Flashlight: A Wireless Communication Method

JING ZHANG, Binghamton University
SHUOQIAN WANG, Binghamton University

We introduce a new technology to communicate between cellphones using ambient light sensor. We utilize flashlight on the cellphone to send messages to a receiver. We implement it using iOS Flashlight API to send signal and using API of I/O receiving the signal. This implementation is more secure than using network communication, but its speed need to be accelerated in future.

CCS Concepts: • **Hardware → Wireless integrated network sensors**; *Wireless devices* ;

Additional Key Words and Phrases: Wireless Sensor Networks, Flashlight API, Cellphone Communication

ACM Reference format:
Jing Zhang and Shuoqian Wang. 2017. Flashlight: A Wireless Communication Method. , , Article (May 2017), 2 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Normally there are two methods to communicate between devices: Bluetooth and network. How to transfer some information without these technologies? In this article, we propose that we can use flashlight to do communication between devices.

The main contributions of our work can be summarized as follow.

- To the best of our knowledge, our work is first communication application based on mobile devices, and it is also designed for WSN.
- It also realize communication from one device to multiple devices.
- Flashlight transmission can better ensure the security during transmitting information.
- We can transfer data without Wi-Fi by using our technology.

2 SYSTEM DESIGN

2.1 Development Tools and Devices

We use development tools as below:

- XCode 7.0
- Objective-C

We use devices as following:

- iPhone
- MacBook

2.2 System Architecture

Our system has two parts: sender and receiver.

Sender Using iOS APIs to control flashlight.
Receiver Get the flashlight signal, translating into '0' and '1'.

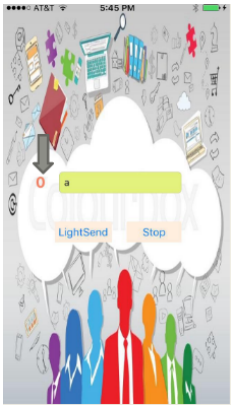

Fig. 1. Sender storyboard.

Here, we can also use Android to implement our system, however, in our case, we just testify our system using iOS. In future, we can realize our system in Android environment.

3 IMPLEMENTATION

We will introduce how we can implement sender and receiver using iOS development tools.

3.1 Sender Implementation

(1) Use 'TextField' to tap message.
(2) Use 'LightSend' iOS UIButton to realize sending flashlight.
(3) Use 'Stop' iOS UIButton to stop transferring flashlight.
(4) Use iOS UILabel to display the binary which is transferring.

Figure 1 shows storyboard for sender application. It contains input text and buttons to send messages and stop messages. Section 3.

3.1.1 Input Message. We use 'TextFiled' to accept input message and enable Keyboard to tap, then after tapping, we handle return key for keyboard to return.

3.1.2 Transition from Character to Flashlight Signal. There are two steps to transform character to flashlight.

(1) Get the string from 'TextFiled' and use a character array to store the string, at last use and operation to get from high to low binary of each character.
(2) There is a class, named AVCaptureDevice, which can control flashlight on and off for iphone. So we utilize the class when we get binary, we turn on flashlight for couple seconds, and turn it off for another couple seconds.

3.1.3 Stop Flashlight. In case that it gets error or it takes too long time to transfer message. We add a stop action to suspend the transmission. We use a value to mark whether customer stops sending or not; Every time click the 'Stop' button, it will change the value and stop the flashlight.

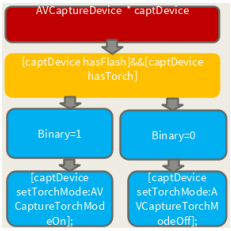

Fig. 2. Sender storyboard.

3.1.4 Display The Binary. The objective is that let customers know which binary the process is sending. And we use a UILabel to display the binary when flashlight is sending.

3.2 Receiver Implementation

The implementation on the receiver side is to get the digital signal '0' and '1' from origin signal. For the given sensor from a MacBook Air (11-inch, Mid 2012), the maximum light level value is limited from 1902 on totally dark space to 4091 on maxmium. About the sensor value, for instance, if any light source directly face to the sensor, the value will increase to be 4091. So all our tests and implementation were indoor and no light source directly face to the sensor. Thus the background light level value is around 2300 to 2500.

3.2.1 Signal Processing. In the tests, we found that it is not easy to figure out '0' and '1' correctlly. First of all, we are not going to implement syncronization part, because that will cause much overhead and we just want to implement a very simple communication method like UDP in internet. So, we cannot rely on any concept about time, like n miliseconds for '0' and m miliseconds for for '1'. But after some experiments, we found that the sensor is working cumulatively. For a short impulse, this short is not about time but the light energy been sent out, the light level value will increase by around 500. For a long impulse, the light value will increase by over 1000. But clearly, if we just use the value directly, the gap between 500 and 1,000 is not big enough. So we do a square to the difference of sensor value minus background value, then divide it by 10. Eventually, for a signal '0', the light level value is around 20,000 to 40,000, for a '1', it is over 100,000.

$$f(x) = \frac{x^2}{10}$$

3.2.2 Threshold Setting. After we finish the signal processing part, the most important thing is to select a threshold to differentiate '0' and '1'. Generally, in any kind of communication, bit error is an important problem. So as in our implementation, but after lots of experiments, we found that two reason mainly cause the bit error problem.

- Time interval is too small between two '0'
- The impulse of '1' is not much higher than '0'

For the first one we are going to discuss and solve it in next subsection, the impulse interval part. About the second one, we optimize it by set the threshold to be 70,000, lower as '0' and higher as '1'. Because we found that it is very rare that for a '0', it light level value is beyond 50,000 and for a '1', it is almost impossible to be below 80,000. And we still give 10,000 available space for a '1'. We need to remind that typically a '1' value would be over 110,000. According to the receiver, we modified the sender side's parameters. For a '0', the flashlight will stay light on for 110 miliseconds, and for a '1', it will stay light on for 440 miliseconds.

3.2.3 Impulse Interval. The last step is to set impulse interval, because the sensor works cumulatively, so it needs interval to let the light level value goes back to the background level. After our tests, we figured that if the sensor receives an impulse of n miliseconds, it needs around 1.2 to 1.5 times n miliseconds to restore to background level. Then we go back to sender side and modified the time interval after sending a '0' to be 140 miliseconds and 640 miliseconds after sending a '1'.

4 PERFORMANCE EVALUATION

For now, the performance of our communication system is very clear by computing from above data, 4 bps for signal of '0' and almost 1 bps for '1'. In the real tests, considering the distribution of '0' and '1' in the natural source, the bit rate is below 2.5 bps.

5 CONCLUSIONS

After this project, we believe that this kind of communication through light is very doable. All our tests were based on the sensor given by and controled by Apple. And after test, we know that its sample rate of the light sensor is only 10 Hz. So I think that is the reason why a signal '0' cannot below 110 miliseconds. If we can get the access to a sensor and set the sample to be 1000 Hz, we believe that a signal '0' can be sent within 5 miliseconds including the impulse interval. If a '0' is that short, a '1' will be much decreased. So we believe that the future of the light communication is very good, maybe can replace the LAN we use today which is based on electricity.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Mo Sha for providing specifications about the application design.

REFERENCES

Received May 2017; revised May 2017; final version May 2017; accepted May 2017