Project 2.1 (Bij hfst 3)

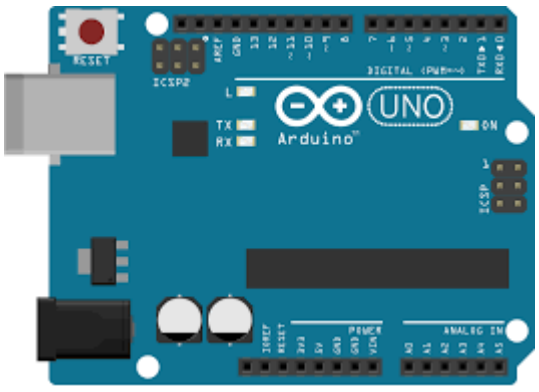
Simpele Calculator met een Arduino

We gaan een simpele calculator met een Arduino maken.

1.Hier volgt een stap voor stap methode.

- Sluit het Arduino-bordje aan op de PC
- Upload de code zoals beneden staat weergegeven.
- ‘Enable’ Seriële monitor (hulpmiddelen – seriële monitor)
- Voer een geldige berekening in (zie de code voor meer instructies)

2. Aansluitschema



Omdat het Arduino bordje alleen met de PC wordt verbonden is er geen aansluitschema.

3. Code

Hieronder volgt de code.

Voer de code in en probeer deze uit.

Je voert de getallen en tekens samen in, bijvoorbeeld "2+3", of 8*2, en dit geeft het resultaat op het beeldscherm.

1		/* Seriële arduino calculator
2		In dit project kun je basis
3		berekeningen maken met de arduino,
4		bijna net zo als met een echte rekenmachine.
5		Hij accepteert 2 cijfers en een operator, en
6		maakt de berekening, die kan zijn of +, -, * of /.
7		Voorbeeld : type "2+3" (zonder quotes en
8		spaties), en arduino geeft het antwoord 5.
9		Type de opgave "7-3" en arduino antwoord 4.
10		
11		juni 2018 ; Luuk Ottjes
12		Deze code is public domain
13		*/
14		
15		// Maar eerst, hoe creëer je de variabelen?
16		// en hoe stuur je de informatie naar arduino?
17		
18		long getal1; // eerste getal van de berekening, zie extra uitleg onder aan de code
19		// stuur die naar de seriële monitor
20		
21		// Dit is een lange variabele,
22		// we kunnen dus grote variabelen gebruiken

23		
24		long getal2; // tweede getal die naar de Seriële Monitor (SM) gestuurd wordt
25		char calSignal; // creëer een karakter voor een variabele om op te slaan
26		// dit is hier de uitkomst van de berekening.
27		
28		long resultaat; // het resultaat van de berekening
29		
30		void setup() {
31		Serial.begin(9600); // begin de seriële communicatie snelheid 9600 baud
32		Serial.println("Stuur me een som");
33		Serial.println("voorbeeld ; 2+3");
34		Serial.println();
35		// print dit naar het scherm via de seriële communicatie, en
36		// prints een lege regel
37		}
38		
39		void loop() {
40		while(Serial.available() > 0) {
41		// zolang de data groter is dan 0 wordt data verstuurd naar arduino,
42		
43		getal1 = Serial.parseInt();
44		// getal1 is het eerste getal
45		
46		// Waarom "Serial.parseInt, gebruiken,
47		// in het geval dat je 23 invoert , wordt dit opgeslagen als
48		// getal1 met als getal 23
49		
50		// als we het commando Serial.read() gebruiken , slaat
51		// Arduino alleen de 2 op.
52		

```
53      calSignal = Serial.read(); // calSignal zal het eerste teken
54      // gebruiken van een serie  (zie ook onder deze code)
55
56    getal2 = Serial.parseInt(); // bewaart de waarde van het 2de getal in getal2
57
59  resultaatFunctie(); // functie om de uitkomst te berekenen
60    Serial.println("De uitkomst van de som = ");
61      Serial.println(resultaat);// Print de uitkomst van de berekening
62    Serial.println(); // maak een lege regel
63      Serial.println("voer opnieuw een som in, aub");
64      Serial.println(); // maak een lege regel
65    }
66  }
67
68  void resultaatFunctie() { // Standaard functie om
69    // de berekening te maken
70
71    switch (calSignal) {
72      // Hier wordt  "switch...case" gebruikt om wat ruime te maken
73      // in de sketch. In principe, een functie die verifieert
74      // diverse "if" statements
75
76      // Hier, verifieert hij wat de waarde is die wordt aangehouden
77      // calSigna.In principe verifieert het goed lopen van
78      // de berekening
79
80      case '+' : // if calSignal is '+'
81      resultaat = getal1 + getal2; // telt de getallen op
82      // en zorgt ervoor dat de het resultaat de waarde van de berekening krijgt
83      break; // stopt de  "case"
84      case '-' : // if calSignal is '-'
```

85		resultaat = getal1 - getal2; // trekt de getallen af
86		// en zorgt ervoor dat de het resultaat de waarde van de berekening krijgt
87		break; // stopt de "case"
88		case '*' : // if calSignal is '*'
89		resultaat = getal1 * getal2; // vermenigvuldigt de getallen
90		// en zorgt ervoor dat de het resultaat de waarde van de berekening krijgt
91		break; // stopt de "case"
92		case '/' : // if calSignal is '/'
93		resultaat = getal1 / getal2; // deelt de getallen op elkaar
94		// en zorgt ervoor dat de het resultaat de waarde van de berekening krijgt
95		// PS: In het geval dat de uitkomst een niet geheel getal is
96		//zal het resulaat het dichtst bijzijnde integere getal worden
97		break; // stopt de "case"
98		default : //als niet een van de tekens is gebruikt...
99		Serial.println("ongelige operator");
100		// veroorzaakt een "error"
101		Serial.println();
102		}
103		}
104		/

long

[Data Types]

Description

Long variables are extended size variables for number storage, and store 32 bits (4 bytes), from -2,147,483,648 to 2,147,483,647.

If doing math with integers, at least one of the numbers must be followed by an L, forcing it to be a long. See the Integer Constants page for details.

Syntax

`long var = val;`

`var` - the long variable name `val` - the value assigned to the variable

Example Code

```
long speedOfLight = 186000L;   // see the Integer Constants page for explanation of the 'L'
```

parseInt()

Description

parseInt() returns the first valid (long) integer number from the serial buffer. Characters that are not integers (or the minus sign) are skipped.

In particular:

- Initial characters that are not digits or a minus sign, are skipped;
- Parsing stops when no characters have been read for a configurable time-out value, or a non-digit is read;

- If no valid digits were read when the time-out (see Serial.setTimeout()) occurs, 0 is returned;

Serial.parseInt() inherits from the Stream utility class.

Syntax

Serial.parseInt()

Serial.parseInt(char skipChar)

Parameters

skipChar: used to skip the indicated char in the search. Used for example to skip thousands divider.

char

[Data Types]

Description

A data type that takes up 1 byte of memory that stores a character value. Character literals are written in single quotes, like this: 'A' (for multiple characters - strings - use double quotes: "ABC").

Characters are stored as numbers however. You can see the specific encoding in the ASCII chart. This means that it is possible to do arithmetic on characters, in which the ASCII value of the character is used (e.g. 'A' + 1 has the value 66, since the ASCII value of the capital letter A is 65). See Serial.println reference for more on how characters are translated to numbers.

The char datatype is a signed type, meaning that it encodes numbers from -128 to 127. For an unsigned, one-byte (8 bit) data type, use the *byte* data type.

Example Code

```
char myChar = 'A';
char myChar = 65;     // both are equivalent
```

read()

Description

Reads incoming serial data. read() inherits from the Stream utility class.

Syntax

Serial.read()

Arduino Mega only:
Serial1.read()
Serial2.read()
Serial3.read()

Parameters

None

Returns

the first byte of incoming serial data available (or -1 if no data is available) - *int*

Example

```
int incomingByte = 0;   // for incoming serial data

void setup() {
        Serial.begin(9600);     // opens serial port, sets
data rate to 9600 bps
}

void loop() {

        // send data only when you receive data:
        if (Serial.available() > 0) {
                // read the incoming byte:
                incomingByte = Serial.read();

                // say what you got:
                Serial.print("I received: ");
                Serial.println(incomingByte, DEC);
        }
}
```