To recapitulate it should be said that we have three files called "RadioCountToLeds.h", "RadioCountToLedsC.nc" and "RadioCountToLedsappc.nc"

In the header file we define the structure of the message and the message is composed by a SenderId and a counter. We have three sensors.. These 3 sensors broadcast messages and in broadcast message there is count. When a sensor receive this message it turns LEDs on and off according to counters that we send. So every time a timer fires it sends a message containing this counter and sensor ID which is declared in “.h” file.

In “filec.nc” we include 3 timers with 3 different times, (we convert hz to Milisec) and the other related interfaces including LED, boot, timer, receiving, sending, starting the radio and managing packets. Then we define the variables in implementation part and the first one is for the packet that we want to send. The important part is the counter which is an unsigned integer with a discrete size( E.g. 16 bytes) that is defined for the packet which we want to send. Then we start the radio and when the radio is started we go in Amcontrol.startdone and we take as parameter (error\_t err) which means if the error is successed we can start the timers, otherwise we try to restart the radio( however every time we start the radio it automatically goes in success part) . Then we have event for stopping the radio in case we don’t need it and we have timers. every time the timers are fired we go inside the fired function and the function starts to increase the counter and then define the message in RCM which here we create the message with function call packet with size of the message that we define in “.h” and we create RCM as the message. If the message is created correctly it returns, otherwise it goes and assign to the counter of RCM message the value counter that we increase before and define the SenderId relating to the timer. Then we send message as broadcast message which is AM broadcast address in Tinyos. So with function I Am send we send message with broadcast to everyone. Then we also have an interface for receiving the message so we have an event for it because when the message arrive to the mote it raise an event and this event is the event for receiving it. Then we want to read the message and for reading it we should check if the size of the message is equal to the Len( the size that we received) so that we can understand we are reading a message of type radio\_count\_msg\_t. we can take the payload of message and put it in the RCM variable. Then we check if the counter mod 10 is equal to zero we turn off all the LEDs, otherwise we turn LEDs on and off according to the SenderId that we are sending. So for this approach we do a bitmask operation and convert the integer number to its binary representation and with & symbol we do bitmask. Then we check what is the result of the counter of message with bitmask. If the result is true for instance it turns on the LED0 and if not it turns it off and we do the same operation for LED1 and LED2. Therefore we receive the message with turning on and off the LEDs.