Franck Perret

ANF-IOT 12 Septembre 2023

CHEAP'EAU

SOLUTIONS INNOVANTES A BAS COUT POUR LE SUIVI DES SYSTEMES DE GESTION DES EAUX URBAINES

Durée du projet : 2020-2022

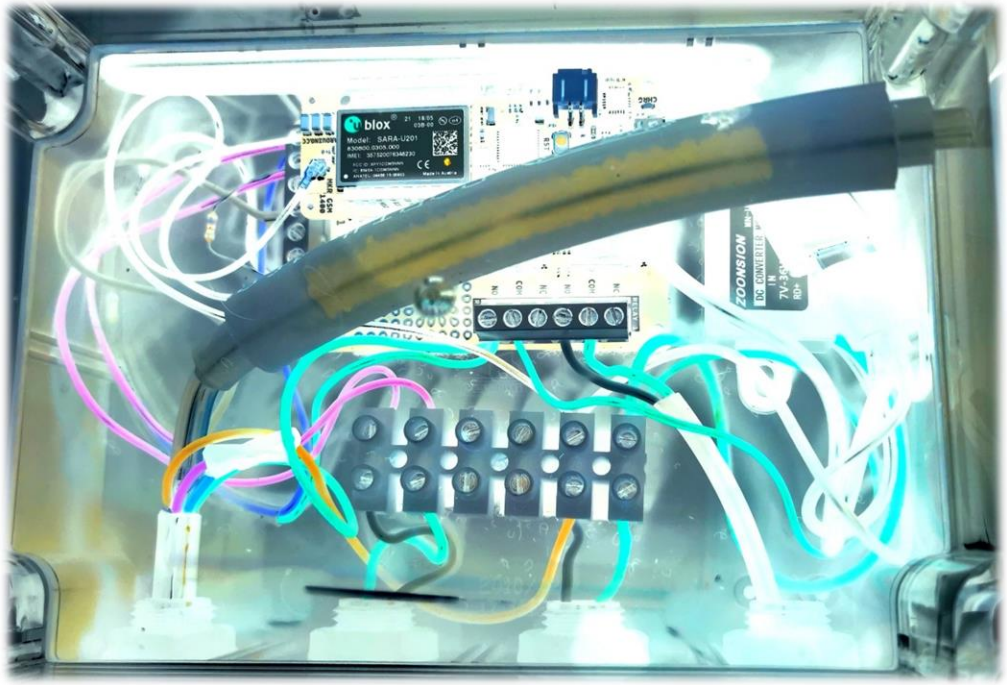
Sommaire

- Le projet
- Choix et développement technique
- Travaux Pratiques

Le projet

Projet Cheap’Eau : Objectifs à atteindre

- Abordable
- Robuste et discret
- Autonome
- Performances
- Connectés
- Simple à construire et à prendre en main



Projet Cheap'Eau : cahier des charges

- Utiliser des capteurs bon marché (< 10-15 € / unité)
- Fabriqués en série pour l'industrie, les transports ou la maison, et immédiatement disponibles dans le commerce
- Panneaux solaires/batterie
- Encapsulage simple
- Les adapter aux matériels informatiques et logiciels libres des communautés d'utilisateurs Arduino®
- Réseau IoT : Lora ou SigFox IoT
- Définir les besoins avec les gestionnaires, BE et citoyens (associations) et assurer le transfert de ces systèmes

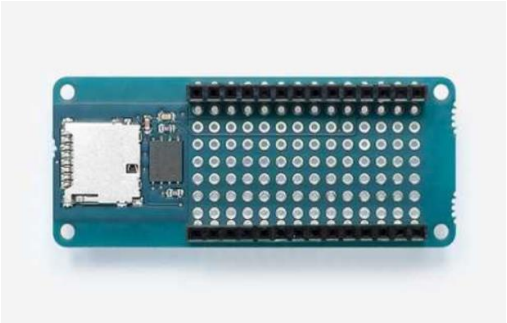
Choix et développement technique

Choix du type de carte Arduino Lora

TTN-UN-868

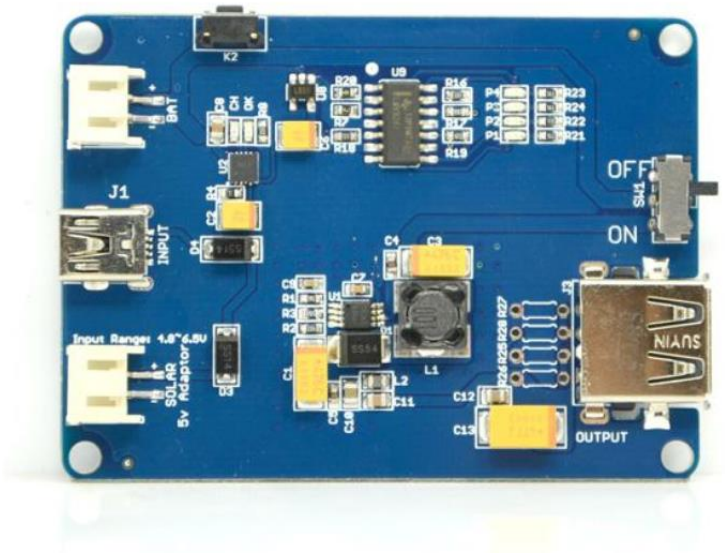
Leonardo
Microcontrôleur : Atmega 32U4
Mémoire flash 32ko
Horloge: 16 Mhz

Mkr WAN 1300 (connectivité Lora) Arduino



Microcontrôleur : Atmel SAM D21
Mémoire flash : 256 ko
Horloge : 48 Mhz

Alimentation électrique

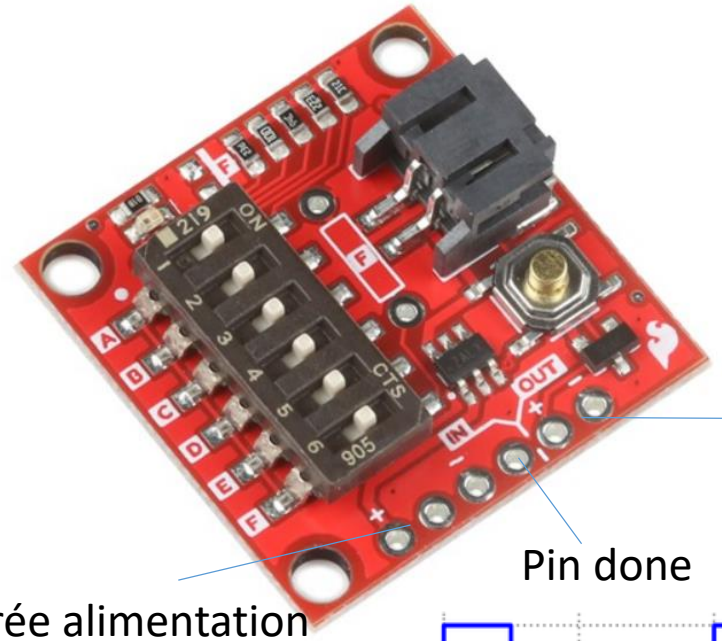


Carte LiPo Rider Pro 106990008

Cellule solaire SOL3W
5,5 V/540 mA - 160 x 138 mm

Batterie LiPo 3.8v 5100mAh

Nano Timer TPL5110



Sortie alimentation électrique

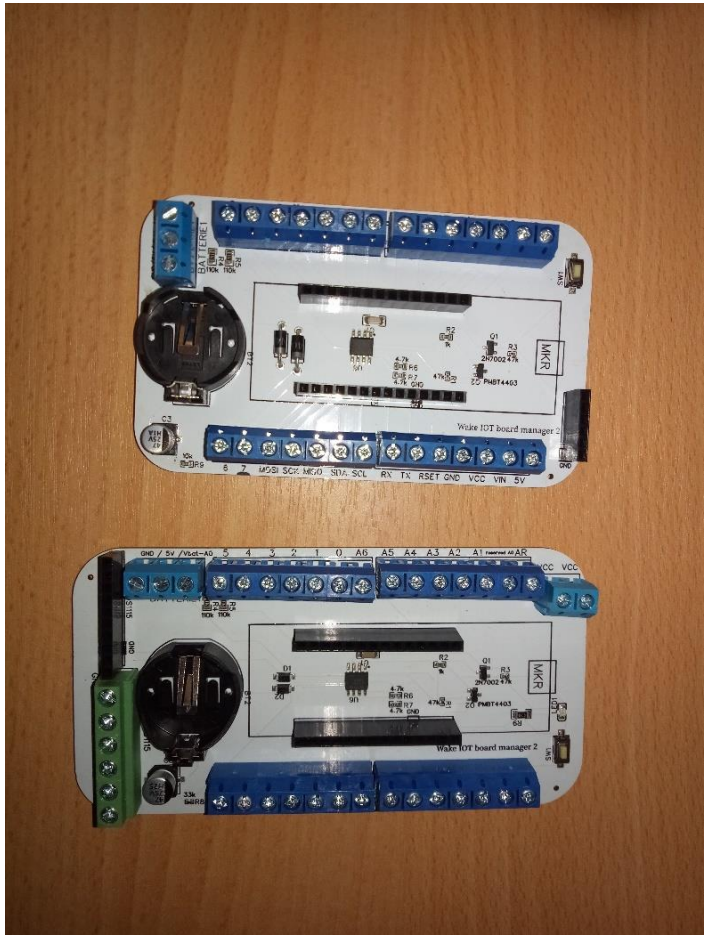
Pin done

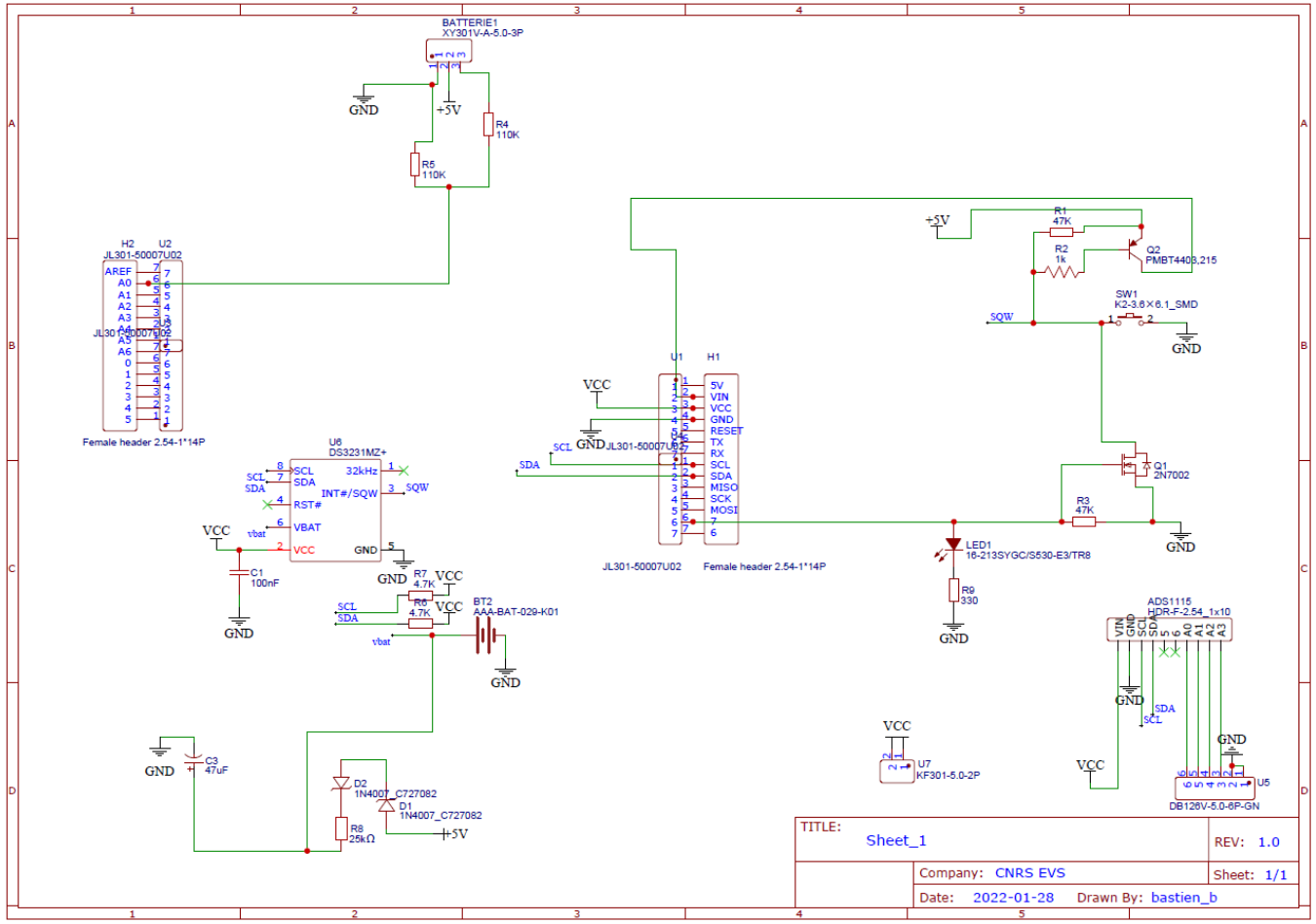
Entrée alimentation électrique

Timer	Switch Combo	Resistance
2-3 s	A+B+C+D+E	7.579 kΩ
3-4 s	A+B+C+D	7.933 kΩ
4 s	A+B+C	8.470 kΩ
5 s	A+B+E	8.844 kΩ
6 s	A+B	9.329 kΩ
10 s	A+C+D+E	11.563 kΩ
~12 s	A+C+D	12.407 kΩ
~13 s	A+C+E	12.742 kΩ
~15 s	A+D+E	13.225 kΩ
~18 s	A+C	13.774 kΩ
~19 s	B+C+D+E	14.243 kΩ
20 s	A+D	14.341 kΩ
~22 s	A+E	14.790 kΩ
~25 s	B+C+D	15.546 kΩ
~28 s	B+C+E	16.075 kΩ
32 s	B+D+E	16.852 kΩ
35 s	B+C	17.754 kΩ
40 s	B+D	18.707 kΩ
~45 s	B+E	19.479 kΩ
~5 min	C+D+E	40.400 kΩ
8 min	C+D	52.995 kΩ
~12 min	C+E	59.694 kΩ
15 min	D+E	72.033 kΩ

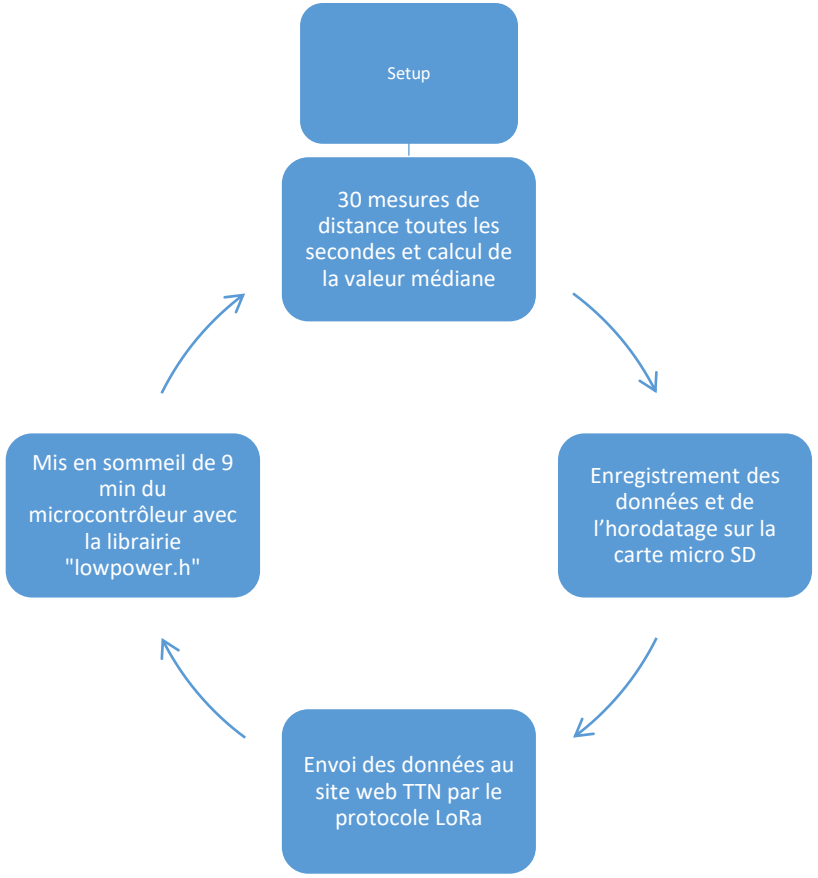
Temps réglable par commutation de switch (2 sec à 15 min)
Consommation électrique à l'arrêt : 35nA pour 2.5v
Précision temporelle : 1%

RTC Wake-up manager

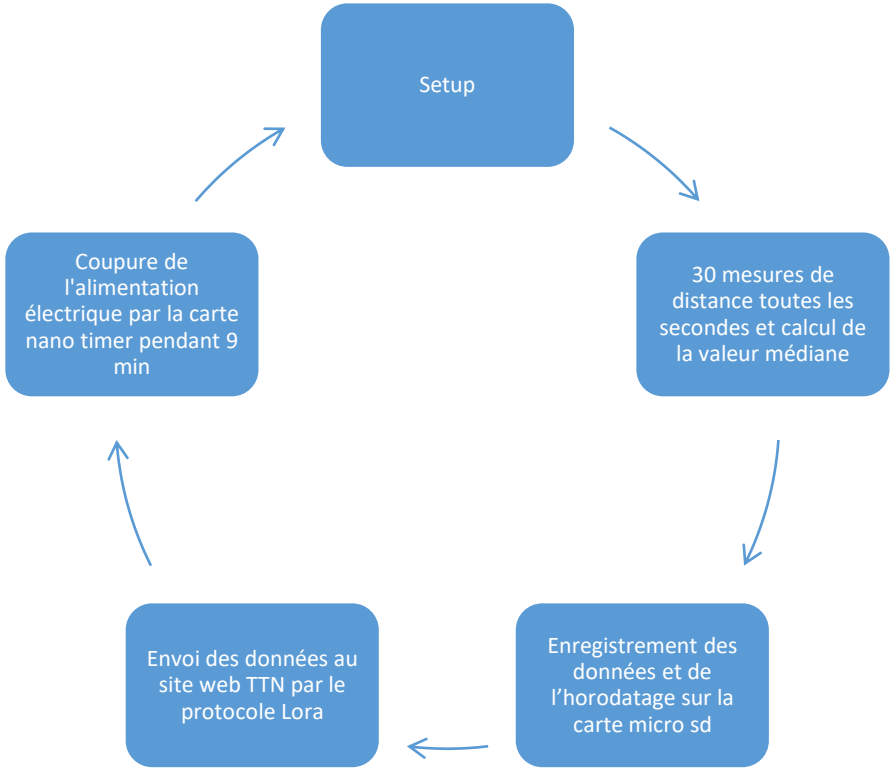




Les programmes arduino



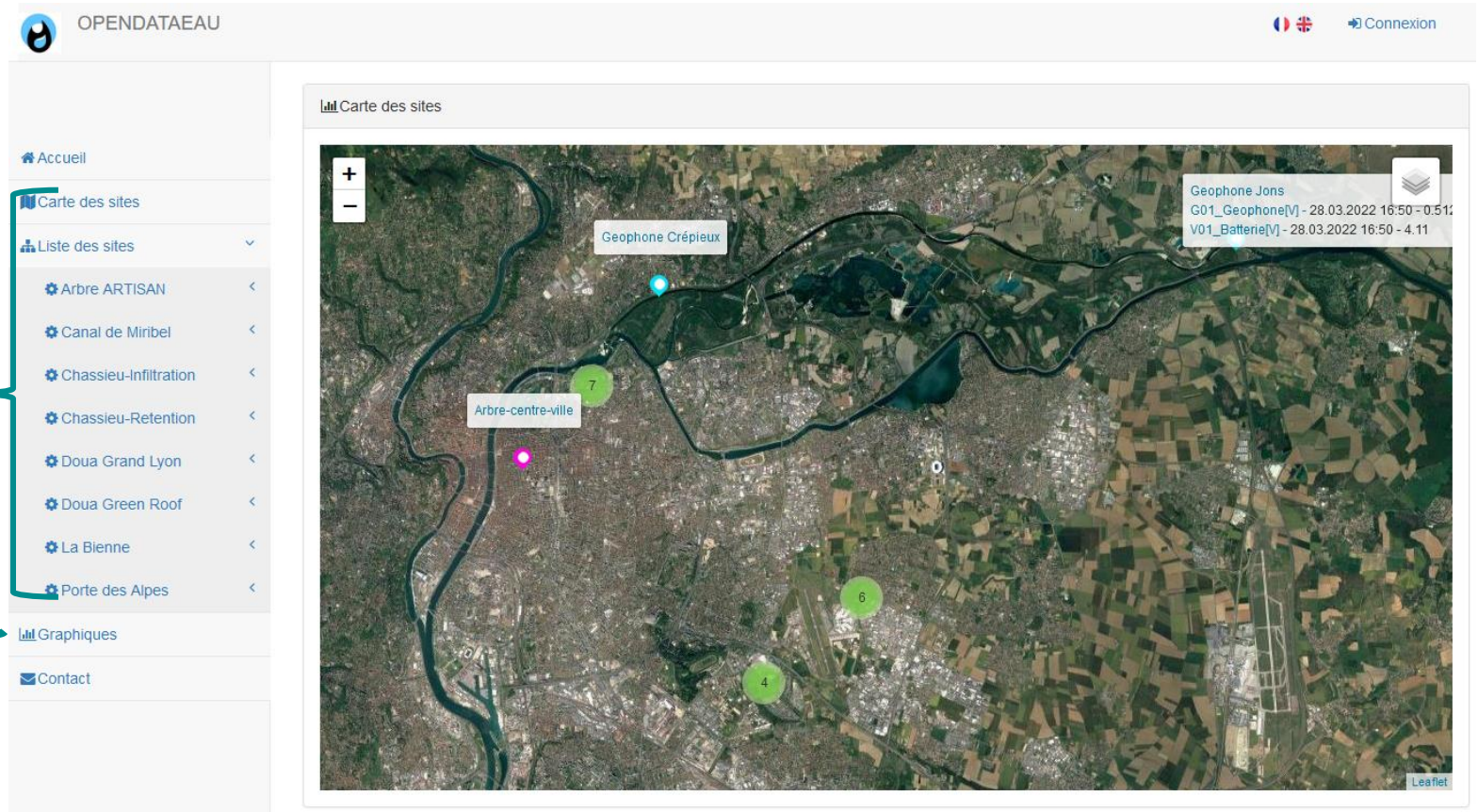
Avec alimentation électrique permanente



Sans alimentation électrique permanente

Plateforme en ligne: http://mind4stormwater.online/ode_Officiel/see//index.php?page=carte

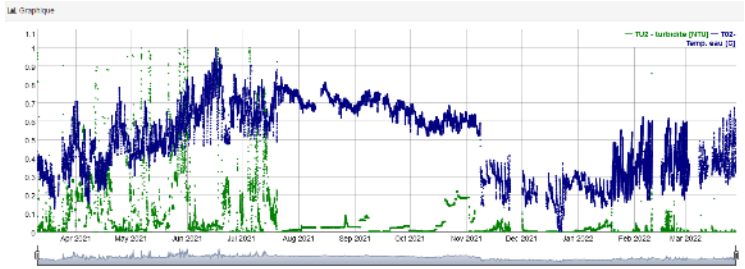
Publication en temps réel et libre accès à la donnée produite



Sites et projets

Données en temps réel

Graphiques multi-paramètres



Les stations sur le terrain

Travaux pratiques

Mode manuel

MKR-CircuitBlanc-Manuel | Arduino 1.8.13

Fichier Édition Croquis Outils Aide

MKR-CircuitBlanc-Manuel

```
/*
  Exemple de code pour faire fonctionner le circuit blanc en mode manuel.
  Pour convertir le circuit en mode automatique, s'inspirer de l'exemple de la libraire "ds3231-master" -> rtc_ds3231_alarm_every_5m
*/

#include "RTClib.h" // bilbiothèque horloge rtc
#include "ds3231.h"
#include <Wire.h> // bibliothèque communication wire


RTC_DS3231 rtc;                                                                                    //DS3231 // déclaration RTC



const int CdeTransistor = 7;


void setup()
{
  rtc.begin();                                                                                      //lance le RTC
  pinMode(CdeTransistor, OUTPUT);                                                                   //Sortie pin 7 HIGH vers transistor
  digitalWrite(CdeTransistor, HIGH);


  // following line sets the RTC to the date & time this sketch was compiled
  //   rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));

  //  rtc.adjust(DateTime(2020, 1, 13, 16, 20, 0));  // Défini une date et une heure pour le RTC (à ne faire qu'une fois)
  // Par exemple pour le 9 avril 2018 à 16:36:20
  // rtc.adjust(DateTime(2018, 4 ,9 , 16 ,36 ,20));


}

void loop()
{

  delay(10000);
                        //Programme à exécuter


  while (1) {
    digitalWrite(CdeTransistor, LOW);                                                               //Sortie pin 7 LOW  POUR MISE EN SOMMEIL DE LA CARTE MKR
    delay(5000);
```

1 Arduino MKR FOX 1200 sur COM20

Mode automatique

Exemple librairie: ds3231-master -> rtc_ds3231_alarm_every_5m

rtc_ds3231_alarm_every_5m | Arduino 1.8.13

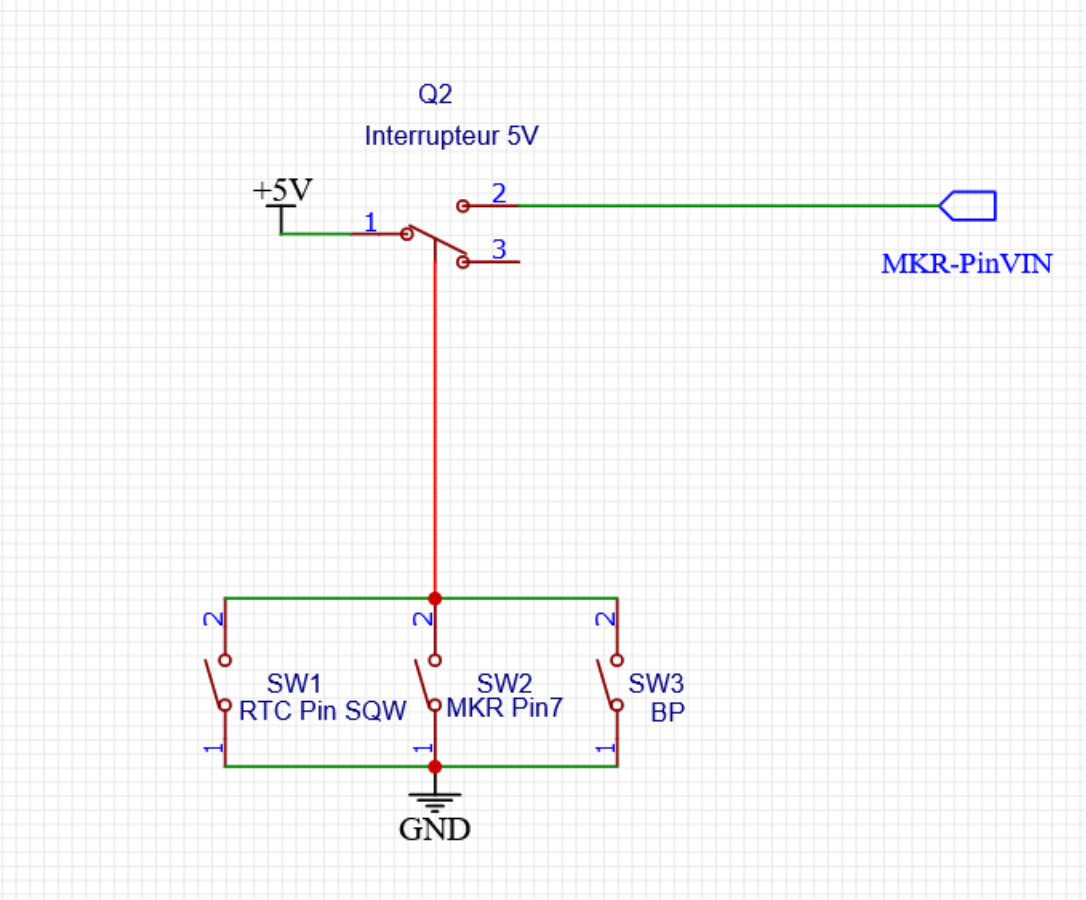
Fichier Édition Croquis Outils Aide

rtc_ds3231_alarm_every_5m

```
uint8_t sleep_period = 5;       // the sleep interval in minutes between 2 consecutive alarms

// how often to refresh the info on stdout (ms)
unsigned long prev = 5000, interval = 5000;

void set_next_alarm(void)
{
    struct ts t;
    unsigned char wakeup_min;

    DS3231_get(&t);

    // calculate the minute when the next alarm will be triggered
    wakeup_min = (t.min / sleep_period + 1) * sleep_period;
    if (wakeup_min > 59) {
        wakeup_min -= 60;
    }

    // flags define what calendar component to be checked against the current time in order
    // to trigger the alarm
    // A2M2 (minutes) (0 to enable, 1 to disable)
    // A2M3 (hour)    (0 to enable, 1 to disable)
    // A2M4 (day)     (0 to enable, 1 to disable)
    // DY/DT          (dayofweek == 1/dayofmonth == 0)
    uint8_t flags[4] = { 0, 1, 1, 1 };

    // set Alarm2. only the minute is set since we ignore the hour and day component
    DS3231_set_a2(wakeup_min, 0, 0, flags);

    // activate Alarm2
    DS3231_set_creg(DS3231_CONTROL_INTCN | DS3231_CONTROL_A2IE);
}

void setup()
{
    Serial.begin(9600);
    Wire.begin();
    DS3231_init(DS3231_CONTROL_INTCN);
    DS3231_clear_a2f();
    set_next_alarm();
}

void loop()
{
```

44 Arduino MKR WAN 1310 sur COM29

Principe de fonctionnement



- SW1 fermé -> Q2 fermé
- SW2 fermé -> Q2 fermé
- SW1 ouvert -> Q2 fermé
- SW2 fermé -> Q2 ouvert

CHEAP'EAU

SOLUTIONS INNOVANTES A BAS COUT
POUR LE SUIVI DES SYSTEMES DE GESTION DES EAUX URBAINES

Merci pour votre attention!

Des questions?