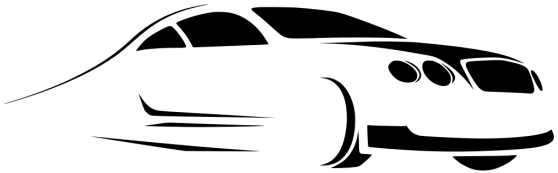
Université de Lille

POLYTECH



PFE 13 : Réseau de capteurs pour parking intelligent

Cartier Baptiste
IMA5 SC - Polytech Lille

Sommaire

1. Contexte
2. Cahier des charges
3. Travail effectué
4. Travail restant et planning

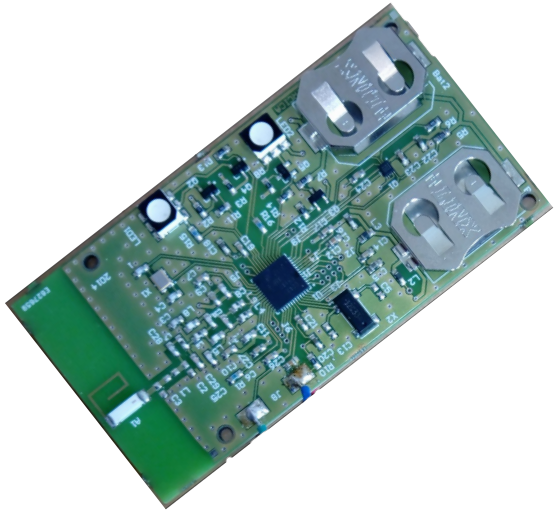
1.
Contexte

Problématique générale

- Fastidieux
- Chronovore
- Polluant

Problématique générale

- Connue
- Existante
- Adaptée



2.
Cahier des charges

Objectif

Déployer un réseau d'objets sans fils possédant des capacités de routage dynamique

Cahier des charges

Important

Routage dynamique RPL

CC430

Taille

Pas important

Durée de vie

Capteur

Affichage

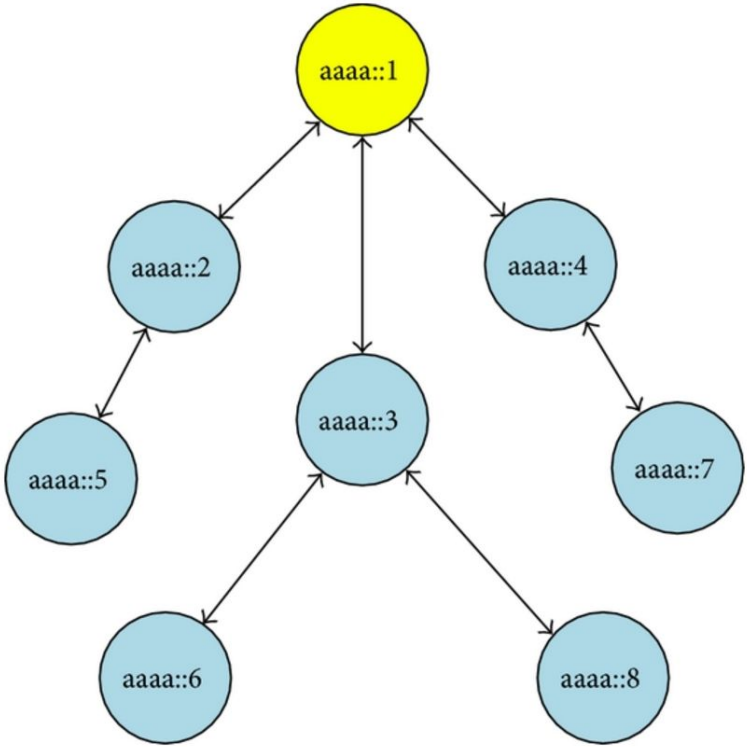
3.
Travail effectué

3.
Travail effectué

Exploration des possibilités et étude de faisabilité

RPL

- IoT
- Robuste
- Dynamique



Systèmes d'exploitation

RIOT OS

RPL

CC430

Connu

Contiki

RPL

Léger

MSP430

Systèmes d'exploitation

Tmote Sky

RIOT et Contiki

64 ko de ROM

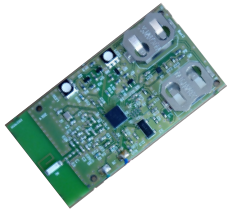
10 ko de RAM

CC430

RIOT

32 ko de ROM

4 ko de RAM



Systèmes d'exploitation

RIOT OS

Pas de réponse

64 ko de ROM

6.5 ko de RAM

Shell

Contiki

Peu de réponse

43 ko de ROM

6.9 ko de RAM

Threads “à la main”

Contiki

ROM	RAM
41 ko	3.8

Configuration

ROM	RAM
35 ko	3.5

Retrait des capteurs
et liaison filaire

ROM	RAM
31 ko	3.4

Retrait du driver du cc2420

3.
Travail effectué

Extraction

Départ

Taille estimée : 28.5 ko

IP

6LowPAN

RPL

MAC

Fonctionnement de Contiki

```
#define PROCESS(name, strname)          \
  PROCESS_THREAD(name, ev, data);       \
  struct process name = { NULL,         \
                          process_thread_##name }
```

Fichier	Nombre d'occurence de "PROCESS"	Importance de "PROCESS"
netstack.c	3	Aucune
netstack.h	1	aucune
resolv.c	18	forte
resolv.h	1	aucune
simple-udp.c	10	forte
tcpip.c	18	forte
tcpip.h	1	aucune
udp-client.c	6	moyenne
udp-socket.c	11	forte

Fonctionnement de Contiki

```
PROCESS_THREAD(tcpip_process, ev, data)
{
  PROCESS_BEGIN();

  tcpip_event = process_alloc_event();

  etimer_set(&periodic, CLOCK_SECOND / 2);

  uip_init();

  NETSTACK_ROUTING.init();

  while(1) {
    PROCESS_YIELD();
    eventhandler(ev, data);
  }

  PROCESS_END();
}
```

Fonctionnement de Contiki

```
static char process_thread_tcpip_process(struct pt *process_pt, process_event_t ev, process_data_t data)
{
  { char PT_YIELD_FLAG = 1; if (PT_YIELD_FLAG) {;} switch((process_pt)->lc) { case 0:;

  tcpip_event = process_alloc_event();

  etimer_set(&periodic, 128UL / 2);

  uip_init();

  rpl_lite_driver.init();

  while(1) {
    do { PT_YIELD_FLAG = 0; (process_pt)->lc = 837; case 837:; if(PT_YIELD_FLAG == 0) { return 1; } } while(0);
    eventhandler(ev, data);
  }

  }; PT_YIELD_FLAG = 0; (process_pt)->lc = 0;; return 3; };
}
```

Fonctionnement de Contiki

```
static char process_thread_tcpip_process(struct pt *process_pt, process_event_t ev, process_data_t data)
{
  { char PT_YIELD_FLAG = 1; if (PT_YIELD_FLAG) {;} switch((process_pt)->lc) { case 0:;

  tcpip_event = process_alloc_event();

  etimer_set(&periodic, 128UL / 2);

  uip_init();

  rpl_lite_driver.init();

  while(1) {
    do { PT_YIELD_FLAG = 0; (process_pt)->lc = 837; case 837:; if(PT_YIELD_FLAG == 0) { return 1; } } while(0);
    eventhandler(ev, data);
  }

  }; PT_YIELD_FLAG = 0; (process_pt)->lc = 0;; return 3; };
}
```

Protothreads

Très léger en RAM

Continuation locale

Evite des machines à états

Augmentation de ROM

Programmation par évènement

Program	Code size, before (bytes)	Code size, after (bytes)	Increase
XNP	931	1051	13%
TinyDB DBBufferC	2361	2663	13%
Mantis CC1000	994	1170	18%
SOS CC1000	1912	2165	13%
Contiki TR1001	823	836	2%
uIP SMTP	1106	1901	72%
Contiki code prop.	1848	1426	-23%

uIP et lwIP

Très léger (5ko pour uIP)

Répond aux RFC minimum

Utilise des protothreads

Base de Contiki

Feature	uIP	lwIP
IP and TCP checksums	x	x
IP fragment reassembly	x	x
IP options		
Multiple interfaces		x
UDP		x
Multiple TCP connections	x	x
TCP options	x	x
Variable TCP MSS	x	x
RTT estimation	x	x
TCP flow control	x	x
Sliding TCP window		x
TCP congestion control		x
Out-of-sequence TCP data		x
TCP urgent data	x	x
Data buffered for rexmit		x

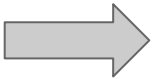
RPL

Liste de toutes les références

Pas de référence direct aux protothread

Protothread caché

Utilise des protothreads

 Protothread à garder

4.
Travail restant et planning

Présentation de l’antenne parabolique

Travail restant

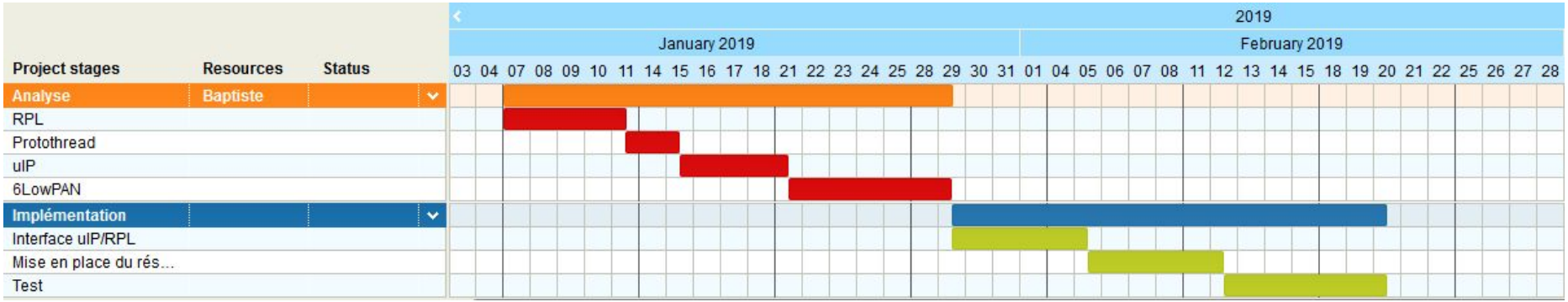
Finir de défricher RPL

Liaison uIP et RPL

6LowPAN

Assemblage et tests

Planning


2019
January 2019
February 2019
Project stages
Resources
Status
03 04 07 08 09 10 11 14 15 16 17 18 21 22 23 24 25 28 29 30 31 01 04 05 06 07 08 11 12 13 14 15 18 19 20 21 22 25 26 27 28
Analyse
Baptiste
RPL
Protothread
uIP
6LowPAN
Implémentation
Interface uIP/RPL
Mise en place du rés...
Test

Merci de votre attention

Séance de réponse aux questions