rd

Inleiding

1. Wat is het fenomeen **microcontrollers?**
   Een microcontroller is een geïntegreerde schakeling die bestaat uit een microprocessor.
   Het wordt gebruikt om een elektronisch apparaat te besturen. Zo een microcontroller bestaat minimaal uit een centrale verwerking eenheid *(CVE)* of bij ons beter gekend als central processing unit (*CPU*), geheugen of memory en een Input en output systeem (*I/O*). Men kan zich heden ten dage geen toestel, groot of klien niet meer indenken dat geen gebruik maakt van microcontrollers.
2. Geschiedenis.
   om de geschiedenis van deze controllers te duiden moeten we terug gaan naar de medio/einde van de 20ste eeuw. Namelijk naar hert jaar 1971. Daar werd dan de eerste 4-bits controller uitgebracht namelijk de *intel 4004*. Later zullen we wel terug komen op wat een bit is, immers dat is een van de bouwstenen van de informatica. Niet veel later kamen er echter betere zoals de 8008 en hogere. Echter zij hadden allemaal 1 ding gemeen namelijk: ze hadden nood aan externe onderdelen of componenten om te kunnen functioneren.
3. Software.
   Het idee was nu dat men door deze controllers een machine bepaalde acties zouden laten uitvoeren, met andere woorden het feit is dat men repetitieve handelingen door machines laten uitvoeren (welke voordelen qua snelheid en zekerheid) . Door gebruik te maken van software kon de mens bepaalde zaken opleggen aan deze controller welke een besturing van het toestel mogelijk maken. Na verloop van tijd zijn er ook controllers uitgewerkt die met 8-16-32 en 64 bits werken.
4. Programma's
   Voordien hebben we gezien dat deze programma's werden (of worden) uitgevoerd met een herhalend karakter. Daarom is het dan ook nodig dat we het *programma* in een soort geheugen zetten welke we op geregelde tijden kunnen *aanspreken* . Dit geheugen bestaat uit een *intern* geheugen. Het probleem stelde zich nu dat we het programma in dit geheugen moeten stockeren, daardoor dat het enigszins beperkt is in grote. Geen nood er is een systeem gevonden dat men ook kan gebruik maken van externe geheugen. Door gebruik te maken van *compilers* en *assemblers* zal het ogenschijnlijk 'lang' programma in een compacte, door de controllers verstaanbare, vorm gegoten worden.
5. Interrupt of programma onderbreking
   Een controller kan gebruik maken van interrupt om *real-time* commando's op te nemen. Stel op en bepaald moment is de controller bezig om een commando uit te voeren. Door een interrupt kan deze uitvoering tijdelijk gestopt (in wacht geplaatst worden) en een andere commando door middel van een *interrupt handler* uitgevoerd worden. Eens deze nieuw handeling afgelopen zal de oorspronkelijke handeling terug vervolgd worden. Dit is in den beginnen een beetje hocus-pocus maat gaandeweg zal je de kracht van interrupt wel begrijpen.
6. Werking
   De controller zal de mogelijkheid hebben om binnen gekomen signalen via sensoren bepaalde handelingen uit te voeren, ze te registreren en in herhaling op het juiste moment uit te voeren. Iets dat je door de verschillende oefeningen te maken ook beter na verloop van tijd zal begrijpen, als je dan zelf de nodige programma's kunt ontwikkelen zal je zien dat er een nieuwe wereld van techniek voor u open gaat. Maar vergeet niet men ka niet leren lopen zonder te vallen!

Programmeren voor microcontrollers

1. In dit onderdeel gaan we trachten te duiden wat we bedoelen in met het programmeren van en microcontrollers. Het is duidelijk dat door het programmeren de uiteindelijke werking van de controllers zullen bepalen. Dit gebeurt door gebruik te maken vaan een programeer taal. Er bestaan verschillende. Meestal gebruiken we c (c++), Het is natuurlijk ook mogelijk met andere programmeertalen. Het enige wat we zeker nodig hebben is een *IDE*(integrated development environment) die een *PIC* (programmable interrupt Controller) aanstuurt.
2. Er bestaat een speciale programmering set voor **Arduino** welke in eerste instantie zullen gebruiken. Naderhand zullen we overschakelen naar een andere taal maar de principes blijven hetzelfde. Het software programma (gratis te downloaden) is een gemakkelijke *step-up* om te leren programmeren en kan men vinden op de Arduino website(*www.Arduino.org*). Men kan deze ophalen voor Microsoft- of Linux based computers. Ook de voor de Apple liefhebbers is er een mogelijkheid.
3. In het programmeren is er steeds een bepaalde structuur te volgen. Namelijk de *top-down* methode. Het is te zeggen men moet steeds van boven (input) naar onder(output) werken. Met de moderne systemen OOP (*object Oriented Programming)* welke we nu gebruiken telt dat ogenschijnlijk niet meer. Maar daar gaan we niet verder op in. Wij moeten, althans, nu nog dat systeem gebruiken. Dus vergeet niet een top-down methode is hier aangewezen. Wij zullen dus beginnen met Arduino sketch.
4. Na al die theorie kunnen we nu overgaan naar het programmeren zelf! Zoals reeds gemeld gaan we in den beginne gebruik maken van het programmering systeem van Arduino zelf. Een programma in Arduino noemt een *sketch* . Dit programma wordt samen gesteld uit verschillende commando's, elk met een bepaalde eigenschap. Een samenstelling van deze commando's zal uiteindelijk de gevraagde functie genereren waarbij we het gevraagde onderdeel kunnen besturen. Op ed volgende website kan men al de commando's vinden. De voertaal is wel Engels (zoals in de meeste programmeertalen). https://www.arduino.cc/reference/en/
5. We weten nu onze commando's of althans we weten nu waar we ze kunnen vinden dus het wordt tijd dat we eens een programma schrijven. Immers we zullen het beter begrijpen als we het zelf eens proberen.
   Vb: als voorbeeld gaan we eens een LED laten knipperen, gebruik makend van een Arduino programma. Wat een LED is kom ik later op terug.

rd

```
void setup() {
4     // initialize digital pin LED_BUILTIN as an output.
5     pinMode(LED_BUILTIN, OUTPUT);
6 }
7
8 // the loop function runs over and over again forever
9 void loop() {
10     digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage
level)
11     delay(1000);                       // wait for a second
12     digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the
voltage LOW
13     delay(1000);                       // wait for a second
14 }
```

6. de software

Hierboven staat een klein programaatje die tot doel heeft een LED (light Emitting Diode). Zoals voordien gezegd zal ik daarover later verder over uitweiden. Laat ons nu eens het programma onder handen nemen.

6.1 We beginnen met de uitdrukking void setup() {

void setup(). Dit is een functie. Een functie is als het volgt samen gesteld input variabele functie (output variabele). In dit geval wordt dat dan void betekend ledig of niets, de functie met naam *setup* is een functie om iets klaar te zetten aan het begin van een programma of programma onderdeel, tussen ronde haakjes wet de functie zal geven als uitgang. Dit kan in het begin raar lijken daarom ga ik een kleine sprong maken naar de grote broer van informatica namelijk wiskunde. Daar in kennen we ook het fenomeen functie neem nu de functie optellen. Deze kunnen we schrijven als: getal som(3,5). Dit geeft als uitwerking 3+5 = 8. Waarin: som is een functie met als waarden a+b, welke een uitslag c geeft. Dus ruwgenomen a = 3, b = 5 en getal = c. dus de uitkomst van de functie met naam som is 3+5 (of het getal 8. Verder gaan we nog meerdere functies zien, alsook de kracht en eenvoud in gebruik ervan. Maar dat is voor later! Twee zaken waar we nog rekening moeten me houden is: een functie zal steeds afgesloten worden met 2 ronde haakjes **()** waar niet altijd iets moet tussen staan. El als meest belangrijke punt is de functie zelf staat tussen twee accolades, een in het begin van de functie en 1 op het einde **{ }.**

6.2 het eerste wat we tegenkomen in deze functie is **// .** Deze twee schuine streepjes zijn een sterk onderdeel in het programmeren. Namelijk deze duiden de mogelijkheid aan voor het achterlaten van commentaar. Dat is iets waar de compiler dus geen rekening zal mee houden, doch voor de programmeur kan dit in latere acties van groot belang zijn. Immers met dit kan de reden van gebruik van d functie aangegeven worden. // geeft de mogelijkheid van 1 regel commentaar toe te voegen. Daartegen kan men door volgende methode */……*/ tussen deze twee uitdrukkingen een ganse blok commentaar toevoegen.

6.3 Het volgende is de functie pinMode(). Let wel aangezien we geen input variabele hebben is deze functie enigszins verschillend van de vorige en heeft ze enkel tussen de obligatoire haakjes de

uitgang variabelen LED_BULLETIN en OUTPUT, gescheiden door een komma. Merk op dat ieder commando afgeslote wordt door een punt komma **;.** Dit is zeer belangrijk en veelal een voortbrenger van fouten!

Zoals gezegd het geheel van de functie wordt afgesloten met een sluit accolade.

6.4 het volgende is de functie loop(). Deze functie zal zorgen voor een constante herhaling van wat er in zijn *body* staat.

De functie digitalWrite(LED_BULLETIN, HIGH) heft als doel de spanning toe te laten aan de LED waardoor het zal oplichten. De aansluitende functie delay(1000) zal zorgen dat dit blijft voor 1 sec. of 1000 ms. *Delay functie wordt aangegeven in ms*. Het zelfde zal dan gebeuren als de LED uitgeschakeld wordt. Welke dan het knipper effect geeft.

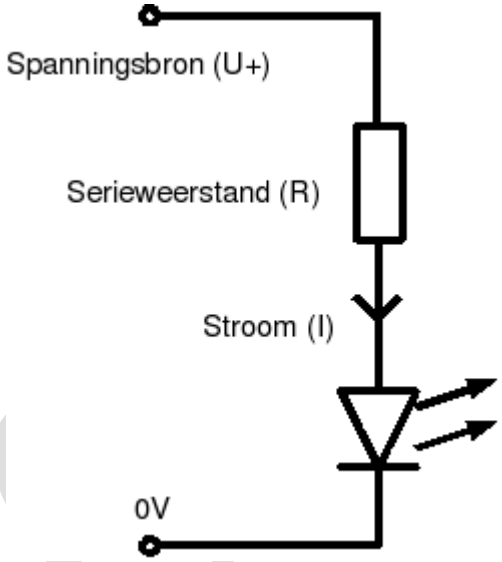
6.5 en dan als laatste hebben we de afsluitende accolade

het enige dat ons dan nog rest is dat op te laden naar onze controller en proberen!

7. Natuurlijk hebben we nu *de hardware* nodig. Of wel de LED. Ik zal nu een beetje uitleg geven wat een LED is.

   L.E.D. of Light Emitting Diode is een elektronisch onderdeel die licht kan afgeven, bij lage spanning en die niet de nadelen heeft van een gloeidraad lamp of gasbuis lamp. Met andere woorden eens spanning geeft die onmiddellijk licht.

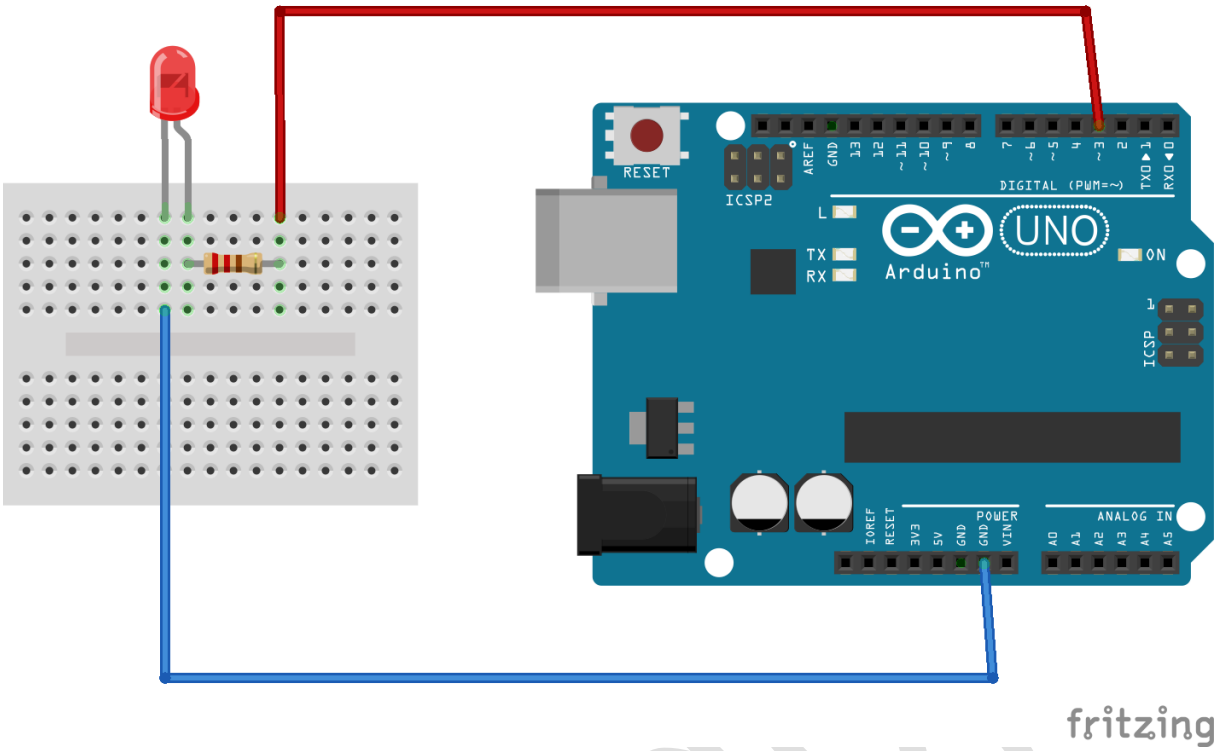
   Een LED werkt steeds gebruik makend van een voorschakel weerstand. Deze weerstand zal zorgen dat er een elektrische stroom door de schakeling vloeit en zo het licht kan produceren.



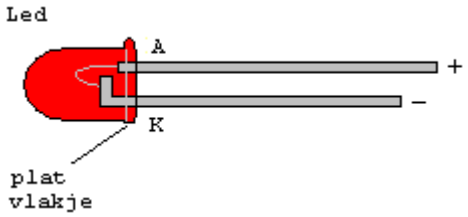
   In bovenstaande tekening ziet men hoe een voorschakel weerstand aangesloten wordt en zijn functie heeft in de schakeling. Volgens de wet van Ohm, kunnen we deze weerstand berekenen. Immers bij een spanning van U(V) gaat er een stroom (I) door de weestand (R). We weten uit de gegevens dat de stroom welke door een LED gaat ongeveer 20 mA moet hebben van waarde. We weten tevens dat we de uitgangspanning gelijk is aan 5V (Arduino). We weten tevens dat volgens de data van een led zijn verbruik ongeveer 1,9 V is. Uit deze gegevens volgt dan: r een minimum waarde van 155 ohm moet hebben. Deze is niet in de courante handel te verkrijgen en we nemen dus 188 Ohm (let wel nooit lager dan 155 ohm). Deze zou berekend dan een stroom leveren van 17.22 ma, welke voldoende is want de volledige sterkte van een LED zal je bijna nooit verkrijgen. Let wel wees steeds bedachtzaam dat je de stroom door de LED NOOIT hoger neemt dan 20 mA, anders is hij een kort maar krachtig leven beschoren!

8. Met deze kennis kunnen we onze schakeling samenstellen, volgens het schema (hier onder)

rd



8.1 hier staat de volledige schakeling uitgebeeld. Laat ons deze eens uitpluizen. Links hebben we een breadboard (naam voor een systeem om gemakkelijk verbindingen te maken). Daarop een LED (rood of een ander kleur. Laat ons eerst even naar de aansluitingen van een LED bekijken.



Bij bovenstaande tekening ziet men hoe een LED in feite er uit ziet. We zien dat we daar een Kathode (-) hebben en een Anode (+). Deze anode gaan we aansluiten aan een weerstand (220 ohm), aangegeven in de internationale kleurcode. Ik hoor u al zeggen 'jamaar we hadden toch 155 berekend?' daarin hebt je gelijk maar er werd ook benadrukt dat de weerstand niet minder dan 155 ohm mocht zijn. In de elektronica is 220 ohm de meest courante waarde (188 is een duurdere uitvoering en niet zo courant te vinden.

De andere kant van de weerstand wordt verbonden met de rode draad aan onze microcontroller op het Arduino bord. Dit via de klem 3 van de digitale output.

De kathode (andere kant van de LED) wordt verbonden via de blauwe draad naar de grond klem van dat zelfde bord.

8.2 we open onze programmeer blad en programmeren bovenstaande code. Opgelet voor dat we onze setup programmeren moeten we eerst ons Arduino bord wijsmaken dat we een LED aangesloten hebben en wel op pin 3. Zouden we dat niet doen zullen we een foutmelding krijgen. Immers dan heeft de controller geen weet waar hij iets moet aansturen. (EXIT Staus_1). Eens dat gedaan kunnen we de rest van de code overnemen. Er zijn een paar zaken waar je zeker moet op letten om geen fouten te generen. Eerst en zeer belangrijk: na iedere volledige lijn code

komt er een *punt-komma.* Vergeet u deze dan weet de processor niet dat die code lijn beëindigd is en leest hij gewoon de volgende als vervolg in, met een fout als gevolg! Ook iets dat misschien handig is, een commando die de processor moet uitvoeren wordt veelal in hoofdletters geschreven. Vergeet zeker de accolades niet!
Bij het programmeren heb je ook verschillende zaken die best zult onthouden als je ze gebruikt. De belangrijkste echter zijn variabelen. Deze worden meestal voorafgegaan door een ordegrootte zoals integer string…. . Maar daar komen we later op terug.

8.3 Een maal de code geschreven in onze programmeer omgeving hebben we twee mogelijkheden, of wel laden we ze op naar het bord wordt dan automatisch gecompileerd (in machinetaal omvormt). Ofwel gaan we ze eerst compileren en verifiëren en laden ze dan op. Dit is een betere methode, toch in het begin. Als we dat zo doen zullen we eerst de fouten kunnen uithalen en dan een juiste gecompileerde code naar ons bord sturen.
Dze methoden zien we in onze programeer omgeving onder de TAB schets.

8.4 laten we dat eens proberen en wonder boven wonder ons programma werkt en onze LED blinkt. Nu kunnen we er een beetje mee spelen en bijvoorbeeld de wachttijd van uit veranderen naar 500ms, terug compileren en opladen en kijken wat het geeft. Wat merken we op?..... juist altijd als je iets verander moet je her compileren en terug opladen. Dat is logisch want de bestaande code wordt herhaald en zolang geen commando om het te stoppen blijft het in het geheugen zitten.

9. We hebben nu onze eerste kleine stapjes gezet in Arduino of liever sturen van een processor. Natuurlijk blijft het niet hierbij en gaan we verder op dit gegeven, met moeilijker en zelfs mooier programmeren en sturingen uitwerken.
Op naar devolgende!