Experiment No. – 4

Title: IoT based Stepper Motor/DC Motor Control with Arduino/Raspberry Pi.

Aim: DC motor control using WEB.

Hardware Requirements: Arduino Board, Dc Motor, ESP8266_Rx ,ESP8266_Tx

Software Requirements: Arduino IDE

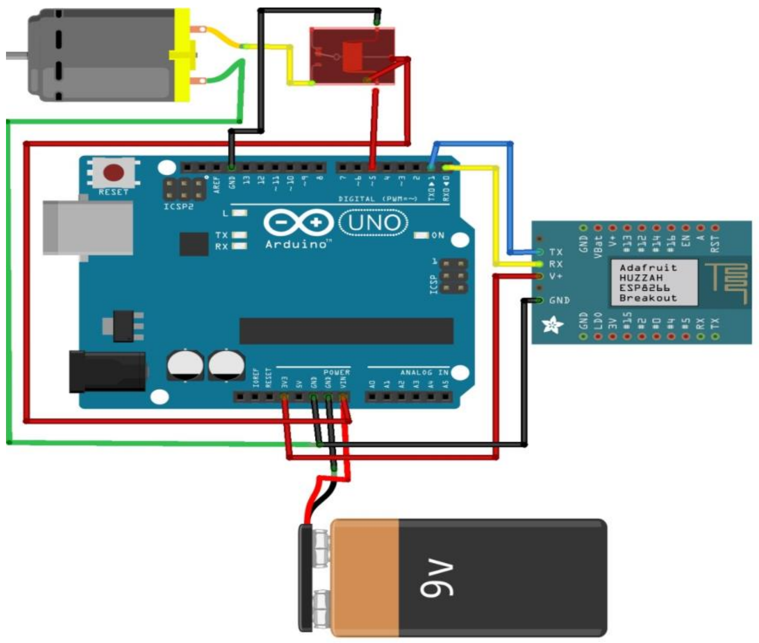
Theory:

Dc Motor:

A DC motor (Direct Current motor) is the most common type of motor. DC motors normally have just two leads, one positive and one negative. If you connect these two leads directly to a battery, the motor will rotate. If you switch the leads, the motor will rotate in the opposite direction.

A motor is an electrical machine which converts electrical energy into mechanical energy. The principle of working of a DC motor is that" whenever a current carrying conductor is placed in a magnetic field, it experiences a mechanical force". The direction of this force is given by Fleming's left hand rule and it's magnitude is given by F = BIL. Where, B= magnetic flux density, I = current and L = length of the conductor within the magnetic field.

Circuit Diagram:



Procedure:

Step 1: Connect the Arduino board to the Micro-IoT Sensor board using the FRC cable provided with the board.

Step 2: Connect the Power supply adaptor and power on the circuit.

Step 3: Open Arduino IDE and create a new sketch (program) using the above pins. take down the details for SSID and password for the Wi-Fi or hotspot and use these in your program.

Step 4: In the Arduino IDE go to tools→Port and select the appropriate COM port.

Step 5: In the Arduino IDE click on the upload button () to compile and download the code into the Arduino UNO. When successfully downloaded the code will start running.

Step 6: This experiment uses the ESP8266 Wi-Fi module. To connect the module to the circuit slide the switch SW6 between 1-2 (WiFi_ON) . Reset the ESP8266 by pressing the switch SW10. Now reset the Arduino board.

Step 7: Connect the DC motor to green terminal connector opposite RELAY1. When the relay is activated the DC motor is powered.

Step 8: Open the serial monitor @115200 baud rate. Note down the IP address displayed on the serial terminal. Open the browser of your mobile or PC. Make sure that the be mobile/PC is in the same network. Type the IP address of the ESP8266. The web page will displayed. The webpage will display the ON / OFF link. Click on the link and observe the behavior on the board.

Observation: We can control the DC motor ON / OFF using the Webpage thus we can see the IoT in action.

Conclusion:

Code:

```
/*
simple demo to switch the DC motor ON-OFF using webserver and iot

 A simple web server that lets you turn on and of an DC motor via a web page.
 This sketch will print the IP address of your ESP8266 module (once connected)
 to the Serial monitor. From there, you can open that address in a web browser
 to turn on and off the DC motor.

*/

#include "WiFiEsp.h"

// Emulate Serial1 on pins 8/7 if not present
#ifndef HAVE_HWSERIAL1
#include "SoftwareSerial.h"
SoftwareSerial Serial1(8, 7); // RX, TX
#endif

char ssid[] = "Microembedded";            // your network SSID (name)
char pass[] = "Micro@123";        // your network password
int status = WL_IDLE_STATUS;

int DCMotorStatus = LOW;

WiFiEspServer server(80);

// use a ring buffer to increase speed and reduce memory allocation
RingBuffer buf(8);

void setup()
{
  Serial.begin(115200);   // initialize serial for debugging
  Serial1.begin(9600);    // initialize serial for ESP module
  WiFi.init(&Serial1);    // initialize ESP module

//  pinMode(5, OUTPUT);
//  pinMode(A2, OUTPUT);
//  digitalWrite(5, 0);
//  digitalWrite(A2, 0);
```

```
  pinMode(9, OUTPUT);
  digitalWrite(9, 0);

  // check for the presence of the shield
  if (WiFi.status() == WL_NO_SHIELD) {
    Serial.println("WiFi shield not present");
    // don't continue
    while (true);

  }

  // attempt to connect to WiFi network
  while (status != WL_CONNECTED) {
    Serial.print("Attempting to connect to WPA SSID: ");
    Serial.println(ssid);
    // Connect to WPA/WPA2 network
    status = WiFi.begin(ssid, pass);
  }

  Serial.println("You're connected to the network");
  printWifiStatus();

  // start the web server on port 80
  server.begin();
}

void loop()
{
  WiFiEspClient 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
    buf.init();                               // initialize the circular buffer
    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
        buf.push(c);                          // push it to the ring buffer

        // printing the stream to the serial monitor will slow down
        // the receiving of data from the ESP filling the serial buffer
        //Serial.write(c);
        // you got two newline characters in a row
        // that's the end of the HTTP request, so send a response
        if (buf.endsWith("\r\n\r\n")) {
          sendHttpResponse(client);
          break;
        }

        // Check to see if the client request was "GET /H" or "GET /L":
        if (buf.endsWith("GET /H")) {
          Serial.println("Turn DC Motor ON");
          DCMotorStatus = HIGH;
         // digitalWrite(5, LOW);
         // digitalWrite(A2, HIGH);
           digitalWrite(9, HIGH);
        }
        else if (buf.endsWith("GET /L")) {
          Serial.println("Turn DC Motor OFF");
          DCMotorStatus = LOW;
          //digitalWrite(5, HIGH);
          //digitalWrite(A2, LOW);
          digitalWrite(9, LOW);
          }
      }
    }

    // close the connection
    client.stop();
    Serial.println("Client disconnected");
  }
}

void sendHttpResponse(WiFiEspClient client)
{
  // 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.print("The DC Motor is : ");
```

```
 if(DCMotorStatus==0)
 {
  client.print("OFF");
  }
  else
  {
    client.print("ON");
   }
  client.println("<br>");
  client.println("<br>");

  client.println("Click <a href=\"/H\">here</a> turn the DC Motor on<br>");
  client.println("Click <a href=\"/L\">here</a> turn the DC Motor off<br>");

  // The HTTP response ends with another blank line:
  client.println();
}

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);

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