VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Electronics Engineering (SENSE)

B.Tech – Electronics & Computer Engineering

ECE4003 – EMBEDDED SYSTEM DESIGN

LAB RECORD

(L64+L65+L66)

Submitted By

18BLC1017 – Dipan Polley

Submitted To

Prof. CHANTHINI BHASKAR

DATE: 09/10/2020

LAB – 10 Working with SPI

LABTASK-1:

AIM:

Write a program to implement a SPI communication between two Arduino Uno. Configure one of the Arduino as master and other as slave. Master is interfaced with 4x4 keypad and slave is connected with 16x2 LCD. Establish a SPI communication between master and slave display each key press on the master to the slave LCD. Simulate and verify this logic on Arduino Uno using Tinkercad circuits simulator.

API REQUIRED:

- **pinMode(pin, mode)** - Configures the specified pin to either input or output.
  - pin: the Arduino pin number to set the mode of.
  - **mode**: INPUT, OUTPUT, or INPUT_PULLUP.
- **digitalWrite(pin, value)** - Write a HIGH or a LOW value to a digital pin.
  - pin: the Arduino pin number.
  - value: HIGH or LOW.
- **delay(ms)** - Pauses the program for the amount of time (in milliseconds)
  - ms: the number of milliseconds to pause.
- **millis()** - Returns the number of milliseconds passed since the Arduino board began running the current program. Data type returned is unsigned long.
- **lcd.begin(cols, rows)-**Initializes the interface to LCD, and specifies dimensions
- **lcd.print(data)/lcd.write(data)** - the character to write to the display
- **lcd.clear()-**Clears the LCD screen and positions the cursor on upper-left corner
- **Keypad(makeKeymap(userKeymap), row[], col[], rows, cols) –** Constructor that instantiates a Keypad object
- **char getKey()** - Returns the key that is pressed, if any.

PROGRAM:

Master:

```
#include <Keypad.h>

#define CLK 10
#define SS 11
#define MOSI 12
#define MISO 13
#define FALSE 0
```

```
#define TRUE !FALSE

const byte nr = 4;
const byte nc = 4;
char keys[nr][nc]= { {'1', '2', '3', 'A'},
                                        {'4', '5', '6', 'B'},
                                        {'7', '8', '9', 'C'},
                                        {'*', '0', '#', 'D'}  };

byte rows[nr] = {9,8,7,6};
byte cols[nc]= {5,4,3,2};
Keypad myKeypad= Keypad(makeKeymap(keys), rows, cols, nr, nc);

unsigned long prevTick = 0.0;
unsigned long lastTime = 0.0;
int clockState = LOW;
int prevClkState = LOW;
unsigned int clkCounter = 0;
unsigned int pressTime;
char keypressed;
int bitNum[4];
int bitSent[] = {FALSE, FALSE, FALSE, FALSE};
int bitsToSend = 0;
int ii;

void setBits(char keypressed);

void setup()
{
        pinMode(CLK, OUTPUT);
        pinMode(MOSI, OUTPUT);
        pinMode(MISO, INPUT);
        pinMode(SS, OUTPUT);
        digitalWrite(CLK, LOW);
        digitalWrite(MOSI, LOW);
        digitalWrite(SS, HIGH);
}

void loop()
{
        //clock
        if ((millis() - prevTick ) >= 1000)
        {
        clockState = !clockState;
        digitalWrite(CLK, clockState);
        prevTick = millis();
        clkCounter++;
   }

        //send key press
        if(bitsToSend > 0)
   {
        if (!(bitSent[bitsToSend - 1]))
      {
         if((clkCounter - pressTime) > (-1 * bitsToSend + 5))
         {
         if((clockState == LOW) && ((millis() - lastTime) > 1000))
           {
```

```
                        digitalWrite(SS, LOW);
                        digitalWrite(MOSI, bitNum[bitsToSend - 1]);
                        bitSent[bitsToSend - 1] = TRUE;
                        bitsToSend--;
                        lastTime = millis();
                    }
                }
            }
        }
            else
        {
                    if(digitalRead(MISO) == HIGH)
            {
                    if((millis() - lastTime) > 2000)
                {
                    digitalWrite(SS, HIGH);
                    digitalWrite(MOSI, LOW);
                }
            }

            //pressing
                    keypressed = myKeypad.getKey();
                    if (keypressed != NO_KEY)
            {
                    setBits(keypressed);
                    pressTime = clkCounter;
                    lastTime = millis();

            bitsToSend = 4;
            for(ii = 0 ; ii < 4 ; ii++)
            {
            bitSent[ii] = FALSE;
            }
                    }
        }


            delay(10);

}


//keypad ip to 4bits
void setBits(char keypressed)
{
            switch(keypressed)
    {
            case '0':
            bitNum[3] = 0;
                    bitNum[2] = 0;
                    bitNum[1] = 0;
                    bitNum[0] = 0;
                    break;
            case '1':
            bitNum[3] = 0;
                    bitNum[2] = 0;
                    bitNum[1] = 0;
                    bitNum[0] = 1;
```

```
            break;
        case '2':
        bitNum[3] = 0;
            bitNum[2] = 0;
            bitNum[1] = 1;
            bitNum[0] = 0;
            break;
        case '3':
        bitNum[3] = 0;
            bitNum[2] = 0;
            bitNum[1] = 1;
            bitNum[0] = 1;
            break;
        case '4':
        bitNum[3] = 0;
            bitNum[2] = 1;
            bitNum[1] = 0;
            bitNum[0] = 0;
            break;
        case '5':
        bitNum[3] = 0;
            bitNum[2] = 1;
            bitNum[1] = 0;
            bitNum[0] = 1;
            break;
        case '6':
        bitNum[3] = 0;
            bitNum[2] = 1;
            bitNum[1] = 1;
            bitNum[0] = 0;
            break;
        case '7':
        bitNum[3] = 0;
            bitNum[2] = 1;
            bitNum[1] = 1;
            bitNum[0] = 1;
            break;
        case '8':
        bitNum[3] = 1;
            bitNum[2] = 0;
            bitNum[1] = 0;
            bitNum[0] = 0;
            break;
        case '9':
        bitNum[3] = 1;
            bitNum[2] = 0;
            bitNum[1] = 0;
            bitNum[0] = 1;
            break;
        case 'A':
        bitNum[3] = 1;
            bitNum[2] = 0;
            bitNum[1] = 1;
            bitNum[0] = 0;
            break;
        case 'B':
        bitNum[3] = 1;
            bitNum[2] = 0;
```

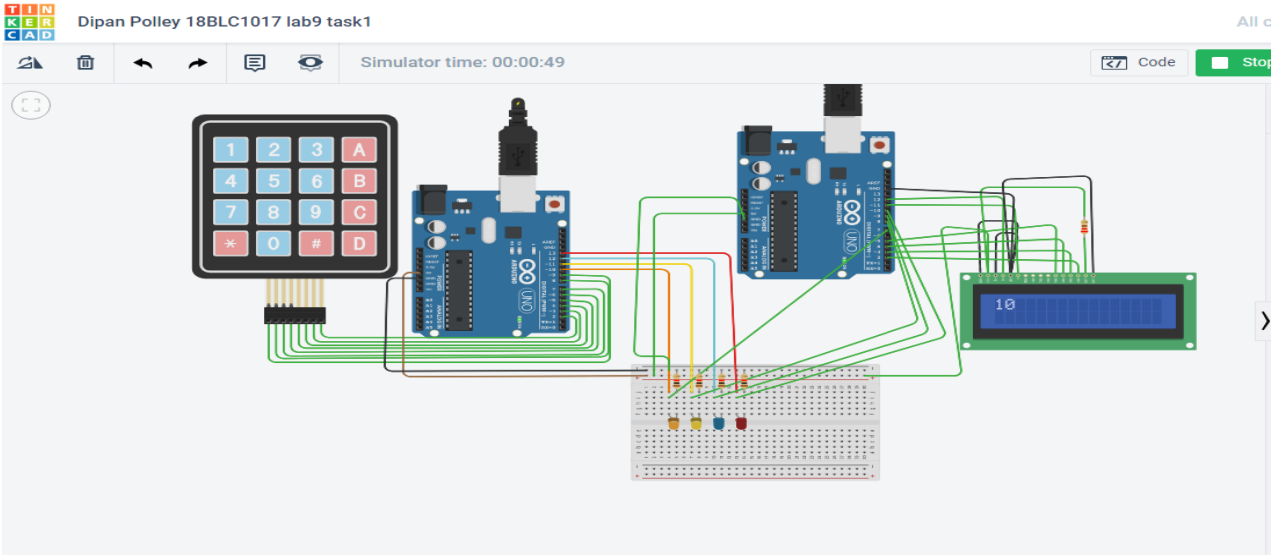
```
                bitNum[1] = 1;
                bitNum[0] = 1;
                break;
        case 'C':
        bitNum[3] = 1;
                bitNum[2] = 1;
                bitNum[1] = 0;
                bitNum[0] = 0;
                break;
        case 'D':
        bitNum[3] = 1;
                bitNum[2] = 1;
                bitNum[1] = 0;
                bitNum[0] = 1;
                break;
        case '*':
        bitNum[3] = 1;
                bitNum[2] = 1;
                bitNum[1] = 1;
                bitNum[0] = 0;
                break;
        case '#':
        bitNum[3] = 1;
                bitNum[2] = 1;
                bitNum[1] = 1;
                bitNum[0] = 1;
                break;
   }
}
```

Slave:

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
int clkState = LOW;
int prevClkState = LOW;
byte data = 0x00;
unsigned long timerStart;
int bitPos = 3;

#define CLK 7
#define SS 8
#define MOSI 9
#define MISO 10

void setup()
{
        pinMode(CLK, INPUT);
        pinMode(MOSI, INPUT);
         pinMode(MISO, OUTPUT);
        pinMode(SS, INPUT);

        digitalWrite(MISO, LOW);
        lcd.begin(16, 2);
        lcd.setCursor(0, 1);
}
```

```
void loop()
{
        clkState = digitalRead(CLK);
        if (clkState != prevClkState)
  {
        prevClkState = clkState;
        if (digitalRead(SS) == LOW)
        {
        if (clkState == HIGH)
       {
        if (digitalRead(MOSI) == LOW)
          {
                data &= ~(0x01 << bitPos);
                bitPos--;
          }
                else
          {
                data |= (0x01 << bitPos);
                bitPos--;
          }
                if (bitPos < 0)
          {
                delay(500);
                digitalWrite(MISO, HIGH);
            delay(1000);
                digitalWrite(MISO, LOW);
            bitPos = 3;
                        lcd.clear();
                                        lcd.print(data);
          }
        }
      }
  }


        delay(10);
}
```

OUTPUT:



TINKERCAD LINK:

https://www.tinkercad.com/things/7UsLo2GZwwO-dipan-polley-18blc1017-lab9-task1/editel?sharecode=e2rb_-jcYVuc-uAWcJ40PqlHPQkl1LW6FGX1dhusL0Q

INFERENCE:

Here we learned the basic working of SPI, and implemented it using 2 Arduinos for inter-device serial communication.

CHALLENGIN TASK:

AIM:

Write a program to implement a SPI communication between two Arduino Uno. Configure one of the Arduino as master and other as slave. Master Arduino connected with LCD and slave Arduino connected with temperature sensor. Transfer temperature value read by slave to master and display it on LCD using SPI communication protocol. Simulate and verify this logic on Arduino Uno using Tinkercad circuits simulator.

API REQUIRED:

- **pinMode(pin, mode)** - Configures the specified pin to either input or output.
  - pin: the Arduino pin number to set the mode of.
  - **mode**: INPUT, OUTPUT, or INPUT_PULLUP.
- **digitalWrite(pin, value)** - Write a HIGH or a LOW value to a digital pin.
  - pin: the Arduino pin number.
  - value: HIGH or LOW.
- **delay(ms)** - Pauses the program for the amount of time (in milliseconds)
  - ms: the number of milliseconds to pause.
- **millis()** - Returns the number of milliseconds passed since the Arduino board began running the current program. Data type returned is unsigned long.
- **lcd.begin(cols, rows)-**Initializes the interface to LCD, and specifies dimensions
- **lcd.print(data)/lcd.write(data)** - the character to write to the display
- **lcd.clear()-**Clears the LCD screen and positions the cursor on upper-left corner
- **bitRead(x, n)** - Reads a bit of a number.
  - **x**: the number from which to read.
  - **n**: which bit to read, starting at 0 for the least-significant (rightmost) bit.

PROGRAM:

Master:

```
#define CLK 10
#define SS 11
#define MOSI 12
#define MISO 13
#define FALSE 0
#define TRUE !FALSE

unsigned long prevTick = 0.0;
unsigned long lastTime = 0.0;
int clockState = LOW;
```

```
int prevClkState = LOW;
unsigned int clkCounter = 0;
unsigned int pressTime;
int bitNum[8];
int bitSent[] = {FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE};
int bitsToSend = 0;
int ii;
float temp, tempc;
byte t;

void setup()
{
 Serial.begin(9600);
        pinMode(CLK, OUTPUT);
        pinMode(MOSI, OUTPUT);
   pinMode(MISO, INPUT);
        pinMode(SS, OUTPUT);
        digitalWrite(CLK, LOW);
        digitalWrite(MOSI, LOW);
        digitalWrite(SS, HIGH);
}

void loop()
{
        //clock
        if ((millis() - prevTick ) >= 1000)
        {
        clockState = !clockState;
        digitalWrite(CLK, clockState);
        prevTick = millis();
        clkCounter++;
   }

        temp = analogRead(A4);
        tempc = (temp*4.88)/10;
        t = tempc;
        tempc -= 50;

        for(int i=0;i<8;i++)
   {
     bitNum[i] = bitRead(t, i);
     Serial.print(bitNum[i]);
   }
        Serial.println();
        delay(1000);

        if(bitsToSend > 0)
   {
        if (!(bitSent[bitsToSend - 1]))
      {
        if((clkCounter - pressTime) > (-1 * bitsToSend + 5))
        {
        if((clockState == LOW) && ((millis() - lastTime) > 1000))
          {
                digitalWrite(SS, LOW);
          digitalWrite(MOSI, bitNum[bitsToSend - 1]);
                bitSent[bitsToSend - 1] = TRUE;
                bitsToSend--;
```

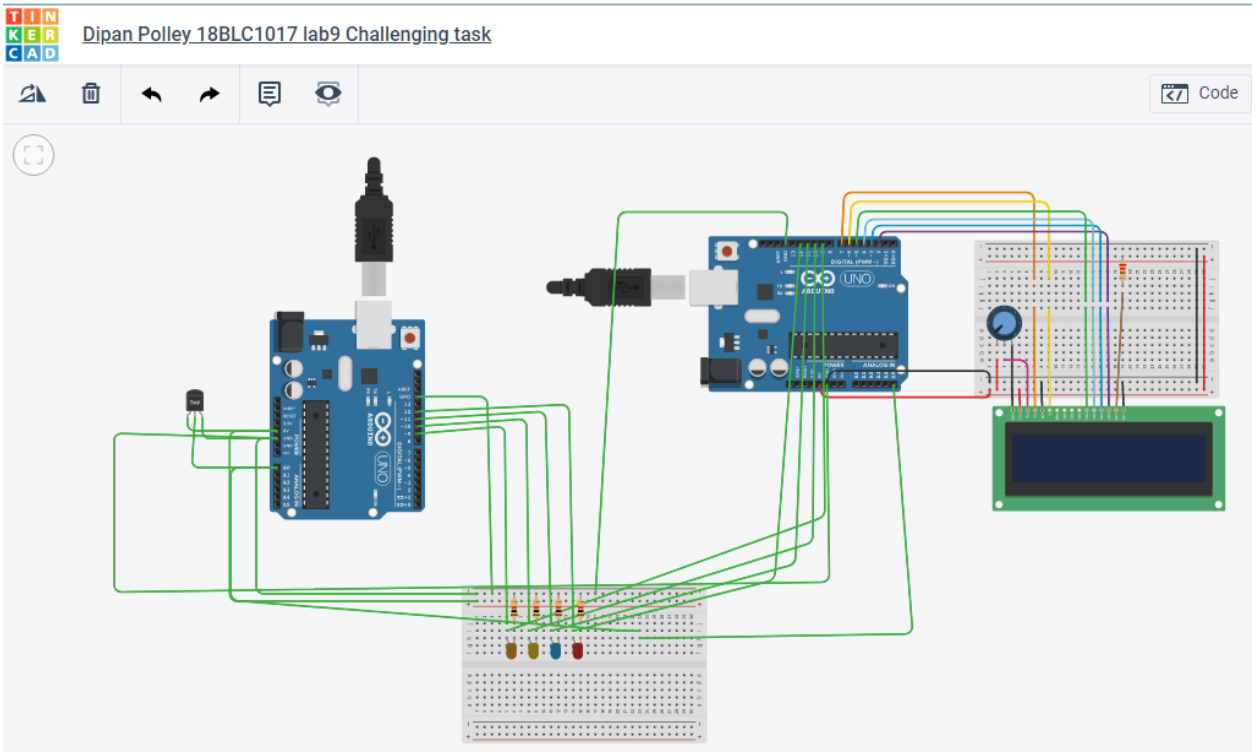
```
                    lastTime = millis();
                  }
                }
              }
            }
                else
            {
                    if(digitalRead(MISO) == HIGH)
                {
                    if((millis() - lastTime) > 2000)
                {
                    digitalWrite(SS, HIGH);
                    digitalWrite(MOSI, LOW);
                }
              }

                    pressTime = clkCounter;
                    lastTime = millis();

                bitsToSend = 8;
                for(ii = 0 ; ii < 8 ; ii++)
              {
                bitSent[ii] = FALSE;
              }
                }
                delay(10);
        }
```

Slave:

```
        #include <LiquidCrystal.h>

        LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
        int clkState = LOW;
        int prevClkState = LOW;
        byte data = 0x00;
        unsigned long timerStart;
        int bitPos = 7;

        #define CLK 7
        #define SS 8
        #define MOSI 9
        #define MISO 10

        void setup()
        {
                pinMode(CLK, INPUT);
                pinMode(MOSI, INPUT);
           pinMode(MISO, OUTPUT);
                pinMode(SS, INPUT);
                digitalWrite(MISO, LOW);
                lcd.begin(16, 2);
                lcd.setCursor(0, 1);
        }

        void loop()
        {
                clkState = digitalRead(CLK);
```

```
            if (clkState != prevClkState)
        {
            prevClkState = clkState;
            if (digitalRead(SS) == LOW)
            {
            if (clkState == HIGH)
           {
            if (digitalRead(MOSI) == LOW)
              {
                    data &= ~(0x01 << bitPos);
                    bitPos--;
              }
                    else
              {
                    data |= (0x01 << bitPos);
                    bitPos--;
              }
                    if (bitPos < 0)
              {
                    delay(500);
                    digitalWrite(MISO, HIGH);
                delay(1000);
                    digitalWrite(MISO, LOW);
                bitPos = 7;
                            lcd.clear();
                                        lcd.print(data);
              }
           }
        }
      }
            delay(10);
}
```

OUTPUT:



TINKERCAD LINK:

https://www.tinkercad.com/things/il4sQTcXozI-dipan-polley-18blc1017-lab9-challenging-task/editel?sharecode=Za2DsGG7hzrZx62xXnrSf6t-cePbdUBaIdhF8nVTZ3k

INFERENCE:

Here, input is obtained from the temperature sensor in one of the Arduinos, then converted to an 8-bit value stored in the 'bitNum' array. These values are sent bitwise using the SPI protocol to the other device, where it is printed on the screen.

RESULT:

Thus, we learned the working of SPI protocol, and simulated its implementation using Arduino UNOs.