Xbox 360 controller demo

Generated by Doxygen 1.8.6

Sat Jun 6 2015 14:54:19

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. The assignment has the following requirements:

- The program can show the states of all the buttons;
- The program can controll the LED's;
- The program can controll the rumble actuator;
- The program can be build with a Makefile;
- The program can run on the Raspberry Pi;
- The software is documentated.

Building the source

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

How it all works

On the **How it all works** (p. **??**) page a detailed description is given on this topic.

Extra's

During this assignment I got excited about **`Doxygen`**. Dogygen is a tool which can be used to produce documentation in broad sense. One of the possibilities is to generate documentation from comments in source file. The webpage you are reading right now is created with Doxygen.

2 Building the source files

Commandline parameters

The software can be build with make which can be started with several commandline parameters. With these parameters several build options will 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. The complete Makefile can be found **here** (p. **??**)

*2|>p215.02206pt|

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 produced 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.

backup Calls clean and produces a tar archive file.

html Produces (this) html documentation.

pdfProduces 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)
  ```

  This creates a list with all the .c into the variable SOURCES.

- **$(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))
  ```

  The used pattern is '%.c' and the replacement is '%.o'. Since the wildcard '%' is used effectively all items in the SOURCES variable will be renamed and stored in the OBJECTS variable.

3 Makefile

```
#
#   Generic Makefile for simple projects.
#
#   (C) 2015, A.W. Janisse
#
#   Macro's:
#       OUTPUT  : Name of the executable
#       PIDIR   : Place for installation on the Raspberry PI
#       CC      : Default compiler. (Note make pi will build for the Raspberry Pi platform)
#       LIBS    : Libraries to use when building
#       CFLAGS  : Compiler flags
#       ZIPDIR  : Directory to put the backup archive files
#

OUTPUT  = xbc
PIDIR   = root@10.0.0.42:/bin
CC      = gcc
```

```
LIBS    = -lusb-1.0
CFLAGS  = -O2 -Wall -Werror
ZIPDIR  = ../backup

### -----[ Do not change anything below this line ]----- ###

# Remove any unwanted leading and trailing spaces
TARGET = $(strip $(OUTPUT))
# Retreive a list of source files (ending with .c)
SOURCES = $(wildcard *.c)
# Replace all .c in the sources list to .o
OBJECTS = $(patsubst %.c, %.o, $(SOURCES))
# Retreive a list of header files (ending with .h)
HEADERS = $(wildcard *.h)
# Build the archive name and set extension
TARNAME = $(ZIPDIR)/$(TARGET)_$(shell date +'%Y%m%d_%H%M')$(TAREXT)
TAREXT  = .tar

.PHONY: all debug clean install info html pdf backup pi

# Rule to perform when just make is executed.
all: $(TARGET)

# implicit rule for building the object files.
%.o: %.c $(HEADERS)
    $(CC) $(CFLAGS) -c $< -o $@

# Don't delete intermediate files when this make is aborted for some reason.
.PRECIOUS: $(TARGET) $(OBJECTS)

# Here the compiling hapens
$(TARGET): $(OBJECTS)
    $(CC) $(OBJECTS) -Wall $(LIBS) -o $@

# Build all with debug information. The X86 toolchain is used.
debug: CFLAGS += -g
debug: all

# Build all with the arm-linux-gcc toolchain.
pi: CC=arm-linux-gcc
pi: all

# Just cleanup by removing the exectable en .obj files.
clean:
    @rm -rf $(TARGET) $(OBJECTS)

# Give information about this Makefile.
info:
    @echo ==================================================================
    @echo Output"   ":   $(TARGET)
    @echo Sources"  ": $(SOURCES)
    @echo Headers"  ": $(HEADERS)
    @echo Objects"  ": $(OBJECTS)
    @echo Libraries: $(LIBS)
    @echo Compiler : $(CC)
    @echo CFlags"   ": $(CFLAGS)
    @echo Zip dir"  ": $(ZIPDIR)
    @echo ==================================================================

# Copy the executable over to the Raspberry PI.
install:
    @echo Connecting...
    @scp $(TARGET) $(PIDIR)
    @echo installation done!

# Create an archive file containing all the essential files for reproduction.
backup: clean
backup:
    @mkdir -p $(ZIPDIR)
    @tar -cf $(TARNAME) .
    @echo Created archive: $(TARNAME)

# Produce the HTML documentation based on the settings in Doxyfile.
html:
    @doxygen
    @echo HTML documentation is generated in doc/html

# Produce the PDF documentation based on the settings in Doxyfile.
pdf:
    $(MAKE) -C doc/latex
    @echo PDF documentation is generated in doc/latex
```

4 How it all works

5 Bug List

File Controller.c (p. **??**)
No known bugs.

File Controller.h (p. **??**)
No known bugs.

File DemoController.c (p. **??**)
No known bugs.

File DemoController.h (p. **??**)
No known bugs.

File Devices.c (p. **??**)
No known bugs.

File Devices.h (p. **??**)
No known bugs.

6 Data Structure Index

6.1 Data Structures

Here are the data structures with brief descriptions:

7 File Index

7.1 File List

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

8 Data Structure Documentation

8.1 Buttons Struct Reference

struct **Buttons** (p. **??**)

```
#include <Controller.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.

8.1.1 Detailed Description

struct **Buttons** (p. **??**)

This structure contains the states of all the buttons and analog joysticks of the Xbox controller. This structure is filled with the function **getControllerInput** (p. **??**).

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

- **Controller.h**

9 File Documentation

9.1 Controller.c File Reference

Implementation for the controller.

```
#include <libusb-1.0/libusb.h>
#include <stdio.h>
#include <assert.h>
#include "Controller.h"
```

Macros

- #define **TIMEOUT** 1000

Functions

- int **setControllerRumble** (libusb_device_handle ∗handle, uint16_t speed)

  Controll the Xbox 360 controller rumble actuator with the desired speed.
- int **setControllerLeds** (libusb_device_handle ∗handle, **Leds** led)

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

  Retreives the state of the buttons and joysticks.

9.1.1 Detailed Description

Implementation for the controller. This is the implementation of the ccontroller.

Author

A.W Janisse

Bug No known bugs.

Version

1.0 First release.

9.1.2 Function Documentation

9.1.2.1 int getControllerInput (libusb_device_handle ∗ *handle,* Buttons ∗ *buttons*)

Retreives the state of the buttons and joysticks.

Parameters

|>p0.15|p0.805|

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

Returns

0 if succefull.

Precondition

A valid pointer (not NULL) to handle and button.

Postcondition

The structure buttons is filled with the current state.

9.1.2.2 int setControllerLeds (libusb_device_handle ∗ *handle,* Leds *led*)

Controll the LED's from the Xbox 360 controller.

Parameters

|>p0.15|p0.805|

handle	is a pointer to the USB device handle.
led	holds the desired pattern for the LED's. See **Leds** (p. **??**) for details

Returns

0 if succefull.

Precondition

A valid pointer (not NULL) to handle.

9.1.2.3 int setControllerRumble (libusb_device_handle ∗ *handle,* uint16_t *speed*)

Controll the Xbox 360 controller rumble actuator with the desired speed.

Parameters

|>p0.15|p0.805|

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

Returns

0 if succefull.

Precondition

A valid pointer (not NULL) to handle.

9.2 Controller.h File Reference

Function prototypes for the controller.

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

Data Structures

- struct **Buttons**

  *struct **Buttons** (p. **??**)*

Macros

- #define **VENDOR_ID** 0x045e

  VENDOR_ID.
- #define **VENDOR_PROD** 0x028e

  VENDOR_PROD.
- #define **EP_IN** 0x81

  EP_IN.
- #define **EP_OUT** 0x02

  EP_OUT.

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 }

  Enum Leds.

Functions

- int **setControllerRumble** (libusb_device_handle ∗handle, uint16_t speed)

  Controll the Xbox 360 controller rumble actuator with the desired speed.
- int **setControllerLeds** (libusb_device_handle ∗handle, **Leds** led)

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

  Retreives the state of the buttons and joysticks.

9.2.1 Detailed Description

Function prototypes for the controller. This contains the prototypes for the controller and eventually any macros, constants or global variables you will need.

Author

A.W Janisse

Bug No known bugs.

Version

1.0 First release.

9.2.2 Macro Definition Documentation

9.2.2.1 #define EP_IN 0x81

EP_IN.

Constant for the endpoint in.

9.2.2.2 #define EP_OUT 0x02

EP_OUT.

Constant for the endpoint out.

9.2.2.3 #define VENDOR_ID 0x045e

VENDOR_ID.

Constant for vendor id. This constant can be used to derive the handle.

9.2.2.4 #define VENDOR_PROD 0x028e

VENDOR_PROD.

Constant for vendor product id. This constant can be used to derive the handle.

9.2.3 Enumeration Type Documentation

9.2.3.1 enum Leds

Enum Leds.

This enumeration contains all the command availlable to controll the LED's. The function **setControllerLeds** (p. **??**) can be used to controll the states of the indvidual LED's.

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.

9.2.4 Function Documentation

9.2.4.1 int getControllerInput (libusb_device_handle $*$ *handle,* Buttons $*$ *buttons*)

Retreives the state of the buttons and joysticks.

Parameters

|>p0.15|p0.805|

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

Returns

0 if succefull.

Precondition

A valid pointer (not NULL) to handle and button.

Postcondition

The structure buttons is filled with the current state.

9.2.4.2 int setControllerLeds (libusb_device_handle $*$ *handle,* Leds *led*)

Controll the LED's from the Xbox 360 controller.

Parameters

|>p0.15|p0.805|

handle	is a pointer to the USB device handle.
led	holds the desired pattern for the LED's. See **Leds** (p. **??**) for details

Returns

0 if succefull.

Precondition

A valid pointer (not NULL) to handle.

9.2.4.3 int setControllerRumble (libusb_device_handle $*$ *handle,* uint16_t *speed*)

Controll the Xbox 360 controller rumble actuator with the desired speed.

Parameters

|>p0.15|p0.805|

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

Returns

0 if succefull.

Precondition

A valid pointer (not NULL) to handle.

9.3 DemoController.c File Reference

Function prototypes for the console driver.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include "Controller.h"
```

Functions

- void **waitForRelease** ()
- void **demoLeds** ()
- void **demoRumble** ()
- void **demoButtons** ()
- void **demoAnalog** ()
- void **demoController** (libusb_device_handle ∗handle)

    vertel over de functie democontroller
- void **demoButtons** (libusb_device_handle ∗h)

Variables

- libusb_device_handle ∗ **h**
- **Buttons buttons**
- bool **to_quit** = false

9.3.1 Detailed Description

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

Author

A.W Janisse

Bug No known bugs.

9.3.2 Function Documentation

9.3.2.1 void demoController (libusb_device_handle ∗ *handle*)

vertel over de functie democontroller

Parameters

|>p0.15|p0.805|

handle is de parameter

9.4 DemoController.h File Reference

Function prototypes for the console driver.

```
#include <libusb-1.0/libusb.h>
#include "Controller.h"
```

Functions

- void **demoController** (libusb_device_handle ∗handle)

  vertel over de functie democontroller

9.4.1 Detailed Description

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

Author

A.W Janisse

Bug No known bugs.

9.4.2 Function Documentation

9.4.2.1 void demoController (libusb_device_handle ∗ *handle*)

vertel over de functie democontroller

Parameters

|>p0.15|p0.805|

handle is de parameter

9.5 Devices.c File Reference

Function prototypes for the console driver.

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

Functions

- int **getDevices** (libusb_device ∗∗devices)

  Retreives a fresh list of connected USB devives.

- int **printAllDevices** (libusb_device ∗∗devices)

  Prints a list of connected USB devices.

Variables

- libusb_device ∗ **xbdevices** [4] = {NULL, NULL, NULL, NULL}

9.5.1 Detailed Description

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

Author

A.W Janisse

Bug No known bugs.

9.5.2 Function Documentation

9.5.2.1 int getDevices (libusb_device ∗∗ *devices*)

Retreives a fresh list of connected USB devives.

Parameters

|>p0.15|p0.805|

devices is a pointer to the list

9.5.2.2 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

|>p0.15|p0.805|

devices is a pointer to the list with devices.

Returns

0 if succefull or error code if fails.

9.6 Devices.h File Reference

Function prototypes for the console driver.

```
#include <libusb-1.0/libusb.h>
#include "Controller.h"
```

Functions

- int **getDevices** (libusb_device ∗∗devices)

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

  Prints a list of connected USB devices.

9.6.1 Detailed Description

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

Author

A.W Janisse

Bug No known bugs.

9.6.2 Function Documentation

9.6.2.1 int getDevices (libusb_device ∗∗ *devices*)

Retreives a fresh list of connected USB devives.

Parameters

|>p0.15|p0.805|

devices is a pointer to the list

9.6.2.2 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

|>p0.15|p0.805|

devices is a pointer to the list with devices.

Returns

0 if succefull or error code if fails.

9.7 main.c File Reference

Function prototypes for the console driver.

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

Functions

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

  this main func...................

9.7.1 Detailed Description

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

Author

A.W Janisse

Version

1.0

Date

28-05-2015

9.7.2 Function Documentation

9.7.2.1 int main (int *argc,* char $*$ *argv[]*)

this main func....................

@ return ??????