

App Inventor + IoT: read button status with Micro:bit I/O pins



(with Basic Connection
tutorial completed)

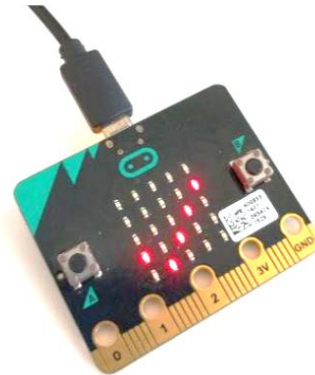
Level: advanced

This tutorial will help you work with App Inventor + IoT and read the status of a button connected to a [micro:bit](#) controller.

- [source .aia](#)

Pairing with Micro:bit

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.

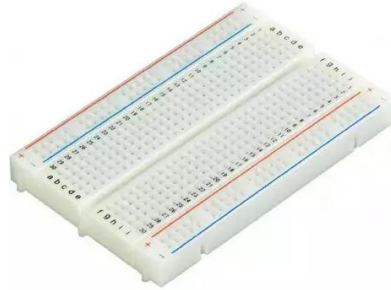
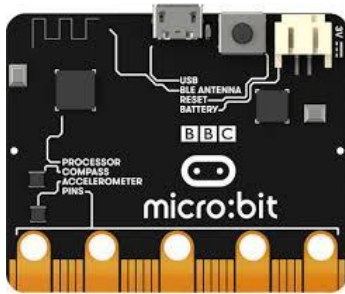


Hardware list

In this project, we are going to detect whether a button (which is connected to Micro:bit) is pressed using App Inventor. When that button is pressed, the bee icon on screen will change to a random position.

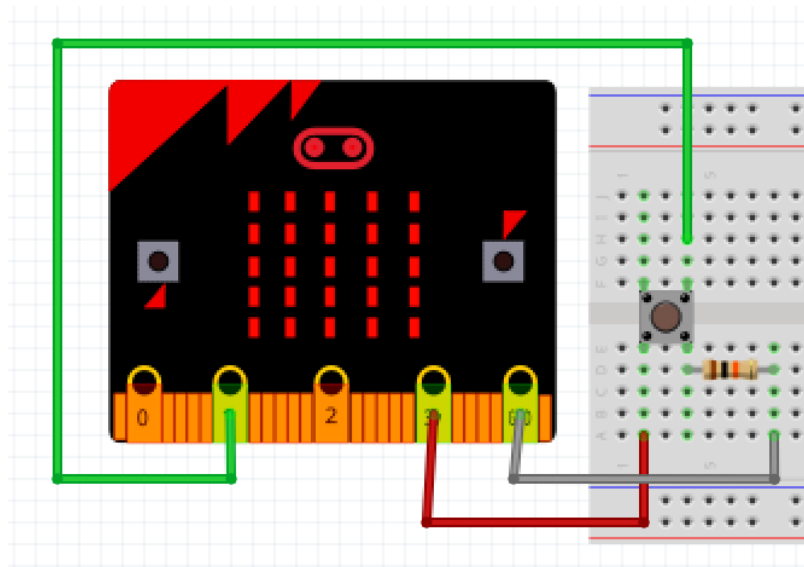
Here are the components you need for this project:

- BBC micro:bit dev board, 1
- breadboard, 1
- wires, 3
- potentiometer, 1

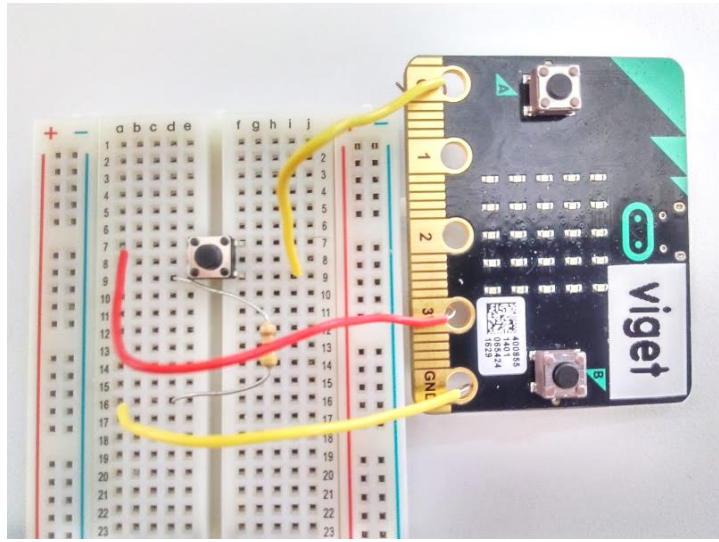


Micro:bit	potentiometer
GND	right pin (grey wire)
P0	middle pin (green wire)
3V	left pin (red wire)

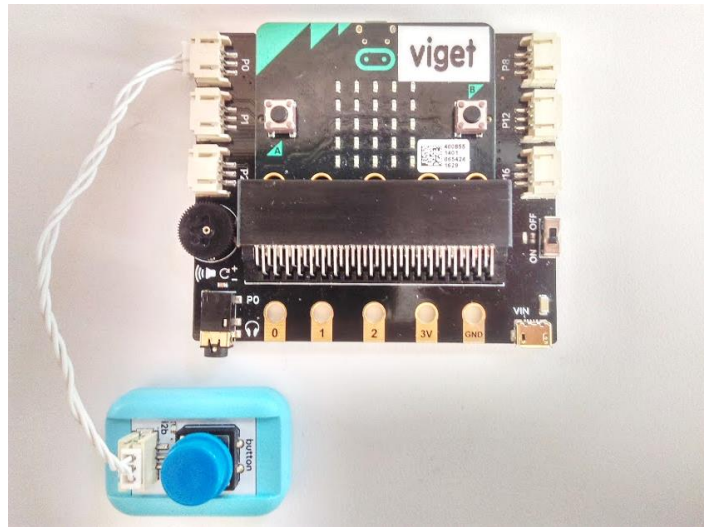
Note: potentiometer is a non-polarized component, which means the difference to connect potentiometer **left** pin to MCU board's 5V pin (another pin to GND pin) is that when you rotate potentiometer shaft clockwise, the pin value will change in opposite way than to connect its **right** pin to MCU board's 5V pin.



Finish as below, let's take a look:



Or you can use extension board, like [DFRobot's BOSON extension board](#):



App Inventor

This app will move a imageSprite randomly when you press the button connected to micro:bit's P0 pin. Technically speaking, App Inventor is asking micro:bit to report its pin status and this is where we connect the button to. First, log into [MIT App Inventor site](#) and create a new project.

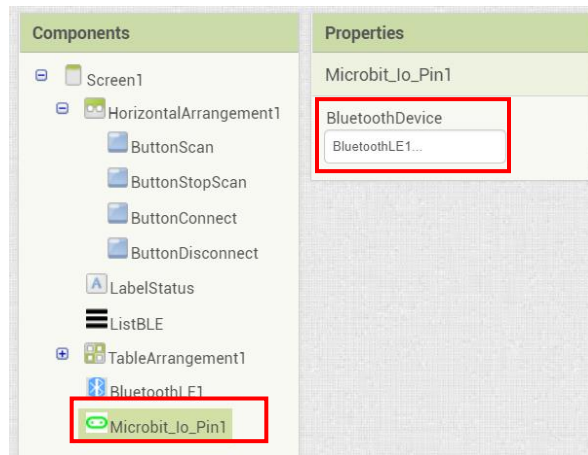
Designer

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.micro:bit.profile.aix>
- Add a **Microbit_IOPin** extension to your app by dragging it onto the Viewer, set its *BluetoothDevice* to "**BluetoothLE1**"(Don't forget!).



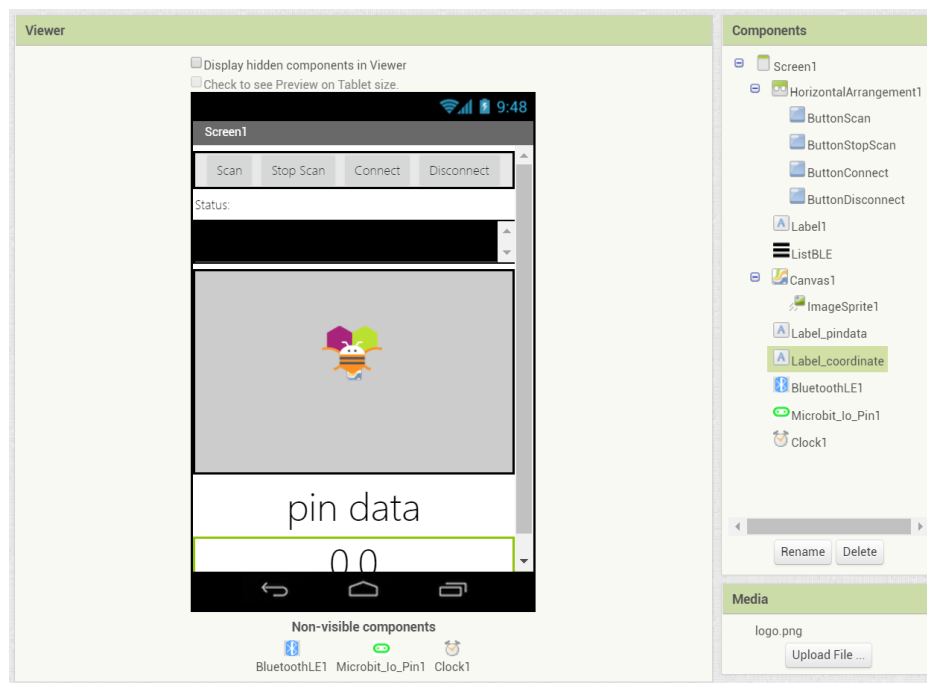
Let's add more components to our app to read the magnetometer status.

- From the Drawing and animation palette, add a **Canvas** component. Set its width to "**Fill**"

parent", height to **"200 pixels"**.

- From the Drawing and animation palette, add an **ImageSprite** component, set its Picture to some cute image (no bigger than the canvas).
- Add a **label** component, set its FontSize to **40** and Text to **"pin data"**. We will update the micro:bit P0 pin value here.
- Add one more **label** component, set its FontSize to **40** and Text to **"0, 0"**. We will update the latest position of the imageSprite here.
- From Sensor palette, add a Clock component, set its TimerInterval to **100**, which means its timer will trigger 10 times per second.

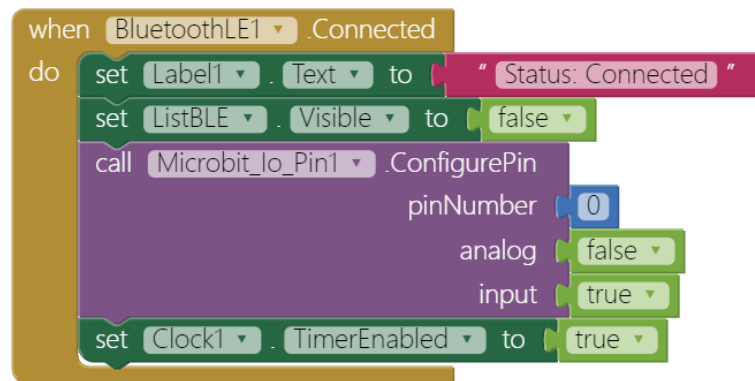
After some adjusting, your designer should look similar to this. It doesn't have to be exactly the same. Feel free to modify the component properties, such as background color, position and text size.



Blocks

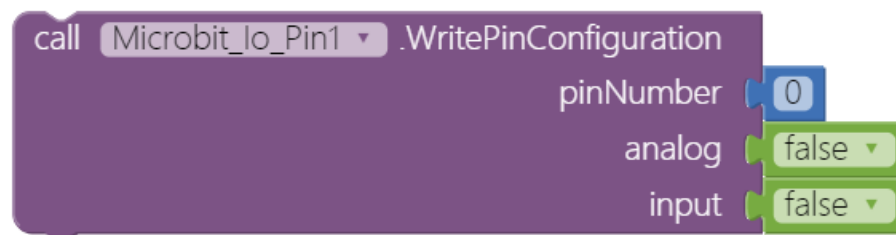
STEP 1: Request updates when connected

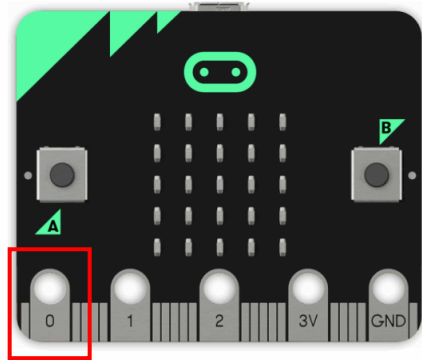
In the **BluetoothLE1.Connected** event, we show messages to tell user that we are connected with micro:bit and set micro:bit's pin status as "**digital input**". Since we are going to read the button status in this project. Check the **Microbit_Io_Pin.ConfigurePin** method, please specify the **pinNumber** to 0 (means P0 pin of micro:bit and set the **analog** field to **false** and the input field to **true**. And don't forget to set **Clock.TimerEnabled** to true, which means we are using the Clock component to read micro:bit pin status periodically.



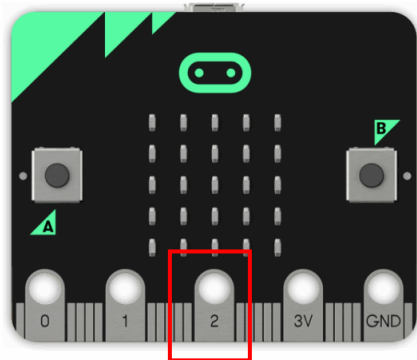
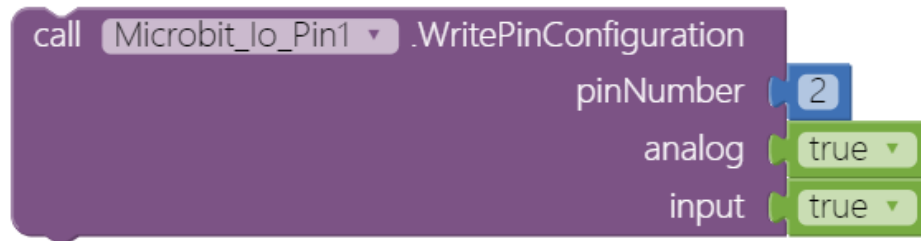
Let's let a look of **Microbit_Io_Pin.WritePinConfiguration** method, it has three parameters: **pinNumber** (pin index), **analog** (true to analog, false to digital) and **input** (true to input, false to output).

This is to set micro:bit's **P0** pin as **digital output**. You can connect component like LED to this pin. For micro:bit I/O pins detail please check this link: <http://microbit.org/guide/hardware/pins/>





And this is to set micro:bit's P2 pin as **analog input**. You can connect component like potentiometer to this pin.



STEP2: read button status periodically

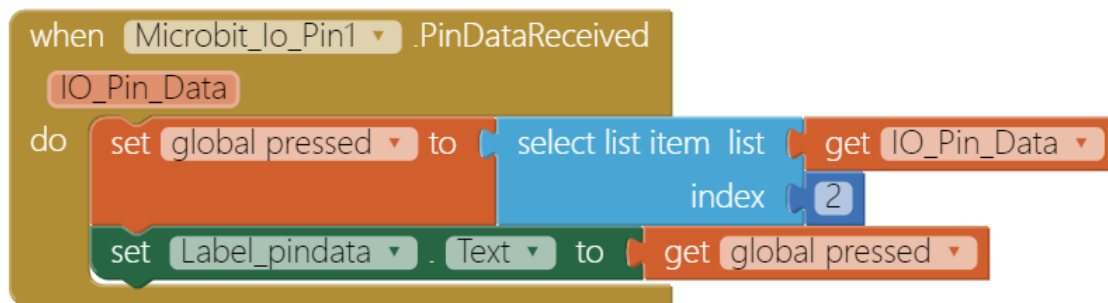
In **Clock.Timer** event, we call **Microbit_Io_Pin.ReadInputPinData** method to read a specified micro:bit pin data, which is **P0** in our case. Since we've set Clock's **TimerInterval** property to 100, this means we are reading micro:bit pin data every 100 millisecond (10 times per second).

*Note: you must configure the pin as **digital input** (in STEP1) to read its data correctly.*



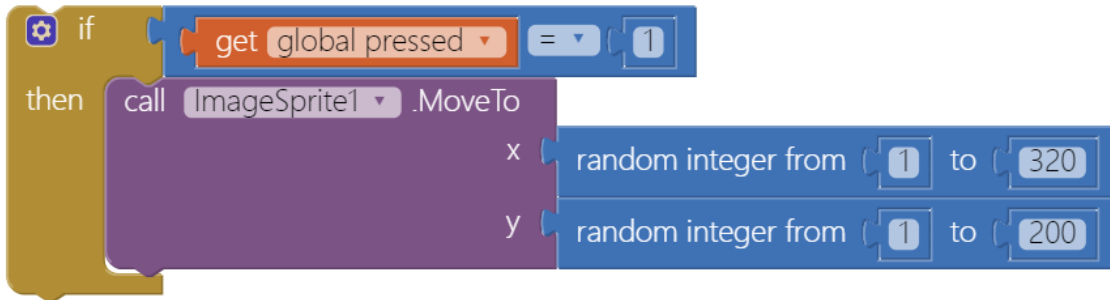
STEP3a:

Microbit_Io_Pin.PinDataReceived event will be called after the pin data is read successfully, and it will return a list (**pin index, pin data**) as a result. Here we use a **pressed** variable and use a **select list item** method to get pin data and show it on the label. **pressed** variable will be **1** when the button is pressed and will be **0** when the button is released.



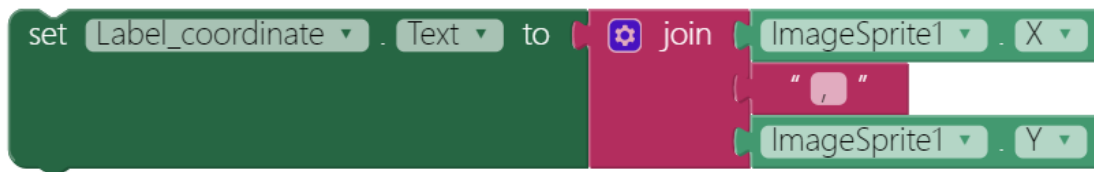
STEP3b: move ImageSprite to random location when button is pressed

To make our app more interactive, we will move the ImageSprite randomly every time the button is pressed (**ImageSprite.MoveTo** method).



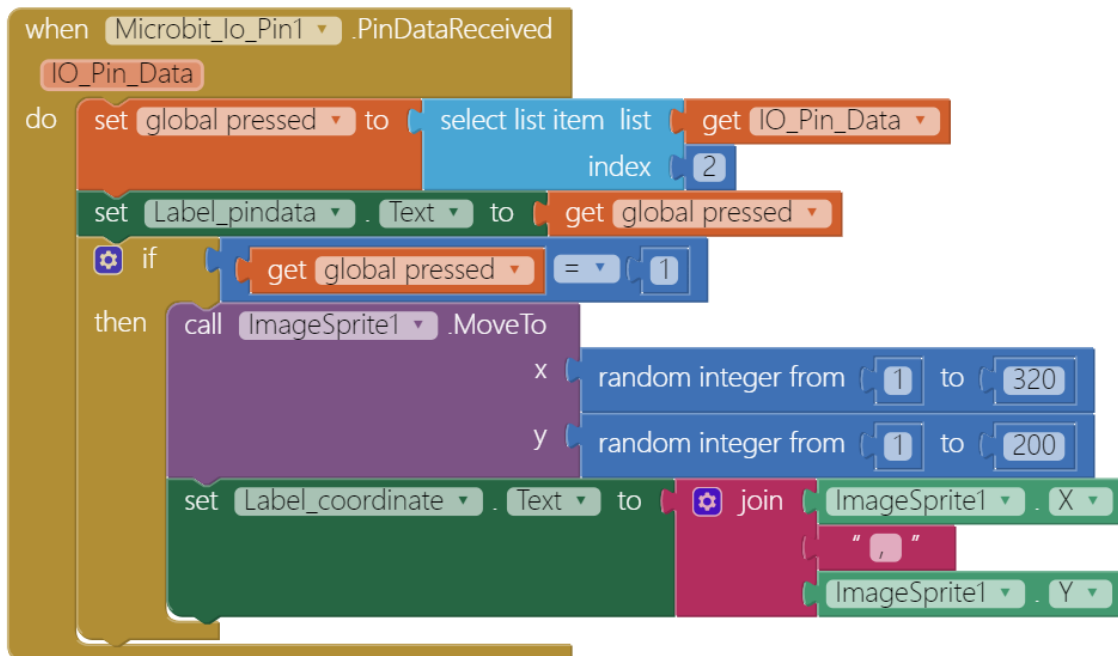
STEP3c: update ImageSprite location on label

We also show the latest ImageSprite position on a label.



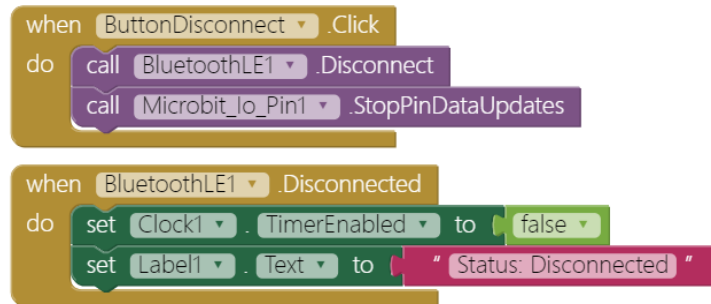
STEP4:

Finally, we put things from STEP3a to 3b together, finished as below:



STEP5: Disconnect from micro:bit

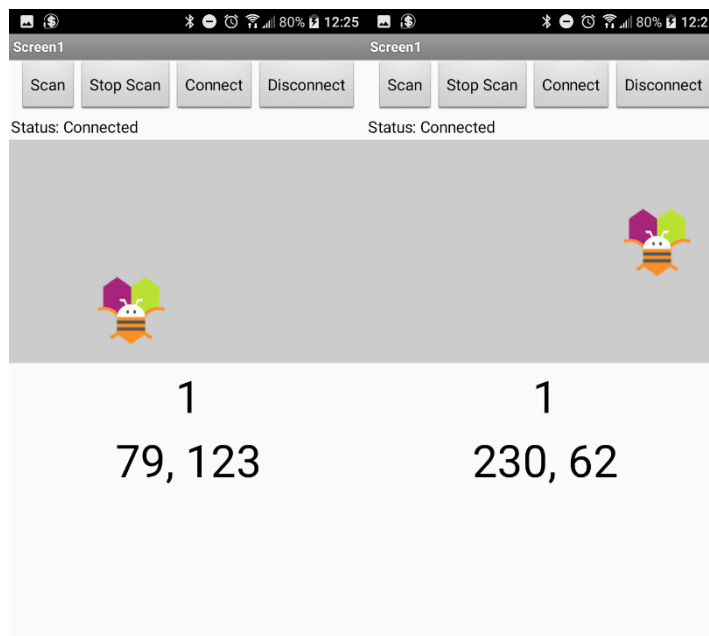
You can disconnect from micro:bit by clicking the **ButtonDisconnect**. This will reset the app to its initial state to wait for next connect request.



Tips

Your app should now be working! Make sure you have paired the Bluetooth on your Android device to your micro:bit. Then test it out by connecting your micro:bit device using the MIT AI2 Companion (if you haven't already) or installing it by .apk.

Press the button, you will see the imageSprite flying on the screen and there will be an **1/0** changing according to button status and the latest imageSprite position on the corresponding labels.



Brainstorming

1. Modify this project, play one sound effect when the button is pressed and play another sound effect when the button is released.
2. Add one more potentiometer to control the ImageSprite to move along the Y-axis.