****

**Procédure de Production d’un Dongle Linky**

Dongle Linky avec envoi sur Datum

**IMREDD | 2024**

Authors: Benoit Couraud

1. **Compteur Linky**

Les specifications du compteur Linky sont disponibles dans le document enedis Enedis-NOI-CPT\_54E. Les informations importantes sont les suivantes :

* Il existe 2 modes pour le compteur Linky. Le mode historique, qui fournit des données limitées et qui propose un débit de communication faible (1200 bauds). Le mode standrad qui fournit un ensemble de données plus grand, et à un débit plus élevé (9600 bauds). Le compteur est mis dans un mode ou dans l’autre suite à une demande faite par le client à Enedis. L’image ci-dessous présente les données disponibles en mode historique (à gauche) et une partie des données disponibles pour le compteur en mode standard. Il est intéressant de regarder la partie puissance. En trame standard, on peut accéder à la puissance instantanée (SINST) soutirée, et injectée. En mode historique, on ne peut que récupérer la puissance apparente soutirée. Ainsi, pour un prosumer, il nous faudra passer le compteur en mode standard.

A white sheet with black text

Description automatically generated

A screenshot of a document

Description automatically generated A blue and white table with text

Description automatically generated

* La donnée est transmise via une liaison série modulée par une porteuse comme indiqué ci-dessous.

A diagram of a waveform

Description automatically generated



En mode standard, voici la configuration série nécessaire :

* + - Débit : 9600 baud
    - un bit de start correspondant à un "0" logique
    - 7 bits pour représenter le caractère en ASCII
    - 1 bit de parité
    - Parité paire
    - Un bit de stop correspondant à un « 1 » logique.

En mode historique, la configuration est la même, mais le débit est de 1200 baud.

* Le compteur peut alimenter un petit appareil de faible puissance, comme indiqué ci-dessous. Pour rappel, 130mW ne permet pas d’alimenter un module de communication qui consommera en effet 400-500mW lors de l’émission d’un message. En LoRa cependant (ou Zigbee), la consommation est bien inférieure, jusqu’à 60 mW.

A blue and white text on a white background

Description automatically generated

1. **Architecture du dongle Linky**

Le dongle qui se connecte sur les compteurs électriques Linky a l’architecture suivante :

A diagram of a circuit

Description automatically generated

Il se connecte sur les 3 sorties I1, I2, A de Linky. Le lien I1-A fournit l’alimentation au dongle tandis que le lien I1-I2 fournit les données.

On peut voir les différentes parties du dongle :

1. La partie démodulation qui s’occupe de traduire le signal envoyé par le compteur Linky. Le compteur module un signal UART (liaison série) à partir d’une porteuse comme indiqué dans la note Enedis-NOI-CPT\_54E. Configuration du dongle en mode standard (9600 bit/s) ou historique (1200 bit/s) se fait en changeant la configuration de lecture Rx par le module de communication & control, ainsi que par un changement de résistance (passage de 12k à 22k pour un signal à 9600 bit/s).
2. Le module de communication et contrôle, qui est un module Wifi ou LoRa dont le codage peut se faire via l’environnement Arduino. C’est lui qui décodera le signal du compteur Linky, veillera à détecter les informations de puissance apparente ou énergie consommée, et d’envoyer les informations à un serveur. Une fois alimenté, ce module va lire la donnée venant du Linky, et l’envoyer à un serveur si la tension de la super capacité du module d’alimentation est suffisamment grande.
3. Le module d’alimentation, qui nécessite une super capacité afin de permettre les sous fonctions suivantes :
   1. Redresser la porteuse afin de pouvoir utiliser une tension continue pour charger et alimenter le système
   2. Une grosse batterie qui va se charger lorsque l’on va l’exposer à cette tension redressée. Il faut faire attention au fait que la tension fournie par le compteur est de 6Vrms, qui sera ensuite redressé (un peu plus haut que 6V) ce qui doit être pris en compte dans le choix de la super capacité qui doit être capable d’accepter 5-7V, tout en ayant une grande capacité (~1.5-3F).
   3. La partie communication du dongle ne doit communiquer des données qu’à condition qu’il y ait suffisamment d’énergie dans la batterie pour permettre un envoi. Cette information peut être connue grâce à la tension de la super capacité. Il faut alors un système permettant de passer le module à ON lorsqu’il y a suffisamment d’énergie dans la capacité. Cela peut difficilement se faire de manière analogique car il faudrait alors un hysteresis. En effet, la tension de la capacité augmente lorsqu’on la charge. Elle peut augmenter jusqu’à une certaine valeur à partir de laquelle nous connecterons la capacitéau module de communication. Mais lorsque la capacité est connectée à une charge, sa tension diminue. Sans un hysteresis dans la mesure de tension de la capacité, alors la charge (le module de communication) sera à nouveau déconnecté avant de pouvoir envoyer le moindre message.
   4. Ainsi, cet hysteresis est fourni grâce à un microcontroleur (PIC) à faible consommation qui va patiemment attendre que la tension de la capacité soit suffisamment grande avant de connecter le module de communication. Cela nécessite malheureusement de programmer un autre microcontroleur.
   5. Le role du module d’alimentation est de connecter la masse (borne moins) de la super capcité avec la masse du module de communication lorsque la tension de la super capacité est suffisamment grande. Cela a pour role d’alimenter le module de communication .

Ce document explique comment produire ces dongle. Cela nécessite les étapes suivantes :

* Commande des PCBs
* Commande des composants
* Soudage des composants
* Programmation du microcontroleur PIC
* Programmation du module de communication
* Connexion de toutes les parties
* Configuration du serveur

L’intégralité des fichiers pour répliquer le dongle se trouve sur le gitlab : git clone <https://tip-imredd.unice.fr/code/energy/dongle_linky.git>

1. **Commande des PCBs**

Le PCB réalisé est disponible ici : <https://tip-imredd.unice.fr/code/energy/dongle_linky/-/tree/master/PCB?ref_type=heads>

Le schematic est disponible ci-dessous :

A diagram of a circuit board

Description automatically generated

1. **Commande des composants**

Les composants peuvent être fournis sur rs-components.

A screenshot of a computer

Description automatically generated

Mdp :imredd

On peut alors voir la commande finale possible pour les dongle :

A screenshot of a computer

Description automatically generated

**Produits dans cette liste**

| **Votre remise vous permet d'économiser :  4,36 €** | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sélection | Code commande | Qté. | Description | Prix HT pour | UDV en cours | | Produits | Effacer |
| Prix HT catalogue | Prix contractuel |
|  |  |  | Diode CMS Vishay, 1A, 100V, DO-214AA (SMB)  [Diode CMS Vishay, 1A, 100V, DO-214AA (SMB)](https://fr.rs-online.com/web/p/products/7103046/)  **20 En stock pour livraison dès le lendemain**  Conforme RoHS | l'unité (par multiple de 20) | 0,071 € | **0,06 €** | 1,21 € | [labelNotFoundf1/PartViewMsgRemoveAll](https://fr.rs-online.com/web/cart/partsList.html?method=removePartsListItem&partsListItem=7103046&partsListId=322108706) |
|  |  |  | Régulateur de tension, MCP1825S-3302E/DB, 500mA, SOT-223 3+Tab broches.  [Régulateur de tension, MCP1825S-3302E/DB, 500mA, SOT-223 3+Tab broches.](https://fr.rs-online.com/web/p/products/6695092/)  **5 En stock pour livraison dès le lendemain**  Conforme RoHS | l'unité (par multiple de 5) | 0,612 € | **0,612 €** | 3,06 € | [labelNotFoundf1/PartViewMsgRemoveAll](https://fr.rs-online.com/web/cart/partsList.html?method=removePartsListItem&partsListItem=6695092&partsListId=322108706) |
|  |  |  | Microcontrôleur, 8bit, 256 (RAM/EEPROM) B RAM, 7 ko, 32MHz, SOIC 8, série PIC12F  [Microcontrôleur, 8bit, 256 (RAM/EEPROM) B RAM, 7 ko, 32MHz, SOIC 8, série PIC12F](https://fr.rs-online.com/web/p/products/7434703/)  **4 En stock pour livraison dès le lendemain**  Conforme RoHS | l'unité (par multiple de 4) | 1,37 € | **1,37 €** | 5,48 € | [labelNotFoundf1/PartViewMsgRemoveAll](https://fr.rs-online.com/web/cart/partsList.html?method=removePartsListItem&partsListItem=7434703&partsListId=322108706) |
|  |  |  | MOSFET onsemi canal N, SOT-23 2,2 A 30 V, 3 broches  [MOSFET onsemi canal N, SOT-23 2,2 A 30 V, 3 broches](https://fr.rs-online.com/web/p/products/6710429/)  **10 En stock pour livraison dès le lendemain**  Conforme RoHS | l'unité (par multiple de 10) | 0,392 € | **0,333 €** | 3,33 € | [labelNotFoundf1/PartViewMsgRemoveAll](https://fr.rs-online.com/web/cart/partsList.html?method=removePartsListItem&partsListItem=6710429&partsListId=322108706) |
|  |  |  | Optocoupleur Montage en surface Vishay, Sortie Transistor 63%  [Optocoupleur Montage en surface Vishay, Sortie Transistor 63%](https://fr.rs-online.com/web/p/products/7104913/)  **10 En stock pour livraison dès le lendemain**  Conforme RoHS | l'unité (par multiple de 10) | 0,611 € | **0,519 €** | 5,19 € | [labelNotFoundf1/PartViewMsgRemoveAll](https://fr.rs-online.com/web/cart/partsList.html?method=removePartsListItem&partsListItem=7104913&partsListId=322108706) |
|  |  |  | Diode Zener Nexperia, 3.3V, dissip. ≤ 830 mW SOD-123F  [Diode Zener Nexperia, 3.3V, dissip. ≤ 830 mW SOD-123F](https://fr.rs-online.com/web/p/products/7920936/)  **10 En stock pour livraison dès le lendemain**  Conforme RoHS | l'unité (en bobine de 10) | 0,024 € | **0,02 €** | 0,20 € | [labelNotFoundf1/PartViewMsgRemoveAll](https://fr.rs-online.com/web/cart/partsList.html?method=removePartsListItem&partsListItem=7920936&partsListId=322108706) |
|  |  |  | MOSFET DiodesZetex canal N, SOT-23 150 mA 60 V, 3 broches  [MOSFET DiodesZetex canal N, SOT-23 150 mA 60 V, 3 broches](https://fr.rs-online.com/web/p/products/2156587/)  **5 En stock pour livraison dès le lendemain**  Conforme RoHS | l'unité (par multiple de 5) | 0,634 € | **0,539 €** | 2,69 € | [labelNotFoundf1/PartViewMsgRemoveAll](https://fr.rs-online.com/web/cart/partsList.html?method=removePartsListItem&partsListItem=2156587&partsListId=322108706) |
|  |  |  | Supercondensateur, 2.5F, 6V c.c., Traversant  [Supercondensateur, 2.5F, 6V c.c., Traversant](https://fr.rs-online.com/web/p/products/2506984/)  **1 En stock pour livraison sous 3 jour(s)**  Conforme RoHS | la pièce | 5,54 € | **4,709 €** | 4,71 € | [labelNotFoundf1/PartViewMsgRemoveAll](https://fr.rs-online.com/web/cart/partsList.html?method=removePartsListItem&partsListItem=2506984&partsListId=322108706) |
|  |  |  | Résistance CMS Vishay série CRCW 0Ω, boitier 0805 (2012M) ±0% 0.5W  [Résistance CMS Vishay série CRCW 0Ω, boitier 0805 (2012M) ±0% 0.5W](https://fr.rs-online.com/web/p/products/8123353/)  **25 En stock pour livraison dès le lendemain**  Conforme RoHS | L'unité (en sachet de 25) | 0,095 € | **0,081 €** | 2,02 € | [labelNotFoundf1/PartViewMsgRemoveAll](https://fr.rs-online.com/web/cart/partsList.html?method=removePartsListItem&partsListItem=8123353&partsListId=322108706) |
|  |  |  | Résistance CMS Panasonic série ERJ 12kΩ, boitier 0805 (2012M) ±1% 0.125W  [Résistance CMS Panasonic série ERJ 12kΩ, boitier 0805 (2012M) ±1% 0.125W](https://fr.rs-online.com/web/p/products/8654245/)  **100 En stock pour livraison sous 3 jour(s)**  Conforme RoHS | L'unité (en sachet de 100) | 0,012 € | **0,01 €** | 1,00 € | [labelNotFoundf1/PartViewMsgRemoveAll](https://fr.rs-online.com/web/cart/partsList.html?method=removePartsListItem&partsListItem=8654245&partsListId=322108706) |
|  |  |  | Résistance CMS Panasonic série ERA 22kΩ, boitier 0805 (2012M) ±0.1% 0.125W  [Résistance CMS Panasonic série ERA 22kΩ, boitier 0805 (2012M) ±0.1% 0.125W](https://fr.rs-online.com/web/p/products/0566226P/)  **5 En stock pour livraison sous 3 jour(s)**  Conforme RoHS | l'unité (conditionné en bobine) | 0,12 € | **0,102 €** | 0,51 € | [labelNotFoundf1/PartViewMsgRemoveAll](https://fr.rs-online.com/web/cart/partsList.html?method=removePartsListItem&partsListItem=0566226P&partsListId=322108706) |
|  |  |  | Résistance CMS Panasonic série ERJP06 10kΩ, boitier 0805 (2012M) ±1% 0.5W  [Résistance CMS Panasonic série ERJP06 10kΩ, boitier 0805 (2012M) ±1% 0.5W](https://fr.rs-online.com/web/p/products/7217820P/)  **5 En stock pour livraison sous 3 jour(s)**  Conforme RoHS | l'unité (conditionné en bobine) | 0,176 € | **0,15 €** | 0,75 € | [labelNotFoundf1/PartViewMsgRemoveAll](https://fr.rs-online.com/web/cart/partsList.html?method=removePartsListItem&partsListItem=7217820P&partsListId=322108706) |
|  |  |  | Résistance CMS Vishay série CRCW 9.1kΩ, boitier 0805 (2012M) ±1% 0.125W  [Résistance CMS Vishay série CRCW 9.1kΩ, boitier 0805 (2012M) ±1% 0.125W](https://fr.rs-online.com/web/p/products/6791755/)  **50 En stock pour livraison dès le lendemain**  Conforme RoHS | l'unité (par multiple de 50) | 0,031 € | **0,026 €** | 1,33 € | [labelNotFoundf1/PartViewMsgRemoveAll](https://fr.rs-online.com/web/cart/partsList.html?method=removePartsListItem&partsListItem=6791755&partsListId=322108706) |
|  |  |  | Résistance CMS Yageo 1kΩ, boitier 0805 (2012M) ±5% 0.25 W, 0.125 W  [Résistance CMS Yageo 1kΩ, boitier 0805 (2012M) ±5% 0.25 W, 0.125 W](https://fr.rs-online.com/web/p/products/1995762/)  **100 En stock pour livraison dès le lendemain** | l'unité (en bande de 100) | 0,007 € | **0,006 €** | 0,59 € | [labelNotFoundf1/PartViewMsgRemoveAll](https://fr.rs-online.com/web/cart/partsList.html?method=removePartsListItem&partsListItem=1995762&partsListId=322108706) |
|  |  |  | Résistance CMS Panasonic série ERJP06 1kΩ, boitier 0805 (2012M) ±1% 0.5W  [Résistance CMS Panasonic série ERJP06 1kΩ, boitier 0805 (2012M) ±1% 0.5W](https://fr.rs-online.com/web/p/products/7217741/)  **5 En stock pour livraison dès le lendemain**  Conforme RoHS | l'unité (par multiple de 5) | 0,176 € | **0,15 €** | 0,75 € | [labelNotFoundf1/PartViewMsgRemoveAll](https://fr.rs-online.com/web/cart/partsList.html?method=removePartsListItem&partsListItem=7217741&partsListId=322108706) |
|  |  |  | Résistance CMS KOA série RK73H 270Ω, boitier 0805 (2012M) ±1% 0.25W  [Résistance CMS KOA série RK73H 270Ω, boitier 0805 (2012M) ±1% 0.25W](https://fr.rs-online.com/web/p/products/6314557/)  **50 En stock pour livraison dès le lendemain**  Conforme RoHS | l'unité (par multiple de 50) | 0,004 € | **0,003 €** | 0,15 € | [labelNotFoundf1/PartViewMsgRemoveAll](https://fr.rs-online.com/web/cart/partsList.html?method=removePartsListItem&partsListItem=6314557&partsListId=322108706) |
|  |  |  |  |  |  |  |  |  |

1. **Programmation du microcontrôleur PIC**

Les microcontroleurs PIC nécessitent un programmateur. J’ai utilisé un PicKit 3.

Il faut également un logiciel spécial pour programmer le PIC. On peut installer MPLAB X IDE : <https://www.microchip.com/en-us/tools-resources/develop/mplab-x-ide>

A screenshot of a computer

Description automatically generated

L’installation nécessite les étapes suivantes :

A screenshot of a computer

Description automatically generated A screenshot of a computer program

Description automatically generated A screenshot of a computer program

Description automatically generated A white box with black text

Description automatically generated

Vous pouvez alors ouvrir le projet disponible en ouvrant MPLAB :

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Dans le main.c, collez le code ci-dessous :

A screenshot of a computer

Description automatically generated

/\*

\* File: main.c

\* Author: Benoit Couraud

\*

\* Created on February 7, 2023, 3:28 PM

\*/

#include <xc.h>

// CONFIG1

#pragma config FOSC = INTOSC // Oscillator Selection (INTOSC oscillator: I/O function on CLKIN pin)

#pragma config WDTE = ON // Watchdog Timer Enable (WDT disabled)

#pragma config PWRTE = OFF // Power-up Timer Enable (PWRT disabled)

#pragma config MCLRE = ON // MCLR Pin Function Select (MCLR/VPP pin function is MCLR)

#pragma config CP = OFF // Flash Program Memory Code Protection (Program memory code protection is disabled)

#pragma config CPD = OFF // Data Memory Code Protection (Data memory code protection is disabled)

#pragma config BOREN = ON // Brown-out Reset Enable (Brown-out Reset enabled)

#pragma config CLKOUTEN = OFF // Clock Out Enable (CLKOUT function is disabled. I/O or oscillator function on the CLKOUT pin)

#pragma config IESO = ON // Internal/External Switchover (Internal/External Switchover mode is enabled)

#pragma config FCMEN = ON // Fail-Safe Clock Monitor

// CONFIG2

#pragma config WRT = OFF // Flash Memory Self-Write Protection (Write protection off)

#pragma config PLLEN = ON // PLL Enable (4x PLL enabled)

#pragma config STVREN = ON // Stack Overflow/Underflow Reset Enable (Stack Overflow or Underflow will cause a Reset)

#pragma config BORV = LO // Brown-out Reset Voltage Selection (Brown-out Reset Voltage (Vbor), low trip point selected.)

#pragma config LVP = OFF // Low-Voltage Programming Enable (High-voltage on MCLR/VPP must be used for programming)

#define \_XTAL\_FREQ 1000000 // Clock a 1MHz

#include <xc.h>

#include <math.h>

unsigned char threshold1 = 180;

unsigned char threshold2 = 187;

char thresholdCounter = 20;//5; //20;

char thresholdCounterAcquisition = 220;//10; //220;

char counterAcquisition = 1;

unsigned char adc\_char;

unsigned char ADC\_Lecture()

{

ADCON0bits.ADON = 1;// &= 0b00000001; //Clearing the Channel Selection Bits

//ADCON0 |= channel<<2; //Setting the required Bits

\_\_delay\_ms(6);

ADCON0bits.GO = 1; //Initializes A/D Conversion

while(ADCON0bits.GO\_nDONE); //Wait for A/D Conversion to complete

//unsigned int output = (ADRESH<<8)+ADRESL;

return (ADRESH); //((ADRESH<<8)+ADRESL); //Returns Result

}

void variable\_delay(int attente)

{

int i;

for (i = 0; i < attente; i++)

{

\_\_delay\_ms(10);

}

}

char counter;

char uart\_data;

char acquisition = 0;

void main(void) {

counter = 1;

//Configure Watchdog to approx 1 sec

CLRWDT();

OPTION\_REGbits.PSA = 1; //prescaler is for wdt

OPTION\_REGbits.PS = 0b111; //prescaler 1:256

INTCON = 0; //disable every interrupt

PIE1=0;

PIR1=0;

INTCONbits.GIE = 0;

// COnfiguration de l'oscillateur

OSCCON = 0b01011010; //configuration clock 1MHz

OSCCONbits.SCS1 = 1; // Selection de l'oscillateur interne (HF)

// OSCCONbits.IRCF = 0b1011; // Oscillateur interne � 1MHz

// Configuration des ports entr�e-sortie

TRISA = 0b00000001; // PIN RA0 configur� en input, les autres en output

LATA = 0x00; // Tous les PIN en �tat bas

ANSELAbits.ANSA0=1;

ADCON1bits.ADCS=000;

ADCON1bits.ADPREF =00;

ADCON0bits.CHS =0000;

ADCON1bits.ADFM=0;

//######################### mainloop #####################################

// INIT UART

APFCONbits.RXDTSEL = 1; //Select RX on RA5 (instead of RA0)

APFCONbits.TXCKSEL = 1; //Select TX on RA5 (instead of RA1)

APFCONbits.P1BSEL = 0 ;

TRISAbits.TRISA4 = 0; // RA4 = TX= output

TRISAbits.TRISA5 = 1; // RA5 = RX = INPUT

//Configuration Data Rate

BAUDCONbits.BRG16 = 1;//0;

SPBRG = 25; // = Fosc/(16\*DataRate)-1; Fosc = 1MHz et DataRate = 9600

//SPBRGH=25>>8;

//SPBRGL = 25&0xFF;

TXSTA = 0b00110110; // Asynchron TX config, sur 8bits,

TXSTAbits.BRGH = 1;

TXSTAbits.SYNC = 0;

RCSTA = 0b10010000; // Asynchron RX config, sur 8 bits

ANSELA = 0x00; // pas d'analog

\_\_delay\_ms(100);

do{

LATAbits.LATA1 =0;

SLEEP();

NOP();

if (acquisition==1)

{

// TXREG = 111;

//while(!TXSTAbits.TRMT);

counterAcquisition = counterAcquisition+1;

// TXREG = counterAcquisition ;

// while(!TXSTAbits.TRMT);

CLRWDT();

if (counterAcquisition>thresholdCounterAcquisition)

{

counterAcquisition = 1;

LATAbits.LATA1 =1;

adc\_char = ADC\_Lecture();

LATAbits.LATA1 =0;

if (adc\_char>threshold2){

acquisition=0;

}

}

}else // start of the microcontroller

{

CLRWDT();

// we start a lecture

LATAbits.LATA1 =1;

adc\_char = ADC\_Lecture();

// TXREG = 222;

// while(!TXSTAbits.TRMT);

// TXREG = adc\_char;

// while(!TXSTAbits.TRMT); // we send the result through UART

LATAbits.LATA1 =0;

if (adc\_char>threshold1) // if the voltage measured at the diode is above threshold 1: we are

// at a stage where the voltage of the battery is below 3.5V, so we do not start to think to switch on

{

counter = counter - 1;

if (counter<1)

{

counter = 1;

}

}else // if the voltage is below the threshold, this means the voltage at the diode is

// kept at 3.2V, whereas the supply voltage is above (4.5V)

{

counter = counter +1;

if (counter>thresholdCounter) // we count the number of time the supply voltage is large enough

{

counter = thresholdCounter;

}

// SLEEP();

}

CLRWDT();

if (counter == thresholdCounter) // when there has been a long enough time that the supply voltage is above

// 4.5V:

{

acquisition = 1; // we say that the "acquisition" starts: acquisition = we count a given

// number of cycles

// TXREG = 11;

// while(!TXSTAbits.TRMT);

// TXREG = adc\_char;

// while(!TXSTAbits.TRMT);

LATAbits.LATA2=1;

}else

{

// TXREG = counter;

// while(!TXSTAbits.TRMT);

LATAbits.LATA2=0;

}

}

}while(1);

}

Vous pouvez alors compiler :

A screenshot of a computer

Description automatically generated

Vous pouvez aussi configurer le pickit 3, et aller dans les options d’alimentation, et assurez vous que la case spécifiant que vous alimentez le device avec le programmateur est bien cochée.

A screenshot of a computer

Description automatically generated

Vous pouvez alors connecter le Pickit 3 à votre ordinateur, et actualiser pour voir l’outil apparaitre dans la liste des programmateurs.

Vous pouvez alors brancher le pickit sur le PCB via les PIN du bas, dont celui de gauche correspond au pin 1 du pickit :

A circuit board with many small red and yellow squares

Description automatically generated

A diagram of a circuit board

Description automatically generated

A black and white image of a graph

Description automatically generated with medium confidence

Le PCB permet au PIN 1 du pickit (MCLR/Vpp) d’être connecté au Vdd du Pic (pin 4 du PIC).

Pour programmer le PIC :

A screenshot of a computer

Description automatically generated

1. **Programmation du module de communication**

On peut désormais flasher le module de com avec notre code de contrôle.

Vous pouvez télécharger l’environnement Arduino.

Vous pouvez alors connecter la carte WeMo a votre ordinateur, et renseigner l’environnement du modèle avec lequel vous travaillerez :

A screenshot of a computer program

Description automatically generated A screenshot of a computer

Description automatically generated

Avec cette url pour ajouter la bonne board :

<https://arduino.esp8266.com/stable/package_esp8266com_index.json>

Ensuite, allez ajouter votre board :

A screenshot of a computer

Description automatically generated

Puis dans le code « 2024\_06\_03\_Appartements\_Linky\_Wemos\_1200 », configurez le WIFI et mdp :

#define STASSID "WIFI\_SSID\_LiveboxXXXX"

#define STAPSK "WIFI\_PASSWORD"

Ainsi que le nom de l’habitation concernée :

String httpRequestData = "linkySensor,sensor\_id=LinkyHouse2 power="+String(puissanceCommuniquee)+",danger="+String(Danger)+",voltage=234.231,frequency=50.2,pf=0.9 "+String(epoch\_time)+"000000000";

Notez que si le code doit être adapté pour un Linky en mode standard :

Il y a une fonction readLinky9600() qui devra être utilisée en lieu et place de readLinky1200().

Aussi, la ligne de configuration de l’UART devra être revue :

        Serial.begin(1200,SERIAL\_7E1);    // Define and start serial monitoring of the Linky TIC

En modifiant le 1200 en 9600.

Enfin, le code peut envoyer les données sur un autre serveur que la TIP, notamment en remplaçant la fonction setup\_wifi\_TIP(); par la fonction setup\_wifiChargeAngels() qu’il faudra décommenter.

1. **Configuration Coté Serveur**

Notre serveur va recevoir des données qu’il va devoir décoder et envoyer dans notre base de données. Cette fonction est réalisée dans le NodeRed du serveur :

<https://tip-imredd.unice.fr/nodes/>

Utilisateur : admin

Mot de passe: BCI9L8biW131f9J2ANsG

Dans l’onglet APIs, il y a un code pour récupérer les données Linky :

A screenshot of a computer

Description automatically generated

Les données décodées sont alors poussées sur l’instance InfluxDB de Datum/TIP.

A screenshot of a computer

Description automatically generated

node.status(new Date().toUTCString());

msg.headers = { "Authorization": "Token 04H\_vigGv-H8F1cZwOqSR5pZr77dwaPXotiU\_Bi7BqofAbbLGG1oUzhQc\_oXZdkEPhrk0zmtAh-MvqnCGBivZA==" };

return msg;

1. **Evolution Access Point**

Nous avons également travaillé sur la configuration du module en mode point d’accés internet afin de fournir un site web qui permet aux gens de rentrer leur wifi SSID et mdp, les stocker dans l’EEPROM, qui pourra ensuite être réutilisé au lieu d’avoir le wifi stocké en dur dans le code :

Une fois le code flashé sur un module ESP Wemo, On peut alors se connecter au wifi « WiFiWIN » créé avec le mdp : password

On tape l’URL : 192.168.1.1

Et on obtient :

A screenshot of a computer

Description automatically generated

Code WEMO pour cela :

/\*

 \* This code will configure ESP8266 in SoftAP mode and will act as a web server for all the connecting devices. It will then turn On/Off 4 LEDs as per input from the connected station devices.

\*/

#include <ESP8266WiFi.h>

#include <ESP8266WebServer.h>

#include <EEPROM.h>

/\*Specifying the SSID and Password of the AP\*/

const char\* ap\_ssid = "WiFiWIN"; //Access Point SSID

const char\* ap\_password= "password"; //Access Point Password

uint8\_t max\_connections=1;//Maximum Connection Limit for AP

int current\_stations=0, new\_stations=0;

String WiFi\_Name = "";

String WiFi\_Password = "";

char value;

//Specifying the Webserver instance to connect with HTTP Port: 80

ESP8266WebServer server(80);

int addr = 0;

int length\_Wifiname\_EEPROM = 20;

//Specifying the Pins connected to LED

//uint8\_t led\_pin=LED\_BUILTIN;

uint8\_t wifi\_pin=LED\_BUILTIN;

struct {

    uint puissancemoy = 0; // we store the average power (but we could store all the previous power)

    uint countermoy = 0;  // counter of tnhe number of times we went to sleep without being able to send data

    String wifinameEEPROM="";

  }

  dataEEPROM;

//Specifying the boolean variables indicating the status of LED

//bool led\_status=false;

bool WiFi\_status=false;

void setup(){

  //Start the serial communication channel

  Serial.begin(115200);

  Serial.println();

  EEPROM.begin(512);

  //Output mode for the LED Pins

  //pinMode(led\_pin,OUTPUT);

  pinMode(wifi\_pin,OUTPUT);

  //Setting the AP Mode with SSID, Password, and Max Connection Limit

  if(WiFi.softAP(ap\_ssid,ap\_password,1,false,max\_connections)==true)

  {

    Serial.print("Access Point is Created with SSID: ");

    Serial.println(ap\_ssid);

    Serial.print("Max Connections Allowed: ");

    Serial.println(max\_connections);

    Serial.print("Access Point IP: ");

    Serial.println(WiFi.softAPIP());

  }

  else

  {

    Serial.println("Unable to Create Access Point");

  }

  //Specifying the functions which will be executed upon corresponding GET request from the client

  server.on("/",handle\_OnConnect);

  //server.on("/ledon",handle\_ledon);

  //server.on("/ledoff",handle\_ledoff);

  server.on("/WiFion",handle\_WiFion);

  server.on("/WiFioff",handle\_WiFioff);

  server.onNotFound(handle\_NotFound);

  server.on("/submitted",handle\_submitted);

  delay(5000);

//to be removed !!!

  Serial.println("Let's read EEPROM");

    for (int i=0;i<20;i++)

    {

      value = EEPROM.read(i);

      Serial.print(i);

      Serial.print("\t");

      Serial.print(value);

      Serial.println();

    }

//end of to be removed

// read bytes (i.e. sizeof(data) from "EEPROM"),

// in reality, reads from byte-array cache

// cast bytes into structure called data

  EEPROM.get(addr,dataEEPROM);

  WiFi\_Name = dataEEPROM.wifinameEEPROM; // we retrieve average power from eeprom

  Serial.print("Data stored for Wifi Name in EEPROM: ");

  Serial.println(WiFi\_Name);

  //Starting the Server

  server.begin();

  Serial.println("HTTP Server Started");

}

void loop() {

  //Assign the server to handle the clients

  server.handleClient();

  //Continuously check how many stations are connected to Soft AP and notify whenever a new station is connected or disconnected

  new\_stations=WiFi.softAPgetStationNum();

  if(current\_stations<new\_stations)//Device is Connected

  {

    current\_stations=new\_stations;

    Serial.print("New Device Connected to SoftAP... Total Connections: ");

    Serial.println(current\_stations);

  }

  if(current\_stations>new\_stations)//Device is Disconnected

  {

    current\_stations=new\_stations;

    Serial.print("Device disconnected from SoftAP... Total Connections: ");

    Serial.println(current\_stations);

  }

  //Turn the LED ON/OFF as per its status set by the connected client

  /\* //LED

  if(led\_status==false)

  {

    digitalWrite(led\_pin,LOW);

  }

  else

  {

    digitalWrite(led\_pin,HIGH);

  } \*/

  if(WiFi\_status==false)

  {

    digitalWrite(wifi\_pin,LOW);

  }

  else

  {

    digitalWrite(wifi\_pin,HIGH);

  }

}

void handle\_OnConnect()

{

  Serial.println("Client Connected");

  server.send(200, "text/html", HTML());

  if (server.method() == HTTP\_POST)

  {

    WiFi\_Name = server.arg("wifiname");

    WiFi\_Password = server.arg ("pwd");

    server.send(200, "text/html", "<!doctype html lang='en'<head><meta charset='utf-8'><meta name='viewpoint' content='width=device-width, initial-scale=1'");

  }

  else

  {

    server.send(200, "text/html", "<!doctype html lang='en'<head><meta charset='utf-8'><meta name='viewpoint' content='width=device-width, initial-scale=1'");

  }

}

//void handle\_ledon()

void handle\_WiFion()

{

  /\*Serial.println("LED ON");

  led\_status=true;

  server.send(200, "text/html", HTML());\*/

  Serial.println("WiFi ON");

  WiFi\_status=true;

  server.send(200, "text/html", HTML());

  for (int i =0; i<length\_Wifiname\_EEPROM; i++)

  {

    Serial.print(EEPROM.read(i));

  }

}

void handle\_submitted()

{

  Serial.println("received password");

  String message = "";

  if (server.arg("wifiname")== "") //Parameter not found

  {

    message = "wifi name not found";

  }

  else  //Parameter found

  {

    message = "WiFi Name = ";

    message += server.arg("wifiname");     //Gets the value of the query parameter

    WiFi\_Name = server.arg("wifiname");

  }

  Serial.println("WiFi Name received: ");

  Serial.println(message);

  Serial.println("storing in EEPROM");

  dataEEPROM.wifinameEEPROM =   WiFi\_Name ;

  EEPROM.put(addr,dataEEPROM);

  EEPROM.commit();

  Serial.println("Data stored in EEPROM. Reading first 20 characters:");

  for (int i=0;i<20;i++)

  {

    value = EEPROM.read(i);

    Serial.print(i);

    Serial.print("\t");

    Serial.print(value);

    Serial.println();

  }

  server.send(200, "text/html", HTMLsubmitted());

}

/\*void handle\_ledoff()

{

  Serial.println("LED OFF");

  led\_status=false;

  server.send(200, "text/html", HTML());\*/

void handle\_WiFioff()

{

  Serial.println("WiFi OFF");

  WiFi\_status=false;

  server.send(200, "text/html", HTML());

  for (int i =0; i<length\_Wifiname\_EEPROM; i++)

  {

    Serial.print(EEPROM.read(i));

  }

}

void handle\_NotFound()

{

  server.send(404, "text/plain", "Not found");

}

String HTML()

{

  String msg="<!DOCTYPE html> <html>\n";

    msg+=" <head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0, user-scalable=no\">\n";

    msg+=" <title>WiFi Control</title>\n";

    msg+=" <style>html{font-family:Helvetica; display:inline-block; margin:0px auto; text-align:center;}\n";

    msg+=" body{margin-top: 50px;} h1{color: #444444; margin: 50px auto 30px;} h3{color:#444444; margin-bottom: 50px;}\n";

    msg+=" .button{display:block; width:80px; background-color:#f48100; border:none; color:white; padding: 13px 30px; text-decoration:none; font-size:25px; margin: 0px auto 35px; cursor:pointer; border-radius:4px;}\n";

    msg+=" .button-on{background-color:#f48100;}\n";

    msg+=" .button-on:active{background-color:#f48100;}\n";

    msg+=" .button-off{background-color:#26282d;}\n";

    msg+=" .button-off:active{background-color:#26282d;}\n";

    msg+=" </style>\n";

    msg+=" </head>\n";

    msg+=" <body>\n";

    msg+=" <h1>ESP8266 Web Server</h1>\n";

    msg+=" <h3>Using Access Point (AP) Mode</h3>\n";

    msg+="<h2>Enter your WiFi network and credentials</h2>\n";

    msg+="<form action=\"/submitted\">\n";

    msg+=" <label for=\"wifiname\">WiFi Name:</label><br>\n";

    msg+=" <input type=\"text\" id=\"wifiname\" name=\"wifiname\" value=\"\"><br>\n";

    msg+=" <label for=\"pwd\">WiFi Password:</label><br>\n";

    msg+=" <input type=\"text\" id=\"pwd\" name=\"pwd\" value=\"\"><br><br>\n";

    msg+=" <input type=\"submit\" value=\"Submit\"> </form>\n";

    msg+=" <p>If you click the \"Submit\" button, the Linky dongle will try to connect to your wifi until you reset it through the dedicated button (see procedure in the datasheet).</p>\n";

/\*

  String msg="<!DOCTYPE html> <html>\n";

    msg+="<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0, user-scalable=no\">\n";

    msg+=" <title>LED Control</title>\n";

    msg+=" <style>html{font-family:Helvetica; display:inline-block; margin:0px auto; text-align:center;}\n";

    msg+=" body{margin-top: 50px;} h1{color: #444444; margin: 50px auto 30px;} h3{color:#444444; margin-bottom: 50px;}\n";

    msg+=" .button{display:block; width:80px; background-color:#f48100; border:none; color:white; padding: 13px 30px; text-decoration:none; font-size:25px; margin: 0px auto 35px; cursor:pointer; border-radius:4px;}\n";

    msg+=" .button-on{background-color:#f48100;}\n";

    msg+=" .button-on:active{background-color:#f48100;}\n";

    msg+=" .button-off{background-color:#26282d;}\n";

    msg+=" .button-off:active{background-color:#26282d;}\n";

    msg+=" </style>\n";

    msg+=" </head>\n";

    msg+=" <body>\n";

    msg+=" <h1>ESP8266 Web Server</h1>\n";

    msg+=" <h3>Using Access Point (AP) Mode</h3>\n";

    msg+="<h2>Enter your Wifi Network and credentials</h2>\n";

    msg+="<form action="/XXXXXX.html">\n";

    msg+=" <label for="fname">Wifi name:</label><br>\n";

    msg+=" <input type="text" id="fname" name="fname" value=""><br>\n";

    msg+=" <label for="lname">Wifi Password:</label><br>\n";

    msg+=" <input type="text" id="lname" name="lname" value=""><br><br>\n";

    msg+=" <input type="submit" value="Submit"> </form>\n";

    msg+=" <p>If you click the "Submit" button, the Linky dongle will try to connect to your wifi until you reset it through the dedicated button (see procedure in the datasheet).</p>\n";

    msg+="   </body>\n";

    msg+="</html>\n";

\*/

    /\*if(led\_status==false)

    {

      msg+="<p>LED Status: OFF</p><a class=\"button button-on\" href=\"/ledon\">ON</a>\n";

    }

    else

    {

      msg+="<p>LED Status: ON</p><a class=\"button button-off\" href=\"/ledoff\">OFF</a>\n";

    }\*/

    if(WiFi\_status==false)

    {

      msg+="<p>WiFi Status: OFF</p><a class=\"button button-on\" href=\"/WiFion\">ON</a>\n";

    }

    else

    {

      msg+="<p>WiFi Status: ON</p><a class=\"button button-off\" href=\"/WiFioff\">OFF</a>\n";

    }

    msg+="</body>\n";

    msg+="</html>\n";

    return msg;

}

String HTMLsubmitted()

{

  String msg="<!DOCTYPE html> <html>\n";

    msg+=" <head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0, user-scalable=no\">\n";

    //msg+=" <title>LED Control</title>\n";

    msg+=" <title>WiFi Control</title>\n";

    msg+=" <style>html{font-family:Helvetica; display:inline-block; margin:0px auto; text-align:center;}\n";

    msg+=" body{margin-top: 50px;} h1{color: #444444; margin: 50px auto 30px;} h3{color:#444444; margin-bottom: 50px;}\n";

    msg+=" .button{display:block; width:80px; background-color:#f48100; border:none; color:white; padding: 13px 30px; text-decoration:none; font-size:25px; margin: 0px auto 35px; cursor:pointer; border-radius:4px;}\n";

    msg+=" .button-on{background-color:#f48100;}\n";

    msg+=" .button-on:active{background-color:#f48100;}\n";

    msg+=" .button-off{background-color:#26282d;}\n";

    msg+=" .button-off:active{background-color:#26282d;}\n";

    msg+=" </style>\n";

    msg+=" </head>\n";

    msg+=" <body>\n";

    msg+=" <h1>ESP8266 Web Server</h1>\n";

    msg+=" <h3>Thank you for the information. We are trying to connect. Check the light status on the Linky dongle, or go back to the wifi information page</h3>\n";

    /\*

    if(led\_status==false)

    {

      msg+="<p>LED Status: OFF</p><a class=\"button button-on\" href=\"/ledon\">ON</a>\n";

    }

    else

    {

      msg+="<p>LED Status: ON</p><a class=\"button button-off\" href=\"/ledoff\">OFF</a>\n";

    }

    \*/

     if(WiFi\_status==false)

    {

      msg+="<p>WiFi Status: OFF</p><a class=\"button button-on\" href=\"/WiFion\">ON</a>\n";

    }

    else

    {

      msg+="<p>WiFi Status: ON</p><a class=\"button button-off\" href=\"/WiFioff\">OFF</a>\n";

    }

    msg+="</body>\n";

    msg+="</html>\n";

    return msg;

}

1. **Annexe : Code WEMO**

// #################################  compile it for LOLIN (WeMO) D1 R1 ##############################

// Install CH34x\_Install\_Windows\_v3\_4

/\* this code comes from https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WiFi/examples/HTTPSRequest

 \*  Interesting also: https://maakbaas.com/esp8266-iot-framework/logs/https-requests/

 \*  if error uploading the code: https://sparks.gogo.co.nz/ch340.html  download and install the driver.

\*/

#include <NTPClient.h> // to get a clock from a synchronisation server

#include <WiFiUdp.h> // needed for the ntp client

//https://lastminuteengineers.com/esp8266-ntp-server-date-time-tutorial/

#include <TimeLib.h> // to convert epoch into days. requires to install library:

const long utcOffsetInSeconds = 3600;

String currentDate = "";

#include <ctype.h>

#include <ESP8266WiFi.h>

#include <WiFiClientSecure.h>

#include <ESP8266HTTPClient.h>

#include <WiFiClient.h>

#include "certs.h"

#ifndef STASSID

#define STASSID "WIFI\_SSID\_LiveboxXXXX"

#define STAPSK "WIFI\_PASSWORD"

// #define STASSID "livinglab"

// #define STAPSK "2021#Bowl@LivingLab"

#endif

#include <EEPROM.h>

//const long utcOffsetInSeconds = 3600;

//##############   variables definition   ###########################

int counterwakeup = 0; // counter to check how many times we wen to wake up before to send requests

String StringToDetect = "SINSTS";

char detectSINSTS[7]; // array that records the characters received

char detectPAPP[5]; // array that records the characters received

bool state = HIGH; // state for the LED

bool start\_record = false;  // boolean triggered to start recording the power measurement

char puissance[5]; // power measured, in char

int puissance\_int = 0; // power measured, in int

uint puissance\_Uint = 0; // power measured, in int

uint puissance\_moyenne = 0; // averagepower measured, in int

uint firststartup = 0;

int indice = 0;  // index for the recording of the power

int debug = 0;  // true when we are in debug mode

uint counter\_moyenne = 0; // counter to compute the average

uint counter\_moyennetemp = 0; // used to store temporary counter to compute the average

int puissanceCommuniquee = 0; // puissance que l'on communique via wifi

bool finished\_recording = false; //specifies if we can go back to sleep (= we received the signal from Linky)

long int timeoutReadLinky = 0; // timeout to stop trying to read linky if there is no data to read

//const int A0 = A0;  // Analog input pin that the potentiometer is attached to

int BatteryVoltage = 0;

int Danger = 0; // indicate if Power > Powermax

int Powermax = 10000; // to be read from Linky, and stored in EEPROM

long int t1 = 0;

long int t2=0;

uint voltageThreshold = 190; //207; // voltage limit to startup the communication

uint voltageThreshold2 = 170; // voltage limit to startup the process

const char\* TIP\_host = "tip-imredd.unice.fr";

const char\* ssid = STASSID;

const char\* password = STAPSK;

X509List cert(cert\_DigiCert\_Global\_Root\_CA);

   // ################ set up wifi + send message function, is triggered only

   void setup\_wifi\_TIP() {

      //WiFi.mode(WIFI\_STA);

    // WiFi.config(ip, gateway\_dns, gateway\_dns);

      WiFi.begin(ssid, password);

  while (WiFi.status() != WL\_CONNECTED) {

    delay(50);

  }

           if (debug==1){

 Serial.println("connected");

 Serial.println("IP address: "); //

Serial.println(WiFi.localIP());//

           }

  configTime(3 \* 3600, 0,"134.59.1.5", "pool.ntp.org", "time.nist.gov");

  time\_t now = time(nullptr);

  while (now < 8 \* 3600 \* 2) {

    delay(500);

    now = time(nullptr);

  }

  struct tm timeinfo;

  gmtime\_r(&now, &timeinfo);

 int epoch\_time = now;

 /\*

// Define NTP Client to get time

WiFiUDP ntpUDP;

NTPClient timeClient(ntpUDP, "pool.ntp.org", utcOffsetInSeconds);

timeClient.update();

int hourvalue = timeClient.getHours();

int minutevalue = timeClient.getMinutes();

int secondvalue = timeClient.getSeconds();\*/

//int epoch\_time = now;

// define starting date

String monthstringstart = "";

String yearstringstart = "";

String daystringstart = "";

String hourstringstart = "";

String minutestringstart = "";

String secondstringstart = "";

String monthstringend = "";

String yearstringend = "";

String daystringend = "";

String hourstringend = "";

String minutestringend = "";

String secondstringend = "";

if (month(epoch\_time)>9) {

   monthstringstart = String(month(epoch\_time));

}else {

   monthstringstart = "0"+String(month(epoch\_time));

}

if (day(epoch\_time)>9) {

   daystringstart = String(day(epoch\_time));

}else {

    daystringstart = "0"+String(day(epoch\_time));

}

if (hour(epoch\_time)>9) {

   hourstringstart = String(hour(epoch\_time));

}else {

    hourstringstart = "0"+String(hour(epoch\_time));

}

if (minute(epoch\_time)>9) {

   minutestringstart = String(minute(epoch\_time));

}else {

   minutestringstart = "0"+String(minute(epoch\_time));

}

if (second(epoch\_time)>9) {

   secondstringstart = String(second(epoch\_time));

}else {

   secondstringstart = "0"+String(second(epoch\_time));

}

 yearstringstart = String(year(epoch\_time));

int counter\_moyenne\_int = counter\_moyenne;

// define end date:

epoch\_time=epoch\_time + 1;//15\*max(1,counter\_moyenne\_int); // IF WE HAVE SLEEP EVERY 10-15 SECONDS

if (month(epoch\_time)>9) {

   monthstringend = String(month(epoch\_time));

}else {

   monthstringend = "0"+String(month(epoch\_time));

}

if (day(epoch\_time)>9) {

   daystringend = String(day(epoch\_time));

}else {

    daystringend = "0"+String(day(epoch\_time));

}

if (hour(epoch\_time)>9) {

   hourstringend = String(hour(epoch\_time));

}else {

    hourstringend = "0"+String(hour(epoch\_time));

}

if (minute(epoch\_time)>9) {

   minutestringend = String(minute(epoch\_time));

}else {

   minutestringend = "0"+String(minute(epoch\_time));

}

if (second(epoch\_time)>9) {

   secondstringend = String(second(epoch\_time));

}else {

   secondstringend = "0"+String(second(epoch\_time));

}

 yearstringend = String(year(epoch\_time));

  WiFiClientSecure client;

  //client.setTrustAnchors(&cert);

  client.setInsecure(); //the magic line, use with caution

  if (!client.connect(TIP\_host, github\_port)) {

    //Connection failed

    return;

  }

  String url = "/nodes/imredd/energyconso/linky";

String httpRequestData = "linkySensor,sensor\_id=LinkyHouse2 power="+String(puissanceCommuniquee)+",danger="+String(Danger)+",voltage=234.231,frequency=50.2,pf=0.9 "+String(epoch\_time)+"000000000";

 client.print(String("POST ") + url + " HTTP/1.0\r\n" + "Host: " + TIP\_host + "\r\n" + "User-Agent: BuildFailureDetectorESP8266\r\n" +  "Content-Length: " + httpRequestData.length()   + "\r\n"+"Connection: close\r\n\r\n"+httpRequestData);

}

   // when voltage is high enough#####################

 /\*   void setup\_wifiChargeAngels() {

      //WiFi.mode(WIFI\_STA);

     WiFi.config(ip, gateway\_dns, gateway\_dns);

      WiFi.begin(ssid, password);

  while (WiFi.status() != WL\_CONNECTED) {

    delay(50);

  }

  configTime(3 \* 3600, 0,"134.59.1.5", "pool.ntp.org", "time.nist.gov");

  time\_t now = time(nullptr);

  while (now < 8 \* 3600 \* 2) {

    delay(500);

    now = time(nullptr);

  }

  struct tm timeinfo;

  gmtime\_r(&now, &timeinfo);

 int epoch\_time = now;

//int epoch\_time = now;

// define starting date

String monthstringstart = "";

String yearstringstart = "";

String daystringstart = "";

String hourstringstart = "";

String minutestringstart = "";

String secondstringstart = "";

String monthstringend = "";

String yearstringend = "";

String daystringend = "";

String hourstringend = "";

String minutestringend = "";

String secondstringend = "";

if (month(epoch\_time)>9) {

   monthstringstart = String(month(epoch\_time));

}else {

   monthstringstart = "0"+String(month(epoch\_time));

}

if (day(epoch\_time)>9) {

   daystringstart = String(day(epoch\_time));

}else {

    daystringstart = "0"+String(day(epoch\_time));

}

if (hour(epoch\_time)>9) {

   hourstringstart = String(hour(epoch\_time));

}else {

    hourstringstart = "0"+String(hour(epoch\_time));

}

if (minute(epoch\_time)>9) {

   minutestringstart = String(minute(epoch\_time));

}else {

   minutestringstart = "0"+String(minute(epoch\_time));

}

if (second(epoch\_time)>9) {

   secondstringstart = String(second(epoch\_time));

}else {

   secondstringstart = "0"+String(second(epoch\_time));

}

 yearstringstart = String(year(epoch\_time));

int counter\_moyenne\_int = counter\_moyenne;

// define end date:

epoch\_time=epoch\_time + 1;//15\*max(1,counter\_moyenne\_int); // IF WE HAVE SLEEP EVERY 10-15 SECONDS

if (month(epoch\_time)>9) {

   monthstringend = String(month(epoch\_time));

}else {

   monthstringend = "0"+String(month(epoch\_time));

}

if (day(epoch\_time)>9) {

   daystringend = String(day(epoch\_time));

}else {

    daystringend = "0"+String(day(epoch\_time));

}

if (hour(epoch\_time)>9) {

   hourstringend = String(hour(epoch\_time));

}else {

    hourstringend = "0"+String(hour(epoch\_time));

}

if (minute(epoch\_time)>9) {

   minutestringend = String(minute(epoch\_time));

}else {

   minutestringend = "0"+String(minute(epoch\_time));

}

if (second(epoch\_time)>9) {

   secondstringend = String(second(epoch\_time));

}else {

   secondstringend = "0"+String(second(epoch\_time));

}

 yearstringend = String(year(epoch\_time));

  WiFiClientSecure client;

  client.setTrustAnchors(&cert);

//String currentDate = String(currentYear) + "-" + String(currentMonth) + "-" + String(monthDay);

 // Serial.println("Date: "+ String(year(epoch\_time))+"-"+ String(month(epoch\_time))+"-"+String(day(epoch\_time))+" "+ String(hour(epoch\_time))+"-"+ String(minute(epoch\_time))+"-"+ String(second(epoch\_time)));

  //Serial.println(currentDate);

  // HTTPClient http;

   //String serverPath = serverName;   // + "?temperature=24.37";

 String httpRequestData = "{\"assetID\": \"6365122519b9aa99b068c09a\",\"startedAt\": \""+yearstringstart+"-"+monthstringstart+"-"+daystringstart+"T"+hourstringstart+":"+minutestringstart+":"+secondstringstart+".000Z\",\"endedAt\": \""+yearstringend+"-"+monthstringend+"-"+daystringend+"T"+hourstringend+":"+minutestringend+":"+secondstringend+".000Z\",\"instantWatts\": "+String(puissanceCommuniquee)+",\"instantWattsL1\": 0,\"instantWattsL2\": 0,\"instantWattsL3\": 0,\"instantAmps\": 0,\"instantAmpsL1\": 0,\"instantAmpsL2\": 0,\"instantAmpsL3\": 0,\"instantVolts\": 0,\"instantVoltsL1\": 0,\"instantVoltsL2\": 0,\"instantVoltsL3\": 0,\"consumptionWh\": 0,\"consumptionAmps\": 0,\"stateOfCharge\": 0}";

  if (!client.connect(github\_host, github\_port)) {

    //Connection failed

    return;

  }

  String url = "/v1/api/assets/6365122519b9aa99b068c09a/consumptions"; //"/a/check";

// https://wokwi.com/projects/327948646817464914

//https://forum.arduino.cc/t/esp32-https-post-request/964599/2

client.stop();

    if (client.connect(github\_host, 443)) {

      client.println("POST " + url + " HTTP/1.0");

      client.println("Host: " + (String)github\_host);

      client.println(F("User-Agent: ESP"));

      client.println(F("Connection: close"));

      client.println(F("Content-Type: application/json"));

      client.println(F("Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..1C9H2giT6xGbDNAVgeuGPuHphHZD0VbkPrDdA\_uNg-8"));

      client.print(F("Content-Length: "));

      client.println(httpRequestData.length());

      client.println();

      client.println(httpRequestData);

    }

     // String serverPath = serverName+String(BatteryVoltage)+"&power="+String(puissanceCommuniquee)+"&counter="+String(counter\_moyenne)+"&danger="+String(Danger)+"&wakeupnumber="+counterwakeup;// + "?temperature=24.37";

      // defines the server name with URL path

  //  http.begin(client, serverPath.c\_str());

   //   http.addHeader("Content-Type", "application/json");

   //   http.addHeader("Authorization", "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..-3S1Fi0ElxpIIZlNmAxBsevFQwBOzaZ9rHAE4fMi1oQ");

      // Send HTTP GET request

  //    int httpResponseCode = http.GET();

   //   int httpResponseCode = http.POST(httpRequestData);

      // Send HTTP GET request

     // int httpResponseCode = http.GET(); //send GET request to the server

//if (httpResponseCode>0) {

//   digitalWrite(0, LOW); // initialisation

//delay(50);

 // digitalWrite(0, HIGH); // initialisation

//delay(50);

// }

}\*/

   // ################ listen to Linky, every 15 seconds #####################

   // ################ listen to Linky, every 15 seconds #####################

void readLinky9600(){

 // Serial.println("reading TIC!");

//digitalWrite(0, LOW); // initialisation

//delay(50);

// digitalWrite(0, HIGH); // initialisation

//delay(50);

  int communication = Serial.available();

  if (communication != 0)  // when we receive a character

  {

    char caracter = Serial.read();

    // we slide the array of detect\_PAPP to keep collecting characters

    detectSINSTS[0]=detectSINSTS[1];

    detectSINSTS[1]=detectSINSTS[2];

    detectSINSTS[2]=detectSINSTS[3];

    detectSINSTS[3]=detectSINSTS[4];

    detectSINSTS[4]=detectSINSTS[5];

    detectSINSTS[5]=detectSINSTS[6];

    detectSINSTS[6]=caracter;

   /\* detectPAPP[0]=detectPAPP[1];

    detectPAPP[1]=detectPAPP[2];

    detectPAPP[2]=detectPAPP[3];

    detectPAPP[3]=detectPAPP[4];

    detectPAPP[4]=caracter;\*/

    if (start\_record == true ) // if we have received "PAPP " or "SINST ", we start recording the power

    {

     if (indice>=5 )//&& (caracter == ' ' || isdigit(caracter)==false)) // if we receive a space, the power information is over. could be more robust by considering anything above 9

      {

        indice = 0;

                 if (debug==1){

        Serial.print("got a power:");

        Serial.println(puissance);

        }

        start\_record = false;

        puissance\_int = String(puissance).toInt();

        puissance\_Uint = (unsigned int)puissance\_int;

        puissance\_moyenne = puissance\_moyenne + puissance\_Uint;

                         if (debug==1){

         Serial.print("Puissance Uint:");

        Serial.println(puissance\_Uint);

        Serial.print("Puissance moyenne:");

        Serial.println(puissance\_moyenne);

        }

        finished\_recording=true;

        if (puissance\_int > Powermax)  // if the power is above 6kW

        {

        puissanceCommuniquee = puissance\_int;

        counter\_moyennetemp = counter\_moyenne;

        counter\_moyenne = 1;

        Danger = 1;

        setup\_wifi\_TIP();

        Danger = 0;

        counter\_moyenne = counter\_moyennetemp;

        }

       /\* if (puissance\_int > Powermax\_test\_relai)  // if the power is above 6kW

        {

          Need\_to\_cut = 1;

        } \*/

        }

      else {  // if we have received a digit, we store it

         if (isdigit(caracter))

         {puissance[indice] = caracter;}

         else puissance[indice] = '0';   //we should discard the number  instead of putting a 0

          indice = indice+1;

        }

    }

    else

    {

    if (detectSINSTS[0]=='S' && detectSINSTS[1]=='I' && detectSINSTS[2]=='N' && detectSINSTS[3]=='S' &&detectSINSTS[4]=='T' && detectSINSTS[5]=='S' )// && detectSINSTS[6]==' ')

    // if (detectPAPP[0]=='P' && detectPAPP[1]=='A' && detectPAPP[2]=='P' && detectPAPP[3]=='P' && detectPAPP[4]==' ' )

    {

       if (debug==1){

        Serial.print("We are here");

       }

       start\_record = true;

          counter\_moyenne = counter\_moyenne+1;

          for (int i = 0; i < sizeof(puissance)/sizeof(puissance[0]); i++){

          puissance[i] = 0;}

    }

    }

  }

}

   // ################ listen to Linky, every 15 seconds #####################

void readLinky1200(){

 // Serial.println("reading TIC!");

//digitalWrite(0, LOW); // initialisation

//delay(50);

// digitalWrite(0, HIGH); // initialisation

//delay(50);

  int communication = Serial.available();

  if (communication != 0)  // when we receive a character

  {

    char caracter = Serial.read();

    // we slide the array of detect\_PAPP to keep collecting characters

  /\*  detectSINSTS[0]=detectSINSTS[1];

    detectSINSTS[1]=detectSINSTS[2];

    detectSINSTS[2]=detectSINSTS[3];

    detectSINSTS[3]=detectSINSTS[4];

    detectSINSTS[4]=detectSINSTS[5];

    detectSINSTS[5]=detectSINSTS[6];

    detectSINSTS[6]=caracter; \*/

    detectPAPP[0]=detectPAPP[1];

    detectPAPP[1]=detectPAPP[2];

    detectPAPP[2]=detectPAPP[3];

    detectPAPP[3]=detectPAPP[4];

    detectPAPP[4]=caracter;

    if (start\_record == true ) // if we have received "PAPP " or "SINST ", we start recording the power

    {

     if (indice>=5 )//&& (caracter == ' ' || isdigit(caracter)==false)) // if we receive a space, the power information is over. could be more robust by considering anything above 9

      {

        indice = 0;

                 if (debug==1){

        Serial.print("got a power:");

        Serial.println(puissance);}

        start\_record = false;

        puissance\_int = String(puissance).toInt();

        puissance\_Uint = (unsigned int)puissance\_int;

        puissance\_moyenne = puissance\_moyenne + puissance\_Uint;

        finished\_recording=true;

        if (puissance\_int > Powermax)  // if the power is above 6kW

        {

        puissanceCommuniquee = puissance\_int;

        counter\_moyennetemp = counter\_moyenne;

        counter\_moyenne = 1;

        Danger = 1;

        setup\_wifi\_TIP();

        Danger = 0;

        counter\_moyenne = counter\_moyennetemp;

        }

       /\* if (puissance\_int > Powermax\_test\_relai)  // if the power is above 6kW

        {

          Need\_to\_cut = 1;

        } \*/

        }

      else {  // if we have received a digit, we store it

         if (isdigit(caracter))

         {puissance[indice] = caracter;}

         else puissance[indice] = '0';   //we should discard the number  instead of putting a 0

          indice = indice+1;

        }

    }

    else

    {

 //   if (detectSINSTS[0]=='S' && detectSINSTS[1]=='I' && detectSINSTS[2]=='N' && detectSINSTS[3]=='S' &&detectSINSTS[4]=='T' && detectSINSTS[5]=='S')

     if (detectPAPP[0]=='P' && detectPAPP[1]=='A' && detectPAPP[2]=='P' && detectPAPP[3]=='P' && detectPAPP[4]==' ' )

    {

          start\_record = true;

          counter\_moyenne = counter\_moyenne+1;

          for (int i = 0; i < sizeof(puissance)/sizeof(puissance[0]); i++){

          puissance[i] = 0;}

    }

    }

  }

}

 void setup() {

  pinMode(0, OUTPUT);

  digitalWrite(0, HIGH); // initialisation

  delay(10);

  digitalWrite(0, LOW);

    delay(100);

   digitalWrite(0, HIGH); // initialisation

  pinMode(5, OUTPUT);

  digitalWrite(5, LOW);

        Serial.begin(1200,SERIAL\_7E1);    // Define and start serial monitoring of the Linky TIC

//Serial.swap();

       BatteryVoltage = analogRead(A0); // we read voltage

                       if (debug==1){

BatteryVoltage = 255;}

if (BatteryVoltage>voltageThreshold2)

//if (BatteryVoltage>0)

   {

     uint addr = 0; //address of eeprom memory

    // we use EEPROM in order to keep the values stored even when the module is sleeping

  struct {

    uint puissancemoy = 0; // we store the average power (but we could store all the previous power)

    uint countermoy = 0;  // counter of tnhe number of times we went to sleep without being able to send data

  //  uint booleenfirststartup = 0;//char str[20] = "";

  } dataEEPROM;

    // commit 512 bytes of ESP8266 flash (for "EEPROM" emulation)

  // this step actually loads the content (512 bytes) of flash into

  // a 512-byte-array cache in RAM

  EEPROM.begin(512);

  // read bytes (i.e. sizeof(data) from "EEPROM"),

  // in reality, reads from byte-array cache

  // cast bytes into structure called data

  EEPROM.get(addr,dataEEPROM);

  puissance\_moyenne = dataEEPROM.puissancemoy; // we retrieve average power from eeprom

  counter\_moyenne = dataEEPROM.countermoy;

 // firststartup = dataEEPROM.booleenfirststartup;

        pinMode(A0, INPUT); // we define the analog pin (ADC PIN below RST)

      counterwakeup = counterwakeup+1; // we count the number of times the dongle woke up and stored data in the power average

      t1 = millis(); // we start to record time to use the timeout to go to sleep if we do not read any data from Linky

       BatteryVoltage = analogRead(A0); // we read voltage

                if (debug==1){

Serial.println("Starting Reading loop");

BatteryVoltage = 255;}

      timeoutReadLinky = 0;

       while(finished\_recording == false && timeoutReadLinky<5000){

             readLinky1200(); // we read Linky data until we have found a data (finished\_recording) or if the timeout is reached

             t2 = millis();

             timeoutReadLinky = t2-t1;

        }

         if (debug==1){

Serial.println("End of Reading loop");}

  //       if(BatteryVoltage>voltageThreshold && finished\_recording) {  // if we have enough voltage, we send the data and reinitialise the values

         if(debug==1 || (BatteryVoltage>voltageThreshold && finished\_recording)) {  // if we have enough voltage, we send the data and reinitialise the values

         puissanceCommuniquee=puissance\_moyenne/counter\_moyenne;

         // puissanceCommuniquee=puissance\_Uint;

         if (debug==1){

          Serial.println("ready to send");

         }

            setup\_wifi\_TIP();

            counterwakeup = 0;

            counter\_moyenne = 0;

            puissance\_moyenne = 0;

        }

// now we store the updated values back in the eeprom before going to sleep

dataEEPROM.puissancemoy =   puissance\_moyenne ;

dataEEPROM.countermoy= counter\_moyenne ;

 //dataEEPROM.booleenfirststartup = 1;

// replace values in byte-array cache with modified data

  // no changes made to flash, all in local byte-array cache

  EEPROM.put(addr,dataEEPROM);

    // actually write the content of byte-array cache to

  // hardware flash.  flash write occurs if and only if one or more byte

  // in byte-array cache has been changed, but if so, ALL 512 bytes are

  // written to flash

  EEPROM.commit();

finished\_recording = false;

   }

 // ESP.deepSleep(20e6); // we go to sleep 10 seconds

  ESP.deepSleep(20e6); // we go to sleep 10 seconds

    }

    void loop() { // nothing here, we go to sleep and do not enter an infinite loop

    }