



RAPPORT MINI-PROJET

**Suivi temps réel de la consommation d'énergie d'un
Sous-marin**



Réalisé par :

- **BANAR Hasnaa**
- **EL KHOUKH Soumia**
- **OKHAYA Achraf**
- **OUFIR Nadia**
- **AIT MOUH Nadia**

Encadré par : Pr Hicham MEDROMI

Table des matières

Remerciement	2
Introduction	3
Cahier de charge :	4
Chapitre 1 :	5
Modélisation et Conception	5
Introduction :	5
Les taches élémentaires :	5
Catégories des tâches :	5
Les performances de l'architecture :	6
Conception :	6
Diagramme de classe :	6
Chapitre 2 : réalisation	8
Introduction :	8
Schématisation :	8
Outils et technologies :	9
Le langage C :	9
MySQL :	10
Android Studio :	10
Code Source :	11
La fonction temps réel :	11
La classe « Point » :	11
La fonction « continuer » :	12
La fonction « Move » :	12
Conclusion :	13
Chapitre 3 : Exécution :	14
Introduction :	14
Exécution :	14
Interface graphique :	16
Conclusion :	16
Conclusion Générale	17

Remerciement

Nous tenons à remercier dans un premier temps, toute l'équipe pédagogique de l'ENSEM et les intervenants professionnels responsables de la formation du département informatique, pour leurs efforts, leurs disponibilités et leurs générosités en termes de partage d'informations.

Nous tenons également à exprimer notre très profonde gratitude à notre cher enseignant Monsieur HICHAM MEDROMI, pour ses précieux conseils, sa patience, sa gentillesse, sa disponibilité et son total dévouement pour donner naissance à ce travail.

On saisit l'occasion pour remercier tous nos enseignants à l'ENSEM qui nous offrent des formations très complètes ce qui nous a aidé à bien aboutir notre projet. Nous tenons également à exprimer notre sincère gratitude envers tous ceux qui nous ont aidé ou ont participé au bon déroulement de ce projet.

Introduction

Un **système d'exploitation mobile** est un système d'exploitation conçu pour fonctionner sur un appareil mobile. Ce type de système d'exploitation se concentre entre autres sur la gestion de la connectivité sans fil et celle des différents types d'interface.

On trouve ces systèmes d'exploitation sur les smartphones : Symbian OS de Nokia, iOS de Apple, BlackBerry OS de RIM, Windows Phone de Microsoft, Palm webOS, Android de Google, Bada de Samsung.

Android, Bada, Firefox OS, Maemo, Tizen et WebOS sont construits sur un noyau Linux ; en revanche le système d'exploitation des iPhones dérive de BSD et de NeXTSTEP qui sont tous deux liés à Unix.

Les deux principaux systèmes en 2019 sont Android et iOS. Deux autres systèmes ont eu leurs développements stoppés et seules des mises à jour de sécurité seront fournies jusqu'à fin 2019 ; ce sont Windows 10 Mobile et BlackBerry OS. Pour les autres systèmes, le développement a été stoppé ou abandonné faute de possibilité de croissance ou de parts de marché suffisamment importantes.

Dans ce projet, on est amené à réaliser une application permettant de faire le suivi de la consommation de l'énergie et la prédiction l'autonomie sur un Sous-Marin. Le Sous-Marin utilise principalement différentes sources d'énergie (thermique, électrique, nucléaire). Ainsi, L'acquisition des données se fait à partir de plusieurs sources différentes, l'application devra donc permettre de faire le suivi de la consommation de chaque module du Sous-marin en temps réel et aussi de pouvoir donner la consommation globale d'énergie en chaque instant. Une couche d'intelligence est primordiale pour pouvoir donner des estimations sur la durée des missions en exploitant des bases de connaissances d'où la nécessité d'utiliser des systèmes multi-agents.

Cahier de charge :

Suivi temps réel de la consommation d'énergie d'un Sous-marin :

L'objectif de ce mini projet est de réaliser une application permettant de faire le suivi de la consommation de l'énergie et la prédiction l'autonomie sur un Sous-Marin.

Le Sous-Marin utilise principalement différentes sources d'énergie (thermique, électrique, nucléaire). Ainsi, L'acquisition des données se fait à partir de plusieurs sources différentes, l'application devra donc permettre de faire le suivi de la consommation de chaque module du Sous-marin en temps réel et aussi de pouvoir donner la consommation globale d'énergie en chaque instant.

Une couche d'intelligence est primordiale pour pouvoir donner des estimations sur la durée des missions en exploitant des bases de connaissances d'où la nécessité d'utiliser des systèmes multi-agents.

La modélisation de l'application devra respecter le caractère distribué de la tâche et aussi elle devra prendre en considération les contraintes temps réel associés.

Travail à faire :

- 1- Distribuer l'architecture ci-dessus en tâche élémentaire (agent) ?
- 2- Regrouper les taches de même catégorie ?
- 3- Caractériser les performances de l'architecture ?
- 4- Proposer l'architecture conceptuelle ?
- 5- Proposer un Ordonnanceur pour cette architecture ?
- 6- Réaliser le programme ?
- 7- Visualiser dynamiquement l'enchaînement des exécutions des taches ?
- 8- Conclure ?

Chapitre 1 :

Modélisation et Conception

Introduction :

Dans ce projet, notre application va devoir exécuter plusieurs tâches en parallèle pour pouvoir faire le suivi de la consommation modulaire et globale du sous-marin alors qu'il est en état de marche. Il va falloir aussi que le sous-marin se déplace vers sa destination, pendant ce temps nous allons détecter sa position via plusieurs capteurs.

Les tâches élémentaires :

L'architecture du sous-marin peut-être distribuer en plusieurs tâches élémentaire agents que va effectuer notre application :

- La détection des mouvements du sous-marin :
 - ✓ Aller
 - ✓ Monter
 - ✓ Descendre
 - ✓ Reculer
 - ✓ Pivoter
- Acquisition de données et source de données
- Suivi de consommation
- Consommation modulaire
- Consommation énergétique
- Echéance
- Tolérance

Catégories des tâches :

Après avoir cité les différentes tâches, on va maintenant les regrouper en quatre catégories comme suit :

- ✓ **Groupe 1** : Monter, descendre, aller, reculer et pivoter.
- ✓ **Groupe 2** :
 - Acquisition de données
 - Source de données

✓ **Groupe 3 :**

- Suivi de consommation
- Consommation modulaire
- Consommation énergétique

✓ **Groupe 4 :**

- Echéance
- Tolérance

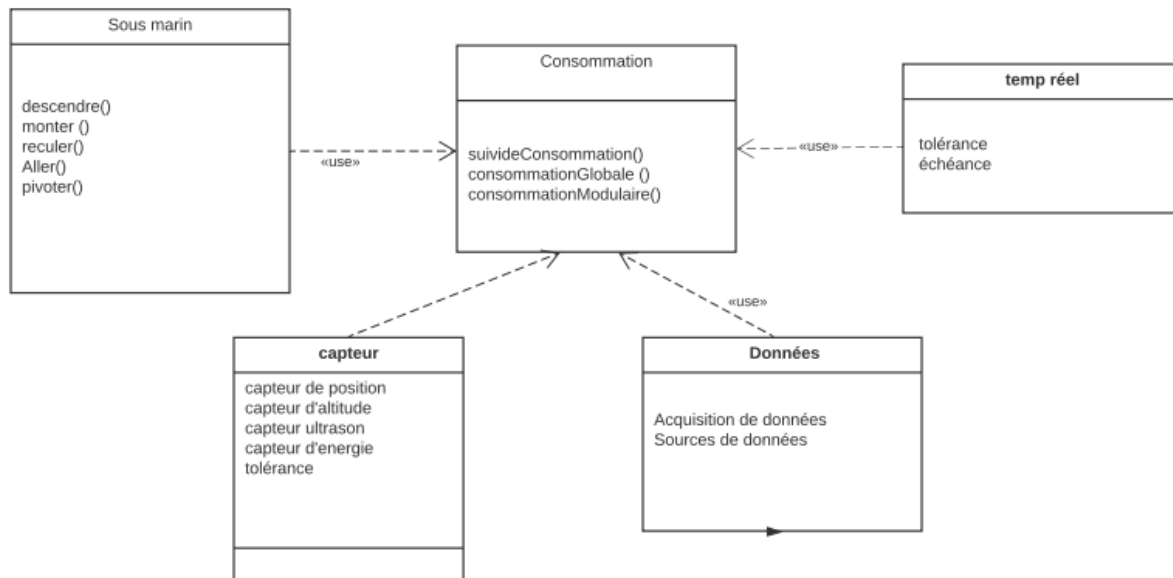
Les performances de l'architecture :

Les groupes de tâches ci-dessus vont toutes s'exécuter en parallèle selon l'échéance qu'on va donner à chacune de ces tâches et donc cette application aura une architecture distribuée temps réel parallèle (MIMD). Une machine parallèle est essentiellement un ensemble de processeurs qui coopèrent et communiquent. Une machine MIMD (instructions multiples, données multiples) est une technique employée pour réaliser le parallélisme. Les machines utilisant MIMD ont un certain nombre de processeurs qui fonctionnent de manière asynchrone et indépendante. À tout moment, différents processeurs peuvent exécuter différentes instructions sur différents éléments de données.

Conception :

Diagramme de classe :

Le diagramme de classe est un schéma utilisé en génie logiciel pour présenter les classes et les interfaces des systèmes ainsi que les différentes relations entre celles-ci. Ce diagramme fait partie de la partie statique d'UML car il fait abstraction des aspects temporels et dynamiques.



La figure ci-dessus représente le diagramme de classes de notre projet, qui est sous forme d'un schéma qui présente les différentes classes utilisées dans notre application, ainsi que les relations entre eux :

- **La classe « Sous-marin »** : c'est une classe qui exécute des méthodes suivantes : descendre, monter, reculer, aller et pivoter.
- **La classe « Consommation »** : c'est la classe qui fait les suivis de consommation globale et modulaire.
- **La classe « temps réel »** : cette classe nous donne l'échéance et la tolérance de chaque tâche.
- **La classe « capteur »** : cette classe contient tous les capteurs utilisés dont le capteur position, le capteur d'altitude, le capteur ultrason, le capteur d'énergie et la tolérance.
- **La classe « données »** : la classe de données est sensé faire l'acquisition de données.

Chapitre 2 : réalisation

Introduction :

Après avoir effectué la modélisation et la conception de notre projet, nous allons maintenant passer à la phase finale, la phase de réalisation. On va d'abord commencer par la schématisation des tâches du projet, puis nous allons choisir l'outil avec lequel nous allons programmer. Pour enfin avoir un code qui répond au cahier de charge.

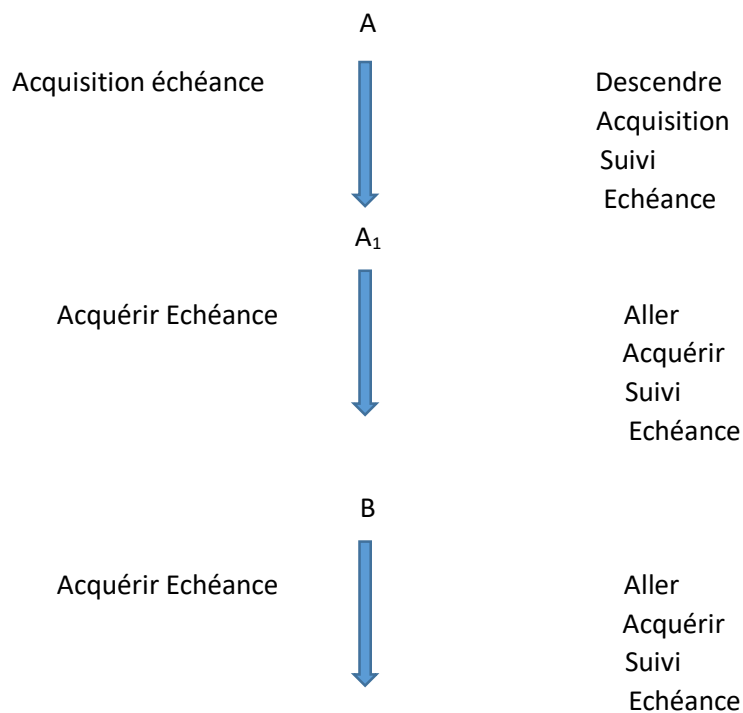
Schématisation :

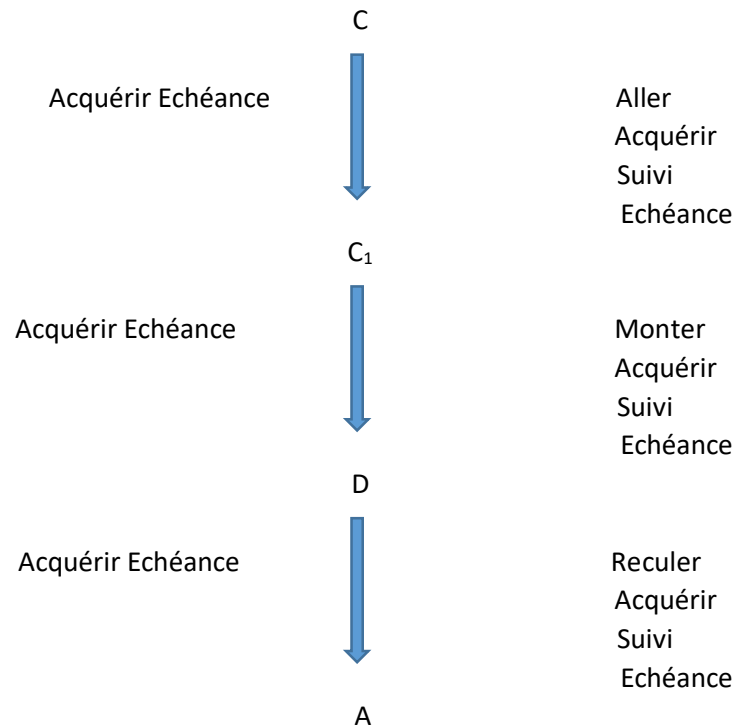
Soit le chemin du sous-marin (ABCD A)



Chacun des points par lequel va passer le sous-marin a des coordonnées selon les axes x et y. On suppose que les points A, B, C, D sont des points primaires et les autres points sont des points secondaires.

Les tâches de notre application s'exécuteront de la manière suivante :





- Echéance tâche => 10s
- Echéance capteur => 2s
- Echéance capteur d'énergie => 5s

Outils et technologies :

Le langage C :

Puisque nous allons travailler avec la notion de temps réel, on a choisi un langage qui respecte cette notion, qui n'est d'autre que le langage C.



C est un langage de programmation impératif généraliste, de bas niveau. Inventé au début des années 1970 pour réécrire UNIX, C est devenu un des langages les plus utilisés, encore de nos

jours. De nombreux langages plus modernes comme C++, C#, Java et PHP ou Javascript ont repris une syntaxe similaire au C et reprennent en partie sa logique. C offre au développeur une marge de contrôle importante sur la machine (notamment sur la gestion de la mémoire) et est de ce fait utilisé pour réaliser les « fondations » (compilateurs, interpréteurs...) de ces langages plus modernes.

MySQL :



MySQL est un système de gestion de bases de données relationnelles (SGBDR). Il est distribué sous une double licence GPL et propriétaire. Il fait partie des logiciels de gestion de base de données les plus utilisés au monde, autant par le grand public (applications web principalement) que par des professionnels, en concurrence avec Oracle, PostgreSQL et Microsoft SQL Server.

Android Studio :



Android Studio est un environnement de développement pour développer des applications mobiles Android. Il est basé sur IntelliJ IDEA et utilise le moteur de production Gradle. Il

peut être téléchargé sous les systèmes d'exploitation Windows, macOS, Chrome OS et Linux¹⁷.

Code Source :

La fonction temps réel :

La bibliothèque **'time.h'** : pour bien illustrer la notion de temps réel, on a utilisé la bibliothèque **time.h**. L'en-tête **time.h** définit quatre types de variables, deux macros et diverses fonctions pour manipuler la date et l'heure (**size_t**, **clock_t**, **time_t** et **struct tm**).

La fonction **time_t** est un type adapté pour stocker l'heure du calendrier.

```
1  #include <stdio.h>
2  #include <time.h>
3
4  void temps_reel(int seconds) {
5      /* Init. */
6      time_t start_time = 0;
7      time_t current_time = 0;
8
9      /* Operate. */
10     start_time = time(NULL);
11     while(current_time-start_time+1 <= seconds) {
12         current_time = time(NULL);
13     }
14 }
```

Figure 1 : code de la fonction "temps_reel"

La classe « Point » :

La classe point affiche les coordonnées du sous-marin selon les axes x,y.

```
15  class point
16  {
17      public:
18      void affiche(void){
19          printf("(%d , %d)\n",i,j);
20      }
21
22      int i;
23      int j;
24  };
```

Figure 2 : la classe "point"

La fonction « continuer » :

Cette fonction permet de faire le déplacement du sous-marin, l'acquisition de position, d'énergie et de donnée.

Après chaque 5s, le capteur d'énergie va acquérir la donnée de la température pour mesurer la consommation énergétique du sous-marin.

Après chaque 2s, le capteur de position acquiert de son côté les coordonnées du sous-marin, avec son altitude.

Après chaque 10s, on nous donne le suivi de consommation.

Pendant ce temps, le sous-marin fait son trajet vers sa destination.

```
26 void continuer(point m,point z )
27 {
28     int c;
29     printf("capteur de position ... \n");
30     temps_reel(2);
31     printf("capteur d'energie ... \n");
32     temps_reel(5);
33     printf("Acquisition de donnees...\n");
34     temps_reel(2);
35     printf("Suivi ... \n");
36     temps_reel(10);
37
38     if (m.i<z.i){
39         for(c=m.i;c<z.i;c++){
40             printf("    trajet en cours...\n");
41         }
42     }
43     else{
44         for(c=z.j;c<m.j;c++){
45             printf("    trajet en cours...\n");
46         }
47     }
48 }
49
50 }
```

Figure 3 : la fonction "continuer"

La fonction « Move » :

La fonction « move » permet nous donner l'emplacement du sous-marin quand il arrive en un point.

```

51 void Move(point l[],int k){
52     printf("Mouvement au point \n");l[k].affiche();
53     if (k==6) {
54         printf("Retour au point de depart : A \n");l[0].affiche();
55     }
56     }
57
58     continuer(l[k],l[k-1]);
59     printf("Je suis au point \n");l[k].affiche();
60
61 }
62

```

Figure 4 : la fonction "move"

La fonction « main » :

```

81 main ()
82 {
83     int c,k=0;
84     point A,A1,B,C,C1,D,l[6];
85     A.i=0;A.j=0;
86     A1.i=0;A1.j=4;
87     B.i=2;B.j=4;
88     C.i=4;C.j=4;
89     C1.i=6;C1.j=4;
90     D.i=6; D.j=0;
91
92
93     l[0]=A;
94     l[1]=A1;
95     l[2]=B;
96     l[3]=C;
97     l[4]=C1;
98     l[5]=D;
99     l[6]=A;
100
101
102     //l={A,A1,B,C,C1,D};
103
104     printf("Demarrage...\n");
105     temps_reel(2);
106     printf("Je suis au point de depart : A \n") ; l[0].affiche();
107     while (k<7 && k!=6 )
108     {
109         printf("entrez 1 pour le point suivant et 0 pour reculer : ");scanf("%d",&c);
110         k=avan_rec(c,k);
111         Move(l,k);
112     }
113     printf("arret" );
114 }

```

Conclusion :

A travers ce chapitre nous avons adressé chaque partie du code, nous allons ensuite passer à l'exécution de ce que nous avons programmé.


Chapitre 3 : Exécution :

Introduction :

Après avoir exhibé notre code source que vous allez aussi trouver dans les fichiers que nous vous avons envoyé, on va maintenant voir son exécution.

Exécution :

Nous constatons ci-dessous que le sous-marin commence par le point A avec les coordonnées (0,0), puis on appuie sur la touche 1, le véhicule va ensuite voyager vers sa prochaine destination, alors que les capteurs sont en train d'acquérir les informations dont nous avons besoin. Pour arriver ensuite au point A₁ de coordonnées (0,4).



```
CAUsers\SQUIMIA\Downloads\projet-sous-marin.exe
Je suis au point de depart : A
(0 , 0)
entrez 1 pour le point suivant et 0 pour reculer : 1
Mouvement au point
(0 , 4)
capteur de position ...
capteur d'energie ...
Acquisition de donnees...
Suivi ...
    trajet en cours...
    trajet en cours...
    trajet en cours...
    trajet en cours...
Je suis au point
(0 , 4)
entrez 1 pour le point suivant et 0 pour reculer : 1
Mouvement au point
(2 , 4)
capteur de position ...
capteur d'energie ...
Acquisition de donnees...
Suivi ...
Je suis au point
(2 , 4)
entrez 1 pour le point suivant et 0 pour reculer : 1
Mouvement au point
(4 , 4)
capteur de position ...
capteur d'energie ...
Acquisition de donnees...
Suivi ...
Je suis au point
(4 , 4)
entrez 1 pour le point suivant et 0 pour reculer : 1
Mouvement au point
(6 , 4)
capteur de position ...
capteur d'energie ...
Acquisition de donnees...
Suivi ...
Je suis au point
(6 , 4)
entrez 1 pour le point suivant et 0 pour reculer :
```

Figure 5 : l'exécution du programme

Lorsqu'il arrive au point de sa destination encore une fois après être passer par D de coordonnées (6,0).

```
C:\Users\SOUMLIA\Downloads\projet-sous-marin.exe
entrez 1 pour le point suivant et 0 pour reculer : 1
Mouvement au point
(6 , 0)
capteur de position ...
capteur d'energie ...
Acquisition de donnees...
Suivi ...
Je suis au point
(6 , 0)
entrez 1 pour le point suivant et 0 pour reculer : 1
Mouvement au point
(0 , 0)
Retour au point de départ : A
(0 , 0)
capteur de position ...
capteur d'energie ...
Acquisition de donnees...
Suivi ...
    trajet en cours...
    trajet en cours...
    trajet en cours...
    trajet en cours...
    trajet en cours...
    trajet en cours...
Je suis au point
(0 , 0)
arret
Process returned 0 (0x0)   execution time : 284.384 s
Press any key to continue.
```

Figure 6 : exécution du code

Interface graphique :

Pour l'interface graphique on trouve 4 boutons de mouvements du sous-marin, puis en bas nous remarquons deux boutons nous affichant les données acquises par les capteurs de positions et d'énergie. Ceci va nous permettre de faire le suivi de la consommation du sous-marin en question.

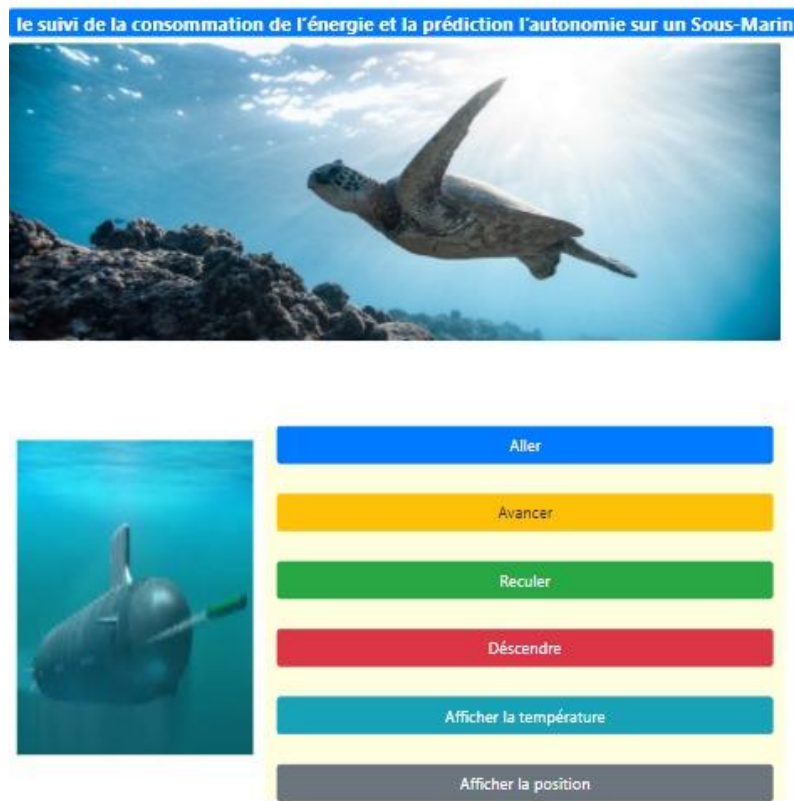


Figure 7 : Interface graphiques

Conclusion :

Dans cette partie, nous avons adressé l'exécution du code source avec toutes les étapes par lesquelles passent le programme, ensuite nous avons vu l'interface graphique de notre application avec ces différents composants.

Conclusion Générale

Notre projet permet le suivi de consommation modulaire et globale d'un sous-marin tout au long de son trajet.

L'objectif était de créer une application temps réel permettant de faire le suivi de la consommation de l'énergie et la prédiction l'autonomie sur un Sous-Marin.

Durant la réalisation de ce projet nous avons implémenté les différentes tâches complexes, ainsi que nous avons augmenté l'efficacité des fonctionnalités de la plateforme. Par ailleurs ce projet a non seulement glorifié les outils appris dans le cours en les pratiquant mais nous a aussi permis de réaliser une plateforme couvrant un fléau actuellement critique dans le monde.

Nous ne pouvons alors qu'affirmer le grand intérêt qu'on porte à ce projet qui nous a permis d'assimiler la notion de « temps réel » et son rôle primordial.