Xbox 360 controller demo

Generated by Doxygen 1.8.6

Fri May 29 2015 10:52:10

Contents

1 Course: DES (EL32), Assignment 10

Author

A.W. Janisse

Assignment description

The main goal of this application is to explore how specific USB hardware can be controlled. For this goal a Datel Xbox 360 controller will be used. Although it is not an original Microsoft one it should be 100% compatible. Below an image of this controller is given.

The software developed for this assignment is using the libusb library. This library provides an abstraction for communication with USB devices.

Please refer to the **Building the source files** (p. 2) page for more information on this topic.

2 Building the source files

Commandline parameters

The can be started with several commandline parameters. With these parameters several build options can be selectected. The table below gives an overview of the parameters and describes what will be build.

Running Make is done through the commandline and can be executed like this: **make** <**paramater**>. Make can also be run without any paramets which will be the same as make all.

Parameter	Builds
all	Build target the executable.
debug	Build with debugger information.
clean	Clean up. Removes the target executable and all object files (.o)
info	Print information regarding the files, used compiler and compiler flags.
pi	Builds the executable with the arm-linux-gcc toolchain. The executable can run on the Raspberry Pi
install	Copy the target executable to the Raspberry Pi. Please note that the ip-address for the Pi must be 10.0.0.42 and the username must be 'root'. Also note the executable will be copied on the target /bin directory.
html	Produces (this) html documentation.
pdf	Produces PDF documentation.

Used built-in functions

Advantage is taken from several built-in functions. These functions are listed and described below.

- **$(wildcard pattern...)**

  Find file names matching a shell file name pattern (not a '' pattern). In the Makefile this command is used like:

```
SOURCES = $(wildcard *.c)
```

- **$(patsubst pattern,replacement,text)**

  Replace words matching pattern with replacement in text. In the Makefile this commands is used like:

```
OBJECTS = $(patsubst %.c, %.o, $(SOURCES))
```

3 Bug List

File xbcontroller.c (p. **5**)

No known bugs.

File xbcontroller.h (p. **8**)

No known bugs.

4 Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

5 File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

6 Data Structure Documentation

6.1 Buttons Struct Reference

Structure representing the buttons and joysticks.

```
#include <xbcontroller.h>
```

Data Fields

- bool **D_UP**
  D-Pad up.
- bool **D_DN**
  D-Pad down.
- bool **D_LEFT**
  D-Pad left.
- bool **D_RIGHT**
  D-pad right.
- bool **START**
  Start button.
- bool **BACK**
  Back button.
- bool **LS_PRESS**
  Left stick press.
- bool **RS_PRESS**
  Right stick press.
- bool **LB**
  Button LB.
- bool **RB**
  Button RB.
- bool **LOGO**
  Xbox logo button.
- bool **SPARE**

Unused.

- bool **A**

  Button A.
- bool **B**

  Button B.
- bool **X**

  Button X.
- bool **Y**

  Button Y.
- uint8_t **Left_trigger**

  Left trigger. Produces a value from 0 to 255.
- uint8_t **Right_trigger**

  Right trigger. Produces a value from 0 to 255.
- int16_t **Left_stick_X**

  Left joystick x-value. Produces a value from -32768 to 32767.
- int16_t **Left_stick_Y**

  Left joystick y-value. Produces a value from -32768 to 32767.
- int16_t **Right_stick_X**

  Right joystick x-value. Produces a value from -32768 to 32767.
- int16_t **Right_stick_Y**

  Right joystick y-value. Produces a value from -32768 to 32767.

6.1.1 Detailed Description

Structure representing the buttons and joysticks.

The documentation for this struct was generated from the following file:

- **xbcontroller.h**

7 File Documentation

7.1 xbc.c File Reference

Function prototypes for the console driver.

```
#include <libusb-1.0/libusb.h>
#include <stdio.h>
#include <unistd.h>
#include "xbcontroller.h"
```

Macros

- #define **THRESHOLD** 150

  Treshold level for displaying analog values.

Functions

- int **main** (int argc, char ∗argv[])

  this main func

7.1.1 Detailed Description

Function prototypes for the console driver. This contains the prototypes for xxxxxxxxxxxxxxxxxxxxxx

Author

A.W Janisse

Version

1.0

Date

28-05-2015

7.1.2 Function Documentation

7.1.2.1 int main (int *argc,* char * *argv[]*)

this main func

@ return ??????

7.2 xbcontroller.c File Reference

Function prototypes for the console driver.

```
#include <stdio.h>
#include "xbcontroller.h"
```

Functions

- void **getDevices** (libusb_device **devices)
  Retreives a fresh list of connected USB devives.
- int **printAllDevices** (libusb_device **devices)
  Prints a list of connected USB devices.
- int **rumble** (libusb_device_handle *handle, uint16_t speed)
  Controll the rumble actuator from the Xbox 360 controller.
- int **setLeds** (libusb_device_handle *handle, enum **leds** led)
  Controll the LED's from the Xbox 360 controller.
- int **getInput** (libusb_device_handle *handle, struct **Buttons** *buttons)
  Retreives the state of the buttons and joysticks.

7.2.1 Detailed Description

Function prototypes for the console driver. This contains the prototypes for xxxxxxxxxxxxxxxxxxxxxx

Author

A.W Janisse

Bug No known bugs.

7.2.2 Function Documentation

7.2.2.1 void getDevices (libusb_device ** *devices*)

Retreives a fresh list of connected USB devives.

Parameters

devices	is a pointer to the list

7.2.2.2 int getInput (libusb_device_handle * *handle*, struct Buttons * *buttons*)

Retreives the state of the buttons and joysticks.

Parameters

handle	is a pointer to the USB device handle.
buttons	is a pointer to a **Buttons** (p. 3) struct which will be filled with the button states and joystick values.

Returns

0 if succefull or error code if fails.

7.2.2.3 int printAllDevices (libusb_device ** *devices*)

Prints a list of connected USB devices.

This function prints information about all the current connected USB devices to the standard console.

Parameters

devices	is a pointer to the list with devices.

Returns

0 if succefull or error code if fails.

7.2.2.4 int rumble (libusb_device_handle * *handle*, uint16_t *speed*)

Controll the rumble actuator from the Xbox 360 controller.

Controll the rumble actuator with the desired speed. The speed can varry from 0 (no rumble) to 0xFFFF (max rumble).

Parameters

handle	is a pointer to the USB device handle.
speed	is the value for the rumble actuator.

Returns

0 if succefull or error code if fails.

7.2.2.5 int setLeds (libusb_device_handle * *handle*, enum leds *led*)

Controll the LED's from the Xbox 360 controller.

Parameters

handle	is a pointer to the USB device handle.
led	holds the desired pattern for the LED's.

Returns

0 if succefull or error code if fails.

7.3 xbcontroller.h File Reference

Function prototypes for the console driver.

```
#include <libusb-1.0/libusb.h>
#include <stdbool.h>
```

Data Structures

- struct **Buttons**

  Structure representing the buttons and joysticks.

Macros

- #define **VENDOR_ID** 0x045e
- #define **VENDOR_PROD** 0x028e
- #define **EP_IN** 0x81
- #define **EP_OUT** 0x02

Enumerations

- enum **leds** {
  all_off = 0x00, **blink_all** = 0x01, **flash_1_on** = 0x02, **flash_2_on** = 0x03,
  flash_3_on = 0x04, **flash_4_on** = 0x05, **led_1_on** = 0x06, **led_2_on** = 0x07,
  led_3_on = 0x08, **led_4_on** = 0x09, **rotate** = 0x0A, **blink_select** = 0x0B,
  blink_slow = 0x0C, **alt** = 0x0D }

  Availlable LED patterns.

Functions

- void **getDevices** (libusb_device **devices)

  Retreives a fresh list of connected USB devives.
- int **printAllDevices** (libusb_device **devices)

  Prints a list of connected USB devices.
- int **rumble** (libusb_device_handle *handle, uint16_t speed)

  Controll the rumble actuator from the Xbox 360 controller.
- int **setLeds** (libusb_device_handle *handle, enum **leds** led)

  Controll the LED's from the Xbox 360 controller.
- int **getInput** (libusb_device_handle *handle, struct **Buttons** *buttons)

  Retreives the state of the buttons and joysticks.

7.3.1 Detailed Description

Function prototypes for the console driver. This contains the prototypes for xxxxxxxxxxxxxxxxxxxxxxx

Author

A.W Janisse

Bug No known bugs.

7.3.2 Enumeration Type Documentation

7.3.2.1 enum leds

Availlable LED patterns.

Enumerator

all_off All led's off.

blink_all All blinking.

flash_1_on 1 flashes, then on

flash_2_on 2 flashes, then on

flash_3_on 3 flashes, then on

flash_4_on 4 flashes, then on

led_1_on 1 on

led_2_on 2 on

led_3_on 3 on

led_4_on 4 on

rotate Rotating (e.g. 1-2-4-3)

blink_select previous setting will be used (all blinking, or 1, 2, 3 or 4 on).

blink_slow Slow blinking∗.

alt Alternating (e.g. 1+4-2+3), then back to previous∗.

7.3.3 Function Documentation

7.3.3.1 void getDevices (libusb_device ∗∗ *devices*)

Retreives a fresh list of connected USB devives.

Parameters

devices	is a pointer to the list

7.3.3.2 int getInput (libusb_device_handle ∗ *handle,* struct Buttons ∗ *buttons*)

Retreives the state of the buttons and joysticks.

Parameters

handle	is a pointer to the USB device handle.
buttons	is a pointer to a **Buttons** (p. 3) struct which will be filled with the button states and joystick values.

Returns

0 if succefull or error code if fails.

7.3.3.3 int printAllDevices (libusb_device ∗∗ *devices*)

Prints a list of connected USB devices.

This function prints information about all the current connected USB devices to the standard console.

Parameters

devices	is a pointer to the list with devices.

Returns

0 if succefull or error code if fails.

7.3.3.4 int rumble (libusb_device_handle * *handle*, uint16_t *speed*)

Controll the rumble actuator from the Xbox 360 controller.

Controll the rumble actuator with the desired speed. The speed can varry from 0 (no rumble) to 0xFFFF (max rumble).

Parameters

handle	is a pointer to the USB device handle.
speed	is the value for the rumble actuator.

Returns

0 if succefull or error code if fails.

7.3.3.5 int setLeds (libusb_device_handle * *handle*, enum leds *led*)

Controll the LED's from the Xbox 360 controller.

Parameters

handle	is a pointer to the USB device handle.
led	holds the desired pattern for the LED's.

Returns

0 if succefull or error code if fails.

Index