

**Hochschule Heilbronn**  
**University of Applied Sciences**

---

**Master Thesis Proposal**

**Modular Control Software for Photogrammetry 3D  
Scanner**

*Development and implementation of a modular C# control and workflow software for a photogrammetry 3D large object scanner*

*by Marc Nauendorf*

<b>Author:</b>	Marc Nauendorf
<b>Student ID:</b>	200882
<b>Program:</b>	Software engineering Master
<b>Faculty:</b>	Faculty IT
<b>Studiengang:</b>	HCI
<b>Research Lab:</b>	UniTyLab
<b>Supervisor:</b>	Prof. Dr. Gerrit Meixner
<b>Co-Supervisor:</b>	-
<b>Academic Year:</b>	2024/2025
<b>Submission Date:</b>	November 2025

## Table of Contents

- 1. Topic
  - 1.1. Problem Statement and Motivation
  - 1.2. Delimitation and Innovation
- 2. Objective of the Master Thesis
- 3. Specific Control Challenges
- 4. Evaluation Criteria and Validation
  
- 1. Topic
  - 1.1. Problem Statement and Motivation
  - 1.2. Delimitation and Innovation
- 2. Objective of the Master Thesis
- 3. Specific Control Challenges
- 4. Evaluation Criteria and Validation

# 1. Topic

**Development and implementation of a modular C# control and workflow software for a photogrammetry 3D large object scanner, with special consideration for an extensible widget-based UI design.**

## 1.1. Problem Statement and Motivation

The **creation of high-resolution 3D models** of life-size objects using photogrammetry requires the **precise, synchronized control** of a complex hardware setup consisting of numerous cameras, motion modules, and lighting.

Existing commercial and open-source solutions show significant limitations:

- **Monolithic software:** Control applications are often inflexible regarding different **hardware configurations** (number/type of cameras and motors) and do not allow easy **testing of novel scan routines**.
- **Architectural rigidity:** Existing **open-source solutions** such as **HSKanner** and **OpenScan** are valuable but focus on specific use cases:
- **HSKanner** uses an architecture with **fixed cameras** and a decentralized setup (camera and Raspberry Pi form a fixed unit), which severely limits scalability and dynamic adjustment of sensor positions.
- **OpenScan** offers a motion component (motor control) but is tailored to **single cameras** and smaller objects, with control taking place directly on the Raspberry Pi.

## 1.2. Delimitation and Innovation

This work addresses the need for a **powerful, scalable open-source solution** that overcomes these deficits.

While other solutions are stationary or small-scale, this project aims at controlling a **life-size, dynamically movable photogrammetry scanner** similar to professional large scanners, but with the goal of **hardware neutrality**.

The **core challenge and innovation** lies in developing a **modular control framework**. This framework must ensure **high modularity** of software components, allowing users to dynamically attach and execute new **control logics** (mathematical scan routines), **diagnostic tools**, and **integration functions** (e.g., Meshroom export) as "widgets" (plugins/scripts) in the main application at runtime.

This not only ensures technical flexibility for future extensions (e.g., live tracking modules) but especially solves the problem of **efficient testing and rapid exchange** of novel and

complex scan routines. The architecture enables the integration of **any type of hardware**—as long as it can be controlled via a standardized interface (e.g., REST API/UART)—into the scanner control.

## 2. Objective of the Master Thesis

The main goal is the conception and prototypical implementation of a **flexible and extensible control software** in C#.

- **C# software architecture:** Building a stable, **modularized C# application** (MVVM or XAML) for central control and data acquisition of the 3D scanner.
- **Hardware integration:** Implementation of communication modules for controlling **motion modules (REST API)**, **lighting (REST API)**, and capturing **camera streams (USB/HTTP)**.
- **Dynamic widget framework:** Development of a framework that enables **runtime integration** and execution of external, user defined C# or script modules (widgets/plugins) in the graphical user interface.
- **Core widgets:** Prototypical implementation of central functions as widgets to demonstrate modularity.

## 3. Specific Control Challenges

- **Asynchronous communication:** The software must **simultaneously** and **asynchronously** query a variety of REST APIs (motion, lighting) and process multiple **camera streams** (high bandwidth). This requires the use of **C# Task Parallel Library (TPL)** and efficient **multithreading** to avoid blocking the user interface.
- **Homogeneous control of heterogeneous modules:** The software must treat N cameras (planned: 12) with their **modules + light** as a unit (synchronized movement), even though they are individually addressed via REST interfaces.

## 4. Evaluation Criteria and Validation

- Successful control of all N modules and lighting (complete API coverage).
- Stable acquisition of all camera streams over a defined period.
- **Successful execution and correctness** of the generated scan routine (the mathematically calculated position must be reached by the modules).
- **Performance evaluation:**

- Measurement of **latency** when controlling motion modules via REST.
- Assessment of **CPU usage** of the C# software during multi-stream operation.
- **Architectural evaluation (modularity):**
  - The primary measure of success is the ability to add and execute a **new widget** (e.g., a **live tracking module** for future use) **without changes to the core code** of the application.