December 2019

Motion activated wireless reconnection

Jonathan D. Hurwitz

Motion activated wireless reconnection

ABSTRACT

The time taken from the removal of a laptop or other device from a bag by a user to the establishment of connection to a wireless network can be significant, e.g., ten seconds or more. This disclosure describes techniques to reduce this delay to nearly zero. Per the techniques, upon detection of motion by on-board motion sensors of a device, the device wakes from a sleep state and automatically scans for and connects to a WiFi network.

KEYWORDS

- Sleep mode
- Inertial measurement unit (IMU)
- WiFi
- Bluetooth
- Network reconnect
- Motion activation
- Motion sensor
- Motion detection

BACKGROUND

Users of devices such as laptops carry the device between locations. Before a move, the user may put the laptop away, and later, take it out again to resume a task. There is a delay between when a laptop is brought out of its bag and opened, and the point at which a wireless network, e.g., WiFi or Bluetooth, has been connected to and made available for use.

When a user puts away the laptop, the device enters a sleep mode. Sleep modes (or states) are generally described, e.g., by the advanced configuration and power interface (ACPI) as follows:

- State G0 (S0): This is the ON state
- State G1: This is a sleep state, with the following sub-types.
  - S1 sleep state: The S1 sleep state is a low wake-latency sleep state. In this state, no system, e.g., CPU or chipset, context is lost and hardware maintains context.
  - S2 sleep state: The S2 sleep state is a low wake-latency sleep state. This state is similar to the S1 sleep state except that the CPU and system cache context is lost, e.g., the OS is responsible for maintaining the caches and CPU context. Control starts from the reset vector of the processor after a wake event.
  - S3 sleep state: The S3 sleep state is a low wake-latency sleep state where all system context is lost except system memory. CPU, cache, and chipset context are lost in this state. Hardware maintains memory context and restores some CPU and L2 configuration context. Control starts from the reset vector of the processor after a wake event.
  - S4 sleep state: The S4 sleep state is the lowest power, longest wake-latency sleep state. In order to reduce power usage to a minimum, it is assumed that the hardware platform has powered off all devices. Platform context is maintained.
- State G2: This is a soft off state, similar to S4 except the OS does not save context.

- State G3: This is a mechanical off state, e.g., the laptop is off electrical power.

DESCRIPTION

This disclosure describes techniques to reduce the delay between the removal of a laptop from its bag (and the opening of its clamshell) and its connection to a wireless network. Per the techniques, upon the detection of motion of a laptop by on-board motion sensors, the laptop wakes from a sleep state and scans for and connects to a WiFi network. The motion sensors can include inertial measurement units (IMUs), e.g., accelerometer, gyroscope, etc., proximity sensors, motion-sensing radar, etc.

Always-on architectures

Per the techniques, motion sensors of a device are powered on while the device is in sleep mode. The techniques are thus most useful when the device is in a sleep states under G1, e.g., the S1-S3 sub-states. It is possible that S4 may be used for long periods of inactivity, such as when a laptop hasn't been used for many days and it is assumed that the user may not come back for a while.

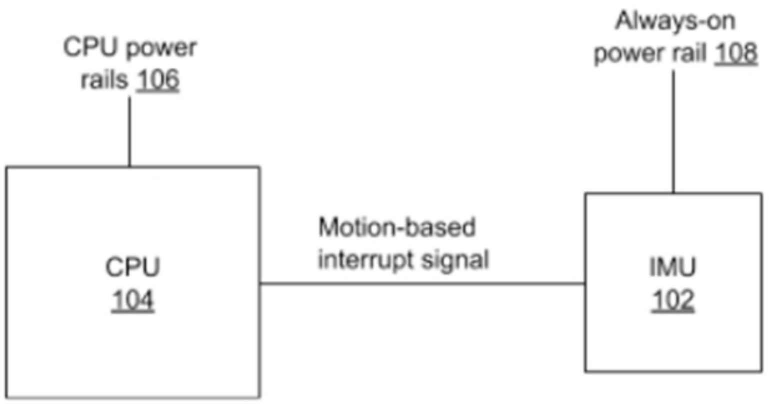

Fig. 1: CPU-IMU connection and IMU connection to a power rail

Fig. 1 illustrates the CPU-to-IMU connection and the IMU connection to a power rail, per the disclosed techniques. As illustrated in Fig. 1, the IMU (102) is connected to a power rail (108) that stays on when the system enters states S1-Sn, where n equals 2, 3, or 4. In particular, the IMU may not be connected to the core power rails (106) of the CPU (104) or downstream regulated rails powered by the CPU line, since such rails go low during sleep states. In this manner, the IMU or other motion sensor is kept on at all times that the machine is not in mechanical off. The power rail of the IMU can come from the device battery; this can be achieved by running the core power management IC(s) (PMIC) to regulate the correct voltage for the always-on rail. However, it is possible that other components may be connected to rails that are themselves connected to the core PMIC. To avoid turning on extra components without adding switching complexity (e.g., by gating power with various FET networks), motion sensors can have a dedicated battery and PMIC. Such a dedicated battery can be charged by the main battery during an awake state. Simply using the core PMIC and system battery can achieve the same end result, but with potentially more power usage.

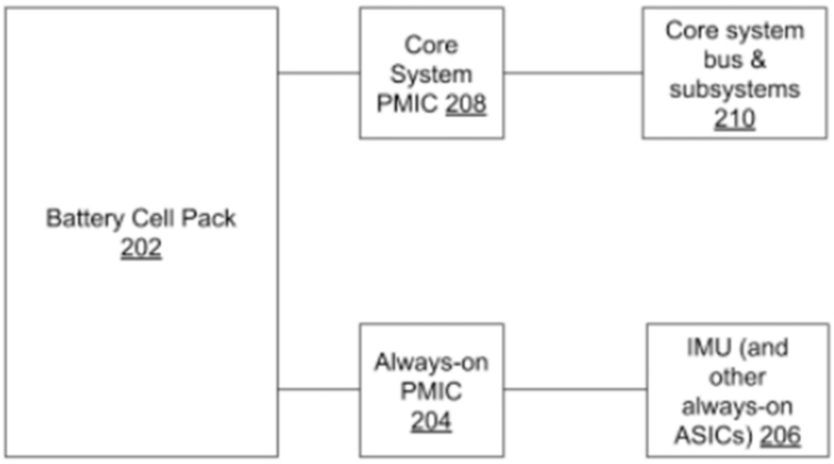
Core System PMIC 208
Core system bus & subsystems 210
Battery Cell Pack 202
Always-on PMIC 204
IMU (and other always-on ASICs) 206

Fig. 2: An auxiliary (always-on) PMIC that exclusively powers the always-on power rails

Alternately, as illustrated in Fig. 2, an auxiliary, always-on PMIC (204) which uses the core system battery (202) as power source can operate the always-on rails used by the IMU and other always-on ASICs (206). The auxiliary PMIC runs parallel to the core system PMIC (208), which also uses the core system battery as power source but operates the core system bus and subsystems (210). In some configurations, the core system PMIC may itself have an always-on block which can be used to drive the IMU and other always-on ASICs. The techniques disclosed herein can be implemented as a system-on-chip, e.g., as a custom ASIC with on-board IMU, memory, etc. In some configurations, the CPU has a platform controller hub (PCH) that takes interrupts such that no polling is needed by the CPU. In some configurations, the interrupts are handled by a form of dedicated controller ASIC or by the CPU itself.

Motion detection that is robust to false positives and false negatives

It is possible for a motion sensor such as an IMU to register false positives and cause unwanted CPU wake-ups. Some examples of where this might occur are as follows:

- Laptop is in the backpack of a walking user.
- Laptop is on a surface inside a moving vehicle and experiences vibration or spurious movement.
- Laptop is in a backpack that gets bumped.

It is desirable to activate the wake-up-and-scan behavior when the motion is intentional, e.g. when the user reaches for the laptop and removes it from its bag. Per the techniques, a motion-periodicity detector detects motion that appears periodic, e.g., due to the user walking or swaying within a vehicle. If periodic motion is detected, no wake signal is sent to the CPU.

To detect periodicity in motion, motion signals generated by the IMU or other motion sensor are low-pass filtered with a cut-off frequency ω_c. The value of ω_c is tuned based on a target false alarm rate. Low-pass filtering can be done in the time-domain, e.g., via moving-average filters, or in the frequency domain. Periodicity in motion signals is detected, for example, by subjecting the filtered motion signals to a low-cost fast Fourier transform (FFT). Alternatively, to optimize the energy consumed by motion and periodicity detection ASICs, the periodicity check can be done in the time domain, e.g., by using peak detection, where if the distance between subsequent peaks of the filtered motion signals lie within a certain range, the signal is determined to be periodic.

The motion signals are passed through a binary classifier, and if positively classified as a reach event, a wake interrupt is sent to the CPU. The motion sense unit of the device can have its own memory space on the ASIC in order to store training data for the binary classifier. For example, movement that triggers a wake but is not followed by the opening of the laptop lid within a certain number of seconds is automatically labeled as a negative (false) wake event. Motion data that is obtained, classified, and followed by the opening of the laptop lid is automatically labeled as a positive (true) wake event. Memory space on the ASIC is provisioned for cases where the CPU does not wake up, e.g., false wake events; for cases where the CPU does wake up, e.g., true wake events, whether detected or not, the powered-on CPU can provide access to main memory (or disk) for the purposes of storing training data. The data labeling can be used for retraining the classifier model and for providing a custom experience for each user, since different users reach for their laptops in different ways. Users are provided with options to turn off automatic wake based on motion data and can control whether motion data is obtained and used for retraining the classifier model. Motion data, if stored, is stored locally on the device and is used only for the purpose of detecting wake events.

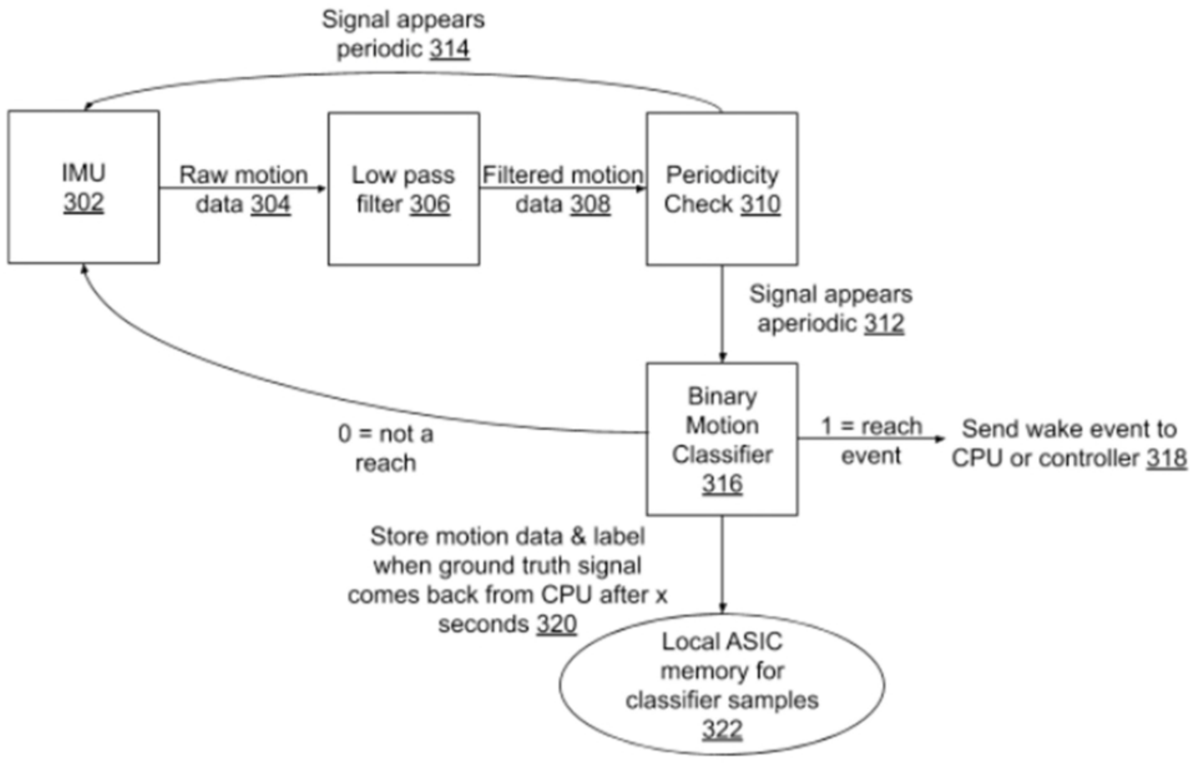
Signal appears periodic 314
IMU 302
Raw motion data 304
Low pass filter 306
Filtered motion data 308
Periodicity Check 310
Signal appears aperiodic 312
Binary Motion Classifier 316
0 = not a reach
1 = reach event
Send wake event to CPU or controller 318
Store motion data & label when ground truth signal comes back from CPU after x seconds 320
Local ASIC memory for classifier samples 322

Fig. 3: Processing pipeline to detect motion in a robust manner

Fig. 3 illustrates an example processing pipeline to detect motion in a robust manner, per techniques of this disclosure. An IMU (302) generates raw motion data (304), which is low-pass filtered (306) to remove high-frequency artifacts. The filtered motion data (308) is tested for periodicity (310). If the signal is detected to be periodic (314), e.g., with a dominant frequency, no wake signal is sent to the CPU. If the signal is detected as aperiodic (312), a binary motion classifier (316) determines if the motion signal is indicative of a reach event or of spurious movement, e.g., a false reach event. If the binary motion classifier determines that a reach event took place, a wake signal is sent to the CPU or controller, e.g., a PCH (318). If the binary motion classifier determines that a reach event did not take place, then no wake signal is sent to the CPU or controller. In either case, a determination is made of the ground truth, e.g., whether the wake event was true or false, based on whether the laptop lid was opened (as reported by the CPU) after a certain number of seconds (320). When permitted by the user, the motion data and ground-truth label are stored in memory (322) for the purposes of re-training the binary motion classifier. Upon receiving a wake signal, the CPU and operating system connect, or scan to connect, to a cached wireless network. Connecting to the wireless network upon the detection of genuine user action, per the techniques of this disclosure, ensures that the laptop is already wirelessly connected by the time the user opens it.

The binary motion classifier can be trained based on the following automatically labeled events:

- **False positive:** The classifier determines that a wake event took place, but the laptop lid is not opened after the passage of a certain number of seconds.
- **False negative:** The classifier determines that no wake event took place, but the laptop was opened within a certain number of seconds.
- **True positive:** The classifier determines that a wake event took place, and the laptop was opened within a certain number of seconds. True positives can be used for training based on class balance, e.g., if there are many true positives, training with more true positives can be avoided.
- **True negative:** The classifier determines that no wake event took place, and the laptop was not opened within a certain number of seconds. True negatives can be used for training based on class balance, e.g., if there are many true negatives, training with more true positives can be avoided.

In each of the above cases, the timeout, e.g., the number of seconds used to determine true/false positive/negative can be determined by design, or by trial-and-error, or if permitted by the users, based on prior laptop open events.

CONCLUSION

The time taken from the removal of a laptop or other device from a bag by a user to the establishment of connection to a wireless network can be significant, e.g., ten seconds or more. This disclosure describes techniques to reduce this delay to nearly zero. Per the techniques, upon detection of motion by on-board motion sensors of a device, the device wakes from a sleep state and automatically scans for and connects to a WiFi network.

REFERENCES

[1] "Sleep states" https://software.intel.com/sites/manageability/AMT_Implementation_and_Reference_Guide/default.htm?turl=WordDocuments%2Fsleepstates.htm accessed on Dec. 26, 2019.

[2] Crisan, Adrian. "Motion on computer." U.S. Patent Application 13/370,687, filed Feb. 10, 2012.

[3] Mucignat, Andrea and Saurabh Gupta. "Motion sensor data processing using various power management modes." U.S. Patent Application 13/561,412, filed Jul. 30, 2012.