Week 1: LED

```
#include <ESP8266WiFi.h>
int IRPin =2;
int led=13;
int value;
void setup(){
pinMode(IRPin,INPUT);
Serial.begin(9600);
pinMode(led,OUTPUT);

}
void loop(){
 value = digitalRead(IRPin);
Serial.println(value);
 if(digitalRead(IRPin)==0)
 {
 digitalWrite(led,HIGH);
 Serial.println("object detected");

 }
 else
```

```
  {
  digitalWrite(led,LOW);
   Serial.println("object not detected");


}
}
```

Week 2: ultrasonic sensor

```
#define ECHOPIN 7        // Pin to receive echo pulse
#define TRIGPIN 8
int led=12;
int a,b;
void setup()
{
  Serial.begin(9600);
  pinMode(ECHOPIN, INPUT);
  pinMode(TRIGPIN, OUTPUT);
  pinMode(led,OUTPUT);
}
```

```
void loop()
{
  digitalWrite(TRIGPIN, LOW);
  delayMicroseconds(2000);
  digitalWrite(TRIGPIN, HIGH);
  delayMicroseconds(1000);
  digitalWrite(TRIGPIN, LOW);
  float a = pulseIn(ECHOPIN, HIGH);
  digitalWrite(led,HIGH);
  b= a*0.0344/2;
  Serial.print(b);
  Serial.println(" cm");

  delay(3000);
}
```

Week3 : Bluetooth

```
#include <SoftwareSerial.h>
SoftwareSerial Bluetooth(8, 9); // RX, TX
int LED = 12; // the on-board LED
int Data; // the data received
void setup() {
  Bluetooth.begin(9600);
  Serial.begin(9600);
  Serial.println("Waiting for command...");
  Bluetooth.println("Send 1 to turn on the LED. Send 0 to turn Off");
  pinMode(LED,OUTPUT);
}
void loop() {
  if (Bluetooth.available()){ //wait for data received
Data=Bluetooth.read();
 if(Data=='1'){
digitalWrite(LED,HIGH);
Serial.println("LED On!");
Bluetooth.println("LED On!");
    }
```

```
    else if(Data=='0'){
digitalWrite(LED,LOW);
 Serial.println("LED Off!");
Bluetooth.println("LED Off ! ");
    }
  else{;}
  }
delay(1000);
}
```

Week 4: RFID

Readinfo:

```
#include <SPI.h>
#include <MFRC522.h>
#define RST_PIN        9
#define SS_PIN        10
MFRC522 mfrc522(SS_PIN, RST_PIN);
void setup() {
  Serial.begin(9600);
```

```cpp
  SPI.begin();

  mfrc522.PCD_Init();

  Serial.println(F("Read personal data"));

}

void loop() {

  MFRC522::MIFARE_Key key;

  for (byte i = 0; i < 6; i++)

  key.keyByte[i] = 0xFF;

  byte block;

  byte len;

  MFRC522::StatusCode status;

  if ( ! mfrc522.PICC_IsNewCardPresent()) {

    return;

  }

  if ( ! mfrc522.PICC_ReadCardSerial()) {

    return;

  }

  Serial.println(F("**Card Detected:**"));

  mfrc522.PICC_DumpDetailsToSerial(&(mfrc522.uid));

  Serial.print(F("Name: "));

  block = 4;

  len = 18;
```

```cpp
  byte buffer2[18];
  block = 1;
  status =
mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH
_KEY_A, 1, &key, &(mfrc522.uid));
  if (status != MFRC522::STATUS_OK) {
    Serial.print(F("Authentication failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
  }
  status = mfrc522.MIFARE_Read(block, buffer2, &len);
  if (status != MFRC522::STATUS_OK) {
    Serial.print(F("Reading failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
  }
  for (uint8_t i = 0; i < 16; i++) {
    Serial.write(buffer2[i]);
  }
  Serial.println(F("\n**End Reading**\n"));
  delay(1000); //change value if you want to read cards faster
  mfrc522.PICC_HaltA();
```

mfrc522.PCD_StopCrypto1();}

Writeinfo:

```
#include <SPI.h>
#include <MFRC522.h>
#define RST_PIN        9
#define SS_PIN        10
MFRC522 mfrc522(SS_PIN, RST_PIN);
void setup() {
  Serial.begin(9600);
  SPI.begin();
  mfrc522.PCD_Init();
  Serial.println(F("Write personal data on a MIFARE PICC "));
}
void loop() {
  MFRC522::MIFARE_Key key;
  for (byte i = 0; i < 6; i++)
  key.keyByte[i] = 0xFF;
  if ( ! mfrc522.PICC_IsNewCardPresent()) {
    return;
```

```cpp
  }
  if ( ! mfrc522.PICC_ReadCardSerial()) {
    return;
  }
  Serial.print(F("Card UID:"));
  for (byte i = 0; i < mfrc522.uid.size; i++) {
    Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
    Serial.print(mfrc522.uid.uidByte[i], HEX);
  }
  Serial.print(F(" PICC type: "));
  MFRC522::PICC_Type piccType =
mfrc522.PICC_GetType(mfrc522.uid.sak);
  Serial.println(mfrc522.PICC_GetTypeName(piccType));
  byte buffer[34];
  byte block;
  MFRC522::StatusCode status;
  byte len;
  Serial.setTimeout(20000L) ;
  // Ask personal data: First name
  Serial.println(F("Type First name, ending with #"));
  len = Serial.readBytesUntil('#', (char *) buffer, 20) ;
  for (byte i = len; i < 20; i++) buffer[i] = ' ';
```

```cpp
  block = 1;
  status =
mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH
_KEY_A, block, &key, &(mfrc522.uid));
  if (status != MFRC522::STATUS_OK) {
    Serial.print(F("PCD_Authenticate() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
  }
  status = mfrc522.MIFARE_Write(block, buffer, 16);
  if (status != MFRC522::STATUS_OK) {
    Serial.print(F("MIFARE_Write() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
  }
  else Serial.println(F("MIFARE_Write() success: "));
  block = 2;
  status =
mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH
_KEY_A, block, &key, &(mfrc522.uid));
  if (status != MFRC522::STATUS_OK) {
    Serial.print(F("PCD_Authenticate() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
```

```
    return;
  }
  status = mfrc522.MIFARE_Write(block, &buffer[16], 16);
  if (status != MFRC522::STATUS_OK) {
    Serial.print(F("MIFARE_Write() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
  }
  else Serial.println(F("MIFARE_Write() success: "));
  Serial.println(" ");
  mfrc522.PICC_HaltA();
  mfrc522.PCD_StopCrypto1();
}
```

Week5 : Temperature and Humidity using Ardunio

```
#include <DHT.h>
#define DHTPIN 8
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
void setup() {
```

```
 Serial.begin(9600);
Serial.println(F("DHT test!"));
dht.begin();
}
void loop() {
 delay(2000);
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  if (isnan(h) || isnan(t))
{
  Serial.println(F("Failed to read from DHT sensor!"));
   return;
 }
 Serial.print(F(" Humidity: "));
 Serial.print(h);
 Serial.print(F("% Temperature: "));
 Serial.print(t);
}
```

Week6: IR sensor

```
#include <ESP8266WiFi.h>
String apiKey = "PFFHC1B2IRN0GGQ0";
const char *ssid =  "Veeresh";
const char *pass =  "veeru1987";
const char* server = "api.thingspeak.com";
#define IRpin D4
WiFiClient client;
int value;
void setup()
{
Serial.begin(115200);
pinMode(IRpin, INPUT);
delay(1000);
Serial.println("Connecting to ");
Serial.println(ssid);
WiFi.begin(ssid, pass);
while (WiFi.status() != WL_CONNECTED)
{
delay(2000);
Serial.print(".");
}
Serial.println("   ");
```

```
Serial.println("WiFi connected");

}

void loop(){

 value = digitalRead(IRpin);

Serial.println(value);

  if(value==0)

  {

  Serial.print("object detected");

  }

  else

  {

  Serial.print("no object detected");

  }

if (client.connect(server,80))

{

String postStr = apiKey;

postStr +="&field1=";

postStr += String(value);

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);

client.stop();


Serial.println("Waiting...");


delay(1000);
}
}
```

Week 7: upload data to the cloud

```
#include <DHT.h>

#include <ESP8266WiFi.h>

String apiKey = "E5E5FA7AF200H6I6";  //

const char *ssid =  "Anuja";
```

```arduino
const char *pass =  "Anuja123";

const char* server = "api.thingspeak.com";

#define DHTPIN D3

 DHT dht(DHTPIN, DHT11);

WiFiClient client;

 void setup()

{

  Serial.begin(115200);

  delay(1000);

  dht.begin();

  Serial.println("Connecting to ");

  Serial.println(ssid);

  WiFi.begin(ssid, pass);

  while (WiFi.status() != WL_CONNECTED)

  {

    delay(2000);

    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...");
  delay(1000);
}
```

Week 8 : retrieve data from cloud

```
#include <DHT.h>

#include <ESP8266WiFi.h>
```

```cpp
String apiKey = "UYK6QJGVZPSG5LF2";   //read
const char *ssid =  "Anuja";
const char *pass =  "Anuja123";
const char* server = "api.thingspeak.com";
#define DHTPIN D3
 DHT dht(DHTPIN, DHT11);
WiFiClient client;
 void setup()
{
  Serial.begin(115200);
  delay(1000);
  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(h);
  postStr +="&field2=";
  postStr += String(t);
  postStr += "\r\n\r\n";
  client.print("X-THINGSPEAKAPIKEY: "+apiKey+"\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...");

    delay(1000);

  }
```

Week 9: TCP server on cloud

```
#include "ESP8266WiFi.h"

#include "DHT.h"

const char* ssid = "Anuja";

const char* password = "Anuja123";

WiFiServer wifiServer(8080);
```

```
DHT dht(D3, DHT11);
void setup() {
  Serial.begin(115200);
  delay(1000);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting..");
  }
  Serial.print("Connected to WiFi. IP:");
  Serial.println(WiFi.localIP());
  wifiServer.begin();
  dht.begin();
}

void loop() {
  WiFiClient client = wifiServer.available();
  if (client) {
    while (client.connected()) {
      while (client.available()>0) {
        float t=dht.readTemperature();
        float h = dht.readHumidity();
```

```
        client.println("humidity :");

        client.println("temperature :");

        client.println(h);

        Serial.println(h);

        client.println(t);

        Serial.println(t);

        delay(2000);

      }

    }

    client.stop();

    Serial.println("Client disconnected");

  }

}
```

Week 10: UDP server on cloud

```
#include <ESP8266WiFi.h>

#include <WiFiUdp.h>

#include <DHT.h>

const char* ssid = "Galaxy A21sE600";
```

```cpp
const char* password = "zilh8480";

const char* udpAddress = "192.168.68.144";

const int udpPort = 1234;

#define DHTPIN D3

#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

WiFiUDP udp;

void setup() {
  Serial.begin(115200);
  Serial.println();
  Serial.println("Connecting to WiFi...");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting");
  }
  Serial.println();
  Serial.println("Connected to WiFi.IP:");
  dht.begin();
}

void loop() {
  delay(10000);
```

```
float temperature = dht.readTemperature();
float humidity = dht.readHumidity();
if (isnan(temperature) || isnan(humidity)) {
  Serial.println("Failed to read from DHT sensor!");
  return;
}
Serial.print("Temperature: ");
Serial.print(temperature);
Serial.print(" °C\tHumidity: ");
Serial.print(humidity);
Serial.println(" %");
Serial.println("Sending data over UDP...");
udp.beginPacket(udpAddress, udpPort);
udp.print("Temperature: ");
udp.print(temperature);
udp.print(" °C, Humidity: ");
udp.print(humidity);
udp.println(" %");
udp.endPacket();
Serial.println("Data sent over UDP.");
}
```