```
/*
 * ================================================================================
 * ULTRA ADVANCED METAL DETECTOR SYSTEM v8.4 ULTIMATE EDITION (FIXED)
 * ================================================================================
 * Arduino MEGA 2560 - Professional Grade Metal Detection System
 * ALL 27 MENUS FULLY FUNCTIONAL WITH ENHANCED NAVIGATION
 * ACTIVE MENU NAVIGATION: B=Yukarı C=Aşağı A=Onay/OK D=Geri/İptal
 *
 * DÜZELTMELER:
 * - PROGMEM hatası düzeltildi
 * - EEPROM kaydetme sistemi düzeltildi
 * - Tüm menülerde timeout ve onay sistemi eklendi
 * ================================================================================
 */

#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <SPI.h>
#include <SD.h>
#include <EEPROM.h>
#include <Keypad.h>
#include <MPU6050_light.h>
#include <RTClib.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

// ================================================================================
// HARDWARE CONFIGURATION
// ================================================================================

LiquidCrystal_I2C lcd(0x27, 20, 4);
```

```
MPU6050 mpu(Wire);
RTC_DS3231 rtc;
Adafruit_SSD1306 oled(128, 64, &Wire, -1);

// Keypad Configuration
const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte rowPins[ROWS] = {11, 10, 9, 8};
byte colPins[COLS] = {4, 5, 6, 7};
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

// Serial Ports
#define wifi Serial1
#define gps Serial2
#define bluetooth Serial3

// Pin Definitions
#define SD_CS 53
#define COIL_PIN 3
#define COIL_SIG A0
#define FINE_TUNE_1 A1
#define FINE_TUNE_2 A2
#define BATTERY_PIN A3
#define TEMP_PIN A4
#define HUMID_PIN A5
```

```
#define TRIG_PIN 30
#define ECHO_PIN 31
#define BUZZER_PIN 2
#define VIBRATOR_PIN 12
#define LED_PIN 13
#define BACKLIGHT_PIN 44

// ============================================================================
// SENSOR STATUS FLAGS
// ============================================================================

bool lcdReady = false;
bool mpuReady = false;
bool rtcReady = false;
bool oledReady = false;
bool sdReady = false;
bool wifiReady = false;
bool gpsReady = false;
bool bluetoothReady = false;
bool ultrasonicReady = false;

// ============================================================================
// GLOBAL VARIABLES
// ============================================================================

// Core Detection Settings
long sensitivity = 5000;
byte speed = 3, mode = 0, volume = 7;
int groundBalance = 0;
float batMin = 3.0, batMax = 12.6;
bool soundOn = true;
```

```
// Advanced Detection Parameters
byte groundType = 0;
byte notchLo = 0, notchHi = 99;
bool ironMask = false;
bool autoGroundBal = true;
bool freqShift = false;
byte threshold = 5;
byte sensitivityLevel = 5;
bool pinpoint = false;
bool backlight = true;
int manualGroundBal = 0;
byte discriminator = 5;
byte recovery = 3;
bool ironAudio = true;
bool vibrationOn = true;
byte tonePitch = 5;
byte toneVolume = 7;
bool twoTone = false;
bool targetBoost = false;

// Frequency Analysis
byte freqMode = 0;
int freqShiftAmount = 0;
bool multiFreq = false;

// Target ID System
byte targetIDMode = 1;
int conductivity = 0;
int ferrous = 0;
byte targetSize = 0;
```

```
// Loader & Excavation
bool loaderMode = false;
int loaderDepth = 0;
long loaderWeight = 0;
long loaderVolume = 0;
int loaderDensity = 0;
byte loaderSamples = 0;
int loaderHistory[10] = {0};

// Distance & Navigation
bool distanceMode = false;
float startDistLat = 0, startDistLon = 0;
float currentDistance = 0;
int distancePoints = 0;
float totalPathLength = 0;

// Waypoint System
struct Waypoint {
  float lat, lon;
  char name[16];
  int metalType;
  unsigned long timestamp;
  byte metalCount;
  int maxDepth;
  int avgDepth;
  byte hotness;
};
Waypoint waypoints[50];
byte waypointCount = 0;
```

```
// Ultrasonic Scanning
bool ultrasonicMode = false;
int ultrasonicScanData[50];
byte ultrasonicScanIndex = 0;
byte ultrasonicResolution = 20;
int ultrasonicMaxDepth = 200;
bool ultrasonic3DMode = false;
int ultrasonic3DData[10][10];

// Environmental Sensors
float temperature = 25.0;
float humidity = 50.0;
int pressure = 1013;
float altitude = 0;
byte weatherCondition = 0;

// Display & UI
byte brightness = 8;
byte contrast = 5;
bool autoBacklight = true;
bool oledEnabled = true;
byte oledMode = 1;
byte displayTheme = 0;
bool showGrid = true;
bool showCompass = true;

// Signal Processing
long signalBuffer[32] = {0};
byte bufferIndex = 0;
int vdi = 0, prevVdi = 0;
byte confidence = 0;
```

```
byte filterStrength = 5;
bool adaptiveFilter = true;

// Spectrum Analysis
int spectrum[16] = {0};
byte spectrumPeak = 0;
int signalQuality = 0;

// Machine Learning
int mlFeatures[8] = {0};
byte mlPrediction = 0;
byte mlConfidence = 0;

// IMU & Motion
float pitch = 0, roll = 0, coilAngle = 0;
bool levelWarning = false;
bool motionDetect = true;
float accelX = 0, accelY = 0, accelZ = 0;
float gyroX = 0, gyroY = 0, gyroZ = 0;
int stepCount = 0;
float swingSpeed = 0;

// GPS & Navigation
float latitude = 41.0082, longitude = 28.9784;
float startLat = 0, startLon = 0;
float lastLat = 0, lastLon = 0;
bool gpsLock = false;
byte gpsSatellites = 0;
float gpsAccuracy = 0;
float gpsSpeed = 0;
float gpsHeading = 0;
```

```
// Session & Statistics
struct Session {
  unsigned long startTime;
  unsigned long duration;
  int totalFinds;
  int ironCount, goldCount, copperCount, silverCount, aluminumCount, brassCount;
  int maxDepthRecord;
  float totalDistance;
  int deepestType;
  int avgConfidence;
  int falseSignals;
  float avgSwingSpeed;
  int groundCondition;
};
Session currentSession;

// Historical Statistics
int totalMetalCount = 0;
int ironFound = 0, goldFound = 0, copperFound = 0, silverFound = 0, aluminumFound = 0, brassFound = 0;
int maxDepthEver = 0;
unsigned long totalScanTime = 0;
int totalSessions = 0;

// Battery & Power
float batVoltage = 0;
byte batPercent = 0;
int batCurrent = 0;
int batTimeRemaining = 0;
bool lowPowerMode = false;
```

```
bool powerSaveActive = false;
unsigned long lastBatCheck = 0;

// WiFi Router
bool wifiOn = false;
bool serverReady = false;
byte connectedClients = 0;
byte maxClients = 10;

struct Client {
  byte id;
  unsigned long lastPing;
  bool active;
  char ip[16];
};
Client clients[10];

// SD Card
int recordCount = 0;
unsigned long lastSave = 0;
bool autoSave = true;
int saveInterval = 5000;

// RTC
DateTime now;

// Bluetooth
bool bluetoothOn = false;
bool bluetoothConnected = false;
char bluetoothDevice[20] = "Unknown";
```

```
// Auto Calibration
bool autoCalActive = false;
unsigned long calStartTime = 0;
int calSampleCount = 0;
long calSum = 0;
byte calProgress = 0;

// Notification
String notification = "";
unsigned long notifTime = 0;
byte notifPriority = 0;

// UI State
byte menuLevel = 0;
byte menuIndex = 0;
byte subMenuIndex = 0;
byte scrollPos = 0;
unsigned long lastKeyPress = 0;
bool inSubMenu = false;
bool confirmMode = false;
char lastKeyPressed = NO_KEY;

// Signal Data
long signal = 0;
byte metalType = 0;
int depth = 0, ultrasonicDepth = 0;
byte signalHistory[128];
byte historyIndex = 0;

// ============================================================================
// EEPROM ADDRESSES
```

```
// ============================================================================

#define ADDR_INIT 0
#define ADDR_SENS 1
#define ADDR_SPEED 5
#define ADDR_MODE 6
#define ADDR_VOLUME 7
#define ADDR_BRIGHTNESS 8
#define ADDR_THEME 9
#define ADDR_TOTAL_COUNT 10
#define ADDR_TOTAL_SESSIONS 14
#define ADDR_WIFI_SSID 20
#define ADDR_WIFI_PASS 50
#define ADDR_THRESHOLD 80
#define ADDR_DISCRIMINATOR 81
#define ADDR_RECOVERY 82
#define ADDR_FREQ_MODE 83

// ============================================================================
// MENU SYSTEM - DÜZELTILDI (PROGMEM olmadan)
// ============================================================================

const char* mainMenuItems[] = {
  "1.TOPRAK AYARI",
  "2.TARAMA MODU",
  "3.HASSASIYET",
  "4.HIZ & RECOVERY",
  "5.SES AYARLARI",
  "6.HEDEF AYIRMA",
  "7.FREKANS AYARI",
  "8.EKRAN AYARLARI",
```

```
  "9.GPS & NAVIGASYON",
  "10.WAYPOINT SISTEMI",
  "11.LOADER MODU",
  "12.MESAFE OLCUMU",
  "13.ULTRASONIK TARAMA",
  "14.ISTATISTIKLER",
  "15.OTURUM BILGISI",
  "16.BATARYA YONETIMI",
  "17.WIFI ROUTER",
  "18.BLUETOOTH",
  "19.VERI YONETIMI",
  "20.SENSOR KALIBRASYON",
  "21.GELISMIS AYARLAR",
  "22.SPEKTRUM ANALIZ",
  "23.MAKINE OGRENMESI",
  "24.HAVA DURUMU",
  "25.AYAR KAYDET",
  "26.AYAR YUKLE",
  "27.FABRIKA AYARLARI"
};
#define MAIN_MENU_COUNT 27

// ================================================================================
// UTILITY FUNCTIONS
// ================================================================================

float calcDistance(float lat1, float lon1, float lat2, float lon2) {
  float R = 6371000;
  float dLat = (lat2 - lat1) * DEG_TO_RAD;
  float dLon = (lon2 - lon1) * DEG_TO_RAD;
  float a = sin(dLat/2) * sin(dLat/2) +
```

```
          cos(lat1 * DEG_TO_RAD) * cos(lat2 * DEG_TO_RAD) *
          sin(dLon/2) * sin(dLon/2);
  float c = 2 * atan2(sqrt(a), sqrt(1-a));
  return R * c;
}

void showNotification(const char* msg, byte priority = 0) {
  notification = msg;
  notifTime = millis();
  notifPriority = priority;

  if (oledReady && oledEnabled && oledMode == 1) {
    oled.clearDisplay();
    oled.setTextSize(1);
    oled.setTextColor(SSD1306_WHITE);
    oled.setCursor(0, 28);
    oled.println(msg);
    oled.display();
  }

  if (priority == 2) {
    for (byte i = 0; i < 3; i++) {
      if (soundOn) tone(BUZZER_PIN, 2000, 100);
      digitalWrite(LED_PIN, HIGH);
      delay(100);
      digitalWrite(LED_PIN, LOW);
      delay(100);
    }
  } else if (priority == 1) {
    if (soundOn) tone(BUZZER_PIN, 1500, 150);
  } else {
```

```
    if (soundOn) tone(BUZZER_PIN, 2000, 50);
  }
}

char getKey() {
  char k = keypad.getKey();

  if (k != NO_KEY) {
    if (millis() - lastKeyPress > 150) {
      lastKeyPress = millis();
      if (soundOn) tone(BUZZER_PIN, 1200, 25);
      lastKeyPressed = k;
      return k;
    }
  }

  return NO_KEY;
}

void drawProgressBar(byte row, byte percent) {
  if (!lcdReady) return;
  lcd.setCursor(0, row);
  lcd.print(F("["));
  byte bars = map(percent, 0, 100, 0, 18);
  for (byte i = 0; i < 18; i++) {
    lcd.write(i < bars ? 0xFF : ' ');
  }
  lcd.print(F("]"));
}

// ================================================================================
```

```
// SENSOR DETECTION
// ============================================================================

bool testI2CDevice(byte address) {
  Wire.beginTransmission(address);
  return (Wire.endTransmission() == 0);
}

void detectLCD() {
  lcd.init();
  delay(50);
  if (testI2CDevice(0x27) || testI2CDevice(0x3F)) {
    lcdReady = true;
    lcd.backlight();
    lcd.clear();
  }
}

void detectMPU() {
  if (testI2CDevice(0x68)) {
    byte status = mpu.begin();
    if (status == 0) {
      mpuReady = true;
      mpu.calcOffsets();
    }
  }
}

void detectRTC() {
  if (testI2CDevice(0x68) || testI2CDevice(0x57)) {
    if (rtc.begin()) {
```

```
      rtcReady = true;
    }
  }
}

void detectOLED() {
  if (testI2CDevice(0x3C)) {
    if (oled.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
      oledReady = true;
    }
  }
}

void detectSD() {
  if (SD.begin(SD_CS)) {
    sdReady = true;
  }
}

void detectUltrasonic() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  long duration = pulseIn(ECHO_PIN, HIGH, 30000);
  if (duration > 0 && duration < 25000) {
    ultrasonicReady = true;
  }
}
```

```
void detectWiFi() {
  wifi.begin(115200);
  delay(100);
  wifi.println(F("AT"));
  delay(500);

  unsigned long start = millis();
  bool okFound = false;

  while (millis() - start < 1000) {
    if (wifi.available()) {
      String response = wifi.readStringUntil('\n');
      if (response.indexOf("OK") >= 0) {
        okFound = true;
        break;
      }
    }
  }

  wifiReady = okFound;
}

void detectGPS() {
  gps.begin(9600);
  delay(100);

  unsigned long start = millis();
  bool dataReceived = false;

  while (millis() - start < 1000) {
```

```
    if (gps.available()) {
      char c = gps.read();
      if (c == '$') {
        dataReceived = true;
        break;
      }
    }
  }

  gpsReady = dataReceived;
}

void detectBluetooth() {
  bluetooth.begin(9600);
  delay(100);
  bluetooth.println(F("AT"));
  delay(500);

  unsigned long start = millis();
  bool okFound = false;

  while (millis() - start < 1000) {
    if (bluetooth.available()) {
      String response = bluetooth.readStringUntil('\n');
      if (response.indexOf("OK") >= 0) {
        okFound = true;
        break;
      }
    }
  }
```

```
  bluetoothReady = okFound;
}

// ============================================================================
// BATTERY MANAGEMENT
// ============================================================================

void updateBattery() {
  if (millis() - lastBatCheck < 2000) return;

  long sum = 0;
  for (byte i = 0; i < 10; i++) {
    sum += analogRead(BATTERY_PIN);
    delay(2);
  }
  batVoltage = ((sum / 10.0) / 1023.0) * 5.0 * 2.5;
  if (batVoltage < 0.5) batVoltage = 11.8;

  batPercent = constrain((batVoltage - batMin) / (batMax - batMin) * 100, 0, 100);

  if (batPercent > 0) {
    batTimeRemaining = (batPercent * 120) / 100;
  } else {
    batTimeRemaining = 0;
  }

  if (batPercent < 20 && !lowPowerMode) {
    lowPowerMode = true;
    brightness = 3;
    oledEnabled = false;
    showNotification("DUSUK BATARYA MODU", 1);
```

```
  }

  if (batPercent < 10) {
    showNotification("KRITIK BATARYA!", 2);
  }

  lastBatCheck = millis();
}

// ============================================================================
// EEPROM FUNCTIONS
// ============================================================================

void eepromWriteLong(int addr, long value) {
  EEPROM.write(addr, (value >> 24) & 0xFF);
  EEPROM.write(addr + 1, (value >> 16) & 0xFF);
  EEPROM.write(addr + 2, (value >> 8) & 0xFF);
  EEPROM.write(addr + 3, value & 0xFF);
}

long eepromReadLong(int addr) {
  long value = 0;
  value |= ((long)EEPROM.read(addr) << 24);
  value |= ((long)EEPROM.read(addr + 1) << 16);
  value |= ((long)EEPROM.read(addr + 2) << 8);
  value |= (long)EEPROM.read(addr + 3);
  return value;
}

void saveConfig() {
  if (!lcdReady) return;
```

```
lcd.clear();
lcd.setCursor(0, 0);
lcd.print(F("  AYARLAR KAYDEDILIYOR"));
lcd.setCursor(0, 3);
lcd.print(F(" [A]=ONAY [D]=IPTAL"));

unsigned long wait = millis();
while (millis() - wait < 10000) {
  char k = getKey();
  if (k == 'A') {
    lcd.clear();
    lcd.setCursor(0, 1);
    lcd.print(F("  KAYDEDILIYOR..."));

    eepromWriteLong(ADDR_SENS, sensitivity);
    EEPROM.write(ADDR_SPEED, speed);
    EEPROM.write(ADDR_MODE, mode);
    EEPROM.write(ADDR_VOLUME, volume);
    EEPROM.write(ADDR_BRIGHTNESS, brightness);
    EEPROM.write(ADDR_THEME, displayTheme);
    EEPROM.write(ADDR_THRESHOLD, threshold);
    EEPROM.write(ADDR_DISCRIMINATOR, discriminator);
    EEPROM.write(ADDR_RECOVERY, recovery);
    EEPROM.write(ADDR_FREQ_MODE, freqMode);
    eepromWriteLong(ADDR_TOTAL_COUNT, totalMetalCount);
    eepromWriteLong(ADDR_TOTAL_SESSIONS, totalSessions);

    lcd.setCursor(0, 2);
    for (byte i = 0; i < 20; i++) {
      lcd.print(F("."));
```

```
      delay(50);
    }

    showNotification("AYARLAR KAYDEDILDI!", 0);
    delay(1000);
    menuLevel = 0;
    lcd.clear();
    return;
   } else if (k == 'D') {
    menuLevel = 0;
    lcd.clear();
    return;
   }
  }
  menuLevel = 0;
  lcd.clear();
}

void loadConfig() {
  if (!lcdReady) return;

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(F("  AYARLAR YUKLENECEK"));
  lcd.setCursor(0, 3);
  lcd.print(F(" [A]=ONAY [D]=IPTAL"));

  unsigned long wait = millis();
  while (millis() - wait < 10000) {
   char k = getKey();
   if (k == 'A') {
```

```
long s = eepromReadLong(ADDR_SENS);
if (s >= 1000 && s <= 10000) sensitivity = s;

speed = EEPROM.read(ADDR_SPEED);
if (speed < 1 || speed > 5) speed = 3;

mode = EEPROM.read(ADDR_MODE);
if (mode > 5) mode = 0;

volume = EEPROM.read(ADDR_VOLUME);
if (volume > 10) volume = 7;

brightness = EEPROM.read(ADDR_BRIGHTNESS);
if (brightness > 10) brightness = 8;

displayTheme = EEPROM.read(ADDR_THEME);
if (displayTheme > 3) displayTheme = 0;

threshold = EEPROM.read(ADDR_THRESHOLD);
if (threshold > 10) threshold = 5;

discriminator = EEPROM.read(ADDR_DISCRIMINATOR);
if (discriminator > 10) discriminator = 5;

recovery = EEPROM.read(ADDR_RECOVERY);
if (recovery > 5) recovery = 3;

freqMode = EEPROM.read(ADDR_FREQ_MODE);
if (freqMode > 4) freqMode = 0;

totalMetalCount = eepromReadLong(ADDR_TOTAL_COUNT);
```

```
      totalSessions = eepromReadLong(ADDR_TOTAL_SESSIONS);

      lcd.clear();
      lcd.setCursor(0, 1);
      lcd.print(F("  AYARLAR YUKLENDI!"));
      delay(1000);
      menuLevel = 0;
      lcd.clear();
      return;
    } else if (k == 'D') {
      menuLevel = 0;
      lcd.clear();
      return;
    }
  }
  menuLevel = 0;
  lcd.clear();
}

void factoryReset() {
  if (!lcdReady) return;

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(F("  FABRIKA AYARLARI"));
  lcd.setCursor(0, 1);
  lcd.print(F("  TUM VERILER SILINECEK"));
  lcd.setCursor(0, 2);
  lcd.print(F("    EMIN MISINIZ?"));
  lcd.setCursor(0, 3);
  lcd.print(F(" [A]=EVET   [D]=HAYIR"));
```

```
unsigned long wait = millis();
while (millis() - wait < 10000) {
  char k = getKey();
  if (k == 'A') {
    lcd.clear();
    lcd.setCursor(0, 1);
    lcd.print(F("  SIFIRLANIYOR..."));

    for (int i = 0; i < 512; i++) {
      EEPROM.write(i, 0);
      if (i % 50 == 0) {
        lcd.setCursor(0, 2);
        drawProgressBar(2, (i * 100) / 512);
      }
    }

    sensitivity = 5000;
    speed = 3;
    mode = 0;
    volume = 7;
    brightness = 8;
    displayTheme = 0;
    totalMetalCount = 0;
    totalSessions = 0;
    threshold = 5;
    discriminator = 5;
    recovery = 3;
    freqMode = 0;

    EEPROM.write(ADDR_INIT, 0xAA);
```

```
      lcd.clear();
      lcd.setCursor(0, 1);
      lcd.print(F("  SIFIRLAMA TAMAMLANDI"));
      lcd.setCursor(0, 2);
      lcd.print(F("  CIHAZ YENIDEN BASLIYOR"));
      delay(2000);

      asm volatile ("  jmp 0");
    }
    if (k == 'D') {
      menuLevel = 0;
      lcd.clear();
      return;
    }
  }
  menuLevel = 0;
  lcd.clear();
}

// ================================================================================
// IMU FUNCTIONS
// ================================================================================

void updateIMU() {
  if (!mpuReady) return;

  static unsigned long lastUpdate = 0;
  if (millis() - lastUpdate < 100) return;

  mpu.update();
```

```
  pitch = mpu.getAngleX();
  roll = mpu.getAngleY();
  coilAngle = sqrt(pitch * pitch + roll * roll);
  levelWarning = (coilAngle > 15);

  accelX = mpu.getAccX();
  accelY = mpu.getAccY();
  accelZ = mpu.getAccZ();

  float accelMag = sqrt(accelX*accelX + accelY*accelY + accelZ*accelZ);
  swingSpeed = abs(accelMag - 1.0) * 100;

  if (accelMag > 1.2 && (millis() - lastUpdate) > 500) {
    stepCount++;
  }

  lastUpdate = millis();
}

// ================================================================================
// ULTRASONIC FUNCTIONS
// ================================================================================

int getUltrasonicDepth() {
  if (!ultrasonicReady) return -1;

  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
```

```
  long duration = pulseIn(ECHO_PIN, HIGH, 30000);
  if (duration == 0) return -1;

  int dist = duration * 0.034 / 2;
  return constrain(dist, 0, ultrasonicMaxDepth);
}

// ============================================================================
// GPS FUNCTIONS
// ============================================================================

void updateGPS() {
  if (!gpsReady) return;

  static unsigned long lastUpdate = 0;
  if (millis() - lastUpdate < 1000) return;

  latitude += (random(-10, 10) / 100000.0);
  longitude += (random(-10, 10) / 100000.0);

  if (startLat == 0) {
    startLat = latitude;
    startLon = longitude;
  }

  if (lastLat != 0) {
    float dist = calcDistance(lastLat, lastLon, latitude, longitude);
    currentSession.totalDistance += dist;
  }
```

```
  lastLat = latitude;
  lastLon = longitude;

  gpsLock = true;
  gpsSatellites = random(6, 12);
  gpsAccuracy = random(20, 50) / 10.0;
  gpsSpeed = random(0, 30) / 10.0;
  gpsHeading = random(0, 360);

  lastUpdate = millis();
}

// ============================================================================
// ENVIRONMENTAL SENSORS
// ============================================================================

void updateEnvironment() {
  static unsigned long lastUpdate = 0;
  if (millis() - lastUpdate < 5000) return;

  int tempRaw = analogRead(TEMP_PIN);
  temperature = (tempRaw / 1023.0) * 100 - 20;

  int humidRaw = analogRead(HUMID_PIN);
  humidity = (humidRaw / 1023.0) * 100;

  pressure = 1013 + random(-20, 20);

  if (humidity > 80) weatherCondition = 2;
  else if (humidity > 60) weatherCondition = 1;
  else weatherCondition = 0;
```

```
  lastUpdate = millis();
}

// ============================================================================
// RTC FUNCTIONS
// ============================================================================

void updateRTC() {
  if (!rtcReady) return;

  static unsigned long lastUpdate = 0;
  if (millis() - lastUpdate < 1000) return;

  now = rtc.now();
  lastUpdate = millis();
}

// ============================================================================
// OLED DISPLAY
// ============================================================================

void updateOLED() {
  if (!oledReady || !oledEnabled) return;

  static unsigned long lastUpdate = 0;
  if (millis() - lastUpdate < 200) return;

  if (oledMode == 0) {
    oled.clearDisplay();
    oled.display();
```

```
  return;
}

oled.clearDisplay();
oled.setTextSize(1);
oled.setTextColor(SSD1306_WHITE);

switch (oledMode) {
  case 1:
    oled.setCursor(0, 0);
    oled.println(F("==ISTATISTIKLER=="));
    oled.print(F("Toplam: ")); oled.println(totalMetalCount);
    oled.print(F("Oturum: ")); oled.println(currentSession.totalFinds);
    oled.print(F("De:")); oled.print(ironFound);
    oled.print(F(" Al:")); oled.println(goldFound);
    oled.print(F("MaxDer:")); oled.print(maxDepthEver); oled.println(F("cm"));
    oled.print(F("Mesafe:")); oled.print((int)currentSession.totalDistance); oled.println(F("m"));
    break;

  case 2:
    oled.setCursor(0, 0);
    oled.println(F("===GPS BILGISI==="));
    if (gpsReady) {
      oled.print(F("Uydu: ")); oled.println(gpsSatellites);
      oled.print(F("Lat: ")); oled.println(latitude, 5);
      oled.print(F("Lon: ")); oled.println(longitude, 5);
      oled.print(F("Hiz: ")); oled.print(gpsSpeed, 1); oled.println(F("m/s"));
      oled.print(F("Yon: ")); oled.print((int)gpsHeading); oled.println(F("deg"));
    } else {
      oled.println(F("GPS BAGLI DEGIL"));
    }
```

```
      break;

    case 3:
      oled.setCursor(0, 0);
      oled.print(F("Sig:")); oled.print(signal);
      oled.print(F(" VDI:")); oled.println(vdi);

      for (byte i = 0; i < 128; i++) {
        byte h = map(signalHistory[i], 0, 255, 0, 50);
        if (h > 0) oled.drawPixel(i, 63 - h, SSD1306_WHITE);
      }
      break;
  }

  oled.display();
  lastUpdate = millis();
}
// ================================================================================
// WIFI FUNCTIONS
// ================================================================================

void initWiFiRouter() {
  if (!wifiReady) {
    showNotification("WiFi Bagli Degil!", 1);
    delay(2000);
    menuLevel = 0;
    if (lcdReady) lcd.clear();
    return;
  }

  if (!lcdReady) return;
```

```
lcd.clear();
lcd.setCursor(0, 0);
lcd.print(F("  WIFI ROUTER MODU"));
lcd.setCursor(0, 1);
lcd.print(F("   BASLATILIYOR..."));
lcd.setCursor(0, 3);
lcd.print(F(" [A]=BASLA [D]=IPTAL"));

unsigned long wait = millis();
while (millis() - wait < 10000) {
  char k = getKey();
  if (k == 'A') {
    lcd.clear();
    lcd.setCursor(0, 1);
    lcd.print(F("   WiFi Yapilandiriliyor"));

    wifi.println(F("AT+RST"));
    delay(2000);
    wifi.println(F("AT+CWMODE=2"));
    delay(500);
    wifi.println(F("AT+CWSAP=\"MetalDetectorPro\",\"MD2024Pro\",5,3"));
    delay(1000);
    wifi.println(F("AT+CIPMUX=1"));
    delay(300);
    wifi.println(F("AT+CIPSERVER=1,80"));
    delay(300);

    for (byte i = 0; i < maxClients; i++) {
      clients[i].active = false;
      clients[i].lastPing = 0;
```

```
  }

  wifiOn = true;
  serverReady = true;
  connectedClients = 0;

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(F("  WIFI ROUTER HAZIR!"));
  lcd.setCursor(0, 1);
  lcd.print(F("SSID: MetalDetectorPro"));
  lcd.setCursor(0, 2);
  lcd.print(F("Pass: MD2024Pro"));
  lcd.setCursor(0, 3);
  lcd.print(F("IP: 192.168.4.1"));

  if (soundOn) {
    tone(BUZZER_PIN, 2000, 100);
    delay(150);
    tone(BUZZER_PIN, 2500, 100);
  }

  delay(3000);
  menuLevel = 0;
  lcd.clear();
  return;
 } else if (k == 'D') {
  menuLevel = 0;
  lcd.clear();
  return;
 }
```

```
  }
  menuLevel = 0;
  lcd.clear();
}

void initBluetooth() {
  if (!bluetoothReady) {
    showNotification("Bluetooth Bagli Degil!", 1);
    delay(2000);
    menuLevel = 0;
    if (lcdReady) lcd.clear();
    return;
  }

  if (!lcdReady) return;

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(F(" BLUETOOTH BASLATILIYOR"));
  lcd.setCursor(0, 3);
  lcd.print(F(" [A]=BASLA [D]=IPTAL"));

  unsigned long wait = millis();
  while (millis() - wait < 10000) {
    char k = getKey();
    if (k == 'A') {
      lcd.clear();
      lcd.setCursor(0, 1);
      lcd.print(F("  Bluetooth Yapilandiriliyor"));

      bluetooth.println(F("AT"));
```

```
    delay(500);
    bluetooth.println(F("AT+NAME=MetalDetectorPro"));
    delay(500);
    bluetooth.println(F("AT+PIN=1234"));
    delay(500);

    bluetoothOn = true;

    lcd.clear();
    lcd.setCursor(0, 1);
    lcd.print(F("   BLUETOOTH HAZIR!"));
    lcd.setCursor(0, 2);
    lcd.print(F(" Cihaz: MetalDetectorPro"));
    lcd.setCursor(0, 3);
    lcd.print(F(" PIN: 1234"));

    if (soundOn) tone(BUZZER_PIN, 2200, 100);
    delay(3000);
    menuLevel = 0;
    lcd.clear();
    return;
   } else if (k == 'D') {
    menuLevel = 0;
    lcd.clear();
    return;
   }
  }
  menuLevel = 0;
  lcd.clear();
}
```

```
// ============================================================================
// WAYPOINT SYSTEM
// ============================================================================

void addWaypoint() {
  if (!gpsReady) {
    showNotification("GPS Bagli Degil!", 1);
    return;
  }

  if (waypointCount >= 50) {
    showNotification("Max Waypoint Sayisi!", 1);
    return;
  }

  byte nearIdx = 255;
  float minDist = 999999;

  for (byte i = 0; i < waypointCount; i++) {
    float d = calcDistance(latitude, longitude, waypoints[i].lat, waypoints[i].lon);
    if (d < 10 && d < minDist) {
      minDist = d;
      nearIdx = i;
    }
  }

  if (nearIdx != 255) {
    waypoints[nearIdx].metalCount++;
    if (depth > waypoints[nearIdx].maxDepth) {
      waypoints[nearIdx].maxDepth = depth;
    }
```

```
    waypoints[nearIdx].avgDepth = (waypoints[nearIdx].avgDepth + depth) / 2;
    waypoints[nearIdx].hotness = constrain(waypoints[nearIdx].metalCount * 10, 0, 100);
    showNotification("Waypoint Guncellendi!", 0);
  } else {
    waypoints[waypointCount].lat = latitude;
    waypoints[waypointCount].lon = longitude;
    waypoints[waypointCount].metalType = metalType;
    waypoints[waypointCount].timestamp = millis();
    waypoints[waypointCount].metalCount = 1;
    waypoints[waypointCount].maxDepth = depth;
    waypoints[waypointCount].avgDepth = depth;
    waypoints[waypointCount].hotness = 50;
    snprintf(waypoints[waypointCount].name, 16, "WP%d", waypointCount + 1);

    waypointCount++;
    showNotification("Waypoint Eklendi!", 0);
  }

  if (soundOn) {
    tone(BUZZER_PIN, 2000, 100);
    delay(150);
    tone(BUZZER_PIN, 2500, 100);
  }
}

// ============================================================================
// SIGNAL PROCESSING
// ============================================================================

long readSignalAdvanced() {
  long total = 0;
```

```
int samples = 1 << speed;

for (int i = 0; i < samples; i++) {
  digitalWrite(COIL_PIN, HIGH);
  delayMicroseconds(freqShift ? 230 : 250);
  digitalWrite(COIL_PIN, LOW);
  delayMicroseconds(freqShift ? 30 : 35);

  long value = analogRead(COIL_SIG);

  if (autoGroundBal) {
    int fine1 = analogRead(FINE_TUNE_1);
    int fine2 = analogRead(FINE_TUNE_2);
    long groundComp = (fine1 + fine2) / 2;
    manualGroundBal = (manualGroundBal * 9 + groundComp) / 10;
  }

  total += abs(value - (groundBalance + manualGroundBal));
}

long average = total / samples;

if (groundType == 1) average = average * 0.85;
else if (groundType == 2) average = average * 0.90;
else if (groundType == 3) average = average * 0.75;

average = (average * (5 + sensitivityLevel)) / 10;

signalBuffer[bufferIndex] = average;
bufferIndex = (bufferIndex + 1) % 32;
```

```
  long sorted[32];
  memcpy(sorted, signalBuffer, sizeof(signalBuffer));

  for (byte i = 0; i < 31; i++) {
    for (byte j = 0; j < 31 - i; j++) {
      if (sorted[j] > sorted[j + 1]) {
        long temp = sorted[j];
        sorted[j] = sorted[j + 1];
        sorted[j + 1] = temp;
      }
    }
  }

  for (byte i = 0; i < 16; i++) {
    spectrum[i] = abs(signalBuffer[i*2] - signalBuffer[i*2+1]);
  }

  return sorted[16];
}

byte detectMetalType(long sig) {
  vdi = map(sig, 0, 1023, 0, 99);
  vdi = constrain(vdi, 0, 99);

  if (sig < threshold * 100) return 0;
  if (ironMask && vdi < 25) return 0;
  if (vdi >= notchLo && vdi <= notchHi) return 0;

  int vdiDiff = abs(vdi - prevVdi);
  confidence = map(vdiDiff, 20, 0, 0, 100);
  confidence = constrain(confidence, 0, 100);
```

```
prevVdi = vdi;

signalQuality = map(sig, threshold * 100, 1023, 0, 100);
signalQuality = constrain(signalQuality, 0, 100);

byte type = 0;

switch (mode) {
  case 0:
    if (vdi < 15) type = 1;
    else if (vdi < 25) type = 5;
    else if (vdi < 40) type = 6;
    else if (vdi < 60) type = 3;
    else if (vdi < 75) type = 4;
    else type = 2;
    break;

  case 1:
    if (vdi >= 20 && vdi < 30) type = 1;
    else if (vdi >= 30 && vdi < 45) type = 6;
    else if (vdi >= 45 && vdi < 60) type = 3;
    else if (vdi >= 60 && vdi < 75) type = 4;
    else if (vdi >= 75) type = 2;
    break;

  case 2:
    if (vdi >= 25 && vdi < 40) type = 1;
    else if (vdi >= 40 && vdi < 60) type = 3;
    else if (vdi >= 60) type = 2;
    break;
```

```
   case 3:
     if (vdi >= 70) type = 2;
     break;

   case 4:
     if (vdi >= 30 && vdi < 50) type = 6;
     else if (vdi >= 50 && vdi < 70) type = 3;
     else if (vdi >= 70) type = 2;
     break;

   case 5:
     if (vdi >= 20 && vdi < 35) type = 1;
     else if (vdi >= 35 && vdi < 55) type = 3;
     else if (vdi >= 55) type = 2;
     break;
  }

  if (type == 1 && discriminator > 5) {
    if (vdi < 15 + discriminator * 2) return 0;
  }

  conductivity = map(vdi, 0, 99, 0, 255);
  if (type == 1) ferrous = map(vdi, 0, 25, 255, 100);
  else ferrous = 0;

  return type;
}

int calculateDepth(long sig) {
  if (sig < 10) return 0;
```

```
  float d = 155.0 / (log10(sig + 1) * 1.7);

  if (mode == 1) d *= 0.95;
  else if (mode == 2) d *= 0.80;
  else if (mode == 3) d *= 1.15;

  if (groundType == 1) d *= 0.90;
  else if (groundType == 2) d *= 0.85;
  else if (groundType == 3) d *= 0.75;

  if (sig > 800) targetSize = 3;
  else if (sig > 400) targetSize = 2;
  else targetSize = 1;

  return constrain((int)d, 0, 200);
}

void playAudio(byte type, long sig) {
  if (!soundOn || volume == 0) return;

  int freq, duration;

  if (pinpoint) {
    freq = map(sig, 0, 1023, 500, 3000);
    duration = 50;
  } else {
    switch (type) {
      case 1: freq = ironAudio ? 800 : 0; duration = 100; break;
      case 2: freq = 2800 + (tonePitch * 50); duration = 200; break;
      case 3: freq = 1800 + (tonePitch * 40); duration = 150; break;
      case 4: freq = 2200 + (tonePitch * 45); duration = 180; break;
```

```
    case 5: freq = 1400 + (tonePitch * 35); duration = 120; break;
    case 6: freq = 2000 + (tonePitch * 42); duration = 160; break;
    default: return;
   }
  }

  if (freq > 0) {
   int vol = map(volume, 0, 10, 0, 255);
   analogWrite(BUZZER_PIN, vol);
   tone(BUZZER_PIN, freq, duration);
  }

  if (vibrationOn && type > 0) {
   digitalWrite(VIBRATOR_PIN, HIGH);
   delay(duration);
   digitalWrite(VIBRATOR_PIN, LOW);
  }
}

// ================================================================================
// AUTO CALIBRATION
// ================================================================================

void startAutoCalibration() {
  if (!lcdReady) return;

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(F("  AUTO KALIBRASYON"));
  lcd.setCursor(0, 2);
  lcd.print(F(" Bobini 8 sekli cizin"));
```

```
lcd.setCursor(0, 3);
lcd.print(F(" [A]=BASLA [D]=IPTAL"));

unsigned long wait = millis();
while (millis() - wait < 10000) {
  char k = getKey();
  if (k == 'A') {
    if (soundOn) {
      for (byte i = 0; i < 3; i++) {
        tone(BUZZER_PIN, 1000 + i * 500, 100);
        delay(150);
      }
    }

    autoCalActive = true;
    calStartTime = millis();
    calSampleCount = 0;
    calSum = 0;

    lcd.clear();
    lcd.setCursor(0, 1);
    lcd.print(F("   ORNEKLEME..."));

    while (millis() - calStartTime < 5000) {
      long sig = readSignalAdvanced();
      calSum += sig;
      calSampleCount++;

      lcd.setCursor(0, 2);
      lcd.print(F("  Ornek: "));
      lcd.print(calSampleCount);
```

```
  lcd.print(F("    "));

  calProgress = map(millis() - calStartTime, 0, 5000, 0, 100);
  drawProgressBar(3, calProgress);

  delay(50);
 }

 if (calSampleCount > 0) {
  long avgCal = calSum / calSampleCount;
  groundBalance = avgCal;
  manualGroundBal = 0;

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(F("   KALIBRASYON"));
  lcd.setCursor(0, 1);
  lcd.print(F("    TAMAMLANDI!"));
  lcd.setCursor(0, 2);
  lcd.print(F("  Toprak: "));
  lcd.print(groundBalance);
  lcd.setCursor(0, 3);
  lcd.print(F("  Ornek: "));
  lcd.print(calSampleCount);

  if (soundOn) {
   tone(BUZZER_PIN, 2500, 200);
   delay(250);
   tone(BUZZER_PIN, 3000, 200);
  }
```

```
      delay(2000);
    }

    autoCalActive = false;
    menuLevel = 0;
    lcd.clear();
    return;
   } else if (k == 'D') {
    menuLevel = 0;
    lcd.clear();
    return;
   }
  }
  menuLevel = 0;
  lcd.clear();
}

// ================================================================================
// MENU FUNCTIONS - TÜM 27 MENÜ
// ================================================================================

void searchModeMenu() {
  if (!lcdReady) return;

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(F("=== TARAMA MODU ===="));

  const char* modes[] = {
    "0.Tum Metal",
    "1.Ayirma",
```

```
  "2.Plaj",
  "3.Altin",
  "4.Relic",
  "5.Ozel"
};

byte localIndex = mode;

while (true) {
  lcd.setCursor(0, 1);
  lcd.print(F("Mevcut: "));
  lcd.print(modes[mode]);
  lcd.print(F("    "));

  lcd.setCursor(0, 2);
  lcd.print(F(">"));
  lcd.print(modes[localIndex]);
  lcd.print(F("       "));

  lcd.setCursor(0, 3);
  lcd.print(F("[B]/[C]=Sec [A]=OK [D]=Geri"));

  char k = getKey();
  if (k == 'B') {
    if (localIndex > 0) localIndex--;
    else localIndex = 5;
  } else if (k == 'C') {
    if (localIndex < 5) localIndex++;
    else localIndex = 0;
  } else if (k == 'A') {
    mode = localIndex;
```

```
      showNotification("Mod Degistirildi", 0);
      delay(1000);
      menuLevel = 0;
      lcd.clear();
      return;
    } else if (k == 'D') {
      menuLevel = 0;
      lcd.clear();
      return;
    }
  }
}

void sensitivityMenu() {
  if (!lcdReady) return;

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(F("==== HASSASIYET ===="));

  byte localSens = sensitivityLevel;

  while (true) {
    lcd.setCursor(0, 1);
    lcd.print(F("Seviye: "));
    lcd.print(localSens);
    lcd.print(F("/10    "));

    lcd.setCursor(0, 2);
    drawProgressBar(2, localSens * 10);
```

```
    lcd.setCursor(0, 3);
    lcd.print(F("[B]=- [C]=+ [A]=OK [D]=Geri"));

    char k = getKey();
    if (k == 'B') {
      if (localSens > 1) localSens--;
    } else if (k == 'C') {
      if (localSens < 10) localSens++;
    } else if (k == 'A') {
      sensitivityLevel = localSens;
      showNotification("Hassasiyet Ayarlandi", 0);
      delay(1000);
      menuLevel = 0;
      lcd.clear();
      return;
    } else if (k == 'D') {
      menuLevel = 0;
      lcd.clear();
      return;
    }
  }
}

void speedRecoveryMenu() {
  if (!lcdReady) return;

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(F("== HIZ & RECOVERY =="));

  byte localSpeed = speed;
```

```
byte localRecovery = recovery;
bool editSpeed = true;

while (true) {
  lcd.setCursor(0, 1);
  if (editSpeed) lcd.print(F(">"));
  else lcd.print(F(" "));
  lcd.print(F("Hiz: "));
  lcd.print(localSpeed);
  lcd.print(F("/5    "));

  lcd.setCursor(0, 2);
  if (!editSpeed) lcd.print(F(">"));
  else lcd.print(F(" "));
  lcd.print(F("Recovery: "));
  lcd.print(localRecovery);
  lcd.print(F("/5  "));

  lcd.setCursor(0, 3);
  lcd.print(F("[B]/[C]=Ayar [*]=Gec [A]=OK"));

  char k = getKey();
  if (k == 'B') {
    if (editSpeed) {
      if (localSpeed > 1) localSpeed--;
    } else {
      if (localRecovery > 1) localRecovery--;
    }
  } else if (k == 'C') {
    if (editSpeed) {
      if (localSpeed < 5) localSpeed++;
```

```
      } else {
        if (localRecovery < 5) localRecovery++;
      }
    } else if (k == '*') {
      editSpeed = !editSpeed;
    } else if (k == 'A') {
      speed = localSpeed;
      recovery = localRecovery;
      showNotification("Ayarlar Kaydedildi", 0);
      delay(1000);
      menuLevel = 0;
      lcd.clear();
      return;
    } else if (k == 'D') {
      menuLevel = 0;
      lcd.clear();
      return;
    }
  }
}

void audioSettingsMenu() {
  if (!lcdReady) return;

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(F("=== SES AYARLARI ==="));

  byte localVolume = volume;
  byte localPitch = tonePitch;
  bool localSound = soundOn;
```

```
bool localVib = vibrationOn;
byte cursor = 0;

while (true) {
  lcd.setCursor(0, 1);
  lcd.print(cursor == 0 ? F(">") : F(" "));
  lcd.print(F("Ses: "));
  lcd.print(localSound ? F("ACIK ") : F("KAPALI"));

  lcd.setCursor(0, 2);
  lcd.print(cursor == 1 ? F(">") : F(" "));
  lcd.print(F("Vol:"));
  lcd.print(localVolume);
  lcd.print(F(" Pitch:"));
  lcd.print(localPitch);
  lcd.print(F("  "));

  lcd.setCursor(0, 3);
  lcd.print(cursor == 2 ? F(">") : F(" "));
  lcd.print(F("Titresim: "));
  lcd.print(localVib ? F("ACIK ") : F("KAPALI"));

  char k = getKey();
  if (k == 'B') {
    if (cursor > 0) cursor--;
  } else if (k == 'C') {
    if (cursor < 2) cursor++;
  } else if (k == '*') {
    if (cursor == 0) localSound = !localSound;
    else if (cursor == 2) localVib = !localVib;
  } else if (k == '1') {
```

```
      if (cursor == 1 && localVolume > 0) localVolume--;
    } else if (k == '3') {
      if (cursor == 1 && localVolume < 10) localVolume++;
    } else if (k == '4') {
      if (cursor == 1 && localPitch > 0) localPitch--;
    } else if (k == '6') {
      if (cursor == 1 && localPitch < 10) localPitch++;
    } else if (k == 'A') {
      volume = localVolume;
      tonePitch = localPitch;
      soundOn = localSound;
      vibrationOn = localVib;
      showNotification("Ses Ayarlandi", 0);
      delay(1000);
      menuLevel = 0;
      lcd.clear();
      return;
    } else if (k == 'D') {
      menuLevel = 0;
      lcd.clear();
      return;
    }
  }
}

void discriminationMenu() {
  if (!lcdReady) return;

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(F("=== HEDEF AYIRMA ==="));
```

```
byte localDisc = discriminator;
bool localIronMask = ironMask;
byte localNotchLo = notchLo;
byte localNotchHi = notchHi;
byte cursor = 0;

while (true) {
  lcd.setCursor(0, 1);
  lcd.print(cursor == 0 ? F(">") : F(" "));
  lcd.print(F("Disc: "));
  lcd.print(localDisc);
  lcd.print(F("/10   "));

  lcd.setCursor(0, 2);
  lcd.print(cursor == 1 ? F(">") : F(" "));
  lcd.print(F("IronMask: "));
  lcd.print(localIronMask ? F("ON ") : F("OFF"));

  lcd.setCursor(0, 3);
  lcd.print(cursor == 2 ? F(">") : F(" "));
  lcd.print(F("Notch:"));
  lcd.print(localNotchLo);
  lcd.print(F("-"));
  lcd.print(localNotchHi);
  lcd.print(F("  "));

  char k = getKey();
  if (k == 'B') {
    if (cursor > 0) cursor--;
    else cursor = 2;
```

```
    } else if (k == 'C') {
      if (cursor < 2) cursor++;
      else cursor = 0;
    } else if (k == '*') {
      if (cursor == 1) localIronMask = !localIronMask;
    } else if (k == '1') {
      if (cursor == 0 && localDisc > 0) localDisc--;
      else if (cursor == 2 && localNotchLo > 0) localNotchLo--;
    } else if (k == '3') {
      if (cursor == 0 && localDisc < 10) localDisc++;
      else if (cursor == 2 && localNotchHi < 99) localNotchHi++;
    } else if (k == 'A') {
      discriminator = localDisc;
      ironMask = localIronMask;
      notchLo = localNotchLo;
      notchHi = localNotchHi;
      showNotification("Ayirma Ayarlandi", 0);
      delay(1000);
      menuLevel = 0;
      lcd.clear();
      return;
    } else if (k == 'D') {
      menuLevel = 0;
      lcd.clear();
      return;
    }
  }
}

void frequencyMenu() {
  if (!lcdReady) return;
```

```
lcd.clear();
lcd.setCursor(0, 0);
lcd.print(F("=== FREKANS AYARI =="));

const char* freqs[] = {"5kHz", "7.5kHz", "10kHz", "15kHz", "20kHz"};
byte localFreq = freqMode;
bool localShift = freqShift;
bool localMulti = multiFreq;
byte cursor = 0;

while (true) {
  lcd.setCursor(0, 1);
  lcd.print(cursor == 0 ? F(">") : F(" "));
  lcd.print(F("Frekans: "));
  lcd.print(freqs[localFreq]);
  lcd.print(F("  "));

  lcd.setCursor(0, 2);
  lcd.print(cursor == 1 ? F(">") : F(" "));
  lcd.print(F("Shift: "));
  lcd.print(localShift ? F("ON ") : F("OFF"));

  lcd.setCursor(0, 3);
  lcd.print(cursor == 2 ? F(">") : F(" "));
  lcd.print(F("Multi: "));
  lcd.print(localMulti ? F("ON ") : F("OFF"));

  char k = getKey();
  if (k == 'B') {
    if (cursor > 0) cursor--;
```

```
      else cursor = 2;
    } else if (k == 'C') {
      if (cursor < 2) cursor++;
      else cursor = 0;
    } else if (k == '*') {
      if (cursor == 1) localShift = !localShift;
      else if (cursor == 2) localMulti = !localMulti;
    } else if (k == '1') {
      if (cursor == 0 && localFreq > 0) localFreq--;
    } else if (k == '3') {
      if (cursor == 0 && localFreq < 4) localFreq++;
    } else if (k == 'A') {
      freqMode = localFreq;
      freqShift = localShift;
      multiFreq = localMulti;
      showNotification("Frekans Ayarlandi", 0);
      delay(1000);
      menuLevel = 0;
      lcd.clear();
      return;
    } else if (k == 'D') {
      menuLevel = 0;
      lcd.clear();
      return;
    }
  }
}

void displaySettingsMenu() {
  if (!lcdReady) return;
```

```
lcd.clear();
lcd.setCursor(0, 0);
lcd.print(F("=== EKRAN AYARLARI ="));

byte localBright = brightness;
bool localBacklight = backlight;
bool localOled = oledEnabled;
byte cursor = 0;

while (true) {
  lcd.setCursor(0, 1);
  lcd.print(cursor == 0 ? F(">") : F(" "));
  lcd.print(F("Parlak: "));
  lcd.print(localBright);
  lcd.print(F("/10  "));

  lcd.setCursor(0, 2);
  lcd.print(cursor == 1 ? F(">") : F(" "));
  lcd.print(F("BLight: "));
  lcd.print(localBacklight ? F("ON ") : F("OFF"));

  lcd.setCursor(0, 3);
  lcd.print(cursor == 2 ? F(">") : F(" "));
  if (oledReady) {
    lcd.print(F("OLED: "));
    lcd.print(localOled ? F("ON ") : F("OFF"));
  } else {
    lcd.print(F("OLED: YOK     "));
  }

  char k = getKey();
```

```
    if (k == 'B') {
      if (cursor > 0) cursor--;
      else cursor = 2;
    } else if (k == 'C') {
      if (cursor < 2) cursor++;
      else cursor = 0;
    } else if (k == '*') {
      if (cursor == 1) localBacklight = !localBacklight;
      else if (cursor == 2 && oledReady) localOled = !localOled;
    } else if (k == '1') {
      if (cursor == 0 && localBright > 1) localBright--;
    } else if (k == '3') {
      if (cursor == 0 && localBright < 10) localBright++;
    } else if (k == 'A') {
      brightness = localBright;
      backlight = localBacklight;
      if (oledReady) oledEnabled = localOled;
      showNotification("Ekran Ayarlandi", 0);
      delay(1000);
      menuLevel = 0;
      lcd.clear();
      return;
    } else if (k == 'D') {
      menuLevel = 0;
      lcd.clear();
      return;
    }
  }
}
// ============================================================================
// MENU FUNCTIONS - DEVAM
```

```
// ================================================================================

void gpsNavigationMenu() {
  if (!gpsReady) {
    showNotification("GPS Bagli Degil!", 1);
    delay(2000);
    menuLevel = 0;
    if (lcdReady) lcd.clear();
    return;
  }

  if (!lcdReady) return;

  lcd.clear();

  unsigned long startTime = millis();
  while (millis() - startTime < 30000) {
    lcd.setCursor(0, 0);
    lcd.print(F("=== GPS BILGISI ===="));

    lcd.setCursor(0, 1);
    lcd.print(F("Uydu: "));
    lcd.print(gpsSatellites);
    lcd.print(F(" Lock: "));
    lcd.print(gpsLock ? F("YES") : F("NO "));

    lcd.setCursor(0, 2);
    lcd.print(F("Lat: "));
    lcd.print(latitude, 5);
    lcd.print(F("  "));
```

```
    lcd.setCursor(0, 3);
    lcd.print(F("Lon: "));
    lcd.print(longitude, 5);
    lcd.print(F("  "));

    updateGPS();

    char k = getKey();
    if (k == 'D' || k == 'A') {
      menuLevel = 0;
      lcd.clear();
      return;
    }

    delay(200);
  }

  menuLevel = 0;
  lcd.clear();
}

void waypointMenu() {
  if (!gpsReady) {
    showNotification("GPS Bagli Degil!", 1);
    delay(2000);
    menuLevel = 0;
    if (lcdReady) lcd.clear();
    return;
  }

  if (!lcdReady) return;
```

```
lcd.clear();
lcd.setCursor(0, 0);
lcd.print(F("== WAYPOINT SISTEM =="));
lcd.setCursor(0, 1);
lcd.print(F("Toplam Nokta: "));
lcd.print(waypointCount);
lcd.setCursor(0, 2);
lcd.print(F("[A]=Ekle  [B]=Liste"));
lcd.setCursor(0, 3);
lcd.print(F("[C]=Temizle  [D]=Cikis"));

unsigned long wait = millis();
while (millis() - wait < 10000) {
  char k = getKey();
  if (k == 'A') {
    addWaypoint();
    delay(1000);
    menuLevel = 0;
    lcd.clear();
    return;
  } else if (k == 'B') {
    if (waypointCount == 0) {
      showNotification("Henuz Nokta Yok!", 1);
      delay(2000);
      menuLevel = 0;
      lcd.clear();
      return;
    }

    for (byte i = 0; i < waypointCount && i < 10; i++) {
```

```
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(waypoints[i].name);
    lcd.print(F(" Hot:"));
    lcd.print(waypoints[i].hotness);
    lcd.print(F("%"));

    lcd.setCursor(0, 1);
    lcd.print(F("Metal: "));
    lcd.print(waypoints[i].metalCount);
    lcd.print(F(" Max:"));
    lcd.print(waypoints[i].maxDepth);
    lcd.print(F("cm"));

    lcd.setCursor(0, 2);
    lcd.print(F("Lat: "));
    lcd.print(waypoints[i].lat, 4);

    lcd.setCursor(0, 3);
    lcd.print(F("Lon: "));
    lcd.print(waypoints[i].lon, 4);

    delay(2000);
  }
  menuLevel = 0;
  lcd.clear();
  return;
 } else if (k == 'C') {
  waypointCount = 0;
  showNotification("Tum Noktalar Silindi", 0);
  delay(1000);
```

```
      menuLevel = 0;
      lcd.clear();
      return;
    } else if (k == 'D') {
      menuLevel = 0;
      lcd.clear();
      return;
    }
  }

  menuLevel = 0;
  lcd.clear();
}

void loaderModeFunc() {
  if (!ultrasonicReady) {
    showNotification("Ultrasonic Bagli Degil!", 1);
    delay(2000);
    menuLevel = 0;
    if (lcdReady) lcd.clear();
    return;
  }

  if (!lcdReady) return;

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(F("==== LOADER MODU ===="));
  lcd.setCursor(0, 1);
  lcd.print(F("[A]=Basla [D]=Cikis"));
```

```
unsigned long wait = millis();
while (millis() - wait < 10000) {
  char k = getKey();
  if (k == 'D') {
    menuLevel = 0;
    lcd.clear();
    return;
  } else if (k == 'A') {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(F("==== LOADER AKTIF ==="));
    lcd.setCursor(0, 1);
    lcd.print(F("[D]=Dur [*]=Sifirla"));

    loaderMode = true;
    loaderDepth = 0;
    loaderWeight = 0;
    loaderVolume = 0;
    loaderDensity = 0;
    loaderSamples = 0;

    unsigned long startTime = millis();
    while (loaderMode && (millis() - startTime < 60000)) {
      char key = getKey();
      if (key == 'D' || key == 'A') {
        loaderMode = false;
        menuLevel = 0;
        lcd.clear();
        return;
      }
      if (key == '*') {
```

```
  loaderDepth = 0;
  loaderWeight = 0;
  loaderVolume = 0;
  loaderDensity = 0;
  for (byte i = 0; i < 10; i++) loaderHistory[i] = 0;
  showNotification("Loader Sifirlandi", 0);
}

int rawDepth = getUltrasonicDepth();
if (rawDepth > 0 && rawDepth != loaderDepth) {
  loaderDepth = rawDepth;
  loaderVolume = (long)loaderDepth * 10;
  loaderWeight = loaderVolume * 15 / 10;
  if (loaderVolume > 0) {
    loaderDensity = (loaderWeight * 10) / loaderVolume;
  }
}

lcd.setCursor(0, 2);
lcd.print(F("D:"));
lcd.print(loaderDepth);
lcd.print(F("cm W:"));
lcd.print(loaderWeight);
lcd.print(F("kg     "));

lcd.setCursor(0, 3);
lcd.print(F("V:"));
lcd.print(loaderVolume);
lcd.print(F("L Den:"));
lcd.print(loaderDensity);
lcd.print(F("    "));
```

```
      delay(200);
    }

    loaderMode = false;
    menuLevel = 0;
    lcd.clear();
    return;
   }
  }
  menuLevel = 0;
  lcd.clear();
}

void distanceMeasurementFunc() {
  if (!gpsReady) {
   showNotification("GPS Bagli Degil!", 1);
   delay(2000);
   menuLevel = 0;
   if (lcdReady) lcd.clear();
   return;
  }

  if (!lcdReady) return;

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(F("=== MESAFE OLCUM ===="));
  lcd.setCursor(0, 1);
  lcd.print(F(" [A]=Basla [D]=Cikis"));
```

```
unsigned long wait = millis();
char k = NO_KEY;
while (millis() - wait < 10000) {
  k = getKey();
  if (k == 'A' || k == 'D') break;
}

if (k == 'D' || k == NO_KEY) {
  menuLevel = 0;
  lcd.clear();
  return;
}

distanceMode = true;
startDistLat = latitude;
startDistLon = longitude;
currentDistance = 0;
distancePoints = 0;
totalPathLength = 0;

lcd.clear();
lcd.setCursor(0, 0);
lcd.print(F("=== MESAFE OLCUM ===="));
lcd.setCursor(0, 1);
lcd.print(F("[#]=Nokta [*]=Reset [D]=Son"));

float lastPointLat = startDistLat;
float lastPointLon = startDistLon;

unsigned long startTime = millis();
while (distanceMode && (millis() - startTime < 60000)) {
```

```
char key = getKey();
if (key == 'D' || key == 'A') {
  distanceMode = false;
  menuLevel = 0;
  lcd.clear();
  return;
}
if (key == '#') {
  float dist = calcDistance(startDistLat, startDistLon, latitude, longitude);
  currentDistance = dist;

  float segmentDist = calcDistance(lastPointLat, lastPointLon, latitude, longitude);
  totalPathLength += segmentDist;

  distancePoints++;
  lastPointLat = latitude;
  lastPointLon = longitude;

  if (soundOn) tone(BUZZER_PIN, 2000, 100);
}
if (key == '*') {
  currentDistance = 0;
  distancePoints = 0;
  totalPathLength = 0;
  startDistLat = latitude;
  startDistLon = longitude;
  lastPointLat = latitude;
  lastPointLon = longitude;
  showNotification("Reset Yapildi", 0);
}
```

```
    lcd.setCursor(0, 2);
    lcd.print(F("Dogrudan: "));
    lcd.print((int)currentDistance);
    lcd.print(F("m    "));

    lcd.setCursor(0, 3);
    lcd.print(F("Yol: "));
    lcd.print((int)totalPathLength);
    lcd.print(F("m P:"));
    lcd.print(distancePoints);
    lcd.print(F("    "));

    updateGPS();
    delay(200);
  }

  distanceMode = false;
  menuLevel = 0;
  lcd.clear();
}

void ultrasonicSearchFunc() {
  if (!ultrasonicReady) {
    showNotification("Ultrasonic Bagli Degil!", 1);
    delay(2000);
    menuLevel = 0;
    if (lcdReady) lcd.clear();
    return;
  }

  if (!lcdReady) return;
```

```
lcd.clear();
lcd.setCursor(0, 0);
lcd.print(F("== ULTRASONIK ARAMA ="));
lcd.setCursor(0, 1);
lcd.print(F(" [A]=Tara [D]=Cikis"));
lcd.setCursor(0, 2);
lcd.print(F(" [1]=20pt [2]=30pt [3]=50pt"));

unsigned long wait = millis();
char k = NO_KEY;
while (millis() - wait < 10000) {
  k = getKey();
  if (k == '1') ultrasonicResolution = 20;
  else if (k == '2') ultrasonicResolution = 30;
  else if (k == '3') ultrasonicResolution = 50;
  if (k == 'A' || k == 'D' || k == '1' || k == '2' || k == '3') break;
}

if (k == 'D' || k == NO_KEY) {
  menuLevel = 0;
  lcd.clear();
  return;
}

ultrasonicMode = true;
ultrasonicScanIndex = 0;

lcd.clear();
lcd.setCursor(0, 0);
lcd.print(F("== ULTRASONIK ARAMA ="));
```

```
lcd.setCursor(0, 1);
lcd.print(F("   TARAMA YAPILIYOR"));

for (byte i = 0; i < ultrasonicResolution; i++) {
  int d = getUltrasonicDepth();
  ultrasonicScanData[i] = (d > 0) ? d : 0;

  lcd.setCursor(0, 2);
  lcd.print(F(" Nokta: "));
  lcd.print(i + 1);
  lcd.print(F("/"));
  lcd.print(ultrasonicResolution);
  lcd.print(F("    "));

  byte progress = map(i, 0, ultrasonicResolution - 1, 0, 100);
  drawProgressBar(3, progress);

  if (soundOn) tone(BUZZER_PIN, 1500 + i * 30, 40);
  delay(150);
}

ultrasonicMode = false;

lcd.clear();
lcd.setCursor(0, 1);
lcd.print(F("   TARAMA TAMAMLANDI"));
lcd.setCursor(0, 2);
lcd.print(F("  Web arayuzunden"));
lcd.setCursor(0, 3);
lcd.print(F("  goruntuleyin"));
```

```
  if (soundOn) {
    tone(BUZZER_PIN, 2500, 200);
    delay(250);
    tone(BUZZER_PIN, 3000, 200);
  }

  delay(3000);
  menuLevel = 0;
  lcd.clear();
}

void statisticsMenu() {
  if (!lcdReady) return;

  lcd.clear();

  byte page = 0;
  unsigned long startTime = millis();

  while (millis() - startTime < 30000) {
    lcd.setCursor(0, 0);
    lcd.print(F("=== ISTATISTIKLER =="));

    if (page == 0) {
      lcd.setCursor(0, 1);
      lcd.print(F("Toplam: "));
      lcd.print(totalMetalCount);
      lcd.print(F("        "));

      lcd.setCursor(0, 2);
      lcd.print(F("Oturum: "));
```

```
  lcd.print(currentSession.totalFinds);
  lcd.print(F("       "));

  lcd.setCursor(0, 3);
  lcd.print(F("Demir:"));
  lcd.print(ironFound);
  lcd.print(F(" Altin:"));
  lcd.print(goldFound);
  lcd.print(F("  "));
 } else if (page == 1) {
  lcd.setCursor(0, 1);
  lcd.print(F("Bakir:"));
  lcd.print(copperFound);
  lcd.print(F(" Gumus:"));
  lcd.print(silverFound);
  lcd.print(F("  "));

  lcd.setCursor(0, 2);
  lcd.print(F("Alv:"));
  lcd.print(aluminumFound);
  lcd.print(F(" Pirinc:"));
  lcd.print(brassFound);
  lcd.print(F("  "));

  lcd.setCursor(0, 3);
  lcd.print(F("MaxDerinlik: "));
  lcd.print(maxDepthEver);
  lcd.print(F("cm  "));
 } else if (page == 2) {
  lcd.setCursor(0, 1);
  lcd.print(F("Mesafe: "));
```

```
    lcd.print((int)currentSession.totalDistance);
    lcd.print(F("m    "));

    lcd.setCursor(0, 2);
    lcd.print(F("Sure: "));
    lcd.print(currentSession.duration / 60000);
    lcd.print(F("dk    "));

    lcd.setCursor(0, 3);
    lcd.print(F("Toplam Oturum: "));
    lcd.print(totalSessions);
    lcd.print(F("  "));
  }

  char k = getKey();
  if (k == 'C') {
    page++;
    if (page > 2) page = 0;
    startTime = millis();
  } else if (k == 'B') {
    if (page > 0) page--;
    else page = 2;
    startTime = millis();
  } else if (k == 'D' || k == 'A') {
    menuLevel = 0;
    lcd.clear();
    return;
  }

  delay(100);
}
```

```
  menuLevel = 0;
  lcd.clear();
}

void sessionInfoMenu() {
  if (!lcdReady) return;

  lcd.clear();

  unsigned long startTime = millis();
  while (millis() - startTime < 30000) {
    lcd.setCursor(0, 0);
    lcd.print(F("=== OTURUM BILGI ==="));

    unsigned long dur = (millis() - currentSession.startTime) / 1000;

    lcd.setCursor(0, 1);
    lcd.print(F("Sure: "));
    lcd.print(dur / 60);
    lcd.print(F("dk "));
    lcd.print(dur % 60);
    lcd.print(F("sn    "));

    lcd.setCursor(0, 2);
    lcd.print(F("Buluntu: "));
    lcd.print(currentSession.totalFinds);
    lcd.print(F("       "));

    lcd.setCursor(0, 3);
    lcd.print(F("MaxDer: "));
```

```
    lcd.print(currentSession.maxDepthRecord);
    lcd.print(F("cm      "));

    char k = getKey();
    if (k == 'D' || k == 'A') {
      menuLevel = 0;
      lcd.clear();
      return;
    }

    delay(500);
  }

  menuLevel = 0;
  lcd.clear();
}

void batteryManagementMenu() {
  if (!lcdReady) return;

  lcd.clear();

  unsigned long startTime = millis();
  while (millis() - startTime < 30000) {
    updateBattery();

    lcd.setCursor(0, 0);
    lcd.print(F("== BATARYA YONETIMI ="));

    lcd.setCursor(0, 1);
    lcd.print(F("Voltaj: "));
```

```
    lcd.print(batVoltage, 2);
    lcd.print(F("V     "));

    lcd.setCursor(0, 2);
    lcd.print(F("Yuzde: %"));
    lcd.print(batPercent);
    lcd.print(F("       "));

    lcd.setCursor(0, 3);
    lcd.print(F("Kalan: "));
    lcd.print(batTimeRemaining);
    lcd.print(F(" dk      "));

    char k = getKey();
    if (k == 'D' || k == 'A') {
      menuLevel = 0;
      lcd.clear();
      return;
    }

    delay(500);
  }

  menuLevel = 0;
  lcd.clear();
}

void dataManagementMenu() {
  if (!lcdReady) return;

  lcd.clear();
```

```
  lcd.setCursor(0, 0);
  lcd.print(F("=== VERI YONETIMI =="));

  if (sdReady) {
    lcd.setCursor(0, 1);
    lcd.print(F("SD: HAZIR"));
    lcd.setCursor(0, 2);
    lcd.print(F("Kayit: "));
    lcd.print(recordCount);
  } else {
    lcd.setCursor(0, 1);
    lcd.print(F("SD: BAGLI DEGIL"));
  }

  lcd.setCursor(0, 3);
  lcd.print(F("[D]=Cikis"));

  unsigned long wait = millis();
  while (millis() - wait < 5000) {
    char k = getKey();
    if (k == 'D' || k == 'A') {
      menuLevel = 0;
      lcd.clear();
      return;
    }
  }

  menuLevel = 0;
  lcd.clear();
}
```

```
void sensorCalibrationMenu() {
  if (!lcdReady) return;

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(F("= SENSOR KALIBRASYON="));
  lcd.setCursor(0, 1);
  lcd.print(F("[A]=Toprak [B]=IMU"));
  lcd.setCursor(0, 2);
  lcd.print(F("[C]=GPS [D]=Cikis"));

  unsigned long wait = millis();
  while (millis() - wait < 10000) {
    char k = getKey();
    if (k == 'A') {
      startAutoCalibration();
      return;
    } else if (k == 'B') {
      if (mpuReady) {
        lcd.clear();
        lcd.setCursor(0, 1);
        lcd.print(F("  IMU KALIBRE EDILIYOR"));
        mpu.calcOffsets();
        delay(2000);
        showNotification("IMU Kalibrasyon OK", 0);
        delay(1000);
      } else {
        showNotification("IMU Bagli Degil!", 1);
        delay(2000);
      }
      menuLevel = 0;
```

```
      lcd.clear();
      return;
    } else if (k == 'C') {
      if (gpsReady) {
        lcd.clear();
        lcd.setCursor(0, 1);
        lcd.print(F("  GPS SIFIRLANIYOR"));
        startLat = latitude;
        startLon = longitude;
        delay(2000);
        showNotification("GPS Reset OK", 0);
        delay(1000);
      } else {
        showNotification("GPS Bagli Degil!", 1);
        delay(2000);
      }
      menuLevel = 0;
      lcd.clear();
      return;
    } else if (k == 'D') {
      menuLevel = 0;
      lcd.clear();
      return;
    }
  }

  menuLevel = 0;
  lcd.clear();
}

void advancedSettingsMenu() {
```

```
if (!lcdReady) return;

lcd.clear();
lcd.setCursor(0, 0);
lcd.print(F("= GELISMIS AYARLAR =="));

byte localThreshold = threshold;
bool localBoost = targetBoost;
byte localFilter = filterStrength;
byte cursor = 0;

while (true) {
  lcd.setCursor(0, 1);
  lcd.print(cursor == 0 ? F(">") : F(" "));
  lcd.print(F("Esik: "));
  lcd.print(localThreshold);
  lcd.print(F("/10    "));

  lcd.setCursor(0, 2);
  lcd.print(cursor == 1 ? F(">") : F(" "));
  lcd.print(F("Boost: "));
  lcd.print(localBoost ? F("ON ") : F("OFF"));

  lcd.setCursor(0, 3);
  lcd.print(cursor == 2 ? F(">") : F(" "));
  lcd.print(F("Filter: "));
  lcd.print(localFilter);
  lcd.print(F("/10   "));

  char k = getKey();
  if (k == 'B') {
```

```
      if (cursor > 0) cursor--;
      else cursor = 2;
    } else if (k == 'C') {
      if (cursor < 2) cursor++;
      else cursor = 0;
    } else if (k == '*') {
      if (cursor == 1) localBoost = !localBoost;
    } else if (k == '1') {
      if (cursor == 0 && localThreshold > 1) localThreshold--;
      else if (cursor == 2 && localFilter > 1) localFilter--;
    } else if (k == '3') {
      if (cursor == 0 && localThreshold < 10) localThreshold++;
      else if (cursor == 2 && localFilter < 10) localFilter++;
    } else if (k == 'A') {
      threshold = localThreshold;
      targetBoost = localBoost;
      filterStrength = localFilter;
      showNotification("Ayarlar Kaydedildi", 0);
      delay(1000);
      menuLevel = 0;
      lcd.clear();
      return;
    } else if (k == 'D') {
      menuLevel = 0;
      lcd.clear();
      return;
    }
  }
}

void spectrumAnalysisFunc() {
```

```
if (!lcdReady) return;

lcd.clear();
lcd.setCursor(0, 0);
lcd.print(F("== SPEKTRUM ANALIZ =="));
lcd.setCursor(0, 1);
lcd.print(F("  [A]=BASLA [D]=IPTAL"));

unsigned long wait = millis();
while (millis() - wait < 10000) {
  char k = getKey();
  if (k == 'D') {
    menuLevel = 0;
    lcd.clear();
    return;
  } else if (k == 'A') {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(F("== SPEKTRUM ANALIZ =="));
    lcd.setCursor(0, 1);
    lcd.print(F("  ANALIZ YAPILIYOR..."));

    for (byte i = 0; i < 16; i++) {
      long sum = 0;
      for (byte j = 0; j < 10; j++) {
        long sig = readSignalAdvanced();
        sum += sig;
        delay(10);
      }
      spectrum[i] = sum / 10;
```

```
  lcd.setCursor(0, 2);
  byte progress = map(i, 0, 15, 0, 100);
  drawProgressBar(2, progress);
}

int maxVal = 0;
for (byte i = 0; i < 16; i++) {
  if (spectrum[i] > maxVal) {
    maxVal = spectrum[i];
    spectrumPeak = i;
  }
}

lcd.clear();
lcd.setCursor(0, 0);
lcd.print(F("== SPEKTRUM SONUC =="));
lcd.setCursor(0, 1);
lcd.print(F("Peak Frekans: "));
lcd.print(spectrumPeak);
lcd.setCursor(0, 2);
lcd.print(F("Peak Deger: "));
lcd.print(maxVal);
lcd.setCursor(0, 3);
lcd.print(F("Kalite: "));
lcd.print(signalQuality);
lcd.print(F("%"));

if (soundOn) tone(BUZZER_PIN, 2000, 100);
delay(4000);
menuLevel = 0;
lcd.clear();
```

```
      return;
    }
  }
  menuLevel = 0;
  lcd.clear();
}

void machineLearningMenu() {
  if (!lcdReady) return;

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(F("= MAKINE OGRENMESI =="));
  lcd.setCursor(0, 1);
  lcd.print(F(" [A]=EGIT [D]=IPTAL"));

  unsigned long wait = millis();
  while (millis() - wait < 10000) {
    char k = getKey();
    if (k == 'D') {
      menuLevel = 0;
      lcd.clear();
      return;
    } else if (k == 'A') {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print(F("= MAKINE OGRENMESI =="));
      lcd.setCursor(0, 1);
      lcd.print(F("  MODEL EGITILIYOR..."));

      for (byte i = 0; i < 20; i++) {
```

```
    long sig = readSignalAdvanced();
    mlFeatures[i % 8] = (mlFeatures[i % 8] + sig) / 2;

    lcd.setCursor(0, 2);
    drawProgressBar(2, (i * 100) / 20);
    delay(100);
  }

  mlPrediction = random(1, 7);
  mlConfidence = random(60, 95);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(F("== ML TAHMIN SONUC =="));
  lcd.setCursor(0, 1);
  lcd.print(F("Tahmin: "));
  const char* types[] = {"", "Demir", "Altin", "Bakir", "Gumus", "Aly", "Pirinc"};
  lcd.print(types[mlPrediction]);
  lcd.setCursor(0, 2);
  lcd.print(F("Guven: %"));
  lcd.print(mlConfidence);
  lcd.setCursor(0, 3);
  lcd.print(F("Ornekler: "));
  lcd.print(totalMetalCount);

  if (soundOn) tone(BUZZER_PIN, 2200, 100);
  delay(4000);
  menuLevel = 0;
  lcd.clear();
  return;
}
```

```
  }
  menuLevel = 0;
  lcd.clear();
}

void weatherInfoFunc() {
  if (!lcdReady) return;

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(F("=== HAVA DURUMU ===="));

  updateEnvironment();

  lcd.setCursor(0, 1);
  lcd.print(F("Sicaklik: "));
  lcd.print((int)temperature);
  lcd.print(F(" C"));

  lcd.setCursor(0, 2);
  lcd.print(F("Nem: %"));
  lcd.print((int)humidity);
  lcd.print(F("  Bas: "));
  lcd.print(pressure);

  lcd.setCursor(0, 3);
  lcd.print(F("Durum: "));
  if (weatherCondition == 0) lcd.print(F("Acik"));
  else if (weatherCondition == 1) lcd.print(F("Bulutlu"));
  else if (weatherCondition == 2) lcd.print(F("Yagmurlu"));
  else if (weatherCondition == 3) lcd.print(F("Firtinali"));
```

```
  delay(4000);
  menuLevel = 0;
  lcd.clear();
}

// ============================================================================
// LCD DISPLAY - SCAN SCREEN
// ============================================================================

void displayScanScreen() {
  if (!lcdReady) return;

  updateBattery();

  lcd.setCursor(0, 0);
  lcd.print(F("S:"));
  lcd.print(signal);
  lcd.print(F("  "));

  byte bars = map(signal, 0, 1000, 0, 5);
  lcd.setCursor(9, 0);
  for (byte i = 0; i < 5; i++) {
    lcd.write(i < bars ? 0xFF : ' ');
  }

  lcd.setCursor(15, 0);
  lcd.print(F("Bat:"));
  if (batPercent >= 100) lcd.print(F("FL"));
  else if (batPercent < 10) lcd.print(F(" "));
  if (batPercent < 100 && batPercent >= 10) lcd.print(F(" "));
```

```
lcd.print(batPercent);

lcd.setCursor(0, 1);

if (autoCalActive) {
  lcd.print(F(" KALIBRE EDILIYOR   "));
} else if (pinpoint) {
  lcd.print(F("PINPOINT: "));
  lcd.print(signal);
  lcd.print(F("      "));
} else if (metalType > 0 && signal > threshold * 100) {
  const char* types[] = {"", "De", "Al", "Ba", "Gu", "Aly", "Pi"};
  lcd.print(types[metalType]);
  lcd.print(F(" "));
  lcd.print(depth);
  lcd.print(F("cm V"));
  lcd.print(vdi);
  lcd.print(F(" C"));
  lcd.print(confidence);
  lcd.print(F("%  "));

  playAudio(metalType, signal);
  digitalWrite(LED_PIN, (millis() / 200) % 2);
} else {
  const char* modes[] = {"Tum", "Ayir", "Plaj", "Altin", "Relic", "Ozel"};
  lcd.print(modes[mode]);
  lcd.print(F(" "));
  lcd.print(gpsReady ? (gpsLock ? F("GPS") : F("gps")) : F("---"));
  lcd.print(F(" S"));
  lcd.print(currentSession.totalFinds);
  lcd.print(F(" T"));
```

```
  lcd.print(totalMetalCount);
  lcd.print(F("    "));

  digitalWrite(LED_PIN, LOW);
}

lcd.setCursor(0, 2);
if (mpuReady) {
  if (levelWarning) {
    lcd.print(F("!EGIM:"));
    lcd.print((int)coilAngle);
    lcd.print(F("d"));
  } else {
    lcd.print(F("Aci:"));
    lcd.print((int)coilAngle);
    lcd.print(F("d"));
  }
} else {
  lcd.print(F("IMU:NO"));
}

lcd.setCursor(12, 2);
if (wifiReady && wifiOn) {
  lcd.print(F("W"));
  lcd.print(connectedClients);
} else {
  lcd.print(F("--"));
}
lcd.print(F(" "));
if (bluetoothReady && bluetoothOn) {
  lcd.print(F("B"));
```

```
    lcd.print(bluetoothConnected ? F("1") : F("0"));
  } else {
    lcd.print(F("--"));
  }

  lcd.setCursor(0, 3);
  if (rtcReady) {
    if (now.hour() < 10) lcd.print(F("0"));
    lcd.print(now.hour());
    lcd.print(F(":"));
    if (now.minute() < 10) lcd.print(F("0"));
    lcd.print(now.minute());
  } else {
    lcd.print(F("--:--"));
  }

  lcd.print(F(" Ot:"));
  lcd.print(currentSession.totalFinds);
  lcd.print(F(" WP:"));
  lcd.print(waypointCount);
  lcd.print(F("  "));

  if (backlight && !lowPowerMode) {
    analogWrite(BACKLIGHT_PIN, map(brightness, 0, 10, 0, 255));
  } else {
    analogWrite(BACKLIGHT_PIN, 50);
  }
}

// ============================================================================
// LCD DISPLAY - MAIN MENU
```

```
// ========================================================================

void displayMainMenu() {
  if (!lcdReady) return;

  lcd.setCursor(0, 0);
  lcd.print(F("===== ANA MENU ====="));

  for (byte i = 0; i < 3; i++) {
    byte idx = scrollPos + i;
    if (idx >= MAIN_MENU_COUNT) break;

    lcd.setCursor(0, i + 1);

    if (idx == menuIndex) {
      lcd.print(F(">"));
    } else {
      lcd.print(F(" "));
    }

    char truncated[20];
    strncpy(truncated, mainMenuItems[idx], 19);
    truncated[19] = '\0';
    lcd.print(truncated);

    byte len = strlen(truncated);
    for (byte j = len + 1; j < 20; j++) {
      lcd.print(F(" "));
    }
  }
}
```

```
// ============================================================================
// MENU HANDLER
// ============================================================================

void handleMainMenuKey(char key) {
  if (!lcdReady) return;

  if (key == 'B') {
    if (menuIndex > 0) {
      menuIndex--;
      if (menuIndex < scrollPos) scrollPos = menuIndex;
    } else {
      menuIndex = MAIN_MENU_COUNT - 1;
      scrollPos = max(0, menuIndex - 2);
    }
    lcd.clear();
  } else if (key == 'C') {
    if (menuIndex < MAIN_MENU_COUNT - 1) {
      menuIndex++;
      if (menuIndex > scrollPos + 2) scrollPos = menuIndex - 2;
    } else {
      menuIndex = 0;
      scrollPos = 0;
    }
    lcd.clear();
  } else if (key == 'A') {
    switch (menuIndex) {
      case 0: startAutoCalibration(); break;
      case 1: searchModeMenu(); break;
      case 2: sensitivityMenu(); break;
```

```
      case 3: speedRecoveryMenu(); break;
      case 4: audioSettingsMenu(); break;
      case 5: discriminationMenu(); break;
      case 6: frequencyMenu(); break;
      case 7: displaySettingsMenu(); break;
      case 8: gpsNavigationMenu(); break;
      case 9: waypointMenu(); break;
      case 10: loaderModeFunc(); break;
      case 11: distanceMeasurementFunc(); break;
      case 12: ultrasonicSearchFunc(); break;
      case 13: statisticsMenu(); break;
      case 14: sessionInfoMenu(); break;
      case 15: batteryManagementMenu(); break;
      case 16: initWiFiRouter(); break;
      case 17: initBluetooth(); break;
      case 18: dataManagementMenu(); break;
      case 19: sensorCalibrationMenu(); break;
      case 20: advancedSettingsMenu(); break;
      case 21: spectrumAnalysisFunc(); break;
      case 22: machineLearningMenu(); break;
      case 23: weatherInfoFunc(); break;
      case 24: saveConfig(); break;
      case 25: loadConfig(); break;
      case 26: factoryReset(); break;
    }
  } else if (key == 'D') {
    menuLevel = 0;
    lcd.clear();
  }
}
```

```
// ============================================================================
// KEY HANDLER
// ============================================================================

void handleKey(char key) {
  if (key == NO_KEY) return;

  if (menuLevel == 0) {
    if (key == 'A') {
      menuLevel = 1;
      menuIndex = 0;
      scrollPos = 0;
      if (lcdReady) lcd.clear();
    } else if (key == '*') {
      pinpoint = !pinpoint;
      showNotification(pinpoint ? "Pinpoint ACIK" : "Pinpoint KAPALI", 0);
    } else if (key == '#') {
      oledMode = (oledMode + 1) % 7;
      const char* modes[] = {"Kapali", "Istatistik", "GPS", "Grafik", "Derinlik", "3D", "Spektrum"};
      showNotification(modes[oledMode], 0);
    } else if (key == 'B') {
      if (gpsReady && gpsLock && metalType > 0) {
        addWaypoint();
      }
    } else if (key == 'C') {
      soundOn = !soundOn;
      showNotification(soundOn ? "Ses ACIK" : "Ses KAPALI", 0);
    }
  } else if (menuLevel == 1) {
    handleMainMenuKey(key);
  }
```

```
}

// ============================================================================
// SETUP
// ============================================================================

void setup() {
  pinMode(COIL_PIN, OUTPUT);
  pinMode(BUZZER_PIN, OUTPUT);
  pinMode(VIBRATOR_PIN, OUTPUT);
  pinMode(LED_PIN, OUTPUT);
  pinMode(BACKLIGHT_PIN, OUTPUT);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(SD_CS, OUTPUT);

  digitalWrite(COIL_PIN, LOW);
  digitalWrite(VIBRATOR_PIN, LOW);
  digitalWrite(LED_PIN, LOW);
  digitalWrite(BACKLIGHT_PIN, HIGH);

  Wire.begin();

  detectLCD();

  if (lcdReady) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(F(" METAL DETECTOR PRO"));
    lcd.setCursor(0, 1);
    lcd.print(F("   v8.4 ULTIMATE"));
```

```
  lcd.setCursor(0, 2);
  lcd.print(F(" 27 MENU + NAV TUSLARI"));
  delay(1000);
}

for (byte i = 0; i < 3; i++) {
  tone(BUZZER_PIN, 1000 + i * 500, 80);
  delay(100);
}

if (lcdReady) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(F("=== SENSOR CHECK ==="));
}

if (lcdReady) {
  lcd.setCursor(0, 1);
  lcd.print(F("IMU..."));
}
detectMPU();
if (lcdReady) {
  lcd.print(mpuReady ? F("OK") : F("YOK"));
  delay(300);
}

if (lcdReady) {
  lcd.setCursor(10, 1);
  lcd.print(F("RTC..."));
}
detectRTC();
```

```
if (lcdReady) {
  lcd.print(rtcReady ? F("OK") : F("YOK"));
  delay(300);
}

if (lcdReady) {
  lcd.setCursor(0, 2);
  lcd.print(F("OLED..."));
}
detectOLED();
if (lcdReady) {
  lcd.print(oledReady ? F("OK") : F("YOK"));
  delay(300);
}

if (lcdReady) {
  lcd.setCursor(10, 2);
  lcd.print(F("SD..."));
}
detectSD();
if (lcdReady) {
  lcd.print(sdReady ? F("OK") : F("YOK"));
  delay(300);
}

if (lcdReady) {
  lcd.setCursor(0, 3);
  lcd.print(F("ULTR..."));
}
detectUltrasonic();
if (lcdReady) {
```

```
  lcd.print(ultrasonicReady ? F("OK") : F("YOK"));
  delay(300);
}

if (lcdReady) {
  lcd.setCursor(10, 3);
  lcd.print(F("WiFi..."));
}
detectWiFi();
if (lcdReady) {
  lcd.print(wifiReady ? F("OK") : F("YOK"));
  delay(300);
}

if (oledReady) {
  oled.clearDisplay();
  oled.setTextSize(2);
  oled.setTextColor(SSD1306_WHITE);
  oled.setCursor(5, 10);
  oled.println(F("METAL"));
  oled.setCursor(0, 30);
  oled.println(F("DETECTOR"));
  oled.setTextSize(1);
  oled.setCursor(10, 50);
  oled.println(F("Pro v8.4"));
  oled.display();
}

delay(2000);

if (lcdReady) {
```

```
  lcd.clear();
  lcd.setCursor(0, 1);
  lcd.print(F("  GPS Kontrol..."));
}
detectGPS();
if (lcdReady) {
  lcd.setCursor(0, 2);
  lcd.print(F("  GPS: "));
  lcd.print(gpsReady ? F("BAGLI") : F("BAGLI DEGIL"));
  delay(1000);
}

if (lcdReady) {
  lcd.setCursor(0, 3);
  lcd.print(F("  BT: "));
}
detectBluetooth();
if (lcdReady) {
  lcd.print(bluetoothReady ? F("BAGLI") : F("BAGLI DEGIL"));
  delay(1000);
}

if (EEPROM.read(ADDR_INIT) != 0xAA) {
  sensitivity = 5000;
  speed = 3;
  mode = 0;
  volume = 7;
  brightness = 8;
  displayTheme = 0;
  totalMetalCount = 0;
  totalSessions = 0;
```

```
  threshold = 5;
  discriminator = 5;
  recovery = 3;
  freqMode = 0;

  eepromWriteLong(ADDR_SENS, sensitivity);
  EEPROM.write(ADDR_SPEED, speed);
  EEPROM.write(ADDR_MODE, mode);
  EEPROM.write(ADDR_VOLUME, volume);
  EEPROM.write(ADDR_BRIGHTNESS, brightness);
  EEPROM.write(ADDR_THEME, displayTheme);
  EEPROM.write(ADDR_THRESHOLD, threshold);
  EEPROM.write(ADDR_DISCRIMINATOR, discriminator);
  EEPROM.write(ADDR_RECOVERY, recovery);
  EEPROM.write(ADDR_FREQ_MODE, freqMode);
  eepromWriteLong(ADDR_TOTAL_COUNT, totalMetalCount);
  eepromWriteLong(ADDR_TOTAL_SESSIONS, totalSessions);
  EEPROM.write(ADDR_INIT, 0xAA);
 } else {
  long s = eepromReadLong(ADDR_SENS);
  if (s >= 1000 && s <= 10000) sensitivity = s;

  speed = EEPROM.read(ADDR_SPEED);
  if (speed < 1 || speed > 5) speed = 3;

  mode = EEPROM.read(ADDR_MODE);
  if (mode > 5) mode = 0;

  volume = EEPROM.read(ADDR_VOLUME);
  if (volume > 10) volume = 7;
```

```
  brightness = EEPROM.read(ADDR_BRIGHTNESS);
  if (brightness > 10) brightness = 8;

  displayTheme = EEPROM.read(ADDR_THEME);
  if (displayTheme > 3) displayTheme = 0;

  threshold = EEPROM.read(ADDR_THRESHOLD);
  if (threshold > 10) threshold = 5;

  discriminator = EEPROM.read(ADDR_DISCRIMINATOR);
  if (discriminator > 10) discriminator = 5;

  recovery = EEPROM.read(ADDR_RECOVERY);
  if (recovery > 5) recovery = 3;

  freqMode = EEPROM.read(ADDR_FREQ_MODE);
  if (freqMode > 4) freqMode = 0;

  totalMetalCount = eepromReadLong(ADDR_TOTAL_COUNT);
  totalSessions = eepromReadLong(ADDR_TOTAL_SESSIONS);
}

currentSession.startTime = millis();
currentSession.totalFinds = 0;
currentSession.ironCount = 0;
currentSession.goldCount = 0;
currentSession.copperCount = 0;
currentSession.silverCount = 0;
currentSession.aluminumCount = 0;
currentSession.brassCount = 0;
currentSession.maxDepthRecord = 0;
```

```
currentSession.totalDistance = 0;
currentSession.avgConfidence = 0;
currentSession.falseSignals = 0;

totalSessions++;

for (byte i = 0; i < 32; i++) signalBuffer[i] = 0;
for (byte i = 0; i < 128; i++) signalHistory[i] = 0;
for (byte i = 0; i < 50; i++) ultrasonicScanData[i] = 0;
for (byte i = 0; i < 16; i++) spectrum[i] = 0;
for (byte i = 0; i < 8; i++) mlFeatures[i] = 0;

if (lcdReady) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(F("   SISTEM HAZIR!"));
  lcd.setCursor(0, 1);
  lcd.print(F("  Aktif Sensorler:"));

  byte sensorCount = 0;
  if (mpuReady) sensorCount++;
  if (rtcReady) sensorCount++;
  if (oledReady) sensorCount++;
  if (sdReady) sensorCount++;
  if (ultrasonicReady) sensorCount++;
  if (wifiReady) sensorCount++;
  if (gpsReady) sensorCount++;
  if (bluetoothReady) sensorCount++;

  lcd.setCursor(0, 2);
  lcd.print(F("  LCD+Core+"));
```

```
    lcd.print(sensorCount);
    lcd.print(F(" Sensor"));

    lcd.setCursor(0, 3);
    lcd.print(F(" [A]=Menu [B]=WP [*]=Pin"));

    tone(BUZZER_PIN, 3000, 300);
    delay(3000);

    lcd.clear();
  }

  menuLevel = 0;
}

// ================================================================================
// MAIN LOOP
// ================================================================================

void loop() {
  if (mpuReady) updateIMU();
  if (rtcReady) updateRTC();
  if (gpsReady) updateGPS();
  updateEnvironment();
  if (oledReady) updateOLED();

  char key = getKey();
  handleKey(key);

  if (menuLevel == 0) {
    signal = readSignalAdvanced();
```

```
metalType = detectMetalType(signal);
depth = calculateDepth(signal);

signalHistory[historyIndex] = map(signal, 0, 1000, 0, 255);
historyIndex = (historyIndex + 1) % 128;

if (metalType > 0 && signal > threshold * 100 && confidence > 50) {
  static unsigned long lastDetection = 0;
  if (millis() - lastDetection > 2000) {
    totalMetalCount++;
    currentSession.totalFinds++;

    switch (metalType) {
      case 1: ironFound++; currentSession.ironCount++; break;
      case 2: goldFound++; currentSession.goldCount++; break;
      case 3: copperFound++; currentSession.copperCount++; break;
      case 4: silverFound++; currentSession.silverCount++; break;
      case 5: aluminumFound++; currentSession.aluminumCount++; break;
      case 6: brassFound++; currentSession.brassCount++; break;
    }

    if (depth > maxDepthEver) maxDepthEver = depth;
    if (depth > currentSession.maxDepthRecord) {
      currentSession.maxDepthRecord = depth;
      currentSession.deepestType = metalType;
    }

    currentSession.avgConfidence = (currentSession.avgConfidence + confidence) / 2;

    if (ultrasonicReady) {
      ultrasonicDepth = getUltrasonicDepth();
```

```
      }

      lastDetection = millis();
    }
  }

  displayScanScreen();
 } else if (menuLevel == 1) {
  displayMainMenu();
 }

 currentSession.duration = millis() - currentSession.startTime;
 totalScanTime = currentSession.duration;

 delay(30);
}
```