

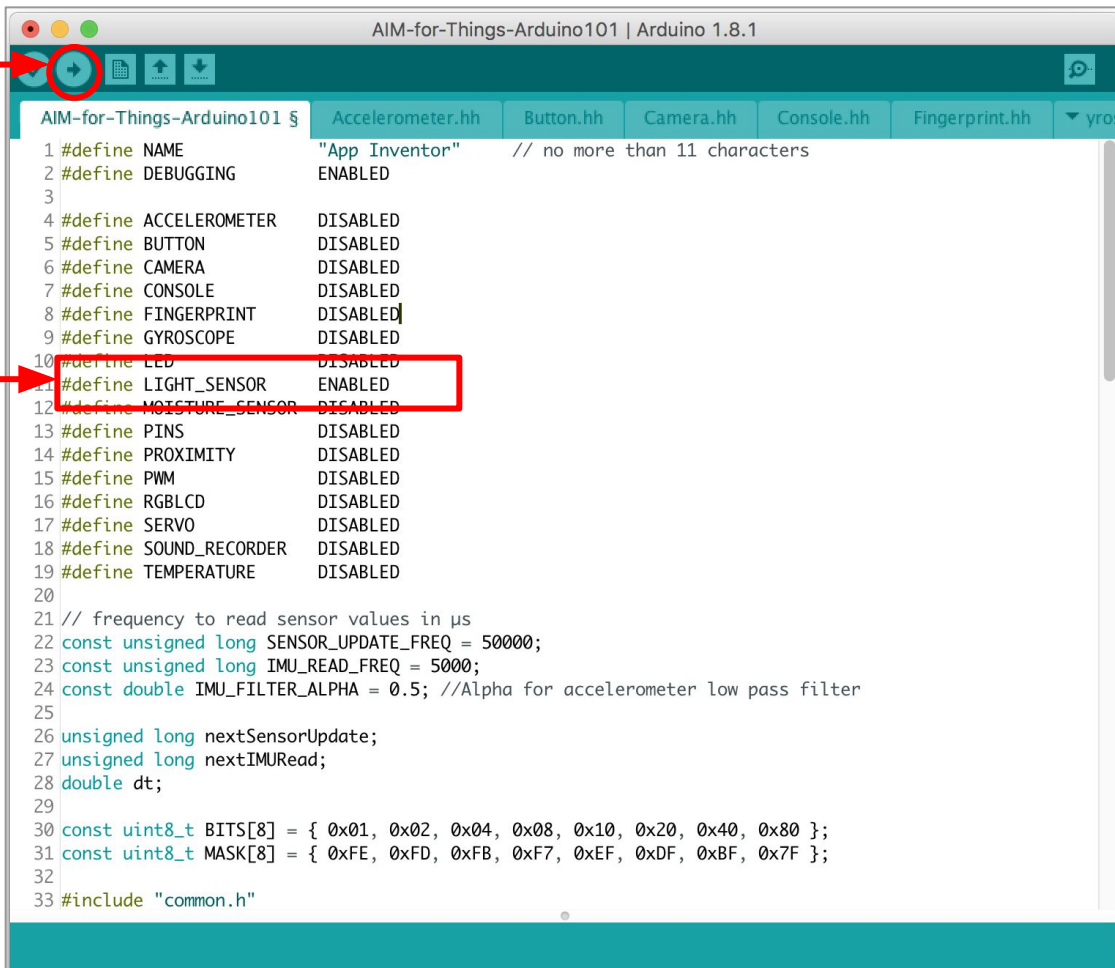
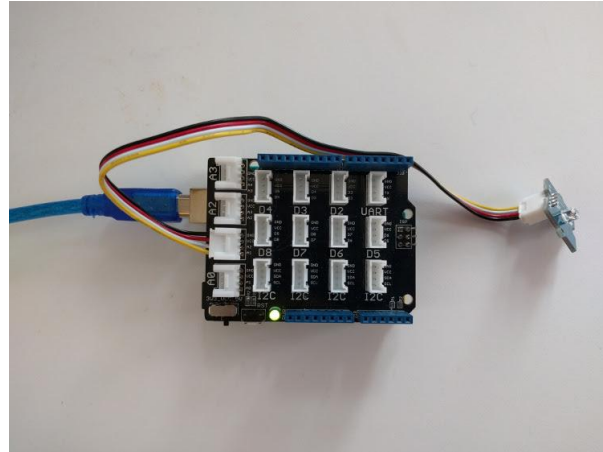


App Inventor + IoT: Light Sensor

This tutorial will help you get started with App Inventor + IoT with a light sensor on an [Arduino 101](#) controller. 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. The light sensor we recommend is the [Grove light sensor](#).

Before you start you should have first completed the [App Inventor + IoT Setup tutorial](#) to set up your Arduino device.

- Connect the light sensor to the Grove board in the A1 pin connector. 
- For this tutorial make sure **LIGHT_SENSOR** is 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.



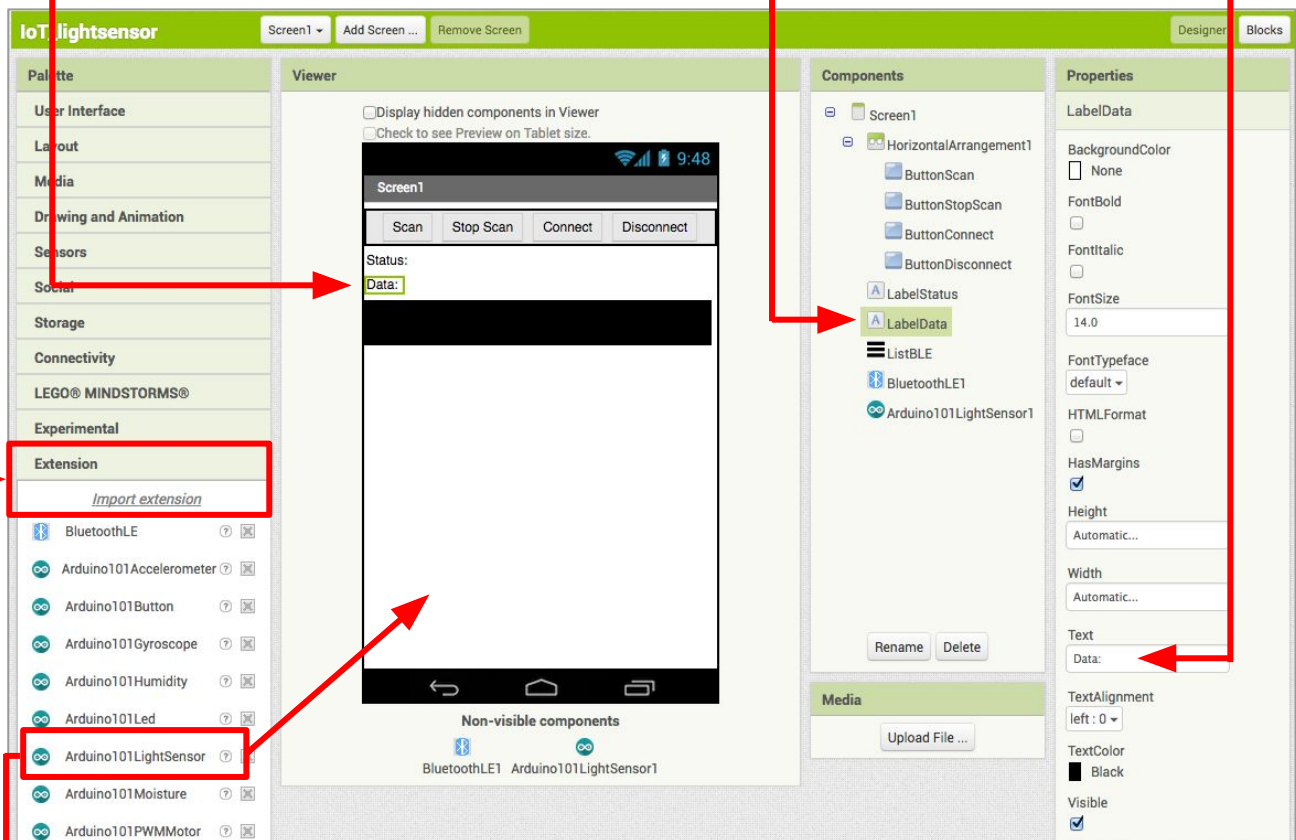
```
AIM-for-Things-Arduino101 | Arduino 1.8.1
Accelerometer.hh Button.hh Camera.hh Console.hh Fingerprint.hh yos

1 #define NAME "App Inventor" // no more than 11 characters
2 #define DEBUGGING ENABLED
3
4 #define ACCELEROMETER DISABLED
5 #define BUTTON DISABLED
6 #define CAMERA DISABLED
7 #define CONSOLE DISABLED
8 #define FINGERPRINT DISABLED
9 #define GYROSCOPE DISABLED
10 #define LED DISABLED
11 #define LIGHT_SENSOR ENABLED
12 #define MOISTURE_SENSOR DISABLED
13 #define PINS DISABLED
14 #define PROXIMITY DISABLED
15 #define PWM DISABLED
16 #define RGBLCD DISABLED
17 #define SERVO DISABLED
18 #define SOUND_RECORDER DISABLED
19 #define TEMPERATURE DISABLED
20
21 // frequency to read sensor values in µs
22 const unsigned long SENSOR_UPDATE_FREQ = 50000;
23 const unsigned long IMU_READ_FREQ = 5000;
24 const double IMU_FILTER_ALPHA = 0.5; //Alpha for accelerometer low pass filter
25
26 unsigned long nextSensorUpdate;
27 unsigned long nextIMURead;
28 double dt;
29
30 const uint8_t BITS[8] = { 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80 };
31 const uint8_t MASK[8] = { 0xFE, 0xFD, 0xFB, 0xF7, 0xEF, 0xDF, 0xBF, 0x7F };
32
33 #include "common.h"
```

Next, you should complete the [App Inventor + IoT Basic Connection](#) tutorial to make the app connect to the Arduino device. If you prefer, you can download the completed .aia file [here](#).

The remaining steps all build off of the the starter code for Basic Connection tutorial and .aia:

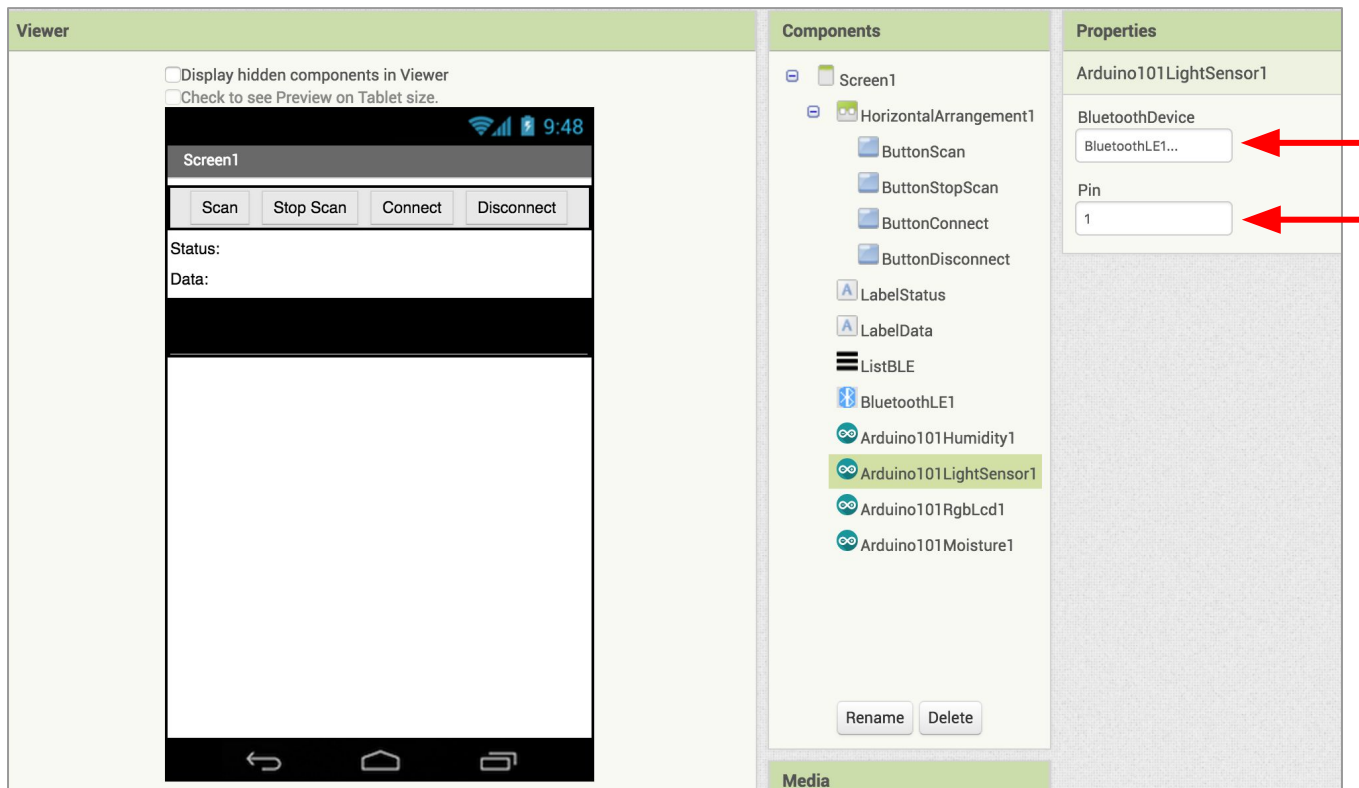
- Drag a **Label** from the User Interface Palette and drop it between **LabelStatus** and **ListBLE**
 - Rename the **Label** "LabelData".
 - Change its text to "Data:".



- Download the [Arduino101](#) extension pack to your computer.
- In the Palette, click on Extension at the bottom and then on "Import extension" and then "Choose File". Find the file on your computer and upload it.
- Add the **Arduino101LightSensor** extension to your app by dragging it onto the Viewer.

Next, we need to let App Inventor know which pin on the Grove board the light sensor is connected to.

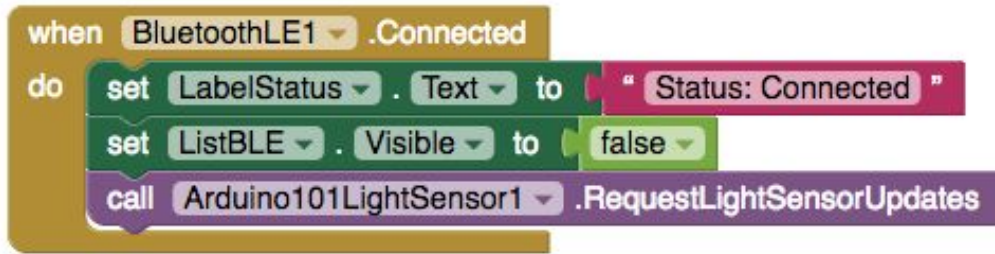
- Click on **Ardunio101LightSensor1** in the Components pane.
- In the Properties pane, click on BluetoothDevice and select **BluetoothLE1**. Under **Pin**, enter the analog pin that matches the one the light sensor is plugged into on the Grove board (in this case A1).
 - *Note: You only need to set the number (1) not the letter (A).*



Now switch to the Blocks Editor view

First, we want to set it up so that we request data updates when the sensor value on the Arduino changes.

- from the Arduino101LightSensor1 drawer in the Blocks pane, add **call Arduino101LightSensor1 .RequestLightUpdates** to the existing **when BluetoothLE1.Connected** block

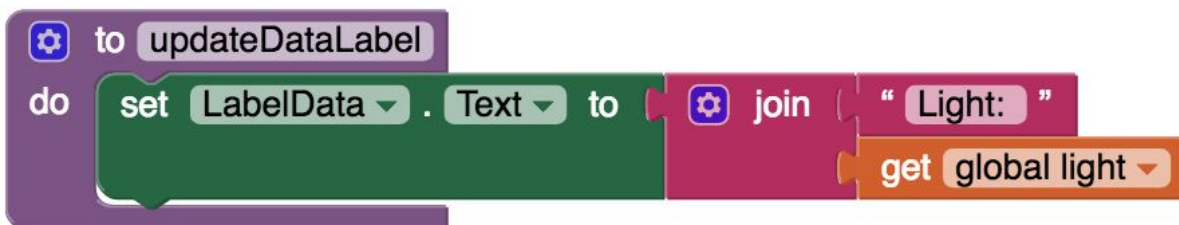


Next we need to store the data we receive from the sensor. From the Variables drawer in the Blocks pane, drag an **initialize global name to** block and name it "light". From the Math drawer add a number block and set it to "0". We'll use this to keep track of the sensor value.



Let's make a new procedure to display the current readings in the **LabelData** when we get new data. 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 in the Blocks pane, add **set LabelData.Text to**.
 - from the Text drawer, add a **join** block.
 - from the Text drawer, add a text block and type **"Light: "**.
 - from the Variables drawer, add **get global light**.



Finally, we need to call the procedure when this data is received.

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



Your app should now be working! Connect your Android device using the companion (if you haven't already). Test it out by shining a light at, or covering, the light sensor. If it is working, you should see the data label change.

