

App Inventor + IoT: Walking robot with LinkIt 7697(BLE)



(with Basic
Connection tutorial
completed)

Level: advanced

This tutorial will help you get started with App Inventor + IoT and control a two-wheeled robot of LinkIt 7697 (Arduino compatible) with buttons on your app!

- [source .ino](#) / [source .aia](#)

Note: This is an advanced project, we assume you have intermediate understanding of App Inventor, Arduino and Bluetooth. Please check the [MIT App Inventor IoT website](#) for help getting started.

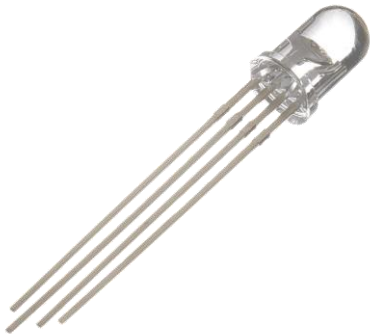
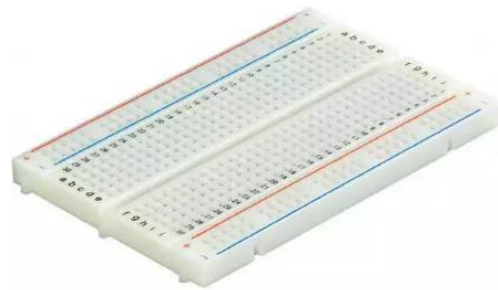
Hardware

[LinkIt 7697](#) is an Arduino compatible dev board with Wi-Fi / BLE. You can use it like just like any other Arduino, and interface with App Inventor through its Bluetooth communication.

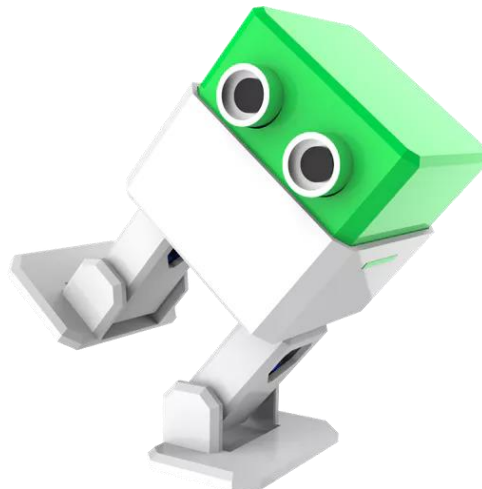
In this project, we are going to drive an [OTTO 2 leg robot](#) (an open source robot project) walking around and control. By four buttons on your app, you can move the robot forwards, backwards, left, and right, as well as stopping it. Here are the components we need to build this project:

- LinkIt 7697 dev board, 1
- breadboard, 1
- wires, several
- Servo, 4


- 2- leg robot chassis ([3D printed file here](#))
- RGB LED (common cathode), 1

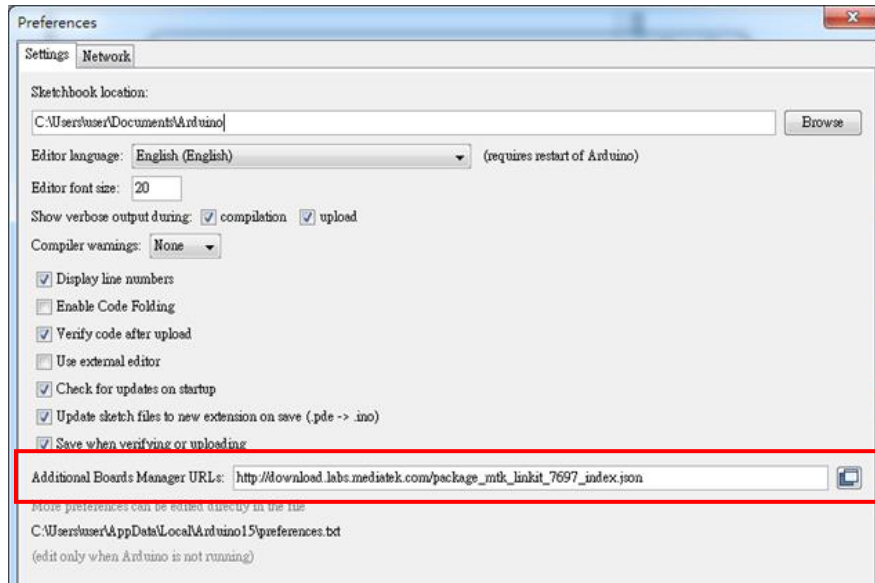


Our sample robot is like below:



Arduino IDE Setup

1. First go to the [Arduino IDE 1.8.x](#) version. Download the .zip file, unzip and click arduino.exe to open the IDE.
2. From **File**  **Preference** menu, enter the link below to Additional Boards Manager URLs field:
 - http://download.labs.mediatek.com/package_mtk_linkit_7697_index.json



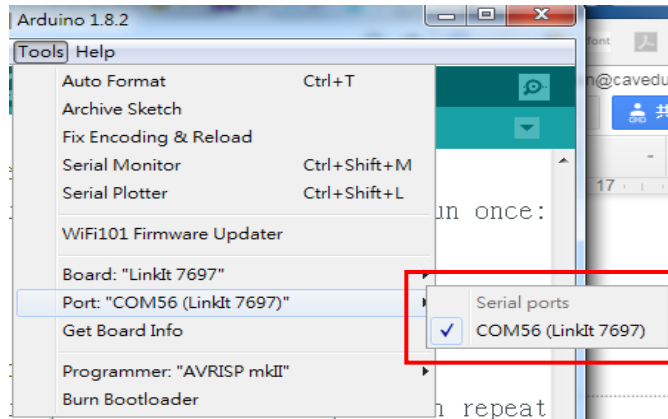
3. Open **Tools/ Board/ Board Manager**, then search "7697" and install the latest version of 7697 SDK.



4. Download and install the [CP2102N driver](#) ([Windows](#) / [MAC/OSX](#)) , then check the COM port in your Device manager. Look for "**Silicon Labs CP210 USB to UART**"

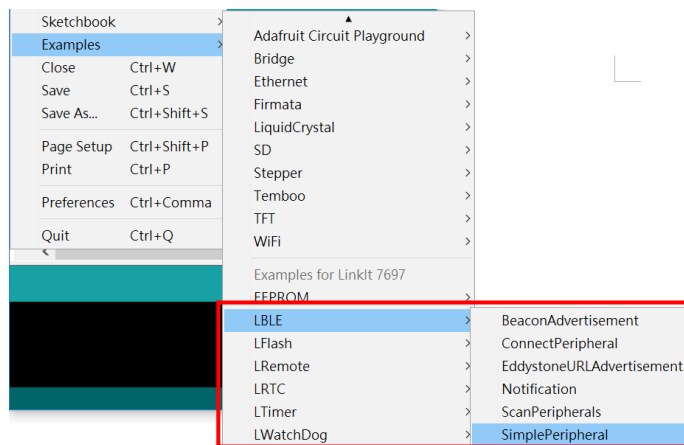
Bridge(COMXX)", this is the COM port number of your LinkIt 7697.

5. For MAC users, it should be something like **"/dev/tty.usbserialXXX..."** and for Windows users, please check the picture below:



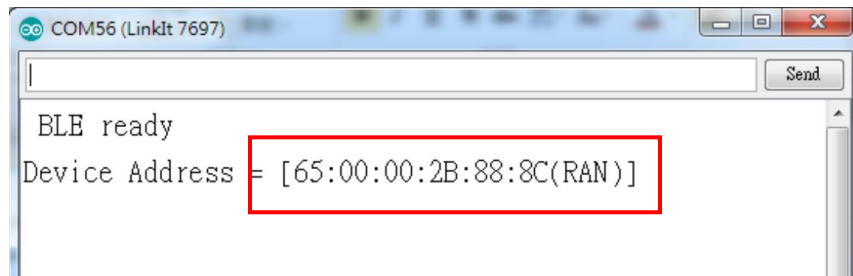
Get the BLE address of LinkIt 7697

1. For safety reasons, not every board has its Bluetooth address marked on the board (Arduino 101 is an exception). In Arduino IDE, first set the board to **"LinkIt 7697"** then open the example **"SimplePeripheral"** from File/Examples/LBLE menu.



2. Compile and upload to your LinkIt 7697 then open Arduino IDE's Serial Monitor. You should see an image like below. The **[XX:XX:XX:XX:XX:XX]** 12-digit string is the Bluetooth address of your LinkIt 7697, we have to modify the **addr** variable value of

your AI2 project. Later we will use the same .ino file to receive command from App Inventor.

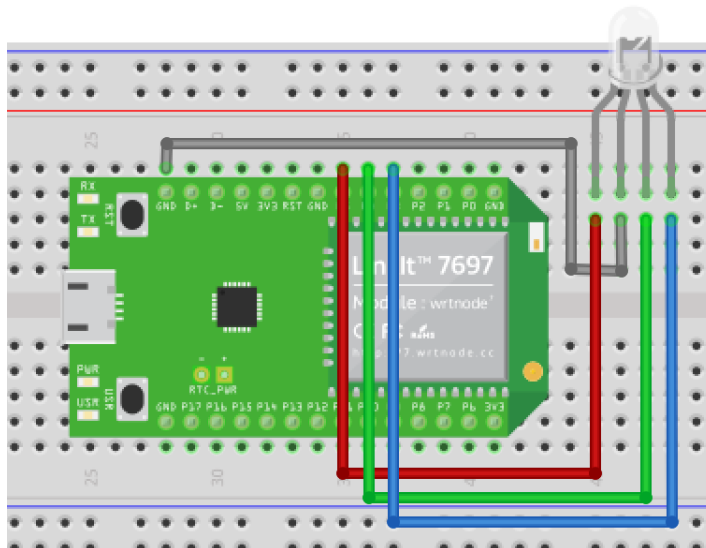


Hardware Assembly

Hardware of this project is a bit complicated, we separated it into two parts: motor and chassis. Here are the components we need to build this project:

1. RGB LED

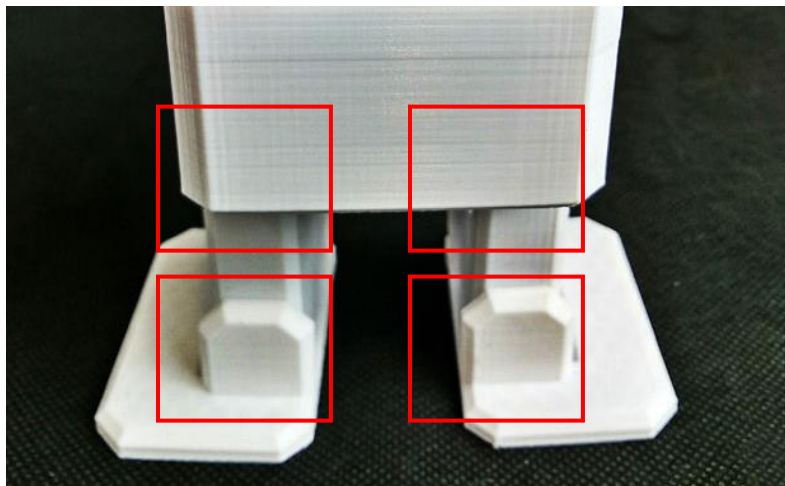
Our robot has a RGB LED as an indicator to show different colors: red, green, blue and white. Your circuit should look like the picture below, and please check the table for more details.



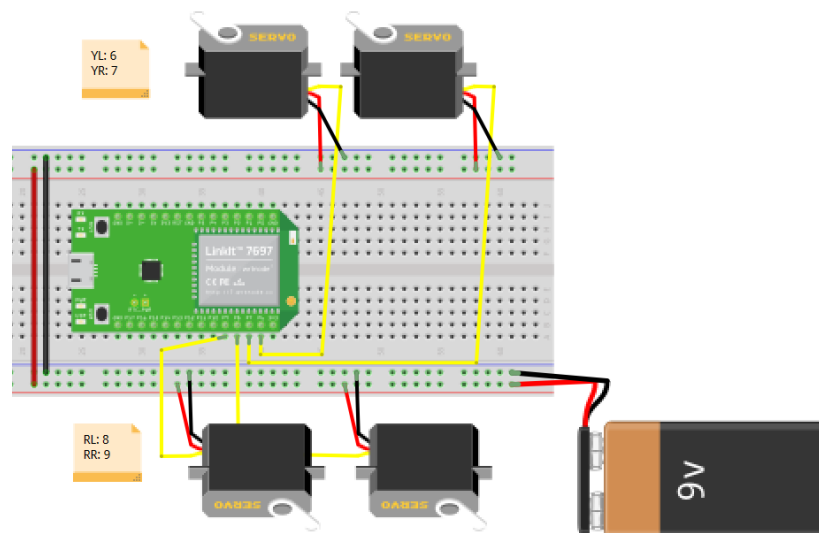
LinkIt 7697	RGB LED
P5	R
P4	G
P3	B
GND	GND

2. Servo

Our robot needs 2 servos for each leg, and they are labeled with **YR(right upper)**, **RR(right lower)**, **YL(Left upper)** and **RL(Left lower)**, as below:

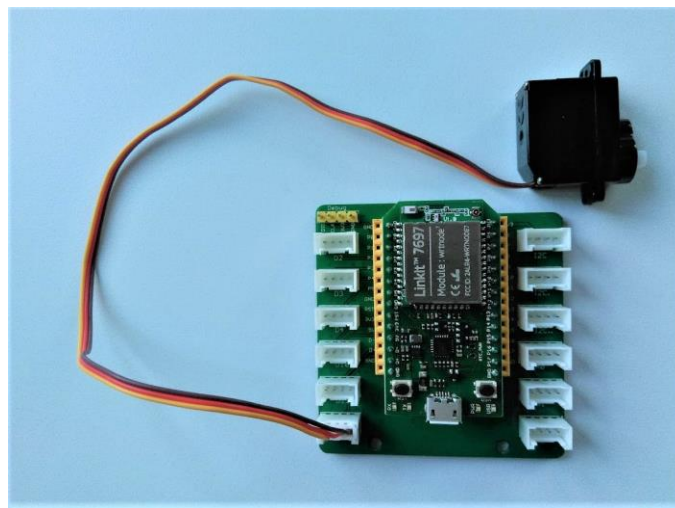


Your circuit should look like the picture below, and please check the table for more details.



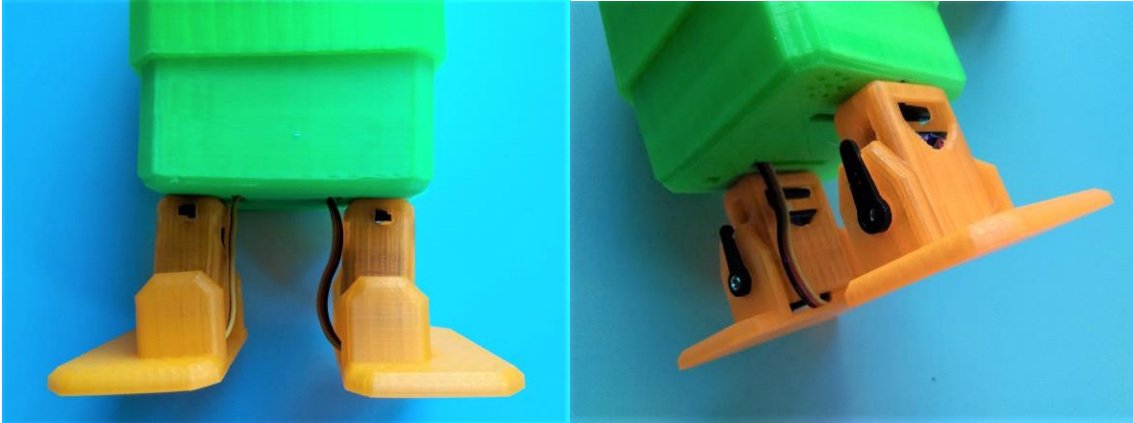
LinkIt 7697	Servos
P6	YL
P7	YR
P8	RL
P9	RR

Or you can use the extension board like [Grove extension shield for 7697](#) or [Robo shield for 7697](#), Additionally, shield also has certain protection for your board to be damaged by surge. Any model of Arduino-compatible servo shield should work fine with the LinkIt 7697.



3. Robot chassis

For 2-leg walking robot, we use an open source, 3D printed walking robot platform: [Otto robot](#). It has two servos for each leg, total are four servos. And we add a RGB LED as an indicator. Here are some photos of our sample robot.



App Inventor

We are going to make the robot move and stop by using buttons on your app. The main idea is to send out a number (integer) when you press a button, and the robot will do a corresponding action. Now let's take a look of the relationship between numbers and actions:

Number sent by App Inventor	Robot action
9	light white
8	light red
7	light green
6	light blue
5	light off
4	turn left
3	turn right
2	go forward
1	go backward

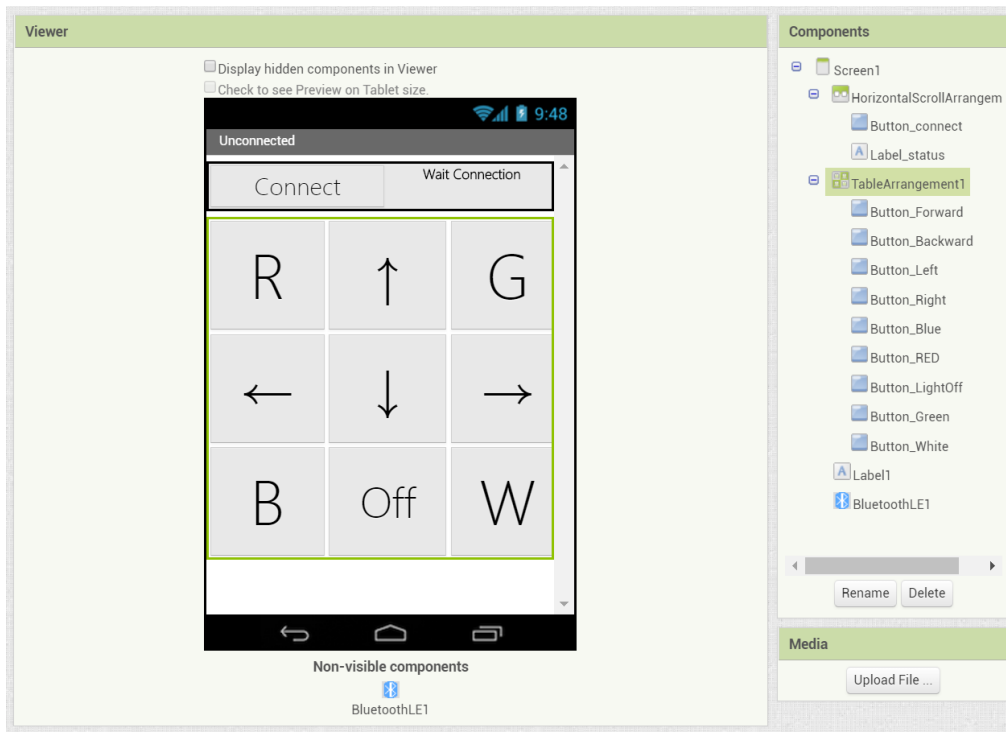
Note: the robot's behavior depends on the physical components you used, like servo and 3D-printed quality are all going to affect how your robot move.

Now log in to your [MIT App Inventor account](#) and create a new project.

Designer

1. The most used components in this project are buttons (to trigger actions) and labels (to show relevant messages).
2. We need to import the BLE extension from this URL:
 - <http://iot.appinventor.mit.edu/assets/com.bbc.micro:bit.profile.aix>
 - add the BLE extension by dragging it into Viewer.
3. Add a **TableArrangement** component, set its width to **Fill parent**, height to **200** pixels, Row to **3** and Column to **3**. Here we set its Visible property to **false**, means that user can not see it before connected LinkIt 7697.
4. For robot control, add four **buttons** into previous **TableArrangement** component. Give each a width of 33% and a height of 100 pixels. And modify the Text to "↑", "↓", "←" and "→", representing different moving direction of robot.
5. For RGB LED control, add five **buttons** into previous **TableArrangement** component. Give each a width of 33% and a height of 100 pixels. And modify the Text to "R", "G", "B", "W" and "Off", representing LED's red, green, blue, white light and light off.

After some adjusting, your designer should look similar to this. It doesn't have to be exactly the same. Feel free to modify the component's background color, position and text size.



Blocks

Let's take a look of our blocks step by step:

1. Variable for Bluetooth address

Please replace the `addr` variable value with what you get from Arduino's Serial Monitor, this is the Bluetooth address of LinkIt 7697.

initialize global `addr` to `" 65:00:00:2B:88:8C "`

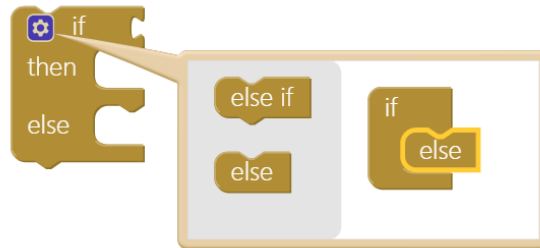
2. Initialize and connect

when `Screen1` .Initialize
do call `BluetoothLE1` .StartScanning

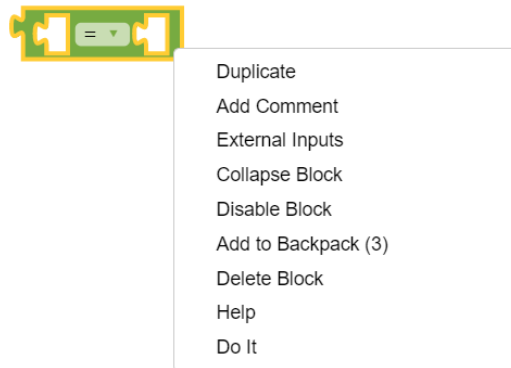
3. In **Screen1.Initialize** event, we ask **BluetoothLE** component to scan for BLE devices nearby (**BluetoothLE1.StartScanning**).

In **Button_connect.Click** event, we are going to connect or disconnect from Bluetooth device depending on the button text.

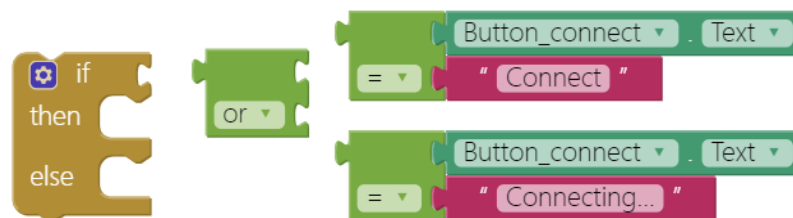
First, add an **if** condition, then click its blue gear to add an **else**.



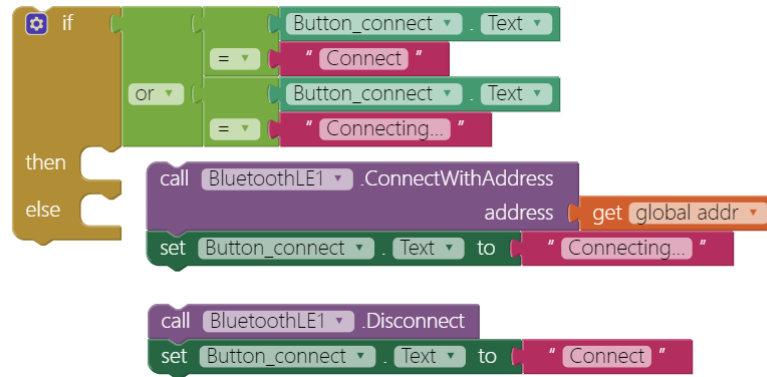
Add an **or** command from logic block, then right-click it and select "**External Inputs**". This will make it into more rectangular shape with input on the right-hand side.



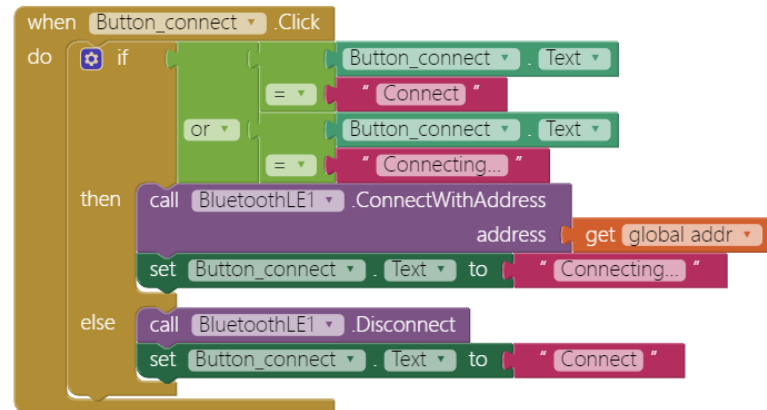
Now we want to check whether the **Button_connect.Text** status is "**Connect**" OR "**Connecting...**", this is how App Inventor decide to connect or disconnect Bluetooth connection with LinkIt 7697. please combine these blocks.



Good! When the **Button_connect**'s text is "**Connect**" OR "**Connecting...**", we will connect to the specified Bluetooth device (**BluetoothLE1.ConnectwithAddress**), which is our LinkIt 7697. If not, then disconnect (**BluetoothLE1.Disconnect**) and show message on Button_connect.

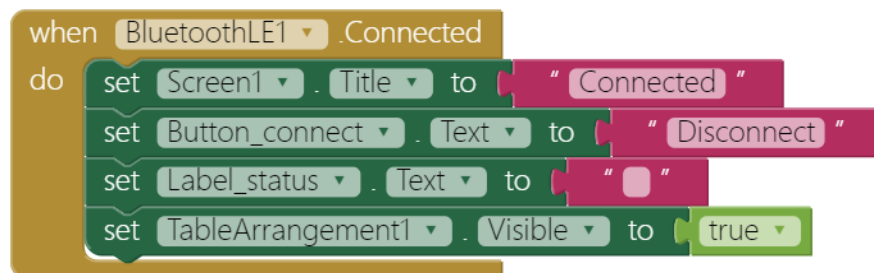


Put everything into **Button_connect.Click** event, and finish like this:



4. BLE Connected

When connected successfully (**BluetoothLE.Connected** event), we show relevant messages on several components and make **TableArrangement** component to show up on the screen.

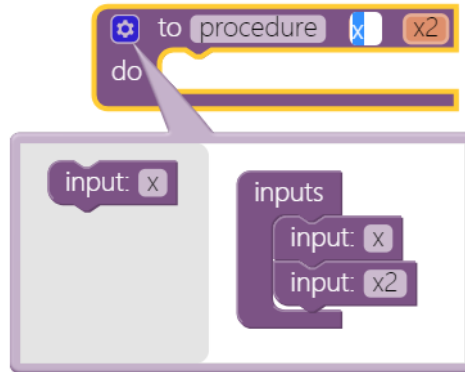


5. Procedure to update message and send Bluetooth command

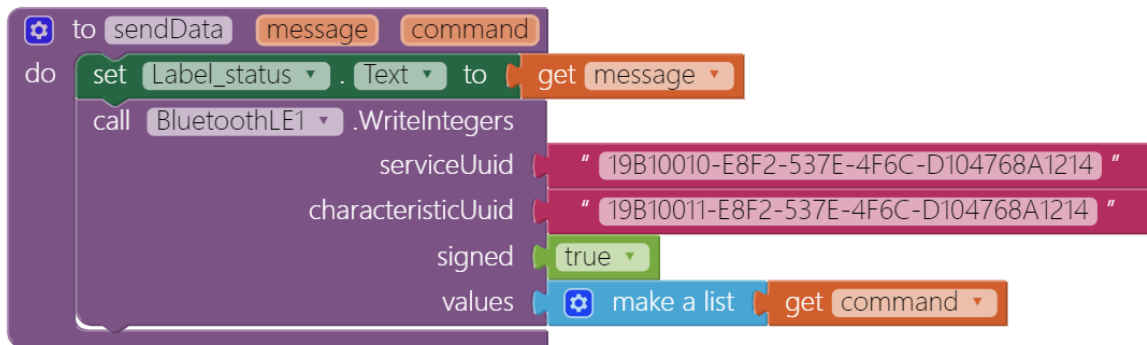
Here we use a procedure to manage what we send and

message update when we click each button.

Please add a procedure and click the blue gear to add two parameters. Rename this procedure to "**sendData**", and two parameters as "**message**" and "**command**", which means the **message** parameter **is used to** update the label and **command** parameter is used to send different command to LinkIt 7697.



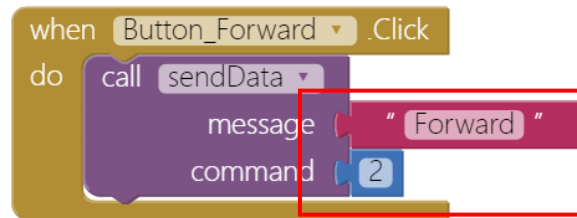
The **message** parameter of **sendData** procedure is used for update message on label, and the **command** parameter is the Bluetooth sent to LinkIt 7697 with **BluetoothLE.WriteIntegers** method. Notice that we have to make command into a list since the format of **BluetoothLE** component to accept.



6. Button Forward to drive robot going forward

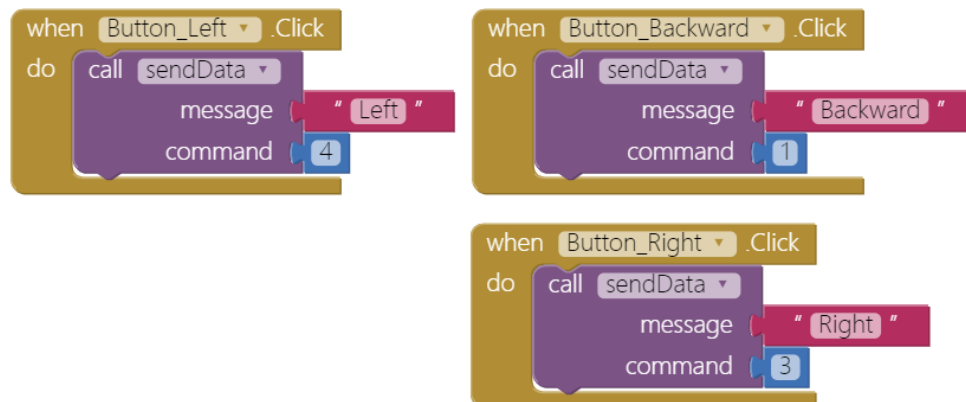
When user click the **Button_Forward**, **sendData** procedure will be called with (**Forward**, **2**) as parameter, which means we are going to show "Forward" on label and send an integer 2 to LinkIt 7697.

You can see the benefit of using procedure to manage the code, it can make our code more readable and flexible.



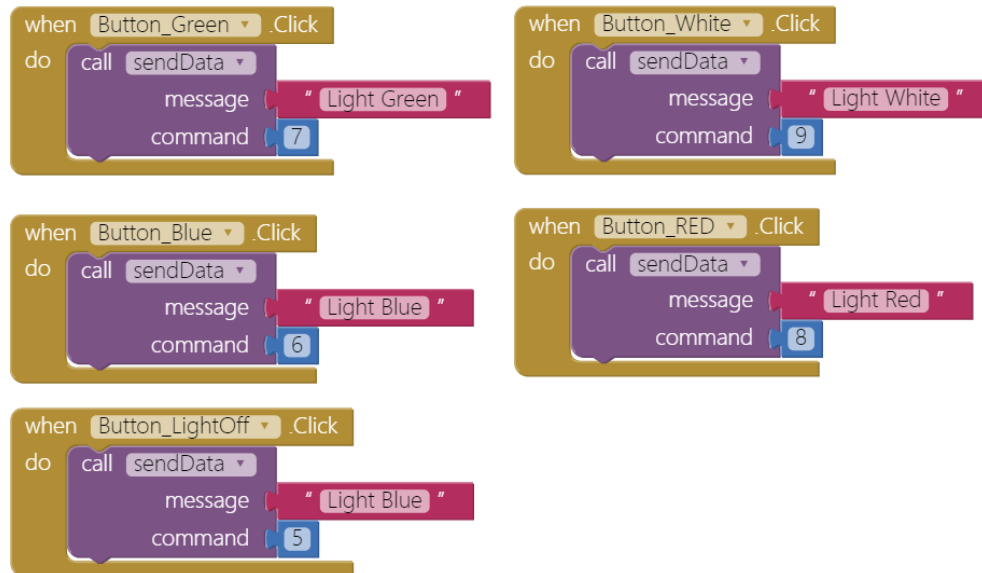
7. Buttons for other robot actions

Buttons to control other robot actions are listed below:



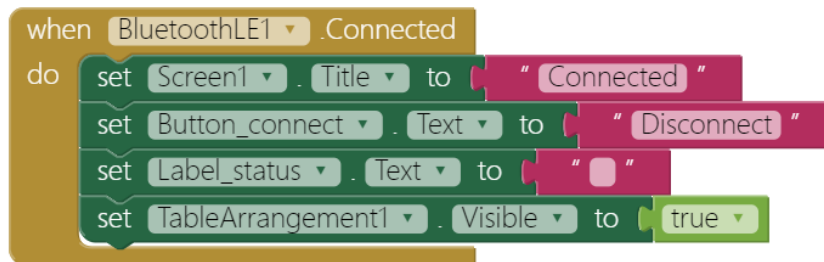
8. Buttons to control RGB LED

Buttons to control RGB LED's lighting colors are listed below:



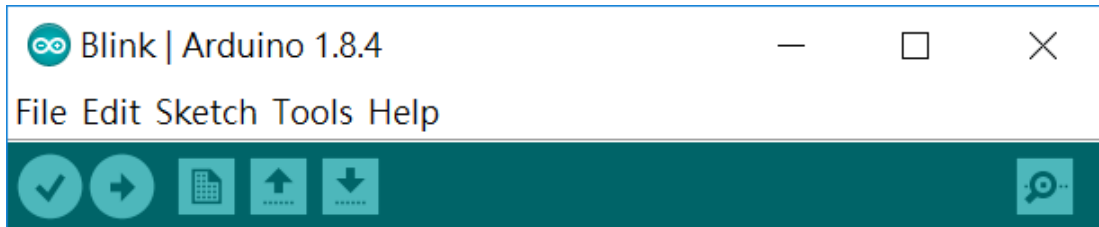
9. Disconnect

The Bluetooth connection will end when you click the **Button_connect** or pressed the **USR** button (D6) of LinkIt 7697. This will reset the app to its initial state to wait for next connect request.



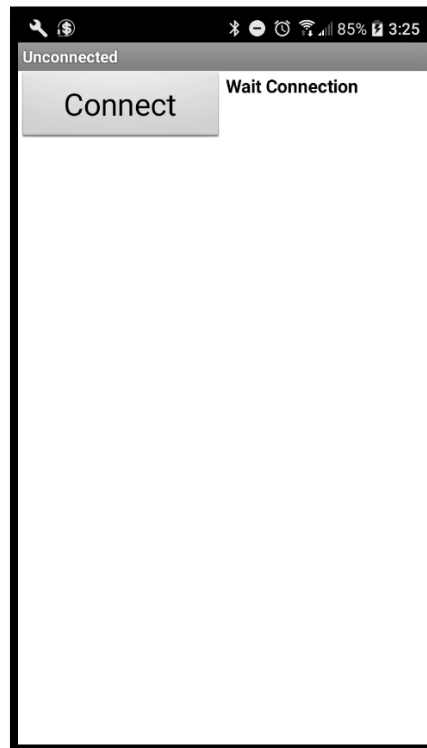
Arduino code

Please download from [here](#) and upload to your LinkIt 7697. Press the **"Upload"** right-arrow button, this will compile and upload the Arduino sketch in Arduino IDE to your LinkIt 7697. Please make sure you see the **"done uploading"** message in the console below.

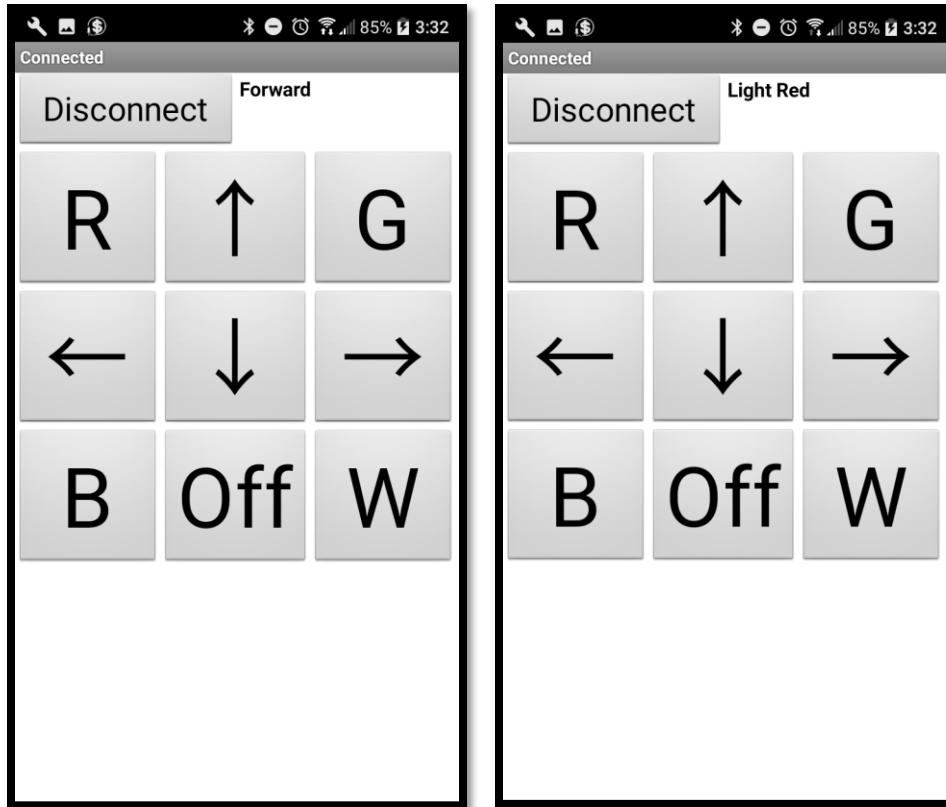


Tips

Make sure your LinkIt 7697 is running correctly as a BLE peripheral. Open your app and click **Connect** button. Click each button to drive your robot moving around.



(app initialized)



Connected and pressed " ↑ " and "R"

Brainstorming

1. Add a Speechrecognizer component and use voice command to drive your robot like "go forward", "go backward", "turn left", "turn right" and "stop". (reference: [LED control tutorial](#))