Fahrradcomputer mit GPS, Datenlogger und drahtloser Datenübertragung

Martin Glunz

11. Februar 2012

1 Motivation

Um Radtouren zu protokollieren verwendete ich in den vergangenen Jahren einen kommerziellen GPS-Datenlogger. Das Gerät ist mechanisch sehr robust und zum Einsatz am Fahrrad geeignet, auch für Touren auf holprigen Wegen.

Allerdings hat das Gerät folgende Nachteile:

- Akkus müssen regelmäßig geladen werden
- Die Daten müssen manuell ausgelesen werden
- Zum Start einer Tour müssen mehrere Werte manuell am Gerät rückgesetzt werden
- In Umgebungen mit schlechtem Empfang (Wald) leidet die Genauigkeit
- Ein Logbuch (wieviele km und Höhenmeter zu welcher Tour) muß manuell geführt werden

Als Ersatz für dieses Gerät habe ich das zu dem Wettbewerb gestellte Oryx-Board genutzt, um den Prototypen eines Fahrradcomputer aufzubauen, der folgende Eigenschaften hat:

- Akku wird laufend durch Solarzellen vollgeladen
- Die Übertragung der Daten erfolgt automatisch bei der Rückkehr von einer Tour auf den Heim-Server
- Es ist keine manuelle Bedienung des Fahrradcomputers zum Start oder Ende einer Tour notwendig
- Der Radimpulsgeber eines konventionellen Fahrradtachos wird zur genaueren Wegerfassung benutzt
- Ein barometrischer Höhenmesser wird zur genaueren Erfassung der gefahrenen Höhenmeter benutzt
- Das Logbuch wird vom Heim-Server automatisch geführt

2 Bestandteile des Systems

Das System besteht aus dem mobilen Teil (dem Fahrradcomputer) und einer Basisstation mit Verbindung über Ethernet zum Heim-Server.

Da für die Basisstation eine dauerhafte Energieversorgung aus dem Stromnetz möglich ist, ist die Basisstation nicht Teil der Entwicklung dieser stromsparenden Anwendung, ebenso nicht der Heim-Server. Diese Komponenten sind zur geplanten Nutzung des Fahrradcomputers und zur Durchführung der Inbetriebnahme des Prototypen notwendig.

Der Fahrradcomputer enthält folgende wesentliche Bestandteile:

- Solarzellen als Energiequelle (extern)
- Laderegler und Akku-Schutzschaltung (LTC4071, Bestandteil des Oryx-Boards)
- Li-Ion-Akku (Bestandteil des Oryx-Board, optional extern mit größerer Kapazität)
- Der Microcontroller LPC11U14 (auf dem Oryx-Board)
- Das Display (Sharp Memory LCD auf dem Oryx-Board)
- Eine Echtzeituhr (PCF8523 auf dem Oryx-Board)
- SPI-Flash als Datenspeicher (S25L004K auf dem Oryx-Board, Nutzung ist optional)
- Ein GPS-Empfänger (extern, ein Modul mit MediaTek MT3318)
- Der Drucksensor (extern, BMP085)
- Ein 2.4GHz Transceiver (extern, ein Modul mit dem nRF24L01+)
- Ein Port zum Anschluß des Radimpulsgebers

Auf dem Oryx-Board sind noch weitere Bausteine vorhanden, diese wurden für den Fahrradcomputer nicht benötigt. Die externen Module wurden auf einem kleinen Erweiterungsboard montiert, das auf den dafür vorgehenen Erweiterungssteckern des Oryx-Board angebracht wurde.

Der 2.4GHz-Transceiver und der Drucksensor bringen für die Anwendung geeignete Low-Power-Modi mit, so daß für diese beiden Teile keine besonderen Vorkehrungen getroffen werden mußten, um einen möglichst niedrigen Stromverbrauch zu realisieren.

Das GPS-Modul bietet keine Möglichkeit eines stromsparenden Betriebs, so daß hier eine Möglichkeit vorgesehen wurde, die Stromversorgung des Moduls abzuschalten. Das auf dem Oryx-Board vorhandene Potentiometer zur Stimulation des A/D-Wandlers wird nicht benötigt, somit kann dessen Power-Switch für die Abschaltung des GPS-Moduls verwendet werden.

Die externen Module verwenden folgende Schnittstellen des Microcontrollers:

- Drucksensor: I2C-Bus
- 2.4GHz-Transceiver: SPI1, geteilt mit dem SPI-Flash
- GPS-Modul: UART
- Radimpulsgeber: Port PIO0_16 / Wakeup

3 Funktion des Systems

3.1 Standby

Im Ruhezustand wird der Akku aus den Solarzellen geladen, der Laderegler LTC4071 überwacht den Ladezustand und verhindert eine Überladung des Akkus.

Der Microcontroller ist im Schlafmodus, alle Peripherien sind inaktiv. Die RTC gibt einen sekündlichen Interrupt aus, der den Controller regelmäßig aufweckt.

Im Ruhezustand zeigt das Display die Uhrzeit und das Datum, der Displayinhalt wird nur einmal pro Minute aktualisiert, um eine minimale Stromaufnahme zu erreichen.

3.2 Start einer Tour

Sobald das Fahrrad in Bewegung gesetzt wird, weckt ein Impuls des Radimpulsgebers den Microcontroller auf. Der erste Radimpuls löst eine Initialisierung der Tourdaten aus (alle Werte werden automatisch auf Null gesetzt). Der Drucksensor wird aktiviert, das GPS-Modul wird eingeschaltet.

3.3 Initialer GPS-Empfang

Die ersten gültigen Daten, die von dem GPS-Modul geliefert werden, werden genutzt, um die Echtzeituhr auf die genaue Uhrzeit zu stellen und die Anzeige des Höhenmeters zu korrigieren (die von dem Drucksensor gelieferte Höhenmeterinformation ist vom momentanen Umgebungsluftdruck abhängig).

Es ist auch eine automatische Kalibrierung des Radumfangs anhand der GPS-Daten geplant (die aus den GPS-Positionsdaten berechnete zurückgelegte Strecke wird mit der aus dem Impulsen des Radgebers akkumulierten Strecke über einige 100m möglichst geradlinig zurückgelegte Enfernung verglichen und ggf. ein Korrekturfaktor für den Radumfang bestimmt).

Nach erfolgreichem Erstempfang wird die Stromversorgung des GPS-Moduls abgeschaltet, um Strom zu sparen.

3.4 Während der Tour

Jeder neue Radimpuls setzt ein Timeout zurück. Aus der Frequenz der Radimpulse wird die aktuelle Geschwindigkeit berechnet und am Display angezeit, die Radimpulse werden akkumuliert um die zurückgelegte Strecke zu bestimmen. Aus den Daten des Drucksensors wird die zurückgelegte Höhendifferenz bestimmt und getrennt nach Auf- und Abstieg akkumuliert.

Einmal pro Minute wird das GPS-Modul eingeschaltet. Bei gutem Empfang liefert das Modul nach ca. 2 Sekunden gültige Positionswerte. Die empfangene Position wird gespeichert. Im Laufe der Tour wird also minütlich die Position gespeichert. Das Zeitintervall ist willkürlich gewählt und kann dem tatsächlichen im Betrieb ermittelten Bedarf angepaßt werden.

Am Display werden folgende Informationen angezeigt:

- Uhrzeit
- Geschwindigkeit
- Tages-Kilometer
- Tour-Timer
- Höhe über NN
- Höhenmeter Aufstieg
- Höhenmeter Abstieg
- GPS-Status

Die Aktualisierung des Displays erfolgt jetzt sekündlich.

3.5 Unterbrechung der Tour

Wenn über einen Zeitraum von 2 Minuten keine Radimpulse einlaufen, versucht der Fahrradcomputer eine Funkverbindung zur Basisstation aufzubauen. Bei einer Unterbrechung der Tour ist die Basisstation nicht in Reichweite, der Verbindungsaufbau schlägt fehl. Der Controller geht in den Schlafmodus und alle Module werden abgeschaltet. Der Akku wird fortlaufend aus den Solarzellen geladen.

3.6 Fortsetztung der Tour

Ein neuer Radimpuls weckt den Controller wieder auf. Da zuvor keine Verbindung zur Basis aufgenommen werden konnte, werden die Tourdaten nicht rückgesetzt, sondern weiter akkumuliert.

3.7 Ende der Tour

enn über einen Zeitraum von 2 Minuten keine Radimpulse einlaufen, versucht der Fahrradcomputer eine Funkverbindung zur Basisstation aufzubauen. Am Ende der Tour ist die Basisstation in Reichweite, der Verbindungsaufbau gelingt. Es werden die akkumulierten Tourdaten und die minütlich gespeicherten Positionsdaten zur Basisstation übertragen. Nachdem alle Daten erfolgreich übertragen wurden erfolgt der Übergang des Controllers in den Ruhezustand. Falls der Verbindungsaufbau fehlschlägt, oder die Daten nicht vollständig übertragen wurden, erfolgt alle 5 Minuten ein erneuter Versuch des Verbindungsaufbaus.

4 Ergebnisse

Der Prototyp wurde unter Verwendung des Oryx-Board und zusätzlicher Module aufgebaut. Die Firmware für den Microcontroller wurde unter Verwendung des zum Oryx-Board gelieferten Treiberpaketes implementiert. Der Aufbau der Hardware gestaltete sich mit dem Oryx-Bord und den zusätzlichen Modulen recht einfach, allerdings ist der Aufbau nicht zum dauerhaften praktischen Einsatz am Fahrrad geeignet. Mit der Entwicklungsumgebung LPCXpresso gestaltete sich die Entwicklung der Firmware für den Fahrradcomputer ebenfalls recht komfortabel. Positiv vermerkt wurde die Verfügbarkeit einer Linux-Version von LPCXpresso, die reibungslos funktionierte. Zur Implementierung der Firmware diente als Gerüst eines der zum Oryx-Board verfügbaren Beispiele, für den nRF24L01 wurde ein frei im Internet verfügbarer Treiber verwendet. Weitere Quellen sind ein Beispielcode für den Drucksensor und ein NMEA-Decoder:

- nRF24L01: Copyright S. Brennen Ball, 2006-2007
- BMP085: BMP085 Test Code by Jim Lindblom
- NMEA: Ausgangspunkt ist ein Stück Code aus dem Projekt "ArduPilot"

Im Ruhezustand wurde eine Stromaufnahme des gesamten Fahrradcomputers von ca. $4\mu A$ gemessen, im Betrieb ist die Stromaufnahme insbesondere bestimmt durch das GPS-Modul deutlich höher. Aus diesem Grund wird das GPS-Modul nur einmal pro Minute für ca. 2...3 Sekunden eingeschaltet. Unter dem Verzicht auf eine genaue Geschwindigkeitsanzeige kann eine deutliche Reduzierung der Stromaufnahme erreicht werden, da dann der Microcontroller für die Zeit zwischen den Radimpulsen in den Schlafmodus gehen kann. Für eine genaue Geschwindigkeitsmessung ist allerdings eine höher aufgelöste Zeitmessung zwischen den Radimpulsen notwendig, hierzu kann z.B. ein Timer oder der SysTick des Cortex-M0 verwendet werden, allerdings kann dann der Controller nicht in den sparsamsten Schlafmodus gehen.

Die Implementierung der Firmware ist noch nicht vollständig, es fehlen folgende Funktionen:

- minütliche Speicherung der Position
- automatische Kalibrierung des Radumfangs

Folgende Funktionen sind vorhanden und funktionieren im Test:

- Ruhezustand
- Geschwindigkeitsmessung über Radpulse
- Höhenmetererfassung über Drucksensor

- Tourdatenerfassung (Tages-km, Timer)
- Erkennung Unterbrechung oder Ende der Tour
- Datenübertragung zur Basisstation
- GPS-Empfang (Erstempfang und minütliches Update während der Tour)

Ein echter Praxistest (also eine Radtour mit dem Fahrradcomputer) wurde aus zwei Gründen bis jetzt noch nicht durchgeführt: Es ist ziemlich kalt draussen, und es besteht die Gefahr, daß der Prototyp bei einer Probefahrt im Gelände durch die Erschütterungen beschädigt oder zerstört wird.

Die Tests wurden mit simulierten Radpulsen und Bewegung des Fahrradcomputers "zu Fuß" durchgeführt.

Nachtrag: Am 11.2.2012 wurde eine Testfahrt mit positivem Ergebnis durchgeführt: alle bisher implementierten Funktionen funktionierten auch unter echten Einsatzbedingungen. Die Höhendifferenz und die Strecklenlänge der Testfahrt wurden von dem Prototype korrekt ermittelt und an die Basisstation gesendet.

Die Auswertung eines Brustgurt-Pulssensors wurde in diesem Prototypen nicht realisiert, da zum Zeitpunkt des Aufbaus des Prototypen keine geeignete Hardware verfügbar war.

5 Basisstation

Um die vom Fahrradcomputer erfaßten Tour- und Trackdaten empfangen und auf den Heim-Server zu übertragen, wurde unter Verwendung eines Evaluation-Boards ein Prototyp der Basisstation aufgebaut.

Die Basisstation bietet folgende Schnittstellen:

- 2.4GHz-Transceiver mit nRF24L01+
- ein OLED-Display
- einen 100MBit/s Ethernet-Port

Die Basisstation wartet ständig auf den Verbindungsaufbau vom Fahrradcomputer. Sobald eine Verbindung zustande kommt, empfängt die Basisstation Paketweise die Daten und bestätigt dem Fahrradcomputer den Empfang.

Auf dem Display wird nach dem erfolgreichen Empfang eine Zusammenfassung der Tourdaten angezeigt. Über das Ethernet wird eine Verbindung zum Heimserver aufgebaut und das Datenpaket abgesetzt. Auf dem Heimserver muß dazu eine passende Anwendung dauerhaft laufen. Alternativ könnten die Tourdaten auch auf einer MicroSD-Karte gespeichert werden, das verwendete Evaluation-Modul hat einen solchen Steckplatz.

Für den Controller des Evaluation-Boards ist eine Portierung des freien TCP/IP-Stacks LWIP verfügbar, ebenso freie Dateisystemtreiber für die SD-Card und ein Treiber für das OLED-Display, so daß sich die Implementierung der Firmware für die Basisstation ebenfalls recht einfach gestaltete. Hier wurde zur Entwicklung die freie IDE Eclipse mit einer Makefile-basierten Projektstruktur gewählt. Als Compiler kommt hier eine freie Portierung des GNU-Compilers (gcc) für den ARM Cortex-M3 zum Einsatz. Diese Umgebung ist problemlos unter Linux einsetzbar und auch parallel zur Entwicklungsumgung LPCXpresso verwendbar, ohne gegenseitige Beeinflussung der beiden IDEs (LPCXpresso basiert auf Eclipse mit einer anderen Portierung des GNU-Compilers).

6 Ausblick

Sicherlich der größte Aufwand bei der Umsetzung des Prototypen in einen tatsächlich praktisch nutzbaren Fahrradcomputer dürfte die Erstellung einer robusten und kompakten Hardware sein. Die Firmware kann mit dem bestehenden Prototypen weiter entwickelt und getestet werden. Mit wenig Aufwand kann an dem Prototypen ein Empfänger für einen Brustgurt-Pulssensor nachgerüstet werden und die dazu notwendige Treiber-Firmware entwickelt werden.