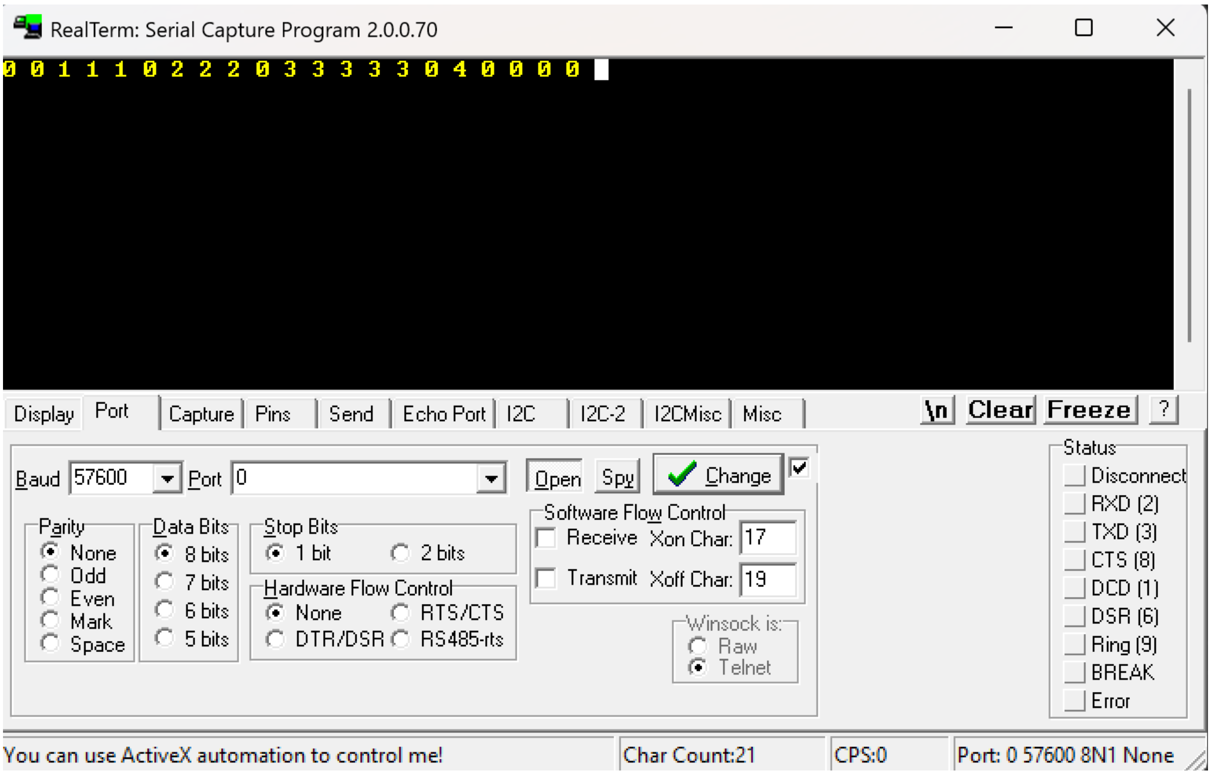
Embedded assignment 7

By Liubomir Stanev

Task 7a:

After pressing buttons or not pressing anything in RealTerm the following numbers are received:



RealTerm: Serial Capture Program 2.0.0.70

```
0 0 1 1 1 0 2 2 2 0 3 3 3 3 3 0 4 0 0 0 0
```

Baud 57600 Port 0

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

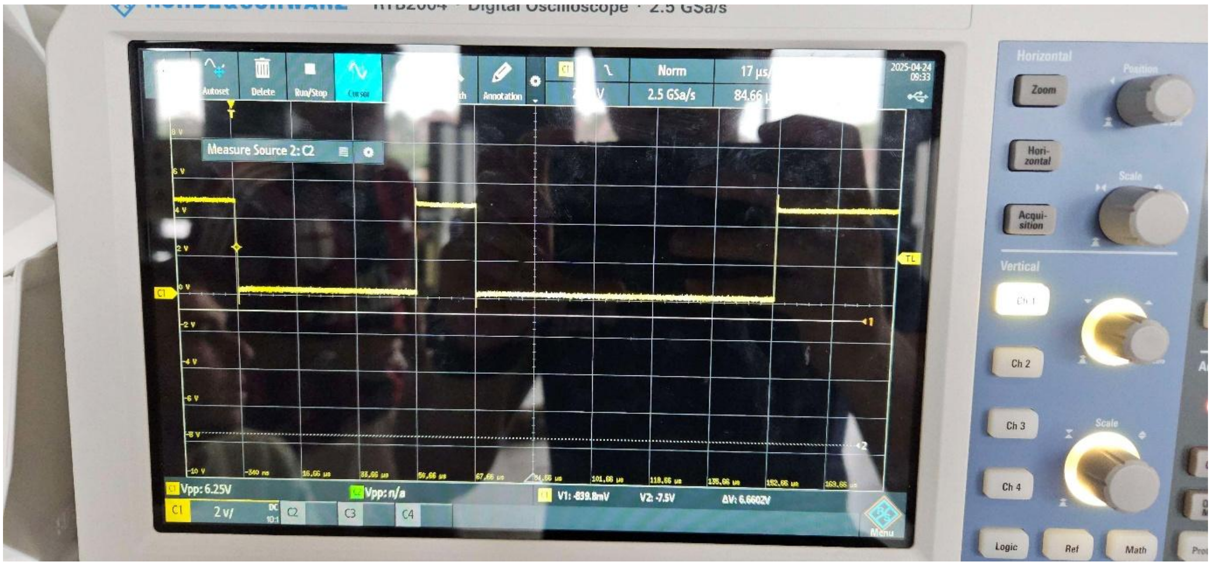
Software Flow Control: □ Receive Xon Char: 17 □ Transmit Xoff Char: 19

Winsock is: ○ Raw ● Telnet

You can use ActiveX automation to control me! | Char Count:21 | CPS:0 | Port: 0 57600 8N1 None

Task 7b and c:

On the following picture, start, stop bits and signals bits can be seen for the 8-bit message between the two Arduinos:



src\main.c

```c
#define F_CPU 16000000UL

#include <stdio.h>
#include <avr/io.h>
#include <util/delay.h>
#include <time.h>
#include <avr/interrupt.h>
#include "usart.h"
#include "i2cmaster.h"
#include "lcd.h"
#include "lm75.h"

#define BAUDRATE 57600
#define BAUD_PRESCALER (((F_CPU / (BAUDRATE * 16UL))) -1)

void usart_init(void);
void usart_send(unsigned char data);



int main(void) {
   uart_init();
   //io_redirect();
   usart_init();
   //i2c_init();
   //LCD_init();
   //lm75_init();

   //shield board init
   DDRD = 0b11110000;
   DDRC = 0x00;
   PORTC = 0b00001111;

   //DDRB = 0b00100000;  //internal LED
   //PORTB = 0b00100000; //internal LED
   //PCICR |= (1 << PCIE1);    // set PCIE0 to enable the group for PCINT7..PCINT0
   //PCMSK0 |= (1 << PCINT8) | (1 << PCINT9) | (1 << PCINT10) | (1 << PCINT11);  // Enable only PCINT0 interrupt from the group
   //UCSR0B|= (1<<RXCIE0);//enable interrupts for RXIE
   //sei(); //enable interrupts

   while (1)
   {
      if (!(PINC & (1 << PC0)))
      {
         usart_send(1);
         _delay_ms(1000);
      }
      else if (!(PINC & (1 << PC1)))
      {
         usart_send(2);
         _delay_ms(1000);
      }
      else if (!(PINC & (1 << PC2)))
      {
         usart_send(3);
         _delay_ms(1000);
      }
      else if (!(PINC & (1 << PC3)))
      {
         usart_send(4);
         _delay_ms(1000);
      }
      else
      {
         usart_send(0);
         _delay_ms(1000);
      }
   }
}

void usart_init(void)
{
   UBRR0H = (uint8_t)(BAUD_PRESCALER>>8);
   UBRR0L = (uint8_t)(BAUD_PRESCALER);
   UCSR0B = (1<<RXEN0)|(1<<TXEN0);
   UCSR0C = ((1<<UCSZ00)|(1<<UCSZ01));
}

void usart_send(unsigned char data)
{
   while(!(UCSR0A & (1<<UDRE0))); //wait for transmit buffer
   UDR0 = data; //data to be sent
}

```

src\main.c

```c
#define F_CPU 16000000UL

#include <stdio.h>
#include <avr/io.h>
#include <util/delay.h>
#include <time.h>
#include <avr/interrupt.h>
#include "usart.h"
#include "i2cmaster.h"
#include "lcd.h"
#include "lm75.h"

#define BAUDRATE 57600
#define BAUD_PRESCALER (((F_CPU / (BAUDRATE * 16UL))) -1)

void usart_init(void);
void usart_send(unsigned char data);



int main(void) {
   uart_init();
   //io_redirect();
   usart_init();
   //i2c_init();
   //LCD_init();
   //lm75_init();

   //shield board init
   DDRD = 0b11110000;
   DDRC = 0x00;
   PORTC = 0b00001111;

   //DDRB = 0b00100000;  //internal LED
   //PORTB = 0b00100000; //internal LED

   UCSR0B|= (1<<RXCIE0);//enable interrupts for RXIE
   sei(); //enable interrupts

   while (1)
   {
      if (!(PINC & (1 << PC0)))
      {
         usart_send(1);
         _delay_ms(1000);
      }
      else if (!(PINC & (1 << PC1)))
      {
         usart_send(2);
         _delay_ms(1000);
      }
      else if (!(PINC & (1 << PC2)))
      {
         usart_send(3);
         _delay_ms(1000);
      }
      else if (!(PINC & (1 << PC3)))
      {
         usart_send(4);
         _delay_ms(1000);
      }
      else
      {
         usart_send(0);
         _delay_ms(1000);
      }
   }
}

void usart_init(void)
{
   UBRR0H = (uint8_t)(BAUD_PRESCALER>>8);
   UBRR0L = (uint8_t)(BAUD_PRESCALER);
   UCSR0B = (1<<RXEN0)|(1<<TXEN0);
   UCSR0C = ((1<<UCSZ00)|(1<<UCSZ01));
}

void usart_send(unsigned char data)
{
   while(!(UCSR0A & (1<<UDRE0))); //wait for transmit buffer
   UDR0 = data; //data to be sent
}

ISR(USART_RX_vect)
{
   volatile unsigned char received_data = UDR0;
   if (received_data == 1) PORTD |= (1 << PORTD4); //led1 on
   if (received_data == 2) PORTD |= (1 << PORTD5); //led2 on
   if (received_data == 3) PORTD |= (1 << PORTD6); //led3 on
   if (received_data == 4) PORTD |= (1 << PORTD7); //led4 on
   if (received_data == 0) PORTD &= ~((1 << PORTD4) | (1 << PORTD5) | (1 << PORTD6) | (1 << PORTD7)); // turn off every LED

}

```