

Arduboy

Starters Guide

Stijn Caerts



JCW

Jeugd, Cultuur en Wetenschap vzw

Copyright © 2019 Stijn Caerts

JEUGD, CULTUUR EN WETENSCHAP VZW

STIJN.CAERTS.BE – WWW.JEUGDCULTUURENWETENSCHAP.BE

Licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License (the "License"). You may not use this document except in compliance with the License. You may obtain a copy of the License at <https://creativecommons.org/licenses/by-nc-sa/4.0/>.



First edition, August 2019

Contents

I	Introduction to C++	
1	Introduction	7
1.1	What is C++?	7
1.2	Syntax	7
1.2.1	Semicolon	7
1.2.2	Curly brackets	7
1.2.3	Comments	8
2	Variables and types	9
2.1	Data types	9
2.2	Variables	9
2.2.1	Declare	10
2.2.2	Scope	10
2.2.3	Operators	11
3	Control flow structures	13
3.1	If-then-else	13
3.1.1	If-then	13
3.1.2	Else	14
3.1.3	Nested if-statements	14
3.2	Switch	15
3.3	While loop	16
3.4	For loop	16

4	Functions and procedures	19
4.1	Defining functions	19
4.1.1	Type	19
4.1.2	Arguments	20
4.2	Calling functions	20

II

Arduboy

5	Arduino	23
5.1	Program structure	23
5.1.1	Global variables	23
5.1.2	The <code>setup()</code> procedure	23
5.1.3	The <code>loop()</code> procedure	24
5.2	Importing libraries	24
5.3	Arduino IDE	24
6	Arduboy	25
6.1	Settings	26
6.1.1	Installing the Arduboy2 library	26
6.1.2	Installing the Arduboy as an Arduino board	26
6.1.3	Selecting the board	26
6.2	Program structure	26
6.3	The Arduboy2 library	27
6.3.1	Init	27
6.3.2	Display	27
6.3.3	Buttons	31
6.3.4	Physics	32
6.3.5	LED	32
6.4	Emulator	33
6.4.1	ProjectABE	33
6.5	Upload program to Arduboy	33
	Bibliography	35
	Index	37



Introduction to C++

1	Introduction	7
1.1	What is C++?	
1.2	Syntax	
2	Variables and types	9
2.1	Data types	
2.2	Variables	
3	Control flow structures	13
3.1	If-then-else	
3.2	Switch	
3.3	While loop	
3.4	For loop	
4	Functions and procedures	19
4.1	Defining functions	
4.2	Calling functions	



1. Introduction

1.1 What is C++?

Programma's voor **Arduino** en **Arduboy** worden geschreven in de programmeertaal C++. Het is niet nodig om de hele programmeertaal te kennen en begrijpen voor je aan de slag kan gaan met programmeren. Daarom geven we hier een beknopt overzicht van de belangrijkste concepten die je nodig hebt om van start te gaan.

In de volgende hoofdstukken komen variabelen en types, controlestructuren (if-then-else, for, while) en functies en procedures aan bod. Tot slot zijn er nog twee hoofdstukken die dieper ingaan op de mogelijkheden van C++, namelijk arrays en lijsten, en klassen en objecten.

Voor een interactieve en uitgebreidere introductie tot C++, kan je terecht bij W3Schools (<https://www.w3schools.com/cpp/>).

1.2 Syntax

1.2.1 Semicolon

Achter elke instructie wordt in C++ een puntkomma ; geplaatst. Deze puntkomma vertelt de compiler dat op die plaats een instructie eindigt. Als je een puntkomma vergeet te plaatsen, is het programma niet correct en zal je het programma niet kunnen compileren. De compiler zal dan een foutmelding geven.

1.2.2 Curly brackets

Accolades {} worden gebruikt om een blok code aan te duiden. De accolades moeten gebalanceerd voorkomen in de code, wat betekent dat voor elke openende accolade { er een bijhorende sluitende accolade } moet zijn. Het gebruik van accolades wordt later geïllustreerd bij de controlestructuren (if-then-else, for, ..), functies en klassen.

1.2.3 Comments

Om de leesbaarheid van je code te verhogen, is het nuttig om commentaar toe te voegen. In deze commentaar beschrijf je wat dit deel van de code juist doet. Hierdoor is het duidelijk wat je juist hebt geprogrammeerd, ook als je later opnieuw je code bekijkt.

In C++ zijn er twee verschillende manieren om commentaar toe te voegen. Commentaar op één lijn wordt aangeduid met `//`. Alle tekst na `//` tot het einde van de lijn wordt beschouwd als commentaar en zal bijgevolg niet uitgevoerd worden.

■ Example 1.1 — Commentaarlijn.

```
// Dit is een lijn commentaar  
int a = 42;
```

◀

Commentaar over meerdere lijnen start met `/*` en eindigt met `*/`. Alle tekst tussen `/*` en `*/` wordt door de compiler genegeerd.

■ Example 1.2 — Commentaar over meerdere lijnen.

```
/*  
Deze commentaar neemt  
meerdere lijnen in beslag.  
*/  
int a = 42;
```

◀

2. Variables and types

Variabelen worden gebruikt om informatie of gegevens bij te houden in het programma. Een variabele is een plaats in het geheugen, met een zelfgekozen naam, waar gegevens in opgeslagen kunnen worden. In C++ heeft elke variabele een type. Een type komt overeen met het soort van gegevens die opgeslagen kunnen worden in zo'n variabele.

2.1 Data types

Op basis van het datatype van een variabele wordt er geheugen gereserveerd om informatie in op te slaan. In C++ zijn de volgende primitieve datatypes gedefinieerd:

Definition 2.1.1 — Primitieve datatypes.

bool	Booleaanse waarde: true of false
char	Karakter
int	Geheel getal
float	Kommagetal (met enkele precisie)
double	Kommagetal (met dubbele precisie)

Deze types worden gebruikt bij variabelen, maar ook bij functies. Deze primitieve datatypes kunnen ook gebruikt worden om complexere types samen te stellen, zoals bijvoorbeeld bij arrays en klassen.



Het type van een variabele is belangrijk voor de compiler omdat het type bepaald hoeveel geheugen er gereserveerd moet worden voor de variabele. Voor het opslaan van een float-variabele zijn er bijvoorbeeld 32 bits nodig, terwijl er voor de opslag van een kommagetal met dubbele precisie (double) 64 bits nodig zijn.

2.2 Variables

Een variabele kan gezien worden als een opslagplaats waaraan een naam verbonden is. In deze opslagplaats kan een waarde geplaatst worden.

2.2.1 Declare

Alvorens informatie kan opgeslagen worden in een variabele, moet de variabele eerst *gedefinieerd* of *gedeclareerd* worden. De declaratie van een variabele vertelt de compiler dat er plaats in het geheugen gereserveerd moet worden om gegevens van het gegeven datatype in op te slaan.

Een nieuwe variabele declareren doe je met een instructie van deze vorm:

```
type naam_variabele;
```

■ Example 2.1 — Declareren van een *integer* variabele.

```
// Declareer een nieuwe variabele van het type int met naam a
int a;
```

Je kan bij het definiëren van een nieuwe variabele er ook meteen een waarde aan toekennen.

■ Example 2.2 — Declareren van een *integer* variabele en een waarde toekennen.

```
/*
   Declareer een nieuwe variabele van het type int met naam a,
   en ken de waarde 42 toe
*/
int a = 42;
```

Bij het kiezen van een variabelenaam moet je met de volgende regels rekening houden:

- namen moeten beginnen met een letter of een laag streepje (_);
- namen kunnen letters, cijfers en lage streepjes bevatten;
- namen zijn hoofdlettergevoelig;
- spaties of speciale karakters zijn niet toegelaten;
- C++-keywords (zoals `int`, `if`, ...) kunnen niet gebruikt worden als variabelenaam.

■ Example 2.3 — Declareren en initialisatie van verschillende datatypes.

```
bool b = true;
char c = 'q';
int i = 42;
float f = 3.14;
double d = 3.14159265;
```

2.2.2 Scope

De *scope* van een variabele is de plaats in het programma waar je een gedefinieerde variabele kan gebruiken. Je kan dit ook beschouwen als de levensduur van de variabele.

De scope van een variabele wordt bepaald door waar de variabele werd gedeclareerd. Dit kunnen we opdelen in 3 verschillende soorten plaatsen.

1. in een functie of blok: **lokale variabelen**,
2. in de definitie van een functie: **parameters**,
3. buiten alle functies: **globale variabelen**.

In het algemeen kan gesteld worden dat de scope van de variabelen beperkt is tot het blok, aangeduid met accolades {}, waarin de variabele gedefinieerd is.

■ Example 2.4 — Scope van variabelen.

```
// Globale variabele g
int g = 12;

void setup() {
  // Lokale variabelen a en b
  int a = 42;

  /*
   Globale variabelen kunnen na declaratie
   overal gebruikt worden
  */
  int b = a + g;

  // Na declaratie moet het type van variabelen niet gespecificeerd worden
  g = b;
}

// Variabelen a en b kunnen buiten de functie setup() niet gebruikt worden
```

◀

2.2.3 Operators

Om de waarde in de variabelen aan te passen, maken we gebruik van *operators*. Hieronder geven we een overzicht van de belangrijkste operators in C++.

Arithmetic operators

Rekenkundige operators worden gebruikt om veelgebruikte wiskundige operaties uit te voeren. In deze voorbeelden gebruiken we de variabelen $A = 10$ en $B = 3$.

Operator	Beschrijving	Voorbeeld
+	Optelling	$A + B \rightarrow 13$
-	Vermindering	$A - B \rightarrow 7$
*	Vermenigvuldiging	$A * B \rightarrow 30$
/	Deling	$A / B \rightarrow 3$
%	Modulus (rest na deling)	$A \% B \rightarrow 1$
++	Verhoogt de waarde van de variabele met 1	$A++ \Rightarrow A \rightarrow 11$
--	Verlaagt de waarde van de variabele met 1	$A-- \Rightarrow A \rightarrow 9$

Assignment operators

De operator voor de toewijzing van een waarde aan een variabele is het gelijkheidsteken (=). Naast deze toewijzingsoperator bestaan er nog andere varianten, die kortere notaties zijn voor een rekenkundige operatie gevolgd door een toewijzing.

Operator	Voorbeeld	Equivalent
=	A = 12	A = 12
+=	A += 2	A = A + 2
-=	A -= 3	A = A - 3
*=	A *= 4	A = A * 4

Comparison operators

Vergelijkende operatoren worden gebruikt om twee waarden met elkaar te vergelijken. Het resultaat van deze operatie is een Booleaanse waarde, true of false.

Operator	Beschrijving	Voorbeeld
==	Gelijk aan	10 == 3 → false
!=	Niet gelijk aan	10 != 3 → true
>	Groter dan	10 > 10 → false
<	Kleiner dan	5 < 10 → true
>=	Groter dan of gelijk aan	10 >= 10 → true
<=	Kleiner dan of gelijk aan	5 <= 10 → true

Logical operators

Logische operatoren worden gebruikt om meerdere Booleaanse waarden te combineren in een formule tot één waarheidswaarde.

Operator	Beschrijving	Voorbeeld
&&	Logische conjunctie (AND)	(true && false) → false
	Logische disjunctie (OR)	(true false) → true
!	Logische negatie (NOT)	! true → false

Voor meer voorbeelden van het gebruik van logische operatoren, kan je een kijkje nemen naar waarheidstabellen [5].

3. Control flow structures

Controlestructuren bepalen de loop van het programma. Zo is het mogelijk om stukken code enkel in specifieke gevallen uit te voeren, of om een blok code meerdere keren te herhalen.

3.1 If-then-else

De eenvoudigste controlestructuur is het *if*-statement. Dit kan verder uitgebreid worden met een *else*-blok voor verdere controle over de uitvoeringsvolgorde.

3.1.1 If-then

Het *if*-statement wordt gebruikt om code uit te voeren die bedoeld is voor specifieke gevallen. Enkel wanneer aan de opgegeven conditie voldaan is, zullen de instructies in de body van het *if*-statement uitgevoerd worden. Als er niet aan de conditie voldaan wordt, zullen de instructies in de body overgeslagen worden.

Definition 3.1.1 — If.

```
if (conditie) {  
    /*  
        instructies tussen deze accolades worden enkel  
        uitgevoerd als de conditie tot true evalueert  
    */  
}
```

3.1.2 Else

Na een if-statement kan optioneel een else-blok geplaatst worden. De instructies in dit blok zullen enkel uitgevoerd worden als aan de conditie van het if-statement **niet** voldaan is. De code ziet er dan uit als volgt.

Definition 3.1.2 — If-then-else.

```
if (conditie) {
    /*
        instructies in dit blok worden uitgevoerd
        als de conditie evalueert tot true
    */
} else {
    /*
        instructies in dit blok worden uitgevoerd
        als de conditie evalueert tot false
    */
}
```

3.1.3 Nested if-statements

Als er meerdere condities zijn waarop getest moet worden, kan je gebruik maken van een genest if-statement. Hierbij plaats je in de else-blok een nieuw if-statement waarbij je de volgende conditie test. Dit ziet er uit als volgt.

■ Example 3.1 — Geneste if-statements.

```
int x = 42;

if (x < 10) {
    x = x * 2;
} else {
    if (x > 50) {
        x = x - 25;
    } else {
        x++;
    }
}
```

Omdat deze geneste if-statements snel onoverzichtelijk worden, kan deze code vereenvoudigd worden als volgt.

■ Example 3.2 — Else-if.

```
int x = 42;

if (x < 10) {
    x = x * 2;
} else if (x > 50) {
    x = x - 25;
} else {
    x++;
}
```



3.2 Switch

Als je een expressie wil testen op meerdere waardes, kan je in plaats van geneste if-statements ook gebruik maken van een *switch*-statement.

Definition 3.2.1 — Switch.

```
switch (expressie) {  
    case x:  
        // instructies  
        break;  
    case y:  
        // instructies  
        break;  
    default:  
        // instructies  
}
```

Hierbij wordt de expressie in de *switch* één keer geëvalueerd. Vervolgens wordt de waarde ervan vergeleken met de waarde in elke *case*. Als de waarden overeenkomen, dan wordt het bijhorende blok van instructies uitgevoerd.

Het *break*-statement zorgt ervoor dat, na een overeenkomstige *case* is gevonden, er niet nog op matches met andere *cases* gezocht moet worden. Het *default* keyword kan dan gebruikt worden om instructies uit te voeren in het geval dat er geen overeenkomstige *case* wordt gevonden.

■ Example 3.3 — Switch – dagen van de week.

```
int dag = 5;  
char c;  
  
switch (dag) {  
    case 1:  
        c = 'M';  
        break;  
    case 2:  
        c = 'D';  
        break;  
    case 3:  
        c = 'W';  
        break;  
    case 4:  
        c = 'd';  
        break;  
    case 5:  
        c = 'V';  
        break;  
    case 6:  
        c = 'Z';  
        break;  
}
```

```
case 7:
    c = 'z';
    break;
default:
    c = 'X';
}
```

Exercise 3.1 — Switch – dagen van de week. Welk karakter is er na het uitvoeren van bovenstaande code (Voorbeeld 3.3) opgeslagen in variabele c?

3.3 While loop

Als je een stuk code meerdere keren achter elkaar wil laten uitvoeren, kan je gebruik maken van een lus. De `while`-lus bevat een conditie en een body met instructies. Zolang de conditie waar is, worden de instructies in de body herhaald uitgevoerd.

Definition 3.3.1 — While.

```
while (conditie) {
    // body van de lus
    instructie;
}
```

■ **Example 3.4 — While.** Deze code berekent het kleinste veelvoud `vv` van het getal `x` dat groter is dan 24.

```
int x = 3;
int vv = 0;

while (vv <= 24) {
    vv += x;
}
```

3.4 For loop

Vaak zullen lussen voorkomen waarbij het aantal herhalingen vooraf vastligt. Deze lussen hebben dan volgende vorm.

■ **Example 3.5**

```
int i = 0;

while (i < 10) {
    // instructies

    i++;
}
```


Hierbij wordt voor het begin van de lus dan eerst de variabele `i` gedeclareerd, die in de lus gebruikt wordt om het aantal iteraties (herhalingen) te tellen. Hiervoor wordt op het einde van elke iteratie de waarde van `i` opgehoogd (`i++`). Dit soort lussen kan eenvoudiger worden geschreven met een `for`-lus.

Definition 3.4.1 — For.

```
for (statement1; statement2; statement3) {  
    // body, bevat een blok code dat telkens herhaald wordt  
}
```

Waarbij:

statement1 één maal uitgevoerd wordt voor het begin van de lus. Vaak is dit de declaratie van een *loop-variabele*, bijvoorbeeld `int i = 0`.

statement2 de conditie van de lus voorstelt.

statement3 telkens na een iteratie van de lus wordt uitgevoerd. Vaak is dit een manipulatie van de loop-variabele, zoals `i++`.

De code uit Voorbeeld 3.5 kan dus als volgt geschreven worden met een `for`-lus.

■ Example 3.6

```
for (int i = 0; i < 10; i++) {  
    // instructies  
}
```



4. Functions and procedures

Functies en procedures zijn blokken code die je op verschillende plaatsen in je code kan gebruiken. In plaats van telkens dezelfde code neer te schrijven, kan je beter de gemeenschappelijke code in een functie plaatsen. Dit verhoogt ook de leesbaarheid van de code.

4.1 Defining functions

De definitie van een functie is opgebouwd zoals hieronder beschreven.

Definition 4.1.1 — Functie.

```
type functienaam(argumenten) {  
    // hier komt de gemeenschappelijke code  
  
    return resultaat;  
}
```

De definitie van een functie bestaat uit de hoofding en de body. In de hoofding wordt eerst het type van de functie genoteerd. Hierna volgt de naam van de functie, en tot slot worden de argumenten van de functie tussen de haakjes geplaatst. Daarna volgt de body van de functie binnen de accolades.

4.1.1 Type

Het type van de functie, is het type van de waarde die de functie teruggeeft. Een functie wordt vanuit de code aangeroepen, waarna de code in de body van de functie wordt uitgevoerd. Het resultaat van deze berekeningen wordt vervolgens teruggegeven aan de plaats waarop deze functie-aanroep plaatsvond. Daar kan het resultaat dan verder gebruikt worden.

De primitieve datatypes uit Definitie 2.1.1 kunnen gebruikt worden als type van een functie. Het is ook mogelijk dat een functie geen resultaat heeft. Zo'n functie noemen we een procedure. Als *return type* gebruiken we dan `void`, het lege type.

4.1.2 Arguments

Via de *argumenten* of *parameters* kan informatie doorgegeven aan de functie. Deze parameters gedragen zich als variabelen binnen in de body van de functie. De verschillende parameters worden in de hoofding van elkaar gescheiden met een komma.

■ Example 4.1 — Functie – optelling.

```
int optelling(int a, int b, int c) {  
    int som = a + b + c;  
    return som;  
}
```

◀

■ Example 4.2 — Functie – minimum.

```
int minimum(int a, int b) {  
    if (a < b) {  
        return a;  
    } else {  
        return b;  
    }  
}
```

◀

4.2 Calling functions

Nadat een functie is gedefinieerd, kan deze functie aangeroepen worden op andere plaatsen in de code. Het aanroepen van een functie doe je als volgt:

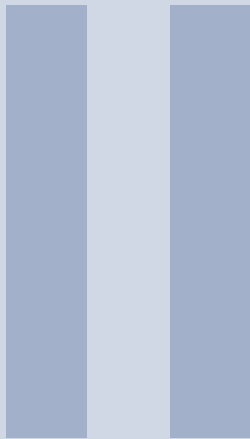
functienaam(argumenten);

De functie `minimum()` uit Voorbeeld 4.2 kan dus op de volgende manier aangeroepen worden.

■ Example 4.3 — Aanroepen van de functie *minimum()*.

```
int x = 42;  
int y = 13;  
  
int resultaat = minimum(x, y);
```

◀



Arduboy

5	Arduino	23
5.1	Program structure	
5.2	Importing libraries	
5.3	Arduino IDE	
6	Arduboy	25
6.1	Settings	
6.2	Program structure	
6.3	The Arduboy2 library	
6.4	Emulator	
6.5	Upload program to Arduboy	
	Bibliography	35
	Index	37

5. Arduino

5.1 Program structure

Elk Arduino-programma heeft dezelfde structuur. De belangrijkste kenmerken die steeds terugkeren zijn de globale variabelen, de `setup()` procedure en de `loop()` procedure. In de volgende secties bespreken we deze in meer detail.

Definition 5.1.1 — Arduino programmastructuur.

```
// Declaratie van globale variabelen

void setup() {

}

void loop() {

}
```

5.1.1 Global variables

Bovenaan het Arduino-programma worden de globale variabelen gedeclareerd. Deze globale variabelen kunnen daarna overal in het programma gebruikt worden, zowel in de `setup()` en `loop()` functie.

5.1.2 The `setup()` procedure

De `setup()` procedure wordt éénmaal uitgevoerd bij het opstarten van het programma.



In de `setup()` procedure ga je waarschijnlijk de waardes van verscheidene variabelen initialiseren. Het is belangrijk dat je deze variabelen declareert **buiten** de `setup()` functie. Anders zijn deze variabelen lokaal, en kunnen ze niet buiten de scope van de `setup()` functie gebruikt worden.

5.1.3 The `loop()` procedure

De `loop()` procedure doet precies wat de naam doet vermoeden. De code in deze procedure wordt steeds opnieuw uitgevoerd. Hier wordt de eigenlijk code van het Arduino-programma geschreven. De herhaling blijft duren tot de stroom van de Arduino wordt losgekoppeld of op de reset-knop wordt gedrukt.

5.2 Importing libraries

Bij het programmeren voor Arduino is het veel voorkomend om gebruik te maken van *libraries*. Libraries bevatten functies die door andere ontwikkelaars zijn geschreven. Ze hebben als doel om de complexiteit van jouw code te verlagen door operaties te implementeren op een lager niveau. Hierdoor moet je bijvoorbeeld niet handmatig alle communicatie met een display programmeren, maar gebruik je hiervoor de functionaliteit van de library.

Het importeren van een library in je code doe je op de volgende manier:

```
#include <Library.h>
```

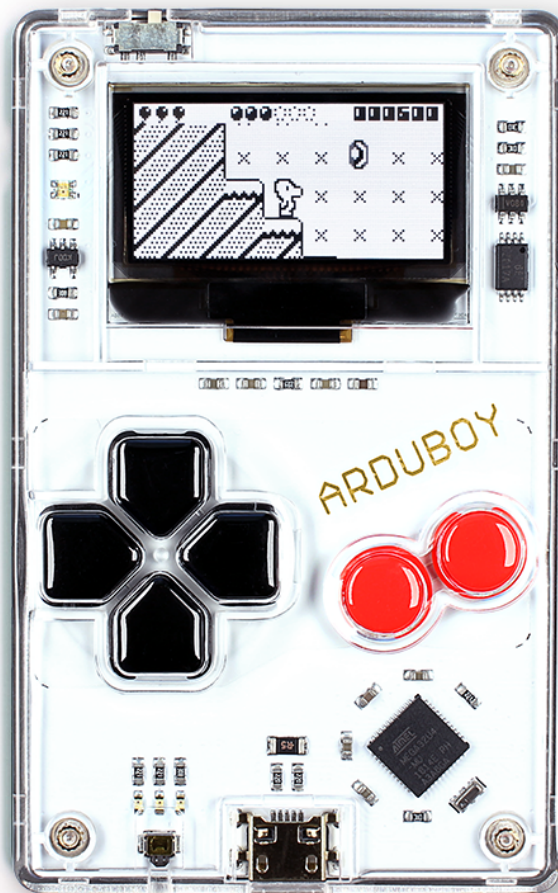
5.3 Arduino IDE

Voor het schrijven van Arduino programma's maken we gebruik van de Arduino IDE. Dit programma is een *Integrated Development Environment*, wat betekent dat het alle tools bevat om programma's te schrijven, compileren en op te laden naar de Arduino. De Arduino IDE kan gratis gedownload worden vanaf de Arduino website¹.



¹<https://www.arduino.cc/en/Main/Software>

6. Arduboy



6.1 Settings

Voor we aan de slag kunnen gaan met het programmeren van spelletjes voor de Arduboy, moeten we eerst enkele instellingen aanpassen in de Arduino IDE. Voor een handleiding met screenshots kan je terecht bij de Arduboy Community [2].

6.1.1 Installing the Arduboy2 library

De *Arduboy2* library bevat een handige interface die er voor zorgt dat we gemakkelijk alle functies van het apparaat kunnen aanspreken. Zo kunnen we ons volledig focussen op het ontwerpen en programmeren van het spel zelf.

Alvorens we deze library kunnen gebruiken, moet ze eerst geïnstalleerd worden in de IDE. Dit doe je als volgt:

1. Klik op **Hulpmiddelen > Bibliotheken beheren...**
2. In het venster *Bibliotheek Beheer* zoek je naar de library **Arduboy2**
3. Installeer de laatste versie door de **Arduboy2** library aan te duiden en op **Installeren** te klikken

Via deze manier kan je ook andere libraries installeren indien je dat wenst.

6.1.2 Installing the Arduboy as an Arduino board

1. Ga naar **Bestand > Voorkeuren**
2. In het scherm *Voorkeuren* geef je bij **Additionele Board Beheer URLs** de volgende URL in: https://arduboy.github.io/board-support/package_arduboy_index.json
3. Klik op OK om het venster *Voorkeuren* te sluiten
4. Ga naar **Hulpmiddelen > Board > Board Beheer...**
5. Zoek naar **Arduboy** en klik op **Installeren**

6.1.3 Selecting the board

Om de IDE te laten weten dat je gaat programmeren voor de Arduboy, moet je het board in de IDE aanpassen. Ga naar **Hulpmiddelen > Board** en kies **Arduboy** in deze lijst.

6.2 Program structure

```
#include <Arduboy2.h>
```

```
Arduboy2 arduboy;
```

```
void setup() {  
    arduboy.begin();  
}
```

```
void loop() {  
    if (arduboy.nextFrame()) {  
        arduboy.clear();
```

```
        // GAME LOGIC
```

```
        arduboy.display();  
    }  
}
```

Bovenstaand codefragment geeft een template van waar je kan starten om een Arduboy game te programmeren. Op de eerste lijn wordt de Arduboy2 library ingeladen. Daarna wordt er een Arduboy2-object aangemaakt. Dit object wordt gebruikt om de functionaliteit van de library op te roepen. In de `setup()` procedure wordt de functionaliteit van de Arduboy geïnitieerd.

6.3 The Arduboy2 library

Hier geven we een overzicht van de belangrijkste functies in de Arduboy2 library [1]. Deze functies kunnen op deze manier gebruikt worden:

```
arduboy.functienaam();
```

Optionele parameters voor de functies staan *schuingedrukt*.

6.3.1 Init

Deze functies worden gebruikt om de functionaliteit van de Arduboy te initialiseren.

Library — **begin()**. Initialiseer de hardware, het logo weergeven, etc. Deze functie moet één keer opgeroepen worden in de `setup()` procedure.

Library — **initRandomSeed()**. Kies een random waarde als *seed* voor de random number generator. Deze methode is het meest effectief wanneer ze na een (semi-)random tijd wordt opgeroepen.

6.3.2 Display

Deze functies hebben betrekking tot het scherm van de Arduboy.

Library — **DISPLAY**. Er zijn de volgende handige constanten voor de breedte en de hoogte van het scherm. Daarnaast zijn er ook constanten voor de kleuren van het scherm.

- WIDTH
- HEIGHT
- BLACK
- WHITE

Library — **clear()**. Wis de display buffer. Het volledige scherm wordt op zwart gezet.

Library — **display()**. Plaats de inhoud van de display buffer op het scherm.

Library — **setFrameRate(fps)**. Stel de frame-rate in.

Parameter

- `int fps`: de gewenste frame-rate in frames per seconde

Library — **nextFrame()**. Geeft aan of het tijd is om het volgende frame weer te geven.

Returns

- `bool`: `true` als het tijd is voor het volgende frame, anders `false`

Draw

Deze functies worden gebruikt om zelf vormen te tekenen op het scherm van de Arduboy.

Library — **drawPixel(x, y, color=WHITE)**. Kleur een enkele pixel in de opgegeven kleur.

Parameters

- `int x`: x-coördinaat van de pixel
- `int y`: y-coördinaat van de pixel
- `int color` (optioneel): kleur → WHITE of BLACK

Library — **drawCircle(x, y, r, color=WHITE)**. Teken een cirkel.

Parameters

- `int x`: x-coördinaat van het middelpunt van de cirkel
- `int y`: y-coördinaat van het middelpunt van de cirkel
- `int r`: straal van de cirkel
- `int color` (optioneel): kleur → WHITE of BLACK

Library — **drawRect(x, y, w, h, color=WHITE)**. Teken een rechthoek.

Parameters

- `int x`: x-coördinaat van de linkerbovenhoek
- `int y`: y-coördinaat van de linkerbovenhoek
- `int w`: breedte
- `int h`: hoogte
- `int color` (optioneel): kleur → WHITE of BLACK

Library — **drawRoundRect(x, y, w, h, r, color=WHITE)**. Teken een rechthoek met afgeronde hoeken.

Parameters

- `int x`: x-coördinaat van de linkerbovenhoek
- `int y`: y-coördinaat van de linkerbovenhoek
- `int w`: breedte
- `int h`: hoogte
- `int r`: straal van de afronding
- `int color` (optioneel): kleur → WHITE of BLACK

Library — **drawTriangle(x0, y0, x1, y1, x2, y2, color=WHITE)**. Teken een driehoek.

Parameters

- `int x0, int x1, int x2`: x-coördinaten van de hoeken
- `int y0, int y1, int y2`: y-coördinaten van de hoeken
- `int color` (optioneel): kleur → WHITE of BLACK

Library — **drawLine(x0, y0, x1, y1, color=WHITE)**. Teken een lijn tussen de opgegeven punten.

Parameters

- `int x0, int x1`: x-coördinaten van de punten
- `int y0, int y1`: y-coördinaten van de punten
- `int color` (optioneel): kleur → WHITE of BLACK

Library — **drawFastVLine(x, y, h, color=WHITE)**. Teken een verticale lijn.

Parameters

- `int x`: x-coördinaat van het bovenste uiteinde
- `int y`: y-coördinaat van het bovenste uiteinde
- `int h`: lengte
- `int color` (optioneel): kleur → WHITE of BLACK

Library — **drawFastHLine(x, y, w, color=WHITE)**. Teken een horizontale lijn.

Parameters

- `int x`: x-coördinaat van het linkse uiteinde
- `int y`: y-coördinaat van het linkse uiteinde
- `int w`: lengte
- `int color` (optioneel): kleur → WHITE of BLACK

Library — **fillCircle(x, y, r, color=WHITE)**. Teken een opgevulde cirkel.

Parameters

- `int x`: x-coördinaat van het middelpunt van de cirkel
- `int y`: y-coördinaat van het middelpunt van de cirkel
- `int color` (optioneel): kleur → WHITE of BLACK

Library — **fillRect(x, y, w, h, color=WHITE)**. Teken een opgevulde rechthoek.

Parameters

- `int x`: x-coördinaat van de linkerbovenhoek
- `int y`: y-coördinaat van de linkerbovenhoek
- `int w`: breedte
- `int h`: hoogte
- `int color` (optioneel): kleur → WHITE of BLACK

Library — **fillRoundRect(x, y, w, h, r, color=WHITE)**. Teken een opgevulde rechthoek met afgeronde hoeken.

Parameters

- int x: x-coördinaat van de linkerbovenhoek
- int y: y-coördinaat van de linkerbovenhoek
- int w: breedte
- int h: hoogte
- int r: straal van de afronding
- int color (optioneel): kleur → WHITE of BLACK

Library — **fillTriangle(x0, y0, x1, y1, x2, y2, color=WHITE)**. Teken een opgevulde driehoek.

Parameters

- int x0, int x1, int x2: x-coördinaten van de hoeken
- int y0, int y1, int y2: y-coördinaten van de hoeken
- int color (optioneel): kleur → WHITE of BLACK

Library — **fillScreen(color=WHITE)**. Vul het scherm op met de opgegeven kleur.

Parameters

- int color (optioneel): kleur → WHITE of BLACK

Text

Deze functies worden gebruikt om tekst weer te geven op het scherm van de Arduboy.

Library — **setCursor(x, y)**. Stel de locatie van de tekst cursor in.

Parameters

- int x: x-coördinaat
- int y: y-coördinaat

Library — **setTextSize(s)**. Stel de grootte van de tekstkarakters in.

Parameter

- int s: tekstgrootte (≥ 1)

Library — **setTextColor(c)**. Stel de tekstkleur in.

Parameter

- int c: kleur → WHITE of BLACK

Library — **setTextBackground(bg)**. Stel de achtergrondkleur van de tekst in.

Parameter

- int bg: achtergrondkleur → WHITE of BLACK

Library — **setTextWrap(w)**. Schakel text wrapping in of uit. Als text wrapping is ingeschakeld wordt de tekst op een nieuwe lijn geplaatst als deze te lang is om op de huidige lijn te passen.

Parameter

- bool w: true om text wrapping in te schakelen, false om uit te schakelen

Library — **print(content)**. Print de opgegeven content op de plaats van de cursor.

Parameter

- content: tekst die weergegeven moet worden

6.3.3 Buttons

Met deze functies kan je interacties via de knoppen van de Arduboy programmeren.

Library — **BUTTONS**. Voor de knoppen op de Arduboy zijn de volgende constanten gedefinieerd.

- | | | |
|------------|----------------|---------------|
| • A_BUTTON | • LEFT_BUTTON | • UP_BUTTON |
| • B_BUTTON | • RIGHT_BUTTON | • DOWN_BUTTON |

Buttons kunnen samengevoegd worden in een mask: LEFT_BUTTON + A_BUTTON

Library — **pressed(buttons)**. Test of de opgegeven knoppen ingedrukt worden.

Parameters

- int buttons: een bit-mask die aangeeft welke knoppen getest moeten worden

Returns

- bool: true als alle knoppen in het opgegeven mask momenteel ingedrukt zijn

Library — **notPressed(buttons)**. Test of de opgegeven knoppen **niet** ingedrukt worden.

Parameters

- int buttons: een bit-mask die aangeeft welke knoppen getest moeten worden

Returns

- bool: true als alle knoppen in het opgegeven mask momenteel **niet** ingedrukt zijn

6.3.4 Physics

Deze functies worden gebruikt om interacties tussen verschillende objecten in je game te berekenen.

Library — **collide(rect1, rect2)**. Controleer of twee rechthoeken elkaar snijden.

Parameters

- `Rect rect1, Rect rect2`: rechthoeken → `Rect(int x, int y, int width, int height)`

Returns

- `bool`: `true` als de opgegeven rechthoeken overlappen

Library — **collide(point, rect)**. Controleer of een punt binnen een rechthoek ligt.

Parameters

- `Point point`: punt → `Point(int x, int y)`
- `Rect rect`: rechthoek → `Rect(int x, int y, int width, int height)`

6.3.5 LED

Via deze functies kan je de RGB LED van de Arduboy aansturen.

Library — **setRGBled(red, green, blue)**. Stel de helderheid van de verschillende kleuren in de RGB LED in. De RGB LED zijn eigenlijk kleine individuele rode, groene en blauwe LEDs die zeer dicht bij elkaar geplaatst zijn. Door de helderheid van de 3 kleuren aan te passen, kan je verschillende kleuren tonen. De helderheid van elke LED kan een waarde aannemen van 0 (volledig uit) tot 255 (volledig aan).

Parameters

- `int red`: rode component
- `int green`: groene component
- `int blue`: blauwe component

Library — **setRGBled(color, val)**. Stel de helderheid van één van de kleuren in, zonder de andere kleuren aan te passen.

Parameters

- `int color`: naam van de kleur: `RED_LED`, `GREEN_LED` of `BLUE_LED`
- `int val`: helderheid, waarde tussen 0 en 255

6.4 Emulator

Als je een game aan het programmeren bent, wil je natuurlijk ook kunnen testen of deze werkt op de manier dat jij wil. Daarom bestaan er emulators voor de Arduboy.

6.4.1 ProjectABE

ProjectABE (<https://felipemanga.github.io/ProjectABE/>) is een online emulator voor de Arduboy. Je kan je eigen game testen door op het startscherm onder **My Projects > New Game** te klikken. In het volgende scherm zie je de emulator, zoals in Figuur 6.1. In de Arduino IDE klik je vervolgens op **Schets > Exporteer gecompileerd Binair bestand**. Hierdoor zal er een bestand met extensie `.hex` aangemaakt worden in dezelfde map als je code. Dit bestand met extensie `.hex` sleep je dan naar de emulator. De emulator zal dan meteen jouw code uitvoeren.

Je kan de emulator bedienen met het toetsenbord of door te klikken op de knoppen op het scherm.

6.5 Upload program to Arduboy

Voor je het programma op de Arduboy plaatst, controleer je best of zeker het juiste board geselecteerd is (zie Sectie 6.1). Daarna zet je de Arduboy aan en verbind je hem met de computer via de USB-kabel. Vervolgens selecteer je de juiste poort via **Hulpmiddelen > Poort** en kies dan de optie waar Arduino bij staat.

Nu ben je volledig klaar om het programma op de Arduboy te plaatsen. Je moet enkel nog op het pijltje (**Uploaden**) klikken en even wachten. Als alles goed gaat, staat je programma op de Arduboy.

Veel gameplezier!

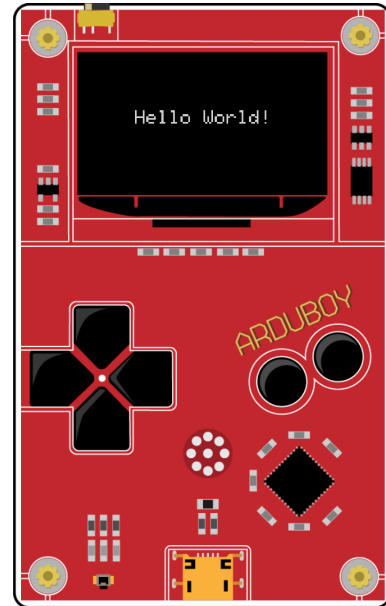


Figure 6.1: Project ABE



Bibliography

- [1] Arduboy2. *Library*. <https://mlxxxp.github.io/documents/Arduino/libraries/Arduboy2/Doxygen/html/index.html> (cited on page 27).
- [2] Jonathan Holmes (crait) - Arduboy Community. *Make Your Own Arduboy Game: Part 1 - Setting Up Your Computer*. <https://community.arduboy.com/t/make-your-own-arduboy-game-part-1-setting-up-your-computer/>. June 2019 (cited on page 26).
- [3] Tutorials Point. *C++ Tutorial*. <https://www.tutorialspoint.com/cplusplus/>.
- [4] W3Schools. *C++ Tutorial*. <https://www.w3schools.com/cpp/>.
- [5] Wikipedia. *Waarheidstabel*. <https://nl.wikipedia.org/wiki/Waarheidstabel> (cited on page 12).

Index

A

Accolade 7
Arduboy 25
Arduino 23
Argument *see also* Parameter

B

Board 26, 33
Booleaanse waarde .. *see also* Datatype, bool

C

C++ 7
Commentaar 8
Controlestructuren 13
 for 16
 if-then-else 13
 switch 15
 while 16

D

Datatype 9
 bool 9
 char 9
 double 9
 float 9
 int 9

E

Emulator 33

F

Functie 19

I

IDE 24

L

Library 24, 27

O

Operatoren 11
 Logisch 12
 Rekenkundig 11
 Toewijzend 12
 Vergelijkend 12

P

Parameter *see also* Variabele, Parameter
Procedure 19
Programmastructuur 23, 26
Puntkomma 7

T

Type *see also* Datatype

V

Variabele 9

 Declareren 10

 Globaal 10, 23

 Lokaal 10

 Parameter 10, 20

 Scope 10