MIOT EXP CODES

EXP NO. 2

```
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the
voltage level)
  delay(10000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the
voltage LOW
  delay(10000);                       // wait for a second
}
=================
int switch = 7;
int led = 12;
void setup()
{
 pinMode(13, INPUT);
 pinMode(12, OUTPUT);
}
void loop()
{
 if(digitalRead(switch) == 1)
 {
 digitalWrite(led, HIGH);

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

EXP 3 DHT11

```
#include <dht.h>
dht DHT;
#define DHT11_PIN 7
void setup(){
Serial.begin(9600);
}
void loop()
{
int chk = DHT.read11(DHT11_PIN);
Serial.print("Temperature = ");
Serial.println(DHT.temperature);
Serial.print("Humidity = ");
Serial.println(DHT.humidity);
delay(1000);
```

EXP 4 STEPPER MOTOR

```
int Pin1 = 10;
int Pin2 = 11;
int Pin3 = 12;
int Pin4 = 13;
int _step = 0;
boolean dir = false;// false=clockwise, true=counter clockwise
int count=0;
void setup()
{
pinMode(Pin1, OUTPUT);
pinMode(Pin2, OUTPUT);
pinMode(Pin3, OUTPUT);
pinMode(Pin4, OUTPUT);
}
void loop()
{
switch(_step){
case 0:
digitalWrite(Pin1, LOW);
digitalWrite(Pin2, LOW);
digitalWrite(Pin3, LOW);
digitalWrite(Pin4, HIGH);
break;
case 1:
digitalWrite(Pin1, LOW);
digitalWrite(Pin2, LOW);
digitalWrite(Pin3, HIGH);
digitalWrite(Pin4, HIGH);
break;
case 2:
digitalWrite(Pin1, LOW);
digitalWrite(Pin2, LOW);
digitalWrite(Pin3, HIGH);
digitalWrite(Pin4, LOW);
break;
case 3:
digitalWrite(Pin1, LOW);
digitalWrite(Pin2, HIGH);
digitalWrite(Pin3, HIGH);
digitalWrite(Pin4, LOW);
break;
case 4:
```

```
digitalWrite(Pin1, LOW);
digitalWrite(Pin2, HIGH);
digitalWrite(Pin3, LOW);
digitalWrite(Pin4, LOW);
break;
case 5:
digitalWrite(Pin1, HIGH);
digitalWrite(Pin2, HIGH);
digitalWrite(Pin3, LOW);
digitalWrite(Pin4, LOW);
break;
case 6:
digitalWrite(Pin1, HIGH);
digitalWrite(Pin2, LOW);
digitalWrite(Pin3, LOW);
digitalWrite(Pin4, LOW);
break;
case 7:
digitalWrite(Pin1, HIGH);
digitalWrite(Pin2, LOW);
digitalWrite(Pin3, LOW);
digitalWrite(Pin4, HIGH);
break;
default:
digitalWrite(Pin1, LOW);
digitalWrite(Pin2, LOW);
digitalWrite(Pin3, LOW);
digitalWrite(Pin4, LOW);
break;
}
if(dir){
_step++;
}else{
_step--;
}
if(_step>7){
_step=0;
}
if(_step<0){
_step=7;
}
delay(1);
}
```

THINGSPEAK DHT11 EXP 5

```
#include <DHT.h>  // Including library for dht

#include <ESP8266WiFi.h>
String apiKey = "H38TEGNC0XKW43BB";     //  Enter your Write API key from ThingSpeak

const char *ssid =  "how2electronics";  // replace with your wifi ssid and wpa2 key
const char *pass =  "alhabibi";
const char* server = "api.thingspeak.com";

#define DHTPIN 0          //pin where the dht11 is connected
DHT dht(DHTPIN, DHT11);

WiFiClient client;
void setup()
{
      Serial.begin(115200);
      delay(10);
      dht.begin();
      Serial.println("Connecting to ");
      Serial.println(ssid);
      WiFi.begin(ssid, pass);
      while (WiFi.status() != WL_CONNECTED)
      {
            delay(500);
            Serial.print(".");
      }
      Serial.println("");
      Serial.println("WiFi connected");
}
void loop()
{

      float h = dht.readHumidity();
      float t = dht.readTemperature();

               if (isnan(h) || isnan(t))
                  {
                  Serial.println("Failed to read from DHT sensor!");
                       return;
                  }

                        if (client.connect(server,80))   //
"184.106.153.149" or api.thingspeak.com
                        {
```

```
                            String postStr = apiKey;
                            postStr +="&field1=";
                            postStr += String(t);
                            postStr +="&field2=";
                            postStr += String(h);
                            postStr += "\r\n\r\n";
                            client.print("POST /update HTTP/1.1\n");
                            client.print("Host: api.thingspeak.com\n");
                            client.print("Connection: close\n");
                            client.print("X-THINGSPEAKAPIKEY:
"+apiKey+"\n");
                            client.print("Content-Type:
application/x-www-form-urlencoded\n");
                            client.print("Content-Length: ");
                            client.print(postStr.length());
                            client.print("\n\n");
                            client.print(postStr);
                            Serial.print("Temperature: ");
                            Serial.print(t);
                            Serial.print(" degrees Celcius, Humidity: ");
                            Serial.print(h);
                            Serial.println("%. Send to Thingspeak.");
                        }
          client.stop();
          Serial.println("Waiting...");

  // thingspeak needs minimum 15 sec delay between updates
  delay(1000);
}
```

MIOT EXP 6 REMOTE LAN

```
// Load Wi-Fi library
#include <ESP8266WiFi.h>

// Replace with your network credentials
const char* ssid     = "Manisha";
const char* password = "manisha@2311";

// Set web server port number to 80
WiFiServer server(80);

// Variable to store the HTTP request
String header;

// Auxiliar variables to store the current output state
String output5State = "off";
String output4State = "off";

// Assign output variables to GPIO pins
const int output5 = D5;
const int output4 = D4;

// Current time
unsigned long currentTime = millis();
// Previous time
unsigned long previousTime = 0;
// Define timeout time in milliseconds (example: 2000ms = 2s)
const long timeoutTime = 2000;

void setup() {
  Serial.begin(115200);
  // Initialize the output variables as outputs
  pinMode(output5, OUTPUT);
  pinMode(output4, OUTPUT);
  // Set outputs to LOW
  digitalWrite(output5, LOW);
  digitalWrite(output4, LOW);

  // Connect to Wi-Fi network with SSID and password
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
```

```
    delay(500);
    Serial.print(".");
  }
  // Print local IP address and start web server
  Serial.println("");
  Serial.println("WiFi connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();
}

void loop(){
  WiFiClient client = server.available();   // Listen for incoming clients

  if (client) {                             // If a new client connects,
    Serial.println("New Client.");          // print a message out in the serial port
    String currentLine = "";                // make a String to hold incoming data from the client
    currentTime = millis();
    previousTime = currentTime;
    while (client.connected() && currentTime - previousTime <= timeoutTime) { // loop while the
client's connected
      currentTime = millis();
      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
        header += c;
        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 (currentLine.length() == 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("Connection: close");
            client.println();

            // turns the GPIOs on and off
            if (header.indexOf("GET /5/on") >= 0) {
              Serial.println("GPIO 5 on");
              output5State = "on";
              digitalWrite(output5, HIGH);
            } else if (header.indexOf("GET /5/off") >= 0) {
              Serial.println("GPIO 5 off");
```

```
            output5State = "off";
            digitalWrite(output5, LOW);
          } else if (header.indexOf("GET /4/on") >= 0) {
            Serial.println("GPIO 4 on");
            output4State = "on";
            digitalWrite(output4, HIGH);
          } else if (header.indexOf("GET /4/off") >= 0) {
            Serial.println("GPIO 4 off");
            output4State = "off";
            digitalWrite(output4, LOW);
          }

          // Display the HTML web page
          client.println("<!DOCTYPE html><html>");
          client.println("<head><meta name=\"viewport\" content=\"width=device-width,
initial-scale=1\">");
          client.println("<link rel=\"icon\" href=\"data:,\">");
          // CSS to style the on/off buttons
          // Feel free to change the background-color and font-size attributes to fit your
preferences
          client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto;
text-align: center;}");
          client.println(".button { background-color: #195B6A; border: none; color: white; padding:
16px 40px;");
          client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}");
          client.println(".button2 {background-color: #77878A;}</style></head>");

          // Web Page Heading
          client.println("<body><h1>ESP8266 Web Server</h1>");

          // Display current state, and ON/OFF buttons for GPIO 5
          client.println("<p>GPIO 5 - State " + output5State + "</p>");
          // If the output5State is off, it displays the ON button
          if (output5State=="off") {
            client.println("<p><a href=\"/5/on\"><button class=\"button\">ON</button></a></p>");
          } else {
            client.println("<p><a href=\"/5/off\"><button class=\"button
button2\">OFF</button></a></p>");
          }

          // Display current state, and ON/OFF buttons for GPIO 4
          client.println("<p>GPIO 4 - State " + output4State + "</p>");
          // If the output4State is off, it displays the ON button
          if (output4State=="off") {
```

```
            client.println("<p><a href=\"/4/on\"><button class=\"button\">ON</button></a></p>");
          } else {
            client.println("<p><a href=\"/4/off\"><button class=\"button
button2\">OFF</button></a></p>");
          }
          client.println("</body></html>");

          // 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 currentLine
          currentLine = "";
        }
      } else if (c != '\r') {  // if you got anything else but a carriage return character,
        currentLine += c;      // add it to the end of the currentLine
      }
    }
  }
  // Clear the header variable
  header = "";
  // Close the connection
  client.stop();
  Serial.println("Client disconnected.");
  Serial.println("");
 }
}
```

MIOT EXP 7 HC05

```
char Incoming_value = 0;                //Variable for storing Incoming_value
void setup()
{
  Serial.begin(9600);         //Sets the data rate in bits per second (baud) for serial data
transmission
  pinMode(13, OUTPUT);        //Sets digital pin 13 as output pin
}
void loop()
{
  if(Serial.available() > 0)
  {
    Incoming_value = Serial.read();      //Read the incoming data and store it into variable
Incoming_value
    Serial.print(Incoming_value);        //Print Value of Incoming_value in Serial monitor
    Serial.print("\n");        //New line
    if(Incoming_value == '1')            //Checks whether value of Incoming_value is equal to
1
      digitalWrite(13, HIGH);  //If value is 1 then LED turns ON
    else if(Incoming_value == '0')       //Checks whether value of Incoming_value is equal
to 0
      digitalWrite(13, LOW);   //If value is 0 then LED turns OFF
  }

}
```