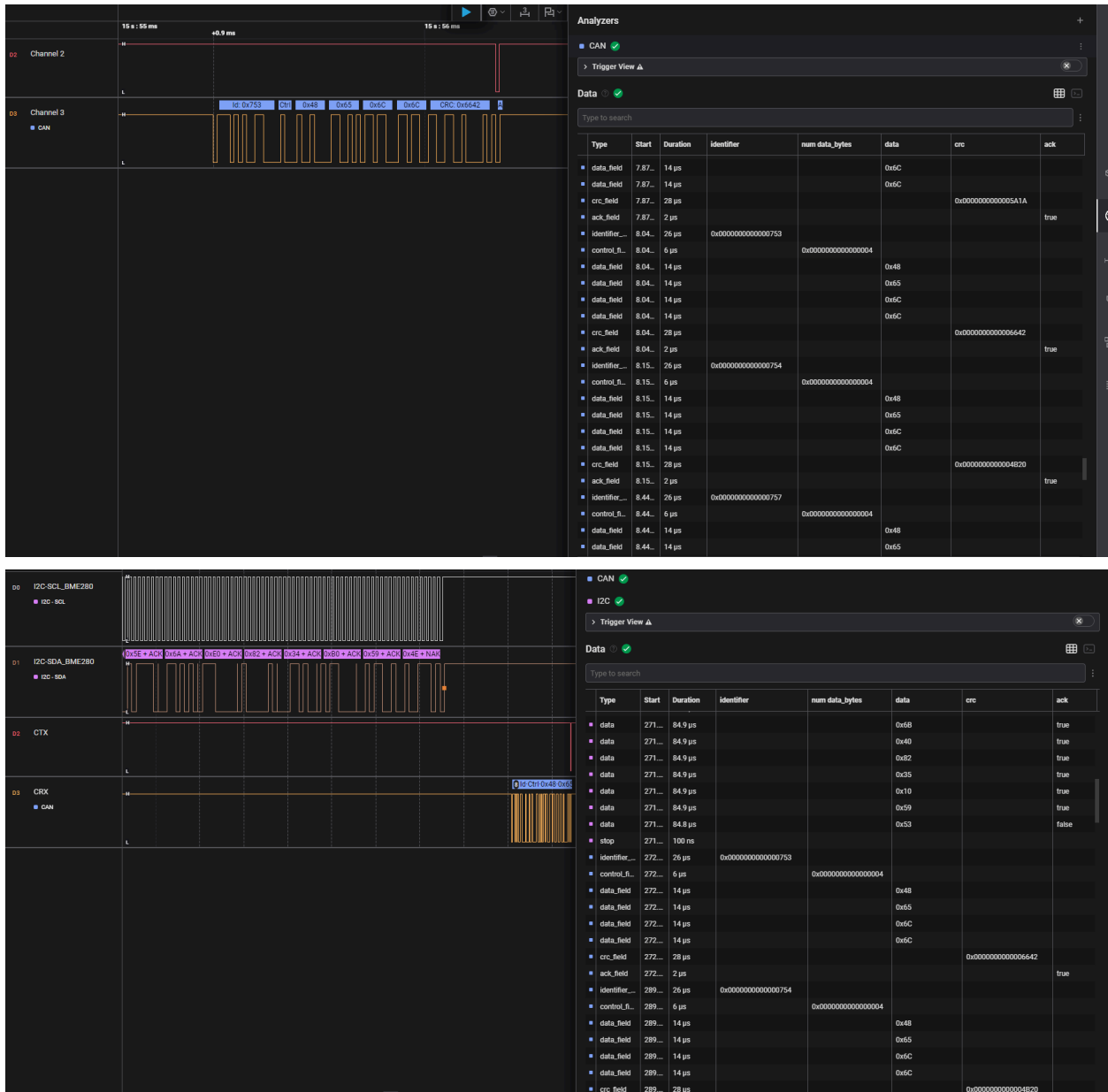
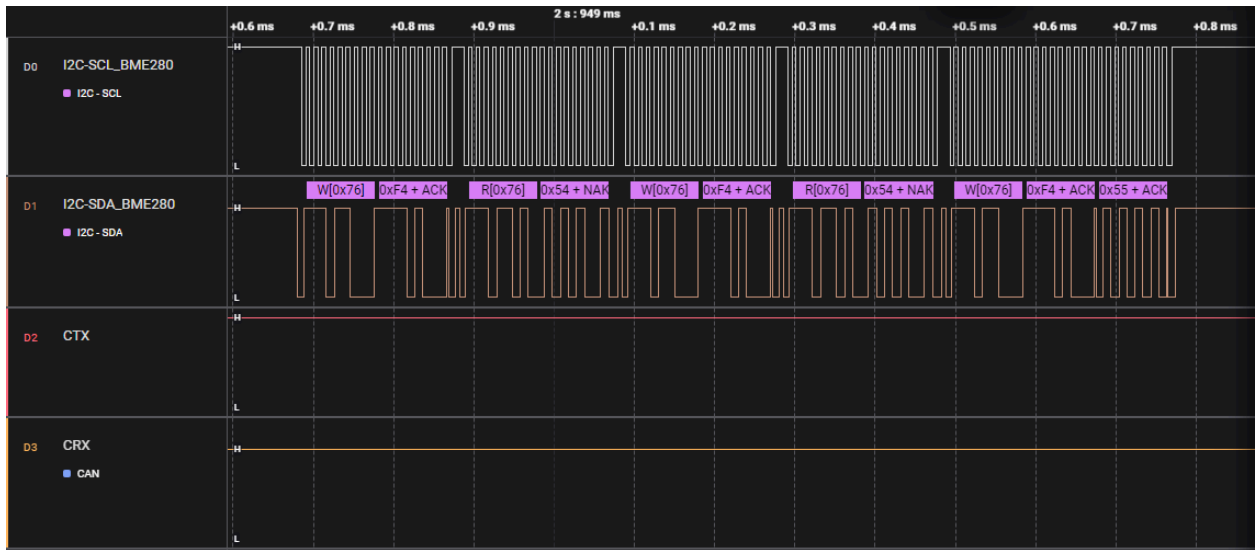
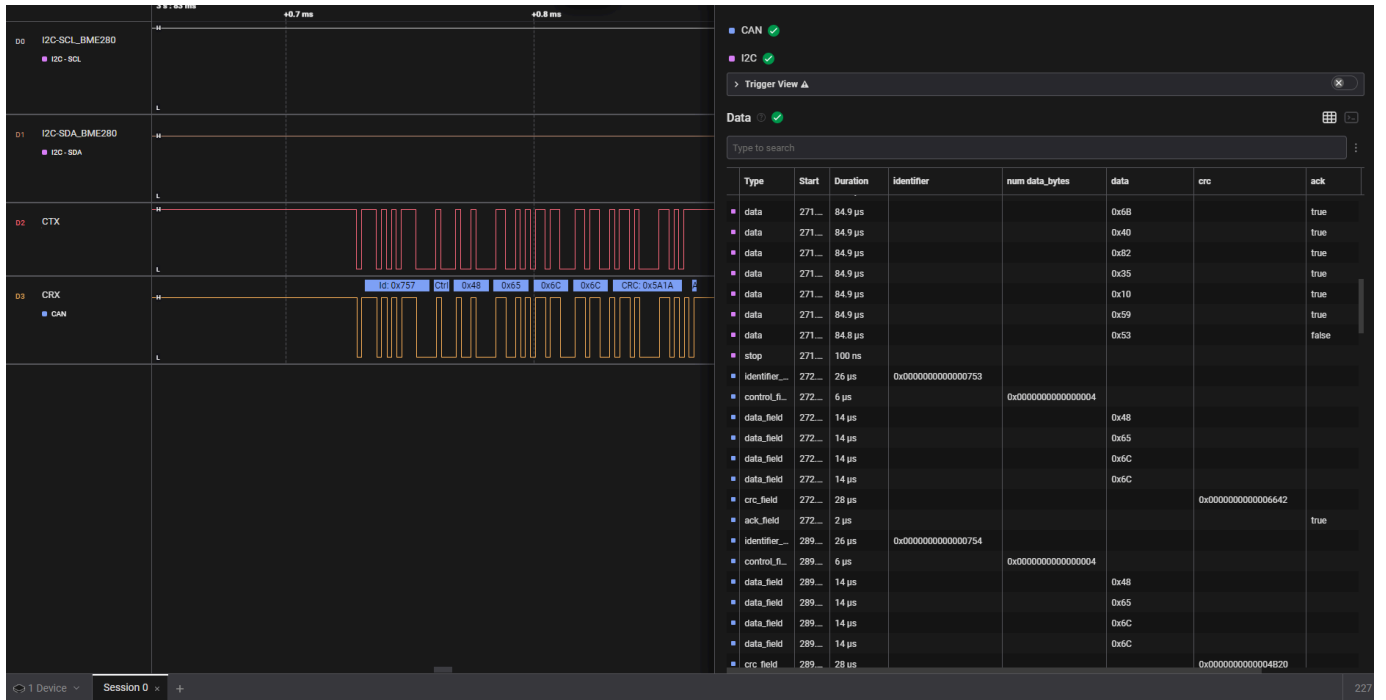


Лабораторна Робота CAN

Скріни з LA, які показують, що ми бачимо три девайси і вони між собою комунікують





Скріни з STM32CubeIDE, як свідоцтво того, що дані з сенсора валідно читаються, “в кан заходимо”:

The screenshot shows the STM32CubeIDE interface. The main editor displays C code for reading sensor data from a BME280 sensor. The code includes a function `bme280_get_sensor_data` that returns sensor data and a `while` loop that reads data and prints it to the console. The console output shows the received data: "Received: 100.127". The variable watch window on the right shows the expression `s_rx_data` with a value of `0x20000250` and a type of `char [128]`. Below the watch window, the details for the expression `[100...127]` are shown, including its decimal, hex, binary, and octal representations.

```

rslt = bme280_get_sensor_data(BME280_ALL, &comp_data, &dev);

if (rslt == BME280_OK) {
    temperature = comp_data.temperature / 100.0;
    humidity = comp_data.humidity / 1024.0;
    pressure = comp_data.pressure / 10000.0;

    sprintf(temp_string, "Temperature: %03.1f C", temperature);
    sprintf(hum_string, "Humidity: %03.1f %%", humidity);
    sprintf(press_string, "Pressure: %03.1f hPa", pressure);
}

if (s_data_received) {
    sprintf(s_usb_buf, "Received: %s\r\n", s_rx_data);
    HAL_CAN_AddTxMessage(&hcan, &s_tx_header, (uint8_t *)s_usb_buf, &s_tx_mailbox);
    s_data_received = 0;

    HAL_GPIO_TogglePin(LED_PIN_GPIO_Port, LED_PIN_Pin);
}

```

Expression	Type	Value
s_rx_data	char [128]	0x20000250 <s_rx_data>
[0...99]	char [100]	0x20000250 <s_rx_data>
[100...127]	char [28]	0x200002b4 <s_rx_data+100>

Name : [100...127]
Details: '\0' <repeats 27 times>
Default: 0x200002b4 <s_rx_data+100>
Decimal: 536871604
Hex: 0x200002b4
Binary: 100000000000000000001010110100
Octal: 04000001264

The screenshot shows the STM32CubeIDE interface. The main editor displays C code for setting the sensor mode and reading data. The code includes a function `bme280_set_sensor_mode` that sets the sensor mode to `BME280_FORCED_MODE` and a `while` loop that reads data and prints it to the console. The console output shows the received data: "Received: 100.127". The variable watch window on the right shows the expression `temp_string` with a value of `50` and a type of `char [50]`. Below the watch window, the details for the expression `temp_string` are shown, including its decimal, hex, binary, and octal representations.

```

while (1) {
    rslt = bme280_set_sensor_mode(BME280_FORCED_MODE, &dev);
    HAL_Delay(40);
    rslt = bme280_get_sensor_data(BME280_ALL, &comp_data, &dev);

    if (rslt == BME280_OK) {
        temperature = comp_data.temperature / 100.0;
        humidity = comp_data.humidity / 1024.0;
        pressure = comp_data.pressure / 10000.0;

        sprintf(temp_string, "Temperature: %03.1f C", temperature);
        sprintf(hum_string, "Humidity: %03.1f %%", humidity);
        sprintf(press_string, "Pressure: %03.1f hPa", pressure);
    }

    if (s_data_received) {
        sprintf(s_usb_buf, "Received: %s\r\n", s_rx_data);
        HAL_CAN_AddTxMessage(&hcan, &s_tx_header, (uint8_t *)s_usb_buf, &s_tx_mailbox);
        s_data_received = 0;

        HAL_GPIO_TogglePin(LED_PIN_GPIO_Port, LED_PIN_Pin);
    }
}

```

Expression	Type	Value
temp_string	char [50]	50
temp_string[0]	char	84 'T'
temp_string[1]	char	101 'e'
temp_string[2]	char	109 'm'
temp_string[3]	char	112 'p'
temp_string[4]	char	101 'e'
temp_string[5]	char	114 'r'
temp_string[6]	char	97 'a'
temp_string[7]	char	116 't'
temp_string[8]	char	117 'u'
temp_string[9]	char	114 'r'
temp_string[10]	char	101 'e'
temp_string[11]	char	58 ':'
temp_string[12]	char	32 ' '
temp_string[13]	char	32 ' '
temp_string[14]	char	67 'C'
temp_string[15]	char	0 '\0'
temp_string[16]	char	0 '\0'