Remerciement

Tout d'abord, je remercie **Allah** le tout puissant de m'avoir donné le courage et la patience nécessaires à mener ce modeste travail à son terme.

Je remercie chaleureusement mon directeur de thèse, le **Dr BELARBI Mostefa**, qui a eu à cœur de développer mes recherches et a apporté toute sa contribution à la mise en œuvre de ce travail. Ses qualités humanistes et scientifiques, ses conseils et son enthousiasme ont été des leçons de père en fille, ils m'ont été très utiles durant cette année de master.

Je tiens également à remercier **Dr. BENGHENI Abdelmalek** et **Dr. ABID Khaled** qui ont accepté d'examiner mon travail, et exprime mon honneur ainsi que ma gratitude de les avoir comme jury dans ma soutenance.

Je tiens à remercier tous les membres du Laboratoire de Mathématiques et Informatique (LIM) de l'Université Ibn Khaldoun de Tiaret.

BOUYAHYAOUI HANANE

Dédicace

À mes parents, mes frères et mes sœurs, à toute ma famille et mes amis.

Merci.

- Randa

Résumé

Avec le développement des technologies dans l'industrie, le militaire, l'aéronautique, les télécommunications, etc., la demande de systèmes temps réel complexes et fiables augmente de plus en plus, grâce à des méthodes de développement avancées, pour des solutions informatiques respectant des contraintes de temps strictes, pour répondre aux besoins.

Dans ce travail, nous avons réalisé une similation de traitement d'échantillons biologiques en vue d'un analyse d'anatomie-pathologique ce dispositif transite par 4 étapes de base partant d'un échantillon biologique (tissu) pour se terminer avec une image de cellules visibles au microscope. le système conçu repose principalement sur le contrôle en temps réel de la synchronisation des différents traitements issus des différents sous-systèmes du process.

Mots clé : capteur, système embarqué temps réel, exécutif temps réel, méthode SA-RT, langage LACATRE, banc de test Anatomie-pathologique

Abstract

With the development of technologies in industry, military, aeronautics, telecommunications, etc., the demand for complex and reliable real-time systems is increasing more and more, thanks to advanced development methods, for IT solutions respecting strict time constraints, to meet the needs. In this work, we carried out a simulation of treatment of biological samples with a view to an anatomy-pathological analysis.This device goes through 4 basic steps starting from a biological sample (tissue) to end with an image of cells visible in the microscope. the system designed is mainly based on real-time control of the synchronization of the various treatments from the various sub-systems of the process.

Keywords : sensor, real-time embedded system, real-time executive, SA-RT method, LACATRE language, Anatomy-pathological test bench.

Table des matières

Liste des tableaux

Table des figures

Acronymes

ADN acide désoxyribonucléique. 7
APAC Asia-Pacific. 27
API Application Programming Interface. 25

BioMEMS Biological Micro Electro Mechanical Systems. 2, 6, 8, 10, 71, 73
BioNEMS Bio Micro and Nano Electro-Mechanical Systems. 8

DDC Diagramme de contexte. 23
DFC Le diagramme de flot de contrôle. 23
DFD Diagramme de flot de données. 23

EMEA Europe, Middle East, and Africa. 27
ETR exécutif temps réel. 24

IHM Interactions homme-machine. 12

LACATRE Langage d'Aide à la Conception d'Applications Temps Réel. 38

MEMS Micro Electro Mechanical Systems. 6

OS Operating system. XI, 15, 27

RAM Random access memory. 28
ROM Read-only memory. 14
RT Real Time. 15
RTOS Real time operating system. XI, 26–28

SA-RT Stuctured Analysis for Real Time0. 21, 22, 29, 37, 38

SCADA Supervisory Control and Data Acquisition. 15

SE système embarqué. 14

TR Temps Réel. 15

u-TAS icro-Total Analysis Systems. 7

Introduction générale

L'histopathologie est une spécialité médicale dont le rôle est d'établir ou de confirmer un diagnostic à partir d'analyses de tissus et/ou de cellules prélevées sur un patient.Ce diagnostic a un rôle majeur dans l'évaluation du diagnostic clinique et le choix du traitement.

L'examen pathologique comprend l'évaluation des facteurs pronostiques et la détection de mutations génétiques dans les cellules cancéreuses. Cela permet de prédire l'efficacité des traitements.

Il existe deux types de diagnostic, la méthode de frottis pour prélever des cellules jusqu'à ce qu'elles soient examinées au microscope immédiatement après la coloration des cellules.

L'autre type est lié aux échantillons des tissus de sorte que les échantillons sont traités par étapes successives jusqu'à ce que des échantillons diagnosticables soient obtenus au microscope.

Les résultats et les images des cellules obtenues au microscope sont généralement stockés dans une carte mémoire pour être étudiés.

Problématique

Le processus de traitement des cellules est effectué dans plusieurs équipements distincts, ce qui nécessite du personnel du laboratoire pour s'occuper de chacune des étapes. Lors de l'examen des cellules au microscope, les informations sur les cellules sont stockées dans une carte de stockage, afin de rechercher certaines images de cellules, qui demande du temps et des efforts.

Objectif

L'objectif principal du projet est :

1. Realiser un programme de simulation col-lecter toutes les pricipales étapes(désyhdratation, mise de paraffine , coloration..ect) du traitement cellulaire en un seul processus comme comme nous avons décrit dans la première méthode (Figure 1), en utilisant lanotion multi-tâches (temps réel).
2. Conception de capteur Biological Micro Electro Mechanical Systems (BioMEMS) qui va traiter et l'analyser les cellules d'échantillons en temps réel comme nous avons représenté en deuxième methode (Figure 1).
3. Realiser une interface pour stocker/rechercher les informations d'examinassions de ces cellules.

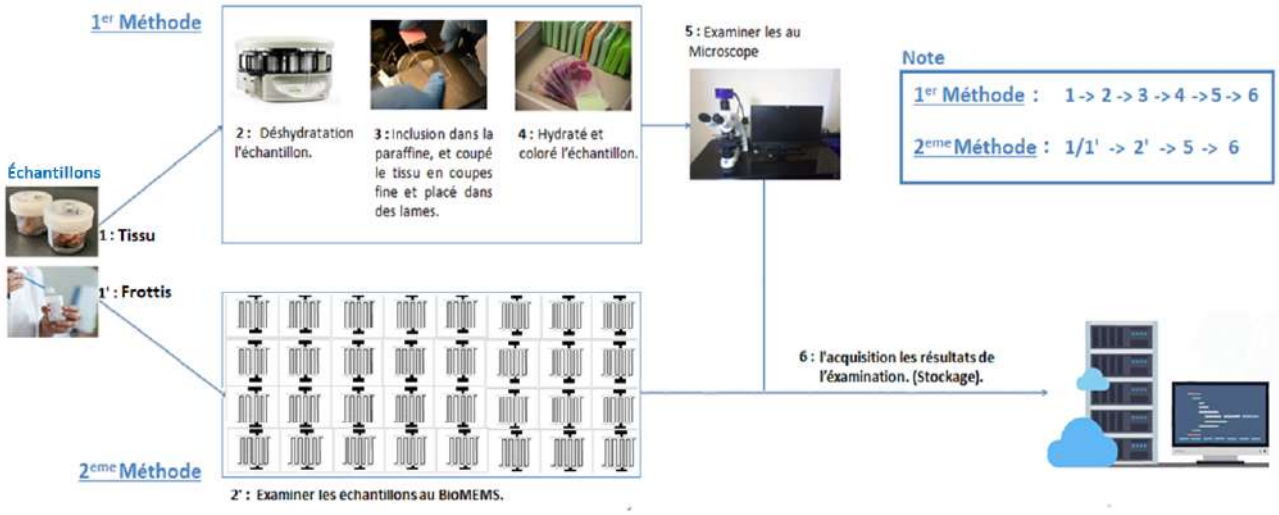

FIGURE 1 – Schéma général de notre travail.

Organisation du rapport

Cette thèse est structurée en quatre chapitres, aisi qu'une introduction générale et une conclusion :

— ***Introduction générale***
Présente une initiation aux systèmes embarqué temps reél et au contexte, à l'énoncé du problème et l'objectif de la thèse.

— ***Chapitre 1 Technologie de capteurs*** Il contient, un aperçu des capteurs avec leur classification et leur type, et quelques concepts de base sur les Biomem.

— ***Chapitre 2 Les système temps réel embarqué***
Présente des concepts sur le système temps réel et le système embarqué et leurs classification et types.

— ***Chapitre 3 Méthode de développement et les plateformes utilisé***
Présente les concepts de la méthode de spécification et du logiciel utilisé.
— ***Chaptire 4 Conception et implémentation***
Présente toute la procédure de réalisation de notre travail, nous donnons une vue détaillée de notre système, en présentant les différentes étapes de développement du noyau conçu.
— ***Conclusion générale***
Nous concluons ce rapport par une synthèse du travail effectué et nous offrons des perspectives pour nos travaux .

Chapitre 1

Technologies de capteurs

1 Introduction

Ce chapitre présente les capteurs qui sont devenus presque balayant toutes les domaines(industrie, recherche scientifique, services, loisirs ...),et qui contrôler de nombreux paramètres physiques (température, force, position,vitesse, luminosité, ...). Le capteur est l'élément indispensable à la mesure de ces grandeurs physiques.

2 Capteur

Un capteur est un dispositif permettant de détecter, en vue de le quantifier et de le représenter, un phénomène physique sous la forme d'un signal, généralement électrique. Le capteur se différencie du détecteur et du senseur par sa possibilité de délivrer une grandeur physique directement utilisable pour une mesure ou une commande [1].

FIGURE 1.1 – Schéma de capteur

2.1 Classification de Capteurs

Généralement on classe les capteurs par :

— ***Capteurs actifs :***Les capteurs actifs transforment directement le mesurande m en grandeur électrique.

— ***Capteurs passif :*** Les capteurs dont le signal électrique délivré est une variation d'impédance sont dits passifs car ils nécessitent une source d'énergie électrique.

2.2 Les différents types de capteurs

Un capteur n'est jamais parfait, il convient de connaître avec la plus grande précision possible son état d'imperfection. De plus, il faut prendre en compte la perturbation apportée au système par la mesure. Le concepteur d'une chaîne instrumentale aura donc des choix à opérer [2].

2.2.1 Les capteurs analogiques

La sortie est une grandeur électrique dont la valeur est une fonction de la grandeur physique mesurée par le capteur. La sortie peut prendre une infinité de valeurs continues. Le signal des capteurs analogiques peut être du type [3] :

— sortie tension ;

— sortie courant ;

— règle graduée, cadran, jauge (avec une aiguille ou un fluide) ;

— etc.

2.2.2 Capteur numérique

Le signal élaboré par le capteur, est directement codé sous une forme numérique au sein même du capteur. Par exemple : Roue codeuse [4].

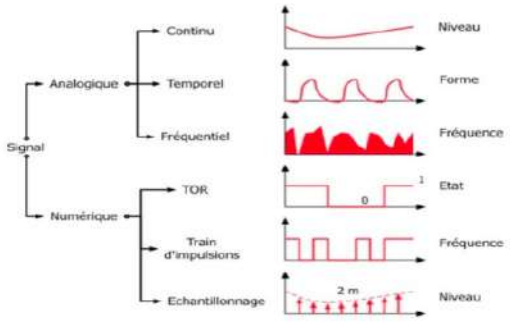

FIGURE 1.2 – Types de signaux analogiques et numériques

2.3 Bio-MEMS, Biocapteurs, Laboratoires sur puces

Plus récemment, une nouvelle branche de Micro Electro Mechanical Systems (MEMS) a émergé. Il s'agit des systèmes Biological Micro Electro Mechanical Systems (BioMEMS) qui sont des dispositifs plus és vers les applications médicales et biomédicales, telles que le dépistage des maladies[5].

maginez [6] :

- ✓ Au cours d'une visite médicale, votre médecin vous fait une prise de sang totalement indolore grâce à un petit patch contenant des centaines de micro-seringues. Il dépose le patch sur une plateforme CDLab qu'il introduit dans un analyseur qui vous rappellera le lecteur CD. En quelques minutes, il reçoit les analyses sur son ordinateur, détermine l'espèce bactérienne ou virale à l'origine de votre infection avant de vous prescrire un traitement ciblé et adéquat.
- ✓ Un micro-dispositif qui, une fois implanté sur la rétine, rétablit la vue à des patients aveugles.
- ✓ Au lieu de recourir à une chirurgie risquée, votre médecin vous injecte dans le sang une microcapsule qui va parcourir votre corps et atteindre l'organe malade. Une fois sur la cible, la capsule est activée à distance pour effectuer le traitement ou libérer les médicaments nécessaires.
- ✓ Un patch ou un microdispositif planté sur ou sous la peau qui, selon la concentration du glucose dans le sang, déclenche ou non une libération d'insuline, évitant ainsi auxdiabétiques les injections régulières et la surveillance permanente de leur glycémie.

Ces possibilités ne sont qu'un exemple d'une révolution en marche grâce aux bio-microtechnologies. Il s'agit d'une miniaturisation des systèmes de détection, d'analyse et de traitement au service de la biologie, de l'environnement, de la médecine et de l'agro-alimentaire. Au cours de la dernière décennie, les premiers développements de miniaturisation ont d'abord concerné des dispositifs électromécaniques ou BioMEMS (pour Micro Electro Mechanical Systems) tels que les accéléromètres et les capteurs de pression qui ont eu un large succès commercial. Cette technologie s'est rapidement étendue aux domaines biologique et médical pour donner naissance à ce qui est connu sous les noms de "BioMEMS", "biosensors", "lab on chips" ou "µ-TAS" (pour Micro-Total Analysis Systems). Ces dispositifs Figure 1.3 sont généralement composés d'une partie microfluidique incluant des microcanaux, des microvalves, des micropompes, des micromixeurs, des microréacteurs pour la manipulation des différents liquides ou objets biologiques, d'une plateforme de détection plus au moins complexe contenant un ou plusieurs capteurs, et de la composante biologique qui peut être intégrée (biorécepteur) ou momentanément introduite pour les besoins d'analyse.

L'explosion de la recherche et des applications dans ce domaine a entraîné une multiplication de la nomenclature et des concepts techniques. Nous tenterons donc tout d'abord d'établir une nomenclature précise basée sur les fonctionnalités et les degrés d'intégration des

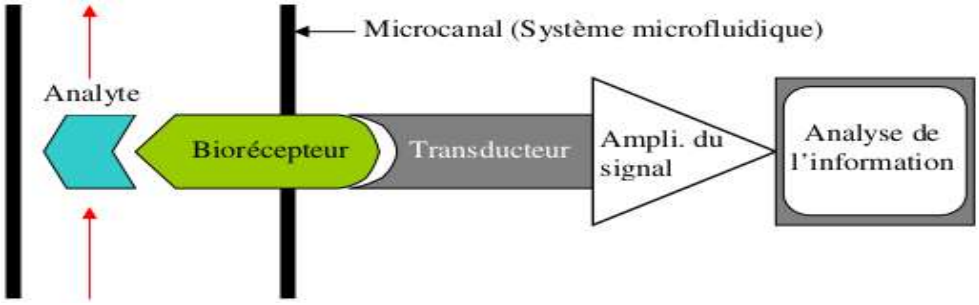

FIGURE 1.3 – Schéma basique d'un BioMEMS.

différents systèmes. Les termes de bioMEMS et bio-microsystèmes sont de plus en plus utilisés pour désigner, d'une manière générale, la grande panoplie des microdispositifs destinés aux sciences de la vie. Ces derniers peuvent être classés en 5 grandes catégories :

1. Micropuces ou «Microarrays» : Elles sont basées essentiellement sur l'hybridation (brins d'ADN) ou la reconnaissance moléculaire (ligand-récepteur) qui sont généralement détectées par fluorescence sans l'intervention d'un transducteur intégré. Actuellement, beaucoup de développements se font sur des puces à protéines, à lipides ou à saccharides, qui s'ajoutent aux puces à ADN déjà commercialisées.
2. Biocapteurs ou «Biosensors» : Ils sont composés de biorécepteurs immobilisés sur un transducteur qui peut être optique ou plus généralement électromagnétique, électrochimique, piézoélectrique, calorimétrique ou acoustique. Le principe de base d'un biocapteur est de transformer une propriété biochimique d'un phénomène biologique en un signal électrique. Le premier biocapteur a été développé en 1950 par Leland Clarke pour mesurer la concentration en oxygène dissout dans le sang grâce à des électrodes fonctionnalisées.
3. Laboratoire sur puces ou «Lab on chips, u-TAS» : Ce sont des microdispositifs multifonctionnels et plus élaborés qui

   permettent, selon leur degré d'intégration, une fonction de transduction, mais surtout de préparation des échantillons, séparation, analyse, de culture cellulaire ou tissulaire. En plus du transducteur, ces composants contiennent un système microfluidique (microcanaux) pour gérer les fluides à analyser.
4. Dispositifs implantables ou «implantable devices» : Ce sont des microsystèmes soit biohybrides soit en contact direct avec un système vivant à l'exemple des organes artificiels, les interfaces cerveau-machine et les systèmes de libération médicamenteuse.

   micro/nanomachines : Ce sont des dispositifs qui peuvent être injectés dans les liquides corporels, qui n'ont pas forcément une composante biologique, mais dont la fonction est de cibler puis d'interagir avec le système vivant.

Cette classification n'est évidemment pas statique. Plusieurs dispositifs peuvent résulter de l'intégration de deux ou plusieurs des éléments cités ci-dessus, ou même faire l'objet d'une combinaison avec des systèmes nanométriques pour former ce qui est appelé les Bio-NEMS (pour Bio-NanoElectroMechanical Systems), à l'exemple des nanomachines destinées au ciblage médical. L'essor de ces micro-dispositifs et l'intérêt montré par les communautés scientifiques et industrielles (Figure 1.4) est expliqué par les multiples avantages que procure le principe même de la miniaturisation, mais aussi par les phénomènes et les propriétés qui émergent à l'échelle du micromètre et du nanomètre. Les avantages les plus évidents sont :

- ✓ Le travail sur des microvolumes d'échantillons biologiques ou médicaux, ce qui se révèle d'un grand secours pour le génie génétique, l'analyse paléontologique ou la médecine légale, où l'échantillon disponible n'est pas souvent suffisant pour une analyse classique.
- ✓ L'intégration plus importante des fonctions, ce qui réduit ainsi le besoin de l'intervention humaine dans la préparation et l'analyse des échantillons, et par conséquent, réduit les problèmes de contamination, de fiabilité et de risques sécuritaires.
- ✓ Une très grande sensibilité et une spécificité améliorée, étant donné que l'analyse pourrait potentiellement se faire sur une cellule ou molécule unique.
- ✓ Réactions plus rapides et plus maîtrisées grâce à la microfluidique et aux propriétés inhérentes à la miniaturisation (rapport surface/volume important, domination des forces interfaciales et de la diffusion moléculaire).
- ✓ Réduction du coût et du temps d'analyse grâce, entre autre, à la parallélisation des systèmes de mesure.

En deux décennies, tous ces avantages ont permis aux BioMEMS d'investir toutes les échelles de la structure du vivant, allant du génie génétique (séquençage, génotypage, pharmacogénomique), en passant par la protéomique (conformations, interactions) et les manipulations cellulaires, jusqu'à l'ingénierie tissulaire. De plus en plus d'applications sont également annoncées pour le diagnostic et le traitement médicaux, les organes artificiels, l'interface cerveau-machine et les sécurités agro-alimentaire et environnementale.

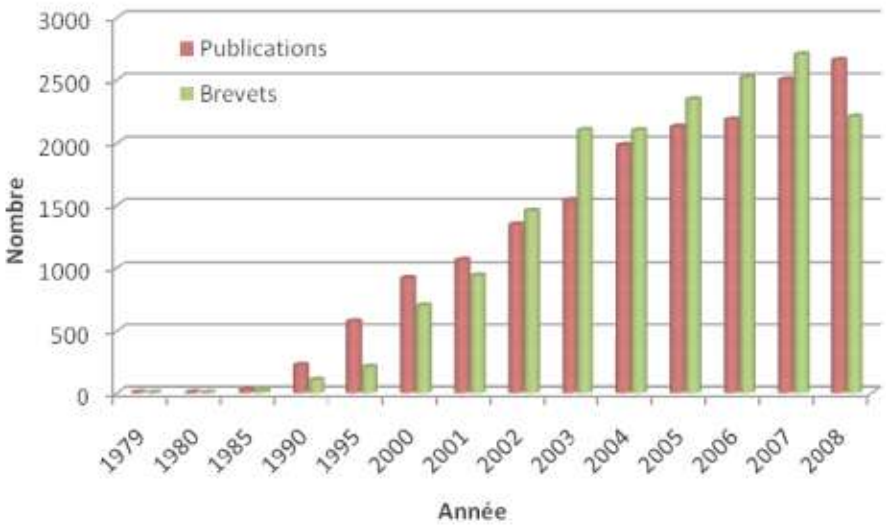

FIGURE 1.4 – Evolution de la production scientifique (publications, brevets) concernant les bioMEMS depuis l'année 1979. [7]

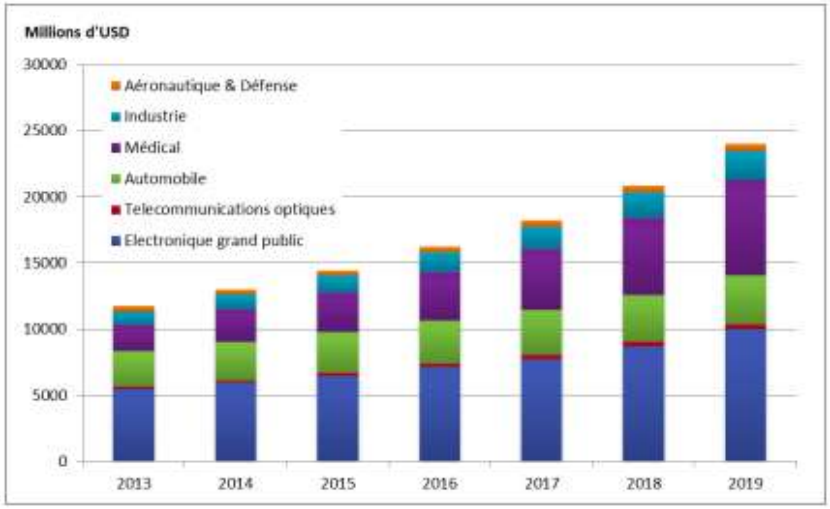

FIGURE 1.5 – Evolution de la taille mondiale des marchés par segments, entre 2013 et 2019. [?]

3 Conclusion

Dans ce chapitre, nous avons présenté sur les capteurs et de leurs caractéristiques et leurs types de manière générale, et nous avons abordé les capteurs BioMEMS en tant que capteurs conviennent à notre étude en termes de transmission et d'analyse d'informations en Anatomie-pathologique.

Chapitre 2

Les systèmes temps réel embarqués

1 Introduction

Les systèmes temps réels sont des systèmes permettant la réalisation d'applications temps réel. Une application est dite temps réel si elle exécute ses actions dans un délai borné. En effet, un système temps réel diffère d'un système de traitement d'informations classique par le fait que la valeur d'une donnée produite ne dépend pas uniquement de la correction de son calcul mais également de la date à laquelle la donnée est disponible.

On retrouve les contraintes temporelles dans de nombreux secteurs d'activités tels que l'aéronautique au travers des systèmes de pilotage embarqués (avions, satellites), l'automatisation d'ensembles industriels au travers des systèmes de contrôle de procédés (usines, centrales nucléaires), la gestion, les télécommunications, l'automobile, la robotique, . . . etc.

2 Systèmes temps réel

2.1 Systèmes

D'une manière générale, un système est vu comme un ensemble composite constitué de personnels, de matériels, de logiciels et de procédures. Ces éléments, en interaction mutuelle, sont organisés pour répondre à un besoin et correspondre à une finalité.

2.1.1 Types de systèmes

Une classification établie par G. Berry [8] se base sur le mode d'interactions entre l'environnement physique et le système. On distingue deux types de systèmes : transformationnel et réactif.[9], la classification des systèmes informatiques est comme suit :

— ***Système transformationnel :*** Un système transformationnel produit des sorties en fonction d'entrées selon un processus de calcul indépendant de l'environnement. Dans ce cas, les instants de production de résultats ne sont pas contraints car pour de tels systèmes, c'est l'exactitude logique qui prime. Un compilateur, un éditeur de factures sont des exemples de systèmes transformationnels.
— ***Système réactif :*** Un système réactif interagit continuellement et instantanément avec son environnement. Ce dernier tient une place importante dans le sens où ce sont les dynamiques de l'environnement qui contraignent la temporalité des interactions. De cette manière, un système réactif élabore et produit des sorties en fonction d'entrées et en réaction à des évènements produits par l'environnement. Un contrôleur de processus industriel, une IHM critique sont des exemples de systèmes réactifs.

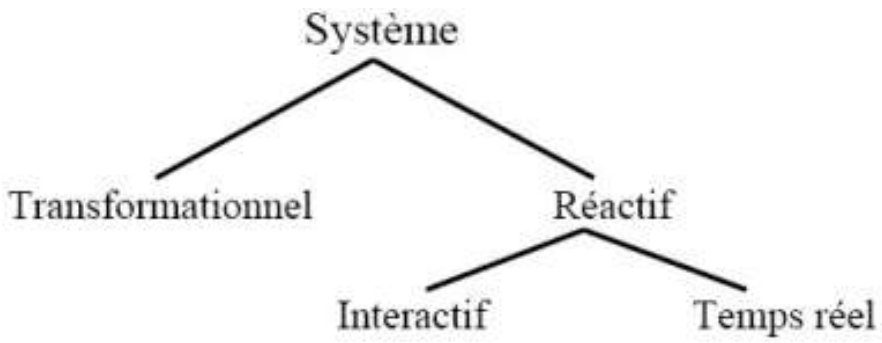

FIGURE 2.1 – : Classification des systèmes[9]

— ***Système transformationnel :*** Un système transformationnel produit des sorties en fonction d'entrées selon un processus de calcul indépendant de l'environnement. Dans ce cas, les instants de production de résultats ne sont pas contraints car pour de tels systèmes, c'est l'exactitude logique qui prime. Un compilateur, un éditeur de factures sont des exemples de systèmes transformationnels.

2.1.2 Du réactif au temps réel

Les définitions des système temps réel sont nombreuses mais complètement convergentes, nous en citerons :

Définition 1

Les systèmes temps réel sont des systèmes réactifs qui permettent l'implantation d'applications où le respect des contraintes temporelles est la principale contrainte à satisfaire. Une application temps réel est une application qui met en œuvre un système informatique dont le

fonctionnement est assujetti à l'évolution dynamique de l'état d'un environnement (procédé) qui lui est connecté et dont il doit contrôler le comportement[10].

Définition 2

Un système temps réel est un système dont l'exactitude des résultats ne dépend pas seulement de l'exactitude logique des calculs mais aussi de la date à laquelle le résultat est produit. Si les contraintes temporelles ne sont pas satisfaites, on dit qu'une défaillance système s'est produite[11].

En effet, la validité d'un système temps réel est conditionnée par deux attributs à savoir les résultats logiques du traitement effectué ainsi que l'aspect temporel, autrement dit, un système temps réel doit satisfaire deux contraintes importantes :

— ***Exactitude logique :*** fournir des sorties adéquates assurant le comportement désiré pour le système suite à des évènements en entrée et aux données communiquées.

— ***Exactitude temporelle :*** respecter des contraintes temporelles associées aux traitements effectués par le système pendant toute la durée de vie du système.

Par conséquent, dans un système temps réel, un résultat hors échéance reste faux même s'il s'avère logiquement correct.

2.1.3 Fonctionnement d'un système temps réel

Un système temps réel est composé essentiellement de deux éléments distincts (Figure 2.2) à savoir une ou plusieurs entités physiques constituant le procédé (le système contrôlé) et un système de contrôle (contrôleur) qui est chargé de surveiller de manière régulière, périodique, l'état du procédé en récupérant les valeurs en provenance des capteurs.

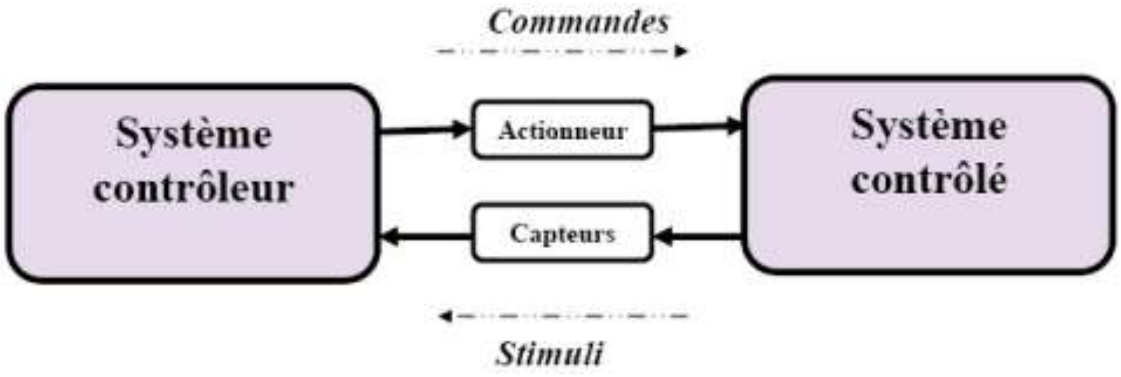

FIGURE 2.2 – : Principe général d'un système temps réel

Les capteurs scrutent les évènements du système contrôlé et fournissent des mesures tandis que les actionneurs effectuent les réactions du système.

2.2 Systèmes embarqués temps réel

Les systèmes embarqués temps réel ont depuis quelques années envahi un grand nombre de domaines. Ils sont présents dans le contrôle industriel, les télécommunications, en aéronautique, en automobile, et surtout la santé . . . etc. La fonction principale d'un système embarqué est celle de contrôler l'évolution de son environnement. En plus d'êtres réactifs, de tels systèmes sont souvent soumis à des contraintes strictes imposées par leurs environnements.

2.2.1 Systèmes embarqués

Voici quelques définitions pour cerner le concept de système embarqué [12] :

- ✓ Un système embarqué (SE) est un système informatisé spécialisé qui constitue une partie intégrante d'un système plus large ou une machine. Typiquement, c'est un système sur un seul processeur et dont les programmes sont stockés en ROM. A priori, tous les systèmes qui ont des interfaces digitales (i.e. montre, caméra, voiture...) peuvent être considérés comme des SE. Certains SE ont un système d'exploitation et d'autres non car toute leur logique peut être implantée en un seul programme.
- ✓ Un système embarqué est une combinaison de logiciel et matériel, avec des capacités fixes ou programmables, qui est spécialement conçu pour un type d'application particulier. Les distributeurs automatiques de boissons, les automobiles, les équipements médicaux, les caméras, les avions, les jouets, les téléphones portables sont des exemples de systèmes qui abritent des SE.
- ✓ Un système embarqué est une composante primordiale d'un système (i.e. un avion, une voiture...) dont l'objectif est de commander, contrôler et superviser ce système.

Par rapport aux autres systèmes informatisés, les systèmes embarqués sont caractérisés par :

- Encombrement mémoire (mémoire limitée, pas de disque en général)
- Consommation d'énergie (batterie : point faible des SE)
- Poids et volume
- Autonomie
- Mobilité
- Communication (attention : la communication affecte la batterie)
- Contraintes de temps réel
- Contraintes de sécurité
- Coût de produits en relation avec le secteur cible

2.2.2 Systèmes embarqués temps réel et leur classification

On distingue deux grandes classes de systèmes TR : le temps réel "dur" et le temps réel "mou". La différence ne tient pas tant dans la valeur du délai de réaction spécifié que dans le respect de ce délai.

Considérons une tâche d'acquisition Ai, périodique de période Ti. Soit Si le début de la tâche Ai et Di la date d'échéance de Ai, c'est à dire la date à laquelle les résultats doivent être disponibles. Le délai de réaction du système spécifié sera donc Di - Si, avec évidemment Di-Si< Ti. Soit Ci la durée du traitement.

— ***Le temps réel "dur"*** Un système TR sera dit "dur" lorsque la totalité du traitement sera achevé avant Di, c'est à dire strictement inclus dans l'intervalle [Si, Di]. Si le traitement s'achève après Di, le système sera en défaut. L'essentiel est de respecter l'échéance de mise à disposition des résultats. On trouve les systèmes TR durs dans

tous les systèmes embarqués destinés au pilotage, dans les systèmes de mesures des machines de physique de particules, dans les robots de chirurgie et autres systèmes critiques. Ces systèmes doivent être parfaitement déterministes afin de garantir le

temps de calcul. Cela exclut l'usage de réseaux non déterministes comme Ethernet, mais aussi les mémoires caches et les processeurs parallèles. Les systèmes d'exploitation des ordinateurs doivent garantir un ordonnancement des tâches compatibles avec ces exigences. Ce sont les OS "temps réel", par exemple VxWorks, VRTX, LynxOS, qui sont normalisés Posix temps réel P1003.4, ou plus vieux HP RTE ou RTX-11.

— ***Le temps réel "mou"*** Un système TR sera dit "mou" lorsque il est toléré que le traitement débute dans l'intervalle [Si, Di] mais s'achève après Di mais avant Ti. Si le traitement s'achève après Ti, le système sera en défaut. Le dépassement de l'échéance Di ne doit pas être systématique, les tolérances étant spécifiées lors de la conception du système. Les systèmes de supervision et une bonne partie des systèmes de

contrôle/commande (SCADA) sont des systèmes temps réel mous, avec des tolérances de dépassement de l'échéance Di plus ou moins sévères selon le procédé sous contrôle.

En temps réel mou, on peut utiliser sous condition des OS non temps réel comme Unix, Linux (y compris RT-Linux qui n'est pas vraiment temps réel) ou Windows.

— ***Le temps réel ferme " firm real-time"*** Temps réel souple avec le manquement occasionnel des échéances.

Ex : projection vidéo (perte de quelques trames d'images).

2.3 Architectures embarquées temps réel

2.3.1 Architecture matérielle

Les systèmes temps réel sont fortement influencés par l'architecture matérielle sur laquelle doit s'exécuter le système informatique. Selon le nombre de processeurs et l'utilisation éventuelle d'un réseau, on distingue[13]

— ***Architecture centralisée*** — Le système est dans ce cas composé d'un calculateur pouvant être monoprocesseur ou multiprocesseur à mémoire partagée, auquel sont reliés des capteurs et des actionneurs. Le calculateur suffit à centraliser l'acquisition, le traitement et la mémorisation des informations. Le système étant centralisé, son architecture matérielle en est simplifiée. En revanche, ce type d'architecture conduit à un type de câblage de type étoile souvent important et couteux plus, un système centralisé est plus vulnérable aux défaillances.

— ***Architecture multi-calculateurs*** Le système est formé d'un ensemble de calculateurs non reliés entre eux où chaque calculateur implémente un certain nombre de fonctionnalités. Dans ce type d'architecture, les câblages sont réduits, ce qui permet de limiter les coûts et d'améliorer la fiabilité. Cependant, la gestion de la communication entre les différents systèmes est un problème crucial.

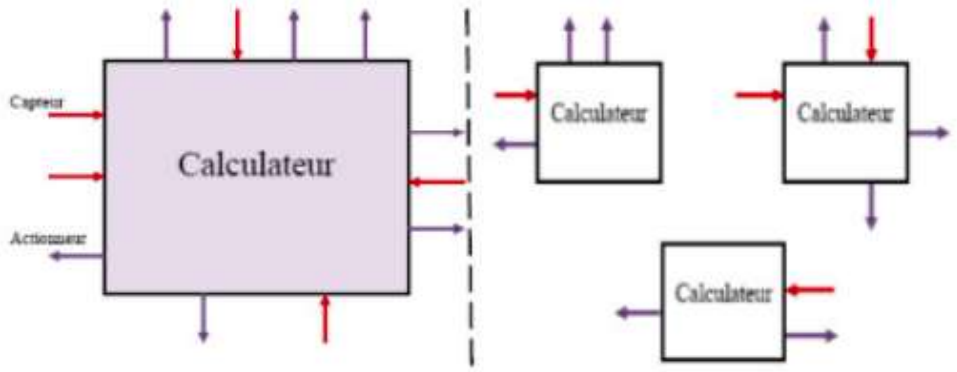

FIGURE 2.3 – Architectures centralisée et multi-calculateurs.

— ***Architecture multi-calculateurs*** C'est une architecture dans laquelle des calculateurs sont reliés entre eux par un bus. Dans ce cas, le comportement global est mieux contrôlé et les calculateurs peuvent partager des capteurs.

— ***Architecture fortement distribuée*** L'architecture est caractérisée par une plus grande autonomie des composants. En effet, les capteurs ainsi que les actionneurs peuvent être directement connectés sur le bus (Figure 2.5).

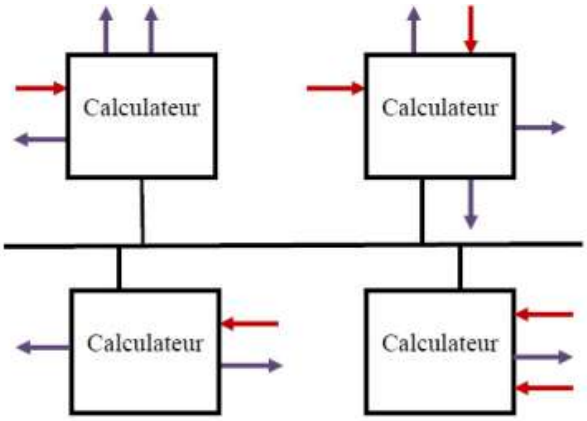

FIGURE 2.4 – Architecture faiblement distribuée.

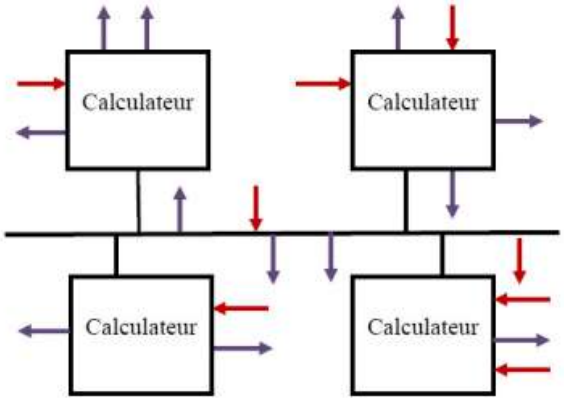

FIGURE 2.5 – Architecture fortement distribuée.

— ***Architecture hétérogène***
De nos jours, en plus d'être fortement distribuées, les architectures embarquées temps réel sont aussi hétérogènes. En effet, les composants peuvent être des microprocesseurs, des microcontrôleurs, différents médias de communication . . . etc. En revanche, la difficulté réside dans la gestion du multiplexage des données issues des capteurs et des calculateurs sur le bus de telle sorte que les contraintes temporelles de chacun des signaux soient satisfaites.

2.4 Ordonnancement temps réel

Pour un système de tâches, l'ordonnancement consiste à décider dans quel ordre exécuter les tâches. Un ordonnancement temps réel strict a le même objectif mais avec la contrainte

supplémentaire qu'aucune instance d'aucune tâche ne rate son échéance. Une des motivations de la théorie de l'ordonnancement, et par extension de l'ordonnancement temps réel, est de concevoir des algorithmes d'ordonnancement qui minimisent les temps de réponse des tâches. Dans le domaine du temps réel, en plus de décrire un algorithme d'ordonnancement, il est nécessaire de proposer un algorithme qui garantit, pour cet ordonnancement, que les échéances soient respectées[14].

2.4.1 Ordonnancement hors-ligne / en ligne

L'ordonnancement des tâches temps réel est à la charge de l'ordonnanceur, qui est une des briques logicielles qui composent le système d'exploitation temps réel. L'ordonnanceur peut être vu comme une tâche spécifique chargée de décider dans quel ordre exécuter les autres tâches. On distingue les ordonnancements hors-ligne* des ordonnancements en ligne*. Pour un ordonnancement hors-ligne, les décisions d'ordonnancement sont connues avant le début de l'ordonnancement. L'ordonnanceur n'a donc pas à calculer ses décisions d'ordonnancement, mais juste à appliquer une politique déjà établie. Au contraire, pour un ordonnancement en ligne, l'ordonnanceur prend ses décisions durant l'ordonnancement en fonction de l'état du système.

2.4.2 Ordonnancement temps réel multiprocesseur

L'ordonnancement temps réel concerne principalement les plates-formes monoprocesseurs. Dans cette thèse, nous nous intéressons aux plates-formes composées de plusieurs processeurs. Ce type de plate-forme est désigné par le terme de plate-forme multiprocesseur. Depuis le début des années 80, le constat qu'il serait impossible d'augmenter indéfiniment la fréquence des processeur a motivé l'intérêt pour les architectures à plusieurs processeurs.

Dans le cas de l'ordonnancement multiprocesseur, l'ordonnanceur a un degré de liberté supplémentaire, à savoir une liberté spatiale. En plus de décider quand exécuter une tâche, l'ordonnanceur doit maintenant décider sur quel processeur l'exécuter. Ainsi, l'ordonnanceur peut décider de faire migrer une tâche ou une instance de tâche d'un processeur vers un autre. L'ordonnancement temps réel multiprocesseur énoncé deux principales approches :

1. l'ordonnancement par partitionnement, pour laquelle les migrations ne sont pas autorisées. Chaque processeur dispose alors de son propre ordonnanceur pour ordonnancer l'ensemble de tâches qui lui a été alloué.
2. l'ordonnancement global, pour laquelle les migrations sont autorisées. Avec cette approche, un ordonnanceur unique ordonnance toutes les tâches du système.

Un des arguments qui peut justifier que la communauté porte un intérêt à ces deux approches est qu'elles sont incomparables. En d'autres termes, il existe des ensembles de tâches qui sont ordonnançables avec un algorithme d'ordonnancement par partitionnement et ne le sont pas

avec un algorithme d'ordonnancement global, et réciproquement (il existe des ensembles de tâches qui sont ordonnançables avec un algorithme d'ordonnancement global et ne le sont pas avec un algorithme d'ordonnancement par partitionnement).

2.4.3 Ordonnancement préemptif / non préemptif

Lorsque plusieurs tâches doivent être ordonnancées de manière concurrente, deux approches sont distinguées : les ordonnancement non préemptifs* et ordonnancements préemptifs*. Dans un ordonnancement non préemptif, une tâche qui a été démarrée ne peut pas être interrompue avant d'avoir terminé son exécution. Avec cette approche, il y a au plus un changement de contexte, c'est-à-dire que l'ordonnanceur remplace le contexte d'exécution d'une instance par celui d'une autre instance. Ces changements de contexte ont un coût qu'il faut prendre en considération durant l'analyse d'ordonnançabilité, bien que l'hypothèse que ce coût soit nul est souvent faite. Dans un ordonnancement préemptif, l'ordonnanceur peut décider d'interrompre une tâche pour en démarrer une autre. L'avantage de cette approche est de réduire les temps de réponse des tâches les plus prioritaires, de mieux utiliser les processeurs et d'obtenir des taux d'ordonnançabilité plus importants. Dans le cas d'un ordonnancement conduit par les priorités, ce type d'interruption se produit lorsqu'une instance plus prioritaire que l'instance en cours d'exécution est activée. Avec ce type d'ordonnancement, des tâches se retrouvent en situation de compétition, notamment lorsqu'elles partagent des ressources communes. Ce problème nécessite de synchroniser les ressources partagées pour garantir leur cohérence[15].

2.5 Notion du processus

Un processus est un ensemble de fonctions dans une certaine séquence qui fournit de la valeur à un client. Les processus sont démarrés par des événements externes clairement définis[16].

D'autre terme plus précis[17] :

- Un ensemble d'instructions à exécuter, pouvant être dans la mémoire morte, mais le plus souvent chargé depuis la mémoire de masse vers la mémoire vive ;
- Un espace d'adressage en mémoire vive pour stocker la pile, les données de travail, etc. ;
- Des ressources permettant des entrées-sorties de données, comme des ports réseau.

L'exécution d'un processus dure un certain temps, avec un début et (parfois) une fin. Un processus peut être démarré par un utilisateur par l'intermédiaire d'un périphérique ou bien par un autre processus : les « applications » utilisateur sont des ensembles de processus[17].

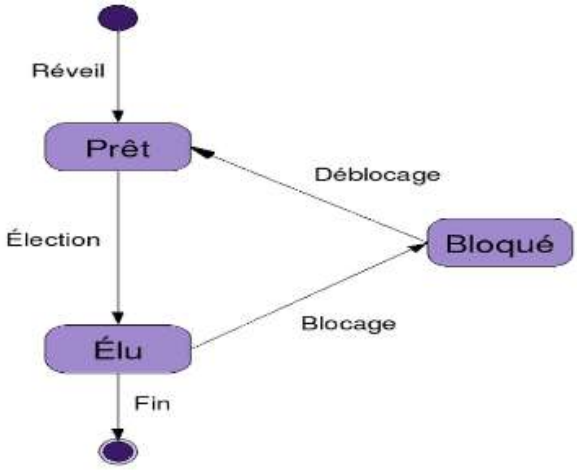

FIGURE 2.6 – Diagramme d'état d'un processus simple.

3 Conclusion

Dans ce chapitre, nous avons présenté les systèmes embarqués temps réel. En plus des fonctionnalités offertes, ces systèmes doivent satisfaire des contraintes temporelles, dont le non respect notamment dans le cas des systèmes à contraintes dures, peut être catastrophique. Le développement des systèmes embarqués temps réel est un processus délicat d'où l'intérêt grandissant pour des méthodologies prenant en charge les exigences entourant ce type de systèmes.

Le chapitre suivant passe en revue les méthodes de développement d'un système temps réel embarqué.

Chapitre 3

Méthode de développement et les plateformes utilisées

1 Introduction

Pour la réalisation de notre projet on utilise l'exécutif temps réel FreeRTOS, qui est strictement contrôlé du pont de vue qualité, robuste, pris en charge, ne contient aucune ambiguïté de propriété intellectuelle et gratuit à utiliser dans des applications commerciales sans aucune obligation d'exposer votre code source propriétaire.

Et pour le développement de ce type de système, on utilise la méthode SA-RT ce qui est programmé dans notre formation de master, Cette méthode permet de réaliser une description graphique et textuelle de l'application en termes de besoins.

2 Les méthodes de spécification pour un système temps réel

Langages de développement, qu'ils soient de spécification, de conception ou d'implémentation, constituent une base pour le développement de tout système. Pour ce qui est des systèmes temps réel en général, le paramètre 'temps' et par conséquent, la notion d'échéance sont cruciaux. Des langages intégrant le temps comme paramètre sont nombreux, nous en citerons les suivants :

— ***JSD :***« Jackson System Design ».
— ***SA-RT :*** « Structured Analysis Real Time »
— ***UML-RT :***« Unified Modeling Language - Real Time ».
— ***MSMC :***« Modélisation Simulation des Machines Cybernétiques ».

Le travail présenté dans cette mémoire basé sur la méthode d'analyse structuré pour les systèmes temps réel SA-RT.

2.1 La méthode SA-RT

Afin de remédier à l'inconvénient majeur de l'analyse structurée et qui concerne son incapacité à spécifier l'aspect dynamique d'un système, le langage SA-RT (Stuctured Analysis for Real Time) [18]fut développé. En plus de la description des processus de transformation, SA-RT prend en compte la spécification de la logique de contrôle du système spécifié. La spécification s'opère d'une manière descendante à travers des niveaux de description permettant de décrire différents sous-systèmes. Cependant, SA-RT constitue un moyen informel de spécification de systèmes temps réel et la vérification de tels systèmes s'avère impossible.

Fonction	Signification	Représentation graphique	
Traitement ou process	Unité de travail qui réalise la transformation des données d'entrée en données de sortie	– Cercle ou bulle – Action décrite par : verbe + nom	
Flot de données	Vecteur nommé reliant deux process, sur lequel circule un ensemble de données de même nature	– Flèche en trait plein – Donnée nommée	
Unité de stockage ou réservoir	Entité ou zone de rangement de données	– Deux traits parallèles – Entité nommée	
Entité externe ou terminateur	Provenance, source ou destination des données	– Rectangle – Entité nommée	

FIGURE 3.1 – Les différents éléments graphiques de la méthode SA-RT.

La SA-RT considère la spécification des systèmes temps réel selon 3 points de vue orthogonaux [19] :

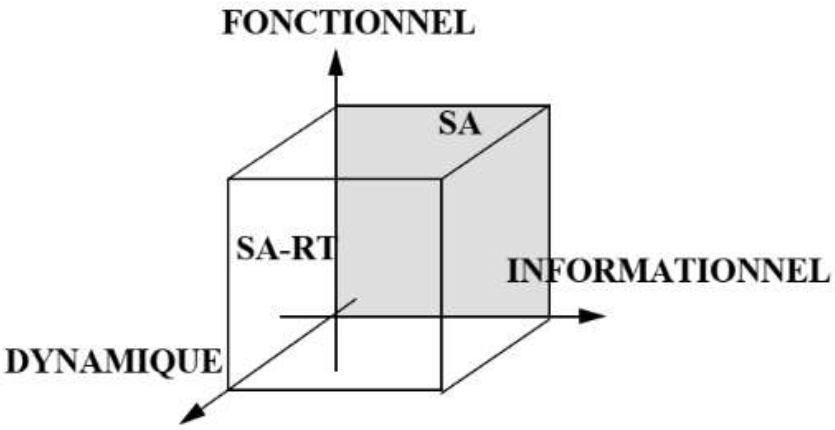

FIGURE 3.2 – Représentation graphique de défirent aspect de SA-RT.

Aspect fonctionnel : Décompose le système en une hiérarchie de fonctions présenté par le diagramme de contexte DDC.

Aspect informationnel : défini les données manipulées par les fonctions présenté par le diagramme de flot de données DFD.

Aspect dynamique : Exprime le contrôle du flux des données et l'activation des fonctions présenté par le diagramme de flot de contrôle DFC.

2.1.1 Objectif de la méthode SA-RT

L'objectif de SA-RT est d'obtenir un modèle des besoins permettant la développent :
- Conformité aux besoins réels de l'Utilisateur.
- Communicabilité (i.e. interprétables sans erreurs).
- Faisabilité.
- Modifiabilité.
- Validabilité.
- Complétude.
- Cohérence.

2.2 Les outils d'implémentation pour un système temps réel

Pour l'implémentation d'un système temps réel on utilise ce qu'on appelle exécutif temps réel.

2.2.1 Exécutif temps réel

Un exécutif temps réel (ETR) est en définitive, un noyau temps réel, le noyau gère les interruptions matérielles, crée les tâches et ou processus, gère la mémoire, se sert d'une interruption horloge pour faire l'ordonnancement des tâches et des processus, plus les pilotes de périphériques pour la gestion des entrées/, protocoles réseaux[20].

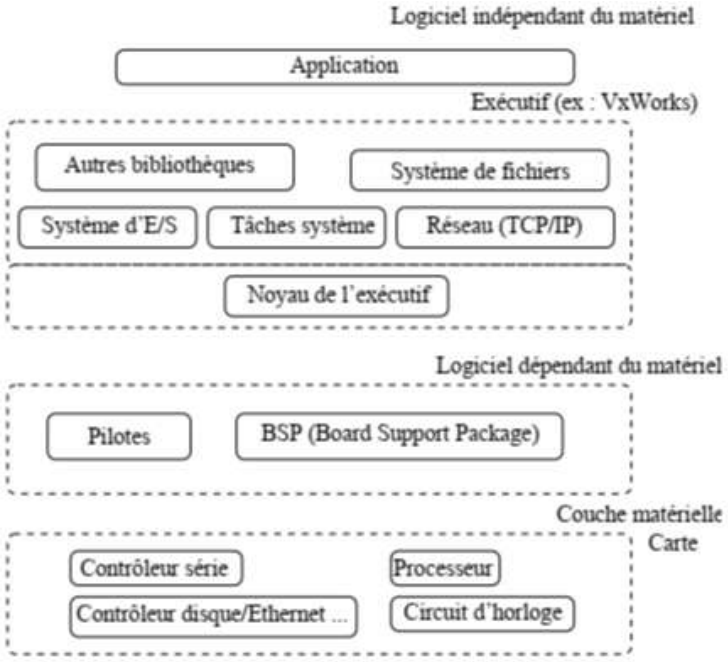

FIGURE 3.3 – Exemple d'un exécutif temps réel Vx Works.

Un Systèmes d'exploitations généralistes Unix, Linux, Microsoft Windows, etc. sont très consommateurs de ressources (processeur, mémoire,...), contrairement a les exécutifs temps réel comme FreeRTOS,Vx Works,POSIX, RTOS, etc. Peu consommateurs de ressources et embarquables.

2.2.2 Les Services offret par un exécutif temps réel

- Ordonnancement et tâches.
- Synchronisation : les sémaphores.
- Communication : Les interfaces série, file de messages, etc.
- Manipulation du temps.
- Gestion des interruptions.

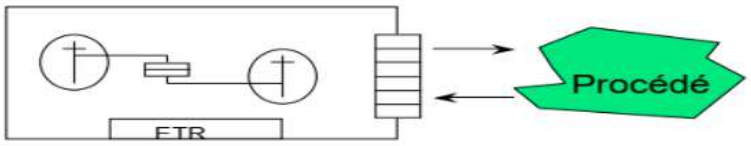

FIGURE 3.4 – Représentation schématique d'un exécutif temps réel.

3 Plateforme matériele

Pour la réalisation de notre projet on utilise l'exécutif temps réel FreeRTOS, et les capteurs d'information et exploiter la simplicité d'utilisation et la licence libre, ces plateformes été développées par des professionnels, strictement contrôlé de qualité, robuste, pris en charge, ne contient aucune ambiguïté de propriété intellectuelle et est vraiment gratuit à utiliser dans des applications commerciales sans aucune obligation d'exposer votre code source propriétaire.

3.1 L'exécutif temps réel FreeRTOS

Pour l'implémentation d'un système temps réel on utilise ce qu'on appelle exécutif temps réel.

Il existe un grand nombre de RTOS, libres, open Sources et propriétaires (RTX, MicroC/OS III, FreeRTOS, VxWorks...)

3.1.1 FreeRTOS

FreeRTOS, comme n'importe quel autre noyau, est un outil purement logiciel, ce n'est qu'un système de fichiers. FreeRTOS nous propose une API de programmation pour gérer un environnement multitâches. La notion de tâche sera présentée par la suite. Vous trouverez ci-dessous le systèmes de fichiers du noyau [21] :

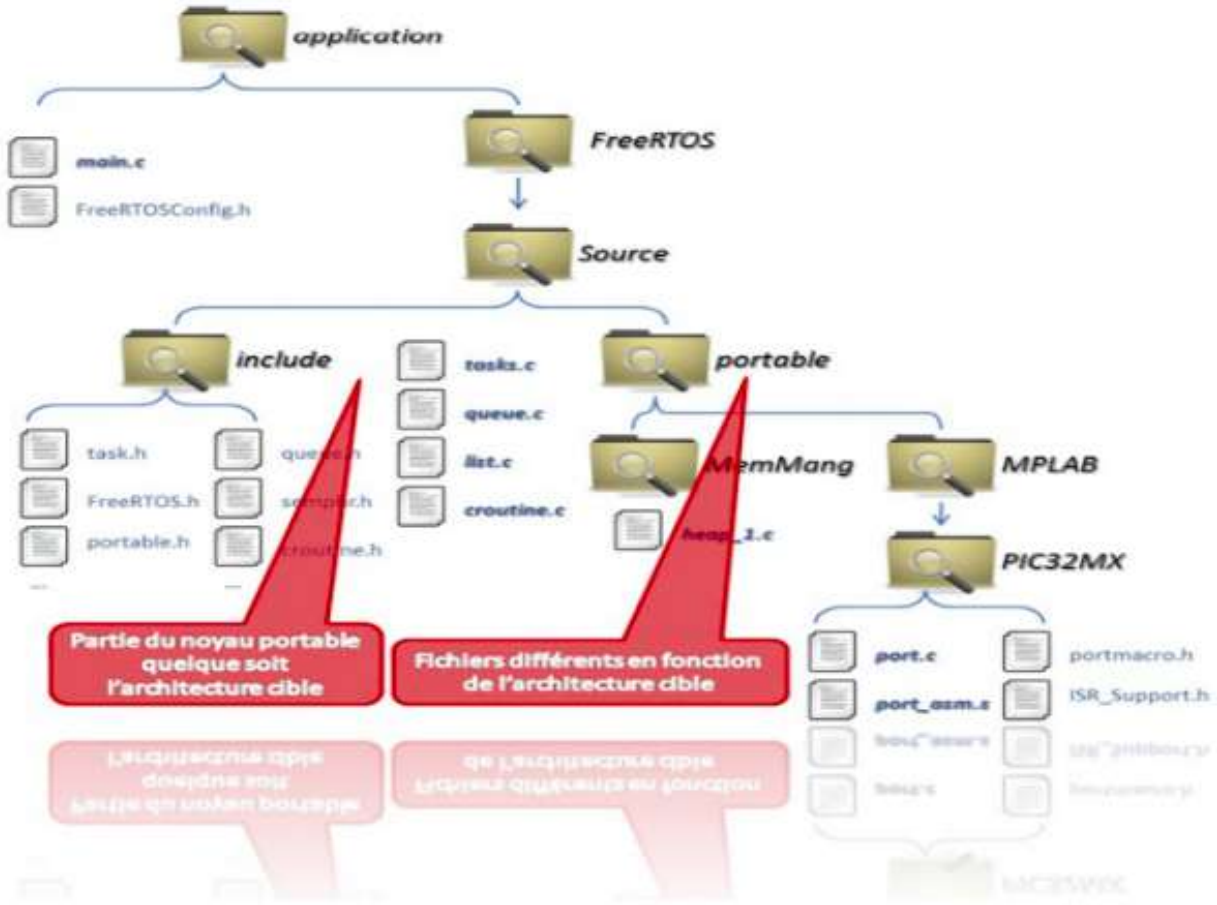

FIGURE 3.5 – Le système de fichiers du noyau « FreeRTOS ».

3.2 Pourquoi FreeRTOS ?

FreeRTOS est le RTOS actuellement le plus utilisé et celui le plus regardé pour démarrer de nouveaux projets :

Noyau de confiance : Avec une robustesse éprouvée, un faible encombrement et une large prise en charge des périphériques, le noyau FreeRTOS est considéré par les entreprises de renommée mondiale comme la norme de facto pour les microcontrôleurs et les petits microprocesseurs.

Accélérez la mise sur le marché : Grâce aux démos préconfigurées détaillées et aux intégrations de référence de l'Internet des objets, il n'est pas nécessaire de déterminer comment configurer un projet. Téléchargez, compilez et accédez instantanément au marché plus rapidement.

Prise en charge d'un large écosystème : Notre écosystème de partenaires offre un large éventail d'options, notamment des contributions de la communauté, un support professionnel, ainsi que des outils intégrés d'IDE et de productivité.

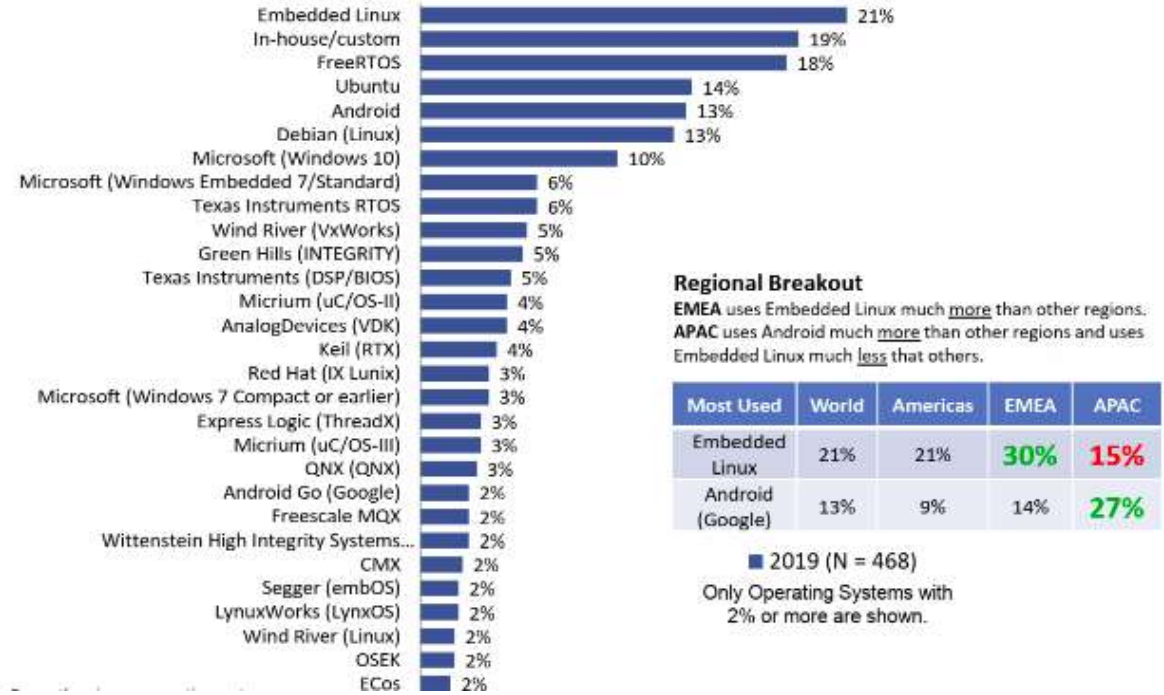

FIGURE 3.6 – Observation de marché des OS et RTOS pour l'embarqué. [22]

Remarque :

EMEA , est une appellation que certaines entreprises utilisent pour désigner la région économique qui regroupe les pays d'Europe, du Moyen-Orient et de l'Afrique.

APAC est un ensemble géographique constitué de l'Extrême-Orient, du sous-continent indien et de l'Océanie.

3.3 Fonctionnalités de FreeRTOS :

— Opération préventive ou coopérative
— Affectation des priorités de tâches très flexible
— Mécanisme de notification des tâches flexible, rapide et léger
— Files d'attente
— Sémaphores
— Mutex

— Minuteurs
— Groupes d'événements
— Vérification du dépassement de la pile
— Enregistrement de trace
— Collecte des statistiques d'exécution des tâches
— Licence commerciale et assistance en option
— Modèle d'imbrication d'interruption complète (pour certaines architectures)
— Pile d'interruption gérée par logiciel le cas échéant (cela peut aider à économiser la RAM)[23].

3.4 Algorithmes d'ordonnancement de FreeRTOS :

Il existe un certain nombre d'algorithmes de planification assez standard qui sont utilisés dans les RTOS, l'algorithme idéal serait capable de planifier tout ensemble de tâches pour lequel il existe un calendrier, échouer normalement lorsque les tâches ne sont pas planifiables et seraient simples à utiliser (ayant des priorités fixes ou "statiques" pour chaque tâche serait un bon début pour être simple...). Bien sûr, un tel algorithme n'existe pas, nous avons donc un nombre d'algorithmes différents que nous pouvons envisager, chacun avec son propre ensemble d'avantages et d'inconvénients. FreeRTOS n'utilise que deux algorithmes d'ordonnancement :

« **round-robin Scheduling** » ,Dans cet algorithme, toutes les tâches de priorité égale obtiennent le processeur en parts égales de temps processeur.

« **FixedPriorityPreemptiveScheduling** » cet algorithmesélectionne les tâches en fonction de leur priorité. En d'autres termes, une tâche à haute priorité obtient toujours le processeur avant une tâche à faible priorité. Une tâche de faible priorité s'exécute uniquement lorsqu'il n'y a pas de tâche de haute priorité à l'état prêt.

Remarque : dans notre projet, nous utilisons l'ordonnancement en mélangeant les deux algorithmes et il est connu sous le nom « PrioritizedPreemptiveSchedulingwith Time Slicing ».

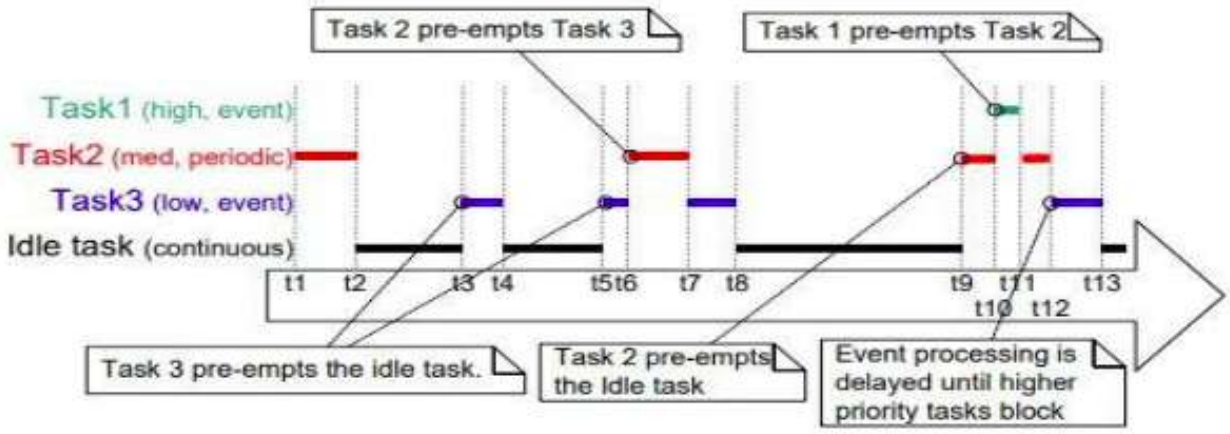

FIGURE 3.7 – Exemple de « Prioritized Preemptive Scheduling with Time Slicing ».

4 Conclusion

Ce chapitre se concentre sur l'utilisation de l'environnement exécutif FreeRTOS qui conviennent de notre travail et le langage SA-RT qui très répandu et très utilisé pour spécifier les systèmes temps réel, a cause de sa simplicité d'utilisation et son pouvoir d'expression, graphique en particulier, le SA-RT possède l'avantage d'être bien lisible car la spécification est faite d'une manière hiérarchique, cependant cette spécification est informelle et ne permet pas de vérifier directement le modèle...

Chapitre 4

Conception et implémentation

1 Introduction

Dans le cadre de la conception et de l'optimisation les methodes d'étude des échantillons de cellules tumorales en temps réel, nous prenons comme cas d'étude L'examen anatomo-pathologique.

2 Spécification informelle du banc de test anapath :

La spécification informelle est un type de cahier de charge pour exprimer les besoin de fonctionnement de ce système on présentant une description de l'architecture matérielle du système et une description des fonctions du système et les paramètres utilisé.

2.1 Aspect matériel :

2.1.1 L'Anatomie pathologique [24]

L'Anatomie pathologique est une spécialité médicale dont le rôle consiste à poser ou confirmer un diagnostic sur base d'analyses des tissus et/ou des cellules prélevées chez le patient. Ce diagnostic a un rôle majeur dans l'évaluation du pronostic clinique et dans le choix thérapeutique.

L'examen anatomo-pathologique comprend l'évaluation de facteurs pronostiques et la mise en évidence de mutations génétiques des cellules tumorales. Ceci permettant de prédire l'efficacité des thérapies.

Matériel : L'ensemble de verrerie et instruments, et appareils utilisées au cours de cette étude est récapitulé dans le tableau suivant :

Verrerie et instruments	Appareillage dispositif	Produits chimique et milieu de culture
-Scalpel et bistouri	-Appareil photo	-l'eau distillé
-Ciseaux	-Microtome	-liquide de fixation (formol 10%°
-Couteau	-Automate	-paraffine
-Flacons	-Etuve type Memmert	-xylène
-Marqueurs	U10	-Hemalaun
-Blouse	-Microscope optique	-acide alcool
-Gants	-Bain marie	-alcools (à85° ,95° et 100°)
- Pinces		
-Pinceaux		
-Porte-bloc		
-Lames et lamelles		
-Cassettes		
-Moules métalliques		
-Un crayon		
-Aiguilles montées de dissection.		

FIGURE 4.1 – Récapitulatif du matériel utilisé.

2.1.1.1 Examen microscopique :

— L'étude de la structure des tissus et des cellules qui les composant
— Préparation du microscope optique confocale ou électronique (une biopsie, autopsie, micropsie, chergui).
— Afin de réalisé cette étude ; nous avons utilise 5 méthodes

2.1.1.1.1 Traitement des tissus : Dans cette étape, les échantillons prélevés sont rincés dans l'eau distillée pendant quelques minutes puis découpes en des petits fragments de 0.5 cm d'épaisseur, qui sont mis dans des cassettes. Ces derniers sont placés dans un bocal contenant du formol à 10% dilué.

Déshydratation : est une étape utilisée pour d'éliminer l'eau intracellulaire pour pouvoir réaliser une coupe fine. Ceci par le passage de cassettes dans des bains

d'éthanol à concentration croissante (70%, 80%, 95%, 100%) (Tableau 4.2), qui déroulement dans un appareil (Leica TP1020)à circuler automatique (Figure 4.3), qui assure une agitation continuelle des paniers contenant les cassettes.

Nettoyage : ce fait par un liquide intermédiaire (xylène) afin d'éliminer les traces d'alcool absolu, suivi par l'infiltration par paraffine dissout à 56°C.

	Réactif	Duré
1	Formol 10%	1h
2	Formol 10%	1h
3	Ethanol 70%	11 à 12h
4	Ethanol 80%	11 à 12h
5	Ethanol 95%	11 à 12h
6	Ethanol 100%	1h
7	Ethanol 100%	1h
8	Ethanol 100%	1h
9	Xylène	11 à 12h
10	Xylène	11à 12h
12	Paraffine	2h

FIGURE 4.2 – Programmation de l'automate.

FIGURE 4.3 – Automate LEICA TP 1020

FIGURE 4.4 – Centre d'enrobage LEICA Arcadia C 2615 et Arcadia H 2224

2.1.1.1.2 Inclusion et mise en blocs : Les échantillons sont mis dans des cassettes puis imprégnées dans paraffine fondus de 54°à 56° (Figure 4.4) pendant 4 h.

Le paraffine est collé dans des moules en acier inoxydable qui contenant l'échantillon, après refroidissement à -2°C un bloc de paraffine est formé.

2.1.1.1.3 Confection des coupes histologiques : Le passage de bloc de paraffine dans un microtome (Figure 4.5), permet de réaliser des sections de 5 µm d'épaisseur, après mise les coupes dans un bain-marie à 45°C (Figure 4.6), pour faire un bon étalement puis récupérées sur les lames port l'objet. Ensuite, les lames sont séchées pendant 5 à 10 minutes à température ambiante, puis elles sont mises dans une étuve à 60°C pendant 1h, pour éliminer la paraffine (Figure 4.7).

FIGURE 4.5 – Microtome rotatif de type LEICA 2125.

FIGURE 4.6 – Bain-marie de type LEICA

FIGURE 4.7 – Etuve Memmert U10.

2.1.1.1.4 Coloration : Après séchage à l'étuve à 37°C pendant au moins deux heures, les lames sont colorées en Hématoxyline-Eosine dont l'hématoxyline colore les noyaux en violet, et l'éosine colore le cytoplasme en rose. Cette coloration a été effectuée manuellement selon le Protocol suivant :

— Déparaffinage par un passage dans deux bains de xylène de 15minutes chacun.

— Réhydratation par passage dans deux bains d'éthanol absolu pendant 5 minutes.
— Un bain d'alcool à 70°C pendant 5 minutes.
— Coloration avec l'hématoxyline pendant 25 minutes.
— Rinçage dans un l'eau de robinet pendant 15 minutes.
— Coloration à l'éosine pendant 15 minutes.
— Lavage à l'eau pour éliminer l'excès de colorant.
— Déshydratation dans l'alcool à 70°C pendant 10 minutes puis dans l'alcool absolu pendant 3 minutes.
— Séchages des lames par papier buvard.
— Clarification dans le xylène pendant 15 minutes.
— Montage des lamelles à l'aide du baume de canada en prenant soins de dégager les bulles d'air.

2.1.1.2 Lecture des lames (observation) : Après avoir été fixées, étalées, et colorées, les lames ont été observées sous microscope optique (Figure 4.8).

FIGURE 4.8 – Microscope lié à un appareil phoyonumérique.

2.2 Aspect fonctionnel :

3 Spécification formelle

La spécification formelle ait faite selon la méthode d'analyse fonctionnelle et opérationnelle des applications temps réel SA-RT, cette méthode permet de réaliser une description

graphique et textuelle de l'application en termes de besoins selon un modèle incrémental.

La SA-RT considère la spécification des systèmes temps réel selon trois points de vue orthogonaux :

— Fonctionnel
— Informationnel
— Dynamique

3.1 Noyau :

3.1.1 Spécification :

Aspect fonctionnel : La répartition du système de commande de machine d' anatomie-pathologique industriel en une hiérarchie de fonctions selon la méthode SA-RT est représentée par le schéma de contexte suivant.

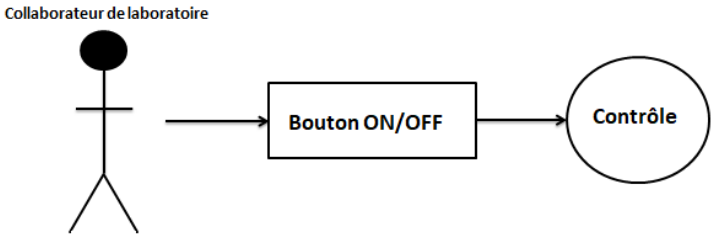

FIGURE 4.9 – Diagramme de contexte.

Aspect informationnel : Le cheminement de flot de données qui transporte les valeurs d'une certaine information qui manipuler par les fonctions est représentée par le diagramme de flot de données suivant.

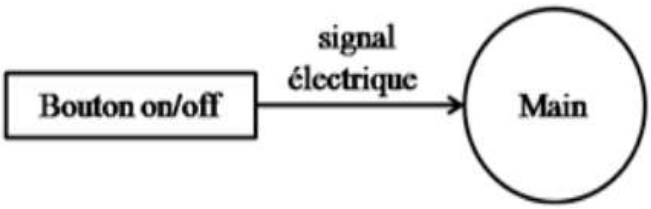

FIGURE 4.10 – Diagramme de flux de données"Start".

Aspect dynamique : Le contrôle du flux de données et l'activation des fonctions est exprimé par le flux de contrôle qui transporte les événements qui conditionnent directement ou indirectement l'exécution des processus de transformation de données, selon la méthode SA-RT est représenté par le schéma de flux de contrôle .

Les événements sont généralement liés à l'activation ou la désactivation :

— E pour l'activation "Activer".
— D pour la désactivation "Désactiver".
— T pour déclencheur (Start) "Trigger".

FIGURE 4.11 – Diagramme de flux de contrôle"Start".

3.1.1.1 Conception La méthode de conception multitâche "LACATRE" est utilisée pour permettre le passage d'un modèle de spécification basé sur une analyse fonctionnelle et structurée qui a été réalisée avec la méthode SA-RT pour programmer des entités :

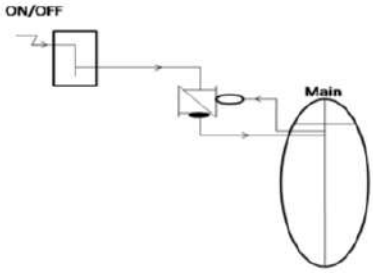

FIGURE 4.12 – Diagramme des tâches "Start".

3.1.1.2 Implémentation On a implémenter les objets LACATRE nécessaires pour le système étudié, en utilise la plateforme de l'exécutif temps réel FreeRTOS.

```
void RoutineDInterruption(void *pvParameters) {
 (void) pvParameters;
while(1){

printf("\ntask traitment de l'intruption \n");
  msg[0] = 'o';
  msg[1] = 'n';
  msg[2] = ' ';
  msg[3] = ' ';
  msg[4] = '\0';
  xQueueSendToBack( mailbox, &msg, ( TickType_t ) 10 );

vTaskSuspend(ROUTIN);
 vTaskResume(fin);
}
}
//================================finTravail=========================
void finTravail(void *pvParameters) {
 (void) pvParameters;
  vTaskSuspend(fin);
 printf( "\ntask fin travail \n\n");

  msg[0] = 'o';
  msg[1] = 'f';
  msg[2] = 'f';
  msg[3] = ' ';
  msg[4] = '\0';

  xQueueSendToBack( mailbox, &msg, ( TickType_t ) 10 );

    vSemaphoreDelete(SerialSemaphore);
    vQueueDelete( mailbox);
    vTaskDelete(lcd);
    vTaskDelete(ROUTIN);

    vTaskEndScheduler();
}
```

FIGURE 4.13 – Code FreeRTOS pour le noyau".

3.1.1.3 Transaction du noyau à l'incrément 1 :

3.1.1.3.1 Les nouvelles fonctionnalités :

1. Pour afficher la température.

   Input : une chaine de caractères.
   — D pour la désactivation "Désactiver".
   — Output : un signal numérique

Pour afficher la température.

FIGURE 4.14 – la tâche afficher..

02- Boite au lettre : Pour stocker les données. Input : une chaine de caractères. Output : une chaine de caractères

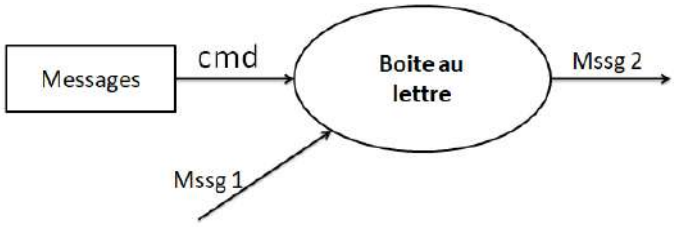

FIGURE 4.15 – La boite au lettre".

3.1.2 Incrément 01 :Déshydratation

3.1.2.1 Spécification :

Aspect fonctionnel :

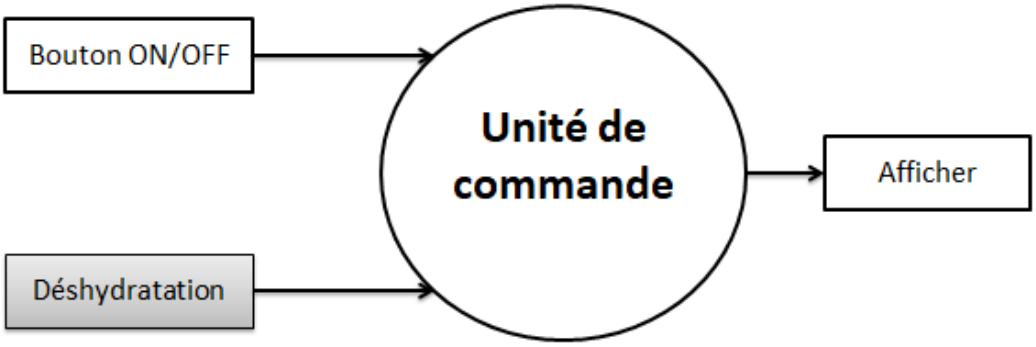

FIGURE 4.16 – Diagramme de contexte "Déshydratation".

Aspect fonctionnel :

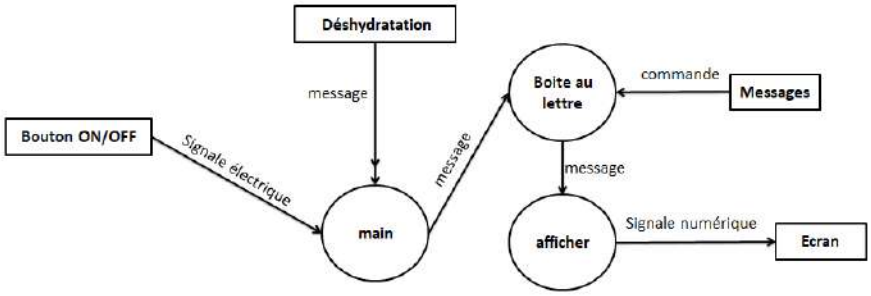

FIGURE 4.17 – Digramme de flot de données "Déshydratation".

Aspect dynamique :

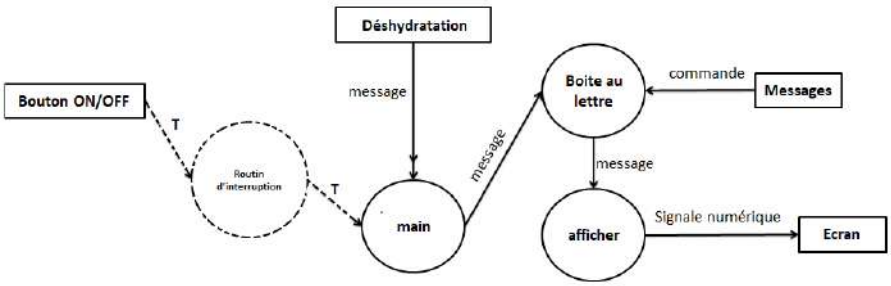

FIGURE 4.18 – Digramme de flot de contrôle "Déshydratation".

3.1.2.2 Conception :

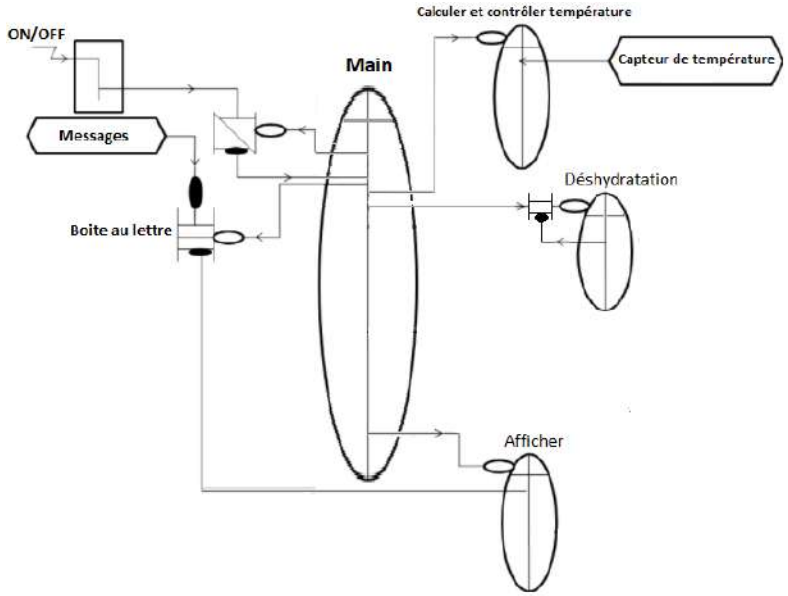

FIGURE 4.19 – Schéma multitâchE "Déshydratation".

3.1.2.3 Implémentation :

```
//============================== Déshydratation ====================================

void deshydra(void *pvParameters) {
 (void) pvParameters;

 printf("\nTask déshydratation \n");


for(i=0;i<5;i++){
      switch(i){     case 0:{
                              printf("\nFormol 10% \n");
                             for(int i=0;i<2;i++){
                            msg[0] = '1';
                            msg[1] = '0';
                            msg[2] = '%';
                            msg[3] = '\0';

                             xQueueSendToBack( mailbox, &msg, ( TickType_t ) 10);
                             vTaskDelay(100);//pendant 1h
                             }

                             break;}

                     case 1:{
                              printf("\nEthanol 70% \n");
                            msg[0] = '7';
                            msg[1] = '0';
                            msg[2] = '%';
                            msg[3] = '\0';

                            xQueueSendToBack( mailbox, &msg, ( TickType_t ) 10 );
                            vTaskDelay(1000);//pendant 11 à 12h
                            break;}
                     case 2:{
                     case 2:{
                               printf("\nEthanol 80% \n");
                            msg[0] = '8';
                            msg[1] = '0';
                            msg[2] = '%';
                            msg[3] = ' ';
                            msg[4] = '\0';
        |                   xQueueSendToBack( mailbox, &msg, ( TickType_t ) 10 );
                            vTaskDelay(1000);//pendant 11 à 12h
                            break;}
                     case 3:{

                                printf("\nEthanol 95% \n");
                             msg[0] = '9';
                             msg[1] = '5';
                             msg[2] = '%';
                             msg[3] = ' ';
                             msg[4] = '\0';
            |                xQueueSendToBack( mailbox, &msg, ( TickType_t ) 10 );
                             vTaskDelay(1000);//pendant 11 à 12h
                             break;}
                      case 4:{

                                printf("\nEthanol 100%\n");
                                for(int i=0;i<3;i++){
                             msg[0] = '1';
                             msg[1] = '0';
                             msg[2] = '0';
                             msg[3] = '%';
                             msg[4] = '\0';
                             xQueueSendToBack( mailbox, &msg, ( TickType_t ) 1000 );}
                             vTaskDelay(100);//pendant 1h

                                                        break;}

                                                    default: break;
                                            }

}
  printf("\nTask déshydratation terminé \n");
vTaskResume(fin);
  //vTaskResume(ROUTIN);
}
```

FIGURE 4.20 – Code FreeRTOS pour la tache afficher".

3.1.2.4 Exécution :

```
randa@randa-Vostro-3558:~/FreeRTOS1$ ./FreeRTOS-Sim
cliquer sur entree pour lancer la simulation
Running as PID: 3281
Timer Resolution for Run TimeStats is 100 ticks per second.

task afficher ************ machine:   off

task traitment de l'intruption

task afficher ************ machine:   on

Task déshydratation

Formol 10%

task afficher ************ machine:   10%

task afficher ************ machine:   10%

Ethanol 70%

task afficher ************ machine:   70%

Ethanol 80%

task afficher ************ machine:   80%

Ethanol 95%

task afficher ************ machine:   95%

Ethanol 100%

task afficher ************ machine:   100%

task afficher ************ machine:   100%

task afficher ************ machine:   100%

Task déshydratation terminé

task fin travail


task afficher ************ machine:   off
Cleaning Up, Exiting.
```

FIGURE 4.21 – L'exécution du tache "Déshydratation".

3.1.2.5 Chronogramme de la séquence d'exécution :

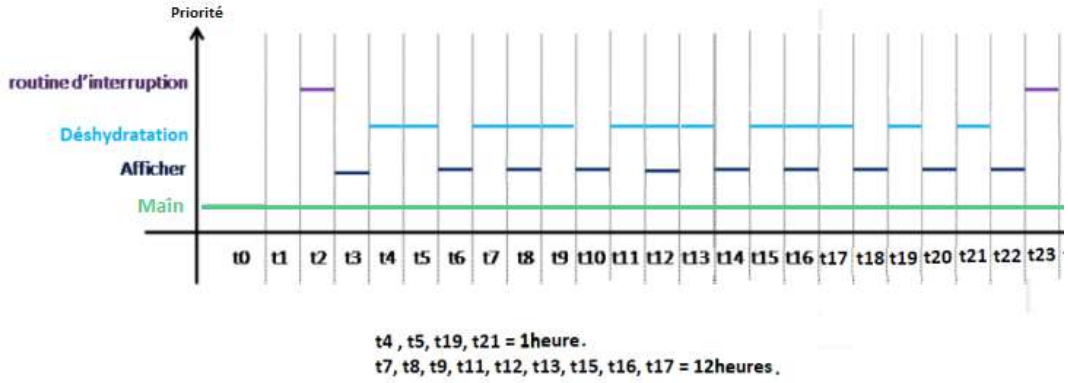

FIGURE 4.22 – Le chronogramme de la séquence d'exécution.

3.1.2.6 Transaction du l'incrément 01 à l'incrément 02 :

3.1.2.6.1 La nouvelle fonctionnalité

☐ Contrôler température : Pour contrôler la température du paraffine.
- — Entrée : un signal analogique.
- — Sortie : une valeur de température numérique en Celsius.

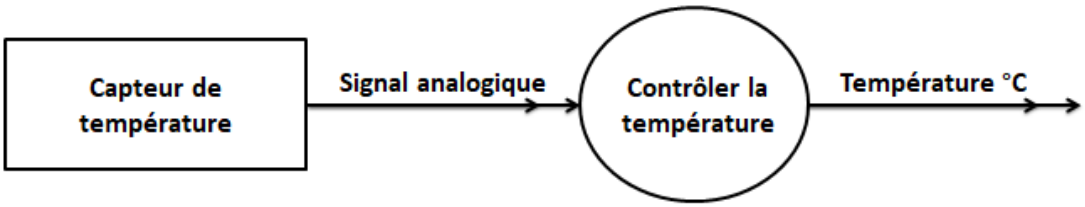

FIGURE 4.23 – La tache contrôler la température.

3.1.3 Incrément 03 : Paraffine

3.1.3.1 Spécification :

Aspect fonctionnel :

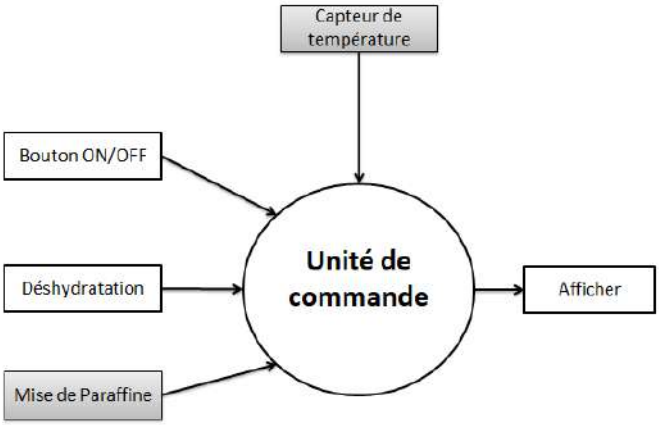

FIGURE 4.24 – Diagramme de contexte "Paraffine".

Aspect informationnel :

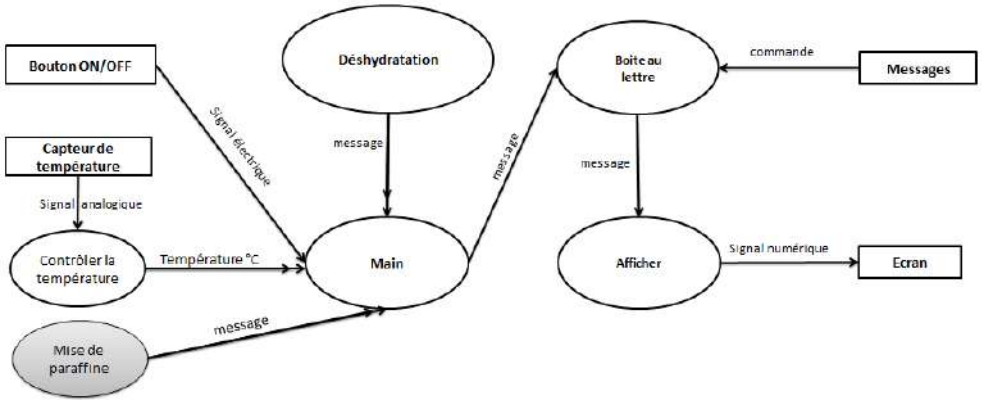

FIGURE 4.25 – Diagramme de flux de données "Paraffine".

Aspect dynamique :

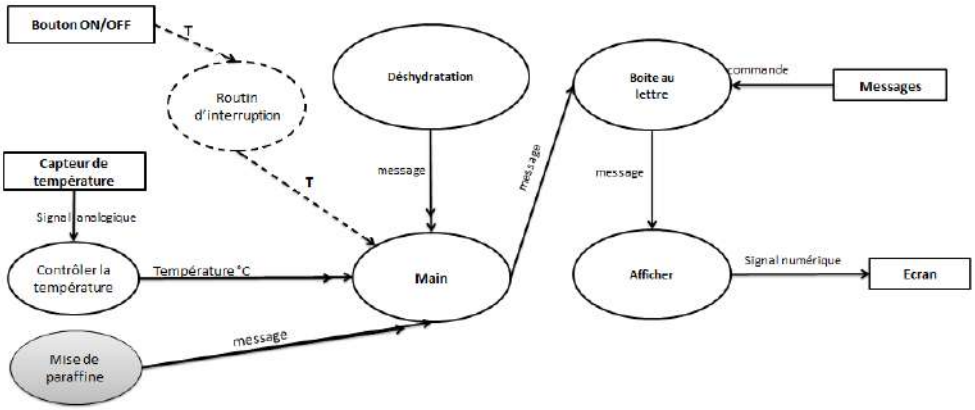

FIGURE 4.26 – Diagramme de flot de contrôle "Paraffine".

3.1.3.2 Schéma LACATRE :

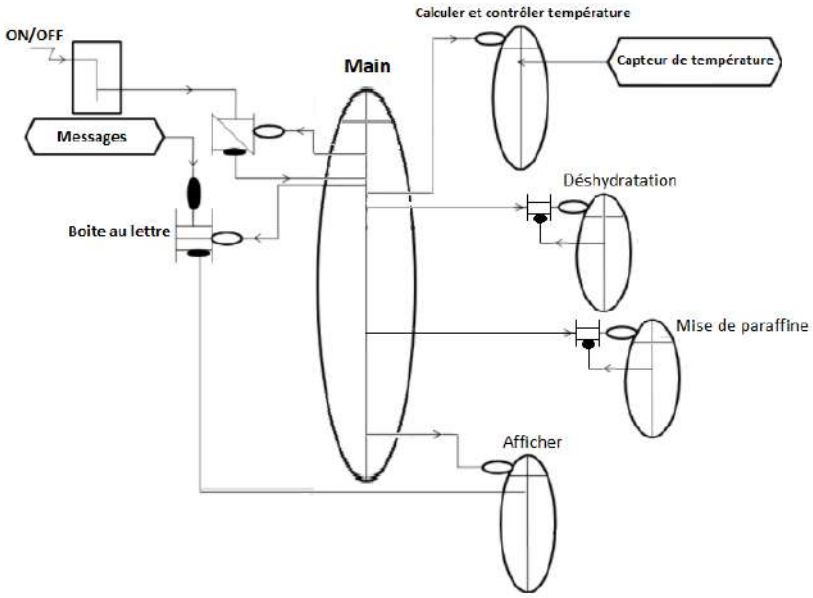

FIGURE 4.27 – Diagramme des taches "Paraffine".

.

3.1.3.3 Implémentation

```
60 //================================ Paraffine ========================
61 void paraffine(void *pvParameters){
62  (void) pvParameters;
63
64 temp ++;
65  if(temp = 5){ //la temperature ici doit etre 56°c
66   msg[0] = 'P';
67   msg[1] = 'a';
68   msg[2] = 'r';
69   msg[3] = 'r';
70   msg[4] = 'a';
71   msg[5] = 'f';
72   msg[6] = 'l';
73   msg[7] = 'n';
74   msg[8] = 'e';
75   msg[9] = ' ';
76   msg[10] = 'o';
77   msg[11] = 'n';
78   msg[12] = '\0';
79
80       printf("Task temperature de paraffine est = %d °C \n",temp);
81  xQueueSendToBack( mailbox, &msg, ( TickType_t ) 10 );
82
83        printf("Task Infusion de paraffine terminé \n \n",temp);
84        vTaskDelay(1000);//pendant 4h
85   }
86
87  temp --;
88  if(temp = -2){ //la temperature ici doit etre 56°c
89
90     printf("Task temperature = %d °C \n",temp);
91    }
92    printf("\nTask formation de paraffinee \n");
93
94   msg[0] = 'o';
95   msg[1] = 'n';
96   msg[2] = '\0';
97   msg[3] = '\0';
98  xQueueSendToBack( mailbox, &msg, ( TickType_t ) 10 );
99  printf("\nTask formation de paraffine terminé\n");
100
101 vTaskResume(fin);
102  //vTaskResume(ROUTIN)
103
104 }
105
```

FIGURE 4.28 – Code FreeRTOS "Paraffine".

3.1.3.4 Exécution

FIGURE 4.29 – L'exécution du tache "Paraffine".

Chronogramme de la séquence d'exécution :

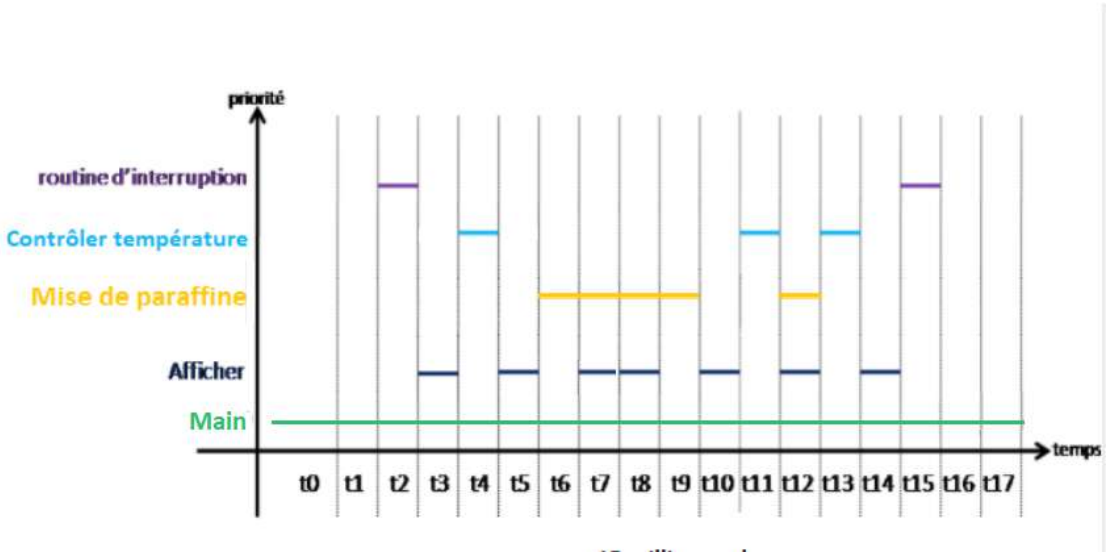

FIGURE 4.30 – Le chronogramme de la séquence d'exécution "Mise du Paraffine".

3.1.4 Incrément 3 : Coupes histologique

3.1.4.1 Spécification :

Aspect fonctionnel : Coupes histologique

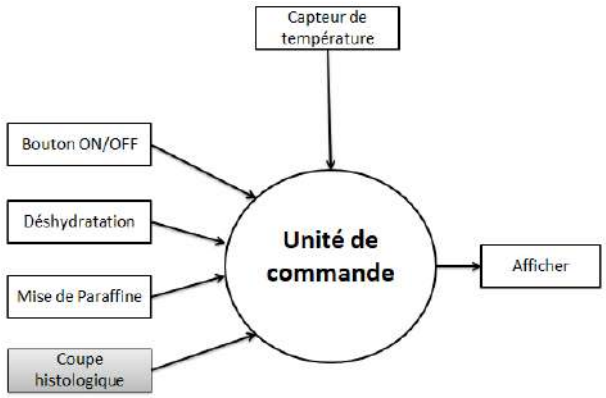

FIGURE 4.31 – Diagramme de contexte "Coupes histologique".

Aspect informationnel :

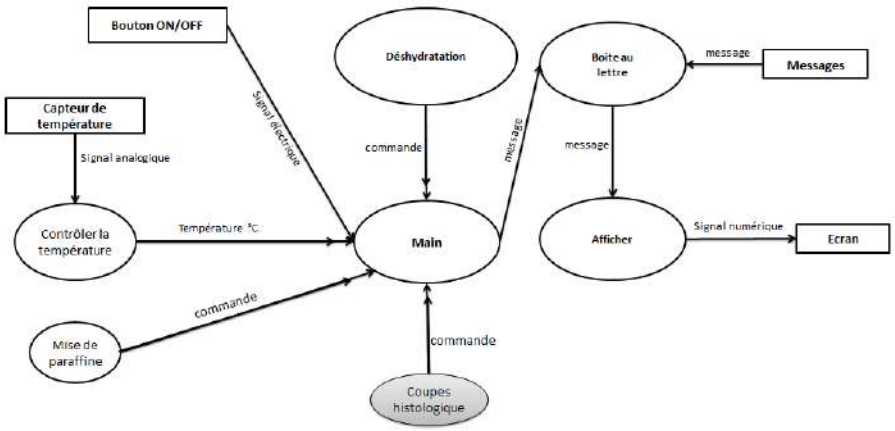

FIGURE 4.32 – Diagramme de flux de données "Coupes histologique".

Aspect dynamique :

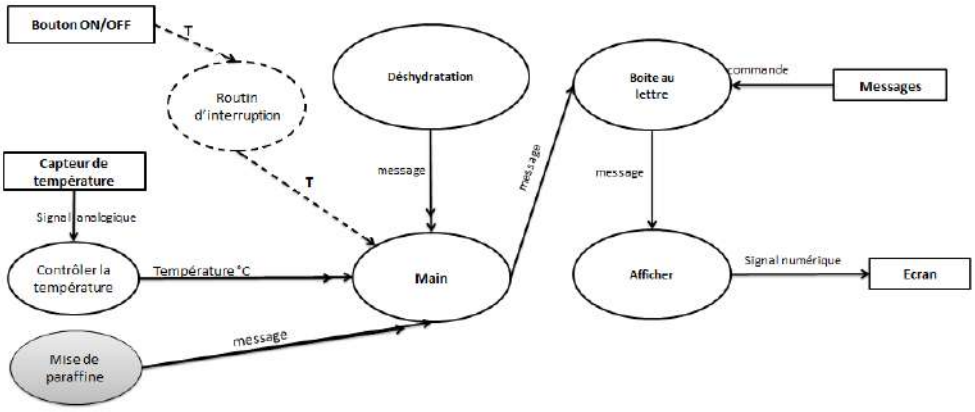

FIGURE 4.33 – Diagramme de flot de contrôle "Coupes histologique".

3.1.4.2 Schéma LACATRE :

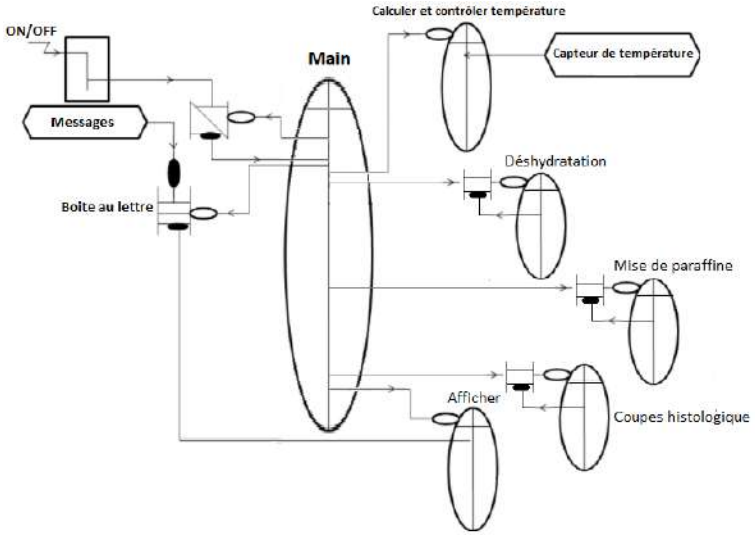

FIGURE 4.34 – Diagramme des taches "Coupes histologique".

3.1.4.3 Implémentation

```
58 //================================= Confection des coupes histologiques ====================
59 void coupe(void *pvParameters) {
60  (void) pvParameters;
61
62  int t = 5000; // 0.5cm -> 5000 µm
63
64
65  printf("\nTask Coupe histologique \n");
66  if(t <= 5000){
67
68   for(t =1; t <5000; t++){
69
70   msg[0] = 'c';
71   msg[1] = 'u';
72   msg[2] = 't';
73   msg[3] = ' ';
74   msg[4] = '\0';
75   xQueueSendToBack( mailbox, &msg, ( TickType_t ) 10 );
76
77   }
78
79  }
80   printf("\nTask Coupe histologique terminé\n");
81 vTaskResume(fin);
82   //vTaskResume(ROUTIN);
83 }
84
```

FIGURE 4.35 – Code FreeRTOS "Coupes histologique".

3.1.4.4 Exécution

```
randa@randa-Vostro-3558:~/FreeRTOS1$ ./FreeRTOS-Sim
cliquer sur entree pour lancer la simulation
Running as PID: 3552
Timer Resolution for Run TimeStats is 100 ticks per second.

task afficher ************ machine:   off

task traitment de l'intruption

task afficher ************ machine:   on

Task Coupe histologique

task afficher ************ machine:   cut

task afficher ************ machine:   cut

task afficher ************ machine:   cut

task afficher ************ machine:   cut

Task Coupe histologique terminé

task fin travail


task afficher ************ machine:   off
Cleaning Up, Exiting.
```

FIGURE 4.36 – L'exécution du tache "Coupes histologique".

3.1.4.5 Chronogramme de la séquence d'exécution :

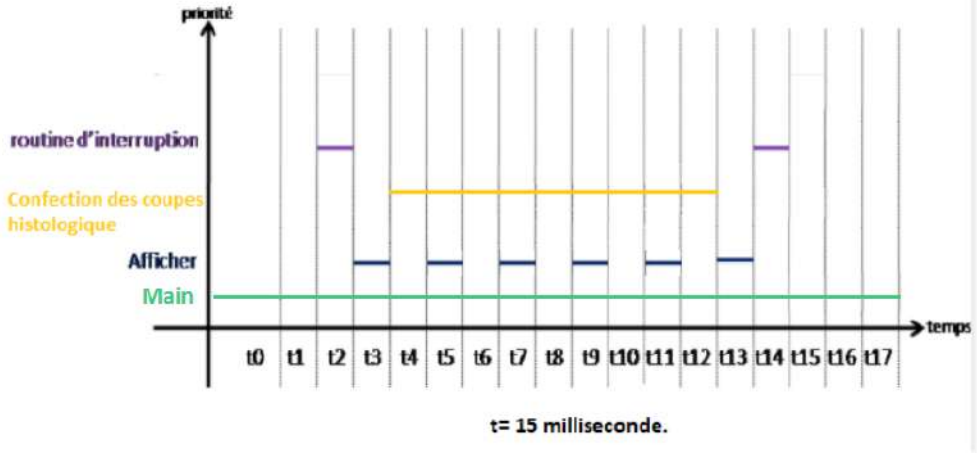

FIGURE 4.37 – Le chronogramme de la séquence d'exécution "Coupes histologique".

3.1.5 Incrément 4 : "Bain-marie

3.1.5.1 Spécification :

Aspect fonctionnel : Bain-marie

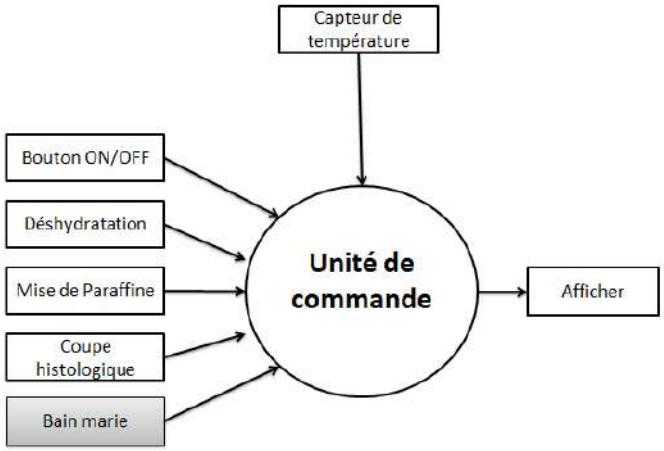

FIGURE 4.38 – Diagramme de contexte "Bain-marie".

Aspect informationnel :

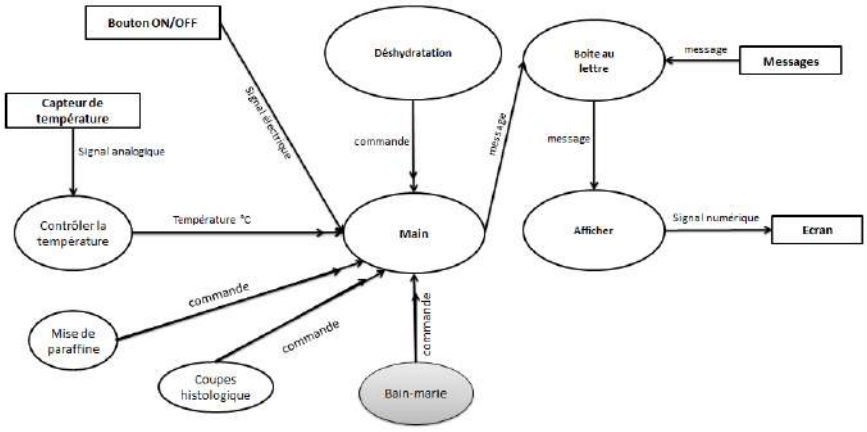

FIGURE 4.39 – Diagramme de flux de données "Bain-marie".

Aspect dynamique :

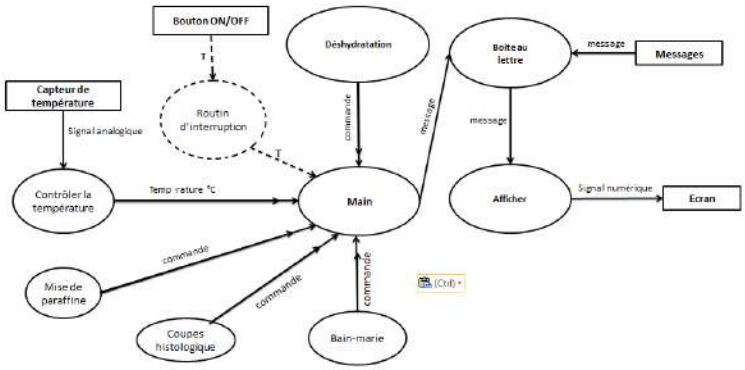

FIGURE 4.40 – Diagramme de flot de contrôle "Bain-marie".

3.1.5.2 Schéma LACATRE :

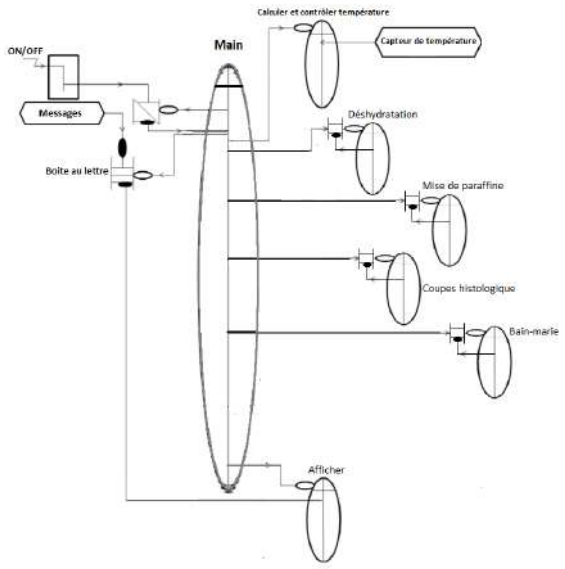

FIGURE 4.41 – Diagramme des taches "Bain-marie".

3.1.5.3 Implémentation

```
61 //================================ Bain-marie ================================
62 void BainMarie(void *pvParameters) {
63
64 //1er Etape: faire un bon étalement des tissus
65
66   printf( "\nMise les coupes du tussi dans un Bain-marie \n\n");
67
68   temp++ ;
69
70   if(temp = 45){  // la temperature doit etre 45°C
71
72     printf("Task temperature = %d °C \n",temp);
73     xQueueSendToBack( mailbox, &msg, ( TickType_t ) 10 );
74    }
75
76   printf("Task Bain-marie terminé \n\n\n");
77   //xQueueSendToBack( mailbox, &msg, ( TickType_t ) 10 );
78
79
80 //2eme Etape: récupérer les coupe de tissus sur les lames
81
82  printf("\n Placer sur les lames \n"); // la température dans cette étape est ambiante
83
84     for(t =0; t <10; t++){
85
86         xQueueSendToBack( mailbox, &msg, ( TickType_t ) 10 );
87         printf("Lame %d  \n",t);
88     }
89
90     vTaskDelay(100);//les lames sont séchées pendant 5 à 10 minutes
91   printf("Task lame terminé \n\n\n");
92 vTaskResume(fin);
93   //vTaskResume(ROUTIN);
94 }
```

FIGURE 4.42 – Code FreeRTOS "Bain-marie".

3.1.5.4 Exécution

```
randa@randa-Vostro-3558:~/FreeRTOS1$ ./FreeRTOS-Sim
cliquer sur entree pour lancer la simulation
Running as PID: 3768
Timer Resolution for Run TimeStats is 100 ticks per second.

task afficher ************ machine:   off

task traitment de l'intruption

task afficher ************ machine:   on

Mise les coupes du tussi dans un Bain-marie

Task temperature = 45 °C

task afficher ************ machine:   on
Task Bain-marie terminé



 Placer sur les lames

task afficher ************ machine:   on
Lame 0

task afficher ************ machine:   on
Lame 1

task afficher ************ machine:   on
Lame 2

task afficher ************ machine:   on

task afficher ************ machine:   on

task afficher ************ machine:   on

task afficher ************ machine:   on

task afficher ************ machine:   on
Task lame terminé



task fin travail


task afficher ************ machine:   off
Cleaning Up, Exiting.
```

FIGURE 4.43 – L'exécution du tache "Bain-marie".

3.1.5.5 Chronogramme de la séquence d'exécution :

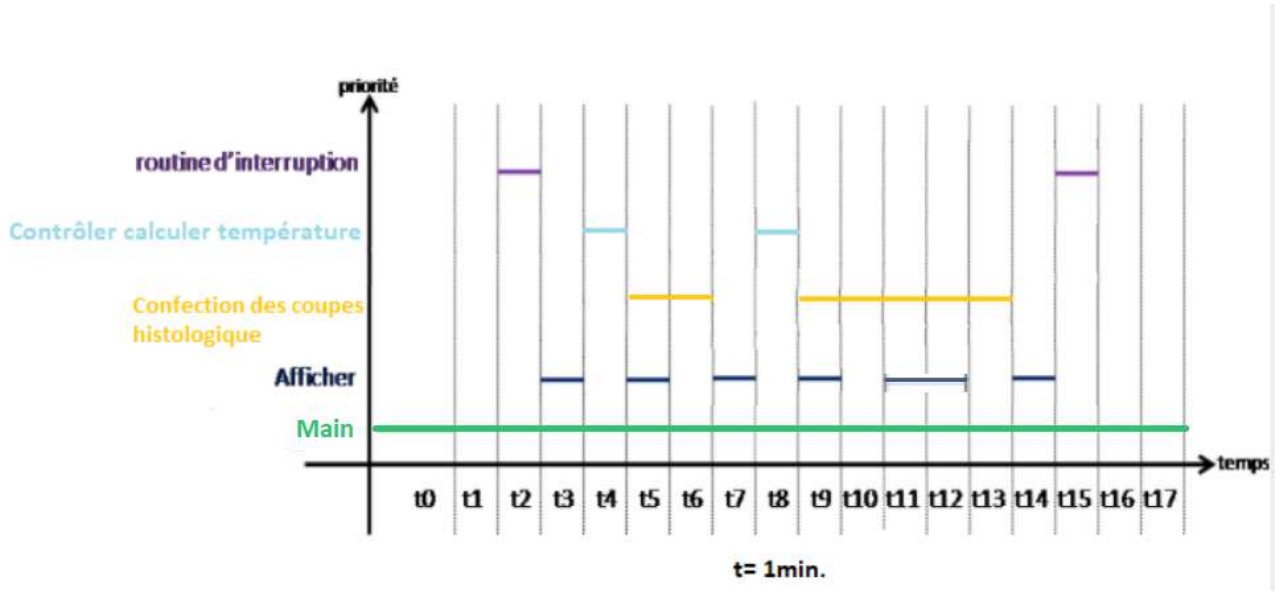

FIGURE 4.44 – Le chronogramme de la séquence d'exécution "Bain-marie".

3.1.6 Incrément 5 : "Etuve"

3.1.6.1 Spécification :

Aspect fonctionnel : Etuve

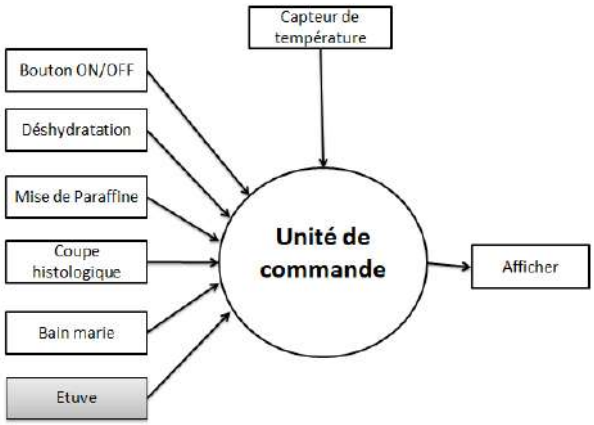

FIGURE 4.45 – Diagramme de contexte "Etuve".

Aspect informationnel :

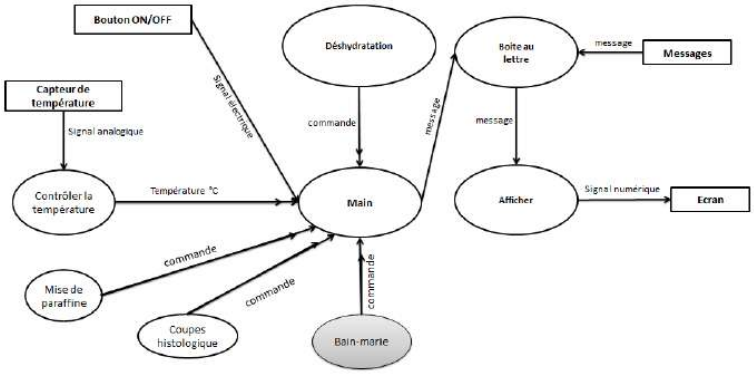

FIGURE 4.46 – Diagramme de flux de données "Etuve".

Aspect dynamique :

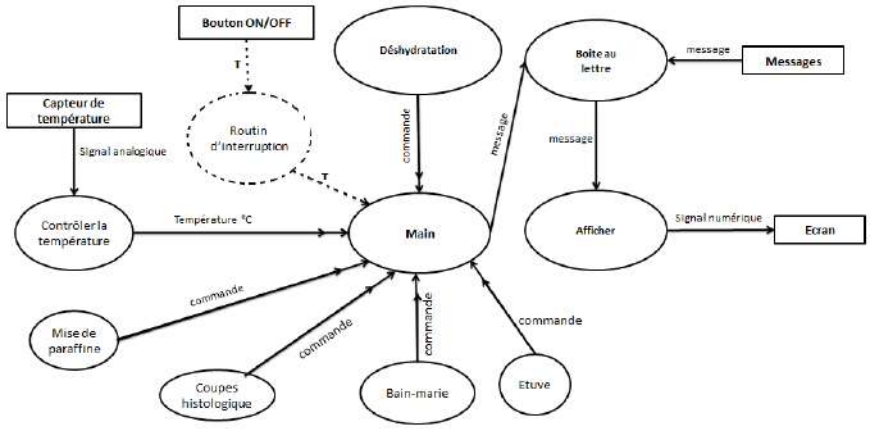

FIGURE 4.47 – Diagramme de flot de contrôle "Etuve".

3.1.6.2 Schéma LACATRE :

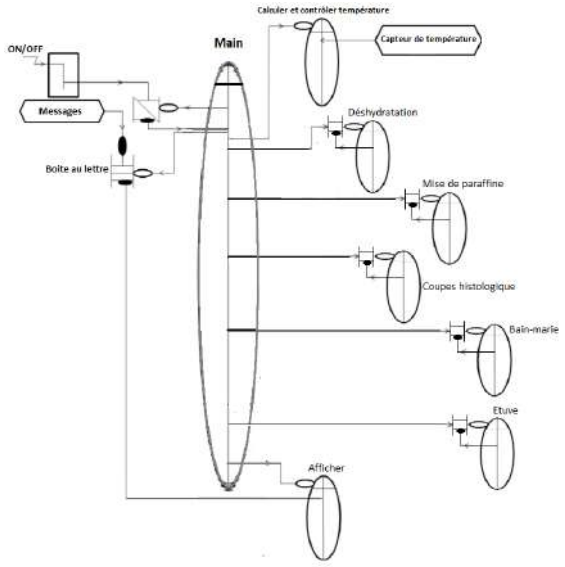

FIGURE 4.48 – Diagramme des taches "Etuve".

3.1.6.3 Implémentation

```
59 //================================ Etuve ================================
60
61 //Pour eliminer la paraffine
62
63 void etuve(void *pvParameters) {
64   printf( "\nEliminer la paraffine \n\n");
65
66  temp ++;
67  if(temp = 60){ //la temperature doit etre 60°C
68     printf("Task temperature = %d °C \n",temp);
69     xQueueSendToBack( mailbox, &msg, ( TickType_t ) 10 );
70     vTaskDelay(100);//les coupe de tissus sont mises dans le étuve pendant
71    }
72
73         printf("Task elimination de paraffine terminé\n");
74
75    vTaskResume(fin);
76   //vTaskResume(ROUTIN);
77 }
78
```

FIGURE 4.49 – Code FreeRTOS "Etuve".

3.1.6.4 Exécution

FIGURE 4.50 – L'exécution du tache "Etuve".

3.1.6.5 Chronogramme de la séquence d'exécution :

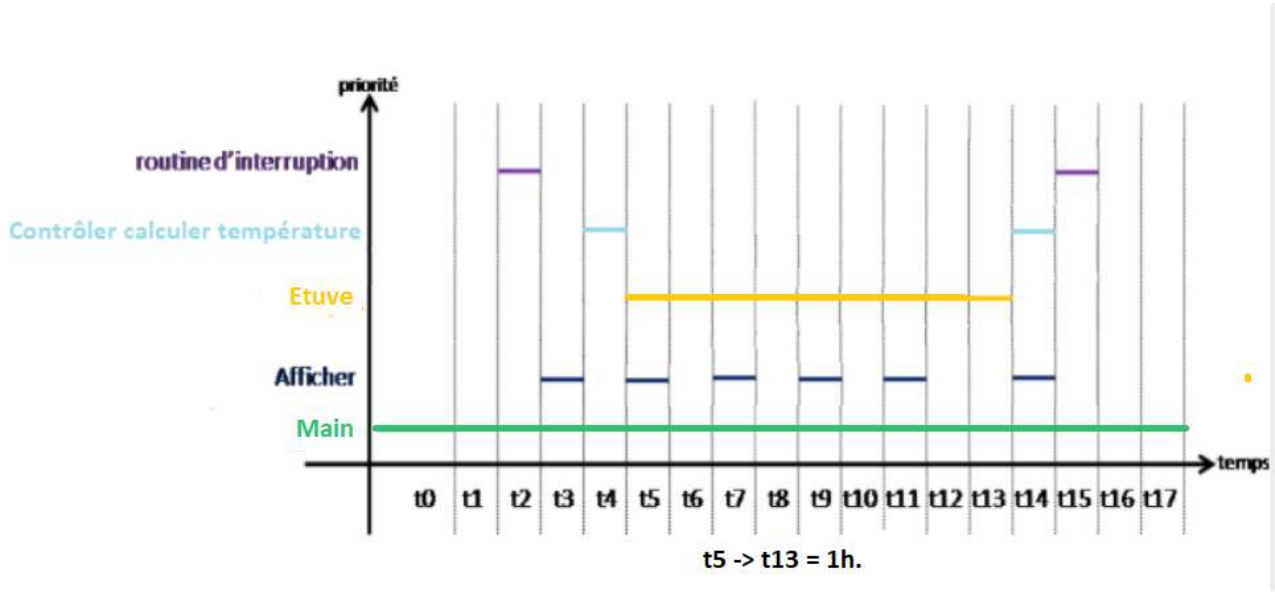

FIGURE 4.51 – Le chronogramme de la séquence d'exécution "Etuve".

3.1.7 Incrément 6 :Coloration

3.1.7.1 Spécification :

Aspect fonctionnel :

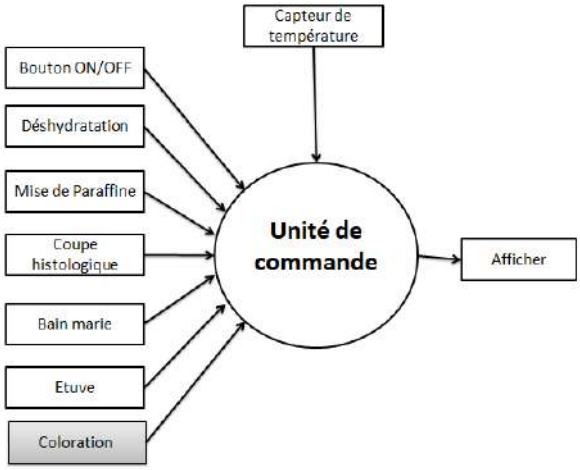

FIGURE 4.52 – Diagramme de contexte "Coloration".

Aspect informationnel :

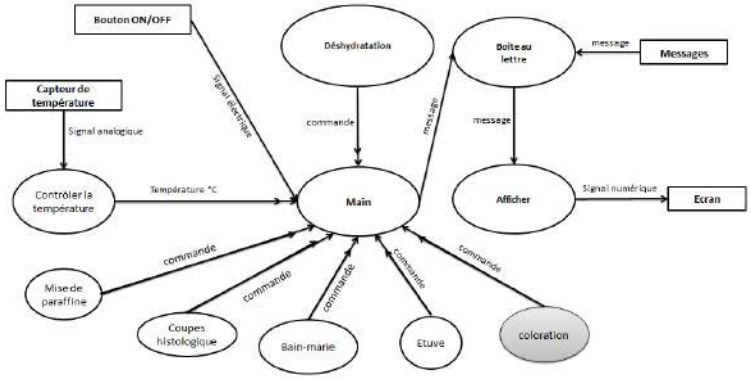

FIGURE 4.53 – Diagramme de flux de données "Bain-marie".

Aspect dynamique :

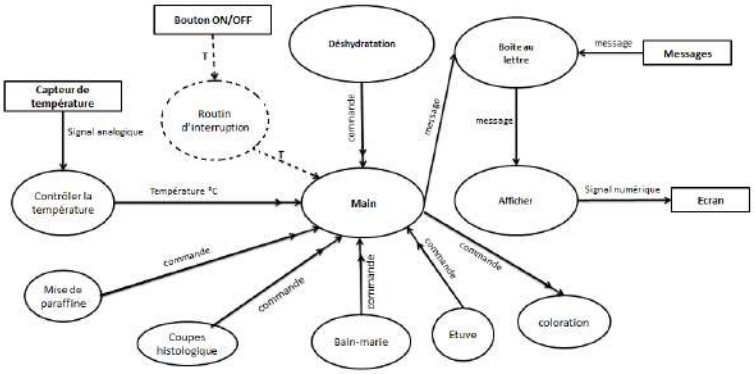

FIGURE 4.54 – Diagramme de flot de contrôle "Coloration".

3.1.7.2 Schéma LACATRE :

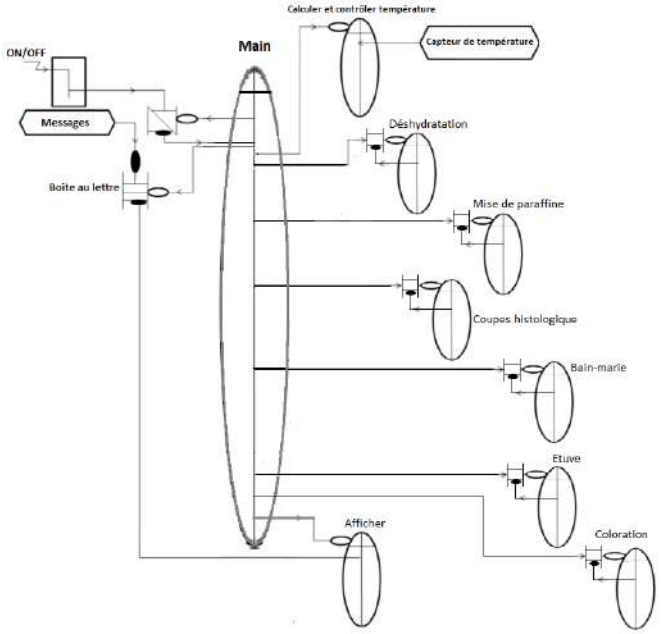

FIGURE 4.55 – Diagramme des taches "Coloration".

3.1.7.3 Implémentation

```
//============================== coloration ================================

void coloration(void *pvParameters) {
 (void) pvParameters;

printf("\nTask Coloration \n");


for(int i=0;i<12;i++){
       switch(i){      case 0:{ //Déparaffinage
                               for(int k=0 ; k<2; k++){
                                 printf("\n Xylène \n");

                                xQueueSendToBack( mailbox, &msg, ( TickType_t ) 10 );
                               // vTaskDelay(100);//pendant 15 min
                                                  }


                                break;}

                       case 1:{//Réhydratation
                                 printf("\nEthanol\n");

                                xQueueSendToBack( mailbox, &msg, ( TickType_t ) 10 );
                                //vTaskDelay(100);//pendant 5 min
                               break;}

                       case 2:{
                                  temp ++;
                                  if(temp = 70){
                                  printf("\n alcool à %d °C \n",temp);

                                xQueueSendToBack( mailbox, &msg, ( TickType_t ) 10 );}
                              // vTaskDelay(100);//pendant 5 min
                       case 3:{

                                  printf("\nColoration avec l'hématoxyline \n");

                                xQueueSendToBack( mailbox, &msg, ( TickType_t ) 10 );
                                //vTaskDelay(100);//pendant 25 minutes
                               break;}
                       case 4:{

                                  printf("\nRinçage dans un l'eau\n");

                                xQueueSendToBack( mailbox, &msg, ( TickType_t ) 10 );
                              //vTaskDelay(100);//pendant 5 minutes

                                                          break;}
                       case 5:{

                                  printf("\n Coloration à l'éosine\n");
```

FIGURE 4.56 – Code FreeRTOS "Coloration"1.

```
113
114                                 xQueueSendToBack( mailbox, &msg, ( TickType_t ) 10 );
115                               //vTaskDelay(100);//pendant 15 minutes.
116
117                                                                     break;}
118                          case 6:{
119
120                                    printf("\n  Lavage à l'eau pour éliminer l'excès de colorant.\n
121
122                                 xQueueSendToBack( mailbox, &msg, ( TickType_t ) 10 );
123
124                                                                     break;}
125                          case 7:{
126
127                                    printf("\n  Déshydratation dans l'alcool à 70°C \n");
128
129                                 xQueueSendToBack( mailbox, &msg, ( TickType_t ) 10 );
130                                 //vTaskDelay(100);//pendant 10 minutes
131                                 break;}
132
133                          case 8:{
134                                 temp ++;
135                                  if(temp = 70){
136                                    printf("\n  Déshydratation dans l'alcool à %d °C \n",temp);
137
138                                 xQueueSendToBack( mailbox, &msg, ( TickType_t ) 10 );}
139                                 //vTaskDelay(100);//pendant 10 minutes
140
141                                                                     break;}
142
143                          case 9:{
144
145                                    printf("\n  Déshydratation dans l'alcool absolu \n");
146
147                                 xQueueSendToBack( mailbox, &msg, ( TickType_t ) 10 );
148                                 //vTaskDelay(100);//pendant 3 minutes.
149
150                                                                     break;}
151                          case 10:{
152
153                                    printf("\n Séchages des lame \n");
154
155                                 xQueueSendToBack( mailbox, &msg, ( TickType_t ) 10 );
156
158                                                                     break;}
159
160                          case 11:{
161
162                                    printf("\n Clarification dans le xylène \n");
163
164                                 xQueueSendToBack( mailbox, &msg, ( TickType_t ) 10 );
165                                 //vTaskDelay(100);//pendant 15 minutes.
166
167                                                                     break;}
168
169                                                                default: break;
170                                                          }
171
172 }
173  printf("\nTask Coloration terminé \n");
174
175 vTaskResume(fin);
176   //vTaskResume(ROUTIN);
177 }
178
```

FIGURE 4.57 – Code FreeRTOS "Coloration"2.

3.1.7.4 Exécution

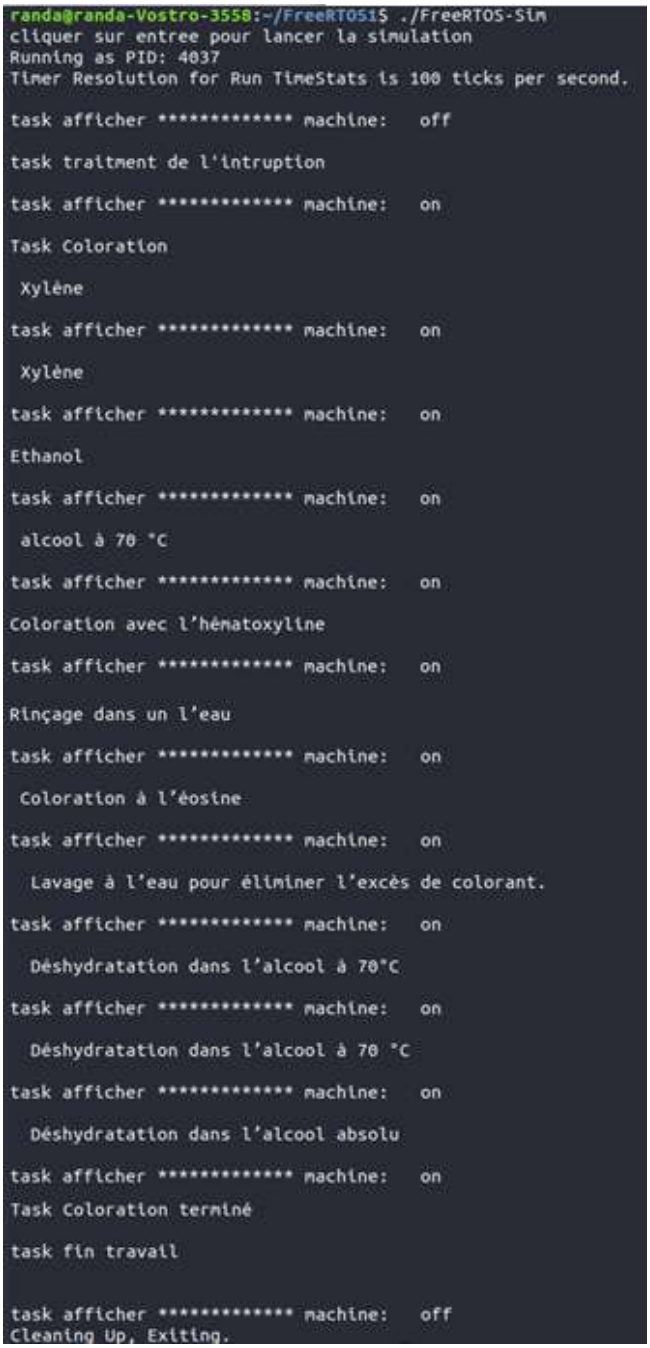

FIGURE 4.58 – L'exécution du tache "Coloration".

3.1.7.5 Chronogramme de la séquence d'exécution :

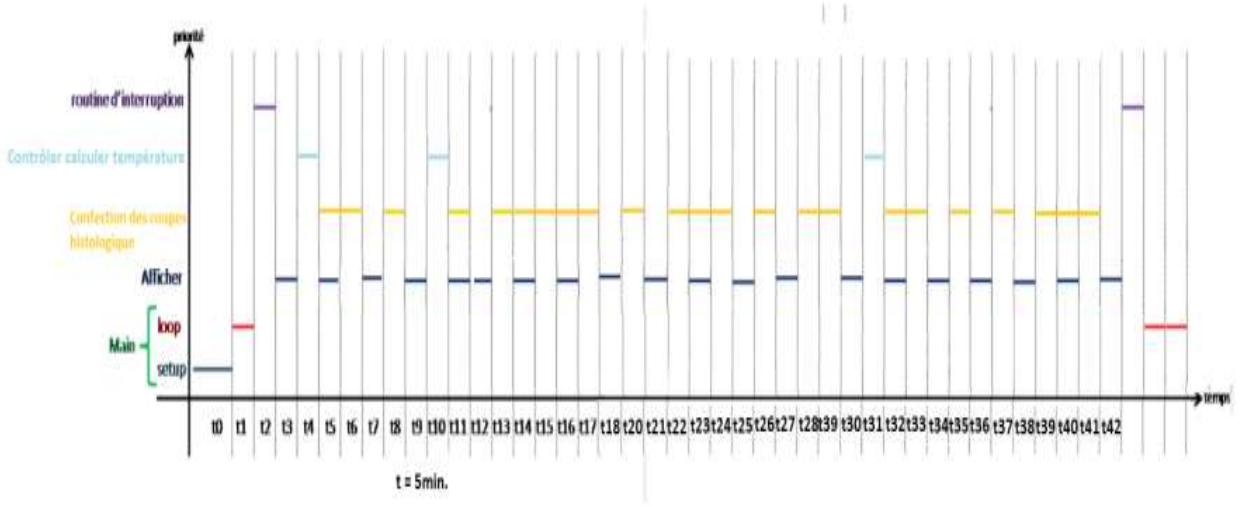

FIGURE 4.59 – Le chronogramme de la séquence d'exécution "Coloration".

Une fois que l'échantillon est passé par toutes ces étapes, nous avons précédemment présenté toutes leurs conceptions. L'échantillon est présent dans des lames pour le diagnostic au microscope.

3.1.8 La conception de BioMEMS

Nous avons présenté une conception détaillée des étapes de traitemet les échantillon de cellules sont traitées afin qu'elles puissent être diagnostiquées, mais cette méthode n'est pas pour tous les échantillons, car il existe des échantillons qui n'ont pas besoin de passer par toutes ces étapes, comme les échantillon de cellules de le cancer du col de l'utérus. Et donc, nous avons conçu un Capteur BioMEMS qui diagnostiquer tous les échantillons tout en donnant une analyse des ces échantillons et sur leur nature en temps réel.

Ces mesures peuvent avoir différents objectifs comme par exemple : La détection de tissus cancéreux.

La cellule de mesure : capteur à électrodes inter digitées , nous avons trois couches : substrat (verre) , électrode (Platine) et résine
la taille du substrat de cote (5000 micromètre) et de hauteur (1000 micromètre).

(a) ***Les électrodes de BioMEMS***

Parameter	Description
$LongueurL(mm)$	2
$NombreN$	40
$W(um)$	30 x 5= 150
$S(Um)$	20 x 5= 100

TABLE 4.1 – les parametres de BioMEMS

(b) ***La structure de capteurs BioMEMS*** Ce capteur contient un réservoir, ce dernier forme un puits carré, et c'est là que l'on va placer l'échantillon biologique à mesurer.

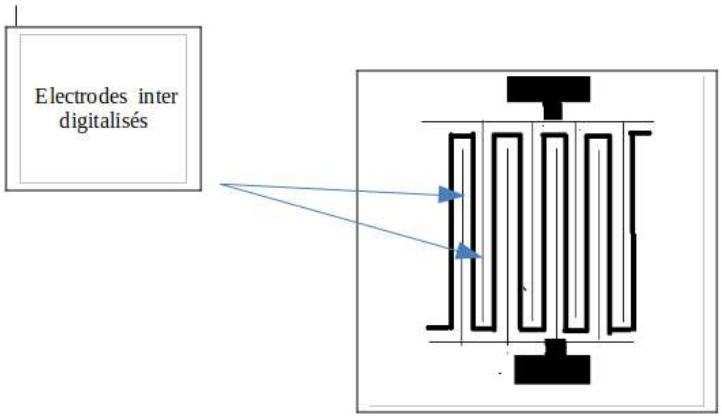

FIGURE 4.60 – Conception générale de BioMEMS.

3.2 La base de donnée

Nous avons créé une interface de base de données stocker (et rechercher) les informations sur les résultats obtenus à partir de capteurs de microscope et de BioMEMS, et qui dédiée aux patientes nomades du cancer du col de l'utérus où elles sont diagnostiquées sur la base de l'analyse d'échantillons cellulaires, ce qui est l'objectif de notre étude.

3.2.1 Architecture du système :

3.2.1.1 Linterface de Bienvenue et de sélection d'utilisation : Il dispose d'une interface d'entré, suivie d'une interface dans laquelle l'utilisateur choisit le type d'utilisation.

FIGURE 4.61 – Linterface de Bienvenue et de sélection d'utilisation.

3.2.1.2 Authentification : Ce fenêtre s'affiche à l'écran, elle demandera d'introduire le nom d'utilisateur et le mot de passe d'un Medcin que déjà attribué par l'administrateur pour commencer à utiliser et gérer les information des patients.

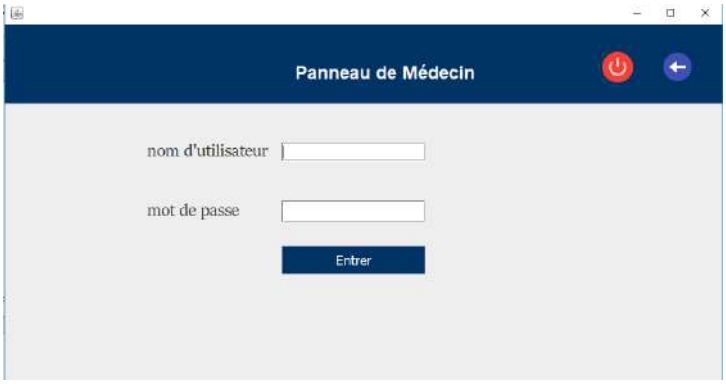

FIGURE 4.62 – Fenêtre dd'authentification de Medcin.

3.2.1.3 Interface des choix de gynécologie : Une fois que le médecin s'est authentifié, l'interface de sélection de la maladie apparait, et qui sera le cancer du col de l'utérus puisque c'est l'objet de l'étude dans cette thèse.

FIGURE 4.63 – Choix de gynécologie.

3.2.1.4 Interface Panneau de Patient : Où un utilisateur peut ajouter, supprimer et rechercher un patient.

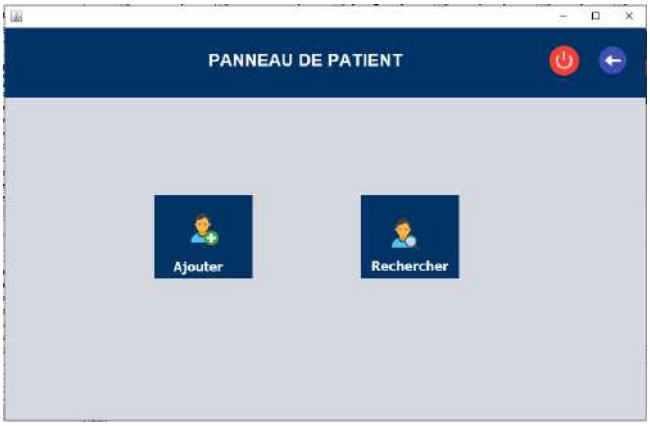

FIGURE 4.64 – Interface panneau de patient.

3.2.1.5 Interface Ajouter Patient : A l'arrivée d'un nouveau patient le médecin l'utilisateur remplit une nouvelle fiche.

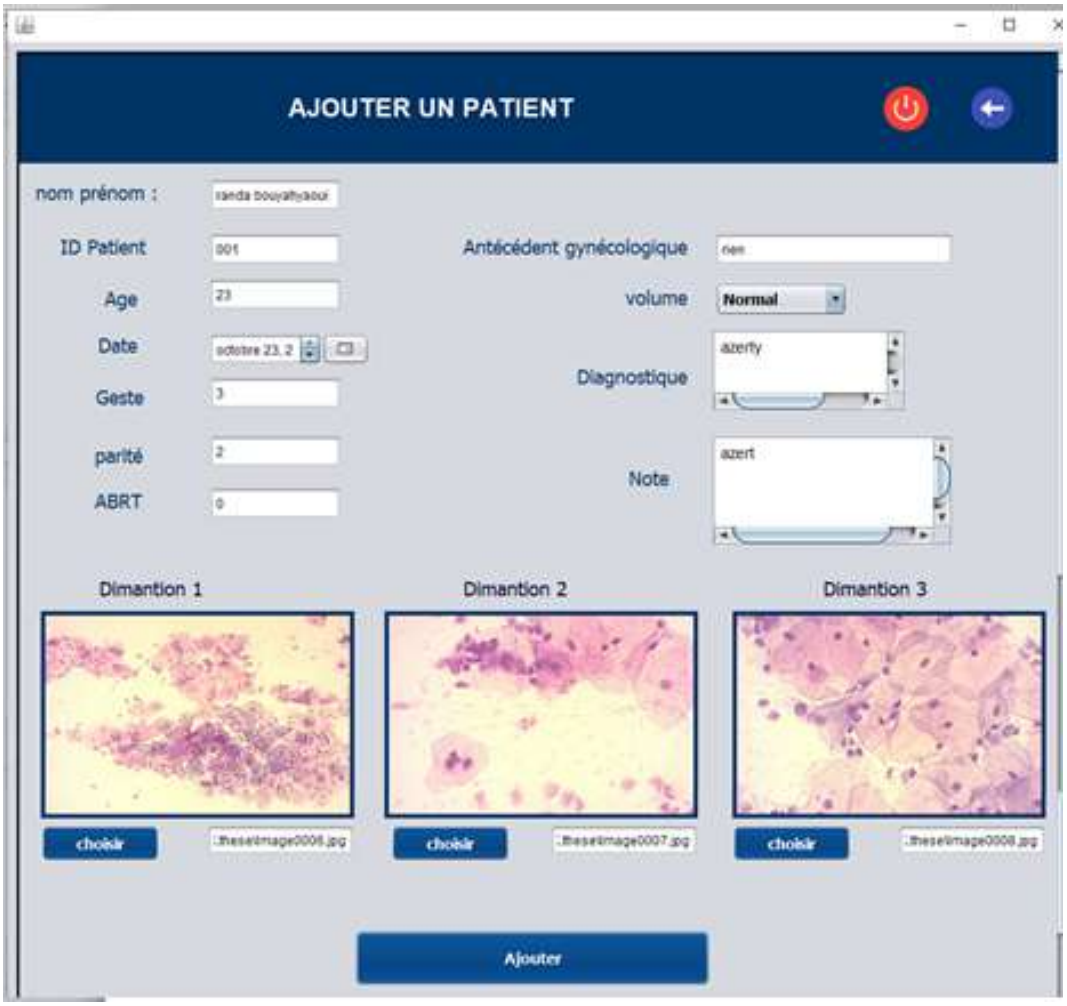

FIGURE 4.65 – Interface Ajouter Patient d'utérs.

3.2.1.6 Interface de recherche de patient :

Pour rechercher, nous avons besoin :
— par ID de patient.
— par date de consultation.

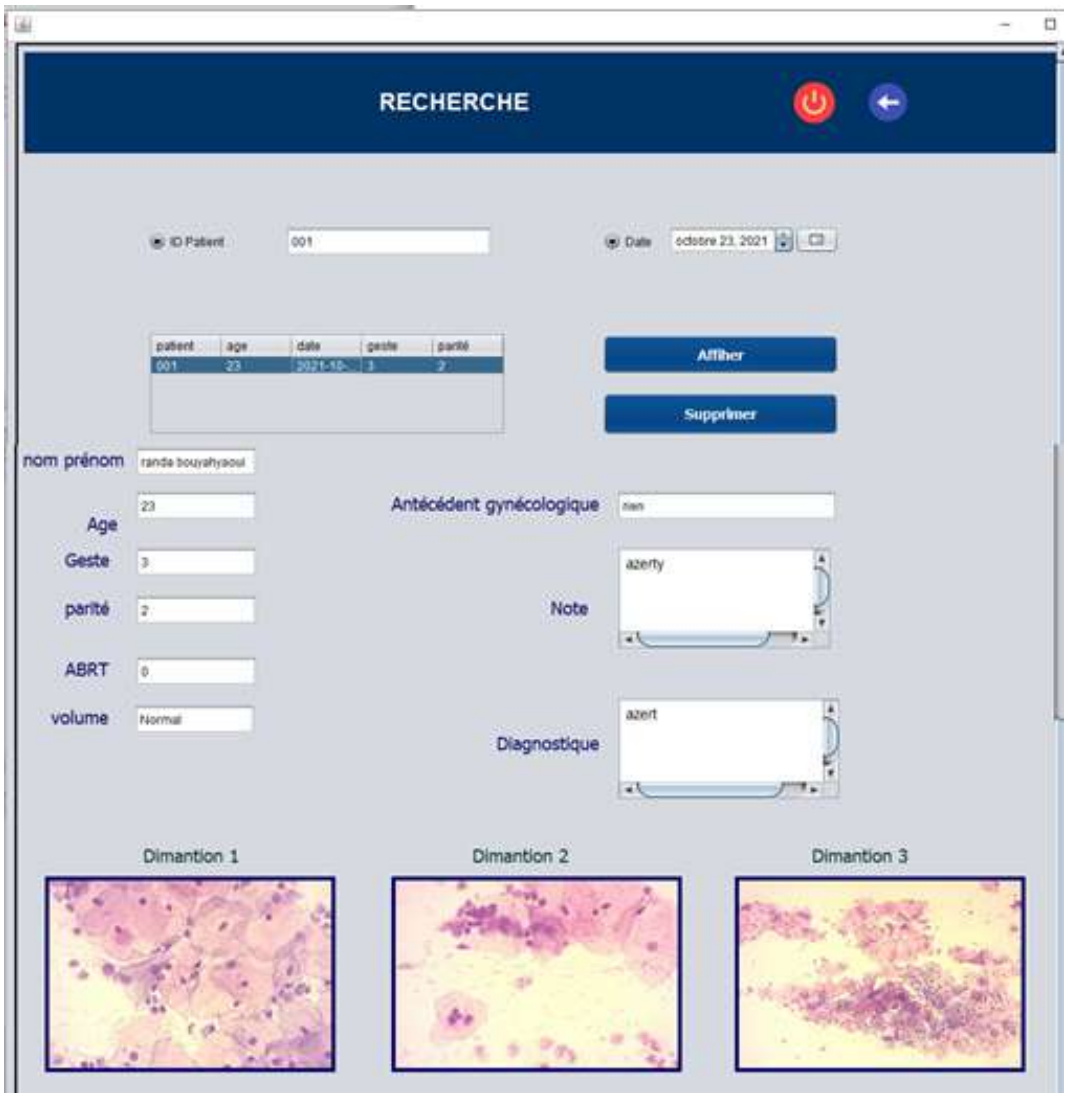

FIGURE 4.66 – Interface de recherche d'un patient.

4 Conclusion

Dans ce chapitre, nous avons démontré une conception détaillée avec l'implemantation de FreeRTOS du processus de traitement des échantillons cellulaires aussi une conception genérale de Biomems et on a réalisé une interface de base de données qui obtient des images de ces cellules examinées.

Conclusion générale

Dans ce travail, nous visons à réaliser des «simulateurs» pour aider au diagnostic cellulaire et au stockage des paramètres.

Nous avons rencontré de nombreuses difficultés, en termes de compréhension du fonctionnement des machines qui traitent les cellules tissulaires, mais on a pu créer un système qui permettront à l'avenir de diagnostiquer les cellules de manière séquentielle et spécifique avec précision, et une conception en BioMEMS. En plus de fournir une interface pour l' acquisition des informations de patients produites par le laboratoire.

Nous aimerions obtenir de meilleurs résultats à l'avenir en réalisant un système de contrôle guidé par capteur BioMEMS.

Nous travaillerons pour être plus précis avec une interface de base de données qui fonctionne sur la recherche par la localisation des images en temps réel.

Bibliographie

[1] https ://www.aquaportail.com/definition-8834-capteurs.html

[2] KHANFAR NAIMA, à (2017) : Etude et réalisation d'une carte électronique destinée au chronométrage des durées de gel dans un milieu froid. Mémoire de fin d'étude pour master, université Larbi Ben M'hidi Oum El Bouaghi

[3] https :F. Baudoin, M. Lavabre, Capteurs : principes et utilisations, Éd. Casteilla, 2007

[4] AKROUMA Yassine, à (2013) : Etude et réalisation d'un capteur numérique de Température DS1620 à PIC16F628A. Mémoire de fin d'étude pour master, université Larbi Ben M'hidi Oum El Bouaghi

[5] Ghallab et Badawy, 2005 ; Lee et al., 2007b ; Becker et al.,1995

[6] Abdennour Abbas. Fabrication et Fonctionnalisation de BioMEMS par Plasma Froid pour l'Analyse de la Biocatalyse en Spectroscopie TeraHertz. Physique [physics]. Université de Lille1, 2010. Français. fftel-00453908f

[7] Le graphique a été réalisé par l'interrogation des bases de données SCOPUS et Science Direct, en recherchant les termes : BioMEMS OR biosensors OR lab on chip OR micro total analysis system, dans title OR abstract OR key words..

[8] Berry, G., Canavé, P., and Gauthier, G. (1987). Programmation synchrone des systèmes réactifs : le langage esterel. Technique et Science Informatique, 6 :305–316.

[9] Harel, D. and Pnueli, A. (1985). On the development of reactive systems. Logics and Models of Concurrent Systems, 13 :477–498.

[10] CNRS (1988). Le temps réel. TSI, 7 :493–500.

[11] Stankovic, J. A. (1988). Misconceptions about real time computing : A serious problem for next-generation systems. Computer, 21 :10–19.

[12] Zoubir Mammeri, CONCEPTION DE SYSTEMES TEMPS REEL, Université Paul Sabatier, TOULOUSE, 2004-2008, P 91.

[13] Kocik, R. (2000). Sur l'optimisation des systèmes distribués temps réel embarqués : application au prototype rapide d'un véhicule électrique semi-autonome. PhD thesis, Université de Rouen

[14] Frédéric Fauberteau. Sûreté temporelle pour les systèmes temps réel multiprocesseurs. Autre [cs.OH]. Université Paris-Est, 2011. Français.

[15] Laurent GEORGE, Nicolas RIVIERRE et Marco SPURI : Preemptive and non-preemptive real-time uniprocessor scheduling. Rapport technique RR-2966, INRIA, Rocquencourt, France, September 1996.

[16] Processus | Apprenez à connaître la définition (heflo.com)

[17] Processus informatique : définition de Processus informatique et synonymes de Processus informatique (français) (leparisien.fr)

[18] Demarco, T. (1979). Structured Analysis and System Speci. Englewood Cliffs.

[19] Le Goc Marc, cour Spécification et conception temps réel Principes de base de SA-RT, Projet Sachem I. U. S. P. I. M, 1996.

[20] Adime Hakime, Conception d'un noyau de calcul pour la contrôle-commande temps réel de la machine-tour,Mémoire de fin d'études en vue de l'obtention du diplôme de Master(LMD),FACULTÉ DES MATHÉMATIQUE ET D'INFORMATIQUE DÉPARTEMENT D'INFORMATIQUE, 2019/2020, p33.

[21] Hugo Descoubes, SYSTEMES TEMPS REEL, 2015-2016, P 5.

[22] ASPENCORE 2019 Embedded Markets Study Integrating IoT and Advanced Technology Designs,Application Development and Processing Environments March 2019, www.eetimes.com.

[23] https ://docplayer.fr/154180273-Noyau-freertos-manuel-du-developpeur.html

[24] SOUICI AMEL et BOUDERBALA MAHDJOUBA, ETUDE ANATOMO-PATHOLOGIQUE DE LA TUBERCULOSE BOVINE AU NIVEAU DE L'ABATTOIR DE RELIZANE, UNIVERSITE ibn khaldoun DE TIARET institut DES SCIENCES VETERINAIRES DEPARTEMENT DE Sante animale,2013-2012.