CS M117 Project Report

iVote: A Bluetooth voting app

Zhuoran Duan (404451176)

Zixuan Fan (904190493)

Likai Wei (204409560)

2016/5/28

Motivation

iClicker is now widely used in college classes, and many professors are using *iClicker* to collect answers from students. However, the product is expensive and inconvenient to carry around, so we had the idea of developing a similar app on iOS using Bluetooth. It can be used within a group and collect different opinions to a specific question. The Scenarios of using the app are classroom voting, meetings and anonymous polling. Also, it can be used without Wi-Fi connection and is relatively energy saving.

Wireless Network Used

We used the iOS Core Bluetooth Framework to implement our project. We choose Bluetooth instead of other wireless network based on the advantages of BTLE (Bluetooth Low Energy). First of all, the app can be used in circumstances where there is no WIFI/LTE connection since the communication is completely depends on Bluetooth. In addition, Bluetooth is a low energy technology so that it will not tremendously reduce your phone's battery life. Finally, since the app is designated for voting in scenarios like classroom and meeting, which are all in the communication

range of Bluetooth technology. Therefore, we can save the cost of renting server by just using iOS Core Bluetooth Framework.

Goals of the Project

The goal of the project is to implement an easy-to-use and lightweight iOS Bluetooth voting app. The user can either create a vote or join a vote. The creator of the vote is able to specify the details, including title, descriptions and different options. The creator is also able to track the result of the vote on the result page, which is being constantly updated. The votes are collected through Bluetooth, which is steady and reliable within small group and close range communications.

Design Process

Our design of the app has two main components: UI design and Bluetooth communication. Our UI and icons are designed using a software called Sketch, after that we implement the UI into Xcode's storyboard. Finally, we implement view controllers to achieve the voting features. All the process all done using Github.

Architecture Details

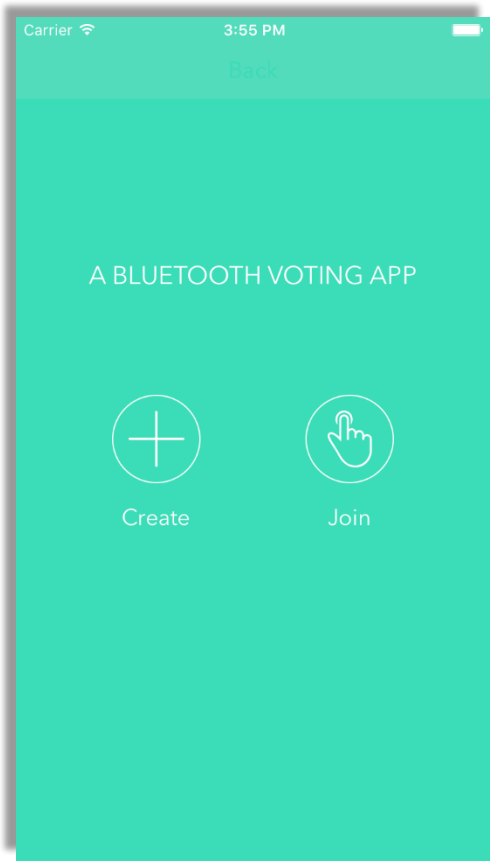
Bluetooth low energy wireless technology is based on the Bluetooth 4.0 specification, which, among other things, defines a set of protocols for communicating between low energy devices. The Core Bluetooth framework is an abstraction of the Bluetooth low energy protocol stack. That said, it hides many of the low-level details of the specification from you, the developer, making it much easier for you to develop apps that interact with Bluetooth low energy devices.

Bluetooth 4.0 has communication range of 100m (330ft) and data rate from 1 Mbit/s to 3 Mbit/s. In addition, each master can have up to 7 slaves in its piconet. Fast frequency hopping such as ACK is used in Bluetooth 4.0.

The core Bluetooth framework has two kinds of devices: central and peripheral. A peripheral typically has the data that is needed by other devices, and a central uses uses the information served up by a peripheral to accomplish some task. Peripherals make their presence known by advertising the data they have over the air. Centrals, on the other hand, can scan for peripherals that might have data they're interested in. When a central discovers such a peripheral, the central can request to connect with the peripheral and begin exploring and interacting with the peripheral's data. The peripheral is then responsible for responding to the central in appropriate ways. In this case, the creator of the vote is central, while the voting users are peripheral.

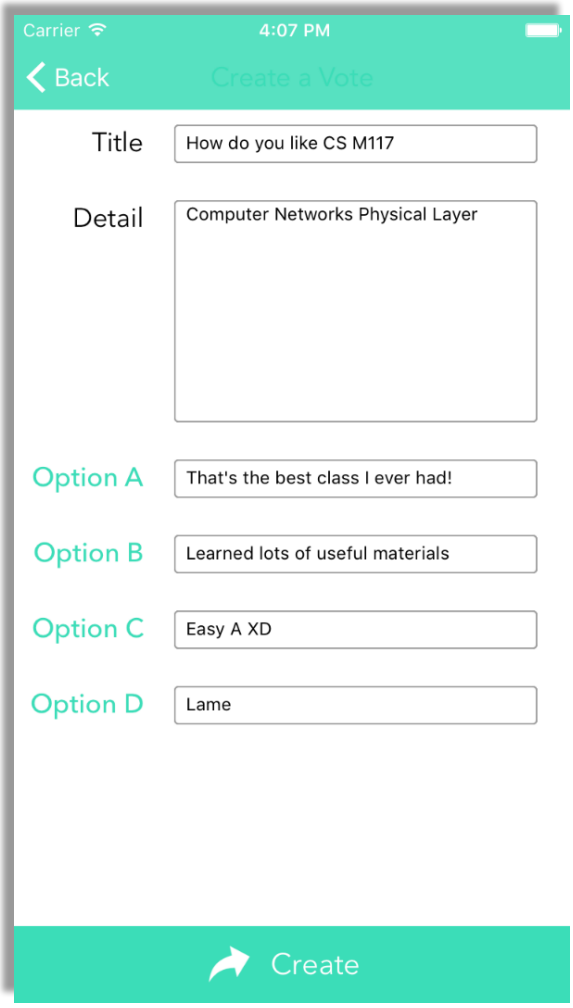
A local central device is represented by a CBCentralManager object, which is used to scan, discover, and connect to advertising peripheral objects; on the other hand, a peripheral device is represented by a CBPeripheralManager object. The peripheral device contains the data needed by the central and is able to transfer the data to central device through advertising packets.

Specifically, in this app, when the user clicks "create" button and go into the result page, a CBCentralManager object is created and the system begins to constantly searching for advertising peripheral objects; when the user clicks "join" button and chooses an answer, a peripheral object containing the answer is created and it begins to send out advertising packets. The CBCentralManager object calls the scanForPeripheralsWithServices method in CoreBluetooth to discover advertising Peripherals, calls the connectPeripheral method to connect to the peripheral, and calls didUpdateValueForCharacteristic to retrieve the data from the peripheral. If the data received is one of the strings "A", "B", "C" or "D", then we add one to the count and print out in the result page. Therefore, the creator of the vote is able to see the voting result. This process is accomplished via Bluetooth, so the transfer is fast, steady and can work when the Wi-Fi connection is weak. Furthermore, the vote is completely anonymous.

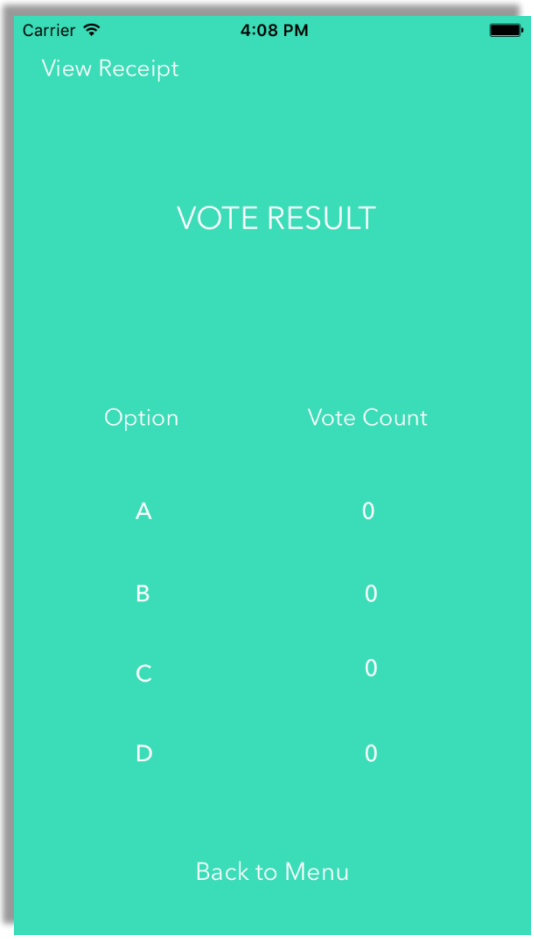


Demo:

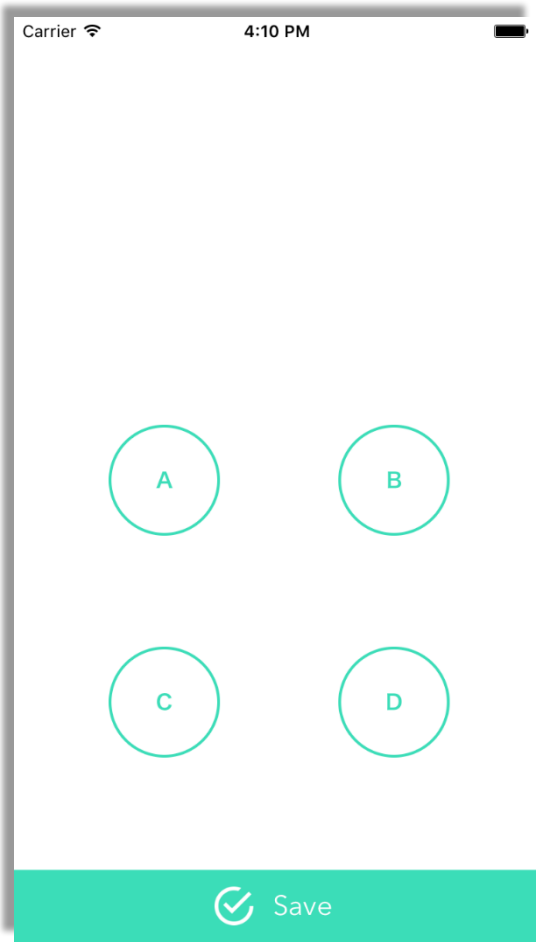
This is the main page of the app. Users can choose to either create or join a vote by clicking on the respective button.



In this page, the user can specify the title, detail and different options for the vote.



This is the vote result page. As mentioned before, a central object is created and it begins to scanning for advertising peripherals. Once it receives votes, the vote count will be updated.



This is the voting page. The user is able to select an answer to the vote and save the options. After the user saved the answer, a peripheral object containing the answer is created and it begins to advertise.

Difficulties

This project is the first time we use swift to develop iOS apps, and learning Swift language and MVC design principle are definitely challenges for us. However, the biggest challenge come from the Core Bluetooth implementation. We spent a lot of time learning the concept of Core Bluetooth, the functions associated with central and peripheral objects, and the implementation of Core Bluetooth framework into our own project.

Limitations & Future Work

Although the app can achieve most of our goals right now, it still has some limitations. Because it is based on Bluetooth, the range of this app is limited to close range voting (within 100 meters). When the distance is larger than the effective range, the creator of the vote may not be able to receive the votes. Also, there can only be eight peripheral devices connected to the central device at the same time, so this app may not be effective within big groups.

There are also several places we can work on in the future. First, we can add the central to peripheral communication in the app, so that the voter can see the options during voting. Second, we also need an effective method to distinguish between different votes, such as assign a different vote ID to each vote when it is created. We actually

implemented the vote ID part in the UI design, but didn't come out with a good way to put it in the controller. Third, we can add more interactions and features to this app, such as voting history, percentage chart for voting, etc.

Reference & Resources

Core Bluetooth Programming Guide, iOS Developer Library, Apple Inc.

Demo: https://www.youtube.com/watch?v=1OOPx5RPj-8

Github: https://github.com/kevinfan23/CSM117