EXPERIMENT NUMBER : 3
DATE : 28/03/2022, Monday

Webserver and IoT Cloud Communication Using Arduino ...

* AIM :

Analyze various communication protocals used in the design of portable devices.

* CODE (Arduino IDE) :

(a) Identification of Available WiFi Networks -

```
/*
  This code demonstrates how to scan WiFi networks. The API is
  almost the same as with the WiFi shield library, the most
  obvious difference being the different file you need to include.
*/

#include "ESP8266WiFi.h" // ESP8266 WiFi.h library

// The setup function runs once when you prepare for reset or
   power the board :
void setup()
{
  Serial.begin(9600); // Initialize serial communication (UART) with
                         baud rate of 9600 bps

  // Set WiFi to station mode and disconnect from an access
     point, if it was previously connected :
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();
  delay(100);
```

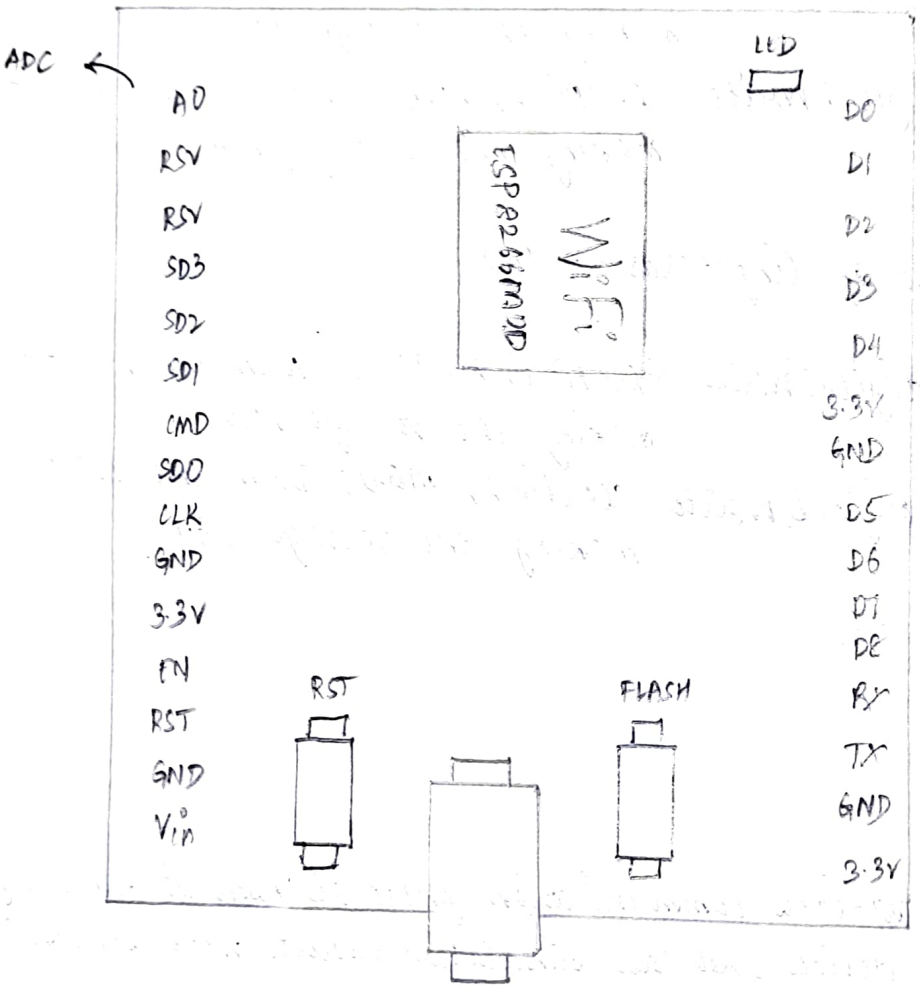
(a) Identification of Available WiFi Networks.

Components used :-

① ESP 8266 WiFi Module

② Power Supply Cable / Battery

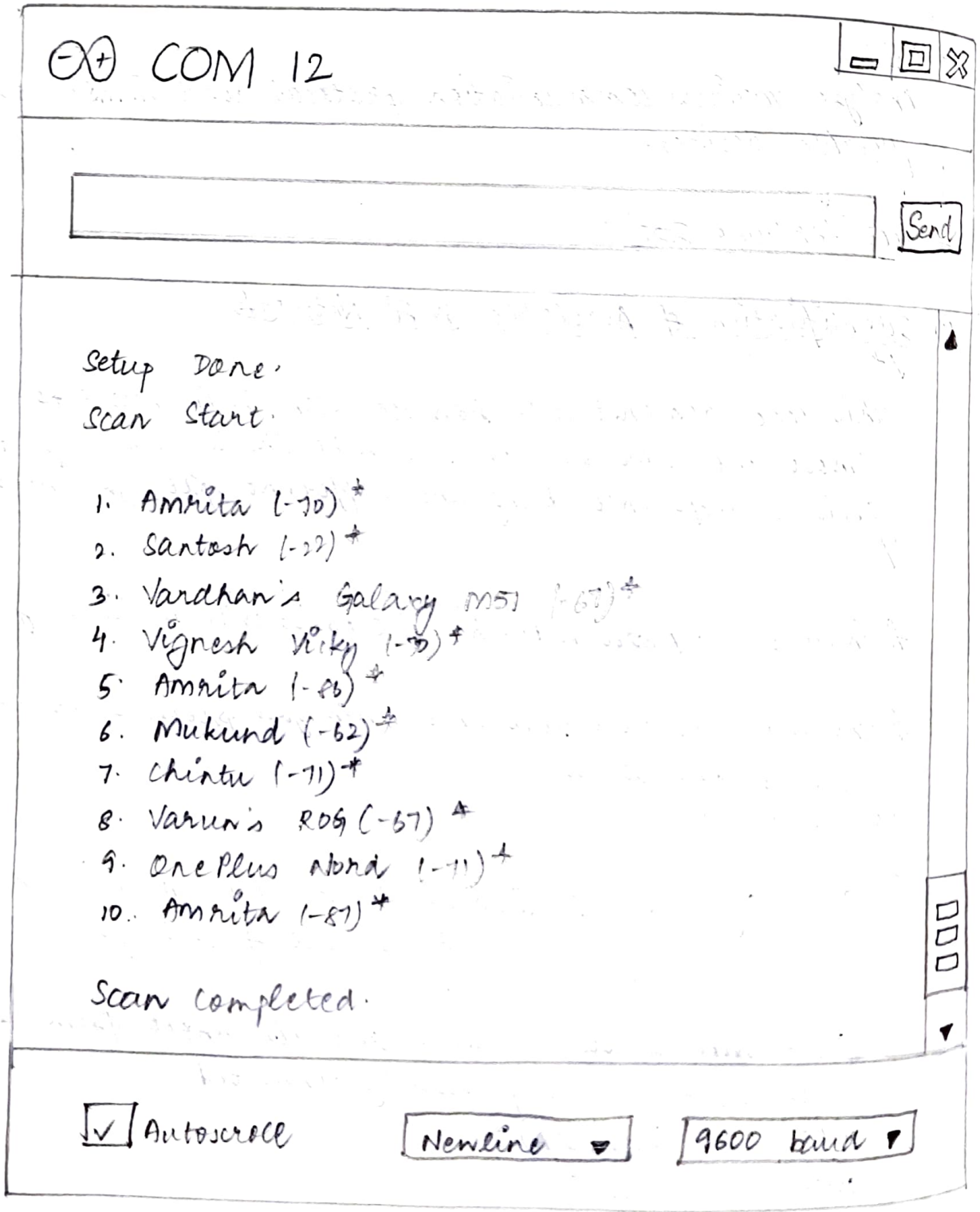
Hardware Labelled Diagram



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

// The loop function runs over and over again forever:
void loop()
{
  Serial.println("scan start.");

  // WiFi.scanNetworks will return the number of network
     found:
  int n = WiFi.scanNetworks();
  Serial.println("scan Completed.");

  if (n==0)
    Serial.println("No networks found.");
  else
  {
     Serial.print(n);
     Serial.println(" Networks Found.");

     // Print the SSID (Service Set Identifier) and RSSI
        (Received Signal strength Indicator) for each networks:
     for (int i=0; i<n; ++i)
     {
        Serial.print(i+1);
        Serial.print(": ");

        Serial.print(WiFi.SSID(i));
        Serial.print(" (");

        Serial.print(WiFi.RSSI(i));
        Serial.print(") ");
```

(a) Identification of Available WiFi Networks -

Arduino 1.6.5 - Serial Monitor View



```
        Serial.println(WiFi.encryptionType(i) == ENC_TYPE_NONE) ? " " : "*");
        delay(10);
    }
}

Serial.println("");
delay(5000); // Wait for five seconds before scanning again
}
```

(b) IoT Cloud Communication using ThingSpeak and NodeMCU -

```
#include <ESP8266WiFi.h>  // ESP8266WiFi.h library

const char* ssid = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";
const char* host = "api.thingspeak.com";
const char* writeAPIKey = "0xxD4xxx07xxSWxx8";

int temp;
void setup()
{
  // Initialize the sensor:
  pinMode(2, OUTPUT);
  pinMode(14, INPUT);

  Serial.begin(115200); // Initialize serial communication (UART)
                        with baud rate of 1,15,200 bps

  delay(1000);

  // connect to WiFi network:
  WiFi.begin(ssid, password);
```

(b) IoT Cloud Communication using ThingSpeak and NodeMCU.

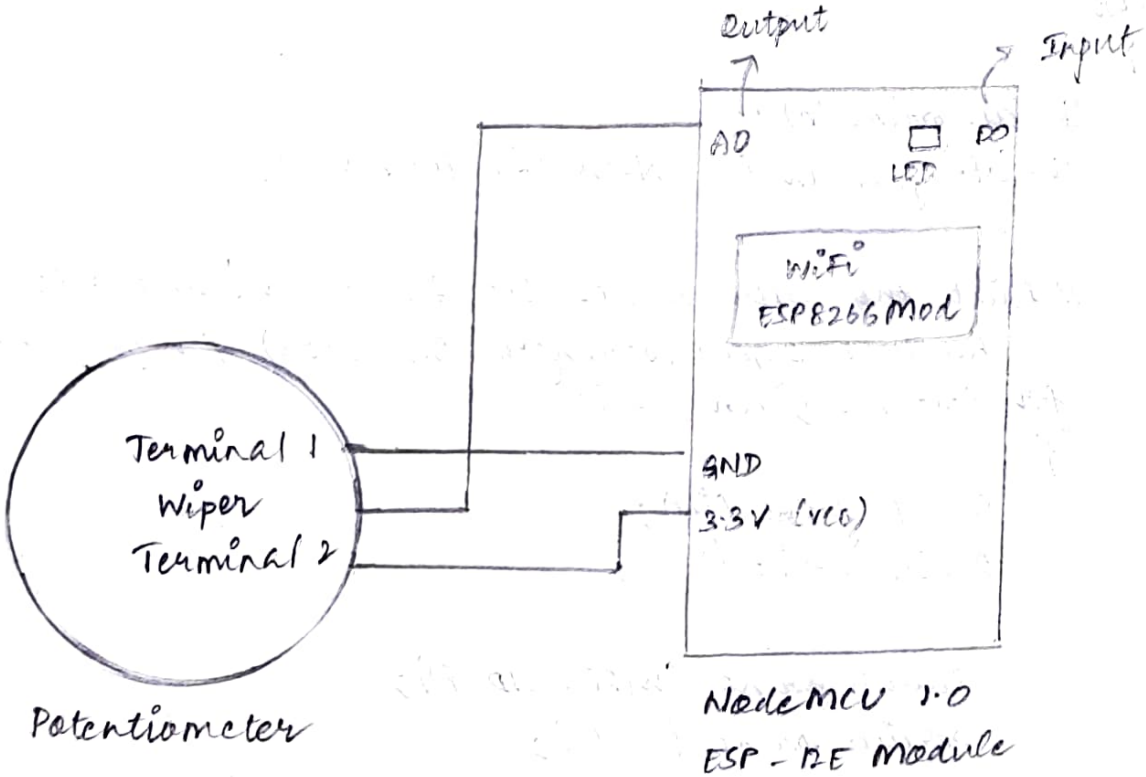
Components Required :-

1. WiFi ESP8266 Board → NodeMCU 1.0 (ESP-12E Module)
2. Potentiometer
3. Battery / Power Supply Cable
4. Connecting Wires

Software Required :-

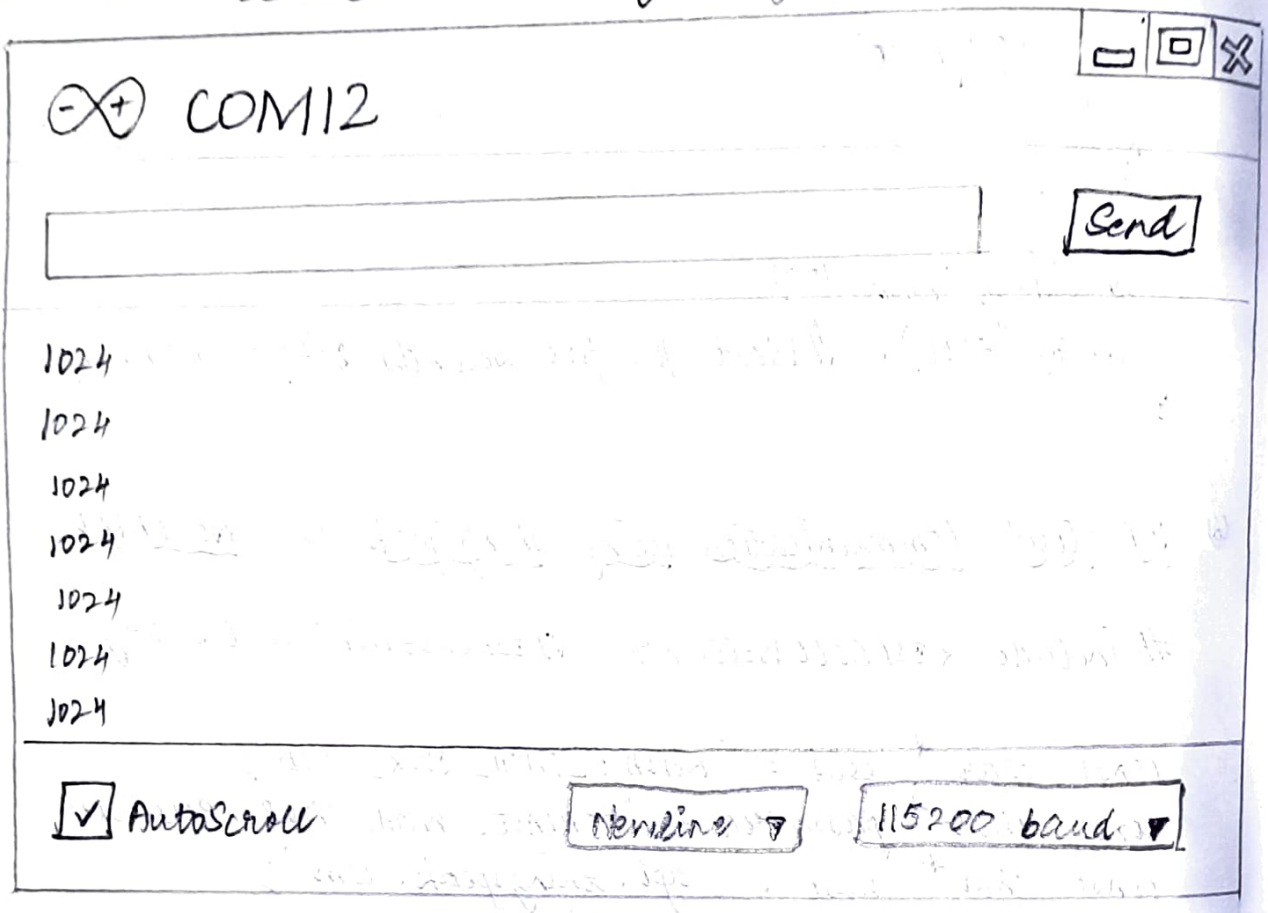
1. Arduino IDE (ESP8266 Board Installed)
2. ThingSpeak Internet of Things (IoT Analytics) (https://thingspeak.com)

Hardware Labelled Diagram

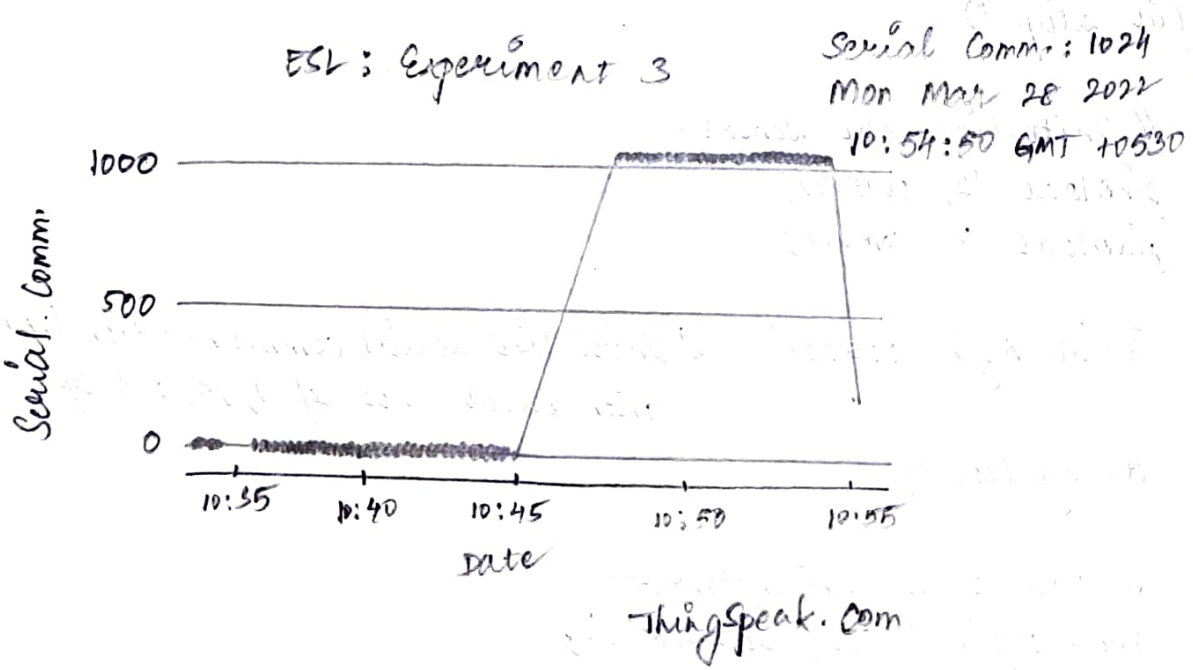


```
// Wait until WiFi connection is established:
while (WiFi.status() != WL_CONNECTED)
 {
  delay(500);
 }
}

// The loop function runs over and over again forever:
void loop()
{
  temp = analogRead(A0);  // Read ADC value from A0

  // Make TCP (Transmission Control Protocol) connections:
  WiFiClient client;
   const int httpPort = 80;

  // wait until the client (NodeMCU) is connected to the
     ThingSpeak Server:
  if (!client.connect(host, httpPort))
  {
    return;
  }

  // Request to the server:
  client.println(sendThingspeak());
  delay(1000);

  if (temp >= 700)
  {
    digitalWrite(2, HIGH);
  }
```

(b) IoT Cloud communication using ThingSpeak and Node MCU.



Arduino 1.6.5 - Serial Monitor View



```
  else
  {
    digitalWrite (2, LOW);
  }

   Serial.print (temp);
   Serial.print ("\n");
}

// Send the ADC value from Client (NodeMCU) to ThingSpeak
   Server in HTML Format:
String sendThingspeak ()
{
  String command =
          String ("GET ") +
           "/update?key=" +
          writeAPIKey +
          "&field1=" +
          String (temp) + "\r\n" +
          "HTTP/1.1 200 OK\r\n" +
          "Host: " + host + "\r\n" +
          "Connection: close\r\n\r\n";
       return command;
}
```

* **INFERENCE:**
Analyze various communication protocols used in the design of portable devices and all simulation results were successfully verified.

Experiment 3

a) Identification of available WiFi Networks

1. Initialize of Serial Communication (UART) with Baud Rate 9600 bps
2. Configure WiFi in “Station Mode” of operation
3. Scan the available WiFi Networks, which will return the no. of available WiFi networks (n).
4. If ‘n’ is zero, then print “ No Networks found”
5. If ‘n’ is non zero, print the SSID and RSSI (Received Signal Strength Indicator) for each networks.

- Initialize of Serial Communication (UART) with Baud Rate 9600 bps

```
Serial.begin(9600);
```

- Two modes available
  1. Station mode - Connect to available WiFi Network
  2. AP mode - Create its own WiFi Network & Devices can connect to this N/W

- Identification of available WiFi Networks

  - Scan the available WiFi Networks, which will return the no. of available WiFi networks (n).

```
int n = WiFi.scanNetworks();
```

Returns

```
n – No.of available Networks identified
```

  - If ‘n’ is zero, then print “ No Networks found” if (n == 0)

```
Serial.println("no networks found");
```

  - If ‘n’ is non zero, print the SSID and RSSI (Received Signal Strength Indicator) for each networks.

```
else
 {
for (int i = 0; i < n; ++i)
  {
        Serial.print(WiFi.SSID(i));
        Serial.print(WiFi.RSSI(i));
        delay(10);
  }
 }
```

b) **IoT Cloud Communication using Thingspeak and NodeMCU**

1. Initialize Serial Communication – UART
2. Initialize the WiFi Network settings and return the current status (Connected or not).
3. Wait until Wifi Connection is established.
4. Read ADC value from A0
5. Wait until the **Client (NodeMCU)** is connected to the Thingspeak Server
6. Send the **ADC value from Client (NodeMCU) to Thingspeak Server** in HTML Format.