# Inleiding

Welkom! Voor u ligt het pakket dat is uitgedeeld door de Ordina medewerkers. Gedurende deze hands on lab gaat u met dit pakket uw eigen IoT Kijkdoos maken en besturen met de “IoT Kijkdoos” app. Als dat gelukt is gaat u de Arduino programmeren door vier challenges uit te voeren. Als iets niet lukt, schroom niet en vraag een van de Ordina medewerkers om u te helpen. Succes!

Het volgende is vereist voor de hands on lab:

* Een Telefoon of Tablet (Android SDK 18/iOS 9 of hoger) met bluetooth low energy
* Een Laptop (Windows of Mac)

Voordat u naar het volgende hoofdstuk gaat kunt u alvast de kartonnen doos in elkaar zetten en het kijkgat maken met de bijgeleverde schaar.

# Hardware - Afbeelding componenten toevoegen

In afbeelding x zijn alle componenten weergegeven die u nodig heeft. Kijk goed of alles aanwezig is. Zo niet, meld dit dan even bij een Ordina medewerker. Onder afbeelding x is per component een korte uitleg gegeven over wat het is er waarvoor het gebruikt gaat worden.

AFBEELDING

## Arduino + USB-kabel (1)

De Arduino is een microcontroller. Het kan analoge input lezen en elektronische componenten aansturen door middel van digital output.

De Arduino is te programmeren met de Arduino IDE. Dit komt later aan bod.

## PCB (2)

De PCB is het zwarte board met het witte Ordina logo erop. Dit board moet op de Arduino worden geklikt, zie afbeelding x.

## Bluetooth 4.0 module (3)

De bluetooth module zorgt voor de draadloze communicatie met de app. Op de module is een sticker geplakt met de naam van de bluetooth module. Deze naam moet u selecteren als u de app gaat gebruiken.

## RGB Led module (4)

U heeft drie RGB led modules gekregen. Twee zullen er gebruikt worden als gewone lampen en een als RGB lamp. In afbeelding x

## Servo Motor (5)

## Buzzer (6)

## 9V batterij en kabel (7+8)

De 9V batterij en kabel zorgen ervoor dat de Arduino volledig draadloos functioneert.

## Dupont draden (9)

Met de Dupont draden koppelt u de elektronische componenten met de PCB.

# De “IoT Kijkdoos” App

Ordina heeft voor deze hands on lab de ‘IoT Kijkdoos’ app gecreëerd. Hiermee zijn de elektronische componenten in de IoT Kijkdoos te besturen via bluetooth. Deze app is te downloaden in zowel Play Store als de App Store. Let wel op dat uw telefoon/tablet voldoet aan de gestelde eis in de inleiding.

Er wordt automatisch gescand als de app geopend wordt. Let op dat u dezelfde naam aanklikt als op de bluetooth module staat. De modules zijn niet voorzien van een wachtwoord, dus het is mogelijk om met die van uw buurman te koppelen. Als de app gekoppeld is kunt u de IoT Kijkdoos besturen door op een van de afbeeldingen te klikken in afbeelding x.

# De Arduino Software

## Arduino IDE installeren (Windows + Mac)

* Ga naar de map ‘Arduino IDE’ en kies het juiste besturingssysteem
* Dubbelklik op het bestand en volg de instructies

## Arduino bestanden openen

Ga naar de map ‘Arduino Bestanden’ en open het bestand Ordina.ino. Als het goed is wordt de Arduino IDE geopend en worden er vijf tabbladen weergegeven, namelijk: Ordina, Buzzer.h, Led.h, RgbLed.h en ServoMotor.h.

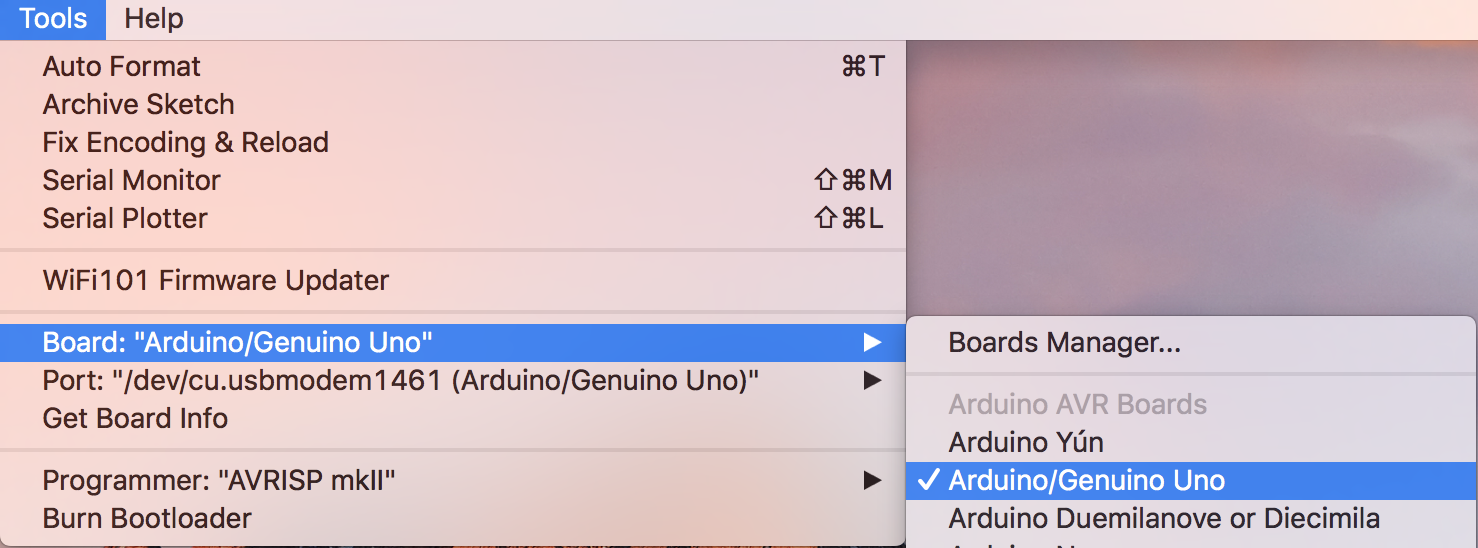
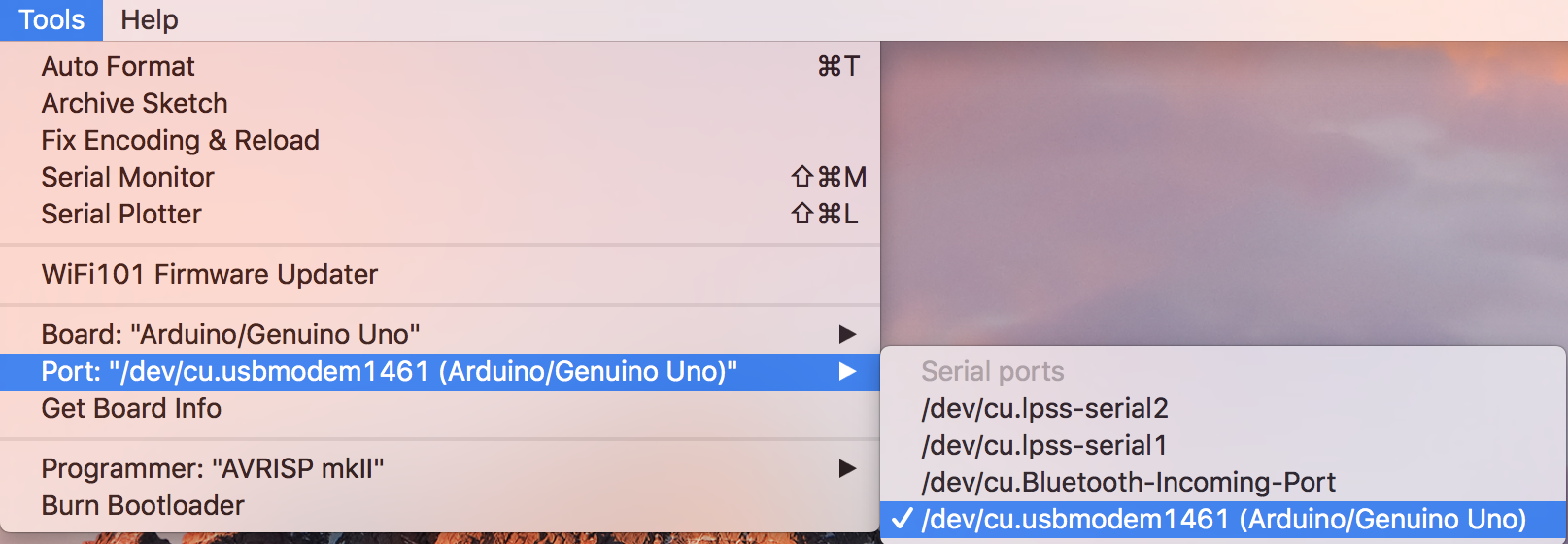
## Sketch Uploaden

Als u de sketch heeft aangepast moet het opnieuw geüpload worden naar de Arduino.

Voordat dit kan moet u eerste het volgende hebben gedaan:

* De Arduino koppelen aan de computer (USB-kabel)
* Het board selecteren (afbeelding x)
  + *Tools -> Board -> Arduino/Genuino Uno*
* De port selecteren (afbeelding x)
  + *Tools -> Port -> ‘USB NAME’ (Arduino/Genuino Uno)*
* Controleren of de code geen error bevat
  + *Klik linksboven op het vinkje*

Als u het bovenstaande heeft uitgevoerd bent u klaar om de sketch te uploaden. Klik linksboven op het pijltje en wacht tot het uploaden is voltooid.



## Debuggen

Stel u heeft de code veranderd en geüpload, maar er gaat iets niet goed en u weet niet waardoor het komt, dan is het tijd om te debuggen. U kunt de Serial.println()methode gebruiken om een waarde in de Serial Monitor te printen (zie functiekaart). Let wel op dat de Serial Monitor tijdens het runnen van de code open staat (*Tools -> Serial Monitor*).

## Code Folding

Als u het fijn vindt om code blokken in elkaar te kunnen schuiven kunt naar *Arduino -> Preferences* gaan en vervolgens ‘Enable Code Folding’aanvinken.

# De Challenges

Gefeliciteerd! Het is u gelukt om een eigen IoT Kijkdoos te maken en te besturen. Ordina heeft voor u vier challenges gemaakt om extra functionaliteit aan de IoT Kijkdoos te geven en te leren programmeren met de Arduino. Succes met het uitvoeren van de challenges!

## Challenge I: Broken Light

Zoals u gemerkt heeft is de lamp aan de rechterkant defect. Aan u de uitdaging om deze weer werkend te krijgen. Tijdens deze challenge kunt u gewend raken aan het veranderen en uploaden van de code. U kunt in het vorige hoofdstuk het kopje ‘Sketch Uploaden’ nogmaals doornemen als u niet meer weet hoe dit moet. Neem tijdens deze challenge ook vooral de tijd om even door de gehele code te scrollen.

*Tips:*

* De lamp kan gemaakt worden in het hoofdbestand (Ordina).

## Challenge II Gradient

De kleur van de discobal kan veranderd worden door de knop van de slider te verplaatsen, maar hoe leuk is het om met één druk op de knop een kleurverloop te zien? Ga naar het bestand RgbLed.h, scroll naar de methode RgbLed::showGradient() en voeg tussen de accolades de code die zorgt voor het kleurverloop.

*Tips:*

* Gebruik de methode RgbLed::writeColor(int rgbValue[3]) in RgbLed.h om de kleur van RGB led te veranderden.
* De functie delay() kan worden gebruikt om tussen het veranderen van de kleuren een pauze te creëren (zie methodeblad).
* U kunt een for loop gebruikt om code duplicatie tegen te gaan (zie methodeblad).

|  |  |
| --- | --- |
| Kleur | R-G-B |
| Off | 0-0-0 |
| Rood | 255-0-0 |
| Geel | 255-255-0 |
| Groen | 0-255-0 |
| Aqua | 0-255-255 |
| Blauw | 0-0-255 |
| Paars | 255-0-255 |

## Challenge III Custom Sound

|  |  |
| --- | --- |
| Noot | Frequentie (Hz) |
| c | 262 |
| d | 294 |
| e | 330 |
| f | 349 |
| # | 370 |
| g | 392 |
| a | 440 |
| b | 494 |
| C | 523 |
| D | 587 |
| E | 659 |
| F | 699 |
| G | 784 |
| A | 880 |
| B | 989 |

Het is tijd om een eigen geluid te maken met de buzzer. Ga naar het bestand Buzzer.h en scroll naar de methode Buzzer::customSound(). Met de functie tone() kunt u de buzzer een bepaalde toon laten produceren. Zie het functieblad voor de syntax en parameters. Klik in de app op XX en druk op de ‘Custom Sound’ om het geluid af te spelen.

Voor de muzikale deelnemers is hiernaast een tabel weergegeven met daarin de noten met bijbehorende frequenties. Experimenteren met andere frequenties kan natuurlijk ook. De minimale frequentie is 31 Hz en de maximale 65535 Hz.

*Tips:*

* De functie delay() kan worden gebruikt om een pauze tussen tonen te creëren (zie functieblad).
* Als dezelfde tonen meerdere keren herhaalt moeten worden kan hiervoor een for loop gebruikt worden (zie functieblad). Dit zorgt voor meer efficiëntie in de code.

## Challenge IV Special Effect

Deze challenge is speciaal. Tot nu toe hebben wij voor u bepaald welke methode er geïmplementeerd moest worden, maar nu is de beurt aan uzelf! Maak een eigen methode in één van de vier classes en zorg dat deze wordt aangeroepen als de character ‘i’ wordt ontvangen. Laat iedereen verstelt staat van uw ‘Special Effect’!

*Tips:*

* Kijk goed naar hoe de aanwezige methodes gecreëerd zijn. Dit gebeurt namelijk in het code blok van de class én erbuiten.

# Het Methodeblad

|  |  |  |
| --- | --- | --- |
| Syntax | Parameters | Uitleg |
| if (condition) { //Statements } | **Condition:** de uitkomst van de conditie moet altijd waar of niet waar zijn. Als de uitkomst waar is worden de regels code binnen het code block uitgevoerd. | De *if loop* wordt gebruikt om een handeling alleen uit te voeren als er aan de conditie wordt voldaan. |
| for (initialization; condition; increment) { //Statements }  Afbeelding 1 De For loop | **Int x = 0:** de variabele x wordt hier verklaard. Het getal 0 is de startwaarde van de *for loop*. Deze kan zelf bepaald worden.  **X < 100:** De conditie. Als X groter of gelijk is aan 100 dan wordt de loop verlaten. Deze waarde kan zelf bepaald worden.  **X++:** de toename (in dit geval). X wordt hier elke ronde met 1 verhoogd. | De *for loop* wordt gebruikt om een handeling meerdere keren te herhalen. In afbeelding 1 wordt bijvoorbeeld het getal van 0 tot 99 geprint. |
| Serial.print(val)  Serial.println(val) | **Val:** de waarde die geprint moet worden in de *Serial Monitor*. Dit kan elke datatype zijn. | Dit wordt vandaag gebruikt voor het debuggen. U gebruikt het bijvoorbeeld als u wil weten of een methode wordt aangeroepen of dat een variabele een bepaalde waarde heeft. Let wel op dat u de *Serial Monitor* heeft geopend (*Tools -> Serial Monitor*). |
| tone(pin, frequence)  tone(pin, frequence, duration) | **Pin**: de output pin van de buzzer.  **Frequence**: de frequentie van de toon in hertz.  **Duration**: de duur van de toon in milliseconden. Deze parameter is optioneel. | De functie wordt gebruikt om een toon te produceren met de buzzer. |
| delay(ms) | **Ms:** de lengte van pauze in milliseconden. | De *delay* functie wordt gebruikt om het programma in zijn geheel voor x aantal milliseconden te pauzeren. |