No Texting While Busy

|  |  |
| --- | --- |
| This tutorial uses the App Inventor Texting Component to receive and send text messages. This version is a variation of the [No Text While Driving](http://www.appinventor.org/content/CourseInABox/textingLocation/NoTexting) tutorial and it shows how to use the ***Texting component over Wifi*** (for devices with no Sim card or Mobile service plan). The app itself is bare bones. It is designed to respond automatically to messages received ***while the user is busy.*** The app simply displays the message in a log and sends an automatic response. The lesson contains several suggestions for enhancements. Learning: In this lesson you will learn to:  * use App Inventor's Texting component; * work with [Google Voice](https://support.google.com/voice/answer/115061?hl=en), both the mobile app and the Google Voice account; * handle received text messages in App Inventor.   **NOTE: To test this app, it is necessary to package and install it on your device.** | ***[Click to watch Preview Video](http://www.youtube.com/watch?v=Al82272L8tw)*** |

## 

## IMPORTANT UPDATE: Google Voice has been updated and is no longer working for use with the texting component. In order to test apps that require the texting component, disable Google Voice in App Inventor and use an Android device that has cellular service (e.g. an Android cellphone).

## Getting Ready

Open [App Inventor with the No Texting While Busy Template](http://ai2.appinventor.mit.edu/?repo=templates.appinventor.mit.edu/trincoll/csp/unit7/templates/NoTextingWhileBusy/NoTextingWhileBusy.asc) in a separate tab and follow along with the following tutorial. Once the project opens use *Save As* to rename your project ***NoTextingWhileBusyV1****.*

***NOTES****.*

1. *In order to implement this app, you will have to create a Google Voice account online and download and install the Google Voice app on your mobile device. (Directions below)*
2. *To test this app, it is necessary to package and install it on your device.*

## IMPORTANT UPDATE: Google Voice has been updated and is no longer working for use with the texting component. In order to test apps that require the texting component, disable Google Voice in App Inventor and use an Android device that has cellular service (e.g. an Android cellphone).

# No Texting While Busy Tutorial

## Basic Idea

This tutorial uses the App Inventor Texting Component to receive and send text messages. This version is a variation of the [No Text While Driving](http://www.appinventor.org/content/CourseInABox/textingLocation/NoTexting) tutorial and it shows how to use the [Texting component over Wifi](http://appinventor.mit.edu/explore/content/google-voice.html) (for devices with no Sim card or cellular service plan). The app itself is bare bones. **It is designed to respond *automatically* to messages received *while the user is busy.* The app simply displays the message in a log and sends an automatic response**. The lesson contains several suggestions for enhancements.

## No Text While Busy UI

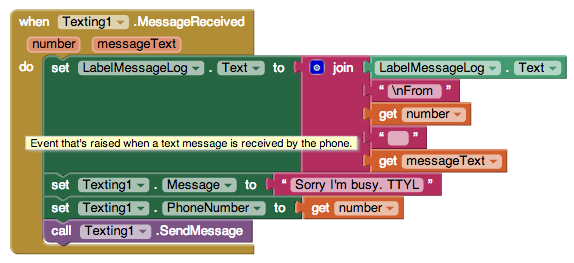
**The User Interface**

1. The very simple UI consists entirely of a *Label* that serves as the message log -- i.e., it is used to display received messages as you see here.
2. The black “Message sent” box that you see on the UI is a notification produced by the Texting component indicating that a message has been sent back to the sender of the received message.
3. ***Non-visible Component:*** A *Texting* component is used to receive and send messages.

# 

# 

## Under the Hood

The following diagram illustrates how the app works. When somebody () sends a text message (“Hi”) to the mobile device running the “No Texting While Busy” app (), the “when Texting.MessageReceived” block () will fire, which will automatically return the message “Sorry I’m busy. TTYL” to the sender.

Hi

Sorry I’m busy. TTYL.

## VoIP Protocol

[Voice over Internet Protocol](http://en.wikipedia.org/wiki/Voice_over_IP) (VoIP) is an application protocol that is used to send telephone calls and text messages over the Internet -- rather than through a mobile carrier such as Verizon or ATT. Text messages sent over VoIP are free up to a certain daily limit.

To send text messages, mobile devices use the [Short Message Service](http://en.wikipedia.org/wiki/Short_Message_Service) (SMS), a service that uses the [Short Message Peer-to-Peer](http://en.wikipedia.org/wiki/Short_Message_Peer-to-Peer) (SMPP) protocol to send text messages from one mobile device to another (peer-to-peer). The SMS service is a very popular, widely-used service. For example, in 2010, more than 6 trillion SMS messages were sent.

SMS message, as you probably know, are 70-160 characters in length. The variation depends on how the text is represented, either as 7-bit, 8-bit, or 16-bit characters. Plain English text can be represented in 7 or 8 bits. But special characters, such as Chinese characters, require 16 bits.

## 

## Google Voice

[Google Voice](http://en.wikipedia.org/wiki/Google_Voice) is an example of a VoIP application. Google provides the Voice service allowing users to send free text messages (up to a daily limit) and inexpensive phone calls. There are many similar services, such as the [What’s App](https://www.whatsapp.com/) service. There is also a Google Voice mobile app, which is available from the Google Play store.

*NOTE: In order to implement this app, you will have to create a Google Voice account online and download and install the Google Voice app on your mobile device.*

## IMPORTANT UPDATE: Google Voice has been updated and is no longer working for use with the texting component. In order to test apps that require the texting component, disable Google Voice in App Inventor and use an Android device that has cellular service (e.g. an Android cellphone).

## Back Under the Hood

Because we are using Google Voice, the details for how this app works are slightly more complicated than the simple picture we showed above. Here’s a more detailed look:

Hi

Sorry I’m busy. TTYL.

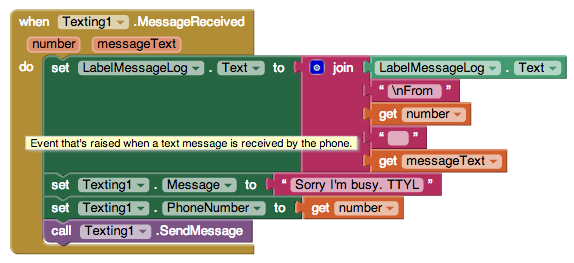
Mobile App

Sorry I’m busy. TTYL.

Hi

Hi

Web App

As you seen in this illustration, when the device () receives a text message, the message (“Hi”) is actually received by the Google Voice mobile app () that you’ve installed on your phone, which will then forward it to your app where it is processed by the *MessageReceived*  block (). This block returns the “Sorry I’m busy” message to the sender, but does so through the Google Voice Web app (). In other words:

* The Google Voice Mobile app installed on the tablet is used to ***receive***  messages, and
* Your Google Voice Web app -- your online Google Voice account -- is used by the tablet to ***send*** messages.

## Setting up a Google Voice Account

*Note: If you have an Android phone with cell service, you do NOT need to set up Google Voice.*

Here are the steps to follow to set up a Google Voice Account:

|  |  |
| --- | --- |
| 1. Sign in to your Gmail account and click on the Google menu, which looks like a tic-tac-toe board, on the top right of your browser: |  |
| 2. Click on the *More* tab at the bottom of the the Google pop-up menu. | GoogleMenuBar.png |
| 3. Click on the *Even More* tab. | GoogleEvenMore.png |
| 4. Find the *Voice* app under the Home and Office menu. | GoogleHomeOffice.png |
| 5. This will bring you to the Google Voice web site, which will look something like this: | GoogleVoice |
| 6. Click ‘Get a Voice number’ and follow the instructions there for setting up your Google Voice phone number and your account. | *NOTE: If you are doing this on a tablet, you will use your Google Voice phone number as the tablet’s phone number. You will need to verify you are a person by using an existing phone number (e.g. teacher’s cell phone number)* |

## Installing the Google Voice App on the Device

Once you’ve set up a Google Voice account, you will want to download and install the Google Voice mobile app on your tablet or phone.

|  |  |
| --- | --- |
| 1. Open your device and run the *PlayStore app.* |  |
| 2. Search for Google Voice. | Google-talking-to-carriers-about-Google-Voice-integration.jpg |
| 3. Download and Install the Voice app. |  |

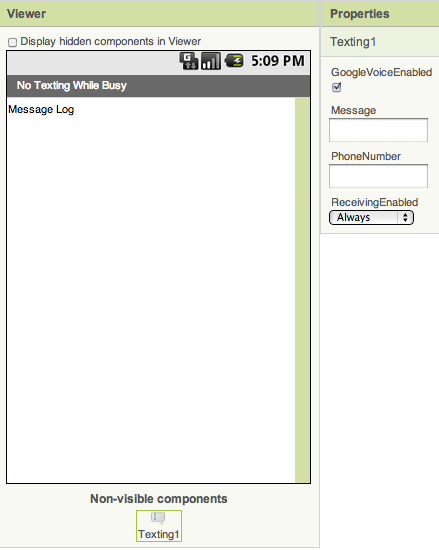
## Configuring the Voice App

Once you’ve installed the Voice app, you will have to configure it:

|  |  |
| --- | --- |
| 1. Start the Voice app and go to the Settings menu. | AppSettings.png |
| 2. Select the ***Sync and Notifications*** setting | SyncAndNotification.png |
| 3. Set the ***Text Notifications*** to one of the following options depending on what type of device you have:   * Tablet: Notifications **on** * Phone: Notifications **off** | This controls whether you want Google Voice to post notifications in the device’s status bar when it receives a text message. These notifications would be in addition to any notifications your *No Texting* app itself provides. |

## The Texting Component

Now that you have successfully created your Google Voice account and installed the Google Voice mobile app on your device, we are ready to create the *No Texting While Busy* app.

The [Texting](http://ai2.appinventor.mit.edu/reference/components/social.html#Texting) component is a non-visible component that is used to send and receive text messages. As you can see here it has four properties:

* **GoogleVoiceEnabled**: If you want to use the component over Wifi, you must have it enabled for Google Voice. Details are provided below on how to set it up for this usage. (Note: If you have an Android cell phone, you should ***uncheck*** this option since you won’t need Google Voice.)
* **Message**: A default message may be specified or left blank.
* **PhoneNumber**: Messages will be sent to this default phone number, which may be left blank.
* **ReceivingEnabled:** This property controls how the phone responds to incoming (received) messages. There are three possible settings:

1. **ReceivingEnabled: OFF.** This corresponds to the numerical value 1. When set to ‘Off’, received messages will be ignored by the app.
2. **ReceivingEnabled: Foreground.** This corresponds to the numerical value 2. When set to ‘Foreground’, the Texting component will receive messages only when it is running in the foreground -- i.e., when it is visible on the phone. This behavior corresponds to the original version of the Texting component.
3. **ReceivingEnabled: Always.** This corresponds to numerical value 3. When set to ‘Always’, the Texting component will receive messages when it is running in the foreground, but also when it is not running. An app that uses a Texting component set to ‘Always’ can receive messages as long as it is installed on the phone. Its behavior in this case is to put a *notification* in the device’s status bar:

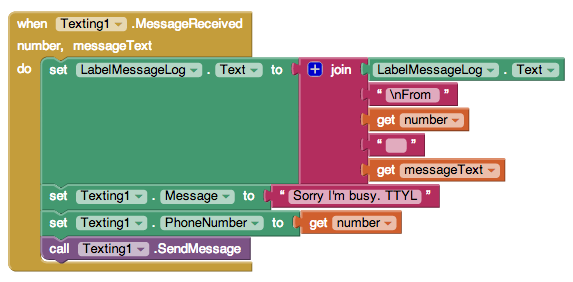


When the user pulls down the status bar and taps on the notification, the app will start up and appear in the foreground. We’ll try this version after we build the app. **For now, make sure you have the Texting component’s properties set as follows:**

* + **GoogleVoiceEnabled: True IMPORTANT UPDATE: Set this property to FALSE - Google Voice has been updated and is no longer working for use with the texting component. In order to test apps that require the texting component, disable Google Voice in App Inventor and use an Android device that has cellular service (e.g. an Android cellphone).**
  + **ReceivingEnabled: Always**

## The Blocks

### This app contains only one main block, which handles all its behavior.



This is the ***MessageReceived*** event handler. Notice that it has two properties -- the ***number***  that sent the message and the ***messageText*** that was received from that number.

If *Texting*.*ReceivingEnabled* property is set to ***Foreground*** and the app is running in the foreground, then this block will fire. (If it is set to ***Always*** and the app has been installed on the phone, then the user will be notified of incoming messages, but this block won’t fire until the user clicks the notification.) It performs 4 simple tasks:

1. Append the incoming message and phone number to the *LabelMessageLog.*
2. Set the Texting component’s message to “Sorry I’m busy. TTYL” (which is short for ‘talk to you later’.)
3. Set the Texting component’s phone number to the sender’s phone number.
4. Send a reply to the sender.

## Setting Up for a Device with Mobile Service

If you are using a device that has mobile service, then no special setup is required. You can use this app, with ***GoogleVoiceEnabled*** set to *false*.

## Setting Up to Use Texting over the Wifi/Internet

If you are using a tablet or a phone that does not have mobile service you can use this app if you follow the instructions given above.

## Running and Testing the App

**In order to test this app *you will need to package/build the app* -- it cannot be tested over wifi with the Companion.** Also, you should not have multiple versions of the app on your device. If you make changes, uninstall the old app from your device and then reinstall the app with the changes. For testing, It is best to do one of the following options:

* Use two texting devices. [Install the app](https://docs.google.com/document/d/1l5jAubqxUZEKbIOPIuGYk26XnxxblDkCidxwRPAuQCM/edit?usp=sharing) on one device and text the app from the second device. The sending device should receive the automatic reply. (NOTE: Don’t activate the app on both devices. If the sending device is also running the app, it will send an automatic reply, which will start an endless cycle of sending and receiving!)
* Work with a partner. [Install the app](https://docs.google.com/document/d/1l5jAubqxUZEKbIOPIuGYk26XnxxblDkCidxwRPAuQCM/edit?usp=sharing) (Don’t run it on the App Inventor Companion) on one of the devices and send messages from the other device. Messages sent to the device running the app should receive the automatic reply. (NOTE: Don’t install the app on both devices, otherwise you could create an endless cycle of sending and receiving.)
* Use your Google Voice account to send yourself a message on the device. Then check that your Google Voice account has received the automatic reply from the device.

**NOTE:** You’ll need to ***authenticate*** before sending messages through the app. A message/request will appear on the device when the app *sends* a message for the first time. If the device has multiple Google accounts on it, select the account that you used to set up the Google Voice account. If the authentication process is successful, you will see a notification that says “finished” at the bottom of the screen. If “finished” doesn’t appear, try closing the app and reopening it and repeat the process until you see “finished”.

## Testing Issues

This is a tricky app to test and debug because there are lots of things that could go wrong to prevent the app from working properly, even if its code is correct. Here are some things to think about:

* Make sure all the devices have proper Wifi connections if you are using Wifi, rather than mobile service.
* Make sure the the texting device has GoogleVoice installed and enabled and the user has a Google Voice account. See the setup instructions given above. **IMPORTANT NOTE: Google Voice has been updated and is no longer working for use with the texting component. In order to test apps that require the texting component, disable Google Voice in App Inventor and use an Android device that has cellular service (e.g. an Android cellphone).**
* **Remember: For each change you make to the app, it will be necessary to** [**re-package it and re-install it on the device.**](https://docs.google.com/document/d/1l5jAubqxUZEKbIOPIuGYk26XnxxblDkCidxwRPAuQCM/edit?usp=sharing)

# Enhancements:

Here are some ideas for programming projects.

* **Customization:** Add a feature that allows the user to input the message that gets sent automatically while busy. For a simple version of this, a Textbox and a Button could be used.
* **Customization:** A more sophisticated version of the above enhancement might be to add a *Listpicker* that lets the user choose the category of ‘busyness’ and then sets the outgoing message to one that is appropriate for that category. For example, the categories might by [driving, studying, working] and the corresponding messages might be [“Driving, TTYL”, “Gotta study now sorry”, “I’m at work now and can’t respond. I’ll get back to you soon..”].
* **Look and Feel:** Improve the overall appearance of the app.
* **Persistence:** Add a TinyDb component to the app so that their custom replies will persist between uses of the app.
* **Settings Screen (Advanced):** Add a second *Settings* screen that allows the user to set certain Texting and/or app properties. For example, for the Texting component, let the user control whether GoogleVoice is enabled and when the app is receiving messages (Off, Always, Foreground). For the app, maybe the custom message setting could be done on this screen?

HINT: You may need to use the [How to: Pass Information Between Screens](https://docs.google.com/document/d/1lnYq4Fuw6DPKohEv1gdqpzNdyugERrw3aI_GPYBh8Y4/edit?usp=sharing) tutorial.

***Nice work! Complete the Self-Check Exercises and Portfolio Reflection Questions as directed by your instructor.***