Entwurf von drahtlosen Anwendungen für Echtzeitkommunikation mit sehr geringem Leistungsbedarf

Dieser Applikationsbericht beschreibt die kosteneffektive Implementierung eines Vertreters der MC1319x-Modem-Familie mit einem kostengünstigen Low-Pin-8-Bit-Mikrocontroller aus der MC9S08QG-Familie zur drahtlosen Anbindung eines dreiachsigen Beschleunigungsmessers MMA7260QT.

Quelle: Ultra-Low-Power Wireless Designs by Radomir Kozub and Pavel Lajsner, Freescale, www.freescale.com/beyondbits, übersetzt von FS

Das Design Team wurde gefragt, ob es ein drahtloses System zur Demonstration eines triaxial arbeitenden Accelerometers (Freescale MMA7260QT) entwickeln könne. Das Ziel bestand darin, die seriellen Kabel des bestehenden Originals durch eine Funkverbindung unter Nutzung der neusten Freescale-Technologie zu ersetzen. Die Accelerometer-Daten sollten in Echtzeit vom Sensor zum PC übertragen werden.

Schlüsselparameter der drahtlosen Demo-Lösung waren:

- Sehr kleiner Footprint (für Hardware und Software)
- Low Power
- Batteriebetrieb
- 20 m Reichweite
- gedruckte Antenne
- Realtime-Fähigkeit
- geringe Kosten
- als Referenz-Design nutzbar

Nach anfänglichen Überlegungen wurde klar, dass die Hauptaufgabe darin bestand, über einen geringen Stromverbrauch eine geringe Leistungsaufnahme zu erreichen. Das Ziel dieser Demonstration war die Durchführung von Accelerometer-Messungen mit einer Rate von 30 Übertragungen pro Sekunde. Bekannt war zwar die mögliche Batterie auf dem Sensor Board, jedoch bestand zunächst Unklarheit darüber, welche Drahtlostechnologie wohl die geeignetste wäre.

Zu Beginn wurden AA- und AAA-Alkaline-Batterien in Betracht gezogen. Dabei zeigte sich, dass das von den Batterien beanspruchte Volumen den Platz auf der Platine dominieren würde. Da eine Very-Low-Power-Technologie demonstriert werden sollte, wurde entschieden, auf Lithium-Primärzellen überzugehen. Der populäre 220-mAh-Typ CR2032 3V erwies sich zusammen mit einem geeigneten Batteriehalter, als optimal, besonders unter dem Gesichtspunkt Energiedichte (bzw. Formfaktor). Damit schien eine erfolgversprechende Demo möglich.

Die richtige Drahtlostechnologie

Der nächste Schritt bestand darin, die bestgeeignete Low-Power-Drahtlostechnologie zu finden, um die Daten vom Accelerometer zum Computer zu senden. Der 2,4-GHz-Industrial-, Scientific- and Medical-Bereich (ISM) wurde ausgewählt, um allgemeine Akzeptanz zu sichern und um Vorteile aus der damit verbundenen geringen Board-Größe sowie der kostengünstigen Drahtantenne (gedruckt) zu ziehen.

Drei gut bekannte Standardtechnologien arbeiten im 2,4-GHz-ISM-Band: WiFi, Bluetooth und IEEE-802.15.4-Technologie. WiFi ist vielleicht am besten bekannt, erlaubt die höchsten Datenraten und ist optimal für den Computerbetrieb. Jedoch ist es nicht für extrem geringe Leistungsaufnahme (Low-Power Operation) bzw. lange Batterielaufzeiten entwickelt. Bluetooth ist in dieser Hinsicht schon viel besser, überbrückt jedoch auch nur sehr geringe Entfernungen bis etwa 10 m und ist für den Betrieb von mobilen Telefonen, Headsets und anderen Handheld-Konsumgütern vorgesehen. Die IEEE-802.15.4-Technologie andererseits wurde von Anfang an dafür entwickelt, Monate oder Jahre ohne Batteriewechsel auszukommen und Entfernungen zwischen 20 und 100 m zu überbrücken. Dabei ist eine kleine Batterie möglich. Der Preis für diese Vorzüge sind geringe Datenraten, wie sie in Sensor- und Steuerungssystemen vorkommen. Aus diesen Gründen wurde eine 802.15.4-basierende Funklösung gewählt, welche die Entwickler ZSTAR nannten.

Demo-Überblick

Die Demo nutzt das ZSTAR-Sensor-Modul und ein 802.15.4-Wireless-Datacollector-Modul, welches mit dem PC verbunden ist (typisch über USB). Es standen mehrere Module zur Verfügung, die als Datacollector infrage kamen. Genutzt wurde eine Ausführung mit drei bedeutenden Freescale-Produkten:

- triaxiales Accelerometer MMA7260QT, genannt Triax
- 16-Pin/8-Bit-Mikrocontroller MC9S08QG8 (MCU)
- 802.15.4-Transceiver MC13191 (Radio Modem)

All diese Bausteine sind für sehr geringe Leistungsaufnahme ausgelegt. Um die Batterie-Lebensdauer noch weiter zu erhöhen, ist es notwendig, die möglichen Low-Power-Arbeitsweisen voll auszuschöpfen, welche diese drei Bausteine anbieten.

Low-Power-Überlegungen

Verschiedene wichtige Faktoren bestimmen bei diesem Design die Leistungsaufnahme. Dies sind:

- **Power-Saving-Betriebsweisen für Triax, MCU und Radio Modem**

Diese werden von jedem Teil des kompletten Kommunikationszyklus´ (die Periode, in der ein Datensatz von Accelerometer-Werten zum PC gesendet wird) bestimmt.

- **Anzahl der pro Sekunde gesendeten Datenpakete**

Dies ist in anderen Worten ausgedrückt, die Übertragungsrate der Kommunicationzyklen.

- **Anzahl der gesamten Pakete pro Sekunde**

Das 802.15.4-Protokoll sieht eine Bestätigung für jedes Datenpaket vor, um Versorgungsleistung zu sparen, wird hier jedoch

Modem-Arbeitsmodus	Beschreibung	typ. Stromverbrauch
Off	gesamtes System aus	0,2 µA
Hibernate	RAM/Register-Inhalt bleibt, Digital IO behalten Status, Referenzoszillator aus	1 µA
Doze	Quarzreferenzoszillator ein, RAM/Register-Inhalt bleibt, Digital IO behalten Status, Ausgangstakt nutzbar, Timer für Wake-up nutzbar	35 µA
Idle	Referenzoszillator läuft, schneller Übergang zu Rx- und Tx-Modes	500 µA
Receive	Empfangsmodus oder CCA	37 mA
Transmit	Sendemodus	30 mA
MCU-Arbeitsmodus	**Beschreibung**	**typ. Stromverbrauch**
Stop1	alle I/O Pins automatisch auf Default gesetzt, interne Schaltung der MCU fast vollständig aus, RAM-Inhalt verloren	475 nA
Stop2	RAM-Inhalt bleibt, alle I/O Pin behalten Status, Wake-up wird als Start vom Reset behandelt	600 nA
Stop3	RAM/Register-Inhalt bleibt, I/O Pin behalten Status aufrecht, Wake-up wird als Interrupt Service behandelt	700 nA
Wait	CPU-Kern wird nicht getaktet, alle peripheren Vorgänge laufen weiter	1 mA
Run (1)	alle peripheren Vorgänge sind gestoppt, CPU läuft mit 32,768 kHz vom Quarz	95 µA
Run (2)	alle peripheren Vorgänge sind gestoppt, CPU läuft mit ~32 kHz vom internen Oszillator (ohne frequenzstarre Schleife)	150 µA
Run (3)	alle peripheren Vorgänge laufen weiter, CPU-Kern-Takt enabled (8 MHz)	3,5 mA
Triax-Arbeitsmodus	Beschreibung	typ. Stromverbrauch
Sleep	Ausgänge deaktiviert	3 µA
Run	triaxiales Accelerometer arbeitet	500 µA

Tabelle 1: Stromverbrauch für Modem, MCU und Triax bei verschiedenen Zuständen

nur nach je acht Übertragungen vom Sensor bestätigt.

- **Paketlänge**

Das Protokoll benötigt, um simpel und problemlos zu arbeiten, einen minimalen Overhead und eine minimale Datengröße. Um die möglichen Arbeitsweisen der genannten Komponenten besser einschätzen zu können, wurden sie in Tabelle 1 zusammengestellt.

System-Zugeständnisse für Low Power

Die Arbeitsweise des Sensor-Boards muss unter dem Aspekt der Leistungsaufnahme betrachtet werden. Da die Anzahl der übertragenen Pakete und deren Größe gleichbleibend ist, sollte zunächst das Tastverhältnis herangezogen werden.

Low Power oder Ruhezeiten innerhalb der Kommunikation

Aufgrund seiner hohen Flexibilität erlaubt der MCU Takt-Modes, welche die Low-Power-Anforderungen unterstützen. Es ist möglich, das Timing entsprechend zu optimieren. Dazu soll zwischen genauem und ungenauem Timing unterschieden werden.

1. Genaues (quarzbasierendes) Receive/Transmit-Timing: Jede Receive/Transmit-Operation kann in der Zeit präzise ausgeführt werden, zusätzlich lässt sich das Empfangsfenster so schmal wie möglich auslegen, um Strom zu sparen.

a) MCU mit 32,768-kHz-Quarz, Modem im Off-Mode

Die MCU läuft mit sehr geringer Geschwindigkeit, während sie auf die nächste Kommunikation wartet. Während zeitkritischer Perioden schaltet die MCU auf die frequenzstarre Schleife (Locked Loop); der Bus-Takt liegt typisch bei 8 MHz. Der Nachteil dieser Lösung ist, dass der Übergang von Off- zu Idle-Mode zwischen 10 und 25 ms benötigt, je nach Start-up-Zeit des Spannungsreglers und des Taktoszillators. Weiterhin geht der Inhalt von Modem-RAM verloren, ebenso der GPIO-Status, daher müssen alle Konfigurationen jedes Mal wiederholt werden. Zwei Quarze (16 MHz für das Modem und 32,768 kHz für die MCU) werden benötigt. Gesamtverbrauch: 200 nA (Modem) + 95 µA (MCU)

b) MCU mit 32,768-kHz-Quarz, Modem im Hibernate-Mode

Das Verhalten ist wie beim a)-Szenario. Die Übergangszeit zum Idle-Mode ist mit 8 bis 20 ms etwas kürzer. Ein Vorteil dieser Arbeitsweise ist aber, dass der Inhalt des RAMs und der GPIO-Status erhalten blei-

Mode	MCU-Takt	Genauigkeit	Modem-Mode	Stromverbrauch
1 a	Xtal 32768 Hz	100 ppm	Off	95 µA
1 b	Xtal 32768 Hz	100 ppm	Hibernate	96 µA
1 c	getaktet vom Modem	20 ppm	Doze	36 µA
2 a	RTI-Takt	30%	Off	1,2 µA
2 b	RTI-Takt	30%	Hibernate	2 µA
2 c	intern 4 kHz	2%	Off	150 µA
2 d	intern 4 kHz	2%	Hibernate	151 µA

Tabelle 2: Betriebsarten, Taktfrequenzen, Genauigkeiten und Stromaufnahmen

ben. Benötigt werden die selben Quarze wie oben. Gesamtverbrauch: 1 µA (Modem) + 95 µA (MCU)

c) MCU getaktet vom internen Oszillator oder extern (via CLKO vom Modem), Modem in Doze-Mode

Die MCU stellt den internen Oszillator bereit, und wenn sie angewiesen wird, den Doze-Mode zu verlassen, so kann sie das sehr schnell. Die MCU setzt das Modem zurück und konfiguriert es. Das Modem kann auch den Referenztakt im Doze-Mode führen. Die MCU programmiert CLKO entsprechend der gewünschten Frequenz und wartet, bis sich die Verhältnisse an CLKO stabilisieren. Sobald dies der Fall ist, schaltet die MCU ihren Takt zur externen Quelle. Ein präzises Timing auch langer Zeitverzögerungen erfolgt im Modem durch einen zeitlich festen Ausstieg aus dem Doze-Mode, während die MCU zur selben Zeit im Schlafzustand ist.

In dieser Konfiguration kann der Designer voll vom automatischen Initialisieren zeitgetriggerter Ereignisse profitieren, wie:

- Aktivierung des Empfangs-Mode
- Aktivierung des Sende-Mode
- Aktivierung des Idle-Mode

Zudem werden die empfangenen Frames während des Eingangs mit einem Zeitstempel versehen, sodass ein präzises Timing zwischen verschiedenen Ereignissen möglich ist. Der Mikrocontroller kann in Low-Power-Modes verbleiben. Dieses Szenario hat eine leicht höhere Leistungsaufnahme gegenüber einem separaten Quarz für die MCU, jedoch genügt hier eben nur ein einziger Quarz.

Ungenaues Paket-Timing

a) MCU Timing wird gesteuert vom internen Oszillator (RTI) und Modem läuft im Off-Mode

Nachteil bei dieser Variante ist, dass das Kommunikations-Timing nicht sehr genau erfolgt, da der interne Oszillator kein präzises Timing erlaubt (nur zu 30%). Daher müssen die Empfangsfenster breiter gemacht werden, was die Leistungsaufnahme negativ beeinträchtigt. Nur ein Quarz wird benötigt. Gesamtstromaufnahme: 200 nA (Modem) + ~1 µA (MCU Sleep-Mode + RTI enabled)

b) MCU Timing wird gesteuert vom internen Oszillator (RTI) und Modem läuft im Hibernate-Mode

Es ergeben sich die gleichen Verhältnisse wie unter a). Ein Unterschied ist der kürzere Übergang zum Idle-Mode. Auch die Re-Initialisierung des Modems ist nicht nötig. Gesamtstromaufnahme: 1 µA (Modem) + 1 µA (MCU Sleep-Mode + RTI enabled)

c) MCU wird vom internen Oszillator mit niedrigster Frequenz (4 kHz) getaktet, Modem im Off-Mode

Der MCU-Bus-Takt wird von einer internen Clock-Referenz getrieben. Er ist zu 2% genau (nach Abgleich), was ausreichend ist für viele Applikationen. Nur ein Quarz wird benötigt. Gesamtstromaufnahme: 200 nA (Modem) + 150 µA (MCU mit 4 kHz)

d) MCU wird vom internen Oszillator mit niedrigster Frequenz (4 kHz) getaktet, Modem im Hibernate-Mode

Es ergeben sich die gleichen Verhältnisse wie unter c). Gesamtstromaufnahme: 1 µA (Modem) + 150 µA (MCU mit 4 kHz). Tabelle 2 stellt Betriebsarten, Taktfrequenzen, Genauigkeiten und Stromaufnahmen zusammen

Sende- und Empfangsereignisse

Wie die Aufstellung der Versorgungsströme bei Senden und Empfang mit dem Modem zeigt, spielt es für den Stromverbrauch eine große Rolle, ob empfangen oder gesendet wird. Daher ist es erforderlich, die Länge der Datenpakete sorgfältig zu optimieren und die Kommunikationsraten mit Überlegung festzulegen. Bild 1 informiert zur Abhängigkeit der Batterielebensdauer von der Häufigkeit der Sendefenster, Bild 2 zeigt die Paketstruktur des Systems, und Bild 3 informiert zur Abhängigkeit der Batterielebensdauer von der Breite der Datenpakete. Ein guter Kompromiss sind 30 Packets pro Sekunde, von hier nach rechts sind Häufigkeit und Lebensdauer indirekt proportional. Auch die Länge der Datenpakete beeinflusst die Batterielebensdauer signifikant. Daher werden nur Rohdaten vom triaxialen Accelerometer gesendet (Näheres im Kapitel 5.3 des ZSTAR Reference Manuals).

Das Standard-IEEE 802.15.4-Compliant-Paket besteht aus:
4 Bytes Preamble
1 Byte SFD (Start of Frame Delimiter)
1 Byte FLI (Frame Length Indicator)
2 Bytes FCS (Frame Check Sequence) plus die Payload Data (125 Bytes max.).

Dies resultiert in einem Overhead für ein Frame von 8 Bytes oder 8 x 32 µs = 256 µs, und die maximale Payload-Übertragungszeit ist 125 x 32 µs = 4000 µs. Die Sendezeit für ein Paket beträgt dann (in µs) 256 + (Payload Bytes x 32). Ein Standard-SMAC-Paket für unser Sensor-Board hat 17 Bytes (inklusive dem 2-Byte-Packet-IEEE 802.15.4-Kontrollfeld). Es gibt auch sogenannte Warm-up- (Übergang von Idle zu Sen-

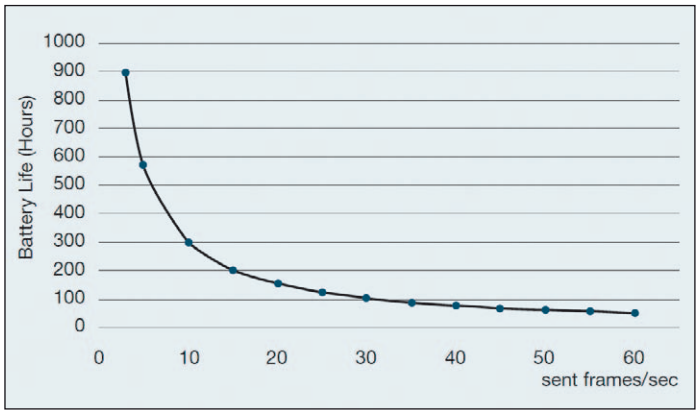

Bild 1: Abhängigkeit der Batterielebensdauer von der Breite der Sendefenster

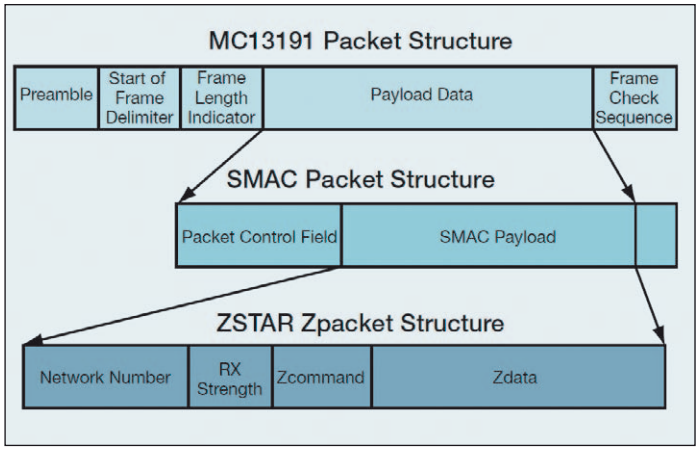

Bild 2: Paketstruktur der Komponenten des Systems

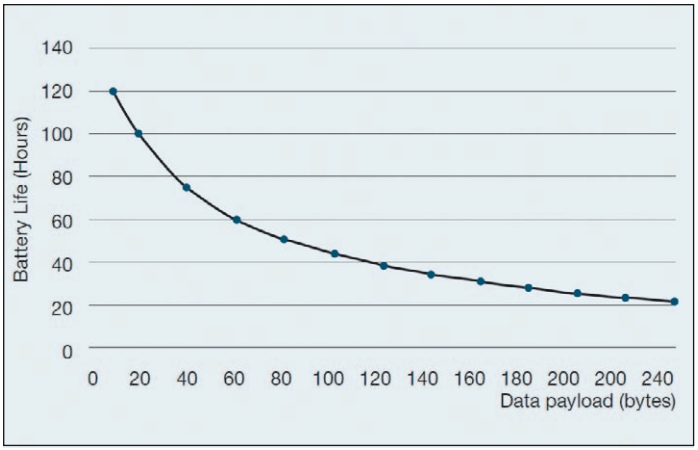

Bild 3: Abhängigkeit der Batterielebensdauer von der Breite der Datenpakete

den) und Warm-down-Perioden (Übergang von Senden zu Idle), welche 144 µs bzw. 10 µs in Anspruch nehmen. Es ist erkenntlich, dass die Paketlängen so klein wie möglich sein sollten. Bild 3 ist allerdings rein theoretisch, da noch eine andere physikalische Begrenzung besteht: Die Lithium-Batterie kann nur kurzzeitig einen konstanten Strom ab einer bestimmten Größe liefern, daher sinkt die Batteriespannung, wenn die Frames zu lang sind (länger als ~50 bis 100 Bytes). Im ZSTAR-Design kam daher ein Tantalkondensator als Speicher 470 µF zum Einsatz. (Berechnung der Kapazität s. Original).

Wie alles zusammenarbeitet

Das Sensor-Board wurde so entwickelt, dass es sehr klein und kostengünstig ist, daher nutzten wir nur einen Quarz am Modem. Weiter benötigten wir das genaue Kommunikations-Timing, um das Modem im Doze-Mode dafür einsetzen zu können, und die MCU erhielt den internen Takt während es aktiv ist, andernfalls läuft die MCU im Stop3-Mode. Es folgt die Beschreibung der Kommunikation zwischen Sensor-Board und USB-Stick anhand der Übertragung eines Datenpakets:

1. Am Beginn der Priode weckt das Modem die MCU aus dem Stop3-Mode auf. Die MCU wiederum weckt das Triax aus dem Schlafmodus (über das SLEEP Pin von Triax). Dann werden die Triax-Ausgänge stabilisiert, bevor eine Messung erfolgen kann. Während dieser Zeit geht das Modem erneut zeitbegrenzt in den Doze-Mode, und auch die MCU kehrt zum Stop3-Mode zurück.

2. Wenn Triax für eine Messung bereit ist, weckt das Modem die MCU nochmals und dieses misst mit dem Triax. Nach Erstellung der Daten durch die MCU, wird der Triax zurück in den Schlafmodus versetzt, das Datenpaket wird gebildet und in den RAM-Puffer des Modems geschoben.

3. Das Modem sendet das Paket über Funk.

4. Um die Verbindung (zwischen Sensor-Board und USB) zu prüfen, wird die Rückmeldung vom USB-Stick erwartet. D der USB-Stick hierzu jedoch eine gewisse zeit benötigt, versetzt die MCU das Modem wieder für kurze Zeit in den Doze-Mode.

5. Nach Ablauf dieser kurzen Zeit erwacht die MCU und die Rückmeldung vom USB-Stick kann im dafür spezifizierten Zeitfenster empfangen werden. Dann wird das Modem in den Doze-Mode gebracht, und zwar bis zum Ende des Zyklus´, und die MCU geht in den Stop3-Mode.

6. Zwischen zwei aufeinander folgenden Packets bleibt die MCU im Stop3-Mode, Triax wird in den Sleep-Mode geschaltet und das Modem ist im Doze-Mode, solange seine Zeitbasis die Zeit bis zum nächsten Packet misst.

Bild 4 skizziert diesen Ablauf bei der Übermittlung eines Datenpakets. Bei der genaueren Einschätzung dieser Kommunikationsmethode konnten wir beobachten, dass der Empfang nach jeder Aussendung den Stromverbrauch beachtlich in die Höhe trieb. Da dies nicht erwünscht war, wurde das Empfangsfenster für die Rückmeldung nur nach jeder achten Rückmeldemöglichkeit geöffnet. In allen anderen Fällen geht das Sensor-Board in den Doze-Mode, sobald die Übertragung stattgefunden hat.

Zusammenfassung

Das ursprüngliche Ziel des ZSTAR-Projekts, in Echtzeit Daten bei limitiertem Leistungsverbrauch zu übertragen, wurde erreicht. Alle Aspekte des Systemdesigns (sowohl Hardware-Architektur als auch Software-Implementierung) wurden erforscht und für die höchste Leistungsfähigkeit optimiert. XYZ-Samples vom triaxialen Accelerometer werden zusammen mit Informationen zur internen Temperatur und zur Höhe der Batteriespannung 30 mal pro Sekunde übertragen. Es findet eine kleine Lithium-CR2032-Batterie Anwendung, welche für ~100 Stunden Sendebetrieb beansprucht werden kann. Der Mittelwert des Stromverbrauchs dieses Systems lässt sich mit 2,5 mA veranschlagen, wobei Messungen dies bestätigen.

Alles in Allem zeigt das ZSTAR-Design eine kosteneffektive Implementierung eines Vertreters der MC1319x-Modem-Familie mit einem kostengünstigen Low-Pin-8-Bit-Mikrocontroller aus der MC9S08QG-Familie zum Zwecke der drahtlosen Anbindung des dreiachsigen Accelerometers MMA7260QT. ◄

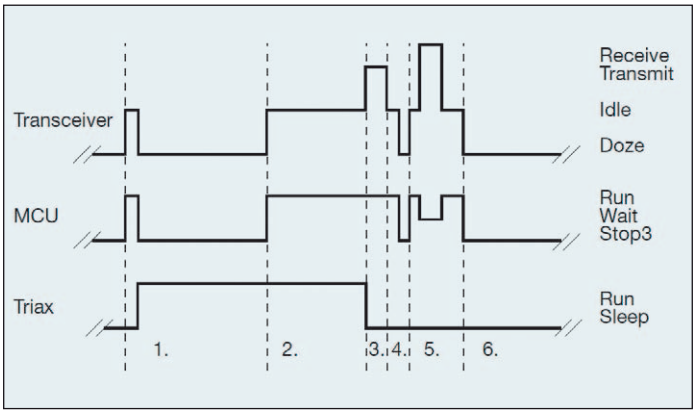

Bild 4: Pegeldiagramm zum Ablauf der Übermittlung eines Datenpakets