RC5 bediende stekkerblok

Besparing en luxe in 1 doosje

Auteur: Bart De Ridder (Zatar) [0905-004]

Iedereen kent het wel... tv's, dvd spelers, recorders, enz. het verbruikt allemaal als je de stekker niet uittrekt. Dat verbruik kan je jaarlijks tot zo'n €20 per toestel kosten. Dit is echter simpel op te lossen. Een docent van de hogeschool Gent kwam af met het idee en enkele van zijn studenten gingen de uitdaging aan.

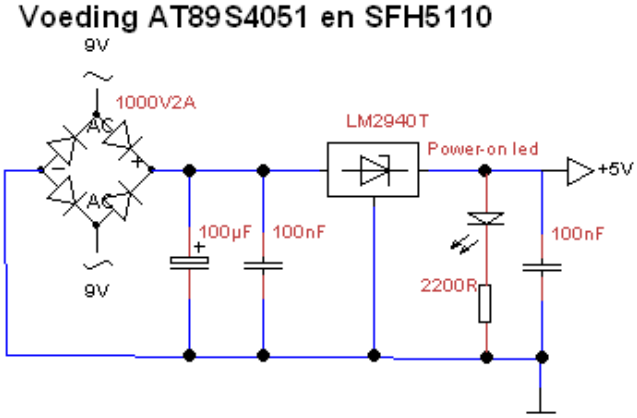
Het was de bedoeling van te zorgen voor gebruiksgemak en besparing op de energierekening. Er moest dus gezocht worden naar een manier om luxe te combineren met zuinig verbruik.

De oplossing hiertoe was redelijk simpel. Als je ervoor zorgt dat het werkt met de afstandsbediening van de tv dan spaar je een extra afstandsbediening uit. Je kan het bedienen vanuit je luie zetel en het zou dan nog eens zuinig zijn bij?

Daar moet even een berekening bij. De meeste van de toestellen schakelen in stand-by stand enkel hun scherm uit. Daarom schakelen we met de stekkerblok en een triac sturing de voedingsspanning uit.
Een toestel verbruikt in stand-by tot 15W. De stekkerblok op zen geheel verbruikt (in theorie) ±0,32W en zelfs slecht ±0,13W bij gebruik van de smd component LP2981-50DBVR.
Laat ons uitgaan van een slechtere situatie. De toestellen op een gewone stekkerblok verbruiken in stand-by allen tezamen 25W. De stekkerblok verbruikt bv. 1W. Dat is dan 24W minder te betalen.

I. De elektronica

Hierbij gaan we beginnen met de voeding van het geheel. In het prototype is gebruik gemaakt van de LM2940T. Dit is een gewone low drop-out 5V spanningsregelaar.

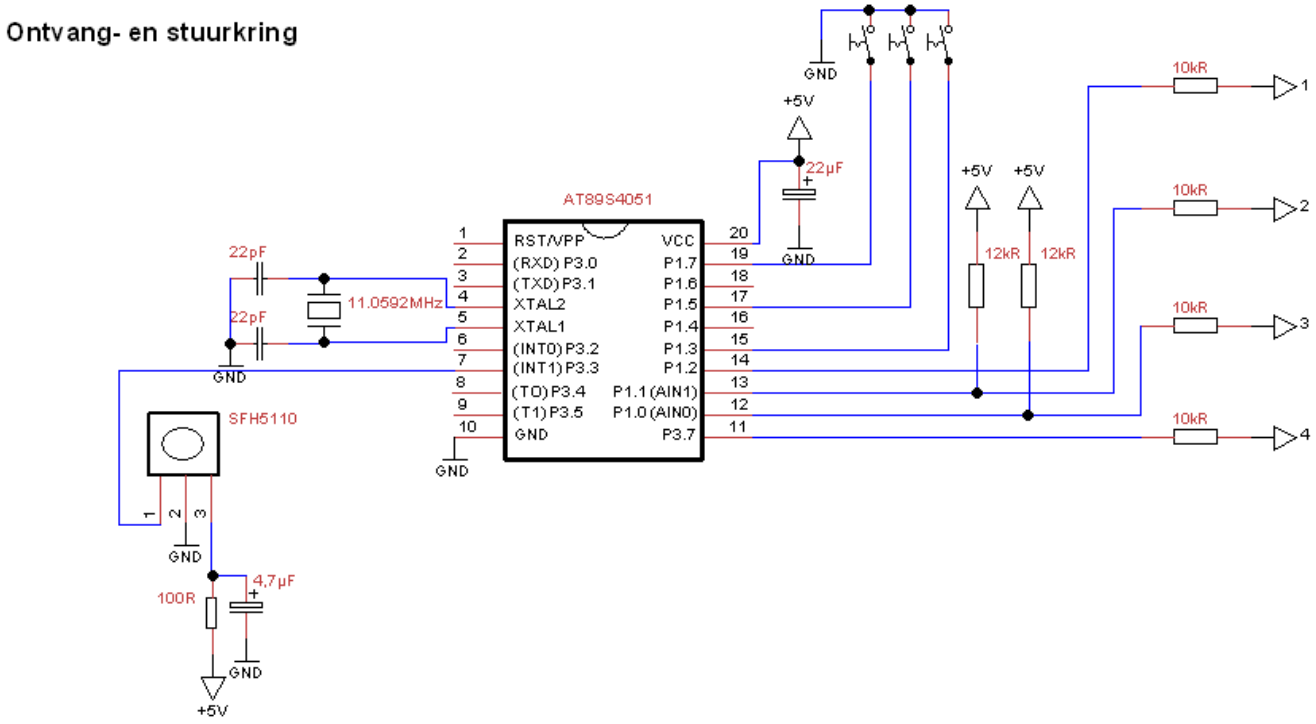


Met een transformator brengen we de netspanning over naar 9V wissel. Die wisselspanning wordt gelijkgericht en afgevlakt met de elco van 100µF.

Om het oscilleren van de spanningsregelaar te voorkomen zijn 2 condensatoren van 100nF aan de in- en uitgang gezet.

Een low-current led geeft aan of de voedingsspanning aanwezig is en dus de stekkerblok inzit.

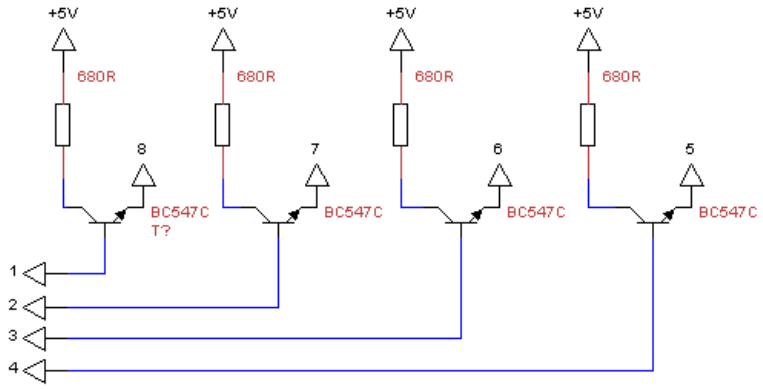
De volgende stap brengt ons bij het ontvang en decodeercircuit.



De SFH5110 (zie fig. 3) is onze infrarood demodulator. Die gaat het 36kHz signaal wegfilteren en op zijn uitgang een reeks van 1tjes en nullen zetten. Die reeks van enen en nullen wordt toegeleverd aan de interruptpin P3.3. In het programma wordt bit per bit bekeken en afhankelijk daarvan de uitgang hoog of laag gemaakt. Hoog komt hierbij overeen met de netspanning aanleggen en laag komt overeen met ze uitschakelen.

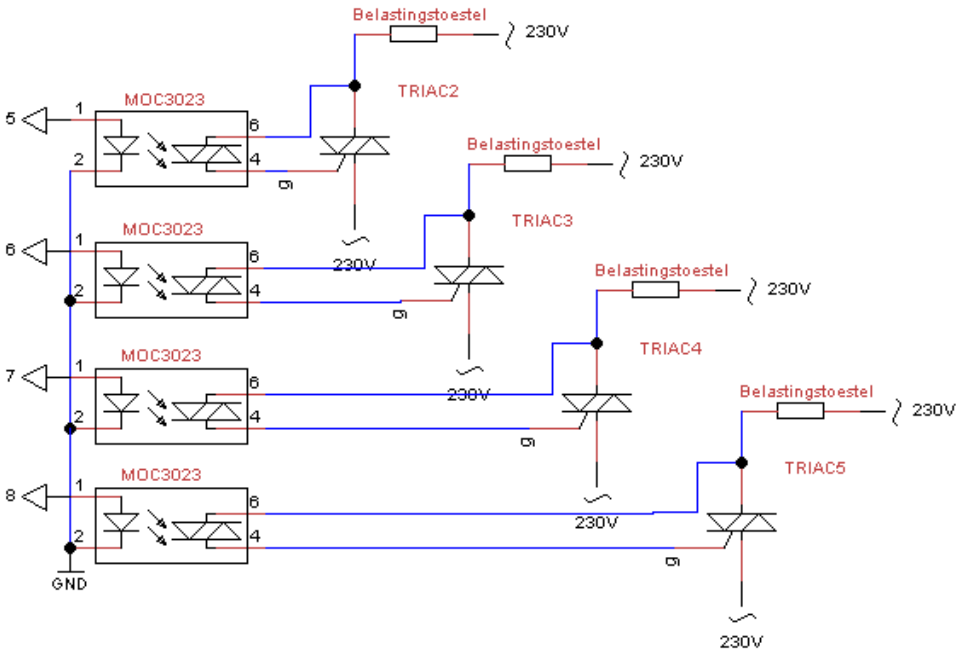
De waarde van het kristal is belangrijk omdat de timing in het programma hierop afgestemd is.
Het ontvangerscircuit zit in een apart doosje zoals te zien is in figuur 1. Dit is ook de reden dat de elco (of een tantaalcondensator) van 22µF te zien is aan de microcontroller. Wanneer de uitgangen hoog gemaakt worden zou de microcontroller meteen stroom moeten leveren. Omdat die stroom eerst de draadlengte moet overbruggen zou dit niet gaan en zou de microcontroller steeds herstarten, resulterend in 0,92 – 1,44V op de uitgangen.
Door de elco is er een kleine buffer die de stroom geeft tijdens de wachttijd.
Aan de pinnen 12 en 13 van de at89S4051 zijn 2 pull-up weerstanden opgenomen omdat die er intern niet zijn.

Wanneer nu een uitgang hoog gemaakt wordt gaat de bijbehorende transistor in geleiding. Gevolg is dat het aangeschakeld optocoupler aangaat en de netspanning wordt aangelegd. De transistoren dienen dus om de optocouplers te schakelen. Ze zijn van het type BC547C omdat deze een grotere versterkingsfactor hebben dan het BC547B type, en dus sneller in verzadiging gaan.



Ze moeten er zitten omdat er per poort een maximum stroom is van 15mA.
Dan rest ons nog het laatste deel van de elektronica en dat is de triac sturing.
Door middel van de optocouplers bekomen we een galvanische scheiding van de netspanning en de elektronicavoeding. Omdat de uitgangen van de optocouplers slechts 60mA aankunnen worden ze gebruikt om een andere triac aan te sturen.

De optocouplers zijn van het type MOC3023. Deze kunnen aangestuurd worden met 5mA. Als je het niet echt doet om zuinig te werken kan je ook type MOC3020 nemen. De voorschakelweerstand moet dan ietwat kleiner worden.



II. Het programma

Vooraleer er een programma kon geschreven worden moest het nodige opzoekwerk gedaan worden. Welk protocol, hoe het herkennen, randcomponenten etc.
Hiervoor is gebruik gemaakt van een afstandsbediening waar je het protocol bij kon instellen. Code 4237 zou bv. RC6 zijn geweest (het veelgebruikte protocol van Philips in de nieuwere toestellen).

Om te weten wat we binnen kregen hebben we aan de uitgang van de demodulator (SFH5110) een oscilloscoop gehangen. Wanneer nu een knop ingedrukt werd kregen we het bijbehorende signaal te zien.

Omdat we werken met interrupts, moeten we zien dat op het moment er een interrupt is, de interrupts afgezet worden:

```
ORG 13h ;interrupt van de ontvanger interruptbron 0
MOV IE,#00h
MOV R6,#01h
RETI
```

MOV IE,#00h zorgt hiervoor. Wanneer we gaan kijken in de datasheet van de at89**S**4051 dan zien we bij interrupt registers het register IE. Dat staat voor Interrupt Enable register. Kortom je bepaalt er welke interrupts aanstaan. Door er de waarde 00h te schrijven zet je alle interrupts uit (#00h slaat op waarde, moest je gewoon 00h schrijven dan wordt er de waarde gezet die op adres 00h staat). We zetten de waarde 01h in register R6. Hiermee gaan we later controleren of er een signaal wordt ontvangen.

De volgende stap is de beginstand. Wanneer de stekkerblok in het stopcontact gestoken wordt moeten de uitgangen laag zijn. Dit doen we door een 0 weg te schrijven naar de uitgangen waar de transistors aanhangen. Dus P1.0, P1.1, P1.2 en P3.7 krijgen een 0 op hun uitgang.
Vervolgens stellen we de timer in als 16 bit mode timer en roepen we de subroutine ***hervat*** aan.

```
begin:      MOV P1,#11111000b
            CLR P3.7
```

```
            MOV TMOD,#01h       ;16 bit mode timer 0
            MOV TCON,#01h
            CALL hervat
```

In de subroutine hervat stoppen we de timer door de timer run-bit te clearen. Daarna stellen we de timer in en zetten we de interrupts aan. Met ***RET*** geven we aan dat we de subroutine beëindigen en terug verdergaan in het programma.

```
hervat:     CLR TR0
            MOV TH0,#0FCh ;(11.052 * 900 µs / 12) stel timer 1 in
            MOV TL0,#0C1h ;op 900 µs bij 11.052MHZ + offest (200*20)µs
            MOV IE,#10000100b ;interrupt 0 aanzetten
            RET
```

We komen dus uit aan de stap ***lus***. CJNE staat voor Compare and Jump If Not Equal. R6 wordt dus vergeleken met de waarde 01h. Indien in R6 een waarde staat verschillend van 01h wordt er naar ***lus*** gesprongen. Er wordt dus gewacht tot er een interrupt optreed want dan wordt in R6 de waarde 01h gezet.
Op het ogenblik dat er iets ontvangen wordt zetten we R6 terug op 0h en springen we naar het label ***ontvangt***.

```
lus:        CJNE R6,#1h,lus
            MOV R6,#0h
            JMP ontvangt
```

We beginnen hier met de timer aan te zetten met ***SETB TR0***. Omdat we niet weten of er iets in de accu zit maken we hem leeg. De TimerFlag clearen we ook. Het signaal dat aan P3.3 hangt is dat van de ontvanger. We steken dit signaal in de carry. Door een Rotate Left through Carry te doen van de accu steken we dus die eerste bit in de accu. We krijgen bv. A = “00000001b” in geval de bit in de carry een 1 was. We roepen de subroutine D1 aan.

```
ontvangt:   SETB TR0    ;OFFSET TIME bijtellen
            CLR A
            CLR TF0
            MOV C,P3.3
            RLC A
            CALL D1     ;ontvangt 2e bit (hoge niveau )
            CALL D1     ;ontvangt 2e startbit (lage niveau)
            MOV R5,A    ;sla de startbits op in register A
```

We krijgen hier weer een lusje. Zolang de TimerFlag op 0 staat wordt er naar D1 gesprongen. Op het moment de timer een overflow krijgt zal er worden verdergegaan. De timer wordt weer afgezet en opnieuw ingesteld. Hij wordt vervolgens terug aangezet en de timerflag wordt gecleared. De volgende bit op P3.3 wordt uitgelezen en in de carry gezet. Opnieuw een Rotate Left through Carry van de accu. Als we ons resultaat van daarstraks erbij nemen en er zit op huidig moment een 1 in de carry dan bekomen we: A = “00000011b”. De carry wordt weer leeggemaakt en we gaan verder in het programma.
We komen weer aan de call instructie en doorlopen lus D1 nog eens.

Dan wordt de waarde van de accu in register R5 gezet.

```
D1:          JNB TF0,D1
             CLR TR0
             MOV TH0,#0FCh       ;(11.0592 * 900 µs / 12) stel timer 1 in
             MOV TL0,#0C1h
             SETB TR0
             CLR TF0
             MOV C,P3.3
             RLC A
             CLR C
             RET
```

In de accu zit nog een waarde die naar register R5 is geschreven, daarom wissen we hem voor we verdergaan. In register R7 steken we de waarde 8h. We gaan de eerste 8 bits van de code ophalen door opnieuw de subroutine D1 aan te roepen. Met ***DJNZ*** wordt er telkens van R7 één afgetrokken, gekeken of het resultaat nul is én indien dit geen nul is terug naar het label ***eerste8*** gesprongen. Na 8 keer de lus te doorlopen hebben we dus de 8 bits die we moeten hebben. We zetten die in register R1 zodat we de accu kunnen clearen en daar verder mee kunnen werken.

```
             CLR A               ;reset accu
             MOV R7,#8h
eerste8:     CALL D1             ;eerste acht bits ophalen
             DJNZ R7,eerste8
             MOV R1,A            ;sla deze waarden op in register 1
             CLR A
```

Nog eens hetzelfde maar dan met registers R2, R3 en R4.

```
             MOV R7,#8h ;haal de laatste 4 waarden van het adres op
midden:      CALL D1 ; + de 1e 4 van het commando
             DJNZ R7,midden
             MOV R2,A ;sla deze waarden op in register 2
             CLR A

             MOV R7,#8h
laatste8:    CALL D1
             DJNZ R7,laatste8
             MOV R3,A ;LSB=R3.0 MSB=R0.3
             CLR A

             MOV R7,#8h
extra:       CALL D1
             DJNZ R7,extra
             MOV R4,A ;LSB=R3.0 MSB=R0.3
             CLR A
```

Nu gaan we uit de eerste 8 bits bepalen of het om een RC5 protocol gaat of niet. We zetten de eerste 8 bits die we in R1 gestoken hebben in de accu. Door een logische AND functie bekomen we de code die we moeten vergelijken. Is bij de eerste stap ***CJNE*** de waarde van R1 al niet gelijk aan de code ernaast dan springen we naar label ***RC6***. In dat label wordt dezelfde controle uitgevoerd om te zien of het een RC6 code betreft.
Komen de 3 codes in R1, R2 en R3 overeen dan wordt er gesprongen naar het label ***check***.

```
RC5_Samsung: MOV A,R1
             ANL A, #00111111b
             MOV R1,A
             CJNE R1,#00010101b,RC6
             CJNE R2,#01010101b,RC6
             CJNE R3,#10100101b,RC6
             JMP check
```

Hier wordt gekeken in welke stand de schakelaars staan. Afhankelijk van de stand gaat de bijbehorende uitgang mee uit met de hoofduitgang P3.7. Dit kan handig zijn om de dvd-recorder aan te laten staan als je iets wil opnemen.
Er zit een vertraging in door de “call tijd” instructie. Die dient om ervoor te zorgen dat eerst het tv-toestel dat aangekoppeld zou zijn in stand-by kan gaan vooraleer de spanning ervan weg te nemen.

```
check:       CLR A
             MOV C,P1.7
             RLC A
             MOV C,P1.5
             RLC A
             MOV C,P1.3
             RLC A
             MOV R4,A
             CALL tijd
             CALL tijd
             JB 20h.0,uitschakelen

inschakelen: MOV P1,#11111111b
             SETB P3.7
             setb 20h.0
             JMP end

uitschakelen: MOV A,P1
             ANL A,R4
             CPL A
             ORL A,#11111000b
             MOV P1,A
             CLR P3.7

tijd:        MOV R1,#0FFh
             MOV R2,#0FFh
             MOV R3,#5d
tlus:        DJNZ R1,tlus
             DJNZ R2,tlus
             DJNZ R3,tlus
             RET
```

Onderdelen Lijst:
IC1 = LM2940T
IC2 = AT89S4051
IC3, IC4, IC5, IC6 = MOC3023
X1 = 11.0592MHz
Rc1 = SFH5110 (TSOP 1736 kan ook maar dan is een andere print nodig)
C1 = 100µF/25V
C2 en C3 = 100nF
C4 = 4,7µF/10V
C5 en C6 = 22pF
C7 = 22µF/10V
R1 = 2200Ω
R2 = 100Ω
R3, R4 = 12kΩ
R5,R6,R7 en R8 = 10kΩ
R9, R10, R11 en R12 = 680Ω
B1 = bruggelijkrichter vanaf 300mA
T1, T2, T3 en T4 = BC547C
T5, T6, T7 en T8 = Triac naar keuze
D1 = low-current led
S1, S2 en S3 = schuifschakelaar

Van dit project zijn de programmacode, het hex bestand, schema en printplaat zijn te downloaden op:
http://www.schematheek.net/index.php?p=magazine&editie=0905&artikel=4

Aandacht!
Ik wil wel nog meedelen dat dit een prototype is dat enkel getest is met een TV en een printboormachine eraan. Dit is thuis enkele keren gebeurd en op de projectverdediging. Er kan dus niet gegarandeerd worden dat het toestel werkt met elk type belasting.
Het bouwen van het toestel is volledig op eigen risico!
Verder hoop ik dat dit je niet afschrikt en veel bouwplezier!!!