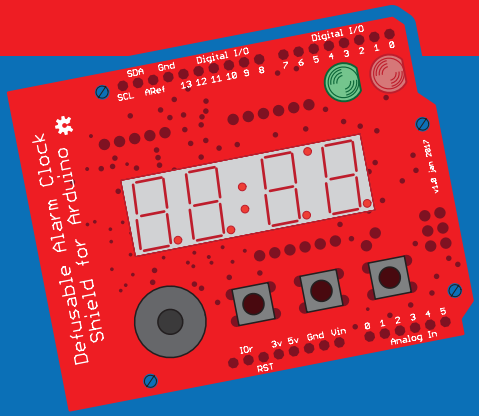
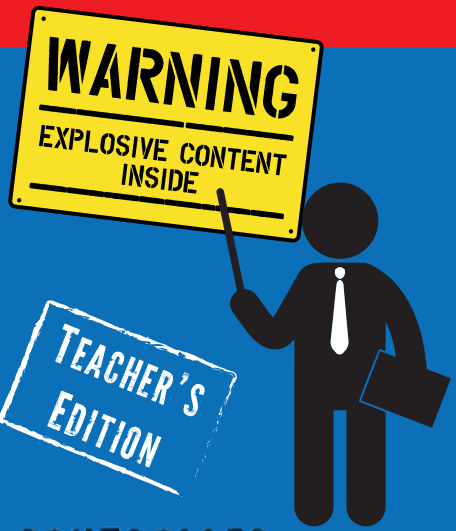
Defusable Alarm Clock Shield for Arduino





How to program your own defusable bomb controller...

Waarschuwing vooraf / disclaimer

Dit shield is niet bedoeld en ook niet ontworpen om mensen de schrik aan te jagen en al helemaal niet voor andere 'slechte' doeleinden. Natuurlijk zou het daar wel voor gebruikt kunnen worden, dat doet u dan geheel op EIGEN risico en nog de vakvereniging i&i nog de ontwikkelaar, nog module leverancier JKSOFT Educational kunnen hiervoor verantwoordelijk gehouden worden. Dit project is duidelijk bedoeld als nuttige, leuke, uitdagende leerervaring en de casus zorgt ervoor dat het nuttige en interessante met elkaar verenigd worden. Als u de module als 'bom' programmeert zorg er dan s.v.p. ook voor dat deze niet per ongeluk ergens achter gelaten wordt en voorkom paniek. Door de 'bom'-draadjes uit de module te halen en er standaard een wekker sketch op te zetten en/of shield niet op uw UNO te laten zitten loopt u geen onnodig risico. Druk s.v.p. uw leerlingen dit ook op het hart als u ze met dit shield laat experimenteren.

N.b. ik heb wel eens op weg naar een i&i overleg mijn Arduino les voor de hogeschool voorbereid in de trein en hoorde toen ook al paar jongeren tegen elkaar zeggen, "waar is die gast mee bezig, lijkt wel bom" en dat zonder dit shield (kwestie teveel films gekeken), maar lijkt me dat voorkomen beter dan genezen is. Verder heel veel plezier met deze workshop.

V1.1 april 2017

Inhoudsopgave

Inleiding

Altijd al uw eigen 'bom' in elkaar willen knutselen en vervolgens zelf 'onschadelijk' willen maken? Grijp nu uw kans! De kans op een daadwerkelijke ontploffing is gelukkig vrij klein (tenzij u condensatoren om gaat polen), maar voor u zijn genoeg uitdagingen in deze workshop ten einde uw Arduino-grenzen te verleggen. Voor de pacifisten onder ons: ook bruikbaar als wekker of stopwatch. In deze workshop leert u (meer) geavanceerd gebruik te maken van uw Arduino vaardigheden.

Doelgroep

Doelgroep van deze versie is de docent Informatica (en/of eventueel NLT) uit de HAVO/VWO bovenbouw welke al enige praktische basiskennis van Arduino heeft en daarnaast bekend is met een gangbare programmeertaal zoals C, C++, C#, Python of Java(Script).

Over ontwikkelaar

Johan Korten is oud-bestuurslid van i&i en heeft zes jaar Informatica in het VO gegeven. Nu is hij docent Informatica aan de Hogeschool van Arnhem en Nijmegen met speciale focus op zorgtechnologie, embedded systemen en mobiele applicaties. Behalve het ontwikkelen van toepassingen rond patiënt simulatie werkt hij aan domotica toepassingen voor hobbymatig en (zorg)professioneel gebruik.

Johan Korten (jakorten@jksoftedu.nl)
+31 6 14239260 / www.jksoftedu.nl

Vereiste Benodigdheden

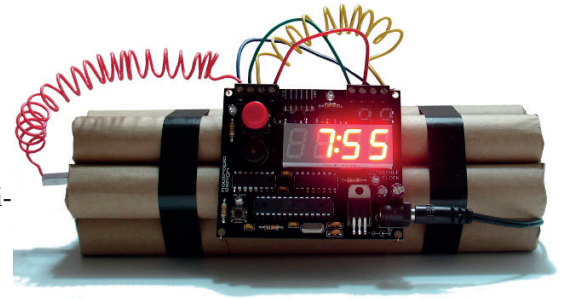
Benodigdheden:
- Laptop met Arduino IDE (en zo mogelijk internet verbinding)
- Arduino UNO of vergelijkbaar
- USB-kabel
- “Defusable Alarm Clock Shield”

Project Achtergrond

Op 6 september 2011 kwam John Baichtal op Makezine.com met een review van Mike Krumpus (Nootropic Design) over een grappige kit die ook wel beetje spannend is, dus waar u gegarandeerd mee scoort bij uw leerlingen maar let op, wellicht dient u ook uw pedagogische vaardigheden aan te spreken want een voor u het weet gaan uw leerlingen er mee aan de haal met alle gevolgen van dien.

Waar gaat het om: een kit waarmee eenvoudig een ‘bom’ die niet zo misstaan in MacGuyver-achtige actiefilms te simuleren is: zelf toevoegen karton van closet- / keukenrol en uiteraard Duct Tape (of elektriciteitstape).

Ik heb voor u deze typische Make klassieker omgezet in een Shield voor uw basis Arduino. Voor we richting de workshop zelf gaan even nog wat technische details waar het shield over beschikt en waar het



Figuur 1. Originele Defusable Clock door nootropicdesign.com

uiteindelijk om draait:

- 4-digits 7-segment LED display
- Real-time clock (DS3231)
- Tilt sensor
- 2 indicatie LED's
- 3 knopjes
- 6 'bom'-draad pinnen
- Buzzer
- Extender Chip voor Segment Display (HT16K33)
- Shield voor op de Arduino UNO Rev3 (of vergelijkbaar)

Voor we naar de opdrachten gaan, zullen we de onderdelen - voor zo ver nodig - bespreken.
Digitale bestanden kunt u terugvinden op: https://github.com/jakorten/defusable_arduino_shield

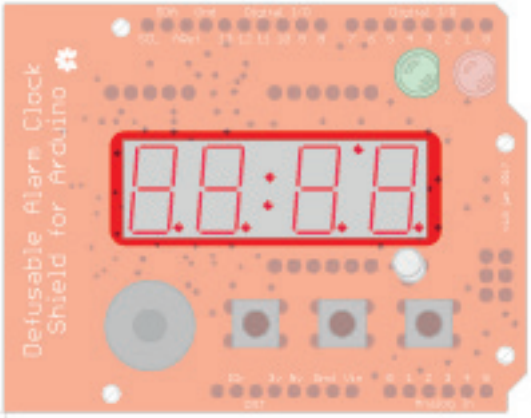
Voorbereiding: Installeer de "BombDisplay" map onder "Arduino" | "libraries" in uw lokale Arduino libraries map en herstart indien nodig de Arduino omgeving.

1. Segment display

Ik heb voor u een segment display uitgekozen waar u alle kanten mee uit kunt: behalve de vier cijfers heeft u een dubbele punt, vier decimale punten en één 'apostrof' tot uw beschikking (zie figuur 2).

Segment displays bestaan eigenlijk gewoon uit een aantal LED's welke op een 'leesbare' manier ingedeeld zijn. Voor zo ver u weinig Natuurkundige kennis heeft is het wel prettig om te weten dat een LED (light emitting diode) een diode

is met de 'merkwaardige eigenschap' dat deze in doorgaande richting licht geeft en in sperrende richting uit staat. Een diode heeft een kathode (de negatieve / GND kant) en een anode (de positieve kant, bijv. 5V). Een segment display is op zo'n manier in elkaar gezet dat de LED's gemultiplext worden: u heeft niet zo veel pinnen nodig als er LED's zijn. Dat is maar goed ook want ons display heeft 28 LED's voor de cijfers plus 7 LED's voor de punten. Toch heeft deze 'slechts' 16 in plaats van 35 pinnen nodig om aangestuurd te worden (hoewel de dubbele punt altijd als paar aangestuurd wordt). De truc is dat bij het display afhankelijk van de variant de anodes (common anode) dan wel de kathodes (common cathode) per segment aan elkaar verbonden zijn en daarnaast de segment-onderdelen ook met elkaar verbonden zijn. Door selectief de common anode/cathode en het betreffende segment onderdeel (bijv. onderste streep voor "_" aan te zetten verschijnt het juiste teken.

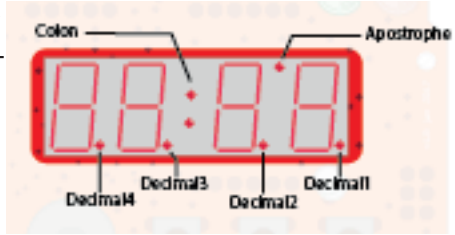


Figuur 2. Zeven segment display.

Vaak worden dergelijke segmenten aangestuurd via een driver IC: deze zorgt ervoor dat diverse functies via nog minder pinnen aangestuurd kunnen worden: zoals u wellicht weet komt u vrijwel altijd pinnen te kort in de embedded informatica wereld. Stel dat wij met een UNO direct zo'n display zouden willen aansturen hadden we voor alle functies van ons shield 16 + 6 = 22 pinnen. De Atmega328 van de UNO heeft (afhankelijk van het exacte model) slechts 19 pinnen dus dan zouden we zonder meer al twee pinnen tekort komen. Er is nog een probleem: de Atmega328 heeft ook slechts 32Kb flashgeheugen (en 2Kb werkgeheugen). Dat lijkt vrij veel, maar als we allerlei voorgeprogrammeerde tekens willen opslaan, zit je maar zo aan de limieten van dat geheugen vast.
Ons shield gebruikt hiervoor de HT16K33 als multiplexer IC. De HT16K33 werkt via i2c en neemt dus eigenlijk geen extra pinnen van de Arduino UNO in beslag met uitzondering van de twee i2c pinnen.

Om meer complexe electronica-onderdelen te begrijpen is het prettig de datasheet van het onderdeel te kunnen

raadplegen (hoewel afhankelijk van de leverancier deze meer of minder leesbaar zijn). Sparkfun.com is een bekende naam in de Maker community en een indicatie dat een onderdeel prettiger in het gebruik is dat deze door hen ook toegepast wordt: https://www.sparkfun.com/products/11441 hier vindt u naast code ook extra uitleg over het onderdeel. Er
De door ons gebruikte display (figuur 3) heeft vier cijfers (van zeven segmenten), vier decimale punten, een apostrof en een dubbele punt. Deze items zijn allemaal aan te sturen door binaire reeksen naar de HT16K33 te sturen. De HT16K33 zet vervolgens bijbehorende pinnen aan en uit, zie verder H8 (HT16K33) en H13 (programmeren van de display).



Figuur 3. Onderdelen van de display.

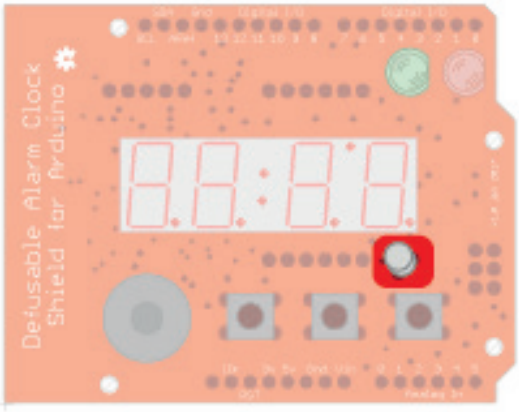
2. Real-Time Clock (RTC)

De Arduino heeft (uiteraard) een ingebouwde klok welke meestal op 16Mhz (of anders 8Mhz) draait. De UNO maakt hiervoor gebruik van een (extern) crystal welke het kloksignaal produceert. Dit is op zich meestal voldoende, maar soms willen we meer: als we werkelijke tijd bij willen houden voldoen de interne Timers qua betrouwbaarheid en persistentie niet meer. Als we dit shield alleen voor aftellen (bijv. timer t.b.v. lanceerinrichting van een modelraket of tijd tot explosie van onze 'bom') willen gebruiken is de interne klok op zich voldoende, maar de LED display kan veel meer en het zou jammer zijn als we daar geen gebruik van zouden kunnen maken.
We hebben het shield daarom voorzien van een RTC chip (DS3231) en een backup-batterijtje (zodat de klok nadat deze ingesteld is zijn tijd onthoudt ook als de controllers geen spanning hebben). We hebben linksonder een header geplaatst, als deze kortgesloten wordt (bijv. met een 'bom-draadje'), reset de RTC zich.

N.b. de meeste ARM processoren (waaronder de SAMD21, te zien als opvolger van de Atmega328 in Arduino context) hebben wel een Real-time klok aan boord en hebben hooguit een extra kristal nodig maar geen aparte RTC chip.

3. Tilt sensor

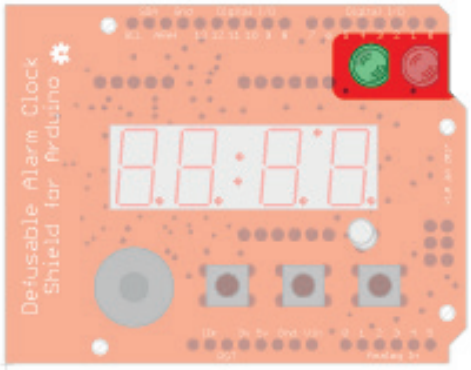
We kunnen ons voorstellen dat u soms een object wilt beveiligen tegen collega's en/of leerlingen of het 'ontmantelen' van de 'bom' lastiger wilt maken: we hebben daarom een tilt sensor toegevoegd (figuur 4). Dit is een zeer eenvoudige sensor met klein balletje welke kortsluiting maakt als de sensor in een bepaalde stand komt. Afhankelijk van of er wel of geen kortsluiting is weet u dat uw 'bomcontroller' al dat niet bewogen is ten opzichte van de beginsituatie. De tilt sensor zit op D16 en heeft een pull-up weerstand (zie 7.1 voor uitleg over pull-ups).



Figuur 4. Tilt-sensor.

4. Twee indicatie LED's

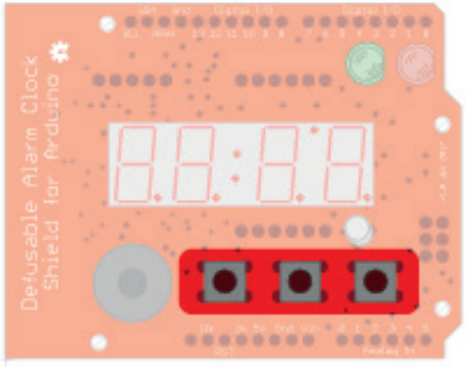
Vrijwel elke Arduino heeft op digitale Pin 13 een te programmeren indicatie LED. We hebben deze op het shield ook doorverbonden. Voor een meer authentieke uitstraling hebben we er voor gekozen een 5mm 'klassieke' LED hiervoor te gebruiken (groen, zie figuur 5). Daarnaast hebben we een tweede LED (A3 == D17) geplaatst (rood, zie figuur 5). U zou deze kunnen gebruiken om te laten zien dat de 'bom' veilig/ontmanteld is of juist 'armed and dangerous'.



Figuur 5. Indicatie LED's.

5. Drie drukknopjes

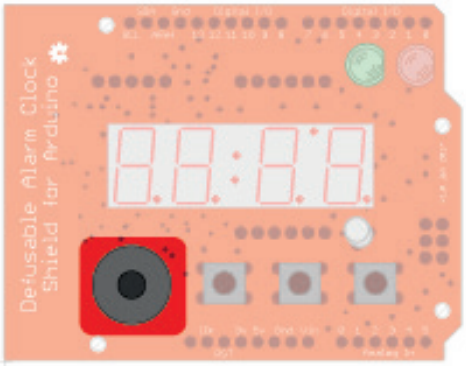
We hebben drie drukknopjes (van het type welke ooit massaal in muizen te vinden was) geplaatst op pinnen D2 (meest rechtse knop A), D3 (middelelste knop B) en D4 (linker knop C). U kunt deze vrij programmeren, bijv. om de klok in te kunnen stellen, een alarm in te kunnen stellen of de 'bom' zogenaamd 'op scherp' te zetten. In figuur 6 ziet u de drie drukknoppen.
De drukknoppen zijn aangesloten middels een pull-down weerstand zodat "HIGH" gelezen wordt als de knop gesloten is en "LOW" als deze losgelaten wordt (zie 7.1 voor uitleg over pull-ups).



Figuur 6. Drukknoppen.

6. Buzzer

U zou de 'bom' als wekker kunnen gebruiken, ook zou u het effect van 'bom' nog wat realistischer kunnen maken door piepjes toe te voegen. Hiervoor hebben we een eenvoudige Piezo-buzzer (figuur 7) toegevoegd op hardware PWM ondersteunende pin D9.



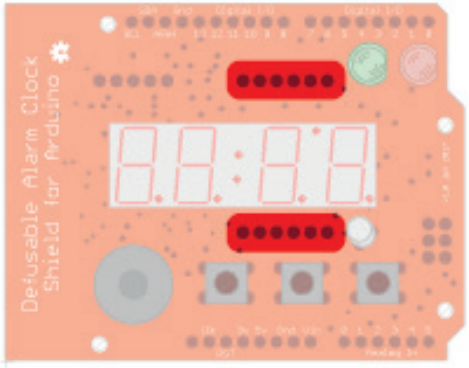
Figuur 7. Buzzer.

7. Bomdraden

Een 'bom' is natuurlijk niets zonder kleurrijke draadjes (zie figuur 1) die dan één-voor-één doorgeknipt dienen te worden om de 'bom' te ontmantelen. Uiteraard zijn sommige van deze draden 'valdraden' waardoor de tijd opeens

een stuk sneller gaat lopen of de 'bom' spontaan 'ontploft' doordat de tijd opeens naar nul schiet. (Dat laten we over aan uw eigen creatitiveit).

Ik heb voor u zes pinnen (zie figuur 8a en 8b) gemaakt waarmee u dergelijke bomdraden kunt simuleren. In embedded systemen werken we vaak met zogenaamde pull-up en pull-down schakelingen. De 'bomdraden' zorgen ervoor dat u beter inzicht krijgt in dir principe. Drie draden maken gebruik van pull-up's en drie van pull-down's. Uiteindelijk bepaalt u de werking zelf met uw software natuurlijk. Meer hierover verderop.

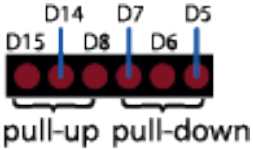


Figuur 8a. Pinnen voor bomdraden.

7.1 Pinnen van de Bomdraden

De pinnen voor de bomdraden zijn zo aangesloten dat u van hartelust kunt experimenteren met het zogenaamde pull-up en pull-down principe.

In figuur 9 is schematisch te zien hoe 'pullen' werkt: als we een digitale pin in Arduino door middel van pinMode(pin#, INPUT) als input definiëren, wordt het interne register van de pin op input gezet. Als we ca. 5V op de pin zetten en dan digitalRead(pin#) uitvoeren, zal deze een HIGH (in C gelijk aan 1 en TRUE) teruggeven. Als we er ca. 0V (GND) op zetten zal dezelfde aanroep een LOW opleveren (in C gelijk aan 0 en FALSE). Helaas hebben inputs van digitale controllers vrijwel altijd last van een vervelend fenomeen: 'floaten': als ze niet actief richting 0V dan wel 5V (op de UNO, kan ook bijv. 3.3V zijn) getrokken worden, wordt er een ongedefinieërde waarde gemeten omdat de pin gaat zweven tussen 0V en 5V.
Om dit op te lossen gaan we dus 'pullen': als we de B pin op 5V aansluiten,



Figuur 8b. Bomdraden: ingezoomd.

dan sluiten we de leespin middels een weerstand aan op de A pin en verbinden die met GND. In dat geval lezen we standaard een HIGH, als de bommenexpert de draad doorknipt lezen we een 0 (GND). N.b. De weerstand voorkomt ongewenste kortsluiting tussen 5V en GND. We kunnen het ook anders aansluiten: Supply B aan GND en dan de leespin via de weerstand naar 5V trekken (pull-up). Zoals te zien in figuur 6b worden D5, D6 en D7 naar beneden getrokken als er geen draad verbonden is naar 5V. D7, D14 en D15 worden standaard naar 5V getrokken tenzij er een draad naar GND verbonden wordt.

Figuur 9. 'Pullen' van bomdraad.

We gaven bij de tilt sensor aan dat deze middels een pull-up weerstand met D16 verbonden is: dit houdt in dat als het balletje in de sensor 'vrij' is, er een HIGH gelezen wordt en bij kortsluiting een LOW (uiteraard treedt er geen kortsluiting op zoals hoe dit in de volksmond gebruikt wordt (waarom niet?!)).

Tip! Voor meer achtergrond over pull-up en pull-down circuits en de formule voor het uitrekenen van de weerstandswaarde: https://learn.sparkfun.com/tutorials/pull-up-resistors.

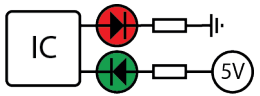
8. De HT16K33 voor Segment aansturing

De HT16K33 is een universele chip, vooral bedoeld voor het aansturen van grotere hoeveelheden LED's, maar ook geschikt om bijv. een klein toetsenbordje aan te sturen.

LED's kunnen op twee manieren aangesloten worden op een aansturingschip: men kan ze sourcen of sinken (zie figuur 6). Bij "sourcen" loopt de stroomrichting vanuit de IC, bij "sinken" loopt de stroomrichting juist naar de IC toe.

Als men een digitale pin gebruikt zoals bij de rode LED, moet het IC (de microcontroller of in dit geval de HT16K33) de stroom leveren om de LED te laten branden. Bij de groene LED levert de stroomvoorziening van in ons geval de UNO de "plus" en heeft de IC slechts de taak de LED te 'sinken' als deze aangezet wordt.

De meeste chips zijn beter in "sinken" dan "sourcen", zo ook onze HT16K33. De (common) source current ligt rond de 25 mA en de (common) sink current rond de 200 mA. Dit heeft wel gevolgen voor de keuze van een LED segment display: in H1 bespraken we kort dat zo'n display een gezamelijke kant heeft, ofwel common anode, ofwel common kathode. Per gebruikt de display ongeveer 8*20mA = ~120mA dus 'sinken' is wel nodig: om de HT16K33 de LED's te laten 'sinken' is er een common kathode display nodig. Chips zoals de HT16K33 gebruikt een truc te zorgen dat zo veel LED's toch relatief weinig stroom verbruiken, ze zetten ze heel snel aan en uit met behulp van pulse-breedte modulatie. De afzonderdelijke LED's worden heel snel achter elkaar (met meerdere kiloherzen) aan en uit gezet. Ons oog ziet dat pas rond de 20Hz. Deze truc wordt direct ook gebruikt om de LED's te kunnen dimmen: door de helft van de tijd de LED uit te laten lijkt deze maar op 'halve kracht' te branden.



Figuur 6. Sourcen vs. Sinken

9. Communicatie met HT16K33 en RTC via i2c

Voor de communicatie tussen zowel de Arduino UNO en het Display (de HT16K33 voor de 7-segmenten LED display) én de real-time klok chip gebruiken we het i2c protocol. De UNO heeft op de analoge pinnen A4 en A5 een extra functie waardoor deze respectievelijk als data en klok pin voor i2c gaan werken. Het idee aan i2c is dat je één draad gebruikt voor het klok signaal en de tweede draad voor de data. Een i2c verbinding wordt daarom ook wel two-wire interface genoemd.

Meestal is i2c master-slave georiënteerd, hoewel een multi-master implementatie mogelijk is. Een sensor of in ons geval RTC-chip is altijd een slave en een microcontroller zoals die van de UNO is dan de master.

De Atmega328 kunnen we instellen als master dan wel slave. De Arduino software heeft hele goede voorbeelden voor zowel master (reader én writer voorbeeld) als slave (ook een reader én writer voorbeeld). Wat helaas mist is een zogenaamde "i2c scanner": code is beschikbaar op de github van dit project (https://github.com/jakorten/defusable_arduino_shield). De scanner loopt alle beschikbare i2c adressen af en rapporteert eventueel gevonden i2c slaves. Als het goed is detecteert u met de i2c scanner zowel de RTC chip als de HT16K33 (deze zit standaard op adres 0x73). Meer over de RTC chip vindt u onder "12. de RTC programmeren".

N.b. links onder op het shield (boven de USB poort van de UNO) hebben we een pinheader gemaakt: met een 'bomdraadje' kunt u de RTC chip resetten door deze twee pinnen kort te sluiten, dit kan soms handig zijn, zeker bij het testen.

Extra informatie:
https://learn.sparkfun.com/tutorials/i2c

https://github.com/sparkfun/Serial7SegmentDisplay

RTC chip informatie:
https://learn.adafruit.com/adafruit-ds3231-precision-rtc-breakout/overview

10. Vrije pinnen

Het is altijd prettig om te weten welke pinnen een shield gebruikt zodat u hier rekening mee kunt houden voor het geval u nog andere uitbreidingen wilt gebruiken. Dit shield is vrij 'greedy' hoewel niet alle pinnen bezet zijn.

Het shield laat de volgende pinnen van de UNO vrij:
- 0 en 1 (standaard seriële RX en TX pinnen van uw UNO processor)
- D10 .. D13
- De speciale pinnen (RST, IORef, ARef, Reserved)
- De 3.3V en VIN worden niet gebruikt de 5V en GND zijn doorverbonden naar het shield

11. Gebruikte Pinnen

Pin	Digitaal / Analoog	Functie	Bijzonderheden
D2	Digitaal	Drukknop A	pull-down weerstand
D3	Digitaal	Drukknop B	pull-down weerstand
D4	Digitaal	Drukknop C	pull-down weerstand
D5	Digitaal	Pinheader voor Bomdraad	pull-down
D6	Digitaal	Pinheader voor Bomdraad	pull-down
D7	Digitaal	Pinheader voor Bomdraad	pull-down
D8	Digitaal	Pinheader voor Bomdraad	pull-up
D9	Digitaal	Buzzer	PWM pin
D13	Digitaal	Groene LED	
D14	Digitaal (op Analoge pin A0)	Pinheader voor Bomdraad	pull-up
D15	Digitaal (op Analoge pin A1)	Pinheader voor Bomdraad	pull-up
D17	Digitaal (op Analoge pin A3)	Rode LED	
A4/SDA	Analoog (i2c)	i2c Data pin	pull-up
A5/SCL	Analoog (i2c)	i2c Clock pin	pull-up

Tabel 1. Overzicht van door het Shield gebruikte pinnen van de Arduino UNO.

In tabel 1 is een overzicht te vinden van de door het Defusable Alarm Clock shield gebruikte pinnen. Op de i2c pinnen na zijn de pinnen uit het overzicht dus niet meer in te zetten. Uitzondering hierop is pin 13, deze is standaard ook al van een LED voorzien en nog steeds als SCK pin te gebruiken. De overige SPI pinnen zijn bewust leeg gelaten zodat de ISP / SPI header gebruikt kan worden om het shield te herprogrammeren.

12. De RTC Chip programmeren

Om de (Real-Time Clock) RTC-chip DS3231 te kunnen gebruiken heeft men de RTClib library nodig. Deze heeft ook een voorbeeld-sketch.

Tip! Het is altijd verstandig om als u op Arduino aan het programmeren bent om onderdelen en/of functies eerst onafhankelijk van elkaar te testen en te programmeren.

Library installeren:
1. Download de Adafruit's RTCLib van: https://github.com/adafruit/RTClib/archive/master.zip
2. Ga in uw Documenten map naar de Arduino/libraries/ en pak master.zip uit.
3. Herstart uw Arduino IDE en open het voorbeeld onder Voorbeelden | RTClib | ds3231

Upload de sketch naar uw Arduino, open de Terminal en controleer de output.
Bovenstaande code zit (ongeveer) in de voorbeeld sketch, daarmee zet u de RTC op een gewenste tijd. Als u de RTC reset door een 'bomdraadje' op de twee reset pootjes te steken en na paar secondes weer te verwijderen zou u in uw Terminal aan de output moeten zien dat de RTC daadwerkelijk opnieuw ingesteld is.

Zeker als u het shield gebruikt al wekker of timer, is het prettig als u de tijd niet steeds opnieuw hoeft in te stellen. Het shield heeft daarom een houdertje voor een CR1220 lithium batterij. Deze onthoudt de eerder door u ingestelde tijd, ook als de rest van het shield en de Arduino UNO geen stroom meer krijgen/leveren.

Tip! Bekijk ook hoe de sketch op een slimme manier de datum/tijd van uw computer en Arduino synchroniseert.

BombClock heeft een voorbeeld van hoe u de Realtime klok samen met het segment display gebruikt.

13. Segment Display Aansturen

We hebben al besproken dat segment display via de HT16K33 chip aangestuurd wordt. Hoe dit daadwerkelijk gaat bespreken we hier. Met deze informatie kunt u uw eigen library voor het besturen van de display maken.

Stap 1. Oscillator van de HT16K33 activeren

```
Wire.beginTransmission(i2c_addr); // i2c_addr is het adres van de chip 0x73
Wire.write(0x21); // oscillator activeren, zie pagina 30 datasheet
Wire.endTransmission();
```

Stap 2. Berichten sturen naar de HT16K33

```
Wire.beginTransmission(i2c_addr);
Wire.write((uint8_t)0x00); // we sturen eerst een $00 zie pagina 30 datasheet

... // daadwerkelijke bericht hier

Wire.endTransmission();
```

Stap 3a. Daadwerkelijke bericht ‘invullen’

```
// het teken bestaat uit twee dubbele bytes
// op ons bordje zijn de eerste drie tekens niet aangesloten
// dus eerst zes keer een 0x00 en dan het gewenste teken:
0b0000 0111 1110 0000 // dit is de dubbele byte voor een ‘0’
```

Stap 3b. Tekens sturen

```
Wire.write(0b11100000); // eerst minst significante byte
Wire.write(0b00000111); // daarna meest significante byte
```

We hebben de complete code ook toegevoegd aan de BombDisplay voorbeelden als “Bomb_LCD_Direct.ino”.

14. Wrap-up

In de voorbeelden ziet u ook “MegaBomb”. Dit is een totaal programma voor uw Defusable Alarm Clock: u kunt hier een werkende ‘bom’ controller mee implementeren.

Tip! Om te zorgen dat u niet horende dol wordt kunt u met “useSoundFeedback” het geluid van de buzzer aan of uit zetten. We hebben deze standaard op false gezet (om familieproblemen te voorkomen ;)).

Tabblad “bombWiresHelper” bevat de code voor de bomdraden. Er staan in de methode “void checkWires()” enkele tips om dit te implementeren. Aan u de vrijheid uw “bom” zo te laten werken dat de het anti-explosieve arduino commando het niet te eenvoudig krijgt bij het ontmantelen van uw “Improvised Explosive Arduino”.

Ik hoop dat u genoten heeft van deze tutorial en er nog veel plezier in kader van uw lessen aan zult beleven.

Teacher's Edition

In de kit:
- Defusable Alarm Clock Shield voor Arduino

Zelf Toevoegen:
- Basis Arduino Kennis
- Computer met Arduino IDE (www.arduino.cc)
- Arduino UNO Rev 3. of compatible
- USB programmeerkabel

Optioneel:
- Batterijvoeding
- Karton van WC-rollen of Keukenpapier
- CR1220 RTC batterij

jakorten@jksoftedu.nl
www.JksoftEdu.nl

How to program your own defusable bomb controller...