Contents

[OBJECTIVE: 1](#_Toc103837839)

[COMPONENTS: 3](#_Toc103837840)

[INTRODUCTION: 3](#_Toc103837841)

[IR Sensor: 3](#_Toc103837842)

[Working: 3](#_Toc103837843)

[Applications: 4](#_Toc103837844)

[Stepper motors: 4](#_Toc103837845)

[Applications of stepper motor: 4](#_Toc103837846)

[CHARACTER L.C.D: 5](#_Toc103837847)

[STEPS: 6](#_Toc103837848)

[CODE: 7](#_Toc103837850)

**LAB MANUAL:**

INTERFACING IR SENSOR AND STEPPER MOTOR WITH TIVA

# OBJECTIVE:

Objective of this lab manual is to interface IR sensor and stepper motor with Tiva Launchpad. And to make a kind of application with these two components.

# COMPONENTS:

Series of components are given below:

* Tiva microcontroller
* Character LCD
* IR sensor
* Stepper motor
* Diode
* Capacitor
* Voltage regulator
* Connecting wires

# INTRODUCTION:

## IR Sensor:

IR sensor is a optoelectronic sensor which uses infrared radiation to detect motion of nearby objects. It consists of following things:

* IR transmitter
* IR receiver
* OUTPUT Led
* Variable resistor

## Working:

How the IR sensor works is simple. First of all, IR led transmits light. We cannot see that light because its wavelength is higher than visible light wavelength. Light emitting angle of IR led is 20-60 degrees. Light emitting angle and intensity of light of led really depends upon type of IR sensor which we are using.

Then we have photodiode. Photodiode acts as an IR receiver because when light falls on it, it starts conducting current. Photodiode in that case is reverse biased. It is also important to understand that amount of current is directly proportional to amount of light which is falling on it.

So IR sensor works when something or some material approaches to it. When there is a motion in its vicinity then due to reflection of light and falling of that light on photodiode, it starts working.

## Applications:

Applications of IR sensor are given below:

1. It is used in Home automations.
2. It is used to detect any burglar in proximity.
3. It is used in metal detection.
4. It is used in in gas leakage detection
5. It is also used in climatology. Etc

## Stepper motors:

Stepper motors are actually DC motors that are designed in such aa way that they move in discrete steps. Stepper motors have multiple coils which are arranged in different groups. And these groups are called phases. If we want to move the motor, we have to energize it in phases.

Stepper motors are actually motors which are used to achieve precise motion control applications. Like other motors, it also come in different shapes and sizes.

Stepper motors are good for applications which involve such performance:

1. Low speed torque applications
2. Speed control applications
3. Positioning applications

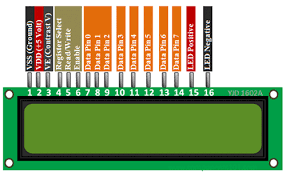
## Applications of stepper motor:

Applications of stepper motor are given below:

1. They are used in printing equipment.
2. They are used in Textile machines.
3. They are used in Gaming industries.
4. They are used in Medical imaging machinery industry.
5. They are used in robotics industries.
6. Welding equipment. Etc

## CHARACTER L.C.D:

We will be displaying numbers on the character lcd. These LCDs are ideal for displaying text/characters only, hence the name ‘Character LCD’. The display has an LED backlight and can display 32 ASCII characters in two rows with 16 characters on each row. Most of the LCD Displays available in the market are 16\*2 (That means, the LCD displays are capable of displaying 2 lines each having 16 Characters a), 20X4 LCD Displays (4 lines, 20 characters).

****

GND should be connected to the ground of Tiva launchpad

VCC is the power supply for the LCD which we connect the 5 volts pin on the Tiva launchpad.

Vo (LCD Contrast) controls the contrast and brightness of the LCD. Using a simple voltage divider with a potentiometer, we can make fine adjustments to the contrast.

RS (Register Select) pin lets the Tiva launchpad tell the LCD whether it is sending commands or the data. Basically this pin is used to differentiate commands from the data.

For example, when RS pin is set to LOW, then we are sending commands to the LCD (like set the cursor to a specific location, clear the display, scroll the display to the right and so on). And when RS pin is set on HIGH we are sending data/characters to the LCD.

R/W (Read/Write) pin on the LCD is to control whether or not you’re reading data from the LCD or writing data to the LCD. Since we’re just using this LCD as an OUTPUT device, we’re going to tie this pin LOW. This forces it into the WRITE mode.

E (Enable) pin is used to enable the display. Meaning, when this pin is set to LOW, the LCD does not care what is happening with R/W, RS, and the data bus lines; when this pin is set to HIGH, the LCD is processing the incoming data.

D0-D7 (Data Bus) are the pins that carries the 8 bit data we send to the display. For example, if we want to see the uppercase ‘A’ character on the display we will set these pins to 0100 0001(according to the ASCII table) to the LCD.

A-K (Anode & Cathode) pins are used to control the backlight of the LCD.

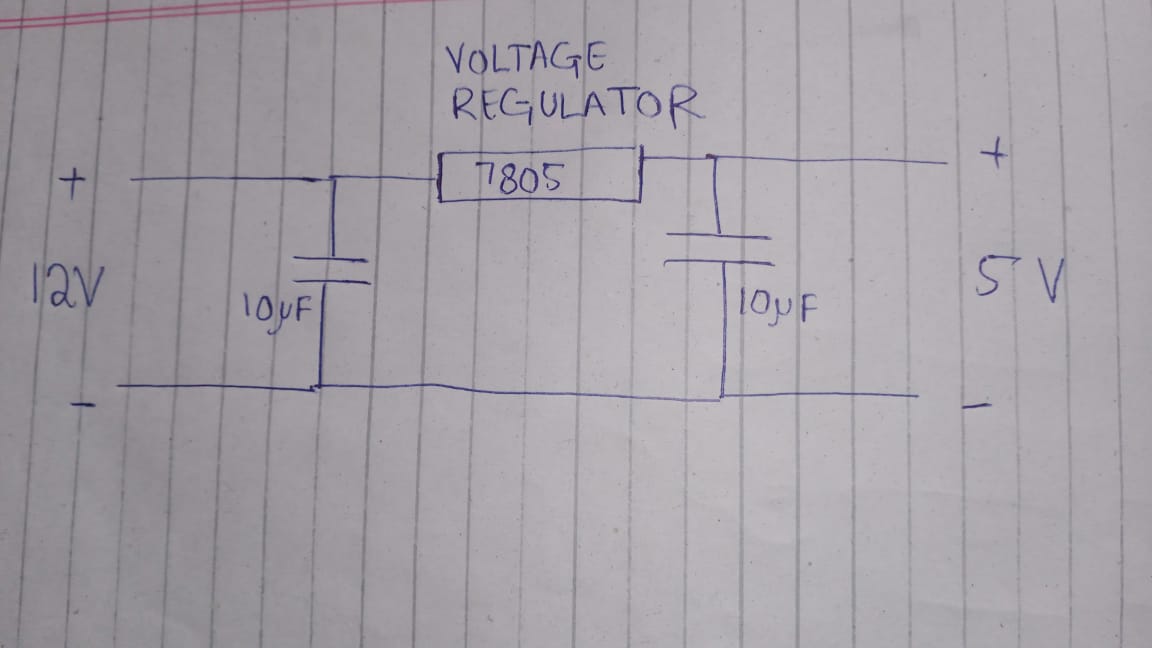


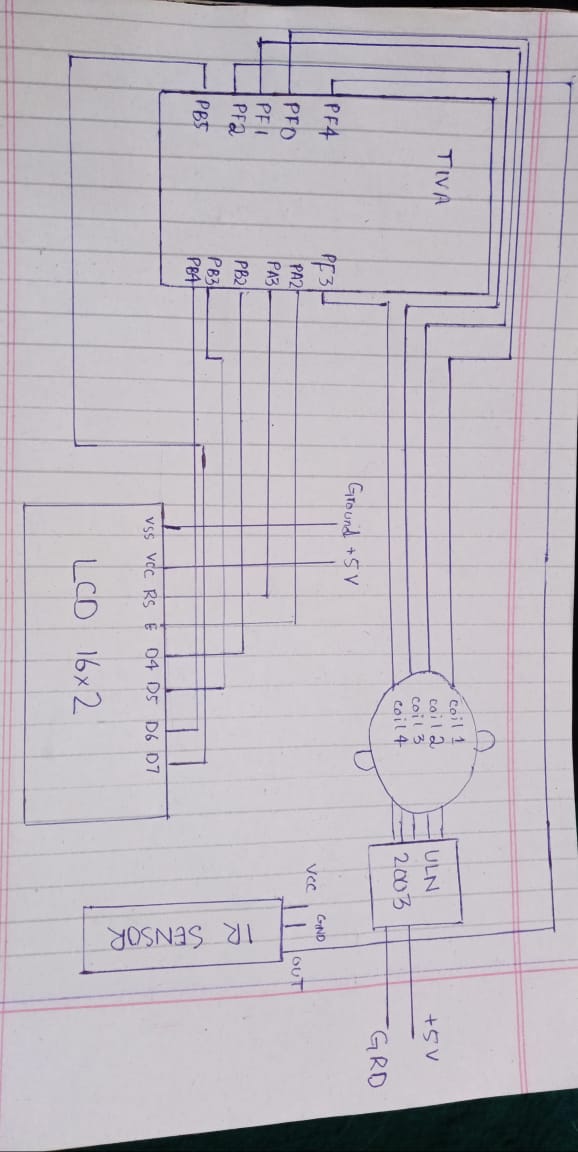
# 

# STEPS:

* Make connections according to schematic diagram.
* We will be giving separate power to microcontroller and separate to rest of system.
* Use Keil software to upload code into your Tiva.
* Push reset button onto your microcontroller before working on it.
* I have used diode for safety purposes
* Voltage regulator 7805 is used to convert and regulate 12V power supply to 5V.
* That 5V is being supplied to LCD 16\*2.
* Connect the ULN2003 driver, which is already attached to stepper motor, to 5V.
* Always remember to common ground all the devices.

# SCHEMATIC DIAGRAM:





# C:\Users\dell\Desktop\flowchartCode.PNG

# CODE:

Here is your code.

#include <stdio.h> // standard C library

#include <string.h>

#include <stdlib.h>

#include <math.h>

#include "TM4C123.h" /\* include register defintion file of TM4C123GH6PM \*/

//========================================================================s

#define LCD\_RS (\*((volatile unsigned long \*)0x40004020)) //RS - PA3

#define lcdEN (\*((volatile unsigned long \*)0x40004010)) //EN - PA2

#define LCD\_DATA (\*((volatile unsigned long \*)0x400050F0)) //DB4-7 are mapped to PB2-5

//Defines for PortA

#define GPIO\_PORTA\_DIR\_R (\*((volatile unsigned long \*)0x40004400))

#define GPIO\_PORTA\_AFSEL\_R (\*((volatile unsigned long \*)0x40004420))

#define GPIO\_PORTA\_DEN\_R (\*((volatile unsigned long \*)0x4000451C))

#define GPIO\_PORTA\_LOCK\_R (\*((volatile unsigned long \*)0x40004520))

#define GPIO\_PORTA\_CR\_R (\*((volatile unsigned long \*)0x40004524))

#define GPIO\_PORTA\_AMSEL\_R (\*((volatile unsigned long \*)0x40004528))

#define GPIO\_PORTA\_PCTL\_R (\*((volatile unsigned long \*)0x4000452C))

//Defines for PortB

#define GPIO\_PORTB\_DATA\_R (\*((volatile unsigned long \*)0x400053FC))

#define GPIO\_PORTB\_DIR\_R (\*((volatile unsigned long \*)0x40005400))

#define GPIO\_PORTB\_AFSEL\_R (\*((volatile unsigned long \*)0x40005420))

#define GPIO\_PORTB\_DEN\_R (\*((volatile unsigned long \*)0x4000551C))

#define GPIO\_PORTB\_LOCK\_R (\*((volatile unsigned long \*)0x40005520))

#define GPIO\_PORTB\_CR\_R (\*((volatile unsigned long \*)0x40005524))

#define GPIO\_PORTB\_AMSEL\_R (\*((volatile unsigned long \*)0x40005528))

#define GPIO\_PORTB\_PCTL\_R (\*((volatile unsigned long \*)0x4000552C))

#define SYSCTL\_RCGC2\_R (\*((volatile unsigned long \*)0x400FE108))

//Define for delays

#define delay4500ns 360

#define delay400us 3200

#define delay50ms 400000

#define delay150ms 1200000

#define delay2000us 16000

#define delay500ms 4000000

#define delay1s 8000000

//---------------------------------------------------------------------------

//---------------------------------------------

void Port\_Init(void);

void lcdENPulse(unsigned long time);

void LCD\_Cmd(unsigned char c);

void LCD\_Init(void);

void LCD\_Write\_Char (char c);

void LCD\_String(char \*string);

void lcdGoto(unsigned char address);

void LCD\_Clear(void);

void delay\_ms(int n);

void delay\_us(int n);

void Delay\_ms(int n);

//---------------------------------------------

long val1=0,val2=0;

unsigned char digits[5]={0};

char factor = -1;

unsigned int cnt = 1;

void resetCalculation(int error\_code){

LCD\_Clear();

switch(error\_code){

case 1:

LCD\_String("Limit exceed");

break;

}

delay\_ms(1000);

LCD\_Clear();

cnt = 1; val1=0;val2=0;factor=-1;

}

//===================================================================

void LCD\_Goto\_XY ( unsigned char x, unsigned char y)

{

unsigned char row\_start\_address [] = {0x80 , 0xC0 };

// Move cursor to (x,y) location on display

LCD\_Cmd (row\_start\_address [y -1] + x - 1);

delay\_ms(170);

}

int main(void){

unsigned int state=0,state2=0;

Port\_Init();

LCD\_Init();

LCD\_String("IR Dispense"); //Show prompt "Enter password"

LCD\_Goto\_XY(6,2);

LCD\_String("Example");

delay\_ms(4000);

//LCD\_Clear(); //Clear screen

GPIOF->DATA = 0;

while(1)

{

int i=0;

state = GPIOF->DATA & 0x10;

if(state){

LCD\_Goto\_XY(1,2);

LCD\_Write\_Char('1');

GPIOF->DATA = 0;

}else{

LCD\_Goto\_XY(1,2);

LCD\_Write\_Char('0');

GPIOF->DATA = 8;

for(i=0; i<250; i++) // take 250 steps to complete one revolution

{

// apply full drive clockwise rotation sequence

GPIOF->DATA = 0x08;

Delay\_ms(10);

GPIOF->DATA = 0x04;

Delay\_ms(10);

GPIOF->DATA = 0x02;

Delay\_ms(10);

GPIOF->DATA = 0x01;

Delay\_ms(10);

}

for(i=0; i<250; i++) // take 250 steps to complete one revolution

{

// apply full drive clockwise rotation sequence

GPIOF->DATA = 0x01;

Delay\_ms(10);

GPIOF->DATA = 0x02;

Delay\_ms(10);

GPIOF->DATA = 0x04;

Delay\_ms(10);

GPIOF->DATA = 0x08;

Delay\_ms(10);

}

}

GPIOF->DATA |= (1<<0);

//GPIOF->DATA = (~state>>3); /\* put it on red LED \*/

/\*

for(i=0; i<250; i++) // take 250 steps to complete one revolution

{

// apply full drive clockwise rotation sequence

GPIOF->DATA = 0x08;

Delay\_ms(10);

GPIOF->DATA = 0x04;

Delay\_ms(100);

GPIOF->DATA = 0x02;

Delay\_ms(100);

GPIOF->DATA = 0x01;

Delay\_ms(400);

}

\*/

}

}//main ends here

//=====================================================================

//-----------------------------------------------------------

void delay\_ms(int n)

{

int i,j;

for(i=0;i<n;i++)

for(j=0;j<3180;j++)

{}

}

/\* Generates a delay in number of miliseocnds wit system clock of 16MHz \*/

void Delay\_ms(int n)

{

int a, b;

for(a = 0 ; a < n; a++)

for(b = 0; b < 3180; b++)

{} /\* execute NOP for one milisecond \*/

}

/\* Micro seconds delay function \*/

void delay\_us(int n)

{

int i,j;

for(i=0;i<n;i++)

for(j=0;j<3;j++)

{}

}

void Port\_Init(void){

//Init for PortB

SYSCTL\_RCGC2\_R |= 0x00000002; // 1) b clock

// delay = SYSCTL\_RCGC2\_R; // delay

GPIO\_PORTB\_CR\_R |= 0x3C; // allow changes to PB2-PB5

GPIO\_PORTB\_AMSEL\_R &= 0x00; // 3) disable analog function

GPIO\_PORTB\_PCTL\_R &= 0x00000000; // 4) GPIO clear bit PCTL

GPIO\_PORTB\_DIR\_R |= 0x3C; // 5.2) PB2-PB5 as OUTPUTS

GPIO\_PORTB\_AFSEL\_R &= 0x00; // 6) no alternate function

GPIO\_PORTB\_DEN\_R |= 0x3C; // 7) enable digital pins to PB2-PB5

//Init for PortA2-3

SYSCTL\_RCGC2\_R |= 0x00000001; // 1) A clock

// delay = SYSCTL\_RCGC2\_R; // delay

GPIO\_PORTA\_CR\_R |= 0x0C; // allow changes to PA2-3

GPIO\_PORTA\_AMSEL\_R &= 0x00; // 3) disable analog function

GPIO\_PORTA\_PCTL\_R &= 0x00000000; // 4) GPIO clear bit PCTL

GPIO\_PORTA\_DIR\_R |= 0x0C; // 5.1) PA2-3 OUTPUTS

GPIO\_PORTA\_AFSEL\_R &= 0x00; // 6) no alternate function

GPIO\_PORTA\_DEN\_R |= 0x0C; // 7) enable digital pins PA2-3

SYSCTL->RCGCGPIO |= 0x20; /\*Enable clock to PORTF \*/

/\* GPIOF->DATA PF3,PF2, PF1, PF0 pin initialization\*/

GPIOF->LOCK = 0x4C4F434B; // unlockGPIOCR register

GPIOF->CR = 0x01; // Enable GPIOPUR register enable to commit

GPIOF->PUR |= 0x11; // Enable Pull Up resistor PF4

GPIOF->DIR |= 0x0F; /\* GPIOF->DATA PF3,PF2, PF1, PF0as output \*/

GPIOF->DEN |= 0x0F; /\* GPIOF->DATA PF3,PF2, PF1, PF0 as digital pins \*/

GPIOF->DIR |= 0x02; //set PF1 as an output and PF4 as an input pin

GPIOF->DEN |= 0x12; // Enable PF1 and PF4 as a digital GPIO pins

}

void lcdENPulse(unsigned long time){ //set EN high for x time

lcdEN |= 0x04;

delay\_us(time);

lcdEN &=~ 0x04; //return en to 0

}

void LCD\_Init(void) {

lcdEN =0x00<<2; //set enable to 0

LCD\_DATA =0x00<<2; //set data to 0

delay\_ms(50);

LCD\_RS=0x00<<2; //RS to 0

LCD\_DATA =0x3<<2; //waking up instructions

lcdENPulse(delay4500ns); //latch enable line for 450 ns

delay\_ms(50);

LCD\_DATA =0x3<<2;

lcdENPulse(delay4500ns);

delay\_ms(2);

lcdENPulse(delay4500ns);

delay\_ms(2);

LCD\_DATA =0x2<<2;

lcdENPulse(delay4500ns);

delay\_ms(2);

//Data display lines are N = 0/1 <-> 1/2 lines, Data length DL = 0/1 <-> 4/8 bits, Character Font F = 0/1 <-> 5x8/5x10

//We need 4-bit data, 2 lines, 5x8 font (0,1,0)

//Cursor I/D to increment and display don't shift

LCD\_Cmd (0x02);delay\_ms(50); /\* 4bit mode \*/

LCD\_Cmd (0x28);delay\_ms(50); /\* Initialization of 16X2 LCD in 4bit mode \*/

LCD\_Cmd (0x0C);delay\_ms(50);/\* Display ON Cursor OFF \*/

LCD\_Cmd (0x06);delay\_ms(50); /\* Auto Increment cursor \*/

LCD\_Cmd (0x01);delay\_ms(50); /\* Clear display \*/

LCD\_Cmd (0x80);delay\_ms(50); /\* Cursor at home position \*/

}

void LCD\_Cmd(unsigned char c){ // function for setting commands

LCD\_RS =0x00; //line low

delay\_us(400);

LCD\_DATA=(c&0xF0)>>2; //send first high nible to PB2-5

lcdENPulse(delay4500ns);

delay\_ms(50);

LCD\_DATA=(c&0x0F)<<2;//set low niblle

lcdENPulse(delay4500ns);

delay\_ms(50);

}

void LCD\_Write\_Char (char c){ //function for setting data

LCD\_RS =0x08; //RS on for sending data

delay\_us(400);

LCD\_DATA=(c&0xF0)>>2; //high nibble

lcdENPulse(delay4500ns);

delay\_ms(1);

LCD\_DATA=(c&0x0F)<<2; //low nibble

lcdENPulse(delay4500ns);

delay\_ms(1);

LCD\_RS =0x00;

}

void lcdGoto(unsigned char address){

LCD\_Cmd(address); delay\_ms(2);

}

void LCD\_Clear(void){

LCD\_Cmd(0x01); delay\_ms(50); LCD\_Cmd(0x00); //clear lcd

LCD\_Cmd(0x02); delay\_ms(50); LCD\_Cmd(0x00); //"Home" LCD

}

void LCD\_String(char \*string) {//function for writing strings

while(\*string) {

LCD\_Write\_Char(\*string);

string++;

}

}