# PipeOrgan_wifi_controller
First version of a wifi pipe organ controller! try it out &lt;3

Project: Pipe Organ WIFI Controller

This project allows users to control the opening and closing of a solenoid valve which manages the airflow into a pipe organ. The system is comprised of a Python-based user interface (UI), an ESP32 connected via serial communication, and additional ESP32 devices that receive data through ESP-NOW (WiFi).

System Overview

The Python script provides a UI for the user to create a grid representing the solenoid valve states (open/close). Users can input the number of rows and columns for the grid, as well as a tempo value that affects the timing of solenoid valve actions.

Once the user has configured the grid and tempo, the Python script constructs an Open Sound Control (OSC) message containing the grid and tempo data. This message is sent over a serial connection to the first ESP32 device. The first ESP32 parses the incoming OSC message and forwards the parsed data to the other ESP32 devices using the ESP-NOW protocol.

Each receiving ESP32 device processes the incoming data and controls the solenoid valve's state based on the received data. The valve states are managed according to the user-defined grid and tempo values.

Communication Flow

Python UI receives user input (grid configuration and tempo).
Python script constructs an OSC message with grid and tempo data.
OSC message is sent over a serial connection to the first ESP32.
First ESP32 parses the OSC message and forwards data to other ESP32 devices using ESP-NOW.
Receiving ESP32 devices control the solenoid valve based on received data (grid configuration and tempo).
User Interaction
Through the UI, users can create a grid representing the solenoid valve states (open/close) and input a tempo value. The grid consists of black and white cells, where black represents an open state and white represents a closed state. Users can toggle cell states by clicking on them. Once the desired grid configuration and tempo are set, the user can click the "Save" button to send the data to the ESP32 devices and control the solenoid valve accordingly.

Building the System

To build the pipe organ controller system, you will need the following components:
1.Python installed (preferably version 3.6 or later)
2.Two or more ESP32 microcontrollers

3.Solenoid valves for controlling airflow into the pipe organ
4.USB cable for connecting the sending ESP32 to the computer (for serial communication)
5.Jumper wires and a breadboard for connecting the ESP32 to the solenoid valve
6.A 5v for the ESP32 connected to the solenoid (check the valve's specifications)

Steps to Build the System

1.Set up the Python environment: Install the required Python libraries for the project. The script requires pyserial, python-osc, and tkinter. You can install these libraries using pip:
2.Flash the ESP32 devices: Upload the provided ESP32 code to each of your ESP32 microcontrollers using the Arduino IDE or a similar tool. Make sure to update the MAC addresses in the "ESP32 connected via Serial" code to match the addresses of your receiving ESP32 devices. Also, update the peerMacA, peerMacB, etc., in the "ESP32 receivers" code to match the addresses of your receiving ESP32 devices.
3.Connect the ESP32 devices: Connect the first ESP32 to your computer using a USB cable. This will allow serial communication between the Python script and the ESP32. Connect the receiving ESP32 devices to the solenoid valve(s) using jumper wires and a breadboard. Make sure to connect the appropriate GPIO pin (as specified in the "ESP32 receivers" code) to the control pin of the solenoid valve. Additionally, connect the power and ground pins of the solenoid valve to a suitable power source.
4.Run the Python script: Execute the provided Python script on your computer. This will open the user interface, allowing you to create a grid, set a tempo, and control the solenoid valve.
5.Configure and control the system: Use the UI to set the desired grid configuration and tempo. Click the "Save" button to send the data to the ESP32 devices and control the solenoid valve accordingly.

Dashed diagram of how the system communicates:

```
+------------+    Serial     +------------+    ESP-NOW    +------------+
|   Python   | ------------> |  ESP32 #1  | ------------> |  ESP32 #2  |
|   Script   |               | (Connected)|               | (Receiver) |
+------------+               +------------+               +------------+
                                                             |
                                                             |
                                                             v
                                                         +------------+
                                                         |  Solenoid  |
                                                         |   Valve    |
                                                         +------------+
```

1. Python Script: The Python script runs on a computer and provides a user interface for creating a grid, setting a tempo, and controlling the solenoid valve.
2. Serial Communication: The Python script communicates with the first ESP32 (ESP32 #1) connected via USB using serial communication. The script sends OSC messages

containing grid data and tempo information to the ESP32.
3. ESP32 #1 (Connected): This ESP32 receives the OSC messages from the Python script via serial communication. It then sends the received data to the receiving ESP32 devices (ESP32 #2) using ESP-NOW communication.
4. ESP-NOW Communication: The first ESP32 communicates with the receiving ESP32 devices (ESP32 #2) wirelessly using the ESP-NOW protocol. It sends the grid data and tempo information to the receivers.
5. ESP32 #2 (Receiver): The receiving ESP32 devices receive the data sent by ESP32 #1 and control the solenoid valve accordingly.
6. Solenoid Valve: The solenoid valve controls the airflow into the pipe organ based on the signals received from the receiving ESP32 devices.

a zoom in on the ESP32 #2...#N (as many as you would like, but be aware that they n is the max amount of rows in your UI)

```
                     3.3V GND ----------------------------->(+) (-)
                       |  |                                  |   |
+---------+  5V USB  +---------+      +------------+       +----------+
| Power   | -------> | ESP32   |      | Relay      |<-----(+)| Step-Up |
| Supply  |          | #N      |      | Module     |-----|   |Converter|
+---------+          +---------+      +------------+     |   +----------+
                       |  |  |          |  |   |         |     (-)
                     GND VCC IO22 -> GND VCC IO22        |      |
                                     (-) (+)             |      |
                                                         v      v
                                                     +------------+
                                                     | Solenoid   |
                                                     |  Valve     |
                                                     +------------+
```

1.Power Supply: The ESP32 #2 is powered with a 5V USB cable.
2.ESP32 #2 (Receiver): This ESP32 receives data wirelessly from the first ESP32 (Connected to the computer via serial). It is connected to a relay module and a step-up converter.
3.Relay Module: The relay module is connected to the ESP32 using GND, VCC, and IO22 pins. It controls the opening and closing of the solenoid valve. One output is connected to the step-up converter and the other to the solenoid.
4.Step-Up Converter: The step-up converter is connected to the 3.3V and GND pins of the ESP32. It transforms the 3.3V coming from the ESP32 to 12V+. The positive output (Out+) is connected to the relay, and the negative output (Out-) is connected to the solenoid valve.
5.Solenoid Valve: The solenoid valve is connected to the relay module and the step-up converter. One output from the relay is connected to the Out+ of the step-up converter, and the other output is connected to the solenoid valve. The Out- from the step-up converter is connected to the solenoid valve.

Hope it is clear enough! write me if you want some pictures to see how everythong

```
is connected :)
```