Dr. AMBEDKAR INSTITUTE OF TECHNOLOGY

(An Autonomous Institute, Affiliated to Visvesvaraya Technological University, Belagavi, Accredited by NAAC, with 'A' Grade)
Near JnanaBharathi Campus, Bengaluru – 560056

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Lab Manual

on

"Internet of Things (IoT)"

Subject Code: 18CSL65

Staffs - In – Charge

Dr. Smitha Shekar B, Associate Professor, Dr. AIT
Ms. Lavanya Santhosh, Assistant Professor, Dr. AIT

2020-21

A. LABORATORY OVERVIEW

Degree:	**BE**	Programme:	**CS & E**
Semester:	**6**	Academic Year:	**2020-21**
Laboratory Title:	**Internet of Things Laboratory**	Laboratory Code:	**18CSL65**
L-T-P-S:	**0-0-1-0**	Duration of SEE:	**3 Hrs**
CIE Marks:	**50**	SEE Marks:	**50**

B. DESCRIPTION

1. PREREQUISITES

- Embedded Systems and Programming skills.
- Basic knowledge of C language
- Basic understanding of using a C library and header files
- Wiring Library Reference

2. BASE COURSE

- **Internet of Things (18CSL65)**

3. COURSE OUTCOMES

At the end of the course, the student will be able to;

CO1: Examine the features and process of integration of Launchpad with IoT applications.
CO2: Discover the role of TCP/UDP protocols in serving as communication models for IoT.
CO3: Interpret the Sensor data collected by interfacing the sensors to the Wi-Fi module on an embedded platform.

4. RESOURSES REQUIRED

- Hardware resources
  - Desktop PC
  - Windows / Linux operating system
  - **Tiva-C LaunchPad (http://www.ti.com/tool/ek-tm4c123gxl)**
  - **CC3100 BoosterPack**
  - **MSP432 P401R**
- Software resources
  - Energia IDE
  - Launchpad Drivers

Prepared by
Dr. Smitha Shekar B
Ms. Lavanya Santhosh

Dr. Siddaraju
HOD,

5. RELEVANCE OF THE COURSE:

- To carry out Mini project and Main Project work

6. GENERAL INSTRUCTIONS:

- Implement the program in Energia IDE.
- External practical examination.
  - All laboratory experiments are to be included
  - Students are allowed to pick one experiment from the lot.
  - Marks distribution: Procedure + Conduction + Viva: 10 + 30 +10 (50)
  - Change of experiment is allowed only once and marks allotted to the procedure part to be made zero.

List of Programs

Sl. No.	Programs
1.	Write a program using different components, a)LEDs for configuring and controlling different on board peripherals b)Switches to configure an LED as an output and a switch as an input Pull up mode on TM4C123 Launchpad using Energia software
2.	Write a program to show a) Analog to Digital Conversion and its output being monitored on the serial monitor b) Pulse Width Modulation(PWM) for changing the intensity of the LED c)UART for serial communication using TM4C123 Launchpad and Energia software
3.	Write a program to connect the Launchpad with Wi-Fi network & print the static IP Addresses on the Serial Monitor
4.	Write a program to connect the Launchpad with Wi-Fi network & print the dynamic IP Addresses on the Serial Monitor
5.	Write a program to connect the Launchpad with Wi-Fi & print the local IP, Subnet Mask, Gateway IP on the Serial Monitor.
6.	Illustrate TCP based Client Server Communication Model.
7.	Illustrate UDP based Client Server Communication Model
8.	Write a program for HTTP based webserver to manipulate the GPIO's of WiFi Module CC3100 and blink a LED(HIGH/LOW).
9.	Write a program for sensor based webserver to manipulate the GPIO's of WiFi Module CC3100 and monitor the sensor data connected with it.
10.	Write a program for HTTP based webserver to control the movement of the RSLK.

Experimenting With the Internet of Things

Overview

In this note, we are going to learn about the Internet of Things using Texas Instruments modules. This lab will guide you through procedures and different set-ups which will be required in getting you started with IOT. This manual does not replace the standard guides available from Texas Instruments, but only supplements them. This manual has been written in a simple mannerso that anyone with basic knowledge of electronics and programming can follow it effectively. After having completed this, you will be confident enough to start with your IOT designs.

Tools Used:

For these labs, the tools that we are going to use are listed below:

1. Tiva-C LaunchPad
   http://www.ti.com/tool/ek-tm4c123gxl
2. SensHub BoosterPack
   http://www.ti.com/tool/boostxl-senshub
3. CC3100 BoosterPack
   http://www.ti.com/tool/cc3100boost
4. Analog Systems lab Kit Pro
   http://www.mikroe.com/aslk-pro

Requirements:

In order to get started with the labs included in this manual, you have to install software packages from Texas Instruments website. You will also be guided to download user guides for specific platforms from Texas Instruments. At the first hand, you should be registered with TI, and then only you can download software packages.

Go to my.ti.com and register yourself. Do remember your login ID and password. This will also facilitate you to post your queries while you want to interact with TI engineers and community members at TI e2e forum.

Now, you should follow the mentioned links to download the software packages and user guides.

1. **Tiva-C series LaunchPad User Guide**. This document enables you to understand the LaunchPad platform and schematics. You are suggested to go through this document first.
   http://www.ti.com/lit/ug/spmu296/spmu296.pdf

2. **TM4C123GH6PM datasheet**. This document is useful while you want to understand the on-chip peripherals available on the TM4C123GH6PM microcontroller featured on the Tiva-C series launchPad.
   http://www.ti.com/lit/ds/spms376e/spms376e.pdf

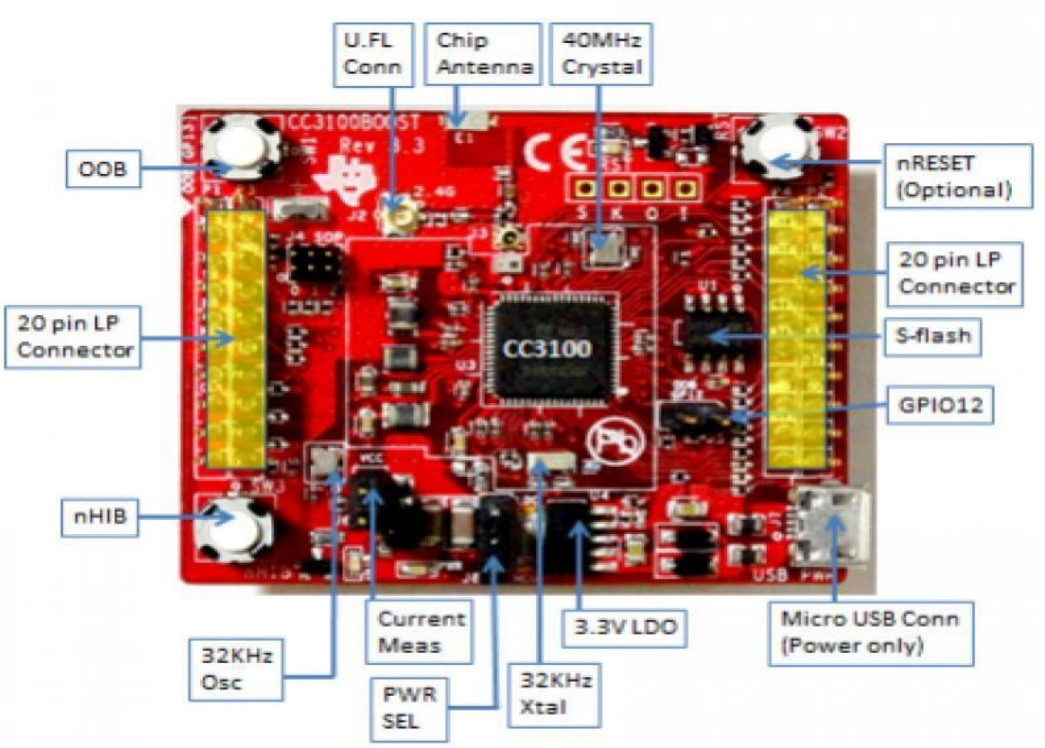
3. **MSP432 P401R** : Ti's first 32 bit Low power + performance MCU. XDS110-ET on board Emulator with fast application UART with optional hardware handshake.

1) Introduction to TM4C123GXL Launchpad

- ARM® Cortex™-M4F 64-pin 80MHz TM4C123GH6PM
- On-board USB ICDI (In-Circuit Debug Interface)
- Micro AB USB port
- Device/ICDI power switch
- BoosterPack XL pinout also supports legacy BoosterPack pinout
- 2 user pushbuttons (SW2 is connected to the WAKE pin)
- Reset button
- 3 user LEDs (1 tri-color device)
- Current measurement test points
- 16MHz Main Oscillator crystal
- 32kHz Real Time Clock crystal
- 3.3V regulator
- Support for multiple IDEs:

The Tiva™ C Series TM4C123G LaunchPad Evaluation Board is a low-cost evaluation platform for ARM Cortex™-M4F-based microcontrollers. The Tiva C Series LaunchPad design highlights the TM4C123GH6PMI microcontroller USB 2.0 device interface, hibernation module, and motion control pulse-width modulator (MC PWM) module. The Tiva C Series Launchpad also features programmable user buttons and an RGB LED for custom applications. The stackable headers of the Tiva C Series TM4C123G LaunchPad BoosterPack XL interface demonstrate how easy it is to expand the functionality with many existing add-on boards as well as future products.

2) Introduction to CC3100 Boosterpack



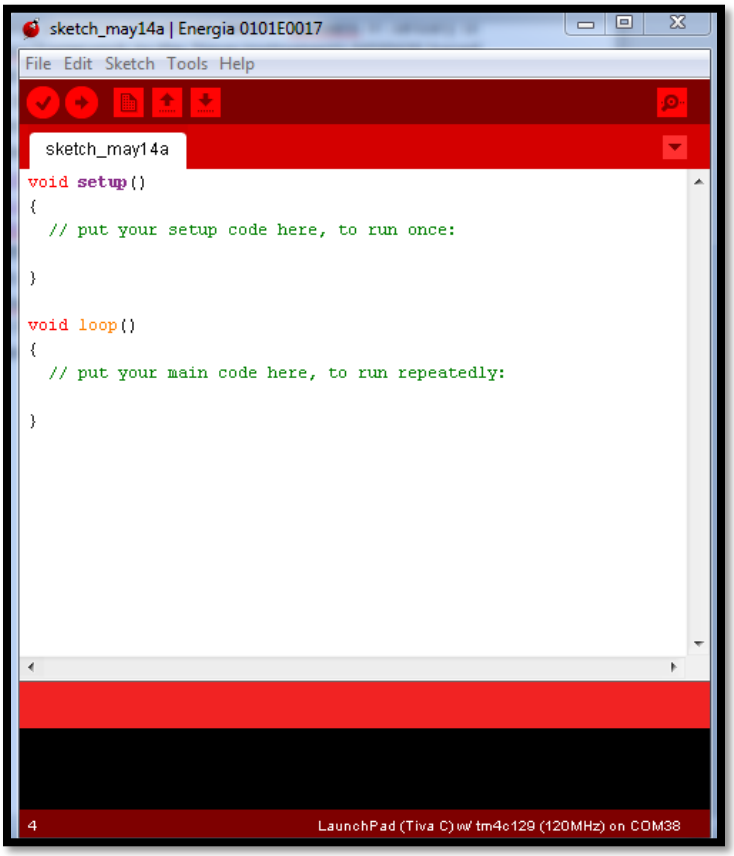
The Texas instrument's CC3100 Simplelink WiFi solution provides the flexibility to add WiFi capability to TI microcontrollers. This internet on chip solution contains everything that you need to build for IOT level designs. Some of the features of this kit are as below

- CC3100 WiFi Network processor
- On board chip antenna
- 2 Push buttons & 4 Led's
- Power from on-board LDO using USB or from MCU.
- 0.8 megabit serial flash
- 40 MHZ crystal,32Khz crystal and oscillator
- USB
- U.FL and chip antenna
- 4 layer PCB

3) Introduction to Energia software

Energia is an open-source electronics prototyping platform started by Robert Wessels in January of 2012 with the goal to bring the Wiring and Arduino framework to the Texas Instruments based LaunchPads.The Energia IDE is cross platform and supported on Mac OS, Windows, and Linux. Energia uses the msp-gcc compiler by Peter Bigot and is based onthe Wiring and Arduino framework. Energia includes an integrated development environment (IDE) that is based on Processing.

Energia Sketch



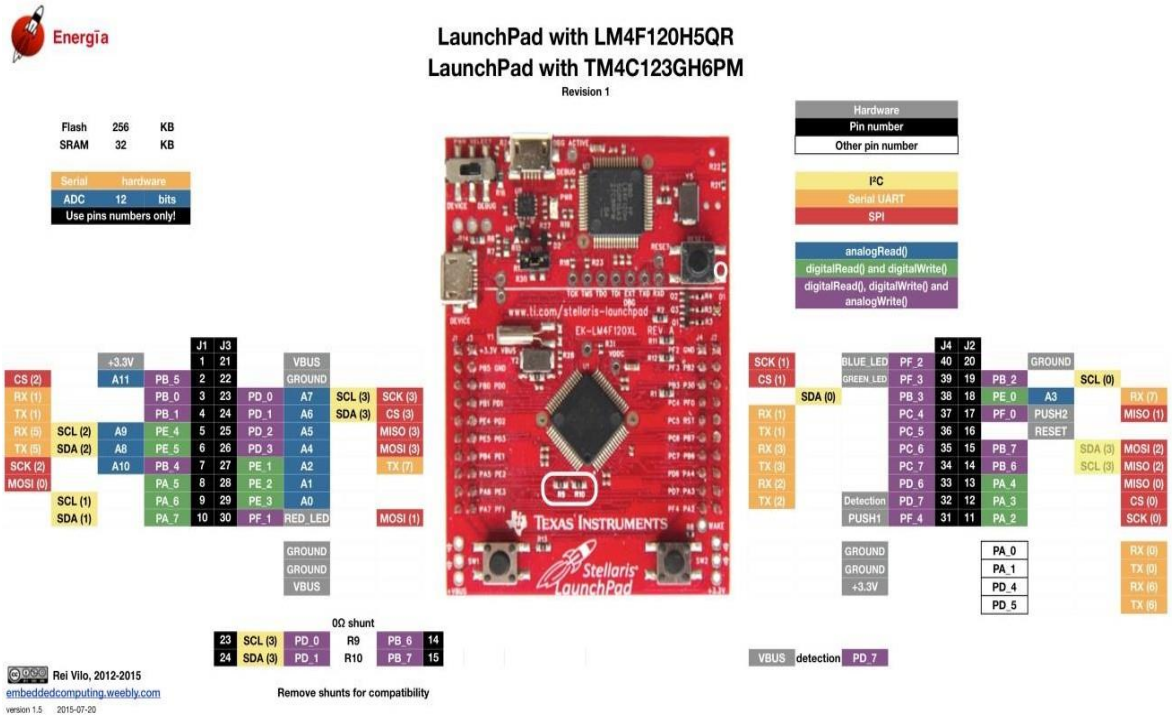
Void Setup()

This is used to initialize the pins of Controller to be used .

Void Loop()

This is used for generating infinite iterations (i.e. repetition) in the program.

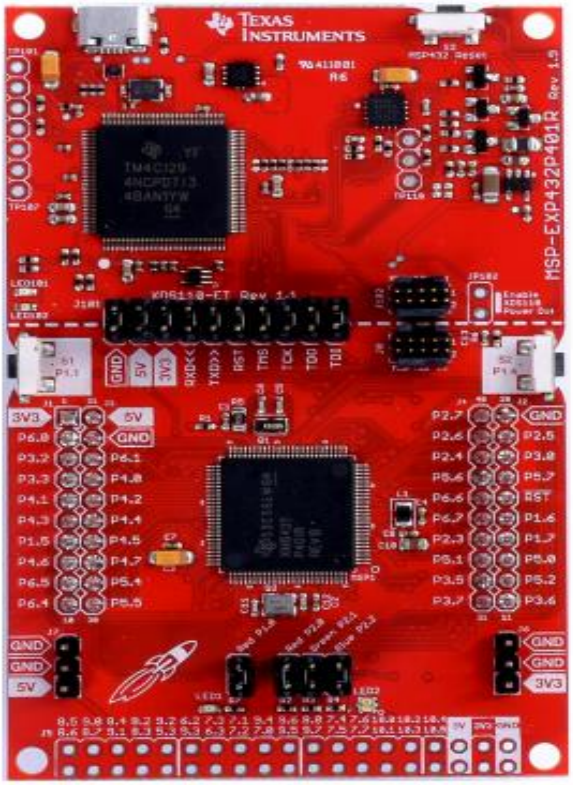
4) TM4C123GH6PM Energia Pinmap



There are 20 Boost XL Pins available in TM4C123G Launchpad and a variety of Texas instruments Boosterpacks available can be connected to those pins, so as to extend the functionality of Launchpad. These pins are multiplexed, which means a single pin can have multiple functions, but only function is active at a given time. With the help of above figure user can select the pins and configure them as required.

5) MSP432P401R Launch pad

- Low-power Arm Cortex-M4F MSP432P401R
- 40-pin LaunchPad development kit standard that leverages the BoosterPack plug-in module ecosystem
- XDS110-ET, an open-source onboard debug probe featuring EnergyTrace+ technology and application UART
- Two buttons and two LEDs for user interaction
- Backchannel UART through USB to PC



The SimpleLink™ MSP-EXP432P401R LaunchPad™ development kit is an easy-to-use evaluation module for the SimpleLink MSP432P401R microcontroller. It contains everything needed to start developing on the SimpleLink MSP432™ low-power +performance Arm® 32-bit Cortex®-M4F microcontroller (MCU), including onboard debug probe for programming, debugging, and energy measurements. The MSP432P401R device supports low-power applications requiring increased CPU speed, memory, analog, and 32-bit performance.

Experiment -1

Write a program using different components,

i) LEDs for configuring and controlling different on board peripherals

ii) Switches to configure an LED as an output and a switch as an input Pull up mode

on TM4C123 Launchpad using Energia software

Objective: In this program you are going to learn how you can control different on boardperipherals on TM4C123 Launchpad using Energia.

Requirements: TIVA Launchpad, Analog sensor.

i)LED blink : In this program we are configuring the on board Red Led as an outputfirst and then Blinking it at 1sec interval.

Functions Used

pinMode()

This function is used to configure any of the GPIO pin as an input or output mode.
The syntax isas shown

pinMode(Pin_No,**Mode**);

Mode: **INPUT, OUTPUT, INPUT_PULLUP or INPUT_PULLDOWN**

digitalWrite()

This function is used to write digital HIGH or digital LOW to the pin that the user desired toconfigure. The syntax is as shown

digitalWrite(pin, HIGH)

PROGRAM

```
void setup()
{
  // put your setup code here, to run once:
  pinMode(RED_LED, OUTPUT);
}

void loop()
{
  // put your main code here, to run repeatedly:
  digitalWrite(RED_LED, HIGH);
  delay(1000);
```

```
  digitalWrite(RED_LED, LOW);
  delay(1000);
}
```

OUTPUT

ii)Switch usage: In this program user is going to first configure an LED as an output and aSwitch as an input Pull Up mode.The state of switch is being monitored to toggle the led.

Functions used

digitalRead()

This function returns the digital status of a pin ,it can have values as HIGH or LOW.The syntaxis as follows

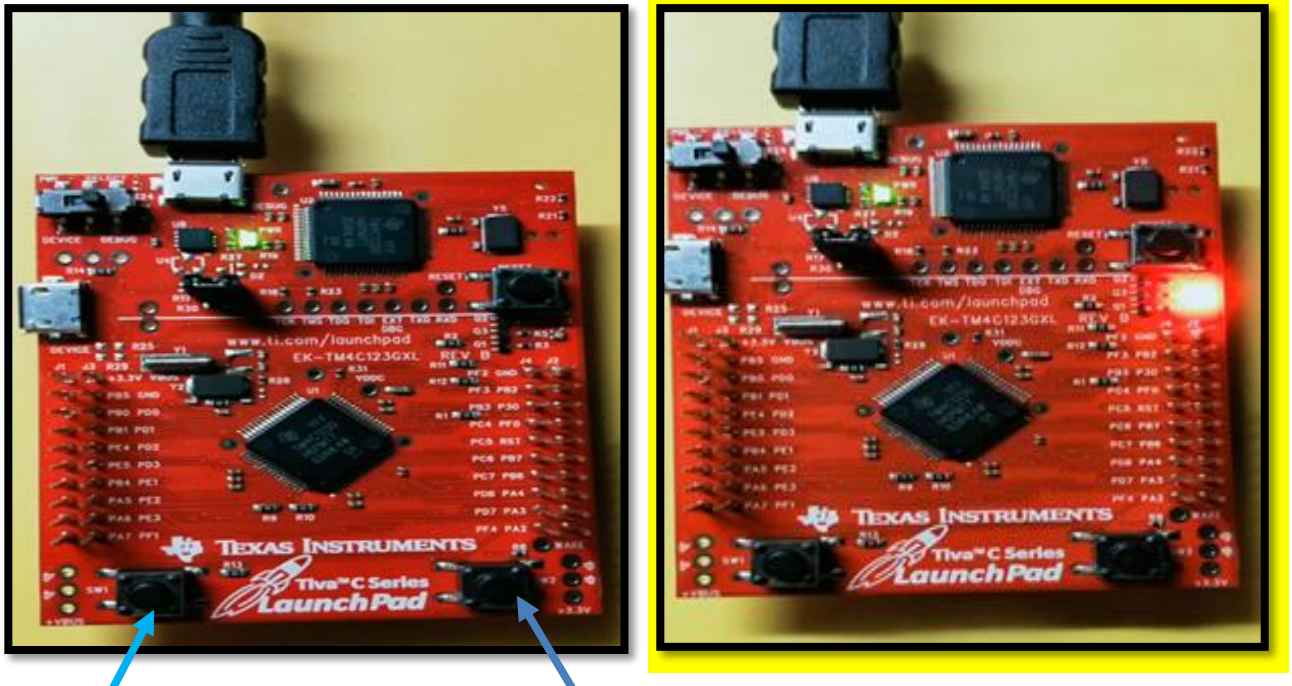
digitalRead(pin)

pin : This specifies the pin number to be read.

PROGRAM

```
void setup()
{
  // put your setup code here, to run once:
  pinMode(RED_LED, OUTPUT);

  pinMode(PUSH1, INPUT_PULLUP);

}

void loop()
{
  // put your main code here, to run repeatedly:
  int SW1 = digitalRead(PUSH1);

  if(SW1 == LOW){
  digitalWrite(RED_LED, HIGH);
  }
  else{
  digitalWrite(RED_LED, LOW);
  }
}
```

OUTPUT



On pressing SWITCH 1 OR SWITCH 2, LED glows

Experiment -2

Write a program to show

i) Analog to Digital Conversion and its output being monitored on the serial monitor

ii) Pulse Width Modulation(PWM) for changing the intensity of the LED

iii) UART for serial communication

using TM4C123 Launchpad and Energia software

i) Analog To Digital Convereter : In this program user is going to learn the usage of ADC in order to convert the analog values to digital values.Here a sensor is interfaced with oneof the ADC channels and the values are being monitored via serial monitor.

Functions used

analogRead()

This function is used to read the value from the specified analog pin. The syntax is as shownanalogRead(pin) - pin must be an analog pin .

analogWrite() :

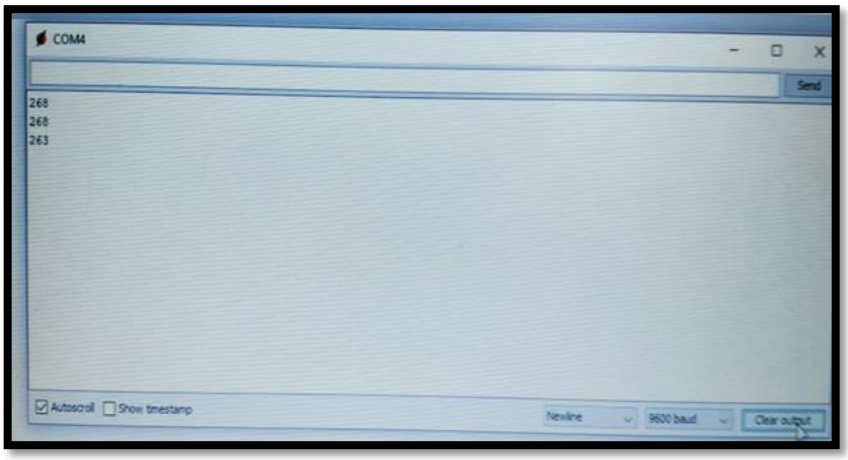
This function is used to write an analog value (PWM wave) to a pin.The syntax is as shownanalogWrite(pin) - pin number must be an analogRead pin

PROGRAM

```
void setup()
{
  // put your setup code here, to run once:
pinMode(A0, INPUT);
Serial.begin(9600);
}
void loop()
{
  // put your main code here, to run repeatedly:
  int a = analogRead(A0);
  Serial.println(a);
  delay(500);
```

}

OUTPUT



ii) PWM Generation : In this program user is going to learn the generation of PWM waveform PWM
stands for pulse width modulation and it is used for various purposes like digital power control,Motor
speed control ,dimming of LED's.PWM wave has duty cycle(%age of time for on period).More the duty
cycle more bright the LED.However in energia 100% duty cycle is represented by 255.

Functions Used

analogWrite(Pin_no,value) This function is used to write an analog value to a pin (i.e. PWMvalue)

-Value is the duty cycle and ranges from 0-255(0 to 100%)

PROGRAM

CASE 1:

```
void setup()
{
 // put your setup code here, to run once:
pinMode(RED_LED, OUTPUT);
}
void loop()
```

```
{
  // put your main code here, to run repeatedly:
  int value;
  for(value = 0; value < 256; value+=10){
  analogWrite(RED_LED, value);
  delay(300);
}
}
```

CASE 2:

```
void setup()
{
  // put your setup code here, to run once:
pinMode(RED_LED, OUTPUT);
}
void loop()
{
  // put your main code here, to run repeatedly:
  int value;
  for(value = 255; value >= 0; value-=10){
  analogWrite(RED_LED, value);
  delay(300);
}
}
```

OUTPUT

CASE 1 **CASE 2**

iii) Serial Communication : In this program user is going to learn the usage of UART for serial communication. UART stands for universal Asynchorous Reciever and Transmitter and using UART ,

we are going to print some values in serial monitor at defined baud rate.

Functions used

Serial.begin()

This function is used to begin the communication .The syntax is as shownSerial.begin(Baud rate)

Baud rate :9600,115200 etc

Serial.available()

This function is used to get the number of bytes (characters) available for reading from theserial port. The syntax is as shown

Serial.read()

This function is used to read the data which is written in the serial monitor.

Serial.print()

This function is used to prints data to the serial port as human-readable ASCII format. Thesyntax is as shown.
Serial.print(variable)

Variable can be integer, char float, double etc.

PROGRAM

```
void setup()
{
  // put your setup code here, to run once:
  Serial.begin(9600);
}
void loop()
{
  // put your main code here, to run repeatedly:
  if(Serial.available()){
  char a = Serial.read();
  Serial.print(a);
  delay(500);
}
}
```

OUTPUT

Experiment - 3

Write a program to connect the Launchpad with Wi-Fi network & print the static IP Addresses on the Serial Monitor

Static IP Address

Objective : In this lab user is going to configure the static IP Address for Launchpad and print that on Serial Monitor**.**

Requirements : TIVA Launchpad,CC3100 Booster pack

Functions Used

WiFi.localIP()

This function returns IP address of the network to which user connected.

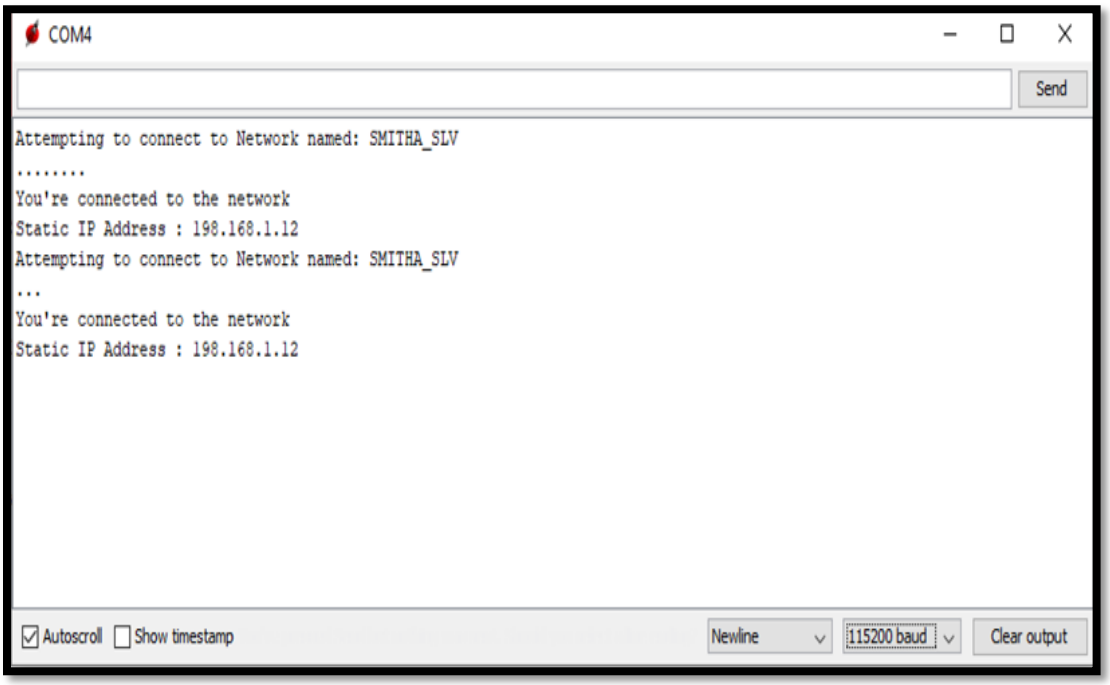
PROGRAM

```
#include <SPI.h>

#include <WiFi.h>

char ssid[] = "SMITHA_SLV";// your network name also called SSID

char password[] = "******";       // your network password


IPAddress ip(198,168,1,12); // Static IP defined
 int keyIndex = 0;                    // your network key Index number (needed only for WEP)

 void setup() {
Serial.begin(115200);      // initialize serial communication
Serial.print("Attempting to connect to Network named: ");  // attempt to connect to Wifi network:

Serial.println(ssid);     // print the network name (SSID);

WiFi.begin(ssid, password);  // Connect to WPA/WPA2 network.

WiFi.config(ip); //Configure Static IP while (

WiFi.status() != WL_CONNECTED) {

Serial.print("."); // print dots while we wait to connect
```

```
    delay(300);
    }
Serial.println("\nYou're connected to the network");
ip = WiFi.localIP();
Serial.print("Static IP Address : ");
Serial.println (ip);  //printing IP ADDRESS
}
void loop()
{
}
```

OUTPUT

The device's Static IP address is printed on the serial monitor.



Experiment - 4

Write a program to connect the Launchpad with Wi-Fi network & print the dynamic IP Addresses on the Serial Monitor

Dynamic IP Address

Objective : In this lab user is going to configure the dynamic IP Address for Launchpad and print that on Serial Monitor**.**

Requirements : TIVA Launchpad,CC3100 Booster pack

Functions Used

- ✓ **Serial.println()**

This function is used to print any data in newline. The syntax is as follows

Serial.println(variable) -Variable may integer ,float,double,char etc

- ✓ **delay()**

This function is used to generate delay of required milli seconds.The syntax is as follows

delay(n) - n must be in ms .Ex if n = 1000 => 1second delay

- ✓ **WiFi.begin()**

This function is used to connect to Wi-Fi by using ssid and password of respective network. The Syntax is as follows

WiFi.begin(ssid, password)

ssid - network username

password - network password

- ✓ **WiFi.status()**

This function is used to return the status of your network connectivity.

WiFi.status()

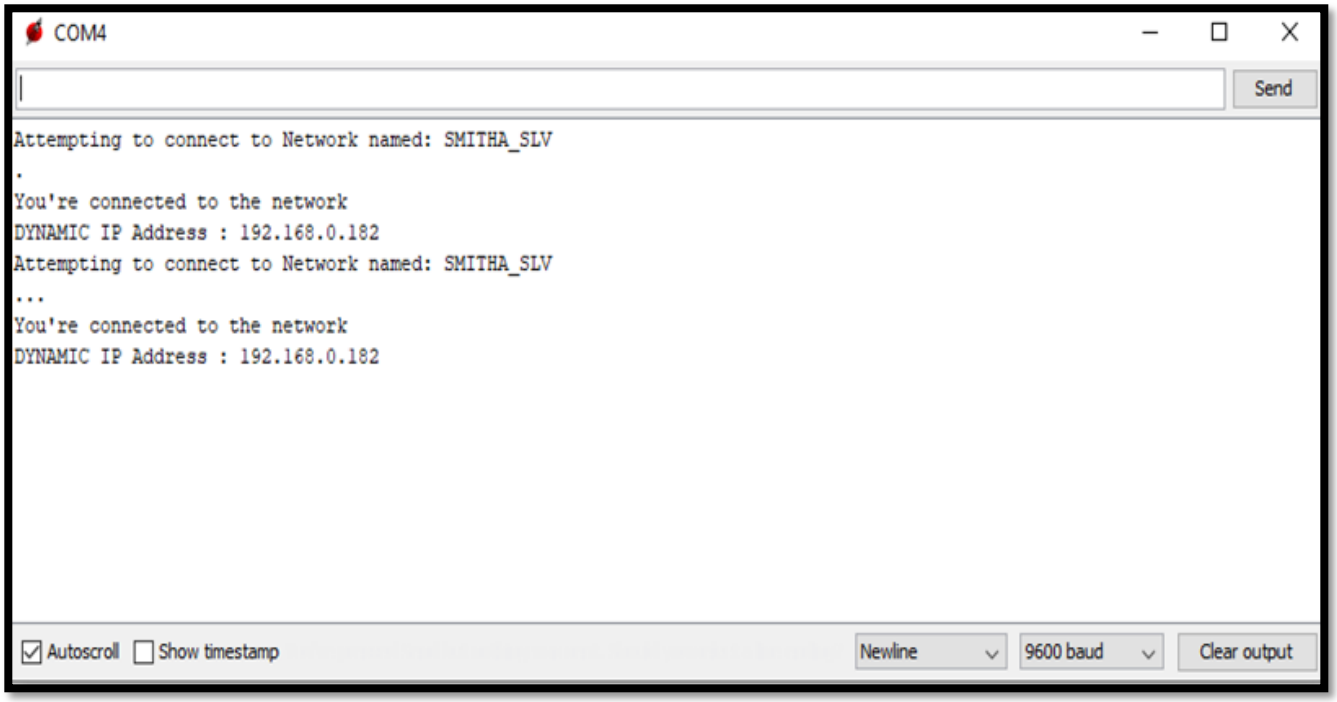
PROGRAM

```
#include <SPI.h>
#include <WiFi.h>

char
ssid[]="SMITHA_SLV";
char password[]="*****";

IPAddress ip;

void setup()

{

 Serial.begin(9600); //Initialize Serial Port

 //attempt to connect to wifi

 Serial.print("Attempting to connect to Network named: ");

 // print the network name (SSID);
 Serial.println(ssid);

 //Connect to WiFI
 WiFi.begin (ssid, password);
 //Wait untill wifi is connected

 while ( WiFi.status() != WL_CONNECTED)

 {

  // print dots while we wait to connect
  Serial.print(".");
  delay(300);

 }


 //If you are connected print the IP Address
 Serial.println("\nYou're connected to the network");
 //wait untill you get an IP address

 while (WiFi.localIP() == INADDR_NONE) {

 // print dots while we wait for an ip addresss
 Serial.print(".");
 delay(300);

 }

 ip=WiFi.localIP();
 Serial.print("Dynamic IP Address : ");
```

```
  Serial.println(ip);
}

void loop()

{
  // put your main code here, to run repeatedly:
  }
```

OUTPUT

The device's Dynamic IP address is printed on the serial monitor.



Experiment -5

Write a program to connect the Launchpad with Wi-Fi & print local IP, Subnet Mask,and Gateway IP on Serial Monitor.

Objective : In this lab user is going to configure the local IP, Subnet mask, gateway IP Address for Launchpad and print that on Serial Monitor**.**

Requirements : TIVA Launchpad,CC3100 Booster pack

Functions Used

WiFi.localIP()

This function returns IP address of the network to which user connected.

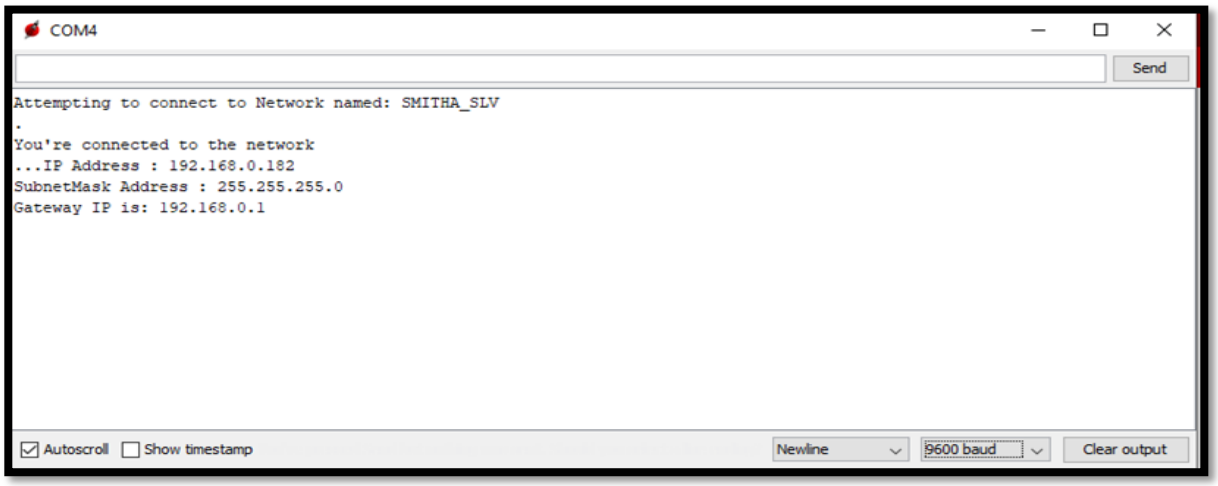
PROGRAM

```
#include <SPI.h>
#include <WiFi.h>

char
ssid[]="SMITHA_SLV";
char password[]="*****";

IPAddress ip,subnet,gateway;

void setup()
{
  Serial.begin(9600); //Initialize Serial Port
  //attempt to connect to wifi
  Serial.print("Attempting to connect to Network named: ");
  // print the network name (SSID);
  Serial.println(ssid);

  //Connect to WiFI
  WiFi.begin(ssid, password);

  //Wait untill wifi is connected
  while ( WiFi.status() != WL_CONNECTED)
  {
    // print dots while we wait to connect
    Serial.print(".");
    delay(300);
  }
```

```
  //If you are connected print the IP Address
  Serial.println("\nYou're connected to the network");

  //wait untill you get an IP address
  while (WiFi.localIP() == INADDR_NONE) {
  // print dots while we wait for an ip addresss
  Serial.print(".");
  delay(300);
  }

  Ip=WiFi.localIP ();
  Serial.print ("IP Address: ");
  Serial.println (ip);
  subnet=WiFi.subnetMask ();
  Serial.print("SubnetMask Address : ");
  Serial.println(subnet);
  gateway=WiFi.gatewayIP();
  Serial.print("Gateway IP is: ");
  Serial.println(gateway);
}

void loop()
{
  // put your main code here, to run repeatedly:

}
```

OUTPUT

The serial monitor will print the local IP, Subnet mask and Gateway IP ofthe device connected.



Experiment -6

Illustrate TCP based Client Server Communication Model.

Objective: This Program will give user a good knowledge about TCP based communication on the bases of client server model.

Requirements : Two TIVA Launch pads, Two CC3100 Booster packs.

Introduction :

The client-server model is a distributed communication framework of network processes among service requestors, clients and service providers. The client-server connection is established through a network or the internet. .
The client-server model is a core network computing concept also building functionality for email exchange and Web/database access. Web technologies and protocols built around the client-server model are:

- Hypertext Transfer Protocol (HTTP)
- Domain Name System (DNS)
- Simple Mail Transfer Protocol (SMTP)
- Telnet

Clients include Web browsers, chat applications, and email software, among others. Servers include Web, database, application, chat and email, etc.

In this program, user will learn how server-client communication uses TCP/IP. Three main things are needed to establish connection between server-client models:

1) Transport protocol (TCP or UDP)

2) Socket

3) IP address and port

Transport protocol (TCP)

TCP is a connection oriented protocol which stands for transfer control protocol. It is a reliable and secure protocol. In TCP, the receiver can generate the acknowledgement of received packet so sender (client) does need to wait for the acknowledgement and if the back response doesn't come or any packet error is generated, it will resend to the client.

Socket

The socket is end point connection for communication between two machines. It is like data connectivity path between two wireless terminals. The socket is required at both sides of server and client.

IP address and port

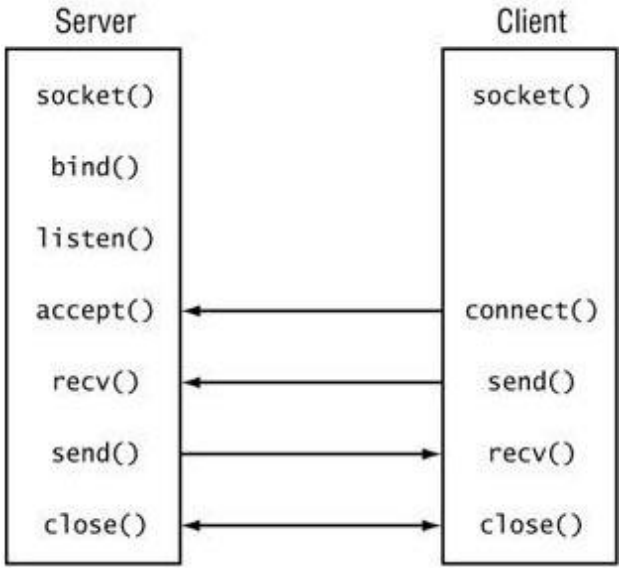
IP address is a unique numerical address of each computer and device. It plays an important role in networking domain with Internet protocol. In server – client model, the server needs to know about which device is connected with it and with whom server establishes the connection. IP address plays an important role in communication between server-client models.

In terms of networking, the port is the location where information is sent. Various protocols assign a different port number. The specific port number is required to communicate between server-client according to which protocol is used.

Server – Client Communication using TCP/IP

Server-client model is communication model for sharing the resource and provides the service to different machines. Server is the main system which provides the resources and different kind of services when client requests to use it.

Let's see, server –client communication through socket programming using TCP/IP. I thought you know the basic C programming and socket. Kindly refer to the tutorial socket and How to create socket in Linux before learning this tutorial.



Server-client communication using TCP/IP

The following tasks are done at client side:

1. Create a socket for communication
2. Configure TCP protocol with IP address of server and port number
3. Connect with server through socket
4. Wait for acknowledgement from server
5. Send message to the server
6. Receive message from server

The following tasks are done at server side:

1. Create a socket for communication
2. Bind the local port and connection address
3. Configure TCP protocol with port number
4. Listen for client connection
5. Accept connection from client
6. Send Acknowledgement
7. Receive message from client
8. Send message to the client

Functions used

WiFiClient.connect()

This function is used to connects to the IP Address and port number .The syntax is as follows

WiFiClient.connect(server ip, server port)

WiFiClient.available()

This function is used to check the amount of data that has been send to the client by server to be read by client. The syntax is as follows

WiFiClient.available()

WiFiClient.read()

This function is used to read the next byte from the server,the client is connected to.The syntax is as follows

WiFiClient.read()

PROGRAM

Server Side

```
#include <SPI.h>

#include<WiFi.>

IPAddress ip(192, 168, 1, 122);

// your network name also called

SSIDchar ssid[] ”SMITHA_SLV";

// your network password char

password[] ="*****";

unsigned char data=0;

  WiFiServer server(9999);

void setup() {
  //Initialize serial and wait for port to open:
  Serial.begin(9600);

  // attempt to connect to Wifi network:

  Serial.print("Attempting to connect to Network named:

  ");

  // print the network name

  (SSID);Serial.println(ssid);

  // Connect to WPA/WPA2 network. Change this line if using open or WEP network.
```

```
WiFi.config(ip);

WiFi.begin(ssid, password);

    while ( WiFi.status() != WL_CONNECTED) {
      // print dots while we wait to

      connectSerial.print(".");

      delay(300);

    }
    Serial.println("\nYou're connected to the

    network");Serial.println("Waiting for an ip

    address");

    while (WiFi.localIP() == INADDR_NONE) {
      // print dots while we wait for an ip

      addresssSerial.print(".");

      delay(300);

    }

    Serial.println("\nIP Address obtained");

    // you're connected now, so print out the

    status:printWifiStatus();

    // start the

    server:

    server.begin();

  }

  void loop()
  {
    // wait for a new client:
    WiFiClient client =

    server.available();if(client==true)

    {

      server.print("HELLO\n");
```

```
    delay(500);

  }
}

void printWifiStatus() {
  // print the SSID of the network you're

  attached to:Serial.print("SSID: ");


  Serial.println(WiFi.SSID());


  // print your WiFi shield's IP address:
  IPAddress IP =

  WiFi.localIP();

  Serial.print("IP Address: ");

  Serial.println(IP);

}
```

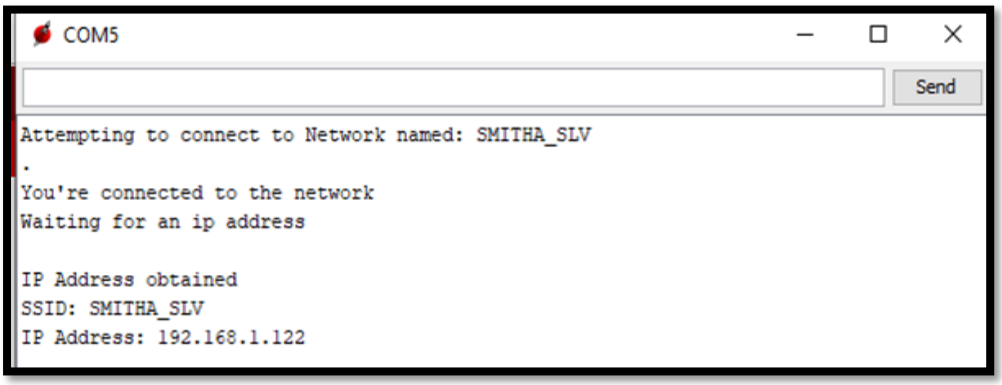
Client Side

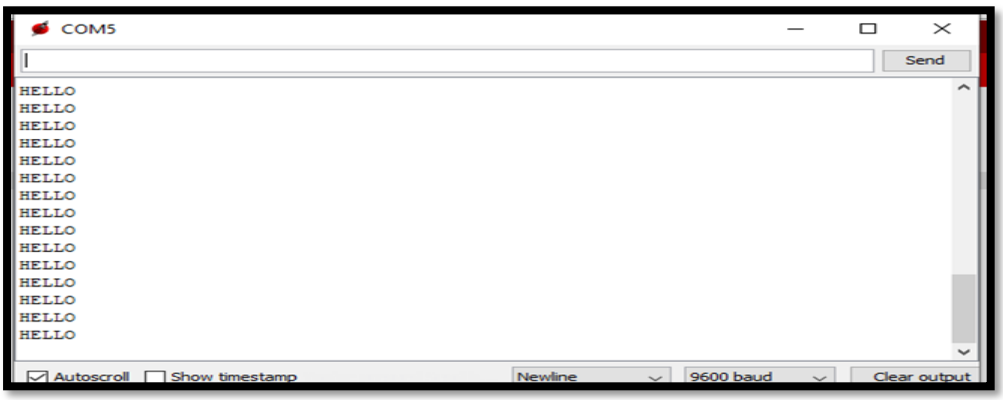
```
#include <SPI.h>
#include <WiFi.h>

char
ssid[]="SMITHA_SLV";
char password[]= "*****";

uint16_t port =9999;  // port number of the server
IPAddress server(192,168,1,122);  // IP Address of the
serverWiFiClient client;

void setup()
{
  Serial.begin(9600);
  WiFi.begin(ssid,
  password);

  while(WiFi.status()!=WL_CONNECTED)
  {
    Serial.println("Waiting for Wi-Fi
     Connection");Serial.print(".");
    delay(300);
  }
```

```
  Serial.println("WiFi Connected");
  Serial.println("Attempting to connect to
  Server");

  while(!client.connect(server,port));
    Serial.println("Connected to
    Server");
}

void loop()
{
  if(client.available())
  {
    char
    c=client.read();
    Serial.print(c);
  }
}
```

OUTPUT

The communication between the server and the client is done by sending and receiving messages from each other using TCP protocol.





Experiment -7

Illustrate UDP based Client Server Communication Model

Objective: In this Program ,user is going to design client server model based on User Datagram communication protocol(UDP).

Requirements : Two TIVA Launchpads, Two CC3100 Booster packs.

Introduction :UDP provides two services not provided by the IP layer. It provides port numbers to help distinguish different user requests.

UDP can also be used in applications that require lossless data transmission when the application is configured to manage the process of retransmitting lost packets and correctly arranging received packets. This approach can help to improve the data transfer rate of large files compared with TCP.

First UDP server creates socket. After creating socket ,it binds IP Address and port number and simultaneously UPD client creates one socket(IP Address ,port).Using the IP Address and port number of server client requests connection establishment. After connecting to the server both can communicate each other.

Functions Used

WiFiUDP.begin()

This function is used to starts WiFIUDP socket and listens the local port.The syntax is as follows

WiFiUDP.begin(port)

WiFiUDP.beginPacket()

This function is used to starts a connection to write UDP data to the remote connection .The syntax is as follows

WiFiUDP.beginPacket(hostIP, port)

WiFiUDP.endPacket()

This function is used to called after writing the UDP data to the remote connections .The syntax is as follows

WiFiUDP.endPacket()

WiFiUDP.write()

This function is used to writes UDP data to the remote connection and must wrapped between beginPacket() and endPacket() .The syntax is as follows

WiFiUDP.write()

WiFiUDP.parsePacket()

This function is used to start processing the next available incoming packet, checks for the Presence of a UDP packet and reports the size. The syntax is as follows

WifiUDP.remoteIP()

This function is used to get the IP Address of the remote connection. The syntax is as follows

WiFiUDP.remoteIP()

WifiUDP.remotePort()

This function is used to get the remote UDP port. The syntax is as follows

WifiUDP.remotePort()

PROGRAM

Server

```
#include <SPI.h>
#include <WiFi.h>

char net[] = "Realme C1";
char pass[] = "donotconnect";
char a[15];
unsigned int LP = 2600;
IPAddress ip(192,168,43,155);
IPAddress ipr(192,168,43,161);

WiFiUDP Udp;
void setup()
{
 Serial.begin(9600);
  WiFi.config(ip);
  WiFi.begin(net,pass);
  while(WiFi.status() != WL_CONNECTED)
  {
  Serial.print(">");
delay(500);
  }
  Serial.print(net);
  Serial.println(":conneted");
  Serial.println("Acquiring IP address");
  while(WiFi.localIP() == INADDR_NONE)
  {
    Serial.print(">");
```

```
    delay(500);
  }
  Serial.print("Acquired IP address is :");
  Serial.println(WiFi.localIP());
 Udp.begin(LP);
}

void loop()
{
//It starts processing the next available
//incoming packet and returns the size in
 //bytes
 int packsize = Udp.parsePacket();
  if(packsize)
  {
    memset(a,0,10);
    Udp.read(a,10);
/*ip address of the
host who sent the current incoming packet*/
    Serial.println(Udp.remoteIP());
/*the port of the host
who sent the current incoming packet*/
    Serial.println(Udp.remotePort());
    Serial.println(a);
 }

 char i =0;
  // put your main code here, to run repeatedly:
  if(Serial.available())
  {
   Serial.readBytes(a,15);
  Udp.beginPacket(ipr,5555);//Starts a connection to write UDP data
  Udp.write(a); //writes UDP data to remote connection
  Udp.endPacket(); //It finishes off the packet and sends it.
  for(i = 0;i<10;i++)
  a[i] = ' ';
  }
  delay(100);
}
```

Client

```
#include <SPI.h>
#include <WiFi.h>

char net[] = "Realme C1";
char pass[] = "donotconnect";
```

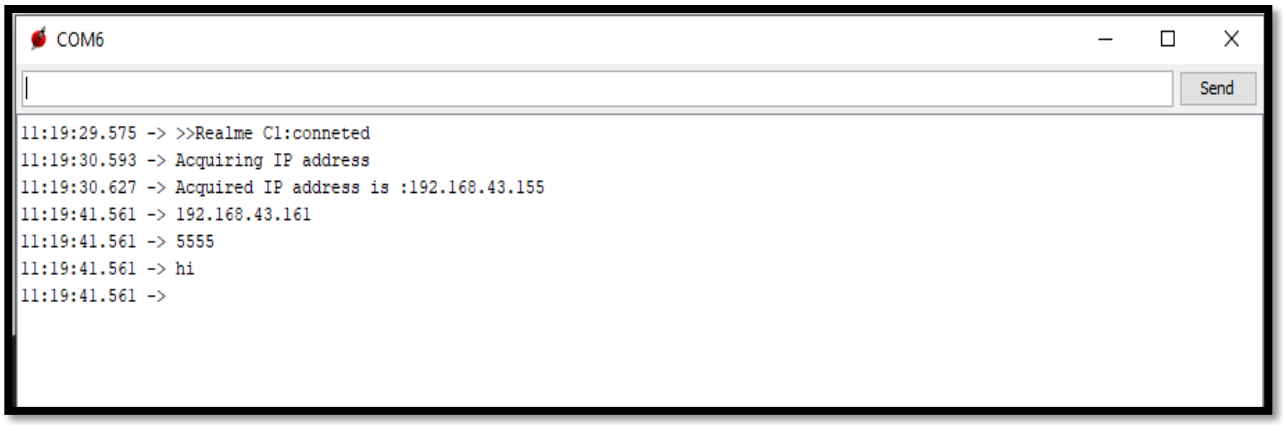
```
char a[15];

unsigned int LP = 5555;
IPAddress ip(192,168,43,155);
WiFiUDP Udp;
void setup()
{
 Serial.begin(9600);
  WiFi.begin(net,pass);
  while(WiFi.status() != WL_CONNECTED)
  {
  Serial.print(">");
delay(500);
  }
  Serial.print(net);
  Serial.println(":conneted");


  Serial.println("Acquiring IP address");
  while(WiFi.localIP() == INADDR_NONE)
  {
    Serial.print(">");
    delay(500);
  }
  Serial.print("Acquired IP address is :");
  Serial.println(WiFi.localIP());
 Udp.begin(LP);

}

void loop()
{
  char i =0;
  // put your main code here, to run repeatedly:
  if(Serial.available())
  {
   Serial.readBytes(a,15);
  Udp.beginPacket(ip,2600);//Starts a connection to write UDP data
  Udp.write(a); //writes UDP data to remote connection
  Udp.endPacket(); //It finishes off the packet and sends it.
  for(i = 0;i<10;i++)
  a[i] = ' ';
  }
  int packsize = Udp.parsePacket();
  if(packsize)
  {
```

```
    memset(a,0,10);
    Udp.read(a,10);
/*ip address of the
host who sent the current incoming packet*/
    Serial.println(Udp.remoteIP());
/*the port of the host
who sent the current incoming packet*/
    Serial.println(Udp.remotePort());
    Serial.println(a);
 }
}
```

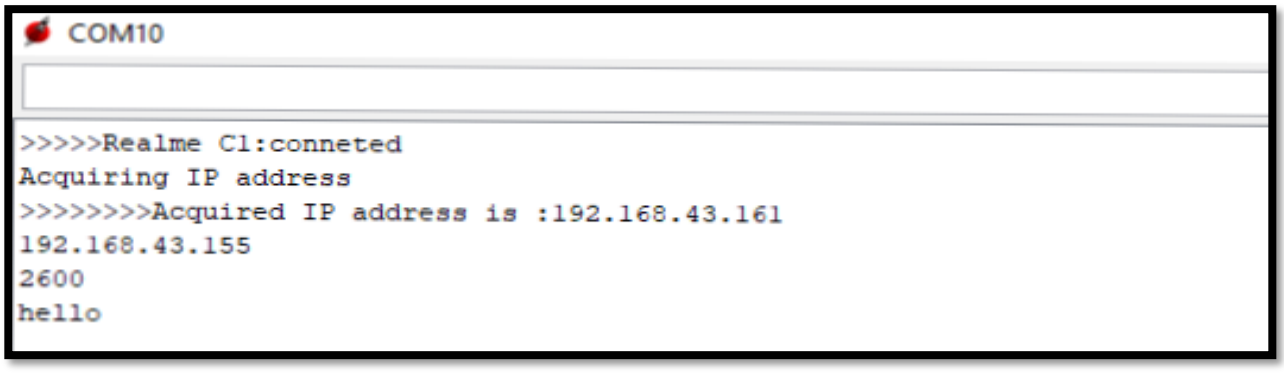
OUTPUT

The communication between the server and the client is done by sending and receiving messages from each other using UDP protocol.

SERVER



CLIENT

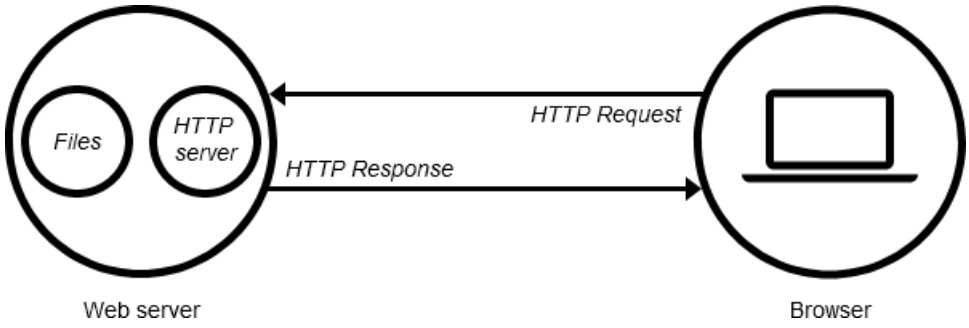


Experiment -8

Write a program for HTTP based webserver to manipulate the GPIO's of WiFi Module CC3100 and blink a LED(HIGH/LOW).

Overview:

A Web server is a program that uses HTTP (Hypertext Transfer Protocol) to serve the files that form Web pages to users, in response to their requests, which are forwarded by their computers' HTTP clients. Dedicated computers and appliances may be referred to as Web servers as well. The process is an example of the client/server model.



At the most basic level, whenever a browser needs a file which is hosted on a web server, the browser requests the file via HTTP. When the request reaches the correct web server (hardware), the *HTTP server* (software) accepts request, finds the requested document (if it doesn't then a 404 response is returned), and sends it back to the browser, also through HTTP.

Requirements : TIVA Launchpad, CC3100 Booster pack.

RED LED:

Procedure :

In this example, a simple web server lets you blink an LED via the web. This example will print the IP address of your WiFi enabled device in Serial Monitor. Once you know the IP address of our board, you can open that address in a web browser to turn on and off the LED on pin 9.

If the IP address of your board is yourAddress:
http://yourAddress/H turns the LED on
http://yourAddress/L turns it off

Our Launchpad TM4C123 and CC3100 communicates using SPI protocol,so SPI header files has to be included in the program ,Also while working with the WiFi libraries ,WiFi header also has to be included. For networks using WPA/WPA2 Personal encryption, you need the SSID and password.

PROGRAM

```
#include <SPI.h>
#include <WiFi.h>
char ssid[] = "SMITHA_SLV";// your network name also called SSID
char password[] = "*****";        // your network password
int keyIndex = 0;                  // your network key Index number (needed only for WEP)
WiFiServer server(80);
        void setup() {
  Serial.begin(115200);      // initialize serial communication
  pinMode(RED_LED, OUTPUT);      // set the LED pin mode
  Serial.print("Attempting to connect to Network named: ");  // attempt to connect to Wifi
network:
   Serial.println(ssid);    // print the network name (SSID);
  WiFi.begin(ssid, password); // Connect to WPA/WPA2 network.
  while ( WiFi.status() != WL_CONNECTED) {
        Serial.print(".");// print dots while we wait to connect
 delay(300);
 }
         Serial.println("\nYou're connected to the network");
  Serial.println("Waiting for an ip address");
   while (WiFi.localIP() == INADDR_NONE) {
   Serial.print(".");       // print dots while we wait for an ip addresss
   delay(300);}
```

```
 Serial.println("\nIP Address obtained");
 printWifiStatus();    // you're connected now, so print out the status
 Serial.println("Starting webserver on port 80");
 server.begin();            // start the web server on port 80
 Serial.println("Webserver started!");
}
void loop() {
 int i = 0;
 WiFiClient client = server.available();  // listen for incoming clients
 if (client) {                     // if you get a client,
 Serial.println("new client");          // print a message out the serial port
 char buffer[150] = {0};               // make a buffer to hold incoming data
 while (client.connected()) {           // loop while the client's connected
  if (client.available()) {            // if there's bytes to read from the client,
   char c = client.read();             // read a byte, then
   Serial.write(c);                   // print it out the serial monitor
    if (c == '\n') {                  // if the byte is a newline character
    // if the current line is blank, you got two newline characters in a row.
    // that's the end of the client HTTP request, so send a response:
    if (strlen(buffer) == 0) {
    // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)
    // and a content-type so the client knows what's coming, then a blank line:
    client.println("HTTP/1.1 200 OK");
    client.println("Content-type:text/html");
    client.println();
            // the content of the HTTP response follows the header:
    client.println("<html><head><title>Energia CC3200 WiFi Web Server</title></head><body
align=center>");
    client.println("<h1 align=center><font color=\"red\">Welcome to  the CC3200 WiFi  Web
Server</font></h1>");
```

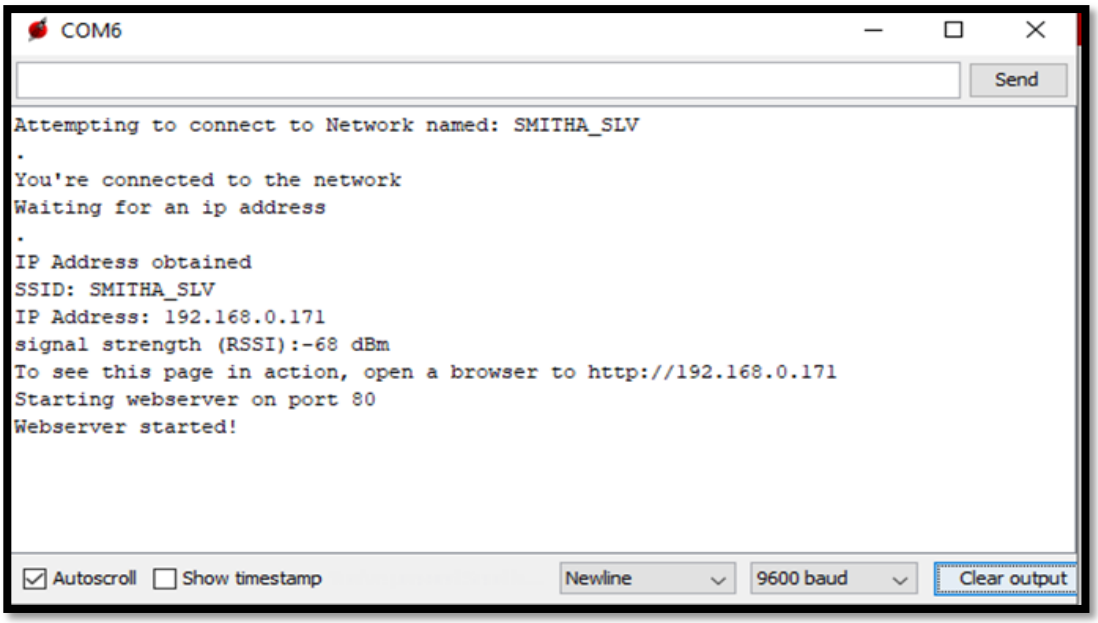
```
            client.print("RED LED <button onclick=\"location.href='/H'\">HIGH</button>");
            client.println(" <button onclick=\"location.href='/L'\">LOW</button><br>");
            client.println();// The HTTP response ends with another blank line:
            break; // break out of the while loop:
            }
           else {      // if you got a newline, then clear the buffer:
            memset(buffer, 0, 150);
            i = 0;
            }
           }
          else if (c != '\r') {    // if you got anything else but a carriage return character,
          buffer[i++] = c;      // add it to the end of the currentLine
          }
         // Check to see if the client request was "GET /H" or "GET /L":
         if (endsWith(buffer, "GET /H")) {
          digitalWrite(RED_LED, HIGH);               // GET /H turns the LED on
         }
         if (endsWith(buffer, "GET /L")) {
         digitalWrite(RED_LED, LOW);                // GET /L turns the LED off
        }
        }
        }
    // close the connection:
    client.stop();
    Serial.println("client disonnected");
           }
}
```

```
//a way to check if one array ends with another array
boolean endsWith(char* inString, char* compString) {
int compLength = strlen(compString);
  int strLength = strlen(inString);
    //compare the last "compLength" values of the inString
  int i;
  for (i = 0; i < compLength; i++) {
   char a = inString[(strLength - 1) - i];
   char b = compString[(compLength - 1) - i];
    if (a != b) {
      return false;
    }
   }
    return true;
   }
 void printWifiStatus() {
  Serial.print("SSID: "); // print the SSID of the network you're attached to:
  Serial.println(WiFi.SSID());
  IPAddress ip = WiFi.localIP();// print your WiFi IP address:
  Serial.print("IP Address: ");
  Serial.println(ip);
  long rssi = WiFi.RSSI(); // print the received signal strength:
  Serial.print("signal strength (RSSI):");
  Serial.print(rssi);
```

```
  Serial.println(" dBm");
 // print where to go in a browser:
  Serial.print("To see this page in action, open a browser to http://");
  Serial.println(ip);
 }
```

This sketch establishes a connection with your WiFi and gets an IP address from the DHCP server available locally. The address is needed to access the simple Web Server and therefore it is printed out on the Serial Monitor as below

OUTPUT



After observing the output in serial monitor ,copy the IPAddress(192.168.0.168) in browser and observe the web Page .

Click on HIGH/LOW button to observe the transition of LED on TIVA board.

Experiment -9

Write a program for sensor based webserver to manipulate the GPIO's of WiFi Module CC3100 and monitor the sensor data connected with it.

Sensor Based Web Server:

Procedure:

A simple web server that shows the value of the analog input pins using a WiFi shield.Aanalog channels 0 to 6 of TM4C123 Launchpad is being monitored using a Server.An Http request is made to access the sensor data from the client/browser.

PROGRAM

```
#include
<SPI.h>
#include
<WiFi.h>

char ssid[] =
"SMITHA_SLV"; char
password[] = "*****";


long sensorValue = 0;

WiFiClient client;

WiFiServer server(80);

void setup()

  {

    Serial.begin(9600);  // initialize serial communication
    pinMode(RED_LED, OUTPUT);          // set the LED
    pin mode Serial.print("Attempting to connect to
    Network named: ");Serial.println(ssid);
    WiFi.begin(ssid, password);

    while ( WiFi.status() != WL_CONNECTED)

    {// print dots while we wait to connectSerial.print(".");
```

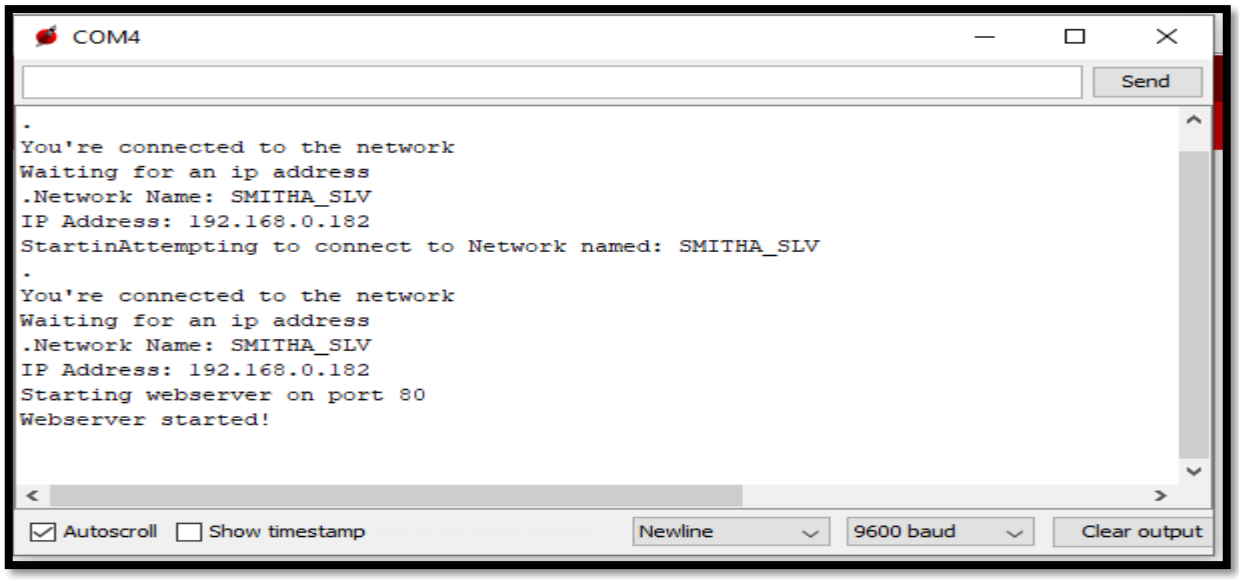
```
 delay(300);

}

Serial.println("\nYou're connected to the
network");Serial.println("Waiting for an ip
address");


while (WiFi.localIP() == INADDR_NONE)

{

 // print dots while we wait for an ip
 addresssSerial.print(".");
 delay(300);

}


PrintWifiStatus();

Serial.println("Starting webserver on port 80");

server.begin();                 // start the web server on port
80Serial.println("Webserver started!");
}


void loop()

{
        client =
        server.available();
        if(client)
    {

                Serial.println("new client");
                boolean
                currentLineBlank=true;
                while(client.connected())

                {

                        if(client.available())

                        {char c=client.read();Serial.write(c);
```

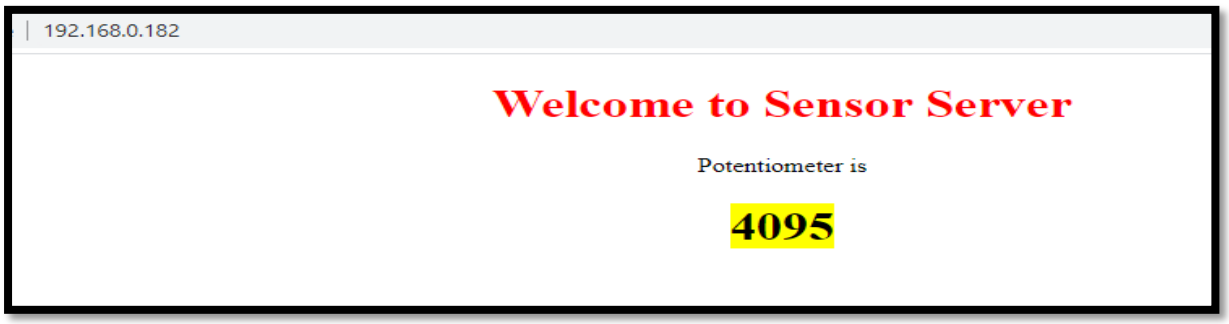
```
                    if(c='\n' && currentLineBlank)

                    {

                        SendResponseServer()
                                ;

                    break;

                    }

                    if(c='\n')

                    {

                        currentLineBlank=tru
                                e;

                    }

                    else if(c!='\r')

                    {

                        currentLineBlank=fal
                                se;

                    }

                }

            }
            delay(1);
            client.stop(
            );
            Serial.println("client disconnected");

        }

}
```

```
void SendResponseServer()

{

        // send a standard http response
   headerclient.println("HTTP/1.1 200
   OK"); client.println("Content-Type:
   text/html");
   client.println("Connection: close"); // the connection will be closed after completion of
theresponse
   client.println("Refresh: 1");  // refresh the page automatically every 2 secs
   client.println();
   client.println("<!DOCTYPE
   HTML>");
   client.println("<html>");
   client.println("<head><title>WiFi Web Server</title></head><body align=center>");

   client.println("<h1 align=center><font color=\"red\">Welcome to
SensorServer</font></h1>");

 // output the value of each analog input pin

     sensorValue =
     analogRead(A0);
     client.print("Potentiometer")
     ; client.print(" is ");
     client.println("<mark>");
     client.println( "<h1>");
     client.println("<mark>");
     client.println(sensorValue);
     client.println("</marked>");
     client.print("</h1>");
     client.println("<br />");
    //  client.println("<meta http-equiv=refresh content=1;URL='//192.168.43.173/'>");
     client.println("</html>");
}
void PrintWifiStatus() {

 // print the SSID of the network you're attached
 to:Serial.print("Network Name: ");
 Serial.println(WiFi.SSID());

 // print your WiFi shield's IP
 address:IPAddress ip =
 WiFi.localIP(); Serial.print("IP
 Address: "); Serial.println(ip);
}
```

OUTPUT

After running the above Program , the Ip address of server will be printed on serial terminal. Copying that Ip address into browser window, a Html page will be displayed with all the analog channel values.





Experiment -10

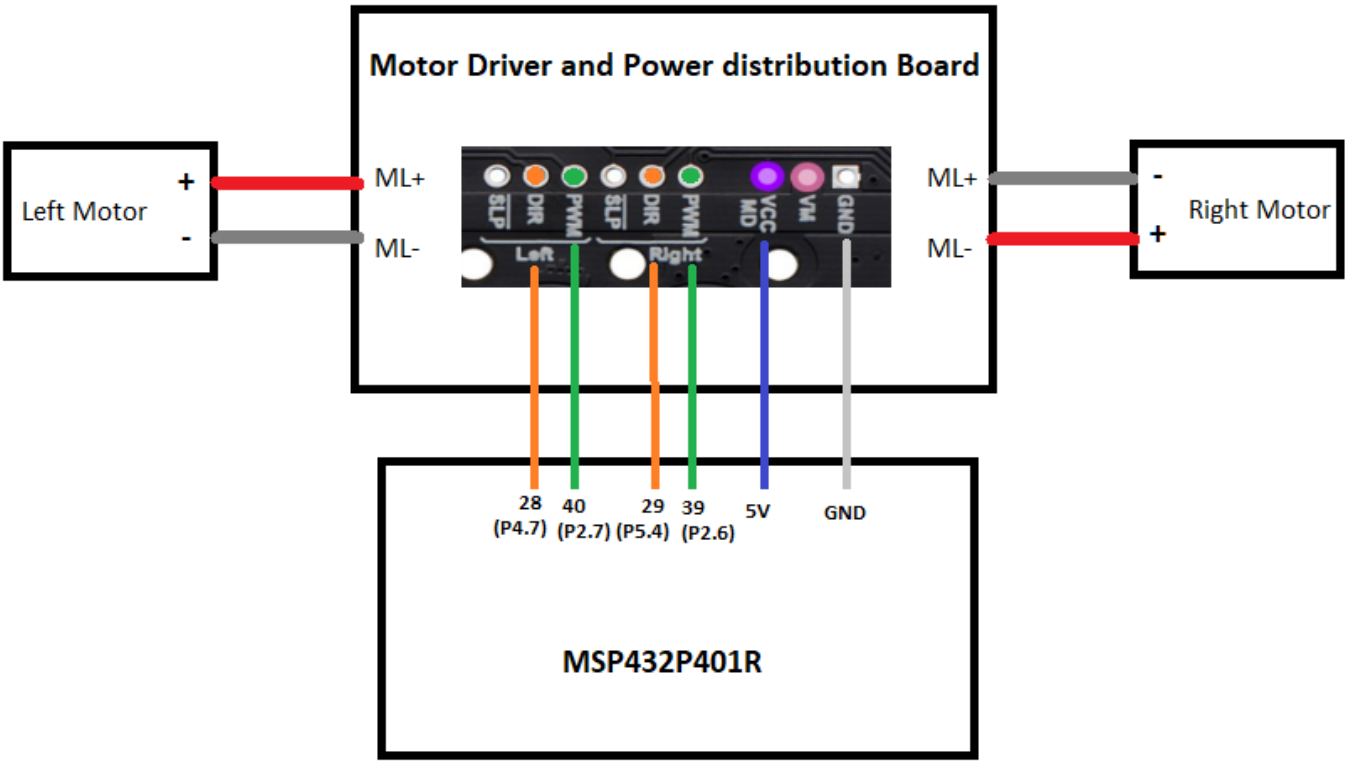
Write a program for HTTP based webserver to control the movement of the RSLK.

ENERGIA INSTALLATION STEPS

1. Go to **energia.nu** website
2. Go to downloads
3. Download Energia IDE and extract it on your PC
4. Follow the link for further reference: https://www.youtube.com/watch?v=S3Q7J8QP0cY

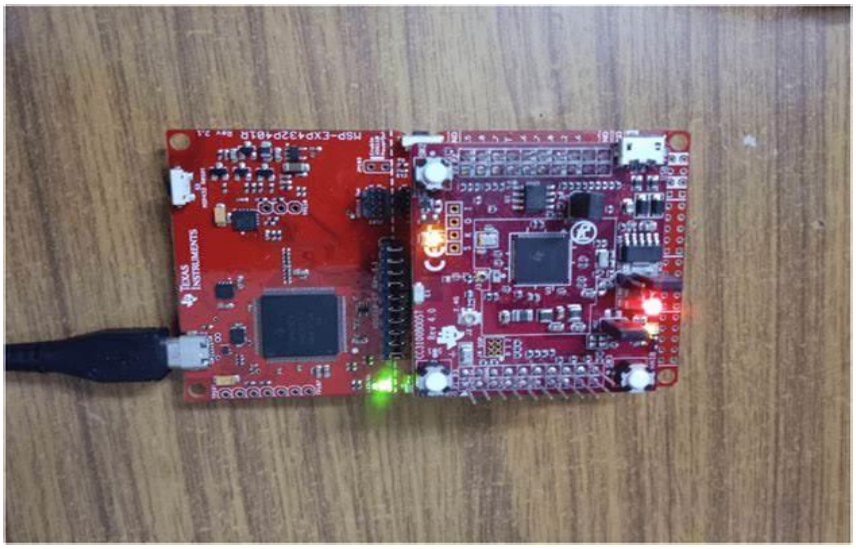
RSLK Connection:

1. Connect a Jumper wire from MSP432 5V pin to Vcc pin of RSLK kit.
2. Connect a Jumper wire from MSP432 Gnd pin to Gnd pin of RSLK kit.
3. Connect pin no P4_7 of MSP432 to left side DIR pin of RLSK Kit.
4. Connect pin no P2_7 of MSP432 to left side PWM pin of RSLK kit.
5. Connect pin no P5_4 of MSP432 to right side DIR pin of RSLK Kit.
6. Connect pin no P2_6 of MSP432 to right side PWM pin of RLSK kit.

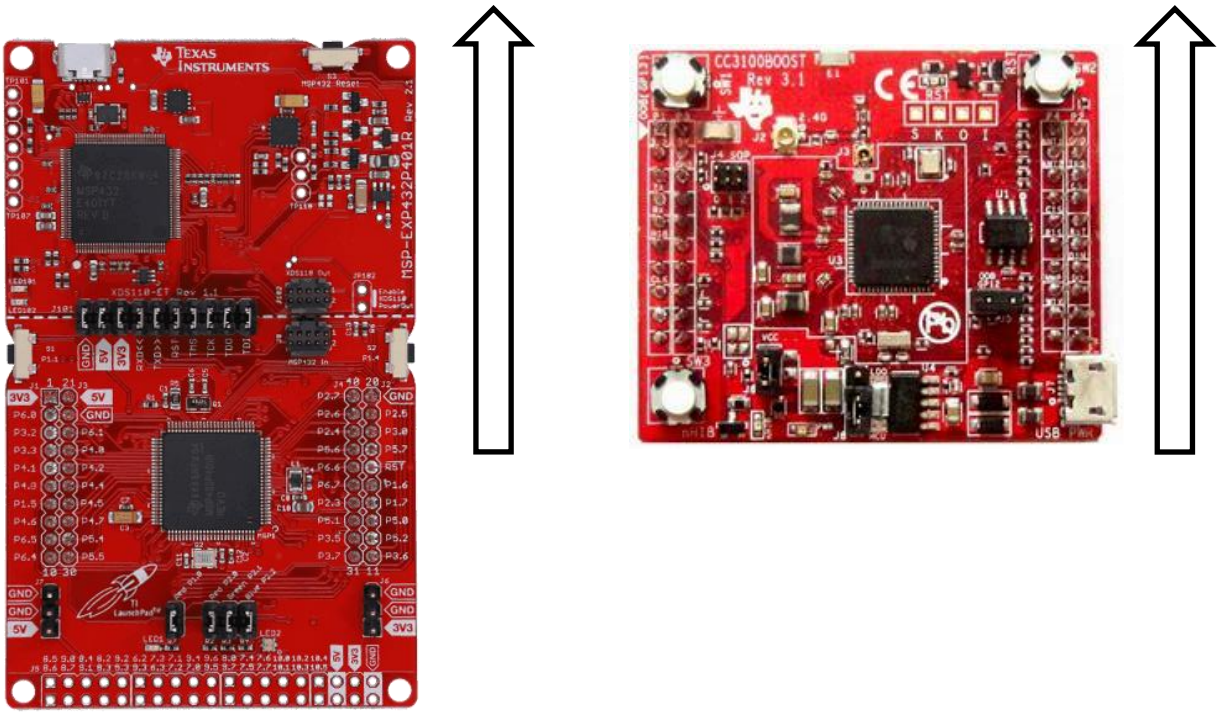


CC3100 WiFi BoosterPack Connection with MSP432 microcontroller

The interfacing of CC3100 WiFi boosterPack's 40 pins on-board which has should be connected serially to the on-board 40 pins of MSP432 micro controller as shown in the below figure.

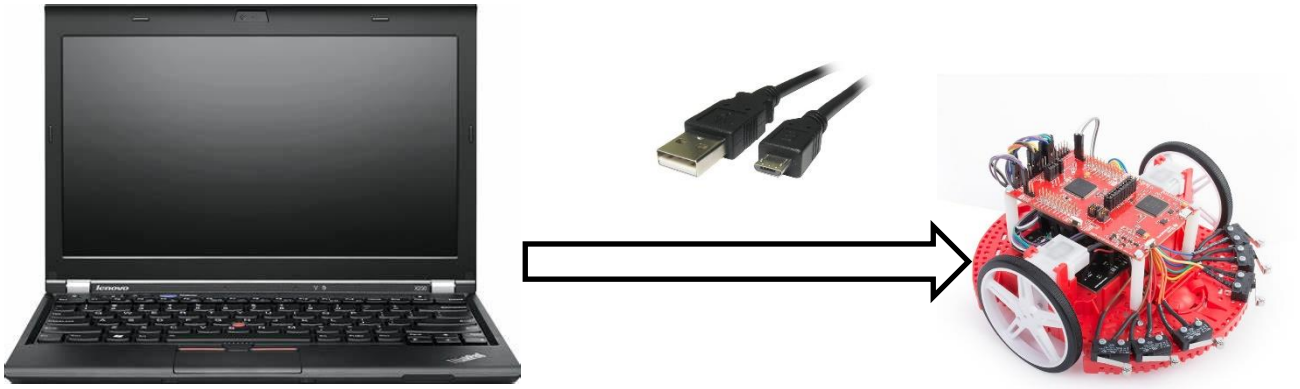


The position of the microcontroller and the boosterPack should be placed as shown below while interfacing them. Connect the boosterPack by lining up the 40-pins by the sides of the boosterPack to the corresponding 40-pins of the microcontroller.



Steps:

1. Keep the microcontroller and CC3100 connected with the RSLK ready.
2. Connect the whole device to the PC using USB.
3. Open Energia software IDE, write the program and upload the code on to the RSLK by selecting the respective COM Port.
4. Make sure the WiFi or the hotspot that the user is using should be same for both the PC and the RSLK kit.
5. Open the serial monitor and copy the IP address that will be printed in the serial monitor after successfully uploading the code on to the RSLK.
6. Now open the internet browser and copy the IP address in the search engine.
7. The virtual buttons that the user have created using HTTP programming will be displayed on the browser.
8. Now remove the USB connection from PC and control the RSLK using HTTP virtual buttons on the webpage.



USB

PROGRAM

```
/*
  WiFi Web Server LED Blink

 A simple web server that lets you blink an LED via the web.
 This sketch will print the IP address of your WiFi (once connected)
 to the Serial monitor. From there, you can open that address in a web browser
 to turn on and off the LED on pin 9.

 If the IP address of your WiFi is yourAddress:
 http://yourAddress/H turns the LED on
 http://yourAddress/L turns it off

 This example is written for a network using WPA encryption. For
 WEP or WPA, change the Wifi.begin() call accordingly.

 Circuit:
 * CC3200 WiFi LaunchPad or CC3100 WiFi BoosterPack
   with TM4C or MSP430 LaunchPad

 created 25 Nov 2012
```

```
 by Tom Igoe
 modified 6 July 2014
 by Noah Luskey
 */
#ifndef __CC3200R1M1RGC__
// Do not include SPI for CC3200 LaunchPad
#define LS 40
#define RS 39
#define LD 29
#define RD 30

#include <SPI.h>
#endif
#include <WiFi.h>

// your network name also called SSID
char ssid[] = "OPPO K1";
// your network password
char password[] = "******";
// your network key Index number (needed only for WEP)
int keyIndex = 0;

WiFiServer server(80);

void setup() {
  Serial.begin(115200);      // initialize serial communication
  pinMode(RED_LED, OUTPUT);      // set the LED pin mode

  // attempt to connect to Wifi network:
  Serial.print("Attempting to connect to Network named: ");
  // print the network name (SSID);
  Serial.println(ssid);
  // Connect to WPA/WPA2 network. Change this line if using open or WEP network:
  WiFi.begin(ssid, password);
  while ( WiFi.status() != WL_CONNECTED) {
    // print dots while we wait to connect
    Serial.print(".");
    delay(300);
  }

  Serial.println("\nYou're connected to the network");
  Serial.println("Waiting for an ip address");

  while (WiFi.localIP() == INADDR_NONE) {
    // print dots while we wait for an ip addresss
    Serial.print(".");
    delay(300);
  }

  Serial.println("\nIP Address obtained");

  // you're connected now, so print out the status
```

```
  printWifiStatus();

  Serial.println("Starting webserver on port 80");
  server.begin();                           // start the web server on port 80
  Serial.println("Webserver started!");

pinMode(LD,OUTPUT);
pinMode(RD,OUTPUT);

}

void loop() {
  int i = 0;
  WiFiClient client = server.available();   // listen for incoming clients

  if (client) {                             // if you get a client,
    Serial.println("new client");           // print a message out the serial port
    char buffer[150] = {0};                 // make a buffer to hold incoming data
    while (client.connected()) {            // loop while the client's connected
      if (client.available()) {             // if there's bytes to read from the client,
        char c = client.read();             // read a byte, then
        Serial.write(c);                    // print it out the serial monitor
        if (c == '\n') {                    // if the byte is a newline character

          // if the current line is blank, you got two newline characters in a row.
          // that's the end of the client HTTP request, so send a response:
          if (strlen(buffer) == 0) {
            // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)
            // and a content-type so the client knows what's coming, then a blank line:
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println();

            // the content of the HTTP response follows the header:
            client.println("<html><head><title>Energia RSLK based WiFi Web Server</title></head><body align=center>");
            client.println("<h1 align=center><font color=\"red\">Welcome to the RSLK WiFi Web Server</font></h1>");
            client.print("ROBOT movement <button onclick=\"location.href='/A'\">FORWARD</button>");
            client.println(" <button onclick=\"location.href='/B'\">BACKWARD</button><br>");
            client.print("Robot Movement <button onclick=\"location.href='/C'\">RIGHT</button>");
            client.println(" <button onclick=\"location.href='/D'\">LEFT</button><br>");
      client.print("Robot Movement <button onclick=\"location.href='/E'\">STOP</button>");
            // The HTTP response ends with another blank line:
            client.println();
            // break out of the while loop:
            break;
          }
          else {      // if you got a newline, then clear the buffer:
            memset(buffer, 0, 150);
            i = 0;
          }
        }
        else if (c != '\r') {    // if you got anything else but a carriage return character,
```

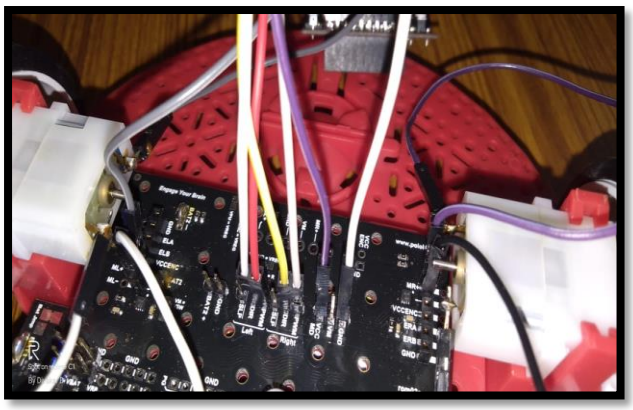
```
          buffer[i++] = c;      // add it to the end of the currentLine
        }

        if (endsWith(buffer, "GET /A")) {
             digitalWrite(LD , LOW);
             digitalWrite(RD ,LOW);
             analogWrite(LS  ,20);
             analogWrite(RS  ,20);
             delay(1000);// GET /H turns the LED on
        }
        if (endsWith(buffer, "GET /B")) {

           digitalWrite(LD , HIGH);
             digitalWrite(RD ,HIGH);
             analogWrite(LS  ,20);
             analogWrite(RS  ,20);
             delay(1000);// GET /L turns the LED off
        }
        if (endsWith(buffer, "GET /C")) {
            digitalWrite(LD , LOW);
            digitalWrite(RD ,LOW);
            analogWrite(LS  ,20);
            analogWrite(RS  ,0);
            delay(1000); // GET /H turns the LED on
        }
        if (endsWith(buffer, "GET /D")) {
              digitalWrite(LD , LOW);
             digitalWrite(RD ,LOW);
            analogWrite(LS  ,0);
            analogWrite(RS  ,20);
             delay(1000);         // GET /L turns the LED off
        }
             if (endsWith(buffer, "GET /E")) {
            analogWrite(LS  ,0);
            analogWrite(RS  ,0);
            delay(1000);         // GET /L turns the LED off
        }
   }
     }
    // close the connection:
    client.stop();
    Serial.println("client disonnected");
   }
}

//
//a way to check if one array ends with another array
//
boolean endsWith(char* inString, const char* compString) {
  int compLength = strlen(compString);
  int strLength = strlen(inString);
```

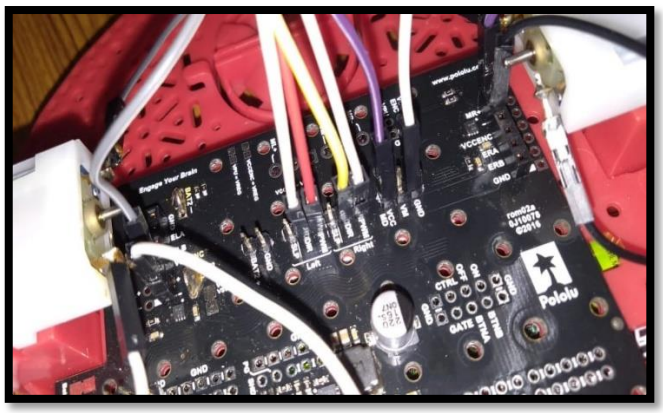
```
  //compare the last "compLength" values of the inString
  int i;
  for (i = 0; i < compLength; i++) {
    char a = inString[(strLength - 1) - i];
    char b = compString[(compLength - 1) - i];
    if (a != b) {
      return false;
    }
  }
  return true;
}

void printWifiStatus() {
  // print the SSID of the network you're attached to:
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());

  // print your WiFi IP address:
  IPAddress ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);

  // print the received signal strength:
  long rssi = WiFi.RSSI();
  Serial.print("signal strength (RSSI):");
  Serial.print(rssi);
  Serial.println(" dBm");
  // print where to go in a browser:
  Serial.print("To see this page in action, open a browser to http://");
  Serial.println(ip);
}
```

OUTPUT

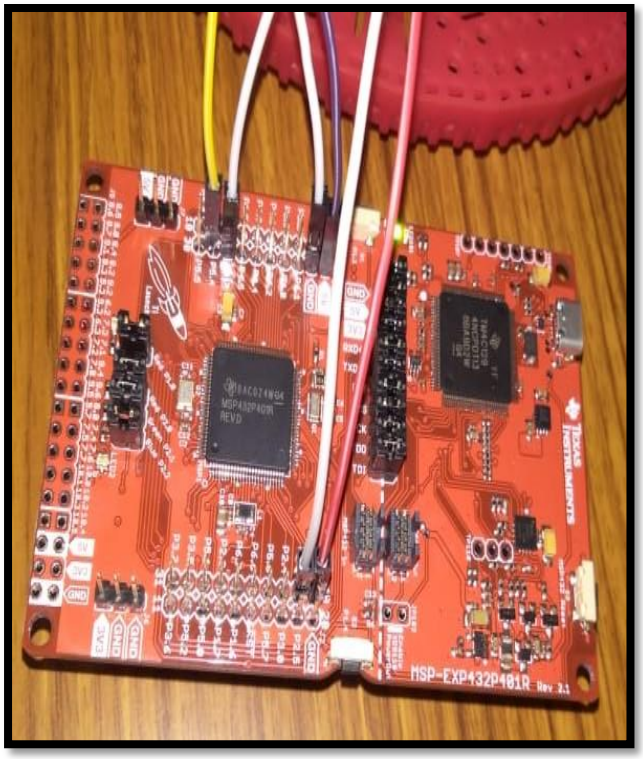
Control RSLK with the virtual buttons on webpage using HTTP protocol

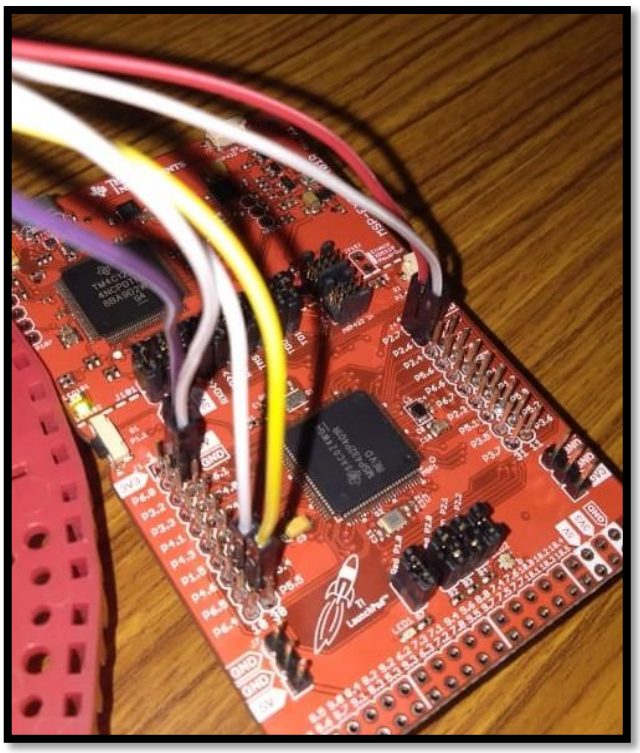
Connections to RSLK

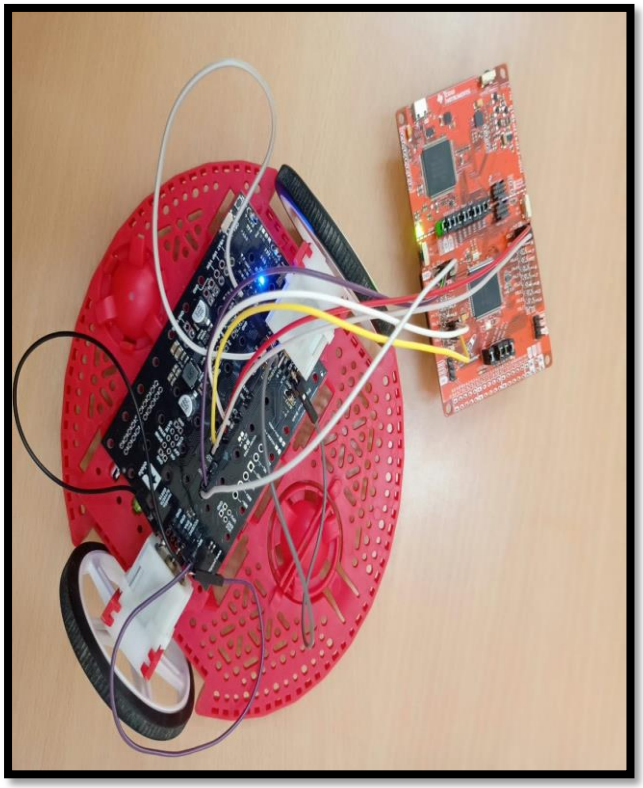


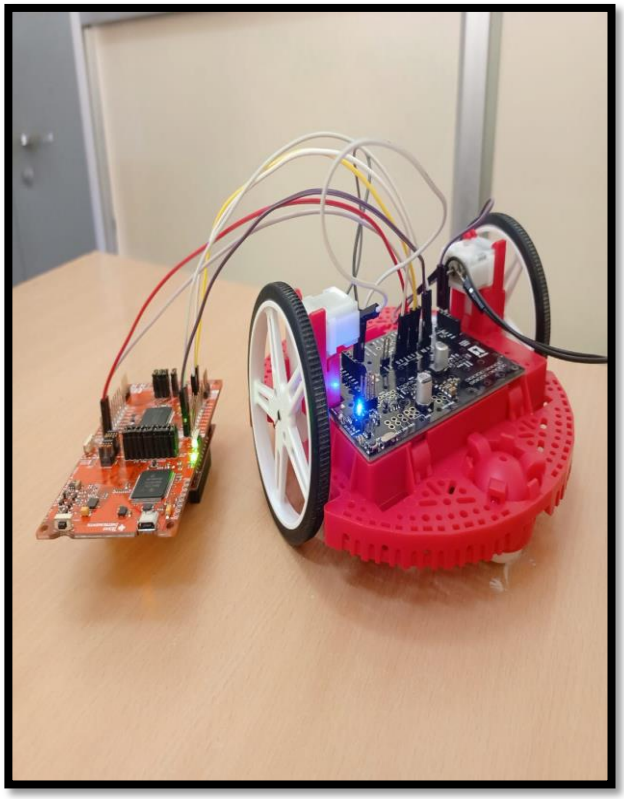


Connections from RSLK to MSP432P401R MCU



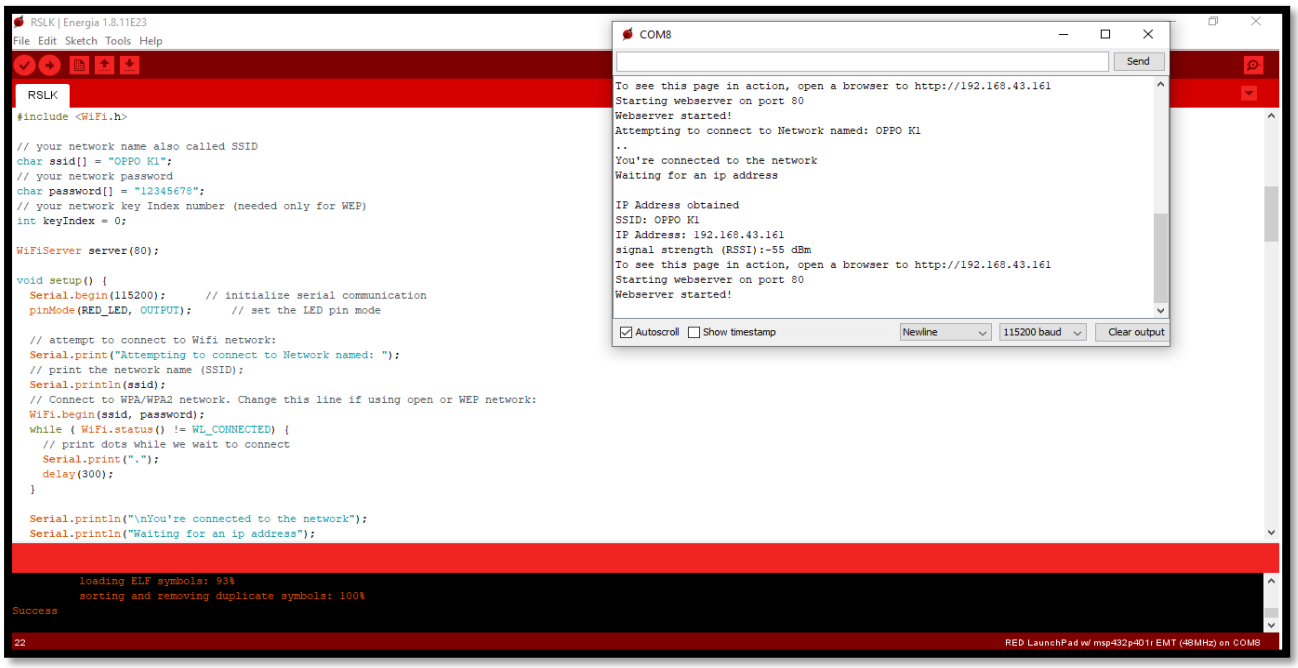






OUTPUT

On the Serial Monitor



Copy the IP address displayed on the serial monitor and paste it on the web browser on any other device(Computer/Phone), acting as a client and check the following

Now , click on the buttons (FORWARD, BACKWARD,RIGHT,LEFT,STOP) to control ROBOT movement.