Université Abbes LAGHROUR-Khenchela
Faculté des Sciences et Technologies
Département de Mathématique et informatique

Mémoire de fin d'études pour

L'obtention du diplome Master

Option : Génie Logiciel et Systèmes Distribuès

Vérification comportementale des systèmes de l'internet des objets par les automates de Büchi

Présenté par :
Amal Salmi

Proposé et dirigé par :
DR.Messaoudi Nabil

Les membres de jury :

Président Dr. Maarouk Toufik
Examinateur M. Benothmane Mohammed
Rapporteur Dr. Messaoudi Nabil

Année universitaire :2022-2023.

Remerciements

Tout d'abord, remercions DIEU tout-puissant de nous avoir donné la patience, la santé et la volonté d'accomplir ce modeste labeur. Je tiens à remercier mon encadreur de recherche, Monsieur Messaoudi Nabil, pour sa guidance, et ses soutiens. Ses connaissances et son expertise m'ont été d'une grande aide pour mener à bien cette recherche. Mes remerciements vont à mes très chers parents de m'avoir toujours donné le courage et d'être l'une des raisons qui me poussent à me battre face à chaque obstacle. Je tiens également à exprimer ma gratitude envers ma famille et mes amis très chers. Je voudrais particulièrement remercier ma tante et son mari, Monsieur Hamidi Abdel Salam.

Table des matières

Liste des figures

Résumé

Le système IoT (Internet des objets) est complexe en raison de la diversité des appareils connectés, des protocoles de communication utilisés et des interactions entre les composants. Cette complexité peut entraîner des problèmes tels qu'un fonctionnement incorrect ou des erreurs. L'utilisation de la spécification UML combinée à la vérification formelle des systèmes IoT permet de garantir la sécurité et la fiabilité en identifiant et en éliminant les erreurs de spécification et de conception. Les automates de Büchi sont utilisés pour spécifier les propriétés de sécurité ou la fiabilité de vivacité des systèmes.

Abstract

The IoT (Internet of Things) system is indeed complex due to the diversity of connected devices, communication protocols used, and interactions between components. This complexity can lead to issues such as incorrect operation or errors. The use of UML specification combined with formal verification of IoT systems helps ensure security and reliability by identifying and eliminating specification and design errors. Büchi automata are used to specify safety properties or the liveness reliability of the system.

introduction général

À travers ce thème nous essaierons d'utiliser les automates de Büchi dans une spécification UML pour une vérification Comportementale des systèmes de l'internet des objets. Dans ce cas, le résultat permet aux automates de Büchi de propriétés de sécurité et de vivacité (Safety and Liveliness), et, si nécessaire, de reconcevoir la spécification. L'utilisation intégrée combinée de la vérification automatique et fonctionnelle d'une spécification d'un système à base d'UML et les modèles projetés des automates de Büchi permet de réduire le temps de conception des systèmes de l'Internet des objets en éliminant en temps opportun les erreurs de spécification et de conception.

La vérification des systèmes de l'Internet des objets (IoT) joue un rôle Important dans leur développement et leur déploiement. L'IoT implique la connectivité de divers objets intelligents, tels que les appareils domestiques, les capteurs environnementaux et les dispositifs médicaux, afin de permettre des fonctionnalités avancées et une automatisation accrue. Cependant, en raison de la complexité croissante des systèmes IoT, il devient essentiel de s'assurer que ces systèmes fonctionnent correctement et en toute sécurité.

La spécification UML (Unified Modeling Language) est un langage de modélisation largement utilisé pour la conception et la documentation de systèmes logiciels. Il fournit un ensemble de notations graphiques permettant de représenter les différents aspects d'un système, y compris sa structure, son comportement et ses interactions. L'UML offre également la possibilité de spécifier le comportement attendu d'un système à l'aide de diagrammes de séquence et d'activités.

La combinaison de la vérification de l'IoT et de l'UML offre un moyen puissant d'assurer la qualité et la sécurité des systèmes IoT. La vérification de l'IoT consiste à analyser et à vérifier les propriétés clés des systèmes IoT, telles que la sécurité, la fiabilité et la conformité aux exigences fonctionnelles. L'UML facilite la spécification et la représentation des comportements attendus du système, permettant ainsi la modélisation formelle et la vérification de ces comportements.

Prenons l'exemple d'une maison intelligente (smart home). Une maison intelligente est un système IoT qui intègre des dispositifs et des capteurs pour automatiser et faciliter la gestion de divers aspects de la maison, tels que le chauffage, le climatiseur, le système et le capteur qui travaillent en fonction de la température.

Ensuite, en utilisant des techniques de vérification formelle, telles que les automates de Büchi, nous pouvons vérifier si le comportement de la maison intelligente satisfait certaines propriétés de fiabilité.

À la fin, la vérification de l'IoT en utilisant l'UML offre un moyen puissant de garantir la qualité et la sécurité des systèmes IoT, tels que les maisons intelligentes. En spécifiant formellement le comportement attendu du système à l'aide de l'UML et en utilisant des techniques de vérification formelle, nous pouvons détecter et éliminer les erreurs de spécification, assurant ainsi le bon fonctionnement du système dans différentes situations et scénarios.

ce mémoir est un organisé suite :

— le premier chapitre presente l'intrnet des objets.

— le deuxiéme chapitre présente uml iot .

— Le troisième chapitre présente la vérification formelle

— Le troisième chapitre présente la vérification de la fiabilité de systéme iot.

Chapitre 1

Internet des objets

1.1 Introduction

L'Internet des Objets (IoT) est devenu un sujet extrêmement populaire de nos jours. Il fait référence à la connexion des objets qui nous entourent sans intervention humaine, permettant ainsi à ces objets d'être connectés à Internet. L'IoT a pénétré de nombreux domaines, tels que la santé, l'industrie, l'agriculture, les maisons intelligentes et même les achats, comme c'est le cas avec Amazon. Il a transformé et simplifié notre façon de vivre et de travailler, offrant des avantages significatifs. Dans le domaine de la sécurité résidentielle, par exemple, l'IoT a permis des améliorations considérables. Les systèmes de sécurité connectés peuvent surveiller les maisons à distance, détecter les intrusions et envoyer des alertes en temps réel aux propriétaires. Cette technologie a apporté une tranquillité d'esprit aux personnes, renforçant ainsi leur sentiment de sécurité. De plus, l'IoT joue un rôle essentiel dans l'amélioration de la qualité de vie des individus grâce à un suivi de leur santé à distance. Les dispositifs portables, tels que les montres connectées et les capteurs médicaux, permettent de recueillir des données précieuses sur la condition physique des individus. Ces données peuvent être transmises aux professionnels de la santé, ce qui facilite le suivi médical à distance et permet une intervention précoce en cas de besoin. Cependant, malgré les avantages évidents de l'IoT, il reste des défis à relever, tels que les problèmes de sécurité et de protection des données.

1.2 Définition de l'internet des objets :

L'Internet des Objets (IoT) est une technologie qui permet aux objets physiques d'être connectés et de communiquer entre eux via Internet. L'IoT repose sur des dispositifs embarqués dans les objets physiques, leur permettant de collecter, d'envoyer et de recevoir des données. Les objets connectés collectent des données qui peuvent être analysées pour prendre des décisions, automatiser des actions ou fournir des informations utiles[1].

Figure 1.1 – internet des objets[2]

1.3 Historique de l'Internet des Objets :

Le terme "Internet of Things" (IoT) ou "Internet des Objets" en français, ce terme est un fruit de la convergence multiples technologie notamment l'internet ,les communication sans files , les systèmes microélectroniques , la nanotechnologie et les systèmes embarqués, les événements marquants sur le chemin de la concrétisation de l'IoT ce sont L'apparition du premier appareil connecté à Internet en 1982 à l'Université Carnegie Mellon, L'introduction de l'informatique omniprésente par Mark Weiser en 1991, qui a présenté une vision précoce de l'IoT dans son article intitulé "L'ordinateur du 21ème siècle», Le développement du WearCam par Steve Mann en 1994, l'une des premières caméras à être connectée à Internet, La popularité croissante de l'informatique ubiquitaire à partir de 1998, permettant l'intégration efficace de l'informatique dans la vie quotidienne, La première utilisation du terme "Internet des objets" par Kevin Ashton en 1999, , un chercheur britannique. Kevin Ashton et son équipe ont proposé l'utilisation des étiquettes RFID (Radio Frequency IDentification) pour permettre la connectivité de tous les objets, L'introduction du premier réfrigérateur intelligent connecté à Internet par la société LG en 2000, marquant l'entrée des objets connectés dans les foyers,Le déploiement massif de la technologie RFID (Radio Frequency Identification) à partir de 2003 et 2004, une technologie clé pour l'IoT permettant l'identification à distance des objets grâce à des étiquettes RFID et L'initiative de l'IPSo Alliance en 2008, visant à promouvoir l'utilisation du protocole IP pour les réseaux d'objets miniatures intelligents[3].
Au fil du temps Au fil du temps, le concept de l'iot a gagné en popularité, L'idée de l'IoT n'est pas nouvelle, mais elle a commencé à se répandre et à susciter un intérêt important vers 2007[4].

1.4 Evolution de l'IOT :

Les premiers objets connectés sont apparus dans les années 1990, mais c'est en 2000 que le fabricant LG a sérieusement envisagé la connexion d'appareils électroménagers à Internet. Au fil des années, les expérimentations ont continué et l'idée de l'Internet des Objets est apparue en 2009, encouragée par l'apparition des smartphones. En 2010, il y avait déjà 12,5 milliards d'appareils connectés à Internet, et les experts estimaient qu'il y aurait 50 milliards d'appareils connectés d'ici 2020. Cependant, ces estimations ne prennent pas en compte l'évolution rapide d'Internet et des avancées technologiques. Le domaine de l'Internet des objets (IoT) est en constante évolution, permettant la connectivité potentielle de tout ce qui existe, des appareils électroménagers aux arbres, en passant par les animaux, les lampes et les chaussures. Cela nécessite l'équipement des objets avec une variété de capteurs, de processeurs et de modules de communication, leur permettant de collecter et de transmettre des données, ainsi que d'interagir avec leur environnement. Avec les avancées rapides de la technologie et de l'Internet, les opportunités de connectivité et les futures avancées de l'Internet des objets continuent de croître[5].

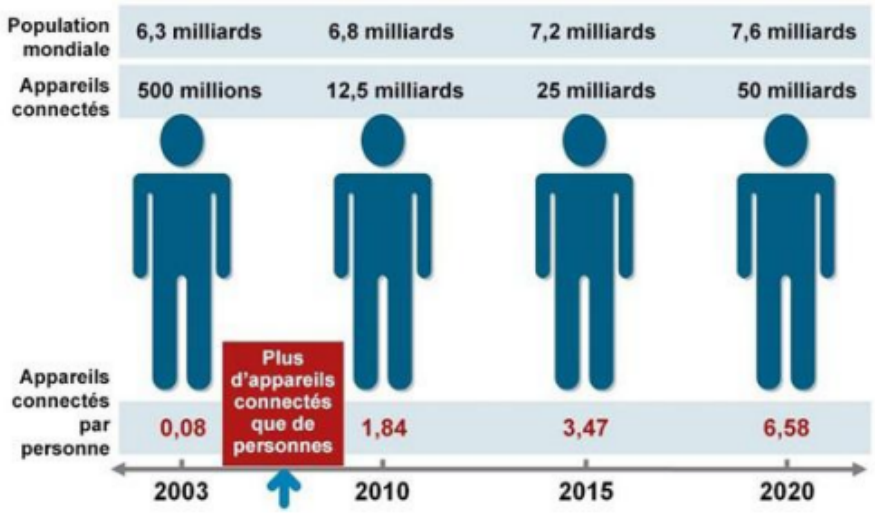


Figure 1.2 – L'évolution d'Ido entre 2003 et 2020[6].

1.5 les composants de iot :

L'Internet des objets (IoT) est composé de plusieurs éléments clés qui travaillent ensemble pour permettre la connexion et la communication entre les appareils intelligents et les objets. Tels que cette éléments les Capteurs, Les actionneurs, Prestations de service, Les plateformes et Réseaux[7].

Voici les principales composantes de l'IoT :

Capteurs : Les capteurs sont des composants essentiels des objets connectés, qui permettent de collecter des données sur leur environnement et de les transmettre à d'autres dispositifs ou systèmes. Voici quelques exemples de capteurs couramment utilisés :

1. Capteurs de température et thermostats : Ils mesurent la température ambiante ou la température d'un objet spécifique. Les thermostats sont des capteurs de température utilisés pour réguler le chauffage ou la climatisation en fonction des valeurs mesurées.
2. Capteurs de pression : Ils mesurent la pression atmosphérique, la pression exercée par un liquide ou un gaz, ou encore la pression sanguine dans le cas de capteurs médicaux.
3. Capteurs d'humidité / niveau d'humidité : Ils mesurent le taux d'humidité relative dans l'air ou la teneur en eau d'un matériau ou d'un milieu.
4. Détecteurs d'intensité lumineuse : Ils mesurent la quantité de lumière présente dans un environnement. Ils peuvent être utilisés pour ajuster l'éclairage en fonction de la luminosité ambiante.
5. Détection de proximité : Ces capteurs détectent la présence ou la proximité d'objets ou de personnes. Ils peuvent être utilisés pour activer des dispositifs lorsque quelqu'un s'approche, comme les portes automatiques ou les interrupteurs de lumière.
6. Étiquettes RFID : Les étiquettes RFID (Radio-Frequency Identification) sont de petites balises électroniques qui contiennent des informations stockées et peuvent être lues à distance à l'aide d'un lecteur RFID. Elles sont couramment utilisées pour le suivi des stocks, le contrôle d'accès, la gestion des actifs, etc.

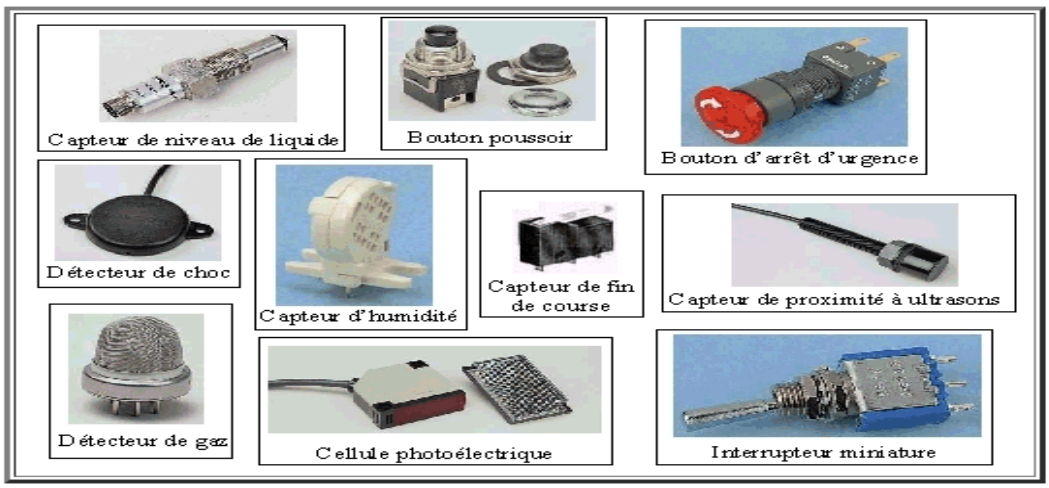


Figure 1.3 – Exemples des capteurs[8].

Les actionneurs : Les actionneurs sont effectivement des composants ou des dispositifs qui convertissent des données numériques en actions physiques. Ils sont responsables de la transformation des signaux de contrôle en mouvement, émission sonore, affichage visuel, etc. Les actionneurs jouent un rôle essentiel dans de nombreux domaines, tels que les systèmes automatisés, l'Internet des objets (IoT) et les systèmes de contrôle.

Voici quelques exemples courants d'actionneurs :

1. Afficheurs : ils affichent des informations visuelles telles que des chiffres, des textes ou des images.
2. Alarmes : elles émettent des signaux sonores ou lumineux pour alerter d'une situation spécifique.
3. Caméras : elles captent des images ou des vidéos.
4. Haut-parleurs : ils produisent des sons ou de la musique.
5. Interrupteurs : ils activent ou désactivent un circuit électrique.
6. Lampes : elles émettent de la lumière.
7. Moteurs : ils convertissent de l'énergie en mouvement.
8. Pompes : elles sont utilisées pour déplacer des fluides.
9. Serrures : elles verrouillent ou déverrouillent des portes ou des mécanismes.
10. Vannes : elles contrôlent le débit ou la direction des fluides.
11. Ventilateurs : ils génèrent un flux d'air.

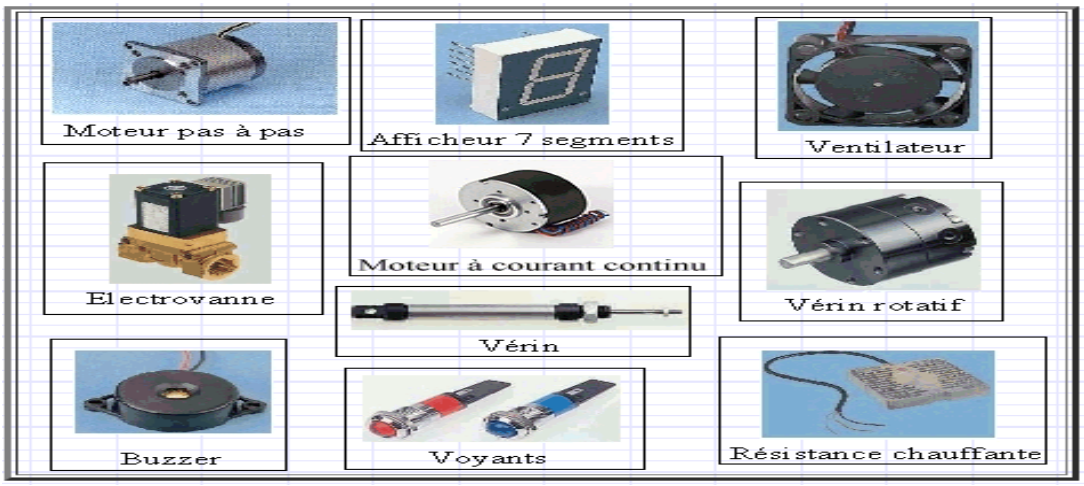


Figure 1.4 – Exemples des actionneurs [9]

Les plateformes : IoT sont considérées comme des intergiciels utilisés pour connecter les différents composants de l'IoT tels que les objets, les personnes et les services à l'environnement IoT. Elles fournissent de nombreuses fonctionnalités telles que l'accès aux appareils, la garantie d'une installation/comportement correct des appareils, l'analyse des données et une connectivité interopérable avec le réseau local, le cloud ou d'autres périphériques.

Prestations de service : Les services cloud offrent diverses prestations de service pour les applications IoT. Par exemple, ils permettent de traiter les Big Data générées par les appareils IoT et de les transformer en informations précieuses. Les services cloud offrent également la possibilité de construire et d'exécuter des applications innovantes qui exploitent les données collectées. De plus, ils facilitent l'optimisation des processus métier en intégrant les données provenant des appareils connectés, ce qui permet de prendre des décisions éclairées et d'améliorer l'efficacité opérationnelle. Les services cloud sont donc des outils essentiels pour exploiter pleinement le potentiel de l'IoT et tirer parti des avantages offerts par les données et les appareils connectés.

Réseaux : Les composants de l' IoT sont interconnectés par le biais de réseaux utilisant différentes technologies, normes et protocoles, à la fois sans fil et filaires. Voici quelques-uns des principaux types de réseaux utilisés dans les déploiements IoT :

Réseaux sans fil :

• Wi-Fi
• Bluetooth
• Zigbee
• LoRaWAN
• SigFox

Réseaux filaires :

Ethernet : Ethernet est le protocole de réseau filaire le plus couramment utilisé. Il offre une connectivité haut débit et est utilisé dans de nombreuses applications IoT, en particulier dans les environnements industriels et commerciaux.

1.6 Architecture de l'iot :

Internet des objets base sur trois couches : couche perception [10] , couche réseau et couche Application[11] .

1.6.1 Couche perception (perception layer) :

la couche de perception ou la couche physique joue un rôle essentiel dans l'architecture de l'IoT (Internet of Things) et constitue la première étape du processus. Cette couche est composée de capteurs tels que des capteurs (capteurs d'humidité, Gas, son et température.....) Et les Actionneurs (son, mouvement, lumières.....) et les caméras. Elle est responsable de capture et collecter les données, Les capteurs convertissent ces mesures physiques en signaux électriques

analogiques, qui sont ensuite numérisés et traités par des microcontrôleurs intégrés aux dispositifs de cette couche.

1.6.2 Couche réseau (network layer) :

La couche réseau : est une couche de l'architecteur l'iot après la couche perception, cette couche est responsable de la gestion de la transmission des données entre les différents objets connectés et les réseaux, ainsi que du routage des données et de la gestion des adresses IP. Cette couche est chargée de garantir la connectivité entre les objets connectés et les réseaux, et assure une communication fiable et sécurisée.

1.6.3 Couche Application (application layer) :

la couche application de l'architecture IoT joue un rôle central dans l'Internet des objets et définit les profils des services intelligents et gère les données de différents types provenant de différentes sources, notamment les objets connectés. Il est également possible d'ajouter une couche middleware entre la couche application et les deux autres couches, qui sert d'interface entre la couche matérielle et les applications. Cette couche permet la gestion des dispositifs, l'agrégation, l'analyse et le filtrage des données, le contrôle d'accès aux services, et permet également de dissimuler la complexité des mécanismes de fonctionnement du réseau pour faciliter le développement des applications. Ces trois couches travaillent ensemble pour permettre le fonctionnement des systèmes IoT. La couche perception collecte les données, la couche réseau assure la connectivité et la transmission des données, et la couche application utilise les données pour créer de la valeur ajoutée.

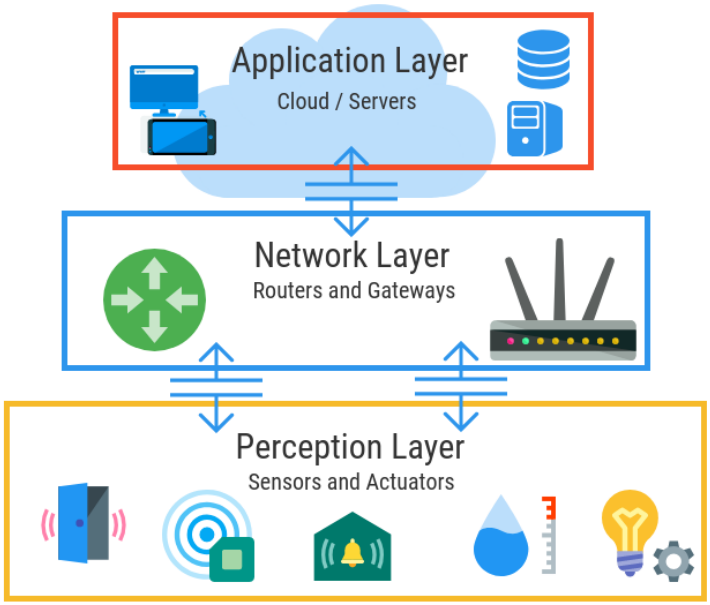


Figure 1.5 – Architecture de iot[12].

1.7 Technologies de l'iot :

L'IoT (Internet of Things) permet la connexion et l'intercommunication des différents objets intelligents via Internet. Pour que cela fonctionne, plusieurs systèmes technologiques sont nécessaires. L'IoT englobe différentes solutions techniques telles que RFID, TCP/IP, technologies mobiles, etc., qui permettent d'identifier les objets, de capturer, stocker, traiter et transférer des données à la fois dans des environnements physiques et entre des contextes physiques et des univers virtuels cette technologie Divisé en sections notamment[13] :

1. **Les technologies de courte portée :**
   - **NFC (Near Field Communication) :** Cette technologie permet des communications sans fil à très courte portée (quelques centimètres) entre des dispositifs compatibles. Elle est utilisée pour des applications telles que le paiement sans contact, le couplage rapide de dispositifs et le partage d'informations entre appareils.
   - **Bluetooth :** Une technologie sans fil populaire utilisée pour la communication à courte portée (jusqu'à quelques dizaines de mètres). Elle est couramment utilisée pour la connexion d'appareils tels que les smartphones, les écouteurs sans fil, les enceintes et les dispositifs domestiques intelligents.
   - **Zigbee :** Une technologie de communication sans fil à faible consommation d'énergie utilisée pour les réseaux de capteurs et de contrôle. Zigbee est souvent utilisé dans les applications domotiques et industrielles pour le contrôle à distance et l'automatisation des appareils.
2. Les technologies de moyenne portée :
   - **Wi-Fi :** Une technologie de réseau sans fil largement utilisée permettant des communications à courte à moyenne portée (quelques dizaines à quelques centaines de mètres). Le Wi-Fi est utilisé pour la connectivité Internet haut débit dans les environnements domestiques, les bureaux, les lieux publics, etc.
   - **Bluetooth Low Energy (BLE) :** Une version économe en énergie de la technologie Bluetooth, conçue pour les appareils IoT. BLE est utilisé pour les applications nécessitant une faible consommation d'énergie, comme les dispositifs de suivi, les wearables (appareils portables) et les capteurs.
3. **Les technologies de longue portée :**
   - **Réseaux cellulaires mobiles :** Les réseaux de téléphonie mobile (comme la 2G, la 3G, la 4G et la 5G) permettent une large couverture géographique pour la connectivité IoT. Ces réseaux offrent des vitesses de transmission de données élevées, mais peuvent nécessiter une consommation d'énergie plus élevée pour les appareils IoT.
   - **Réseaux radio bas-débit (LoRaWAN) :** LoRaWAN est une technologie de communication à longue portée conçue spécifiquement pour l'IoT. Elle utilise des fréquences radio libres de droits pour permettre une communication à longue portée

(plusieurs kilomètres) avec une faible consommation d'énergie. LoRaWAN est souvent utilisé pour les applications de suivi, de surveillance et d'agriculture intelligente. Ces différentes technologies de l'IoT offrent des options variées en termes de portée, de débit de données, de consommation d'énergie et de coût, ce qui permet de les adapter aux besoins spécifiques des différentes applications de l'IoT.

1.8 Les domaines d'internet des objets

Les applications de l'IoT (Internet des objets) sont extrêmement diversifiées et offrent de nombreuses opportunités dans divers secteurs. Voici quelques exemples d'applications dans certains domaines spécifiques,tel que l'agriculture,La maison intelligence, santé, le transport et la logistique :

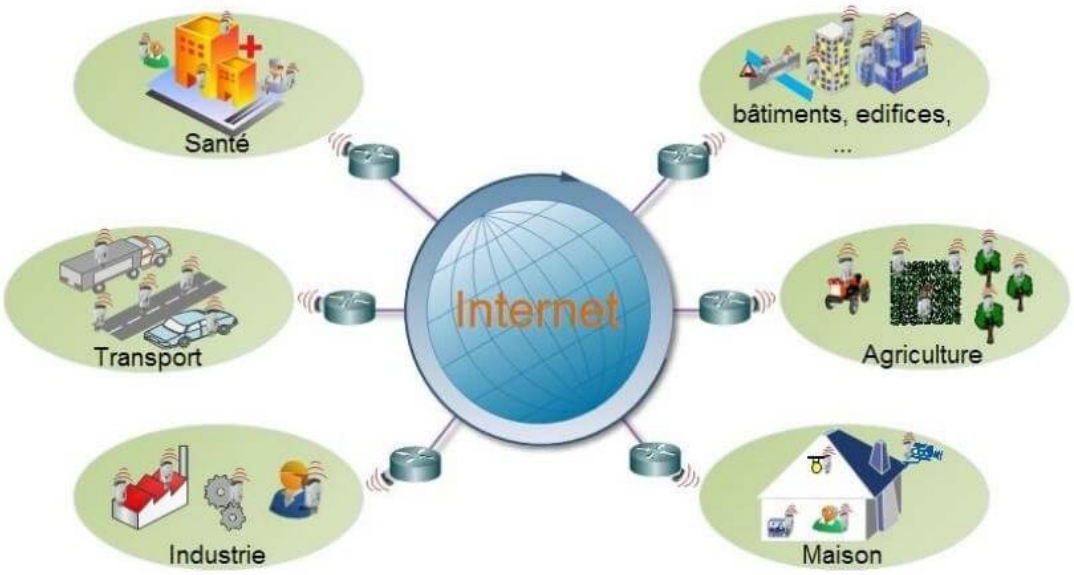


Figure 1.6 – les domaines de l'internet des objets[14].

1.8.1 la Maison intelligente :

Les maisons intelligentes, également appelées maisons connectées, utilisent des dispositifs IoT tels que des capteurs et des actionneurs pour améliorer la vie des occupants et offrir un confort accru, en particulier aux personnes âgées ou handicapées. Les capteurs IoT mentionnés, tels que les capteurs de température, de luminosité, d'humidité, de mouvement, de qualité de l'air, de détection d'eau et de fenêtre/porte, sont utilisés pour collecter des données sur l'environnement de la maison. Ces données sont ensuite utilisées pour prendre des décisions et contrôler les actionneurs appropriés. Les actionneurs IoT, tels que les actionneurs d'éclairage, de chauffage, de volets et stores, de serrures, d'appareils électroménagers et de système de sécurité, permettent de mettre en œuvre les décisions prises à partir des données collectées par les capteurs[15].

Figure 1.7 – L'Internet des objets et la Maison intelligente[16]

Fonctionnalités les maisons intelligentes basées sur l'IoT :

Les maisons intelligentes tirent pleinement parti de l'Internet des objets (IoT) pour créer un environnement connecté et réactif. Grâce à l'intégration de capteurs et d'actionneurs IoT, ces maisons sont capables de surveiller et d'ajuster automatiquement les conditions de l'environnement domestique pour améliorer le confort, la sécurité et l'efficacité énergétique. L'IoT est utilisé dans divers domaines des maisons intelligentes, tels que le contrôle des portes et des fenêtres, La santé [17], la sécurité de la maison et la gestion de l'énergie [18] , ainsi que la gestion des produits alimentaires[19].

1. Sécurité de la maison : L'IoT permet la mise en place de systèmes de sécurité avancés, tels que les détecteurs d'incendie connectés, les caméras de surveillance intelligentes, les alarmes et les capteurs de mouvement. Cela permet aux propriétaires de maison d'être alertés en cas d'incendie, d'intrusion ou d'autres situations d'urgence. Les capteurs dé-

tectent les anomalies ou les mouvements suspects, déclenchant des alarmes ou envoyant des notifications aux propriétaires via leurs appareils connectés.

1. Gestion de l'énergie : L'IoT peut être utilisé pour optimiser l'utilisation de l'énergie dans une maison intelligente. Par exemple, des capteurs connectés peuvent surveiller les conditions environnementales telles que la température, l'humidité et la luminosité, et ajuster automatiquement le chauffage, la climatisation et l'éclairage en fonction de ces paramètres. Cela permet d'économiser de l'énergie lorsque la maison est inoccupée ou de répondre aux préférences des occupants, contribuant ainsi à une meilleure efficacité énergétique

1. Contrôle des portes et des fenêtres : Les maisons intelligentes utilisent souvent des serrures et des systèmes de fenêtres connectés, ce qui permet aux occupants de verrouiller, déverrouiller et surveiller leurs portes et fenêtres à distance à l'aide de leur smartphone ou d'autres dispositifs connectés. Cela offre un plus grand niveau de sécurité et de commodité, permettant aux occupants de contrôler l'accès à leur domicile, même lorsqu'ils ne sont pas physiquement présents.

1. Gestion des produits alimentaires : Dans certaines maisons intelligentes, des capteurs peuvent être utilisés pour surveiller les niveaux des produits alimentaires. Ces capteurs peuvent détecter si certains produits alimentaires sont présents dans la maison et peuvent envoyer des notifications aux occupants pour les informer de la disponibilité ou de l'épuisement de ces produits. Cela peut aider les occupants à gérer leur inventaire alimentaire plus efficacement, évitant ainsi les pénuries ou les pertes alimentaires.

1. La santé : le domaine de la santé aujourd'hui offre plusieurs applications pour aider les habitants de la maison intelligent et sur tout en situation de handicap, atteintes de maladies neuro-dégénératives telles que la maladie d'Alzheimer ou encore des personnes âgées , Par exemple :

   • Dans le cas de la maladie d'Alzheimer, les capteurs de mouvement peuvent ne pas détecter de mouvement à un endroit spécifique. Cela peut être considéré comme un indicateur que le patient n'a pas bu suffisamment d'eau ou mangé suffisamment de nourriture.

   • Les appareils qui s'enroulement autre de la main comme la montre suivre l'activité physique, mesurer le taux de sucre, le nombre de calories brûlées.....

Figure 1.8 – Exemple de la santé Maison intelligentes[20]

1.9 Conclusion :

Dans ce chapitre, nous avons présenté l'Internet des objets, qui désigne les objets connectés à Internet et capables de communiquer entre eux. Nous avons également abordé son évolution, son historique, sa technologie et ses composants. Nous avons discuté de l'architecture de l'IoT ainsi que de ses domaines d'application, tels que la maison intelligente. Le chapitre suivant portera sur l'étude de la UML pour IOT

Chapitre 2

uml iot

2.1 introduction

Le monde vit à l'ère des technologies de l'information et de la communication, où différents objets et appareils sont connectés les uns aux autres via Internet. L'Internet des objets, ou IoT (Internet of Things), est considéré comme l'un des développements technologiques les plus importants de la dernière décennie. L'IoT fait référence au réseau d'appareils et d'objets qui se connectent et interagissent les uns avec les autres, qu'il s'agisse d'appareils mobiles tels que les smartphones et les tablettes, ou d'appareils connectés à l'environnement environnant tels que les voitures, la maison intelligente, les appareils médicaux et autres. Lorsqu'il s'agit de concevoir et de développer des systèmes IoT complexes, il devient essentiel de disposer d'outils facilitant le processus de description, d'analyse et de conception de ces systèmes. C'est là qu'intervient le langage de modélisation unifié (UML). UML est un langage de dessin qui a été développé pour fournir un ensemble de symboles et de concepts normalisés utilisés pour décrire les systèmes logiciels et les systèmes complexes en général. Ils fournissent des méthodes clairement énoncées pour représenter et documenter les composants du système, leurs relations et leur comportement.

2.2 Définition UML pour iot ou UML4iot :

UML pour l'IoT est une approche qui utilise le langage de modélisation UML (Unified Modeling Language) pour le développement d'un système IoT et la spécification des comportements du système.
UML pour l'IoT est un outil important dans le développement de systèmes IoT, car il permet de représenter et de comprendre les complexités et les interactions entre les objets du système, ce qui contribue à améliorer le processus de conception, d'implémentation et de documentation de ces systèmes complexes.
UML pour l'IoT étend et adapte le langage de modélisation unifiée (UML) pour représenter et comprendre les systèmes IoT complexes. Les ingénieurs créent des diagrammes UML pour clarifier les éléments, les interactions et l'intégration des objets dans un système IoT. Ces diagrammes aident à comprendre, documenter et analyser les systèmes, et à prendre des décisions de conception appropriées[21].

2.3 Définition de uml :

UML est un langage de modélisation standardisé utilisé dans le domaine du développement logiciel et des systèmes d'information. Il fournit une notation graphique pour représenter visuellement les différentes parties d'un système, telles que les classes, les objets, les relations, les interactions, les processus, etc.

UML permet de décrire les aspects structurels et comportementaux d'un système informatique de manière claire et compréhensible. Il fournit un ensemble de diagrammes, tels que les diagrammes de classes, les diagrammes de séquence, les diagrammes d'activité, les diagrammes de composants, les diagrammes de déploiement, etc., qui permettent de représenter différents aspects d'un système et les relations entre eux[22].

2.4 La modélisation d'un système IoT

La modélisation d'un système IoT est essentielle pour représenter ses différents aspects de manière structurée. Cette modélisation peut être réalisée à l'aide de divers diagrammes, qui sont des représentations visuelles des modèles UML (Unified Modeling Language). Ces diagrammes permettent de décrire le système de manière générale, en spécifiant ses fonctionnalités et ses exigences.
Les diagrammes peuvent être divisés en deux types : statique et dynamique[23].
Les diagrammes statiques représentent la structure du système et comprennent notamment le diagramme de classes,diagramme de cas d'utilisation, le diagramme d'objets, le diagramme de déploiement et le diagramme de composants. Diagramme de classes : Met l'accent sur les classes du système, leurs attributs, leurs méthodes et leurs relations. Il montre la structure statique du système.

- Diagramme d'objets : Se concentre sur les instances spécifiques des classes et les liens entre elles. Il montre l'état des objets à un moment donné dans le système.
- Diagramme de déploiement : Décrit la configuration matérielle du système IoT. Il indique les différents nœuds matériels (par exemple, les dispositifs IoT, les serveurs, les capteurs) et leurs connexions physiques.
- Diagramme de composants : Met en évidence les composants logiciels du système et leurs interactions. Il montre les dépendances entre les différents composants et comment ils sont assemblés pour former le système.
- Diagramme de classes : Il représente les classes du système, leurs attributs, leurs méthodes et leurs relations, telles que l'héritage, l'agrégation et l'association.

D'autre part, les diagrammes dynamiques représentent le comportement du système. Ils comprennent le diagramme d'états-transitions, le diagramme de séquence, le diagramme d'activité et le diagramme de collaboration.

- Diagramme d'états-transitions : Définit les différents états d'un objet dans le système IoT et les transitions entre ces états. Il montre comment l'objet réagit aux événements et comment il change d'état.
- Diagramme de séquence : Montre l'ordre d'exécution des messages entre les objets du système. Il met en évidence les interactions temporelles entre les objets et aide à comprendre le flux des opérations.

- Diagramme d'activité : Met en évidence le flux d'activités dans le système IoT. Il montre les actions, les décisions et les branchements conditionnels dans un processus donné.
- Diagramme de collaboration : Illustre les interactions entre les objets du système. Il montre comment les objets coopèrent pour atteindre un objectif commun.

2.4.1 diagramme d'activité :

les diagrammes d'activité sont l'un des types de modélisation graphique les plus couramment utilisés dans UML (Unified Modeling Language) pour représenter le flux de contrôle d'un système. Ils sont particulièrement utiles pour visualiser et communiquer les processus, en mettant l'accent sur les actions, les décisions et le contrôle entre les différentes étapes et le flux de travail d'un processus.
Les diagrammes d'activité permettent de représenter de manière visuelle et intuitive le déroulement d'un processus en montrant comment les différentes étapes s'enchaînent. Chaque étape est représentée par un nœud dans le diagramme, et les transitions entre les nœuds représentent les flux de contrôle entre les différentes étapes. Les actions à effectuer à chaque étape peuvent être spécifiées à l'aide de notations textuelles ou graphiques.
Les décisions conditionnelles peuvent également être représentées dans un diagramme d'activité à l'aide de nœuds de décision. Ces nœuds ont des sorties conditionnelles qui déterminent quel chemin sera pris en fonction d'une condition spécifiée. Cela permet de modéliser les différentes possibilités et les choix qui peuvent être faits dans le flux de contrôle.
De plus, les diagrammes d'activité peuvent inclure des boucles pour représenter des itérations ou des répétitions de certaines étapes. Cela permet de modéliser les parties du processus qui peuvent être exécutées plusieurs fois jusqu'à ce qu'une condition de sortie soit satisfaite[24].

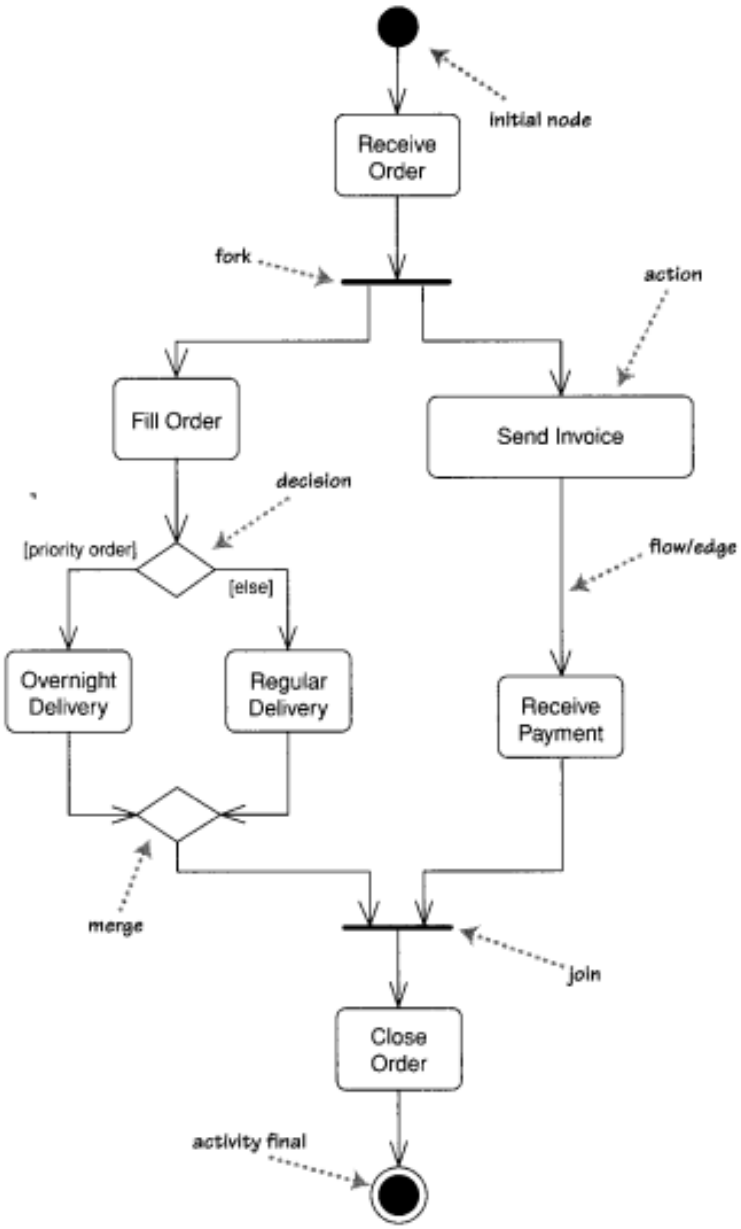


Figure 2.1 – Un schéma d'activité simple[25]

Un exemple simple de diagramme d'activité serait : Début -> Étape 1 -> Étape 2 -> Étape 3 -> Fin.

Diagramme d'activité de iot :

Le diagramme d'activité est utilisé pour représenter le flux de contrôle dans un système IoT. il peut être utilisé pour décrire le comportement des objets ou des composants du système, ainsi que les actions, les décisions et les transitions entre les états. Le diagramme d'activité dans un système IoT peut être divisé en plusieurs sections principales :

Début : indique le point de départ de la séquence des opérations dans le système.

Activités : représentent les différentes étapes qui sont exécutées dans le système. Ces activités peuvent inclure des objectifs spécifiques tels que la collecte de données, leur analyse, la prise de décisions, l'envoi de signaux aux dispositifs connectés, etc.

États : représentent les différents états dans lesquels le système peut se trouver, tels qu'un état sans connexion au réseau ou un état de fin d'une opération spécifique.

Décisions : représentent les points de prise de décision dans le système, où une option appropriée est choisie en fonction de conditions spécifiques.

Transitions : représentent le changement d'état du système d'un état à un autre, y compris les transitions directionnelles, les transitions conditionnelles, etc. Ces transitions représentent le changement de flux et de contrôle dans le système.

En utilisant le diagramme d'activité, vous pouvez représenter visuellement et comprendre le flux des opérations dans un système IoT de manière claire. Vous pouvez utiliser différents symboles pour représenter les activités, les états et les décisions, ainsi que des liens et des flèches pour relier ces éléments et illustrer le flux entre eux.

Par exemple diagramme d'activité dans un système IoT qui décrit le flux des opérations pour une interaction entre un appareil intelligent, un réseau IoT et un serveur cloud. Le diagramme d'activité peut inclure des activités telles que la connexion au réseau, la collecte de données, leur analyse, la prise de décisions basées sur les données collectées, l'envoi de signaux de contrôle aux dispositifs connectés, etc. Par exemple, un diagramme d'activité simple pourrait ressembler à ceci : • Début

• Connexion au réseau
• Collecte de données à partir des dispositifs connectés
• Analyse des données collectées
• Décision de stockage des données si elles dépassent une limite spécifique
• Envoi de signaux de contrôle aux dispositifs connectés
• Fin

2.4.2 Diagramme de séquence :

Le diagramme de séquence est l'un des types de diagrammes UML (Unified Modeling Language) et il est utilisé pour décrire l'interaction entre les objets ou les acteurs dans un système. Il

représente la séquence temporelle des messages échangés entre ces différents objets ou acteurs.

Le diagramme de séquence met l'accent sur l'ordre d'exécution des messages, montrant comment les objets collaborent entre eux pour accomplir une fonctionnalité spécifique. Il peut être utilisé pour modéliser des scénarios de système complets ou pour se concentrer sur des interactions spécifiques à l'intérieur d'un système.

Les diagrammes de séquence sont utiles pour visualiser et comprendre les instructions entre les différents composants d'un système, en montrant comment ils interagissent et communiquent les uns avec les autres dans le cadre d'un scénario donné. Ils sont couramment utilisés dans l'analyse et la conception de logiciels pour représenter le flux d'exécution des processus et les échanges de données entre les objets ou les acteurs impliqués[25].

Exemple de diagramme de séquence :

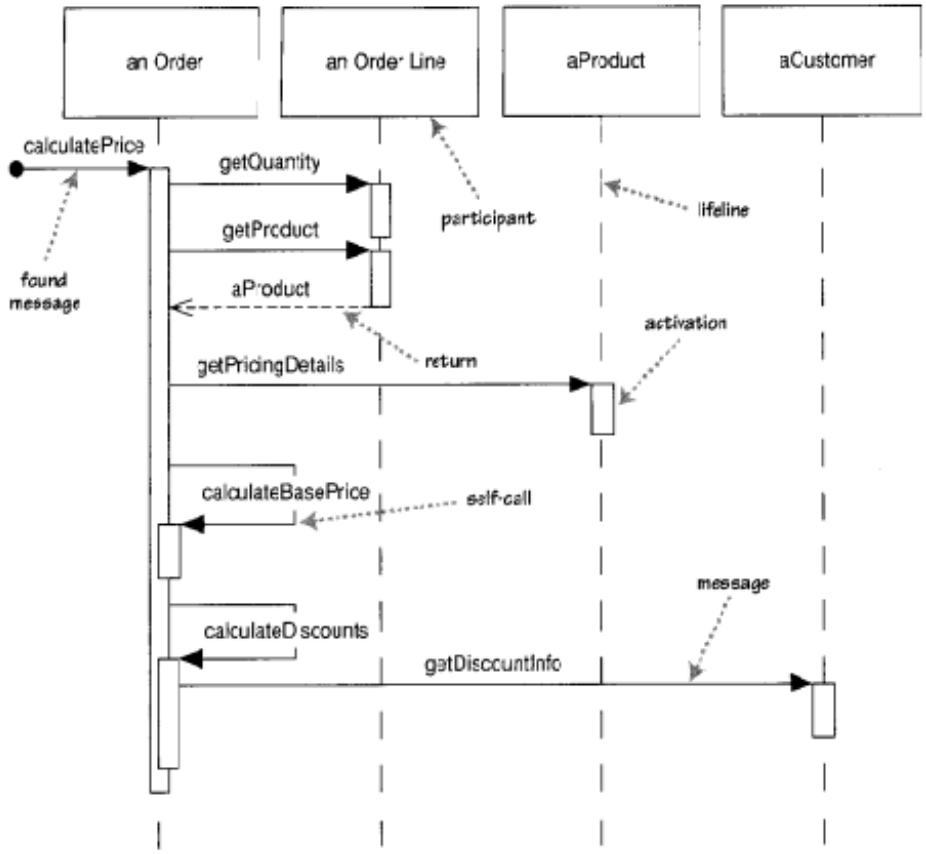


Figure 2.2 – Exemple de diagramme de séquence pour le contrôle centralisé[26]

Le diagramme de séquence IoT :

Le diagramme de séquence (Sequence Diagram) est l'un des diagrammes utilisés dans le langage de modélisation UML (Unified Modeling Language) et joue un rôle essentiel dans la conception et la représentation des interactions et des communications entre les composants des systèmes de l'Internet des objets (IoT).

Le diagramme de séquence vise à décrire la séquence des événements et des interactions entre les composants du système de manière séquentielle et chronologique. Ces interactions sont représentées sous la forme d'objets et de messages échangés entre ces objets sur une période de temps donnée.

Le rôle du diagramme de séquence dans la conception des systèmes IoT réside dans sa capacité à fournir une vision globale des interactions entre les différents composants et leur communication les uns avec les autres. Ce diagramme aide à comprendre le flux de travail et la séquence des événements dans le système, ainsi qu'à déterminer le timing nécessaire pour chaque interaction et la réaction attendue.

Grâce au diagramme de séquence, il est possible d'analyser et de concevoir plusieurs scénarios, de représenter le flux des données et les traitements temporels des événements, et de spécifier les échanges entre les appareils, les utilisateurs et les serveurs dans le système IoT.

De plus, le diagramme de séquence peut être utilisé pour déterminer les états des appareils, le temps de réponse, et les détails des communications qui se produisent entre eux. Il offre une vision globale des interactions et des communications entre les composants d'un système IoT, ce qui aide à améliorer la conception et la coordination entre les différents appareils, tout en favorisant l'intégration harmonieuse entre eux.
Pour créer un diagramme de séquence d'un système IoT, il est nécessaire de spécifier les processus connectés entre eux ou les objets impliqués dans le système, ainsi que les messages échangés entre ces objets. Les messages représentent les méthodes ou les événements échangés.
La composition du diagramme de séquence comprend plusieurs éléments clés qui sont représentés et organisés selon les conventions de modélisation UML. Voici une explication des principaux éléments du diagramme de séquence :

Les Objets (Objects) : Les objets sont les entités actives dans le système. Ils peuvent être des instances de classes, des composants matériels, des utilisateurs, etc. Chaque objet est représenté par une boîte rectangulaire avec son nom précédé par " :". Par exemple, " :Capteur", " :Contrôleur".

Les Messages (Messages) : Les messages représentent les interactions entre les objets. Ils peuvent être des appels de méthodes, des demandes de services, des envois de données, etc. Les messages sont représentés par des flèches dirigées entre les objets, avec des noms de messages au-dessus de ces flèches. Par exemple, " « demandeMesure()", "« envoiDonnees()".

Les Lifelines (Lignes de vie) : Les lignes de vie représentent l'existence temporelle d'un objet. Elles sont représentées par des lignes verticales qui se prolongent de haut en bas, juste à côté de l'objet correspondant. Une ligne de vie peut être interrompue pour représenter la destruction ou la fin de vie d'un objet.

Le Timing (Timing) : Le timing peut être représenté par des contraintes de temps sur les messages ou par des activations conditionnelles des objets. Les contraintes de temps peuvent être indiquées sous la forme de contraintes sur les messages, par exemple, ”[temps : < 5ms]”. Les activations conditionnelles peuvent être représentées par des fragments conditionnels dans le diagramme.

Les Conditions (Conditions) : Les conditions peuvent être utilisées pour représenter des décisions ou des branches dans le flux du diagramme. Elles sont généralement indiquées à l'aide de fragments conditionnels, tels que ”alt” (alternative) ou ”opt” (option), qui montrent différentes options d'exécution en fonction de certaines conditions.

La Collaboration (Collaboration) : La collaboration représente la relation entre plusieurs objets dans le système. Elle peut être représentée par des lignes pointillées reliant les objets concernés, avec un nom de collaboration au-dessus de ces lignes.

2.5 Limportonce de La modélisation dans l'iot

1. **[1.]La mdilisation permet de compréhension d'un système iot :** l'analyse et la modélisation aident à comprendre le fonctionnement du système et les interactions entre ses différents composants dans l'environnement de l'IoT. Détection des erreurs et anticipation des problèmes : L'analyse et la modélisation du système permettent d'identifier les points faibles potentiels et les dysfonctionnements possibles dans la structure et les interactions. Cela permet de détecter précocement les erreurs potentielles et de prendre des mesures pour les éviter ou réduire leur impact sur les performances du système.
2. **Conception et développement efficaces :** Les modèles UML aident à concevoir le système de manière organisée et logique. L'analyse et l'amélioration de la conception, ainsi que l'identification des composants clés et des détails d'interaction et d'intégration entre eux, permettent le développement efficace d'un système IoT évolutif et performant.
3. **Gestion de la complexité :** L'IoT implique souvent des systèmes complexes avec de nombreux composants interconnectés. La modélisation des systèmes aide à gérer cette complexité en identifiant les composants clés, leurs relations et les flux de données entre eux. Cela permet de visualiser et de comprendre la structure globale du système.
4. **Prise de décision éclairée :** La modélisation des systèmes permet d'évaluer différentes options de conception et de prendre des décisions éclairées. En créant des modèles, il est possible d'explorer et de simuler différents scénarios, d'analyser les avantages et les inconvénients de chaque approche, et de choisir la meilleure solution pour répondre aux besoins spécifiques de l'IoT.

2.6 conclusion

Ce chapitre a fait l'objet d'une étude sur uml pour l'iot. Nous avons présenté les différents concepts de uml iot et la modélisation de système iot (diagramme d'activité et diagramme de séquence de système iot) et l'importance de la modélisation dans iot.
Le chapitre suivant sera consacré à l'étude la vérification formelle.

Chapitre 3

La vérification formelle

3.1 Introduction

Dans le domaine de l'informatique et du développement de logiciels, la vérification formelle joue un rôle primordial pour garantir la qualité des applications et des systèmes. La vérification formelle est une méthode utilisée pour vérifier le bon fonctionnement des systèmes et détecter d'éventuelles erreurs ou problèmes. Les techniques de vérification formelle permettent d'analyser en profondeur les spécifications, les modèles et les programmes afin de prouver leur conformité à des propriétés spécifiques. Cela comprend la vérification de l'absence d'erreurs de logique, la conformité aux exigences fonctionnelles et de sécurité, et la détection de conditions indésirables ou de situations critiques.

3.2 La vérification :

La vérification est une technique basée sur les modèles se réfère à l'utilisation de modèles précis et non ambigus pour décrire le comportement possible d'un système, utilise plusieurs méthode, tels que le test, simulation et la vérification formelle, simulation est une technique utilise pour vérification d'un la fonctionnent correct d'un système et le test est une technique de vérification utilisée pour garantir de la qualité et du bon fonctionnement d'un système aux attentes [27].

3.3 La vérification formelle :

La vérification formelle est une techniques de vérification et basée sur des techniques formelles pour garantir l'exactitude, qui permet de garantir la correction des systèmes, qui vise à prouver de manière mathématique et logique que certaines propriétés souhaitées sont satisfaites, La vérification formelle offre des avantages significatifs permet une analyse exhaustive du système ou du logiciel. Elle peut détecter des erreurs difficiles à identifier par des tests et fournir des preuves de correction qui peuvent être vérifiées et reproduites . Les trois principales méthodes de vérification formelle sont prises en compte : model checking, l'algèbre de processus et la Preuve de théorème[28].

3.4 Model checking :

Le model checking, également appelé vérification de modèles, est une technique formelle utilisée pour vérifier si un système ou un modèle satisfait certaines propriétés spécifiées. Il s'agit d'une méthode systématique qui explore Complètement toutes les configurations possibles d'un modèle et vérifie si elles respectent les spécifications données. Le model checking généralement la création d'un modèle formel du système ou du logiciel à vérifier, ainsi que la spécification des propriétés que le système doit satisfaire. Le modèle peut être représenté sous forme de graphe, de machine d'états finis ou d'autres formalismes appropriés. La spécification des propriétés peut

inclure des contraintes temporelles, des invariants ou des exigences fonctionnelles spécifiques , Il utilise des outils spécifiques pour examiner le modèle et vérifier s'il satisfait un ensemble de propriétés spécifiées, tel que des outil de Spin Uppal[29] .

Le problème de vérification de modèle, ou model checking, peut être décrit comme suit : Étant donné une structure de Kripke M = (S, R, L) qui représente un système concurrent à états finis, et une formule logique temporelle φ exprimant une spécification souhaitée, le but est de trouver l'ensemble de tous les états dans S qui satisfont Kripke φ[30] .

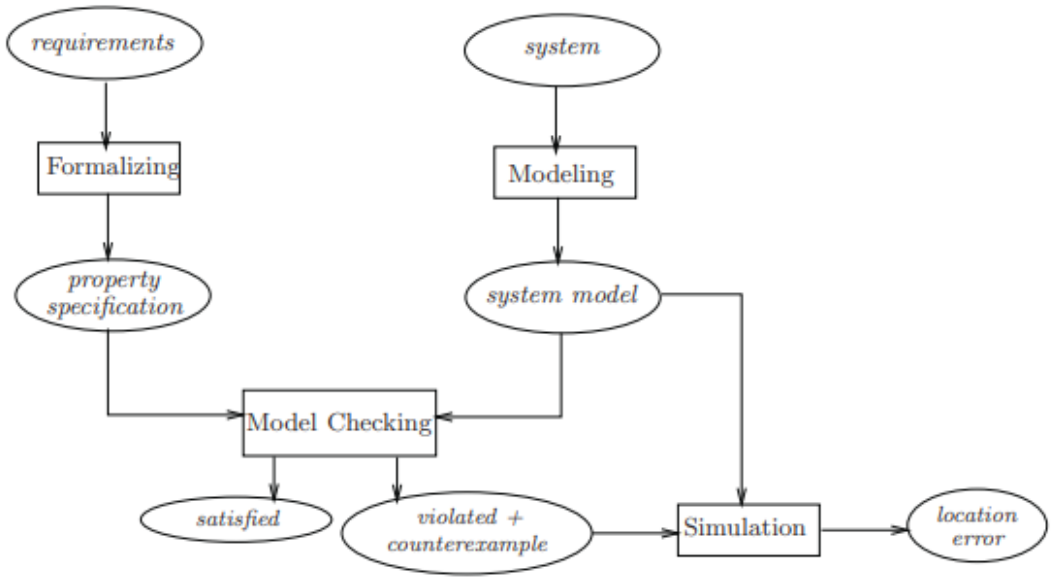


Figure 3.1 – Vue schématique de la démarche de model-checking[31] .

Le processus de model checking se déroule en plusieurs phases distinctes [32] :

1. **La modélisation :** Est une modélisation consiste à représenter le système à vérifier à l'aide du langage de description de modèle du vérificateur utilisé.
2. **La simulation :** simulations peuvent être effectuées pour valider rapidement le modèle, et la propriété à vérifier est formalisée à l'aide du langage de spécification des propriétés. Dans la phase d'exécution.
3. **Exécution :** le vérificateur de modèle est utilisé pour vérifier la validité de la propriété dans le modèle du système
4. **Analyse :** on vérifie si la propriété est satisfaite ou no. En cas de non satisfaite, on analyse le contre-exemple généré et on peut affiner le modèle, la conception ou la propriété

3.5 Logique temporelle linéaire (LTL) :

La logique temporelle linéaire (Linear Temporal Logic - LTL) est un formalisme utilisé pour spécifier et raisonner sur des propriétés temporelles dans les systèmes informatiques[33].

1. **Les variables de propositionnelles :** Les variable de propositiennelle sont des proposinelle sont des symboles utilisé pour représentes des proposition p ,q , s,… ou bien (q0, q1, q2, . . .) .
   Dans l'énoncé donné, l'ensemble AP est défini comme suit :
   AP = p, q, s, ..., q0, q1, ...

2. **Les connecteurs logiques :** Sont utilisés pour former des expressions logiques plus complexes à partir de propositions simples . Voici une description des cinq symboles de connecteurs logiques mentionnés :

   1. ¬ (négation) : Ce symbole est utilisé pour négation d'une proposition. Par exemple :¬p signifie "non p" ou "la proposition p est fausse".
   2. ∨ (disjonction) : Ce symbole est utilisé pour exprimer la disjonction ou le "ou" inclusif entre deux propositions. Par exemple, p ∨ signifie "p ou q" ou "au moins l'une des propositions p et q est vraie".
   3. ∧ (conjonction) : Ce symbole est utilisé pour exprimer la conjonction ou le "et" entre deux propositions. Par exemple, p ∧q signifie "p et q" ou "les propositions p et q sont toutes les deux vraies".
   4. → (implication) : Ce symbole est utilisé pour exprimer l'implication ou le "si... alors". Par exemple, p → q signifie "si p alors q" ou "si la proposition p est vraie, alors la proposition q doit être vraie".
   5. ↔ (équivalence) : Ce symbole est utilisé pour exprimer l'équivalence ou le "si et seulement si". Par exemple, p ↔ q signifie "p est équivalent à q" ou "les propositions p et q sont toutes les deux vraies ou toutes les deux fausses". Ces symboles de connecteurs logiques permettent de former des expressions logiques complexes et de spécifier des relations entre les propositions dans la logique propositionnelle.

3. **Les opérateurs temporels :** sont utilisés en logique temporelle pour exprimer des propriétés qui dépendent de l'ordre des événements dans le temps. Voici une description des quatre opérateurs temporels mentionnés : • X (prochain-neXt,).
   • G (toujours-Globally-always).
   • F (parfois-Future-eventually).
   • U (jusqu'à-Until).

4. **Logique temporelle linéaire (LTL) :** la LTL est une forme de spécification de la logique temporelle qui permet de décrire le comportement temporel des systèmes et de vérifier formellement si les propriétés spécifiées sont satisfaites. Les formule de logique temporelle linéaire (LTL) :
   — ∀p ∈ AP, p ∈LT L,
   — ∀φ ∈ LTL, ¬φ ∈ LTL,

— si φ1, φ2 ∈ LTL then, φ1 ∼ *P* φ2 ∈LTL ; ∼ *P*∈ ∨, ∧ , →, ↔
— si φ1, φ2 ∈ LTL then, Xφ1, Gφ1, Fφ1, φ1Uφ2 ∈ LT L

5. **Sémantique LTL sur les traces :** Les formules LTL sont évaluées sur des chemins ou des traces, qui sont des séquences infinies d'états. Une formule LTL peut être soit satisfaite, ce qui signifie qu'elle est vraie pour au moins une trace, soit insatisfaite, ce qui signifie qu'elle n'est vraie pour aucune trace. Comme défini dans la sémantique de LTL.
La sémantique de la formule LTL φ est définie comme un langage W ords(φ) qui contient tous les mots infinis sur l'alphabet 2 AP qui satisfont φ .
Définition Soit φ une formule LTL sur AP.words(φ) = σ ∈ (2AP)ω| ∈ |= φ
La sémantique est formalisée par la satisfaction de la relation |=. Soit les notations suivantes :

   - σ est un mot infini défini comme A0A1A2 · · · et il appartient à l'ensemble des mots infinis sur l'alphabet AP.
   - Un chemin est une séquence infinie d'états représentée par π = s0s1 . . . sn+1 . . .
   - σi est le suffixe de σ à partir de l'ième élément.
   - Pour tout i supérieur ou égal à 0, l'ensemble des propositions atomiques vraies dans l'état si, noté L(si), est inclus dans l'ensemble des propositions atomiques AP.
   - trace σ = A0A1A2 · · · ∈ (2AP)ω

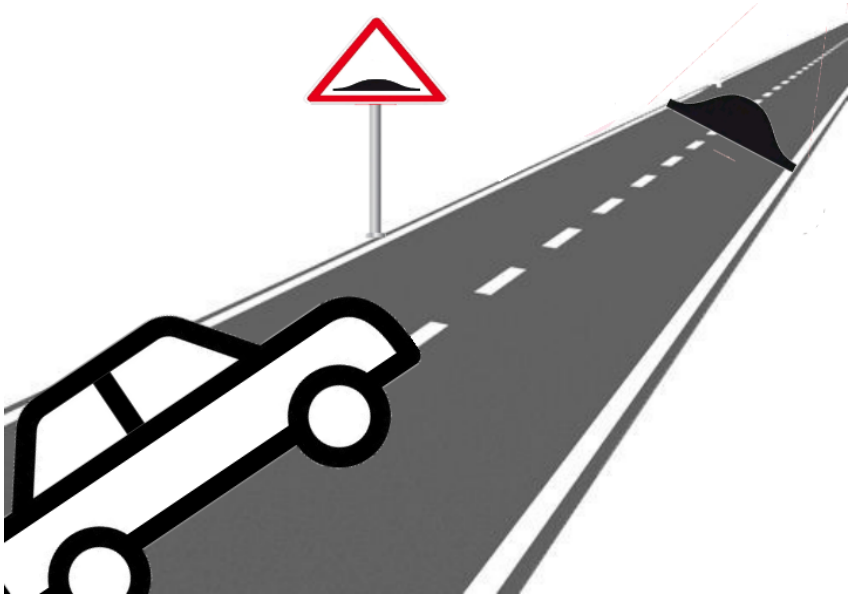
σ	$\models$	true		
σ	$\models$	p	ssi	$A_0 \models p, (p \in A_0)$
σ	$\models$	$\neg\varphi$	ssi	$\sigma \not\models \varphi$
σ	$\models$	$\varphi_1 \vee \varphi_2$	ssi	$\sigma \models \varphi_1$ ou $\sigma \models \varphi_2$
σ	$\models$	$\mathbf{X}\varphi$	ssi	$\sigma_1 \models \varphi$
σ	$\models$	$\varphi_1 \mathbf{U} \varphi_2$	ssi	$\exists j \geq 0, \sigma_j \models \varphi_2$ et $(\forall k < j, \sigma_k \models \varphi_1)$
σ	$\models$	$\mathbf{G}\varphi$	ssi	$\forall i \geq 0, \sigma_i \models \varphi$
σ	$\models$	$\mathbf{F}\varphi$	ssi	$\exists j \geq 0, \sigma_j \models \varphi$

Figure 3.2 – Sémantique LTL sur les traces[33].

LTL on utilise des opérateurs temporels pour exprimer ces propriétés. Voici les principaux opérateurs temporels utilisés en LTL :

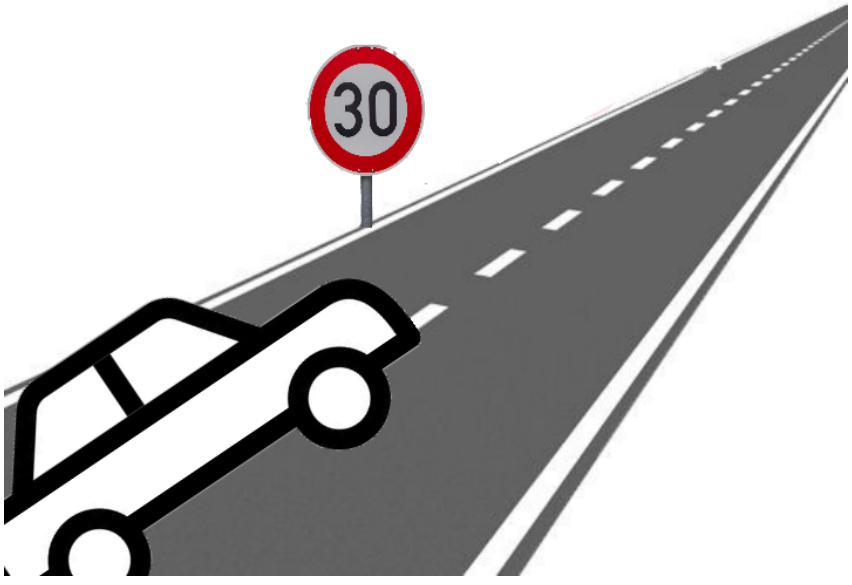
   - X (Next) : Cet opérateur exprime ce qui doit être vrai dans le prochain état ou le prochain instant de temps. Par exemple, X(p) signifie que la proposition p doit être

vraie dans l'état ou l'instant de temps suivant. Par exemple, Par exemple,que la voiture continue de rouler, puis rencontre un panneau dos d'âne, suivi d'un dos d'âne. "n next" indique un événement ou une action qui se produit immédiatement après un autre, dans ce cas, la voiture rencontre d'abord un panneau dos d'âne, puis prend un dos d'âne.



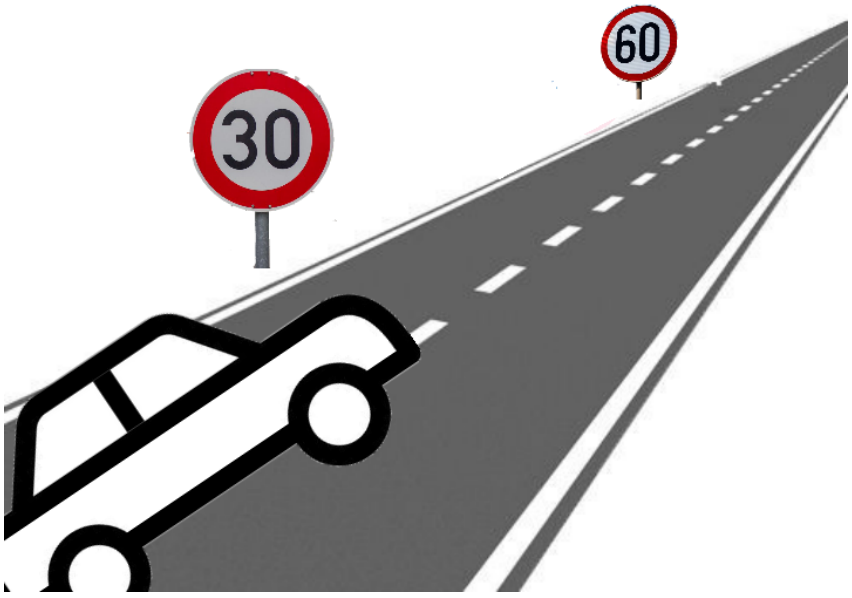
Intuition des opérateurs temporels :

- G (Globally) : Cet opérateur exprime que la propriété doit être vraie pendant toute la séquence d'états ou tout l'intervalle de temps considéré. Par exemple, G(p) signifie que la proposition p doit être vraie à chaque instant de temps dans la séquence. Par exemple, "la voiture en allant toujours à une vitesse de 30 km/h". Cela signifie que l'action de conduire à une vitesse de 30 km/h est constante ou régulière.



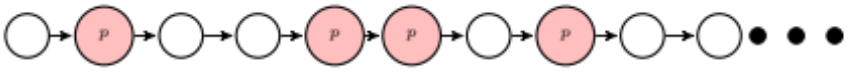
- F (Future) : Cet opérateur exprime que la propriété doit être vraie à un certain moment dans le futur, à un instant de temps ultérieur. Par exemple, F(p) signifie que la proposition p sera vraie à un certain moment dans le futur. Par exemple, ”à l’avenir, il y aura virage signalé”. Cela signifie que l’apparition de virage signalé se produira à un moment donné dans le futur.

- U (Until) : Cet opérateur exprime une relation temporelle entre deux propriétés. Il spécifie que la première propriété doit être vraie jusqu’à ce que la deuxième propriété devienne vraie. Par exemple, p U q signifie que la proposition p doit être vraie jusqu’à ce que la proposition q devienne vraie. Par exemple, ”la voiture roule à une vitesse de 30 km/h jusqu’à un certain point, puis elle revient à 60 km/h”. Cela indique la limite ou la durée pendant laquelle la voiture maintient une vitesse de 30 km/h avant de changer à 60 km/h.

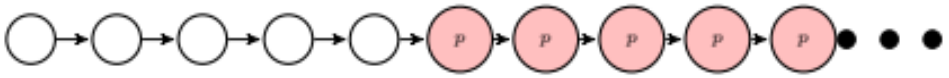


La combination entre les opérateurs

- GFp (infiniment souvent-Infinitely often p)



- FGp (Almost always p-presque toujours),



3.6 Model checking ltl :

Un algorithme de model-checking LTL est une procédure de décision qui, pour une structure de Kripke K et une formule LTL φ, renvoie ”oui” si K $\models$ φ (K satisfait φ), et ”non” (accompagné d'un contre-exemple) si K $\not\models$ φ (K ne satisfait pas φ). Le contre-exemple consiste en un préfixe fini approprié d'un chemin infini dans K où φ n'est pas vérifiée. L'algorithme de model-checking présenté dans cette approche repose sur l'utilisation d'automates de Büchi non déterministes pour représenter les formules LTL. L'objectif principal de l'algorithme est de tenter de réfuter l'affirmation selon laquelle la structure de Kripke K satisfait la formule LTL φ en recherchant un chemin π dans K tel que π satisfait la négation de φ (¬φ). Si un tel chemin est trouvé, un préfixe de ce chemin est renvoyé en tant que contre-exemple. En revanche, si aucun chemin de ce type n'est trouvé, on conclut que K satisfait φ [35].
T |= φ ssi T races(T) ⊆ words(φ) ssi
Traces(T) ∩ ((2^{AP}) $\omega \setminus$ words(φ)) = φ

ssi T races(T) ∩ words(¬φ) = φ
Soit A un NBA, avec Lω(A) = words(¬φ) on à T |= φ SSI T races(T) ∩ Lω(A) = φ

Algorithm 1 Model-checking des formules LTL

Input : Une structure de Kripke K et une formule φ.
Output : "oui" si $K \models \varphi$, sinon, "non" plus un contre-exemple.

Construire un ABN $A_{\neg\varphi}$ tel que $\mathcal{L}_\omega(\mathcal{A}) = words(\neg\varphi)$
Construire le système de transition produit $\mathcal{T} \otimes \mathcal{A}$

Vérifier la propriété de persistence $\mathcal{T} \otimes \mathcal{A} \models \mathbf{FG}\neg F$ (eventually forever).
▷ F état final de ABN
if la persistence est vérifiée **then**
  return "oui"
else if la persistence n'est pas vérifiée **then**
  return "non"
end if

Figure 3.3 – Algorithme model checking de ltl[36]

3.7 Le produit synchrone :

est une construction utilisée pour combiner un système de transition (T) et un automate de Büchi non déterministe (A) afin de vérifier des propriétés de spécification sur le système. e produit synchrone permet d'effectuer la vérification des propriétés de spécification exprimées dans le langage LTL en utilisant des techniques basées sur les automates de Büchi.
Le produit synchrone est défini comme suit [37] :
Soit T = (S, Act, →, S0, AP, L) un système de transition et A = (Q, Σ, δ, Q0, F) un automate de Büchi non déterministe.
Le produit synchrone T ⊗ A est défini comme T synchronisé avec A, et est représenté par T ⊗ A = (S × Q, Act, →', S'0, AP', L').
Les composantes du produit synchrone sont les suivantes : Les états du produit synchrone sont définis comme les paires d'états de T et A : S' = S × Q.
L'ensemble d'actions reste le même : Act. La relation de transition →' est définie par la règle suivante : Pour tout état (s, q) ∈ S' et toute action α ∈ Act, sα → s' si et seulement si s → s' dans T et q' ∈ δ(q, L(s')).
Les états initiaux du produit synchrone sont définis comme suit : S'0 = ⟨s0, q⟩ : s0 ∈ S0, q ∈ δ(Q0, L(s0)).
L'ensemble des propositions atomiques AP' est égal à l'ensemble des états de l'automate A : AP' = Q.
La fonction d'étiquetage L' associe à chaque état du produit synchrone (s, q) ∈ S' l'état q de l'automate A : L'(⟨s, q⟩) = q.

3.7.1 Automates à états finis :

les automates finis sont des modèles mathématiques qui permettent de reconnaître des langages réguliers, c'est-à-dire des ensembles de mots qui peuvent être reproduits par une expression régulière. Les automates finis sont composés d'un ensemble fini d'états, d'un alphabet fini de symboles, d'une fonction de transition qui indique comment l'automate change d'état en fonction du symbole lu, et d'un ensemble d 'états finaux qui déterminent les mots reconnus par l'automate[38].

1. Les éléments principaux d'un automate à états finis sont les suivants :
2. Ensemble fini d'états : Il s'agit d'un ensemble de tous les états possibles dans le système modélisé. Chaque état représente une configuration spécifique du système.
3. Ensemble fini de symboles en entrée : Il s'agit de l'ensemble des symboles ou des événements qui peuvent être donnés en entrée à l'automate. Ces symboles peuvent déclencher des transitions d'état.
4. Fonction de transition : La fonction de transition définit comment l'automate évolue d'un état à un autre en fonction de l'état actuel et du symbole d'entrée donné. Elle spécifie quel état sera atteint après la transition.

Les automates à états finis peuvent être représentés de différentes manières, notamment :

1. Diagrammes de transition : Les diagrammes de transition, également appelés diagrammes d'états, sont des représentations graphiques des états, des symboles d'entrée et des transitions entre les états. Chaque état est représenté par un nœud, et les transitions sont représentées par des flèches avec les symboles d'entrée associés.
2. Tables de transition : Les tables de transition fournissent une représentation tabulaire des transitions entre les états. Chaque ligne de la table correspond à un état, et les colonnes représentent les symboles d'entrée. Les cellules de la table indiquent l'état atteint après une transition spécifique.
3. Notations formelles : Il existe différentes notations formelles pour décrire les automates à états finis, telles que les expressions régulières, les grammaires formelles, les langages formels, etc. Ces notations permettent une représentation plus précise et formelle des automates.

Il y a plusieurs type de automates fini notamment les automates fini déterministes AFD, les automates fini non déterministes AFN, les automates de buchi . Un automate fini est défini par un quintuplet (Q, Σ , ô, I, F) :

— Q est un ensemble d'états.

— Σ est l'alphabet des symboles d'entrée.

— ô est la fonction de transition qui associe à chaque état et symbole d'entrée une transition vers un autre état.

— I est l'état initial.

— F est l'ensemble des états finaux.

exemples de automate fini

automate fini **Automate fini déterministe :** Un automate fini déterministe (AFD) est

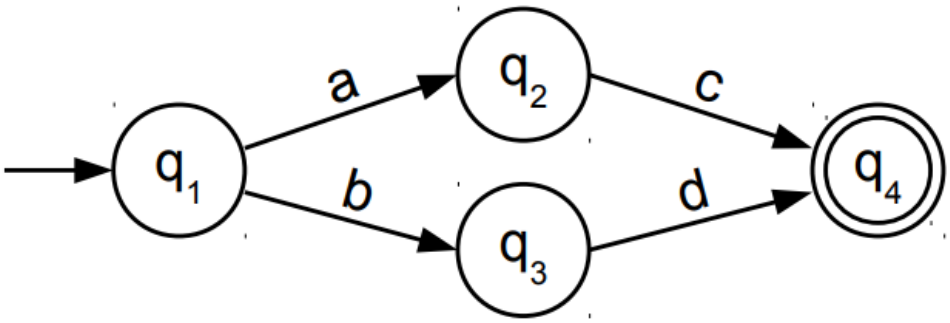


Figure 3.4 – exemple de automate fini[39]

un type d'automate fini dans lequel chaque état possède une transition définie pour chaque symbole d'entrée. Autrement dit, à partir d'un état donné et d'un symbole d'entrée donné, il n'y a qu'une seule transition possible vers un autre état. L'AFD suit donc un comportement précis et prévisible lorsqu'il traite une séquence d'entrée. Il n'y a pas d'ambiguïté dans les choix de transitions.

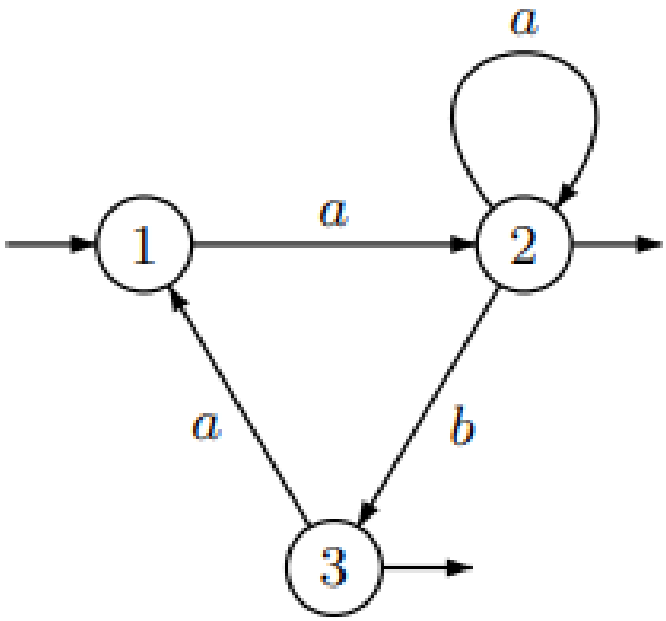


Figure 3.5 – exemple de Automates d´éterministes[40]

Automate fini non déterministe : un automate fini non déterministe (AFND) est un type d'automate fini dans lequel un état peut avoir plusieurs transitions possibles pour un même symbole d'entrée. Cela signifie qu'à partir d'un état donné et d'un symbole d'entrée donné, l'AFND peut non seulement passer à un autre état, mais également à plusieurs états simultanément. L'AFND a donc une certaine notion de non-déterminisme et peut représenter des comportements plus complexes que l'AFD.

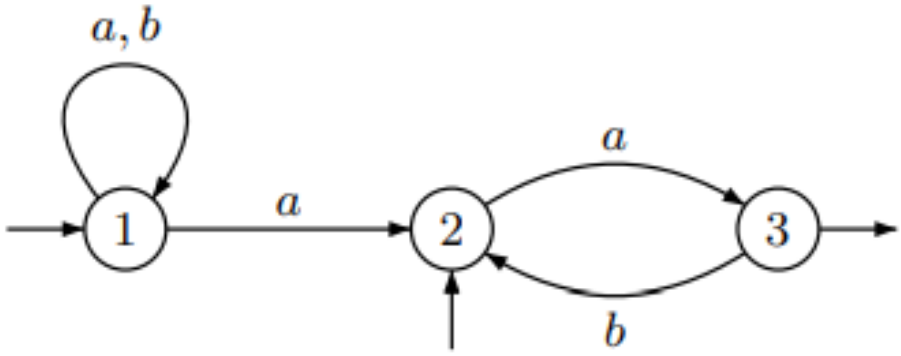


Figure 3.6 – Un automate non d´éterministe[41]

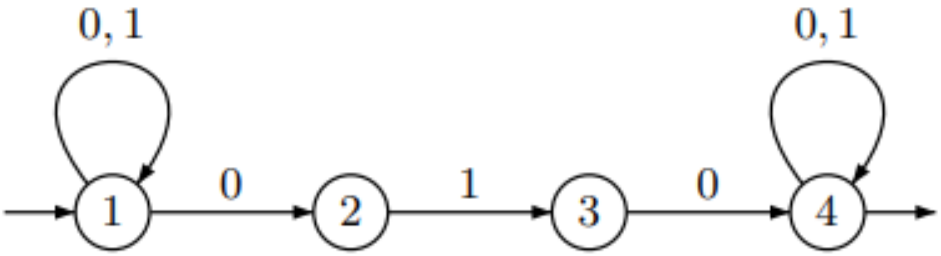
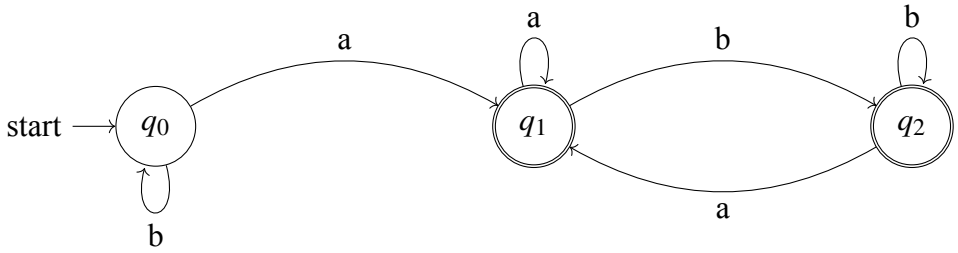


Figure 3.7 – Un automate non d´éterministe pour les mots contenant la chaine 010[42]

3.7.2 Automates de Büchi

Les automates de Büchi est un type des automates, et ce sont des automates finis, Il permette de spécifier et vérifier les formelle ou la vérification d'un système. Les automates de Büchi ressemblent à un automate sur un mots infinis, mais avec une déférence importante, les états finals sont représentés par des états double cercle plutôt que par des transitions sortantes, cette représentation met en évidence la notion d'infini, car un mot infini n'a pas de fin clairement définit[43] . Les automates de Büchi défini formellement comme A=Q,E,S,q0, F • Q= est un ensemble des états. • E= Alphabet d'entre. • q0= ensemble des états initiaux. • F= ensembles des états finals • S = ensemble de transition

exemple de l'automate de Büchi :



un exemple simplifié d'un automate de Büchi :

- **Q :** {q0, q1, q2}
- **A :** {a, b}
- **I :** {q0}
- **T :** {q1, q2}
- **E :**{(q0, a -> q1),(q0, b -> q0),(q1, a -> q1),(q1, b ->q2),(q2, a -> q1),(q2, b -> q2)}

3.8 Produit entre deux automates :

Le produit de deux automates est une construction qui permet de combiner deux automates pour en former un nouveau qui reconnaît le langage qui est l'intersection des langages reconnus par les deux automates initiaux. Formellement, si on a deux automates A1 = (Q1, Σ, δ1, q01, F1) et A2 = (Q2, Σ, δ2, q02, F2), leur produit A1 × A2 est défini comme suit :

- Q1 × Q2 est l'ensemble des états de l'automate produit
- Σ est l'alphabet commun aux deux automates
- (q01, q02) est l'état initial de l'automate produit
- (q1, q2) → (p1, p2) s'il existe une lettre $a \in \Sigma$ telle que q1 →p1 dans A1 et q2 →p2 dans A2
- F1 × F2 est l'ensemble des états finaux de l'automate produit.

exemple de produit de automates Dans cet exemple, le contenu contient le produit de deux automates P1 et P2, ainsi que P3, qui est le résultat du produit[44].

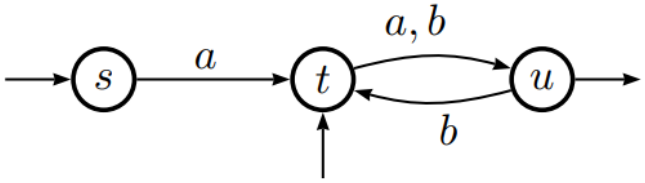


Figure 3.8 – Automates p1

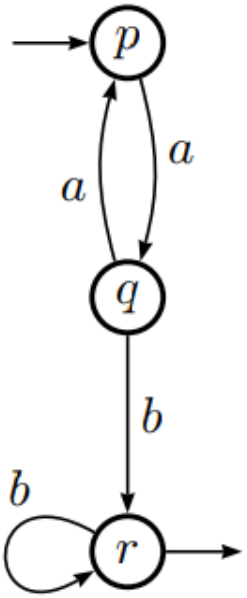


Figure 3.9 – Automates p2

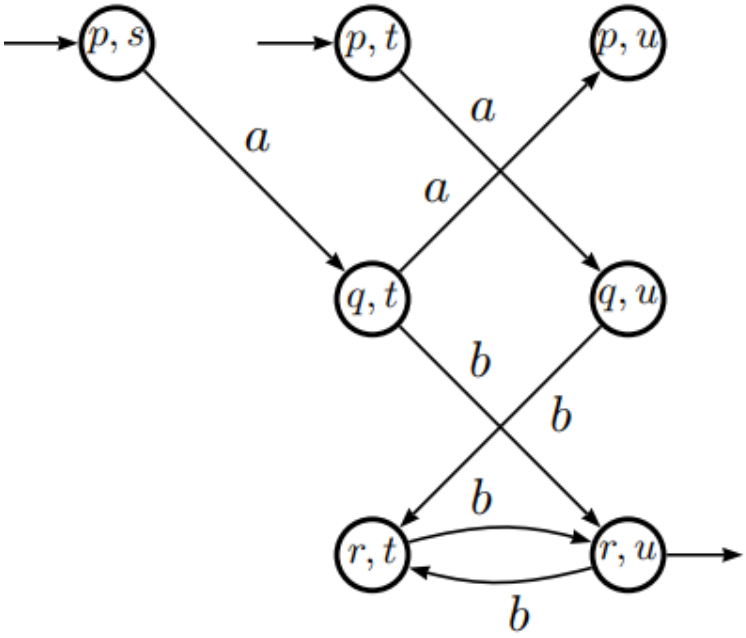


Figure 3.10 – Automates p3(le produit de deux automates d´eterministes est un automate déterministe.)

dans la Figure 3.8 la deffinition

— États : {s, t, u}

— Alphabet : {a, b}

— État initial : {s,t}

— États finaux : {u}

— Transitions :{(s, a → t), (t, a → u), (t, b → u), (u, b → u)}

dans la Figure 3.9 la deffinition

— États : {p , q, r}

— Alphabet : {a, b}

— État initial : {p}

— États finaux : {r}

— Transitions :{(p, a → q), (q, a → p), (q, b → r), (r, b → r)}

Et Figure 3.10 est un produit de p1 et p2

3.9 Spin et Promela

3.9.1 Spin

Spin (Simple Promela Interpreter) est un logiciel open source qui est utilisé comme outil de vérification pour les systèmes informatiques. SPIN peut être utilisé de deux manières principales : en mode simulation pour simuler les comportements possibles du système, et en mode vérification pour déterminer si le modèle satisfait ou non une propriété.
La stratégie de vérification de Spin repose sur la recherche de contre-exemples pour vérifier si un comportement spécifié est respecté. Il utilise un modèle et une propriété donnés pour déterminer l'espace d'états des automates correspondants. Spin accepte ou utilise le langage Promela (PROcess MEta LAnguage) pour spécifier les comportements des systèmes et utilise des techniques de model checking pour vérifier automatiquement les propriétés de comportement spécifiées dans le langage PROMELA. Le model checking consiste à explorer de manière complète tous les états possibles d'un système afin de vérifier si certaines propriétés sont satisfaites ou non[45].

3.9.2 promela

PROMELA (PROcess MEta LAnguage) est un langage de modélisation qui se concentre sur la description de systèmes logiciels concurrents. Le terme PROMELA est importante car l'abstraction est souvent essentielle pour une vérification réussie. Ce langage est conçu pour faciliter la recherche de bonnes abstractions de conceptions de systèmes, et il met l'accent sur la modélisation de la synchronisation et de la coordination des processus plutôt que sur le calcul. Les éléments de base de PROMELA incluent les processus asynchrones, les canaux de messages tamponnés et non tamponnés, les instructions de synchronisation et les données structurées. Contrairement à d'autres langages de programmation, PROMELA n'a pas de notion de temps ou d'horloge, ni de nombres à virgule flottante. Bien que cela puisse rendre difficile la modélisation de certaines fonctions de calcul complexes, cela facilite la modélisation et la vérification du comportement des clients et des serveurs dans les réseaux de processeurs. Enfin, il convient de noter que PROMELA n'est pas destiné à être utilisé comme langage d'implémentation, mais plutôt comme langage de description de système, Le langage offre également des structures de sélection avec des gardes et des boucles, permettant de contrôler le flux d'exécution. Le langage Promela présente plusieurs similitudes avec le langage C[46], Voici quelques-unes des similitudes clés entre Promela et le langage C :

- Les structeur de codution (if -else – do)
- La déclaration de variable de cinq types de données : bite, short, int, byte, Bool .
- Les processus sont des objets globaux qui représentent des entités concurrentes. Chaque processus peut être activé et exécuté indépendamment des autres processus. Le mot ”run”

est utilisé pour activer un processus à partir d'un autre processus, "init" utilisée pour l'initialisation Comme dans cette exemple qui affiche le message "Hello world !" :

```
proctype processus_declaration() {
    printf(" Hello world!\n");
}
init { run  }
```

En PROMELA (Process Meta Language), un canal est un type de données qui facilite la communication entre les processus. Un canal est associé à un type de message spécifique, ce qui signifie que seuls des messages de ce type peuvent être envoyés et reçus via ce canal.
Un canal en PROMELA offre deux opérations principales :

1. **L'opération d'envoi (send) :** Elle permet à un processus d'envoyer un message à travers le canal. Lorsqu'un processus effectue une opération d'envoi, il spécifie le canal sur lequel le message doit être envoyé et fournit le message lui-même.

2. **L'opération de réception (receive) :** Elle permet à un processus de recevoir un message à partir d'un canal spécifique. Lorsqu'un processus effectue une opération de réception, il spécifie le canal à partir duquel il souhaite recevoir un message et attend que le message soit disponible sur ce canal.

et limite de 255 canaux qui peuvent être créés dans un modèle PROMELA donné.
La déclaration suivante définit un canal nommé `nom_channel` qui peut contenir jusqu'à quatre éléments de type entier (int) :

```

chan nom_channel [4] of tint};
```

3.10 Conclusion :

tout au long du chapitre,Nous avions discuté des définitions de la vérification et la vérification formelle les automates de Büchi, Les automates à états finis, model checking , model checking ltl. Nous avions également abordé le sujet du produit de deux automates. De plus, vous avez mentionné les concepts de Spin et de Promela.

Chapitre 4

vérification la fiabilité de systéme iot

4.1 introduction :

Le système IoT (Internet of Things) est un système complexe qui intègre des appareils connectés et des réseaux pour collecter, communiquer et analyser des données. Cependant, en raison de cette complexité, il est possible que des problèmes surviennent, tels qu'un fonctionnement incorrect du système ou des erreurs de communication entre les dispositifs. Dans cette partie, nous aborderons le sujet de la vérification de la fiabilité du système IoT.

L'objectif principal de la vérification du système IoT est de s'assurer que les dispositifs et les réseaux fonctionnent de manière fiable et sécurisée, conformément aux attentes et aux spécifications définies. La vérification permet de détecter et de résoudre les problèmes potentiels, de garantir la conformité aux normes de sécurité et d'optimiser les performances globales du système.
Pour mener à bien notre recherche sur la vérification du système IoT, nous avons choisi d'utiliser un exemple concret : le modèle de maison intelligente, également connu sous le nom de smart home.

4.2 scinario de maison intelligente :

La maison intelligente, ou ”smart home”, est un domaine de l'IoT où des composants ou dispositifs communiquent entre eux. Dans cet exemple, les dispositifs sont le ”système”, le ”chauffage”, le ”climatiseur” et le ”capteur”. J'utilise ces dispositifs pour vérifier le système de contrôle de la température en utilisant le chauffage et le climatiseur.
Le rôle du capteur : est essentiel dans un système de maison intelligente, car il capte la température.
Le rôle du chauffage et du climatiseur : Le rôle du chauffage et du climatiseur est également essentiel dans un système de maison intelligente, car ils fonctionnent à une certaine température.
le rôle de systéme : Le système est l'une des parties les plus importantes d'un système de maison intelligente, car il est responsable de l'allumage et de l'extinction du chauffage et du climatiseur en fonction des informations reçues du capteur. Il assure la coordination des différents dispositifs et garantit le bon fonctionnement du système dans le but de maintenir une température confortable à l'intérieur de la maison.
Pour concevoir et vérifier ce modèle, nous utiliserons le diagramme d'activité et le diagramme de séquence. Nous utiliserons l'outil StarUML. Le diagramme d'activité représentera les processus, tandis que le diagramme de séquence illustrera les scénarios normaux et anormaux, ainsi que les contre-exemples pour la propriété ”never claim”.
Nous appliquerons la méthode de vérification formelle appelée ”model checking” pour vérifier le système en utilisant les contre-exemples générés par le diagramme de séquence. Pour cela, nous utiliserons le langage PROMELA, spécialement conçu pour la modélisation et la vérifica-

tion des systèmes concurrents, ainsi que l'outil SPIN Model Checker, particulièrement adapté à la vérification des protocoles de communication, des systèmes embarqués et des systèmes distribués.
En suivant ces étapes de conception, de modélisation et de vérification, nous pourrons garantir le bon fonctionnement du système de contrôle de la température de notre modèle de maison intelligente IoT.

4.3 le diagramme d'activité :

Le diagramme d'activité est une partie de la modélisation UML qui permet de représenter les activités et les processus du système IoT, tels que le système de contrôle de la température utilisant les systèmes de refroidissement et de chauffage, dans notre cas. Ce diagramme permet d'illustrer les différentes étapes et actions effectuées par le système pour maintenir la température souhaitée dans la maison intelligente. Il montre le flux des activités, les décisions prises, les branches conditionnelles et les boucles qui peuvent exister dans le système. Cela offre une vue claire du fonctionnement global du système et permet de mieux comprendre les processus impliqués dans le contrôle de la température.

Le diagramme suivant représente l'activité du scénario précédent :

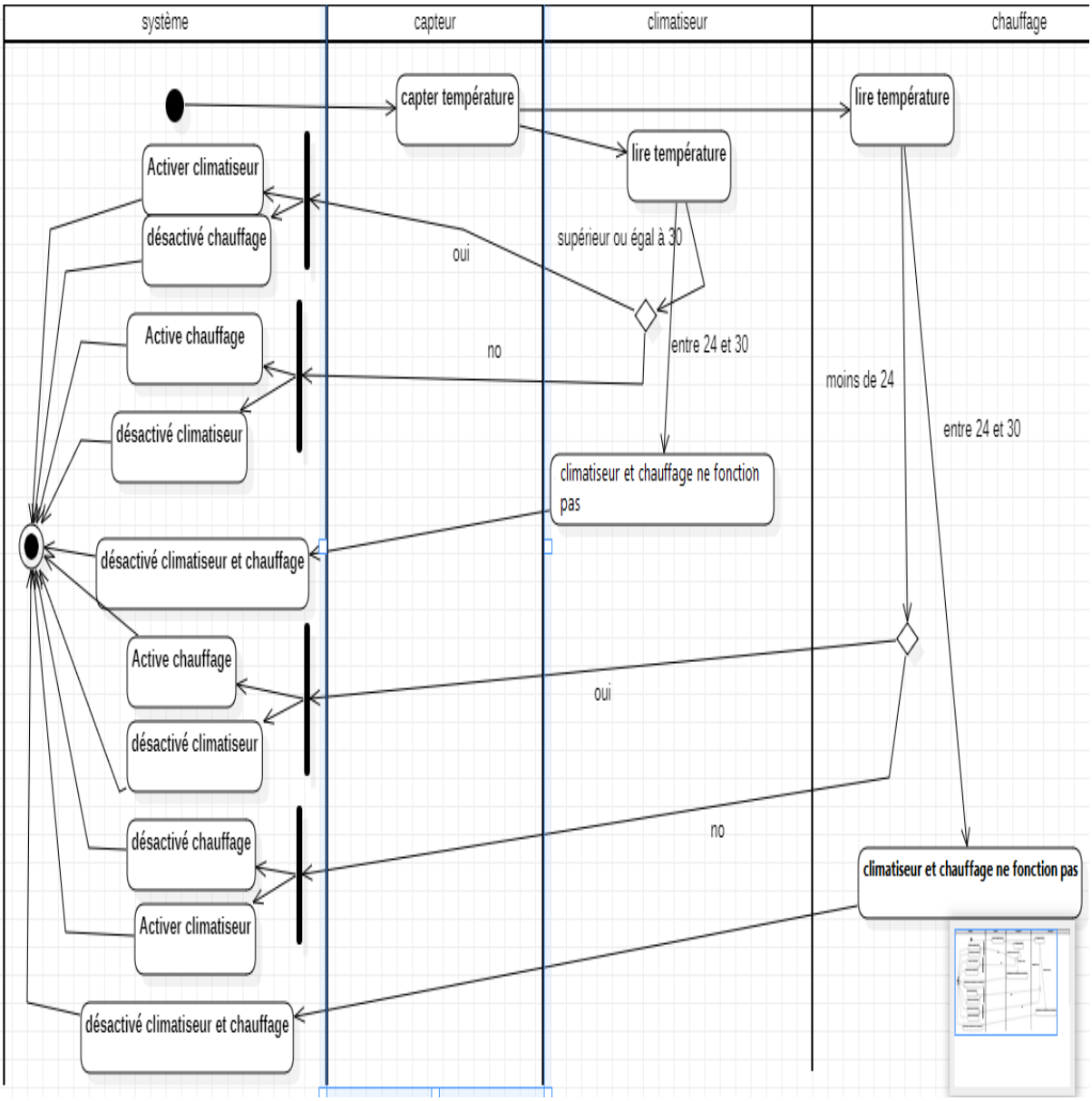


Figure 4.1 – diagramme d'activité

4.3.1 Le code du la transformation du diagramme d'activit

Le code suivant représente la transformation du diagramme d'activité :

```
 mtype = { clim_on, clim_off, chauf_on, chauf_off }
chan system_reaction = [0] of { mtype };
bool etat_clim = false;
bool etat_chauf = false;

int temperature = 18;
        bool climatisation = false;
    bool chauffagee= false;
active  [1] proctype capteur()
{
 end: do
  :: d_step { temperature++; }
// augmentation aléatoire de la température
   :: d_step { temperature--; }
// diminution aléatoire de la température
        od
        }
active [1] proctype climatiseur()
{
        end: do
:: (temperature < 30) ->
system_reaction!clim_off; etat_clim=false;
:: (temperature >= 30) ->
system_reaction!clim_on; etat_clim=true;
:: (temperature > 24 && temperature < 30)
-> system_reaction!clim_off; etat_clim=false
od
}
active [1] proctype chauffage()
{
end: do
:: (temperature < 24) ->
system_reaction!chauf_on; etat_chauf=true;
:: (temperature >= 24) ->
system_reaction!chauf_off; etat_chauf=false;
:: (temperature > 24 && temperature < 30)
-> system_reaction!chauf_off; etat_chauf=false;
```

```
od
        }
active [1] proctype systeme()
{
end: do
::system_reaction?chauf_on ->
if
:: (temperature < 5) -> temperature = temperature + 7;
:: (temperature < 10) ->temperature = temperature + 5;
:: (temperature < 15) ->temperature = temperature + 3;
:: else temperature = temperature + 1;
        fi
::system_reaction?clim_on ->
        if
:: (temperature > 50) -> temperature = temperature - 12;
:: (temperature > 35) -> temperature = temperature - 10;
:: (temperature > 30) -> temperature = temperature - 8;
:: else temperature = temperature - 1;
        fi
::system_reaction?_ -> temperature = temperature +  2;
::system_reaction?_ -> temperature = temperature -  2;
        od
        }
 ltl p1{ [](  !climatisation||  !chauffagee  ) }
```

4.3.2 Explication de code :

ce code defini quatre processuse communique entre eux

Définition des variables :

mtype : définit un ensemble de valeurs possibles pour le type, comprenant "`clim_on`", `clim_off`", "`chauf_on`" et "`chauf_off`".

`system_reaction` : canal utilisé pour l'échange de signaux entre les composants du système.

`etat_clim` et `etat_chauf` : variables booléennes (vrai ou faux) qui déterminent l'état des systèmes de climatisation et de chauffage respectivement.

température : variable représentant la température actuelle.

Le processus capteur : ce processucse capter la températeur (Augmente ou diminue la température de manière aléatoire).

Le processus climatiseur : Il surveille la température et envoie un signal (`clim_off`) sur le canal (`system_reaction`) lorsque la température est inférieure à 30.

Il surveille la température et envoie un signal (`clim_on`) sur le canal (`system_reaction`)

lorsque la température est supérieure ou égale à 30.
Il met à jour l'état du système de climatisation (`etat_clim`) en fonction du signal envoyé.
le processus chauffage : Il surveille la température et envoie un signal (`chauf_on`) sur le canal (`system_reaction`) lorsque la température est inférieure à 24.
Il surveille la température et envoie un signal (`chauf_off`) sur le canal (`system_reaction`) lorsque la température est supérieure ou égale à 24.
Il met à jour l'état du système de chauffage (`etat_chauf`) en fonction du signal envoyé.
le processus systéme : Le processus "système" est responsable de la réception des signaux sur le canal `"system_reaction"` et de l'ajustement de la température en conséquence.
Lorsqu'un signal est reçu sur le canal `"system_reaction"` , le processus "système" effectue une action en fonction du type de signal.

4.3.3 Présentation des automates

1. **Le automate de capteur :**

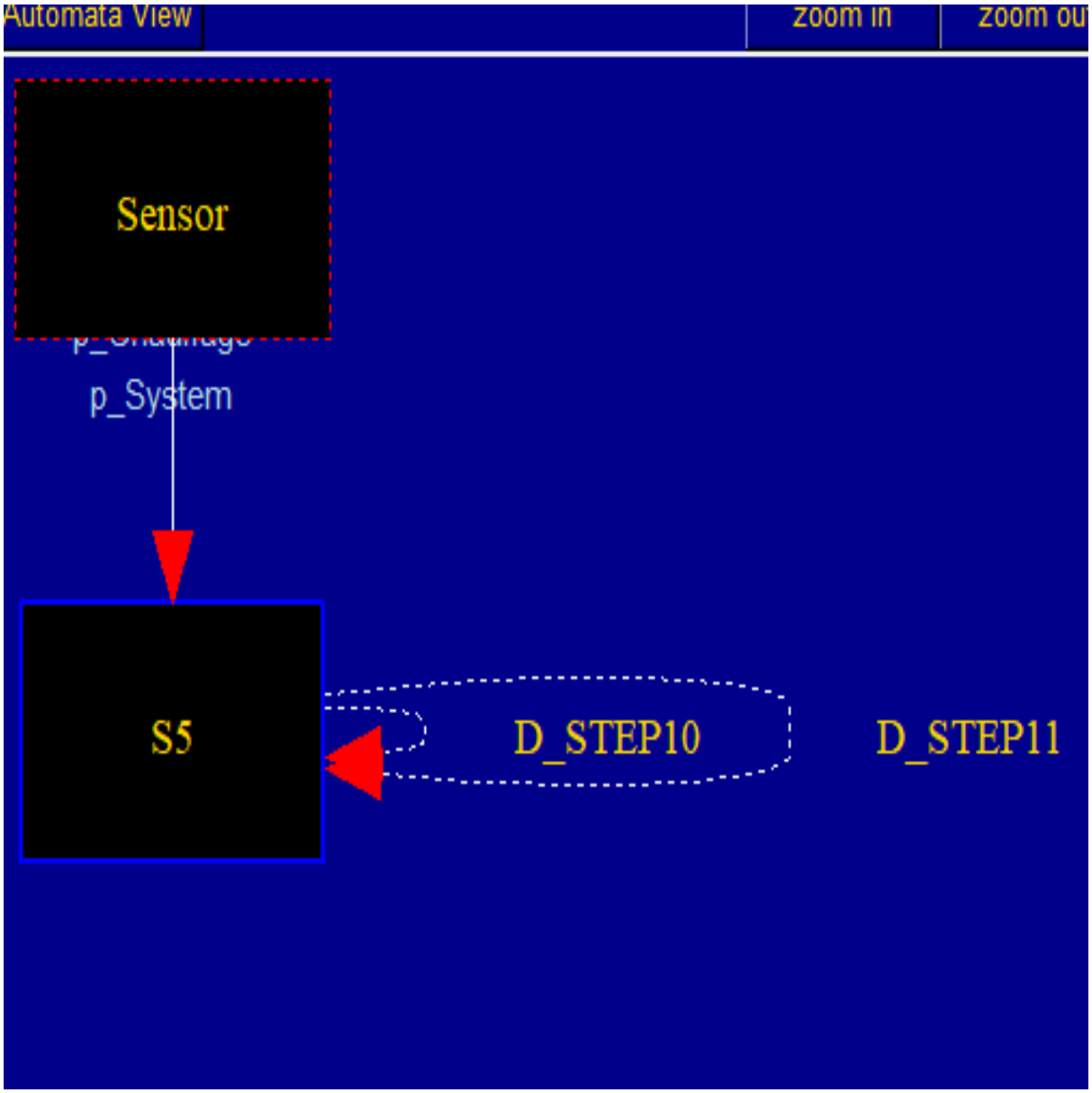


Figure 4.2 – L'automate de capteur

2. **L'automate de chauffage :**

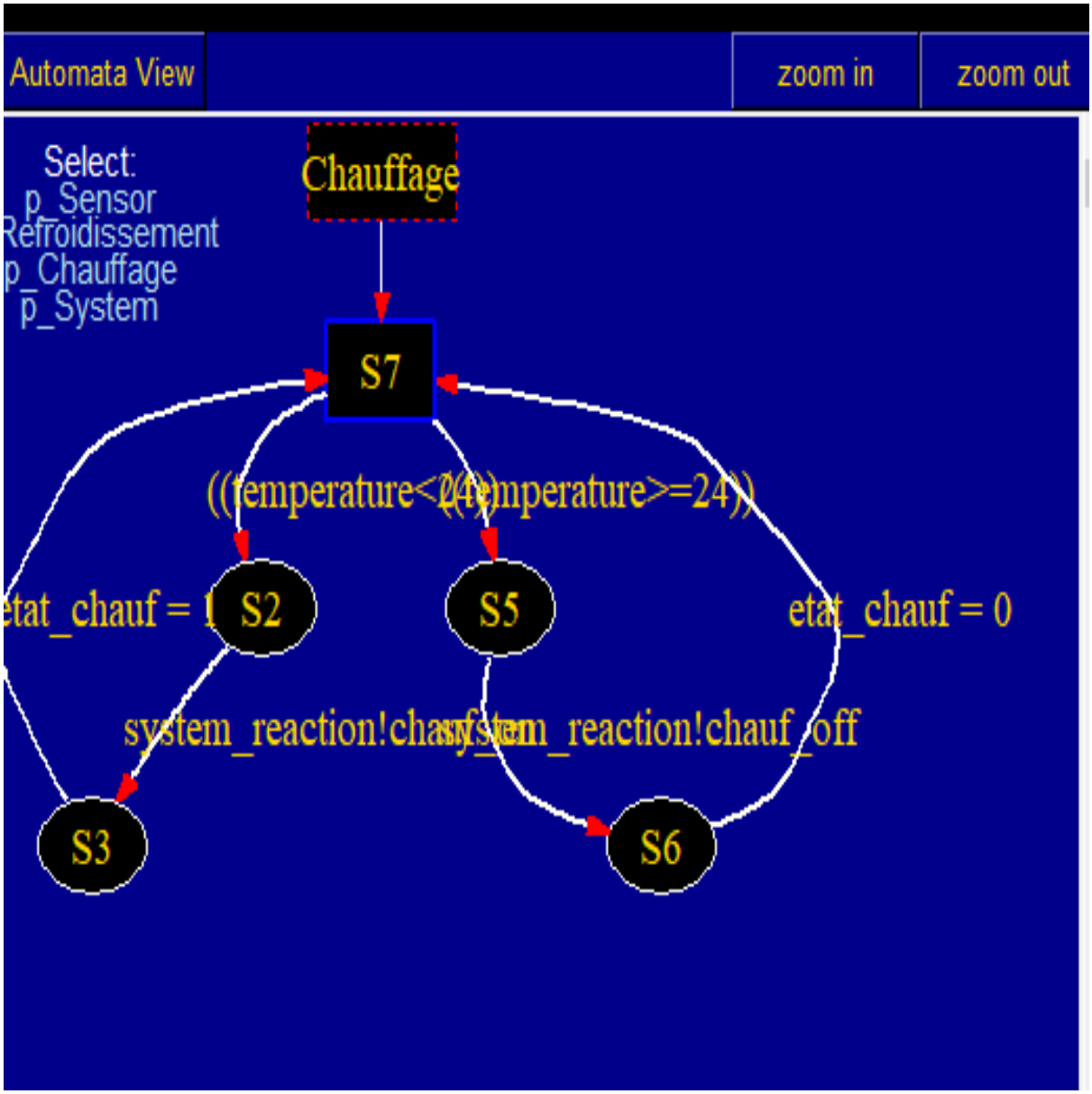


Figure 4.3 – L'automate de chauffage

3. **L'automate de climatiseur :**

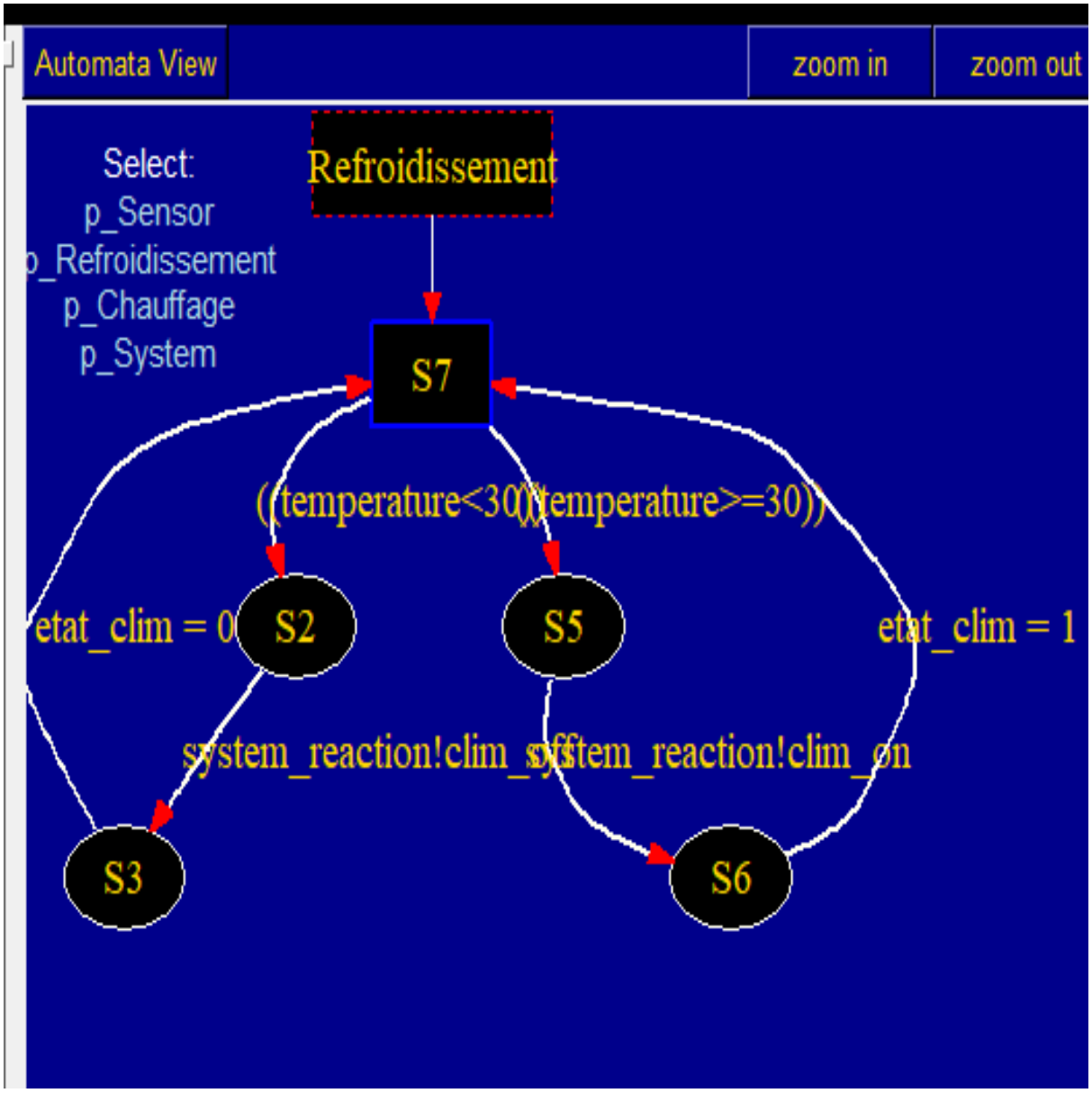


Figure 4.4 – L'automate de climatiseur

4. **L'automate de système :**

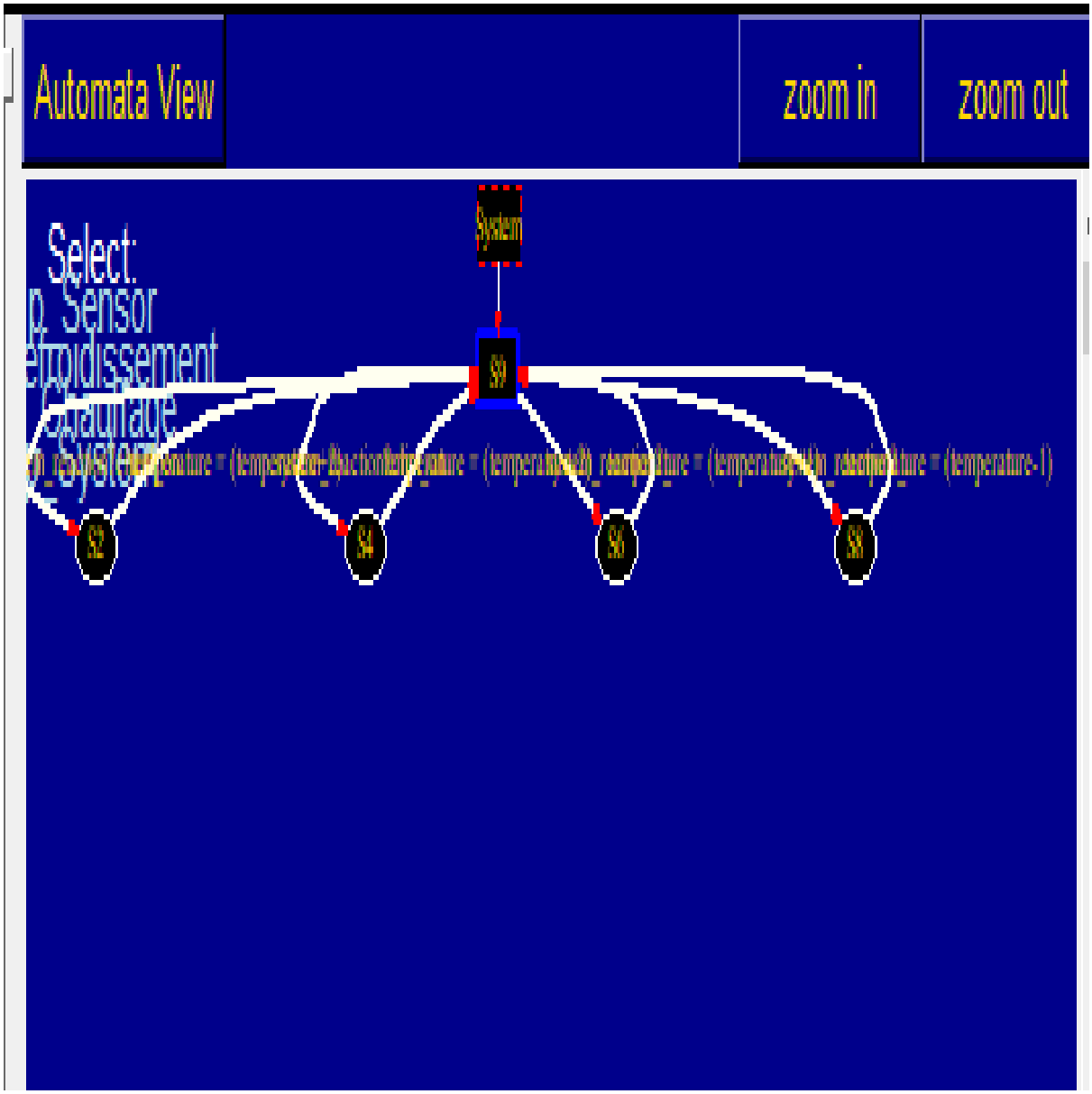


Figure 4.5 – L'automate de système

5. **L'automate de LTL :**

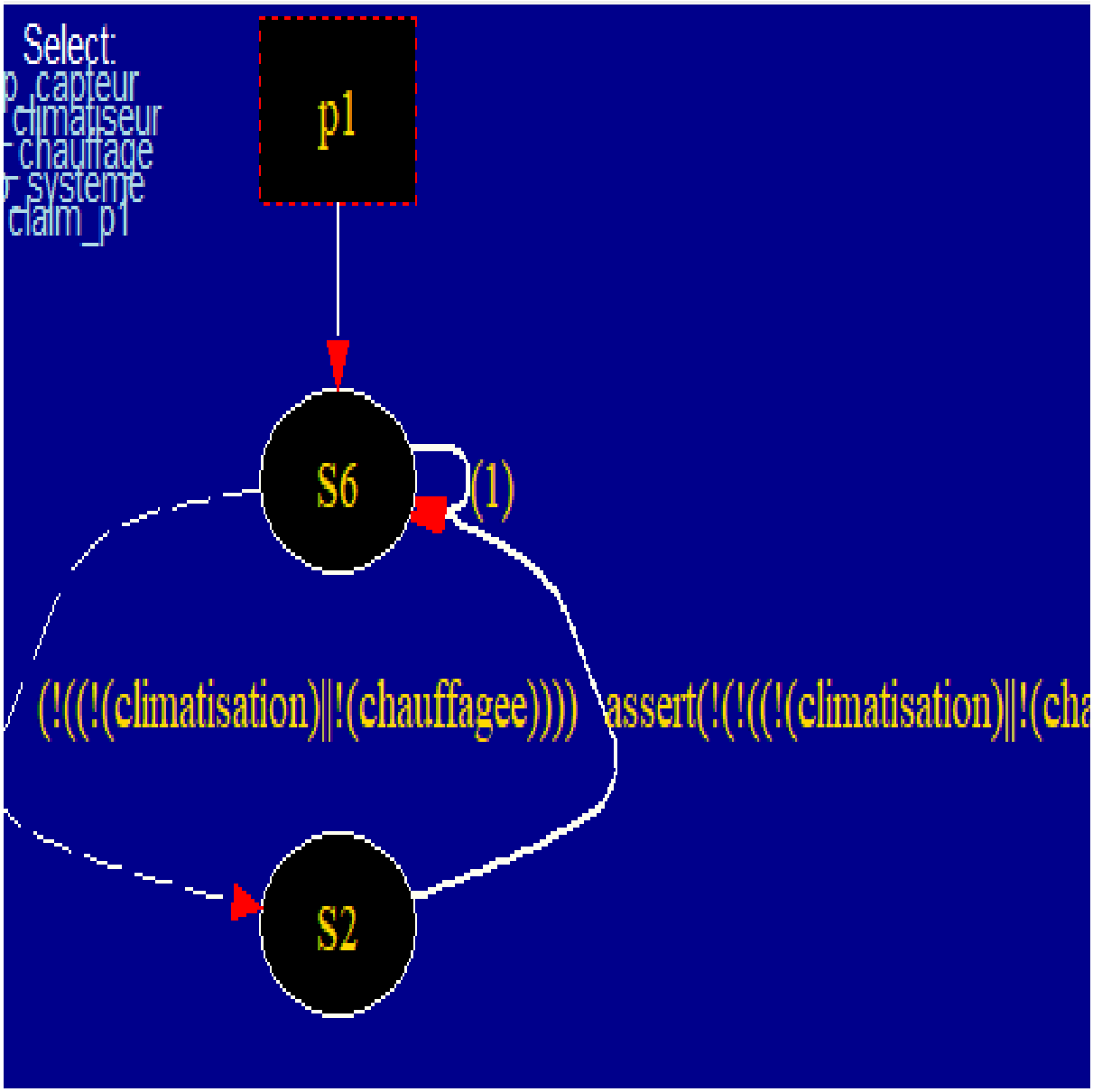


Figure 4.6 – L'automate de LTL

4.3.4 Simulation avec SPIN model checker :

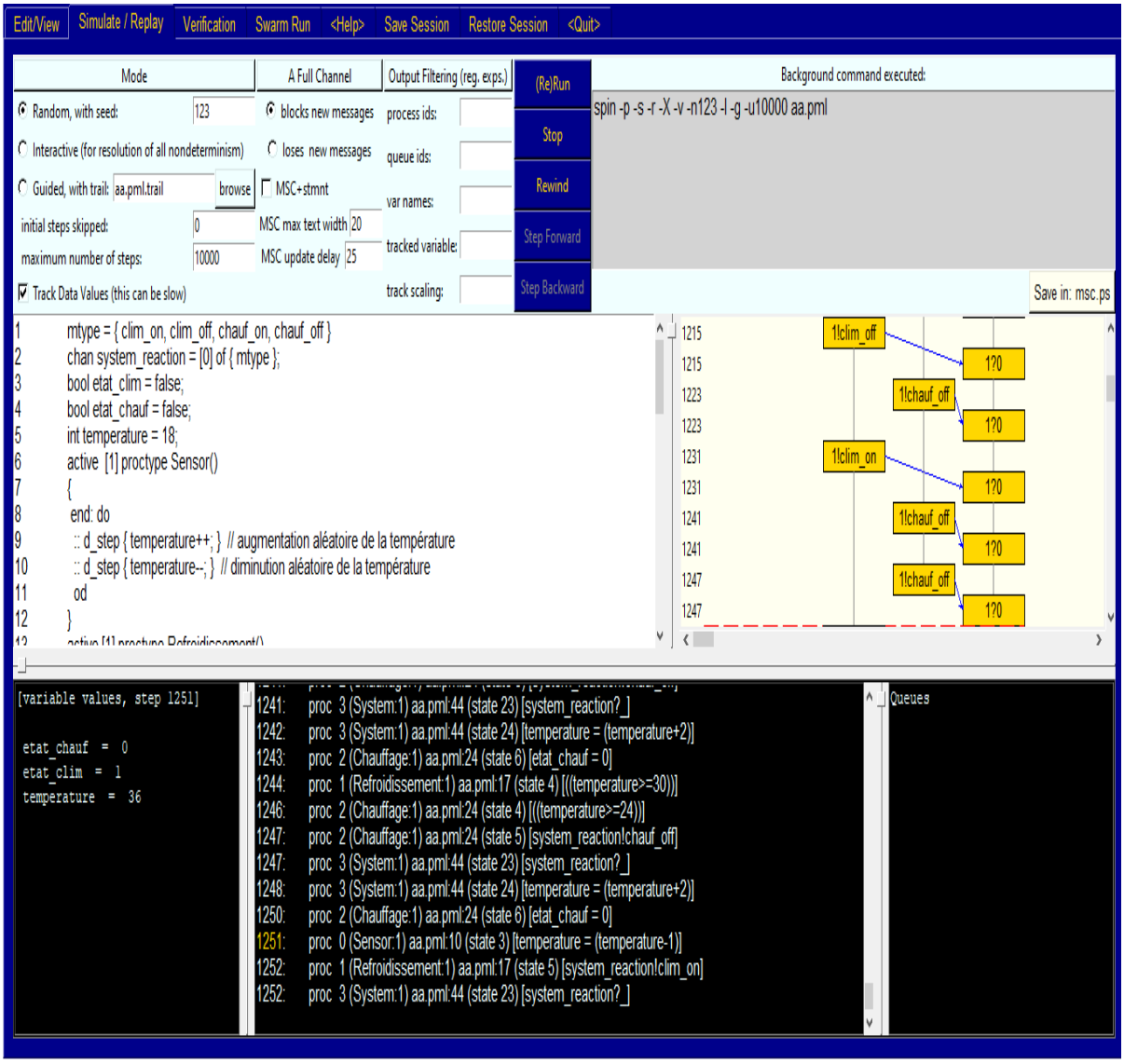


Figure 4.7 – Simulation avec SPIN model checker

4.4 Le diagramme de séquence :

Le diagramme de séquence, représente le scénario normal d'utilisation du système, décrivant les interactions entre les différents composants du système, tels que sensor, system, climatiseur et de chauffage. Il peut également être utilisé pour modéliser des scénarios anormaux ou des contre-exemples, où des situations inattendues peuvent se produire, telles qu'une panne du système.

le diagramme suivant presanté le diagramme de séquence :

1. **Scénario normale :**

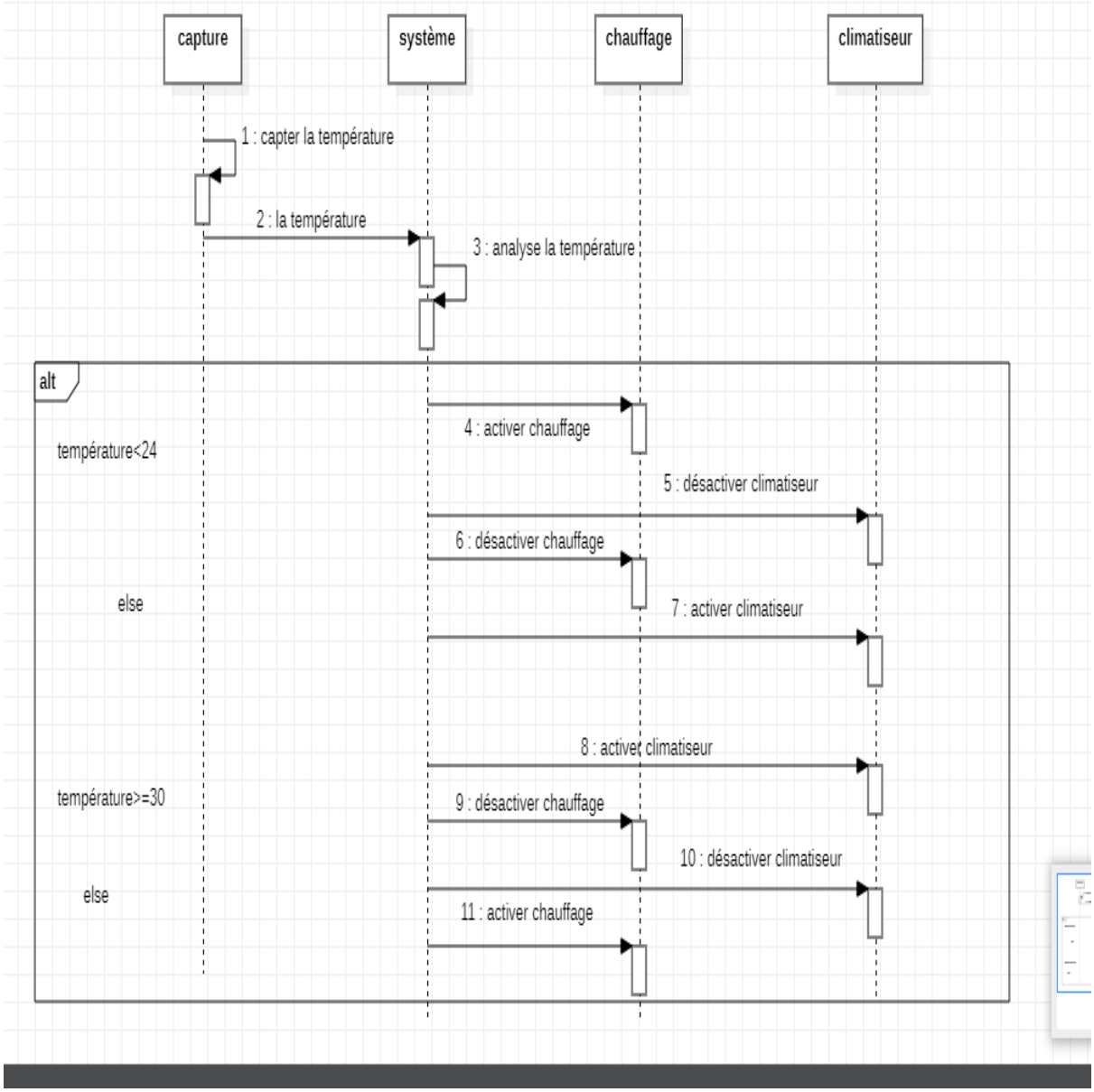


Figure 4.8 – diagramme de séquence

la propriété LTL ltl p1{ [](!climatisation || !chauffagee) } Cette propriété garantit qu'à tout moment, soit le climatiseur n'est pas activé, soit le chauffage n'est pas activé. Si les deux étaient activés en même temps à n'importe quel instant, la propriété serait violée.

En d'autres termes, la propriété p1 assure qu'il n'y a pas de conflit entre l'utilisation du climatiseur et du chauffage dans le système considéré. Cela permet d'éviter des situations indésirables où ces deux appareils fonctionnent simultanément et peuvent entraîner des incohérences dans le système de régulation de la température.

2. **Scénario Anormale :**

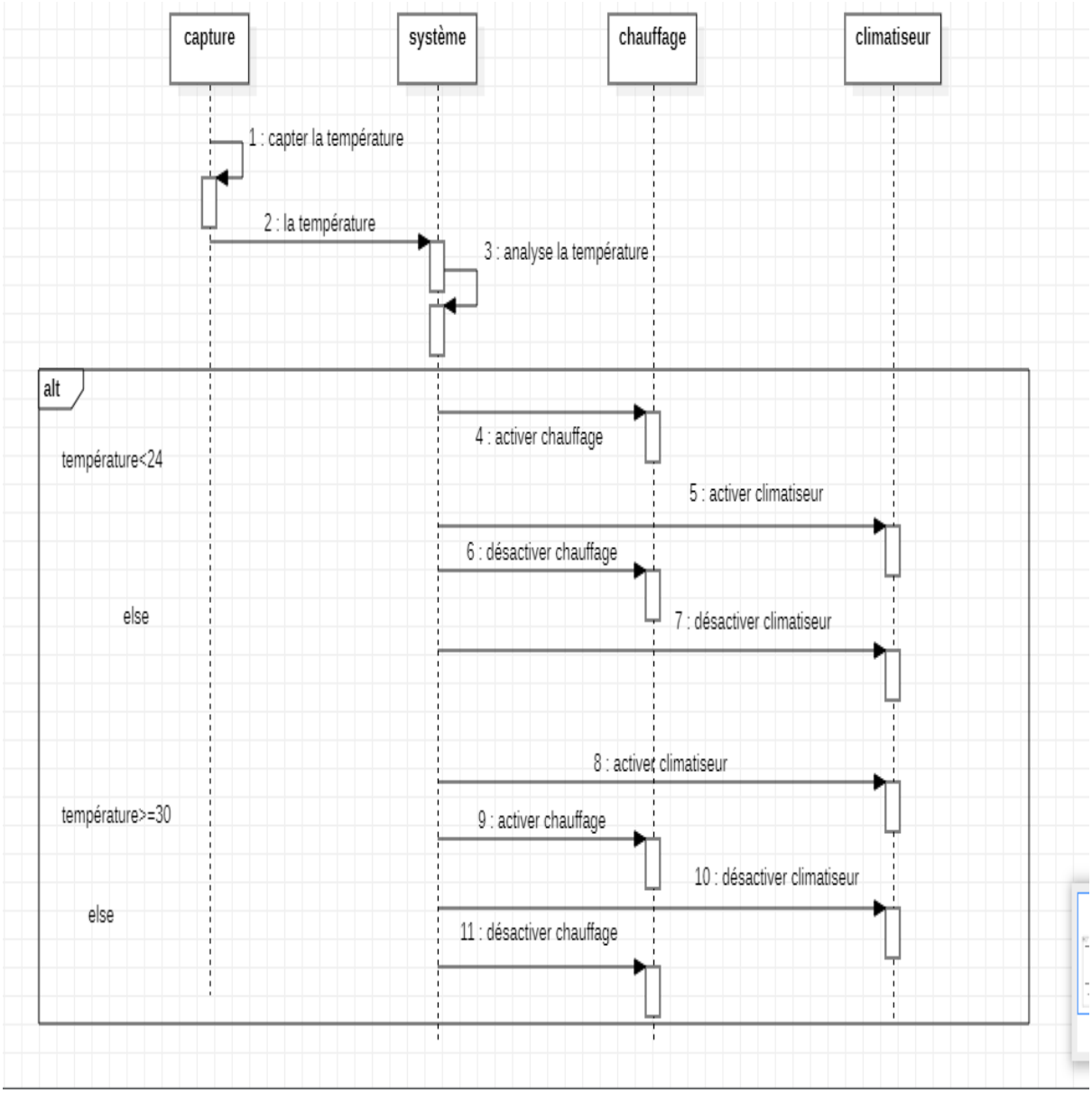
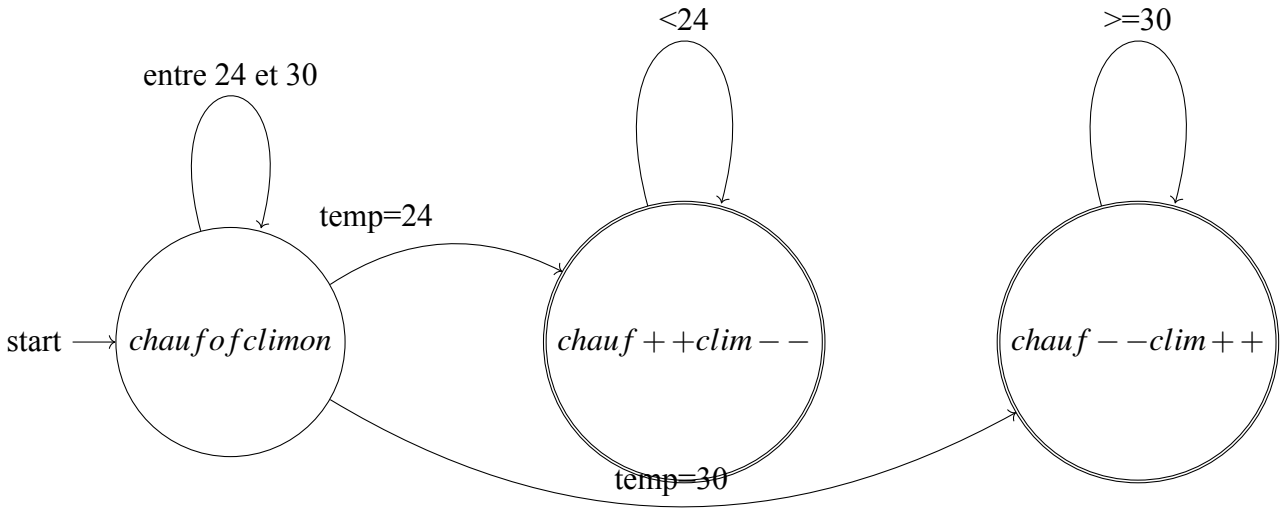


Figure 4.9 – diagramme de séquence Scénario Anormale

4.5 l'automate de Büchi :



4.6 Conclusion :

en utilise Spin, nous avons vérification de la fiabilite du systeme iot (chauffage , système ,capteur et climatiseur)en utilisé un diagramme d'activité et deux diagramme de séquence, tels que diagramme de séquence de scénario normale et scénario Anormale.

Conclusion Générale

Ce mémoire aborde différents aspects de l'Internet des objets (IoT) et de la vérification formelle, en mettant l'accent sur l'utilisation du model checking (Spin) pour vérifier la fiabilité des systèmes IoT, en particulier en se concentrant sur le modèle d'une maison intelligente.

La première partie présente une vue d'ensemble de l'Internet des objets, en mettant l'accent sur les maisons intelligentes en particulier.

La deuxième partie met en avant l'utilisation de l'UML (Unified Modeling Language) pour modéliser un système, en soulignant son importance dans les systèmes IoT.

Le chapitre 3 se concentre sur la vérification formelle, une méthode puissante pour garantir la fiabilité des systèmes en effectuant des analyses approfondies et systématiques, y compris les techniques de model checking.

Ensuite, les concepts théoriques sont appliqués à la vérification d'un modèle de maison intelligente. L'UML est utilisé pour spécifier le comportement attendu de la maison intelligente, en prenant en compte les dispositifs et capteurs impliqués dans la gestion du chauffage et de la climatisation en fonction de la température.

En utilisant les automates de Büchi, la vérification est effectuée pour s'assurer que le comportement de la maison intelligente satisfait les propriétés de fiabilité spécifiées.

Bibliographie

[1] K. K. Patel, S. M. Patel, and P. Scholar, “Internet of things-iot : definition, characterisT-tics, architecture, enabling technologies, application&future challenges,” International journal of engineering science and computing, vol. 6, no. 5, 2016

[2] https ://micro.ai/blog/best-examples-of-iot

[3] S. SAHRAOUI, “Mécanismes de sécurité pour l’intégration des rcsfs à l’iot (internet of things),” Université de Batna 2 09/11/2016. page (46-47)

[4] D. P. K. E. MAZOBA, “L’étude de l’internet des objets et contrôle d’accès aux données.,” Université Panafricaine - Licence en Génie Informatique 2015. https ://www.memoireonline.com/11/19/11246/L-etude-de-l-internet-des-objets-et-contrle-d-acces-aux-donnees.html

[5] M. Larras and D. Khalfouni, “Thème : Défis de sécurité de l’internet des objets - problèmes et solutions,” 2018/2019. Page 11.

[6] M. Larras and D. Khalfouni, “Thème : Défis de sécurité de l’internet des objets - problèmes et solutions,” 2018/2019, Page 12.

[7] Hadjadj Walid,thème étude de cas sur un système médical domotique controlé par un SMA, 2017/2018 page 21/22 .

[8] http ://c.herblot.free.fr/cours42009/portail/capteur/capteurs.htm

[9] http : //technologieaucollege.free.fr/ressources_web/ressources2. techno.free.fr/mecanique/systemes/actionneur.htm

[10] M. Nor and houda Bendjema Roufaida, “Mécanismes de sécurité pour l’intégration des rcsfs à l’iot (internet of things),” Université 8Mai 1945 – Guelma , Juillet 2019

[11] S. SAHRAOUI, “Mécanismes de sécurité pour l’intégration des rcsfs à l’iot (internet of things),” Université de Batna 2 09/11/2016. page (52,52)

[12] https ://sbr-technologies.com/iot-architecture/

[13] M. Larras and D. Khalfouni, “Thème : Défis de sécurité de l’internet des objets - problèmes et solutions,” 2018/2019. Page 15-16-17

[14] https ://wikimemoires.net/2019/09/domaines-d-applications-de-l-iot/

[15] ”Smart Homes : Past, Present, and Future” - Article publié dans IEEE Transactions on Consumer Electronics en 2018. Il examine l’évolution des maisons intelligentes, les technologies utilisées et les perspectives futures.Auteur, Titre de la source, Éditeur, Année.

[16] M. H. Zakaria and M. M. M. Othman, “thème vers des bâtiments intelligents pour l’élevage de volaille,” 2018/2019.

[17] M. Hanane and B. Amina, “Conception d’une maison intelligente avec les réseaux m2m/iot,” 16 / 09 / 2021.

[18] M. H. Zakaria and M. M. M. Othman, “thème vers des bâtiments intelligents pour l’élevage de volaille,” 2018/2019

[19] P. K. Paul and S. K. Paul, ”Smart Refrigerator : A Review of the State-of-the-Art and Future Perspectives,” 2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), Silchar, India, 2018, pp. 1-6.

[20] M. Hanane and B. Amina, “Conception d’une maison intelligente avec les réseaux m2m/iot,” 16 / 09 / 2021. Page 13

[21] Geller, M and de Moura Meneses, Anderson Alvarenga,Modelling IoT systems with UML : a case study for monitoring and predicting power consumption,American Journal of Engineering and Applied Sciences,2021.

[22] M. Fowler, “Uml distilled : a brief guide to the standard object modeling language,” 2004.

[23] La¨etitia Matignon ,3. UML - Unified Modeling Language Diagrammes statiques,https ://-perso.liris.cnrs.fr/laetitia.matignon/index/ISI32012/cours2diagStatiq.pdf.

[24] M. Fowler, “Uml distilled : a brief guide to the standard object modeling language,” 2004.

[25] . Fowler, “Uml distilled : a brief guide to the standard object modeling language,” 2004

[26] . Fowler, “Uml distilled : a brief guide to the standard object modeling language,” 2004

[27] Baier, C, Katoen, J.-P. (2008), Principles of Model Checking. MIT Press.

[28] Baier, C., Katoen, J.-P. (2008). Principles of Model Checking. MIT Press.

[29] Baier, C., Katoen, J.-P. (2008). Principles of Model Checking. MIT Press.

[30] M. Toufik, “cour de validation et vérification des systèmes distribués,”

[31] Baier, C., Katoen, J.-P. (2008). Principles of Model Checking. MIT Press.

[32] Baier, C., Katoen, J.-P. (2008). Principles of Model Checking. MIT Press.

[33] M. Toufik, “cour de validation et vérification des systèmes distribués,”

[34] M. Toufik, “cour de validation et vérification des systèmes distribués,”

[35] M. Toufik, “cour de validation et vérification des systèmes distribués,”

[36] M. Toufik, “cour de validation et vérification des systèmes distribués,”

[37] M. Toufik, “cour de validation et vérification des systèmes distribués,”

[38] les automates,” https ://www.i3s.unice.fr/ fedou/wwwUnice/CoursOFI f iles/OFI8.pd f

[39] Automates finis,” Jean-Eric Pin”

[40] Automates finis,” Jean-Eric Pin”

[41] Automates finis,” Jean-Eric Pin”

[42] Automates finis,” Jean-Eric Pin”

[43] CARNINO, “doctorat de l’université paris-est autour des automates : génération aléa- toire et contribution à quelques extensions,” le 5 décembre 2014.

[44] S. Lombardy, “Automates notions de base notes de cours,” 200

[45] G. Holzmann, “The spin model checker,” 2003. 608 pages

[46] A. CHAMI, “ thème VÉrification de processus bpel À l’aide de promela-spin,” MARS 2008