Muhammad Saad Malik

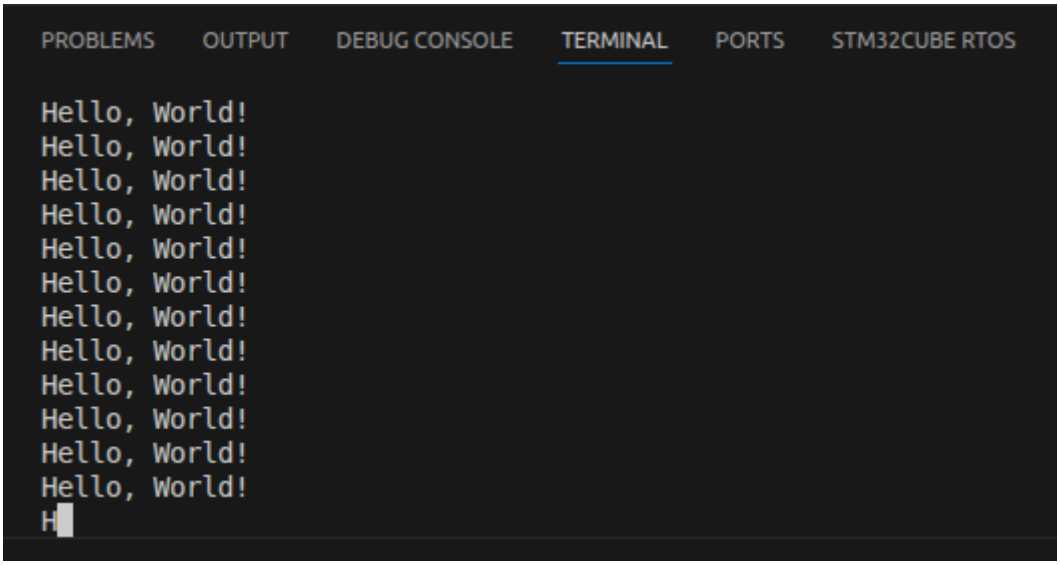
Muhammad Hassan Shahid

Code Changes Snippets:

```c
#include "main.h"
#include "usb_device.h"
#include "stm32f3xx_hal_uart.h"
```

```c
while (1)
{
  /* USER CODE END WHILE */
  HAL_UART_Transmit(huart: &huart1, pData: (uint8_t *)"Hello, World!\r\n", Size: 15, Timeout: HAL_MAX_DELAY);
  /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
}
```

Results Terminal Snippets:



Evaluation Questions:

- In the Hello World task, why is HAL_UART _Transmit() placed inside the while(1) loop, and what could go wrong if it’s placed outside?

Answer:

It’s placed inside while loop to repeatedly send the "Hello, World!" message forever — the code in while(1) is the program's main loop, so placing HAL_UART_Transmit() there causes continuous/transmitted output.

If we place it outside loop:

- It will run only once at startup (not continuously).
- If called before MX_USART1_UART_Init() or before clocks/GPIO for USART are ready, huart1 is uninitialized and the call can fail or cause undefined behavior.
- Using HAL_MAX_DELAY in a blocking transmit before system/USB/RTOS init can hang startup (blocks indefinitely if TX never completes), preventing other initializations or watchdog servicing.
- In RTOS/interrupt-driven systems, a blocking call outside the main loop can interfere with scheduler/interrupt startup.
- Without delays or non-blocking APIs, moving it outside could either spam once at an unexpected time or produce no visible output if the receiver isn't ready.

- You used the command screen /dev/ttyACM0 115200 in this lab. What is the purpose of this command, and what would happen if the baud rate specified does not match the one configured in STM32CubeMX?

Answer:

Purpose: screen /dev/ttyACM0 115200 opens a serial terminal on the host connected to the USB/serial device node /dev/ttyACM0 and configures the host-side UART parameters (baud = 115200, default 8N1). It lets you view and send raw serial data to the STM32 over that port.

If the baud rate doesn't match the one configured in CubeMX (BaudRate = 115200):

- Output becomes garbled or unreadable (junk characters) because bit timing differs.
- You may see framing errors or lost/corrupted bytes; sometimes a few characters decode correctly, sometimes none.
- At large mismatches communication effectively fails (appears like no output).
- Small mismatches can produce intermittent errors rather than total failure.

- Mismatched parity/stop-bit settings produce similar corruption even with the same baud.