward Error Correction). 4 valeurs possibles : 4/5, 4/6, 4/7 et 4/8 (4 bits supplémentaires tous les 8 bits).

Les paramètres **BW, SF et CR sont programmables**, au niveau du module LoRa, et compte tenu des valeurs autorisées par la norme en Europe, on obtient 28 valeurs de débit de transmission utile.

SF	BW (kHz)	Débit binaire utile en bits/s			
		CR=4/5	CR=4/6	CR=4/7	CR=4/8
7	250	10938	9115	7812	6836
7	125	5469	4557	3906	3418
8	125	3125	2604	2232	1953
9	125	1758	1465	1256	1099
10	125	977	814	698	610
11	125	537	448	384	336
12	125	293	244	209	183

et donc une portée théorique…



Nota : Le temps maximum d'occupation du canal radio (duty-cycle) imposé par l'utilisation de fréquences libres (ISM ici) est un facteur limitant le nombre de paquets pouvant être émis par un équipement. Par exemple, le résultat de la limitation à 1 % sur la bande 868 MHz est un temps de transmission de 36 secondes par heure et par canal pour chaque terminal.

RFM95

Le module de transmission LoRa que nous allons utiliser est un module intégré de la famille RFM9xW.

Les modules sont dotés du circuit Semtech SX1272 qui utilise une variante du codage à spectre étalé Chirp appelée LoRa™. Le module peut également utiliser les types de modulation OOK (On-Off Keying ou tout ou rien) et FSK (Frequency Shift Key) pour maintenir la compatibilité avec d'autres normes telles que IEEE 802.15.4 et wMBUS.

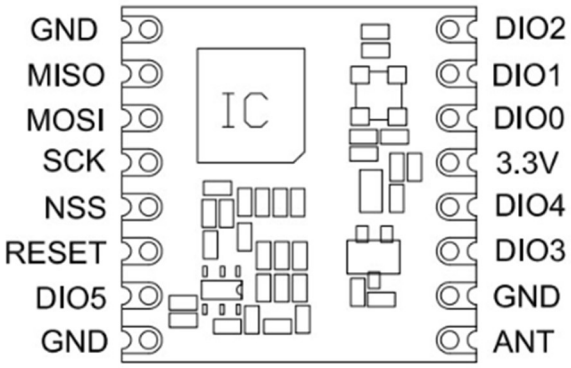
LoRa offre une meilleure immunité aux interférences par rapport à la modulation FSK conventionnelle, améliorant la coexistence sur les canaux encombrés.

Notre modèle est un modèle RFM95.

Caractéristiques du module :

- Plage de fréquences RF : bande ISM 868 / 915 MHz
- Bande passante du canal : 125 kHz, 250 kHz ou 500 kHz
- Puissance de sortie RF : jusqu'à +20 dBm
- Sensibilité du récepteur : -119 dBm @ 1,2 kb/s (FSK), -148 dBm @ 18 b/s (LoRa, SF=12, CR=4/6)
- Modulation : LoRa™, FSK, GFSK, MSK, GMSK ou OOK

- Vitesse de transmission OTA : jusqu'à 250 kb/s (FSK), jusqu'à 37,5 kb/s (LoRa™)
- Portée jusqu'à 15 km (sans obstacles)
- Moteur de paquet jusqu'à 256 octets avec CRC
- Correction d'erreur de transmission (FEC, Forward Error Correction) avec taux de codage sélectionnable
- RSSI à plage dynamique de 127 dB
- Communication hôte : SPI série
- Capteur de température et indicateur de charge de batterie faible
- Alimentation : +1,8 V à +3,6 V c.c.
- Indice de température de fonctionnement : -20 à +70 °C



Il sera interfacé par un bus SPI et alimenté directement en 3.3v par la carte module ESP32 DEVKIT.

https://www.hoperf.com/modules/lora/RFM95.html?/modules/index.html&gclid=EAIaIQobChMI7uilgMmv7gIVQvhRCh21eQIPEAAYASAAEgL-EfD_BwE

(5) Gestion de l'heure NTP et RTC

Afin de rendre l'information compréhensible d'un point de vue temporelle, nous avons placé ici une horloge temps réelle (real-time clock ou RTC) au travers d'un circuit DS1307. Ce circuit a besoin au préalable de se synchroniser par l'intermédiaire d'une horloge précise et disponible.

Citons quelques exemples de systèmes permettant de se synchroniser :

- La montre : pas très précis et manuel
- Le GPS : très précis mais réalisable qu'en extérieur
- Le DCF77 : horloge atomique émise en grande onde 77khz et diffusée depuis l'Allemagne, très précis mais demande une bonne réception et un temps de synchronisation long
- Le NTP ou Network Time Protocol : très précis et présent sur les réseaux informatiques via des serveurs NTP.

Nous utiliserons ici le service NTP pour des raisons pratiques.

Le NTP

Le NTP est un protocole qui transite sur les réseaux informatiques depuis 1985. Il est basé sur des horloges ayant une précision atomique. Cette heure est diffusée sur des serveurs qui à leur tour diffusent l'heure vers des clients. Les algorithmes de diffusion prennent en compte, les temps de transit de l'information, les écarts constatés, le choix du meilleur serveur et les corrections locales de l'horloges internes. On peut donc ainsi atteindre une précision supérieure au 1/10 de microseconde.

Attention, le protocole diffuse l'heure en temps universel uniquement.

Vous pouvez vous synchroniser en vous connectant à un internet (via le WIFI de votre portable an Accès Point par exemple ou sur un réseau ouvert) et accéder au serveur de temps (pool.ntp.org) via des commandes de synchronisation.

Le RTC via le DS1307

L'accès à une source de temps précise n'est pas toujours possible. La disponibilité des réseaux dépend de la qualité de signal. Afin de palier à ces pertes de réseaux, nous allons utiliser un circuit DS1307 permettant de sauvegarder et d'entretenir l'heure. Le DS13007 est une horloge temps réel (aussi appelé "RTC" ou "Real Time Clock"). C'est une horloge numérique autonome qui donne l'heure quand on la lui demande.

Caractéristique du circuit :

- Fonction : clock/Calendar
- Fonctionnalités : année bissextile, NVSRAM, sortie d'onde carré
- Taille mémoire : 56B
- Format horaire : HH:MM:SS (12/24 h)
- Format de date : AA-MM-JJ-jj
- Interface : I²C, 2 fils, série
- Tension - Alimentation : 4,5V ~ 5,5V
- Tension - Alimentation, batterie : 2V ~ 3,5V
- Courant -max. : 200µA à 5V
- Température de fonctionnement : 0°C ~ 70°

Pour fonctionner, il faudra prévoir une batterie qui permettra de sauvegarder l'heure en cas de coupure ainsi qu'un oscillateur de 32. 768khz. La communication se fera via un bus I2C.

L'heure sera programmée en utilisant le code BCD (Binary Coded Decimal) qui est un codage de la base 10 sur un format binaire. Ce code est facilement interprétable pour l'homme mais demande une conversion pour microprocesseur.

Exemple de BCD :

CHIFFRE	BCD		
5	0101		
	5		
15	0001	0101	
	1	5	
153	0001	0101	0011
	1	5	3

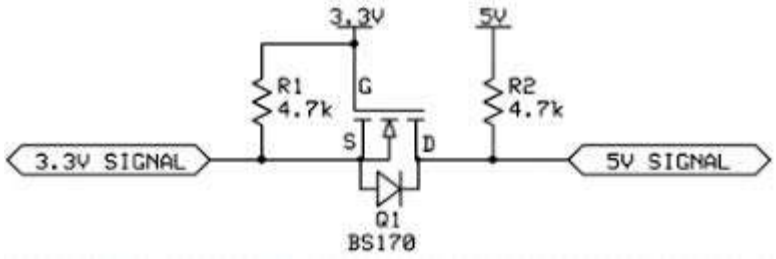
Le DS1307 est joignable à l'adresse (0x68), configurable et consultable grâce à la table de registre suivante:

Table 2. Timekeeper Registers

ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
00h	CH	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12	10 Hour	10 Hour	Hours				Hours	1–12 +AM/PM 00–23
		24	PM/ AM							
03h	0	0	0	0	0	DAY			Day	01–07
04h	0	0	10 Date		Date				Date	01–31
05h	0	0	0	10 Month	Month				Month	01–12
06h	10 Year				Year				Year	00–99
07h	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08h–3Fh									RAM 56 x 8	00h–FFh

0 = Always reads back as 0.

Le DS1307 demande une alimentation en +5V, heureusement la carte ESP32 DEVKIT possède un régulateur et une tension en +5V. Toutefois, afin que les éléments puissent discuter en utilisant les mêmes niveaux de tension, il va être judicieux de prévoir une conversion de niveau comme ci-dessous.



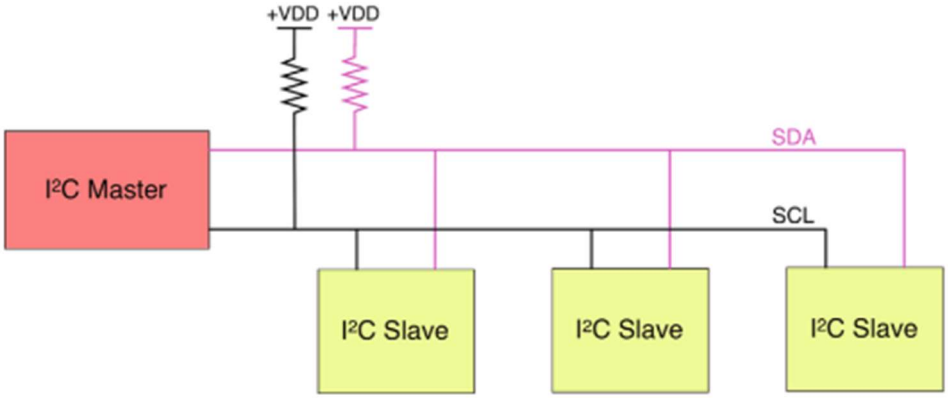
Le bus I2c

Le bus I2c est un bus synchrone développé par Philips afin de faire communiquer le processeur avec les différents circuits des télévisions (Inter-Integrated Circuit).

Il utilise 2 fils uniquement (SDA et SCL) et permet des communications de 100Kbit/s et pouvant aller jusqu'à 400 Kbits en standard.

Il fonctionne sur le principe maitre/esclave. Chaque esclave possède une adresse unique sur le bus. L'horloge est gérée par le maitre (même si l'esclave peut temporairement demander une « pause » du signal).

L'adresse de chaque esclave est soit configurée par le constructeur soit configurable par des pattes du circuit (A0, A1, A2 : 3 bits par exemple).



Les 2 lignes sont tirées au niveau de tension VDD à travers des résistances de tirage (pull-up).

Les deux lignes SDA et SCL ont des sorties de type collecteur ouvert. Les équipements sont donc câblés sur le bus par le principe du « ET câblé », ce qui veut dire qu'en cas d'émission simultanée de deux équipements, la valeur 0 écrase la valeur 1.

Le nombre maximum d'équipements est limité par le nombre d'adresses disponibles, 7 bits d'adressage, soit 128 périphériques. Un bit R/W (lecture ou écriture) indique le sens de communication (maitre→esclave ou esclave→maitre).

Messages :

Tout message est initié et terminé et par le maître

Structure d'un message :

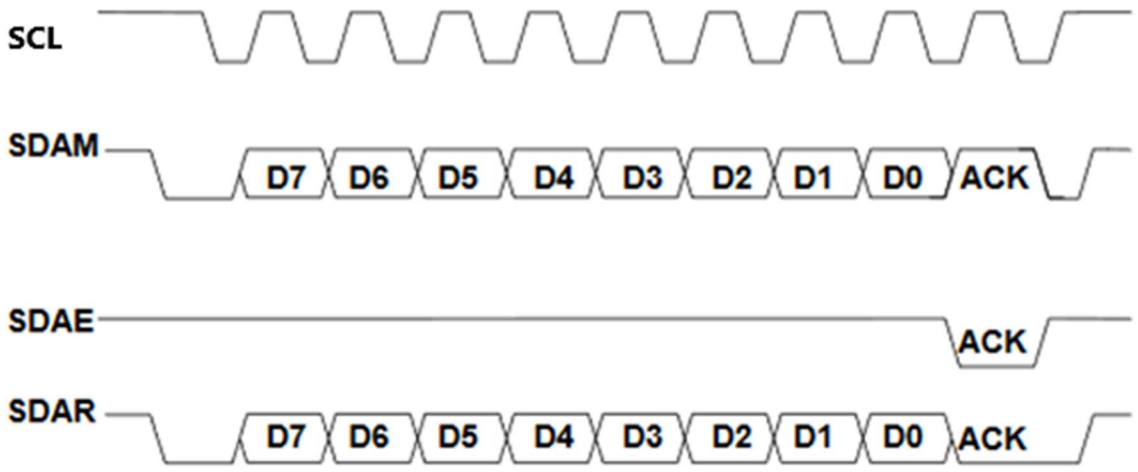
- Commence par une condition START
- Finit par une condition STOP
- Constitué de plusieurs trames: (suite de 8 bits)
  - 1 ou 2 trames d'adresse (à qui on s'adresse)
  - 1 ou plusieurs trames de données (ce qu'on lui dit)

Après avoir vérifié que le bus est libre, puis pris le contrôle de celui-ci, le circuit en devient le maître :

Principe de transmissions d'un octet :

- La condition de départ : SDA passe à 0 et SCL reste à 1
- La condition d'arrêt : SDA passe à 1 et SCL reste à 1
- Le maître transmet le bit de poids fort D7 sur SDA
- SCL tombe à 0 et la donnée est prise en compte en appliquant un niveau 1 sur SCL

- Lorsque SCL retombe à 0, il poursuit avec D6, etc. jusqu'à ce que l'octet complet soit transmis
- Il envoie le bit ACK à 1 en scrutant l'état réel de SDA
- L'esclave doit imposer un niveau 0 pour signaler que la transmission s'est déroulée correctement
- Le maître voit le 0 (collecteur ouvert) et peut passer à la suite



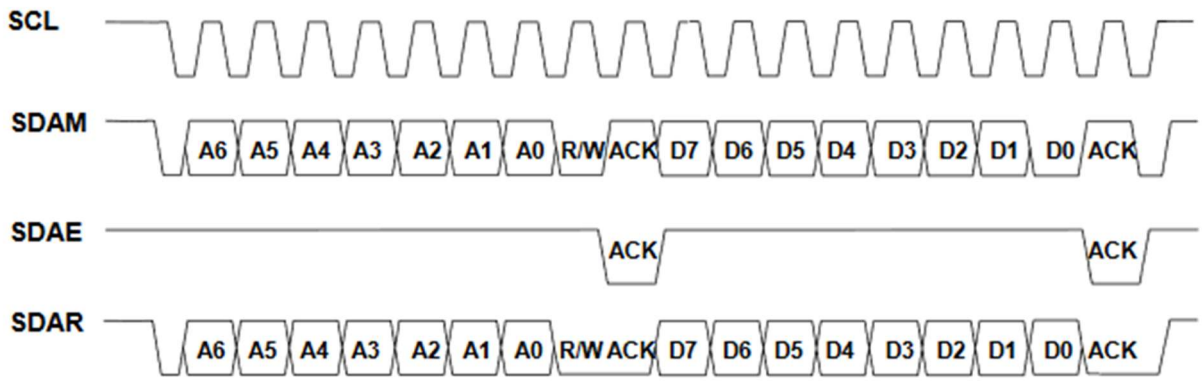
SCL : Horloge imposée par le maître
SDAM : Niveaux de SDA imposés par le maître
SDAE : Niveaux de SDA imposés par l'esclave
SDAR : Niveaux de SDA réels résultants

Accès à l'esclave en écriture de données

En gardant les mêmes principes :

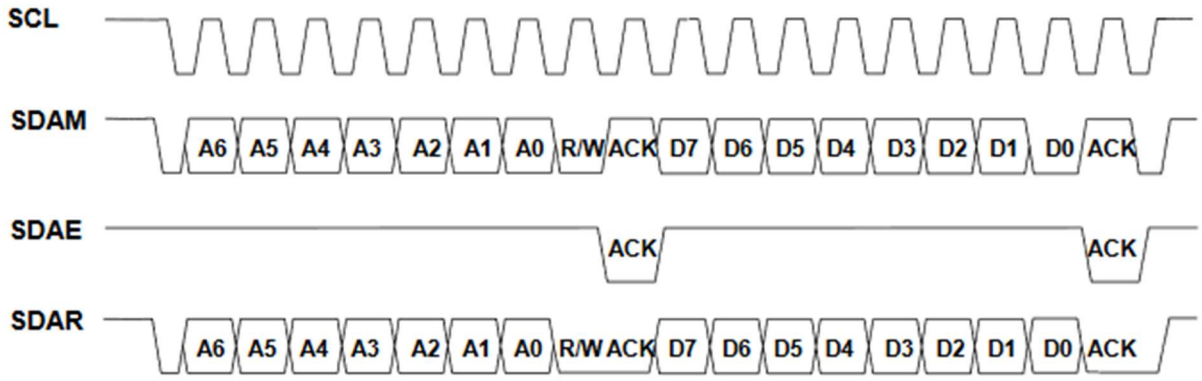
- Envoi de l'adresse de destination
- Sélection du mode écriture (R/W à 0)
- Envoi de la donnée

  Note :il peut être nécessaire d'attendre ACK avant de poursuivre (écriture dans des mémoires, etc.)



Accès à l'esclave en lecture de données

- Sélection du mode écriture (R/W à 1)



Problème

Le bus I2C est de conception destiné à accueillir plusieurs maîtres. Le problème commun à tous les réseaux utilisant un canal de transmission unique → comment arbitrer ?

Chaque maître peut prendre possession du bus dès que celui-ci est libre (possibilité que deux maîtres prennent la parole en même temps).

Pas de problème électrique → collecteur ouvert

Problème logique → éviter la corruption des données due à la collision des bits transmis

Solution

Avant de parler, le protocole vérifie que le bus est libre.

Prise de contrôle effectif, mais vérification de l'état des lignes SDA et SCL. Plusieurs cas :

1. Différents maîtres envoient les mêmes données en même temps : aucun conflit, cas rare.
2. Un maître impose un '0' : il va donc relire obligatoirement un '0' et continuera à transmettre.

3. Un maître cherche à appliquer un ‘1’ sur le bus
    a. S’il lit ‘1’, il continue à transmettre.
    b. S’il lit ‘0’, un autre maître a pris la parole en même temps : il perd l’arbitrage, il arrête d’émettre, mais continue à lire.

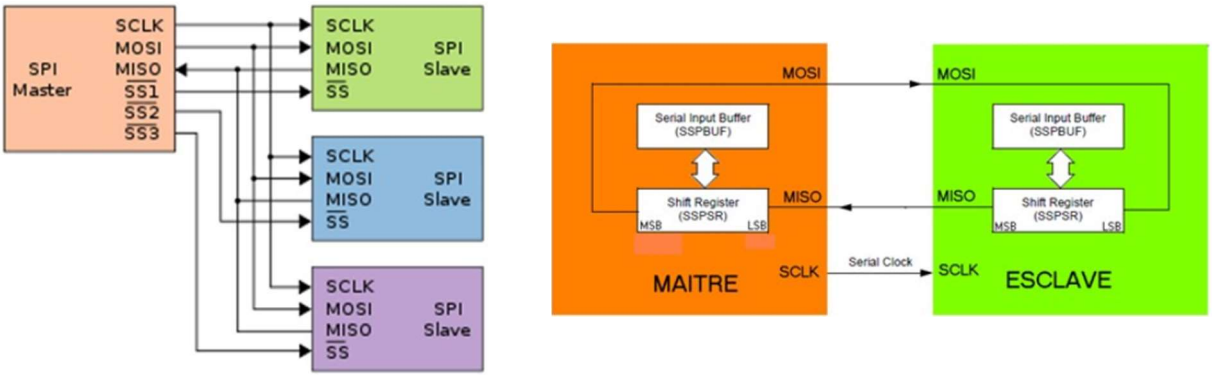
Le bus SPI

Un bus SPI (Serial Peripheral Interface) est un bus de données 8 bits série synchrone qui opère en mode full-duplex (émission et réception simultanée). Les circuits communiquent selon un schéma maître-esclave.

Plusieurs esclaves peuvent coexister sur un même bus, dans ce cas, la sélection du destinataire se fait par une ligne appelée Slave Select (SS).

Quatre signaux sont nécessaires à la transmission des données.

- SCLK — Serial Clock, Horloge (généré par le maître)
- MOSI — Master Output, Slave Input (généré par le maître)
- MISO — Master Input, Slave Output (généré par l'esclave)
- SS — Slave Select, Actif à l'état bas (généré par le maître)



Les données transitent aux coups d’horloge et passent par les buffers ou registre à décalage du sous ensemble sélectionné. A la première série de coups d’horloge (8 ou 9 fronts d’horloge), le maitre envoie un ordre à l’esclave. A la seconde série de coups d’horloge (8 ou 9 fronts d’horloge), l’esclave répond au maitre pendant que le maitre peut à nouveau parler.

Les collisions de données sont évitées grâce à la sélection du boitier par le signal SS. Les boitiers qui ne sont pas sélectionnées se mettent en hautes impédances (circuit ouvert).

Ce bus est spécifique aux communications courtes distances inter composants et peut atteindre 20Mbits/s. En revanche, il n’y a pas de protocole qui permettent de contrôler l’informations (erreur possible) et le bus ne tolère qu’un seul maitre. Ce n’est donc pas à proprement dit un protocole.

Autre difficulté du bus SPI, il n’est pas normalisé. Initialement créé par Motorola, rapidement d’autres constructeurs ont optés pour ce type de liaison. Conséquence, on retrouve sur le marché un grand nombre de signaux et des chronogrammes différents.

Cependant, les constructeurs de microprocesseur et de kit type « Arduino » ont prévu des tables de configuration et des bibliothèques permettant de gérer les différents modes de transmissions.

Mode SPI	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

Le CPOL détermine si au repos l'horloge est au niveau BAS (CPOL=0) ou HAUT (CPOL=1)

Le CPHA détermine à quel front de l'horloge les données sont transmises. CPHA=0 les données sont valides au premier front d'horloge,CPHA=1 elles sont valides au deuxième front.

Cas 1 : CPHA = 0, les données sont valides au premier front du signal d'horloge. La polarité CPOL détermine s'il s'agit d'un front montant ou descendant.

Pour CPOL=0, au repos, l'horloge est au niveau BAS; le premier front est donc un front montant.

Pour CPOL=1, au repos, l'horloge est sur le niveau HAUT; le premier front est donc un front descendant.

La polarité de l'horloge n'ayant pas d'influence sur le moment où le premier bit de données est valide elle n'a pas d'effet sur le format du transfert de données (voir figure ci-dessous).

Cas 2 : CPHA = 1, les données sont réceptionnées avec le deuxième front du signal d'horloge.

Pour CPOL=0, au repos, l'horloge est au niveau BAS et monte au niveau HAUT après le premier front, le deuxième front est donc un front descendant.

Pour CPOL=1, au repos, l'horloge est au niveau HAUT et descend au niveau BAS après le premier front; le deuxième front est donc un front montant.

SI7034

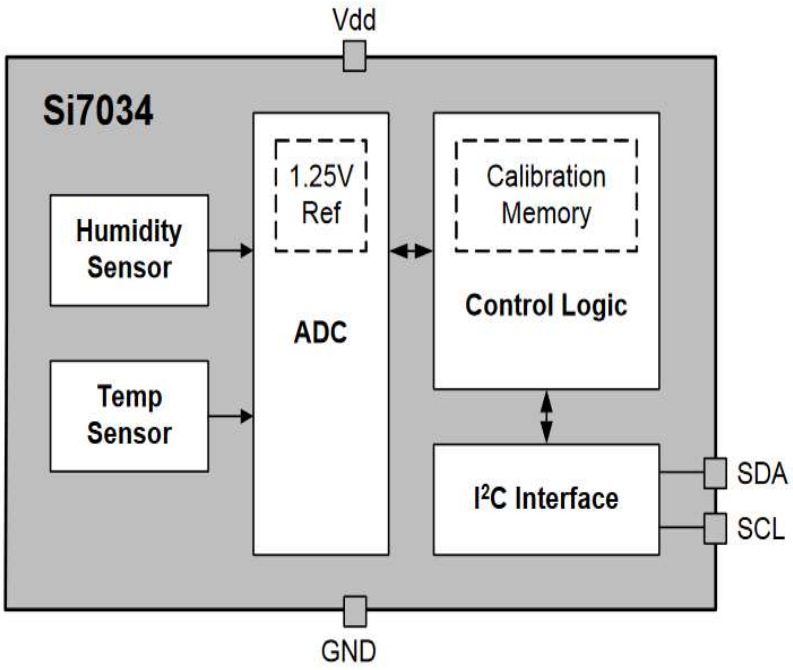
La famille Si70xx de sondes d'humidité et de température numériques comportent une sonde d'humidité, une sonde de température et une interface série I2C, étalonnage et conditionnement de signal intégré dans un boîtier CMOS QFN/DFN compact.

Quand l'air entre en contact avec la sonde en polyamide exposée, l'humidité dans l'air provoque un changement de capacité dans la sonde que le conditionnement intégré convertit en relevé d'humidité. Ceci est ensuite transmis à l'utilisateur par le biais de l'interface I2C.

Caractéristiques :

- Plage de tension d'utilisation étendue : 1,7 à 2 V
- Plage de température de fonctionnement : -40 → +125°C
- Précision : 0.4 °C, ±4%HR
- Résolution : 16 bits (humidité), 16 bits (température)

- Faible consommation 150uA, 60nA standby current
- Interface série I2C
- Chauffage sur puce
- Excellente stabilité à long terme
- Etalonnage complet
- Filtre de protection pré-installé
- 2*2 mm



Le capteur de température sera alimenté en 1.8V, un régulateur de tension (MCP1700T-180) permet de créer cette tension à partir du 3.3V de la carte ESP32. De même il faudra un translateur de tension sur le bus I2C, entre le module ESP32 et le capteur SI7034, cette fonction est réalisée par le circuit PCA9306.

Afficheur LCD

Un afficheur LCD (Liquid Crystal Display) doit permettre d'afficher les différentes valeurs d'humidité, de température et de pression. Cet afficheur doit pouvoir s'alimenter en 3.3V. Nous prendrons un LCD alphanumérique de 4 lignes de 20 caractères, avec interface parallèle.

http://www.farnell.com/datasheets/50586.pdf