Report 1st exercise

Vincenzo Santomarco 914688 | Roberto Rocco 920721 | Arianna Ricci 921275

GitHub repo: `https://github.com/Robyroc/IOT_Homework`

The solution we proposed for this exercise is a simple extension of the **RadioCountsToLeds** code seen in class. The underlying structure of the code is exactly the same: we used similar interfaces and events, the only changes are the logic and the behavior that changes with the ID.

We implemented in the **Receive.Receive()** event handler the led control logic as it has been described in the assignment:

```
event message_t* Receive.receive(message_t* bufPtr, void* payload, uint8_t len) {
    counter++;
  if (len != sizeof(radio_id_msg_t))
    return bufPtr;
  else {
      radio_id_msg_t* rm = (radio_id_msg_t*)payload;
      if (rm->counter % 10 == 0){
        call Leds.led0Off();
        call Leds.led1Off();
        call Leds.led2Off();
      }
      switch(rm->sender_id){
        case 1:
          call Leds.led0Toggle();
          break;
        case 2:
          call Leds.led1Toggle();
          break;
        case 3:
          call Leds.led2Toggle();
          break;
      }
    return bufPtr;
  }
}
```

To solve the different behaviour problem we took advantage of the **TOS_NODE_ID** macro, which gave to the three different motes three different ids (1, 2 and 3 respectively). We used this information to assign different frequencies to the timers and to "sign" the outgoing packets:

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

event void AMControl.startDone(error_t err) {
  if (err == SUCCESS){
    uint16_t interval=0;
    id = TOS_NODE_ID;
    switch(id){
      case 1:
        interval=1000;
        break;
      case 2:
        interval=333;
        break;
      default:
        interval=200;
        break;
    }
      call MilliTimer.startPeriodic(interval);
  }
  else
      call AMControl.start();
}
```