50+ Apps with Raspberry Pi, ESP32 and Arduino

MIT App Inventor Projects

Dogan Ibrahim

elektor

LEARN DESIGN SHARE

MIT App Inventor Projects

50+ Android and iOS Apps with Rasberry Pi, ESP32 and Arduino

Dogan Ibrahim

elektor

LEARN DESIGN SHARE

● This is an Elektor Publication. Elektor is the media brand of
Elektor International Media B.V.
78 York Street, London W1H 1DP, UK
Phone: (+44) (0)20 7692 8344

● British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

● **ISBN 978-1-907920-89-9**
EISBN 978-1-907920-97-4
EPUB 978-1-907920-96-7

© Copyright 2020: Elektor International Media b.v.
Prepress Production: D-Vision, Julian van den Berg
First published in the United Kingdom 2020

Preface

Statistics show that the number of smartphones sold to end-users in the last decade from 2007 to 2017 has been steadily increasing. In 2016, around 1.5 billion smartphones were sold to end-users worldwide. In 2017 this number increased to around 1.54 billion, which is a significant increase over just one year. In the fourth quarter of 2016, 81.7% of all smartphones sold to end users were phones with operating Android. This number increased to 85.9% in the first quarter of 2018, 85.9% (source: https://www.statista.com).

Developing apps for mobile phones is not an easy task and can require extensive knowledge of programming. Program development also takes a considerable amount of time. Android-based apps are available in the Google Play Store. Most of these apps are free of charge and can easily be downloaded to your mobile device. The problem with most of these apps is that they are not tested by any authority and therefore are available as you find them. Also, most of these applications contain advertisements which can be annoying for their users. It is however, possible to purchase more professional apps without any built-in advertisements.

This book is about developing apps for Android and iOS compatible mobile phones and tablets using MIT App Inventor. MIT App Inventor is a visual graphical programming language based on blocks. Users drag and drop visual objects (called blocks) to create applications that can run on mobile devices. MIT App Inventor was developed as an educational tool to teach visual programming skills to newcomers, especially children in primary and secondary education. Moreover, it is used in higher education institutions as well as by professional programmers. MIT App Inventor enables users to quickly create complex programs to run on mobile devices. For example, an application program can easily and quickly be developed on a mobile device to display ambient temperature and humidity readings. The development of such a program using classical text-based programming languages requires significant programming skills and will take much longer time to develop.

MIT App Inventor projects can be in standalone mode, or use an external processor. In standalone mode, the developed application runs on the mobile device only (e.g. Android or iOS). In external processor-based applications, the mobile device communicates with an external microcontroller-based processor, such as a Raspberry Pi, Arduino, ESP8266, ESP32, etc.

In this book, many tested and fully working projects have been developed both in standalone mode and using an external processor. Full design steps, block programs, and QR codes are given for all projects. In external processor-based applications (for example using the Raspberry Pi), block diagrams, circuit diagrams, and full program listings are provided with complete documentation for all projects.

Users will find all program listings on the Web site of this book. The MIT App Inventor programs can easily be imported to your MIT App Inventor projects. Additionally, the programs of the external processor-based projects can be uploaded to the appropriate external processors to save you the time of typing them.

Although all of the projects have been tested using an Android mobile phone, they should equally work on Android tablets and iOS compatible mobile phones and tablets without any changes.
I hope you like reading the book and find it useful for your next Android-based apps project.

Prof Dr Dogan Ibrahim
London, 2020

Chapter 1 • Introduction

1.1 Text-Based Programming

Traditionally computer programming has been text-based where the programmer writes sets of instructions in text form. These instructions are then compiled and turned into a binary form that can be understood by a computer. There are many text-based programming languages in use today, such as Basic, C, Pascal, etc. Text-based coding has traditionally been used in teaching the concepts of programming. When we look at the text-based programming languages, we notice that they are designed by engineers, for engineers. This makes coding difficult and learners have to spend many long times of training to learn to program efficiently. For several decades many people have tried to address this problem and designed various programming languages to make the task of programming easier. Most of these programming languages have been text-based. In the last decade, programming languages such as Visual Basic, Visual C++, and Visual C# have been popular. Although these languages are text-based they provide a graphical user interface (GUI) type interface where users can place boxes, labels, buttons, sliders, etc on the screen to make the end product more user-friendly. The code under these programming languages is still text-based where the user is expected to write the code to control the components placed on the screen.

1.2 Block-Based Visual Programming

In recent years there has been an increase in the development and use of block-based visual programming tools as the solution to the above problem. The idea here is that the programmer is given a set of visual blocks and all that is required is to connect these building blocks in a logical way to create the required application program. Block-based programming has been introduced to children event at the primary education level and recent surveys show that children at an early age develop interesting applications. Block-based programming is currently used all over the world and it is estimated that over 50 million users actively use block coding.

Scratch was one of the early block-based programming languages which are currently used by children of all ages. It is an online tool where children can create projects by joining blocks. Figure 1.1 shows the online screen of Scratch. Using Scratch, children can mix various tools in their programs, such as music, sound effects, graphics, etc. Scratch programming environment consists of three main sections: a *stage area*, *blocks of palettes*, and a **coding area**. Users bring the required palettes into the coding area and join them to make the final code. The stage area shows the results, such as the animation. Scratch is very popular in the United Kingdom and the United States and there are several *coding clubs* that children join to share their projects with others. Many children create interesting applications including simple but interesting games as well. Scratch is used as the introductory computer programming language in many primary and secondary schools. After gaining experience with Scratch, children are introduced to Python (and Java) as their second language. Scratch is also used in some higher education institutes, such as it is used in the first week of the Harvard University introductory computer science course.

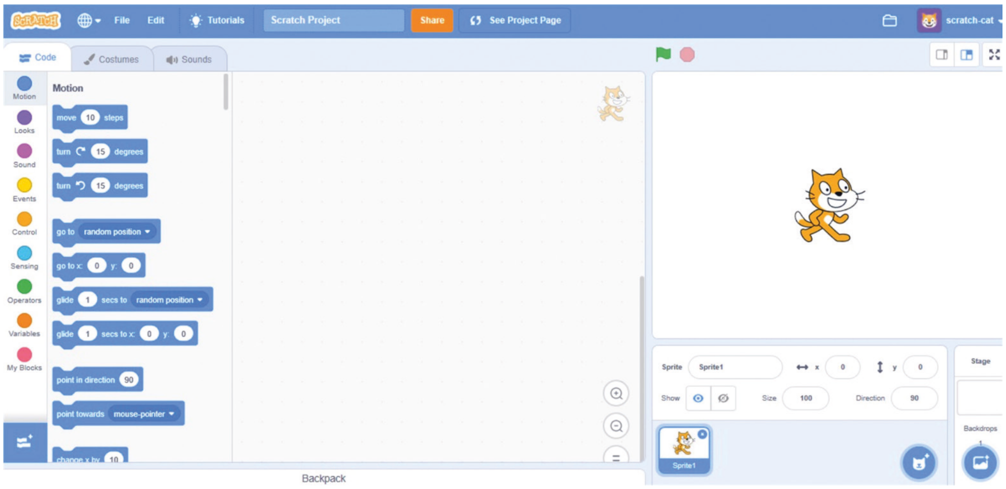


Figure 1.1 Scratch development environment

Many people claim that the block-based visual programming tool does not teach the principles of programming. This is not true since children at an early stage understand the principles of programming and it becomes easier for them to develop complex text-based programs in later life. When people hear of block-based programming, they tend to associate it with teaching children or beginners to programming languages. Although block-based programming is popular among children, it can also be used by adults and professional programmers to develop projects quickly and with little effort.

Block-based programming has the advantage that it is easy to modify a program because all that is required is to manipulate the blocks. Another very important advantage of block-based programming is that complex programs can be developed in a few seconds instead of hours required with text-based languages. For example, consider the project where it may be required to read the ambient temperature and send it to someone's mobile phone as an SMS message. This project will probably take less than 30 minutes to develop using a block-based programming language, assuming that there is a block to handle SMS messages. The same program, when written using text-based programming, can easily take several days to develop and test. This is because the SMS block hides away all of the complexities of establishing communication with the receiver and sending packets of data. Another advantage of block-based programming is that the users do not have to memorize the syntax of the language. For example, in a text-based programming language missing a semicolon in a program can result in errors which sometimes can take some time to find out the cause of the error.

This book is about using the MIT App Inventor to develop projects. This is a block-based web programming language that is currently very popular all over the world. MIT App Inventor is an online tool and it is free of charge (there is also an offline version). It was developed originally by Google, but now it is maintained by the Massachusetts Institute of Technology (MIT). App Inventor allows people of all ages to use to develop programs for mobile phones. It supports both the Android and the iOS operating systems (iOS support started by the 8th of July, 2019). The final program is compatible with both operating systems and can be installed and used on both Android and iOS compatible mobile phones and tablets. In this book, only the Android operating system is considered since the developed programs can be uploaded to both operating systems and they are fully compatible and will work without any modifications.

MIT App Inventor is GUI based and is similar to Scratch and StarLogo, where developers drag and drop and join visual blocks to create an application. Many blocks are offered in the MIT App Inventor that enable users to create projects using components such as text boxes, labels, buttons, sliders, checkboxes, switches, notifiers, camcorders, cameras, text to speech components, speech recognizer, drawing and animation components, web tools, sensors, maps, storage components, Bluetooth connectivity and so on. These components are organized under the heading **Palette** and are placed at the left-hand side of the MIT App Inventor startup screen as shown in Figure 1.2.

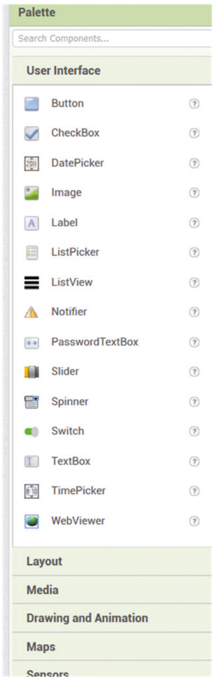


Figure 1.2 MIT App Inventor components

When a new application is started, a mobile phone image is shown in the middle part of the screen. The development of a project is in two stages: **Designer** and **Blocks**. A project starts in the Designer stage where the user places the required components onto the mobile phone image to build the view of the final application. Some components are hidden and are only shown outside at the bottom of the phone image. After designing the screen layout, the user clicks the Blocks menu where the second stage of the development starts. Here, the block program is constructed by clicking, dragging, dropping and joining the required blocks on the mobile phone image.

When the design is complete it is required to test the project. Here, the user has the option of either using a built-in Emulator, to connect to the mobile phone using a USB cable, or to upload the developed application to the mobile phone using a wireless Wi-Fi link. Emulator option is useful if the user has no mobile phone at the time of the development, or if an Android or iOS compatible mobile phone is not available. The second option is useful if there is no Wi-Fi connection where the developed application is uploaded to the mobile phone via a USB cable. The third option is the preferred option where the developed block program is uploaded to the mobile phone using a Wi-Fi link. In this book, we will be using this third option to upload the program.

Perhaps the easiest way to understand how to use the MIT App Inventor is to look at a very simple example. In the example below the stages of the development are summarized.

1.3 Example

This is perhaps the simplest example one can build. This example aims to show the stages of developing a project using the MIT App Inventor.

In this example, we will insert a button and a textbox on the mobile phone screen. Pressing the button will display the message **Hello there!** in the textbox.

Design: Click menu option **Designer**. Figure 1.3 shows the mobile phone screen where a button and a textbox are placed on the mobile phone screen.

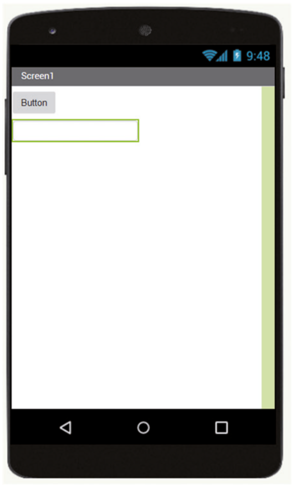


Figure 1.3 Design of the project

Block Program: Click menu option **Blocks**. Figure 1.4 shows the block program. It is clear from these blocks that when the button is clicked the message **Hello there!** will be displayed in the textbox.

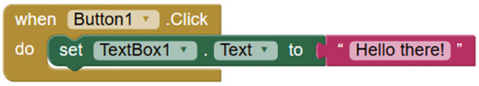


Figure 1.4 Block program of the example

Build the program: Click menu option **Build** in the top menu and select **App (provide QR code for .apk)**. Wait until the QR code of the application is generated as shown in Figure 1.5

http://ai2.appinventor.mit.edu/b/fk5y

Figure 1.5 QR code of the application

Upload the program to your mobile phone: Start apps **MIT AI2 Companion** (we will see in later chapters) on your mobile phone and scan the QR code. Click to install the application on your mobile phone

Testing: Start the application on your mobile device to test it. The output in this example is shown in Figure 1.6.

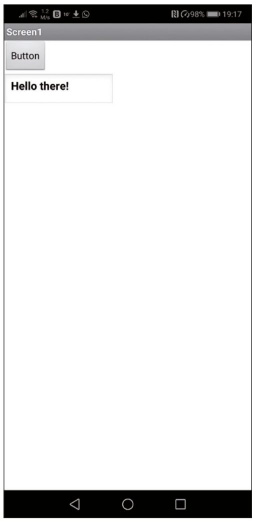


Figure 1.6 Testing the application

MIT App Inventor projects can either be in standalone mode, or they use an external processor. In standalone mode, the developed application runs only on the mobile device (e.g. Android or iOS). In external processor-based applications, the mobile device communicates with an external microcontroller-based processor, such as Raspberry Pi, Arduino, ESP8266, ESP32, etc.

In this book, many tested and fully working projects have been developed both in standalone mode and also using an external processor. Full design steps, block programs, and QR codes are given for all projects. In external processor-based applications (for example using the Raspberry Pi), block diagrams, circuit diagrams, and full program listings are given with complete documentation for all projects.

Users will find all the program listing on the Web site of the book. The MIT App Inventor programs can easily be imported to your MIT App Inventor projects. Additionally, the programs of the external processor-based projects can be uploaded to the appropriate external processor to save you the time of typing them.

Chapter 2 • Setting up the MIT App Inventor

2.1 Overview

In this chapter, we will look at the various ways MIT App Inventor can be used to create projects. The nice thing about MIT App Inventor (called the App Inventor in this book for short) is that the PC software is cloud-based and there is no need to install it before use. Note that in this and future chapters, all references to the Android operating system are also valid for iOS.

The following are required to create applications using the App Inventor:

A Computer and Operating System:

- Macintosh (with Intel processor): Mac OS X 10.5 or higher
- Windows: Windows XP, Windows Vista, Windows 7 or higher
- GNU/Linux: Ubuntu 8 or higher, Debian 5 or higher (Note: GNU/Linux live development is only supported for WiFi connections between computer and Android device.)

Internet Access and Internet Browser:

- Mozilla Firefox 3.6 or higher
- Apple Safari 5.0 or higher
- Google Chrome 4.0 or higher
- Microsoft Internet Explorer is not supported

Android Compatible Phone or Tablet:

- Android Operating System 2.3 ("Gingerbread") or higher

Optionally:

- You can use the Emulator if you do not have an Android compatible phone or tablet
- You will need to connect the Android device to a PC using a USB cable if you do not have a Wi-Fi link.

There are 3 options for setting up and using App Inventor. These are described in the next section briefly.

2.2 Setting Up the App Inventor

Option 1 - Using an Android Device with Wi-Fi

This is the recommended option where software is developed on a PC and then uploaded (installed) to an Android device using Wi-Fi for testing (see Figure 2.1).

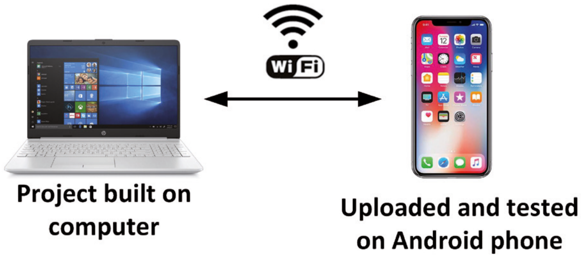

Wi Fi
Project built on computer
Uploaded and tested on Android phone

Figure 2.1 Option 1 – Using the Android device with Wi-Fi link

This option requires the apps **MIT AI2 Companion** to be installed from the Play Store to your Android device as shown in Figure 2.2.

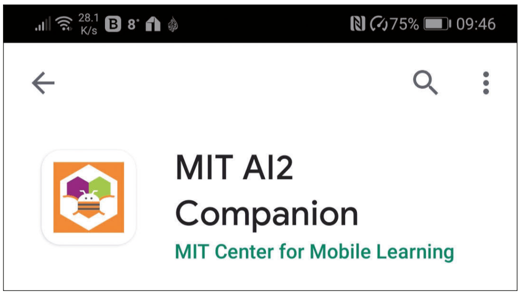


Figure 2.2 Install the MIT AI2 Companion to your Android device

After you create your project, the next step is to upload (install) it to your Android device for testing. The steps to upload your project to the Android device are as follows (these steps will become clearer when we look at the steps to create a simple project in the next section):

- After the project is complete, click **Connect** and then **AI Companion** as shown in Figure 2.3, or, choose **Build** and then **App (provide QR code for .apk)** to install the project permanently on your Android device.

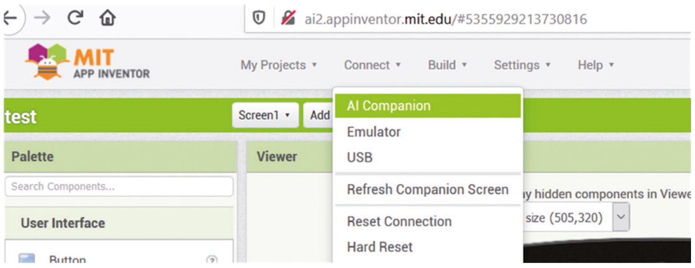
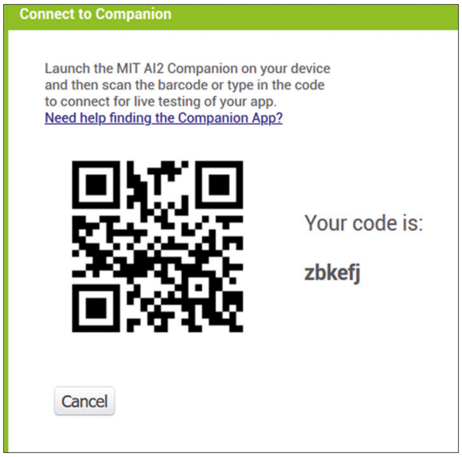


Figure 2.3 Click Connect and then AI Companion

- Wait until the project is compiled.

- A dialog with a QR code will be displayed on your PC screen as shown in Figure 2.4



Connect to Companion

Launch the MIT AI2 Companion on your device and then scan the barcode or type in the code to connect for live testing of your app.
Need help finding the Companion App?

Your code is:

zbkefj

Figure 2.4 The QR code

- Start the apps **MIT AI2 Companion** on your Android device and click the **scan the displayed QR code** (note that the QR code is only valid for 2 hours) and hold your device to scan the displayed QR code

- After a few seconds, the project will be uploaded to your device. Follow the instructions to install the project.

- You can now test your project on the Android device

Alternatively, you can enter the 6-character code displayed next to the QR code (Figure 2.4) to your Android device to upload the project.

Option 2 - Using an Android Device With Direct Connection to the PC
In this option, it is assumed that there is no Wi-Fi link. The Android device is connected to the PC using a suitable USB cable as shown in Figure 2.5.

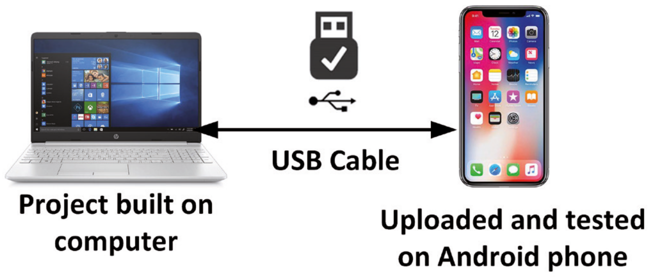

USB Cable
Project built on computer
Uploaded and tested on Android phone

Figure 2.5 Option 2 – Connect the Android device to the PC

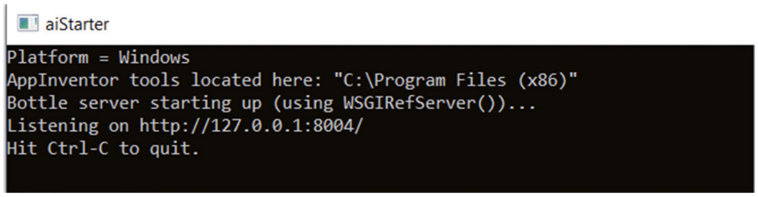
Here, the application is built on the computer and is uploaded to the Android device through a USB cable. This option requires a driver to be loaded to the Windows-based PCs (there is no need to load a driver for the Mac or Linux machines). The steps are as follows (you must install from an account that has administrator privileges). See also the following link:

https://appinventor.mit.edu/explore/ai2/setup-device-usb

- Download the installer from the link:

  appinv.us/aisetup_windows

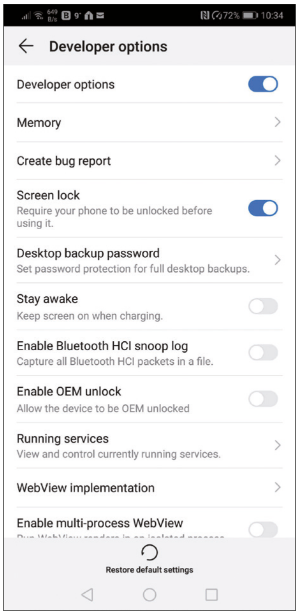
- Locate the file **MIT_Appinventor_Tools_2.3.0_win_setup.exe** in your Downloads file
- Open the file and click through the steps to install the file. It is recommendable to not change the installation directory. You will find the path to the file is: **C:\Program Files\Appinventor\commands-for-Appinventor**
- Download and install the apps MIT AI2 Companion from the Play Store to your Android device (see Figure 2.2)
- Using the USB cable (also the emulator) requires the use of the program named **aiStarter**. There should be a shortcut on your Windows-based computer to this program (On a Mac, aiStarter runs automatically when you log in), or you should be able to locate it in your **Start** menu. Start the **aiStarter**, you should see the **aiStarter** icon in your taskbar. You should see a window as shown in Figure 2.6 (On GNU/Linux, **aiStarter** will be in the folder /usr/google/commands-for-Appinventor and you'll need to launch it manually. You can launch it from the command line with **/usr/google/appinventor/commands-for-Appinventor/aiStarter)**



```
Platform = Windows
AppInventor tools located here: "C:\Program Files (x86)"
Bottle server starting up (using WSGIRefServer())...
Listening on http://127.0.0.1:8004/
Hit Ctrl-C to quit.
```

Figure 2.6 aiStarter startup window

- We now have to enable USB Debugging on the Android device. Go to **Settings**, then **System,** and enable both the **Developer options** and **USB debugging** (see Figure 2.7). USB debugging is normally on the same page but under the heading **DEBUGGING**. You may find that the Developer Options are hidden (especially on Android 4.2 and newer) by default. If this is the case, go to **Settings** and **About phone**, and tap **Build number** 7 times. Then return to the previous screen to find and enable **Developer options**, including USB Debugging.


Developer options

Developer options

Memory

Create bug report

Screen lock
Require your phone to be unlocked before using it.

Desktop backup password
Set password protection for full desktop backups.

Stay awake
Keep screen on when charging.

Enable Bluetooth HCI snoop log
Capture all Bluetooth HCI packets in a file.

Enable OEM unlock
Allow the device to be OEM unlocked

Running services
View and control currently running services.

WebView implementation

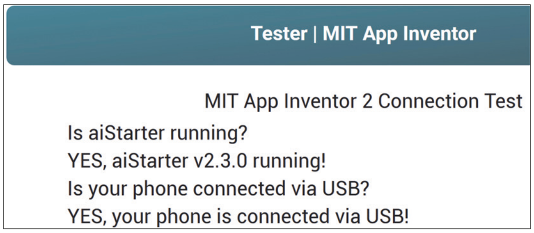
Enable multi-process WebView

Restore default settings

Figure 2.7 Enable the Developer options

- Connect your Android device to your computer using a USB cable, and make sure that it is not mounted as a drive on your computer. You may have to go to **My Computer** (on Windows) and right-click to disconnect any drives (e.g. eject) that were mounted when you connected your Android device. The device should be connected as a **mass storage device** (not as a **media device**, i.e. as transferring files, but not as sending pictures). You may get the message saying **Allow USB Debugging?**. Press **OK**.

- Go to the **Connection Test Page** (appinventor.mit.edu/test.html) to make sure that the connection is ok. You should see the window as in Figure 2.8 after a successful connection.


Tester | MIT App Inventor

MIT App Inventor 2 Connection Test

Is aiStarter running?

YES, aiStarter v2.3.0 running!

Is your phone connected via USB?

YES, your phone is connected via USB!

Figure 2.8 Successful connection between Android and the PC

- Once you complete your project and there is a successful USB connection between the computer and Android device, you should click **Connect** and then **USB** to upload your project to the Android device.

Option 3 - Using the Emulator Without an Android Device
In this option, it is assumed that the user has no Android devices. The application is built on a computer and then emulated using a virtual Android mobile phone which is displayed on the screen, as shown in Figure 2.9 (see link: https://appinventor.mit.edu/explore/ai2/setup-emulator)

Project built on computer and also emulated on computer

Figure 2.9 Using the Emulator without an Android device

You will have to install software on your computer before the emulator can be used. The steps are:

- Download the installer from the link:

  appinv.us/aisetup_windows

- Locate the file **MIT_Appinventor_Tools_2.3.0_win_setup.exe** in your Downloads file

- Open the file and click through the steps to install the file. You are recommended not to change the installation directory. You will find that the path to the file is: **C:\Program Files\Appinventor\commands-for-Appinventor**

- Using the emulator (as with the direct link) requires the aiStarter program to run. This program was installed in the previous step and you should run it manually (On a Mac, aiStarter will start automatically when you log in to your account and it will run in the background). You should see a screen similar to Figure 2.6.

- After you complete your project on the computer, click Connect and then Emulator as shown in Figure 2.10

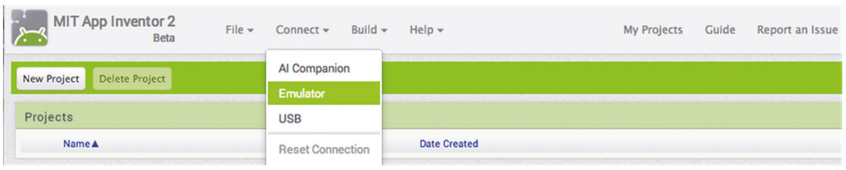
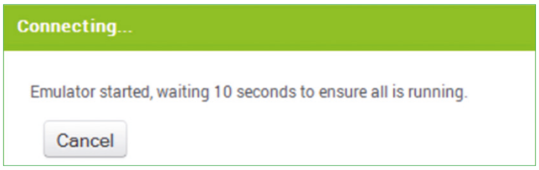


Figure 2.10 Start the emulator

- You should see a message saying the emulator is connecting (see Figure 2.11). You might have to wait a few minutes for the connection to complete.

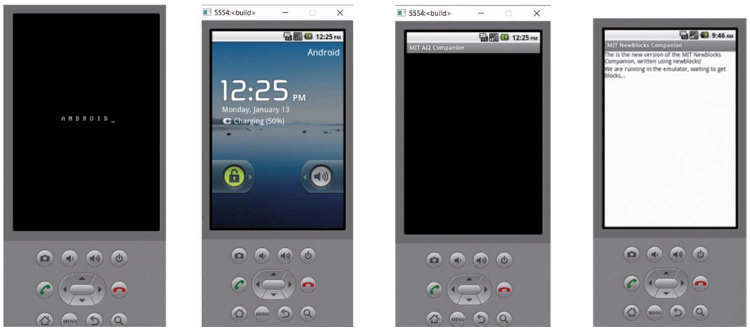


Connecting...

Emulator started, waiting 10 seconds to ensure all is running.

Figure 2.11 Emulator connecting message

- You should see a virtual Android mobile phone displayed with the screen changing until the project screen is displayed, as shown in Figure 2.12.



The is the new version of the MIT Newblocks Companion, written using newblocks!
We are running in the emulator, waiting to get blocks...

Figure 2.12 Displaying the virtual Android phone

Note: if you get **aiStarter** not available messages even though the software has already been started, you should click to run the program **adbrestart** in directory C:\Program Files (x86)\AppInventor\commands-for-appinventor\adbrestart.

2.3 Summary

In this chapter, we learned how to set up the MIT App Inventor software. There are 3 different methods we can test our developed project with, depending on whether or not we have an Android device and Wi-Fi. Method 1 is the most commonly used and it assumes that the user has an Android device and also the device is assumed to be connected to the same Wi-Fi network as the computer where the software has been developed. Method 2 assumes that there is an Android device but no Wi-Fi link. Here, the Android device is connected to the computer using a USB cable. In method 3 it is assumed the user has no Android device and the developed program is emulated on a virtual Android phone on a computer screen. In the next chapter, we will develop some simple projects using MIT App Inventor.

Chapter 3 • Simple MIT App Inventor projects

3.1 Overview

In the last chapter, we learned how to set up MIT App Inventor (it will simply be called App Inventor in this chapter). In this chapter, we will develop simple projects to make readers familiar with the various features of App Inventor. *Note that the QR codes given in the projects were valid only for 2 hours at the time they were created, and they cannot be used to install the apps to your mobile phone. They are only given here for completeness*.

3.2 Starting App Inventor and the Startup Screen

You should use Firefox or Google Chrome for App Inventor project development since Microsoft Internet Explorer is not currently supported. You should create a google email account if you do not already have one. There are several ways that you can start App Inventor:

- Enter the words **MIT App Inventor** to the Google search engine and click to start the project
- Enter the following link to your web browser:

  https://appinventor.mit.edu/explore/get-started

- You should be presented with the startup screen: **Getting Started with MIT App Inventor screen**
- Click **Create Apps!** On the top left-hand side of the screen to start App Inventor.

The Startup Screen

Figure 3.1 shows the App Inventor startup screen. The middle part of the screen is the Viewer section which is the working space and it shows the layout of a mobile phone. On the left-hand side, you will see a Palette with many components that you can drag to the Viewer. The Palettes are under different categories, such as User Interface, Layout, Media, Drawing, and Animation, etc. On the right-hand side of the screen, we have the Properties section, where the properties of the components (e.g. colour, shape, behaviour, etc) can be configured. On the top part of the screen, we have several menu options. File is used to create a new file, Connect is used to connect to the MIT AI2 Companion apps, or the Emulator, or an Android device using a USB cable. A list of the user projects can be displayed by clicking My Projects. Two very important menu options are the Designer and Blocks. Option Designer is selected when we wish to develop a new project by placing various components onto the Viewer. The default screen when App Inventor is started is the Designer screen. Option Blocks is selected when we wish to develop visual programs to control the components laid on the Viewer. App Inventor programs are in the form of visual blocks that are interconnected (e.g. snapped together) to accomplish a task. The user does not need to have any text-based programming skills to use App Inventor. This will become clearer when we develop projects.

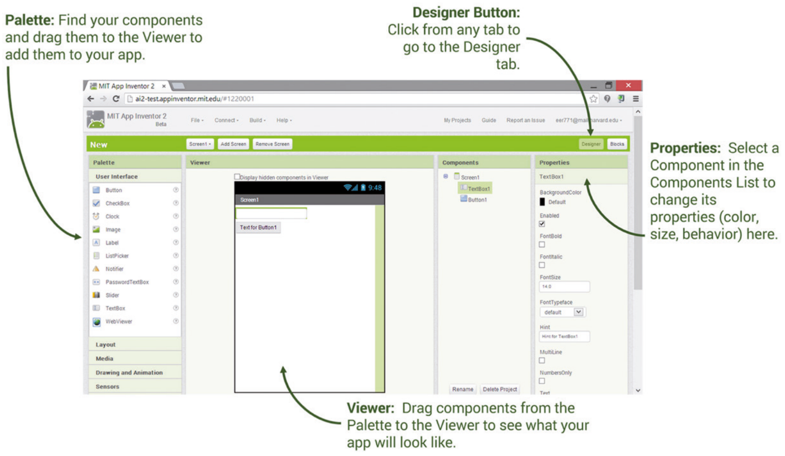

Palette: Find your components and drag them to the Viewer to add them to your app.
Designer Button: Click from any tab to go to the Designer tab.
Properties: Select a Component in the Components List to change its properties (color, size, behavior) here.
Viewer: Drag components from the Palette to the Viewer to see what your app will look like.
MIT App Inventor 2
Palette
Viewer
Components
Properties
Button
CheckBox
Clock
Image
Label
ListPicker
Notifier
PasswordTextBox
Slider
TextBox
WebViewer
Layout
Media
Drawing and Animation
Sensors
Screen1
TextBox1
Button1
Text for Button1
Designer
Blocks

Figure 3.1 The Designer startup screen (taken from the MIT App Inventor web site)

The Blocks screen is shown in Figure 3.2. The middle part is the Viewer where the visual programming components are selected from the component-specific drawers on the left-hand side of the screen.

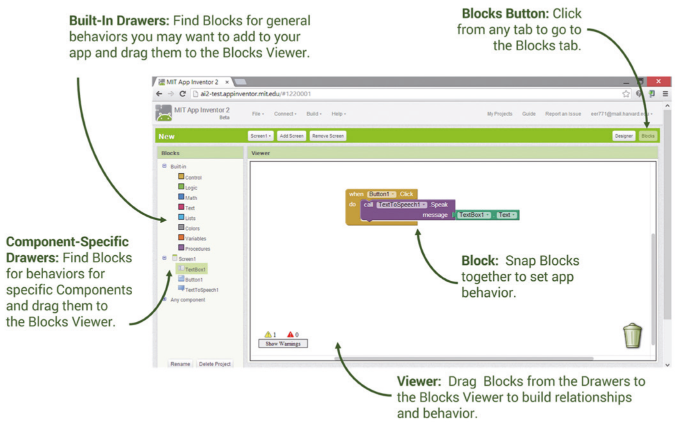

Built-In Drawers: Find Blocks for general behaviors you may want to add to your app and drag them to the Blocks Viewer.
Blocks Button: Click from any tab to go to the Blocks tab.
Component-Specific Drawers: Find Blocks for behaviors for specific Components and drag them to the Blocks Viewer.
Block: Snap Blocks together to set app behavior.
Viewer: Drag Blocks from the Drawers to the Blocks Viewer to build relationships and behavior.
when Button1 .Click
do call TextToSpeech1 .Speak
message TextBox1 . Text
Control
Logic
Math
Text
Lists
Colors
Variables
Procedures
Screen1
TextBox1
Button1
TextToSpeech1
Any component
Show Warnings

Figure 3.2 The Blocks screen (taken from the MIT App Inventor web site)

In the projects in this book an Android mobile phone is used to test the developed projects. Additionally, a Windows 10 based laptop is used to develop the projects using the App Inventor. It is assumed that the mobile phone is connected to the same Wi-Fi router as the laptop.

3.3 Project 1 – Using a Button to Generate Sound

Description: In this project, we will have a button on our Android screen. When the button is clicked a bell sound will be generated by the Android phone which can be heard on the speakers.

Aim: This project aims to show how a button can be used together with a sound clip.

Steps: The steps to develop the project are given below:

- Start App Inventor and click **Create Apps!**
- Login by entering your email address and password
- Click **Start new project**
- Give a name to your project, e.g. **BellSound** and click **OK**
- You will be presented with the Designer menu
- Click and drag a **Button** from the **User Interface** to the **Viewer**
- Set the properties of the button on the right-hand side as follows (only the required changes are shown):

  FontBold: tick
  Text: Click to sound

- Figure 3.3 shows the screen so far

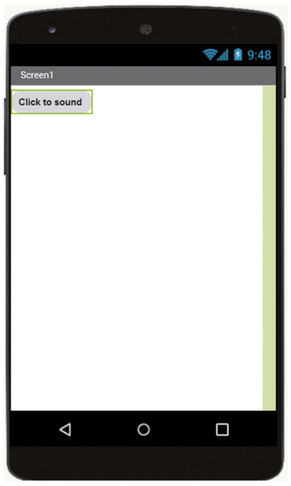


Figure 3.3 Place a button onto the Viewer

- Click **Media** in the Palette and drag and drop **Sound** onto the **Viewer**. Notice that the sound icon is displayed outside the mobile phone.
- Now we have to upload our sound file. There are many free of charge MP3 sound files that can be used in projects. In this project, the telephone bell sound file

named **96521_kleinhirn2000_reception-bell1.mp3** is used from the site https://freesound.org/people/Werra/sounds/78565/. This generates a short bell sound. Click to download the file in a folder.

- Make sure that Sound1 is highlighted on the right-hand side and click **Upload File**. Browse and select the sound file you have downloaded. You should see the filename displayed just above Upload File

This completes the visual design of the project in the Designer menu. Now, we have to select the Blocks menu and do the visual programming part of the project. The steps are:

- Click **Blocks**

- Click **Button1** from the Palette on the left-hand side

- Click on block **when Button1.Click do** as shown in Figure 3.4. You should now see that this block is displayed in the Viewer

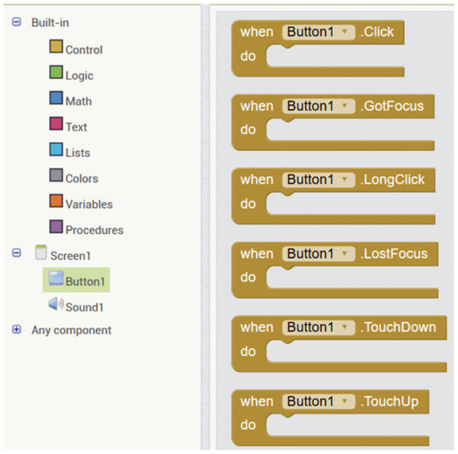


Figure 3.4 Click on when Button1 Click do

- Click **Sound1** on the left-hand side and select **call Sound1.Play**

- Join **call Sound1.Play** with the **when Button1.Click do** as shown in Figure 3.5

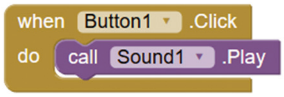


Figure 3.5 Join the two blocks

This completes the design of the visual program. As can be seen from Figure 3.5, when the button is pressed, we are telling the program to play sound1.

We should now compile and upload our project to the Android mobile phone. The steps are:

- Click **Build** and then **App (provide QR code for .apk)** and wait until the project is compiled
- You should be presented with the QR code as shown in Figure 3.6 (this code is valid only for 2 hours and is given here for completeness). Start the apps **MIT AI2 Companion** on your mobile phone and click **scan QR code**. Hold the phone to view the QR code on the screen
- Accept to download the project to your Android device and then click **Install** to install it.
- You might see the message **Blocked by Play Protect**. Click **INSTALL ANYWAY** and then click **Open** to run the project as an app on your Android device
- You should see that the app with the name **BellSound** is installed on your Android device
- You should hear the bell sound when you click the button on your mobile phone.

http://ai2.appinventor.mit.edu/b/4y36

Figure 3.6 The QR code of the project

You may want to use the emulator if you do not have an Android device. The steps are as follows:

- Click to start program **aiStarter** o your computer
- Click **Connect** followed by **Emulator** and wait until the emulator is connected as shown in Figure 3.7