# Inleiding

Beste deelnemer, allereerst welkom! Ordina heeft voor deze hands-on lab de IoT Kijkdoos ontworpen. Dit is de kijkdoos van vroeger, maar dan met elektronische componenten die bestuurd kunnen worden door een app.

Voor u ligt een 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.
* Een Laptop (Windows of Mac).

U kunt nu alvast de kartonnen doos in elkaar zetten en het kijkgat maken. Er gaan een aantal stanleymessen rond voor het kijkgat.

# Hardware

In afbeelding x zijn alle hardwarecomponenten 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 en hoe het aangesloten dient te worden.

AFBEELDING

## Arduino + USB-kabel (1)

De Arduino is een microcontroller. Het kan analoge input lezen en elektronische componenten aansturen door middel van digitale output. Het programmeren van de Arduino komt later aan bod.

## PCB (2)

De PCB is het zwarte board met het witte Ordina logo erop. Hierin zitten de elektrische circuits van de elektronische componenten verwerkt. Hierdoor worden het aantal losse draden gereduceerd en is er minder kans dat er hardware fouten ontstaan. Dit board moet met de ijzeren pinnen op de Arduino worden geklikt.

## Dupont draden (3)

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

## Bluetooth 4.0 module (4)

|  |  |
| --- | --- |
| Bluetooth | PCB |
| VCC | 3.3V |
| GND | GND |
| TX | D10 |
| RX | D11 |

De bluetooth module zorgt voor de draadloze communicatie met de “IoT Kijkdoos” app. Deze module is direct aan te sluiten aan op de PCB, zie tabel x. Op de module is een sticker geplakt met de naam van de bluetooth module. Deze naam moet u in de app selecteren als u de IoT Kijkdoos wil besturen. Dit wordt in het volgende hoofdstuk verder uitgelegd.

## RGB Led module (5)

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

## Servo Motor (6)

## Buzzer (7)

## 9V batterij + kabel (8)

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

# De “IoT Kijkdoos” App

Ordina heeft voor deze hands on lab de “IoT Kijkdoos” app gecreëerd. Hiermee kunnen de elektronische componenten in de IoT Kijkdoos bestuurd worden 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.

# 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 sketch openen

Ga naar de map “Arduino Sketch” 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.

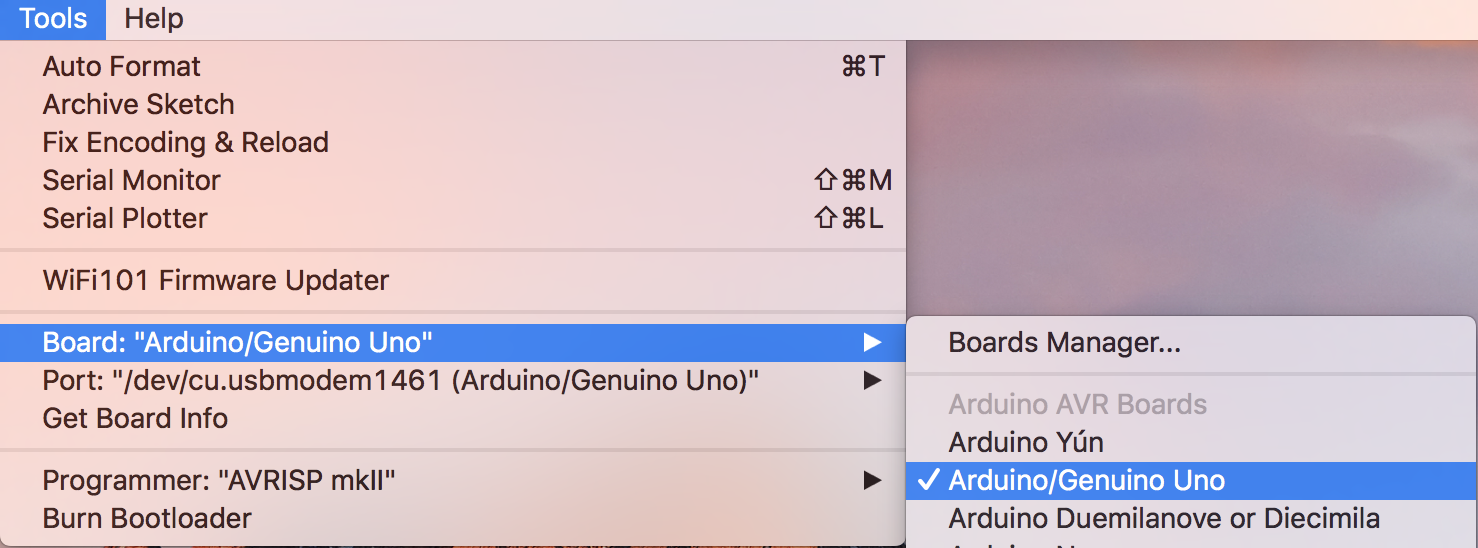
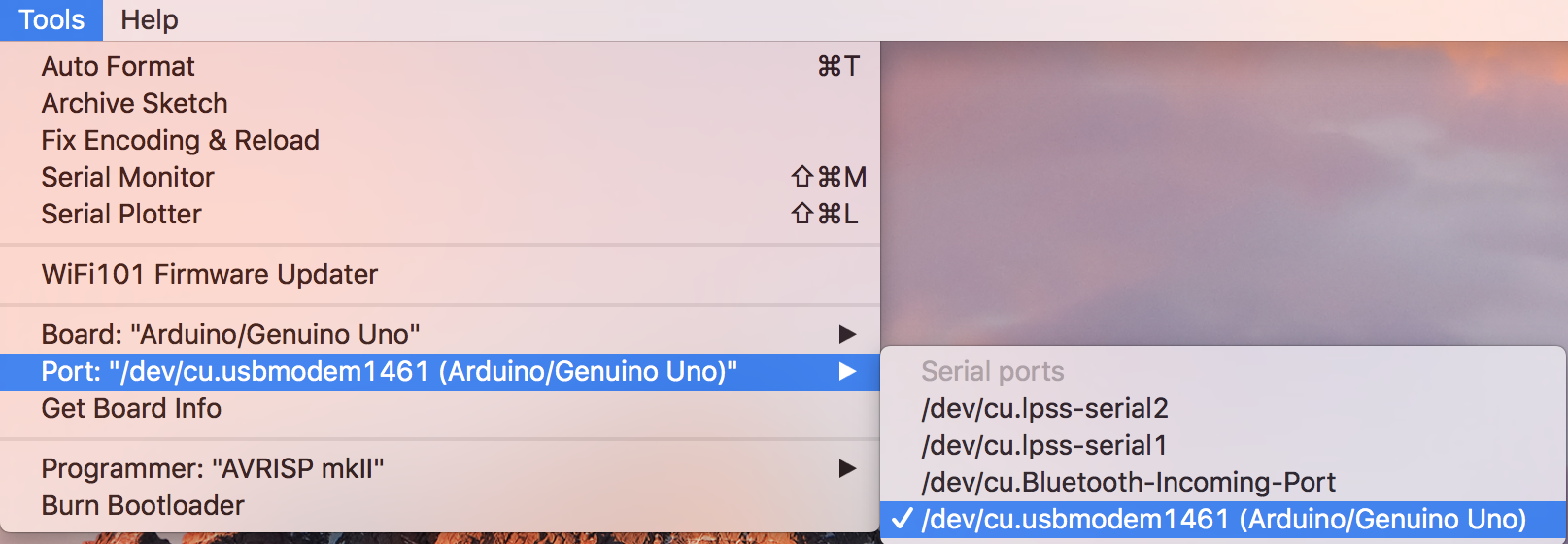
## Sketch Uploaden

De sketch moet naar de Arduino geüpload worden als de code veranderd is.

Voordat dit kan moet u eerst het volgende doen:

* De Arduino koppelen aan de computer (USB-kabel).
* Het juiste 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 Serial.println()gebruiken om een waarde in de Serial Monitor te printen (zie functiekaart). Let er wel op dat de Serial Monitor geopend is tijdens het runnen van de code (*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.

# Challenges

Gefeliciteerd! Het is u gelukt om een eigen IoT Kijkdoos te maken en besturen. Ordina heeft voor u vier challenges bedacht om extra functionaliteit aan de IoT Kijkdoos te geven en te leren programmeren met de Arduino. Alle functies die u voor de challenges nodig heeft zijn onderstreept en zijn nader toegelicht in het functieblad. U kunt de Arduino website raadplegen als de uitleg niet duidelijk genoeg is (www.arduino.cc). 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 nogmaals het kopje   
“Sketch Uploaden” doornemen als dit niet meer duidelijk is. 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.

Er is voor u een array gemaakt met daarin alle kleuren van de discobal, genaamd “allColors”. U kunt deze gebruiken in een for loop. De variabele “nColors” staat gelijk aan de lengte van de array.

*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.

*Extra:*

Mocht dit te eenvoudig zijn, creëer dan een nieuwe array met andere kleuren en maak er spectaculaire lichtshow van.

## 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.

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
* Bij herhalende tonen kunt u een for loop gebruiken om codeduplicatie tegen te gaan.

## 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 (hoofdbestand). 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 functieblad

|  |  |  |
| --- | --- | --- |
| 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 codeblok alleen uit te voeren als er aan de conditie wordt voldaan. |
| for (initialization; condition; increment) { //Statements }  Afbeelding 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 codeblok meerdere keren uit te voeren. In afbeelding 1 wordt bijvoorbeeld het getal van 0 tot 99 geprint. |
| Serial.print(val)  Serial.println(val) | **Val:** de waarde die geprint wordt in de Serial Monitor. Dit kan elke datatype zijn. | *Serial.println()* wordt vandaag gebruikt voor het debuggen. U gebruikt het bijvoorbeeld als u wil weten of een methode wordt aangeroepen of wat de waarde van een variabele is. Let er wel op dat u de Serial Monitor heeft geopend tijdens het runnen van de code (*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. | *Tone()* wordt gebruikt om een toon te produceren met de buzzer. |
| delay(ms) | **Ms:** de lengte van de pauze in milliseconden. | *Delay()* wordt gebruikt om het programma voor x aantal milliseconden te pauzeren. |