Gaia Smart Cities

Summer Training Report

Department of Electronics and Communication Engineering

Dr. B.R. Ambedkar National Institute of Technology

Jalandhar (Punjab)-144011, INDIA

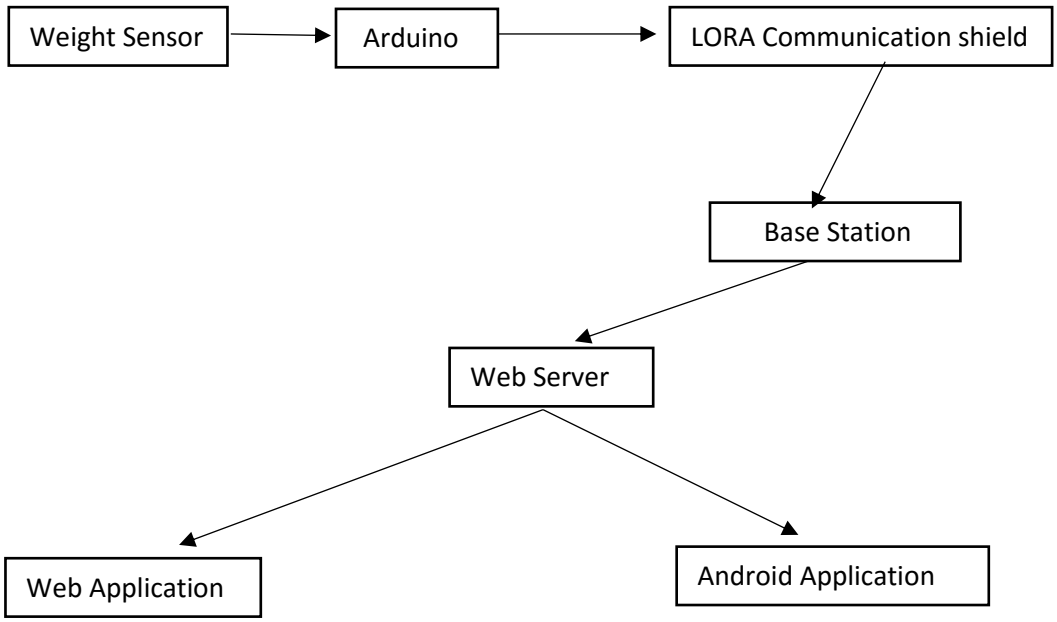
January-June, 2015

Name: Gadiyaram Sri Sai Praveen
Semester : VII
Roll No.:13104073
Class: ECE/G3
Session: 01/06/2016 – 15/07/2016.

INDEX

Project Overview

Structure of the Project :



Description :

Smart Warehouse is an IoT based project designed for the users to monitor the weight of the products of a store, godown, houses etc…., where there is an application of weighing machine. Data collected from the weight sensors are being uploaded to the web server through the help of Arduino and LORA communication shield. Sensors data is perfectly framed in the Database tables. This sensor data is visualized in the form of filling view in Android. Whenever data hits the server user gets notified through this android application.We have used Google's Firebase Cloud Messaging for the notification part.This Android application came with an offline feature, data is stored in the app, but is not the updated one.

Introduction to Android

Android is a mobile operating system developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets. Android's user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input. In addition to touchscreen devices, Google has further developed Android TV for televisions, Android Auto for cars, and Android Wear for wrist watches, each with a specialized user interface. Variants of Android are also used on notebooks, game consoles, digital cameras, and other electronics.

Android may be based on Linux, but it's not based on the type of Linux system you may have used on your PC. Linux makes up the core part of Android, but Google hasn't added all the typical software and libraries you'd find on a Linux distribution like Ubuntu. Linux means the Linux kernel. A kernel is the core part of any operating system.

Android's default user interface is mainly based on direct manipulation, using touch inputs that loosely correspond to real-world actions, like swiping, tapping, pinching, and reverse pinching to manipulate on-screen objects, along with a virtual keyboard Game controllers and full-size physical keyboards are supported via Bluetooth or USB. The response to user input is designed to be immediate and provides a fluid touch interface, often using the vibration capabilities of the device to provide haptic feedback to the user. Internal hardware, such as accelerometers, gyroscopes and proximity sensors[54] are used by some applications to respond to additional user actions, for example adjusting the screen from portrait to landscape depending on how the device is oriented.

android architecture or **Android software stack** is categorized into five parts:

1. linux kernel
2. native libraries (middleware),
3. Android Runtime
4. Application Framework
5. Applications

1) **Linux Kernal**
   It is the heart of android architecture that exists at the root of android architecture. **Linux kernel** is responsible for device drivers, power management, memory management, device management and resource access.

2) **Native Libraries**
   On the top of linux kernel, their are **Native libraries** such as WebKit, OpenGL, FreeType, SQLite, Media, C runtime library (libc) etc. The WebKit library is responsible for browser support, SQLite is for database, FreeType for font support, Media for playing and recording audio and video formats.

3) **Android Runtime**
   In android runtime, there are core libraries and DVM (Dalvik Virtual Machine) which is responsible to run android application. DVM is like JVM but it is optimized for mobile devices. It consumes less memory and provides fast performance.

4) **Application Framework**
   n the top of Native libraries and android runtime, there is android framework. Android framework includes **Android API's** such as UI (User Interface), telephony, resources, locations, Content Providers (data) and package managers. It provides a lot of classes and interfaces for android application development.

5) **Applications**
   On the top of android framework, there are applications. All applications such as home, contact, settings, games, browsers are using android framework that uses android runtime and libraries. Android runtime and native libraries are using linux kernel.

Android Core Components

An android **component** is simply a piece of code that has a well defined life cycle e.g. Activity, Receiver, Service etc.

The **core building blocks** or **fundamental components** of android are activities, views, intents, services, content providers, fragments and AndroidManifest.xml.

Activity : An activity is a class that represents a single screen. It is like a Frame in AWT.

View : A view is the UI element such as button, label, text field etc. Anything that you see is a view.

Intent : Intent is used to invoke components. It is mainly used to:

- Start the service
- Launch an activity
- Display a web page
- Display a list of contacts
- Broadcast a message
- Dial a phone call etc.

Service : Service is a background process that can run for a long time.

There are two types of services local and remote. Local service is accessed from within the application whereas remote service is accessed remotely from other applications running on the same device.

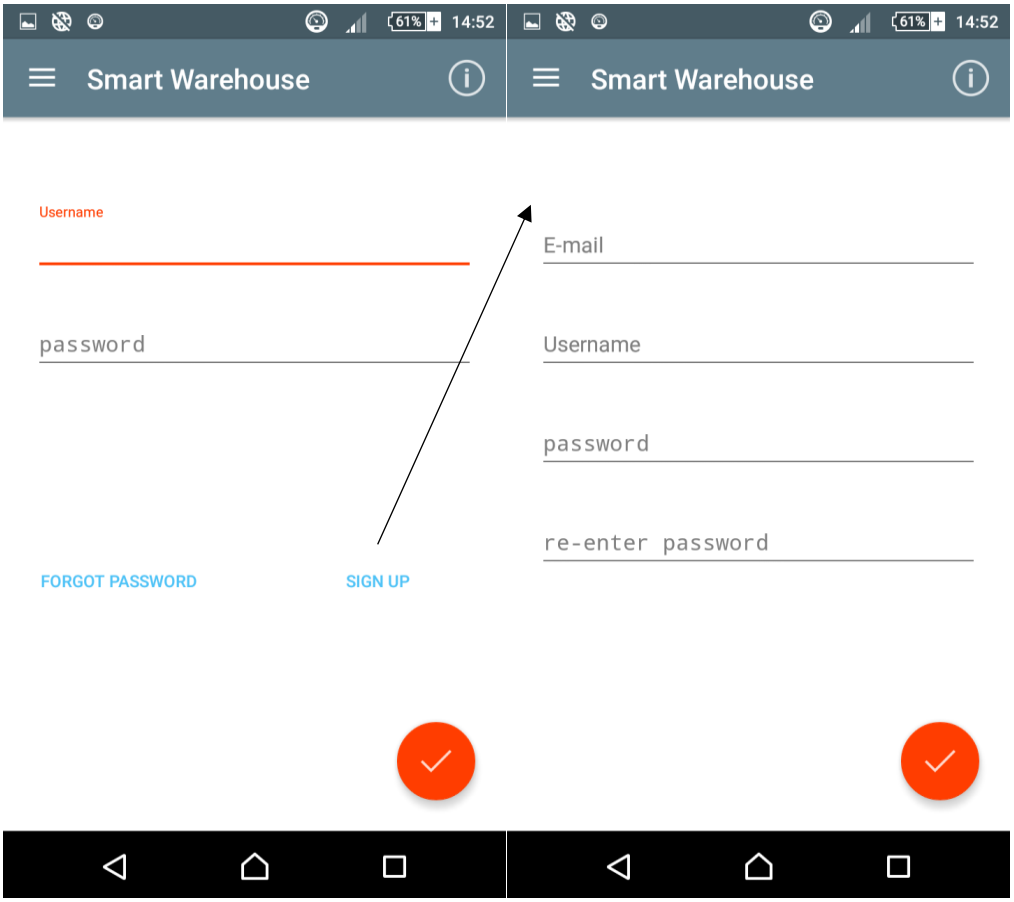
Content Providers : Content Providers are used to share data between the applications.

Fragment : Fragments are like parts of activity. An activity can display one or more fragments on the screen at the same time.

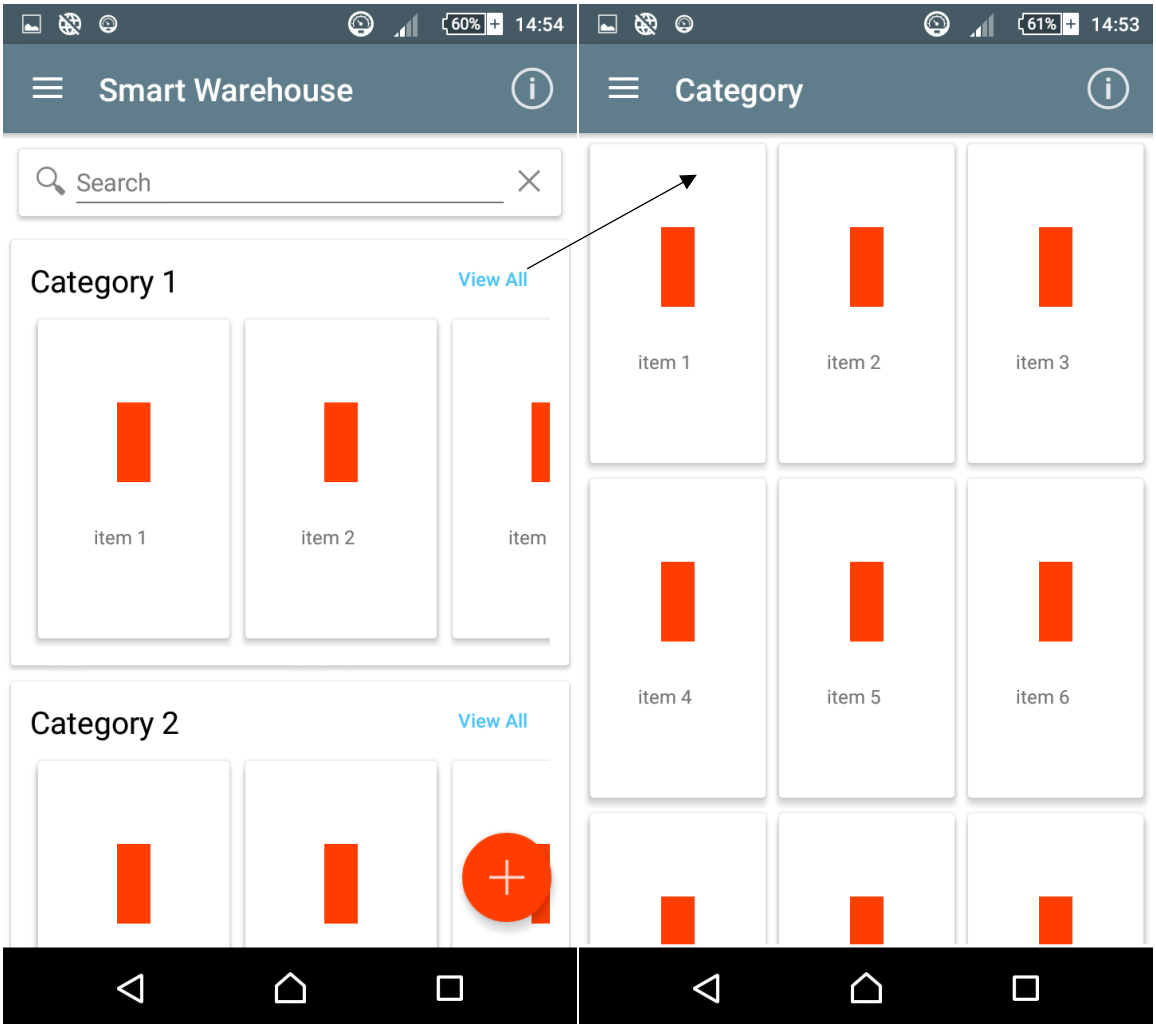
Android Manifest File : It contains informations about activities, content providers, permissions etc. The **AndroidManifest.xml file** contains information of your package, including components of the application such as activities, services, broadcast receivers, content providers etc.

Layouts Designed

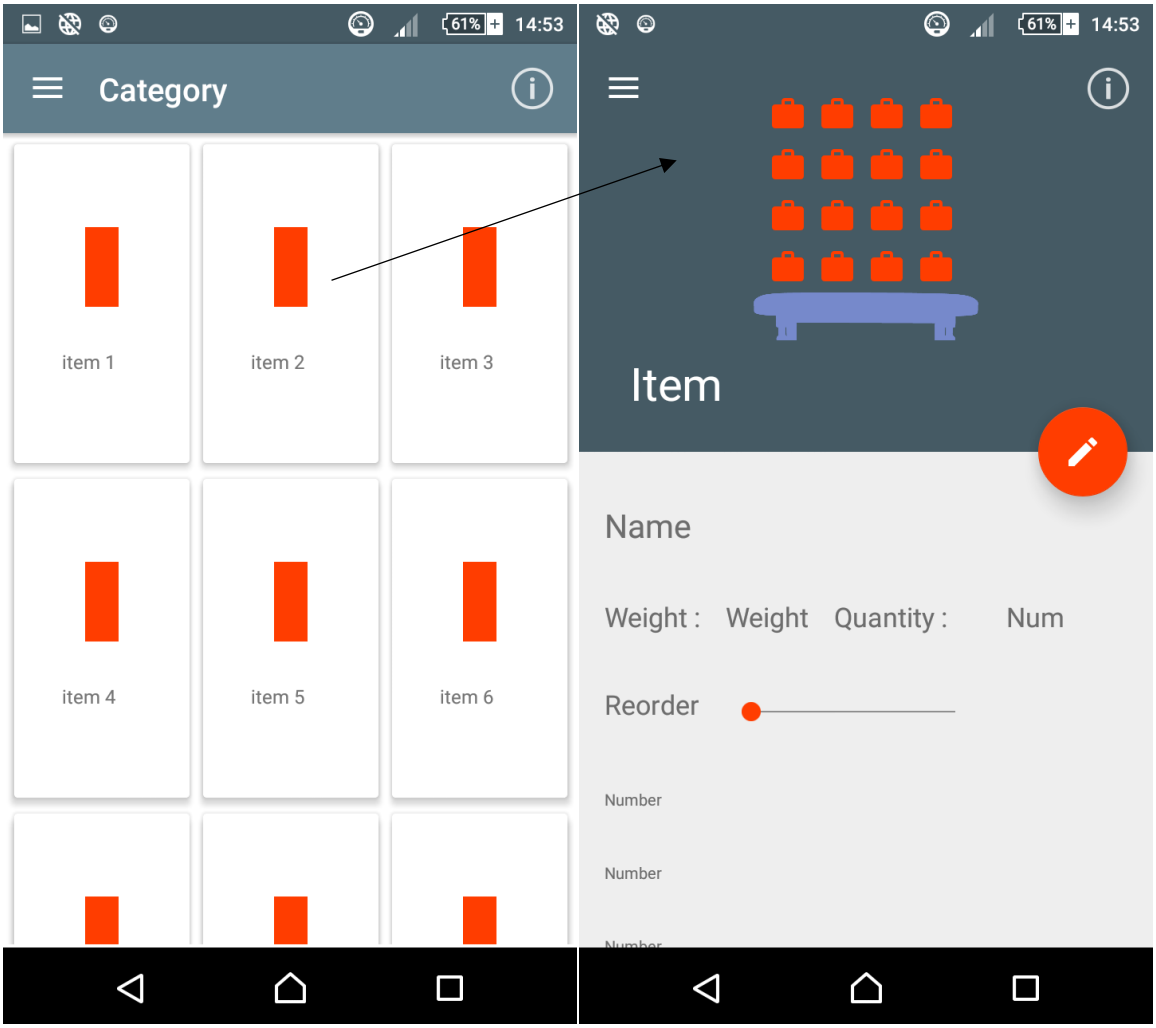
Login Activity and SignUp Activity :-



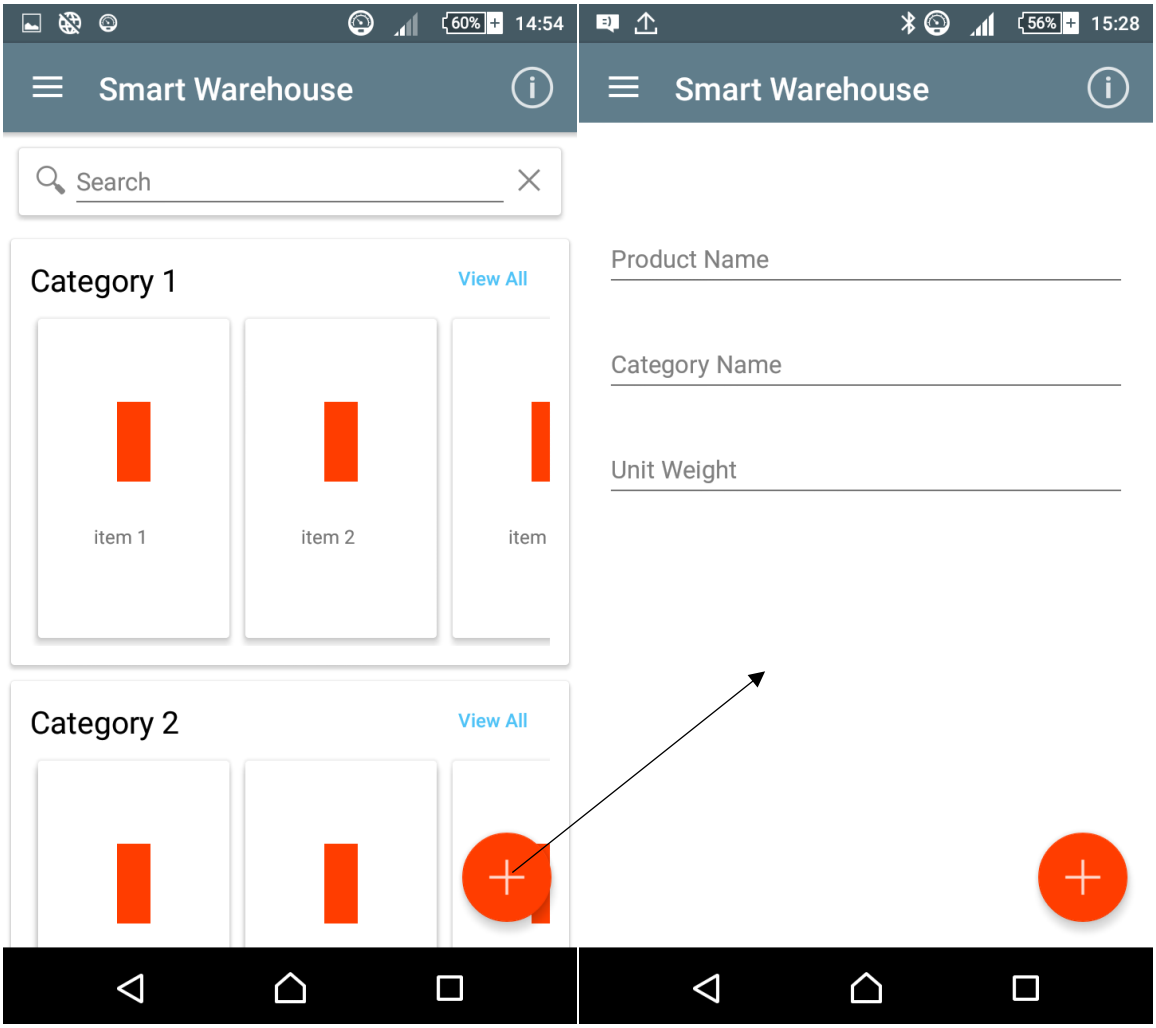
Main Activity and Viewall Actvity :-



Item Activity :-



Add product Activity :-



Design Elements

Material design is a comprehensive guide for visual, motion, and interaction design across platforms and devices. Android now includes support for material design apps. To use material design in your Android apps, functionality available in Android 5.0 (API level 21) and above.

Android provides the following elements for you to build material design apps:

- A new theme
- New widgets for complex views
- New APIs for custom shadows and animations

Coordinator Layout :

CoordinatorLayout is a super-powered **FrameLayout**.

CoordinatorLayout is intended for two primary use cases:

- As a top-level application decor or chrome layout
- As a container for a specific interaction with one or more child views

By specifying Behaviors for child views of a CoordinatorLayout you can provide many different interactions within a single parent and those views can also interact with one another. View classes can specify a default behavior when used as a child of a CoordinatorLayout using the Default Behavior annotation.

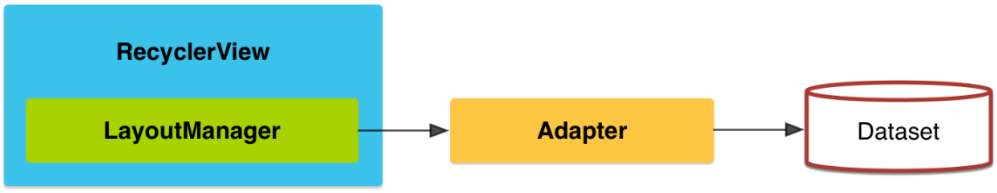
Behaviors may be used to implement a variety of interactions and additional layout modifications ranging from sliding drawers and panels to swipe-dismissable elements and buttons that stick to other elements as they move and animate.

Children of a CoordinatorLayout may have an anchor. This view id must correspond to an arbitrary descendant of the CoordinatorLayout, but it may not be the anchored child itself or a descendant of the anchored child. This can be used to place floating views relative to other arbitrary content panes.

RecyclerView :

The RecyclerView is a new ViewGroup that is prepared to render any adapter-based view in a similar way. It is supposed to be the successor of ListView and GridView, and it can be found in the latest support-v7 version. One of the reasons is that RecyclerView has a more extensible framework, especially since it provides the ability to implement both horizontal and vertical layouts. Use the RecyclerView widget when you have data collections whose elements change at runtime based on user action or network events.
If you want to use a RecyclerView, you will need to work with the following:

- **RecyclerView.Adapter** - To handle the data collection and bind it to the view
- **LayoutManager** - Helps in positioning the items
- **ItemAnimator** - Helps with animating the items for common operations such as Addition or Removal of item



Furthermore, it provides animation support for ListView items whenever they are added or removed, which had been extremely difficult to do in the current implementation. RecyclerView also begins to enforce the ViewHolder pattern too, which was already a recommended practice but now deeply integrated with this new framework.
For more details, see this detailed overview.

Compared to ListView

RecyclerView differs from its predecessor ListView primarily because of the following features:

- **Required ViewHolder in Adapters** - ListView adapters do not require the use of the ViewHolder pattern to improve performance. In contrast, implementing an adapter for RecyclerView requires the use of the ViewHolder pattern.
- **Customizable Item Layouts** - ListView can only layout items in a vertical linear arrangement and this cannot be customized. In contrast, the RecyclerView has a RecyclerView. LayoutManager that allows any item layouts including horizontal lists or staggered grids.
- **Easy Item Animations** - ListView contains no special provisions through which one can animate the addition or deletion of items. In contrast, the RecyclerView has theRecyclerView.ItemAnimator class for handling item animations.
- **Manual Data Source** - ListView had adapters for different sources such as ArrayAdapter andCursorAdapter for arrays and database results respectively. In contrast, theRecyclerView.Adapter requires a custom implementation to supply the data to the adapter.
- **Manual Item Decoration** - ListView has the android:divider property for easy dividers between items in the list. In contrast, RecyclerView requires the use of aRecyclerView.ItemDecoration object to setup much more manual divider decorations.
- **Manual Click Detection** - ListView has a AdapterView.OnItemClickListener interface for binding to the click events for individual items in the list. In contrast, RecyclerView only has support for RecyclerView.OnItemTouchListener which manages individual touch events but has no built-in click handling.

Components of a RecyclerView :

LayoutManagers

A RecyclerView needs to have a layout manager and an adapter to be instantiated. A layout manager positions item views inside a RecyclerView and determines when to reuse item views that are no longer visible to the user.
RecyclerView provides these built-in layout managers:

- **LinearLayoutManager** shows items in a vertical or horizontal scrolling list.
- **GridLayoutManager** shows items in a grid.
- **StaggeredGridLayoutManager** shows items in a staggered grid.

To create a custom layout manager, extend the RecyclerView.LayoutManager class.

Here is Dave Smith's talk on custom layout manager

RecyclerView.Adapter

RecyclerView includes a new kind of adapter. It’s a similar approach to the ones you already used, but with some peculiarities, such as a required ViewHolder. You will have to override two main methods: one to inflate the view and its view holder, and another one to bind data to the view. The good thing about this is that first method is called only when we really need to create a new view. No need to check if it’s being recycled.

ItemAnimator

RecyclerView.ItemAnimator will animate ViewGroup modifications such as add/delete/select that are notified to adapter. DefaultItemAnimator can be used for basic default animations and works quite well. See the section of this guide for more information.

Collapsing Toolbar Layout :

CollapsingToolbarLayout is a wrapper for Toolbar which implements a collapsing app bar. It is designed to be used as a direct child of a AppBarLayout.

1. CoordinatorLayout

A powerful FrameLayout that specifies behavior for child views for various interactions. Allows floating views to be anchored in layout.

2. AppBarLayout

Is essentially a LinearLayout (vertical). It helps respond to its children's scroll events (scroll gestures). Responsible for implementing many features of material design's app bar. Depends heavily on being used as a direct child within CoordinatorLayout.

3. CollapsingToolbarLayout

Wrapper for Toolbar that makes the header image collapse into the Toolbar adjusting its title size.

What's left is the ImageView which holds our actual header's image and Toolbar which we're familiar with.

4. NestedScrollView
It's an special scroll view for the smooth scrolling effect, inside this place the desired content . Here in this example will add several Cards as its children.

Straight from the **developer's blog**:

Flags include :

1. **scroll**: this flag should be set for all views that want to scroll off the screen - for views that do not use this flag, they'll remain pinned to the top of the screen.
2. **enterAlways**: this flag ensures that any downward scroll will cause this view to become visible, enabling the 'quick return' pattern .
3. **enterAlwaysCollapsed**: When your view has declared a minHeight and you use this flag, your View will only enter at its minimum height (i.e., 'collapsed'), only re-expanding to its full height when the scrolling view has reached it's top.
4. **exitUntilCollapsed**: this flag causes the view to scroll off until it is 'collapsed' (its minHeight) before exiting .

Note : all views using the scroll flag must be declared before views that do not use the flag.This ensures that all views exit from the top, leaving the fixed elements behind.

For CollapsingToolbarLayout XML Tag set the layout_scrollFlags property with **scroll|exitUntilCollapsed .**

Collapse Mode

- Parallax scrolling with the ImageView is achieved by simply setting its layout_collapseMode to parallax.
- The Toolbar must use pin as its collapseMode because we want the Toolbar to persist and remain on top as the user scrolls down.

Custom Views

A well-designed custom view is much like any other well-designed class. It encapsulates a specific set of functionality with an easy to use interface, it uses CPU and memory efficiently, and so forth. In addition to being a well-designed class, though, a custom view should:

- Conform to Android standards
- Provide custom styleable attributes that work with Android XML layouts
- Send accessibility events
- Be compatible with multiple Android platforms.

The Android framework provides a set of base classes and XML tags to help you create a view that meets all of these requirements. This lesson discusses how to use the Android framework to create the core functionality of a view class. All of the view classes defined in the Android framework extend **View** class.

- Define custom attributes for your view in a <declare-styleable> resource element
- Specify values for the attributes in your XML layout
- Retrieve attribute values at runtime
- Apply the retrieved attribute values to your view.

AsyncTask

AsyncTask is an abstract class provided by Android which helps us to use the UI thread properly. This class allows us to perform long/background operations and show its result on the UI thread without having to manipulate threads.

Android implements single thread model and whenever an android application is launched, a thread is created. Assuming we are doing network operation on a button click in our application. On button click a request would be made to the server and response will be awaited. Due to single thread model of android, till the time response is awaited our screen is non-responsive. So we should avoid performing long running operations on the UI thread. This includes file and network access.

To overcome this we can create new thread and implement run method to perform this network call, so UI remains responsive.

But since Android follows single thread model and Android UI toolkit is not thread safe, so if there is a need to make some change to the UI based on the result of the operation performed, then this approach may lead some issues.

AsyncTask should ideally be used for operations that take few seconds. Some tasks keep the thread running for long time so in that case it is recommended to use java.util.concurrent package such as Executor, ThreadPoolExecutor and FutureTask.

AsyncTask has four steps:

1. **doInBackground:** Code performing long running operation goes in this method. When onClick method is executed on click of button, it calls execute method which accepts parameters and automatically calls doInBackground method with the parameters passed.
2. **onPostExecute:** This method is called after doInBackground method completes processing. Result from doInBackground is passed to this method.
3. **onPreExecute:** This method is called before doInBackground method is called.
4. **onProgressUpdate:** This method is invoked by calling publishProgress anytime from doInBackground call this method.

The task can be cancelled by invoking cancel(boolean) method. This will cause subsequent calls to **isCancelled()**to return true. After invoking this method, **onCancelled(Object)** method is called instead of onPostExecute() after doInBackground() returns.

SQLite Android

SQLite is an Open Source database. SQLite supports standard relational database features like SQL syntax, transactions and prepared statements. The database requires limited memory at runtime (approx. 250 KByte) which makes it a good candidate from being embedded into other runtimes.

SQLite supports the data types TEXT (similar to String in Java), INTEGER (similar to long in Java) and REAL (similar to double in Java). All other types must be converted into one of these fields before getting saved in the database. SQLite itself does not validate if the types written to the columns are actually of the defined type, e.g. you can write an integer into a string column and vice versa.

SQLite is embedded into every Android device. Using an SQLite database in Android does not require a setup procedure or administration of the database.

To create and upgrade a database in your Android application you create a subclass of the SQLiteOpenHelperclass. In the constructor of your subclass you call the super() method of SQLiteOpenHelper , specifying the database name and the current database version.

In this class you need to override the following methods to create and update your database.

- onCreate() - is called by the framework, if the database is accessed but not yet created.
- onUpgrade() - called, if the database version is increased in your application code. This method allows you to update an existing database schema or to drop the existing database and recreate it via the onCreate()method.

Both methods receive an SQLiteDatabase object as parameter which is the Java representation of the database.

The SQLiteOpenHelper class provides the getReadableDatabase() and getWriteableDatabase() methods to get access to an SQLiteDatabase object; either in read or write mode.

The database tables should use the identifier _id for the primary key of the table. Several Android functions rely on this standard.

SQLiteDatabase :

SQLiteDatabase is the base class for working with a SQLite database in Android and provides methods to open, query, update and close the database.

More specifically SQLiteDatabase provides the insert() , update() and delete() methods.

In addition it provides the execSQL() method, which allows to execute an SQL statement directly.

The object ContentValues allows to define key/values. The key represents the table column identifier and thevalue represents the content for the table record in this column. ContentValues can be used for inserts and updates of database entries.

Queries can be created via the rawQuery() and query() methods or via the SQLiteQueryBuilder class .

rawQuery() directly accepts an SQL select statement as input.

query() provides a structured interface for specifying the SQL query.

SQLiteQueryBuilder is a convenience class that helps to build SQL queries.

Cursor :

A query returns a Cursor object. A Cursor represents the result of a query and basically points to one row of the query result. This way Android can buffer the query results efficiently; as it does not have to load all data into memory.

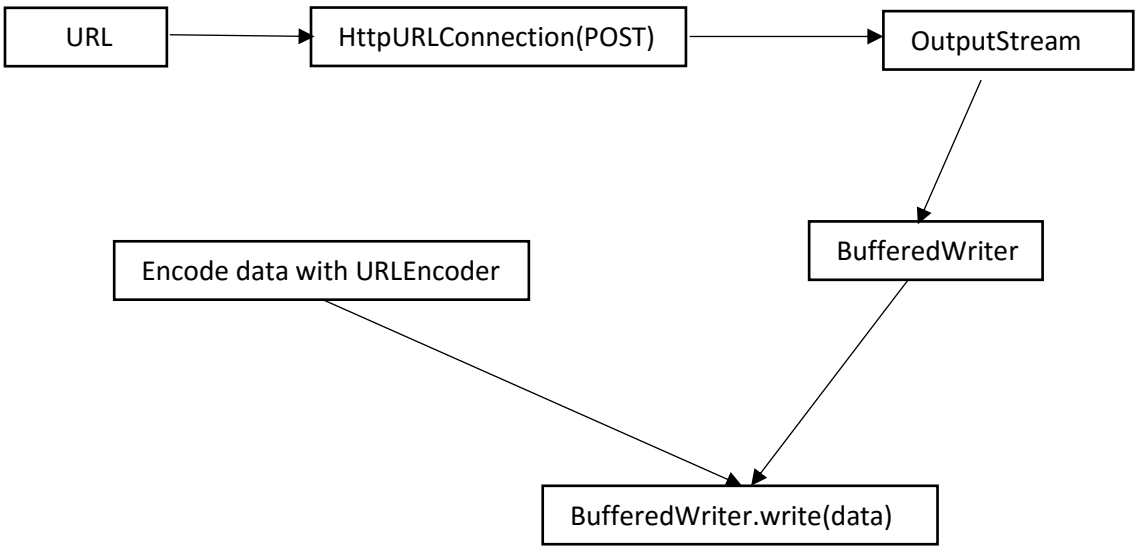
To get the number of elements of the resulting query use the getCount() method.

To move between individual data rows, you can use the moveToFirst() and moveToNext() methods. The

isAfterLast() method allows to check if the end of the query result has been reached.

Sending Data to server

Always call server with the use of thread and handlers, if not using request with thread then server request will lock activity to complete request , if user will interact with activity before server request complete then activity will give ANR (FORCE CLOSE ERROR) problem. App will Crash suddenly.



This is the process of sending a POST request to server

URL :

The **Java URL** class represents an URL. URL is an acronym for Uniform Resource Locator. It points to a resource on the World Wide Web.

A URL contains many information:

1. **Protocol:** In this case, http is the protocol.
2. **Server name or IP Address:** In this case, www.abc.com is the server name.
3. **Port Number:** It is an optional attribute. If port number is not mentioned in the URL, it returns -1.
4. **File Name or directory name:** In this case, index.php is the file name.

HttpURLConnection :

It is a URLConnection with support for HTTP-specific features. HttpURLConnection is used to send GET/POST requests to the server and to take the response from it.

OutputStream :

OutputStream class is an abstract class.It is the superclass of all classes representing an output stream of bytes. An output stream accepts output bytes and sends them to some sink.

BufferedWriter :

The **Java.io.BufferedWriter** class writes text to a character-output stream, buffering characters so as to provide for the efficient writing of single characters, arrays, and strings.

Server side Scripting

The entire user information and products information are maintained in databses in online server. In order to insert,update,delete personal or products information from server, we use **php** and **mysql**.

PHP :

The PHP Hypertext Preprocessor (PHP) is a programming language that allows web developers to create dynamic content that interacts with databases. PHP is basically used for developing web based software applications.

MYSQL :

MySQL is the most popular Open Source Relational SQL database management system. MySQL is one of the best RDBMS being used for developing web-based software applications.

JSON Decoding

JSON :

JSON or JavaScript Object Notation is a lightweight text-based open standard designed for human-readable data interchange. Conventions used by JSON are known to programmers, which include C, C++, Java, Python, Perl, etc.

- JSON stands for JavaScript Object Notation.
- The format was specified by Douglas Crockford.
- It was designed for human-readable data interchange.
- It has been extended from the JavaScript scripting language.
- The filename extension is **.json.**
- JSON Internet Media type is **application/json.**
- The Uniform Type Identifier is public.json.

Uses of JSON :

- It is used while writing JavaScript based applications that includes browser extensions and websites.
- JSON format is used for serializing and transmitting structured data over network connection.
- It is primarily used to transmit data between a server and web applications.
- Web services and APIs use JSON format to provide public data.
- It can be used with modern programming languages.

Syntax of JSON :

- Data is represented in name/value pairs.
- Curly braces hold objects and each name is followed by ':'(colon), the name/value pairs are separated by , (comma).
- Square brackets hold arrays and values are separated by ,(comma).

JSON Data types :

Number	double- precision floating-point format in JavaScript
String	double-quoted Unicode with backslash escaping
Boolean	true or false
Array	an ordered sequence of values
Value	it can be a string, a number, true or false, null etc
Object	an unordered collection of key:value pairs
Whitespace	can be used between any pair of tokens
null	empty

Encoding JSON in PHP:

PHP **json_encode()** function is used for encoding JSON in PHP. This function returns the JSON representation of a value on success or FALSE on failure.

String data=**json_encode**($value[,$options=0]);

- **value** – The value being encoded. This function only works with UTF-8 encoded data.
- **options** – This optional value is a bitmask consisting of JSON_HEX_QUOT, JSON_HEX_TAG, JSON_HEX_AMP, JSON_HEX_APOS, JSON_NUMERIC_CHECK, JSON_PRETTY_PRINT, JSON_UNESCAPED_SLASHES, JSON_FORCE_OBJECT.

Decoding JSON in Java/Android :

JSONObject and JSONArray are used in order to parse a json string.

Ex:-

```
JSONArray jsonarray = new JSONArray(jsonStr);
for (int i = 0; i < jsonarray.length(); i++) {
   JSONObject jsonobject = jsonarray.getJSONObject(i);
   String name = jsonobject.getString("name");
   String url = jsonobject.getString("url");
}
```

Decoding JSON in PHP :

PHP json_decode() function is used for decoding JSON in PHP. This function returns the value decoded from json to appropriate PHP type.

- **json_string** – It is an encoded string which must be UTF-8 encoded data.
- **assoc** – It is a boolean type parameter, when set to TRUE, returned objects will be converted into associative arrays.
- **depth** – It is an integer type parameter which specifies recursion depth
- **options** – It is an integer type bitmask of JSON decode, JSON_BIGINT_AS_STRING is supported.

Firebase Cloud Messaging

An FCM implementation includes an app server that interacts with FCM via HTTP or XMPP protocol, and a client app. You can compose and send messages using the app server or the Notifications console.

Firebase Notifications is built on Firebase Cloud Messaging and shares the same FCM SDK for client development. For testing or for sending marketing or engagement messages with powerful built-in targeting and analytics, you can use Notifications. For deployments with more complex messaging requirements, FCM is the right choice.

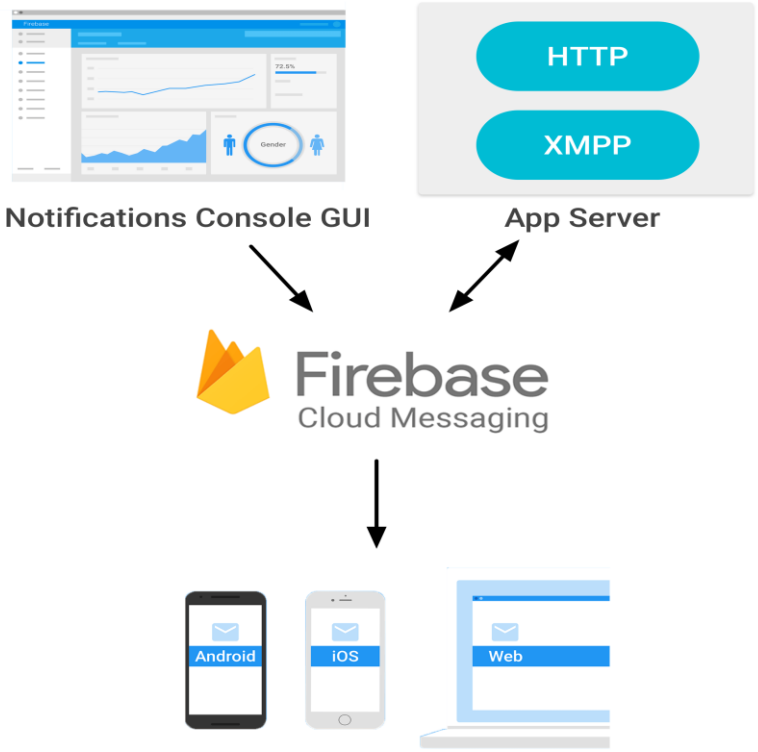
The server side of Firebase Cloud Messaging consists of two components:

- **FCM connection servers** provided by Google. These servers take messages from an app server and send them to a client app running on a device. Google provides connection servers for HTTP and XMPP.
- An **app server** that you must implement in your environment. This app server sends data to a client app via the chosen FCM connection server, using the appropriate XMPP or HTTP protocol.

With FCM, you can send **two types of messages** to clients:

- **Notification messages**, sometimes thought of as "display messages."
- **Data messages**, which are handled by the client app.

A notification message is the more lightweight option, with a **2KB** limit and a predefined set of user-visible keys. Data messages let developers send up to **4KB** of custom key-value pairs. Notification messages can contain an optional data payload, which is delivered when users tap on the notification.



Setup with Android :

Open your project build.gradle file and add the following dependencies

```
dependencies {
    classpath 'com.android.tools.build:gradle:2.1.0'
    classpath 'com.google.gms:google-services:3.0.0'
}
```

Now open your app module build.grade and add these dependencies

```
compile 'com.google.firebase:firebase-core:9.0.1'
compile 'com.google.firebase:firebase-messaging:9.0.1'
```

At the end of the file we need to apply our google play servcies plugin :

```
apply plugin: 'com.google.gms.google-services'
```

Sync your gradle and you will have the firebase setup in your project.

Creating TokenService :

Each device need to be uniquely identified so that we can check success/failure and target individual user. Firebase provides **FirebaseInstanceId** class which takes care of creating unique device token for the current device.
The token is refreshed randomly based on duration or other things so to avoid checking each time for new token, We have to use a Service class that extends**FirebaseInstanceIdService**.
In it, the **onTokenRefresh** is called each time a new token is generated.

Registering Device online :

For our TokenService to actually work we need to make sure it is added to our**AndroidManifest.xml** file. Open up the manifest and inside our **application** tag add the service lines.

```
<service
    android:name=".TokenService">
    <intent-filter>
       <action android:name="com.google.firebase.INSTANCE_ID_EVENT"/>
    </intent-filter>
</service>
```

Creating Message Service :

We have to extend **FirebaseMessagingService**. This class has a method **onMessageReceived** which is called when a FCM is received in foreground.

Notifications in Android

Notifications are used to alert users on some events that requires their attention. Notifications alert the users through various forms:

- Display a status bar icon
- Flash the LED
- Vibrate
- Ringtones

Status bar Notifications :

Status bar notifications indicates some ongoing background services such as downloading or playing music or displays an alert/information.To see the notification user needs to open the notification drawer. Notifications are handled by Notification Manger in Android. Notification Manager is a system service used to manage notifications.

To create a status bar notification we need to use two classes, NotificationManager, Notification.
Lets see an simple example to display notification in status bar.
The NotificationManager is a system Service used to manage Notification. Get a reference to it using the getSystemService() method.

```
NotificationManager notificationManager = (NotificationManager)
getSystemService(NOTIFICATION_SERVICE);
```

Using the Notification Manager you can trigger new notifications, modify existing ones, or cancel that are no longer required.

Notification should contain - icon, text, Time Stamp.
text - is the short description of the alert such as 'New message', 'New email'
Time Stamp- Time stamp is used to sort the notifications, NotificationManager will sort the notifications.

```
Notification notification = new Notification(R.drawable.ic_launcher, "New Message",
System.currentTimeMillis());
```

Next, lets create a pending intent which will be invoked when the message in status bar is clicked. Pending intents are intents and an action to do so that it can passed to some other applications which can execute this pendingintent.

```
Intent notificationIntent = new Intent(this, MainActivity.class);
PendingIntent pendingIntent = PendingIntent.getActivity(this, 0,notificationIntent, 0);
```

I have passed the current activity to pendingIntent. The pendingINtent specified will be fired of a user clicks notification item.

The simplest approach is to use the setLatestEventInfo() method to specify the title and text fields.

```
notification.setLatestEventInfo(context, notificationTitle, notificationMessage,
pendingIntent)
```

Finally post your notification to the status bar with a id. If a notification with same id already exists, it will get replaced with updated information.

Weight Sensors and Its Working

Weight sensor also called a **load cell** is a physical element (or transducer if you want to be technical) that can translate pressure (force) into an electrical signal.

Types of Load Cell

1. Hydraulic Load Cells

Hydraulic load cells use a conventional piston and cylinder arrangement to convey a change in pressure by the movement of the piston and a diaphragm arrangement which produces a change in the pressure on a Bourdon tube connected with the load cells.

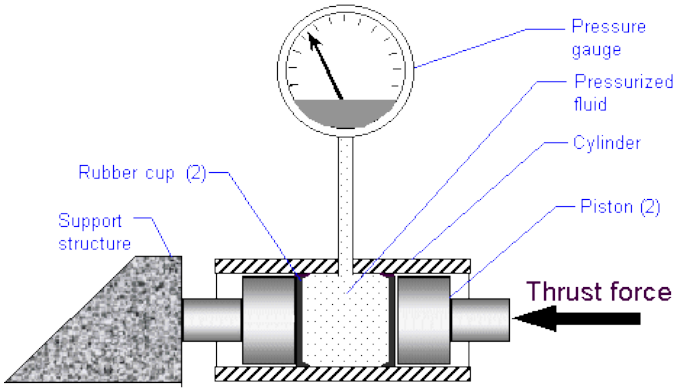


Diagram of a Hydraulic Load Cell from Nikka's Rocketry

2. Pneumatic Load Cells

Pneumatic load cells use air pressure applied to one end of a diaphragm, and it escapes through the nozzle placed at the bottom of the load cell, which has a pressure gauge inside of the cell.

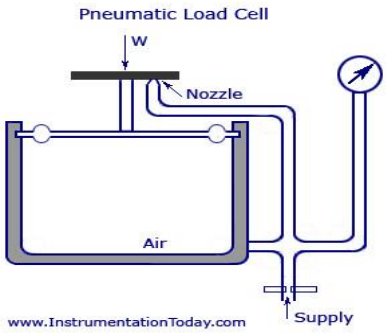
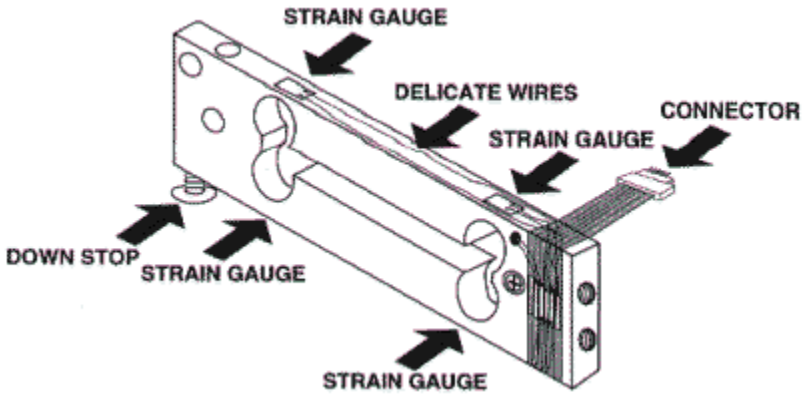


Diagram of a pneumatic load cell from Instrumentation Today

3. Strain Gauge Load Cells

And lastly (though there are many other less common load cell set ups), there is a strain gauge load cell, which is a mechanical element of which the force is being sensed by the deformation of a (or several) strain gauge(s) on the element.



Strain gauge load cell diagram from Scalenet.com

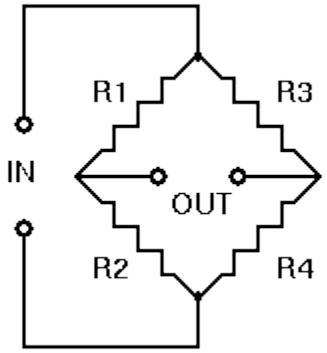
In bar strain gauge load cells, the cell is set up in a "Z" formations so that torque is applied to the bar and the four strain gauges on the cell will measure the bending distortion, two measuring compression and two tension. When these four strain gauges are set up in a wheatctone bridge formation, it is easy to accurately measure the small changes in resistance from the strain gauges.

Strain Gauge Basics

A strain gauge is a device that measures electrical resistance changes in response to, and proportional of, strain (or pressure or force or whatever you so desire to call it) applied to the device. The most common strain gauge is made up of very fine wire, or foil, set up in a grid pattern in such a way that there is a linear change in electrical resistance when strain is applied in one specific direction, most commonly found with a base resistance of 120Ω, 350Ω, and 1,000Ω.

Each strain gauge has a different sensitivity to strain, which is expressed quantitatively as the gauge factor (GF). The gauge factor is defined as the ratio of fractional change in electrical resistance to the fractional change in length (strain).

A good way of taking small changes in resistance and turning it into something more measurable is using a wheat stone bridge. A wheatstone bridge is a configuration of four resistors with a known voltage applied like this:

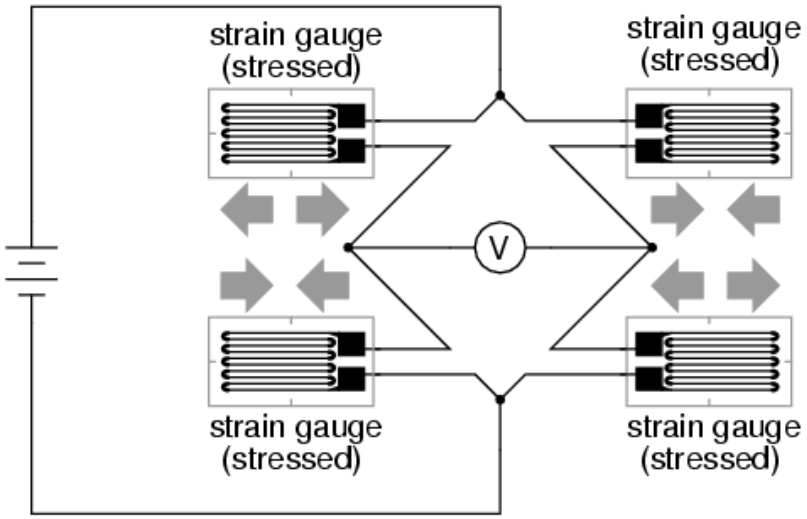


where Vin is a known constant voltage, and the resulting Vout is measured.
If $R1/R2 = R3/R4$ then Vout is 0, but if there is a change to the value of one of the resistors, Vout will have a resulting change that can be measured and is governed by the following equation using ohms law:

$$Vout = [(R3/(R3 + R4) - R2/(R1 + R2))] * Vin$$

By replacing one of the resistors in a wheatstone bridge with a strain gauge, we can easily measure the change in Vout and use that to assess the force applied.

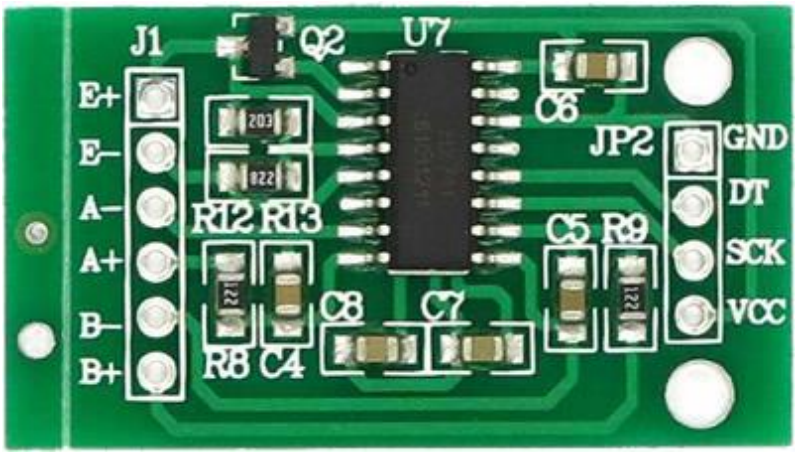
Full-bridge strain gauge circuit



COMPONENTS

1. The Load Cell I used :

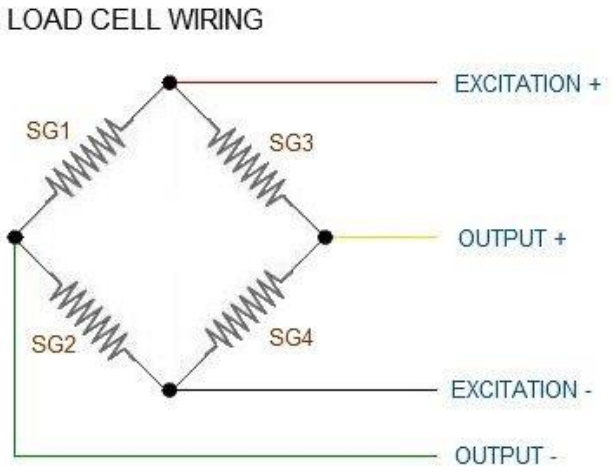
2. Load Cell Amplifier HX71

The HX711 amplifier 

The HX711 load cell amplifier is used to get measurable data out from a load cell and strain gauge.

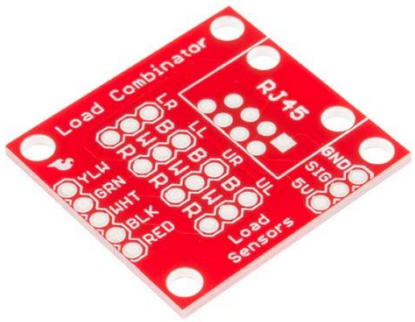
Strain gauges are two wired organized metal foil or wires that are set up in such a way that the resistance changes when it is compressed or stretched. When a strain gauge is placed on something (usually metallic in nature) its resistance changes based on the stress experienced by that something. When a single strain gauge is hooked up to a metallic cell, we are calling that a load sensors, which have three output wires. Load cells usually has four strain gauges hooked up in a wheatstone bridge formation, which have four output wires.

The HX711 Load Cell Amplifier accepts five wires from the load cell. These pins are labeled with colors; **RED, BLK, WHT, GRN, and YLW.** These colors correspond to the conventional color coding of load cells, where red, black, green and white wires come from the strain gauge on the load cell and yellow is an optional ground wire that is not hooked up to the strain gauge but is there to ground any small outside EMI (electromagnetic interference). Sometimes instead of a yellow wire there is a larger black wire, foil, or loose wires to shield the signal wires to lessen EMI.



Four strain gauges (SG1 through 4) hooked up in a wheatstone bridge formation

3. LOAD CELL COMBINATOR

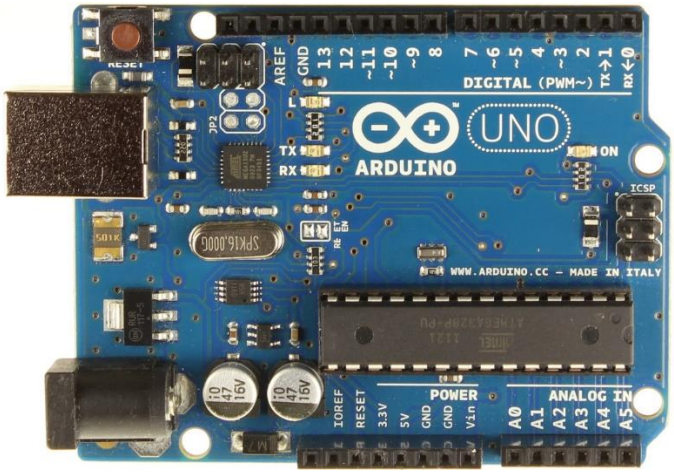


If you would like to set up four single load sensors with our combinator board and amplifier, connect the five pins labeled RED, BLK, WHT, GRN, YLW to the matching pins on the HX711.

The combinator board also has room for an 8 pin RJ45 socket, which can be used to connect your project via Ethernet cables for long distance applications.

Another nice thing about our combinator board is that most home scales use four single strain gauge load sensors, so this is a handy board for hacking your own scales at home

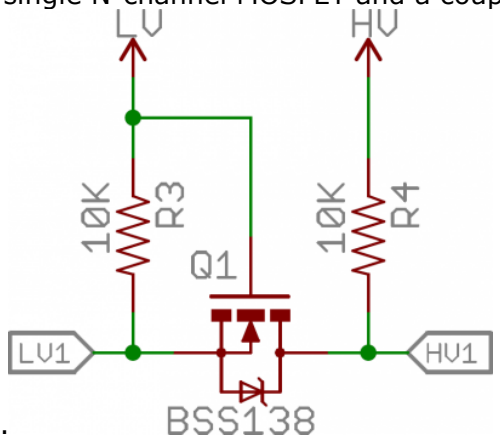
4. ARDUINO UNO



Arduino/Genuino Uno is a microcontroller board based on the ATmega328P (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

5. Bi-Directional Logic Level Converter

This is basically one level-shifting circuit on the board, which is repeated four times to create four level-shifting channels. The circuit uses a single N-channel MOSFET and a couple pull-up resistors to realize bi-directional level shifting.



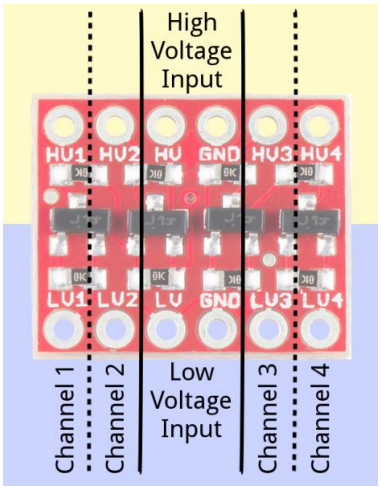
The bi-directional level-shifting circuit used on all four channels of the BD-LLC.

Through some semiconductor magic, this circuit can shift a low voltage signal to high *and/or* shift a high-voltage signal to a low voltage. A 0V signal on one end remains a 0V signal on the other.

Voltage Inputs

The pins labeled ***HV***, ***LV***, and two ***GND***'s provide high and low **voltage references** to the board. Supplying a steady, regulated voltage to both of these inputs is **required**.

The voltage supplied to the *HV* and *GND* inputs should be higher than that supplied to the *LV* side. For example, if you're interfacing from 5V to 3.3V, the voltage on the *HV* pin should be 5V, and the voltage on *LV* sould be 3.3V.



Data Channels

There are four separate data channels on the BD-LLC, each capable of shifting data to and from high and low voltages. These pins are labeled ***HV1***, ***LV1***, ***HV2***, ***LV2***, ***HV3***, ***LV3***, ***HV4***, and ***LV4***. The number at the end of each label designates the channel of the pin, and the *HV* or *LV* prefix determines whether it's on the high or low side of the channel.

A low-voltage signal sent in to *LV1*, for example, will be shifted up to the higher voltage and sent out *HV1*. Something sent in *HV3* will be shifted down and sent out of *LV3*. Use as many of these channels as your project requires. You don't have to use every single one.

6. LoRa Module(SX1272)

The SX1272 LoRa module uses the SPI pins for communication. The SPI port allows more speed communication. The Multiprotocol Radio Shield allows to connect two communication modules at the same time to Arduino or Intel Galileo. This means a lot of different combinations are possible using any of the radios available for Arduino.

LoRa Communication

LoRa is a new, private and spread-spectrum modulation technique which allows sending data at extremely low data-rates to extremely long ranges. The low data-rate (down to few bytes per second) and LoRa modulation lead to very low receiver sensitivity (down to **-134 dBm**), which combined to an output power of **+14 dBm** means extremely large link budgets: up to **148 dB.**, what means more than **22km (13.6 miles)** in **LOS** links and up to **2km (1.2miles)** in **NLOS** links in urban environment (going through buildings).

Libelium's LoRa module works in both 868 and 900 MHz ISM bands, which makes it suitable for virtually any country. Those frequency bands are lower than the popular 2.4 GHz band, so path loss attenuation is better in LoRa. In addition, 868 and 900 MHz are bands with much fewer interference than the highly populated 2.4 GHz band. Besides, these low frequencies provide great penetration in possible materials (brick walls, trees, concrete), so these bands get less loss in the presence of obstacles than higher bands.

The great performance of LoRa in all these 3 features (good sensitivity, low path loss, good obstacle penetration) makes LoRa a disruptive technology enabling really long range links. This is specially important in urban scenarios, with very difficult transmission conditions. To sum up, LoRa can get long ranges in Smart Cities deployments, so it reduces dramatically the size of the backbone network (repeaters, gateways or concentrators).

LoRa VS LoRaWAN

Libelium currently offers two options of this type of radio technology: LoRa and LoRaWAN

- LoRa contains only the link layer protocol and is perfect to be used in P2P communications between nodes. LoRa modules are a little cheaper that the LoRaWAN ones. It works in the 868 and 900MHz bands.
- LoRaWAN includes the network layer too so it is possible to send the information to any Base Station already connected to a Cloud platform. LoRaWAN modules may work in the 868/900/433MHz bands.

- LoRa mode

The innovative LoRa mode is the most interesting included in this module. It is an advanced and private modulation that increases the range comparing to classic modulations. The LoRa long range mode provides ultra-long range spread spectrum communication and high interference immunity whilst minimizing current consumption. It combines digital spread spectrum, digital signal processing, and forward error correction coding to achieve unprecedented performance. LoRa also provides significant advantages in both blocking and selectivity over conventional modulation techniques.

LoRa has three configurable parameters: the bandwidth (BW), the coding rate (CR) and the spreading factor (SF). The combination of these values defines the transmission mode. There are ten predefined modes in the API, including the largest distance mode, the fastest mode, and eight other intermediate modes that Libelium has found interesting.

Mode	BW	CR	SF	Sensitivity (dB)
1	125	4/5	12	-134
2	250	4/5	12	-131
3	125	4/5	10	-129
4	500	4/5	12	-128
5	250	4/5	10	-126
6	500	4/5	11	-125.5
7	250	4/5	9	-123
8	500	4/5	9	-120
9	500	4/5	8	-117
10	500	4/5	7	-114

- Power gain and sensibility

When configuring a node and a network, one important parameter is related with power gain and sensibility.

Power level (dBm) at which the module transmits conducted power.

Its possible values are:

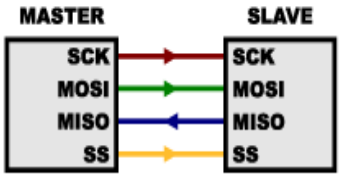
Parameter	SX1272 power level
'L'	0 dBm
'H'	7 dBm
'M'	14 dBm

It is also possible to set the conducted power indicating the quantity as a parameter in the function setPower.

- RSSI of one packet and RSSI of the channel

It reports the Received Signal Strength of the last received packet and the current value of the Received Signal Strength in the selected channel. The RSSI of the packet is the meaningful one: if its value is greater than the sensitivity the packet sent is going to be successfully detected, otherwise the packet will be lost. The RSSI of the channel reports the signal level detected in every moment, even if it is not signal being transmitted, so it provides also noise level information. In the case the user develops a multi-hop network, this parameters only indicate the signal strength of the last hop, so it does not provide an accurate quality measurement of a multihop link.

SPI COMMUNICATION



The LoRa Module Uses SPI communication to get instructions from Arduino Uno and to give back data.
Here Master is our Arduino Uno and Slave is the Lora Module.

SPI has a "synchronous" data bus, which means that it uses separate lines for data and a "clock" that keeps both sides in perfect sync. The clock is an oscillating signal that tells the receiver exactly when to sample the bits on the data line. In SPI, only one side generates the clock signal (usually called CLK or SCK for Serial ClocK). The side that generates the

clock is called the "master", and the other side is called the "slave". There is always only one master (which is almost always your microcontroller), but there can be multiple slaves (more on this in a bit).

When data is sent from the master to a slave, it's sent on a data line called **MOSI**, for "Master Out Slave In". If the slave needs to send a response back to the master, the master will continue to generate a prearranged number of clock cycles, and the slave will put the data onto a third data line called **MISO**, for "Master In / Slave Out".

Slave Select tells the slave that it should wake up and receive / send data and is also used when multiple slaves are present to select the one you'd like to talk to.The SS line is normally held high, which disconnects the slave from the SPI bus. (This type of logic is known as "active low," and you'll often see used it for enable and reset lines.) Just before data is sent to the slave, the line is brought low, which activates the slave.