Application note

Using UART in interrupt mode with STM32F103 microcontrollers

Introduction

This application note was developed as a work in the discipline of Embedded Systems Programming at UFMG - Prof. Ricardo de Oliveira Duarte - Department of Electronic Engineering.

This document gives an example of how to use the UART peripheral in interrupt mode with an SMT32F103 microcontroller, and explains the concepts involved in the project and the steps you need to follow to develop an application equal or similar to this one.

The application example project was developed to be used with STM32F1 family and was tested with STM32F103C8T6 microcontroller (blue pill). It can be found at the following link: https://github.com/carolmayumimg/STM32F103_UART_IT

Contents

1. Overview

The main task was developing an application capable of receiving a character sequence from the computer via UART and presenting it on an OLED Display, then, sending it back to the computer, so the data travels in both directions. This kind of execution is appropriate to short messages, exchanged at a low frequency. It also can be used as an inspector to the communication's correct implementation.

In this application, we have used a STM32F103C8T6 MCU, an OLED display, two LEDs and a USB - UART TTL converter. The LEDs are used to indicate the transmission or reception of a message, the converter enables the UART communication with a PC and the OLED shows the message received.

This demonstration is useful to new users at the STM environment or at the UART communication, once it is simple and complete. They can learn and understand how the data transfer occurs by executing this program.

At first, this application is a learning tool, but it is possible to use the concept in more complex projects, by adapting or adding new features.

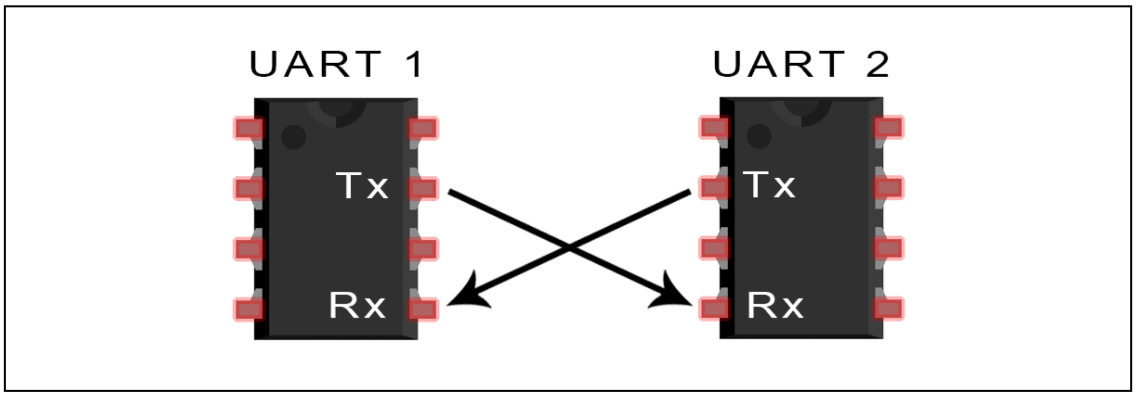
2. Concepts

2.1. UART Communication

In UART communication, two devices are directly connected, and only two wires (connections) are needed. The data is transmitted in serial form, asynchronously, from the RX pin of the transmitting to the RX pin of the receiving. Once there is no mutual clock between the devices, both must be configured with the same baud rate, which is the frequency of the data transfer, given in bits per second. Also, the transmitting adds start and stop bits to the data packet.

In this configuration, both devices can be either transmitter or receiver.

Figure 1 - UART connection



In this application, UART was used in interrupt mode. In this case, every time a message is received or sent, an interruption is generated. This is useful in more complex applications or in the ones where an action must be taken immediately when data is received or sent.

The HAL functions for sending and receiving data via UART in interrupt mode are:

- HAL_UART_Transmit_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size)
- HAL_UART_Receive_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size)

where *huart* is the UART handle used, *pData* is the array containing the data to be transmitted or where the data received will be written, and *Size* is the size of the array.

And the callbacks for each one of the functions above are:

- HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart)
- HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)

These functions are called at the end of the interruption if there were no errors during the process of sending or receiving the data.

2.2. USB – UART TTL Converter

This is an IC that converts the USB into serial communication and vice versa. With this component we are able to communicate the MCU and the PC via UART. It must be connected in an USB port of the computer, and using a software, like RealTerm, it is possible to observe what the PC is receiving and to send data to the microcontroller.

3. Methodology

In the serial communication, the MCU receives the character sequence one by one and shows it on the display. We defined the '\r' character as the end of the message, so when it is detected, this transmission is done, the entire message is on screen, and the MCU sends the received message back to the PC. This resource can be used as an error check if the original and sent message are compared.

The LED associated to the MCU reception blinks when there is message incoming, and the LED associated to the MCU transmitting blinks when it is sending something. This event helps the user get known what is happening without further search.

4. Firmware description

The application firmware was developed using two softwares: STM32CubeMX, where the pinout, peripheral and clock configurations were defined, and System Workbench for STM32, in which the actual code was written.

This project uses two communication protocols: UART, to send and receive data from/to the computer, and I2C, to communicate with the OLED controller. Also, the GPIO peripheral is used to set the two LEDs as outputs. The configurations that must be defined in STM32CubeMX of each one of the peripherals is described below.

- UART: select USART1 in asynchronous mode with a baud rate of 9600 bps, 8 bits of word length, no parity and 1 stop bit. Moreover, since this application uses UART in interrupt mode, you must enable USART1 global interrupt.
- I2C: select I2C1 in fast mode with 400kHz clock speed and a duty cycle equal to 2.
- GPIO: select two unused general ports and set them as GPIO_Output. Change their user label to LED_RX and LED_TX.

The source and include files in the project are from a library [1] used for the OLED display.

User include files:

- ssd1306.h: contains the prototypes of the functions used to write on and deal with the OLED display.
- fonts.h: defines some parameters of some font sizes for the OLED display.

User source files:

- ssd1306.c: contains the functions used to write on and deal with the OLED display.
- fonts.c: includes some font size definitions for the OLED display.

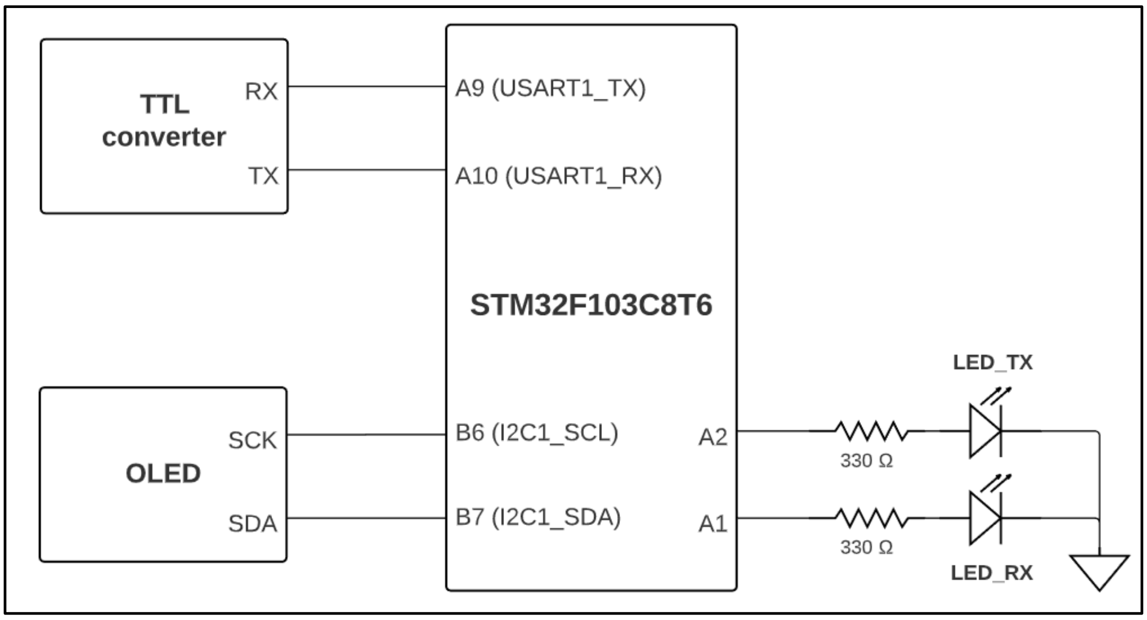
5. Hardware description

The hardware needed for this application are:

- TTL: USB to UART converter
- 2 LED’s
- 2 330Ω resistors (each one connected in series with one LED)
- SSD1306 OLED display

The following figure shows the connections between the devices.

Figure 2 - Schematic of the application

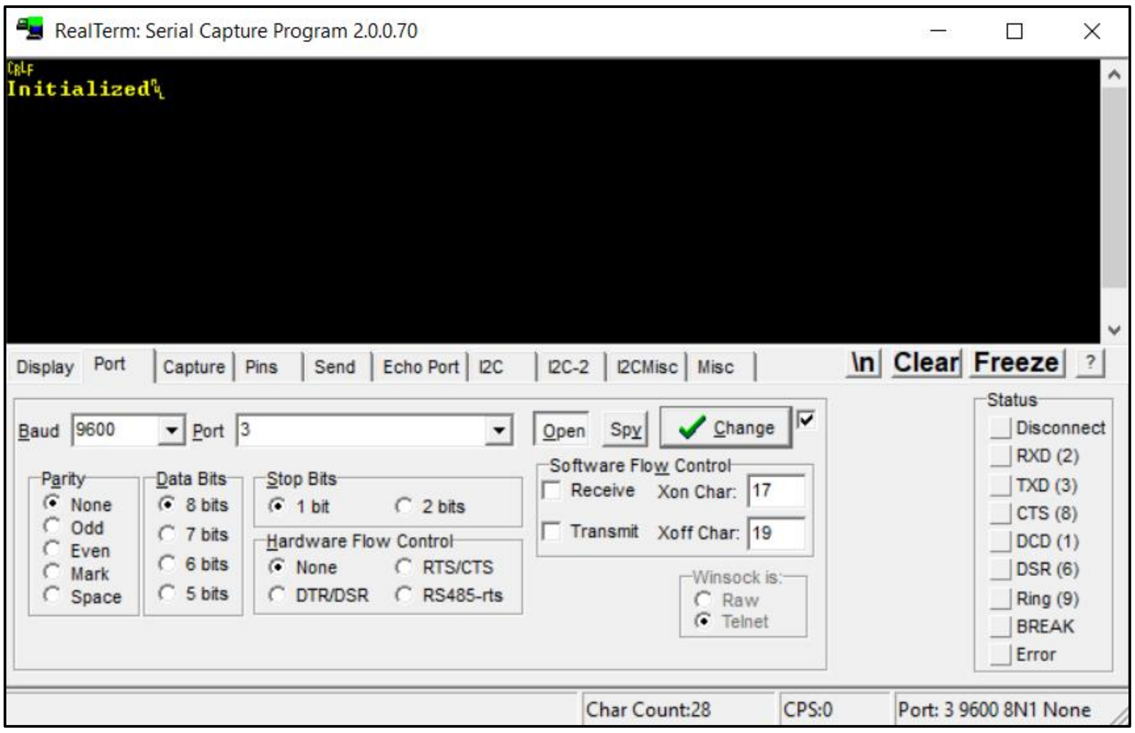


6. Running the application

The software used to send data via UART to the microcontroller and to see what was sent to the computer is RealTerm. To use it, in the 'Port' tab, select a Baud rate of 9600 and the COM port where the TTL converter is connected, then click 'Change'. In this application, the STM32 receives only one character at a time. To send a character in RealTerm, just click in the screen and type the desired character. To send '\r', and end the communication, click ENTER.

When the program starts, the OLED displays “UART Initialized” and the MCU sends “Initialized” to the computer.

Figure 3 - Data displayed in RealTerm when the program starts



RealTerm: Serial Capture Program 2.0.0.70

```
Initialized
```

Baud 9600 Port 3

Parity: [✓ None] Odd Even Mark Space

Data Bits: [✓ 8 bits] 7 bits 6 bits 5 bits

Stop Bits: [✓ 1 bit] 2 bits

Hardware Flow Control: [✓ None] RTS/CTS DTR/DSR RS485-rts

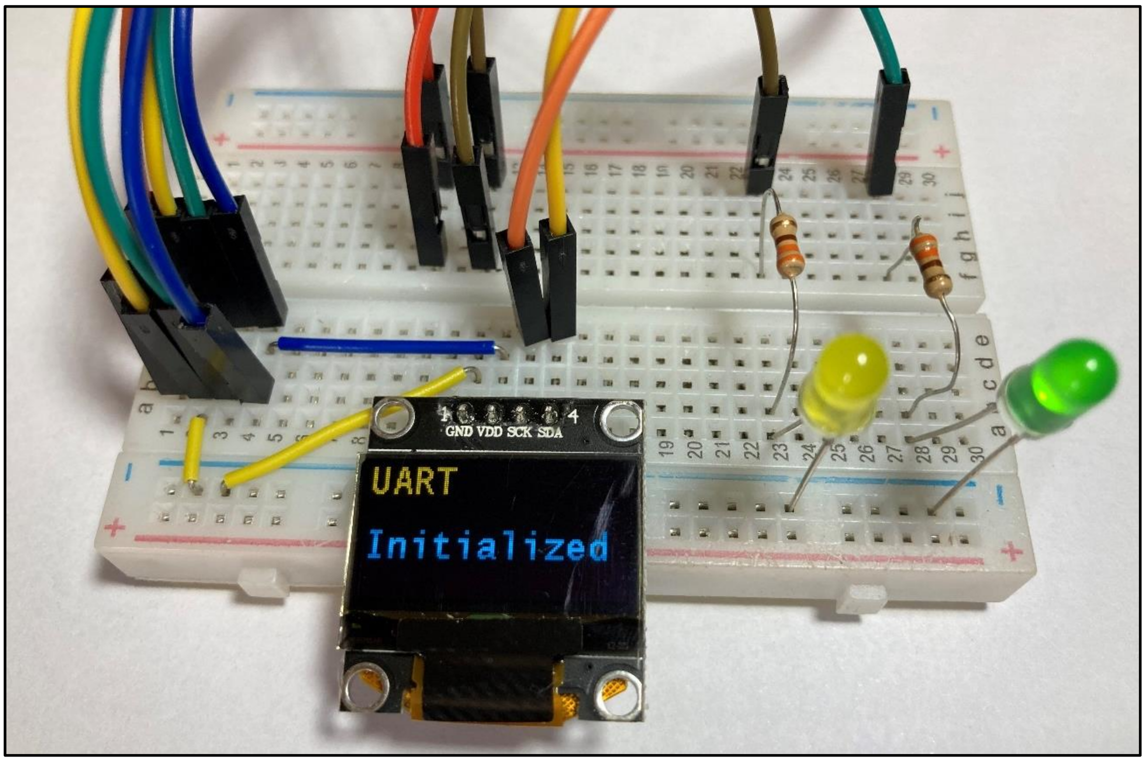
Software Flow Control: Receive Xon Char: 17 Transmit Xoff Char: 19

Winsock is: Raw [✓ Telnet]

Status: Disconnect RXD (2) TXD (3) CTS (8) DCD (1) DSR (6) Ring (9) BREAK Error

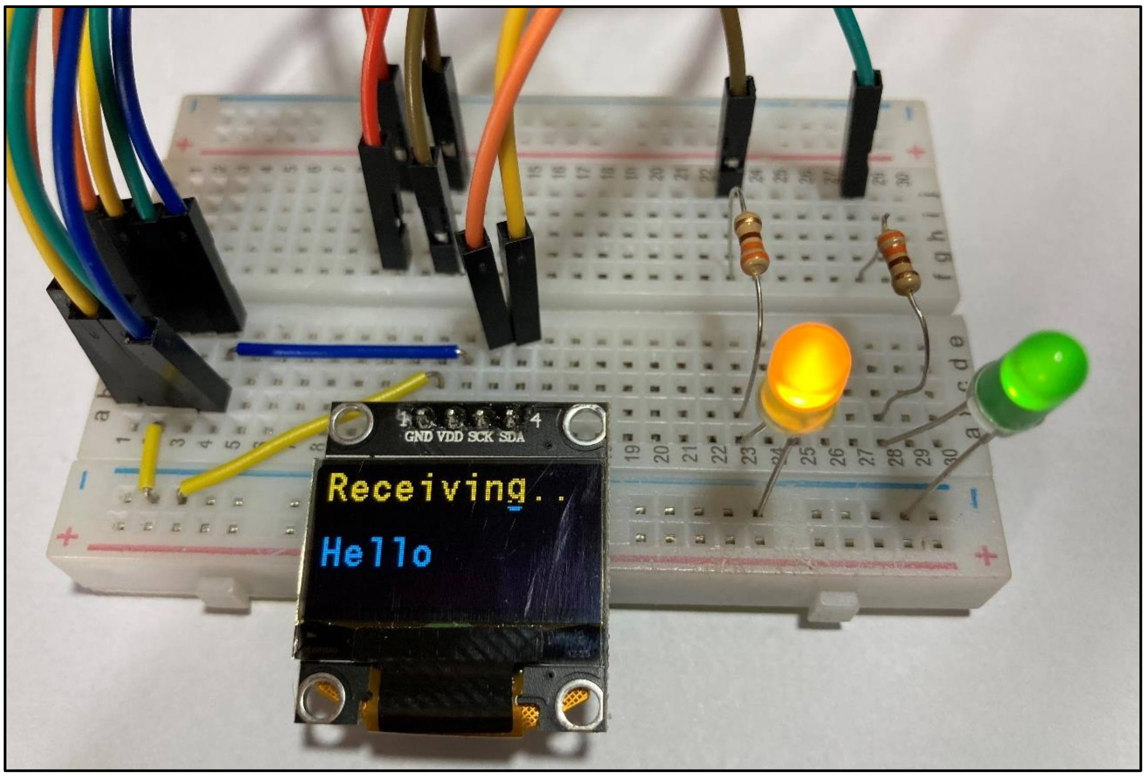
Char Count:28 CPS:0 Port: 3 9600 8N1 None

Figure 4 - OLED display when the program starts



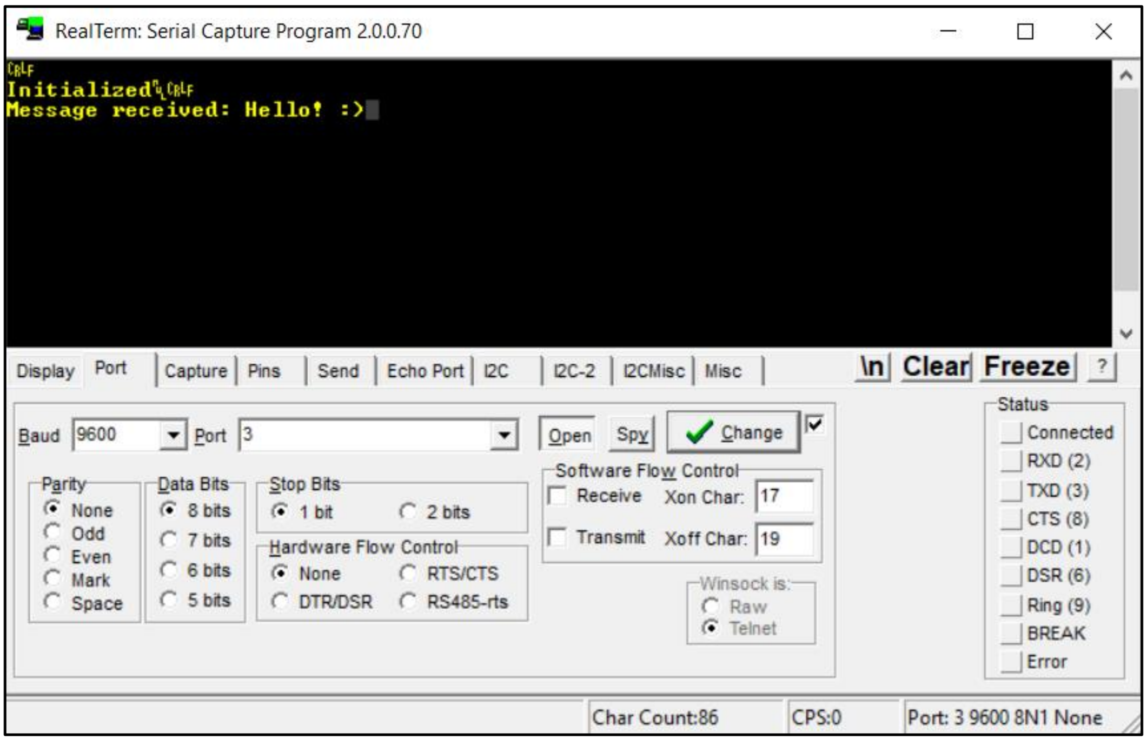
When the microcontroller starts to receive the characters, the OLED displays “Receiving..” and the string formed by the characters received until that moment. Each time a new character is received, the display updates showing the previous string concatenated with the new character. This process repeats until ‘\r’ is received.

Figure 5 - OLED display during the reception of data



When the STM32 receives '\r', the final string is displayed in the OLED and the MCU sends "Message received:" followed by the string to the computer, which is shown in RealTerm.

Figure 6 – Data displayed in RealTerm at the end of the communication



```
CRLF
Initialized LF CRLF
Message received: Hello! :)
```

Baud: 9600 Port: 3

Parity: [✓ None] Odd Even Mark Space

Data Bits: [✓ 8 bits] 7 bits 6 bits 5 bits

Stop Bits: [✓ 1 bit] 2 bits

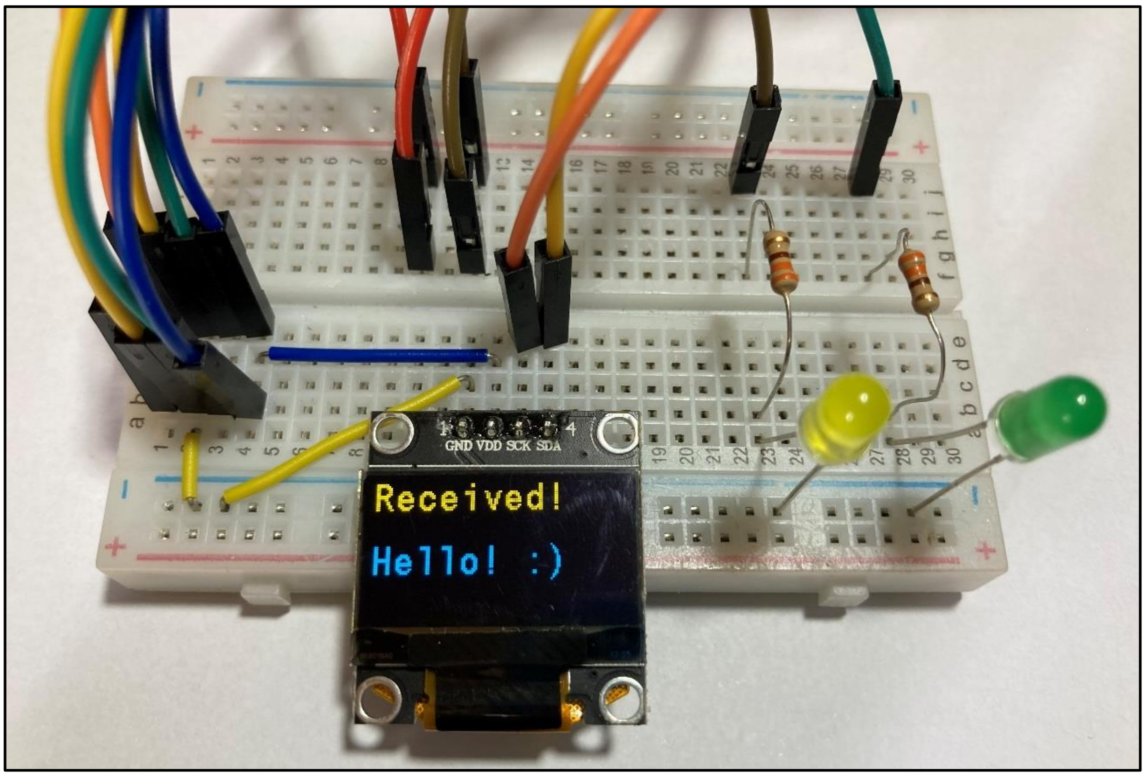
Hardware Flow Control: [✓ None] RTS/CTS DTR/DSR RS485-rts

Xon Char: 17 Xoff Char: 19

Winsock is: Raw [✓ Telnet]

Char Count:86 CPS:0 Port: 3 9600 8N1 None

Figure 7 - OLED display at the end of the communication



7. Abbreviations

MCU	Microcontroller Unit
UART	Universal Asynchronous Receiver/Transmitter
TTL	Transistor Transistor Logic
bps	Bits per second

8. References

[1] OLED SSD1306 library. Available in: https://github.com/SL-RU/stm32libs
[2] UART Communication. Available in: https://www.circuitbasics.com/basics-uart-communication/

9. Revision history

Table 1. Document revision history

Date	Revision	Changes
30-Oct-2020	1	Initial release