- Accueil

Projets domotiques dans la Maison Storck

Projets domotiques dans la Maison Storck

Feed

Contrôle et supervision de la piscine

Août 30th 2014

By: Clément

108 comments

A A

Après une longue absence sur le blog, voici mon dernier projet : connecter la piscine à la domotique.

Pourquoi ?

Nous avons une piscine équipée d'un système de régulation « automatique » du pH et du chlore (par électrolyse). Sauf que la régulation du chlore n'est pas si « automatique » que ça. L'appareil que nous avons est le JustSalt+ de Pool Technologie. La production de chlore est définie selon un pourcentage du temps de fonctionnement et non selon la mesure du chlore présent dans l'eau. Seuls les modèles apparus plus tard (comme le JustSalt PRO) sont équipés d'une sonde ORP (mesure du taux de chlore) et sont donc capables d'ajuster intelligemment la production de chlore.
Avec le temps nous avons remarqué que le chlore varie énormément selon l'ensoleillement, la fréquentation et la présence de la couverture. Ainsi, lorsque la piscine est fermée, j'ai remarqué que le réglage doit être d'environ 5% et de 50% à 80% lorsqu'elle est ouverte. Mais ceci dépend également du taux de stabilisateur présent dans l'eau. En gros, c'est tout sauf pratique et comme résultat, le taux de chlore était rarement bon…

Solution

Pour y remédier et avoir enfin un système vraiment automatique, l'idée est simple, il suffit de rajouter une sonde ORP/Redox dans le système de filtration et si le taux est inférieur à un seuil, activer l'électrolyse jusqu'à atteindre un autre seuil…
Mais pourquoi s'arrêter là ? :)

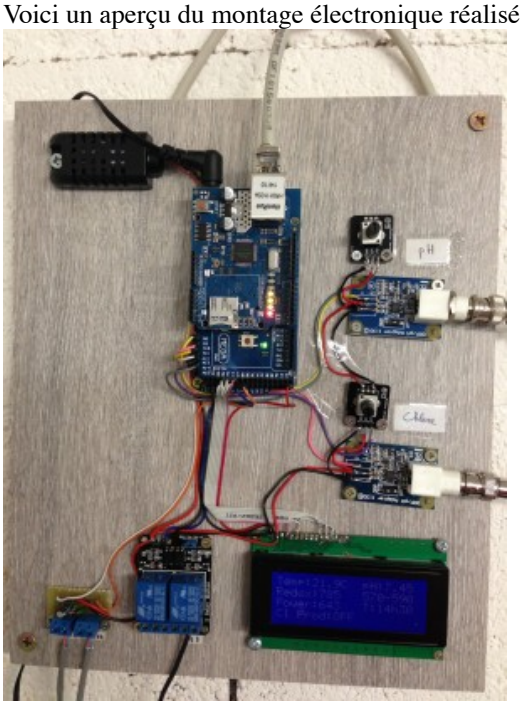
Le cahier des charges est donc le suivant :

Activation Pompe à Eau et Pompe à chaleur individuelle.

- activation de l'électrolyse selon taux de chlore
- le système doit être autonome et fonctionner même en cas de panne du serveur domotique
- possibilité de forcer à distance le fonctionnement de l'électrolyseur de la pompe (ON/OFF, Manuel/Auto)
- affichage de graphiques dans l'interface domotique de la maison
  - taux de chlore Brome
  - taux de pH

    - température de l'eau
    - température et humidité du local piscine
  - mesure de la consommation électrique du système de filtration
  - calcule du temps de fonctionnement de la filtration par 24h
  - détection d'inondation dans le local
  - affichage local des éléments mesurés sur un écran
  - calibration manuelle des sondes pH et ORP

Au vu du nombre d'éléments à connecter, je suis parti sur un Arduino Mega avec un shield Ethernet Wiznet.
Voici un aperçu du montage électronique réalisé :



Mesure du taux de chlore

J'ai acheté une sonde ORP/Redox sur eBay pour environ 25€ (les premiers prix).

Ces sondes mesurent le potentiel d'oxydoréduction (appelé Redox) définie entre -2000mV et +2000mV. Si le Redox est supérieur à 0, l'eau est oxydante. Plus cette valeur est élevée, plus il y a de chlore dans l'eau.
Cet article décrit très bien le principe des sondes ORP : http://www.piscine-clic.com/news/2014/02/potentiel-redox-kezako/

Le problème est que cette sonde sort un courant très faible, on ne peut pas la lire précisément de cette manière, il faut y ajouter un amplificateur qui va par la même occasion relever la tension à 0V-5V pour la lire via un Arduino.

Pour cela j'ai acheté un Adaptateur pH/ORP 1130 chez Go Tronic (http://www.gotronic.fr/art-adaptateur-ph-orp-1130-12112.htm) pour moins de 30€ :

La documentation complète de cet adaptateur se trouve ici : http://www.phidgets.com/docs/1130_User_Guide

Notez que pour avoir une mesure précise à l'entré de l'Arduino, il vaut mieux avoir une alimentation de qualité. N'utilisez pas le port USB de l'Arduino, cela bypass le régulateur 5V interne.

Mesure du pH

Le principe est exactement le même que pour la sonde OPR. La piscine était déjà équipée d'une sonde pH, j'ai simplement mis un Y sous le régulateur pH pour lire la valeur de la sonde depuis l'adaptateur.

La formule de calcul est à adapter. Elle est indiquée sur la documentation de l'adaptateur.

Mesure de la température et humidité du local

J'ai acheté une sonde DHT21/AM2301 pour 3€ sur eBay. Il faut utiliser la libraire DHT.h pour lire les valeurs sur l'Arduino.

Contrôle de l'électrolyseur

L'électrolyseur est équipé d'une entré type contact sec qui permet de connaitre l'état d'un volet automatique (piscine ouverte ou fermée). Le but est d'activer un mode de production différent quand la piscine est fermée. C'est pratique, mais pour cela il faut avoir une couverture électrique.
J'ai relié cette entrée à la sortie d'un relai. Du coup, l'Arduino fait croire à l'électrolyseur que la piscine est fermée pour stopper la production.

Détection d'inondation

J'ai simplement connecté 2 fils qui arrivent au sol espacés de deux millimètres. En cas d'eau, cela fera contact et l'Arduino détectera un état haut sur son entrée numérique. Pour simplifier le câblage et ne pas rajouter de résistance, j'ai configuré l'entrée en Pullup interne (*pinMode(pin_water_sensor, INPUT_PULLUP)*).

Pour l'instant, l'Arduino envoie une information xPL sur le réseau en cas d'inondation. Je changerais cela par une requête HTTP directe à PushingBox pour ne pas dépendre du serveur domotique.

Mesure de la consommation électrique

J'ai acheté une pince ampèremétrique (30A SCT-013-030 Non-invasive AC current sensor Split Core Transformer BA) pour 5€ sur eBay que j'ai placé autour de la phase de l'arrivée électrique de la pompe.

Pour le connecter à l'Arduino il faut réaliser un petit circuit électronique de 3 résistances et une capacité afin de relever la tension alternative proche de +2,5v qui est au départ autour de 0V (l'Arduino ne peut lire des tensions négatives).

Ce site explique le montage à réaliser : http://www.homautomation.org/2013/09/17/current-monitoring-with-non-invasive-sensor-and-arduino/

J'ai dû adapter la valeur de la résistance de Burden pour avoir une lecture plus juste. Ensuite, il suffit d'utiliser la library Arduino EmonLib.h d'EnergyMonitor qui s'occupe de récupérer des échantillons de la valeur alternative à une fréquence très élevée pour en récupérer la consommation électrique.

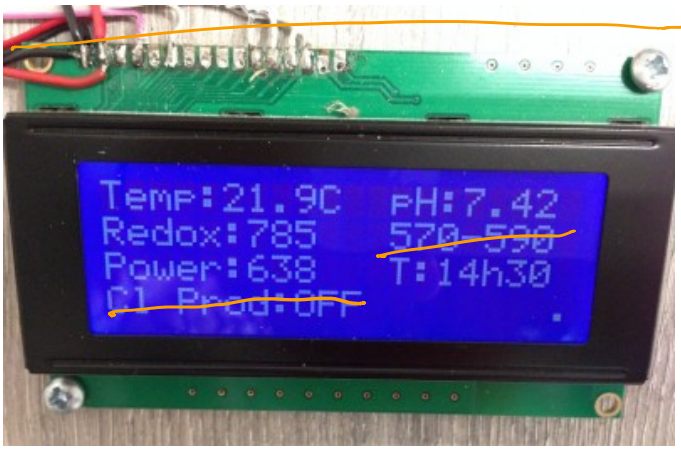
Température de l'eau

Cela fait quelque temps que c'est en place, j'utilise une sonde Oregon THWR800 qui transmet à mon RFXCom la température.

Écran LCD

J'ai acheté un écran LCD 20×4 caractères pour 5€ sur eBay et j'utilise la librairie LiquidCrystal.h.

Petit conseil, ne pas acheter les écrans avec interface I2C intégrée, ça simplifie le câblage, mais ça oblige à utiliser une libraire spécifique à chaque constructeur car il n'y a pas de « normes » pour le câblage interne de ces modules…



Sur la première ligne sont affichés la température du local et le pH.
Sur la deuxième ligne, le Redox (soit le taux de chlore). À droite sont indiqués les seuils désirés du Redox. On voit qu'à ce moment, le chlore était beaucoup trop élevé… Les seuils peuvent être ajustés manuellement via le potentiomètre de calibration (pour le redox, je calibre le seuil et non la sonde).
Sur la troisième ligne, on voit la puissance électrique consommée en Watt par le système de filtration et la durée de filtration (14h30 par jour). La durée de filtration doit être de : température de l'eau divisée par deux. On est bon car l'eau est à 28°C.
Enfin, la dernière ligne indique l'état de l'électrolyseur.

Envoie des données vers le système domotique

J'ai utilisé l'excellente librairie xPL (développé par un Français!) : https://github.com/olebrun/xPL.Arduino
L'Arduino envoie toutes les 30 secondes les valeurs :

```
192.168.100.104:43932 [xpl-trig/sensor.basic: xpl-arduino.piscine -> * - piscine[ph]=7.18]
192.168.100.104:43932 [xpl-trig/sensor.basic: xpl-arduino.piscine -> * - piscine[redox]=  690]
```

```
192.168.100.104:43932 [xpl-trig/sensor.basic: xpl-arduino.piscine -> * - piscine[temp]=24.00]
192.168.100.104:43932 [xpl-trig/sensor.basic: xpl-arduino.piscine -> * - piscine[humidity]=64.30]
192.168.100.104:43932 [xpl-trig/sensor.basic: xpl-arduino.piscine -> * - piscine[justsaltstate]=off]
192.168.100.104:43932 [xpl-trig/sensor.basic: xpl-arduino.piscine -> * - piscine[current]= 638]
192.168.100.104:43932 [xpl-trig/sensor.basic: xpl-arduino.piscine -> * - piscine[state]=on]
```

Ensuite le serveur domotique utilise le programme xPL-Perl pour enregistrer automatiquement les données dans une base RRDTools et en tracer des graphiques toutes les 5min.

Contrôle de la production à distance

Il est possible de forcer la production depuis le système domotique. Je ne l'utilise pas car le but est que cela soit autonome. Cependant, il suffit d'envoyer cette commande xPL :

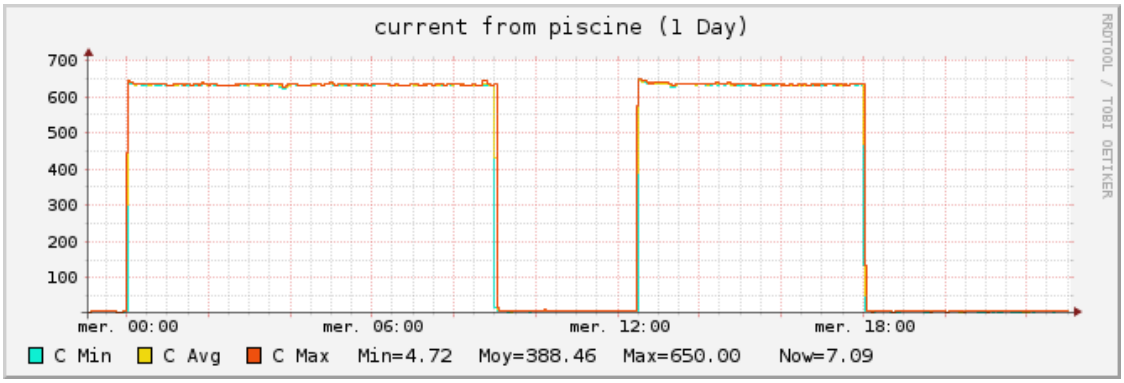
```
xpl-sender -m xpl-cmnd -t xpl-arduino.piscine -c justsalt.on000000
```

Les « 0 » rajoutés à la fin sont à cause d'un bug dans la libraire XPL (corrigé depuis) qui n'accepte que les commandes de 8 caractères…

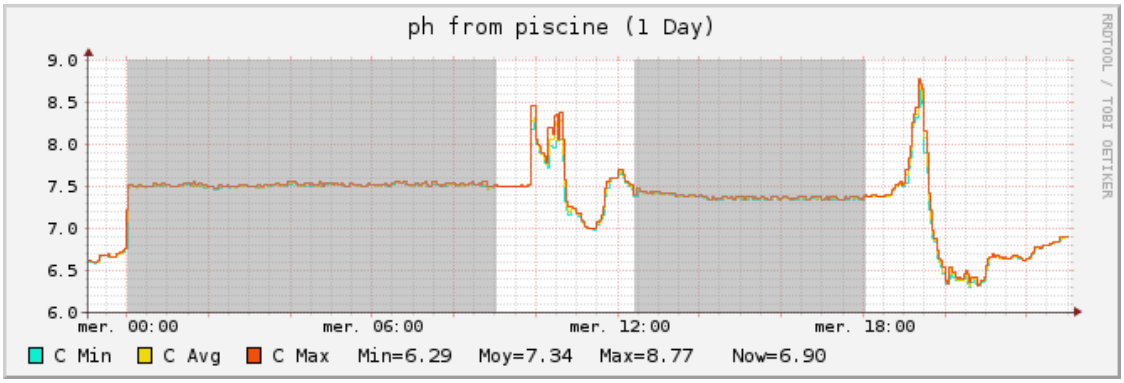
Génération des graphiques et interprétations

Le serveur domotique génère automatiquement des graphiques de type RRD pour les heures, jours, semaines, mois et années.

Ci-dessous, le résultat de la consommation électrique. On voit que la pompe consomme environ 650W mais également qu'il n'y a pas eu de production de chlore (car l'appareil consomme environ 40W). C'est grâce à cette mesure qu'on en déduit l'état et ainsi le temps de fonctionnement de la filtration.

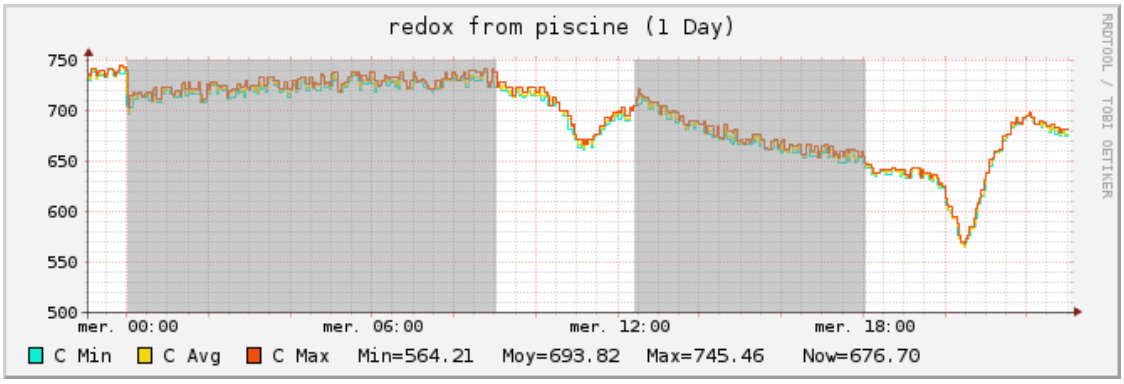


Ci-dessous, le graphique de la valeur du pH. Les zones grises représentent les périodes de filtrations. Lorsque l'eau stagne dans les tuyaux, la sonde mesure des valeurs erronées. On en déduit que le pH varie très peu et qu'il plafonne à 7.5. C'est bon signe car l'appareil qui régule le pH est configuré à 7.5 :)



Ci-dessous, le graphique du Redox (le taux de chlore). Encore une fois, il faut seulement regarder les zones grises.
On constat que de minuit à 9h du matin, le chlore a très légèrement augmenté. Je ne suis pas certain de pouvoir l'expliquer, mais le chlore augmente très doucement lorsque la piscine est

fermée (et la production désactivée).
L'information intéressante est de midi à 18h, la piscine était ouverte et le temps ensoleillé. On remarque donc à quel point le chlore diminue comparé à la nuit. Ce n'était pas encore suffisant pour déclencher la production de chlore car celui-ci était déjà trop élevé mais la valeur idéale est entre 570 et 590mV.



Code Arduino

```
#include <avr/wdt.h>       // Watchdog
#include <SPI.h>
#include <Ethernet.h>
#include <EthernetUdp.h>
#include "xPL.h"
#include "DHT.h"                    // DTH21 Temperature and Humidity.
#include <LiquidCrystal.h>  // LCD
#include "EmonLib.h"           // Power current/sensor
EnergyMonitor emon1;         // Create an instance for power current


uint8_t pin_ph = A8;                        // Pin pH Probe
uint8_t pin_redox = A9;                     // Pin Redox Probe
uint8_t pin_relay_justsalt = 41;            // Pin to Relay to JustSalt. If state=hig
uint8_t pin_dth = A10;                      // Pin sensor DTH21 Temperature and Humid
int pin_water_sensor = 39;                          // Pin water leak sensor. Is High
uint8_t pin_power_sensor = A11;             // Pin to power sensor.
uint8_t pin_potentiometer_redox = A12; // Pin to potentiometer used to define redox
uint8_t pin_potentiometer_ph = A13;     // Pin to the potentiometer used to calibrat

float ph_sensor_value = 0.0;                    // value read in Volt (0 to 5)
float ph_value_float = 0.0;                         // pH value from 0.0 to 14.0 in
char ph_value_char[5];                          // pH value from 0 to 14 in char

float redox_sensor_value = 0.0;              // value read in Volt (0 to 5)
float redox_value_float = 0.0;              // redox value from -2000 to 2000 mV in
char redox_value_char[5];                           // redox value from -2000 to 200
int redox_max;                                      // define the max value of redox
int redox_min;                                      // define the min value of redox
int redox_range_delta = 20;                // used to calculate the range (from-to)

float temperature_float = 0.0;
char temperature_char[5];
float humidity_float = 0.0;
char humidity_char[5];

float power_value_float = 0.0;                  // Power consumption in Watt
char power_value_char[5];

bool filtration_bool = 0;                               // Filtration state. 0 is o
int counter_filtration = 0;                             // Count minutes of filtrat

unsigned long lastReadingTime = 0;
int count_time_30s = 0;                                         // used to trigger
int count_time_30min = 0;                                               // used to
int count_time_24h = 0;                                         // used to trigger

byte mac[] = { 0x00, 0xAC, 0xAE, 0x3F, 0xF1, 0xAD };     // Production MAC address
IPAddress broadcast(192, 168, 100, 255);
```

```
EthernetUDP Udp;
xPL xpl;

LiquidCrystal lcd(42, 43, 44, 45, 46, 47);
DHT dht(pin_dth, DHT21);


void setup()
{
        //Watchdog part
        MCUSR &= ~_BV(WDRF);                    // Clear the reset bit
        WDTCSR |= _BV(WDCE) | _BV(WDE); // Disable the WDT
        WDTCSR = 0;

        Serial.begin(57600);
        pinMode(pin_relay_justsalt, OUTPUT);
        digitalWrite(pin_relay_justsalt, 1);    // do not start clorine production
        pinMode(pin_water_sensor, INPUT_PULLUP);
        digitalWrite(pin_water_sensor, HIGH);

        dht.begin();                             // Start tempature and humidity se
        emon1.current(11, 66);               // Power Current: input pin, calibrati

        lcd.begin(20, 4);                        // Init LCD screen, 4 lignes by 20
        lcd.clear();
        // Print the default text on the LCD.
        lcd.setCursor (0, 0);
        lcd.print("Temp:");
        lcd.setCursor (12, 0);
        lcd.print("pH:");
        lcd.setCursor (0, 1);
        lcd.print("Redox:");
        lcd.setCursor (0, 2);
        lcd.print("Power:");
        lcd.setCursor (12, 2);
        lcd.print("T:");
        lcd.setCursor (14, 2);
        lcd.print("...");
        lcd.setCursor (0, 3);
        lcd.print("Cl Prod:");


        Serial.println(F(""));
        Serial.println(F("Starting (v1.0)"));
        printMac(mac);
        delay(100);       // delay to boot in case of multiple DHCP requests from o
        {
                Serial.println(F("Failed to configure Ethernet using DHCP"));
        }
        printIP();                      // Show IP in serial monitor
        Udp.begin(xpl.udp_port);

        xpl.SendExternal = &SendUdPMessage;        // pointer to the send callback
        xpl.AfterParseAction = &AfterParseAction;  // pointer to a post parsing act
        xpl.SetSource_P(PSTR("xpl"), PSTR("arduino"), PSTR("piscine")); // paramete

        wdt_enable(WDTO_4S); //enable 4s watchdog
}


void loop()
{


        xpl.Process();  // heartbeat management

        // Parser part. Read input XPL message
        if(Udp.parsePacket())
        {
                char xPLMessageBuff[XPL_MESSAGE_BUFFER_MAX];
                Udp.read(xPLMessageBuff, XPL_MESSAGE_BUFFER_MAX);       // read the
                xpl.ParseInputMessage(xPLMessageBuff);              // parse messag
        }

        // Protect if millis return to 0 (every 50 days)
        if (millis() - lastReadingTime < 0)
```

```
        {
                lastReadingTime = millis();
        }

        // Show datas on LCD every 2 seconds
        if ((millis() - lastReadingTime) >= 2000)
        {
                // pH Part
                ph_sensor_value = analogRead(pin_ph) * 5000.0 / 1023.0 / 1000.0; //
                ph_value_float = (0.0178 * ph_sensor_value * 200.0) - 1.889;      //
                //Serial.println(ph_value_float);
                //Serial.println(analogRead(pin_potentiometer_ph));
                // add calibration
                ph_value_float = ph_value_float + (analogRead(pin_potentiometer_ph)-
                lcd.setCursor (15, 0);
                lcd.print("     ");     // Clean lcd old digits
                lcd.setCursor (15, 0);
                lcd.print(ph_value_float, 2);

                // Redox Part
                redox_sensor_value = analogRead(pin_redox) * 5000.0 / 1023.0 / 1000
                redox_value_float = ((2.5 - redox_sensor_value) / 1.037) * 1000.0;
                lcd.setCursor (6, 1);
                lcd.print("      ");    // Clean lcd old digits
                lcd.setCursor (6, 1);
                lcd.print(redox_value_float, 0);
                // get min-max redox values accepted
                int potentiometer_redox = analogRead(pin_potentiometer_redox);
                potentiometer_redox = map(potentiometer_redox, 0, 1023, 300, 900);
                redox_min = potentiometer_redox/10*10 - redox_range_delta/2;
                redox_max = potentiometer_redox/10*10 + redox_range_delta/2;
                lcd.setCursor (12, 1);
                lcd.print(redox_min);
                lcd.print("-");
                lcd.print(redox_max);

                // DHT Temp and humidity Part
                temperature_float = dht.readTemperature();
                humidity_float = dht.readHumidity();
                lcd.setCursor (5, 0);
                lcd.print("       ");   // Clean lcd old digits
                lcd.setCursor (5, 0);
                lcd.print(temperature_float, 1);
                lcd.print("C");

                // Relay JustSalt part. Chlorin Production
                lcd.setCursor (8, 3);
                if (digitalRead(pin_relay_justsalt) == 0)
                {
                        lcd.print("ON ");
                }
                else
                {
                        lcd.print("OFF");
                }

                // Power sensor
                double Irms = emon1.calcIrms(1480);  // Calculate Power current (Ir
                power_value_float = Irms * 232.0;
                lcd.setCursor (6, 2);
                lcd.print("      ");    // Clean lcd old digits
                lcd.setCursor (6, 2);
                lcd.print(power_value_float, 0);

                // Power state
                if (power_value_float > 300)    // Power is more than 300W
                {
                        filtration_bool = 1;        // on: filtration in progress
                }
                else
                {
                        filtration_bool = 0;
                        digitalWrite(pin_relay_justsalt, 1);    // salt production
                }

                // Water leak detection
```

```
        if (digitalRead(pin_water_sensor) == LOW)
        {
                lcd.setCursor (19, 3);
                lcd.print("!");
        }
        else
        {
                lcd.setCursor (19, 3);
                lcd.print(".");
        }
        //Serial.print(digitalRead(pin_water_sensor));
        //lcd.setCursor (19,3);
        //lcd.print(digitalRead(pin_water_sensor));


        count_time_30s++;       // Count 15 cycles for sending XPL every 30s
        lastReadingTime = millis();
}

// Send datas as xPL Message every 30 seconds
if (count_time_30s == 15)
{
        // pH Part
        dtostrf(ph_value_float , 3, 2, ph_value_char);
        print_sensor_value("pH", analogRead(pin_ph), ph_value_float);
        send_xpl_message("ph", ph_value_char);

        // Redox Part
        dtostrf(redox_value_float, 5, 0, redox_value_char);
        print_sensor_value("Redox", analogRead(pin_redox), redox_value_float
        send_xpl_message("redox", redox_value_char);

        // DHT Temp and humidity Part
        //Send temperature to XPL
        dtostrf(temperature_float , 3, 2, temperature_char);
        send_xpl_message("temp", temperature_char);          //send xpl messa
        //Send humidity to XPL
        dtostrf(humidity_float , 3, 2, humidity_char);
        send_xpl_message("humidity", humidity_char);    //send xpl message

        // Relay JustSalt part
        if (digitalRead(pin_relay_justsalt) == 0)
        {
                send_xpl_message("justsaltstate", "on");        // on: elect
        }
        else
        {
                send_xpl_message("justsaltstate", "off");
        }

        // Power sensor
        dtostrf(power_value_float , 4, 0, power_value_char);
        send_xpl_message("current", power_value_char);  //send xpl message

        // Power state
        if (filtration_bool == 1)                   // Power is more than 3
        {
                send_xpl_message("state", "on");        // on: filtration i
        }
        else
        {
                send_xpl_message("state", "off");
        }

        // Water leak sensor
        if (digitalRead(pin_water_sensor) == LOW)
        {
                send_xpl_message("leak", "on");         // water leak detec
        }


        //if (count_time_30min % 2 == 0)                // every 1min, used
        //{
                if (filtration_bool == 1)                   // if filtratio
                {
                        counter_filtration++;
```

```
                        }
                        //itoa (counter_filtration, char test, 10);
                        //send_xpl_message("timer", String(counter_filtration));
                        count_time_24h++;
                //}

                count_time_30s = 0;
                count_time_30min++;
        }


        if (count_time_30min == 60)             // every 30min (60*30s)
        {

                if ((redox_value_float > redox_max) && (filtration_bool == 1))
                {
                        digitalWrite(pin_relay_justsalt, 1);
                }
                if ((redox_value_float < redox_min) && (filtration_bool == 1))
                {
                        digitalWrite(pin_relay_justsalt, 0);
                }

                count_time_30min = 0;
        }

        if (count_time_24h == 2880)                     // every 24h (1440*1min)
        {
                //Serial.println(count_time_24h);
                lcd.setCursor (14, 2);
                lcd.print("      ");
                lcd.setCursor (14, 2);
                lcd.print(counter_filtration/60);
                lcd.print("h");
                if (counter_filtration%60 < 10)
                {
                        lcd.print("0");
                }
                lcd.print(counter_filtration%60);

                counter_filtration = 0;
                count_time_24h = 0;
        }


        wdt_reset();  //Reset the Watchdog timer
}



// Send UDP Message
void SendUdPMessage(char *buffer)
{
        Udp.beginPacket(broadcast, xpl.udp_port);
        Udp.write(buffer);
        Udp.endPacket();
}

// Print MAC Address
void printMac (const byte *buf)
{
        Serial.print(F("MAC: "));
        for (byte i = 0; i < 6; ++i)
        {
                if (buf[i] >= 0 && buf[i] <= 16)
                        Serial.print(F("0"));
                Serial.print( buf[i], HEX );
                if (i < 5)
                        Serial.print(F(":"));
        }
        Serial.println("");
}

// Print IP address
void printIP()
```

```
{
        // print your local IP address:
        Serial.print(F("My IP address: "));
        for (byte thisByte = 0; thisByte < 4; thisByte++)
        {
                Serial.print(Ethernet.localIP()[thisByte], DEC);
                Serial.print(F("."));
        }
        Serial.println();
}

// Print debug info into serial monitor
void print_sensor_value(char* name, int sensor_value, float value)
{
        //print the results to the serial monitor for debug:
        Serial.print(name);
        Serial.print(F(" sensor: "));
        Serial.print(sensor_value);
        Serial.print(F(" output: "));
        Serial.println(value);
}

// Send XPL Message
void send_xpl_message(char* type, char* current)
{
        xPL_Message msg;
        msg.hop = 1;
        msg.type = XPL_TRIG;
        msg.SetTarget_P(PSTR("*"));
        msg.SetSchema_P(PSTR("sensor"), PSTR("basic"));
        msg.AddCommand_P(PSTR("device"), PSTR("piscine"));
        msg.AddCommand("type", type);
        msg.AddCommand("current", current);
        xpl.SendMessage(&msg);
}

// Parse input XPL messages
// usage: xpl-sender -m xpl-cmnd -t xpl-arduino.piscine -c justsalt.on000000
// should have a length of 8 for class because of a bug in the lib :(. That's why I
void AfterParseAction(xPL_Message * message)
{
        if (xpl.TargetIsMe(message))
        {
                // If we get an XPL packet, then turn on or off the Chlorine produc
                if (message->IsSchema_P(PSTR("justsalt"), PSTR("on000000")))
                {
                        digitalWrite(pin_relay_justsalt, 0);
                        Serial.println(F("Turning ON Justsalt"));
                        send_xpl_message("justsaltstate", "on");
                }
                if (message->IsSchema_P(PSTR("justsalt"), PSTR("off00000")))
                {
                        digitalWrite(pin_relay_justsalt, 1);
                        Serial.println(F("Turning OFF Justsalt"));
                        send_xpl_message("justsaltstate", "off");
                }
        }
        // show all messages
        //Serial.println(message->toString());
}
```

Pour télécharger le code rendez-vous ici : https://codebender.cc/sketch:38226

Intégration finale

Voici le résultat « final » (les câbles ne sont pas encore bien rangés…).

Share this

Arduino, Domotique, XPL

Arduino, Domotique, electrolyse, orp, ph, piscine, redox

Commentaires

- Commentaires (105)
- Trackbacks (3)
- rss
- Leave a comment
- 

  Cortexd

  septembre 2nd, 2014 à 17:32

  Return to top

  Bravo ! super article.

  Répondre
- 

  Jean-Marie

  septembre 19th, 2014 à 20:28

  Return to top

  Bravo ! ca va bien me servir, je suis en train de chercher de la doc et je n'avais rien trouvé sur la mesure du chlore même chez le pro de fourniture de piscine

  merci

  Répondre

  - 

    Clément