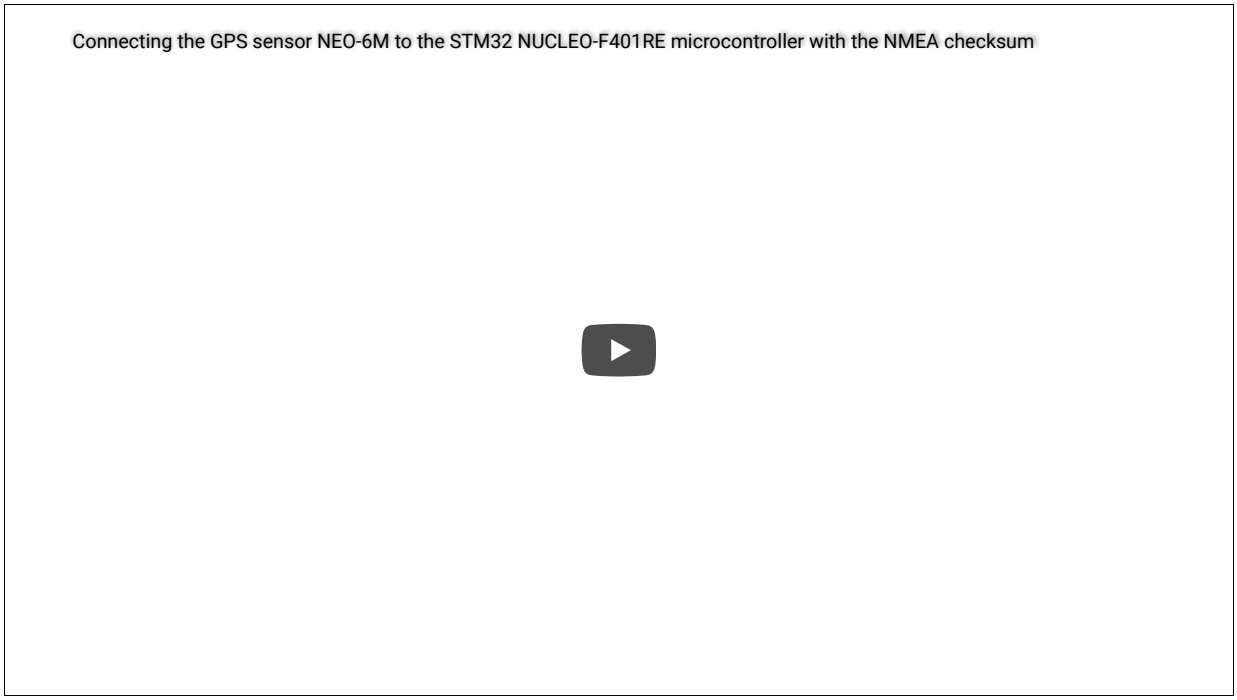
Connecting the GPS sensor NEO-6M to the STM32 NUCLEO-F401RE microcontroller with the NMEA 0183 checksum validation, in STM32CubeIDE.



The GPS NEo-6M board pins are connected to the NUCLEO-F401RE microcontroller the following way: GND to GND, VCC to 3V3, TX to D2 (PA10), [the GPS' RX is not connected]. Do not forget to enable for the UART1 in the NVIC settings both - the global interrupt and the DMA interrupt. I talk about at 10:03 in the video. We use the UART1 for receiving the GPS data. It can be another UART's number if another microcontroller is used.

Connecting the GPS sensor NEO-6M to the STM32 NUCLEO-F401RE microcontroller with the NMEA 0183 checksum validation, in STM32CubeIDE (the part of the code, which was generated automatically by the configurator, is not included)

```
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

#include "string.h"
#include "stdlib.h"
#include "stdio.h"

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

uint8_t flag = 0;

// this interrupts changes flag to 1 as soon as the uint8_t buff[300] is full
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart) {

    flag = 1;

}

// function to calculate checksum of the NMEA sentence
// -4, but not -3 because the NMEA sentences are delimited with \r\n, and there also is the invisible \r in the end
int nmea0183_checksum(char *msg) {

    int checksum = 0;
    int j = 0;

    // the first $ sign and the last two bytes of original CRC + the * sign
    for (j = 1; j < strlen(msg) - 4; j++) {
            checksum = checksum ^ (unsigned) msg[j];
    }

    return checksum;
}

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
UART_HandleTypeDef huart1;
UART_HandleTypeDef huart2;
DMA_HandleTypeDef hdma_usart1_rx;
```

```
/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_DMA_Init(void);
static void MX_USART1_UART_Init(void);
static void MX_USART2_UART_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
 * @brief  The application entry point.
 * @retval int
 */
int main(void) {
    /* USER CODE BEGIN 1 */

    /* USER CODE END 1 */

    /* MCU Configuration--------------------------------------------------------*/

    /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
    HAL_Init();

    /* USER CODE BEGIN Init */

    /* USER CODE END Init */

    /* Configure the system clock */
    SystemClock_Config();

    /* USER CODE BEGIN SysInit */

    /* USER CODE END SysInit */

    /* Initialize all configured peripherals */
    MX_GPIO_Init();
    MX_DMA_Init();
    MX_USART1_UART_Init();
    MX_USART2_UART_Init();
    /* USER CODE BEGIN 2 */

    uint8_t buff[255];
    char buffStr[255];
    char nmeaSnt[80];

    char *rawSum;
    char smNmbr[3];

    // The Equator has a latitude of 0°,
    //the North Pole has a latitude of 90° North (written 90° N or +90°),
    //and the South Pole has a latitude of 90° South (written 90° S or −90°)

    char *latRaw;
    char latDg[2];
    char latMS[7];
    char *hemNS;

    // longitude in degrees (0° at the Prime Meridian to +180° eastward and −180° westward)
    // that is why 3
    char *lonRaw;
    char lonDg[3];
    char lonMS[7];
    char *hemEW;

    char *utcRaw; // raw UTC time from the NMEA sentence in the hhmmss format
    char strUTC[8]; // UTC time in the readable hh:mm:ss format

    char hH[2]; // hours
    char mM[2]; // minutes
    char sS[2]; // seconds

    uint8_t cnt = 0;

    HAL_UART_Receive_DMA(&huart1, buff, 255);

    /* USER CODE END 2 */

    /* Infinite loop */
    /* USER CODE BEGIN WHILE */
    while (1) {
            if (flag == 1) { // interrupt signals that the buffer buff[300] is full

                    /*

                     $ - Start delimiter
                     * - Checksum delimiter
                     , - Field delimiter

                     1. $GNGLL log header
                     2. Latitude (Ddmm.mm) [The Equator has a latitude of 0°, the North Pole has a latitude of 90° North (written 90° N or +90°)]
                     3. Latitude direction (N = North, S = South)
                     4. Longitude (DDDmm.mm) [0° at the Prime Meridian to +180° eastward and −180° westward]
                     5. Longitude direction (E = East, W = West)
                     6. UTC time status of position (hours/minutes/seconds/decimal seconds) hhmmss
                     7. Data status: A = Data valid, V = Data invalid
                     8. Positioning system mode indicator
                     9. *xx Checksum
                     10. [CR][LF] Sentence terminator. In C \r\n (two characters).
                      or \r Carriage return
                      or \n Line feed, end delimiter

                     */

                    memset(buffStr, 0, 255);

                    sprintf(buffStr, "%s", buff);

                    // if we want to display the incoming raw data
```

```
//HAL_UART_Transmit(&huart2, buff, 255, 70);

// splitting the buffStr by the "\n" delimiter with the strsep() C function
// see http://www.manpagez.com/man/3/strsep/
char *token, *string;

string = strdup(buffStr);

// actually splitting the string by "\n" delimiter
while ((token = strsep(&string, "\n")) != NULL) {

        memset(nmeaSnt, 0, 80);

        sprintf(nmeaSnt, "%s", token);

        // selecting only $GNGLL sentences, combined GPS and GLONASS
        // on my GPS sensor this good NMEA sentence is always 50 characters
        if ((strstr(nmeaSnt, "$GNGLL") != 0) && strlen(nmeaSnt) > 49 && strstr(nmeaSnt, "*") != 0) {

                rawSum = strstr(nmeaSnt, "*");

                memcpy(smNmbr, &rawSum[1], 2);

                smNmbr[2] = '\0';

                uint8_t intSum = nmea0183_checksum(nmeaSnt);

                char hex[2];

                // "%X" unsigned hexadecimal integer (capital letters)
                sprintf(hex, "%X", intSum);

                // checksum data verification, if OK, then we can really trust
                // the data in the the NMEA sentence
                if (strstr(smNmbr, hex) != NULL) {

                        //if we want display good $GNGLL NMEA sentences
                        //HAL_UART_Transmit(&huart2, nmeaSnt, 50, 70);
                        //HAL_UART_Transmit(&huart2, (uint8_t*) "\n", 1, 200);

                        cnt = 0;

                        // splitting the good NMEA sentence into the tokens by the comma delimiter
                        for (char *pV = strtok(nmeaSnt, ","); pV != NULL; pV = strtok(NULL, ",")) {

                                switch (cnt) {
                                case 1:
                                        latRaw = strdup(pV);
                                        break;
                                case 2:
                                        hemNS = strdup(pV);
                                        break;
                                case 3:
                                        lonRaw = strdup(pV);
                                        break;
                                case 4:
                                        hemEW = strdup(pV);
                                        break;
                                case 5:
                                        utcRaw = strdup(pV);
                                        break;
                                }

                                cnt++;

                        }  // end for()

                        memcpy(latDg, &latRaw[0], 2);
                        latDg[2] = '\0';

                        memcpy(latMS, &latRaw[2], 7);
                        latMS[7] = '\0';

                        memcpy(lonDg, &lonRaw[0], 3);
                        lonDg[3] = '\0';

                        memcpy(lonMS, &lonRaw[3], 7);
                        lonMS[7] = '\0';
                        char strLonMS[7];
                        sprintf(strLonMS, "%s", lonMS);

                        //converting the UTC time in the hh:mm:ss format
                        memcpy(hH, &utcRaw[0], 2);
                        hH[2] = '\0';

                        memcpy(mM, &utcRaw[2], 2);
                        mM[2] = '\0';

                        memcpy(sS, &utcRaw[4], 2);
                        sS[2] = '\0';

                        strcpy(strUTC, hH);
                        strcat(strUTC, ":");
                        strcat(strUTC, mM);
                        strcat(strUTC, ":");
                        strcat(strUTC, sS);
                        strUTC[8] = '\0';

                        HAL_UART_Transmit(&huart2, (uint8_t*) hemNS, 1, 200);
                        HAL_UART_Transmit(&huart2, (uint8_t*) " ", 1, 200);
                        HAL_UART_Transmit(&huart2, (uint8_t*) latDg, 2, 200);
                        HAL_UART_Transmit(&huart2, (uint8_t*) "\241", 1, 200);
                        HAL_UART_Transmit(&huart2, (uint8_t*) latMS, 7, 200);
                        HAL_UART_Transmit(&huart2, (uint8_t*) "\', ", 3, 200);

                        HAL_UART_Transmit(&huart2, (uint8_t*) hemEW, 1, 200);
                        HAL_UART_Transmit(&huart2, (uint8_t*) " ", 1, 200);
                        HAL_UART_Transmit(&huart2, (uint8_t*) lonDg, 3, 200);
                        HAL_UART_Transmit(&huart2, (uint8_t*) "\241", 1, 200);
                        HAL_UART_Transmit(&huart2, (uint8_t*) strLonMS, strlen(strLonMS), 200);
                        HAL_UART_Transmit(&huart2, (uint8_t*) "\', UTC: ", 8, 200);

                        HAL_UART_Transmit(&huart2, (uint8_t*) strUTC, 8, 200);
                        HAL_UART_Transmit(&huart2, (uint8_t*) "\n", 1, 200);

                } // end of of the checksum data verification

        } // end of $GNGLL sentences selection
```

```
                        } // end of splitting the buffStr by the "\n" delimiter with the strsep() C function
                        flag = 0; // we are ready to get new data from the sensor
                    } // end of one interrupt/full-buffer cycle
                    HAL_Delay(200);
                    /* USER CODE END WHILE */
                    /* USER CODE BEGIN 3 */
                }
                /* USER CODE END 3 */
}
```