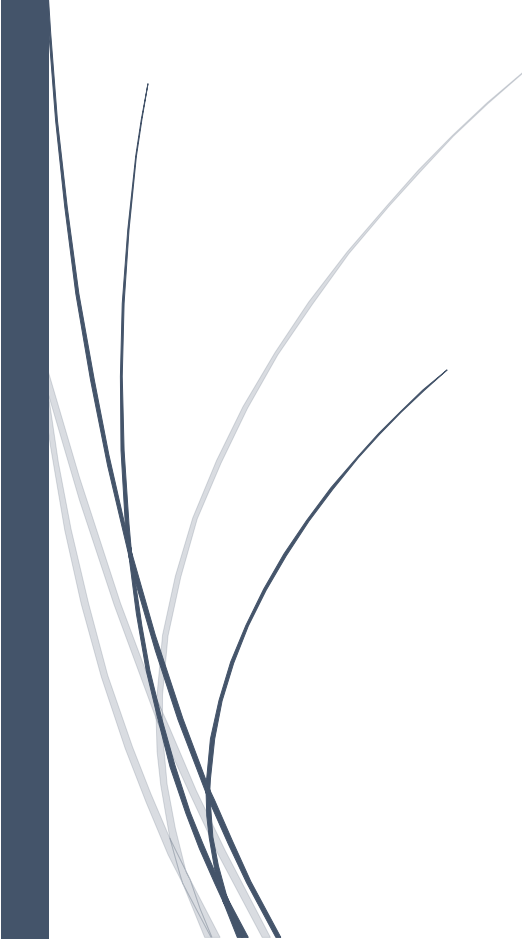


A dark blue vertical bar is on the left. A blue arrow points right from it, containing the date.

31/08/2015

Informe técnico

NRF24l01+ sobre Galileo

Several thin, curved lines in dark blue and light grey originate from the bottom left and curve upwards and to the right.

Luis Alfredo Muñoz Molina
11-10681

Índice

Objetivo General:.....	2
Objetivos Específicos:.....	2
Introducción	3
SPI (Serial Peripheral Interface)	4
SCK y DATA	4
MISO Y MOSI.....	4
Selector de esclavos (SS).....	6
NRF24L01+	7
Radio	7
Transmisor.....	7
Receptor	7
Sintetizador de RF.....	7
ShockBurst Mejorado	7
Manejo de potencia	8
Interfaz del anfitrión.....	8
Diagrama de bloques del módulo de RF	8
Modos de operación del nRF24l01+	9
Canales de RF	10
Forma del paquete de transmisión.....	10
Largo del payload, dinámico y estático.....	11
Comandos de SPI.....	12
Mapa de Registros.....	14
Galileo Gen 2	26
Primeras configuraciones Galileo Gen 2.....	27
Instalación en Linux:.....	27
Instalación en Windows	29
Conexión con el Galileo.....	30
En Linux.....	30
En Windows	31
Recomendaciones.....	32
Comunicación con nRF24l01+	34
Resultados	36
Bibliografía	37

Objetivo General:

Realizar la Configuración y Pruebas de Transmisión y Alcance del Módulo de NRF24L01+ sobre la plataforma Galileo

Objetivos Específicos:

- Estudiar Características de Modulo NRF24L01+ (1 semana)

Actividades:

- Revisión del Manual de del Módulo NRF24L0+,
- Estudiar Arquitectura interna, registro, niveles de tensión.
- Programación de parámetros del módulo.

- Estudiar Características y Software de Programación del Galileo , puerto SPI (1 semana)

Actividades:

- Revisión del Manual de Galileo, su Arquitectura interna,
- Estudio de la interfaz de comunicación SPI, niveles de tensión.

- Realizar Programación de Configuración Básica del NRF24L01+ (2 semanas)

Actividades:

- Programación y Pruebas de conexión entre el modulo y el Galileo
- Programación de driver para el modulo para el flujo de datos bidireccional
- Programación de rutinas de cambio de parámetros inalámbricos del módulo, canal de RF, tipo de Modulación,

- Realizar Pruebas de Conectividad punto a punto a Baja Velocidad 250 Kbps, Verificando ancho de banda ocupado, canal asignado, y alcance. (1 semana)

- Realizar Pruebas para Velocidades de 1Mbps y 2Mbps. (1 semana)

Introducción

La placa de desarrollo Intel Galileo Gen 2 está basada en arquitectura Intel y está diseñada para creadores, estudiantes, educadores y entusiastas de la electrónica.

La placa proporciona a los usuarios un entorno de desarrollo tanto de hardware como de software totalmente abierto.

Posee un procesador Intel Quark Soc. X1000, 32 bits, un núcleo, un hilo, compatible con la arquitectura de conjunto de instrucciones del procesador Intel Pentium, que funciona hasta velocidades de 400Mhz.

El nRF24L01+ es un transceptor de 2.4 GHz con un protocolo de banda base incorporado (Enhanced ShockBurst™), adecuado para aplicaciones de ultra baja potencia inalámbrica. Está diseñado para trabajar en la banda de frecuencia ISM internacional establecida por la ITU. Para diseñar un sistema de radio con el nRF24L01+ solo se necesita un microcontrolador y algunos componentes pasivos. Se puede operar y configurar a través del SPI (Serial Peripheral Interface).

El mapa de registros es accesible a través del SPI.

El nRF24L01+ envía la información modulada en GFSK, el usuario puede modificar parámetros como lo son el canal de frecuencia, potencia de salida y velocidad de datos en el aire. Existen tres velocidades 250 Kbps, 1 Mbps, 2 Mbps.

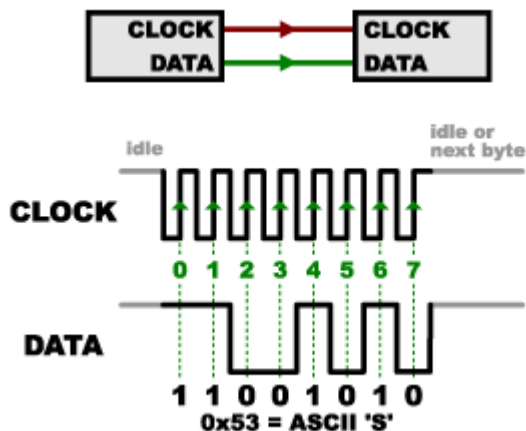
Para este proyecto se desea configurar y medir el alcance que tienen los módulos de RF sobre el Galileo Gen 2 ya mencionado, para ello se conectara el modulo al Galileo vía SPI, se creara un driver capaz de configurar los registros siguiendo el *DataSheet* que se encuentra en la página de *Nordic semiconductor* y se probara el alcance de estos módulos.

SPI (Serial Peripheral Interface)

El bus de SPI es una interfaz de comunicación serial síncrona que se utiliza para la comunicación entre los microcontroladores y pequeños periféricos, como registros, sensores, etc. Se utiliza sobre todo en sistemas embebidos. La interfaz fue desarrollada por Motorola.

SCK y DATA

SPI posee líneas separadas de datos y de reloj, con lo cual puede mantener ambos lados en perfecta sincronía. El reloj es una señal oscilante que le dice al receptor cuando muestrear los bits en la línea de datos, esto podría ser por el flanco de subida o por el flanco de bajada de la señal del reloj, la hoja de datos del fabricante especificara cual usar. Cuando el receptor detecta el flanco a utilizar, vera de inmediato la línea de datos y leerá el próximo bit.

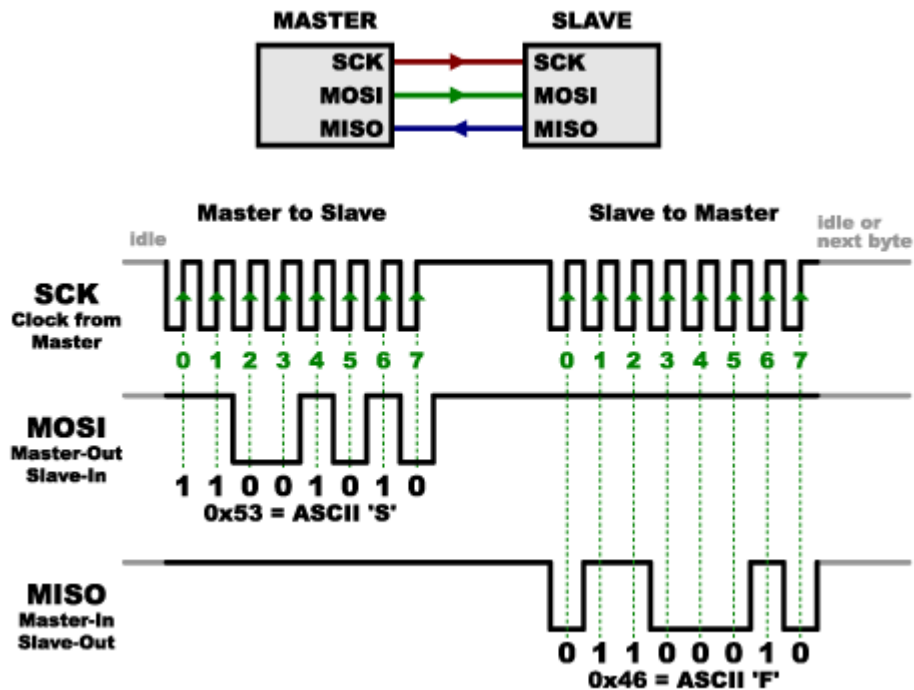


MISO Y MOSI

Hasta ahora solo hemos visto como enviar en un solo sentido, para la respuesta es un poco más complicado, en SPI, solo un lado genera la señal del reloj, (generalmente llamado CLK o SCK para *Serial Clock*) dicho lado se le llama “maestro”, y el otro lado se llama “esclavo”. Siempre hay un solo maestro (que es casi siempre el

microcontrolador), pero puede haber múltiples esclavos.

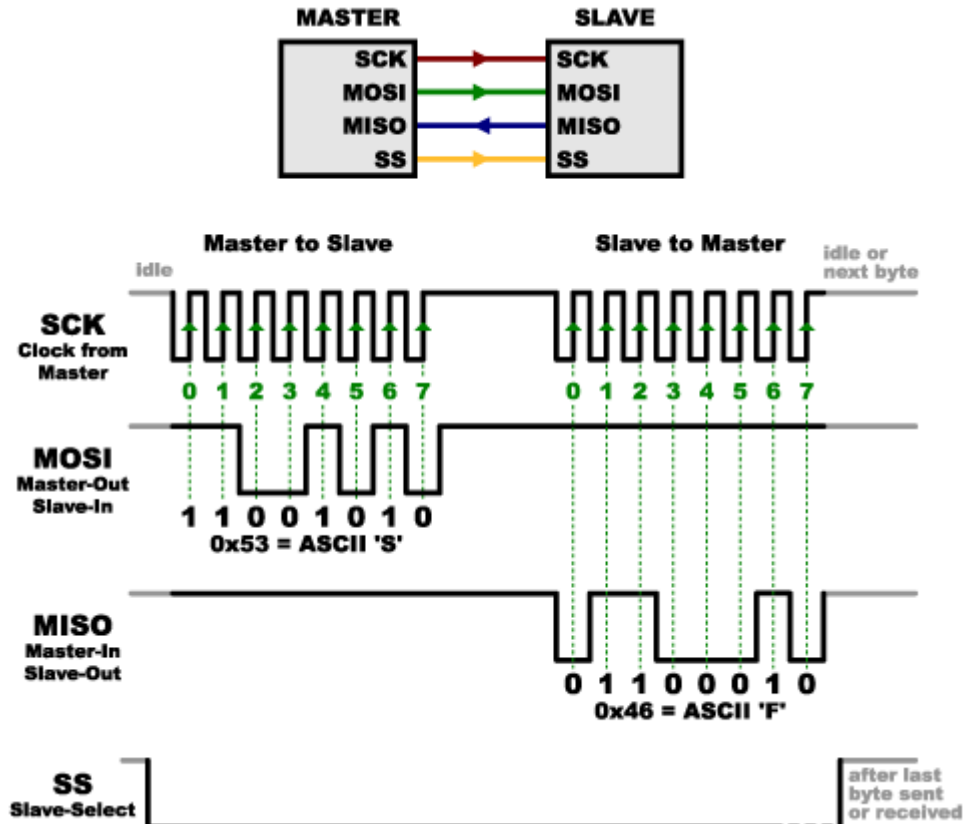
Cuando los datos se envían de “maestro” a “esclavo”, se envían por una línea de datos llamado MOSI (Master Out / Slave In). Si el esclavo necesita enviar una respuesta al maestro, el maestro seguirá generando un numero preestablecido de ciclos de reloj, y el esclavo pondrá los datos en otra línea de datos llamada MISO (Master In / Slave Out).



Debido a que el maestro siempre está generando señales de reloj, se debe saber de antemano cuando el esclavo tiene que devolver los datos y la cantidad de datos que serán devueltos. En la práctica no es un problema ya que SPI usualmente se utiliza para hablar con dispositivos que tienen una estructura de mando muy específica. Por ejemplo, si envía el comando “leer registro 7”, usted sabe que el dispositivo le va a enviar siempre, por ejemplo, 2 bytes. Hay que tener en cuenta que SPI es *full dúplex*, por lo tanto en algunas situaciones puede transmitir y recibir datos al mismo tiempo.

Selector de esclavos (SS)

Este es el último pin y se utiliza para decirle al “esclavo” cuando debe estar activo para recibir o enviar datos. También se utiliza para seleccionar un esclavo con el que se quiere hablar si se tiene múltiples esclavos.



NRF24L01+

Radio

1. Utiliza la banda de 2.4 GHz establecida por la ITU para ISM.
2. Posee velocidad de datos en el aire de 250 Kbps, 1 Mbps y 2 Mbps.
3. 126 canales de RF.
4. Interfaz común de transmisión y recepción.
5. Modulación GFSK.
6. Separación entre canales de 1 MHz no se superponen a 1 Mbps.
7. Separación entre canales de 2 MHz no se superponen a 2 Mbps

Transmisor

1. Potencia de salida programable entre 0, -6, -12 o -18 dBm
2. 11.3 mA a 0dBm de potencia de salida.

Receptor

1. Controlador automático de ganancia rápido, para mejorar el rango dinámico.
2. Filtros de canal integrados.
3. 13.5 mA a 2Mbps
4. A 2 Mbps la Sensibilidad es -82 dBm.
5. A 1 Mbps la Sensibilidad es -85 dBm.
6. A 250 Kbps la Sensibilidad es -94 dBm

Sintetizador de RF

1. Sintetizador completamente integrado.
2. 16 MHz cristal.

ShockBurst Mejorado

1. Largo del payload dinámico entre 1 y 32 bytes.
2. Manejo automático de paquetes.
3. Manejo automático de transacciones de paquetes.
4. 6 Pipes disponibles como receptores, multireceptor.

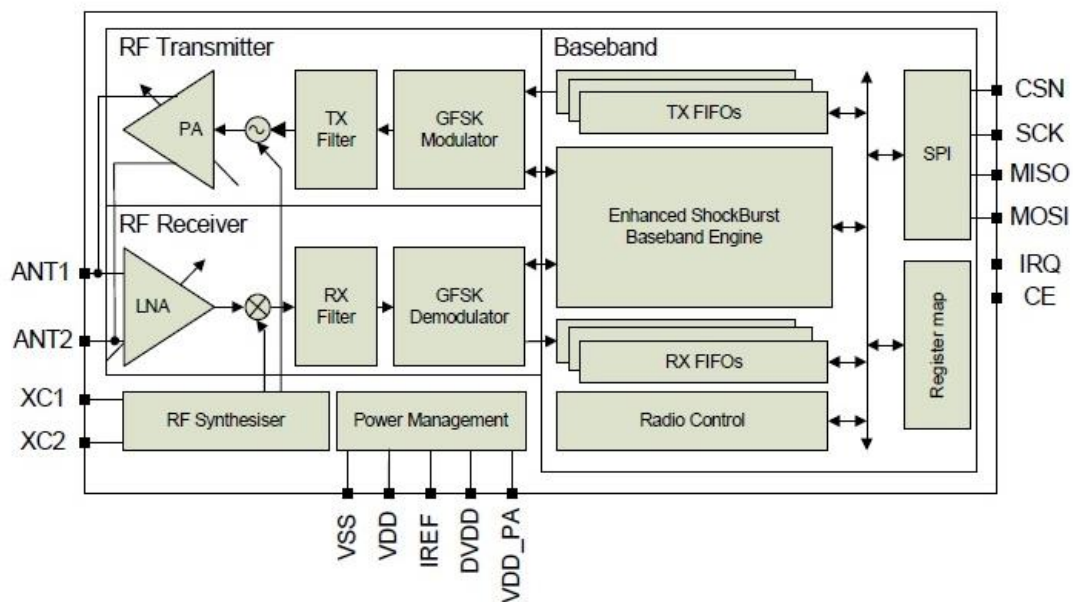
Manejo de potencia

1. Regulador de voltaje integrado.
2. Rango de alimentación entre 1.9 y 3.6 voltios.

Interfaz del anfitrión

1. pines de hardware para SPI.
2. Máximo 10 Mbps.
3. 3 FIFO's de transmisión y 3 de recepción de 32 bytes cada una.

Diagrama de bloques del módulo de RF



Modos de operación del nRF24l01+

- Power Down Mode: este modo nos permite tener un bajo consumo de energía, ya que la corriente que se utiliza es muy baja, se mantienen los registros y el SPI se mantiene activo, esperando para volver a activarse. Se entra en este modo cambiando el bit de PWR_UP del registro de CONFIG a 0.
- Standby-I Mode: poniendo el bit de PWR_UP en 1, el dispositivo entra en Standby I, es usado para minimizar la corriente utilizada manteniendo un corto tiempo de arranque. En este modo solo una parte del cristal oscilador está activo. Cambia al modo activo solo cuando CE está en alto cuando CE vuelve a estar en bajo el nRF24l01 vuelve al modo de StandBy-I, en ambos casos, tanto transmisor como receptor.
- Standby-II mode: entra en este estado si un dispositivo se encuentra en modo TX, con su FIFO de TX vacío y con CE en alto. Un reloj extra es activado y es usada más corriente comparado con el standby-I.
- RX mode: es un modo activo donde el modulo es usado como receptor. Para entrar en este modo el modulo tiene que tener los bits de PWR_UP, PRIM_RX en alto y el pin de CE en alto.
En RX mode el receptor demodula las señales del canal de RF, presentando constantemente la data demodulada al protocolo de banda base. El protocolo de banda base se encuentra constantemente buscando por un paquete valido. Si se encuentra un paquete valido (validado por el CRC y que coincida con la dirección) el payload del paquete es enviado a un espacio vacío en la FIFO de recepción. Si no hay espacios vacíos, el paquete se desecha.
- TX mode: es un modo activo donde el modulo es usado como transmisor. Para entrar en este modo el modulo tiene que tener los bits de PWR_UP en alto, PRIM_RX en bajo, tiene que haber un payload en el FIFO de transmisión y el pin

de CE en alto por más de 10 us.

El modulo se queda en modo de transmisión hasta que termina de enviar todo los paquetes. Si CE=0, el modulo vuelve al modo de standby-I. Si CE=1 el estado del FIFO de transmisión determina la próxima acción. Si el FIFO de transmisión no está vacío, el modulo mantiene el modo de TX y transmite el siguiente paquete. Si el FIFO de transmisión está vacío el modulo va al modo de standby-II. Es importante recordar no mantener el modo de transmisión por más de 4ms al mismo tiempo.

Canales de RF

Es importante fijar tanto receptor como transmisor en el mismo canal para que se pueda comunicar uno con otro.

A velocidades de 250 Kbps y 1 Mbps se tiene un ancho de banda de 1 MHz mientras que a 2 Mbps se tiene un ancho de banda de 2MHz.

La frecuencia donde empieza la señal se consigue mediante la siguiente formula:

$$F=2400 + RF_CH \text{ [MHz]}$$

Donde RF_CH es el número ubicado en el registro 5.

Forma del paquete de transmisión

Preámbulo 1 byte	Dirección 3-5 bytes	Campo de control del paquete 9 bits	Payload 0-32 bytes	CRC 1-2 bytes
------------------	---------------------	-------------------------------------	--------------------	---------------

1. El preámbulo es una secuencia de bits usado para sincronizar el demodulador del receptor a los bits entrantes.
2. La dirección es el nombre que le dimos al pipe de transmisión, debe coincidir con el pipe de recepción que tiene el receptor.
3. El campo de control se divide a su vez en

Largo del payload 6bits	PID 2bits	NO_ACK 1 bit
-------------------------	-----------	--------------

- 3.1. Largo del payload como su nombre lo indica, es el largo del payload, máximo 32.
- 3.2. El PID es un identificador del paquete.
- 3.3. NO_ACK si está en alto quiere decir que no hace falta que se envíe el “acknowledgment”
4. Payload la data enviada.
5. CRC (verificación por redundancia cíclica) es el mecanismo de detector de errores en el paquete. Puede ser de 1 byte o de 2 bytes y es calculado sobre la dirección, el campo de control del paquete y el payload.

Largo del payload, dinámico y estático

Como ya se ha mencionado el payload puede variar entre 1 y 32 bytes, sin embargo existen dos maneras de trabajarlos, de manera dinámica y de manera estática.

- Estática: si trabajamos con esta configuración tenemos un tamaño fijo de data a enviar, ambos, tanto receptor como transmisor tienen que saber el tamaño de la data de antemano, si se envía menos data el programa fallara, y si se envía más data el programa funcionara pero cortara la data.
- Dinámica: en esta configuración el transmisor podía enviar cualquier cosa (siempre que esté en el rango de los 32 bytes) y el receptor puede averiguar su tamaño de manera dinámica, de esta manera no hace falta la necesidad de conocer el tamaño total.

Comandos de SPI

Nombre del comando	palabra del comando (binario)	# bytes de data	operación
R_REGISTER	000A AAAA	1 a 5 y el LSByte primero	Comando de lectura y del registro de estado. AAAAA = 5 bits de dirección del mapa de registros
W_REGISTER	001A AAAA	1 a 5 y el LSByte primero	Comando de escritura y del registro de estado. AAAAA = 5 bits de dirección del mapa de registros. Solo se puede ejecutar en “power down mode” o en “standby modes”
R_RX_PAYLO AD	0110 0001	1 a 32 y el LSByte primero	Lee el payload recibido disponible. El payload es borrado del FIFO después de ser leído. Usado en modo de recepción.
W_TX_PAYLO AD	1010 0000	1 a 32 y el LSByte primero	Escribe en el payload de transmisión. Usado en modo de transmisión.
FLUSH_TX	1110 0001	0	Vacía la FIFO de transmisión. Usado en modo de transmisión.
FLUSH_RX	1110 0010	0	Vacía la FIFO de recepción. Usado en modo de recepción.

			No debe ser usado durante la transmisión del reconocimiento, y que puede causar fallos.
REUSE_TX_PAYLOAD	1110 0011	0	Usado en modo de transmisión. Reúsa el último payload transmitido. Se mantiene activo hasta que W_TX_PAYLOAD o FLUSH_TX son ejecutados.
R_RX_PL_WID	0110 0000	1	Lee el tamaño del payload que se encuentra de primero en el FIFO de RX
W_ACK_PAYLOAD	1010 1PPP	1 a 32 y el LSByte primero	Usado en modo de RX. Escribe un payload para ser transmitido, junto al paquete ACK en el pipe PPP (PPP pertenece al rango de 000 a 101).
W_TX_PAYLOAD_NO_ACK	1011 0000	1 a 32 y el LSByte primero	Usado en modo de transmisor. Desactiva el auto reconocimiento en ese paquete en específico.
NOP	1111 1111	0	No hace nada. Puede usarse para leer el registro de STATUS

Mapa de Registros

El nRF24l01+ posee 30 registros, que se utilizan para su configuración, ellos son:

Dirección (Hexadecimal)	mnemotécnico	Bit	Valor por defecto	Tipo	Descripción
00	CONFIG				Registro de configuración
	Reservado	7	0	R/W	Solo se permite '0'
	MASK_RX_DR	6	0	R/W	Interrupción de mascara causada por RX_DR 1: la interrupción no se refleja en el pin de IRQ 0: se refleja RX_DR activando la interrupción en el pin de IRQ
	MASK_TX_DS	5	0	R/W	Interrupción de mascara causada por TX_DS 1: la interrupción no se refleja en el pin de IRQ 0: se refleja TX_DS activando la interrupción en el pin de IRQ
	MASK_MAX_RT	4	0	R/W	Interrupción de mascara causada por MAX_RT 1: la interrupción no se refleja en el pin de IRQ 0: se refleja MAX_RT activando la interrupción en el pin de IRQ

Dirección (Hexadecimal)	mnemotécnico	Bit	Valor por defecto	Tipo	Descripción
	EN_CRC	3	1	R/W	Habilitar CRC. Se fuerza a 1 si uno de los bits del registro EE_AA está en alto.
	CRCO	2	0	R/W	Esquema de codificación del CRC 0: 1 byte 1: 2 bytes
	PWR_UP	1	0	R/W	1: encendido 0: apagado(power down mode)
	PRIM_RX	0	0	R/W	Define si es transmisor o receptor 1: receptor ; 0: transmisor
01	EE_AA Enhanced ShockBurst				Habilitar el auto reconocimiento
	Reservado	7:6	00	R/W	Solo se permite '00'
	ENAA_P5	5	1	R/W	Habilita auto reconocimiento de la data en el pipe 5
	ENAA_P4	4	1	R/W	Habilita auto reconocimiento de la data en el pipe 4
	ENAA_P3	3	1	R/W	Habilita auto reconocimiento de la data en el pipe 3
	ENAA_P2	2	1	R/W	Habilita auto reconocimiento de la data en el pipe 2
	ENAA_P1	1	1	R/W	Habilita auto reconocimiento de la data en el pipe 1

Dirección (Hexadecimal)	mnemotécnico	Bit	Valor por defecto	Tipo	Descripción
	ENAA_P0	0	1	R/W	Habilita auto reconocimiento de la data en el pipe 0
02	EN_RXADDR				Habilitar las direcciones de recepción
	Reservado	7:6	00	R/W	Solo se permite '00'
	ERX_P5	5	0	R/W	Habilitar data en el pipe 5
	ERX_P4	4	0	R/W	Habilitar data en el pipe 4
	ERX_P3	3	0	R/W	Habilitar data en el pipe 3
	ERX_P2	2	0	R/W	Habilitar data en el pipe 2
	ERX_P1	1	0	R/W	Habilitar data en el pipe 1
	ERX_P0	0	0	R/W	Habilitar data en el pipe 0
03	SETUP_AW				Configurar el largo de las direcciones (para todos los pipes)
	Reservado	7:2	000000	R/W	Solo se permite '000000'
	AW	1:0	11	R/W	Largo de las direcciones: '00': ilegal '01': 3 bytes '10': 4 bytes '11': 5 bytes El byte menos significativo se usa si el largo es menor de 5 bytes

Dirección (Hexadecimal)	mnemotécnico	Bit	Valor por defecto	Tipo	Descripción
04	SETUP_RETR				Configurar retransmisiones
	ARD	7:4	0000	R/W	Retraso para retransmitir ‘0000’ retraso de 250us ‘0000’ retraso de 500us ‘0000’ retraso de 750us ‘0000’ retraso de 4000us
	ARC	0:3	0011	R/W	Contador de retransmisiones ‘0000’ retransmisiones deshabilitadas ‘0001’ máximo 1 retransmisión si falla el AA ‘1111’ máximo 15 retransmisión si falla el AA AA (auto reconocimiento)
05	RF_CH				Canal de RF
	Reservado	7	0	R/W	Solo se permite ‘0’
	RF_CH	6:0	000001 0	R/W	Configura la frecuencia a la cual trabaja el nRF24l01+
06	RF_STEUP				Registro de configuración de RF
	CONT_WAVE	7	0	R/W	Cuando está en alto habilita la transmisión continua en la portadora
	reservado	6	0	R/W	Solo se permite ‘0’

Dirección (Hexadecimal)	mnemotécnico	Bit	Valor por defecto	Tipo	Descripción
	RF_DR_LOW	5	0	R/W	Se coloca la tasa de datos a 250 Kbps.
	PLL_LOCK	4	0	R/W	Forzar señal de bloqueo del PLL
	RF_DR_HIGH	3	1	R/W	Selecciona la tasa de datos entre las altas velocidades (este bit es “don't care” si RF_DR_LOW está en alto) Codificación: [RF_DR_LOW,RF_DR_HIGH] 00 – 1Mbps 01 – 2Mbps 10 – 250kbps 11 – reservado
	RF_PWR	2:1	11	R/W	Selecciona la potencia de salida en modo de TX ‘00’: -18 dbm ‘01’: -12 dbm ‘10’: -6 dbm ‘11’: -0 dbm
	Obsoleto	0			No afecta
07	STATUS				Registro de estado
	Reservado	7	0	R/W	Solo se permite ‘0’
	RX_DR	6	0	R/W	Interrupción de data lista. 1 cuando llega nueva data al FIFO de RX.

Dirección (Hexadecimal)	mnemotécnico	Bit	Valor por defecto	Tipo	Descripción
	TX_DS	5	0	R/W	Interrupción de data enviada. 1 cuando el paquete en el FIFO es transmitido. Si el auto reconocimiento está activo, será 1 cuando llegue el ACK.
	MAX_RT	4	0	R/W	Interrupción ya que se ha alcanzado el máximo número de retransmisiones. 1 si se alcanzó el MAX_RT
	RX_P_NO	3:1	111	R	Número del pipe de datos para la carga útil disponible para la lectura del FIFO de RX 000-101: número del pipe 110: no se usa 111: RX FIFO vacía
	TX_FULL	0	0	R	Bandera del FIFO de TX 1: FIFO full 0: espacio disponible en el FIFO
08	OBSERVE_TX				
	PLOS_CNT	7:4	0	R	Cuenta paquetes perdidos. La cuenta llega hasta 15, la cuenta se reinicia escribiendo en el RF_CH.
	ARC_CNT	3:0	0	R	Cuenta paquetes retransmitidos. La cuenta se reinicia cuando empieza la transmisión de un nuevo paquete.

Dirección (Hexadecimal)	mnemotécnico	Bit	Valor por defecto	Tipo	Descripción
09	RPD				
	Reservado	7:1	000000	R	
	RPD	0	0	R	Detector de potencia recibida, es 1 si la potencia recibida es mayor que -64 dBm y 0 si es menor
0A	RX_ADDR_P0	39:0	0xE7E7E7E7	R/W	Dirección del pipe 0 de recepción, con un largo máximo de 5 bytes (el byte menos significativo se escribe primero. Se escribe el número de bytes que se definió en SETUP_AW)
0B	RX_ADDR_P1	39:0	0xC2C2C2C2	R/W	Dirección del pipe 1 de recepción, con un largo máximo de 5 bytes (el byte menos significativo se escribe primero. Se escribe el número de bytes que se definió en SETUP_AW)
0C	RX_ADDR_P2	7:0	0xC3	R/W	Dirección del pipe 2 de recepción. Solo el byte menos significativo. Los bytes más significativos son iguales a los del RX_ADDR_P1
0D	RX_ADDR_P3	7:0	0xC4	R/W	Dirección del pipe 3 de recepción. Solo el byte menos significativo. Los bytes más significativos son iguales a los del RX_ADDR_P1

Dirección (Hexadecimal)	mnemotécnico	Bit	Valor por defecto	Tipo	Descripción
0E	RX_ADDR_P4	7:0	0xC5	R/W	Dirección del pipe 4 de recepción. Solo el byte menos significativo. Los bytes más significativos son iguales a los del RX_ADDR_P1
0F	RX_ADDR_P5	7:0	0xC6	R/W	Dirección del pipe 5 de recepción. Solo el byte menos significativo. Los bytes más significativos son iguales a los del RX_ADDR_P1
10	TX_ADDR	39:0	0xE7E7 E7E7 7	R/W	Dirección de transmisión. Usada solo en el modo de transmisión (el byte menos significativo se escribe primero), colocar RX_ADDR_P0 igual a esta dirección para tener AA si se tiene Enhanced Shockburst activo.
11	RX_PW_P0				
	Reservado	7:6	00	R/W	Solo se permite '00'
	RX_PW_P0	5:0	0	R/W	numero de bytes permitidos en el payload de RX el data pipe 0 (1-32 bytes) 0 no se usa el pipe 000001=1 byte 100000=32 bytes

Dirección (Hexadecimal)	mnemotécnico	Bit	Valor por defecto	Tipo	Descripción
12	RX_PW_P1				
	Reservado	7:6	00	R/W	Solo se permite '00'
	RX_PW_P1	5:0	0	R/W	Es el número de bytes permitidos en el payload de RX el data pipe 1 (1-32 bytes) 0 no se usa el pipe 000001=1 byte 100000=32 bytes
13	RX_PW_P2				
	Reservado	7:6	00	R/W	Solo se permite '00'
	RX_PW_P2	5:0	0	R/W	Es el número de bytes permitidos en el payload de RX el data pipe 2 (1-32 bytes) 0 no se usa el pipe 000001=1 byte 100000=32 bytes
14	RX_PW_P3				
	Reservado	7:6	00	R/W	Solo se permite '00'
	RX_PW_P3	5:0	0	R/W	Es el número de bytes permitidos en el payload de RX el data pipe 3 (1-32 bytes) 0 no se usa el pipe 000001=1 byte 100000=32 bytes

Dirección (Hexadecimal)	mnemotécnico	Bit	Valor por defecto	Tipo	Descripción
15	RX_PW_P4				
	Reservado	7:6	00	R/W	Solo se permite '00'
	RX_PW_P4	5:0	0	R/W	Es el número de bytes permitidos en el payload de RX el data pipe 4 (1-32 bytes) 0 no se usa el pipe 000001=1 byte 100000=32 bytes
16	RX_PW_P5				
	Reservado	7:6	00	R/W	Solo se permite '00'
	RX_PW_P5	5:0	0	R/W	Es el número de bytes permitidos en el payload de RX el data pipe 5 (1-32 bytes) 0 no se usa el pipe 000001=1 byte 100000=32 bytes
17	FIFO STATUS				Registro del estado del FIFO
	Reservado	7	0	R/W	solo se permite '0'
	TX_REUSE	6	0	R	Usado por el dispositivo en modo transmisión. Poner el rfce en alto por al menos 10 us para reusar el último payload transmitido. El reusó está activo mientras W_TX_PAYLOAD o

					FLUSH_TX están ejecutándose
Dirección (Hexadecimal)	mnemotécnico	Bit	Valor por defecto	Tipo	Descripción
	TX_FULL	5	0	R	Bandera de FIFO de TX full 1: FIFO DE TX full 0: hay espacio disponible
	TX_EMPTY	4	1	R	Bandera de FIFO de TX vacío 1: FIFO de TX vacío 0: hay data
	Reservado	3:2	00	R/W	Solo se permite '00'
	RX_FULL	1	0	R	Bandera de FIFO de RX full 1: FIFO DE RX full 0: hay espacio disponible
	RX_EMPTY	0	1	R	Bandera de FIFO de RX vacío 1: FIFO de RX vacío 0: hay data
N/A	ACK_PLD	255: 0	X	W	Usado solo en modo de recepción. Solo se puede tener un máximo de 3 paquetes ACK pendientes.
N/A	TX_PLD	255: 0	X	W	Usado solo en modo de transmisión. Se implementa como en FIFO de 3 niveles.
N/A	RX_PLD	255: 0	X	R	Usado solo en modo de recepción. Se implementa como en FIFO de 3 niveles. Todos los pipes de recepción comparten el mismo FIFO.

Dirección (Hexadecimal)	mnemotécnico	Bit	Valor por defecto	Tipo	Descripción
1C	DYNPD				Habilitar payload dinámico
	Reservado	7:6	0	R/W	Solo se permite '00'
	DPL_P5	5	0	R/W	Habilita el payload dinámico en el pipe 5 (requiere EN_DPL y ENAA_P5)
	DPL_P4	4	0	R/W	Habilita el payload dinámico en el pipe 4 (requiere EN_DPL y ENAA_P4)
	DPL_P3	3	0	R/W	Habilita el payload dinámico en el pipe 3 (requiere EN_DPL y ENAA_P3)
	DPL_P2	2	0	R/W	Habilita el payload dinámico en el pipe 2 (requiere EN_DPL y ENAA_P2)
	DPL_P1	1	0	R/W	Habilita el payload dinámico en el pipe 1 (requiere EN_DPL y ENAA_P1)
	DPL_P0	0	0	R/W	Habilita el payload dinámico en el pipe 0 (requiere EN_DPL y ENAA_P0)
1D	FEATURE				Registro de características
	Reservado	7:3	0	R/W	Solo se permite '00000'
	EN_DPL	2	0	R/W	Habilita el payload dinámico
	EN_ACK_PAY	1	0	R/W	Habilita payload con reconocimiento
	EN_DYN_ACK	0	0	R/W	Habilita el comando W_TX_PAYLOAD_NOACK

Galileo Gen 2

1. Posee un procesador de aplicaciones Intel Quark SoC x1000, de 32 bits, un núcleo y es compatible con la arquitectura de conjunto de instrucciones (ISA) del procesador Intel Pentium, que funciona a velocidades de hasta 400 MHz.
2. Es compatible con una amplia gama de interfaces de entrada y salida estándar de la industria, como la ranura de microSD, el puerto anfitrión USB y el puerto cliente USB.
3. Posee una DDR3 de 256MB, SRAM integrada de 512 kb, una memoria NOR flash de 8MB y EEPROM de 8kb de serie en placa; además, admite una tarjeta microsd de hasta 32 GB.
4. Tiene compatibilidad de hardware y pines con una amplia gama de protectores Arduino Uno R3.
5. Programable a través del entorno de desarrollo integrado (IDE) de Arduino, compatible con los sistemas operativos anfitriones Microsoft Windows, Mac OS y Linux.
6. Compatibilidad con la versión Yocto 1.4 Poky de Linux.
7. Posee 12 puertos GPIO de alta velocidad.

Primeras configuraciones Galileo Gen 2

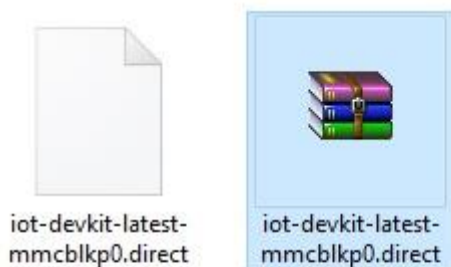
La placa de desarrollo de Intel se puede utilizar varios sistemas operativos, se mencionaran algunas:

1. Imagen SPI, esta versión viene preinstalada en la placa y no necesita una memoria microSD, pero tiene muchas limitaciones, por ejemplo no guarda ningún *sketch* de arduino luego de reiniciarse, no posee los controladores para los módulos de *WIFI*, etc.
2. Imagen “SD-Card”, esta versión necesita una memoria microSD, pero tiene beneficios que no tiene la imagen de SPI. Posee los controladores de *WIFI*, OpenCV, ALSA, y Node.js por mencionar algunos.
3. Imagen *Iot* Devkit, esta versión también necesita una memoria microSD y su proceso de instalación es un poco más complicado, sin embargo posee varias características interesantes. Es muy sencillo establecer conexiones de red y posee más herramientas para desarrolladores, soporta C, C++, Python y JavaScript.

En este proyecto nos enfocaremos en la imagen del *IOT* (Internet de las cosas). Se necesitan una tarjeta micro SD formateada con FAT32 y que sea mayor a 1GB y menor a 32GB y una computadora con lector de microSD (con un adaptador también funciona).

Instalación en Linux:

Primero nos descargamos el software de la página de la *Iot* [1] en la sección de imágenes, seleccionamos la última versión. Luego de descargarla la descomprimos y



debería quedar algo como esto

Ya con esto hecho procedemos a introducir la memoria microSD a la computadora, abrimos una terminal y ejecutamos el siguiente comando como administrador

```
luis@Luis-pc ~ $ sudo fdisk -l
[sudo] password for luis:

Disco /dev/sda: 500.1 GB, 500107862016 bytes
255 cabezas, 63 sectores/pista, 60801 cilindros, 976773168 sectores en total
Unidades = sectores de 1 * 512 = 512 bytes
Tamaño de sector (lógico / físico): 512 bytes / 512 bytes
Tamaño E/S (mínimo/óptimo): 512 bytes / 512 bytes
Identificador del disco: 0xa0c9de0b

Dispositivo Inicio Comienzo Fin Bloques Id Sistema
/dev/sda1 2048 37750783 18874368 27 WinRE NTFS oculto
/dev/sda2 * 37750784 37955583 102400 7 HPFS/NTFS/exFAT
/dev/sda3 37955584 773348600 367696508+ 7 HPFS/NTFS/exFAT
/dev/sda4 773349374 976771071 101710849 f W95 Ext'd (LBA)
/dev/sda5 968689664 976771071 4040704 82 Linux swap / Solaris
/dev/sda6 773349376 968689663 97670144 83 Linux

Las entradas de la tabla de particiones no están en el orden del disco

Disco /dev/sdb: 4009 MB, 4009754624 bytes
255 cabezas, 63 sectores/pista, 487 cilindros, 7831552 sectores en total
Unidades = sectores de 1 * 512 = 512 bytes
Tamaño de sector (lógico / físico): 512 bytes / 512 bytes
Tamaño E/S (mínimo/óptimo): 512 bytes / 512 bytes
Identificador del disco: 0x02006a22

Dispositivo Inicio Comienzo Fin Bloques Id Sistema
/dev/sdb1 * 2048 7831551 3914752 c W95 FAT32 (LBA)
```

En la imagen se pueden observar 2 memorias, una es el disco duro y la otra es una microSD, anotamos donde se encuentra, en este caso /dev/sdb, hay que recalcar que no es sdb1 sino sdb.

Ahora en la terminal nos movemos hacia la carpeta donde esta descargado y extraído el software de la *Iot* y ejecutamos el siguiente comando como administrador

```
sudo dd if=software_descargado of=/dev/diskname bs=3M conv=fsync
```

Al final tendremos algo como esto

```
luis@Luis-pc /media/luis/Acer/Users/luisalfredo/Downloads/IOT $ sudo dd if=iot-devkit-latest-mmcbkp0.direct of=/dev/diskname bs=3M conv=fsync
[sudo] password for luis:
450+1 registros leídos
450+1 registros escritos
1417675776 bytes (1,4 GB) copiados, 62,0676 s, 22,8 MB/s
```

Con esto ya tenemos lista nuestra microSD para ponerla en nuestro Galileo Gen 2.

Instalación en Windows

La instalación en Windows es más sencilla, necesitamos un programa llamado Win32Disk-Manager que podemos conseguir fácilmente, primero, necesitamos ejecutar el programa como administrador, en la ventana oprimimos la carpeta que se encuentra arriba a la derecha y buscaremos el software de la *Iot*, luego de ubicarlo, en la ventana que se encuentra debajo de device (arriba a la derecha) seleccionaremos la microSD que tenemos, luego oprimimos write y esperamos, al final en la microSD tendremos algo como esto.



Conexión con el Galileo

Ya instalado el sistema operativo en la microSD la colocamos en la ranura para microSD de nuestra placa.

Existen 4 maneras para conectarnos a nuestra terminal del Galileo

1. Utilizando el header UART.
2. Utilizando una conexión de Ethernet.
3. Conexión cableada a un punto de acceso.
4. Conexión por medio de WIFI.

Nosotros nos centraremos en la conexión cableada a router.

Primero colocamos la microSD en el galileo, conectamos el galileo y el punto de acceso mediante el cable de Ethernet, luego de esto encendemos nuestro Galileo.

Esperamos un poco a que arranque el sistema operativo, este por defecto trae el cliente DHCP activo, así que automáticamente pedirá un numero IP al arrancar.

En Linux

Para saber que numero IP le dio nuestro punto de acceso al galileo nos metemos en la consola y ejecutamos las siguientes líneas de comando:

```
Sudo apt-get install nmap
```

Luego de instalarlo ejecutamos lo siguiente

```
Sudo nmap -sP 192.168.1.0/24
```

Esto se encargara de mapear toda nuestra red y nos dirá que numero IP le fue asignado a nuestro galileo, la salida del programa es así:

```
sabrina@Sabrina-PC ~ $ sudo nmap -sP 192.168.1.0/24
[sudo] password for sabrina:

Starting Nmap 6.40 ( http://nmap.org ) at 2015-08-30 22:35 VET
Nmap scan report for 192.168.1.1
Host is up (0.0076s latency).
MAC Address: 2C:B0:5D:2B:72:66 (Netgear)
Nmap scan report for 192.168.1.3
Host is up (0.12s latency).
MAC Address: 14:F6:5A:BA:D9:57 (Unknown)
Nmap scan report for 192.168.1.8
Host is up (0.080s latency).
MAC Address: C4:62:EA:31:70:4D (Samsung Electronics Co.)
Nmap scan report for 192.168.1.12
Host is up (0.010s latency).
MAC Address: 10:0B:A9:2A:95:B8 (Intel Corporate)
Nmap scan report for 192.168.1.11
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 3.56 seconds
```

Se puede observar que sale el nombre de los equipos utilizados en la red, esto se debe a que el programa los reconoce por su dirección MAC

En Windows

Este método sirve en cualquier plataforma, para ello debemos entrar en nuestro punto de acceso escribiendo en cualquier explorador de internet 192.168.1.1 aquí se nos pedirá un usuario y contraseña, solicítenselo al administrador, por defecto el nombre de usuario y la contraseña son: admin o nombre de usuario: admin y contraseña: password, una vez ahí buscamos una tabla que contenga todos los dispositivos vinculados, también se le puede conseguir con el nombre de tabla DHCP, ahí podremos ver el IP que fue asignado a nuestro galileo.

Ya conocido el numero IP por fin es hora de la conexión.

En Linux

En Linux es bastante sencillo, solamente usamos el protocolo ssh con nombre de usuario root e IP la que conseguimos, en este caso no se nos pedirá contraseña ya que no posee, la sintaxis es:

ssh root@Direccion_IP

Una vez ejecutado nos preguntara si estamos seguros, colocamos que sí y nos habremos conectado, para colocarle contraseña se puede usar el comando passwd el cual nos permite cambiar la contraseña.

En Windows

En Windows también nos conectaremos con el protocolo ssh, sin embargo antes de eso es necesario descargar un software llamado Putty, el cual nos permitirá conectarnos.

Una vez descargado, abrimos putty y en donde dice *IP address* colocamos el IP antes encontrado, nos volverá a preguntar si estamos seguros de conectarnos, colocamos que sí, luego nos preguntara el nombre de usuario, aquí colocamos root y si hace falta contraseña nos la pedirá luego de pedir el usuario, y ya nos habremos conectado.

Recomendaciones

1. El galileo viene con un editor de texto llamado vi, sin embargo se puede instalar otro, se recomienda nano, para descargarlo buscan el instalador en internet y utilizan el comando wget (en la terminal del galileo) seguido del link de descarga al archivo, de esta manera de descargara directamente al galileo, también lo pueden descargar en sus computadoras y pasarlos por scp (caso de Linux) o winSCP (caso de Windows), e instalarlo.
2. Para evitar el problema de buscar la dirección IP cada vez que se va a conectar se recomienda colocar un IP fijo, en esta distribución de Linux, se necesita ir a la carpeta de test que se encuentra en la carpeta de connman que se puede encontrar escribiendo en la consola:


```
cd /usr/lib/connman/test
```

Una vez en esta carpeta ejecutamos el programa *get-service* este nos dará un dato necesario para cambiar el IP, al ejecutarlo la salida nos da:

```

root@kali:~# ./usr/lib/connman/daemon --test --wired-services
[ /net/connman/service/ethernet_c8a030ab323a_cable ]
IPv6.Configuration = { Method=auto, Privacy=disabled }
AutoConnect = true
Name = Wired
Nameservers = [ 192.168.1.1 ]
Provider = { }
Favorite = true
Domains.Configuration = [ ]
Timeservers.Configuration = dbus.Array([], signature=dbus.Signature('s'), variant_level=1)
State = online
Proxy = { Method=direct }
Nameservers.Configuration = [ ]
IPv4 = { Netmask=255.255.255.0 Gateway=192.168.1.1 Method=dhcp Address=192.168.1.100 }
Timeservers = dbus.Array([dbus.String(u'192.168.1.1')], signature=dbus.Signature('s'), variant_level=1)
IPv6 = { }
Domains = [ home.gateway ]
Ethernet = { Interface=eth0 MTU=1500 Method=auto Address=C8:A0:30:AB:32:3A }
Security = [ ]
Proxy.Configuration = { }
Type = ethernet
Immutable = false
IPv4.Configuration = { Method=dhcp }

```



Lo que nos interesa aquí es la línea encerrada en un círculo, lo copiamos, ya que nos hará falta en el próximo paso.

Luego de que tenemos esto, ejecutamos el *set-ipv4-method* sin embargo este programa necesita varios parámetros:

- El primero es el dato que acabamos de conseguir.
- El segundo es donde elegimos que deseamos, si queremos que sea por dhcp, manual o lo apagamos, para colocarlo fijo, escribimos manual
- El tercero es la dirección IP que se desea, hay que recordar que tiene que estar en el rango de la red local y no tiene que estar ocupado.
- Luego viene la máscara de la red.
- Por último de *Gateway*, los últimos dos parámetros dependen también de su red local.

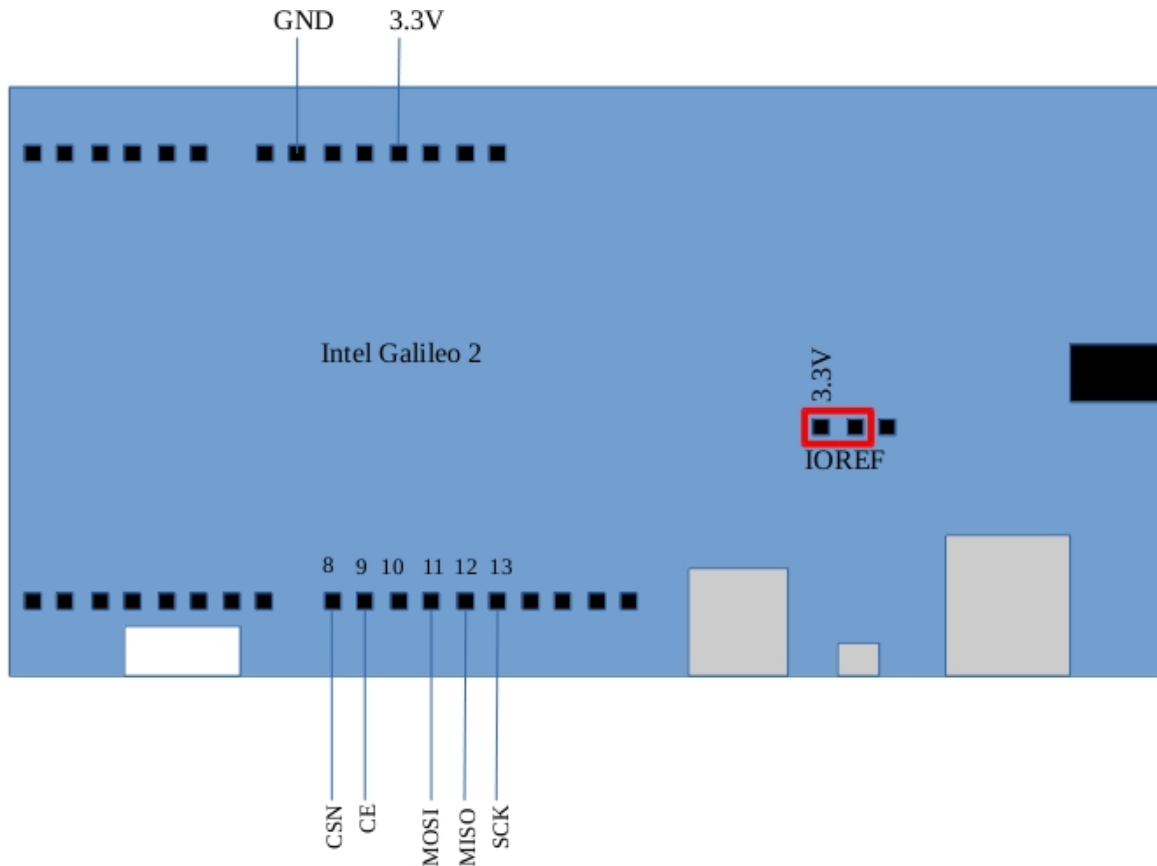
La sintaxis se vería así:

```
./set-ipv4-method <service> [off|dhcp|manual <address> [netmask] [gateway]]
```

Cuando ejecutemos esta línea de comando nuestra conexión ssh quedará inservible ya que hemos cambiado exitosamente el número IP.

Comunicación con nRF24I01+

La comunicación se estableció mediante SPI, para ello se utilizaron cables *jumper* macho-hembra, ya que los puertos del RF son todos machos y los del galileo son todos hembra, de esta manera logramos conectar los dispositivos, aquí se muestran los puertos del galileo:



Los puertos de CSN y CE son escogidos mediante software, en este programa se escogieron el 8 y el 9, es importante observar que el puente marcado en rojo, se debe colocar de esa manera, de lo contrario se obtendrán 5 voltios en vez de 3.3 voltios, recordemos que el nRF24L01+ utiliza 3,3 voltios de alimentación, colocarle 5 voltios podría causar un daño al equipo.

Ya que tenemos los dispositivos conectados es hora de empezar a hablar del software, para poder hablar con el SPI del galileo, se utilizó la biblioteca mraa, esta biblioteca está basada en C, C++ y se puede adaptar a JavaScript y a Python.

Existen interfaces que te permiten sacar el mayor provecho a la biblioteca mraa, y te harán más fácil la comprensión de la misma:

C	GPIO	I2c	aio	pwm	spi	UART	common
Clases C++	Clase GPIO	Clase I2c	Clase aio	Clase pwm	Clase spi	Clase UART	Clase common

Para este proyecto se utilizó la interfaz en C++, y específicamente la clase GPIO y la clase SPI, para más detalles de la librería ir a [2], la clase GPIO nos permitió elegir los puertos de CE, CSN y manipularlos libremente, mientras que la clase de SPI nos permitió mediante sus funciones, fijar la frecuencia del SPI y escribir bytes por el SPI hacia el RF, esto nos permitió crear un controlador, que puede manipular los registros, escribir datos y leer datos del RF.

El algoritmo del programa consiste básicamente en

1. Declarar todas las variables necesarias de SPI y GPIO.
2. Arrancar la configuración de todos los Registros.
3. Fijar el nombre de los Pipes que se van a utilizar para transmitir y para recibir.
4. Vaciar los FIFO'S tanto de transmisión como de recepción.
5. Activar el modo de recepción.
6. Imprimir los registros para verificar que todo está en orden.
7. Entrar en un ciclo el cual permanentemente está chequeando el registro 7 para ver si se recibió data.
8. Guardar la data en un buffer.
9. Cuando se acabe imprimirla en un archivo.
10. Salir del loop.
11. Abrir el archivo, fragmentarlo y enviarlo por sockets a la computadora de origen.

Resultados

Distancia (m)	Velocidad	Paq. enviados	Paq. Recibidos	Paq. Devueltos
25	250Kbps	500	500	483
25	1Mbps	500	497	299
25	2Mbps	500	490	325
30	250Kbps	500	500	462
30	1Mbps	500	500	320
30	2Mbps	500	494	410
36	250Kbps	500	500	491
36	1Mbps	500	499	340
36	2Mbps	500	491	416
42	1Mbps	500	489	291
42	2Mbps	500	489	315
48	250Kbps	500	484	88
48	1Mbps	500	0	0
48	2Mbps	500	0	0
54	250Kbps	500	323	131

El experimento se realizo en el campus de la universidad, bajo el puente de la biblioteca para que no existieran obstrucciones por parte de las personas que hacen vida en la universidad.

Las medidas se realizaron en un principio a una altura de 1,5 metros y conforme nos alejábamos se iban perdiendo más paquetes, que cumple con lo esperado. Sin embargo en un punto ya no recibíamos nada, y decidimos elevar a una altura de 2 metros aproximadamente a los módulos lo cual logro que una mayor cantidad de paquetes llegaran.

Bibliografía

[1] <http://iotdk.intel.com/>

[2] <http://iotdk.intel.com/docs/master/mraa/>

<https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi>

<https://www.nordicsemi.com/eng/Products/2.4GHz-RF/nRF24L01P>

<http://eleceng.dit.ie/frank/arm/BareMetalLPC1114/nrf24l01/index.html>