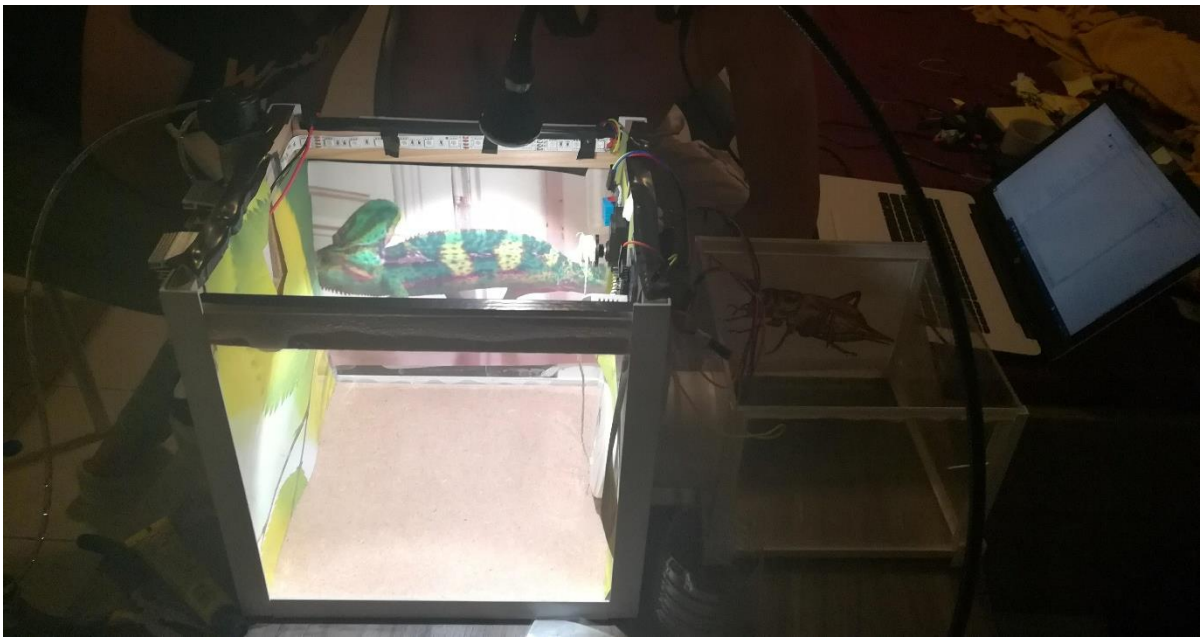
Polytech Nice-Sophia : Ecole polytechnique Universitaire

MCGANNON Sean TOGNAN Antonin VAUJANY Alexandre

Terrarduino :



Projet Arduino Peip2 2017/2018

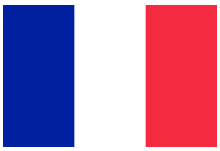
Université Nice Sophia Antipolis | POLYTECH NICE-SOPHIA

MCGANNON Sean TOGNAN Antonin VAUJANY Alexandre

Sommaire :

MCGANNON Sean TOGNAN Antonin VAUJANY Alexandre

Introduction :



Cherchant à apprendre de nouvelles choses sur des thèmes encore inconnus et voulant faire un projet en rapport avec différents animaux nous avons décidé de choisir le projet d'un Terrarium connecté que nous avons baptisé le **Terrarduino.**

Notre but étant de créer un Terrarium contrôlable à distance et qui puisse permettre aux personnes possédant un reptile de passer moins de temps à s'occuper d'eux leurs permettant de partir en vacances sans soucis ni tracas.

Nous avions aussi comme but de créer un projet à but utile et commercial. Le Terrarduino était le projet parfait pour tout cela.

Lien vers notre GitHub : https://github.com/TognanAntonin/Terrarduino

Curious of learning new things on themes unknown to us and wanting to do a project in relation with animals, we decided to create a connected terrarium which we baptised « Terrarduino ».

Our goal was to buid, out of our own hands, a terrarium controlable thanks to your phone, allowing owners of these animals to spend less time taking care of them and allow them to go on vacation without having to worry for their precious pet.

We also wanted to create something which is useful for our society and potentially lucrative. The terrarduino seemed to be the perfect project.

Polytech Nice-Sophia : Ecole polytechnique Universitaire

MCGANNON Sean TOGNAN Antonin VAUJANY Alexandre

Polytech Nice-Sophia : Ecole polytechnique Universitaire

MCGANNON Sean TOGNAN Antonin VAUJANY Alexandre

Poblématique :

Comment permettre aux reptiles de vivre en autonomie dans un terrarium ?

Université Nice Sophia Antipolis | POLYTECH NICE-SOPHIA

MCGANNON Sean TOGNAN Antonin VAUJANY Alexandre

Matériels utilisés :

Liste de notre matériel utilisé :

- Capteur de température/humidité
- Plaque chauffante
- Module RTC
- Radiateur
- Peltier
- Servomoteur
- Trappe
- Pompe
- Brumisateur
- Laser/Récepteur
- UCA Board
- 8 Relais Modules
- Bande de LED
- Adaptateur 220V/12V
- Terrarium

MCGANNON Sean TOGNAN Antonin VAUJANY Alexandre

Espèces adaptées au Terrarduino :

Rappel :

Un terrarium est un emplacement préparé et adapté pour l'élevage et l'entretien de reptiles ou insectes ou autres petits animaux terrestres. Il doit pouvoir s'adapter aux besoins des animaux ou des végétaux qui vont y vivre.

Selon les espèces, il existe différentes caractéristiques et contraintes auxquelles il faut répondre telles que les dimensions, l'éclairage, la régulation de la température, la régulation de l'hygrométrie ou encore l'aération.

Nous avons donc adapté notre code à prendre des variables différentes afin qu'il puisse s'adapter à plusieurs espèces de reptiles différents. Pour l'instant, nous avons créé une base de données pour 3 espèces différentes. Mais il suffit juste d'agrandir cette base donnée pour avoir encore plus de reptiles adaptés à notre Terrarduino.

Nos 3 espèces pouvant vivre dans notre Terrarduino celons leurs caractéristiques :

Polytech Nice-Sophia : Ecole polytechnique Universitaire

MCGANNON Sean TOGNAN Antonin VAUJANY Alexandre

Le Pogona Vitticeps

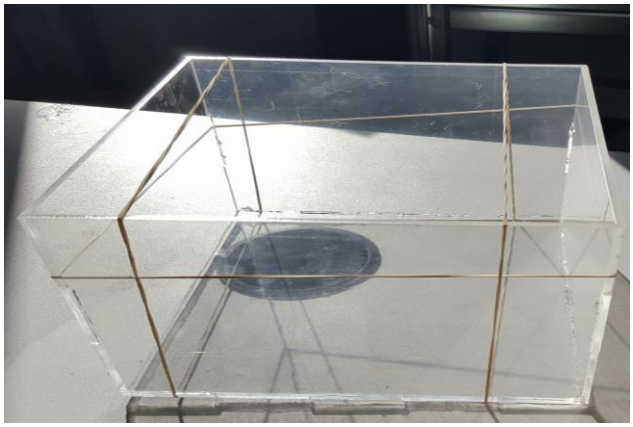
Le Caméléon Casqué du Yémen

Le Gecko Léopard

MCGANNON Sean TOGNAN Antonin VAUJANY Alexandre

Montage :

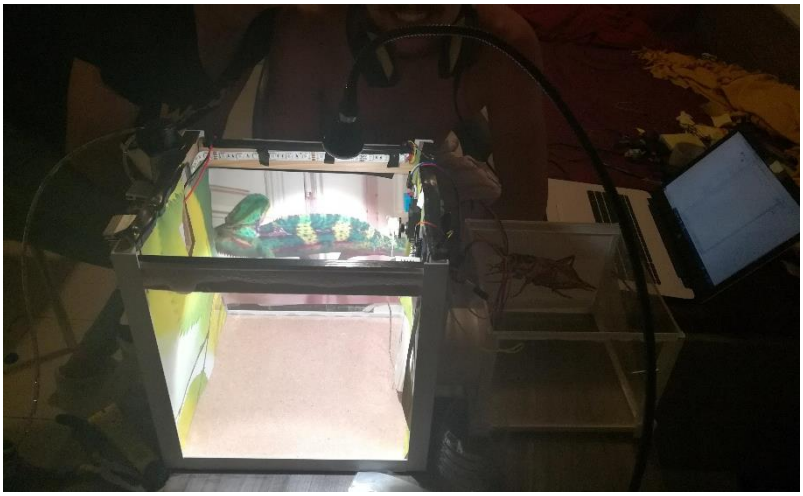
Tout d'abord, nous avons eu l'idée de créer une boîte qui nous servirait de terrarium à partir de plaques de plexiglas. Nous l'avons créé à partir d'un générateur de boîte trouvé sur internet et du découpeur laser du FabLab qui nous a découpé des plaques de plexiglas aux dimensions suivantes : 25cm pour la largeur et la profondeur et 30cm pour la hauteur.



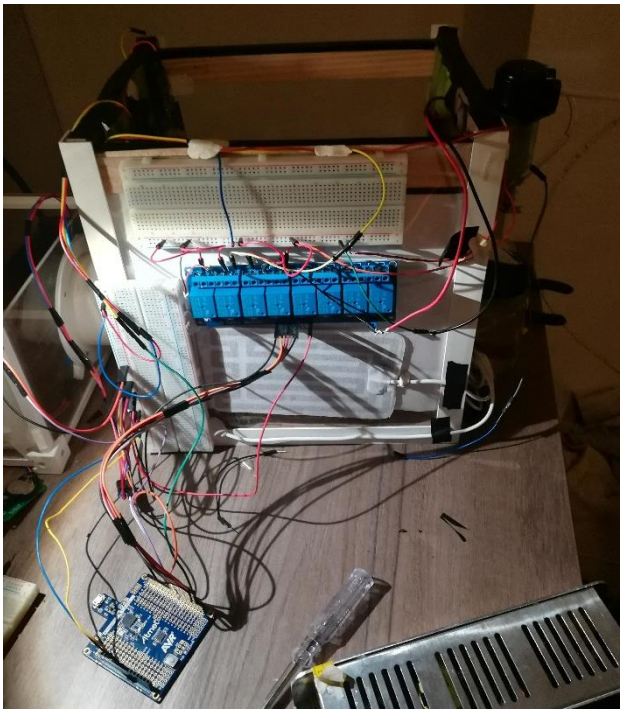
Cette boîte n'étant pas adaptée pour notre projet (trop petite et trop fragile), nous avons donc décidé servirait de boîte pour les grillons. Nous l'avons transformée pour la rendre plus esthétique et pour cacher la présence de rayures sur certains côtés.

Nous avons ensuite créé la boîte principale. Avec de l'ingéniosité, de l'huile de coude et beaucoup de temps passé au Fablab.

MCGANNON Sean TOGNAN Antonin VAUJANY Alexandre



Nous avons ensuite branché tous nos différents composant autour de se Terrarduino.



Codage :

Nous avons passé énormément de temps sur la partie codage.

En effet, de nombreuses lignes de code ont été utile pour faire marcher les différents composants électroniques et aussi pour communiquer avec notre Téléphone ou ordinateur grâce à l'UCA board.

Tous les codes sont disponibles sur notre GitHub ainsi que dans l'annexe. (Pas tous les codes dans l'annexe).

Difficultés rencontrées :

Ce projet n'a pas été facile, beaucoup de contrariétés ont freiné notre progression.

Tout d'abord, la réception tardive de certains composants nous a fait perdre du temps. Le manque d'organisation a aussi fortement contribué à ce retard, ce qui nous a obligé en conséquent à faire une grande partie du projet au dernier moment.

Ensuite, les problèmes d'ordres électriques et informatiques. Des librairies introuvables alors que nous les avions enregistrés, des codes qui marches séparément mais pas ensemble, des fils qui ne marchais pas, des petites anomalies qui nous ont fait perdre du temps car nous ne savions pas la source de ces erreurs.

Conclusion :

Ce projet a été très enrichissant pour nous. Nous avons appris de nombreuses choses sur différents sujets (électronique, reptile, organisation…)

A cause de notre manque d'organisation, nous avons manqué de temps. Nous aurions bien aimé plus aboutir notre projet, passer par d'autres composants, changer certaines choses …

Finalement, sur un plan plus personnel, nous avons apprécié ce projet malgré quelques problèmes. Il nous a fait progresser sur le travail en équipe, la gestion du temps, l'organisation et aussi individuellement.

Merci de votre lecture.

Nous voulions aussi remercier l'université de Nice, Mr Ferrero et Mr Masson parce que sans eux nous n'aurions jamais pu faire ce premier projet éducatif.

Annexe :

Laser :

```
int Laser = 6;

int Detector = 7;


void setup(){

Serial.begin (9600);

pinMode(Laser, OUTPUT);

pinMode(Detector, INPUT);

}


void loop(){

digitalWrite(Laser, HIGH);

boolean val = digitalRead(Detector);

Serial.println(val);

}
```

Servomoteur :

```
#include <Servo.h>


/* Créer un objet Servo pour contrôler le servomoteur */

Servo monServomoteur;


void setup() {


// Attache le servomoteur à la broche D9

monServomoteur.attach(9);

}


void loop() {
```

```
// Fait bouger le bras de 0° à 180°
for (int position = 0; position <= 180; position++) {
monServomoteur.write(position);
delay(15);
}

// Fait bouger le bras de 180° à 10°
for (int position = 180; position >= 0; position--) {
monServomoteur.write(position);
delay(15);
}
}
```

Système de condensation :

```
#include "DHT.h" // Librairie des capteurs DHT
#define DHTPIN 8 // pin sur lequel est branché le DHT
#define DHTTYPE DHT11 // DHT 11 (AM2302)

DHT dht(DHTPIN, DHTTYPE);

float h = 0;
float t = 0;

void setup() {
Serial.begin(9600);
dht.begin();

}

void loop() {
```

```
h = dht.readHumidity();

t = dht.readTemperature();

float hic = dht.computeHeatIndex(t,h,false);


if (isnan(t) || isnan(h)) {

Serial.println("Lecture impossible!");

}

else {

Serial.print("Humidité : ");

Serial.print(h);

Serial.print(" %\t");

Serial.print("Température :");

Serial.print(t);

Serial.println(" *C");

Serial.print("Indice de temp. : ");

Serial.print(hic);

Serial.println(" *C");

}


// faire les conditions selon les reptiles en question

// faire les conditions selon les données recues par les capteurs.




delay(1000);


}
```

Connexion UCA board :

```
#include <lmic.h>

#include <hal/hal.h>
```

```
#include <SPI.h>

#include <Wire.h>

#include "LowPower.h"


//Sensors librairies


#include "DHT.h"

#define DHTPIN D6 // what pin we're connected to //à changer pour le deuxième capteur

#define DHTTYPE DHT22 // DHT 11 //ou DHT22




#define debugSerial Serial

#define SHOW_DEBUGINFO

#define debugPrintLn(...) { if (debugSerial) debugSerial.println(__VA_ARGS__); }

#define debugPrint(...) { if (debugSerial) debugSerial.print(__VA_ARGS__); }




//Commented out keys have been zeroed for github - the lines can be copied to a keys.h file and real keys inserted



// This EUI must be in little-endian format, so least-significant-byte

// first. When copying an EUI from ttnctl output, this means to reverse

// the bytes. For TTN issued EUIs the last bytes should be 0xD5, 0xB3,

// 0x70.

static const u1_t PROGMEM APPEUI[8] = { 0x04, 0x00, 0x00, 0x72, 0x00, 0x5F, 0x63, 0x20 };
```

```
void os_getArtEui (u1_t* buf) {

memcpy_P(buf, APPEUI, 8);

}


// This should also be in little endian format, see above.

static const u1_t PROGMEM DEVEUI[8] = { 0x90, 0xA3, 0x00, 0xD0, 0x7E, 0xD5, 0xB3, 0x70 };

void os_getDevEui (u1_t* buf) {

memcpy_P(buf, DEVEUI, 8);

}


// This key should be in big endian format (or, since it is not really a

// number but a block of memory, endianness does not really apply). In

// practice, a key taken from ttnctl can be copied as-is.

// The key shown here is the semtech default key.

static const u1_t PROGMEM APPKEY[16] = { 0x30, 0x39, 0x18, 0xF1, 0xBC, 0x43, 0x96, 0xF5, 0x83, 0x2A, 0xF0, 0xB6,
0x93, 0x0B, 0xAE, 0xCD };

void os_getDevKey (u1_t* buf) {

memcpy_P(buf, APPKEY, 16);

}




static osjob_t sendjob;


// global enviromental parameters

static float temp = 0.0;

//static float pressure = 0.0;

static float humidity = 0.0;

static float batvalue;
```

```
DHT dht(DHTPIN, DHTTYPE);



// Pin mapping
const lmic_pinmap lmic_pins = {
.nss = 10,
.rxtx = LMIC_UNUSED_PIN,
.rst = 8,
.dio = {2, 7, 9},
};


// ----------------------------------------------------------------------------
// Functions
// ----------------------------------------------------------------------------



// Schedule TX every this many seconds (might become longer due to duty
// cycle limitations).
unsigned int TX_INTERVAL = 300;


void setDataRate() {
switch (LMIC.datarate) {
case DR_SF12:
#ifdef SHOW_DEBUGINFO
debugPrintLn(F("Datarate: SF12"));
#endif
TX_INTERVAL = 4800;
break;
```

```
case DR_SF11:
#ifdef SHOW_DEBUGINFO
debugPrintLn(F("Datarate: SF11"));
#endif
TX_INTERVAL = 2400;
break;
case DR_SF10:
#ifdef SHOW_DEBUGINFO
debugPrintLn(F("Datarate: SF10"));
#endif
TX_INTERVAL = 1200;
break;
case DR_SF9:
#ifdef SHOW_DEBUGINFO
debugPrintLn(F("Datarate: SF9"));
#endif
TX_INTERVAL = 600;
break;
case DR_SF8:
#ifdef SHOW_DEBUGINFO
debugPrintLn(F("Datarate: SF8"));
#endif
TX_INTERVAL = 60;
break;
case DR_SF7:
#ifdef SHOW_DEBUGINFO
debugPrintLn(F("Datarate: SF7")); //nous
#endif
```

```
TX_INTERVAL = 60; //on peut changer cette valeur pour changer le délai des relevés.
break;
case DR_SF7B:
#ifdef SHOW_DEBUGINFO
debugPrintLn(F("Datarate: SF7B"));
#endif
TX_INTERVAL = 180;
break;
case DR_FSK:
#ifdef SHOW_DEBUGINFO
debugPrintLn(F("Datarate: FSK"));
#endif
TX_INTERVAL = 180;
break;
default: debugPrint(F("Datarate Unknown Value: "));
debugPrintLn(LMIC.datarate); TX_INTERVAL = 600;
break;
}
}


extern volatile unsigned long timer0_millis;
void addMillis(unsigned long extra_millis) {
uint8_t oldSREG = SREG;
cli();
timer0_millis += extra_millis;
SREG = oldSREG;
sei();
}
```

```
void do_sleep(unsigned int sleepyTime) {

unsigned int eights = sleepyTime / 8;

unsigned int fours = (sleepyTime % 8) / 4;

unsigned int twos = ((sleepyTime % 8) % 4) / 2;

unsigned int ones = ((sleepyTime % 8) % 4) % 2;


#ifdef SHOW_DEBUGINFO

debugPrint(F("Sleeping for "));

debugPrint(sleepyTime);

debugPrint(F(" seconds = "));

debugPrint(eights);

debugPrint(F(" x 8 + "));

debugPrint(fours);

debugPrint(F(" x 4 + "));

debugPrint(twos);

debugPrint(F(" x 2 + "));

debugPrintLn(ones);

delay(500); //Wait for serial to complete

#endif



for ( int x = 0; x < eights; x++) {

// put the processor to sleep for 8 seconds

LowPower.powerDown(SLEEP_8S, ADC_OFF, BOD_OFF);

}

for ( int x = 0; x < fours; x++) {

// put the processor to sleep for 4 seconds

LowPower.powerDown(SLEEP_4S, ADC_OFF, BOD_OFF);
```

```
}
for ( int x = 0; x < twos; x++) {
// put the processor to sleep for 2 seconds
LowPower.powerDown(SLEEP_2S, ADC_OFF, BOD_OFF);
}
for ( int x = 0; x < ones; x++) {
// put the processor to sleep for 1 seconds
LowPower.powerDown(SLEEP_1S, ADC_OFF, BOD_OFF);
}
addMillis(sleepyTime * 1000);
}



long readVcc() {
long result;
// Read 1.1V reference against AVcc
ADMUX = _BV(REFS0) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
delay(2); // Wait for Vref to settle
ADCSRA |= _BV(ADSC); // Convert
while (bit_is_set(ADCSRA,ADSC));
result = ADCL;
result |= ADCH<<8;
result = 1126400L / result; // Back-calculate AVcc in mV
return result;
}


void updateEnvParameters()
{
temp = dht.readTemperature();
```

```
humidity = dht.readHumidity();

batvalue = (int)(readVcc()/10); // readVCC returns in tens of mVolt



#ifdef SHOW_DEBUGINFO

// print out the value you read:

Serial.print("Humidity : ");

Serial.println(humidity);

Serial.print("TÂ°c : ");

Serial.println(temp);

Serial.print("Vbatt : ");

Serial.println(batvalue);

#endif


}



void onEvent (ev_t ev) {

#ifdef SHOW_DEBUGINFO

Serial.print(os_getTime());

Serial.print(": ");

#endif

switch (ev) {

case EV_SCAN_TIMEOUT:

#ifdef SHOW_DEBUGINFO

debugPrintLn(F("EV_SCAN_TIMEOUT"));

#endif
```

```
break;

case EV_BEACON_FOUND:

#ifdef SHOW_DEBUGINFO

debugPrintLn(F("EV_BEACON_FOUND"));

#endif

break;

case EV_BEACON_MISSED:

//debugPrintLn(F("EV_BEACON_MISSED"));

break;

case EV_BEACON_TRACKED:

//debugPrintLn(F("EV_BEACON_TRACKED"));

break;

case EV_JOINING:

#ifdef SHOW_DEBUGINFO

debugPrintLn(F("EV_JOINING"));

#endif

break;

case EV_JOINED:

#ifdef SHOW_DEBUGINFO

debugPrintLn(F("EV_JOINED"));

#endif

setDataRate();

// Ok send our first data in 10 ms

os_setTimedCallback(&sendjob, os_getTime() + ms2osticks(10), do_send);

break;

case EV_RFU1:

#ifdef SHOW_DEBUGINFO

debugPrintLn(F("EV_RFU1"));

#endif
```

```
break;
case EV_JOIN_FAILED:
#ifdef SHOW_DEBUGINFO
debugPrintLn(F("EV_JOIN_FAILED"));
#endif

lmicStartup(); //Reset LMIC and retry
break;
case EV_REJOIN_FAILED:
#ifdef SHOW_DEBUGINFO
debugPrintLn(F("EV_REJOIN_FAILED"));
#endif

lmicStartup(); //Reset LMIC and retry
break;
case EV_TXCOMPLETE:

#ifdef SHOW_DEBUGINFO
debugPrintLn(F("EV_TXCOMPLETE (includes waiting for RX windows)"));
#endif

if (LMIC.txrxFlags & TXRX_ACK)
#ifdef SHOW_DEBUGINFO
debugPrintLn(F("Received ack"));
#endif

if (LMIC.dataLen) {
```

```
#ifdef SHOW_DEBUGINFO
debugPrintLn(F("Received "));
debugPrintLn(LMIC.dataLen);
debugPrintLn(F(" bytes of payload"));
#endif

}
// Schedule next transmission
setDataRate();
do_sleep(TX_INTERVAL);
os_setCallback(&sendjob, do_send);
break;
case EV_LOST_TSYNC:
#ifdef SHOW_DEBUGINFO
debugPrintLn(F("EV_LOST_TSYNC"));
#endif
break;
case EV_RESET:
#ifdef SHOW_DEBUGINFO
debugPrintLn(F("EV_RESET"));
#endif
break;
case EV_RXCOMPLETE:
// data received in ping slot
#ifdef SHOW_DEBUGINFO
debugPrintLn(F("EV_RXCOMPLETE"));
#endif
break;
case EV_LINK_DEAD:
```

```
#ifdef SHOW_DEBUGINFO
debugPrintLn(F("EV_LINK_DEAD"));
#endif
break;
case EV_LINK_ALIVE:
#ifdef SHOW_DEBUGINFO
debugPrintLn(F("EV_LINK_ALIVE"));
#endif
break;
default:
#ifdef SHOW_DEBUGINFO
debugPrintLn(F("Unknown event"));
#endif
break;
}
}



void do_send(osjob_t* j) {
// Check if there is not a current TX/RX job running
if (LMIC.opmode & OP_TXRXPEND) {
debugPrintLn(F("OP_TXRXPEND, not sending"));
} else {
// Prepare upstream data transmission at the next possible time.
// Here the sensor information should be retrieved


updateEnvParameters();
```

```
#ifdef SHOW_DEBUGINFO

debugPrint(F("T="));

debugPrintLn(temp);


debugPrint(F("H="));

debugPrintLn(humidity);

debugPrint(F("BV="));

debugPrintLn(batvalue);

#endif

int t = (int)((temp) * 10.0);

int h = (int)(humidity * 2.0);

int bat = batvalue; // multifly by 10 for V in Cayenne


unsigned char mydata[11];

mydata[0] = 0x1;

mydata[1] = 0x67;

mydata[2] = t >> 8;

mydata[3] = t & 0xFF;

mydata[4] = 0x2;

mydata[5] = 0x68;

mydata[6] = h & 0xFF;

mydata[7] = 0x3;

mydata[8] = 0x2;

mydata[9] = bat >> 8;

mydata[10] = bat & 0xFF;


LMIC_setTxData2(1, mydata, sizeof(mydata), 0);

debugPrintLn(F("PQ")); //Packet queued
```

```
}
// Next TX is scheduled after TX_COMPLETE event.
}

void lmicStartup() {
// Reset the MAC state. Session and pending data transfers will be discarded.
LMIC_reset();

LMIC_setLinkCheckMode(1);
LMIC_setAdrMode(1);
LMIC_setClockError(MAX_CLOCK_ERROR * 1 / 100); // Increase window time for clock accuracy problem


// Start job (sending automatically starts OTAA too)
// Join the network, sending will be
// started after the event "Joined"
LMIC_startJoining();
}

void setup() {
Serial.begin(115200);
delay(1000); //Wait 1s in order to avoid UART programmer issues when a battery is used

Serial.begin(115200);

#ifdef SHOW_DEBUGINFO
debugPrintLn(F("Starting"));
delay(100);
```

```
#endif


Wire.begin();


updateEnvParameters(); // To have value for the first Tx



// LMIC init

os_init();

lmicStartup();


}


void loop() {

os_runloop_once();

}
```

Eclairage :

```
//clk sur l'entrée 5

//data sur l'entrée 4

//rst sur l'entrée 3


#include <DS1302.h>


// Init the DS1302

DS1302 rtc(3,4,5); //rst,data,clk

const int bande = 2;

Time t;


void setup() {
```

```
// put your setup code here, to run once:

rtc.halt(false);

rtc.writeProtect(false);

digitalWrite(bande,HIGH);

pinMode(bande,OUTPUT);

Serial.begin(9600);


}


void loop() {

t = rtc.getTime();

Serial.println(rtc.getTimeStr());

delay(5000);

if ((t.hour > 07) && (t.hour < 19)) {

digitalWrite(bande, LOW);

Serial.println(" Light on ");

}

else {

digitalWrite(bande, HIGH);

Serial.println(" Light off ");

}

}
```

Le reste des codes sont sur notre Github.