Winning Design Strategies for the Wearables Market

Introduction

Wearable devices, from smart watches to portable health and fitness trackers, are changing many aspects of our daily lives. The desktop revolution of the 1980s ushered in an era of unprecedented personal productivity for the Information Age. The advent of laptops in the 1990s, coinciding with the expansion of the Internet, freed us from the tethers of power cords and Ethernet cables. Then the explosive growth of cell phones and smartphones brought us unprecedented mobility and wireless connectivity. Today's "wrist-top revolution," coupled with the meteoric rise of the Internet of Things (IoT), is taking mobility to a whole new level: wearable computing.

In this article, we'll examine the concepts behind the user-experience-driven design methodologies that are being used to create some of the most successful wearable products on the market. We'll also consider the features and functions that drive a wearable product's energy budget and computational requirements including the selection of microcontrollers (MCUs) that meet the product's design requirements.

New Realities of the Wrist-Top Revolution

Smart watches, activity trackers, wearable GPS devices, heart rate monitors and smart glasses are prime examples of the wearable products that generated an estimated $8 billion in global sales in 2013, according to the Futuresource Consulting. Offering novel combinations of sophisticated functionality, easy-to-use connectivity, compact form factors, ultra-low-power processing and wireless connectivity, wearable devices are giving rise to entirely new classes of personal electronics that help us stay healthier, better informed and better equipped than ever before.

Although several leading smartphone manufacturers began experimenting with bulky wrist-top versions of their existing handset products several years ago, the wrist-top revolution kicked into high gear in early 2012 when innovative upstarts like the Pebble Smartwatch leapfrogged the smartphone makers with a new class of lightweight wrist-top devices that made it easier for end users to leverage the smartphones they already own. Garmin, Samsung, Sony, Fitbit, Magellan (see Figure 1) and other consumer electronics makers also joined the wrist-top revolution with their own smartwatches, activity trackers and other wearable products.

This disruptive market environment has also encouraged the emergence of small, agile startups whose innovative products such as the Misfit Shine (see Figure 1) fitness tracker are successfully competing for market share with established players.

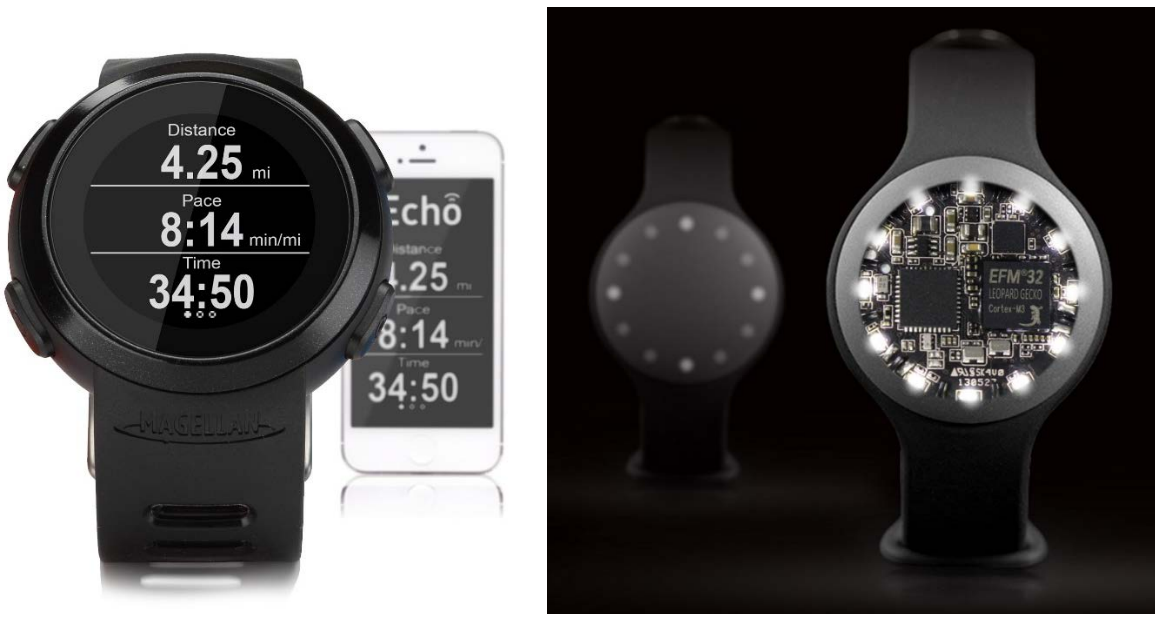

Figure 1. The Magellan Echo smart sports watch (left) and the Misfit Shine (right) leverage Silicon Labs' EFM32 Gecko MCU to achieve exceptional energy efficiency and long battery life.

A successful wearable device must deliver the right combination of price, performance, functionality and battery life, as well as a unique look, feel and behavior to differentiate itself from its competitors. MCUs, sensors, wireless electronics and attractive user interfaces must be shoehorned into a small footprint that can be comfortably worn on the wrist or elsewhere on one's body. Since such form-factor constraints leave little room for a battery, wearable systems must be extremely energy-efficient to achieve the longest possible operating periods between battery replacements or charges.

User Experience Drives Winning Designs

Integrating these diverse elements into a market-winning product requires complex design trade-offs to balance power, performance, functionality and form factor. Several manufacturers have successfully navigated this unfamiliar territory using a so-called "user experience-driven" design methodology that inverts many of the conventional priorities and practices used by embedded developers.

The design process for an embedded system typically begins with defining the functions and capabilities that will serve as the project's top-level drivers. Conversely, designing a wearable product frequently begins with defining the "user experience" it will need to produce. These requirements define a product by the way it looks, feels and interacts with the end user, as well as the impressions, feelings and emotions it evokes. The next step in this design process is to translate the user experience into a “use case,” a set of top-level functional requirements used to define the product’s hardware and software elements.

Apple was one of the early pioneers of this strategy. They used it with great success to define new markets -- and capture existing ones. If you have any doubts about the importance of a well-crafted user experience, consider how the Apple iPod’s unique control wheel, jewel-like case designs and easy-to-use iTunes software helped the company transform and eventually dominate the digital music player market.

Defining the User Experience

The requirements that define a wearable product’s user experience fall into two categories:

- **Functionality** – the unique look, feel, features and functions that differentiate a wearable product.
- **Ease of use** – a set of requirements that enables easy set-up, intuitive operation and minimal maintenance. Long battery life plays an important role in ease of use since having to recharge a wearable device every few days can be frustrating and cause users to abandon the product.

Together, these elements define a user experience that can easily be translated into a use case forming the foundation of a product’s design. Depending on the application, defining the user experience might involve designing a wearable case that has an inviting texture, ergonomic shape and design elements that convey a specific feeling. Other products might require creation of special visual paradigms for controls and displays that make complex operations simple and intuitive.

Defining the Use Case

Once a product’s user experience has been clearly defined, it must then be translated into a use case whose functional requirements will drive the wearable product’s design. A detailed use case can provide important information that makes it easier to perform accurate trade-off studies for nearly every aspect of a wearable design.

A use case should include the tasks the wearable device is expected to perform, the required resources and the conditions in which it is expected to operate. These details typically include the types of data the device will collect, how it will interact with users and other devices, anticipated operating environment (temperature, water resistance, impact resistance, etc.), operational modes (data collection and analysis, user interactions, communication, etc.) and how frequently it synchronizes with other devices.

Armed with these guidelines, the design team can start to identify the sensing, computing and communication components that best meet the application’s requirements. Meanwhile, the bill of materials (BOM) cost and energy budgets are developed in parallel with the preliminary design requirements, giving the team the necessary parameters to converge on an optimal design approach

Use Case Aids Energy Management

Because battery life plays such an important role in wearable designs, let's take a closer look at the energy-related portion of the use-case-driven design process.

To accurately model how design choices affect a wearable device's battery life, the use case should include detailed descriptions of factors affecting energy consumption, such as:

- The type of data the device must collect from the outside world and how frequently it must be collected.
- Whether the user interacts with the device via an app, touch display, buttons or both. If so, what types of information does it communicate, and how frequently will it be used?
- How the device communicates with other wearables, a smart phone, a local network or the Internet. Power requirements can vary widely depending on the wireless interface (e.g. Bluetooth, Wi-Fi or ZigBee) and how it is implemented.
- How often the device synchs or exchanges data with its peers or a host system. (Synching too frequently with a host system such as a smartphone can decrease battery life.)

Once assembled, the use case should provide a detailed picture of the system's various operating modes and how much time the device will spend in each one. This will be the foundation of a system energy budget and inform any design tradeoffs needed to maximize battery life.

Use Case Aids MCU Selection, Optimization

The energy-related portion of the use case should include as much information as possible about the sensing, control and computational tasks the wearable device will be performing, as well as which of these tasks will be performed by the MCU or one of its peripherals. This will aid in selecting the MCU family that best meets the wearable application's requirements and developing strategies that make the most of the MCUs energy-friendly features.

You can create a good initial estimate or your wearable application's computational requirements by identifying the software functions and algorithms it will have to perform and how frequently they will occur. Consider, for example, a fitness monitor in which the MCU senses the user's physical activity through a multi-axis accelerometer, monitors cardiac activity using an IR proximity sensor, and uses additional sensors to detect temperature, humidity, blood oxygen level and even ultraviolet (UV) light exposure.

The MCU must then filter the raw sensor data to eliminate noise and other artifacts before determining the actual step count and frequency or combine it with heart rate data to differentiate between specific types of activities and other biometric inputs.

Among the several excellent 32-bit processor architectures used in modern MCUs, the ARM Cortex family of 32-bit RISC CPUs has emerged as the leading processing core for embedded designs thanks to its efficient architecture, easily-extensible instruction set, and large base of development tools and code libraries. Over the years, ARM has created several series of its Cortex CPU, each optimized around a specific set of requirements. The ARM Cortex-M series of processor cores, for example, which was developed specifically for deeply-embedded MCUs, is used in applications where performance must be balanced against energy efficiency and low solution cost. The Cortex-M series provides core options that address a wide range of wearable design attributes including price, battery life, processing requirements and type of display (see Table 1.)

Table 1. The ARM Cortex-M series is designed to address diverse of design considerations.

Attribute	Basic	Intermediate	Advanced
Display	None/LED	Small LCD	Medium LCD
Processing Needs	Low	Medium/No FPU	High / FPU
Battery Life	Longest	Long	Medium
Price	Lowest	Low	Medium
CPU Class	Cortex-M0+	Cortex-M3	Cortex-M4

Within the Cortex-M series, the M3 and M0+ cores are optimized for cost-sensitive applications that still require high-performance computing and fast system response to real-world events with low dynamic and static power consumption. The more complex and capable M4 core offers dramatically faster completion of the computation-intensive algorithms frequently used in bio-monitoring applications. Its enhanced instruction set includes a library of powerful digital signal processing (DSP) functions. The M4 core's single-precision floating point unit (FPU) can dramatically shorten execution times, reducing the period the MCU must remain awake and thereby minimizing its overall energy footprint.

Sleep Deep for Long Life

To reduce the MCU's impact on the wearable platform's energy budget, it is important to minimize the frequency and duration of any task that requires it to awaken from a low-power sleep mode. As such, the use case should include the frequency with which the MCU's various tasks are expected to occur and whether their execution is event- or schedule-driven.

One of the primary ways to optimize a low-power embedded design is to find the lowest sleep mode that still provides adequate response to real-time events. Most MCUs using the Cortex-M processing core support multiple sleep modes.

Silicon Labs' EFM32 Gecko family, for example, uses standard 32-bit ARM Cortex-M cores combined with an energy-optimized set of peripherals and clocking architecture. The EFM32 architecture has been designed from the ground up specifically for energy-sensitive applications. The architecture features a range of power modes that enable developers to achieve the optimal energy efficiency required by wearables (see Table 2).

Sleep/Standby
(Known as EM1 mode for EFM32 MCUs) – Enables quick return to active mode (usually via interrupt) at the expense of slightly higher power consumption. In this mode, power consumption for EMF32 = 45 µA/MHz; typical equivalent 32-bit MCU = 200 µA.

Deep Sleep
(EM2 mode for EFM32) – Leaves the MCU's critical elements active while disabling high-frequency system clocks and other non-essential loads. In this mode, power consumption for EMF32 is as low as 900 nA; typical equivalent 32-bit MCU = 10 µA to 50 µA.

Stop
(EM3 mode for EFM32) A deeper version of Deep Sleep Mode that enables further power savings while retaining limited autonomous peripheral activity and fast wakeup. In this mode, power consumption for EFM32 = 0.59 µA; typical equivalent 32-bit MCU = 10 µA to 30 µA.

Off
(EM4 or shutoff mode for EFM32) – This "near-death" state preserves the minimum compliment of functionality needed to trigger wakeup from an external stimulus. The energy savings comes at the cost of significantly longer wake-up time. In this mode, power consumption for EFM32 = 20 nA (420 nA with RTC running); typical equivalent 32-bit MCU = 1.5 µA.

Backup Battery Mode
A unique EFM32 feature that offers an attractive alternative to Shutoff Mode, preserving a few more critical functions and enabling much faster wake-up.

Table 2. EFM32 MCUs provide a range of power modes useful for wearable applications.

EFM32 with 3 V power supply Application from memory	EMO Run Mode	EM1 Sleep Mode	EM2 Deep Sleep	EM3 Stop Mode	EM4 Shutoff Mode
Current Consumption	110 µA/MHz	45 µA/MHz	0.9 µA	0.6 µA	20 nA
Wake-up time	-	0	2 µs	2 µs	160 µs
CPU (Cortex-M3/M0)	On	-	-	-	-
High frequency peripherals	Available	Available	-	-	-
Low frequency peripherals	Available	Available	Available	-	-
Asynchronous peripherals	Available	Available	Available	Available	-
Full CPU and SRAM retention	On	On	On	On	-
Power-on reset/brown-out detector	On	On	On	On	On
Wake-up events	Any	Any	32 kHz peripherals	Async IRQ, I^2C slave, analog comparators, voltage comparators	Reset, GPIO rising/falling edge

Smart Peripherals Equally Smart Designs

Many MCUs are equipped with at least a few peripherals that perform routine timekeeping, I/O and housekeeping tasks while the CPU remains in one of its low-power sleep modes. Some MCUs are also equipped with autonomous peripherals that perform multiple functions (e.g., counters/timers, ADCs, DACs, GPIOs, serial transceivers, etc.) without CPU intervention. For example, all of the on-chip peripherals supported by EFM32 Gecko MCUs can function autonomously and remain active in one or more of the device's sleep modes. This contrasts sharply with other MCUs in which the lowest energy-saving modes only support very basic keep-alive functions, such as GPIO wake-ups and real-time clock (RTC) operation.

In addition to the counter/timer, ADCs, DACs, GPIOs and serial communication elements offered by other MCUs, the EFM32 MCU family's set of peripherals includes:

- A capacitive sense controller that senses touchpad contact and coordinates within an n-by-n grid (up to 16 points) without CPU intervention. The CPU remains active down to the EM2 energy mode.
- An LCD driver that can drive numeric LCD or TFT display via DMAs from memory without CPU intervention.
- Analog comparators that enable monitoring of threshold voltages for alert/alarm conditions without CPU intervention.
- The Low Energy UART (LEUART) with DMA, which can remain in EM2 while receiving a large amount of data without waking up the CPU.
- A Low Energy Sensor (LESENSE) interface that can work with up to 16 analog sensors including resistive, inductive, and capacitive to create an autonomous state machine.

The activities of most peripheral functions, including serial communications, counters/timers, analog and digital comparators and higher-level I/Os, are coordinated by a separate low-power reflex bus (known as the Peripheral Reflex System or PRS). Events and signals from one peripheral can be used as input signals or triggers by other peripherals to ensure timing-critical operation and reduced software overhead. These advanced features enable EFM32 Gecko MCUs to deliver exceptional 32-bit computational performance while enabling extremely low-energy wearable designs and long battery life.

The Power of Software in Minimizing Energy Consumption

To simplify the development process, speed time to market and achieve optimal energy efficiency, wearable product designers must consider the effectiveness and ease of use of the development ecosystem provided by their MCU vendor. To make the development process easier, faster and more efficient, developers should use a comprehensive, simple-to-use platform that provides everything they need to complete their projects, from initial concept to final product. Equally important, the ecosystem should offer tools that enable developers to optimize their product's energy consumption.

For example, Silicon Labs' Simplicity Studio™ development platform (available to developers at no charge at www.silabs.com/simplicity-studio) includes real-time energy profiling and analysis tools for estimating power consumption and balancing performance and energy efficiency. The Simplicity Studio energyAware Battery Calculator helps developers estimate current consumption and battery life. Developers can select EFM32 MCU Energy Modes and battery configuration and estimate power consumption before writing any code. The energyAware Profiler analyzes current consumption in real-time, enabling the developer to identify areas of code that should be optimized if current draw is too high.

Simply by viewing a graphical output of current consumption, the developer can quickly see if there are any significant increases in consumption. Taking this a step further, having the ability to click on a point in the graph, profile the application and show the corresponding line of C code associated with the current consumption is extremely valuable to developers. This capability enables developers to trace spikes in current consumption to specific lines of code and perform optimizations. Ultimately, this energy-aware capability can be especially critical in battery-powered wearable applications where every microamp and even nanoamp counts.

Conclusion

Designing winning products for the wrist-top revolution requires a deep understanding of the new realities of wearable application requirements and a fresh approach to integrating complex technologies and high-performance components into space-and power-constrained designs. Smart watches, portable fitness trackers, smart glasses and other wearable computing devices are changing everything we know about designing portable electronics.

Wearables are rewriting the rules for design engineers who must seamlessly integrate sophisticated sensing, computing, display and wireless technologies into affordable, compelling, ultra-compact designs that can operate for months on a single user-replaceable battery or other limited energy sources. New wearable computing products are appearing on the market at a faster pace, resetting our expectations for the end user experience with each design innovation. And this wrist-top revolution has only begun.

Learn more about Silicon Labs' Internet of Things solutions at www.silabs.com/IoT