**INTRODUCTION**

Hello, and welcome to the video tutorial guide of the Exosite library on the Energia IDE for Texas Instruments LaunchPads. This tutorial will walkthrough and demonstrate the use-case of the Exosite library.

Two software packages will be discussed: the Exosite barebones-template solution and a real-world example in the form of the Exosite BBQ Companion Combination. Both of these use-cases have code available for the Energia and Code Composer Studio IDEs.

**BAREBONES - ENERGIA**

The barebones solution will require the following hardware: an MSP430F5529 LaunchPad and the CC3000 Wi-Fi transceiver BoosterPack.

The barebones solution will collect data from five analog-to-digital channels and five digital input/output channels for every Exosite write cycle. These values will be made available and displayed on the Exosite dashboard configured specifically for this application.

The barebones solution is able to read the state of virtual switches located on the Exosite dashboard.

After copying the folder titled “Exosite” into the msp430 libraries directory, run Energia, and navigate to File 🡪 Examples 🡪 Exosite 🡪 WifiReadWriteString

Modify the configuration variables to fit your Exosite and network credentials. The CIK data string can be obtained after you have created an account at <https://ti.exosite.com>, and have added a device. Instructions for creating an account and adding a device can be found in the readme file located in the folder directory “Exosite\examples\WifiReadWriteString”. Enter your ssid and password in the respective character array strings.

Once the user credentials have been updated, compile and run the code. You will want to open a COM port to see a detailed reporting of the device’s status.

For this video demonstration, I am using an Android OS based smart phone, running a free app called “PdaNet+” available at the Google Play store. This app creates a Wi-Fi hotspot as an Access Point for the CC3000 BoosterPack to connect to.

Once the CC3000 successfully connects to the access point, LED1 will illuminate on the LaunchPad. The barebones solution will now attempt to authenticate its connection to Exosite and perform the Exosite Read/Write operation. Depending on the result of the Exosite connection, the status message will update accordingly on the COM port. If successful, LED2 will illuminate on the LaunchPad. At this point, the barebones will loop the process of connecting to Exosite and perform the read/write of data operation. The iteration rate of updating data is dependent on the strength and speed of your Internet connection.

When looking at the Exosite Dashboard example created for this demo, the user can see the values of the ADC channels over time, the present state of the digital input/output channels, toggle two virtual switches to a high or low value, and view an image overlay representation of the barebones hardware.

**BBQ COMPANION – ENERGIA**

The BBQ Companion solution will require the following hardware: the MSP430F5529 LaunchPad, the CC3000 Wi-Fi transceiver BoosterPack, and the ADS1118 BoosterPack.

The BBQ Companion solution will write the temperature value captured by the ADS1118.

The BBQ Companion solution will read the temperature threshold, a countdown timer start value, and whether to display temperatures as Celsius or Fahrenheit as defined by the user through the Exosite Dashboard example created for this specific configuration.

After copying the folder titled “ExositeBBQCompanion” into the msp430 libraries directory, run Energia, and navigate to File 🡪 Examples 🡪 ExositeBBQCompanion 🡪 Exosite\_ADS1118\_WifiReadWriteString. The code for this project was written on top of the barebones solution’s template.

Just as in the barebones solution, modify the configuration variables to fit your Exosite and network credentials. Once the user credentials have been updated, compile and run the code. You will want to open a COM port to see a detailed reporting of the device’s status.

The LEDs will illuminate equivalently to the corresponding operation of the barebones’s solution.

The ADS1118 BoosterPack’s LCD updates the temperature and countdown timer values after each Exosite Read/Write cycle. The temperature threshold read from the Exosite Dashboard is used to compare the actual captured temperature to warn the user if exceeded.

**BAREBONES - CCS**

The Code Composer Studio barebones solution provides two running modes: the first mode uses network and Exosite credentials that are hard coded in the supporting files, while the second mode enables the option of configuring the network and Exosite credentials through the COM port, storing this information in Flash memory. If required, the second mode enables developers to generate a closed-release executable that prevents end-users from modifying the original source code. Navigate to inc 🡪 common.h to select the program’s running mode. By enabling #define EN\_COM\_CONFIG, the user provides the option to configure credentials through the COM port, while disabling this line requires the user to hardcode the credentials. The following credentials must be considered: Exosite CIK, Device Name, Model Name, SSID, Passphrase, Security Type, and MAC Address. To hardcode these credentials refer to the following files, and search the keyword “HARDCODED.”

* Application 🡪 demo.c = // HARDCODED - Model Name
* Inc 🡪 common.c = // HARDCODED - CIK, SSID, Passphrase, Security Type
* Inc 🡪 common\_adv.c = // HARDCODED - Device Name
* Exosite 🡪 Exosite\_hal.c = //HARDCODED - MAC Address

**BBQ COMPANION – CCS**

Similar to the Energia example of the BBQ Companion project, the Code Composer Studio example performs an Exosite Write to update the associated dashboard with the temperature value captured by the ADS1118 BoosterPack, and Exosite Reads the temperature threshold and count-up timer limit to update the device locally. The major difference is that the CCS’s timer example counts-up rather than counting-down, but will still expire and alert based on a user defined value. This demo also provides the running mode option to define Exosite and network credentials through hardcoded or COM port input.