Programme ton robot en C

1/ L'ÉVEIL D'OKTAL

Jean-Michel Hautbois

L’Éveil d’Oktal

L'Éveil d'Oktal

Programme ton robot en C

Jean-Michel Hautbois

YOSELI
.press

Illustrations et couverture : Clément Berneron

Édition et impression :
BoD · Books on Demand GmbH, Überseering 33, 22297 Hambourg, Allemagne

ISBN: 978-2-3227-8059-4
Dépôt légal : mai 2026

Loi n° 49-956 du 16 juillet 1949 sur les publications destinées à la jeunesse

https://press.yoseli.org/

Remerciements

À Claire, ma compagne de tant d'années. À côté de qui tout devient possible, même les projets entrepris en silence.

À mes parents, qui ont déposé entre mes mains le premier ordinateur et n'ont jamais pensé à me demander de le refermer. À mes frères, Guillaume et Flavian, et un merci particulier à Flavian pour son regard sur le manuscrit, précieux pour affiner le ton et le rythme.

À Léandre, Armand et Élise, mes trois étincelles. Une pensée particulière pour Léandre, mon tout premier relecteur et testeur, qui a traqué les passages trop compliqués avec la rigueur d'un ingénieur en herbe. Armand et Élise, promis, les prochains passeront entre vos mains. Et à toute la tribu qui gravite autour, neveux, nièces, beaux-parents, cousins, les autres : vous n'avez rien demandé, et c'est aussi pour vous que ce livre existe.

À Clément Berneron, qui a donné à Oktal ses yeux cyan et ce regard curieux qui traverse toutes les pages. Sans son trait, le robot serait resté une idée sur une feuille.

À John Maloney, co-créateur de Scratch et créateur de MicroBlocks, dont le travail a inspiré des générations de jeunes programmeurs. C'est en voyant des collégiens découvrir la joie de programmer avec des blocs colorés que l'idée de ce livre est née : leur offrir un pont vers le code réel.

Enfin, à toutes celles et ceux qui font vivre le logiciel libre et l'open source. Compilateurs, noyaux, RTOS, bibliothèques, outils de build : la plupart sont écrits par des gens qui partagent leur travail sans rien demander. Ce livre n'existerait pas sans Zephyr, sans la micro :bit Educational Foundation, sans le projet GNU, sans la grande famille Linux dans laquelle je navigue depuis plus de vingt ans, et sans mille briques anonymes que personne ne remercie jamais. Si tu lis ces pages, tu tiens le fruit de ce partage. Le libre, c'est ça : ouvrir le capot pour que d'autres puissent construire la suite.

Jean-Michel Hautbois — Acigné, mai 2026

À toi qui ouvres ce manuel

Ce document a été retrouvé dans les Archives de la Friche de Silicium, secteur 7-G, sous une pile de circuits imprimés.

Coordonnées d'origine : zone grise. Date d'archivage : tampon illisible. Époque : indéterminée.

La **Friche de Silicium**, c'était le Centre de Recherche en Robotique Autonome. On y fabriquait les **Unités OKT**, douze robots compacts numérotés de OKT-1 à OKT-12, conçus pour explorer et entretenir ce que les humains ne voulaient plus toucher.

Puis quelque chose s'est propagé dans les systèmes. On l'appelle **le Signal**. Les robots ont commencé à dysfonctionner, puis à s'éteindre un par un. Les humains ont évacué. Le centre a été scellé et oublié.

Le manuel que tu tiens entre les mains n'est pas neuf. Quelqu'un l'a possédé avant toi. Quelqu'un qui a pris le temps d'annoter les marges, de souligner des passages, de griffonner des avertissements.

Ces annotations portent un nom : les *marginalia*. Au Moyen Âge, les moines copistes y laissaient des notes dans les marges des manuscrits. Des messages d'un lecteur à l'autre, traversant les siècles.

Nous ne savons pas qui a laissé ces *marginalia*. Les Archives l'appellent simplement **l'Archiviste**. Ses notes sont parfois obscures, parfois salvatrices. Certaines mentionnent une mystérieuse **OKT-7** et des coordonnées étranges. D'autres contiennent des codes hexadécimaux que personne n'a encore déchiffrés.

Une chose est sûre : l'Archiviste connaissait ces robots mieux que personne. Il ou elle les a réparés, documentés, et a tenu assez longtemps pour laisser ces traces.

Lis ses notes. Certaines t'éviteront de faire les mêmes erreurs que lui.

— Les Archives de la Friche

Table des matières

Table des figures

Liste des tableaux

Introduction

Bienvenue dans *L'Éveil d'Oktal* !

Tu as déjà collé des blocs ensemble. Des variables, des boucles, un *au démarrage* qui déclenche la suite, un *toujours* qui répète en boucle. Avec MakeCode, Scratch ou MicroBlocks, tu sais déjà faire quelque chose qui marche.

Maintenant, on passe à la vraie machine. Le langage **C** te donne un contrôle direct sur la mémoire, le temps, et le matériel. C'est le langage qui fait tourner ton micro-onde, les voitures, les satellites, le noyau Linux. Et bientôt, ton robot.

Pourquoi le C ?

Le langage C est utilisé partout : dans les téléphones, les voitures, les fusées, et bien sûr... les robots ! C'est un langage *bas niveau*, ce qui veut dire qu'il parle presque directement au processeur.

Et pourquoi pas Rust, Python ou autre ?

Tu as peut-être entendu parler de **Rust**, un langage moderne qui promet plus de sécurité que le C. Ou de **MicroPython**, qui permet de programmer des microcontrôleurs en Python. Alors pourquoi apprendre le C ?

- **Zephyr est écrit en C** : le système d'exploitation que nous utilisons est entièrement en C. C'est son langage natif, avec une documentation complète et des milliers d'exemples.
- **Le C est universel** : le noyau Linux (qui accepte aussi Rust depuis 2022), les firmwares, 90% des systèmes embarqués sont en C. Apprendre le C, c'est acquérir une compétence qui te servira partout.
- **Rust sur Zephyr est récent** : le support officiel existe depuis 2024, mais il reste un module optionnel limité à certaines cartes. La documentation et les exemples sont encore rares comparés au C.
- **MicroPython a ses limites** : excellent pour prototyper, mais moins adapté au contrôle temps réel précis dont un robot a besoin.

Une fois que tu maîtriseras le C, tu pourras facilement apprendre Rust ou d'autres langages. Les concepts fondamentaux (pointeurs, mémoire, compilation) seront déjà acquis !

Pourquoi ce livre ?

Il existe beaucoup de ressources pour apprendre à programmer. Alors pourquoi ce livre ?

Un chemin unique : des blocs vers le C. La plupart des livres font passer de MakeCode à Python (MicroPython). C'est plus facile, mais ça ne t'apprend pas comment fonctionne vraiment un ordinateur. Le C, lui, te montre tout : la mémoire, les registres, le temps d'exécution. C'est le langage des systèmes embarqués professionnels : téléphones, voitures, satellites, robots industriels.

Un vrai système d'exploitation. Pas d'Arduino ici, mais **Zephyr RTOS**, utilisé dans l'industrie. Device Trees, configuration Kconfig, drivers. Des compétences directement transférables vers Linux embarqué ou l'automobile.

Comprendre le matériel, pas juste l'utiliser. Ce livre ne se contente pas de te dire « appelle cette fonction pour avancer ». Tu vas voir *comment* le micro:bit parle au contrôleur moteur via I2C, *pourquoi* on utilise du PWM pour varier la vitesse, *ce qui se passe* au niveau des bits et des registres.

De la robotique, pas juste des LEDs. Contrôleur PID, odométrie, cartographie et navigation autonome. À la fin, ton robot explorera une pièce et construira sa propre carte.

Ce livre est né d'une frustration

Les tutoriels en ligne sautent des étapes. Les livres sur le C parlent de programmes sur PC. Les documentations de Zephyr sont écrites pour des ingénieurs. Entre les blocs colorés et le code professionnel, il y a un fossé que personne ne t'aide à franchir.
Ce livre comble ce fossé. Pas à pas. Sans supposer que tu sais déjà. En expliquant le *pourquoi* autant que le *comment*.

Ce que tu vas apprendre

- Écrire des programmes en C
- Comprendre les variables, boucles, fonctions et structures
- Programmer un vrai robot : le Maqueen Plus V3
- Utiliser Zephyr RTOS, un système d'exploitation pour robots
- Faire explorer une pièce à ton robot de manière autonome !

Le matériel

Tu vas travailler avec :

- Un robot **Maqueen Plus V3** de DFRobot
- Une carte **BBC micro:bit V2**
- Un capteur **LIDAR** pour détecter les obstacles
- Un ordinateur avec **Zephyr RTOS**

Pourquoi le Maqueen Plus V3 ?

Le Maqueen Plus V3 a un atout majeur : ses **encodeurs de roue**. Ces capteurs comptent les rotations des roues, ce qui permet de mesurer précisément la distance parcourue et la vitesse. Sans encodeurs, impossible de faire de l'odométrie ou un contrôle PID correct.
D'autres robots micro:bit fonctionnent pour les premiers chapitres. Pour la navigation autonome, les encodeurs sont indispensables. Voir la section E.5 pour les alternatives.

Structure du livre

Ce livre est divisé en quatre parties progressives :

Partie 1 — Les fondations Les bases du C : variables, conditions, boucles. Tu peux suivre ces chapitres avec un simulateur, sans robot physique.
Partie 2 — Construire des programmes Fonctions, pointeurs, tableaux, structures. Toujours possible sans robot.
Partie 3 — Programmer le robot On passe aux choses sérieuses ! Tu découvres Zephyr, le matériel, l'I2C, les LEDs. Un robot micro:bit est nécessaire.
Partie 4 — Robotique autonome Odométrie, contrôle PID, cartographie, navigation. Les **encodeurs sont indispensables** pour cette partie.

Tu peux donc commencer sans robot et en acquérir un avant la partie 3. Mais si tu veux aller jusqu'à la navigation autonome, choisis un robot avec encodeurs dès le départ !

Comment utiliser ce livre

Chaque chapitre suit le même plan :

1. **Objectifs** : ce que tu vas apprendre
2. **Rappel blocs** : le lien avec ce que tu connais
3. **Le concept en C** : la nouvelle syntaxe
4. **Sur le robot** : mise en pratique
5. **Exercices** : pour t'entraîner (★ facile, ★★ moyen, ★★★ difficile)
6. **Mini-projet** : un défi à relever

Le Terminal de l'Ingénieur (Solutions et Bonus)

Parce qu'un bon ingénieur ne travaille jamais sans documentation, ce livre est accompagné d'un site web immersif : **Le Terminal OktalOS**.

C'est là que tu trouveras :

- **Les solutions complètes** de tous les exercices (fichiers `.c`)
- **Les errata** et mises à jour du livre
- **Des archives cryptées** qui révèlent l'histoire cachée de la Friche

Accès au Terminal

Connecte-toi à l'adresse suivante :

`https://friche.oktalos.org`

Note : Le terminal est protégé. Pour accéder aux archives sécurisées, tu devras entrer les **Codes de Synchronisation** que tu trouveras dissimulés dans ce livre (ex : `0xXXXXXXXX`). Garde l'œil ouvert !

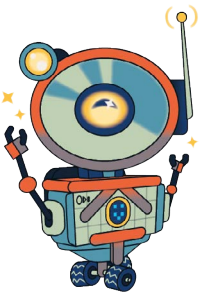
Le Playground C (Tester ton code)

Tu n'as pas encore de robot sous la main ? Pas de problème ! Pour tester les concepts de base du langage C (variables, conditions, boucles, fonctions...), tu peux utiliser notre **Playground en ligne** :

`https://code.oktalos.tech`

Ce playground te permet d'écrire, compiler et exécuter du code C directement dans ton navigateur. Il est parfait pour les exercices des chapitres 1 à 9 (avant de passer au robot).

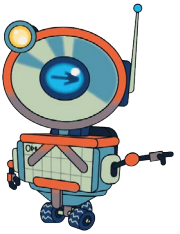
L'Unité Oktal



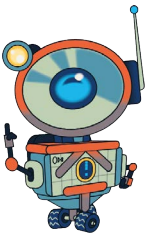
Devant toi, une **Unité OKT-8**. Un robot compact, oublié dans la Friche de Silicium. Ses circuits sont intacts, mais sa mémoire a été effacée. Il ne bouge pas.
Tu vas lui réapprendre à rouler. À voir. À comprendre où il est.
C'est à toi de lui redonner vie.

Au fil des chapitres, Oktal t'accompagnera dans ton apprentissage. Quand il apparaît, c'est qu'il a quelque chose à te montrer :

« Souviens-toi… »
un point déjà vu réapparaît

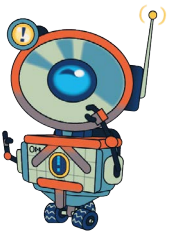


« Regarde ici ! »
quelque chose d'important



« Aha ! »
moment de découverte

« À toi de coder ! »
exercice à essayer



« Attention ! »
erreur courante ou piège

« Bien joué ! »
tu as réussi quelque chose

« Piège courant… »
ne t'y laisse pas prendre

« Réfléchissons… »
pause pour comprendre

> *Première règle : ne jamais s'attacher à une Unité Oktal. Deuxième règle : on finit toujours par s'attacher.*

Prêt ? C'est parti !

PROLOGUE

[OK-8]

BIENVENUE, TECHNICIEN.

Friche de Silicium, Hangar Principal

La porte grince. Tu retiens ton souffle.

Devant toi, le hangar s'étend dans la pénombre. Des rayons de lumière filtrent à travers le toit percé, dessinant des colonnes dorées dans la poussière. Partout, des étagères rouillées, des câbles qui pendent, des établis couverts d'outils oubliés.

La végétation a commencé à envahir les lieux. Mousse sur le béton. Lierre sur les murs.

Tes pas résonnent dans le silence. Prudemment, tu avances...

* * *

Entre deux boîtes poussiéreuses, sur une étagère, un livre attire ton regard.

Un vieux manuel technique à la couverture usée. Des annotations manuscrites couvrent les marges. Une écriture serrée, précise. Une main inconnue est passée ici avant toi, et a pris le temps de noter ses découvertes, ses astuces, ses mises en garde.

L'Archiviste. C'est le seul nom qui revient dans ses notes.

En feuilletant les pages, tu vois l'encre appuyer de plus en plus fort. Certains mots sont soulignés trois fois. D'autres, raturés puis réécrits.

> *« Le Signal se propage. OKT-5 ne répond plus depuis ce matin. OKT-12 s'est arrêtée en pleine tâche. »*

Plus loin, une liste. Douze noms. Douze unités. Toutes barrées sauf une : OKT-7, entourée d'un point d'interrogation.

> *« OKT-7 aurait survécu plus longtemps. Elle aurait laissé des traces : des codes étranges, des messages cryptés. Personne ne sait où elle s'est arrêtée. »*

La dernière note est datée du jour de l'évacuation. Trois mots griffonnés à la hâte :

> *« Friche abandonnée. Fuyez. »*

* * *

Un reflet accroche ton œil.

Là, contre l'étagère du fond. Une forme. Petite, compacte. Un robot.

Il est couvert de poussière et de toiles d'araignée. Ses yeux sont éteints : deux ovales sombres sous un casque arrondi. Sur le dessus, des lettres à peine lisibles : **OKT-8**.

L'Unité 8.

À genoux devant elle, le manuel ouvert, tu cherches la page qui explique comment la démarrer. Là : un schéma du panneau de contrôle. Un cercle jaune sur le torse, avec un chiffre à l'intérieur.

Ta main hésite au-dessus du bouton. Tu pourrais refermer le manuel. Repartir. Laisser le hangar à sa poussière.

Tu appuies.

* * *

Un déclic. Un bourdonnement sourd.

Les yeux du robot clignotent une fois. Deux fois.

Puis s'illuminent d'un cyan doux.

OKT-8 lève la tête vers toi.

* * *

Pendant que le robot termine son démarrage, tu explores le reste du hangar.

Contre le mur du fond, des écrans alignés sur un long bureau métallique. De vieilles machines, comme celles qu'on voit dans les films d'époque. Des *terminaux* : des ordinateurs simples, juste un écran et un clavier, faits pour taper des commandes.

La plupart sont fissurés ou renversés. Mais l'un d'eux semble intact.

Tu t'approches. L'écran est noir, éteint. Un câble d'alimentation pend sous le bureau, sectionné net. Plus loin, une prise murale. Tu écartes la poussière, dénudes les fils avec précaution, et tu branches.

Un claquement sec.

L'écran s'éveille. Noir profond, puis des lignes de texte cyan apparaissent, une par une :

```
OktalOS v2.7.42 — Friche de Silicium

Bienvenue, Technicien.
```

Oktal te regarde. Il attend.

Partie I

Les fondations

Chapitre 1

De la programmation visuelle au C

Objectifs

À la fin de ce chapitre, tu sauras :
- Comprendre la différence entre les blocs visuels et le C
- Reconnaître la structure d'un programme C
- Écrire et compiler ton premier programme
- Afficher un message sur le terminal du robot

Tu viens d'un autre outil ?

Ce chapitre utilise des exemples MakeCode, mais les concepts sont identiques si tu viens de :
- **MicroBlocks** : blocs similaires, même logique
- **Scratch** : tu connais déjà les boucles et conditions
- **Open Roberta** : même principe de blocs colorés
- **EduBlocks** : tu es déjà proche du code Python !

L'annexe E présente ces outils en détail.

1.1 Rappel : la programmation par blocs

La programmation par blocs, c'est assembler des pièces colorées. MakeCode, MicroBlocks, Scratch : le principe est le même. Chaque couleur représente une catégorie (tableau 1.1).

Couleur	Catégorie	Exemples
Bleu	Base	`afficher texte`, `pause`
Rose	Entrée	`lorsque bouton A pressé`
Vert	Boucles	`toujours`, `répéter`
Cyan	Communication série	`série écrire texte`
Rouge	Variables	`définir variable`

Tab. 1.1 : Les catégories de blocs MakeCode

1.1.1 La structure d'un programme MakeCode

Tout programme MakeCode pour micro:bit a deux blocs principaux :
La figure 1.1 montre la structure de base d'un programme MakeCode.

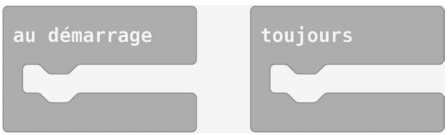


Fig. 1.1 : Structure de base MakeCode

- **au démarrage** : le code à l'intérieur s'exécute **une seule fois** quand le micro:bit démarre
- **toujours** : le code à l'intérieur se **répète indéfiniment**

Ce que tu sais déjà

En utilisant la programmation par blocs, tu as appris des concepts fondamentaux :

- Un bloc au démarrage (ou similaire) pour l'initialisation
- Un bloc toujours (ou *forever*) pour la boucle principale
- Les **instructions** s'exécutent dans l'ordre, de haut en bas
- On peut faire des **choix** avec si...alors

Tout cela existe aussi en C !

1.2 Le concept en C : du texte au lieu de blocs

En C, on n'utilise pas de blocs visuels. On écrit du **texte**. Ce texte s'appelle le **code source**.

1.2.1 La correspondance Blocs → C

Voici comment les blocs visuels se traduisent en C. La figure 1.2 montre un exemple MakeCode — le principe est identique pour tous les outils.

Fig. 1.2 : Programme MakeCode “Bonjour”

Le code C équivalent (mêmes couleurs pour la correspondance). Les deux premières lignes (`#include`) importent des outils : ignore-les pour l'instant, on les expliquera à la fin du chapitre.

```c
#include <zephyr/kernel.h>
#include <zephyr/sys/printk.h>

int main(void)
{
    printk("Hi!\n");
    return 0;
}
```

MakeCode	C
au démarrage	`int main(void) { ... }`
série écrire texte ”texte”	`printk("texte\n");`

Tab. 1.2 : Correspondance Blocs → C (base)

> *Les blocs cachent la complexité. Le C te la montre. C'est un cadeau, pas une punition.*

1.2.2 Ligne par ligne

Décortiquons ce programme. Ce code utilise Zephyr et ne fonctionne que sur le robot (pour tester sur PC, voir la section 1.5).