Wires

H f G
Hochschule für Gestaltung
Schwäbisch Gmünd

David Glock
Kai Magnus Müller

Dozent
Fabian Gronbach

4. Semester
Interface Design II
Wintersemester 2017 / 2018

Inhalt

Einleitung

Wires ist eine Prototypingumgebung, die es erlaubt Hardwareprototyping mit screenbasierten Prototypen zu verbinden.

Research

Aktuell existieren eine Reihe von Prototyping Tools, die es Nutzern möglichst einfach machen wollen von Screendesigns zu interaktiven Prototypen zu gelangen. Die Tools verfolgen verschiedene Ansätze bei der Programmierung der Prototypen und können verschieden komplexe Interaktionen realisieren.

Die Entwicklung von Hardwareprojekten, wie zu Beispiel im IoT Bereich benötigt eigene Tools, die meist inkompatibel sind und im Fall von VVVV zwar Grafikausgabe unterstützen, aber nicht auf UIs ausgelegt sind. Will man Projekte, wie zum Beispiel IoT Apps für Raumsteuerungen testen, fehlen Programme die beide Seiten verbinden.

Nutzer für ein solches Programm wären Designstudenten oder Arbeitende im Bereich Design.

Analyse des bisherigen Prozesses

Als erstes wurden die bestehenden Prototypingtools und Hardware kategorisiert und gesammelt. Aus eigenen Erfahrungen und den Erfahrungen anderer Studenten wurde der typische Prozess der zum Bearbeiten von Hardwareprojekten nötig ist aufgestellt.

Dazu wurde ein Szenario aufgestellt in dem das Ziel ist eine Smart Home App mit Anbindung an Wettersensoren zu entwickeln. Die einzelnen Schritte wurden auf Schwierigkeiten und Probleme untersucht.

Post-It Sammlung zur Recherche von Hardware, Sprachen, Screendesign Tools und einem kurzen Projektablauf.

Typischer Ablauf

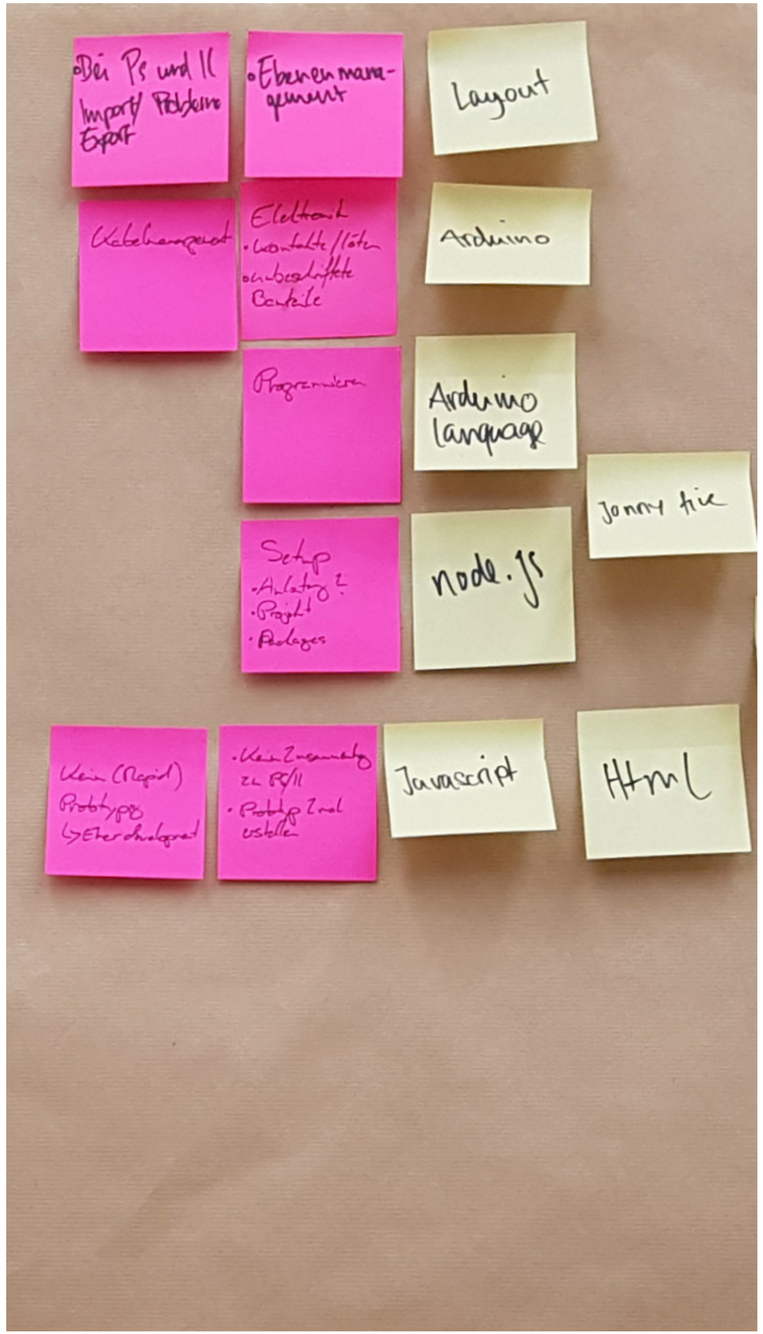
Layout
Zunächst wird gewöhnlich das Layout der Screens in Illustrator oder Sketch erstellt.

Arduino
Parallel dazu oder im nächsten Schritt wird die Hardware mit einem Arduino Board gebaut. Hier stellen sich erste Probleme, wie Kabelmanagement und unbeschriftete Bauteile.

Arduino programmieren
Die Arduino Boards müssen programmiert werden, was nicht unbedingt anfängerfreundlich ist oder zum Repertoire von Designstudenten gehört.

Daten senden / empfangen
Um Daten an Arduinos zu senden oder zu empfangen sind node.js Server nötig, so kann mit der Anwendung und der Hardware kommuniziert werden. Hier stellen sich eine Reihe von Problemen, wie fehlende oder unvollständige Anleitungen. Außerdem gibt es wenig Lösungen die UI mit Hardware verbinden wollen, daher müssen selbstständig verschiedene Ansätze zusammengesammelt werden.

Screens importieren
Zuletzt müssen die Prototypen nun mit Hilfe von JavaScript selbst erstellt werden, da gängige Prototyping Tools hier nicht kompatibel sind. Hier gibt es nun keine Integration in gängige Screendesign Programme mehr und die verschiedenen Elemente müssen von Hand importiert und eingesetzt werden.



Typischer projektablauf im Detail. Links sind Pain Points der einzelnen Projektschritte markiert.

Analyse bestehender Prototyping Tools

Als nächstes wurde eine Auswahl an Prototyping Tools und ihrer Funktionen analysiert. Framer Studio, Principle und Origami stellen die drei vorherrschenden Wege dar, Prototypen zu erstellen.

Die wichtigsten Funktionen der Programme wurden zusammengetragen, außerdem die Nachteile der Programme.

Für weitere Betrachtungen wurde Principle außen vorgelassen, da die Timeline basierten Prototypen nicht genügend komplex werden können.

Framer und Origamis Funktionen wurden anschließend mit Hilfe der Kano Analyse kategorisiert, um die wichtigsten Funktionen zu erkennen.

Wichtige Funktionen sind eine Dokumentation und Ressourcen, das gruppieren von Elementen, Shortcuts und eine Vorschau der Prototypen. Begeisterungsmerkmale sind eine Trennung von Screen design und der Programmierung, Funktionen zum Teilen und das Einfügen von Codebausteinen.

Programmierprinzip
Bei der Entscheidung, ob das Prototyping Tool patch basiert oder über klassischen Code funktionieren soll waren vor allem die Nutzergruppen entscheidend. Während es vereinzelt Studenten gibt, die mit einem Codebasierten Tool besser zurechtkommen, gibt es eine größere Zahl, denen mit erweiterten Hilfestellungen geholfen werden kann. Zusätzlich sind bei Patches Syntax Fehler einfacher zu erkennen. Aus diesen Gründen wurde für eine Patch basierte Umgebung entschieden.

Analyse von Framer als kombiniertes Screen Design und Prototyping tool		Das setze ich voraus	Das würde mich sehr freuen	Das ist mir egal	Das würde mich sehr stören	
Dokumentation	Vorhanden	✓				Basismerkmal
Dokumentation	Nicht vorhanden				✓	
Forum für Fragen	Vorhanden	✓				Basismerkmal
Forum für Fragen	Nicht vorhanden				✓	
Import von Sketch / Illustrator	Vorhanden		✓			Leistungsmerkmal
Import von Sketch / Illustrator	Nicht vorhanden				✓	
Verschiedene Artboards	Vorhanden	✓				Basismerkmal
Verschiedene Artboards	Nicht vorhanden				✓	
Liste von Elementen	Vorhanden	✓				Basismerkmal
Liste von Elementen	Nicht vorhanden				✓	
Layer gruppieren	Vorhanden	✓				Basismerkmal
Layer gruppieren	Nicht vorhanden				✓	
Layer verschachteln / Elementgruppen	Vorhanden	✓				Basismerkmal
Layer verschachteln / Elementgruppen	Nicht vorhanden				✓	
Elemente bearbeiten	Vorhanden	✓				Basismerkmal
Elemente bearbeiten	Nicht vorhanden				✓	
Bilder Importieren	Vorhanden	✓				Basismerkmal
Bilder Importieren	Nicht vorhanden				✓	
Export von Elementen	Vorhanden			✓		Unerheblichesmerkmal
Export von Elementen	Nicht vorhanden			✓		
Trennung von Screen design und Erstellen von Funktionalität	Vorhanden			✓		Unerheblichesmerkmal
Trennung von Screen design und Erstellen von Funktionalität	Nicht vorhanden			✓		
Cloudspeicher / öffnen	Vorhanden			✓		Unerheblichesmerkmal
Cloudspeicher / öffnen	Nicht vorhanden			✓		
Online teilen	Vorhanden		✓			Begeisterungsmerkmal

Die Features von Framer mit Hilfe der Kano Analyse klassifiziert.

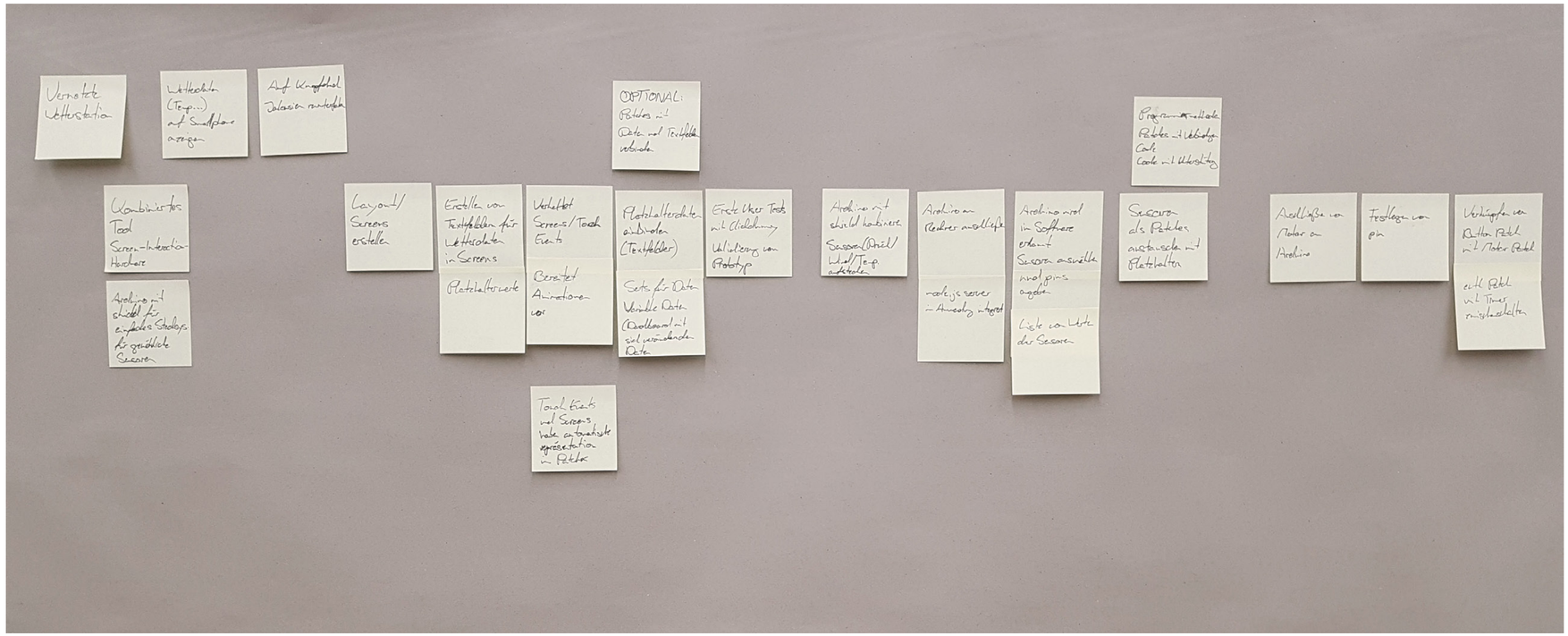
Erkenntnisse

Anhand der Erkenntnisse und dem Szenario wurden so die erforderlichen Funktionen für den Use Case aufgestellt.

Häufig kommt es in Projekten vor, dass das Screen design der Hardware voraus ist. Zum Testen von Prototypen wurde deshalb eine Funktion eingeführt, die es erlaubt anstelle von Sensordaten Platzhalterwerte auszugeben. So kann der Prototyp simuliert werden, ohne dass schon Hardware verfügbar sein muss.

Debugging ist eine weitere wichtige Funktion, um Anwendern zu helfen. Deshalb ist eine einfach zu bedienende und aufschlussreiche Debugging Funktion wichtig für die Anwendung.

Zuletzt sollen die Prototypen vorgeführt werden, dazu gibt es eine Exportfunktion, die die wichtigsten Entscheidungen auflistet.



Der beispielhafte Use Case im Detail. Der Nutzererstellt hier eine App, angefangen mit dem Layout bis zur Einbindung von Sensoren.

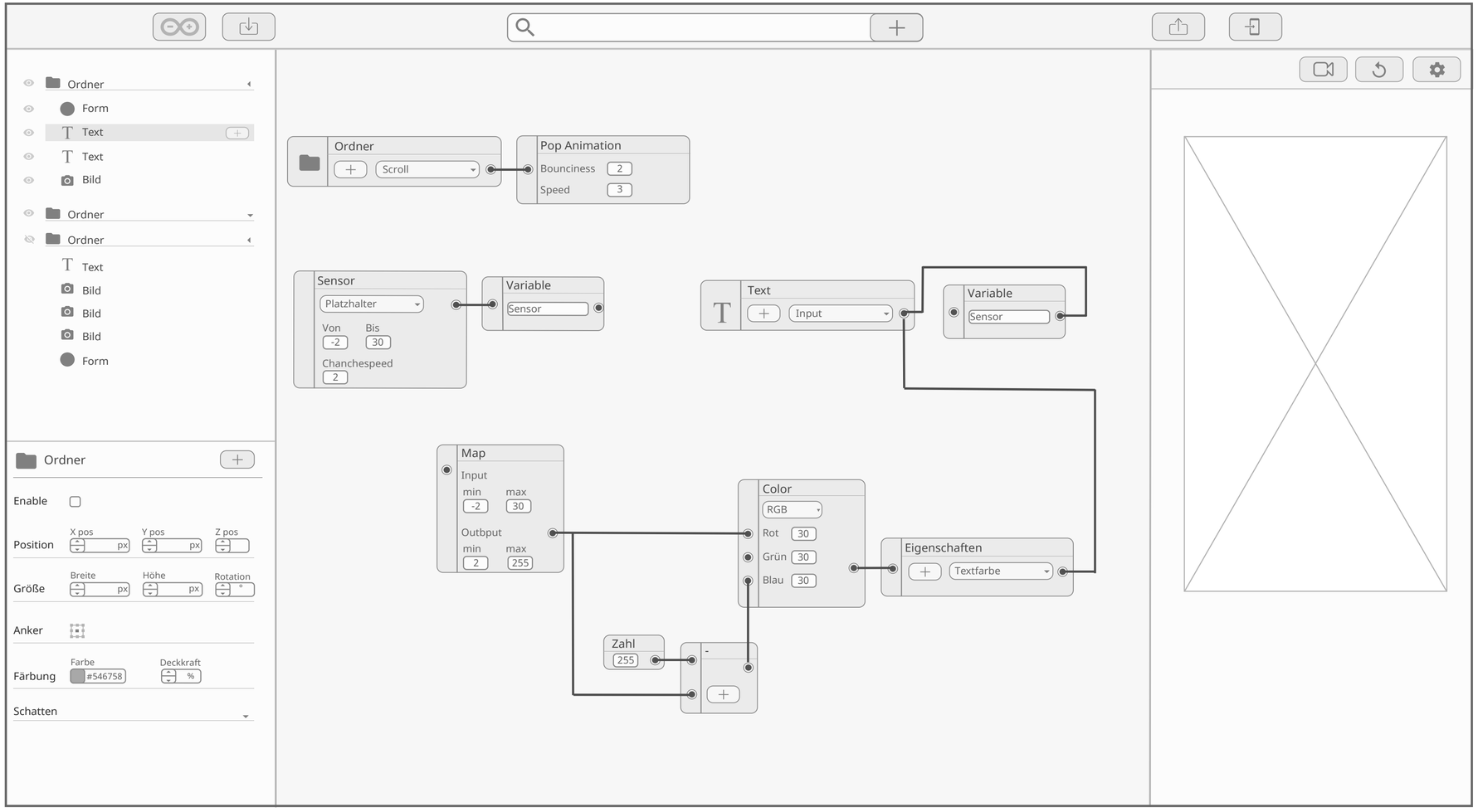
Wireframes

Auf Grund der vorangegangenen Recherche haben wir beschlossen ein Patchbasiertes Prototyping Tool mit den aus unseren Analysen hervorgegangenen Featurs zu erstellen. Dazu haben wir in wöchentlichen Iterationen Wireframes erstellt, um die Funktionen sinnvoll in unserem Programm zu integrieren.
Wir haben mit Wireframes in Graustufen begonnen und uns dann iterativ an unser Endprodukt herangetastet.

Erste Iteration

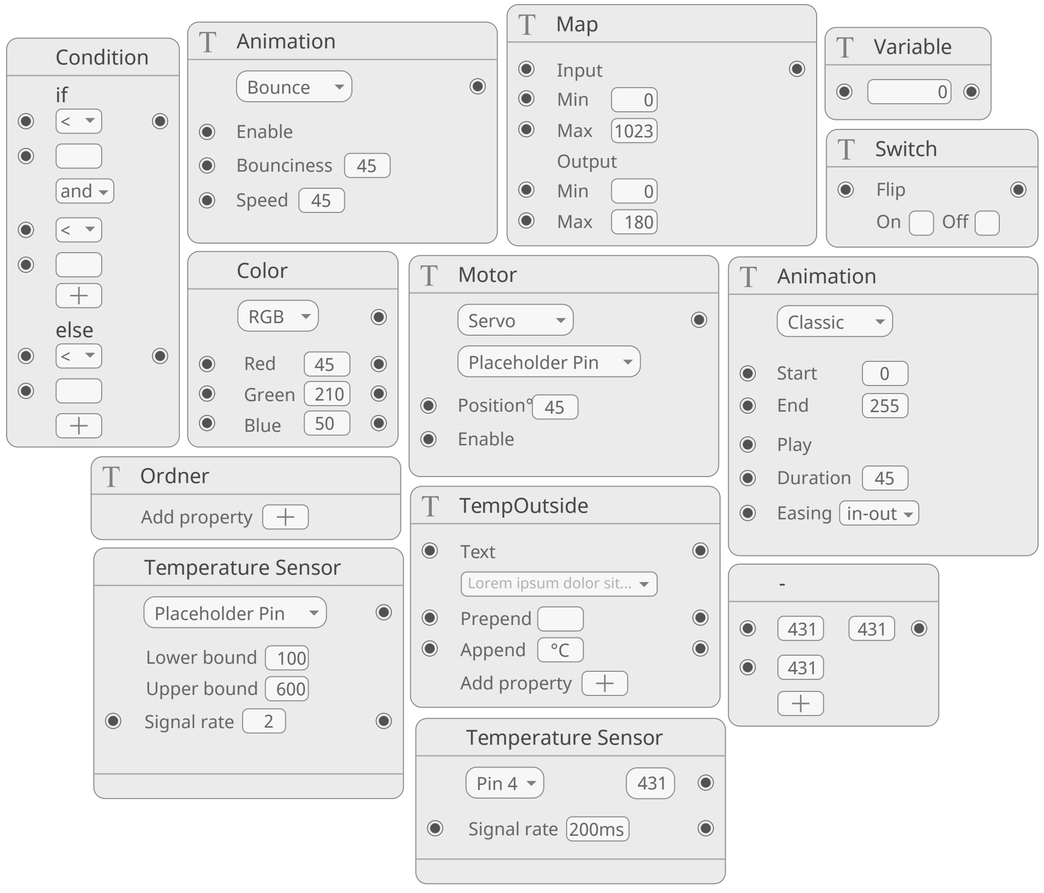
Nach dem wir mit Papier und stift unser grundlegendes Layout festgelegt hatten haben wir begonnen etwas mehr ins Detail zu gehen und an Hand unseres Usecases Wireframes zu bauen. Dabei war uns in erster Linie wichtig alle Funktionen einzubauen.

Das Programmfenster ist in eine Kopfleiste, zwei Randbereiche und eine zentrale Canvas unterteilt. Die Kopfleiste bietet grundlegende Funktionen, in den Randbereichen befindet sich der Prototyp und die zu bearbeitenden Elemente. Auf der Canvas wird dem Prototyp mit Hilfe von Patches Funktion verliehen.
Die Patches sind über gerade Linien mit einander verbunden. Die einzelnen Patches haben in dieser Version teilweise noch Patches, die gleichzeitig für In- undOutput zuständig sind.



Zweite Iteration

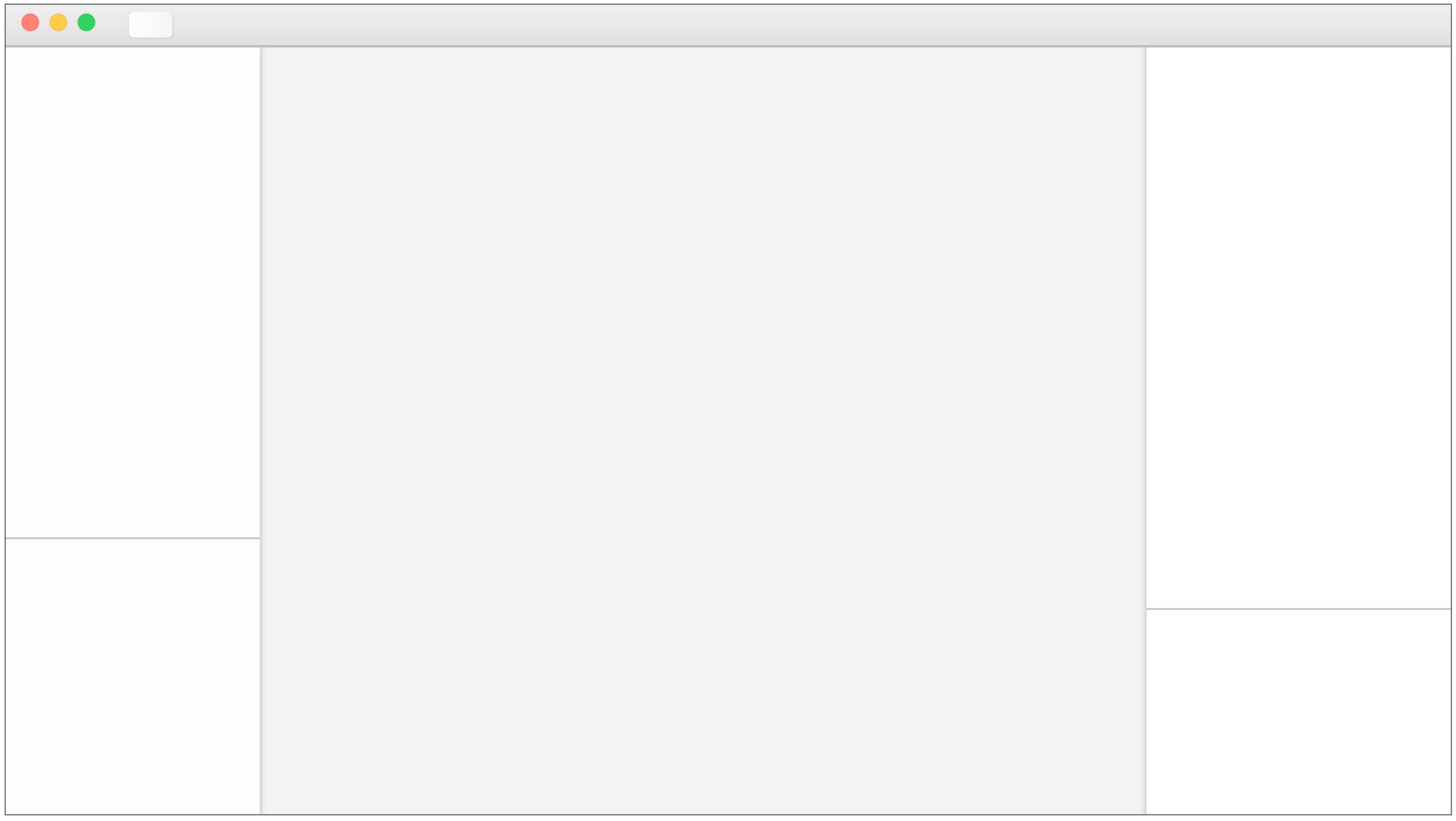
In der zweiten Iteration haben wir uns mehr mit den Patches an sich beschäftigt.
Wir haben festgelegt, das In- und Output Pins getrennt werden. Dabei sollen sich alle Input Pins links und alle Output Pins rechts befinden. So fließen die Daten, gemäß der der Leserichtung, von links nach rechts. Das hilft der Verstädlichkeit des gebauten Konstrukts, sowie der Übersichtlichkeit.
Des Weiteren haben wir alle Patches auf ihre Funktionalität überprüft und wenn nötig angepasst.



Zweite Iteration

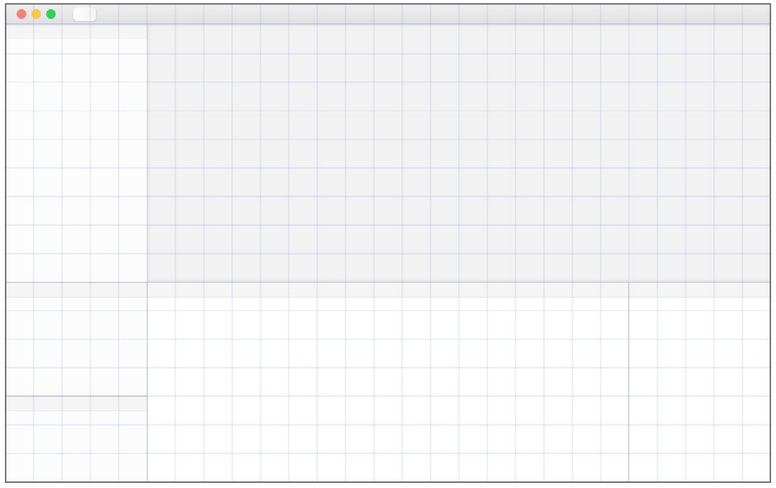
Im nächten Schrtitt haben wir uns mit dem Design und der Farbgebung unserer Anwendung beschäftigt.

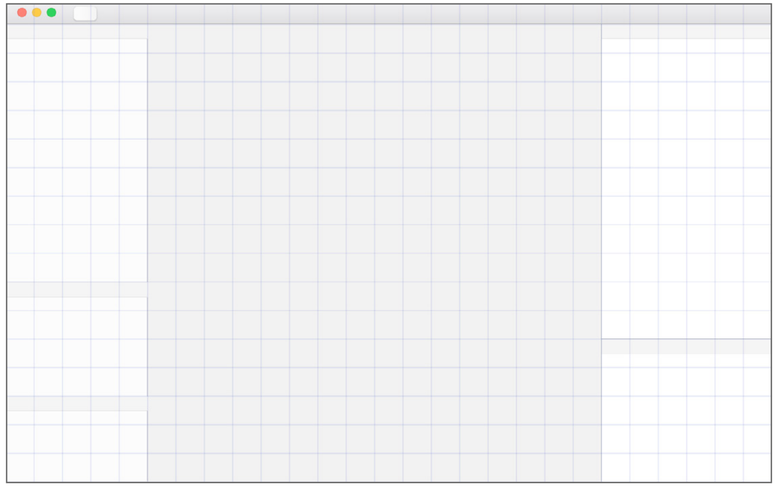
Wir haben uns dafür entschlossen uns an den Human Guidelines von Apple zu orientieren. Wir wollten, dass das Programm mit den hellen und neutralen Grautönen leicht und freundlich erscheint und Anwender nicht mit zu viel Farbe oder Dunkelheit im Hintergrund erschlägt.

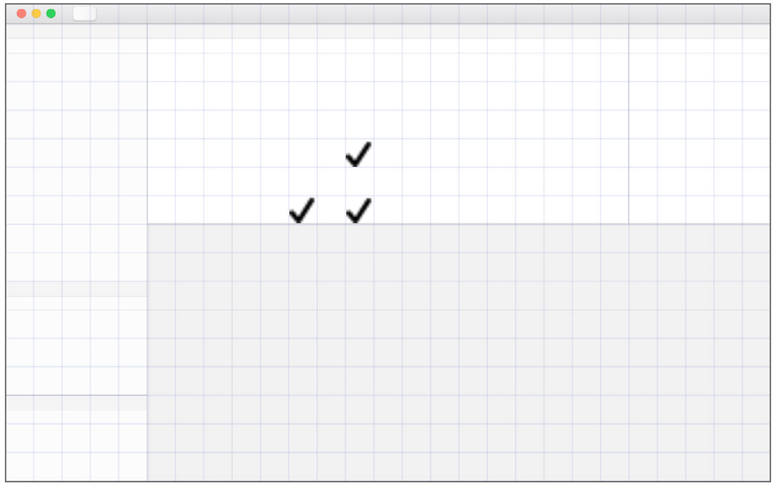


Zweite Iteration

Layout
Wir haben ein Raster erstellt in dem man die Fenster, die Inhalte und zB. die Prototypvorschau, die in vorherigen Entwürfen nur rechts und links angeordnet werden konnten, nun flexibel an jedem Fensterrand anordnen kann.
Das hilft beispielsweise dabei einen 16:9 Prototypen platzsparend (wie im linken Beispiel) anzuzeigen.
Zudem kann so jeder Nutzer seinen Arbeitsplatzselbst einrichten.







Zweite Iteration

Patches
Weiter haben wir uns mit der Farbgebung und dem Aussehen unserer Patches beschäftigt
.
Wir haben verschiedene Farbvariationen (wie auf der nächsten Seite zu sehen) gebildet und Befragungen von Komilitonen sowie kleinere Brauchbarkeitstest und Analysen durchgeführt. Dabei haben wir Wert darauf gelegt so viele verschiedene Ansätze wie in der gegebenen Zeit möglich zu testen, um am Ende das bestmögliche Ergebnis zu erhalten.


Motor
Servo
Placeholder Pin
Position 45 °
Enable
TempOutside
Text
Lorem ipsum dolor sit...
Prepend
Append °C
Add property
Temperature Sensor
Placeholder Pin
Lower bound 100
Upper bound 600
Signal rate 2
Animation
Bounce
Enable
Bounciness 45
Speed 45
Ordner
Add property
Color
RGB
Red 45
Green 210
Blue 50
Map
Input
Min 0
Max 1023
Output
Min 0
Max 180
TempVal
0
-
255 431
431
Condition
if
and
else

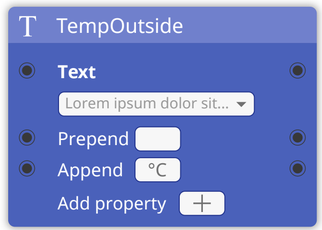
Zweite Iteration

Patches

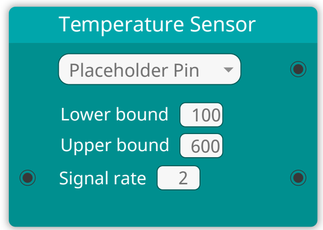
Letzendlich haben wir uns für ein gesättigteres, nicht allzu aufregendes Farbschema entschieden, damit die Anwendung bei Nutzung nicht zu grell wird.

Wir haben die Patches in vier Kategorien unterteilt: Elemente, Hardware, Neutrale und Aussehen. Elemente haben einen ruhigen Blauton, da sie ohne weitere Patches erst einmal statisch sind. Hardware Patches verwenden das Grün das auch Arduino verwendet, um eine semantische brücke zum Arduino zu knüpfen. Neutrale Patches haben ein neutrales Grau und das Aussehen betreffende Patches ein „aufregendes“ Lila.

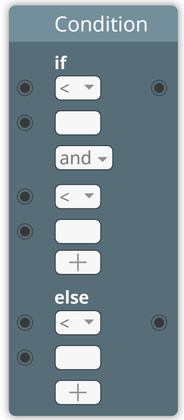
Elemente



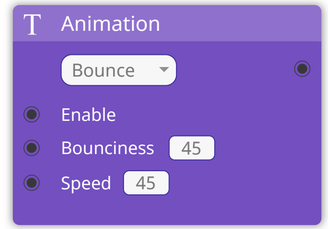
Hardware



Neutrale

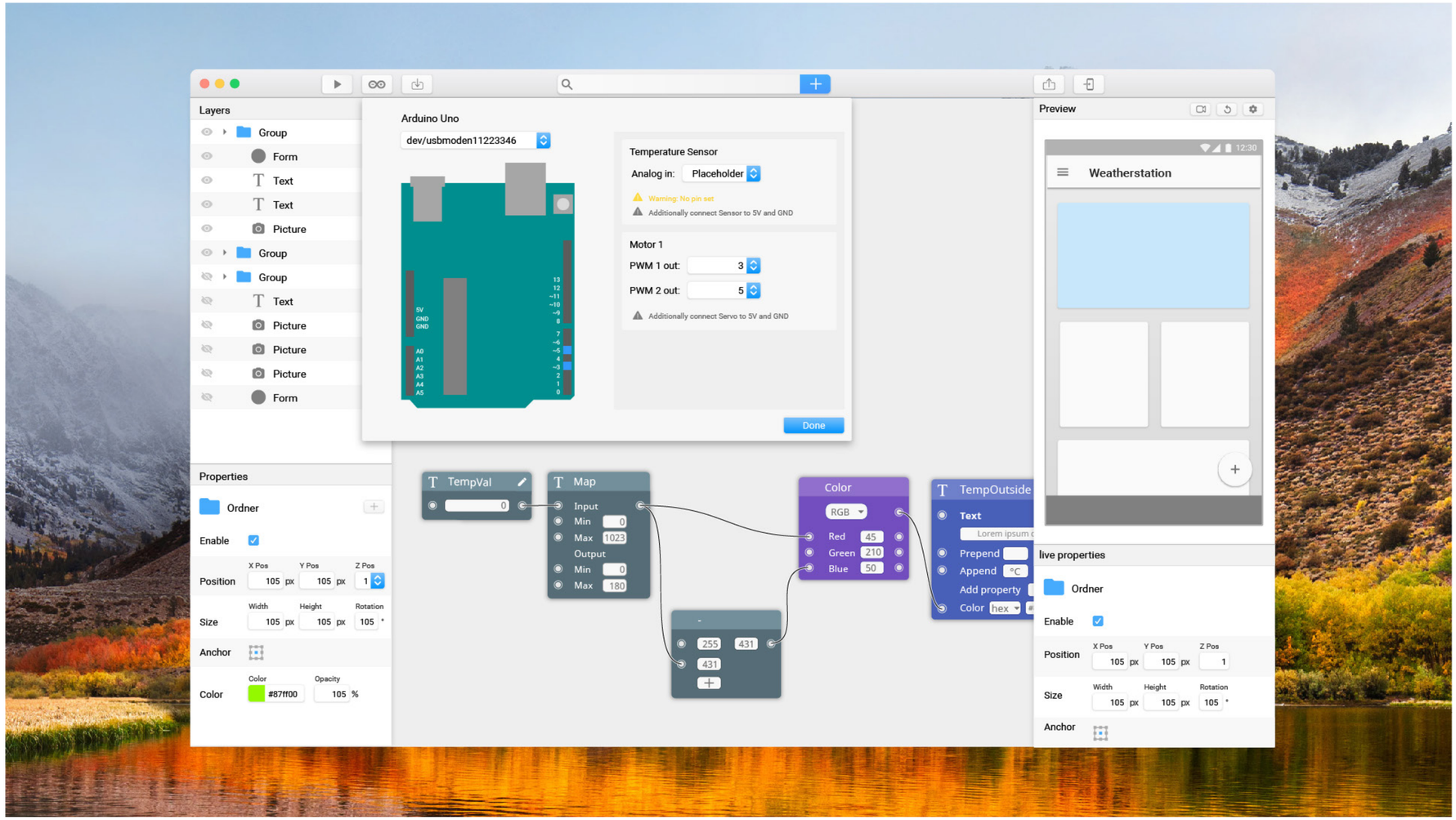


Aussehen



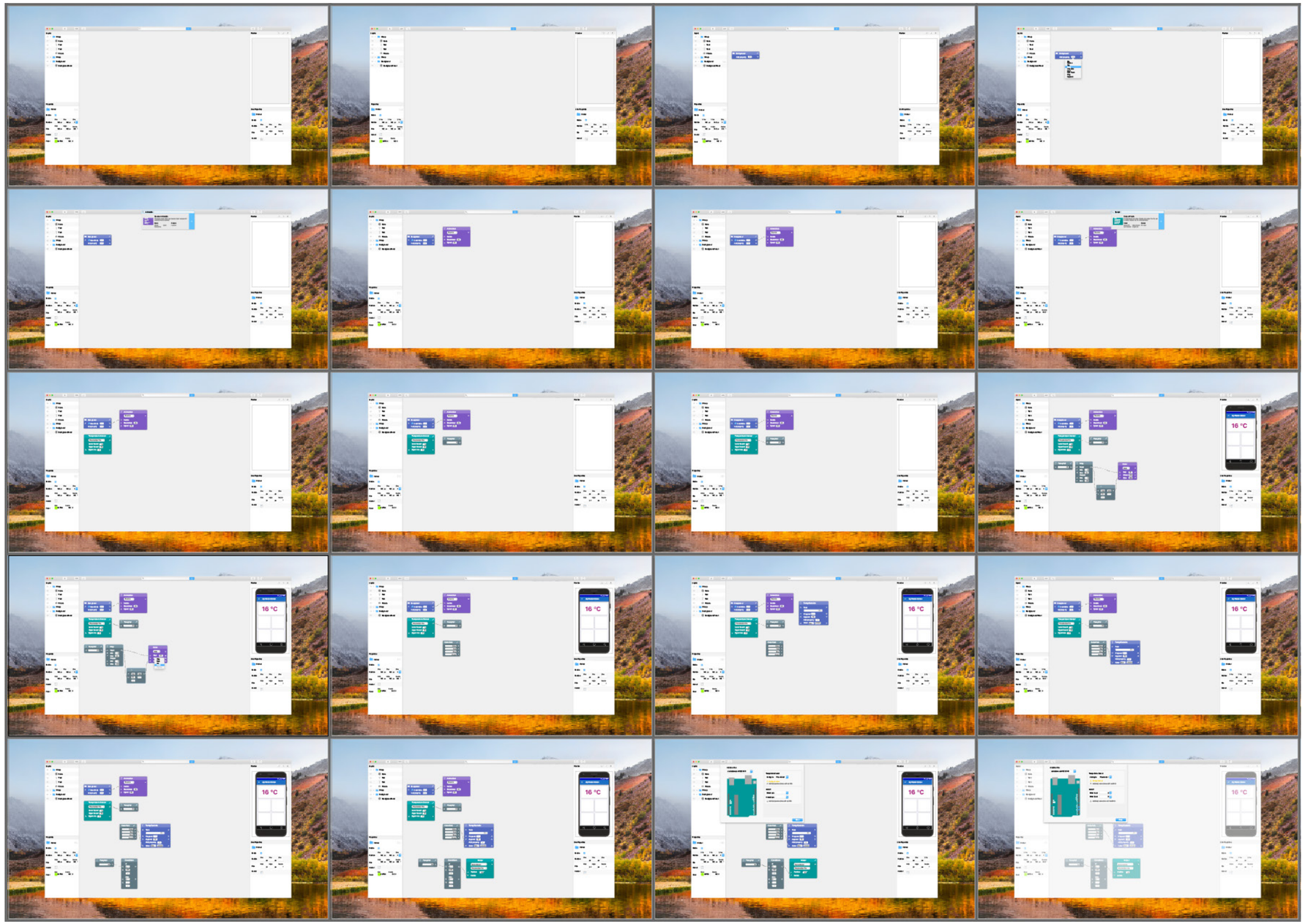
Dritte Iteration

In unserer dritten Iteration haben wir erstmals alle Komponenten zusammengefügt und auch die restlichen Elemente an unsere festgelegten Richtlinien angepasst. Zudem haben wir begonnen die, für unseren Use Case relevanten, Popups zu gestalten. Wir haben, um die Verbindungen der einzelnen Patches zueinander übersichtlicher zu gestalten, die Pins hell eingefärbt. Die Verbindungslinien, die im Vergleich zu den vorherigen Entwürfen nun nicht mehr eckig sind, bekamen eine Kontur, um sie etwas von den Patches und dem Hintergrund abzuheben.
Wir haben uns in diesem Schritt auch für eine Schrift entschieden. Gemäß der Human Guidelines verwenden wir in unserer Anwendung die SF Pro von Apple.



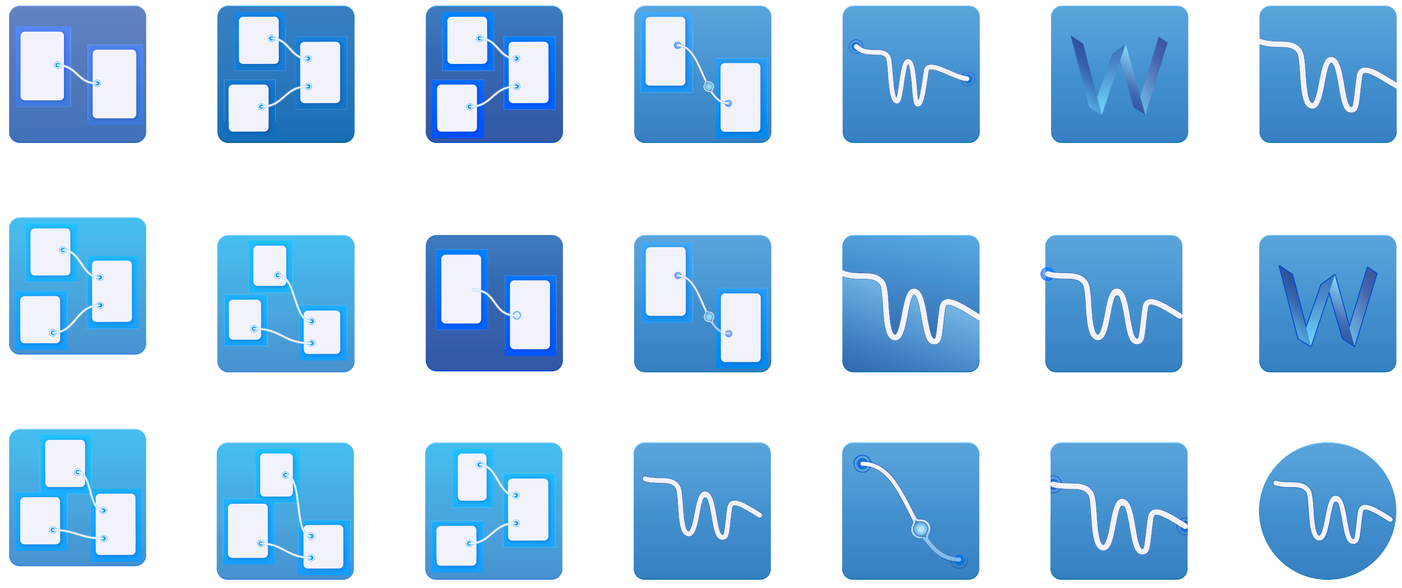
Letzte Iterationen

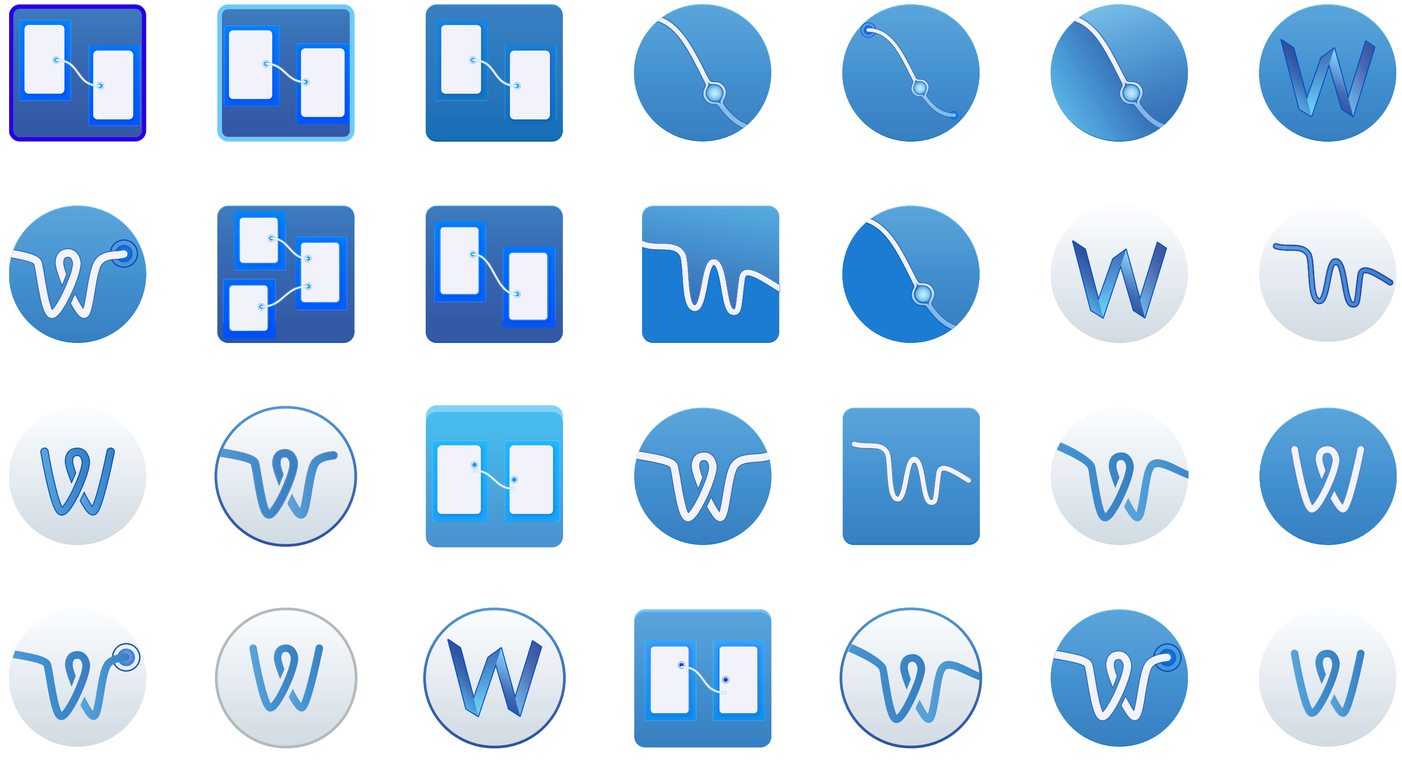
In unseren letzten beiden Iterationen haben wir nur noch Kleinigkeiten am Aussehen der Screens geändert und uns mehr darauf konzentriert alle für unseren Use Case relevanten Screens zu erstellen. Wärend dieser letzten Iterationen haben wir die Screens nocheinmal komplett auf ihre Sinnhaftigkeit und Funktionalität überprüft um letzte Fehler zu beheben.



Logo

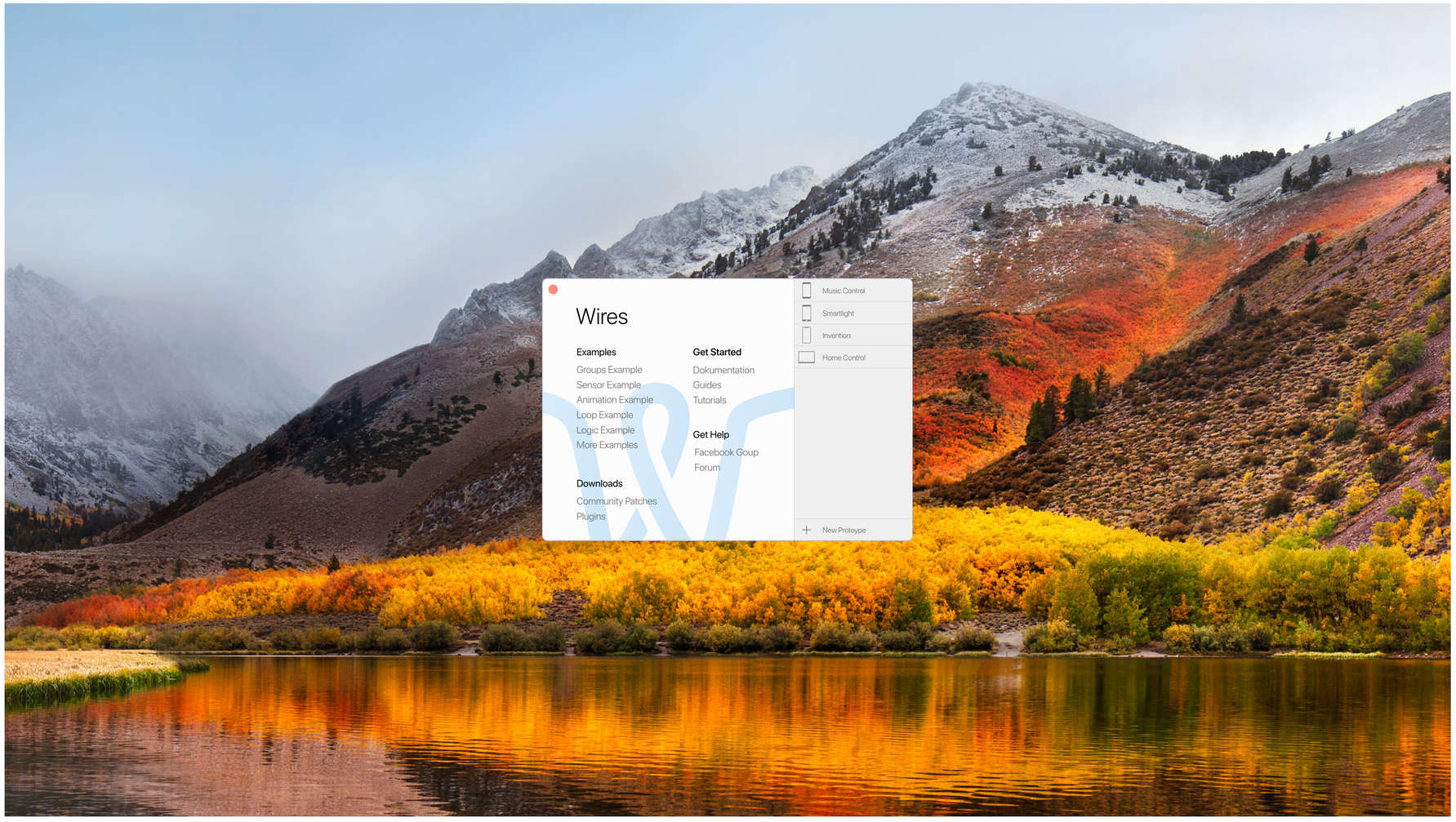
Um unserem Produkt eine Identität zu geben haben wir ein Logo gestaltet. Alle Logoentwürfe orientieren sich an den im Programm verwendeten Farben.
Für das Logo war es uns wichtig eine Beziehung zwischen dem Namen und unserer Anwendung herzustellen.
Unser erster Ansatz war es eine stark vereinfachte Form unserer Patches als Logo zu verwenden. Das wurde aber sehr Kleinteilig und hatte nicht viel mit unserem Namen zu tun. Deshalb haben wir als nächstes versucht den Debug Modus, der einzigartig in unserem Programm ist, als Logo in Form von Perlen, die durch ein Kabel wandern, zu nutzen.
Schließlich haben wir uns mehr auf den Namen unserer Anwendung konzentriert und uns daher für ein „W“, dass aus einem Kabel gelegt wird entschieden.





Anwendung

Die Prototyping App ist eine native macOS Anwendung und basiert auf den macOS Visual Design Guidelines. Der Name Wires und das Icon symbolisieren die Verbindungslinien mit denen die Patches verknüpft werden.



Start Panel

Nach dem Start zeigt Wires ein Panel, mit wichtigen Funktionen zum Einstieg, benutzten Dateien und Links zu Beispielen und Hilfen.

Neue und erfahrene Nutzer können so schneller an relevante Bereiche kommen.

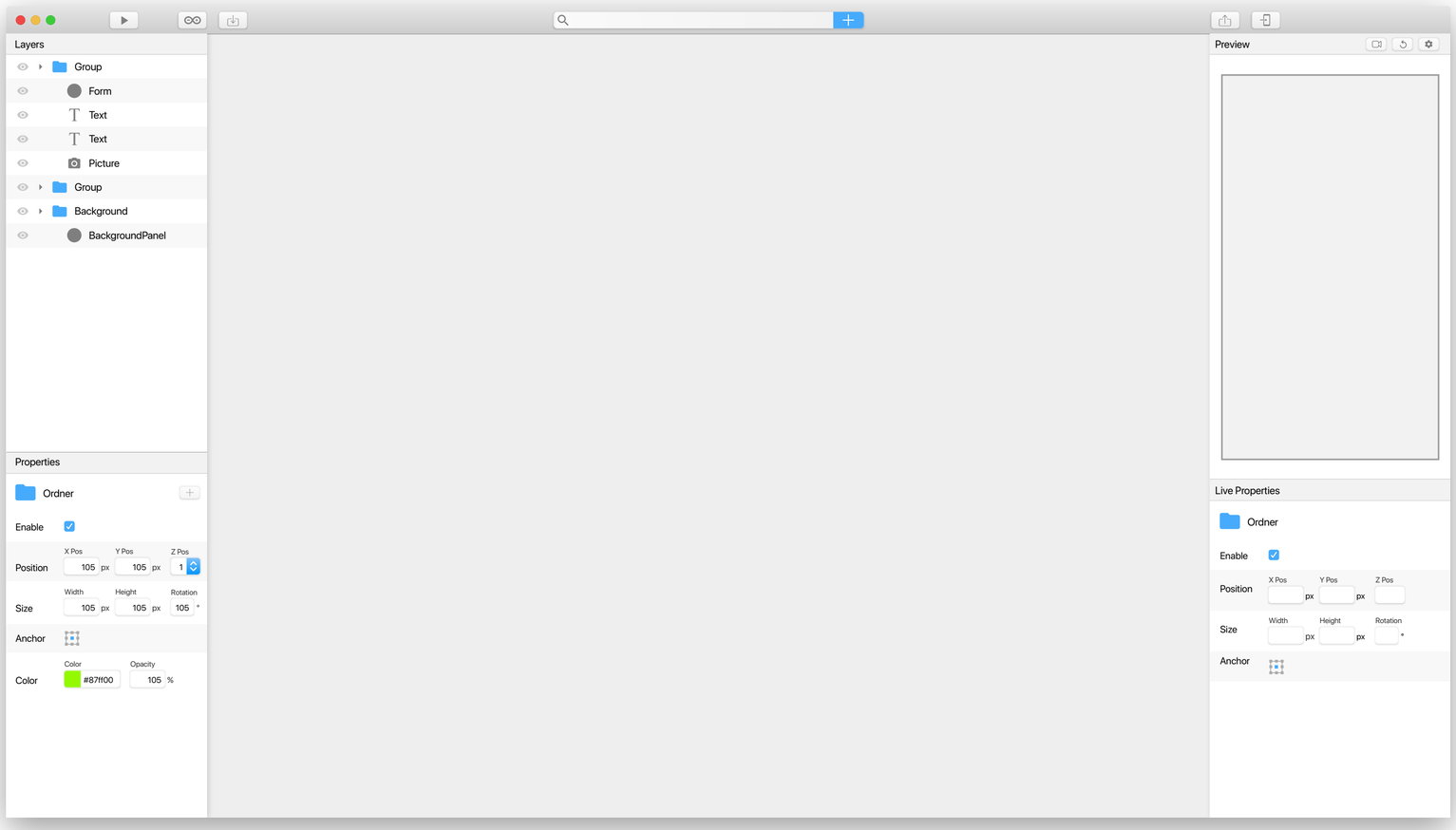
Hauptfenster

Wenn ein neues Projekt erstellt wird ist die Anwendung zunächst leer. In der leeren Ebenenliste fordert eine Grafik zum Importieren per Drag and Drop auf. Ist eine Datei importiert werden die Ebenen angezeigt. Über einen Plus-Button können die Ebenen als Patches auf dem Canvas platziert werden.

Hauptfenster

Suche
Über die Suchfunktion können Patches, die nicht Ebenenelemente sind zum Prototypen hinzugefügt werden.

Ebenen
Ist ein Projekt mit gestalteten Screens aus Sketch oder Illustrator importiert, werden die Ebenen in der Ebenenliste angezeigt. Sie können von hier aus als Patch im Prototypen erzeugt werden.



Vorschau
Im Vorschaufenster wird der aktuelle Prototyp angezeigt. Die Vorschau läuft in Echtzeit ab, wenn Werte im Canvas geändert werden können sie hier direkt betrachtet werden.

Properties
Ausgewählte Ebenen zeigen hier ihre Eigenschaften, wie Dimensionen oder Farbe an. Die Werte können auch verändert werden, wenn die importierten Werte nicht stimmen sollten.

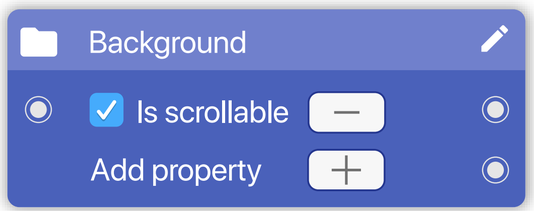
Live Properties
Gleichzeitig zum Prototypen werden hier die Eigenschaften von ausgewählten Elementen angezeigt. Die Werte werden aktualisiert, je nachdem was im Prototyp geschieht.

Canvas
Auf dem Canvas geschieht die eigentliche Arbeit, das Programmieren der Funktionen des Prototyps. Es nimmt entsprechend mehr Raum ein als die anderen Panels.

Patches

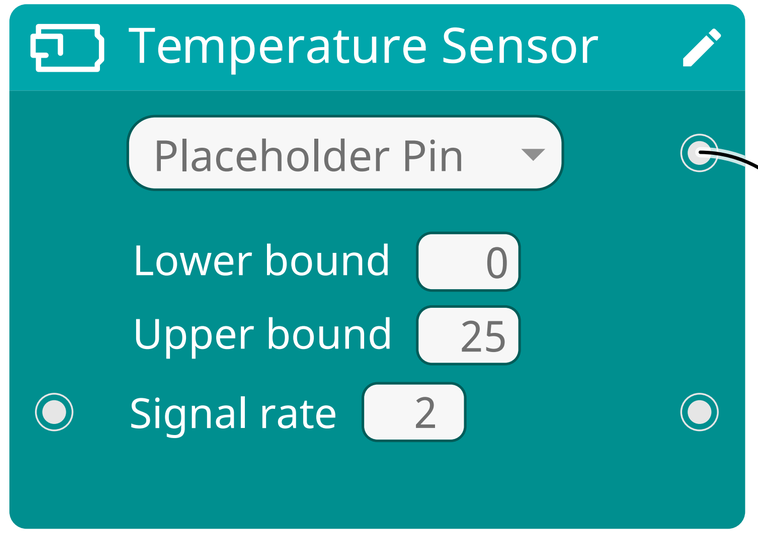
Patches repräsentieren die verschiedenen Elemente, die im Prototyp angezeigt werden können, Sensoren der Hardware, Eigenschaften von Elementen oder Logik. Je nachdem zu welcher Kategorie sie gehören sind sie zu Wiedererkennung eingefärbt.

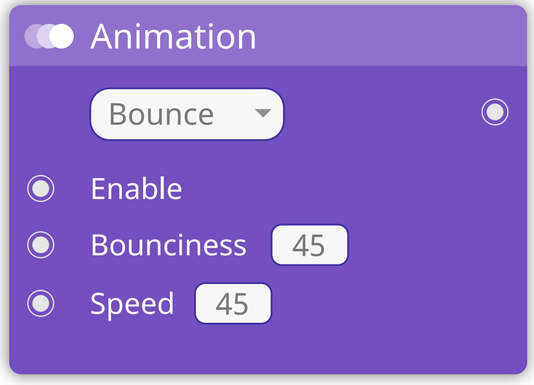
Layout
Patches bestehen aus einem Header mit Name und Icon, und einem Body aus Eigenschaften. Die Eigenschaften können verschiedene Formen annehmen, von Textfeldern bis Checkboxes.



Eigenschaften hinzufügen
Über die Plus-Buttons, beziehungsweise Minus-Buttons können Eigenschaften zu Patches hinzugefügt oder weggenommen werden. Formelemente können viele verschiedene Eigenschaften besitzen, die nicht so oft eingesetzt werden um zu rechtfertigen, sie immer zu zeigen.

Sensoren
Sensoren können neben von Arduino Boards empfangenen Daten auch Platzhalterdaten ausgeben. Um sinnvolle Werte für die unterschiedlichen Prototypen zu bekommen, können die Abtastrate, Ober- und Untergrenze festgelegt werden. Der Patch gibt dann zufällige Werte aus.

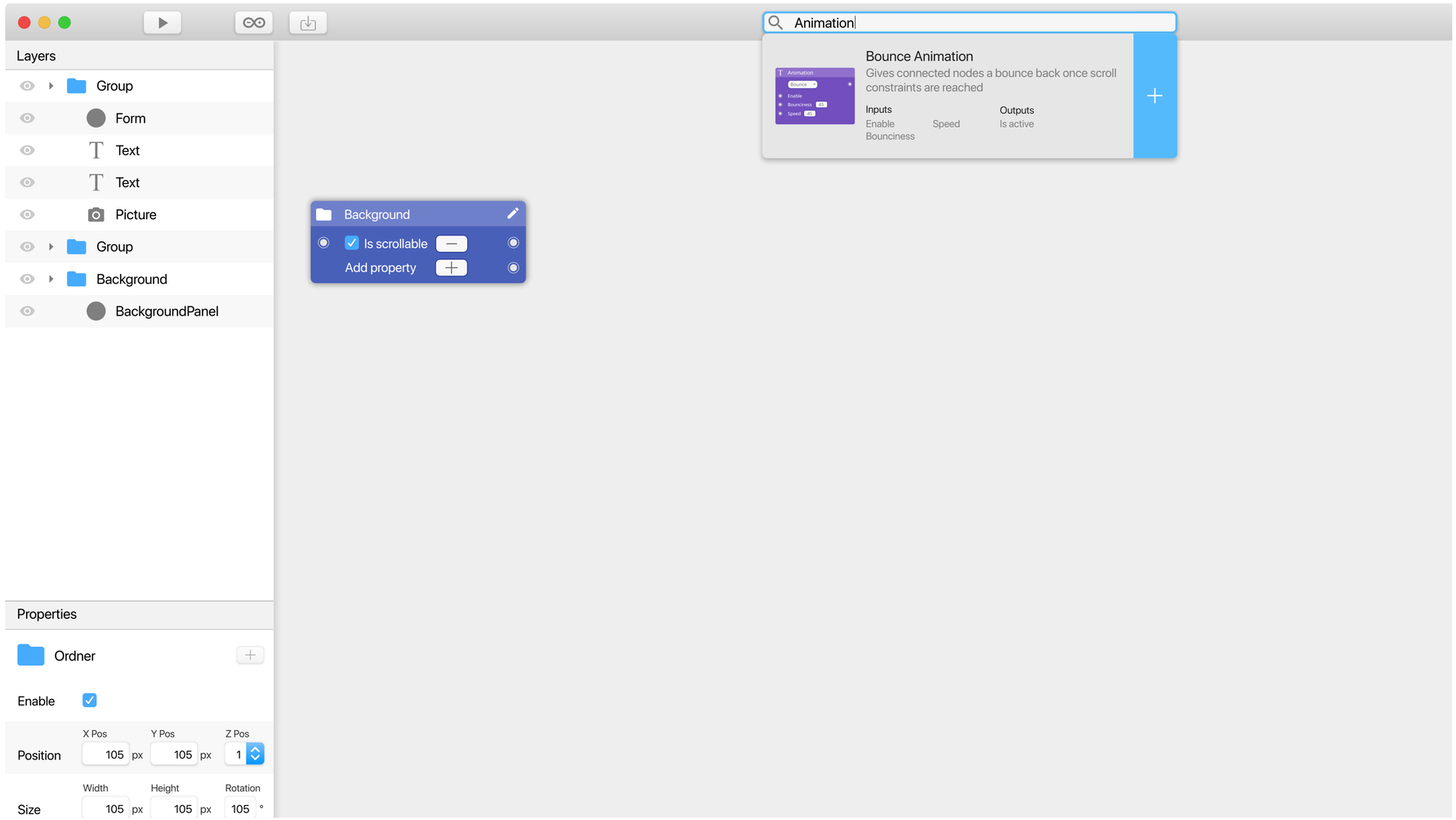




Inputs und Outputs
Links und rechts der Eigenschaften befinden sich Ein- und Ausgänge der Daten. Somit ergibt sich eine natürliche Leserichtung von links nach rechts.

Patches erzeugen

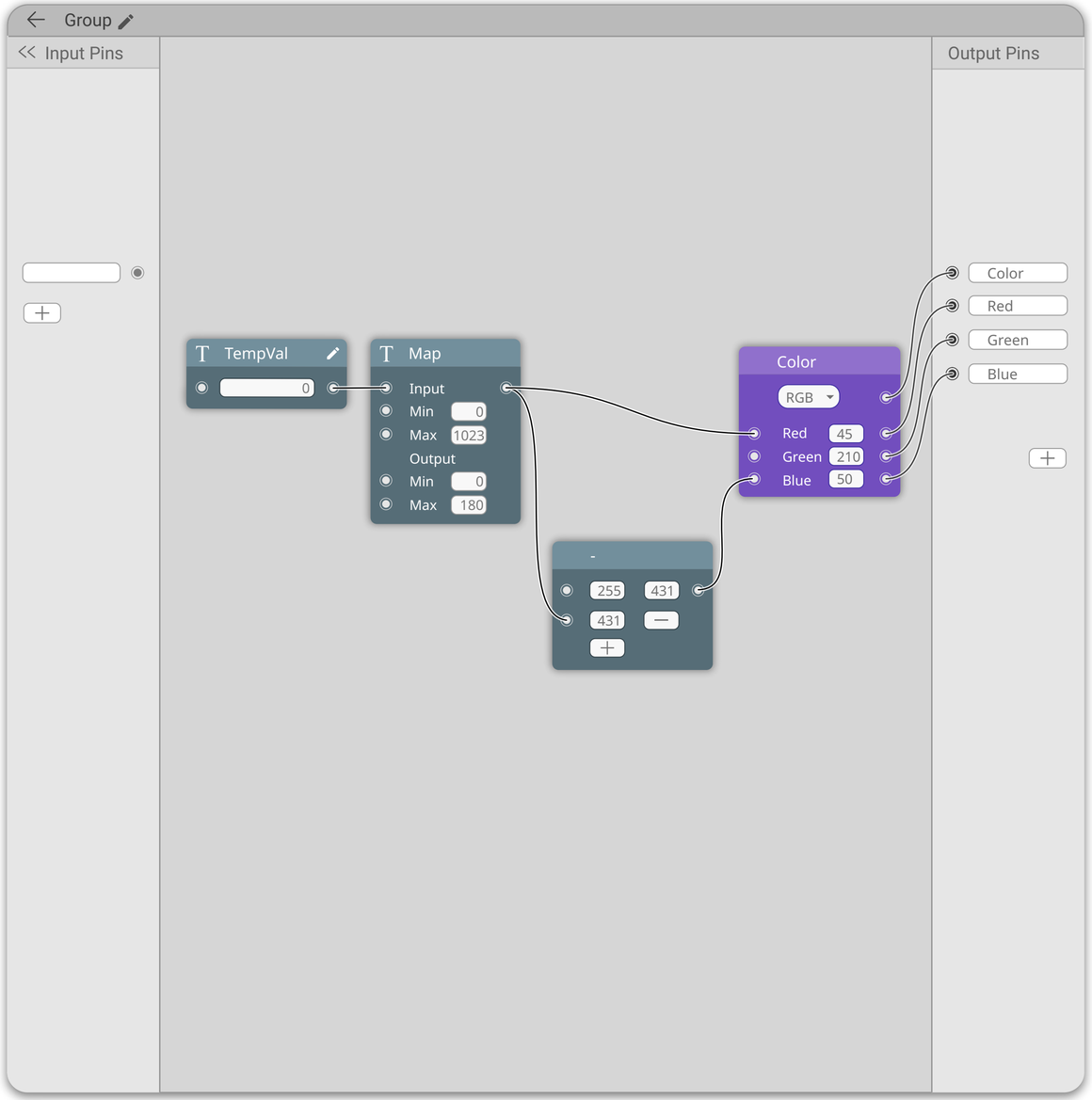
Patches können auf drei verschiedene Arten erzeugt werden. Wie erwähnt können sie aus der Layer Liste erstellt werden. Zusätzlich gibt es eine Suche für alle anderen Patches. Die Suche zeigt Vorschläge mit Namen, Beschreibung, Vorschau und Liste der Inputs und Outputs.



Gruppieren

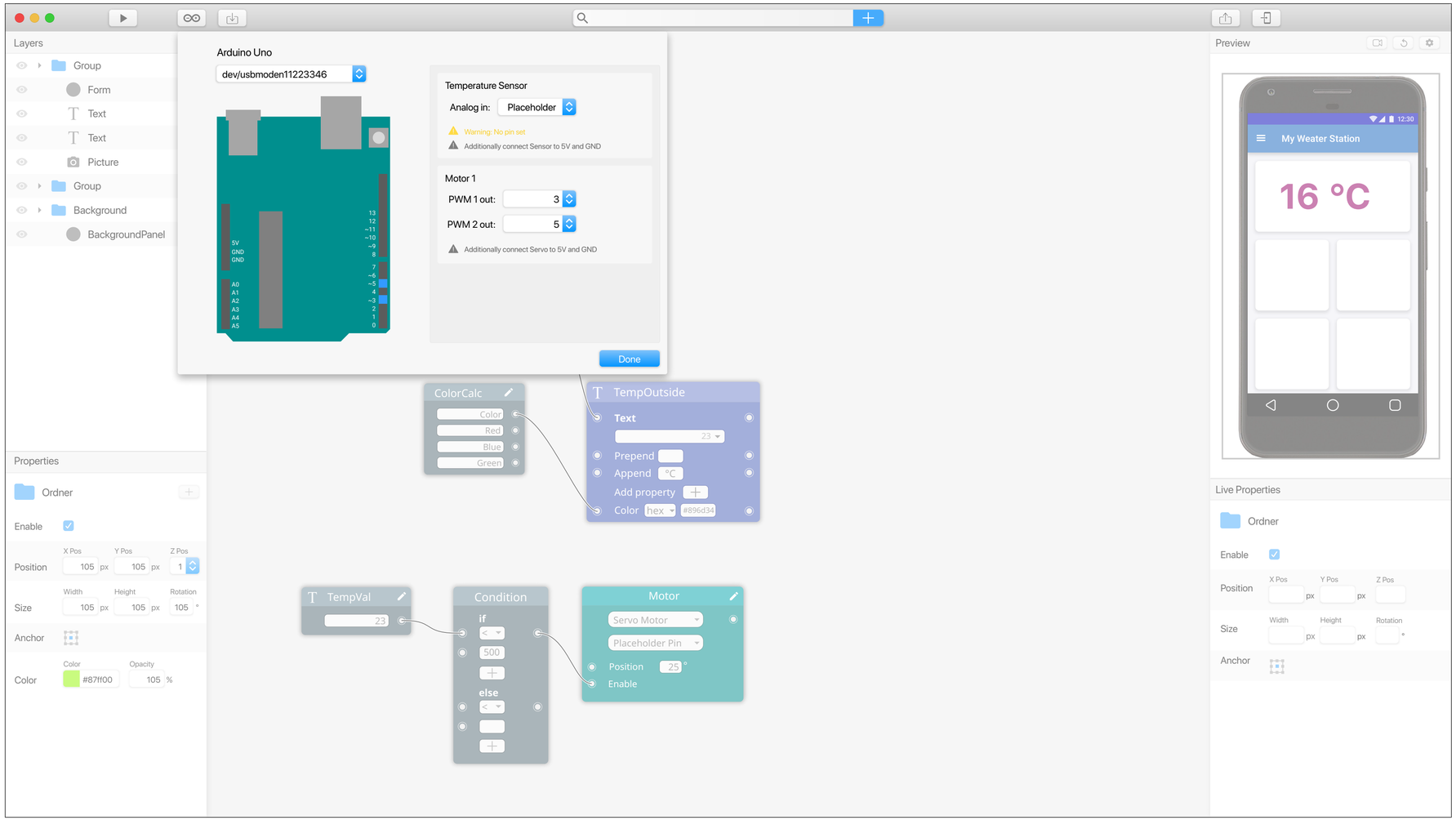
Mehrere Patches können zusammengefasst werden um die Übersicht zu erhöhen und den Prototypen zu vereinfachen.

Werden Patches gruppiert, zoomt die Canvas in eine neue Ansicht. Hier können die Outputs und Inputs der Gruppe festgelegt werden. Sie werden einfach mit den Outputs und Inputs der Patches verbunden und Übernehmen deren Namen. Zuletzt kann die Gruppe benannt werden. Wenn die Gruppe vollständig ist kann die Ansicht verlassen werden und zoomt wieder zurück zum Canvas. Die Gruppe ist nun ein eigener Patch, der wie gewohnt verwendet werden kann.



Arduino Panel

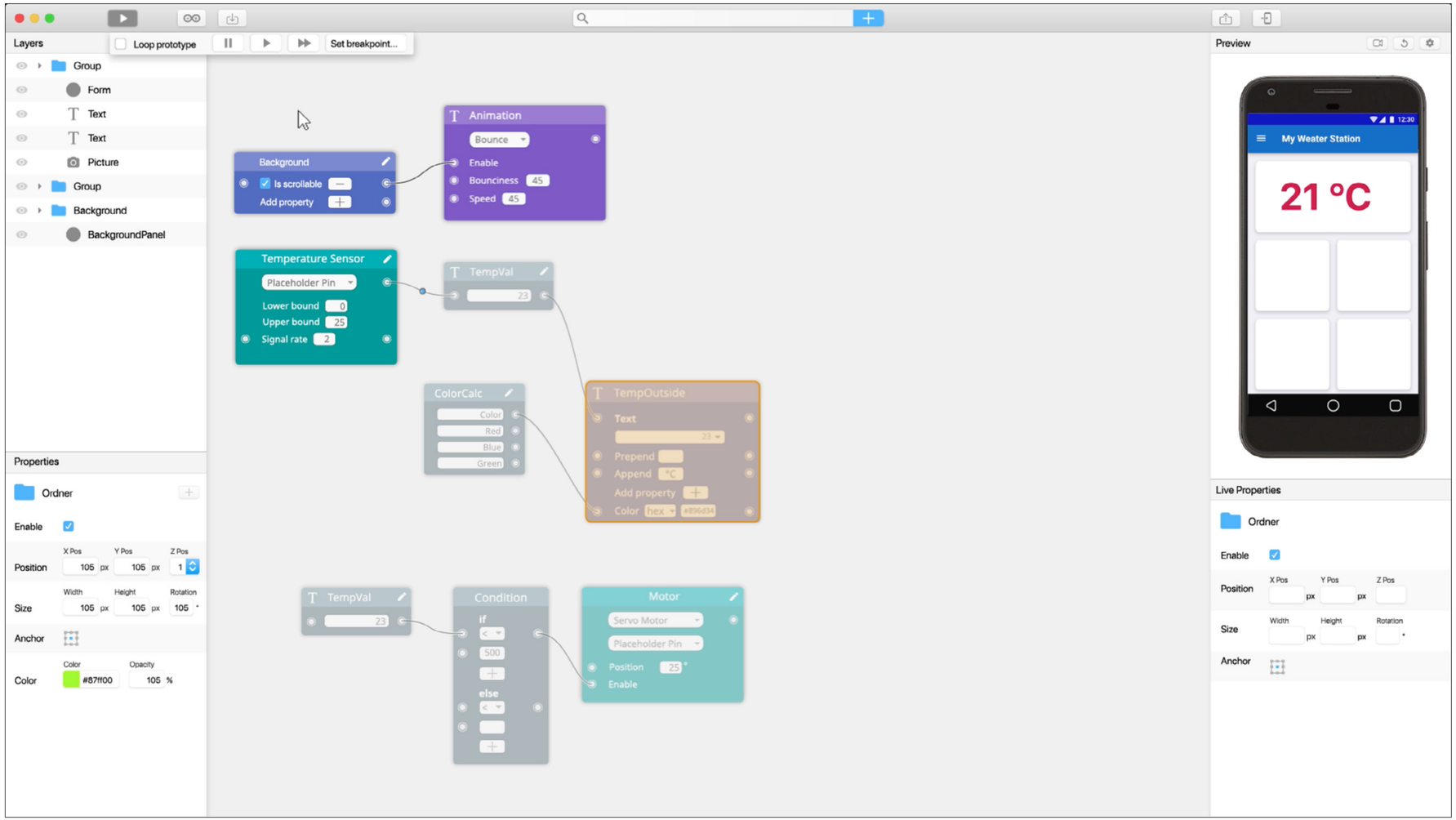
Um alle Sensoren auf einmal konfigurieren zu können gibt es ein Dialog in dem alle Hardwarepatches und ihre Pins aufgelistet werden. Hier kann das aktive Arduino Board ausgewählt werden und eine Vorschau des Boards und der belegten Pins wird gezeigt. Mit wenigen Klicks lassen sich hier für die ganze Hardware des Projekts die Pins setzen. Falls Platzhalter gesetzt sind werden Warnungen angezeigt.



Belegte Pins werden in der Vorschau des Boards blau markiert.

Debugging

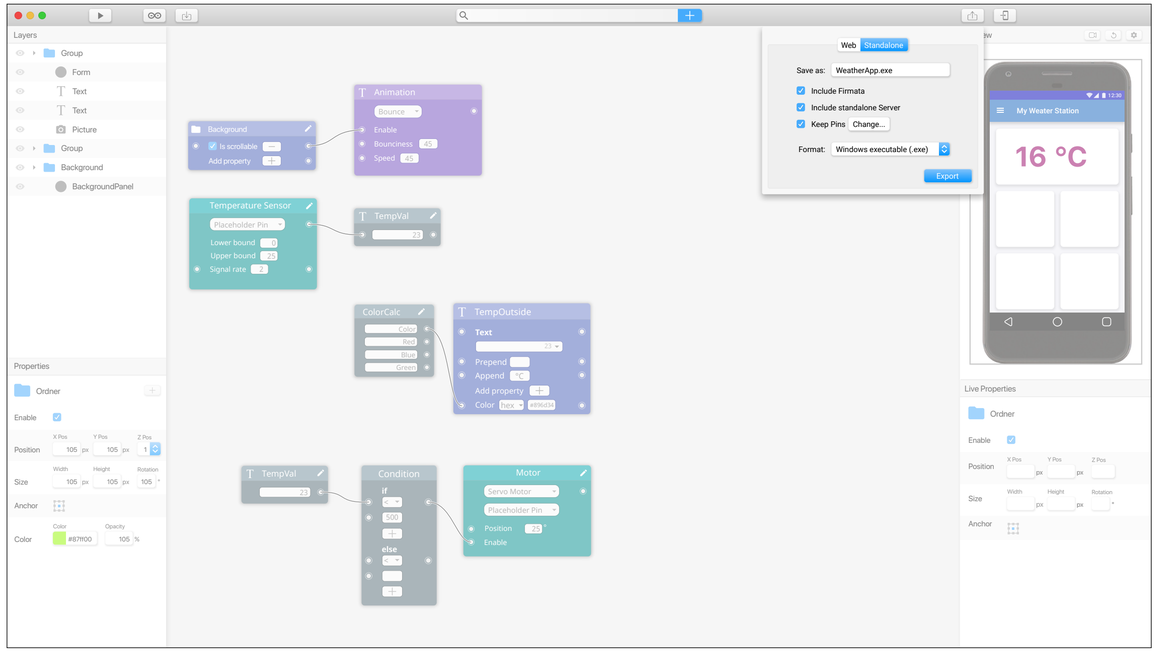
Um den Prototypen zu testen kann im Debug Modus der Datenfluss zwischen den Patches verlangsamt und visualisiert werden. Wird der Prototyp hier gestartet laufen Perlen die Verbindungen ab, je nachdem wo gerade Werte verschickt werden. Außerdem lassen sich Patches markieren und wie gewohnt als Breakpoints setzen. Die Ausführung des Prototyps stoppt dann sobald eine Perle die jeweiligen Patches erreicht.



Aus dem Sensor Patch wandert eine Perle Richtung Variable. Das Textfeld „TempOutside“ ist als Breakpoint markiert. Hier stoppt die Ausführung.

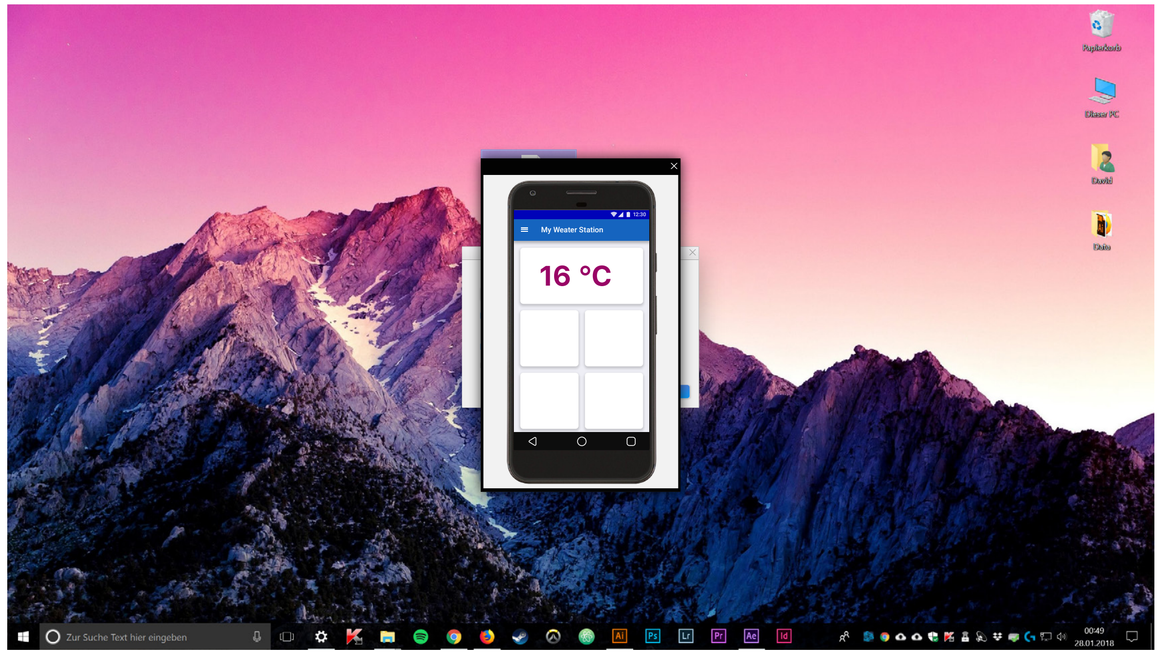
Export

Über einen Export Dialog lassen sich standalone Versionen der Prototypen erstellen. Es lassen sich hier noch einmal die Pins einstellen, je nachdem ob die Empfänger Hardware zur Verfügung haben oder nicht. Alternativ frägt der Prototyp auf dem Zielrechner noch einmal nach. Der exportierte Prototyp besteht aus einer einzelnen Vorschau, ohne die Möglichkeit den Prototypen zu verändern, um die Anwendung leicht zu halten.



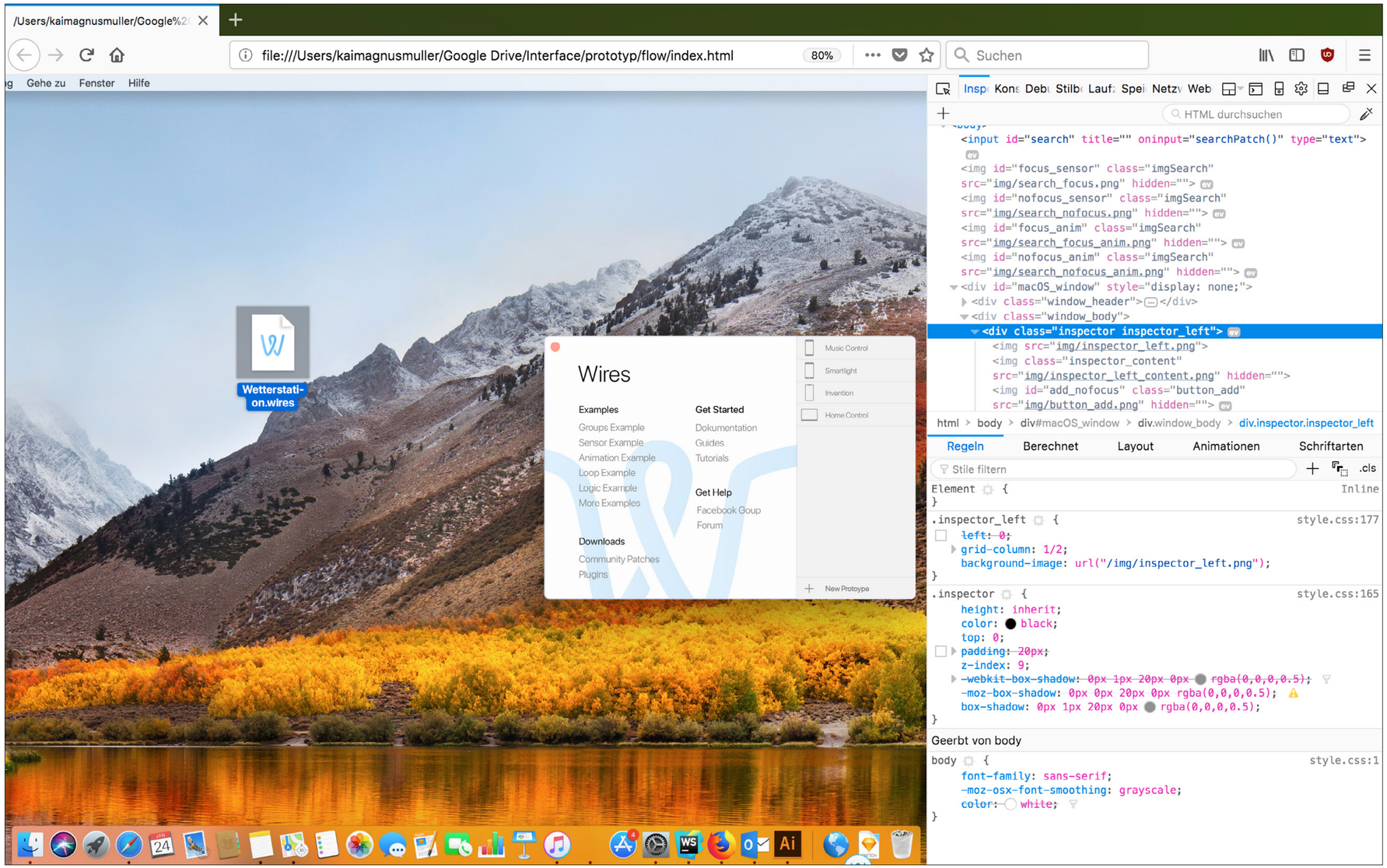
Über den Export Dialog können die Prototypen auf anderen Geräten wiedergegeben werden.

Zum Beispiel auf einem Windows PC



Prototyp

Um die Anwendung zu testen und vorzuführen, wurde ein Prototyp entwickelt. Der Prototyp basiert auf JavaScript, mit einem erwähnten node.js Server, der die Daten des Arduino Weiterleitet. Der Prototyp erlaubt es Patches zu erstellen, suchen und verknüpfen. Komplexere Funktionen, wie das Gruppieren und exportieren von Prototypen wurde mit Hilfe von After Effects visualisiert.



Fazit

Wires ist ein Prototyping Tool, das es zum ersten Mal erlaubt Screen design und Hardware Prototyping zu verbinden. Dazu wurden die Funktionen bestehender Programme analysiert und realistische Use Cases aufgestellt, um die nötigen Funktionen zu finden und zu gestalten. Nutzer können auf einfache Art und Weise Prototypen erstellen und bekommen bei jedem ihrer Schritte genügend Hilfestellung um auch mit wenig Erfahrung sicher an Ergebnisse zu kommen.