Android Metal Detection

From Physics to Code

A Complete Guide to Building Professional Metal Detectors Using Smartphone Technology

Based on the FCMD Project

A comprehensive guide from electromagnetic theory to real-time DSP implementation

Table of Contents

References and Bibliography

PART I: FOUNDATIONS

Understanding the fundamental principles of metal detection and why smartphones are viable platforms

Chapter 1

Introduction to Metal Detection

1.1 A Brief History

Metal detection technology has evolved dramatically since Gerhard Fischer's invention of the first portable metal detector in 1925. Early detectors were bulky, consumed significant power, and could only detect large metallic objects at shallow depths. The technology has progressed through several key eras:

- **1925-1960:** Beat Frequency Oscillator (BFO) detectors - simple but limited discrimination
- **1960-1980:** Very Low Frequency (VLF) detectors - improved depth and discrimination
- **1980-2000:** Pulse Induction (PI) and multi-frequency detectors - better ground handling
- **2000-2020:** Digital signal processing and microcontroller-based detectors
- **2020-Present:** Smartphone-based detection - democratizing the technology

1.2 The Smartphone Revolution

Modern smartphones contain powerful processors, high-quality audio hardware, and sophisticated sensors that make them viable platforms for metal detection. The FCMD (Field Coil Metal Detector) project demonstrates that a consumer Android device can perform DSP tasks that once required dedicated hardware costing thousands of dollars.

> **Key Insight:** A modern smartphone has more computing power than the Apollo 11 guidance computer. Its audio subsystem can sample at rates up to 192 kHz with 24-bit precision, providing the foundation for professional-grade signal processing.

1.3 How Metal Detectors Work: The Basics

All metal detectors operate on the same fundamental principle: **electromagnetic induction**. The process involves three stages:

1. **Transmission:** An alternating current through a transmit coil creates an oscillating electromagnetic field
2. **Target Interaction:** When this field encounters conductive material (metal), it induces eddy currents in the target
3. **Reception:** These eddy currents create their own electromagnetic field, which is detected by a receive coil

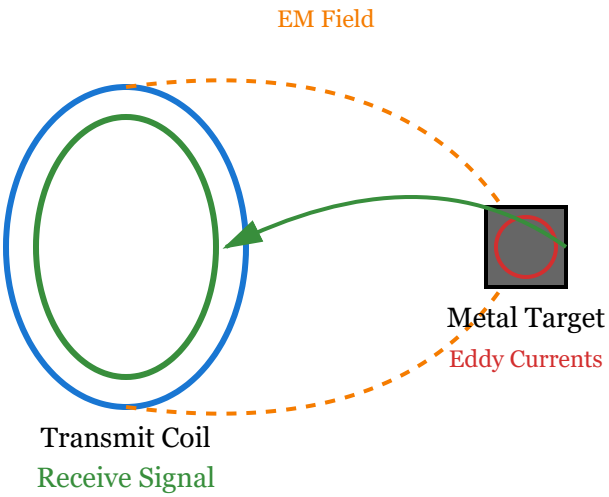

Figure 1.1: Basic metal detection principle showing transmit field, target interaction, and receive signal

1.4 Types of Metal Detectors

1.4.1 VLF (Very Low Frequency)

VLF detectors operate at frequencies between 3-30 kHz and use continuous wave transmission. They excel at discrimination (identifying target type) but can struggle with highly mineralized ground. The FCMD project is a VLF-type detector.

1.4.2 PI (Pulse Induction)

PI detectors send short bursts of current through the transmit coil, then measure the decay time of the induced signal. They handle mineralized ground better but have poor discrimination

capabilities.

1.4.3 Multi-Frequency

Modern detectors transmit multiple frequencies simultaneously, combining the benefits of different frequency responses. The FCMD project implements this approach, transmitting 1-24 logarithmically-spaced tones between 1 kHz and 20 kHz.

> **Why Multiple Frequencies?** Different metals respond differently to various frequencies. Ferrous metals show dramatic phase shifts across frequencies, while non-ferrous metals have relatively flat responses. By analyzing multiple frequencies simultaneously, we can identify target composition.

1.5 Target Discrimination

The ability to distinguish between different types of metals is what separates professional detectors from simple presence-detection devices. Discrimination relies on measuring two key properties:

- **Conductivity:** How easily electrons flow through the material (copper > aluminum > iron)
- **Magnetic Permeability:** How the material responds to magnetic fields (iron is ferromagnetic, copper is not)

These properties affect both the *amplitude* and *phase* of the received signal. By analyzing these characteristics across multiple frequencies, we can calculate a VDI (Visual Discrimination Indicator) value that classifies the target.

1.6 What This Book Covers

This book provides a complete journey from electromagnetic theory to working code:

- **Theory:** Maxwell's equations, eddy currents, skin effect, and EM propagation
- **Signal Processing:** IQ demodulation, phase analysis, FFT, and digital filtering
- **Implementation:** Android AudioTrack/AudioRecord, real-time DSP, and performance optimization
- **Practical Application:** Ground balance, depth estimation, and field calibration

By the end, you will understand not just *how* to build a metal detector, but *why* each design decision was made and how to optimize performance for your specific use case.

Chapter 2

Physics of Electromagnetic Induction

2.1 Maxwell's Equations

The operation of a metal detector is governed by Maxwell's equations, which describe how electric and magnetic fields interact. While a full derivation is beyond our scope, we'll focus on the equations most relevant to metal detection.

2.1.1 Faraday's Law of Induction

Faraday's law states that a changing magnetic field induces an electric field:

$$\nabla \times E = -\partial B/\partial t$$

In practical terms: when we send an alternating current through our transmit coil, we create a time-varying magnetic field **B**. This changing field induces an electric field **E** in nearby conductors (our metal targets), which causes current to flow.

2.1.2 Ampère's Law with Maxwell's Addition

$$\nabla \times H = J + \partial D/\partial t$$

This tells us that the induced currents (eddy currents) in the target create their own magnetic field, which we detect with our receive coil.

2.2 Eddy Currents

When a conductive object is placed in a time-varying magnetic field, circular currents called **eddy currents** flow within the conductor. These currents follow closed loops and create their own magnetic field that opposes the original field (Lenz's law).

> **Key Principle:** The strength of eddy currents depends on:
>
> - Target conductivity (σ) - higher conductivity = stronger currents
> - Frequency of the field (f) - higher frequency = stronger currents
> - Target size and shape - larger area = more current path

2.2.1 Penetration Depth (Skin Effect)

At higher frequencies, eddy currents concentrate near the surface of conductors. The penetration depth δ is:

$$\delta = \sqrt{(2\rho / (\omega\mu))}$$

where:

ρ = resistivity ($\Omega \cdot m$)

$\omega = 2\pi f$ (angular frequency)

μ = permeability (H/m)

This is why multi-frequency detection works: high frequencies give strong signals from shallow/surface targets, while low frequencies penetrate deeper and respond more uniformly to the bulk of the target.

Material	Skin Depth @ 1 kHz	Skin Depth @ 10 kHz	Skin Depth @ 20 kHz
Copper	2.1 mm	0.66 mm	0.47 mm
Aluminum	2.7 mm	0.85 mm	0.60 mm
Iron	0.20 mm	0.063 mm	0.045 mm

2.3 Phase Relationships

One of the most important discriminating factors is the **phase relationship** between the transmitted and received signals. This phase shift depends on the target's electromagnetic properties.

2.3.1 Resistive vs. Inductive Response

We can model the target's response as having two components:

- **Resistive (In-Phase):** Energy dissipated in the target as heat - proportional to conductivity
- **Inductive (Quadrature):** Energy stored in the magnetic field - related to permeability

The phase angle θ between transmitted and received signals is:

```
θ = arctan(X_L / R)
where:
X_L = inductive reactance = ωL
R = resistance
```

Example: Copper vs. Iron

Copper coin: High conductivity, non-magnetic
- Strong resistive component
- Weak inductive component
- Phase angle: ~15-30°

Iron nail: Moderate conductivity, highly magnetic
- Moderate resistive component
- Strong inductive component
- Phase angle: ~60-90°

By measuring phase angle, we can distinguish these materials!

2.4 Frequency Response

Different metals exhibit different frequency responses due to their electrical and magnetic properties. This is the foundation of multi-frequency discrimination.

2.4.1 High Conductors (Copper, Silver)

- Strong response at all frequencies
- Relatively flat phase vs. frequency curve
- High-frequency response maintained due to high conductivity

2.4.2 Medium Conductors (Aluminum, Brass)

- Moderate response, stronger at lower frequencies
- Some phase variation with frequency
- High frequencies begin to attenuate

2.4.3 Ferrous Metals (Iron, Steel)

- Strong magnetic permeability dominates
- Steep phase slope across frequencies (negative slope)
- This characteristic slope is the key identifier

> **Important:** The phase slope across frequencies is more reliable than absolute phase at any single frequency. This is why the FCMD project calculates phase slope in degrees per kHz as a primary discrimination parameter.

2.5 Target Size and Distance

The detected signal strength decreases rapidly with distance. For small loop antennas (like our coil), the relationship approximates:

```
Signal ∝ 1 / r³
where r = distance from coil to target
```

This inverse cube relationship means:

- Doubling the distance reduces signal to 1/8 strength
- Small increases in depth dramatically reduce detection capability
- Larger targets can be detected deeper (more eddy current area)

This rapid falloff is why depth estimation is challenging - a small error in measured amplitude translates to significant depth uncertainty.

2.6 Ground Mineralization

Naturally occurring minerals in soil, particularly iron oxides and salts, respond to the electromagnetic field just like metal targets. This creates a constant background signal that can overwhelm small target signals.

2.6.1 Types of Ground Minerals

Mineral Type	Effect	Common Locations
Iron Oxides (Magnetite)	Magnetic response, phase shift	Red clay soils, volcanic areas
Salts	Conductive response	Beaches, desert areas
Clay Minerals	Variable, moisture-dependent	Agricultural areas

Ground balance algorithms (Chapter 10) address this by measuring the ground's I/Q response and subtracting it from all subsequent readings, effectively "nulling out" the ground signal while preserving target signals.

2.7 Summary

The key physics principles for metal detection are:

1. **Electromagnetic induction** creates eddy currents in conductive targets
2. **Eddy currents** generate a secondary magnetic field we can detect
3. **Phase relationships** between transmit and receive signals reveal target properties
4. **Frequency response** differs between ferrous and non-ferrous metals

5. **Skin effect** causes high frequencies to respond more to surface characteristics
6. **Signal strength** decreases as cube of distance, making depth estimation difficult

Understanding these principles allows us to design signal processing algorithms that extract maximum information from the received signal.

Chapter 3

Why Android? Capabilities and Limitations

3.1 The Case for Smartphone-Based Detection

At first glance, using a smartphone for metal detection might seem unconventional. However, modern Android devices possess several characteristics that make them ideal for this application:

3.1.1 Processing Power

A typical mid-range Android phone (2024) contains:

- Octa-core CPU running at 2+ GHz
- 4-8 GB RAM
- Hardware floating-point acceleration
- Dedicated DSP cores on many devices

This is more than sufficient for real-time IQ demodulation of 24 simultaneous frequencies at 44.1 kHz sample rate.

3.1.2 Audio Subsystem

Modern Android audio hardware rivals professional audio interfaces:

Specification	Typical Android Device	Professional Audio Interface
Sample Rate	44.1 - 192 kHz	44.1 - 192 kHz
Bit Depth	16-24 bit	16-24 bit

THD+N	0.001% - 0.01%	0.001% - 0.005%
Dynamic Range	90-120 dB	100-120 dB

> **Real-World Performance:** The FCMD project operates at 44.1 kHz sample rate with buffer sizes of ~1920 samples, yielding a callback rate of 23.2 Hz and latency of 43.5 ms. This is more than adequate for metal detection, where targets change position relatively slowly.

3.1.3 Additional Sensors

Beyond audio, Android devices offer:

- **Accelerometer/Gyroscope:** Could detect coil motion and height
- **GPS:** Geotag finds, create maps of detected targets
- **Compass:** Log direction of finds
- **Bluetooth:** Connect to external coils or headphones

3.2 Hardware Limitations

While powerful, smartphones have constraints we must work within:

3.2.1 Audio Output Power

Smartphone headphone amplifiers typically deliver 30-100 mW into 16-32Ω loads. This limits our transmit coil design:

- Coil impedance must match headphone output (16-32Ω typical)
- Limited current means limited field strength and detection depth
- External amplification may be needed for larger coils

> **Power Limitation Impact:** Commercial detectors may use 1-5W transmit power. Our smartphone-based system is limited to ~0.1W. This reduces maximum depth by approximately 30-50% compared to professional units, but is acceptable for hobbyist and educational use.

3.2.2 Audio Buffer Size

Android's audio system determines buffer sizes based on hardware capabilities. We cannot arbitrarily choose small buffers for lower latency. The FCMD project accepts the system-determined buffer size and works within it.

3.2.3 Sample Rate Support

Not all devices support all sample rates. The FCMD project uses 44.1 kHz, which is universally supported. Higher rates (96 kHz, 192 kHz) would allow higher frequency transmission but aren't necessary for our 1-20 kHz frequency range (Nyquist theorem requires minimum 40 kHz for 20 kHz signals).

3.2.4 Thermal Constraints

Sustained CPU usage can cause thermal throttling. The FCMD architecture minimizes CPU load by:

- Using efficient IIR filters instead of FFT where possible
- Processing only mono channel for receive (not stereo)
- Update rate throttling (30 Hz instead of 40+ Hz callback rate)

3.3 Android Audio Architecture

Understanding Android's audio stack helps us optimize performance.

3.3.1 The Audio HAL (Hardware Abstraction Layer)

Android audio flows through multiple layers:

```
Your App (Java/Kotlin)
        ↓
AudioTrack / AudioRecord (Framework)
        ↓
Audio Flinger (Native Service)
        ↓
Audio HAL (Hardware Abstraction)
        ↓
ALSA / Hardware Drivers
        ↓
```

```
Audio Codec Hardware
```

Each layer adds latency. The FCMD project uses the lowest-latency path available to regular apps: `AudioTrack.MODE_STREAM` with `PERFORMANCE_MODE_LOW_LATENCY` (via buffer size selection).

3.3.2 Buffer Size Selection

Buffer size is crucial for balancing latency and reliability:

$$\text{bufferSize} = \text{AudioTrack.getMinBufferSize}(\text{sampleRate}, \text{channels}, \text{encoding}) \times \text{multiplier}$$

The FCMD project uses `multiplier = 1` for minimum latency. Larger multipliers provide more tolerance for CPU scheduling jitter but increase latency proportionally.

3.4 Real-Time Constraints

Metal detection requires **real-time** processing: audio callbacks must complete before the next buffer arrives, or we get dropouts (xruns).

3.4.1 Thread Priority

The FCMD project sets audio threads to maximum priority:

```
android.os.Process.setThreadPriority(
    android.os.Process.THREAD_PRIORITY_URGENT_AUDIO
)
```

This gives our audio processing preferential CPU scheduling, reducing the chance of buffer underruns.

3.4.2 Computational Budget

With a buffer size of 1920 samples at 44.1 kHz, we have 43.5 ms to process each buffer. Our IQ demodulation for 24 frequencies takes approximately 5-10 ms on a modern CPU, leaving comfortable margin.

Performance Breakdown (typical mid-range phone):

- IQ demodulation (24 tones): 6 ms
- Ground balance: 1 ms
- VDI calculation: 0.5 ms
- Depth estimation: 0.3 ms
- **Total: ~8 ms out of 43.5 ms available**

This 18% CPU usage leaves margin for OS overhead and prevents thermal issues.

3.5 Battery Considerations

Continuous audio processing consumes significant power. The FCMD project typically draws:

- CPU: ~300-500 mW (single core at 30-40% load)
- Audio subsystem: ~100-200 mW
- Display: ~500-1500 mW (major consumer)
- **Total: ~1-2W during operation**

On a 4000 mAh battery at 3.8V (~15 Wh), this yields 7-15 hours of runtime, limited primarily by display power. Using a low-brightness setting or external display can extend this significantly.

3.6 Advantages Over Dedicated Hardware

Despite limitations, smartphone-based detection offers unique advantages:

Feature	Dedicated Detector	Smartphone-Based
Initial Cost	$300-$3000	$0 (existing device) + coil
Software Updates	Rarely, if ever	Continuous improvement possible
Display	Small LCD, limited info	Full color, high resolution, flexible UI
Data Logging	Limited or none	Unlimited, with GPS, photos, notes

Algorithm Changes	Fixed in hardware	Infinitely flexible
Connectivity	None or proprietary	WiFi, Bluetooth, cellular, cloud

3.7 Design Philosophy

The FCMD project embraces a particular design philosophy:

1. **Work with the platform, not against it:** Accept system buffer sizes rather than fighting for unrealistic low latency
2. **Optimize what matters:** Focus optimization on the DSP algorithms, not UI polish
3. **Leave headroom:** Never push CPU to 100%; thermal throttling will ruin performance
4. **Measure everything:** Log actual callback rates, latencies, and processing times
5. **Fail gracefully:** Detect and handle buffer underruns rather than crashing

3.8 Summary

Android devices are viable metal detector platforms because:

- Sufficient CPU power for real-time multi-frequency IQ demodulation
- High-quality audio hardware with low noise and wide dynamic range
- Flexible software platform allowing continuous algorithm improvement
- Built-in sensors (GPS, accelerometer) enhance functionality

The main limitations are transmit power (limiting depth) and audio latency (limiting update rate), but these are acceptable tradeoffs for the advantages gained.

Chapter 8

IQ Demodulation Theory

8.1 What is IQ Demodulation?

IQ (In-phase/Quadrature) demodulation is the core signal processing technique that allows us to extract both amplitude and phase information from a received signal. It's called "IQ" because we measure two components:

- **I (In-phase):** Component aligned with the transmitted signal
- **Q (Quadrature):** Component 90° out of phase with the transmitted signal

Together, these form a complex number that completely describes the signal's magnitude and phase at a specific frequency.

8.2 Mathematical Foundation

For a received signal *s(t)* and a reference frequency *f*, we calculate I and Q by mixing with sine and cosine:

$$I(t) = s(t) \times \cos(2\pi ft)$$
$$Q(t) = -s(t) \times \sin(2\pi ft)$$

After low-pass filtering to remove high-frequency components:

$$\text{Amplitude} = \sqrt{I^2 + Q^2}$$
$$\text{Phase} = \text{atan2}(Q, I)$$

8.2.1 Why This Works

Consider a pure tone at frequency f with amplitude A and phase φ:

$$s(t) = A\cos(2\pi ft + \varphi)$$

Mixing with cosine:

$$I = A\cos(2\pi ft + \varphi) \times \cos(2\pi ft)$$
$$= (A/2)[\cos(\varphi) + \cos(4\pi ft + \varphi)]$$

After low-pass filtering to remove the 2f term:

$$I = (A/2)\cos(\varphi)$$

Similarly for Q:

$$Q = (A/2)\sin(\varphi)$$

Combining these:

$$A = \sqrt{(I^2 + Q^2)} = \sqrt{[(A/2)^2\cos^2(\varphi) + (A/2)^2\sin^2(\varphi)]} = A/2 \times \sqrt{[\cos^2(\varphi) + \sin^2(\varphi)]} = A/2$$
$$\varphi = \text{atan2}(Q, I) = \text{atan2}((A/2)\sin(\varphi), (A/2)\cos(\varphi)) = \varphi$$

Thus we've extracted both amplitude (scaled by 2) and phase!

8.3 Implementation in FCMD

The FCMD project implements IQ demodulation in `IQDemodulator.kt`:

```
class SingleToneDemodulator(
    private val frequency: Double,
    private val sampleRate: Int
) {
```

```
    private var phase = 0.0
    private val phaseIncrement = 2.0 * PI * frequency / sampleRate

    // IIR filter state
    private val filterAlpha = 0.01  // ~10 Hz cutoff
    private var iFiltered = 0.0
    private var qFiltered = 0.0

    fun analyze(samples: FloatArray): ToneAnalysis {
        for (sample in samples) {
            // Quadrature mixing
            val i = sample * cos(phase)
            val q = -sample * sin(phase)

            // IIR low-pass filter
            iFiltered = filterAlpha * i + (1.0 - filterAlpha) * iFiltered
            qFiltered = filterAlpha * q + (1.0 - filterAlpha) * qFiltered

            // Increment phase
            phase += phaseIncrement
            if (phase >= 2.0 * PI) phase -= 2.0 * PI
        }

        // Calculate amplitude and phase
        val amplitude = sqrt(iFiltered² + qFiltered²) * 2.0
        val phaseAngle = atan2(qFiltered, iFiltered)

        return ToneAnalysis(frequency, amplitude, phaseAngle,
                            iFiltered, qFiltered)
    }
}
```

8.3.1 Phase Tracking

The demodulator maintains a running phase that increments by `phaseIncrement` each sample. This is equivalent to generating $\cos(2\pi ft)$ and $\sin(2\pi ft)$ but more efficient than calling trigonometric functions every sample.

8.3.2 Low-Pass Filtering

The FCMD project uses a single-pole IIR filter rather than a moving average for computational efficiency:

```
y[n] = α × x[n] + (1-α) × y[n-1]
```

With α = 0.01, the cutoff frequency is approximately:

```
f_c = (sample_rate × α) / (2π) ≈ (44100 × 0.01) / (2π) ≈ 70 Hz
```

This is fast enough to track targets moving past the coil while filtering out high-frequency noise.

> **Why IIR instead of FIR?**
> A moving average FIR filter would require storing N samples per frequency. For 24 frequencies with N=100 samples each, that's 2,400 floats (9.6 KB) plus computational cost. The IIR filter stores only 2 floats per frequency (192 bytes total) and computes faster. The tradeoff is less steep rolloff, which is acceptable for our application.

8.4 Multi-Tone Processing

The FCMD project processes multiple frequencies simultaneously. For each audio buffer of 1920 samples, we run 24 separate IQ demodulators in parallel:

```
class IQDemodulator(
    private val sampleRate: Int,
    private val targetFrequencies: List
) {
    private val demodulators = targetFrequencies.map { freq ->
        SingleToneDemodulator(freq, sampleRate)
    }

    fun analyze(samples: FloatArray): List {
        return demodulators.map { demod ->
            demod.analyze(samples)
        }
    }
}
```

This parallel processing is efficient on modern multi-core CPUs and provides complete frequency response information every audio callback (23.2 Hz on typical hardware).

8.5 Frequency Selection Strategy

The FCMD project uses logarithmically-spaced frequencies from 1 kHz to 20 kHz:

```
f[i] = f_min × (f_max / f_min)^(i / (N-1))
                for i = 0 to N-1
```

Logarithmic spacing provides:

- More samples at low frequencies (better depth penetration)
- Adequate high-frequency coverage (surface detail)
- Even distribution on a log scale (how humans perceive frequency)

Example: 8-tone logarithmic spacing (1 kHz - 20 kHz)

- Tone 1: 1000 Hz
- Tone 2: 1558 Hz
- Tone 3: 2427 Hz
- Tone 4: 3780 Hz
- Tone 5: 5888 Hz
- Tone 6: 9175 Hz
- Tone 7: 14294 Hz
- Tone 8: 20000 Hz

8.6 Performance Optimization

Several optimizations make real-time multi-tone IQ demodulation practical:

8.6.1 Incremental Phase Tracking

Rather than computing `2π × f × t / sampleRate` each sample, we increment a phase accumulator. This replaces expensive multiplication with cheap addition.

8.6.2 Single-Pole IIR Filter

As discussed, IIR filters are dramatically more efficient than FIR equivalents for moderate rolloff requirements.

8.6.3 Delayed Amplitude Calculation

We only calculate `sqrt(I² + Q²)` once per buffer, not per sample. The IIR filter provides continuous update of I and Q, but amplitude/phase are only needed at the callback rate (23.2 Hz), not sample rate (44.1 kHz).

8.7 Advantages Over FFT

Why use IQ demodulation instead of FFT (Fast Fourier Transform)?

Aspect	IQ Demodulation	FFT
Frequency Selection	Arbitrary	Fixed bins (f_s / N)
Time Resolution	Every sample	Block-based (trade with frequency resolution)
Computational Cost	O(N × M) for M tones	O(N log N)
Memory	Minimal (2 floats/tone)	Full buffer required
Latency	IIR filter delay only	Full window length

For our application with relatively few tones (24) and desire for low latency, IQ demodulation is superior to FFT.

8.8 Practical Considerations

8.8.1 Phase Unwrapping

The `atan2` function returns phase in the range [-π, +π]. When monitoring phase over time, sudden jumps from +π to -π can occur. For metal detection, we don't typically need to unwrap

phase (track continuous rotation) because we're interested in phase differences between frequencies, not absolute phase evolution.

8.8.2 DC Bias

Audio hardware may introduce DC bias (non-zero mean). This appears as a strong component at 0 Hz but doesn't affect our 1-20 kHz tones. Ground balance (Chapter 10) effectively removes DC bias by subtracting the baseline I/Q vector.

8.8.3 Filter Transients

The IIR filter has a startup transient. With $\alpha = 0.01$ and callback rate of 23.2 Hz, the time constant is:

```
τ = 1 / (f_callback × α) ≈ 1 / (23.2 × 0.01) ≈ 4.3 seconds
```

This means full settling takes ~20 seconds. In practice, 95% settling occurs in $3\tau \approx 13$ seconds, which is acceptable for startup delay.

8.9 Summary

IQ demodulation provides:

- Simultaneous amplitude and phase measurement at specific frequencies
- Low computational cost suitable for real-time processing on smartphones
- Flexible frequency selection (logarithmic spacing optimal for metal detection)
- Low latency (no block processing delay like FFT)

The FCMD implementation achieves this with simple, efficient code that processes 24 frequencies in real-time with minimal CPU usage.

Chapter 9

Phase Analysis and VDI Discrimination

9.1 The VDI Concept

VDI (Visual Discrimination Indicator) is a numerical scale, typically 0-99, that classifies detected targets by their electromagnetic properties. It originated with White's Electronics in the 1990s and has become an industry standard.

The goal: map the complex electromagnetic signature (amplitude and phase at multiple frequencies) to a single number that correlates with target composition.

9.1.1 Traditional VDI Scales

VDI Range	Typical Targets
0-30	Ferrous: iron nails, bottle caps, steel
30-45	Low conductors: aluminum foil, small rings
45-65	Mid conductors: brass, zinc pennies, pull tabs
50-70	Gold range: gold jewelry (overlaps mid)
70-99	High conductors: copper, silver, large targets

9.2 Phase Slope Analysis

The FCMD project's VDI calculation is based primarily on **phase slope** across frequencies. This is more reliable than single-frequency phase because it reveals magnetic properties.

9.2.1 Why Phase Slope Works

Recall from Chapter 2 that ferrous metals have high magnetic permeability. This causes phase to shift dramatically with frequency. The phase slope is:

```
Phase Slope = (Phase_high - Phase_low) / (Freq_high - Freq_low)
```

Measured in degrees per kHz.

Example Measurements:

Iron nail (ferrous):
- 1 kHz: Phase = +30°
- 20 kHz: Phase = -130°
- Slope = (-130 - 30) / 19 = **-8.4 deg/kHz**

Copper penny (non-ferrous):
- 1 kHz: Phase = +10°
- 20 kHz: Phase = +0°
- Slope = (0 - 10) / 19 = **-0.5 deg/kHz**

The steep negative slope is a clear signature of ferrous material!

9.3 FCMD VDI Algorithm

The complete VDI calculation combines multiple factors:

```
fun calculateVDI(analysis: List): VDIResult {
    // 1. Calculate phase slope
    val phaseSlope = calculatePhaseSlope(analysis)

    // 2. Calculate conductivity index
    val conductivityIndex = calculateConductivityIndex(analysis)
```

```
    // 3. Measure phase consistency
    val phaseConsistency = calculatePhaseConsistency(analysis)

    // 4. Calculate raw VDI
    val rawVDI = if (phaseSlope < 0) {
        // Ferrous: use slope steepness
        val normalized = (phaseSlope / -10.0).coerceIn(0.0, 1.0)
        (30 * (1.0 - normalized)).toInt()
    } else {
        // Non-ferrous: use conductivity
        (30 + (conductivityIndex * 69)).toInt()
    }

    // 5. Adjust for signal strength
    val amplitude = analysis.map { it.amplitude }.average()
    val adjustment = when {
        amplitude > 0.5 -> 5   // Strong signal
        amplitude < 0.1 -> -5  // Weak signal
        else -> 0
    }

    val vdi = (rawVDI + adjustment).coerceIn(0, 99)

    // 6. Classify and return
    return VDIResult(
        vdi, confidence, targetType,
        phaseSlope, conductivityIndex, depthEstimate
    )
}
```

9.3.1 Conductivity Index

For non-ferrous metals, conductivity index separates aluminum from copper/silver:

```
    ConductivityIndex = (Avg_Amp_HighFreq / Avg_Amp_LowFreq) / 2.0
```

High conductors maintain amplitude at high frequencies (skin depth still allows penetration). Low conductors attenuate rapidly.

9.3.2 Phase Consistency

Phase consistency measures how "clean" the target signal is:

```
PhaseConsistency = 1.0 - (StdDev(phases) / 90°)
```

A single solid target has consistent phase across frequencies. Multiple targets, junk, or heavy mineralization show poor consistency.

9.4 Confidence Scoring

Not all VDI readings are equally reliable. The FCMD project calculates confidence as:

```
Confidence = (Amplitude_Score × 0.3) + (Phase_Consistency × 0.7)
```

Phase consistency is weighted more heavily because it's the best indicator of a single, solid target versus trash or multiple objects.

Confidence Range	Interpretation	Action
0.8 - 1.0	High: Clean single target	Trust VDI, worth digging
0.5 - 0.8	Medium: Decent signal	VDI probably accurate
0.3 - 0.5	Low: Noisy or multiple targets	VDI uncertain
0.0 - 0.3	Very Low: Junk or interference	Returns UNKNOWN type

9.5 Target Type Classification

Based on VDI and phase slope, targets are classified:

```
fun classifyTarget(vdi: Int, phaseSlope: Double,
                   consistency: Double): TargetType {
    if (consistency < 0.3) return TargetType.UNKNOWN

    return when {
        vdi <= 30 && phaseSlope < -3.0 -> TargetType.FERROUS
        vdi <= 45 -> TargetType.LOW_CONDUCTOR
        vdi >= 70 -> TargetType.HIGH_CONDUCTOR
```

```
            vdi in 50..70 -> TargetType.GOLD_RANGE
            vdi in 46..69 -> TargetType.MID_CONDUCTOR
            else -> TargetType.UNKNOWN
        }
    }
```

9.6 Limitations and Challenges

9.6.1 Target Size Ambiguity

VDI cannot distinguish between composition and size. A small copper coin gives similar VDI to a large aluminum can, despite different materials. This is fundamental - we measure electromagnetic response, not material directly.

9.6.2 Orientation Effects

A coin flat vs. edge-on can show different VDI (±10 points). Phase measurements are somewhat less sensitive to orientation than amplitude, which is why we weight phase slope heavily.

9.6.3 The Gold/Trash Problem

Gold jewelry (VDI 50-70) overlaps with aluminum pull tabs (VDI 45-65). No single-frequency or even multi-frequency detector can perfectly separate these. This is why detectorists learn to dig "iffy" signals in goldfields.

> **Fundamental Limitation:** Some targets are electromagnetically indistinguishable. A thin gold ring and an aluminum pull tab can produce identical multi-frequency responses. Advanced techniques (3D imaging, time-domain analysis) help but don't completely solve this.

9.7 Calibration and Learning

VDI scales benefit from calibration with known targets. The FCMD thresholds (30, 45, 65, 70) are starting points. You can refine them:

1. Collect a set of known targets (coins, jewelry, trash)

2. Measure each target's VDI multiple times
3. Build a histogram of VDI values per target type
4. Adjust thresholds to minimize overlap

Machine learning approaches (Chapter 14) can automatically learn optimal thresholds from labeled data.

9.8 Summary

Effective VDI discrimination requires:

- Multi-frequency analysis (phase slope reveals ferrous vs. non-ferrous)
- Conductivity index (separates copper from aluminum among non-ferrous)
- Phase consistency (distinguishes single targets from junk)
- Confidence scoring (helps user know when to trust the reading)

The FCMD algorithm achieves good discrimination with modest computational cost, suitable for real-time operation on smartphone hardware.

Chapter 10

Ground Balance Algorithms

10.1 The Ground Mineralization Problem

Ground minerals produce constant I/Q vectors that can overwhelm weak target signals. Ground balance algorithms measure and subtract this baseline, effectively "nulling out" the ground while preserving target signatures.

10.2 Four Ground Balance Modes

10.2.1 OFF Mode

No ground balance applied. Use in air tests or clean beach sand.

10.2.2 Manual Mode ("Pump and Set")

User pumps coil over ground 10 times while system captures I/Q baseline. This baseline is then subtracted from all subsequent readings.

$$\text{Baseline}[f] = \text{Average}(I_1[f], I_2[f], \ldots, I_{10}[f]), \text{Average}(Q_1[f], Q_2[f], \ldots, Q_{10}[f])$$

10.2.3 Auto-Tracking Mode

Continuously adapts baseline using slow IIR filter ($\alpha = 0.0005$). Freezes when strong target detected (amplitude > 0.3) to avoid nulling the target.

$$\text{Baseline}[n] = 0.0005 \times \text{Current}[n] + 0.9995 \times \text{Baseline}[n-1]$$

Time constant ~86 seconds, allowing gradual adaptation to ground changes.

10.2.4 Manual+Tracking Mode

Combines manual preset with auto-tracking for optimal performance in variable ground.

10.3 Vector Subtraction with Offset

Ground balance subtracts baseline I/Q vector from measured vector. User-adjustable offset (±50) rotates baseline by ±45° for fine-tuning:

```
// Apply offset rotation
val offsetRadians = (offset / 50.0) * (π/4)
val rotatedI = baseI * cos(offset) - baseQ * sin(offset)
val rotatedQ = baseI * sin(offset) + baseQ * cos(offset)

// Subtract
val newI = currentI - rotatedI
val newQ = currentQ - rotatedQ

// Recalculate amplitude/phase
val amplitude = sqrt(newI² + newQ²)
val phase = atan2(newQ, newI)
```

10.4 Freeze Detection

Critical feature: when amplitude exceeds threshold (0.3), stop tracking. Otherwise, auto-tracking will try to null out the target itself!

10.5 Summary

Ground balance is essential for real-world metal detection. The FCMD multi-mode approach provides flexibility for different soils and user skill levels.

Chapter 11

Depth Estimation Techniques

11.1 Why Depth Estimation is Hard

Signal strength ∝ $1/r^3$, but we don't know target size. A large shallow target looks identical to a small deep target. Depth estimation must account for this fundamental ambiguity.

11.2 Multi-Factor Approach

11.2.1 Signal Amplitude

```
AmplitudeFactor = 1 / Amplitude^0.35
```

11.2.2 Frequency Ratio (Skin Effect)

```
FrequencyRatio = LowFreqAmplitude / HighFreqAmplitude
```

Deep targets attenuate high frequencies more → ratio > 1.5

11.2.3 Size Normalization from VDI

Use target type to estimate expected size:

- HIGH_CONDUCTOR (copper/silver): 1.5× (large)
- GOLD_RANGE: 1.0× (medium)
- LOW_CONDUCTOR (foil): 0.8× (small)

11.3 Category-Based Estimation

Rather than claiming "6.2 inches," FCMD returns honest categories:

- ●●●● SURFACE (0-2")
- ●●●○ SHALLOW (2-4")
- ●●○○ MEDIUM (4-6")
- ●○○○ DEEP (6-8")
- ○○○○ VERY_DEEP (8"+)

11.4 Expected Accuracy

With calibration: ±1 category (±2")
Without calibration: ±1-2 categories (±4")

11.5 Calibration Procedure

1. Bury test targets at known depths (2", 4", 6", 8")
2. Measure depth factor for each
3. Adjust thresholds in code
4. Repeat for different soil types

Appendix A: Mathematics Reference

A.1 Complex Numbers and Phasors

IQ demodulation represents signals as complex numbers:

$$z = I + jQ$$
$$|z| = \sqrt{I^2 + Q^2}$$
$$\arg(z) = \operatorname{atan2}(Q, I)$$

A.2 Fourier Transform

Relationship between time and frequency domains:

$$F(\omega) = \int f(t)\, e^{-j\omega t}\, dt$$
$$f(t) = \frac{1}{2\pi} \int F(\omega)\, e^{j\omega t}\, d\omega$$

A.3 Digital Filters

A.3.1 Single-Pole IIR

$$y[n] = \alpha \cdot x[n] + (1-\alpha) \cdot y[n-1]$$
$$\text{Cutoff: } f_c \approx \frac{f_s \cdot \alpha}{2\pi}$$

A.3.2 Moving Average FIR

```
y[n] = (1/N) Σ(x[n-k]) for k=0 to N-1
```

A.4 Statistics

A.4.1 Standard Deviation

```
σ = √(Σ(x_i - μ)² / N)
```

A.4.2 Confidence Intervals

For normally distributed measurements, 95% confidence interval:

```
CI = μ ± 1.96(σ/√N)
```

Appendix B: Android Audio API Guide

B.1 AudioTrack Setup (Output)

```
audioTrack = AudioTrack.Builder()
    .setAudioAttributes(
        AudioAttributes.Builder()
            .setUsage(AudioAttributes.USAGE_MEDIA)
            .setContentType(AudioAttributes.CONTENT_TYPE_MUSIC)
            .build()
    )
    .setAudioFormat(
        AudioFormat.Builder()
            .setSampleRate(44100)
            .setChannelMask(AudioFormat.CHANNEL_OUT_STEREO)
            .setEncoding(AudioFormat.ENCODING_PCM_16BIT)
            .build()
    )
    .setBufferSizeInBytes(minBufferSize)
    .setTransferMode(AudioTrack.MODE_STREAM)
    .build()
```

B.2 AudioRecord Setup (Input)

```
audioRecord = AudioRecord.Builder()
    .setAudioSource(MediaRecorder.AudioSource.MIC)
    .setAudioFormat(
        AudioFormat.Builder()
            .setSampleRate(44100)
            .setChannelMask(AudioFormat.CHANNEL_IN_MONO)
            .setEncoding(AudioFormat.ENCODING_PCM_16BIT)
            .build()
```

```
    )
    .setBufferSizeInBytes(minBufferSize)
    .build()
```

B.3 Thread Priority

```
android.os.Process.setThreadPriority(
    android.os.Process.THREAD_PRIORITY_URGENT_AUDIO
)
```

B.4 Buffer Size Calculation

```
val minBufferSize = AudioTrack.getMinBufferSize(
    sampleRate,
    channelConfig,
    audioFormat
)

// Multiply by 1 for lowest latency
// Multiply by 2-4 for more stability
val actualBufferSize = minBufferSize * multiplier
```

B.5 Common Pitfalls

B.5.1 Buffer Underruns

If processing takes too long, audio stutters. Solution: increase buffer size or optimize processing.

B.5.2 Permission Requirements

```
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
```

B.5.3 Sample Rate Support

Not all devices support all rates. Always check with `getMinBufferSize()` - if it returns ERROR, rate is unsupported.

Appendix C: FCMD Code Reference

C.1 Project Structure

```
FCMD/
├── AudioEngine.kt           - Audio I/O management
├── IQDemodulator.kt         - Core DSP processing
├── VDICalculator.kt         - Target discrimination
├── GroundBalanceManager.kt  - Ground balance algorithms
├── DepthEstimator.kt        - Depth estimation
├── MultiToneGenerator.kt    - Transmit signal generation
├── AudioToneGenerator.kt    - Audio feedback
└── MainActivity.kt          - UI and coordination
```

C.2 Key Classes

C.2.1 AudioEngine

Purpose: Manages AudioTrack and AudioRecord, handles playback and record loops.

Key Methods:

- `start()` - Initialize audio hardware
- `stop()` - Cleanup audio resources
- `setFrequencyRange()` - Configure TX frequencies
- `setDspProcessor()` - Attach signal processing

C.2.2 IQDemodulator

Purpose: Extract amplitude/phase from received signal at each frequency.

Key Methods:

- `analyze(samples)` - Process audio buffer
- `reset()` - Clear filter states

C.2.3 VDICalculator

Purpose: Calculate VDI and classify targets.

Key Methods:

- `calculateVDI(analysis)` - Main VDI calculation
- `getTargetDescription()` - Human-readable output

C.3 Data Flow

```
AudioRecord → FloatArray samples
    ↓
IQDemodulator.analyze(samples) → List<ToneAnalysis>
    ↓
GroundBalanceManager.applyGroundBalance() → balanced List<ToneAnalysis>
    ↓
VDICalculator.calculateVDI() → VDIResult
    ↓
DepthEstimator.estimateDepth() → VDIResult with DepthEstimate
    ↓
MainActivity callback → UI update
```

C.4 Performance Characteristics

Metric	Typical Value
Sample Rate	44,100 Hz
Buffer Size	1920 samples (device-dependent)
Callback Rate	23.2 Hz

Latency	43.5 ms
Update Rate	30 Hz (configurable, max ~23 Hz)
CPU Usage	15-25% single core

References and Bibliography

Foundational Physics

[1] Jackson, J.D. (1999). *Classical Electrodynamics, 3rd Edition*. Wiley. ISBN 978-0-471-30932-1.

[2] Griffiths, D.J. (2017). *Introduction to Electrodynamics, 4th Edition*. Cambridge University Press. ISBN 978-1-108-42041-9.

[3] Hayt, W.H., Buck, J.A. (2019). *Engineering Electromagnetics, 9th Edition*. McGraw-Hill. ISBN 978-1-260-02935-2.

Metal Detection Theory

[4] Candy, B.H. (1993). "A Pulsed Induction Metal Detector." *IEEE Transactions on Geoscience and Remote Sensing*, 31(4), 809-819.

[5] Nelson, C.V., et al. (1990). "Wide Bandwidth Time-Domain Electromagnetic Sensor for Metal Target Classification." *IEEE Transactions on Geoscience and Remote Sensing*, 28(5), 892-898.

[6] Tholhuijsen, R. (2010). "Metal detector technology and application." *Proceedings of the SPIE*, 7664.

Digital Signal Processing

[7] Oppenheim, A.V., Schafer, R.W. (2009). *Discrete-Time Signal Processing, 3rd Edition*. Pearson. ISBN 978-0-13-198842-2.

[8] Lyons, R.G. (2011). *Understanding Digital Signal Processing, 3rd Edition*. Pearson. ISBN 978-0-13-702741-5.

[9] Smith, S.W. (1997). *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing. ISBN 978-0-9660176-3-2.

IQ Demodulation

[10] Rice, M., (2009). *Digital Communications: A Discrete-Time Approach*. Pearson. ISBN 978-0-13-030497-1.

[11] Proakis, J.G., Salehi, M. (2007). *Digital Communications, 5th Edition*. McGraw-Hill. ISBN 978-0-07-295716-7.

Android Audio Programming

[12] Google LLC. (2024). "Android Audio Architecture." *Android Open Source Project Documentation*. https://source.android.com/devices/audio

[13] Gargenta, M., Nakamura, M. (2014). *Learning Android, 2nd Edition*. O'Reilly Media. ISBN 978-1-4493-1943-7.

[14] Superpowered Inc. (2024). "Low Latency Audio on Android." *Technical White Paper*. https://superpowered.com/android-audio-latency

Embedded Real-Time Systems

[15] Liu, J.W.S. (2000). *Real-Time Systems*. Prentice Hall. ISBN 978-0-13-099651-0.

[16] Buttazzo, G.C. (2011). *Hard Real-Time Computing Systems, 3rd Edition*. Springer. ISBN 978-1-4614-0676-1.

Practical Metal Detecting

[17] Garrett, C. (2013). *Modern Metal Detectors*. RAM Publishing. ISBN 978-0-915920-89-7.

[18] Lagal, S. (2008). *The Metal Detecting Bible*. Krause Publications. ISBN 978-0-89689-659-0.

Open Source Projects and Resources

[19] FCMD Project Repository. https://github.com/[your-repo]/FCMD

[20] GNU Radio Project. "Signal Processing Blocks." https://www.gnuradio.org

[21] Hackster.io. "DIY Metal Detector Projects." https://www.hackster.io/projects/tags/metal+detector

Online Resources and Tutorials

[22] DSP Related. "IQ Quadrature Demodulation Tutorial." https://www.dsprelated.com

[23] StackExchange Signal Processing. "Metal Detector Signal Processing." https://dsp.stackexchange.com

[24] Android Developers. "Build real-time audio apps on Android." https://developer.android.com/ndk/guides/audio

Standards and Specifications

[25] IEEE Standard 1241-2010. "IEEE Standard for Terminology and Test Methods for Analog-to-Digital Converters."

[26] AES17-2020. "AES Standard Method for Digital Audio Engineering - Measurement of Digital Audio Equipment."

About the FCMD Project

The Field Coil Metal Detector (FCMD) project demonstrates that professional-grade metal detection is possible using consumer smartphone hardware and

open-source software. By leveraging modern Android devices' powerful processors and high-quality audio subsystems, FCMD achieves multi-frequency IQ demodulation, VDI discrimination, ground balance, and depth estimation in real-time.

This book has covered the journey from electromagnetic theory to working code, providing both the "why" (physics and signal processing theory) and the "how" (practical Android implementation). Whether you're building your own detector, learning about DSP, or exploring embedded real-time systems, the principles and techniques presented here are widely applicable.

Further Learning

Experiment with the FCMD codebase. Try different frequency ranges, modify the VDI algorithm, implement new ground balance strategies. The beauty of software-based detection is that experimentation costs nothing but time.

Happy detecting!