Android²

Bob Lee & Eric Burke

Square, Inc.

Square


Pay Eric Burke
$25.00
VISA 1111
Sign here.
Cancel
I agree to pay the above total amount according to my card issuer agreement.
Continue

Overview

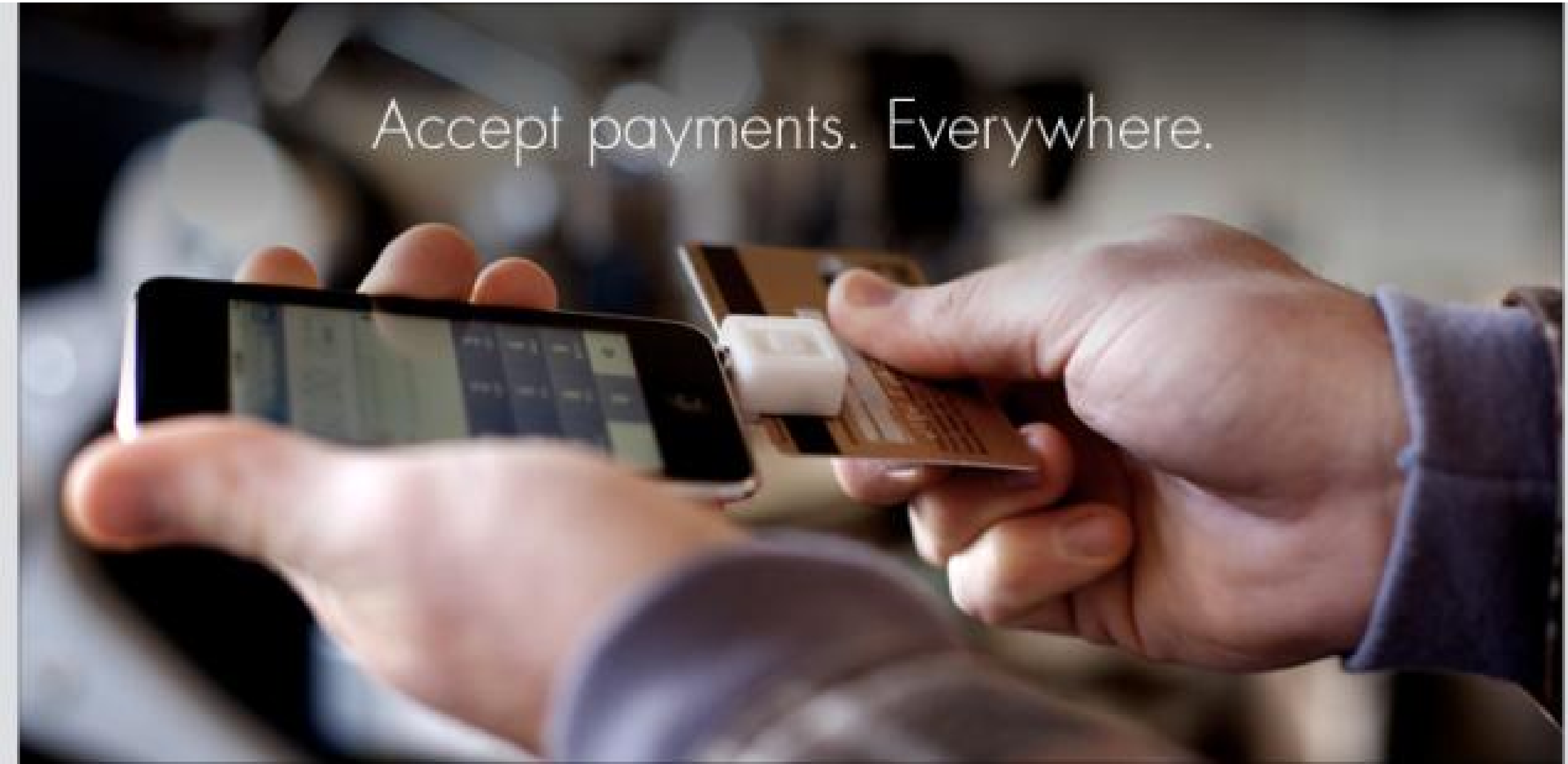
> Squarewave

> Retrofit

- I/O
- Shake detection
- REST

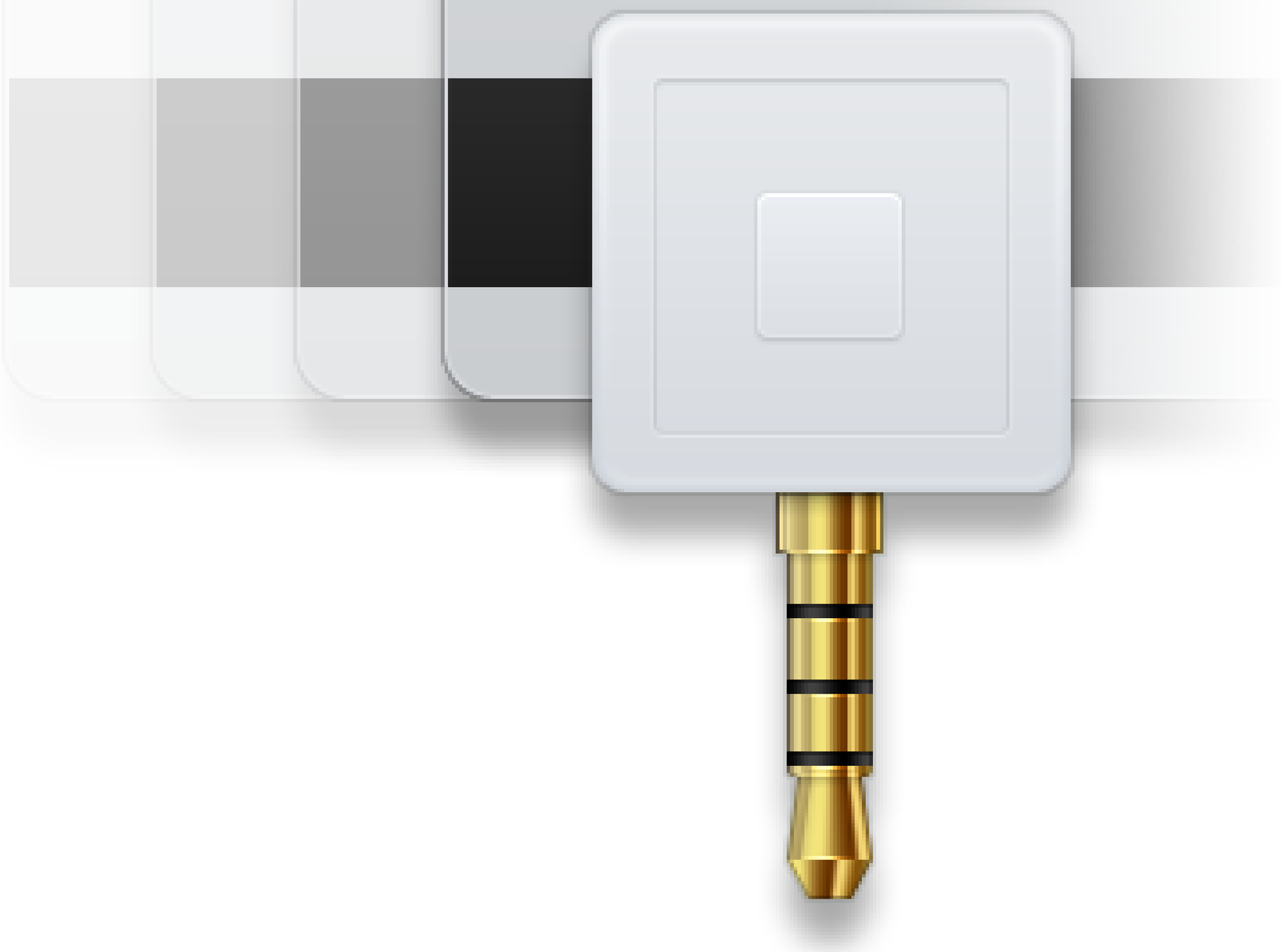
> Point-of-sale API

Squarewave: Magnetic Stripe Decoder

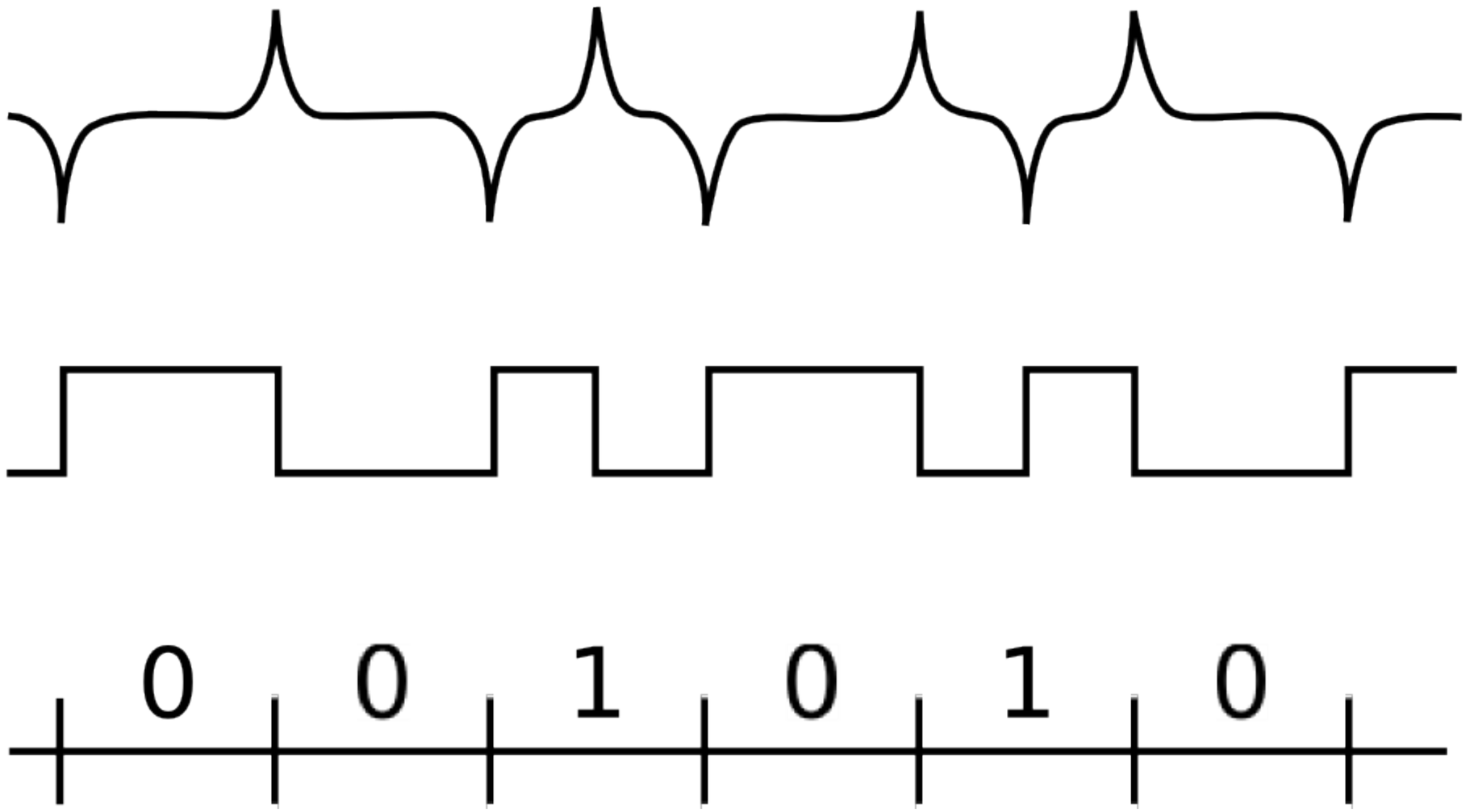


Square Reader

Acts as a microphone

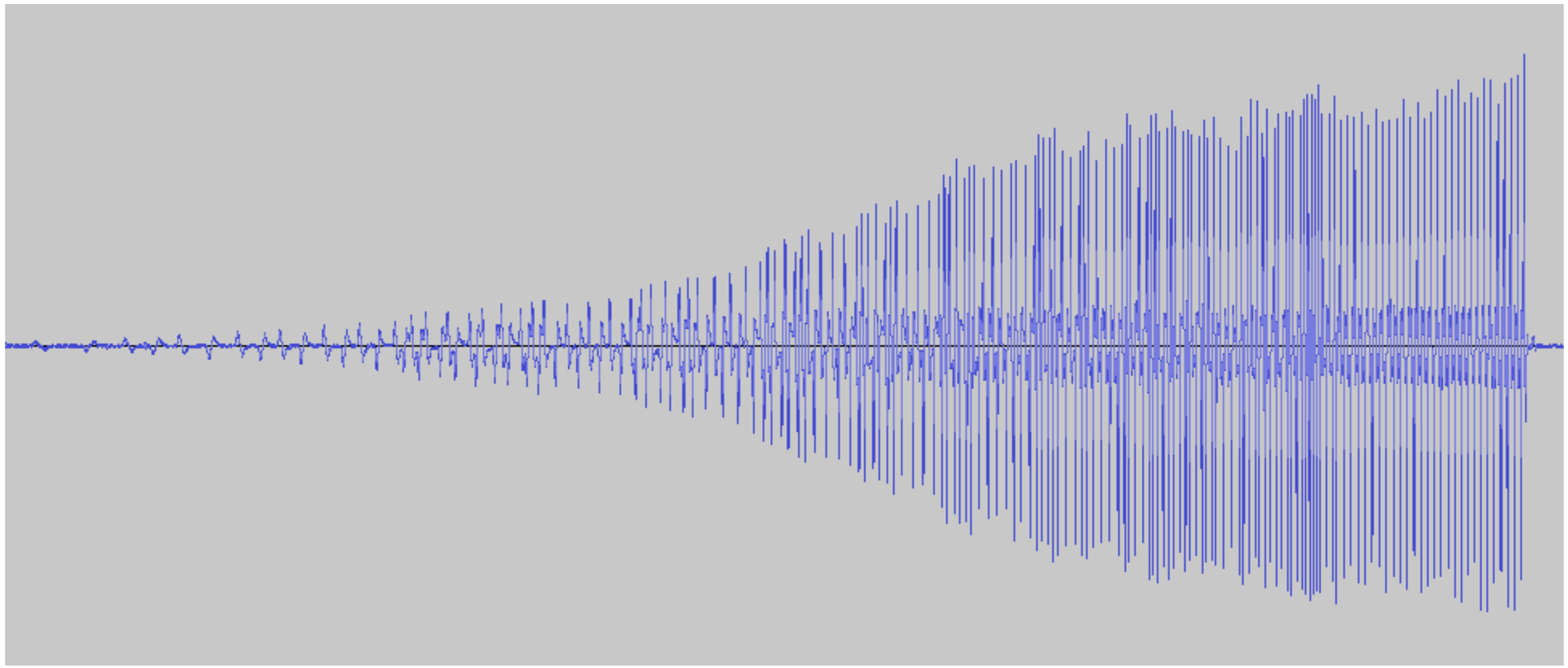


Ideal Waveform

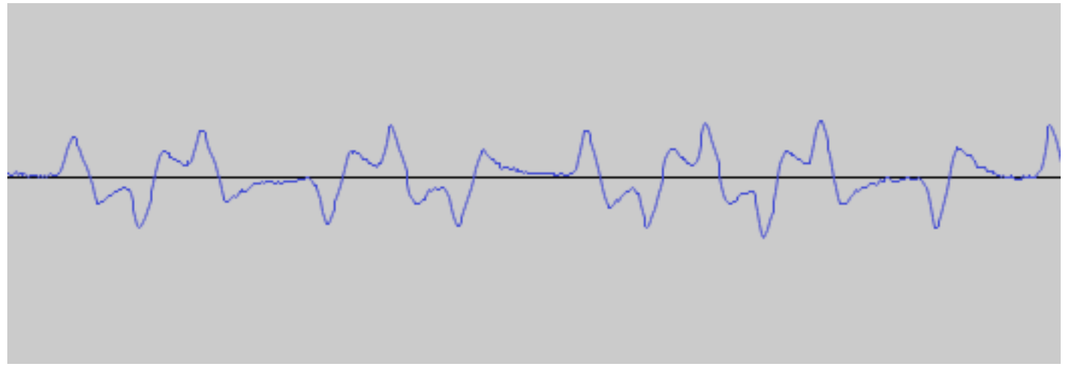



Sprint
11:27 AM
Charge
Doctor Evil
Settings
$10.00
Enter an optional note
Clear
9
8
6
7
5
3
4
2
1
0
00
SAMSUNG

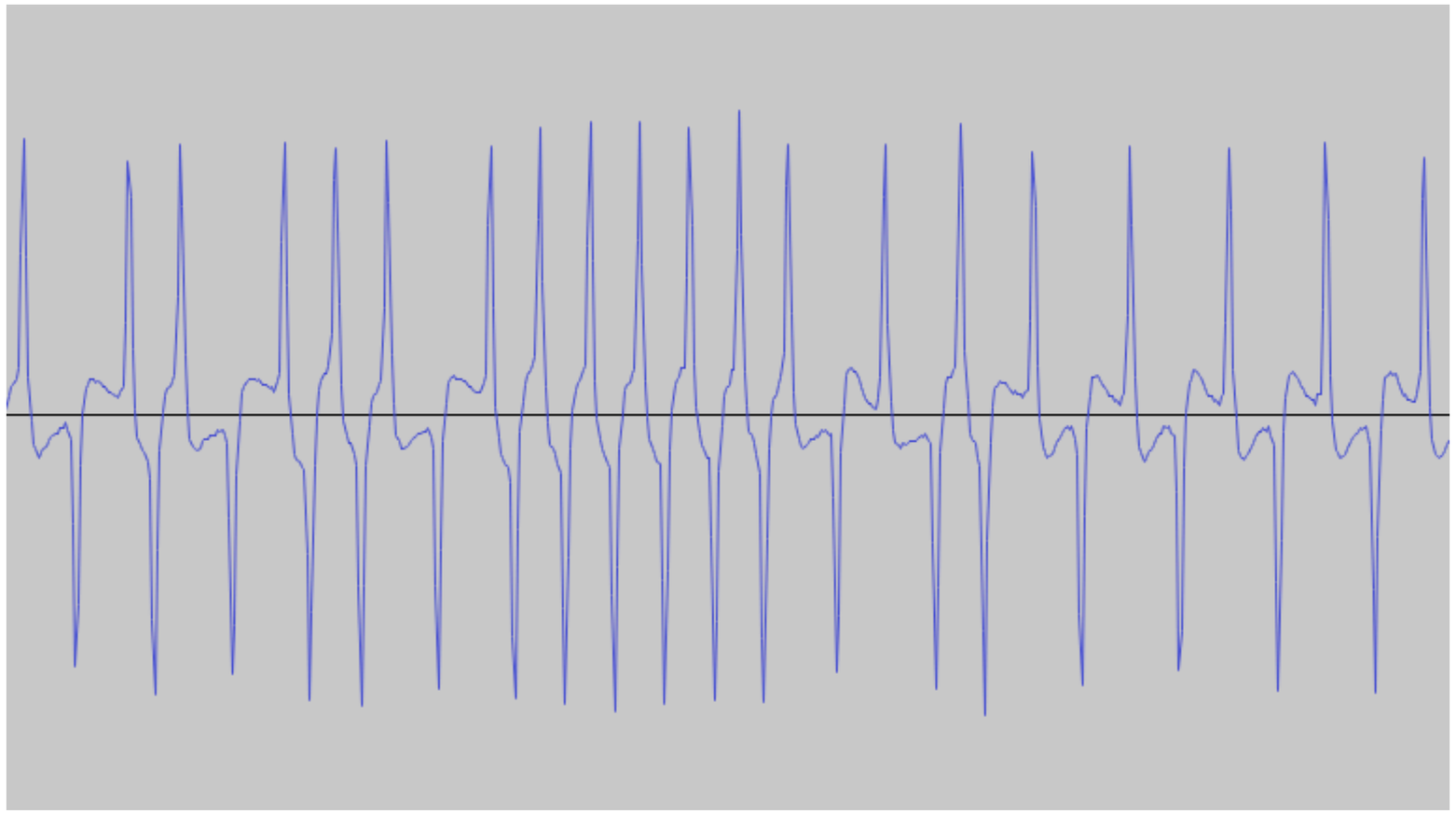
Actual Swipe Recording



Swipe Start



Swipe End



Challenges

Challenges

> Swipe speed

Challenges

> Swipe speed

> Device sample rate

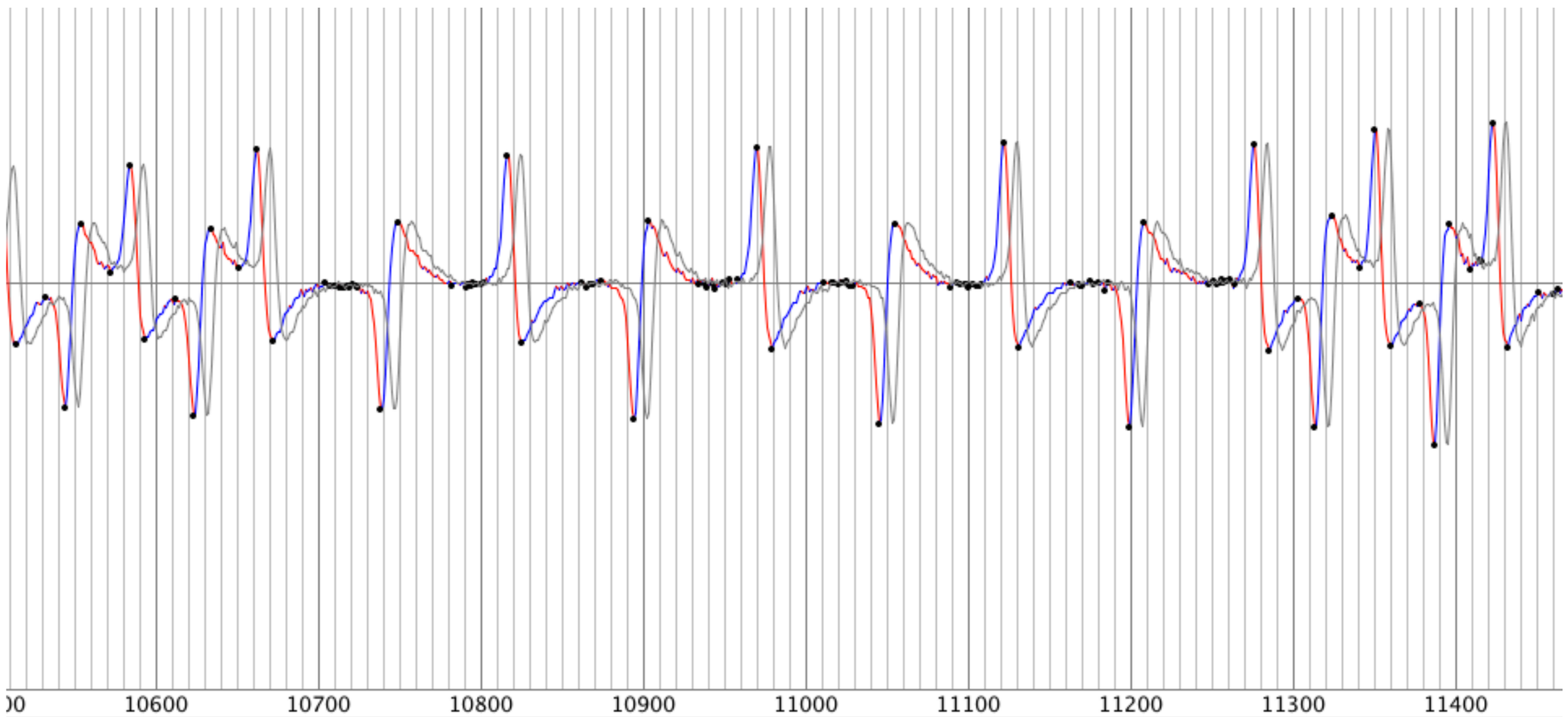
Challenges

> Swipe speed
> Device sample rate
> Audio correction

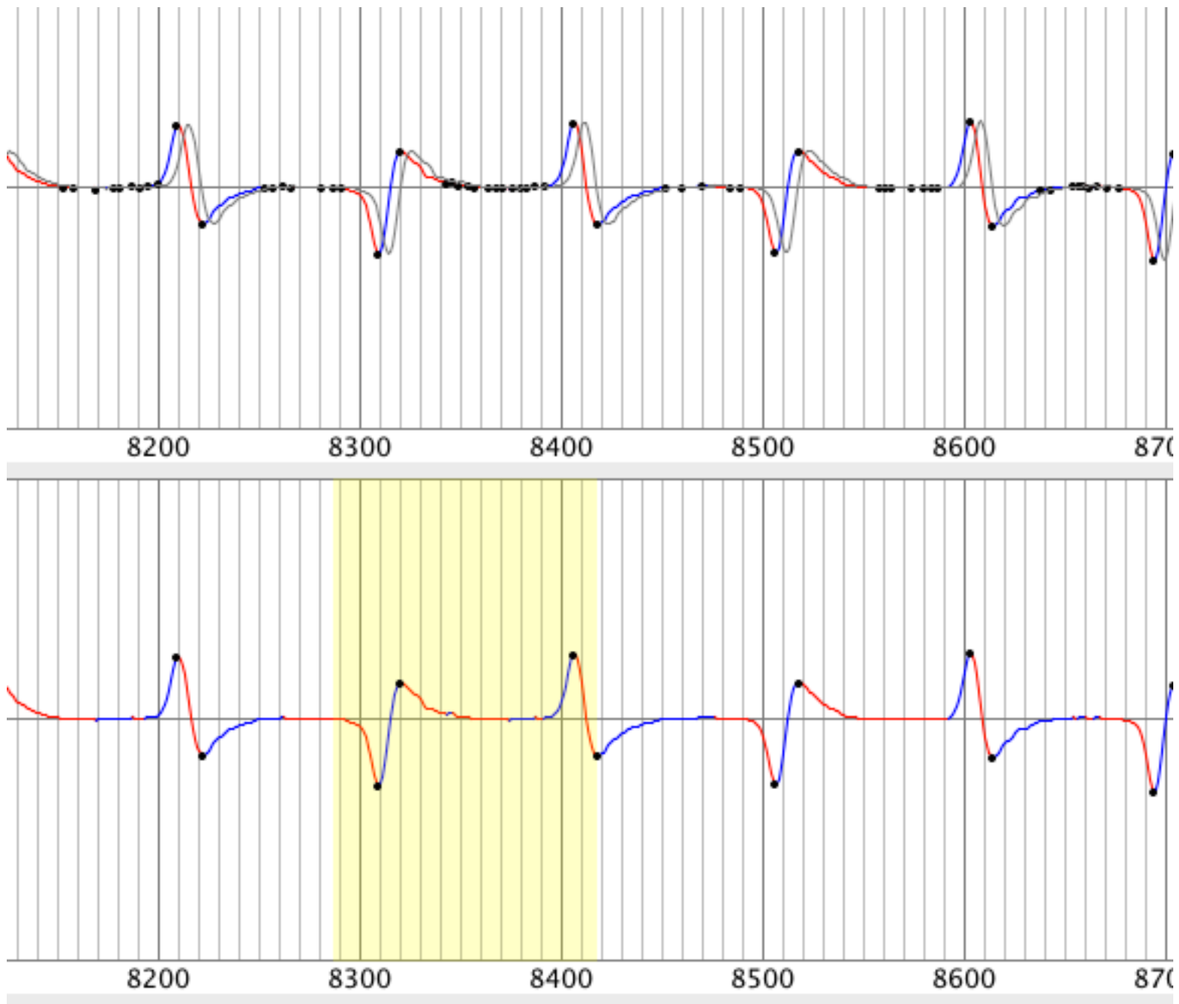
Step 1: Swipe Detection

Step 2: Denoising

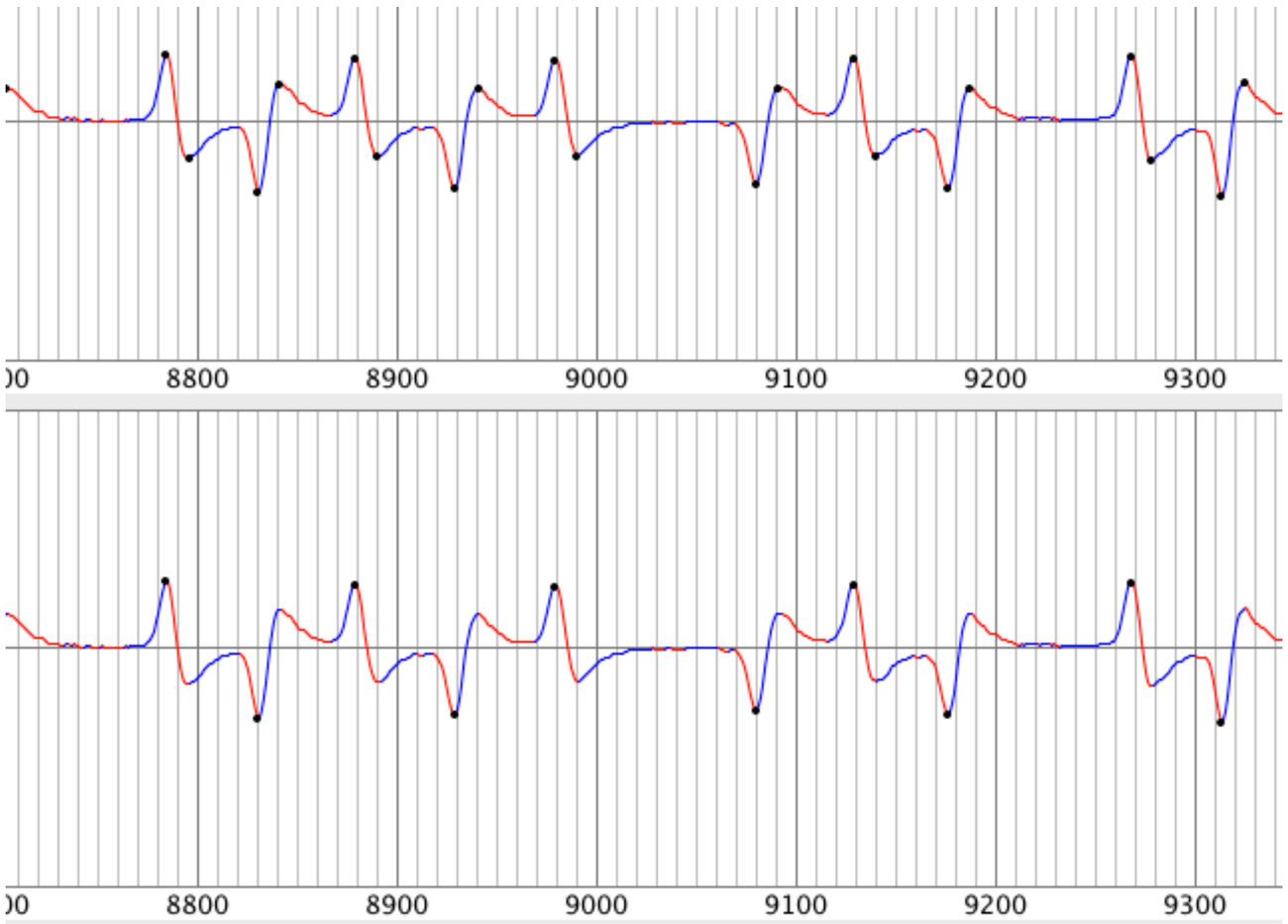
Step 3: Peak Detection



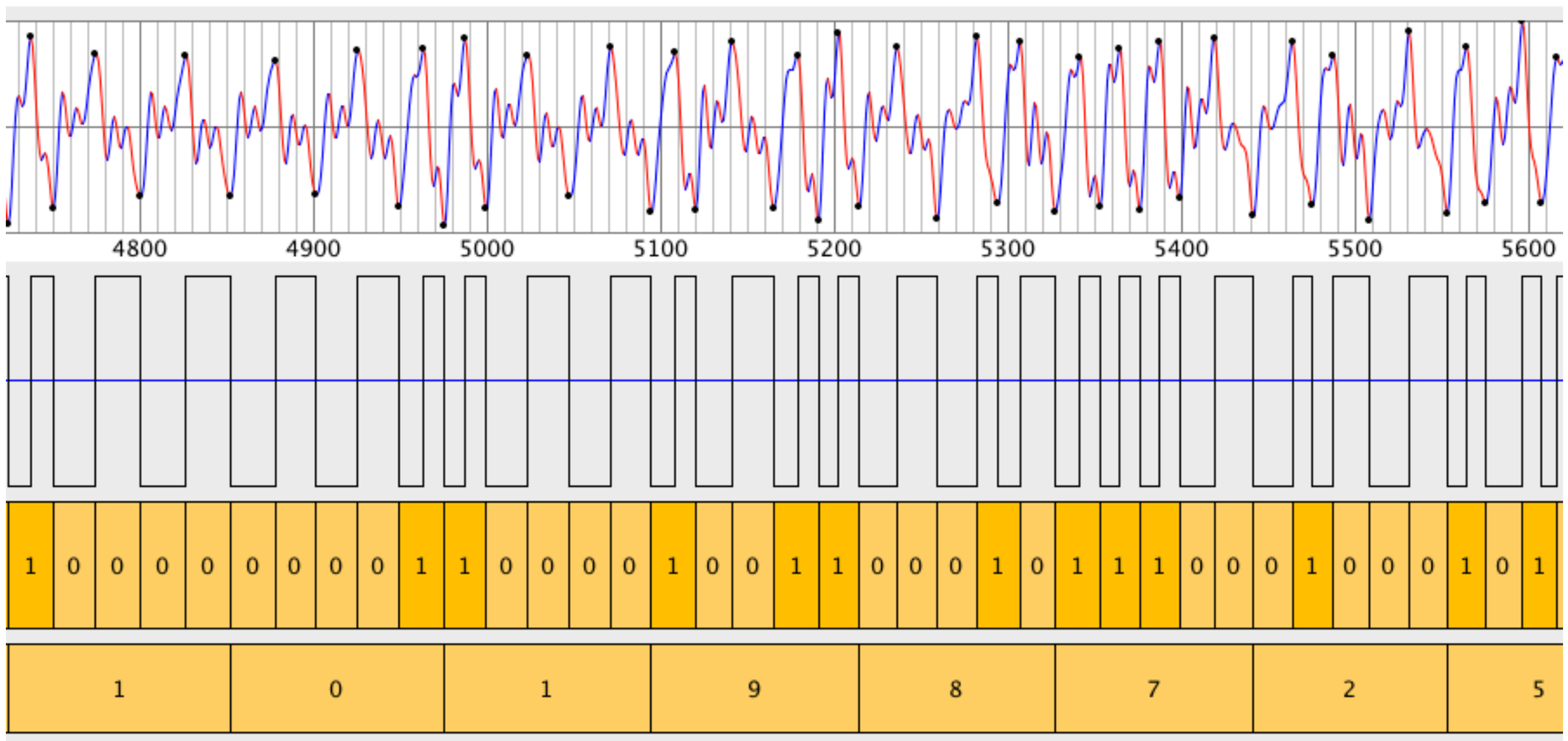
Step 4: Window Peak Removal



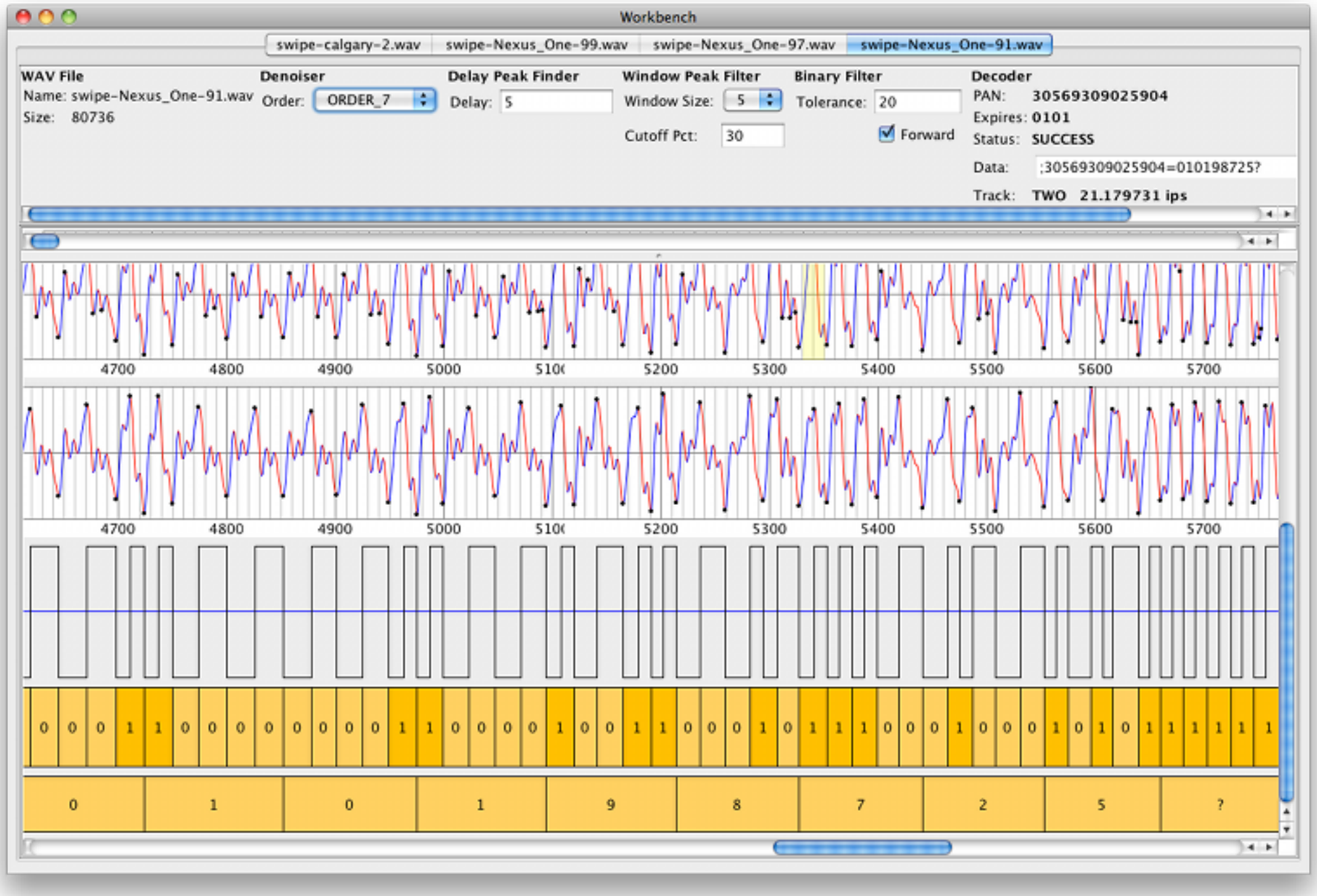
Step 5: Consecutive Peak Removal



Step 6: Decoding



Workbench


Workbench
swipe-calgary-2.wav
swipe-Nexus_One-99.wav
swipe-Nexus_One-97.wav
swipe-Nexus_One-91.wav
WAV File
Name: swipe-Nexus_One-91.wav
Size: 80736
Denoiser
Order: ORDER_7
Delay Peak Finder
Delay: 5
Window Peak Filter
Window Size: 5
Cutoff Pct: 30
Binary Filter
Tolerance: 20
Forward
Decoder
PAN: 30569309025904
Expires: 0101
Status: SUCCESS
Data: ;30569309025904=010198725?
Track: TWO 21.179731 ips
4700 4800 4900 5000 5200 5300 5400 5500 5600 5700
0 1 0 1 9 8 7 2 5 ?

95% Accuracy

95% Accuracy

> Record hundreds of swipes

95% Accuracy

> Record hundreds of swipes

> Decode all, record results

95% Accuracy

> Record hundreds of swipes

> Decode all, record results

> Adjust parameters

95% Accuracy

> Record hundreds of swipes
> Decode all, record results
> Adjust parameters
> Repeat

95% Accuracy

> Record hundreds of swipes
> Decode all, record results
> Adjust parameters
> Repeat
> After finding best options...

95% Accuracy

> Record hundreds of swipes
> Decode all, record results
> Adjust parameters
> Repeat
> After finding best options...
> Repeat entire process on failed swipes

Retrofit

> Extends Android and Java

> Apache-licensed

> Modules

- **`core`**
- **`io`**
- **`http`**
- **`android`**

> **`http://github.com/square/retrofit`**

Square for Android Persistence

> Queues
> Key-value pairs
> No SQL

Persistent Queue

> Sending data to a server

- Emails (Receipts)
- Image uploads
- Payments
- Analytics
- Crash dumps

> Histories

Traditional Approaches

> SQLite

- Operations are O(log(n))
- Rollback journal requires multiple operations
- Write ahead log has other tradeoffs
- **`xDeviceCharacteristics`**

> File-per-element

- 4k/entry minimum
- Several I/O operations

QueueFile

> All operations are O(1)

> Writes sync

> Writes are atomic

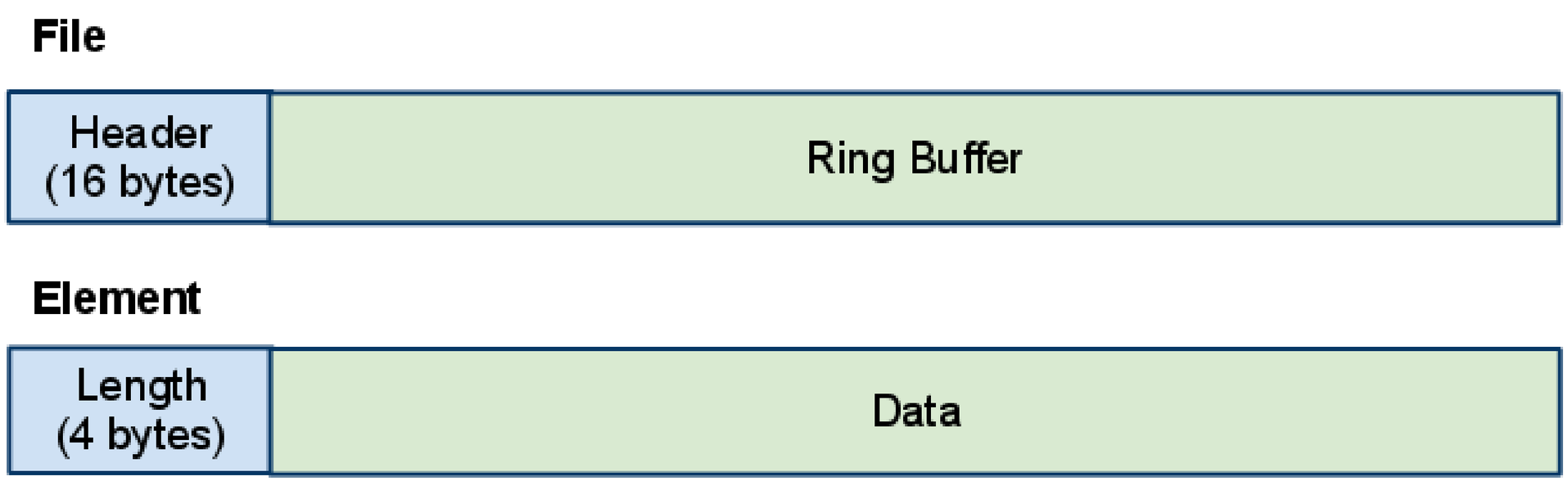
```
public class QueueFile {
  public QueueFile(File file) throws IOException { ... }

  public void add(byte[] data) throws IOException { ... }
  public void add(byte[] data, int offset, int count)
      throws IOException { ... }

  public byte[] peek() throws IOException { ... }
  public void remove() throws IOException { ... }

  public int size() { ... }
  public void clear() throws IOException { ... }
  public void close() throws IOException { ... }
}
```

The Implementation

> Depends somewhat on YAFFS

- Yet Another Flash File System
- Android's preeminent file system
- Supports atomic sector writes

> Writing the header commits a change

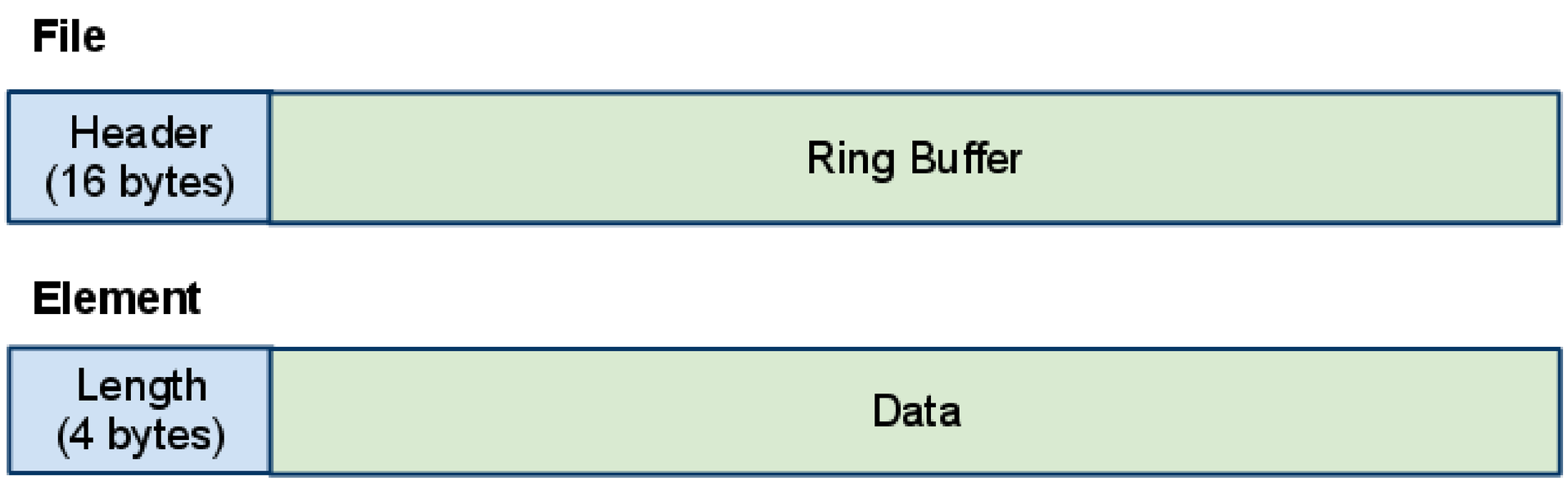


QueueFile.add()

> Write element data

> Write header (16 bytes < 4k)

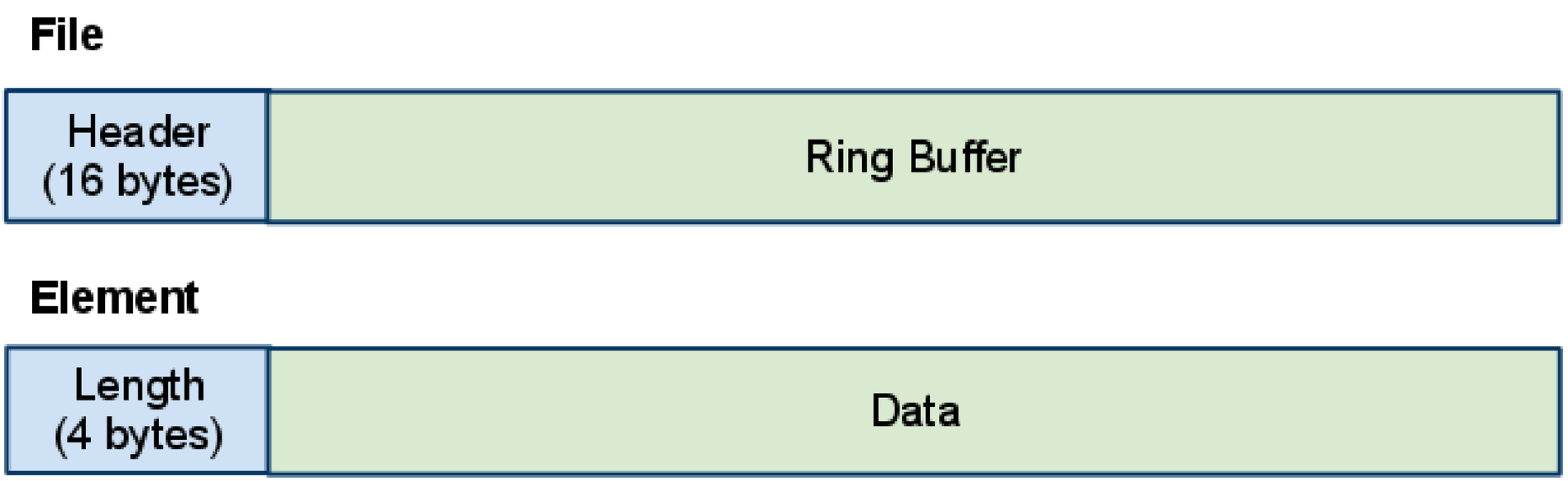
> Update in-memory state



`QueueFile.remove()`

> Write header

> Update in-memory state



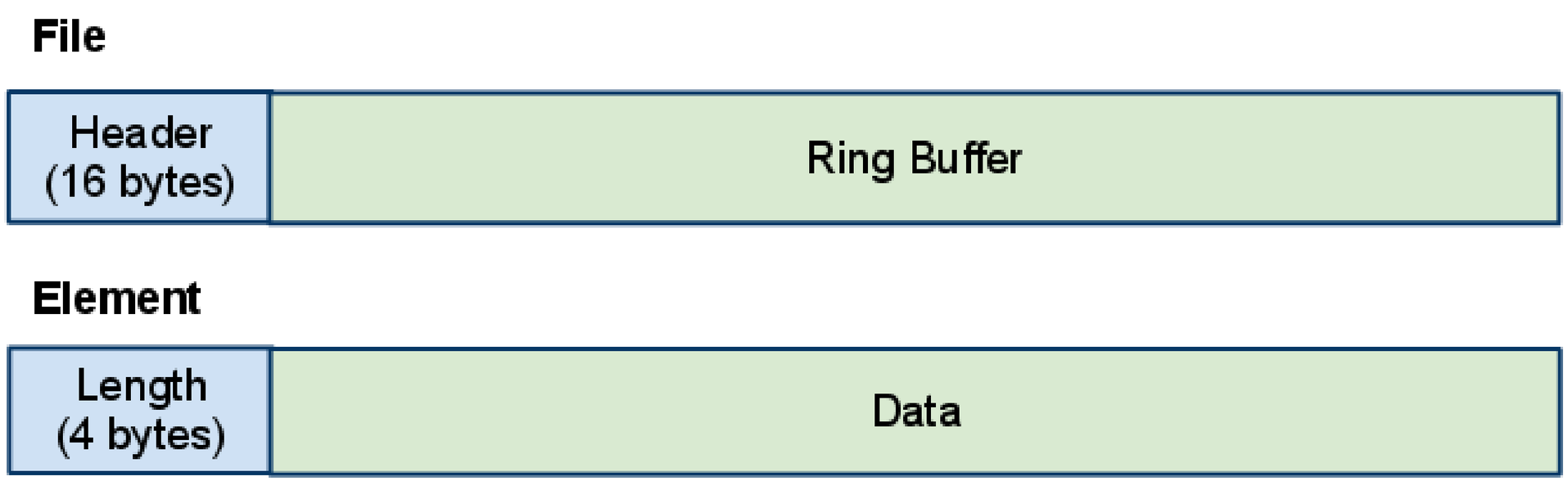
Buffer expansion

> **`file.setLength(oldLength << 1)`**

> Make ring buffer contiguous

> Write header (including file length)

> Update in-memory state



Future Features

> Support file systems without atomic segment writes

- Rollback journal

> Batch writes

- Optimistic batching
- > 3 orders of magnitude throughput


Etch A Sketch®

Shake to Clear Signature

Using the Accelerometer

```
public class HelloAccelerometer extends Activity
    implements SensorEventListener {

  @Override protected void onResume() {
    super.onResume();

    SensorManager sensorMgr = (SensorManager) getSystemService(
        Context.SENSOR_SERVICE);
  Sensor accelerometer = sensorMgr.getDefaultSensor(
        Sensor.TYPE_ACCELEROMETER);

  sensorMgr.registerListener(this, accelerometer,
        SensorManager.SENSOR_DELAY_GAME);
  }

  public void onSensorChanged(SensorEvent event) {
    float ax = event.values[0];
    float ay = event.values[1];
    float az = event.values[2];
  }
```

Accelerometer Values

Accelerometer Values

> x, y, and z acceleration

Accelerometer Values

> x, y, and z acceleration

> Units are m/s^2

Accelerometer Values

> x, y, and z acceleration

> Units are m/s^2

> Acceleration applied to device *minus force of gravity*

Accelerometer Values

> x, y, and z acceleration

> Units are m/s^2

> Acceleration applied to device *minus force of gravity*

> When flat on a table, Z acceleration = +9.81 (0 - -9.81)

Device at Rest

Magnitude (Pythagorean)

```
public class Magnitude extends Activity
    implements SensorEventListener {

  public void onSensorChanged(SensorEvent event) {
    float ax = event.values[0];
    float ay = event.values[1];
    float az = event.values[2];

    double magnitude = Math.sqrt(ax * ax + ay * ay + az * az);
  }
}
```

Magnitude Graph

Threshold

Data Rates Vary by Device

	NORMAL	UI	GAME	FASTEST
Dell Streak	47	47	47	47
LG Ally	5	5	5	5
Motorola Backflip	89	89	89	89
Samsung Epic 4G	5	10	19	91
HTC Desire	43	43	43	43

Solution: Variable Size Window

> Did the magnitude exceed the threshold?

> Store true/false readings in a queue

> Queue holds readings from last 500ms

> When 75% of readings are true, shake

ShakeDetector

```
public class ShakeDetector {
  public ShakeDetector(Listener listener) {
    ...
  }

  public void start(SensorManager sensorMgr) {
    ...
  }

  public void stop() {
    ...
  }

  /** Listens for shakes. */
  public interface Listener {
    /** Called on the main thread when the device is shaken. */
    void hearShake();
  }
}
```

Using ShakeDetector

```
public class ShakeDemo extends Activity implements
ShakeDetector.Listener {

  private ShakeDetector shakeDetector = new ShakeDetector(this);

  @Override protected void onResume() {
    super.onResume();

    SensorManager sensorMgr = (SensorManager) getSystemService(
        Context.SENSOR_SERVICE);
    shakeDetector.start(sensorMgr);
  }

  @Override protected void onPause() {
    shakeDetector.stop();
  }

  public void hearShake() {
    // The phone was shaking...
  }
```