EMBEDDED SYSTEMS LAB SPRING 2021

DATE : 17TH MARCH 2021
NAME : AKULA SRI SAI KIRAN
ROLL : 18EE10007
OBJECTIVE : DATA COMMUNICATION USING USART and send "I I T Kharagpur" at different baudrates.

APPARATUS :

1. ARDUINO IDE in a LAPTOP
2. ARDUINO UNO ATMEGA328P BOARD
3. USB CONNECTING WIRE

ALGORITHM :

We have to communicate through USART, so will make use of USART0. First, we enable the transmitter and then choose the asynchronous mode of communication and then as each extended ASCII character is 8-bits length, we select 8bits. We have to wait till UDR is empty and ready to send the data so we use a while loop which waits till UDR is empty and after it is free we send the next ASCII character.

19.10.4 UCSRnC – USART Control and Status Register n C

Bit	7	6	5	4	3	2	1	0	
	UMSELn1	UMSELn0	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

- **Bits 7:6 – UMSELn1:0 USART Mode Select**

These bits select the mode of operation of the USARTn as shown in Table 19-4.

Table 19-4. UMSELn Bits Settings

UMSELn1	UMSELn0	Mode
0	0	Asynchronous USART
0	1	Synchronous USART
1	0	(Reserved)
1	1	Master SPI (MSPIM)(1)

Note: 1. See Section 20. "USART in SPI Mode" on page 166 for full description of the master SPI mode (MSPIM) operation

Table 19-7. UCSZn Bits Settings

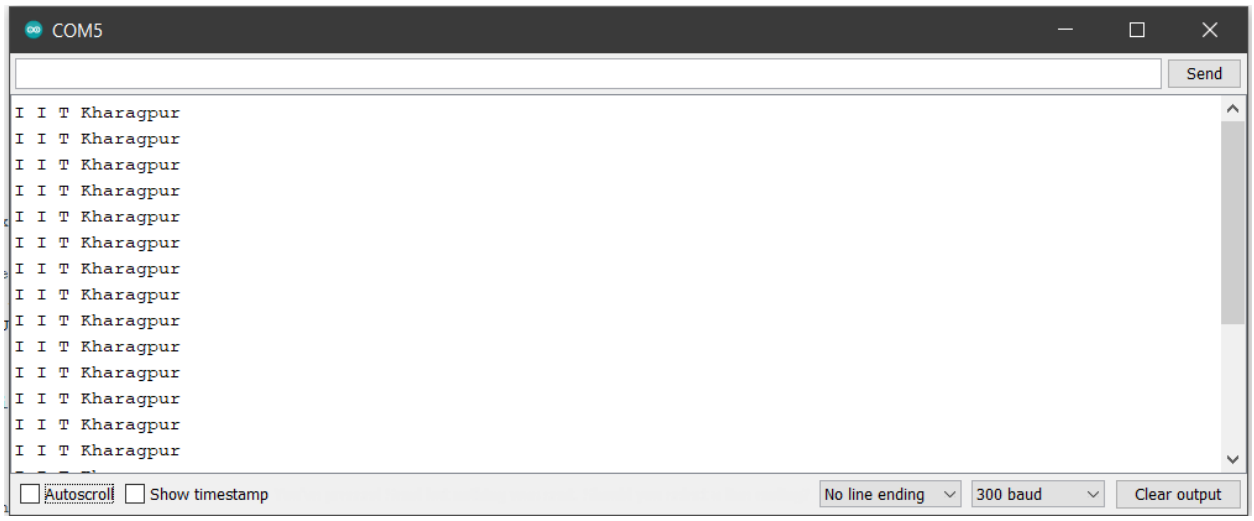
UCSZn2	UCSZn1	UCSZn0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

CODE :

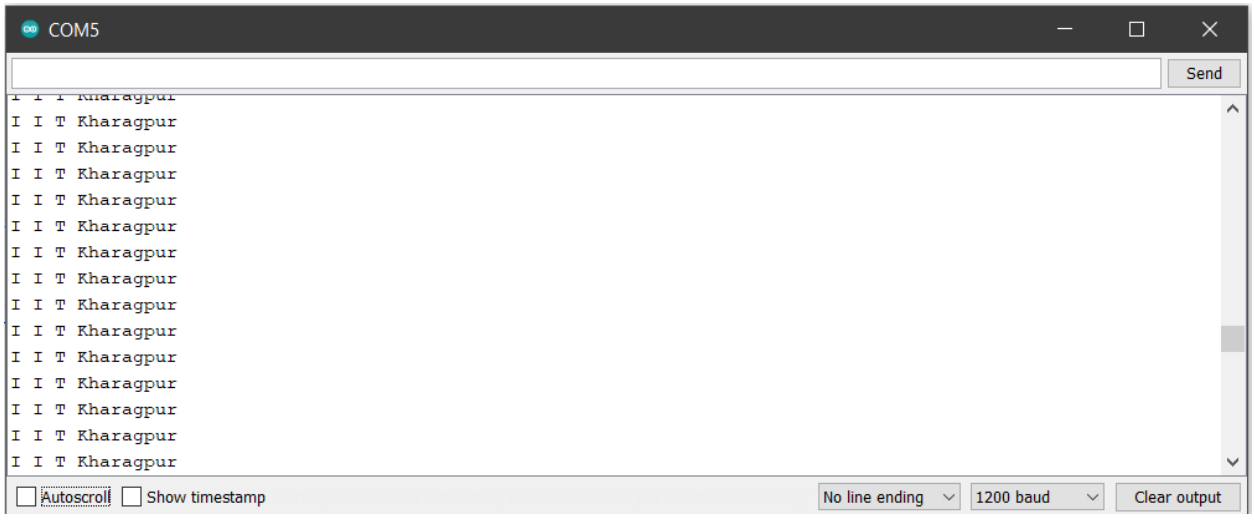
```
#include <avr/io.h>                          // header file
unsigned char req[]="I I T Kharagpur\n";
                                  // string in a character array
unsigned char length = 16;        //  the string has 16 charecters
unsigned char i=0;
                        //  index for traversing through the string

void setup()
{
      // put your setup code here, to run once:
        UCSR0B |= (1<<TXEN0);
      //TXEN0-Transmitter Enable (Tx pin)
        UCSR0C &= (~(1<<UMSEL01))|(~(1<<UMSEL00))|(~(1<<UCSZ02));
      //UMSEL01:UMSEL01 = 0b00 -> Asynchronous Mode
        UCSR0C |= (1<<UCSZ00)|(1<<UCSZ01);
      //  UCSZ02:UCSZ00  = 0b011  -> 8-bit

  UBRR0H = 0x00;
  UBRR0L = 0x67;
  //Buad Rate (Bd) = fosc/(16(UBBRn + 1), fosc = 16MHz,
  //Baud Rate = 300   UBBRn = ((10^6)/300)-1          = 3332  = 0xD04
  //Baud Rate = 1200  UBBRn = ((10^6)/1200)-1         = 832   = 0x340
  //Baud Rate = 2400  UBBRn = ((10^6)/2400)-1         = 415   = 0x19F
  //Baud Rate = 4800  UBBRn = ((10^6)/4800)-1         = 207   = 0xCF
  //Baud Rate = 9600  UBBRn = ((10^6)/9600)-1         = 103   = 0x67
  //Baud Rate = 19200   UBBRn = ((10^6)/19200)-1      = 51  = 0x33
  //Baud Rate = 38400   UBBRn = ((10^6)/38400)-1      = 25  = 0x19
  //Baud Rate = 57600   UBBRn = ((10^6)/57600)-1      = 16  = 0x10
  //Baud Rate = 74880   UBBRn = ((10^6)/74880)-1      = 12  = 0x0C
  //Baud Rate = 115200  UBBRn = ((10^6)/115200)-1     = 7   = 0x07
}
```

```
void loop()
{
  // put your main code here, to run repeatedly:
  while(!(UCSR0A & (1<<UDRE0)));  // wait till UDR is empty

  UDR0 = req[i];                  // send the ASCII charecter
  i = (i+1)%length;
                        // prepare the index for the next ASCII charecter

}
```

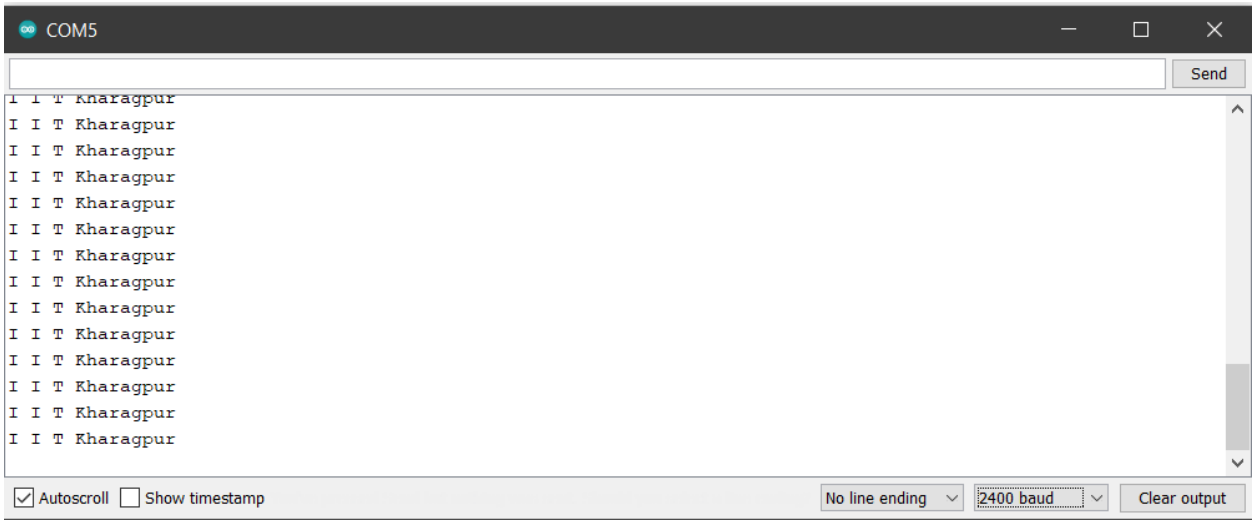
RESULTS :



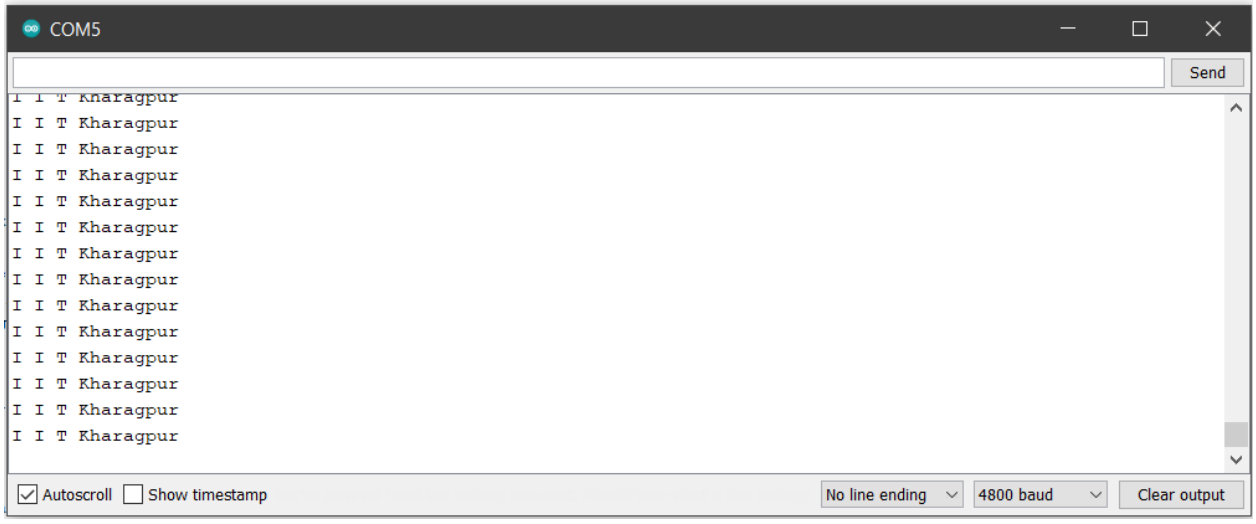
BAUDRATE = 300 , UBRR0H:UBRR0L = 0xD04



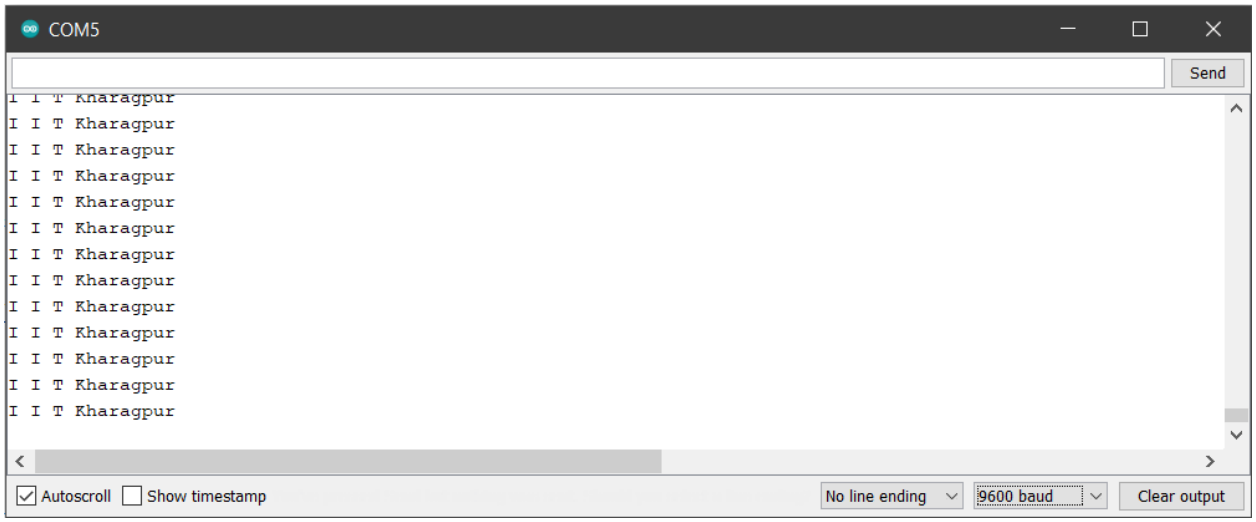
BAUDRATE = 1200 , UBRR0H:UBRR0L = 0x340



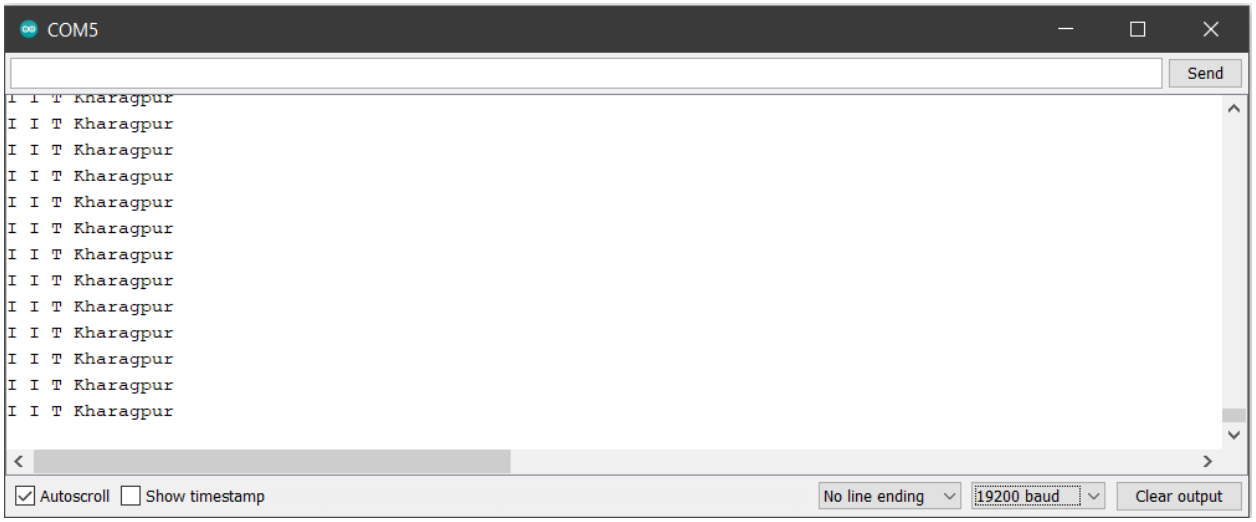
BAUDRATE = 2400 , UBRR0H:UBRR0L = 0x19F



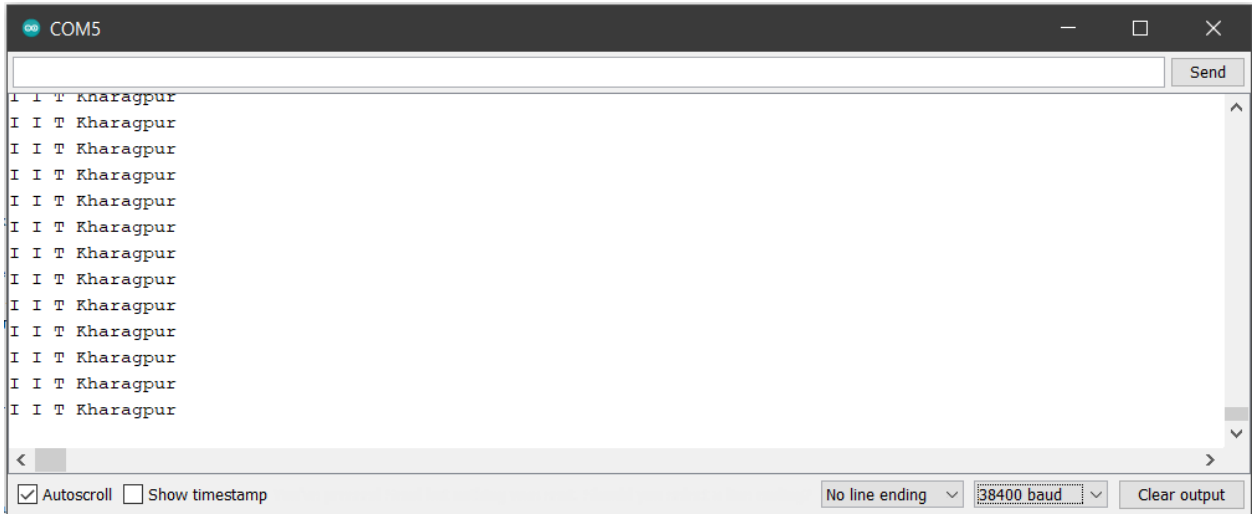
BAUDRATE = 4800, UBRR0H:UBRR0L = 0x00CF



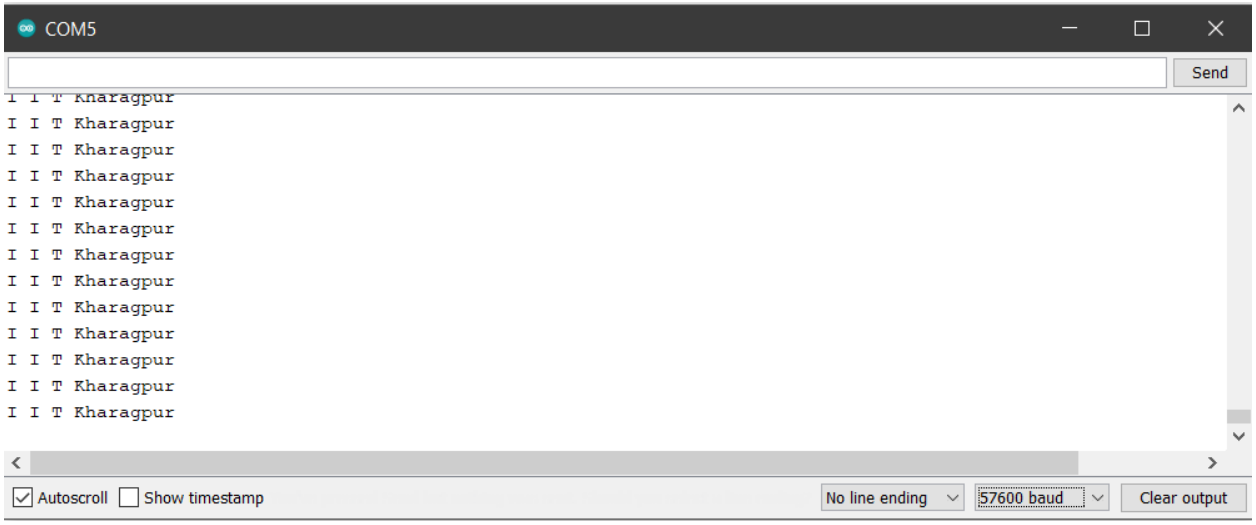
BAUDRATE = 9600 , UBRR0H:UBRR0L = 0x0067



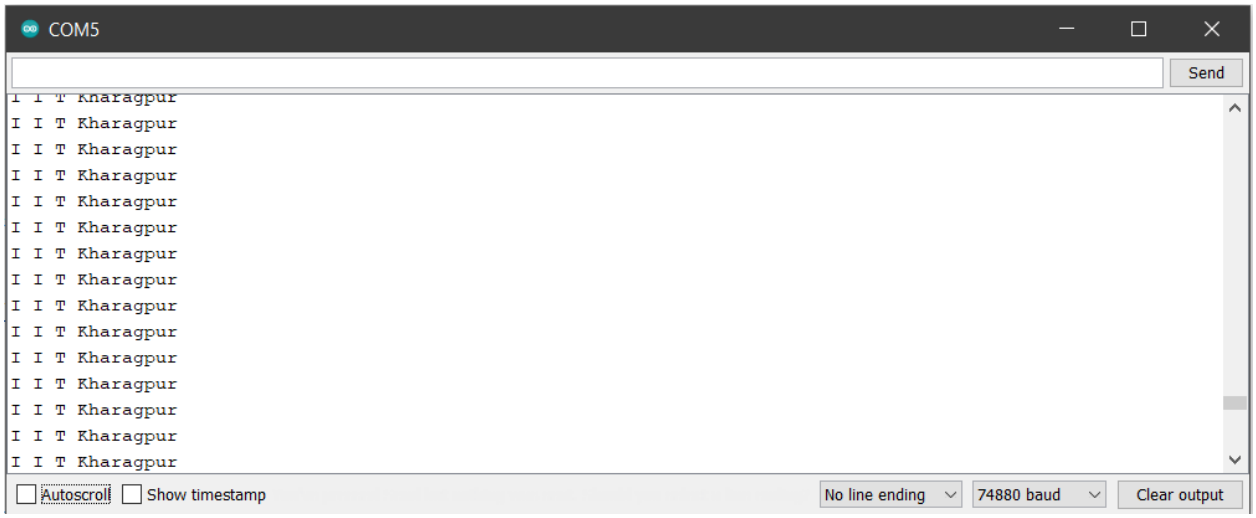
BAUDRATE = 19200 , UBRR0H:UBRR0L = 0x0033



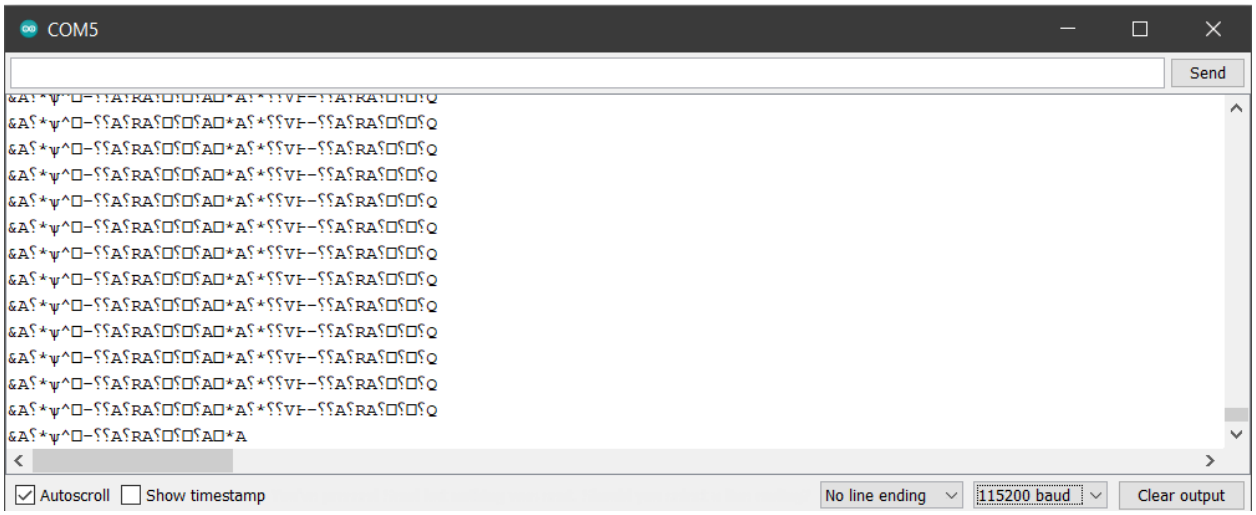
BAUDRATE = 38400 , UBRR0H:UBRR0L = 0x0019



BAUDRATE = 57600 , UBRR0H:UBRR0L = 0x0010



BAUDRATE = 74880 , UBRR0H:UBRR0L = 0x000C



BAUDRATE = 115200 , UBRR0H:UBRR0L = 0x0007

DISCUSSIONS :

1. **The baud rate is the rate at which information is transferred in a communication channel.**
2. **We change the baud rates by changing the UBRRn values corresponding to the desired baud rate which we need.**
3. **We have used Serial Monitor to see the transmission of ASCII characters.**
4. **As we increase the baud rate till 57600 everything was fine but from 74800 I was able to see some error in serial monitor, and at 115200 the whole output in the serial monitor is an error, so I can say that the maximum baud rate that my Arduino can go for this communication is perfect till 57600 and slightly with some errors in 74800.**

For video simulation in ARDUINO click here: USART_18EE10007