Getting Started with MIT App Inventor 2 and Arduino

Getting started building Android apps that interact and control your Arduino.

Here's the contents that are going to be covered in this post:

- Introducing MIT App Inventor 2
- Why MIT App Inventor 2 is a good choice?
- Accessing MIT App Inventor 2
- How to control Arduino with MIT App Inventor (Bluetooth)
- MIT App Inventor Overview
- Projects with MIT App Inventor
- Android Apps for Arduino with MIT App Inventor 2

Introducing MIT App Inventor 2

**RELATED CONTENT:** [Like ESP8266? Check out Home Automation Using ESP8266](#)

MIT App Inventor 2 is a simple and intuitive free service for creating Android applications. If you want to start with MIT App Inventor, you don’t need to download or install any program in your computer as the software is cloud-based, so you build the apps directly in your browser (Chrome, Mozilla, Safari, Internet Explorer, etc).

For instance, you only need an internet connection for building the apps.

Why MIT App Inventor is a good choice?

- MIT App Inventor 2 is intuitive and simple to use.
- You don’t have to be an expert in programming or design to build awesome apps that can do useful stuff.
- Creating the design is as easy as selecting and placing widgets in the smartphone screen.
- The code is done with drag and drop puzzle blocks.

Anyone can learn how to build their own apps with MIT App Inventor 2 with a few hours of practice.

Accessing MIT App Inventor 2

To access MIT App Inventor 2 go to [http://appinventor.mit.edu/explore/](http://appinventor.mit.edu/explore/) and press the orange **Create Apps** button.
To access the app builder, you need a Google account. Follow the on-screen steps to login into MIT App Inventor 2. After that, you’ll be presented with the following dashboard (we’ll cover how to use the dashboard in the MIT App Inventor 2 Overview section):

How to Control Arduino with MIT APP Inventor (Bluetooth)

To establish a connection between the Arduino and your Android app, you need a Bluetooth communication protocol. For that, you need a Bluetooth module.

The most common Bluetooth modules used with the Arduino are the HC-05 bluetooth, HC-04 and HC-06.

For more information about the bluetooth module, you can check this blog post: [Reviews – HC-05 Bluetooth Module](http://example.com)

The Bluetooth module works with serial data. This means that the Arduino sends information and the Bluetooth module receives it via serial (and vice-versa).

The following figure explains how the information flows from the Android app to the Arduino.

Your smartphone sends information to the Bluetooth module via Bluetooth. Then, the Bluetooth module sends the information via serial communication to the Arduino.

This flow also works the other way around: the Arduino sends information to the Bluetooth module that sends it to the smartphone via Bluetooth.
MIT App Inventor Overview

Go to http://appinventor.mit.edu/explore/ and press Create Apps button.

Next, click on Start new project as shown in figure below.

You’ll be asked to give your project a name. As we’re just exploring the MIT App Inventor 2 features, you can name it test.

Click OK. Your project is automatically saved.

If you go to Projects > My Projects you can see all your saved projects.

Click on the project name to open the app builder.

Designer

You’ll be presented with the Designer tab as shown in the following figure.
At 1) you select whether you are on the **Designer** or in the **Blocks Editor** tab. With MIT App Inventor you have 2 main sections: **Designer** and **Blocks**.

The designer gives you the ability to add buttons, add text, add screens and edit the overall app look.

The **Blocks** section allows you to create custom functionality for your app, so when you press the buttons it actually does something with that event.

2) **The Palette** contains the components to build the app design like buttons, sliders, images, labels, etc...

3) **It's the Viewer**. This is where you drag the components to build the app look.

4) **Components**. You can see all the components added to your app and how they are organized hierarchically.

5) **Properties**. This is where you select your components’ properties like color, size and orientation.

**Blocks Editor**

Open the **Blocks** editor tab.

In the **Blocks** editor tab, you have several sections:
1) contains the built-in blocks for creating the app’s logic. This is what makes the app define the buttons functionalities, send commands to Arduino, connect to the Bluetooth module, etc. You have several blocks grouped by categories:

- **Control**: if/else statements, while loops, etc...
- **Logic**: True, False, equal, not equal, etc...
- **Math**: math operators.
- **Text**: blocks that deal with text.
- **Lists**: blocks for handling lists.
- **Colors**: blocks to handle colors, like choosing a color, make color and split colors.
- **Variables**: initialize variables, setting variables values, get variables values, etc...
- **Procedures**: procedures are like functions. A procedure is a sequence of code blocks with a given name. Later, you can call that sequence of blocks instead of creating the same long sequence.

Inside each group, you have blocks that you can drag to the **Viewer**. In the **Viewer**, you drag the blocks and join them in a specific way to make something happen.

We recommend that you navigate inside the blocks section and explore what’s inside. The blocks look like puzzle pieces that fit into each other or not. If you can’t do something with certain blocks, they won’t fit.

In the **backpack** you save code blocks to use later. You move blocks to the **dustbin** to delete them.

**Projects with MIT App Inventor**

This was just a quick introduction to the MIT App Inventor. Now, it’s time to start building apps!

Here’s a list of our popular Arduino and MIT App Inventor projects:

- Android App – RGB LED with Arduino and Bluetooth
- Arduino – Control 2 DC Motors Via Bluetooth
**Control your Arduino with Voice Commands**

### Arduino - Control 2 DC Motors Via Bluetooth (Perfect To Build a Robot)

How to control 2 DC motors via bluetooth with an Android app created with MIT App Inventor 2.

In a previous tutorial ([click here to see that project](#)), I was controlling 1 DC motor using an app called “BlueTerm”. That app did the job, but it’s not ideal to send constantly different commands in an easy manner.

The app that you’re going to build is perfect to control any Arduino pin or to integrate with your own robot car.

#### Bluetooth module HC-05

To establish the bluetooth communication between your smartphone and your Arduino, you need a bluetooth module. This project uses the HC-05 bluetooth module

![Bluetooth module HC-05](image)

This bluetooth module works with serial data. This means that the Arduino sends information and the bluetooth module receives it via serial (and vice-versa).

By default the HC-05 bluetooth module operates at a baud rate of 9600.

#### Creating your Android app

The Android App will be created using a free web application called [MIT App Inventor](http://ai2.appinventor.mit.edu).

You need a Google account to sign up for MIT App Inventor and here’s the login page: [http://ai2.appinventor.mit.edu](http://ai2.appinventor.mit.edu).

**Click here to download the .aia file.**

If you go to the Projects tab, you can upload the [.aia file for this project](#).
With MIT App Inventor you have 2 main sections: designer and blocks. The designer is what gives you the ability to add buttons, add text, add screens and edit the overall app look.
The blocks sections is what allows to create custom functionality for your app, so when you press the buttons it actually does something with that information.

I recommend that you start by following this project and using the app without modifying it.

If want to make any changes to the app, when you’re done and you want to install the app in your smartphone, go to the Build tab.

- You can either generate a QR code that you can scan with your smartphone and automatically install the app in your smartphone.
- Or you can download the .apk file, connect your smartphone to your computer and move the .apk file to the phone.

Simply follow the installation wizard to install the app and it’s done!

**Code**

For this project, you don’t need to install any Arduino libraries.

So, you just have to download or copy the following code to your Arduino IDE, and upload it to your Arduino board. Make sure that you have the right Board and COM port selected.

```cpp
/*
 * created by Rui Santos, http://randomnerdtutorials.com
 * Control 2 DC motors with Smartphone via bluetooth
 */

int motor1Pin1 = 3; // pin 2 on L293D IC
int motor1Pin2 = 4; // pin 7 on L293D IC
int enable1Pin = 6; // pin 1 on L293D IC
int motor2Pin1 = 8; // pin 10 on L293D IC
int motor2Pin2 = 9; // pin 15 on L293D IC
int enable2Pin = 11; // pin 9 on L293D IC
int state;
int flag=0; //makes sure that the serial only prints once the state
int stateStop=0;

void setup() {
  // sets the pins as outputs:
  pinMode(motor1Pin1, OUTPUT);
  pinMode(motor1Pin2, OUTPUT);
  pinMode(enable1Pin, OUTPUT);
  pinMode(motor2Pin1, OUTPUT);
  pinMode(motor2Pin2, OUTPUT);
  pinMode(enable2Pin, OUTPUT);
  // sets enable1Pin and enable2Pin high so that motor can turn on:
  digitalWrite(enable1Pin, HIGH);
  digitalWrite(enable2Pin, HIGH);
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

void loop() {
  //if some date is sent, reads it and saves in state
  if(Serial.available() > 0){
  ```
state = Serial.read();
flag=0;
}
// if the state is 'F' the DC motor will go forward
if (state == 'F') {
digitalWrite(motor1Pin1, HIGH);
digitalWrite(motor1Pin2, LOW);
digitalWrite(motor2Pin1, LOW);
digitalWrite(motor2Pin2, HIGH);
if(flag == 0){
Serial.println("Go Forward!");
flag=1;
}
}

// if the state is 'R' the motor will turn left
else if (state == 'R') {
digitalWrite(motor1Pin1, HIGH);
digitalWrite(motor1Pin2, LOW);
digitalWrite(motor2Pin1, LOW);
digitalWrite(motor2Pin2, LOW);
if(flag == 0){
Serial.println("Turn LEFT");
flag=1;
}
delay(1500);
state=3;
stateStop=1;
}
// if the state is 'S' the motor will Stop
else if (state == 'S' || stateStop == 1) {
digitalWrite(motor1Pin1, LOW);
digitalWrite(motor1Pin2, LOW);
digitalWrite(motor2Pin1, LOW);
digitalWrite(motor2Pin2, LOW);
if(flag == 0){
Serial.println("STOP!");
flag=1;
}
stateStop=0;
}
// if the state is 'L' the motor will turn right
else if (state == 'L') {
digitalWrite(motor1Pin1, LOW);
digitalWrite(motor1Pin2, LOW);
digitalWrite(motor2Pin1, LOW);
digitalWrite(motor2Pin2, HIGH);
if(flag == 0){
Serial.println("Turn RIGHT");
flag=1;
}
delay(1500);
state=3;
stateStop=1;
}
// if the state is 'B' the motor will Reverse
else if (state == 'B') {
digitalWrite(motor1Pin1, LOW);
digitalWrite(motor1Pin2, HIGH);
digitalWrite(motor2Pin1, HIGH);
digitalWrite(motor2Pin2, LOW);
if(flag == 0){
Serial.println("Reverse!");
flag=1;
}
}
//For debugging purpose
//Serial.println(state);

Note: before uploading the code, make sure you have the TX and RX pins disconnected from the bluetooth module

Launching your app

If you haven’t generated the .apk file in a previous step, you can click here to download the .apk file (which is the Android app installation file).

Move that file to your smartphone and open it. Follow the installation wizard to install the app.

Turn on your smartphone’s Bluetooth.
Tap on the newly installed app.
Press the “Connect” button to connect your application to your Arduino Bluetooth module.
Now you can easily control the 2 DC motors with your app:

**Android App - RGB LED with Arduino and Bluetooth**

In this project you’re going to build an Android app to control the color of an RGB LED with a smartphone via bluetooth.

**RELATED CONTENT:** Like ESP8266? Check out Home Automation Using ESP8266
If you’re not familiar with RGB LEDs, read the following post: How do RGB LEDs work?
Download Android Apps for Arduino with MIT App Inventor 2
**Parts Required**

Here's a complete list of the parts required for this project:

- Arduino
- Bluetooth module HC-04 or HC-05 or HC-06
- RGB LED (common anode)
- 3 x 220Ohm resistor
- Breadboard
- Jumper wires

Additionally, you also need a smartphone with bluetooth.

**Creating the Android App**

You need a Google account to sign up for MIT App Inventor and here's the login page: [http://ai2.appinventor.mit.edu](http://ai2.appinventor.mit.edu).

After login in go to **Projects > Import project (.aia) from my computer** and upload the .aia file. [Click here to download the .aia file](http://ai2.appinventor.mit.edu).
After uploading the .aia file, you'll see the application on the MIT App Inventor Software.

**Designer**

With MIT App Inventor you have 2 main sections: **designer** and **blocks**.

The designer is what gives you the ability to add buttons, add text, add screens and edit the overall app look. In the App Inventor software the app looks like this:

![App Inventor Software](image)

The app looks different in the software and in your smartphone. This is the how the app looks in our smartphone:

![App Smartphone View](image)
Blocks
Then, you have the blocks section. The blocks sections is what allows to create custom functionality for your app, so when you press the buttons it actually does something with that information. These are the blocks for this app (click on the image to zoom):

Installing the App
To install the app in your smartphone, go to the Build tab.
• You can either generate a QR code that you can scan with your smartphone and automatically install the app in your smartphone.
• Or you can download the .apk file, connect your smartphone to your computer and move the .apk file to the phone.

Simply follow the installation wizard to install the App and it’s done!

**Code**

Download or copy the following code to your Arduino IDE, and upload it to your Arduino board. Make sure you have the right Board and COM port selected.

Note: before uploading the code, make sure you have the TX and RX pins disconnected from the bluetooth module!

```c
/*
 * Rui Santos
 * Complete Project Details http://randomnerdtutorials.com
 */
#define max_char 12
char message[max_char]; // stores your message
char r_char; // reads each character
byte index = 0; // defines the position into your array
int i;
int redPin = 11; // Red RGB pin -> D11
int greenPin = 10; // Green RGB pin -> D10
int bluePin = 9; // Blue RGB pin -> D9
int redValue = 255; // Red RGB pin -> D11
int greenValue = 255; // Green RGB pin -> D10
int blueValue = 255; // Blue RGB pin -> D9
String redTempValue; // Red RGB pin -> D11
String greenTempValue; // Green RGB pin -> D10
String blueTempValue; // Blue RGB pin -> D9
int flag = 0;
char currentColor;

void setup() {
    pinMode(redPin,OUTPUT);
    pinMode(bluePin,OUTPUT);
    pinMode(greenPin, OUTPUT);
    // initialize serial communication at 9600 bits per second:
    Serial.begin(9600);
}

void loop() {
    // while is reading the message
    while(Serial.available() > 0) {
        flag = 0;
        // the message can have up to 12 characters
        if(index < (max_char-1)) {
            r_char = Serial.read(); // Reads a character
            message[index] = r_char; // Stores the character in message array
            if(r_char=='R') {
                currentColor = 'R';
                redTempValue = "";
            } else if(r_char=='G') {
                currentColor = 'G';
                greenTempValue = "";
            } else if(r_char=='B') {
                currentColor = 'B';
                blueTempValue = "";
            }
            if(currentColor == 'R' && r_char!='R') {
                redTempValue += r_char;
            } else if(currentColor == 'G' && r_char!='G') {
                greenTempValue += r_char;
            } else if(currentColor == 'B' && r_char!='B') {
                blueTempValue += r_char;
            }
            index++; // Increment position
            message[index] = "\0"; // Delete the last position
        }
    }
    if(flag == 0) {
        analogWrite(redPin, 255-redTempValue.toInt());
        analogWrite(greenPin, 255-greenTempValue.toInt());
        analogWrite(bluePin, 255-blueTempValue.toInt());
        /*Serial.print('R');
        Serial.println(redTempValue);
        Serial.print('G');
        Serial.println(greenTempValue);
        Serial.print('B');
        Serial.println(blueTempValue);*/
    }
}
```

Serial.println(greenTempValue);
Serial.print('B');
Serial.println(blueTempValue);
Serial.print("MESSAGE ");/*
Serial.println(message);
flag=1;
for(i=0; i<12; i++){
    message[i] = '\0';
}
//resets the index
index=0;
}
}

view raw Projects/MIT-App-Inventor/Control_RGB_LED.ino

Schematics

Follow the schematic diagram in the following figure to wire your circuit.

Note: If you’re using an RGB LED common cathode, you need to connect the longer lead to GND.

Important: Here’s the bluetooth module connections to the Arduino:

1. Bluetooth module TX connects to Arduino RX
2. Bluetooth module RX connects to Arduino TX

Here’s how your circuit should look:
**Launching the App**

If you haven’t generated the .apk file in a previous step, you can [click here to download the .apk file](#) (which is the Android App installation file).

Move that file to your smartphone and open it.

Follow the installation wizard to install the app.

Turn on your smartphone’s Bluetooth.
Make sure you pair your smartphone with the bluetooth module – search for paired devices in your smartphone’s bluetooth settings.

Then, open the newly installed app.
Tap on the **Connect bluetooth** button to connect via bluetooth to the arduino bluetooth module.
Troubleshooting

1. I can't upload code to my Arduino board.

Check if you have the TX and RX cables from the bluetooth module disconnected.

When you're uploading code to your Arduino you should disconnect the TX and RX cables from the bluetooth module. These pins are needed for serial communication between the Arduino and your computer.

2. I can't find my bluetooth module device.

Make sure you have paired your smartphone with your bluetooth module. Go to your bluetooth settings and search for the available devices. Your bluetooth module device should appear (it’s often called: linvor, HC-06, HC-04, HC-05 ...). Pair with it. If it asks for a password, it’s 1234.

3. The app doesn't interact with the Arduino.

If your Android app is connected to your bluetooth module, it should display the “Connected” message (as shown below). Otherwise, press the “Connect Bluetooth” to establish a bluetooth communication.

Double-check your bluetooth module connections:
1. Bluetooth module TX connects to Arduino RX
2. Bluetooth module RX connects to Arduino TX

4. My bluetooth module is asking for a password.

If your bluetooth module asks for a password, type 1234.