Rapport de projet

Rapport de projet
Analyseur d'eau de piscine

ELE3000 - Projets personnels en génie électrique

Groupe 02

Directeur de projet : Ph.D. Jean Pierre David, ing.

Hiver 2021

Département de génie électrique

École Polytechnique de Montréal

Date du laboratoire : 23 avril 2021

White Anthony 1901428

Table des matières

Table des figures

Liste des tableaux

1 Introduction

1.1 Définition du problème

Dans la vie de tous les jours, on peut facilement s'acheter un thermomètre à alcool et des papiers à pH pour analyser la qualité d'eau de sa piscine. Cependant, cela prend un temps considérable et l'on peut facilement faire des erreurs de mesures. C'est pour cela qu'il existe des moniteurs numériques pour pallier ce problème. En revanche, ils sont souvent très chers.

Prenons par exemple le pHin [7]. Le pHin est un analyseur d'eau de piscine au prix de 515.95$. Cela peut être considéré comme une grande dépense pour propriétaire de piscine. En revanche, le pHin permet de s'y connecter avec une application cellulaire par internet, il peut recommander les produits à utiliser selon le problème identifié et il permet de visualiser des graphiques des différentes mesures qu'il analyse. Ces fonctions sont intéressantes, mais non nécessaires pour quelqu'un qui possède une petite piscine personnelle.

En effet, le fait d'alerter l'utilisateur lorsqu'il y a un problème dans sa piscine suffit. De plus, les analyseurs d'eau sur le marché permettent rarement de configurer les plages de valeurs voulues pour les différentes mesures analysées. Il est certain que ces plages de valeurs sont standards, mais il se pourrait qu'un utilisateur veuille configurer sa piscine (ou bassin d'eau) différemment.

C'est pour cela que je m'engage à concevoir un analyseur de qualité d'eau à bas prix. Ce moniteur affichera plusieurs types de mesures. L'utilisateur pourra configurer des plages de valeurs pour ces mesures. Lorsque le moniteur détecte une valeur hors de ces plages, il enregistrera l'événement et l'utilisateur pourra le consulter plus tard. De plus, le moniteur fonctionnera avec une communication sans-fil, qui permet à l'utilisateur de s'y connecter à distance.

Il y a plusieurs types de mesures importantes pour l'eau d'une piscine, voici les plus importantes : le niveau de l'eau, la température, le pH, l'alcalinité et le TDS. Puisque le projet doit être faisable en trois mois, j'ai choisi d'analyser la température, le pH et le TDS.

Puisque les raisons de surveiller la température et le pH sont triviales, je ne vais pas trop m'expliquer à ce sujet. Cependant, puisque le TDS est une mesure moins connue (mais tout de même importante). Je vais approfondir sur ce sujet.

Le Total Dissolved Solids (total de solides dissous) est une mesure de concentration de particules organiques et inorganiques dans l'eau. Au fil du temps, l'eau d'une piscine accumule de la saleté qui provient du vent et des gens qui s'y baignent. De plus, l'utilisation de produit comme le chlore fait monter le TDS de l'eau. Or, il est important de garder le TDS de sa piscine en dessous de 2000 ppm. Un dépassement de cette valeur peut causer une baisse de l'efficacité du chlore [5] et peut même causer de l'irritation à la peau et les yeux des baigneurs.

1.2 Revue de la littérature et de la documentation

1.2.1 Propriétés d'une piscine à chlore

Pour faire un analyseur de piscine à chlore, il a premièrement fallu que je renseigne à ce sujet. J'ai donc cherché quel était le pH et de TDS voulu pour une piscine à chlore. De plus, j'ai recherché si le TDS était une mesure importante.

À ce qui trait au TDS, j'ai trouvé qu'un niveau trop haut de TDS [12] :

1. Réduit l'efficacité des produits comme le chlore
2. Augmente la corrosion du bassin en raison d'une conductivité plus grande
3. Cause des taches sur la surface du bassin

En consultant plusieurs sites, je n'ai pas trouvé de règlement officiel. Par contre, les articles et blogues disent que le TDS d'une piscine à chlore doit être inférieur à 1500-2000 ppm. (Ce site par exemple [13])

Pour le pH, les valeurs voulues sont entre (7.2-7.6) [6].

1.2.2 Microcontrôleur

Puisque mon projet utilise un ESP32-WROOM, il a fallu que je consulte son datasheet plusieurs fois [1].

1.2.3 Capteurs

Pour m'assurer que je pouvais utiliser un capteur pour chaque type de mesure, j'ai recherché pour des projets Arduino qui répondaient à mes besoins. Heureusement, j'ai pu trouver un projet pour chaque type de mesure. Cependant, j'ai constaté que le capteur de pH allait être le plus dur à utiliser, puisque sa calibration est compliquée et il est très sensible à son environnement.

1.2.4 Application C#

Pour réaliser mon application Windows Form, j'ai beaucoup consulté la documentation que Microsoft offre à ce sujet. Par exemple, pour configurer un évènement de réception de données[8].

1.2.5 Forums

En programmant, j'ai résolu plusieurs problèmes rencontrés en me fiant à des forums.

1.2.6 Calibration de TDS

Ceci article[4] qui m'a beaucoup aidé à calibrer mon TDS-mètre. C'est un article qui vient de l'Université de North Dakota.

1.2.7 Conception du TDS mètre

Ce tutoriel [3] de Michael Ratcliffe m'a beaucoup servi pour faire la conception de mon TDS mètre.

2 Spécifications fonctionnelles

2.1 Entrés

(a) **Température**

L'analyseur sera muni d'un capteur pour mesurer la température du bassin d'eau qu'il analyse. Ce capteur communique avec un microcontrôleur pour que celui-ci traite les données reçues adéquatement.

(b) **TDS (total dissolved solids)**

L'analyseur sera muni d'un capteur de TDS pour mesurer le taux de solides dissous dans bassin d'eau qu'il analyse. Ce capteur communiquera avec un microcontrôleur pour que celui-ci traite les données reçues adéquatement.

(c) **pH**

L'analyseur sera muni d'un capteur de ph pour mesurer l'acidité du bassin d'eau qu'il analyse. Ce capteur communiquera avec un microcontrôleur pour que celui-ci traite les données reçues adéquatement.

(d) **Plages de valeurs**

L'utilisateur peut entrer des seuils, qui déterminent les plages de valeurs voulues pour les mesures. Ces commandes sont interprétées par l'analyseur et il change son mode de fonctionnement en conséquence.

2.2 Sorties

(a) **Température**

L'analyseur a en sortie la température qu'il mesure (en °C) chaque heure.

(b) **TDS**

L'analyseur a en sortie le TDS qu'il mesure (en ppm) chaque heure.

(c) **pH**

L'analyseur a en sortie le pH qu'il mesure chaque heure.

(d) **Répertoire d'événements non désirés**

L'analyseur a en sortie un répertoire des évènements non désirés.

(e) **Application Windows**

L'analyseur est capable de se connecter par Bluetooth à un ordinateur ayant Windows. Les diverses données affichées sur l'écran LCD seront aussi affichées sur l'application. Les évènements non désirés enregistrés par le microcontrôleur y seront aussi affichés (avec la raison et la date).

2.3 Fonctions

(a) **Connexion Bluetooth**

L'analyseur peut se connecter par Bluetooth à un ordinateur ayant Windows comme système d'exploitation. Une application Windows "PoolAnalyserPro" peut envoyer des commandes à l'analyseur et recevoir des données de celui-ci. La communication sans-fil doit être réussie 19 fois sur 20, et ce, à une distance de 100m

(b) **Conversion tension-température**

L'analyseur peut convertir une tension analogique (0-3.3V) provenant du capteur en valeur (°C). Cette conversion doit respecter les contraintes suivantes :

- Mesurer des températures allant de 5°C à 50°C.
- Avoir une exactitude des mesures de 5% de la réalité et des fluctuations moindres de 5%.
- Lorsque l'environnement est stable, les fluctuations de mesures doivent être d'au plus 3%

(c) **Conversion tension-TDS**

L'analyseur peut convertir une tension analogique (0-3.3V) provenant du capteur en valeur (ppm). La conversion doit respecter les contraintes suivantes :

- Mesurer des TDS allant de 0 à 2650 ppm.
- Avoir une exactitude des mesures de 5% de la réalité.
- Lorsque l'environnement est stable, les fluctuations de mesures doivent être d'au plus 3%

(d) **Conversion tension-pH**

L'analyseur peut convertir une tension analogique (0-3.3V) provenant du capteur en valeur (pH). La conversion doit respecter les contraintes suivantes :

- Mesurer des pH allant de 0 à 14.
- Avoir une exactitude des mesures de 5% de la réalité.
- Lorsque l'environnement est stable, les fluctuations de mesures doivent être d'au plus 3%

(e) **Affichage de température**

Lorsqu'une température sort du système, l'analyseur l'affiche. L'affichage doit être réussi 19 fois sur 20.

(f) **Affichage de TDS**

Lorsqu'un TDS sort du système, l'analyseur l'affiche. L'affichage doit être réussi 19 fois sur 20.

(g) **Affichage de pH**

Lorsqu'un pH sort du système, l'analyseur l'affiche. L'affichage doit être réussi 19 fois sur 20.

(h) **Affichage des évènements non désirés**

Lorsqu'un évènement non désiré est stocké dans le système, l'analyseur permet à l'utilisateur de les consulter dans un répertoire (style logs). L'enregistrement doit être réussi au moins 19 fois sur 20.

(i) **Suppression des évènements non désirés**

Lorsque l'utilisateur a fini de consulter son répertoire d'évènements non désirés, il peut les supprimer. La suppression doit être réussie au moins 19 fois sur 20.

2.4 Facteurs humains

(a) **Plages des valeurs**

L'utilisateur peut prescrire les plages de valeurs voulues pour chaque type de mesure. Par exemple, si l'utilisateur met une plage de valeurs entre 20 et 32°C pour la température et que la température mesure sort de cette plage, l'analyseur enregistrera l'événement indésirable. Ces valeurs seront limitées à ceux compatibles avec les capteurs.

(b) **Application Windows : connexion Bluetooth**

À L'aide de l'application Windows, l'utilisateur pourra configurer la connexion entre l'analyseur et son ordinateur. Il pourra donc choisir le port de communication voulu et le baud rate.

2.5 Réaction aux erreurs

(a) **Communication avec l'utilisateur**

Lorsque l'usager effectue une erreur (valeur entrée non possible, connexion sans-fil interrompue), l'utilisateur en est informé par l'application.

(b) **Erreur de communication sans-fil**

Lorsqu'une erreur de communication sans-fil arrive, l'analyseur ignorera les données reçues par la communication. Si aucune nouvelle donnée n'arrive pour trop longtemps, un message d'erreur sera communiqué à l'utilisateur.

2.6 Diagramme de flux de données

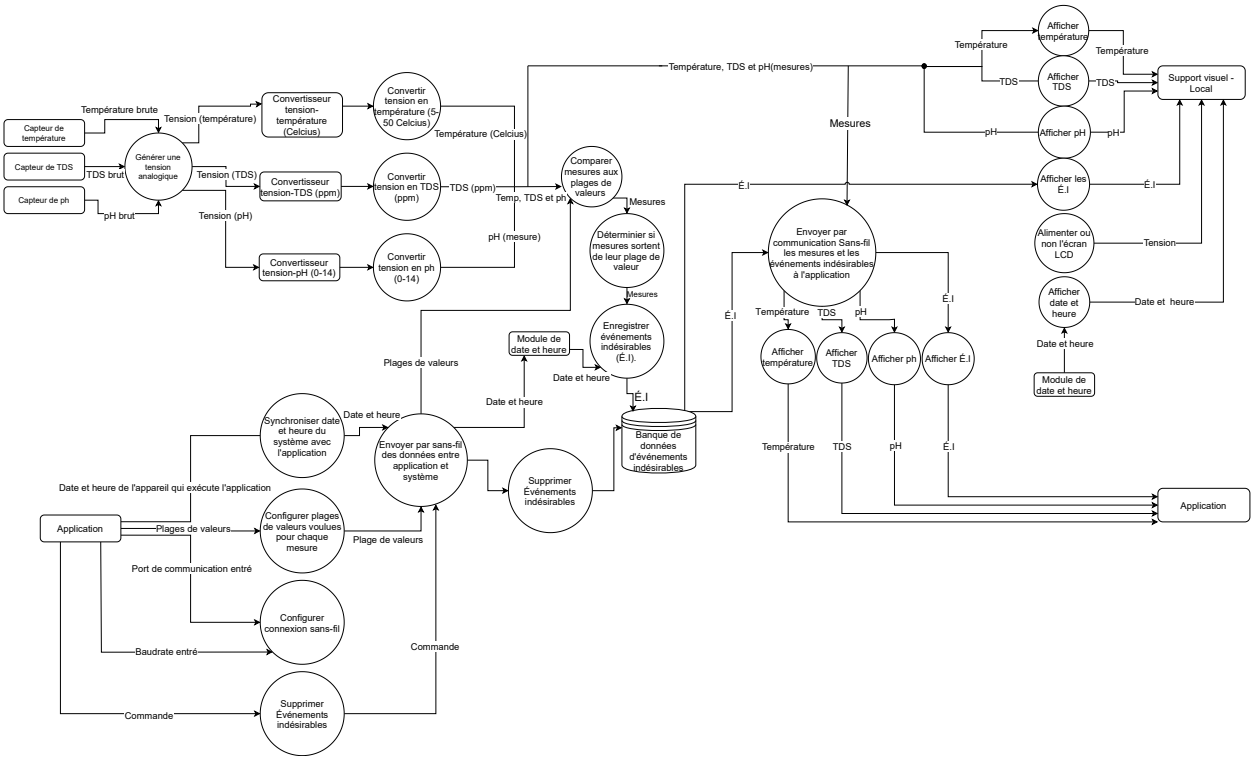

FIGURE 1 – Diagramme de flux de données

2.6.1 Dictionnaire pour le projet

1. **Mesure** : Peut être l'un des trois types de mesures captés par le prototype (température, TDS ou pH).
2. **Plage de valeur** : Intervalle de valeurs voulues pour un type de mesure. Une plage de valeur est composée d'un minimum et d'un maximum et d'un type de mesure.
3. **Évènement indésirable (É.I)** : Évènement qui survient lorsqu'une mesure captée sort de la plage de valeur pour cette mesure. Un É.I est composé d'un type de mesure, d'une raison (trop haut ou trop bas) et d'une date.

3 Design préliminaire

3.1 Exploration des approches de résolution des problèmes

3.1.1 Système central

L'analyseur doit être en mesure de capter analyser de plusieurs tensions. De plus, Il doit afficher et communiquer des informations. Il est donc clair qu'il faut un microcontrôleur pour effectuer ces tâches.

3.1.2 Température

Pour mesurer la température, il faut un capteur résistant à l'eau et qui peut mesurer assez précisément les températures typiques de l'eau.

3.1.3 TDS

Pour mesurer le TDS, il faut un capteur résistant à l'eau et qui peut mesurer assez précisément les TDS typiques de l'eau d'une piscine.

3.1.4 pH

Pour mesurer le pH, il faut un capteur résistant à l'eau et qui peut mesurer assez précisément les pH typiques de l'eau.

3.1.5 Date et heure

Pour permettre à l'analyser d'associer une date à un événement indésirable, il doit être capable de prendre une date et de compter précisément à partir de celle-ci. On peut donc utiliser un compteur de millisecondes (fonction millis() dans Arduino), le RTC interne du microcontrôleur ou un RTC externe.

3.1.6 Affichage local

Pour afficher localement les données recueillies par l'analyseur, je dois utiliser un support visuel. À ma connaissance, les meilleures solutions sont soit un écran alphanumérique à sept segments, ou soit un petit écran LCD.

3.1.7 Application logicielle

L'analyseur utilisera une application pour deux raisons : afficher les mesures captées à distance et permettre à l'usager de configurer l'analyseur. Il existe plusieurs formes d'applications. Il faut donc que je choisisse la plateforme qui exécutera mon application ainsi que le langage de programmation que j'utiliserai.

3.2 Étude des alternatives et justification de la solution retenue

3.2.1 Choix de microcontrôleur

Selon mon expérience, j'ai fait le cours ELE3312, où j'ai utilisé un STM32F446RE. Cependant, ce microcontrôleur ne peut pas communiquer par Bluetooth par lui-même. Il faut donc lui connecter un module transmetteur/receveur Bluetooth pour qu'il réponde au besoin du projet. Cela m'intéresse peu puisque je veux minimiser le coût total du produit et donc son nombre de composantes.

J'ai donc continué mes recherches et j'ai trouvé qu'il existait une gamme de microcontrôleurs STM32 qui avait des fonctionnalités Bluetooth intégrées. Cependant, ces microcontrôleurs coûtent très cher≈ 100$. En effet, ils sont très performants. Même trop performant pour les simples besoins du projet.

Après plus de recherches, j'ai découvert les microcontrôleurs ESP32. Ces microcontrôleurs se spécialisent dans la communication sans-fil (WiFi ou Bluetooth) et ils sont reconnus pour leur faible prix. Malgré le fait que je ne suis pas très familier avec ces microcontrôleurs, j'ai décidé de m'y lancer. Pour faciliter la programmation, j'ai décidé d'utiliser Arduino, qui me permet de faire abstraction des composantes de bas niveau du microcontrôleur. J'ai pu acheter deux ESP32 WROOM pour 26$.

3.2.2 Choix de capteur de température

Après quelques recherches, j'ai trouvé que le capteur de température DS18B20. Il répond à toutes les contraintes du projet et j'ai pu m'en procurer un au prix de 20$.

3.2.3 Choix de capteur de TDS

Comme pour le capteur de température, j'ai trouvé et considéré utiliser un capteur de TDS SEN0244 pour mon projet. Cependant, j'ai réalisé qu'il ne pouvait pas mesurer une valeur de TDS plus grande que 1000 ppm. Comme il a été mentionné, le TDS maximal qu'une piscine doit avoir est de 2000 ppm. Ce capteur est donc insuffisant pour les contraintes du projet.

Puisque je n'ai pas trouvé d'autre capteur de TDS fait pour un microcontrôleur répondant aux besoins du projet, j'ai décidé d'en fabriquer un artisanalement. Ainsi, je pourrai ajuster les performances du capteur selon les contraintes du projet.

3.2.4 Choix de capteur de pH

Le choix de capteur de pH n'a pas été compliqué, puisque c'est le seul que j'ai trouvé en ligne et que je pouvais recevoir dans un délai raisonnable. J'ai donc décidé d'utiliser un capteur E-201-C. Les spécifications de la sonde sont les suivantes. [10]

General Specifications:

Input supply voltage: 5V
Working current: 5 – 10mA
Detection concentration range: PH 0 – 14
Detection range of temperature: 0 – 80 degC
Response Time:: ≤ 5S
Stability Time: ≤ 60S
Output: Analog
Power Consumption: ≤ 0.5W
Working Temperature: -10 to +50 deg C
Working Humidity: 95%RH (nominal humidity 65%RH)
Weight: 25g
PCB Dimension: 42mm x 32mm x 20mm

FIGURE 2 – Spécifications générales pour le capteur à pH E-201-C

On peut voir que l'intervalle de mesure pour le pH respecte le cahier des charges (0-14), ainsi que la température de fonctionnement (-10 - 50°C).

3.2.5 Choix de module RTC

D'un, il serait possible de programmer une date dans le microcontrôleur et compter à partir de celle-ci avec un compteur à interruption aux millisecondes. Avec Arduino, cela se fait facilement avec la fonction millis(), qui retourne le temps en millisecondes depuis que le programme s'exécute. Cependant, cette méthode n'est pas assez précise [9] et l'heure finale obtenue après un certain temps serait trop différente de la réalité. Cette solution n'est donc pas retenue.

De deux, les forums semblent dire que le module RTC interne du ESP32 n'est pas très précis. Cette solution ne m'intéresse donc pas puisque mon projet requiert de garder une heure assez précise sur un grand laps de temps.

Finalement, un module RTC externe permet de garder en mémoire une date et une heure et de compter à partir de celle-ci de façon très précise [9]. De plus, un RTC permet de conserver l'heure comptée même si le microcontrôleur ne l'alimente plus. Cela est intéressant puisque ça éviterait de perdre l'heure d' enregistrer lorsque l'alimentation de l'analyseur est coupée. C'est donc cette solution qui est retenue. Plus précisément, j'utilise un module RTC DS1307 que j'ai déjà en ma possession et qui m'a coûté 3,20$.

3.2.6 Choix de support visuel

D'un, un écran alphanumérique à sept segments pourrait être un bon choix, puisque c'est facile à programmer. Cependant, sa simplicité vient limiter mes options d'affichage. Puisque mon projet demande d'afficher beaucoup d'informations en même temps, ce n'est pas la solution qui est retenue.

De deux, un petit écran LCD est plus compliqué à programmer. Cependant, cette solution offre une liberté d'affichage quasi infinie. Cela m'intéresse beaucoup. Premièrement puisque j'ai déjà de l'expérience avec ce genre d'écran. Deuxièmement, cette solution me permet de rendre mon produit plus attrayant avec de meilleurs effets visuels. Comme il est mentionné ci-haut, le projet demande l'affichage de beaucoup d'information. Donc, c'est la solution qui est retenue. Plus précisément, j'utilise un écran TFT ILI9341 qui m'a coûté 21,09$.

Choix d'application logiciel

Mon choix initial était de faire une interface graphique avec Python. Cependant, j'ai vite réalisé que c'était très compliqué, surtout avec un ordinateur ayant Windows.

Pour un ordinateur ayant Windows, j'ai trouvé que la solution la plus simple et efficace est d'utilité une application Windows Form (C) programmée en utilisant Visual Studio. Cette plateforme offre plusieurs librairies conçues pour interagir avec les périphériques d'un ordinateur Windows (ex :[8]).

3.3 Description des modules et architecture générale du système

3.3.1 Prise de Mesure

1. **En-tête**
   - Titre : Prise de mesure
   - Version/date : 1.0/21 février 2021
2. **Fonctions réalisées**
   - Partie matérielle :
     - ☐ Capter la température brute de l'environnement (vibrations de molécules d'eau) et générer une tension adéquate
     - ☐ Capter le TDS brut de l'environnement (conductivité de l'eau) et générer une tension adéquate
     - ☐ Capter le ph brut de l'environnement (concentration de H^+) et générer une tension adéquate
   - Partie logicielle :
     - ☐ Convertir tension analogique en température (°C)
     - ☐ Convertir tension analogique en TDS (ppm)
     - ☐ Convertir tension analogique en pH
3. **Information retenue**
   - Aucune
4. **Interface**
   - Interface matérielle :
     - ☐ Sondes connectées à des capteurs qui génèrent des tensions analogiques
     - ☐ Envoie les tensions analogiques au module de prise de mesure logiciel
   - Interface logicielle :
     - ☐ Reçoit les tensions de l'interface matérielle
     - ☐ Envoie les mesures converties au comparateur et aux modules d'affichage (LCD et logiciel)
   - Entrées
     - ☐ Température brute (excitation thermique des particules d'eau)
     - ☐ TDS brut (conductivité de l'eau)
     - ☐ pH brut (concentration de particules H^+)
   - Sorties
     - ☐ Température (°C)
     - ☐ TDS (ppm)
     - ☐ pH (en échelle logarithmique 0-14)

5. **Remarque / Restriction / Procédure**
   - La sonde pour le TDS devra être conçue artisanalement, puisqu'il n'existe aucun capteur pour micro-contrôleur permettant la prise de mesure de plus de 1000 ppm
   - À faire fonctionner en premier puisqu'il s'agit du module le plus important du projet
6. **Procédure de tests**
   - Partie matérielle :
     - ☐ Mesurer à l'aide d'un multimètre si les capteurs génèrent réellement des tensions analogiques
   - Partie logicielle :
     - ☐ Vérifier que les mesures converties représentent bel et bien le milieu que l'on mesure :
       - — Pour la température, comparer à la valeur d'un thermomètre à alcool
       - — Pour le TDS, préparer des solutions avec des concentrations de solides prédéfinis puis comparer avec les valeurs obtenues
       - — Pour le ph, préparer des solutions avec des ph prédéfinis puis comparer avec les valeurs obtenues

3.3.2 Comparateur

1. **En-tête**
   - Titre : Comparateur
   - Version/date : 1.0/21 février 2021
2. **Fonctions réalisées**
   - Comparer les mesures aux plages de valeurs
   - Déterminer si les mesures sortent de leur plage de valeur
   - Enregistrer la date, l'heure et la valeur de la mesure si elle sort de la plage de valeur. En d'autres mots, enregistrer l'événement indésirable (É.I)
   - À partir d'une date et heure donnée, être capable mémoriser la date et l'heure actuelle.
3. **Information retenue**
   - Banque d'événements indésirables
   - Date et heure actuelle
4. **Interface**
   - Reçoit les mesures du module de prise de mesure logiciel
   - Envoie les événements indésirables aux supports visuels (LCD et logiciel)
   - Entrées
     - ☐ Température (°C)
     - ☐ TDS (ppm)
     - ☐ pH (en échelle logarithmique 0-14)
   - Sorties
     - ☐ Événements indésirables (É.I)
5. **Remarque / Restriction / Procédure**
   - Devra être conçue après la prise de mesure
6. **Procédure de tests**
   - Contrôler l'environnement analysé pour que les mesures prises sortent des plages de mesures (ex : chauffer un bassin d'eau). Ensuite, s'assurer que le module enregistre bien l'événement indésirable.
   - Vérifier que la date et heure mémorisée concorde avec la réalité

3.3.3 Application logicielle - Interface usager

1. **En-tête**
   - Titre : Application logicielle - interface usager
   - Version/date : 1.0/21 février 2021
2. **Fonctions réalisées**
   - Permettre à l'utilisateur de configurer sa connexion Bluetooth
   - Permettre à l'utilisateur de configurer des plages de valeurs pour les mesures
   - Permettre à l'utilisateur de supprimer les événements indésirables
3. **Information retenue**
   - Plage de valeur pour chaque mesure
   - Information pour la connexion Bluetooth (Port et baud rate)
4. **Interface**
   - Envoie au comparateur les plages de valeurs pour chaque mesure par communication Bluetooth
   - Entrées
     - ☐ Commandes de l'utilisateur
       - — Plages de valeurs pour les mesures (doubles)
       - — Port de communication
       - — Baudrate (int prédéfini)
   - Sorties
     - ☐ Plages de valeurs pour les mesures
5. **Remarque / Restriction / Procédure**
   - Devra être conçu en dernier puisqu'il est possible de tout faire fonctionner sans un interface usager complet
6. **Procédure de tests**
   - Utiliser l'interface usager et s'assurer que les fonctions réalisées se font comme prévu

3.3.4 Application Logicielle - affichage

1. **En-tête**
   - Titre : Application Logicielle - affichage
   - Version/date : 1.0/21 février 2021
2. **Fonctions réalisées**
   - Afficher mesures de température (à chaque heure)
   - Afficher mesures de TDS (à chaque heure)
   - Afficher mesures de ph (à chaque heure)
   - Afficher les événements indésirables
3. **Information retenue**
   - Aucune
4. **Interface**
   - Reçoit les mesures du module de prise de mesure logiciel
   - Reçoit les événements indésirables du comparateur
   - Affiche les trois mesures de la prise de mesure
   - Comporte des boutons et boîtes de textes pour les fonctionnalités de l'interface usager.
   - Comporte un tableau pour naviguer les événements indésirables
   - Entrées
     - ☐ Température (°C)
     - ☐ TDS (ppm)
     - ☐ pH (en échelle logarithmique 0-14)
   - Sorties
     - ☐ Affichage de température
     - ☐ Affichage de TDS
     - ☐ Affichage de pH
     - ☐ Tableau d'É.I
5. **Remarque / Restriction / Procédure**
   - N'a pas besoin d'être conçu rapidement puisque ses fonctionnalités peuvent être testées sans un support visuel complet
6. **Procédure de tests**
   - S'assurer que le module affiche bien les diverses mesures
   - S'assurer que le module reçoit bien les informations par Bluetooth

3.3.5 Affichage local

1. **En-tête**
   - Titre : Affichage local
   - Version/date : 1.0/21 février 2021
2. **Fonctions réalisées**
   - Afficher mesures de température (à chaque heure)
   - Afficher mesures de TDS (à chaque heure)
   - Afficher mesures de ph (à chaque heure)
   - Afficher la date et l'heure
   - Afficher les événements indésirables
   - Permettre à l'utilisateur d'allumer l'écran (on/off)
   - Permettre à l'utilisateur d'allumer l'écran pendant 10 secondes (pushbutton)
3. **Information retenue**
   - Aucune
4. **Interface**
   - Reçoit les mesures du module de prise de mesure logiciel
   - Reçoit les événements indésirables du comparateur
   - Affichera trois gradateurs qui représentent les trois mesures du module de prise de mesure
   - Affiche les É.I un à la fois (en cycle)
   - Entrées
     - ☐ Affichage de température
     - ☐ Affichage de TDS
     - ☐ Affichage de pH
     - ☐ Événements indésirables
     - ☐ Les minimums et les maximums des plages de valeurs
   - Sorties
     - ☐ Affichage de température
     - ☐ Affichage de TDS
     - ☐ Affichage de pH
     - ☐ Affichage d'É.I
     - ☐ affichage des min/max des plages de valeurs
5. **Remarque / Restriction / Procédure**
   - Devra être conçu au plus vite pour permettre de bien voir le résultat des autres modules
6. **Procédure de tests**
   - S'assurer que le module affiche bien les diverses mesures, l'heure et les É.I

3.3.6 Communication sans-fil application-système

1. **En-tête**
   - Titre : Communication sans-fil application-système
   - Version/date : 1.0/21 février 2021
2. **Fonctions réalisées**
   - Envoyer par communication sans-fils les plages de valeurs des mesures
   - Envoyer une commande pour que l'analyseur supprime ses événements indésirables
   - Recevoir les É.I de l'analyseur
3. **Information retenue**
   - Aucune
4. **Interface**
   - Envoie par Bluetooth les plages de valeurs des mesures
   - Envoie par Bluetooth une commande pour que l'analyseur supprime les événements indésirables stockés
   - Reçoit les événements indésirables du comparateur
   - Entrées
     - ☐ Événements indésirables
   - Sorties
     - ☐ Plages de valeurs
5. **Remarque / Restriction / Procédure**
   - Faire à la fin
6. **Procédure de tests**
   - S'assurer que l'application reçoit bien des événements indésirables (sur l'affichage logiciel)
   - S'assurer que le système reçoit bien les plages de valeurs (sur l'affichage LCD)
   - S'assurer que l'analyseur supprime ses événements indésirables lorsque la commande respective est envoyée

3.3.7 Communication sans-fil système-application

1. **En-tête**
   - Titre : Communication sans-fil système-application
   - Version/date : 1.0/21 février 2021
2. **Fonctions réalisées**
   - Envoyer par communication sans-fil les mesures
   - Envoyer par communication sans-fil les événements indésirables
   - Recevoir par communication sans-fil une commande de l'application pour supprimer les É.I
3. **Information retenue**
   - Aucune
4. **Interface**
   - Envoie par Bluetooth les trois mesures et les événements indésirables
   - Reçoit une commande de l'application pour supprimer les événements indésirables
   - Entrées
     - ☐ Commande pour supprimer les É.I
   - Sorties
     - ☐ Température (°C)
     - ☐ TDS (ppm)
     - ☐ pH (en échelle logarithmique 0-14)
5. **Remarque / Restriction / Procédure**
   - Faire à la fin
6. **Procédure de tests**
   - S'assurer que le module affiche bien les diverses mesures
   - S'assurer que le l'analyseur supprime les É.I lorsqu'il reçoit la commande

3.3.8 Architecture modulaire du système

En regroupant les parties connexes du DFD, on obtient l'architecture modulaire du système :

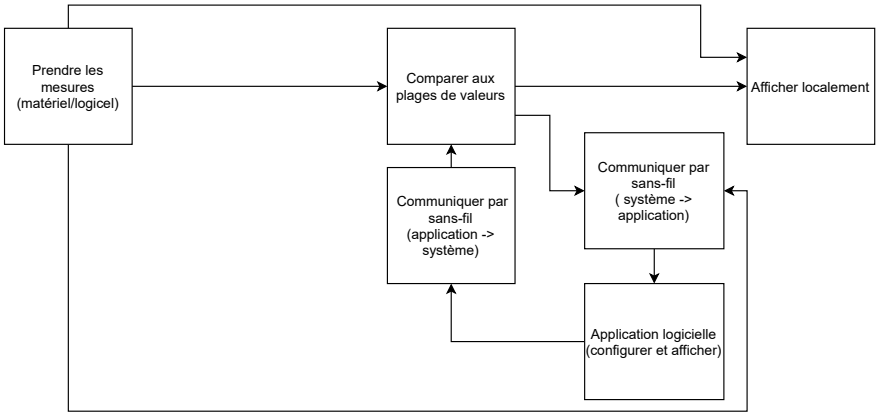

FIGURE 3 – Architecture modulaire du système

En changeant les modules pour les composantes réelles choisies dans la section 3.1, on obtient le schéma suivant :

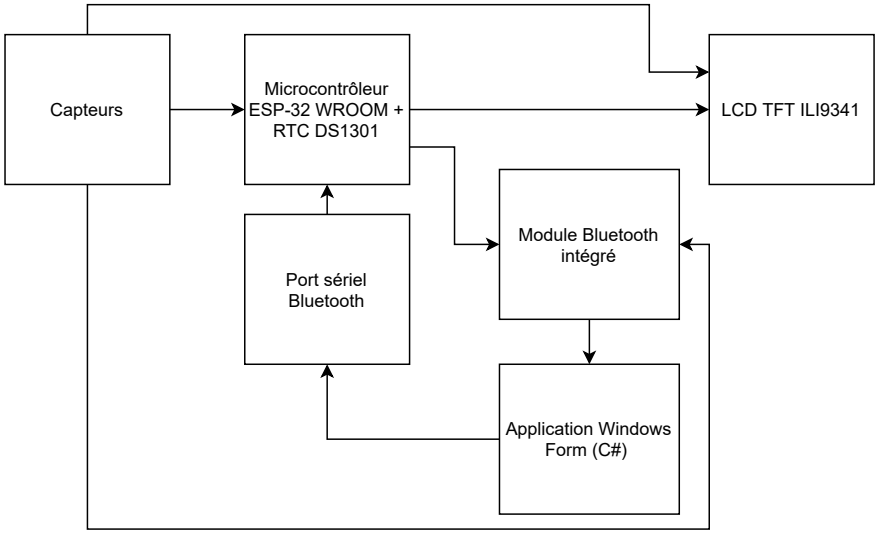

FIGURE 4 – Choix des composantes pour chaque module

Maintenant que les modules ont été définis et que les composantes ont été choisies, on peut élaborer le design détaillé du prototype.

4 Design détaillé

4.1 Schéma général

4.2 Capteur de température

Pour mesurer la température d'une piscine, le Pool Analyser Pro utilise un capteur de température DS18B20.

Ce capteur utilise un 1-Wire Bus System pour communiquer avec le microcontrôleur. La librairie One-Wire et DallasTemperature de Arduino a été utilisée pour accélérer le prototypage. Ce capteur n'a pas besoin d'être calibré pour fonctionner

Le code spécifique à ce capteur est ci-dessous [2] :

```
#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS 27

float temperature;

OneWire ds(ONE_WIRE_BUS);

DallasTemperature sensors(&ds)

void setup(void)
{
 // Start up the library
 sensors.begin();
}

void loop(void)
{
 // call sensors.requestTemperatures() to issue a global temperature
 // request to all devices on the bus
/*********************************************************************/
 sensors.requestTemperatures(); // Send the command to get temperature readings
/*********************************************************************/
 temperature = sensors.getTempCByIndex(0));
  delay(1000);
}
```

La variable "temperature" représente donc la température mesurée par le capteur et est ensuite traitée par l'analyseur.

4.3 Capteur de TDS

4.3.1 Mise en contexte

Le capteur de TDS est plutôt un capteur de conductivité. En effet, en mesurant la conductivité d'un liquide (EC) on peut trouver son TDS respectif. Cependant, ce n'est pas aussi simple qu'un facteur de conversion. Il faut plutôt se fier au tableau de conversions ci-dessous [4]. Il faut noter que la distance entre les connecteurs A est d'environ 1 cm, on peut donc utiliser directement les valeurs d'EC comme conductance.

EC (μS/cm)	TDS approx. (ppm)	fact. de conv. (γ)
100	50	0.50
500	300	0.60
1000	650	0.65
1500	1050	0.70
2000	1450	0.72
2500	1850	0.74
3000	2250	0.75
3500	2650	0.76
4000	3500	0.77

TABLE 1 – Tables des taux de conversion EC ⇒ TDS selon l'EC

Le capteur est composé d'un fil Type A à deux dents, d'un connecteur femelle type 8, de deux fils "jumpers" et d'une résistance R_{eau}. Le circuit est donc comme suit :

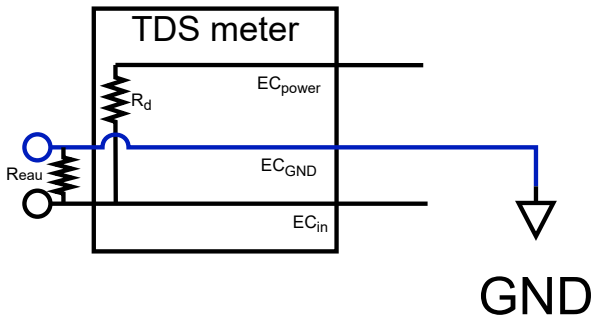

FIGURE 5 – Circuit pour le TDS-mètre

Pour trouver la conductivité de l'eau, il faut premièrement trouver la résistance de celle-ci. Pour faire cela, on injecte une tension dans l'entrée EC_power. Ensuite, TDS-mètre effectue une division de potentiel avec la résistance de 1KΩ et la résistance de l'eau.

Il faut noter qu'on ne peut pas utiliser une tension continue pour mesurer cette résistance. Le cas échéant, la tension polariserait les molécules d'eau et la résistance de l'eau augmenterait considérablement. Il faut donc utiliser une tension alternative. En d'autres mots, il faut injecter une tension continue, mais très brève.

Lorsqu'on veut effectuer une mesure, on envoie donc une tension de 3.3V à l'entrée EC_power à l'aide d'une pin GPIO en mode sortie. Après un bref instant, on mesure la tension à l'entrée EC_in à l'aide d'une

pin en mode ADC entrée. La résistance de l'eau se calcule alors comme suit :

$$R_{eau} = \frac{V_{in} \cdot R_d}{V_{power} - V_{in}} - R_i \tag{1}$$

R_i est la résistance d'entrée de la pin d'entrée. Expérimentalement, j'ai connecté une résistance de 100Ω sur la même pin et à la masse. J'ai ensuite configuré la pin mode sortie et toujours active à 3.178V (mesuré lorsque la pin était flottante. J'ai donc mesuré la tension à la pin (2.545V). En calculant à l'aide de la formule de division de potentiel, j'ai trouvé :

$$R_i = \left(\frac{3.178V}{2.545V} \cdot 100\ \Omega\right) = 24.87\ \Omega$$

Cependant, on ne peut pas convertir directement cette résistance en conductivité. En effet, la nature même du TDS mètre fait en sorte que la conductivité obtenue sera à une constante près de la réalité.

C'est ce que Micheal Ratcliffe a considéré dans son design [3]. La formule pour trouver la conductivité devient alors :

$$EC = \frac{1000}{R_{eau} \cdot K} \tag{2}$$

Puisque l'EC varie selon la température, il faut la compenser. Pour faire cela, on ramène l'EC à une température de 25°C :

$$EC_{25} = \frac{EC}{(1 + 0.019 \cdot (Temperature - 25.0))} \tag{3}$$

Le facteur de compensation 0.019 est propre à l'eau et est une valeur empirique. On peut maintenant calculer le TDS du liquide :

$$\text{TDS} = EC_{25} \cdot (1000\gamma) \tag{4}$$

Les deux facteurs de 1000 facilitent le calcul effectué par le microcontrôleur, et rend l'EC en μS.

4.3.2 Choix de R_d

R_d effectue une division de potentiel avec R_{eau} à partir d'une tension de 3.3V. Cette tension est ensuite mesurée avec un convertisseur analogique 0-3.3V avec une résolution de 4096. Pour avoir un capteur optimal, il faut que l'intervalle des tensions mesurées soit le plus proche de 0-3.3V.

J'ai décidé que la valeur maximale de TDS que je voulais mesurer était de 2650 ppm. D'un parce que c'est assez supérieur à 2000 ppm (maximum recommandé pour une piscine à chlore. De deux, c'est une valeur que je peux utiliser pour calibrer mon capteur en suivant une procédure prédéfinie [4].

Expérimentalement, j'ai trouvé que la résistance de l'eau à un TDS d'environ 2650 ppm est de 95Ω. Cela diffère du calcul de Micheal et pose problème. En effet, je ne peux pas utiliser une résistance R_d dans les alentours de 95Ω, parce que la résistance totale du circuit serait trop petite et le courant généré pourrait

R_d	V_{dmin}	V_{dmax}	Range
10	2.99	3.28	0.30
20	2.73	3.27	0.54
47	2.21	3.23	1.02
220	1.00	3.00	2.00
470	0.55	2.72	2.16
1000	0.29	2.26	1.98

TABLE 2 – Plages de valeurs de V_d selon R_d

endommager le microcontrôleur. De plus, j'ai trouvé (expérimentalement) que la résistance de l'eau pour un TDS de 265 (valeur très petite pour une piscine à chlore) est de 2184Ω. J'ai donc calculé les plages de tensions que j'allais lire sur la pin EC_{in} selon la valeur de R_d avec la formule suivante :

$$V_d = 3.3 \text{ V} \frac{R_{eau}}{R_{eau} + R_d} \tag{5}$$

On peut voir que la résistance de 470Ω offre la plus grande plage de valeur. Cependant, elle impose un V_{dmin} de 0.55 V, ce qui se rapproche trop du bruit. La valeur la plus intéressante est donc 220Ω, qui offre une plage de valeur de 2 V et un V_{dmin} 1 V.

4.3.3 Calibration

Pour calibrer l'appareil, il faut calculer la conductivité mesurée pour un liquide dont on connaît la conductivité réelle. Ensuite, un trouve un facteur K (eq2) qui ramène la conductivité mesurée à la conductivité réelle. En effectuant une série de tests avec une solution à 2650 ppm (EC = 3500μS). J'ai trouvé une valeur de $K = 10.5$.

4.3.4 Fonctionnement

Voici le code simplifié qui mesure le TDS d'un liquide :

```
void setup(void)
{
    pinMode(EC_pin, INPUT);
    pinMode(EC_power, OUTPUT)
}

void loop(void)
{
    if(timeToMeausureTds)
    {
        digitalWrite(EC_power, HIGH);
        float rawVoltage = analogRead(EC_pin);
        digitalWrite(EC_power, LOW);

        float voltage = rawVoltageToVoltage(EC_pin);
        float R_eau = voltageToResistance(voltage);
        float ec = resistanceToEc(R_eau);
        float tds = ecToTds(ec);

    }
}
```

Dans le vrai code, l'analyseur prend une mesure au 5 secondes, sinon il y a des risques d'endommager le capteur. De plus, le code vient en grande partie de Michael Ratcliffe [3].

En pratique, on remarque que valeurs mesurées fluctuent considérablement d'un à l'autre. Pour pallier à cela, l'analyseur prend 10 mesures, enlève les 3 plus grandes et les 3 plus petites et calcule la moyenne des quatre restantes. Je me suis inspiré du code pour le capteur de pH pour faire cela.

4.4 Capteur de pH

4.4.1 Mise en contexte

Le capteur de pH est le est composé d'un amplificateur E-201-C et d'une sonde à pH. Le bout de la sonde est fait un électrode en vitre avec une couche sensible aux atomes H^+. Lorsque la sonde est submergée dans un liquide, la tension produite par celle-ci est comparée à la tension produite par une solution de KCl, qui est neutre.

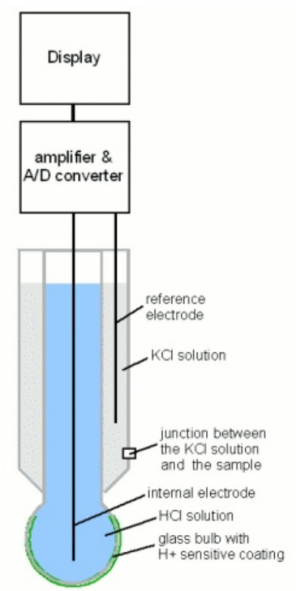

FIGURE 6 – Schéma de l'électrode à pH [10]

L'amplificateur est composé d'un connecteur BNC pour y brancher la sonde. De plus, il possède une pin pour l'alimentation (5V) et une pin P_o pour la sortie du pH. Le schéma de l'amplificateur se trouve ci-dessous.

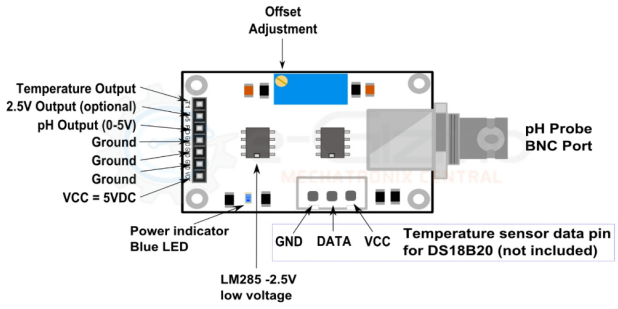

FIGURE 7 – PCB de l'amplificateur à pH [10]

Il est à noter que ce schéma est pour la version 1.0 du board. En faisant quelques recherches, il me semble

que j'ai une autre version. Malheureusement, je n'ai pas pu trouver de datasheet pour cette version. En fait, c'est ce document qui m'a été transmis par le magasin auquel j'ai acheté le capteur (ABRA).

En termes de différences, la 2e version ne possède pas de pin pour un DS18B20 et possède un 2e potentiomètre pour configurer un seuil de pH. Si le pH de la solution mesurée dépasse ce seuil, une DEL rouge allume et un signal d_o devient haut. Cette fonctionnalité n'est pas utilisée dans mon prototype.

Puisque la tension d'alimentation est de 5V, il faut s'assurer que la tension de sortie (P_o) ne dépasse pas 3.3 V. Selon la figure ci-dessous, cela ne pose pas problème puisque la tension de sortie ne dépasse pas 3.071V (je ne compte pas mesurer un pH en dessous de 4). Il serait tout de même intéressant d'intégrer un circuit de protection dans un produit commercialisé.

pH Value	Output
4	3.071
7	2.535
10	2.066

FIGURE 8 – Tensions émises selon des pH de référence [10]

4.4.2 Calibration

Pour calibrer le pH-mètre, il faut premièrement régler son potentiomètre. Ce potentiomètre détermine un offset de tension en sortie. Sans cet offset, le pH-mètre émet une tension négative, nulle ou positive lorsque la solution est basique, neutre et acide respectivement. Cela pose problème pour un microcontrôleur puisqu'il ne peut pas mesurer une tension négative.

Il faut donc que la tension émise pour un pH de 7 soit de 2.535 V. Puisqu'un pH de 7 équivaut à une tension nulle (avant l'offset), on peut simplement régler le potentiomètre en court-circuitant les bornes du connecteur BNC de l'amplificateur. Ensuite, on mesure la tension P_o et on règle le potentiomètre jusqu'à ce que la tension soit de 2.535V.

Ensuite, il faut trouver la fonction qui transforme la valeur lue (analogRead) en pH. Selon le document technique [10], la relation est la suivante :

$$pH = m \cdot analogRead + b \tag{6}$$

Il faut donc trouver les paramètres m et b en mesurant les valeurs de analogRead pour des solutions dont on connait le pH. Pour préparer des solutions avec des pH prédéfini, j'ai utilisé des poudres à pH. Avec des solutions ayant des pH de 6.86 et 9.18, j'ai obtenu les résultats ci-dessous.

pH	analogRead
6.86	3550
9.18	2900

TABLE 3 – analogRead selon le pH

On peut donc trouver l'équation qui relie l'analogRead et le pH de la solution.

$$pH = -0.00357 \cdot analogRead + 19.53077 \tag{7}$$

4.4.3 Fonctionnement

Lorsqu'on veut mesurer le pH d'une solution, on submerge la sonde du pH-mètre dans le liquide. Ensuite, la pin P_o du pH-mètre étant connecté à la pin G35 du ESP32, le microcontrôleur lit la tension analogique et la convertie en valeur numérique (analogRead).

Selon le document technique, il faut prendre 10 mesures, enlever les deux plus grandes et plus petites valeurs et calculer la moyenne de l'échantillon pour obtenir une mesure de pH. Le code simplifié pour faire cela est ci-dessous [10].

```
const int analogInPin = A0;
int sensorValue = 0;
unsigned long int avgValue;
float b;
int buf[10],temp;
void setup() {
 Serial.begin(9600);
}

void loop()
 {
    for(int i=0;i<10;i++)
 {
    buf[i]=analogRead(analogInPin);
    delay(10);
 }
 for(int i=0;i<9;i++)
 {
    for(int j=i+1;j<10;j++)
    {
        if(buf[i]>buf[j])
        {
            temp=buf[i];
            buf[i]=buf[j];
            buf[j]=temp;
        }
    }
 }
 avgValue=0;
 for(int i=2;i<8;i++)
 avgValue+=buf[i];
 float pHVol=(float)avgValue/4095.0/6;
 float phValue = -0.00357 * pHVol + 19.5077;
 Serial.print("sensor = ");
 Serial.println(phValue);

 delay(20);
```

Ensuite, la valeur du pH se trouve dans la variable ”phValue” et est traitée par l'analyseur.

4.5 Module RTC

Le DS1301 utilise une communication I2C pour envoyer et recevoir des données. Pour accélérer le prototypage, j'ai utilisé la libraire RTCLib. Cette libraire me permet d'instancier un objet RTC_DS1307, qui possède des fonctions pour configurer et lire l'heure du RTC.

Pour le faire fonctionner, il a simplement fallu que je change les pins de communication I2C dans le fichier qui définit l'objet RTC_DS1307 (21 = SDA, 22 = SCL).

4.6 Écran LCD

4.6.1 Mise en contexte

L'écran LCD utilité est un TFT ILI9341 (320x240). Cet écran utilise une communication SPI. Pour accélérer le prototypage, j'ai utilisé la libraire TFT_eSPI. Cette libraire me permet d'instancier un objet TFT_eSPI, qui possède plusieurs fonctions pour afficher des pixels sur l'écran.

Il a fallu que je modifie quelques fichiers de configuration de la libraire pour la configurer à mon prototype. Puisque la librairie est faite pour un Arduino, j'ai dû changer les pins utilisées pour concorder avec mon ESP32.

Pour l'affichage, j'ai utilisé l'exemple de la libraire ”TFT_Meter”. Cet exemple afficher trois ”meters” qui oscillent entre 0 et 100 (int) de façon sinusoïdale. Pour intégrer cet exemple dans mon projet, j'ai créé la librairie TFT_Meters. Dans cette libraire, j'ai copié les fonctions de l'exemple ”TFT_Meter” et je les ai rendues plus générales (en ajoutant un pointeur de TFT_eSPI en entrée).

J'ai aussi modifié les fonctions pour qu'ils puissent afficher en valeur numérique la température (double de 0-100), le TDS (int de 0 à 4000) et le pH (double de 0-14) adéquatement.

Cependant, l'affichage du pointeur pour la valeur de pH ne fonctionne pas correctement. Le fait l'intervalle de valeurs possible est réduit par rapport à 0-100 (elle est de 0-14 pour le pH) semble compliquer les déplacements du pointeur. Puisque cela n'affecte pas le fonctionnement principal du système, je n'ai pas plus essayé de le réparer

De plus, d'autres fonctions d'affichage se trouvent dans la librairie PapLib (Pool analyser pro Library), qui est plus spécifique au projet.

Voici un aperçu de l'affichage :

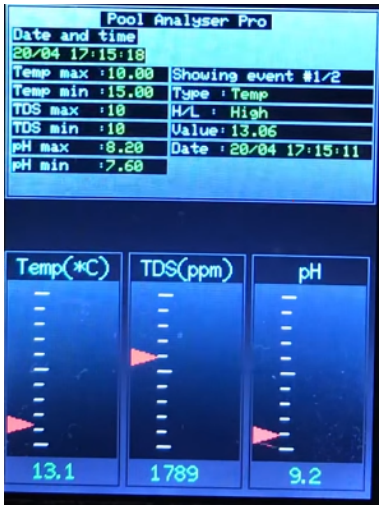

FIGURE 9 – Affichage du Pool Analyser Pro

4.6.2 Alimentation

Pour alimenter le LCD, il lui faut une tension continue de 5V. De plus, il faut un condensateur entre l'alimentation et la masse pour stabiliser le 5V. Sinon, l'écran LCD devient instable et ne peut pas fonctionner. Un condensateur de 220μF fonctionne bien.

Pour allumer les DELs, on doit mettre une tension haute sur la pin LED. Mon plan initial était d'ajouter un bouton externe qui lorsque pesé, allait imposer une tension haute sur une pin attaché à un interrupt. Cet interrupt allait imposer une tension haute à la pin connecté au LED du LCD à une tension de 3.3 V . Cependant, j'ai réalisé qu'une tension de 3.3V donnait une luminosité trop faible, surtout si quelqu'un l'utilisait dehors en plein jour. Pour ce prototype, l'alimentation des DELs est simplement le 5V du ESP32.

Expérimentalement, j'ai remarqué que le fil pour la pin CS du LCD est très sensible. Il suffit de rapprocher son doigt du fil et l'écran cesse de fonctionner. L'ajout d'un condensateur de 100μF corrige ce problème.

4.7 Application Windows

L'application logicielle est un Windows Form programmé en C#. Il s'agit d'une programmation événementielle qui agit comme interface usager pour l'utilisateur et comme afficheur d'information.

Voici l'affichage de l'application :

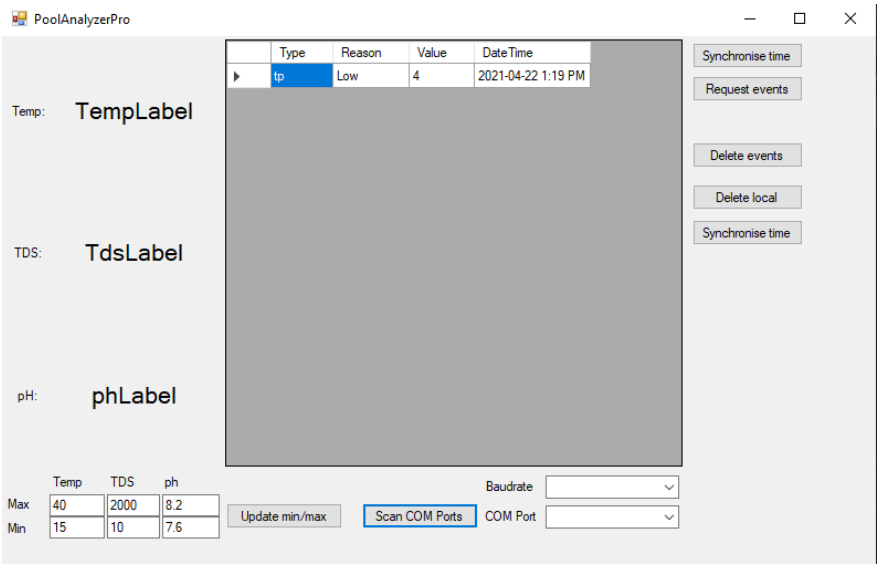

FIGURE 10 – Affichage de l'application Pool Analyzer Pro

1. On peut voir à gauche les éléments TempLabel, TDSLabel et phLabel. Ce sont les emplacements où les valeurs de température, de TDS et de pH s'affichent
2. En bas de l'affichage des mesures on peut voir des "textboxs". L'utilisateur peut donc entrer les valeurs pour les plages de valeurs de chaque mesure. Lorsque L'utilisateur veut changer les valeurs sur l'analyseur, il pèse sur le bouton "Update min/max". Cela envoie la commande appropriée à l'analyseur.
3. Au milieu de trouve un "data grid view", qui est lié à une liste d'"UnwantedEvent". Cette classe possède un type, une raison, une valeur et une date. Ce sont les événements indésirés reçus par l'analyseur.
4. À la droite des min/max se trouvent les éléments pour configurer la connexion Bluetooth. Par souci de temps, cette fonctionnalité n'a pas été implémentée.
5. À la droite du "data grid view" se trouve quelques autres boutons. Le premier permet de synchroniser l'heure de l'application avec l'analyseur.à
6. Le deuxième permet d'aller chercher les événements indésirés de l'analyseur (n'a pas été implémenté au complet,
7. Le troisième permet de supprimer les événements non désirés et envoie une commande à l'analyseur pour qu'il fasse pareille
8. Le quatrième permet de supprimer les événements non désirés, mais seulement sur l'application
9. Le dernier ne fait rien

4.8 Communication Bluetooth

4.8.1 Communication Bluetooth de l'application

Pour implémenter la communication Bluetooth dans l'application Windows Form, j'utilise le SerialDataReceivedEventHandler ainsi que l'objet SerialPort des librairies C#. Cela me permet d'envoyer des données avec serialPort.Write("message") ; et de traiter les données que je reçois avec l'événement DataReceivedHandler() qui s'exécute lorsque des données sont reçues sur un port de communication sériel de l'ordinateur qui

exécute l'application.

Pour configurer un port COM de l'ordinateur en mode Bluetooth, il suffit d'aller dans les paramètres avancés de Bluetooth.

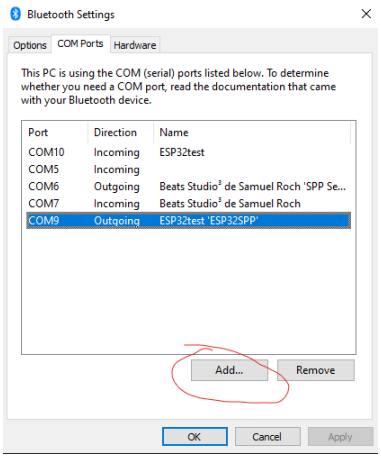

FIGURE 11 – Paramètres avancés de Bluetooth

Ensuite on peut lier un port COM à un appareil Bluetooth.

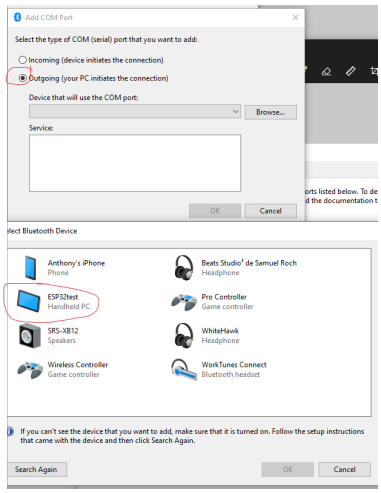

FIGURE 12 – Ajout d'un appareil Bluetooth à un port COM

Et le tour est joué. Dans l'application, il suffit de créer un SerialPort avec le port COM en question et la communication peut se faire. Par manque de temps, je n'ai pas rendu la connexion Bluetooth configurable dans l'application.

4.8.2 Communication Bluetooth de l'analyseur

Pour communiquer par Bluetooth sériel, j'utilise la libraire BluetoothSerial d'Arduino. Puisque je ne peux pas attacher d'interrupt à cette communication, je dois utiliser les deux coeurs du ESP32. Le premier pour gérer l'affichage et l'analyse et le deuxième pour recevoir des messages.

Dans Arduino, cela se fait facilement en utilisant la fonction xTaskCreatePinnedToCore((fonction, nom, espaceMemoire, NULL, priorité, tache, coeur)) [11]. Puisque la fonction loop() d'Arduino s'exécute déjà sur le coeur 0 du esp32, je n'ai qu'à ajouter une fonction pour la réception Bluetooth attachée au coeur 1. Le squelette du code pour l'analyseur devient alors :

```
void btRxTaskCode( void * pvParameters );
TaskHandle_t btRxTask;

void setup()
{
    xTaskCreatePinnedToCore(btRxTaskCode, "btRxTask", 10000, NULL, 1, &btRxTask, 1);
    setup();
}

void loop()
{
    analyserCode();
}

void btRxTaskCode( void * pvParameters )
{
    //while(true);
    for(;;)
    {
        btRxCode();
    }

}
```

Comme cela, le code de l'analyseur et du récepteur Bluetooth roule en parallèle.

4.8.3 Envoi de message

Pour communiquer, des strings avec une forme particulière sont envoyés entre l'application et l'analyseur. Pour ce faire, j'utilise les fonctions

- SerialBluetooth.print(message + String('\n')) pour l'analyseur
- serialPort.Write(message + '\n') pour l'appication

La forme des messages est toujours xx_yyyyyyyyy.. où xx est le nom de la commande, et yyy.. sont les données à envoyer.

Comme on peut voir, la fin d'un message se caractérise toujours par un '\n'. Cela est pour facilité la réception des messages, qui est expliquée . Les tableaux des différents messages se trouvent ci-dessous. un exemple de message pour une température trop haute à 20.0°C arrivé de 21 avril à midi pile serait : th_0200|00|00|12|21|04. Les "||" ne sont que là pour séparer les différentes données et n'apparaissent pas dans les messages.

Préfixe	**Message**	**Données**
tp_	Donnée de température	Température(4)
td_	Donnée de TDS	TDS(4)
ph_	Donnée de pH	pH(3)
th_	Envoie d'une température trop haute	Température(4)seconde(2)minute(2)heure(2)jour(2)mois(2)
tl_	Envoie d'une température trop basse	Température(4)seconde(2)minute(2)heure(2)jour(2)mois(2)
sh_	Envoi d'un TDS trop haut	TDS(4)seconde(2)minute(2)heure(2)jour(2)mois(2)
sl_	Envoi d'un TDS trop bas	TDS(4)seconde(2)minute(2)heure(2)jour(2)mois(2)
hh_	Envoi d'un pH trop haut	pH(2)seconde(2)minute(2)heure(2)jour(2)mois(2)
hl_	Envoi d'un pH trop bas	pH(2)seconde(2)minute(2)heure(2)jour(2)mois(2)

TABLE 4 – Messages Bluetooth envoyés de l'analyseur à l'application

Préfixe	**Message**	**Données**
tm_	L'heure de l'application	seconde(2)minute(2)heure(2)jour(2)mois(2)
mm_	Changement de min/max	tempMax(4)tempMin(4)tdsMax(4)tdsMin(4)pHMax(3)pHMin(3)
dl_	Supression d'événements indésirables	Aucune

TABLE 5 – Messages Bluetooth envoyés de l'application à l'analyseur

4.8.4 Réception de message

Pour que l'analyseur puisse recevoir des messages, il est constamment en train de vérifier s'il a reçu de l'information. Le cas échéant, un tampon de char accumule les caractères reçus. Lorsqu'un caractère '\n' est reçu, cela indique que le message a fini d'être transmis et le message peut être décodé selon le tableau 5. Le code simplifié ressemble à ceci :

```
void btRxTaskCode( void * pvParameters );
TaskHandle_t btRxTask;


char btReceivedChar;

void setup()
{
    xTaskCreatePinnedToCore(btRxTaskCode, "btRxTask", 10000, NULL, 1, &btRxTask, 1);
    setup();
}

void loop()
{
    analyserCode();
}

void btRxTaskCode( void * pvParameters )
{
    //while(true);
    for(;;)
    {
    if(SerialBT.available())
    {
      btReceivedChar = SerialBT.read();
      if(btReceivedChar != '\n')
      {
        btReceivedString += String(btReceivedChar);
      }
      else
      {
        decodeMessage();
      }

}
```

La réception de message du côté de l'application fonctionne pareillement. La seule différence est que l'application a un "interrupt" dédié à la réception de message.

4.9 Comparateur

Le module de comparaison est assez trivial. Voici le code simplifié pour la comparaison de température.

```
void btRxTaskCode( void * pvParameters );
TaskHandle_t btRxTask;


char btReceivedChar;

void setup()
{
    setup();
}

void loop()
{
    if(timeToCheckTemperature)
    {
        temperature = measureTemperature();
        if(temperature > tempMax)
        {
            sendTempTooHigh(temperature);
        }
        else if(temperature < tempMin)
        {
            sendTempTooLow(temperature);
        }
    }

    ...
}
```

4.10 Réalisation du Prototype

4.10.0.1 Stratégie de réalisation

En commençant l'assemblage de mon prototype, j'ai réalisé qu'une bonne stratégie serait de faire fonctionner un capteur en son entièreté. Ensuite, j'allais pouvoir faire les deux autres sans trop faire d'erreur. J'ai donc fait le capteur de température et son affichage sur l'écran. Ensuite, j'ai fait les deux autres. Finalement, j'ai configuré la communication Bluetooth et l'application.

4.10.0.2 Communication Bluetooth

Puisque la communication était la partie la plus nébuleuse pour moi, j'ai fait son implémentation par étapes. Au début, je me suis contenté d'un simple exemple Arduino pour m'assurer que la communication était possible. Ensuite, j'ai fait un programme C qui me permettait de communiquer par commande de console. Finalement, j'ai tout intégré dans l'application finale.

4.10.0.3 Prototype

Je suis donc arrivé à monter le prototype, qui ressemble à ceci :

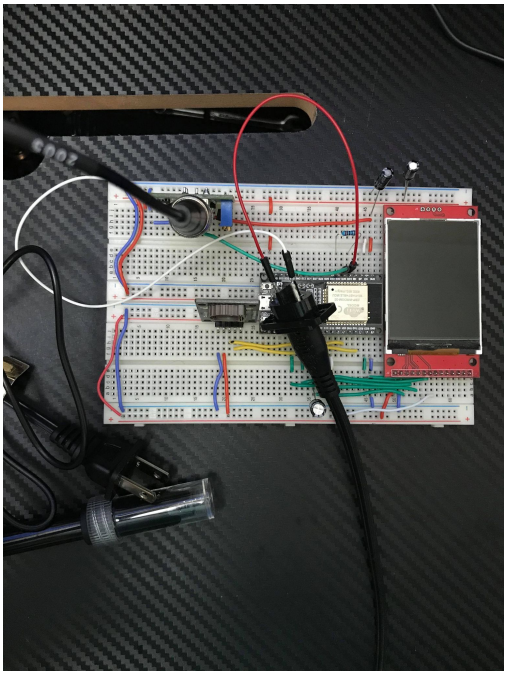

FIGURE 13 – Image du prototype

4.11 Résultats préliminaires et itérations du design

4.11.1 Capteur de TDS

Au début, j'ai suivi à la lettre la procédure de Micheal Ratcliffe [**?**] puisque les valeurs qu'il voulait mesurer concordaient avec les miennes. Par contre, j'ai réalisé que les valeurs pour mes résistances de l'eau différaient. Donc, il a fallu que j'analyse plus en profondeur la situation.

1. J'ai essayé de mettre une résistance en série avec la résistance de l'eau. Comme cela je pourrais choisir un R_d plus proche de $R + R_{eau}$. Cependant, j'ai vite compris que cela allait enlever de la résolution à mes mesures.
2. J'ai donc essayé d'utiliser une résistance de 220Ω (qui était la solution idéale selon mes calculs), en espérant que cela ne brise pas mon microcontrôleur. Heureusement, ça l'a marché.

4.11.2 Ajout de la connexion Bluetooth

Lorsque j'ai configuré la connexion Bluetooth sur l'analyseur, mon capteur de TDS a cessé de fonctionner (la tension mesurée était toujours 3.3V). J'ai alors pensé que le Bluetooth occasionnait des délais, ce qui rendait mon pulse de tension trop long et donc la résistance de l'eau devenait trop grande. J'ai donc essayé de supprimer les interrupts pendant que je mesurais un TDS, mais rien ne fonctionnait.

C'est en cherchant dans des forums que j'ai constaté que seulement les pins ADC 32-36 et 39 fonctionnaient lorsque la fonctionnalité Bluetooth était utilisée. Il a donc fallu que je change les plans de mon prototype.

5 Validation

5.1 Plan de tests

Cette section contient tous les tests de validations qui seront effectués pour assurer le fonctionnement global du prototype.

5.1.1 Température et communication Bluetooth

1. **Objectifs**
   - Vérifier si l'analyseur d'eau peut mesurer une température et l'afficher adéquatement sur l'écran LCD.
   - Vérifier si les températures mesurées concordent avec la réalité.
   - Vérifier si les messages Bluetooth s'envoient correctement à des distances différentes.
2. **Outils nécessaires/procédures**
   - Outils :
     - ☐ Bassin d'eau.
     - ☐ Source d'eau réglable (ex : robinet avec eau chaude/froide).
     - ☐ Thermomètre à alcool.
   - Procédures :
     - ☐ Remplir le bassin d'eau (5°C, 20°C ou 50°C).
     - ☐ Insérer la sonde du thermomètre et le thermomètre à alcool dans l'eau.
     - ☐ Vérifier que la température affichée change sur l'écran LCD.
     - ☐ Attendre que les valeurs mesurées se stabilisent.
     - ☐ Prendre 100 mesures dans avec un intervalle de 5 secondes entre chaque mesure.
     - ☐ Compter le nombre de messages reçus par l'application, le faire une une distance de 10, 30, 70 et 100m
     - ☐ Calculer la moyenne des mesures ainsi que le minimum et le maximum.
     - ☐ Répéter l'expérience pour les autres températures.
3. **Entrées à appliquer**
   - Aucune
4. **Sorties escomptées**
   - La température s'affiche adéquatement sur l'écran LCD
   - La température mesurée par le capteur concorde avec celui du thermomètre
   - Une communication Bluetooth réussie à une distance de 100m
5. **Résultats acceptables**
   - Température qui s'affiche adéquatement sur l'écran LCD au moins 19 fois sur 20.
   - Moyenne des températures captée à un maximum de 5% de la valeur du thermomètre à alcool.
   - Minimum et maximum inférieur à 3% de la moyenne des mesures.
   - Communication Bluetooth réussit au moins 19 fois sur 20 à une distance de 100m

5.1.2 TDS

1. **Objectifs**
   - Vérifier si l'analyseur d'eau peut mesurer un TDS et l'afficher adéquatement sur l'écran LCD.
   - Vérifier si les TDS mesurés concordent avec la réalité.
2. **Outils nécessaires/procédures**
   - Outils :
     - ☐ Bassin d'eau distillée.
     - ☐ Sel de table (NaCl)
   - Procédures :
     - ☐ Rincer le bassin d'eau avec de l'eau distillée .
     - ☐ Remplir le bassin d'eau avec une tasse (250ml) d'eau distillée.
     - ☐ Ajouter du sel à l'eau distillée (0g, 0.567g, 1.7g).
       Note : la solution de 0.567g s'obtient en ajoutant le 2/3 du volume d'eau d'une solution avec 1.7g de sel.
     - ☐ Insérer la sonde du TDS-mètre dans l'eau.
     - ☐ Attendre que les valeurs mesurées se stabilisent.
     - ☐ Prendre 100 mesures à 5 secondes d'intervalle.
     - ☐ Calculer la moyenne des mesures ainsi que le minimum et le maximum.
     - ☐ Comparer la moyenne mesurée à celle théorique (0g = 0ppm, 0.567g = 883.33ppm, 2g = 2650ppm).[4]
     - ☐ Répéter l'expérience pour les autres TDS.
3. **Entrées à appliquer**
   - Eau ayant une certaine quantité de sel dissous
4. **Sorties escomptées**
   - TDS qui s'affiche adéquatement sur l'écran LCD
   - TDS qui concorde avec la réalité
5. **Résultats acceptables**
   - TDS qui s'affiche adéquatement sur l'écran LCD au moins 19 fois sur 20.
   - Moyenne des TDS captés à un maximum de 5% de la valeur théorique.
   - Minimum et maximum inférieur à 3% de la moyenne des mesures.

5.1.3 pH

1. **Objectifs**
   - Vérifier si l'analyseur d'eau peut mesurer un pH et l'afficher adéquatement sur l'écran LCD
   - Vérifier si les pH mesurés concordent avec la réalité
2. **Outils nécessaires/procédures**
   - Outils :
     - ☐ Bassin qui peut contenir du liquide
     - ☐ Poudre pour solution à pH 4, 6.83 et 9.18
   - Procédures :
     - ☐ Rincer le bassin d'eau avec de l'eau distillée .
     - ☐ Remplir le bassin d'eau avec une tasse (250ml) d'eau distillée.
     - ☐ Ajouter le contenu du sachet de poudre pour la solution à pH (4, 6.83 ou 9.18).
     - ☐ Mélanger le contenu jusqu'à ce qu'il n'ait plus de solide dans l'eau
     - ☐ Insérer la sonde du pH-mètre dans l'eau.
     - ☐ Attendre que les valeurs mesurées se stabilisent.
     - ☐ Prendre 100 mesures à 5 secondes d'intervalle.
     - ☐ Calculer la moyenne des mesures ainsi que le minimum et le maximum.
     - ☐ Comparer la moyenne mesurée à celle théorique (4.01, 6.86 ou 9.18).
     - ☐ Répéter l'expérience pour les autres pH.
3. **Entrées à appliquer**
   - Eau ayant un certain pH
4. **Sorties escomptées**
   - pH qui s'affiche adéquatement sur l'écran LCD
   - pH qui concorde avec la réalité
5. **Résultats acceptables**
   - pH qui s'affiche adéquatement sur l'écran LCD 19 fois sur 20.
   - Moyenne des TDS captés à un maximum de 5% de la valeur théorique.
   - Minimum et maximum inférieur à 5% de la moyenne des mesures.

5.1.4 Enregistrent d'évènements non désirés

1. **Objectifs**
   - Vérifier si l'analyseur enregistre bien les événements non désirés
2. **Outils nécessaires/procédures**
   - Outils :
     - ☐ L'analyseur.
     - ☐ L'application.
   - Procédures :
     - ☐ Selon la température mesurée, imposer une température maximale inférieure à cette valeur.
     - ☐ Imposer une mesure de température à chaque seconde.
     - ☐ Prendre 100 mesures.
3. **Entrées à appliquer**
   - tempMin < température mesurée
4. **Sorties escomptées**
   - É.I qui s'enregistre dans l'analyser. On peut le voir sur l'affichage et les informations sont correctes.
   - La date et heure enregistrée concorde avec la réalité.
5. **Résultats acceptables**
   - E.I qui s'enregistre correctement au moins 19 fois sur 20.
   - E.I qui s'affiche correctement sur l'application 19 fois sur 20

5.1.5 Suppression d'évènements non désirés

1. **Objectifs**
   - Vérifier si l'analyseur supprime bien les événements non désirés
2. **Outils nécessaires/procédures**
   - Outils :
     - ☐ L'analyseur.
     - ☐ L'application.
   - Procédures :
     - ☐ Selon la température mesurée, imposez une température maximale inférieure à cette valeur.
     - ☐ Imposer une mesure de température à chaque seconde.
     - ☐ À chaque mesure prise (et É.I généré), supprimer les É.I .
     - ☐ répéter 100 fois.
     - ☐ compter les fois réussies.
3. **Entrées à appliquer**
   - tempMin < température mesurée
   - Commande de suppression d'É.I
4. **Sorties escomptées**
   - É.I qui s'efface de l'analyseur et de l'application
5. **Résultats acceptables**
   - E.I qui s'efface correctement au moins 19 fois sur 20.

5.2 Résultats et analyse

5.2.1 Température

5.2.1.1 Résultats

Moyenne	5.6
Diff (%)	0.06
Max	-
Min	-
Max(%)	-
Min(%)	-

Moyenne	20.6
Diff (%)	3.39
Max	20.7
Min	20.4
Max(%)	0.49
Min(%)	0.97

Moyenne	51.2
Diff (%)	1.19
Max	-
Min	-
Max(%)	-
Min(%)	-

TABLE 6 – Résultats d'échantillons pour une eau à 5.3°C, 19.9°C et 50.6°C

- De plus, l'affichage sur l'écran LCD fonctionne 20 fois sur 20
- La communication Bluetooth fonctionne 100% du temps à 10m
- La communication Bluetooth fonctionne 84% du temps à 30m
- La communication Bluetooth fonctionne 0% du temps à 70m
- La communication Bluetooth fonctionne 0% du temps à 100m

Note : Il a été impossible de déterminer les min/max pour les températures de 5°C et 50°C. En effet, je ne peux pas imposer une température constante autre que la température pièce pendant environ 1h30. Donc, j'ai pris quelques mesures ponctuelles et j'ai trouvé la moyenne des échantillons.

5.2.1.2 Analyse

1. Le test de fluctuation pour les températures de 5°C et 50°C5° 50° sont inconcluants. Il me faudrait plus de ressources de laboratoire pour faire ces tests.
2. Les résultats sont bel et bien à un écart maximal de 5% de la réalité. Cela respecte donc le cahier des charges.
3. Les min/max pour la température pièce sont bel et bien à un écart maximal de 3% de la moyenne. Cela respecte donc le cahier des charges.
4. L'affichage de la température est réussi 100% du temps. Cela respecte donc le cahier des charges.
5. La communication Bluetooth est réussie 0% du temps à une distance de 100m. Cela ne respecte donc pas le cahier des charges. En fait, il n'y a qu'à une distance de 10m que le seuil de 95% est respecté. Cela est très problématique pour mon prototype, puisque son but est de permettre à son utilisateur d'être loin de sa piscine lorsqu'il l'analyse. Cela est donc un point très important à améliorer.

Bref, la mesure de température respecte le cahier des charges pour son exactitude, pour son affichage local, et sa fluctuation à température pièce. De plus, elle est inconcluante pour les fluctuation à des températures de 5°C et 50°C. Malheureusement, le test pour la communication Bluetooth ne respecte pas les spécifications fonctionnelles.

5.3 TDS

5.3.0.1 Résultats

Moyenne	0
Diff (%)	0
Max	0
Min	0
Max(%)	0
Min(%)	0

Moyenne	780.54
Diff (%)	11.60
Max	791.25
Min	772.75
Max(%)	1.37
Min(%)	0.99

Moyenne	2657.58
Diff (%)	0.85
Max	2719.25
Min	2613
Max(%)	2.32
Min(%)	1.68

TABLE 7 – Résultats d'échantillons pour une eau à 0 ppm, 883 ppm et 2650 ppm

- De plus, l'affichage du TDS sur l'écran LCD fonctionne 20 fois sur 20

5.3.0.2 Analyse

1. On remarque que le capteur de TDS effectue une erreur de 11.6% de la réalité lorsqu'il essaie de mesurer un TDS de 883 ppm. Cela de respecte donc pas le cahier des charges. Pour les solutions ayant un TDS de 0 ppm et 2650 ppm, le capteur respecte le cahier des charges en termes d'exactitude.
2. Le capteur respecte le cahier des charges en termes de fluctuations pour toutes les valeurs de TDS.
3. L'analyseur respecte de cahier des charges en termes d'affichage de TDS.

Bref, l'analyseur respecte le cahier des charges pour ce qui touche le TDS pour tout sauf pour l'exactitude à un TDS de 883. Évidemment, c'est un problème à corriger. Le problème vient sûrement du fait que le capteur a été calibré avec une solution ayant un TDS de 2650 ppm. Donc, la solution à 883 ppm étant loin de cette valeur, les mesures s'éloignent de la réalité.

5.4 pH

Moyenne	3.6
Diff (%)	11.11
Max	3.6
Min	3.6
Max(%)	0
Min(%)	0

Moyenne	6.78
Diff (%)	1.14
Max	6.83
Min	6.73
Max(%)	0.70
Min(%)	0.77

Moyenne	9.08
Diff (%)	1.07
Max	9.15
Min	9.02
Max(%)	0.74
Min(%)	0.69

TABLE 8 – Résultats d'échantillons pour une eau à un pH de 4.01, 6.86 et 9.18

- De plus, l'affichage du TDS sur l'écran LCD fonctionne 20 fois sur 20

1. On remarque qu'aucun des résultats ne respecte le cahier des charges pour un pH de 4. En effet, le capteur sature à cette valeur de pH. Il semble que les tensions obtenues pour un pH donné ne concorde pas avec la fiche technique du capteur.
2. Pour les deux autres mesures, le capteur de pH respecte de cahier des charges en termes d'exactitude et en termes de fluctuations.
3. L'analyseur respecte de cahier des charges en termes d'affichage de pH.

Bref, le seul problème du capteur de pH semble être de mesurer un pH trop acide. Heureusement, le pH voulu pour une piscine à chlore est entre 7.2 et 7.6. Cependant, il faudrait quand même arranger ce problème pour respecter les spécifications fonctionnelles.

5.4.1 Enregistrement et suppression d'événements indésirables

5.4.1.1 Résultats

1. L'enregistrement des évènements indésirables est réussi 20 fois sur 20
2. La suppression des évènements indésirables est réussie 20 fois sur 20

5.4.1.2 Analyse

1. L'enregistrement des évènements indésirables respecte le cahier des charges.
2. La suppression des évènements indésirables respecte le cahier des charges.
3. L'envoi de date et heure est adéquat. De plus, la date et heure est précise à une minute près, et ce, même après un long instant (quelques jours).

6 Conclusion

6.1 Prix total du prototype

Un de mes buts était de concevoir un prototype le moins cher possible et qui était concurrentiel avec le marché. Voici donc un tableau des prix des différentes composantes.

Pièce	Prix ($)
ESP32	13.25
RTC DS1301	3.2
LCD TFT ILI 9341	19.99
pH E-201-C	32.91
Temp. DS18B20	23.68
Capteur TDS	13.5
Autres pièces	1.25
Prévision pour PCB	10
Prévision pour printing 3D	50
Total	167.78

TABLE 9 – Liste de composantes et leur prix

On peut donc voir que le prix est considérablement plus bas que ce que l'on peut trouver sur le marché. Cependant, ce prix vient avec des inconvénients.

6.2 Évaluation des limites et incertitudes

1. La communication Bluetooth devient inadéquate à partir d'une distance de 30m. Cela est problématique puisque mon produit est fait pour qu'un utilisateur puisse analyser sa piscine à une assez grande distance (de chez lui).
2. Le capteur de TDS est inexact pour un TDS faible. Cela est problématique, mais au moins le capteur de TDS est exact pour un TDS d'environ 2000, qui est le point critique pour une piscine à chlore.
3. Le capteur de pH est incapable de mesurer un pH trop acide (environ 4). Au moins cette valeur ne se trouve pas dans la plage normale de pH pour une piscine à chlore. Le capteur est tout de même précis pour des valeurs qui comprennent les valeurs de pH standards (7.2-7.8).
4. Il me faudrait plus de ressources pour tester la fluctuation des mesures pour des températures de 5°C et 50°C. Par contre, les résultats sont positifs pour une température pièce.

6.3 Proposition des travaux subséquents

Pour rendre mon application plus efficace, j'améliorais les points suivant :

1. Changer l'alimentation (120VAC/5VDC) pour une alimentation à pile. Il faudrait un abaisseur de tension qui consomme le moins de courant que possible.
2. Changer la communication Bluetooth pour une connexion Wi-Fi. De plus, ajouter une fonctionnalité qui permet à l'analyseur d'envoyer tous ses événements indésirables enregistrés si jamais l'application était hors ligne pour un certain temps.
3. Concevoir et fabriquer une sonde à TDS (ne pas utiliser une prise pour mur) qui serait plus performant.
4. Concevoir et fabriquer un PCB pour le prototype
5. Concevoir une coque par impression 3D qui rendrait le prototype résistant à la pluie.

7 Apprentissage continu

1. Ce projet m'a permis d'en apprendre beaucoup sur le prototypage et du design qui vient avec. J'ai pu voir la complexité de trouver un produit potentiel et la difficulté de le rendre concurrentiel sur le marché.
2. Ce projet m'a aussi fait réaliser que les choses ne vont pas toujours comme on le veut. Même si on passe beaucoup de temps à penser à une solution, il y a toujours des détails qui peuvent s'échapper.
3. Ce projet m'a beaucoup fait travailler mes compétences en programmation. J'ai même pu apprendre un langage que je ne connaissais pas, ce C# (même si c'est très familier au C/C++.
4. Ce projet m'a permis d'en apprendre davantage sur ma piscine, surtout pour le TDS. Avant, j'ignorais que cette mesure était existait.

Références

[1] Espressif Systems. *ESP32-WROOM-32 Datasheet*. https://www.espressif.com 2021.

[2] Konstantin Dimitrov. *Arduino thermometer with DS18B20*. https://create.arduino.cc 2016.

[3] Michael Ratcliffe. *Three Dollar EC - PPM Meter [Arduino]*. https://hackaday.io 2015.

[4] Tom Scherer, Miranda Meehan *Using Electrical Conductivity and Total Dissolved Solids Meters to Field Test Water Quality*. https://www.ag.ndsu.edu 2019.

[5] Williams. Kent *What's the Fuss over Total Dissolved Solids ?, PPOA Pumproom Press*. What's the Fuss over Total Dissolved Solids?.

[6] *pH and Chlorine values for good water quality*. https://www.pahlen.com.

[7] *Site d'achat pour le pHin*. https://www.poolsuppliescanada.ca.

[8] Microsoft *SerialPort.DataReceived Event*. https://docs.microsoft.com.

[9] *Accuracy of millis() for keeping time*. https://forum.arduino.cc.

[10] e-Gizmo. *PH Sensor E-201-C : Technical Manual Rev 1r0*. .

[11] Random Nerd Tutorial *Create Tasks in Different Cores – Example*. https://randomnerdtutorials.com.

[12] PHTA Recreational Water Quality Committee *Total Dissolved Solids : The Facts*. https://aquamagazine.com 2019.

[13] Zach Morris *Total Dissolved Solids : The Ultimate Guide For Aquatic Operators*. https://clearcomfort.com.