#include <ESP8266WiFi.h>

#include <PubSubClient.h>

#include <ATT\_IOT.h> //AllThingsTalk IoT library

#include <SPI.h> //required to have support for signed/unsigned long type..

char deviceId[] = "EBQu7B7GhoO2aKO9fuWMhy7P"; // Your device id comes here

char clientId[] = "sushmapedineedi\_o3cmFxNq"; // Your client id comes here;

char clientKey[] = "mkRHrL4Y"; // Your client key comes here;

const char\* ssid = "sush";

const char\* password = "sushma13579";

ATTDevice Device(deviceId, clientId, clientKey); //create the object that provides the connection to the cloud to manager the device.

#define httpServer "api.AllThingsTalk.io" // HTTP API Server host

#define mqttServer httpServer // MQTT Server Address

char j;

// Define PIN numbers & id's for assets

int Digitalsensor = D0; // Digital Actuator is connected to pin D0 on nodemcu -> built in led

int DigitalsensorId = 1; // the pin numbers go above 9, so we use a separate id for the assets

//required for the device

void callback(char\* topic, byte\* payload, unsigned int length);

WiFiClient ethClient;

PubSubClient pubSub(mqttServer, 1883, callback,ethClient);

void setup()

{

// pinMode(Digitalsensor, INPUT); // initialize the digital pin as an output.

// pinMode(D5,OUTPUT);

Serial.begin(9600); // init serial link for debugging

setup\_wifi();

delay(1000); //give the Ethernet shield a second to initialize:

while(!Device.Connect(&ethClient, httpServer)) // connect the device with the IOT platform.

Serial.println("retrying");

Device.AddAsset(DigitalsensorId, "obstacle status", "Products discription", false, "string"); // Create the Digital Actuator asset for your device

while(!Device.Subscribe(pubSub)) {} // make certain that we can receive message from the iot platform (activate mqtt)

Serial.println("retrying");

}

void setup\_wifi() {

delay(10);

// We start by connecting to a WiFi network

Serial.println();

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

}

void loop()

{

if(Serial.available()>0)

{

while(Serial.available()==0);

j=Serial.read();

check();

}

Device.Process();

}

void check()

{

if(j=='1')

{

Device.Send("SAFE.....", DigitalsensorId);

//digitalWrite(D5,HIGH);

}

else if(j=='0')

Device.Send("WARNING!.....", DigitalsensorId);

}

// Callback function: handles messages that were sent from the iot platform to this device.

void callback(char\* topic, byte\* payload, unsigned int length)

{

/\* String msgString;

{ //put this in a sub block, so any unused memory can be freed as soon as possible, required to save mem while sending data

char message\_buff[length + 1]; //need to copy over the payload so that we can add a /0 terminator, this can then be wrapped inside a string for easy manipulation

strncpy(message\_buff, (char\*)payload, length); //copy over the data

message\_buff[length] = '\0'; //make certain that it ends with a null

msgString = String(message\_buff);

msgString.toLowerCase(); //to make certain that our comparison later on works ok (it could be that a 'True' or 'False' was sent)

}

int\* idOut = NULL;

{ //put this in a sub block, so any unused memory can be freed as soon as possible, required to save mem while sending data

int pinNr = Device.GetPinNr(topic, strlen(topic));

// Serial.print("Payload: "); //show some debugging

// Serial.println(msgString);

// Serial.print("topic: ");

// Serial.println(topic);

if (msgString == "false") {

Serial.println("a");

digitalWrite(relay,HIGH);

}

else if (msgString == "true") {

Serial.println("b");

digitalWrite(relay,LOW);

}

if (pinNr == DigitalActuatorId)

{

if (msgString == "false") {

digitalWrite(DigitalActuator, HIGH); //change the actuator status to false -> the pin appears to be inversed, so set high

idOut = &DigitalActuatorId;

}

else if (msgString == "true") {

digitalWrite(DigitalActuator, LOW); //change the actuator status to true -> the pin appears to be inversed, so set high

idOut = &DigitalActuatorId;

}

}

}

if(idOut != NULL) //Let the iot platform know that the operation was succesful

Device.Send(msgString, \*idOut); \*/

}