# 9.0. Arduino Programmeren voor Beginners – Deel 9: Tekst Invoer


We hebben in de laatste 8 lessen veel gezien en geleerd, hoop ik. In dit laatste deel van onze kleine Arduino Programmeren cursus, we zijn al bij les 9, gaan we kijken hoe we via de Seriële Monitor gegevens kunnen laten invoeren door de gebruikers van ons programma, voor zowel tekst als nummers.

---

# Inhoudsopgave

9.0. Arduino Programmeren voor Beginners – Deel 9: Tekst Invoer ..... 1
9.1. Invoer via de Seriële Monitor ..... 3
9.2. Lezen van een String van de Seriële Monitor ..... 4
9.3. De eenvoudige String naar Nummer Conversie ..... 9
9.4. Andere String object methoden (functies) ..... 12
9.5. Laatste gedachten ..... 13

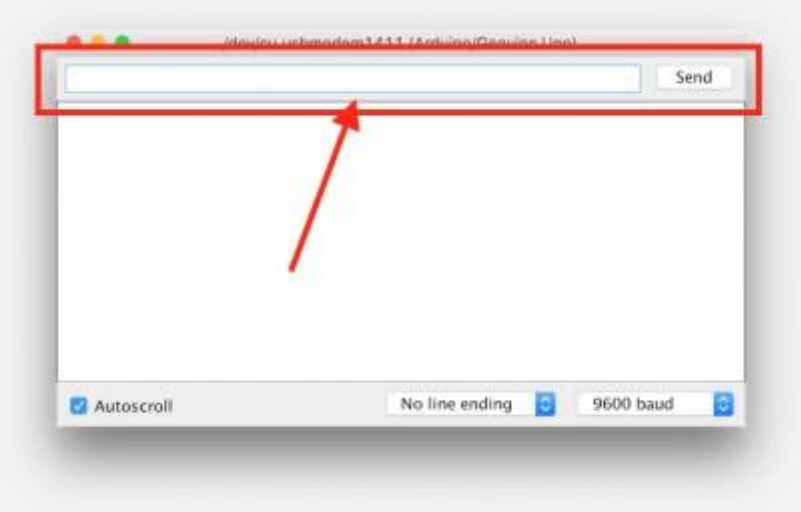
## 9.1. Invoer via de Seriële Monitor

We hebben de Seriële Monitor al veel gebruikt om te kijken wat de output (uitvoer) van onze programma's was. Maar ja, daar hebben we niet veel aan als het programma statisch is en we dus zelf geen gegevens kunnen invoeren. Wat hebben we immers aan een rekenmachine als we geen berekeningen kunnen opgeven?

De reden waarom ik dat deel heb overgeslagen is dat dit helaas wat voorkennis vereist, omdat we tekst (en nummers) teken voor teken moeten gaan inlezen. Beetje maffe aanpak als je het mij vraagt, maar ja, aan de andere kant was de Arudino nooit ontworpen met de gedachte dat een beeldscherm en een monitor aangesloten zouden worden.

De Arduino is niet ontworpen met toetsenbord en beeldscherm in gedachten, daarom zijn simpele functies zoals invoer lezen en uitvoer weergeven wat lastiger.

De Seriële Monitor heeft echter wel een manier om invoer (input) terug te sturen naar de Arduino, wat de Arduino dus weer kan lezen en kan gebruiken in ons programma. Hieronder zie je een afbeelding waar we deze invoer kunnen doen. Hier typen we onze tekst en klikken we vervolgens op de "Send" knop om tekst naar de Arduino te sturen.




*Figuur 9 - 1 Arduino Seriële Monitor – Tekst invoer*

Tip: Herstart jouw Arduino Programma m.b.v. de Seriële Monitor

Elke keer als we de Seriële Monitor openen, zal de Arduino herstarten en de het scherm leeg maken, zodat we een nette en schone output (uitvoer) van ons programma kunnen zien.

Je kunt dit dus ook gebruiken om een programma opnieuw te starten: Sluit de monitor en open de monitor opnieuw en jouw programma start netjes opnieuw op. Ideaal als je wat herhaaldelijk wilt testen.

Herstart een programma: Sluit en Heropen de Seriële Monitor

## 9.2. Lezen van een String van de Seriële Monitor

Laten we even vlug nog eens door het basis stappen voor het werken met de Arduino lopen.

Wanneer de Arduino start (door het te verbinden met de USB-aansluiting of via een externe stroom adapter), dan start het meteen het programma dat op dat moment in het geheugen van de Arduino zit. Dit geheugen, in tegenstelling tot het geheugen van b.v. een PC, wordt NIET leeg gemaakt als we de stroom van de Arduino halen.

We weten dat de Arduino vervolgens eerst een enkele keer de “setup()” functie gaat uitvoeren en als dat gedaan is, eindeloos “loop()” blijft herhalen tot we de stroombron dus van de Arduino loskoppelen.

Dit is zeer logisch, want de Arduino (of beter: de Microcontroller) is speciaal bedacht om een programma te draaien zodra het aangaat, waarbij het dus de voorbedachte of taken gaat uitvoeren. Dit kan het lezen van sensoren zijn, aanzetten van motortjes, maar dus ook het lezen van de seriële poort, en dat laatste is waar we onze kleine nieuwe functie voor gaan gebruiken.

Meer geavanceerde methoden om informatie van de seriële poort te lezen, omvat het inlezen van individuele bytes, welke gezien kunnen worden als het lezen van individuele sensors of schakelaars (toets-aanslagen).

Ik heb deze kleine functie alleen maar in elkaar gezet voor deze kleine cursus.
Via de seriële monitor geeft het een vraag weer, en wacht tot de gebruiker iets intypt en met een druk op de ENTER-toets de invoer afsluit.

```
1  String WachtOpInvoer(String Vraag) {
2    Serial.println(Vraag);
3
4    while(!Serial.available()) {
5      // wacht op input
6    }
7
8    return Serial.readStringUntil(10);
9  }
```

We hebben al eerder met functies gewerkt, en we zijn dus in staat zelf functies te maken. Laten we daarom even snel door de voorgaande code lopen om te kijken hoe en wat we doen in deze functie.

Als eerste geven we de functie dus een naam “WachtOpInvoer” welke een “String” teruggeeft – vergeet dus niet dat dit het object String is, met een hoofdletter “S” – en de vraag als parameter aanneemt. De “vraag” parameter is ook een “String” object. Het antwoord, een String, is wat de gebruiker heeft ingetypt.

Dit wil dus zeggen dat we de functie als volgt kunnen aanroepen:  Antwoord = WachtOpInvoer("Mijn vraag");

Omdat we de vraag leesbaar voor de gebruiker willen hebben, sturen we de vraag naar de Seriële Monitor, daarvoor heb ik de bekende “Serial.println” functie gebruikt, waarmee we de gebruiker dus de gestelde vraag presenteren welke we als parameter hebben meegegeven aan de functie.

Daarna zien we iets raars in de “while”-loop. Ik hoop dat je de “while”-loop nog kunt herinneren? De “while”-loop controleert eerst een conditie, indien de conditie waar is zal het code blok worden uitgevoerd, en de loop weer herhaald worden.

De conditie in de “while”-loop is “niet Serial.available”. “Serial.available() ” is een methode van het “Serial” object, welke teruggeeft hoeveel bytes het heeft gelezen van de seriële input. Dus als er nog niets getypt is, zal de return dus 0 (nul) zijn.

Serial.available() geeft het aantal, van de seriële poort gelezen bytes terug.
Als er nog geen bytes gelezen werden dan is het antwoord nul.

Nu moeten we nog even teruggrijpen op hoe een boolean als nummer wordt opgeslagen. Dus dat false als nul opgeslagen is, en dat de “while”-loop een true conditie wil.
De operator not ( ! ) gooit deze nul (= false) om naar een true.

Dus de “while”-loop blijft lopen, zolang er geen seriële input is, immers, dat geeft nul als antwoord, wat hetzelfde is als false, en met “not” ervoor wordt dat dus een true.

Omdat we hier alleen maar willen wachten en eigenlijk verder niets anders willen doen, laten we het code blok van de loop gewoon leeg. Ik heb er wel even een opmerking in gezet zodat we weten dat het bewust leeg is en waarom de loop hier bestaat.

Als een gebruiker eenmaal een karakter invoert (een char wat weer hetzelfde is als een byte), zal de “Serial.available()” een nummer teruggeven welke niet gelijk is aan nul. De not operator maakt hiervan het tegenovergestelde wat resulteert in een “false” waardoor de “while”-loop wordt verlaten.

In regel 8 gebruiken we het “return” statement – wat, zoals je misschien nog weet, het antwoord van de functie terug zal sturen.

In deze zelfde regel zien we ook een methode van Serial die we aanroepen:
Serial.readStringUntil(10); .

Serial.readStringUntil() geeft de gelezen bytes van de seriële invoer en zal het lezen stoppen tot een bepaalde karakter ontvangen wordt of er te lang geen activiteit was (de gebruiker heeft te lang niets ingetypt).

Omdat deze functie meteen terugkomt met “niks” als we geen bytes hebben gezien via de seriële invoer, kunnen we deze dus niet meteen aanroepen en zijn we dus genoodzaakt eerder beschreven “while”-loop te gebruiken om te wachten tot er werkelijk iets is om te lezen.

De parameter die we aan deze methode (functie) geven is een karakter (of char) welke we als “einde invoer” gaan zien. We willen dat het drukken op de ENTER-toets dat doet, en de ASCII waarde hiervoor is nummer 10 (zie de ASCII tabel) en daarom gebruiken we dus nummer 10 als onze parameter.

Stel we zouden willen dat de invoer afgesloten wordt als men de A toets indrukt, dan zou dat nummer 97 (uitgaande van een kleine “a”, welke de ASCII code 97 heeft).

Het antwoord van “Serial.readStringUntil” wordt vervolgens meteen aan het “return” statement doorgegeven zodat ons programma er iets mee kan doen.

Even een klein demo programma:

```
1  void setup() {
2    // set the speed for the serial monitor:
3    Serial.begin(9600);
4    Serial.println("De Arduino is wakker ...");
5
6    String OntvangenAntwoord;
7
8    OntvangenAntwoord = WachtOpInvoer("Voer a.u.b. jouw voornaam in.");
9    Serial.print("Jouw voornaam is: ");
10   Serial.println(OntvangenAntwoord);
11
12
13   OntvangenAntwoord = WachtOpInvoer("Voer a.u.b. jouw achternaam in.");
14   Serial.print("Jouw achternaam is: ");
15   Serial.println(OntvangenAntwoord);
16
17   Serial.println("Jouw Arduino is nu klaar met de setup functie en gaat nu oneindig
     de loop functie uitvoeren ...");
18 }
19
20 void loop() {
21   // leave empty for now
22 }
23
24 String WachtOpInvoer(String Vraag) {
25   Serial.println(Vraag);
26
27   while(!Serial.available()) {
28     // wacht op input
29   }
30
31   return Serial.readStringUntil(10);
32 }
```

Om te voorkomen dat we eindeloos dezelfde vraag krijgen, plaatsen we on code, zoals gebruikelijk, in de “setup()” functie.

We definiëren dus eerst een variabele, van het data type “String” object, omdat we hierin het antwoord willen opvangen.

Vervolgens zien we twee vergelijkbare delen van ieder 3 regels lang, waarin we eerst onze vraag stellen (om een naam vragen), en vervolgens het antwoord weergeven.
Dus elke keer als we een vraag stellen, gebruiken we onze “WachtOpInvoer()” functie.
Een maal ontvangen (dus de gebruiker heeft iets getypt en op ENTER gedrukt), geven we een stukje tekst weer, gevolgd door het antwoord van de gebruiker.

Als zowel voor- als achternaam afgehandeld zijn, geven we nog even een melding dat de “setup()” functie klaar is en de eindeloze herhaling van de (lege) “loop()” functie is begonnen.

Invoer van Gehele Nummers via de Seriële Monitor

We kunnen nu dus tekst invoer afhandelen, maar kunnen we ook nummers op die manier invoeren?

De uitdaging bij het lezen van nummers is dat onze invoer uit tekst bestaat. Zoals we al eens eerder hebben besproken bij Deel 3, tekst en nummers worden op een andere manier opgeslagen. Daarbij komt dan ook nog eens dat tekst niet alleen uit nummers bestaat, maar ook uit letters, speciale tekens, etc.

Laten we nog een keer door die uitleg lopen: Stel we hebben gebruiker die het nummer “3” invoert en op ENTER drukt.
Het antwoord van onze functie is dus een string welke slechts 1 karakter bevat. Als we in de ASCII-tabel kijken dan zien we dat “3” het nummer 51 levert.
Dit moeten we dus omzetten naar een echt nummer. Laten we eens kijken wat er gebeurt als we ons beperken tot gehele nummers (int).

De lastige conversie van Tekst naar Nummer

Je mag dit stukje overslaan als je wilt, maar het is leuk om even door te lopen omdat het je helpt met de manier van denken, mocht je in de toekmst complexere programma’s gaan schrijven. Er zijn overigens efficientere methoden om tekst om te zetten naar een nummer!

Het omzetten van tekst naar nummer kan een beetje saai zijn; je leest een karakter, en converteert het naar een nummer, stap voor stap.
Stel we hebben het nummer “3456” als een string en we willen hier dus een int van maken.

We beginnen bij het einde van de tekst, en dus van rechts naar links gaan werken, dus beginnen met het karakter “6”, wat in ASCII het nummer 54 is.
Omdat de nummers in een ASCII tabel bij 48 begint, moeten we dus 48 aftrekken van 54 om het nummer 6 te krijgen.

“6” = ASCII 54, “0” = ASCII 48 => 54 (“6”) – 48 (“0”) = 6
Antwoord = 6

Omdat we van rechts naar links werken, is het volgende karakter een “5”. “5” heeft de ASCII-waarde 53, dus daar gaan we weer, of toch niet?
Niet dus! Even terugdenkende aan de manier hoe we tellen; het nummer “5” staat eigenlijk voor

“50”!
Dus moeten we onze formule aanpassen zodat het voorziet in 10-tallen, 100-tallen, 1000-tallen etc.

Zie je hier al een patroon?
Als we van rechts naar links gaan, wordt er steeds een extra “0” toegevoegd aan een nummer. En als we beginnen te tellen met “0” (nul) dan zouden we het volgende kunnen zeggen met betrekking tot ons voorbeeld:
Voor de 6 voegen nul  nullen toe, voor de 5 èèn nul (50), voor de 4 twee nullen (400) en voor de 3 drie nullen (3000).

Om het makkelijker te maken: laten we de tekst eens omdraaiden, dus “6543” wordt nu “3456”.

Herinner je nog dat arrays met tellen bij nul beginnen? Hetzelfde geldt hier, omdat we eigenlijk ook hier met een array werken.

Dus we definiëren een string (3456) en een integer (0) en draaien de string (6543).

Vervolgens maken we een loop waarbij wel elk element van de array gaan bekijken. Niet vergeten: we hadden de string omgedraaid!
Voor het eerste element (6) doen we de ASCII-truc, dus 54 – 48, wat resulteert in 6.

Voor het tweede element, doen we ook weer de ASCII truc, maar vermenigvuldigen dat met 10, dus ( 53 – 48 ) * 10, wat dus 50 wordt, en tellen dit op bij het voorgaande nummer, dus: 50 + 6 = 56.

Voor het derde element doen we dit ook weer, maar dan keer 100, dus ( 52 – 48 ) * 100, wat dus 400 wordt, en tellen dit bij het vorige resultaat op: 400 + 56 = 456.

En weer voor het laatste element, maar dan keer 1000, dus ( 51 – 48 ) * 1000, en weer opgetelt: 3000 + 456 = 3456.

Nu hebben we nog een truc nodig om te voorzien in 10, 100 en 1000, en die is er ook, want we kunnen dat schrijven als “10 tot de macht”.
Ons eigenlijke rijtje is namelijk: 1, 10, 100, 1000 …

100 = 10 tot de macht 0, is hetzelfde als 1,
101 = 10 tot de macht 1, is hetzelfde als 10,
102 = 10 tot de macht 2, is hetzelfde als 10 × 10 (=100), en
103 = 10 tot de macht 3, is hetzelfde als 10 × 10 × 10 (=1000).

Je bent misschien niet, of niet zo, bekend met het werken met “tot de macht”, maar misschien herinner je nog dat we het hier eerder over gehad hebben. In dit geval is het superhandig in een loop als we dus 1, 10, 100, 1000 etc. nodig hebben in onze loop. Onze loop begint namelijk bij nul en eindigt (in ons voorbeeld) met 3 – niet vergeten he: 0, 1, 2, 3 … in tegenstelling tot hoe mensen tellen: 1, 2, 3, 4.

Vergeet ook niet:

“Tot de macht” zegt hoe vaak een nummer gebruikt wordt in een vermenigvuldiging, of in andere woorden:
Hoe vaak we 1 moeten vermenigvuldigen met dit nummer.

10 tot de macht 3, wordt geschreven als 103.
En als we bovenstaande tekst lezen dan wil dat dus zeggen dat we “1” drie keer met 10 moeten vermenigvuldigen en dat wordt dus:

10 tot de macht 3 = 103 = 1 × 10 × 10 × 10 = 1000

Nog wat voorbeelden:

10 tot de macht 1 = 101 = 1 × 10 = 10
2 tot de macht 4 = 24 = 1 × 2 × 2 × 2 × 2 = 16
3 tot de macht 2 = 32 = 1 × 3 × 3 = 9

Als we met machten rekenen, dan hebben we een speciaal geval, en wel “tot de macht nul”. Welk getal je ook neemt, “tot de macht nul” levert altijd 1.
Als je nog even terugleest naar de regel die ik je heb gegeven dan klopt dit ook.

Lees het onderstaande als: vermenigvuldig “1” nul keer met “2”. Dus levert dat … “1” als antwoord:

2 tot de macht 0 = 20 = 1

OK nu we een beetje afgedwaald zijn … terug naar tekst naar nummer converteren.

## 9.3. De eenvoudige String naar Nummer Conversie

Het moge duidelijk zijn dat voorgaande methode een dikke ellende is als we dit iedere keer moeten doen als we een tekst naar een nummer gaan converteren, en daarbij komt ons String object weer als goede hulp uit de bus. Het object heeft namelijk een methode (functie) die dit allemaal voor ons doet.

Nogmaals: de belangrijkste reden waarom we eerst door de moeilijke methode gegaan zijn, is zodat je een manier van denken gaat ontwikkelen om zelf handige trucs in de toekomst te bedenken als je moeilijkere programma’s gaat schrijven. Het geeft echter aan waarom we functies en methoden hebben, waarbij we eerder geschreven code kunnen hergebruiken en onszelf dus veel werk kunnen besparen.

Dus het “String” object heeft een methode die “toInt()” heet, welke de string tekst omzet naar een integer (int).

De methode “toInt()” begint met het eerste teken in de tekst, als dit geen nummer is, dan stopt het meteen en geeft als antwoord nul terug. Mocht dit echter geen nul zijn, dan gaat het verder naar het volgende teken in de tekst, en blijft dit herhalen tot het een teken tegen komt welke geen nummer is, of tot de tekst op is. Vervolgens pakt de methode alle gevonden nummers en zet ze om en geeft het resultaat terug als antwoord. Niet vergeten dus: zodra een niet-nummer gevonden wordt, stopt de methode en converteert het wat het gevonden heeft aan nummers.

Een paar voorbeelden:

| toInt() voorbeelden | |
|---|---|
| String | Geconverteerd naar int |
| 1234 | 1234 |
| 12Hallo34 | 12 |
| Hallo1234 | 0 |
| 12.3 | 12 |
| 0123 | 123 |

Ik heb de nummers in de string voorbeelden expres dik gedrukt, zodat je beter ziet wat er wel en wat er niet omgezet gaat worden.

Hier een voorbeeld, waarbij we deze methode gebruiken, samen met onze eerder bedachte functie:

```
1  void setup() {
2    // set the speed for the serial monitor:
3    Serial.begin(9600);
4    Serial.println("Jouw Arduino is wakker ...");
5
6    String OntvangenAntwoord;
7
8    OntvangenAntwoord = WachtOpInvoer("Voer een geheel nummer in.");
9
10   Serial.print("De tekst invoer was: ");
11   Serial.println(OntvangenAntwoord);
12
13   Serial.print("Geconverteerd naar een geheel nummer is dit: ");
14   Serial.println(OntvangenAntwoord.toInt());
15 }
16
17 void loop() {
18   // leave empty for now
19 }
20
21 String WachtOpInvoer(String Vraag) {
22   Serial.println(Vraag);
23
24   while(!Serial.available()) {
```

```
25    // wacht op input
26   }
27
28   return Serial.readStringUntil(10);
29  }
```

Dit voorbeeld lijkt natuurlijk veel op het voorgaande voorbeeld, maar een van de dingen die meteen zou moeten opvallen is dat we “String” object’s methode “toInt()” aanroepen. En dat doen we via de variabele “OntvangenAntwoord”, gevolgd door een punt ( . ) en vervolgens de naam van de methode “toInt()”. We hebben dit al eens eerder gedaan, zoals je kunt zien bij alles “Serial” aanroepen die we gedaan hebben met “begin” en “print”.

Een methode van een Object wordt aangeroepen door de variabele naam van het object, gevolgd door een punt en de methode naam, aan te roepen.

Invoer van Nummers, met Nummers achter de komma, via de Seriële Monitor

We hebben dus net gezien hoe we gehele nummers uit tekst kunnen omzetten, en je kunt je voorstellen dat dit met getallen et nummers achter de komma nog lastiger is. Overigens: niet vergeten dat bij het programmeren, we in het “Engels” werken en daarbij wordt er dus niet een komma maar een punt gebruikt voor cijfers achter de komma. Dus 1 en een half is dut niet 1,5 (Nederlands) maar 1.5 (Engels!).

Gelukkig hoeven we ook hier geen moeilijke dingen te doen, omdat het “String” object hiervoor een vergelijkbare methode heeft en wel “toFloat()“.

Vergeet dus niet dat nummers met cijfers achter de komma een “float” heet.

```
1   void setup() {
2    // set the speed for the serial monitor:
3    Serial.begin(9600);
4    Serial.println("Jouw Arduino is wakker ...");
5
6    String OntvangenAntwoord;
7
8    OntvangenAntwoord = WachtOpInvoer("Voer een nummer in.");
9
10   Serial.print("De tekst invoer was: ");
11   Serial.println(OntvangenAntwoord);
12
13   Serial.print("Geconverteerd naar een float nummer is dit: ");
14   Serial.println(OntvangenAntwoord.toFloat());
15  }
16
17  void loop() {
```

```
18   // leave empty for now
19  }
20
21  String WachtOpInvoer(String Vraag) {
22    Serial.println(Vraag);
23
24    while(!Serial.available()) {
25      // wacht op input
26    }
27
28    return Serial.readStringUntil(10);
29  }
```

## 9.4. Andere String object methoden (functies)

We hebben net twee methoden bekeken van het “String” object, maar er zijn we nog meer. Kijk zeker eens naar de Arduino Reference pagina van het String object waar nog meer methoden getoond worden. Helaas is deze pagina in het Engels, maar misschien snap je het e.e.a. wel.

We vinden op die pagina methoden om b.v. een string korter te maken door overbodige spaties weg te halen, of b.v. een string helemaal in hoofdletters of kleine letter te zetten.

De Arduino Reference pagina is een goede start voor het zoeken naar nog meer Arduino functies en methoden als je aan de slag gaat met Arduino Programmeren.

N.b.: Het Engelse woord “reference” wil zoiets zeggen als “naslagwerk”. Dus een “referentie” gids zeg maar. Deze worden door programmeurs vaak gebruikt om even snel te kijken hoe het ook alweer zat als we even vastlopen.

Ik zal hieronder een aantal van deze methoden laten zien, en wil je zeker aanmoedigen om met een aantal van deze methoden te gaan spelen.

Methoden

Ik heb hieronder de meest gebruikte methoden van het “String” object gezet, en er een link van gemaakt naar de betreffende Arduino Reference pagina.

String() – Maak een String object

charAt() – Benader een specifiek karakter in een string

compareTo() – Vergelijk twee strings

concat() – String aan een andere string plakken

endsWith() – Kijk of een string eindigt met een specifiek stukje tekst

equals() – Kijk of twee strings identiek zijn (hoofdletter gevoelig!)

equalsIgnoreCase() -Kijk of twee strings identiek zijn (NIET hoofdletter gevoelig!)

getBytes() – Kopieerd de individuele bytes van een string in een buffer

indexOf() – Zoekt een karakter of string in een string en geeft de locatie terug

lastIndexOf() – Zoekt het laatste voorkomen van een karakter of string in een string en geeft de locatie terug

length() – Geeft de lengte terug van een string, maar zonder het NULL karakter mee te tellen!

remove() – Verwijder 1 of meer karaketers van een string

replace() – Vervang 1 of meer karakters in een string met een andere string

setCharAt() – Verander een individueel karakter in een string, naar een ander karakter

startsWith() – Kijk of een string met een bepaalde tekst begint

substring() – Haal een stukje van de string op

toCharArray() – Converteer een"String" object naar een "string" array van karakters

toInt() – Converteer een string tekst naar een integer

toFloat() -Converteer een string naar een string tekst naar een float

toLowerCase() – Maak alle letters in een string, kleine letters

toUpperCase() -Maak alle letters in een string, hoofdletters

trim() – Verwijder alle spaties aan het begin en einde van een string.

## 9.5. Laatste gedachten …

We hebben nu 9 lessen doorlopen van onze minicursus voor het Arduino Programmeren voor beginners in de programmeertaal "C", en daarbij hebben we het gratis Arduino IDE gebruikt en een eenvoudige Arduino Uno.

Ik hoop dat je er plezier aan gehad hebt, en dat je wat dingen geleerd hebt.

Je zou nu instaat moeten zijn om een simpel programa te schrijven voor jouw Arduino. De meeste dingen zitten vast nog niet goed in je hoofd, maar dat is een kwestie van veel doen en vaak opzoeken. Maar,… maak je niet ongerust. Dingen die je vaak gebruikt ga je vanzelf onthouden. En … het opzoeken van dingen is na jaren programmeren nog steeds heel normaal, ook voor mij. En ik schreef mijn eerste programma's in 1978. Daarom zijn er ook zo veel referentie boeken en websites die je daarbij kunnen helpen.

Bedenk ook dat dit slechts het begin is van het werken met een Arduino en het programmeren in het algemeen …

Het werken met een Arduino bestaat uit twee aspecten: het programmeren en het werken met electronica. Beiden vereisen basiskennis om er mee aan de slag te gaan.

De volgende stap is het spelen met wat electronica natuurlijk, zoals sensoren, schakelaar, lampjes, motors, relays, en zelfs schermpjes.

Als je naar het Tweaking4All menu kijkt (links), kijk dan eens on “Hardware” “Arduino”, waar ik al wat eenvoudige voorbeelden heb gezet, maar ook complexere en mooie projecten zoals met ledstrips werken.

Met wat fantasie en handigheid kun ook misschien ook een robot bouwen in de toekomst …