DTMFTalk: A DTMF-Based Realization of IoT Remote Control for Smart Elderly Care

Shih-Chun Yuan[1], Shun-Ren Yang[1,2](✉), I-Fen Yang[1], and Yi-Chun Lin[1]

[1] Department of Computer Science, National Tsing Hua University, Hsinchu 30010, Taiwan
[2] Institute of Communications Engineering, National Tsing Hua University, Hsinchu 30010, Taiwan
sryang@cs.nthu.edu.tw

Abstract. Smart elderly care becomes a popular technology to remotely assist the aged at home in recent years because of the population ageing. This paper demonstrates the DTMF-based realization of IoT remote control for telecommunication operators to achieve the smart elderly-care service. By utilizing IoTtalk, we implement a system, DTMFTalk, which supports IoT remote control via conventional circuit-switched DTMF signaling during a phone call conversation. DTMFTalk can monitor the Call State of the smart phone, capture DTMF keys from the in progress call, and send the key values to the IoTtalk server. Afterwards, the smart elderly-care devices can gain the related IoT instructions from the IoTtalk server. Through the delay measurement experiment for DTMFTalk, we observe that DTMFTalk can constantly and accurately recognize the DTMF keys as long as the user holds the desired DTMF keys with enough period.

Keywords: Internet of Things (IoT) · Smart elderly care · Dual Tone Multi-Frequency (DTMF) · Remote control

1 Introduction

In recent years, the smart elderly-care services [2,3,8,9] start developing along with the population ageing. It is becoming more and more common that the aged live alone or family members leave the aged staying at home alone due to working. It is inconvenient for most of the aged to manipulate a lot of essential equipment at home, and danger may even happen in case of carelessness. Thus, the elderly-care service becomes a popular technology to take care of the aged at home. The service can combine some IoT techniques with medical devices, home devices and emergency notification facilities to establish the smart elderly-care service, which can achieve the goal of remotely assisting the aged.

D.-J. Deng et al. (Eds.): SGIoT 2019, LNICST 324, pp. 16–27, 2020.
https://doi.org/10.1007/978-3-030-49610-4_2

Although the smart elderly-care service can integrate with IoT services, there are some existing problems for IoT. The transporting paths of IoT services may involve the public network which cannot guarantee the quality of service (QoS), and various network security problems also exist. Additionally, most of the users install Apps on their smart phones to access IoT services. While the foreground Apps will consume great amounts of power, and the background Apps may cause the users to miss real-time events. Moreover, the cost is very high for managing and maintaining end-to-end Internet-based IoT service platforms with telecom-grade quality. Thus, extra developing the extensive systems is not cost effective for Internet-based IoT service providers.

Circuit-switched telephony network is one of the successful telephony network mechanism in history, and it offers some solutions to existing IoT service problems [6]. For Qos, the transporting paths between the CPEs and the switches are based on the private network, which keeps telecom-grade quality and security. With the telephone number provided by the trust telecommunications operator and routed by their switches, most network security dangers can be blocked. Besides, the CPEs can handle real-time events with lower power consumption like the condition that the users access IoT services deployed on the CPEs through the telephone numbers. Moreover, it is easier and lower cost for telecommunication operators to accommodate IoT services in the existing extensive telecom systems. The related charging mechanisms have been developed for over 100 years and become reliable and flexible, so that IoT services can be charged without extra overhead [7].

In this paper, we propose the approach for telecommunication operators to integrate Dual Tone Multi-Frequency (DTMF) signaling with IoT [5,10] for Smart Elderly Care. DTMF signaling based on circuit-switched telephony network can also solve some existing IoT problems. DTMF signaling supplies push-button telephones with much higher dialing speed than the dial-pulse signaling used in conventional rotary telephone sets, and it prompts users to select options from menus by sending appropriate DTMF signals from their telephones. Due to the DTMF signaling applications in various fields for years, it becomes one of the familiar technologies for the aged. The smart elderly-care service is provided for the aged, so it is more appropriate for us to choose DTMF signaling to coordinate with the aged habits. Furthermore, we design a system to support IoT remote control via conventional circuit-switched DTMF signaling during a phone call conversation, and we choose to utilize IoTtalk as our IoT service platform.

The following describes the paper organization. Sect. 2 illustrates our system architecture. Sect. 3 further explains how our system captures DTMF signals. Sect. 4 provides the conclusion of our system.

2 The Software Architecture of DTMFTalk

To support IoT remote control via conventional circuit-switched DTMF signaling during a phone call conversation, we design and implement an Android App, DTMFTalk, following the application development/execution framework

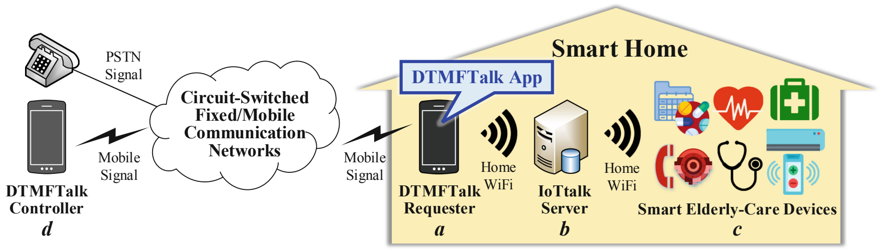


Fig. 1. System architecture

of IoTtalk. Figure 1 illustrates our system architecture, which consists of four components: a DTMFTalk requester a, an IoTtalk server b, several smart elderly-care devices c, and a DTMFTalk controller d. The DTMFTalk requester a is an Android smart phone installed with our DTMFTalk App, while the DTMFTalk controller d is a normal cell or PSTN phone. When the DTMFTalk requester a and the DTMFTalk controller d are in a circuit-switched phone call conversation, the DTMFTalk App is responsible for capturing the DTMF keys embedded inside the audio stream of the call from d, and further transferring the captured DTMF keys to the IoTtalk server b. According to the DTMF keys, the IoTtalk server b continuously manipulates the expected smart elderly-care devices c.

2.1 Overview of the DTMFTalk Software Architecture

The main tasks of DTMFTalk in the DTMFTalk requester a are: (1) to detect if the DTMFTalk controller d (after the authentication and authorization procedures) operates a call conversation with the DTMFTalk requester a, and (2) to capture the DTMF keys from the audio stream and send the keys to the IoTtalk server b. Figure 2 shows the detailed software modules of our DTMFTalk. As illustrated in Fig. 2, DTMFTalk implements two Android application components: the Call Detecting Service (CDS; see Fig. 2(I)) and the DTMF Handling Activity (DHA; see Fig. 2(II)), which handle the above mentioned Tasks 1 and 2, respectively. Note that a typical Android smart phone exercises a state machine for call control in the Android telephony service. Thus, to achieve Task 1, the CDS of DTMFTalk constantly monitors the state machine to retrieve the current call status. The CDS is essentially an Android service, which is responsible for detecting if a call is in progress, and controlling when to perform the DHA. Initially, after DTMFTalk is activated, the CDS is executed in the back end. Whenever a starts to engage in a phone call conversation, the CDS will call to perform the DHA until the phone call conversation is completed. On the other hand, the DHA is an Android activity, which implements the IoTtalk device-side IDA and DA functionalities to capture DTMF keys and send those keys to the IoTtalk server b, respectively. Besides, it provides a user interface for displaying DTMF key related information to the user of a.

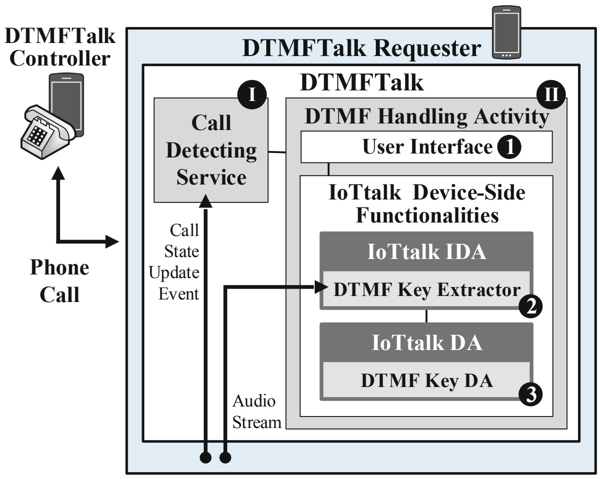


Fig. 2. Software modules of DTMFTalk

2.2 The Functionalities of the CDS

The CDS should, in the back end, continually monitor the call state machine (in particular, its state transitions) in the Android telephony service. The Android telephony call state machine contains three states: the IDLE state, the RINGING state, and the OFFHOOK state. The IDLE state represents that the Android phone has no call activities; the RINGING state represents that a new call arrival occurred and is ringing or waiting; the OFFHOOK state represents that at least one call exists which is dialing, active, or on hold, and no calls are ringing or waiting. By tracking the call state updates, the CDS can be aware if a call is ongoing. We accomplish this via the Android broadcast mechanism. In Android, the system and Apps can send broadcast messages when events of interest, e.g., a call state update in our case, occur. Other Apps can then utilize the Android component `BroadcastReceiver` to register to receive these specific broadcasts. In the implementation, our CDS extends (or inherits) the `BroadcastReceiver` class, in which we specify a receiver that can be notified of a system broadcast message in case a call state update event occurs.

By parsing the received broadcast messages of the call state update events, the CDS can determine the new state after each state update and proceed accordingly. Given the initial state IDLE, a typical state-update sequence and the corresponding reaction of the CDS upon each update are illustrated as follows.

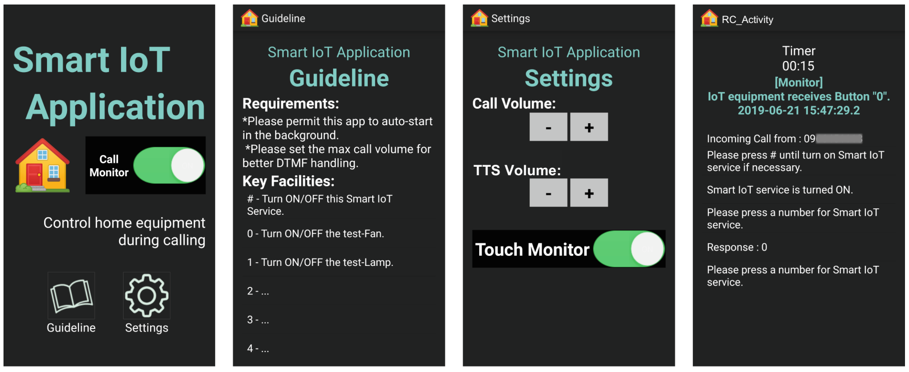
– Update 1 - The new state is RINGING: The DTMFTalk controller invites the DTMFTalk requester, or vice versa, for a call. The CDS can first authenticate and authorize the DTMFTalk controller's telephone number (retrieved via the Caller ID telephone service) before activating the subsequent DTMFTalk tasks.

– Update 2 - The new state is OFFHOOK: The DTMFTalk requester or the DTMFTalk controller accepts the call invitation. The CDS will launch the DHA.
– Update 3 - The new state is IDLE: The DTMFTalk requester or the DTMFTalk controller terminates the call. The CDS will then end the DHA.

2.3 The Functionalities of the DHA

The DHA implements the IoTtalk IDA and DA functionalities for DTMF key capture and delivery to the IoTtalk server, requesting the IoTtalk server to conduct the expected IoT remote control. In fact, the DHA manipulates an IDF *DTMFKey* from the perspective of IoTtalk. Therefore, the main task of the DHA is to update and synchronize the value of the corresponding *DTMFKey* IDF module in the IoTtalk engine. The detailed settings/configurations within the IoTtalk engine for the proper operation of the DHA will be discussed in the later sections. In the following, we summarize the functionalities of the DHA's three components: the *User Interface (UI)*, the *DTMF Key Extractor*, and the *DTMF Key DA*.

– **UI** (Fig. 2(II-1)). The UI is responsible for displaying App views. Besides, the UI also incorporates into the Android component *Text To Speech (TTS)* capability to transform the content of the App views into sounds, assisting the user of the DTMFTalk requester (an elderly person) to use the DTMFTalk App in a more user-friendly manner. Figure 3 shows a snapshot of the UI.


Smart IoT Application
Call Monitor
Control home equipment during calling
Guideline Settings

Guideline
Smart IoT Application
Guideline
Requirements:
*Please permit this app to auto-start in the background.
*Please set the max call volume for better DTMF handling.
Key Facilities:
- Turn ON/OFF this Smart IoT Service.
0 - Turn ON/OFF the test-Fan.
1 - Turn ON/OFF the test-Lamp.
2 - ...
3 - ...
4 - ...

Settings
Smart IoT Application
Settings
Call Volume:
- +
TTS Volume:
- +
Touch Monitor

RC_Activity
Timer
00:15
[Monitor]
IoT equipment receives Button "0".
2019-06-21 15:47:29.2
Incoming Call from : 09[redacted]
Please press # until turn on Smart IoT service if necessary.
Smart IoT service is turned ON.
Please press a number for Smart IoT service.
Response : 0
Please press a number for Smart IoT service.

(a) Home Page (b) Guideline Page (c) Settings Page (d) RC Activity Page

Fig. 3. DTMFTalk user interfaces

– **DTMF Key Extractor** (Fig. 2(II-2)). The DTMF Key Extractor uses the Android `AudioRecord` object to access the audio pieces from the audio stream of the ongoing phone call. Afterwards, the DTMF Key Extractor analyzes the audio pieces through the Fast Fourier Transform to determine whether DTMF keys are contained in the call. Moreover, the DTMF Key Extractor can also pass DTMF key related information to the UI, updating the content of the App views. The details of this component will be given in Sect. 3.

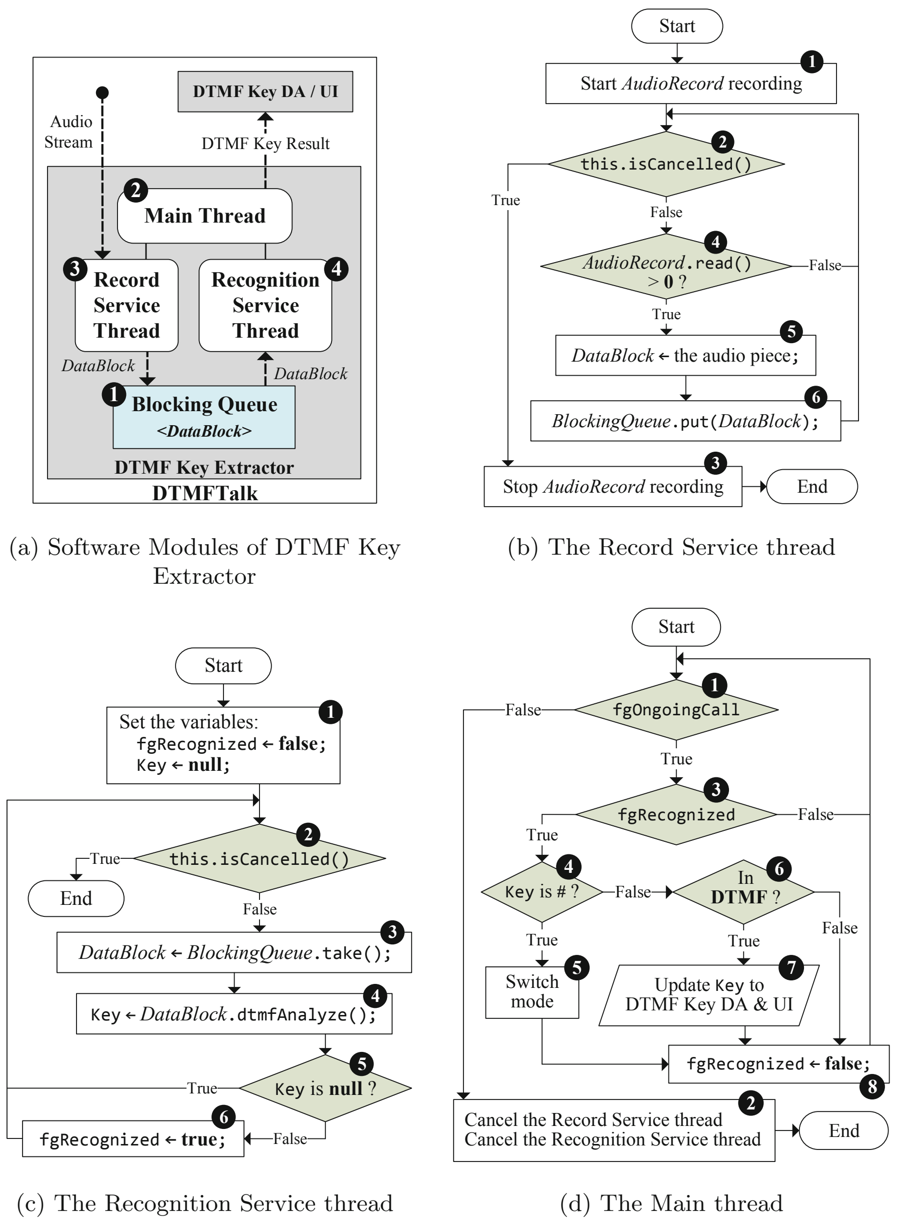
- **DTMF Key DA** (Fig. 2(II-3))**.** The DTMF Key DA is responsible for updating the DTMF key value to the IoTtalk engine. The DTMF Key DA first formulates an HTTP-based RESTful API request by attaching the DTMF key value derived by the DTMF Key Extractor in the message body. Then, the DTMF Key DA delivers the request to the DTMFKey IDF module in the IoTtalk engine to synchronize the IDF value.

3 The Design and Implementation of the DTMF Key Extractor

This section details how we design and implement the DTMF Key Extractor in the DHA of DTMFTalk to capture the DTMF keys from the audio stream of an ongoing phone call. We note that some existing Android Apps, e.g., "DTMF decoder" [4] and "MobIVRS" [12], can already achieve this. These Apps typically perform sound recording first and then conduct decoding analysis on the recorded audio stream. Our DTMF Key Extractor mainly follows the framework of the "MobIVRS" App, which is the most complete open source among the consider Android Apps. However, the "MobIVRS" App, which is actually based on the older Android version, operates with some more inefficient methods and components for most of the current smart phones. Besides, human voices can easily lead the "MobIVRS" App to incorrectly recognize the DTMF keys. In the DTMF Key Extractor, we rearrange the application framework to improve operational efficiency. Moreover, we also introduce and implement the two modes, the *TALKING Mode* and the *DTMF Mode*, in the Main thread to promote the recognition accuracy.

3.1 The Operation of the DTMF Key Extractor

As Fig. 4a illustrates, the DTMF Key Extractor operates a Java-library Blocking Queue (Fig. 4a(1)) and three threads, one for the Main thread (Fig. 4a(2)) and two for the Record Service (Fig. 4a(3)) and the Recognition Service (Fig. 4a(4)), to achieve its tasks. The Blocking Queue is employed to store the *DataBlock* objects, encapsulating the audio pieces recorded by the Record Service, for DTMF key recognition by the Recognition Service. The Blocking Queue has the property that if full, it will block the "producer" (i.e., the Record Service in our case) from putting new objects until the space is released by the "consumer" (i.e., the Recognition Service in our case). In the following, we explain the operation the DTMF Key Extractor in terms of the interactions of its three concurrent threads.


DTMF Key DA / UI
Audio Stream
DTMF Key Result
Main Thread
Record Service Thread
Recognition Service Thread
DataBlock
Blocking Queue <DataBlock>
DTMF Key Extractor
DTMFTalk
Start
Start AudioRecord recording
this.isCancelled()
True
False
AudioRecord.read() > 0 ?
DataBlock ← the audio piece;
BlockingQueue.put(DataBlock);
Stop AudioRecord recording
End
Set the variables: fgRecognized ← false; Key ← null;
DataBlock ← BlockingQueue.take();
Key ← DataBlock.dtmfAnalyze();
Key is null ?
fgRecognized ← true;
fgOngoingCall
fgRecognized
Key is # ?
In DTMF ?
Switch mode
Update Key to DTMF Key DA & UI
fgRecognized ← false;
Cancel the Record Service thread
Cancel the Recognition Service thread

(a) Software Modules of DTMF Key Extractor

(b) The Record Service thread

(c) The Recognition Service thread

(d) The Main thread

Fig. 4. Software modules and operations of the DTMF key extractor

The Record Service Thread. The Record Service thread employs an Android `AudioRecord` object to record the audio stream of the ongoing call. According to the designated audio source, sampling rate, encoding audio format, etc., the `read()` member function of the `AudioRecord` object allows to obtain (from the audio stream) a sampled, quantized, and encoded audio piece in signal format (0 or 1). The Record Service thread then formulates this audio piece as a *DataBlock* object for storing in the Blocking Queue. The operation of the Record Service thread is illustrated in Fig. 4b and is explained as follows. In Step 1, the Record Service thread starts recording by creating an *AudioRecord* object and executing its `startRecording()` method. Next, Steps 2–6 repeatedly read/transform an audio piece and store it in the Blocking Queue until the Record Service thread is terminated by the Main thread. Step 2 first checks whether the thread is cancelled by the Main thread. If **true**, Step 3 will stop recording using the *AudioRecord* object (via the `stop()` method), and then the thread ends; otherwise, Step 4 attempts to read an audio piece from the *AudioRecord* object. In Step 4, if the return value of the `read()` method equals **0**, nothing can be read and the procedure moves back to Step 2 for the next iteration; otherwise, Step 5 transforms the recorded audio piece (stored in a buffer) into a *DataBlock* object, and Step 6 puts the *DataBlock* object into the Blocking Queue.

The Recognition Service Thread. The Recognition Service thread takes and analyzes each *DataBlock* object from the Blocking Queue, checking if any DTMF tone sound is embedded in the audio stream. In case a DTMF tone sound is found, the Recognition Service thread will pass the DTMF key value to the Main thread. Note that the Recognition Service thread uses a Fast Fourier Transform (FFT) to analyze each *DataBlock* object based on the time domain, which will be detailed in the next subsection. The operation of the Recognition Service thread is illustrated in Fig. 4c and is explained as follows. Step 1 initializes the two involved variables, `fgRecognized` (a flag) and `Key`, as **false** and **null**, respectively. Next, Steps 2–6 repeatedly retrieve and analyze a *DataBlock* object from the Blocking Queue until the Recognition Service thread is terminated by the Main thread. Step 2 first checks whether the thread is cancelled by the Main thread. If **true**, the thread ends; otherwise, Step 3 uses the `take()` method to retrieve a *DataBlock* object from the Blocking Queue. Then, Step 4 analyzes the content of the *DataBlock* object, attempting to derive the embedded DTMF key value and store the result in `Key`. Step 5 further determines if `Key` contains a **null** value. If not, a valid DTMF key value is found, and Step 6 will update the shared flag variable `fgRecognized` as **true** to reflect this condition to the Main thread.

The Main Thread. The Main thread controls the operations of the Record Service and Recognition Service threads, and updates valid DTMF key values to the DTMF Key DA and the UI. We note that in a wireless environment, with non-negligible probabilities, normal human audio pieces and even background environment noises can be coincidentally recognized as valid DTMF keys. If the users can first "reveal" when they intend to remotely control (via the DTMF keys) the smart elderly-care devices, the above-mentioned incorrectly identified DTMF keys can be easily ignored, leading to a significantly reduced false DTMF-key detection probability. Besides, in this case, the DTMF Key Extractor can be allowed to only focus on those meaningful DTMF key detections when the users are indeed operating DTMF keys. For this, the Main thread implements two modes: the TALKING mode and the DTMF mode. When the two parties of the call are in conversation, the Main thread stays at the TALKING mode; on the other hand, when the user of the DTMFTalk controller is pressing DTMF keys for remote control, the Main thread stays at the DTMF mode. Moreover, the specific DTMF key '#' is employed to switch between the two modes.

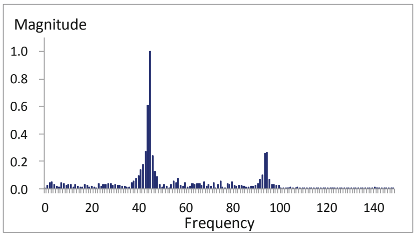
The operation of the Main thread is illustrated in Fig. 4d and is explained as follows. The whole thread repeatedly checks the call status, switches the current mode if necessary, and updates a recognized DTMF key to the DTMF Key DA and the UI until the call is completed. Step 1 first checks the status of the flag, `fgOngoingCall`, which is controlled by the CDS (see Fig. 2(I)), shared with the Main thread of the DTMF Key Extractor, and is used to indicate if the monitored call is still ongoing. If **false**, Step 2 cancels the Record Service and Recognition Service threads, and then the Main thread ends; otherwise, Step 3 checks if the status of the flag, `fgRecognized`, has been set as **true** by the Recognition Service thread, indicating that a DTMF key has been successfully recognized. If a DTMF key has been detected and Step 4 finds that the key is the control key '#', Step 5 switches the mode from **TALKING** to **DTMF**, or from **DTMF** to **TALKING**, depending on the current mode. However, if another DTMF key is found and the current mode is **DTMF** (i.e., the IoT remote control mode) at Step 6, the key can be regarded as a valid key and will be reported to the DTMF Key DA and the UI at Step 7. Finally, before moving back to Step 1 for the next iteration, the `fgRecognized` flag should be reset as **false** at Step 8 to allow the Recognition Service thread to recognize the next DTMF key. Note that for the synchronization among the Main, the Record Service, and the Recognition Service threads, some sophisticated constructs are implemented for the protection of the shared variables. Nevertheless, these details are not presented in this paper due to space limitation.

3.2 The Fast Fourier Transform for DTMF Signal Detection

In Step 4, the Recognition Service thread (see Fig. 4c) performs Fourier analysis for DTMF signal detection [1]. As shown in Fig. 5a, a DTMF signal consists of two tones - with frequencies taken from two mutually exclusive groups: one frequency from the low group (697 Hz, 770 Hz, 852 Hz, 941 Hz), while the other frequency from the high group (1209 Hz, 1336 Hz, 1477 Hz). For example, pressing '3' will generate a 697-Hz tone from the low frequency group along with a 1477-Hz tone from the high frequency group at the same time. Based on such DTMF working principle, the DTMF signal detection in the Recognition Service thread can be achieved as follows.

	1209 Hz	1336 Hz	1477 Hz
697 Hz	1	2	3
770 Hz	4	5	6
852 Hz	7	8	9
941 Hz	*	0	#

(a) DTMF keypad



(b) The *spectrum* array

Fig. 5. The fast Fourier transform for DTMF signal detection

– When a *DataBlock* object encapsulating the sampled/encoded audio piece for a DTMF signal in the time domain is given, the Cooley-Tukey algorithm [11,13], the most commonly used fast Fourier transform (FFT), is applied to compute the discrete Fourier transform (DFT), a frequency domain representation, of the DTMF signal. This allows to reveal the frequency components that are present in the *DataBlock* object. The output of the Cooley-Tukey FFT algorithm is stored in a `double` array, *spectrum*, of size 150, where the array indices of *spectrum* represent the frequency samples (unit: 15.4 Hz) while the array value of *spectrum* under a given array index represents the corresponding DFT magnitude normalized within [0, 1]. As an example, Fig. 5b illustrates the content of the *spectrum* array after the DTMF signal of '3' is transformed and recorded.
– To determine the corresponding low and high frequencies for the DTMF signal represented by *spectrum*, the Recognition Service thread divides the array indices of *spectrum* into two halves: the low frequencies [0, 74] and the high frequencies [75, 149]. Afterward, the Recognition Service thread searches the frequency with the maximum DFT magnitude in each half, and saves the obtained low frequency in the first half and high frequency in the second half into $lowMax$ and $highMax$, respectively. The $lowMax$ and $highMax$ can be used to look up the corresponding mapping in the DTMF frequency table in Fig. 5a. If matched, the DTMF signal can then be recognized. For example, in Fig. 5b, $lowMax$ contains 45 with respect to a frequency value approximately 697 Hz, while $highMax$ contains 95 with respect to a frequency value approximately 1477 Hz, which matches the frequency pair of the DTMF signal '3'. Thus, the Recognition Service thread can store the valid key character '3' in Key.

3.3 Real Testbed Deployment

We have implemented the functionalities of DTMFTalk, and deployed a real testbed (as shown in Fig. 6) to justify the feasibility of our DTMFTalk as an approach to IoT remote control, especially for elderly-care applications.

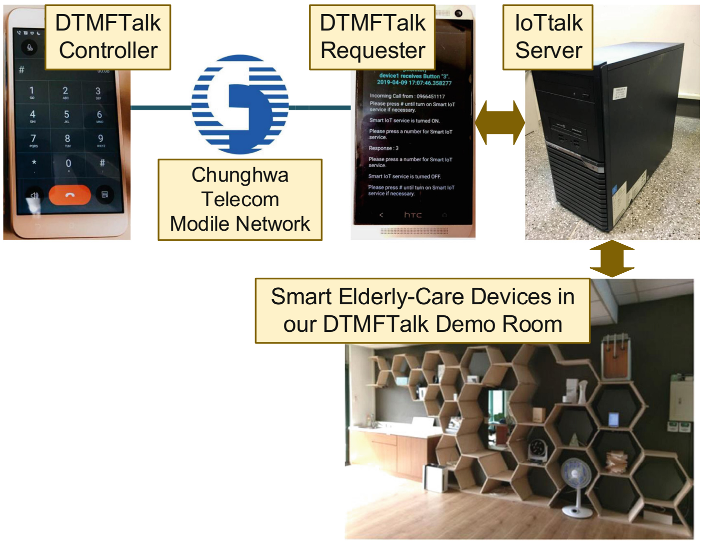

DTMFTalk Controller
Chunghwa Telecom Modile Network
DTMFTalk Requester
IoTtalk Server
Smart Elderly-Care Devices in our DTMFTalk Demo Room

Fig. 6. A real testbed deployment

4 Conclusion

This paper demonstrates how we achieve the integration between DTMF signaling and IoT service. By choosing IoTtalk as the IoT service platform, we design a system called DTMFTalk to support IoT remote control via conventional circuit-switched DTMF signaling during a phone call conversation. DTMF signaling is operated through telephony network, which offers some solutions to existing IoT service problems, such as QoS, network security, power consumption, charging mechanism, maintaining cost and so on. DTMFTalk can monitor the Call State of the smart phone, capture DTMF keys from the in progress call, and send the key values to the IoTtalk server for IoT controlling. Therefore, when DTMFTalk executes in the phone call conversation, the user of the DTMFTalk controller is able to remotely control the smart elderly-care devices by pressing the specific DTMF keys according to the requests coming from the user of the DTMFTalk requester. We perform the delay measurement, which shows that the DTMFTalk controllers can efficiently operate DTMFTalk (within 3 s). Besides, we also explore the background environment noise effect for DTMFTalk. In conclusion, DTMFTalk can constantly and accurately recognize the DTMF keys as long as the user of the DTMFTalk controller holds the desired DTMF keys with enough period.

Acknowledgment. The authors would like to thank Prof. Yi-Bing Lin, CS, NCTU, Taiwan, for his invaluable discussions and suggestions. This work was supported in part by the Ministry of Science and Technology (MOST), Taiwan, under Contracts MOST-108-2221-E-007-034-, MOST-107-2221-E-007-117-MY3, and MOST-106-2923-E-002-005-MY3, in part by the Ministry of Economic Affairs (MOEA), Taiwan, under Contract MOEA-107-EC-17-A-02-S5-007, and in part by the Ministry of Education Higher Education Sprout Project.

References

1. Bhavanam, S.N., Siddaiah, P., Reddy, P.R.: FPGA based efficient DTMF detection using split goertzel algorithm with optimized resource sharing approach. In: 2014 Eleventh International Conference on Wireless and Optical Communications Networks (WOCN), pp. 1–8, September 2014. https://doi.org/10.1109/WOCN.2014.6923072
2. Borelli, E., et al.: Habitat: an IoT solution for independent elderly. Sensors **19**(5), 1258 (2019). https://doi.org/10.3390/s19051258
3. Ghasemi, F., Rezaee, A., Rahmani, A.M.: Structural and behavioral reference model for IoT-based elderly health-care systems in smart home. Int. J. Commun. Syst. **32**(12), e4002 (2019). https://doi.org/10.1002/dac.4002. https://onlinelibrary.wiley.com/doi/abs/10.1002/dac.4002
4. Jasiun, P.: DTMF decoder (2011). https://github.com/pjasiun/dtmf-decoder
5. Johar, R.A., Fakieh, E., Allagani, R., Qaisar, S.M.: A smart home appliances control system based on digital electronics and GSM network. In: 2018 15th Learning and Technology Conference (L&T), pp. 52–58, February 2018. https://doi.org/10.1109/LT.2018.8368485
6. Lin, Y., Lin, R., Chen, Y., Twu, C., Yang, S.: Deploying the first PSTN-based IoT mechanism. IEEE Wirel. Commun. **25**(6), 4–7 (2018). https://doi.org/10.1109/MWC.2018.8600748
7. Lin, Y.B., Sou, S.I.: Charging for Mobile All-IP Telecommunications. Wiley, Chichester (2008)
8. Liu, Y., et al.: A novel cloud-based framework for the elderly healthcare services using digital twin. IEEE Access **7**, 49088–49101 (2019). https://doi.org/10.1109/ACCESS.2019.2909828
9. Lu, Y., Lin, C.: The study of smart elderly care system. In: 2018 Eighth International Conference on Information Science and Technology (ICIST), pp. 483–486, June 2018. https://doi.org/10.1109/ICIST.2018.8426110
10. Noman, A.T., Rashid, H., Chowdhury, M.A.M., Islam, M.S.: Design implementation of a microcontroller based low cost DTMF controlled acoustic visual detecting robot to monitor child aged person. In: 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), pp. 1–6, February 2019. https://doi.org/10.1109/ECACE.2019.8679401
11. Sedgewick, R., Wayne, K.: Algorithms, 4th edn. Addison-Wesley Professional, Reading (2011)
12. Sharma, N., Jain, R., Garg, M., Arya, S.: Mobivrs (2015). https://github.com/SharmaNishant/MobIVRS
13. Sorensen, H., Heideman, M., Burrus, C.: On computing the split-radix FFT. IEEE Trans. Acoust. Speech Signal Process. **34**(1), 152–156 (1986). https://doi.org/10.1109/TASSP.1986.1164804