Travaux Pratiques

L'objectif de ces travaux pratiques est d'implémenter des protocoles de synchronisation et des protocoles de ranging en utilisant les bonnes propriétés de marquage temporel des signaux UWB.

Table of Contents

---

1. Étape préliminaire

1.1. Ressources

Table 1. Ressources disponibles

Ressource	URL
DecaDuino library	https://www.irit.fr/~Adrien.Van-Den-Bossche/decaduino/
DecaWiNo's page	https://wino.cc/decawino/
SiSP protocol	https://www.irit.fr/~Adrien.Van-Den-Bossche/sisp.pdf

1.2. Vérification de l'environnement de travail

- Matériellement, chaque étudiant dispose d'un PC, d'une carte DecaWiNo, d'un câble USB, d'un oscilloscope et d'une connectique de raccordement à l'oscilloscope.
- Logiciellement, l'environnement Arduino est utilisé pour l'ensemble de ces exercices. Sur le poste de travail, vérifier la présence :
  - du logiciel Arduino,
  - de l'*add-on* TeensyDuino : `File > Examples > Teensy`
  - de la librairie DecaDuino : `File > Examples > DecaDuino`

Si l'un de ces points n'est pas vérifié, procéder à l'installation avant de poursuivre. Le site web indiqué en début de document contient toutes les informations pour procéder.

2. Prise en main du transceiver UWB

2.1. Prise en main des sketchs exemples

L'objectif de cette section est de prendre en main le PLME-SAP.

La librairie DecaDuino comprend plusieurs sketchs d'exemples disponibles dans `File > Examples > DecaDuino`.

1. Ouvrir le sketch `DecaDuinoSender` et le sketch `DecaDuinoReceiverSniffer`, lister les primitives utilisées et comprendre leur utilité à l'aide de la documentation de DecaDuino.
2. Constater l'absence d'adressage dans les messages, et modifier les sketchs de manière à mettre en place une isolation logique entre les équipes de TP : chaque équipe ne doit recevoir que les messages qui la concerne.

2.2. Conception d'un premier protocole simple

L'objectif de cette partie est de coder et tester un premier protocole simple de type `DATA+ACK`*.*

1. Implémenter un simple DATA+ACK avec des valeurs `timeout` et `retries` paramétrables.
2. Visualiser les temps d'émission, IFS, timeout, etc. à l'aide de l'oscilloscope.

Pour visualiser les temps, programmer le MCU pour générer des états hauts/bas sur une ou deux GPIO et utiliser un oscilloscope pour visualiser les signaux.

Veiller à systématiquement désactiver la réception avant de demander l'émission d'un message.

3. Synchronisation MAC

3.1. Implémentation d'une synchronisation en étoile

L'objectif de cette section est d'utiliser le PLME-SAP et les primitives d'estampillage temporel pour implémenter un premier protocole de synchronisation simple en étoile.

Le transceiver DW1000 présent sur le DecaWiNo permet d'estampiller très précisément les messages émis et reçus. Pour cela, DecaDuino met à disposition du développeur plusieurs primitives. Dans cet exercice, nous allons les utiliser pour implémenter un premier protocole de synchronisation en étoile, comme illustré sur la figure suivante.

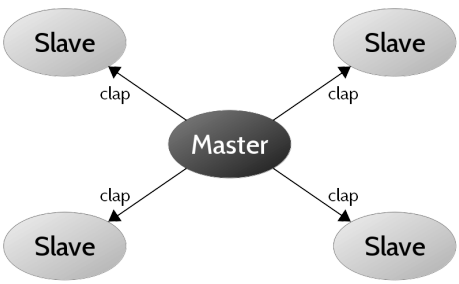

Figure 1. Synchronisation en étoile

Régulièrement, le nœud *Master* diffuse un message `clap` indiquant l'heure à laquelle il envoie le clap. Les nœuds *Slave* reçoivent ce message, calculent leur *offset* avec le nœud maître et l'appliquent sur leur heure locale.

1. Identifier les primitives nécessaires
2. Concevoir le protocole :
   a. Proposer la séquence des messages et leur format,
   b. Pour chaque type de nœud (maître, esclave), préciser les algorithmes locaux,
   c. Proposer un automate.
3. Implémenter et tester

Pour visualiser la qualité de la synchronisation, programmer le MCU pour générer un front sur une GPIO et utiliser un oscilloscope pour visualiser cette différence sur plusieurs DecaWiNos.

3.2. Implémentation d'une synchronisation de SiSP

L'objectif de cette section est d'utiliser le PLME-SAP et les primitives d'estampillage temporel pour implémenter le protocole SiSP.

Dans cet exercice, nous allons implémenter un second protocole de synchronisation. Cette fois, il n'y a plus de hiérarchie entre les nœuds participants, comme illustré sur la figure suivante.

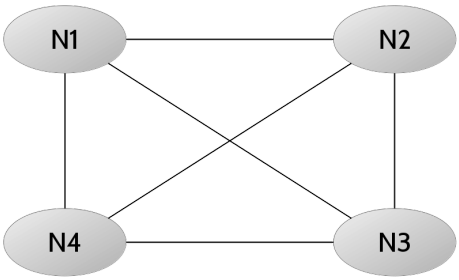

Figure 2. Synchronisation maillée sans hiérarchie

Ici, chaque nœud diffuse régulièrement un message `clap`, mais la synchronisation sera obtenue par une série de consensus, par les nœuds participants, selon le principe du protocole SiSP.

1. Identifier les primitives nécessaires
2. Concevoir le protocole :
   a. Proposer la séquence des messages et leur format,
   b. Pour chaque type de nœud (maître, esclave), préciser les algorithmes locaux,
   c. Proposer un automate.
3. Implémenter et tester

Là encore, pour visualiser la qualité de la synchronisation, programmer le MCU pour générer un front sur une GPIO et utiliser un oscilloscope pour visualiser cette différence sur plusieurs DecaWiNos. Il est également possible d'utiliser la LED RGB présente sur le DecaWiNo pour illustrer très visuellement le consensus de synchronisation.

4. Ranging (synchronisation fine)

L'objectif de cette section est d'utiliser le PLME-SAP et les primitives d'estampillage temporel pour implémenter des protocoles de ranging.

Grâce à l'estampillage très précis des messages à l'émission et à la réception (15 ps), il est possible d'utiliser les messages émis et reçus par le DWM1000 pour mesurer le *temps de vol radio* et obtenir ainsi une évaluation relativement précise de la distance séparant les deux nœuds. C'est le processus de *ranging*.

4.1. Sketchs exemples

La librairie DecaDuino comprend plusieurs sketchs d'exemples implémentant dans des protocoles de *ranging*. Ils sont disponibles dans `File > Examples > DecaDuino`.

- A l'aide de la documentation présente en ligne (Documentation de DecaDuino et tutoriel intitulé "*Using DecaWiNo and ranging protocols*"), prendre en main les différents sketchs d'exemple et comprendre le fonctionnement des protocoles Two-Way Ranging (TWR) et Symmetric Double Sided Two-Way Ranging (SDS-TWR).

- Procéder à quelques tests. Là encore, constater l'absence d'adressage dans les messages et modifier les sketchs de manière à isoler logiquement les équipes entre elles.
- Constater par la pratique l'effet de la dérive des horloges (*skew*) sur la précision du *ranging* dans TWR.

4.2. Implémentation de 2M-TWR

L'objectif de cette section est de proposer une implémentation du protocole Two-Messages TWR (2M-TWR).

Le protocole 2M-TWR (cf. cours) est une amélioration du protocole TWR avec une réduction à deux messages. Les estampilles t2 et t3 sont envoyées dans le message `ACK`, ce qui permet d'économiser le dernier message comme illustré par la figure suivante.

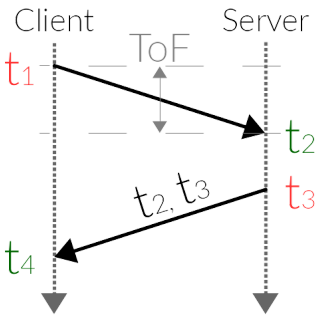

Figure 3. Diagrammes de séquence de TWR et 2M-TWR

Procéder à la conception, à l'implémentation et au test du protocole.

1. Identifier les primitives nécessaires
2. Concevoir le protocole :
   a. Proposer un format pour les messages échangés,
   b. Pour chaque type de nœud (maître, esclave), préciser les algorithmes locaux,
   c. Proposer un automate.
3. Implémenter et tester

4.3. Implémentation de N-TWR

L'objectif de cette section est de proposer une implémentation du protocole N-TWR.

Le protocole N-TWR (cf. cours) est une amélioration du protocole 2M-TWR où plusieurs ancres (serveurs) peuvent être sollicitées par le mobile (client) avec un unique message. Les estampilles t2 et t3 de chaque ancres sont envoyées dans les messages `ACK`.

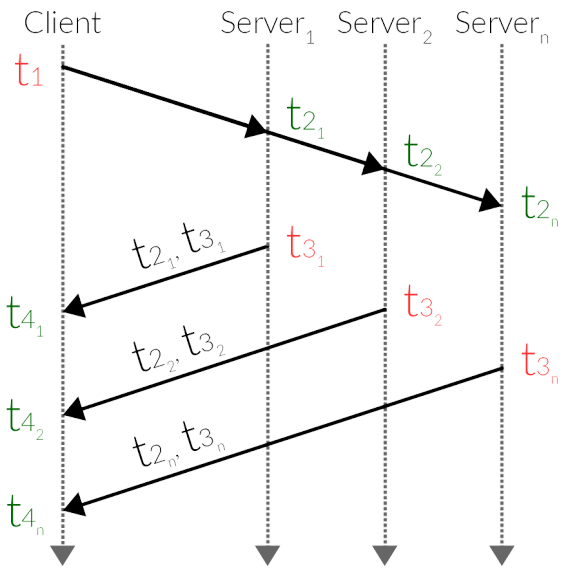

Figure 4. Diagrammes de séquence de BB-TWR pour trois nœuds en triangle

Procéder à la conception, à l'implémentation et au test du protocole.

1. Identifier les primitives nécessaires
2. Concevoir le protocole :
   a. Proposer un format pour les messages échangés,
   b. Pour chaque type de nœud (maître, esclave), préciser les algorithmes locaux,
   c. Proposer un automate.
3. Implémenter et tester

4.4. Implémentation de BB-TWR

L'objectif de cette section est de proposer une implémentation du protocole Broadcast-Based TWR (BB-TWR).

Le protocole BB-TWR (cf. cours) est une version diffusée (*broadcast*) et généralisée à n nœuds du protocole 2M-TWR, comme illustré par la figure suivante.

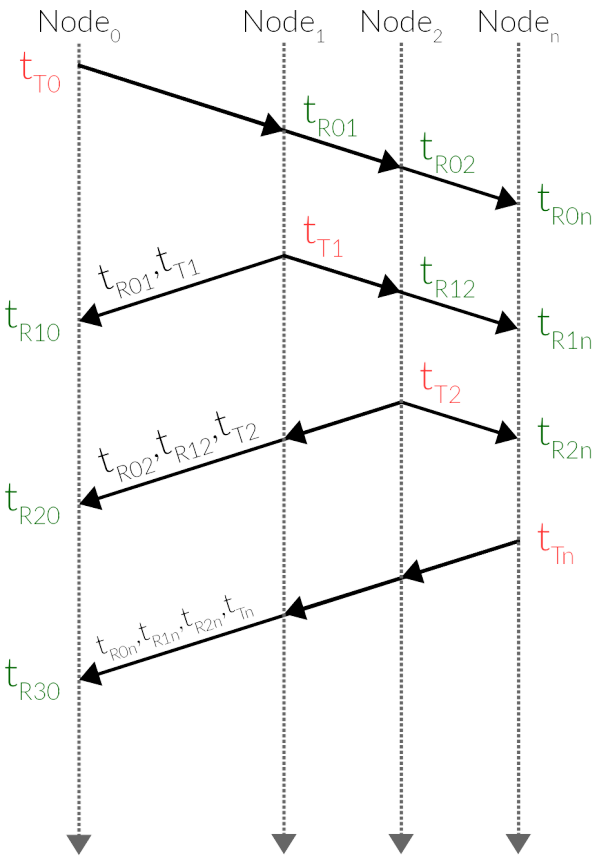

Figure 5. Diagrammes de séquence de BB-TWR pour trois nœuds en triangle

Procéder à la conception, à l'implémentation et au test du protocole.

1. Identifier les primitives nécessaires
2. Concevoir le protocole :
   a. Proposer un format pour les messages échangés,
   b. Pour chaque type de nœud (maître, esclave), préciser les algorithmes locaux,
   c. Proposer un automate.
3. Implémenter et tester