izmdb-1.0.0

Una biblioteca para usar el protocolo MDB (Multi-Drop Bus) con dispositivos USB.

Introducción

Esta biblioteca de *C* implementa una API sencilla basada en *libusb* para encontrar, establecer comunicacion, enviar y recibir paquetes de información con dispositivos que utilizan el protocolo MDB.

El desarrollador solo requiere incluir el encabezado `izmdb.h` para tener acceso a las funciones necesarias.

También está incluida una implementación parcial de la API USBXpress para acceso de bajo nivel a la comunicación con el dispositivo y un paquete de *Python* que expone directamente la funcionalidad de la biblioteca de *C* y un juego de funciones de alto nivel para el envío y recepción de datos via MDB.

Guía rápida

El proceso para enviar comandos y recibir la respuesta del dispositivo es simple. A continuación describimos los pasos generales para hacerlo desde *C* y *Python*.

💡 El paquete fuente izmdb incluye programas de demostración del uso de la API tanto en en *C* como en *Python*:

- `izmdb-demo.c`
- `lowlevel-izmdb-demo.c`
- `python/izmdb-demo.py`
- `python/lowlevel-izmdb-demo.py`

En lenguaje C

1. Incluir el encabezado `izmdb.h`
2. `izmdb_open()`: Inicializar la comunicacion con el dispositivo. Solo es necesario hacerlo una vez al inicio del programa.
3. `izmdb_send()`: Preparar el comando y argumentos como un arreglo de

Hortensias No 22 Fraccionamiento Jacarandas, Zapopan, Jalisco. CP. 45160 Tel. / Fax 01 (33) 3833-0373
www.asic-linux.com.mx **info@asic-linux.com.mx**

bytes y enviarlo al dispositivo.
4. `izmdb_receive()`: Leer la respuesta del dispositivo.
5. `izmdb_close()`: Finalizar la comunicacion con el dispositivo antes de terminar el programa.

En Python

1. Importar el módulo `izmdb`
2. `izmdb.open()`: Inicializar la comunicacion con el dispositivo. Solo es necesario hacerlo una vez al inicio del programa.
3. `izmdb.send()`: Preparar el comando y argumentos como un arreglo de bytes y enviarlo al dispositivo.
4. `izmdb.receive()`: Leer la respuesta del dispositivo.
5. `izmdb.close()`: Finalizar la comunicacion con el dispositivo antes de terminar el programa.

Instalación

Para la biblioteca de *C*:

```
make
make install # Como root
```

Para el módulo de *Python*:

```
make python-install
```

U opcionalmente instalar ambos en un solo comando:

```
make install-all
```

Compilar los programas de demostración:

```
make demos
```


Hortensias No 22 Fraccionamiento Jacarandas, Zapopan, Jalisco. CP. 45160 Tel. / Fax 01 (33) 3833-0373
www.asic-linux.com.mx **info@asic-linux.com.mx**

API

C

Estructuras de datos y tipos

izmdb_packet

Representa a un paquete de información para la comunicación MDB con el dispositivo.

```
struct izmdb_packet {
  unsigned char *stream;  # Datos contenidos en el paquete
  int  size;              # Cantidad de bytes a los que apunta
packet.stream
};
```

izmdb_handle

Estructura de datos opaca utilizada para referirse a un dispositivo.

Estados de retorno

IZMDB_ERR	**-1**
IZMDB_OK	**1**
IZMDB_NOMEM	**-2**
IZMDB_INVALID_PACKET	**-3**
IZMDB_DEV_NOT_FOUND	**-4**
IZMDB_READ_ERROR	**-5**
IZMDB_INVALID_PARAM	**-6**
IZMDB_INVALID_PACKET_FROM_DEVICE	**-7**
IZMDB_CANNOT_OPEN_DEV	**-8**
IZMDB_WRITE_ERROR	**-9**

Hortensias No 22 Fraccionamiento Jacarandas, Zapopan, Jalisco. CP. 45160 Tel. / Fax 01 (33) 3833-0373
www.asic-linux.com.mx **info@asic-linux.com.mx**

Funciones

int izmdb_open (int vendor_id, int product_id, izmdb_handle **handle);

Busca el primer dispositivo USB con identificacion VENDOR_ID:PRODUCT_ID y si lo encuentra, devuelve IZMDB_OK y HANDLE apunta a un handle para el dispositivo, utilizable por izmdb_send_packet() e izmdb_receive_packet().

int izmdb_close (izmdb_handle *handle);

Cierra la comunicación con el dispositivo USB referido por HANDLE.

int izmdb_send (izmdb_handle *handle, unsigned char *data, int datasize);

Envía al dispositivo manejado por HANDLE la cantidad de bytes DATASIZE apuntados por DATA.

int izmdb_receive (izmdb_handle *handle, unsigned char **databuf, int *datasize);

Recibe en el buffer DATABUF de tamaño apuntado por DATASIZE los datos contenidos en el siguiente paquete de comunicación disponible en el dispositivo manejado por HANDLE.

Regresa en el entero apuntado DATASIZE la cantidad de datos recibidos.

int izmdb_pack (unsigned char *pkdata, int datasize, struct izmdb_packet *packet);

Empaquetar con Simple Coin Protocol la secuencia de comando MDB (Comando + argumentos) de longitud DATASIZE a la que apunta PKDATA.

La secuencia de comando va empaquetada asi:

```
| encabezado | comando + datos | terminador |
    0xC1          carga util          0xC2
```

La carga útil se codifica de manera que si esta incluye los bytes `0xC0`, `0xC1` y `0xC2` estos son escapados con un byte `0xC0` y codificados en la siguiente forma:

```
0xC0 --> 0xC0 0x00
```


Hortensias No 22 Fraccionamiento Jacarandas, Zapopan, Jalisco. CP. 45160 Tel. / Fax 01 (33) 3833-0373
www.asic-linux.com.mx **info@asic-linux.com.mx**

```
0xC1 --> 0xC0 0x01
0xC2 --> 0xC0 0x02
```

Todos los demás bytes son copiados tal cual al paquete.

Si el empaquetamiento es exitoso devuelve IZMDB_OK y PACKET apunta al paquete codificado.

int izmdb_unpack (struct izmdb_packet *packet, unsigned char **pkdata, int *datasize);

Desempaquetar los datos contenidos en el paquete al que apunta PACKET.

Esta es la operación inversa de izmdb_pack().

Si el desempacamiento es exitoso devuelve IZMDB_OK, PKDATA apunta a los datos descodificados y DATASIZE apunta a un entero que representa la cantidad de bytes descodificados.

int izmdb_send_packet (izmdb_handle *handle, struct izmdb_packet *packet);

Envía el paquete MDB apuntado por PACKET al dispositivo USB manejado por HANDLE.

int izmdb_receive_packet (izmdb_handle *handle, struct izmdb_packet *packet);

Recibe un paquete MDB y lo almacena en el paquete apuntado por PACKET desde el dispositivo USB manejado por HANDLE.

Se espera que el campo 'stream' de PACKET apunta a un area reservada de memoria de tamaño suficiente y el campo 'size' indica el tamaño de la misma.

Python

Para usar el paquete **izmdb** solo es necesario importarlo en tu código:

```
import izmdb
```


Hortensias No 22 Fraccionamiento Jacarandas, Zapopan, Jalisco. CP. 45160 Tel. / Fax 01 (33) 3833-0373
www.asic-linux.com.mx **info@asic-linux.com.mx**

Funciones

izmdb_handle izmdb.open(int vid, int pid)

Buscar el dispositivo con USB VendorID VID y USB ProductID PID y establecer comunicacion con el.

Regresa un handle enlazado con ese dispositivo.

string error_decode (int errnum)

Regresa una cadena que representa el número de error ERRNUM.

int izmdb.close(izmdb_handle handle)

Cerrar la comunicación con el dispositivo manejado por HANDLE.

En caso de éxito devuelve IZMDB_OK.

int izmdb.send(izmdb handle, bytes bytes)

Crea un paquete MDB codificando el contenido del arreglo BYTES y lo envía al dispositivo manejado por HANDLE.

BYTES debe ser un objeto bytes, generado por ejemplo con una llamada a bytes():

```
b = bytes([0x01, 0x12, 0x0F, 0x00])
```

En caso de éxito devuelve IZMDB_OK.

bytes izmdb.receive(handle, datasize)

Recibe el siguiente paquete MDB disponible en el dispositivo manejado por HANDLE y devuelve su contenido descodificado como un arreglo de bytes dentro de un objeto iterable de clase *bytes*.

Hortensias No 22 Fraccionamiento Jacarandas, Zapopan, Jalisco. CP. 45160 Tel. / Fax 01 (33) 3833-0373
www.asic-linux.com.mx **info@asic-linux.com.mx**