All code is written inside the modules

Timer 1 sec = 1024 ms

Call command : call RadioControl.start();

```
Led0Timer.startOneShot(TIMER_LEDS_MILLI);
Fires one time after TIMER_LEDS_MILLI eg. 512ms == 0.5 sec
```

Commands and event handlers are not async and cannot be interrupted
So they must be kept sort. If something comes while any of those is runniing , it is lost

Async command cant call sync command

Tasks are async and can be interrupted

```
RoutingMsgTimer.startPeriodic(TIMER_PERIOD_MILLI);
TIMER_PERIOD_MILLI = curr_time +- Dt
Eg. TIMER_PERIOD_MILLI=60*1024 , it will fire in 60 sec
```

Only routing message at the beginning is oneshot, everything else is periodic

Open and close radio with SplitControl.start()/stop()

```
mrpkt = (RoutingMsg*) (call RoutingPacket.getPayload(&tmp, sizeof(RoutingMsg)));
```

Creates pointer to allocated struct of the message we want to send
Message is inside tmp

```
message_t tmp;
```

```
mrpkt->senderID=TOS_NODE_ID;
        mrpkt->depth = curdepth;
```

```
call RoutingAMPacket.setDestination(&tmp, AM_BROADCAST_ADDR);
        call RoutingPacket.setPayloadLength(&tmp, sizeof(RoutingMsg));

        enqueueDone=call RoutingSendQueue.enqueue(tmp);
```

2 Types of messages : notifyParentMsg and routingMsg

notifyParentMsg

```
m->senderID=mr->senderID;
m->depth = mr->depth;
m->parentID = mr->parentID;
```

routingMsg

```
mrpkt->senderID=TOS_NODE_ID;
mrpkt->depth = curdepth;
```

```
call RoutingMsgTimer.startOneShot(TIMER_FAST_PERIOD);
```

`TIMER_FAST_PERIOD` should probably be changed to a larger value so that all nodes have bootef

Probably wont need the serial

```
HOW MANY NODES BOOT MUST BE PASSED AS A PARAMETER
```

In the simulation

```
for i in range(0,10):
    m=t.getNode(i)
    m.bootAtTime(10*t.ticksPerSecond() + i)
```

```
Meybe topology file as 2nd parameter

Optionally have simulation time as 3rd parameter
```

May need to create a separate struct for the info we want to send
Eg. The measurement and the number of nodes

To accommodate for this data sending we are gonna need task like

```
task void sendRoutingTask()
```

5 different structs = 10 ActiveMessage id

Routing message is for tree structure building
In TAG it is used only at the beginning

Good strategy for sending packets is after sending wait for sendDone and a little bit more

The Serial Part of the code can be completely removed

What do you mean by “Το συγκεκριμένο πρόγραμμα ΔΕΝ απαιτεί συγχρονισμό των κόμβων με βάση το TAG και δεν απαιτεί να ανοιγοκλείνετε τον ασύρματο.”
Answer: in essence we are only going to need a timer for when to send info, the query above means that we don’t open and close the radio

We are gonna need go get how long the simulation has been runnin for some part
It is already implemented.

The send timing for every node is gonna be implemented not with one shot but with repeating

For the second part: if a node loses father, it will be nice to keep a list of potential fathers at the same depth as the original.

Workflow:
First set timer for when to send info upwards
Then implement the random name part