7. Analog to Digital Converter (ADC)

The WNC processor module on the 2nd Generation IoT Starter Kit (SK2) from AT&T includes and Analog-to-Digital Converter (ADC) peripheral which is handy for translating analog voltage signals coming from sensors into numbers which can be processed and manipulated by the CPU.

Your SK2 uses this peripheral in one of two ways, based on the position of a jumper on the board:

1. It is connected to the ALS-PT19 ambient light sensor. This allows your programs to observe the amount of light shining on the SK2.
2. Additionally, the ADC is routed to the 60-pin Expansion connector.

This chapter introduces the ADC peripheral – how it works followed by examples exercising the ambient light sensor.

Note: Shining too bright of a light on the ambient light sensor could cause it to output a voltage greater than allowed, which could damage the WNC M18Q2FG module. Please refer to the comments in the ADC section of the *SK2 Hardware User's Guide* and the AT&T IoT Starter Kit *2nd Gen Python Peripheral API Guide*.

Topics

7.1. Analog to Digital Conversion (ADC)

Analog to Digital converters (ADC) convert analog signals, such as light or temperature, into digital (i.e. numeric) values.

Analog signals are frequently generated by sensors, such as the ALS-PT19 ambient light sensor found on the SK2. This sensor outputs a voltage signal – that is, it varies the voltage value on its output pin – based upon how bright the light that is shining upon it.

While it's convenient that this sensor coverts the analog light value to an analog voltage value, it is still not in a form that can be used by a digital processor. For this reason, many processors, such as the WNC M18Q2FG module on the SK2, include an ADC for translating voltages into numbers.

With sensor values converted to a digital number, digital processors can run algorithms on them and/or use them for control inputs to their systems.

7.1.1. Details about the SK2 ADC

From the SK2 hardware reference manual we see that our WNC processor includes a single ADC input pin. The ADC specifications include:

- A 16-bit converter, sampled at 2.4MHz for an ENOB of 15 bits
- The ADC's input voltage range is 0.1V to 1.7V

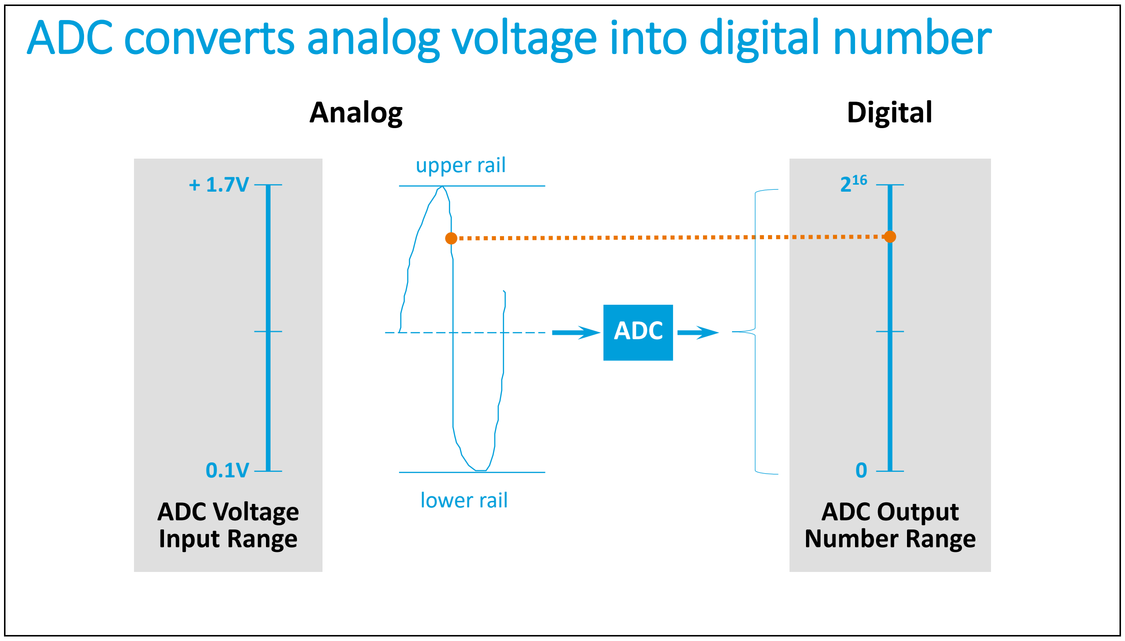
The ADC's 16-bit resolution provides 2^{16} discrete values (i.e. 64K values) that can be assigned across the ADC input voltage range. Error that builds up in the converter reduces the resolution of the converter to an Effective Number Of Bits (ENOB) of 2^{15}. (This is a rather small error compared to many other processor's on-chip ADCs.)

Note: *Resolution* and *Error* are discussed in the next section.

7.1.2. How an ADC Works

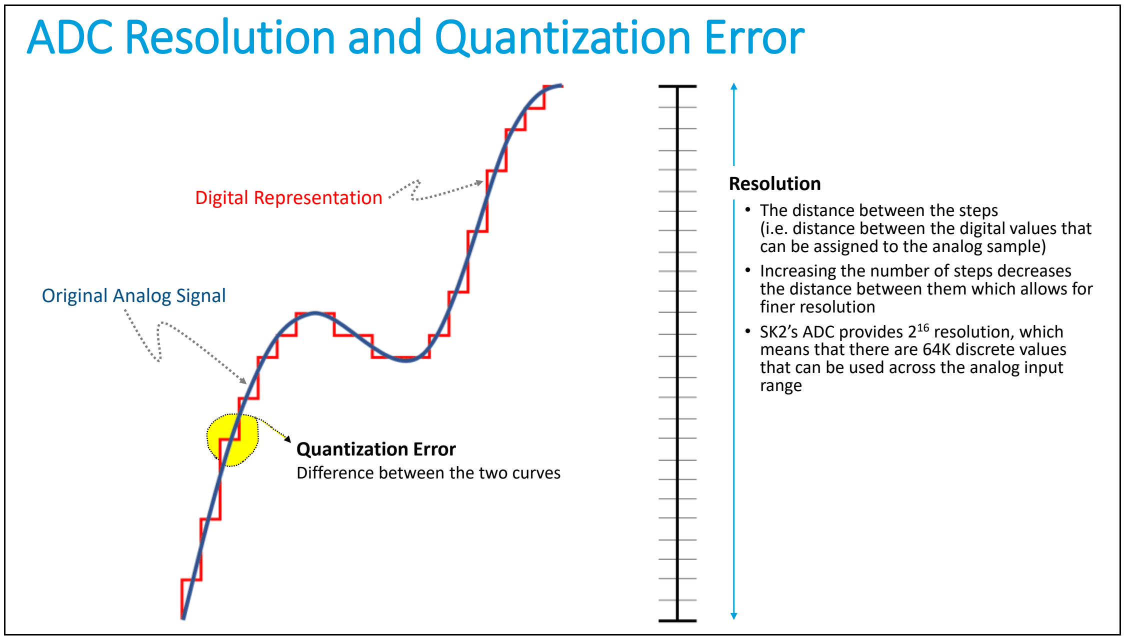
As described earlier in the chapter, ADCs work by converting voltages into numbers. Voltage signals can be found in many places in our world. For example, the batteries used in everyday products provide a voltage output. But they can also be found in the output of real-world sensors, such as the ambient light sensor found on the SK2.

If a voltage falls within the specification range of our ADC's input pin, we'll be able to convert it to a number. If the voltage source you want to sample is too large or small to work with the SK2's ADC, your hardware design will need to scale the voltage appropriately.



As the voltage changes up or down, the ADC assigns a larger or smaller number in order to represent the analog signal as accurately as possible. Though, that doesn't mean that it's a perfect process. Even with the large 216 range of numbers that can be assigned by our ADC, analog signals cannot be perfectly represented by discrete numbers.

From the graphic below, we see the error between the original analog signal and its digital representation. This is called *quantization error* since the absolute analog value must be assigned to the finite digital quanta.



ADC Resolution and Quantization Error

Resolution

- The distance between the steps (i.e. distance between the digital values that can be assigned to the analog sample)
- Increasing the number of steps decreases the distance between them which allows for finer resolution
- SK2's ADC provides 2^{16} resolution, which means that there are 64K discrete values that can be used across the analog input range

Thankfully, the high resolution of the SK2 ADC and its high internal conversion clock minimizes the amount of error.

7.2. ADC Application Programming Interface (API)

Both the C/C++ and Python software development kits include a software interface that allows software programs to read the digital value from the analog input pin.

The ADC API includes three simple methods:

Description	C/C++	Python
ADC Initialization	`adc_init()`	`iot_hw.adc()`
Read analog input	`adc_read()`	`.read()`
Close the ADC	`adc_deinit()`	`.close()`

The functions are examined further through the examples later in the chapter.

You can also find more information about the API in the Peripheral API guides. Both C/C++ and Python API guides can be found under the “Documents” section of the AT&T IoT Starter Kit product page.

7.3. ADC Implementations on the SK2

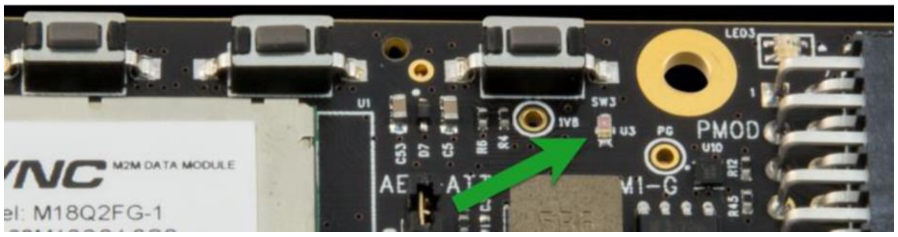
The SK2 board connects the ADC analog input pin from the WNC M18Q2FG processor module to two different resources on the SK2.

- ALS-PT19 Ambient Light Sensor soldered to the top of the SK2
- 60-pin Expansion Connector found on the bottom of the SK2

A jumper connector (JP2) allows you to specify which resource you want your program to access. The next two sections briefly describe each of these two resources, but we recommend that you refer to the SK2 hardware guide for more detail.

7.3.1. ALS-PT19 Ambient Light Sensor Sensor

The ALS-PT19 is a low cost ambient light sensor, consisting of phototransistor in miniature surface mount device.



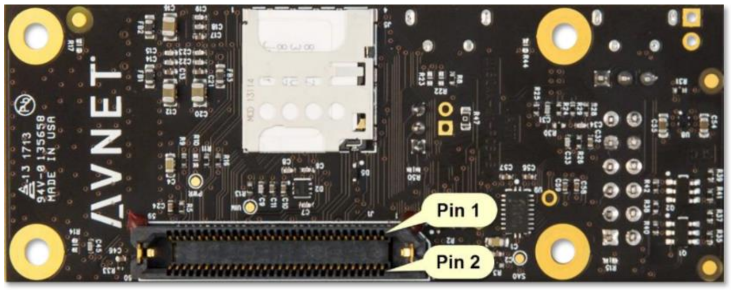
This simple sensor makes it convenient to control your program based on the amount of light hitting your SK2.

We use these types of devices frequently in our everyday lives. In order to save power, our phone screens brighten or darken based on the light hitting them. Similarly, automobile dashboard lights turn on automatically as the evening darkens. In our case, the light sensor provides a convenient way to experiment with the ADC input.

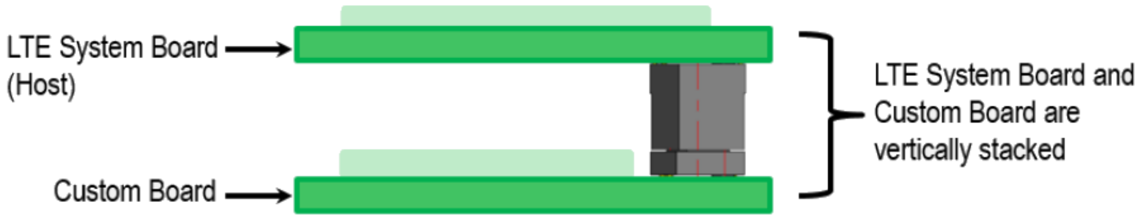
Note: Shining too bright of a light on the ambient light sensor could cause it to output a voltage greater than allowed, which could damage the WNC M18Q2FG module. Please refer to the comments in the ADC section of the *SK2 Hardware User's Guide* and the AT&T IoT Starter Kit *2nd Gen Python Peripheral API Guide*.

7.3.2. Expansion Connector

There are a great many additional sensors that provide analog signals. It's likely your system will need one or more of them. Towards that end, the analog input pin was also routed to the SK2 60-pin Expansion Connector. This allows you to build and connect your own hardware module to the SK2; or, you can purchase the LTE IoT Breakout Carrier which allows you to plug in one or two of the many MikroE Click peripheral modules. (There's over 300 to choose from.)



Expansion Connector Pin Numbering



7.4. Using the Ambient Light Sensor

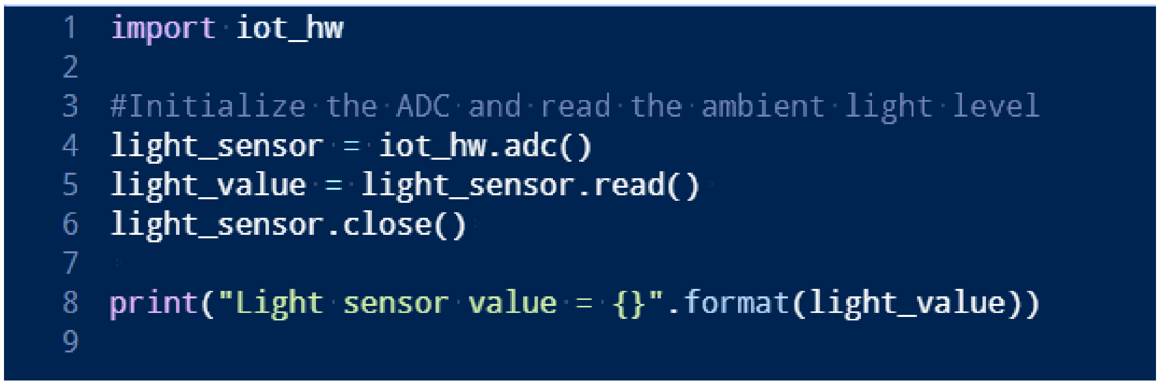
7.4.1. Code Examples

Before executing either of the following examples, make sure the JP2 jumper is set to pins 1 & 2 as this connects the WNC module's ADC pin to the SK2's ALP-PT19 ambient light sensor.

7.4.1.1. Python Example: Getting the Ambient Light Value

Using the Python SDK peripheral API, it's easy to read a value from the ADC input. As described in Section 7.2, you first need to initialize the ADC sensor with the *iot_hw.adc()* function call. With the handle returned by the initialization function, you can then read one (or more) samples from the analog to digital converter.

Listing 7.1 - adc_example.py



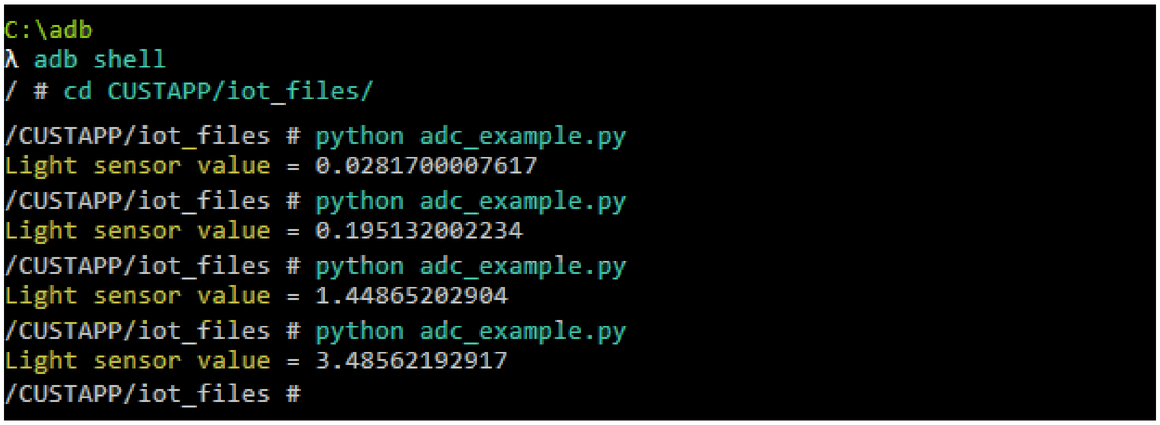
```python
import iot_hw

#Initialize the ADC and read the ambient light level
light_sensor = iot_hw.adc()
light_value = light_sensor.read()
light_sensor.close()

print("Light sensor value = {}".format(light_value))

```

Running this program four times, increasing the light hitting the sensor each time, our results were:

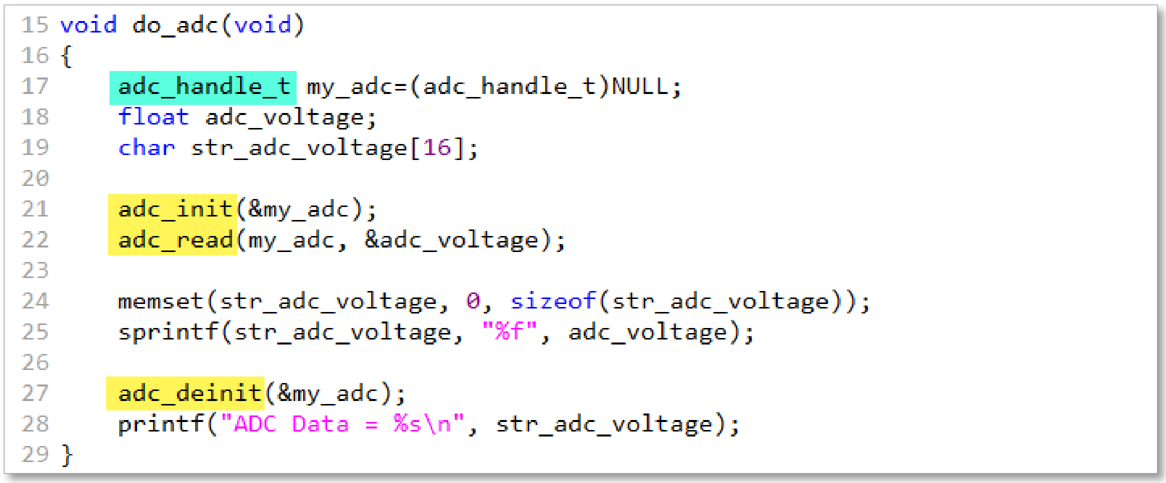


```
C:\adb
λ adb shell
/ # cd CUSTAPP/iot_files/
/CUSTAPP/iot_files # python adc_example.py
Light sensor value = 0.0281700007617
/CUSTAPP/iot_files # python adc_example.py
Light sensor value = 0.195132002234
/CUSTAPP/iot_files # python adc_example.py
Light sensor value = 1.44865202904
/CUSTAPP/iot_files # python adc_example.py
Light sensor value = 3.48562192917
/CUSTAPP/iot_files #
```

This simple program illustrates using the ADC to read the on-board ambient light sensor. You'll find a few additional programs that use the ADC in the Python examples provided with the Python SDK.

7.4.1.2. C/C++ Example: Getting the Ambient Light Value

Like Python, the C code for reading the ambient light sensor with ADC module API is quite simple. The following example was taken from the IoT Monitor (SK2 out-of-box example) program – specifically the *do_adc2m2x()* function from the `m2x.c` source file.



```c
void do_adc(void)
{
    adc_handle_t my_adc=(adc_handle_t)NULL;
    float adc_voltage;
    char str_adc_voltage[16];

    adc_init(&my_adc);
    adc_read(my_adc, &adc_voltage);

    memset(str_adc_voltage, 0, sizeof(str_adc_voltage));
    sprintf(str_adc_voltage, "%f", adc_voltage);

    adc_deinit(&my_adc);
    printf("ADC Data = %s\n", str_adc_voltage);
}
```

Page left intentionally blank