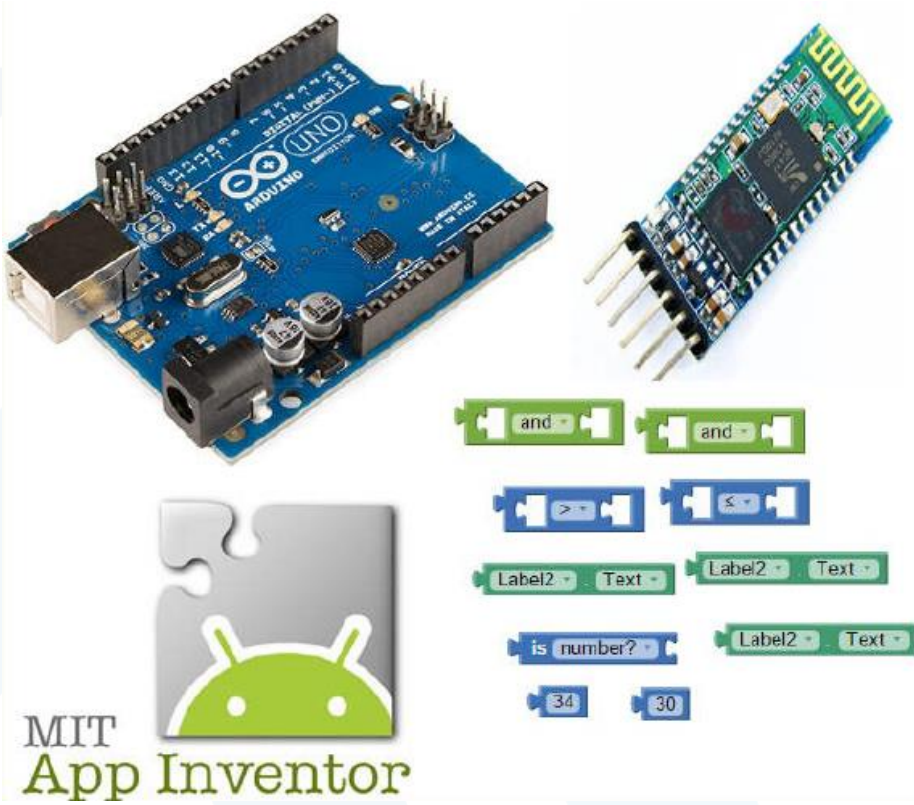
كلية الهندسة
قسم هندسة الروبوت والأنظمة الذكية

جَامعة المَنارة
MANARA UNIVERSITY

تطبيقات ميكاترونيك1

محاضرة 4

Arduino and Android using MIT app inventor



د. فادي متوج
د.عيسى الغنام

2024-2023

What is MIT App Inventor?

- MIT App Inventor is an innovative beginner's introduction to programming and app creation that transforms the complex language of text-based coding into visual, drag-and-drop building blocks.

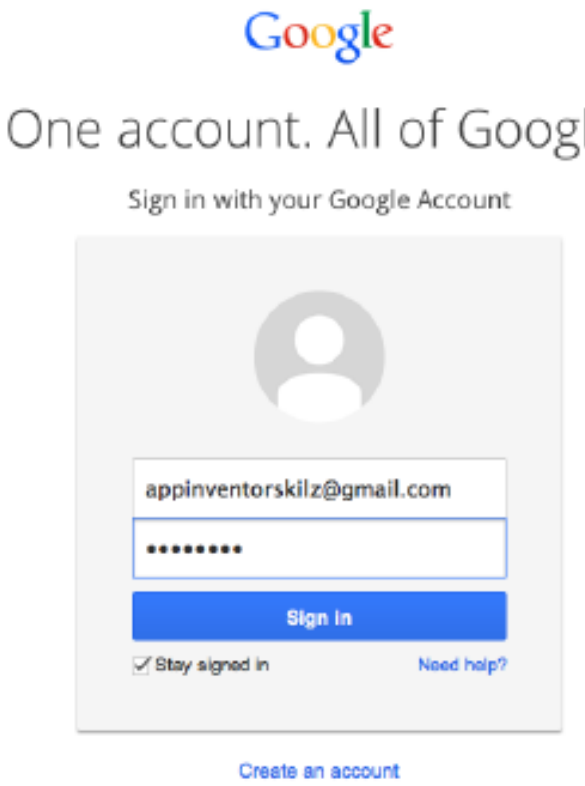
جَامعة
المَنارة
MANARA UNIVERSITY

TalkToMe: Our first App Inventor app

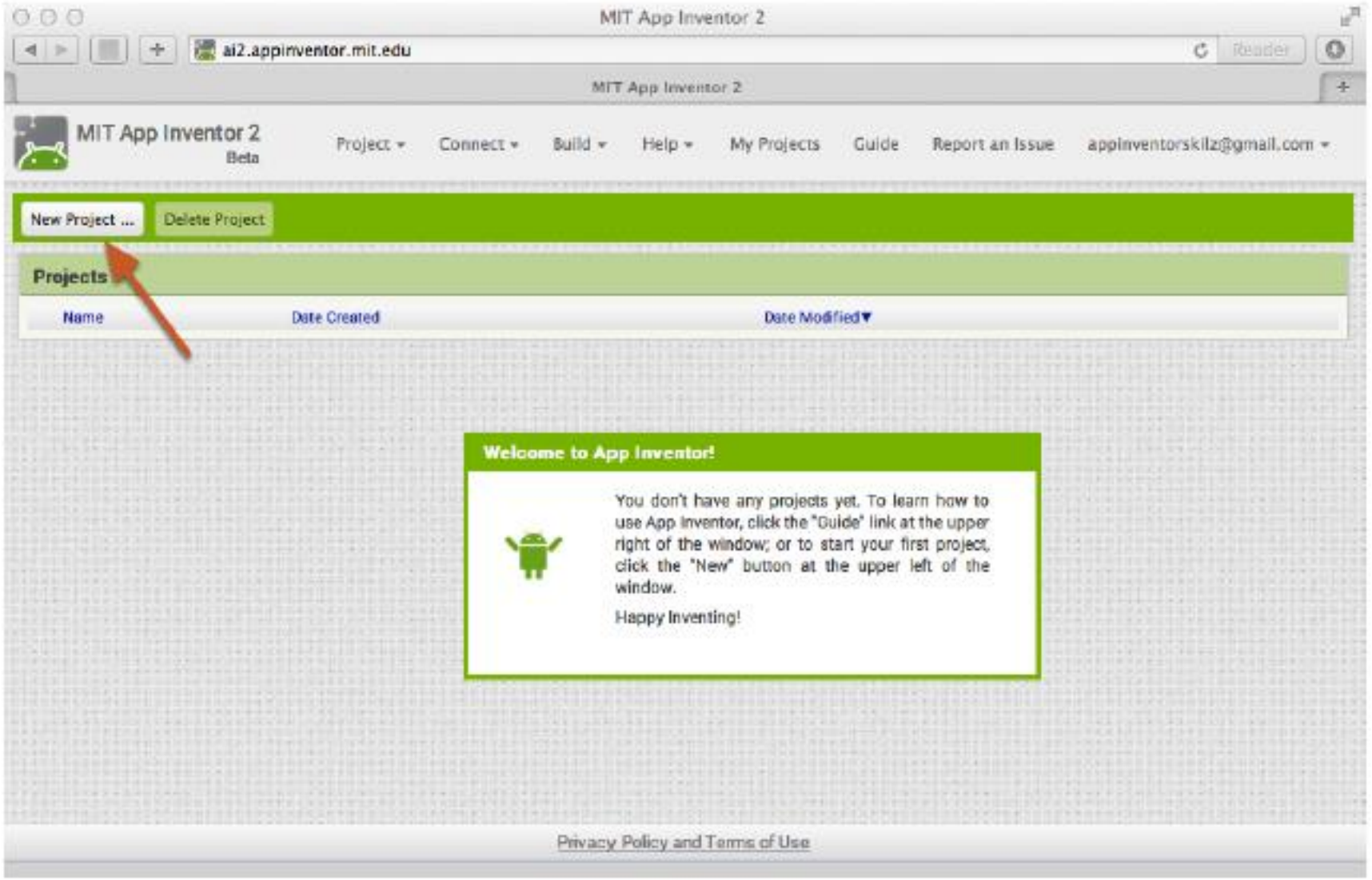
جَـامعة
المَـنارة
MANARA UNIVERSITY

Log in to App Inventor with a gmail (or google) user name and password.

Use an existing gmail account or school-based google account to log in to ai2.appinventor.mit.edu
To set up a brand new gmail account, go to accounts.google.com/SignUp



Start a new project.

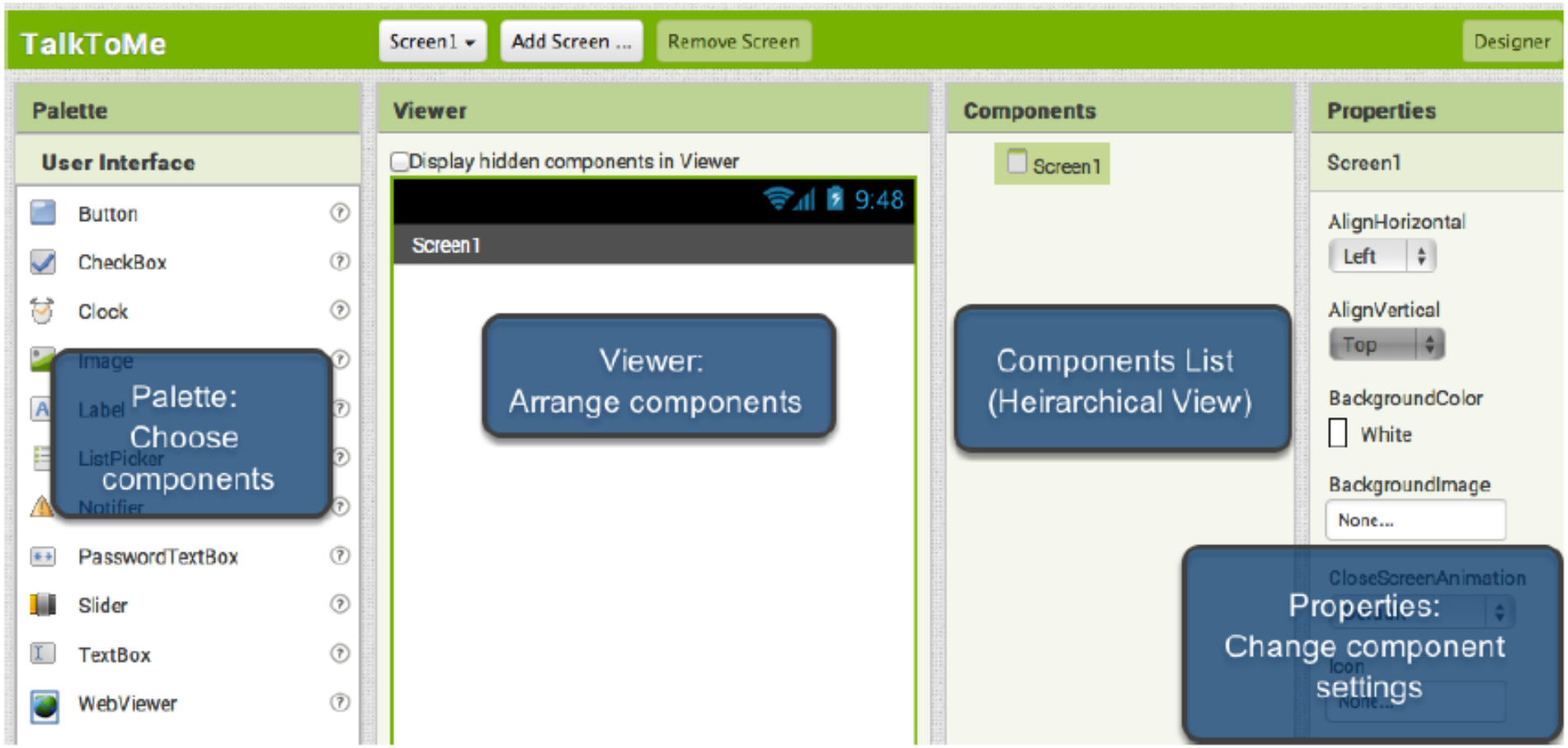


Name the project "TalkToMe" (no spaces!)

Type in the project name (underscores are allowed, spaces are not) and click OK.

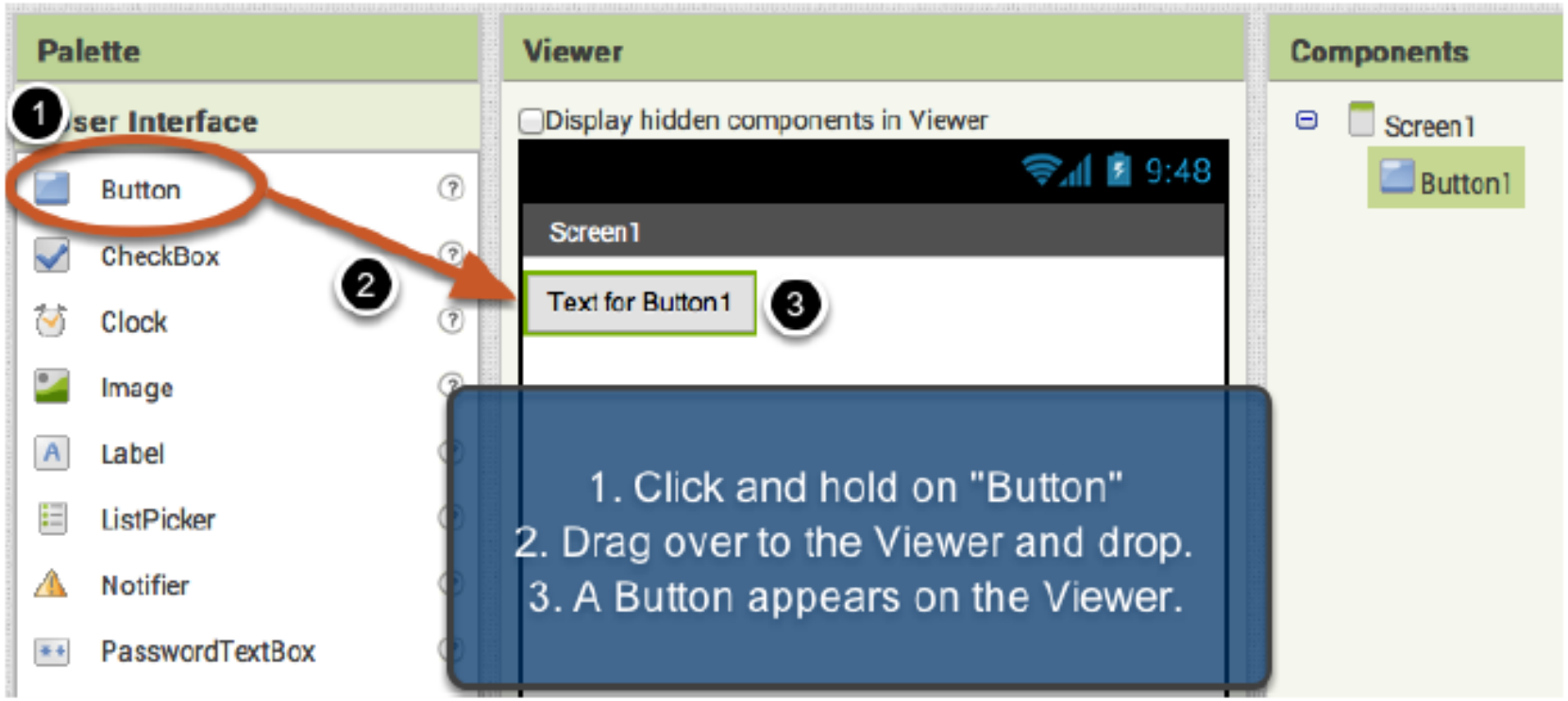
You are now in the Designer, where you lay out the "user interface" of your app.

The Design Window, or simply "Designer" is where you lay out the look and feel of your app, and specify what functionalities it should have. You choose things for the user interface things like Buttons, Images, and Text boxes, and functionalities like Text-to-Speech, Sensors, and GPS.



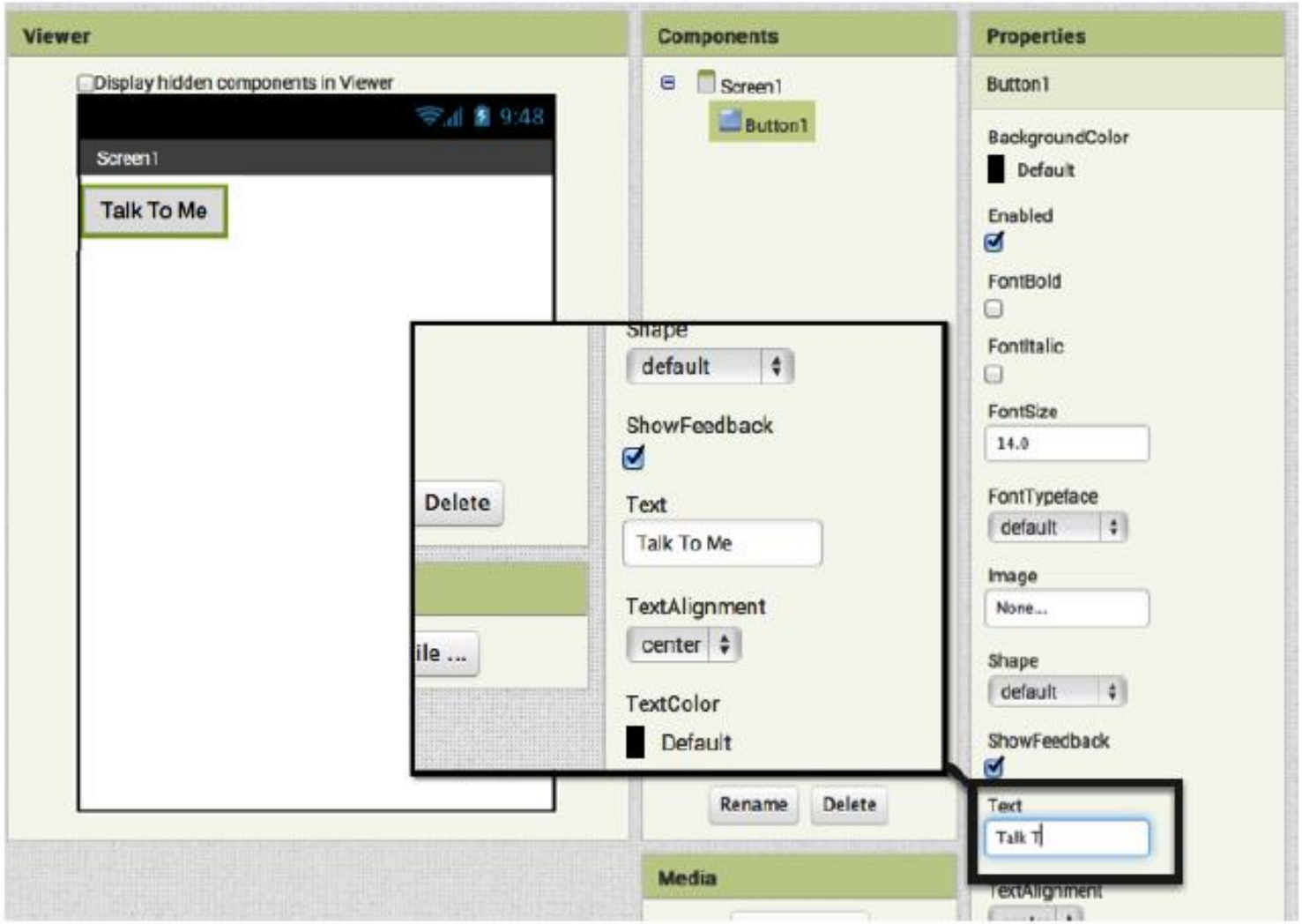
Add a Button

Our project needs a button. ***Click and hold*** on the word "Button" in the palette. ***Drag*** your mouse over to the Viewer. ***Drop*** the button and a new button will appear on the Viewer.



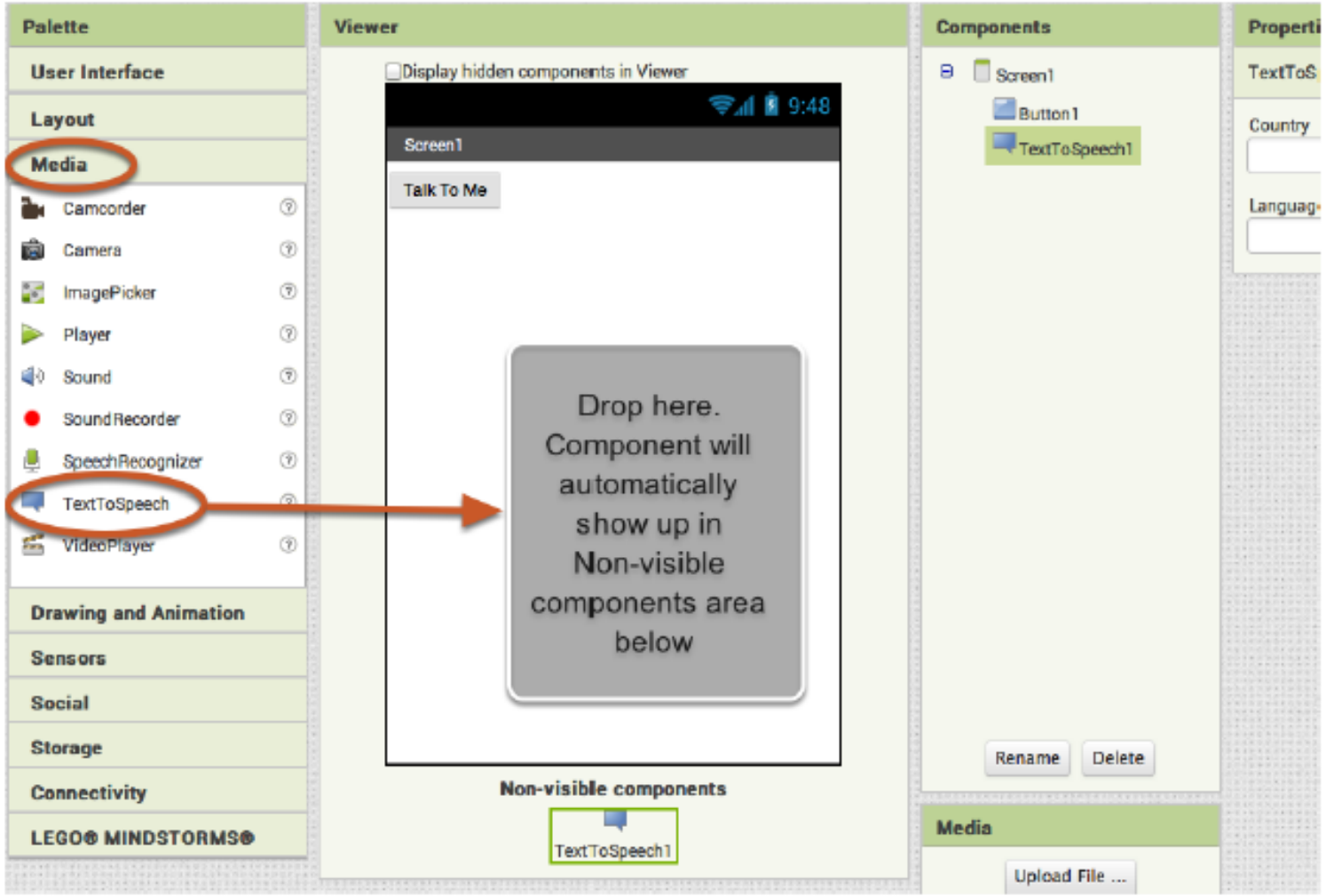
Change the Text on the Button

On the properties pane, change the text for the Button. Select the text "Text for Button 1", delete it and type in "Talk To Me". Notice that the text on your app's button changes right away.



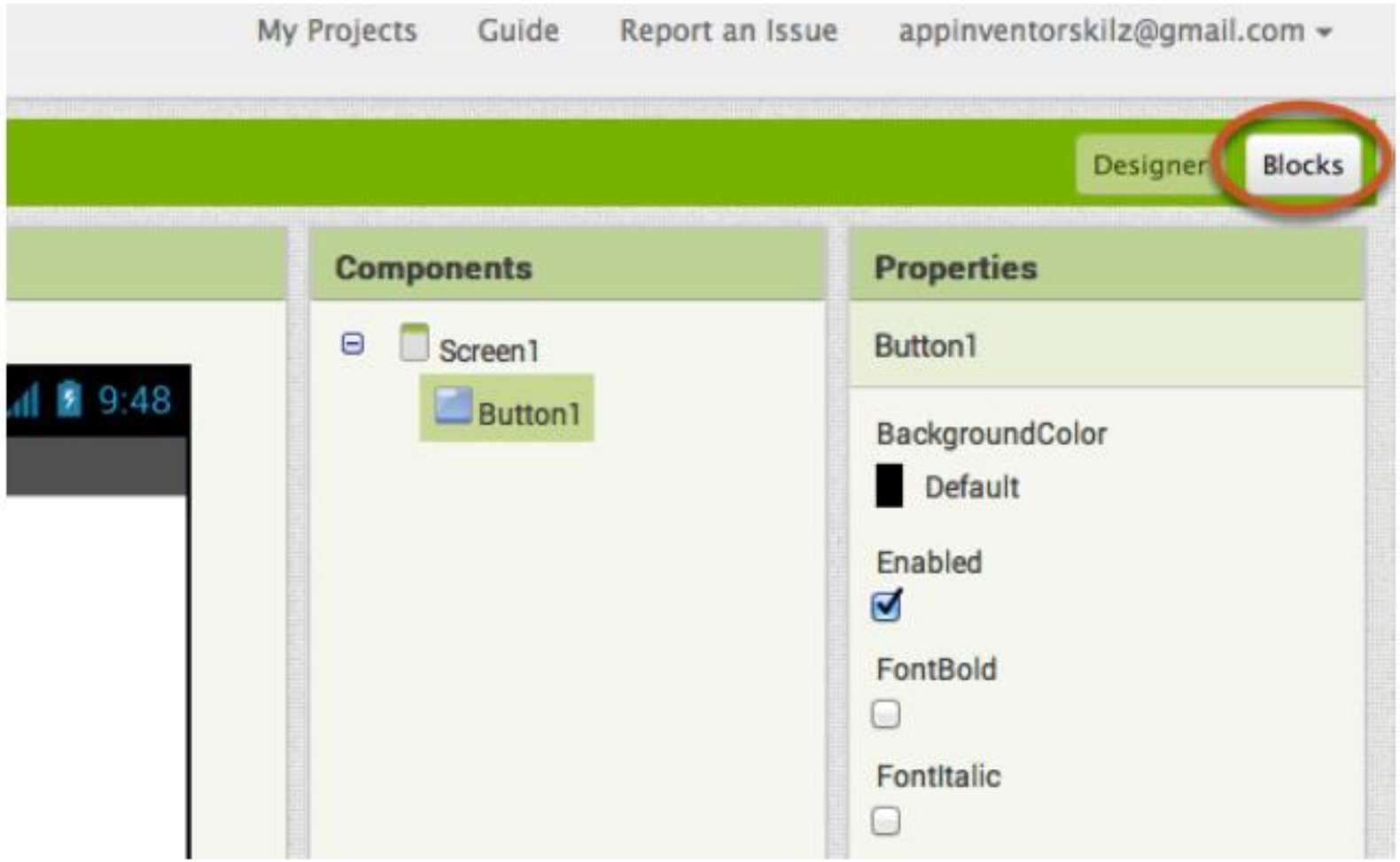
Add a Text-to-Speech component to your app

Go to the Media drawer and drag out a TextToSpeech component. Drop it onto the Viewer. Notice that it drops down under "Non-visible components" because it is not something that will show up on the app's user interface. It's more like a tool that is available to the app.



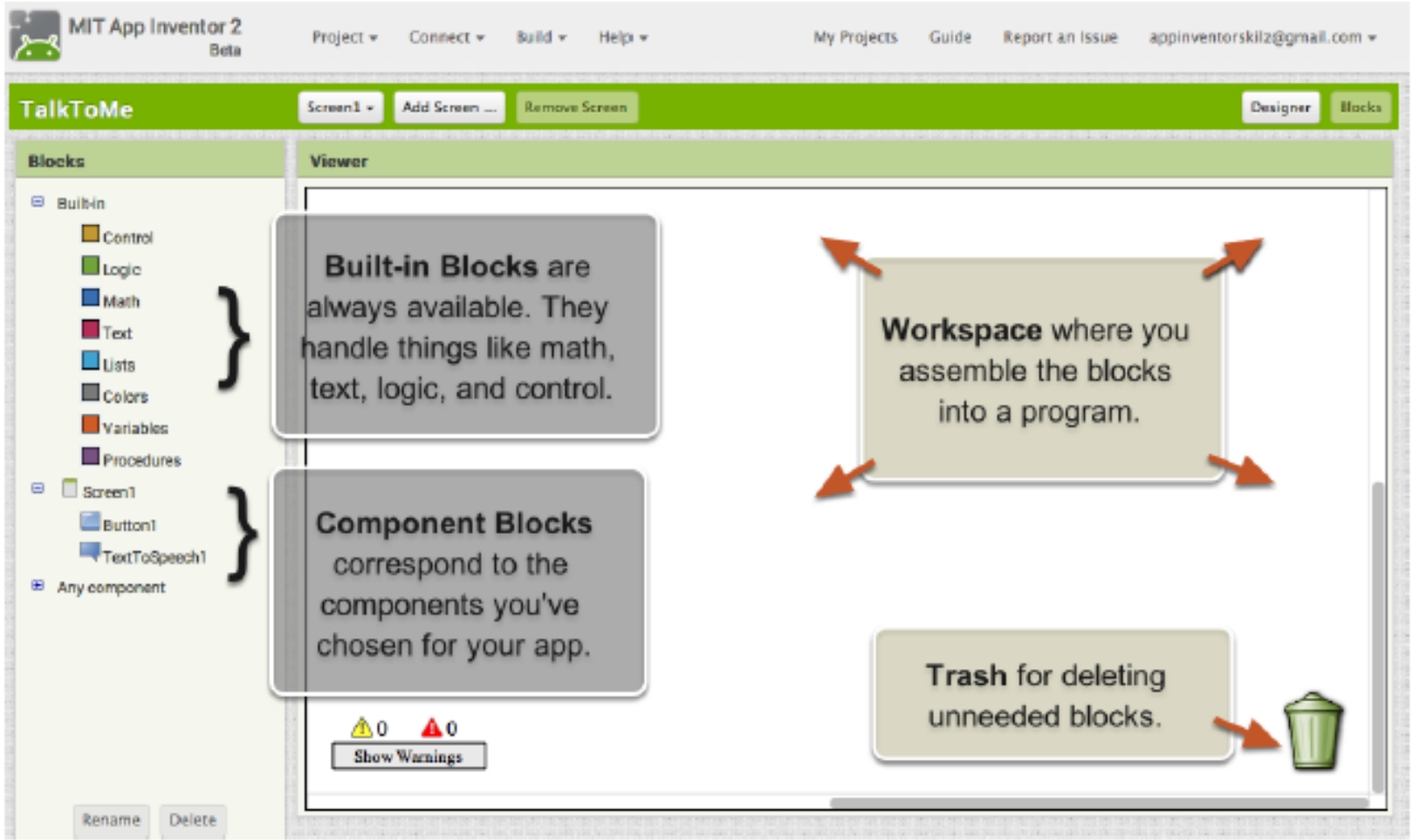
Switch over to the Blocks Editor

It's time to tell your app what to do! Click "Blocks" to move over to the Blocks Editor. Think of the Designer and Blocks buttons like tabs -- you use them to move back and forth between the two areas of App Inventor.



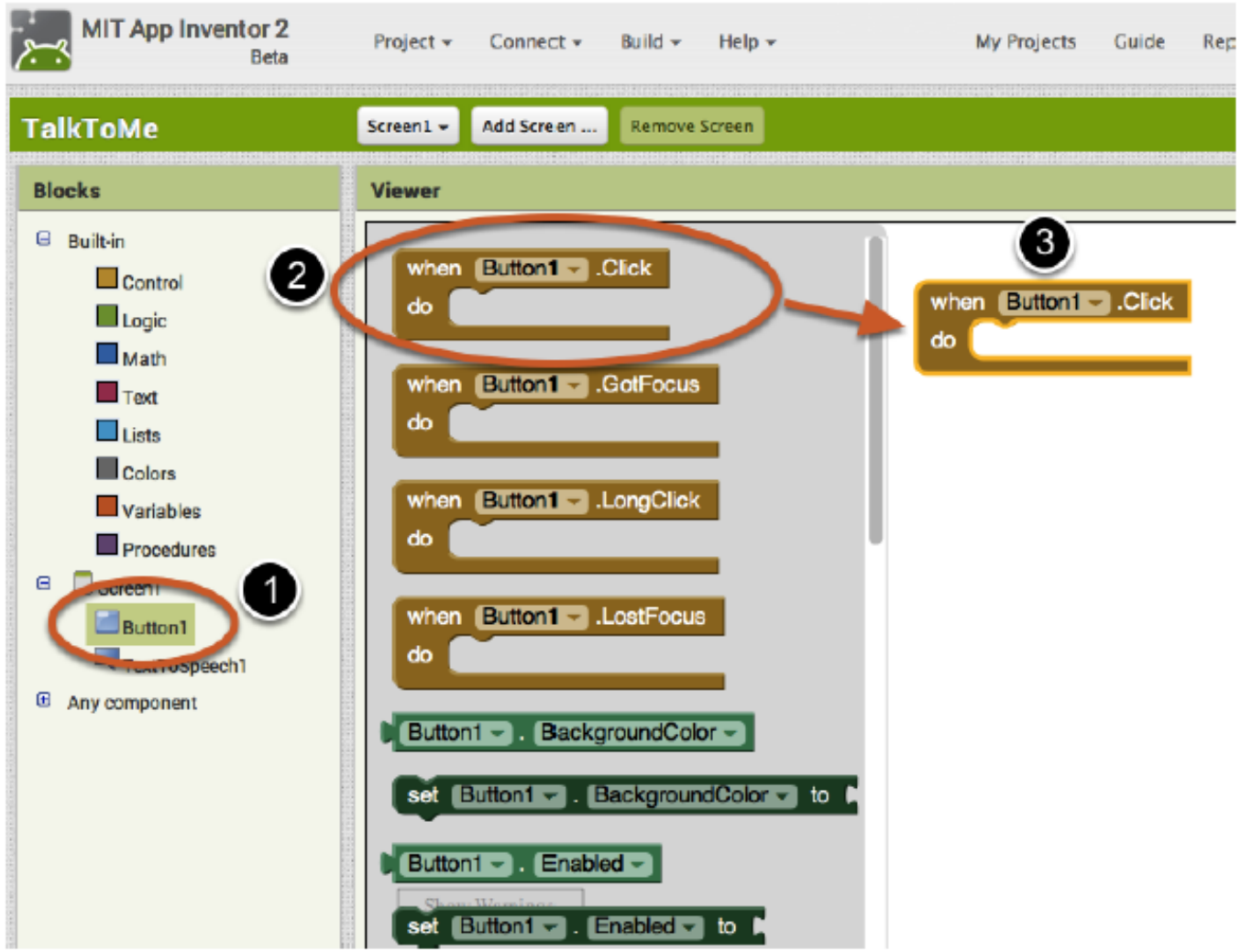
The Blocks Editor

The Blocks Editor is where you program the behavior of your app. There are Built-in blocks that handle things like math, logic, and text. Below that are the blocks that go with each of the components in your app. *In order to get the blocks for a certain component to show up in the Blocks Editor, you first have to add that component to your app through the Designer.*



Make a button click event

Click on the Button1 drawer. Click and hold the ***when Button1.Click do*** block. Drag it over to the workspace and drop it there. This is the block that will handle what happens when the button on your app is clicked. It is called an "Event Handler".

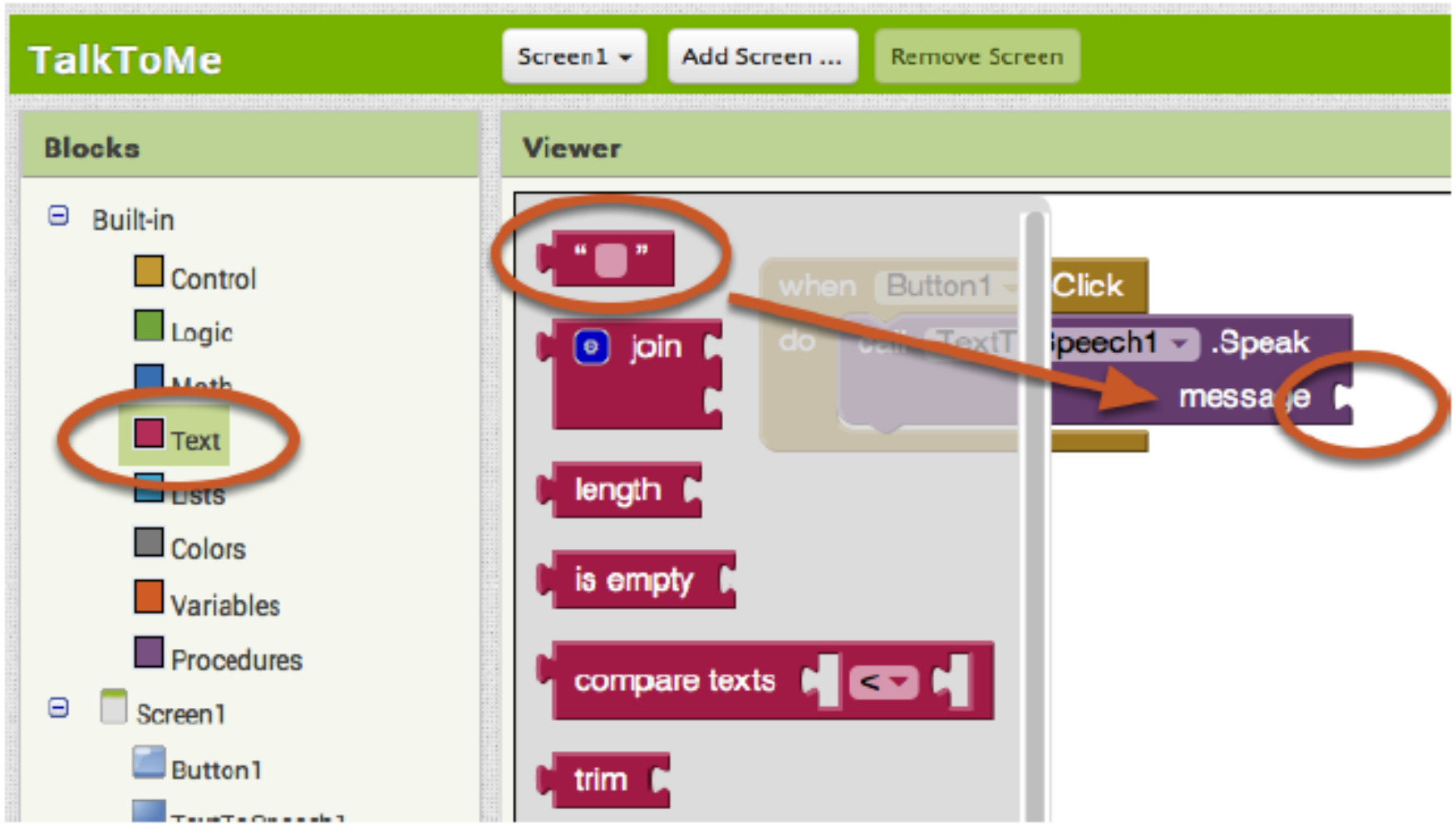


Program the TextToSpeech action

Click on the TextToSpeech drawer. Click and hold the ***call TextToSpeech1.Speak*** block. Drag it over to the workspace and drop it there. This is the block that will make the phone speak. Because it is inside the Button.Click, it will run when the button on your app is clicked.

Fill in the message socket on TextToSpeech.Speak Block

Almost done! Now you just need to tell the TextToSpeech.Speak block what to say. To do that, click on the Text drawer, drag out a ***text*** block and plug it into the socket labeled "message".

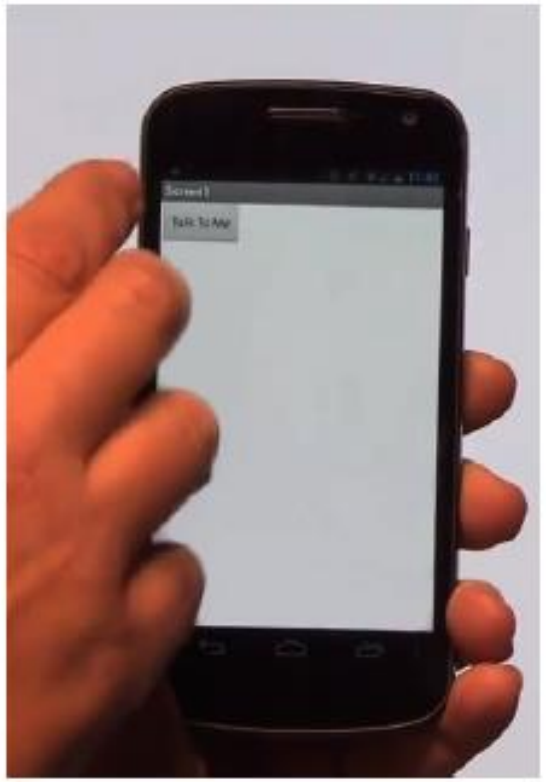


Specify what the app should say when the button is clicked

Click on the text block and type in "Congratulations! You've made your first app." (Feel free to use any phrase you like, this is just a suggestion.)

Now test it out!

Go to your connected device and click the button. Make sure your volume is up! You should hear the phone speak the phrase out loud.



Great job!

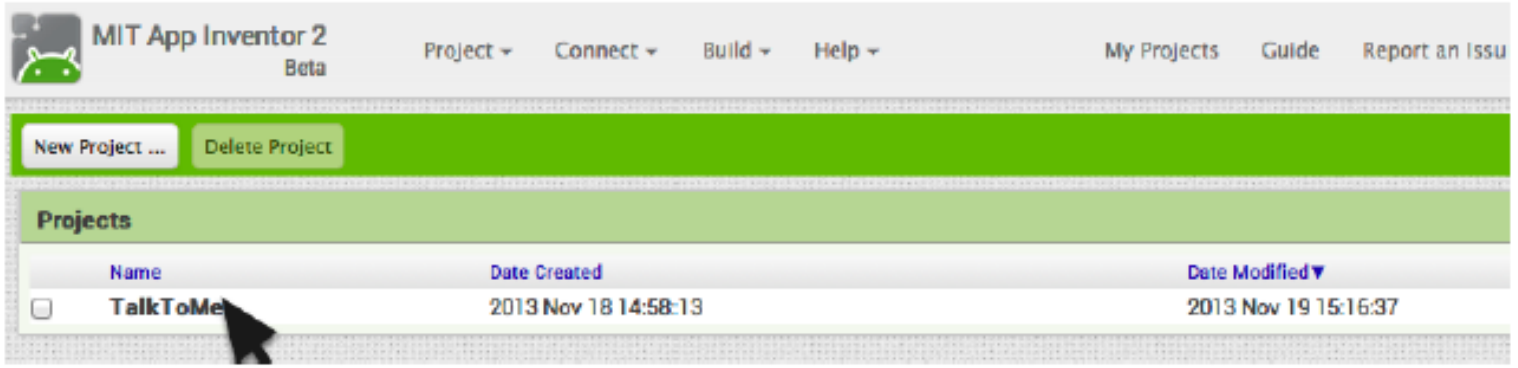
Now move on to TalkToMe Part 2 to make the app respond to shaking and to let users put in whatever phrase they want.

TalkToMe Part 2: Shaking and User Input

This tutorial shows you how to extend the basic TalkToMe app so that it responds to shaking, and so that the user can make the phone say any phrase s/he types in.

Open the "TalkToMe" project that you worked on in the last tutorial.

App Inventor will always open the last project you worked on, so you may automatically be taken into your TalkToMe app.

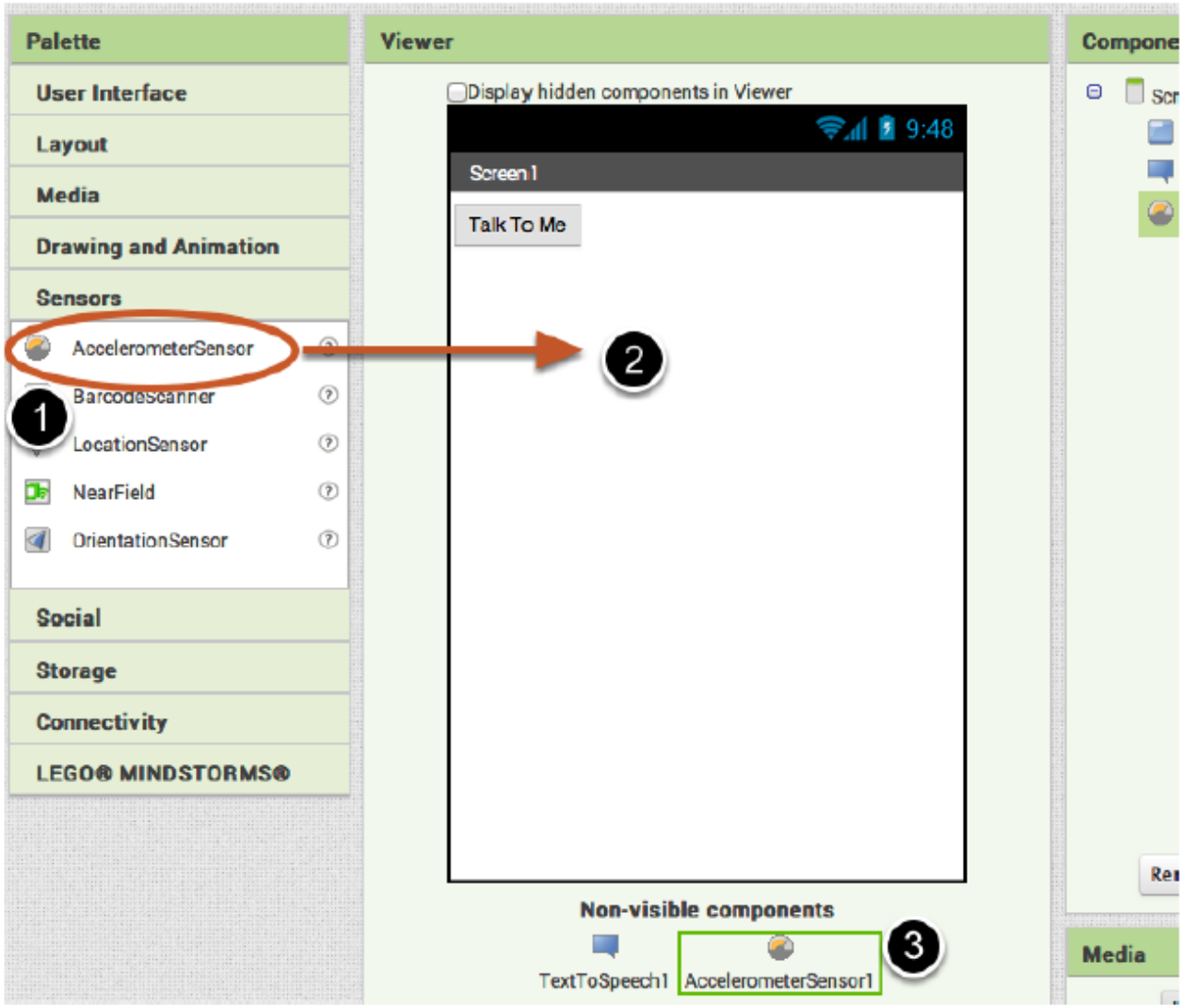


Go to the Designer Tab

Your project may open in the Designer. If it does not, click "Designer" in the upper right.

Add an Accelerometer Sensor

In the **Sensors** drawer, drag out an AccelerometerSensor component and drop it onto the Viewer. (It's a non-visible component, so it drops to the bottom of the screen.) NOTE: emulator users should skip this part and proceed to the next section of this tutorial called "Say Anything". (The emulator can not respond to shaking!)

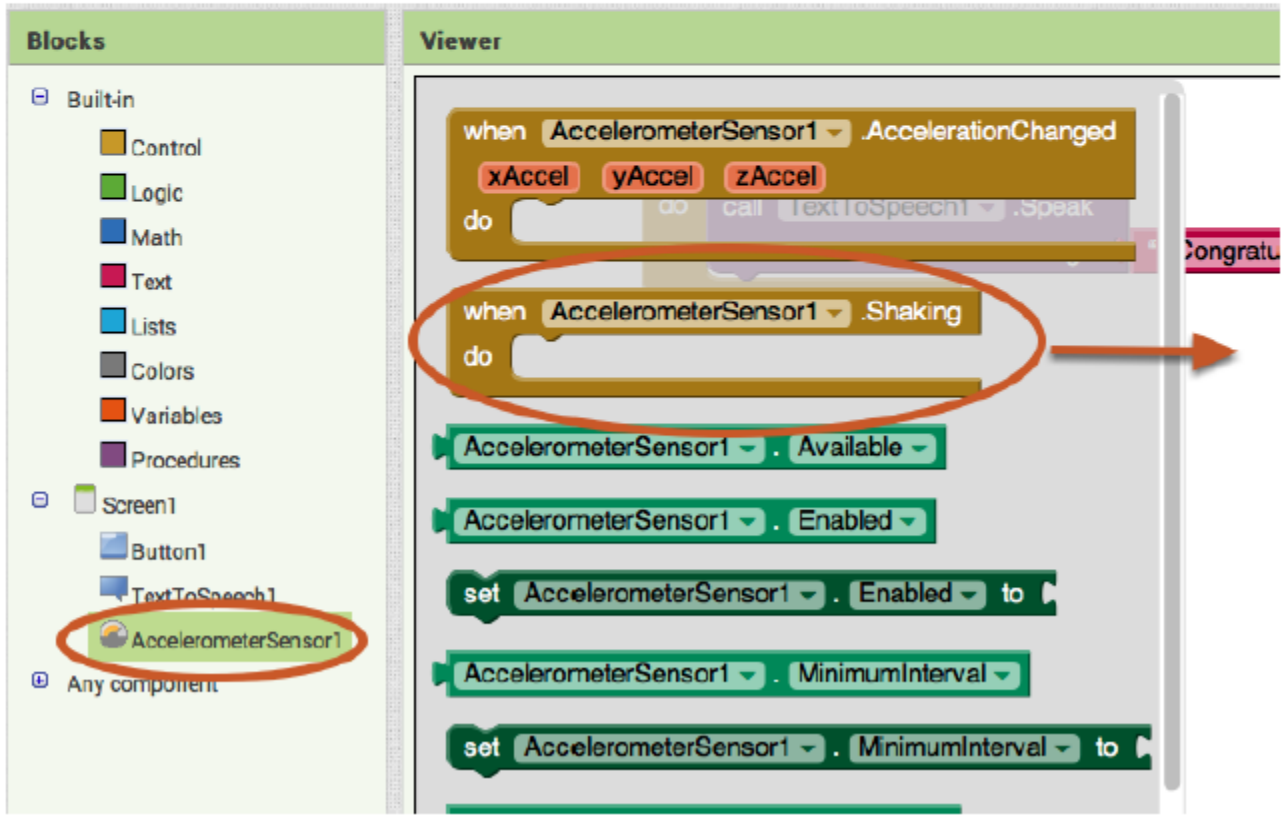


Go to the Blocks Editor

Click "Blocks" to program the new Accelerometer Sensor that you just added.

Program the Accelerometer Shaking event

Click the AccelerometerSensor1 drawer to see its blocks. Drag out the ***when AccelerometerSensor1.Shaking do*** block and drop it on the workspace.

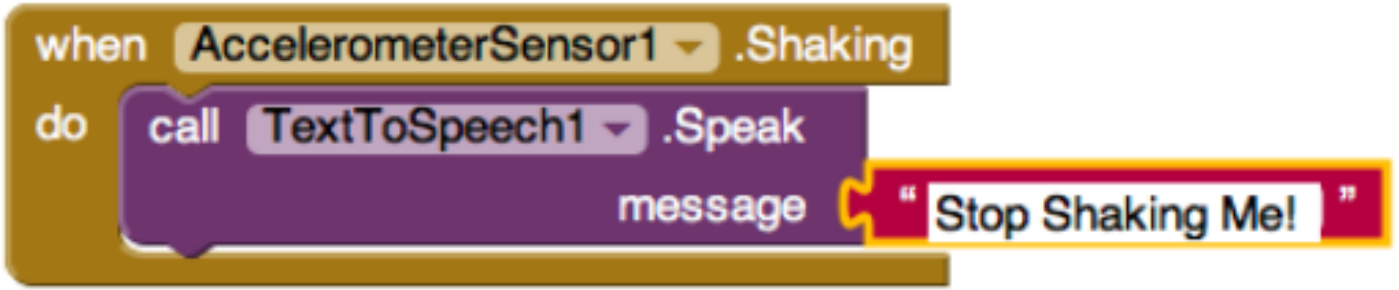


What do we want the app to do when the accelerometer detects shaking?

Copy and paste the blocks that are currently inside the when Button1.Click event handler. You can select the purple block, then hit the key combination on your computer to copy and then to paste. You'll have a second set of blocks to put inside the when Accelerometer.Shaking block.
(Alternatively, you can drag out a new *call TextToSpeech1.Speak block* from the TextToSpeech drawer, and a new pink *text block* from the Text drawer.)

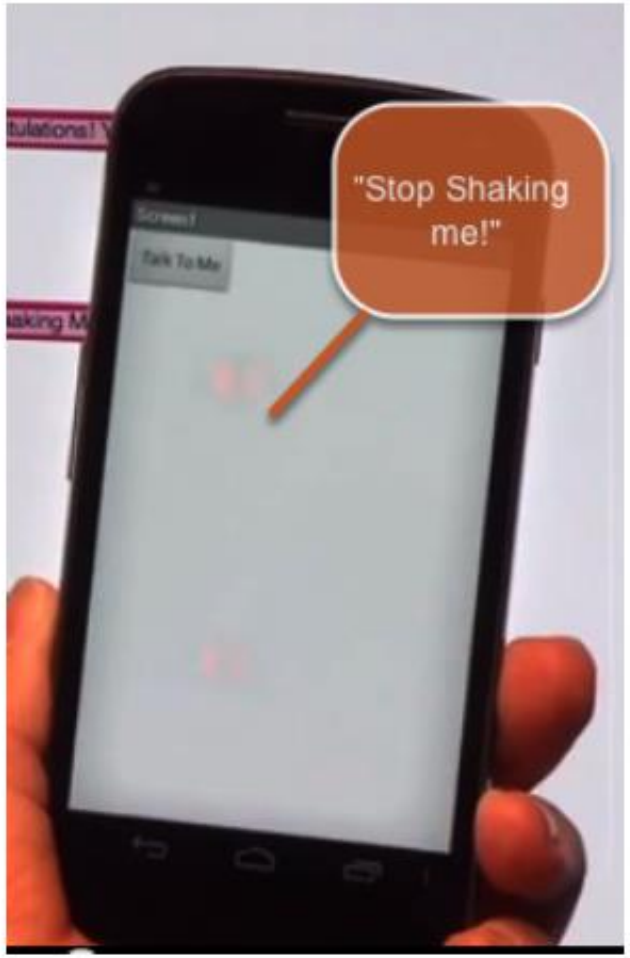
Change the phrase that is spoken when the phone is shaking.

Type in something funny for when the phone responds to shaking.



Test it out!

You can now shake your phone and it should respond by saying "Stop shaking me!" (or whatever phrase you put in.)

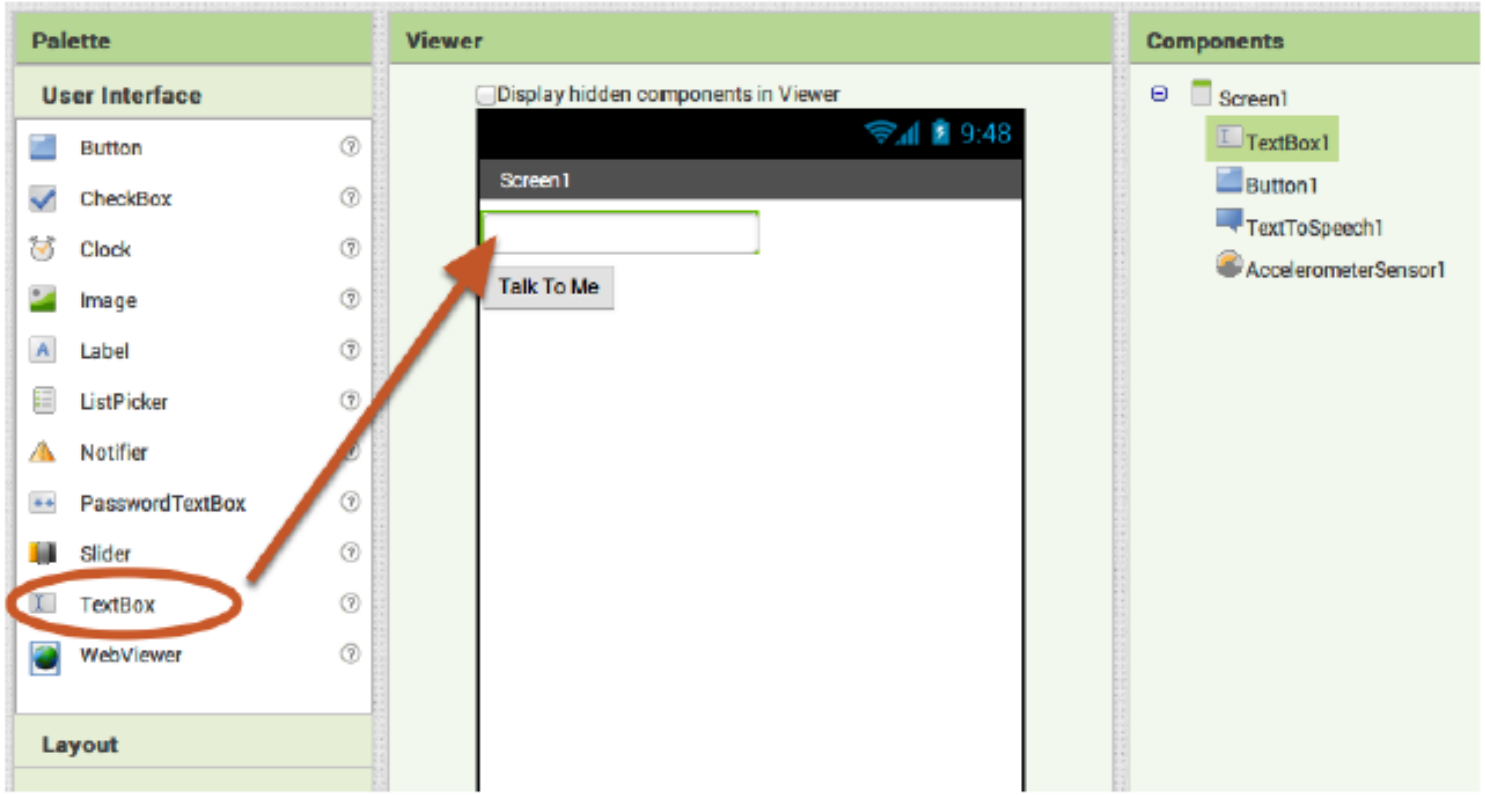


Say Anything

Is your phone talking to you? Cool! Now let's program the button click so that it causes the phone to speak whatever phrase the user put into the text box. Go back to the Designer.

Add a Text Box to your user interface.

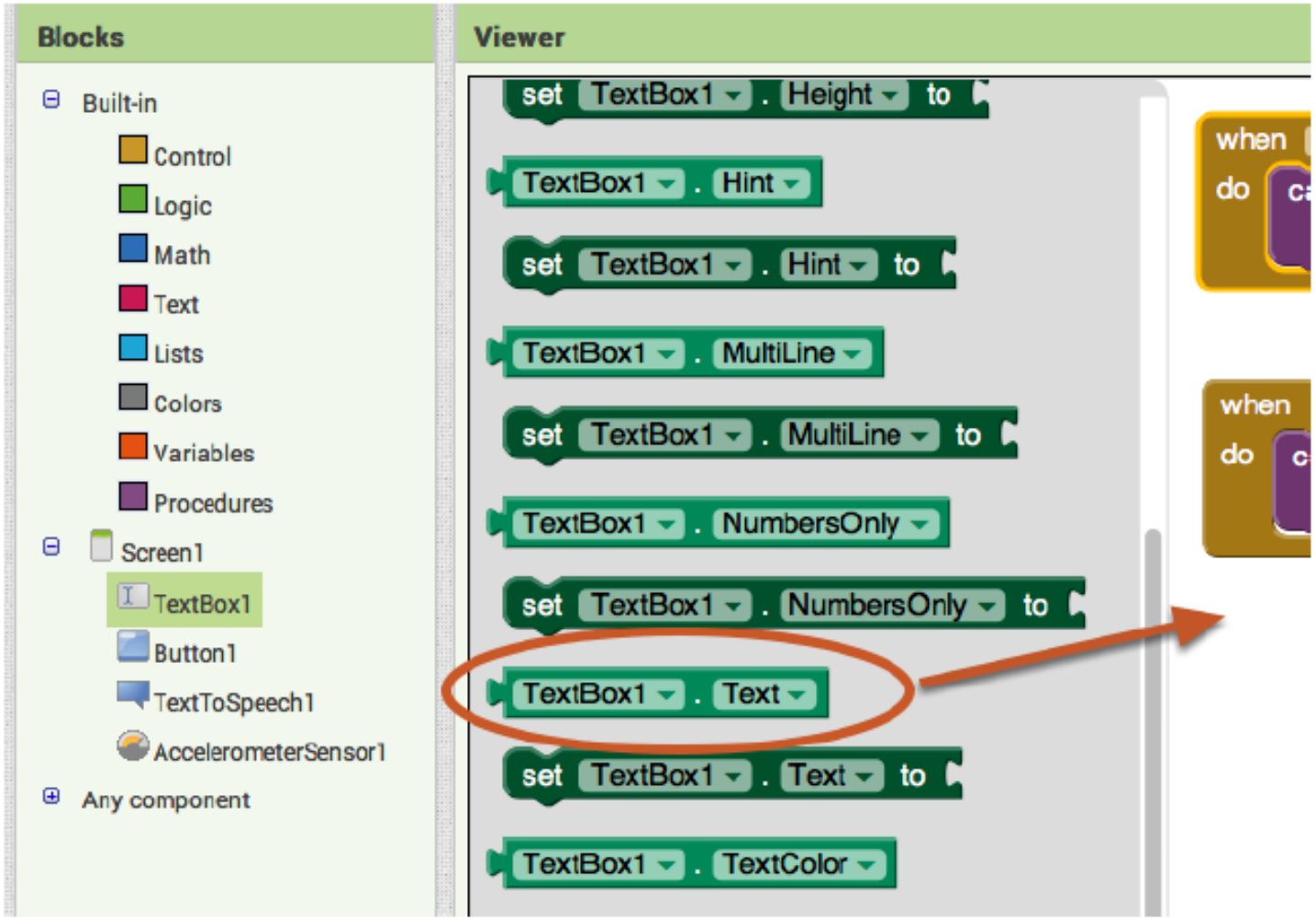
From the User Interface drawer, drag out a TextBox and put it above the Button that is already on the screen.



Back to the Blocks Editor!

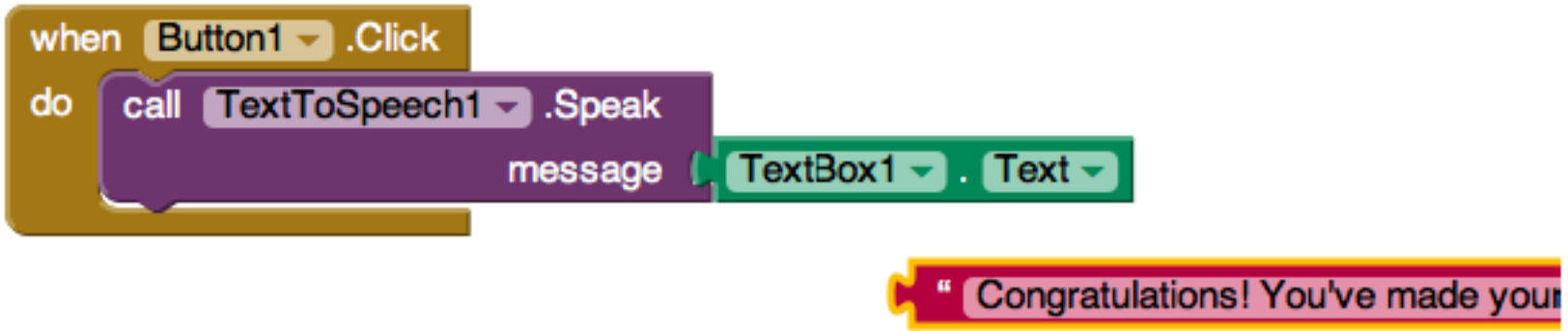
Get the text that is typed into the TextBox.

Get the text property of the TextBox1. The green blocks in the TextBox1 drawer are the "getters" and "setters" for the TextBox1 component. You want your app to speak out loud whatever is currently in the TextBox1 Text property (i.e. whatever is typed into the text box). Drag out the ***TextBox1.Text*** getter block.



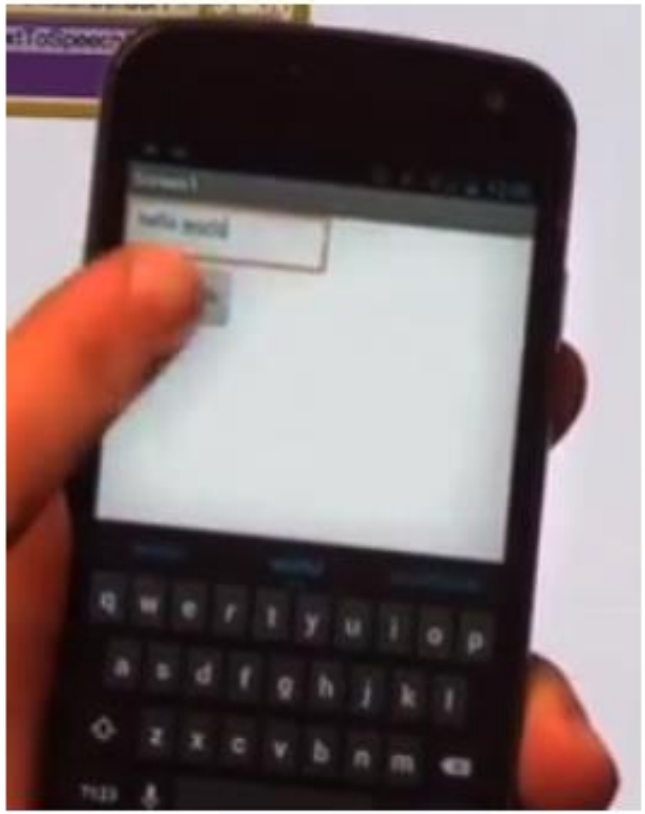
Set the Button Click event to speak the text that is in the Text Box.

Pull out the "congratulations..." text box and plug in the TextBox1.Text block. You can throw the pink text block away by dragging it to the trash in the lower right corner of the workspace.



Test your app!

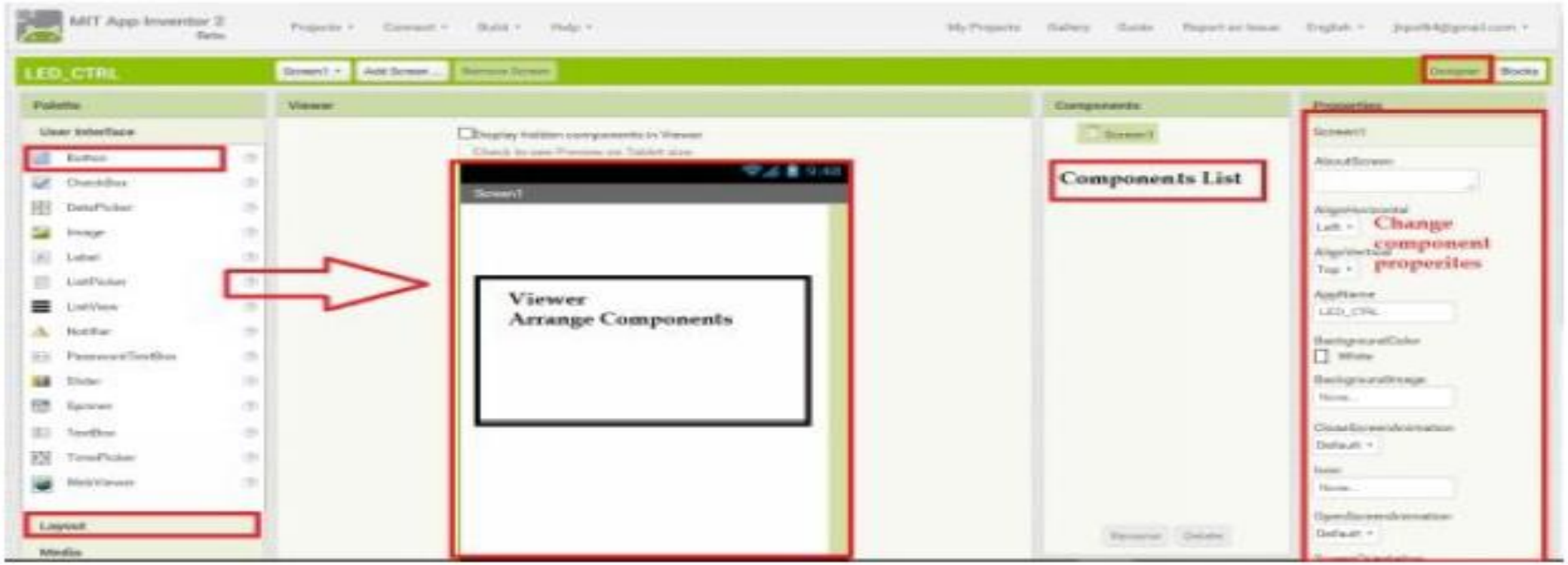
Now your app has two behaviors: When the button is clicked, it will speak out loud whatever words are currently in the Text Box on the screen. (if nothing is there, it will say nothing.)
The app will also say "Stop Shaking Me" when the phone is shaken.



Turn On and OFF an LED (Hello World)

- Name your new project as LED_CTRL or whatever name you would like for your first app and click ok to enter into the main screen.

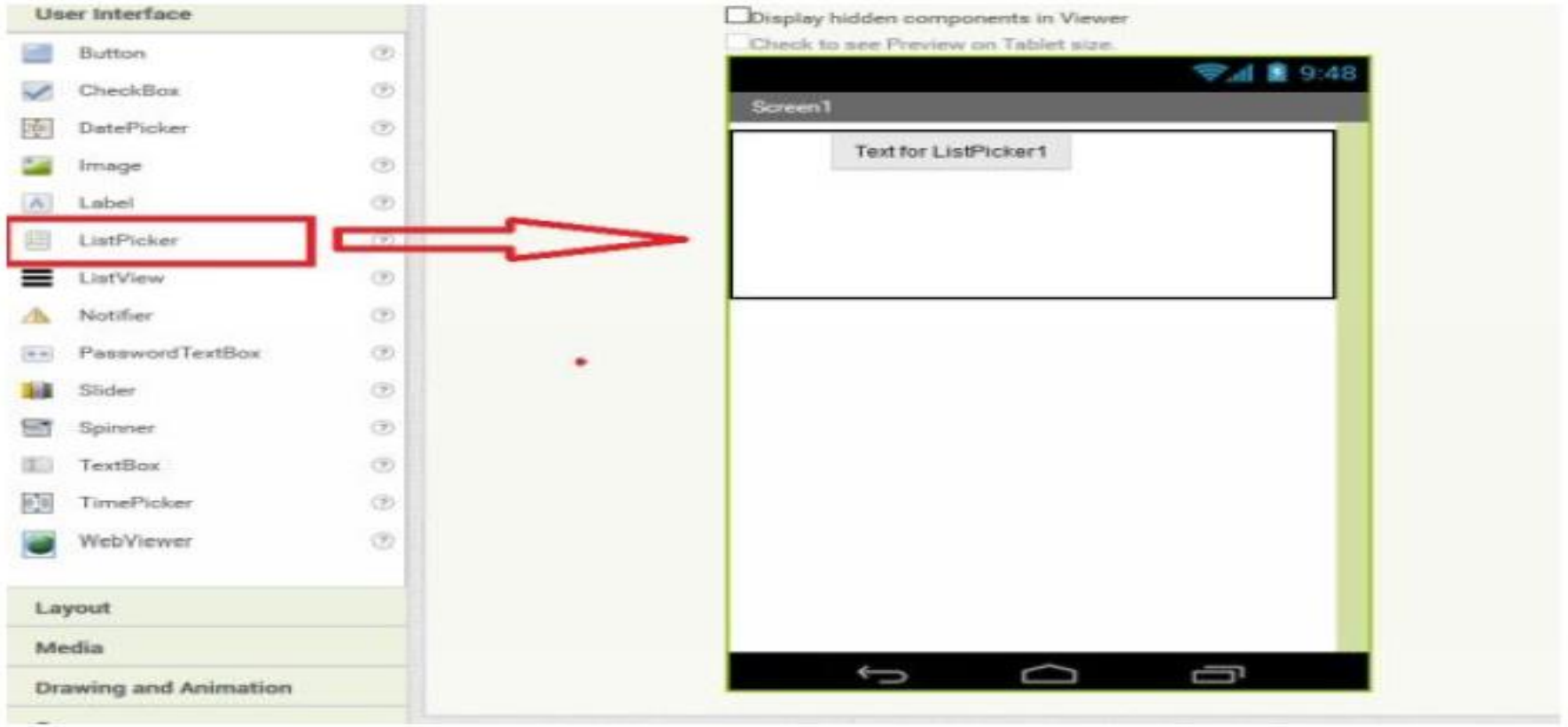
- When you click ok after naming your app, you will be moved to a main screen to make your app, the screen that appears on your window is called as Designer, where you need to design user interface like buttons, widget, text pane, label this window act as a controller and there is also another window where you need to program the graphical codes to make your function to work.



- On the left side you can see palette, the palette which contains interface, layout, Media, Drawing and Animation, Sensors, Social, Connectivity etc.,
- To create your user interface on the main screen you need to pick buttons and your required stuffs from the palette.
- Click Layout on the left side below palette, Pick Table Arrangement and drop into the Viewer.

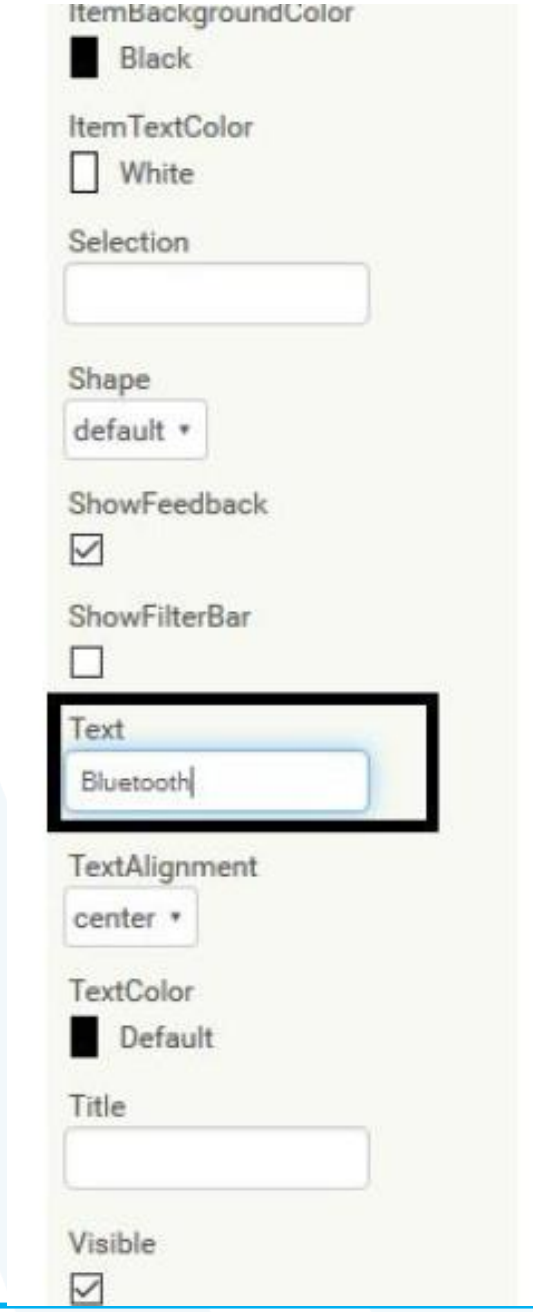
- On the left side of the table properties, change columns and rows to 3 and also change the width to fill parent from automatic.

- Pick Listpicker from User Interface and drop into first row in the Viewer screen. This used to connect to your Bluetooth device. When you click this button it will display available Bluetooth devices.

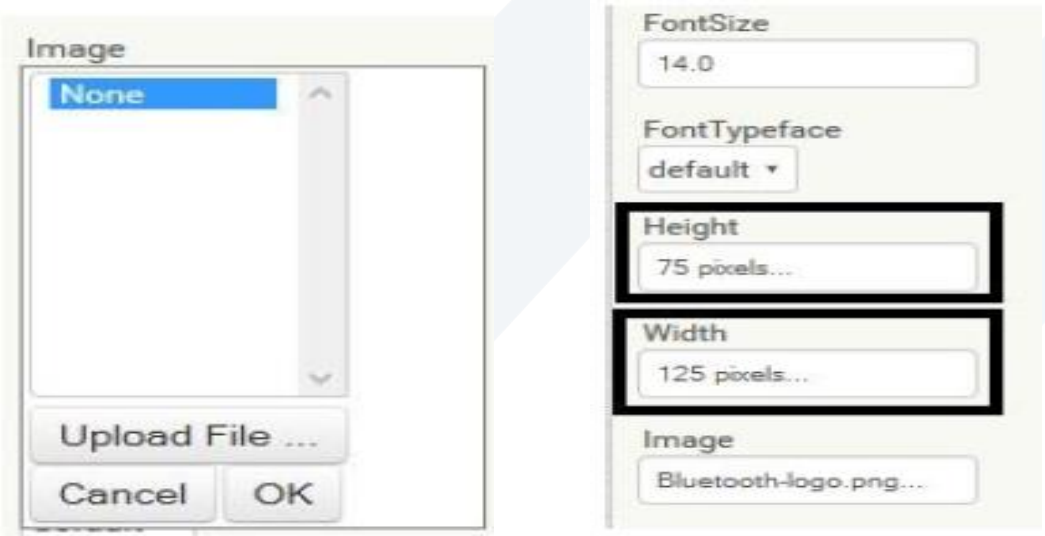


- Select ListPicker from the Components List

- Go to list picker properties and change the text to Bluetooth and you can also upload an image file to the list picker.

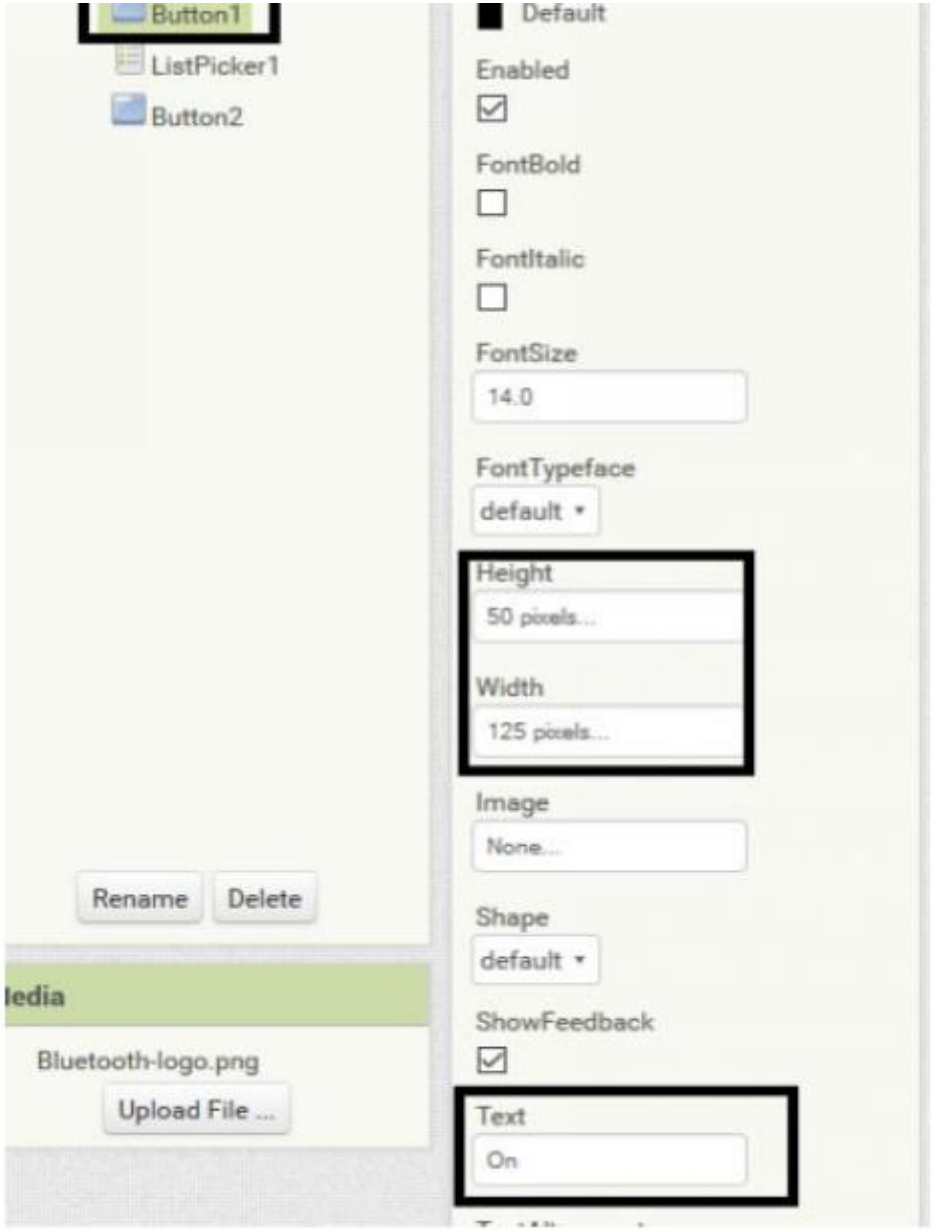


- Click on the None tab below Image and upload your Bluetooth picture to set it to Listpicker.
- Click the upload file and browse to the image you want to use for the ListPicker, if the Viewer Screen displays the Image for the ListPicker in larger size you need to alter the height and width of the ListPicker by going to the properties for the Listpicker.



- Next pick a button and drop into second row of the Viewer screen and add another button to the same row.

- Click on the first button and go the button properties and change the name of the button to ON do the same for the second button by changing the name of the button to OFF.

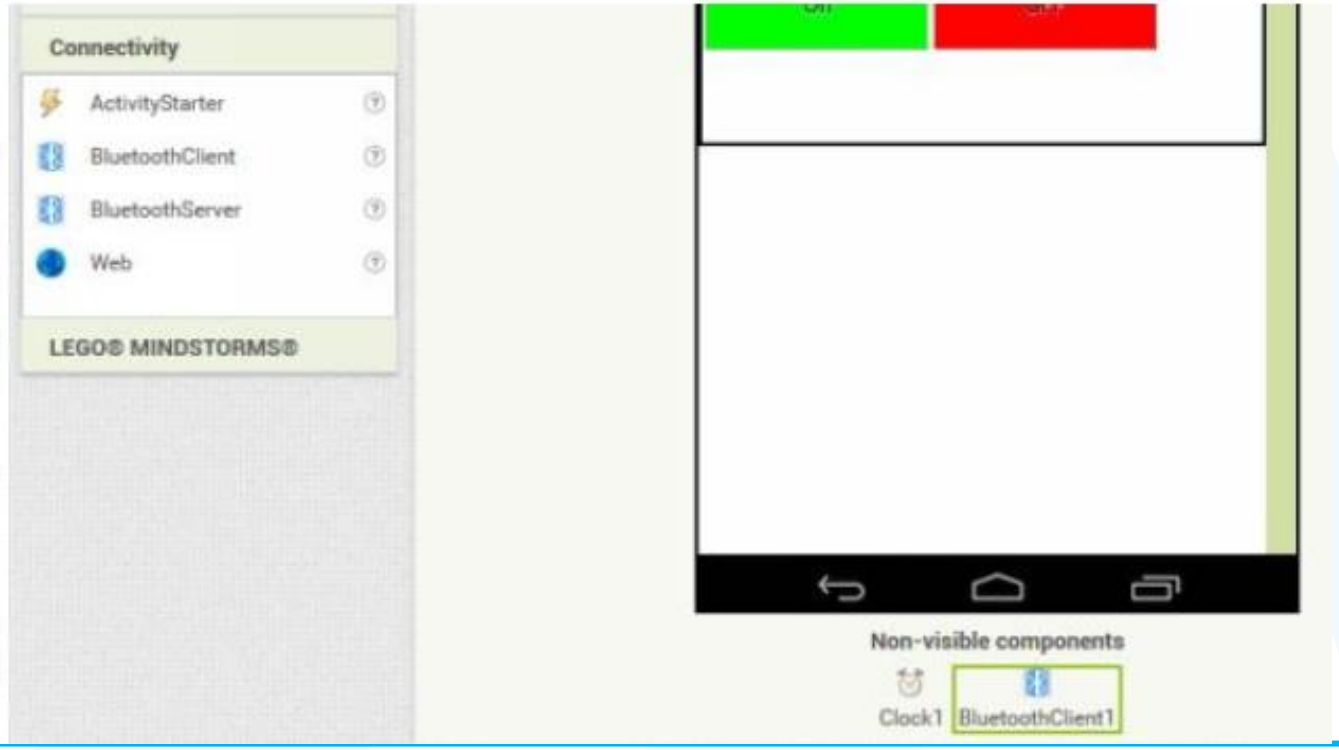


- Change button color by changing the background color from properties of the appropriate button.

- Pick and place Label from user interface and place into the 3rd row of Viewer Screen. Go t properties and remove the Text label and leave this as empty, this label will later on used for indicating whether our app has connected to an Bluetooth or not.

- We have almost completed User Interface for our first Android app, we also need some Non-visible components to make this app, the first one is the clock which can pick and placed from the sensor palette.
- It provides the instant in time using the internal clock on the phone. It can fire a timer at regularly set intervals and perform time calculations, manipulations, and conversions. We are using clock to cause activities to occur at a preset interval.

- Last step in this tutorial pick Bluetooth client form the connectivity palette and place it inside the Viewer screen. The Bluetooth client component which allows us to connect to other device and make us to send and receive data's between Bluetooth devices.



- We have completed the user interface design for the first app, now this design requires program to run behind this and do certain functions, let's move to programming.
- Click on the blocks to get into programming mode


LED_CTRL
Built in blocks
Blocks associated with user interface blocks
pick and place blocks here and arrange it like a puzzle to complete program
Pick and place the blocks into dustbin will delete the part of code

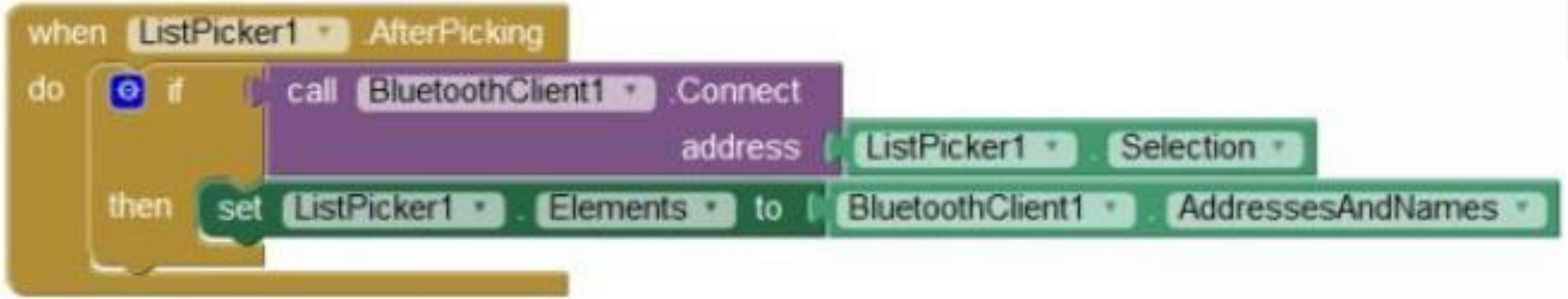
- Click ListPicker1 and select when.listpicker1.BeforePicking block and place into programming window.

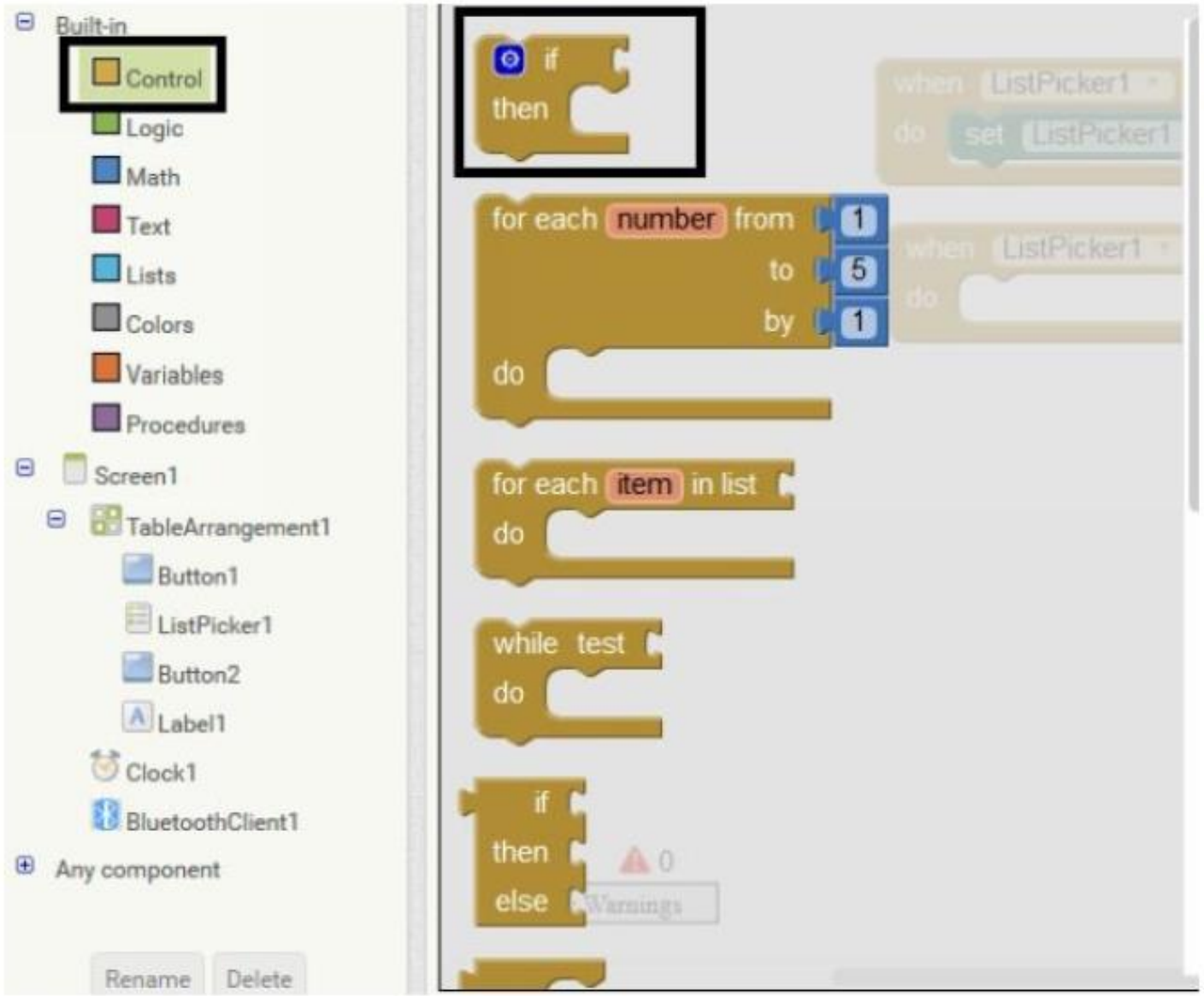
- After placing the first block and scroll down in the list picker blocks and select set.Listpicker1.element to

- Go to Bluetooth client and pick Bluetooth client address and names and complete the first block.

- The first block which explains, when you click the list picker button, it should display the available Bluetooth devices that are already connected to your device previously.

- Now complete the second block by going to the different components and pick and place the required block into programming window.

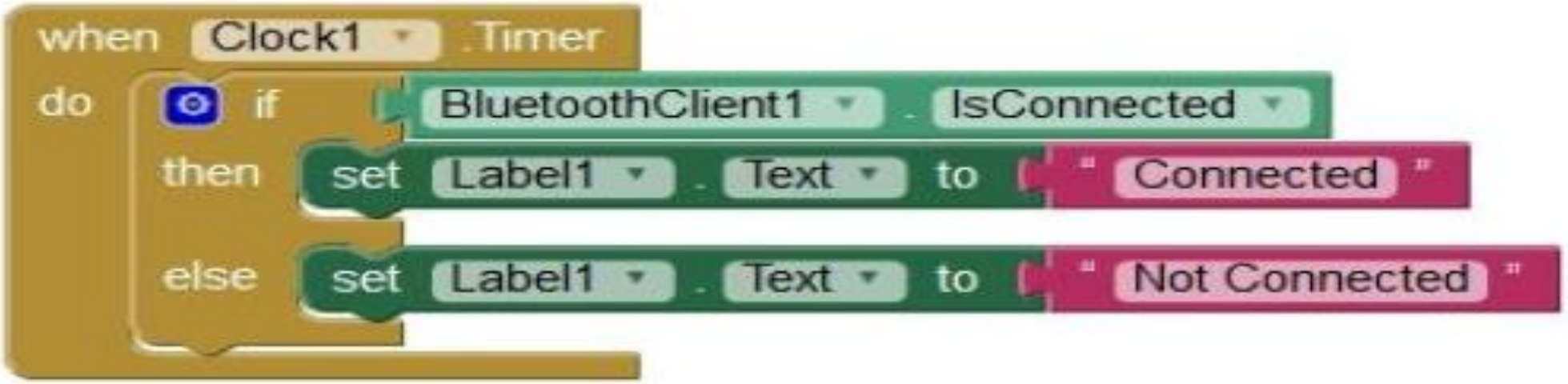


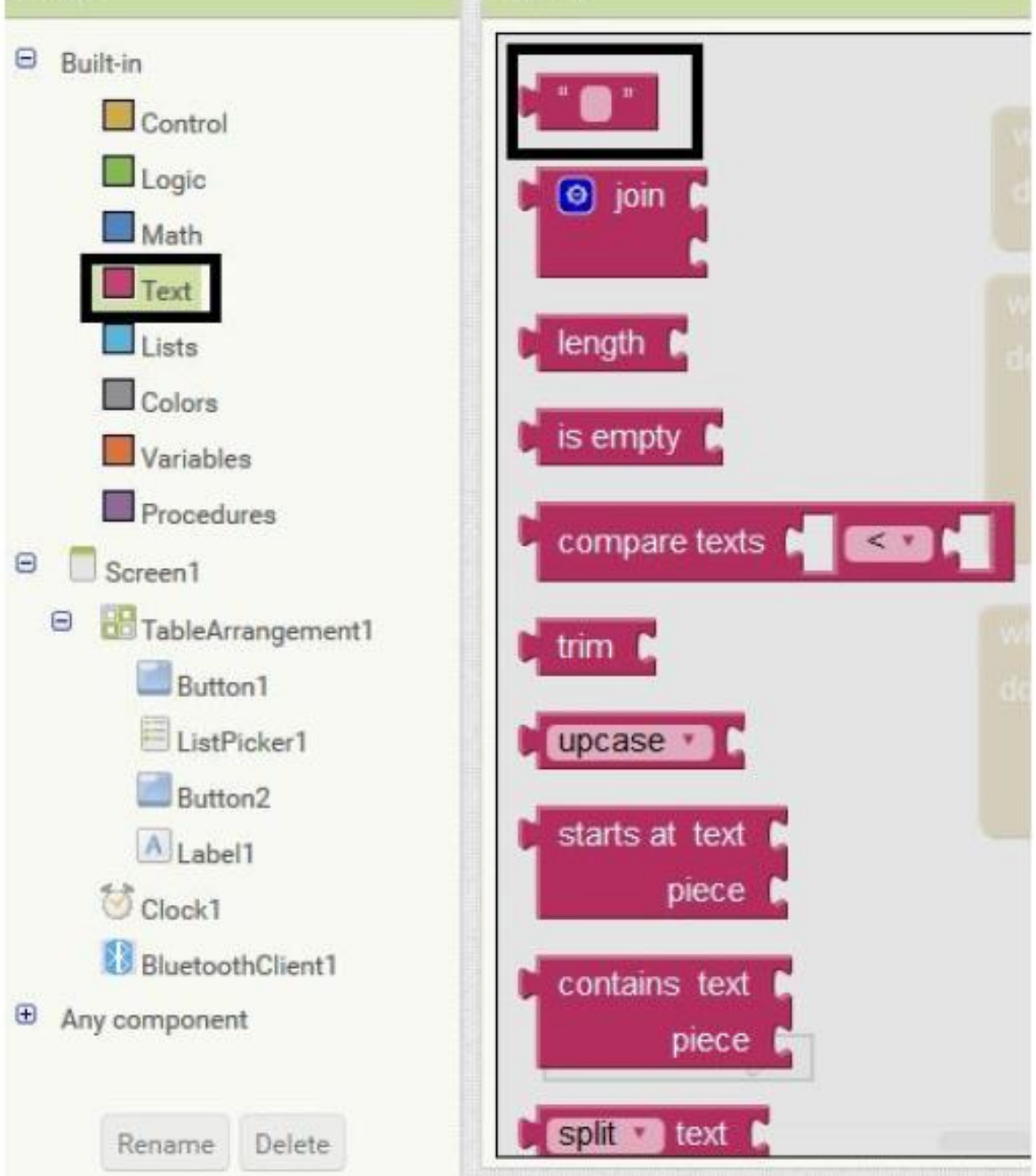


- If then block can be pick from the built in functions.

- Second block which explains, when you pick a Bluetooth device from the available list the current device that you picked should be used as a current device and Bluetooth address will be stored and the app will communicate with the selected device.

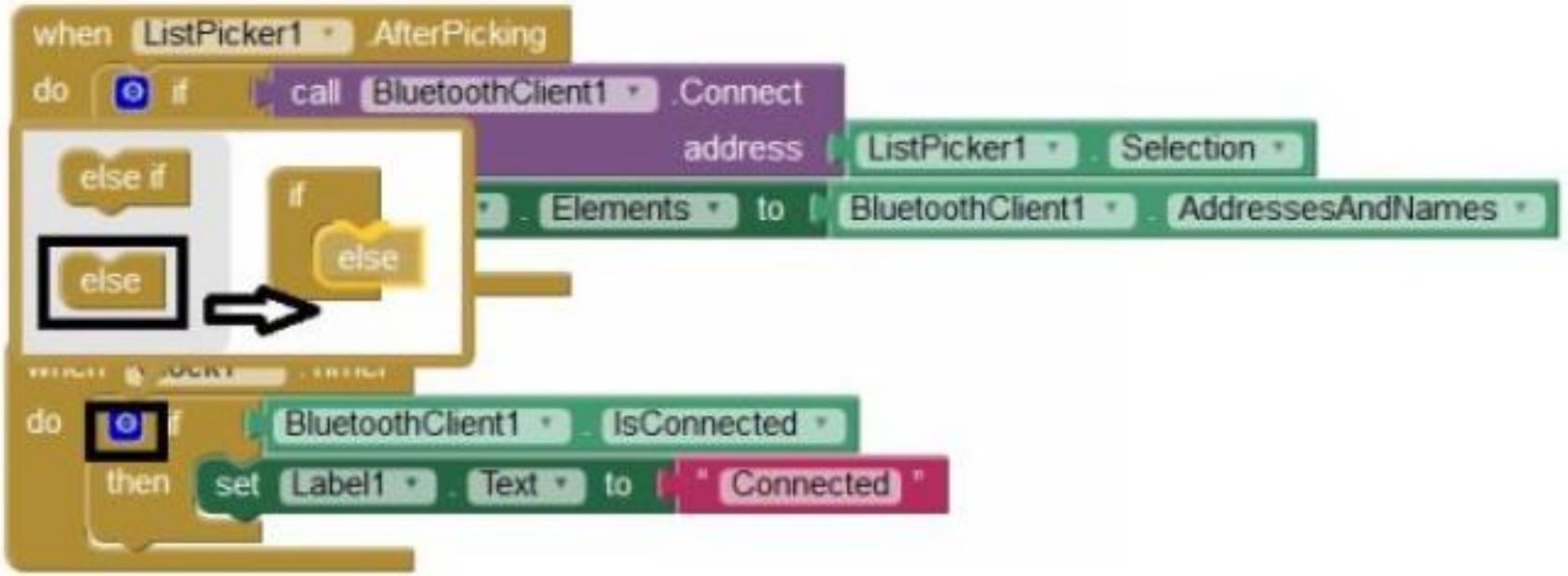
- The third block which is placed inside a timer, this block will assign Connected text to Text Label on the third row if the Bluetooth device is connected if the device is not connected Text label will display message as Not Connected.



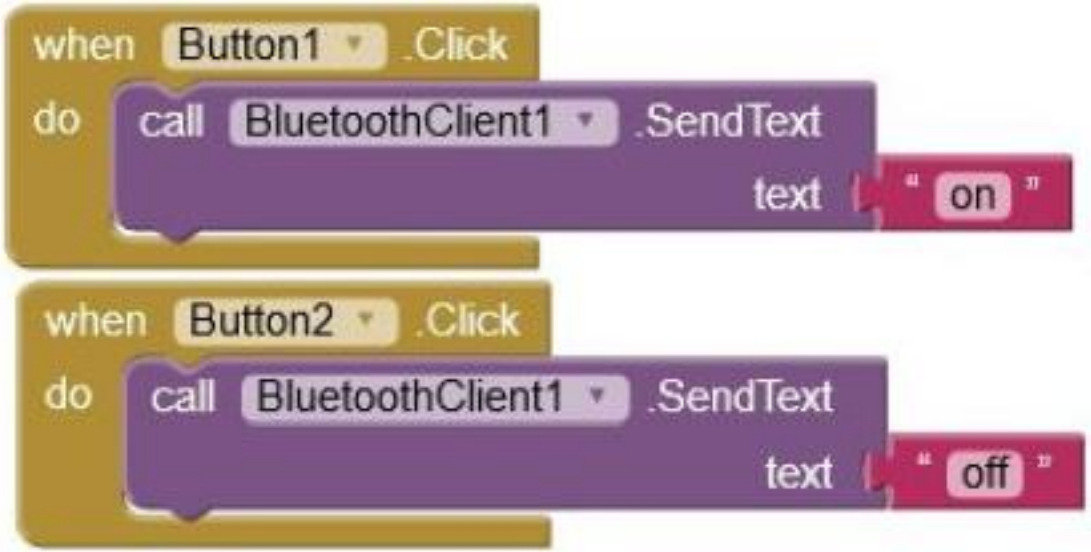


- Text string block can be added from Text in built function

- We have already seen how to add If then block from controls, in the above program for screen shot we have added If then else block, which is not available in the controls, w need to place usual if then block and click on the blue icon on the if then block and place the else function to the block.

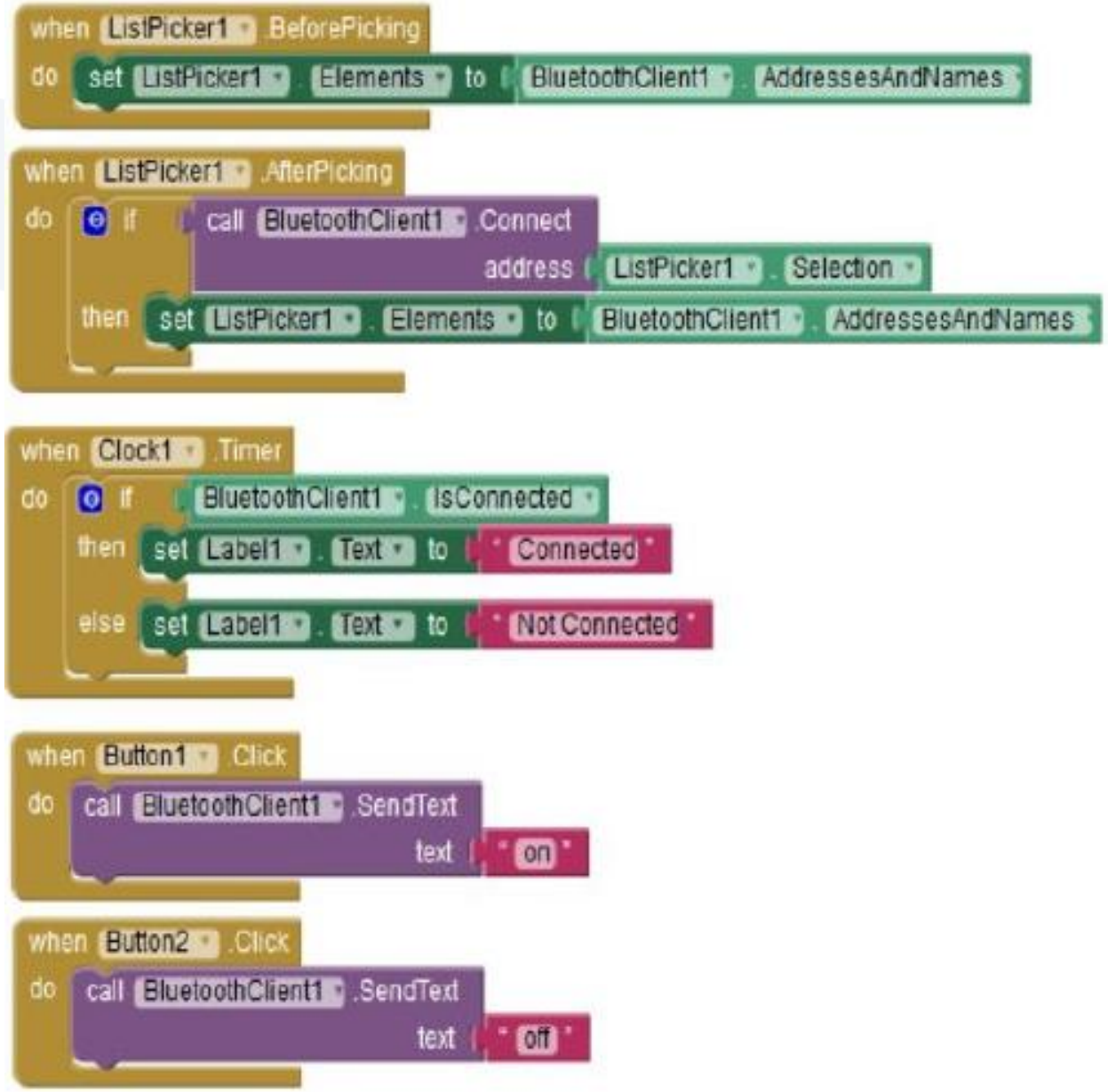


- You need to create a control task for the two button we created in the Viewer screen.



- When you press the first button it will send “on” and for the second button it is “off”. We connect to a Bluetooth device attached to Arduino, when Arduino receive these two string we have to program based on these string and make a decision to turn On and Off an LED.

Complete block for the program to control an LED connected to an Arduino



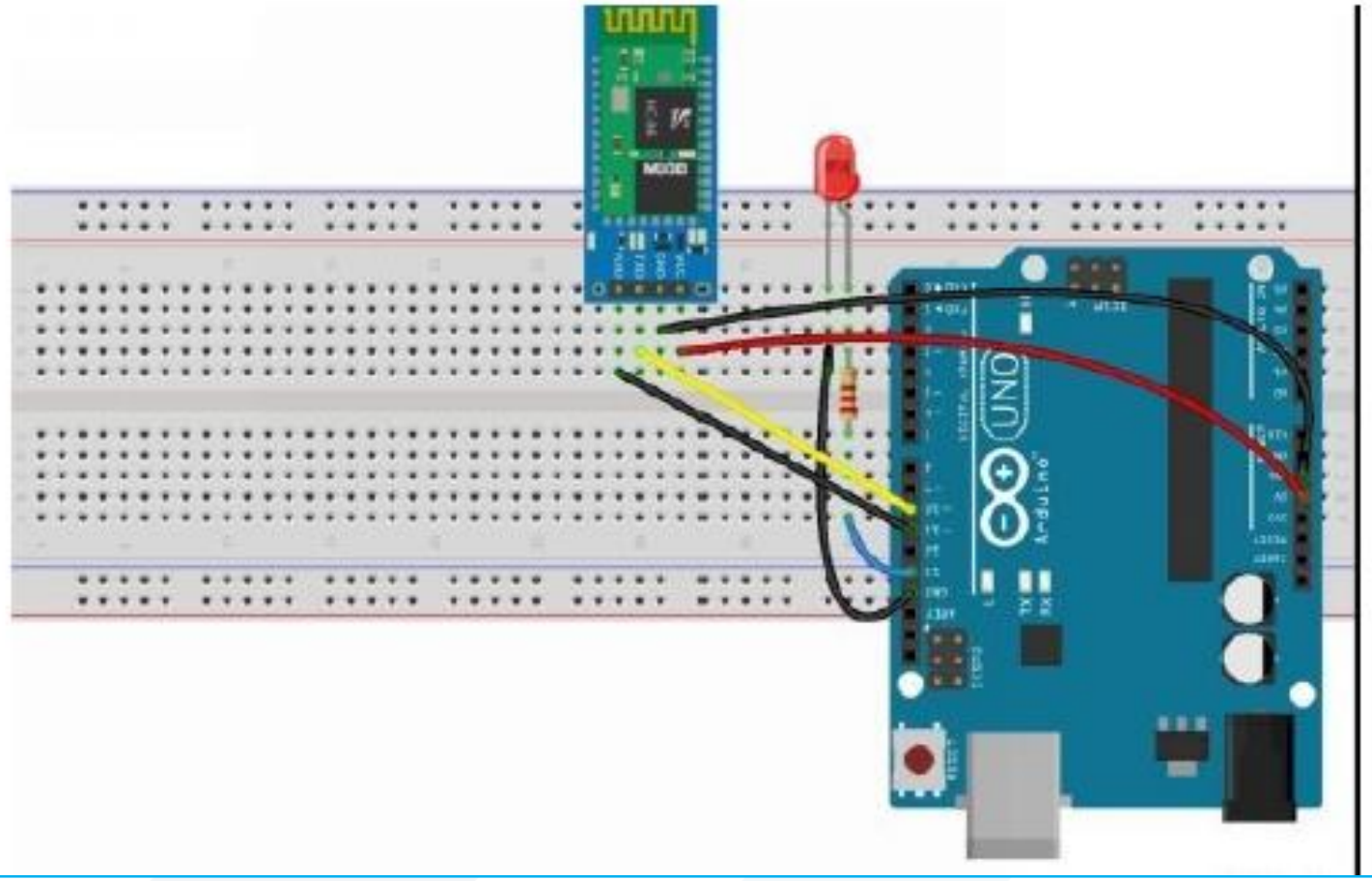
Creating Android App

Convert your block into android app. Click build and select app (save .apk to my computer).

- You need to add HC-05 or HC-06 or whatever model you are using to your android device, connect your Bluetooth device VCC to 5v and Gnd to Ground of Arduino, go to your Bluetooth and turn on your Bluetooth and search for devices, connect to your Bluetooth device using default password, most Bluetooth devices will be ‘1234’ or ‘0000’.

جَامعة
المَنارة
MANARA UNIVERSITY

Connecting Bluetooth and programming Arduino



- Connect your LED to 13th pin of Arduino, if you don’t want to connect an LED still you can do this project, by default an LED is connected to the 13th pin of Arduino.
- Connect TX of Bluetooth to 10th pin and RX of Bluetooth to 11th pin. You can also connect it to the available TX and RX pins that is 0 and 1, when you connect to these two pins you cannot connect your Arduino to your PC or Laptop.

```
#include <SoftwareSerial.h>
SoftwareSerial BT(10, 11);       //TX, RX respectively
String state;       // string to store incoming message from bluetooth
void setup() {
BT.begin(9600);       // bluetooth serial communication will take place on pin 10 and 11
Serial.begin(9600);       // serial communication to check the data on serial monitor
pinMode(13, OUTPUT);    // LED connected to 13th pin
}
void loop() {
while (BT.available()){            //Check if there is an available byte to read
delay(10);                 //Delay added to make thing stable
char c = BT.read();       //Conduct a serial read
state += c;                 //build the string- either “On” or “off”
}
if (state.length() > 0) {
Serial.println(state);
if(state == “on”)              // if the received string is on, turn on led connected to the pin 13
{digitalWrite(13, HIGH); }
else if(state == “off”)        // if the received string is off, turn off led connected to the pin 13
{digitalWrite(13, LOW); }
state =””;}}                  //Reset the variable
```