Verslag IRCommander

Frank Wibbelink

19 april 2015

Samenvatting

Inhoudsopgave

1 Opdracht

Dit verslag behandelt de eindopdracht van CAO2. Het doel van de opdracht is kennismaken met microcontrollers en digitale signaalverwerking. Ik heb deze opdracht ingevuld met het uitlezen van een afstandsbediening op een *BeagleBone Black* (BBB).

2 Achtergrond

2.1 PRU

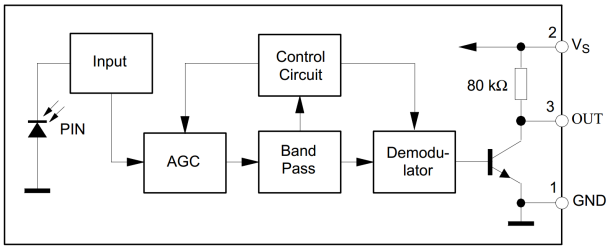
De PRU (2 aanwezig op de BBB) is een realtimeprocessor die los van de CPU (en daarmee de kernel) draait op 200 MHz. Deze processor doet niet aan pipelining en de instructieset bestaat uit load/store, branching, bit shifts en optellen/aftrekken, waarvan de meeste 1 klokcyclus kosten om uit te voeren. Er zijn 32 registers beschikbaar (geen ingebouwde stack of window) en een tabel met veelgebruikte offsets en verder vindt alle communicatie via I^2C plaats. Dit betekent voor de PRU-code dat er specifieke geheugenadressen –uit de voorgenoemde tabel– gebruikt moeten worden om naartoe te schrijven of van te lezen. Hoewel er een C-compiler voor de PRU door TI wordt geleverd, is het aanvankelijk handiger voor dit project om code in PASM te schrijven, om zeker te zijn van de timing.
De I/O-pinnen op het bord zijn op meerdere manieren te gebruiken, waaronder GPIO –pin wordt gekoppeld aan een geheugenadres– en rechtstreeks naar r30 (output) of r31 (input) op een van de PRU's. De modus voor een pin kan worden ingesteld met een *device tree overlay*. Ik heb hier een simpele editor voor gemaakt, *tools/pinselect.html*, op basis van de tabel van Derek Molloy[1].

2.2 Device Tree

De Flattened Device Tree[2] is een datastructuur om onderdelen van de hardware van een computer te beschrijven. Het is bedoeld als een nieuwe abstractielaag waardoor drivers los komen te staan van de kernel, opdat er niet voor elke SoC of andere hardware een nieuwe kernel geleverd hoeft te worden.
Bijkomend voordeel is dat er *overlay*s –extra stukken device tree– ingeladen kunnen worden terwijl het systeem draait. Voor de BBB betekent dit dat de I/O-pinnen die normaal voor HDMI worden ingezet, ook te gebruiken zijn voor GPIO door de overlay uit te laden en een andere in te laden.

2.3 Infraroodsensor

De gebruikte infraroodsensor (*TSOP1238*) bestaat hoofdzakelijk uit een 38 KHz band-pass-filter met wat componenten om basisstraling te filteren en het signaal te normaliseren. Dit wordt vervolgens aan een transistor gehangen die open gaat als het signaal hoog is. De output-pin is met een grote weerstand verbonden aan de 3,3 V en wordt geaard als de transistor geleidt (open is). Zie *doc/TSOP1138.pdf* voor de datasheet van een vergelijkbare sensor.
Er is voor gekozen om het signaal niet te inverteren met nog een weerstand en transistor, maar om in de PRU-code met het geïnverteerde signaal te werken.



Figuur 1: Diagram TSOP1138

2.4 Infraroodprotocol

Infraroodprotocollen bestaan uit een digitaal signaal dat met *on-off keying* op een draaggolf van ongeveer 38KHz wordt gezet. Praktisch elke fabrikant of zelfs elke productlijn heeft een eigen techniek om het digitale signaal op te bouwen. In dit geval is het protocol van een afstandsbediening van Panasonic bestudeerd.
Met *rawtiming* is gekeken hoe de knopcodes worden gemoduleerd. Uit analyse (zie *doc/rawtiming_analyse.xlsx*) volgt dat on-off keying op een schaal van ¿0,1ms gebeurt en er grofweg 3 categorieën voor zijn: kort, lang en "begin". Verder is een hoog signaal altijd kort, wat erop neerkomt dat alleen de duur van een laag signaal informatie draagt. Tot slot bestaat een pakket altijd uit 48 lage en 49 hoge signalen. Dit komt overeen met het protocol zoals beschreven in Arduino-IRremote[3].
Met de indeling in 3 categorieën kunnen pakketten worden uitgelezen. De basis hiervoor is te zien in *read_signal*. Dit programma leest IR-pakketten en toont ze op het scherm.

3 Programma

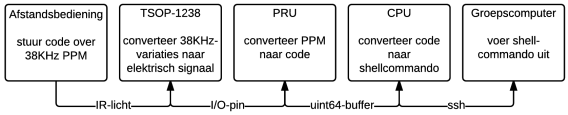
3.1 Compilatie

Voor het compileren van de C-code zijn gcc, glib en make nodig. Om te cross-compileren vanaf andere architecturen is de variabele *CROSS_COMPILE=arm-arago-linux-gnueabi-* en de SDK van TI nodig.
Voor de PRU-code zijn make en pasm nodig, waarvan de laatste beschikbaar is via de am335x_pru_package[4]. Om de Device Tree Overlay (*BB-BONE-IRMUX-00A0.dts*) te compileren is de Device Tree Compiler nodig. Het programma heeft in sommige distributies een patch nodig.[5]

3.2 Werking

Allereerst moet de device tree overlay voor de I/O-pinnen (1 pin in dit geval) en de PRU ingeladen worden. Dit wordt gedaan door "BB-BONE-IRMUX"naar */sys/devices/bone_capemgr.*/slots* te sturen. Zie ook *tools/laad_overlay*.
Hierna kan *ircommander* worden opgestart. Dit programma laadt het configuratiebestand en de PRU-driver in. Vervolgens wordt het PRU-programma ingeladen en uitgevoerd. Dit programma leest continu de IR-sensor uit tot er een volledig pakket is gelezen, waarna hij het als een uint64 in de buffer zet. De buffer is een eenvoudige cyclische array van 256 elementen. Zodra de buffer is geüpdatet (of er gebleken is dat hij vol zit), verstuurt de PRU een interrupt naar de CPU.
Ondertussen wacht het hoofdprogramma op interrupts van de PRU en zoekt de bijbehorende tekst voor de nieuwe pakketten in de buffer. Als er geen tekst te vinden is en de configuratie bevat *assignNewCommands=true*, vraagt het programma om een nieuwe tekst in te voeren.
Door als tekst een commando op te geven en de uitvoer te pipen naar de shell (of naar een shell over

SSH), is de afstandsbediening te gebruiken voor het opstarten van programma's en voor mediatoetsen. Dit is te zien in *config/ircommander.example.conf*, *tools/ircommander_groepscomputer* en de demovideo.



Figuur 2: Gegevensstroom in een voorbeeldopstelling (*tools/ircommander_groepscomputer*)

4 Toekomstige verbeteringen

Er kan op een aantal punten verbetering gebracht worden in het programma:

- Lopend gemiddelde: de PRU-code leest de gemiddelde waarde momenteel uit door de pin een aantal keer uit te lezen in vaste blokken. Door ruis (vooral bij zonlicht) lezen kleine blokken wel eens de verkeerde waarde uit, terwijl grote blokken geen signalen kunnen herkennen als de afstandsbediening het signaal verandert halverwege een blok. Met een lopend gemiddelde kan het exacte moment worden bepaald waarop de sensor van hoog naar laag wisselt en vice versa. Als alternatief kan natuurlijk ook een analoog circuit worden gebouwd.

- Meerdere modi: het C-programma staat nu één commando per knopcode toe. Door knoppen (of commando's op de standaardinvoer) toe te wijzen aan het wisselen van de modus, kan een knop meerdere resultaten geven. Hoewel deze functie ook kan worden vervuld door het programma waar naar gepiped wordt, leidt dit tot onduidelijke commando's in de huidige opzet, waarin gepiped wordt naar bash.

- Het omzetten van knopcodes naar commando's outsourcen: het stateless knopcodes omzetten naar tekst kan ook met *sed* uitgevoerd worden. Hierdoor wordt het programma compacter en de afhankelijkheid van glib vervalt. Om het bovenstaande punt ook te verwerken, is *sed* alleen niet voldoende.

5 Verwijzingen

1. http://derekmolloy.ie/beaglebone/beaglebone-gpio-programming-on-arm-embedded-linux/
2. https://en.wikipedia.org/wiki/Device_tree
3. https://github.com/shirriff/Arduino-IRremote/blob/master/IRremote.cpp
4. https://github.com/beagleboard/am335x_pru_package/tree/master/pru_sw/utils/pasm_source
5. https://eewiki.net/display/linuxonarm/BeagleBone+Black#BeagleBoneBlack-Upgradedistro%22device-tree-compiler%22package