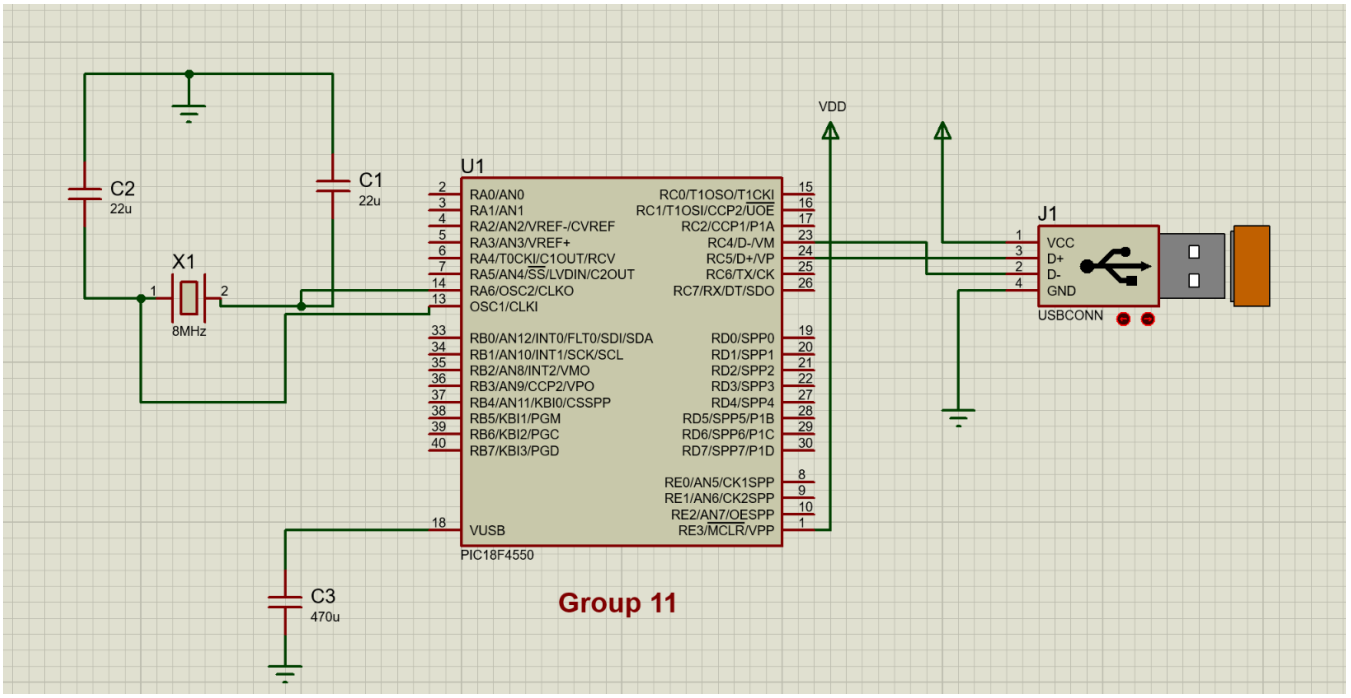
CO326 - Industrial Networks
Lab 06 - USB Port I/O

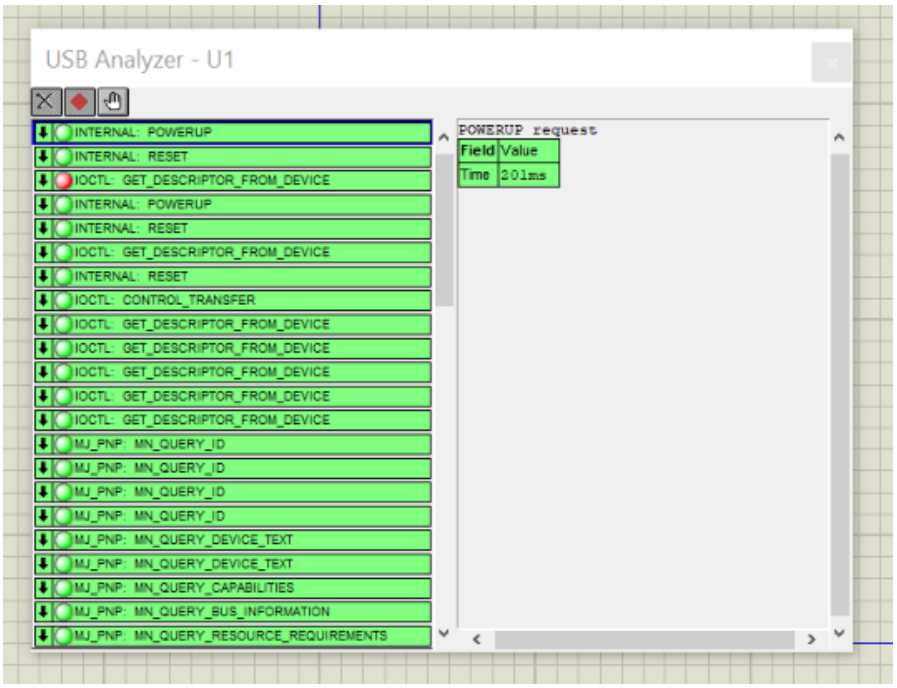
E17219 Nawarathna K.G.I.S.
E17212 Morais K.W.G.A.N.D.
E17230 Nishankar S.

Lab Example

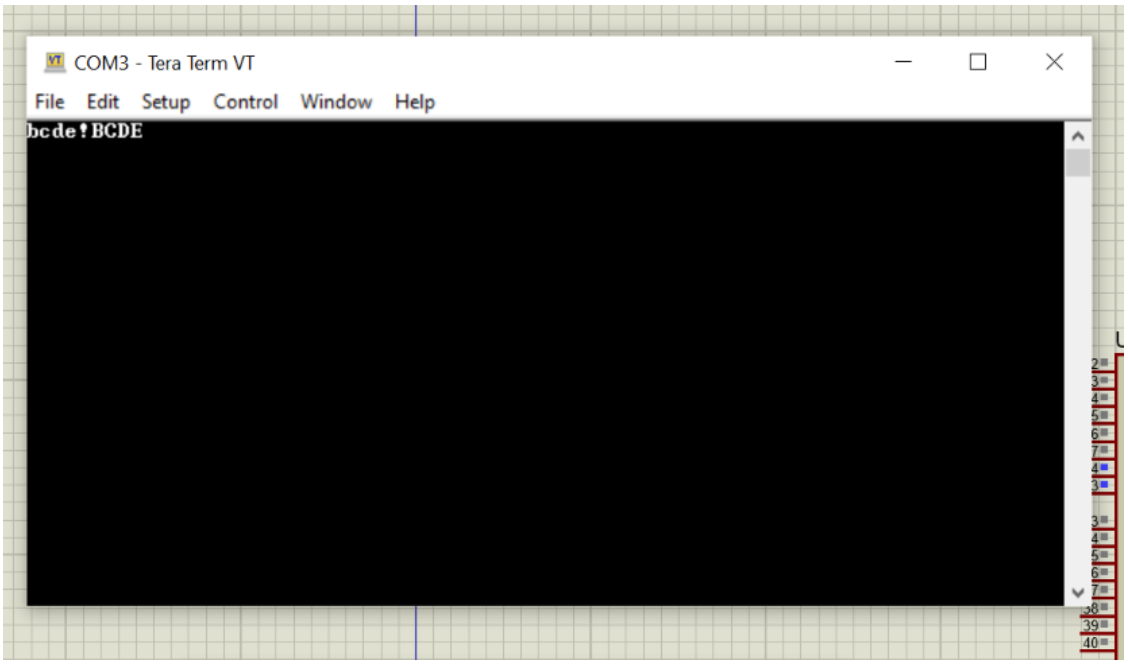
- Circuit Diagram



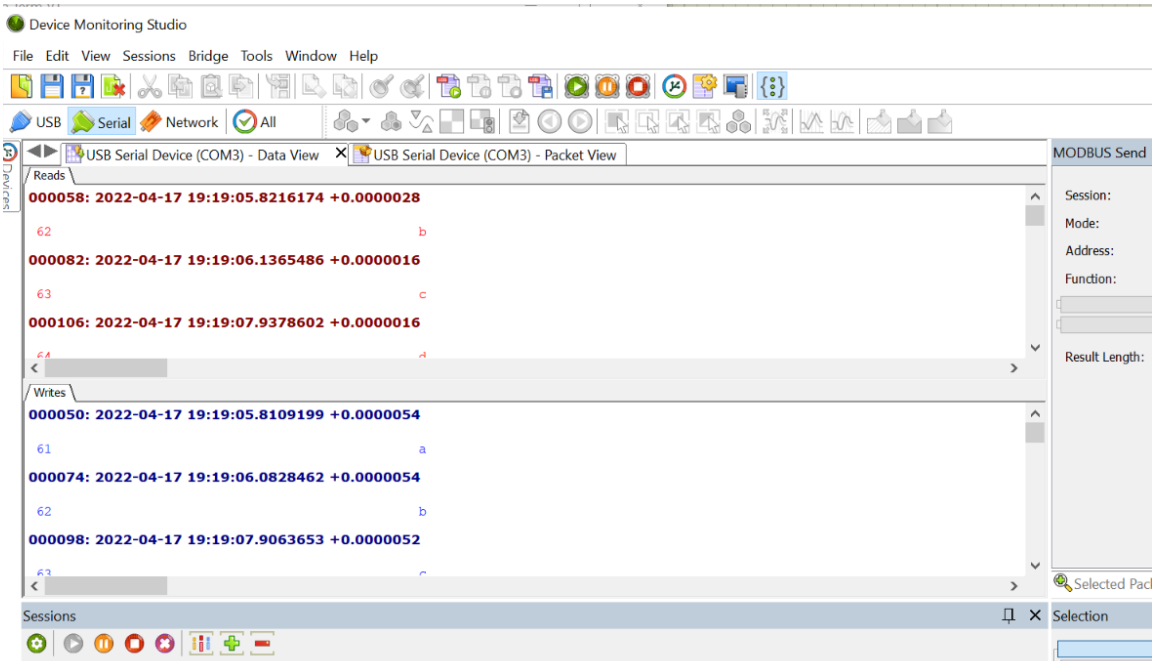
- USB Analyzer

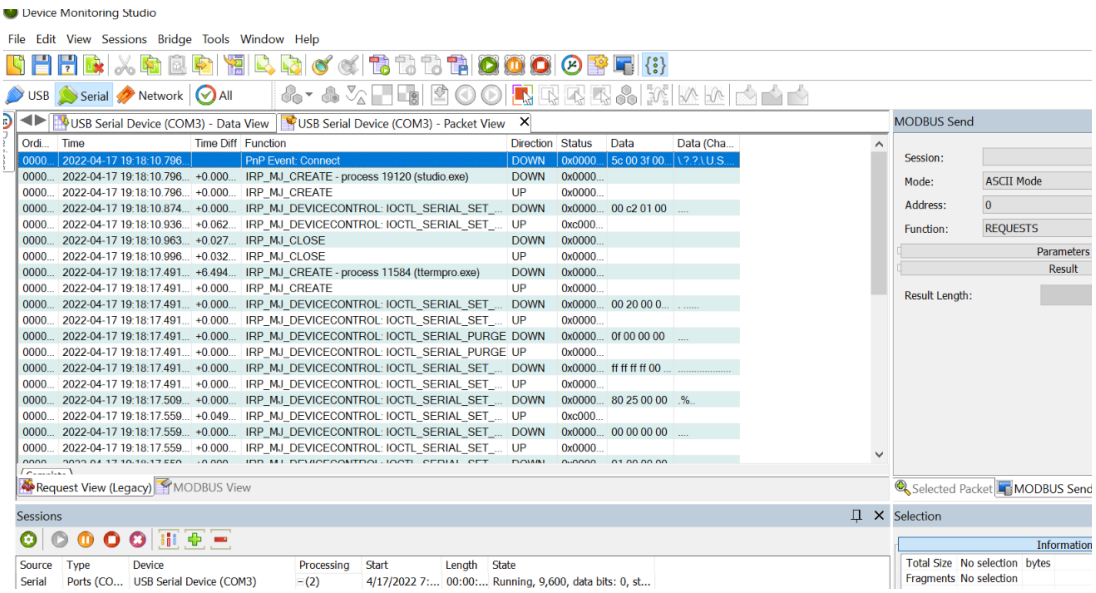


- Tera Term Terminal when the input is “abcd ABCD”



- Device Monitoring Studio





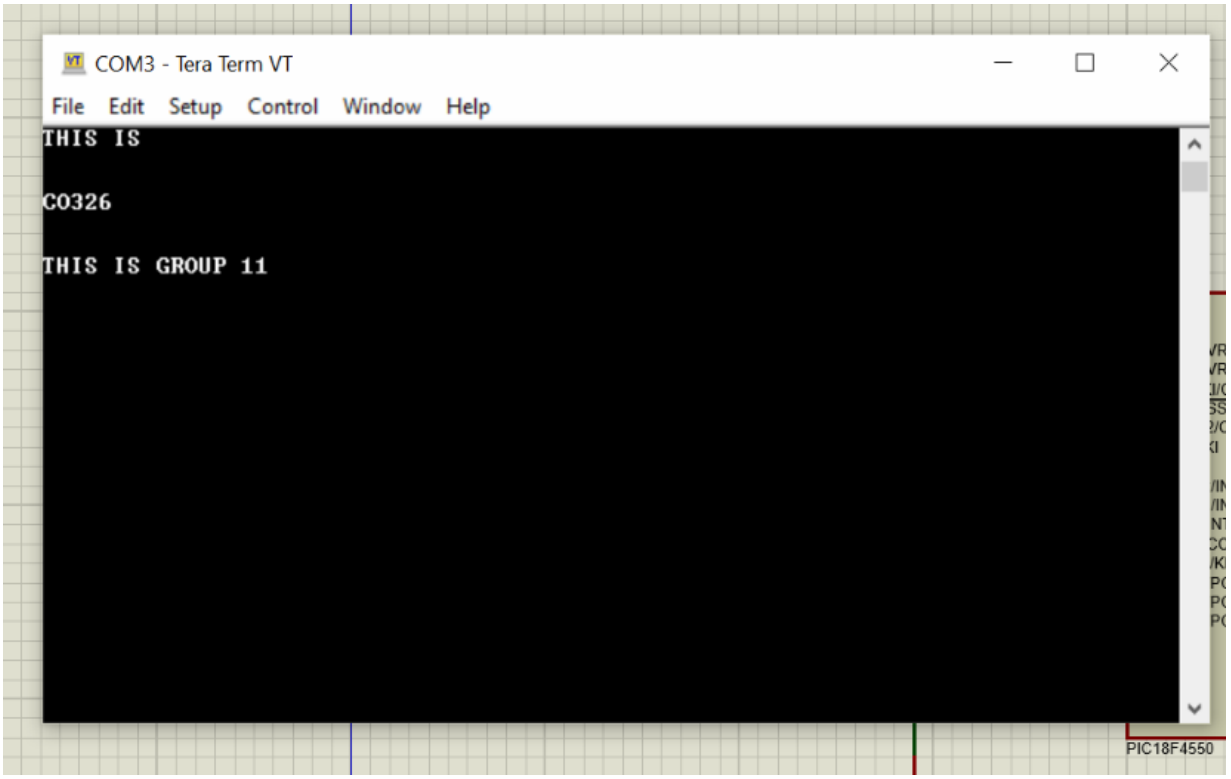
- Code from MPLAB

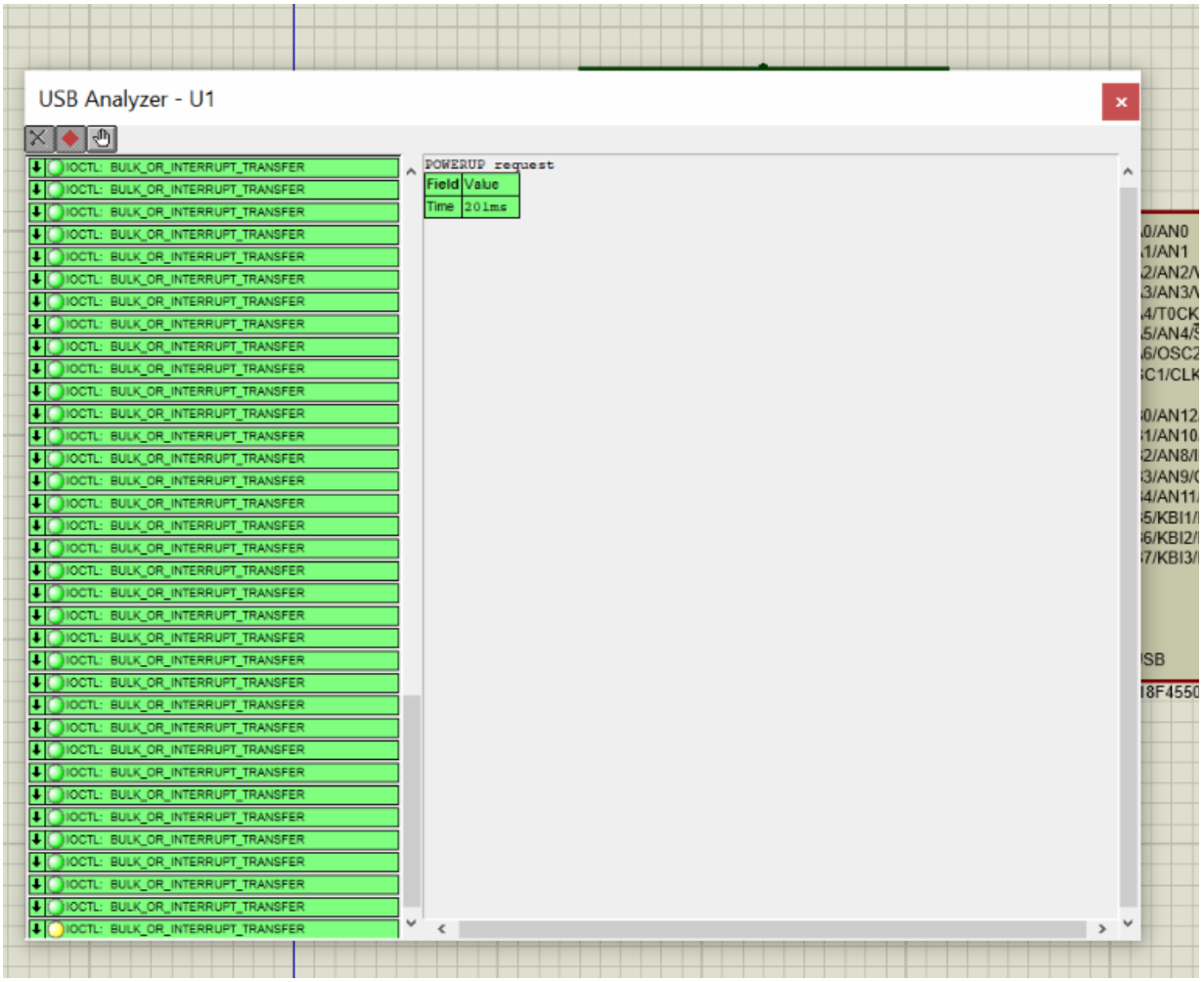
```
static bool buttonPressed;
static char buttonMessage[] = "Button pressed.\r\n";
static uint8_t readBuffer[CDC_DATA_OUT_EP_SIZE];
static uint8_t writeBuffer[CDC_DATA_IN_EP_SIZE];

void APP_DeviceCDCBasicDemoInitialize()
{
    line_coding.bCharFormat = 0;
    line_coding.bDataBits = 8;
    line_coding.bParityType = 0;
    line_coding.dwDTERate = 9600;

    buttonPressed = false;
}

void APP_DeviceCDCBasicDemoTasks()
{
    /* If the USB device isn't configured yet, we can't really do anything
     * else since we don't have a host to talk to.  So jump back to the
     * top of the while loop. */
    if( USBGetDeviceState() < CONFIGURED_STATE )
    {
        return;
    }

       if( USBIsDeviceSuspended()== true )
    {
        return;
    }


    if(BUTTON_IsPressed(BUTTON_DEVICE_CDC_BASIC_DEMO) == true)
    {

        if(buttonPressed == false)
        {
                        if(mUSBUSARTIsTxTrfReady() == true)
            {
                putrsUSBUSART(buttonMessage);
                buttonPressed = true;
            }
        }
    }
    else
    {

        buttonPressed = false;
    }


    if( USBUSARTIsTxTrfReady() == true)
    {
        uint8_t i;
        uint8_t numBytesRead;

        numBytesRead = getsUSBUSART(readBuffer, sizeof(readBuffer));


        for(i=0; i<numBytesRead; i++)
        {
            switch(readBuffer[i])
```

```
        {
                        case 0x0A:
            case 0x0D:
                writeBuffer[i] = readBuffer[i];
                break;


            default:
                writeBuffer[i] = readBuffer[i] + 1;
                break;
        }
    }

    if(numBytesRead > 0)
    {
                putUSBUSART(writeBuffer,numBytesRead);
    }
  }

  CDCTxService();
}
```

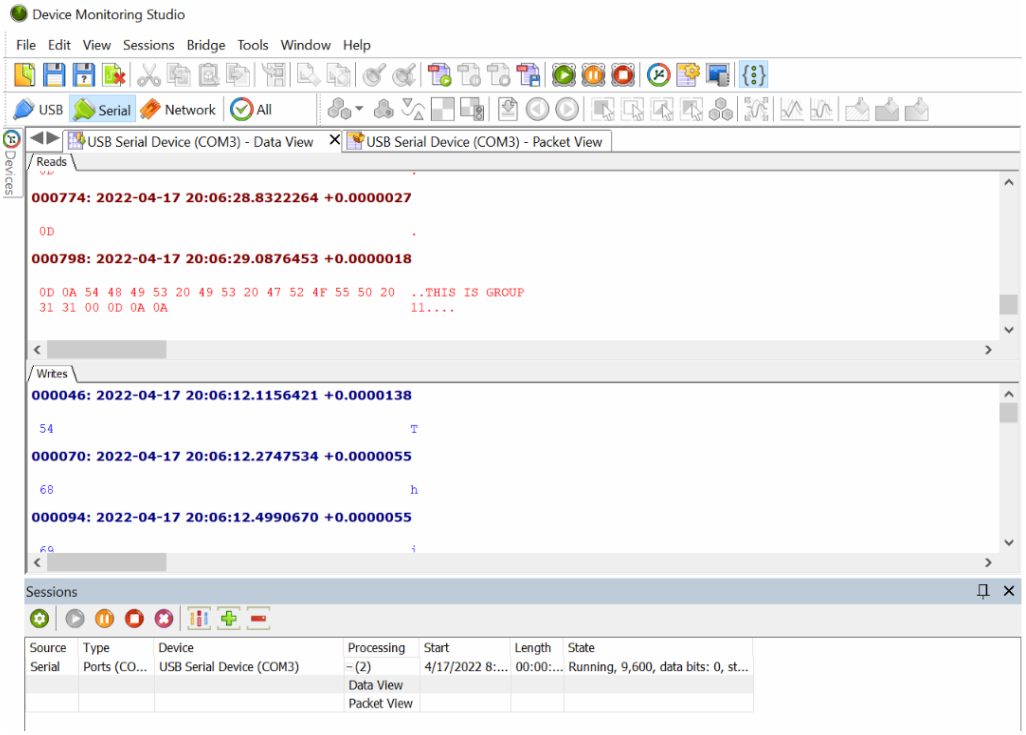
Lab Task

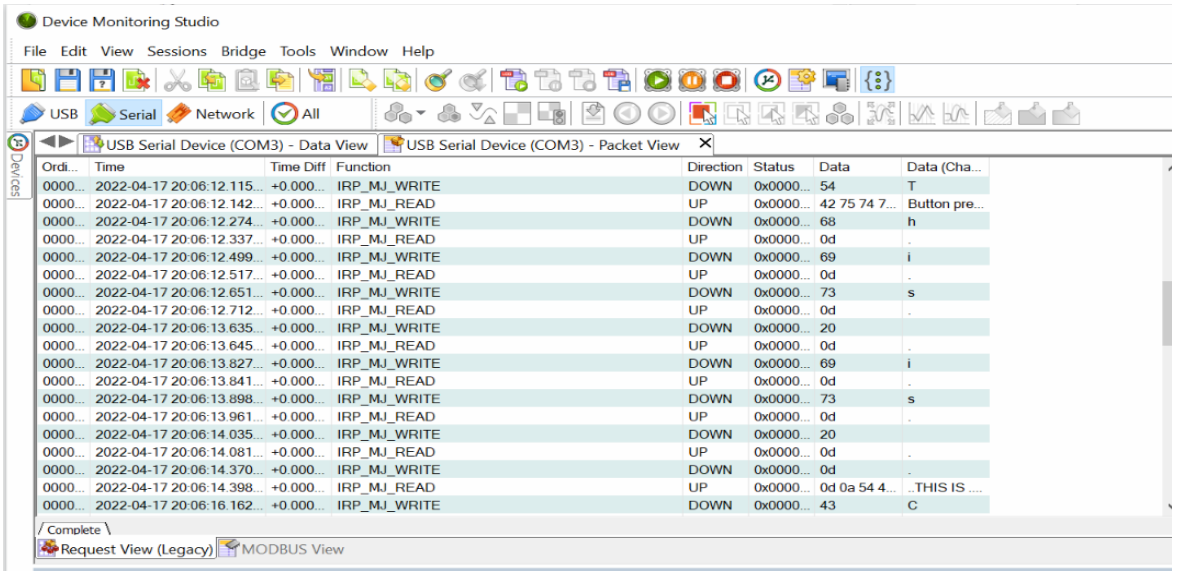
- Tera Term Terminal





- Device Monitoring Studio





- Code from MPLAB

```
#include "system.h"
#include <stdint.h>
#include <string.h>
#include <stddef.h>
#include "usb.h"
#include "app_led_usb_status.h"
#include "app_device_cdc_basic.h"
#include "usb_config.h"

static bool buttonPressed;
static char buttonMessage[] = "Button pressed.\r\n";
static uint8_t readBuffer[CDC_DATA_OUT_EP_SIZE];
static uint8_t writeBuffer[CDC_DATA_IN_EP_SIZE];
static bool ready = false;
static uint8_t bIndex = 2;
```

```
void APP_DeviceCDCBasicDemoInitialize()
{
    line_coding.bCharFormat = 0;
    line_coding.bDataBits = 8;
    line_coding.bParityType = 0;
    line_coding.dwDTERate = 9600;

    buttonPressed = false;
}


void APP_DeviceCDCBasicDemoTasks()
{

    if( USBGetDeviceState() < CONFIGURED_STATE )
    {
        return;
    }

    if( USBIsDeviceSuspended()== true )
    {
        return;
    }


    if(BUTTON_IsPressed(BUTTON_DEVICE_CDC_BASIC_DEMO) == true)
    {

        if(buttonPressed == false)
        {

            if(mUSBUSARTIsTxTrfReady() == true)
            {
                putrsUSBUSART(buttonMessage);
                buttonPressed = true;
            }
        }
    }
    else
    {

        buttonPressed = false;
    }


    if( USBUSARTIsTxTrfReady() == true)
    {
        uint8_t i;
        uint8_t numBytesRead;

        if (blndex < 3) {
            memset(writeBuffer, 0x00, CDC_DATA_IN_EP_SIZE);
            memcpy(writeBuffer, "\r\n", 2*sizeof(uint8_t));
        }

        numBytesRead = getsUSBUSART(readBuffer, sizeof(readBuffer));

        /* For every byte that was read... */
```

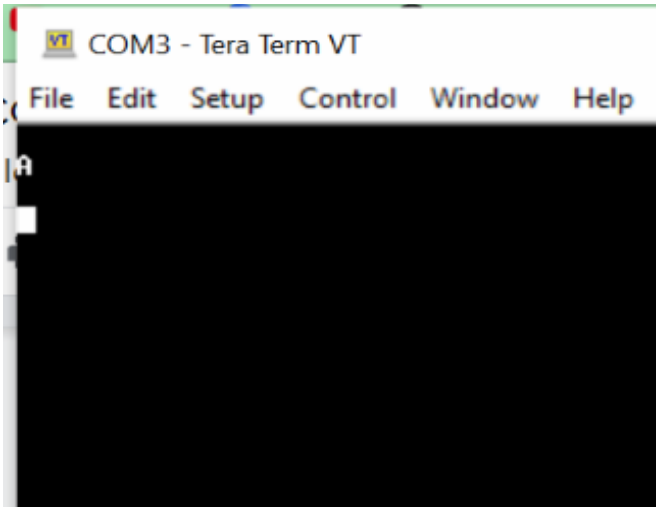
```
        for(i=0; i<numBytesRead; i++)
        {
            //if the current character is the new line character
            if (readBuffer[i] == 0x0D || readBuffer[i] == 0x0A)
            {
                //data is ready to be written
                ready = true;
                break;
            }

            //for all the lowercase characters
            if(readBuffer[i] > 96 && readBuffer[i] < 123)
            {
                //make the character lower
                writeBuffer[bIndex] = readBuffer[i] - 32;
            }
            else
            {
                //for every other character, write the character
                writeBuffer[bIndex] = readBuffer[i];
            }

            //increment the index of the character index
            bIndex++;
            ready = false;

        }


        //sending back the data after converting them to uppercase
        if(ready == true && numBytesRead > 0)
        {
            /* After processing all of the received data, we need to send out
             * the "echo" data now.
             */


            memcpy(&writeBuffer[bIndex+1], "\r\n\n", 3*sizeof(uint8_t));
            putUSBUSART(writeBuffer, bIndex+4);
            bIndex = 2;
        }

        //if the ready signal is not set, the newline character is not received
        //in that case, put that to the writebuffer
        else if (ready == false && numBytesRead > 0)
        {
            putUSBUSART(writeBuffer, numBytesRead);
        }

    }

    CDCTxService();
}
```

Problems and issues you encountered and how you solved them.

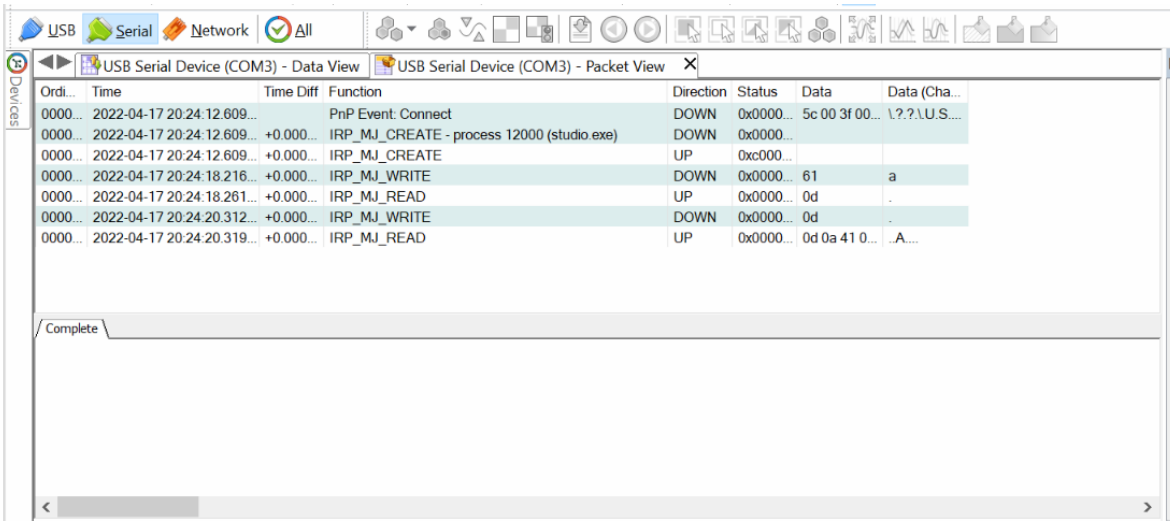
1. When closing the tera term console while proteus simulation was still running, the proteus windows crashed after throwing an error.
2. It was not able to load the Proteus firmware project with PIC18F4550 as the controller. But after Virtual USB Driver was installed for the proteus, it was able to do so.
3. Sometimes the device manager did not recognize the COM3 port. But when the proteus was restarted this issue was able to resolve.

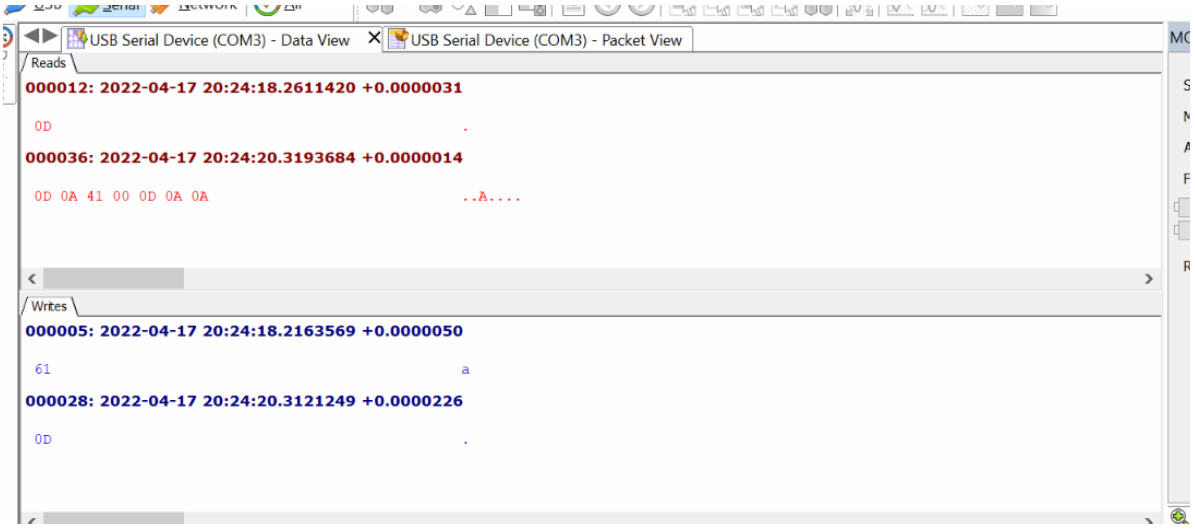
Explanations

- Give a letter you typed and what is observed on the Tera Term.



- Give screenshots of the USB monitor relevant to the letter you type and the letter displayed on the Tera Term.





- One type of packet is IN and the other is OUT. Explain each case discussing why they become IN and OUT packets.

  It was observed in the Device Monitor,it was able to see that there are two sorts of packets which are labelled as IN and OUT. IN packets are data packets received by the COM port. Data is represented under 'read' since the COM port accepts an input. OUT packets are used to represent data sent out through the COM port. The data marked as 'write' is delivered to an external controller through the COM port.