App Inventor Getting Started

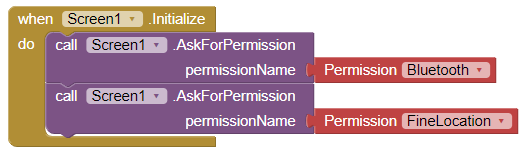
1. Visit <https://appinventor.mit.edu/>
2. Click on the "[Get Started](http://appinventor.mit.edu/explore/get-started)" button
   1. Click on "[Setup Instructions](http://appinventor.mit.edu/explore/ai2/setup)" and follow option 1 to install "MIT AI2 Companion" on your Android or iOS device.
   2. Go back to "Get Started" and click on "[Designer and Blocks Editor Overview](http://appinventor.mit.edu/explore/designer-blocks)"
      1. Designer is used to design the user interface
      2. Blocks Editor is used to implement the functionality
3. Go back to "Get Started" and click on "[Beginner Tutorials](http://appinventor.mit.edu/explore/ai2/beginner-videos)"
   1. Try out one or more tutorials:
      1. Click on "[Create Apps!](http://ai2.appinventor.mit.edu/)" and sign in to a Google account to launch the tool.
      2. Follow the instructions under Get Started > Setup Instructions > Option 1 [Instructions](http://appinventor.mit.edu/explore/ai2/setup-device-wifi) to connect your phone to App Inventor.
      3. Follow the tutorial instructions.
4. To learn how to package the app into an executable .apk file, go back to "Get Started" and click on "[Packaging and Sharing Apps](http://appinventor.mit.edu/explore/ai2/share)"

Using App Inventor with BLE

1. Scan and Connect

To learn how to install the BLE extension and create a basic BLE app that will scan/connect, visit: <http://iot.appinventor.mit.edu/assets/tutorials/MIT_App_Inventor_Basic_Connection.pdf>

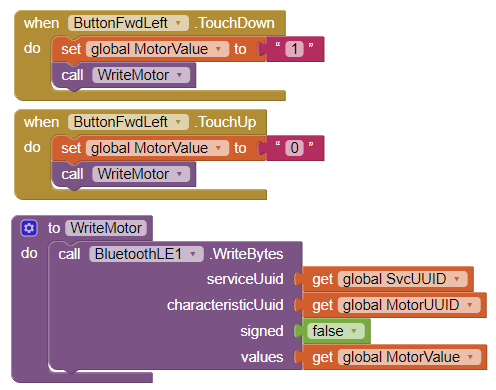
* 1. **You must use the latest version of the BLE extension. The link shown in the PDF file is not the latest.** You should use to go this site and download the AIX file for BluetoothLE:
     1. http://mit-cml.github.io/extensions/
  2. **You must add an additional block from this example for scanning to work** – BLE on Android 10 and above requires Bluetooth and fine location permissions so you must request them:



1. Writing a Characteristic

Once you are connected, here is an example of how you might write to a characteristic for button TouchDown and button TouchUp events using a single sub-routine to do the BLE write without response (WritBytes).

Note: The UUIDs for the service and characteristic were set using global variables. You can copy/paste the values from the Bluetooth® configurator in ModusToolbox™.



1. Reading a Characteristic

To read a characteristic, you need to do two things:

* 1. Initiate the read with an appropriate "call" block such as:

call BluetoothLE1.ReadBytes

call BluetoothLE1.ReadIntegers

call BluetoothLE1.ReadFloats

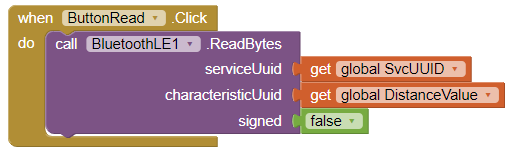
* 1. Add the corresponding "when" block which will be executed when values are received such as:

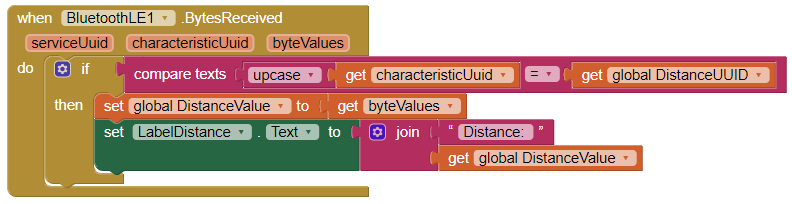
when BluetoothLE1.Bytes Received

when BluetoothLE1.Integers Received

when BluetoothLE1.Floats Received

You should use "if" constructs to check the UUID to make sure the value you received is the one you expected. You can even have multiple "if" constructs to handle multiple read values in a single block. Note that the UUID that comes back from the BytesRecieved function is lower case, so it must be converted if your variable is in upper case (which it will be if you copy/pasted the value from the Bluetooth® configurator.





1. Enabling Notifications

To enable notifications, you also need to do two things:

* 1. Register for notifications with an appropriate "call" block such as:

call BluetoothLE RegisterForBytes

call BluetoothLE1.RegisterForIntegers

call BluetoothLE1.RegisterForFloats

* 1. Add the corresponding "when" block which will be executed when values are received. This is identical to what is done for a manual read and in fact the same block will be executed either for data from an explicit read or from a notification:

when BluetoothLE1.Bytes Received

when BluetoothLE1.Integers Received

when BluetoothLE1.Floats Received

