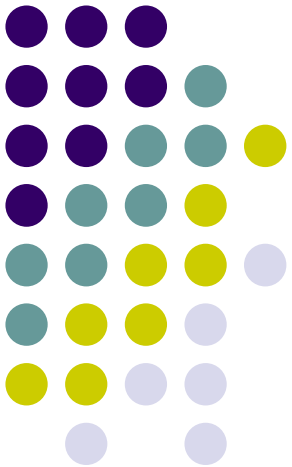
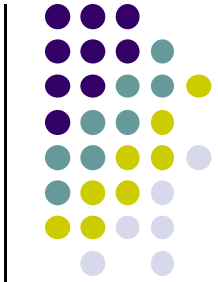
CS 403X Mobile and Ubiquitous Computing

Lecture 12: Activity Recognition

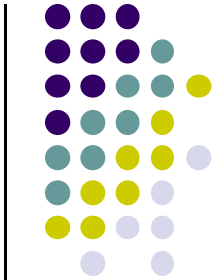
Emmanuel Agu





Activity Recognition Using Google API

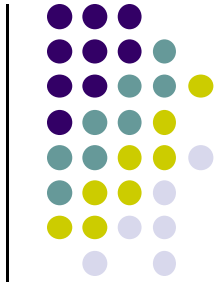
Activity Recognition



- Activity Recognition? Detect what user is doing?
  - Part of user's context
- Examples: sitting, running, driving, walking
- Why? App can adapt it's behavior based on user behavior
- E.g. If user is driving, don't send notifications

https://www.youtube.com/watch?v=S8sugXgUVEI

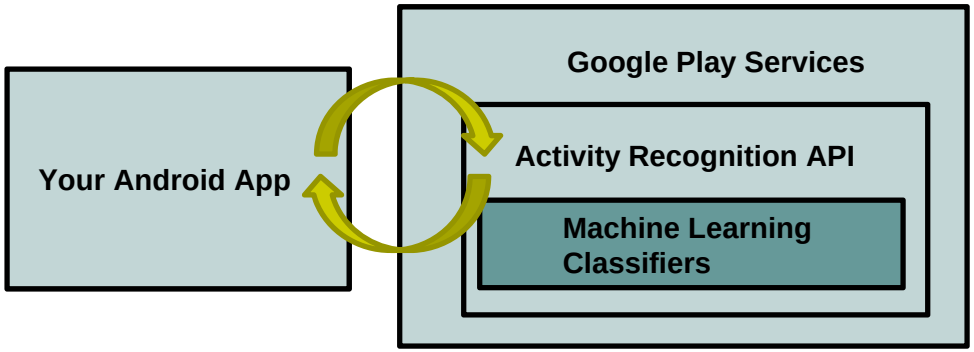
Google Activity Recognition API



- API to detect smartphone user's current activity
- Programmable, can be used by your Android app
- Currently detects 6 states:
  - In vehicle
  - On Bicycle
  - On Foot
  - Still
  - Tilting
  - Unknown

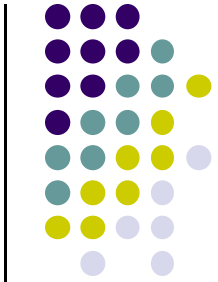
Google Activity Recognition API

- Deployed as part of Google Play Services



Activity Recognition Using Google Fit

Ref: How to Recognize User Activity with Activity Recognition by Paul Trebilcox-Ruiz on Tutsplus.com tutorials



- Example code for this tutorial on gitHub:

  https://github.com/tutsplus/Android-ActivityRecognition

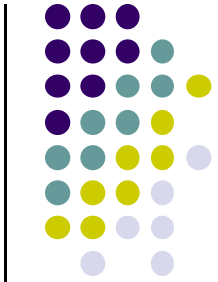
- Google Activity Recognition can:
  - Recognize user's current activity (Running, walking, in a vehicle or still)

- Project Setup:
  - Create Android Studio project with blank Activity (minimum SDK 14)
  - In **build.gradle** file, define latest Google Play services (8.4) as dependency

```
compile 'com.google.android.gms:play-services:8.4.0'
```

Activity Recognition Using Google Fit

Ref: How to Recognize User Activity with Activity Recognition by Paul Trebilcox-Ruiz on Tutsplus.com tutorials



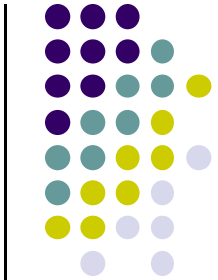
- Create new class **ActivityRecognizedService** which extends **IntentService**
- **IntentService:** type of service, asynchronously handles work off main thread as Intent requests.
- Throughout user's day, **Activity Recognition API** sends user's activity to this IntentService in the background
- Need to program this Intent to handle incoming user activity

```
01  public class ActivityRecognizedService extends IntentService {
02
03      public ActivityRecognizedService() {
04          super("ActivityRecognizedService");
05      }
06
07      public ActivityRecognizedService(String name) {
08          super(name);
09      }
10
11      @Override
12      protected void onHandleIntent(Intent intent) {
13      }
14  }
```

Called to deliver User's activity

Activity Recognition Using Google Fit

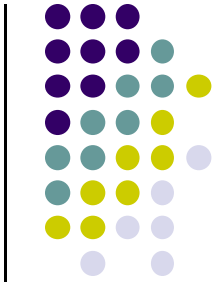
Ref: How to Recognize User Activity with Activity Recognition by Paul Trebilcox-Ruiz on Tutsplus.com tutorials



- Modify **AndroidManifest.xml** to
  - Declare **ActivityRecognizedService**
  - Add com.google.android.gms.permission.ACTIVITY_RECOGNITION permission

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.tutsplus.activityrecognition">

    <uses-permission
android:name="com.google.android.gms.permission.ACTIVITY_RECOGNITION" />

    <application
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <service android:name=".ActivityRecognizedService" />
    </application>

</manifest>
```

Requesting Activity Recognition



- In **MainActivity.java**, To connect to Google Play Services:
  - Provide **GoogleApiClient** variable type + implement callbacks

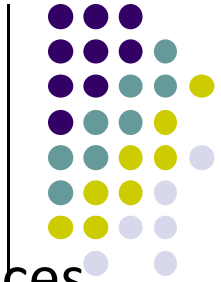
```
01 public class MainActivity extends AppCompatActivity implements GoogleApiClient.ConnectionCallbacks,
02 GoogleApiClient.OnConnectionFailedListener {
03
04     public GoogleApiClient mApiClient;
05
06     @Override
07     protected void onCreate(Bundle savedInstanceState) {
08         super.onCreate(savedInstanceState);
09         setContentView(R.layout.activity_main);
10     }
11
12     @Override
13     public void onConnected(@Nullable Bundle bundle) {
14
15     }
16
17     @Override
18     public void onConnectionSuspended(int i) {
19
20     }
21
22     @Override
23     public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {
24
25     }
}
```

Handle to Google Activity Recognition client

Called if sensor (accelerometer) connection

Called if Google Play connection fails

Requesting Activity Recognition



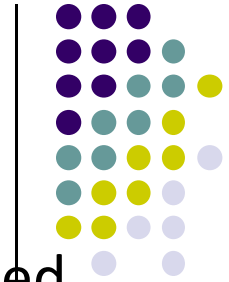
- In onCreate, initialize client and connect to Google Play Services

```
01  @Override
02  protected void onCreate(Bundle savedInstanceState) {
03      super.onCreate(savedInstanceState);
04      setContentView(R.layout.activity_main);
05
06      mApiClient = new GoogleApiClient.Builder(this)
07              .addApi(ActivityRecognition.API)
08              .addConnectionCallbacks(this)
09              .addOnConnectionFailedListener(this)
10              .build();
11
12      mApiClient.connect();
13  }
```

Request ActivityRecognition.API

Associate listeners with our instance of GoogleApiClient

Requesting Activity Recognition



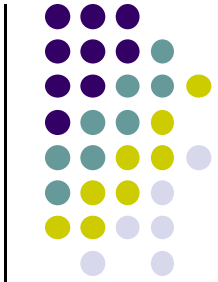
- Once **GoogleApiClient** has connected, **onConnected( )** is called
- Need to create a **PendingIntent** that goes to our **IntentService**
- Also set how often API shold check user's activity in milliseconds

```
1@Override
2public void onConnected(@Nullable Bundle bundle) {
3    Intent intent = new Intent( this, ActivityRecognizedService.class );
4    PendingIntent pendingIntent = PendingIntent.getService( this, 0, intent, PendingIntent.FLAG_UPDATE_CURRENT );
5    ActivityRecognition.ActivityRecognitionApi.requestActivityUpdates( mApiClient, 3000, pendingIntent );
6}
```

Build intent to send to IntentService

How often to check user's activity (in milliseconds)

Handling Activity Recognition



- Our app tries to recognize the user's activity every 3 seconds
- **onHandleIntent** called every 3 seconds, Intent delivered
- In **onHandleIntent( )** method of **ActivityRecognizedService**
  - Validate that received intent contains activity recognition data
  - If so, extract **ActivityRecognitionResult** from the Intent
  - Retrieve list of possible activities by calling **getProbableActivities( )** on **ActivityRecognitionResult** object

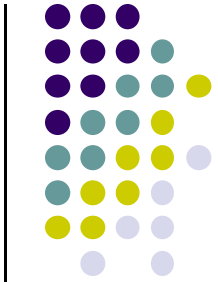
```
1@Override
2protected void onHandleIntent(Intent intent) {
3   if(ActivityRecognitionResult.hasResult(intent)) {
4      ActivityRecognitionResult result = ActivityRecognitionResult.extractResult(intent);
5      handleDetectedActivities( result.getProbableActivities() );
6   }
7}
```

Called to deliver user's activity as an Intent

Extract Activity Recognition object from Intent

Get list of probable activities

Handling Activity Recognition



- Simply log each detected activity and display how confident Google Play services is that user is performing this activity

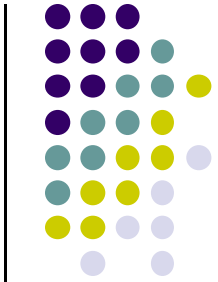
```
private void handleDetectedActivities(List<DetectedActivity> probableActivities) {
    for( DetectedActivity activity : probableActivities ) {
        switch( activity.getType() ) {
            case DetectedActivity.IN_VEHICLE: {
                Log.e( "ActivityRecogition", "In Vehicle: " + activity.getConfidence() );
                break;
            }
            case DetectedActivity.ON_BICYCLE: {
                Log.e( "ActivityRecogition", "On Bicycle: " + activity.getConfidence() );
                break;
            }
            case DetectedActivity.ON_FOOT: {
                Log.e( "ActivityRecogition", "On Foot: " + activity.getConfidence() );
                break;
            }
            case DetectedActivity.RUNNING: {
                Log.e( "ActivityRecogition", "Running: " + activity.getConfidence() );
                break;
            }
            case DetectedActivity.STILL: {
                Log.e( "ActivityRecogition", "Still: " + activity.getConfidence() );
                break;
            }
            case DetectedActivity.TILTING: {
                Log.e( "ActivityRecogition", "Tilting: " + activity.getConfidence() );
                break;
            }
```

Switch statement on activity type

Sample output

```
1  E/ActivityRecogition: On Foot: 92
2  E/ActivityRecogition: Running: 87
3  E/ActivityRecogition: On Bicycle: 8
4  E/ActivityRecogition: Walking: 5
```

Handling Activity Recognition



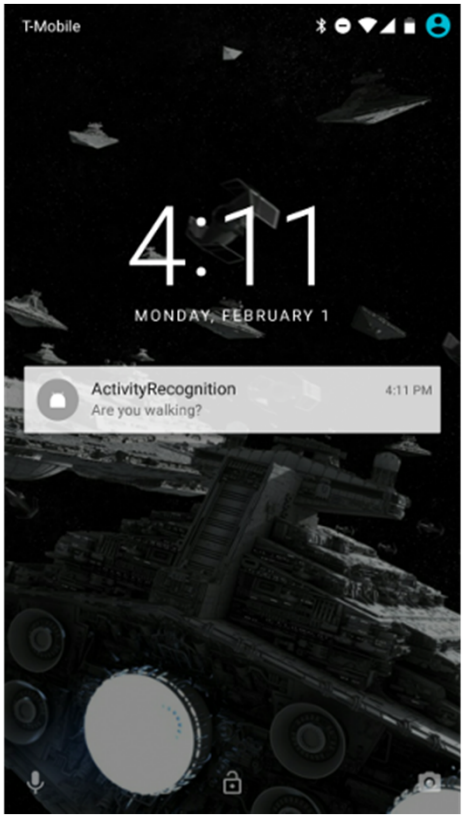
- If confidence is > 75, activity detection is probably accurate
- If user is walking, ask “Are you walking?”

```
            case DetectedActivity.WALKING: {
                Log.e( "ActivityRecogition", "Walking: " + activity.getConfidence() );
                if( activity.getConfidence() >= 75 ) {
                    NotificationCompat.Builder builder = new NotificationCompat.Builder(this);
                    builder.setContentText( "Are you walking?" );
                    builder.setSmallIcon( R.mipmap.ic_launcher );
                    builder.setContentTitle( getString( R.string.app_name ) );
                    NotificationManagerCompat.from(this).notify(0, builder.build());
                }
                break;
            }
            case DetectedActivity.UNKNOWN: {
                Log.e( "ActivityRecogition", "Unknown: " + activity.getConfidence() );
                break;
            }
        }
    }
}
```

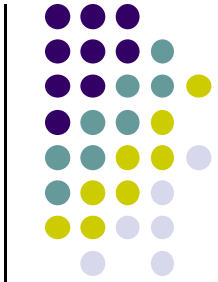
Sample Output of Program

- Sample displayed on development console

```
E/ActivityRecogition: On Foot: 92
E/ActivityRecogition: Running: 87
E/ActivityRecogition: On Bicycle: 8
E/ActivityRecogition: Walking: 5
```



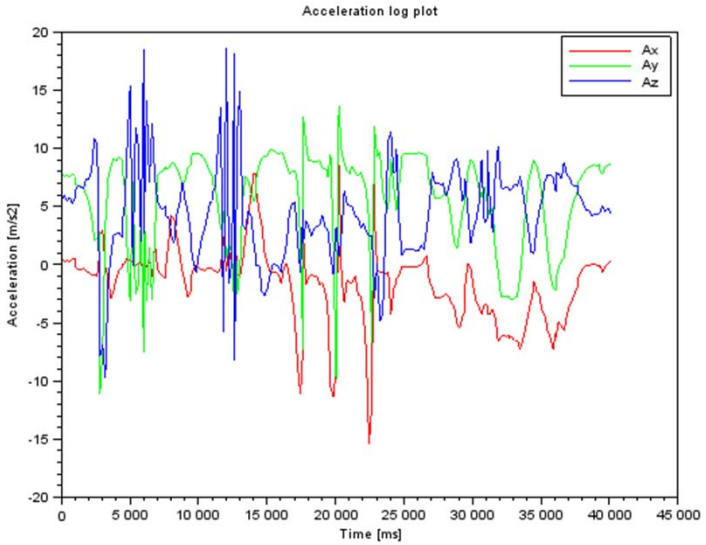
- Full code at: https://github.com/tutsplus/Android-ActivityRecognition



How Activity Recognition Works

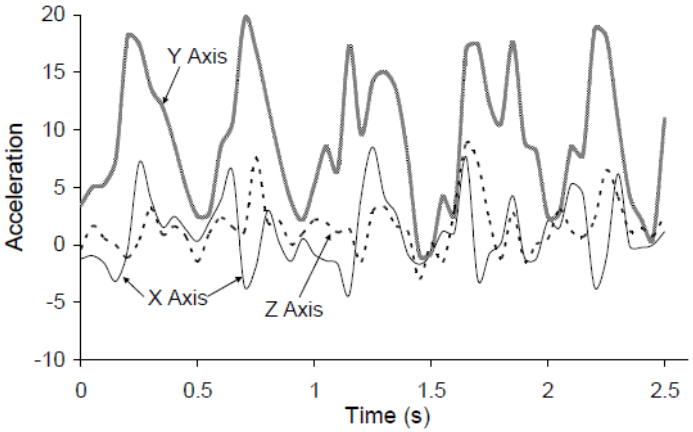
Activity Recognition

- **Goal:** Want our app to detect what activity the user is doing?
- **Classification task:** which of these 6 activities is user doing?
  - Walking,
  - Jogging,
  - Ascending stairs,
  - Descending stairs,
  - Sitting,
  - Standing

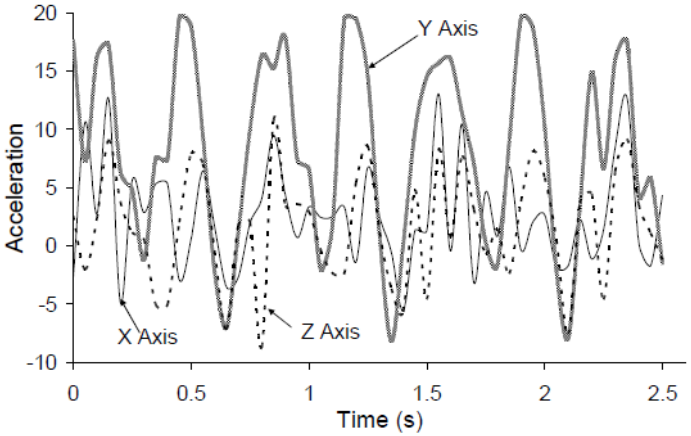


- Typically, use machine learning classifers to classify user's accelerometer signals

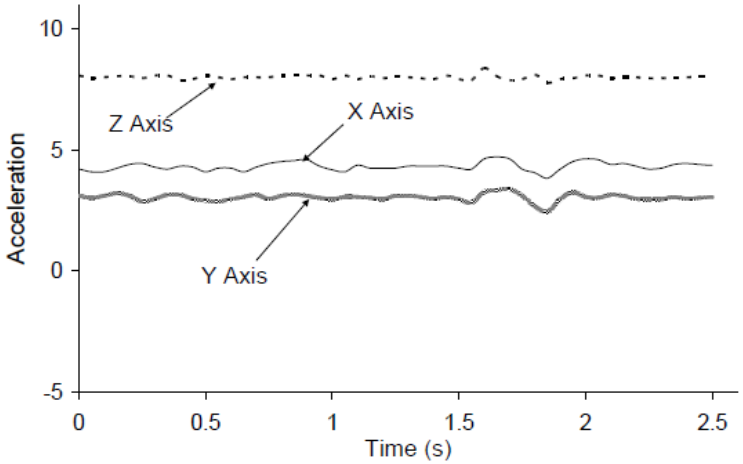
Example Accelerometer Data for Activities



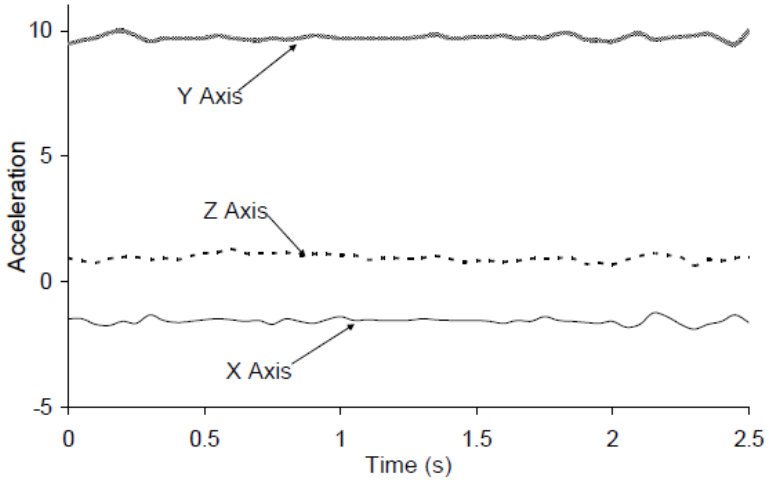
(a) Walking



(b) Jogging

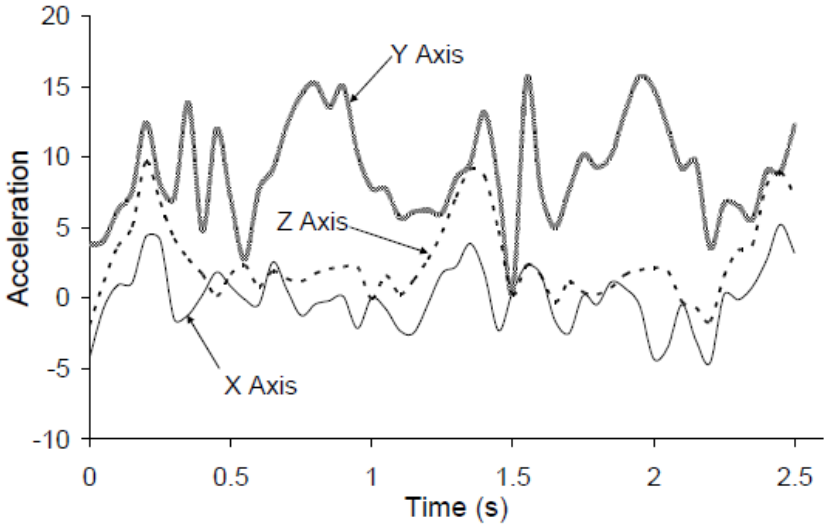


(e) Sitting

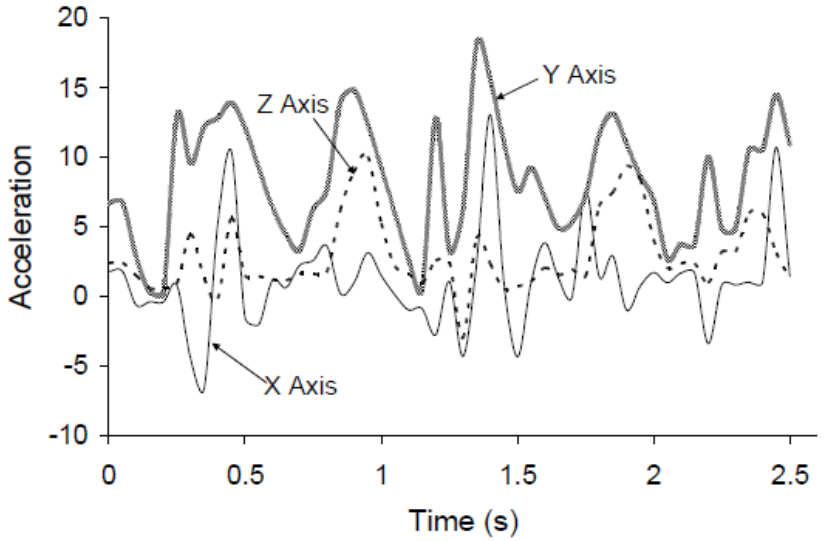


(f) Standing

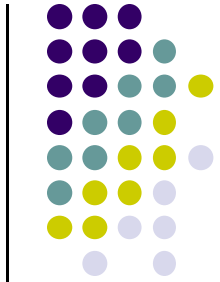
Example Accelerometer Data for Activities



(c) Ascending Stairs

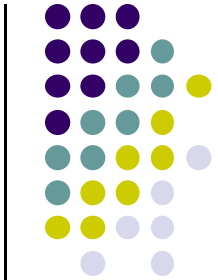


(d) Descending Stairs

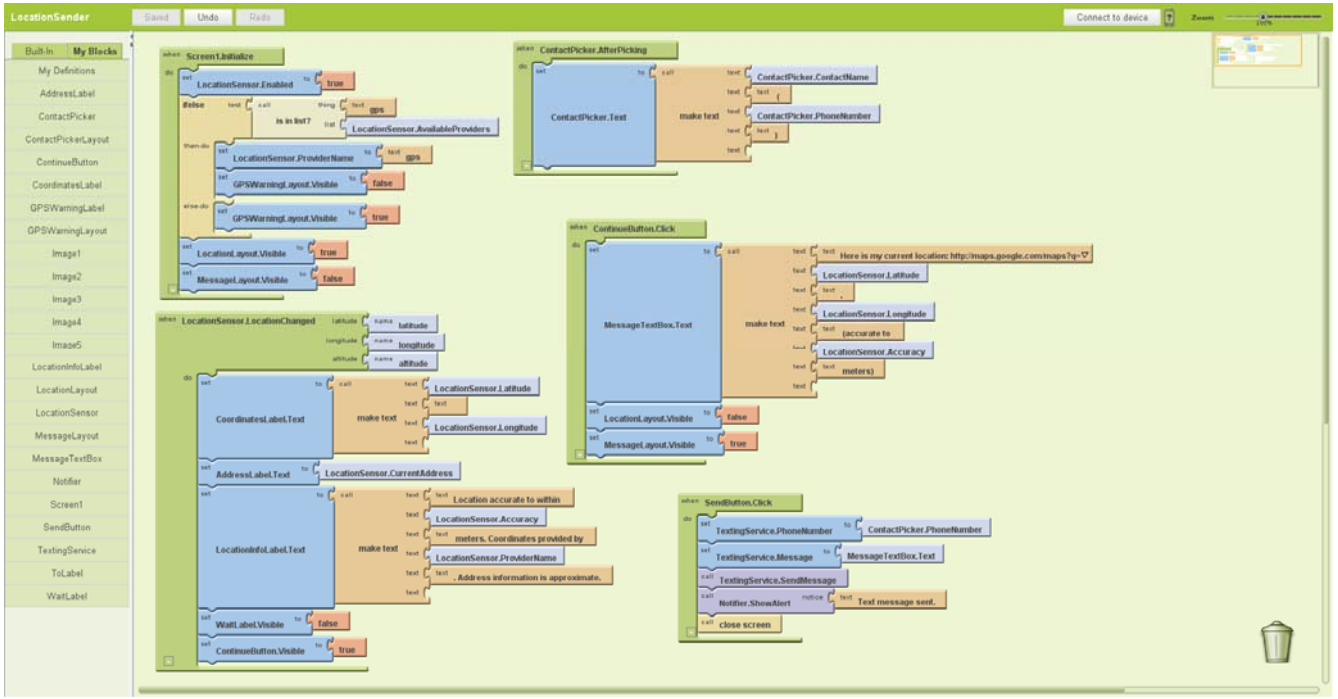


Alternate Implementation Options

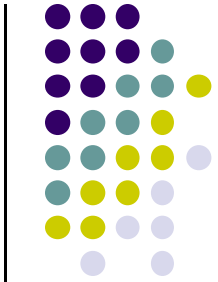
AppInventor (http://appinventor.mit.edu/)



- MIT project, previously Google
- Use lego blocks to build app, easy to learn
- **Pro:** Quick UI development
- **Con:** sensor access, use third party modules restricted

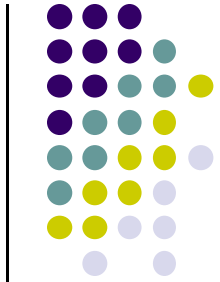


PhoneGap



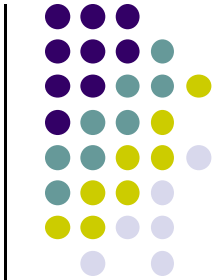
- Develop Apps using HTML, CSS, javascript
- **Pro:** Access to most native APIs, sensors, UI
- **Con:** Need to know HTML, CSS javascript

More?



- Multi-platform development tools
- iOS?

References



- Head First Android
- Android Nerd Ranch, 2nd edition
- Busy Coder's guide to Android version 6.3
- CS 65/165 slides, Dartmouth College, Spring 2014
- CS 371M slides, U of Texas Austin, Spring 2014