PSoC Academy: How to Create a PSoC BLE Android App
Lesson 2: Configure the CapSense Component

Hello, I'm Alan Hawse. Welcome to Cypress Academy. In this video I'm going to show you how to create a project that you will use to connect to the Android app. So first we need to make PSoC 4 BLE firmware that we'll be able to connect to.

I'm going to build the project so that it starts with a blinking blue LED when the device is not connected. It's got a red LED that you'll be able to turn on and off from the app and a CapSense slider that you'll be able to read the values of.

First you need to start by creating a new project. File, new, project. This is going to be "PSoC 4100BLE / PSoC 4200BLE". And we'll call this "capsenseled" project. Okay, we'll start with an empty schematic. Go. Once we got the project created we'll start by adding the components that we'll need for this project in the schematic.

First we'll put in the BLE component from the component catalog into our project. Then we'll add the CapSense component, which we'll use for the CapSense slider object in our project. Then I'll need two pins. One pin for the red LED and one pin for the blue LED. I'll use digital output pins to drive the two different LEDs. I'll copy it onto the screen. I'll change its name to red. I'll

set its initial drive state to “high” because the LED is active low and we would like the LED to be off when the chip turns on. Then I’ll get another digital pin – a digital output pin. This one, I’ll change its name to blue.

I’ll need a PWM to drive the blue output, so I’ll grab the UDB based PWM from the catalog. Because the blue LED is active low I’m going to want to invert the output of the PWM. So I’ll grab a Not gate out of the library. I’ll put it on the screen, and then I’ll connect to the output of the PWM and then I’ll connect the blue LED to that. All right. Cool. So then I’m going to change the configuration of the PWM, so that it’s a one output PWM, and then I’ll need a clock to drive the PWM. So I’ll grab a clock component out of the library. We’ll set this clock to be driving the PWM, and I’ll configure it to one kilohertz. This will give us a blinking LED. How cool is that?

I would like to not leave the reset input of the PWM just hanging, so I’ll attach a logic low, because it’s an active high reset. So I’ll grab a logic low out of the component catalog and I’ll connect it to the reset pin.

I’m going to change the name of the PWM so its got a sensible name to interface to it from the firmware. This circuit is the PWM interface circuit. It drives the blue LED when you’re not connected, and when you are connected it will stop driving it and it will go off. The red LED is going to be the LED that

we will switch from our Android app. I'm going to disable the hardware connection as I'll only interface to this pin from the software. When you flip it inside of your Android app it will turn on, and when you flip the switch the other way it will turn off.

All right. So now I need to configure the CapSense component. I'm going to start by giving it the name capsense because I like to type less, and then I'm going to add a linear slider to the project. It will have five sensors because on the board there are five sensors, and you can see them neatly labeled on the silkscreen. OK, I'm going to change the name of BLE to make that just a hair simpler.

At this point we have our schematic completely configured. In the next lesson we're going to configure the BLE itself, which means we are going to set up the profile, set up the services, and set up the characteristics that you'll use to interface to with your Android application.