# Chapter 2: Using the WICED SDK to Connect Inputs and Outputs

## Objective

IoT devices need to measure/control something. To do this, we need to learn how to read/write the pins of the chip.

Our shield will allow you to:

1. Light LEDs
2. Read mechanical buttons
3. Read CapSense buttons
4. Read Analog Coprocessor values (temperature, humidity, ambient light)
5. Display on an OLED

### Setting up a new WICED board support package

Platform = board support package

We have created and provided a platform for the baseboard/shield combination

Two critical files: platform.c and platform.h:

1. These define the resources such as LEDs and buttons and also initialize them for you
2. LEDs and buttons are: WICED\_SH\_LED0, WICED\_SH\_LED1, WICED\_SH\_MB0, WICED\_SH\_MB1

### Documentation

Documentation is in Components > Platform Functions

* Most functions require an init to work – will silently fail without (analogous to start in PSoC)

### Creating a new WICED Studio project

#### Directory Structure

The build process uses make which creates some requirements on project setup:

New project goes inside Apps folder (any number of subfolders may also be included). Must have:

1. Folder with the name of the project
2. Make file with the EXACT SAME NAME as the project with .mk at the end
3. C source file (usually same name as project with .c at the end)

Makefile contains:

1. Application name – MUST BE A UNIQUE STRING IN THE ENTIRE WORKSPACE and can have NO TRAILING SPACES
2. List of C source files
3. Maybe other stuff which we will cover later…

Point out that stuff that is bold and in all caps is critical.

#### Make Target

Make target format must be EXACTLY:

folder.folder.project-platform download run

Can leave out download/run if you want to just build without a kit attached.

Can right click, New… to get a new make target based on existing. Need to remove Copy of (including space) from the name

To build, double click the make target

Mention driver installation instructions if the device doesn’t show up

Explain when clean is needed (change to non C or h file such as certificate)

#### C file

C file:

1. Must have #include “wiced.h” at the top
2. Must call wiced\_init();before any other WICED calls

### Peripherals

1. GPIO functions and where documentation is located
2. PWM (mention init and start, mention need to de-init pin)
3. Debug Printing – mention that WPRINT\_APP\_INFO is on by default and that there are others that can be enabled
4. UART – STDIO\_UART is in platform already set of 115200 baud, Rx and Tx
5. I2C

## Exercises

Exercises – 90 minutes

Point out basic vs. advanced

Point out questions to answer in exercises

**The point of this chapter is to learn and practice fundamental skills and understand how to deal with issues**

## Chapter 2 Demos

Exercise 01 – Drag WW101\_2\_CYW943907AEVAL1F from windows explorer into Eclipse Platforms folder

Exercise 02 – Blink LED

Create folders: ww101/02/02\_blinkled

Create Files:

02\_blinkled.c:

/\* Blink LED1 on the shield with a frequency of 2 Hz \*/

**#include** "wiced.h"

**void** **application\_start**( )

{

wiced\_init(); /\* Initialize the WICED device \*/

/\* The LED is initialized in platform.c. If it

\* was not, you would need the following:

\* wiced\_gpio\_init(WICED\_SH\_LED1, OUTPUT\_PUSH\_PULL); \*/

**while** ( 1 )

{

/\* LED off \*/

wiced\_gpio\_output\_low( WICED\_LED1 );

wiced\_rtos\_delay\_milliseconds( 250 );

/\* LED on \*/

wiced\_gpio\_output\_high( WICED\_LED1 );

wiced\_rtos\_delay\_milliseconds( 250 );

}

}

02\_blinkled.mk

**NAME :**= App\_WW101\_02\_02\_blinkled

**$(NAME)\_SOURCES :**= 02\_blinkled.c

In code, show how to right click and open declaration on wiced\_gpio\_init

Make target: ww101.02.02\_blinkled-WW101\_2\_CYW943907AEVAL1F download run

Show some common errors – error in make target name, C file name, or folder path

Show how to copy/paste projects