🚀 VERBESSERUNGSVORSCHLÄGE & ROADMAP

Projekt: Arduino Control Panel
Stand: 23. Oktober 2025
Version: v2.5 (mit Live-Statistik-Widget)

✅ KÜRZLICH IMPLEMENTIERT

Live-Statistik-Widget (v2.5) 🎉

- ✅ **Echtzeit-Monitoring** von Testläufen
- ✅ **Live-Chart** mit Zykluszeiten
- ✅ **Anomalie-Erkennung** (>150% Durchschnitt)
- ✅ **Trend-Analyse** (↑ schneller / ↓ langsamer / → stabil)
- ✅ **Fortschrittsbalken** mit Prozent-Anzeige
- ✅ **Dock-Widget** Integration (rechte Seite)
- ✅ **Signal-System** für Live-Updates

Status: Basis implementiert, Live-Updates funktionsfähig
Datei: `live_stats_widget.py`

Dashboard-Verbesserungen

- ✅ Enhanced Dashboard mit Widget-Management
- ✅ Drag & Drop für Widgets
- ✅ Anpassbare Layouts
- ✅ Widget-Auswahl-Dialog

🔧 NOCH ZU VERVOLLSTÄNDIGEN

Live-Statistik-Widget - Feinschliff

Priorität: Mittel
Aufwand: 1-2 Stunden

- ☐ **stop_monitoring() Integration**
  - Datei: `ui/sequence_tab.py`
  - Methode: `set_running_state(is_running)`
  - Zeilen hinzufügen (~3 Zeilen)
  - Status: Vorbereitet, muss nur eingefügt werden
- ☐ **Anomalie-Erkennung verfeinern**
  - Aktuell: Fixer Schwellwert (150%)
  - Verbesserung: Adaptive Schwellwerte
  - Konfigurierbar machen
- ☐ **Cycle Counter Präzision**
  - Echte Zyklus-Anzahl aus Sequenz holen
  - Aktuell: Fixer Wert (100)

Dateien:

```
ui/sequence_tab.py        # stop_monitoring() Aufruf
live_stats_widget.py      # Anomalie-Logik
```

🎯 KURZFRISTIG (1-4 Wochen)

1. Makro-System vervollständigen ⭐ ⭐ ⭐

Priorität: Hoch
Aufwand: 8-12 Stunden

- [ ] Makro-Aufzeichnung implementieren
- [ ] Makro-Wiedergabe mit Timing
- [ ] Import/Export von Makros
- [ ] Makro-Editor mit Timeline
- [ ] Keyboard-Shortcuts für Makros

Nutzen: Automatisierung wiederkehrender Aktionen

Dateien:

```
ui/macro_system.py        # Basis vorhanden, vervollständigen
core/macro_recorder.py    # Neu erstellen
```

---

2. Sensor-System erweitern ⭐ ⭐

Priorität: Mittel
Aufwand: 4-6 Stunden

- [ ] Weitere Sensoren hinzufügen
  - Ultraschall-Distanz
  - Lichtsensor
  - Bewegungssensor
- [ ] Sensor-Kalibrierung
- [ ] Sensor-Alarme (Schwellwerte)
- [ ] Sensor-Logging in Datenbank

Nutzen: Umfangreicheres Monitoring

Dateien:

```
ui/sensor_tab.py          # Erweitern
core/sensor_manager.py    # Neu erstellen
```

---

3. Export-Funktionen ⭐ ⭐

Priorität: Mittel
Aufwand: 3-5 Stunden

- [ ] CSV-Export von Testergebnissen
- [ ] Excel-Export mit Formatierung
- [ ] JSON-Export für Automation
- [ ] Batch-Export mehrerer Tests

Nutzen: Datenanalyse außerhalb der Anwendung

Dateien:

```
analysis/export_manager.py  # Neu erstellen
ui/archive_tab.py           # Export-Buttons hinzufügen
```

📊 MITTELFRISTIG (1-3 Monate)

4. Erweiterte Statistik & Analytics ⭐ ⭐ ⭐

Priorität: Hoch
Aufwand: 12-16 Stunden

- [ ] **Langzeit-Trends**
  - Performance über Wochen/Monate
  - Degradations-Erkennung
  - Vorhersage-Modelle
- [ ] **Vergleichs-Analysen**
  - Test A vs. Test B
  - Hardware-Vergleiche
  - Optimierungs-Tracking
- [ ] **Statistik-Dashboard**
  - KPIs (Key Performance Indicators)
  - Heatmaps
  - Korrelations-Analysen

Nutzen: Datengetriebene Optimierung

Dateien:

```
analysis/advanced_stats.py      # Neu
analysis/prediction_model.py    # Neu
ui/analytics_dashboard.py       # Neu
```

---

5. Hardware-Profil-Management ⭐ ⭐

Priorität: Mittel
Aufwand: 6-8 Stunden

- [ ] Hardware-Profile speichern
  - Board-Typ (Uno, Mega, Nano, etc.)
  - Pin-Konfiguration
  - Sensor-Setup
- [ ] Profil-Import/Export
- [ ] Schneller Profil-Wechsel
- [ ] Auto-Erkennung von Boards

Nutzen: Arbeit mit mehreren Setups vereinfachen

Dateien:

```
core/hardware_profiles.py      # Neu
ui/profile_manager_dialog.py   # Neu
```

---

6. Remote-Zugriff & Monitoring 🌐

Priorität: Niedrig
Aufwand: 16-24 Stunden

- [ ] Web-Interface
  - Dashboard-Ansicht
  - Live-Monitoring
  - Start/Stop von Tests
- [ ] REST API
  - HTTP-Endpunkte
  - WebSocket für Live-Updates
  - Authentifizierung
- [ ] Mobile App (optional)
  - iOS/Android
  - Push-Benachrichtigungen

Nutzen: Überwachung von überall

Technologien:

Backend: Flask oder FastAPI
Frontend: React oder Vue.js
WebSocket: Socket.IO

🎨 LANGFRISTIG (3+ Monate)

7. Plugin-System ⭐⭐⭐

Priorität: Mittel
Aufwand: 20-30 Stunden

- [ ] Plugin-API definieren
- [ ] Plugin-Manager UI
- [ ] Beispiel-Plugins erstellen
- [ ] Plugin-Repository/Store

Nutzen: Erweiterbarkeit durch Community

Beispiel-Plugins:

- Custom Sensoren
- Spezielle Auswertungen
- Hardware-Integrationen
- Export-Formate

---

8. Machine Learning Integration 🤖

Priorität: Niedrig
Aufwand: 30-40 Stunden

- [ ] **Anomalie-Erkennung mit ML**
  - Trainieren auf historischen Daten
  - Automatische Pattern-Erkennung
  - Frühwarnsystem
- [ ] **Optimierungs-Vorschläge**
  - Sequenz-Optimierung
  - Timing-Optimierung
  - Ressourcen-Optimierung
- [ ] **Predictive Maintenance**
  - Hardware-Verschleiß vorhersagen
  - Wartungs-Empfehlungen

Technologien:

scikit-learn
TensorFlow/PyTorch (optional)

---

9. Kollaborations-Features 👥

Priorität: Niedrig
Aufwand: 16-24 Stunden

- [ ] Multi-User Support
- [ ] Projekt-Sharing
- [ ] Kommentare & Annotationen
- [ ] Versions-Kontrolle für Sequenzen
- [ ] Team-Dashboard

Nutzen: Zusammenarbeit im Team

🐛 BEKANNTE BUGS & FIXES

Kritisch

- [ ] **Keine bekannten kritischen Bugs** ✅

Mittel

- [ ] Dashboard-Widget-Positionierung manchmal nicht persistent
- [ ] Sehr lange Sequenzen (>1000 Schritte) verlangsamen UI

Niedrig

- [ ] Sensor-Polling-Interval nicht sofort aktiv (erst nach erstem Poll)
- [ ] Chart-Reset löscht nicht alle Datenpunkte sofort

🔍 CODE-QUALITY IMPROVEMENTS

Refactoring

- [ ] **Core-Module aufräumen**
  - Redundanten Code entfernen
  - Bessere Fehlerbehandlung
  - Type Hints hinzufügen
- [ ] **UI-Module standardisieren**
  - Einheitliches Styling
  - Gemeinsame Base-Klassen
  - Konsistentes Signal-Handling

Testing

- [ ] **Unit Tests schreiben**
  - Core-Module (>80% Coverage)

    - Analysis-Module (>70% Coverage)
- [ ] **Integration Tests**
    - Sequenz-Runner
    - Database-Worker
- [ ] **UI Tests (optional)**
    - PyQt Testing

Dokumentation

- [ ] **API-Dokumentation**
    - Docstrings vervollständigen
    - Sphinx-Dokumentation generieren
- [ ] **User Manual**
    - Benutzerhandbuch (PDF/HTML)
    - Video-Tutorials
- [ ] **Developer Guide**
    - Architektur-Übersicht
    - Plugin-Entwicklung
    - Contribution Guidelines

📦 PACKAGING & DISTRIBUTION

Standalone-Builds

- [ ] **Windows Executable**
    - PyInstaller Setup
    - Installer (NSIS oder Inno Setup)
- [ ] **macOS App Bundle**
    - .dmg erstellen
- [ ] **Linux Package**
    - .deb / .rpm
    - AppImage

Installation

- [ ] **pip installierbar machen**
    - setup.py / pyproject.toml
    - PyPI Upload
- [ ] **Auto-Update Mechanismus**

🎯 PRIORISIERUNGS-MATRIX

Feature	Nutzen	Aufwand	Priorität
Live-Stats Feinschliff	Hoch	Niedrig	⭐⭐⭐
Makro-System	Hoch	Mittel	⭐⭐⭐
Export-Funktionen	Mittel	Niedrig	⭐⭐
Sensor-Erweiterung	Mittel	Mittel	⭐⭐
Erweiterte Statistik	Hoch	Hoch	⭐⭐⭐
Hardware-Profile	Mittel	Mittel	⭐⭐
Remote-Zugriff	Niedrig	Sehr Hoch	⭐
Plugin-System	Mittel	Hoch	⭐⭐
ML Integration	Niedrig	Sehr Hoch	⭐
Kollaboration	Niedrig	Hoch	⭐

💡 QUICK WINS (< 2 Stunden)

Diese Features können schnell implementiert werden und bringen sofort Nutzen:

1. **Keyboard Shortcuts** ⚡
   - Strg+S: Sequenz speichern
   - Strg+R: Test starten
   - Strg+Q: Beenden
   - F5: Dashboard refresh
2. **Dark/Light Theme Toggle** 🌓
   - Theme-Switcher in Menü
   - Persistente Einstellung
3. **Letzte Aktionen anzeigen** 📜
   - "Recent Actions" Widget erweitern
   - Mehr Details zeigen
4. **Pin-Status Icons** 📌
   - Visuelle Indikatoren (🟢 🔴 🟡)
   - Im Pin-Overview
5. **Schnellstart-Favoriten** ⭐
   - Sequenzen als Favoriten markieren
   - Quick-Access im Dashboard

🎨 UI/UX IMPROVEMENTS

Design

- [ ] Moderneres Icon-Set
- [ ] Animierte Übergänge
- [ ] Bessere Farbschemata
- [ ] Responsives Layout

Usability

- [ ] Tooltips überall
- [ ] Context-Menüs (Rechtsklick)
- [ ] Drag & Drop für Sequenzen

- [ ] Undo/Redo für Sequenz-Editor

🔒 SICHERHEIT & STABILITÄT

Sicherheit

- [ ] Verschlüsselung für gespeicherte Configs
- [ ] Zugriffs-Kontrolle für Remote-API
- [ ] Audit-Log für alle Aktionen

Stabilität

- [ ] Bessere Exception-Handling
- [ ] Crash-Reporter
- [ ] Auto-Save bei Absturz
- [ ] Recovery-Modus

📊 METRIKEN & MONITORING

App-Metriken

- [ ] Nutzungs-Statistiken (lokal)
- [ ] Performance-Monitoring
- [ ] Error-Tracking

Hardware-Metriken

- [ ] CPU/RAM-Nutzung
- [ ] Serielle Verbindungs-Qualität
- [ ] Response-Zeiten

🌟 COMMUNITY FEATURES

- [ ] GitHub Issues Integration
- [ ] Forum/Discord einrichten
- [ ] Beispiel-Projekte sammeln
- [ ] Contributors anzeigen
- [ ] Changelog prominent zeigen

📝 NÄCHSTE SCHRITTE (Empfohlen)

Diese Woche:

1. ✅ Live-Stats stop_monitoring() fertigstellen (1h)
2. CSV-Export implementieren (3h)
3. Keyboard Shortcuts hinzufügen (1h)

Diesen Monat:

1. Makro-System vervollständigen (12h)
2. Sensor-System erweitern (6h)
3. Unit Tests schreiben (8h)

Dieses Quartal:

1. Erweiterte Statistik (16h)
2. Hardware-Profile (8h)
3. Dokumentation (12h)

💬 FEEDBACK & IDEEN

Hast du weitere Ideen?

- Erstelle ein GitHub Issue
- Füge es hier hinzu
- Diskutiere im Team

Kontakt:

- GitHub: [dein-repo]
- Email: [deine-email]

Zuletzt aktualisiert: 23. Oktober 2025
Version: 2.5.0

✅ CHANGELOG

v2.5.0 (23. Oktober 2025)

- ✅ Live-Statistik-Widget hinzugefügt
- ✅ Echtzeit-Monitoring von Testläufen
- ✅ Anomalie-Erkennung implementiert
- ✅ Trend-Analyse mit Icons
- ✅ Live-Chart für Zykluszeiten
- ✅ Dock-Widget Integration

v2.4.0 (vorher)

- Enhanced Dashboard
- Visual Editor für Sequenzen
- Board-Konfiguration
- Sensor-System Basis
- Makro-System Basis

Status: 🚀 Aktiv in Entwicklung
Maintainer: Drexler Dynamics Team