DEVELOPER AND USER DOCUMENTATION

Team_5

Astitva Gupta 2018101085

Jalees Jahanzaib 2018101001

Trusha Sakharkar 2018101093

1. INTRODUCTION

1.1. Objective

This document provides comprehensive guidelines and step-by-step instructions for working with the project: **Audio Volume Control Based on Distance of Students.**

2. SCOPE

2.1. Expected Audience

This document is intended to be used in the classrooms for the control of the volume of the speakers employed there based upon the number of students sitting in the class. When the number of students is less, High audio volume is both disturbing to teachers and students and waste of electricity.

2.2. Components provided along with the project

In this project, the following items are included:

a) A box with the hardware circuit with the code already loaded into it.
b) Sensor Nodes.
c) Connecting wires to spread the nodes across the classroom.
d) Link to the source code

2.3. Limitations/Constraints

a) There is a noticeable lag in the actual sound of the teacher or whoever is speaking into the Mic and the audio produced by the speaker. This is due to the lag caused by Bluetooth latency in transmission of sound received on the phone.
b) In continuation of the previous point, the latency also poses yet another problem. The sound of the speaker is again received at the phone which creates unwanted noise.

2.4. Deliverables

This project mainly aims to harvest distance of the closest student from the nodes, continuously and send them over through a server. Also, the distance obtained controls the volume of speaker present in the classroom.

3. PRODUCT OPERATIONAL REQUIREMENTS

3.1. Hardware requirements

a) Operating Environment

Any PC with any operating system that supports Arduino software is required for building the project from scratch and loading it to the board.

b) ESP 32 NodeMCU Development Board

This board contains the processor for running the code.

c) Ultrasonic sensor to measure nearest student distance

Here we have used the sound sensor: **SR04.**
Documentation of sensor: https://elecfreaks.com/estore/download/EF03085-HC-SR04_Ultrasonic_Module_User_Guide.pdf

d) Server

Here we upload the values detected by the sensor on:

OneM2M server hosted by the college: http://onem2m.iiit.ac.in/

e) Other components for making the circuit

This includes connecting wires to complete the circuit.

f) Components required for deployment

1. **Power:** This includes a power source (any power outlet would work), an adapter and a USB type B wire to connect the ESP board to the power source

2. **WiFi:** The setup requires WiFi connection (preferably a steady one that works 24*7) in order to send the values to the server for analytics.

3.2. Software requirements

1. Arduino software along with the drivers required to work with the Node MCU, and one-M2M server
2. The code (shipped along with the project)

3. An Android phone (Android 5.0 and above) with the following two apps installed:
    1. **Bluetooth Loudspeaker:** (Android)
    https://play.app.goo.gl/?link=https://play.google.com/store/apps/details?id=wimlog.com.myandroidtest4&ddl=1&pcampaignid=web_ddl_1
    2. **ESW Specific app** (Android)**:**
    https://github.com/Astitva13/ESWProjectBluetooth/releases
    3. **ESW Plotting Graph** (Java)**:**
    https://github.com/Astitva13/ESWPlotting/releases

4. ASSEMBLING THE SYSTEM

4.1. Hardware component

The circuit consists of the ESP32 board and five ultrasonic sensors connected using wires.

The connections of all the five sensors correspond the **Pins 22, 23, 25, 26, 33** and **Pin 27** for trigger in the NodeMCU. These will be already connected to the NodeMCU, thus all you need to do is to connect is the **USB cable** to the ESP32 NodeMCU.

4.2. Software component (Compiling and loading the code onto the board)

- Connect the laptop to the hardware setup (built above)
- Open Arduino and download the drivers required for uploading the code on the board. Also download the required header files. (refer the documentation specific to your operating system)

- Copy paste the following code into a new file:

```
#include "BluetoothSerial.h"
#include "WiFi.h"
#include "HTTPClient.h"
BluetoothSerial ESP_BT;
//// defines pins numbers
///char apiKey[] = "4UWVCVL1KQAZ0H45";     //  Enter your Write API key from ThingSpeak
//unsigned long int channel=883330;
char* wifi_ssid = "esw-m19@iiith";
char* wifi_pwd = "e5W-eMai@3!20hOct";
String cse_ip = "onem2m.iiit.ac.in";
String cse_port = "443";
String server = "http://"+cse_ip+":"+cse_port+"/~/in-cse/in-name/";
String createCI(String server, String ae, String cnt, String val)
{
  HTTPClient http;
  http.begin(server + ae + "/" + cnt + "/");
  Serial.println(server + ae + "/" + cnt + "/");
  http.addHeader("X-M2M-Origin", "admin:admin");
  http.addHeader("Content-Type", "application/json;ty=4");
  http.addHeader("Content-Length", "100");
  http.addHeader("Connection", "close");
  int code = http.POST("{\"m2m:cin\": {\"con\":\"" + String(val) + "\"}}");
  http.end();
  Serial.println(code);
  delay(300);
}

//const char* server = "api.thingspeak.com";
//WiFiClient client;
int MAXXX=0;
const int trigPin = 27;
const int echoPin1 = 22;
const int echoPin2 = 23;
const int echoPin3 = 25;
const int echoPin4 = 26;
const int echoPin5 = 33;

const int pins[10]={22,23,25,26,33};


// defines variables
long duration;
int distance,count=0,mini;
//int incoming;
//int LED_BUILTIN = 2;


void setup() {
  Serial.begin(115200); //Start Serial monitor in 9600
  delay(10);
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();
  delay(100);
  WiFi.begin(wifi_ssid, wifi_pwd);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.println("Connecting to WiFi..");
  }
  Serial.println("Connected to the WiFi network");
  ESP_BT.begin("AJT Audio Controller"); //Name of your Bluetooth Signal
  Serial.println("Bluetooth Device is Ready to Pair");
  //pinMode (LED_BUILTIN, OUTPUT);//Specify that LED pin is output

  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin1, INPUT); // Sets the echoPin as an Input
  pinMode(echoPin2,INPUT);
  pinMode(echoPin3,INPUT);
  pinMode(echoPin5,INPUT);
  pinMode(echoPin4,INPUT);// Sets the echoPin as an Input
  Serial.begin(115200); // Starts the serial communication
  //ThingSpeak.begin(client);

}

void loop() {
  if(WiFi.status() != WL_CONNECTED) {
    Serial.println("Connecting to Wifi " + String(wifi_ssid) + " with password " + String(wifi_pwd));
    WiFi.begin(wifi_ssid, wifi_pwd);
    while(WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
    }
  }
  mini=5000;
  for (int i=0;i<=4;i++) {
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    // Sets the trigPin on HIGH state for 10 micro seconds
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    // Reads the echoPin, returns the sound wave travel time in microseconds
    duration = pulseIn(pins[i], HIGH);
    // Calculating the distance
    distance= duration*0.034/2;
    if(distance<mini && distance!=0)
    mini=distance;
    MAXXX=mini;
    // Prints the distance on the Serial Monitor
    delay(100);
    Serial.print("Distance for  Sensor ");
    Serial.print(i+1);
    Serial.print(" : ");
    Serial.println(distance);

  }

  Serial.print("Minimum Distance : ");
  Serial.println(mini);
//  if (ESP_BT.available()) //Check if we receive anything from Bluetooth
  {
    if(mini>=100   )
          ESP_BT.println(1);
      else
          ESP_BT.println(2);
            //ESP_BT.println(distance);

  }
  //delay(2000);//1000*60*5=300000
  //String xx=    String( 170  + rand()%9);
  String sensor_string =String(MAXXX);
  createCI(server, "Team5_Audio_volume_control_based_on_distance_of_students", "node_1", sensor_string);
  delay(5000);
  }
```

- Change the WiFi SSID and password to that of your network.
- Also update the thingspeak channel number and API key.
- Compile and upload it. If in case, you face any issues while uploading it, press the reset button on the board during the loading time.
- Open the serial monitor. You should be able to see WiFi connected, and the values read by the board, and sent to the server.
- Your setup is now ready for deployment.

NOTE

In an ideal case, the project should contain the hardware setup with the code uploaded in it. If it does not, or the circuit delivered is damaged, then get a new board and sensor and construct the hardware circuit from scratch and upload the code onto it. Also you might need to repartition the NodeMCU to support WiFi and Bluetooth simultaneously.

4.3. Deployment

1. Insert the hardware component onto a box, that would allow sound to pass through it.
2. Place it in a place where the WiFi signal is strong enough.
3. Connect it to the power supply.
4. Now the board shall send values to the servers as required. Leave the setup there for several days in order to get the data for a reasonable number of days.
5. Now open the Android ESW app. Then select the option "Connect to AJT Audio Controller" if prompted otherwise it will connect to it automatically connect if present nearby.
6. Then connect to the classroom speakers. After this open the Bluetooth Loudspeaker app and turn option to send data.

4.4. Analytics

The graph is made scrapping data from one m2m server. In the graph you can see the minimum distance values of students plotted against the date and time.

1. Observing the graph, we can find out the following things:

    a) The date and time when there were maximum number of students by seeing the low values.
    b) The average number of students in that classroom.
2. Using this information, we can understand, how much the volume level should be kept for specific classrooms, that too dynamically.

4. END USER FEATURES

This includes both packaged and non-packaged features, that can be implemented using this project.

4.1. Measuring the extent to which a classroom is occupied

Referring to the data on the one m2m server we can conclude over time the most and least attended classes. This can help in many situations to increase or decrease the number of seats available for a course accordingly and also get a analytic of which course is among favourite of students.

4.2. Noise Control

The excess noise in any room causes reverberation. Reducing the volume when not needed at high level helps reducing the reverberation and efficient listening.

4.3. Energy Efficiency

When the speakers work at full volume, they require high amount of power which is indeed wasted when the number of student sis low. By implementing this project the power input can be cut and energy can be saved.

5. FAQ

5.1. While uploading the data to the server (onem2m), it shows error code 400

Use the esw WiFi connection (available at certain places in campus) and make sure that the node should have a registered MAC address with the network.