|  |
| --- |
|  |
| UBC eHealth Summer Camp |
| App #3: uFall - Patient Orientation Monitor |
|  |



|  |
| --- |
| © UBC eHealth Summer Camp; Matthew Toom; 2011 |

## Telehealth

One definition of Telehealth is the delivery of health-related services and information via telecommunications technologies. Sometimes it is useful to monitor a patient’s condition remotely over the internet, especially if they have a condition that predisposes them to certain risks.

## Case Study

Consider our sample patient. She is a 67 year old woman who lives on her own in a cabin on Great Beaver Lake near Fort St. James, British Columbia. She suffers from osteoporosis and has a history that has included 2 falls. For an elderly individual with weak bones, a fall can result in permanent injuries and prevent the victim from calling for help. If her body orientation could be automatically tracked, then it would be possible to send an ambulance to help!

**Did you know?**

Falls are the top cause of accidents in people over the age of 65. Falls are also the main cause of serious injuries and accidental deaths in older people.  
  
*Source: American Academy of Family Physicians, 2000*

## body1.jpguFall - Patient Orientation App

The case study is designed to exemplify one possible application for a Telehealth solution. It will serve to motivate our next programming project, which turns a Google Nexus One phone into an eHealth device that tracks a person’s body orientation and could detect if a patient has fallen down for a prolonged period of time. Imagine that the phone is attached to a person’s waist in a phone holster. Then, if they lay down or stand up, the phone can detect their orientation!

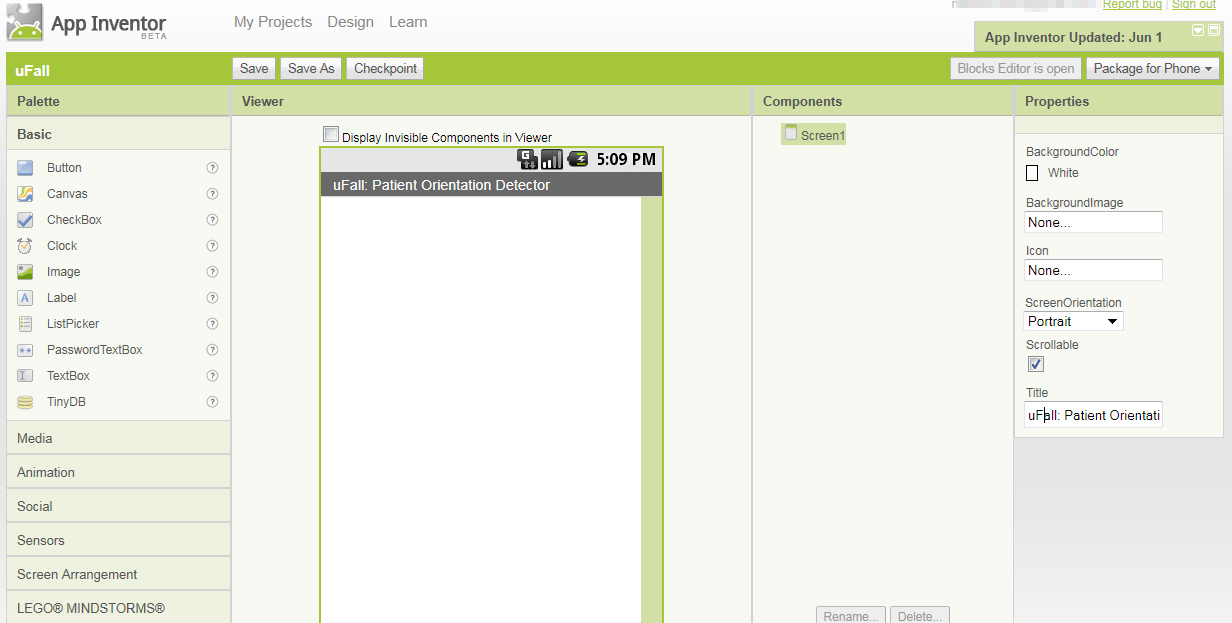
In reality, many similar devices exist that automatically detect falls in elderly people:  
  
  
Check out the video here: <http://www.fallwatch-project.eu/>   
Have a look at this: <http://www.fatronik.com/documentos/cientificas/MEDETEL.pdf>  
Google this: “A Smart Sensor to Detect the Falls of the Elderly”   
  
  


**Login to AppInventor & Create a Project**

Log into AppInventor at <http://appinventor.googlelabs.com/> and create a project called *uFall*.

## Set Title for the Project

Under *Components*, select *Screen1* and then change the title to *“uFall - Patient Orientation Detector”.* Make sure you change the ScreenOrientation to Portrait. This is important in order to prevent the phone from automatically rotating the display!



## Phase 1

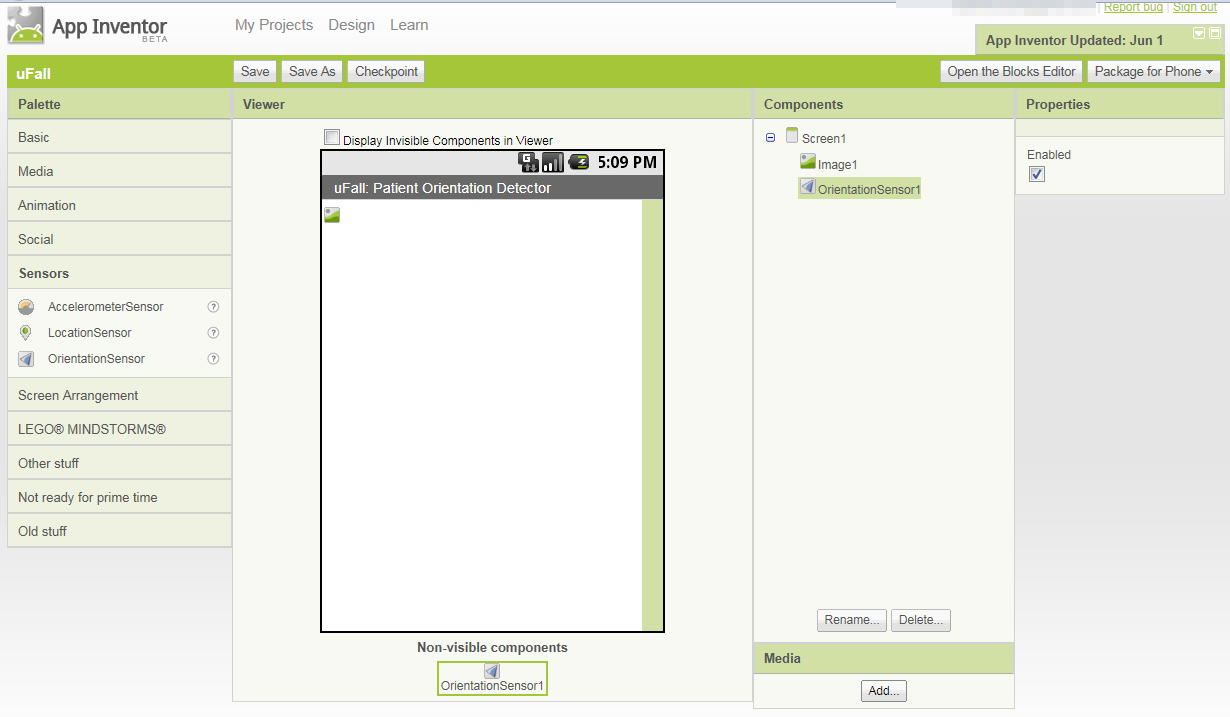
## The first phase of the project is to design an App that will attempt to detect the orientation of the patient who is wearing the phone in a holster.

## Add Components in the Designer View

We will be working with a new object type called an *Image*, which is an image of something! For our uFall app, we will make it an image that shows the status of the person who has the phone holstered to them. First add an *Image* object to the canvas.

We will also add an *OrientationSensor* object to the canvas. The *OrientationSensor* object is found in the Palette under *Sensors* (not under *Basic* like most of the other items we have worked with so far). To add the *OrientationSensor* just drag and drop it onto the canvas. It doesn’t appear on the canvas like other objects because it is “non-visible” to AppInventor.

Once you have added the *Image* and *OrientationSensor*, you should have the following:

  
  
Note the *OrientationSensor* is shown at the bottom!

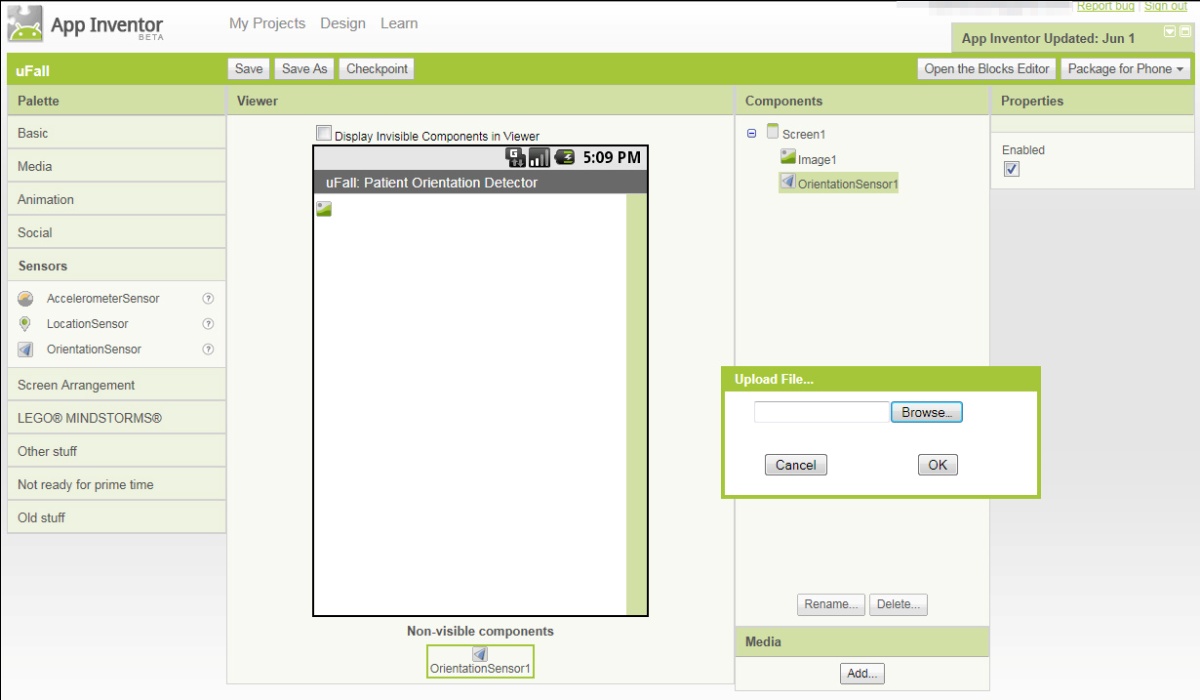
## Uploading Status Images to the App

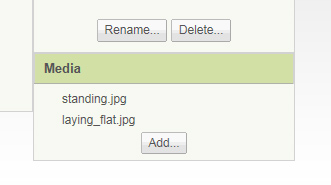
We will need to upload images for the Image object. We have pre-made some for you. First, download them from the web onto your computer’s desktop:

* [Download Standing Image](http://www.toom.ca/ehealth/standing.jpg)
* [Download Laying Flat Image](http://www.toom.ca/ehealth/laying_flat.jpg)

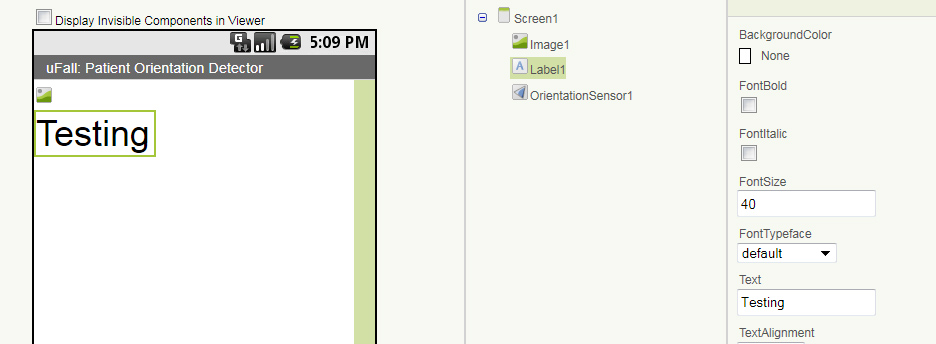
Now you will need to upload these images to Google AppInventor. Do not change the file names, they should be “standing.jpg” and “laying\_flat.jpg” exactly as written and in lowercase. There is nothing special about the images, they are just some JPEGs that we made!

To upload the images, click *“Add…”* under the *Media* tab, as in the example below. Then click *“Browse…”* and select a file. You can only upload one at a time.

  
  
When you are done, you should have the following under *“Media”*:



To do some testing, we will add a *Label* beneath our image. Just drag in a *Label* object from the Palette. We can change the text to “Testing” although it doesn’t really matter. Be sure to set the fontSize to 40. Now, we have the following:

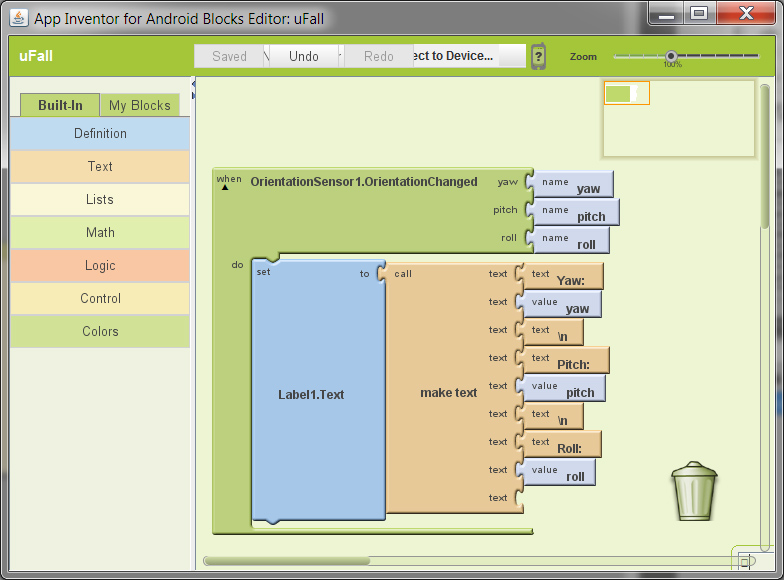


## Testing the OrientationSensor in the Blocks Editor

Great. Now, let’s hook this thing up! We want to detect when the phone is laying flat versus sitting upright.

To do this, we utilize data from the *OrientationSensor*. To play with the *OrientationSensor*, we will report all the values for testing and learning purposes.

Now, open the *Blocks Editor* by hitting the “Open the Blocks Editor” button. Under “My Blocks” drag in an *OrientationSensor1.OrientationChanged* puzzle piece. Under *My Blocks > Label1*, add a *set Label1.Text* puzzle piece onto the canvas. Then, insert a *make text* piece and join together the values. To get the purple-coloured value pieces, look under *My Blocks > My Definitions*. You will notice text blocks with a backslash and then the letter n. This *\n* is a special character in computer programming which represents a new line. When you are done, your blocks should look exactly like this:

  
  
Now, try running the App by clicking *“Connect to Device…”* and then the serial number of your phone.

Can you figure out how the yaw, pitch and roll relate to the phone being flat vs. upright?

**Figuring out if the Phone is Upright or Flat**

We want the image in the Android phone to display the person laying down (which is “laying\_flat.jpg”) when the phone is flat. And, we want it to display the person standing (which is “standing.jpg”) when the phone is upright.

The following is challenging material but it explains how we will detect when the phone is upright:

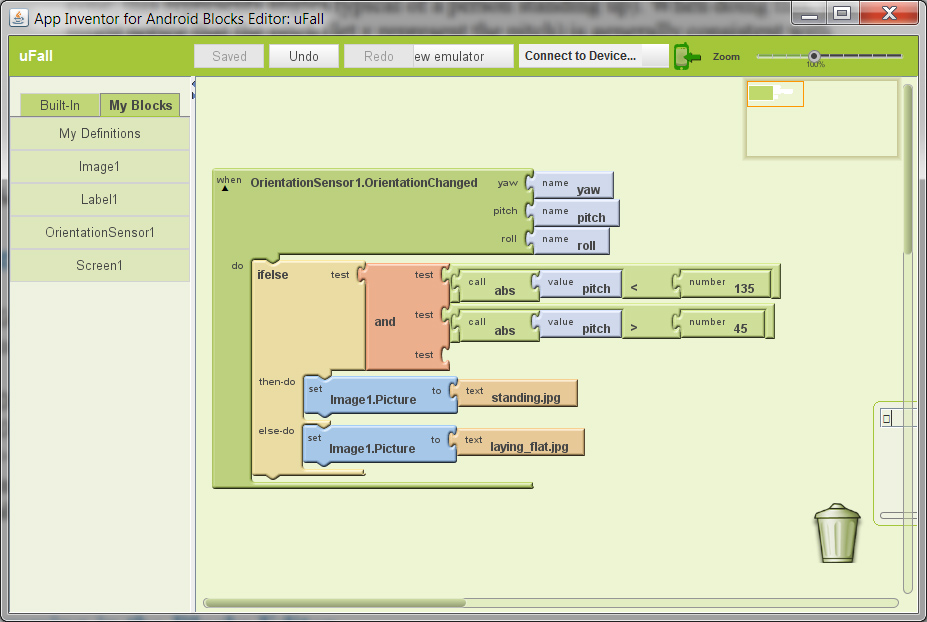
* When the phone is held upright, the pitch is -90. When it is held upside down, the pitch is 90. When the phone is flat and face-up, the pitch is 0. When the phone is flat and face-down, the pitch is -180/180 (it will jump from 180 to -180 as you rotate). Try this for yourself.
* If you hold the phone upright and rotate in a conical motion (like an ice cream cone; this represents angles typical of a person standing up). When doing this, you might notice that the pitch (let *x* represent the pitch) is generally consistent with 45 < |*x|* < 135. The absolute value of the pitch (think of negative numbers as being positives) always seems to be between about 45 and about 135. We can use this condition to check when the phone is upright! If the condition is true, then the phone is closer to being upright. Otherwise, the phone is closer to being flat.

So, now we will update the status image based on the pitch of the phone!

**Coding the App**

Go into your Designer view quickly and delete the *Label* that says Testing (it should be called *Label1*). We don’t need it anymore, since now we know how the pitch works. To delete the *Label*, select it in the *Designer* and hit the *DELETE* key on your keyboard. Or click the *Delete* button which is just above the *Media* section in the *Designer*.

Delete all your blocks in the *Blocks Viewer* and start from scratch. Re-add the *OrientationSensor1.OrientationChanged* and then add an if/else block from the *Built-In > Control* menu. Connect up an *and* puzzle piece from the *Built-In > Logic* menu. Code in the mathematical conditions described above. The *abs* puzzle piece represents the absolute value of a number. As shown in the example below, you can set the value of the *Image* to one of the pre-uploaded media files by using a *Text* object to describe that media. Make sure you get the file name exactly right!



## Test your App

We are done Phase 1. Test your App by clicking “Connect to Device…” and selecting your phone’s serial number. Your phone should know whether or not the phone is flat. If you imagine that the phone is in a holster to your body, then it can detect whether you are standing or not.



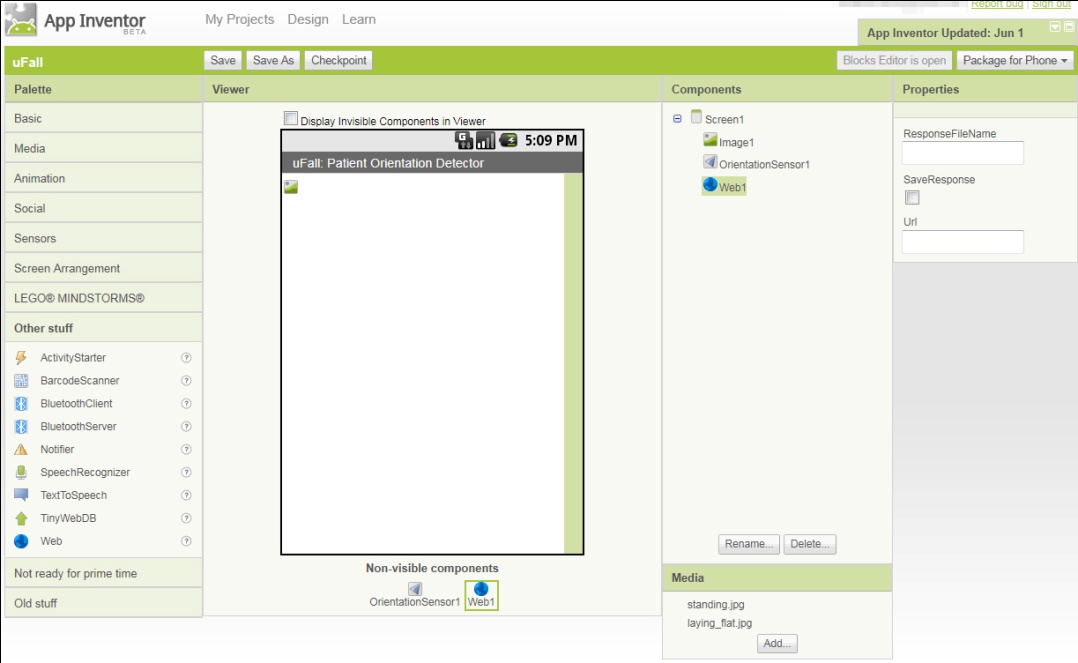
## Phase 2

Now, we will enable your App to transmit your status to a server. If this were a real Telehealth remote monitoring device, we would need to track your status!

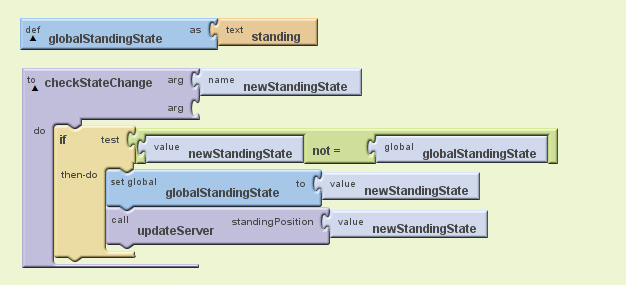
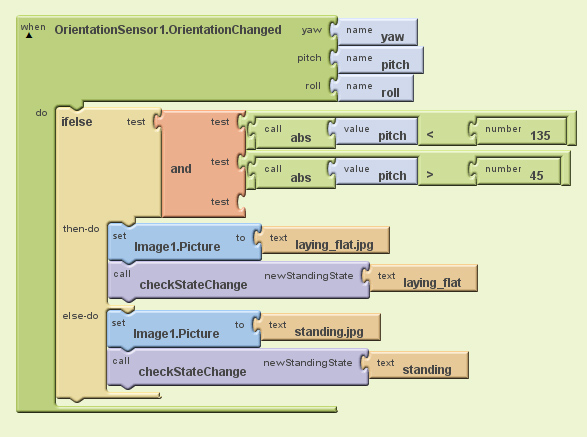
## Add a Web object

Before we add the *Web* object, we need to make sure that the internet is working. Go to the browser on your phone and enter your CWL username & password. Then test for sure that the internet works but browsing to http://ehealth.med.ubc.ca/

Go back to the *Designer* view, and add a *Web* object to your app by dragging it onto the canvas. You’ll find the *Web* object in the Palette under the *“Other Stuff”* heading. When you are done, it should look like the following:



## Determining if the Standing State has Changed (Challenging)

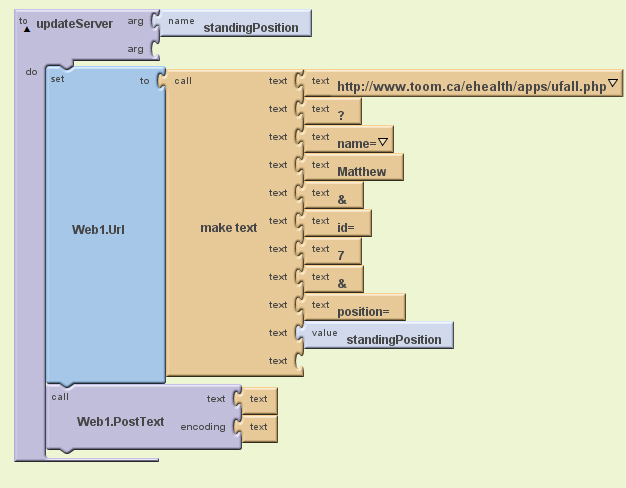
Now, back to the *Blocks Editor*! We want the *Web* object to contact the server via the internet (report changes in state) only at the instant when the state changes from standing to laying or vice versa. However, the *OrientationSensor1.OrientationChanged* is constantly checking many times per second if the orientation state has changed. Remember in the test exercise how quickly the pitch number updates as you move the phone! Many times per second. So, we set up a function called *checkStateChange* that will detect the moment the state has changed. That way, it doesn’t flood the server. It does this using a global variable called *globalStandingState*. If the state has just changed (the current value of *globalStandingState* does not equal the new value), then the updateServer function contacts the server to update the standing state.  
  
  
Now, we must update our previous block (the red arrows point at the updated parts). Every time the *OrientationSensor1.OrientationChanged* happens (anytime the phone’s orientation changes even slightly, which happens many times per second), we check if the state has changed:  
  


## Sending the Updated State to the Server (Challenging)

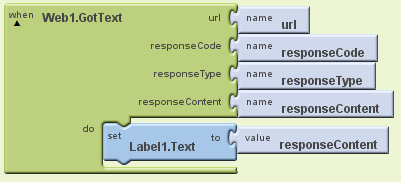
In the figure that shows *checkStateChange*, notice how it runs *updateServer* with the new state. Now we must build that function. That function passes the standing/laying state, when it changes, over the internet to the server. The server is public and allows anybody to view the status (for the purposes of demonstration).

You can check the status of the server in your web browser simply by typing in the following address: <http://www.toom.ca/ehealth/apps/ufall_view.php>

It is very important that you change the Name below and the Id number below (see the red arrows) before trying your App. You should change “Matthew” to your name. Everyone has their own Id number. Put your Id number in. DO NOT use someone else’s Id number. If you don’t know your Id number, then ask us!

  
  
Now you are done! Load up your web browser and check out the server. Then start your app and you can track your status over the internet inside the web browser. You can also see what other camp students are doing!

## Determining if the Standing State has Changed (Optional)

If you go back to the Designer view and you add a Label, then you can use the Label to track the response of the server, as shown below. If you don’t do this step, then the phone will receive the server response but not display it (which is totally okay).  
  


## Final Solution for the Blocks Editor

Below shows a completed solution for the Blocks Editor to build the uFall application.

