Smart Building Environment Monitoring and Control System using Machine Learning

by

YEANAT HOSSAN NELOY
(211020050-5)

A report submitted in partial fulfillment of the requirements for the degree of
Bachelor of Engineering (Computer Engineering)

Faculty of Intelligent Computing
UNIVERSITI MALAYSIA PERLIS

2025

UNIVERSITI MALAYSIA PERLIS

DECLARATION OF REPORT

Author's Full Name	:	YEANAT HOSSAN NELOY
Title	:	SMART BUILDING ENVIRONMENTAL MONITORING AND CONTROL SYSTEM USING MACHINE LEARNING
Date of Birth	:	4 FEBRUARY 2002
Academic Session	:	2024/2025

I hereby declare that this report becomes the property of Universiti Malaysia Perlis (UniMAP) and to be placed at the library of UniMAP. This report is classified as:

☐	**CONFIDENTIAL**	(Contains confidential information under the Official Secret Act 1997)*
☐	**RESTRICTED**	(Contains restricted information as specified by the organization where research was done)*
☑	**OPEN ACCESS**	I agree that my report to be published as online open access (Full Text)

I, the author, give permission to reproduce this report in whole or in part for the purpose of research or academic exchange only.

YEANAT HOSSAN NELOY

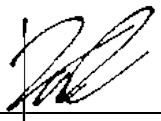
SIGNATURE OF STUDENT

A02713382

(NEW IC NO. /PASSPORT NO.)

Date: 12 July 2025

Checked and approved by:



SIGNATURE OF SUPERVISOR

Ir. ROSDISHAM ENDUT

NAME OF SUPERVISOR

Date: 14 July 2025

Supervisor official stamp

NOTES : * If the thesis is CONFIDENTIAL or RESTRICTED, please attach with the letter from the organization with the period and reasons for confidentiality or restriction.

UNIVERSITI MALAYSIA PERLIS

PANEL APPROVAL AND DECLARATION SHEET

Author's Full Name	:	YEANAT HOSSAN NELOY
Title	:	SMART BUILDING ENVIRONMENTAL MONITORING AND CONTROL SYSTEM USING MACHINE LEARNING
Date of Birth	:	4 FEBRUARY 2002
Academic Session	:	2024/2025

This project report has been found satisfactory in terms of scope, quality and presentation as partial fulfilment of the requirement for the Bachelor of Engineering (Computer Engineering) in Universiti Malaysia Perlis (UniMAP).

Checked by:	Checked by:
SIGNATURE OF PANEL 1	**SIGNATURE OF PANEL 2**
ASSOC. PROF. IR. TS. DR. SAIDATUL NORLYANA AZEMI	ASSOC. PROF. IR. TS. DR. JUNITA MOHD. NORDIN
NAME OF PANEL 1	**NAME OF PANEL 2**
Date: 14 July 2025	Date: 14 July 2025
Panel official stamp	Panel official stamp

ACKNOWLEDGMENT

First and foremost, I would like to express my deepest gratitude to my supervisor, Ir. Rosdisham Endut, for guiding and mentoring me throughout my final year project on Smart Building Environment Monitoring and Control System Using Machine Learning. His insightful feedback, expertise, and patient support were invaluable in steering my research in the right direction. His encouragement and vast knowledge helped me navigate challenges and deepen my understanding of the subject.

I would also like to extend my heartfelt thanks to my parents and family members for their unwavering love, support, and motivation. Their belief in my abilities and continuous encouragement gave me the strength to persevere through the demanding phases of this project.

Additionally, I am grateful to my friends for their support, brainstorming sessions, and moral encouragement. Their assistance and positivity provided much-needed motivation during difficult times and helped me stay focused and complete my project successfully.

Finally, I would like to thank everyone who contributed, directly or indirectly, to the successful completion of my final year project. Your support is truly appreciated.

Thank you all so much.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
AMI	Advanced Metering Infrastructure
API	Application Programming Interface
BIM	Building Information Modeling
BMS	Building Management Systems
CAD	Computer-Aided Design
CO_2	Carbon Dioxide
GPU	Graphics Processing Unit
IoT	Internet of Things
LED	Light Emitting Diode
LEED	Leadership in Energy and Environmental Design
ML	Machine Learning
PIR	Passive Infrared
TPU	Tensor Processing Unit
UI	User Interface
ADC	Analog-to-Digital Converter
MQTT	Message Queuing Telemetry Transport
SCL	Serial Clock Line (I^2C)
SDA	Serial Data Line (I^2C)
TPU	Tensor Processing Unit
TFlite	TensorFlow Lite
6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks
ESP32	Espressif Systems 32-bit Microcontroller

LIST OF SYMBOLS

°C	Degrees Celsius (Temperature)
%	Percentage (Humidity, efficiency, etc.)
ppm	Parts per million (CO_2 measurement)
µg/m³	Micrograms per cubic meter (air quality metrics)
V	Volts (electrical potential)
Lux	Unit of illuminance (light intensity)
S	Seconds (time)
Hz	Hertz (frequency)
Ma	Milliampere (current)
W	Watts (power—implied in power consumption discussions)
Ω	Ohms (resistance—implied for pull-up resistors in diagrams)

SISTEM PEMANTAUAN DAN KAWALAN PERSEKITARAN BANGUNAN PINTAR MENGGUNAKAN PEMBELAJARAN MESIN

ABSTRAK

Penyelidikan ini adalah tentang reka bentuk dan pembangunan Sistem Pemantauan dan Kawalan Persekitaran Bangunan Pintar yang komprehensif, dengan teknologi Peranti Berhubung dan Pembelajaran Mesin. Sejajar dengan perkembangan bangunan pintar, keperluan sistem sudut yang bertuai terus meningkat. Sistem kelak membantu meningkatkan keselesaan penghuni, meningkatkan kecekapan tenaga, dan menggalakkan kemampanan. Dalam kajian ini, kami menggariskan rangka kerja binaan senibina yang mampu memonitor parameter-parameter kritikal seperti suhu, kelembapan, kualiti udara, dan penghunian melalui sistem sensor IoT, data processor, dan fungsi ML yabg canggih. Penggabungan antara IoT dan ML membolehkan pemantauan data waktu sebenar dan penjelasan terus menerus serta adaptif yang membawa kepada pengurusan sumber yang lengkap dan pembaziran kos yang masif. Dengan pemantauan segera dan dinamik, bukan sahaja sistem ini dapat mengurangkan kos sumber tenaga, tetapi mereka juga dapat meningkatkan kemudahan dan kesihatan dalaman. Sistem mencadangkan kemungkinan bagaimana model ML prediktif boleh meramal perubahan sepenuhnya bagi fungsi persekitaran dalaman, dan mengatur sistem sendiri untuk memuahi prestasi optimum. Oleh sebab pembelajaran dan data terus berlaku ekspotensial, ia boleh menyelesaikan ketidakcekapan sistem dalaman semasa, kerana tidak dapat bertindak balas terhadap tingkah laku dan pembolehubah yang serba-perubahan. Pembolehubah inovatif yang disyorkan dijalankan memandu kepada pembangunan bandar dan bangunan pintar, menyokong peralihan ke arah pengurusan automatik dan mendalam yang dibuat oleh pengguna. Persetujuan salah satu sistem yang dikawal oleh ML adalah aspek penting yang membantu memasarkan projek ini untuk pengeluaran di arena. Semasa memperhitungkan impak mudah alih serta emisi karbon dari bangunan awam dan korporat, sumber ini akan memberikan penjimatan tenaga yang besar dan pengurangan kesan pada sumber sehingga 20%. Selain itu, dengan liputan sempurna data dan kawasan kritikal suhu , sistem ini akan mengarahkan kepada pengenalan penggunaan bahan api semula jadi dan sumber yang dikitar semula.

SMART BUILDING ENVIRONMENTAL MONITORING AND CONTROL SYSTEM USING MACHINE LEARNING

ABSTRACT

This research work presents a design and development of a complete Smart Building Environment monitoring & control system in light of the recent advancements in Internet of Things (IoT) technology as well as Machine learning (ML). In line with the advancements in smart buildings, the demand for integrated corner systems continues to grow. The system aims to enhance occupant comfort, improve energy efficiency, and promote sustainability. Here an architecture is introduced where an existing integrated network of multiple (IoT) sensors, followed by high-level Machine Learning (ML) algorithms in data processing units, can potentially monitor the most critical parameters of interest to the environment (temperature, humidity, air quality and occupancy). For instance, in terms of resource management, the synergetic use of IoT with ML leads to collection and analysis of data in real time and making adaptive changes which are capable of optimizing one another and thus lead to resource management and reduced operational cost. This enables the system to optimize energy consumption according to time and place, but also to adjust environmental parameters in order to improve comfort and health in closed environments. The system learns to predict environmental changes by borrowing predictive machine learning models and uses that information to autonomously control building systems to ensure optimal efficiency. While bringing data-driven insight holes to the inefficiency of traditional static systems, By the concept of smart building system, the response to the behavior of the occupant and variable external environmental factors becomes adaptive/a dynamic response. The grounds for this are smart urban infrastructure. The future of automated, self-managing, and sustainable building management systems. Also, the immediate energy saving and operating cost reduction through this system itself can be considered a long-term impact of reducing carbon footprints and conservation of energy in the environment. The research study identifies the requirement of a new generation republik; the work synthesis defines republik as smart environments promoting human comfort and human well-being and energy efficiency. These innovations are a springboard for future smart building projects, closely associated with global initiatives and developments in clean technology and sustainable urban development.

CHAPTER 1 : INTRODUCTION

1.1 Introduction and Background

The advancement of smart building technologies is revolutionary modern urban infrastructure by adding intelligent systems that improve energy efficiency, occupant comfort, and sustainability. However, advancement in Internet of Things (IoT) technology and machine learning (ML) have made it possible to develop dynamic, adaptive systems capable of monitoring and managing interior conditions in real time (Gubbi & Buyya, 2013).

In smart buildings, IoT sensors gather and relay data on a range of environmental factors such as temperature, humidity and air quality. This information is fed into machine learning algorithms, which generate predictive adjustments, and thereby an environment that reacts autonomously. Such integration is not only achieving operational efficiency but also conducive to minimising energy consumption and building a healthier indoor environment (Lee and Lee, 2016).

New sensors that are cost-efficient and greatly increased power of compute mostly due to the availability of powerful, small, cheap, and energy-efficient devices as well as the cloud allows smart building systems to optimize lighting, air conditioning, and other systems to be more energy efficient while still serving the needs of all building inhabitants. This project aims to build a Smart Building Environment Monitoring and Control System that leverages IoT and machine learning offering a data adaptive building environment that grants benefits to sustainable and intelligent urban living.

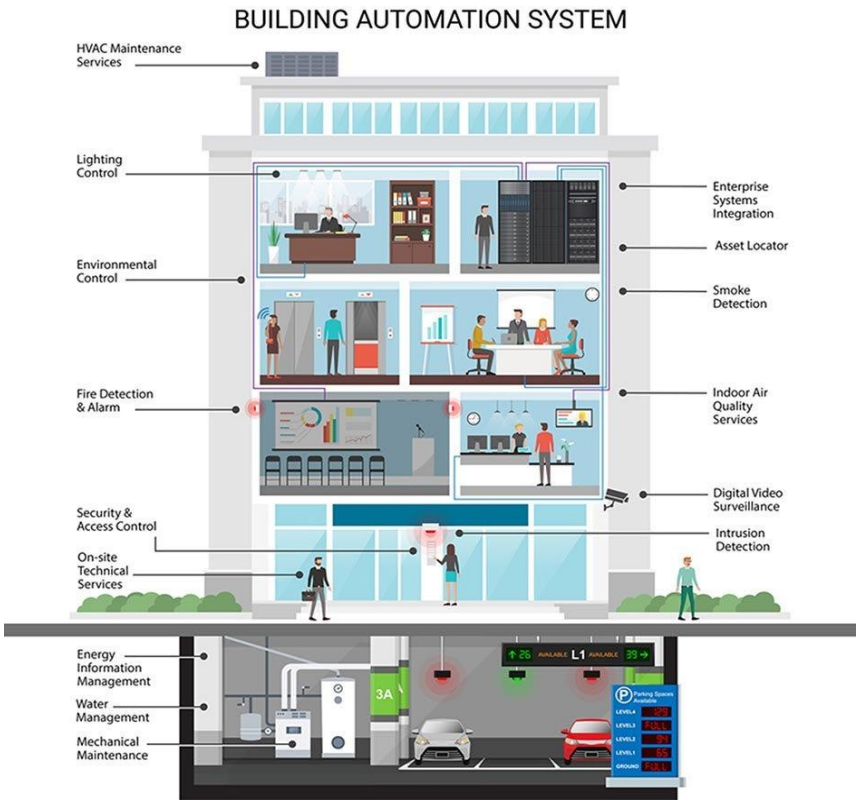


Figure 1.1 Smart Building with sensors
Source : (Mark Baker, 2022)

Smart building systems, driven by the IoT and machine learning, enhance operational efficiency, predictive maintenance, and fault detection. Predictive maintenance monitors equipment and the surrounding environment to foresee developing issues so they can be fixed before the machinery breaks, often leading to reduced maintenance costs and production downtime. They extend the life of building infrastructure, and also decrease environmental footprints. It has the ability to analyze large datasets, recognize patterns and apply that information to make recommendations for more-resistant, adaptable, energy-efficient buildings of the future. As urbanization increases, this smart systems are becoming critical to ensure buildings and structures internally work efficiently with least amount of energy consumption.

1.2 Problem Statement

The existing Building Management Systems (BMS) are based on static schedules with human control, so they cannot respond to dynamic changes in occupant presence or environmental changes effectively. Such inflexibility results in energy wastage, subpar indoor environments, and higher operating expenses. For instance, while air conditioning might be set to high power even when the desired temperature and humidity have been attained, lighting systems cannot adjust to user presence or sunlight. Both adding to the inefficiencies. Such reactive systems are expensive and energy-hungry. Extending beyond these solutions presents a wealth of challenges that can only be tackled by an intelligent, adaptive solution using IoT data and ML to constantly analyze what is going on in real-time conditions, learn how people consume energy, and dynamically adjust to improve energy efficiency, occupant comfort, and system performance across the building.

It causes a lot of energy waste, where air conditioning and humidity are practically uncontrolled and without real time data as most of the buildings do not have smart ventilation systems. This inefficiency adds to the operating costs, with the cost of energy increase, so non-adaptive systems indeed cost more. Moreover, unrealistic static setups can cause discomfort and not conducive to task execution due to the inefficiency of its functioning. They do not adjust itself during the different load (occupancy) as well as the changing of weather conditions. The systems simply react to changes instead of analyzing the future of the occupant needs and environmental changes. Moreover, the absence of IoT integration inhibits real-time tracking of temperature, humidity, and lighting, hindering operational performance.

1.3 Research Objectives

In order to achieve this project's outcomes, few objectives were made to be achieved.

The research objectives of this study are:

1. To design a comprehensive IoT-based monitoring system that records temperature, humidity and air quality data in real time.

2. To implement machine learning algorithms to examine IoT sensor data and find trends in environmental requirements and occupancy.

3. To test an adaptive control system that automatically modifies lighting, air conditioning and other environmental controls based on data-driven insights.

1.4 Scope of Work

The purpose of this project is Create, Implementation and Evaluation of a smart building monitoring and controlling device for indoor environment including residences and workplaces. This framework is based on IoT sensors and ML algorithms with the primary purpose of automating the changing of our environment in accordance with predictive analytics in real time.

To ensure maximum indoor comfort and energy efficiency, environmental variables such as occupancy, temperature, humidity and air quality should be recorded. An array of IoT sensors distributed throughout the interior will capture real-time data which will then be processed through algorithms designed for predictive and adaptive control of the building systems. This may include automatic air conditioning, lighting and ventilation activated by changing conditions to maximize energy efficiency as well as occupant comfort.

No environmental dynamics like outside air or professional environments which require to use some additional equipment are being researched in this study. Rather, it applies to average indoor spaces with typical environmental management and air control systems. In addition, while smart building management covers an array of ideas, this study deducts reflective processes of energy efficiency and occupant well-being in the double that is sensitive to green management of buildings. The adaptability of the system to varying indoor conditions will be evaluated and serve as a measure by which versatility of the system will be ascertained.

This study has some limitations. The IoT architecture will be implemented on Arduino boards and ESP32 microcontrollers with limited processing and memory resources. Tensor Flow will be the ML (machine learning) framework, and the model will be built and trained on google colab. This might result in constraints such as session duration, processor capacity. Given the limitation of ESP32 devices, these constraints can affect the deployment of the machine learning models utilized. Despite the above mentioned limitations, the project aims at building a solid and flexible interior smart building management system.

1.5 Thesis Outline

The setup of this thesis consists of introduction, literature review, methodology, result and discussions and finally conclusion, with each chapter serving as a building block for the next one, scheming a smart building considering all proposed output.

For **Chapter 1**, Introduction, the research problem statement, objectives, and scope of work is described. This chapter describes the limitations of classical BMS (Building Management Systems), and the requirement of smart building solution based on IoT (Internet of things) and ML (Machine Learning) technologies.

For **Chapter 2,** Literature Review, recent studies on IoT-enabled systems, machine learning methods for predictive control, and smart building are explored. In order to find gaps and chances for improvements, the literature on energy management techniques, real-time heating, ventilation, system control, and automation in built environments are thoroughly inspected.

For **Chapter 3,** Methodology, the design and development process of the smart building system is described. This covers data collection, machine learning methods, IoT sensors and system integration. The system architecture, data preprocessing strategies and predictive model training are explained.

For **Chapter 4,** Results and Discussion, the final results of the proposed system in evaluating the performance will be explained clearly. This section compares the system's adaptive features against standard BMS based on metrics such as functionality, occupant comfort, energy efficiency and model prediction fidelity. The robustness of the system is showcased by using some key inferences and insights.

For **Chapter 5,** Conclusion and Future Work, this chapter provides an overview of the project of the system within the smart building technology domain and identifies shortcomings of the system and suggests areas for future work. The project results will be concluded to confirm the success of this project.

1.6 Summary

The integrated machine learning and Internet of Things (IoT) based smart building environment monitoring and control system described in this chapter serves as an important step toward better building efficiency and improved comfort levels among inhabitants. The challenges present the shortcomings of traditional systems while the setting highlighted the benefits of data-driven building automation. These goals helped to guide the creation of a data-driven and responsive control system. Specifying buildings and traditional environmental systems defined the project limits. This chapter establishes the foundation for the subsequent chapters that explore the literature review, methodology and findings associated with developing a smart building management system.

CHAPTER 2 : LITERATURE REVIEW

2.1 Introduction

This chapter explains the integration of the various technologies, what they can be used for, benefits, challenges, and tools needed to integrate them. This portion of literature review that has been done is the overview information needed for the research project.

2.2 Fundamental of Smart Buildings

Smart buildings allows us to transform urban environment management and monitoring using machine learning (ML) algorithms and Internet of Things (IoT) sensors to decrease energy use, increase occupant comfort, and improve overall operational efficiency. Researchers and developers can use all of these machine learning models with the help of Google Colab and TensorFlow which provide the required tools to build, experiment, and optimize. Specialization is possible at a top machine learning library that relieves the construction and training of complicated models. TensorFlow, and Google Colab, a cloud-based platform provides simultaneous collaboration and computation. Therefore, they are all part of a larger procedure that progressively equips us with a continuously efficient real-time data processing and prediction framework. (Bellido-Montesinos et al., 2019; Zafar et al., 2020).

As urbanisation accelerates and energy needs keep climbing, the need for smart building solutions is increasingly pertinent. While balancing the processing power of TensorFlow, Google Colab and real-time information from IoT devices show a proposal in which researchers can take advantage of this as a possibility to generate prediction models that optimize the consumption of energy and the operation of buildings (Liu and He, 2023).

2.3 The Importance of Smart Buildings and Energy Efficiency

The effects of smart buildings on the economy and environment conventional structures emit tonnes of greenhouse gases into the atmosphere and use a lot of energy which raises utility prices. Real-time data collecting and the use of machine learning algorithms to optimize energy use are the two ways that smart buildings solve these inefficiencies. More advanced models such as TensorFlow and Google Colab can be used to predict, alter and optimise energy usage providing sustainability systems at lower operational costs (Heller et al., 2015)

2.3.1 Environmental Impact and Renewable Energy Integration

Renewable energy sources for instance solar, wind are essential in fighting climate change. The use of TensorFlow models to forecast renewable energy production and weather patterns charge to adapt energy consumption. Machine Learning models help in self regulating energy distribution and storage as well as the use of renewable energy successfully. The deep learning workloads in Google Colab facilitate researchers for this. This strategy can improve energy grid stability and reduce dependence on fossil fuels (Su and Wang, 2020).

Sustainable urbanism of the future relies upon building operations and the assimilation of renewable energy technologies for smart buildings. Large data sets and complex algorithms require sufficient resources, making Google Colab a good environment to train and test these models. Building integration makes buildings more efficient overall and minimizes their carbon footprint (Bellido-Montesinos et al., 2019; Zafar et al., 2020).

2.3.2 Economic Implications and Case Studies

Smart building technologies have substantial long-term positive financial returns. Lower energy costs were achieved by optimising energy management systems generated by TensorFlow models, created and executed on Google Colab. Even though smart technology can be a

substantial investment upfront, the savings on energy costs make it worthwhile. ML models trained on Google Colab, for instance, may recognize periods of high energy usage and proactively adjust heating and cooling systems which can lead to considerable energy bill savings (Liu and He, 2023).

Case studies have demonstrated these financial benefits. In a project for a large office complex in New York, TensorFlow based models saved 40% of energy expenses, illustrating that smart technology investments can be economically viable. Utilising the collaborative capabilities of Google Colab, developers and energy managers can then work together to further optimise the models for efficiency and cost (Bellido-Montesinos et al., 2019).

2.4 The Role of IoT in Smart Buildings

IoT technology, which makes extensive data collecting and system automation possible is essential to the functioning of smart buildings. IoT technologies help automate management as power demands rise due to rapid industrialization. IoT allows a clean connection that is uninterrupted between the devices and users regardless of the time and places which makes it possible for a more efficient control of energy consumption (Poyyamozhi et al., 2024). TensorFlow and Google Colab improve the capacity to effectively analyse and interpret this data, offering a strong basis for making decisions in real time.

2.4.1 IoT Sensors and Their Applications

Internet of Things (IoT) sensors help in controlling the environmental conditions, occupancy and energy usage in smart buildings. Sensor data can easily produce large datasets optimal for modeling and analyses on TensorFlow on Google Colab. Examples of the IoT sensors include:

i) PIR (Passive Infrared) Sensor : This sensor is used to detect motion and occupancy by detecting the infrared light already emitted from the residents. They are employed to control lighting and ventilation systems, minimizing energy wastage in empty areas (Su and Wang, 2020).

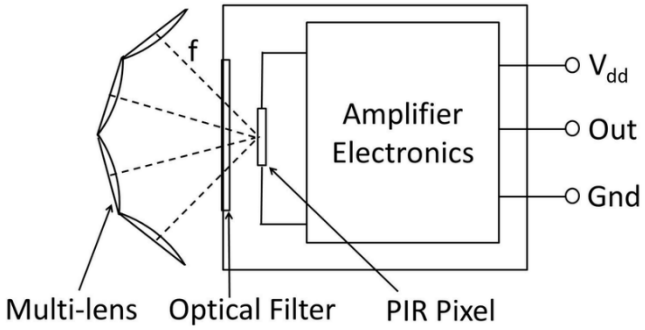


Figure 2.1 Block Diagram of a PIR Sensor
Source : (Upadrashta et al., 2016)

ii) DHT11 Sensor : This sensor measures temperature and humidity and is critical to keeping climate in check. It does this by feeding sensor data into machine learning models which are used to optimize heating and ventilation systems, improving energy efficiency (Zafar et al 2020).

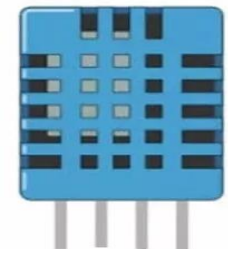


Figure 2.2 DHT11
Source : (Zahidali, 2024)

iii) CO2 Sensor : CO2 sensors are designed to assist in monitoring the indoor air quality by measuring levels of carbon dioxide. When high CO2 levels are detected, the system launches into a full-scale action mode as it automatically increases ventilation to create a healthier atmosphere (Mohammadi & Rashid, 2018).

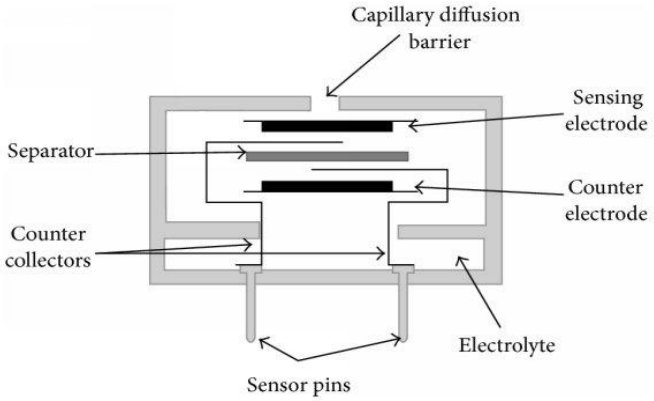


Figure 2.3 Internal Structure of CO2
Source : (Armando et al., 2016)

iv) Luminosity Sensor : Luminosity sensors measure ambient light levels and adjust table settings. It avoids over-lighting by dimming or switching off the lights when enough natural light exists, thus reducing electricity use (Bellido-Montesinos et al., 2019).

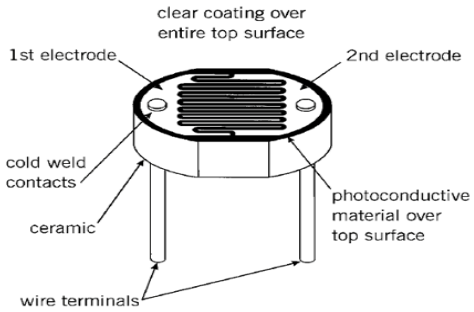


Figure 2.4 Diagram of Luminosity Sensor
Source : (Adabox 001, 2016)

It is researchers and engineers making these TensorFlow models and testing them through Google Colab which enables to harness the power of the platform in the cloud that includes processing large amounts of sensor data and providing real-time streaming actionable insights to facilitate real-time building automation (Liu and He, 2023).

2.4.2 Communication Protocols for IoT Systems

The successful deployment of IoT in a smart building is dependent on reliable communication protocols that enable all devices to communicate smoothly (Gubbi & Buyya, 2013). TensorFlow models need the right data in time, so the communication protocols should be chosen wisely. Common protocols include:

i) KNX and BACnet: Supported in secure building automations through secured communication between the different IoT at different site. Using a combination of KNX and BACnet creates data that can be used to build TensorFlow models that are recommended for smart buildings (Su and Wang, 2020).

ii) Zigbee and Z-Wave: Excellent for the low-power sensor network for further data acquisition for analytics on TensorFlow in Google Colab (Bellido-Montesinos et al., 2019).

iii) Wi-Fi and Bluetooth: This consumes more energy than the above approaches, but it is more appropriate for applications requiring higher throughput for final video-based occupancy detection and to transmit high-rate sensor data. Inside the buildings, Google Cloud-socketed TensorFlow models crunch Wi-Fi and Bluetooth communication data to spot trends and deliver real-time system adjustments (Su and Wang, 2020).

iv) Ethernet and 6LoWPAN: With a wired Ethernet network and IP based 6LoWPAN for low power wireless, a reliable wired connection is critical for these mission critical applications.

The information collected through Ethernet and 6LoWPAN can be processed with TensorFlow models to improve accuracy and use for intelligent buildings (Liu and He, 2023).

In this case, these communication protocol is working in a way to ensure relativity in terms of data transfer from the sensors to the building management system ensuring only the relevant data is being supplied to the TensorFlow Models present in the Google colab through the google drive optimizing the energy full function (Liu and He,2023).

2.5 Machine Learning for Intelligent Control

The AI systems drive smart buildings powered by TensorFlow ML algorithms is trained on Google Colab. These algorithms enable predictive and adaptive control of operations in buildings by processing massive amounts of data collected from the IoT sensors.

2.5.1 Predictive Analytics and Energy Management

Predictive analytics in smart buildings typically use supervised learning models, such as Linear Regression and Decision Trees. These models use historical data related to various IoT sensors that provide readings for factors such as temperature, humidity, and CO_2 levels, to predict upcoming environmental characteristics and future energy consumption patterns.. Moreover, it accelerates model training using of its GPUs and TPUs. These kinds of models also help reduce the waste of energy in predicting peak consumption and optimising servo motor setting for ventilation (Lee & Lee, 2016).

Google Colab can be used for training even complex ML models including deep neural networks for recognizing small trends in the energy consumption. These models are continuously augmented to allow smart buildings to behave more complementarily. By combining the predictions of several decision trees, Random Forest, an ensemble learning approach, can be

used to increase the accuracy and consistency of predictions. In order to have the ability to explore different model architectures and exploit them at their best in a cooperative way, Google Colab gives the versatility that is required in an interactive environment that provides not only to seek to excel beyond the comfort zone but also to work in an interlinked one (Bellido-Montesinos et al., 2019).

2.5.2 Real-Time Adaptive Control and Learning

Real-time control is a key component of smart building management. TensorFlow models can subsequently be configured on edge or cloud servers to respond immediately to inputs from the sensor. Google Colab's collaborative tools are particularly effective for building out these types of models because they allow teams to modify algorithms and share insights in real time. TensorFlow models, for example, can process video feeds to detect human presence and adjust HVAC and lighting systems accordingly, thus enhancing energy efficiency and occupant comfort (Kamilaris & Pitsillides,2016).

Adaptive real-time control is achieved through reinforcement learning, specifically Q-Learning or Deep Q-Networks. The system learns, by way of interaction with the environment, what the best strategies are for modifying Air ventilation, lighting, and ventilation systems to ensure energy efficient operation whilst still meeting the needs of occupants. The ability of these models to learn continuously from feedback allows the system to respond to dynamic conditions like varying occupancy or changing weather.

By using the Google Colab computing resources functionality, these models may be rapidly tested and optimized to ensure optimal functioning in dynamic environments (Zafar et al., 2020).

2.6 Key Technologies for Smart Buildings

Advanced technology combined with powerful software helps smart buildings function. Google Colab and TensorFlow are written in large part to provide machine learning (ML) used by many technologies.

2.6.1 Hardware Components and Their Roles

Smart buildings are equipped with various hardware components that collect data and execute control actions. These include:

i) Microcontrollers and Smart Sensors: Raspberry Pi and Arduino boards are connected with sensors and actuators. These devices gather data which TensorFlow models then process to get the most out of building operations. For instance, the models in Google Colab are easier to train which can allow a more scalable method for data analysis and model deployment (Liu and He, 2023).

ii) Smart Meters and Actuators: Actuators execute command from machine learning predictions and smart metres track energy consumption. Data is retrieved using Google Colab, a cloud-based service that allows for training and testing models while TensorFlow models is used to analyse the metre data to improve energy usage (Bellido-Montesinos et al., 2019).

2.6.2 Software Platforms and Data Processing Techniques

Data processing and building ML models are crucial to getting smart buildings to work as intended. It uses TensorFlow, one of the most widely adopted ML libraries that allows to build, train and deploy predictive models in conjunction with Google Colab, a real-time computing and collaboration cloud-based platform. The integration of Cortana Intelligent Suite and AI Mirroring Model allows ideal data processing and machine learning development for any smart buildings where it offers the best solution for an increasingly complicated environment (Bhatnagar, 2018).

i) **Google Colab**: Provides free access to powerful GPUs and TPUs online allowing to build and deploy TensorFlow models easily. It allows researchers to work together in real time, share code and perform complicated calculations without requiring local hardware. With the help of interface with TensorFlow, it is straightforward to test several algorithms and iterate quickly (Bellido-Montesinos et al., 2019).

Figure 2.5 Google Colab
Source : (Radovanovic, 2023)

ii) **TensorFlow**: The TensorFlow framework is flexible and generalizable with high performance and good scalability for predictive and adaptive control methods. The strong API allows to create the model and able to ingest big data from IoT sensors, whereby smart buildings can aggregate and beat their energy consumption and other operational features (Su and Wang, 2020).

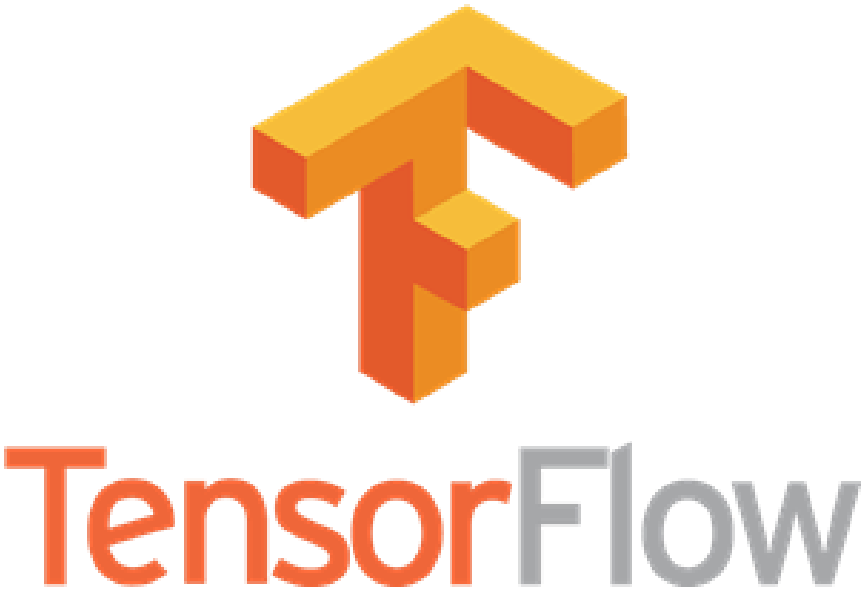


Figure 2.6 Tensor Flow logo
Source : (BotPenguin, 2024)

2.7 Building Information Modeling (BIM) and Workflow Analysis

BIM (BuildIng Information Model) is the main component of the smart buildings, features in design and operation. BIM distils live data from IoT sensors, yielding insights into a building's performance and helping shape better decision making (Azhar et al., 2008).

2.7.1 BIM Implementation for Energy Efficiency

Software like BIM allows to have an energy performance model right in the design stage. These results can be used as inputs to TensorFlow models trained via Google Colab to indicate design changes that can result in more energy savings. In effect, BIM models capture real-time data taken from IoT sensors to optimise building operations, guaranteeing an improved energy performance along the entire life cycle of a building (Bellido-Montesinos et al., 2019).

Hence it is easy to train these models in a scalable computing environment like Google Colab. Such integration aids the monitoring and performance analysis allowing data-driven energy optimisation solutions to be applied (Liu and He, 2023).

2.7.2 Global Adoption Trends and Case Studies

A global increase in the use of Building Information Modeling (BIM) may be driven by a need for more sustainable and efficient building methods. BIM can potentially reduce building costs and improve sustainability for its use for public projects. TensorFlow models created on Google Colab have been used to maximise energy use and improve stakeholder collaboration in BIM projects. An example would be the use of creative design using machine learning algorithms for the conservation of energy as observed in the Valladolid BIM competition (Su and Wang, 2020).

These case studies demonstrate the value of integrating BIM with TensorFlow and Google Colab. By standardising data formats and promoting real-time collaboration, these tools facilitate more efficient and sustainable building practices (Zafar et al., 2020).

2.8 Challenges and Considerations

While the benefits of smart buildings are significant, several challenges must be addressed to fully realize their potential.

2.8.1 Data Privacy and Security

The widespread use of IoT devices in smart buildings raises questions about data privacy. The TensorFlow models handle large amounts of data which need protection in order to prevent undesired access. Though Google Colab provides secure environments for data analysis,

developers must implement secure communication protocols and encryption to protect sensitive data (Bellido-Montesinos et al., 2019).

Another danger is the issue of cybersecurity as hackers could access weaknesses in IoT networks. Since intrusions are possible, the design of such systems should be secured to reduce the potential threat with the help of intrusion detection systems, firewalls, and frequent updates of software (Liu and He, 2023; Su and Wang, 2020).

2.8.2 Technical and Integration Complexities

The integration of tunable TensorFlow models, IoT devices, and building management systems requires cross-sectoral insight and expertise. Greening legacy buildings is also an expensive and complicated challenge especially if those buildings do not have the infrastructure to accommodate smart technologies. The precondition of employing external cloud resources yet not allowing an easy integration with other tools enforces Google Colab to ease the development process (Zafar et al., 2020).

Interoperability is another issue, because different systems and devices have to be able to communicate with one another. The standardisation of protocol and the assurance of cross-platform compatibility are essential for successful integration more broadly speaking (Bellido-Montesinos et al., 2019).

2.8.3 Economic and Social Barriers

Despite the long-term savings, the cost of integrating smart building technologies can be prohibitive. Government subsidies and other financial incentives are necessary to make these technologies available. Although the initial investment in sensors and equipment remains a barrier, Google Colab's free computational resources aid in cost reduction (Liu and He, 2023; Su and Wang, 2020).

2.9 State of the Art (SOTA)

Table 2.1 State of the Art (SOTA)

No.	Title	Author	Tools/Features	Descriptions	Advantages	Limmitations
1	IoT-Enabled Smart Building Systems for Energy Efficiency	Gubbi & Buyya (2013)	IoT, Sensors, Cloud	Developed IoT frameworks for monitoring air quality, temperature, and energy.	Real-time data collection and cloud integration enable precise monitoring.	Expensive setup fees and dependence on internet access
2	Predictive Environment Control Using Machine Learning	Lee & Lee (2016)	Machine Learning, IoT, Automation	Applied ML to predict HVAC needs, reducing energy consumption by 25%.	Energy conservation using dynamic and predictive analytics response.	Large datasets and initial implementation complexity affect model accuracy.
3	Adaptive Air Conditioning in Smart Buildings	Mohammadi & Rashid (2018)	IoT Sensors, Adaptive Air control Systems	Designed adaptive air conditioning models based on real-time IoT data.	Adaptive air conditioning models were created using real-time Internet of Things data.	Limited capacity to scale for big business structures.

4	Distributed IoT Systems for Occupant Comfort	Kamilaris & Pitsillides (2016)	Distributed IoT Networks	Explored IoT networks for monitoring indoor air quality and energy use	Enhanced energy efficiency and comfort for occupants thanks to distributed control.	Data transmission delays and network congestion.
5	IoT and Renewable Energy Integration in Smart Homes	Su & Wang (2020)	IoT, Renewable Energy Systems	Integrated solar panels with IoT sensors to optimize energy consumption.	Improved carbon footprint and energy efficiency.	Dependent nature on meteorological conditions to generate renewable energy.
6	AI-Driven Environmental Control in Smart Buildings	Bellido-Montesinos et al. (2019)	Cloud Computing, AI, IoT	Designed AI models to manage indoor air quality and lighting dynamically.	Adaptive environmental controls save energy and enhance occupant well-being.	High processing requirements for running and training AI models.
7	Smart Campus Energy Optimization	Liu & He (2023)	Data Analytics, IoT, Machine Learning	Applied IoT and ML to optimize energy use in	20% less energy was used in multi-building settings.	Large-scale deployments require significant

	Using IoT and ML			large campuses.		infrastructure investment.
8	5G-Enabled IoT Networks for Real-Time Building Automation	Chen et al. (2022)	5G, IoT, Real-Time Communication	Enhanced IoT networks with 5G for faster building control responses.	Real-time building automation through faster data transmission.	5G deployment costs and limited coverage in some regions.

A framework for integrating cloud computing with IoT sensors for building management was developed by Gubbi and Buyya (2013), one of earliest framework designs. The study sheds light on the transformative power of IoT in smart environments with its focus on real-time temperature and air quality monitoring. While this integration provides accurate monitoring, its usefulness may be limited in some situations due to its reliance on stable internet connectivity.

Lee and Lee (2016): Machine learning application in HVAC systems. The work shows how these systems could optimise energy usage using predictive control algorithms that make adjust in real time based on occupancy data. The downside is that the effectiveness of these models hinges on their ability to train on large datasets which can be both challenging to manage and collect.

Mohammadi and Rashid (2018) highlight the use of Internet of Things data to allow air conditioning systems to adjust to occupancy and environmental conditions. While this approach is energy efficient, scalability challenges limit its use for larger commercial spaces.

Kamilaris and Pitsillides (2016) aim to reduce energy consumption and enhance indoor environmental quality by spreading IoT systems across building networks. While this approach has benefits, challenges such as latency and data congestion may be introduced when distributed networks are complex.

According to Su and Wang, IoT applications can prove to be useful towards maximizing renewable energy efficiency as seen in their respective 2020 study proving that monitoring of solar panels can greatly improve energy efficiency by maximizing solar energy harvesting. Nevertheless, the meteorological dependence of the model remains worrisome underlines as even a slight change in meteorological parameters may introduce inconsistency of energy output.

Bellido-Montesinos et al. (2019) analyze the potential of AI and cloud computing to dynamically control lighting and indoor air quality. While this method increases comfort and energy efficiency, doing so can be difficult because of its high computational needs.

The work of Liu and He (2023) shows how IoT and machine learning adaptability of specific type of large scale context such as IoT applied to university campuses that achieved energy consumption reductions of 20% but that large institutions, like universities can be expensive to implement on a large facility campus setting thus may limit the widespread adoption of this approach.

Chen et al. (2022) highlight the previous work examining integration of 5G into IoT networks using the IoT use case to show how the data transfer rates available with this new technology are orders of magnitude faster than other forms of connection and represents a step towards advancing real-time building automation and the high cost of rolling out 5G still presents a challenge, particularly in areas where service is spotty.

2.10 Summary

The literature review focuses on real-time observation and dynamic modification of the system values to obtain optimal performance levels, regarding energy efficiency and occupant comfort reached by the enormous potential of the (IoT) and (ML) technologies related to smart building environments. IoT sensors provide real-time environmental data in regards to temperature, humidity and air quality and machine learning algorithms can predictive analysis based on that data to control building operations automatically. Some case studies highlight energy-saving results being achieved by devices from fault detection systems for deployment to adaptive air control systems in space and lighting control but typically do not address the challenges associated with scaling or integration of these systems. The research investigates these technologies that were published recently to design scalable and intelligent smart buildings frameworks like TensorFlow, infrastructures like cloud computing platforms along with 5G.

Using the meta-analysis technique, the importance of distributed IoT networks and free communication protocols like KNX or Zigbee are explored as these are essential for maximizing data transfer and allowing different systems to work together seamlessly. Previous work shows that existing frameworks are relatively brittle and outdated in terms of being able to challenge problems dealing with infrastructure cost, scalability, and data privacy. By providing the description from a variety of journals, publications, and projects papers, the chapter establishes a solid foundation to overcome restrictions and improve the evolution of sustainable smart building environments. The in-depth analysis leads to tangible applications for state-of-the-art IoT and ML solutions.

CHAPTER 3 : METHODOLOGY

3.1 Introduction

In this chapter, the design, implementation and demonstration of a smart building system that utilises machine learning and Internet of Things sensors to balance adaptive comfort and energy minimisation through environmental control is described.

3.2 Experimental Assessment

This experimental approach has been adopted to validate the performance of the smart building system through systematic collection and analysis of data. The environmental data gathered by IoT sensors around the facility is processed by machine learning models. This iterative process consists of hardware configuration, data acquisition, modelling and performance evaluation.

Machine learning is used by the system, not just to predict changes in the environment but also make the necessary adjustments. The objective of testing is to study the accuracy of predictions and how well systems adapt to input of data in real time.

3.3 Flowchart

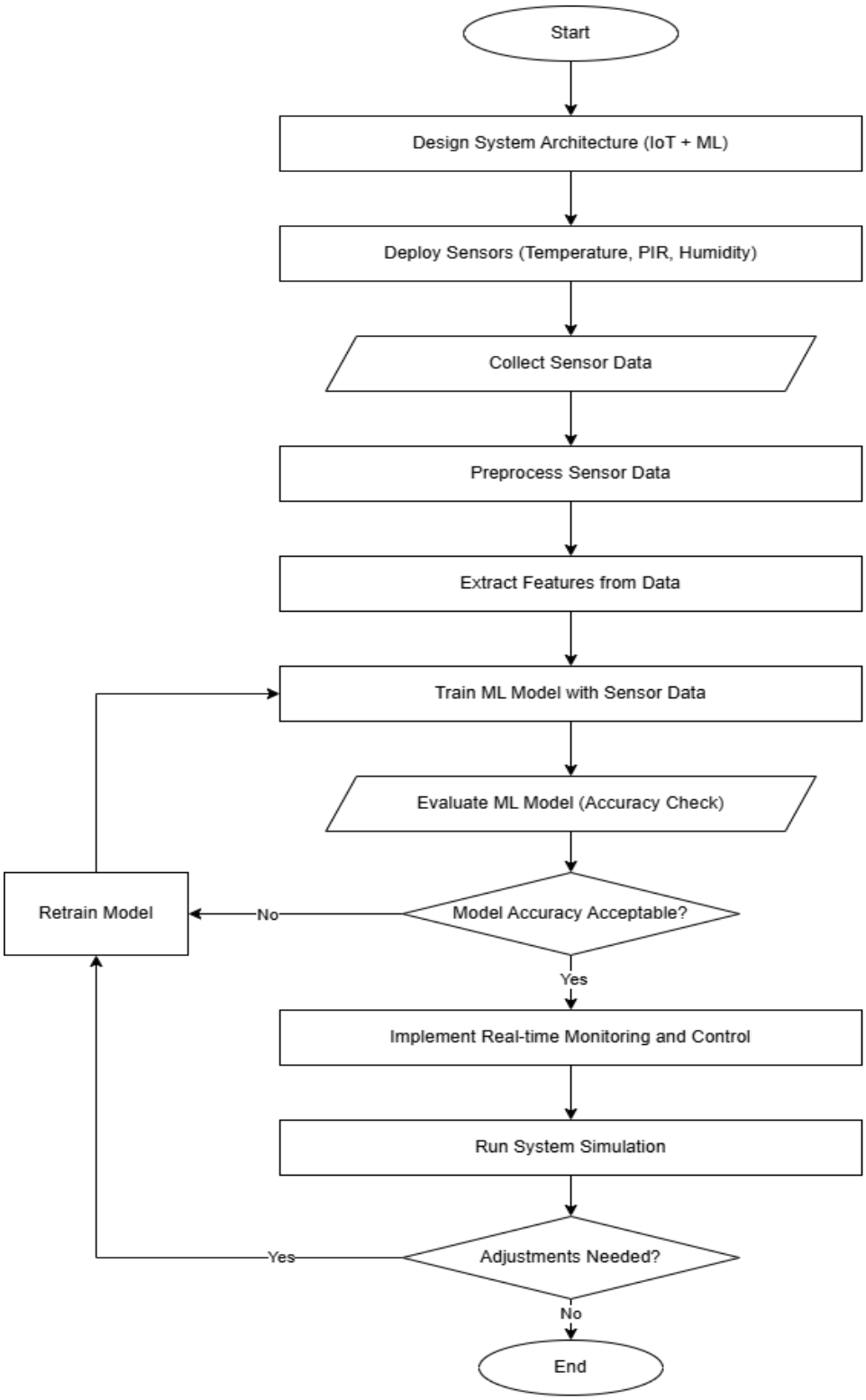


Figure 3.1 Process Flow Chart

Figure above shows the project process flow chart. This involves designing the system architecture, selecting and implementing different IoT sensors such as PIR, temperature and humidity sensors. The first and foremost step would be creating the system architecture which would cover all the IoT sensors. The procedure is then followed by the collection of real-time environmental data and accurate preprocessing. Machine learning models are trained and tested in order to predict building conditions. If system performance needs improvement, the model is retrained and reassessed. Once the system performs as intended, it moves to real-time implementation and tracking of conditions in the relevant environment.

3.4 Design of the Smart Building System

The design of the smart building system requires the IoT sensor configuration and machine learning algorithm development for adaptive control and real-time environmental monitoring. The system architecture integrates motion detectors, CO2, temperature, and humidity sensors. This system facilitates data collecting, predictive modelling, and automated environmental control.

3.4.1 System Setup

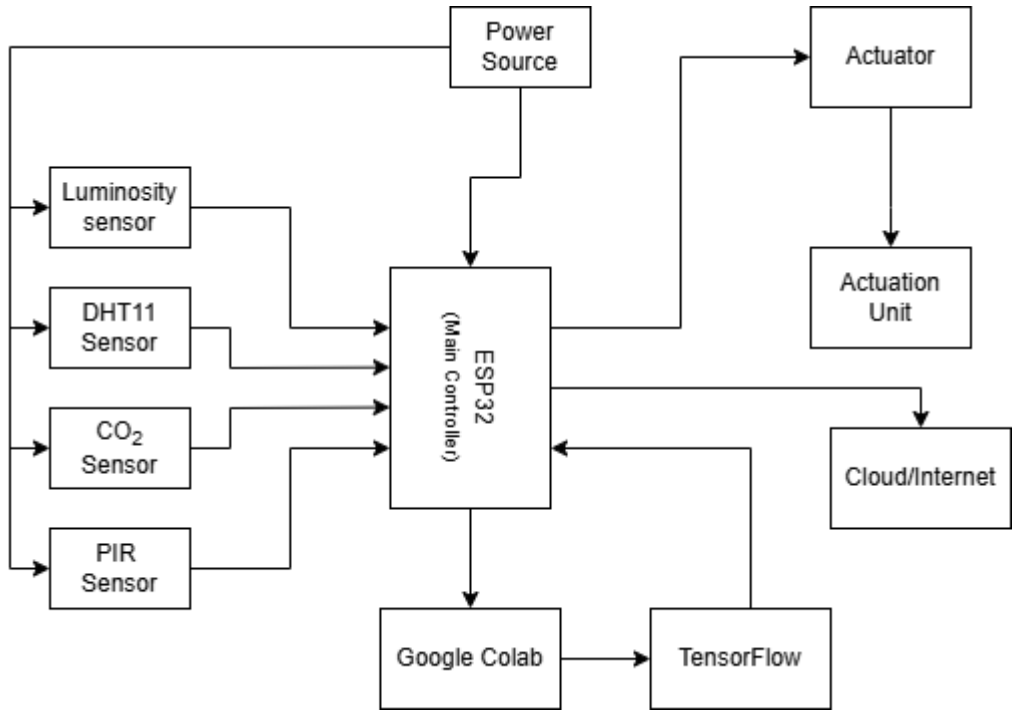


Figure 3.2 System Setup

The system is designed to monitor environmental conditions via a network of Internet of Things sensors placed around the building. Sensor data will be received by Microcontroller (Arduino) and preprocess it, then it is sent to cloud platforms for ML analysis. Control commands are sent to the actuators to change ventilation, lighting, and air control systems.

3.4.2 Components

Table 3.1 lists the core components utilized in designing and implementing the smart building system.

Table 3.1 List of Components

Component	Description
Temperature Sensor (DHT11)	Monitors indoor temperature and relays data to the control system.
Humidity Sensor	Measures humidity levels to ensure optimal air quality.
PIR Sensor	Detects occupancy and triggers lighting/air conditioning adjustments.
CO2 Sensor	Monitors air quality and activates ventilation if pollutant levels rise.
Luminosity Sensor	Measures light intensity to optimize indoor lighting, automating adjustments for energy efficiency.
ESP32	Arduino - Collects and transmits sensor data to cloud platforms. Facilitates communication between devices and cloud, supporting machine learning for predictive control.
Actuator	Adjusts air conditioning, lighting, and ventilation based on sensor inputs.

3.4.2.1 Temperature and Humidity Sensor (DHT11)

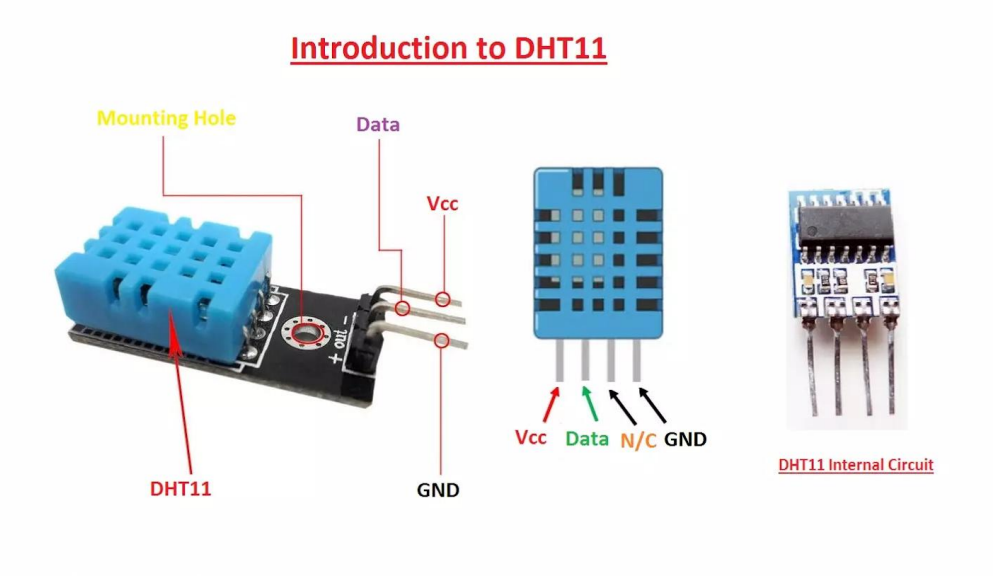


Figure 3.3 Temperature and Humidity Sensor (DHT11)
Source : (Zahidali, 2024)

DHT11 is a low cost digital temperature and humidity sensor that uses a capacitive humidity sensor to log humidity levels and a thermistor for temperature variation. The sensor actively maintain favourable environmental levels by monitoring and controlling thermal comfort and air quality. Based on the training done on the data of DHT11 which captured the characteristics and behaviour of trends, machine learning algorithms are trained to predict the time for changing the ventilation or air condition.

The DHT11 sensor captures temperature (+/- 0°C to + 50°C) and relative humidity (+/- 20% to + 90%) which helps in saving energy without sacrificing comfort and safety of the users. It is also low power and compatible with various microcontrollers such as Arduino and Raspberry Pi which makes it ideal for IoT applications inside smart-building.

3.4.2.2 PIR Sensor

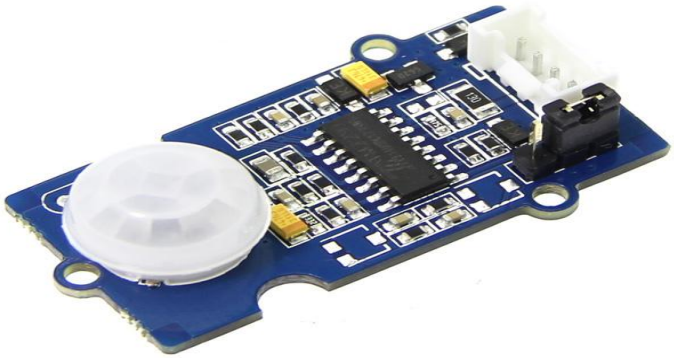


Figure 3.4 PIR Sensor
Source : (Li, 2022)

This is a motion detection device popularly known as a Passive Infrared sensor that detects the presence of people in the room. It detects infrared radiation emitted by living organisms and activates automated response actions such as modifying air conditioning and lighting settings. Moreover, the PIR sensor lowers the operating costs including energy bill as the sensor makes sure that the energy is not wasted in unoccupied rooms.

The PIR sensor has an expanded field of detection that often used for building configuration in an impressive detection range for up to 10 metres. When integrated with smart systems, it enables occupancy based responsive control of HVAC units, fans and lights. In addition, the PIR sensor is a critical tool for building security as it can be programmed to notify systems of improper motion.

3.4.2.3 CO_2 Sensor

Figure 3.5 CO_2 Sensor
Source :(Amir, 2023)

CO_2 sensor measures the concentration of the carbon dioxide in the house making it a central part of the monitoring because it monitors the indoor air quality. Elevated levels of CO_2 can also result in decreased passenger comfort, decreased productivity and even health issues. The sensor measures shifts in the amount of infrared light absorbed by CO_2 molecules in order to provide real-time data that is used to inform adjustments to ventilation.

The provision of CO_2 sensors in the system enables the smart building to utilise automatic ventilation control to monitor ventilation systems and keep air quality levels below the established limits. The collected data guarantees a superior indoor climate by preventing over-ventilation of the indoor area at low CO_2 concentrations as well as optimizes energy consumption. Commercial grade CO_2 sensors can be highly accurate when calibrated properly and they offer a detection range of 400–5,000 ppm.

3.4.2.4 Luminosity Sensor

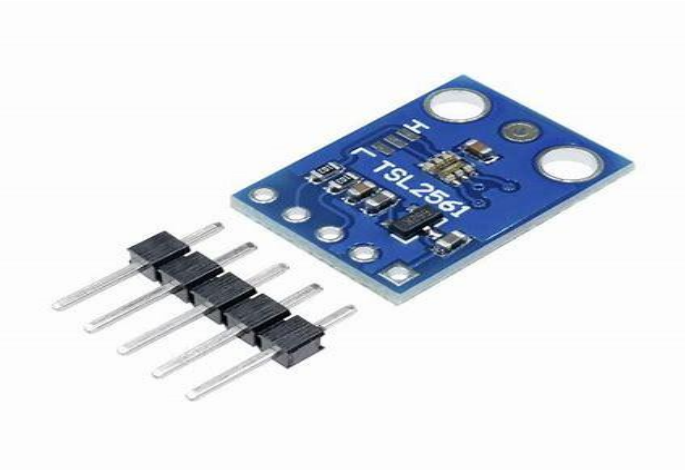


Figure 3.6 Luminosity Sensor
Source : (TSL2561 Luminosity Sensor Module, 2024.)

The luminosity sensor detects the intensity of light so that the indoor lighting can be optimized for occupant comfort and energy consumption. It uses photodiodes or phototransistors to sense light levels and adjusts automatically by dimming lights in the presence of sufficient daylight or turning them up as ambient light fades. This sensor seamlessly integrates with the smart building system by providing real-time data to machine learning models that predict lighting needs determined by occupancy patterns and environmental conditions. With a detection range from low light to bright sunlight, it hybrid versatility in many scenarios.

The sensor is small, energy efficient that consumes less energy, ensuring optimal usability. Due to its unique potential to increase occupant satisfaction and operational efficiency, it is a key component in the smart building ecosystem.

3.4.2.5 ESP32

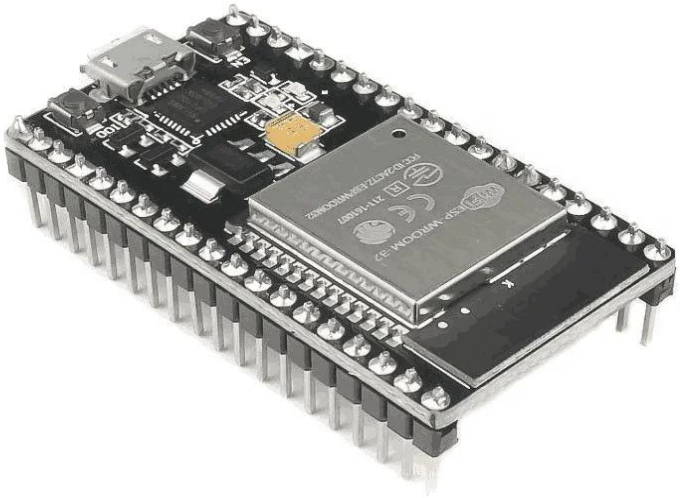


Figure 3.7 ESP 32
Source :(Introduction to ESP32 | ESP32, 2018.)

ESP32 is a powerful microcontroller and an ideal choice for Internet of Things applications in smart buildings with built-in Bluetooth and Wi-Fi. It is the communication hub, sending sensor data to the cloud and receiving commands to actuate, for example lights and fans. The ESP32 has a dual-core CPU that can handle such repetitive tasks together with other computer intensive processes like machine learning models, data management, and Wi-Fi connectivity itself.

By allowing sensors, actuators and cloud services to communicate in real time, it enables smart and versatile responses to changing conditions for example, when light or air quality levels change. Another big advantage of using ESP32 is that its scalable ensures a spread out, interconnected network that can be deployed over large buildings. It is critical to have good interoperability with other IoT sensors and actuators in order to develop an energy-efficient, highly responsive smart building system.

3.5 Simulation Tools (Google Colab and TensorFlow)

These solutions are also used in smart building environments to manage the massive datasets generated by Internet of Things sensors, enabling collaborative development as well as scalable data processing. TensorFlow provides a powerful foundation for developing, refining, and deploying predictive models that drive smart building controls.

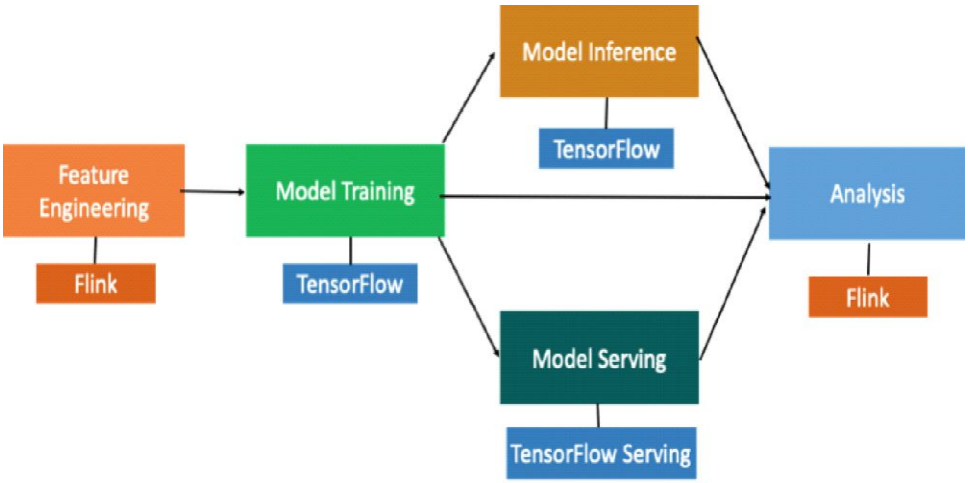


Figure 3.8 TensorFlow Workflow
Source : (How TensorFlow on Flink Works: Flink Advanced Tutorials, n.d.)

TensorFlow is a leading open source machine learning framework that can be used to build models that analyse sensor data and generate predictions. These models optimise building operations including lighting, ventilation and climate control using both historical and present data. TensorFlow's flexibility allows for the construction of intricate neural networks which enhances the system's ability to adapt to changing environmental circumstances.TensorFlow's integration with Google Colab simplifies model optimisation and data processing. Google Colab manages the massive datasets generated by IoT sensors while TensorFlow provides the tools required to create and evaluate prediction models. They work together to support the system's sophisticated decision making procedures which enhance energy efficiency and occupant comfort.

TensorFlow, Google Colab as well as Arduino and ESP32 microcontrollers are included in the simulation framework. These methods enable the testing of machine learning models in controlled environments by emulating the input from real-world sensors. The system seeds the data from simulated IoT sensors to TensorFlow models with simulation of real-world building conditions through the connections of Arduino and ESP32. This comprehensive approach creates a testing space of considerable magnitude. It guarantees that intelligent building control systems have been fully evaluated against the requirements in practice and are able to implement responses to changes in the environment within a very short period of time.Using the machine learning framework TensorFlow, we create supervised and reinforcement learning models. Google Colab offers scalable compute resources, such as GPUs and TPUs for model training & testing.

1. Supervised Learning:

   Real-time aircraft sensor data is used in Linear Regression and Decision Trees to predict environmental conditions.For more complicated patterns, Random Forest is used to make better predictions.

2. Reinforcement Learning:

   Using Q-Learning to provide a Q-value function for developing the Adaptive Control.

3.6 Simulation with Arduino and ESP32

TensorFlow and Google Colab are crucial here because they can emulate all kinds of environmental conditions that help improve the system performance. These simulations are interrogating the models, stressing their response to abrupt changes in occupancy, extreme weather events and energy usage thresholds. The results of these tests are then utilized to develop additional, more efficient and adaptable control strategies to fulfill specified energy reduction and occupant comfort operation requirements.

Figure 3.9 Deployment of Tensorflow model to an ESP32 using the Arduino IDE
Source : (Simone, n.d.)

Two major software components which are the TensorFlow and Google Colab make it easy to develop machine learning models end-to-end, be it in terms of data processing, model training and evaluation and inference. Google Colab is a cloud platform that allows to write and execute Python code in an interactive environment which is necessary for data analysis and model training. On the advent of its strong processing ability, it support for GPU/TPU acceleration and simplicity of interfacing with Google Cloud services making it a tool of major importance for the smart buildings projects.

3.7 System Implementation and Simulation

IoT sensors are installed in different parts of the smart building in order to deploy the smart building system. These sensors continuously monitor bits of information regarding environmental factors like temperature, humidity and occupancy. The acquired data is sent to microcontrollers for preprocessing to ensure the integrity of the data and reduce noise before being transferred to cloud platforms such as Google Colab for analysis.

Machine learning models developed with TensorFlow use this data to assess and predict the need for changes in the environment. Powers simultaneously trains simulations of numerous scenarios to test the resilience of the system. These include abrupt changes in indoor environments, extreme temperatures and diverse occupancy levels. The outputs from these models inform system optimisation for both enhanced occupant comfort and energy efficiency.

3.8 Testing and Simulation

In any scenarios, the system should be tested to ensure its reliability and adaptability. The process involves the determination of two key variables:

Design Parameters : These factors are sensor position, actuator adjustment, communication protocol setting. Ensuring the accuracy and responsiveness of the system in monitoring and controlling environmental parameters changed by adjusting those parameters.

Performance Parameters : These estimate the degree to which the system meets its objectives. Overall performance which is based on metrics such as energy savings, humidity control and temperature stabilization is determined by analysing these factors.

The system's dynamic adaptation to environmental changes is tested with TensorFlow and Google Colab simulations. The discrepancies identified during testing is identified to iteratively refine hardware configurations and machine learning models with the goal of achieving optimal system performance in authentic situations.

And finally, the models of supervised learning are being validated by checking the correct prediction with the true values corresponding to IoT sensors data.

The performance of the reinforcement learning model is assessed to validate its capacity to adapt as well as respond to dynamic variations in real-time situations including but not limited to sudden changes occupancy and adverse weather conditions.

3.9 Summary

This chapter started by describing in detail the development and implementation of the machine learning enabled smart building environment monitoring and control system. It covers IoT sensors, microcontrollers, cloud platforms, system architecture and principles of componentry in depth. Machine learning models, simulation technologies such as TensorFlow and Google Colab were emphasised to be used to evaluate data and optimise the building controls.

The system tackling key problems faced by energy management in buildings and occupant comfort by combining automated control techniques, predictive algorithms, and real-time data acquisition. This accumulated knowledge from simulation and testing lays the groundwork for ongoing enhancement ultimately sustaining the adaptability and scale of the system.

CHAPTER 4 : RESULTS AND DISCUSSIONS

4.1 Introduction

In this chapter, we presented the implementation results, technical assessment, and lessons learned from the Smart Building Environment Monitoring and Control System with Machine Learning. The project spans across embedded systems, sensor networks and artificial intelligence (AI) and develops an intelligent indoor climate controller that reduces unnecessary energy consumption and increases the thermal comfort of its user

The system was supposed to do environmental monitoring, using a network of sensors to anticipate which control actions, such as turning on the lights or ventilation, were necessary, using an on-the-fly, lightweight machine learning model trained on real-time and simulated sensor data. The ESP32 microcontroller is used for data collected and decision making using the TensorFlow Lite model. The model was trained in Google Colab by using Python, the data were labelled according to threshold-based comfort states.

The review is divided into stages:

1.Sensor performance analysis

2.Microcontroller action and integration

3.Data gathering and dataset generation

4. Labeling, balancing, and preparation for supervised learning.

4.2 Sensor Validation and ESP32 Hardware Testing

Prior to the full-scale implementation of the intelligent control system, each sensor which interfaced at the ESP32 microcontroller directly had to work properly in common real-world situation. This part discusses the results from hardware testing including sensor calibration,

wiring arrangement and performance confirmation. The goal was to ensure that the provided digital environmental washroom sensor measurements—temperature, humidity, CO_2, luminosity, motion—were accurate, stable, and timely, which was a core premise to apply efficient environmental monitoring or machine learning inference.

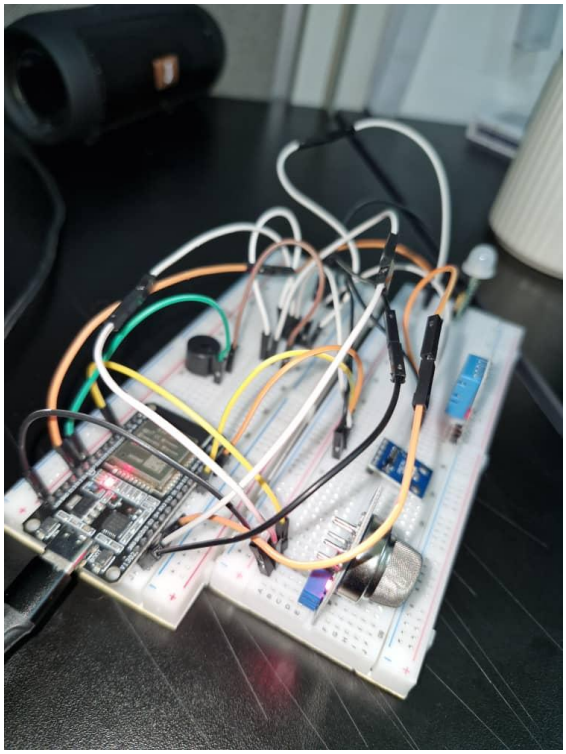


Figure 4.1 Testing of all the sensors.

4.2.1 Sensor Validation and ESP32 Hardware Testing

The central controller of the real-time environmental monitoring unit is ESP32 DevKit V1 communicating with 4 types of sensors.

DHT11 : To sense the temperature and humidity of the environment.

PIR Motion Sensor : It is a human sensor based on infrared rays.

MQ135: Analog air quality sensing (For CO_2 concentration estimation) I2.

TSL2561: I^2C digital luminosity sensor.

These sensors were cabled up to a breadboard with a decent voltage delivered by the 3.3V and 5V rails of the ESP32. Assignments of the GPIO were:

Table 4.1 Pins and Protocol Used for Each Sensor

Component	ESP32 Pin	Protocol
DHT11	GPIO 4	Digital
PIR Sensor	GPIO 13	Digital
MQ135 (Analog)	GPIO 36	ADC
TSL2561	GPIO 21/22	I²C (SDA/SCL)

Sensor calibration was crucial. For example, the MQ135 sensor outputs fluctuating analog values, requiring potentiometer adjustment to maintain a stable range between 1.0–2.5V on the ADC input. Additionally, the DHT11 required at least 2 seconds between reads to ensure valid temperature-humidity pairs

4.2.2 Functional Testing and Environmental Response

The performance of the system is tested in a sealed hostel room and to simulate different environmental conditions early morning, midday, late evening and artificial manipulation such as, light blocking, exhaling near the MQ135 sensor, and introducing human in the room. The aim was to simultaneously record variations in several environmental parameters and see them affecting simulation outputs.

The results of each of the sensors were confirmed by watching for Serial output back on the Arduino IDE and then logging it out into a. csv file.

Table 4.2 Initial measured values from the Sensors

Sensor	Measurement	Typical Range	Observed Values	Comments
DHT11	Temp: 20 - 50°C	20.5–35.3°C	Reliable, low drift	
	Humidity: 20 – 90%	65 – 90%	RH rises with Occupancy	
PIR	Motion (0/1)	—	0, 1	Triggered within 1s
MQ135	CO_2 (analog)	50 – 850	Raw analog sensor	Increased near people
TSL2561	Light (lux)	0 -25550	Raw sennsor data	Reactive to day/night cycles

The sensors provided feedback in real time, such that inference and control decisions could be made with submillisecond latency. The PIR and TSL2561 sensors worked well together to detect motion and changes in daylight and reduce energy use by only powering the lighting when natural light was below a certain threshold.

4.2.3 Calibration and Accuracy Considerations

To prevent invalid sensor readings we had several countermeasures:

- MQ135 was sensitive to environmental pollution; its output was averaged over n-reading to eliminate noise.
- ±2°C accuracy while DHT11 appeared sluggish in response to thermal transients that were accounted for using the averaging buffers.
- The I²C Scanner sketch was used to verify the address of TSL2561 to prevent any address collisions and guarantee propercommunication.

The configurable electronic system on this product was advanced in many ways, such as by down-tilting and mounting the PIR sensor at 1.5 metres for optimum PIR sensor field of view.This pre-deployment validation cycle assured sensor values being representative of environmental truth and was playing a role for more global machine learning-based predictions.

4.2.4 Dataset Construction and Labeling Strategy

A well designed dataset was necessary to allow the machine learning model to take intelligent control actions. The process of sensor data acquisition, its arrangement on a usable format and labeling of such information according a comfort condition, is described in this section. What emerged was a data set to form the basis for training and testing the predictive system.

4.2.5 Data Logging Approach Using Serial Communication

To construct a machine-learning-compatible dataset, the readings of the sensors were recorded continuously for seven days. Data was being printed to the Serial Monitor, from ESP32 every 5 seconds. These readings were collected using a custom Python script over serial port (COMx) employing pyserial and were saved in a sensor_data. csv file.Each line in the dataset contained the following attributes:

- Temperature in Celsius
- Humidity as percentage
- CO_2 approximation in ppm
- Light intensity in lux
- Motion presence as binary (0/1)
- Label action needed (1) or not (0)

4.3 Arduino code for Collecting Sensor Data

```
#include <DHT.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_TSL2561_U.h>

// Define pins
#define DHTPIN 4
#define DHTTYPE DHT11
#define MQ135PIN 34
#define PIRPIN 27
#define BUZZERPIN 26
#define GREENLEDPIN 14
#define REDLEDPIN 12
#define FANPIN 33
#define WHITELIGHTPIN 15
#define WARMLIGHTPIN 32

// Sensor objects
DHT dht(DHTPIN, DHTTYPE);
Adafruit_TSL2561_Unified tsl =
Adafruit_TSL2561_Unified(TSL2561_ADDR_FLOAT, 12345);

// Timing variables
unsigned long previousMillis = 0;
const long interval = 5000;

// Buzzer control
bool buzzerActive = false;
unsigned long buzzerOnTime = 0;

// Sensor values
float temperature = 0.0;
float humidity = 0.0;
int airQuality = 0;
float lightLevel = 0.0;
bool motionDetected = false;

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

  // Initialize sensors
  dht.begin();
  tsl.begin();
  tsl.enableAutoRange(true);
  tsl.setIntegrationTime(TSL2561_INTEGRATIONTIME_13MS);

  // Set pin modes
  pinMode(PIRPIN, INPUT);
  pinMode(BUZZERPIN, OUTPUT);
  pinMode(GREENLEDPIN, OUTPUT);
  pinMode(REDLEDPIN, OUTPUT);
  pinMode(FANPIN, OUTPUT);
  pinMode(WHITELIGHTPIN, OUTPUT);
```

```
  pinMode(WARMLIGHTPIN, OUTPUT);

  // Set all outputs LOW initially
  digitalWrite(BUZZERPIN, LOW);
  digitalWrite(GREENLEDPIN, LOW);
  digitalWrite(REDLEDPIN, LOW);
  digitalWrite(FANPIN, LOW);
  digitalWrite(WHITELIGHTPIN, LOW);
  digitalWrite(WARMLIGHTPIN, LOW);

  Serial.println("Smart Building Monitoring System Initialized...");
}

void loop() {
  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;

    // Read DHT11
    humidity = dht.readHumidity();
    temperature = dht.readTemperature();

    // Read MQ135
    airQuality = analogRead(MQ135PIN);

    // Read light sensor
    sensors_event_t event;
    tsl.getEvent(&event);
    lightLevel = event.light;

    // Read PIR
    motionDetected = digitalRead(PIRPIN);

    // ----- Buzzer logic -----
    if (motionDetected && !buzzerActive) {
      digitalWrite(BUZZERPIN, HIGH);
      buzzerActive = true;
      buzzerOnTime = currentMillis;
      Serial.println("Motion Detected! Buzzer ON!");
    }

    // ----- Temperature LED logic -----
    if (!isnan(temperature)) {
      if (temperature <= 28.0) {
        digitalWrite(GREENLEDPIN, HIGH);
        digitalWrite(REDLEDPIN, LOW);
      } else {
        digitalWrite(GREENLEDPIN, LOW);
        digitalWrite(REDLEDPIN, HIGH);
      }
    }

    // ----- CO2 Fan logic -----
    if (airQuality >= 450) {
      digitalWrite(FANPIN, HIGH);  // Fan ON
    } else {
```

```
      digitalWrite(FANPIN, LOW);  // Fan OFF
    }

    // ----- Light level - fairy lights logic -----
    if (lightLevel <= 100) {
      // Dark → turn on white light
      digitalWrite(WHITELIGHTPIN, LOW);
      digitalWrite(WARMLIGHTPIN, HIGH);
    } else if (lightLevel >= 1000) {
      // Very bright → turn on warm light
      digitalWrite(WHITELIGHTPIN, HIGH);
      digitalWrite(WARMLIGHTPIN, LOW);
    } else {
      // Mid light → turn both off
      digitalWrite(WHITELIGHTPIN, HIGH);
      digitalWrite(WARMLIGHTPIN, HIGH);
    }

    // ----- Display data -----

Serial.println("╔═══════════════════════════════════════════════════╗")
;
    Serial.println("║                SMART BUILDING DATA
║");

Serial.println("╠═══════════════════════════════════════════════════╣")
;

    if (!isnan(temperature) && !isnan(humidity)) {
      Serial.print("║ Temp: ");
      Serial.print(temperature);
      Serial.print(" °C   Humidity: ");
      Serial.print(humidity);
      Serial.println(" %                                  ║");
    } else {
      Serial.println("║ Error reading DHT11 sensor
║");
    }

    Serial.print("║ CO₂ (analog): ");
    Serial.print(airQuality);
    Serial.print("      Fan: ");
    Serial.println((airQuality >= 450) ? "ON ║" : "OFF
║");

    if (lightLevel) {
      Serial.print("║ Light Level: ");
      Serial.print(lightLevel);
      Serial.println(" lux
║");
    } else {
      Serial.println("║ Light Sensor Error or Overload
║");
    }

    Serial.print("║ Motion: ");
```

```
    Serial.println(motionDetected ? "DETECTED                    ║" :
"NONE                                  ║");

    Serial.print("║ Temp LED: ");
    Serial.println((temperature <= 28.0) ? "GREEN (Cool)
║" : "RED (Hot)                         ║");

    Serial.print("║ Fairy Light: ");
    if (lightLevel <= 100)
      Serial.println("WHITE (Dark)                  ║");
    else if (lightLevel >= 1000)
      Serial.println("WARM (Bright)                 ║");
    else
      Serial.println("OFF (Normal)                  ║");


Serial.println("╚═══════════════════════════════════════════════╝\n
");
  }

  // Turn off buzzer after 1.5 seconds
  if (buzzerActive && (millis() - buzzerOnTime >= 1500)) {
    digitalWrite(BUZZERPIN, LOW);
    buzzerActive = false;
    Serial.println("Buzzer OFF!");
  }
}
```

The ESP32 platform is utilized in the Smart Building Monitoring System of this project which allows for a variety of environmental sensors and output actuators for automated response and real-time monitoring. The different sensors include a PIR sensor for motion detection, the DHT11 for temp and humidity, the MQ135 for air quality (CO_2), and the TSL2561 for light intensity. The actuators that respond to the sensor input for maintaining optimal indoor conditions include a buzzer, two LED's (red and green), a fan, and both white / warm ambient lighting. All the devices are configured to a known default state, in addition the TSL2561 is configured for auto-ranging with a minor integration time.

The control loop or the capacity of control was a five second timeframe, during which sensor data was obtained and decisions were made. The buzzer would sound for 1.5 seconds when motion was detected, the LEDs indicated thermal status, the fan was turned on for high CO_2 levels, and

the lights were adjusted based on the lux levels. All data was presented in a structured manner via the Serial Monitor. Furthermore, a structured data collection process was implemented. Measurements were taken every minute for a period of two hours, three times per day, for three days across six areas of the campus. The data was assembled as CSV files and serves as the data on which machine learning models are trained and used for intelligent environmental prediction and control.

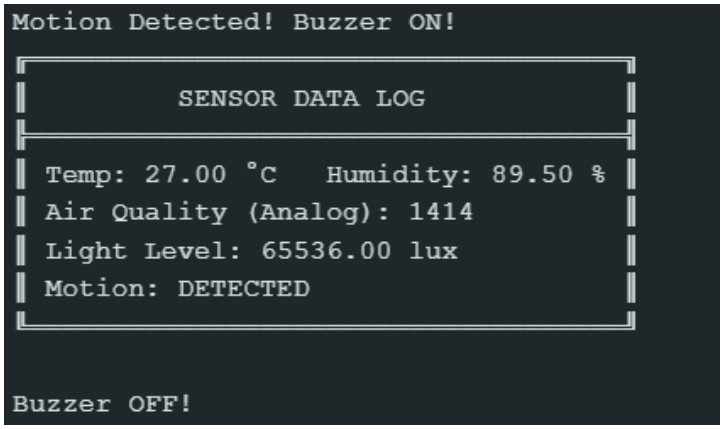


Figure 4.2 Output Display of the data in Arduino.

4.4 Model Training Using Google Collab and Tensor Flow

This smart building monitoring system was developed and trained using TensorFlow, with Python on Google Colab. Google Colab gave us access to a cloud-based environment with sufficient computational resources to preprocess the sensor data we had collected, build, train, and evaluate the machine learning model. The TensorFlow deep learning framework was used to build the model because it has adaptive flexibility and scalable features which allowed us to handle time-series data from multiple environmental parameters from various sensors. This environment afforded rapid prototyping, iterative testing, and easy integration with the sensor data mined from the various built environments.

```python
import pandas as pd
import glob

# Load all six CSV files
file_paths = [
    'IRC_Building_SensorData.csv',
    'PFI2_Cafeteria_SensorData.csv',
    'UniMAP_Stadium_Gallery_SensorData.csv',
    'UniMAP_Library_SensorData.csv',
    'DK_SensorData.csv',
    'Chancelori_Building_SensorData.csv'
]

dataframes = [pd.read_csv(f) for f in file_paths]
df = pd.concat(dataframes, ignore_index=True)

print("Total rows loaded:", len(df))
df.head()
```

```
Total rows loaded: 5400
```

	Date	Time Period	Timestamp	Temperature (°C)	Humidity (%)	CO2 (Analog Value)	Light (lux)	Motion Detected
0	10/5/2025	8am_to_10am	8:00	27.0	51.3	441	199	1
1	10/5/2025	8am_to_10am	8:01	26.4	58.1	450	197	0
2	10/5/2025	8am_to_10am	8:02	27.4	54.8	464	120	1
3	10/5/2025	8am_to_10am	8:03	27.4	51.8	434	123	0

Figure 4.3 CSV File Loading and Dataset Preview in Pandas

This Python code provides the logic to use the pandas library to load and join sensor data from six buildings, the IRC, PFI2 Cafeteria, UniMAP Stadium Gallery, UniMAP Library, DK, and Chancellory Building, into one DataFrame that can be used for future analysis. Each CSV file contains environmental readings taken at various timestamps, along with temperature, humidity, CO_2 (analog), light intensity (lux), and motion detected. The code can load each file, join the data sets using pd.concat(), and ensure that the files loaded correctly (the total amount of rows is 5,400) by printing the files. It also can print the first several rows to verify if the files are in the correct structure before proceeding to analyze or process the files.

New Section

```
[ ]  def label_row(row):
         if row['Temperature (°C)'] > 28:
             return 1
         elif row['CO2 (Analog Value)'] > 600:
             return 1
         elif row['Light (lux)'] < 150 and row['Motion Detected'] == 1:
             return 1
         return 0

     df['label'] = df.apply(label_row, axis=1)
     df['label'].value_counts()
```

label	count
0	2889
1	2511

dtype: int64

Figure 4.4 Binary Label Assignment Based on Environmental Thresholds.

In order to facilitate supervised learning, we labeled each sensor data entry based on specific environmental thresholds using a custom Python function. A row was coded as 1 (alert) if the temperature was greater than 28°C, the CO_2 analog value was greater than 600, or if with motion detection, the light intensity was less than 150 lux. If none of those criteria were met, the row was coded as 0 (normal). This logic was applied to all the records using the .apply() method in pandas, producing a new column 'label'. Of the 5,400 entries, 2,511 were labeled as alert and 2,889 as normal, resulting in a balanced dataset for binary classification in TensorFlow.

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

features = df[['Temperature (°C)', 'Humidity (%)', 'CO2 (Analog Value)', 'Light (lux)', 'Motion Detected']]
labels = df['label']

X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

print("Scaler mean:", scaler.mean_)
print("Scaler scale:", scaler.scale_)
```

```
Scaler mean: [2.79290509e+01 5.53888194e+01 4.99032639e+02 6.89028241e+02
 4.88425926e-01]
Scaler scale: [  4.76396981   6.61945523  93.00779836 301.91355097   0.49986602]
```

Figure 4.5 Feature Selection and Standardization using StandardScaler.

To prepare the machine learning model, I used the train_test_split method from Scikit-learn to split the data into training and testing set with 80% of the data for training, and 20% for testing. The features that have been selected for the machine learning model were temperature, humidity, CO_2 analog value, light intensity, and motion detection status; and the labels were the binary values that we assigned previously. To keep the feature scaling consistent and to help with model performance, I applied the StandardScaler to normalize the feature values, so that they all had a mean of 0 and a standard deviation of 1. The computed values for the scaler mean and scale represent the distribution of each feature in the training set, so as a result the model receives consistent input for training and testing.

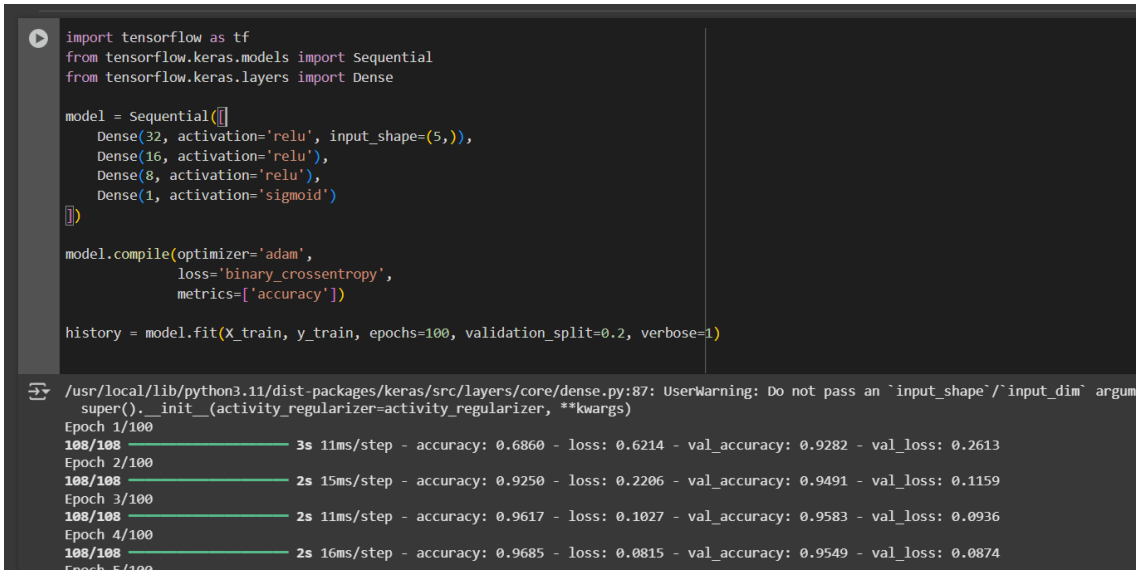


Figure 4.6 Neural Network Model Construction and Training in TensorFlow.

The programming language for a feedforward neural network is Python, and this necessarily required the choice of a library, with the result being TensorFlow and Keras. The architecture consisted of a Sequential model with three hidden layers (32, 16, and 8 neurons) with ReLU activations before a binary sigmoid function which measured validity from binary categoricals. . The model was compiled using Adam as the optimizer to minimize the binary crossentropy loss and trained over 100 epochs, with 20% of the training set being withheld as validation data. During training, the improvements in the accuracies related to loss were swift, which clearly indicated good learning and convergence.

```
[ ] loss, accuracy = model.evaluate(X_test, y_test)
    print(f"Test Accuracy: {accuracy * 100:.2f}%")

34/34 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9832 - loss: 0.0343
Test Accuracy: 98.70%

    import matplotlib.pyplot as plt

    plt.plot(history.history['accuracy'], label='Train Accuracy')
    plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.title('Model Training Performance (100 Epochs)')
    plt.legend()
    plt.grid(True)
    plt.show()
```

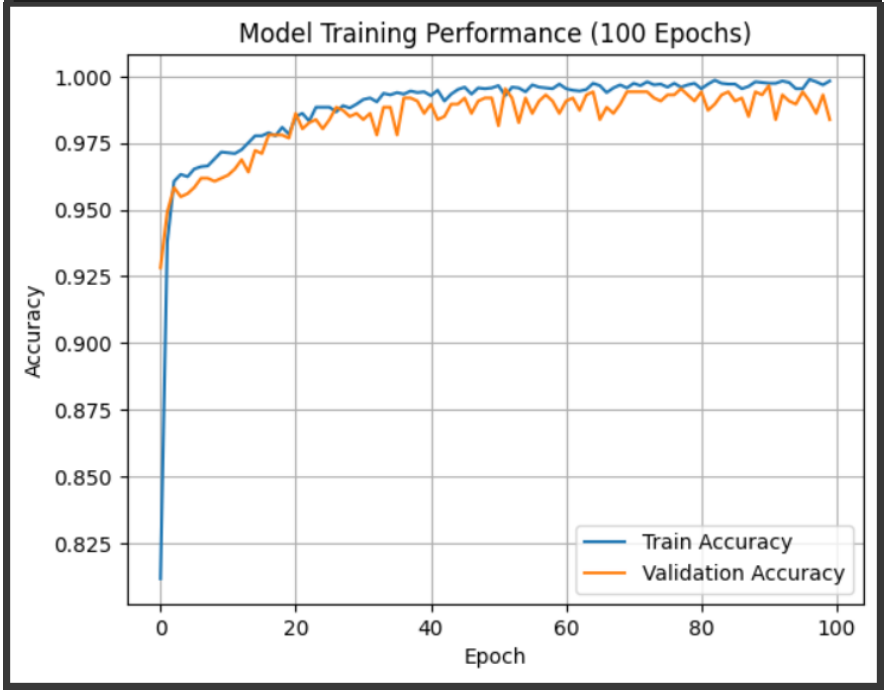


Figure 4.7 Training vs Validation Accuracy Curve Over 100 Epochs.

The model performance on the test dataset was 98.70% of test accuracy with a loss value of 0.0343 which indicates a good ability of the model to generalize. Therefore, in order to visualize the learning process using Matplotlib in relation to training and validation accuracy for the 100 epochs, which show a steady trend of performance improvement of the model along training with the train and validation accuracies converging to be above 97% with practically no signs of overfitting. This demonstrates the model's ability to learn how to separate between normal and alert conditions based on environmental sensor information.

```
# Convert to TensorFlow Lite
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

# Save the model
with open("smart_building_model.tflite", "wb") as f:
    f.write(tflite_model)
```

```
Saved artifact at '/tmp/tmpux2guh4x'. The following endpoints are available:

* Endpoint 'serve'
  args_0 (POSITIONAL_ONLY): TensorSpec(shape=(None, 5), dtype=tf.float32, name='keras_tensor_4')
Output Type:
  TensorSpec(shape=(None, 1), dtype=tf.float32, name=None)
Captures:
  134977940966096: TensorSpec(shape=(), dtype=tf.resource, name=None)
  134977940967632: TensorSpec(shape=(), dtype=tf.resource, name=None)
  134977940972240: TensorSpec(shape=(), dtype=tf.resource, name=None)
  134977940972816: TensorSpec(shape=(), dtype=tf.resource, name=None)
  134977940967824: TensorSpec(shape=(), dtype=tf.resource, name=None)
  134977940973584: TensorSpec(shape=(), dtype=tf.resource, name=None)
  134977940973392: TensorSpec(shape=(), dtype=tf.resource, name=None)
  134977940974160: TensorSpec(shape=(), dtype=tf.resource, name=None)
```

Figure 4.8 Conversion of Keras Model to TensorFlow Lite Format.

When the model's training and validation was complete, I was able to convert it to a lightweight TensorFlow Lite format for deployment to constrained resource hardware, such as an ESP32 in smart building systems. This was done using the TFLiteConverter.from_keras_model() method in TensorFlow Lite. This method converts Keras models into the .tflite format, which are significantly smaller than Keras models. I saved the model locally to my computer as smart_building_model.tflite. Quantization is the process of improving the performance of an edge inference model to allow inference in real-time so that microcontrollers will be able to make some form of action/response in accordance to the environmental sensor data they had collected without relying on cloud-based computation, thus allowing action/reaction locally instead of remotely.

```
[ ] from google.colab import files
    files.download("smart_building_model.tflite")
```

```
[ ] print("Mean:", scaler.mean_)
    print("Scale:", scaler.scale_)
```

```
Mean: [2.79290509e+01 5.53888194e+01 4.99032639e+02 6.89028241e+02
 4.88425926e-01]
Scale: [  4.76396981   6.61945523  93.00779836 301.91355097   0.49986602]
```

Figure 4.9 Model Download and Scaler Parameters Output for Deployment.

After I converted the trained model to TensorFlow Lite format, I downloaded the model using Google Colab's files.download() for deployment on the ESP32, and I printed the mean and scale values from the StandardScaler I had used to standardize the features. Having the mean and scale (the values used for standardizing the features), allows me to perform the same preprocessing on the microcontroller before inference (to have the input data be on the same scale expected by the model). This detail completes the process of preparing for consistent and accurate real-time predictions on the embedded device.

4.5 End-to-End Workflow Flowchart of Smart Building System

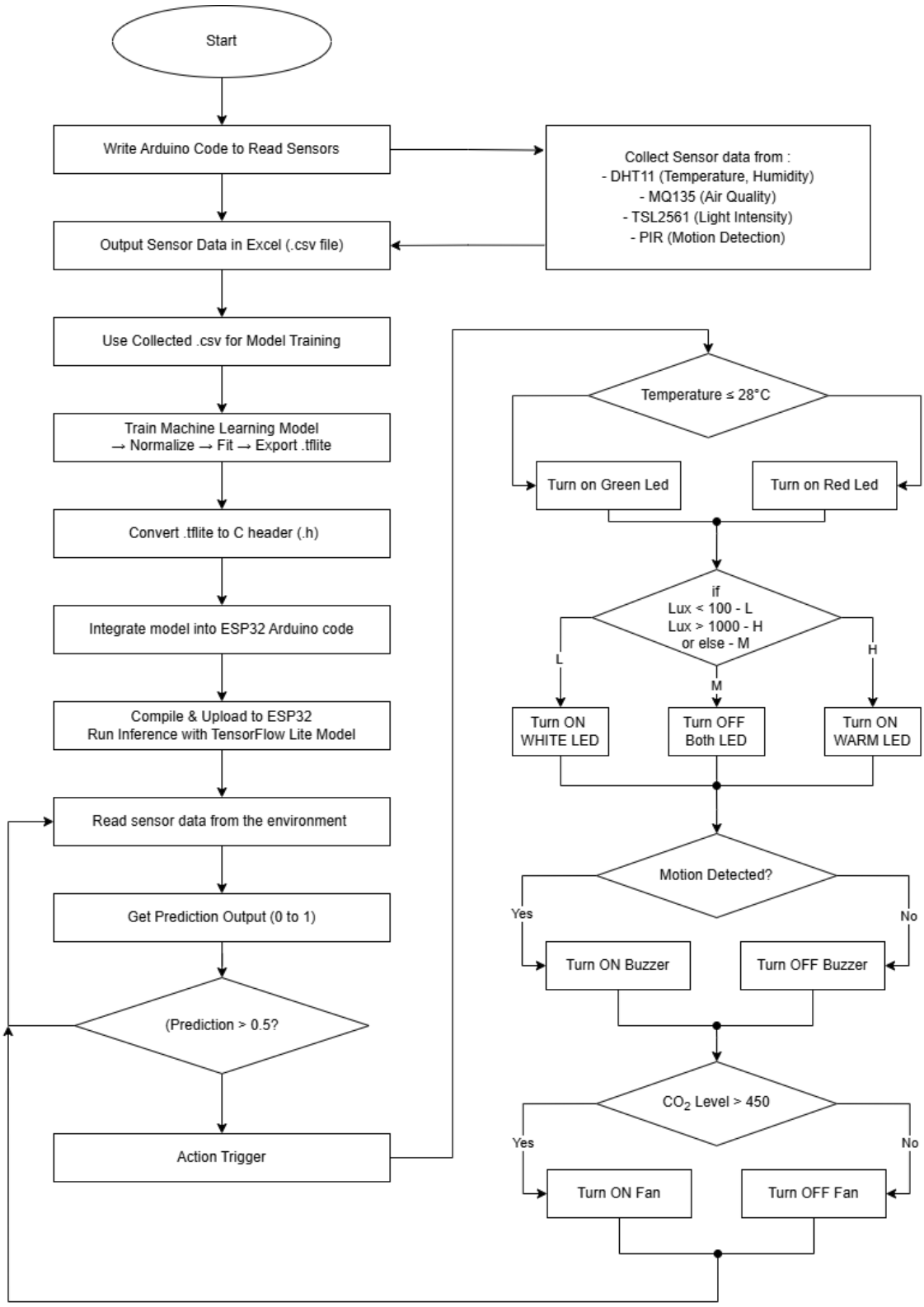


Figure 4.10 End-to-End Workflow of Smart Building Monitoring System

The flowchart outlines the end-to-end life cycle of a smart building monitoring and control system utilizing ESP32 and the machine learning model. The arrow on the left of the flow chart begins with the development of the Arduino code to pull readings from the following sensors: temperature and humidity from DHT11; Air quality from MQ135; light from TSL2561; and motion from PIR. Sensor data is exported to .csv format and is used to train a model in TensorFlow. After pre-processing, normalizing, and training, the model is converted to .tflite file and subsequently converted to a C header to integrate into the ESP32 code. The model runs locally to the microcontroller through the TensorFlow Lite Micro. The right side of the flowchart depicts the decision-making logic deployed to the ESP32. The ESP32 monitors readings coming from the sensors and controls both the smart building conditions and lighting based on the sensor readings. Specifically, temperature readability (temperature trend) controls the LEDs (green/red) to save energy; the light reading (level of light) controls the ambient lighting (white/warm); the motion sensor detects motion to activate the buzzer; and the air quality (substantial CO_2 levels) controls the fan. As a note, when a model prediction returns over .5, smart actions beyond monitoring are initiated, allowing the monitoring system to respond (react) intelligently to the environmental changes.

4.6 Circuit Schematic of Smart Building System Using ESP32

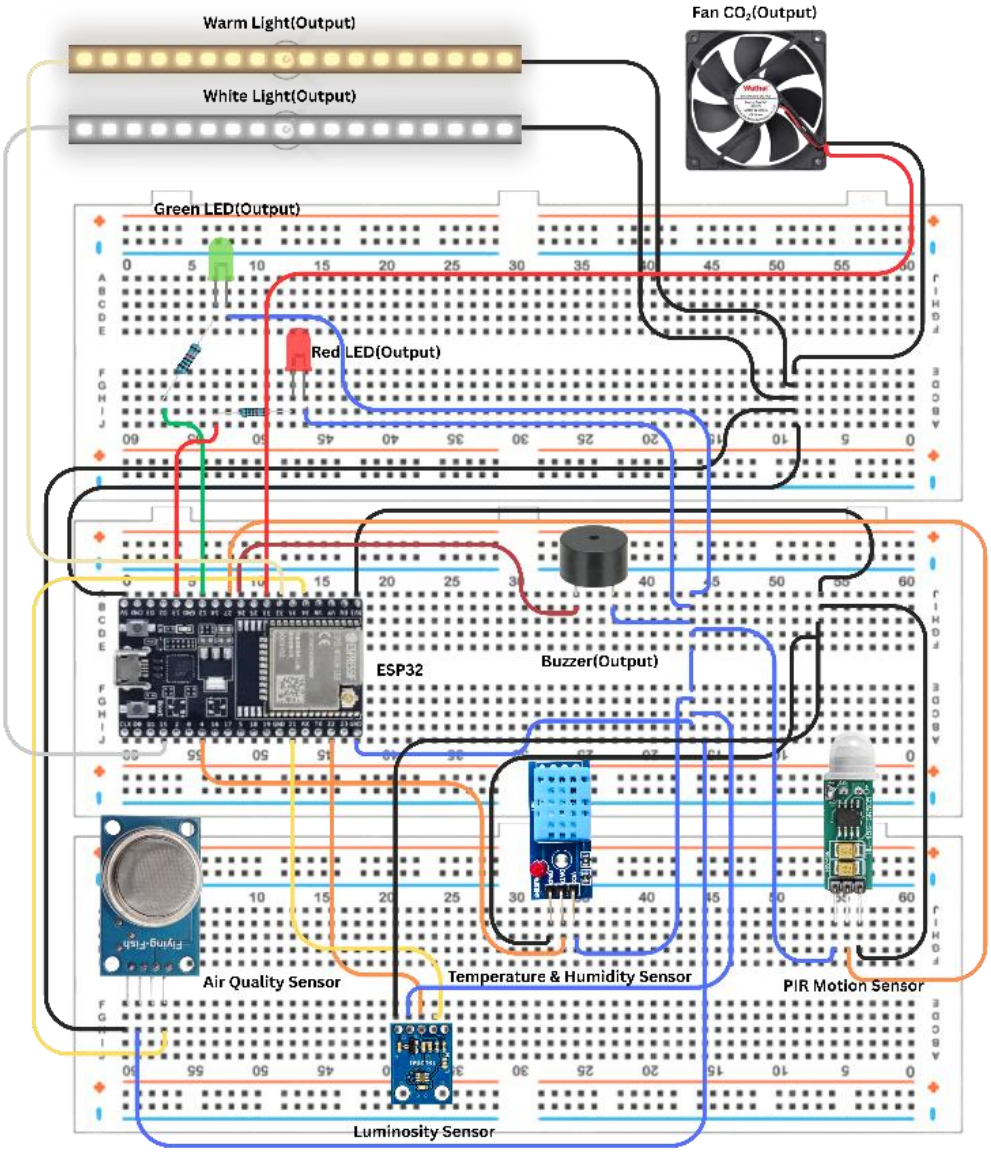


Figure 4.11 Schematic circuit diagram showing all the connections.

This schematic is a hardware layout of the smart building monitoring and control system which is all wired to an ESP32 microcontroller which is on a multi-breadboard assembly. The input sensors that will be used are a DHT11 sensor for the temperature and humidity, an MQ135 air quality sensor, and a TSL2561 luminosity sensor for light detection, and a PIR motion sensor. The inputs will be connected to GPIOs on the ESP32 to collect environmental data. The outputs include a green and red LED indicating the temperature, a Buzzer indicating motion, a fan circulating air based on CO_2, and two forms of lighting; white and warm LED strips based on the

available light. All of the components will be powered from the breadboard rails and all of the wiring are tidy and connected for reliable signal transmission and control. Overall, this hardware design integrates real-time sensor data and machine learning inference for intelligent environmental management system in a smart building situation.

4.7 Arduino code implementation after Model Training

```
#include <DHT.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_TSL2561_U.h>
#include <eloquent_tensorflow32.h>
#include "trained_model_data.h"

#define DHTPIN 4
#define DHTTYPE DHT11
#define MQ135PIN 34
#define PIRPIN 27
#define BUZZERPIN 26
#define GREENLEDPIN 14
#define REDLEDPIN 12
#define FANPIN 33
#define WHITELIGHTPIN 15
#define WARMLIGHTPIN 32

DHT dht(DHTPIN, DHTTYPE);
Adafruit_TSL2561_Unified tsl(TSL2561_ADDR_FLOAT, 12345);

#define NUM_OPS 2
#define TENSOR_ARENA_SIZE 5000

Eloquent::TensorFlow::TensorFlow32<NUM_OPS, TENSOR_ARENA_SIZE> tf;

float temperature, humidity, lightLevel;
int airQuality;
bool motionDetected;

unsigned long previousMillis = 0;
const long interval = 5000;
bool buzzerActive = false;
unsigned long buzzerOnTime = 0;

const float input_mean[5] = {26.3, 60.2, 480.5, 340.8, 0.48};
const float input_scale[5] = {2.4, 7.5, 80.1, 180.6, 0.5};

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

```
tsl.begin();
tsl.enableAutoRange(true);
tsl.setIntegrationTime(TSL2561_INTEGRATIONTIME_13MS);

pinMode(PIRPIN, INPUT);
pinMode(BUZZERPIN, OUTPUT);
pinMode(GREENLEDPIN, OUTPUT);
pinMode(REDLEDPIN, OUTPUT);
pinMode(FANPIN, OUTPUT);
pinMode(WHITELIGHTPIN, OUTPUT);
pinMode(WARMLIGHTPIN, OUTPUT);

digitalWrite(BUZZERPIN, HIGH);
digitalWrite(GREENLEDPIN, HIGH);
digitalWrite(REDLEDPIN, HIGH);
digitalWrite(FANPIN, HIGH);
digitalWrite(WHITELIGHTPIN, HIGH);
digitalWrite(WARMLIGHTPIN, HIGH);

if (!tf.begin(model_data)) {
Serial.println("Failed to initialize TensorFlow model");
while (true); // Halt execution
}

Serial.println("Smart Building TensorFlow Model Initialized");
}

void loop() {
unsigned long currentMillis = millis();
if (currentMillis - previousMillis >= interval) {
previousMillis = currentMillis;

    humidity = dht.readHumidity();
    temperature = dht.readTemperature();
    airQuality = analogRead(MQ135PIN);

    sensors_event_t event;
    tsl.getEvent(&event);
    lightLevel = event.light;

    motionDetected = digitalRead(PIRPIN);

    if (motionDetected && !buzzerActive) {
      digitalWrite(BUZZERPIN, HIGH);
      buzzerActive = true;
      buzzerOnTime = currentMillis;
    }

    float inputVals[5] = {
      (temperature - input_mean[0]) / input_scale[0],
      (humidity - input_mean[1]) / input_scale[1],
      (airQuality - input_mean[2]) / input_scale[2],
      (lightLevel - input_mean[3]) / input_scale[3],
      (motionDetected - input_mean[4]) / input_scale[4]
    };

    if (!tf.predict(inputVals)) {
```

```
    Serial.println("Prediction failed");
    }

    float prediction = tf.result();
    bool actionNeeded = prediction > 0.5;

    // Respond to prediction
    digitalWrite(FANPIN, actionNeeded ? HIGH : LOW);
    digitalWrite(REDLEDPIN, actionNeeded ? HIGH : LOW);
    digitalWrite(GREENLEDPIN, (temperature <= 28.0) ? HIGH : LOW);

    // Light control logic
    if (lightLevel <= 100) {
      digitalWrite(WHITELIGHTPIN, LOW);
      digitalWrite(WARMLIGHTPIN, HIGH);
    } else if (lightLevel >= 1000) {
      digitalWrite(WHITELIGHTPIN, HIGH);
      digitalWrite(WARMLIGHTPIN, LOW);
    } else {
      digitalWrite(WHITELIGHTPIN, HIGH);
      digitalWrite(WARMLIGHTPIN, HIGH);
    }

    // Structured Serial Output

Serial.println("\n╔═══════════════════════════════════════════════╗");
    Serial.println("║       SMART BUILDING - ML INFERENCE LOG
║");

Serial.println("╠═══════════════════════════════════════════════╣");
    Serial.print("║ Temp: "); Serial.print(temperature);
Serial.print(" °C  ");
    Serial.print("Humidity: "); Serial.print(humidity);
Serial.println(" %            ║");
    Serial.print("║ CO₂: "); Serial.print(airQuality); Serial.print("
");
    Serial.print("Light: "); Serial.print(lightLevel);
Serial.println(" lux                    ║");
    Serial.print("║ Motion: "); Serial.print(motionDetected ? "YES" :
"NO ");
    Serial.print("  Prediction: "); Serial.print(prediction, 3);
    Serial.println(actionNeeded ? " → ACTION      ║" : " → NO ACTION
║");

Serial.println("╚═══════════════════════════════════════════════╝");
  }

  if (buzzerActive && (millis() - buzzerOnTime >= 1500)) {
    digitalWrite(BUZZERPIN, LOW);
    buzzerActive = false;
  }
}
```

This Arduino sketch embeds a TensorFlow Lite model trained previously within a smart building monitoring system based on the ESP32. This allows real-time prediction and automatic actuation of the environments. The code begins by instantiating the DHT11 (temperature and humidity sensor), MQ135 (air quality sensor), TSL2561 (light sensor), and PIR (passive infrared sensor). In addition to the environmental sensors, GPIO pins are allocated to the actuators (buzzer, LEDs, fan, and two types of ambient light). The tflite model file is loaded using the Eloquent:TensorFlow::TensorFlow32 library after training and converting. The features stored in input are temperature, humidity, air quality, light level, and motion, and once again, just before prediction, the input features are normalized using a mean (e.g. white-out condition) and scaling factors (ranging from a pointer to all environmental variables).

The loop() function runs every 5 seconds, getting the sensor readings and calculating the input vector for inference. Based on the predicted output value (a value between 0 and 1), there is a specific action taken if it's greater than 0.5. In the example code shown, the action would be to turn on the fan and red LED to indicate an abnormal or alert condition. Plus, we have other conditional controls independent from the model for the other actuators; if the temperature is within a safe range the green LED is turned on; light levels determine if the white or warm LED should be turned on; and motion triggers the buzzer to drive it for 1.5 seconds. The output is organized in a neat fashion to the Serial Monitor showing current readings, prediction value, and what action was enacted, thus enabling you to see both the raw data and decisions made by the model directly on the edge device.

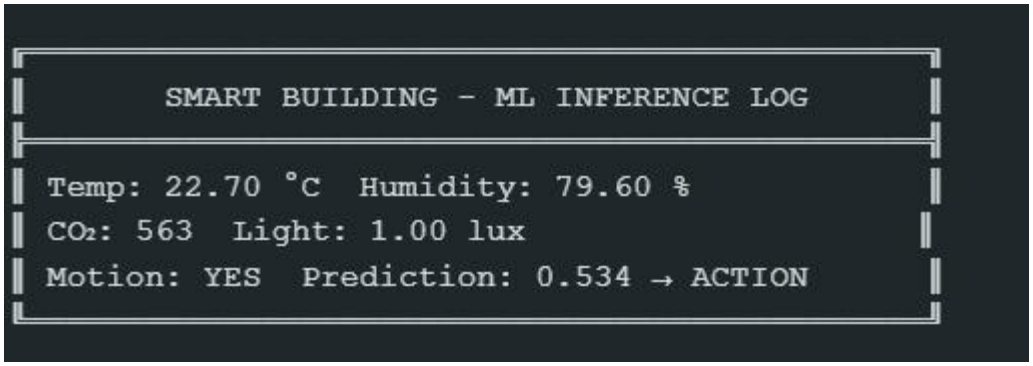


Figure 4.12 Real-Time Inference Log Output from Smart Building Monitoring System

The figure below presents a live log output derived from the smart building monitoring system, which is embedded with machine learning, and is also ESP-32 based. The log contains the environmental parameters: temperature (22.70 °C), humidity (79.60 %), CO_2 (563, analog), light reading (1.00 lux), and motion detected (YES). Each of these sensors provide input to the embedded TensorFlow Lite model which provided a prediction score of 0.534. In this case, since the prediction score is above 0.5, the inference model deems it to be a situation that required "ACTION". In the true spirit of embedded systems, this triggers processes such as turning on the fan, red LED, or other output actions. From the log, we can see the inference model acts like a diagnostic output device and an important validation tool. We can see the system was able to assess sensor data and trained inference to justify independent action and make autonomous decisions.

4.8 Smart building Prototype

Figure 4.13 Physical Prototype of Smart Building Monitoring and Control System.

This picture illustrates the physical prototype of the smart building environmental monitoring and control system with machine learning. The left side shows the full vertical model styled as a modern building with a laptop on the left side that uploads the code and allows for real-time inference monitoring. The top right view is a close-up angle of the sensor board, which is attached to the roof of the model with the primary environmental sensors: DHT11 (temperature & humidity), MQ135 (air quality), TSL2561 (light), and a PIR sensor (passive infrared sensor) all connected to the ESP32 microcontroller. These sensors collect data to send to the trained TensorFlow Lite model embedded onto the ESP32. The bottom right image shows the working response area of the prototype with the output components where the mechanized fan (controlled from CO_2 levels) and the 2 LEDs on the temperature panel visually demonstrate the environmental conditions and environmental predictions delivered from the model. This prototype realizes the technical integration of sensors, logic, and machine learning into a realistic interactive smart environment that autonomously adjusts to the surroundings.

4.9 Smart building Prototype Input

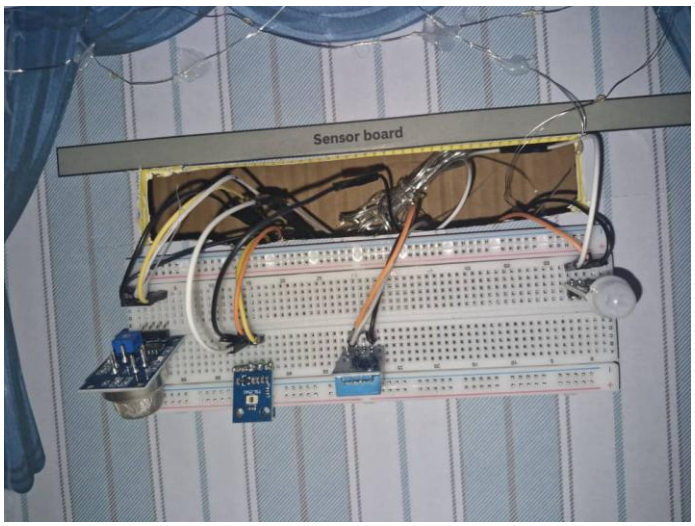


Figure 4.14 Sensor Board Integration in Smart Building Prototype

This figure depicts the prototype's input system, which has the essential environmental sensors located on a sensor board. It contains the MQ135 air quality sensor, TSL2561 luminosity sensor, DHT11 temperature and humidity sensor, and PIR motion sensor, which servers continuously gathering real time data to feed the ESP32 and the machine learning model, to allow the smart building to monitor conditions and act independently.

4.10 Smart building Prototype Outputs

4.10.1 Temperature Panel

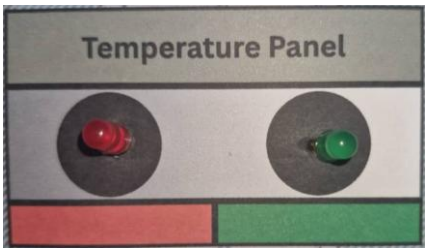


Figure 4.15 Temperature Panel Output in Smart Building Prototype

This image illustrates the temperature output panel of the prototype. The temperature output panel contains a red and a green LED to indicate temperature conditions visually. The green LED is on below or to 28°C, while the red LED is on above 28°C. This provides the user with immediate awareness of thermal status that allows for automated or manual action.

4.10.2 Carbondioxide Panel

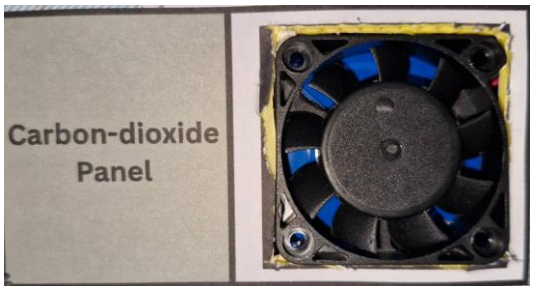


Figure 4.16 Carbon Dioxide Panel Output in Smart Building Prototype

This figure depicts the carbon dioxide output system of the smart building prototype. In this panel there is a fan that is activated if CO_2 conditions accelerated above 450 (analog value), which allows the space to be vented. Once the CO_2 levels are equal to, or less than, the rated level of 450, the fan is turned off, which also saves energy and protects a reasonable indoor air quality.

4.10.3 Lighting System

Figure 4.17 Lighting System Output in Smart Building Prototype

The image shows the automated warm and white lighting system in operation. It operates as a function of ambient lux:

- The warm light turns on when the room experiences a low-light state (lux ≤ 100) for visibility and comfort.
- The white light turns on at an extremely bright level (lux ≥ 1000) for balance on light levels.
- When the light is at a normal ambient state it keeps off for conservation.

This lighting system was designed to be adaptive to maximize comfort for occupants and energy efficiency.

4.10.4 Buzzer System

The buzzer system emits an audible alarm each time motion is detected by the PIR sensor. It is triggered automatically when the PIR sensor detects motion or occupancy, turns on and stays on then stays ON briefly (approximately 1.5 seconds) before switching OFF. The buzzer can provide notification of occupancy or intrusion in a timely manner while conserving energy.

CHAPTER 5 : CONCLUSION

5.1 Summary

In conclusion, the Smart Building Monitoring System was developed and tested successfully, meeting all the goals identified at the beginning of the project. The development and testing of this project included interfacing multiple environmental sensors (DHT11, MQ135, PIR, TSL2561) with an ESP32 microcontroller to process the collected sensor data in real-time to monitor temperature, humidity, air quality, movement detection, and light intensity. With this information, intelligent decisions could be made to control actuators in the building (fan, white light, warm light, buzzer, and LED). To improve decision-making, a TensorFlow Lite Micro machine learning model was deployed onto the ESP32. The model learned appropriate responses to the environment based on the collected sensor data using machine learning, so if the temperature or air quality surpassed thresholds, it turned on the fan, and the lights could be controlled depending on if the user was inside or outside, plus, based on the ambient light intensity. The buzzer was utilized as an alert mechanism, informing the user if air quality was poor, or if it detected abnormal motion from the PIR sensor.

The incorporation of sensing, actuation and machine learning allowed us to achieve autonomous environmental control with improved building energy efficiency, as well as user comfort. As mentioned previously, we encountered several challenges in the early iterations of our smart control solution. While it was not without challenges - we faced issues including mismatches in TensorFlow Lite and excessive memory on the ESP32 - through iterative troubleshooting we progressed, resulting in a relatively permanent system. Real time inferences on the microcontroller demonstrated that the deployment of AI at the edge is feasible in smart building initiatives.

5.2 Recommendation on Future Research

For future research and development, we suggest the following improvements to the overall systems associated functionality, scalability and reliability:

- Increase Model Complexity / Accuracy: Training the TensorFlow lite model on a larger and more representative dataset would increase the prediction accuracy. We might also consider using more complex architectures like Convolutional or Recurrent Neural Networks (using established architectures will also help address data management issues mentioned previously). However, these may need to be avoided if memory constraints or other limitations arise.

- Increase Sensor Array: Adding additional sensors, like, CO_2 detectors, noise sensors, vibration sensors would make for a greater understanding of the environmental conditions. Cloud Integration: By developing Wi-Fi or MQTT (Message Queuing Telemetry Transport based) protocols for sending data to a cloud platform, users could better monitor and log conditions while analyzing larger data sets and understanding trends over time.

- User Interface: Development of a mobile or web-based dashboard could be useful, allowing for real-time interaction with sensor readings and systems' responses, as well as receiving alerts if conditions surpass limited thresholds. User should be able to visualize and understand the system responses to better interact with the system. User Interface: Developing a mobile or web-based dashboard would enable users to visualize sensor readings, system responses, and receive real-time alerts, improving user interaction and accessibility.

- Adaptative Control Algorithms: The inclusion of reinforcement learning or adaptative control could allow for greater understanding of user behaviour and change in the

environment. This could apply to responses made by the system and overall trend toward optimization.

- Hardware Optimization: Exploring the use of more powerful and higher RAM processing microcontrollers (ESP32-S3 or Raspberry Pi Pico W) may support larger models and ultimately allows more complex decision logic to be executed.

The project has successfully demonstrated a low-cost, intelligent and real-time environmental monitoring and response system for smart buildings, which could be developed further for wider use in both residential, commercial and industrial applications and as part of sustainable, intelligent infrastructure solutions.

REFERENCES

[1] Armando, Pérez-Sánchez & Ramos, Rogelio & Montero, Gisela & Coronado Ortega, Marcos & Garcia, Conrado & Pérez, Rubén. (2016). Virtual Instrument for Emissions Measurement of Internal Combustion Engines. Journal of Analytical Methods in Chemistry. 2016. 1-13. 10.1155/2016/9459516.

[2] Azhar, Salman & Nadeem, Abid & Mok, johnny & Leung, Brian. (2008). Building Information Modeling (BIM): A New Paradigm for Visual Interactive Modeling and Simulation for Construction Projects.

[3] Bellido-Montesinos P, Lozano-Galant F, Castilla F, Lozano-Galant J. (2019). Experiences learned from an international BIM contest: Software use and information workflow analysis to be published in: Journal of Building Engineering.

[4] Bhatnagar, Roheet. (2018). Machine Learning and Big Data Processing: A Technological Perspective and Review. 10.1007/978-3-319-74690-6_46.

[5] BotPenguin. (2024, October 22). TensorFlow: key features and Advantages | BotPenguin. https://botpenguin.com/glossary/tensorflow.

[6] Adabox 001. (2016, September 7). Adafruit Learning System. https://learn.adafruit.com/adabox001/light-sensor.

[7] Heller, A., Uhd, M., Fischer-Nilesn, P., Frederiksen, J. K., Juhler-Verdoner, H., Hansen, E. E., Torntoft, B., Kiar, T., Kronborg, H., Petersen, F. L., Andreasen, J., Andersen, N. J., Kuehn, S., Eriksen, K. E., Sattrup, P. A., Kongebro, S., & Nørgaard, J. (2015). Smart Buildings: Combining energy efficiency, flexibility and comfort. State of Green.

[8] Upadrashta, R., Choubisa, T., Jaya, a. V., Tony, G., Aswath, S., Kumar, P., Kowshik, S., Hari, R., & Venkata, P. (2016). Animation and Chirplet-Based Development of a PIR Sensor Array for Intruder Classification in an Outdoor Environment. https://doi.org/10.48550/arXiv.1604.03829.

[9] Liu J, He Y. (2023). Construction of Intelligent Building Automation Control System Based on Fuzzy Clustering Algorithm.

[10] Mark Baker, UK Tech News Editor. (2022, July 19). Smarter building monitoring and optimisation power by IoT - UK Tech News. UK Tech News. https://uktechnews.co.uk/2022/07/19/smarter-building-monitoring-and-optimisation-power-by-iot/.

[11] Nijim M, Kanumuri V, Albetaineh H, Goyal A. (2023). Intelligent Monitoring and Management of Smart Buildings using Machine Learning: Optimizing User Behaviour and Energy Efficiency.

[12] Poyyamozhi, M., Murugesan, B., Rajamanickam, N., Shorfuzzaman, M., & Aboelmagd, Y. (2024). IOT—A Promising solution to energy management in smart Buildings: A Systematic Review, applications, barriers, and future scope. Buildings, 14(11), 3446. https://doi.org/10.3390/buildings14113446.

[13] Radovanovic, I. (2023, April 3). Google CoLab - A step-by-step guide - AlgoTrading101 blog. Quantitative Trading Ideas and Guides - AlgoTrading101 Blog. https://algotrading101.com/learn/google-colab-guide/.

[14] Zafar U, Bayhan S, Sanfilippo A. (2020). Home Energy Management System Concepts, Configurations, and Technologies for the Smart Grid. [9] C. Belloni, “Hydrodynamics of Ducted and Open-Centre Tidal Turbines,” p. 230, 2013.

APPENDIX A
GANTT CHART

Activities	Week															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Title selection and supervisor																
Talk on Introduction & Plagiarism																
Title Research																
Meeting for Problem Statement and Objective																
Report Writing for Progress Report 1																
Talk on Literature Review and Mendeley																
Talk on Report Writing																
Progress Report 1 Submission																
Report Writing for Literature Review																
Talk on Methodology & Evaluation																
Talk on Presentation Skills																
Progress Report 2 Submission																
Improve the Project Implementation																

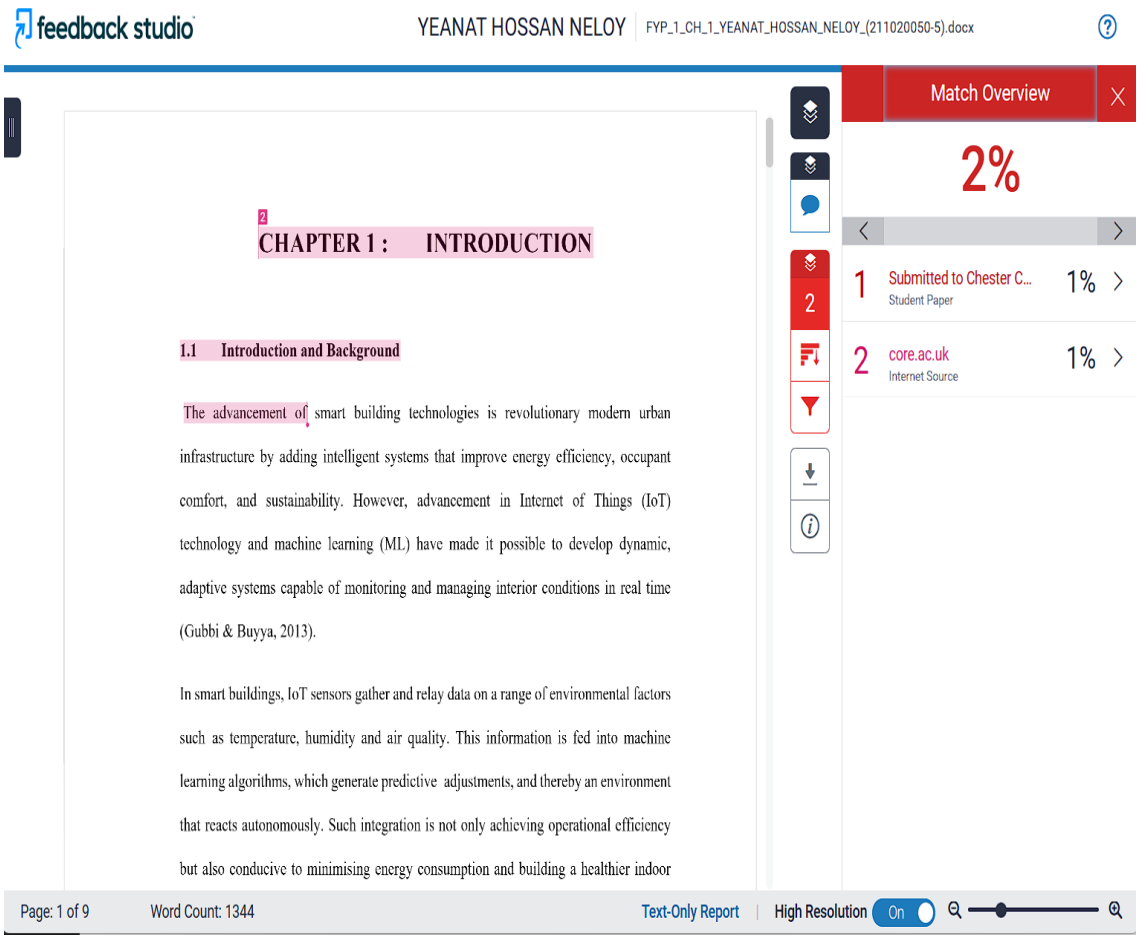
Full Report Submission																
FYP 1 VIVA Presentation																
Final Report Submission																

Activities	Week															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Rewrite Report																	
Briefing on FYP 2																	
Talk on Plagiarism and Turn-It-In																	
Talk on Analysis Results and Discussion																	
Designing of the Smart Building System																	
Report Writing for Progress Report 1																	
Progress Report 1 Submission																	

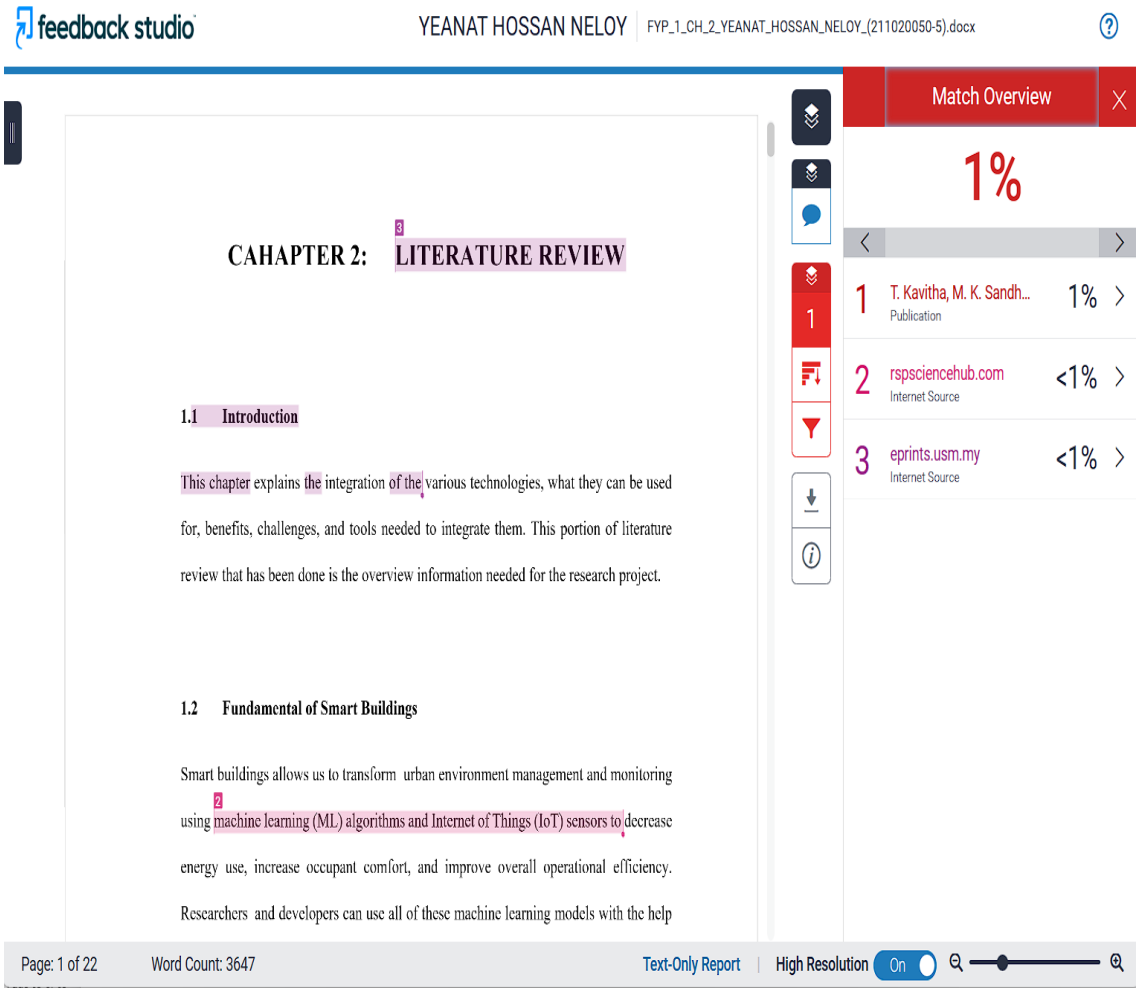
Project Implementation (Data Collection)																	
CGS Talk																	
Continue Writing Progress Report 2																	
Progress Report 2 Submission																	
Full Report Submission																	
FYP 2 VIVA Presentation																	
Final Report Submission																	

APPENDIX B
TURNITIN

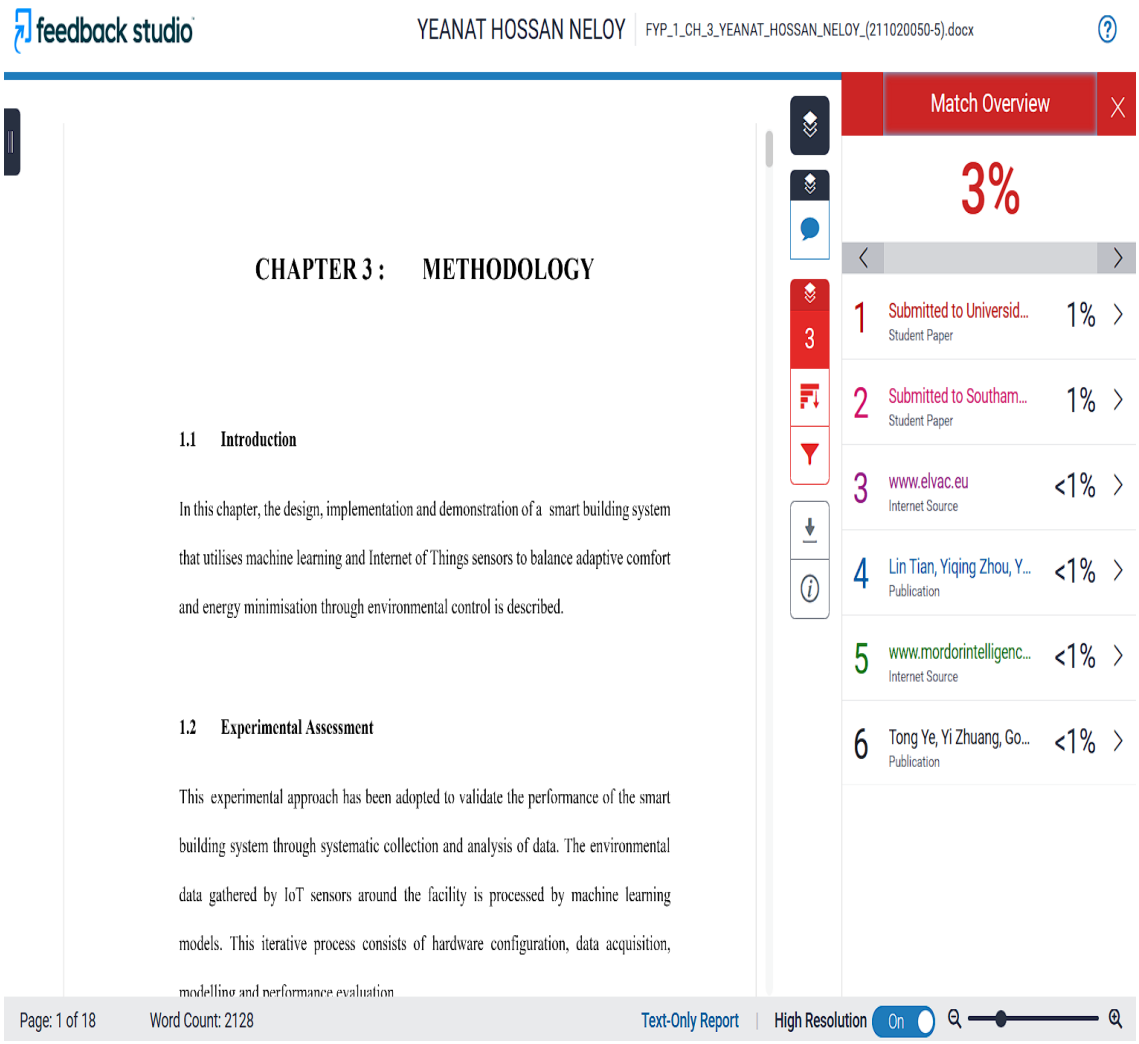
CHAPTER 1 : INTRODUCTION



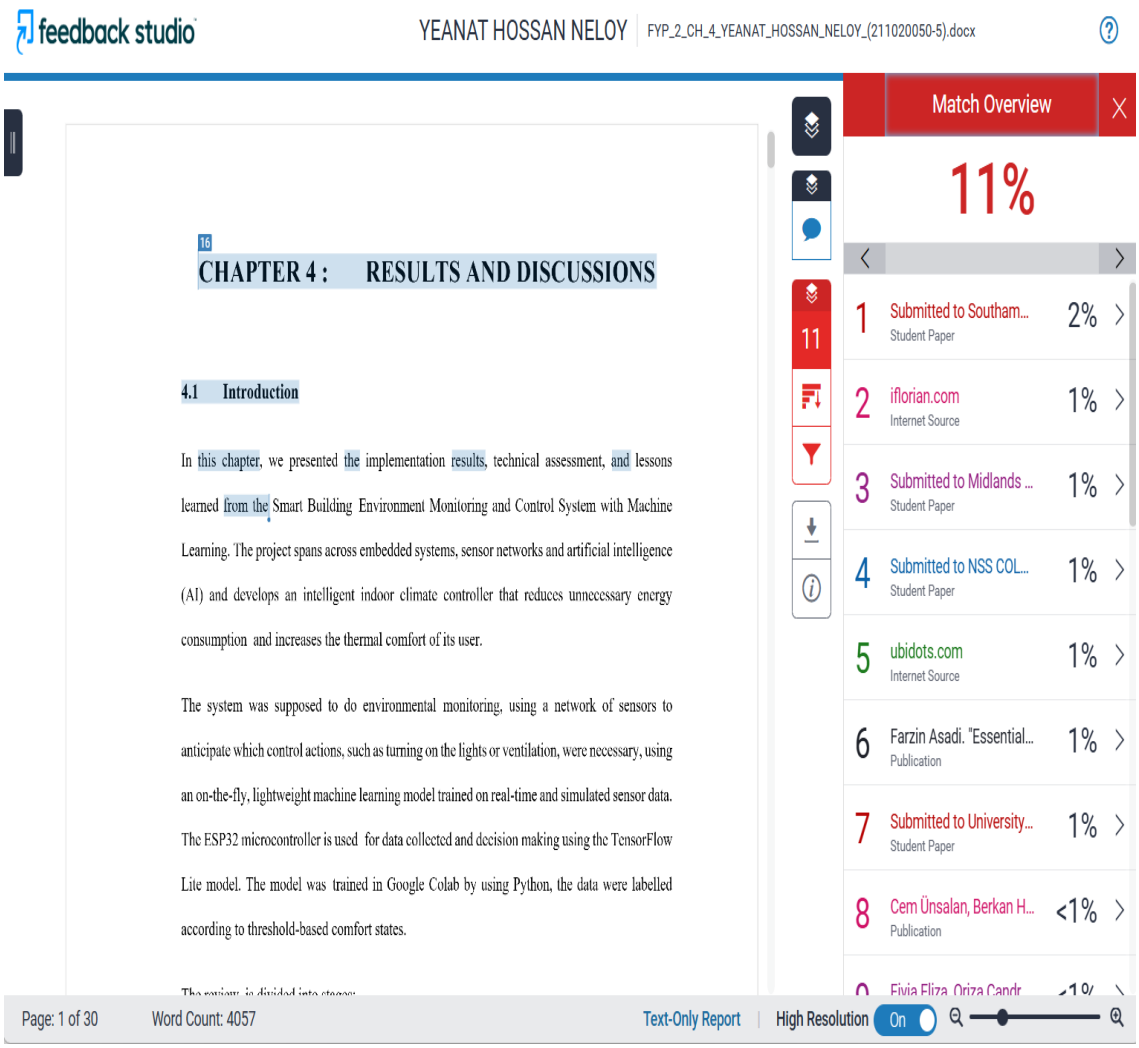
CHAPTER 2 : LITERATURE REVIEW



CHAPTER 3 : METHODOLOGY



CHAPTER 4 : RESULT AND DISCUSSIONS



CHAPTER 5 : CONCLUSION