3-3A: BLE Simple Central – Control of 3-2a LED Service

Welcome back to Cypress Academy, PSoC 6 101. In the video 3-3a I showed you how to use PSoC 6 BLE to build a BLE Central that could control the project 3-2a BLE Periperial LED Dimmer.

In this video we are going to start building a complete BLE remote control for the robot arm. When the remote control is done you will be able to use the uart, capsense and the bosch motion sensor to control the PSoC BLE Robot.

I think that I will start this project by making a copy of the project from video 3-3 and carving it up. Lets start by editing the schematic. Open up the ble component customizer and delete the LED Service… then add in the motor service by right click dd service and picking the file motor service.

Now run generate application so that we get all of the ble updates into the middleware and generated source. You are going to get a bunch of errors… but don’t worry about them for now.

Now lets carve up the firmware into a more manageable structgure. Ill follow the template that I have in the MainCotroller.. meaning Ill have the ble stuff isolated into bleTask.h and .c … and Ill have a uart based controller in uartTask.h and uartTaskc.

Start by making a file bleTask.h

I know that we are going to want to send changes in motor positions… so let me make an enum with the motors. I going to need two functions, one to make relative changes and one to make absolute changes.. so let me create two function protypes.. .writeMotorRelative and Writemotorabsolute… when I call those function Ill send them a motor number and a percent.

The last thing that is needed in the bletask header file is a definition of the task.

Make a file bleTask.c by right clicking the source files folder and picking add new item… c file … then call it bleTask.c

Lets add includes for project.h, stdio.h freertos.h semaphore.h

Now I am going to go to main\_cm4.c and move from the top of the writeLED function through the end of the ble task into bleTask.c

I am going to modify the writeLed function… so Ill copy the function prototype from the bleTask.h and replace the writeLed function declaration.

Instead of brightness lets printout the information

If you recall from the previous video, in order to write, we need to know which handle to write to. If you remember from the BLE Peripheral, there are four possible handles that we are interested in. (show the gatt database from the ble) M1, M2, M1 Relative and M2 relative.

When we do the service discovery, our BLE stack will discover those characterstics and build a array of handles for those characterstics called cy\_ble\_customCServ [ which service] dot customServChar [ which index] dot customServCharHandle[0]

So in order to figure out the handle we need to do 4 if statements that lookup the correct handle based on M1 or M2 and Relative or absolute.

After that you assign the percent to the right variable… then write it using the Cy\_BLE\_GATTC\_WriteChractersticValue function.

Now I need to change the scanner to look for the motor service instead of the led service. ill change the comment, then the index and finally the printf message

When I wrote this code originally I was lazy and didn’t put in the BLE Semaphore. …. Lets fix that … first Ill add the semaphore to the top of the fi then Ill copy the interrupt service routine and bleTask from the previous BLE project, then and paste it into my bleTask. And finally, Ill chop out the stuff about event groups. Now we have a nice generic bleTask handler.

Now I need to add semaphore into FreeRTOS.h, and update the max syscall priority.

Almost done.

Make the file uartTask.h. This file will only have the pragma once and the definition of the uartTask.

Now one more cheat. I am going to just copy the uartTask from the BLE MainController into my project as almost exactly what we want.

First, add “bleTask.h” to the top. Then Ill add key commands for o … p …j … l… that will just call the writeMotorPosition function…. Which I will also add to the help printout.

OK program your development kit.

When I look on the uart, it starts searching for device… and right quick you can see that it find the robot… and both connection lights turn on.

Now when I press the o and o buttons you can see the arm move back and forth. Sweet!

You can post your comments and questions in our PSoC 6 community or as always you are welcome to email me at alan\_hawse@cypress.com or tweet me at @askioexpert with your comments, suggestions, criticisms and questions.