

Full-Featured Pedometer Design Realized with 3-Axis Digital Accelerometer

By Neil Zhao

Introduction

Pedometers, now popular as an everyday exercise progress monitor and motivator, can encourage individuals to compete with themselves in getting fit and losing weight. Early designs used a weighted mechanical switch to detect steps, plus a simple counter. When these devices are shaken, one can hear a metal ball sliding back and forth, or a pendulum striking stops as it swings.

Today, advanced pedometers rely on *microelectromechanical systems* (MEMS) inertial sensors and sophisticated software to detect true steps with high probability; MEMS inertial sensors permit more accurate detection of steps and fewer false positives. Taking advantage of the low cost and minimal space- and power requirements of MEMS inertial sensors, pedometers are being integrated into an increasing number of portable consumer electronic devices—such as music players and mobile phones. The small, thin, low-power [ADXL335](#), [ADXL345](#), and [ADXL346](#) 3-axis accelerometers from Analog Devices are very suitable for such applications.

This article, based on a study of the characteristics of each step a person takes, describes a reference design using the 3-axis ADXL345 accelerometer in a full-featured pedometer that can recognize and count *steps*, as well as measure *distance*, *speed*, and—to an extent—*calories* burned.

The ADXL345's proprietary (patent pending), on-chip, 32-level first-in, first-out (FIFO) buffer can store data and operate on it for pedometer applications to minimize host processor intervention, thus saving system power—a big concern for portable devices. Its 13-bit resolution (4 mg/LSB) allows pedometers to even measure low-speed walking (where each step represents about 55 mg of acceleration change) with reasonable accuracy.

Understanding the Model

From the characteristics that can be used to analyze running or walking, we choose *acceleration* as the relevant parameter. The three components of motion for an individual (and their related axes) are forward (*roll*), vertical (*yaw*), and side (*pitch*), as shown in Figure 1. The ADXL345 senses acceleration along its three axes: *x*, *y*, and *z*. The pedometer will be in an unknown orientation,

so the measurement accuracy should not depend critically on the relationship between the motion axes and the accelerometer's measurement axes.

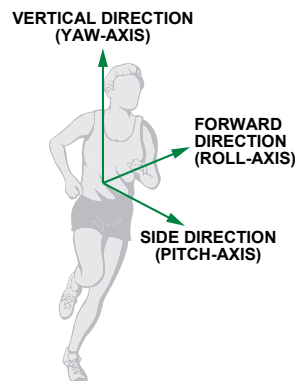


Figure 1. Definition of each axis.

Let's think about the nature of walking. Figure 2 depicts a single step, defined as a unit cycle of walking behavior, showing the relationship between each stage of the walking cycle and the change in vertical and forward acceleration.

Figure 3 shows a typical pattern of *x*-, *y*-, and *z*-measurements corresponding to vertical, forward, and side acceleration of a running person. At least one axis will have relatively large periodic acceleration changes, no matter how the pedometer is worn, so peak detection and a dynamic threshold-decision algorithm for acceleration on all three axes are essential for detecting a unit cycle of walking or running.

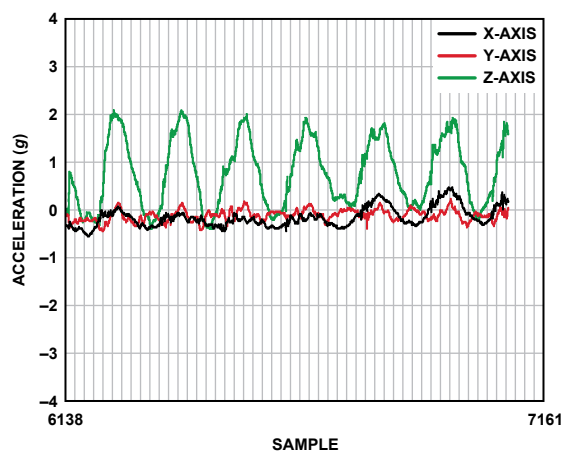


Figure 3. Typical pattern of *x*-, *y*-, and *z* accelerations measured on a running individual.

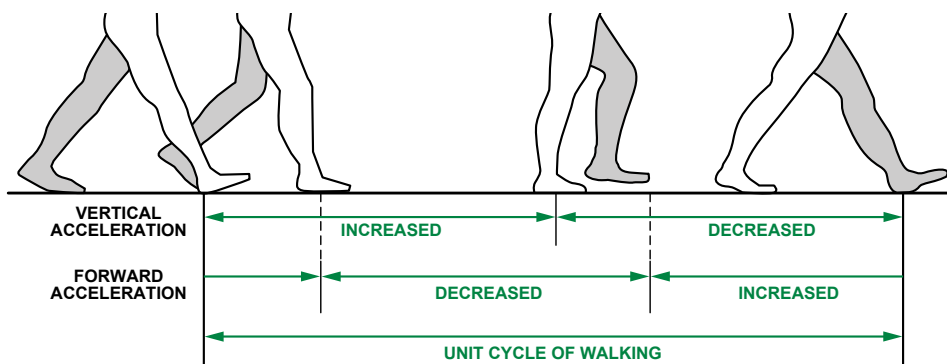


Figure 2. Walking stages and acceleration pattern.

Algorithm

Steps Parameter

Digital Filter: First, a digital filter is needed to smooth the signals shown in Figure 3. Four registers and a summing unit can be used, as shown in Figure 4. Of course, more registers could be used to make the acceleration data smoother, but the response time would be slower.

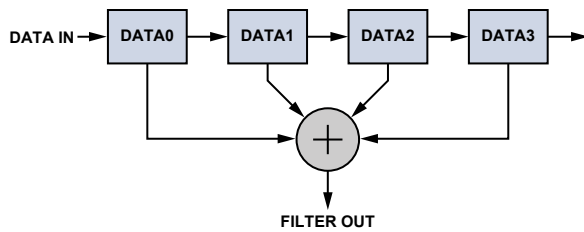


Figure 4. Digital filter.

Figure 5 demonstrates the filtered data from the most active axis of a pedometer worn by a walking person. The peak-to-peak value would be higher for a runner.

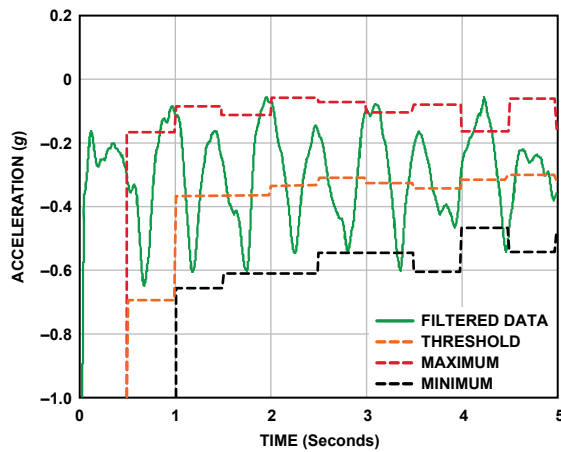


Figure 5. Filtered data of the most active axis.

Dynamic Threshold and Dynamic Precision: The system continuously updates the maximum and minimum values of the 3-axis acceleration every 50 samples. The average value, $(Max + Min)/2$, is called the *dynamic threshold level*. For the following 50 samples,

this threshold level is used to decide whether steps have been taken. As it is updated every 50 samples, the threshold is *dynamic*. This choice is adaptive and fast enough. In addition to dynamic threshold, dynamic precision is also used for further filtering as shown in Figure 6.

A linear-shift-register and the dynamic threshold are used to decide whether an effective step has been taken. The linear-shift-register contains two registers, a *sample_new* register and a *sample_old* register. The data in these are called *sample_new* and *sample_old*, respectively. When a new data sample comes, *sample_new* is shifted to the *sample_old* register unconditionally. However, whether the *sample_result* will be shifted into the *sample_new* register depends on a condition: If the changes in acceleration are greater than a predefined precision, the newest sample result, *sample_result*, is shifted to the *sample_new* register; otherwise the *sample_new* register will remain unchanged. The shift register group can thus remove the high-frequency noise and make the decision more precise.

A step is defined as happening if there is a negative slope of the acceleration plot ($sample_new < sample_old$) when the acceleration curve crosses below the dynamic threshold.

Peak Detection: The step counter calculates the steps from the *x*-axis, *y*-axis, or *z*-axis, depending on which axis's acceleration change is the largest one. If the changes in acceleration are too small, the step counter will discard them.

The step counter can work well by using this algorithm, but sometimes it seems too sensitive. When the pedometer vibrates very rapidly or very slowly from a cause other than walking or running, the step counter will also take it as a step. Such invalid vibrations must be discarded in order to find the true rhythmic steps. *Time window* and *count regulation* are used to solve this problem.

Time window is used to discard the invalid vibrations. We assume that people can run as rapidly as five steps per second and walk as slowly as one step every two seconds. Thus, the interval between two valid steps is defined as being in the time window [0.2 s to 2.0 s]; all steps with intervals outside the time window should be discarded.

The ADXL345's feature of user-selectable output data rate is helpful in implementing the time window. Table 1 shows the configurable data rate (and current consumption) at $T_A = 25^\circ\text{C}$, $V_S = 2.5\text{ V}$, and $V_{DD I/O} = 1.8\text{ V}$.

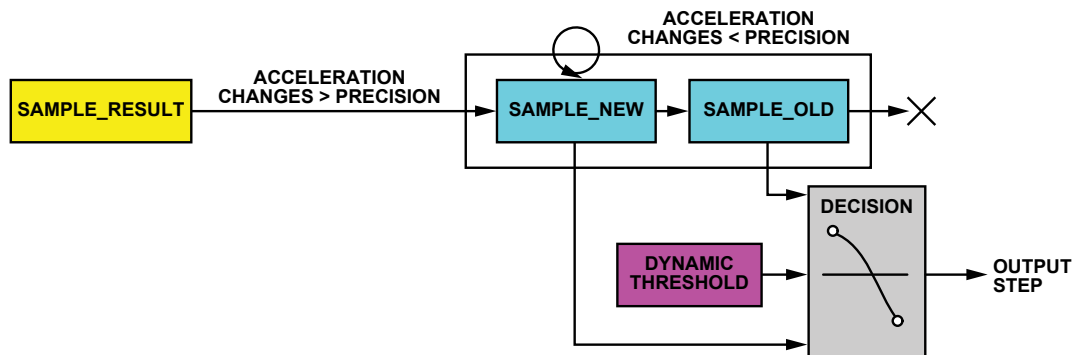


Figure 6. Dynamic threshold and dynamic precision.

Table 1. Data Rate and Current Consumption

Output Data Rate (Hz)	Bandwidth (Hz)	Rate Code	I _{DD} (μA)
3200	1600	1111	145
1600	800	1110	100
800	400	1101	145
400	200	1100	145
200	100	1011	145
100	50	1010	145
50	25	1001	100
25	12.5	1000	65
12.5	6.25	0111	55
6.25	3.125	0110	40

This algorithm uses a 50-Hz data rate (20 ms). A register named *interval* records how many times the data have updated during the two steps. If the value of interval is between 10 and 100, it means that the time between two steps is in the valid window; otherwise, the interval is outside the time window and the step is invalid.

Count regulation determines whether steps are part of a rhythmic pattern. The step counter has two working states: searching regulation and found out regulation. When the step counter starts working, it works in searching regulation mode. Suppose that *in regulation* exists after four continuous valid steps. Then the result is refreshed and displayed, and the step counter will work in found out regulation mode. Working in this mode, the step count would be refreshed after every valid step. But if even one invalid step is found, the step counter will return to searching regulation mode and search for four continuous valid steps.

Figure 7 shows the algorithm flowchart for the steps parameter.

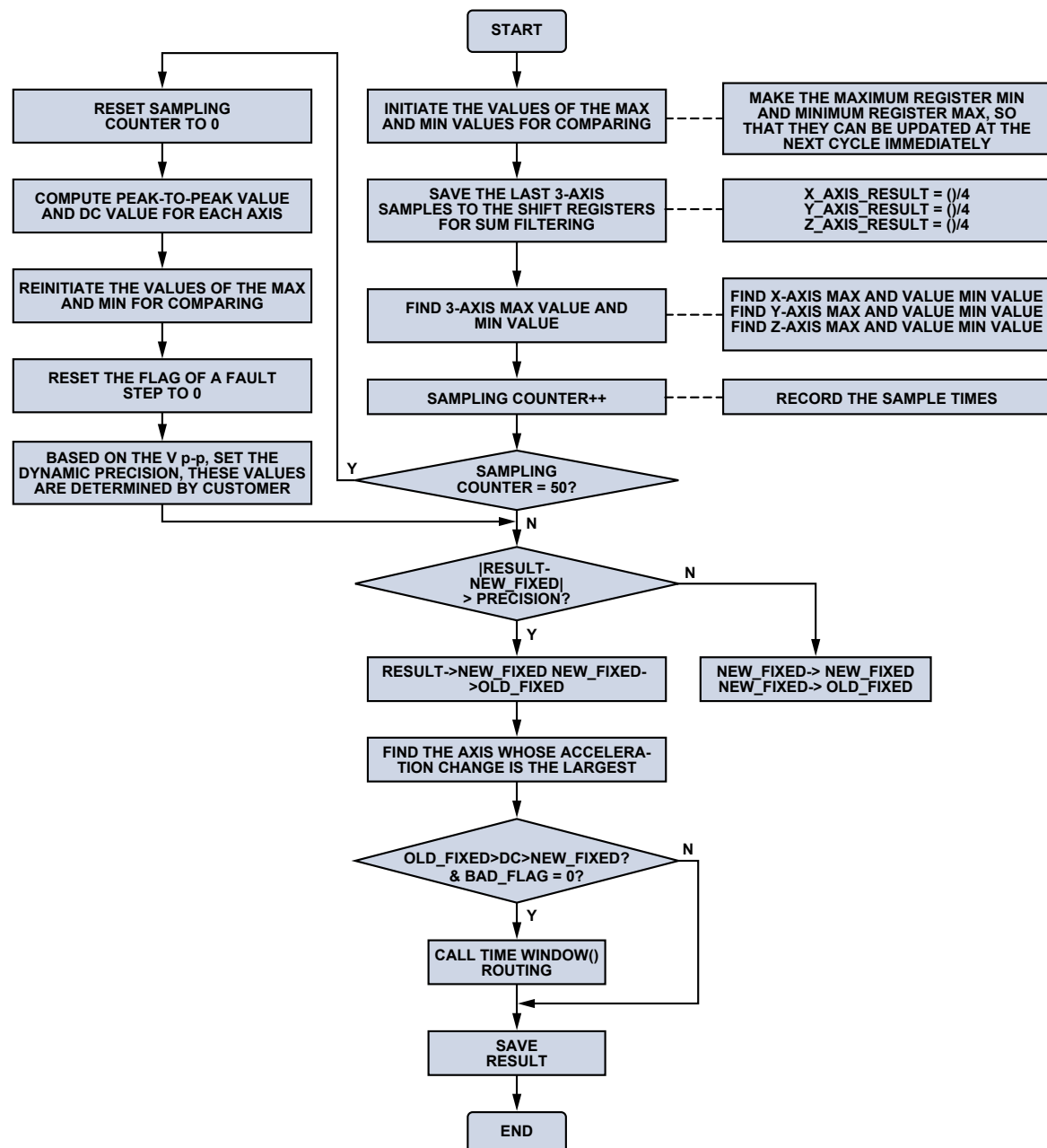


Figure 7. Steps parameter algorithm flowchart.

Distance Parameter

After computing the steps parameter according to the algorithm above, we can use Equation 1 to get the distance parameter.

$$\text{Distance} = \text{number of steps} \times \text{distance per step} \quad (1)$$

Distance per step depends on the speed and the height of user. The step length would be longer if the user is taller or running at higher speed. The reference design updates the distance, speed, and calories parameter every two seconds. So, we use the steps counted in every two seconds to judge the current stride length. Table 2 shows the experimental data used to judge the current stride.

Table 2. Stride as a Function of Speed (steps per 2 s) and Height

Steps per 2 s	Stride (m/s)
0~2	Height/5
2~3	Height/4
3~4	Height/3
4~5	Height/2
5~6	Height/1.2
6~8	Height
>=8	1.2 × Height

An interval of 2 s can be calculated accurately from the number of samples. Referring to the 50-Hz data rate, the processor can send the corresponding command to the PC every 100 samples. The processor uses a variable named *m_nLastPedometer* to record the step count at the beginning of every 2-s interval and a variable named *m_nPedometerValue* to record the step count at the end of every 2-s interval. Then the steps per 2 s is calculated by *m_nPedometerValue* minus *m_nLastPedometer*.

Although the data rate is 50 Hz, the ADXL345's on-chip FIFO makes it unnecessary for the processor to read the data every 20 ms, minimizing the burden on the host processor. The buffer has four modes: *bypass*, *FIFO*, *stream*, and *trigger*. In FIFO mode, data from measurements of the *x*-, *y*-, and *z*-axes are stored in FIFO. When the number of samples in FIFO equals the level specified in the samples bits of the FIFO_CTL register, the watermark interrupt is set. As previously discussed, people can run as fast as five steps per second, so the result should be refreshed every 0.2 s to show the real-time result. The processor only needs to fetch data from the ADXL345 every 0.2 s; it can be awakened by *watermark interrupt*. The other functions of the FIFO are also very useful. Using *trigger* mode, the FIFO can let us know what happens before the interrupt. Since the proposed solution does not use the other FIFO functions, they will not be discussed any further.

Speed Parameter

$\text{Speed} = \text{distance}/\text{time}$, so Equation 2 can be used to get the speed parameter, as steps per 2 s and stride have all been calculated according to the algorithm above.

$$\text{Speed} = \text{steps per 2 s} \times \text{stride}/2 \text{ s} \quad (2)$$

Calories Parameter

There is no accurate means for calculating the rate of expending calories. Some factors that determine it include body weight, intensity of workout, conditioning level, and metabolism. We can estimate it using a conventional approximation, however. Table 3 shows a typical relationship between calorie expenditure and running speed.

Table 3. Calories Expended vs. Running Speed

Running Speed (km/h)	Calories Expended (C/kg/h)
8	10
12	15
16	20
20	25

From Table 3, we can get (3).

$$\text{Calories (C/kg/h)} = 1.25 \times \text{running speed (km/h)} \quad (3)$$

The unit of the speed parameter used above is m/s; converting km/h to m/s gives Equation 4.

$$\begin{aligned} \text{Calories (C/kg/h)} &= 1.25 \times \text{speed (m/s)} \times 3600/1000 \\ &= 4.5 \times \text{speed (m/s)} \end{aligned} \quad (4)$$

The calories parameter would be updated every 2 s with the distance and speed parameters. So, to account for a given athlete's weight, we can convert Equation 4 to Equation 5 as indicated. Weight (kg) is a user input, and one hour is equal to 1800 two-second intervals.

$$\begin{aligned} \text{Calories (C/2 s)} &= 4.5 \times \text{speed} \times \text{weight}/1800 \\ &= \text{speed} \times \text{weight}/400 \end{aligned} \quad (5)$$

If the user takes a break in place after walking or running, there would be no change in steps and distance, speed should be zero, then the calories expended can use Equation 6 since the caloric expenditure is around 1 C/kg/hour while resting.

$$\text{Calories (C/2 s)} = 1 \times \text{weight}/1800 \quad (6)$$

Finally, we can add calories for all 2-second intervals together to get the total calories expended.

Hardware Connection

The ADXL345 is easy to connect to any processor using I²C® or SPI digital communications protocols. Figure 8 shows a simplified schematic of the demonstration equipment, which is powered by 3-V batteries. The /CS pin of the ADXL345 is tied to V_S on the board to choose I²C mode. A low-cost, precision analog microcontroller, the ADuC7024, is used to read data from the ADXL345, implement the algorithm, and send the result to the PC via a UART. SDA and SCL, the data and clock for I²C bus, are connected from the ADXL345 to the corresponding pins of ADuC7024. Two interrupt pins of the ADXL345 are connected to IRQ inputs of ADuC7024 to generate various interrupt signals and wake up the processor.

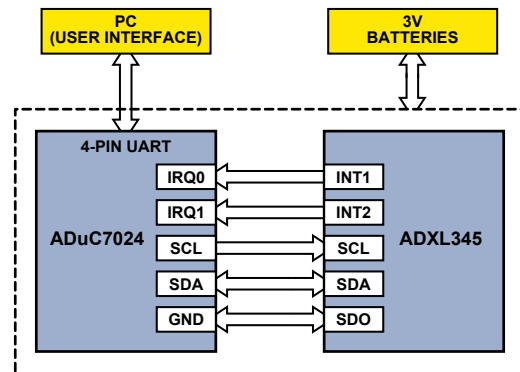
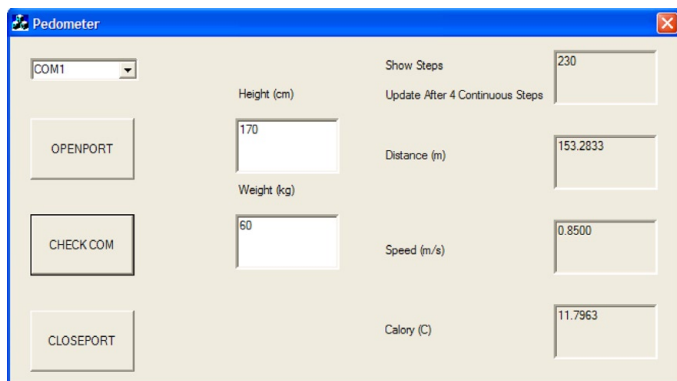


Figure 8. Simplified schematic of hardware system.

User Interface

The user interface displays the test data and responds to commands from the operator. The serial port should be opened and communications links should be started after the user interface (UI) is running. The demo can then run continuously. Figure 9 shows the test demo when the user is walking or running with the pedometer. Users can input their own height and weight data. Distance, speed, and calorie parameters will be calculated based on these data.



The screenshot shows a software window titled "Pedometer". It contains several input fields and buttons. On the left, there is a dropdown menu set to "COM1", an "OPENPORT" button, a "CHECK.COM" button, and a "CLOSEPORT" button. In the center, there are input fields for "Height (cm)" with the value "170" and "Weight (kg)" with the value "60". On the right, there are four display boxes showing calculated values: "Show Steps" at "230", "Update After 4 Continuous Steps" (empty), "Distance (m)" at "153.2833", "Speed (m/s)" at "0.8500", and "Calory (C)" at "11.7963".

Figure 9. Test demo when the user is walking or running with the pedometer.

Conclusion

The ADXL345 is an excellent accelerometer for pedometer applications. Taking advantage of its small, thin, 3-mm × 5-mm × 0.95-mm plastic package, pedometers using it can be found in medical instruments, as well as fancy consumer electronic devices. Its low 40-μA power requirement in measurement mode and 0.1 μA in standby mode make it an ideal choice for battery-powered products. Substantial savings in power result from the embedded FIFO, which minimizes the host processor's load. Also, the selectable output data rate can be used to save a timer

in the processor. The 13-bit resolution makes it possible for small peak-to-peak changes to be detected, leading to the possibility of high-accuracy pedometers. Finally, combining the 3-axis output feature and the algorithm described above, users can wear the pedometer in just about any location and position.

A couple of further ideas: If the application is extremely cost-sensitive, or if an analog-output accelerometer is preferred, the ADXL335—a small, thin, low power, complete 3-axis accelerometer with signal-conditioned voltage output—is recommended. If PCB size is of critical importance, the ADXL346 is recommended. This low-power device, with even more built-in features than the ADXL345, is supplied in a small, thin, 3-mm × 3-mm × 0.95-mm plastic package. Its supply voltage range is 1.7 V to 2.75 V.

Acknowledgements

The author would like to thank Charles Lee and Harvey Weinberg for their technical expertise.

References

1. Data sheets and additional product information on all Analog Devices products can be found at www.analog.com.
2. www.analog.com/en/mems/low-g-accelerometers/products/index.html.

Author

Neil Zhao [neil.zhao@analog.com] is a field applications engineer in ADI's China Applications Support Team, where he has been working for more than two years. He is responsible for technical support for ADI's Core Products and Technology Group across China. Neil graduated in January 2008 from Beihang University with a master's degree in communication and information systems.

