# 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:

ew Project	Delete Project				
rojects		an an an an an an an an	an chilense co		
Name		Da	ite Create	d	
Name		Da	ite Create	d	
	Create	new App I	nventor	project	
	Create	<b>new App I</b> oject name:	nventor test	project	
	Create	e <b>new App I</b> oject name:	<b>Inventor</b> test	project	

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.

MIT App Inventor	r 2 eta	Project - Connect - Build - Help -	My Projects Guide Rep	ort an Issue
test		Screen1 • Add Screen Remove Screen		Designer Bloc
Palette		Viewer	Components	Properties
User Interface		Display hidden components in Viewer	Screen1	Screen1
Button	•	<b>⊜_/  1</b> 9:48		AboutScreen
CheckBox	•	Screen		
DatePicker	0			AlignHorizontal
Mage Image	•			Left 🔻
A Label	۲			AlignVertical
ListPicker	۲			Тор 🔹
ListView	•			BackgroundColor
Notifier	0			White
PasswordTextBox	۲			Backgroundimage
Slider	0			None
Spinner	1			CloseScreenAnimation
TextBox	0			Default
TimePicker	•			Icon
WebViewer	•			OpenScreenAnimation
				Default
Layout			Rename Delete	ScreenOrientation
Media				

### 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:

Width		
Automat	ic	
Fill parent	nt	
O pix	els	
Cancel	ОК	

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

Automat	ic
OFill parer	nt
●50 pi	xels
Cancel	ок

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:



www.86duino.com



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

MIT App Inventor 2 Beta	ojects Connect Build	Language Help My	Guide Report an Issue	86duino@gmail.com
ClickMe	Screen1 • Add Screen	Remove Screen		Designer Block
Blocks	Viewer			
Control Logic Math Text Lists Colors Variables Procedures Control Math Text Lists Colors Control Contr	get     get     i     get     i     get     i     i     i     i     initialize local     i     i     i     formatical initialize local     i     formatical     format	ame) to t		Û

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:

C alz.appinventor.mit.edu/	5598486549495808	▶ ☆ 目 ♥ ☆ ₩ ▼ =
Beta		ISSUE
ClickMe	Screen1  Add Screen Remove Screen	Designer Blocks
Blocks	Viewer	
Built-in		
Control	when Button1 .Click	
Logic	do set Label1 . Text . to C	
Math		
Text		
Lists		
Colors		
Variables		
Procedures		
Screen		
Alabell	<u>∧</u> 1 <u>∧</u> 0	
Laven	Show Wessings	

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

ai2.appinventor.mit.edu/4 Beta	5596480549493808	/ 目 ♣ 沓 ₩  ▼   = sue
lickMe	Screen 1 • Add Screen Remove Screen	Designer Blocks
llocks	Viewer	
Built and     Control     Logic     Math     Text     Lists     Colors     Variables     Procedures	when Buttons Oick do set Labelin Fick to I	
<ul> <li>Screen1</li> <li>Button1</li> <li>Label1</li> <li>Any component</li> </ul>	▲ 0 Show Warnings	

• 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

when Button1 Click do set Label1 , Text to ( Hello World) *		
when Button2 Click do eet Label1 . Text to to to		
▲ 0 ▲ 0 Show Warnings	Û	

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.

🗲 🕲 ai2.appir	ventor.mit.	edu/#55984865	49495808			- C 🚽	. 🖛 🕨 = Bing		٩	☆自	÷	ŵ	48.   <del>-</del>	1
IIT App Invento	or 2 Beta	Projects *	Connect *	Build *	Language	Help	My Projects	Guide	Repo	t an Ie	86du	ino@gm *	ail.com	
Лe		Screen1 •	Add Screen	Remove Scr	een							Designer	Bloc	ks
,		Viewer				0	omponents			Propertie	s			
nterface		Display h	idden component	ts in Viewer			Screen1			Button1				
itton	٦				9 📓 🖍 🦃	:48	Button1			Backgrour	ndColor			
eckBox	ClickMel	Progress Bar												
tePicker					20	าช								
age					20	<i>,</i>								
bel	Comp	iling part 1												
stPicker														
stView	۲		С	lear					1					
tifier	•	_								FontSize				
sswordTextBox	۲									20				
der	•									FontTypef	ace			
	0									default	~	J		
inner	~													

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

	Designate	Casarad	ild Language Hale		0	d in Gameilann	^
IT App Inventor 2 Beta	Projects	v v	Language Help	My Projects Guide	Issue 80	ouino@gmail.com *	
Лe	Screen1 •	Add Screen Rem	rove Screen			Designer Blocks	
	Viewer		Barcode link for ClickMe		Properties		
nterface	Display h	idden components in	<b>MATCH</b>	1	Button1		
itton ieckBox itePicker iage	Screen1	Click N		ton1 el1 ton2	BackgroundCo Default Enabled TontBold	lor	
bel stPicker stView stifier		Clea	OK Note: this barcode is only valid for hours. See the FAQ for info on how to share your app with others.	2	FontItalic FontSize 20		
ider Jinner					FontTypeface default Image	Y	

• 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:

IIT App Inventor	r 2 eta	Projects	Connect *	Build Language Help	My Projects Guide	Repor	tan ie	86duino@	Pgmail.com *	Â
Лe		Screen1 •	Add Screen	Opening Cli	ckMe.apk	×		Desig	ner Block	ks -
		Viewer		You have chosen to open:			operties			
nterface		Display his	den component	ClickMe.apk which is: ank File (1.3 MB)			rtton1			
itton	۲			from: http://ai2.appinventor.mit.e	du		ickgroun	dColor		
eckBox	۲	Screen1		What should Firefox do with this file?			Default			
itePicker	۲		Clic	O Open with Browse			abled			
age	۲			<ul> <li>Save File</li> </ul>			ntBold			12
bel	()			Do this <u>a</u> utomatically for files like	e this from now on.					
stPicker	()						intItalic			
stview			CI		OK Cano	cel	intSize			
sswordTextBox							20			and a
der							FontTypefa	ice		
inner	•						default	*		
xtBox	•						Image			~

### 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:

) 🕙 ai2.appinventor.mit.edu/#5598486	6549495808 📃 🤍 C 🦊 💌 Bing
MIT App Inventor 2 Beta	Projects Connect Build Language Help My
ClickMe	My projects
	Start new project
Palette	Import project (.aia) from my computer Components
User Interface	Import project (.aia) from a repository 🛛 🗧 🗍 Screen
Button 🧿	Delete Project 3 9:48
Create new App	p Inventor project
Create new App Project name:	p Inventor project Bluetooth_Connect

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:

MIT App Inventor 2 ×	+			- <b>-</b> ×
🔶 🕲 ai2.appinventor. <b>mit.edu</b> /#64427	740872118272	■■ マ (	C 😓 🔻 🕨 - Bing	▶ ☆自 ♣ ☆ * ► =
MIT App Inventor 2 Beta	Projects Connec * *	Build Language * *	Help My * Projects Guide	Report an 86duino@gmail.com s
Bluetooth_Connect	Screen1 • Add Screen	. Remove Screen		Designer Blocks
Palette	Viewer		Components	Properties
User Interface	Display hidden compor	ents in Viewer	B Screen1	ListPicker1
Button     ○       Image     ○	Screen1 Text for ListPicker1	<b>♥</b> // M № 994	S EistPicker1	BackgroundColor Default ElementsFromString Enabled FontBold FontBold FontBild FontSize 14.0

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:

Bluetooth_Connect
Option 1
Option 2
Option 3
Option 4

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:

Palette	Viewer		Co	mponents	Propert	ies		
User Interface	Display hidden components in Viewer		Θ	Screen1	Bluetoo	thClient	1	
Layout	<b>⇒⊿ №</b> 9:4	8		ElistPicker1	Charact	rEncodi	ng	
Media	Screent	- 1		BluetoothClient1	UTF-8			
Drawing and Animation	Select Bluetooth Device				Delimite	rByte		
Sensors					0			
Social					HighByt	eFirst		
Storage					Secure			
Connectivity								
ActivityStarter (*)	1							
BluetoothClient (?	1							
BluetoothServer (?)								
SWeb 🕐								
LEGO® MINDSTORMS®								
				Rename Delete				
	Non-visible components							

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:

### 86DUIND

www.86duino.com

) 🕲 al2.appinventor. <b>mit.ed</b>	<b>u</b> /#644274	0872118272	C 😽 👻 🕨 Bing	▶ ☆ 自 ♣ 舍 ₩ ▼
MIT App Inventor	2 ta	Projects Connect Build Language * * * * * * Screen 1 * Add Screen Remove Screen	Help My • Projects Guide	Report an 86duino@gmail.com Issue * Designer Blocks
Palette		Viewer	Components	Properties
User Interface		Display hidden components in Viewer	Screen1	Button1
Duthing     CheckBox     CheckBox     Image     Label     Label     LabPicker     ListView     Notifier     PasswortTextBox     Sider     Sider     TextBox     TextBox     Wabdiser		Soreen1 Select Bluetooth Device	Button BisetoothClient	BackgroundColor ■ Default Enabled ■ FontBold = FontBale 20 FontTypeface default Image None Shape

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: nn.

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

when ListPicker1 v BeforePicking do set ListPicker1 v Elements v to b		
1 A 1 Show Warnings	Û	

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:

when ListPicker1 • BeforePicking do set ListPicker1 • Elements • to BluetoothClient1 • AddressesAndNar	nes 🔪
when ListPicker1 AfterPicking	
▲ 1 ▲ 0 Show Warnings	Û

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:

when	ListPicker1 . AfterPi	cking			1
do 🚺	if (call Bluetoc	thClient1 .Connect			
t	en set Button1 v	address (	ListPicker1 .	Selection •	

The 2<sup>nd</sup> 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:

do set ListPicker1 • . Elements • to	BluetoothClient1 * AddressesAndNames *	
when ListPicker1 .AfterPicking		
do 👩 if 🔓 call BluetoothClient1 🔻	Connect	
	address (ListPicker1 . Selection .	
then set Button1 . Visible t	o t <mark>i true v</mark>	
when Button1 .Click		
do call BluetoothClient1 .SendText		
text		~
	F	

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

when         ListPicker1 •         .BeforePicking           do         set         ListPicker1 •         . Elements •         to         BluetoothClient1 •         . AddressesAndNames •	
when ListPicker1 AfterPicking	
do 💿 if 📙 call BluetoothClient1 💌 .Connect	
address (ListPicker1 V). Selection V	
then set Button1 . Visible to to true .	
when Button1 Click	
do call BluetoothClient1 .SendText	
text (, "a"	
	$\bigcirc$
Show Warnings	

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.

86DUIND www.86duino.com

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:



86DUIND www.86duino.com

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:

#### 86Duind

www.86duino.com



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  $\cdot$  an<sup> $\Gamma$ </sup> OK  $_{\rm J}$  respond indicate the module is functioning as expected.

- AT+NAMEaaa : Change the device name to <sup>r</sup> aaa <sub>J</sub>
- AT+PIN1234 : Change the pairing password to  $\lceil 1234 \rfloor$
- 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.

4	COM5	- 🗆 🗙	
AT		Send	
Enter AT commands: OK			
✔ Autoscroll	No line ending	✓ 9600 baud ✓	

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 2<sup>nd</sup> 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
```



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:

86DUIND www.86duino.com





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:

Screen1	
Select Bluetooth Device	
Click Me	

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:

MIT App Inventor 2 ×	+					×
e ai2.appinventor.mit.edu/#6442740	3872118272 📕 🛡 🧭		🕨 - Bing 🔎	☆自∔☆	*	=
User Interface Layout	Display hidden components in Viewer	8	Screen1	HorizontalArrangeme AlignHorizontal	.nt1	^
HorizontalArrangement     TableArrangement     TableArrangement     VerticalArrangement     T	Select Bluetooth Device		BluetoothClient1	Left v AlignVertical Top v		
Media Drawing and Animation				Visible showing		
Sensors Social				Fill parent Height		
Storage Connectivity				Automatio		
LEGO® MINDSTORMS®						

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:

MIT App Inventor 2	×	+		- • ×			
🗲 🛞 ai2.appinventor.mit.e	🔄 🛞 ai2.appinventor.mit.edu/#6442740872118272 🔤 🗸 🔍 💺 💌 👂 🏟 🗭 🖈 🕫 💌						
MIT App Invento	o <b>r 2</b> Beta	Projects Connect Build Languag * * * * *	pe Help My R ★ Projects Guide	eport an 86duino@gmail.com Issue *			
Bluetooth_Connect	t	Screen1 • Add Screen Remove Screen		Designer Blocks			
Palette		Viewer	Components	Properties			
User Interface		Display hidden components in Viewer	😑 📃 Screen1	Button1			
Button	() ()	Screen1	9:48 ElistPicker1 ElistPicker1	BackgroundColor Default			
DatePicker	0	Select Bluetooth Device	BluetoothClient1	Enabled			
Label	0	LED1		FontBold			
ListPicker	(7) (7)			FontItalic			
▲ Notifier	0			FontSize			
PasswordTextBox	?			14.0			

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

Screen1				<b>≷⊿l 🛿</b> 9:4	18
Select Bluetooth Device					
LED1	LED2	LED3	LED4	LED5	

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 2<sup>nd</sup> HorizontalArrangement on Viewer and change the Text property to "Button 1", as shown below:

	MIT App Inventor 2	×	+				- • ×
(+)	ai2.appinventor.mit.edu/#	644274	0872118272		- v G 4	😓 🔻 🕨 🕶 Bing	₽ ☆ 🗎 🖡 🏦 🦇 💌 🚍
~	MIT App Inventor 2 Beta		Projects Connect v v	Build Lar	nguage He * *	elp My F Projects Guide	Report an 86duino@gmail.com Issue v
Blu	etooth_Connect		Screen1 • Add Screen .	. Remove Screen			Designer Blocks
Pal	ette		Viewer			Components	Properties
Us	er Interface		Display hidden compor	ents in Viewer		B Screen1	Screen1
	Button CheckBox	? ?	Screen1	<del>,</del>	al 🛿 9:48	E ListPicker1	AboutScreen
20	DatePicker Image	(?) (?)	Select Bl	uetooth Device		Button1	AlignHorizontal
A	Label ListPicker	() ()	LED1 LED2	LED3 LED4	LED5	Button4	AlignVertical Top
	ListView Notifier	(?) (?)	Button 1			<ul> <li>HorizontalArrangement2</li> <li>CheckBox1</li> </ul>	BackgroundColor
••	PasswordTextBox Slider	? ?				BluetoothClient1	BackgroundImage None
	Spinner TextBox	? ?					CloseScreenAnimation Default
	TimePicker WebViewer	() ()					Icon None

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

#### 86Duind

www.86duino.com

MIT App Inventor 2	×	+		- • ×
( ai2.appinventor.mit.e	<b>du</b> /#644274	0872118272 🗖 🧟 🧭	🖶 💌 🕨 - Bing 🔎	☆自 ∔ 合 ⊮ ◄ Ξ
Bluetooth_Connect	t	Screen1 • Add Screen Remove Screen		Designer Blocks
Palette		Viewer	Components	Properties
User Interface		Display hidden components in Viewer	B Screen1	Screen1
Button	7	9:48 📓 9:48	ElistPicker1	AboutScreen
CheckBox	0	Screen1	HorizontalArrangement1	
DatePicker	0	Select Bluetooth Device	Button1	AlignHorizontal
image	0		Button2	Left 🗸
A Label	0	LED1 LED2 LED3 LED4 LED5	Button4	AlignVertical
E ListPicker	(?)		Button5	Тор 🛩
ListView	0	Button1 Button2 Button3	😑 🔤 HorizontalArrangement2	BackgroundColor
A Notifier	0		CheckBox1	White
PasswordTextBox	(?)		CheckBox2	BackgroundImage
Slider	0		CheckBox3	None
Spinner	0		BluetoothClient1	CloseScreenAnimation
TextBox	(?)			Default
TimePicker	(?)			Icon
WebViewer	0			None
				OpenScreenAnimation
Layout			Contraction of the second s	Default
Media		Non-visible components	riename Delete	ScreenOrientation

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:

Components	Properties		
😑 🔲 Screen1	HorizontalArrangement1		
E ListPicker1	AlignHorizontal		
Button1 Button2 Button3 Button4	AlignVertical Top Visible hidden		
Buttons     MorizontalArrangement2     CheckBox1     CheckBox2     CheckBox3	Width Fill parent Height Automatic		

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:

Viewer			
Display hidden components in Viewer			
<b>⇒.</b> al 8 9:48			
Screen1			
Select Bluetooth Device			
Non-visible components			
BluetoothClient1			

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:

#### 86Duind

www.86duino.com

MIT App Inventor 2 Beta	Projects • Connect • Build • Help • My Projects Guide Report an Issue 🚱 86duino	⊉gmail.com ∗
Bluetooth_Connect	Screen1 • Add Screen Remove Screen De	signer Blocks
Blocks	Viewer	
Built-in  Control Control Cogic Math Text Lists Colors Yariables Procedures Screent ListPicker1	when ListPicker1 BeforePicking do et ListPicker1 AterPicking do if call BluetoothClient1 Connect address ListPicker1 Selection + then set HorizontalArrangement1 Visible to true + set HorizontalArrangement2 Visible to true + set Clock1 TrueFinabled to t true +	
Button1 Button2 Button3 Button4 Button5 Butto	▲ 0 ▲ 0 Show Warnings	Û

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:

#### 86Duind

www.86duino.com



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.

whe	n Button6 🝸 .Click
do	call BluetoothClient1 .Disconnect
	set (HorizontalArrangement1 • ). Visible • ) to (false •
	set (HorizontalArrangement2 • ). Visible • ) to (false •
	set Button6 • . Visible • to false •
	set Clock1 . TimerEnabled . to false

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)
    {
```

86Duino www.86duino.com



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.

Screen1	
Select Bluetooth Device	
LED1 LED2 LED3 LED4 LED5	
Button1 Button2 Button3	1.
Disconnect	