CODE DEVELOPMENT

```
#include <WiFi.h>
#include <WiFiUdp.h>
#include <PubSubClient.h>
#include <NTPClient.h>
#define WIFISSID "BMS_Students" // Enter WifiSSID here
#define PASSWORD "8147661037" // Enter password here
#define TOKEN "BBUS-9ZOOiHWU50hF9BWOraqWaBFmBACt68" //
#define MQTT_CLIENT_NAME "mymqttclient" // MQTT client Name
// * Define Constants
#define VARIABLE_LABEL "Test_data" // ubidots variable label
#define DEVICE_LABEL "ECG_MONITORING_SYSTEM" // ubidots device label
#define SENSORPIN A0 // Set the A0 as SENSORPIN
Char mqttBroker[] = "industrial.api.ubidots.com";
char payload[10000];
char topic[150];
// Space to store values to send
char str_sensor[10];
char str_millis[20];
double epochseconds = 0;
double epochmilliseconds = 0;
double current_millis = 0;
double current_millis_at_sensordata = 0;
double timestampp = 0;
int j = 0;
/**************
 Auxiliar Functions
**************/
WiFiClient ubidots;
PubSubClient client(ubidots);
WiFiUDP ntpUDP;
```

```
NTPClient timeClient(ntpUDP, "pool.ntp.org");
oid callback(char* topic, byte* payload, unsigned int length) {
 char p[length + 1];
 memcpy(p, payload, length);
 p[length] = NULL;
 Serial.write(payload, length);
 Serial.println(topic);
}
void reconnect() {
 // Loop until we're reconnected
 while (!client.connected()) {
 Serial.println("Attempting MQTT connection...");
// Attemp to connect
 if (client.connect(MQTT_CLIENT_NAME, TOKEN, "")) {
 Serial.println("Connected");
 } else {
 Serial.print("Failed, rc=");
 Serial.print(client.state());
 Serial.println(" try again in 2 seconds");
 // Wait 2 seconds before retrying
 delay(2000);
 }
 }
}
/**************
 Main Functions
**************/
void setup() {
 Serial.begin(115200);
 WiFi.begin(WIFISSID, PASSWORD);
 // Assign the pin as INPUT
```

```
pinMode(SENSORPIN, INPUT);
 Serial.println();
 Serial.print("Waiting for WiFi...");
while (WiFi.status() != WL_CONNECTED) {
 Serial.print(".");
 delay(500);
 }
Serial.println("");
 Serial.println("WiFi Connected");
 Serial.println("IP address: ");
 Serial.println(WiFi.localIP());
 timeClient.begin();
 client.setServer(mqttBroker, 1883);
 client.setCallback(callback);
 timeClient.update();
 epochseconds = timeClient.getEpochTime();
 epochmilliseconds = epochseconds * 1000;
 Serial.print("epochmilliseconds=");
 Serial.println(epochmilliseconds);
 current_millis = millis();
 Serial.print("current_millis=");
 Serial.println(current_millis);
}
void loop() {
 if (!client.connected()) {
 reconnect();
 j = 0;
 }
Serial.print("j=");
 Serial.println(j);
 sprintf(topic, "%s%s", "/v1.6/devices/", DEVICE_LABEL);
```

```
 sprintf(payload, "%s", ""); // Cleans the payload
 sprintf(payload, "{\"%s\": [", VARIABLE_LABEL); // Adds the variable label
 for (int i = 1; i <= 3; i++)
 {
 float sensor = analogRead(SENSORPIN);
 dtostrf(sensor, 4, 2, str_sensor);
 sprintf(payload, "%s{\"value\":", payload); // Adds the value
 sprintf(payload, "%s %s,", payload, str_sensor); // Adds the value
 current_millis_at_sensordata = millis();
 timestampp = epochmilliseconds + (current_millis_at_sensordata - current_millis);
 dtostrf(timestampp, 10, 0, str_millis);
 sprintf(payload, "%s \"timestamp\": %s},", payload, str_millis); // Adds the value
 delay(150);
 }
float sensor = analogRead(SENSORPIN);
 dtostrf(sensor, 4, 2, str_sensor);
 current_millis_at_sensordata = millis();
 timestampp = epochmilliseconds + (current_millis_at_sensordata - current_millis);
 dtostrf(timestampp, 10, 0, str_millis);
 sprintf(payload, "%s{\"value\":%s, \"timestamp\": %s}]}", payload, str_sensor, str_millis);
 Serial.println("Publishing data to Ubidots Cloud");
 client.publish(topic, payload);
 Serial.println(payload);
 // client.loop();
}
```