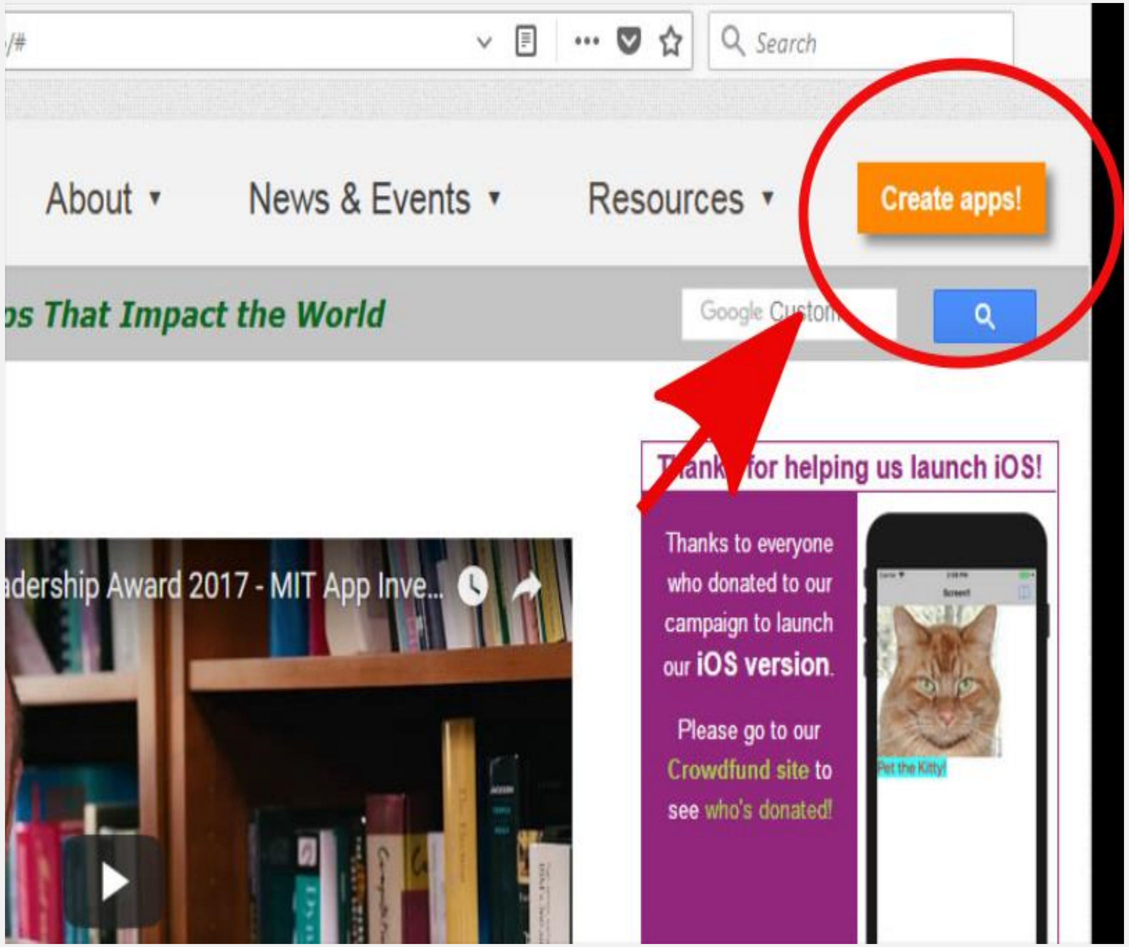
MIT APP INVENTOR - Interfacing Arduino

INTRODUCTION

- App Inventor is a cloud-based tool, by which you can build android and iOS applications using web browser.
- MIT App Inventor website offers all the support you’ll need to learn how to build your own apps.
- *Let’s build an app!*



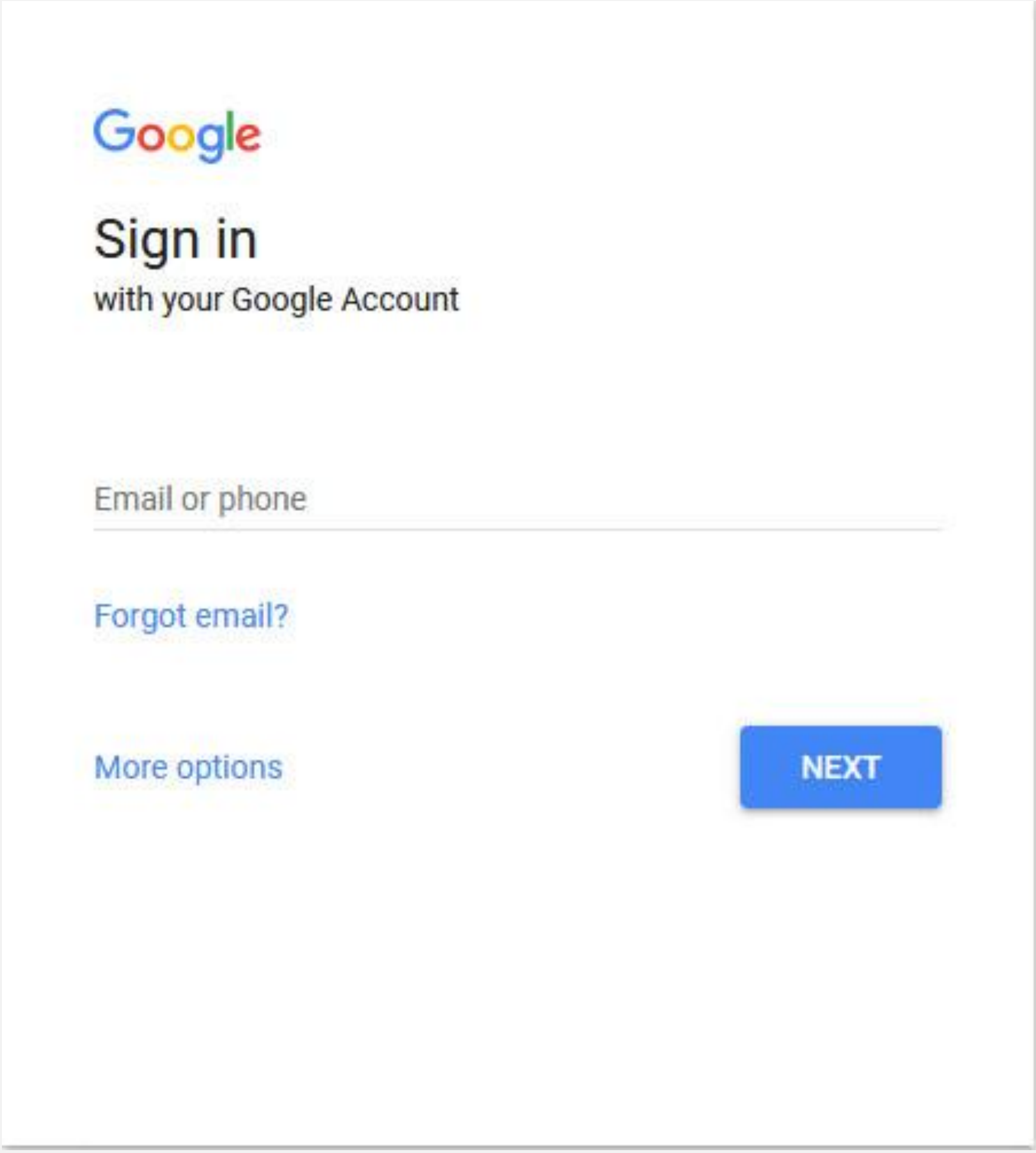
Thanks for helping us launch iOS!

Thanks to everyone who donated to our campaign to launch our iOS version.

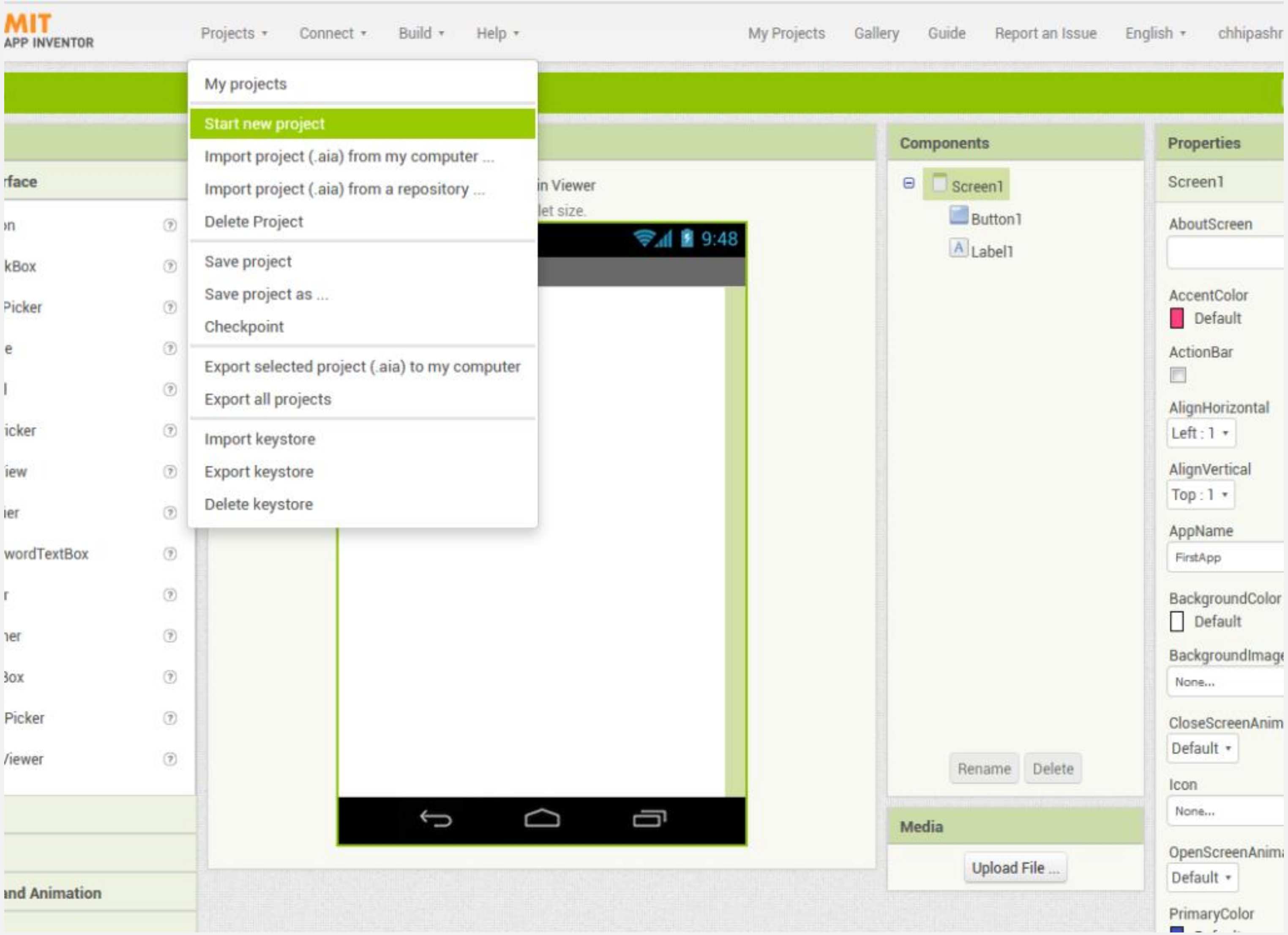
Please go to our Crowdfund site to see who's donated!

www.ai2.appinventor.mit.edu.

By clicking “Create Apps!” button from any page on this website you can get MIT app login page.



Log in to App Inventor using an existing Google account.



AboutScreen

AccentColor
🟥 Default

ActionBar
□

AlignHorizontal
Left : 1 ▾

AlignVertical
Top : 1 ▾

AppName
FirstApp

BackgroundColor
□ Default

BackgroundImage
None...

CloseScreenAnim
Default ▾

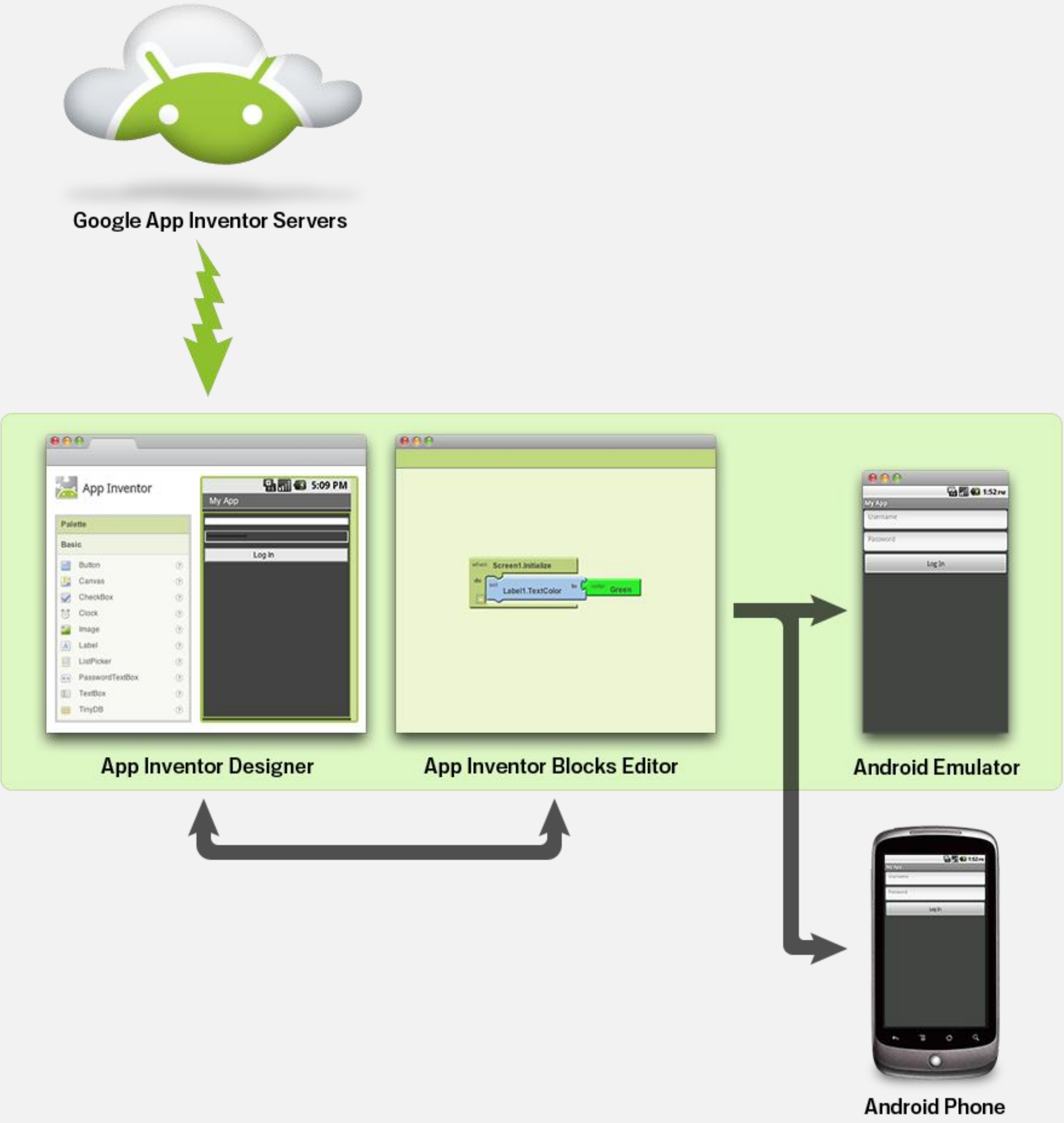
Icon
None...

OpenScreenAnim
Default ▾

PrimaryColor

To start making your application, click on the Projects menu on the top left of the screen and select the “Start new project” and then give the name of your project.

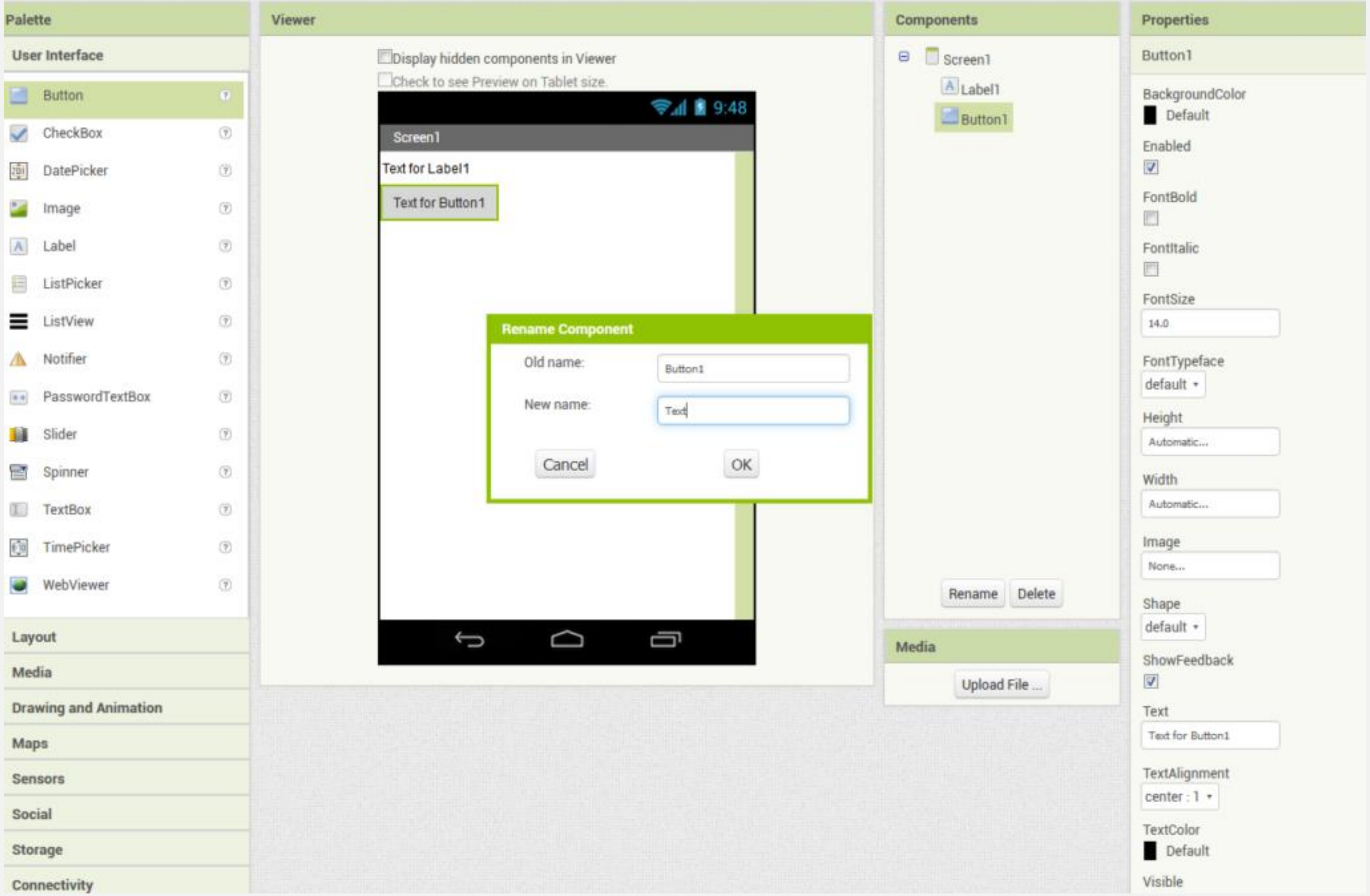
How it all works?



You build apps by working with:

- **Designer** - select the components for your app.
- **Blocks Editor** - assemble program blocks that specify how the components should behave. You assemble programs visually, fitting pieces together like pieces of a puzzle.

- The app appears on the phone step-by-step as you add pieces to it, so you can test your work as you build.
- Once the app is complete, we can package the app and produce a stand-alone application to install.
- The App Inventor development environment is supported for Mac OS X, GNU/Linux, and Windows operating systems and Android.



□ Display hidden components in Viewer

□ Check to see Preview on Tablet size.

Rename Component

Old name: Button1

New name: Text

Button1

BackgroundColor: Default

Enabled: ☑

FontBold: □

FontItalic: □

FontSize: 14.0

FontTypeface: default

Height: Automatic...

Width: Automatic...

Image: None...

Shape: default

ShowFeedback: ☑

Text: Text for Button1

TextAlignment: center : 1

TextColor: Default

Visible

- The designer panel appears with five palette:
  - **USER INTERFACE PALETTE**

    We choose things for the user interface things like Buttons, Images, Text boxes to the palette where we can layout the “user interfaces” of our app.
  - **VIEWER PALETTE**

    In viewer palette we will be able to arrange the outlook of our app.

- **COMPONENTS PALETTE**

Component palette displays all the components placed in the app in an order

- **MEDIA PALETTE**

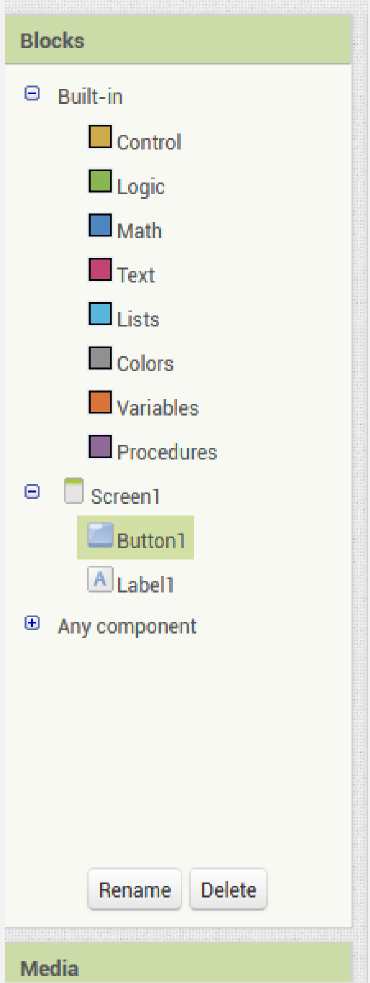
Media palette is used to insert any external media to the app.

- **PROPERTIES PALETTE**

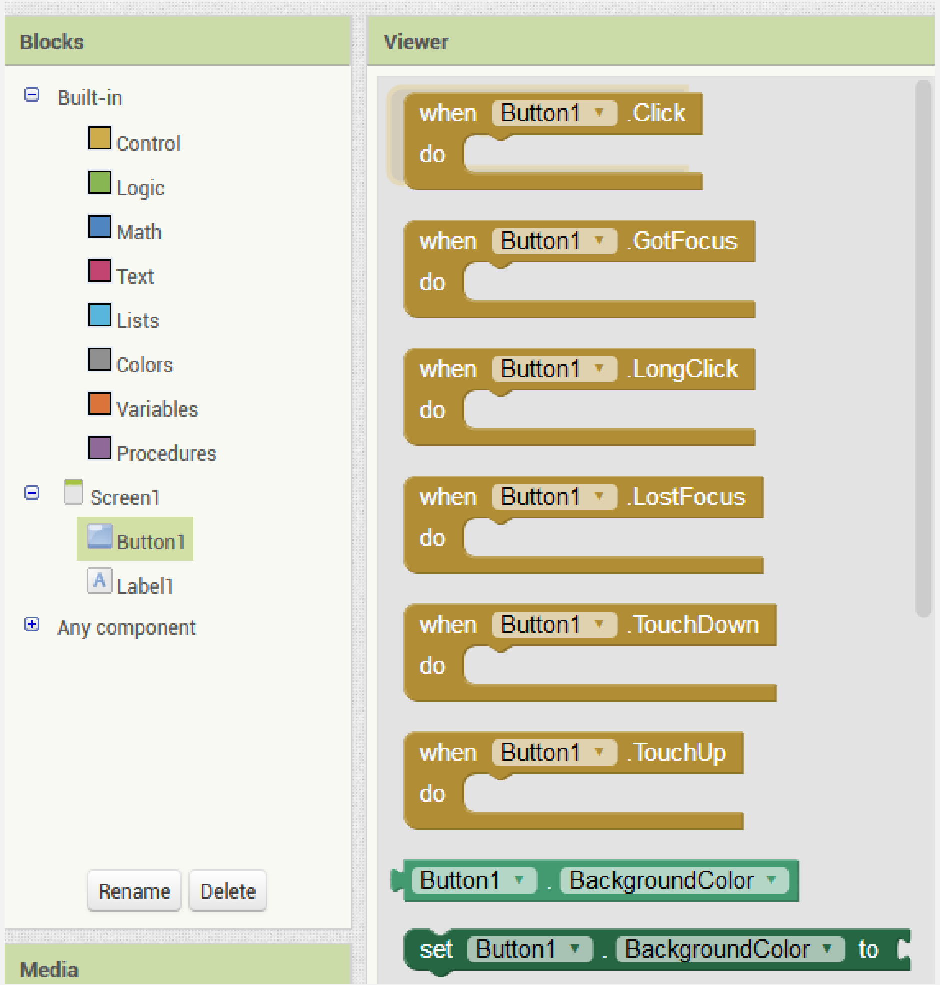
We can change the properties of the components like height, width, text, color to the properties panel.

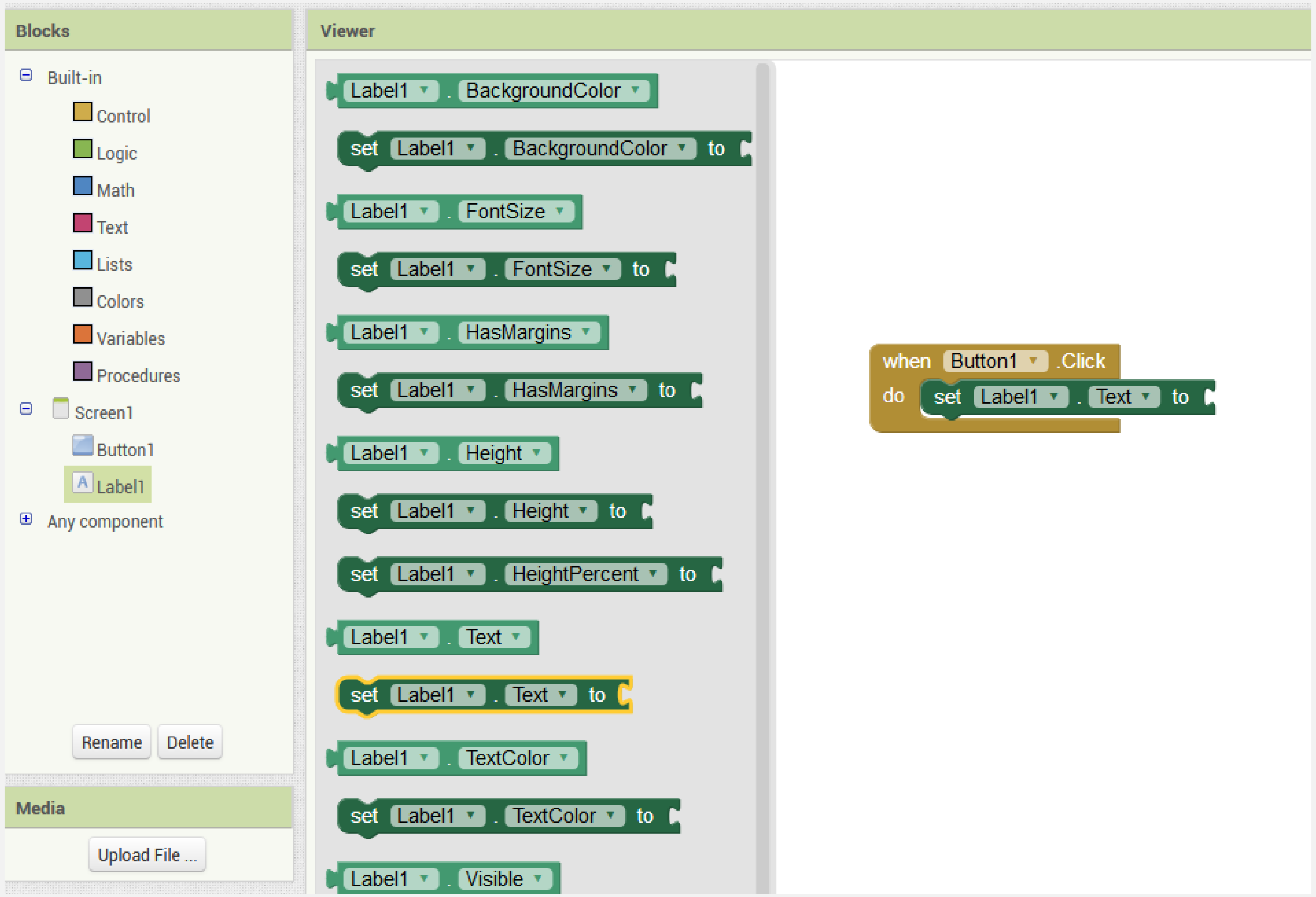
BLOCK EDITOR

- 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 with each components you have added.

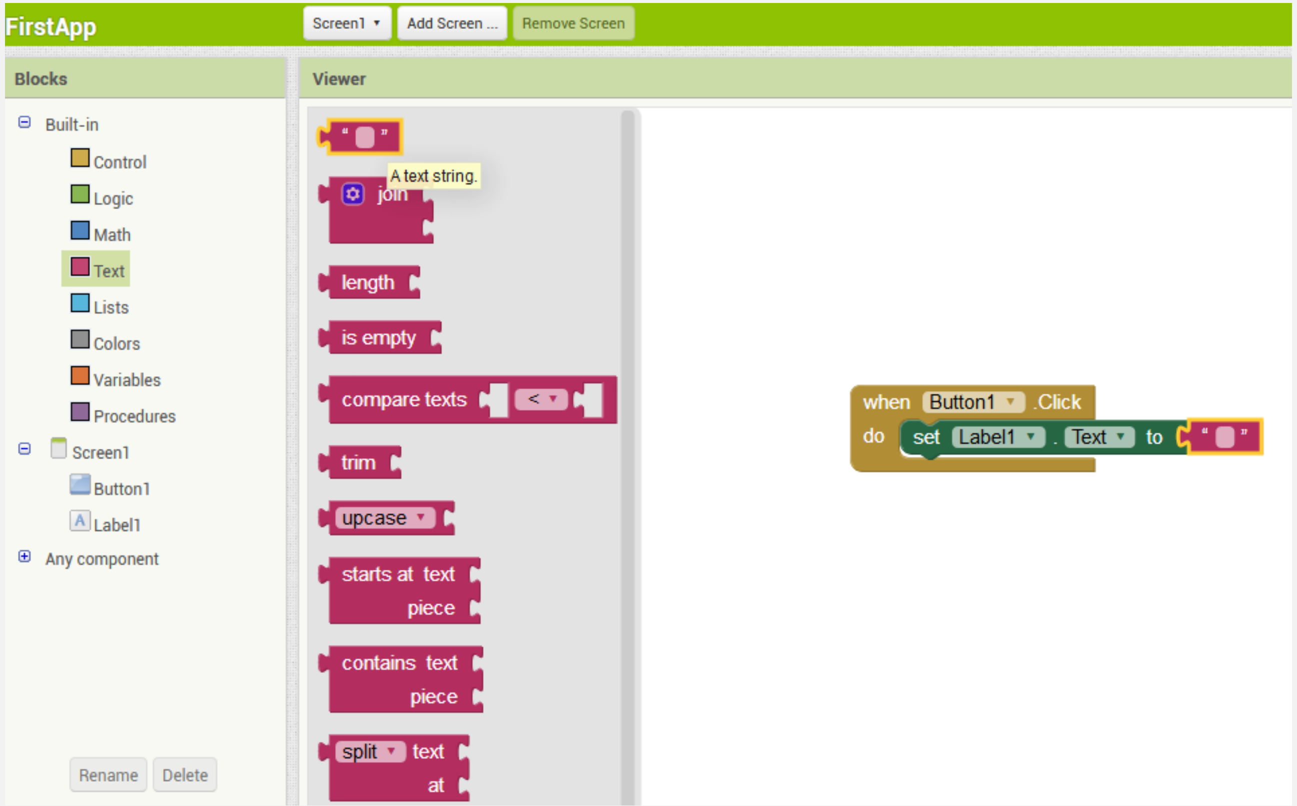


- Click on the Button1 drawer.
- Click and hold the "when Button1.Click do" block.
- Drag it over to the workspace and drop it there.
- This is the block that will handle what happens when the button on your app is clicked.
- It is called an "Event Handler".

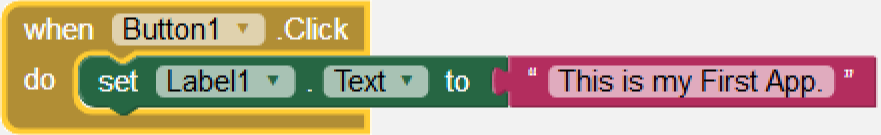




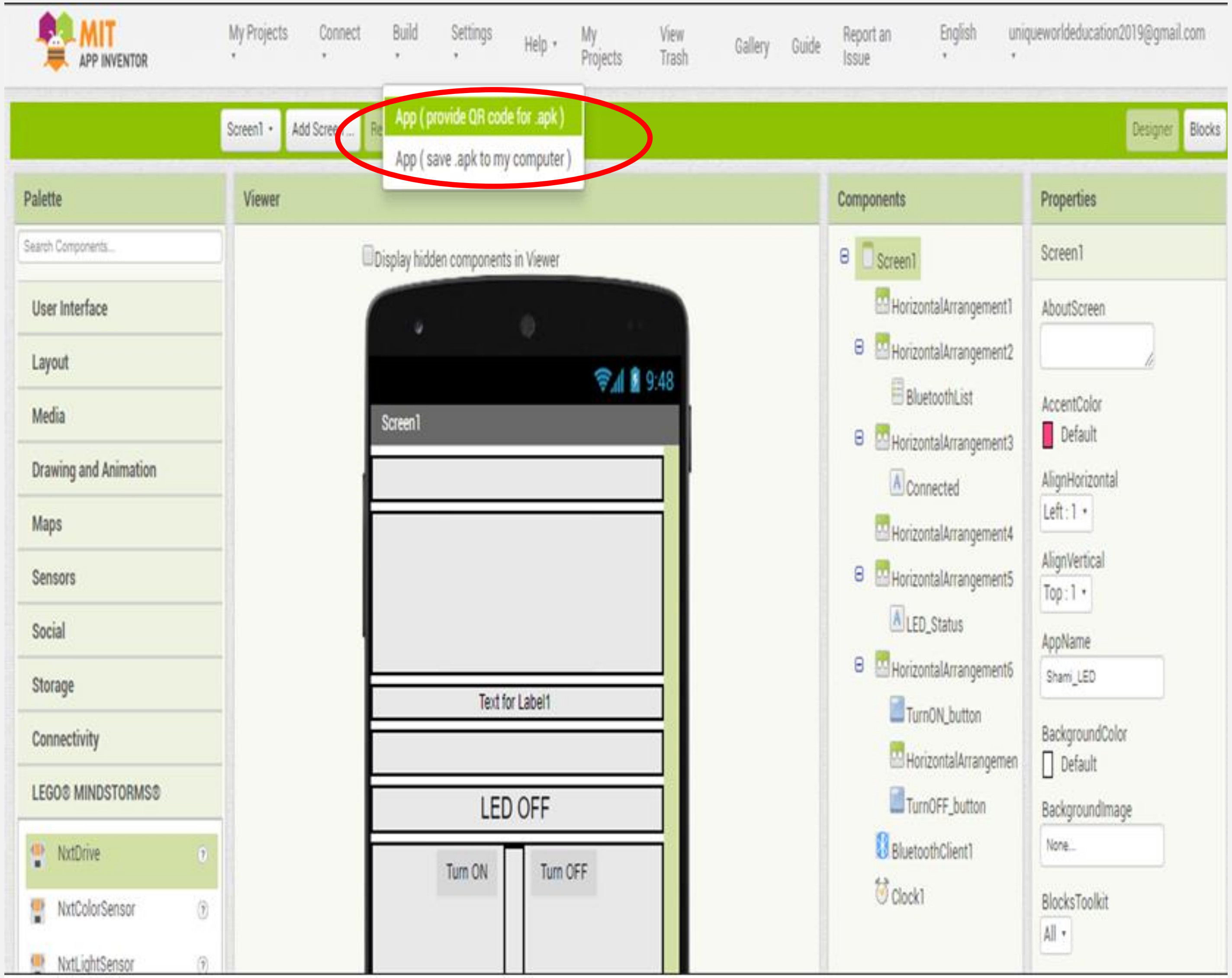
- Now click on the Label1.
- Click and hold the “set Label1 text to” block.
- Drag it inside the button click, it will run when the button is pressed.



- At last, click on the text drawer, drag out a text block and plug it into the socket labelled to and write anything that you want to display.
- Click on the text block and write anything.



```
when Button1 .Click
do  set Label1 . Text to " This is my First App. "
```



AboutScreen

AccentColor: Default

AlignHorizontal: Left : 1

AlignVertical: Top : 1

AppName: Shami_LED

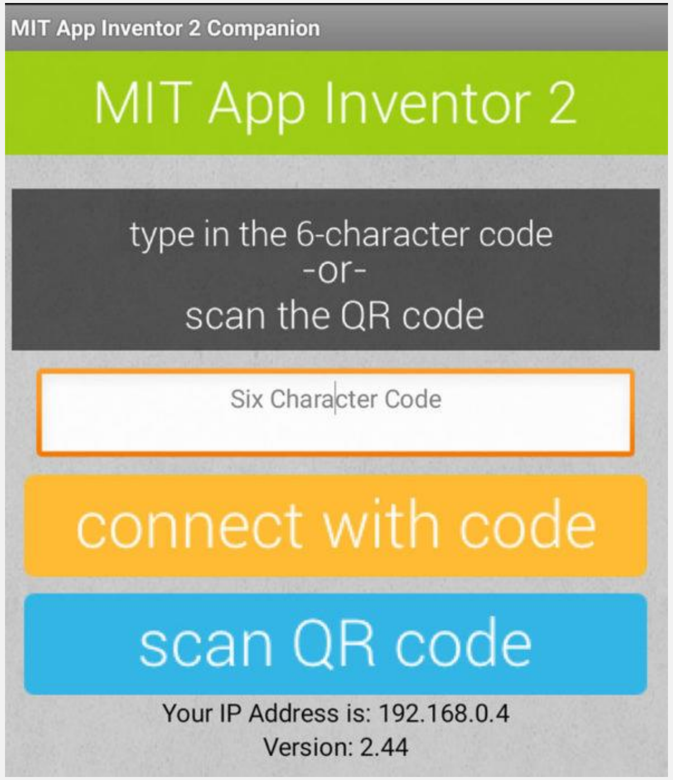
BackgroundColor: Default

BackgroundImage: None...

BlocksToolkit: All

- Once we program the behaviour of our app we can build our app.

  We can choose to provide QR code for .apk file of the app which can be installed in our android phone.



MIT App Inventor 2 Companion

MIT App Inventor 2

type in the 6-character code
-or-
scan the QR code

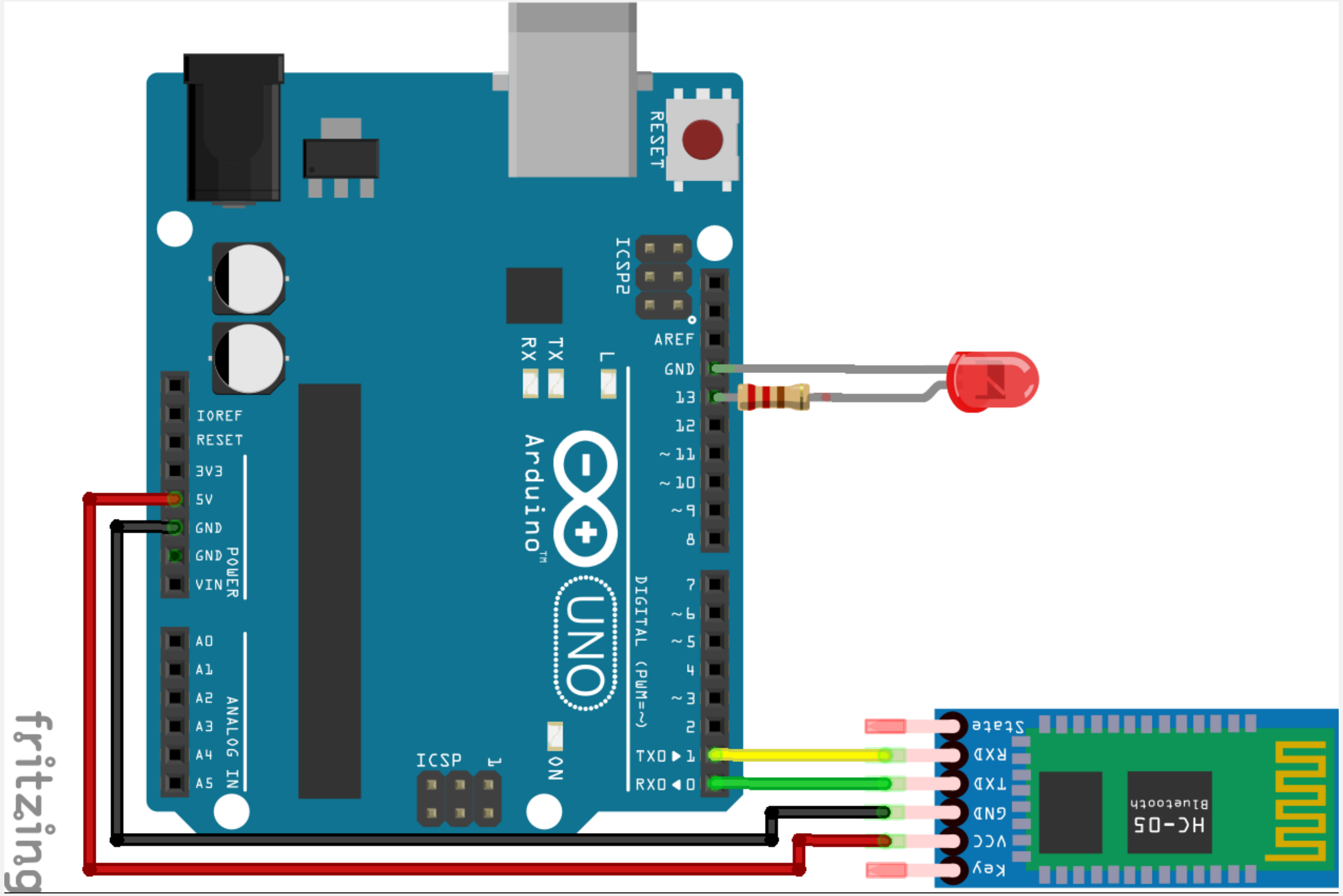
Your IP Address is: 192.168.0.4
Version: 2.44

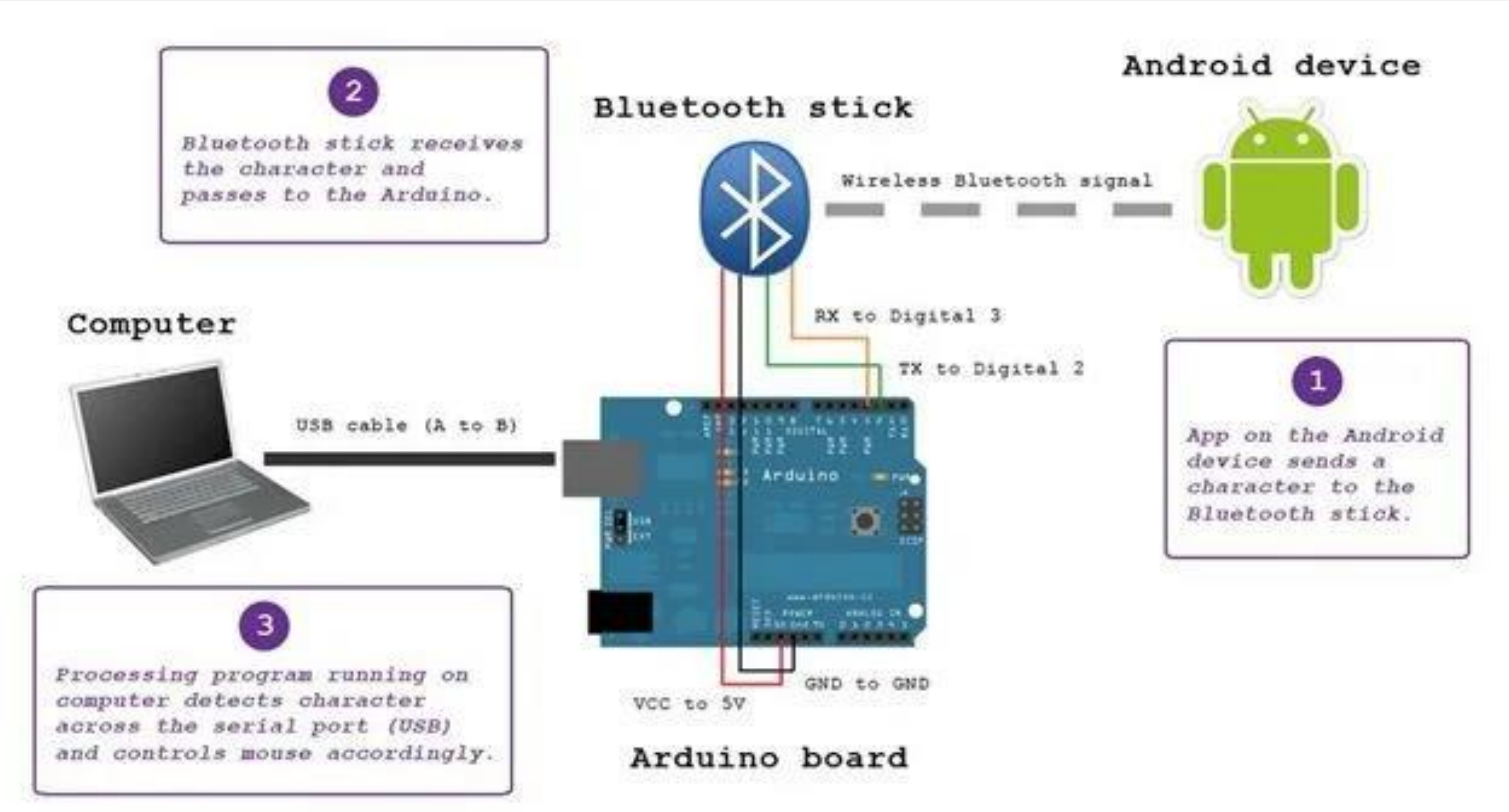
- Download and install the MIT App Inventor 2 Companion on the phone.

  Open the QR Code from the App inventor website.

Arduino Interfacing - 01

Switch ON/OFF LED

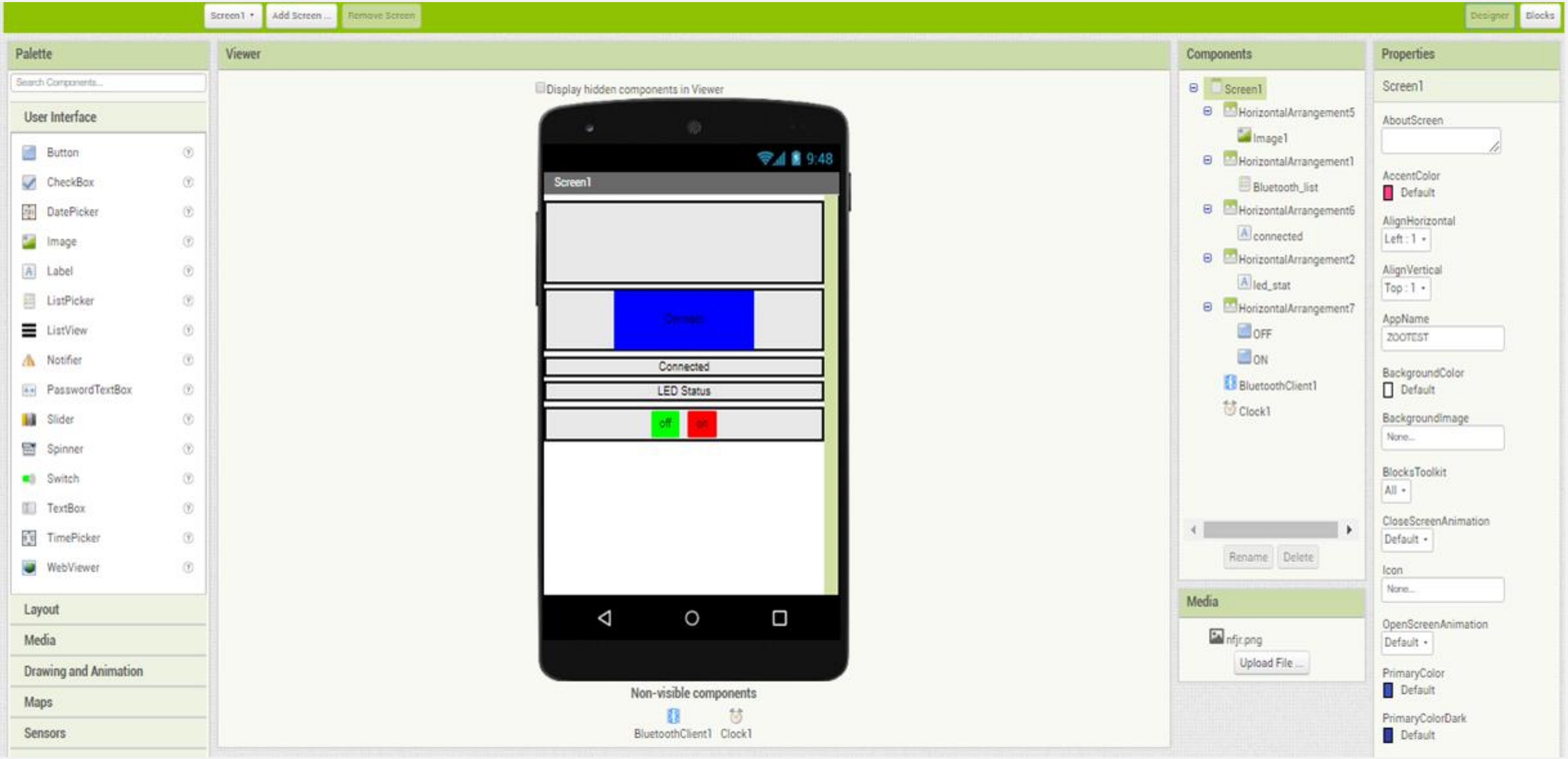




2 Bluetooth stick receives the character and passes to the Arduino.

1 App on the Android device sends a character to the Bluetooth stick.

3 Processing program running on computer detects character across the serial port (USB) and controls mouse accordingly.



AboutScreen

AccentColor: Default

AlignHorizontal: Left : 1

AlignVertical: Top : 1

AppName: ZOOTEST

BackgroundColor: Default

BackgroundImage: None...

BlocksToolkit: All

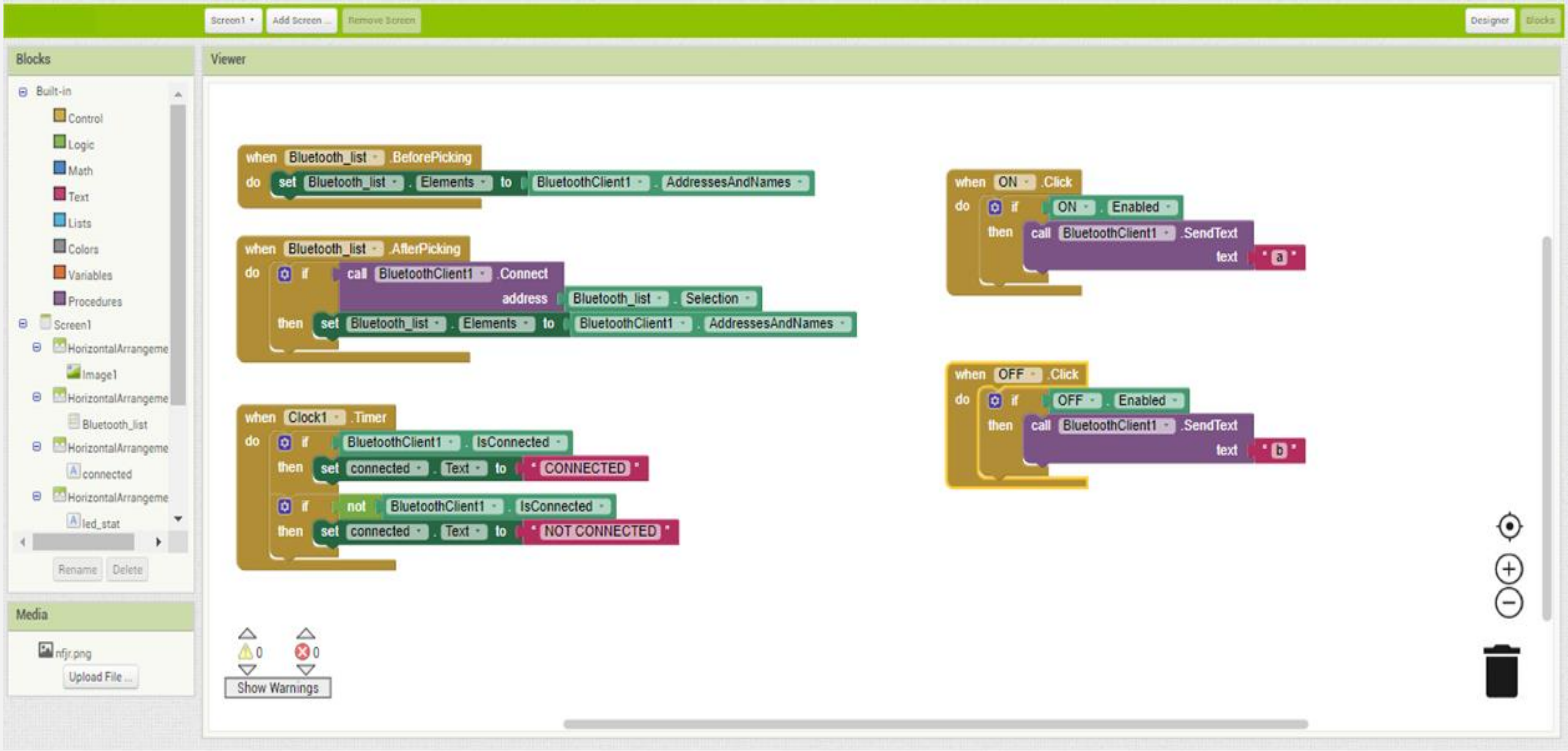
CloseScreenAnimation: Default

Icon: None...

OpenScreenAnimation: Default

PrimaryColor: Default

PrimaryColorDark: Default



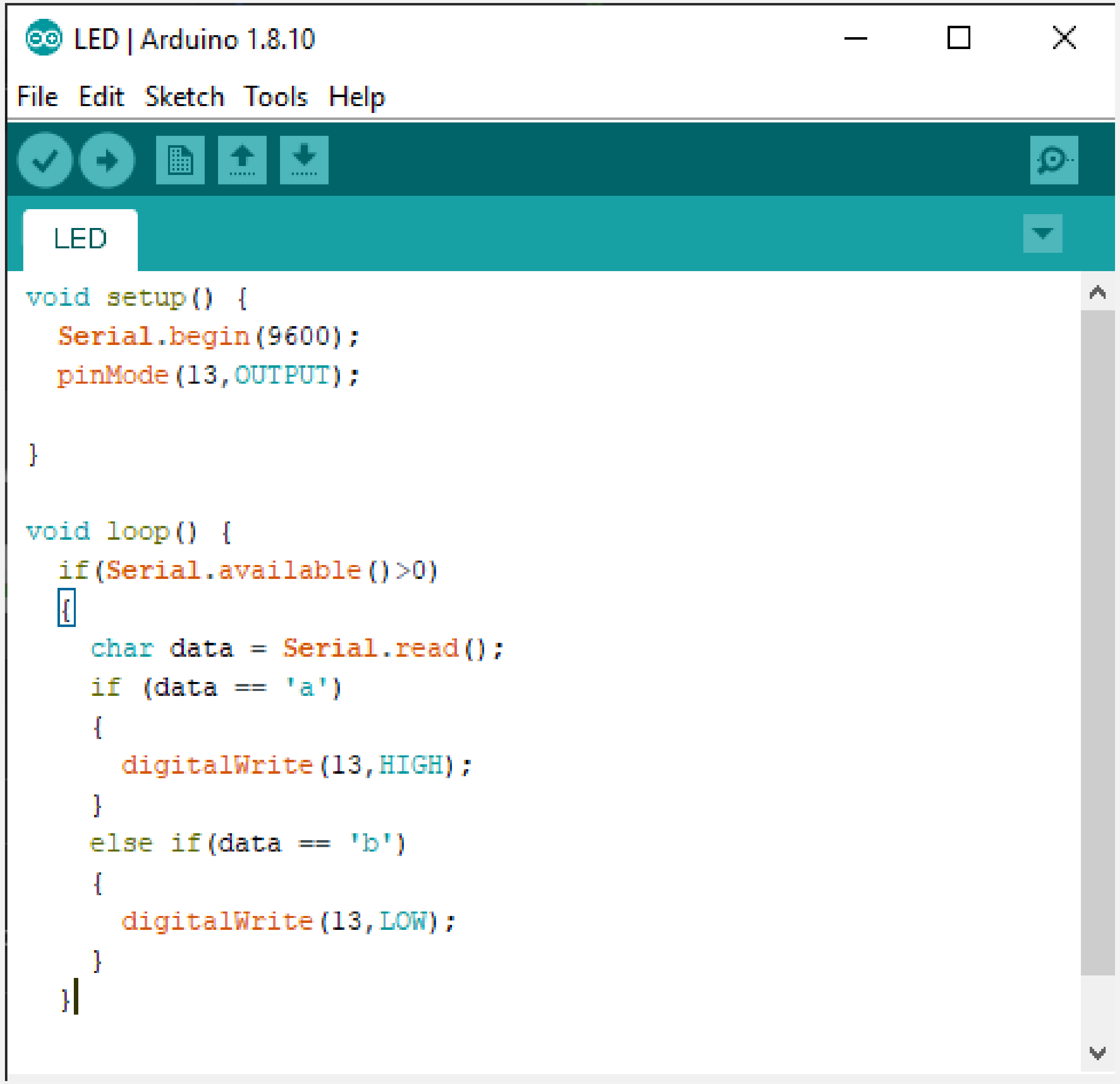
```
when Bluetooth_list .BeforePicking
do  set Bluetooth_list . Elements to BluetoothClient1 . AddressesAndNames

when Bluetooth_list .AfterPicking
do  if   call BluetoothClient1 .Connect
              address Bluetooth_list . Selection
    then set Bluetooth_list . Elements to BluetoothClient1 . AddressesAndNames

when Clock1 .Timer
do  if   BluetoothClient1 . IsConnected
    then set connected . Text to " CONNECTED "
    if   not BluetoothClient1 . IsConnected
    then set connected . Text to " NOT CONNECTED "

when ON .Click
do  if   ON . Enabled
    then call BluetoothClient1 .SendText
              text " a "

when OFF .Click
do  if   OFF . Enabled
    then call BluetoothClient1 .SendText
              text " b "
```



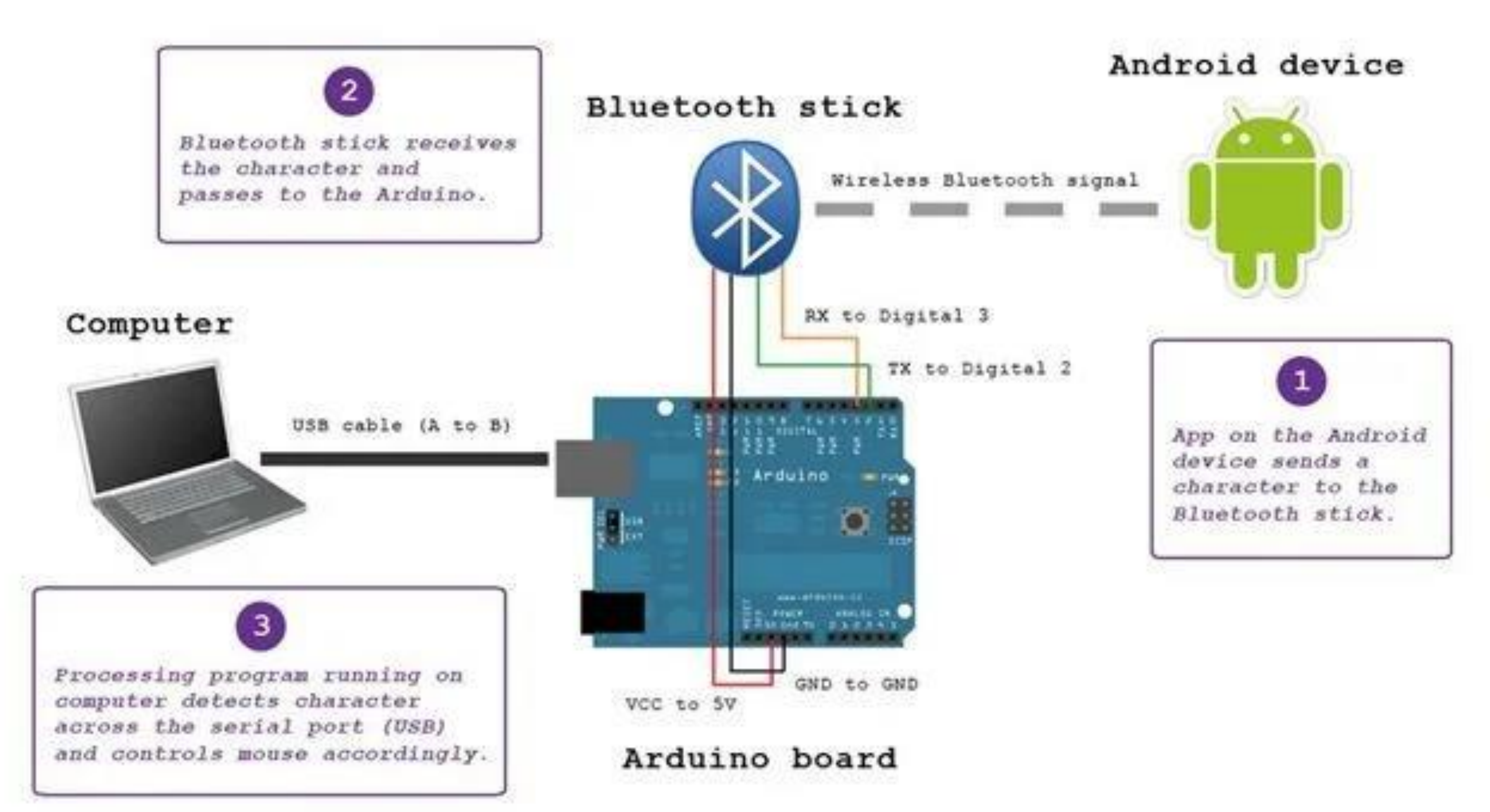
```
void setup() {
  Serial.begin(9600);
  pinMode(13,OUTPUT);

}

void loop() {
  if(Serial.available()>0)
  {
    char data = Serial.read();
    if (data == 'a')
    {
      digitalWrite(13,HIGH);
    }
    else if(data == 'b')
    {
      digitalWrite(13,LOW);
    }
  }
```

Arduino Interfacing - 02

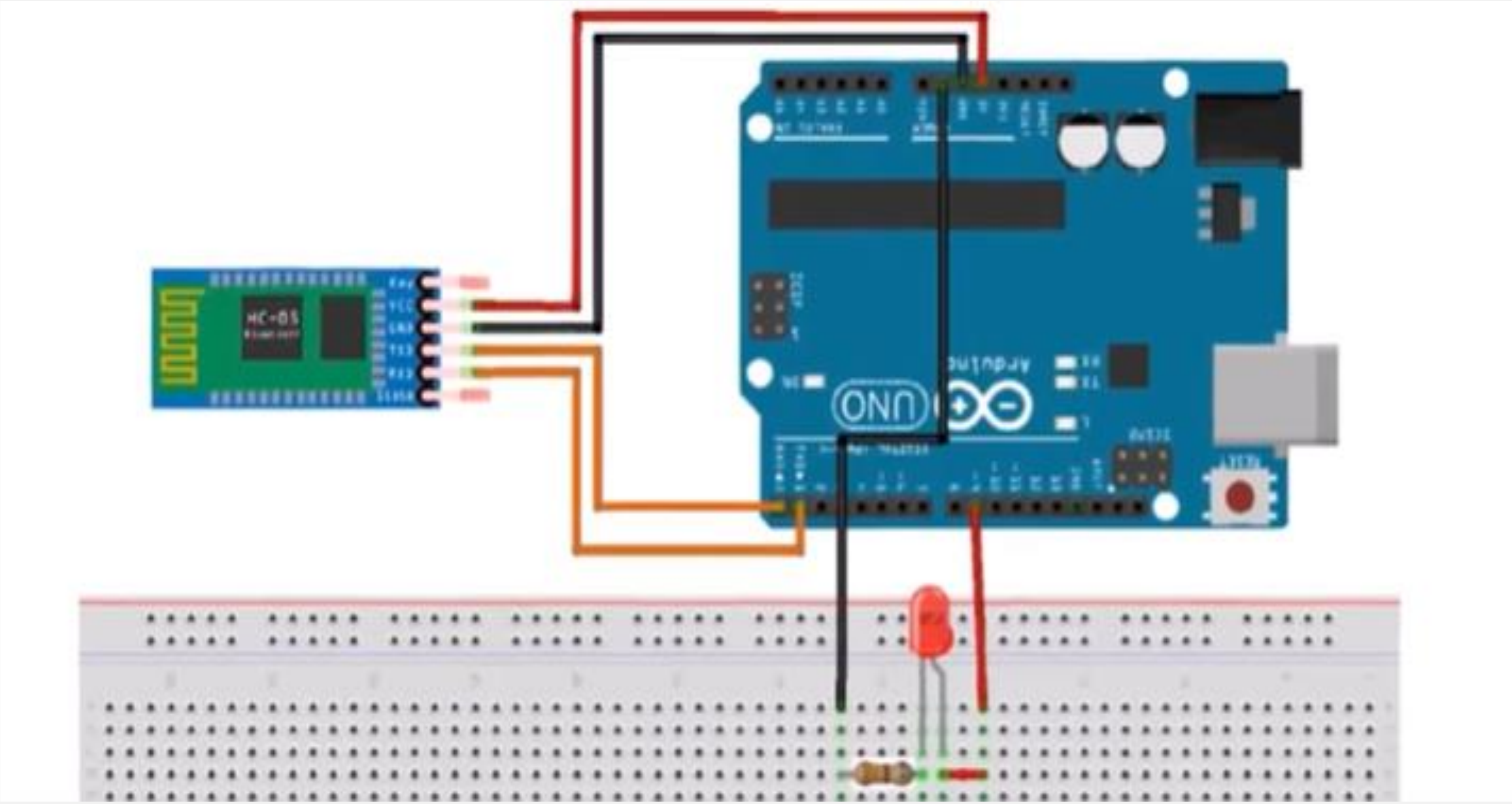
USE APP TO CONTROL BRIGHTNESS OF LED

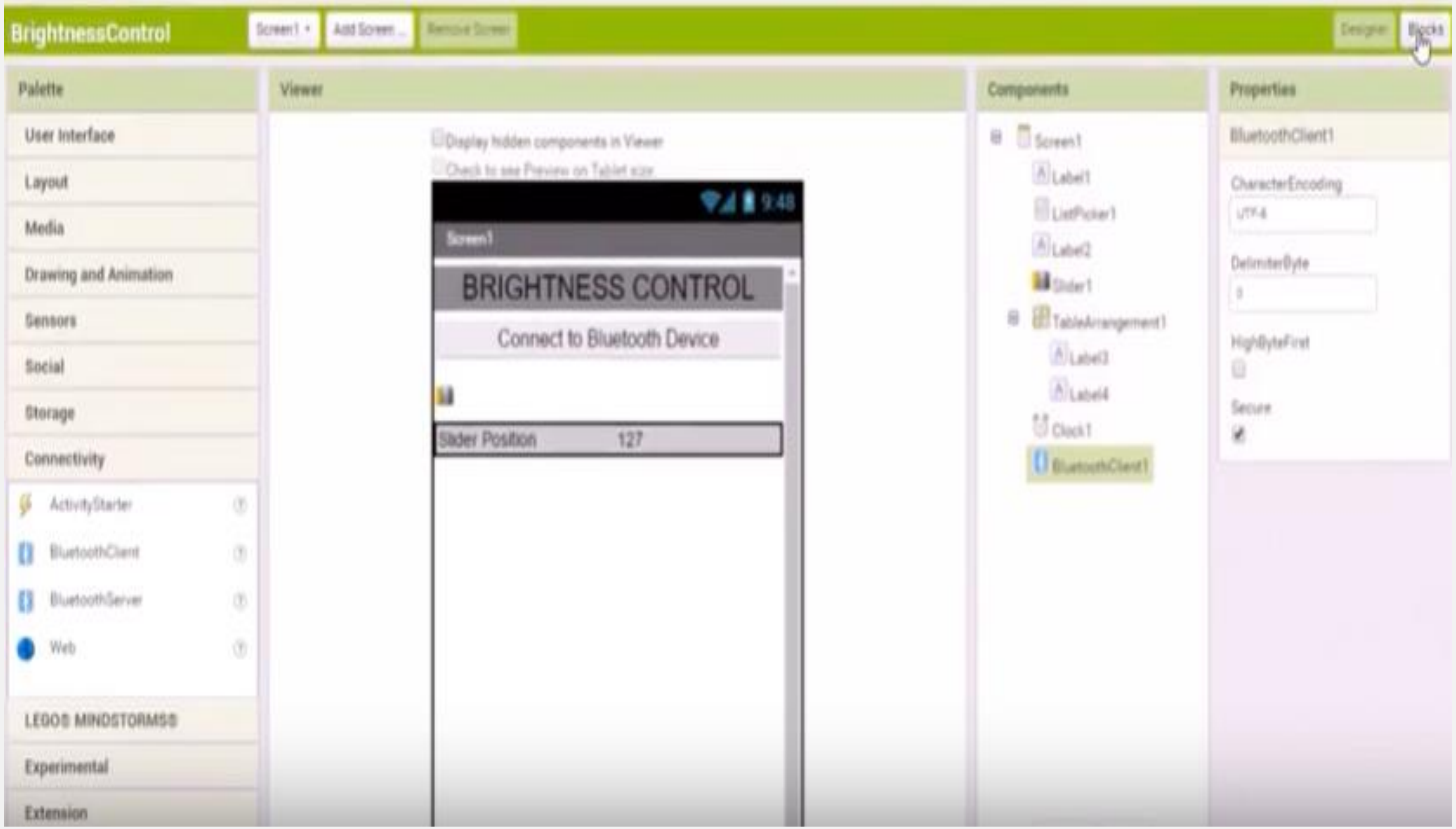


1 App on the Android device sends a character to the Bluetooth stick.

2 Bluetooth stick receives the character and passes to the Arduino.

3 Processing program running on computer detects character across the serial port (USB) and controls mouse accordingly.


HC-05
Key
VCC
GND
TXD
RXD
STATE
UNO
Arduino



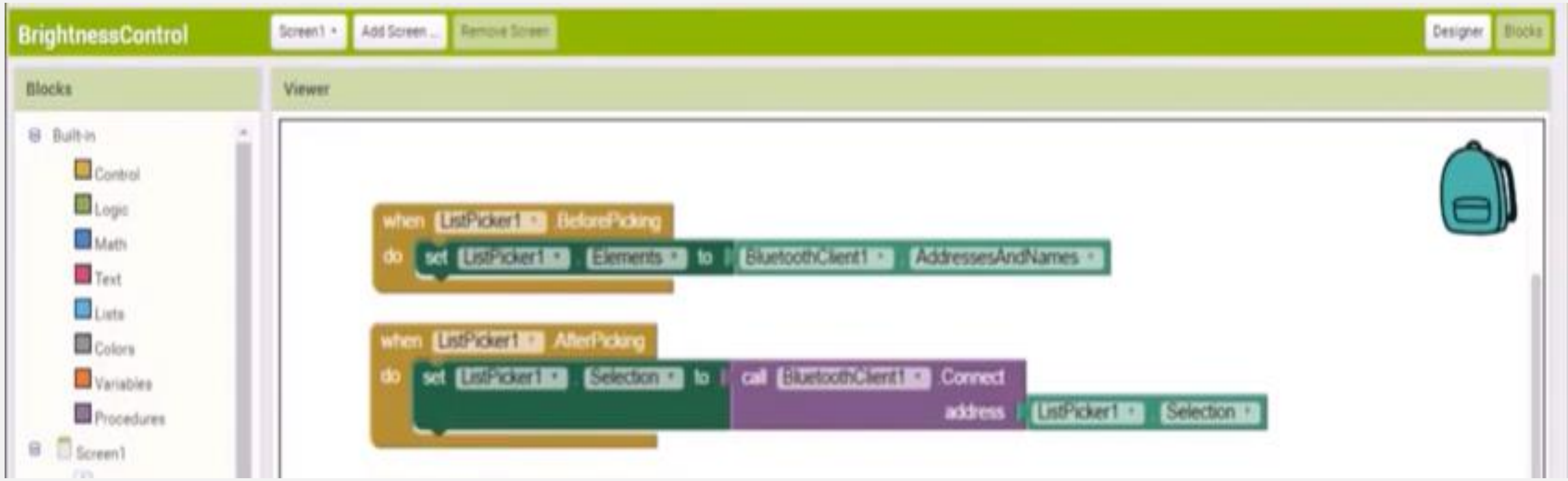
BluetoothClient1

CharacterEncoding
UTF-8

DelimiterByte
0

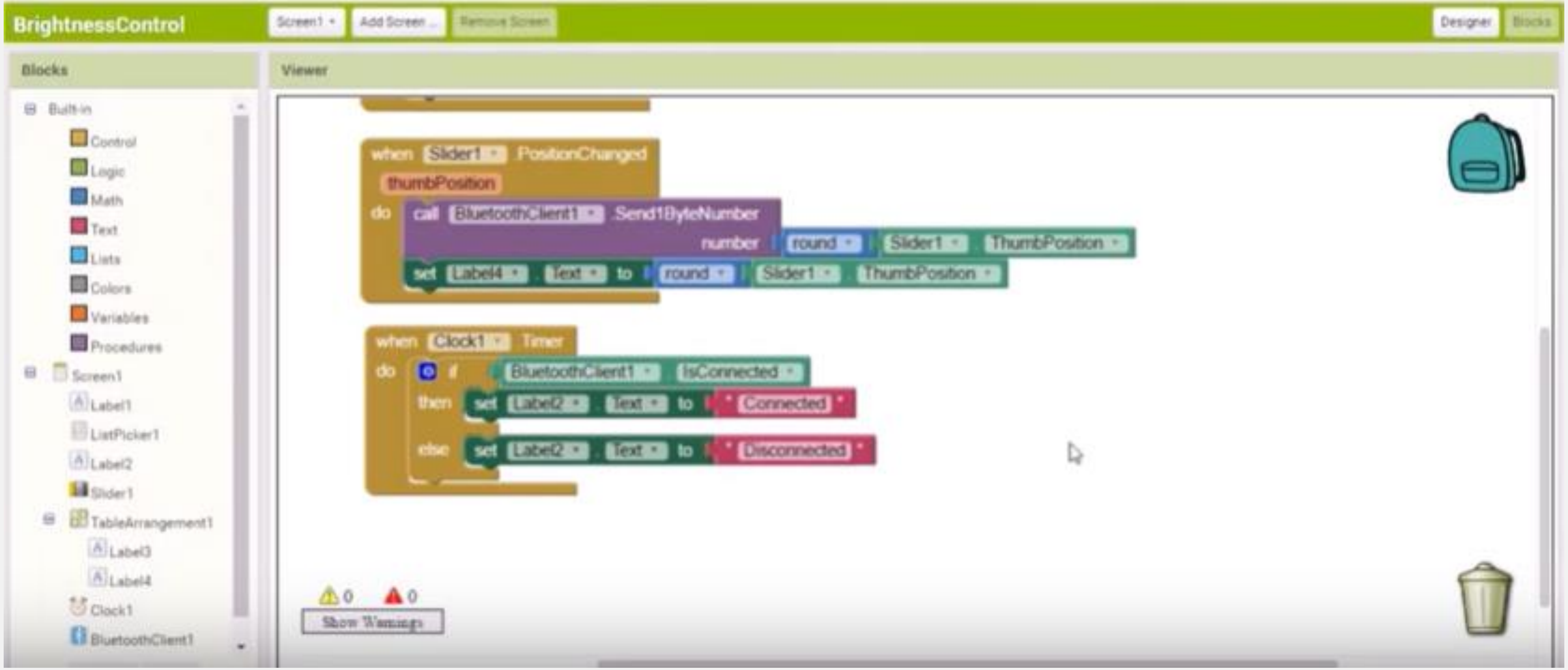
HighByteFirst
□

Secure
☑



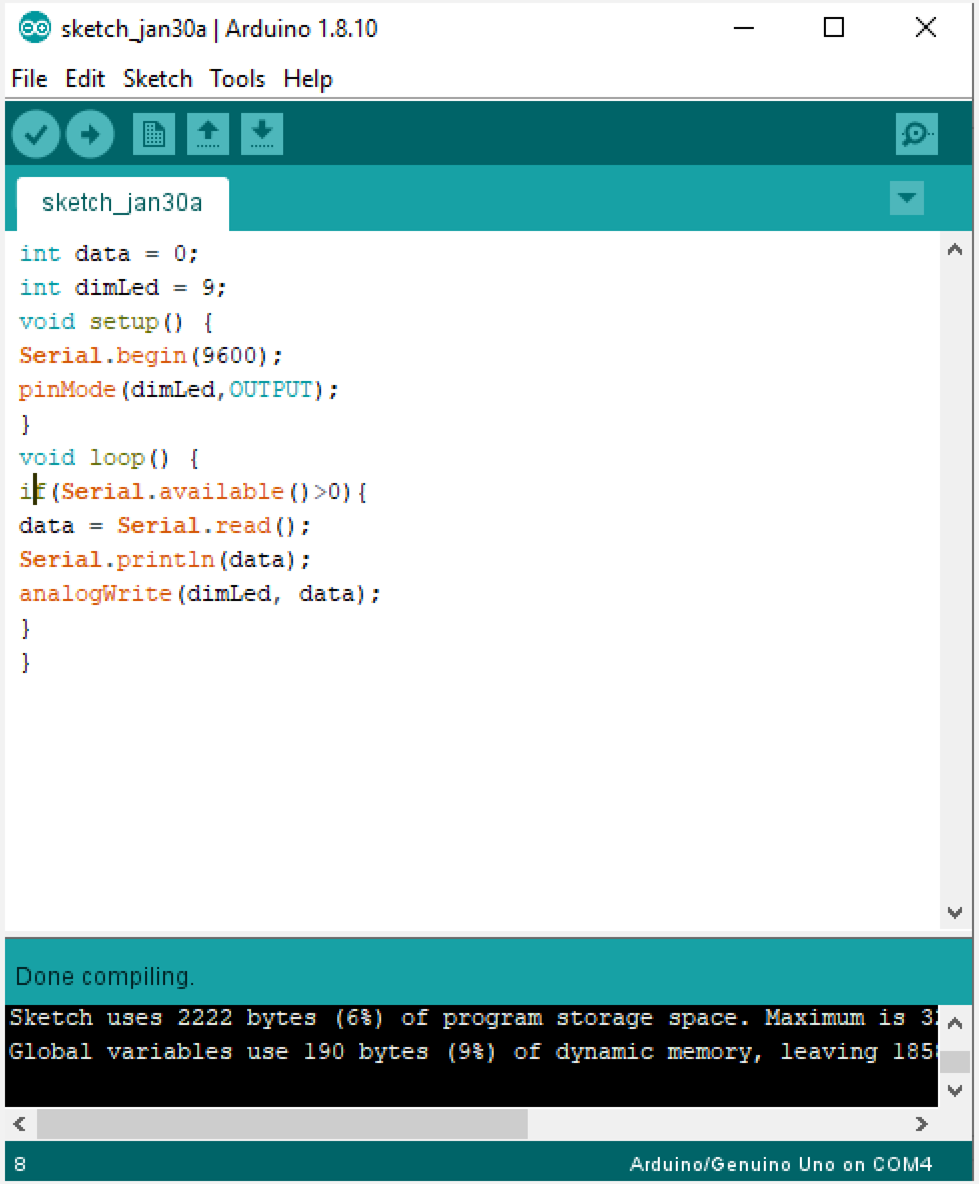
```
when ListPicker1 . BeforePicking
do  set ListPicker1 . Elements to  BluetoothClient1 . AddressesAndNames

when ListPicker1 . AfterPicking
do  set ListPicker1 . Selection to  call BluetoothClient1 .Connect
                                               address  ListPicker1 . Selection
```



```
when Slider1 .PositionChanged
 thumbPosition
do  call BluetoothClient1 .Send1ByteNumber
                            number  round  Slider1 . ThumbPosition
    set Label4 . Text to  round  Slider1 . ThumbPosition

when Clock1 .Timer
do  if  BluetoothClient1 . IsConnected
    then  set Label2 . Text to  " Connected "
    else  set Label2 . Text to  " Disconnected "
```


sketch_jan30a | Arduino 1.8.10
File Edit Sketch Tools Help
sketch_jan30a

```
int data = 0;
int dimLed = 9;
void setup() {
Serial.begin(9600);
pinMode(dimLed,OUTPUT);
}
void loop() {
if(Serial.available()>0){
data = Serial.read();
Serial.println(data);
analogWrite(dimLed, data);
}
}
```

Done compiling.

```
Sketch uses 2222 bytes (6%) of program storage space. Maximum is 3
Global variables use 190 bytes (9%) of dynamic memory, leaving 185
```

8
Arduino/Genuino Uno on COM4

THANK YOU!