Project Goal:

Get basic understanding of each of the communication protocols i.e UART, I2C and SPI. Establish SPI communication between two microcontroller using SPI and UART to interface the microcontrollers with a terminal to display the data they received.

Resources Used:

- https://elec-club-iitb.github.io/tutorials/avr
- http://www.electonicwings.com/
- https://www.wevolver.com/article/i2c-vs-spi-protocols-differences-pros-cons-use-cases
- https://www.youtube.com/watch?v=Vw7pzzdusWA
- https://www.kanda.com/blog/microcontrollers/all-you-need-to-know-about-avr-isp-updi-jtag-pdi-and-tpi/
- https://www.youtube.com/watch?v=96tTxLVungc
- https://youtu.be/3_omxGIL0kw?si=pN0PVu372nyrEQap
- https://youtu.be/kNO0TIz5C3s?si=Zro9YBRxQkjgAO6c
- https://www.youtube.com/watch?v=wzOjVhEkk3c
- https://www.electronicwings.com/avr-atmega/atmega1632-watchdog-timer
- https://github.com/ExploreEmbedded/oneMicro-ATmega128/blob/master/Code%20Library/Tutorial%20Examples/00-libfiles/spi.c
- https://www.youtube.com/results?search_query=communicate+between+two+microcontrollers+using+spi+complete+code+and+explanation
- https://www.avrfreaks.net/s/topic/a5C3l000000U0Y5EAK/t015909
- https://ww1.microchip.com/downloads/en/devicedoc/doc2467.pdf
- https://www.electronicwings.com/avr-atmega/atmega1632-gpio-ports-and-registers
- https://www.electronicwings.com/avr-atmega/interfacing-lcd-16x2-in-4-bit-mode-with-atmega-16-32-
- https://www.electronicwings.com/avr-atmega/7-segment-display-interfacing-with-atmega16-32
- https://www.youtube.com/results?search_query=lcd+in+atmega+128+8+bit+mode
- https://www.youtube.com/results?search_query=nisso+academy+lcd+programming
- https://www.youtube.com/watch?v=EcPrnucLkUM
- https://www.electronicwings.com/avr-atmega/lcd16x2-interfacing-with-atmega16-32
- https://datasheetspdf.com/product/519148/CA/LCD-1602A/index.html
- https://www.electronicwings.com/sensors-modules/lcd-16x2-display-module

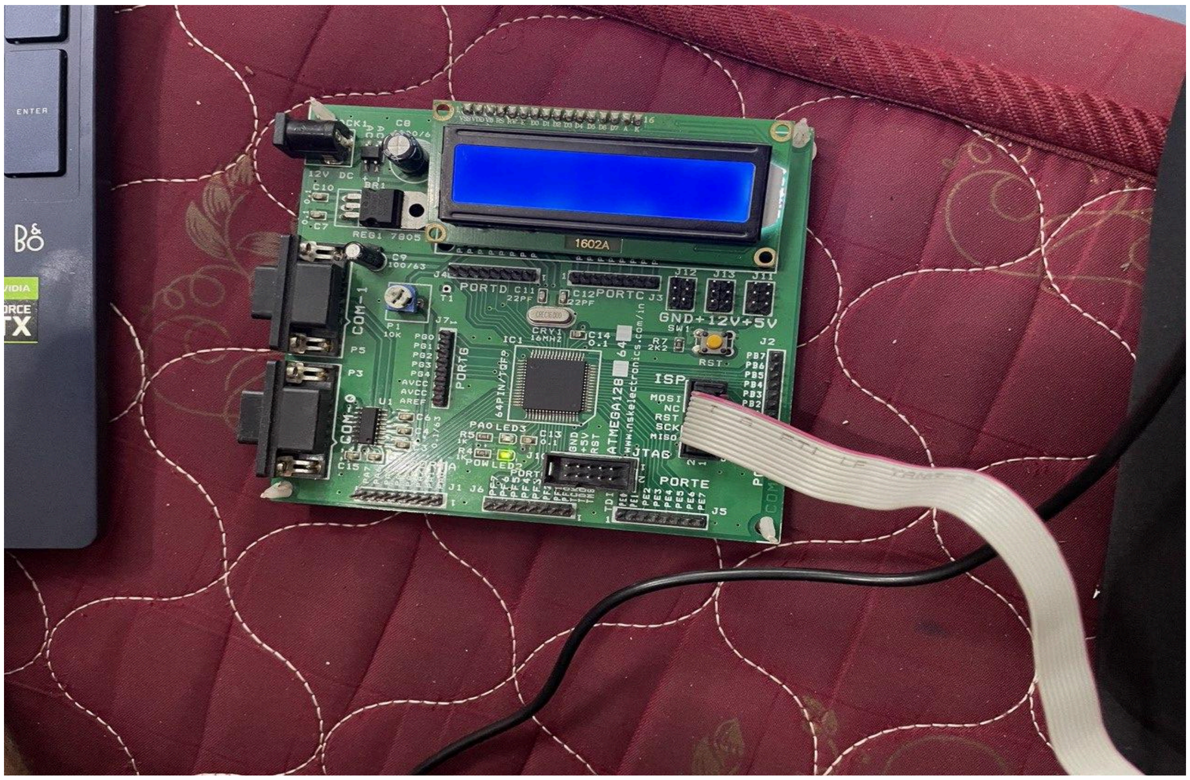
DAY 0:

Read a bit about SPI and I2C protocol.
Tried to understand basic principle and faced difficulty in clearly understanding terms like SCK, MOSI, MISO etc.

DAY 1:(11/3)

Installed Microchip studio and Real term software
Researched on Electronic wings about ATmega 128
Specific videos about microcontroller, registers in microcontrollers etc
(Resources included in work update)
Tried installing avrdude but failed.
Difficulty while installing avrdude, first followed given stes, but avrdude wasn't responding in output of Microchip.

DAY 2:(13/3)



Saw Atmega 128 uc , researched about ISP and JTAG interfaces, their differences and use.
Researched about USBasp, avrdude and their use
Basically after we write a code in microchip studio, it is translated to hexfile, which is uploaded to the USBasp by the avrdude software, Now we do not need laptop if we have alternate power supply options, USBasp serially feeds the input to the ATmega128 uc. Output of the Atmega uc is given back to the USBasp which affects the next input based on interrupts etc.
Tried installing the avrdude and USBasp software again on laptop but failed.

DAY 3:(14/3)

Finally Installed the avrdude and USBasp software on laptop and successfully implemented basic LEDBlinking code using ATmega128.
Problem faced was basically that Avrdude software installed but separate USBasp driver was not installed on laptop, used Zadig to basically install that driver.
After installing first had to copy paste the hexfile from microchip studio onto a separate hexflies folder on C: drive, they dumped the code using cmd prompt on laptop, this was lengthier process as everytime a file was saved, i would have to copy paste the save file to run it using command prompt.
Tried adding external tool on Microchip directly which dumps the code directly using USBasp which worked successfully, so now i have dont have to copy paste my hexfile everytime.

DAY 4:(15/3)

`Worked more on LEDBlinking code and different registers used in Timer and Counter.
Didn’t study Timers and counter in detail, just basic understanding on how a delay can be implemented using a Timer and counter and read data sheet.
(electronic wings)

DAY 5:(17/3)

Started with UART protocol for data transmission. Understanding the basic concept of UART communication.
(research and youtube videos)

DAY 6:(18/3)

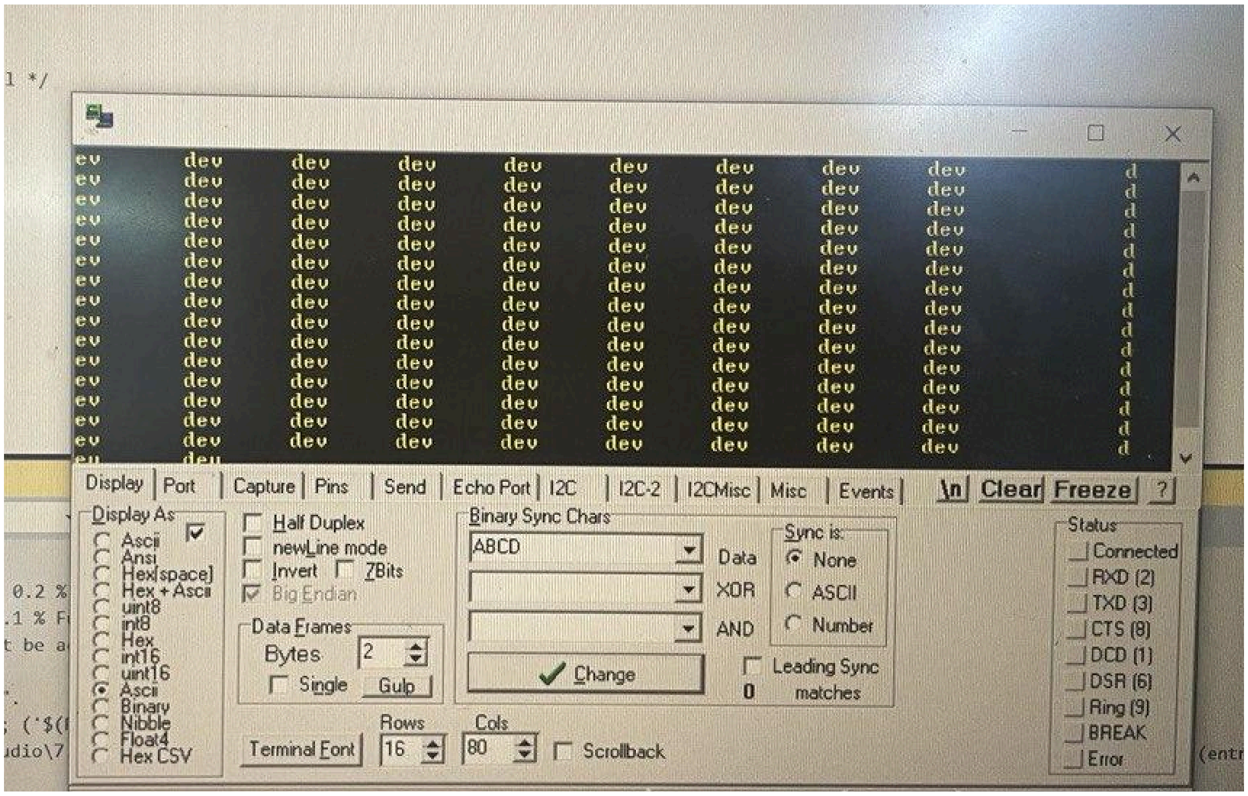
Tried understanding codes in UART and different registers in UART.
I am not yet very comfortable with reading data sheet so i referred to electronic wings website and studied about various registers from there.
(didn’t write codes myself)

DAY 7: (22/3)

Wrote a code for UART and tried implementing. I studied about registers from electronic wings which had some correction so i had to change almost all registers used after referring to the data sheet itself. Faced difficulty in debugging got the first time as i wasn't knowing what was going wrong. Learnt about different debugging methods. Like LED on off. Blinking at different frequencies etc.
The problem was in writing frequency as 9600 and not 9600UL which was not written in example codes and i was told by Tamil that it is the format for atmega128 and he was doing the same error earlier.

```
 * Author : Dev Arora
 */

#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <stdlib.h>
#include <stdio.h>
#define Bud 9600UL
#define ubrr ((F_CPU / (Bud*16)) - 1)

void UART_init(unsigned long BUD)
{
	UCSR0B = UCSR0B | (1<< TXEN0);
	UCSR0C =  (1<< UCSZ00) | (1<<UCSZ01);
	UBRR0L= ubrr;
	UBRR0H= ubrr >> 8;

}
void UART_TxChar(char ch){
	while (! (UCSR0A & (1<<UDRE0)));
	UDR0=ch;
}
void UART_SendString(char *str)
{
	unsigned char j=0;

	while (str[j]!=0)		/* Send string till null */
	{
		UART_TxChar(str[j]);
		j++;
	}
}
```

```
int main(void)
{
   char ch=9;
         UART_init(9600);

   while (1)
   {
                  UART_TxChar(ch);
                  UART_SendString("dev");
                  PORTA ^=0xff;
                  _delay_ms(1000);
   }
}
```



```
ev      dev     dev     dev     dev     dev     dev     dev     dev          d
ev      dev     dev     dev     dev     dev     dev     dev     dev          d
ev      dev     dev     dev     dev     dev     dev     dev     dev          d
ev      dev     dev     dev     dev     dev     dev     dev     dev          d
ev      dev     dev     dev     dev     dev     dev     dev     dev          d
ev      dev     dev     dev     dev     dev     dev     dev     dev          d
ev      dev     dev     dev     dev     dev     dev     dev     dev          d
ev      dev     dev     dev     dev     dev     dev     dev     dev          d
ev      dev     dev     dev     dev     dev     dev     dev     dev          d
ev      dev     dev     dev     dev     dev     dev     dev     dev          d
ev      dev     dev     dev     dev     dev     dev     dev     dev          d
ev      dev     dev     dev     dev     dev     dev     dev     dev          d
ev      dev     dev     dev     dev     dev     dev     dev     dev          d
ev      dev     dev     dev     dev     dev     dev     dev     dev          d
ev      dev     dev     dev     dev     dev     dev     dev     dev          d
```

Display As: Ascii
Big Endian ✓
Binary Sync Chars: ABCD
Data Frames: Bytes 2
Sync is: None
Rows 16 Cols 80

DAY 8: (25/3)

Studied about SPI communication from the data sheet itself, read example codes given in datasheet and studied about various registers and their uses as given in data sheet itself.

DAY 9: (26/3) (Extra session)

Tried writing code for SPI master and slave and sending the data received by the slave through UART on the terminal on laptop. The codes were

SPI Master:

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <stdlib.h>
#include <stdio.h>
#define Bud 9600UL

int main(void){
        DDRA=0xff;
        PORTA=0xff;
        _delay_ms(1000);
        DDRB=DDRB | (1<<2) |(1<<0);
        PORTB=PORTB & 0b11111110;
        DDRB=DDRB | (1<<1);
        SPCR= SPCR | (1<<SPE) | (1<<MSTR) | (1<<SPR0);

        while(1){
        SPDR='d';
  while (!(SPSR & (1<<SPIF)));
        _delay_ms(100);
        SPDR='e';
        while (!(SPSR & (1<<SPIF)));
        _delay_ms(100);
        SPDR='v';
        while (!(SPSR & (1<<SPIF)));
        _delay_ms(100);
        SPDR=' ';
        while (!(SPSR & (1<<SPIF)));
        _delay_ms(100);
        }
        return 0;
}
```

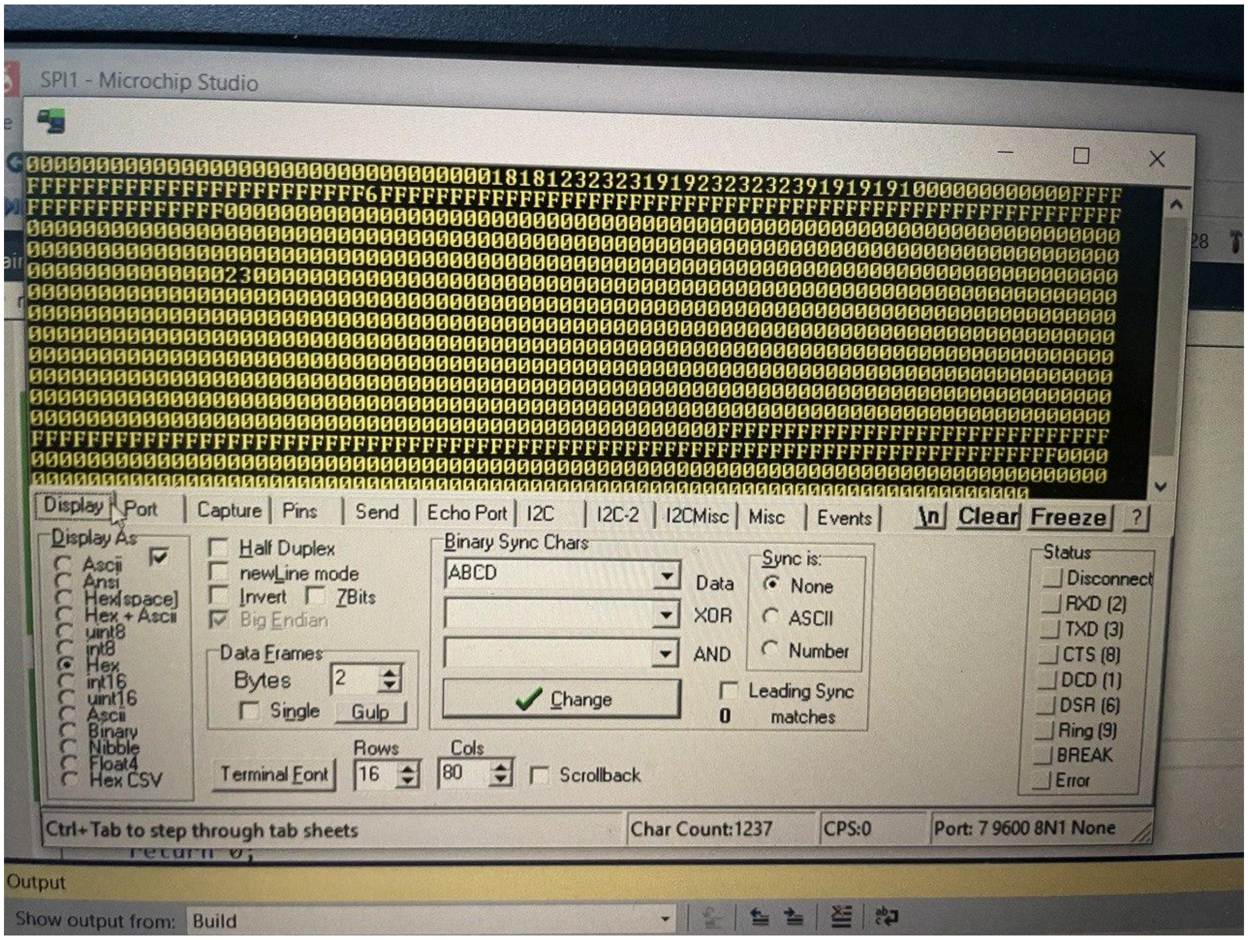
SPI Slave:

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <stdlib.h>
#include <stdio.h>
```

```
#define Bud 9600UL
#define ubrr ((F_CPU / (Bud*16)) - 1)

void UART_init(unsigned long BUD)
{
	UCSR0B = UCSR0B | (1<< TXEN0);
	UCSR0C =  (1<< UCSZ00) | (1<<UCSZ01);
	UBRR0L= ubrr;
	UBRR0H= ubrr >> 8;

}
void UART_TxChar(char ch){
	while (! (UCSR0A & (1<<UDRE0)));
	UDR0=ch;
}

char SPI_SlaveReceive(void)
{
	while(!(SPSR & (1<<SPIF)));
	PORTA^=0xff;
	return SPDR;
}

int main(void)
{

	DDRB =DDRB | (1<<3);
	SPCR=SPCR | (1 <<SPE);
	UART_init(9600);
	PORTA=0xff;
	_delay_ms(1000);
	while(1){
	char c=SPI_SlaveReceive();
	_delay_ms(50);
	UART_TxChar(c);
	_delay_ms(50);
}
return 0;
}
```

DAY 10 : (27/3)

I started with the debugging process,
 Earlier i was using wrongs pins for SPI communication of port e which were also mention in data sheet when i search debut they were not relevant for my work ig. The pins required were of pot b for basic spi communication used in Pb(0-3). After connection the right pins i forgot to connect them to common ground, i was powering them through laptop which was not working, i has to used external source of power(10V) connecting 5V was even not sufficient.
Even after this there ewas error in my slave code for recieving the data(error in boolean algebra part) which i debugged using LED on off after every step to know which step had error.
Finally my code was implemented and communication was established using SPI and was showing on the terminal.
I was sending dev without using loop which was also giving error as it was printing random error like “ddeev””ddevv””deevv” etc. this i debugged after adding delay after every character sent by the master to the slave as the charecter was in the SPDR register for longer time which was read by the UART register hence there was random error in printing data received on terminal.


SPI1 - Microchip Studio

```
000000000000000000000000000000000181812323231919232323239191919100000000000FFFF
FFFFFFFFFFFFFFFFFFFFFFF6FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFF000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000230000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000FFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
```

Display As: Hex
Data Frames: Bytes 2
Binary Sync Chars: ABCD
Sync is: None
Rows 16 Cols 80
Ctrl+Tab to step through tab sheets | Char Count:1237 | CPS:0 | Port: 7 9600 8N1 None

Earlier code which worked. Without ‘ddevv type error.

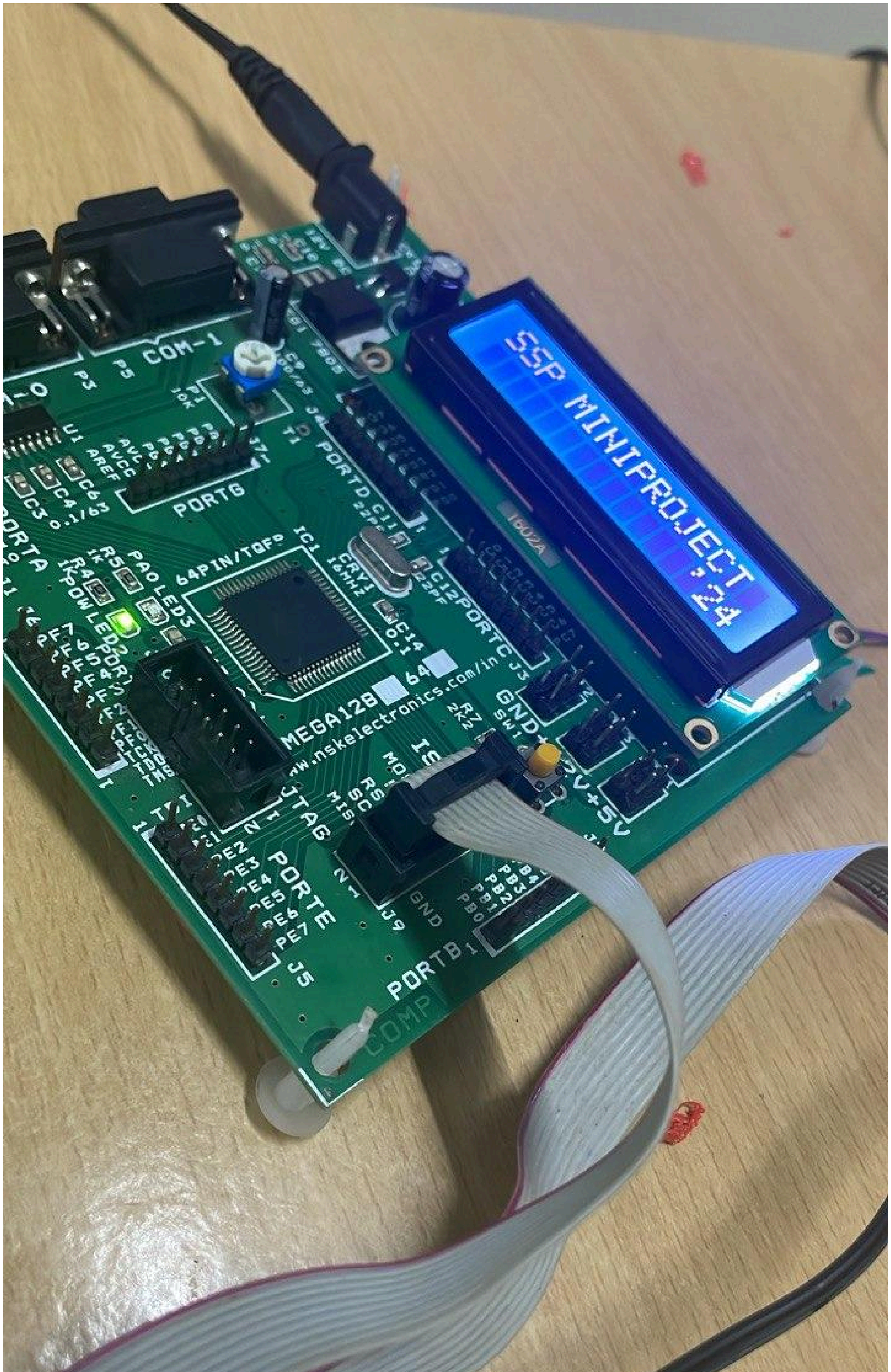
DAY 11: (28/3)

Started studying about LCD, saw relevant videos on youtube and from electronic wings, downloaded and read the datasheet on LCD1602A. The commands for various function were not in the datasheet but i found them on electronic wings.
I wrote the code for LCD display after referring to the code on electronic wings and after referring to youtube video but it didnot work.
Code written:

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#define RS PG2
#define RW PG1
#define EN PG0
#define LCD_Data_Dir DDRC
#define LCD_Command_Dir DDRG
#define LCD_Data_Port PORTC
#define LCD_Command_Port PORTG

void LCD_Command(unsigned char c)
{
        PORTC= c;
        PORTG &= ~(1<<RS);
        PORTG &= ~(1<<RW);
        PORTG |= (1<<EN);
        _delay_us(10);
        PORTG &= ~(1<<EN);
        _delay_ms(3);
}
void LCD_Init (void){
        DDRC=0xff;
        DDRG=0xff;
        DDRA=0xff;
        _delay_ms(30);
        LCD_Command (0x38);
        _delay_ms(2);
        LCD_Command (0x0C);
        _delay_ms(2);
        LCD_Command (0x06);
        _delay_ms(2);
        LCD_Command (0x01);
        _delay_ms(2);
        LCD_Command (0x80);
        _delay_ms(2);
        PORTA=0xff;
        _delay_ms(1000);
        PORTA=0x00;
        }

void LCD_Char (unsigned char char_data)   /* LCD data write function */
{
        LCD_Data_Port= char_data;
        LCD_Command_Port |= (1<<RS);      /* RS=1 Data reg. */
        LCD_Command_Port &= ~(1<<RW);     /* RW=0 write operation */
        LCD_Command_Port |= (1<<EN);      /* Enable Pulse */
        _delay_us(1);
        LCD_Command_Port &= ~(1<<EN);
        _delay_ms(1);
}
void LCD_String (char *str)               /* Send string to LCD function */
{
        int i;
        for(i=0;str[i]!=0;i++)            /* Send each char of string till the NULL */
        {
                LCD_Char (str[i]);
        }
}
void LCD_Clear()
{
        LCD_Command (0x01);               /* clear display */
        LCD_Command (0x80);               /* cursor at home position */
}


int main(void){
        LCD_Init();
        LCD_Clear();
        LCD_String("GOODNIGHT");
        LCD_Command(0xC0);
        LCD_String("");

        return 0;

}
```

DAY 12: (29/3)

Implemented the LCD code on hardware, it turned out that the code was correct it was just that the i didn’t know that the LCD wasn’t working. There was some issue in the contrast setting in the LCD i was using that is why i wasn’t getting any output. ONly 1 out of the 3 LCD in lab work which we figured out later.


SSP MINIPROJECT
'24
COM-1
PORTG
PORTD
PORTC
PORTA
PORTE
PORTB
64PIN/TQFP
MEGA128 64
www.nskelectronics.com/in
JTAG
ISP
GND
+5V
1602A

DAY 13: (30/3)

Wrote a basic version of traffic signal implementation in which i only use one microcontroller and display output which is based on the current input which i take from the two input pins. I was taking input externally by connecting them by hand to 5V or GND in various cases and checking input after every 4s. But i couldn't implement this code as the 1 working LCD was being used by my project partner.

DAY 14: (31/3)

I finally implemented the final deliverable in which setup an array of fixed number of elements, each element of the array was read after a fixed interval and it was shown on the lcd, now each element of this array was sent one by one to the other microcontroller and was displayed on PC using UART protocol. I faced hardware issue as taking input was random which i couldn't figure out why but whatever input i got i was successfully able to transmit successfully, since only one LCD was working i couldn't perform printing task on the other microcontroller so i just sent it to terminal.

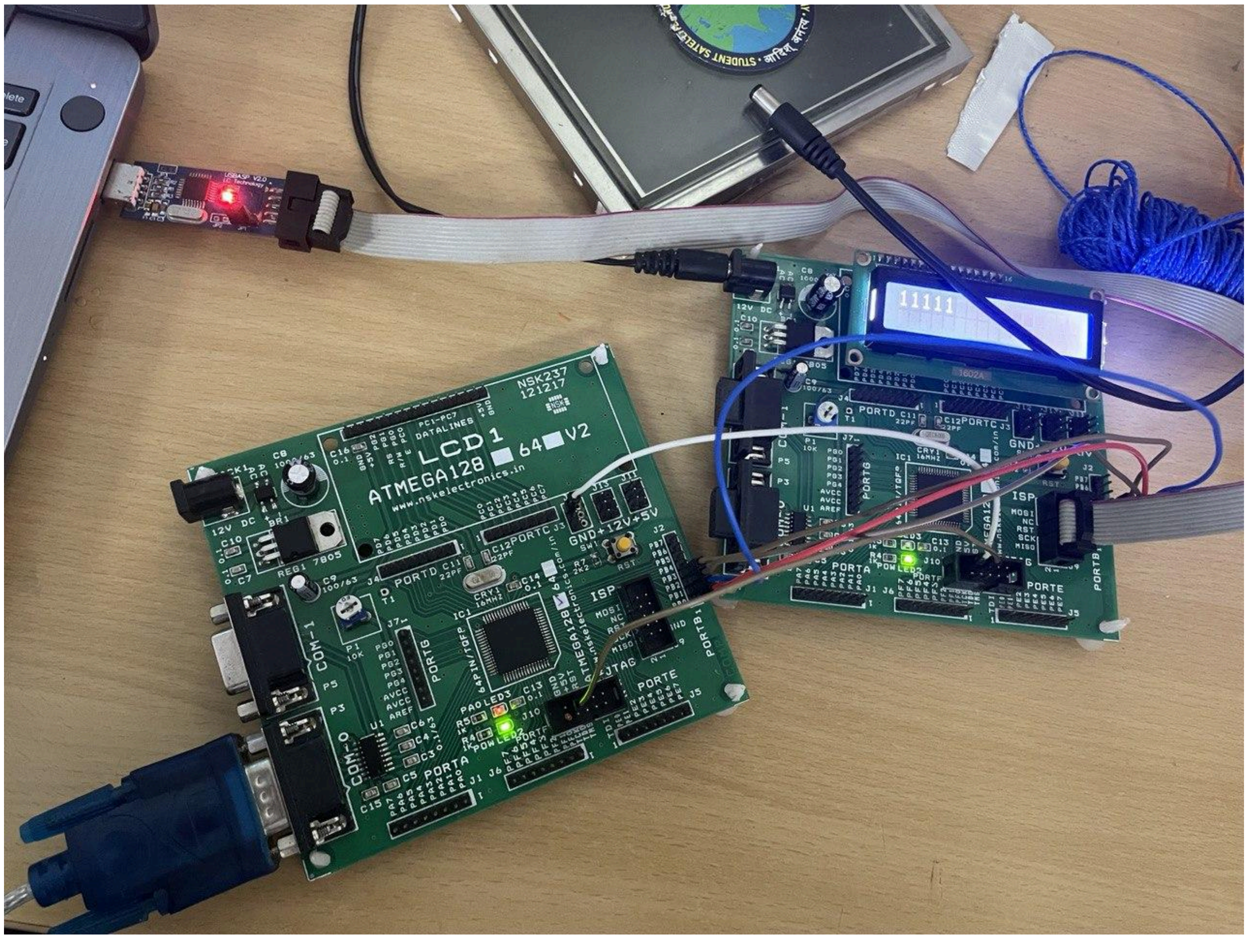
```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <stdlib.h>
#include <stdio.h>
#define Bud 9600UL
#define RS PG2
#define RW PG1
#define EN PG0
#define S1 PB7
#define S2 PB6
#define LCD_Data_Dir DDRC
#define LCD_Command_Dir DDRG
#define LCD_Data_Port PORTC
#define LCD_Command_Port PORTG

void LCD_Command(unsigned char c)
{
	PORTC= c;
	PORTG &= ~(1<<RS);		/* RS=0 command reg. */
	PORTG &= ~(1<<RW);		/* RW=0 Write operation */
	PORTG |= (1<<EN);	/* Enable pulse */
	_delay_us(10);
	PORTG &= ~(1<<EN);
	_delay_ms(3);
```

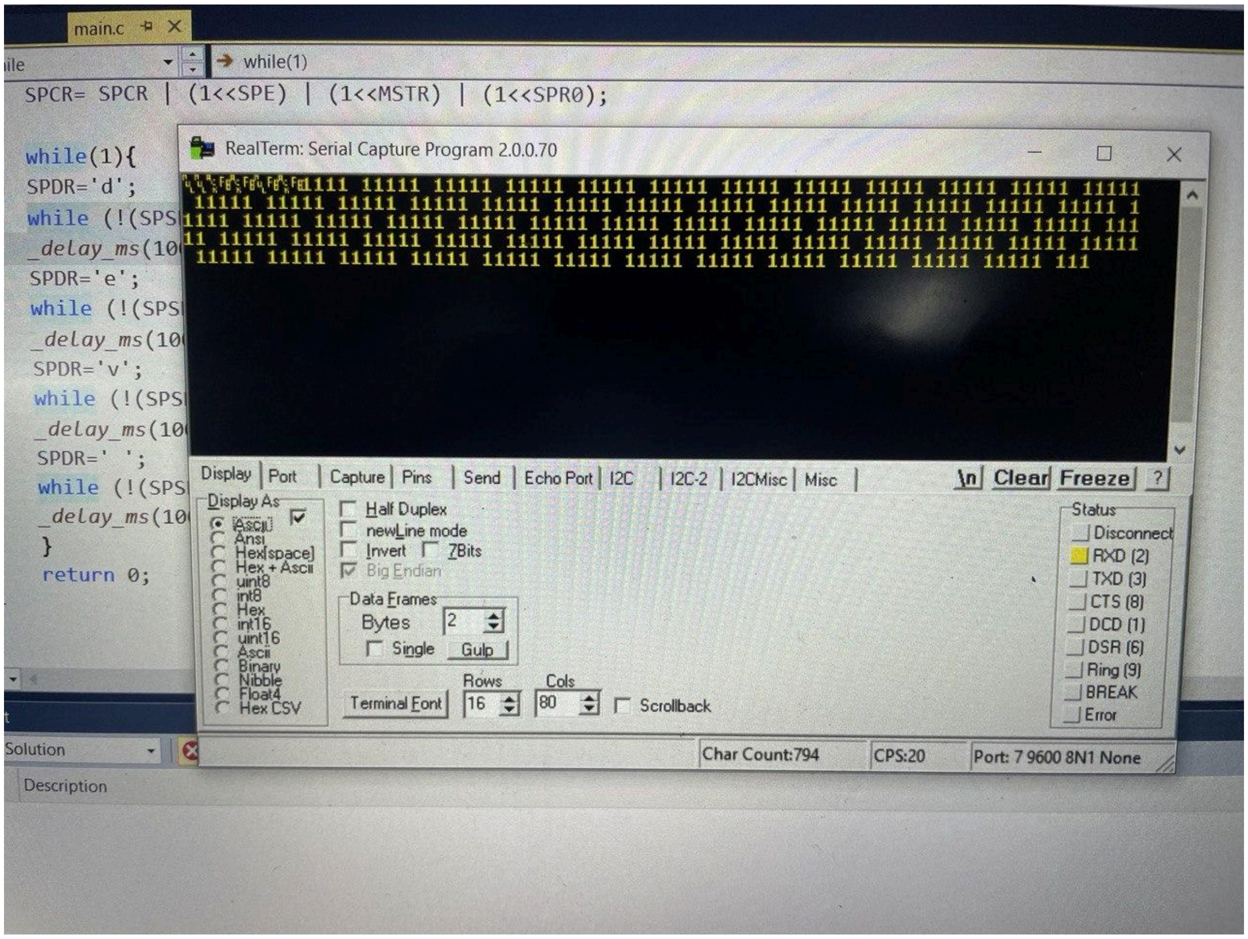
```
}
void LCD_Init (void){
        DDRC=0xff;
        DDRG=0xff;
        _delay_ms(30);
        LCD_Command (0x38);
        _delay_ms(2);/* Initialization of 16X2 LCD in 8bit */
        LCD_Command (0x0C);
        _delay_ms(2); /* Display ON Cursor OFF */
        LCD_Command (0x06);
        _delay_ms(2); /* Auto Increment cursor */
        LCD_Command (0x01);
        _delay_ms(2); /* clear display */
        LCD_Command (0x80);
        _delay_ms(2);
}

void LCD_Char (unsigned char char_data)   /* LCD data write function */
{
        LCD_Data_Port= char_data;
        LCD_Command_Port |= (1<<RS);     /* RS=1 Data reg. */
        LCD_Command_Port &= ~(1<<RW);/* RW=0 write operation */
        LCD_Command_Port |= (1<<EN);     /* Enable Pulse */
    _delay_us(1);
        LCD_Command_Port &= ~(1<<EN);
        _delay_ms(1);
}
void LCD_String (char *str)             /* Send string to LCD function */
{
        int i;
        for(i=0;str[i]!=0;i++)          /* Send each char of string till the NULL */
        {
                LCD_Char (str[i]);
        }
}
void LCD_Clear()
{
        LCD_Command (0x01);              /* clear display */
        LCD_Command (0x80);              /* cursor at home position */
}


int main(void){
//      DDRB = 0x00;
```

```
//		DDRA=0xff;
//		DDRE=0xff;
//		DDRD=0xff;
//		DDRF=0xff;
//		PORTA='1';;
//		PORTE='2';
//		PORTD='3';
//		PORTF='4';
//
//		LCD_Init();
//		while(1){
//		if (PINB==0x00){
//		LCD_Clear();
//		LCD_String("D");
//		_delay_ms(4000);
//		}
//		else if(PINB==0b01000000){
//			LCD_Clear();
//			LCD_String("C");
//			_delay_ms(4000);
//		}
//
//		else if(PINB==0b10000000){
//			LCD_Clear();
//			LCD_String("A");
//			_delay_ms(4000);
//		}
//
//		else if(PINB==0b11000000){
//			LCD_Clear();
//			LCD_String("B");
//			_delay_ms(4000);
//		}
//		}
//
//		return 0;
// }
DDRA=0xff;
PORTA=0xff;
_delay_ms(1000);
DDRB=DDRB | (1<<2) |(1<<0);
PORTB=PORTB & 0b11111110;
DDRB=DDRB | (1<<1);
SPCR= SPCR | (1<<SPE) | (1<<MSTR) | (1<<SPR0);
```

```
LCD_Init();
LCD_Clear();
char Queue[5];
for(int i=0; i<5; i++){
        Queue[i]='0';
}
DDRF=0X00;
for(int i=0; i<5; i++){
        PORTA=0xff;
        _delay_ms(5000);
        PORTA=0x00;
        if(PINF==0b00000000){
                Queue[i]='1';
                LCD_String("1");
                _delay_ms(10);

        }
   else if(PINF==0B10000000){
                Queue[i]='2';
                LCD_String("2");
                _delay_ms(10);
}
   else if(PINF==0b11000000){
           Queue[i]='3';
                LCD_String("3");
        }
    else {
                Queue[i]='4';
                LCD_String("4");
                _delay_ms(10);
        }
//              LCD_Char(Queue[i]);
//              _delay_ms(1000);
}
while(1){
        for(int i=0; i<5; i++){
                SPDR=Queue[i];
                while (!(SPSR & (1<<SPIF)));
                _delay_ms(100);
        }
        SPDR=' ';
        while (!(SPSR & (1<<SPIF)));
        _delay_ms(100);
}
```

```
return 0;
}
```


USBASP V2.0
NSK237
121217
LCD1
ATMEGA128 64 V2
www.nskelectronics.in
PORTD
PORTC
PORTG
PORTA
PORTE
PORTB
ISP
JTAG
COM-1
COM-0
GND +12V +5V
11111



```c
SPCR= SPCR | (1<<SPE) | (1<<MSTR) | (1<<SPR0);

while(1){
SPDR='d';
while (!(SPS
_delay_ms(10
SPDR='e';
while (!(SPS
_delay_ms(10
SPDR='v';
while (!(SPS
_delay_ms(10
SPDR=' ';
while (!(SPS
_delay_ms(10
}
return 0;
```

RealTerm: Serial Capture Program 2.0.0.70

```
[illegible]1111 11111 11111 11111 11111 11111 11111 11111 11111 11111 11111 11111
 11111 11111 11111 11111 11111 11111 11111 11111 11111 11111 11111 11111 11111 1
1111 11111 11111 11111 11111 11111 11111 11111 11111 11111 11111 11111 11111 111
11 11111 11111 11111 11111 11111 11111 11111 11111 11111 11111 11111 11111 11111
 11111 11111 11111 11111 11111 11111 11111 11111 11111 11111 11111 11111 111
```

Display As: [✓ Ascii] ✓

Half Duplex ☐ newLine mode ☐ Invert ☐ 7Bits ☐ Big Endian ✓

Data Frames: Bytes 2, Single ☐

Rows 16, Cols 80, Scrollback ☐

Status: RXD (2) 🟨

Char Count:794 CPS:20 Port: 7 9600 8N1 None