



# IITBSSP MINI PROJECT

## COMMUNICATION PROTOCOL (SPI)

[Dev Arora]





# MINI PROJECT

## What is this project?

- Software Installation
- Hardware and DataSheet
- UART
- SPI
- LCD



## FINAL DELIVERABLE?



# TIMELINE

## WEEK 1:

- Downloaded Microchip Studio and Real Term
- ATmega128, USBasp
- LED Blink
- UART Understanding

## WEEK 2:

- Implemented UART
- SPI Understanding



# TIMELINE

## WEEK 3:

- Implemented SPI
- LCD Datasheet
- Implemented LCD
- Implemented Simpler Version of Final Deliverable.

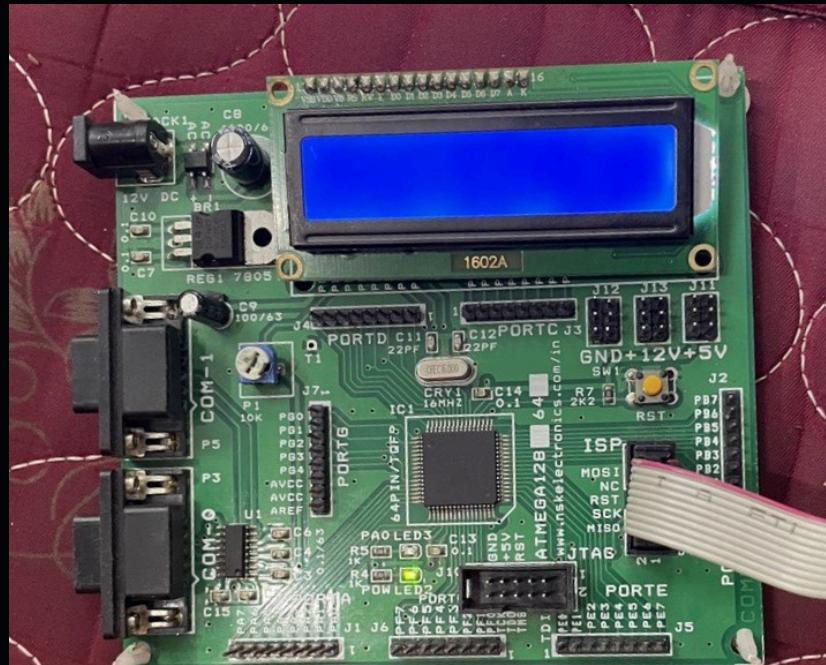


# HARDWARE



DB9 Cable

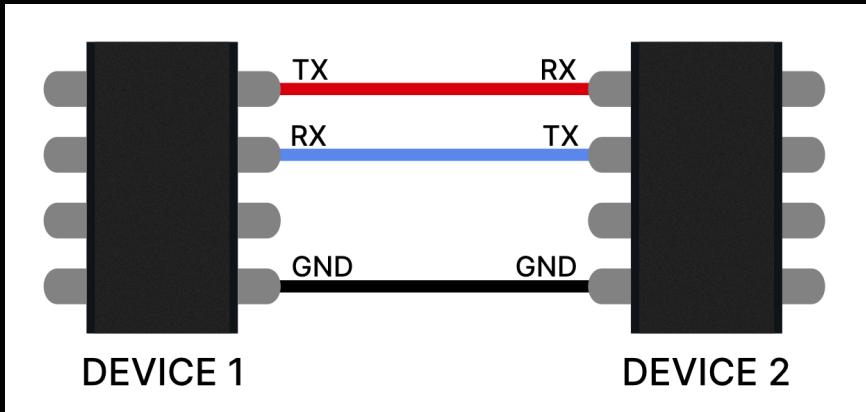
ATmega128  
Development Board



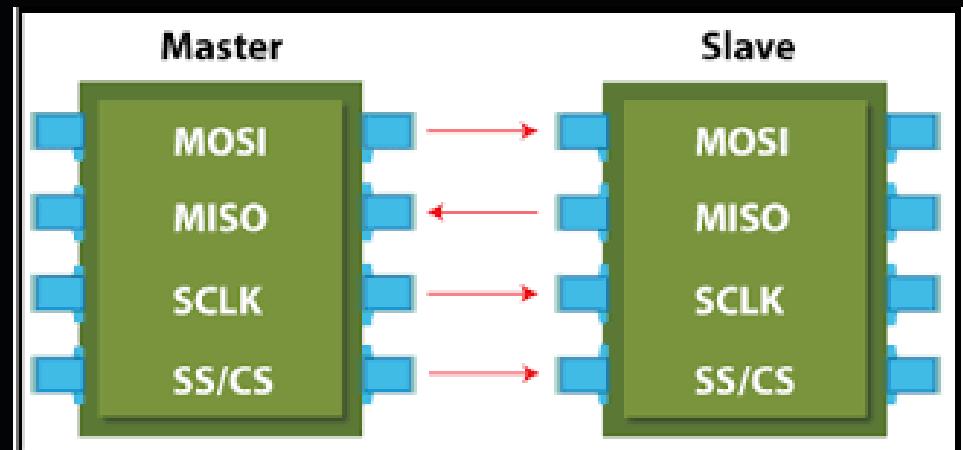
USBasp ISP  
Programmer



# MAJOR LEARNINGS?



UART



SPI



# MAJOR LEARNINGS?

## UART:

- 2 wire interface
- 2 devices
- Full Duplex
- Easiest to understand and code.



# MAJOR LEARNINGS?

## SPI:

- 4 wire interface
- Multiple slaves, 1 master
- Full duplex
- Faster than USART and I2C
- Commonly used for high speed communication



# MAJOR LEARNINGS?

## LCD:

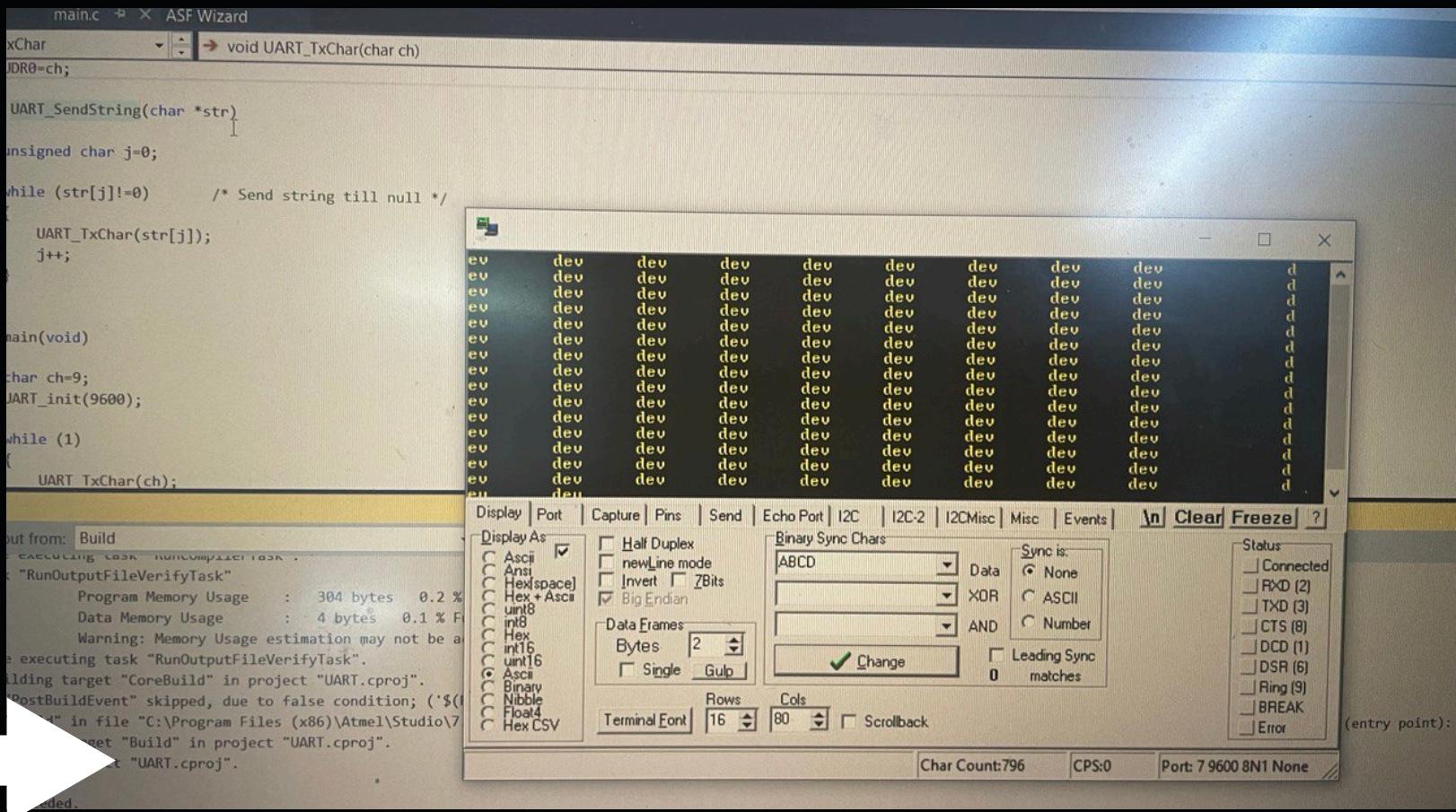
Commonly Used LCD16x2 Commands

Code (HEX)	Command to LCD	Execution Time
0x01	Clear the display screen	1.64ms
0x06	Shift the cursor right (e.g. data gets written in an incrementing order, left to right)	40 us
0x0C	Display on, cursor off	40 us
0x0E	Display on, cursor blinking	40 us
0x80	Force the cursor to the beginning of the 1st line	40 us
0xC0	Force the cursor to the beginning of the 2nd line	40 us
0x10	Shift cursor position to the left	40 us
0x14	Shift cursor position to the right	40 us
0x18	Shift entire display to the left	40 us
0x1C	Shift entire display to the right	40 us
0x38	2 lines, 5x8 matrix, 8-bit mode	40 us
0x28	2 lines, 5x8 matrix, 4-bit mode	40 us
0x30	1 line, 8-bit mode	40us
0x20	1 line, 4-bit mode	40us



# OUTPUTS

## UART:



The screenshot shows the Atmel Studio IDE interface. On the left, the code editor displays the main.c file with the following content:

```
main.c  X  ASF Wizard

xChar
void UART_TxChar(char ch)
{
    JDR0=ch;

    UART_SendString(char *str)
    {
        unsigned char j=0;

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

    main(void)
    {
        char ch=9;
        JUART_init(9600);

        while (1)
        {
            UART_TxChar(ch);
        }
    }
}
```

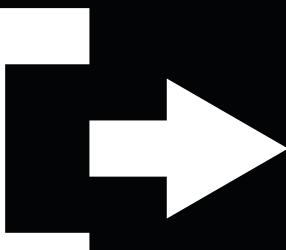
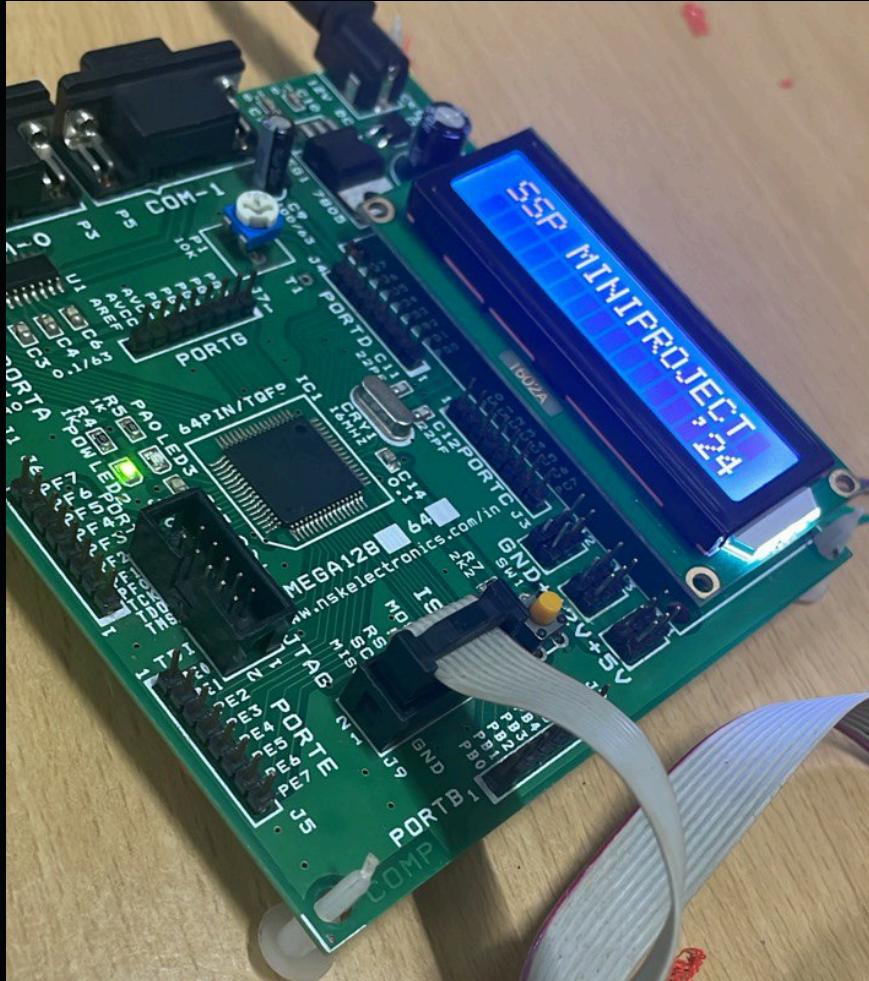
On the right, the terminal window shows the output of the program. The terminal configuration is set to Port: 7 9600 8N1 None. The output consists of a continuous stream of the character 'dev' repeated 24 times across 8 columns.

At the bottom of the terminal window, the status bar shows "Char Count:796 CPS:0 Port: 7 9600 8N1 None".



# OUTPUTS

## LCD:



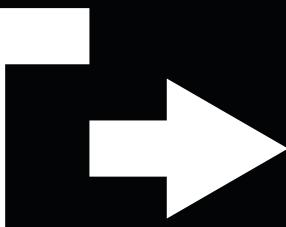


# OUTPUTS

## QUEUE:

```
/*
#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 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 */
}
```



Master



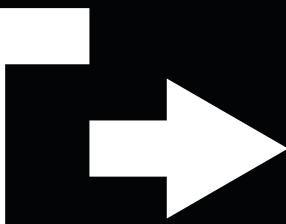
# OUTPUTS

## QUEUE:

```
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++)
    {
        LCD_Char (str[i]);
    }
}
void LCD_Clear()
{
    LCD_Command (0x01);      /* clear display */
    LCD_Command (0x80);      /* cursor at home position */
}

int main(void){
DDRA=0xFF;
```



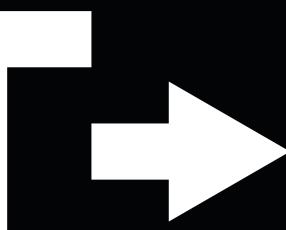
Master



# OUTPUTS

## QUEUE:

```
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);  
    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);  
        }  
    }  
}
```



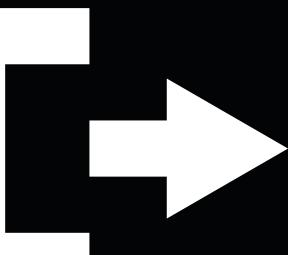
Master



# OUTPUTS

## QUEUE:

```
    else if(PINF==0B110000000){  
        Queue[i]='3';  
        LCD_String("3");  
    }  
    else {  
        Queue[i]='4';  
        LCD_String("4");  
        _delay_ms(10);  
    }  
}  
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;  
}
```



Master



# OUTPUTS

## QUEUE:

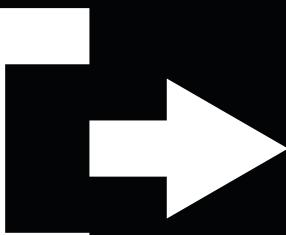
```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <stdlib.h>
#include <stdio.h>
#define Bud 9600UL
#define ubrr ((F_CPU / (Bud*16)) - 1)

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);
```



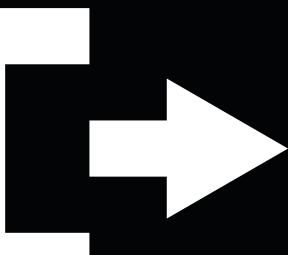
Slave



# OUTPUTS

## QUEUE:

```
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;
}
```

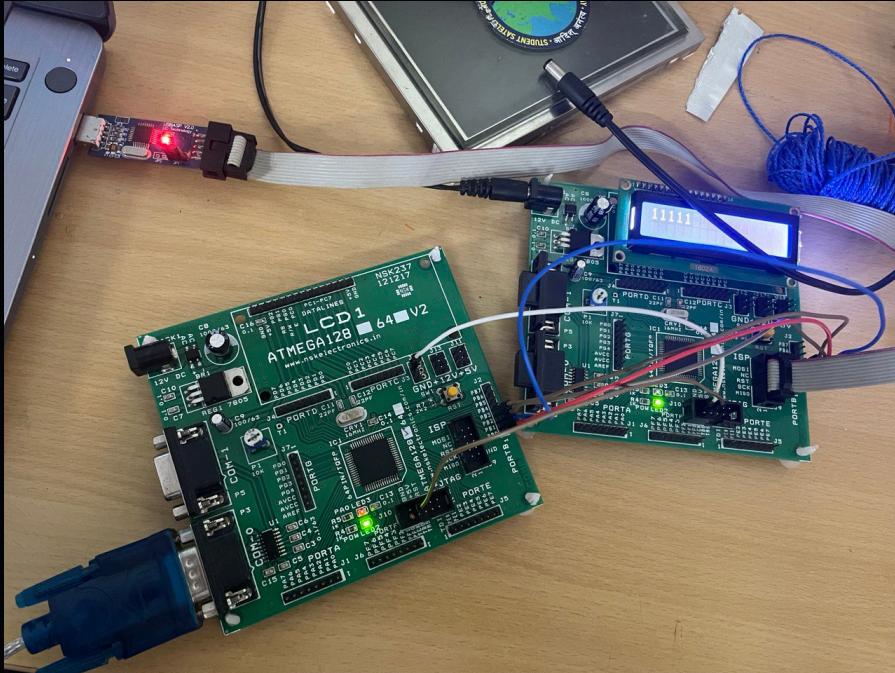


Slave



# OUTPUTS

## QUEUE:



```
main.c : X
main.c
File Edit View Help
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;
}

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

The screenshot shows a development environment with a code editor and a terminal window. The code editor contains C code for a while loop that repeatedly sends characters 'd', 'e', 'v', and a space over a serial port. The terminal window, titled 'RealTerm: Serial Capture Program 2.0.0.70', displays the received binary data in hex format. The status bar at the bottom indicates the port settings: 7 9600 8N1 None.



# CHALLENGES FACED

Some Challenges I faced were:

- Software setup, USBasp driver avrdude
- SPI communication, PINS and wrong output
- Hardware issue in LCD and Taking input





# ACKNOWLEDGEMENT, MOTIVATION AND FEEDBACK



*Thank you!*

