MIT APP INVENTOR 3

di FRANCESCO FICILI

Proseguiamo lo studio e la conoscenza di App Inventor, il tool di sviluppo per applicazioni Android creato dal MIT e basato su un linguaggio di programmazione grafico.
In questa terza puntata analizzeremo due interessanti famiglie di componenti, ossia Storage e Sensors.

Nella scorsa puntata di questo corso ci siamo lasciati dopo aver introdotto il concetto di componente in MIT App Inventor (i componenti sono gli elementi che permettono di aggiungere funzionalità all'app in sviluppo...) ed aver constatato e sperimentato quanto potente sia questo strumento, che ci permette sostanzialmente di creare l'interfaccia grafica della nostra app, di accedere alle varie funzionalità specifiche del nostro smartphone e di utilizzare i vari sensori che si trovano al suo interno e compongono la dotazione hardware.
Abbiamo anche avuto modo di conoscere le interfacce di comunicazione presenti all'interno degli smartphone e, a contorno della definizione del

concetto di componenti, con un esempio pratico siamo riusciti a sperimentare l'utilizzo del componente notifier, che permette di notificare all'utente un messaggio al verificarsi di un certo evento.

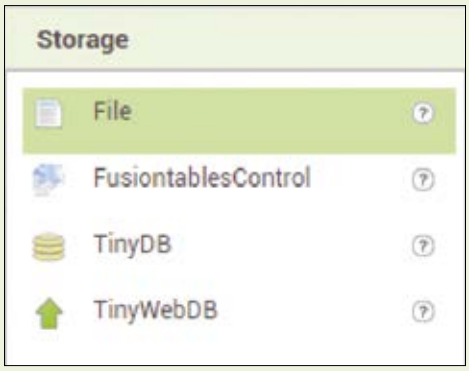

Fig. 1 - Famiglia di componenti storage.

In questa puntata riprendiamo da dove ci siamo fermati e analizziamo in dettaglio due nuove famiglie di componenti di App Inventor: la famiglia di componenti **Storage** e la famiglia di componenti **Sensors**.
Anche in questa puntata, come nelle precedenti, per brevità utilizzeremo spesso l'acronimo AI per dire App Inventor.

I componenti Storage

La famiglia di componenti Storage contiene componenti per la gestione di file e dei data-base; in essa è inoltre presente un componente per la gestione delle Google Fusion Tables, che tuttavia non tratteremo in questo corso. La **Fig. 1** riporta una schermata che raggruppa la famiglia di componenti Storage nell'IDE di App Inventor.
Ciò detto, passiamo adesso a un'analisi più di dettaglio dei singoli componenti ed in particolare di proprietà, eventi e metodi che li caratterizzano.
Ricordiamo anche brevemente la classificazione dei colori per i behaviours dei componenti:

- Proprietà: verde;
- Eventi: beige;
- Metodi: viola.

File Storage

Si tratta di un componente non visibile (non-visible component) che permette di effettuare operazio-

Fig. 2 - Layout della app di esempio per l'utilizzo del componente File.

PROPRIETÀ	
SINTASSI	DESCRIZIONE
Nessuna	-
EVENTI	
SINTASSI	DESCRIZIONE
GotText(text text)	Evento che indica che il contenuto di un dato file è stato letto e fornisce il risultato della lettura tramite il parametro 'text'.
METODI	
SINTASSI	DESCRIZIONE
AppendToFile(text text, text fileName)	Appende il parametro 'text' al file puntato dal parametro 'fileName'.
Delete(text fileName)	Cancella il file puntato dal parametro 'fileName'.
ReadFrom(text fileName)	Legge il contenuto del file puntato dal parametro 'fileName'.
SaveFile(text text, text fileName)	Salva il contenuto del parametro 'text' sul file puntato dal parametro 'fileName'.

Tabella 1 - Proprietà, Eventi e Metodi del componente File.

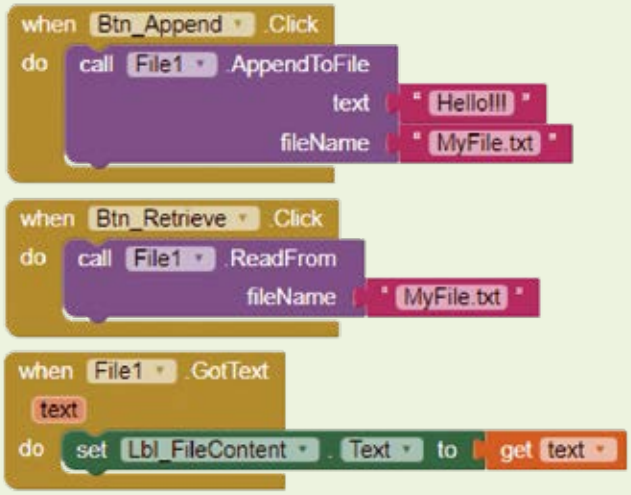

Fig. 3 - Codice grafico per l'app di esempio d'uso del componente File.

ni di salvataggio e lettura su un generico file. Per impostazione predefinita, il componente lavora sulla memoria privata dell'app in esecuzione, ma se il path di salvataggio inizia con il simbolo slash ("/"), allora il file viene creato rispetto alla directory relativa */sdcard*. Proprietà, eventi e metodi del componente sono riportati sinteticamente in **Tabella 1.**
Vediamo un semplice esempio pratico che ci faccia comprendere come funziona il componente File: creiamo un nuovo progetto in AI e aggiungiamo i seguenti componenti:

- 2 componenti Button (chiamati Btn_Append e Btn_Retrieve);
- 1 componente Label (chiamato Lbl_FileContent);
- 1 componente File.

A questo punto il layout della nostra nuova app dovrebbe somigliare a quello riportato nella **Fig. 2**.

Ciò che ci proponiamo di fare è semplicemente di salvare la stringa "Hello!!!" su file alla pressione del tasto "Append" e di recuperarla e stamparla sulla label alla pressione del tasto "Retrieve".
Passiamo ora alla schermata "Blocks" ed inseriamo il codice grafico riportato in **Fig. 3**. Come potete vedere sempre dalla **Fig. 3**, abbiamo utilizzato i due metodi "AppendToFile" e "ReadFrom", collegati agli eventi "Button.Click" dei pulsanti Btn_Append e Btn_Retrieve, rispettivamente per salvare e leggere dal file "MyFile.txt".
Il metodo di lettura è asincrono, nel senso che non restituisce immediatamente il testo letto, ma bisogna utilizzare l'evento asincrono "GotText" per

Fig. 4 - Esecuzione dell'app di esempio tramite l'emulatore.

PROPRIETÀ	
SINTASSI	DESCRIZIONE
Nessuna	-
EVENTI	
SINTASSI	DESCRIZIONE
Nessuno	-
METODI	
SINTASSI	DESCRIZIONE
ClearAll()	Cancella tutti i dati contenuti nel DB.
ClearTag(text tag)	Cancella la entry associata al tag passato come parametro.
any GetTags()	Ritorna la lista di tutti i tags contenuti nel DB.
any GetValue(text tag, any valueIfTagNotThere)	Ritorna il valore immagazzinato in corrispondenza del tag passato come parametro. Se il tag non è presente ritorna il valore associato al parametro "ValueIfTagIsNotThere".
StoreValue(text tag, any valueToStore)	Salva permanentemente il valore passato tramite il parametro "ValueToStore" associato tag passato tramite il relativo parametro.

Tabella 2 - Proprietà, Eventi e Metodi del componente TinyDB.

accedere ai dati recuperati dal file stesso.
Possiamo testare questa app usando un comune smartphone oppure l'emulatore: una volta avviata, premendo una volta sul tasto "Append" e la successiva sul tasto "Retrieve" si otterrà il risultato illustrato in **Fig. 4**.
Come avrete potuto notare da questo esempio applicativo, il componente File è molto utile per salvare informazioni in maniera non volatile ed è estremamente semplice da utilizzare, tuttavia può risultare poco efficiente quando con esso si debbano gestire grosse quantità di dati; in tali situazioni sono più efficienti i componenti database.

TinyDB e TinyWebDB

MIT App Inventor mette a disposizione due tipi differenti di gestione dei DB: TinyDB e TinyWebDB. Il primo è un semplice servizio di storage locale che permette di salvare dati associandoli a dei tag, in maniera da agevolare la successiva fase di recupero. Ogni app ha una sezione di memoria riservata per lo storage locale alla quale TinyDB si appoggia, tuttavia non è possibile utilizzarla per scambiare dati tra diverse app, ma è possibile sfruttarla per scambiare dati tra diversi screen in una app multi screen.
Analizziamo, come di consueto, eventi, metodi e proprietà di questo componente, aiutandoci con la **Tabella 2**.
Come detto in precendenza, AI implementa anche un componente che permette di gestire Database Remoti, chiamato TinyWebDB: si tratta di un set di servizi di comunicazione con un WebService che gestisce un database, molto vantaggioso nel caso in cui ci sia la necessità di scambiare dati tra app diverse o tra dispositivi diversi. Chiaramente occorre

Fig. 5 - Layout dell'app relativo all'esempio di utilizzo del componente TinyDB.

anche impostare la parte lato server. Non ci dilungheremo nella trattazione di questo componente, limitandoci, in questo corso, ad evidenziarne le potenzialità, analizzandone eventi, metodi e proprietà; questi sono meglio descritti nella **Tabella 3**.
Passiamo ora ad un semplicissimo esempio di uti-

PROPRIETÀ	
SINTASSI	DESCRIZIONE
ServiceURL	Questa proprietà serve ad impostare la URL del webservice con il quale il componente comunica per la gestione del DB
EVENTI	
SINTASSI	DESCRIZIONE
GotValue(text tagFromWebDB, any valueFromWebDB)	Questo evento informa che il valore richiesto al WebService è stato recuperato e viene passato insieme al relativo tag.
ValueStored()	Questo evento informa che il WebService ha completato una precedente richiesta di salvataggio.
WebServiceError(text message)	Questo evento indica che il WebService ha segnalato un errore.
METODI	
SINTASSI	DESCRIZIONE
GetValue(text tag)	Questo metodo serve ad inviare una richiesta di lettura del dato marcato dal tag passato come parametro al DB WebService.
StoreValue(text tag, any valueToStore)	Questo metodo serve ad inviare una richiesta di scrittura del dato marcato dal tag passato come parametro al DB WebService.

Tabella 3 - Proprietà, eventi e metodi del componente TinyWebDB.

Fig. 6 - Codice grafico relativo all'esempio di utilizzo del componente TinyDB.

lizzo del componente TinyDB. Creiamo dunque una nuova app e aggiungiamo i seguenti componenti:

- 2 componenti Button (chiamati Btn_Store e Btn_Retrieve);
- 3 componenti Label (chiamati Lbl_Label1, Lbl_Label2, Lbl_ReqTag);
- 1 componente TextBox (chiamato Txt_ValueToStore);
- 1 componente TinyDB.

L'esempio è del tutto simile al precedente, ma stavolta salveremo il contenuto della textbox marcandolo con un tag sul data-base e lo recupereremo utilizzando il medesimo tag, per poi stamparlo sulla label.
Una volta completate le operazioni descritte in precedenza, il vostro layout dovrebbe essere del tutto simile a quello riportato nella **Fig. 5**.
A questo punto diamo uno sguardo al codice grafico di App Inventor, riportato nella **Fig. 6**. Come potete vedere, tale codice è molto semplice: alla pressione del pulsante Btn_Store viene salvato il valore del campo text del textbox, associandolo al tag "MyTag", mentre alla pressione del pulsante Btn_Retrieve viene interrogato il data-base alla ricerca del dato contrassegnato dal tag "MyTag" e il risultato viene scritto sulla label Lbl_reqTag.
Il risultato ottenuto con l'emulatore è riportato nella finestra di dialogo illustrata nella **Fig. 7.**

Fig. 7 - Risultato ottenuto tramite l'emulatore.

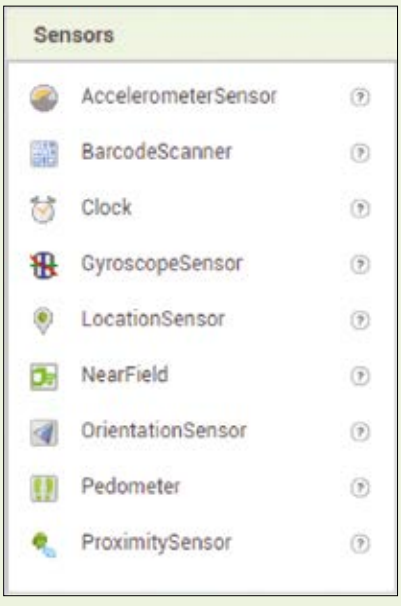

Fig. 8 - Famiglia di componenti Sensors.

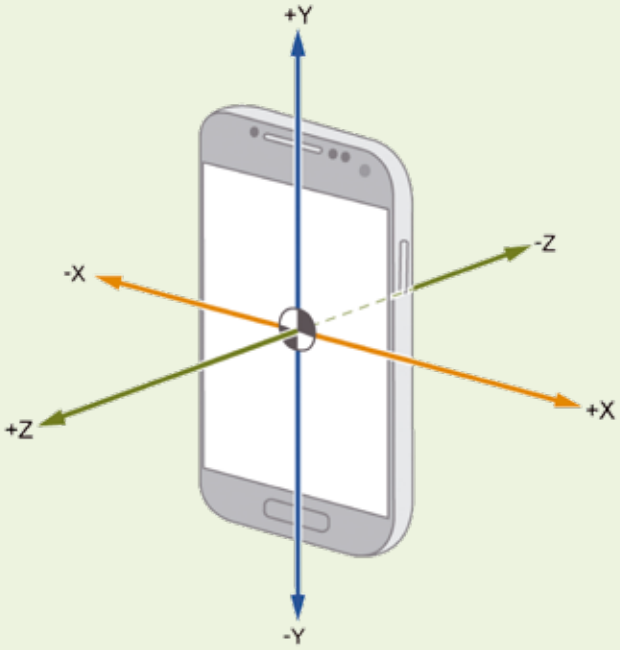

Fig. 9 - Tipica configurazione degli assi di un accelerometro, su uno smartphone Android.

I componenti Sensors

Passiamo ora all'analisi di un'altra famiglia di componenti: la famiglia di componenti Sensors. All'interno di questo gruppo si trovano componenti per la gestione dei vari sensori che tipicamente si trovano all'interno di un comune smartphone, sia fisici (come giroscopi, sensori di prossimità, GPS, ecc.), che virtuali (come ad esempio il pedometro, che in genere non rappresenta un dispositivo fisico reale, ma piuttosto una specifica elaborazione dei dati provenienti da un altro sensore, come ad esempio un accelerometro).
In **Fig. 8** è riportato uno screenshot che rappresenta i componenti presenti in questa specifica famiglia. In questa trattazione noi ci limiteremo all'analisi di alcuni componenti di questa famiglia (i componenti Accelerometer, Orientation e Pedometer), lasciando ai lettori più volenterosi l'approfondimento sugli altri.

Accelerometro

Questo componente è in grado di interfacciare l'accelerometro interno, rilevando accelerazioni in unità del SI (m/s2) oppure condizioni di "shaking". Le accelerazioni sono rilevate sui tre assi: XAccel, YAccel, ZAccel. La **Fig. 9** riporta una tipica configurazione degli assi di misura di un accelerometro in

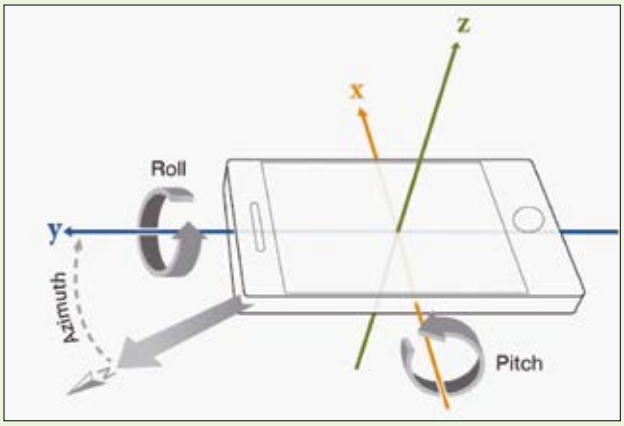

Fig. 10 - Angoli di roll, pitch e azimut in un classico dispositivo android.

PROPRIETÀ	
SINTASSI	DESCRIZIONE
Available	Proprietà che indica se il sensore è disponibile sul dispositivo.
Enabled	Proprietà che indica se il sensore è abilitato.
MinimumInterval	Intervallo minimo di rilevamento (in ms) tra eventi di shaking.
Sensitivity	Proprietà che specifica quanto è sensibile l'accelerometro. La codifica è su tre valori: 1 = bassa, 2 = moderata, 3 = alta.
XAccel	Valore dell'accelerazione sull'asse X.
YAccel	Valore dell'accelerazione sull'asse Y.
ZAccel	Valore dell'accelerazione sull'asse Z.
EVENTI	
SINTASSI	DESCRIZIONE
AccelerationChanged(number xAccel, number yAccel, number zAccel)	Evento che indica che l'accelerazione è cambiata su uno dei tre assi e contestualmente passa i valori delle accelerazioni.
Shaking()	Evento che indica che si è verificato un evento di shaking.
METODI	
SINTASSI	DESCRIZIONE
Nessuno	-

Tabella 4 - Proprietà, eventi e metodi del componente Accelerometer.

PROPRIETÀ	
SINTASSI	DESCRIZIONE
Available	Proprietà che indica se il sensore è disponibile sul dispositivo.
Enabled	Proprietà che indica se il sensore è abilitato.
Azimuth	Ritorna l'angolo di azimut del dispositivo.
Pitch	Ritorna l'angolo di pitch del dispositivo.
Roll	Ritorna l'angolo di roll del dispositivo.
Magnitude	Ritorna un numero da 0 a 1 che indica quanto il dispositivo è inclinato.
Angle	Ritorna un angolo che indica la direzione verso la quale il dispositivo è inclinato.
EVENTI	
SINTASSI	DESCRIZIONE
OrientationChanged (number azimuth, number pitch, number roll)	Evento che indica che è cambiato l'orientamento del dispositivo, e fornisce tra i parametri i tre angolo di roll, pitch ed azimut.
METODI	
SINTASSI	DESCRIZIONE
Nessuno	-

Tabella 5 - Proprietà, eventi e metodi del componente Orientation Sensor.

un dispositivo Android.
In **Tabella 4** sono riportati, come di consueto, eventi, metodi e proprietà del componente.

Orientation Sensor

Il sensore di orientamento è un componente non visibile che permette di determinare l'orientamento spaziale del dispositivo. Questo componente fornisce i seguenti tre valori, in gradi:

- Rollio (Roll): valore dell'angolo di rollio espresso in gradi;
- Beccheggio (Pitch): valore dell'angolo di beccheggio in gradi;
- Azimut: valore dell'angolo di azimut in gradi; questo angolo vale 0° in direzione nord, 180° in direzione sud, 90° in posizione est e 270 gradi in posizione ovest; le informazioni su questo angolo possono essere utilizzare per implementare bussole digitali.

In **Fig. 10** è riportata una rappresentazione grafica che chiarisce ulteriormente come sono misurati gli

PROPRIETÀ	
SINTASSI	DESCRIZIONE
Distance	Questa proprietà ritorna la distanza approssimativa percorsa in metri.
ElapsedTime	Ritorna il tempo trascorso in millisecondi da quando il pedometro è stato attivato.
SimpleSteps	Ritorna il numero di "Simple Step" eseguiti da quando il pedometro è stato attivato.
StopDetectionTimeout	Questa proprietà serve a settare la durata in millisecondi del timeout di idle, ossia il tempo trascorso il quale senza che vengano rilevati dei passi il componente passa in "stopped" state.
StrideLength	Setta la lunghezza del passo medio in metri.
WalkSteps	Ritorna il numero di "Walk Step" eseguiti da quando il pedometro è stato attivato. Si definisce "Walk Step" il passo tipico eseguito per muoversi in avanti.
EVENTI	
SINTASSI	DESCRIZIONE
SimpleStep(number simpleSteps, number distance)	Evento che indica l'esecuzione di un "Simple Step".
WalkStep(number walkSteps, number distance)	Evento che indica l'esecuzione di un "Walk Step".
METODI	
SINTASSI	DESCRIZIONE
Pause()	Mette il pedometro in pausa.
Reset()	Resetta tutti i contatori.
Resume()	Riavvia il pedometro.
Save()	Salva tutti i contatori del pedometro in memoria non volatile.
Start()	Avvia il pedometro.
Stop()	Arresta il pedometro.

Tabella 6 - Proprietà, eventi e metodi del componente pedometer.

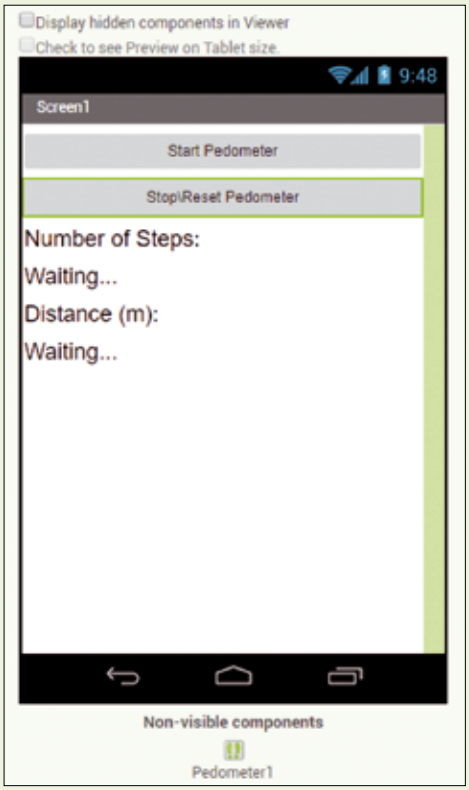

Fig. 11 - Layout dell'applicazione di esempio

angoli di roll, pitch ed azimut. Anche in questo caso esaminiamo eventi, metodi e proprietà, aiutandoci con la **Tabella 5**.

Pedometer

L'ultimo componente della famiglia sensor che analizziamo in questa puntata del corso è il Pedometer (o pedometro): si tratta di un sensore virtuale, che sfrutta le informazioni provenienti dall'accelerometro interno per determinare se la persona che porta con sè lo smartphone ha compiuto dei passi; ciò viene ottenuto analizzando l'accelerazione in una combinazione di direzioni che fa presupporre che si stia camminando, perché corrisponde al movimento in avanti e dal basso in alto e viceversa, il tutto contemporaneamente.
Inoltre, sempre con i componente pedometer, configurando opportunamente un parametro (strideLength) è possibile anche stimare la distanza percorsa.
Esaminiamo anche in questo caso eventi, metodi e proprietà, aiutandoci con la **Tabella 6**, che li raccoglie e li descrive.

Esempio pratico sull'uso dei componenti Sensors

Passiamo a questo punto al nostro consueto esempio pratico, utile a sperimentare i concetti che abbiamo esposto. Per l'esempio abbiamo deciso di utilizzare l'ultimo componente descritto (ossia il Pedometro) per realizzare una semplice applicazione contapassi, con in più una misura approssimativa della distanza percorsa in metri.
Per realizzare questa applicazione posizioniamo i seguenti elementi grafici:
• 2 Buttons (Btn_Start e Btn_Stop);

Fig. 12 - Codice grafico dell'applicazione di esempio

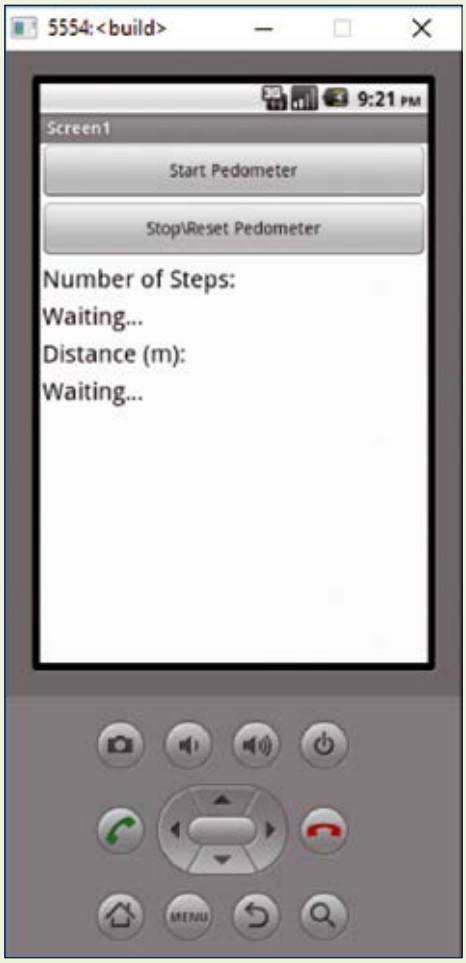

Fig. 13 - Screenshot dell'applicazione di esempio in emulazione.

- 4 Labels (Lbl_Label1, Lbl_Label2, Lbl_StepNum, Lbl_Distance);
- 1 componente pedometer.

A questo punto organizziamo il layout come proposto in **Fig. 11** e passiamo sulla Block view.
La realizzazione dell'app, sfruttando le potenzialità del componente pedometer, risulta piuttosto semplice; infatti sfruttiamo gli eventi button.click sui pulsanti Btn_Start e Btn_Stop rispettivamente per avviare e arrestare il contapassi (invocando i metodi **Start()** e **Stop()**) ed inoltre utilizziamo l'evento Button.LongClick per resettare i contatori (chiamando in sequenza i metodi **Reset()** e **Stop()** e resettando le label così da riportare il valore 0).
Poi sfruttiamo l'evento Pedometer.WalkStep per contare i passi, settando ad ogni occorrenza dell'evento stesso i valori delle due label Lbl_SteNum ed Lbl_Distance, pari ai valori dei parametri WalkSteps e Distance che possiamo recuperare tramite gli appositi metodi di get.
Il codice grafico che implementa quanto spiegato in questo esempio pratico è riportato nella **Fig. 12**.
Invece nella **Fig. 13** trovate uno screenshot dell'applicazione che gira in emulazione sull'emulatore, mentre nella **Fig. 14** riportiamo un ulteriore screenshot ma stavolta catturato da un dispositivo reale, durante il funzionamento con l'app in esecuzione.

Conclusioni

In questa terza puntata abbiamo iniziato ad analizzare più in dettaglio una delle caratteristiche più interessanti di MIT Appinventor, ossia i componenti, concentrandoci in particolare sulle famiglie Storage e Sensors.
Nelle prossime puntate continueremo su questa linea, studiando nuovi componenti (l'elenco completo dei componenti di App Inventor lo trovate nella seconda puntata del corso) passando anche in rassegna componenti che permettono di interagire con dell'hardware esterno allo smartphone.
Quindi vi diamo appuntamento al mese prossimo!

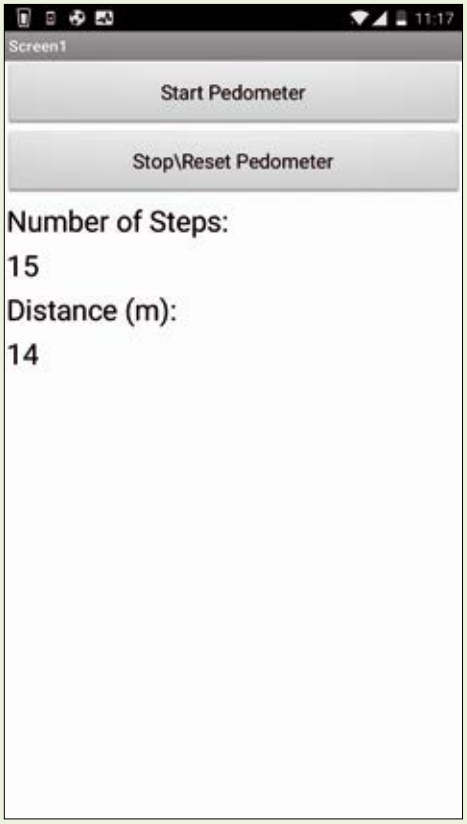

Fig. 14 - Screenshot dell'applicazione di esempio su un dispositivo reale.