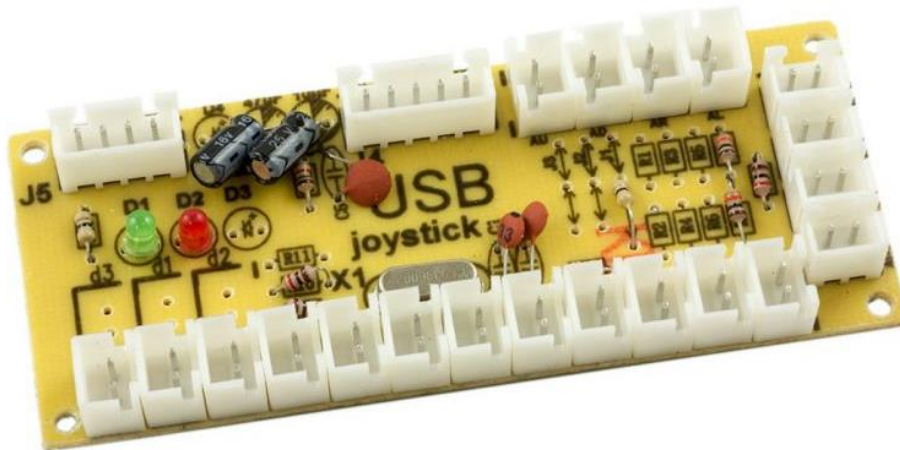


Lectura interruptores en Linux con tarjeta “Generic USB joystick” de DragonRise para aplicaciones IoT

V03
mayo 2020
junavarg
junavarg@hotmail.com



Motivación

En los proyectos de IoT es frecuente necesitar manejar información que se obtiene por el cierre o apertura de interruptores: pulsadores, interruptores, contactos reed, conmutadores, finales de carrera, etc.

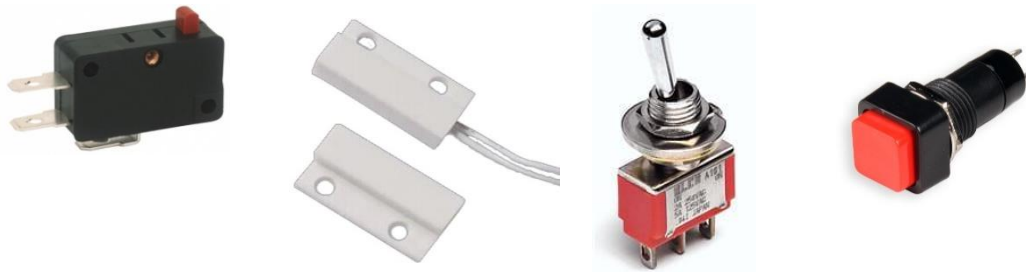


Ilustración 1. Ejemplo de interruptores conectables a sistemas IoT

Una forma de conectar estos interruptores es haciendo uso de puertos de entrada de los GPIO presentes en los SBC “single-board computers” (por ejemplo Raspberry pi) o “single-board microcontrollers” (por ejemplo Arduino). Los principales problemas de esta aproximación son:

- Cableado muy desordenado especialmente en sistemas que precisen un alto número de interruptores.
- Posibilidad de avería en la placa al manipular en contactos no estructurados y desprotegidos y con diferentes tensiones.
- Número limitado de interruptores disponibles sin posibilidad de escalado.
- En proyectos con orientación comercial el montaje, mantenimiento y reparación son más complicados.
- Mayor dificultad de desarrollo, depuración y pruebas en entornos PC/Mac al no disponer estos de los elementos hardware (bus GPIO).

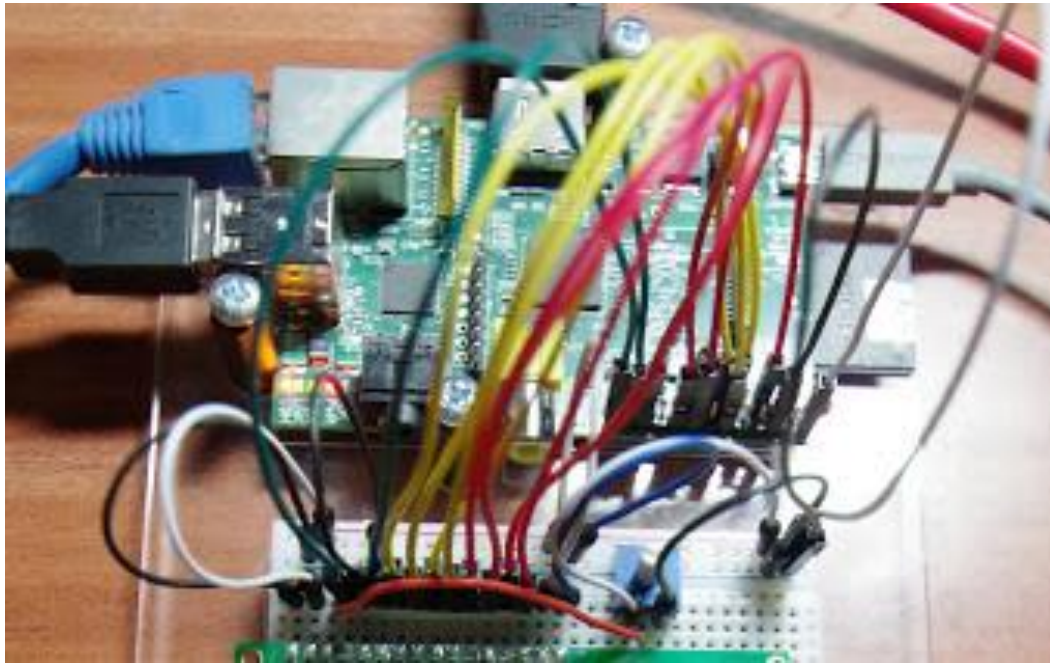


Ilustración 2. Conexiones en conector GPIO de una Raspeberri PI haciendo uso de una breadboard.

Estas dificultades son las que motivaron la búsqueda de una solución que permitiera dotar de entrada de información binaria de forma más flexible y con bajo coste a los sistemas basados en SBC. La solución que se ha encontrado como más conveniente es el empleo de bancos de interruptores conectables vía USB. Un ejemplo de esta aproximación es el teclado de un ordenador con conexión USB.

Otro ejemplo que para el tipo de aplicaciones es aún más conveniente para este tipo de aplicaciones es el empleo de un dispositivo tipo joystick. Entre estos dispositivos destaca por su bajo coste, ubicuidad y estandarización, las tarjetas "Generic USB joystick" con controlador DragonRise. El empleo de tarjetas USB de joystick aporta la ventaja en sistemas empotrados o empotrables tipo Linux (por ejemplo Raspbian) la disponibilidad de controladores de dispositivo que facilitan notablemente la integración del conjunto de interruptores en sistema.

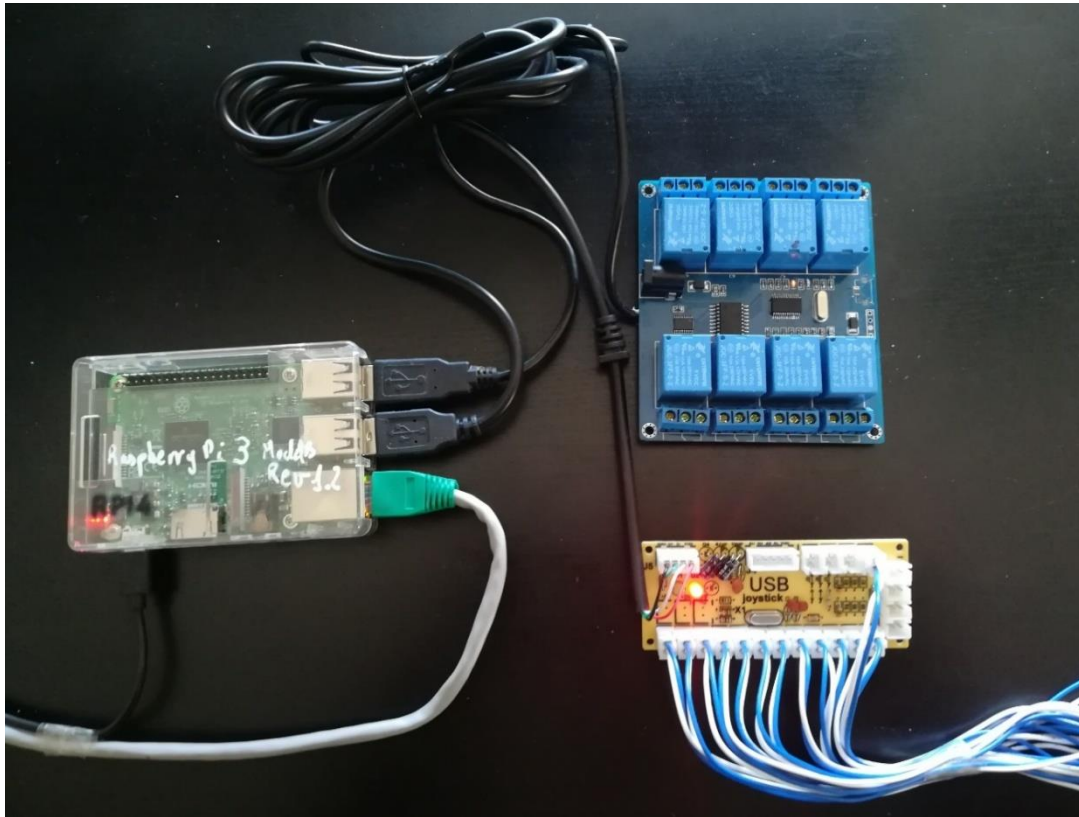


Ilustración 3. Tarjeta DragonRise con 13 conexiones para interruptores conectada a un puerto USB de una Raspberry Pi. La imagen también muestra una placa de 8 relés conectada a otro puerto. La foto permite apreciar la mejor estructuración y limpieza de las conexiones.

El sistema así concebido es fácilmente escalable haciendo uso de más tarjetas USB y, en caso necesario, de concentradores (hubs) USB.

No todo son ventajas, la introducción de la tarjeta aumenta la complejidad del sistema por lo que el tiempo confirmará si la fiabilidad del sistema queda comprometida. Por lado la tarjeta provocará un mayor consumo en el bus USB que puede afectar al funcionamiento del conjunto de dispositivos conectados a éste bus. Este punto puede ser parcialmente aliviado eliminando el led rojo que permanente está encendido en la tarjeta o, de forma definitiva, , haciendo uso de concentradores USB con alimentación propia pese a aumentar la complejidad del conjunto.

Descripción de la placa DragonRise

La placa a emplear se muestra en la figura

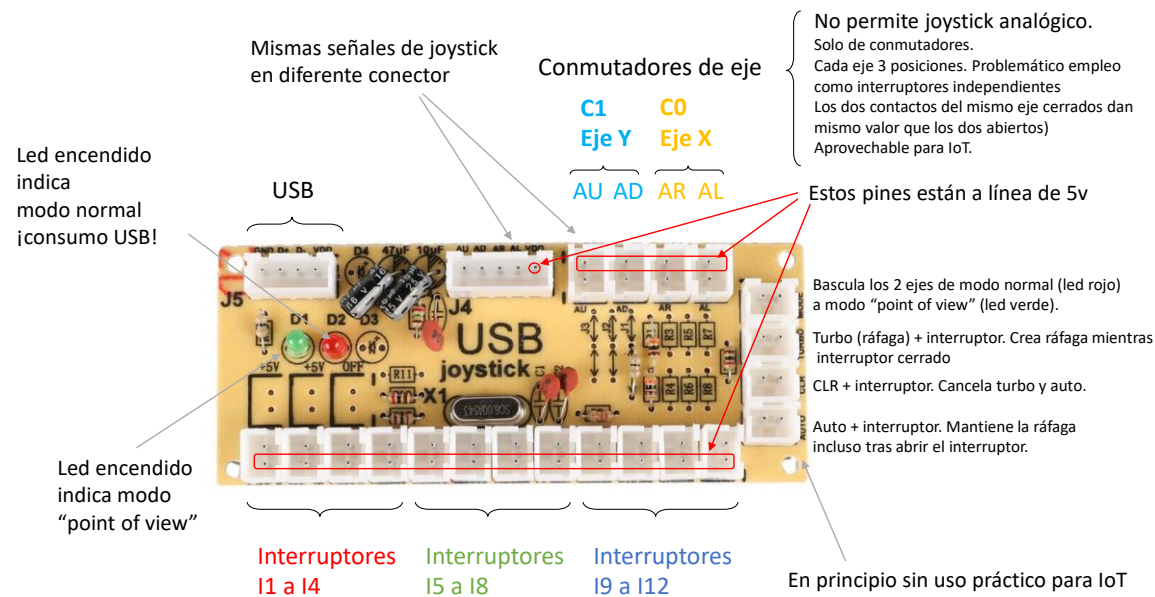


Ilustración 4. Tarjeta DragonRise con comentarios de los puntos más relevantes

¡Ojo!. Al contrario que la mayoría de tarjetas, la tarjeta Dragonrise destina la mayor (y más expuesta) parte la superficie de cobre a la línea de potencial de 5v en lugar de a GND o tierra (0v).

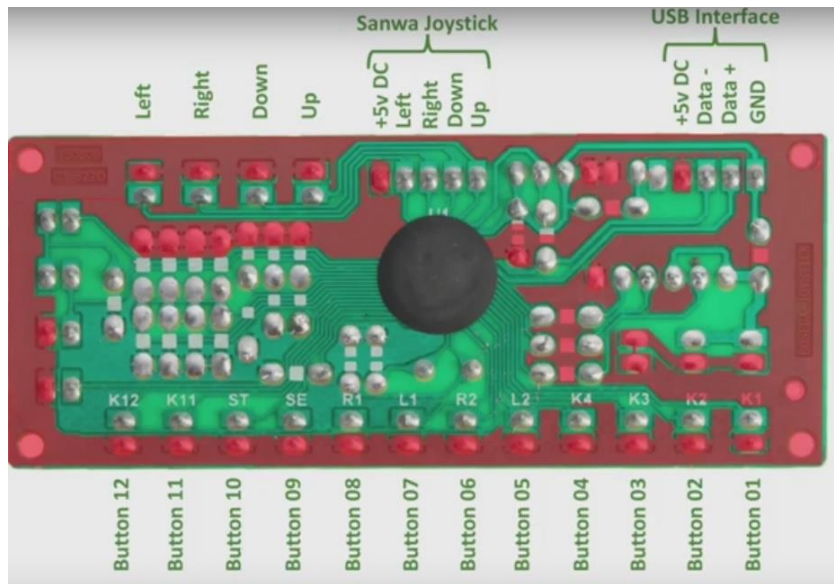


Ilustración 5. Reverso de la tarjeta DragonRise donde se muestra que la superficie más extensa no es tierra-común sino la línea de tensión de 5 voltios.

Esto implica que, en general, solo se podrán conectar interruptores y conmutadores “flotantes” que no tengan tensión alguna en ninguno de sus terminales.

Más información en <https://www.s-config.com/zero-delay-usb-joystick-encoder/>

Programa dragonrise

El programa para Linux objeto de este proyecto se puede descargar en <https://github.com/junavarg/dragonrise>.

Devuelve por stdout una estructura JSON con información de eventos y estado de interruptores y ejes(conmutadores) de la tarjeta 'Generic USB joystick' de DragonRise para uso en IoT.

También permite la publicación en un MQTT broker esta información.

Use: dragonrise [options] [device_file1] [device_file2]... ")

Opciones:

-mqpub <url>

Especifica la URL de broker MQTT y la raíz de un topic (basetopic) donde publicar el estado cada vez que se produzca un evento. El formato de url es

protocol://[user[:password]@]host.dominio.tld:puerto/base_topic

Opciones para protocol: tcp, ssl,ws,wss

Ejemplos:

-mqpub=tcp://host.dominio.dom:1883/base_topic

-mqpub=ssl://pepe@host.dominio.dom:8883/base_topic

-mqpub=ws://host.dominio.dom:80/base_topic

-mqpub=wss://pepe:p2ssw0d@host.dominio.dom:443/base_topic

-mqpub2

-mqpub3

Broker adicionales a los que el programa envía eventos

Los mensajes se publican en 'clean session' con qos 0 y con 'retained flag' para que en cada nueva conexión el subcriptor reciba un mensaje con el estado actual.

-cbc

Check Broquer Certificate. Habilita que se verifique el certificado presentado por el broker MQTT en los protocolos protegidos por TLS así como la cadena de certificación hasta el root certificate

Detalles técnicos

El programa está realizado en Golang y se puede generar código ejecutable para x86 y arm6, arm7.

Estas primeras lecturas tras la apertura del fichero consisten en 12 lecturas de interruptores seguidas de 7 lecturas ejes/conmutadores (total 19 lecturas) para anotar internamente el estado actual. Cada lectura es de 8 bytes con una estructura idéntica (ver apartado [Anexo I. Interfaz Linux para joystick /dev/input/js0](#)). Sin embargo la tarjeta empleada solo facilita conexiones físicas para 2 ejes/conmutadores (sanwa joystick) por lo que sólo tendrán información los números 0 y 1.

Se permite que un mismo proceso pueda atender a más de una tarjeta Dragonrise.

El programa elimina los eventos espurios que aleatoriamente se generan para ejes/conmutadores que no están físicamente disponibles en la tarjeta.

El mensaje JSON que se saca por stdout tiene el siguiente formato:

```
{ "time":<unix_time>,  
  "device": "<devicefile>", "event": { "type":<t>, "sensor":<s>, "value":<v> }, "switches": [v,v,v,v,v,v,v,v,v,v,v,v], "axes": [v,v,0,0,0,0,0,0] }
```

Donde te puede tener valores: 0: Valor inicial (no evento real), 1: evento de interruptor, 2: evento de eje/conmutador, -1: Error

Esta misma información es la que se guarda en un fichero como estado actual en el fichero '*<device_file>.dat*' en */var/lib/dragonrise/*.

La siguiente tabla muestra diferentes mensajes por stdout en función de la situación reportada al ejecutar

```
dragonrise /dev/input/js0 2>/dev/null
```

Situación	Salida JSON por stdout
Arranque de programa con una tarjeta conectada al bus USB y asociada con dispositivo de fichero "js0"	<pre>{ "time":1590225475,"device":"js0", "event":{"type":0,"sensor":0,"value":0}, "switches":[1,0,0,0,0,0,0,0,0,0,0,0], "axes": [0,0,0,0,0,0,0,0] }</pre>
Evento en interruptores en una tarjeta conectada al bus USB y asociada con dispositivo de fichero "js0"	<pre>{ "time":1590225866,"device":"js0", "event":{"type":1,"sensor":0,"value":0}, "switches": [0,0,0,0,0,0,0,0,0,0,0,0], "axes": [0,0,0,0,0,0,0,0] }</pre>
Evento en eje/conmutador en una tarjeta conectada al	<pre>{ "time":1590226116,"device":"js0",</pre>


```

maxTarjetas = 10 //número máximo de tarjetas (files devices) que se almacenaran en un array
const max_swt = 12 //número máximo de interruptores
const max_com = 7 //número máximo de ejes-conmutadores

//variables globales

var (
    numTarjetas int //numero de tarjetas gestionadas por el proceso dragonrise. Siempre numTarjetas <=
maxTarjetas
    tarjeta [maxTarjetas]dragonrise // array de estructuras de ultimo evento y estados
    switches [maxTarjetas][maxSwt]int16 // array de interruptores
    conmutadores [maxTarjetas][maxCom]int16 // array de ejes/conmutadores
    fDispositivo [maxTarjetas]string //nombre de fichero de dispositivo
    hDispositivo [maxTarjetas]*os.File //handle de fichero de dispositivo
    fEstado [maxTarjetas]string //nombre de fichero de estado
    hEstado [maxTarjetas]*os.File //handle de fichero de estado
)

```

Se permite que un mismo proceso pueda atender a más de una tarjeta Dragonrise. La estructura arriba descrita realmente es un array indizada por número de tarjeta.

Conexión y publicación en broker MQTT

La documentación de la librería MQTT de golang empleada se puede consultar en <https://godoc.org/github.com/eclipse/paho.mqtt.golang> .

La conexión al broker MQTT se realiza con un “ClientID” que se genera con la MAC del primer adaptador de red enumerado en el sistema al que se le añade, separados con un guion el nombre del device_file especificado en la línea de comando.

Los mensajes se publican en 'clean session' con qos 0 y con 'retained flag' para que en cada nueva conexión el subcriptor reciba un mensaje con el estado actual.

EL “topic” de subscripción de la cola se obtiene componiendo

- el elemento base_topic especificado en la url de la opción -mqbub con

- un carácter “/”
- el device_file especificado en la línea de comando correspondiente la tarjeta.
- la cadena “/event”

Se permite hasta 2 brokers adicionales (opciones -mqpub2 y -mqpub3). El topic base es el mismo que el especificado en el primer broker (opción -mqpub).

Cada conexión a un broker corresponde un cliente MQTT. El programa emplea un ClientID que es un hash MD5 truncado a 10 caracteres hexadecimales de la concatenación de la dirección MAC , el device file y el índice del cliente MQTT (empezando en 0).

La siguiente tabla muestra diferentes mensajes recibidos en el topic del cliente subscriptor en función de la situación reportada al ejecutar:

```
dragonrise -mqpub tcp://test.mosquitto.org:1883/topic12345 /dev/input/js0 2>/dev/null
```

Situación	Mensaje JSON recibido en topic cliente subscriptor
Conexión inicial al topic del broker	<pre>{ "time":1590231059,"device":"js0", "event":{"type":1,"sensor":0,"value":1}, "switches":[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],"axes":[0,0,0,0,0,0,0,0]}</pre> <p>El broker transmite el último mensaje recibido por el broker desde el publicador (emitido con retention=true). Sirve sensores de la tarjeta.</p>
Evento en interruptores en una tarjeta conectada al bus USB y asociada con dispositivo de fichero “js0”	<pre>{ "time":1590231550,"device":"js0", "event":{"type":1,"sensor":0,"value":0}, "switches":[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],"axes":[0,0,0,0,0,0,0,0]}</pre>
Desconexión de la tarjeta del bus USB y asociada con dispositivo de fichero “js0”	<pre>{ "time":1590231778,"device":"js0", "event":{"type":-1,"reason":"Dispositivo dejó de estar disponible"}}</pre>
Conexión de la tarjeta al bus USB y asociada con dispositivo de fichero “js0”	<pre>{ "time":1590231857,"device":"js0", "event":{"type":0,"sensor":0,"value":0}, "switches":[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],"axes":[0,0,0,0,0,0,0,0]}</pre>

Parada de programa dragonrise	<pre>{ "time":0, "device": "js0", "event": { "type": -1, "reason": "Mensaje LWT emitido por broker. No disponible cliente MQTT que pu dispositivo" }}</pre> <p>Broker detecta la pérdida de conexión y envía mensaje LWT a subscriptores</p>
Arranque de programa con una tarjeta conectada al bus USB y asociada con dispositivo de fichero "js0"	<pre>{ "time":1590232743, "device": "js0", "event": { "type": 0, "sensor": 0, "value": 0 }, "switches": [1,0,0,0,0,0,0,0,0,0,0,0], "axes": [0,0,0,0,0,0,0] }</pre>
Caída de broker	(no llega mensaje)
Recuperación de broker	<pre>{ "time":1590250662, "device": "js0", "event": { "type": 1, "sensor": 0, "value": 1 }, "switches": [1,0,0,0,0,0,0,0,0,0,0,0], "axes": [0,0,0,0,0,0,0] } (retained, enviado por broker)</pre> <p>..... pasado un tiempo ...</p> <pre>{ "time":1590250996, "device": "js0", "event": { "type": 1, "sensor": 0, "value": 0 }, "switches": [0,0,0,0,0,0,0,0,0,0,0,0], "axes": [0,0,0,0,0,0,0] } (enviado por dragonrise con último de es</pre>

Con respecto los dos últimos casos:

- 1) El subscriptor (mosquito_sub) no se entera de que el broker se ha caído. ¿Es esto siempre así? ¿Formas de enterarse de la caída de la comunicación?
- 2) Cuando subscriptor reconecta recibe un mensaje "retained" (el último recibido por el broker). Aunque parece un evento normal, el subscriptor puede darse cuenta de que es un mensaje viejo por el valor tiempo.
- 3) ¿Alguna formas de saber que es "retained"?
- 4) Más tarde , cuando el publicador logra reconectar, este envía un mensaje con el último mensaje almacenado en su fichero de estado, posiblemente con un estado diferente al del mensaje "retained" recibido anteriormente.
Desde el punto de vista del subscriptor, es un evento de interruptor (tipo 1) pero con el reloj retrasado.
- 5) Formalmente, ¿No sería mejor que el publicador mandase (¿además inmediata y posteriormente?) un mensaje de inicialización con tiempo y estado actual?

Funcionamiento con brokers MQTT de pruebas

Se han realizado pruebas con diferentes brokers públicos de pruebas con los siguientes resultados:

Server	Broker	Transporte	Ports	Ok
test.mosquitto.org	Mosquitto	tcp	1883	Ok
		ssl	8883	Ok
		ssl ¹	8884	NP ²
		ws	8080	Ok
		wss	8081	Ok
mqtt.eclipse.org		tcp	1883	Ok
tcp://broker.emqx.io		tcp	1883	OK
broker.hivemq.com	HiveMQ	tcp	1883	OK
		ssl	8000	KO ³
broker.mqttdashboard.com ³	HiveMQ	tcp	1883	OK
mqtt.flespi.io user: (Flespi token) umHbtbcvEQclGOglT0mvxn8j1kB4ubKNofMFgO1Ls3U YF07XOIctglkpKdHKNkY sin password	flsepi	tcp	1883	Ok
		Ssl	883	Ok
		ws	80	Ok
		wss	443	OK

test.mosca.io	mosca	tcp	1883	ND ⁴
		ws	80	ND ⁴

NOTAS:

- 1) Autenticación con certificado de cliente
- 2) NP: No probado
- 3) Network Error : EOF
- 4) ND: No disponible

Reglas UDEV para DragonRise

Con el fichero de reglas se pretende estabilizar el nombre del fichero de dispositivo en función de la posición en el bus USB. Esto es especialmente importante en el caso de que se conecten al mismo bus más de una tarjeta de tipo joystick que provoque la aparición de varios ficheros de dispositivos `/dev/input/js<n>` sin que se pueda garantizar una correspondencia fija entre fichero de dispositivo y tarjeta.

Instalación manual de reglas

Se edita el fichero de reglas *31-dragonrise.rules*

```
sudo nano /etc/udev/rules.d/31-dragonrise.rules
```

con el siguiente contenido

```
# Fichero de reglas UDEV adecuadas pra Raspberry Pi
# para invocar inicializacion de tarjeta "Generic USB Joystick" de DragonRise el momento de su conexion/detección
# Autor Junav version 1 26/04/2020
# Este fichero debe ubicarse en /etc/udev/rules.d/

# La utilidad
# udevadm info -a -n /dev/input/js0
# reflejará
# KERNELS=="1-1.2" para placa DragonRise conectada directamente a puerto USB externo 1 de RPI (2 en el bus)
# KERNELS=="1-1.3" para placa DragonRise conectada directamente a puerto USB externo 2 de RPI (3 en el bus)
```

```
# Se hace notar que en una RPi el puerto USB 1 está empleado internamente por la tarjeta de red.
#
# Si se emplea un HUB, en lugar de una conexión directa, el tercer nivel de información despues del inicial "KERNEL" daría
# KERNELS=="1-1.3.1" para placa conectada a puerto USB 1 del HUB en puerto USB externo 2 de RPI
# KERNELS=="1-1.3.3" para placa conectada a puerto USB 3 del HUB en puerto USB externo 2 de RPI

# En su caso ajustar estas reglas según sea preciso

# Deteccion de chip de DragonRise idVendor=0079 idProduct=0006 asignando un symlink fijo segun numeracion bus USB, ejecucin de un programa
con parametro de numeracion
ACTION=="add", KERNEL=="js*", KERNELS=="1-1.2", ATTRS{idVendor}=="0079", ATTRS{idProduct}=="0006", SYMLINK+="dragonrise_2"
ACTION=="add", KERNEL=="js*", KERNELS=="1-1.3", ATTRS{idVendor}=="0079", ATTRS{idProduct}=="0006", SYMLINK+="dragonrise_3"
ACTION=="add", KERNEL=="js*", KERNELS=="1-1.4", ATTRS{idVendor}=="0079", ATTRS{idProduct}=="0006", SYMLINK+="dragonrise_4"
ACTION=="add", KERNEL=="js*", KERNELS=="1-1.5", ATTRS{idVendor}=="0079", ATTRS{idProduct}=="0006", SYMLINK+="dragonrise_5"

#Si se quiere ejecutar automaticamente el programa dragonrise (previamente ubicado en /usr/local/bin) se puese hacer aqui. Por ejemplo
#ACTION=="add", KERNEL=="js*", KERNELS=="1-1.5", ATTRS{idVendor}=="0079", ATTRS{idProduct}=="0006", SYMLINK+="dragonrise_5",
RUN+="/usr/local/bin/dragonrise -mqpub=tcp://host.dominio.dom:1883/topic_base /dev/dragonrise_5"

#Deteccion desenchufe en puerto USB de placa dragonrise y borrado de fichero de estado del programa dragonrise
ACTION=="remove", ENV{ID_VENDOR_ID}=="0079", ENV{ID_MODEL_ID}=="0006", ENV{DEVPATH}=="/devices/platform/soc/3f980000.usb/usb1/1-1/1-1.2",
RUN+="/bin/rm /var/lib/dragonrise/dragonrise_2.dat"
ACTION=="remove", ENV{ID_VENDOR_ID}=="0079", ENV{ID_MODEL_ID}=="0006", ENV{DEVPATH}=="/devices/platform/soc/3f980000.usb/usb1/1-1/1-1.3",
RUN+="/bin/rm /var/lib/dragonrise/dragonrise_3.dat"
ACTION=="remove", ENV{ID_VENDOR_ID}=="0079", ENV{ID_MODEL_ID}=="0006", ENV{DEVPATH}=="/devices/platform/soc/3f980000.usb/usb1/1-1/1-1.4",
RUN+="/bin/rm /var/lib/dragonrise/dragonrise_4.dat"
ACTION=="remove", ENV{ID_VENDOR_ID}=="0079", ENV{ID_MODEL_ID}=="0006", ENV{DEVPATH}=="/devices/platform/soc/3f980000.usb/usb1/1-1/1-1.5",
RUN+="/bin/rm /var/lib/dragonrise/dragonrise_5.dat"
```

Para asegurar la recarga de la reglas.

```
$ sudo udevadm control --reload-rules
```

En caso de que se necesite editar las reglas para adaptarse a un caso concreto (por ejemplo con presencia de concentradores-hubs USB), se puede consultar el Anexo I de este documento.

Anexo I. Interfaz Linux para joystick */dev/input/js0*

En Linux, la tarjeta DragonRise se identifica en el bus USB como:

```
$ lsusb
...
Bus 001 Device 008: ID 0079:0006 DragonRise Inc. PC TWIN SHOCK Gamepad
...
```

El subdirectorio */dev/input* contiene los ficheros de dispositivos de como teclados, ratones, tabletas, joystick, etc.

El fichero */proc/bus/input/devices* contiene información relevante de estos periféricos de entrada y su correspondencia con los ficheros de dispositivos.

```
cat /proc/bus/input/devices
```

Más información en: <https://thehackerdiary.wordpress.com/2017/04/21/exploring-devinput-1/>

Los fichero de dispositivo con el formato *jsN* (por ejemplo */dev/input/js0*) están sujetos a la especificación Joystick API de Linux que se puede consultar en <https://www.kernel.org/doc/Documentation/input/joystick-api.txt>.

La lectura de este fichero de dispositivo se hace por bloques de 8 bytes que corresponde a un evento. Esta lectura queda bloqueada a la espera de un evento (cambio en un interruptor).

NO se precisa privilegios de root.

time				value		event type	axis/button number
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7

Time corresponde a un timestamp en milisegundos desde “algún momento pasado”

Tipo de evento señala si el evento corresponde a un botón (interruptor) o en un eje de joystick:

0x01 -> botón

0x02 -> eje

Tras hacer el open del dispositivo las primeras lecturas corresponden a los estados actuales de todos los dispositivos. NO se trata de eventos reales sino “sintéticos”. Para diferenciarlos de los reales el valor del campo tipo de evento se combina en unión lógica con el valor 0x80, dando los valores

0x81 -> Estado inicial de botón

0x82 -> Estado inicial de eje

Numero de eje o número de botón. Identifica el botón (interruptor) o eje del evento. Botones y ejes llevan numeradores de identificación separados iniciados en 0.

Valor. Es el valor comunicado del botón o eje. Los dos bytes están en “little endian”.

Para botones:

0x0000 si abierto

0x0001 si cerrado

Para ejes/conmutadores:

0x0000 si centrado. En el caso de ejes implementado con conmutadores (no analógicos) ambos conmutadores de cada semieje abiertos (o ambos cerrados)

0x7FFF si máximo positivo (decimal 32767). En el caso de ejes implementado con conmutadores (no analógicos) conmutador de semieje positivo cerrado y conmutador de semieje negativo abierto.

0x8001 si máximo negativo (decimal -32767). En el caso de ejes implementado con conmutadores (no analógicos), conmutador de semieje negativo cerrado y conmutador de semieje positivo abierto.

En el caso de la tarjeta DragonRise, el driver reconoce 12 interruptores y 7 ejes/conmutadores.

En el modo normal se tiene visibilidad del eje 0 (eje X, AL-AR) y del eje 1 (eje Y AU-AD).

En el modo alternativo “point of view” estos conectores se expresan en los ejes 5 y 6 respectivamente.

El interruptor AU (Up) cerrado corresponde al máximo ¡¡negativo!! del eje Y.

El interruptor AD (Down) cerrado corresponde al máximo ¡¡positivo!! del eje Y.

El interruptor AR (Right) cerrado corresponde al máximo positivo del eje X.

El interruptor AL (Left) cerrado corresponde al máximo negativo del eje X.

NOTA. Se ha observado que, de forma aleatoria, la tarjeta deja escapar falsos eventos del inexistente eje 2 a valor 0.

Anexo II. Procedimiento de descubrimiento para construcción de reglas UDEV

El procedimiento es el siguiente:

Se conecta la tarjeta DragonRise en el puerto USB exterior número 3.

Se verifica la existencia de fichero de dispositivo */dev/input/js0* .

Se ejecuta el comando siguiente. Se señala en rojo la info relevante para la construcción de la regla UDEV

```
$ udevadm info -a -n /dev/input/js0
```

```
Udevadm info starts with the device specified by the devpath and then
walks up the chain of parent devices. It prints for every device
found, all possible attributes in the udev rules key format.
A rule to match, can be composed by the attributes of the device
and the attributes from one single parent device.
```

```
looking at device '/devices/platform/soc/3f980000.usb/usb1/1-1/1-1.4/1-1.4:1.0/0003:0079:0006.0008/input/input7/js0'
  KERNEL=="js0"
  SUBSYSTEM=="input"
  DRIVER==""
```

```
looking at parent device '/devices/platform/soc/3f980000.usb/usb1/1-1/1-1.4/1-1.4:1.0/0003:0079:0006.0008/input/input7'
  KERNELS=="input7"
  SUBSYSTEMS=="input"
  DRIVERS==""
```

```
ATTRS{name}=="DragonRise Inc. Generic USB Joystick "  
ATTRS{phys}=="usb-3f980000.usb-1.4/input0"  
ATTRS{properties}=="0"  
ATTRS{uniq}==""
```

looking at parent device '/devices/platform/soc/3f980000.usb/usb1/1-1/1-1.4/1-1.4:1.0/0003:0079:0006.0008':

```
KERNELS=="0003:0079:0006.0008"  
SUBSYSTEMS=="hid"  
DRIVERS=="dragonrise"  
ATTRS{country}=="21"
```

looking at parent device '/devices/platform/soc/3f980000.usb/usb1/1-1/1-1.4/1-1.4:1.0':

```
KERNELS=="1-1.4:1.0"  
SUBSYSTEMS=="usb"  
DRIVERS=="usbhid"  
ATTRS{authorized}=="1"  
ATTRS{bAlternateSetting}==" 0"  
ATTRS{bInterfaceClass}=="03"  
ATTRS{bInterfaceNumber}=="00"  
ATTRS{bInterfaceProtocol}=="00"  
ATTRS{bInterfaceSubClass}=="00"  
ATTRS{bNumEndpoints}=="02"  
ATTRS{supports_autosuspend}=="1"
```

looking at parent device '/devices/platform/soc/3f980000.usb/usb1/1-1/1-1.4':

```
KERNELS=="1-1.4"  
SUBSYSTEMS=="usb"  
DRIVERS=="usb"  
ATTRS{authorized}=="1"  
ATTRS{avoid_reset_quirk}=="0"  
ATTRS{bConfigurationValue}=="1"  
ATTRS{bDeviceClass}=="00"  
ATTRS{bDeviceProtocol}=="00"  
ATTRS{bDeviceSubClass}=="00"  
ATTRS{bMaxPacketSize0}=="8"  
ATTRS{bMaxPower}=="500mA"  
ATTRS{bNumConfigurations}=="1"  
ATTRS{bNumInterfaces}==" 1"  
ATTRS{bcdDevice}=="0107"  
ATTRS{bmAttributes}=="80"  
ATTRS{busnum}=="1"
```

```
ATTRS{configuration}=="  
ATTRS{devnum}=="20"  
ATTRS{devpath}=="1.4"  
ATTRS{devspec}==" (null)"  
ATTRS{idProduct}=="0006"  
ATTRS{idVendor}=="0079"  
ATTRS{ltm_capable}=="no"  
ATTRS{manufacturer}=="DragonRise Inc. "  
ATTRS{maxchild}=="0"  
ATTRS{product}=="Generic USB Joystick "  
ATTRS{quirks}=="0x0"  
ATTRS{removable}=="removable"  
ATTRS{speed}=="1.5"  
ATTRS{urbnum}=="28247844"  
ATTRS{version}==" 1.00"
```

looking at parent device '/devices/platform/soc/3f980000.usb/usb1/1-1':

```
KERNELS=="1-1"  
SUBSYSTEMS=="usb"  
DRIVERS=="usb"  
ATTRS{authorized}=="1"  
ATTRS{avoid_reset_quirk}=="0"  
ATTRS{bConfigurationValue}=="1"  
ATTRS{bDeviceClass}=="09"  
ATTRS{bDeviceProtocol}=="02"  
ATTRS{bDeviceSubClass}=="00"  
ATTRS{bMaxPacketSize0}=="64"  
ATTRS{bMaxPower}=="2mA"  
ATTRS{bNumConfigurations}=="1"  
ATTRS{bNumInterfaces}==" 1"  
ATTRS{bcdDevice}=="0200"  
ATTRS{bmAttributes}=="e0"  
ATTRS{busnum}=="1"  
ATTRS{configuration}=="  
ATTRS{devnum}=="2"  
ATTRS{devpath}=="1"  
ATTRS{idProduct}=="9514"  
ATTRS{idVendor}=="0424"  
ATTRS{ltm_capable}=="no"  
ATTRS{maxchild}=="5"  
ATTRS{quirks}=="0x0"
```

```
ATTRS{removable}=="unknown"  
ATTRS{speed}=="480"  
ATTRS{urbnum}=="549"  
ATTRS{version}==" 2.00"
```

looking at parent device '/devices/platform/soc/3f980000.usb/usb1':

```
KERNELS=="usb1"  
SUBSYSTEMS=="usb"  
DRIVERS=="usb"  
ATTRS{authorized}=="1"  
ATTRS{authorized_default}=="1"  
ATTRS{avoid_reset_quirk}=="0"  
ATTRS{bConfigurationValue}=="1"  
ATTRS{bDeviceClass}=="09"  
ATTRS{bDeviceProtocol}=="01"  
ATTRS{bDeviceSubClass}=="00"  
ATTRS{bMaxPacketSize0}=="64"  
ATTRS{bMaxPower}=="0mA"  
ATTRS{bNumConfigurations}=="1"  
ATTRS{bNumInterfaces}==" 1"  
ATTRS{bcdDevice}=="0414"  
ATTRS{bmAttributes}=="e0"  
ATTRS{busnum}=="1"  
ATTRS{configuration}==" "  
ATTRS{devnum}=="1"  
ATTRS{devpath}=="0"  
ATTRS{idProduct}=="0002"  
ATTRS{idVendor}=="1d6b"  
ATTRS{interface_authorized_default}=="1"  
ATTRS{ltm_capable}=="no"  
ATTRS{manufacturer}=="Linux 4.14.50-v7+ dwc_otg_hcd"  
ATTRS{maxchild}=="1"  
ATTRS{product}=="DWC OTG Controller"  
ATTRS{quirks}=="0x0"  
ATTRS{removable}=="unknown"  
ATTRS{serial}=="3f980000.usb"  
ATTRS{speed}=="480"  
ATTRS{urbnum}=="25"  
ATTRS{version}==" 2.00"
```

looking at parent device '/devices/platform/soc/3f980000.usb':

```
KERNELS=="3f980000.usb"
SUBSYSTEMS=="platform"
DRIVERS=="dwc_otg"
ATTRS{busconnected}=="Bus Connected = 0x1"
ATTRS{buspower}=="Bus Power = 0x1"
ATTRS{bussuspend}=="Bus Suspend = 0x0"
ATTRS{devspeed}=="Device Speed = 0x0"
ATTRS{driver_override}==(null)"
ATTRS{enumspeed}=="Device Enumeration Speed = 0x1"
ATTRS{fr_interval}=="Frame Interval = 0x1d4b"
ATTRS{ggpio}=="GGPIO = 0x00000000"
ATTRS{gnptxfsize}=="GNPTXFSIZ = 0x01000306"
ATTRS{gotgctl}=="GOTGCTL = 0x001c0001"
ATTRS{gpvndctl}=="GPVNDCTL = 0x00000000"
ATTRS{grxfsize}=="GRXFSIZ = 0x00000306"
ATTRS{gsnpsid}=="GSNPSID = 0x4f54280a"
ATTRS{guid}=="GUID = 0x2708a000"
ATTRS{gusbcfg}=="GUSBCFG = 0x20001700"
ATTRS{hcd_frrem}=="HCD Dump Frame Remaining"
ATTRS{hcddump}=="HCD Dump"
ATTRS{hnp}=="HstNegScs = 0x0"
ATTRS{hnp capable}=="HNPCapable = 0x1"
ATTRS{hprt0}=="HPRT0 = 0x00001405"
ATTRS{hptxfsize}=="HPTXFSIZ = 0x02000406"
ATTRS{hsic_connect}=="HSIC Connect = 0x1"
ATTRS{inv_sel_hsic}=="Invert Select HSIC = 0x0"
ATTRS{mode}=="Mode = 0x1"
ATTRS{mode_ch_tim_en}=="Mode Change Ready Timer Enable = 0x0"
ATTRS{rd_reg_test}=="Time to read GNPTXFSIZ reg 10000000 times: 940 msecs (94 jiffies)"
ATTRS{regdump}=="Register Dump"
ATTRS{regoffset}=="0xffffffff"
ATTRS{regvalue}=="invalid offset"
ATTRS{rem_wakeup_pwrtn}=="
ATTRS{remote_wakeup}=="Remote Wakeup Sig = 0 Enabled = 0 LPM Remote Wakeup = 0"
ATTRS{spramdump}=="SPRAM Dump"
ATTRS{srp}=="SesReqScs = 0x1"
ATTRS{srp capable}=="SRPCapable = 0x1"
ATTRS{wr_reg_test}=="Time to write GNPTXFSIZ reg 10000000 times: 350 msecs (35 jiffies)"
```

looking at parent device '/devices/platform/soc':

```
KERNELS=="soc"
```

```
SUBSYSTEMS=="platform"
DRIVERS=="
ATTRS{driver_override}==(null)"

looking at parent device '/devices/platform':
KERNELS=="platform"
SUBSYSTEMS=="
DRIVERS=="
```

Se construye un fichero de reglas en

```
$ sudo nano /etc/udev/rules.d/31-dragonrise.rules
...
# Deteccion de chip de Dragonrise idVendor=0079 idProduct=0006 asignando un symlink fijo segun numeracion bus USB, ejecucin de un programa
con parametro de numeracion
ACTION=="add", KERNEL=="js*", KERNELS=="1-1.4", ATTRS{idVendor}=="0079", ATTRS{idProduct}=="0006", SYMLINK+="dragonrise_4",
RUN+="/usr/local/bin/dragonrise 4"
```

Para asegurar la recarga de la reglas (no es necesario si se reeditan).

```
$ sudo udevadm control --reload-rules
```

Se inserta en el puerto USB exterior 3 (4 en el bus) la tarjeta DragonRise y se comprueba la existencia del enlace simbólico.

```
ls -l /dev
...
lrwxrwxrwx 1 root root          9 abr 26 11:21 dragonrise_4 -> input/js0
...
```


Anexo III. Interfaz software con `/dev/hdiraw0`

Otra posibilidad de obtener el estado de los interruptores conectados a la tarjeta DragonRise es el empleo del fichero de dispositivo `/dev/hdiraw0`. De esto este fue el primer intento pero se descartó por encontrar más conveniente el interfaz proporcionado por `/dev/input/js0`.

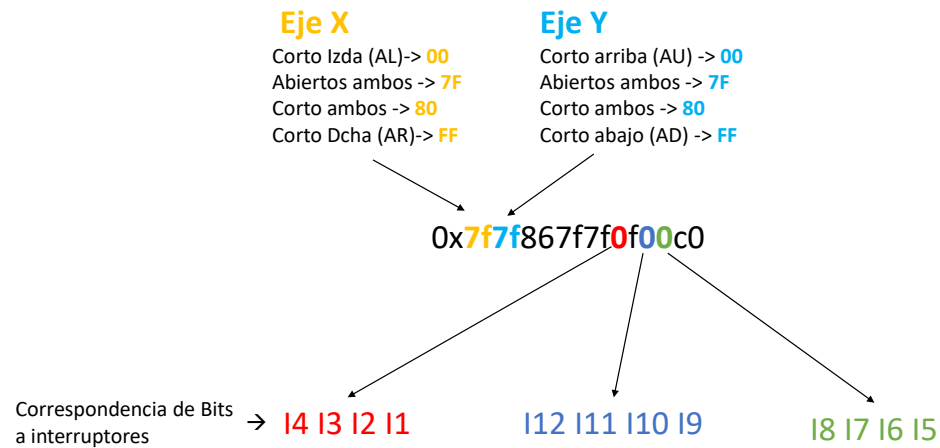
A la detección el sistema crea un fichero de dispositivo `/dev/hidrawN` (Human Interface Device RAW), donde “N” típicamente será “0” si no hay otros dispositivos.

```
$ ls -l /dev/hid*  
crw----- 1 root root 247, 0 abr 19 06:40 /dev/hidraw0
```

Obsérvese que solo permite leer y escribir a root.

<https://www.kernel.org/doc/Documentation/hid/hidraw.txt>

Mediante un programa que, con privilegios de root, lea constantemente 8 bytes de este dispositivo y los exprese en consola en hexadecimal se obtiene algo similar a:



Así por ejemplo:

si se mantiene cortocircuitado solo el I1, se obtiene: 0x7f7f867f7f1f00c0

si se mantiene cortocircuitado solo el I8, se obtiene: 0x7f7f867f7f0f08c0

si se mantiene cortocircuitado solo el I11, se obtiene: 0x7f7f867f7f0f40c0

El empleo de este fichero dispositivo para la lectura de los eventos tropieza con los siguientes inconvenientes:

- No parece un sistema standard. Otros dispositivos similares pueden tener una interpretación diferente de los bytes leídos.
- La lectura no se bloquea a la espera de eventos. Se producen múltiples lecturas que hay que ignorar a la espera de un cambio que signifique un evento. Esto supone un gasto de CPU (en torno a 2%) excesivo para esta función.

- El proceso de lectura hay que hacerlo con privilegios de root.