Mote-mote rádió kommunikáció

Előadó: Kincses Zoltán

message_t

```
typedef nx_struct message_t {
  nx_uint8_t header[sizeof(message_header_t)];
  nx_uint8_t data[TOSH_DATA_LENGTH];
  nx_uint8_t footer[sizeof(message_footer_t)];
  nx_uint8_t metadata[sizeof(message_metadata_t)];
} message_t;
```

A szükséges interface-ek

- Boot
  - booted()
- Leds
  - get(), set(), led0On(), led1On(), led2On(), led0Off(), ...
- Timer<precision_tag>
  - startPeriodic(), startOneShot(), fired(), ...
- Packet
  - clear(), getPayload(), payloadLength(), ...
- AMSend
  - send(), sendDone(), cancel(), getPayload(), ...
- Receive
  - receive()
- SplitControl
  - start(), stop(), startDone(), stopDone()

Az interface-eket biztosító komponensek

- MainC
  - Boot interface
- LedsC
  - Leds interface
- TimerMilliC()
  - Timer<TMilli> interface
- ActiveMessageC
  - SplitControl interface
- AMSenderC(am_id_t AMId)
  - AMSend, Packet   interface-ek
- AMReceiverC(am_id_t AMId)
  - Receive interface

Üzenetküldés rádión keresztül

```
#ifndef BLINKTORADIO_H
#define BLINKTORADIO_H

#endif
```

```
#include <Timer.h>
#include "BlinkToRadio.h"

configuration BlinkToRadioAppC {
}
implementation {
    components MainC;
    components LedsC;
    components BlinkToRadioC as App;
    components new TimerMilliC() as Timer0;

    App.Boot -> MainC;
    App.Leds -> LedsC;
    App.Timer0 -> Timer0;
}
```

```
#include <Timer.h>
#include "BlinkToRadio.h"

module BlinkToRadioC {
    uses interface Boot;
    uses interface Leds;
    uses interface Timer<TMilli> as Timer0;
}
implementation {
}
```

```
COMPONENT=BlinkToRadioAppC
CFLAGS = -DCC2420_DEF_CHANNEL=1
include $(MAKERULES)
```

```
#include <Timer.h>
#include "BlinkToRadio.h"

module BlinkToRadioC {
    uses interface Boot;
    uses interface Leds;
    uses interface Timer<TMilli> as Timer0;
}
implementation {

    uint16_t counter;

    event void Boot.booted() {
        call Timer0.startPeriodic(TIMER_PERIOD_MILLI);
    }

    event void Timer0.fired() {
        counter++;
        call Leds.set(counter);
    }
}
```

```
#ifndef BLINKTORADIO_H
#define BLINKTORADIO_H

enum{
    TIMER_PERIOD_MILLI=250,
};

#endif
```

Üzenet struktúra megadása

```
typedef nx_struct BlinkToRadioMsg {
  nx_uint16_t counter;
} BlinkToRadioMsg;
```

Üzenet elküldése

```
#include <Timer.h>
#include "BlinkToRadio.h"

module BlinkToRadioC {
    uses interface Boot;
    uses interface Leds;
    uses interface Timer<TMilli> as Timer0;
    uses interface Packet;
    uses interface AMSend;
    uses interface SplitControl as AMControl;
}
implementation {

    uint16_t counter;

    event void Boot.booted() {
        call Timer0.startPeriodic(TIMER_PERIOD_MILLI);
    }

    event void Timer0.fired() {
        counter++;
        call Leds.set(counter);
    }
}
```

```
implementation {

    uint16_t counter;
    bool busy = FALSE;
    message_t pkt;

    event void Boot.booted() {
        call AMControl.start();
    }

    event void AMControl.startDone(error_t err) {
        if (err == SUCCESS) {
            call Timer0.startPeriodic(TIMER_PERIOD_MILLI);
        }
        else {
            call AMControl.start();
        }
    }
```

```
event void Timer0.fired() {
    counter++;
    if (!busy) {
        BlinkToRadioMsg* btrpkt = (BlinkToRadioMsg*)(call Packet.getPayload(&pkt, sizeof(BlinkToRadioMsg)));
        btrpkt->counter = counter;
        if (call AMSend.send(AM_BROADCAST_ADDR, &pkt, sizeof(BlinkToRadioMsg)) == SUCCESS) {
            busy = TRUE;
        }
    }
}

event void AMSend.sendDone(message_t* msg, error_t error) {
    if (error == SUCCESS) {
        busy = FALSE;
    }
}

event void AMControl.stopDone(error_t err) {}
```

```
#include <Timer.h>
#include "BlinkToRadio.h"

configuration BlinkToRadioAppC {
}
implementation {
    components MainC;
    components LedsC;
    components BlinkToRadioC as App;
    components new TimerMilliC() as Timer0;
    components ActiveMessageC;
    components new AMSenderC(AM_BLINKTORADIO);

    App.Boot -> MainC;
    App.Leds -> LedsC;
    App.Timer0 -> Timer0;
    App.Packet -> AMSenderC;
    App.AMSend -> AMSenderC;
    App.AMControl -> ActiveMessageC;

}
```

```
#ifndef BLINKTORADIO_H
#define BLINKTORADIO_H

enum{
    AM_BLINKTORADIO = 6,
    TIMER_PERIOD_MILLI=250,
};

typedef nx_struct BlinkToRadioMsg {
  nx_uint16_t counter;
} BlinkToRadioMsg;

#endif
```

Üzenet fogadása

```
#include <Timer.h>
#include "BlinkToRadio.h"

module BlinkToRadioC {
    uses interface Boot;
    uses interface Leds;
    uses interface Timer<TMilli> as Timer0;
    uses interface Packet;
    uses interface AMSend;
    uses interface SplitControl as AMControl;
    uses interface Receive;

}
implementation {

    uint16_t counter;
    bool busy = FALSE;
    message_t pkt;

    event void Boot.booted() {
```

```
event void AMSend.sendDone(message_t* msg, error_t error) {
    if (error == SUCCESS) {
        busy = FALSE;
    }
}

event message_t* Receive.receive(message_t* msg, void* payload, uint8_t len) {
    if (len == sizeof(BlinkToRadioMsg)) {
        BlinkToRadioMsg* btrpkt = (BlinkToRadioMsg*)payload;
        call Leds.set(btrpkt->counter);
    }
    return msg;
}


event void AMControl.stopDone(error_t err) {}
```

```
#include <Timer.h>
#include "BlinkToRadio.h"

configuration BlinkToRadioAppC {
}
implementation {
    components MainC;
    components LedsC;
    components BlinkToRadioC as App;
    components new TimerMilliC() as Timer0;
    components ActiveMessageC;
    components new AMSenderC(AM_BLINKTORADIO);
    components new AMReceiverC(AM_BLINKTORADIO);

    App.Boot -> MainC;
    App.Leds -> LedsC;
    App.Timer0 -> Timer0;
    App.Packet -> AMSenderC;
    App.AMSend -> AMSenderC;
    App.AMControl -> ActiveMessageC;
    App.Receive -> AMReceiverC;

}
```

BlinkToRadioAppC