

App Inventor + IoT: Micro:bit Temperature

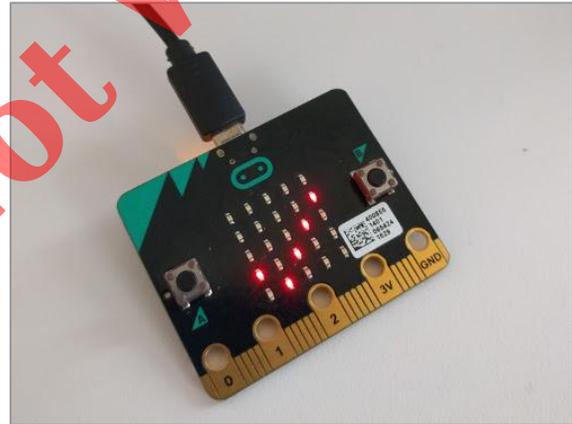
20
min

(with Basic Connection
tutorial completed)

This tutorial will help you get started with App Inventor + IoT and the temperature sensor on a [micro:bit](#) controller.

First, you will need to pair your phone or tablet to the micro:bit controller, using these [directions](#). Your device must be paired with the micro:bit in order for the app to work.

Next, you should complete the [App Inventor + IoT Basic Connection](#) tutorial to make a basic connection to the micro:bit device. If you prefer, you can download the completed .aia file [here](#).



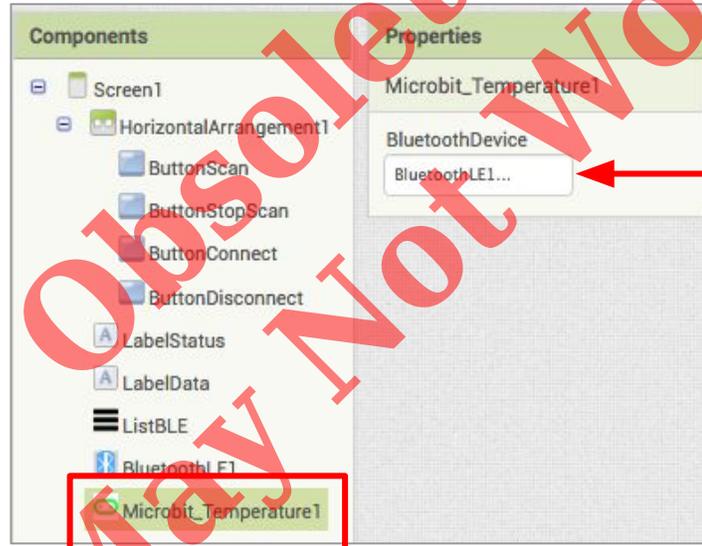
The remaining steps all build off of the the starter code for Basic Connection tutorial and .aia.

First, we need to add the necessary extension.

- In the Palette window, click on Extension at the bottom and then on "Import extension" and click on "URL".
 - Paste in this URL:
`http://iot.appinventor.mit.edu/assets/com.bbc.microbit.profile.aix`
- Add the **Microbit_Temperature** extension to your app by dragging it onto the Viewer.

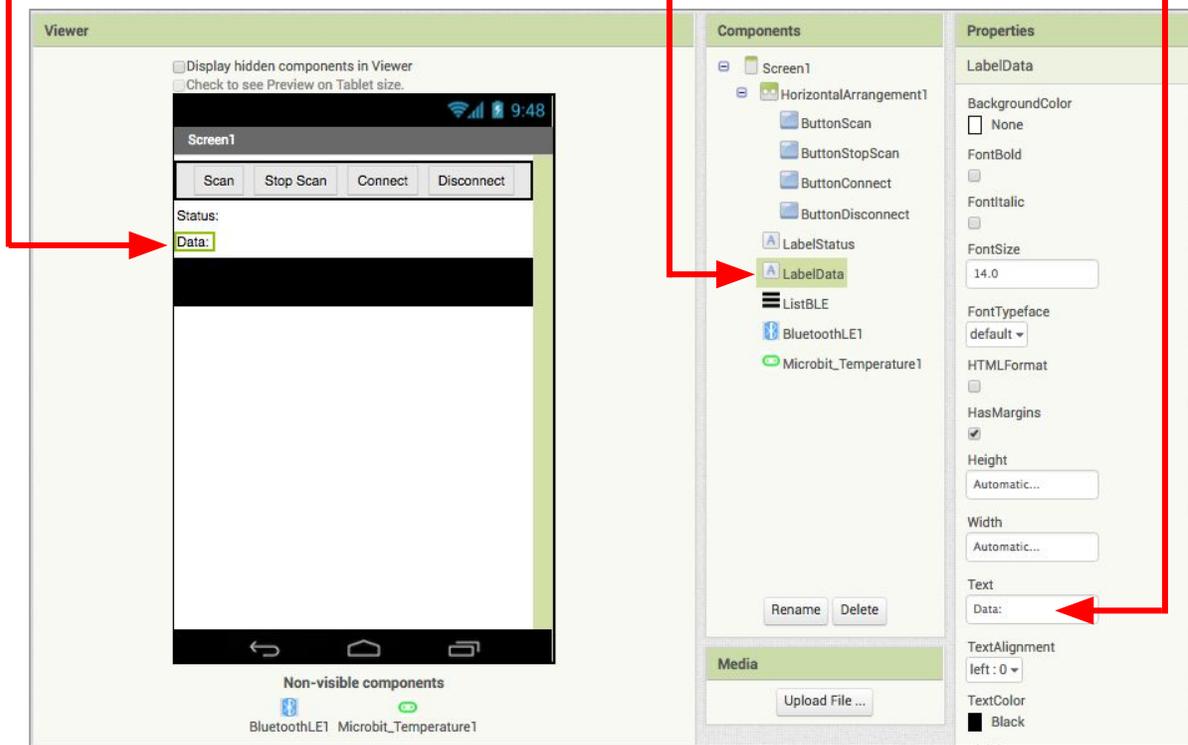
The screenshot shows the App Inventor interface with four main panels: Palette, Viewer, Components, and Properties. In the Palette window, the 'Extension' section is expanded, and the 'Import extension' button is highlighted with a red box. A red arrow points from this button to the Viewer window. In the Viewer window, a mobile app interface is shown with buttons for 'Scan', 'Stop Scan', 'Connect', and 'Disconnect', and labels for 'Status:' and 'Data:'. A red arrow points from the 'Microbit_Temperature' extension in the Components list to the Viewer. Another red arrow points from the 'Microbit_Temperature' extension in the Components list to the 'Microbit_Temperature' extension in the Components list. The Properties window on the right shows the settings for the selected 'Microbit_Temperature' extension.

- Click on **Microbit_Temperature1** in the Components pane.
- In the Properties tab for the **Microbit_Temperature1**
 - Set *BluetoothDevice* to "BluetoothLE1".



- Drag a **Label** from the User Interface Palette and drop it between **LabelStatus** and **ListBLE**

- Rename the **Label** "LabelData".
- Change its text to "Data: ".



Now switch to the Blocks Editor view

First, we want to request data updates when the sensor value on the micro:bit changes.

- from **Microbit_Temperature1** in the Blocks pane, add **call Microbit_Temperature1.RequestTemperatureUpdates** to the existing **when BluetoothLE1.Connected** block from the Basic Connection tutorial.

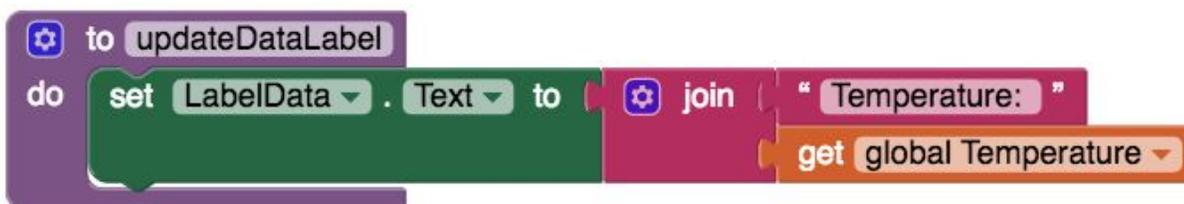


Next, we need to store the data we receive from the sensor. From the Variables drawer in the docs pane, drag an **initialize global name to** block and name it "Temperature". From the Math drawer add a number block and set it to "0". We'll use this to keep track of the sensor value.



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 **LabelData** in the Blocks pane, add **set LabelData.Text to**.
- from the Text drawer connect a **join** block.
 - From the Text drawer, connect a text block and type **"Temperature: "**
 - From the Variables drawer connect a **get global Temperature**.



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

- From the **Microbit_Temperature1** drawer in the Blocks pane, drag **when Microbit_Temperature1.TemperatureReceived**
 - from the Variables drawer, add **set global Temperature**.
 - Hover over the orange "temperature" in **.TemperatureReceived** to see the **get temperature** block. Drag the **get temperature** block from this window and snap to **set global Temperature**.
 - from the Procedures drawer, add **call updateDataLabel**.



Your app should now be working! Connect your micro:bit device using the companion (if you haven't already). Test it out by closing your hand around the micro:bit. If it is working, you should see the data label change (but probably very slowly).

