Interactex

# Overview

Interactex is a toolset to create iPhone and iPod applications that interact with hardware components such as sensors and actuators. Figure below shoes the Architecture of the toolset.

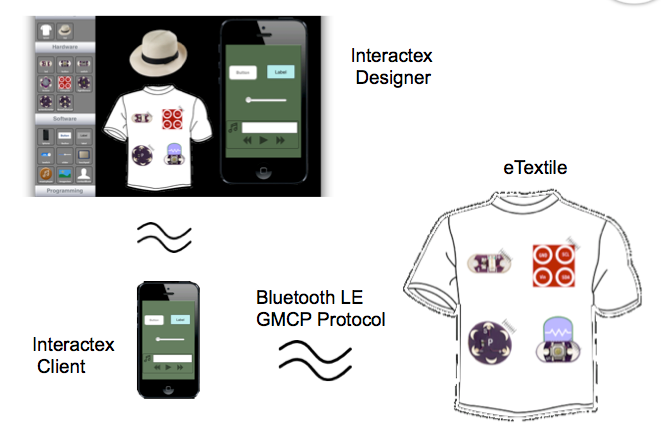


Figure 1 Architecture of the Interactex Environment

The Interactex Designer is initially used in order to visually create a Software.

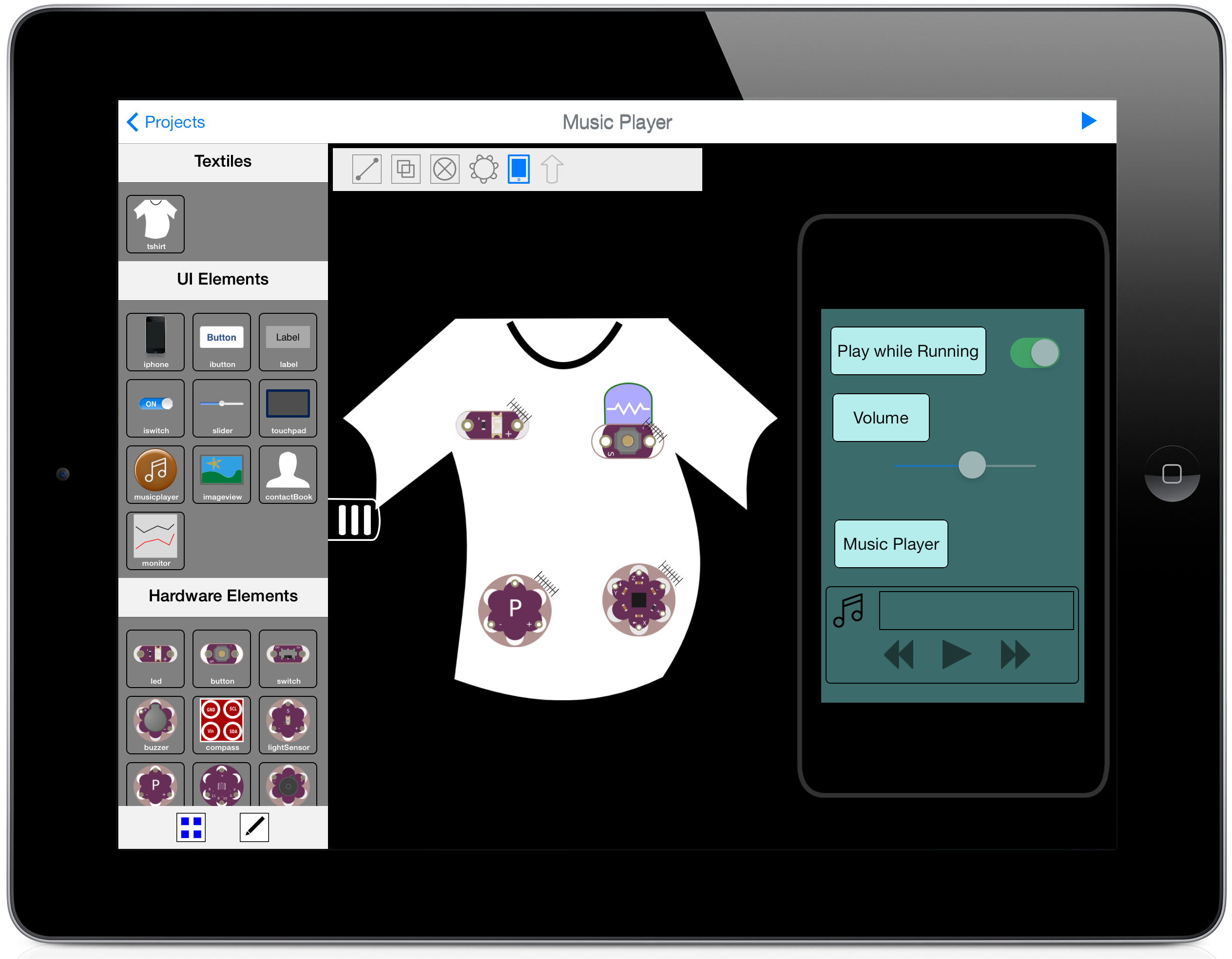


Figure Main Interface Interactex Designer

The Program is then transferred to the Interactex Client over Bluetooth or WiFi, which runs the program.

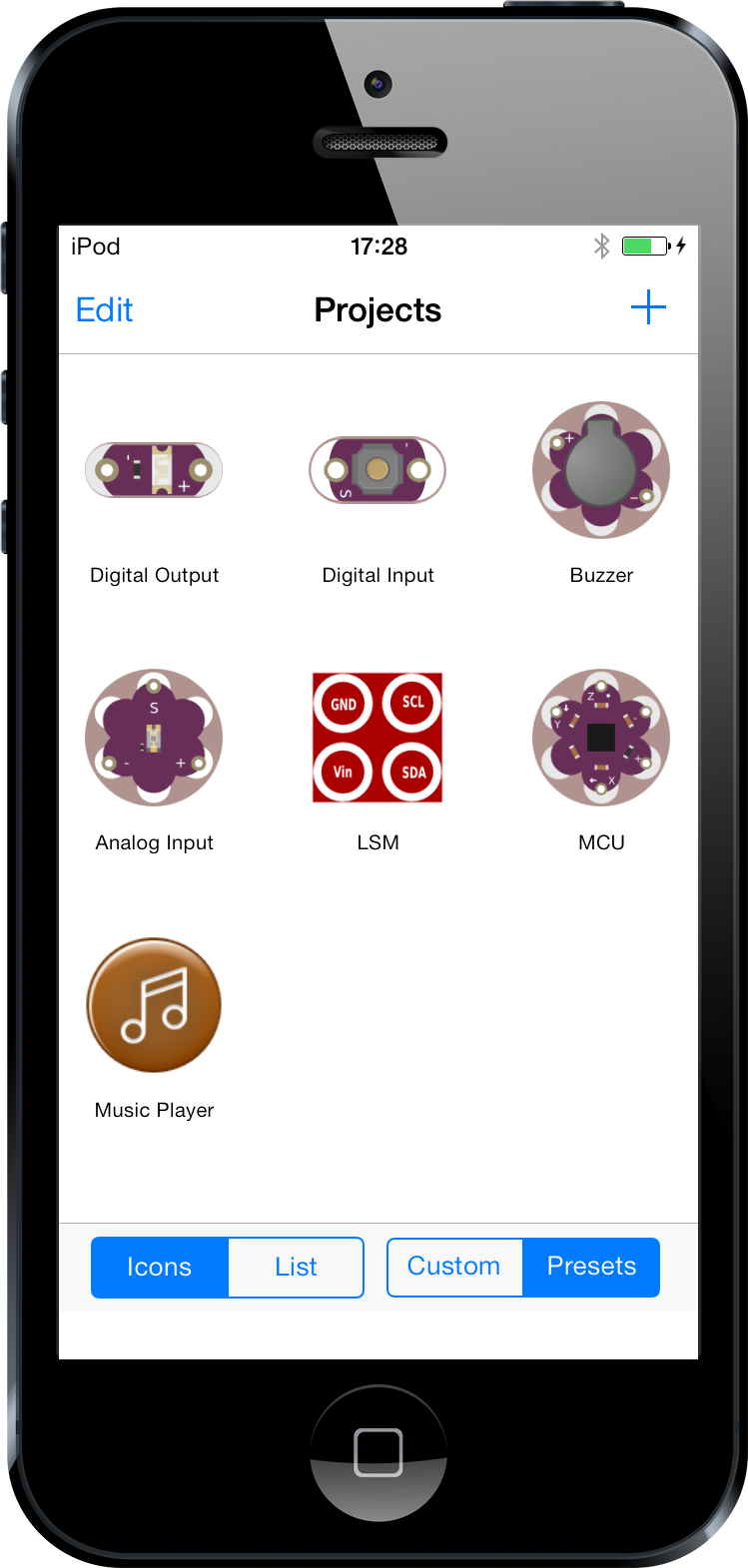


Figure An App running on the Interactex Client

The Interactex Client then communicates with the microcontroller integrated into an eTextile using Bluetooth Low Energy.

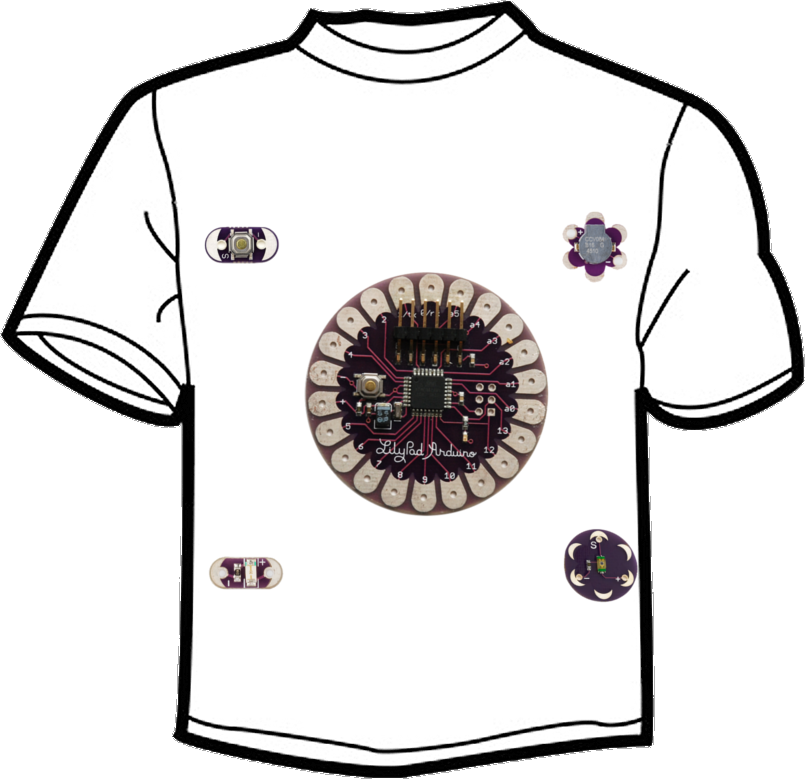


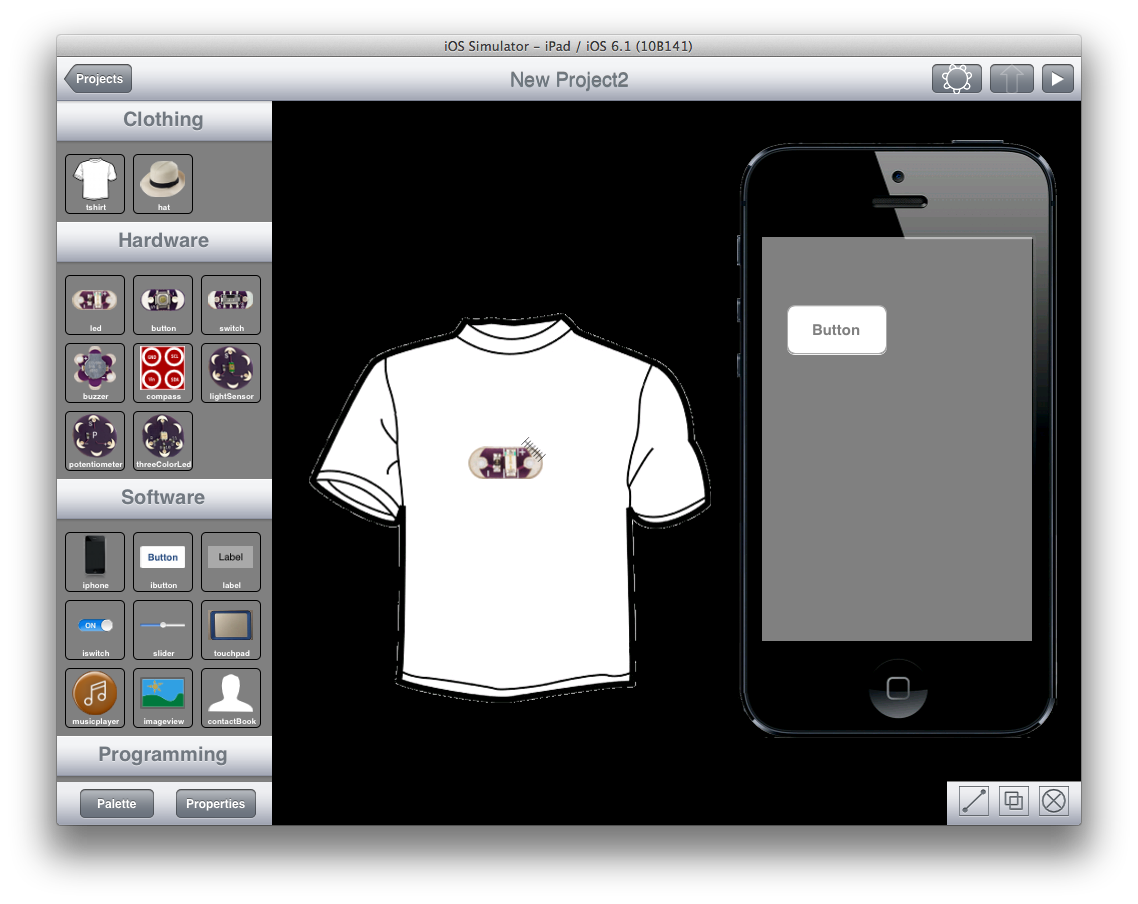
Figure an eTextile with an Arduino Lilypad and some Sensors and actuators attached

# Interactex Designer

Programs in the Interactex Designer are created by Drag and Dropping elements and by drawing connections between them. A Rule Based System built into the Interactex Designer enables users to define behavior: Events of an object are linked to methods of another object. The “Pressed” event of a button can be connected to the “TurnOn” event of an LED. Programs can be simulated on the Designer before uploading them to the Client. More complex programming elements such as conditions and operators make it possible to define more complex behaviors. A Comparator Object can be used together with a Number Value and a Temperature Sensor in order to take an action whenever the temperature reaches a certain value. Such an action can be starting a phone call, playing or stopping music, etc. The Interactex Designer offers two modes, the Edition mode and the Simulation mode. During the Edition mode, users create an application by drag and dropping elements from a Palette into the Project View. During the simulation mode, users can test and debug the created applications. Both modes are described next with an example.

# Tutorial

This example application will turn on a LED using buttons on the iPhone. In the image below, a button has been added to the iPhone object, a T-Shirt and a LED have been added to the project.



In order to make a program that turns on the LED when the button is pressed, both objects need to be connected. In order to connect objects, the connection switch should be activated. The connection switch is found on the bottom-right side of the screen and looks like this:

Macintosh HD:Users:avenix:Workspaces:wearablesm2m:Documentation:Images:6 - Demo2.png

Drawing a line from one object to the other creates a connection between them. When drawing a connection between the button and the LED, a popup will appear. This popup displays events from the source object (the Button) on the left side and matching methods or actions from the target object (the LED) on the right side.



Selecting the *touchDown* event to the *turnOn* method will cause the LED to turn on when the button is pressed.

In order to test this simple application, the Simulation mode can be started by pressing the Play button on the top right side of the screen. While simulating, pressing the button on the iPhone should turn on the LED’s light, as depicted below.

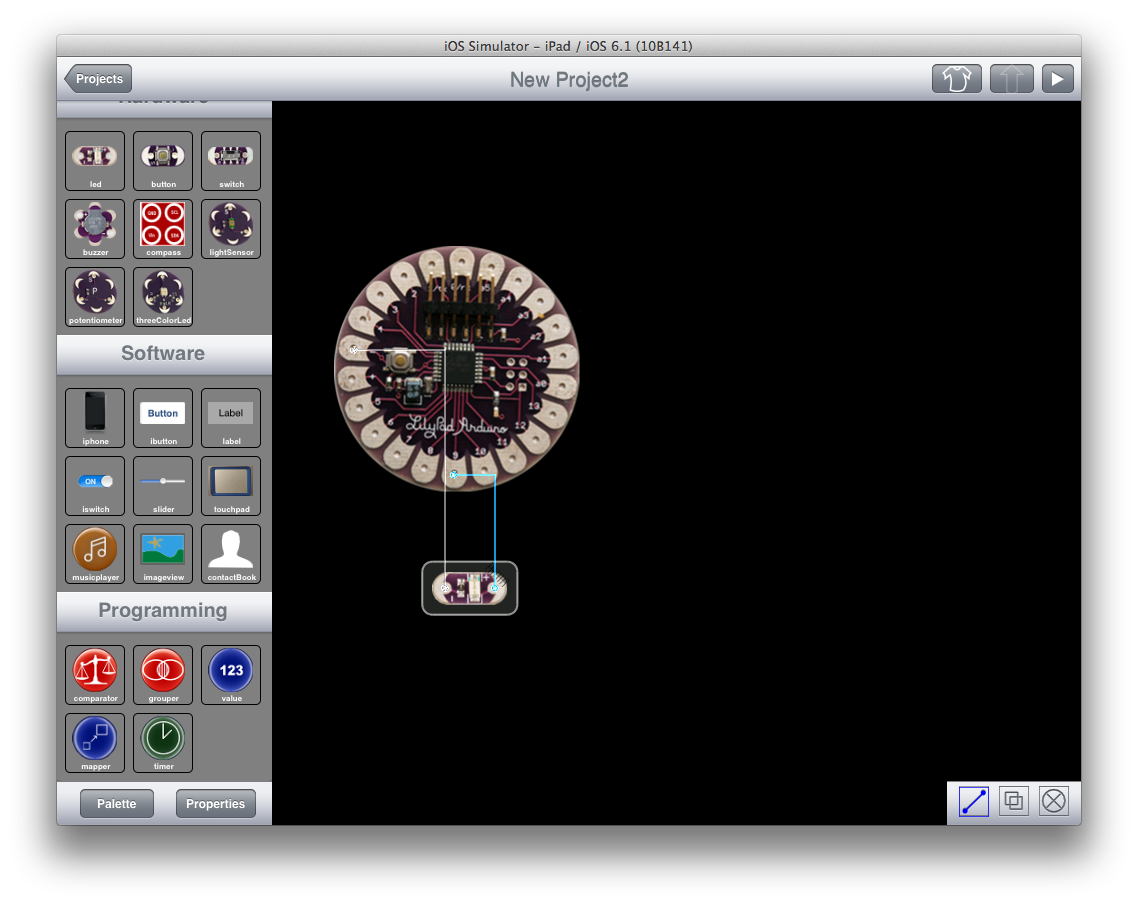


In order to switch back to Edition mode, the cross on the top-right side of the screen has to be pressed.

Before this can work on the hardware side, the application has to know what board’s pins the LED will be connected to. This can be done on the Hardware View. To enter the hardware view, the button that looks like a Lilypad on the top-right side of the screen has to be pressed.

Macintosh HD:Users:avenix:Workspaces:wearablesm2m:Documentation:Images:9 - Demo5.png

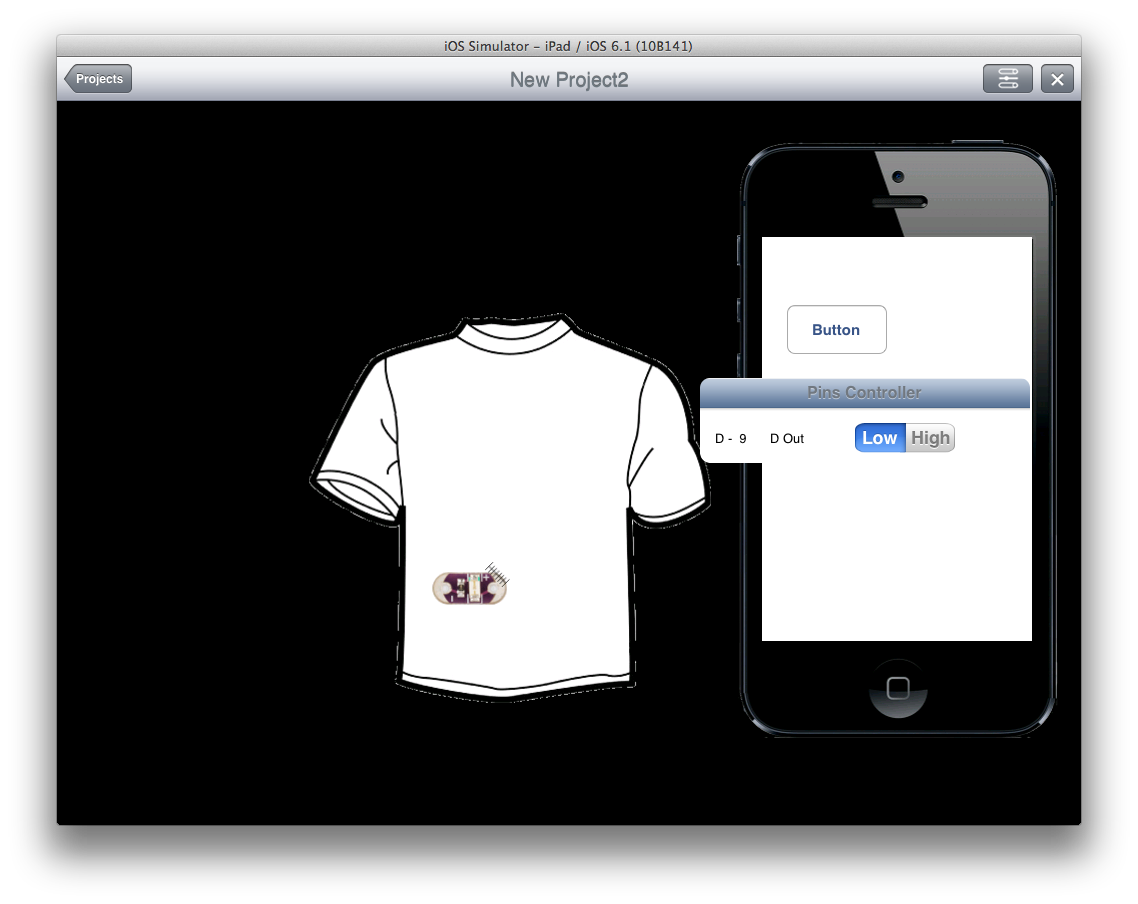
The – pin of the LED is automatically wired. The + pin can be connected to any valid board’s pin by drawing a line with the finger while the connection switch is activated (in the same way as connections between objects are created).



After hardware components are wired, if the application still needs to be debugged, the Pins Controller feature can be used. This feature is available during Simulation Mode and can be opened using the button next to the stop button that looks like this:

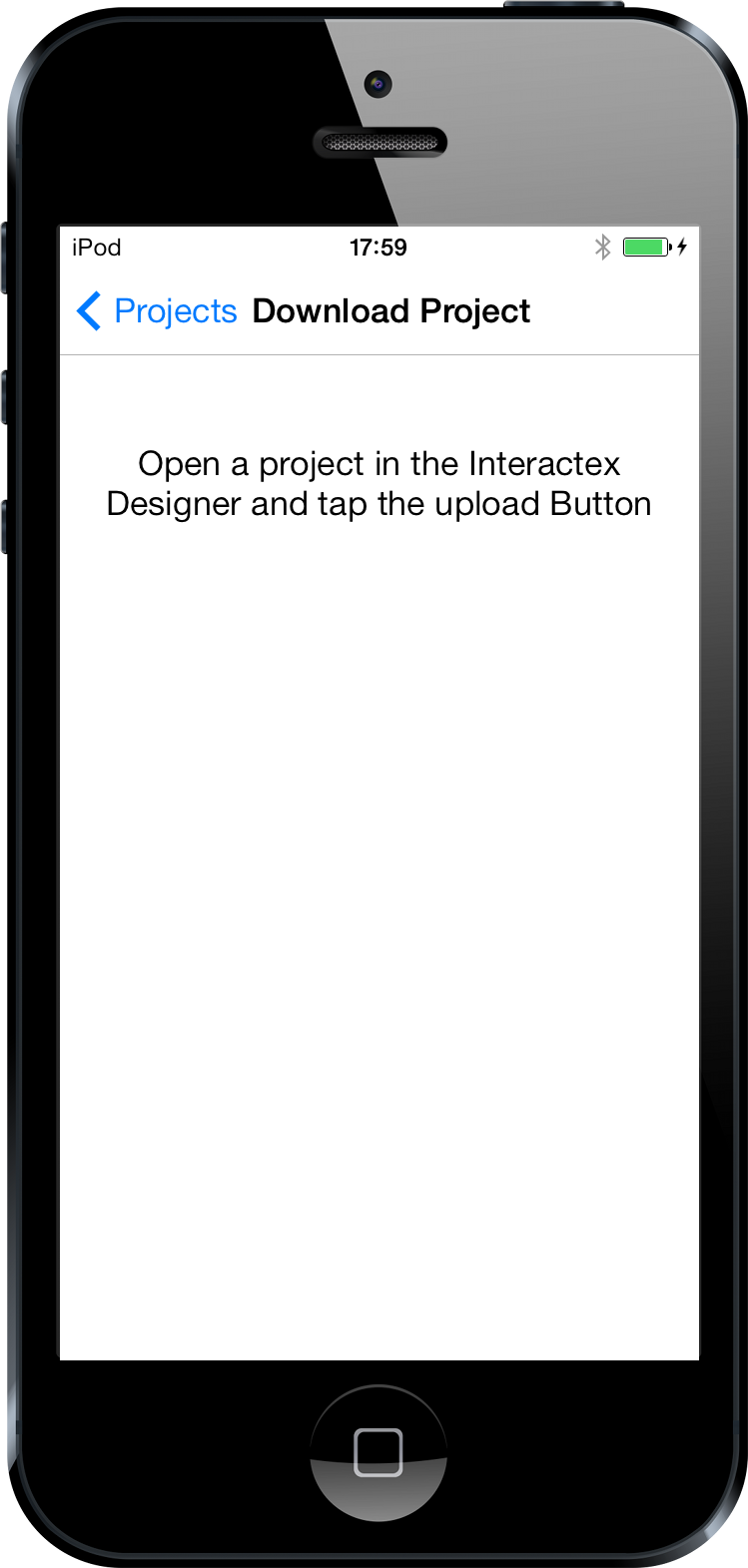
Macintosh HD:Users:avenix:Workspaces:wearablesm2m:Documentation:Images:11 - Demo7.png

The Pins Controller displays pin values and allows users to change them in order to observe hardware’s behavior.



In order to transfer the application to the iPhone (or iPod):

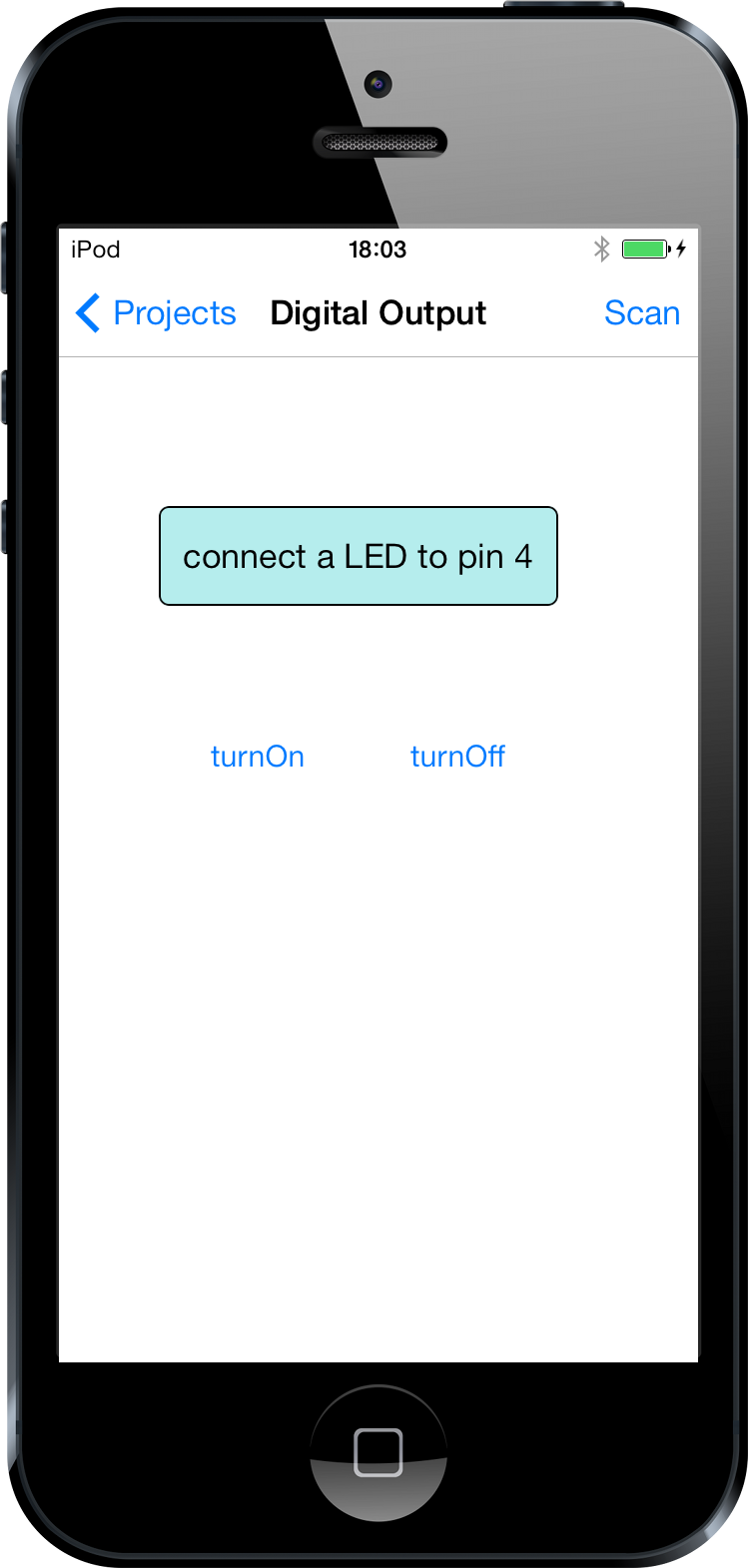
1. Both devices (iPad and iPhone) need to either be connected to the same network, or have Bluetooth activated.
2. In the Interactex Client, choose a + icon to go to the Download mode:



1. The push button (the last icon in the Designer’s toolbar – an arrow pointing up) becomes enabled in the Designer. After pressing it, the application is transferred to the Client Application.

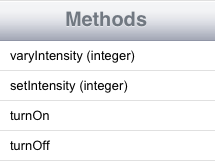
Macintosh HD:Users:avenix:Desktop:Screen Shot 2013-12-13 at 5.37.46 PM.png

1. By tapping the Scan button at the top-right side of the screen, the Client Application will scan for nearby Bluetooth 4.0 devices. Once a device is found which implements one of the supported Services (see section Supported Devices), the text “Start” will replace the previous “Scan” text. By pressing the “Start” Button on the top-right side of the screen a connection with the hardware is established and the application starts running. After that, the “Stop” text will replace the “Start” text. When the “Stop” button is pressed, the device disconnects from the hardware. By going back to the projects screen at any time, the device disconnects from the hardware.

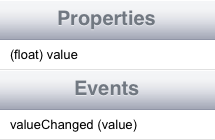


# Event – Method Mechanism

Events are the cause (or triggers) of the methods. Not every event can be connected to every method. The parameters a method expects need to match the values an event delivers. For example, a method such as *setIntensity* of the LED expects an integer number (as can be seen below - parameters appear between brackets).



An event such as the *valueChanged* event of the slider delivers the value property, which is another number (float in this case). Because both parameters are compatible, it is possible to connect *valueChanged* to *setIntensity.* This will cause the intensity of the LED to be modified with a slider.



# Features

|  |  |
| --- | --- |
| UI Elements | Description |
| Button | A button that can be pressed. Generates events when pressed and when released. |
| Label | Displays text and numbers |
| Switch |  |
| Slider | Delivers a value which can be used as input for other objects (ex. Frequency of a buzzer) |
| Touchpad | Generates events when user performs following gestures on it: tap, double tap, pinch, pan, long press. |
| Music Player | Accesses user’s music library and offers methods such as: play, stop, next and previous. |
| Image View | Displays an image. |
| Contact Book | Accesses user’s contact book and offers functionality to iterate through contacts and make calls. |
| Monitor | Displays sensor readings over time. At the moment it offers two methods: setValue1 and setValue2. Value1 will be displayed as a blue line and Value2 red. |

|  |  |
| --- | --- |
| Hardware Elements |  |
| LED |  |
| Button |  |
| Switch |  |
| Buzzer | Represents a Lilypad Buzzer. It can be turned on, turned off , and its frequency can be set. |
| Compass | Accelerometer and Magnetometer. Should be connected to the SCL and SDA pins for I2C communication. |
| Light Sensor | Represents a Lilypad Light Sensor. It offers an event: valueChanged which notifies when the reading of the sensor changed. |
| Temperature Sensor | Represents a Lilypad Temperature Sensor. It works similar to the Light Sensor. It offers an event: valueChanged which notifies when the reading of the sensor changed. |
| Potentiometer | Generates events according to three modes: *always*, *InRange* and *Once*. The *Always* mode will trigger an event whenever the hardware value changed. The *InRange* mode generates an event when the hardware value changed and this value lies within a certain range, which can be configured in the object’s properties. The Once mode will trigger an event once when the value lies within a certain range. |
| Three-Color LED | Not supported yet on the hardware side. |
| Vibe Board | Represents a vibration board. It works similar to the Buzzer, It can be turned on, turned off , and its frequency can be set. |
| Accelerometer | It offers methods for reading x, y and z. Should be connected to three analog input pins. |

|  |  |
| --- | --- |
| Programming Elements |  |
| Comparator | Compares two numbers A and B and generates an event depending on its configuration. If A is bigger than B and the Comparator is in ‘bigger’ mode, then the ‘conditionIsTrue’ even gets triggered. This event can, like any other event, be connected to other object’s methods. In order to set the values A and B that should be compared, connect an event that delivers a number value (such as the intensityChanged event of the Light Sensor) to the setValue1 method of the Comparator. To do this, a line should be drawn starting at the Light Sensor and ending on the Comparator. |
| Grouper | Compares two Boolean values (values that can be either true or false) and generates an event depending on whether both of them are true or only one of them is true. It is connected in a similar way to the Comparator. |
| Number Value | Represents a number (equivalent to a variable in programming). Can be used for example together with the comparator in order to detect when a specific object’s property (such as the buzzer’s frequency or the LED’s intensity) reaches a specific value. Generates an event when its value changes. This is the event that can be connected to the comparator. |
| Boolean Value | Represents a Boolean value (equivalent to a variable in programming). It can be used for example together with a grouper condition. |
| String Value | Represents a constant String (equivalent to a constant variable in programming). It can be used to set the text of a label. In the future, it will be formateable such that users can mix text and numbers, while these numbers could be sensor values. |
| Mapper | Scales and constrains a value. Can be used to make numbers fit within a certain range. For example, the slider produces by default values between 0 and 255 and the buzzer produces frequencies between 0 and 20000. The mapper can be used to make such range conversions. Its current implementation offers a linear function y = ax + b which means that incoming values x get multiplied by *a* and added *b*. Values are clamped to the range [min max]. Generates an event whenever the value changes. |
| Timer | Generates an event after x time. |
| Sound | Represents a sound. It offers a single method to play it. |