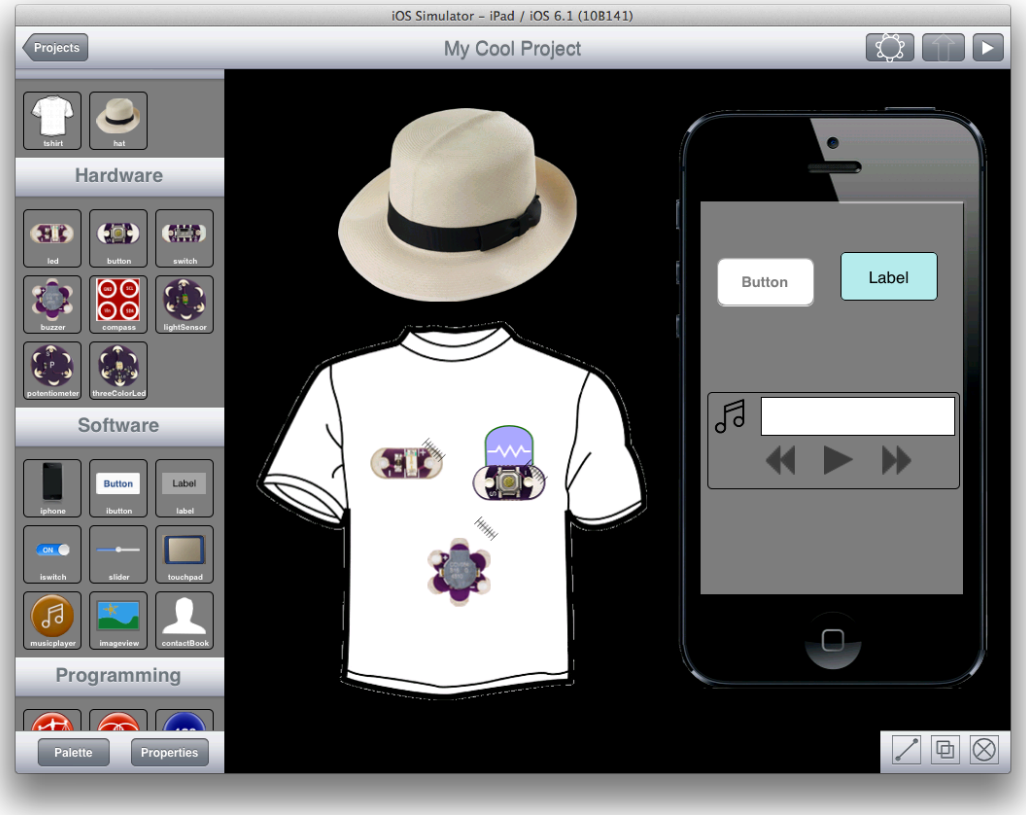
TangoHapps

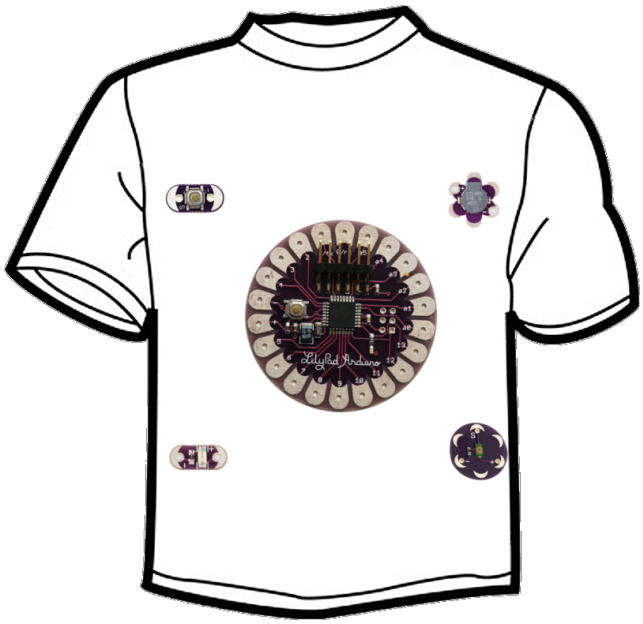
Overview

TangoHapps is a toolset to create iPhone and iPod applications that interact with hardware components such as sensors and actuators.

The toolset consists of three tools or applications: TangoHapps, TangoHappsClient (or Client Application) and the Arduino Program (or Sketch).

TangoHapps is an iPad tool used to create, configure, test and debug applications. TangoHappsClient is an iPhone (or iPod) tool that runs applications created with TangoHapps. The Arduino program runs on the Arduino board and constantly sends and receives hardware values.

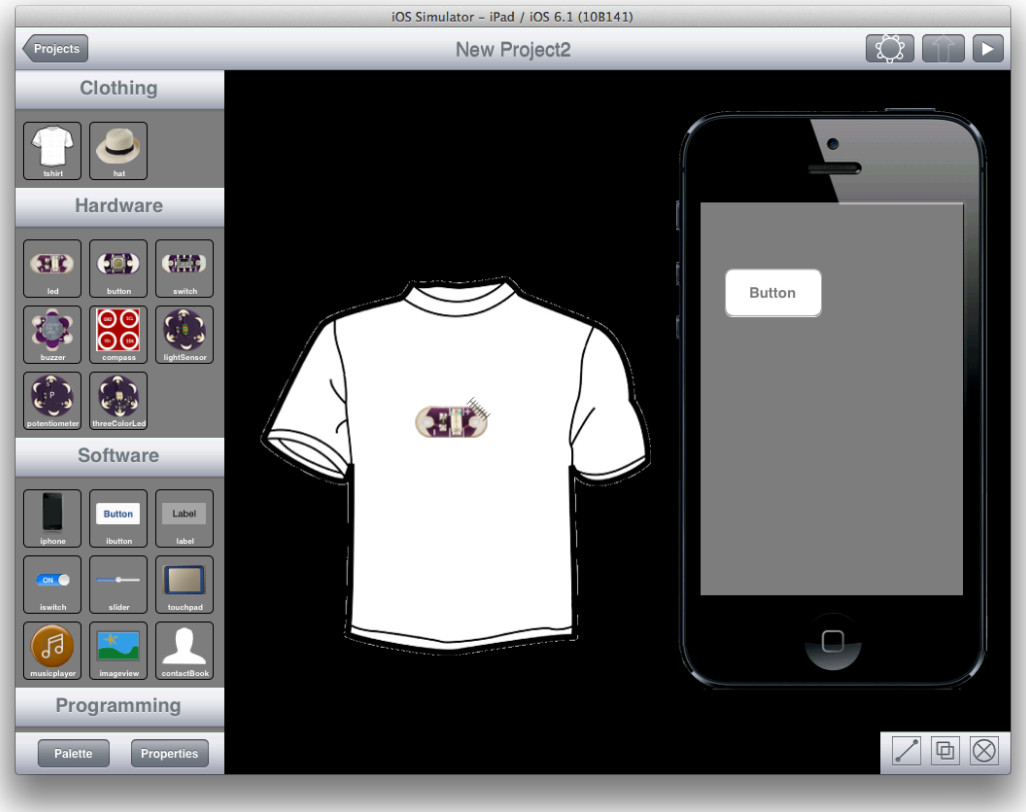




TangoHapps consists of two modes, the Edition mode and the Simulation mode. During the Edition mode, users create an application by drag and dropping elements from a Palette into the Project View. During the simulation mode, users can test and debug the created applications. Both modes are described next with an example.

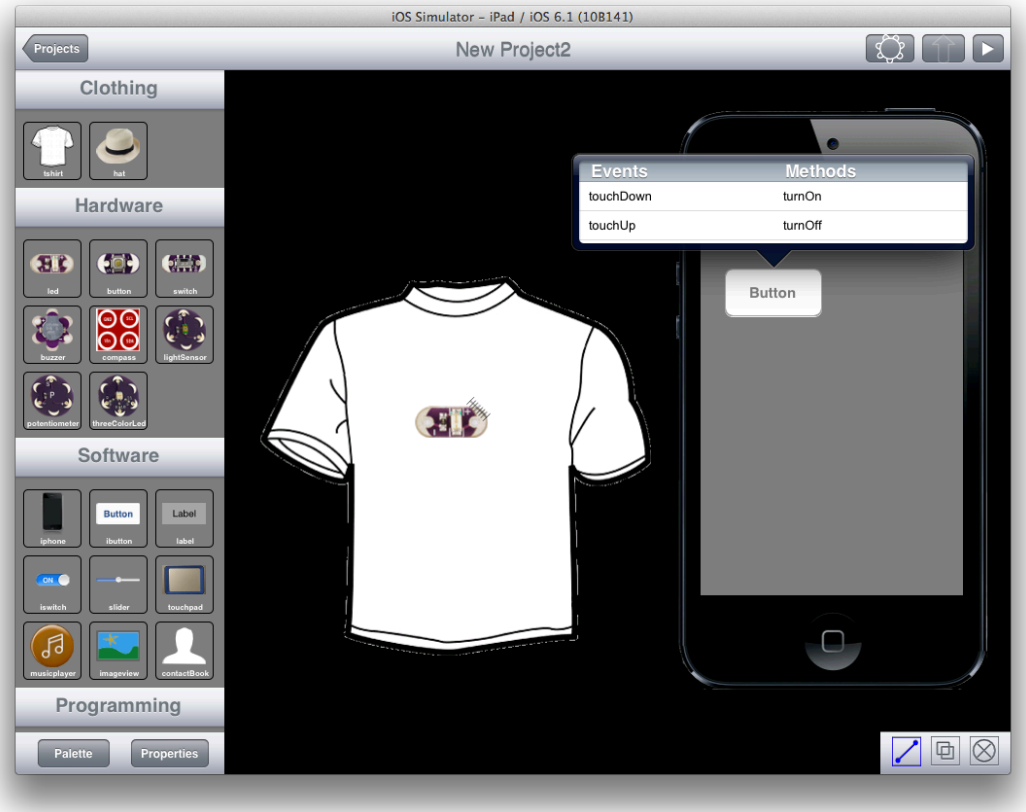
Example

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:

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.



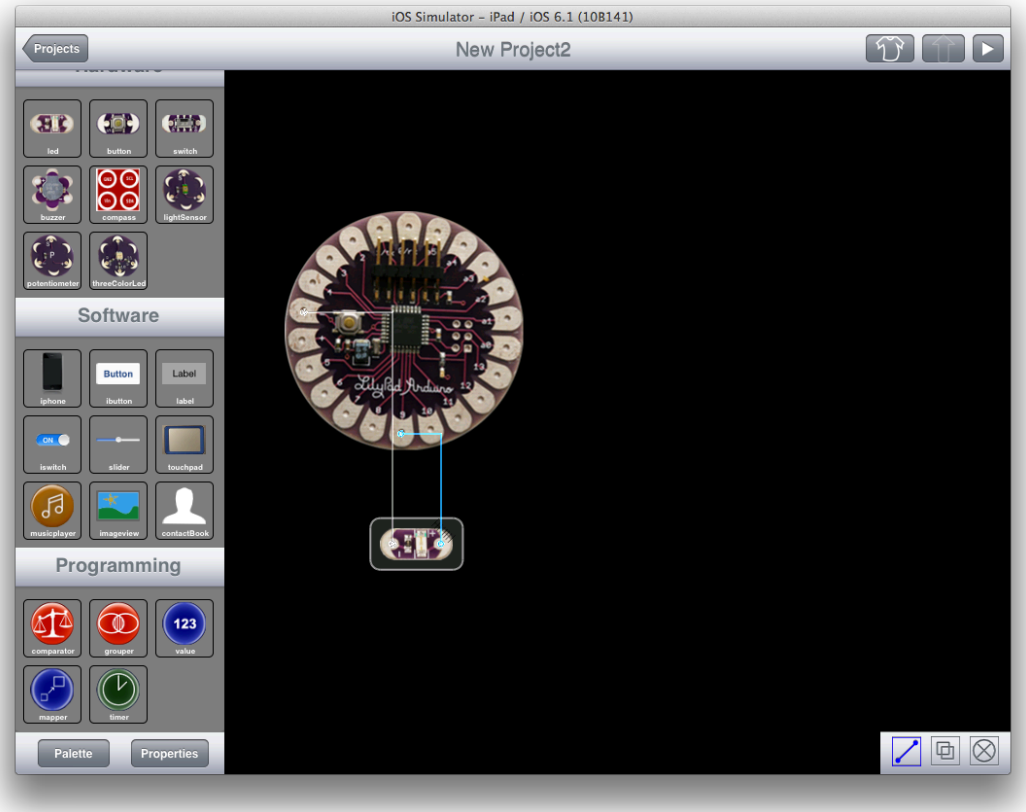
Selecting the *touchDown* event to the *turnOn* method will cause the LED to turn on when the button is pressed.

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.

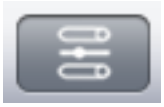
In order to switch back to Edition mode, the cross on the top-right side of the screen has to be pressed.

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.

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:

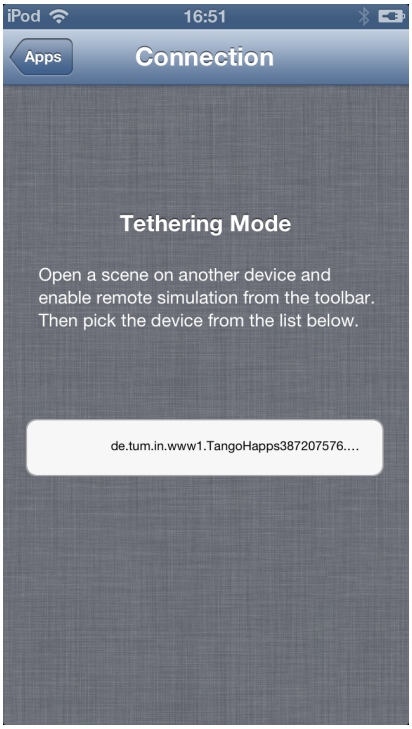


The Pins Controller displays pin values and allows users to change them in order to observe hardware's behavior.

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

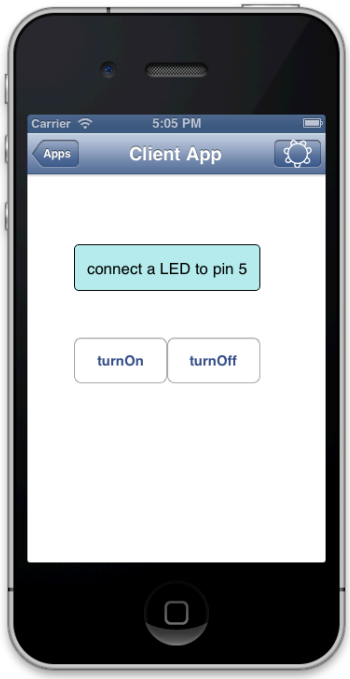
1. Both devices (iPad and iPhone) need to either be connected to the same network, or have Bluetooth activated.
2. The Client Application should be in Remote mode:

3. The Client Application will find and display different iPad devices it can be connected t o. The appropriate one should be selected from the list:



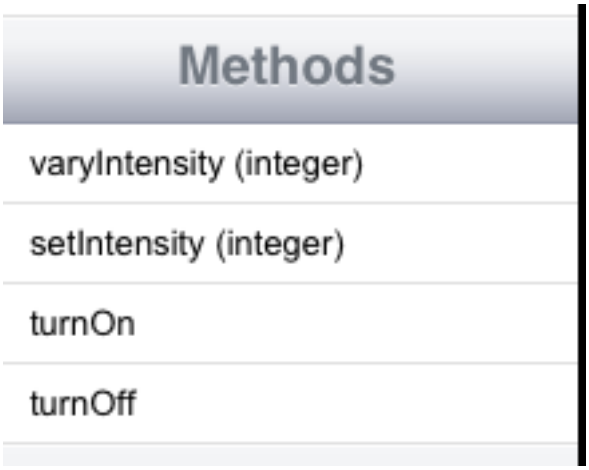
4. The push button becomes enabled in TangoHapps. After pressing it, the application is transferred to the Client Application.

5. The Client Application can be connected to the hardware by pressing the button with a lilypad image on the top-right side of the screen.



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	
Button	
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.

Hardware Elements	
LED	
Button	
Switch	
Buzzer	
Compass	Accelerometer + Magnetometer. Uses I2C.
Light Sensor	
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.

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.
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.
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.