Motorregeling met embedded Linux

Embedded Linux is een interessante optie voor motoraansturing, maar de vraag is of de realtimeprestaties goed genoeg zijn. Om dat uit te zoeken, heeft Fontys Hogeschool ICT samen met een aantal mkb'ers, NXP en Vanderlande Industries een case uitgevoerd rond de regeling van een sorteersysteem.

Ruud Ermers
Ed Schouten
Laurens Timmermans
Jeroen Waterman

Linux is steeds meer gemeengoed aan het worden in embedded systemen. Het platform is gratis te verkrijgen en biedt een rijke omgeving voor applicatieontwikkeling. Aan de andere kant is er relatief zware hardware nodig en is het van oudsher geen realtime systeem. Vooral voor machineaansturing is dat laatste bezwaarlijk. Het lectoraat Architectuur van Embedded Systemen van Fontys Hogeschool ICT doet daarom samen met een tiental mkb'ers onderzoek naar de toepasbaarheid van embedded Linux in embedded en mechatronische systemen, gesteund door NXP en Vanderlande en door een Raak-subsidie (Regionale Aandacht en Actie voor Kenniscirculatie).

Om inzicht te krijgen in het realtimegedrag van Linux hebben we een praktische case opgezet rond een systeem van Vanderlande. Daarbij hebben we de bestaande motorbesturing, op dat moment werkend met twee 8 bit Atmel AVR-processoren, vervangen door een 32 bit Arm9-processor met Linux. Dit was voornamelijk een *proof of concept*, want als het met twee AVR-chips werkt, is er weinig reden om een zware Arm9-processor in te zetten. De trend richting embedded Linux op het Arm9-platform is echter duidelijk waarneembaar.

Vanderlande levert systemen en de bijbehorende diensten voor bagageafhandeling op luchthavens en in distributiecentra van

De loop sorters van Vanderlande bestaan uit onafhankelijke modules die hun vracht via een kiepbeweging kunnen uitsorteren.

post- en exprespakketdiensten. Voor dat laatste doelgebied zet het zogenaamde *loop sorters* in, lopende banden in een lusvormige configuratie die op een bepaalde locatie hun vracht eraf kunnen kieperen voor verdere distributie. Ze zijn opgebouwd uit een rail, waarover op zichzelf staande transportmodules bewegen.

Grote brei

De transportmodules hebben een eigen controller aan boord voor de sorteerbeweging en het verwerken van de infraroodcommunicatie, waarmee het centrale systeem ze aanstuurt. De sorteerbeweging wordt opgewekt door een borstelloze DC-motor en geregeld met Hall-sensoren, die zo'n 4500 interrupts per seconde genereren.

Uit de bestudering van het huidige systeem kwam al vrij snel naar voren dat het motorregelalgoritme een goede kandidaat was voor het verkrijgen van inzicht in het realtimekarakter van embedded Linux. Het draait namelijk in een subroutine van 20 kHz: net wel of net niet haalbaar voor het open besturingssysteem op een 200 MHz Arm-processor. Uit eigen onderzoek en de literatuur bleek 1 kHz nog eenvoudig haalbaar te zijn en 100 kHz weer een stap te ver. De frequentie van het algoritme kon eventueel nog worden teruggeschroefd, maar dat zou wel een impact hebben op de kwaliteit van de regeling.

De infrarooddetectie is met deze hardware niet mogelijk, omdat deze informatie gesampled wordt op 100 kHz, te snel dus. Omdat dit gedeelte eigenlijk minder interessant was, besloten we om hier verder weinig moeite in te steken. Het versturen van informatie naar de transportmodule kan ook wel op andere manieren.

Uit het vooronderzoek volgde ook de keuze voor een 32 bit omgeving. Maar eigenlijk lag dit al op voorhand vast omdat NXP betrokken was bij het onderzoek en we graag gebruik wilden maken van hun kennis rond embedded Linux. De keuze viel dan ook al snel op de LPC3250, een Arm9-gebaseerde processor van NXP. Arm7-processoren hebben we vermeden omdat die geen MMU aan boord hebben, waardoor Linux-ontwikkeling lastiger wordt.

Om geen wijzigingen aan de hardware te hoeven maken, besloten we een truc toe te passen die we ook in een eerdere case hadden ingezet: we verwijderden de bestaande microprocessor en verbonden de benodigde I/O-lijnen rechtstreeks – met een kleine elektronische aanpassing hier en daar – met het ontwikkelbordje. Na aardig wat soldeerwerk resulteerde dit in een opstelling met een grote brei draadjes.

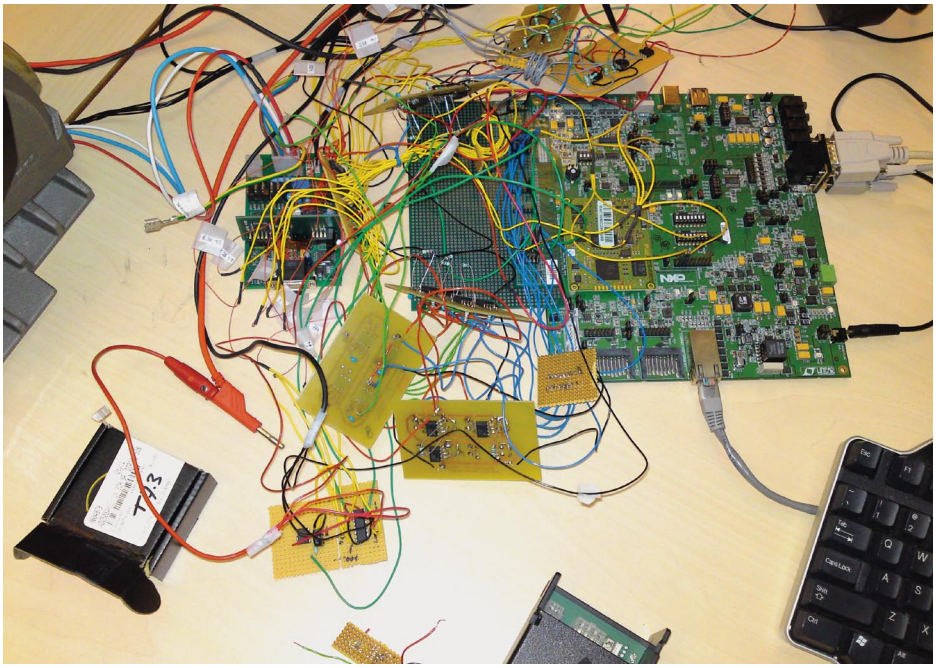
Tikloos

In een eerder onderzoek heeft ons lectoraat al gesteld dat mkb-bedrijven zorgvuldig moeten bekijken of het ontwikkelsysteem de toegepaste processor en periferie ondersteunt. Voor dit project gebruikten we daarom het Phytec Phycore Rapid Development-bord. Dit wordt geleverd met een kant-en-klare Linux-omgeving en diverse tools. NXP biedt ondersteuning via het LTIB-gereedschap (Linux Target Image Builder), een opensource tool om *board support packages* te maken. Alle benodigdheden voor het compileren en builden van de software zijn daardoor aanwezig, en het root-bestandssysteem met kernel, bibliotheken en benodigde applicaties is eenvoudig en flexibel te genereren. Bovendien hoeven ontwikkelaars zich dankzij een collectie scripts niet te verdiepen in cross-compileren, automake en applicatie- en bibliotheekafhankelijkheden.

Van origine is Linux geen realtime systeem. Zo draaien de standaard timers op een nauwkeurigheid van 100 Hz en zijn de bijbehorende bibliotheekfuncties vooral ingericht voor coöperatieve multitasking. De afgelopen jaren is er echter behoorlijk gesleuteld aan het realtimegedrag. Bekend zijn de uitbreidingen waarbij Linux als taak draait in een RTos (RTLinux, RTAI), maar met name de laatste twee jaar is ook het OS zelf aangepast ten behoeve van realtimeprestaties. Er zijn onder meer hoge resolutie timers (HR-timers) toegevoegd, die op een aanzienlijk hogere frequentie werken. Sinds versie 2.6.18 is realtimegedrag een van de instellingen bij het compileren van de kernel.

Daarmee zijn we er echter nog niet. Veel hangt af van de koppeling tussen de kernel enerzijds en de gebruikte processor en aangesloten periferie anderzijds. Voor de LPC3250 ontbrak bij aanvang van het project de timer-interface – althans de HR-variant – die ervoor zorgt dat de interrupts van de systeemklok in de software terechtkomen en die daarmee het kloppend hart vormt van het realtime systeem. Twee projectleden hebben de interface daarom zelf geschreven en als patch gedoneerd aan de Linux-gemeenschap. Ondertussen is deze bijdrage opgenomen in de standaard kernel. Ook ontbraken de ADC- en PWM-drivers en waren in de GPIO-driver (General-Purpose I/O) geen interrupts mogelijk. Ook deze software hebben we zelf gemaakt of aangepast, met de hulp van enkele studenten die de minor Embedded Systems volgden aan Fontys Hogeschool ICT.

Verder hebben we het realtimegedrag verfijnd door de kernel *tickless* te maken. Normaal genereert de timer periodiek een interrupt, zeg honderd keer per seconde. Om een bepaalde periode te wachten, telt het systeem het aantal interrupts tot de gewenste tijd is verstreken. Bij een tik-



Voor de test werd de bestaande module direct verboden met een Phytec Phycore Rapid Development-bord.

loze kernel treden timerinterrupts echter alleen op na de gewenste periode, waardoor er veel minder interrupts afgehandeld hoeven te worden en het systeem daar minder tijd aan kwijt is.

Vloeiend

Gewapend met dit geoptimaliseerde embedded systeem hebben we de bestaande code van Vanderlande omgezet. Dankzij C-constructen zoals macro's, defines en conditionele compilatie konden we de code binnen twee dagen omschrijven tot een kernelmodule. De 20 kHz timerroutine draait daarbij als kernelthread die HR-timers gebruikt. De timer wordt iedere keer geherprogrammeerd voor de volgende interrupt, met inachtneming van de correctie voor de tijd die het afwerken van de code in beslag neemt. Dit levert een nauwkeurige frequentie op. De main()-functie van de bestaande code bevat ook een loop voor een aantal achtergrondtaken, zoals het starten van de AD-conversie. Die constructie is in een kernelmodule niet praktisch. Na enige rondgraven in de codestructuur besloten we de achtergrondtaken te combineren en aan de 20 kHz loop toe te voegen. Dit verhoogt de frequentie van de AD-conversie, maar heeft geen directe invloed op de motorregeling.

Om de besturing van buitenaf te kunnen beïnvloeden, hebben we enkele kernelmodules aan het systeem toegevoegd die de interactie met *user space*-toepassingen mogelijk maken via het standaard Virtual File Sytem van Linux. De Uart-interface gebruikten we om gesimuleerde infraroodinformatie naar de kernelmodule te sturen.

Het consortium was initieel opgericht om inzicht te krijgen in de benodigde expertise en moeite voor ontwikkeling op embedded Linux. Daarom hebben we nauwgezet bijgehouden hoeveel tijd alles in beslag nam. Om een idee te geven: het vooronderzoek kostte ongeveer tachtig uur, het bouwen van de hardware-interfaces veertig uur (niet ons sterkste punt), het initieel werkend maken van het Phytec-bord rond de twintig uur en het schrijven van de kernelmodules en het patchen van de drivers tachtig uur.

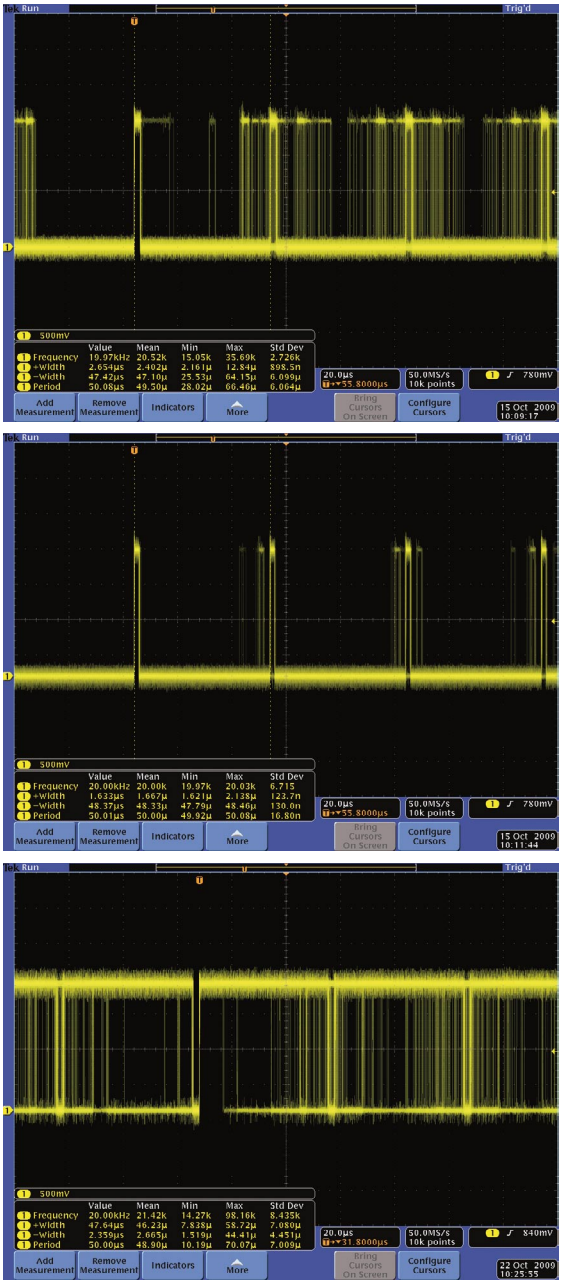
Om de vraag te beantwoorden of de 20 kHz loop haalbaar is, hebben we een oscilloscoop aangesloten op een pin die een logisch hoog signaal geeft bij het optreden van de timer-interrupt en een logisch laag seintje bij het verlaten van de routine. Uit metingen zonder draaiende motor en communicatie bleek dat de interrupts inderdaad optreden met een gemiddelde van 20 kHz. De regelmatigheid was echter vrij laag, de standaarddeviatie bedroeg 2,72 kHz. De reden hiervoor is dat het systeem ook de andere interrupts die optreden, moet bedienen. Het was dus zaak om uit te zoeken waar die andere interrupts vandaan kwamen. Omdat er vrijwel geen netwerkactiviteit was, ging de aandacht uit naar de ADC. Metingen toonden aan dat deze met een frequentie van ongeveer 2 kHz interrupts genereert. Dit probleem zou opgelost kunnen worden door de prioriteit van de ADC-interrupts of de samplefrequentie aan te passen.

Als de motor wordt aangezet, genereren de Hall-sensoren op ongeveer 4,5 kHz interrupts, vergelijkbaar met de originele AVR-oplossing. Onder deze omstandigheden stijgt de processorbelasting fors, doordat er nu verschillende soorten interrupts zijn van diverse bronnen. Als de loop niet in kernel space maar in user space zou draaien, zou de aansturing nu niet haalbaar zijn. De transportmodule reageerde evenwel zeer vloeiend op externe commando's. We hebben dus geen verdere moeite gestoken in het stabiel maken van de 20 kHz loop – overigens ook omdat de tijd dit niet toeliet.

Roet in het eten

Deze case toont aan dat met een getweakt embedded-Linux-systeem een motorsturing met een regelfrequentie van 20 kHz nog nét te realiseren is. Het platform is hierbij bijzonder stabiel gebleken en het nieuwe realtimemodel werkt uitstekend, ook in weerwil van de hardware-interrupts. Linux op een Arm9-processor kan dus een interessante optie zijn voor motoraansturing. Aspecten als onderhoudbaarheid, uitbreidbaarheid, robuustheid en ontwikkeltijd zijn daarbij van groot belang en als belangrijke meerwaarde aan te voeren.

Toch wil dit niet zeggen dat een willekeurige toepassing met embedded Linux dit realtime gedrag zal halen. Veel hangt af van de op het systeem aanwezige applicaties, drivers (met name netwerkdrivers willen nog vaak roet in het eten gooien) en de afstemming van de diverse systeemonderdelen op elkaar. Linux maakt het in ieder geval mogelijk om een groot aantal onderdelen hiervan zelf in te stellen door de realtime Posix-bibliotheken, prioriteitsmechanismen, HR-timers, moderne drivertechnieken en instelbare schedulingalgoritmes aan te bieden.



Met de motor en communicatie uitgeschakeld bleek de loop de beoogde 20 kHz te halen, maar met een grote afwijking (boven). Dat bleek door de ADC te komen, die met 2 kHz interrupts afvuurt; na uitschakelen van deze converter werd het beeld veel netter (midden). Na het aanzetten van de motor genereren de Hall-sensoren interrupts op ongeveer 4,5 kHz en stijgt de processorbelasting aanzienlijk (onder).

Ruud Ermers, Ed Schouten en Laurens Timmermans zijn verbonden aan het lectoraat Architectuur van Embedded Systemen bij Fontys Hogeschool ICT. Ermers is projectleider Embedded Linux. Jeroen Waterman werkt bij Vanderlande Industries als controls development engineer. Op vrijdag 25 juni organiseert Fontys het gratis toegankelijke seminar 'Embedded Linux voor het mkb - a guide to professional development'. Meer informatie kunt u vinden op www.fontys.nl/embeddedsystems/embeddedlinux.

Redactie Pieter Edelman