NANOCHIP SOLUTIONS

Lab Workbook

Completion of the programs in accordance with the given specification

for the course on

IOT PROTOCOLS AND THEIR APPLICATIONS

M. KOUSHIK SAMPATH 9919005250

OBJECTIVE:

Completion of the programs and checking the output on WiFi

LAB DURATION:

Estimated time for completion of lab: 50 min.

BACKGROUND:

Understanding of esp8266 wifi module features and wifi protocol.

DESCRIPTION:

Demonstration of the implementation of Wi-Fi protocol (IEEE 802.11 n) using Wi Fi Module on an Arduino Uno Board.

Logging data to thinkspeak cloud using WiFi module.

Hardware:

Ardunio Uno board

Espressif ESP8266 ESP-01

Description of ESP-01:

ESP8266 module is a microcontroller board with Wi-Fi capability which is designed to operate as a standalone device as well as a slave device when it is connected to a microcontroller (master).

ESP8266 module is of low cost and comes pre-programmed with an AT command set firmware, hence connect the module to Arduino device or any other microcontroller and get WiFi-ability.

This module has a powerful on-board processing and storage capability that allows it to be integrated with the sensors and other application through its GPIOs.

ESP8266 module has its own SoC or system on chip, it utilizes Tensilica L106 32-bit processor clocked at 80MHz.

It has 32KB instruction RAM, 32KB instruction cache RAM, 80KB user data RAM and 16 KBETS system data RAM.

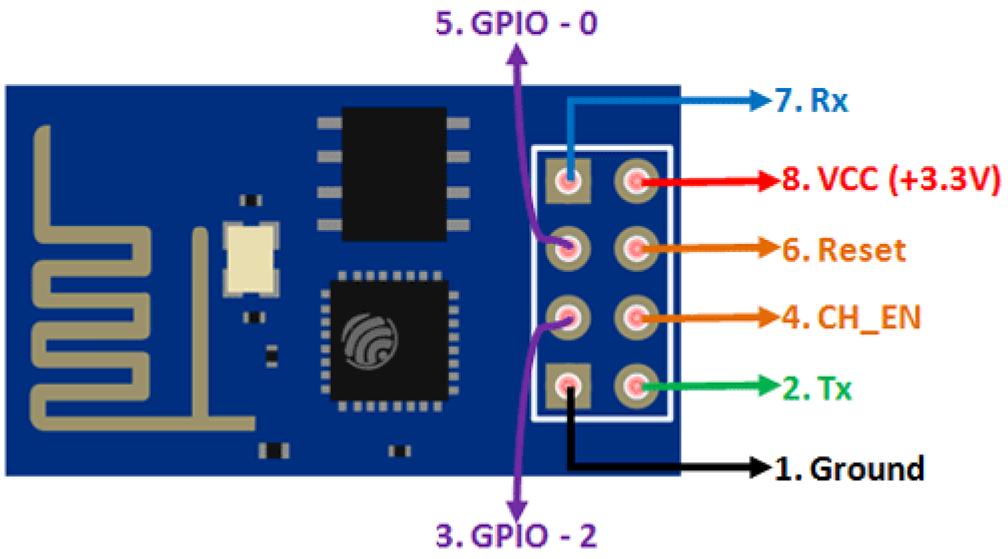
It can operate from 2.5V to 3.6V (nominal 3.3V) and **5V will kill the board**. It has average current consumption about 80mA.

It supports 802.11n (2.4 GHz) and can communicate with a speed of 72.2 Mbps (max).

It supports these network protocols: IPv4, TCP/UDP/HTTP.

ESP8266 WiFi module can be interfaced through UART and with the help of a wide range of AT Commands, the Microcontroller can control the ESP Module.

Pin Diagram:



The pin diagram is also printed on the back of ESP8266 module, which makes our tasks much easier.

ThingSpeak cloud platform:

Create an account on ThingSpeak https://thingspeak.com/

Create a new channel with one field label

Get the API Key

Review the “Update a Channel Feed” Url

Make note of the channel’s API Key to use later in your code

Wiring/Connecting arduino to ESP8266-01 module:

- The wiring here will have the TX of the microcontroller connected to the RX on the module and the RX on the microcontroller talking to the TX on the module
- There’s only one serial on the Arduion Uno but we need it for debugging and also for the module.
- We can’t use one Serial for both.
- We can use code called SoftwareSerial to make a Virtual Serial port on the Arduino
- This means that instead of connecting to the TX/RX we’ll connect to another pin set and have SoftwareSerial send commands to those pins instead.

Solution:

```
#include <SoftwareSerial.h>
#M.KOUSHIK SAMPATH
#9919005250
```

```
#define Rx 2

#define Tx 3

SoftwareSerial esp8266(Rx, Tx);   //Pin 10 and 11 act as RX and TX. Connect them to TX
        and RX of ESP8266


#define DEBUG true

int number;

String AP = "Replace with your network ID";  // Wi-Fi SSID

String PASS = " Replace with your network channel"; // Wi-Fi Password

String API = " Replace with your channel thinkspeak channel ID";   // WRITE API Key

String HOST = "api.thingspeak.com";

String PORT = "80";

String field = "field1";

String response = "";


int countTrueCommand;

int countTimeCommand;

boolean found = false;

int valSensor = 1;


void setup()

{

  Serial.begin(115200);

  Serial.println("Initilizing WiFi");

  esp8266.begin(115200);

  sendCommand("AT",5,"OK");

  sendCommand("AT+CWMODE=1",5,"OK");
  sendCommand("AT+CWJAP=\""+ AP +"\",\""+ PASS +"\"",15,"OK");
  countTrueCommand = 0;
}

  void loop()
  {
        valSensor = random(100);
        String getData = "GET /update?api_key="+ API+"&"+field+"="+String(valSensor);
       sendCommand("AT",5,"OK");
       sendCommand("AT+RST",10,"invalid");
       sendCommand("AT+CIPMUX=1",5,"OK");
       sendCommand("AT+CIPSTART=0,\"TCP\",\""+ HOST +"\","+ PORT,15,"OK");
       sendCommand("AT+CIPSEND=0," +String(getData.length()+4),4,">");
       esp8266.println(getData);delay(1500);countTrueCommand++;
       sendCommand("AT+CIPCLOSE=0",5,"OK");
       Serial.println(valSensor);
       Serial.print(getData);
       Serial.print(",");
       Serial.println(getData.length());
       valSensor = random(100000); //random value, change with sensor value if using sensor
       countTrueCommand = 0;
       delay(10000);



 }
```

```
void sendCommand(String command, int maxTime, char readReplay[]) {
Serial.print(countTrueCommand);
Serial.print(". at command => ");
Serial.print(command);
Serial.print(" ");
while(countTimeCommand < (maxTime*1))
{
esp8266.println(command);//at+cipsend
if(esp8266.find(readReplay))//ok
{
found = true;
break;
}
countTimeCommand++;

}
if(found == true)
{
Serial.println("OK");
countTrueCommand++;
countTimeCommand = 0;
}
if(found == false)
{

Serial.println("Fail");
countTrueCommand = 0;
countTimeCommand = 0;

}

found = false;
}
```

Compile the above sketch in Arduino IDE and flash the hex on to Arduino board.

Set the baud rate of ***Serial Terminal to 115200.***

Check the ESP8266 response on Serial Terminal.

Check that the field1 chart of your channel is getting updated with the latest feed.

LAB INSTRUCTIONS:

1. Create a folder D:/<your name>/IOT_Protocols and save your file to this folder
2. Create files with py extension say temp.py
3. Execute and test output

LEARNING OUTCOMES:

The student will develop the understanding of logic building, writing algorithm and flowchart for a given problem

Understanding of how to initialize WiFi module and testing connectivity to thinkspeak cloud server and testing that the latest feed from the sensor is getting updated.

***

Lab Workbook

Completion of the programs in accordance with the given

specification

for the course on

IOT PROTOCOLS AND THEIR APPLICATIONS

OBJECTIVE:

Understanding of bluetooth

LAB DURATION:

Estimated time for completion of lab: 50 min.

BACKGROUND:

Understanding of Aurdio uno board, bluetooth module

DESCRIPTION:

Interfacing the HC-06 Bluetooth module with Arduino

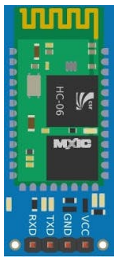
HC-06 Bluetooth module:

HC-06 is a Bluetooth module designed for establishing short range wireless data communication (<100 meters). It is very easy to interface and communicate. It can be interfaced with almost all microcontrollers or processors as it uses UART interface.

This module has the ability to transmit files at speed up to 2.1Mbps and works on Bluetooth 2.0 communication protocol. Unlike the HC-05 Bluetooth module, this module can only act as a slave device.

- Operating Voltage: 3.3V - 6V
- Operating Frequency range: 2.402 GHz - 2.480 GHz

Pinout:



will only be needing the four pins in the HC-06 Bluetooth module.

- RXD: Serial Data Receive Pin. Used for serial input. 3.3V logic
- TXD: Serial Data Transmit Pin. Used for serial output. 3.3V logic
- GND: Ground
- VCC: +5V

Hardware components:

- Arduino Nano - You could use any other development board if you wish
- HC-06 Bluetooth module
- Resistors - 1kΩ and 2kΩ
- Male-to-Male Jumper wires
- Male-to-Female Jumper wires

Software:

- Arduino IDE

Code:

```
 #include <SoftwareSerial.h>
#M. koushik sampath
#9919005250
   SoftwareSerial mySerial(2,3);

   int ledpin=13;

   void setup()
     {
            mySerial.begin(9600);

                  Serial.begin(9600);

                              pinMode(ledpin,OUTPUT);
                               }
                            void loop()
                             {
                               int i;

                         if (mySerial.available())
                              {
                          i=mySerial.read();
                         Serial.println("DATA RECEIVED:");
                          if(i=='1')
                            {
                        digitalWrite(ledpin,1);
                        Serial.println("led on");
                            }
                       if(i=='0')
                         {
                       digitalWrite(ledpin,0);
                       Serial.println("led off");
                         }
                        }
                          }
```

Compile and upload the sketch to your Arduino development board.

Testing the Bluetooth Communication

- After the set up of Bluetooth module, let's test the communication between this module and another device.
- On Android device phone or tablet install Serial Bluetooth terminal
- Make sure you have turned on Bluetooth on your device.
- Open the menu icon on the top right corner of the screen and tap on 'Connect a device - secure'. As we have not paired the Bluetooth module with the device, tap on 'Scan for devices'. You will see the Name of your Bluetooth module. To pair with your module, the device will ask for the password. After you provide the correct password, you will see that the device and the module are connected. The LED on the Bluetooth module will flash every 2 seconds.
- Open the Serial monitor in the Arduino IDE and set the correct baud rate. Type 1 from the Bluetooth terminal app on phone to turn on onboard LED on arduino board and send 0 1 from the Bluetooth terminal app on phone to turn off onboard LED on arduino board.

LAB INSTRUCTIONS:

4. Create a folder D:/<your name>/IOT_Protocols and save your file to this folder
5. Create arduino script, choose the board and port correctly, compile and upload the binary/hex to target board
6. Control the on-board led with the help of app on the android phone
7. Test the output

LEARNING OUTCOMES:

Student will develop the understanding wireless short range communication to control devices remotely

***

Lab Workbook

Completion of the programs in accordance with the given specification

for the course on

IOT PROTOCOLS AND THEIR APPLICATIONS

OBJECTIVE:

Completion of the programs and checking the output

LAB DURATION:

Estimated time for completion of lab: 50 min.

BACKGROUND:

Understanding of what is RFID? How It Works? Interface RC522 RFID Module with Arduino

DESCRIPTION:

Wiring an RC522 RFID Module to an Arduino using available SPI Interface. Developing Arduino scripts for the following

a. Reading an RFID Tag to reader memory.

b. To store RFID Tags.

c. Security Access implementation

```
#include <SPI.h>
#include <MFRC522.h>
//#include <LiquidCrystal.h>
#M.koushik sampath
#9919005250

#define RST_PIN 9
#define SS_PIN 10

byte readCard[4];
String MasterTag = "20C3935E";      // REPLACE this Tag ID with your Tag ID!!!
String tagID = "";

// Create instances
MFRC522 mfrc522(SS_PIN, RST_PIN);
//LiquidCrystal lcd(7, 6, 5, 4, 3, 2); //Parameters: (rs, enable, d4, d5, d6, d7)
//Read new tag if available
boolean getID()
{
  // Getting ready for Reading PICCs
  if ( ! mfrc522.PICC_IsNewCardPresent()) { //If a new PICC placed to RFID reader continue
  return false;
  }
  if ( ! mfrc522.PICC_ReadCardSerial()) { //Since a PICC placed get Serial and continue
  return false;
  }
  tagID = "";
  for ( uint8_t i = 0; i < 4; i++) { // The MIFARE PICCs that we use have 4 byte UID
  //readCard[i] = mfrc522.uid.uidByte[i];
  tagID.concat(String(mfrc522.uid.uidByte[i], HEX)); // Adds the 4 bytes in a single String variable
  }
  tagID.toUpperCase();
  mfrc522.PICC_HaltA(); // Stop reading
  return true;
}
void setup()
{
  // Initiating
    Serial.begin(9600);          // initialize serial communication at 9600
  SPI.begin(); // SPI bus
  mfrc522.PCD_Init(); // MFRC522
  //lcd.begin(16, 2); // LCD screen

  //lcd.clear();
  Serial.println(" Access Control ");
  Serial.println("Scan Your Card>>");
}

void loop()
{

  //Wait until new tag is available
  while (getID())
  {
    //lcd.clear();
    //lcd.setCursor(0, 0);

    if (tagID == MasterTag)
    {
        Serial.println(" Access Granted!");
        // You can write any code here like opening doors, switching on a relay, lighting up an LED, or
anything else you can think of.
      }
      else
      {
        Serial.println(" Access Denied!");
      }



        Serial.println(tagID);


      delay(2000);


    }
    }
```

//Solution B

```
    #include <SPI.h>      //include the SPI bus library
    #include <MFRC522.h>  //include the RFID reader library


    #define SS_PIN 10  //slave select pin
    #define RST_PIN 5  //reset pin


    MFRC522 mfrc522(SS_PIN, RST_PIN);  // instatiate a MFRC522 reader object.
    MFRC522::MIFARE_Key key;          //create a MIFARE_Key struct named 'key', which will hold the card
information


    //this is the block number we will write into and then read.
    int block=2;


    byte blockcontent[16] = {"Embedded System"};  //an array with 16 bytes to be written into one of the 64
card blocks is defined
    //byte blockcontent[16] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};  //all zeros. This can be used to delete a block.


    //This array is used for reading out a block.
    byte readbackblock[18];


    void setup()
    {
     Serial.begin(9600);        // Initialize serial communications with the PC
     SPI.begin();               // Init SPI bus
     mfrc522.PCD_Init();        // Init MFRC522 card (in case you wonder what PCD means: proximity coupling device)
     Serial.println("Scan a MIFARE Classic card");

    // Prepare the security key for the read and write functions.
    for (byte i = 0; i < 6; i++) {
     key.keyByte[i] = 0xFF;  //keyByte is defined in the "MIFARE_Key" 'struct' definition in the .h file of the library
    }
   }

   void loop()
   {
    // Look for new cards
    if ( ! mfrc522.PICC_IsNewCardPresent()) {
     return;
    }

    // Select one of the cards
    if ( ! mfrc522.PICC_ReadCardSerial())
    {
     return;
    }
     Serial.println("card selected");

     //the blockcontent array is written into the card block
     writeBlock(block, blockcontent);

     //read the block back
     readBlock(block, readbackblock);
     //uncomment below line if you want to see the entire 1k memory with the block written into it.
     //mfrc522.PICC_DumpToSerial(&(mfrc522.uid));

     //print the block contents
     Serial.print("read block: ");
     for (int j=0 ; j<16 ; j++)
     {
      Serial.write (readbackblock[j]);
```

```
  }
  Serial.println("");
}




//Write specific block
int writeBlock(int blockNumber, byte arrayAddress[])
{
  //this makes sure that we only write into data blocks. Every 4th block is a trailer block for the
access/security info.
  int largestModulo4Number=blockNumber/4*4;
  int trailerBlock=largestModulo4Number+3;//determine trailer block for the sector
  if (blockNumber > 2 && (blockNumber+1)%4 == 0){Serial.print(blockNumber);Serial.println(" is a trailer
block:");return 2;}
  Serial.print(blockNumber);
  Serial.println(" is a data block:");

  //authentication of the desired block for access
  byte status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, trailerBlock,
&key, &(mfrc522.uid));
  if (status != MFRC522::STATUS_OK) {
      Serial.print("PCD_Authenticate() failed: ");
      Serial.println(mfrc522.GetStatusCodeName(status));
      return 3;//return "3" as error message
  }

  //writing the block
  status = mfrc522.MIFARE_Write(blockNumber, arrayAddress, 16);
  //status = mfrc522.MIFARE_Write(9, value1Block, 16);
  if (status != MFRC522::STATUS_OK) {
        Serial.print("MIFARE_Write() failed: ");
        Serial.println(mfrc522.GetStatusCodeName(status));
        return 4;//return "4" as error message
  }
  Serial.println("block was written");
}




//Read specific block
```

```
int readBlock(int blockNumber, byte arrayAddress[])
{
  int largestModulo4Number=blockNumber/4*4;
  int trailerBlock=largestModulo4Number+3;//determine trailer block for the sector

  //authentication of the desired block for access
  byte status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, trailerBlock, &key, &(mfrc522.uid));

  if (status != MFRC522::STATUS_OK) {
         Serial.print("PCD_Authenticate() failed (read): ");
         Serial.println(mfrc522.GetStatusCodeName(status));
         return 3;//return "3" as error message
  }

//reading a block
byte buffersize = 18;//we need to define a variable with the read buffer size, since the MIFARE_Read method below needs a pointer to the variable that contains the size...
status = mfrc522.MIFARE_Read(blockNumber, arrayAddress, &buffersize);//&buffersize is a pointer to the buffersize variable; MIFARE_Read requires a pointer instead of just a number
  if (status != MFRC522::STATUS_OK) {
          Serial.print("MIFARE_read() failed: ");
          Serial.println(mfrc522.GetStatusCodeName(status));
          return 4;//return "4" as error message
  }
  Serial.println("block was read");
}
```

LAB INSTRUCTIONS:

8. Create a folder D:/<your name>/IOT_Protocols and save your file to this folder
9. Create ardunino scripts
10. Execute and test output

LEARNING OUTCOMES:

The student will develop the understanding of logic building, writing algorithm and flowchart for a given problem

Understanding of how to initialize Bluetooth on raspberry pi, installing blue dot app on android phone and checking the connectivity with the server.

***

Lab Workbook

Completion of the programs in accordance with the given specification

for the course on

IOT PROTOCOLS AND THEIR APPLICATIONS

OBJECTIVE:

Completion of the programs and checking the output on thinksboard platform

LAB DURATION:

Estimated time for completion of lab: 150 min.

BACKGROUND:

Understanding of Thinksboard platform, Python programming, Raspberry pi board

DESCRIPTION:

a) Write a program on Raspberry Pi to publish temperature, humidity, air quality, light data to MQTT broker

Solution:

#Replace thingsBoard IoT Platform Server IP and Access Token

# This Program will send dummy data (Temperature and Humidity datas) to ThingsBoard #IoT Platform at the interval of 5 Seconds.

#M.KOUSHIK SAMPATH

#9919005250

```
import os
import time
import sys
import json
import random
import paho.mqtt.client as mqtt

# Function to read sensor values
def read_from_sensor():
    temp = random.randint(25,45)
    hum = random.randint(50,60)
    air = random.randint(55,60)
    light = random.randint(100,180)
    return temp, hum, air,light
    # Thingsboard platform credentials

THINGSBOARD_HOST = 'YOUR THINGSBOARD HOST IP'
    ACCESS_TOKEN = 'THINGSBOARD DEVICE ACCESS TOKEN'



    INTERVAL = 5
    sensor_data = {'temperature' :0,'humidity':0,'air_quality':0,'light_intensity':0}
    next_reading = time.time()
    client = mqtt.Client()
    client.username_pw_set(ACCESS_TOKEN)
    client.connect(THINGSBOARD_HOST,1883,60)
    client.loop_start()

    try:
        while True:
            temp,hum,air,light = read_from_sensor()
            print("Temperature:",temp, chr(176) + "C")
            print("Humidity:", hum,"%rH")
            print("Air Quality:", air,"%")
            print("Light Intensity:",   light,"lux")
            sensor_data['temperature'] = temp
            sensor_data['humidity'] = hum
            sensor_data['air_quality'] = air
            sensor_data['light_intensity'] = light

            client.publish('v1/devices/me/telemetry',json.dumps(sensor_data),1)
            next_reading += INTERVAL
            sleep_time = next_reading-time.time()
            if sleep_time >0:
                time.sleep(sleep_time)



    except KeyboardInterrupt:
        pass
```

```
client.loop_stop()
client.disconnect()
```

LAB INSTRUCTIONS:

11. Create a folder D:/<your name>/IOT_Protocols and save your file to this folder
12. Create files with py extension say temp.py
13. Execute and test output

LEARNING OUTCOMES:

The student will develop the understanding of logic building, writing algorithm and flowchart for a given problem

Understanding of how to setup thingsboard platform and build the dashboard to print the required sensor data

***

Lab Workbook

Completion of the programs in accordance with the given specification

for the course on

IOT PROTOCOLS AND THEIR APPLICATIONS

OBJECTIVE:

Completion of the programs and checking the output

LAB DURATION:

Estimated time for completion of lab: 150 min.

BACKGROUND:

Understanding of python socket programming, Raspberry pi board

DESCRIPTION:

Write a program to create TCP server on Raspberry Pi and respond with humidity or and temperature data to TCP client when requested.

Solution:
#Script name: TCP_Server.py
#M.KOUSHIK SAMPATH
#9919005250

```
import socket
import Adafruit_DHT as dht
import math
import time
import struct
sensor = dht.DHT11
dht11_pin = 4
server = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
server.setsockopt(socket.SOL_SOCKET,socket.SO_REUSEADDR,1)
#ip=socket.gethostbyname(socket.gethostname())
ip="172.16.11.151"
port=1234
address = (ip,port)
server.bind(address)
server.listen(1)
print("[-] start listening on"+str(ip)+":"+str(port))
client,addr=server.accept()
print("[-] Got a Connection from "+str(addr[0])+":"+str(addr[1]))
try:
    while True:
        data=client.recv(1024)
        data = data.decode()

        print("[-] Received from the client : " +data)
        if(data=="humidity"):
            time.sleep(3)
            humidity, temp = dht.read_retry(sensor, dht11_pin)
            print(humidity)
            print(temp)
            if math.isnan(temp) == False and math.isnan(humidity) == False:
                print("-------------------")
                print("Temparature = %.02f C"%(temp))
                print("-------------------")
                print("Humidity = %.02f%%"%(humidity))
                print("-------------------")
                hstr=str(humidity)
                client.send(b'Humidity value is :'+ hstr.encode())
        elif(data=="temp"):
            time.sleep(3)
            humidity, temp = dht.read_retry(sensor, dht11_pin)
            print(humidity)
            print(temp)
            if math.isnan(temp) == False and math.isnan(humidity) == False:
                print("-------------------")
                print("Temparature = %.02f C"%(temp))
                print("-------------------")
                print("Humidity = %.02f%%"%(humidity))
                print("-------------------")
                tstr=str(temp)
                client.send(b'Temperature value is :'+tstr.encode())
        elif(data=="close"):
            client.send("goodbye".encode())
            client.close()
            server.close()
            break
        else:
            client.send("Invalid data".encode())
            print("invalid data.. Try Again")
except KeyboardInterrupt:
    client.close()
    server.close()
    #socket.close()
```

#Script name – TCP_Client.py

```
import socket
client =socket.socket()
client.connect(("172.16.11.151",1234))
def communicate(data):
    client.send(data)
    return
while 1:
    response = raw_input("enter the command:")
    #print(inputcmd1)
    communicate(response)
    reply = client.recv(1024)
    print(reply.decode())
    print("----------------------")
    if(reply=="goodbye"):
        client.close()
        break
```

LAB INSTRUCTIONS:

14. Create a folder D:/<your name>/IOT_Protocols and save your file to this folder
15. Create files with py extension say temp.py
16. Setup the hardware (Raspberry pi Connect the temperature sensor to the correct pins)
17. First run server (Ex: python TCP_Server.py)
18. Then execute the client (Ex: python TCP_Client.py)
19. Test output

LEARNING OUTCOMES:

The student will develop the understanding of logic building, writing algorithm and flowchart for a given problem

Understanding of how client and server are communicating with TCP

***