

IOT AND CLOUD COMPUTING LAB

By,

Asst. Prof. S.Asra

M.E(Computer science & Engineering)

B.E(Computer Science & Engineering)

**Diploma(Computer Science &
Engineering)**



IOT AND CLOUD COMPUTING LAB

Course	B.Tech.-VI-Sem.	L	T	P	C
Course Code	22CSPC64	-	-	2	1

Course Outcomes (COs) & CO-PO Mapping (3-Strong; 2-Medium; 1-Weak Correlation)

COs	Upon completion of course the students will be able to	PO4	PO5	PO9	PSO2
CO1	identify various IoT devices	3	3	3	3
CO2	use IoT devices in various applications	3	3	3	3
CO3	develop automation work-flow in IoT enabled cloud environment	3	3	3	3
CO4	take part in practicing and monitoring remotely	3	3	3	3
CO5	make use of various IoT protocols in cloud	3	3	3	3



List of Experiments

Week	Title/Experiment
1	Install necessary software for Arduino and Raspberry Pi.
2	Familiarization with Arduino and Raspberry Pi board.
3	Write a program to transfer sensor data to a Smartphone using Bluetooth on Arduino.
4	Write a program to implement RFID using Arduino.
5	Write a Program to monitor temperature and humidity using Arduino and Raspberry Pi.
6	Write a Program to interface IR sensors with Arduino using IoT Cloud Application.
7	Write a Program to upload temperature and humidity data to the cloud using an Arduino or Raspberry Pi.
8	Write a program to retrieve temperature and humidity data from the cloud using Arduino and Raspberry Pi.
9	Write a program to create a TCP server on cloud using Arduino and respond with humidity data to the TCP client when requested.
10	Write a program to create a UDP server on cloud using Arduino and respond with humidity data to the UDP client when requested.

References

1. IoT and Cloud Computing Lab Manual, Department of CSE, CMRIT, Hyd.

Micro-Projects: Student should submit a report on one of the following/any other micro-project(s) approved by the lab faculty before commencement of lab internal examination.

1. Air Pollution Meter.
2. Smart Garbage Collector.
3. Weather monitoring system.
4. Baggage Tracker.
5. Circuit Breakage Detection.
6. Anti-Theft Flooring System.
7. IoT Based Smart Street Light.
8. IoT based Gas Leakage Monitoring system.
9. IoT Based Smart Irrigation System.
10. IoT Based Water Level Monitoring System.




01

Week-10




AIM: Write a program to create UDP Server on cloud using Arduino and Respond with humidity data to UDP Client when requested.





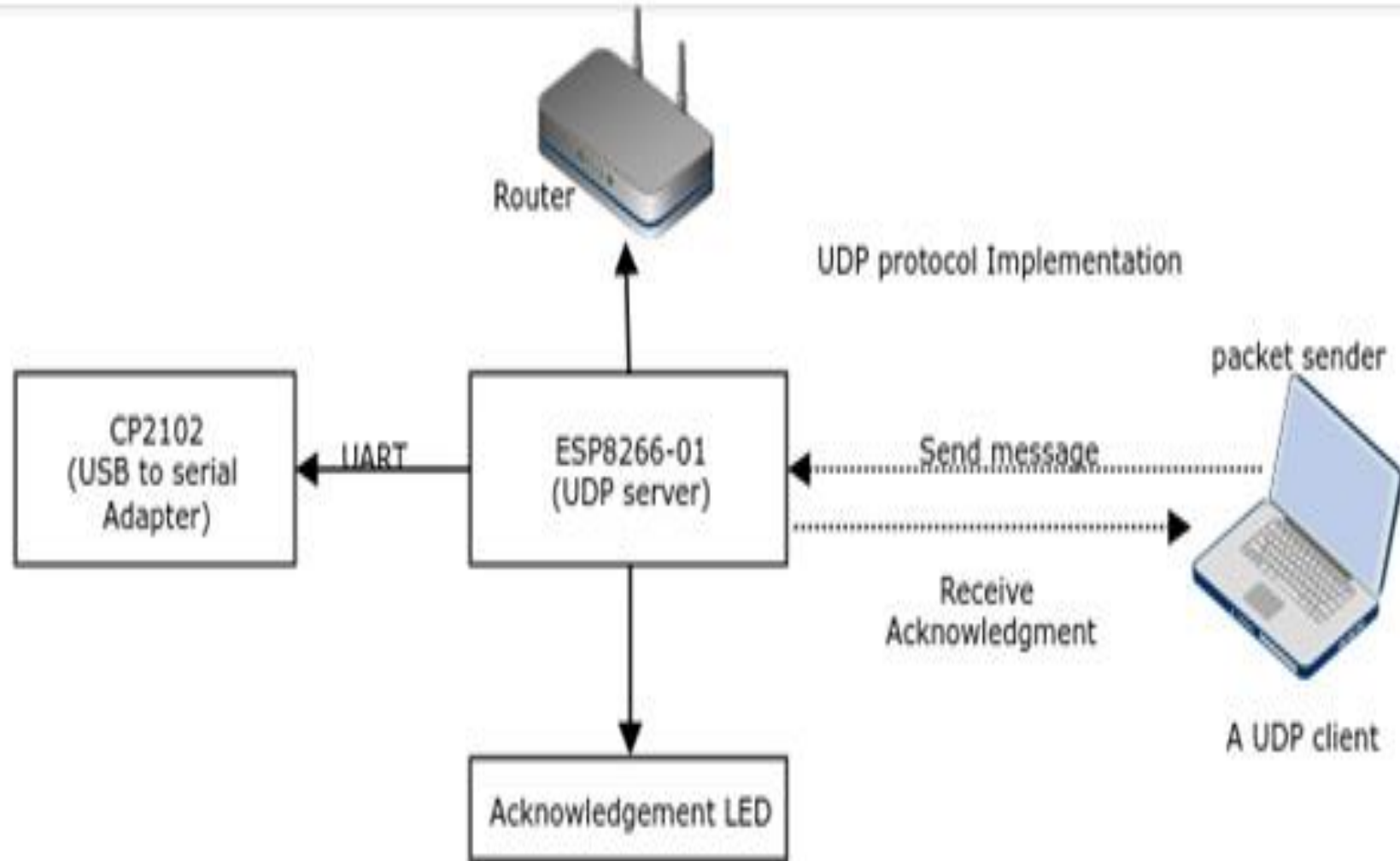
The UDP protocol has a small overhead of 8 bytes which makes it more suitable for use in the Internet of Things. In this project, the application of UDP protocol in IoT will be demonstrated. In this project, an ESP8266 Wi-Fi modem will be configured as UDP server and a laptop will be used as UDP Client. Both Client and server will be co-located communicating through same Wi-Fi router so, the ESP board will act as a local server. The ESP module working as server checks for the UDP packet received from the client on a particular port. When a valid packet is arrived, an acknowledge packet is sent back to the client to the same port it has been sent out. In response to receiving packet and sending acknowledgement, the ESP modem switches on an LED as visual indication of successful Client-Server Communication. This application is very useful as it demonstrates server/client communication over UDP protocol.

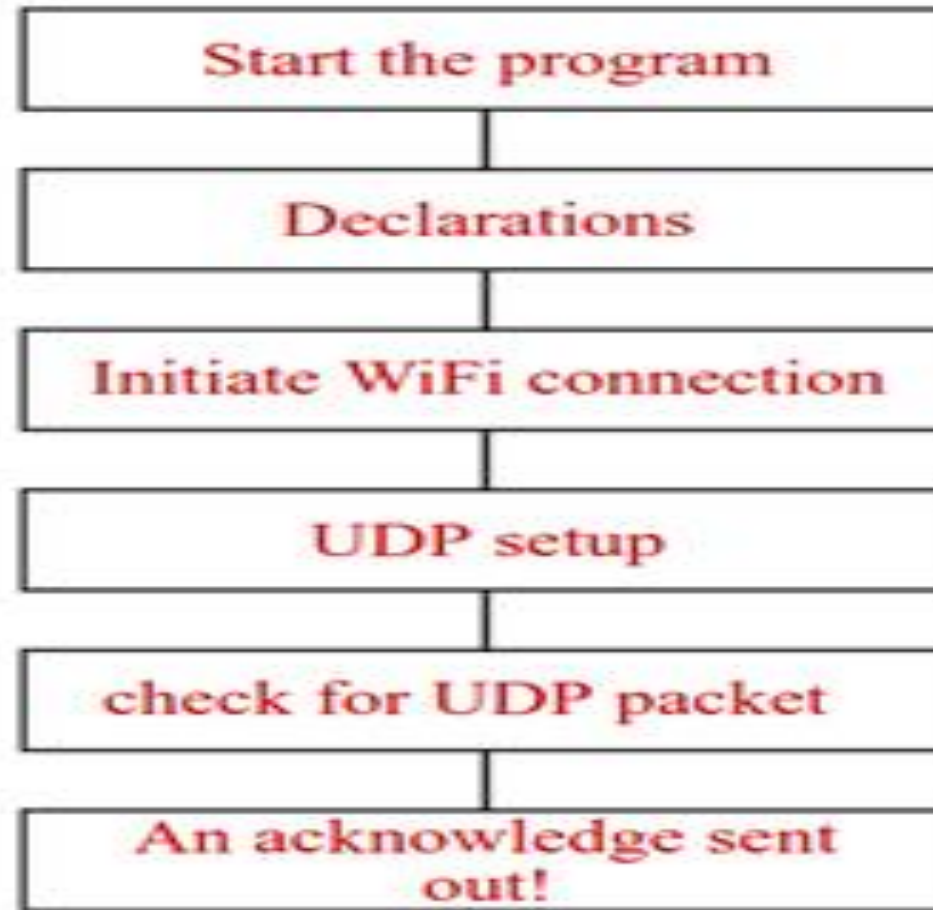




It can be said that the IOT device designed in this experiment is a simple UDP server with LED indicator. It is designed by interfacing an LED with the ESP-8266 Wi-Fi module. The Wi-Fi module as well as LED light are powered continuously with the help of a USB to Serial Converter. The Wi-Fi module needs to be loaded with a firmware that could receive data over UDP protocol and respond with an acknowledgement. The Arduino UNO is used to flash the firmware code on the ESP8266 module. The ESP module can also be flashed with code using a FTDI converter like CP2102. The firmware itself is written in the Arduino IDE.










Software Required –

- ThingSpeak server
- Arduino IDE

ESP8266 board needs to be loaded with the firmware code. In this tutorial, the firmware code is written using Arduino IDE. It is loaded to the ESP8266 board using the Arduino UNO. A generic ESP8266 board is used in this project. This board does not have any bootstrapping resistors, no voltage regulator, no reset circuit and no USB-serial adapter. The ESP8266 module operates on 3.3 V power supply with current greater than or equal to 250 mA. So, CP2102 USB to serial adapter is used to provide 3.3 V voltage with enough current to run ESP8266 reliably in every situation.






The ESP8266 Wi-Fi Module is a self contained SOC with integrated TCP/IP protocol stack that can access to a Wi-Fi network. The ESP8266 is capable of either hosting an application or off loading all Wi-Fi networking functions from another application processor. Each ESP8266 module comes pre-programmed with an AT command set firmware.

The Chip Enable and VCC pins of the module are connected to the 3.3 V DC while Ground pin is connected to the common ground. The chip enable pin is connected to VCC via a 10K pull up resistor. The RESET pin is connected to the ground via a tactile switch where the pin is supplied VCC through a 10K pull up resistor by default. The Tx and Rx pins of the module are connected to the RX and TX pins of the Arduino UNO. The GPIO-0 pin of the module is connected to ground via a tactile switch where the pin is supplied VCC through a 10K pull up resistor by default.

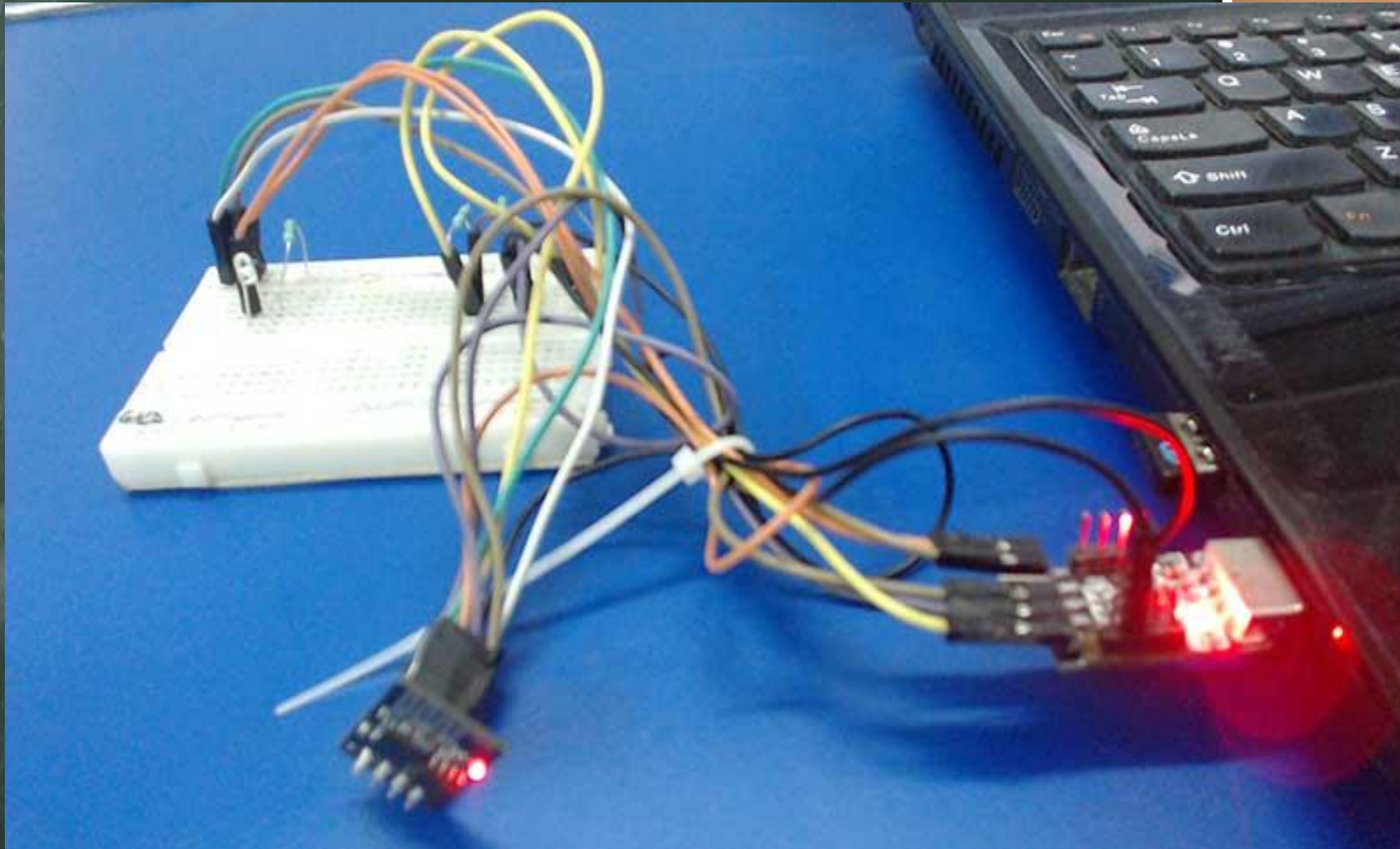




Write the firmware code in the Arduino IDE and connect the Arduino board with the PC via USB cable. Open Arduino IDE and go to Tools->Port and select the Arduino board (Arduino UNO). It may look like /dev/ttyABM0 (Arduino/Genuino Uno). Select the correct port name. The port name can be different in different IDE setups.

Then Open Serial monitor in the Arduino IDE by navigating to Tools->Serial Monitor and set the baud rate to 115200 bauds per second. Pass 'AT' and 'AT+GMR' commands to test the connection between the Arduino and ESP-01 module. Try different settings for the 'Line ending' option of the serial monitor like Both NL & CR. Try different combinations until the ESP module starts interacting correctly with the serial monitor.





st. Prof. S.Asra
CSE Department, CMRIT

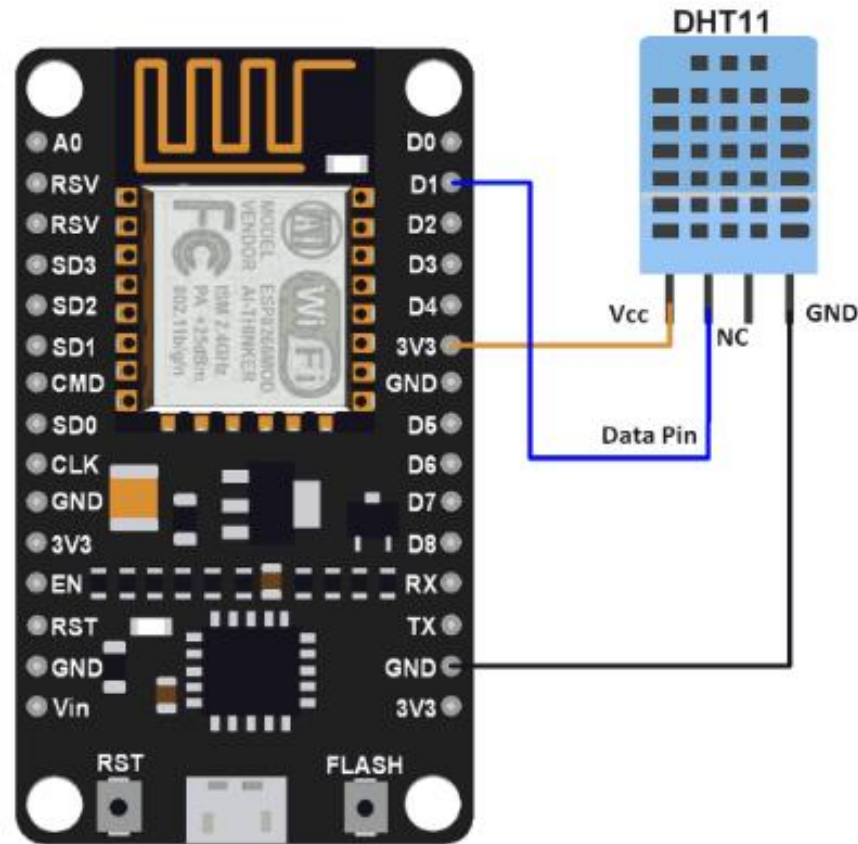


DHT-11 Sensor – DHT-11 is a temperature and humidity sensor. The DHT11 sensor consists of two main components – one is Humidity sensing component and other is NTC temperature sensor (or Thermistor).

The Thermistor is actually a variable resistor that changes its resistance with change in temperature. They both sense the temperature and humidity of area and give the output to the IC (which is placed on back side of sensor). The sensor has four pins – VCC, Ground, data Out and NC. The VCC and Ground pins are connected to the common VCC and Ground respectively. The Data Out pin of the sensor is connected to PD7 pin of the Arduino board via 10K pull-up resistor.



Connection Diagram DHT11 with NodeMCU



NodeMCU interfaced with DHT11





```
8  #define DHTPIN D3
9  #define DHTTYPE DHT11
10 DHT dht(DHTPIN, DHTTYPE);
11 WiFiUDP udp;
12 void setup() {
13     Serial.begin(115200);
14     Serial.println();
15     Serial.println("Connecting to WiFi...");
16     WiFi.begin(ssid, password);
17     while (WiFi.status() != WL_CONNECTED) {
```

Output Serial Monitor X

Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM15')

```
-----
Temperature: 30.60 °C Humidity: 59.00 %
Sending data over UDP...
Data sent over UDP.
Failed to read from DHT sensor!
Temperature: 31.00 °C Humidity: 57.00 %
Sending data over UDP...
Data sent over UDP.
```

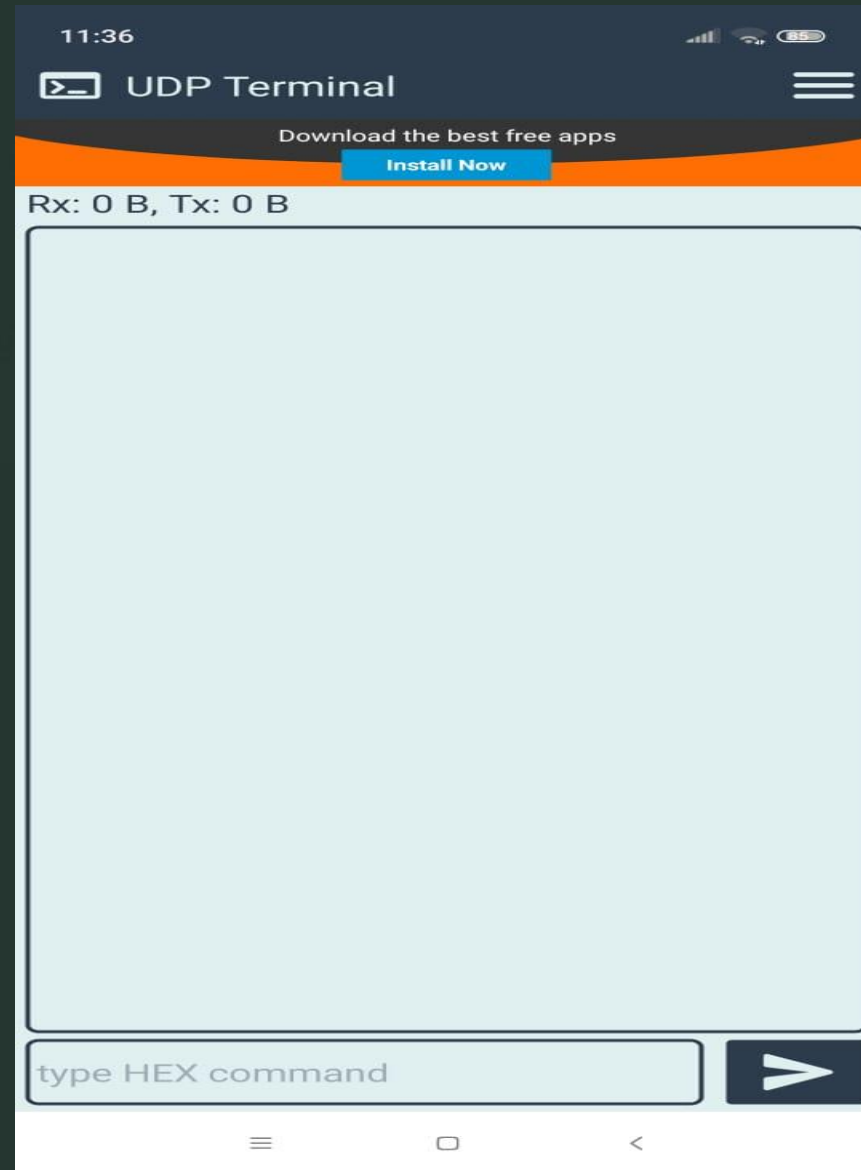
 Nifty bank
-3.40%



Search



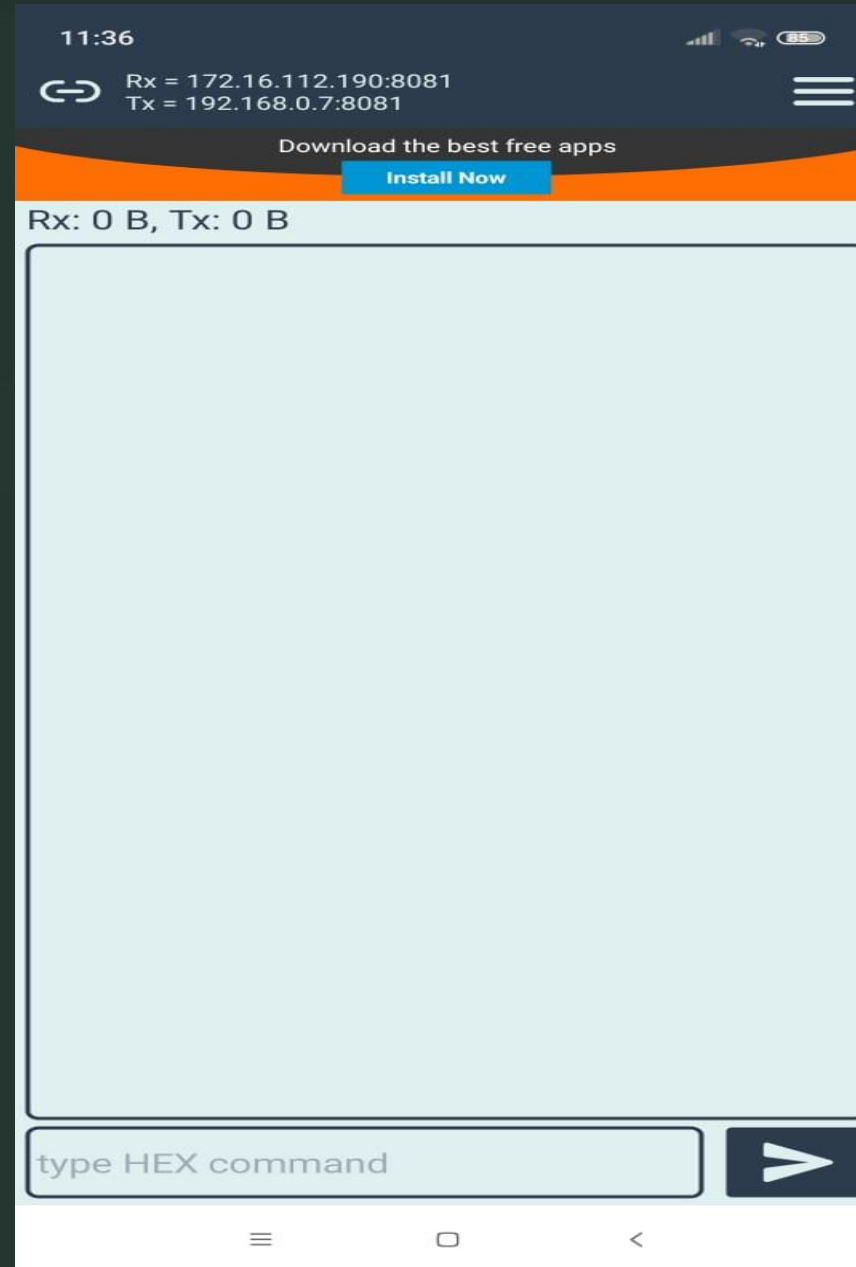
UDP terminal 1st screen



Asst. Prof. S.Asra
CSE Department, CMRIT



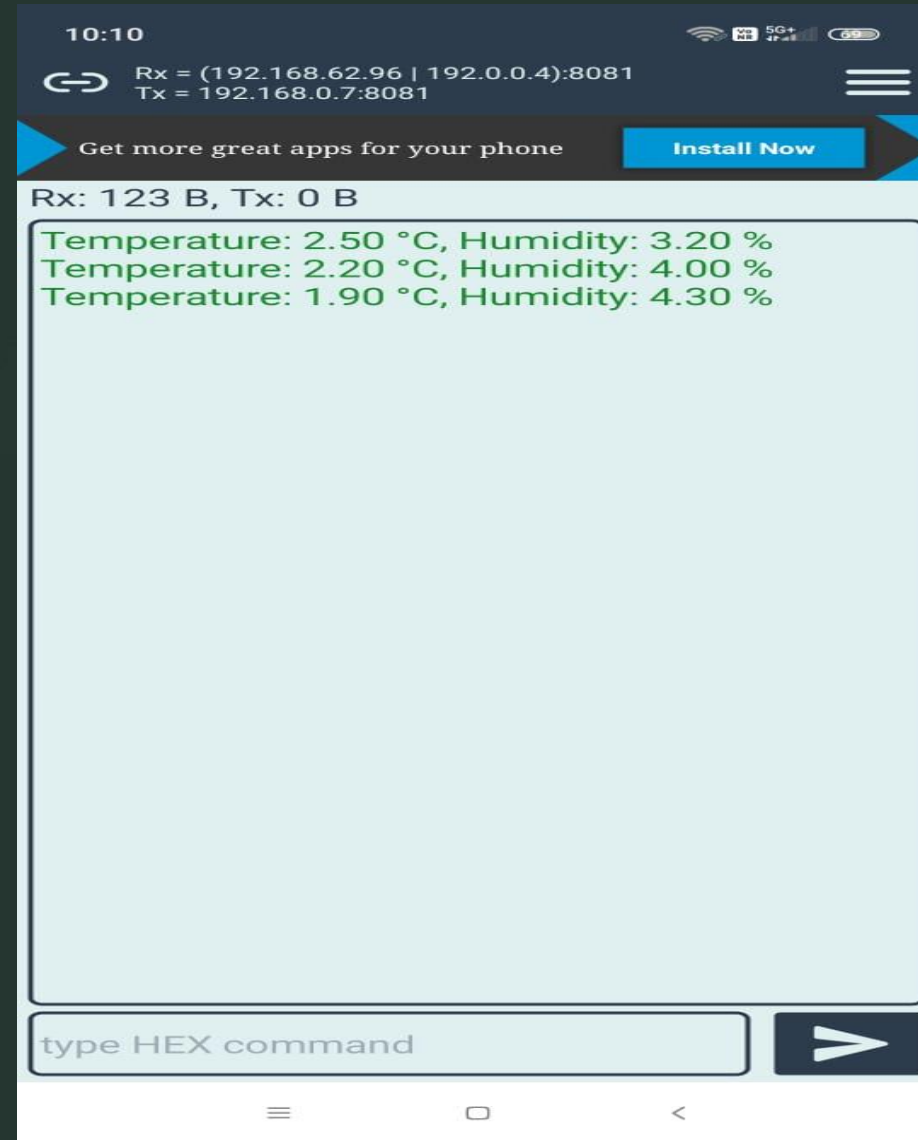
UDP terminal 2nd screen



Asst. Prof. S.Asra
CSE Department, CMRIT



UDP terminal Output screen



Asst. Prof. S.Asra
CSE Department, CMRIT



THANK YOU

