pico_lora

1.0.0

Generated by Doxygen 1.10.0

Chapter 1

pico_lora

This project has been created to implement a smart lighting system proof of concept for "Progettazione di Sistemi Operativi" exam at University of Milan accademic year 2023/2024.

This particular repository is linked with `STM32_Smart_Lighting_System` which is the other part of the project. This repository in particular contains a firmware for the Raspberry Pi Pico which, with the use of `Wave Share Pico Lora shield`, is used to manage all the communication part of the project. In the roles folder there are two different implementations:

- router is used to send commands to the lighting system from remote
- client is used on the lighting part to receive and send additional data to the router

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

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

Chapter 5

Class Documentation

5.1 Comm Class Reference

High level abstraction layer of the serial communication module.

`#include <comm.hh>`

Public Member Functions

- Comm (Protocol *proto, uint8_t packet_size)
  Class constructor.
- void send_data (uint8_t *data)
  This function is used to send data via the serial interface.
- uint8_t * read_data ()
  This function is used to read the serial interface received data.

5.1.1 Detailed Description

High level abstraction layer of the serial communication module.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Comm()

```
Comm::Comm (
            Protocol * proto,
            uint8_t packet_size )
```

Class constructor.

Parameters

proto	a pointer to a hardware communication protocol abstraction class
packet_size	of the communication protocol defined for the overall application

5.1.3 Member Function Documentation

5.1.3.1 read_data()

```
uint8_t * Comm::read_data ( )
```

This function is used to read the serial interface received data.

Returns

a pointer to the serial interface received data This function return value may be either a pointer to the actual data or, to avoid making this function blocking, if any data has been received, a pointer to an array filled with 0x00. The returned data length will be exactly the same as @refe Comm packet size.

5.1.3.2 send_data()

```
void Comm::send_data (
            uint8_t * data )
```

This function is used to send data via the serial interface.

Parameters

data	the data that will be sent The assumption for the `data` parameter is that the length is the same assumption Comm.

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

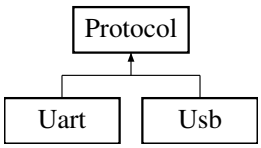
- include/Comm/comm.hh

5.2 Protocol Class Reference

virtual class which represent the basic structure for the serial communication abstraction layer

`#include <protocol.hh>`

Inheritance diagram for Protocol:



Public Member Functions

- virtual void send_data (uint8_t *source, uint8_t size)=0

  This function is used to send data with the serial communication protocol.
- virtual void read_data (uint8_t *dest, uint8_t size)=0

  This function is used to read data from the serial communication protocol.
- virtual bool there_is_data ()=0

  this function is used to check if there is any data on the receive buffer of the serial interface

5.2.1 Detailed Description

virtual class which represent the basic structure for the serial communication abstraction layer

5.2.2 Member Function Documentation

5.2.2.1 read_data()

```
virtual void Protocol::read_data (
            uint8_t * dest,
            uint8_t size )  [pure virtual]
```

This function is used to read data from the serial communication protocol.

Parameters

dest	a buffer to store the received data
size	the amount of data that will be read

Implemented in Uart, and Usb.

5.2.2.2 send_data()

```
virtual void Protocol::send_data (
            uint8_t * source,
            uint8_t size )  [pure virtual]
```

This function is used to send data with the serial communication protocol.

Parameters

source	a buffer containing the data that will be sent
size	the `source` length

Implemented in Uart, and Usb.

5.2.2.3 there_is_data()

```
virtual bool Protocol::there_is_data ( )  [pure virtual]
```

this function is used to check if there is any data on the receive buffer of the serial interface

Implemented in Uart, and Usb.

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

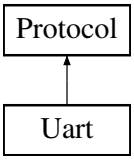
- include/Comm/protocol.hh

5.3 Uart Class Reference

Uart abstraction layer This class is an implementation of the Protocol virtual class, this class can be used with the Comm class to create a full communication protocol abstraction stack.

`#include <uart.hh>`

Inheritance diagram for Uart:



Public Member Functions

- Uart (uart_inst_t *uart=UART_ID, uint br=BAUD_RATE, uint tx=TX_PIN, uint rx=RX_PIN, uint data_↩ bits=DATA_BITS, uint stop_bits=STOP_BITS, uart_parity_t parity=PARITY)

  Uart constructor
- void send_data (uint8_t *, uint8_t) override

  This function is used to send data with the serial communication protocol.
- void read_data (uint8_t *, uint8_t) override

  This function is used to read data from the serial communication protocol.
- bool there_is_data () override

  this function is used to check if there is any data on the receive buffer of the serial interface

5.3.1 Detailed Description

Uart abstraction layer This class is an implementation of the Protocol virtual class, this class can be used with the Comm class to create a full communication protocol abstraction stack.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 Uart()

```
Uart::Uart (
            uart_inst_t * uart = UART_ID,
            uint br = BAUD_RATE,
            uint tx = TX_PIN,
            uint rx = RX_PIN,
            uint data_bits = DATA_BITS,
            uint stop_bits = STOP_BITS,
            uart_parity_t parity = PARITY )
```

Uart constructor

parity bit setting

Parameters

uart	pointer to the uart interface that will be used
br	baudrate that will be used for the communication
tx	tx pin for the uart interface of choice
rx	rx pin for the uart interface of choice
data_bits	number of data bits that will be used in uart communication, it can be in the interval [5, 9]
stop_bits	number of stop bits that will be used for the communication, it can be either 1 or 2
parity	thte parity option which can be any of uart_parity_t defined in `pico-sdk`

5.3.3 Member Function Documentation

5.3.3.1 read_data()

```
void Uart::read_data (
            uint8_t * ,
            uint8_t  )  [override], [virtual]
```

This function is used to read data from the serial communication protocol.

Parameters

dest	a buffer to store the received data
size	the amount of data that will be read

Implements Protocol.

5.3.3.2 send_data()

```
void Uart::send_data (
            uint8_t * ,
            uint8_t  )  [override], [virtual]
```

This function is used to send data with the serial communication protocol.

Parameters

source	a buffer containing the data that will be sent
size	the `source` length

Implements Protocol.

5.3.3.3 there_is_data()

```
bool Uart::there_is_data ( )  [override], [virtual]
```

this function is used to check if there is any data on the receive buffer of the serial interface

Implements Protocol.

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

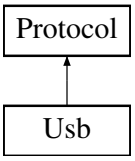
- include/Comm/uart.hh

5.4 Usb Class Reference

usb abstraction layer for the serial communication

`#include <usb.hh>`

Inheritance diagram for Usb:



Public Member Functions

- **Usb** ()

  class constructor
- void send_data (uint8_t *, uint8_t) override

  This function is used to send data with the serial communication protocol.
- void read_data (uint8_t *, uint8_t) override

  This function is used to read data from the serial communication protocol.
- bool there_is_data () override

  this function is used to check if there is any data on the receive buffer of the serial interface

5.4.1 Detailed Description

usb abstraction layer for the serial communication

5.4.2 Member Function Documentation

5.4.2.1 read_data()

```
void Usb::read_data (
            uint8_t * ,
            uint8_t  )  [override], [virtual]
```

This function is used to read data from the serial communication protocol.

Parameters

dest	a buffer to store the received data
size	the amount of data that will be read

Implements Protocol.

5.4.2.2 send_data()

```
void Usb::send_data (
            uint8_t * ,
            uint8_t  )  [override], [virtual]
```

This function is used to send data with the serial communication protocol.

Parameters

source	a buffer containing the data that will be sent
size	the `source` length

Implements Protocol.

5.4.2.3 there_is_data()

```
bool Usb::there_is_data ( )  [override], [virtual]
```

this function is used to check if there is any data on the receive buffer of the serial interface

Implements Protocol.

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

- include/Comm/usb.hh

Chapter 6

File Documentation

6.1 include/Comm/comm.hh File Reference

This file contains all the high level abstraction of the communication module.

```
#include <hardware/uart.h>
#include <protocol.hh>
```

Classes

- class Comm

  High level abstraction layer of the serial communication module.

6.1.1 Detailed Description

This file contains all the high level abstraction of the communication module.

Author

sioel0

Version

1.0.0

6.2 comm.hh

Go to the documentation of this file.

```
00001
00009 #ifndef LIB_HH
00010 #define LIB_HH
00011
00012 #include <hardware/uart.h>
00013 #include <protocol.hh>
00014
00019 class Comm {
00020   private:
00024     Protocol *_proto;
00030     uint8_t _packet_size;
00034     uint8_t* _mcu_received_data;
00038     uint8_t* _NO_DATA_AVAILABLE;
00039   public:
00045     Comm(Protocol* proto, uint8_t packet_size);
00052     void send_data(uint8_t* data);
00060     uint8_t* read_data();
00061 };
00062
00063 #endif
```

6.3 include/Comm/protocol.hh File Reference

This file contains the virtual class which must be implemented to create a serial communication abstraction layer.

```
#include <stdint.h>
```

Classes

- class Protocol

  virtual class which represent the basic structure for the serial communication abstraction layer

6.3.1 Detailed Description

This file contains the virtual class which must be implemented to create a serial communication abstraction layer.

Author

sioel0

6.4 protocol.hh

Go to the documentation of this file.

```
00001
00007 #ifndef PROTOCOL_HPP
00008 #define PROTOCOL_HPP
00009
00010 #include <stdint.h>
00011
00016 class Protocol {
00017
00018   public:
00024     virtual void send_data(uint8_t* source, uint8_t size) = 0;
00025
00031     virtual void read_data(uint8_t* dest, uint8_t size) = 0;
00032
00036     virtual bool there_is_data() = 0;
00037
00038     virtual ~Protocol() {}
00039 };
00040
00041 #endif /* PROTOCOL_HPP */
```

6.5 include/Comm/uart.hh File Reference

This file contains the Uart abstraction layer implementation.

```
#include <hardware/uart.h>
#include <pico/stdlib.h>
#include <cstdint>
#include <protocol.hh>
```

Classes

- class Uart

  Uart abstraction layer This class is an implementation of the Protocol virtual class, this class can be used with the Comm class to create a full communication protocol abstraction stack.

Macros

- #define **UART_ID** uart0 /∗∗ default uart interface ∗/
- #define **TX_PIN** 0 /∗∗ default tx pin ∗/
- #define **RX_PIN** 1 /∗∗ default rx pin ∗/
- #define **BAUD_RATE** 9600 /∗∗ default baudrate ∗/
- #define **DATA_BITS** 8 /∗∗ default data bits number ∗/
- #define **STOP_BITS** 1 /∗∗ default number of stop bits ∗/
- #define **PARITY** UART_PARITY_NONE /∗∗ default parity bit setting ∗/

6.5.1 Detailed Description

This file contains the Uart abstraction layer implementation.

Author

sioel0

6.6 uart.hh

Go to the documentation of this file.

```
00001
00007 #ifndef UART_HH
00008 #define UART_HH
00009
00010 #include <hardware/uart.h>
00011 #include <pico/stdlib.h>
00012 #include <cstdint>
00013 #include <protocol.hh>
00014
00015 #define UART_ID uart0
00016 #define TX_PIN 0
00017 #define RX_PIN 1
00018 #define BAUD_RATE 9600
00020 #define DATA_BITS 8
00021 #define STOP_BITS 1
00022 #define PARITY UART_PARITY_NONE
00031 class Uart : public Protocol {
00032
00033   private:
00034     uart_inst_t *_uart;
00035     uint _br;
00036     uint _tx;
00037     uint _rx;
00038     uint _data_bits;
00039     uint _stop_bits;
00040     uart_parity_t _parity;
00042   public:
00053     Uart(
00054       uart_inst_t *uart = UART_ID,
00055       uint br = BAUD_RATE,
00056       uint tx = TX_PIN,
00057       uint rx = RX_PIN,
00058       uint data_bits = DATA_BITS,
00059       uint stop_bits = STOP_BITS,
00060       uart_parity_t parity = PARITY
00061     );
00062
00068     void send_data(uint8_t*, uint8_t) override;
00069
00075     void read_data(uint8_t*, uint8_t) override;
00076
00080     bool there_is_data() override;
00081 };
00082
00083 #endif
```

6.7 include/Comm/usb.hh File Reference

This file contains the usb abstraction layer implementation.

```
#include <protocol.hh>
#include <cstdint>
```

Classes

- class Usb

  usb abstraction layer for the serial communication

Macros

- #define **READ_TIMEOUT** 100 ∗ 1000 /∗∗ max usb read timeout 100ms∗/
- #define **MAX_BUFF_SIZE** 10 /∗∗ maximum data buffer size ∗/

Functions

- void **received_data** (void ∗)

  data reception callback function

6.7.1 Detailed Description

This file contains the usb abstraction layer implementation.

Author

sioel0

6.8 usb.hh

Go to the documentation of this file.

```
00001
00007 #ifndef USB_HH
00008 #define USB_HH
00009
00010 #include <protocol.hh>
00011 #include <cstdint>
00012
00013 #define READ_TIMEOUT 100 * 1000
00014 #define MAX_BUFF_SIZE 10
00019 void received_data(void*);
00020
00025 class Usb : public Protocol {
00026   public:
00030     Usb();
00031
00037     void send_data(uint8_t*, uint8_t) override;
00038
00044     void read_data(uint8_t*, uint8_t) override;
00045
00049     bool there_is_data() override;
00050 };
00051
00052 #endif
```

Index