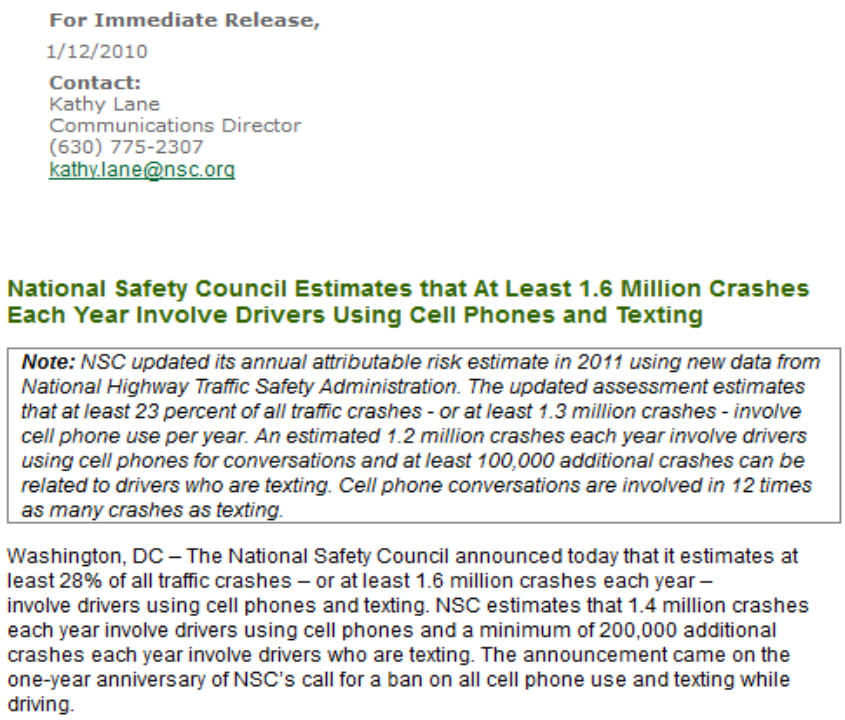
Sms y Localización GPS

En este capítulo veremos la aplicación **No escribas mientras conduzcas**, capaz de responder automáticamente a los mensajes de texto que reciba tu teléfono móvil mientras conduces. El creador de este programa fue un estudiante de informática y lo desarrolló utilizando **AppInventor**. Es parecido a uno creado por *State Farm Insurance* que alcanzó mucha fama.

Demuestra la facilidad que tiene AppInventor para acceder a algunas propiedades de los teléfonos Android, como la generación de mensajes de texto (SMS), la gestión de las bases de datos, la capacidad para leer en voz alta un texto y el sensor de ubicación.

En Estados Unidos, el 28% de los accidentes de tráfico estaban provocados por conductores que hablaban por su teléfono móvil en el año 2010. Más de 200.000 tuvieron lugar mientras el que conducía estaba escribiendo un mensaje de texto. Precisamente, por esto, muchos países han prohibido el uso de móviles cuando se está al volante. Puedes ver el texto completo desde aquí.



For Immediate Release,
1/12/2010

Contact:
Kathy Lane
Communications Director
(630) 775-2307
kathy.lane@nsc.org

National Safety Council Estimates that At Least 1.6 Million Crashes Each Year Involve Drivers Using Cell Phones and Texting

Note: *NSC updated its annual attributable risk estimate in 2011 using new data from National Highway Traffic Safety Administration. The updated assessment estimates that at least 23 percent of all traffic crashes - or at least 1.3 million crashes - involve cell phone use per year. An estimated 1.2 million crashes each year involve drivers using cell phones for conversations and at least 100,000 additional crashes can be related to drivers who are texting. Cell phone conversations are involved in 12 times as many crashes as texting.*

Washington, DC – The National Safety Council announced today that it estimates at least 28% of all traffic crashes – or at least 1.6 million crashes each year – involve drivers using cell phones and texting. NSC estimates that 1.4 million crashes each year involve drivers using cell phones and a minimum of 200,000 additional crashes each year involve drivers who are texting. The announcement came on the one-year anniversary of NSC's call for a ban on all cell phone use and texting while driving.

Daniel Finnegan, un alumno que asistía en el año 2010 a un curso de programación de **AppInventor** tuvo una idea para solucionar el problema de los conductores y los teléfonos móviles. Creó un aplicación capaz de responder automáticamente cualquier mensaje de texto. Envía un mensaje al remitente del tipo *"I'm driving right now, I'll contact you shortly"*. Después de plantear la idea inicial, pensó en una serie de mejoras:

- **Los usuarios pueden modificar las respuestas de la aplicación para adaptarla a diferentes situaciones**: por ejemplo, si estas en el cine viendo una película o cenando con los amigos, podrás personalizar la contestación.

- **La aplicación lee el texto en voz alta**: Aunque sepas que responderá automáticamente al remitente, es muy posible que la curiosidad haga que cojas el móvil.

- Se **puede incluir tu ubicación en la respuesta**: Si el remitente te está esperando para cenar, querrá saber cuánto tiempo tardarás en llegar. Podemos informarle de nuestra ubicación sin necesidad de coger el teléfono. La aplicación se encargará de todo.

Algunas semanas más tarde, después de publicarse la aplicación en en el sitio web de **AppInventor**, *State Farm Insurance* creó una en Android llamada **On the Move**, cuyo funcionamiento es muy parecido a **No escribas mientras conduzcas** . Curioso , ¿ no te parece ?.

¿ Qué aprenderé ?

La aplicación de este capítulo es más compleja que las vistas hasta ahora. Por eso, iremos construyendo sus funcionalidades de una en una. Vamos a empezar con el sistema responsable de generar las respuestas automáticas. En la creación de este bloque veremos los siguientes puntos:

- El componente **Texting** se utiliza para enviar textos y procesar los mensajes recibidos.
- Usaremos un formulario de entrada para enviar mensajes personalizados.
- El componente **TinyDB** se emplea para guardar el mensaje que ha creado el usuario y conservarlo despúes de que se cierre la aplicación.
- El evento **Screen.Initialize** es necesario para abrir la respuesta personalizada cuando se inicia la aplicación.
- El componente **Text-to-Speech** se encarga de leer los mensajes en voz alta.
- El componente **LocationSensor** es el responsable de informar sobre la ubicación del conductor.

Empezamos

Para que esta aplicación funcione deberemos de trabajar con **Text-to-Speech Extended**. Se trata de un módulo capaz de convertir un texto en palabras. Está incluido en la versión 2 (y posteriores) del sistema operativo Android. Si tu teléfono cumple esto no hay nada de lo que preocuparse.

Diseñar los componentes

La interfaz de usuario de la aplicación es muy sencilla: tiene una etiqueta que muestra la respuesta automática, un cuadro de texto y un botón para modificar dicho mensaje. También se necesitará arrastrar hasta el área de trabajo un componente **Texting**, uno **TinyDB**, otro **TextToSpeech** y, por último, uno **LocationSensor**. Todos ellos aparecerán en el área **Non-Visible components** (componentes no visibles).

Para construir la interfaz se deberá trabajar con los siguientes componentes:

Tipo de componente	Grupo de Palette	Cómo lo llamaremos	Finalidad
Label	**Basic**	Informacion	Permite que el usuario sepa cómo funciona la aplicación.
Label	**Basic**	Respuesta	Es la respuesta que se enviará al remitente.
TextBox	**Basic**	Nueva_Respuesta	Aquí escribirá el usuario un mensaje personalizado.
Button	**Basic**	Enviar	El usuario utilizará este botón para enviar la respuesta.
Texting	**Social**	Texting1	Procesa el texto.
TinyDB	**Basic**	TinyDB	Guarda la respuesta en la base de datos.
TextToSpeech	**Other stuff**	TextToSpeech1	Lee el texto en voz alta.
LocationSensor	**Sensors**	LocationSensor1	Identifica la ubicación del teléfono.

Utiliza los siguientes puntos para definir las propiedades de los componentes:

- Escribe el texto *"El texto que se encuentra abajo se enviará como respuesta al SMS recibido cuando la aplicación esté abierta."* en la propiedad **Text** de **Informacion**.

- Escribe el texto *"Estoy conduciendo, en breve me pondré en contacto contigo."* en la propiedad **Text** de **Respuesta**.

- Deja en blanco el valor del campo **Text** de **Nueva_Respuesta**. Así, el campo de texto donde el usuario escribirá su mensaje aparecerá sin contenido.

- Escribe el texto *"Escribe una nueva respuesta."* en la propiedad **Hint** de **Nueva_Respuesta**.

- Escribe el texto *"Modificar respuesta."* en la propiedad **Text** de **Enviar**.

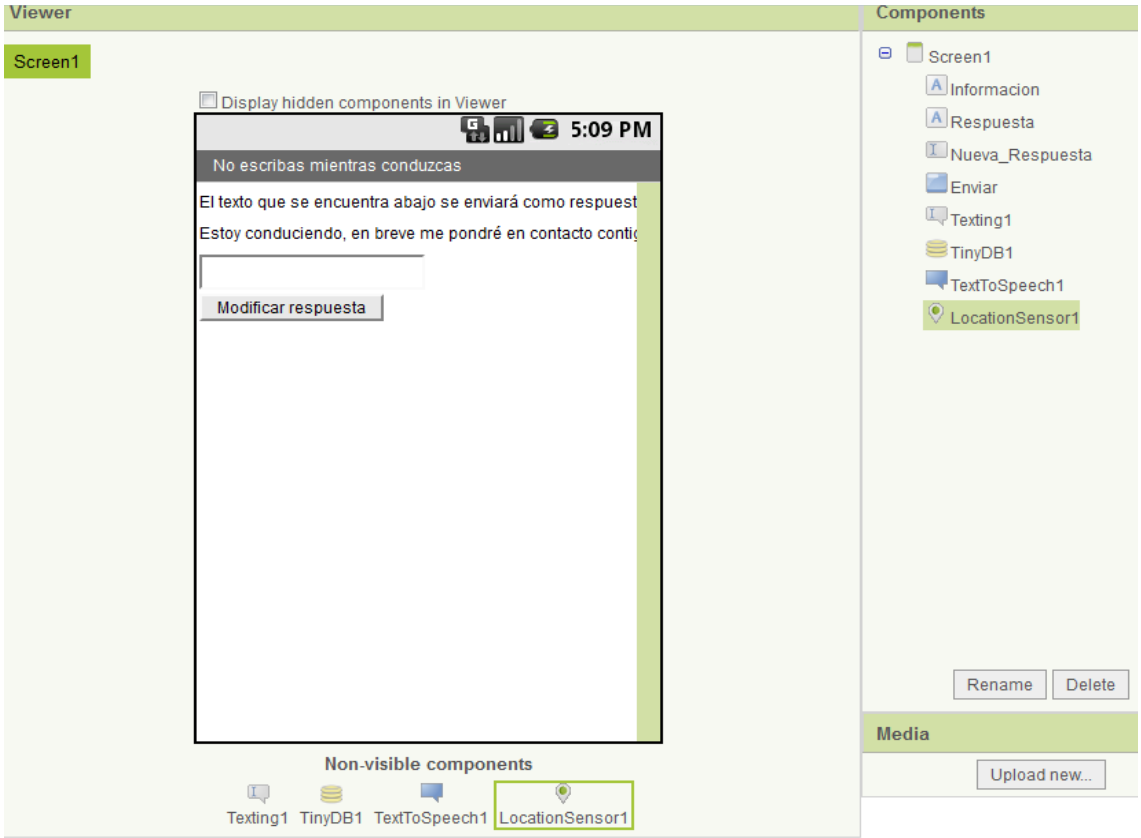
- Arrastra desde **Social** el elemento **Texting.**

- Arrastra desde **Basic** el elemento **TinyDB**

- Arrastra desde **OtherStuff** el elemento **TextToSpeech.**

- Arrastra desde **Sensors** el elemento **LocationSensor.**

  De momento habrás obtenido la siguiente pantalla:



Guarda el proyecto con el nombre **Noescribasmientrasconduzcas** en tu PC, se generará un fichero **"Noescribasmientrasconduzcas.apk"**

Definir el comportamiento de los componentes

Empezaremos programando el comportamiento del sistema de respuesta automática. Después, iremos añadiendo más funcionalidades.

Programar una respuesta automática

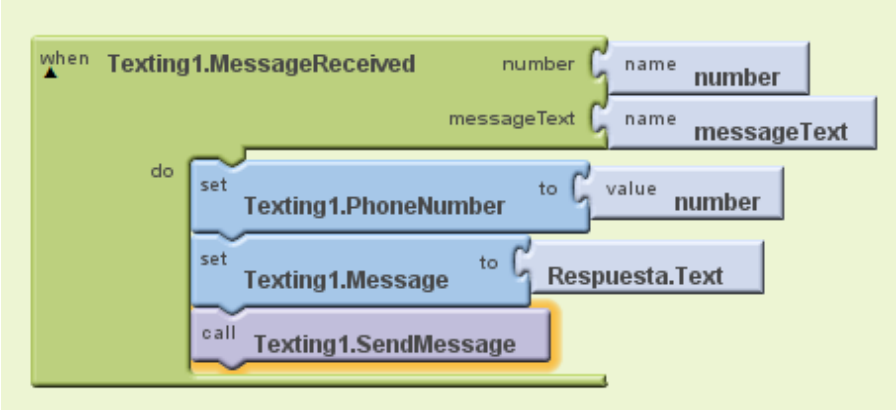
Para programar la respuesta automática recurriremos al componente **Texting** de **AppInventor.** Para leer los mensajes este componente cuenta con un bloque **Texting.MessageReceived**. Abre el **Blocks Editor** y arrastra este bloque hasta el área de trabajo. Para programar el texto de respuesta colocaremos el bloque **Texting1.SendMessage** dentro de **Texting.MessageReceived**.

Texting1.SendMessage se encargará de enviar el texto, así que lo primero será indicarle qué debe enviar y quién será el destinatario. Esto lo haremos antes de llamar a **Texting1.SendMessage**.

La siguiente tabla muestra todos los bloques que necesitamos para definir el comportamiento del contestador automático:

Tipo de bloque	Cajón	Finalidad
Texting1.MessageReceived	**Texting**	Controlador de eventos que se lanzará cuando el teléfono reciba un mensaje de texto.
Set Texting1.PhoneNumber to	**Texting**	Define la propiedad PhoneNumber antes de hacer el envío.

Tipo de bloque	Cajón	Finalidad
value number	**My Definitions**	El número de teléfono del remitente que nos envió el mensaje.
Set Texting1.Message to	**Texting**	Define la propiedad Message antes de hacer el envío.
Respuesta.Text	**Respuesta**	El mensaje de texto que ha escrito el usuario
Texting1.SendMessage	**Texting**	Envía el mensaje.



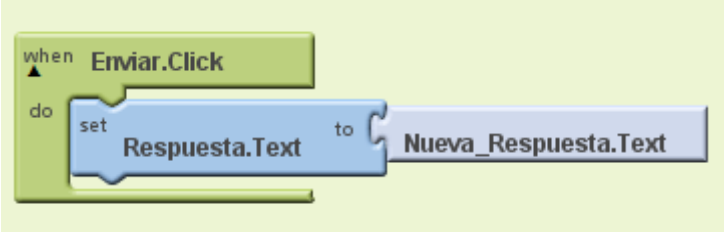
Cuando el teléfono recibe un mensaje de texto, se lanzará el evento **Texting1.MessageReceived**. El *número de teléfono del remitente* es el argumento **number** y el *mensaje recibido* es el **messageText**. Para que la aplicación pueda responder automáticamente, deberá enviar un mensaje de texto al remitente. Para que se pueda mandar, empezaremos por definir dos propiedades de **Texting**: **PhoneNumber** y **Message.Texting**. A la primera se le asigna el **número de teléfono del remitente** y a la segunda el texto que hayamos escrito en **Respuesta** (“ Estoy conduciendo, en breve me pondré en contacto contigo. ”). Cuando se hayan definido, la aplicación llamará a **Texting.SendMessage** para que proceda con el envío de la respuesta.

Guarda primero el proyecto con el nombre **Noescribasmientrasconduzcas2**, se generará el fichero **Noescribasmientrasconduzcas2.apk** que es el que debes copiar a tu teléfono y probarlo .Para probar este comportamiento necesitarás un segundo teléfono. Si no tienes ninguno, puedes registrarte en **Google Voice** (que parece sólo funciona en Estados Unidos) o en un servicio similar. Luego, manda mensajes de texto desde aquí a tu teléfono móvil.

PRUEBA!!!!

Escribir una respuesta personalizada

En el componente **TextBox** que añadimos al principio de la aplicación y que llamamos **Nueva_Respuesta**, escribiremos la respuesta personalizada de nuestra contestación. Cuando toques el botón **Enviar** copiaremos el texto que has escrito (estará en **Nueva_Respuesta**) en **Respuesta**, que contendrá el mensaje que enviará nuestro contestador automáticamente.



Guarda primero el proyecto con el nombre **Noescribasmientrasconduzcas3**, se generará el fichero **"Noescribasmientrasconduzcas3.apk"** que es el que debes copiar a tu teléfono y probarlo si no te funciona correctamente ahora.

PRUEBA!!!!

Guardar la respuesta personalizada en una base de datos

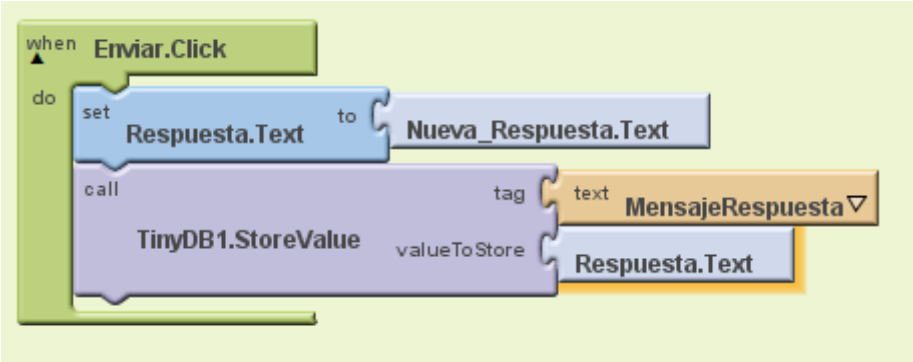
Si el usuario cierra el programa, cuando vuelva a abrirlo la contestación se habrá perdido y se empleará la opción predeterminada, para evitar esto hemos de guardar las respuestas en un elemento con memoria a largo plazo: *la base de datos del teléfono.*

Para almacenar *un dato persistentemente* debemos utilizar el componente **TinyDB**. Será el encargado de guardar la información en la base de datos del dispositivo Android. **TinyDB** tiene dos funciones: **StoreValue** y **GetValue**. La primera permite guardar información en la base de datos del dispositivo y la segunda recuperarla desde la base de datos en una variable o una propiedad.

Empezaremos modificando el controlador de eventos **Enviar.Click** para que guarde la información de forma persistente. Para ello usaremos los bloques de la tabla de abajo.

Tipo de bloque	Cajón	Finalidad
TinyDB1.StoreValue	**TinyDB1**	Almacena el mensaje personalizado en la base de datos del teléfono.
Text (“MensajeRespuesta”)	**Text**	Se utiliza para etiquetar la información.
Respuesta.Text	**Respuesta**	Ahora el mensaje de respuesta está aquí.

Esta aplicación emplea **TinyDB** para recuperar el texto contenido en **Respuesta** y almacenarlo en la base de datos. Cuando almacenamos algo en la base de datos tenemos que asignarle una etiqueta para identificarlo.



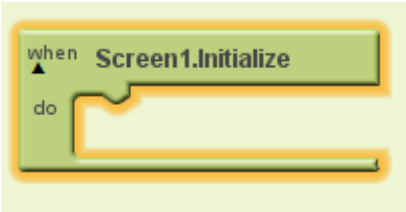
En este caso es **MensajeRespuesta**. Es un identificador único que se asigna a la información almacenada. Como veremos en la próxima sección, siempre utilizaremos la misma etiqueta (**MensajeRespuesta**) para abrir los datos en la base.

Recuperar la respuesta personalizada cada vez que se abra la aplicación

Como ya vimos en el capítulo 3, cuando se abra la aplicación **AppInventor** ejecuta todos los bloques que se encuentren dentro de **Screen1.Initialize**, por lo que deberemos comprobar en ese momento si tenemos alguna respuesta personalizada en la base de datos. En caso afirmativo, la abrirá mediante la función **TinyDB.GetValue**. La siguiente tabla muestra todos los bloques que se necesitarán para recuperar la respuesta personalizada de la base de datos e incorporarla cuando se abra la aplicación.

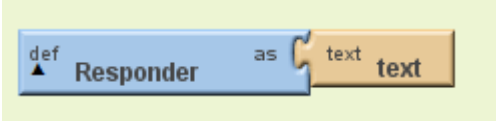
Tipo de bloque	Cajón	Finalidad
def variable (“responder”)	**Definition**	Una variable temporal donde se guardará la información.
Text (“”)	**Text**	Inicialmente la variable puede tener cualquier valor.
Screen1.Initialize	**Screen1**	Este evento se lanza al abrir la aplicación.
Set global response to	**My Definitions**	A esta variable le asignaremos el valor recuperado d ella base de datos.
TinyDB1.GetValue	**TinyDB1**	Recupera el texto de la respuesta que está almacenado en la base de datos.
Text (“MensajeRespuesta”)	**Text**	Asigna este valor a la etiqueta de TinyDB.GetValue, asegurándose de que el texto es el mismo que se utilizó anteriormente en TinyDB.StoreValue.
If	**Control**	Pregunta si el valor que se ha recuperado contiene algún texto.
>	**Math**	Comprueba si la longitud del valor recuperado es mayor que 0.
Length text	**Text**	Comprueba si la longitud del valor recuperado es mayor que 0.
global response	**My Definitions**	Esta variable contendrá el valor recuperado de TinyDB1.GetValue.
Number (0)	**Math**	Compara este valor con la longitud de la respuesta.
Set **Respuesta**.Text to	**Respuesta**	Si hemos recuperado algo se colocará en **Respuesta**.
global Responder	**My Definitions**	Esta variable contiene el valor recuperado de TinyDB1.GetValue.

La primera vez que el usuario abre la aplicación no habrá ninguna respuesta personalizada guardada en la base de datos, por lo que deberemos usar la predeterminada almacenada en **Respuesta**. Como va a ser modificada, las próximas veces que abra la aplicación deberemos recuperarla de la base de datos y colocarla en **Respuesta**. Cuando se abra la aplicación, se lanzará el evento **Screen1.Initialize** (arrastra el bloque **Initialize** desde **Screen1** que se encuentra en **My Blocks**).



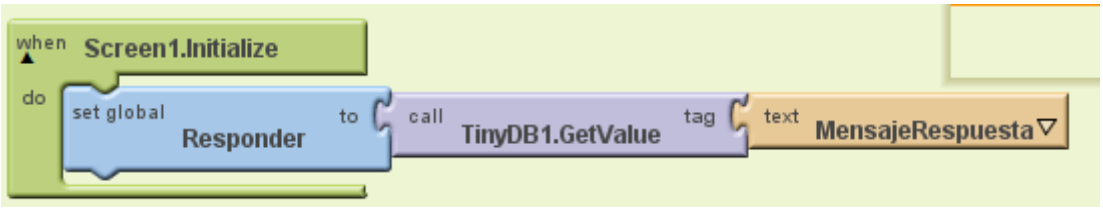
Se llamará a **TinyDB1.GetValue** mediante la etiqueta de **MensajeRespuesta** es decir, la misma que se empleó cuando se guardó la respuesta personalizada del usuario en la base de datos. El valor que se recupere de aquí se colocará en la variable **Responder**. Así, la aplicación podrá verificarla antes de colocarla en **Respuesta**.

Lo primero será definir la variable **Responder** que vamos a utilizar, para ello arrastra desde **Built-in** en la sección **Definition** el bloque **def variable as**, cambia el nombre ***def*** *por* ***Responder.***



Arrastra el bloque **set global Responder to** dentro de Screen1.**Initialize**. Concatena en esa ranura del set global a la llamada **call TinyDB1.GetValue tag** y vuelve a concatenar el texto **text MensajeRespuesta** de la siguiente manera: Ves a **Built-In**, sección **text** y arrastra el bloque **text text**, cambia el nombre **text** por **MensajeRespuesta**.

Añade después de la ranura **as** otro bloque de **text** llamado **text**. Ahora en **My Definitions** de **My blocks** ya nos aparece la nueva variable **Respuesta** que nos acabamos de crear.

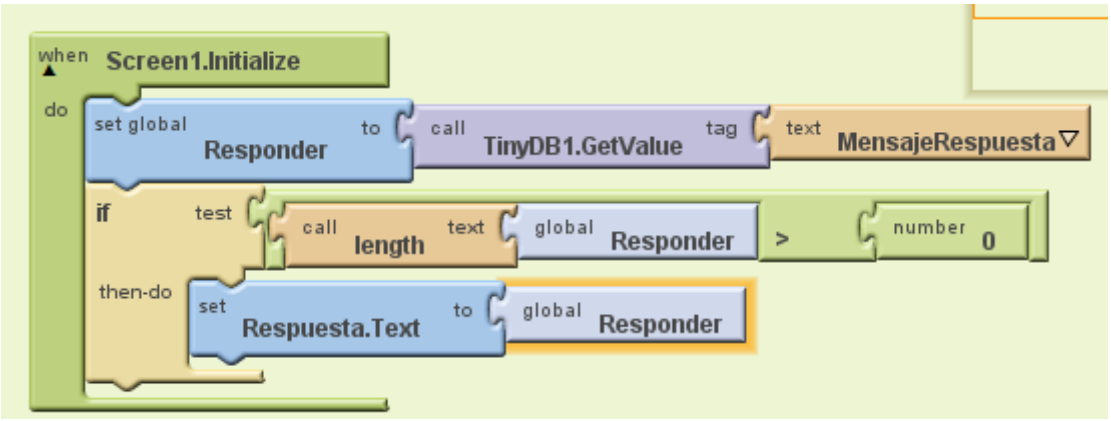


Ahora hemos de comprobar si el valor recuperado de **TinyDB** es un texto vacío o no, es decir, hay algún valor o no porque es la primera vez que abrimos la aplicación y no hay ninguna respuesta personalizada guardada en la base de datos. Para ello preguntaremos si la longitud del texto recuperado es mayor que 0 y colocaremos el valor recuperado dentro de **Respuesta**, sino no modificaremos el valor de **Respuesta** y se trabajará con la respuesta predeterminada. Para ello arrastramos el bloque **If test then-do** de la sección **control** de **Built-In**, dentro de **Initialize** justo debajo de set global response que teníamos.

Arrastramos el **bloque mayor** (“>”) de la sección **Math** de **Built-In** en la ranura **test**; Añadimos el bloque **call length text** de la sección **text** de **built-In** en la parte **izquierda** de la comparación del **“>”**. Y concatenamos en su ranura la variable **global response** de **My Definitions**. En la parte derecha del mayor, creamos un “**0**” como hemos dicho anteriormente para saber si la longitud de lo recuperado en la base de datos es mayor que 0, es decir, hay algo en la base de datos.

Como hemos dicho si esto sucede colocamos le valor recuperado de la base de datos que ahora tenemos en la variable response en **Respuesta**, para ello arrastro a la altura del **then-do del if,** el bloque **set Respuesta.Text to** y le concateno **global response** de My Definitions.

Gráficamente nos queda lo siguiente:



Guarda primero el proyecto con el nombre **Noescribasmientrasconduzcas4**, se generará el fichero **"Noescribasmientrasconduzcas4.apk"** que es el que debes copiar a tu teléfono.
Esta prueba no la vamos a realizar, nos creeremos que funciona....Sigamos....

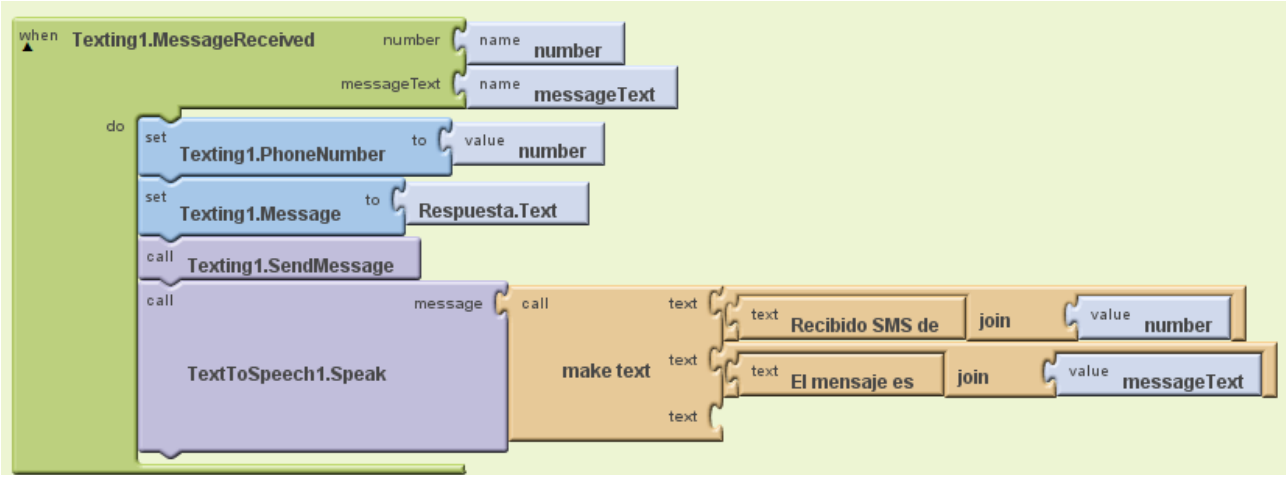
PRUEBA!!!!

Leer en voz alta los mensajes recibidos

En esta sección modificaremos la aplicación para que, cuando recibamos un mensaje, el teléfono lea en voz alta el número del remitente y el contenido. La idea es que, cuando estemos al volante y oigamos que se ha recibido un mensaje, aunque sepamos que hay una aplicación en marcha que se encargará de contestar automáticamente al remitente, evitemos la tentación de coger el teléfono para ver quién nos escribe y qué nos cuenta. Gracias a **text-to-speech**, escucharemos el mensaje recibido sin soltar el volante.

Crearemos una llamada a **TextToSpeech.Speak** (leerá en voz alta el mensaje recibido) desde el controlador de eventos **Texting.MessageReceived** que programamos con anterioridad. Es decir, arrastra el bloque **call TextToSpeech. Speak message** dentro y al final del bloque **MessageReceived** ya creado.

El bloque **call make text** (del cajón **text**) construirá las palabras que se leerán en voz alta y lo concatenamos a la salida de **Speak**. Para crearlo se unen las siguientes cadenas de texto: "**Recibido SMS de** ", el número de teléfono desde donde se envió (**value number**), el **texto** (". El mensaje es, ") y, por último, el contenido del aviso recibido (**value messageText**). Arrastramos el bloque **call make text** a la ranura del Speak. Despúes arrastramos un bloque **text text** a la salida de la ranura del **call make text** que acabamos de añadir. Cambiamos el campo **text** del bloque **text** para personalizarlo a "**Recibido SMS de**". Seguimos construyendo el **make text** y añadimos ahora el número de teléfono con **value number** desde **My Definitions**. Seguimos con el **texto ". El mensaje es,"** y terminamos con el contenido del aviso recibido **value messageText**.



Guarda primero el proyecto con el nombre **Noescribasmientrasconduzcas5** se generará el fichero **"Noescribasmientrasconduzcas5.apk"** que es el que debes copiar a tu teléfono y probarlo si no te funciona correctamente ahora a través de otro teléfono.

PRUEBA!!!!

Agregar a la respuesta información sobre la ubicación del teléfono

En la aplicación podemos utilizar los datos sobre la ubicación del teléfono para proporcionar algo más de información en la respuesta que enviamos automáticamente desde nuestro terminal. **AppIinventor** cuenta con el componente **LocationSensor**, capaz de interactuar con el sistema GPS del teléfono. Además de recuperar información sobre la latitud y la longitud de la posición donde está el terminal, también puede consultar el contenido de *Google Maps* para indicar el nombre de la calle.

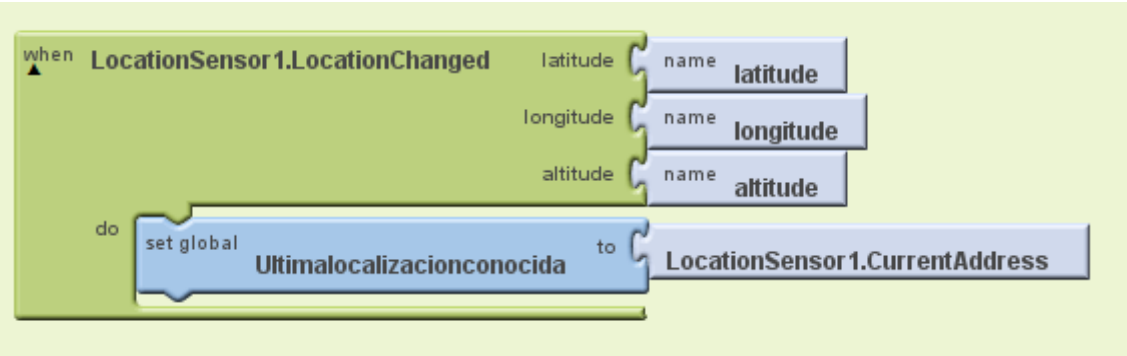
La aplicación debería de responder al controlador de eventos **LocationSensor.LocationChanged**. Antes incluso de que el teléfono sea capaz de leer el valor de la posición del sensor, ya habremos cambiado de ubicación (*LocationChanged*) y el teléfono tendrá que intentar establecer la nueva posición, guardando la dirección actual dentro de una variable llamada **Ultimalocalizacionconocida**. Más tarde, modificaremos el mensaje de respuesta para incluir la dirección que obtendremos de esta variable.

Los bloques necesarios para configurar el sensor de ubicación son:

Tipo de bloque	Cajón	Finalidad
def variable (“Ultimalocalizacionconocida”)	**Definition**	Crea una variable donde guardaremos la dirección obtenida en la última lectura del sensor de ubicación.
Text (“Desconocido”)	**Text**	Establece un valor predeterminado para el caso de que no esté funcionando el sensor de ubicación del teléfono.
LocationSensor1.LocationChanged	**LocationSensor1**	Este elemento se lanzará con la primera lectura de la posición del teléfono y cada vez que se cambie la ubicación.
Set global Ultimalocalizacionconocida to	**My Definitions**	Prepara esta variable para utilizarla más tarde.
LocationSensor1.CurrentAddress	**LocationSensor1**	Es la dirección postal en la que nos encontramos.

La variable será:

y el bloque asociado cuando cambie la posición del GPS es:



Aún tenemos que hacer que la aplicación incorpore la información sobre nuestra posición en el mensaje que enviará automáticamente al remitente.

Incluir nuestra posición en el mensaje de respuesta

Gracias a "**Ultimalocalizacionconocida**", podemos modificar el controlador de eventos **Texting1.MessageReceived** para añadir información sobre nuestra posición al mensaje de respuesta.

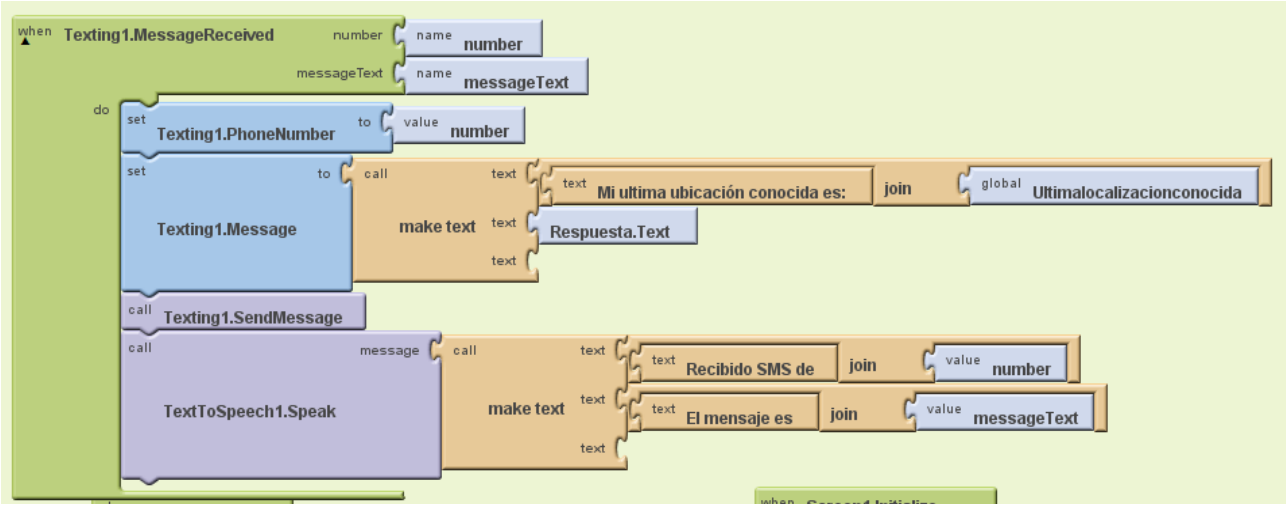
Utilizaremos los siguientes bloques:

Tipo de bloque	Cajón	Finalidad
make text	**Text**	Si hay datos sobre la posición del terminal, se construirá un objeto de texto.
Respuesta.Text	**MesasageTextBox**	Éste es el mensaje (personalizado) del cuadro de texto.
Text (" Mi última ubicación conocida es:")	**Text**	Este texto se adjuntará al final del mensaje personalizado. Empieza con un espacio en blanco.
global Ultimalocalizacionconocida	**LocationSensor**	Ésta es la dirección postal.

Este comportamiento está relacionado con el evento **LocationSensor1.LocationChanged** y con la variable **Ultimalocalizacionconocida**. En vez de enviar un mensaje que contenga el texto almacenado en **Respuesta.Text**, la aplicación empezará por construir, en su lugar, un texto mediante **make text**. Combina el de la respuesta guardado en **Respuesta.Text** con "**Mi última ubicación conocida es:** ", seguida de la variable **Ultimalocalizacionconocida**.

El valor predeterminado de **Ultimalocalizacionconocida** es **Desconocido**. Así pues, si el sensor de la posición aún no ha generado ninguna lectura, la segunda parte del mensaje mostrará el texto "**Mi última ubicación conocida es: desconocido**". Pero, si ha habido una lectura, mostrará la dirección postal.

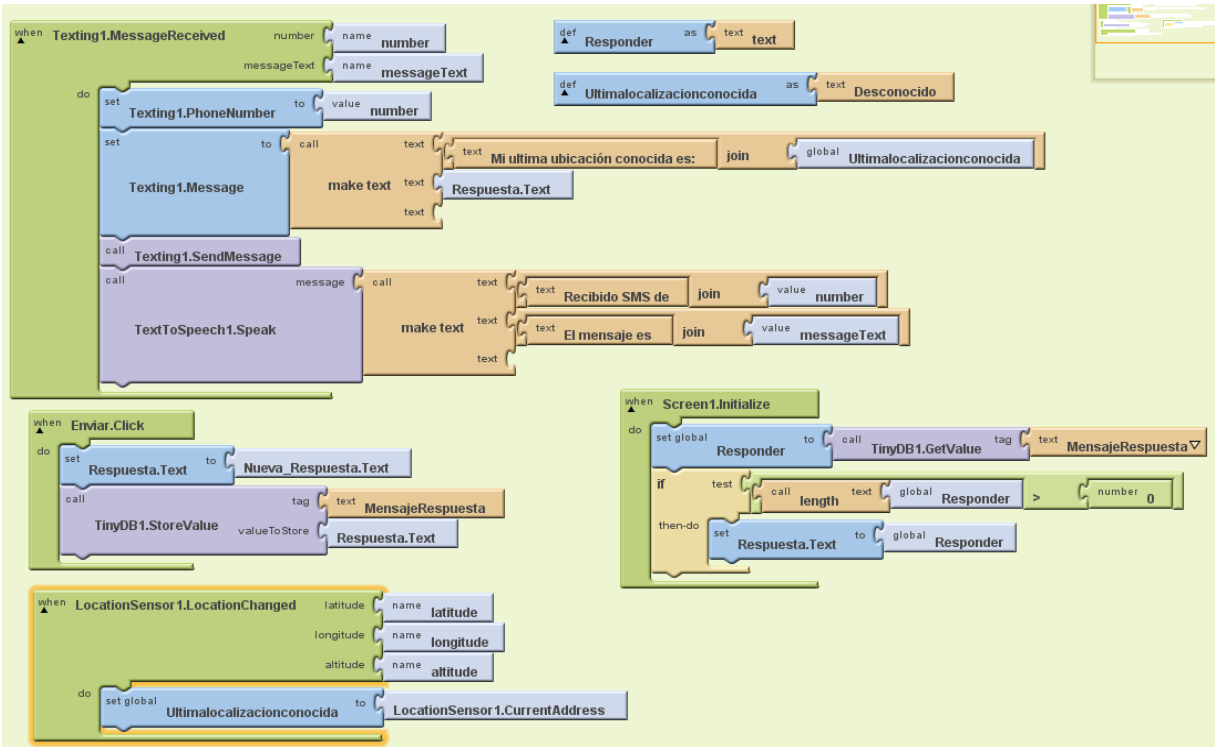
Dentro del **Texting1.Received** cambia el **Respuesta.Text** por un **make text** como en la figura siguiente:



Guarda primero el proyecto con el nombre **Noescribasmientrasconduzcas6,** se generará el fichero **"Noescribasmientrasconduzcas6.apk"**que es el que debes copiar a tu teléfono y probarlo.

PRUEBA!!!!

La aplicación finalizada:



Variaciones

- Escribe una versión que permita que el usuario defina una serie de mensajes personalizados para ciertos números de teléfono. Deberás utilizar los bloques condicionales (if) para que comprueben los números del remitente.

- Escribe una versión que envíe respuestas personalizadas basándose en la posición del remitente. Así pues, si la aplicación determina que te encuentras en la habitación 222 de un hotel y su remitente se halla en otro cuarto del mismo hotel, puedes enviar un mensaje que diga “En este momento, Javier se encuentra en la habitación 222.”.

- Escribe una versión que emita una alarma cuando el número del remitente se ubique dentro de una lista de notificación.