



Mini-projet BOBARIUM

David SALLÉ (david.salle@ensemblescolaire-niort.com)

Ce document est mis à disposition selon les termes de la licence

[Creative Commons BY-NC-SA 3.0](https://creativecommons.org/licenses/by-nc-sa/3.0/)



Version du document : v0.7

Date : 25/10/2024

Table des matières

1 - Introduction.....	2
2 - Cahier des charges.....	3
2.1 - Présentation.....	3
2.2 - Objectifs.....	3
2.3 - Expression du besoin.....	4
2.4 - Contraintes.....	4
2.5 - IHM.....	5
2.6 - Ressources.....	6
2.6.1 - Système d'exploitation.....	6
2.6.2 - Capteur de Bobarium.....	7
2.6.3 - Capteur ultrason.....	7
2.6.4 - Logiciels.....	8
3 - Évaluation.....	9
3.1 - Travail à rendre.....	9
3.2 - Critères d'évaluations.....	10
5 - Éléments de codage.....	11
5.1 - Pour la communication.....	11
5.1.1 - Protocole.....	11
5.1.2 - Sécurité.....	12
5.2 - Pour l'IHM.....	13
5.2.1 - Les outils React Native.....	13
5.2.2 - Transformer les valeurs en carte.....	13

1 - Introduction

Ce mini-projet de synthèse doit permettre à des équipes de 2 étudiants de ré-exploiter les notions abordées en architecture, système d'exploitation, multi-tâches, programmation réseaux, mobile et les outils de développement.

2 - Cahier des charges

2.1 - Présentation

Une équipe de chercheurs de l'université de Balnave vient de découvrir un nouvel élément : le **Bobarium**. Dans la classification de Mendeleïev, il s'agit de l'élément 119, un métalloïde aux propriétés étonnantes puisqu'il permettrait de tripler les performances des panneaux photovoltaïques actuels !

Période	1A		2A												13A		14A		15A		16A		17A		18A		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22					
1	hydrogène 1 H 1.00794	nom de l'élément (gaz, liquide ou solide à 0°C et 101,3 kPa) numéro atomique symbole chimique masse atomique relative ou [celle de l'isotope le plus stable]																				13 Al 26,9815386	14 Si 28,0855	15 P 30,973762	16 S 32,06	17 Cl 35,4527	18 Ar 39,948
2	lithium 3 Li 6,941	beryllium 4 Be 9,012182											bor 5 B 10,811	carbone 6 C 12,0107	azote 7 N 14,00644	oxygène 8 O 15,9994	fluor 9 F 18,9984032	néon 10 Ne 20,1797									
3	sodium 11 Na 22,98976928	magnésium 12 Mg 24,3050	3 BB	4 VB	5 VB	6 VB	7 VB	8 VB	9 VB	10 VB	11 IB	12 IB	aluminium 13 Al 26,9815386	silicium 14 Si 28,0855	phosphore 15 P 30,973762	soufre 16 S 32,06	chlore 17 Cl 35,4527	argon 18 Ar 39,948									
4	potassium 19 K 39,0983	calcium 20 Ca 40,078	scandium 21 Sc 44,955912	titane 22 Ti 47,867	vanadium 23 V 50,9415	chrome 24 Cr 51,9961	manganèse 25 Mn 54,938045	fer 26 Fe 55,845	cobalt 27 Co 58,933195	nickel 28 Ni 58,6934	cuivre 29 Cu 63,546	zinc 30 Zn 65,39	gallium 31 Ga 69,723	germanium 32 Ge 72,61	arsenic 33 As 74,92160	sélénium 34 Se 78,96	brome 35 Br 79,904	krypton 36 Kr 83,80									
5	rubidium 37 Rb 85,4678	strontium 38 Sr 87,62	yttrium 39 Y 88,90585	zinc 40 Zn 91,224	niobium 41 Nb 92,90638	molybdène 42 Mo 95,94	technétium 43 Tc 97,9072	rhodium 44 Rh 101,07	paladium 45 Pd 106,42	argent 46 Ag 107,8682	cadmium 47 Cd 112,411	indium 48 In 114,818	étain 49 Sn 118,710	antimoine 50 Sb 121,760	tellure 51 Te 127,60	iode 52 I 126,90447	xénon 54 Xe 131,29										
6	césium 55 Cs 132,9054519	baryum 56 Ba 137,327	lanthanides 57-71		hafnium 72 Hf 178,49	tantale 73 Ta 180,94788	tungstène 74 W 183,84	rénium 75 Re 186,207	osmium 76 Os 190,23	iridium 77 Ir 192,227	platine 78 Pt 195,084	or 79 Au 196,966569	mercure 80 Hg 200,59	thallium 81 Tl 204,3833	plomb 82 Pb 207,2	bismuth 83 Bi 208,98040	polonium 84 Po [209]	astate 85 At [209]	radon 86 Rn [222]								
7	francium 87 Fr [223]	radium 88 Ra [226]	actinides 89-103		rutherfordium 104 Rf [261]	dubnium 105 Db [262]	seaborgium 106 Sg [266]	bohrium 107 Bh [264]	hassium 108 Hs [269]	meitnerium 109 Mt [268]	darmstadtium 110 Ds [271]	roentgenium 111 Rg [272]	copernicium 112 Cn [285]	unnilium 113 Uut [284]	flerovium 114 Fl [289]	unseptennium 115 Uup [288]	livermorium 116 Lv [292]	unnonium 117 Uus [293]	unbinilium 118 Uub [294]								
	lanthane 57 La 138,90547	cérium 58 Ce 140,116	praseodyme 59 Pr 140,90765	néodyme 60 Nd 144,242	prométhium 61 Pm [144]	samarium 62 Sm 150,36	europium 63 Eu 151,964	gadolinium 64 Gd 157,25	terbium 65 Tb 158,92535	dysprosium 66 Dy 162,50	holmium 67 Ho 164,93032	erbium 68 Er 167,259	ytterbium 69 Yb 168,93421	lutécium 70 Lu 173,04													
	actinium 89 Ac [227]	thorium 90 Th 232,03806	protactinium 91 Pa 231,03688	uranium 92 U 238,02891	néptunium 93 Np [237]	plutonium 94 Pu [244]	américium 95 Am [243]	curium 96 Cm [247]	berkelium 97 Bk [247]	californium 98 Cf [251]	esboium 99 Es [252]	fermium 100 Fm [257]	mendélévium 101 Md [258]	nobélium 102 No [259]	lawrencium 103 Lr [262]												
	métaux alcalins	alcalino-terreux	lanthanides	actinides	métaux de transition	métaux pauvres	métalloïdes	non-métaux	halogènes	gaz nobles						primordial	désintégrant d'autres éléments		synthétiques								

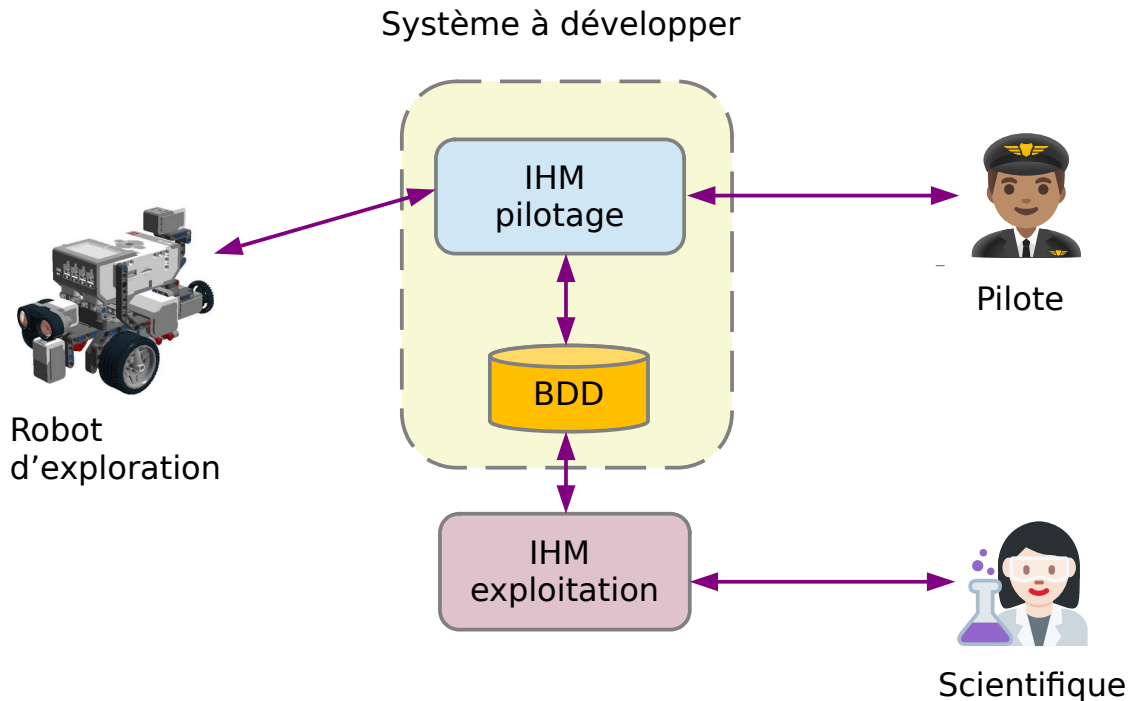
Bobarium (119)

Ce métalloïde ne se trouve sur terre que dans les régions naturellement radioactive et donc dangereuses pour l'exploration. Plus précisément, on le trouve dans le sous-sol de grottes où une fumée épaisse empêche l'utilisation de caméra traditionnelle.

L'idée serait donc de piloter à distance un **robot d'exploration** équipé d'un capteur de Bobarium pour cartographier une zone donnée. Il serait équipé d'un capteur ultra-son en remplacement d'une caméra pour visualiser son environnement comme ce que peuvent faire les chauves souris (les ultra-sons sont utilisables dans les épaisses fumées contrairement aux ondes lumineuses)

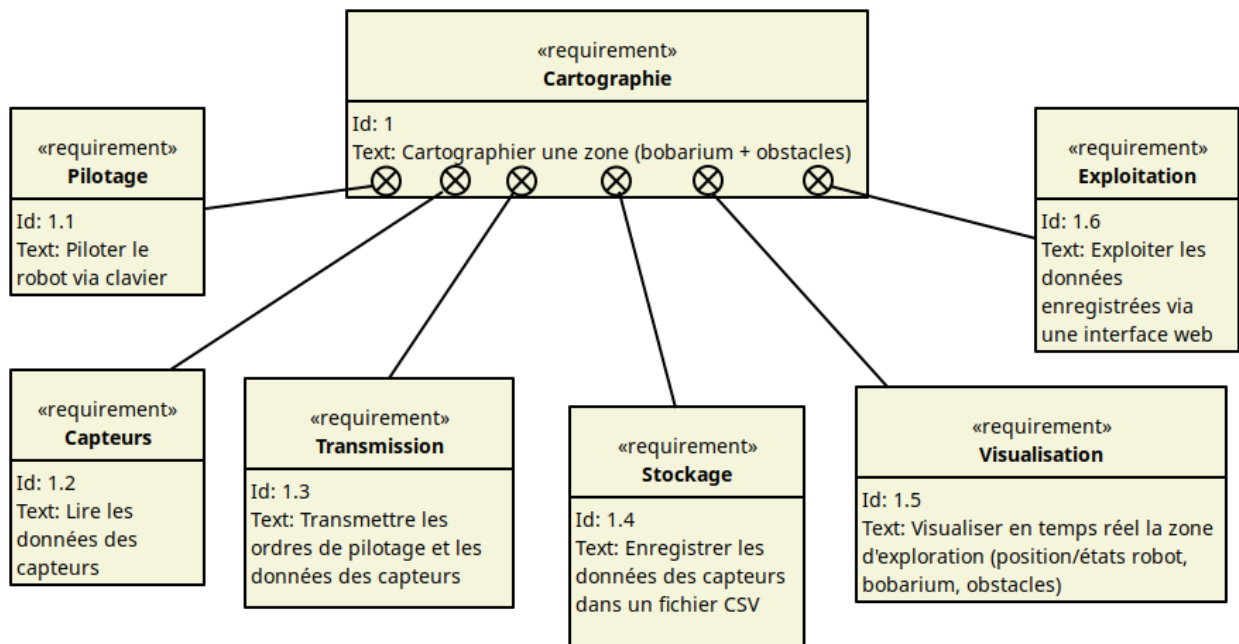
2.2 - Objectifs

Pouvoir **piloter** à distance un robot en évitant les obstacles et **recupérer** les mesures de présence de Bobarium effectuées sur le terrain pour les **enregistrer** dans une base de données.



2.3 - Expression du besoin

Ci-dessous le diagramme SysML des exigences. Chaque bloc exprime une fonctionnalité à réaliser.



2.4 - Contraintes

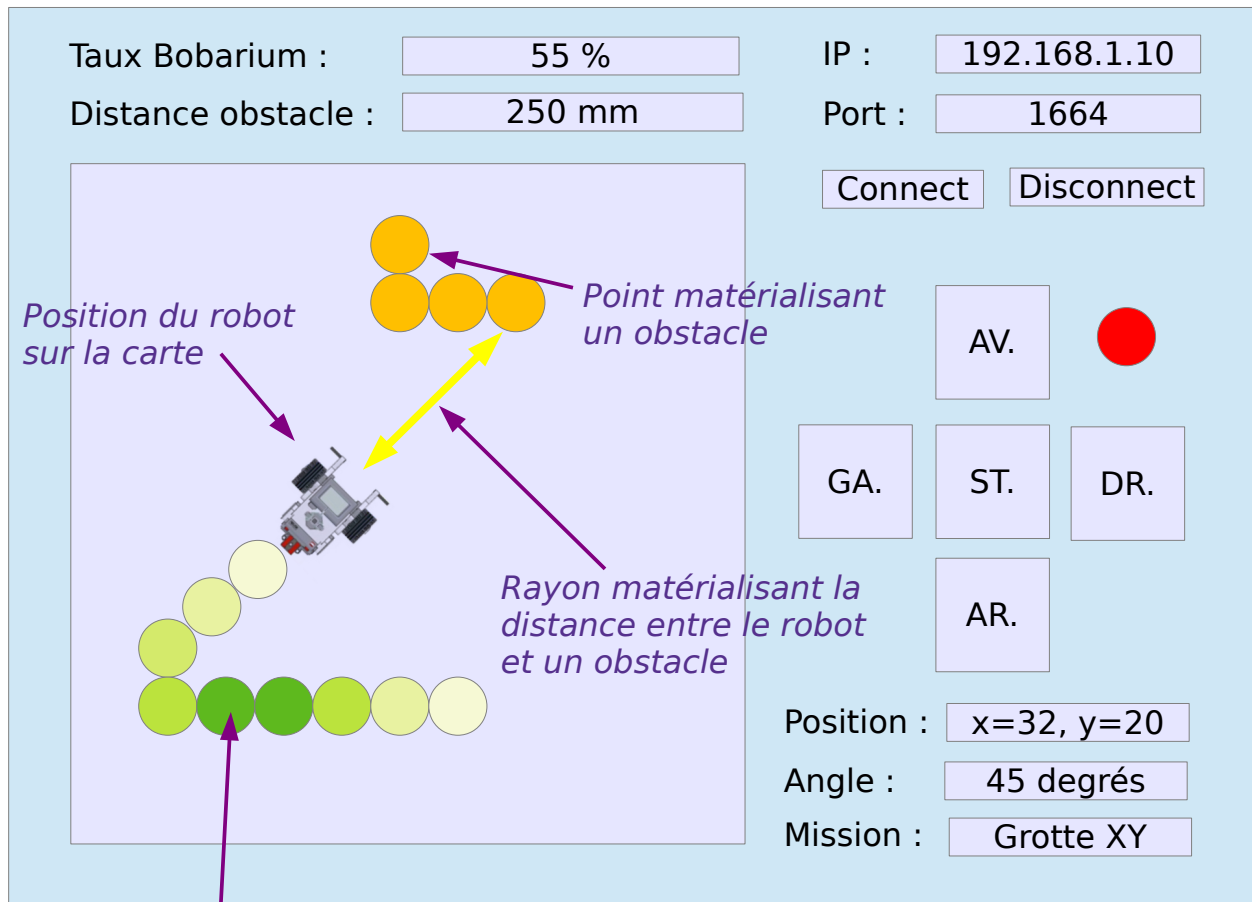
Afin de ne pas surcharger le diagramme des exigences, les contraintes et précisions liées à chaque exigence sont rappelées dans le tableau ci-dessous.

Exigence	Contraintes/précisions
1.1 Piloteage	Le pilotage du robot se fera via un smartphone. On retrouvera les ordres de pilotage associés à des boutons (1er niveau) ou les accéléromètres (2nd niveau) <ul style="list-style-type: none">• avancer/reculer (bouton ou/et accéléromètres)• tourner gauche/droite (bouton ou/et accéléromètres)• stopper (bouton ou/et accéléromètres)• lever/baisser barre (bouton)• allumer/éteindre LEDS (bouton)• débiter/arrêter enregistrement données CSV
1.2 Capteurs	Le robot renverra au système différentes informations <ul style="list-style-type: none">• le taux de Bobarium mesuré dans le sol (en pourcentage)• la distance robot/obstacle (en mm)• la position angulaire du robot (en degrés)• la position angulaire des 2 roues (en degrés)
1.3 Transmission	La transmission des ordres de pilotage et des informations se fera via une liaison réseau wifi. <ul style="list-style-type: none">• le protocole HTTP sera privilégié ainsi que le format JSON pour l'encodage des informations• les données issues des capteurs du robot devront arriver toutes les secondes environ (voir 1.5)• idéalement cette liaison sera chiffrée pour éviter toute fuite d'informations (2nd niveau)
1.4 Stockage	Le stockage des informations s'effectuera dans un fichier CSV sur le robot de manière à ce qu'en fin de mission, on puisse récupérer ce fichier pour traitement par l'équipe scientifique. Le fichier est nommé selon le nom de la mission, la date et l'heure de début d'enregistrement
1.5 Visualisation	Le pilote du robot pourra visualiser en temps réel (délai=1s) sur une IHM la zone d'exploration avec : <ul style="list-style-type: none">• la position du robot (à calculer à partir des données des capteurs : xr, yr, angle)• les obstacles (xo, yo)• les points d'échantillons des taux de Bobarium (xs, ys, couleur) Voir le détail de l'IHM attendue au 2.5
1.6* Exploitation	Via une IHM web

(*) : non demandées dans ce mini-projet

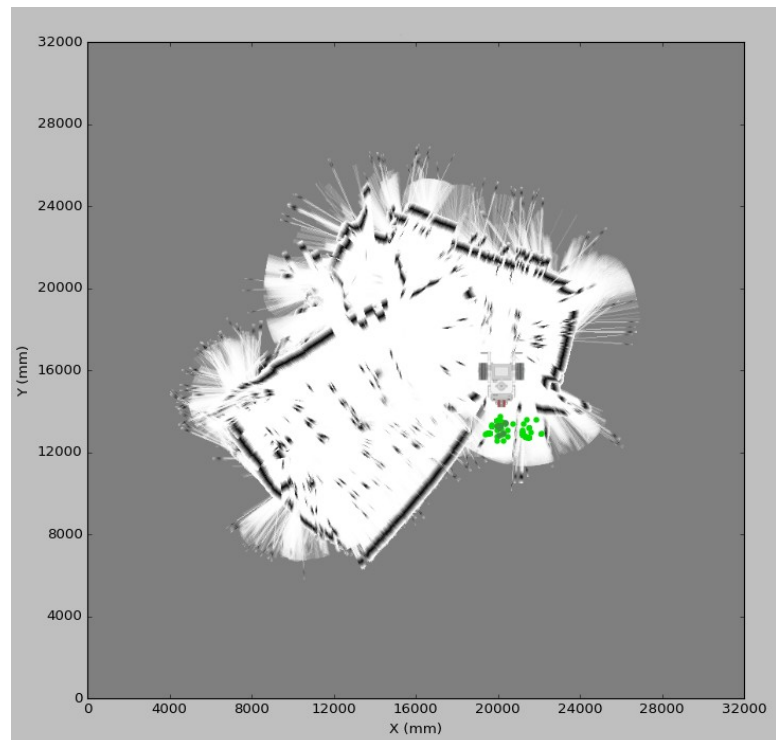
2.5 - IHM

Proposition d'IHM de pilotage à adapter selon l'écran (pc ou smartphone) :



Points de mesure Bobarium: plus le taux est important plus la couleur est foncée

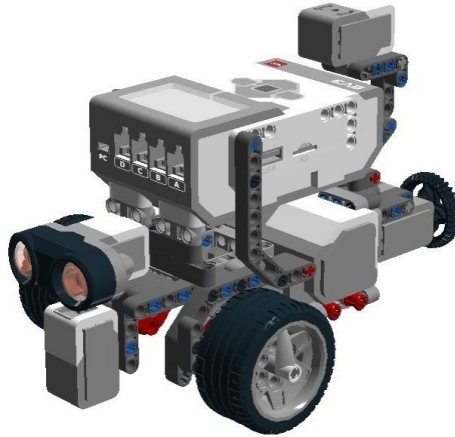
Résultat possible en dessinant chaque rayon se terminant par un point sombre pour matérialiser les obstacles



2.6 - Ressources

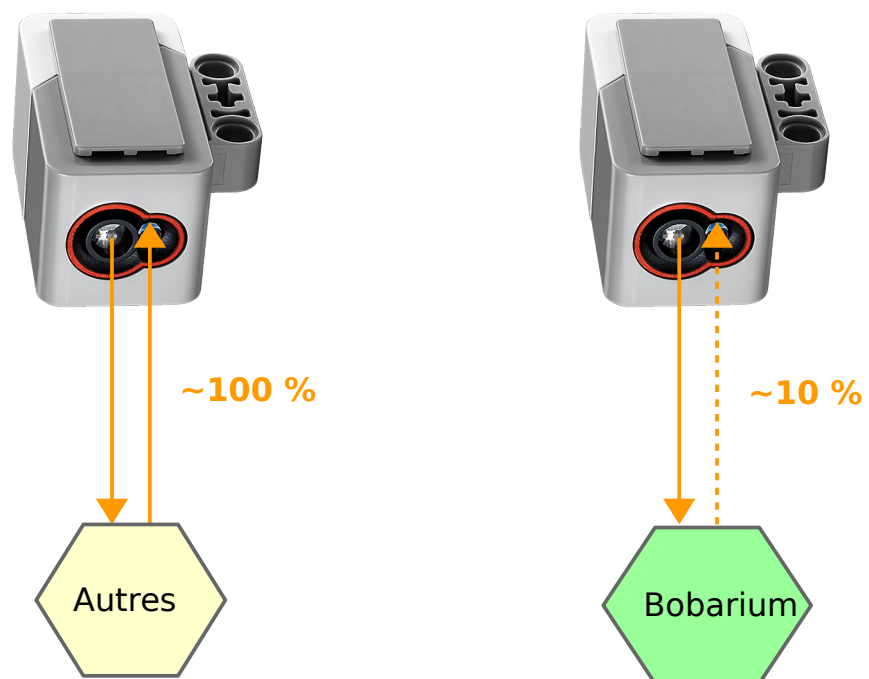
2.6.1 - Système d'exploitation

Le robot utilisé sera un **EV3 Mindstorm** de chez LEGO équipé d'un système d'exploitation Linux Debian 9 (Stretch) et de l'environnement Micropython Pybricks. De plus, un dongle wifi vous permettra de pouvoir piloter à distance et récupérer les mesures effectuées.



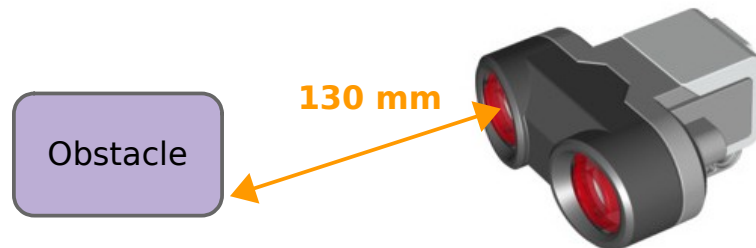
2.6.2 - Capteur de Bobarium

Le robot portera un capteur de Bobarium. Ce capteur envoie un signal spécial dans le sol. Si celui-ci revient à 100 % c'est qu'il n'y a pas de Bobarium en dessous. Si le signal ne revient que faiblement par exemple à 10 %, c'est que du Bobarium se trouve en dessous et qu'il a absorbé une grande partie de l'énergie du signal d'origine.



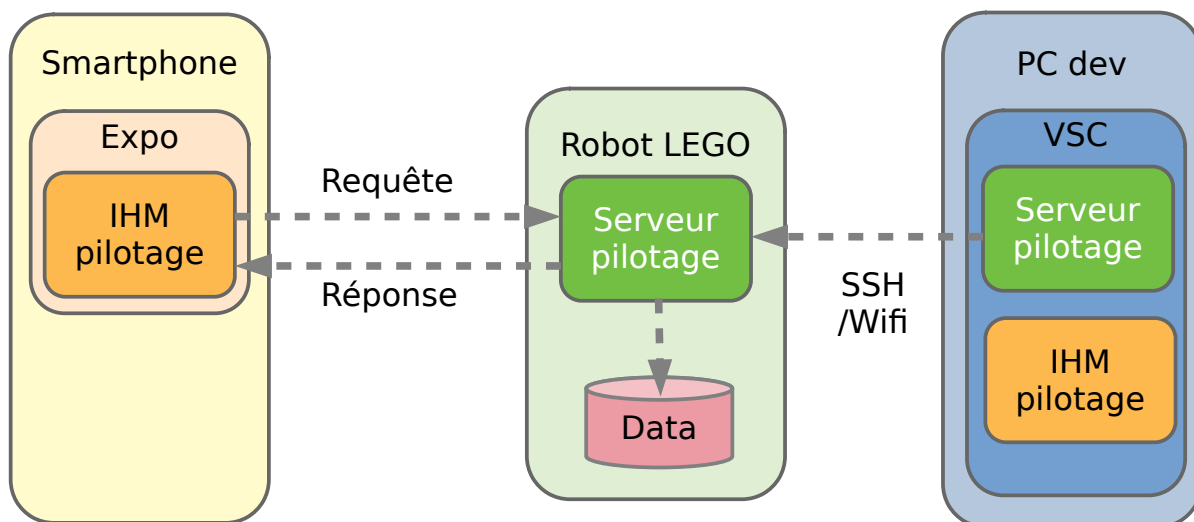
2.6.3 - Capteur ultrason

Le robot utilisera également un capteur ultrason. Ce dernier permettra de mesurer la distance entre d'éventuels obstacles et le robot. Le pilote à l'instar d'une chauve souris pourra ainsi éviter les obstacles présents sur la zone de recherche.



2.6.4 - Logiciels

Ci-dessous le synoptique global de votre environnement de développement



Le PC de développement du serveur de pilotage Python utilisera :

- un éditeur texte, par exemple **Visual Studio Code**
- l'extension **Lego EV3 Mindstorm EV3 Micropython** pour développer à distance avec le robot
- l'outil **ssh** pour déboguer le programme à distance sur le robot

Le PC de développement de l'IHM utilisera :

- un éditeur texte, par exemple **Visual Studio Code**
- L'application **Expo.go**
- Le framework **React Native**

3 - Évaluation

3.1 - Travail à rendre

A la fin du mini-projet il vous faudra rendre par équipe, via Moodle un fichier texte contenant le lien vers le dépôt github de votre projet. Ce dernier contiendra :

- un dossier **robot** avec le code source Python du programme serveur embarqué dans le robot
- un dossier **ihm** avec le code source React Native de l'IHM sans le dossier « node_modules » (utiliser un fichier .gitignore)
- un fichier **README.md** avec la documentation du projet
 - *qui a fait quoi dans l'équipe ?*
 - *comment installer et utiliser le produit ?*
 - *comment fonctionne le protocole réseau ?*

Parce que c'est un mini-projet complexe, vous devrez prioriser vos tâches dans votre travail.

Côté **robot**, on pourra ordonner les tâches de manière incrémentale :

1. un serveur TCP simple qui répond à une requête HTTP (déjà vu en TP)
2. idem précédent mais qui retourne de fausses valeurs en JSON
3. idem précédent mais qui retourne les vraies valeurs issues des capteurs
4. idem précédent mais qui reçoit et décode la requête de pilotage
5. idem précédent mais qui exécute les ordres décodés
6. idem précédent mais qui chiffre et déchiffre les messages
7. idem précédent mais qui travaille en multi-tâches
8. idem précédent mais qui synchronise les tâches (Semaphore ou Queue)

Côté **IHM**, on pourra ordonner les tâches de manière incrémentale :

1. une IHM avec des éléments inactifs
2. idem précédent mais qui effectue une requête HTTP au robot lors de l'appui sur un bouton
3. idem précédent mais qui reçoit/analyse/affiche la réponse (valeurs des capteurs)
4. idem précédent mais qui récupère les données toutes les 1s
5. idem précédent mais qui utilise les accéléromètres
6. idem précédent mais qui chiffre et déchiffre les messages
7. idem précédent mais qui calcule/dessine la position du robot, Bobarium, obstacles

3.2 - Critères d'évaluations

Voir la grille d'évaluation donnée à titre indicatif dans Moodle. Elle est susceptible d'évoluer...

CDC :: Pilotage (1.1)	Aucun pilotage possible 0 points	Une partie des ordres possibles depuis les boutons 1 points	Tous les ordres sont fonctionnels en utilisant les boutons 2 points	Le robot est pilotable avec les boutons et en utilisant les accéléromètres 3 points
CDC :: Capteurs (1.2)	Aucun capteur accessible 0 points	Une partie des capteurs est accessible 1 points	Tous les capteurs sont accessibles 2 points	Tous les capteurs sont accessibles et la gestion des capteurs en erreur est prise en compte 3 points
CDC :: Transmission (1.3)	Aucun échange entre le robot et le smartphone 0 points	Echanges partiels en qualité (HTTP+JSON?) ou en quantité (requête+réponse) 1 points	Echanges complets 2 points	Les échanges sont complets et chiffrés/déchiffrés 3 points
CDC :: Stockage (1.4)	Aucune données enregistrées sur le robot 0 points	Quelques données enregistrées dans un fichier CSV 1 points	Toutes les données sont enregistrées dans un fichier CSV 2 points	Le nom du fichier CSV est horodaté en fonction de l'ordre d'enregistrement reçu (nom mission + jour + heure) 3 points
CDC :: Visualisation (1.5)	Aucune information affichée 0 points	Informations affichées textuellement 1 points	Une partie des informations affichées graphiquement 2 points	Toutes les informations sont affichées textuellement et graphiquement sur la carte 3 points
PROJ :: délais	Très en retard 0 points	Un peu en retard 1 points	Dans les délais 2 points	
PROJ :: dépôt github	Dépôt quasiment ou pas utilisé 0 points	Dépôt peu ou mal utilisé 1 points	Dépôt bien organisé avec de nombreux commits voire branches 2 points	
PROJ :: Scrum	Aucune gestion de projet 0 points	Gestion de projet agile partielle 1 points	Gestion de projet agile avec Scrum (US+backlog+sprints+burndown) 2 points	
CODE :: organisation	Organi-quoi ? 0 points	Un semblant d'organisation émerge 1 points	Bien organisé (POO, découpage) 2 points	
CODE :: commentaires	Aucun commentaire ou presque 0 points	Pas assez de commentaires 1 points	Code bien commenté 2 points	
CODE :: multi-tâches	Pas multi-tâches 0 points	Multi-tâches mis en oeuvre mais pas fonctionnel 1 points	Multi-tâches mis en place et fonctionnel 2 points	
DOC :: README.md	Aucune documentation 0 points	Documentation présente mais défailante soit en qualité, soit en quantité 1 points	Documentation présente en qualité et quantité 2 points	

5 - Éléments de codage

Tous les éléments de codage donnés ci-dessous ne forment qu'une trame que vous pouvez suivre ou pas. Les étudiants plus à l'aide devraient pouvoir complètement s'en passer.

5.1 - Pour la communication

5.1.1 - Protocole

Les protocoles « texte » sont souvent les plus simples à mettre en œuvre. Le tableau ci-dessous situe le protocole BOBARIUM par rapport au modèle OSI :

<i>Couche OSI (numéro)</i>	<i>Couche OSI (nom)</i>	<i>Description</i>
7	application	Protocole BOBARIUM
6	présentation	Protocole BOBARIUM
5	session	Protocole BOBARIUM
4	transport	TCP
3	réseau	IP
2	liaison	Wifi + Ethernet
1	physique	Wifi + Ethernet

Afin de ne pas réinventer la roue, on choisira de s'appuyer sur le protocole **HTTP** et l'encodage **JSON** pour la mise en œuvre du protocole BOBARIUM.

Une requête en provenance du smartphone pourra ainsi être encodée ainsi et envoyée à l'aide de la fonction **fetch()** :

```
{
  "mouvement": "AVANCER",
  "vitesse": 100,
}
```

Et la réponse du serveur embarqué dans le robot pourra être encodée comme suit :

```
{
  "angle": 241,
  "distance": 78,
  "bobarium": 99,
  "moteur_gauche": 360,
  "moteur_droite": 360
}
```

En y intégrant l'en-tête HTTP cela pourrait donner :

```
HTTP/1.1 200 OK\r\n
Content-Length: 90\r\n
Content-Type: application/json; charset=UTF-8\r\n
\r\n
{"angle": 241, "distance": 78, "bobarium": 99, "moteur_gauche":
360, "moteur_droite": 360}
```

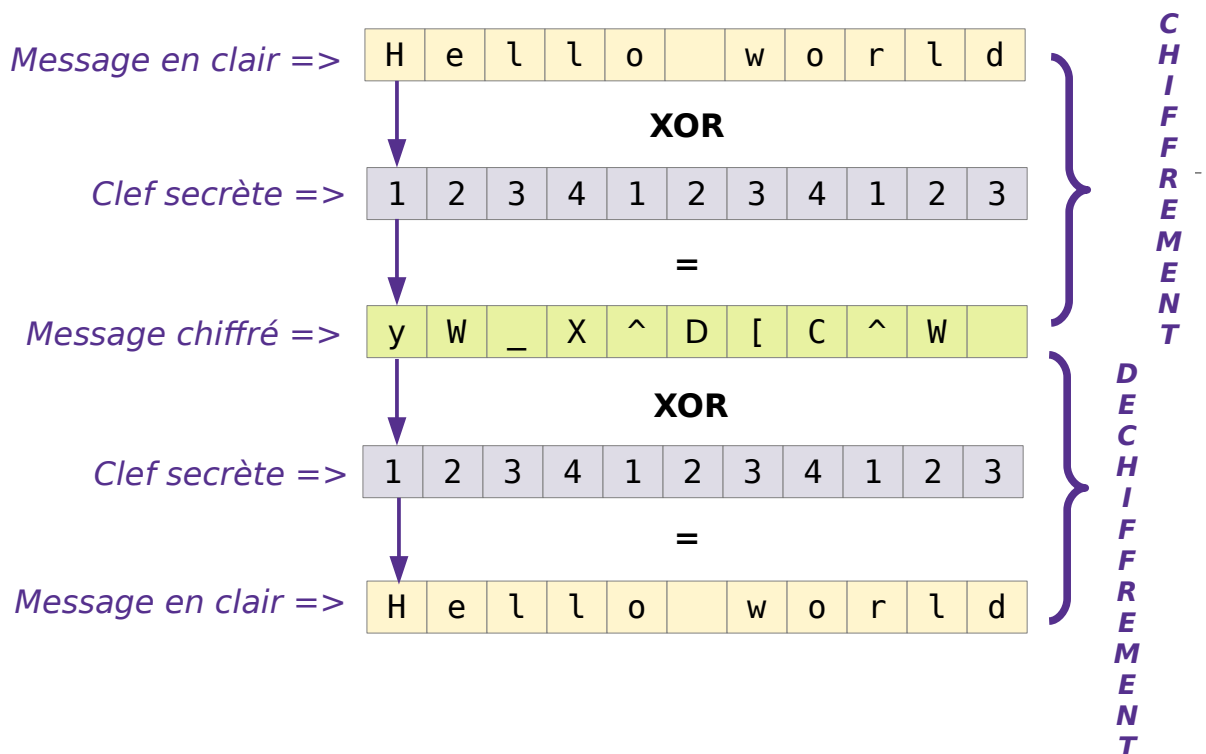
5.1.2 - Sécurité

Le chiffrement XOR est relativement aisé à mettre en œuvre en C++ (opérateur ^). Le principe est le suivant, pour chaque caractère :

```
H => 0x48 en ASCII => 0100 1000
                        XOR (^)
1 => 0x31 en ASCII => 0011 0001
                        -----
                        0111 1001 => 0x79 en ASCII => y
```

1) code ASCII du message en clair :

2) code ASCII de la clef secrète : 1



L'opération de déchiffrement est la même que l'opération de chiffrement.

5.2 - Pour l'IHM

5.2.1 - Les outils React Native

Lors d'un précédent TP, vous avez eu l'occasion de découvrir l'environnement de développement React Native. Voici les composants qui pourront vous être utiles pour ce mini-projet :

- ➔ La fonction **fetch()** permettra de communiquer en réseau avec le robot de manière asynchrone.
- ➔ Le composant **react-native-svg** ou **react-native-canvas** devraient vous permettre de dessiner sur un écran
- ➔ La fonction **setTimeout()** vous permettra d'envoyer une requête pour récupérer les valeurs du robot à intervalle régulier.

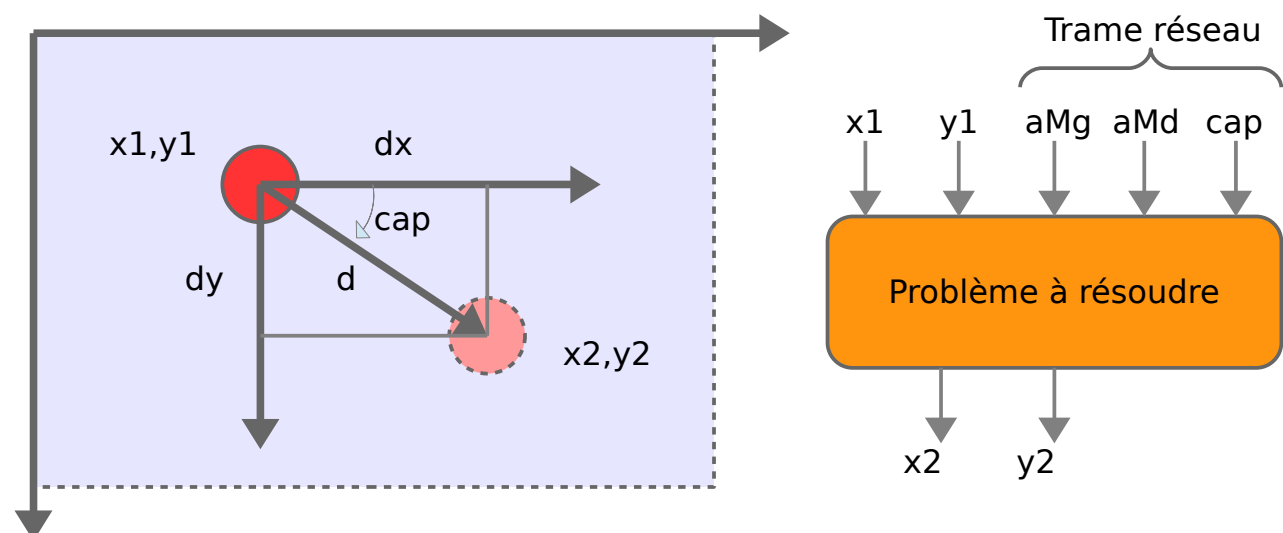
5.2.2 - Transformer les valeurs en carte

Une première approche simplificatrice consistera à considérer que le robot ne peut effectuer que certains mouvements :

- avancer ou reculer (les 2 moteurs avancent à la même vitesse)
- tourner sur place (les 2 moteurs tournent en sens inverse)

Ainsi on évite les mouvements en arc de cercle plus difficile à calculer et tracer.

Voici une modélisation dans le plan de la position du robot



Détails des variables :

- $x1, y1$: position actuelle du robot
- $x2, y2$: nouvelle position du robot
- aMg, aMd : respectivement angle du moteur gauche et droit
- cap : cap vers où « regarde » le robot

Algorithme de calcul de la nouvelle position du robot (idem pour obstacle)

```
# Ce qui est connu au début
diametre_roue = 56
x1 = 0
y1 = 0

# Ce qui est récupéré de la trame issue du robot
aMg = # angle du moteur gauche (issu du tachymètre)
aMd = # angle du moteur droite (issu du tachymètre)
cap = # cap du robot (issu du gyroscope)

# Ce qui est calculé
perimetre_roue = (2 * 3.14 * (diametre_roue / 2))
aMgMd = ((aMg - aMg_old) + (aMd - aMd_old)) / 2
d = (aMgMd * perimetre_roue) / 360
dx = d * cos(cap)
dy = d * sin(cap)
x2 = x1 + dx
y2 = y1 + dy

# Mémoire des angles des 2 roues pour prochain calcul
aMg_old = aMg
aMd_old = aMd
```

La couleur du nouveau point est fonction du taux de Bobarium. Plus il est sombre, plus il y a de Bobarium dans le sol.