App Inventor + IoT: Button

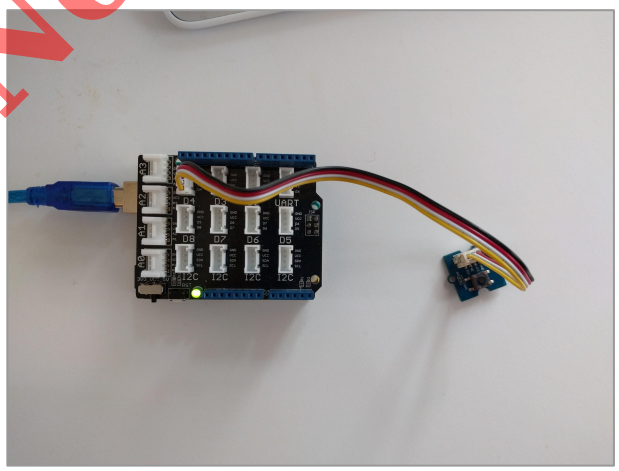
30 min

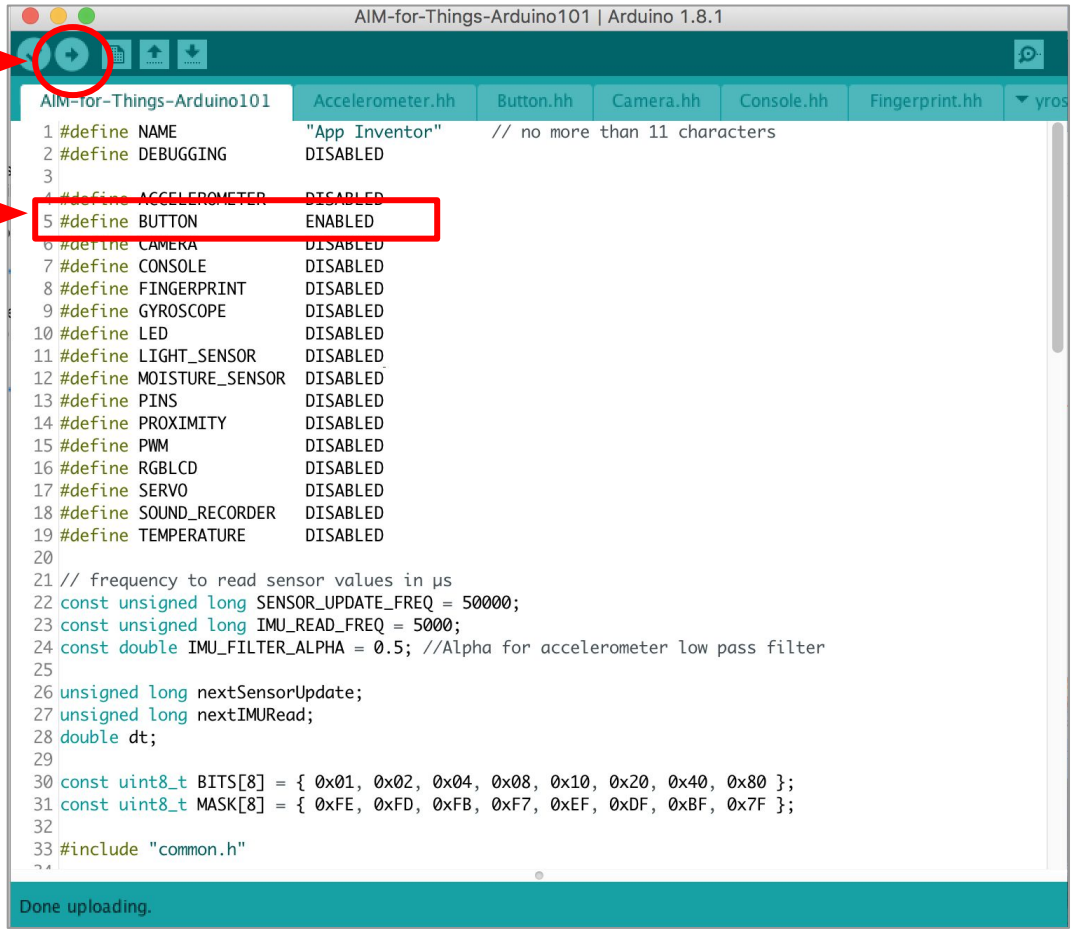
(with IoT Setup and Basic Connection tutorials completed)

This tutorial will help you get started with App Inventor + IoT and a button 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 button we recommend is the Grove button sensor.

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

- Connect the button to the Grove board in the D4 pin connector.
- For this tutorial make sure **BUTTON** 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

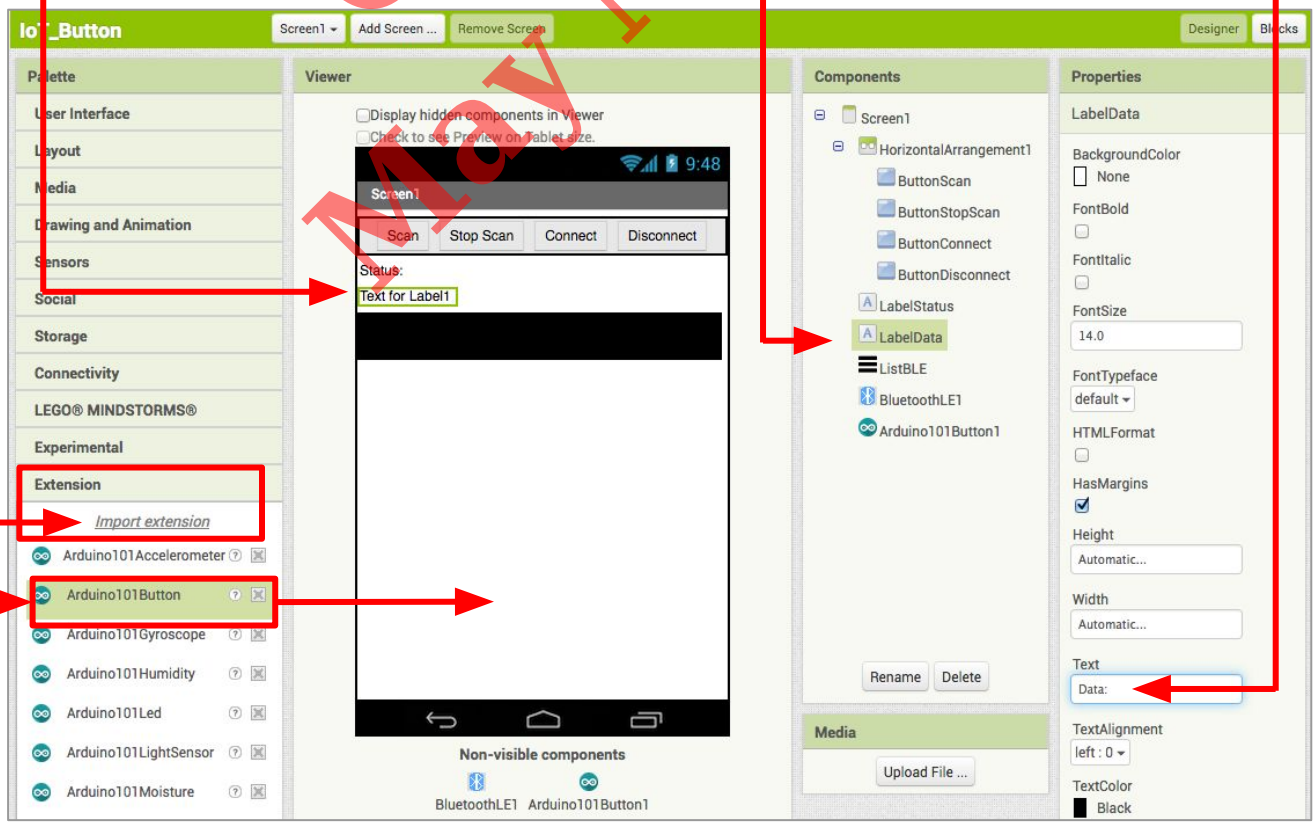




Next, you should complete the App Inventor + IoT Basic Connection tutorial to make the app with a basic connection to the Arduino device. If you prefer, you can download the completed .aia file here.

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

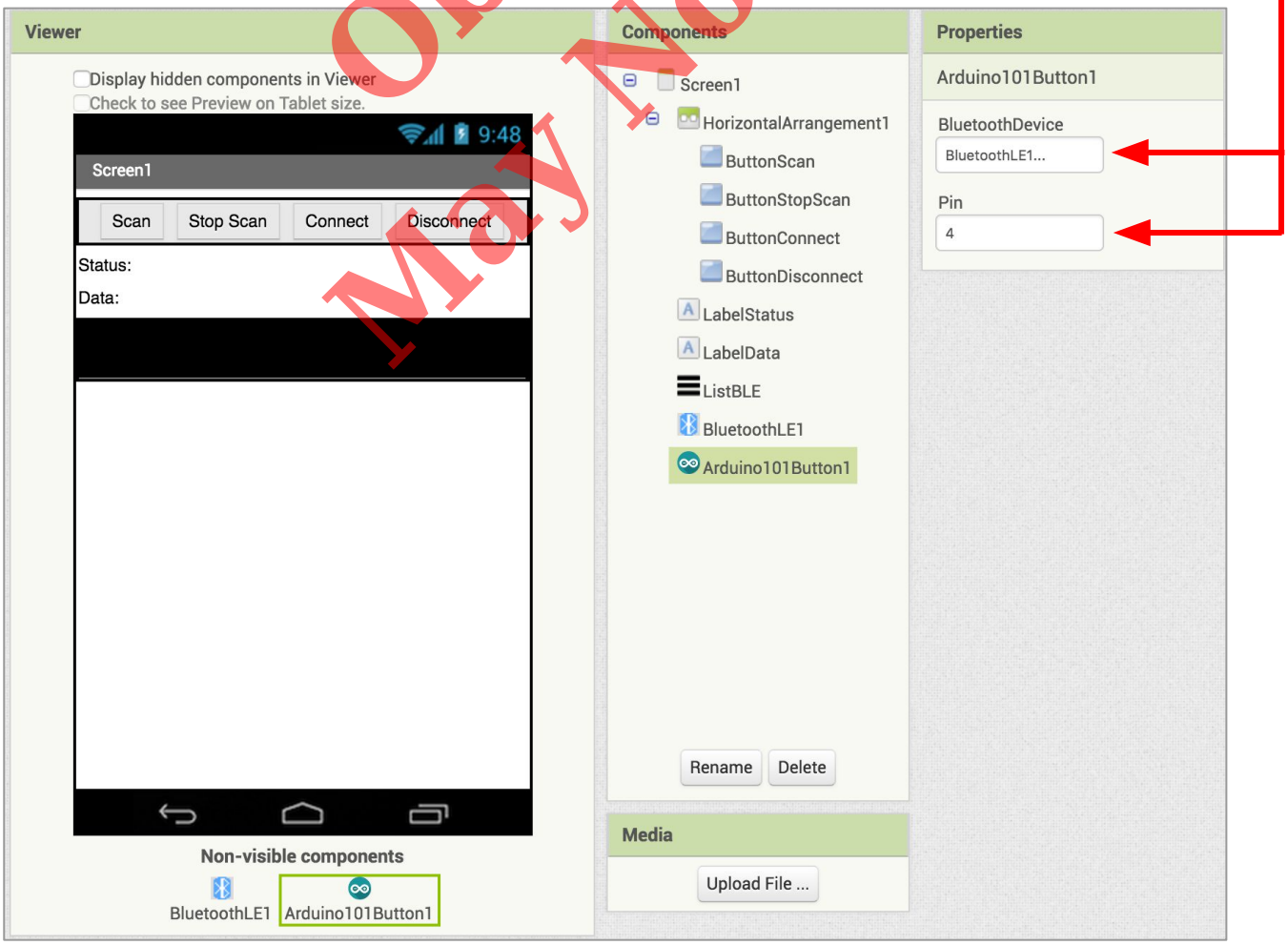
- Drag a **Label** from the User Interface Palette and drop it between **LabelStatus** and **ListBLE**
  - Rename the **Label** "LabelData".
  - Change its text to “Data: “.



- 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/resources/edu.mit.appinventor.iot.arduino101.aix
- Add the **Arduino101Button** 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 button is connected to:

- Click on **Arduino101Button** in the Components pane.
- In the Properties pane, click on BluetoothDevice and select **BluetoothLE1**. Under **Pin**, enter the digital pin that matches the one the button is plugged into on the Grove board (in this case D4).
    - *Note: You only need to set the number (4) not the letter (D).*



Now switch to the Blocks Editor view

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

- from Arduino101Button1 in the Blocks pane, add **call Arduino101Button1.RequestButtonStateUpdates** to the existing **when BluetoothLE1.Connected** block

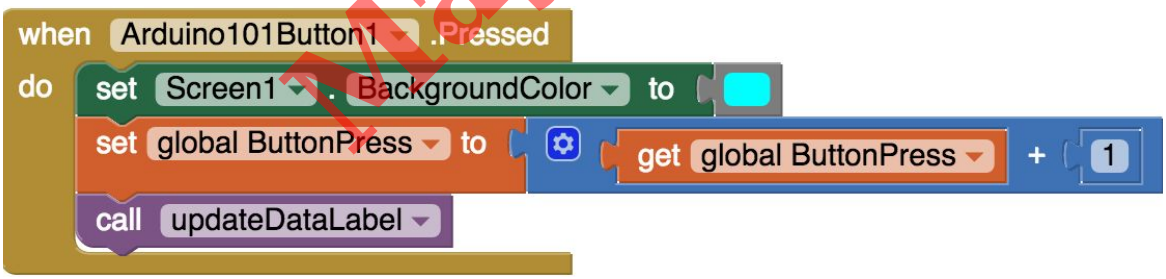
Next we need to store the data we receive from the sensor. From the Variables drawer in the Blocks pane, drag an **initialize global name to** block and name it "buttonPresses". From the Math drawer add a number block and set it to "0". We'll use this to keep track of the number of times the button is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

Let's make a new procedure to display the number of times the button is pressed 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 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 **"Button Presses: "**
  - From the Variables drawer connect a **get global buttonPresses.**

The Button has two states that we can code for: When the Button is ***Pressed*** and when the Button is ***Released***. This means that you can have some things happen each time the button is pressed or held down, and some things happen each time the button is released. Let's see how each of these work.
First, let's program the behavior for when the button is pressed down. From Arduino101Button1 in the Blocks pane drag in the
when Ardunio101Button1.Pressed block.

- From Screen1 in the Blocks pane, add in **set Screen1.BackgroundColor to.**
  - And from Color drawer in the Blocks pane, connect a **Cyan** block.
- From the Variables drawer, add **set global ButtonPress to.**
  - Then from the Math drawer connect an **Addition** block.
    - Next, from the Variables drawer add **get globalButtonPress** to the Addition block.
    - From the Math drawer, add a **0** and change it to **1.**
- Finally, from the Procedures drawer add **call updateDataLabel.**



Now let's code for when the button is released. From the Arduino101Button1 drawer in the Blocks pane, drag in the **when Ardunio101Button1.Released** block.

- From Screen1 in the Blocks pane, add in **set Screen1.BackgroundColor to.**
  - And from the Color drawer, connect a **White** block.

Your app should now be working! Connect your Arduino device using the MIT AI2 Companion (if you haven't already). Test it out by pressing and releasing the button. You should see the counter go up each time and the screen's background turn Cyan. When you release the button the screen should turn back to White.