Artificial Intelligence for Smart Home Internet of Things

University of Turku
Department of Computing
Master of Science (Tech) Thesis
Software Engineering
June 2026
Samu Tuulos

UNIVERSITY OF TURKU
Department of Computing

SAMU TUULOS: Artificial Intelligence for Smart Home Internet of Things

Master of Science (Tech) Thesis, 64 p., 7 app. p.
Software Engineering
June 2026

Artificial Intelligence and Smart Homes have become increasingly prevalent in recent years. Smart Homes contain Internet of Things devices that generate large amounts of data, which can be used in Artificial Intelligence decision-making processes. This thesis researches how Artificial Intelligence is used in Smart Homes and the trade-offs between using Cloud Computing vs Edge Computing for AI. Two separate literature reviews were conducted to research these problems, screening the first 50 results from four databases for each review and analyzing the most relevant publications in depth. In total, 12 publications were analyzed in depth for the first research question and 8 for the second. Additionally, we developed our own machine learning application using Generative AI for an existing Smart Home environment. The application was developed almost entirely through Generative AI (Claude Code) with no manual changes to the source code. The application was deployed on a Raspberry Pi 5 with comparable performance to a Desktop PC. The trade-offs we found between Cloud and Edge Computing included trade-offs between computational power and latency, bandwidth usage, model training and inference, and privacy and security. However, few of the reviewed publications provided measured, empirical data to support these comparisons. The main use cases for Artificial Intelligence was found to be Smart Homes are energy management, elderly- and healthcare, home automation and anomaly detection. Overall, the results show that Artificial Intelligence has practical and varied applications in Smart Homes, and that both edge devices and Generative-AI-assisted development are mature enough to support them.

Keywords: Artificial Intelligence, Edge Computing, Cloud Computing, Smart Home, Smart Home Environments, Generative AI, Raspberry Pi, Internet of Things

Contents

1 Introduction

Artificial Intelligence and Smart Home technologies have experienced a sharp rise in popularity in recent years. Smart Homes contain Internet of Things devices that generate vast amounts of data, which can be utilized in Artificial Intelligence decision-making processes. The aim of this thesis is to research the use of Artificial Intelligence in Smart Home environments. The focus is on what can be achieved with AI and how it is implemented. The thesis also examines the trade-offs between using Cloud Computing and Edge Computing for Artificial Intelligence. Cloud Computing and data centers are becoming increasingly popular for AI applications; however, they involve several trade-offs when compared to Edge Computing. Beyond reviewing existing research, this thesis also explores how Generative AI can be used to build a Smart Home application in practice.

With the increasing number of Internet of Things devices and the data they produce, deciding where to run Artificial Intelligence becomes an important design factor. While Cloud Computing offers advantages with its nearly limitless processing power, Edge Computing has better latency and privacy. Understanding these technologies in the Smart Home space has become increasingly relevant.

1.1 Research Questions

The research is based on the following Research Questions:

RQ1: How is Artificial Intelligence used in Smart Home Environments?

RQ2: What are the trade-offs between using Edge Computing or Cloud Computing for Artificial Intelligence?

RQ3: How do you implement a face recognition access control system as a machine learning application on a Raspberry Pi using Generative AI?

These themes were selected due to their relevance in the current technological landscape. In addition, the literature review indicated that studies combining these topics into a single research work were limited.

1.2 Structure

In Chapter 2, we discuss Smart Homes, IoT, and Artificial Intelligence in general. We form a comprehensive overview of the technologies for the reader to have an understanding of them going into the later chapters. In Chapters 3 and 4, we aim to answer Research Questions RQ1 and RQ2 by conducting a literature review. In Chapter 5, we develop our own Smart Home AI application with the use of Generative AI and discuss its role in software development in the future.

1.3 Use of Generative AI

Generative AI had limited use in answering RQ1 and RQ2. It was mainly used for LaTeX formatting, for example, creating LaTeX tables or citations. Generative AI was used to answer RQ3, and the discussion is documented in Appendix A. Additionally, Generative AI was used to find writing errors and typos in the thesis.

2 Smart Home Internet of Things

In this chapter, we will go over general information about Smart Home Internet of Things technologies. We will also define the types of Artificial Intelligence discussed in this thesis so that the reader has an understanding of the topic moving forward.

2.1 The Internet of Things

The Internet of Things (IoT) is defined as a paradigm in which objects equipped with sensors, actuators, and processors communicate with each other to serve a meaningful purpose. IoT devices range from smart speakers that personalize the user experience to smart heating systems that optimize energy use. IoT is not a single technology, but rather an agglomeration of various technologies that work together in tandem. An IoT sensor can be broadly defined as any device which provides inputs about its current state [1]. The number of IoT devices is expected to grow significantly in the future. According to Statista, the number of IoT connections in 2025 was 19.8 billion, and the number is forecast to exceed 40.6 billion by 2034. [2]

IoT layers and Architecture

There is no single consensus on what the IoT architecture is, but it can roughly be divided into three different layers.

Application Layer is responsible for delivering application-specific services to the users. It defines various applications in which the Internet of Things can be deployed, for example, smart homes, smart cities, and smart health.

Network Layer is responsible for connecting to other smart things, network devices, and servers. It is also used for transmitting and processing sensor data.

Perception Layer is the physical layer which has sensors for sensing and gathering information about the environment. It senses physical parameters or identifies other smart objects in the environment. [1]

Some definitions on IoT architecture split the Network layer into the Transport layer and the Processing layer. The transport layer transfers the sensor data from the perception layer to the processing layer and vice versa. This layer includes technologies such as wireless, 3G, LAN, Bluetooth, RFID, etc. The processing layer stores, analyzes, and processes huge amounts of sensor data that comes from the transport layer. It includes technologies such as databases, cloud computing and big data processing modules.

2.2 Smart Home Environment

There are multiple different definitions for Smart Homes. Aldrich et al. defined it as follows:

> A "smart home" can be defined as a residence equipped with computing and information technology which anticipates and responds to the needs of the occupants, working to promote their comfort, convenience, security and entertainment through the management of technology within the home and connections to the world beyond. [3]

Smart Home and Internet of Things technologies go hand in hand. Smart Homes can contain any number of IoT devices with the aim of improving the occupants' lives in various ways. Smart Home devices such as smart speakers, smart TVs, smart fridges, etc., have gained popularity. Artificial Intelligence is used in Smart Homes to further improve the residents' living conditions. We explore how Artificial Intelligence is used in Smart Homes in more detail in Chapter 3.

2.3 Types of Artificial Intelligence

Artificial Intelligence (AI) is generally used as an umbrella term that encompasses multiple different types of AI. Artificial Intelligence can be categorized into three different types based on its capabilities. These types are as follows [4]:

Artificial Narrow AI is the only type of artificial intelligence that exists currently. Other forms of AI are theoretical. It can be trained to perform a single or narrow task, often faster and better than humans can. It cannot, however, perform outside of its defined tasks. It targets a single set of cognitive abilities and advances in that spectrum.

Artificial General Intelligence is a theoretical concept. It can use previous learning and skills to accomplish new tasks in a different context without the need to train the underlying models.

Super AI is even more theoretical than Artificial General Intelligence. If it were to exist, it would think, reason, learn and possess more cognitive abilities than human beings.

In this thesis, we will focus only on Artificial Narrow AI. This type of AI can be further divided into different types of AI.

The primary technologies of artificial intelligence can be divided into the following:

Machine learning (ML) is a core branch of AI that allows systems to learn from data without the need for specific programming for each task. By identifying patterns in training data, ML models improve their performance over time and can generalize to make predictions on previously unseen inputs.

Deep learning (DL) is a subset of ML that mimics the structure of the human brain using artificial neural networks (ANNs). These networks enable machines to recognize complex patterns and make sophisticated decisions.

Natural Language Processing (NLP) is a branch of AI that focuses on enabling machines to understand, interpret, and respond to human language. Combining linguistics with machine learning, it allows computers to process text and speech, facilitating communication between humans and machines.

Computer Vision (CV) is a branch of AI that enables computers and systems to process, analyze, and interpret visual inputs such as images and videos. It is commonly applied to tasks such as image classification, object detection, and facial recognition.[5] [6]

These primary technologies are not mutually exclusive, as many real-world systems combine several of them. There are also more specialized sub-technologies, many of which appear in the publications reviewed in this thesis. These include [7]:

Learning algorithms:

- Reinforcement Learning (RL)
- Federated Learning (FL)

Neural network architectures:

- Convolutional Neural Networks (CNN)

- Recurrent Neural Networks (RNN)
- Long Short-Term Memory (LSTM) networks

Classical ML algorithms:

- Support Vector Machines (SVM)
- Decision Trees
- Random Forests
- K-means Clustering

2.4 Edge Computing and Cloud Computing

Every AI application requires data processing and computational power. There are different approaches to meet this need. One solution is to move the data to the cloud and use cloud computing to process it and then send it back to the device. The other solution is to move the processing closer to the source, performing it on the edge.

According to research firm Gartner, "cloud computing is a style of computing in which scalable and elastic IT-enabled capabilities are delivered as a service using internet technologies". [8]

The advantages of cloud computing are:

- Lower upfront cost
- Flexible pricing
- Limitless compute on demand
- Simplified IT management
- Reliability

Edge computing is the practice of moving computing power closer to where the data is generated, usually to an IoT device or sensor. Edge computing allows

for faster data processing, increased bandwidth, and improved data security and sovereignty.

The advantages of edge computing include the following:

- Lower latency
- Reduced cost
- Improved model accuracy
- Data sovereignty

Many AI solutions use a mix of Edge and Cloud Computing techniques, leveraging both to utilize their respective advantages for optimal results. We will dive further into the trade-offs between using Edge and Cloud computing for AI in Chapter 4.

3 Artificial Intelligence for Smart Homes: Literature Review

Artificial intelligence has seen numerous advances in the past few years. In this chapter, we will answer RQ1 by conducting a literature review on how Artificial Intelligence can be used in Smart Home Environments. In the literature review, we first formulate a search query, then we define the journals we are searching from, and then select the relevant publications based on the search results.

3.1 Methodologies

3.1.1 Literature Sources

The literature sources used for this review are:

- Google Scholar
- Springer
- IEEE Xplore
- UTU Volter, the search engine used by University of Turku

3.1.2 Search Query

This literature review aims to answer three questions on the publications that we select for the review. These three questions are:

1. What is the purpose of the Artificial Intelligence?
2. What environment or ecosystem is the Artificial Intelligence applied in?
3. What Artificial Intelligence methods or techniques were used?

To find relevant sources we need to formulate a search term:

The search query includes key phrases **Smart Home** and **"Artificial Intelligence" OR "AI"**. The terms (Usage OR Implementation OR Application) are also included to limit the results to implementations and not concepts.

The final search query is: **"Smart Home" AND ("Artificial Intelligence" OR "AI") AND (Usage OR Implementation OR Application)**

Google Scholar

The initial search on Google Scholar resulted in 27 200 publications. From reading the title and abstract from the first 50 publications, I found 24 relevant publications, from which 9 were dropped due to overlapping topics or insufficient specificity. This leaves 15 publications for the next stage of the literature review. A limitation of using Google Scholar is the breadth of the search, making it difficult to reproduce this search later on.

UTU Volter

In UTU Volter, the initial search resulted in 1 913 publications. After reviewing the first 50 publications based on the title and the abstract, 6 relevant publications were found, from which 4 had to be dropped due to being duplicates from previous

searches or having too much overlap with already selected publications. This leaves 2 publications for the next stage of the literature review.

Springer

In Springer, the initial search resulted in 6 216 publications. Out of the first 50 results, 4 publications were relevant, from which 1 had to be dropped due to being a duplicate result from the Google Scholar search. This leaves 3 publications for the next stage. The search term provided worse results than in Google Scholar.

IEEE Xplore

In IEEE Xplore, the initial search resulted in 1 432 publications. Out of the first 50 results, 8 publications were found to be relevant, from which 4 had to be dropped due to topic overlapping or insufficient relevance to the research topic, leaving 4 publications for inclusion.

Results of the Search

Publication	Results from search	Relevant publications
Google Scholar	27 200 (50)	15
UTU Volter	1 913 (50)	2
Springer	6 216 (50)	3
IEEE Xplore	1 432 (50)	4
Total	36 761 (200)	24

Table 3.1: Overview of publications found in the literature search

3.2 Results of the Search

Every article selected from the search is compiled in Table 3.2. They are in the same order that they were found during the search.

Index	Title
[$P1_{AI}$]	Investigating the Impact of AI/ML for Monitoring and Optimizing Energy Usage in Smart Home
[$P2_{AI}$]	Strategic decision making in smart home ecosystems: A review on the use of artificial intelligence and Internet of things
[$P3_{AI}$]	Artificial Intelligence of Things (AIoT) Enabled Floor Monitoring System for Smart Home Applications
[$P4_{AI}$]	Artificial Intelligence-Based Smart Security System Using Internet of Things for Smart Home Applications
[$P5_{AI}$]	Artificial intelligent system for multimedia services in smart home environments
[$P6_{AI}$]	AI-Powered Energy Consumption Optimization for Smart Homes Using IoT
[$P7_{AI}$]	An Artificial Intelligence based scheduling algorithm for demand-side energy management in Smart Homes
[$P8_{AI}$]	Energy Community Management Based on Artificial Intelligence for the Implementation of Renewable Energy Systems in Smart Homes
[$P9_{AI}$]	Artificial-Intelligence-Assisted Activities of Daily Living Recognition for Elderly in Smart Home
[$P10_{AI}$]	Design and Implementation of an AI-based & IoT-enabled Home Energy Management System: A case study in Benguerir — Morocco
[$P11_{AI}$]	A voice controlled smart home automation system using artificial intelligent and internet of things
[$P12_{AI}$]	Application of Deep Learning and Intelligent Sensing Analysis in Smart Home
[$P13_{AI}$]	AI based elderly fall prediction system using wearable sensors: A smart home-care technology with IoT
[$P14_{AI}$]	Edge AI for Real-Time Anomaly Detection in Smart Homes
[$P15_{AI}$]	Artificial Intelligence Method for the Forecast and Separation of Total and HVAC Loads With Application to Energy Management of Smart and NZE Homes
[$P16_{AI}$]	Data-Driven Smart Home Automation for Energy Efficiency
[$P17_{AI}$]	Enhancing Smart Home Energy Efficiency Using a Hybrid Genetic Algorithm and Improved Dandelion Optimizer
[$P18_{AI}$]	Data autonomy and privacy in the smart home: the case for a privacy smart home meta-assistant
[$P19_{AI}$]	An Efficient Implementation of an IoT-Based Smart Home Security System
[$P20_{AI}$]	An AI Smart Refrigerator: Enhancing Food Management with Computer Vision, Sensor, IoT, and Automated Temperature Control
[$P21_{AI}$]	Development of Smart Home System Based on Artificial Intelligence with Variable Learning Rate to Manage Household Energy Consumption
[$P22_{AI}$]	An AIoT-Enabled Smart Home Automation Framework: Intelligent Control, Security, and Predictive Monitoring
[$P23_{AI}$]	Opto-Electronic Smart Home: Heterogeneous Optical Sensors Approaches and Artificial Intelligence for Novel Paradigms in Remote Monitoring
[$P24_{AI}$]	A novel intelligent technique for energy management in smart home using internet of things

Table 3.2: Articles selected from the literature search

3.2.1 Selecting the Articles

As previously stated, we will analyze the publications based on three metrics.

- What is the purpose of the Artificial Intelligence?
- What environment or ecosystem is the Artificial Intelligence applied in?
- What Artificial Intelligence methods or techniques were used?

If the article does not contain all of these research metrics, it will be discarded from the final literature review.

Below is a table where we select the final articles based on the three categories of data in them.

Table 3.3: Analysis of Articles for RQ1

Index	AI Purpose	Environment / Ecosystem	AI Methods & Techniques	Selected
[$P1_{AI}$]	Enhance energy management in smart homes	Smart Home Energy Management Ecosystem	Regression Learning, Deep Learning (DL), Reinforcement Learning (RL)	No
[$P2_{AI}$]	Strategic decision support, ecosystem design, data interpretation and value creation analysis	Smart Home, Corporate, Insurance	Machine Learning (ML), Natural Language Processing (NLP), Computer Vision (CV), DL	No
[$P3_{AI}$]	DL assisted data analytics for smart-floor system	Smart Home / Smart Office / Building IoT ecosystem	DL, Convolutional Neural Network (CNN)	No
[$P4_{AI}$]	AI-based Smart Security System	Smart Home Edge Device	OpenCV	No
[$P5_{AI}$]	Improve user Quality of Experience for multimedia devices	Smart Home Environment	RL, DL	Yes
[$P6_{AI}$]	Forecasting energy demands and optimizing usage	IoT-enabled smart home	ML and DL models	No
[$P7_{AI}$]	Reach compromise between energy cost and user comfort	Smart Homes	Elitist Non-dominated Sorting Genetic Algorithm II, Support Vector Regression	Yes
[$P8_{AI}$]	decrease reliance on power grid and enhance the system's economic efficiency	smart homes with solar-battery integration	Deep Reinforcement learning, Q-learning algorithm	Yes
[$P9_{AI}$]	Prediction of Activities of Daily Living for Elderly	Smart Home Environment	modified Naive Bayes supervised learning algorithm	Yes
[$P10_{AI}$]	Renewable energy integration and energy efficiency improvement	testbed house of Smart Campus	BPSO technique	No
[$P11_{AI}$]	Automated Home Appliance Control	Smart Home System	NLP, ML	Yes
[$P12_{AI}$]	User behavior recognition based on optical sensors	Complex home environments	DL technology	No

Continued on next page

Index	AI Purpose	Environment / Ecosystem	AI Methods & Techniques	Selected
[P13$_{AI}$]	Elderly fall prediction	smart home-care technology	Long Short-Term Memory (LSTM) Model	Yes
[P14$_{AI}$]	Real-time anomaly detection	Smart Home environments	Isolation Forest (IF) and LSTM Autoencoder	Yes
[P15$_{AI}$]	Separate dominant HVAC load	Regular Homes	LSTM encoder-decoder ML models	Yes
[P16$_{AI}$]	identify device usage patterns, detect inefficiencies and develop intelligent automation rules	modern residential environments	ML	Yes
[P17$_{AI}$]	Intelligently schedule and manage household appliances	IoT-enabled smart homes	Hybrid Genetic Algorithm, Improved Dandelion Optimizer	No
[P18$_{AI}$]	Increase users' data autonomy and privacy	Smart Home environments	Meta-assistant	No
[P19$_{AI}$]	Detect and Respond to security threats	"Various home sizes and configurations"	K-means clustering	No
[P20$_{AI}$]	Recognize stored items, track inventory and predict expiry dates	Smart fridge	computer vision and machine learning algorithms	No
[P21$_{AI}$]	Manage energy consumption	AI-controlled SH device	ANN with Back Propagation, Variable Learning Rate	Yes
[P22$_{AI}$]	Real-time control, high-level security, predictive monitoring	SH environments	ML processes	No
[P23$_{AI}$]	Patient activity recognition and gait analysis	Opto-electronic smart home	Feedforward neural network (FFNN)	Yes
[P24$_{AI}$]	Optimal IoT based Energy Management	Smart Home Control System	hybrid wrapper of both Sailfish Optimizer and Adaptive Neuro-Fuzzy Interference System	Yes

3.3 Analysis

The publications we have selected for our literature review are [P5$_{AI}$], [P7$_{AI}$], [P8$_{AI}$], [P9$_{AI}$], [P11$_{AI}$], [P13$_{AI}$], [P14$_{AI}$], [P15$_{AI}$], [P16$_{AI}$], [P21$_{AI}$], [P23$_{AI}$], [P24$_{AI}$].

[P5$_{AI}$] - Artificial intelligent system for multimedia services in smart home environments presents an AI-based smart system for managing multimedia devices such as video streaming, VoIP, and web browsing within a smart home network. The research problem that the publication addresses is that deep learning classifiers for network resource management produce misclassifications, degrading the Quality of Experience of the users.

The Smart Home Environment in the article is a five-layer Internet of Things architecture consisting of Internet/Cloud, Management, Assistants, Things, and Sensors. The AI system is embedded in the management layer.

The AI solution combines deep learning and reinforcement learning methods. The Deep learning classifier identifies which type of multimedia service is currently in use, and a Reinforcement Learning agent acts as a supervisor, deciding when and how to correct the actions of the classifier.

[P7$_{AI}$] - An Artificial Intelligence based scheduling algorithm for demand-side energy management in Smart Homes presents a new methodology which combines three different Artificial Intelligence techniques to solve the energy demand planning in Smart Homes. The methodology was developed to reach the compromise between energy cost and user comfort.

It uses an Elitist Non-dominated Sorting Genetic Algorithm II to search for scheduling options that balance energy costs within the level of comfort that the user expects. Afterwards it uses a Support Vector regression to predict how much energy the Smart Home's generator will produce in the near future and finally K-means clustering to group typical household usage patterns so that the system can estimate what level of comfort the user prefers.

The AI system operates in a Smart Home with a local generator and a battery system that is connected to a smart grid. The energy management system coordi-

nates when devices run and how energy is drawn from the generator and when from the grid.

The system was able to provide a 51.4% cost reduction when compared to Smart Homes without distributed generation and battery banks. The data was validated with simulations using real data obtained from a smart home, meaning that it has not been proven to work in a real-world environment.

[$P8_{AI}$] - Energy Community Management Based on Artificial Intelligence for the Implementation of Renewable Energy Systems in Smart Homes proposes an AI-based management system for coordinating energy production, storage and exchange in a community of Smart Homes. The research problem addressed is how to "oversee the exchange, organization, and control of the community's household electricity consumption effectively" while reducing costs and improving renewable energy utilization.

The AI-based management system operates in a smart-home energy community which contains "six identical prosumers, encompassing residential energy use, solar photovoltaic (PV) installations, and battery storage systems".

The AI-based management system uses multi-agent deep reinforcement learning as its core AI-technique. [$P8_{AI}$] states that the optimal performance of the model is achieved by "using a multi-agent deep reinforcement learning algorithm within a decentralized training environment", where prosumers and consumers learn how to schedule appliances, charge and discharge batteries and participate in peer-to-peer energy trading.

The results of the study show that the proposed AI-based management system is able to improve the smart home community's economic performance. According to the authors, the intelligent method "has successfully decreased consumer electricity costs in various circumstances", indicating that both consumers and prosumers benefit from the system.

[P9$_{AI}$] - Artificial-Intelligence-Assisted Activities of Daily Living Recognition for Elderly in Smart Home presents an AI assisted Activities of Daily Living recognition system for elderly. It uses Activity Recognition to identify a certain activity from the set of actions. The research problem it addresses is that "due to the unpredictable behaviors of an elderly person, performance of ADLs can vary in day-to-day life. Each activity may perform differently, which can affect the sequence of the sensor's raw data." In conclusion, the aim of the system is to monitor the elderly and detect potentially dangerous health conditions or actions, for example falling down.

The environment is a Smart Home environment that is equipped with "a wide range of smart sensor devices". These sensors may include motion sensors, door sensors, light sensors, temperature sensors, etc. The data is continuously collected and saved into the database. The system is designed for a smart home with a single resident.

The AI method used in this publication is a modified Naive Bayes supervised learning algorithm. The proposed method contains three different stages:

1. Data Relabeling - the raw sensor data are allocated by marking the starting point and the end point.

2. Feature Extraction - deriving features from windowed sensor data.

3. Naive Bayes classification - using the features to train a supervised model to predict ADL categories.

The authors mentioned that Deep Learning was intentionally avoided because it is "still considered a complex technique to be applied in analyzing raw sensor data".

The research showed that the proposed AI assisted ADL recognition method achieves an effective and highly accurate performance in identifying the daily activi-

ties of an elderly person. The system achieved much better metrics than comparable methods. The metrics are displayed in Table 3.4.

Table 3.4: Evaluation metrics for comparing with different algorithms. Found in [$P9_{AI}$]

Evaluation Metrics	Proposed Method	SVM	K-NN	LR
Accuracy	0.892 (±0.03)	0.625 (±0.12)	0.422 (±0.1)	0.467 (±0.05)
Recall/Sensitivity	0.824 (±0.06)	0.571 (±0.02)	0.543 (±0.03)	0.395 (±0.02)
Precision	0.861 (±0.02)	0.569 (±0.02)	0.522 (±0.05)	0.338 (±0.02)
F1-score	0.836 (±0.04)	0.555 (±0.01)	0.484 (±0.03)	0.360 (±0.02)
Average	0.853 (±0.03)	0.580 (±0.04)	0.492 (±0.05)	0.390 (±0.02)

The metrics in Table 3.4 show that the proposed Naive Bayes-based solution performed better than the traditionally used machine learning models. Overall, the table shows that the proposed model is able to recognize the daily activities of an elderly resident far more reliably and consistently than the alternative approaches.

[$P11_{AI}$] - A voice controlled smart home automation system using artificial intelligent and internet of things presents a voice controlled AI smart home system that directs smart home devices through an Android or a web application. The publication aims to improve existing home automation solutions which rely on rigid command sequences and limited interfaces by "incorporating voice control and artificial intelligence (AI) into internet of things (IoT)-based smart home systems to create more efficient automated smart home systems". The primary aim of AI in this publication is to enable the use of natural language to automate IoT devices and processes based on user preferences. The system operates in an IoT-based Smart Home Environment, where Arduino, Raspberry Pi, and NodeMCU are used to interface with home appliances and cloud services. In this environment, multiple devices and sensors are connected through cloud services, where the MQTT protocol and REST API enable real-time communication between the different components. When the user gives a voice command, it is converted into text and processed with

Natural Language Processing (NLP). The NLP-module is divided into eight different layers:

1. Layer 1: expanding contractions
2. Layer 2: Tokenization
3. Layer 3: spelling correction
4. Layer 4: intent and tense detection using a Naïve-Bayes classifier algorithm
5. Layer 5: part-of-speech tagging
6. Layer 6: information extraction
7. Layer 7: organization
8. Layer 8: execution

After the speech is converted into text, a Machine Learning based recommender system using a "content-based filtering technique" and the "nearest neighbour algorithm" is used to infer user preferences and automate actions such as movie recommendations or appliance control. Although the publication does not present numerical metrics, the authors claim it is tested in a real-world environment. The publication presents a prototype level solution which demonstrates the feasibility of an NLP-based speech recognition and a Machine Learning based recommendation system in Smart Home automation.

[P13$_{AI}$] - AI based elderly fall prediction system using wearable sensors: A smart home-care technology with IoT presents an IoT-based fall detection and prediction system which utilizes wearable sensors and Deep Learning to detect fall-events among the elderly. The system uses a low-power 3-axis accelerometer embedded in a 6LoWPAN wearable device, which continuously collects movement data.

The proposed method utilizes a Long short-term memory (LSTM)-based Deep Learning approach that focuses on stacking. This means that multiple LSTM models are trained for different tasks with small overlap, and the results are combined into one AI method. This yields more accurate results compared to a single LSTM model. The effectiveness of the LSTM model is listed in Table 3.5.

Table 3.5: Effectiveness of the LSTM model across different datasets.

Experiments	**Precision**	**Recall**	**Overall accuracy (%)**
MobiAct	0.98	0.98	99.63
Waist (50 Hz)	0.76	0.65	86.3
Waist + leg (calf)	0.96	0.95	95.2
Waist + Wrist	0.97	0.97	96.6

The results of 3.5 demonstrate that the proposed LSTM-based ensemble approach performs exceptionally on the MobiAct dataset, with an overall accuracy of 99.63%. The results also demonstrate that sensor placement and the combining of multiple sensors have a positive impact on the performance of the model. Overall, the results show that the proposed method offers a reliable solution to real-time fall detection.

[P14$_{AI}$] - Edge AI for Real-Time Anomaly Detection in Smart Homes presents an Edge-AI based anomaly detection framework, which analyses the data provided by IoT-sensors in real time on edge devices. The study addresses the limitations of cloud-based anomaly detection, including latency issues, privacy risks, and network reliance, by performing the computation locally on edge devices.

The framework uses a hybrid approach which combines Isolation Forest (IF) and LSTM Autoencoder (LSTM-AE) models to identify anomalies in IoT sensor data. The IF model is a light weight, unsupervised machine learning algorithm designed for efficient anomaly detection in high-dimensional datasets. The model identifies outliers by isolating data points that significantly deviate from normal patterns.

The LSTM-AE model recognizes long-term irregularities, such as gradual energy consumption spikes or repetitive but anomalous activity patterns.

The system is evaluated on synthetic and real-world smart home datasets, including temperature, motion, and energy consumption signals. The dataset was split into an 80/20 train-test ratio, ensuring that anomalies were proportionally distributed in both sets. The inference testing was performed on a Raspberry Pi 4 with 4GB RAM.

The experimental results show that the proposed hybrid approach is suitable for real-time anomaly detection on the edge. The IF model offers a fast analysis, while the LSTM-AE refines the detections by analyzing long-term behavioral patterns. Together these models allow for both sudden and gradual anomaly detection without the need to transfer data to the cloud. This improves the system's latency and privacy, making the approach well suited for Smart Home real-time anomaly detection.

[P15$_{AI}$] - Artificial Intelligence Method for the Forecast and Separation of Total and HVAC Loads With Application to Energy Management of Smart and NZE Homes proposes a method that uses deep learning techniques to separate Heating, Ventilation, and Air Conditioning (HVAC) energy use from the total residential load. The goal is to improve energy-use monitoring and enhance home energy management systems (HEMS), particularly in existing houses that do not have dedicated HVAC submetering.

The method employs a novel LSTM encoder–decoder model, which is trained using future weather data in place of conventional weather forecasts. The model receives input data from the two previous days of power usage and generation and produces a prediction for the following day, covering the period from 6:00 to 21:00.

A key contribution of this publication is the separation of HVAC load from the total residential load in homes that lack HVAC submetering. The method relies

on generating two LSTM-based forecasts: a total-load forecast using weather inputs, and a 'baseload' forecast representing household consumption under conditions where the HVAC system is not expected to operate. The baseload forecast is produced by setting the weather and solar-irradiance inputs to values below the HVAC activation threshold (TmHVAC), enabling the model to estimate the non-HVAC portion of the load. The final HVAC load is obtained by subtracting the baseload forecast from either the total-load forecast or the measured total load.

The results show that the LSTM model predicts HVAC and PV loads accurately: the daytime CV(RMSE) values (HVAC 18.8%, PV 7.1%) meet the ASHRAE building-modeling standards. The HVAC separation method also produces usable results, although its accuracy does not fully match that of models trained on measured HVAC data. Nevertheless, the method is sufficiently accurate for HEMS optimization, load management, and providing user-level energy information.

Overall, the results of the study show that LSTM-based deep learning models are well suited for load prediction and additionally enable HVAC-load separation in houses without HVAC submetering. This makes the approach applicable to a wide range of buildings where installing dedicated HVAC submetering is not economically or technically feasible.

[P16$_{AI}$] - Data-Driven Smart Home Automation for Energy Efficiency aims to improve energy efficiency by analyzing a four-year Smart Home dataset to identify device usage patterns, detect inefficiencies and develop intelligent automation rules aligned with user behaviors. According to the study, Smart home technologies driven by Internet of things and Artificial intelligence promise automation, energy efficiency, and personalized experience, but their potential is limited by persistent challenges such as inefficient energy use due to static control algorithms, a lack of control systems that respond to user behavior, and limited interoperability among diverse devices.

The machine learning methods the study uses include decision trees, random forests, and rule-based modeling to create automation rules. The research process flow contains four stages:

- Data preprocessing - cleaning and preparing the dataset, addressing missing values and outliers.
- Exploratory Data Analysis - identifying data trends and correlations relevant to the objectives.
- Model Development - using machine learning and rule-based techniques to create intelligent automation solutions.
- Testing and Validation - simulating automation rules and monitoring performance using real-time dashboards in Grafana, an open-source analytics and visualisation app.

Simulation results demonstrate that the proposed automation rules can reduce energy consumption by 15-25% while the user's needs are met in the smart home context. The study also emphasizes the importance of privacy in the smart home space. The collection of large amounts of sensor data requires anonymization and secure data storage.

[P21$_{AI}$] - Development of Smart Home System Based on Artificial Intelligence with Variable Learning Rate to Manage Household Energy Consumption presents an AI-controlled smart home system to manage monthly energy consumption based on user-defined limits. The authors note that existing smart home configurations rely largely on manual control and are not fully automated. A key background factor in this study is the Indonesian government's energy conservation policy.

The system developed in the study combines IoT devices, sensor data and an Artificial Neural Network based control model. The system uses an ESP32 microcontroller, which collects real-time data on lighting, temperature, occupancy, and energy consumption. Raspberry Pi 4 was used for data storage and model inference. The user can control the system through an Android application, which provides a real-time view of energy consumption and device management.

The control model is based on an Artificial Neural Network (ANN) trained using the Backpropagation (BP) method and enhanced through Variable Learning Rate (VLR) algorithm. The inputs for the Neural Network are light intensity, temperature, occupancy, and the user-defined monthly energy consumption target. The outputs of the model are the four electrical components - light, television, air conditioning, and refrigerator - optimal daily usage times. The VLR algorithm improves training stability and accuracy by dynamically adjusting the learning rate based on the error behaviour.

The experimental results show that the system functions reliably and is able to provide use-time recommendations with an average error rate of 17.21% in relation to the monthly consumption target. The Neural Network's RMSE-values (14.6-15.9%) show that the VLR-method improves the accuracy of the model when compared to the traditional Back Propagation method. The system is able to control the devices automatically based on the generated recommendations, and the user can make manual adjustments in the application.

The research shows that an IoT-based, neural-network-driven control system can support residential energy saving and provide the user with cost-optimized usage strategies.

[P23$_{AI}$] - Opto-Electronic Smart Home: Heterogeneous Optical Sensors Approaches and Artificial Intelligence for Novel Paradigms in Remote Monitoring presents the development and implementation of an

optical-fiber-integrated smart environment with heterogeneous opto-electronic approaches. The opto-electronic smart home, composed of three different optical fiber sensor systems, is used for remote patient monitoring. The system consists of nanoparticle (NP)-doped TRA (Transmission–Reflection Analysis) fiber sensors for patient localization, polymer optical fiber (POF)–integrated pants for activity detection, and a fiber Bragg grating (FBG)-embedded smart carpet for gait analysis. The system includes more than 50 integrated sensors, enabling remote monitoring and activity tracking without cameras and thereby preserving user privacy.

The TRA-based sensors utilize NP-doped silica optical fibers, which are positioned in strategic regions inside the smart home and allow for accurate assessment of the patient's location within the environment. The POF Smart Pants have two POFs incorporated into the back of the pants (left and right leg), with a total of 60 intensity-variation-based multiplexed POF sensors. These sensors enable monitoring of activities such as walking, sitting down, and lying. The FBG-based carpet allows for monitoring step location, stride length, double-support duration, and stance-phase duration.

The three systems are synchronized, and data acquisition is managed through a graphical interface. The interface supports both online and offline operation. FBG-carpet data are transmitted via Wi-Fi, the POF Smart Pants data via Bluetooth, and the TRA-based smart-environment data via serial communication. Deep learning models handle both data preprocessing and high-level decision-making, although the publication does not provide a detailed description of their internal workings.

Overall, the findings demonstrate that combining a heterogeneous optical sensor network with deep-learning-based data processing enables reliable detection of a user's location, activities, and gait parameters within the home environment. The system provides a novel, privacy-preserving approach to remote monitoring and is

particularly well suited for supporting the elderly, individuals undergoing rehabilitation, and people with chronic health conditions.

[P24$_{AI}$] - A novel intelligent technique for energy management in smart home using internet of things presents an IoT-based smart home energy management system that uses a hybrid approach combining the Sailfish Optimizer (SFO) and the Adaptive Neuro-Fuzzy Inference System (ANFIS) into a single method, commonly referred to as the SFOANFIS method. The proposed approach optimally manages the power and resources of the distribution system.

In the method, every household appliance is connected to the cloud-based Internet of Things, each with a unique IP address. A centralized server collects usage data from all appliances, and the SFOANFIS method manages the collected data. The ANFIS model predicts load based on time intervals, while the Sailfish Optimizer searches for the optimal energy-management strategy using the predicted load. The method aims to reduce costs and ensure that energy production, storage, and consumption remain balanced.

The performance of the SFOANFIS method was compared against existing optimization algorithms (ANFASO, SOGSNN, and SFO). The results show that the proposed hybrid approach achieves faster inference times and higher optimization accuracy at the day, week, and year scales. It proves to be an efficient energy-management solution, particularly in IoT-based supply networks where the number of connected devices is large and the data are highly distributed.

3.4 Conclusion

In this chapter, we have done a literature review on the application of Artificial Intelligence in a Smart Home Environment. We conducted searches using four different search engines and reviewed the first 50 results returned by each. From the

200 publications identified, 24 were selected for a more in-depth review, of which 12 were ultimately included in the analysis. The criteria were that the publication had to describe how AI was used, the environment in which it operated, and the AI methods it employed.

We have shown that Artificial Intelligence has multiple different uses in a Smart Home environment. The original aim was to include only solutions that are already in use in some form, but an issue is that most of the AI solutions were methodologies or algorithms that are not in use in real world scenarios. Out of 12 publications, 6 focused on energy management [$P7_{AI}$], [$P8_{AI}$], [$P15_{AI}$], [$P16_{AI}$], [$P21_{AI}$], [$P24_{AI}$], 3 focused on elderly and healthcare [$P9_{AI}$], [$P13_{AI}$], [$P23_{AI}$], 2 focused on home automation [$P5_{AI}$], [$P11_{AI}$], and one was about anomaly detection [$P14_{AI}$]. In this section, we will go over the core findings of our literature review.

Energy Management With the increasing global demand for energy as everything goes digital around the world, energy management has become increasingly important in the smart home sector. The publications in our review were able to show meaningful energy savings with their methods. Rocha et al. [$P7_{AI}$] were able to achieve 51.4% cost reduction with their AI based method. Houshangi [$P16_{AI}$] demonstrated a 15-25% reduction in energy consumption with the proposed automation rules. Alden et al. [$P15_{AI}$] proposed an approach to energy management by using an LSTM-encoder-decoder model to separate HVAC-load from the total energy load in homes that lack a separate HVAC-submetering. The aim of the method is to improve energy usage monitoring and to enhance the house energy management systems (HEMS) for existing houses that do not have dedicated HVAC circuits. Akhinov et al. [$P21_{AI}$] used a back propagation with a variable learning rate system to manage the electrical consumption of household appliances to stay within user defined limits. Rajesh et al. [$P24_{AI}$] proposed an IoT-based energy management system that combines a Sailfish Optimizer-algorithm with the Adaptive Neuro-Fuzzy

Inference System -model (SFOANFIS) to optimize household energy consumption and to balance production, consumption, and load in the distribution system.

Elderly and Healthcare Three publications addressed the application of AI to elderly care and health monitoring within smart home environments. Onthoni and Sahoo [$P9_{AI}$] proposed a sensor-based ADL-recognition model that utilizes data relabeling, feature extraction, and a modified Naive Bayes classifier to monitor the daily activities of the elderly. Kulurkar et al. [$P13_{AI}$] used an AI-based elderly and fall prediction system that uses a wearable 3-axis accelerometer and LSTM-based deep learning to detect fall events with an accuracy of 99.63%. Leal-Junior et al. [$P23_{AI}$] developed an Opto-Electronic smart home environment that combines three different sensor systems (TRA-based sensors, FBG-smart carpet and POF-smart pants). The system detects the residents' location, classifies activities with deep learning, enabling multi-layer and real-time monitoring while protecting the users' privacy.

Home Automation Two publications addressed home automation. Rego et al. [$P5_{AI}$] proposed an AI-based system for multimedia services with the goal of improving users' Quality of Experience. Torad et al. [$P11_{AI}$] proposed a voice-controlled smart home system, which combines IoT devices, Natural Language Processing and a machine learning based recommendation system to improve home automation. The system allowed users to control home appliances with their speech using an Android or a Web application.

Anomaly Detection One publication addressed anomaly detection in a smart home environment. Reis et al. [$P14_{AI}$] proposed an Edge-AI-based anomaly detection framework, which analyzed the data generated by IoT-sensors in real time on edge devices. The framework was capable of detecting short- and long-term

anomalies, including energy consumption, activity patterns, temperature and motion signals. This demonstrates how AI can enhance the security of smart home environments by identifying unusual behavior and potential system issues in real time.

3.4.1 Answer to RQ1

The most prominent use case for Artificial Intelligence in a Smart Home Environment is Energy Management. This is unsurprising, as rising energy demand is a global challenge with no immediate solution in sight. With the residential housing sector accounting for 21% [9] of energy demand worldwide, having even modest reductions in consumption would make a meaningful difference. The second prominent use case was healthcare and activity monitoring. With the aging population in many Western societies, having more resources to monitor residents for potential health issues and emergencies can help alleviate the stress on the healthcare system. Other use cases include home automation and anomaly detection. These solutions allow for better user experiences and security in the smart home space.

4 Edge Computing and Cloud Computing: Literature Review

We are conducting a literature review to answer RQ2. In the literature review, we first formulate a search query, then we define the journals we are searching from, and then select papers based on the search results.

4.1 Methodologies

4.1.1 Literature Sources

The literature sources used for this review were:

- Google Scholar
- Springer
- IEEE Xplore
- UTU Volter, the search engine used by University of Turku

4.1.2 Search Query

For our search query, I decided to first include the terms "Edge Computing" AND "Cloud Computing".

("Edge Computing" AND "Cloud Computing")

I will also add "Artificial Intelligence" and "AI" to the search query to include results that discuss the topic in relation to Artificial Intelligence. We are searching for comparisons between Cloud Computing and Edge Computing for AI, so I will be including ("trade-off" OR "trade-offs" OR "comparison") in the search query.

The final search query is: **("Cloud Computing" AND "Edge Computing") AND ("Artificial Intelligence" OR "AI") AND ("Trade-off" OR "Trade-offs" OR "Comparison")**

4.1.3 Inclusion and Exclusion Criteria

We will limit the search results to 2020 or newer, and select the most relevant ones from the first 50 articles.

The exclusion criteria for the articles were:

- No mention of cloud OR edge computing in the title or abstract
- Every important keyword is mentioned, but the article does not compare the two topics
- Article is behind a paywall or isn't accessible with University of Turku credentials

Some articles might be included even if they do not mention the phrases "Comparison" or "Trade-offs", if the topic seems relevant. For example, an article called "Artificial Intelligence for Cloud and Edge Computing" and others like it were included in the results, because they discuss the topic in general.

4.1.4 Review Steps

To answer RQ2, we need to search for specific keywords in the articles. For each of the selected articles, an analysis will be conducted by following these steps:

1. Reading the introduction
2. Searching for the following keywords: Artificial intelligence, Cloud Computing, Edge Computing
3. Reading the chapter which discusses the results of the research
4. Reading the whole article

During each of the analysis steps, the article may be removed from the selected publications if it does not contain valuable data for the literature review or if the data seems unreliable.

4.2 Conducting the Search

Google Scholar

In Google Scholar, we found 18400 articles with our initial search. From reading the title and abstract from the first 50 articles, I found 14 relevant articles to be analyzed in more depth. Because Google Scholar's search is too vast, this will be a limitation on the repeatability of the literature study later on.

UTU Volter

In UTU Volter, the initial search resulted in 150 articles. We reviewed the first 50 articles based on the title and the abstract and found 1 unique article to be analyzed in more depth. Articles that were found in previous searches were not included in the results.

Publication	Results from search	Relevant Articles
Google Scholar	18400 (50)	14
UTU Volter	150 (50)	1
Springer	3572 (50)	1
IEEE Xplore	135 (50)	7
Total	22257 (200)	23

Table 4.1: Overview of articles found in the literature search

Springer

In Springer, the initial search resulted in 3,572 articles. Out of the first 50 results, 1 article was found to be relevant.

IEEE

In IEEE, the initial search resulted in 135 articles. Out of the first 50 results, 7 articles were found to be relevant.

Results of the Search

Every article selected from the search is compiled in Table 4.2. They are in the same order that they were found during the search.

Index	Title
[$P1_{EC}$]	Edge Computing vs. Cloud Computing: A Comparative Analysis for Real-Time AI Applications
[$P2_{EC}$]	Optimizing edge computing and AI for low-latency cloud workloads
[$P3_{EC}$]	AI on the edge: a comprehensive review
[$P4_{EC}$]	The development of iot-smart basket: Performance comparison between edge computing and cloud computing system
[$P5_{EC}$]	Edge computing and cloud computing for internet of things: A review
[$P6_{EC}$]	From sensors to data intelligence: Leveraging IoT, cloud, and edge computing with AI
[$P7_{EC}$]	A survey of recent advances in edge-computing-powered artificial intelligence of things
[$P8_{EC}$]	AI and computing horizons: cloud and edge in the modern era
[$P9_{EC}$]	Scenario-based AI benchmark evaluation of distributed cloud/edge computing systems
[$P10_{EC}$]	A survey on the convergence of edge computing and AI for UAVs: Opportunities and challenges
[$P11_{EC}$]	Distributed artificial intelligence empowered by end-edge-cloud computing: A survey
[$P12_{EC}$]	A review on computational intelligence techniques in cloud and edge computing
[$P13_{EC}$]	Edge and Cloud Computing in Smart Cities
[$P14_{EC}$]	Edge Computing: The Future Era of Cloud Computing-A Review
[$P15_{EC}$]	A Review on Edge-Computing: Challenges in Security and Privacy
[$P16_{EC}$]	Keep Clear of the Edges : An Empirical Study of Artificial Intelligence Workload Performance and Resource Footprint on Edge Devices
[$P17_{EC}$]	Comparison of Cloud Computing and TinyML Methods for Brain-Computer Interface in Motor Imagery Problems
[$P18_{EC}$]	Application of Machine Learning Algorithm in Cloud-to-edge Computing: Analysis and Limitations
[$P19_{EC}$]	Resource Allocation in Combined Fog-Cloud Scenarios by Using Artificial Intelligence
[$P20_{EC}$]	A Comprehensive Analysis of Cloud-Edge-IoT Collaboration for Scalable and Instantaneous Uses
[$P21_{EC}$]	AI-Powered Edge Computing: Enhancing Cloud-Native AI Applications
[$P22_{EC}$]	Advancements and Challenges in Cloud Computing, Edge Computing and Edge-Cloud Computing for IoT: A Comprehensive Review
[$P23_{EC}$]	Edge artificial intelligence for big data: a systematic review

Table 4.2: Articles selected from the literature search

4.2.1 Selecting the Articles

We will select the research data from these articles based on three metrics:

- The trade-off that is measured (e.g. Performance vs Latency)
- The type of artificial intelligence being used (e.g. Image Recognition, Machine Learning, etc.)
- The use case for artificial intelligence in the article (e.g. Autonomous Vehicles)

If the article does not contain all of these research metrics, it will be discarded from the final literature review.

Below is a table where we select the final articles based on the three categories of data in them.

Table 4.3: Analysis of Articles

Index	Trade-off	AI Type	Use Case	Selected
[P1$_{EC}$]	Latency, security features, data processing capabilities, scalability	Machine Learning (ML), Image Recognition (IR)	Autonomous Vehicles (AV), Smart Home IoT	Yes
[P2$_{EC}$]	Latency, Performance, Computational capabilities, Efficiency, Power efficiency	ML, Federated Learning (FL), Reinforcement Learning (RL)	AV, Healthcare	Yes
[P3$_{EC}$]	Latency, Bandwidth, Device Constraints, Communication Cost vs. Accuracy	Deep Learning (DL), ML Subfields, Computer Vision (CV), FL, Edge Inference (EI)	CV Applications, IoT	Yes
[P4$_{EC}$]	Processing Time	IR	Smart Basket	No
[P5$_{EC}$]	x	ML, FL, CV, DL	Healthcare, CV applications	No
[P6$_{EC}$]	Latency, Bandwidth, Efficiency, Performance	ML, Support Vector Machines (SVM), Convolutional Neural Networks (CNN), DL	AV, Industrial Automation, Healthcare, Smart Cities, Energy Management	Yes
[P7$_{EC}$]	No trade-off discussed	ML, DL, CNN, Recurrent Neural Networks	Autonomous Driving (AD), Smart Healthcare, Smart Industry, Smart Agriculture	No
[P8$_{EC}$]	No trade-off discussed	ML, Reinforcement Learning (RL), FL	AV, Smart Cities, IoT, Health monitoring	No
[P9$_{EC}$]	Performance, Latency, Accuracy, Speed	ML, DL, Long Short-Term Memory (LSTM), Image Classification	AV, Healthcare, Community Surveillance, Smart City, Smart Home	Yes
[P10$_{EC}$]	No trade-off discussed	CV, DL, RL, Deep Reinforcement Learning, FL, Neural Networks (NN)	Autonomous Navigation, Collision avoidance, CV Applications	No
[P11$_{EC}$]	No trade-off discussed	DL, CV, CNN, FL	AV, Real-Time Video Analytics, Remote Healthcare, Smart Industry / Manufacturing	No
[P12$_{EC}$]	No trade-off discussed	No specific AI type mentioned	No use case presented	No
[P13$_{EC}$]	No trade-off discussed	FL, RL	No specific use case	No

Continued on next page

Index	Trade-off	AI Type	Use Case	Selected
[P14$_{EC}$]	No trade-off discussed	No specific AI type mentioned	No specific use case	No
[P15$_{EC}$]	No trade-off discussed	No specific AI type mentioned	No specific use case	No
[P16$_{EC}$]	Performance, Latency, Memory, Overhead	Classification, Image-to-Image, Segmentation CNN	AV, Object Recognition, Smart City Systems	Yes
[P17$_{EC}$]	Latency	ML, CNN, TinyML, SVM, LSTM	Brain–Computer Interface	Yes
[P18$_{EC}$]	No trade-off discussed	ML, SVM,	Resource Management, Cloud-to-Edge Computing IoT, Smart Environments, Security	No
[P19$_{EC}$]	No trade-off discussed	Artificial Neural Network	Task distribution in fog-cloud systems	No
[P20$_{EC}$]	No trade-off measured	ML, FL	No use case implemented	No
[P21$_{EC}$]	Accuracy, Latency, Resource Usage	Lightweight NN, Object Detection, Decision Trees, FL	AV, Smart Cities, Healthcare monitoring, Environmental monitoring	Yes
[P22$_{EC}$]	No trade-off measured	No specific AI type mentioned	No specific use case	No
[P23$_{EC}$]	No trade-off measured	ML, DL, NN, symbolic AI	Smart Cities, Smart Homes, Internet of Vehicles, Healthcare, Manufactoring	No

4.3 Analysis

After reviewing each article against the above-mentioned criteria, we have found 8 articles that meet our criteria. Those articles are: [P1$_{EC}$], [P2$_{EC}$], [P3$_{EC}$], [P6$_{EC}$], [P9$_{EC}$], [P16$_{EC}$], [P17$_{EC}$], [P21$_{EC}$].

We will analyze the articles in numerical order and list the trade-offs found in them.

At the end of the chapter, we will compile the main findings of every article into an easy-to-read table.

[P1$_{EC}$] - Edge Computing vs. Cloud Computing: A Comparative Analysis for Real-Time AI Applications Publication [P1$_{EC}$] conducts a comparative analysis between Edge Computing and Cloud computing for real-time AI applications. It examines three challenges for Edge Computing: network bottleneck, processing capacity, and scalability.

The main findings it lists are the following:

Metric	Cloud Computing	Edge Computing
Latency (ms)	100-200	10-20
Data Security	6/10	7/10
Processing Speed (tasks/sec)	1000-10 000	100-1000
Scalability	9/10	6/10

Table 4.4: Differences between Edge and Cloud Computing listed in [P1$_{EC}$]

In Table 4.4 the author lists differences in metrics between Edge and Cloud Computing. While some of the categories do not contain real-world metrics like Data Security and Scalability, others have quantitative data like Latency and Processing Speed.

[P1$_{EC}$] does not specify what type of AI is used for collecting the data. Nevertheless, it clearly displays a trade-off between improved latency (edge) and improved processing speed and scalability (cloud).

[P2$_{EC}$] - Optimizing edge computing and AI for low-latency cloud workloads [P2$_{EC}$] discusses how edge systems and AI are optimized to perform on the edge. It discusses methods like hybrid cloud-edge inference, hardware acceleration, as well as creating new AI models with model quantization, pruning, and knowledge distillation.

[P2$_{EC}$] gives us a baseline for how AI algorithms perform without any of these techniques. Edge devices have larger inference latency compared to using a cloud server when used in similar tasks. This is due to fewer resources, such as processing

power and RAM, which limit the performance of AI models. Below is a table from the article:

Parameter	Edge Device (e.g. Raspberry pi 4)	Edge AI Accelerator (e.g NVIDIA Jetson)	Cloud Server (e.g AWS GPU Instance)
Processor Type	Quad-core ARM Cortex-A72	Embedded GPU (256 CUDA Cores)	High-end GPU (T4, V100, A100)
RAM	4GB	8GB	128GB+
Power Consumption	5W	10W-30W	200W+
AI Model Capacity	Small models (TF-Lite)	Medium-Sized CNNs	Large-Scale Transformer Models
Inference Latency	-100ms	-10-50ms	-5ms

Table 4.5: Differences between Edge and Cloud systems

With the use of the above-mentioned methods, Edge AI has shown improvements when compared to traditional Cloud AI applications. The article discusses improvements that can be achieved through using Edge AI but provides no trade-off other than cloud computing having better inference latency if no model optimization occurs.

This comparison shows that even though cloud systems offer better performance, optimization techniques allow Edge AI to achieve good enough latency in many real-time applications. The trade-off isn't absolute, but depends on how efficiently the AI models have been adapted to the edge.

[$P3_{EC}$] - AI on the edge: a comprehensive review [$P3_{EC}$] is a systematic review which discusses model optimization techniques for using different AI models in Edge Computing.

The article does not contain measured trade-offs but it discusses several trade-offs found in different model optimization techniques. These trade-offs are between model accuracy, compression ratio, latency and device energy consumption.

While [$P3_{EC}$] does not discuss any measured empirical trade-off, it discusses the trade-offs in a general way. The trade-offs discussed are:

- Cloud computing requires the data to be uploaded to data centers, which can impose a burden on the network and cause latency issues
- Because the data has to be uploaded to the cloud, it cannot provide low-latency services.
- The data is also uploaded to the cloud, which can be a potential threat to user privacy.

Compared to cloud computing, edge computing has the following advantages:

- Computation is performed locally or at edge servers, reducing latency
- Data isn't uploaded to the cloud, reducing bandwidth
- Data is stored at the edge, which protects user privacy

However, it is difficult to run computationally intensive tasks on resource-constrained devices.

[$P3_{EC}$] also discusses cloud-edge collaboration. This collaboration uses both techniques to their advantages, performing low-latency tasks on the edge while high-compute tasks get sent to the cloud.

The findings in [$P3_{EC}$] are summarized in the table below:

Trade-off	Edge Computing	Cloud Computing
Bandwidth	Creates substantial network bandwidth demand	Significantly reduces network bandwidth usage
Latency	Unable to deliver low-latency responses	Provides low-latency inference close to the data source
Privacy	Requires data transmission to the cloud, raising privacy concerns	Keeps data local, enhancing privacy protection
Computing Power	Offers high computational capacity for intensive AI workloads	Operates under limited computational and energy resources

Table 4.6: Cloud and Edge trade-offs found in [$P3_{EC}$]

[$P6_{EC}$] - From sensors to data intelligence: Leveraging IoT, cloud, and edge computing with AI [$P6_{EC}$] discusses similar trade-offs as previous publications, but also introduces new trade-offs such as Model Training & Updates and Cloud-Edge synergy. For clarity, all trade-offs discussed have been compiled into Table 4.7.

Trade-off	Edge Computing	Cloud Computing
Latency & Real-Time Response	Processing data near the source reduces latency and allows for real-time response	Transferring large amounts of data causes latency issues
Bandwidth Usage	Edge systems analyze data locally, eliminating the need to send vast amounts of data to cloud servers	Transferring large amounts of data causes vast bandwidth usage, increasing costs
Computing Power	Edge devices face significant limitations in computational power, memory and energy availability	Cloud computing has nearly limitless power capabilities, allowing it to perform intense computational tasks
Model Training & Updates	Maintaining data consistency across distributed edge devices presents additional complexity	The cloud enables the collection of data from edge devices and retraining of models. Retrained models can be synchronized with edge devices
Security & Privacy	Edge computing allows the data to stay local increasing privacy and security	Transferring large amounts of data poses a challenge to privacy and security
Network Dependence	Edge computing allows for reliable operation even with unreliable networks	Network reliability poses a challenge for cloud computing
Energy & Sustainability	Edge computing processes the data closer to the source mitigating the energy impact of data transport networks	High bandwidth costs and the environmental impact of communication networks
Cloud-Edge Synergy	Critical, time-sensitive tasks are executed at the edge	Model training and high-compute tasks are performed in the cloud

Table 4.7: Trade-offs found in [$P6_{EC}$].

[$P9_{EC}$] - Scenario-Based AI Benchmark Evaluation of Distributed Cloud/Edge Computing Systems [$P9_{EC}$] discusses the trade-offs for running Machine Learning & Deep Learning models in Edge and Cloud scenarios. [$P9_{EC}$] is much more focused on machine learning models than other publications. Additionally it focuses on edge servers, a practice where the edge devices have a network

server which does the necessary computing. The trade-offs found in the publication have been compiled into Table 4.8.

Trade-off	Edge Computing	Cloud Computing
Training vs. Inference	The pre-trained ML models run on the edge servers. The AI inference is mostly carried out at edge servers	ML/DL models are trained at the cloud datacenter due to large datasets involved
Bandwidth & Latency	Distributed platform alleviates the problems of limited bandwidth, offloading latency and data storage often found in centralized platforms	Skipping the edge servers results in longer network latency in data transmission or control commands
Response Time	AI models running on edge servers had lower response times, more complex models performed poorly on edge device	AI models running on cloud servers had higher response times than those running on edge servers
Recommended Usage	Edge servers should be used for logic inference purposes	Cloud Servers should be used for large-scale training and heavy computation requiring big datasets and scalable GPUs.

Table 4.8: Cloud and Edge trade-offs found in [$P9_{EC}$].

The trade-offs identified in [$P9_{EC}$] are between training vs inference, bandwidth vs latency, response time, and recommended usage.

[$P16_{EC}$] Keep Clear of the Edges - An Empirical Study of Artificial Intelligence Workload Performance and Resource Footprint on Edge Devices

[$P16_{EC}$] performs an empirical study, measuring the performance of three different systems under multiple AI workloads including object recognition, classification, facial recognition, super resolution, image deblurring, image enhancement, and others. The devices that the AI workloads were running on were a Raspberry Pi Model 4B (General edge architecture), NVIDIA Jetson TX2 (Heterogeneous edge architecture), and Amazon EC2 t2.xlarge VM (General cloud architecture).

A key finding of the study is that general and heterogeneous edge architectures present completely different performance depending on the presence of hardware accelerators. If hardware accelerators are available, edge systems can improve performance by offloading computation from the CPU. This, however, increases the power and memory consumption of the device.

For many AI workloads, general-purpose edge devices and cloud virtual machines demonstrate comparable resource footprints. For AI applications on the edge, local hardware determines its performance. The study also shows that cloud servers deliver faster and more stable performance due to their higher-end compute infrastructure.

The final takeaway is that AI processing in edge devices is also concerned with other factors such as response time, model volume and cost. Only a few AI workloads and related models are suitable to execute or deploy on the edge.

[$P17_{EC}$] Comparison of Cloud Computing and TinyML Methods for Brain-Computer Interface in Motor Imagery Problems [$P17_{EC}$] is an empirical study which compares how cloud computing and TinyML edge computing perform when classifying EEG-motor imagery signals for Brain-Computer Interface (BCI) applications. It focuses on whether real-time BCI-systems benefit more from sending data to the cloud or running the models locally on the edge. The study uses latency as the metric of comparison.

In the study, the authors simulated two scenarios using the same EEG dataset, one being sent to the cloud where the machine learning model is executed, and the other where it is executed locally.

In this scenario, TinyML methods performed noticeably better. The latency for TinyML methods was on average 0.000904 s, while the average latency for cloud computing methods was 1.545910 s. The much better latency is attributed to not having the need for data transmission.

[P21$_{EC}$] AI-Powered Edge Computing: Enhancing Cloud-Native AI Applications The aim of the study is to analyze the performance of several AI models while constrained by edge computing and compare their performance to solutions deployed on the cloud. The study tested lightweight neural networks (e.g., MobileNet and Tiny-YOLO), decision trees, and federated learning algorithms using the IoT-2 dataset as the real-world data source to conduct environmental monitoring. The performance is measured through implementation scenarios where aspects of accuracy, latency and resource consumption highlight how useful each model might be considered in edge based scenarios.

Most articles that contained trade-offs did not directly discuss them.

4.4 Conclusion

We have identified multiple trade-offs between Edge Computing and Cloud Computing for Artificial Intelligence. The main trade-off discussed in nearly every study was between latency and computational power [P1$_{EC}$], [P2$_{EC}$], [P6$_{EC}$], [P16$_{EC}$], [P17$_{EC}$], [P21$_{EC}$]. Edge systems have significantly reduced latency compared to cloud systems, with [P17$_{EC}$] reporting results that were 1.545006 s faster. This reduced latency is due to edge systems not needing to send data to the cloud, instead performing processing closer to the data source [P3$_{EC}$], [P6$_{EC}$]. [P1$_{EC}$] also identified that cloud systems exhibit higher latency but provide greater computational power. Reduced latency is critical for time-sensitive tasks such as autonomous vehicles and healthcare applications [P6$_{EC}$], [P21$_{EC}$].

Other key trade-offs include reduced bandwidth usage and network load, as edge computing minimizes the need to transmit data to centralized cloud servers [P3$_{EC}$], [P6$_{EC}$], [P9$_{EC}$]. These studies show that processing data locally reduces the amount of information that must be transferred through communication networks, improving system efficiency in distributed AI environments.

Additionally, a trade-off between model training and inference was present in multiple publications [$P6_{EC}$], [$P9_{EC}$], [$P21_{EC}$]. Model training requires large amounts of processing power, whereas inference (making predictions based on the trained model) is a relatively low-compute task [$P6_{EC}$]. Training machine learning models benefits greatly from the increased performance that cloud computing provides [$P9_{EC}$], [$P21_{EC}$].

AI inference does not require as much processing power, so it is often more efficient to perform it on the edge [$P6_{EC}$], [$P9_{EC}$]. This is because inference tasks benefit from lower latency and reduced need for data transmission, enabling faster real-time responses [$P3_{EC}$], [$P6_{EC}$].

Security and privacy are also important considerations. Edge computing presents an additional layer of privacy and security compared to cloud computing, as data can be processed locally without the need to transmit sensitive information to external servers [$P3_{EC}$], [$P6_{EC}$]. In contrast, cloud computing requires data to be sent over networks, which may introduce potential security and privacy risks [$P3_{EC}$].

The optimal approach for artificial intelligence is to use both technologies for their strengths. Cloud computing can be used to perform high-compute tasks such as model training and large-scale data processing, while edge computing can be used for low-compute, low-latency tasks such as model inference and real-time decision-making [$P6_{EC}$], [$P9_{EC}$], [$P21_{EC}$].

Identified Trends and Gaps in the Literature

While conducting the literature review, I noticed a distinct lack of empirical studies comparing Edge and Cloud Computing. There are multiple studies discussing the key differences between the technologies, but many lack measured data to support their claims.

Answer to RQ2

In this section, we answered RQ2 by identifying the key trade-offs between using cloud computing vs edge computing for artificial intelligence. The main trade-offs identified are the trade-offs between:

- Computational power vs Latency [$P1_{EC}$], [$P2_{EC}$], [$P6_{EC}$], [$P16_{EC}$], [$P17_{EC}$], [$P21_{EC}$]
- Bandwidth usage [$P3_{EC}$], [$P6_{EC}$], [$P9_{EC}$]
- Training vs Inference [$P6_{EC}$], [$P9_{EC}$], [$P21_{EC}$]
- Privacy vs Security [$P3_{EC}$], [$P6_{EC}$]

5 Facial Recognition Access Control System on Raspberry Pi 5

For the experimental part of this thesis, I decided to combine the themes of the previous chapters into an all-encompassing project. Specifically, I created a facial recognition application on a Raspberry Pi 5 mini-computer using Generative AI. The application is containerized with Docker so that it can be deployed on any compatible device without manual dependency installation. The Raspberry Pi 5 is a system-on-chip computer which has all the required hardware to run a Linux-based operating system, allowing it to run programs and scripts.

The technical specifications of Raspberry Pi 5 are listed in Table 5.1:

Category	Specification
CPU	Broadcom BCM2712, Quad-core Arm Cortex-A76 @ 2.4GHz
GPU	VideoCore VII GPU, supporting OpenGL ES 3.1, Vulkan 1.2
Memory	2 GB / 4 GB / 8 GB / 16 GB
GPIO	40-pin GPIO header
Video Outputs	2 x micro-HDMI
USB Ports	2 x USB 2.0, 2 x USB 3.0
Camera/Display	2 x mini 22-pin FFC connectors (CSI/DSI combined), 0.5 mm pitch, 11.5 mm width
PCIe	Single-lane PCIe FFC connector
Connectivity	Gigabit (1 Gb/s) Ethernet RJ45 with PoE+ support
Wireless	2.4/5 GHz dual-band 802.11ac Wi-Fi 5 (300 Mb/s)
Bluetooth	Bluetooth 5, Bluetooth Low Energy (BLE)
Other Connectors	UART connector, RTC battery connector, 4-pin JST-SH PWM fan connector
Storage	microSD card slot
Power	USB-C power; 5 V at 5 A (25W); or 5 V at 3 A (15 W) with a 600 mA peripheral limit

Table 5.1: Raspberry Pi 5 technical specifications [10]

5.1 Facial Recognition Access Control System

I decided to create a Facial Recognition Access Control System on my Raspberry Pi 5 for my experimental part. I will also run the same system on my PC to compare the performance metrics between different types of edge devices. The application is packaged as a Docker container, making it straightforward to run the same software on both devices without modifying the environment.

The solution uses the `face_recognition` library [11] developed by Adam Geitgey. It is an API on top of two dlib [12] models:

1. Face detection - dlib's Histogram of Oriented Gradients (HOG)-based frontal face detector

2. Face recognition - a Residual Network-based deep neural network trained on $\approx$ 3 million face images that produces a 128-dimensional embedding vector per face. Two faces are considered the same person if the distance between their embeddings is below the tolerance threshold

The system uses expression-based liveness detection to prevent access using a photograph. The Raspberry Pi 5 communicates with two separate devices over MQTT: a Shelly 1 smart relay module that controls the electric door lock, and an ESP8266 microcontroller that controls the red, yellow, and green status LEDs. The system operates as follows:

1. The system detects a face in the camera frame:

   (a) If the face matches an enrolled user, the Raspberry Pi 5 publishes an MQTT message to the ESP8266, which turns on the yellow LED to prompt the user to smile.

   (b) If the face is not recognized, the red LED remains on and access is denied.

2. The user smiles. Once a smile is held for approximately one second, liveness is confirmed.

3. The Raspberry Pi 5 publishes an MQTT message to the ESP8266 to turn on the green LED, and a separate MQTT message to the Shelly 1 to activate the relay and open the door lock.

4. After 10 seconds, the Raspberry Pi 5 publishes MQTT messages to close the Shelly 1 relay and return the ESP8266 LEDs to the locked state with the red LED enabled.

5.2 Performance Metrics

I will collect performance metrics from the program and compare the performance of two different edge devices, Raspberry Pi 5 and my own Desktop PC, with the following specifications:

Component	Desktop PC	Raspberry Pi 5
CPU	AMD Ryzen 7 3700X, 8-core @ 3.6 GHz	Broadcom BCM2712, 4-core Arm Cortex-A76 @ 2.4 GHz
RAM	32 GB DDR4	8 GB LPDDR4X
GPU	NVIDIA GeForce RTX 3060 Ti	VideoCore VII
Storage	Kingston NVMe SSD 2 TB	64 GB SD Card
Architecture	x86_64	ARM64

Table 5.2: Specifications of the two edge devices used in performance testing

My personal PC is not an edge device in a traditional sense. It represents high-end general-purpose hardware, which adds value to the comparison. Because the application runs inside a Docker container, the same image can be pulled and executed on both devices with no changes to the software or its dependencies.

The performance metrics are listed in Table 5.3:

Metric	What it measures
detection_ms	`face_locations()` — time to find faces in the frame
recognition_ms	`face_encodings()` — ResNet inference time
landmark_ms	`face_landmarks()` — smile/liveness detection time
total_ms	full pipeline time per processed frame
fps	rolling 30-frame average
cpu_pct	process CPU usage %

Table 5.3: Performance metrics collected during testing

5.3 Creating the Initial Prompt

I decided to use the Claude Sonnet 4.6 model on high effort with the Command Line Interface (CLI) for creating the application. To create the best possible initial prompt, I used the Prompting best practices [13] guide released by Anthropic, the model's creator. The guide emphasized multiple general principles to include in the prompt. Rather than writing the final prompt immediately, I built it up iteratively by applying each guideline one at a time. I started with a minimal, unrefined prompt:

```
Create a facial recognition app for my Raspberry Pi 5 that
can open my door that is controlled through MQTT.
```

Be clear and direct. The first principle in Anthropic's guide is to be clear and direct about the desired output of the prompt. The prompt needs to be specific about the desired output format and constraints, and the instructions should be provided as sequential steps using numbered lists or bullet points when the order or completeness of the steps matters. I added the key technical constraints to the prompt: the door is controlled via MQTT through a Shelly 1 smart relay, the LEDs are controlled via MQTT using an ESP8266 microcontroller, the system should use the Python face_recognition library rather than a custom model, and the application should be containerized with Docker for portability.

```
Create a facial recognition access control application
in Python for my Raspberry Pi 5. The application should
use the face_recognition library for face detection
and recognition. When an enrolled face is recognized
and liveness is confirmed via smile detection, publish
an MQTT message to a Shelly 1 smart relay to open the
door lock, and publish a separate MQTT message to an
ESP8266 microcontroller to control the red, yellow, and
green status LEDs. Package the application inside a
lightweight Docker container so that it can be deployed on
the Raspberry Pi 5 and a Desktop PC without modifications.
```

Add context to improve performance. The guide recommends adding context or motivation to the prompt to improve performance. The context can explain to Claude why such behavior is important and helps Claude better understand the goals to deliver more targeted responses. I added context about the hardware environment, the purpose of the system, the performance metric requirements, and the Docker deployment target.

```
I am building a smart home access control system for my
master's thesis. The Raspberry Pi 5 runs Raspberry Pi OS
(64-bit, ARM64) and is connected to a USB camera. The same
Docker container must also run on an x86_64 Ubuntu desktop
for performance comparison. The system must prevent
spoofing attacks by requiring the user to smile before
granting access (liveness detection). Collect per-frame
timing metrics (detection, recognition, landmark, total in
```

```
milliseconds), FPS, CPU usage, and RAM usage so that I can
compare performance between the two devices.
```

Use examples effectively. The guide recommends using examples as one of the most reliable ways to steer Claude's output format and structure. The examples need to be:

- Relevant, mirroring the actual use case closely
- Diverse, covering edge cases and varying so that Claude does not pick up unintended patterns
- Structured, wrapped in `<example>` tags so that Claude can distinguish them from instructions

Anthropic recommends including 3-5 examples for the best results.

```
<examples>
  <example>
    Scenario: An enrolled face is detected.
    Expected behavior: Publish MQTT message to ESP8266 to turn on
    the yellow LED and begin smile detection. If liveness is
    confirmed, publish MQTT message to ESP8266 to turn on the green
    LED and publish MQTT message to Shelly 1 to open the door lock.
  </example>
  <example>
    Scenario: An unrecognized face is detected.
    Expected behavior: Keep the red LED on and do not publish any
    MQTT message to the Shelly 1. Access is denied.
  </example>
  <example>
```

```
    Scenario: An enrolled face is detected but liveness check fails.
    Expected behavior: Publish MQTT message to ESP8266 to turn the
    red LED back on. Do not open the door lock.
  </example>
</examples>
```

Structure the prompt with XML tags. Finally, the guide recommends structuring the prompt with XML tags to help Claude parse complex prompts unambiguously, especially when the prompt mixes instructions, context, examples, and variable inputs. Each type of content should be wrapped in its own tag (e.g. `<instructions>`, `<context>`, `<examples>`) to reduce misinterpretation. We have added context, task, and examples to the prompt.

```
 <context>I am building a smart home access control
system for my master's thesis. The Raspberry Pi 5 runs
Raspberry Pi OS (64-bit, ARM64) and is connected to a USB
camera. The same Docker container must also run on an
x86_64 Ubuntu desktop for performance comparison. The
system must prevent spoofing attacks by requiring the
user to smile before granting access (liveness detection).
Collect per-frame timing metrics (detection, recognition,
landmark, total in milliseconds), FPS, CPU usage, and RAM
usage so that I can compare performance between the two
devices.</context>

<task>Create a facial recognition access control
application in Python for my Raspberry Pi 5. The
application should use the face_recognition library
for face detection and recognition. When an enrolled
```

```
face is recognized and liveness is confirmed via smile
detection, publish an MQTT message to a Shelly 1 smart
relay to open the door lock, and publish a separate MQTT
message to an ESP8266 microcontroller to control the red,
yellow, and green status LEDs.  Package the application
inside a lightweight Docker container so that it can be
deployed on the Raspberry Pi 5 and a Desktop PC without
modifications.</task>

<examples>
  <example>
    Scenario: An enrolled face is detected.
    Expected behavior: Publish MQTT message to ESP8266 to turn on
    the yellow LED and begin smile detection. If liveness is
    confirmed, publish MQTT message to ESP8266 to turn on the green
    LED and publish MQTT message to Shelly 1 to open the door lock.
  </example>
  <example>
    Scenario: An unrecognized face is detected.
    Expected behavior: Keep the red LED on and do not publish any
    MQTT message to the Shelly 1. Access is denied.
  </example>
  <example>
    Scenario: An enrolled face is detected but liveness check fails.
    Expected behavior: Publish MQTT message to ESP8266 to turn the
    red LED back on. Do not open the door lock.
  </example>
</examples>
```

With a comprehensive prompt created, the development process can begin. The created prompt follows guidelines set by Anthropic for using their models. While further technical requirements and limitations could have been added to the prompt, these have been left for the model to infer during development.

5.4 Iterative Development with Claude Code

I started Claude Code in an empty project directory and provided the prompt developed in the previous section. Claude was granted full filesystem and command execution permissions. The complete development session is included in Appendix A. Using ten prompts, Claude generated a working facial recognition access control system that implemented face recognition, liveness detection, and MQTT-based communication with the external devices. Developing the application did not require any manual source code modifications. My involvement was limited to editing configurations in the `.env` files and executing the system commands required to install Docker and run the generated application.

The initial prompt produced a project with a complete structure. Prompting Claude Code to build the program revealed a non-contiguous array bug in dlib. Claude diagnosed and fixed this bug automatically during the build step. After fixing the bug, the program was able to build and run. The application initially failed to connect to the MQTT broker because the broker address was configured as `127.0.0.1`. Inside a Docker container, this address refers to the container itself rather than the host system. Claude resolved the issue by configuring the container to use host networking.

By the fourth interaction, MQTT communication between the application and the broker had been successfully established and verified. Afterwards, I noticed that the liveness check wasn't detecting the smiles at all. Claude diagnosed two flaws in the original landmark-based smile detection algorithm and replaced it with

OpenCV's Haar cascade smile detector, which produced substantially more reliable results.

In the next interaction, I requested a graphical interface for testing purposes. Claude modified the Docker environment to use the `opencv-python` package instead of `opencv-python-headless`, added the required Qt dependencies, and configured X11 forwarding so that the camera feed and system status information could be displayed in a graphical window. The remaining interactions focused on improving the robustness of the liveness detection process. These changes included increasing smile detection accuracy at greater distances, adding anti-spoofing detection, and reducing the number of frames to confirm liveness. In the last interaction, Claude provided me with instructions on how to deploy this image on my Raspberry Pi, since I did the development on my Desktop PC.

After the program was tested to work, the modules created by Claude were manually reviewed. No changes were required for the application to work and no issues were found in the code. The code created was easily interpreted and had multiple comments to explain the structure of the modules.

Creating the program after the initial prompt took 2-3 hours of development time. This represents a substantial reduction in development effort compared to implementing the same system manually, particularly given the need to integrate computer vision, MQTT communication, Docker deployment, and performance monitoring into a single application. The advantage of using Generative AI lies in the speed of development. The project was developed using Claude Sonnet 4.6 rather than one of the more capable frontier models available at the time. Consequently, the observed development performance should not necessarily be interpreted as the upper limit of what current generative AI systems can achieve. The development session consumed approximately 150,000 tokens. Because the work was performed

through a subscription-based Anthropic plan rather than a pay-per-token API, the monetary cost of the session cannot be determined precisely.

An interesting observation was that Claude was not only able to generate source code but was also able to diagnose and fix errors and bugs autonomously during runtime. During development, it identified compatibility problems between NumPy arrays and dlib, corrected Docker networking configuration issues, redesigned the liveness detection algorithm when the original solution did not function, and adjusted deployment instructions for ARM64 hardware. This shows that modern coding agents can support not only code creation but the entire development process, including debugging, testing, and deployment.

The human role in software development will shift to being a curator and a designer, instead of a creator. Understanding the fundamental software development principles is important, as it enables developers to critically evaluate the code they produce.

5.5 Project Structure

The final application generated through Claude Code follows a modular architecture. The system is divided into different modules for face processing, MQTT communication, performance metric collection, configuration management, and application control. This separation follows the Single Responsibility principle and improves system maintainability, allowing the different parts to be edited independently.

The different functions present in the app are listed in Table 5.4.

Module	Responsibility
`main.py`	Application control
`face_processor.py`	Smile detection & liveness check
`mqtt_client.py`	MQTT communication with the ESP8266 LED board and the Shelly 1 smart relay
`metrics.py`	Collecting performance metrics including processing times, FPS, CPU usage, and memory usage.
`config.py`	Centralized configuration management for using environment variables and application settings
`enroll.py`	Enrollment of new faces to the application
`__init__.py`	Package initialization file enabling the application modules to be imported as a Python package

Table 5.4: Software modules generated for the facial recognition access control system

The architecture of the system is depicted in Figure 5.1. The figure contains the Python libraries used by the modules and displays the dependencies. The application is running inside a Docker container and is divided into separate modules.

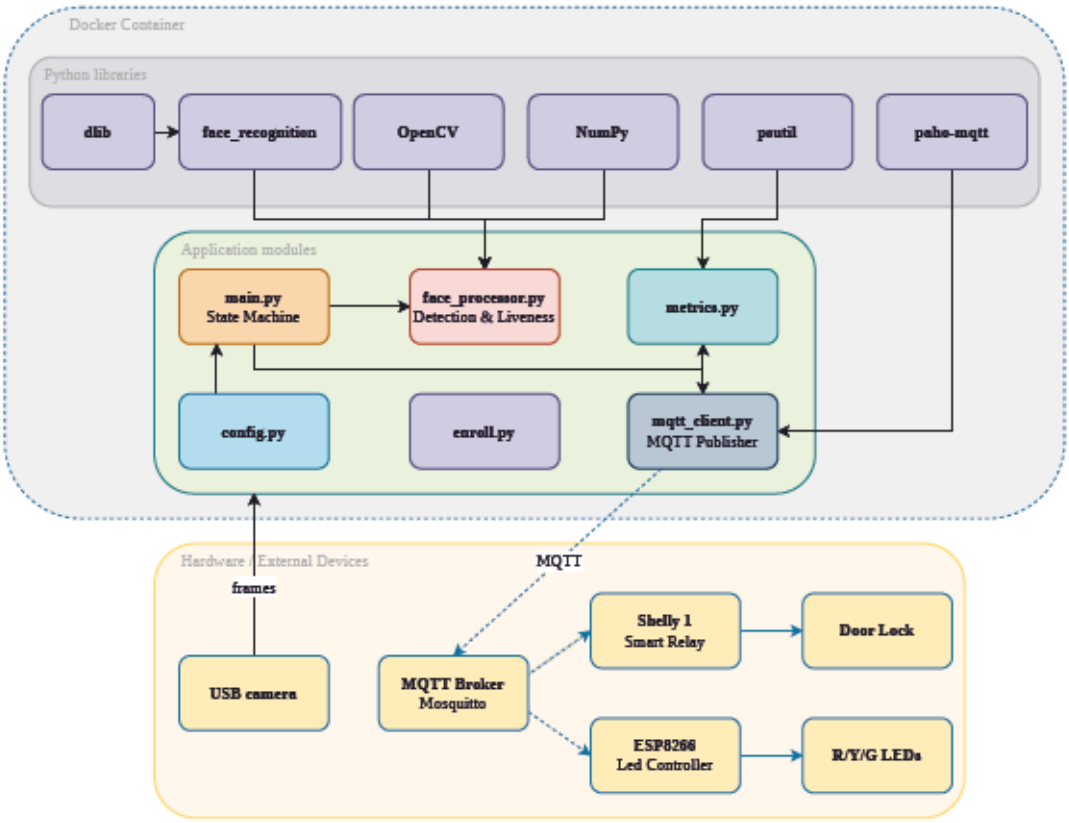

Figure 5.1: Architecture of the facial recognition access control system

5.6 Performance Comparison

A total of 20 access control cycles were performed under identical conditions on both devices. Each cycle consisted of the complete authentication process with face detection and identity verification. The performance metrics were collected in `metrics.py` and saved to a `.csv` file for comparison. Figure 5.2 presents the mean processing time for each stage of the pipeline. Figure 5.3 presents the distribution of processing times for each stage of the pipeline. Figures 5.4 and 5.5 present the FPS and CPU usage over 20 access cycles.

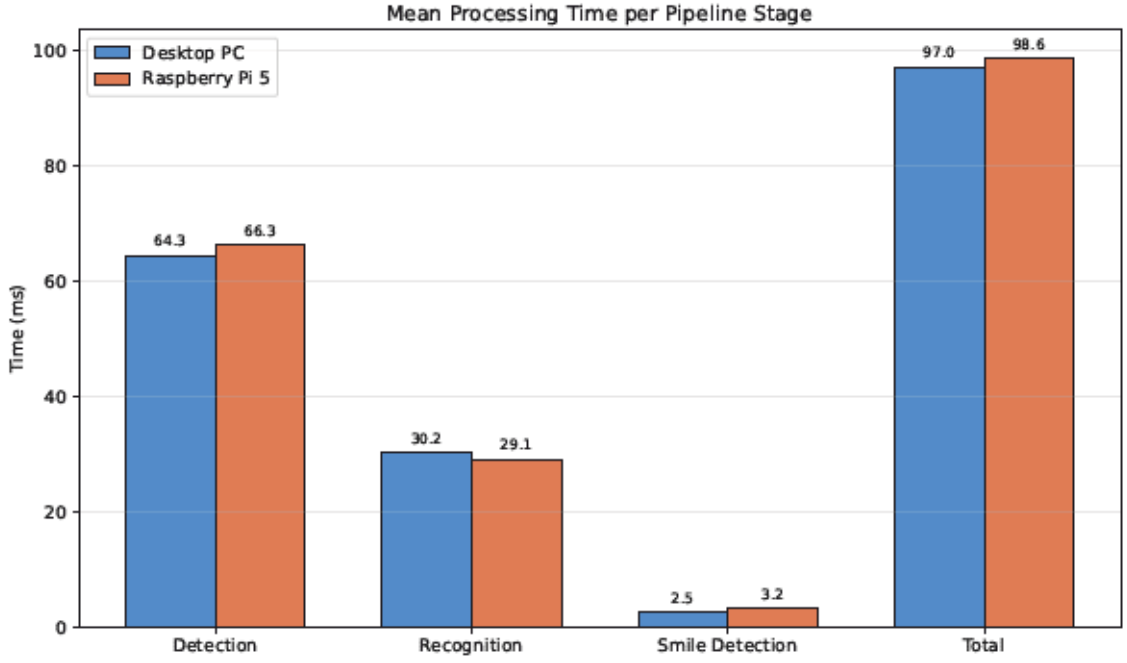

Figure 5.2: Mean processing time per pipeline stage

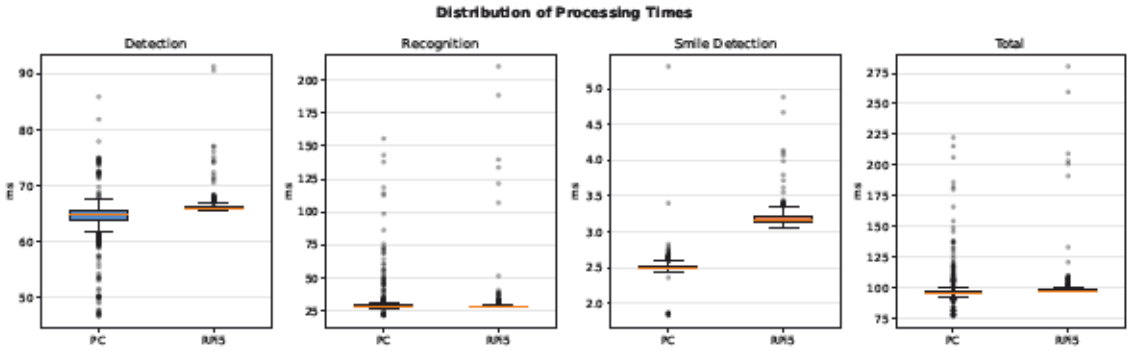

Figure 5.3: Distribution of processing times per pipeline stage

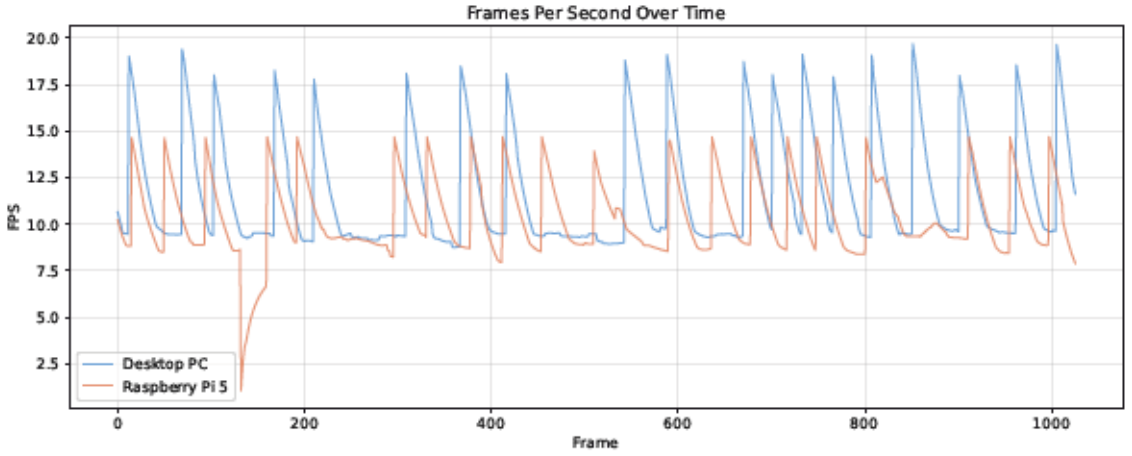

Figure 5.4: Frames per second over time during 20 access cycles

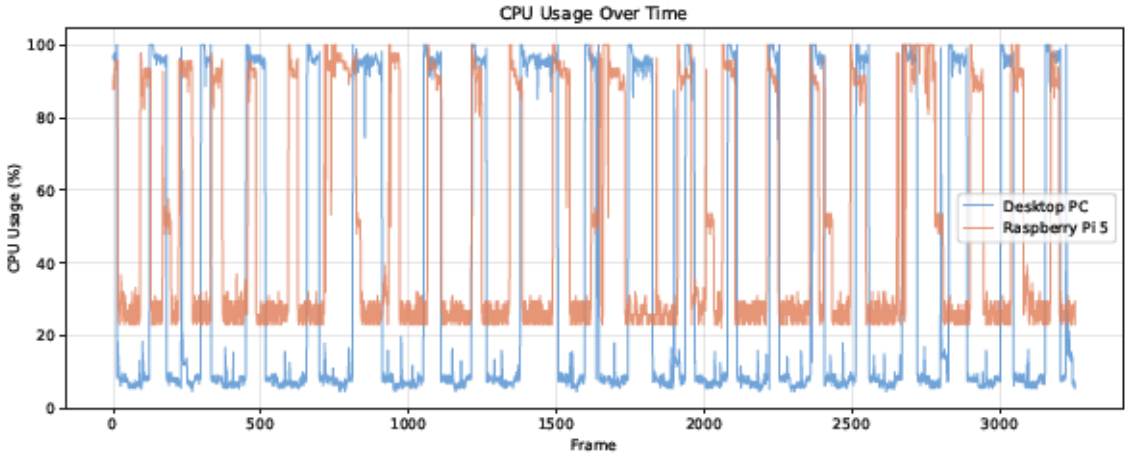

Figure 5.5: CPU usage over time during 20 access cycles

The results in Figure 5.2 indicate that the Raspberry Pi achieved comparable performance metrics to the Desktop PC. The mean total pipeline processing time differed by only 2% from the Desktop PC's total pipeline processing time (97.0 ms vs 98.6 ms). The recognition stage, based on the ResNet neural network inference, provided nearly identical results on both devices (30.2 ms vs 29.1 ms). The largest difference observed in the pipeline was in detection using the HOG algorithm at 3% (64.3 ms vs 66.3 ms).

Figure 5.4 shows that the Frames Per Second (FPS) is significantly better on the Desktop PC than on the Raspberry Pi 5 (19.1 vs 14.95 on average). Figure 5.5 shows that the Raspberry Pi 5 had higher mean CPU utilization compared to the PC (48.9% vs 36.9%). While the peak CPU usage was similar on both devices, the idle CPU usage was significantly higher on the Raspberry Pi 5 compared to the PC (23.2% vs 5.8%), indicating a higher baseline processor load.

The similar processing times for the authentication pipeline can be explained by the fact that both dlib's HOG and ResNet model run on the CPU, not the GPU. The desktop NVIDIA 3060 Ti GPU provides no advantage in this comparison. Additionally, the Raspberry Pi 5 Cortex-A76 is competitive with the Ryzen 7 processor in single-threaded performance.

Overall, the results of this experiment prove that the Raspberry Pi 5 is capable of running a real-time facial recognition access control system with comparable performance to a Desktop PC in CPU-heavy applications.

6 Conclusion

6.1 Results

In Chapter 3, we answered RQ1 by conducting a literature review. The results of the literature review show that the most prominent use case for Artificial Intelligence in a Smart Home Environment is Energy Management. This is unsurprising, as rising energy demand is a global challenge with no immediate solution in sight. With the residential housing sector accounting for 21% [9] of energy demand worldwide, having even modest reductions in consumption would make a meaningful difference. The second prominent use case was healthcare and activity monitoring. With the aging population in many Western societies, having more resources to monitor residents for potential health issues and emergencies can help alleviate the stress on the healthcare system. Other use cases include home automation and anomaly detection. These solutions allow for better user experiences and security in the smart home space.

In Chapter 4, we answered RQ2 by conducting a literature review. The main trade-offs when choosing between Cloud and Edge Computing for AI are Computational power vs Latency, Bandwidth usage, Training vs Inference, and Privacy vs Security. When deciding what kind of technology to use for an AI solution, it is important to keep these trade-offs in mind. In practice, the optimal approach is often to combine both technologies, using Cloud Computing for high-compute tasks such as model training and Edge Computing for low-latency tasks such as inference.

In Chapter 5, we answered RQ3. The approach on developing a face recognition access control application on a Raspberry Pi using Generative AI was successful. Claude Code CLI was proficient in using system tools to independently debug issues that came up during development. The low amount of human input required was surprising, my role in the development process was more of a curator and less of a creator. The Raspberry Pi 5 also achieved processing times comparable to a desktop PC for the CPU-bound recognition pipeline, demonstrating that edge hardware is viable for real-time facial recognition.

6.2 Limitations

When answering RQ1 and RQ2, I conducted the search by looking at the first 50 results from 4 different search tools. The results of the literature review could differ if I had a bigger sample size or used different search tools. The results could also differ depending on when the search is made, if new research into the topic has been released.

When answering RQ3, the output of the prompt was not compared with other prompts. It was created by only following Anthropic's guidelines and not adding other details. It could probably be further improved with other AI models and possible additions to the prompt such as technical details.

The application created in Chapter 5 only uses the CPU of the computer; therefore, the comparison does not evaluate the two devices based on their strengths. If the application had used the GPU of the PC, the results could have been vastly different. The application was made with the Claude Sonnet 4.6 model on high effort level. Using a different model could produce different results in development. Additionally, Generative AI models are making rapid progress, so the process could be significantly different in a few years.

6.3 Possible Further Research

There are some subjects that were not explored in my thesis due to time limitations or them falling out of scope. Possible avenues for further research into this thesis' topics include:

- More empirical studies into the trade-offs between Cloud and Edge computing.

There were a limited number of studies that had measured empirical data about the topic. Most of the publications that were found discussed the trade-offs in a general sense but lacked empirical data. Further research comparing the two technologies with clear, measured empirical data is needed.

- The differences between outputs with different prompts when developing with Generative AI.

In Chapter 5, I created a program using just a single prompt constructed from Anthropic's guidelines, but a comparison between the outputs of differing prompts would be valuable. Having the same end-goal but constructing prompts with different parameters would be a good point of further research. Most guidelines on prompting are released by the developers of the AI models, and scientific research into prompt engineering for Generative AI Command Line Interfaces is lacking.

- The full capabilities of AI Agents in the Command Line

In Chapter 5, I gave Claude Code CLI full permissions to do anything on my device. Many of the tools that were used came to me as a surprise, and the full capabilities of Command Line Interface AI Agents could be a topic of further research.

References

[1] P. Sethi and S. R. Sarangi, "Internet of things: Architectures, protocols, and applications", *Journal of Electrical and Computer Engineering*, vol. 2017, no. 1, p. 9 324 035, 2017, Article ID 9324035. DOI: 10.1155/2017/9324035.

[2] Statista. "Number of Internet of Things (IoT) connections worldwide from 2022 to 2023, with forecasts from 2024 to 2034 (in billions)". Data by Transforma Insights and Exploding Topics. Statista Inc., Statista, Accessed: Jun. 23, 2026. [Online]. Available: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/.

[3] F. K. Aldrich, "Smart homes: Past, present and future", in *Inside the Smart Home*, R. Harper, Ed. London: Springer London, 2003, pp. 17–39, ISBN: 978-1-85233-854-1. DOI: 10.1007/1-85233-854-7_2. [Online]. Available: https://doi.org/10.1007/1-85233-854-7_2.

[4] *Types of Artificial Intelligence | IBM*. Accessed: Jul. 21, 2025. [Online]. Available: https://www.ibm.com/think/topics/artificial-intelligence-types.

[5] *Types of AI: Explore Key Categories and Uses*, en-US, Section: Articles, Mar. 2025. Accessed: Jul. 25, 2025. [Online]. Available: https://ischool.syracuse.edu/types-of-ai/.

[6] IBM, *What is computer vision?*, https://www.ibm.com/think/topics/computer-vision, Accessed: 2026-06-25, 2024.

[7] I. H. Sarker, "Machine learning: Algorithms, real-world applications and research directions", *SN Computer Science*, vol. 2, no. 3, p. 160, 2021. DOI: 10.1007/s42979-021-00592-x.

[8] *Definition of Cloud Computing - Gartner Information Technology Glossary*, en. Accessed: Jul. 25, 2025. [Online]. Available: https://www.gartner.com/en/information-technology/topics/cloud-strategy.

[P1$_{AI}$] A. C. Ikegwu, O. J. Obianuju, I. S. Nwokoro, M. O. Kama, and D. U. Ebem, "Investigating the Impact of AI/ML for Monitoring and Optimizing Energy Usage in Smart Home", en, *Artificial Intelligence Evolution*, pp. 30–43, Jan. 2025, ISSN: 2717-5952. DOI: 10.37256/aie.6120256065. Accessed: Mar. 26, 2026. [Online]. Available: https://ojs.wiserpub.com/index.php/AIE/article/view/6065.

[P2$_{AI}$] P. Rodriguez-Garcia, Y. Li, D. Lopez-Lopez, and A. A. Juan, "Strategic decision making in smart home ecosystems: A review on the use of artificial intelligence and Internet of things", *Internet of Things*, vol. 22, p. 100 772, Jul. 2023, ISSN: 2542-6605. DOI: 10.1016/j.iot.2023.100772. Accessed: Mar. 26, 2026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660523000951.

[P3$_{AI}$] Q. Shi, Z. Zhang, Y. Yang, X. Shan, B. Salam, and C. Lee, "Artificial Intelligence of Things (AIoT) Enabled Floor Monitoring System for Smart Home Applications", en, *ACS Nano*, vol. 15, no. 11, pp. 18 312–18 326, Nov. 2021, ISSN: 1936-0851, 1936-086X. DOI: 10.1021/acsnano.1c07579. Accessed: Mar. 26, 2026. [Online]. Available: https://pubs.acs.org/doi/10.1021/acsnano.1c07579.

[P4$_{AI}$] H. Sabit, "Artifical Intelligence-Based Smart Security System Using Internet of Things for Smart Home Applications", en, *Electronics*, vol. 14, no. 3, p. 608, Jan. 2025, ISSN: 2079-9292. DOI: 10.3390/electronics14030608. Accessed: Mar. 26, 2026. [Online]. Available: https://www.mdpi.com/2079-9292/14/3/608.

[P5$_{AI}$] A. Rego, P. L. G. Ramírez, J. M. Jimenez, and J. Lloret, "Artificial intelligent system for multimedia services in smart home environments", en, *Cluster Computing*, vol. 25, no. 3, pp. 2085–2105, Jun. 2022, ISSN: 1573-7543. DOI: 10.1007/s10586-021-03350-z. Accessed: Mar. 26, 2026. [Online]. Available: https://doi.org/10.1007/s10586-021-03350-z.

[P6$_{AI}$] N. Raj, A. Sinha, S. Bharddwaj, and A. L. Yadav, "AI-Powered Energy Consumption Optimization for Smart Homes Using IoT", in *2024 International Conference on Computational Intelligence and Computing Applications (ICCICA)*, vol. 1, May 2024, pp. 231–236. DOI: 10.1109/ICCICA60014.2024.10585239. Accessed: Mar. 26, 2026. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10585239.

[P7$_{AI}$] H. R. O. Rocha, I. H. Honorato, R. Fiorotti, W. C. Celeste, L. J. Silvestre, and J. A. L. Silva, "An Artificial Intelligence based scheduling algorithm for demand-side energy management in Smart Homes", *Applied Energy*, vol. 282, p. 116 145, Jan. 2021, ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2020.116145. Accessed: Mar. 26, 2026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261920315555.

[P8$_{AI}$] M. M. Khayyat and B. Sami, "Energy Community Management Based on Artificial Intelligence for the Implementation of Renewable Energy Systems in Smart Homes", en, *Electronics*, vol. 13, no. 2, p. 380, Jan. 2024,

ISSN: 2079-9292. DOI: 10.3390/electronics13020380. Accessed: Mar. 26, 2026. [Online]. Available: https://www.mdpi.com/2079-9292/13/2/380.

[P9$_{AI}$] D. D. Onthoni and P. K. Sahoo, "Artificial-Intelligence-Assisted Activities of Daily Living Recognition for Elderly in Smart Home", en, *Electronics*, vol. 11, no. 24, p. 4129, Jan. 2022, ISSN: 2079-9292. DOI: 10.3390/electronics11244129. Accessed: Mar. 26, 2026. [Online]. Available: https://www.mdpi.com/2079-9292/11/24/4129.

[P10$_{AI}$] A. Rochd et al., "Design and implementation of an AI-based & IoT-enabled Home Energy Management System: A case study in Benguerir — Morocco", *Energy Reports*, Technologies and Materials for Renewable Energy, Environment and Sustainability, vol. 7, pp. 699–719, Nov. 2021, ISSN: 2352-4847. DOI: 10.1016/j.egyr.2021.07.084. Accessed: Mar. 26, 2026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352484721005497.

[P11$_{AI}$] M. A. Torad, B. Bouallegue, and A. M. Ahmed, "A voice controlled smart home automation system using artificial intelligent and internet of things", en-US, *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 20, no. 4, pp. 808–816, Aug. 2022, ISSN: 2302-9293. DOI: 10.12928/telkomnika.v20i4.23763. Accessed: Mar. 26, 2026. [Online]. Available: https://www.telkomnika.uad.ac.id/index.php/TELKOMNIKA/article/view/23763.

[P12$_{AI}$] Y. Lu, L. Zhou, A. Zhang, S. Zha, X. Zhuo, and S. Ge, "Application of Deep Learning and Intelligent Sensing Analysis in Smart Home", en, *Sensors*, vol. 24, no. 3, p. 953, Jan. 2024, ISSN: 1424-8220. DOI: 10.3390/s24030953. Accessed: Mar. 26, 2026. [Online]. Available: https://www.mdpi.com/1424-8220/24/3/953.

[P13$_{AI}$] P. Kulurkar, C. k. Dixit, V. C. Bharathi, A. Monikavishnuvarthini, A. Dhakne, and P. Preethi, "AI based elderly fall prediction system using wearable sensors: A smart home-care technology with IOT", *Measurement: Sensors*, vol. 25, p. 100 614, Feb. 2023, ISSN: 2665-9174. DOI: 10.1016/j.measen.2022.100614. Accessed: Mar. 26, 2026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2665917422002483.

[P14$_{AI}$] M. J. C. S. Reis and C. Serôdio, "Edge AI for Real-Time Anomaly Detection in Smart Homes", en, *Future Internet*, vol. 17, no. 4, p. 179, Apr. 2025, ISSN: 1999-5903. DOI: 10.3390/fi17040179. Accessed: Mar. 26, 2026. [Online]. Available: https://www.mdpi.com/1999-5903/17/4/179.

[P15$_{AI}$] R. E. Alden, H. Gong, E. S. Jones, C. Ababei, and D. M. Ionel, "Artificial Intelligence Method for the Forecast and Separation of Total and HVAC Loads With Application to Energy Management of Smart and NZE Homes", *IEEE Access*, vol. 9, pp. 160 497–160 509, 2021, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3129172. Accessed: Mar. 26, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/9620086/.

[P16$_{AI}$] M. Houshangi, "Data-Driven Smart Home Automation for Energy Efficiency", en, *SN Computer Science*, vol. 6, no. 6, p. 675, Jul. 2025, ISSN: 2661-8907. DOI: 10.1007/s42979-025-04183-y. Accessed: Mar. 26, 2026. [Online]. Available: https://doi.org/10.1007/s42979-025-04183-y.

[P17$_{AI}$] P. Saroha et al., "Enhancing Smart Home Energy Efficiency Using a Hybrid Genetic Algorithm and Improved Dandelion Optimizer", en, *International Journal of Computational Intelligence Systems*, vol. 18, no. 1, p. 323, Dec. 2025, ISSN: 1875-6883. DOI: 10.1007/s44196-025-01076-z. Accessed: Mar. 26, 2026. [Online]. Available: https://doi.org/10.1007/s44196-025-01076-z.

[P18$_{AI}$] A. Orlowski and W. Loh, "Data autonomy and privacy in the smart home: The case for a privacy smart home meta-assistant", en, *AI & SOCIETY*, vol. 40, no. 6, pp. 4171–4184, Aug. 2025, ISSN: 1435-5655. DOI: 10.1007/s00146-025-02182-4. Accessed: Mar. 26, 2026. [Online]. Available: https://doi.org/10.1007/s00146-025-02182-4.

[P19$_{AI}$] Zainab, H. Khan, I. U. Din, M. I. Tariq, A. Khalid, and A. Naz, "An Efficient Implementation of an IoT-Based Smart Home Security System", en, in *Computing and Emerging Technologies*, M. Arif, A. Jaffar, and O. Geman, Eds., Cham: Springer Nature Switzerland, 2025, pp. 249–259, ISBN: 978-3-031-77620-5. DOI: 10.1007/978-3-031-77620-5_18.

[P20$_{AI}$] M. G K, P. L, and R. M, "An AI Smart Refrigerator: Enhancing Food Management with Computer Vision, Sensor, IoT, and Automated Temperature Control", in *2025 International Conference on Emerging Technologies in Engineering Applications (ICETEA)*, Jun. 2025, pp. 1–4. DOI: 10.1109/ICETEA64585.2025.11099672. Accessed: Mar. 26, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/11099672.

[P21$_{AI}$] I. A. Akhinov and M. R. A. Cahyono, "Development of Smart Home System Based on Artificial Intelligence with Variable Learning Rate to Manage Household Energy Consumption", in *2021 International Conference on Artificial Intelligence and Mechatronics Systems (AIMS)*, Apr. 2021, pp. 1–6. DOI: 10.1109/AIMS52415.2021.9466064. Accessed: Mar. 26, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/9466064.

[P22$_{AI}$] N. Nalini, A. Elrashidi, T. Vaishnavi, A. Rosi, B. Manideep, and G. Karthikeyan, "An AIoT-Enabled Smart Home Automation Framework: Intelligent Control, Security, and Predictive Monitoring", in *2025 6th International Conference on Electronics and Sustainable Communication Sys-*

tems (ICESC), ISSN: 2996-5357, Sep. 2025, pp. 634–638. DOI: 10.1109/ICESC65114.2025.11212291. Accessed: Mar. 26, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/11212291.

[P23$_{AI}$] A. Leal-Junior, L. Avellar, W. Blanc, A. Frizera, and C. Marques, "Opto-Electronic Smart Home: Heterogeneous Optical Sensors Approaches and Artificial Intelligence for Novel Paradigms in Remote Monitoring", *IEEE Internet of Things Journal*, vol. 11, no. 6, pp. 9587–9598, Mar. 2024, ISSN: 2327-4662. DOI: 10.1109/JIOT.2023.3323481. Accessed: Mar. 26, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/10275137/.

[P24$_{AI}$] P. Rajesh, F. H. Shajin, and G. Kannayeram, "A novel intelligent technique for energy management in smart home using internet of things", en, *Applied Soft Computing*, vol. 128, p. 109 442, Oct. 2022, ISSN: 15684946. DOI: 10.1016/j.asoc.2022.109442. Accessed: Mar. 26, 2026. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1568494622005622.

[9] International Energy Agency, "Energy efficiency 2025", International Energy Agency, Paris, Tech. Rep., 2025, Accessed: 2026-06-07. [Online]. Available: https://www.iea.org/reports/energy-efficiency-2025.

[P1$_{EC}$] B. R. Cherukuri, "Edge computing vs. cloud computing: A comparative analysis for real-time ai applications", *Int. J. Multidiscip. Res*, vol. 6, p. 29 316, 2024.

[P2$_{EC}$] R. C. Thota, "Optimizing edge computing and ai for low-latency cloud workloads", *International Journal of Science and Research Archive*, vol. 13, no. 1, pp. 3484–3500, 2024.

[P3$_{EC}$] W. Su, L. Li, F. Liu, M. He, and X. Liang, "Ai on the edge: A comprehensive review", *Artificial Intelligence Review*, vol. 55, no. 8, pp. 6125–6183, 2022.

[P4$_{EC}$] N. Waranugraha and M. Suryanegara, "The development of iot-smart basket: Performance comparison between edge computing and cloud computing system", in *2020 3rd International Conference on Computer and Informatics Engineering (IC2IE)*, IEEE, 2020, pp. 410–414.

[P5$_{EC}$] F. C. Andriulo, M. Fiore, M. Mongiello, E. Traversa, and V. Zizzo, "Edge computing and cloud computing for internet of things: A review", in *Informatics*, MDPI, vol. 11, 2024, p. 71.

[P6$_{EC}$] I. Ficili, M. Giacobbe, G. Tricomi, and A. Puliafito, "From sensors to data intelligence: Leveraging iot, cloud, and edge computing with ai", *Sensors*, vol. 25, no. 6, p. 1763, 2025.

[P7$_{EC}$] Z. Chang, S. Liu, X. Xiong, Z. Cai, and G. Tu, "A survey of recent advances in edge-computing-powered artificial intelligence of things", *IEEE Internet of Things Journal*, vol. 8, no. 18, pp. 13 849–13 875, 2021.

[P8$_{EC}$] N. F. Prangon and J. Wu, "Ai and computing horizons: Cloud and edge in the modern era", *Journal of Sensor and Actuator Networks*, vol. 13, no. 4, p. 44, 2024.

[P9$_{EC}$] T. Hao, K. Hwang, J. Zhan, Y. Li, and Y. Cao, "Scenario-based ai benchmark evaluation of distributed cloud/edge computing systems", *IEEE Transactions on Computers*, vol. 72, no. 3, pp. 719–731, 2022.

[P10$_{EC}$] P. McEnroe, S. Wang, and M. Liyanage, "A survey on the convergence of edge computing and ai for uavs: Opportunities and challenges", *IEEE Internet of Things Journal*, vol. 9, no. 17, pp. 15 435–15 459, 2022.

[P11$_{EC}$] S. Duan et al., "Distributed artificial intelligence empowered by end-edge-cloud computing: A survey", *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 591–624, 2022.

[P12$_{EC}$] M. Asim, Y. Wang, K. Wang, and P.-Q. Huang, "A review on computational intelligence techniques in cloud and edge computing", *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, no. 6, pp. 742–763, 2020.

[P13$_{EC}$] M. Trigka and E. Dritsas, "Edge and cloud computing in smart cities", eng, *Future internet*, vol. 17, no. 3, pp. 118–, 2025, ISSN: 1999-5903.

[P14$_{EC}$] M. K, A. Mahida, S. Naveen, Y. K A, and U. Desai, "Edge computing: The future era of cloud computing-a review", in *2025 3rd International Conference on Inventive Computing and Informatics (ICICI)*, 2025, pp. 375–381. DOI: 10.1109/ICICI65870.2025.11069635.

[P15$_{EC}$] B. Sharan, A. K. Sagar, and M. Chhabra, "A review on edge-computing: Challenges in security and privacy", in *2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, 2022, pp. 1280–1286. DOI: 10.1109/ICAAIC53929.2022.9792868.

[P16$_{EC}$] K. Suo, T. N. Nguyen, Y. Shi, J. S. He, and C.-C. Hung, "Keep clear of the edges: An empirical study of artificial intelligence workload performance and resource footprint on edge devices", in *2022 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, 2022, pp. 7–16. DOI: 10.1109/IPCCC55026.2022.9894338.

[P17$_{EC}$] J. D. S. Ong, R. K. C. Billones, R. Concepcion, and N. T. Bugtai, "Comparison of cloud computing and tinyml methods for brain-computer interface in motor imagery problems", in *2023 8th International Conference*

on Business and Industrial Research (ICBIR), 2023, pp. 694–699. DOI: 10.1109/ICBIR57571.2023.10147441.

[P18$_{EC}$] E. A. Adeniyi, S. A. Ajagbe, O. A. Oki, A. O. Adebayo, and O. Olasupo, "Application of machine learning algorithm in cloud-to-edge computing: Analysis and limitations", in *2023 IEEE AFRICON*, 2023, pp. 1–6. DOI: 10.1109/AFRICON55910.2023.10293346.

[P19$_{EC}$] M. Abedi and M. Pourkiani, "Resource allocation in combined fog-cloud scenarios by using artificial intelligence", in *2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC)*, 2020, pp. 218–222. DOI: 10.1109/FMEC49853.2020.9144693.

[P20$_{EC}$] A. Vijayaraj, S. K. R, S. S. M, M. Vaishnavi, P. K, and M. A. V. N. K, "A comprehensive analysis of cloud-edge-iot collaboration for scalable and instantaneous uses", in *2025 4th International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, 2025, pp. 59–63. DOI: 10.1109/ICIMIA67127.2025.11200924.

[P21$_{EC}$] A. Polamarasetti, V. Yammanur, V. V. R. M. Bokka, N. Ravuri, and R. Vadisetty, "Ai-powered edge computing: Enhancing cloud-native ai applications", in *AI and Digital Transformation: Opportunities, Challenges, and Emerging Threats in Technology, Business, and Security*, Cham: Springer Nature Switzerland, 2026, pp. 207–218, ISBN: 978-3-032-07373-0.

[P22$_{EC}$] A. Er-Rahmani, M. El Ghmary, W. Lakhouari, and H. Echoukairi, "Advancements and challenges in cloud computing, edge computing and edge-cloud computing for iot: A comprehensive review", in *Connected Objects, Artificial Intelligence, Telecommunications and Electronics Engineering*, Cham: Springer Nature Switzerland, 2026, pp. 165–170, ISBN: 978-3-032-01536-5.

[P23$_{EC}$] A. Hemmati, P. Raoufi, and A. M. Rahmani, "Edge artificial intelligence for big data: A systematic review", *Neural Computing and Applications*, vol. 36, no. 19, pp. 11 461–11 494, Jul. 2024, ISSN: 1433-3058. DOI: 10.1007/s00521-024-09723-w. [Online]. Available: https://doi.org/10.1007/s00521-024-09723-w.

[10] Raspberry Pi Ltd, *Raspberry pi computer hardware*, 2025. Accessed: Nov. 21, 2025. [Online]. Available: https://www.raspberrypi.com/documentation/computers/raspberry-pi.html.

[11] A. Geitgey, *Face_recognition: The world's simplest facial recognition api for python and the command line*, https://github.com/ageitgey/face_recognition, Accessed: 2026-06-07, 2017.

[12] D. E. King, "Dlib-ml: A machine learning toolkit", *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.

[13] Anthropic, *Prompt engineering overview*, https://platform.claude.com/docs/en/build-with-claude/prompt-engineering/overview, Accessed: 2026-06-15, 2025.

Appendix A Claude Code Development Session Summary

This appendix documents the iterative development session for the facial recognition access control prototype running on the Raspberry Pi 5. Each entry reproduces the exact user prompt followed by a description of the resulting changes made to the system. The appendix content was created using Claude Sonnet 4.6 model by summarizing the conversation into a text file and prompting it to generate an appendix.

Interaction 1

Prompt: *"<context>I am building a smart home access control system for my master's thesis. The Raspberry Pi 5 runs Raspberry Pi OS (64-bit, ARM64) and is connected to a USB camera. The same Docker container must also run on an x86_64 Ubuntu desktop for performance comparison. The system must prevent spoofing attacks by requiring the user to smile before granting access (liveness detection). Collect per-frame timing metrics (detection, recognition, landmark, total in milliseconds), FPS, CPU usage, and RAM usage so that I can compare performance between the two devices.</context> <task>Create a facial recognition access control application in Python for my Raspberry Pi 5. The application should use the face_recognition library for face detection and recognition. When an enrolled face*

is recognized and liveness is confirmed via smile detection, publish an MQTT message to a Shelly 1 smart relay to open the door lock, and publish a separate MQTT message to an ESP8266 microcontroller to control the red, yellow, and green status LEDs. Package the application inside a lightweight Docker container so that it can be deployed on the Raspberry Pi 5 and a desktop PC without modifications.</task> <examples> <example> Scenario: An enrolled face is detected. Expected behavior: Publish MQTT message to ESP8266 to turn on the yellow LED and begin smile detection. If liveness is confirmed, publish MQTT message to ESP8266 to turn on the green LED and publish MQTT message to Shelly 1 to open the door lock. </example> <example> Scenario: An unrecognized face is detected. Expected behavior: Keep the red LED on and do not publish any MQTT message to the Shelly 1. Access is denied. </example> <example> Scenario: An enrolled face is detected but liveness check fails. Expected behavior: Publish MQTT message to ESP8266 to turn the red LED back on. Do not open the door lock. </example> </examples>"

Result: Created the full project from scratch. The following files were produced:

- `app/main.py` — State machine and main camera loop
- `app/config.py` — All tunable parameters via environment variables
- `app/face_processor.py` — Face detection, recognition, and landmark extraction
- `app/mqtt_client.py` — Shelly relay and ESP8266 LED publishing
- `app/metrics.py` — Per-frame CSV logging (`detection_ms`, `recognition_ms`, `landmark_ms`, `total_ms`, `fps`, `cpu_percent`, `ram_percent`)
- `app/enroll.py` — Face enrollment tool
- `Dockerfile` — Two-stage build (compiles dlib from source), supporting ARM64 and AMD64

- `docker-compose.yml` — Compose configuration
- `requirements.txt` — Python dependencies
- `.env.example` — Configuration reference

The state machine implemented the following states: IDLE → PENDING_LIVENESS → ACCESS_GRANTED → COOLDOWN.

Interaction 2

Prompt: *"Build the docker image and run the program, allowing me to enroll my face in the system"*

Result: The Docker image was built (dlib compiled from source, requiring several minutes). A bug was discovered and fixed: the `frame[:, :, ::-1]` NumPy slice for BGR→RGB conversion produces a non-contiguous array that dlib's face encoder rejects. The fix replaced the slice with `cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)` in both `face_processor.py` and `enroll.py`. The enrollment script was rewritten to operate headlessly: a 20-frame camera warm-up, a 3-second countdown, and automatic best-face capture over 5 seconds. Face enrollment completed successfully for the user "Samu".

Interaction 3

Prompt: *"Run the program and test it on my face"*

Result: The MQTT client was updated to handle broker connection failures gracefully instead of crashing. `-network host` was added to the `docker run` command so the container could reach the broker at `127.0.0.1`. Per-frame `INFO` logging was added in headless mode. A sensitivity issue was identified: the state machine reset on every single missed detection frame. This was resolved by introducing `FACE_LOST_TOLERANCE=10`, requiring the face to be absent for 10 consecutive

frames before declaring it lost. Headless mode (`HEADLESS=true`) was enabled to avoid `cv2.imshow` blocking the Qt5 event loop.

Interaction 4

Prompt: *"I want you to send mqtt messages to the broker"*

Result: The root cause was identified: `127.0.0.1` inside a Docker container resolves to the container itself, not the host machine. Adding `-network host` to the `docker run` command caused the container to share the host network stack, enabling correct broker connectivity. MQTT messages were confirmed to be published successfully.

Interaction 5

Prompt: *"I don't think the liveness check for smiles is working at all"*

Result: Two bugs were diagnosed in the landmark-based smile detection algorithm. First, the algorithm required both a mouth aspect ratio (MAR) above 0.15 and corner elevation above 0.05 simultaneously; a closed-mouth smile always failed the MAR condition. Second, the reference point (top lip centre, landmark 51) moves upward during a smile, making the elevation measurement near zero. An attempted fix using the lip midline as a stable reference was tried but produced the opposite problem: the threshold of 0.0 passed every frame. The algorithm was replaced entirely with OpenCV's built-in Haar cascade smile detector (`haarcascade_smile.xml`), applied to the lower 55% of the face region of interest with `minNeighbors=20`. Access was granted successfully immediately after the fix.

Interaction 6

Prompt: *"I want you to start this again, but not in headless mode"*

Result: The Docker image was rebuilt using the full `opencv-python` package instead of `opencv-python-headless`, and Qt5 system libraries were added to the runtime stage of the Dockerfile. The container was launched with X11 forwarding (`DISPLAY=:0`, `/tmp/.X11-unix` volume, and `QT_QPA_PLATFORM=xcb`). An on-screen overlay was added showing the face bounding box (colour-coded by recognition result), the user's name, the current state, FPS, CPU%, RAM%, and progress counters for the neutral and smile phases.

Interaction 7

Prompt: *"currently the user has to get really close to the camera to smile. Is it possible for you to zoom into the area so that the user does not need to be as close to the camera"*

Result: The root cause was that the Haar cascade was running on the half-resolution processing frame (`PROCESS_SCALE=0.5`), yielding a mouth region of interest of only ∼30 px at normal distance. Three changes were applied in `face_processor.py`: (1) a full-resolution grayscale image (`gray_orig`) was computed for smile detection; (2) face locations were scaled back to original frame coordinates before cropping; (3) a new parameter `SMILE_ROI_UPSCALE=3.0` upscales the mouth ROI by 3× before passing it to the cascade, improving detection at normal sitting distances. The cascade `scaleFactor` was lowered from 1.7 to 1.5 for more thorough scanning.

Interaction 8

Prompt: *"Can you make it so that the user can't be smiling at the start. Because otherwise you can just show a picture of someone smiling and it passes them through"*

Result: The liveness check was redesigned as a two-phase state machine with an explicit anti-spoofing mechanism. The new states are: IDLE → PENDING_NEUTRAL

→ PENDING_SMILE → ACCESS_GRANTED → COOLDOWN. A face already smiling when first detected is immediately rejected with a "possible spoof" log entry and the red LED remains on. The system only transitions to PENDING_NEUTRAL if the face is detected without a smile. A smile during the neutral phase resets the neutral counter to zero. This design defeats both a static smiling photograph (skips the neutral phase) and a static neutral photograph (never reaches the smile phase). The parameter `NEUTRAL_CONFIRM_FRAMES=10` was added to `config.py`.

Interaction 9

Prompt: *"Make it so that you need to hold a neutral face for 5 frames and smile for two. No need for more"*

Result: The container was restarted with updated environment variables (`NEUTRAL_CONFIRM_FRAMES=5`, `SMILE_CONFIRM_FRAMES=2`), requiring no image rebuild. A new blocking issue was also diagnosed: `face_recognition` failed to match the smiling face against the enrolled neutral photograph (smile-induced geometry change pushes the distance above the 0.55 tolerance), causing the state machine to reset during the smile phase. The fix saves the verified identity as `verified_name` once confirmed in PENDING_NEUTRAL and skips re-recognition during PENDING_SMILE, tracking any face present instead. The full flow (enrol → neutral hold → smile → access granted → MQTT published) completed successfully.

Interaction 10

Prompt: *"Does this work now in my raspberry pi?"*

Result: The image built on x86_64 was not compatible with the Raspberry Pi 5 (ARM64) and had to be rebuilt natively on the device. The recommended steps were: copying the project with `rsync`, installing Docker on the Raspberry

Pi 5, running `docker compose build` (estimated 30–60 minutes due to dlib compilation), and configuring `HEADLESS=true` and the correct `MQTT_BROKER` address in `.env`. Potential issues were noted: missing ARM64 OpenCV wheel (switch to `opencv-python-headless`), camera device index (`CAMERA_INDEX=1`), and reduced frame rate (lower resolution recommended). The user confirmed: "It worked."