HOCHSCHULE KONSTANZ TECHNIK, WIRTSCHAFT UND GESTALTUNG
UNIVERSITY OF APPLIED SCIENCES

Teamprojekt Modulare Plattform für intelligente IoT-Sensoren

Konstanz den, 21. Februar 2017

Team:	Fabian Gendusa & Thomas Gnädig
Zeitraum:	WS 16/17
Fachbetreuer:	I. Schoppa
Fakultät:	Informatik
Studiengang:	Angewandte Informatik

1

Vorwort

Unnötig?

2

Das Projekt

2.1 Projekt Beschreibung

Im Rahmen des Studiums ist ein mit 12 ECTS gewichtetes Teamprojekt vorgesehen. Das Projekt IoT Sensor kombiniert drei selbst erstellte Platinen zu einem funktionierenden System.

Auf der Hauptplatine wird ein Mikrocontroller der Art MSP470FR und drei PMOD Schnittstellen angebracht. Über die PMOD Stecker soll mit den anderen Board kommuniziert werden.

Die Kommunikationsplatine wird mit USB am PC verbunden und mithilfe des FT231X auf UART umgewandelt. Das umgewandelte Signal wird dann über einen PMOD Stecker via UART an das Main-Board weitergeleitet. auf der Platine befindet sich auch eine galvanische Trennung, welche den Stromkreis des Computers vom Stromkreis des Mikrocontrollers trennt. Hierfür ist der ADUM1401 verantwortlich, welcher das Signal über Lichtwellen weiter gibt.

Die Anzeige Platine stellt eine 7 Segmentanzeige zur Verfügung, welche die Visualisierung von Beispielsweise Sensorwerten übernimmt. hierzu Dient der AS1108 welcher die Übersetzung einer Zahl auf die Sieben Segmentanzeige übernimmt. Die Zahlen werden über SPI vom Mainboard erhalten.

Die Sensorplatine, nicht teil des Projekts, wird auch über SPI angesprochen und liefert Messwerte an das Main-Board. Kann bei bedarf bereits Fertig gekauft werden.

2.2 Arbeitsschritte

2.2.1 Bauteilsuche

Nachdem die Vorgabe vollständig ist, kann mit der Bauteilsuche begonnen werden. Einige Teile waren auch bereits vorgegeben. Kriterien beim auswählen der Bauteile waren:

- die Funktion, Angaben müssen erfüllt werden
- die Verfügbarkeit, Teile müssen einzeln bestellbar und auch in Zukunft noch Angeboten sein
- der Preis

2.2.2 Platinen Design

Nachdem alle Teile gefunden sind und auch alle Datenblätter vorliegen, kann mit dem Design der Platine begonnen werden. hierzu haben wir das Tool Pulsonix verwendet, nachdem der Schaltplan hierin entworfen war, konnte das Schematic erstellt werden, die größte Hürde hierbei war das einstellen des korrekten Rasters für Bauteile und Vias. zu beachten war, dass manche Bauteile möglichst nah an anderen Bauteilen angebracht werden, so zb beim FT231X, wo ein differentielles Signal anlag. Auch zu beachten war an den PMOD Steckern sollten zwei Kondensatoren angebracht werden um Störsignale welche beispielsweise beim anstecken entstehen raus zu filtern.

2.2.3 Hardware bestücken

Die Hardware wurde bei `www.digikey.com` bestellt und war innerhalb kurzer zeit angeliefert. Um die Platinen zu bestellen, mussten zunächst Gerberfiles erstellt werden, welche dann an Q-print electronic GmbH geschickt wurde. Zum bestücken der Platine war Teamarbeit gefragt, während einer das Bauteil möglichst exakt auf den kleinen Pads positioniert muss der Zweite das erste Beinchen fest löten. die wohl größte Herausforderung hierbei war das befestigen des FT321X auf der Kommunikationsplatine. Nachdem diese Herausforderung gemeistert war, stellten die anderen Bauteile keine größeren Probleme mehr dar. Als Änderung für neue Platinen wäre bei der Anzeigenplatine ein Größerer *RSET* Widerstand einzubauen, da der eingebaute 10 $k\Omega$ etwas zu klein dimensioniert ist. wenigstens 11,35 $k\Omega$ siehe Datenblatt AS1108 Seite 9.

2.2.4 Softwareentwicklung

Es musste nicht nur ein Treiber für den MSP geschrieben werden um über UART und SPI mit unseren Boards zu kommunizieren, sondern auch ein Treiber, welcher die Kommunikation zwischen PC und Kommunikationsplatine übernimmt. Die genauere beschreibung folgt im nächsten Kapitel.

3

Entwickelte-Software

3.1 Python

Für die Programmierung, um auf das Kommunikationsboard zugreifen zu können, haben wir uns für Python entschieden, da es hier relativ einfache und schnelle wege gibt um c librarys einzubinden und zu benutzen. Wir greifen hier auf die vom Hersteller bereitgestellte library "ftd2xxßurück.

3.1.1 MSP430connect.py

dieses Script dient zur simplen Übertragung eines durch CCS erstellten Programms über die Kommunikationsplatine auf den Microcontroller. Dieses Script kann sowohl mit Parameter -p [Dateipfad] ausgeführt werden, wobei dann lediglich das Programm heruntergeladen wird und anschließend das Programm beendet, oder ohne Parameter um es als CLI zu benutzen, mit dem Parameter h kann eine kleine Hilfe zum Programm angezeigt werden. Wie die Das Programm zum herunterladen auf den Microkontroller mit CCS erstellt werden muss, wird auch hier beschrieben. Das Script verwendet unseren MSP430.py Treiber, welcher einfache Methoden zum öffnen eines Devices, lesen und schreiben sowie zurücksetzen dieses Devices anbietet. Diese Methoden können für weitere Programme genutzt werden.

3.2 C

3.2.1 Einstiegs Codes

da wir uns das erste mal mit CCS und dem MSP430FR befassen, haben wir uns zunächst ein paar simple Anwendungen überlegt um ein Gefühl für die Umgebung zu bekommen.Hierzu haben wir vor allem die auf dem Experimentierboard angebrachten LEDs als ausgabe verwendet um möglichst schnell und einfach Feedback zu erhalten. Eine einfache Anwendung war das erstellen eines Lauflichts, welches über Tastendruck zusätzlich beeinflusst werden konnte (CD\Code\runled.c). Auch den auf dem Board eingebauten Wärmesensor haben wir ausgelesen und dessen wert direkt an die LEDs angelegt (CD\Code\adc.c). Dass sich die Zahl beim auflegen des Fingers auf den Sensor verändert hat war ein kleines aber feines Erfolgserlebnis.

3.2.2 UART

Für die Kommunikation von UART haben wir einen kleinen Treiber geschrieben, welcher zum initialisieren von maximal 2 UART schnittstellen verwendet werden kann, eine Funktion um eine Zeichenkette zu lesen oder schreiben wird auch bereit gestellt (CD\Code\UART\uart.clh).

3.2.3 SPI

Auch für die beiden SPI Schnittstellen wurde ein Treiber geschrieben, dieser ist ebenfalls auf der CD vorhanden(CD\Code\SPI\spi.clh). ein Beispielcode zum ansprechen der Siebensegmentanzeige befindet sich ebenfalls in diesem Ordner, hier ist es uns bis heute leider nicht gelungen, den Treiberbaustein korrekt einzustellen, die Funktionalität der SPI Schnittstelle ist aber bereits geprüft und funktioniert wie in diesem Beispielcode beschrieben.

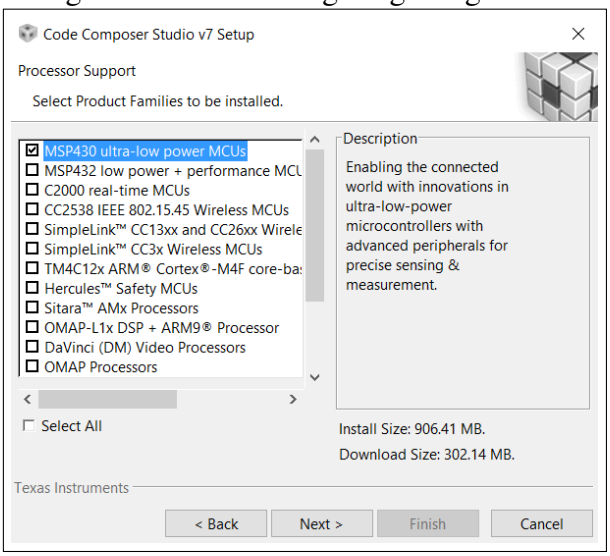
4

Hilfsmittel/Arbeitsumgebung

zum erstellen des Schaltplans und der Gerberfiles haben wir das auf den Laborrechnern bereitgestellte Tool Pulsonix verwendet.
Die Python Programmierung haben wir im Notepad++ durchgeführt.
Um den MSP430FR zu beschreiben griffen wir auf das vom Hersteller bereitgestellte Tool CCS zurück.

4.0.1 Installation und anwendung des CCS

Die aktuellste Version des CCS kann unter folgender URL bei ti direkt heruntergeladen werden `http://processors.wiki.ti.com/index.php/Download_CCS`.
Nach dem Ausführen der css_setup.exe sollte im Schritt "processor support"wenigstens die Familie MSP430 wie in folgendem Screenshot gezeigt ausgewählt werden.



Nach der Installation des Programms kann ein Workspace angelegt oder ein bereits

bestehender genutzt werden. beim erstellen eines Neuen Projekts sollte darauf geachtet werden, dass auch der richtige Microcontroller ausgewählt wurde, folgender Screenshot zeigt die korrekte Einstellung

durch diese Auswahl werden bereits die richtigen librarys eingebunden. hier kann auch ein kleines Beispielprogramm erstellt werden, welches eine LED blinken lässt. Die weitere Programmierung ist wie in herkömmlichen Entwicklungsumgebungen. Eine nette und hilfreiche Erweiterung ist der Debugmodus, welcher uns schon oft weiter geholfen wird, hier kann das Programm auf dem MSP430 direkt laufengelassen, angehalten und fortgesetzt werden.
Ein bereits fertiger Workspace-Ordner mit allen hier aufgelisteten Programme ist auch auf der CD zu finden.