Informática

Programando el controlador TX con Python

Raphael jacob

El controlador TX salió al mercado en el verano de 2009. En Windows, esto es muy fácil de controlar con ROBOPro, pero si un programa se ejecuta en Linux o incluso en otra arquitectura, como ARM (controlador TXT), rápidamente alcanza los límites. Pero el problema se puede resolver, porque el controlador TX también se puede controlar con Python.

Definición del problema

Todas las interfaces delante del controlador TX (en adelante, TX-C) y el controlador TXT se pueden controlar en todas las plataformas con Python, pero el TX-C puede
- al menos "por defecto" - contrólelo solo bajo Windows con ROBOPro o usando una biblioteca ya compilada, por ejemplo con un programa en C. ¿No debería haber una forma alternativa de controlar el TX directamente con Python?

idea

Así que comencé a investigar qué información sobre el TX es proporcionada por fischertechnik o ha sido descubierta por los fanáticos. fischertechnik proporciona los archivos de encabezado e instrucciones, así como una descripción del sistema operativo para la biblioteca mencionada anteriormente en su página de inicio [1].

Rápidamente resultó que el protocolo en el bus de expansión (RS-485) es idéntico al de la conexión USB y una conexión Bluetooth. Entonces hubo otras tres fuentes:

- Christoph Nießen examina elBus de expansión
- Thomas Kaiser analiza otra cosa más precisamente el Comandos del protocolo
- Ad van der Weiden tiene uno convertidor desde el TX-C hasta el Robo-Interface-Extension construido

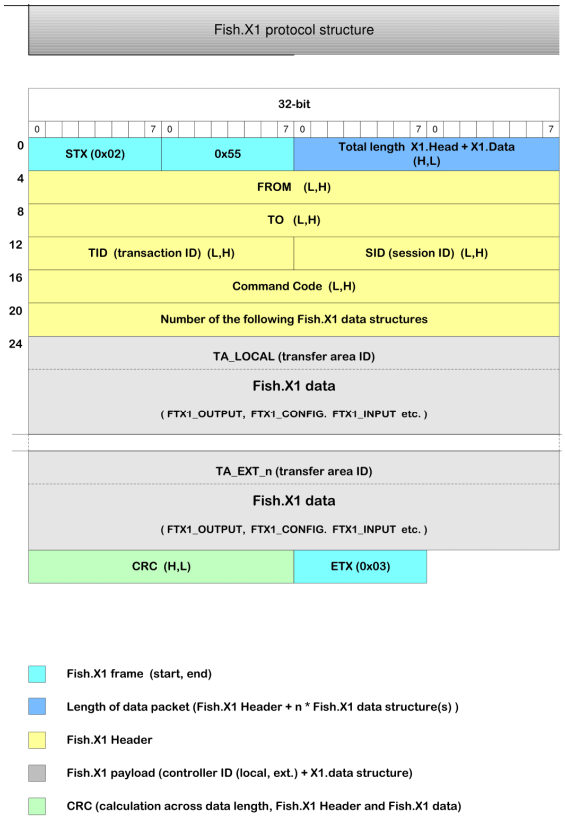

Fig. 1: Estructura del paquete (fuente: MSC, [1])

análisis

En el segundo paso, conecté el TX-C a una computadora, abrí ROBOPro y comencé la prueba de interfaz. Al mismo tiempo tengo de fondoWireshark la comunicación grabada. Como resultado, tuve una gran montaña de personajes HEX. Primero reduje todo a los datos del usuario y luego lo comparé con mis cuatro fuentes. MSC proporciona un gráfico para el protocolo X.1 en el que se describe muy bien la estructura del paquete (Fig. 1).

Offset	Length	Contents and meaning
0	1	STX (0x02) Marks the beginning of a Fish.X1 data packet with the following byte
1	1	0x55
2	2	Length (H,L) of a data packet Consists of the header length (20 bytes) and the payload length.
4	4	Bus address, 4 bytes (L,H), <FROM> (data packet transmitter) Possible values: BUS_ADR_WILDCARD=1, BUS_ADR_MASTER=2, BUS_ADR_SLAVE1=3 ... BUS_ADR_SLAVE8=10
8	4	Bus address, 4 bytes (L,H), <TO> (data packet receiver) Possible values, see above
12	2	Transaction **ID** (TID), 2 bytes (L,H) Each data packet is marked by the transmitter with a sequential number; the response to a request is returned by the message receiver with the same TID.
14	2	Session **ID** (SID), 2 bytes (L,H) The ROBO TX interface internally manages a session ID, which is reassigned every time there is a change in task from local to remote or remote to local. For example, the session ID is reassigned by the firmware if a timeout occurs during communication. The transmitter can then respond accordingly.
16	4	Fish.X1 command code (tele-code), 4 bytes (L,H) Defines a unique request or reply code. The command code thus describes the supplied data structure in the payload.
20	4	Number of subsequent Fish.X1 data structures, 4 bytes (L,H). (= n)
24	4	**Transfer area ID**, 4 bytes (L,H) The ID specifies for which ROBO TX Controller, and thus for which transfer area subsection, the subsequent data structure is addressed. The data structure specified via the command code is updated from the individual transfer area. The interfaces can be configured as master (always local) or as slaves (to a master). A corresponding ID is specified here via the transfer area ID: TA_LOCAL=0, TA_EXT_1=1, TA_EXT_2= 2, etc. The master (TA_LOCAL) manages the IO values of the individual slaves in its transfer area and can thereby assign the current IO data using the share memory ID.
28	m	Message payload, dependent upon the transmitted data structure. The payload, together with the share memory ID, can also have a length of 0, typically a request or a reply to a request. The structures defined in the header file ROBO_TX_FW.H are set as the payload. The command code identifies the supplied data structure.
24 + n*(4+m)	2	CRC, 2 bytes (H,L) The CRC is calculated from byte 2 (data length) to the end of the payload.
24 + 2 n*(4+m)	1	ETX (0x03) Marks the end of a Fish.X1 message

Fig.2: Explicaciones (fuente: MSC)

La estructura de los paquetes es relativamente simple:∂

- Cada paquete comienza con un preámbulo que consta de dos bytes (0x02 y 0x55)
- ∂ Cada paquete termina con 0x03
- ∂ Cada paquete consta de un preámbulo, longitud, encabezado, datos, una suma de comprobación y el final
- ∂ La longitud es la suma del número Bytes en el encabezado (20) y el número de bytes en los datos
- ∂ El encabezado contiene el remitente, el receptor, el ID de transacción, el ID de sesión, el código de comando y el número de bloques de datos transmitidos.
- ∂ Las direcciones del remitente y del destinatario se fijan cuando se transfiere a través de USB; Para paquetes desde la PC al TX-C, el remitente es 2 y el receptor es 1
- ∂ El ID de transacción aumenta con cada paquete enviado.
- ∂ El ID de sesión se puede establecer en 0 al principio hasta que el TX-C establezca un nuevo ID de sesión
- ∂ El número de datos transmitidos bloques indica cuántos datos TX-Cs se transmiten
- ∂ La suma de comprobación está de acuerdo con la especificación una suma de comprobación CRC que se calcula a partir de todos los bytes desde la longitud hasta el final del paquete de datos

El código de comando fue el mayor problema. Esto describe lo que realmente se transmite aquí y, por lo tanto, cómo se ve la estructura de los datos. Esto no se describe en las fuentes oficiales y, desafortunadamente, algunos de los códigos requeridos faltan en los análisis del bus de expansión. Al enviar, el código es un número (aquí: 1, 2, 5, 6, 7) y en la dirección opuesta es este número más 100.

A excepción de I^2C, los datos en sí consisten en un bloque por extensión controlada. Dentro de un bloque, el ID del área de transferencia viene primero y luego los datos del usuario. El ID del área de transferencia es 0 para el maestro y luego cuenta hasta 8 para la octava extensión. La estructura de los datos del usuario es diferente para cada código de comando, entraré en eso en un momento.

Finalmente, pude analizar los paquetes de datos más de cerca y descubrir cómo están estructurados los datos para algunos paquetes. Como señaló Thomas Kaiser, la suma de comprobación no es un CRC, sino un

Reste todos los bytes de la longitud hasta el final de los datos del número 0x00.

Lamentablemente, no pude identificar todo, por lo que pedí ayuda a fischertechnik. fischertechnik me proporcionó todos los datos necesarios. Un gran agradecimiento al Sr. Knecht, sin quien este desafío no podría haberse superado. Desafortunadamente, estos datos tampoco estaban actualizados al 100%, pero en algún momento pude resolver el rompecabezas. Ahora sabía cuándo debía enviarse qué paquete con qué código de comando y cómo debían estructurarse los datos en él.

Códigos de comando

Al analizar los datos intercambiados entre ROBOPro y TX-C, descubrí que se utilizan los códigos de comando 1, 2, 5, 6, 7, 101, 102, 105, 106 y 107. La siguiente descripción general describe el significado de estos códigos:

Eco (1)

∂ No contiene ningún dato

∂ Enviado cuando se abre la conexión

Respuesta de eco (101)

∂ Responder al eco∂ No

contiene ningún dato∂

Será ignorado

IO remoto (2)

∂ Envía datos de salida y datos a
Control de los contadores a los TX-Cs y se utiliza para consultar datos de entrada.

∂ Envía un bloque de datos al maestro
y cualquier extensión conectada

∂ El ID de reinicio del contador se utiliza para cada
Restablecer el contador a 0 aumentado en uno

∂ El ID del motor con el que se realiza la sincronización se escribe en el campo Motor Sync (0: sincronización desactivada)

∂ El ID de comando de motor debe aumentarse en uno cada vez que se cambia la distancia o el socio de sincronización de un motor; el motor vuelve a mover la distancia especificada

	4 Bytes			
0	Transfer Area ID			
4	Counter Reset ID 1		Counter Reset ID 2	
8	Counter Reset ID 3		Counter Reset ID 4	
12	Motor Sync 1	Motor Sync 2	Motor Sync 3	Motor Sync 4
16	Output Duty 1		Output Duty 2	
20	Output Duty 3		Output Duty 4	
24	Output Duty 5		Output Duty 6	
28	Output Duty 7		Output Duty 8	
32	Motor Distance 1		Motor Distance 2	
36	Motor Distance 3		Motor Distance 4	
40	Motor Command ID 1		Motor Command ID 2	
44	Motor Command ID 3		Motor Command ID 4	

Fig.3: Esquema de bytes Envío IO remoto

Respuesta de E / S remota (102)

∂ Responder a IO remoto

∂ Contiene los valores de entrada, información de los contadores e información sobre el control de motor ampliado

∂ Contiene un bloque de datos para el maestro y cada extensión conectada

∂ Los campos de valor de entrada contienen el
valor de entrada totalmente calculado. No es necesario realizar ninguna conversión para los diferentes modos.

∂ Los campos Estado del contador almacenan si la entrada del contador correspondiente está abierta (0x01) o cerrada (0x00)

∂ Counter Count es el valor del contador.

∂ Desde el ID de reinicio del contador del último
reinicio del contador completado, se puede ver si el último reinicio ya se ha realizado

∂ Desde el ID de comando del motor del último
Los comandos del motor completados muestran si el último comando ya se ha completado

	4 Bytes			
	Transfer Area ID			
0	Input Value 1		Input Value 2	
4	Input Value 3		Input Value 4	
8	Input Value 5		Input Value 6	
12	Input Value 7		Input Value 8	
16	Counter 1 State	Counter 2 State	Counter 3 State	Counter 4 State
20	Counter 1 Count		Counter 2 Count	
24	Counter 3 Count		Counter 4 Count	
28	Counter Reset ID 1		Counter Reset ID 2	
32	Counter Reset ID 3		Counter Reset ID 4	
36	Motor Command ID 1		Motor Command ID 2	
40	Motor Command ID 3		Motor Command ID 4	
44	Empty			

Fig.4: Respuesta de E / S remota del esquema de bytes

Escritura de configuración remota (5)∂

Envío de la configuración de entrada a los TX-C

∂ Envía un bloque de datos al maestro y a todas las extensiones conectadas

∂ Dependiendo del modo, se escribe un byte en los campos de Configuración de entrada. Si se va a medir un valor analógico, el valor debe multiplicarse por 128:

modo	**Digital**	**Término análogo**
tensión	0x00	0x80
Resistencia 5k	0x01	0x81
Resistencia 15k	0x02	0x82
Ultrasónico	0x03	0x83

	4 Bytes			
0	Transfer Area ID			
4	0x01	0x01	0x01	0x01
8	Input 1 Config	Input 2 Config	Input 3 Config	Input 4 Config
12	Input 5 Config	Input 6 Config	Input 7 Config	Input 8 Config
16	0x01	0x01	0x01	0x01
20	0x00	0x00	0x00	0x00
24	0x00	0x00	0x00	0x00
28	0x00	0x00	0x00	0x00
32	0x00	0x00	0x00	0x00

Fig.5: Esquema de bytes para escritura de configuración remota

Respuesta de escritura de Remote Config (105)∂

Responder a la escritura de Remote Config No

∂ contiene ningún dato

∂ Será ignorado

Información (6)

∂ Consulta de metadatos de maestros y extensiones.

∂ Envía un bloque de datos vacío al maestro y a todas las extensiones conectadas

Respuesta de información (106)

∂ Responder a la información

∂ Contiene un bloque de datos para el maestro y cada extensión conectada

∂ La dirección MAC de Bluetooth es los dos puntos ya existen en los datos

	4 Bytes		
0	Transfer Area ID		
4	Name des TX-Controller		
8			
12			
16			
20	Name des TX-Controller	Bluetooth MAC-Adresse	
24	Bluetooth MAC-Adresse		
28			
32			
36	Bluetooth MAC-Adresse	Empty	
40	Empty		
44			
48			
52			
56	Empty	Version 1	Version 2
60	Empty		
64			

Fig.6: Esquema de bytes de escritura de información

Estado (7)

∂ Consulta de extensiones conectadas

∂ Envía un bloque de datos vacío al maestro

Respuesta del estado (107)

∂ Responder al estado

∂ Contiene un bloque de datos del maestro.

∂ El estado de extensión de 8 bytes indica si la extensión correspondiente está conectada. Si el byte es 0, se separa; si es 1, esta extensión está conectada

	4 Bytes			
0	Transfer Area ID			
4	Extension 1 State	Extension 2 State	Extension 3 State	Extension 4 State
8	Extension 5 State	Extension 6 State	Extension 7 State	Extension 8 State
12	Empty			
16				
20				
24				

Fig.7: Esquema de bytes de respuesta de estado

software

La decisión de desarrollar el software en Python fue clara de antemano, ya que el firmware de la comunidad se basó en Python desde el principio y quería mantener la compatibilidad con él. Torsten Stuehn tiene la biblioteca para controlar las entradas y salidas del controlador TXT *ftrobopy* escrito. Para seguir siendo lo más compatible posible aquí también, había decidido diseñar las llamadas a funciones de la misma manera en el nivel de abstracción más alto. Residencia en *ftrobopy* Yo tengo el software *fttxpy* bautizado.

Niveles de abstracción

Para no perder la vista general, he dividido el software en cinco niveles de abstracción:

TXSerial: Aquí se abstraen la conexión en serie y el protocolo X.1. Una vez establecida la conexión, los paquetes se pueden enviar y recibir aquí. La información se almacena en un formato de datos uniforme, que contiene el remitente, el receptor, el código de comando y los bloques de datos para el maestro y las extensiones.

ftTX: Aquí se gestionan los datos de todas las entradas y salidas. Un proceso en segundo plano se encarga de mantener la conexión enviando todos los datos de salida al TX-C de la forma más permanente posible y leyendo todos los datos de entrada. Si se ha cambiado una configuración de entrada, se transfiere automáticamente al TX-C. Además, las extensiones conectadas se gestionan aquí para poder leer la versión de firmware y el nombre de la extensión, por ejemplo.

Para poder editar los datos almacenados más fácilmente, aquí se proporcionan funciones con las que, por ejemplo, se puede configurar la velocidad del motor o se puede configurar una entrada. Estas Las funciones se utilizan en los siguientes niveles y se hacen más fáciles de usar.

fttxpy: Este es el primer nivel con el que el usuario entrará en contacto; representa la combinación del maestro y las extensiones Para usar la biblioteca, el usuario llama a esta clase. El primer TX-C conectado se selecciona automáticamente. Después de llamar la clase, se emite automáticamente el nombre y la versión de firmware del maestro y todas las extensiones conectadas.

robotx: Esta clase representa un solo TX-C en una red. Las clases para la abstracción de varios dispositivos periféricos como motores, lámparas, botones y otros sensores están disponibles aquí. La función *robotx* se llama para direccionar un TX-C específico de la red. Sin argumentos obtienes un objeto del maestro, con un número como argumento obtienes la extensión correspondiente.

los *Sensores* y *Actuadores*: Las clases más importantes son responsables de los distintos sensores y actuadores. A continuación se ofrece una descripción detallada de estas clases.

programación

Para usar el software, necesita una computadora con Linux y Python 3 o un controlador TXT con el firmware de la comunidad. En el CFW es *fttxpy* ya preinstalado, tienes que estar en otros dispositivos *fttxpy* descargar y luego instalar como cualquier otro módulo de Python. Para hacer esto, ingrese:

```
instalación de python3 setup.py
```

Para los primeros intentos, es mejor abrir el intérprete interactivo de Python. Para importar la biblioteca y establecer la conexión, ingrese lo siguiente:

```
desde fttxpy importar fttxpy tx =
fttxpy ()
```

Si solo se conecta un TX-C sin extensiones, dice, por ejemplo:

```
Se encontraron 1 controlador (es) TX.
¡Seleccionando automáticamente el primero!

Conectado a ROBO TX-308 versión 1.30
```

Si hay extensiones conectadas, también hay:

```
Extensiones encontradas: Ext 1 (ROBO
TX-309, 1.30), Ext 2 (ROBO TX-309,
1,30)
```

Para acceder al maestro o una extensión, ingrese lo siguiente:

```
maestro = tx.robotx ()
ext1 = tx.robotx (1)
ext2 = tx.robotx (2)
. . .
```

Los objetos maestro, ext1, ... ahora ofrecemos las clases para controlar las entradas y salidas. Se pueden crear varios objetos para cada TX-C, pero no para las entradas y salidas. Si tienes un solo El objeto de salida ya no es necesario, debe incluirsedel (<objeto>) para ser eliminado. Entonces se puede crear un nuevo objeto.

motor

Un motor puede hacer mucho más de lo que mucha gente piensa. No solo se puede controlar en términos de dirección y velocidad, sino que también puede recorrer distancias definidas y sincronizarse con otro motor.

En primer lugar, nos gustaría variar un motor solo en términos de dirección y velocidad:

```
# Primero creamos un objeto motor

m1 = motor maestro (1)
# espere hasta que el usuario presione Enter
input ()
# Velocidad 128 derecha m1.setSpeed
(128)
aporte ()
# Velocidad 512 derecha m1.setSpeed
(512)
aporte ()
# Velocidad 128 izquierda
m1.setSpeed   (-128)
aporte ()
# Velocidad 512 izquierda
m1.setSpeed   (-512)
aporte ()
# Parada
m1.setSpeed   (0)
# O
m1.stop ()
```

Conducir una distancia determinada y luego detenerse automáticamente también es muy fácil:

```
# Establecer la distancia en 1000
m1.setDistance (1000)
# Velocidad en 300 m1.setSpeed
(300)
# Consulta si se ha alcanzado la posición

mientras no m1.finished ():
        pasaporte
# Establezca la velocidad y la distancia en 0

m1.stop ()
imprimir ("Listo")
```

La consulta finalizado () da Cierto o Falso regreso. Cuando se alcanza la posición, también lo es el valorCierto, de lo contrario Falso.

El comando parada() establece la velocidad y la distancia en 0. Si desea detener el motor antes de que llegue a su destino y luego dejarlo ir sin interrumpir esta medición de distancia, debe setSpeed (0) usar.

El arte de la robótica consiste en hacer que un robot en movimiento avance en línea recta. Esto también es con el setDistance-Función posible:

```
# Primero creamos dos objetos
motores
m1 = motor maestro (1)
m2 = motor maestro (2)
# Distancia para M1: 1000
m1.setDistance (1000)
# Distancia para M2: 1000
# Sincronizar M2 a M1
m2.setDistance (1000, m1)
# Velocidad de 512 m1.setSpeed
(512)
m2.setSpeed   (512)
# Esperando la posición alcanzada
mientras no m1.finished () y no
m2.finished ():
        pasaporte
imprimir ("Listo")
m1.stop ()
```

```
m2.stop ()
```

El comando parada() desactiva la sincronización. Si desea volver a sincronizar los motores, debe setDistance () Entregue nuevamente al socio de sincronización. Como en ROBOPro, un motor solo se puede sincronizar con otro motor en el mismo TX-C.

Salida

Una salida "O" es suficiente para controlar lámparas, electroválvulas o zumbadores, por ejemplo. Esto solo se puede controlar en intensidad, pero no en dirección:

```
# Crear objeto de salida o5 =
master.output (5)
# Intensidad 128
o5.setLevel (128)
aporte ()
# Intensidad 255
o5.setLevel (255)
aporte ()
# Apagar
o5.setLevel (0)
aporte ()
```

Botón

Probablemente el sensor más importante, pero también el más simple, es el botón. Así es como se lee en:

```
desde el tiempo de importación dormir
# Generar objeto de entrada i1 =
master.input (1)
mientras que es cierto:
        # Botón una vez por segundo
leer y sacar
        imprimir (i1.state ())
        dormir (1)
```

Resistencias

El sensor de brillo y el sensor de temperatura NTC son resistencias. Estos se pueden evaluar muy fácilmente. Además, la temperatura del sensor de temperatura NTC se puede convertir en grados Celsius:

```
# Generar objeto de entrada i2 =
master.resistor (2)
# Leer valor bruto
imprimir (i2.value ())
# Salida de la temperatura en grados
Celsius
print (str (i2.ntcTemperature ()), "grados
Celsius")
```

tensión

Si desea utilizar un sensor de color o comprobar el voltaje de la batería, proceda de la siguiente manera:

```
# Generar objeto de entrada i3 =
master.colorsensor (3)
# Tensión de salida en milivoltios print (str
(i3.voltage ()), "mV")
# Color de salida - ¡experimental! imprimir
(i3.color ())
```

La llamada color () es muy experimental. Acabo de sacar los valores *ftrobopy* copiado. No sé si los valores también te funcionarán. Tarea para los profesionales: escriba una función para dar salida al color en sus condiciones.

Sensor de seguimiento

En la RoboCup Junior, un robot de la disciplina Rescue Line tiene que seguir una línea, entre otras cosas. fischertechnik ofrece el sensor de seguimiento para este propósito. Esto suministra 0 V o 9 V en la salida. La evaluación es muy sencilla:

```
# Generar objeto de entrada i4 =
master.trailfollower (4)
# Imprimir estado actual de salida
(i4.state ())
```

Para evaluar la pista, por supuesto, debe crear dos objetos de entrada para las dos entradas y evaluarlos con un poco de lógica.

Sensor de distancia

Si desea medir una distancia con el sensor de distancia ultrasónico de fischertechnik, debe proceder de la siguiente manera:

```
# Generar objeto de entrada i5 =
master.ultrasonic (5)
# Impresión de distancia de
corriente de salida (i5.distance ())
```

Palabra final

No sé con qué propósito se desarrolló originalmente el protocolo X.1, pero no es adecuado para la transferencia de datos más rápida posible entre una PC y un TX con hasta nueve extensiones. Se desperdició mucho espacio en muchos lugares, comenzando con la dirección del remitente y el destinatario, que está completamente sobredimensionada con cuatro bytes (cuatro bits habrían sido suficientes), hasta el código de comando con cuatro bytes (hay algunos códigos más de los que se muestran aquí, pero un byte hubiera sido suficiente) y finalmente el número de estructuras de datos con cuatro bytes (un maestro y ocho extensiones, es decir, un máximo de nueve estructuras de datos; cinco bits hubieran sido suficientes para esto) se puede ver cuán ineficiente es el solo es el encabezado del paquete, que se transmite con cada paquete al que se llega. Muchos bytes no utilizados podrían haberse eliminado de muchos lugares de los paquetes. Los 10 ms prometidos por fischertechnik

La cuadrícula ya no se puede mantener con solo cinco extensiones. Aquí, se requieren 14 ms por ciclo de bus [2]; El protocolo requiere 22,4 ms para ocho extensiones.

Durante el análisis, noté en algún momento que puede elegir libremente el ID de transacción. Aquí también es posible "saltar" libremente durante la conexión, por lo que estos dos bytes podrían haberse guardado.

Independientemente de si activa el sensor de distancia como digital o analógico, siempre entrega valores, y si desconecta el sensor durante la conexión, se conserva el valor anterior.

fuentes

[1] fischertechnik GmbH: *Recopilación de los archivos más importantes.*.

[2] Ad van der Weiden: *Traza del analizador lógico de la comunicación RS485 entre un maestro y 5 esclavos*.