PSoC Academy: How to Create a PSoC BLE iOS App
Lesson 6: Create the App

00:00:08 Hello. My name is Alan Hawse. Welcome back to Cypress Academy. In this lesson, I'm going to walk you through the creation of an iPhone app to communicate with your PSoC 4 BLE CapSense LED Board. In this lesson, I'm not going to try to teach you Swift. If you need to learn Swift, please go watch Paul Hegarty's excellent videos of Stanford CS193P on iTunes U. First a little background.

00:00:35 Apple provides a framework called Core Bluetooth that is completely documented in the Development Center. There are two key objects in that framework, the CBCentralManager, which is an object to represent the GAP Central that resides inside of the iPhone. The second key object is CBPeripheral, which represents a specific GAP peripheral. The way this works is you tell the CBCentralManager to listen for peripherals.

00:01:07 It then calls you back with a CBPeripheral object that contains the advertising packet data from any peripheral that it can hear. Then you can ask the CBCentralManager to create a connection to that peripheral. Once a connection is made, you can then do a service discovery. Once the service discovery is done, you will have an array of CB services that will be attached to your CBPeripheral object.

00:01:38 Then, you can ask any CB service to discover the characteristics that are part of that individual service. When that is complete, each of your CB services will have an array of CB characteristics that are attached to it. Once you have a CB characteristic, you can perform actions on that characteristic, like Reading and Writing.

00:02:06 For example, if you want to Write the LED with a 1 to turn it on, you use the CB characteristic Write function to change the value. This example project follows the iOS MVC programming paradigm. In this app, the M, or the Model, is an object that I created that represents the Bluetooth world, and the CapSense LED board. The V, or the View, is the storyboard with all of the buttons and labels that I placed on it. Then the C, or the Controller, or the View Controller, is the object that I created that takes actions from the view and ask the Model to perform those actions.

00:02:50 When those actions are complete inside of the Model, it sends the NSNotification which is an Apple iOS-provided functionality, to notify the View Controller that the Model has been updated. Do not be confused by the BLE notification and the NSNotification, as they're really fundamentally different things. A BLE notification occurs when something changes on the peripheral device, which then sends it out over Bluetooth, which then will trigger an NSNotification inside of the Model to tell the View Controller that something has changed.

00:03:30 For example, when the Bluetooth peripheral sends a new

CapSense value via the BLE Notification, the model will receive the changed data. It will then need to notify the View Controller of the change. The View Controller will listen for the change, and when it hears it, it can update the view or the display with the change to value. Before I show you the detailed code, I'm going to show you how the app works.

00:03:59 Inside of the app I've put printing, that through each of the steps will display on the screen. Okay. First I'll start up the CBCentralManager. The CBCentralManager, once it's started, will notify the View Controller that it's going, and it will print out that the Bluetooth is on, and will turn the next label blue. Then I'll tell the CBCentralManager that I'm interested in listening for devices. So I'll start searching for devices.

00:04:30 It immediately hears this device, and registers that fact by displaying it on the screen. Then I'll tell the CBCentralManager to actually make a connection to the device. So that tells the Model to tell the CBCentralManager to make the connection. Once that's occurred, it will say, Oh! You can discover the services on that device. So, I'll press Discover Services.

00:04:58 That will cause the peripheral to do a service discovery and tell me all of the services that are available on the device. I can then discover the specific CapSense and LED characteristic, which are part of the service. Once that’s done, I can actually start sending commands to those characteristics. The first command that I showed you with CySmart is the Write the LED characteristic. When I flip the switch, the red LED turns on, and off, and on, and off. Then I can turn on Notifications for CapSense.

00:05:33 When I turn on Notifications, I’ll start getting updates every time the CapSense is notified, and then if I put my finger on it, it’ll show me all of the different values, from 0 up to 100. Then when I don’t have my finger on it, I’ll get 65,535, which is also known as 0xFFFF in hexadecimal, and that shows that I can hear Notifications from the peripheral that we built.

00:06:02 I can then turn off the Notifications and press the Disconnect function, and the LED will start blinking again. Now I can walk you through the app and show you all of the parts for creating this app.