CÓDIGO DE AGUS- PROTOCOLO P2P

1) Instalar MPLABX, el compilador XC32 y las librerías

2) Abrir MPLABX, File->**Open proyect** y seleccionarlo de donde lo hayamos guardado.

3) Para que reconozca las **librerías** ya que editar el proyecto:
   Clicar con el botón dch e ir a propiedades.
   En la *sección XC32(Global Options) -> xc32 –gcc*, cambiar la categoría a *Preprocessing and messages*. En *include directories* los path en el que debe buscar en cada include, es decir, añadir el fichero Include de Node FW y el Include de libs.
   CWSN LSI Node\Include
   ..\libs\Microchip\Include

4) PROBLEMAS:
   a. _DISABLE_OPENADC10_CONFIGPORT_WARNING
      Simplemente en los nuevos micros OpenADC10 funciona con otros canales, en los antiguos tenían la parte analógica en el puerto B. Aparece en la librería del compilador.

   b. SUPRESSPLIB_WARNING
      En su versión se usaba una librería más antigua (plib). *Add "_SUPPRESS_PLIB_WARNING" to the preprocessor definitions in the project settings to get rid of it (though now that PLIB has been removed from the XC32 distribution, the warning really ought to have been removed as well).*

   Project properties -> xc32-gcc -> option preprocessing and messages -> Preprocesor macros -> add
   "_SUPPRESS_PLIB_WARNING";"_DISABLE_OPENADC10_CONFIGPORT_WARNING"

   c. Interrupt priority IPLx is deprecated. Specify as 'IPL1{AUTO|SOFT|SRS}' instead. [enabled by default]
   Sustituir en NodeHAL.c y en los transceptores (MRF24J40.c y MRF49XA.c) la definición de las interrupciones iplx por **iplxAUTO**.

5) Cuando se conecta un nuevo nodo, al hacer el escáner y detectar ya una conexión establecida intenta unirse. Para ello le va a enviar un comando CMD_P2P_CONNECTION_REQUEST en un paquete.

```
//Juan: Looking for a peer. Sending a request with my P2P info
//attached...
MiApp_FlushTx(mb);
MiApp_WriteData(CMD_P2P_CONNECTION_REQUEST, mb);
MiApp_WriteData(*currentChannel, mb);
MiApp_WriteData(P2PCapacityInfo->Val, mb);

#if ADDITIONAL_NODE_ID_SIZE > 0
    BYTE i;
    for(i = 0; i < ADDITIONAL_NODE_ID_SIZE; i++){
        MiApp_WriteData(AdditionalNodeID[i], mb);
    }
#endif

#if defined(IEEE_802_15_4)
    #if defined(ENABLE_ACTIVE_SCAN)
        //Juan: If ActiveScanIndex is 0xFF, the protocol stack
        //will try to establish a connection with any device.
        if(ActiveScanIndex == 0xFF){
            SendPacket(TRUE, myPANID, NULL, TRUE, FALSE, mb);
        }
        else{
            //Juan: Joining the network found whose info is
            //stored in ActiveScanIndex slot.
            resultado = MiApp_SetChannel(ActiveScanResults[ActiveScanIndex].Channel, mb);
            if (resultado) Printf("\r\n\n Cambio de canal correcto");
            else Printf("\r\n\n Cambio de canal erróneo");
            resultado = SendPacket(FALSE, ActiveScanResults[ActiveScanIndex].PANID,
                    ActiveScanResults[ActiveScanIndex].Address, TRUE, FALSE, mb);
            if (resultado) Printf("\r\n\n Envío correcto");
            else Printf("\r\n\n Envío erróneo");
        }
    #else
        SendPacket(TRUE, myPANID, NULL, TRUE, FALSE, mb);
    #endif
    //Juan: request has been sent.
```

En problema es que cuando le llega la petición, si tiene activado **ENABLE_SLEEP**, independiente de si está dormido o no ignora el paquete y no responde.

```
if (MIWI_rxMsg->flags.bits.command) {
    // if comes here, we know it is a command frame
    switch(MIWI_rxMsg->Payload[0] ) {
        #if defined(ENABLE_HAND_SHAKE)
            case CMD_P2P_CONNECTION_REQUEST:
                {
                    BYTE status = STATUS_SUCCESS;

                    // if a device goes to sleep, it can only have one
                    // connection, as the result, it cannot accept new
                    // connection request
                    #ifdef ENABLE_SLEEP
                        AuxMAC_DiscardPacket(transceiver);
                        break;
```

Si no definimos dicha variable el proceso de respuesta es correcto. El transmisor lo acepta y le envía un mensaje de vuelta.

```
// request accepted, try to add the requesting
// device into P2P Connection Entry
status = AddConnection(mb);
```

```
if((status == STATUS_SUCCESS || status == STATUS_EXISTS) &&
    MiApp_CB_AllowConnection(LatestConnection) == FALSE ) {

    //Juan: this if() seems absurd to me, as in P2P
    //is intended to check the connection table in
    //cases SUCCESS and EXISTS with MiAppCB function
    //and it is defined as always TRUE in a macro...
    ConnectionTable[LatestConnection].status.Val = 0;
    status = STATUS_NOT_PERMITTED;
}

// prepare the P2P_CONNECTION_RESPONSE command
MiApp_FlushTx(mb);
MiApp_WriteData(CMD_P2P_CONNECTION_RESPONSE, mb);
MiApp_WriteData(status, mb);
//if(status == STATUS_SUCCESS || status == STATUS_EXISTS){ //Juan: modified as below.
if(status == STATUS_SUCCESS || status == STATUS_EXISTS || status == STATUS_UPDATED){
    MiApp_WriteData(P2PCapacityInfo->Val, mb);
    #if ADDITIONAL_NODE_ID_SIZE > 0
        for(i = 0; i < ADDITIONAL_NODE_ID_SIZE; i++){
            MiApp_WriteData(AdditionalNodeID[i], mb);
        }
    #endif
}

AuxMAC_DiscardPacket(transceiver);

// unicast the response to the requesting device
```

Con estas soluciones se ha conseguido que los dos nodos se conecten al mismo canal en cada una de las interfaces usando el **protocolo P2P**. Además se ha comprobado que tanto envío como transmisión de paquetes es correcto.