00:00:09 Welcome back. At this point we've completely configured our schematic. The schematic will define all of the firmware that will run inside of our project. Remember there will be two things going on. There will a CapSense slider. And there will be a red LED. The CapSense slider will be able to talk to the app. The red LED- you'll be able to turn on and off. So now we need to configure the BLE component, so that it's got those two characteristics in it. We'll start by double clicking the BLE. Now, that we've got the BLE customizer open I need to configure the profile.

00:00:42 This is going to be totally a custom profile, so I'll select custom. The device will act as a GATT server, and the GAP roll will be peripheral. GATT and GAP are two of the most important concepts in understanding this whole system. You can think of GAP as the way the two devices connect together. The iPhone will serve as the GAP central. And the BLE device will serve as the GAP peripheral.

00:01:14 The GAP central will talk to the GAP peripheral. The GAP peripheral will advertise that it's around. And the GAP central will hear that it's around and make the connection. And I'll show you how to do that in a few video. As I said, GAP is how devices connect. GATT is how devices trade information. This board, which you're going to configure, will run a GATT server.

00:01:40 Remember, a GATT server is a database that runs inside of your firmware that remembers information that the GATT client, which will be running on the iPhone will grab information out of or write information into. GATT is how this device and your iPhone trade information back and forth. Okay. At this point we have the generic configuration done for the BLE component. The next thing that we need to do is we need to customize the profiles.

00:02:10 So this is essentially setting up the GATT date base that the iPhone will be able to read out of. Creator handily gives you a custom service to start from. I'm going to start by deleting this custom service, and I'm going to create my own service. So I will go to the server, will need to add a custom service. I'm going to create a service that has to two characteristics in it. One characteristic will be for the LED. The other characteristic will be for the CapSense.

00:02:41 Those two characteristics are going to be grouped together into a custom service. That service is going to be called the ledcapsense service, and it will have it's own unique identifier. Those unique identifiers are called universally unique identifiers. That UUID we're going to tell the iPhone app so that it will be able to find devices that have that service. All right.

00:03:08 So first I'm going to change the name of the service. And I'm going to called it ledcapsense. This is going to give me the ability in my firmware to talk to that specific service. Then I'm going to take the first characteristic, which Creator calls Custom Characteristic, and

I’m going to change the name. And the name of this characteristic as I discussed earlier is going to be called led. That characteristic is going to represent the state of the LED.

00:03:40 This characteristic needs to be able to be read and written remotely. So I’ll enable the Write flag. And I’ll enable the Read flag. I’m not going to use the Custom Descriptor, so I’m going to delete it. The next thing I need is a CapSense characteristic, so I’ll add another custom characteristic. Just like before I’ll rename it, so that I’ll have sensible names of my API. So this characteristic for some reason – oh, yeah, because it’s CapSense -

00:04:12 I’m going to call it capsense. This characteristic is not going to be writable remotely. That doesn’t make any sense. You’ll only be able to read it. So I’m going to click Read. The other thing that’s interesting and that you’d like to be able to do is the other side would like to be able to be notified when things change. In order to be notified you have to click the Notify flag and that will create another special characteristic called the Client Characteristic Configuration Description – that’s a mouthful.

00:04:45 So I like to rename that. And I’m going to rename that thing to capsensecccd. Now I’m not, once again, going to use the custom descriptor, so I’m going to delete that. When you’re looking at this thing remotely from a GATT browser once of the useful things to be able to do is to find out a human readable form of the characteristic.

00:05:12 And so in order to do that you can add a special descriptor called the Characteristic User Description. And I'm going to add that to the LED. And I'm going to type in something sensible. This is LED uint8. So on the remote side you'll be able to see the name of this characteristic – to have human readable information. And I'm going to do the same thing for CapSense. I'm going to add the Characteristic User Description.

00:05:44 And I'm going to call this capsense uint16. So on the remote side I'll know that it gives me an unsigned int16 in human readable form. The next thing that needs to be configured - and I should have configured it originally, is I should have set the type of the capsense service. It needs to be a uint16. The default is uint8. Okay. This is good. So we've got the LED set up.

00:06:14 It's a uint8. It has a Characteristic User Description, which we'll be able to look at from the remote side called led uint8. We've got the capsense characteristic, which is a uint16, two bites in length. That's good. It has a CCCD – Client Characteristic Configuration Descriptor. Yes, a big long word. And then the last thing is has a Characteristic User Description called capsense uint16.

00:06:46 Now, we need to do more thing. Each of these things needs to have a specific 128 bit UUID. This is so the other side will be able to identity each of the characteristics and the service. So Creator gives it a default and just for grins and to make it easy I'm going to say that the service will be blah, blah, blah, blah zero.

00:07:14 The LED characteristic will be blah, blah, blah, blah one. And the CapSense will be blah, blah, blah, blah two. At this point we've set up the general information for the component. It's a custom profile. I just made the custom profile. It's a GATT server. In other words it's running the database. And it's a GAP peripheral, which means it can attach to the central.

00:07:45 I've set up the profile. I've created the custom service. It's got the two custom characteristics. That's all configured. The last thing I need to do is set up the GAP settings. Inside of the GAP settings there's a couple of things I want to do. First of all, I want to give the device a name. So I'm going to call it capled. Then I need to set up the advertising settings. For this setup I'm just going to advertise all the time. So I'm going to turn off the time out.

00:08:14 So this device you'll be able to connect to it. It'll advertise all the time when it's not connected. The last thing I'm going to do is inside of the advertising packet that's being sent out every 20 to 30 milliseconds – I want to put some information that will help the other side identify us. So the first thing I'm going to put in is the name that I just gave it will be accessible in each of the advertising packets. So it will show up on the remote side as something identifiable.

00:08:44 Then the last thing I wanted to do is I wanted to advertise the service that it has available. On the iPhone side, I'm only going to look for devices that are advertising this specific service that we just created. Okay. At this point the schematic is completely created. We've got the BLE set up. We've got the CapSense set up. We've got the blinking LED, which will indicate the state of the connection.

00:09:15 And we've got the red LED set up. Now, you need to configure your pins. So go to the DWR. The first pin you need to set will be P4[0]. This is the modulation capacitor for CapSense. The next thing you need to do is you need to connect the CapSense slider to the appropriate pins on the board. In this module that's P2[1] through 2[5].

00:09:40 So P2[1], P2[2], P2[3], P2[4], P2[5]. The next thing I need to do is I need to connect the blue LED to P3[7]. And I need to connect the red LED to P2[6]. All right. At this point the pins are set correctly,

00:10:13 the schematic is set up, and so to help myself here in a minute with the firmware I'm going go ahead and do a Generate Application. Now, that the application is generated. Go to the next lesson where we'll write the firmware.