

# Android<sup>2</sup>

Bob Lee & Eric Burke  
Square, Inc.

# Square









# Overview

> Squarewave

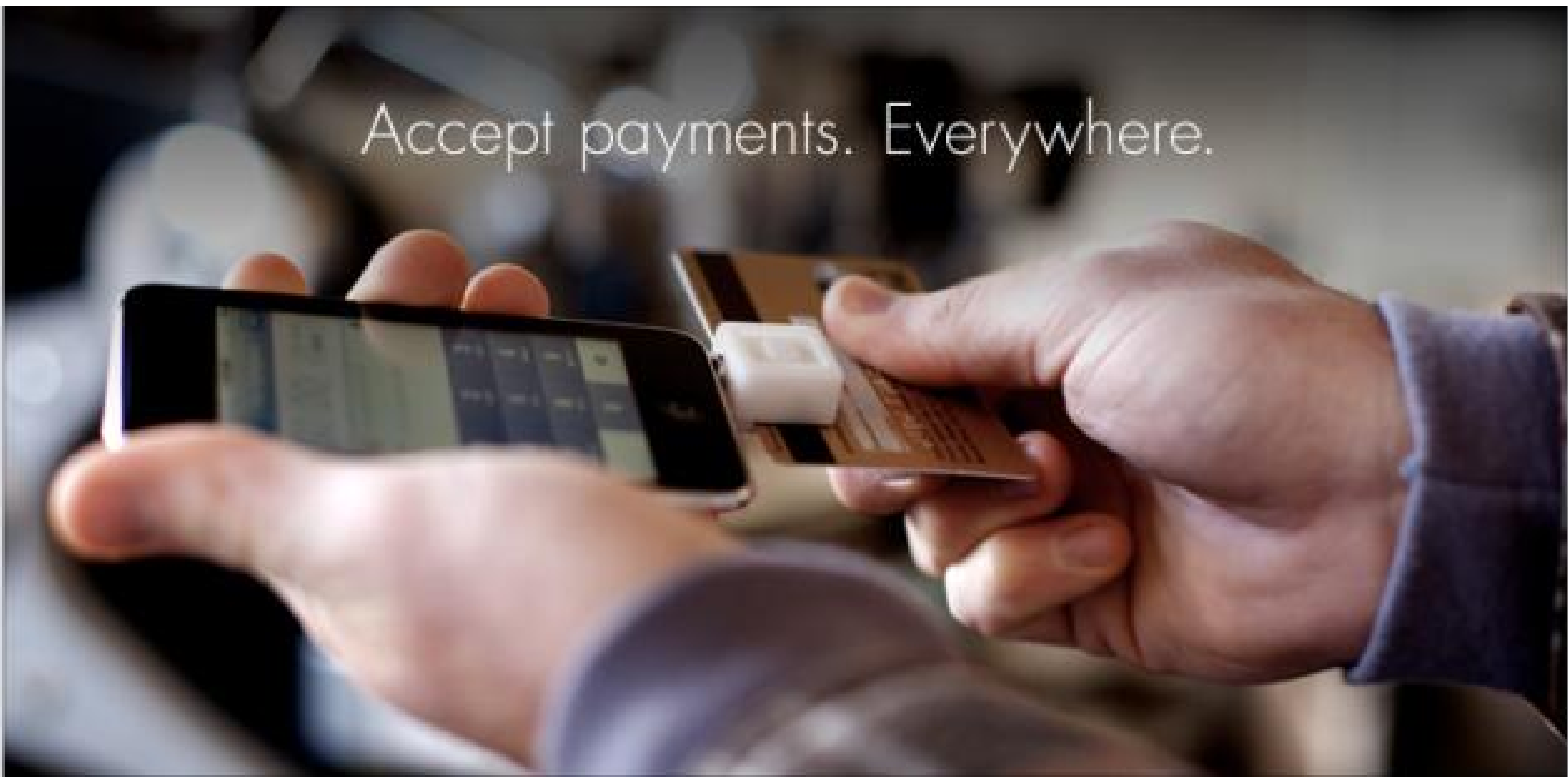
> Retrofit

- I/O
- Shake detection
- REST

> Point-of-sale API

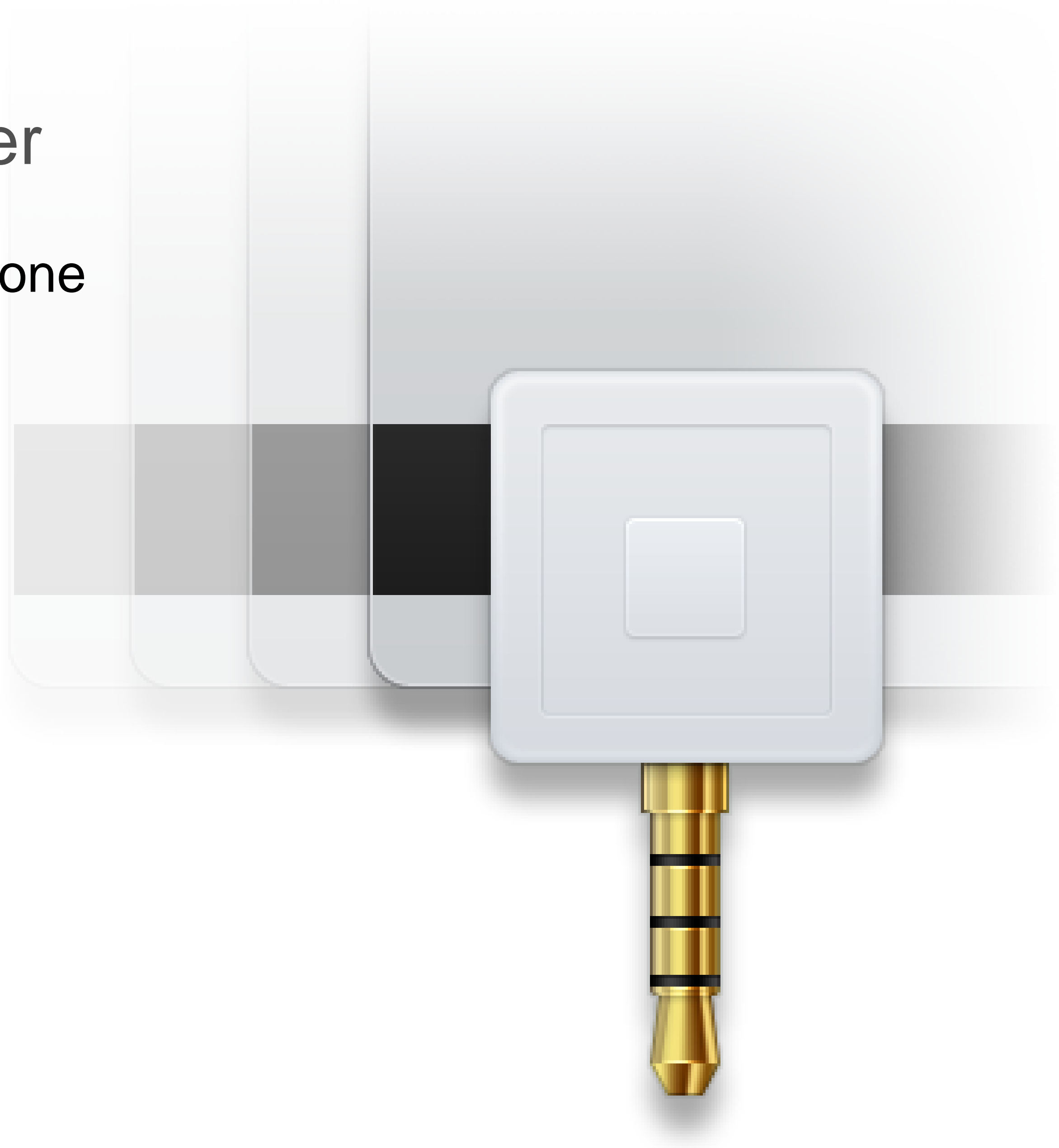
# Squarewave: Magnetic Stripe Decoder

Accept payments. Everywhere.

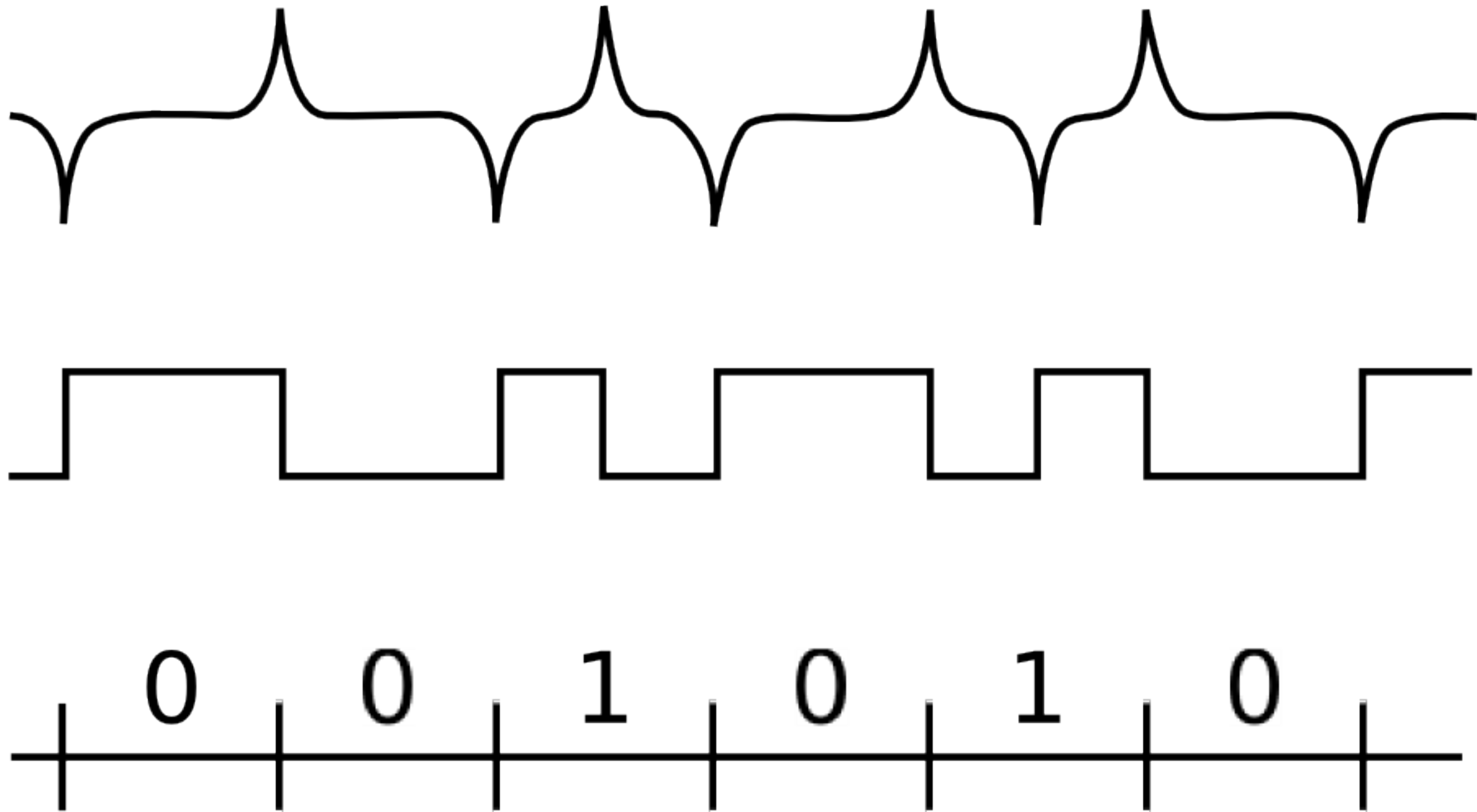


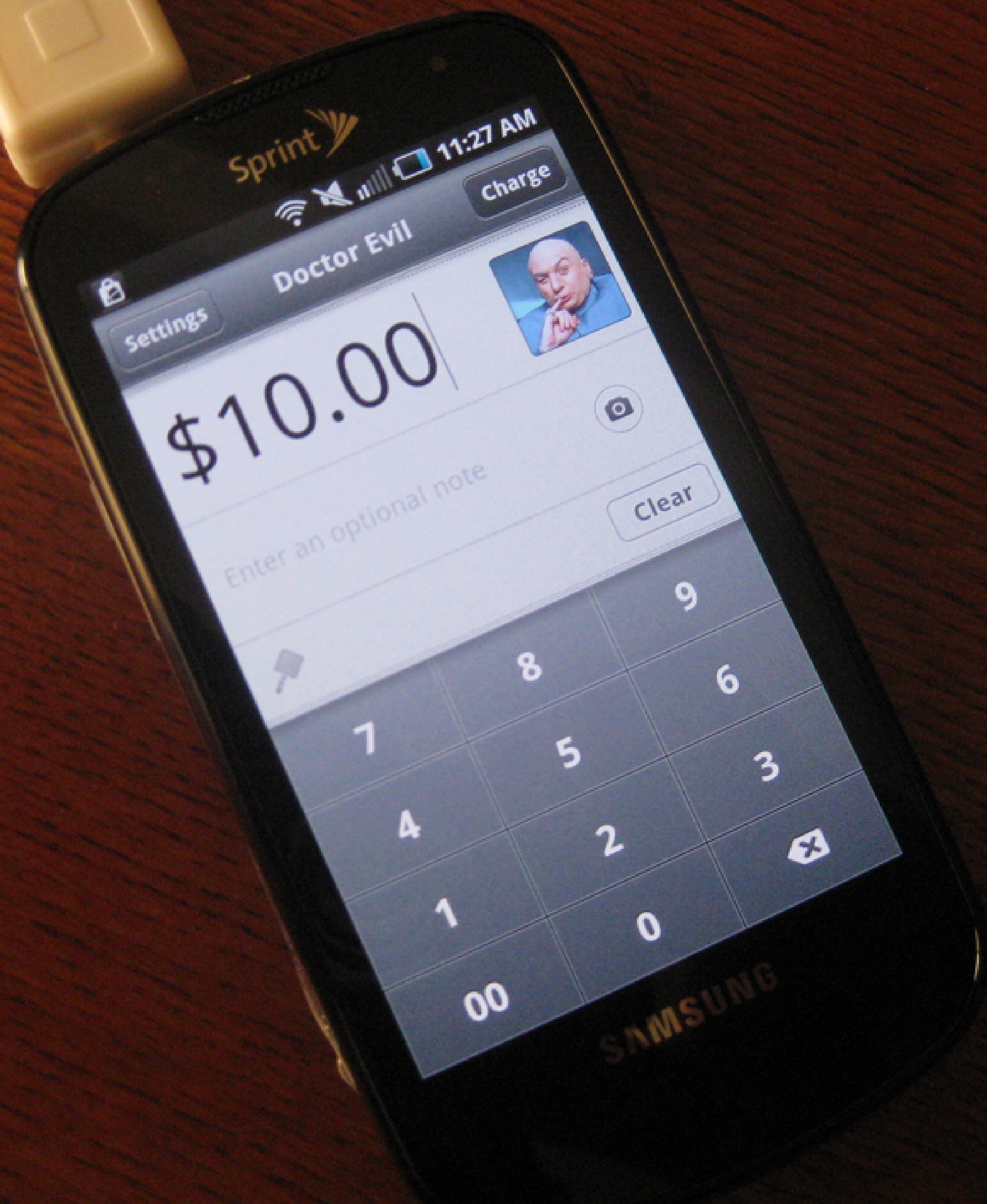
# Square Reader

Acts as a microphone



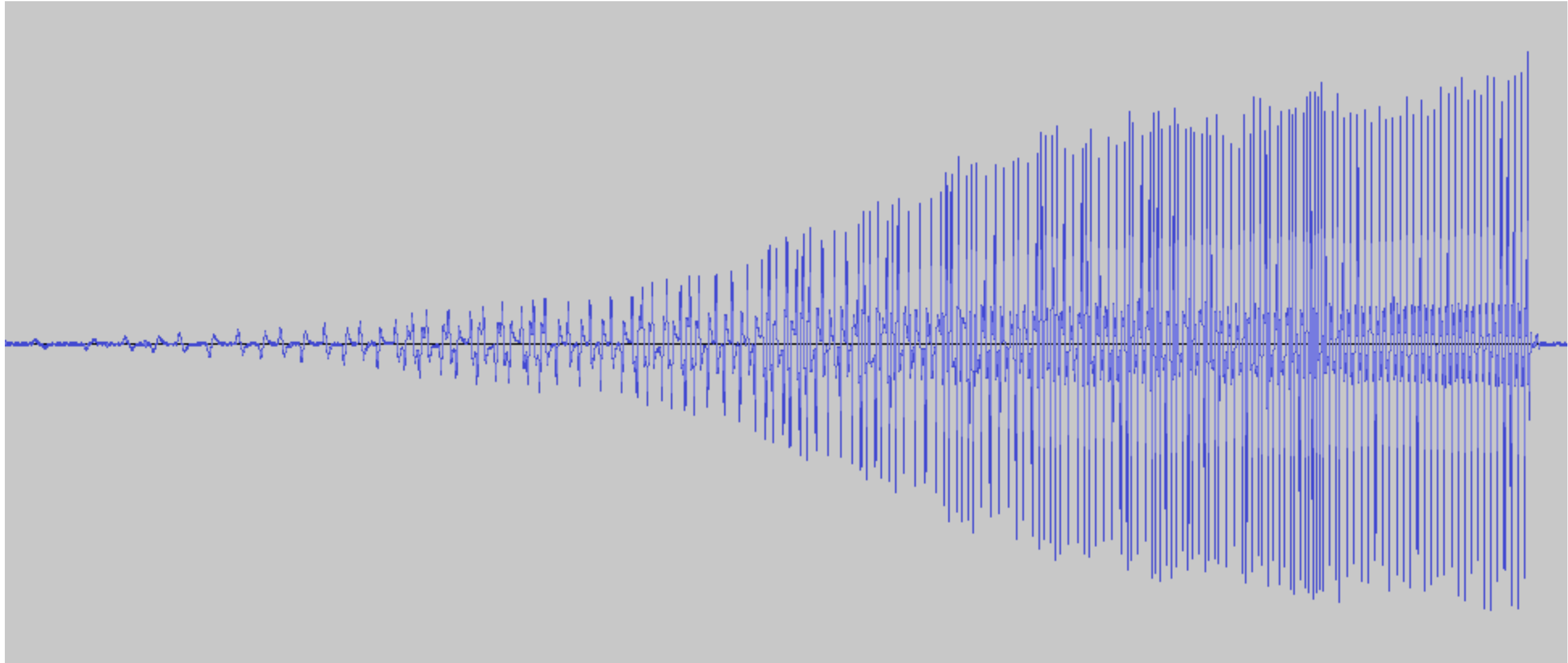
# Ideal Waveform



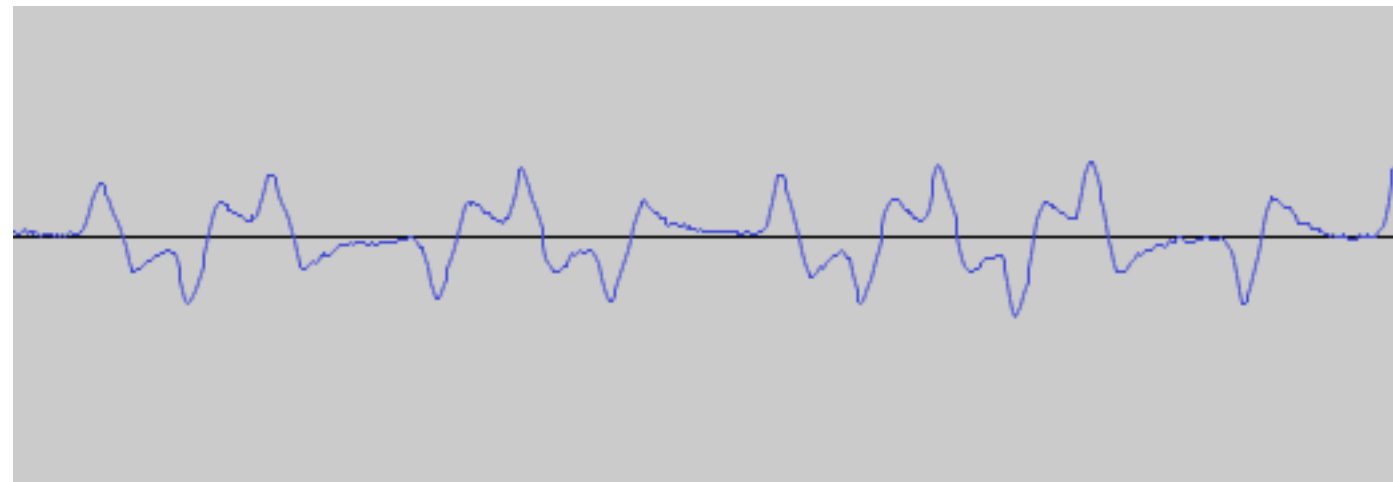




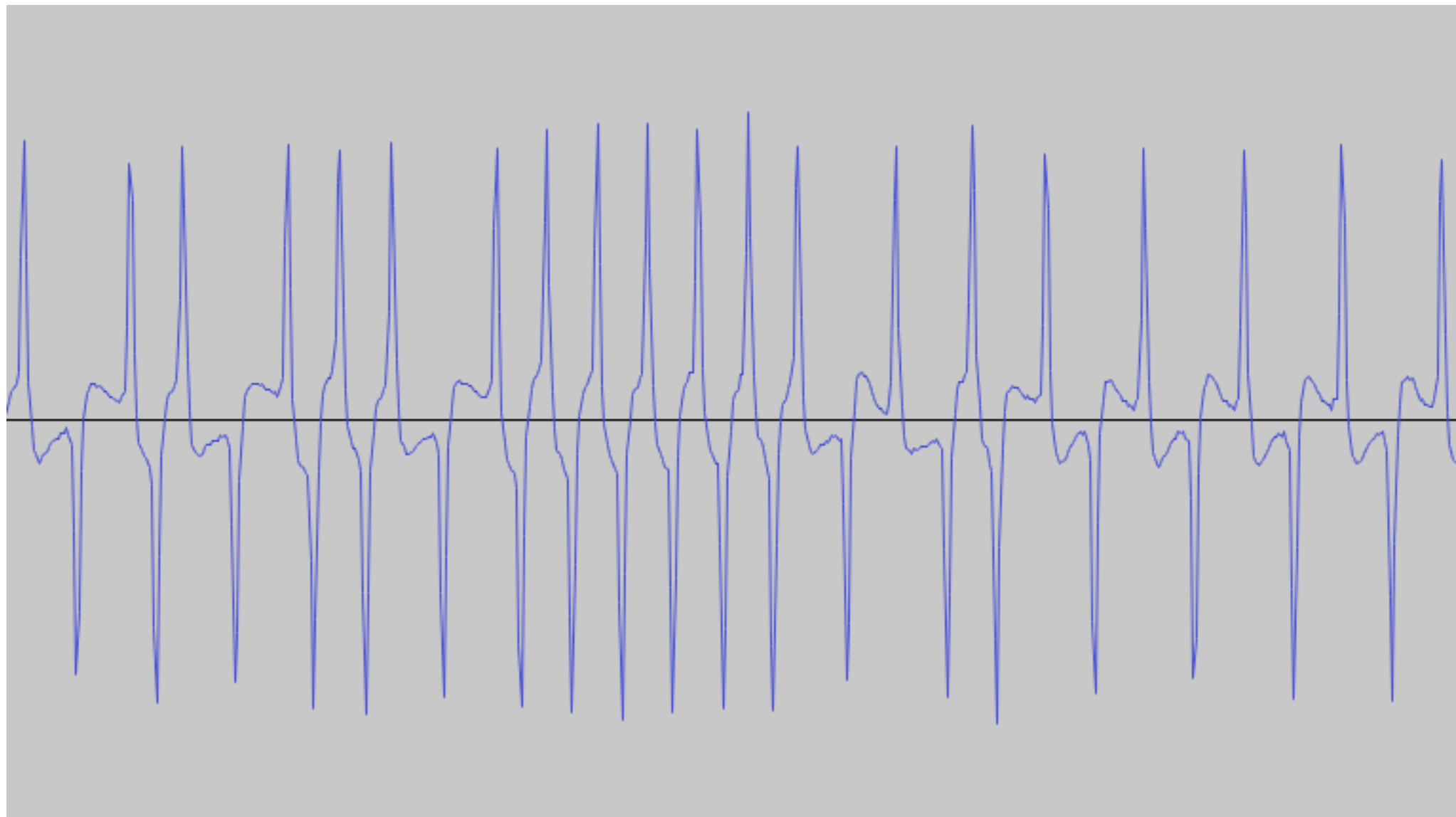
# Actual Swipe Recording



# Swipe Start



# Swipe End





# Challenges

# Challenges

> Swipe speed

# Challenges

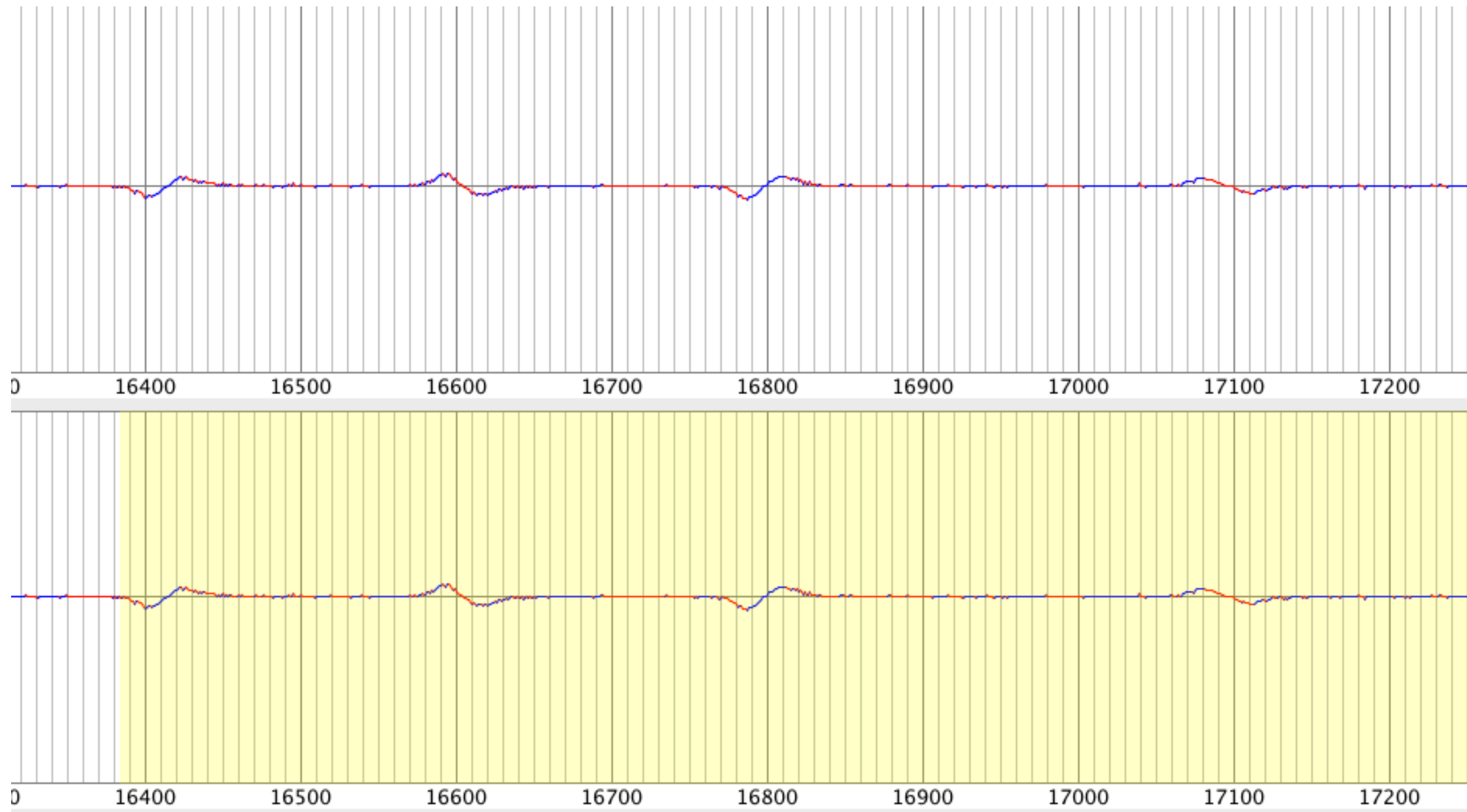
- > Swipe speed
- > Device sample rate



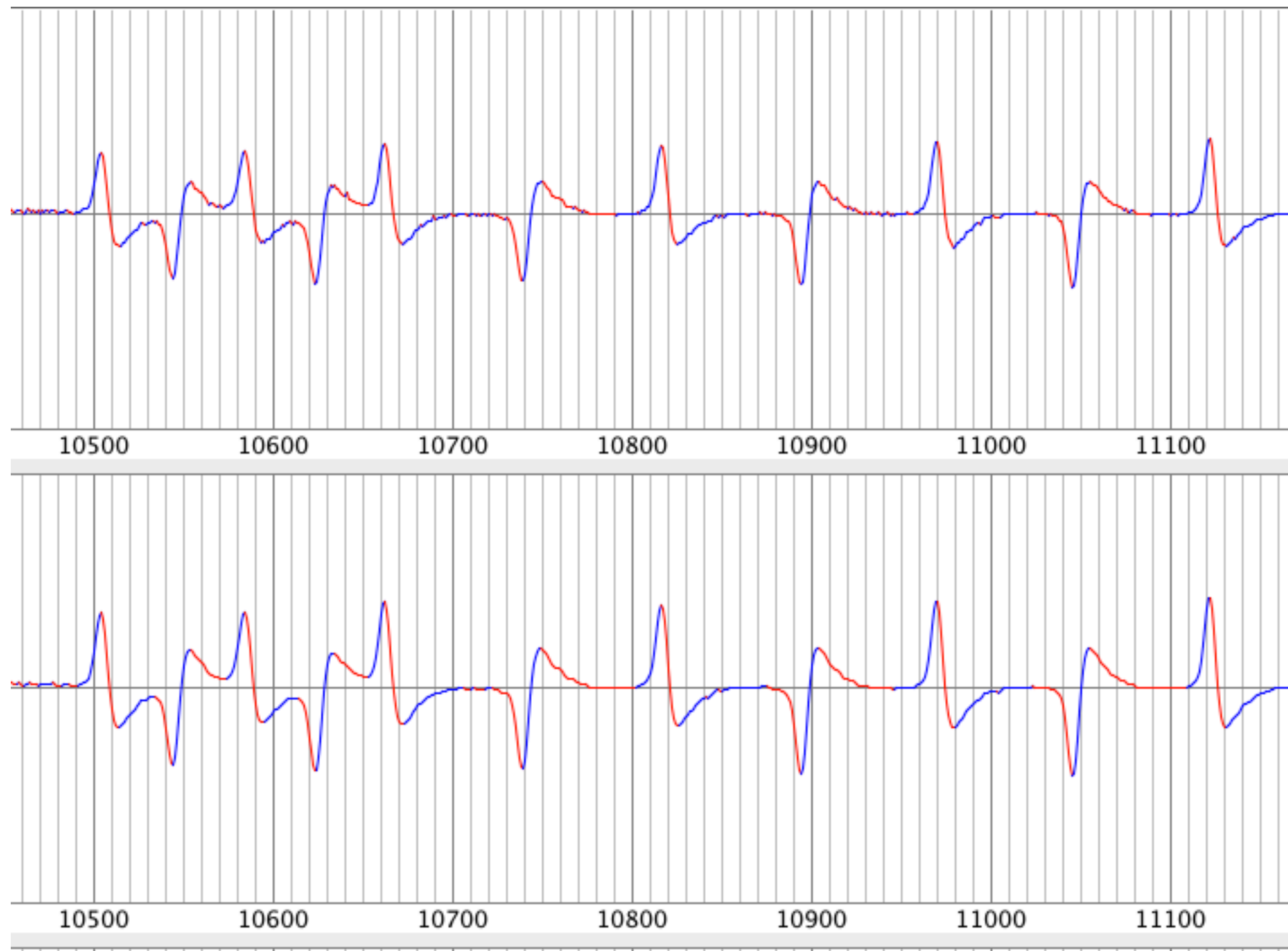
# Challenges

- > Swipe speed
- > Device sample rate
- > Audio correction

# Step 1: Swipe Detection

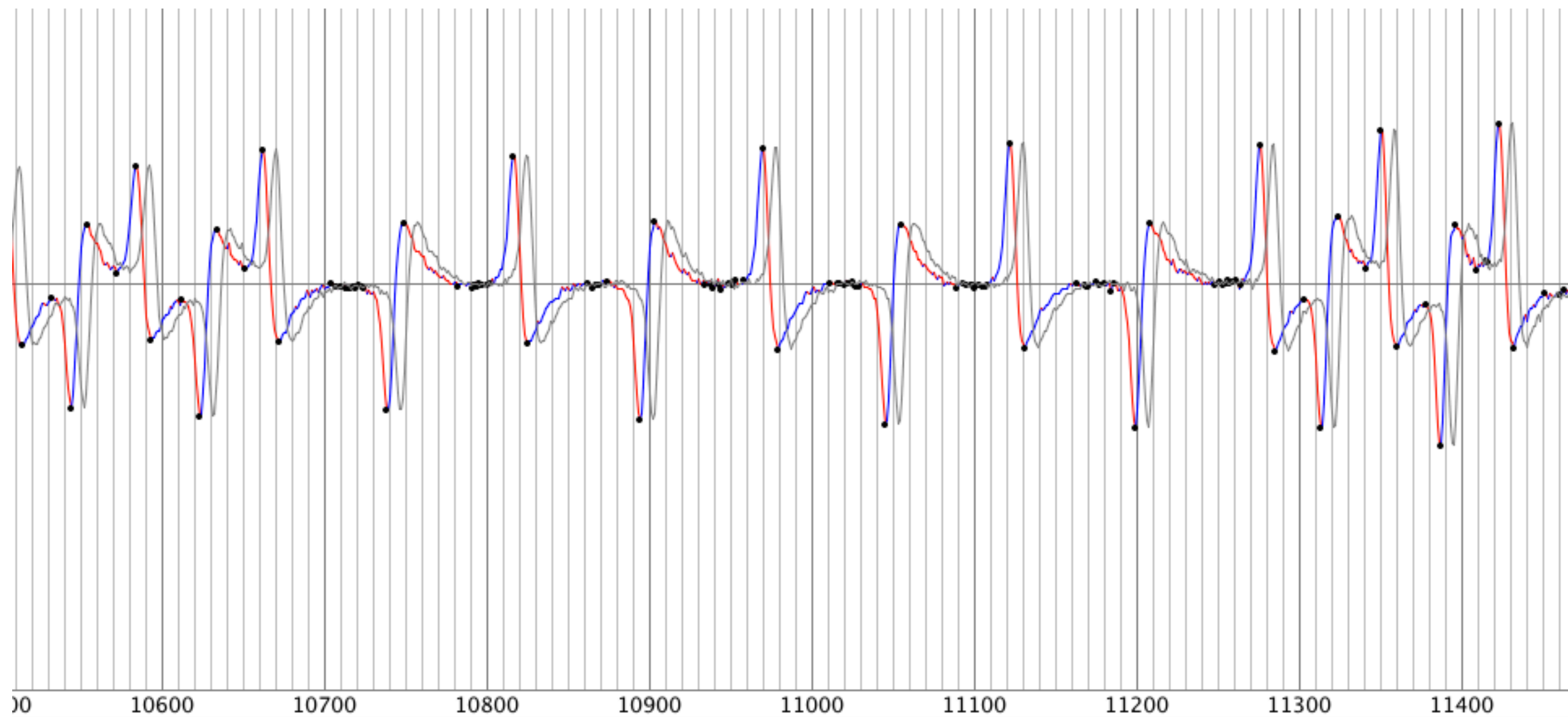


## Step 2: Denoising

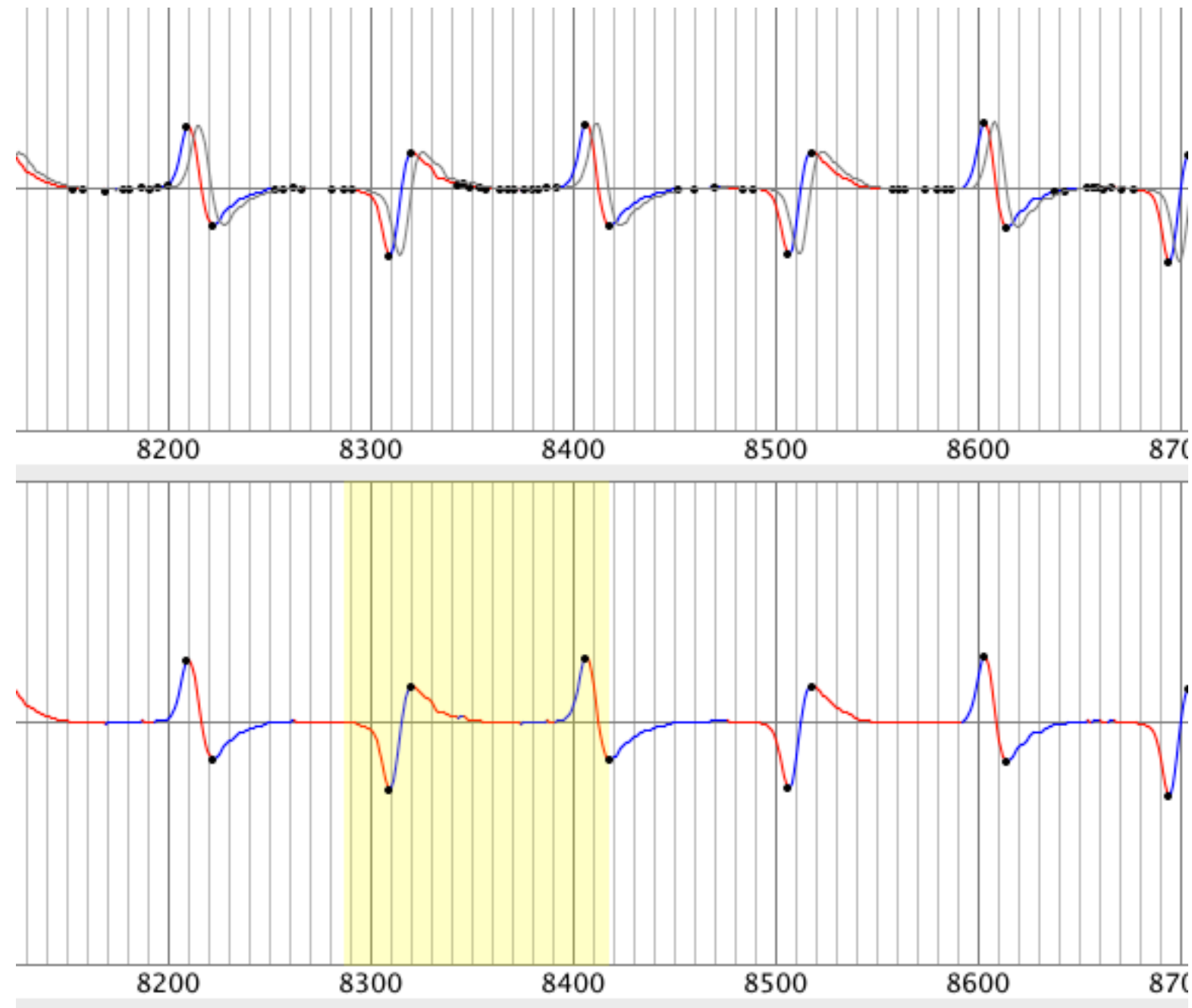




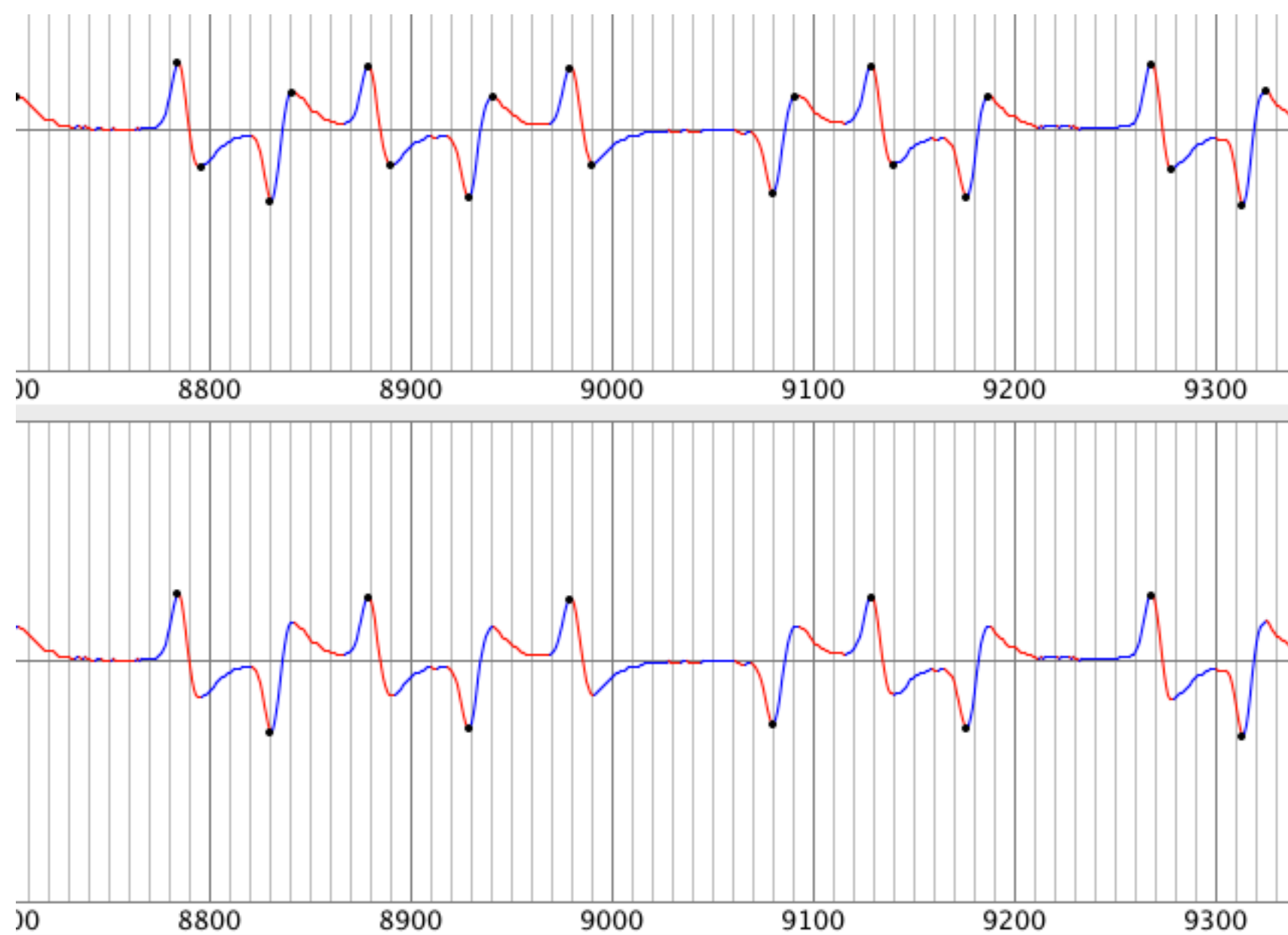
# Step 3: Peak Detection



# Step 4: Window Peak Removal

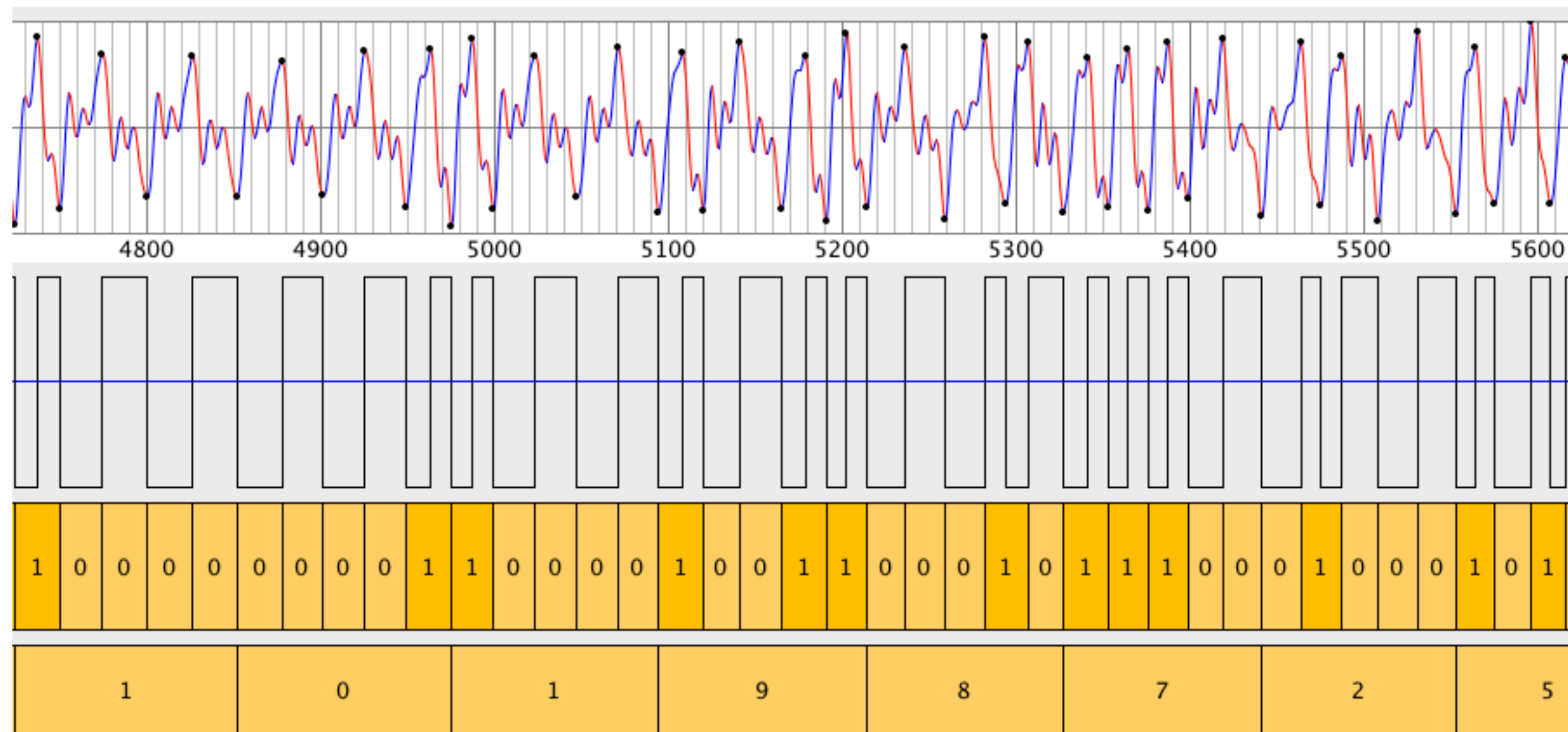


# Step 5: Consecutive Peak Removal

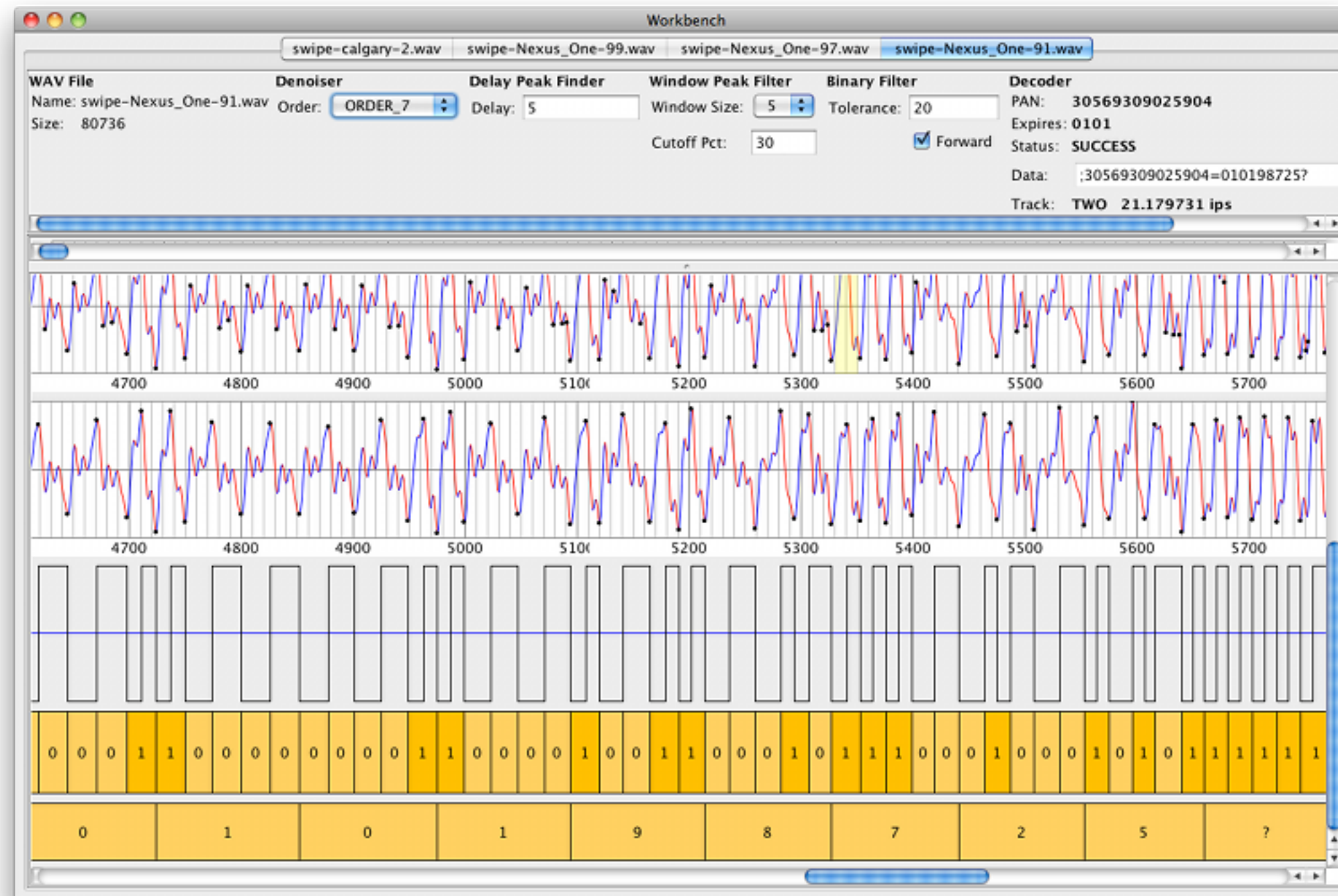




# Step 6: Decoding



# Workbench



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





[adultswim.com]



# 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

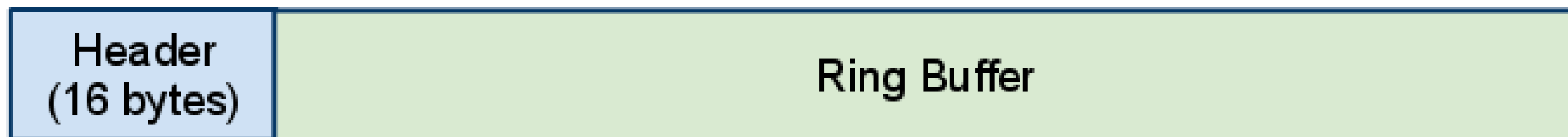
```
public class QueueFile {  
    public QueueFile(File file) throws IOException { ... }  
  
    public void add(byte[] data) throws IOException { ... }  
    public void add(byte[] data, int offset, int count)  
        throws IOException { ... }  
  
    public byte[] peek() throws IOException { ... }  
    public void remove() throws IOException { ... }  
  
    public int size() { ... }  
    public void clear() throws IOException { ... }  
    public void close() throws IOException { ... }  
}
```



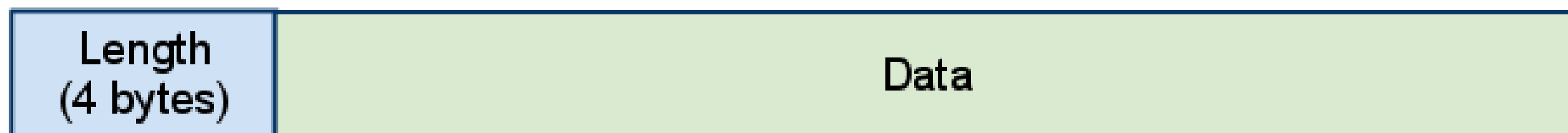
# The Implementation

- > Depends somewhat on YAFFS
  - Yet Another Flash File System
  - Android's preeminent file system
  - Supports atomic sector writes
- > Writing the header commits a change

## File



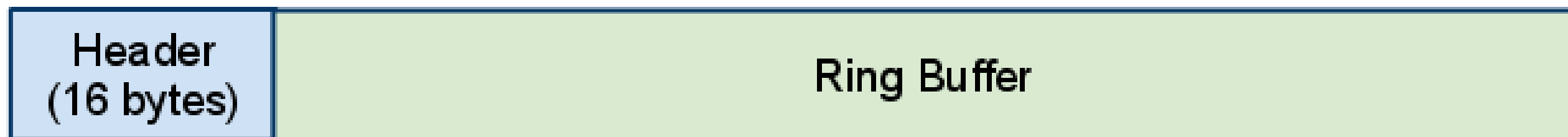
## Element



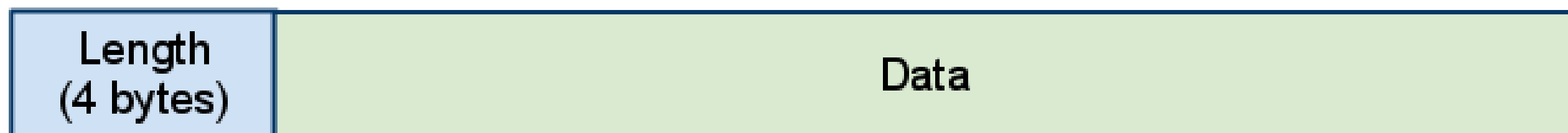
# QueueFile.add()

- > Write element data
- > Write header (16 bytes < 4k)
- > Update in-memory state

## File



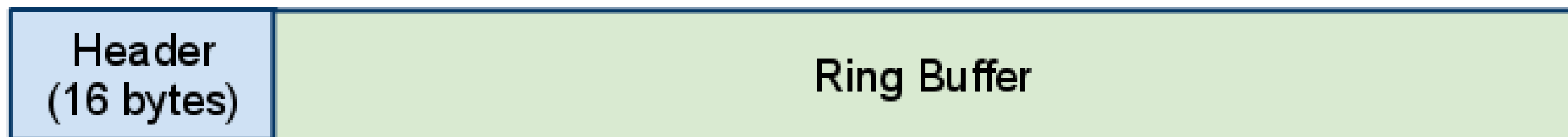
## Element



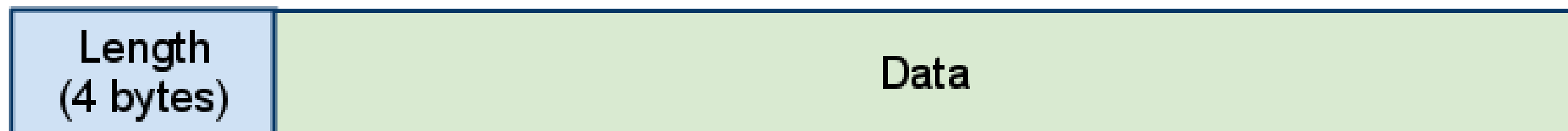
# QueueFile.remove()

- > Write header
- > Update in-memory state

## File



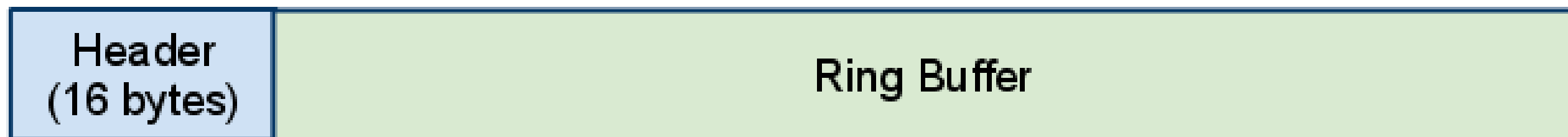
## Element



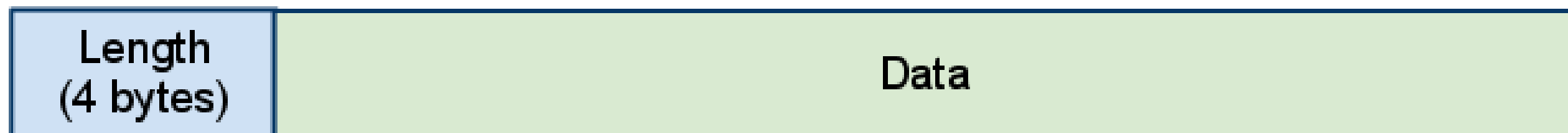
# Buffer expansion

- > `file.setLength(oldLength << 1)`
- > Make ring buffer contiguous
- > Write header (including file length)
- > Update in-memory state

## File



## Element



# Future Features


- > Support file systems without atomic segment writes
  - Rollback journal
- > Batch writes
  - Optimistic batching
  - > 3 orders of magnitude throughput






# Shake to Clear Signature

Pay Eric Burke  
**\$54.23**



X

 8431 Sign here.

**Cancel** I agree to pay the above total amount according to my card issuer agreement. **Continue**

# 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  $\text{m/s}^2$

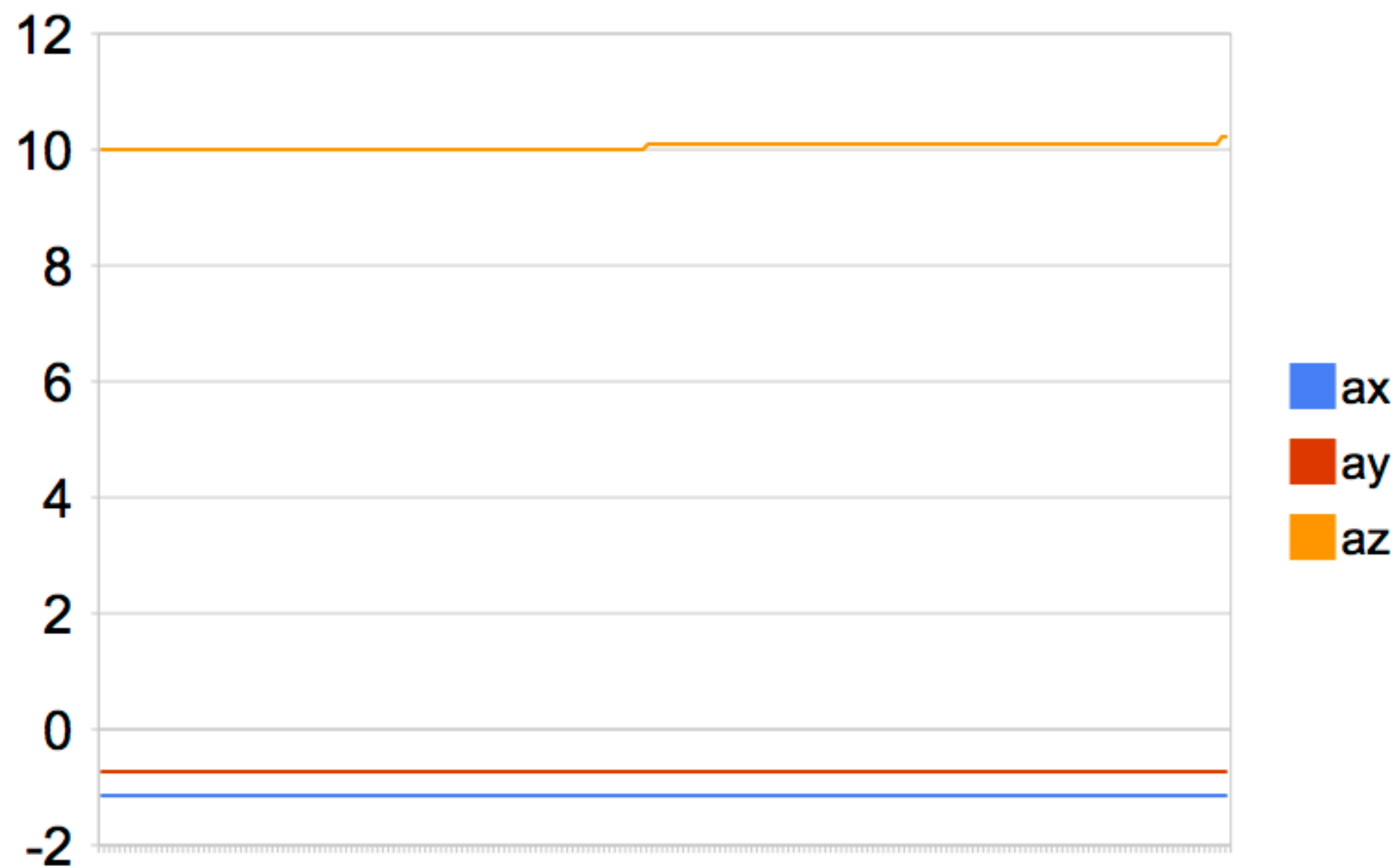
# Accelerometer Values

- > x, y, and z acceleration
- > Units are  $\text{m/s}^2$
- > Acceleration applied to device *minus force of gravity*

# Accelerometer Values

- > x, y, and z acceleration
- > Units are  $\text{m/s}^2$
- > Acceleration applied to device *minus force of gravity*
- > When flat on a table, Z acceleration = +9.81 (0 - - 9.81)

# Device at Rest





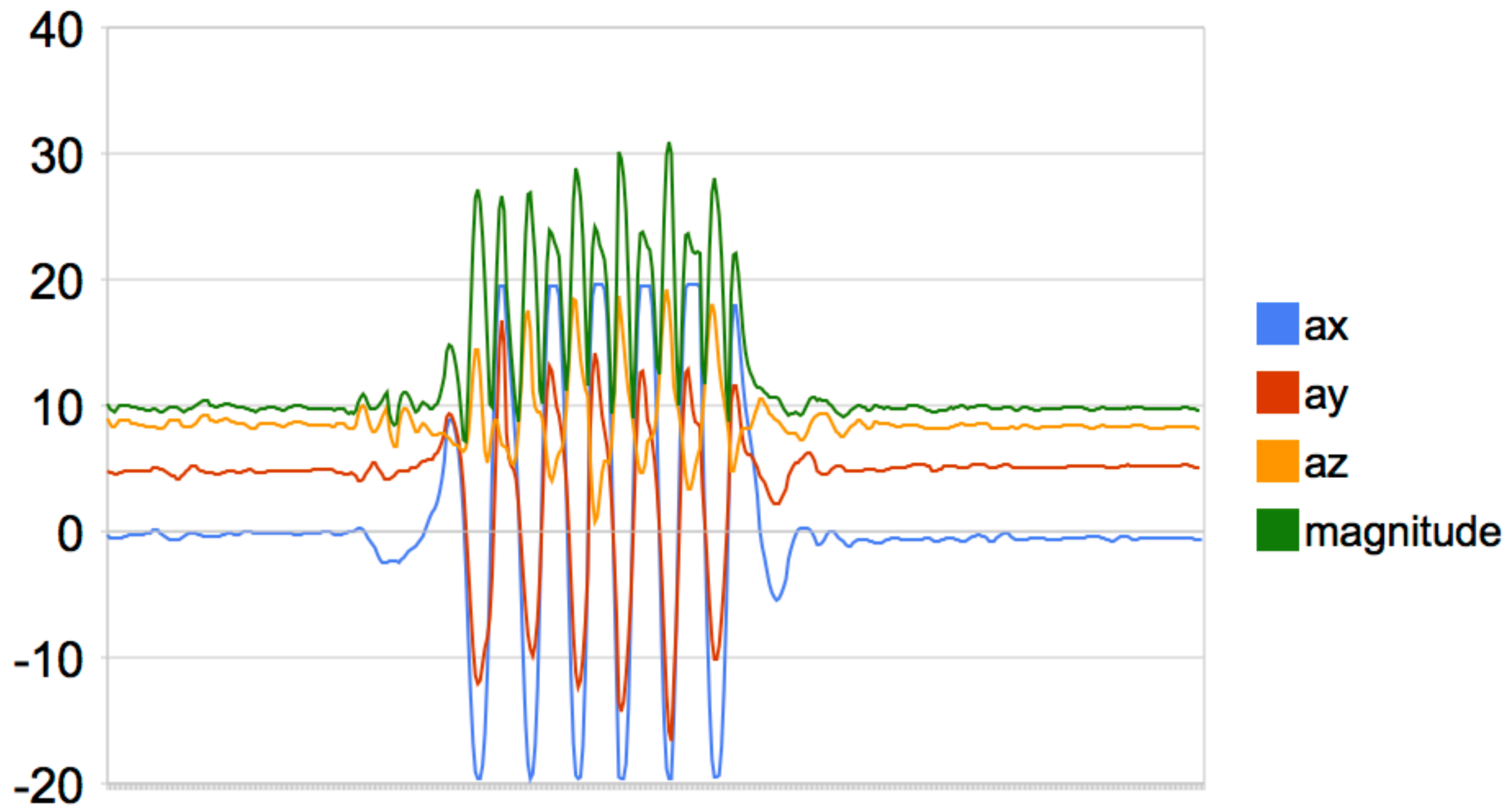
# Magnitude (Pythagorean)

```
public class Magnitude extends Activity
    implements SensorEventListener {

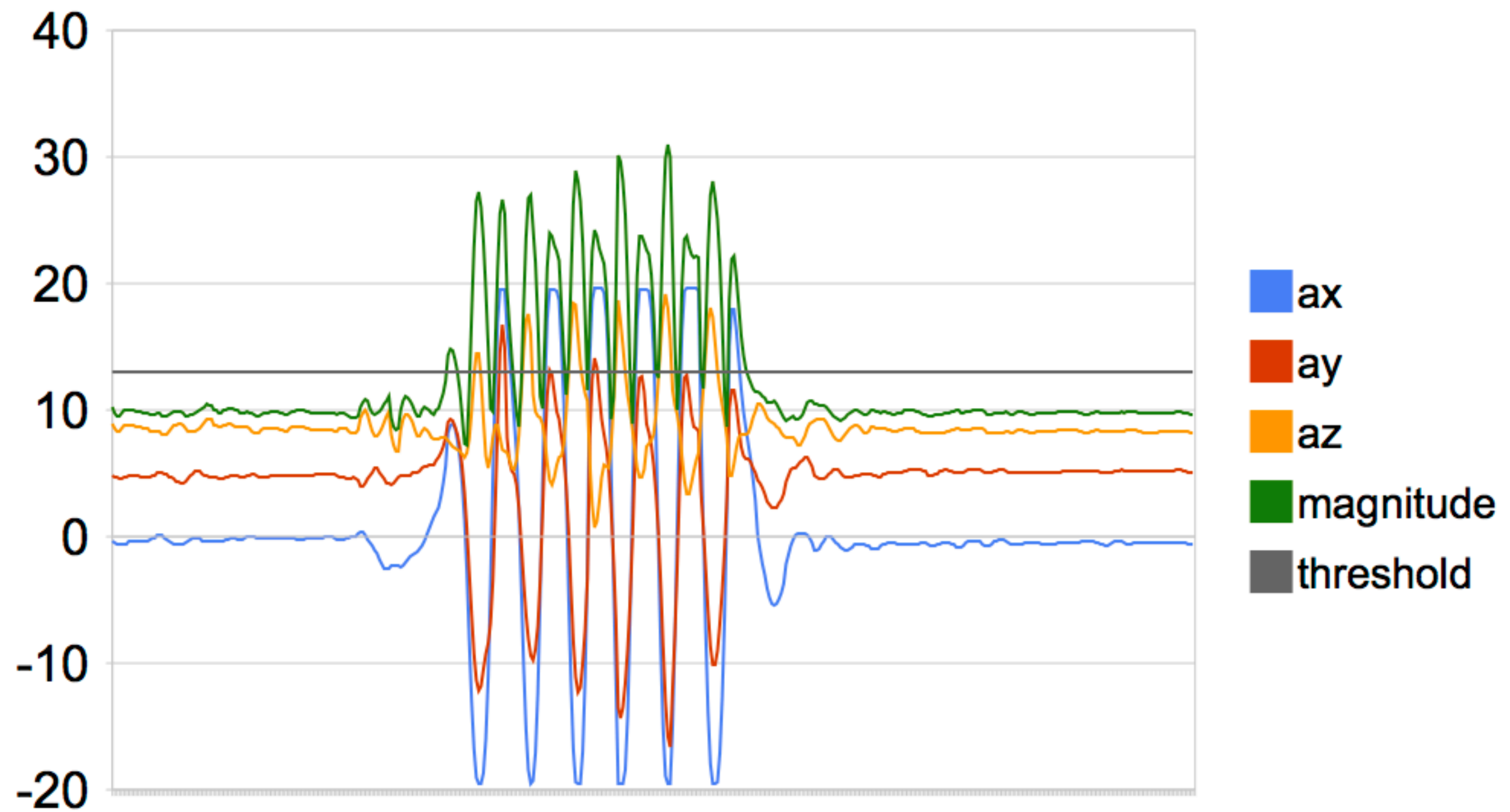
    public void onSensorChanged(SensorEvent event) {
        float ax = event.values[0];
        float ay = event.values[1];
        float az = event.values[2];

        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...
    }
}
```





# RestAdapter

- > Makes RESTful clients a breeze
- > Currently supports JSON responses
- > All user code runs in main thread

# Example

```
public interface AccountService {  
    @Path("login")  
    void login(  
        @Named("email") String email,  
        @Named("password") String password,  
        Callback<LoginResponse> callback);  
}
```

# ServerCall

- > UI handler for server calls
- > Handles all dialogs
  - Progress
  - Operation failed
  - Network unavailable
  - Server unavailable
  - Unexpected error
- > Supports retries

# Examples

> Login

> Signup

> Authorization

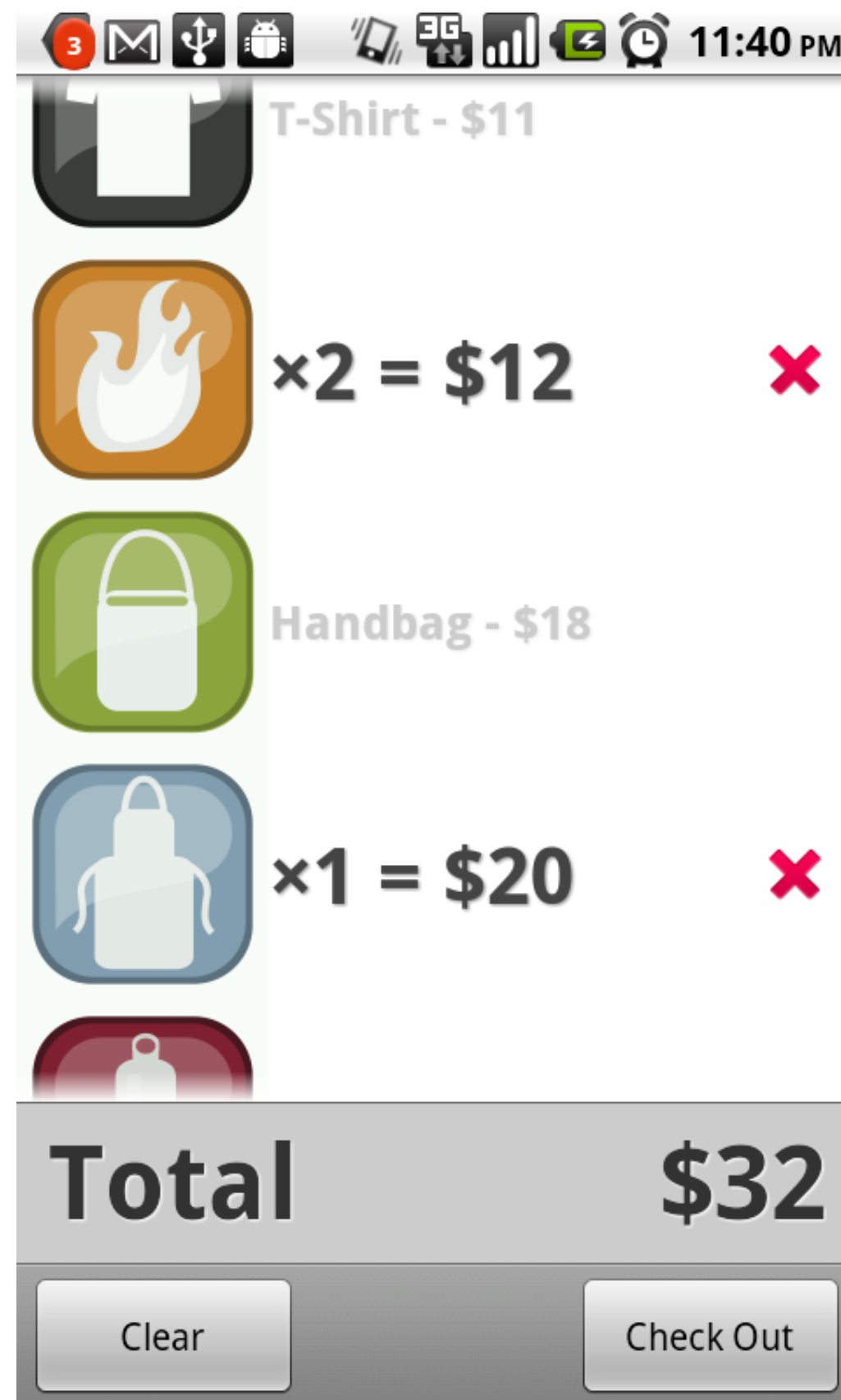
# Example

```
public class LoginActivity extends SquareActivity {
    @Inject Session session;
    private Login login;

    @Override protected void onCreate(Bundle state) {
        super.onCreate(state);
        login = new Login(session);
        ...
    }

    public void login() {
        login.call();
    }
    ...
    class Login extends ServerCall {
        private final Session session;
        @Inject private Login(Session session) {
            super("Logging in...", "Login failed.");
            this.session = session;
            manageWith(LoginActivity.this);
        }
        @Override protected void callServer(Callback<SimpleResponse> callback) {
            session.login(getEmail(), getPassword(), callback);
        }
        @Override protected void onSuccess() {
            finish();
        }
    }
}
```

# Point-of-Sale API





# Example

```
public class TwoCents extends Activity {
    @Override public void onCreate(Bundle state) {
        super.onCreate(state);

        Square square = new Square(this);
        if (square.installationStatus()
            != Square.InstallationStatus.AVAILABLE) {
            square.requestInstallation();
        } else {
            LineItem advice = new LineItem.Builder()
                .price(2, Currency.USD) // 2 cents
                .description("Advice")
                .build();
            square.squareUp(Bill.containing(advice));
        }
    }
}
```

# Thank You!

> **`http://squareup.com/`**

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

> **We're hiring!**

# Example

```
public class TwoCents extends Activity {
    @Override public void onCreate(Bundle state) {
        super.onCreate(state);

        Square square = new Square(this);
        if (square.installationStatus()
            != Square.InstallationStatus.AVAILABLE) {
            square.requestInstallation();
        } else {
            LineItem advice = new LineItem.Builder()
                .price(2, Currency.USD) // 2 cents
                .description("Advice")
                .build();
            square.squareUp(Bill.containing(advice));
        }
    }
}
```