UART-GPIO

Date: 28/08/2025	**UART-Gpio Report**
Andrea Zarahi Rubio Quezada	**A01645257**

Introduction

The serial communication UART (Universal Asynchronous Receiver/Transmitter) is one of the protocols more used on embedded systems due to its simplicity and compatibility with devices. Typically, microcontrollers include dedicated UART peripherals to handle data transmission efficiently. In this proyect, however, the UART protocol was implemented via bit-banging- manually toggling a GPIO pin to emulate the transmitter's behavior. This approach provides an in-depth understanding of how UART works at the lowest level and highlights the importance of precise timing in digital communication.

Theory

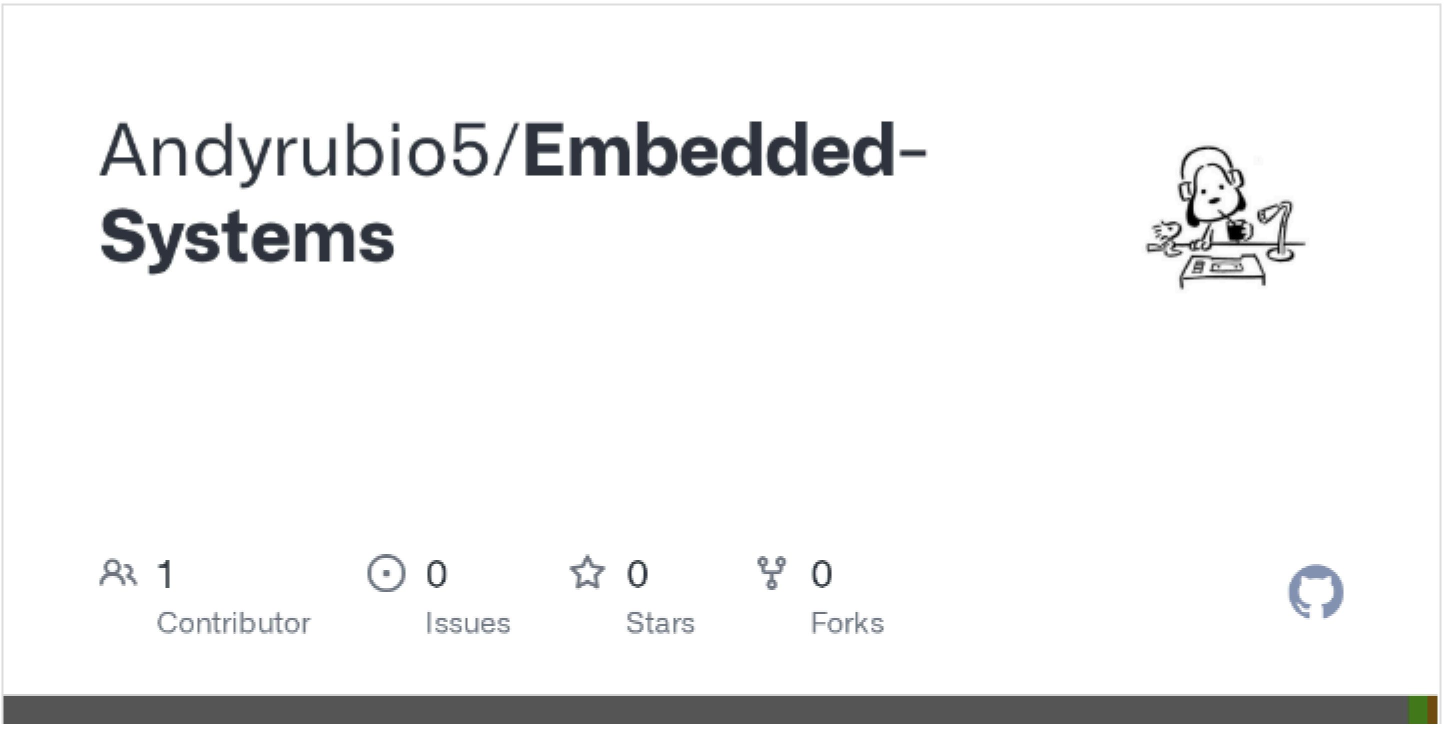
- UART 8N1 Format: The most common UART frame includes:
  - the start bit (logic 0)
  - 8 data bits, transmitted LSB first.
  - 1 stop bit (logic 1)
- Baud Rate: Defines transmission speed in bits per second. At 9600 bps, each bit last approximately 104 us.
- Bit-Banging: A technique where the software manually drivers the GPIO pin to reproduce a communication protocol. While less efficient than hardware peripherals, it is value to educational purpose, debugging, or cases where the hardware lacks UART.
- DWT(Data Watchpoint and Trace): A module in the ARM Cortex-M7 that provides a cycle counter (CYCCNT). It counts CPU cycles and is used here to generate precise timing intervals for UART bit-banging.

Implementation

The system was developed on an STM32H7 Nucleo board. Pin PA5 was selected as the transmission (TX) pin. The program was written in C using CMSIS direct register access, without relying on HAL, to ensure maximum timing accuracy.

Code

On the link you can se de main of the proyect, the main is commented with all the informacion.



Andyrubio5/**Embedded-Systems**

1 Contributor 0 Issues 0 Stars 0 Forks

Key Points of the Code:

- GPIO Configuration: PA5 set as push-pull output, high speed, no pull up/down.
- Helpers tx_high() and tx_low(): inline functions to set or reset the TX pin using the BSRR register.
- timing with DWT: The CYCCNT counter is initialized and used to align each bit to its exact duration.
- UART Transmission (UART_SendByte):
    - Disable interrupts to avoid jitter.
    - Send start bit (low).
    - Send the 8 data bits (LSB first).
    - Send stop bit (high).

    - Re-enable interrupts.

Explanation

The UART_SendByte() function takes one byte and generates the corresponding waveform:

1. **Idle Line**: The line remains high when not transmitting.
2. **Start Bit**: The line is driven low for 1 bit-time to signal the start of a frame.
3. **Data Bits**: Each of the 8 bits of the byte is transmitted from least significant to most significant.
    - A logical 1 → pin set high.
    - A logical 0 → pin set low.
      Each bit is held for exactly one bit-time, calculated using BIT_CYCLES.
4. **Stop Bit**: The line is driven high for at least 1 bit-time.

With this method, the GPIO pin emulates a real UART transmitter, and any UART receiver set to 9600-8N1 can correctly interpret the data.

Results

On a Keysight DSOX1204G oscilloscope, the waveform of the character 'H' (0x48) was observed.

The frame consisted of:

- Start bit = 0.
- Data bits = 00010010 (LSB → MSB, representing 0x48).
- Stop bit = 1.

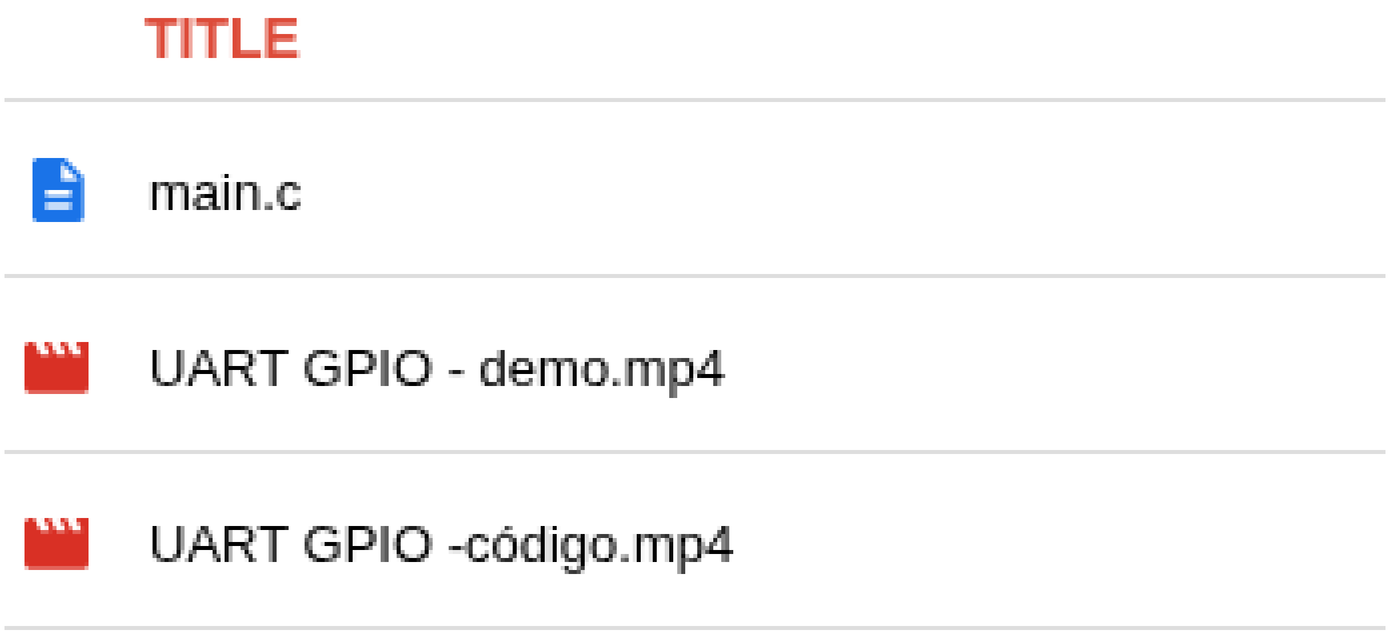
The measured bit duration was approximately 104 µs, consistent with 9600 bps.

Using a logic analyzer with UART decoding at 9600-8N1, the character 'H' was correctly decoded.

The signal was stable, thanks to the use of the DWT and BSRR

Video Result:

On this link is the video explanations and analisis



TITLE

main.c

UART GPIO - demo.mp4

UART GPIO -código.mp4

Conclusion

The implementation of UART using bit-banging on the STM32H7 proved to be both functional and precise, successfully transmitting characters with correct timing at 9600 bps.

Although less efficient than using the built-in UART hardware (since it occupies CPU time and disables interrupts during transmission), this method is an excellent way to:

- Understand how UART frames are structured at the bit level.
- Appreciate the importance of precise timing in digital communications.

- Learn to leverage the DWT cycle counter for exact delays in embedded applications.

This project demonstrates that with simple GPIO operations and cycle-accurate timing, a full communication protocol can be emulated purely in software, strengthening both theoretical and practical understanding of embedded system design.