Intro to App Inventor and Application with Bluetooth Connectivity

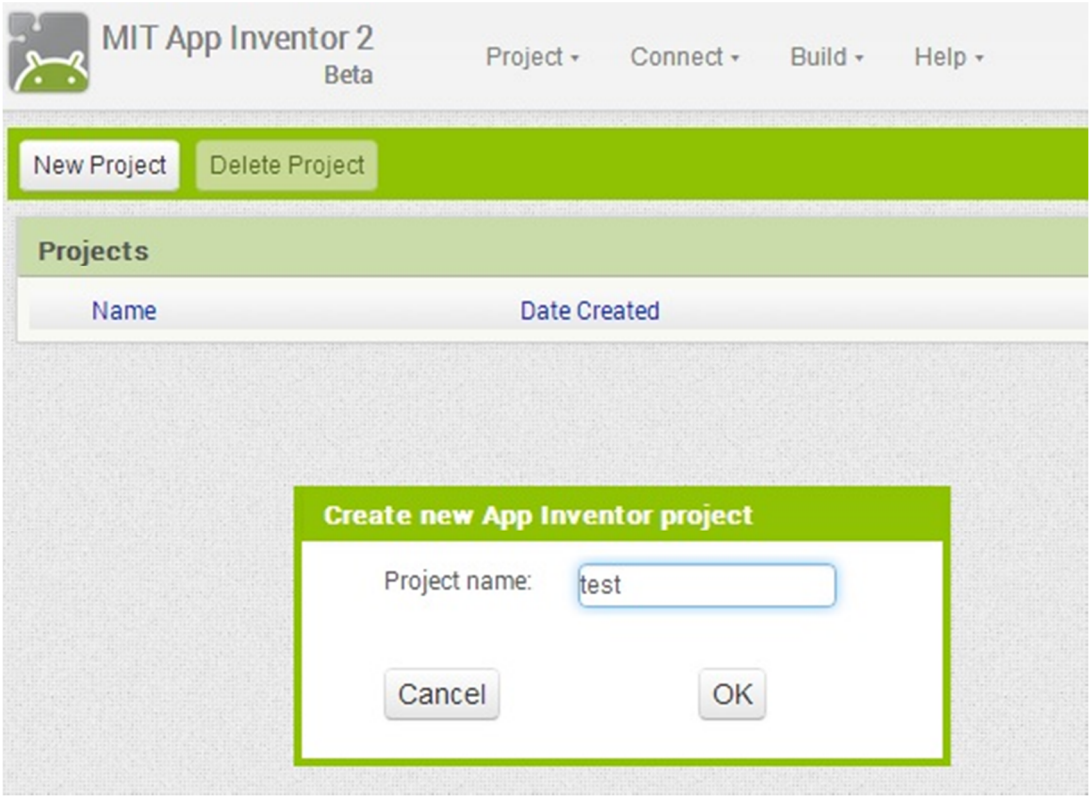
1. Introduction to the App Inventor Development Environment

App Inventor is a web-based online graphical mobile application development environment for Android devices, where you can create an application by simply drag and connect a series of function blocks.

To develop application using App Inventor, you can use one of the support browsers pointing to the following URL:

- http://ai2.appiventor.mit.edu/

You need a Google account to use App Inventor. After login to the system using a Google ID, from the App Inventor Designer menu, you can click on Start new project to start a new program, as shown below:

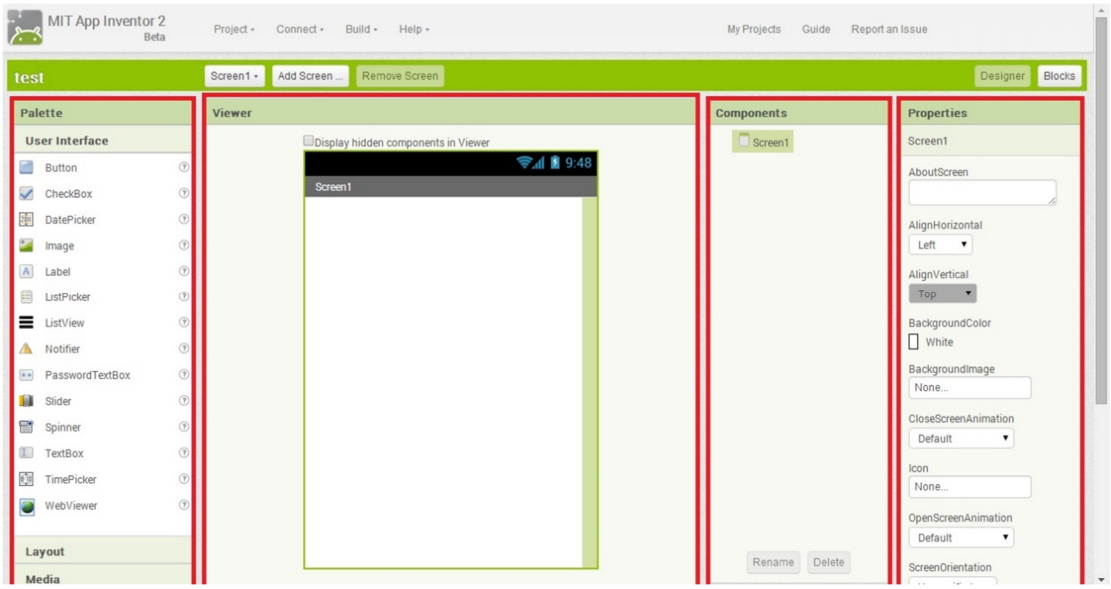


Enter project name and click OK to continue.

At this point, the App Inventor Designer is showing 4 separate sections, Palette, Viewer, Components and Properties.

http://appinventor.mit.edu/explore/designer-blocks.html

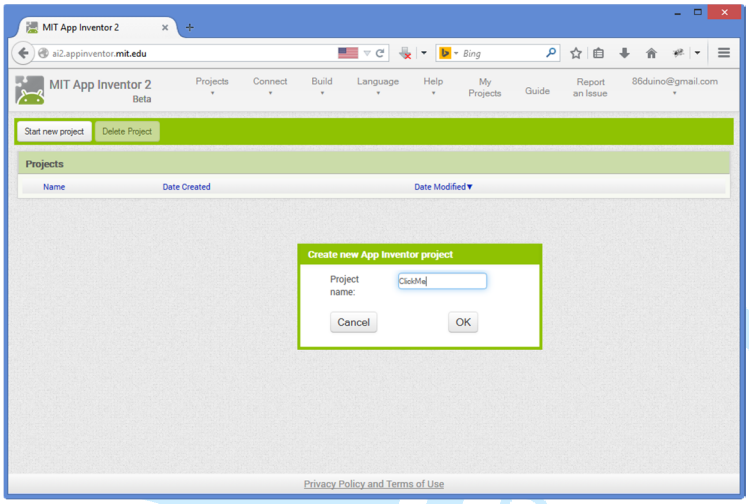
- **Palette**: This section contains different components which you can drag onto the Viewer to add them to your application. This is a familiar feature to the .NET developer.
- **Viewer**: This section provides a preview screen for your application where you can drag and drop components from the Palette section onto the screen and arrange the components to see how your app will look like.
- **Components**: This section lists all of the components that are added to your application. By clicking on a component, the selected component' s properties are shown on the Properties section.
- **Properties**: This section displays all of the properties associated with a selected component and provides the interface for you to edit and change the setting or value for each of these properties.



2. First sample app

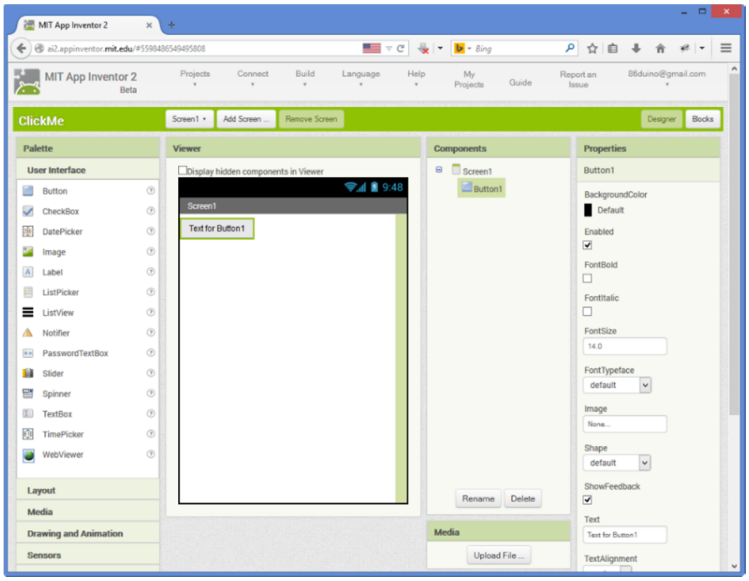
In this first exercise, we will go through the steps to create a simple application where you can click a button to change display character.

From the App Inventor menu, click on Start new project to create a new project and enter ClickMe as the project name, as shown below:

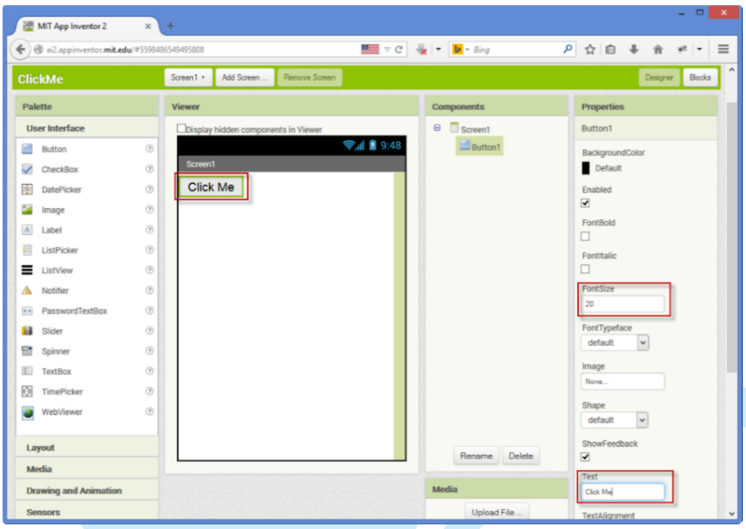


After clicking on OK to create the app, the App Inventory design and layout screen is shown. This is where you can add components and change the layout for the app.

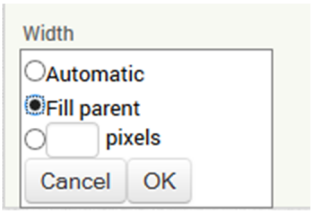
We need a button for the app. From the Palette section, click and drag the Button component onto the Viewer section to add the button control to the project as Button1, as shown below:



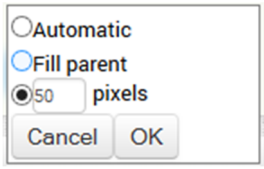
From the Components section, click and select Button1. At this point, the Properties section to the right is associated with Button1. From the Properties section, change FontSize to 20 and Text property for Button1 to "Click Me", as shown below:



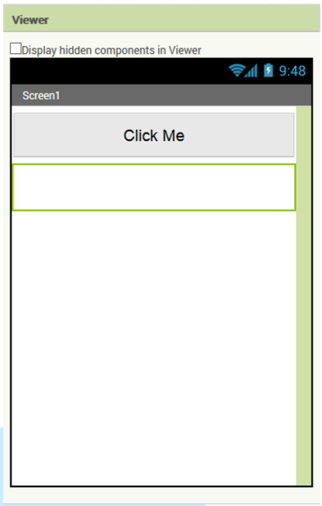
You can also change the button's width and height from the Properties pane. Click on the Width property and select the Fill parent option, as shown below:



Click on the Height property select the pixels option and enter 50 pixels for button height, as shown below:

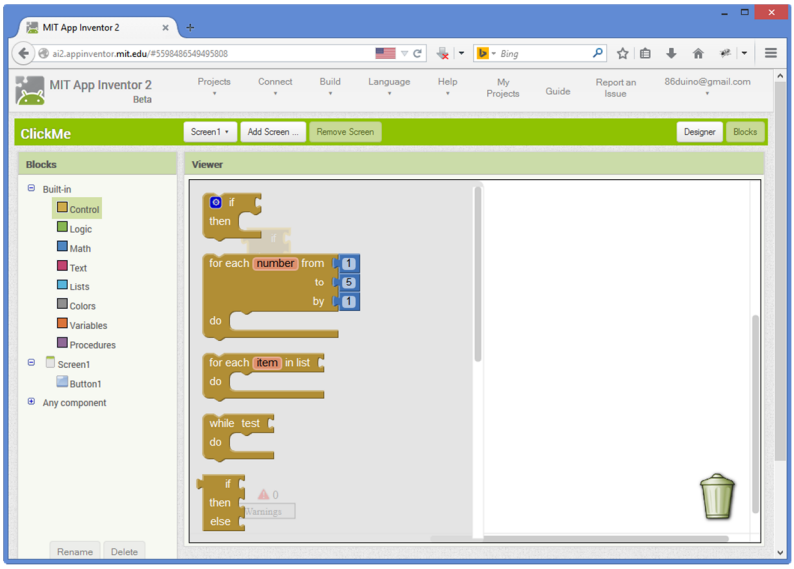


From the Palette section, click and drag the Label component onto the Viewer section to add the label control to the project as Label1. From the Components section, click and select Label1. From the Properties section, clear the entry in the Text properties, set the Width to fill parent and the Height to 50 pixels, as shown below:

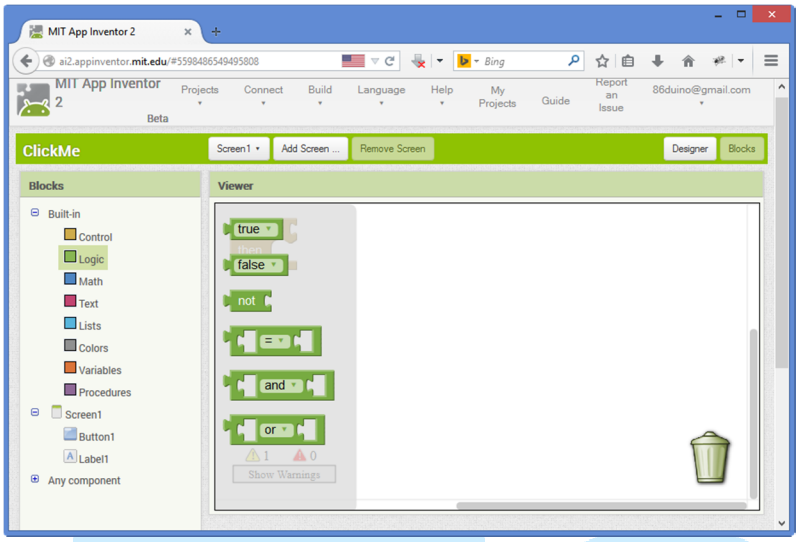


In the previous steps, we put together the app’ s layout in Designer mode and have not implemented any program logic. App Inventor has 2 different view modes, Designer and Blocks modes. In Designer mode, the Viewer section provides a graphical interface to view and design the app’ s layout. In Blocks mode, the Viewer section provides a graphical interface where you can construct the app’ s logic and function by dragging different component from the Blocks section.

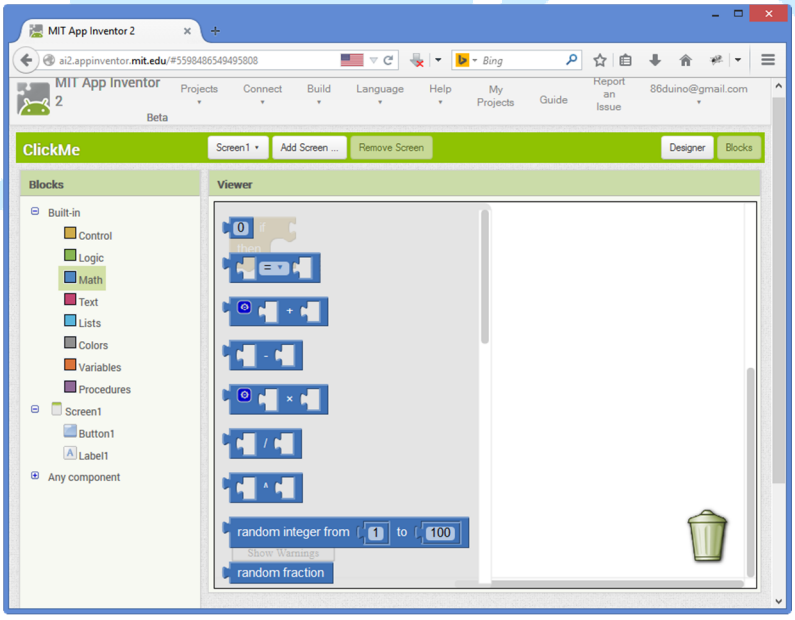
There are multiple group of controls with different functions and features. The Control group provides different conditional flow control (If-then, Do-While, for-each &etc.) logic blocks, as shown below:



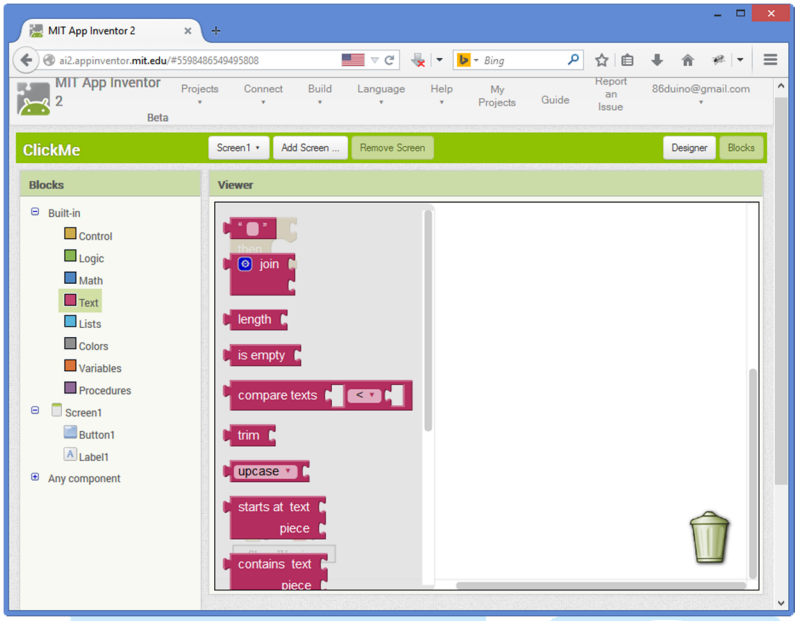
The Logic group provides the following:



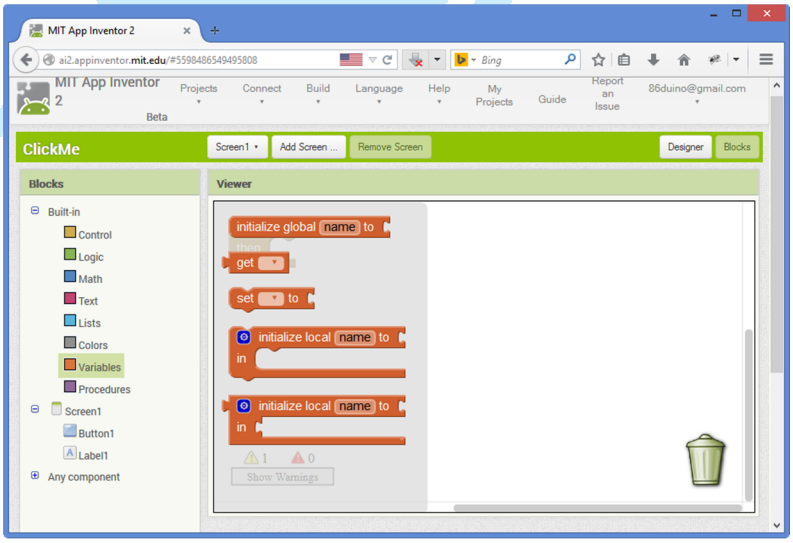
The Math group provides the following math function blocks:



The Text group provides the following text and string handling components:

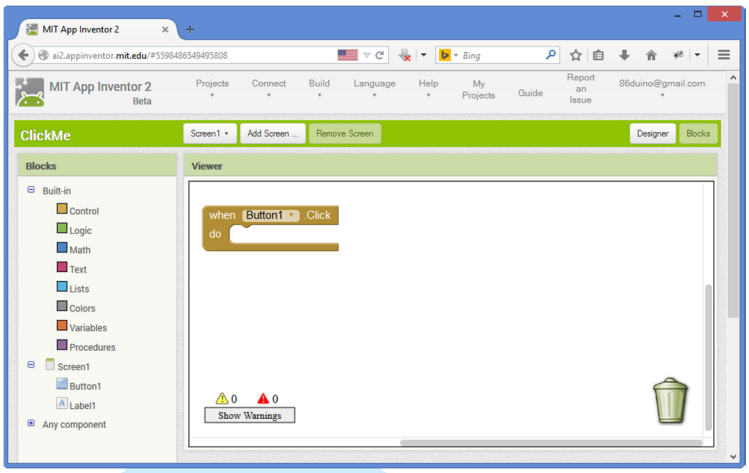


The Variables group provides function blocks that work with different variables needed for the app, as shown below:

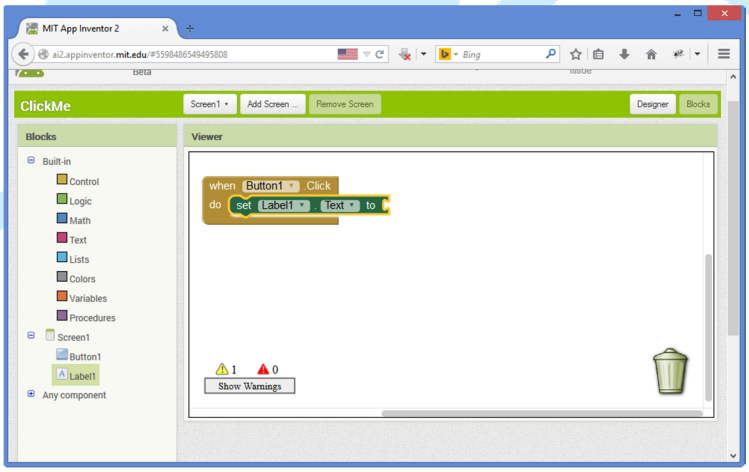


In the following section, we will go through the steps to add program logic to the app from the Blocks mode, adding program function blocks to the ClickMe app:

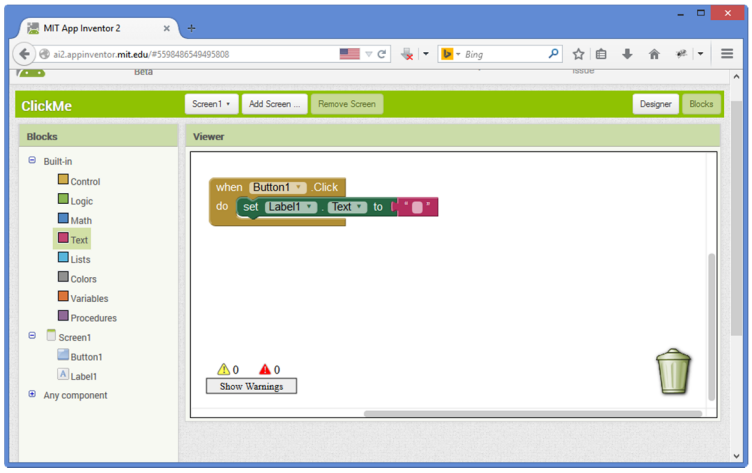
- From the Screen1\Button1 component group, click on the "when Button1.Click" component block and place it on the Viewer section, as shown below:



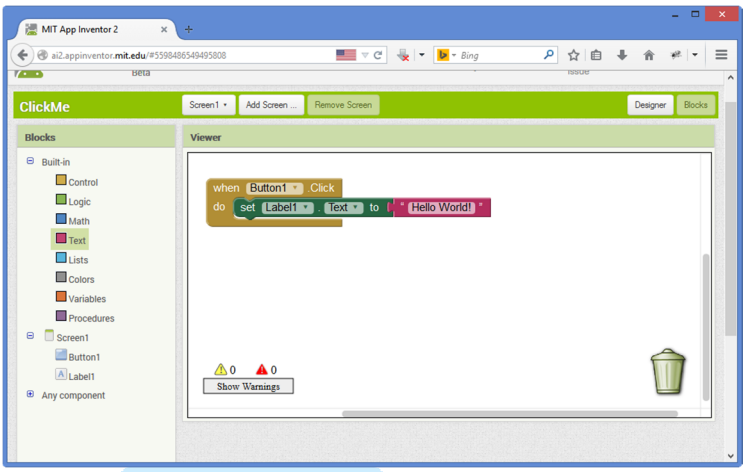
- From the Screen1\Label1 component group, click on the "set Label1.Text to" component block and place it on the Viewer section, as shown below:



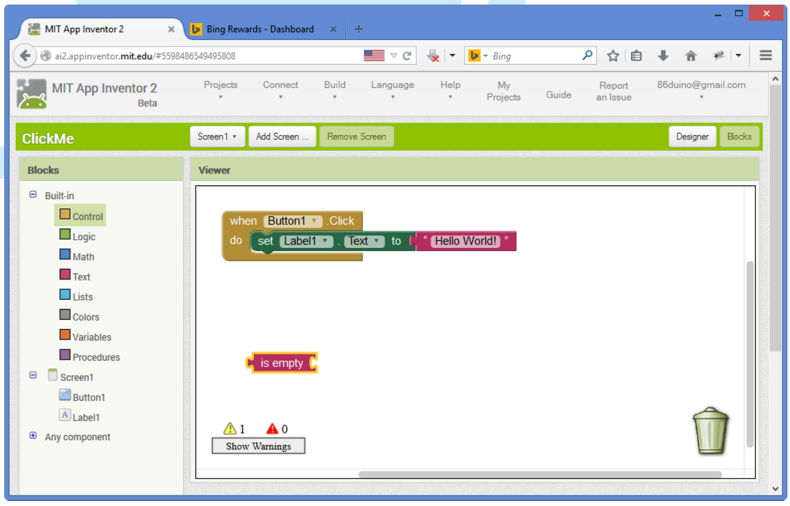
- From the Built-in\Text component group, click on the text string component block and place it on the Viewer section, as shown below:



- Click on the empty box between the double-quotes on the text string component and enter "Hello World!", as shown below:



Each of the program function block has different shape for the connection and receptor, used to control program flow. When one of these function blocks is incorrectly placed, warning and error are shown on the Viewer section, such as the following:



The "is empty" function block is attached as part of the program flow and cause the warning message shown next to the yellow triangle. To remove the "is empty" function block, you can click and drag the component to the trash can at the lower right corner of the screen. Or, simply click to select the component and press the delete key.

With only one button and program logic for a single click event, we cannot demonstrate interaction with the program effectively. Let's add a second button to clear Label1 text content.

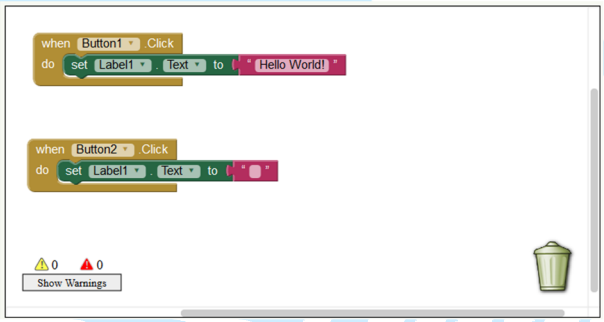
From Designer mode, drag a Button control from the Palette section to the Viewer and perform the following:

- Change the FontSize property to 20

- Change the Text property to Clear
- Change the Width property to Fill parent
- Change the Height property to 50 pixels

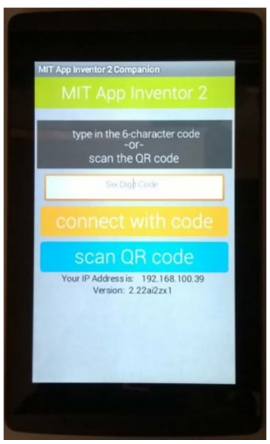
From the Blocks mode, add the following function blocks (as shown in figure below):

- "When Button2.click" from the Screen1\Button12 component group
- "Set Label1.Text" from the Screen1\Label1 component group
- Text string component from the Built-in\Text component group

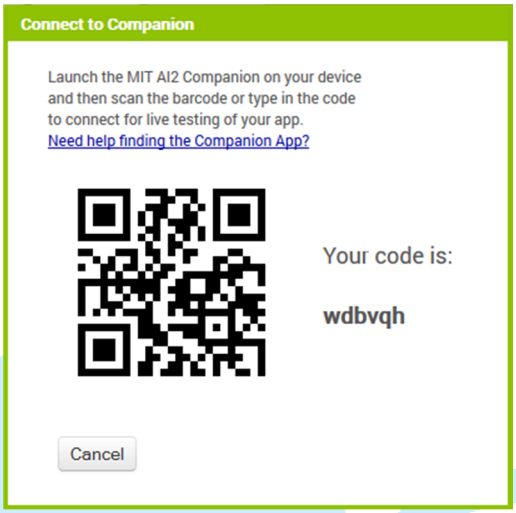


At this point, we have all of the intended function for the app. Before building and testing the app, we need to establish connectivity to a device or an emulator.

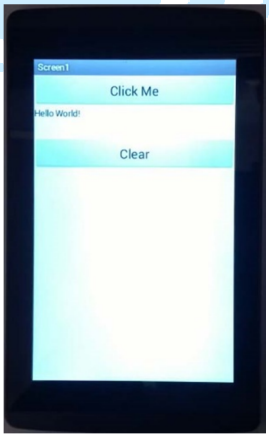
For this exercise, we will use a real device. To use a real device, we need to install the "MIT AI2 Companion" app from the app store. Install and launch the app on the target device. After the MIT AI2 Companion app is launched, you have the option to enter a six digit code or use the scan QR code option to connect to App Inventor, as shown below:



From the App Inventor' s Connect menu, click on AI Companion to bring up the following screen:



From the target device, you can enter the six digit code, or scan the QR code to establish connectivity to the App Inventor. Once connected, the app will display on the device, as shown below:

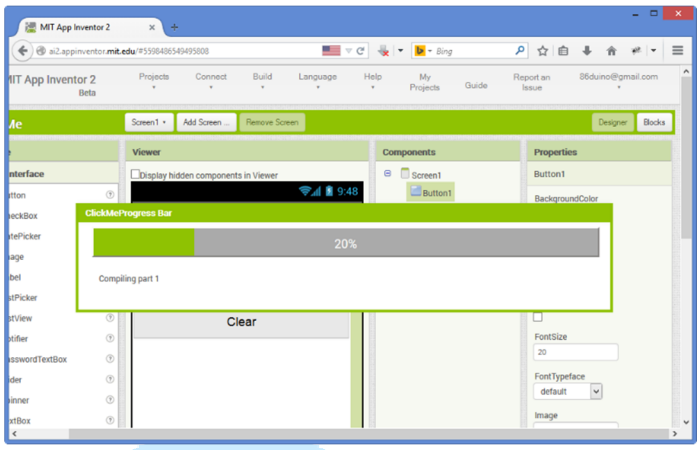


With connection to the App Inventor established, the changes you make to the app will reflect on the device in real time.

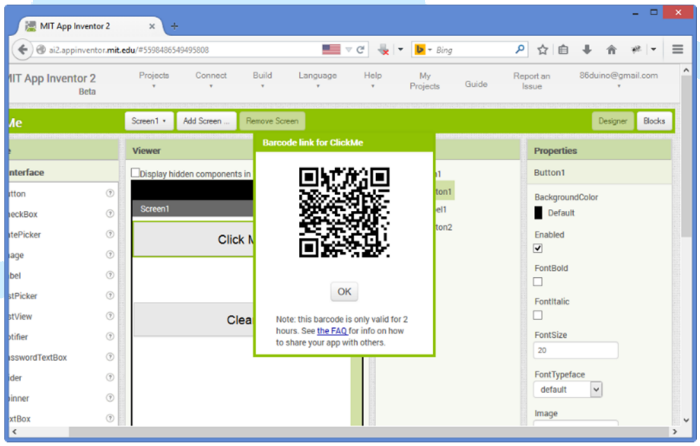
While the MIT AI2 Companion app can connect to App Inventor for testing and debugging, the app is not installed to the device. Once disconnected, the app will not remain on the device.

Let' s go through the following steps to install the app to the device:

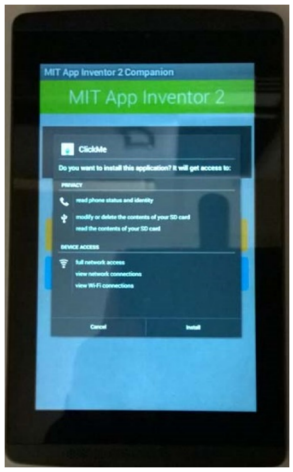
- From the App Inventor Build menu, click on App (provide QR code for .apk) to build the app. App Inventor shows the following progress as it build the app.



- After the build is done, a QR code is provided for the MIT AI2 Companion to install the app, as shown below:

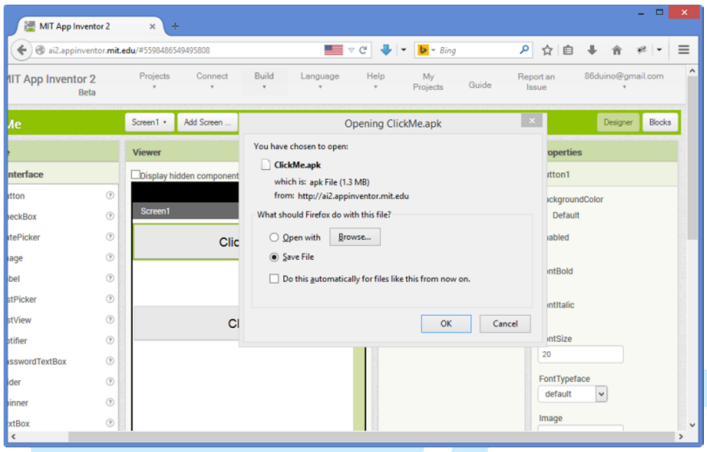


- After scanning the QR code using MIT AI2 Companion, the following screen is shown on the device, asking for permission to install the app.



Another option to install the app is to compile and download the .apk installation file, download and copy the file to the target device' s local storage and install the app manually:

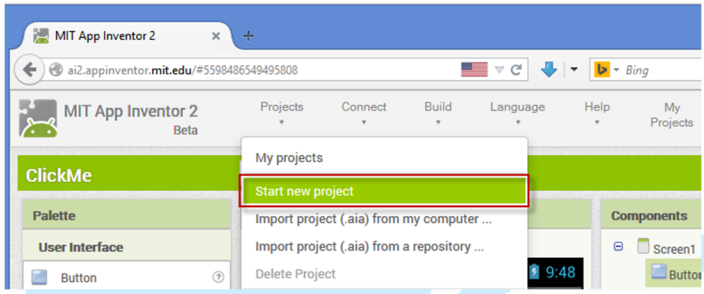
- From the App Inventor Build menu, and select the “App (save .apk to my computer)” option to build the app.
- After the build process is completed, App Inventor provides the option to download the generated .apk file, as shown below:

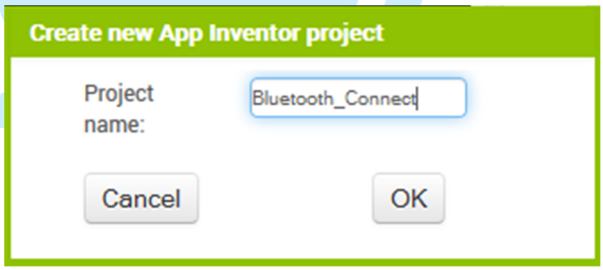


3. Second sample app

In this second exercise, let’ s create an app that connect to the EduCake using Bluetooth.

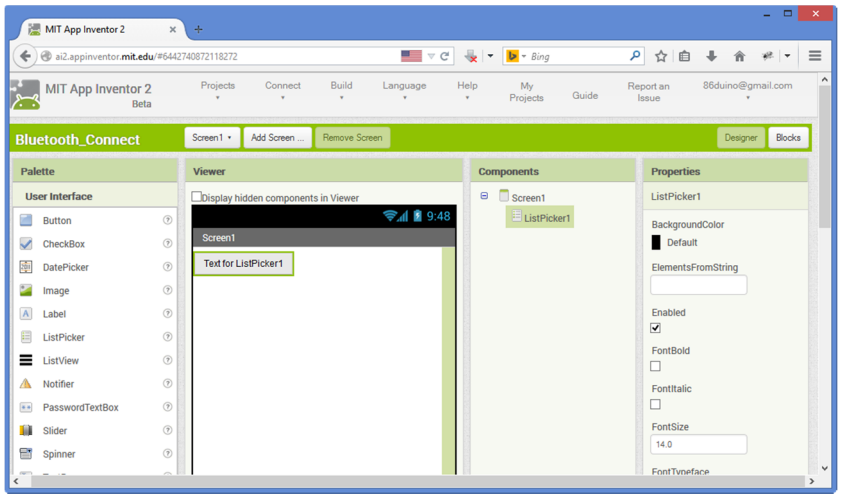
From App Inventory Project menu, click on Start new project, as shown below, to bring up the new project wizard:





Enter “Bluetooth_Connect” as project name and click OK to continue.

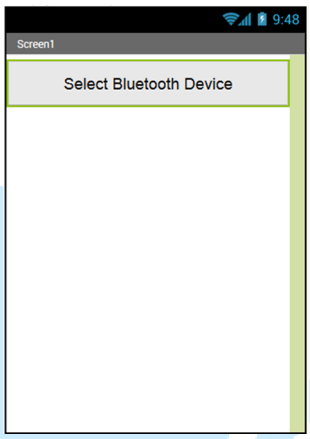
From the Palette section, select and drag the ListPicker component to the Viewer section, as shown below:



Select ListPicker1 in the Components section and change the associated properties as follow:

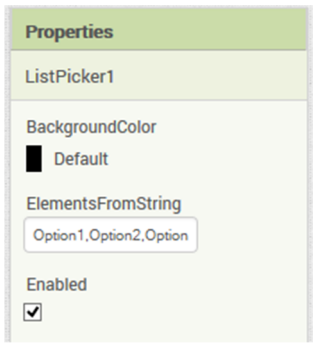
- Change the Text property to "Select Bluetooth Device" and FontSize to 20.
- Change the Width property to "Fill parent" and Height to 50 pixels

The screen in the Viewer section should looks similar to the following:



The ListPicker component provide similar function as Button that function like menu options, where you can click on one of the selection to get to the specified option, such as the following:

There are different methods to configure the ListPicker. One of the method is the ElementsFromString Properties, as shown below:

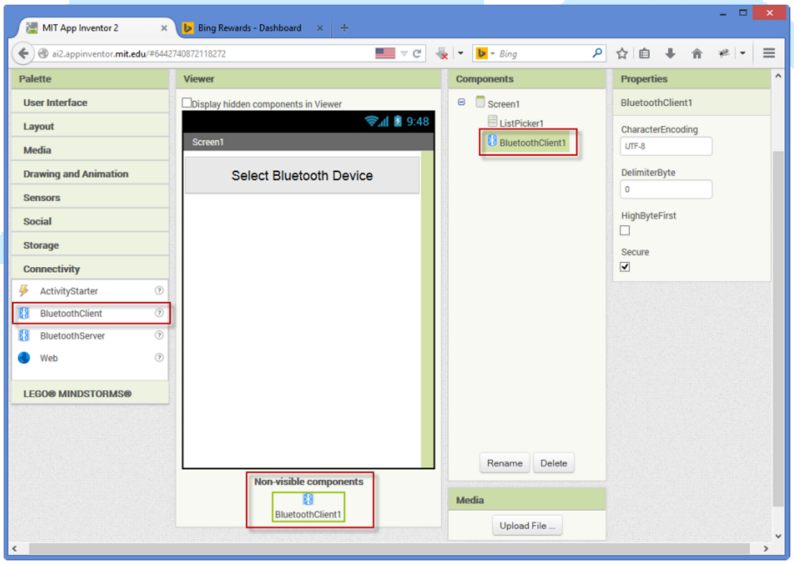


The other method is programmatically within the app which provide more flexibility, as shown below:

Here is another method:

Let’ s continue with the app. From the Connectivity component group in the Palette section, select and drag the BluetoothClient component to the Viewer.

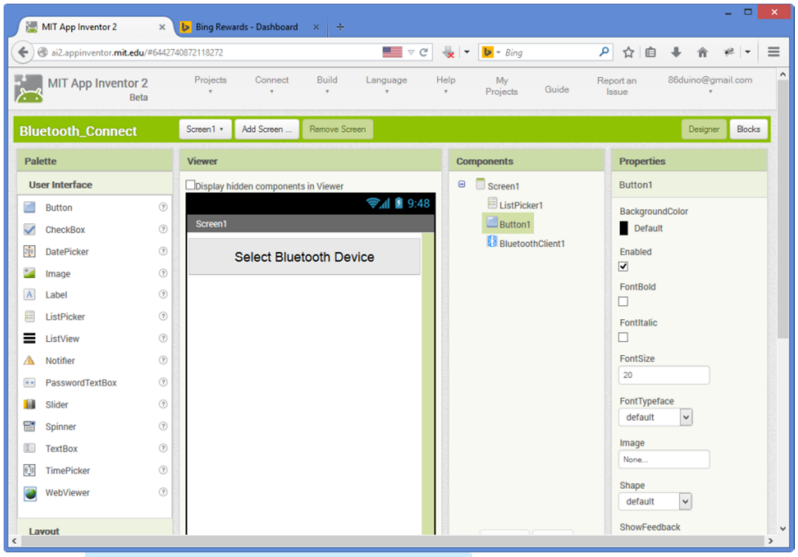
Since the BluetoothClient component does not have user interface, it’ s a non-visible component and not shown on the Viewer. You can see the BluetoothClient1 component is added in the Components section, as shown below:



From the User Interface component group in the Palette section, select and drag a Button control onto the Viewer and change the button properties as follow:

- Change the Text property to “Click Me” and FontSize to 20
- Change the Width property to “Fill parent” and Height to 50 pixels
- Change the Visible property to hidden

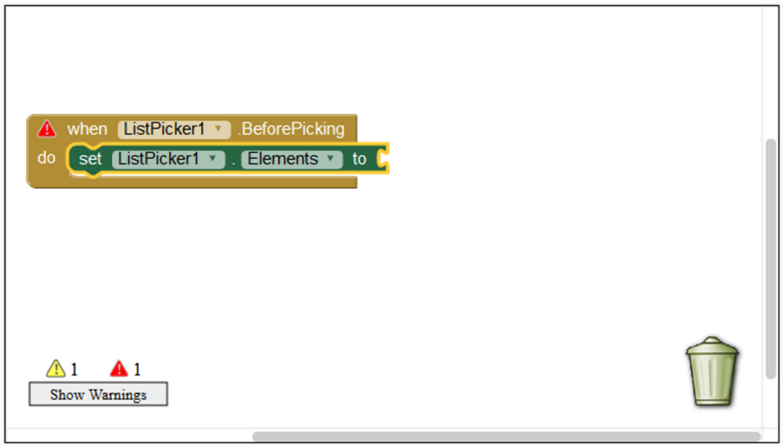
At this point, the App Inventor screen should looks like the following:



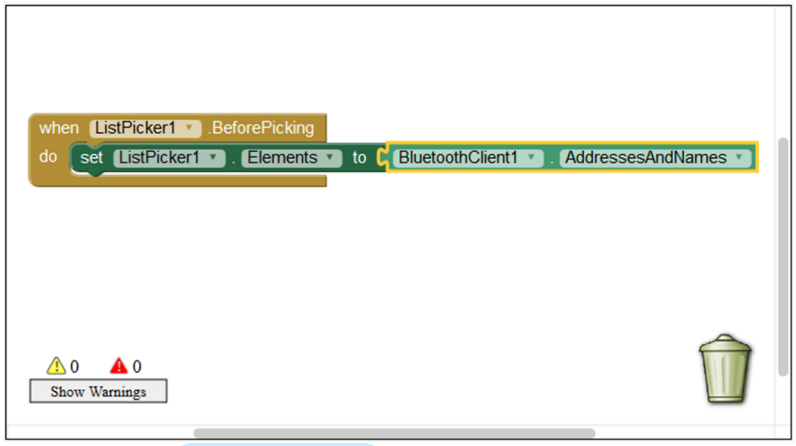
Next, click on the Blocks button (from the App Inventor menu) to switch to Blocks mode and add programming logic to the app.

From the Screen1\ListPicker1 component group, select and add the following components:

- "when ListPicker1.BeforePicking"
- "set ListPicker1.Elements"

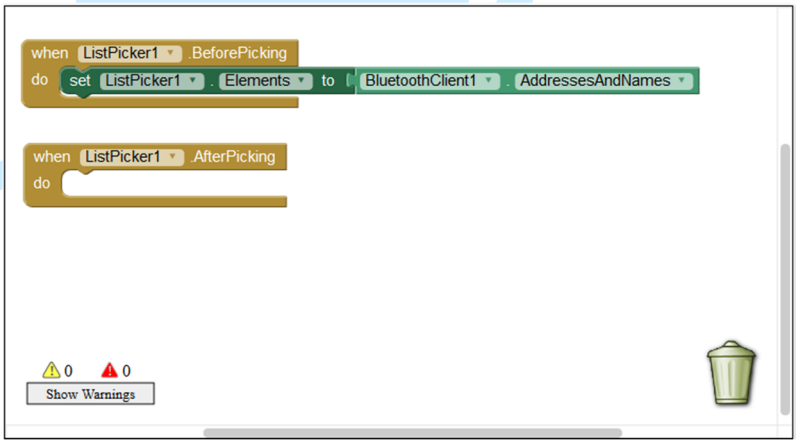


From the Screen1\BluetooothClient1 component group, select and add the "BluetoothClient1.AddressAndNames" component, as shown below:



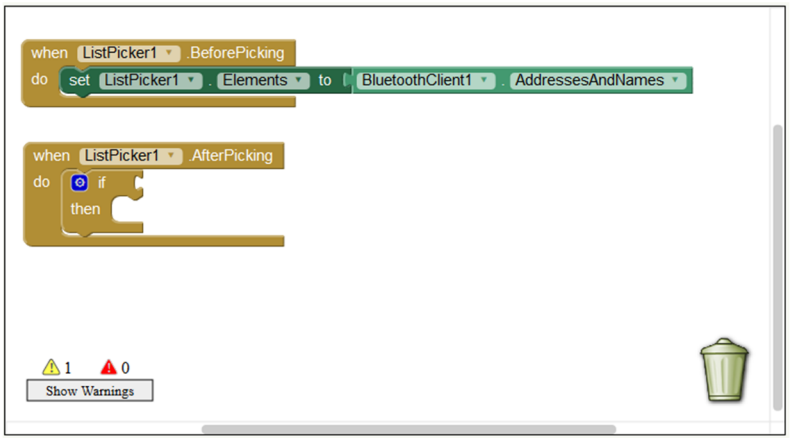
The above group of components provide the function to retrieve the list Bluetooth devices paired with the Android device.

Continue and add the "when ListPicker1.AfterPicking" component from the Screen1\ListPicker1 component group to the Viewer section:

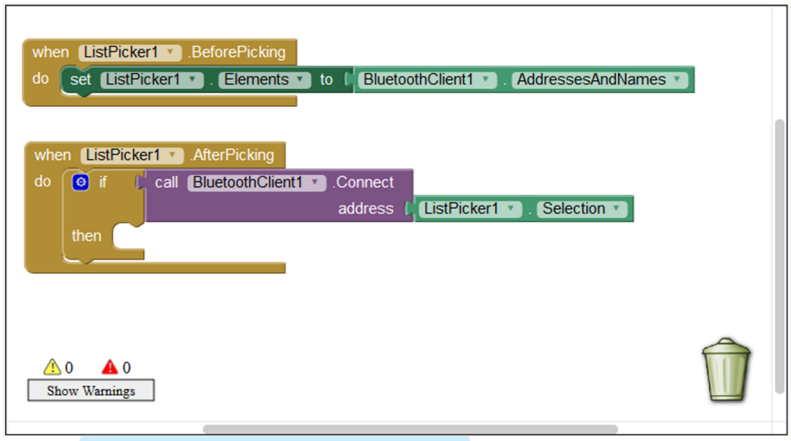


The ListPicker1.AfterPicking component is an event handler after an item is selected.

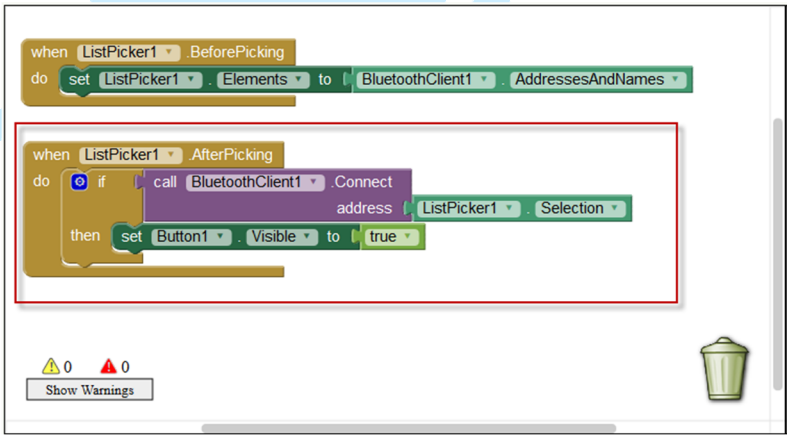
From the Built-in\Control group, select and add the "If then" component, a conditional handler, to the Viewer section:



Next, select and add the "call BluetoothClient1.Connect address" and "ListPicker1.Selection" components and link to the "if" condition:

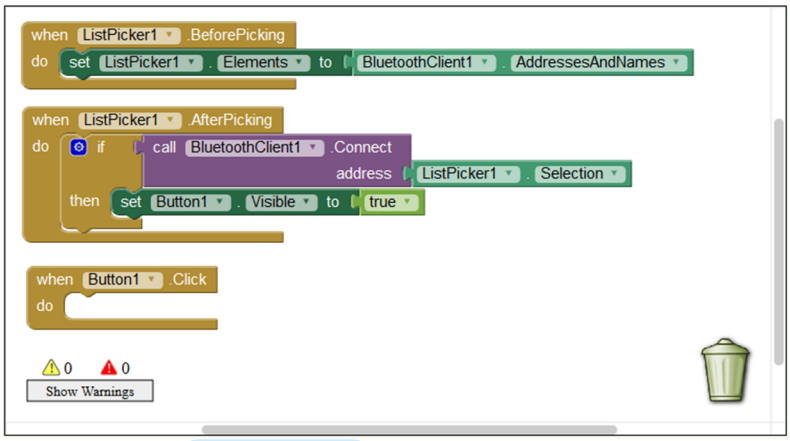


Next, add "Button1.Visible" component (from the Screen1\Button1 component group) and "true" component (from the Built-in\Logic component group) and link to the "then" condition:

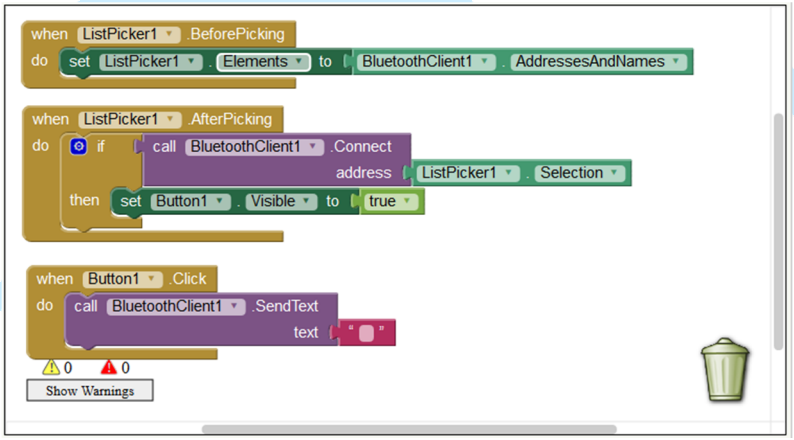


The 2nd group of components, highlighted within the red rectangular frame, is part of an event handler to change the "Click Me" button (Button1) from hidden to visible after a Bluetooth device is selected from the list.

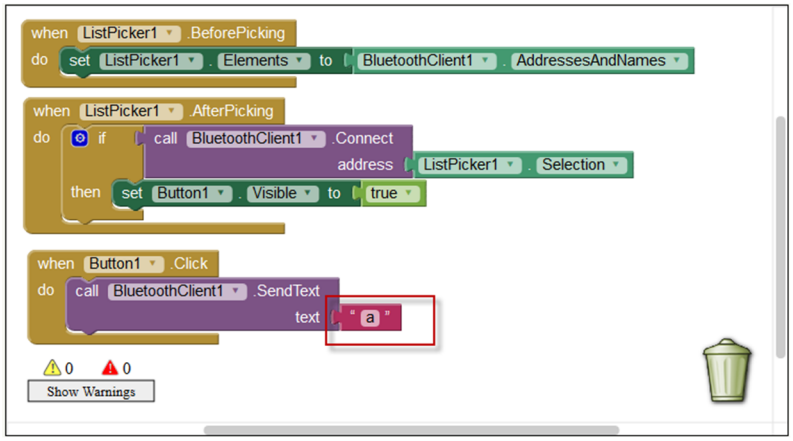
Next, add the "Button1.Click" component (event handler for the "Click Me" button) to the app:



Add the "BluetoothClient1.SendText" and the blank text entry component to the app:

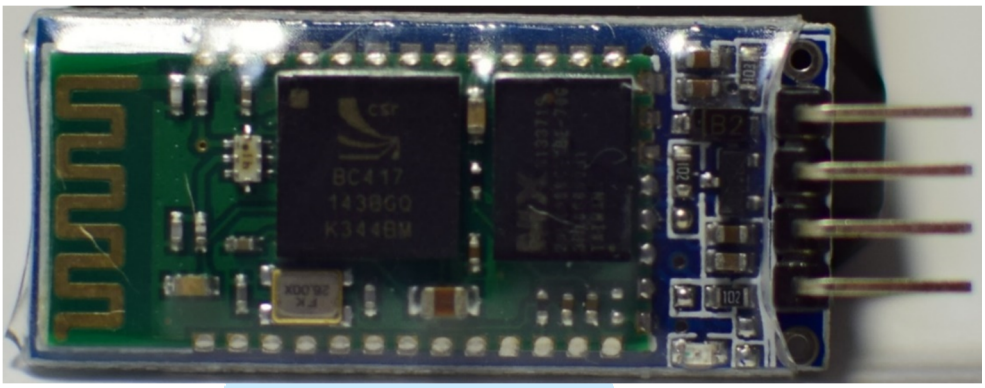


From the Viewer section, click on the black text entry component and enter the character "a":



The last group of function block provide the program logic to send the text character "a", via the BluetoothClient1 connection to the connected Bluetooth device.

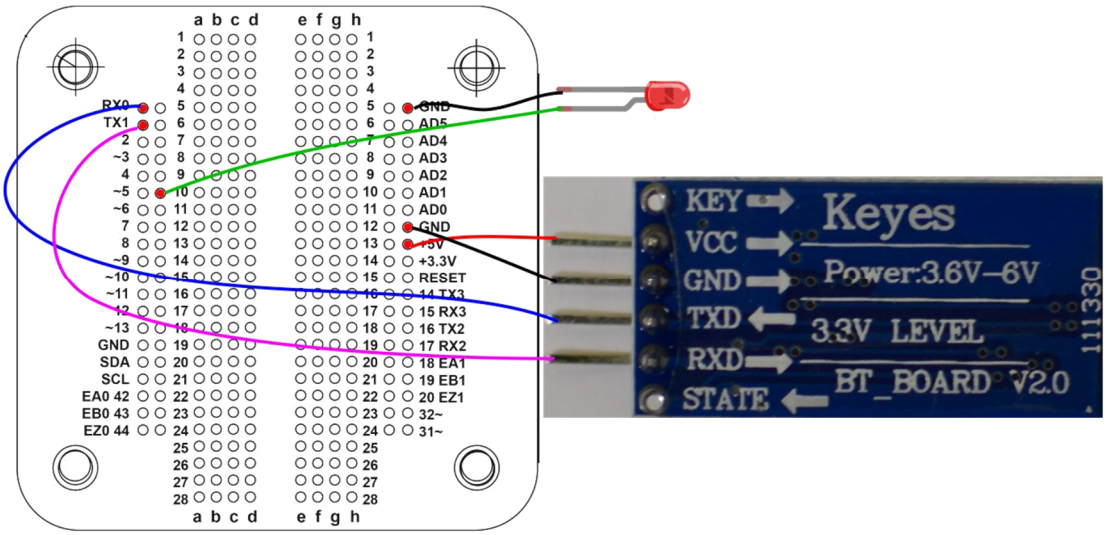
Next, let’ s take a look at the Bluetooth module used with the EduCake. For this exercise, we are using one of the popular Bluetooth module, HC-06, as shown below:



HC-06 module top view

HC-06 module back view

For the exercise in this section, a HC-06 Bluetooth module and LED are attached to the EduCake, as shown below:



From the 86Duino IDE, enter the following code to configure the HC-06 module:

```
void setup()
{
  Serial.begin(9600);
  Serial.println("Enter AT commands:");
  Serial1.begin(9600);  // configure baud rate to communicate
                        // with the HC-06 module
}
void loop()
{
  // Read data from HC-06 and transmit data to the PC
  if (Serial1.available())
    Serial.write(Serial1.read());
  // Relay data from the PC to the HC-06 module
  if (Serial.available())
    Serial1.write(Serial.read());
}
```

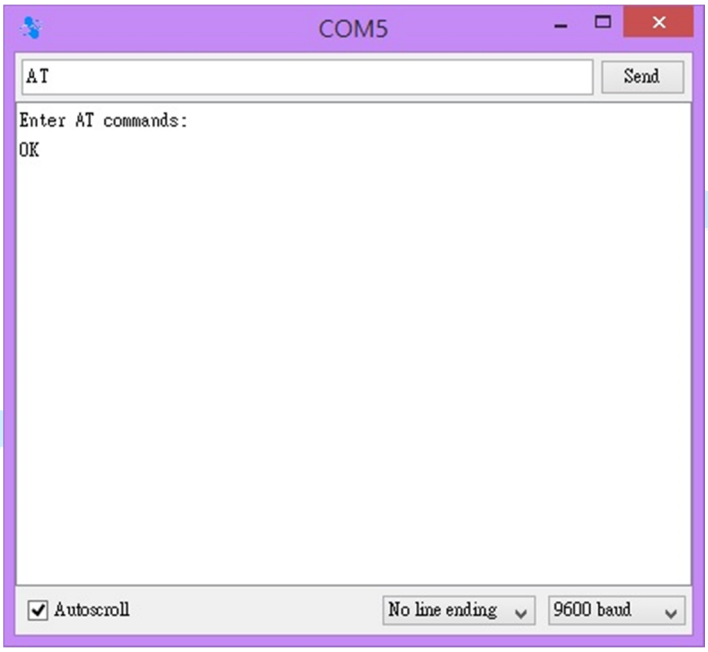
With the above code, communication link for the 86Duino IDE' s Serial Monitor can communicate with the Bluetooth interface, as shown below:

After power on and prior to establishing connectivity, the LED on the HC-06 module is blinking and remain in AT command mode, where we can change the module' s device-name, password, transmission speed (baud rate) and etc., using the following AT command:

- AT: Check to see whether the module is functioning, an「OK」respond indicate the module is functioning as expected.
- AT+NAMEaaa: Change the device name to「aaa」
- AT+PIN1234 : Change the pairing password to「1234」
- AT+VERSION : Request the module' s version information
- AT+BAUD1 : Change the baud rate to 1200
- AT+BAUD2 : Change the baud rate to 2400
- AT+BAUD3 : Change the baud rate to 4800
- AT+BAUD4 : Change the baud rate to 9600
- AT+BAUD5 : Change the baud rate to 19200
- AT+BAUD6 : Change the baud rate to 38400
- AT+BAUD7 : Change the baud rate to 57600

The first 3 AT command in the above list are commonly used.

- To change the module' s device name to "abc", enter "AT + NAMEabc". The module respond with the message "OKsetname" to indicate success.
- To change password to 1234, enter "AT + PIN1234". The module respond with the message "OKsetPIN" to indicate success.



While it' s not necessary to reboot the module after changing the password, you need to turn off power and restart the device, in order for device name change to take effect.

With the app created earlier for this 2nd exercise and the codes to configure the HC-06 module, we can move on to the next step to control an LED via Bluetooth, with the following codes:

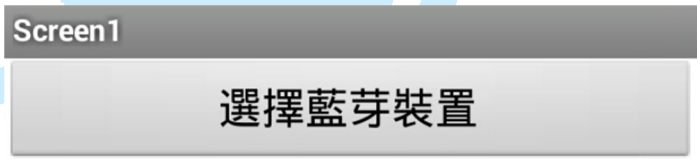
```
char ch;
int LED=0; // Initialize LED status variable
void setup()
{
   Serial.begin(9600); // Configure dev machine serial port
                                    // baud rate
```

```
Serial1.begin(9600); // Configure baud rate for Bluetooth
                                    // connection
   pinMode(5, OUTPUT); // Configure pin #5 as output
}

void loop()
{
   if (Serial1.available()) // check for available data
   {
      ch = Serial1.read(); // when data is present, read 1 byte
      if (ch == 'a' ) // Check whether received data is   “a”
      {
         if (LED==LOW) // check whether pin 5 is low
         {
            digitalWrite(5, HIGH); // set pin 5 to high
            LED=HIGH; // change LED status variable to high
         }
         else
         {
            digitalWrite(5, LOW);
            LED=LOW;
         }
      }
   }
```

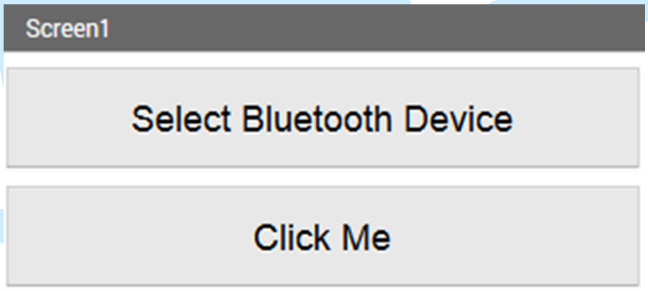
With the above codes in place, let’ s direct our focus to the Android device. First, make sure the Android device is paired with the HC-06 Bluetooth module, as shown below:

Next, launch the Bluetooth_Connect app and click on Select Bluetooth Device, as shown below:



After clicking on “Select Bluetooth Device”, the app launch a new screen to show the paired HC-06 module and other Bluetooth devices, as shown below:

Once connection to the HC-06 module is established, the "Click Me" button will show, as shown below:

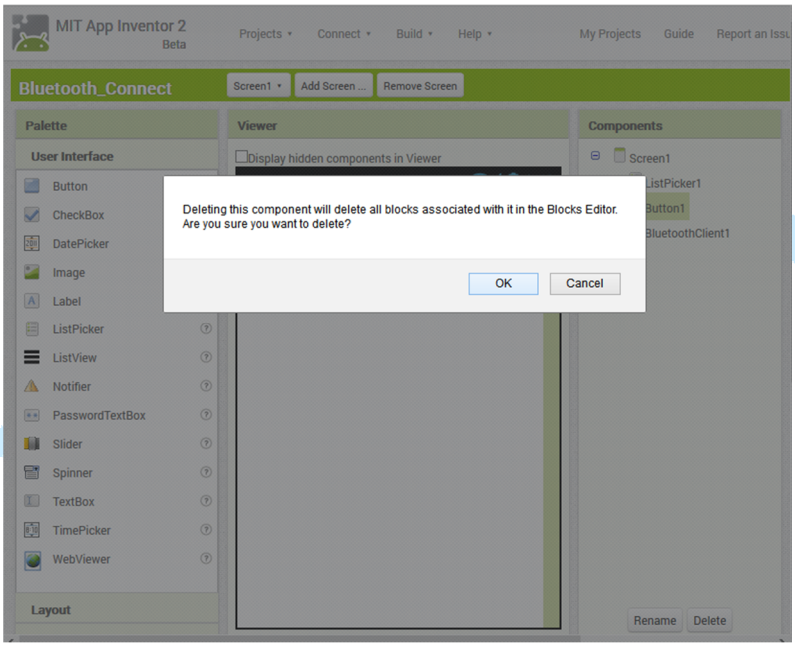


When you click on the "Click Me" button, the LED on the EduCake should light up. Clicking on the "Click Me" button again to turn off the LED.

4. Third sample app

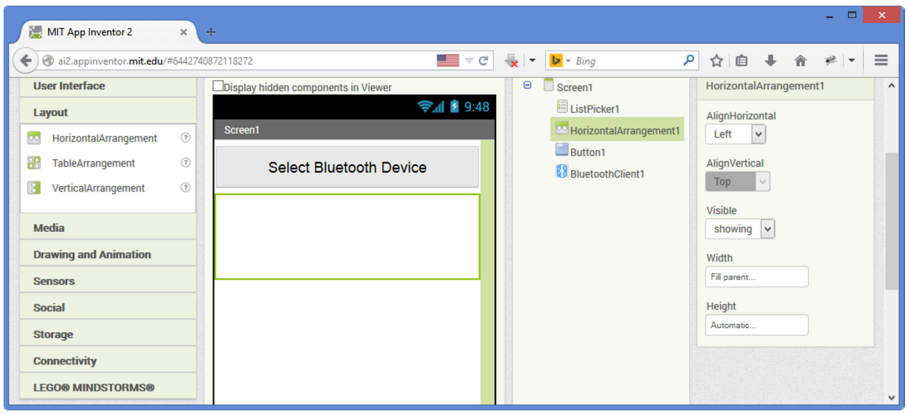
Continue from the second sample app, in this third exercise, we will add more function to the app. If you like to save the app prior to making changes, from the Project menu select the Save project as … option to save the app in a different name.

Continue from the second sample app, delete Button1 from the Components section, as shown below:

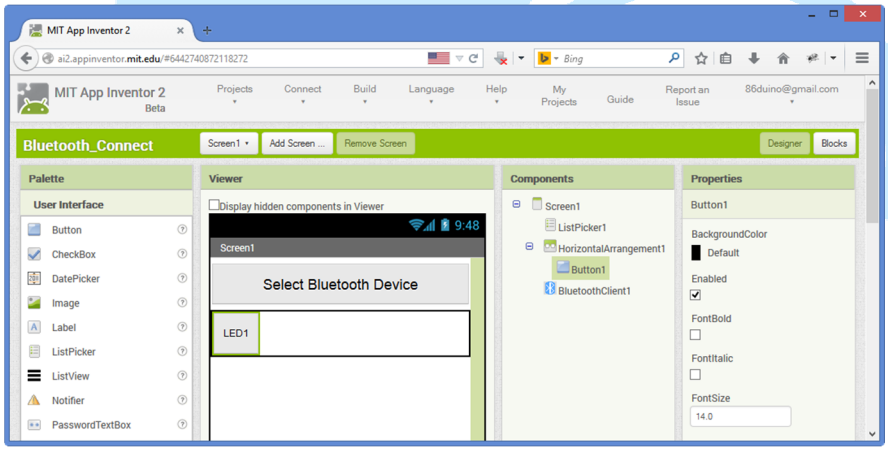


The sample app in the previous section control one LED. In this exercise, we will extend the app to control 5 LEDs. In addition, we will add 3 additional push button to the circuitry.

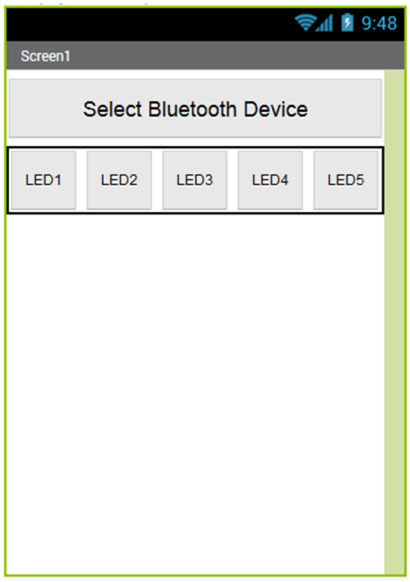
We need to place 5 button controls, align horizontally, to control the 5 LEDs. To accomplish this, we need to use the HorizontalArrangement layout component. From the Palette section, in the Layout group, select and drag the HorizontalArrangement component onto the Viewer and change the Width property to "Fill parent", as shown below:



From the Palette section, select and drag a button control onto the Viewer section, place the button control inside the HorizontalArrangement component, change the Text property to "LED1" and Height to 50 pixels, as shown below:

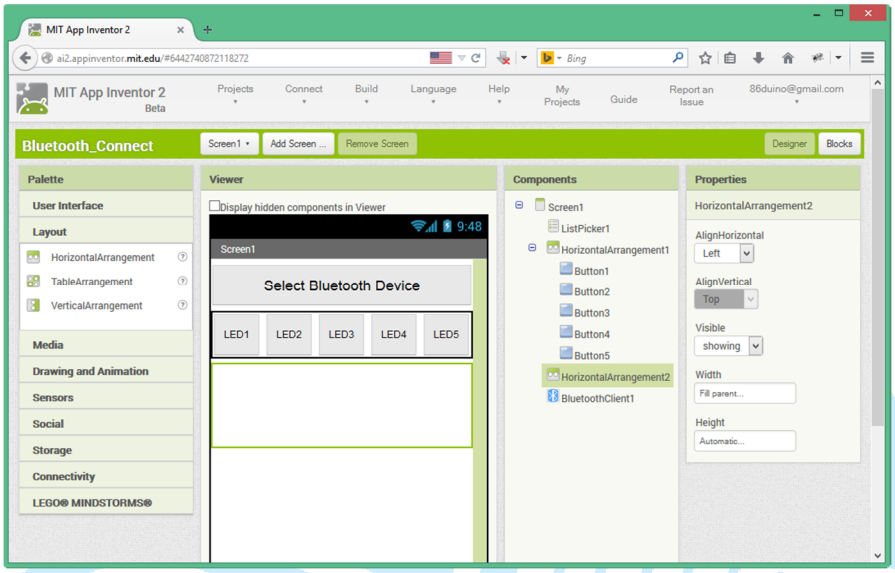


Repeat the same process to add 4 more button controls with Text property, LED2, LED3, LED4 and LED5, as shown below:

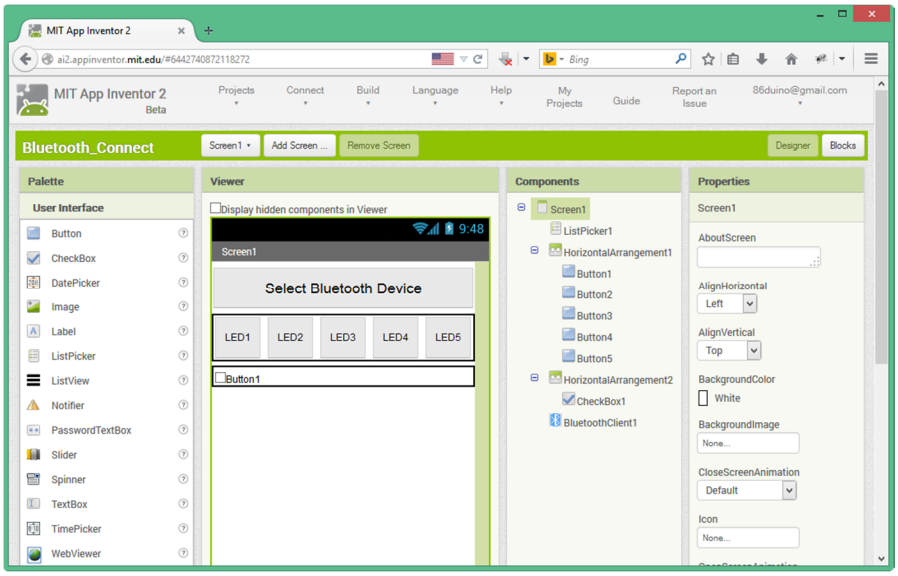


Depending on the device you are targeting, the 5 LED button controls may not evenly align on the display. To fix this, you can change the width property for all 5 LED button control to "Fill parent".

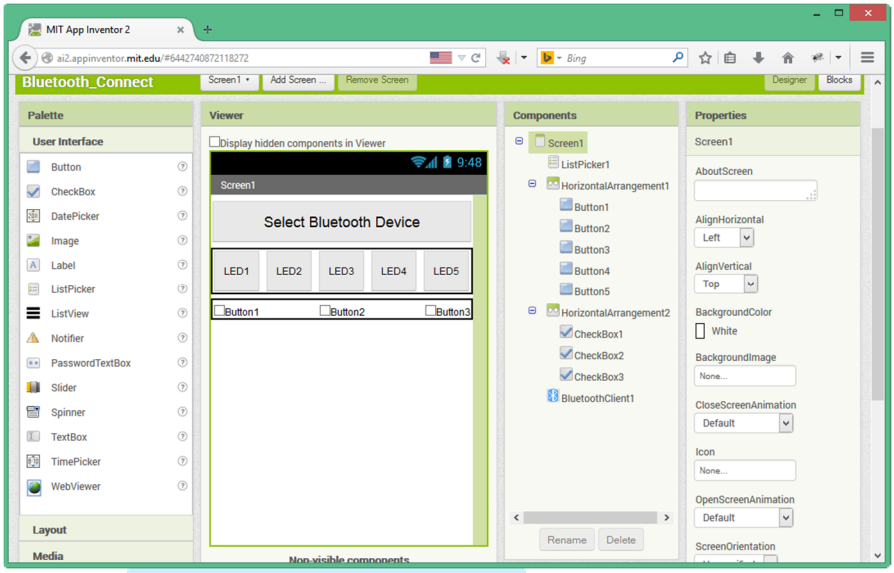
Next, drag another HorizontalArrangement component onto the Viewer and change the width property to "Fill parent", as shown below:



From the User Interface component group, select and drag the CheckBox component onto the 2nd HorizontalArrangement on Viewer and change the Text property to "Button 1", as shown below:

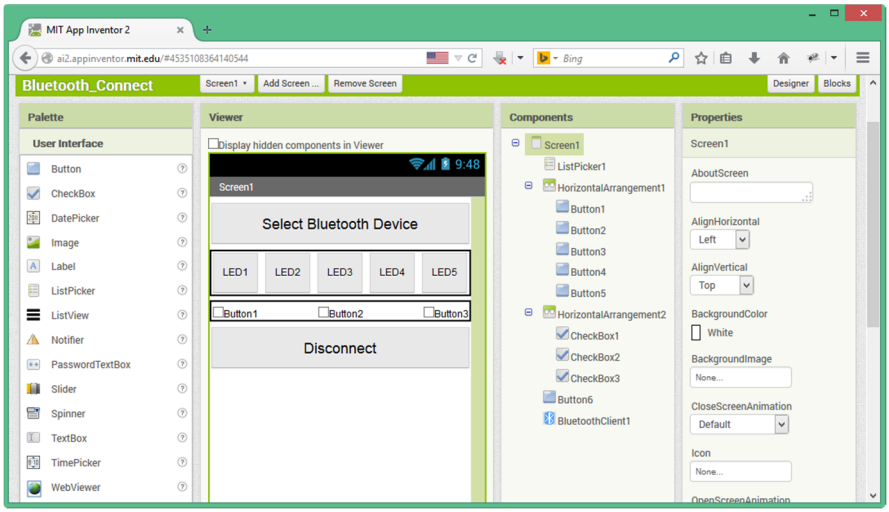


Using the same process, add two more CheckBox and change the text property to Button2 and Button3, as shown below:

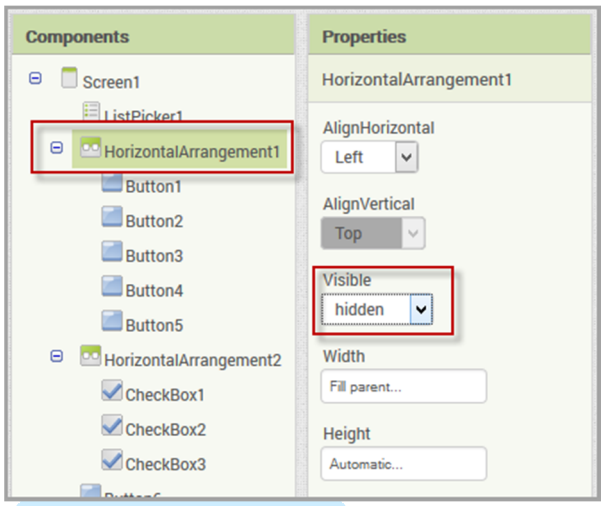


These three CheckBox components are linked to the push-button, which are part of the circuit attached to the EduCake for this exercise, and will be used to indicate when each of the push button is pressed.

Next, add one more button component onto the Viewer, change the Fontsize to "20", width property to "Fill parent" and the Text property to "Disconnect", as shown below:

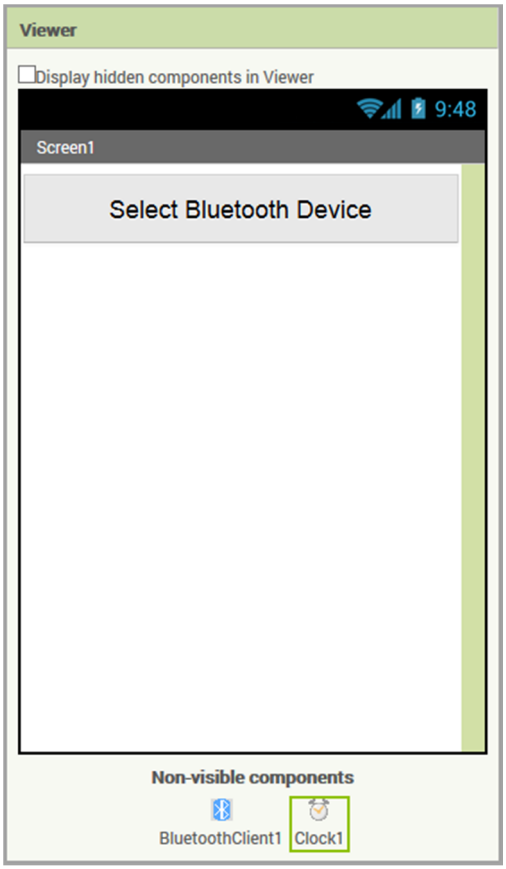


When the application is launched, prior to establishing Bluetooth connectivity, the button controls for the LEDs, CheckBox controls for the push-button and the Disconnect button should not be visible. To hide all 5 button controls for LED, we can simply change the Visible property HorizontalArrangement1 component to "hidden", as shown below:



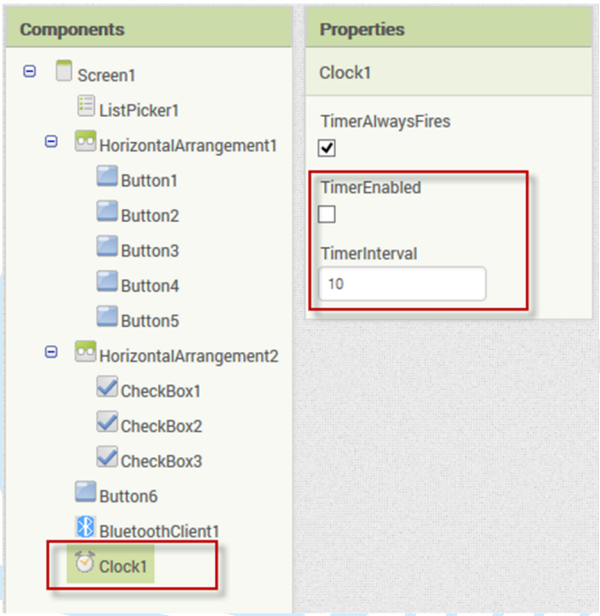
To hide all three CheckBox components, change the Visible property for HorizontalArrangement2 component to "hidden". To hide the Disconnect button, change the Visible property for Button6 to "hidden".

Next, from the Sensors component group in the Palette section, select and drag the Clock component onto the Viewer. The Clock component is a non-visible component and does not shows up on the app screen, as shown below:



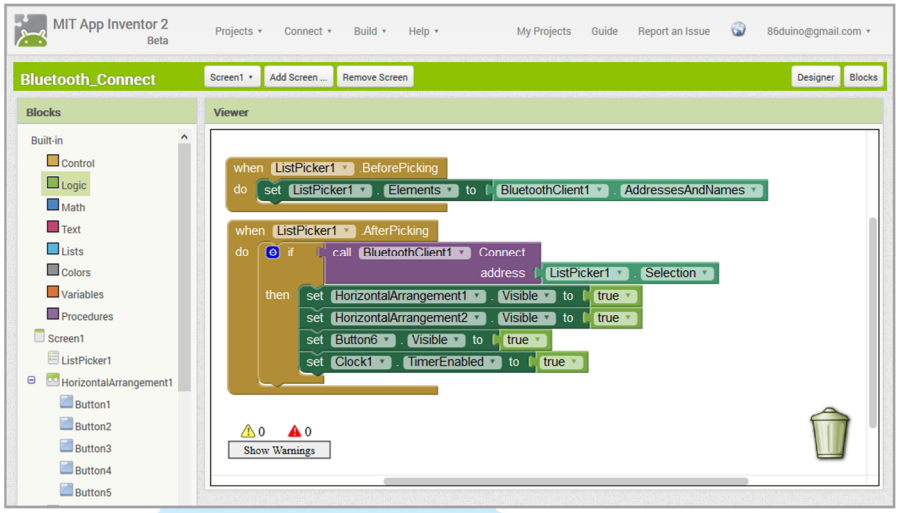
Change the following properties for Clock1 component:

- Uncheck the TimerEnabled checkbox
- Change the TimerInterval property to 10 millisecond

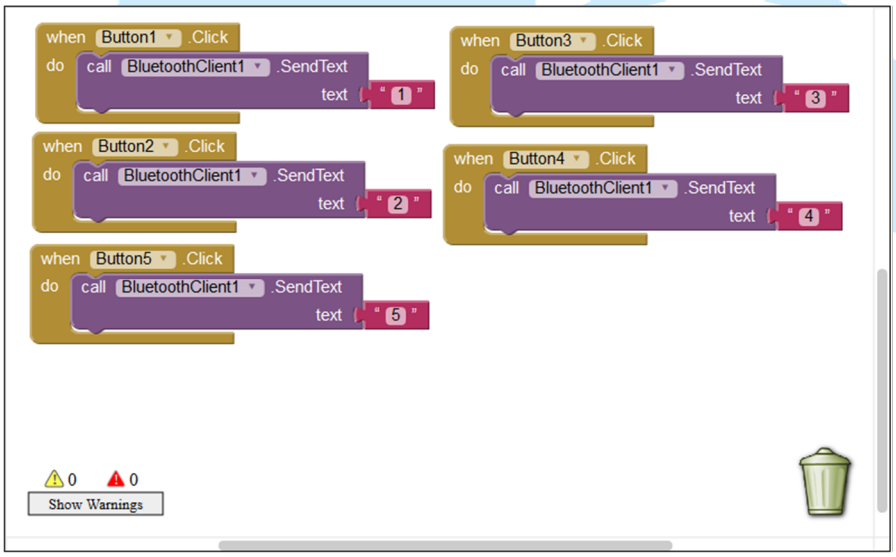


Next, with all of the required layout components in place, we will switch to the Blocks mode to add programming logic.

Similar to the app in the previous section, after establishing Bluetooth connectivity, we need to make the 5 LED button controls, 3 push-button Checkbox controls and the Disconnect button visible, and set TimerEnabled property for Clock1 to true. To do this, we need to change the Visible property for HorizontalArrangement1 (for the 5 LED button controls), HorizontalArrangement2 (for the 3 push-button Checkbox controls) and Button6 (Disconnect button) from hidden to visible, via the following function blocks:

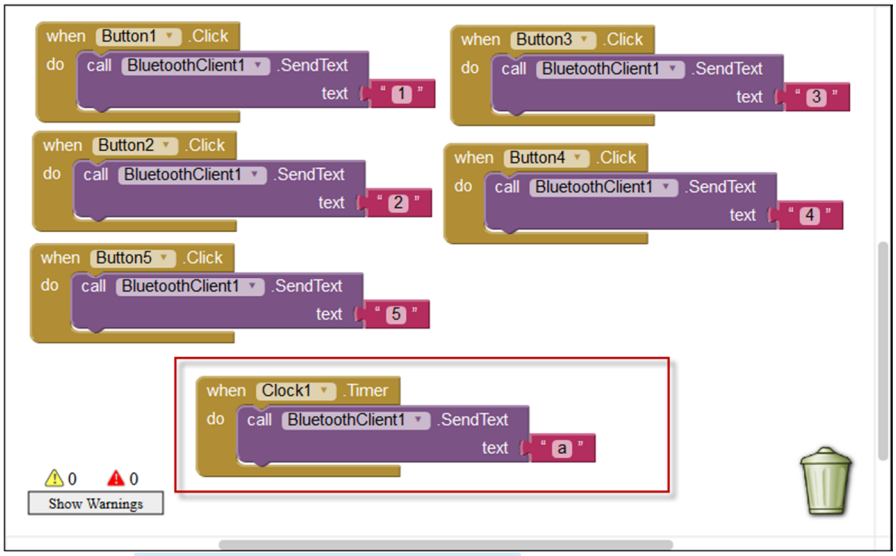


Next, we need to add function block to control the 5 LEDs via the Button.Click event and the BluetoothClient.SendText, as shown below:



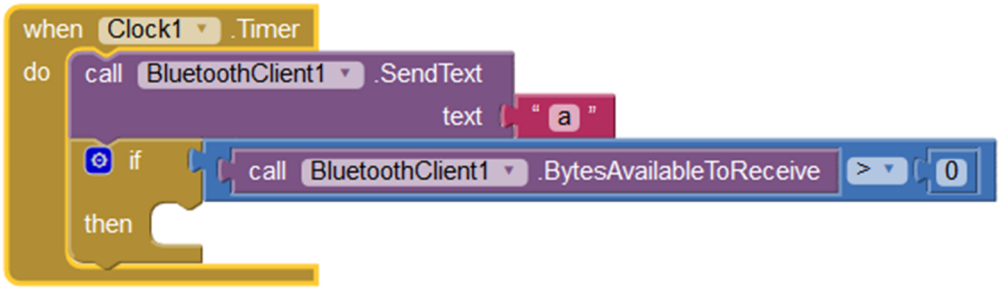
When each of these 5 buttons is clicked, the BluetoothJClient component send the corresponding value (1, 2, 3, 4 or 5) to EduCake to turn on the associated LED.

Next, the app needs to have a way to check whether any of the 3 push-button, part of the circuitry attached to the EduCake, is pressed, using Clock1.Timer event to send polling signal to EduCake, as shown below:

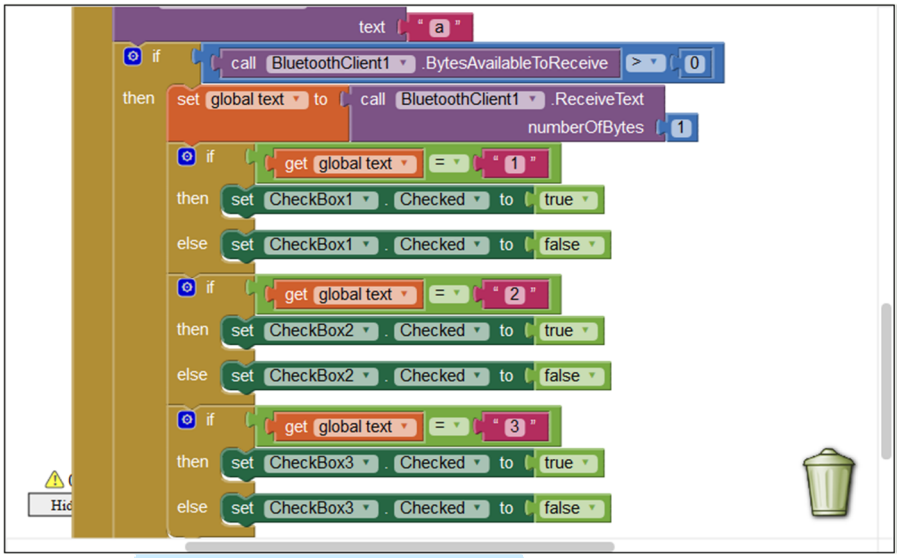


The Clock1.Timer event is set to fire every 10 millisecond. Each time the timer event fires, it triggers the BluetoothClient1 component to send the character "a" to EduCake.

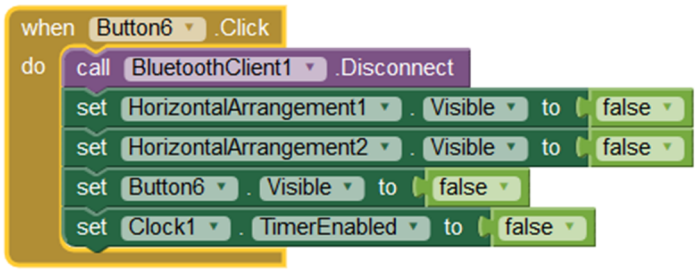
Next, we need a function block to check whether there incoming data received by the BluetoothClient1 with the following function block:



Then, add the following function block to retrieve the received data and enable the corresponding CheckBox:

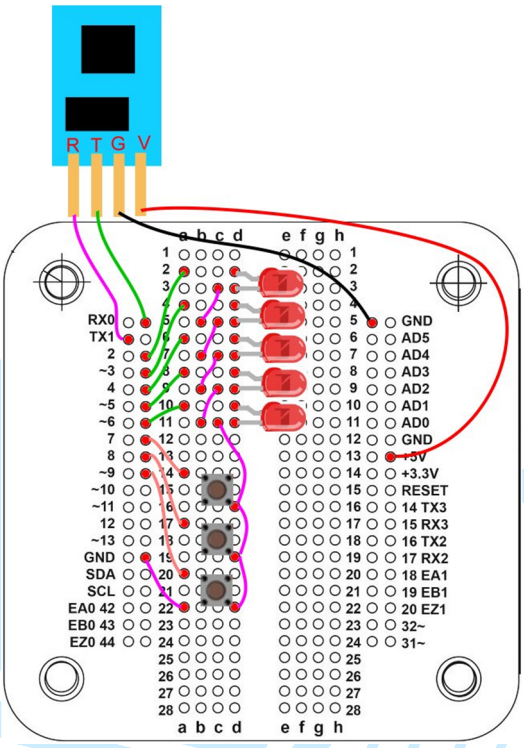


Next, we need to add function block for Button6, the Disconnect button. When Disconnect button is clicked, in addition to disconnect the Bluetooth connection, the app needs to hide the 5 LED button controls and the 3 push-button CheckBox and disable Clock1 from triggering timer event, using the following function block.



You may be wondering why Clock1 component is needed as part of the process to communicate with the EduCake. The Clock1 component is used to address timing issue associate with sending and receiving data. Within a mechanism to control timing, data transmitted to the EduCake via the Bluetooth connection is not predictable, can be too fast for the EduCake to respond to and cause data to be lost. Instead of transmitting data in an unpredictable manner, the app on the EduCake is written to wait for the Android device to request data to be send by transmitting the character "a". When EduCake received the character "a", it will return the intended data, where the Android device is ready to receive the data.

Next, we will attach 4 more LEDs to the EduCake and have 5 LEDs total on the circuit and another 3 push-button, as shown in the circuitry below:



From the 86Duino IDE, enter the following code:

```
char ch;
int LED[5]={0,0,0,0,0}; //  把 5 個腳位狀態先存到陣列裡面

void setup() {
   Serial.begin(9600); //  這行主要是設定和電腦的 COM  通訊的速度
   Serial1.begin(9600); //  這行主要是設定和藍芽  通訊的速度
   for(int a=2;a<7;a++) //  先設定這五個腳位為輸出模式
      pinMode(a, OUTPUT);
   for(int a=7;a<10;a++) //  先設定這三個腳位為輸入模式
      pinMode(a, INPUT);
}

void loop() {
   if (Serial1.available() > 0)
   {
```

```
ch = Serial1.read();
  if(ch == 'a' ) //  等待  Android 端傳送"a"過來
  {
    if(digitalRead(7)) //  偵測 pin7 按鈕是否按下
      Serial1.print("1");
    else if(digitalRead(8)) //  偵測 pin8 按鈕是否按下
      Serial1.print("2");
    else if(digitalRead(9)) //  偵測 pin9 按鈕是否按下
      Serial1.print("3");
    else
      Serial1.print("0");
  }
  else
  {
    if(LED[ch-'0'-1]==LOW)
    {
      LED[ch-'0'-1]=HIGH; //  儲存 LED 的狀態
      digitalWrite(ch-'0'+1, HIGH);
    }
    else
    {
      LED[ch-'0'-1]=LOW;
      digitalWrite(ch-'0'+1, LOW);
    }
  }
```

In the above code, it' s written to handle character value. The simplest method to convert to numeric value is to subtract 0. The code uses a 5 position array to handle the 5 LEDs, with the range from 0 to 4, by subtracting 1, we get the corresponding value, with the following line of

code:

```
if(LED[ch-'0'-1]==LOW)
```

On the Android device, when the LED1 Button is click which cause the app to send a “1” to EduCake, it’s corresponding to the first position in the array, which is “0”.

With the above code running on the EduCake, clicking on LED1 through LED5 on the Android device should turn on the corresponding LED on the EduCake. Pressing the push-button on the EduCake should cause the corresponding CheckBox to indicate the button is pressed.