App Inventor + IoT: Building a Healthy Plant Monitoring App

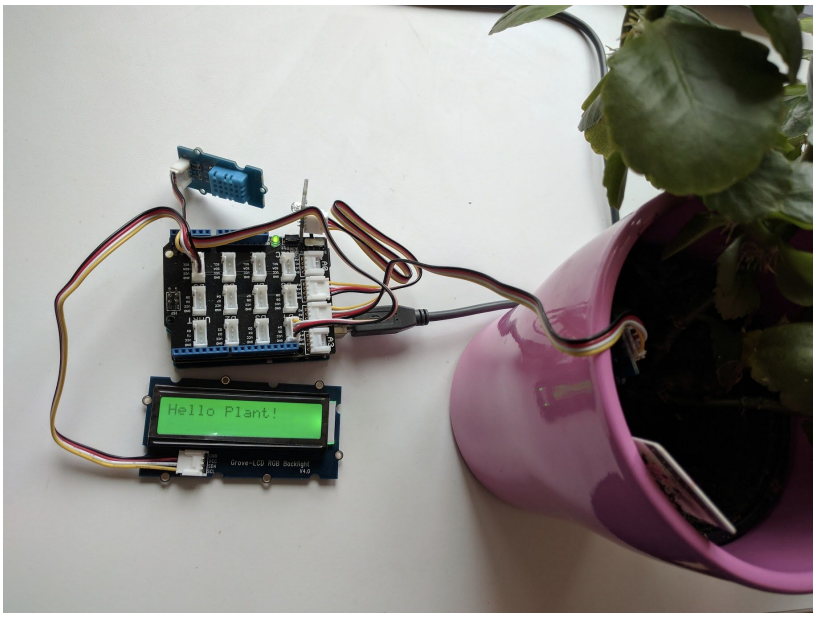
This tutorial will help you get started with building an App that connects and respond to the physical world - often termed the Internet of Things or "IoT".

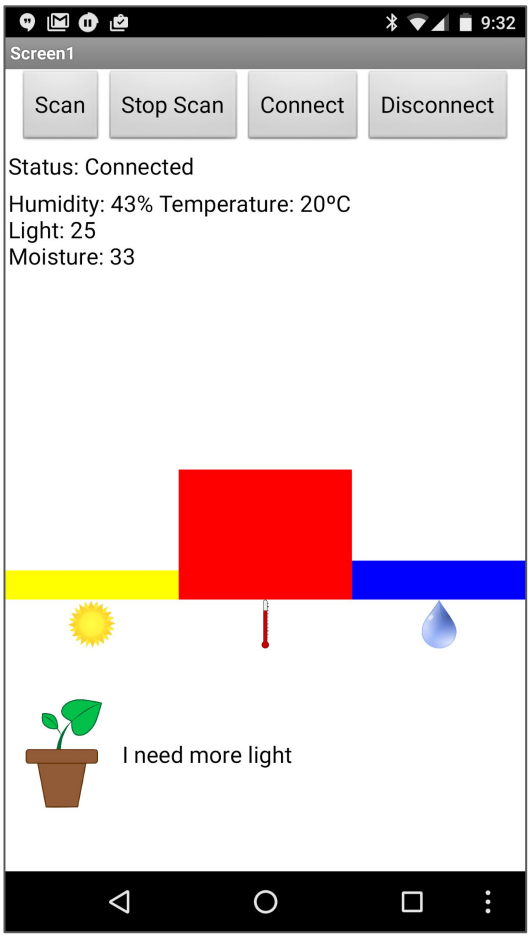
In this project we're going to learn how to build an app that connects to a microcontroller called Arduino 101 via Bluetooth. You can use this equipment to monitor various conditions (i.e., light, humidity, temperature, moisture) that can help you track the overall health of a plant. We will also learn how to graph this data.

We are also using a Seeed Grove shield for this tutorial. You do not need to use this board, but it does make things easier.

This Build-It tutorial assumes you have a basic understanding of App Inventor. If this is your first app, you should first complete at least the introductory app Hello Purr. It is also recommended you do a slightly more advanced app like Paint Pot.

You will first need to set up your Arduino 101 to work with App Inventor by following the steps found here.

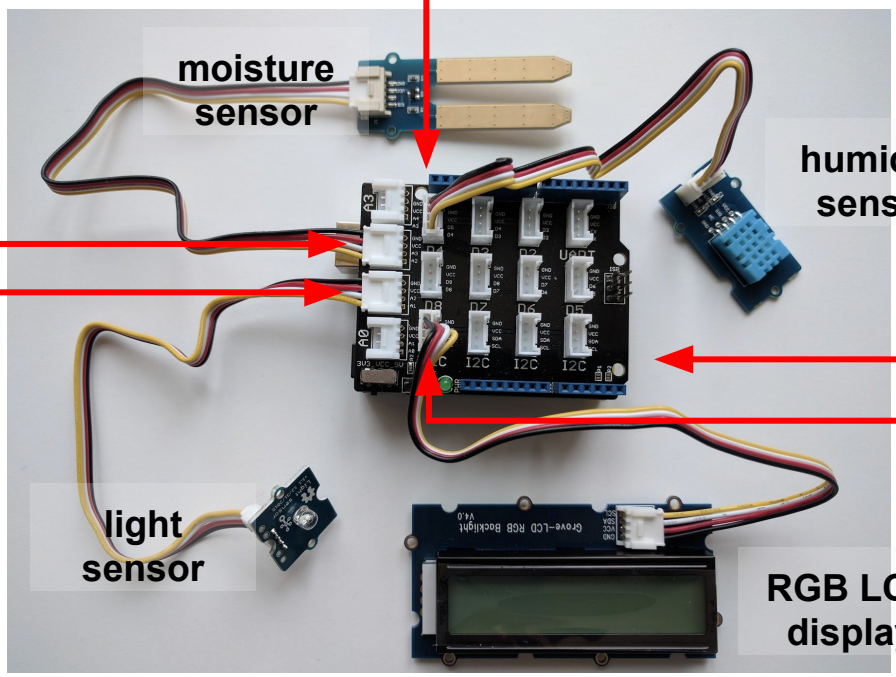




Setting up the Arduino

Let's start by connecting all the sensors we're going to use to our Arduino. For this project, we are also using a Seeed Grove shield attached to the top of our Arduino. While the Grove board isn't necessary, it makes things much easier.
We will also need the following components for this tutorial:

- A moisture sensor
- A light sensor
- A humidity sensor (also works as a temperature sensor)
- An RGB LCD Display

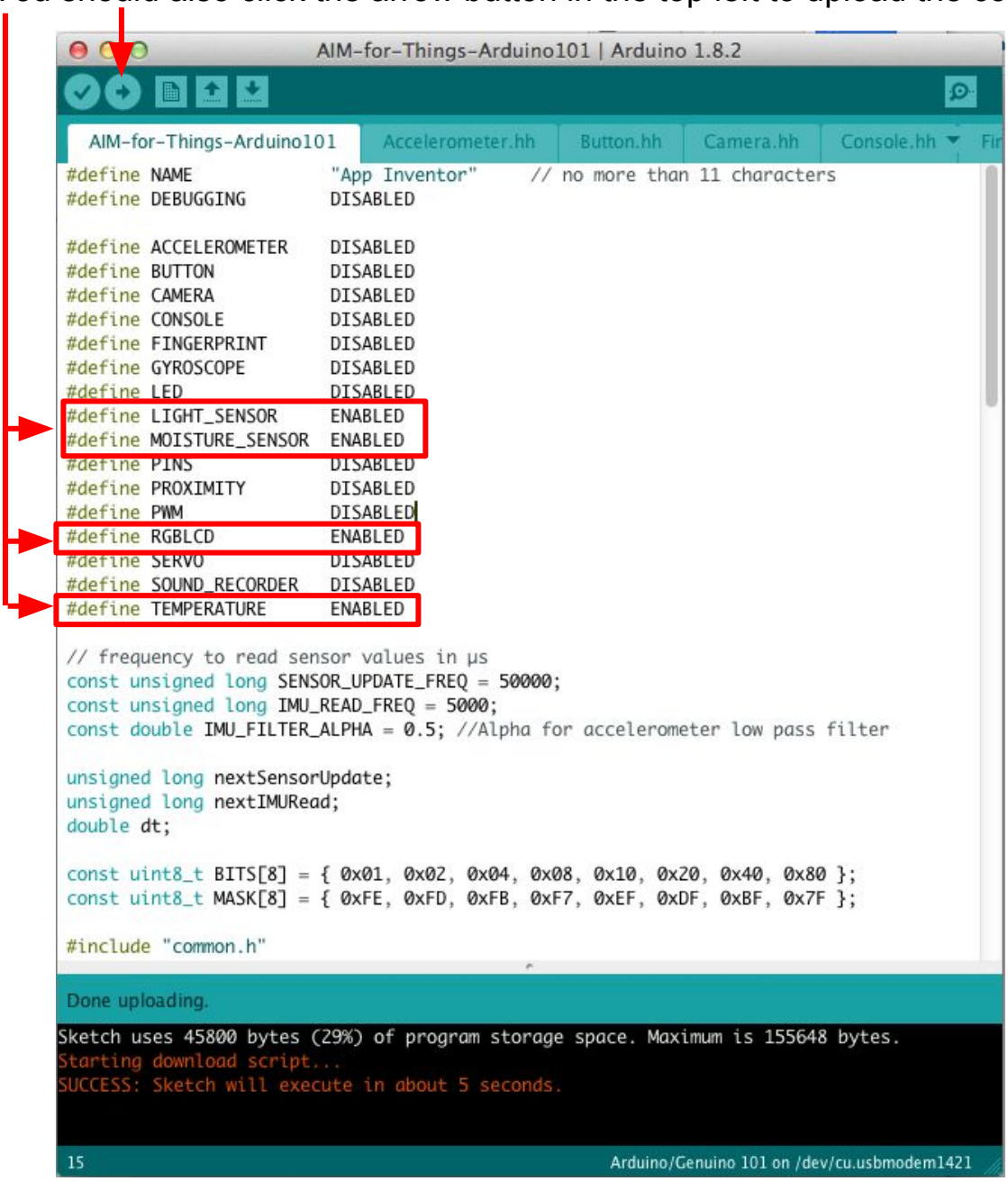


We're going to attach 3 sensors (Light, Humidity, and Moisture) and an RGB LCD Display.

- Attach the **Light Sensor** to the **A1** slot on the Grove board
- Attach the **Moisture Sensor** to the **A2** slot on the Grove board.
- Attach the **Humidity Sensor** to the **D4** slot on the Grove board
- Attach the **RGB LCD Display** to *any* of the **I2C** slots.

First, we need to make sure we have the correct Arduino code running. Plug in your Arduino and open the AIM-for-Things-Arduino101.ino file (from the Setup tutorial above).

- For this tutorial make sure **LIGHT_SENSOR**, **MOISTURE_SENSOR**, **RGBLCD**, and **TEMPERATURE** are set to **ENABLED** and all others are set to **DISABLED**
- You should also click the arrow button in the top left to upload the code



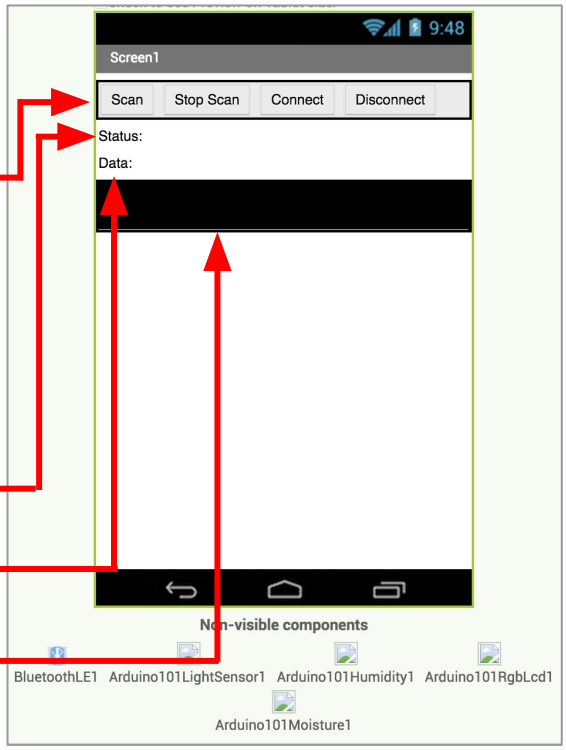
Note: You can set your NAME variable in the code above to something that is easily identifiable, instead of “App Inventor”. Especially in a classroom where there might be many bluetooth devices, naming each device will help when connecting through the app.

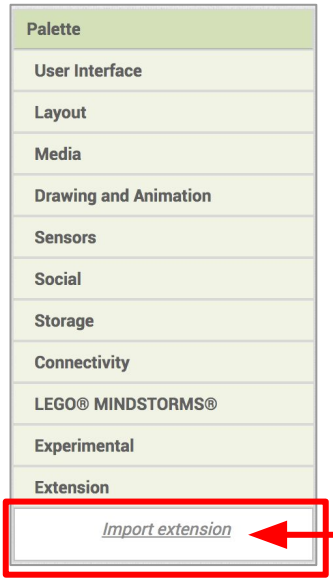
Building the App in App Inventor

Start a new project in App Inventor and name it HealthyPlantMonitor.

First, we need to set up some buttons to find and connect to our Arduino over Bluetooth.

- Drag a *HorizontalArrangement* from the Layout drawer in the Palette and add 4 Buttons to it.
- Rename the buttons: **ButtonScan**, **ButtonStopScan**, **ButtonConnect**, and **ButtonDisconnect**
- Change their text to "Scan", "Stop Scan", "Connect", and Disconnect"
- Below the *Horizontal Arrangement* add a Label. Rename it **LabelStatus** and change its text to "Status: "
- Below that add another Label. Rename it **LabelData** and change its text to "Data: "
- Below **LabelData**, add a ListView. Rename it **ListBLE**.

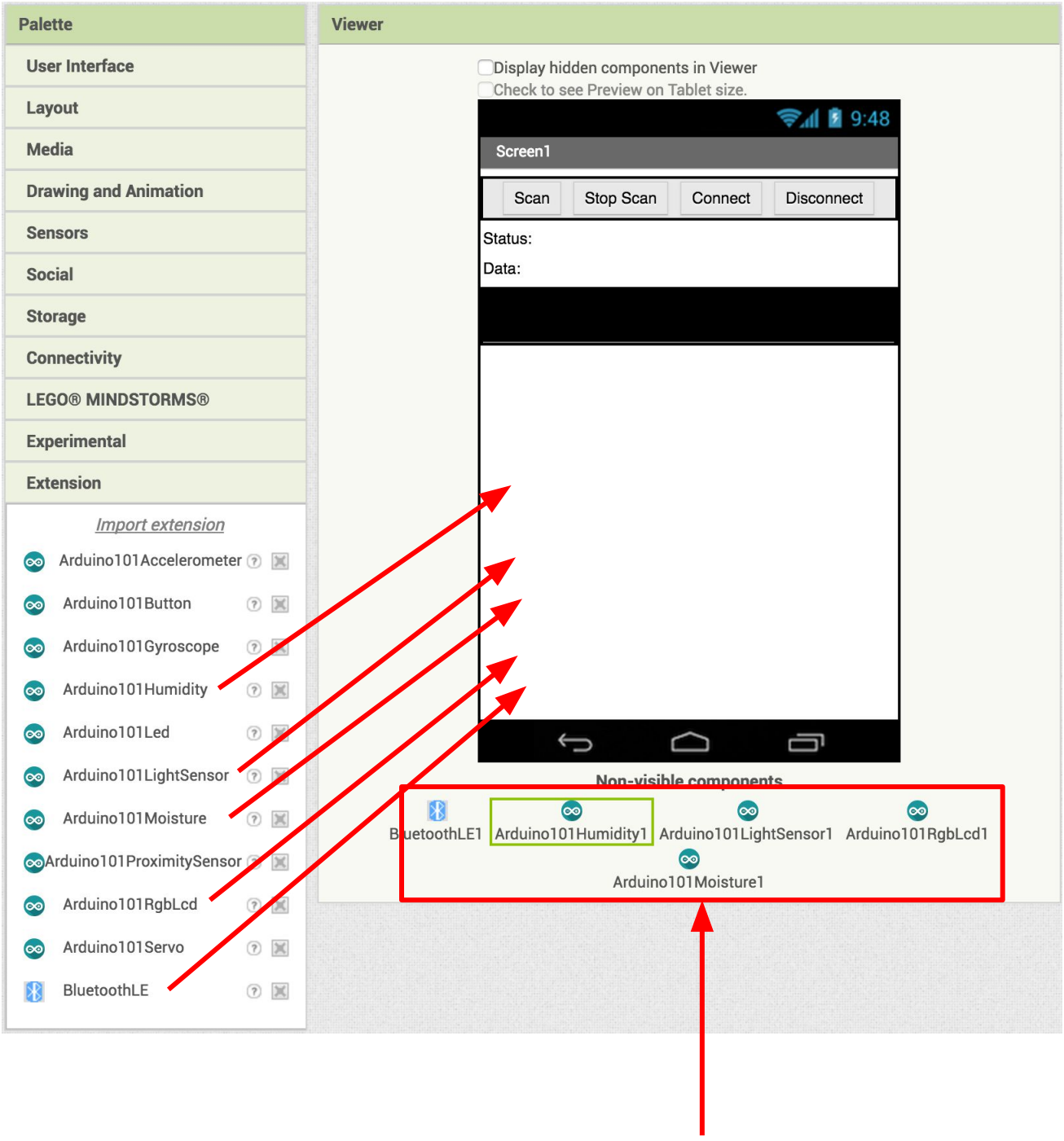




Next, we need to install the various extensions we need for our app.

- Download edu.mit.appinventor.iot.arduino101.aix and edu.mit.appinventor.ble.aix to your computer.
- For both files, in the Palette, click on Extension at the bottom and then on "Import extension" and then "Choose File".
- Find the extensions on your computer and upload them.

From the extensions list, add the following extensions to your app by dragging them onto the Viewer: BluetoothLE, Arduino101LightSensor, Arduino101Moisture, Arduino101Humidity, and Arduino101RgbLcd

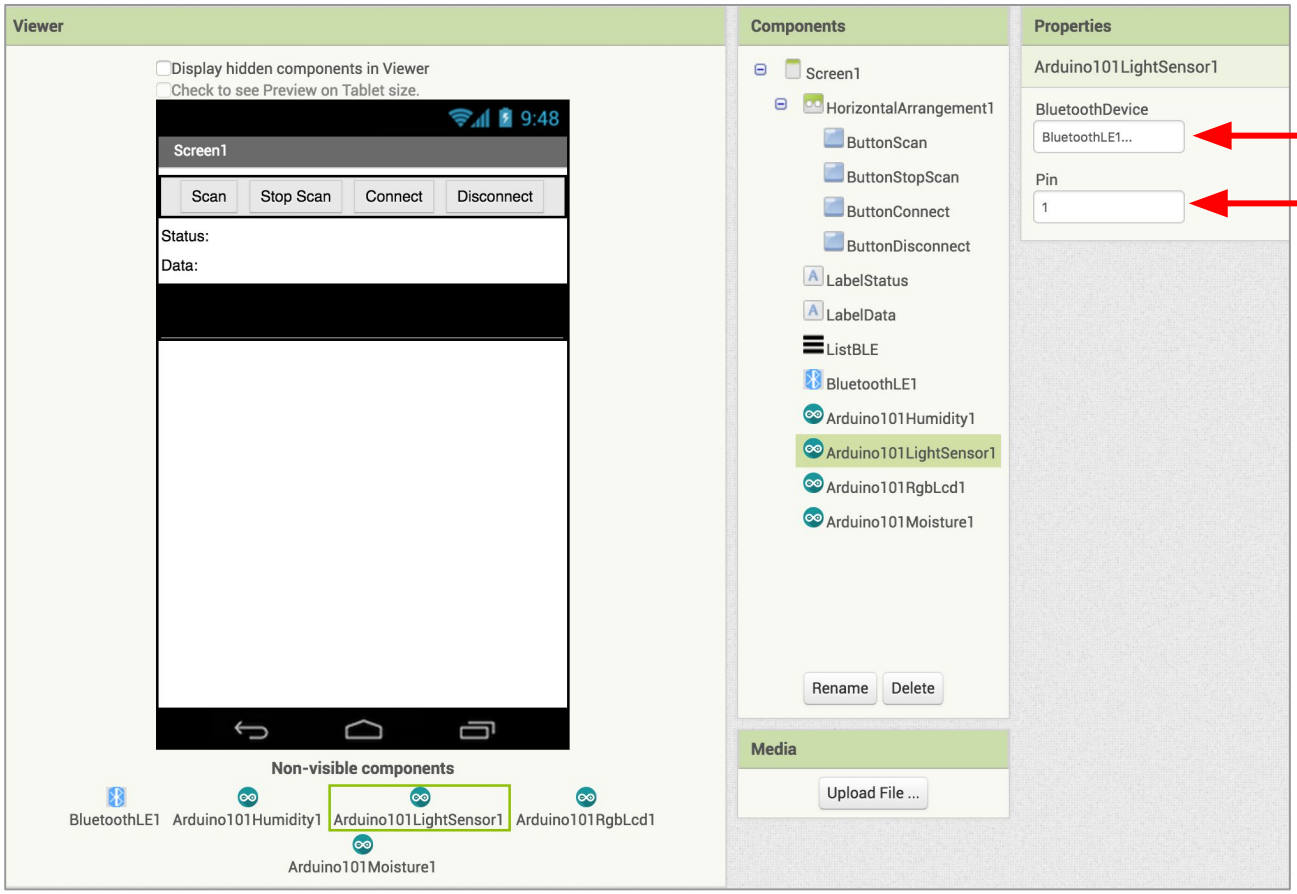


After they are dragged onto the Viewer they will appear below the main screen. Don't worry if you see an error about integers, we'll fix that in a minute.

Next, we need to let App Inventor know which pins on the Grove board the different sensors and the LCD screen are connected to.

First, let's set the pin for the Light Sensor

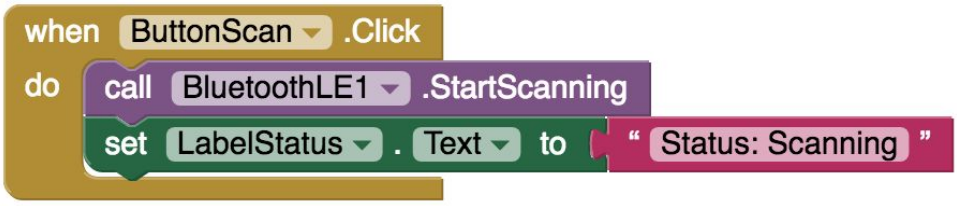
- Click on **Arduino101LightSensor1** in the Components pane.
- In the Properties pane, click on BluetoothDevice and select **BluetoothLE1**. Under **Pin**, enter *only the number* that corresponds to the analog pin the light sensor is plugged into on the Grove board (in this case A1).
    - *Note: You only need to set the number (1) not the letter (A)*
- Now, let's do the same thing for the rest of the sensors
    - Click on **Arduinio101Moisture1**, set its BluetoothDevice to **BluetoothLE1** and set its Pin to 2
    - Click on **Arduino101Humidity1**, set its BluetoothDevice to **BluetoothLE1** and set its pin to 4
- For the **Ardunio101RgbLcd1** you only have to select the Bluetooth device (**BlutetoothLE1**); App Inventor will take care of the rest.



Switch to the Blocks Editor view

We want to set up the app to scan for available Bluetooth devices. To do this, we will use the ButtonScan button to set the Bluetooth component to start scanning, and change the status label.

- From ButtonScan in the Blocks pane drag out **when ButtonScan.Click**
  - from BluetoothLE1 add **call BluetoothLE1.StartScanning**
  - from LabelStatus add **set LabelStatus.Text to**
    - From the Text Drawer add a text block and type in **"Status: Scanning"**



Next, we'll have the app stop scanning and change the status label when we press the ButtonStopScan

- From ButtonStopScan in the Blocks pane drag out **when ButtonStopScan.Click**
  - from BluetoothLE1 add **call BluetoothLE1.StopScanning**
  - from LabelStatus add **set LabelStatus.Text to**
    - From the Text Drawer add a text block and type in **"Status: Stopped Scanning"**

We need to populate the device list with all the available Bluetooth devices

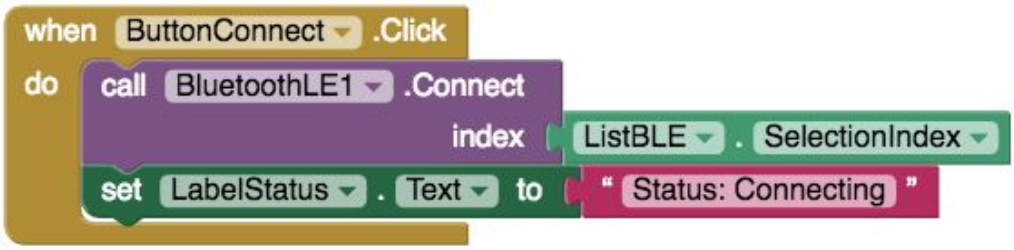
- From the BlueToothLE1 in the Blocks pane drag out **when BluetoothLE1.DeviceFound**
  - from ListBLE add **set ListBLE.ElementsFromString to**
    - From BluetoothLE1 drag out and snap in **BluetoothLE1.Devicelist**

Now we need to have our App connect to the Arduino over Bluetooth, along with all of our sensor extensions.

- From ButtonConnect in the Blocks pane drag out **when ButtonConnect.Click**
  - From the BluetoothLE1 Drawer add **call BluetoothLE1.Connect index**
    - From BluetoothLE1 drag out and connect **ListBLE.SelectionIndex** (This sets the Bluetooth device to the one picked from the list.)

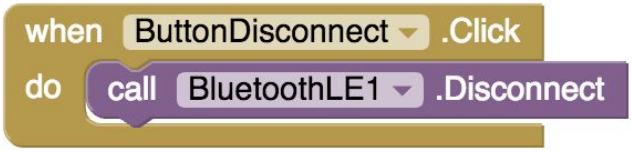
We also want to let the user know we are trying to connect to the device.

- From LabelStatus drag out **set LabelStatus Text to**
  - From the Text Drawer add a text block and type in **"Status: Connecting"**



Next we want to be able to disconnect from the Bluetooth device.

- From the ButtonDisconnect in the Blocks pane drag out **when ButtonDisconnect.Click**
  - From the BluetoothLE1 Drawer add **call BluetoothLE1.Disconnect**



We also want to know when the Bluetooth device successfully disconnects (to know pressing the button above worked)

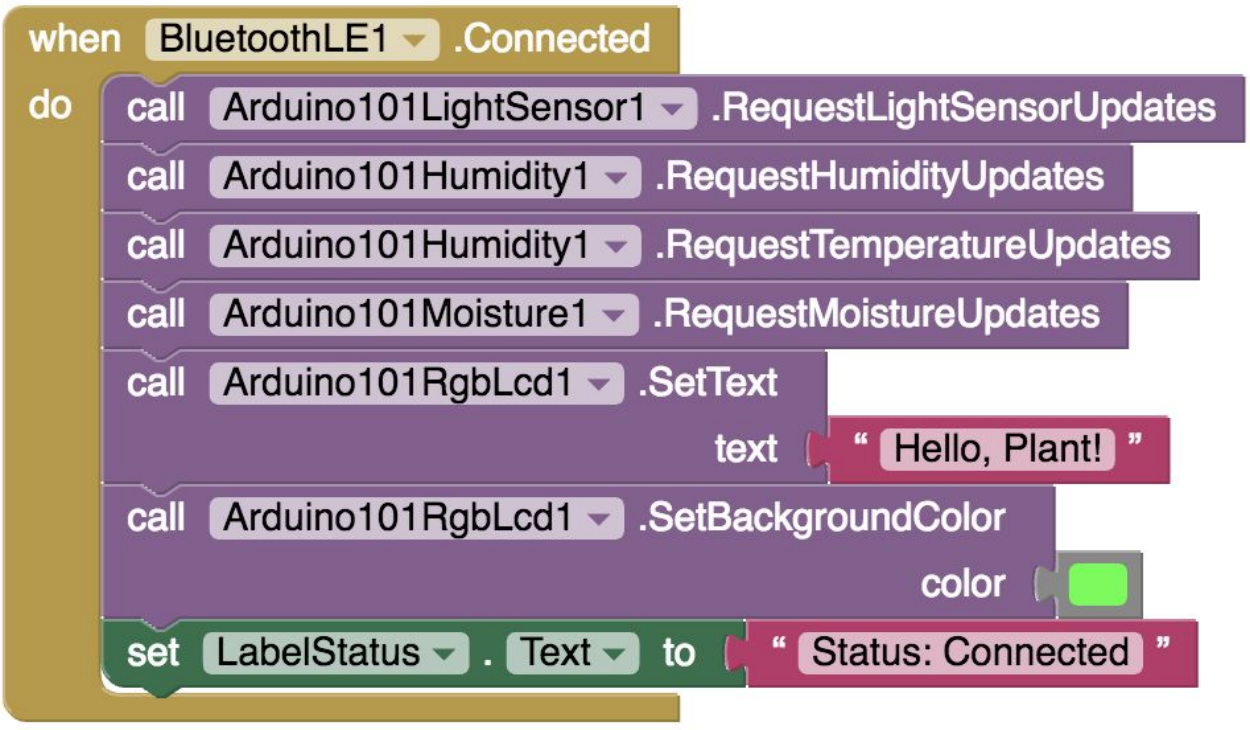
- From BluetoothLE1 in the Blocks pane drag out **when BluetoothLE1.Disconnected**
  - from LabelStatus add **set LabelStatus.Text to**
  - From the Text Drawer add a text block and type in **"Status: Disconnected"**

Next we want to set it up to request data updates when the values for each of our sensors on the Arduino changes.

- From the BluetoothLE1 in the Blocks pane drag out **when BluetoothLE1.Connected**
  - from Arduino101LightSensor1 add **call Arduino101LightSensor1.RequestLightSensorUpdates**
  - from Arduino101Humidity1 add **call Arduino101Humidity1.RequestHumidityUpdates**
  - from Arduino101Humidity1 add **call Arduino101Humidity1.RequestTemperatureUpdates**
  - from Arduino101Moisture1 add
    - **call Arduino101Moisture1.RequestMoistureUpdates**

Let's also send a message to the Arduino's LCD to make sure it is working.

- from Arduino101RGBLcd1 add **call Arduino101RGBLcd1.SetText**
  - From the Text Drawer add a text block and type in **"Hello Plant"**
- from Arduino101RGBLcd1 add **call Arduino101RGBLcd1.SetBackgroundColor color**
  - From the Color drawer add the **Green block**
- from LabelStatus add **set LabelStatus.Text to**
  - From the Text drawer add a text block and type in **"Status: Connected"**



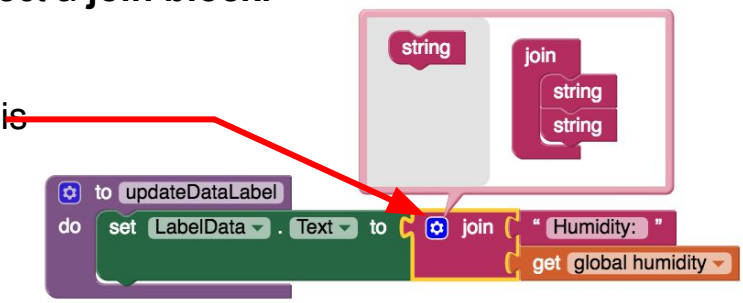
Now we need to store the data we receive from each of the sensors.

- From Variables drawer drag four **initialize global name to** blocks and name them **light**, **moisture**, **temperature**, and **humidity.**
    - set each one to a value of **0**

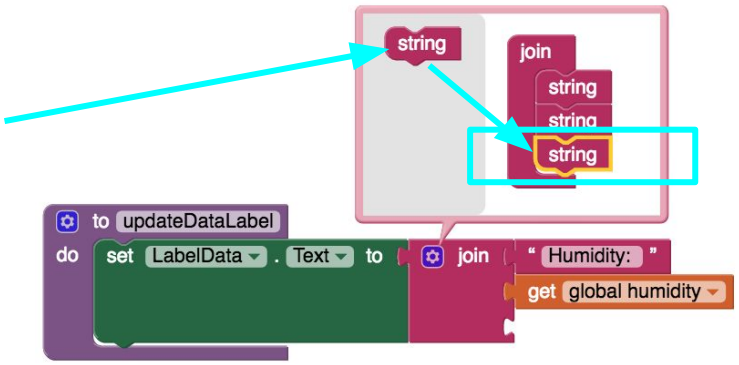
Let's make a new procedure to display the current readings in the LabelData. You can create a procedure by dragging out a purple procedure block from the Procedures Drawer in the Blocks pane. Let's rename it **updateDataLabel**

- From the LabelData Drawer add **set LabelData.Text to**
    - From the Text Drawer connect a **join block.**

You'll notice that the join only has two slots at first and we have 8 items! This is an easy fix. In the **Join** block you'll see a blue gear, click on it and a new window appears.

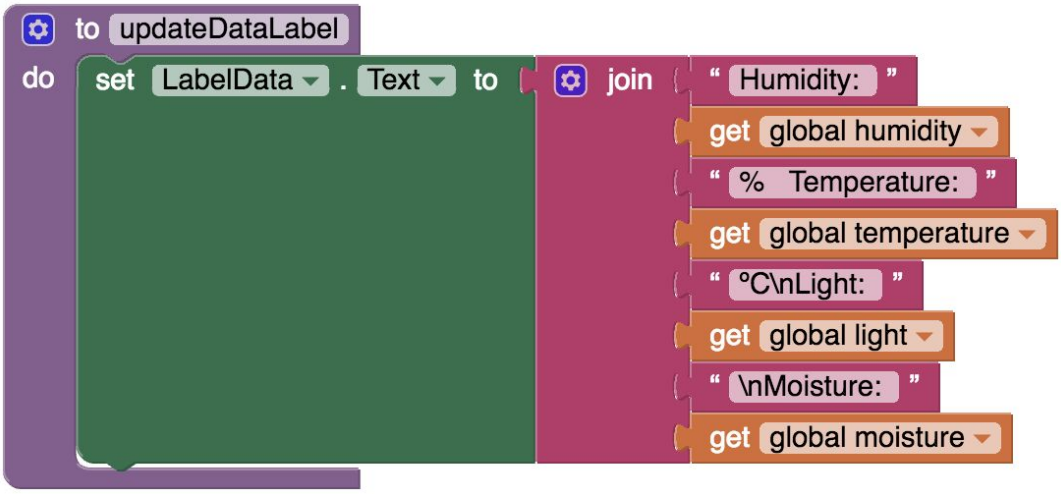


Then drag the **string** block on the left side under the **string** blocks inside the join. This will add a new slot. Do this 6 times in total.



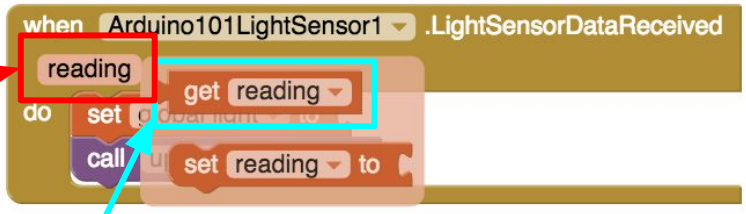
    - Add the following blocks to the join (for some of them it might be easier to copy and paste than to type them yourself):
        - from the Text Drawer add a text block and type in **"Humidity: "**
        - from the Variables Drawer add a block **get global humidity**
        - from the Text Drawer add a text block and type in **"% Temperature: "**
        - from the Variables Drawer add a block **get global temperature**
        - from the Text Drawer add a text block and type in **"ºC\nLight: "**
        - from the Variables Drawer add a block **get global light**
        - from the Text Drawer add a text block and type in **"\nMoisture: "**
        - from the Variables Drawer add a block **get global moisture**

Once we're done, the final procedure should look like this:

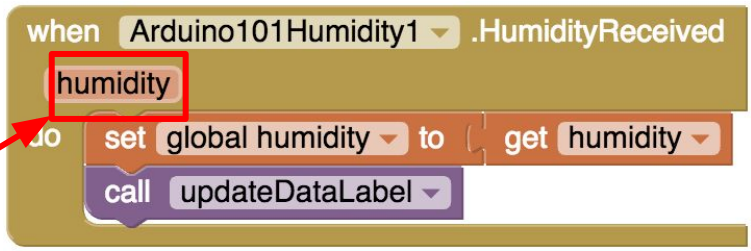


Next, we want to update the labels when we receive data from the sensors.

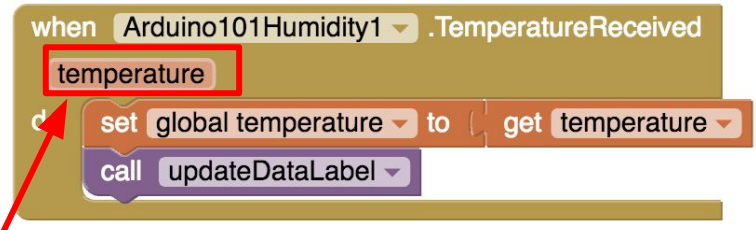
- From Arduino101LightSensor1 drag **when Ardunio101LightSensor1.LightSensorDataReceived**
- from Variables add **set global light to**
  - hover over the orange "reading" in the **.LightSensorDataReceived** block to see the **get reading** block.
  - Drag the **get reading** block from this window and snap to **set global light to**
- from Procedures add **call updateDataLabel**



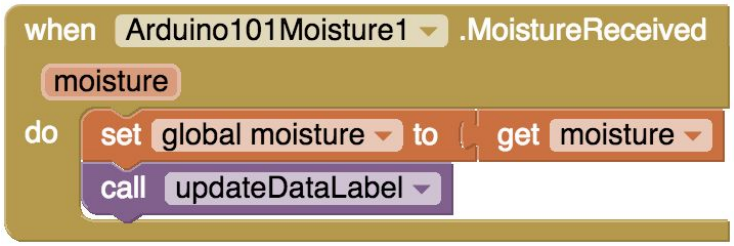
- From Arduino101Humidity1 drag **when Ardunio101Humidity1 .HumidityReceived**
  - from Variables add **set global humidity**
  - hover over the orange "humidity" to see **get humidity**
  - drag the **get humidity** block and snap to **set global humidity to**
  - from Procedures add **call updateDataLabel**



- From Arduino101Humidity1 drag **when Ardunio101Humidity1 .TemperatureReceived**
  - from Variables add **set global temperature to**
  - hover over “temperature” to see the **get temperature** block
  - drag the **get temperature** block and snap to **set global temperature to**
  - from Procedures add **call updateDataLabel**

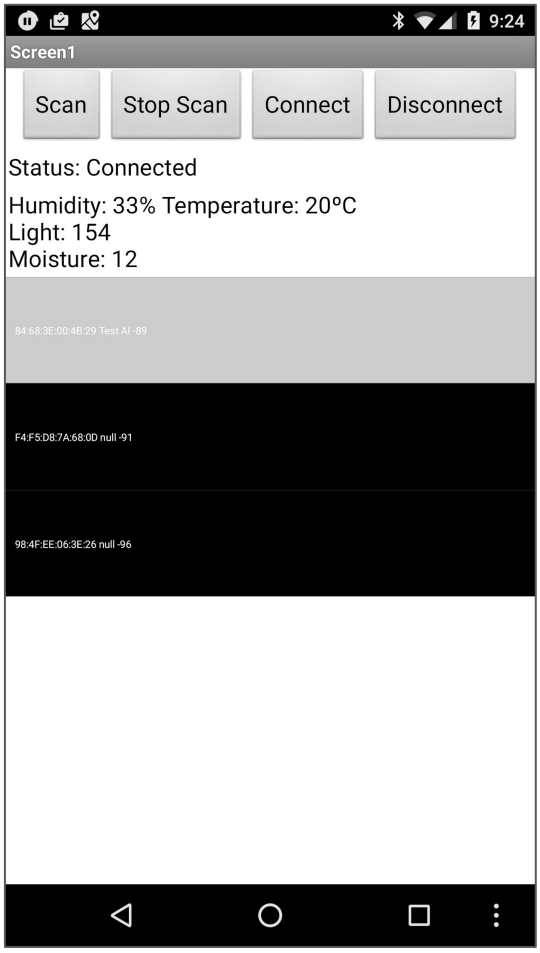


- From Arduino101Moisture1 drag **when Ardunio101Moisture1 .MoistureDataReceived**
  - from Variables add **set global moisture to**
  - hover over “moisture” to see the **get moisture** block
  - drag the **get moisture** block and snap to **set global moisture to**
  - from Procedures add **call updateDataLabel**



If you haven't already, now would be a good time to test out your app using the MIT AI2 companion. Once you've connected your device and run the Arduino .ino code, test the app using the following steps:

- Click the **Scan** button
  - You should see a list of BLE Devices
- When you see your device click **Stop Scan**
- Click on your device name in the list
- Click **Connect**
  - If your device successfully connects your **LabelStatus** should change to "Status: Connected".
- If everything works you should see the data changing for all the different sensors.
  - Try covering the light sensor, holding the humidity sensor in your hand, or getting the moisture sensor wet and see if the values change.

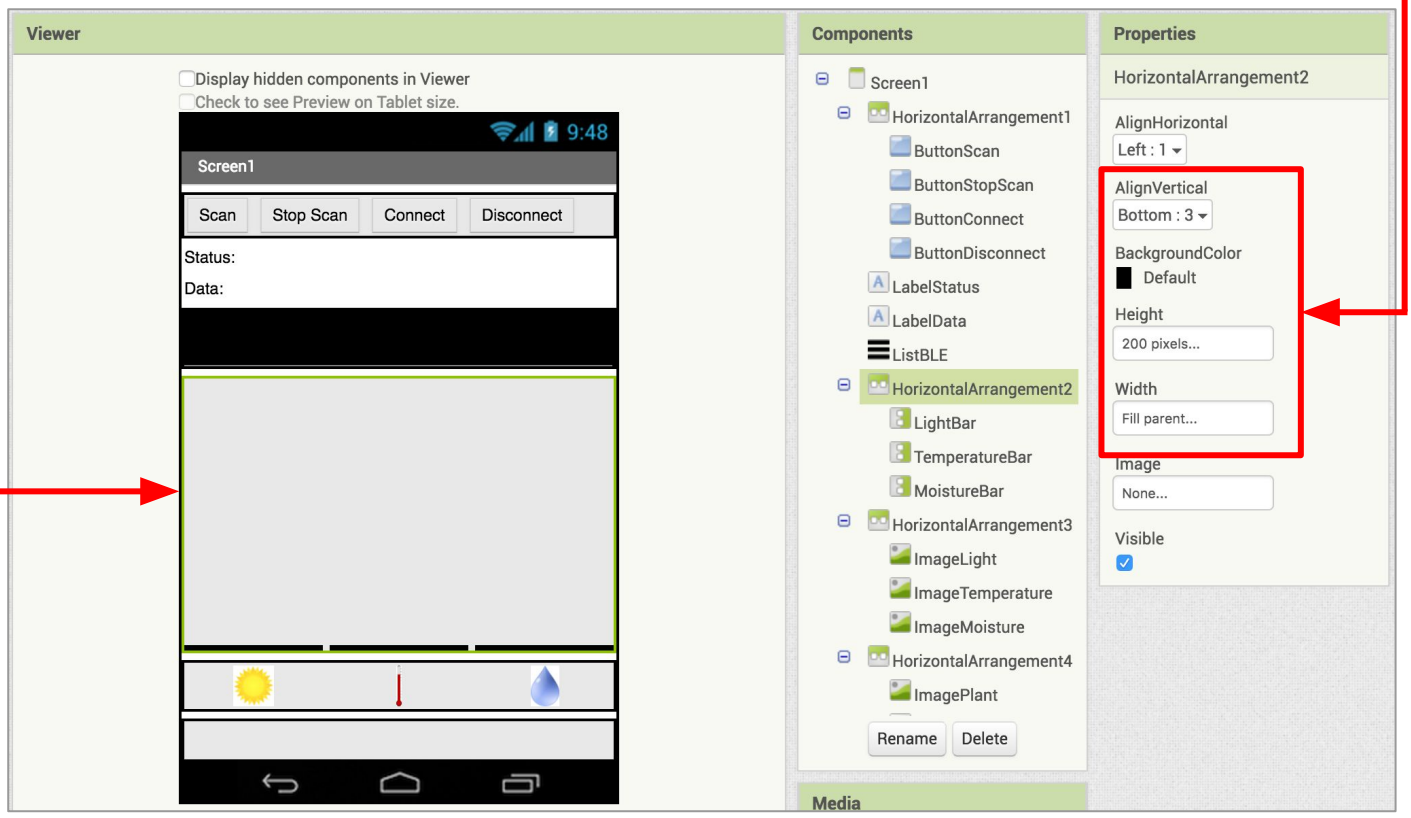


Now let's make it look nicer by adding some colorful bars to graph some of our data.

Switch to the Designer view

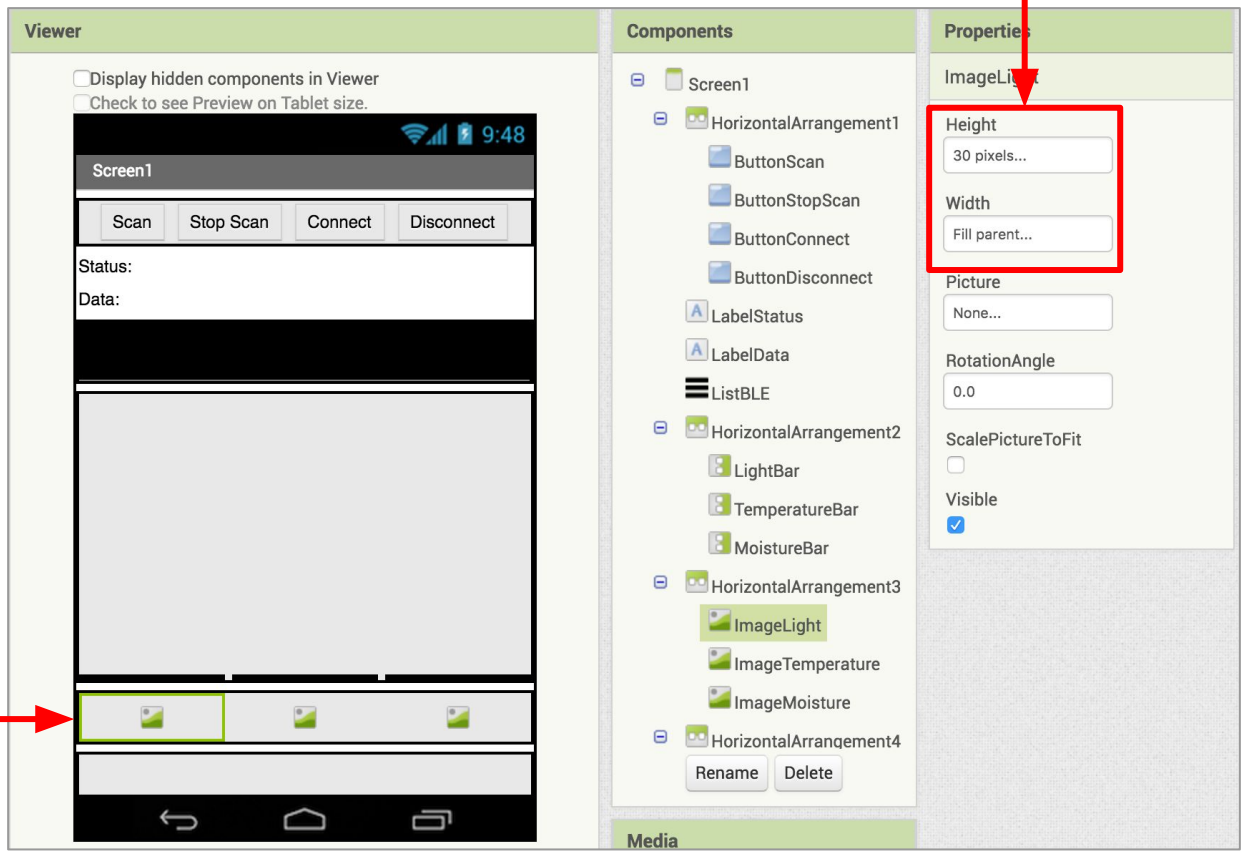
We need to create the area for the bar graphs.

- Drag a *HorizontalArrangement* from the Palette and place it below **ListBLE**
  - Set its properties as follows:
    - *AlignVertical*: **Bottom: 3**
    - *Height*: **200px**
    - *Width*: **Fill parent**
  - Add 3 *VerticalArrangements* inside the *HorizontalArrangement* and rename them **LightBar, TemperatureBar, MoistureBar**
  - Set each *VerticalArrangement*'s height to **0px** and width to **Fill Parent**
    - Now set **LightBar** *BackgroundColor* to Yellow, **TemperatureBar** *BackgroundColor* to Red, and **MoistureBar** *BackgroundColor* to Blue



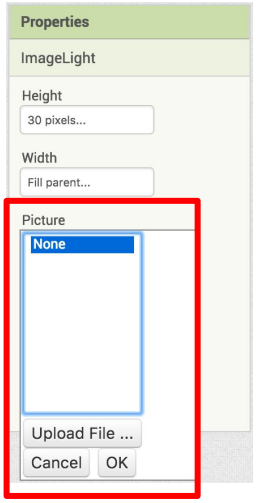
Let's create a legend so we know what each bar represents.

- Drag a *HorizontalArrangement* from the Palette and place it below *HorizonalArrangement2*
  - Leave its *Height* at **Automatic** and set its *Width* to **Fill parent**
  - From the User Interface Palette, drag 3 **Image** components onto the *Horizontal Arrangement*, and rename them "ImageLight", "ImageTemperature", and "ImageMoisture"
    - Set each Image's properties to:
      - *Height* to **30px** and *Width* to **Fill Parent**



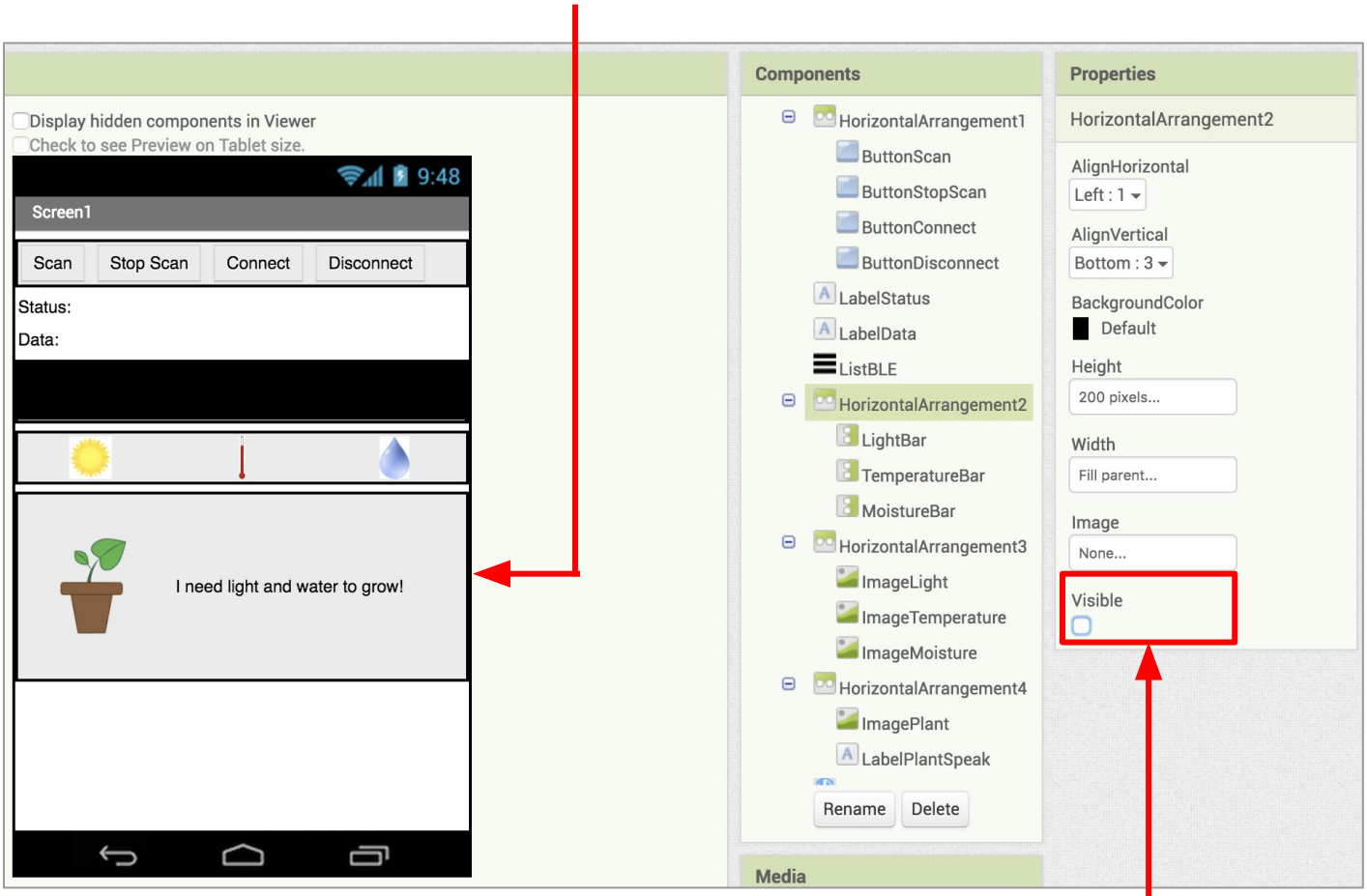
Now we want to add the images for the legend.

- Download the following 3 pictures to your computer:
  - Sunlight, Thermometer, and WaterDrop
- Under the Properties pane for **ImageLight**, click on Picture.
  - In the pop-up window click on "Upload File…"
  - Find the Sunlight image on your computer and upload it
  - Repeat this process for **ImageTemperature** and **ImageMoisture**



Let's set up a space so that our plant can "talk" to us based on its status.

- Drag a *HorizontalArrangement* from the Palette and place it below *HorizontalArragement3*
  - Set its *AlignVertical* to **Center: 2**, *Height* to **130px**, and *Width* to **Fill parent**
  - Download the PottedPlant picture to your computer and then Upload it to the project
    - Drag an **Image** component onto *HorizontalArrangement4*, and set its picture to the Potted Plant picture.
    - Rename the Image "ImagePlant"
    - Set its *Height* to **70px** and *Width* to 70**px**
  - Drag a **Label** to the right of ImagePlant, rename it "LabelPlantSpeak" and change its *text* to "I need light and water to grow!"

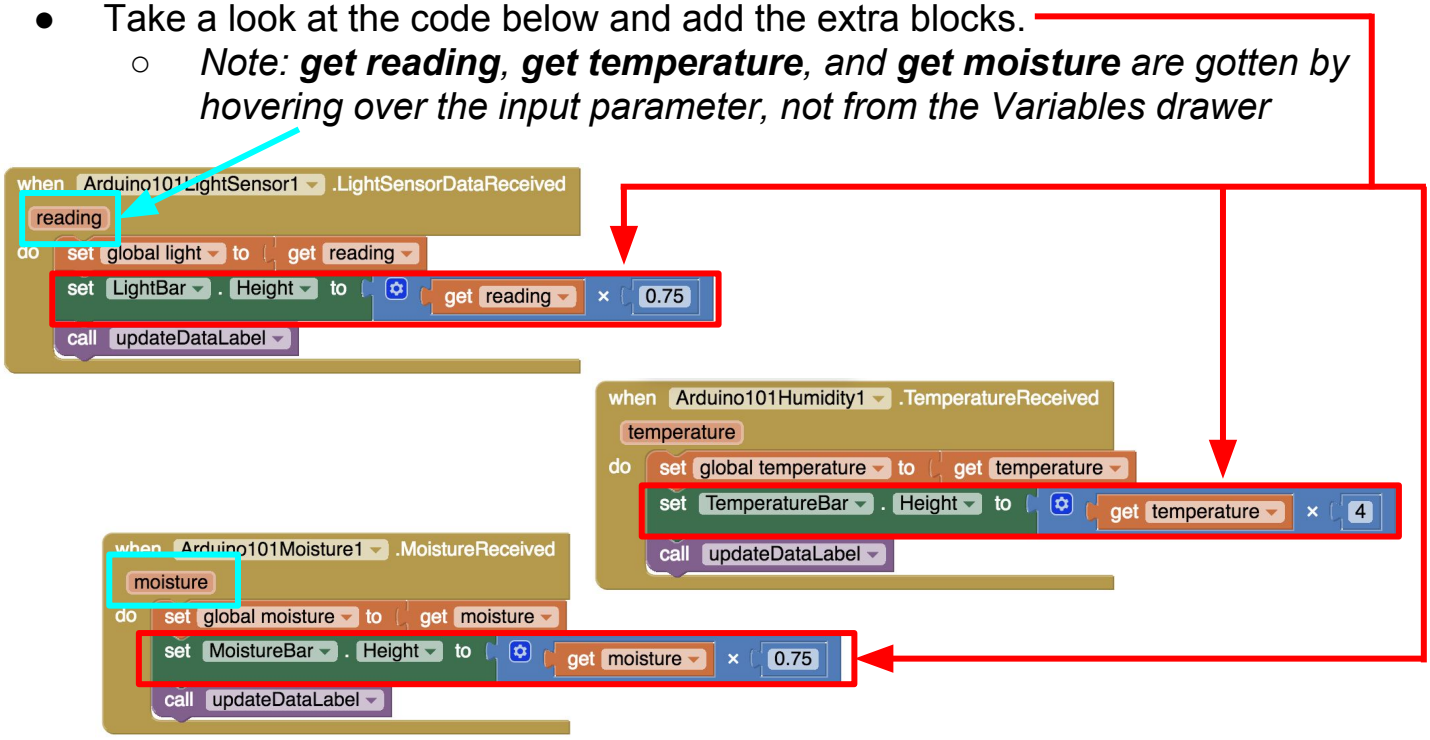


NOTE: If HorizontalArrangement4 is off the screen, try hiding HorizontalArragement2 temporarily by clicking the Visible button in HorizontalArragement2's Properties pane.

Switch to the Blocks Editor view

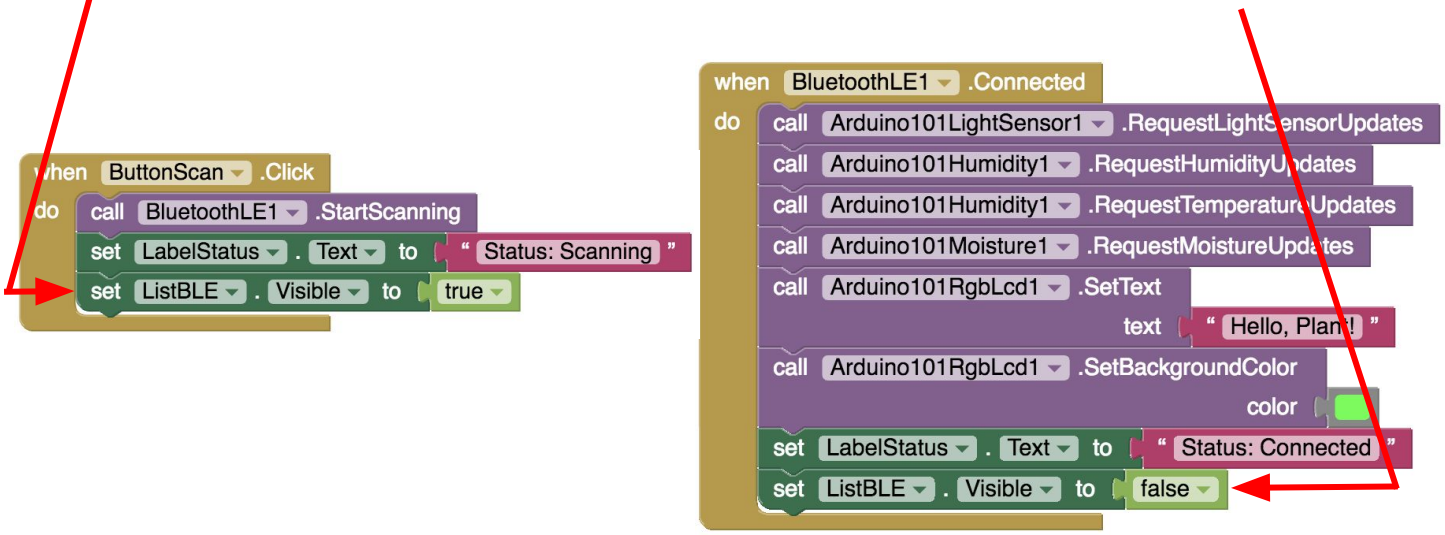
To update the graph as we get data, we're going change the last 3 groups of blocks we made. By setting the height of each of the vertical arrangements, we can create bars reflecting the sensor values by their change in size.

- Take a look at the code below and add the extra blocks.
  - *Note: **get reading**, **get temperature**, and **get moisture** are gotten by hovering over the input parameter, not from the Variables drawer*



Finally, we should hide and show **ListBLE** depending on if we need to see it or not.

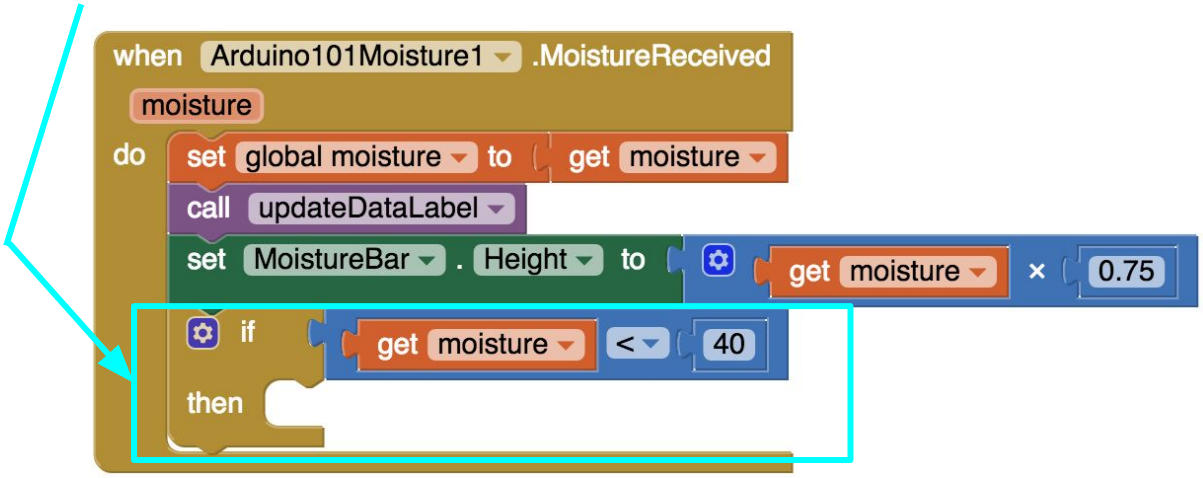
- To **when ButtonScan.Click** add **set ListBLE.Visible to true**
- To **when BluetoothLE1.Connected** add **set ListBLE.Visible to false**



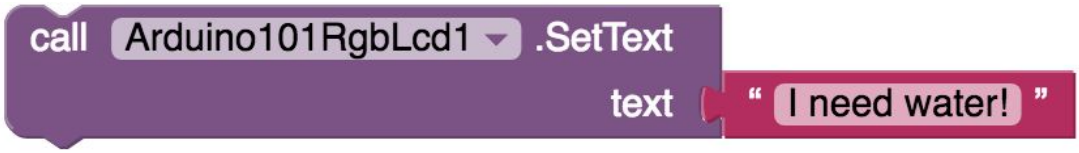
Now try out your app using the MIT AI2 companion - when the sensor data changes, the bar graphs should also go up and down.

A few other things you could do to enhance your app!

- Using "if" statements, change the **LabelPlantSpeak** depending on the plant's conditions (e.g., too hot, too dry, when it is watered).



- You could also send or receive messages to the LCD.



- Or, change the colors of the graph bars when the conditions change.

This is just one example of how App Inventor + IoT can work together to help us understand, and change, our everyday lives. If you come up with more, be sure to share them with us! You can reach us by emailing appinventor@mit.edu . Enjoy!