Call-Integrated Audio Stream Injection for Reliable Remote Pointing over Telephony Channels

Taito Yoshimura[1], Yoshihiro Kawakami[2], Mayu Miyake[1], Taiga Tsuzuki[1], and Yuki Matsuda[1][0000−0002−3135−4915]

[1] Okayama University, Okayama, Okayama 700–082, Japan
{taito.yoshimura, mayu.miyake, taiga.tsuzuki}@cocolab.jp, yukimat@okayama-u.ac.jp
[2] Kawakami Co., Ltd., Asakuchi, Okayama 719–0241, Japan

Abstract. Remote industrial troubleshooting often relies on ordinary telephone calls, particularly at customer sites with limited internet connectivity. Our previous telephony-based remote pointing method transmits audible control signals during a call and decodes them on the customer side to activate LEDs or buzzers attached to machine components. However, when such signals are played externally and captured by the operator-side smartphone microphone, speech-oriented audio processing may attenuate or distort them. This paper presents call-integrated audio stream injection for remote pointing over telephony channels. A custom calling application generates audible control signals and mixes them directly into the outgoing call audio stream, bypassing the operator-side microphone while preserving the existing customer-side decoder and actuators. Preliminary experiments compared the proposed method with an acoustic-input baseline based on the operator-side input scheme of our previous system. Although each ID was transmitted only once per condition, the proposed method decoded all eight IDs at a signal tone duration of 75 ms in both quiet and simulated noisy environments. In contrast, the acoustic-input baseline decoded 5/8 and 4/8 IDs at the same tone duration, respectively. These results suggest that the proposed approach may improve transmission reliability without requiring customer-side video or broadband connectivity.

Keywords: Remote pointing · Telephony channels · Audible-sound communication · Industrial maintenance support · Customer support.

1 Introduction

Many industrial manufacturers continue to provide remote maintenance and troubleshooting support through ordinary telephone calls. This practice remains especially important for equipment installed in geographically remote production sites, where stable internet connectivity cannot always be assumed. In such environments, video calls, remote desktop tools, and cloud-based collaboration

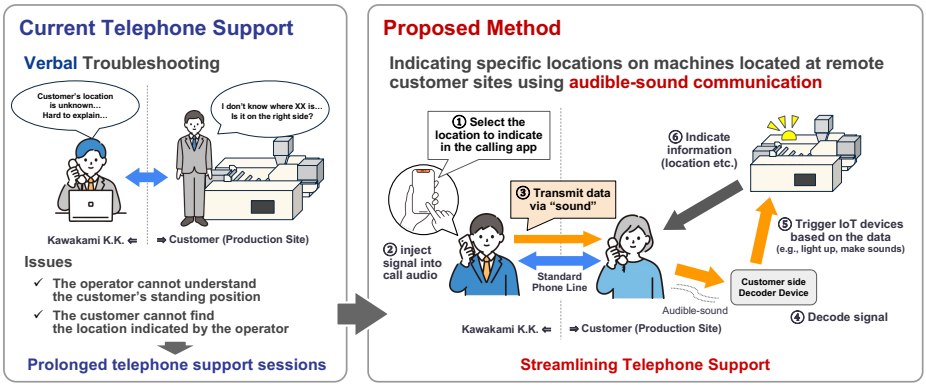


Fig. 1. Overview of this study.

systems are difficult to use, and support interactions depend primarily on verbal communication. However, for machines with complex mechanical structures and many visually similar components, verbal explanations alone are often insufficient. Operators and customers may fail to establish a shared understanding of the customer's viewpoint, the orientation of the machine, or the exact component being discussed. These misunderstandings can lead to prolonged support sessions, repeated explanations, incorrect operations, and unnecessary on-site visits.

Our previous work [7] introduced audible-sound remote pointing over telephone channels, in which an operator transmits a target ID during a voice call and a customer-side embedded device decodes the signal to activate an LED or buzzer attached to the corresponding machine component. This approach enables situated guidance without requiring video communication or additional customer-side network infrastructure. However, the previous operator-side implementation generated signal tones on an external device and acoustically fed them into a telephone. When a smartphone is used, microphone capture and speech-oriented processing, including noise cancellation, echo suppression, and automatic gain control, may attenuate or distort these non-speech tones. This external acoustic-input path therefore limits the reliability and practicality of smartphone-based operation.

In this paper, we propose a call-integrated audio stream injection method for reliable remote pointing over telephony channels as shown in Fig. 1. The key idea is to avoid treating the control signal as an external acoustic event on the operator side. Instead of playing a signal sound into the smartphone microphone, the proposed system integrates signal generation and transmission into a custom calling application. The operator can make a voice call, select the target location on the equipment, and transmit the corresponding control signal from the same smartphone interface. The generated signal is injected into the call audio stream within the application-controlled calling process, allowing

the system to bypass the operator-side microphone input path and mitigate the influence of smartphone noise-cancellation mechanisms.

The proposed design preserves the practical advantages of the original audible-sound remote pointing method. The customer side does not need to install a dedicated communication application, prepare a video-call environment, or use broadband internet access. The received sound can still be played through an ordinary telephone receiver, captured by an embedded device, decoded as control information, and mapped to actuators installed on the target machine. Thus, the system remains suitable for industrial environments where conventional telephone support remains a reliable and widely available communication method. At the same time, the operator-side workflow becomes more practical because signal transmission is integrated into a smartphone application rather than requiring a separate PC-based sound generation tool.

The contributions of this paper are threefold. First, we identify the operator-side smartphone microphone input path as a practical source of signal attenuation in audible-sound communication over telephony channels. Second, we propose a call-integrated audio stream injection method that transmits control signals from within a custom calling application, thereby bypassing smartphone noise-cancellation mechanisms applied to external microphone input. Third, we present a system architecture that maintains compatibility with customer-side embedded decoding and actuation devices, enabling situated physical guidance for remote industrial support over ordinary telephone audio on the customer side. Through this work, we aim to improve the reliability and deployability of remote pointing systems for real-world maintenance scenarios where smartphones are increasingly used as the primary communication device.

2 Related Work

2.1 Acoustic Data Transmission as an Alternative Communication Channel

Sound has long been used not only for human communication but also as a carrier of encoded information. Before digital communication networks became widespread, analog channels were already used to transmit non-verbal information. For example, Bain's chemical telegraph introduced an early mechanism for scanning and reproducing information at a remote location [3]. Later, Ives *et al.* demonstrated image transmission over telephone lines, which became an important technical basis for facsimile communication [5]. These historical systems show that channels originally designed for human-perceivable signals can also be repurposed to convey machine-readable data.

Recent studies have revisited acoustic communication in the context of mobile, ubiquitous, and embedded systems. A major line of work uses high-frequency or inaudible sound so that data can be embedded without disturbing users. Morise *et al.* embedded lyrics and artist information into music by using mosquito sounds, enabling synchronized information retrieval while preserving the listening experience [8]. Suzuki *et al.* proposed *AnnoTone*, which embeds annotation

information into videos as high-frequency audio signals and supports context-aware video editing on smartphones [10]. Ahn *et al.* used inaudible frequencies for robot-to-robot communication in manufacturing environments without relying on internet connectivity [1]. Okano *et al.* improved robustness under noisy conditions by combining superposed acoustic signals with redundant coding, achieving high recognition accuracy [9]. Bai *et al.* proposed *BatComm*, a near-ultrasonic acoustic communication system that enables high-throughput data transmission between commodity devices [2]. Hanspach *et al.* also showed that acoustic signals propagating through air can form low-bandwidth mesh networks without conventional wireless or internet-based communication [4]. Together, these studies demonstrate that acoustic signals can function as flexible communication links for devices and services operating under infrastructure constraints.

Another line of work employs audible sound as a communication medium. Although audible signals can be perceived by users, they have the advantage of being compatible with a wider range of speakers, microphones, and legacy audio channels. Katzschmann *et al.* used acoustic communication to control a soft robotic fish underwater, where radio-based communication is difficult [6]. Yang *et al.* proposed *DTMFTalk*, which uses Dual-Tone Multi-Frequency signals during voice calls for IoT remote control [11]. Their work is particularly relevant to the present study because it shows that ordinary voice-call channels can be used to transmit control information in environments where internet-based communication is not always available.

2.2 Remote Pointing over Telephone-Based Audible Communication

Building on the idea of using telephony channels as a data path, our previous study proposed a remote pointing method for industrial telephone support [7]. An operator selected a target position on the machine, and the selected identifier was encoded as a sequence of audible tones transmitted through an ordinary telephone call. On the customer side, an embedded device captured the received sound, decoded the identifier, and activated a corresponding peripheral device, such as an LED or buzzer, attached to the target component. This mechanism allowed the operator to indicate a physical location on remote equipment while continuing the voice conversation with the customer.

The previous study demonstrated that such audible-sound remote pointing can work in noisy industrial environments. At the same time, it revealed an important limitation: the prototype generated signal sounds on a PC and delivered them acoustically through a telephone device, and as described in Section 1, this acoustic input path is subject to speech-oriented call audio processing when the operator uses a smartphone. The reliability of the overall remote pointing system can therefore be limited not only by customer-side decoding accuracy but also by the operator-side audio input path.



(a) Previous operator-side acoustic-input configuration (b) Microphone-mode controls during a call

Fig. 2. Previous acoustic-input configuration [7] and iPhone audio-processing controls.

2.3 Positioning of This Study

The studies reviewed above demonstrate the potential of acoustic signals as a communication channel under infrastructure constraints. However, many inaudible or near-ultrasonic approaches assume audio bandwidths that are not available in ordinary telephony channels, which are primarily optimized for speech. For telephone-based remote support, audible sound is therefore a more suitable carrier, as also demonstrated by DTMF-based voice-call control systems such as *DTMFTalk*. Meanwhile, our previous remote pointing system introduced signal sounds as external audio captured by a telephone or smartphone microphone, which limits its reliability and deployability in practical smartphone-based support settings. This study addresses this operator-side transmission problem by introducing *call-integrated audio stream injection*: the contribution lies in redesigning the operator-side transmission architecture so that remote pointing can be performed more reliably without changing the customer-side reception and actuation system.

3 Proposed Method

3.1 Concept

In this study, we extend the remote pointing method of our previous work [7] for practical smartphone-based operation. The previous operator-side configuration required a separate PC to generate signal tones, as shown in Fig. 2(a), and introduced them acoustically through the calling device. When a smartphone is used, speech-oriented audio processing, such as noise cancellation, may attenuate

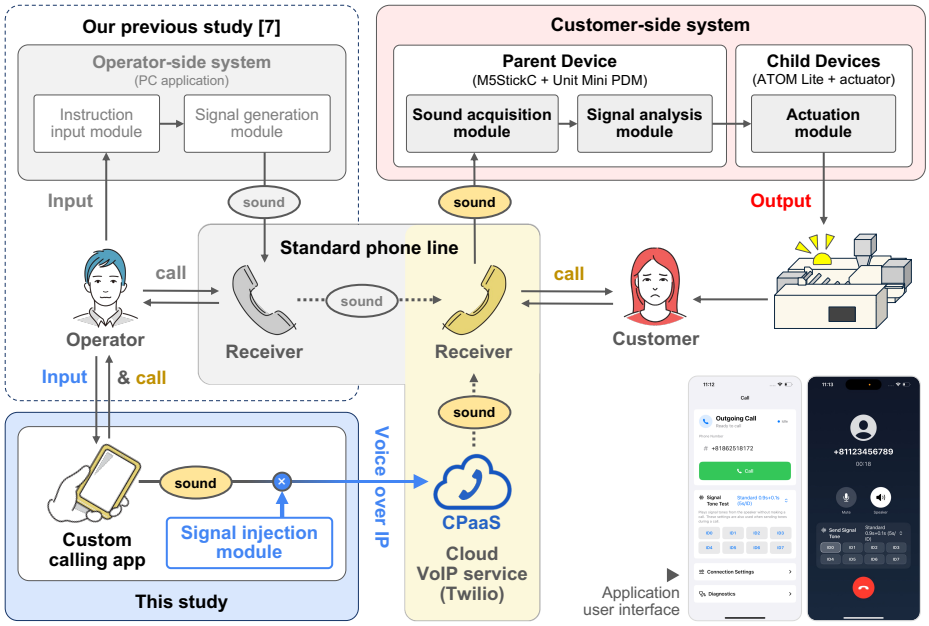


Fig. 3. Proposed call-integrated remote pointing architecture.

or distort these non-speech tones. Although operators can change the microphone mode during a call using the control shown in Fig. 2(b), relying on manual configuration adds operational burden and does not ensure that the signals are transmitted without modification. A practical system should therefore introduce signal tones directly into the call audio stream without relying on acoustic input or user-managed audio settings.

The proposed method is *call-integrated audio stream injection*: signal tones are generated inside the calling application and mixed directly into the call audio stream (Fig. 3). A system realizing this method must satisfy three design requirements. First, the operator can perform calling, voice conversation, target ID selection, and signal transmission entirely on a single smartphone. Second, the signal tones enter the call without passing through the microphone of the operator-side smartphone. Third, the customer side can continue to use the reception system of our previous work, that is, the telephone receiver audio and the embedded device for decoding and actuation. The operator's workflow is to call the customer from a custom iOS calling application, select the target ID on the call screen while continuing the voice conversation, and send the signal from the same application. The application internally synthesizes the audible signal corresponding to the selected ID and injects it into the call uplink connected through the Twilio Voice SDK. On the customer side, a reception, decoding, and actuation system compatible with our previous work is used.

3.2 Operator-Side iOS Application

The operator side of the proposed system uses a custom iOS calling application. The application places and controls telephone calls using the Twilio Voice SDK[3] and operates as an iOS call session in cooperation with CallKit. As shown in the lower right of Fig. 3, the call screen provides buttons for muting, switching to the speaker, selecting a tone-duration preset, and sending IDs 0 to 7. The tone-duration presets are a feature for the experiments; a simplified screen is assumed for practical use.

In a standard iOS calling workflow using the native phone audio path, applications cannot arbitrarily add generated audio to the outgoing call stream; the transmitted audio is essentially limited to what is captured by the microphone. The Twilio Voice SDK, however, provides a custom audio device mechanism that replaces the audio input and output processing of a call with an application-specific implementation. Our application incorporates a custom audio device based on the sample implementation provided with the SDK[4] and uses this replacement to inject signal tones into the call. The voice captured by the microphone and the signal tones generated inside the application are mixed within the application, and the mixed audio is passed to the telephone line as the outgoing call audio. In other words, the signal tones are added directly to the outgoing call audio immediately before transmission, without passing through the microphone or propagating through the air. In addition, when a send button is pressed, the same signal tone is also played on the operator-side device, so the operator can confirm the transmission timing by ear while sending the signal to the other party. This mixing process is implemented with AVAudioEngine, the standard audio processing framework of iOS, and the generated signal waveform is supplied to both the outgoing audio path for the other party and the local playback path for operator confirmation.

3.3 Signal Encoding

The signal encoding scheme is identical to that of our previous work [7]. Each ID from 0 to 7 is represented as 3 bits $b_2b_1b_0$, with the most significant bit first. The even parity bit p is computed as $p = b_2 \oplus b_1 \oplus b_0$. The signal for one ID consists of five tones: a start signal, the parity bit, and the three ID bits, arranged in the order $[\mathrm{Start}, p, b_2, b_1, b_0]$.

Each tone is a composite sound of two superimposed frequencies; the frequency assignments are shown in Table 1. These frequency pairs were selected so that the tones are readily distinguishable and were intended to form relatively unobtrusive chords. For example, ID 5 is represented as 101 in 3 bits and its even parity is 0, so the transmitted sequence is $[\mathrm{Start}, 0, 1, 0, 1]$. Even parity can detect an odd number of bit errors but cannot detect 2-bit errors.

The iOS application dynamically generates these signal tones as 48 kHz monaural PCM buffers. The customer-side decoder uses the FFT bins at 1500 Hz,

[3] `https://www.twilio.com/docs/voice/sdks/ios`
[4] `https://github.com/twilio/voice-quickstart-ios`

Table 1. Frequency combinations

Symbol	Frequency 1 (Note)	Frequency 2 (Note)
Start	1481 Hz (F#)	2468 Hz (D#)
0	1481 Hz (F#)	1975 Hz (H)
1	1975 Hz (H)	2468 Hz (D#)

2000 Hz, and 2500 Hz for its decisions; among the bins spaced at 250 Hz intervals, these are the closest to the generated frequencies. Let T denote the tone duration and G the inter-tone gap; the transmission time for one ID is then $5(T+G)$. The tone duration and the inter-tone gap can be changed in the application, and the conditions used in this study are described in Section 4.

3.4 Customer-Side Decoder

The customer-side reception system follows the configuration of our previous work. In normal operation, an M5StickC with an external I2S microphone captures the audio from the telephone receiver, and the decoded ID is broadcast via ESP-NOW to ATOM Lite child devices. In the implementation used for the evaluation, the decoding result was shown on the M5StickC screen, and the FFT results computed by the M5StickC were sent to a PC over HTTP and saved as CSV logs.

The current receiver implementation performs an FFT approximately every 25 ms and records the frequency components from 250 Hz to 4000 Hz at 250 Hz intervals. The detection threshold is set to three times the average of the 16 bins, computed from a single frame captured 15 s after the M5StickC starts up. The start signal is detected when the pair of 1500 Hz and 2500 Hz components persists for at least 60% of the tone duration. After that, in each bit window, the receiver checks whether the two corresponding frequencies exceed the threshold in at least 60% of the frames within the window. If both bit 0 and bit 1 are judged valid, the bit represented by the frequency pair with the larger amplitude sum is selected. An ID is output only when all four bit windows are successfully determined and the even parity check is satisfied; otherwise, no ID is output, and the result is classified as a non-detection.

4 Evaluation

4.1 Common Experimental Conditions

The purpose of this evaluation is to clarify how the operator-side transmission method, that is, acoustic input or stream injection, affects reception performance at short tone durations. The evaluation consists of an experiment in a quiet environment and an experiment in a noisy environment; this subsection describes the conditions common to both.

Two transmission methods were compared: an acoustic-input baseline following the operator-side input scheme of our previous work [7], and the proposed stream-injection method. In the acoustic-input baseline, to use the same signal generator as the proposed method, signal tones were generated by the custom smartphone application on an iPhone 14 Pro (iOS 18.3) and played at the maximum volume. These tones were acoustically captured by the microphone of another smartphone used for the telephone call and were then transmitted to the customer side as ordinary call audio. The calling smartphone used in this baseline condition was an iPhone 15 Pro (iOS 26.5.2). This setup reproduces the external acoustic input path of the previous system while using the newly developed application to generate signal tones under the same encoding and timing conditions.

In the proposed method, the same custom iOS application generated the signal tones inside the operator-side smartphone and injected them directly into the outgoing call audio stream. The operator-side smartphone used in the proposed method was an iPhone 14 Pro (iOS 18.3). The application was implemented using Twilio Voice SDK version 6.13.6 and iOS CallKit. The signal tones injected by the proposed method were set to a peak PCM amplitude of 0.2, where 1.0 corresponds to full scale.

For both methods, the customer-side telephone handset was set to speaker mode and its volume was set to the maximum level. The customer-side telephone, M5StickC, external microphone, decoding process, and logging method were kept identical across all conditions. The call route also differed between the two methods in its operator-side leg: the acoustic-input baseline placed a standard call over the mobile carrier's voice network from the native phone application of the calling iPhone, whereas the proposed method placed a VoIP call through the Twilio Voice SDK, which was then terminated to the ordinary telephone network. In both methods, the call reached the customer-side telephone through the ordinary telephone network; therefore, the customer-side reception path was kept the same, while the operator-side transmission configuration differed. This evaluation thus compares the two operator-side configurations as complete systems, including their call routes, rather than isolating the signal input path as a single variable.

The signal conditions are as follows. The inter-tone gap is fixed at 100 ms in all conditions. Twelve tone durations are tested: nine durations from 900 ms down to 100 ms in 100 ms steps, plus 75 ms, 50 ms, and 25 ms. For each combination of transmission method and tone duration, IDs 0 to 7 are transmitted once each in order; the reception result shown on the M5StickC screen is recorded, and the FFT results are saved as CSV logs. To keep the customer-side conditions consistent, the customer-side telephone, the M5StickC and external microphone that capture the receiver audio, the decoding process, and the logging method are identical across all conditions. As the customer-side telephone, we used the base unit and the bundled cordless handset of a Sharp digital cordless telephone JD-G33CL (one-handset configuration)[5].

[5] `https://jp.sharp/phone/products/jdg33/`

For summarizing the results, the effective bit rate is defined as the amount of data per ID (3 bits) divided by the transmission time of one ID (5 tones × (tone duration + 100 ms gap)). For example, the effective bit rate is 3 bits / 5.0 s = 0.60 bps at a 900 ms tone duration and 3 bits / 1.0 s = 3.00 bps at 100 ms.

4.2 Analysis Method

As in our previous work [7], reception was judged by whether the customer-side signal analysis module correctly output the transmitted ID. Because the customer-side actuation module is unchanged from our previous system, successful decoding was used as the evaluation criterion instead of directly recording actuator activation. Each transmitted ID was treated as one trial, and the result shown on the M5StickC screen was compared with the transmitted ID. A trial was counted as a success when the same ID as the transmitted one was displayed, as a misrecognition when a different ID was displayed, and as a non-detection when no ID was output because the decoder rejected the signal as invalid. Misrecognition is treated separately as a safety-critical silent error because the erroneous result passes the parity check and causes the system to output an incorrect ID instead of rejecting the signal. In the tables, ✓ denotes success and - denotes non-detection; for misrecognition, the misrecognized ID is shown as a number. The CSV logs of the FFT results were not used to judge reception success; they were used for post-hoc analysis of the trials in which misrecognition occurred.

4.3 Experiment in a Quiet Environment

The experiment in the quiet environment was conducted in a laboratory at our university. On the customer side, the bundled cordless handset of the JD-G33CL was set to speaker mode, and the external microphone connected to the M5StickC was placed approximately 10 cm away from the speaker of the handset (Fig. 4).

Results. As shown in Table 2, both methods succeeded for all IDs at tone durations of 200 ms or longer. At shorter durations, however, a difference emerged: the acoustic-input baseline succeeded for 4 of 8 IDs at 100 ms and 5 of 8 at 75 ms, whereas the proposed method succeeded for all eight IDs at both durations. The shortest tone duration at which all IDs succeeded in this experiment was 200 ms for the acoustic-input baseline and 75 ms for the proposed method. In terms of the transmission time of one ID, these correspond to 1.5 s and 0.875 s, respectively, both of which are substantially shorter than the 5.0 s of the standard condition.

Fig. 5 shows the FFT logs received in the 200 ms condition, in which both methods succeeded for all eight IDs. In the figure, the three signal frequencies (1500 Hz, 2000 Hz, and 2500 Hz) are highlighted in color, the other analyzed frequencies are shown in gray, and each shaded band indicates the span of one

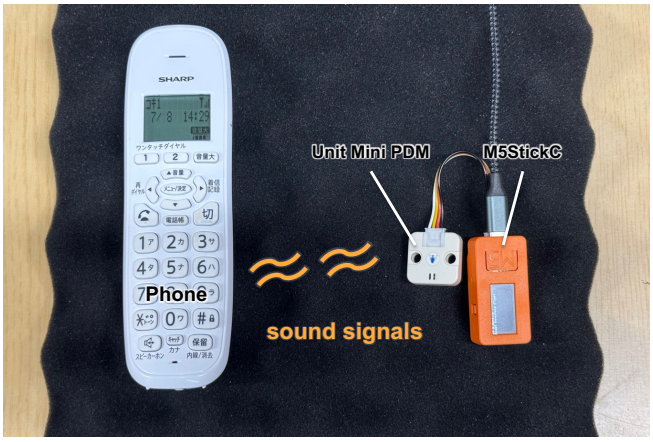


Fig. 4. Device arrangement in the quiet-environment experiment.

Table 2. Reception results in the quiet environment. The 900–200 ms row summarizes all tested tone durations in this range.

Method	Signal tone duration [ms]	Signal							
		ID 0	ID 1	ID 2	ID 3	ID 4	ID 5	ID 6	ID 7
Acoustic input (as in [7])	900–200	✓	✓	✓	✓	✓	✓	✓	✓
	100	✓	-	✓	-	✓	7	-	✓
	75	✓	✓	7	✓	✓	-	-	✓
	50	-	-	-	-	-	-	-	✓
	25	-	-	-	-	-	-	-	-
Stream injection (proposed)	900–200	✓	✓	✓	✓	✓	✓	✓	✓
	100	✓	✓	✓	✓	✓	✓	✓	✓
	75	✓	✓	✓	✓	✓	✓	✓	✓
	50	✓	-	✓	-	✓	7	✓	✓
	25	-	-	-	-	-	-	-	-

decoded ID. Even in this fully successful condition, the 2500 Hz component was dominant and the 1500 Hz component was relatively weak with the acoustic-input baseline, whereas the amplitudes of the three signal frequencies were nearly uniform with the proposed method.

At 50 ms and below, both methods degraded, and neither succeeded at 25 ms. The results in this region were likely limited not only by the difference in the transmission path but also by constraints on the receiver side: the FFT interval of approximately 25 ms, the required hit ratio within each decision window, and the minimum duration for start signal detection.

Error Analysis. In total, three misrecognitions passed the parity check and were decoded as incorrect IDs: two with the acoustic-input baseline and one with the proposed method. With the acoustic-input baseline, ID 5 was misrecognized

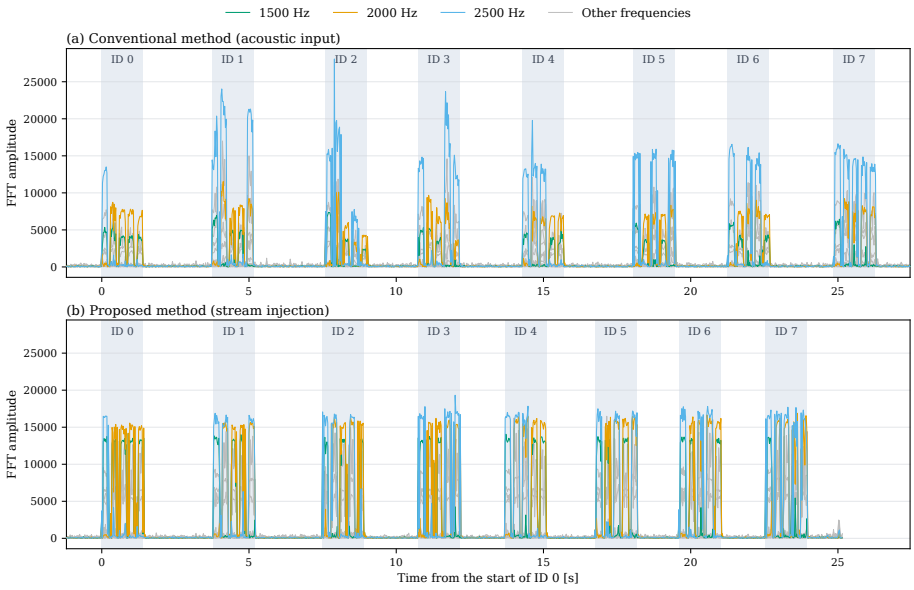


Fig. 5. Received FFT logs of all eight IDs (quiet environment / 200 ms): (a) acoustic-input baseline, (b) proposed method.

as ID 7 at 100 ms, and ID 2 was misrecognized as ID 7 at 75 ms. With the proposed method, ID 5 was misrecognized as ID 7 at 50 ms. In all of these cases, two 0s flipped to 1s and the signal was decoded as ID 7; these are silent errors that even parity cannot detect.

4.4 Experiment in a Noisy Environment

The noisy-environment experiment was conducted in the same laboratory, and the operating sound of a noodle-bundling machine recorded in a factory was played back around the customer-side receiving equipment. For reception on the customer side, we used the same receiving box as in our previous work [7] (Fig. 6). The receiving box has a microphone attached to the inside of its top face; the JD-G33CL handset was placed inside the box in speaker mode, and the M5StickC was also placed inside the box. All other conditions were the same as in Subsection 4.1.

Table 3 shows the reception results in the noisy environment. At 200 ms and longer, both methods succeeded for all IDs. At 100 ms, the acoustic-input baseline succeeded for 5 of 8 IDs while the proposed method succeeded for all 8; at 75 ms, the success counts were 4/8 and 8/8, respectively. Therefore, the shortest tone duration at which all IDs succeeded in the noisy environment was 200 ms for the acoustic-input baseline and 75 ms for the proposed method. The transmission time of one ID is 1.5 s in the 200 ms condition of the acoustic-input baseline,

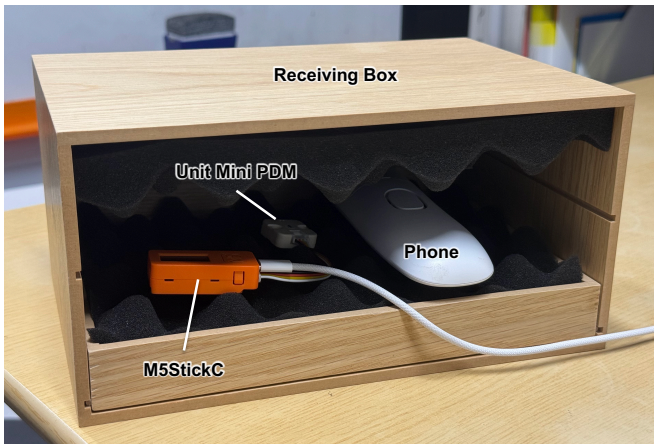


Fig. 6. Receiving box used in the noisy-environment experiment.

Table 3. Reception results in the noisy environment. The 900–200 ms row summarizes all tested tone durations in this range.

Method	Signal tone duration [ms]	Signal							
		ID 0	ID 1	ID 2	ID 3	ID 4	ID 5	ID 6	ID 7
Acoustic input (as in [7])	900–200	✓	✓	✓	✓	✓	✓	✓	✓
	100	✓	✓	-	-	✓	✓	-	✓
	75	-	-	-	✓	✓	7	✓	✓
	50	-	-	-	-	-	-	-	-
	25	-	-	-	-	-	-	-	-
Stream injection (proposed)	900–200	✓	✓	✓	✓	✓	✓	✓	✓
	100	✓	✓	✓	✓	✓	✓	✓	✓
	75	✓	✓	✓	✓	✓	✓	✓	✓
	50	-	-	-	-	-	-	✓	-
	25	-	-	-	-	-	-	-	-

whereas it is 0.875 s in the 75 ms condition of the proposed method. In terms of the effective bit rate, these correspond to 2.00 bps and approximately 3.43 bps, respectively; that is, the proposed method reduced the transmission time by approximately 42% while maintaining successful reception of all IDs. At 50 ms and below, neither method achieved stable decoding, presumably because of the same receiver-side constraints on time resolution as in the quiet environment.

In the noisy environment, the acoustic-input baseline produced one silent error at 75 ms: the transmitted ID 5 was misrecognized as ID 7. In contrast, no misrecognition was observed for the proposed method in this noisy-environment experiment.

Table 4. Current codewords and their susceptibility to ID 7 misrecognition.

ID	p	b_2	b_1	b_0	d_H to ID 7	$0 \rightarrow 1$ boundaries
0	0	0	0	0	4	0
1	1	0	0	1	2	1
2	1	0	1	0	2	1
3	0	0	1	1	2	1
4	1	1	0	0	2	0
5	**0**	**1**	**0**	**1**	**2**	**2**
6	0	1	1	0	2	1
7	1	1	1	1	0	0

5 Discussion

The experimental results in the quiet and noisy environments suggest that the proposed method can improve the operability of remote pointing and reduce the transmission time required for each signal. Integrating call control, voice conversation, target ID selection, and signal transmission into a single iOS application eliminates the need for a separate external sound-source device and an acoustic-input procedure, thereby simplifying operation during telephone support. In terms of reception performance, the improved results at short tone durations, together with the failures at 50 ms and below, suggest that the minimum usable tone duration is determined not only by the operator-side transmission path but also by the customer-side reception state machine and the temporal resolution of the FFT.

5.1 Remaining Receiver-Side Error Mechanisms

Four silent errors were observed in the experiments: three in the quiet environment and one in the noisy environment. Three of them decoded ID 5 as ID 7, and the remaining one misrecognized ID 2 as ID 7. In remote pointing, activating an LED or buzzer at the wrong location may lead the customer to an incorrect operation, so such silent errors pose a greater practical risk than simple non-detections. This subsection shows that the repeatedly observed misrecognition as ID 7 can be explained as a combination of three factors: (1) the susceptibility to misrecognition inherent in the current code assignment, (2) the contamination of boundary frames in the receiver-side window decisions, and (3) the frequency imbalance caused by the transmission path of the acoustic-input baseline.

Code-Level Susceptibility. Table 4 shows the decision sequence of each ID, consisting of the parity bit and the 3 data bits. The decision sequence of ID 7 is $[1, 1, 1, 1]$, and the Hamming distance between ID 7 and each of IDs 1 to 6 is 2. That is, any of these IDs turns into ID 7 when two 0s change to 1s, and this error passes the even parity check because it is a 2-bit error. The single misrecognition of ID 2 as ID 7 can be explained by this general mechanism.

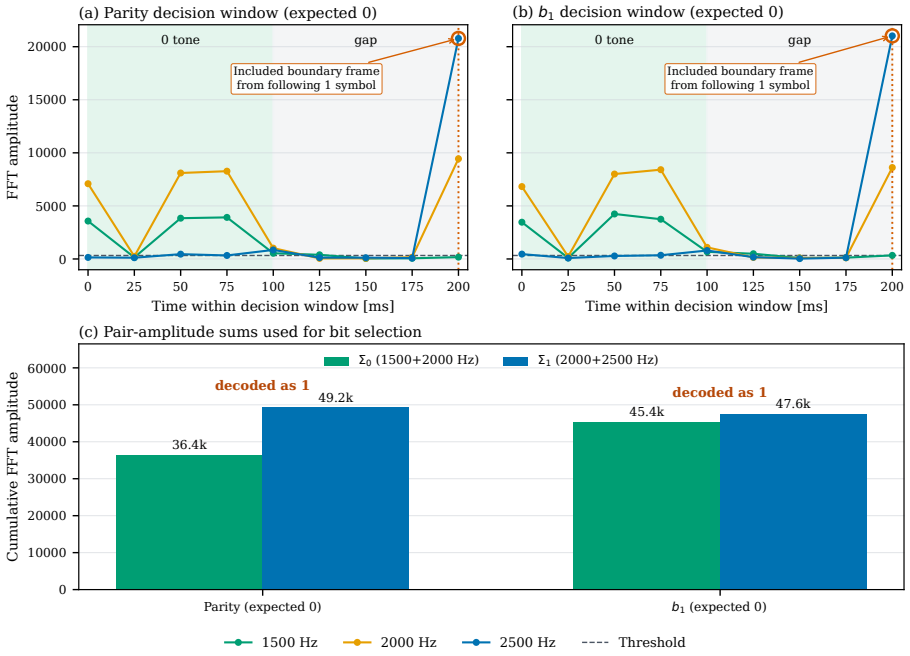


Fig. 7. Misrecognition of ID 5 as ID 7 (acoustic-input baseline / 100 ms).

Among these IDs, the data bits of ID 5 are 101, and its decision sequence is $[0, 1, 0, 1]$. ID 5 is the only ID whose 4-bit decision sequence contains two boundaries where a 0 is immediately followed by a 1. Structurally, therefore, ID 5 is the most susceptible to the error in which the following 1 tone leaks into the preceding 0 window at a window boundary: when both the parity bit and the middle bit flip from 0 to 1, the sequence is decoded as $[1, 1, 1, 1]$, that is, ID 7. We consider this property of the codewords to be one reason why the misrecognition of ID 5 as ID 7 was observed repeatedly.

Boundary-Frame Contamination in Window Decisions. This boundary contamination was confirmed in an actual misrecognition case. Fig. 7(a) and (b) show magnified views of the parity window and the middle-bit window when ID 5 was misrecognized as ID 7 with the acoustic-input baseline at 100 ms in the quiet environment. The orange circles indicate the boundary frames originating from the following 1 tones. Each decision window consists of a 100 ms 0 tone, a 100 ms gap, and the frame at its end boundary. The end boundary of both windows contained a large 2500 Hz component, presumably originating from the immediately following 1 tone. Because this frame was added to a window that should contain 0, not only the 1500 Hz and 2000 Hz pair representing 0 but also the 2000 Hz and 2500 Hz pair representing 1 became valid.

Fig. 7(c) shows the cumulative pair amplitudes that the receiver implementation used for bit selection. In the parity window, the amplitude sum was 49.2×10^3 for 1 versus 36.4×10^3 for 0 (FFT amplitude, arbitrary units); in the middle-bit window, it was 47.6×10^3 for 1 versus 45.4×10^3 for 0. The bit 1 was thus selected in both windows, the two 0s in the transmitted sequence $[0, 1, 0, 1]$ were both judged as 1, and the final received sequence became $[1, 1, 1, 1]$.

The window decision of the current receiver implementation allows this contamination for the following reasons. First, each frame is added to the current window before the end time of the window is checked, so the first frame of the next symbol, located at the end boundary, can be included in the preceding window. Second, the decision window for one bit spans the tone duration plus the 100 ms gap, and the frames within the gap are also included in the amplitude accumulation; the gap therefore does not function as a margin that absorbs timing deviations in reception. When the start of a window is delayed by the detection time of the start signal or by the FFT execution timing at approximately 25 ms intervals, the end of the window also shifts toward the next symbol, and the beginning of the following tone can enter the preceding decision window even though the gap exists. In addition, under short tone durations, the required number of hits per window drops to as few as 2 frames, so a single boundary frame has a relatively large influence on the decision. The implementation also favors 1: when both 0 and 1 are valid and their amplitude sums are equal, 1 is selected.

Frequency Imbalance in the Acoustic-Input Baseline. The remaining question is why these weaknesses manifested as misrecognitions mainly in the acoustic-input baseline. We compare the ratio of the amplitude sum for 1 to that for 0. In the misrecognition case described above, this ratio was approximately 1.35 in the parity window and 1.05 in the middle-bit window; the bit-1 pair exceeded the bit-0 pair in both windows. In contrast, under the conditions in which the proposed method received the same ID 5 correctly, the ratio remained approximately 0.43 at 100 ms in the quiet environment and 0.54 at 75 ms in the noisy environment; the bit-0 pair stayed dominant even though boundary frames were included. This difference suggests that the two methods differ not only in the presence of boundary contamination but also in the amplitude balance of the received frequency components.

Each symbol is synthesized at transmission by combining two frequencies with equal amplitude. Therefore, if the transmission path introduces little imbalance, the amplitude ratio of the two corresponding components within the same FFT frame is expected to be close to 1 at the receiver. We examined this amplitude ratio for the 100–900 ms conditions in the quiet environment. The values here are not calibrated sound pressures but relative ratios of the FFT amplitudes computed by the M5StickC. In terms of the median across IDs at each tone duration, the ratios of the proposed method were approximately 0.87–1.14 for the 1500 Hz / 2000 Hz ratio of bit 0 and approximately 1.05–1.91 for the 2500 Hz / 2000 Hz ratio of bit 1. In the acoustic-input baseline, the same ratios

were approximately 0.46–0.62 and 1.67–3.01, respectively; that is, the ratios of the proposed method stayed generally closer to 1, whereas the acoustic-input baseline showed a clear imbalance in which the lower 1500 Hz component was small and the higher 2500 Hz component was large. Under this imbalance, the 0 pair (1500 Hz and 2000 Hz) has a lower cumulative amplitude than it should, while the 1 pair (2000 Hz and 2500 Hz) has a higher one. Consequently, when a 2500 Hz component leaks in from a boundary frame, the amplitude sum for 1 easily exceeds that for 0. Moreover, in the acoustic-input baseline, this imbalance was not constant over time. Even for the start signals transmitted with identical content before each of IDs 0 to 7, the 2500 Hz / 1500 Hz amplitude ratio in the 700 ms condition varied in the range of approximately 1.5–3.1.

We consider the causes of this imbalance as follows. Both methods shared the JD-G33CL handset and the microphone on the M5StickC side, so the overall tendency of the 2500 Hz component to be relatively large may reflect the frequency characteristics of the receiving-side acoustic path, including the handset. The two methods also differed in their call routes and speech codecs (Subsection 4.1), and this difference may likewise have contributed to the static imbalance. On the other hand, the temporal fluctuation of the amplitude ratio was observed only in the acoustic-input baseline, whereas the spectral shaping of speech codecs is largely time-invariant. This fluctuation therefore suggests an additional contribution from the time-varying audio processing that only the acoustic-input baseline passes through, such as the smartphone microphone input, automatic gain control, and noise suppression. However, because each stage of the transmission path was not measured individually in this experiment, we cannot attribute the cause to a specific device or process. In any case, the attenuation of the 1500 Hz component and the persistence of the 2500 Hz component in the acoustic-input baseline exposed the weaknesses of the code assignment and window decision described above.

This analysis indicates that suppressing silent errors requires improvements not only in the transmission path but also in both the code and the window decision. For the window decision, the amplitudes could be accumulated only over the tone interval, treated as a half-open decision window that excludes the gap, so that the gap serves as a guard interval absorbing timing deviations. This could suppress the leakage of adjacent symbols and stabilize reception at even shorter tones. At the 25 ms condition, however, the tone duration is comparable to the FFT interval, and even satisfying the current hit condition of at least 2 frames is difficult. Further shortening of the tones therefore requires redesigning the FFT interval and the required number of hits, together with repeated evaluations under identical conditions. For the code, the introduction of a CRC or redundant coding should also be considered to detect 2-bit errors.

5.2 DTMF as an Alternative Encoding

Our previous work represented each symbol using a pair of frequencies selected to be readily distinguishable while forming relatively unobtrusive chords [7]. This

choice was not based on a comparison with DTMF. DTMF is itself a viable two-frequency alternative for voice-call control [11], and a single DTMF digit has sufficient symbol capacity to represent the eight target IDs.

We retained the previous encoding in this study to preserve compatibility with the existing customer-side decoder and actuation system and to keep the evaluation focused on the operator-side audio injection method. The current M5StickC decoder analyzes 1500, 2000, and 2500 Hz components using FFT bins spaced at 250 Hz. The approximately 500 Hz separation between these components is compatible with the coarse frequency resolution of the current embedded decoder. Standard DTMF frequencies are more closely spaced within each frequency group and cannot be directly distinguished using the current FFT-bin configuration; adopting DTMF would therefore require redesigning the receiver-side frequency detection, for example by using finer-resolution analysis or a dedicated DTMF detector. The current protocol also includes a start symbol and an even-parity bit, providing application-level framing and limited error detection that a single DTMF digit by itself does not provide. A multi-digit DTMF protocol with redundancy or a checksum could, however, provide equivalent functionality.

DTMF is familiar as an ordinary telephone tone and could potentially reduce transmission time when a single digit is used. In contrast, the current frequency combinations were intended to be relatively unobtrusive, but the present study does not compare the two encodings perceptually; therefore, their relative user comfort remains unknown. Importantly, the proposed stream-injection architecture is not inherently tied to the current encoding and could also transmit DTMF waveforms after corresponding changes to the receiver and communication protocol. Future work should compare both encodings under the same call route, injection method, framing, and error-detection conditions in terms of transmission reliability, latency, and perceived annoyance.

5.3 Limitations and Practical Implications

This experiment has several limitations. First, only one set of equipment was used, and generalizability across different smartphone models, OS versions, call routes, speaker volumes, and microphone placements has not been verified. In addition, the two methods differed in the operator-side leg of the call route (the mobile carrier's voice network and a VoIP leg), so the observed differences cannot be attributed to the microphone input path alone; acoustically feeding the signal tones into the microphone during a call placed by the proposed application would isolate the input path under an identical call route, which remains as future work. Second, the noisy environment was reproduced in the laboratory by playing back, from an iPhone, the operating sound of a noodle-bundling machine recorded in a factory; it does not fully reproduce the sound propagation, reflections, and frequency characteristics of an actual factory. In addition, we did not compare conditions with stepwise changes in the noise level, so the relationship between noise intensity and reception performance remains unclear. Third, because IDs

0 to 7 were transmitted only once per condition in this evaluation, confidence intervals for the success and misrecognition rates cannot be discussed.

The tone duration used in practical operation should be determined by balancing transmission time against the risk of misrecognition. In actual telephone support, shortening the transmission time is important because the signal tones may temporarily interrupt the conversation. At the same time, it is also important not to indicate a wrong location. Therefore, the operating parameters should not be chosen solely to minimize transmission time, but should be decided after repeated experiments confirm the success rate and the occurrence of silent errors.

A representative use case is repeated telephone troubleshooting of industrial machinery installed at a remote production site where stable internet connectivity is unavailable. The customer-side receiver can be kept with the supported machine, while compact LED or buzzer modules can be installed in advance on components that commonly require inspection and reused across multiple support sessions. During a call, the customer places an ordinary telephone in speaker mode near the receiver; when the operator selects a target ID, the LED or buzzer on the corresponding component is activated. The customer can thereby locate the physical position indicated by the operator without installing a customer-side application, connecting the machine to the internet, or setting up a video call. This use case is particularly suited to machines with many visually similar components, for which verbal descriptions alone can leave the target location ambiguous and the preinstalled hardware can be reused over repeated support sessions.

6 Conclusion

This paper presented call-integrated audio stream injection for smartphone-based remote pointing over telephony channels. A custom iOS application mixes internally generated signal tones directly into the outgoing call audio while retaining the existing customer-side decoder and actuators.

In both quiet and simulated noisy conditions, the shortest tone duration with 8/8 successful decodes decreased from 200 ms for the acoustic-input baseline to 75 ms for the proposed method. Because each ID was transmitted only once per condition, these findings should be regarded as preliminary; nevertheless, they indicate that call stream injection can improve transmission stability at short tone durations.

In future work, we will statistically evaluate the success rate and the occurrence of silent errors through repeated and randomized trials, including stepwise changes in the noise level and different smartphone models and call routes. We will also compare the current encoding with a DTMF-based protocol and investigate improved receiver-side time resolution, error detection and correction, redundancy, and retransmission strategies, and clarify operating conditions under which the system can be used safely and quickly in actual telephone support.

References

1. Ahn, S., Kim, D., Ahn, S.H.: Dynamic inaudible frequency shifting communication for multi-robot collaboration in manufacturing. Manufacturing Letters **44**, 1225–1231 (2025). https://doi.org/10.1016/j.mfglet.2025.06.142
2. Bai, Y., Liu, J., Lu, L., Yang, Y., Chen, Y., Yu, J.: BatComm: enabling inaudible acoustic communication with high-throughput for mobile devices. In: Proceedings of the 18th Conference on Embedded Networked Sensor Systems. pp. 205–217. SenSys '20 (2020). https://doi.org/10.1145/3384419.3430773
3. Bain, A.: Certain Improvements in Producing and Regulating Electric Currents and Improvements in Electric Time-Pieces and in Electric Printing and Signal Telegraphs. British Patent No. 9745 (1843)
4. Hanspach, M., Goetz, M.: On Covert Acoustical Mesh Networks in Air. arXiv preprint arXiv:1406.1213 (2014). https://doi.org/10.48550/arXiv.1406.1213
5. Ives, H.E., Horton, J.W., Parker, R.D., Clark, A.B.: The Transmission of Pictures over Telephone Lines. Bell System Technical Journal **4**(2), 187–214 (1925). https://doi.org/10.1002/j.1538-7305.1925.tb00946.x
6. Katzschmann, R.K., DelPreto, J., MacCurdy, R., Rus, D.: Exploration of underwater life with an acoustically controlled soft robotic fish. Science Robotics **3**(16), eaar3449 (2018). https://doi.org/10.1126/scirobotics.aar3449
7. Matsuda, Y., Miyake, M., Tsuzuki, T., Kawakami, Y.: Remote Pointing over Telephony Channels: Audible-Sound Communication for Situated Industrial Support. In: Lecture Notes in Computer Science, The 28th International Conference on Human-Computer Interaction (HCII '26). vol. 16727, pp. 90–104 (2026). https://doi.org/10.1007/978-3-032-30038-6_6
8. Morise, M., Yamashita, Y.: Proposal on a lyric-synchronized music listening system using a watermark based on mosquito sound. IPSJ SIG Technical Reports **2012-MUS-96**(4), 1–6 (2012), `https://ipsj.ixsq.nii.ac.jp/records/83401`, (in Japanese)
9. Okano, Y., Teramoto, S., Noda, J.: Ad-hoc aerial acoustic communication for long-range propagation. In: IPSJ SIG Technical Reports, 2013-UBI-37(50). pp. 1–7 (2013), `https://ipsj.ixsq.nii.ac.jp/records/90985`, (in Japanese)
10. Suzuki, R., Sakamoto, D., Igarashi, T.: AnnoTone: Record-time Audio Watermarking for Context-aware Video Editing. In: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems. pp. 57–66. CHI '15 (2015). https://doi.org/10.1145/2702123.2702358
11. Yang, S.R., Yuan, S.C., Lin, Y.C., Yang, I.F.: DTMFTalk: a DTMF-based realization of IoT remote control for smart-home elderly care. Mobile Networks and Applications **27**(1), 196–207 (2022). https://doi.org/10.1007/s11036-020-01641-0