GCO

Dead Air: an investigative phone experience

A small british village on a remote island has suddenly stopped communicating with the mainland. You try calling them one day, and accidentally find an internal archive system, which has recorded the last calls made by all the phones, but only one half of the conversation. What has happened, why and how? And do you even want to know?

Story Overview

In the village of Little Wimbleton-by-the-sea, on the island of Nottingsard, there was a horror from centuries ago (big bad). The Company™ found high quality tin ore there, and commenced mining operations. This awoke the big bad, which started making the residents disappear/go insane/ <INSERT BAD THING HERE>.

When you call the village, you accidentally dial the wrong number, and access the telephone's internal archive system. It guides you through the system, explaining that due to storage limitations, you are only able to access one half of the conversation, and only the transcript is stored, not the actual audio. This means that the calls will instead be spoken by a robotic voice, and you will have to try and work out who is calling who. While connected to this system, you are able to dial any phone number in the system, and the last call that that phone recieved will be played, but only the side of the number you dialled. For example:

- "Hello?"
- "oh, hello"
- "yes"
- "no"
- "yes"
- "oh no! Thats terrible! I will have to tell her at once, thank you"
- "of course"
- "bye"

Functionality

When first lifting the phone, It will automatically call the internal archive system, and an automated voice will introduce the game and how to call numbers. After that, dialling a number will play the last call that that phone made (but only the side of the number that you selected). To quit a call prematurely, put the phone down, and when it is picked up again the phone will go back to the menu. If the phone is put down for 1 minute (or the game is quit from the menu), then the session ends, and the next time the phone is picked up, the game will restart.

The arduino will detect how many switch contacts are made before there is a gap of over 200ms, then send that number over serial to unity. Once there is a gap between keypresses of more than 2 seconds, if the number is 4 digits long then play the relevant audio file,

otherwise play some kind of error message (“The number you have called does not exist on our system”).

Controller Design

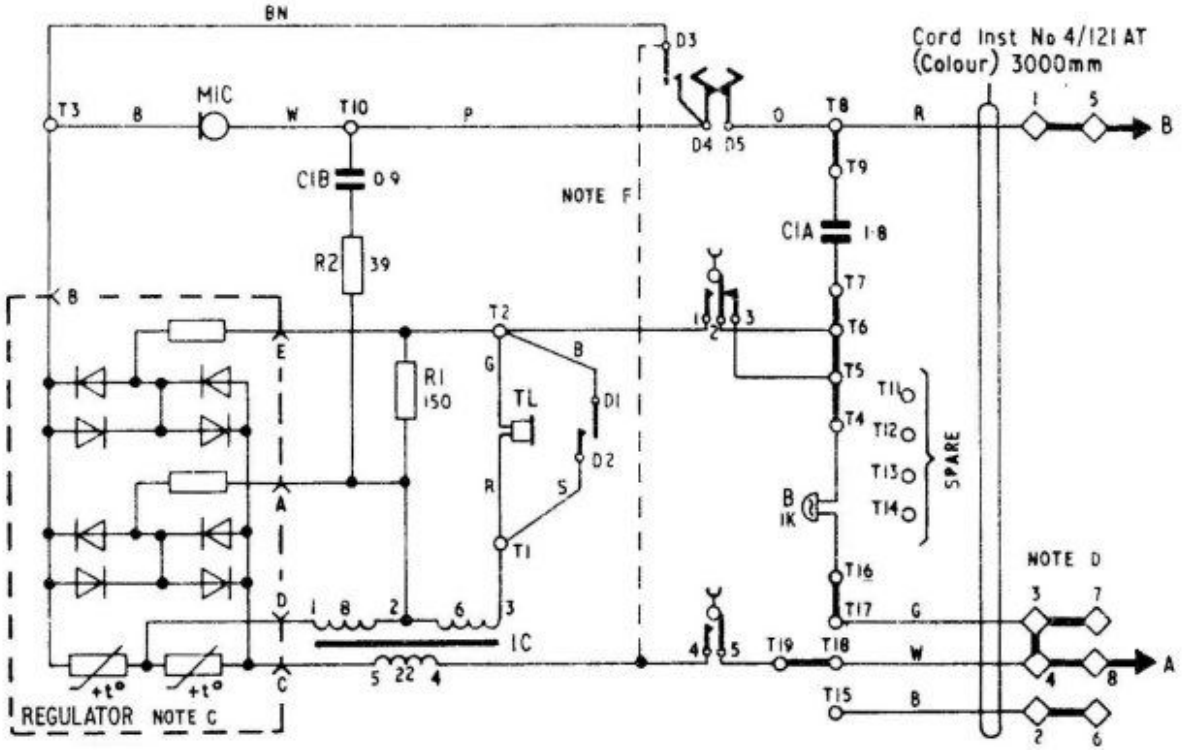
Looking through the GPO phones made at the time, I knew I wanted a rotary dial. I love the feel of them, and it also makes interfacing them with an arduino much easier. Rotary phones usually work by sending pulses corresponding to the number dialled (amounts can differ by region), at a rate of 10 pulses per second. By waiting until there is a gap of ⅕ of a second, the dialled number can be determined. The only thing left to decode was which phone exactly I wanted. I didn't like the look of the 121, 150, 162 or 232, however the 332, 706 and 746 all looked nice.

The benefit of the 706 over the 746 is the removable regulator. The regulator was a device that made the phone less sensitive, as when a user was too close to the exchange, they could sometimes pick up signals from radio stations and taxi radios. By the time of the 746, this was made a part of the main circuit board, however it was an optional feature in the 706. This would reduce the complexity of the circuit that I have to reverse engineer and tap into.

One potential issue I will have with the circuit is switch bouncing. Because of the sensitivity of an arduino, even the slightest wobble in the contact switch can make it think a break was made, which will disrupt the counting. I suspect that at the time these phones were used, the systems were a lot less sensitive, especially since they were electromechanical in nature, so it would not have been a concern. If this is correct, then I will have to add a switch debouncing circuit into the phone. This can be made using a capacitor and a resistor, which works by slowly lowering the voltage of the line. That way, any short disconnects will not lower the voltage enough to trigger a state change. This will fix any issue with the addition of only 2 extra components. I may also be able to solve this issue in software, by measuring the time difference between successive pulses, and cutting any that happen too quickly, however I would prefer to do this in hardware.

One feature that may be useful is the ability to ring the bells. The issue with that is that the system is entirely mechanical, and powered by an electromagnet. These can draw a lot of power, and if I power it off a usb port, then it may cause damage to the arduino or the usb port. I may have to use an external power supply in order to get the required power, but I would like to avoid this if at all possible.

Another nice thing about choosing an older phone is that they are well documented and easy to service. For example, it was relatively easy to track down a full schematic of the 746:



It is interesting to note that all of the connections labelled T are screw terminals on the back of the phone. This means that by just utilising the T connections, I can completely avoid all soldering, making it easy to change/fix any connections if needed. I believe that the bold lines between T connections indicate small pieces of metal that are placed at the screw terminals, which can be easily removed. This was probably originally to allow for technicians to make quick adjustments to the phones, but will let me disconnect all of the important items from the circuit, to prevent any connections that may cause incorrect or undesired operations.

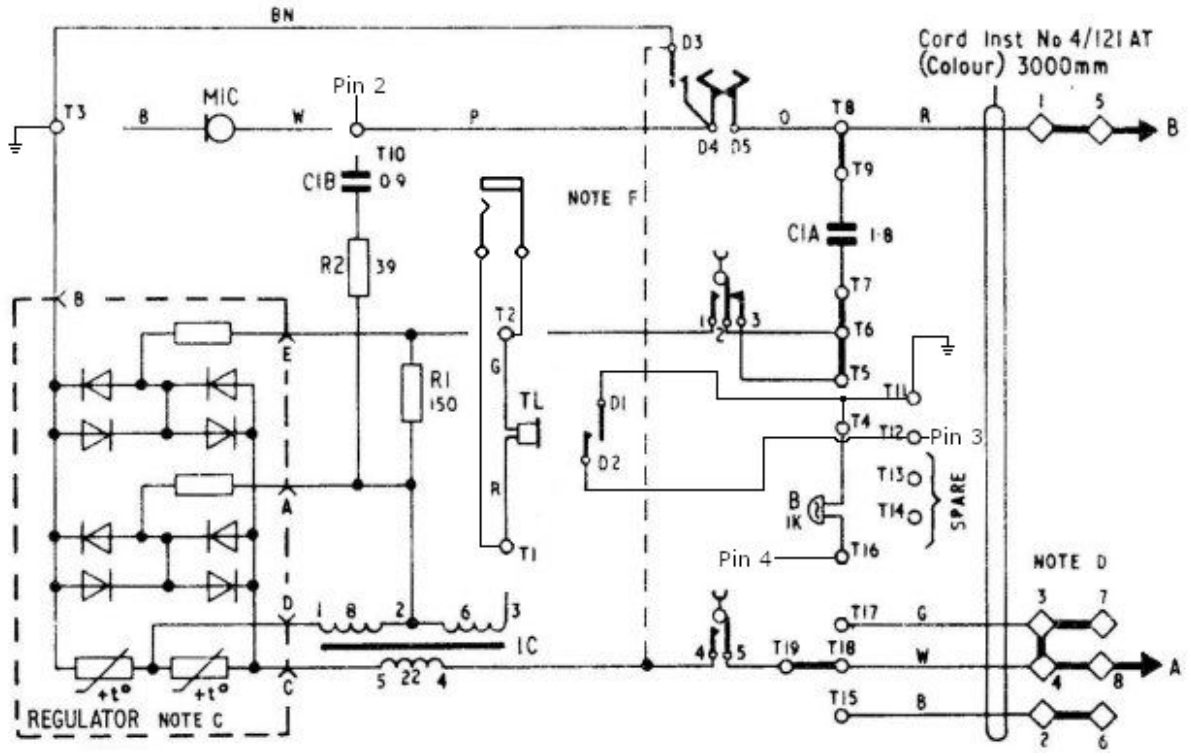
The dial itself is the arrangement near the top, labelled with d3/d4/d5. I believe that while the dial is rotating, the connection between D4 and D5 is broken, and the connection between D3 and D4 pulses at 10Hz. The bells are the symbol labelled B, connected between T4 and T16. The final important connections are T1 and T2, which I believe includes the switch at the top of the phone (closed by putting the phone down), and the speaker in the handset. I should be able to test for a short between T1 and T2 to determine if the phone is down, and play audio across the T1 and T2 pins to play the audio logs. However, testing for a short may interfere with the audio playing. If this is the case, I may have to try and disconnect the switch or the speaker, and use the wires directly instead of through the screw terminals. Alternatively, I could route them to two of the four unused terminals, T11, T12, T13 and T14.

My final proposed circuit is this:

Remove the blue wire connected to T3, and the white wire and capacitor connected to T10. This removes the microphone from the circuit. Connect a ground wire to T3, and a wire from T10 to the arduino. If a debouncing circuit is needed, place it between T10 and the arduino.

Remove everything except the green wire from T2, and connect the blue wire from T2 onto T11. Disconnect everything except the red wire from T1, and move the S (?) wire to T12.
Connect an audio cable to T1 and T2, and leave the rest of it hanging out of the back.
Connect a ground wire to T11, and a wire from T12 to the arduino.
Remove the connections between T4 & T5, and T16 & T17, connect a ground wire to T4 and a wire from T16 to the arduino.
This should be all the connections needed to make the controller functional.

Final Circuit Diagram:



Key electronic components:

- Arduino uno: Handles IO and serial connectivity
- Old GPO style phone (706/746): Acts as a case and provides the rotary phone element
- small capacitor and resistor: Debounces the contact switch (these may already be built into the phone)
- Audio cable. Used to route computer audio through to the speaker in the handset of the phone.

Key User Stories:

- As a user, I would like to be able to dial a number on the phone, and have the game play the correct audio file.

- As a user, I would like to be able to hear the relevant audio file though the handset of the telephone.

- As a user, I would like to be able to put the phone down to end the playing of an audio file prematurely.

- As a developer, I would like the phone to send messages about its state through the serial interface

- As a user, I would like there to be an intriguing story with a puzzle element