Chapter 9: WICED Academy Shield

In order to add inputs and outputs that can be measured and controlled by the WICED device, we have created a shield board that can be connected to the WICED kit. Some of its features are:

- An Arduino interface to the WICED board
- A PSoC used as an analog front end (AFE) to several sensors, to read CapSense buttons, and to provide a programmable analog voltage
- LEDs and buttons
- Several different sensors such as temperature, humidity, ambient light, and a potentiometer
- A U8G OLED display

9.1 Features

9.1.1 PSoC4

The heart of the shield is a PSoC 4 (CY8C4A45AZI-483). This PSoC combines flexible Analog Front Ends, programmable Analog Filters, and high-resolution Analog-to-Digital converters along with an efficient-yet-powerful ARM® Cortex®-M0+ signal processing engine. The data from the PSoC can be accessed over I2C, UART, or SPI. In our case, I2C is used as the communication mechanism.

The PSoC 4 is capable of sensing voltage, current, resistance, inductance and capacitance. For our purposes, we use:

Resistance sensing for measuring temperature
Current sensing for measuring ambient light

Voltage sensing for measuring a potentiometer
Capacitance sensing for measuring humidity

9.1.2 LEDs

There are seven LEDs on the shield. Four are associated with the CapSense buttons (although they can be controlled independently via I2C if desired as you will see later). One is associated with the proximity loop on the board – it turns on whenever proximity is detected. The other two LEDs are controlled by the base board. All LEDs are active low, but the inversion is handled by the PSoC using the SmartIO block so that they appear to be active high to the WICED base board.

The two independently controlled LEDs are connected to the Arduino header as follows:

LED	Header Pin
LED1	D5
LED2	D11

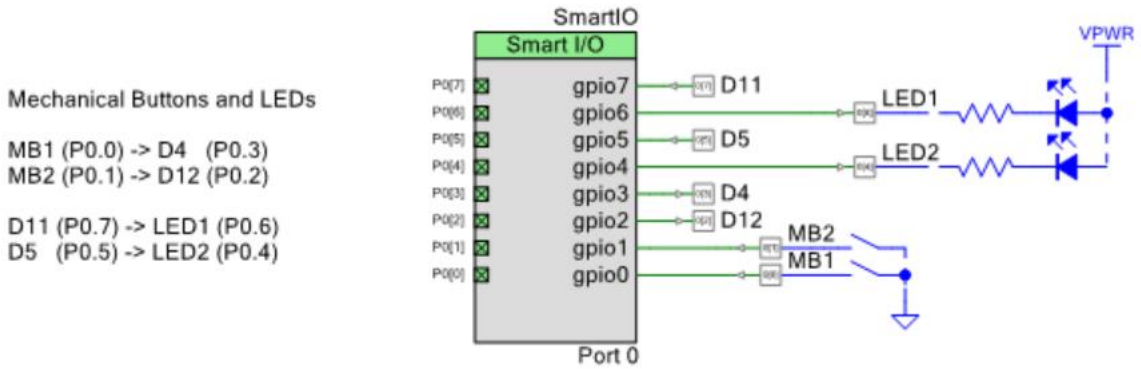
9.1.3 Mechanical Buttons

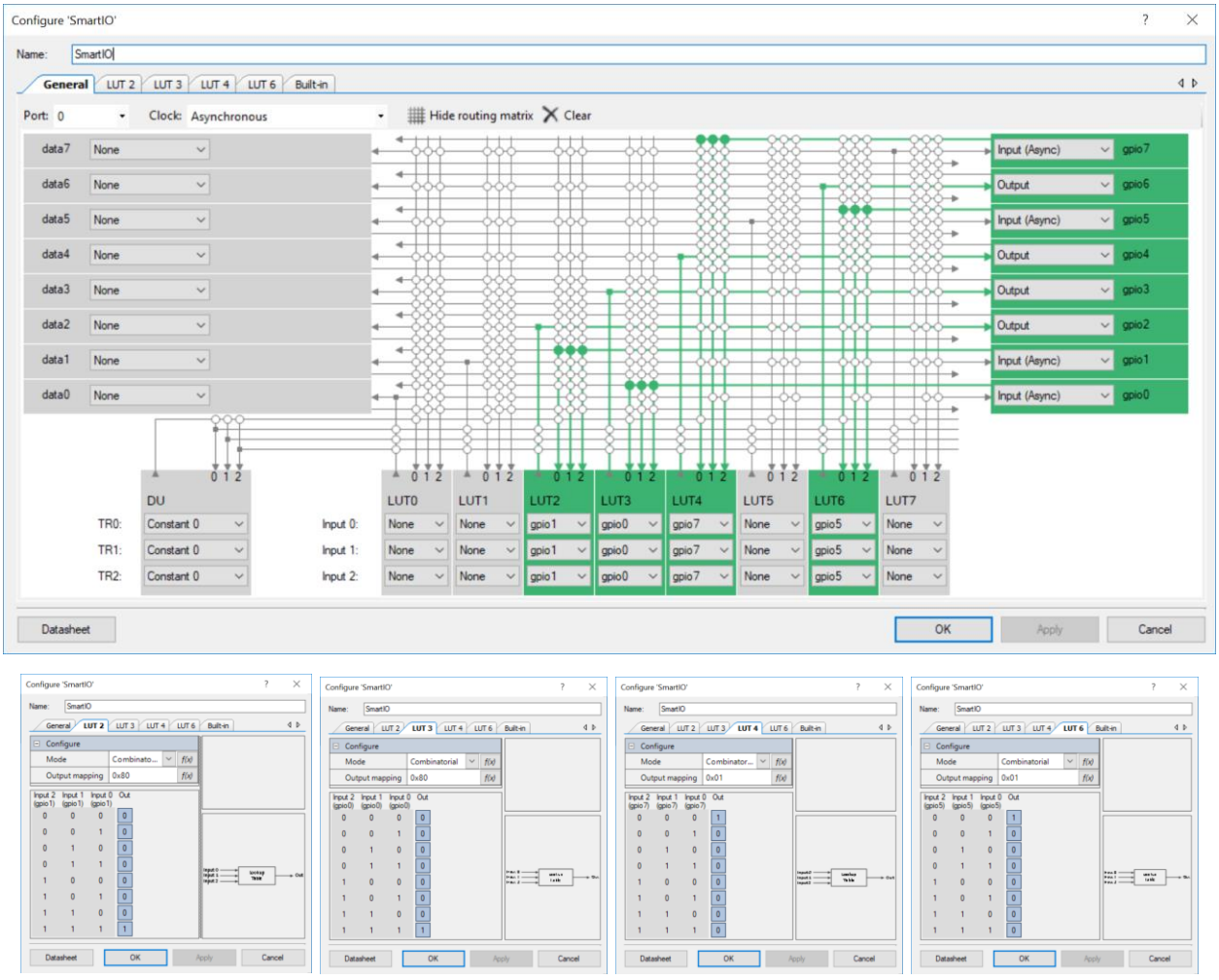
There are two mechanical buttons on the shield. These are active low and a pullup resistor is included for each of them on the shield. They are connected to the Arduino header as follows:

BUTTON	Header Pin
MB1	D4
MB2	D12

The state of the mechanical buttons can also be read via I2C as you will see later.

The two independent LEDs and the mechanical buttons are all controlled by the SmartIO block in PSoC. The schematic and the SmartIO configuration are shown below.





9.1.4 CapSense Buttons

There are four CapSense buttons. These buttons are scanned by the PSoC and their state is reported via an I2C register (see the I2C section later).

By default, each CapSense button will light an LED when it is touched. The LEDs can be "decoupled" from the CapSense buttons if desired by setting bit 0 in the I2C LED Control register. Once that is done, the LED Value register can be used to control the LEDs independently from the CapSense buttons.

9.1.5 Proximity

There is a proximity sensor that runs around the outer edge of the board. The proximity sensor state is reported over I2C (see the I2C section later). There is an LED dedicated to the proximity sensor – it turns on whenever proximity is detected.

9.1.6 Thermistor

The temperature is calculated by measuring voltage across a thermistor. The actual temperature calculation is handled by a PSoC component called “Thermistor” which greatly simplifies the coding required. The schematic and firmware are based on code example CE211321. The temperature value can be read over the I2C interface (see I2C section below for details) and is reported in degrees Celsius.

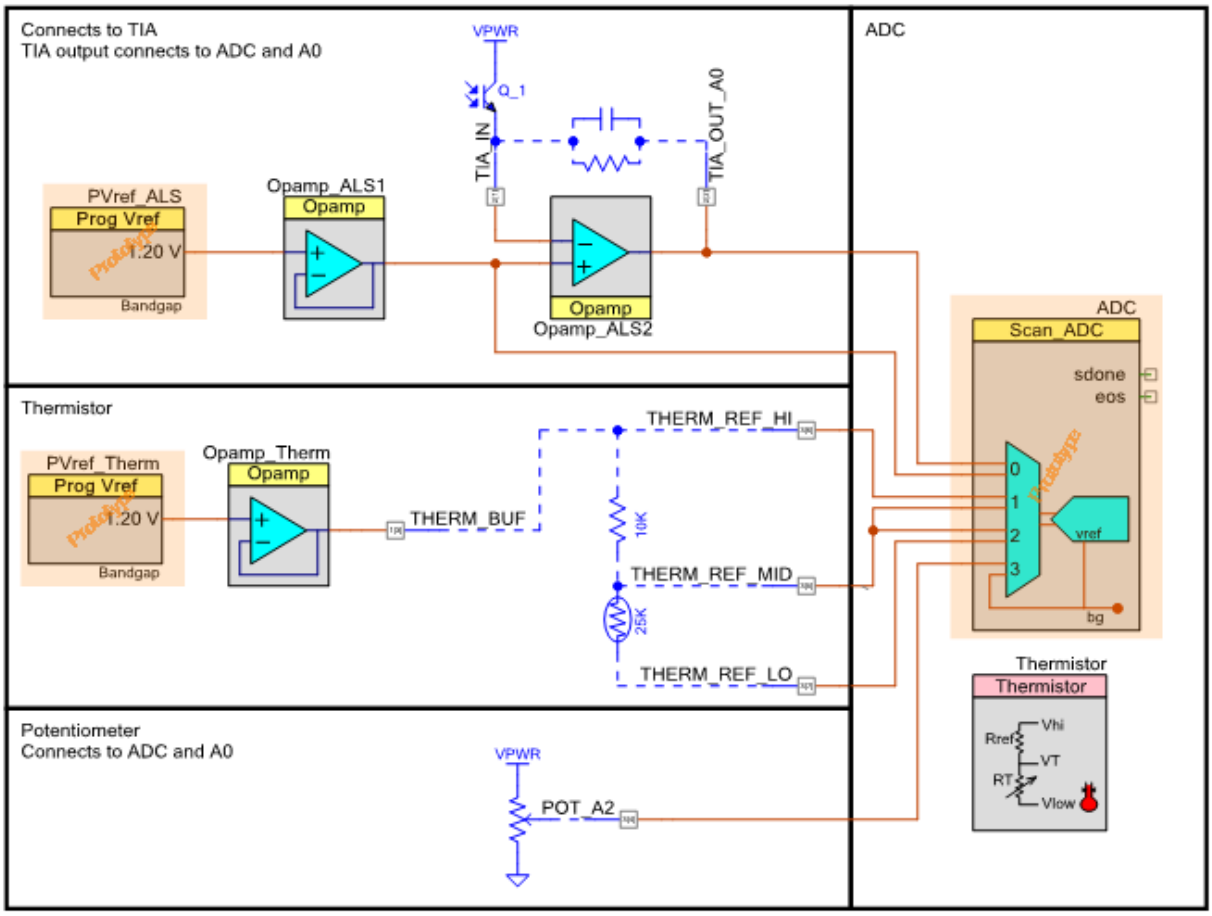
9.1.7 Ambient Light Sensor

The ambient light is calculated by measuring current through a photo-transistor. The schematic and firmware are based on code example CE211252. The light value can be read over the I2C interface (see I2C section below for details) and is reported in Lux. In addition, the raw value at the output of the TIA can be measured at Arduino pin A0.

9.1.8 Potentiometer

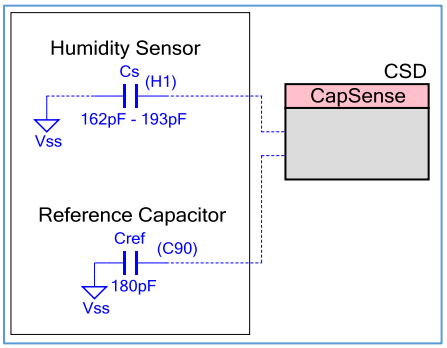
The voltage of the potentiometer is measured and can be read over the I2C interface (see I2C section below for details) and is reported in Volts. The ADC range is limited to 0 – 2.4V. In addition, the raw POT voltage is available at Arduino pin A2.

The schematic for the Thermistor, Ambient Light Sensor, and Potentiometer is shown below.



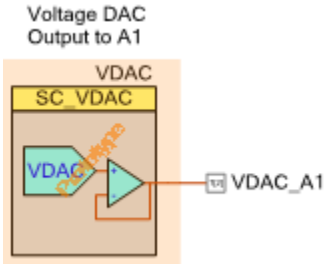
9.1.9 Humidity

The humidity is calculated by measuring capacitance of a humidity sensor using the CapSense block. The schematic and firmware are based on code example CE211322. The humidity value can be read over the I2C interface (see I2C section below for details) and is reported as a percentage.



9.1.10 DAC Output

A voltage DAC output is available on Arduino pin A1. The DAC voltage can be set in the range from 0V - 2.4V over the I2C interface (see I2C section below for details).



9.1.11 PSoC I2C Slave

The I2C interface is an EZI2C slave. That is, the first byte of a write into the slave is an offset to the set of I2C registers. The remaining bytes (if any) are the data to be written starting at the offset. For I2C reads from the slave, the offset is whatever was set in the previous write.

The I2C slave is assigned to 7-bit address 0x42 and is configured for a speed of 100 kHz. It is connected to Arduino pins D14 (SDA) and D15 (SCL).

The I2C register map is as follows:

Offset	Description	Format	Details
00	DAC Voltage	4 Byte Float	Desired DAC voltage in Volts
04	LED Value	1 Byte	CapSense LED values if LED Control bit 1 is 1. Mapping is: Bit 0: CSLED0 Bit 1: CSLED1 Bit2: CSLED2 Bit3: CSLED3
05	LED Control	1 Byte	Bit 0 sets how the 4 CapSense LEDs are controlled: 0 = CapSense Control 1 = Base Board Control via the LED Value Register
06	Button State	1 Byte	State of CapSense, Proximity, and Mechanical Buttons: Bit 0: CapSense B0 Bit 1: CapSense B1 Bit 2: CapSense B2 Bit 3: CapSense B3 Bit 4: Mechanical Button 1 Bit 5: Mechanical Button 2 Bit 6: Proximity

Offset	Description	Format	Details
07	Temperature	4 Byte Float	Temperature reported in °C
0B	Humidity	4 Byte Float	Humidity reported in %
0F	Ambient Light	4 Byte Float	Ambient light reported in Lux
13	Potentiometer	4 Byte Float	Potentiometer reported in Volts

9.1.12 U8G OLED Display

The shield contains a U8G OLED display with an I2C interface. The OLED is an I2C Slave with an address of 0x3C which can be controlled from the WICED baseboard using the I2C interface connected to Arduino pins D14 and D15.

9.1.13 Arduino pins

The Arduino pin connections between the shield and the base board are shown below.

Arduino	WICED Pins	Shield Function
A0	N/A*	Ambient Light TIA Output
A1	N/A*	DAC Voltage
A2	N/A*	Potentiometer
D4	WICED_BUTTON1 WICED_GPIO_10	Mechanical Button MB1
D5	WICED_LED1 WICED_GPIO_12 WICED_PWM_3	LED1
D11	WICED_LED2 WICED_GPIO_6 WICED_PWM_1	LED2
D12	WICED_BUTTON2 WICED_GPIO_8	Mechanical Button MB2
D18	WICED_GPIO_48	I2C_SDA
D18	WICED_GPIO_49	I2C_SCL

* The analog pins do not connect directly to the WICED device for the board we are using. Instead, they connect to a separate ADC on the base board. The ADC can be read using I2C.

9.2 Appendix A: Programming the PSoC

The PSoC on the shield is pre-programmed with the firmware that contains the functionality described above. If, for some reason, you want to modify that functionality or you need to re-program the firmware into the kit, please refer to the following sections.

9.2.1 PSoC Creator Project

The project workspace is included with the class files at:

WW-101 Files\ww101-shield\firmware\Shield.cywrk

To open the workspace in PSoC Creator, double-click on the workspace (cywrk) file. Note, you must have PSoC Creator 4.1 or later installed to open the project.

All of the projects should be built using the "Release" build option in PSoC Creator.

The workspace contains 4 projects:

1. Shield: The main shield project as described in the main body of this document.
2. Bootloader: A bootloader which allows the shield firmware to be bootloaded. It is included in the Shield project. See the next section for bootloading instructions.
3. TestProgram4M: A test project for a CY8CKIT-044 kit which can be used to test the functionality of the shield. See Appendix B for details of the test program.

9.2.2 Bootloading

The project contains an I2C bootloader. You can bootload the project by connecting the shield to any PSoC Pioneer kit whose Kitprog I2C pins connect to Arduino header pins D14 and D15. For example, the PSoC 4M Pioneer kit (CY8CKIT-044) will work for this purpose.

To put the PSoC into bootloader mode, hold down MB1 and MB2 simultaneously and turn the POT across its full range until LED1 and LED2 flash in an alternating pattern. Then, you can use the PSoC Bootloader Host to load the new firmware. The I2C address is 0x42.

Note: If you are using the CY8CKIT-044 with the test program that is included in the workspace, you must also put that kit into bypass mode to bootload the shield. To do that, hold down SW2 (about 5 seconds) until the red LED in the tri-color LED begins to flash. This disables the LCD update from the CY8CKIT-044 which frees up the I2C bus so that the KitProg can use it for bootloading.

The Bootloadable firmware file can be found in the workspace at:

WW101 Files\ww101-shield\firmware\Shield.cydsn\CortexM0p\ARM_GCC_541\Release\Shield.cyacd

9.3 Appendix B: CY8CKIT-044 Shield Test Program

As mentioned in Appendix A, the shield project workspace contains a project called "TestProgram4M" which can be used along with CY8CKIT-044 Pioneer kit to test the functionality of the shield board.

9.3.1 Test Procedure

At power-up, the LCD on the shield will display test information for each of the shield's features. Each feature will say Pass or Fail next to them depending on the test status. Some of the features are self-tested while others require user input. The test procedure is outlined below. Once all tests pass, a green LED on the base board will turn on. The DAC, humidity, and temperature are self-tested so they should say Pass right away if the shield is operating properly. For the button test, touch each CapSense button and press each mechanical button. Once you do that, then the Buttons test should say Pass. Also, note that an LED should turn on for each button when they are being pressed.

Feature	Method	Test Procedure
Buttons	Manual	Press each CapSense button and Mechanical button. An LED must turn on next to each button when it is pressed.
DAC	Automatic	The voltage is swept by the baseboard and the resulting voltage is measured on Arduino pin A1.
Potentiometer	Manual	Sweep the pot across its range. The voltage is measured on Arduino A2.
Ambient Light Sensor	Manual	Cover the light sensor and then shine a light on it.
Humidity	Automatic	The humidity reading is examined for a valid result.
Temperature	Automatic	The temperature reading is examined for a valid result.

9.3.2 Alternate Screens

In addition to the main test screen, there are additional screens with more detailed information. Press user button SW2 on the base board to toggle between the following screens:

1. Main test screen: This is the main test results page as described above.
2. Analog Values Screen: Shows readings for temperature, humidity, illumination, potentiometer, Arduino pin A0 and Arduino pin A1. Note that pin A1 is the DAC output which is continually swept by the test program in 100mV increments.
3. Base ADC Screen: Shows raw ADC readings in mV from A0, A1, and A2.
4. Buttons Screen: Shows real-time values for the CapSense buttons, proximity sensor, and mechanical buttons. The mapping is:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Unused	Proximity	MB2	MB1	CS3	CS2	CS1	CS0

The buttons screen also shows the LED value register and the LED control register, but these are not controlled by the test program so they should always read 0.