🛠️ Edge AI 구현 예제 코드

> 이 문서는 아쿠아포닉스 시스템에서 Edge AI를 구현하기 위한 실용적인 코드 예제를 제공합니다.

📋 목차

---

1. 데이터 수집 시스템

1.1 ESP32 센서 데이터 수집 코드

```
/**
 * aquaponics_data_collector.ino
 * 아쿠아포닉스 시스템 센서 데이터 수집
 *
 * 하드웨어: ESP32, DS18B20, DHT22, pH 센서
 */

#include <OneWire.h>
#include <DallasTemperature.h>
#include <DHT.h>
#include <WiFi.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>
#include <SD.h>
#include <SPI.h>

// 핀 설정
#define ONE_WIRE_BUS 4      // DS18B20
#define DHT_PIN 5           // DHT22
#define PH_SENSOR_PIN 34    // pH 센서 (아날로그)
#define TDS_SENSOR_PIN 35   // TDS 센서 (아날로그)
#define LIGHT_SENSOR_PIN 32 // 조도 센서 (아날로그)
#define SD_CS_PIN 15        // SD 카드 CS 핀

// 센서 객체
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature waterTempSensor(&oneWire);
DHT dht(DHT_PIN, DHT22);

// WiFi 설정
const char* ssid = "YOUR_WIFI_SSID";
const char* password = "YOUR_WIFI_PASSWORD";

// 데이터 서버 (선택적)
const char* serverUrl = "http://your-server.com/api/data";

// 데이터 구조체
struct SensorData {
    float waterTemp;      // 수온 (°C)
    float airTemp;        // 실내 온도 (°C)
    float humidity;       // 습도 (%)
    float pH;             // pH 값
    float tds;            // TDS (ppm)
```

```
    float lightLevel;     // 조도 (lux 근사값)
    unsigned long timestamp;
};

// 데이터 버퍼 (Edge AI 입력용)
const int BUFFER_SIZE = 288;  // 24시간 (5분 간격)
SensorData dataBuffer[BUFFER_SIZE];
int bufferIndex = 0;

// 샘플링 간격
const unsigned long SAMPLE_INTERVAL = 300000;  // 5분
unsigned long lastSampleTime = 0;

void setup() {
    Serial.begin(115200);
    Serial.println("아쿠아포닉스 데이터 수집 시스템 시작");

    // 센서 초기화
    waterTempSensor.begin();
    dht.begin();

    // SD 카드 초기화
    if (!SD.begin(SD_CS_PIN)) {
        Serial.println("SD 카드 초기화 실패!");
    } else {
        Serial.println("SD 카드 초기화 성공");
    }

    // WiFi 연결 (선택적)
    WiFi.begin(ssid, password);
    int attempts = 0;
    while (WiFi.status() != WL_CONNECTED && attempts < 20) {
        delay(500);
        Serial.print(".");
        attempts++;
    }

    if (WiFi.status() == WL_CONNECTED) {
        Serial.println("\nWiFi 연결됨");
        Serial.println(WiFi.localIP());
    }
}

void loop() {
```

```
    unsigned long currentTime = millis();

    if (currentTime - lastSampleTime >= SAMPLE_INTERVAL) {
        lastSampleTime = currentTime;

        // 센서 데이터 읽기
        SensorData data = readAllSensors();

        // 버퍼에 저장 (순환 버퍼)
        dataBuffer[bufferIndex] = data;
        bufferIndex = (bufferIndex + 1) % BUFFER_SIZE;

        // 시리얼 출력
        printSensorData(data);

        // SD 카드에 저장
        saveToSD(data);

        // 서버에 전송 (WiFi 연결 시)
        if (WiFi.status() == WL_CONNECTED) {
            sendToServer(data);
        }
    }
}

SensorData readAllSensors() {
    SensorData data;

    // 수온 읽기
    waterTempSensor.requestTemperatures();
    data.waterTemp = waterTempSensor.getTempCByIndex(0);

    // 실내 온습도 읽기
    data.airTemp = dht.readTemperature();
    data.humidity = dht.readHumidity();

    // pH 읽기 (보정 필요)
    int phRaw = analogRead(PH_SENSOR_PIN);
    data.pH = convertToPH(phRaw);

    // TDS 읽기
    int tdsRaw = analogRead(TDS_SENSOR_PIN);
    data.tds = convertToTDS(tdsRaw, data.waterTemp);
```

```
    // 조도 읽기
    int lightRaw = analogRead(LIGHT_SENSOR_PIN);
    data.lightLevel = convertToLux(lightRaw);

    // 타임스탬프
    data.timestamp = millis();

    return data;
}

float convertToPH(int rawValue) {
    // pH 센서 보정 공식 (센서에 따라 조정 필요)
    // 일반적인 아날로그 pH 센서 기준
    float voltage = rawValue * 3.3 / 4095.0;
    float pH = 7.0 + ((2.5 - voltage) / 0.18);
    return constrain(pH, 0.0, 14.0);
}

float convertToTDS(int rawValue, float temperature) {
    // TDS 센서 변환 (온도 보정 포함)
    float voltage = rawValue * 3.3 / 4095.0;
    float compensationCoefficient = 1.0 + 0.02 * (temperature - 25.0);
    float compensatedVoltage = voltage / compensationCoefficient;
    float tds = (133.42 * compensatedVoltage * compensatedVoltage * compensatedVoltage
                - 255.86 * compensatedVoltage * compensatedVoltage
                + 857.39 * compensatedVoltage) * 0.5;
    return tds;
}

float convertToLux(int rawValue) {
    // 조도 센서 근사 변환
    // 정확한 값은 센서 데이터시트 참조
    return rawValue * 0.5;  // 대략적인 lux 값
}

void printSensorData(SensorData data) {
    Serial.println("========= 센서 데이터 =========");
    Serial.printf("수온: %.2f°C\n", data.waterTemp);
    Serial.printf("실내온도: %.2f°C\n", data.airTemp);
    Serial.printf("습도: %.2f%%\n", data.humidity);
    Serial.printf("pH: %.2f\n", data.pH);
    Serial.printf("TDS: %.0f ppm\n", data.tds);
    Serial.printf("조도: %.0f lux\n", data.lightLevel);
    Serial.println("==============================");
```

```
}

void saveToSD(SensorData data) {
    File dataFile = SD.open("/aquaponics_data.csv", FILE_APPEND);
    if (dataFile) {
        // CSV 형식으로 저장
        dataFile.printf("%lu,%.2f,%.2f,%.2f,%.2f,%.0f,%.0f\n",
            data.timestamp,
            data.waterTemp,
            data.airTemp,
            data.humidity,
            data.pH,
            data.tds,
            data.lightLevel
        );
        dataFile.close();
    }
}

void sendToServer(SensorData data) {
    HTTPClient http;
    http.begin(serverUrl);
    http.addHeader("Content-Type", "application/json");

    StaticJsonDocument<256> doc;
    doc["water_temp"] = data.waterTemp;
    doc["air_temp"] = data.airTemp;
    doc["humidity"] = data.humidity;
    doc["ph"] = data.pH;
    doc["tds"] = data.tds;
    doc["light"] = data.lightLevel;
    doc["timestamp"] = data.timestamp;

    String jsonString;
    serializeJson(doc, jsonString);

    int httpResponseCode = http.POST(jsonString);

    if (httpResponseCode > 0) {
        Serial.printf("서버 전송 성공: %d\n", httpResponseCode);
    } else {
        Serial.printf("서버 전송 실패: %s\n", http.errorToString(httpResponseCode).c_str()
    }
```

```
    http.end();
}

// Edge AI 입력용 - 최근 데이터 가져오기
void getRecentData(float* output, int numSamples) {
    int startIdx = (bufferIndex - numSamples + BUFFER_SIZE) % BUFFER_SIZE;

    for (int i = 0; i < numSamples; i++) {
        int idx = (startIdx + i) % BUFFER_SIZE;
        int outIdx = i * 6;

        output[outIdx + 0] = normalizeValue(dataBuffer[idx].waterTemp, 15.0, 30.0);
        output[outIdx + 1] = normalizeValue(dataBuffer[idx].pH, 5.0, 9.0);
        output[outIdx + 2] = normalizeValue(dataBuffer[idx].airTemp, 15.0, 35.0);
        output[outIdx + 3] = normalizeValue(dataBuffer[idx].humidity, 30.0, 90.0);
        output[outIdx + 4] = normalizeValue(dataBuffer[idx].tds, 0.0, 1000.0);
        output[outIdx + 5] = normalizeValue(dataBuffer[idx].lightLevel, 0.0, 10000.0);
    }
}

float normalizeValue(float value, float minVal, float maxVal) {
    return (value - minVal) / (maxVal - minVal);
}
```

---

2. 이상 탐지 모델

2.1 Python - 오토인코더 모델 학습

```
"""
anomaly_detection_train.py
아쿠아포닉스 이상 탐지 모델 학습 스크립트

필요 패키지: tensorflow, pandas, numpy, scikit-learn
"""

import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow import keras
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
import os

# 데이터 로드
def load_data(csv_path):
    """
    SD 카드에서 수집된 CSV 데이터 로드
    """
    df = pd.read_csv(csv_path, names=[
        'timestamp', 'water_temp', 'air_temp', 'humidity',
        'ph', 'tds', 'light'
    ])

    # 결측치 처리
    df = df.dropna()

    # 특성 선택
    features = ['water_temp', 'air_temp', 'humidity', 'ph', 'tds', 'light']
    X = df[features].values

    return X, features

# 데이터 전처리
def preprocess_data(X):
    """
    정규화 및 학습/테스트 분할
    """
    scaler = MinMaxScaler()
    X_scaled = scaler.fit_transform(X)

    X_train, X_test = train_test_split(X_scaled, test_size=0.2, random_state=42)
```

```
    return X_train, X_test, scaler

# 오토인코더 모델 정의
def build_autoencoder(input_dim, encoding_dim=3):
    """
    ESP32에서 실행 가능한 경량 오토인코더
    """
    # 인코더
    input_layer = keras.layers.Input(shape=(input_dim,))
    encoded = keras.layers.Dense(8, activation='relu')(input_layer)
    encoded = keras.layers.Dense(encoding_dim, activation='relu')(encoded)

    # 디코더
    decoded = keras.layers.Dense(8, activation='relu')(encoded)
    decoded = keras.layers.Dense(input_dim, activation='sigmoid')(decoded)

    # 오토인코더
    autoencoder = keras.Model(input_layer, decoded)

    # 인코더만 분리 (잠재 공간 분석용)
    encoder = keras.Model(input_layer, encoded)

    return autoencoder, encoder

# 모델 학습
def train_model(X_train, X_test, epochs=100):
    """
    오토인코더 모델 학습
    """
    input_dim = X_train.shape[1]
    autoencoder, encoder = build_autoencoder(input_dim)

    autoencoder.compile(
        optimizer='adam',
        loss='mse'
    )

    # 조기 종료 콜백
    early_stopping = keras.callbacks.EarlyStopping(
        monitor='val_loss',
        patience=10,
        restore_best_weights=True
    )
```

```
    history = autoencoder.fit(
        X_train, X_train,
        epochs=epochs,
        batch_size=32,
        validation_data=(X_test, X_test),
        callbacks=[early_stopping],
        verbose=1
    )

    return autoencoder, encoder, history

# 이상 임계값 계산
def calculate_threshold(autoencoder, X_train, percentile=95):
    """
    정상 데이터 기반 이상 임계값 계산
    """
    reconstructions = autoencoder.predict(X_train)
    mse = np.mean(np.power(X_train - reconstructions, 2), axis=1)
    threshold = np.percentile(mse, percentile)

    return threshold

# TensorFlow Lite 변환
def convert_to_tflite(model, output_path):
    """
    TensorFlow Lite Micro용 변환
    """
    converter = tf.lite.TFLiteConverter.from_keras_model(model)
    converter.optimizations = [tf.lite.Optimize.DEFAULT]
    converter.target_spec.supported_types = [tf.float16]

    tflite_model = converter.convert()

    with open(output_path, 'wb') as f:
        f.write(tflite_model)

    print(f"모델 저장됨: {output_path}")
    print(f"모델 크기: {len(tflite_model) / 1024:.2f} KB")

    return tflite_model

# C 헤더 파일 생성 (Arduino용)
def convert_to_c_header(tflite_model, output_path):
```

```
    """
    TFLite 모델을 C 배열로 변환
    """
    with open(output_path, 'w') as f:
        f.write("// Auto-generated TensorFlow Lite model\n")
        f.write("// Aquaponics Anomaly Detection Model\n\n")
        f.write("#ifndef ANOMALY_MODEL_H\n")
        f.write("#define ANOMALY_MODEL_H\n\n")
        f.write(f"const unsigned int model_len = {len(tflite_model)};\n")
        f.write("const unsigned char model_data[] = {\n  ")

        for i, byte in enumerate(tflite_model):
            f.write(f"0x{byte:02x}")
            if i < len(tflite_model) - 1:
                f.write(", ")
            if (i + 1) % 12 == 0:
                f.write("\n  ")

        f.write("\n};\n\n")
        f.write("#endif // ANOMALY_MODEL_H\n")

    print(f"C 헤더 파일 생성됨: {output_path}")

# 메인 실행
if __name__ == "__main__":
    # 데이터 로드
    X, features = load_data("aquaponics_data.csv")
    print(f"로드된 데이터: {X.shape[0]} 샘플, {X.shape[1]} 특성")

    # 전처리
    X_train, X_test, scaler = preprocess_data(X)

    # 모델 학습
    autoencoder, encoder, history = train_model(X_train, X_test)

    # 이상 임계값 계산
    threshold = calculate_threshold(autoencoder, X_train)
    print(f"\n이상 탐지 임계값: {threshold:.6f}")

    # TFLite 변환
    tflite_model = convert_to_tflite(autoencoder, "anomaly_model.tflite")

    # C 헤더 파일 생성
    convert_to_c_header(tflite_model, "anomaly_model.h")
```

```
    # 스케일러 파라미터 저장 (ESP32에서 사용)
    print("\n정규화 파라미터:")
    for i, feature in enumerate(features):
        print(f"  {feature}: min={scaler.data_min_[i]:.2f}, max={scaler.data_max_[i]:.2f}"
```

2.2 ESP32 - 이상 탐지 실행

```
/**
 * anomaly_detector.ino
 * ESP32에서 TensorFlow Lite Micro 이상 탐지 실행
 */

#include <TensorFlowLite_ESP32.h>
#include "tensorflow/lite/micro/all_ops_resolver.h"
#include "tensorflow/lite/micro/micro_error_reporter.h"
#include "tensorflow/lite/micro/micro_interpreter.h"
#include "tensorflow/lite/schema/schema_generated.h"

#include "anomaly_model.h"  // 변환된 모델 헤더

// 모델 설정
const int kInputSize = 6;   // 입력 특성 수
const float kAnomalyThreshold = 0.015;  // Python에서 계산된 임계값

// TFLite 변수
tflite::MicroErrorReporter micro_error_reporter;
tflite::AllOpsResolver resolver;
const tflite::Model* model = nullptr;
tflite::MicroInterpreter* interpreter = nullptr;
TfLiteTensor* input = nullptr;
TfLiteTensor* output = nullptr;

// 텐서 아레나 (모델 크기에 따라 조정)
constexpr int kTensorArenaSize = 8 * 1024;
uint8_t tensor_arena[kTensorArenaSize];

// 정규화 파라미터 (Python에서 계산된 값)
const float normMin[] = {15.0, 15.0, 30.0, 5.0, 0.0, 0.0};
const float normMax[] = {30.0, 35.0, 90.0, 9.0, 1000.0, 10000.0};

void setup() {
    Serial.begin(115200);

    // 모델 로드
    model = tflite::GetModel(model_data);
    if (model->version() != TFLITE_SCHEMA_VERSION) {
        Serial.println("모델 스키마 버전 불일치!");
        return;
    }
```

```
    // 인터프리터 생성
    static tflite::MicroInterpreter static_interpreter(
        model, resolver, tensor_arena, kTensorArenaSize, &micro_error_reporter);
    interpreter = &static_interpreter;

    // 텐서 할당
    TfLiteStatus allocate_status = interpreter->AllocateTensors();
    if (allocate_status != kTfLiteOk) {
        Serial.println("텐서 할당 실패!");
        return;
    }

    // 입출력 텐서 포인터
    input = interpreter->input(0);
    output = interpreter->output(0);

    Serial.println("이상 탐지 모델 로드 완료!");
    Serial.printf("입력 크기: %d, 출력 크기: %d\n",
        input->dims->data[1], output->dims->data[1]);
}

float detectAnomaly(float waterTemp, float airTemp, float humidity,
                    float pH, float tds, float light) {
    // 입력 데이터 정규화
    float sensorData[] = {waterTemp, airTemp, humidity, pH, tds, light};

    for (int i = 0; i < kInputSize; i++) {
        float normalized = (sensorData[i] - normMin[i]) / (normMax[i] - normMin[i]);
        normalized = constrain(normalized, 0.0, 1.0);
        input->data.f[i] = normalized;
    }

    // 모델 추론 실행
    TfLiteStatus invoke_status = interpreter->Invoke();
    if (invoke_status != kTfLiteOk) {
        Serial.println("추론 실패!");
        return -1;
    }

    // 재구성 오차 계산 (MSE)
    float mse = 0;
    for (int i = 0; i < kInputSize; i++) {
        float diff = input->data.f[i] - output->data.f[i];
        mse += diff * diff;
```

```
    }
    mse /= kInputSize;

    return mse;
}

void loop() {
    // 센서 데이터 읽기 (실제 센서 연동 필요)
    float waterTemp = readWaterTemp();
    float airTemp = readAirTemp();
    float humidity = readHumidity();
    float pH = readPH();
    float tds = readTDS();
    float light = readLight();

    // 이상 탐지 실행
    float anomalyScore = detectAnomaly(waterTemp, airTemp, humidity, pH, tds, light);

    Serial.printf("이상 점수: %.6f (임계값: %.6f)\n", anomalyScore, kAnomalyThreshold);

    if (anomalyScore > kAnomalyThreshold) {
        Serial.println("⚠️ 이상 감지! 시스템 점검 필요");
        // 알림 전송, LED 점멸, 부저 등
        triggerAlert(anomalyScore);
    } else {
        Serial.println("✅ 정상 상태");
    }

    delay(60000);  // 1분 간격
}

void triggerAlert(float score) {
    // 경고 알림 구현
    // 예: 부저, LED, 푸시 알림 등
}

// 센서 읽기 함수들 (실제 구현 필요)
float readWaterTemp() { return 22.5; }  // 예시 값
float readAirTemp() { return 24.0; }
float readHumidity() { return 65.0; }
float readPH() { return 7.0; }
```

```
float readTDS() { return 150.0; }
float readLight() { return 5000.0; }
```

---

3. 시계열 예측 모델

3.1 Python - LSTM 모델 학습

```
"""
water_quality_predictor_train.py
수질 예측을 위한 LSTM 모델 학습
"""

import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow import keras
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt

# 시퀀스 데이터 생성
def create_sequences(data, seq_length, forecast_horizon):
    """
    시계열 데이터를 학습용 시퀀스로 변환

    Args:
        data: 정규화된 데이터
        seq_length: 입력 시퀀스 길이 (과거 몇 개를 볼지)
        forecast_horizon: 예측 시점 (몇 스텝 후를 예측할지)
    """
    X, y = [], []
    for i in range(len(data) - seq_length - forecast_horizon):
        X.append(data[i:(i + seq_length)])
        y.append(data[i + seq_length + forecast_horizon - 1, 0])  # pH 예측
    return np.array(X), np.array(y)

# 경량 LSTM 모델 (ESP32용)
def build_lightweight_lstm(seq_length, num_features):
    """
    ESP32에서 실행 가능한 경량 LSTM 모델
    """
    model = keras.Sequential([
        keras.layers.LSTM(16, input_shape=(seq_length, num_features),
                          return_sequences=False),
        keras.layers.Dense(8, activation='relu'),
        keras.layers.Dense(1)
    ])

    model.compile(optimizer='adam', loss='mse', metrics=['mae'])
    return model
```

```
# 모델 학습
def train_predictor():
    # 데이터 로드
    df = pd.read_csv("aquaponics_data.csv", names=[
        'timestamp', 'water_temp', 'air_temp', 'humidity',
        'ph', 'tds', 'light'
    ])

    # pH를 첫 번째 열로 (예측 대상)
    features = ['ph', 'water_temp', 'air_temp', 'humidity', 'tds', 'light']
    data = df[features].values

    # 정규화
    scaler = MinMaxScaler()
    data_scaled = scaler.fit_transform(data)

    # 시퀀스 생성
    # 12개 샘플 (5분 간격 = 1시간) 입력 → 4시간 후 pH 예측
    SEQ_LENGTH = 12
    FORECAST_HORIZON = 48  # 4시간 (5분 * 48 = 240분)

    X, y = create_sequences(data_scaled, SEQ_LENGTH, FORECAST_HORIZON)

    # 학습/테스트 분할
    split = int(len(X) * 0.8)
    X_train, X_test = X[:split], X[split:]
    y_train, y_test = y[:split], y[split:]

    print(f"학습 데이터: {X_train.shape}, 테스트 데이터: {X_test.shape}")

    # 모델 생성 및 학습
    model = build_lightweight_lstm(SEQ_LENGTH, len(features))

    history = model.fit(
        X_train, y_train,
        epochs=50,
        batch_size=32,
        validation_data=(X_test, y_test),
        callbacks=[
            keras.callbacks.EarlyStopping(patience=10, restore_best_weights=True)
        ]
    )

    # 평가
```

```
    loss, mae = model.evaluate(X_test, y_test)
    print(f"\n테스트 MAE: {mae:.4f}")

    # pH 역정규화 범위
    ph_min, ph_max = scaler.data_min_[0], scaler.data_max_[0]
    actual_mae = mae * (ph_max - ph_min)
    print(f"실제 pH MAE: ±{actual_mae:.3f}")

    # TFLite 변환
    converter = tf.lite.TFLiteConverter.from_keras_model(model)
    converter.optimizations = [tf.lite.Optimize.DEFAULT]
    tflite_model = converter.convert()

    with open("ph_predictor.tflite", "wb") as f:
        f.write(tflite_model)

    print(f"\n모델 크기: {len(tflite_model) / 1024:.2f} KB")

    return model, scaler, history

if __name__ == "__main__":
    model, scaler, history = train_predictor()
```

---

4. Edge Impulse 연동

4.1 Edge Impulse 데이터 업로드 스크립트

```
"""
upload_to_edge_impulse.py
Edge Impulse Studio로 데이터 업로드
"""

import requests
import json
import pandas as pd
import time

# Edge Impulse 설정
API_KEY = "ei_xxxxx"  # Edge Impulse API 키
PROJECT_ID = "12345"  # 프로젝트 ID

def upload_sample(data, label, api_key):
    """
    Edge Impulse에 단일 샘플 업로드
    """
    url = f"https://ingestion.edgeimpulse.com/api/training/data"

    headers = {
        "x-api-key": api_key,
        "x-label": label,
        "Content-Type": "application/json"
    }

    payload = {
        "protected": {
            "ver": "v1",
            "alg": "none"
        },
        "signature": "0",
        "payload": {
            "device_name": "aquaponics_esp32",
            "device_type": "ESP32",
            "interval_ms": 300000,  # 5분 간격
            "sensors": [
                {"name": "waterTemp", "units": "Cel"},
                {"name": "pH", "units": "pH"},
                {"name": "airTemp", "units": "Cel"},
                {"name": "humidity", "units": "%"},
                {"name": "TDS", "units": "ppm"},
                {"name": "light", "units": "lux"}
```

```
            ],
            "values": data
        }
    }

    response = requests.post(url, headers=headers, json=payload)
    return response.status_code == 200

def upload_dataset(csv_path, api_key):
    """
    CSV 파일의 모든 데이터를 Edge Impulse에 업로드
    """
    df = pd.read_csv(csv_path, names=[
        'timestamp', 'water_temp', 'air_temp', 'humidity',
        'ph', 'tds', 'light'
    ])

    # 윈도우 크기 (12샘플 = 1시간)
    window_size = 12

    for i in range(0, len(df) - window_size, window_size):
        window = df.iloc[i:i+window_size]

        # 값 리스트 생성
        values = []
        for _, row in window.iterrows():
            values.append([
                row['water_temp'],
                row['ph'],
                row['air_temp'],
                row['humidity'],
                row['tds'],
                row['light']
            ])

        # 라벨 결정 (이상/정상)
        # 여기서는 pH 기준으로 간단히 라벨링
        avg_ph = window['ph'].mean()
        label = "normal" if 6.5 <= avg_ph <= 7.5 else "anomaly"

        success = upload_sample(values, label, api_key)

        if success:
            print(f"샘플 {i//window_size + 1} 업로드 완료 (라벨: {label})")
```

```
        else:
            print(f"샘플 {i//window_size + 1} 업로드 실패")

        time.sleep(0.5)  # API 제한 방지

if __name__ == "__main__":
    upload_dataset("aquaponics_data.csv", API_KEY)
```

4.2 Edge Impulse 모델 사용 (Arduino)

```
/**
 * edge_impulse_aquaponics.ino
 * Edge Impulse에서 생성된 모델 사용
 */

// Edge Impulse에서 다운로드한 라이브러리
#include <aquaponics_anomaly_inferencing.h>

// 센서 데이터 버퍼
static float features[EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE];

void setup() {
    Serial.begin(115200);
    Serial.println("Edge Impulse 아쿠아포닉스 이상 탐지");

    // 모델 정보 출력
    Serial.printf("특성 수: %d\n", EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE);
    Serial.printf("라벨 수: %d\n", EI_CLASSIFIER_LABEL_COUNT);
}

void loop() {
    // 센서 데이터 수집 (1시간치 = 12샘플)
    collectSensorData(features);

    // 신호 구조체 생성
    signal_t signal;
    int err = numpy::signal_from_buffer(features, EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE, &sig

    if (err != 0) {
        Serial.println("신호 생성 실패");
        return;
    }

    // 추론 실행
    ei_impulse_result_t result = {0};
    err = run_classifier(&signal, &result, false);

    if (err != EI_IMPULSE_OK) {
        Serial.printf("추론 실패: %d\n", err);
        return;
    }

    // 결과 출력
```

```
    Serial.println("========== 분류 결과 ==========");
    for (size_t ix = 0; ix < EI_CLASSIFIER_LABEL_COUNT; ix++) {
        Serial.printf("  %s: %.2f%%\n",
            result.classification[ix].label,
            result.classification[ix].value * 100);
    }

    // 이상 탐지 알림
    if (result.classification[0].value < 0.7) {  // "normal"이 70% 미만
        Serial.println("⚠️ 잠재적 이상 감지!");
        triggerAlert();
    }

    Serial.println("==============================\n");

    delay(300000);  // 5분 대기
}

void collectSensorData(float* buffer) {
    // 12개 샘플 수집 (5분 간격 시뮬레이션)
    int samplesPerWindow = 12;
    int featuresPerSample = 6;

    for (int i = 0; i < samplesPerWindow; i++) {
        int idx = i * featuresPerSample;

        buffer[idx + 0] = readWaterTemp();
        buffer[idx + 1] = readPH();
        buffer[idx + 2] = readAirTemp();
        buffer[idx + 3] = readHumidity();
        buffer[idx + 4] = readTDS();
        buffer[idx + 5] = readLight();

        if (i < samplesPerWindow - 1) {
            delay(1000);  // 실제로는 5분 대기 필요
        }
    }
}

void triggerAlert() {
    // 경고 구현
}

// 센서 읽기 함수들 (실제 구현 필요)
```

```
float readWaterTemp() { return 22.5 + random(-10, 10) / 10.0; }
float readPH() { return 7.0 + random(-5, 5) / 10.0; }
float readAirTemp() { return 24.0 + random(-10, 10) / 10.0; }
float readHumidity() { return 65.0 + random(-50, 50) / 10.0; }
float readTDS() { return 150.0 + random(-20, 20); }
float readLight() { return 5000.0 + random(-500, 500); }
```

📝 다음 단계

1. **데이터 수집 시작**: 최소 1개월간 센서 데이터 수집
2. **라벨링**: 이상 상황 발생 시 기록 (물고기 스트레스, 식물 문제 등)
3. **모델 학습**: 수집된 데이터로 모델 학습
4. **배포 및 테스트**: ESP32에 모델 배포 후 실시간 테스트
5. **피드백 반영**: 실제 사용 중 발견된 문제점 개선

작성일: 2026-01-15
버전: 1.0