Smart Alert System for Videx Intercom

Oliver Butler
University of Southampton
ob3g21@soton.ac.uk

Abstract

This project explores connecting a "dumb" Videx 6272N intercom substation to the internet for remote access and IoT connection. This was enabled using ESP32-C3 microcontrollers to capture signals from the intercom and send messages using MQTT to a central broker, notifying subscribed devices and applications. A discord bot receives these messages to alert any servers it is installed on and request for the door to be opened.

1. Introduction

1.1. Background

Intercom systems are a prevalent communication infrastructure used in residential, commercial and institutional settings. They are widely used in apartment builds to facilitate communication between visitors and residents for access control. Each system primarily consists of a master station located at the entrance of an apartment complex and substation units (slaves) installed in each block. Visitors can access master stations to communicate with a corresponding substation; residents use substations to decide whether the visitor is allowed access. Substations can be equipped with a microphone and speaker to communicate between either party, but are not used in all systems.

1.2. Problem Statement

Traditional intercom systems, predominantly those designed with wired infrastructure, face significant limitations stemming from their inability to be wirelessly controlled. They lack the flexibility and convenience offered by wireless connectivity, hindering remote access and integration with modern smart home technologies.

Numerous modern smart intercom systems exist but are expensive to install or replace in large apartment complexes. The cost can differ between £1,000 to £10,0000 depending on the units installed and how many, excluding installation and maintenance fees [1]. Furthermore, they commonly suffer from subscription services, security vulnerabilities and possess limited product lifespans with discontinued support rendering devices obsolete. Furthermore, much older intercoms usually cannot be individually upgraded, meaning residents are unable to upgrade a substation without replacing the whole intercom system.

Neglecting to upgrade outdated intercom systems can disproportionately affect individuals with disabilities. Individuals who are hard of hearing may be unable to be adequately notified when someone rings their system. The lack of remote access or advanced features in traditional systems can also hinder their ability to communicate effectively or independently access the building, exacerbating existing barriers to accessibility.

1.3. Proposed Plan and Goals

This project plans to design and implement a system which can enable smart features for a Videx 6272N substation. The overall design should be relatively cheap and simple to implement allowing a range of compatibility with smart home devices and other online applications.

2. Research

2.1. Background Research

Wired intercom systems typically transfer data between the master and slave in two configurations.

In a multi-wire configuration, wires are routed to a connection terminal for each attribute. For example, one for playing an alert noise, one for voice in, one for voice out and one for controlling the door. This configuration allows for more direct connections between devices, potentially resulting in better signal quality and reliability. However, it may require more wiring, which can increase installation complexity and cost, especially for larger systems. These systems are much easier to modify as you only need to send/receive on each wire for the desired purpose.

In a bus configuration, a data bus is used to send all information on one wire. This topology streamlines wiring requirements, particularly beneficial for larger systems. However, it can introduce challenges related to signal quality and susceptibility to interference. These systems are sig-

nificantly harder to modify as the composition of messages is not widely available and decoding would be outside the project's scope.

2.2. Similar Projects

Companies have made products to create a universal controller to manage a substation. The switchbot allows wireless control for pressing buttons on an intercom [2]. Being easy to set up and relatively cheap at £30, it would enable people to access the apartment remotely. However, it needs a hub to be controlled outside the network which would cost an additional £35. The device also only allows buttons to be pressed on the machine and won't detect when the substation is called.

Typically, hobbyists use microcontrollers connected to the motherboard to retrieve and send signals when the intercom is called. As previously explained, substations using a multi-wire configuration are relatively simple to modify primarily by monitoring each connection terminal for the respective in/output [3]. Single-wire configurations, where data is transferred over a bus, typically exploit a buzzer or speaker which activates once called, sending an alert to the user. This is done by soldering directly onto the buzzer and monitoring the voltage, using a microphone to monitor noises or an accelerometer to detect vibrations. The latter two methods are easier to set up, but less reliable as outside sound can trigger messages being sent. Activating the door is also significantly harder; simple methods usually entail using another microcontroller to press the button manually, similar to the switchbot.

3. Design

3.1. Current System and Limitations

Unfortunately, the intercom system for this project is the Videx 6272N, as it is installed in my apartment. As seen in Figure 1, all data is transferred over a data bus; therefore making it challenging to determine when the system is alerted. Additionally, this intercom system features a telephone connected with an RJ-11 telephone wire for audio in/out and a screen for video input, but will not be used due to the lack of equipment to capture the information and time. The video output is also proprietary with no documentation. Furthermore, as I was only permitted to do this project by my landlord assuming everything was assembled back to the original state, directly soldering anything onto the circuit board should be a last resort.

3.2. Planning, Hardware Requirements and Communication Protocol

Luckily, detecting when the system is alerted should be relatively easy. When the substation is called, a ringtone can be heard inside the connected telephone. This is an

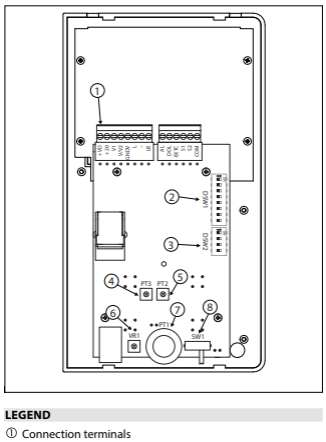


Figure 1. Videx 6272N board. The connection terminal, labelled a 'L', signifies the Bus Line where data is transferred. Sourced from [4].

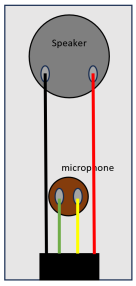


Figure 2. Phone Internal. Four wires connect to the RJ-11 port, two connected to the speaker and two to the microphone. The dimensions are 17.5cm x 3.8cm. Outputs roughly 1.7V AC when powered at high volume.

added benefit as previous implementations required having the substation exposed to connect the microcontroller as the speaker was built directly onto the motherboard. Figure 2 shows the wiring of the telephone. Therefore to detect when the system is called, a microcontroller should connect to the main speaker and send a message when activated. This would require soldering but with a very low risk of damaging components. There is also ample space to hide the device inside, although no easy way to power it without drilling a hole through the plastic.

To open the door, the best technique without directly soldering onto the board seems to be to use a solenoid powered by a microcontroller to press the button directly. This is a crude solution, but relatively simple to implement, similar to the Switchbot.

On hand, I currently have access to a couple of ESP32-C3 microcontrollers and one solenoid. These will be used as they are very small and can fit inside the phone casing. As the microcontrollers will be set at home near a

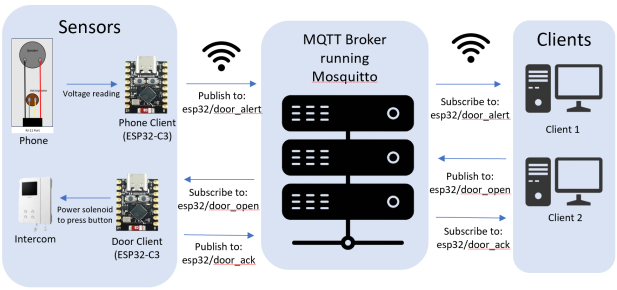


Figure 3. Overall Achitecture. The broker, run on a local server, controls messages between the sensors and clients. An ESP32-C3 microcontroller will be connected to the telephone speaker, publishing a message to "esp32/door_alert" when turned on, notifying all clients. Clients can then request to open the door by publishing a message to "esp32/door_open" activating a solenoid powered by an additional ESP32-C3. An acknowledgement will be sent back on "esp32door_ack" to notify clients the door was opened.

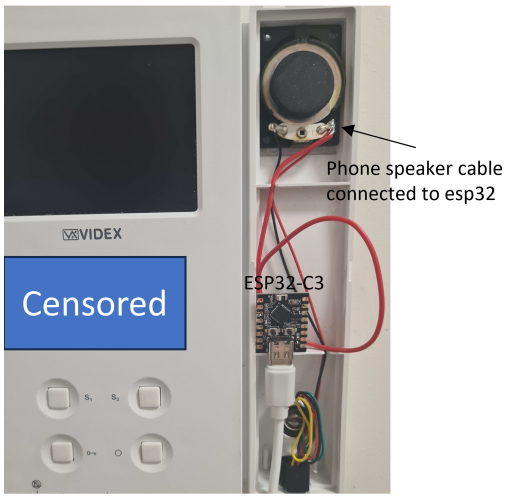


Figure 4. Internal phone wiring to ESP32-C3

WiFi access point, there is no reason to use any other protocol such as Bluetooth or LoRa. All hardware will be connected to a power socket so low-powered protocols like 6LoWPaN, Bluetooth LE or Zigbee are unnecessary. The device also only supports 2.4GHz WiFi and Bluetooth5LE natively, therefore all data will be transferred over WiFi.

To allow for seamless integration with other applications, MQTT will be used for transferring messages between the clients and ESP32-C3s. This will allow multiple clients to independently send/receive messages and expand upon the system with their application of choice. The broker will be run on my local apartment server but could be replaced with a Raspberry Pi or desktop PC. Clients can decide how to send and receive the information; this could be connected to an SMS server, Home Assistant or other messaging services.

A Discord Bot will be implemented to allow clients to receive the MQTT messages and send requests. This was chosen as all my flatmates use Discord, it is free to correct bots for, unlike SMS or other messaging services and integrates permissions for security. A full breakdown of the architecture can be seen in Figure 3.

4. Implementation

4.1. Hardware Wiring and Communication

To facilitate communication between the sensors and clients, an MQTT broker was established using Mosquitto, an open-source MQTT broker. This ran inside the local network on a server. All clients and sensors connected to the broker were given usernames and passwords to protect against unauthorised access. Currently, the server is accessible solely within the local network, as it has yet to be port-forwarded. Since all sensors and applications ran on

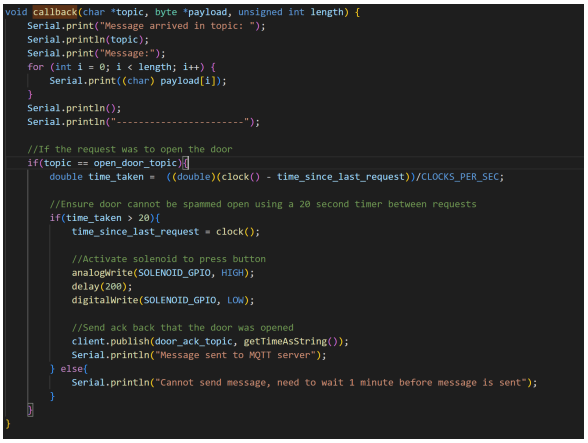
```
void loop() {
  //write_to_board_led();
  delay(200);

  int phone_state = analogRead(PHONE_GPIO);
  Serial.println(phone_state);

  //If the phone is on/plays a loud alarm sound
  if(phone_state < 2000){
    //Get the time between the last sent MQTT message
    double time_taken =  ((double)(clock() - time_since_last_request))/CLOCKS_PER_SEC;
    //If the time > 60 seconds
    if(time_taken > 60){
      //Publish a new message to esp32/Door_Alert
      time_since_last_request = clock();
      client.publish(door_opened_topic, getTimeAsString());
      Serial.println("Message sent to MQTT server");
    } else{
      Serial.println("Cannot send message, need to wait 1 minute before message is sent");
    }
  }


  client.loop();
}
```

Figure 5. ESP32-C3 Phone Client code for reading the speaker voltage and sending MQTT messages

the local network, it was deemed unnecessary but could be changed if needed.

Detection of calls to the substation was accomplished through an ESP32-C3 microcontroller powered by a nearby outlet and connected to the positive speaker solder joint seen in Figure 4. Note the telephone casing can be closed if a hole is drilled to pass the USB-C cable through. The analogue input is wired to GPIO 2; readings typically rested at 4095 when the speaker was inactive with random small changes between 3000-4000. During incoming calls when the speaker emitted noise, readings fluctuated within 0-2000. This triggers the publication of an MQTT message to the esp32/Door_Alert topic containing the timestamp of the request. The code for this can be seen in Figure 5.

For opening the door, another ESP32-C3 monitored mes-



```
void callback(char *topic, byte *payload, unsigned int length) {
    Serial.print("Message arrived in topic: ");
    Serial.println(topic);
    Serial.print("Message:");
    for (int i = 0; i < length; i++) {
        Serial.print((char) payload[i]);
    }
    Serial.println();
    Serial.println("-----------------------");

    //If the request was to open the door
    if(topic == open_door_topic){
        double time_taken =  ((double)(clock() - time_since_last_request))/CLOCKS_PER_SEC;

        //Ensure door cannot be spammed open using a 20 second timer between requests
        if(time_taken > 20){
            time_since_last_request = clock();

            //Activate solenoid to press button
            analogWrite(SOLENOID_GPIO, HIGH);
            delay(200);
            digitalWrite(SOLENOID_GPIO, LOW);

            //Send ack back that the door was opened
            client.publish(door_ack_topic, getTimeAsString());
            Serial.println("Message sent to MQTT server");
        } else{
            Serial.println("Cannot send message, need to wait 1 minute before message is sent");
        }
    }
}
```

Figure 6. Microcontroller code for powering the solenoid and receiving/sending MQTT messages

sages sent to the esp32/Open_Door topic. When a message is received, a solenoid is activated to press the bottom left key button seen in Figure 4, opening the door. Since the solenoid can't be directly powered by the microcontroller, the ESP32-C3 controls a switch that connects the solenoid to a power supply.

Unfortunately, when wiring and testing, the solenoid was damaged and I was unable to get it working. Due to the time constraints, another solenoid could not be obtained. No similar hardware was also on hand either. Nevertheless, the code depicted in Figure 6 outlines the second microcontroller functionality, powering the solenoid if the last request occurred over a minute ago. This would open the door to which an acknowledgement is published to esp32/Door_ACK.

4.2. Discord Interface

A Discord bot was developed to bridge the interface between the users and microcontrollers. Two threads are necessary to run — one for receiving MQTT messages and the other for controlling the bot. The bot uses a dedicated channel "#door-alert" for all servers it is connected to, listening and sending only on that channel. Upon receiving a message from the esp32/Door_Alert topic, the Discord bot notifies users within the channel. Additionally, users may request to open the door using the "!opendoor" command, which in turn sends a message to the esp32/Open_Door topic. If an acknowledgement is sent back, another message is again sent to notify users the door was opened. Since Discord integrates permissions for channels, users can also be given access with read/write permissions. This would add a level of security to ensure only authorised users can request for the door to open.

5. Future Work and Conclusion

Currently, the system works well, having undergone a day-long test without any unexpected alerts. However, due to time constraints, the door activation hardware could not be integrated, resulting in the inability to remotely open the door; the code should work assuming another solenoid can be obtained. The Discord client worked well in notifying my flatmates whenever the substation was called. There may exist more optimal methods for remote door opening but given the project's limitations, this approach appeared to be the most viable. However, for roughly £5.82 in components for two ESP32-C3s and a solenoid (excluding wires, resistors etc), this project is very simple and cheap to recreate.

Numerous enhancements could be made to improve this project. Currently, the audio input/output and video output are not captured. Given additional time, implementing audio/video transfer via the Discord client using UDP packets would be feasible. However, capturing audio and video requires physically picking up the phone, which triggers a switch on the board (seen in Figure 1) to activate the microphone and video. This would require another microcontroller to flip a switch on the board seen in Figure 1, which would require soldering the board and additional hardware I currently do not own.

6. Acknowledgements

I want to dedicate this project to Max Chaston (mdic1g21), my flatmate, who gave me this idea and all the hardware I used in this project (including the solenoid I broke). He also helped to test the system by calling the substation from the ground floor. Please note, if you replicate this project by yourself, it would be 10x easier to test if you could wirelessly call the substation as walking up and down 6 flights of stairs only to find out an error occurred can become tiresome.

References

[1] Forbel. "How to choose an apartment intercom system." (2021), [Online]. Available: https://forbel.com/blog/how-to-choose-an-apartment-intercom-system-key-factors-to-consider (visited on 08/05/2024).

[2] Switchbot. "Switchbot bot." (n/a), [Online]. Available: https://uk.switch-bot.com/products/switchbot-bot (visited on 08/05/2024).

[3] Spiro. "Videx buzzer alert/automatic door opening switch on pressing buzzer." (2021), [Online]. Available: https://community.home-assistant.io/t/videx-buzzer-alert-

automatic-door-opening-switch-on-pressing-buzzer/323761 (visited on 08/05/2024).

[4] Videx. "Videx art. 6272n intercom manual." (2016), [Online]. Available: https://videxuk.com/wp-content/uploads/2017/01/66251122_6272N_ENUK_V1-3.pdf (visited on 08/05/2024).