sketch_aug03a | Arduino 1.8.13 (Windows Store 1.8.42.0)

File Edit Sketch Tools Help

sketch_aug03a §

```
#include <dht.h>
#define dht_apin A0

dht DHT;

void setup(){

  Serial.begin(9600);
  delay(500);
  Serial.println("DHT11 Humidity & temperature Sensor\n\n");
  delay(1000);

}

void loop(){
    DHT.read11(dht_apin);
    Serial.print("Current humidity = ");
    Serial.print(DHT.humidity);
    Serial.print("%  ");
    Serial.print("temperature = ");
    Serial.print(DHT.temperature);
    Serial.println("C  ");
    delay(5000);

}
```

sketch_aug03a | Arduino 1.8.13 (Windows Store 1.8.42.0)

File Edit Sketch Tools Help

sketch_aug03a §

```
long interval = 100;  // interval at which LED blinks

void setup()
{
  //The Following are our output
  pinMode(ledPin,OUTPUT);
  pinMode(buzzerPin,OUTPUT);

  //Button is our Input
  pinMode(buttonPin, INPUT);

  // Wait before starting the alarm
  delay(5000);
}

void loop()
{
  // To chech whether the motion is detected or not
  if (digitalRead(motionPin)) {
    buzzer_mode = true;
  }

  // If alarm mode is on,blink our LED
  if (buzzer_mode){
    unsigned long currentMillis = millis();
    if(currentMillis - previousMillis > interval) {
      previousMillis = currentMillis;
      if (ledState == LOW)
        ledState = HIGH;
```

```
#define TOKEN "12345678"


char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;


const char eventTopic[] = "iot-2/evt/status/fmt/json";
const char cmdTopic[] = "iot-2/cmd/led/fmt/json";




WiFiClient wifiClient;
void callback(char* topic, byte* payload, unsigned int payloadLength) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < payloadLength; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();


  // Switch on the LED if an 1 was received as first character
```

sketch_aug03a §

```
int Publish(char* payload, int payload_size) {
  int rc = -1;
  MQTTClient client = {0};
  MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;
  MQTTClient_message pubmsg = MQTTClient_message_initializer;
  MQTTClient_deliveryToken token = {0};

  MQTTClient_create(&client, opts.address, opts.clientid,
                    MQTTCLIENT_PERSISTENCE_NONE, NULL);
  conn_opts.keepAliveInterval = 60;
  conn_opts.cleansession = 1;
  conn_opts.username = k_username;
  conn_opts.password = CreateJwt(opts.keypath, opts.projectid, opts.algorithm);
  MQTTClient_SSLOptions sslopts = MQTTClient_SSLOptions_initializer;

  sslopts.trustStore = opts.rootpath;
  sslopts.privateKey = opts.keypath;
  conn_opts.ssl = &sslopts;

  unsigned long retry_interval_ms = kInitialConnectIntervalMillis;
  unsigned long total_retry_time_ms = 0;
  while ((rc = MQTTClient_connect(client, &conn_opts)) != MQTTCLIENT_SUCCESS) {
    if (rc == 3) {  // connection refused: server unavailable
      usleep(retry_interval_ms * 1000);
      total_retry_time_ms += retry_interval_ms;
      if (total_retry_time_ms >= kMaxConnectRetryTimeElapsedMillis) {
        printf("Failed to connect, maximum retry time exceeded.");
        exit(EXIT_FAILURE);
      }
```