//WIFI and ADC2 Channels are not working together hence using ADC1 channels

#include <stdlib.h>

#include <string.h>

#include <WiFi.h>

#include <PubSubClient.h>

#define ThermistorPin 35

#define LDR\_PIN 33

#define SOIL\_MOISTURE\_PIN 34

#define MOTOR 12

#define R1 10000

#define C1 (float)1.009249522e-03

#define C2 (float)2.378405444e-04

#define C3 (float)2.019202697e-07

// Update these with values suitable for your network.

const char\* ssid = "hidden";

const char\* password = "qwerty12";

const char\* mqtt\_server = "m15.cloudmqtt.com";

#define mqtt\_port 16951

#define MQTT\_USER "mnkobdft"

#define MQTT\_PASSWORD "xOftMxYCCo91"

#define MQTT\_SERIAL\_PUBLISH\_LIGHT "stechiez/agree/light"

#define MQTT\_SERIAL\_PUBLISH\_TEMPERATUE "stechiez/agree/temp"

#define MQTT\_SERIAL\_PUBLISH\_SOIL "stechiez/agree/soil"

#define MQTT\_SERIAL\_RECEIVER\_MOTOR "stechiez/agree/motor"

#define MQTT\_SERIAL\_RECEIVER\_MOTOR\_STATUS "stechiez/agree/mstatus"

WiFiClient wifiClient;

PubSubClient client(wifiClient);

float avg[3]={0,0,0};

char light\_array[7];

char soil\_array[7];

char temp\_array[7];

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(".");

}

randomSeed(micros());

Serial.println("");

Serial.println("WiFi connected");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

}

void reconnect() {

// Loop until we're reconnected

while (!client.connected()) {

Serial.print("Attempting MQTT connection...");

// Create a random client ID

String clientId = "ESP32Client-";

clientId += String(random(0xffff), HEX);

// Attempt to connect

if (client.connect(clientId.c\_str(),MQTT\_USER,MQTT\_PASSWORD)) {

Serial.println("connected");

//Once connected, publish an announcement...

client.publish("/icircuit/presence/ESP32/", "hello world");

// ... and resubscribe

client.subscribe(MQTT\_SERIAL\_RECEIVER\_MOTOR);

} else {

Serial.print("failed, rc=");

Serial.print(client.state());

Serial.println(" try again in 5 seconds");

// Wait 5 seconds before retrying

delay(3000);

}

}

}

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

Serial.println("-------new message from broker-----");

Serial.print("channel:");

Serial.println(topic);

Serial.print("data:");

Serial.write(payload, length);

Serial.println();

if(\*payload == '0')

{

digitalWrite(MOTOR, 0);

}

else if(\*payload == '1')

{

digitalWrite(MOTOR, 1);

}

}

void publishSerialData(const char \*pub\_str,const char \*serialData){

if (!client.connected()) {

reconnect();

}

client.publish(pub\_str, serialData);

}

void setup() {

Serial.begin(115200);

pinMode(MOTOR, OUTPUT);

Serial.setTimeout(500);// Set time out for

setup\_wifi();

client.setServer(mqtt\_server, mqtt\_port);

client.setCallback(callback);

reconnect();

digitalWrite(MOTOR, 0);

}

//NTC Temparature Reading and Processing

float getTemperature(void)

{

static int avgArrayIndex=0;

int Vo;

float logR2,R2, T, Tc;

Vo = analogRead(ThermistorPin);

R2 = R1 \* (4096.0 / (float)Vo - 1.0);

logR2 = log(R2);

T = (1.0 / (C1 + C2\*logR2 + C3\*logR2\*logR2\*logR2));

Tc = T - 273.15;

avg[avgArrayIndex++] = Tc;

if(avgArrayIndex > 2)

avgArrayIndex = 0;

Tc = (avg[0] + avg[1] + avg[2])/3;

return Tc;

}

float getMoisturePercentage(void)

{

float moisture\_percentage;

int sensor\_analog;

sensor\_analog = analogRead(SOIL\_MOISTURE\_PIN);

moisture\_percentage = ( 100 - ( (sensor\_analog/4096.00) \* 100 ) );

moisture\_percentage = (float)moisture\_percentage;

return moisture\_percentage;

}

float getLightPercentage(void)

{

int ldrRawVal;

float percentage;

ldrRawVal = analogRead(LDR\_PIN);

percentage = ((float)((ldrRawVal\*100)/4096));

// percentage = 100 - percentage;

return percentage;

}

void loop() {

float lightpercentage = getLightPercentage();

float temp = getTemperature();

float soilMoisturePer = getMoisturePercentage();

String temp\_s(temp);

String soilPer\_s(soilMoisturePer);

String lightPer\_s(lightpercentage);

bool mstatus = digitalRead(MOTOR);

String mstatus\_s(mstatus);

if(soilMoisturePer > 80.00)

{

digitalWrite(MOTOR, 0);

}

publishSerialData(MQTT\_SERIAL\_RECEIVER\_MOTOR\_STATUS,mstatus\_s.c\_str());

delay(2500);

publishSerialData(MQTT\_SERIAL\_PUBLISH\_SOIL,soilPer\_s.c\_str());

delay(250);

publishSerialData(MQTT\_SERIAL\_PUBLISH\_TEMPERATUE,temp\_s.c\_str());

delay(250);

publishSerialData(MQTT\_SERIAL\_PUBLISH\_LIGHT,lightPer\_s.c\_str());

delay(250);

client.loop();

Serial.print("Temperature: ");

Serial.print(temp);

Serial.print(" C, LDR: ");

Serial.print(lightpercentage,0);

Serial.print("%");

Serial.print(" Moisture Percentage = ");

Serial.print(soilMoisturePer);

Serial.println("%");

// delay(400);

}