Comunicació radio utilitzant els mòduls nRF24L01
Xarxes de Comunicacions

Guillem Prenafeta

Semestre Tardor 2023-24

1 Introducció

En aquest treball s'estudiarà l'aplicació d'una comunicació radio entre dos mòduls **nRF24L01** de la casa Nordic. Aquests mòduls es poden posar en qualsevol disseny ja que només necessiten alimentació, una comunicació SPI, i un GPIO.

Encara que aquests mòduls poden utilitzar-se com a simples transmissors o receptors, fent així una comunicació en una sola direcció, s'estudiarà el cas on es vol una comunicació full duplex, és a dir, amb dos direccions de dades. Fàcilment es pot extrapolar a una comunicació de una sola direcció de dades amb acknoledge per part del receptor (si aquest no envia de dades de tornada).

Tot el treball de *Software*[1] que conté la programació dels dispositius es pot trobar en un repositori obert de GITHUB[2]. Aquest repositori conté les llibreries en C per controlar al mòdul, on l'usuari només ha de definir una funció per enviar dades per un 3-wire SPI, dos GPIOs de sortida i un GPIO d'entrada amb interrupcions actives per canvi de nivell alt a baix. També es mostren alguns exemples per poder aplicar bé les llibreries.

2 Marc teòric

El mòdul nRF24L01 és un transceiver de la banda de 2.4 GHz, és a dir, la banda de l'espectre pertany a les ones ràdio. És capaç de transmetre a 250 $kbps$, 1 $Mbps$ i 2 $Mbps$. És un dispositiu de baixa potència, per tant és bastant ideal per aplicacions IoT. És capaç de gestionar 6 canals de dades diferents (6 data pipeline) i té una tecnologia desenvolupada pel fabricant anomenada *Enhanced Shockburst* que no és més que un protocol gestionada pel propi transceiver que s'encarrega principalment de la gestió d'errors, acknowledgement, mides de paquets de dades variables i a més a més està dissenyat per consumir poc.

El mòdul es un simple xip molt petit de 4x4 mm encara que se li ha d'afegir una alimentació que pot variar entre 1.9 a 3.6V ja que internament ja té un regulador. I també necessita un cristall de 16 MHz, i es clar necessita d'una entena per transmetre els senyals que pot estar impresa directament a la PCB. Si es vol arribar més lluny amb la comunicació hi ha mòduls PCB que contenen l'antena més algun LNA per augmentar la potència d'emissió/recepció.

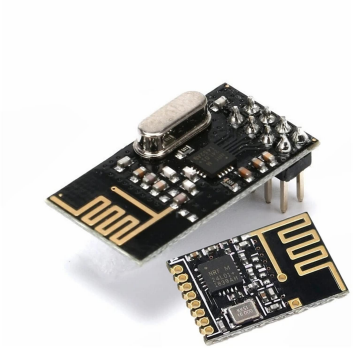
Les proves s'han fet amb un mòdul nRF24L01 que ja venia integrat en una PCB que contenia l'antena i el cristall. Lo recomanable es fer servir aquests mòduls ja que són relativament petits, econòmics i fàcils de trobar.

Aquest mòduls pot servir per un gran nombre d'aplicacions com controls remots de RF, RFID, joguines, Ultra low power sensor networks, sensors entre d'altres.

[1] El software està escrit en C i s'ha testejat i verificat amb els microcontroladors de texas instruments fent un enllaç radio d'uns metres

[2] El repositori es diu **nRF24L01** i l'usuari és **guillemmmm** (https://github.com/guillemmmm/nRF24L01)

(a) Encapsulat chip nRF24L01 (4x4mm)



(b) Mòdul PCB nRF24L01 amb antena integrada (proves fetes amb aquest mòdul)

(c) Mòdul PCB nRF24L01 amb LNA per augmentar el rang i amb connexió coaxial per connectar una antena

Figura 1: Mòdul nRF24L01

3 Comunicació estàndard TX/RX

Per establir una comunicació simple punt a punt es necessiten mínim de dos mòduls nRF24L01, un que actua com a transmissor i l'altre com a receptor.

Primer de tot s'ha d'establir el canal de freqüència, encara que la banda de comunicació es la de 2.4 GHz realment el rang de freqüències pot variar de $f \in [2.400\ GHz,\ 2.525\ GHz]$, per tant s'ha de programar un registre del mòdul que escull el canal, el RF_CH, que va des de 0 a 127.

$$F_0 = 2400 + RF\_CH\ \ [MHz]$$

.

En aquest mode més simple de comunicació el transmissor envia un paquet de dades al aire i el receptor el rep. El protocol de transmissió se l'haurà de muntar l'usuari afegint camps de CRC, confirmació de que s'ha rebut, etc. Això comporta bastant software, per això el fabricant ja ens dona un protocol que fa tot això fent servir els recursos del propi nRF24, el *Enhanced Shockburst*.

Només afegir que per enviar el paquet de dades, el microcontrolador que vols transmetre ha d'escriure el paquet de dades a la FIFO de transmissió del nRF24 i activar la transmissió posant el chip enable (CE) a 1 durant almenys $10\mu s$. D'altre banda el microcontrolador que vol rebre ha hagut de posar el seu nRF24 en mode recepció i quan arribi el paquet, el nRF24 commutarà el GPIO de recepció (si s'han activat les interrupcions del mòdul) i les dades rebudes s'enmagetzemaràn a la FIFO de recepció. Es pot trobar un programa d'exemple al GITHUB, dins la carpeta d'exemples, a la carpeta de simple transmission.

4 Protocol Comunicació Duplex

Com ja s'ha mencionat el fabricant ens proporciona un protocol full duplex que ja be implementat en els xips nRF24. Aquest protocol s'anomena *Enhanced Shockburst* i permet tenir gran llibertat en la nostra comunicació. Els seus principals aventatges són:

- **Mida de missatge variable** El paquet de dades té un camp on indica la mida del missatge. Per tant els nostres missatges poden tenir mides diferents permetent-nos enviar diferents missatges així aprofitant energia i temps al no haver d'enviar una longitud fixa.
- **Adreça** La adreça permet dirigir-se a diferents receptors permetent fer xarxes més extenses de comunicacions.

- **Acknowledge** També ofereix la possibilitat del acknoledge. És a dir, el transmissor espera la confirmació del receptor conforme ha rebut el missatge, si apareix qualsevol error serà capaç de retransmetre el missatge N cop tal que ho haguem definit, i si hi ha més errors saltarà una interrupció indicant-ho. Això ens facilita molt la comunicació i li dona certa seguretat.
- **Control d'errors** El paquet també conte de 1 a 2 bytes de CRC evitant que el missatge contingui errors.

L'estructura del paquet de dades és el següent:

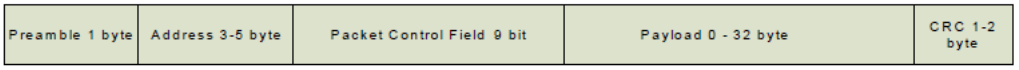


Figura 2: Estructura paquet de dades

On el Preamble es una cadena de 0's i 1's (0x55) per sincronitzar el receptor. Després hi ha l'adreça per poder enviar diferents missatges a diferents receptors, es pot aconseguir un gran rang d'adreces. A continuació tenim el packet control field, que conté la mida del missatge (de 0 a 32 bytes), el PID (identificador del paquet) i finalment el no-ACK que indica si el receptor ha de contestar amb un acknowledge. A continuació es troba el missatge en si i finalment el CRC. Si el CRC és incorrecte el receptor no acceptarà el paquet.

El camp de PID (identificador de paquet), que potser no és molt obvi és interessant a l'hora de descartar paquets. Ja que si el paquet s'envia més d'un cop ha d'haver-hi certa manera de saber si el paquet rebut es nou. Per tant el PID indica si un missatge és nou. A continuació es presenta el diagrama de flux tant del transmissor com del receptor de com gestionen aquest PID i que te a veure amb el paquet rebut.

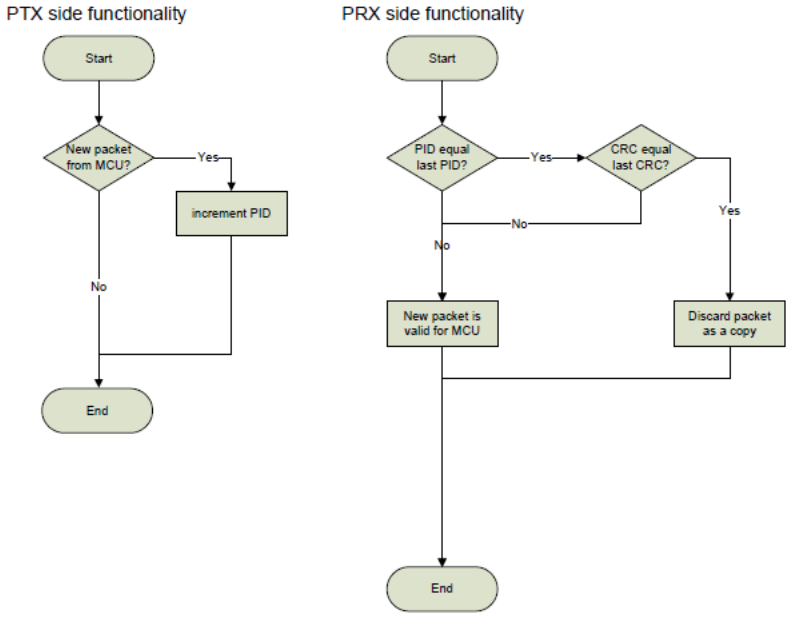


Figura 3: Diagrama de flux PID on PTX correspon al transmissor i PRX al receptor

Una altre característica interessant però que no s'ha provat és el *MultiCeiver*. Aquest mode permet que el transmissor parli amb 6 *esclaus* a la vegada sense haver d'anar canviant l'adreça manualment. Pot ser molt útil per fer xarxes de comunicació, com per exemple per una xarxa de sensor IoT.

Finalment s'explicarà de manera gràfica el protocol on per exemple que passa quan falla un acknowledge o quan es perd un paquet. Primer s'explicarà una transacció on tot va correctament, després una on es perd

el paquet, una altre on falla el acknowledge i finalment una en la que s'arriba al màxim de retransmissions acabant amb una comunicació fallida.

4.1 Transacció Normal

Primer de tot explicarem la comunicació normal, on el transmissor inicialment carrega el paquet a la FIFO de transmissió (UL1). Important destacar que el receptor sempre està en mode receptor amb el CE a 1, Abans de posar-se en receptor, el receptor ha carregat un paquet de resposta a la FIFO de acknowledge payload, on sinò s'han carregat dades només s'enviara de resposta un acknowledge.

Un cop s'envia el paquet, el transmissor automàticament es posarà en mode receptor, tardant en canviar de mode un temps màxim de 130 μs, nosaltres com a MCU externa no hem hagut de fer res. Automàticament, el receptor es posarà com a transmissor i enviarà el acknowledge amb el paquet de dades de resposta. Si el CRC rebut es correcte quan el receptor rebi el paquet de dades el guardarà a la FIFO de recepció (RX_DR) generant així la interrupció de recepció (DL).

Quan el transmissor rebi el paquet de acknowledge sent el CRC correcte, aquest farà saltar la interrupció de recepció (RX_DR) junt amb la de transmissió (TX_DS) indicant que s'ha enviat el paquet anterior també. En el següent paquet que torni a enviar el transmissor (quan la MCU escrigui a la TX FIFO) llavors el PID haurà augmentat en 1 indicant que és un nou paquet. I quan el receptor el rebi, farà saltar tant la interrupció de recepció (RX_DR) com la de transmissió (TX_DS) ja que vol dir que el paquet de dades enviat junt amb l'acknowledge anterior s'ha enviat correctament.

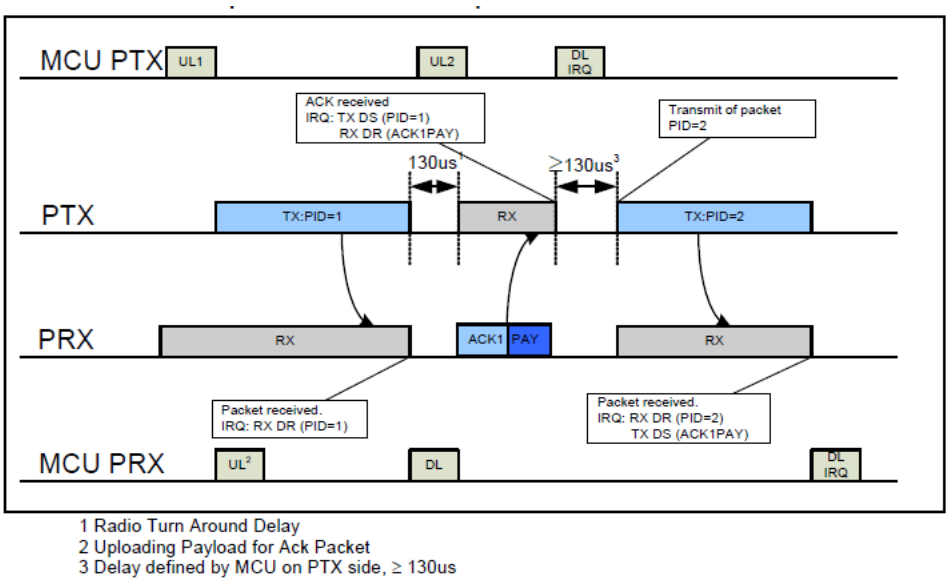


Figura 4: Transacció correcta de paquets

4.2 Paquet perdut en la transacció

En canvi, si es perd el paquet de transmissió durant la transacció, el receptor seguirà estant escoltant. El transmissor en quan envii el paquet es posarà a escoltar, però com el receptor segueix rebent, el transmissor tampoc rebrà res. Llavors el transmissor estarà esperant fins cert temps (ARD, configurable) i quan aquest es superi tonrarà a enviar el mateix paquet un altre cop. Seguirà repetint-se aquest proces fins que rebi el acknowledge del receptor o fins a superar el nombre màxim de retransmissions definit a la configuració (MAX_RT).

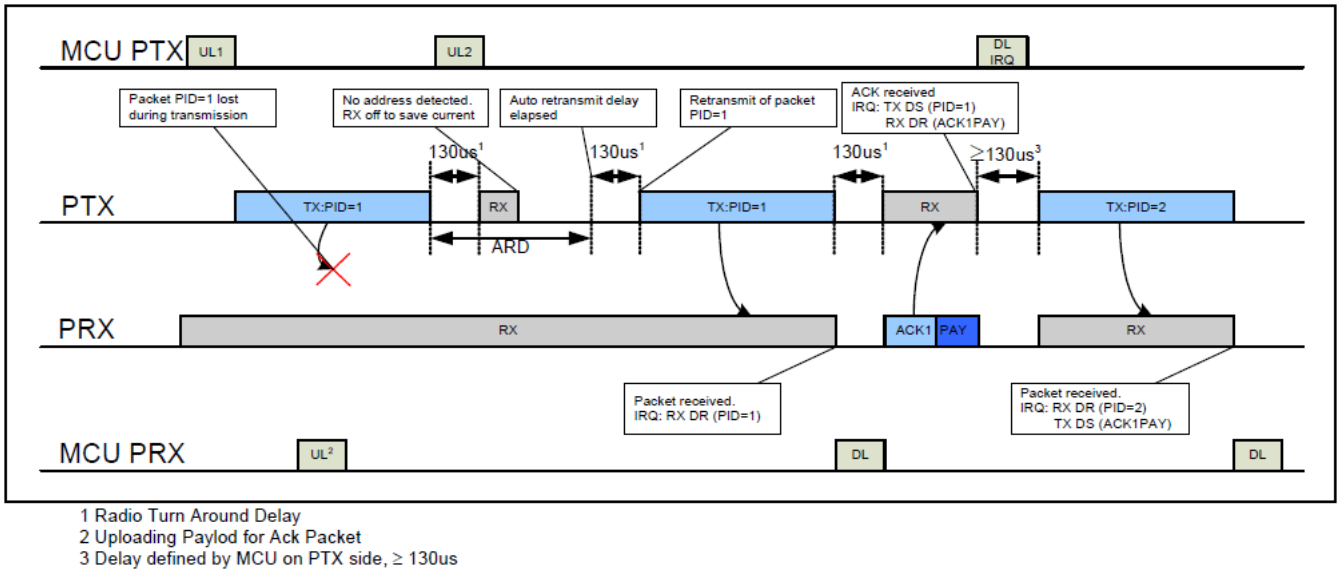


Figura 5: Transacció amb pèrdua de paquet

4.3 Acknowledge perdut en la transacció

Ara apareix un altre cas on el transmissor envia el paquet correctament però és ara el paquet de resposta + acknowledge que falla. Llavors el transmissor després de esperar el temps (ARD) o rebre el paquet incorrecte retransmetrà el mateix paquet. Aquí és quan entra en joc el PID, ja que el receptor sabent que és el mateix paquet (mateix PID que no s'ha incrementat), no guardarà el nou paquet a la FIFO de recepció però si que tonrarà a retransmetre el paquet de acknowledge + missatge. Finalment quan el transmissor rebi el acknowledge la comunicació haurà sigut efectuada correctament canviant a un nou missatge i augmentant el PID.

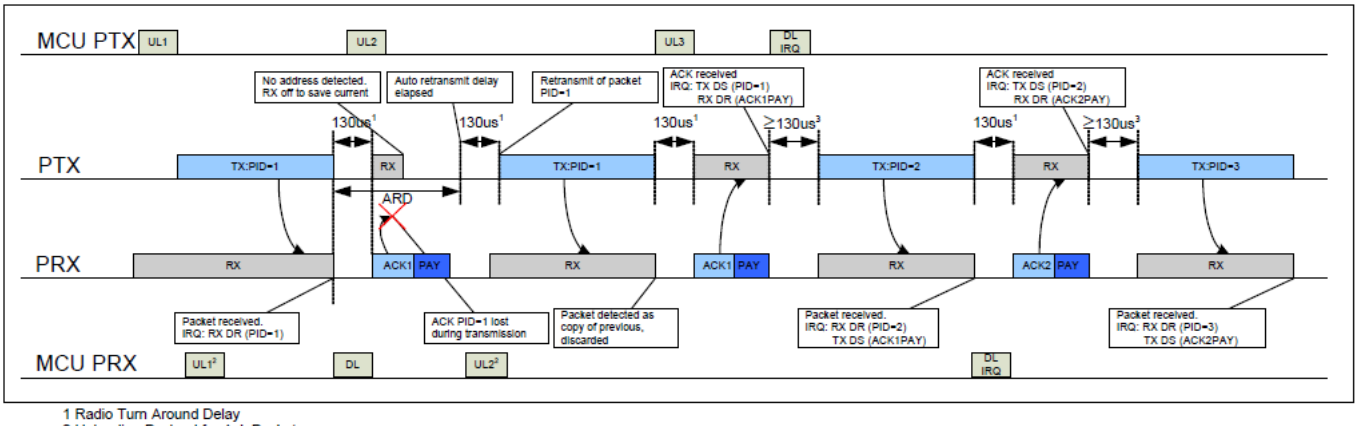
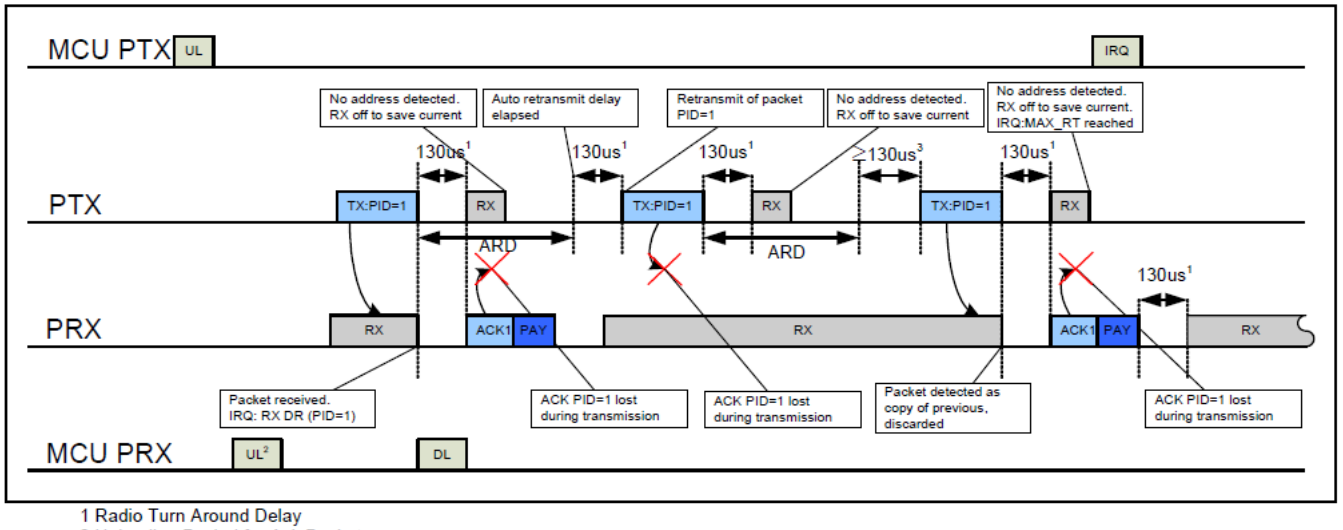


Figura 6: Transacció correcta de paquets

4.4 Nombre màxim de retransmissions assolit

Finalment, si el paquet enviat es perd continuament, o el paquet de resposta (acknowledge) també, el transmissor arribarà al nombre màxim de retransmissions (MAX_RT programat al configurar el mòdul) saltant així la interrupció de (MAX_RT). La MCU sabrà llavors que hi ha hagut un problema amb la

comunicació. És la única manera amb que pot saber que la comunicació falla i a partir d'aquó pot prendre les mesures necessàries.



1 Radio Turn Around Delay
2 Uploading Paylod for Ack Packet
3 Delay defined by MCU on PTX side, ≥ 130us

Figura 7: Transacció correcta de paquets

5 Resultats

Finalment per concloure, s'explicarà on s'ha aplicat aquesta comunicació i si ha sigut útil. Durant l'assignatura de Laboratori de Sistemes Electrònics II, s'havia de fer una mena de balancí on en cada extrem es situava un motor BLDC. L'objectiu era controlar l'angle amb un PID. Ja que és un objecte que sempre està amb moviment, accedir al seu control de manera física (amb cables) pot ser pesat. Per tant s'ha fet un control inal·làmbic mitjançant un enllaç radio.

Per tant s'ha fet servir el protocol *Enhanced Shockburst* per enviar diferents tipus de missatges. La veritat és que la comunicació ha anat perfecte, on des de un control amb joysticks depenent del mode podia enviar instruccions diferents (diferent mida de missatge) i el balancí responia amb l'estatus actual del sistema.

Gràcies a la comunicació ha pogut fer molt més còmode el control i també gràcies a la robustesa del protocol no hi ha hagut ningun error, tenint en compte que s'està fent servir una banda lliure on hi ha moltes possibles fonts d'error i soroll és bastant acceptable.

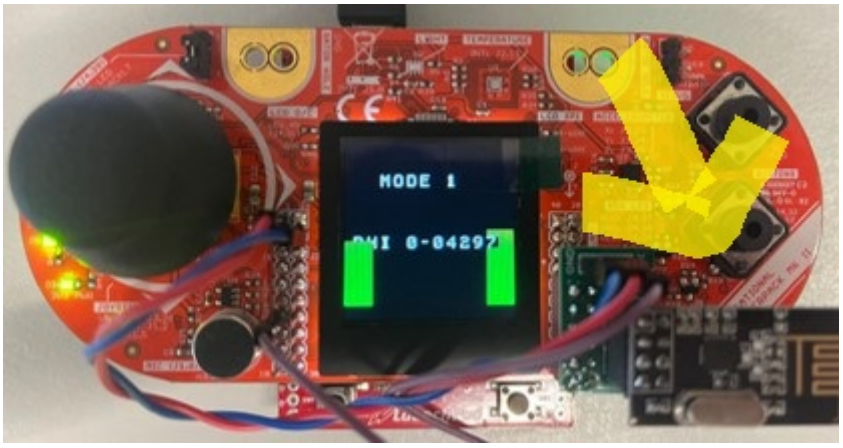


Figura 8: Mòdul nRF24L01 col·locat al comandament que controla el balancí

6 Conclusions

En aquest breu projecte, s'ha pogut establir un enllaç ràdio adequadament on també s'ha estudiat amb una mica més de profunditat el protocol de la comunicació. S'han pogut veure que el conceptes tractats a l'assignatura, des del model de capes OSI fins al tractament d'errors, són de certa manera aplicats a la comunicació i en productes funcionals actualment es fan servir.

També remarcar la importància de les comunicacions i la seva robustesa ja que ens poden facilitar molt la vida en certes aplicacions. I finalment, afegir que encara que el projecte del comandament no és troba al github, totes les llibreries de control del mòdul nRF24L01 i alguns exemples d'aplicacions es poden trobar al github escrits en C https://github.com/guillemmmm/nRF24L01/.

7 Annex. Explicació codi exemple

A continuació es presenta el codi d'exemple. On s'ha eliminat les funcions que controlen els recursos de hardware del microcontrolador per fer-ho més breu.

7.1 Exemple master

```
1
2  nRF24L01_init(); //we use pipe0 communicaiton, default adress
3  nRF24L01_powerUP(PTXmode); //TX mode
4
5  while(1)
6  {
7
8      //missatge
9      char array[] = "HOLA MUNDO"
10     nRF24L01_write_tx_payload_PTX((uint8_t*)array, sizeof(array), true); //enviem
11     while(RADIOint==false){ //esperem int
12     }
13     RADIOint=false;
14     uint8_t status = nRF24L01_nop();
15     if((status&nrf24l01_STATUS_TX_DS)==nrf24l01_STATUS_TX_DS){ //si es que s'ha enviat
       correctament
16         //llegim missatge
17         do{
18             nRF24L01_payload_width(&payloadRXLEN); //obtenim longitud del missatge ja que es
        variable
19             nRF24L01_read_rx_payload(payloadRX, payloadRXLEN); //i fem processament
20         }while(!nRF24L01_FIFO_RX_empty()); //mentres estigui plena (quan buida passa a true)
21         //ja tenim les dades
```

```
22      } else if((status&nrf24l01_STATUS_MAX_RT)==nrf24l01_STATUS_MAX_RT){//error de
        transmissio s'haura de torna a enviar
23          __nop();
24      }
25
26      nRF24L01_clear_IRQ();
27      delay100us(100); //10ms
28  }
```

Primer de tot inicialitzem el mòdul i el configurem com a transmissor (l. 2-3). Un cop ja estigui inicialitzat, creem el missatge i l'escrivim a la FIFO de transmissió, com hem posat *true*, aquest s'enviarà directament activant el CE de la radio. Un cop enviat esperem la interrupció de la radio, que pot ser de que s'ha enviat correctament o de que ha arribat al número màxim de retransmissions (error comunicació).

Si tot és correcte, llegim la longitud del missatge i extrèiem el missatge del mòdul radio. Finalment netejem interrupcions i ja hem fet tots la comunicació.

7.2 Exemple slave

```
1
2   nRF24L01_init(); //we use pipe0 communicaiton, default adress
3   nRF24L01_powerUP(PTRmode); //RX mode
4
5       //escribim resposta inicial
6   char array[] = "RESPOSTA HOLA MUNDO"
7   nRF24L01_write_tx_payload_PRX((uint8_t*)array, sizeof(array), 0); //pipe 0
8
9   while(1)
10  {
11
12      //missatge
13      nRF24L01_write_tx_payload_PTX((uint8_t*)array, sizeof(array), true); //enviem
14      while(RADIOint==false){ //esperem int
15      }
16      RADIOint=false;
17      uint8_t status = nRF24L01_nop();
18      if((status&nrf24l01_STATUS_TX_DS)==nrf24l01_STATUS_TX_DS){ //si es que s'ha enviat
        correctament
19          //llegim missatge
20          do{
21              nRF24L01_payload_width(&payloadRXLEN); //obtenim longitud del missatge ja que es
         variable
22              nRF24L01_read_rx_payload(payloadRX, payloadRXLEN); //i fem processament
23          }while(!nRF24L01_FIFO_RX_empty()); //mentres estigui plena (quan buida passa a true)
24          //ja tenim les dades
25
26          nRF24L01_write_tx_payload_PRX((uint8_t*)array, sizeof(array), 0); //pipe 0 //posem
        resposta
27      }
28      nRF24L01_clear_IRQ();
29  }
```

En aquest exemple la diferència respecte el master, és que configurem com a mode RX i posem la radio a escoltar. Això si, sempre que volguem enviar un missatge l'haurem de carregar abans de la transmissió.