FUTURE UNIVERSITY IN EGYPT
FACULTY OF COMPUTERS AND INFORMATION TECHNOLOGY

IOT Home Automation

"IHomA"

Prepared by

Abdallah Abdelmonem	**ID:** 20203048
Mennatallah Ashraf Mahmoud	**ID:** 20203498
Khaled Ibrahim	**ID:** 20204990
Galal Mohamed	**ID:** 20205070
Islam mohamed Ibrahim	**ID:** 20202210
Ahmad Mohamed Aldahshan	**ID:** 20203310

Supervised by

Prof. Dr./ Mohamed Ahmed Hussein

July 2024

Acknowledgement

We would like to thank everyone who made it possible for students like us. It has been a terrific opportunity to gain a lot of experience in real-time projects followed by knowledge of how to develop and assess actual projects.
At the beginning, we would like to express our gratitude and thanks to Allah, the Almighty, for providing us with innumerable opportunities, blessings, and knowledge that have allowed us to eventually achieve what we have.

We extend our sincere gratitude to **Professor/ Mohamed Ahmed Hussein** for his role as project supervisor.
There will never be enough thanks for all of his amazing assistance and support.

Every time we attend his meeting, we leave feeling inspired and uplifted. This idea would never have come to light without his continuous support and advice. He served us as both a father and a wise mentor that we turned to when we needed his support and expertise. It was a great privilege and honor to work and study under his guidance.

Table of Contents

Table of figures:

Abstract

This document presents a comprehensive overview of the Internet of Things (IOT) Home Automation project. The project aims to develop a smart home system that can be controlled remotely using a mobile application. The system will be based on IOT technologies, such as sensors, microcontrollers, and wireless communication, to create a seamlessly integrated and automated home environment.

The primary objectives include enhancing convenience, comfort, and security for residents, as well as optimizing energy efficiency. Key features of the system include remote monitoring and control, real-time alerts and notifications, and voice control integration. By leveraging these technologies, the IOT Home Automation project aspires to provide homeowners with a user-friendly interface to manage and customize their home settings, ensuring a more efficient, secure, and sustainable living environment.

The project encompasses various components such as smart lighting, climate control, security systems, and appliance management. It also aims to offer scalability, allowing for the easy addition of new devices and features over time. The anticipated benefits include cost savings on utilities, improved home management through data analytics, and enhanced safety measures. This project represents a significant step towards realizing the full potential of smart home technology in everyday life.

1. Introduction

1.INTRODUCTION

1.1 Introduction

Home automation via the Internet of Things is the use of smart devices and sensors to control and manage various aspects of the home, such as temperature, lighting, and security. Smart sensors can detect any unusual activity, send alerts to homeowners, and even integrate with other security measures such as locks. IOT devices can communicate with each other and with a central controller, such as a smartphone or tablet, via the Internet. Home automation via IOT can make your home life more comfortable, safe and efficient.

1.2 Motivation

The adoption of IOT in home automation is driven by several compelling factors, all aimed at improving the overall quality of life of homeowners such as:

- **comfort:**

IOT home automation systems allow homeowners to easily control and monitor different aspects of their homes. Whether it's adjusting the thermostat or turning off the lights.

- **Remote monitoring and control:**

Homeowners can check the status of their security systems, adjust settings, and receive alerts from anywhere with an Internet connection, providing peace of mind and flexibility.

- **Enhance security:**

IOT enhances home security by providing advanced monitoring and access control systems. Smart cameras, motion sensors, and smart locks can be integrated to create a comprehensive security ecosystem. Alerts and notifications keep homeowners informed of any suspicious activities, contributing to a safer living environment.

- **Time saving automation:**

Automation is a major driver of IOT adoption in home settings. By automating routine tasks, homeowners can save time and focus on the more important aspects of their lives.

1.3 Problem Statement

IOT covers many aspects including:

- **Energy efficiency:**

Traditional homes often lack effective energy management, resulting in unnecessary energy consumption and high utility bills. Inefficient HVAC systems contribute to wasted energy.

- **Security:**

Security concerns, such as unauthorized break-ins and entry, are common in homes. Traditional security systems may lack real-time monitoring and flexibility.

- **Remote monitoring:**

Homeowners often face challenges monitoring their homes while they are away, leading to safety concerns and potential issues like leaks or power outages.

This project for designing and implementation a home automation system to control home's devices remotely using IOT technology.

1.4 Objectives

The main objectives of developing the proposed IOT system include:

1. Developing the user interface for the smart home system, which will allow the user to access, control and monitor appliances and home appliances remotely via a smartphone.
2. Implementation of the energy efficiency subsystem, which will use light sensors, temperature sensors, motion sensors, etc. to adjust lighting and heating/cooling according to the presence of people and external weather conditions.
3. To implement the security subsystem, which will use cameras, infrared sensors, magnetic door sensors, smoke sensors, gas sensors, etc. to detect any threats or dangers in the home and alert the user or authorities accordingly.
4. One device can control another device, such as an alarm, by sending a command to the heater to turn it on.
5. Monitor and optimize energy usage by controlling lighting, heating, cooling, and appliances based on real-time data and usage patterns.
6. Implement smart security systems including cameras, motion sensors, door/window sensors, and smart locks to ensure the safety of the home.
7. Ensure compatibility and seamless integration with existing home devices and systems.

8. Allow users to customize settings and automation routines according to their preferences and daily schedules.
9. Provide instant alerts and notifications for events such as security breaches, fire alarms, water leaks, and other critical incidents.
10. Design the system to be easily scalable, accommodating the addition of new devices and features as needed.
11. Develop an intuitive and easy-to-use interface for managing home automation systems, ensuring accessibility for all users.
12. Integrate with popular voice assistants like Amazon Alexa, Google Assistant, and Apple Siri for hands-free control of home devices.
13. Enable scheduling of tasks such as turning lights on/off, adjusting thermostats, watering plants, and more based on time or triggers like sunset and sunrise.
14. Integrate safety features such as smoke detectors, carbon monoxide detectors, and emergency shut-off systems for gas and water.
15. Reduce utility bills and maintenance costs through efficient management and proactive maintenance of home systems.

1.5 Time Plan

Task Name	Start	Finish
Requirment	Wed 11/1/23	Thu 11/30/23
Analysis	Tue 11/7/23	Thu 11/30/23
Design	Tue 11/14/23	Wed 11/29/23
Mobile application	Fri 12/1/23	Sat 12/30/23
Embedded smart home	Fri 12/1/23	Sat 12/30/23
Structure	Fri 12/1/23	Sat 12/30/23
Function	Thu 12/7/23	Sat 12/30/23
Realtime database	Thu 12/7/23	Sat 12/30/23
ERD	Thu 12/7/23	Sat 12/30/23
Schema	Thu 12/14/23	Sat 12/30/23
Usecase	Thu 12/14/23	Sat 12/30/23
Sequence diagram	Thu 12/14/23	Sat 12/30/23
Class diagram	Thu 12/14/23	Sat 12/30/23
First-Term documentation	Tue 12/12/23	Sat 12/30/23
Implementation	Mon 1/1/24	Wed 1/17/24
Mobile application	Mon 1/1/24	Tue 1/30/24
Application GUI	Mon 1/1/24	Tue 1/30/24
Class diagram	Sun 1/7/24	Tue 1/30/24
Realtime database	Sun 1/14/24	Tue 1/30/24
Embedded smart home	Thu 2/1/24	Wed 2/28/24
Function	Thu 2/1/24	Wed 2/28/24
Manufacturing	Wed 2/7/24	Wed 2/28/24
Integrating mobile app with embedded system	Wed 2/14/24	Wed 2/28/24
Unit testing	Fri 3/1/24	Sat 3/30/24
Mobile application	Thu 3/7/24	Sat 3/30/24
Embedded system	Thu 3/14/24	Sat 3/30/24
Realtime database	Mon 4/1/24	Tue 4/30/24
Evalutation	Sun 4/7/24	Tue 4/30/24
Scenario	Sun 4/14/24	Tue 4/30/24
Mobile application	Wed 5/1/24	Thu 5/30/24
Embedded smart home	Tue 5/7/24	Thu 5/30/24
Realtime database	Tue 5/14/24	Thu 5/30/24
Final documentation	Tue 5/14/24	Thu 5/30/24

Figure 1.Time Plan

2. Background

2. BACKGROUND

2.1 Introduction

Internet of Things (IOT), refers to the collective network of connected devices and the technology that facilitates communication between devices and the cloud, as well as between the devices themselves.

- ⇨ **IOT works:**
  - As a standard Internet of things system works by gathering and exchanging data in real time. There are three components for an IOT system:
    - Smart devices
    - IOT application
    - A graphical user interface

In addition to some topics. that will provide background information on the different subjects we will be using:

- System development life cycle.
- Artificial intelligence.
- Microcontrollers.
- Motors.
- Sensors.
- Cameras.

2.2 System development life cycle (SDLC)

The Software Development Life Cycle (SDLC) refers to a methodology with clearly defined processes for creating high-quality software. in detail, the SDLC methodology focuses on the following phases of software development:

- Requirement analysis
- Design
- Implementation
- Testing
- Evaluation

Figure 2.SDLC

- **Requirement Analysis:** Gather and document stakeholder needs to create a detailed requirement specification for guiding all development activities.
- **Design**: Develop system architecture and detailed design specifications, guiding developers in building the system according to requirements.
- **Implementation**: Convert design into functional code, performing initial testing to ensure individual components work correctly.
- **Testing**: Conduct thorough testing to identify and fix defects, ensuring the system meets all requirements and functions correctly.
- **Evaluation (Deployment)**: Install and configure the system in the production environment, monitor performance, and make necessary adjustments for optimal operation

2.2.1 Software development life cycle define

SDLC or the Software Development Life Cycle is a process that produces software with the highest quality and lowest cost in the shortest time possible. SDLC provides a well-structured flow of phases that help an organization to quickly produce high-quality software which is well-tested and ready for production use.

2.2.2 Software development life cycle models:

- **Waterfall Model**

The waterfall model is a software development model that involves a linear and sequential approach, where each phase of the project is completed before moving on to the next one. The phases are typically: requirements analysis, design, implementation, testing, and operation. The waterfall model is useful for projects with well-defined and clear requirements, where there is little room for error or change.

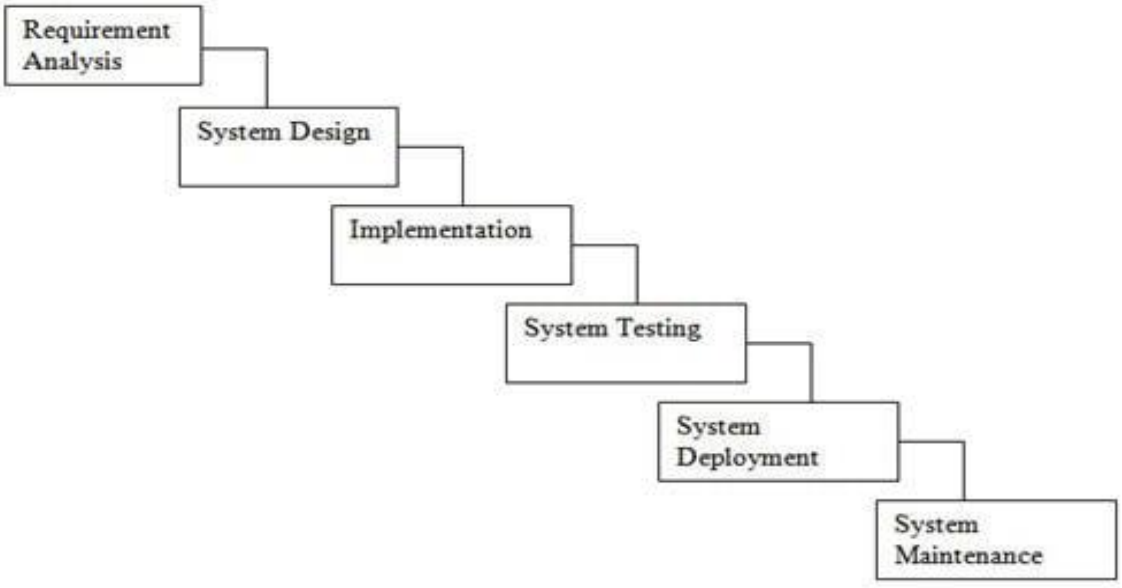


Figure 3.Waterfall Model

The phases in this process model are:

1. **Requirement Analysis**

In this phase, all requirements of the software product are gathered in this phase and documented in software requirement specification document.

2. **System Design**

   In this phase, overall structure of the software product is designed based on requirement analysis phase.

3. **Implementation**

   The development of software starts in this phase. It develops in small programs which are called units. These units are tested according to their functionality and integrated in the next phase.

4. **System Testing**

   In this phase, integrate all units which are developed in implementation phase. After integration the whole product is tested to check whether it meets its goals. Software defects and bugs are reported if they are available then fix and retested.

5. **System Deployment**

   When function and non-functional requirements are tested and validated then the software is deployed in the customer's environment.

6. **System Maintenance**

   If some problems are faced in the customer's environment, then solve these problems in maintenance phase. Also, some enhancement can do in this phase if user is not fully satisfied

Advantages of waterfall model

- It is very easy to understand and use.
- Each phase has a specific deliverable and review process.
- Phases are processed and completed one at a time.
- Works well for small projects where requirements are well understood.
- Clearly defined phases.
- Well understood deliverables.
- Arranging tasks easily.
- Process and results are documented.

Disadvantages of waterfall model

- Working software is available late during the life cycle.
- It has a lot of risks.
- A poor model for large projects.
- It is not fit for projects where requirements are changed frequently.
- Difficult to measure progress during phases.
- It cannot manage changing requirements.

• Adjusting scope during the process model can end a project.

- **Agile Model**

Agile is an approach to project management that centers around incremental and iterative steps to completing projects. The incremental parts of a project are carried out in short-term development cycles. The approach prioritizes quick delivery, adapting to change, and collaboration rather than top-down management and following a set plan.

In the Agile process, there is continuous feedback, allowing team members to adjust to challenges as they arise and stakeholders an opportunity to communicate consistently. Though originally created for software development, the Agile approach is now widely used in executing many different types of projects and in running organizations.

The phases in this process model are:

1. Requirement Gathering: - In this step, the development team must gather the requirements, by interaction with the customer. development team should plan the time and effort needed to build the project. Based on this information you can evaluate technical and economic feasibility.

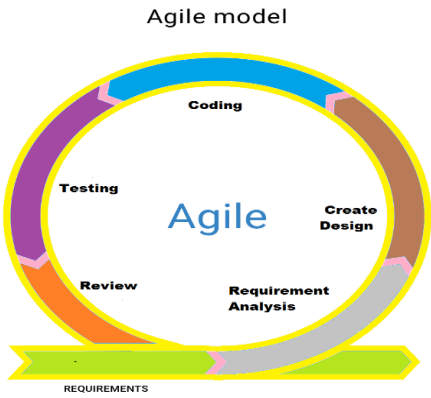


Figure 4.Agile model

2. Design the Requirements: - In this step, the development team will use user-flow-diagram or high-level UML diagrams to show the working of the new features and show how they will apply to the existing software. Wire framing and designing user interfaces are done in this phase.

3. Construction / Iteration: - In this step, development team members start working on their project, which aims to deploy a working product.

4.Testing / Quality Assurance:- Testing involves Unit Testing, Integration Testing, and System Testing**.** A brief introduction of these three tests is as follows:

- **Unit Testing:** - Unit testing is the process of checking small pieces of code to ensure that the individual parts of a program work properly on their own. Unit testing is used to test individual blocks (units) of code.
- **Integration Testing:** - Integration testing is used to identify and resolve any issues that may arise when different units of the software are combined.
- **System Testing:** - Goal is to ensure that the software meets the requirements of the users and that it works correctly in all possible scenarios.

5.Deployment: - In this step, the development team will deploy the working project to end users.

6.Feedback: - This is the last step of the Agile Model**.** In this, the team receives feedback about the product and works on correcting bugs based on feedback provided by the customer.

Advantages of Agile model:

- Increased collaboration
- Increased project visibility
- Better alignment to business needs
- Delivery speed/time to market
- Project risk reduction
- Faster response to competitive threats

Disadvantages of Agile model:

- Organizations can resist change in adoption
- Inadequate training or education
- Not enough leadership participation
- Not enough organizational knowledge of Agile

- **Spiral Model**

The spiral model is a software development life cycle (SDLC) model that provides a structured, iterative approach to software development. In its schematic representation, it looks like a spiral with many rings. The exact number of spirals is unknown and can vary from project to project. Each loop of the spiral is called a phase of the software development process.

The project manager can vary the exact number of stages needed to develop the product depending on the risks of the project.

Since the project manager dynamically determines the number of stages, the project manager has an important role in product development using the spiral model.

It is based on the idea of a spiral, where each iteration of the spiral represents a complete software development cycle, from requirements gathering and analysis to design, implementation, testing, and maintenance.

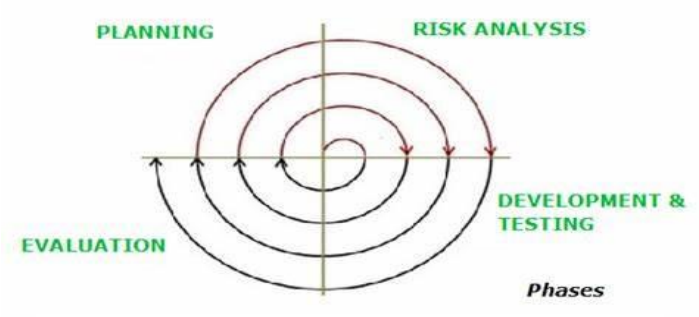


Figure 5.Spiral Model

The phases in this process model are:

1. Planning

The first phase of the Spiral Model is the planning phase, where the scope of the project is determined, and a plan is created for the next iteration of the spiral.

2. Risk Analysis

In the risk analysis phase, the risks associated with the project are identified and evaluated.

3. Engineering

In the engineering phase, the software is developed based on the requirements gathered in the previous iteration.

4. Evaluation

In the evaluation phase, the software is evaluated to determine if it meets the customer's requirements and if it is of high quality.

Advantages of Spiral model:

- Risk Handling
- Good for large projects
- Flexibility in Requirements
- Customer Satisfaction
- Iterative and Incremental Approach
- Emphasis on Risk Management
- Improved Communication
- Improved Quality

Disadvantages of Spiral model:

- Complex
- Expensive
- Too much dependability on Risk Analysis
- Difficulty in time management
- Complexity
- Time-Consuming
- Resource Intensive

2.3 The Internet of Things (IOT)

As explained previously, that The Internet of Things (IOT), is a network of physical devices. These devices can transfer data between each other without human intervention. IOT devices are not limited to computers or machines. IOT can include anything equipped with a sensor that has been assigned a unique identifier (UID). The primary goal of the Internet of Things is to make devices that can communicate with each other (and users) in real time and IOT system has three components include:

- **Smart devices:** This is a device, like a television, security camera, or exercise equipment that has been given computing capabilities. It collects data from its environment, user inputs, or usage patterns and communicates data over the internet
- **IOT application:** An IOT application is a collection of services and software that integrates data received from various IOT devices. It uses artificial intelligence (AI) technology to analyze this data and make informed decisions. These decisions are communicated back to the IOT device and the IOT device then responds intelligently to inputs.
- **A graphical user interface:** The IOT device can be managed through a graphical user interface. Common examples include a mobile application or website that can be used to register and control smart devices.

Examples of the Internet of Things (IOT):

You probably use IOT devices every day. The list below shows some IOT devices you may be familiar with:

- **Smart Home Devices**: Smart home devices such as thermostats and home security systems can operate autonomously to help with daily tasks. For example, you can program your smart thermostat to automatically set to the cooling setting before you get home from work. Or you may receive a notification from your security camera letting you know someone is at the door when you're not home.
- **Connected cars:** There are many ways vehicles, such as cars, can be connected to the internet. It can be through smart ashcans, infotainment systems, or even the vehicle's connected gateway. They collect data from the accelerator, brakes, speedometer, odometer, wheels, and fuel tanks to monitor both driver performance and vehicle health.
- **Wearable technologies:** One of the most common examples of the Internet of Things are smart watches. Wearable IOT technology like Fitbits and Apple Watches connect to other devices (like your smartphone) to share data. They usually also connect to the Internet to track GPS locations.
- **Smart cities:** IOT applications have made urban planning and infrastructure maintenance more efficient. Governments are using IOT applications to tackle problems in infrastructure, health, and the environment.

2.4 Artificial Intelligence (AI)

Artificial Intelligence (AI) has rapidly evolved from a distant future concept into a tangible force reshaping various aspects of our lives. In the world of home automation, artificial intelligence is at the forefront of the technological revolution, ushering in a new era of smart living. With the ability to analyze data, make intelligent decisions, and adapt to user preferences, AI-powered systems are transforming our homes into intuitive and responsive environments.

Home automation, powered by artificial intelligence, goes beyond connectivity and remote control. It involves deploying sophisticated algorithms and machine learning models that enable devices to learn from user behavior, predict needs, and autonomously improve operations. This paradigm shift in home automation is enabling homeowners to experience a level of convenience, efficiency, and customization that was previously the stuff of science fiction.

Imagine a home that learns your daily routine, adjusts lighting and temperature accordingly, anticipates your preferences, and also proactively manages energy consumption. This is the role of artificial intelligence in home automation. From smart thermostats that improve climate control to smart security systems that can distinguish between familiar and unfamiliar faces, artificial intelligence provides a level of intelligence and adaptability that dramatically elevates the capabilities of automated home systems.

2.5 Mobile Application

Mobile application development is the process of designing and building software for mobile devices, such as smartphones and tablets. These applications can be pre-installed on the device, downloaded from an app store, or accessed via a mobile web browser. The development process involves several stages, including design, development, testing, deployment, and maintenance.

- **Types of Mobile Applications:**
  - **Native Mobile Applications:** Developed specifically for one platform (iOS or Android) using the respective platform's programming languages and tools.
  - **Cross-platform Native Mobile Applications:** Built using frameworks like React Native or Xamarin that allow the same codebase to be used for multiple platforms.
  - **Hybrid Mobile Applications:** These apps are essentially web applications packaged in a native app shell. They are built using web technologies like HTML, CSS, and JavaScript, and can run on any platform with a web view component.
  - **Progressive Web Applications (PWAs):** These are web applications that deliver a mobile app-like experience through the browser.

- **Mobile Application Development Lifecycle:**
  - **Development:** Writing the code and building the application.
  - **Testing:** Ensuring the app works correctly on different devices and operating systems.
  - **Release:** Distributing the app through app stores or other channels.
  - **Monitoring:** Tracking app performance and user feedback.
  - **Analysis:** Using data to make improvements and fix issues.

2.6 Programing languages

Beyond the realm of mere convenience, the integration of IOT in home automation seeks to redefine our interaction with living spaces. Envision a home where lighting adjusts dynamically, climate control aligns with individual preferences, and security systems intelligently distinguish residents from intruders. This project is dedicated to bringing this vision to life, leveraging the power of IOT and emphasizing the critical role of programming languages, particularly in the context of Wi-Fi connectivity.

2.6.1 Python: A Foundation for Wi-Fi-Powered Connectivity:

Python, renowned for its readability and versatility, serves as a foundational element in our exploration of programming languages for Wi-Fi enabled IOT home automation. Its extensive library support and seamless integration capabilities make it an ideal choice for managing devices over Wi-Fi networks. From establishing communication protocols to handling data processing and analytics, Python's prowess ensures a robust foundation for our interconnected ecosystem.

2.6.2 C/C++: Resource-Constrained Devices:

In scenarios involving resource-constrained devices and low-level system programming, C and C++ take center stage. This project will showcase the role of these languages in optimizing Wi-Fi connectivity, ensuring optimal performance where resources are limited. From programming microcontrollers to managing network protocols, C/C++ become indispensable tools in achieving seamless connectivity.

2.6.3 Challenges and Solutions:

As we delve into the intricacies of home automation, the project will address challenges and propose solutions related to Wi-Fi connectivity. Standardizing communication protocols, such as MQTT and COAP, will be explored, highlighting their role in facilitating efficient data exchange over Wi-Fi networks. Additionally, the implementation of a robust middleware layer becomes crucial to bridge the gap between diverse devices, ensuring a harmonious and interconnected smart home experience.

2.7 Microcontrollers

A microcontroller is a small computer on a single integrated circuit that is designed to control specific tasks within electronic systems. It combines the functions of a central processing unit (CPU), memory, and input/output interfaces, all on a single chip. Microcontrollers are widely used in embedded systems, such as home appliances, automotive systems, medical devices, and industrial control systems. They are also used in consumer electronics products, such as gaming systems, digital cameras, and audio players.

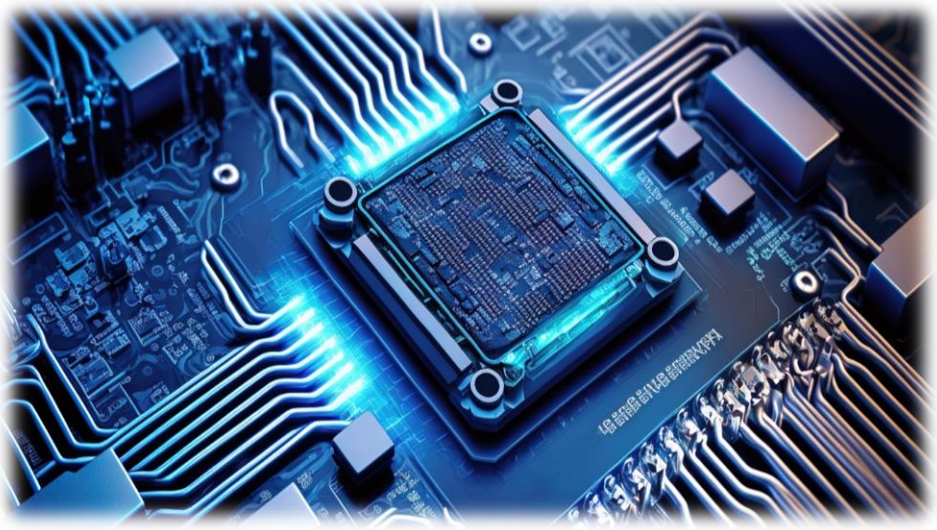


Figure 6.Microcontrollers

2.7.1 Arduino

2.7.1.1 Introduction

The Arduino hardware is an open-source circuit board that has a microprocessor and input/output (I/O) pins. These pins are used for communication and controlling physical objects such as LEDs, servos, and buttons. The board is typically powered via USB or an external power supply, which in turn allows it to power other hardware and sensors.

Arduino also has an open-source software component. This means that you can view schematics of every board available. The Arduino integrated development environment (IDE) is written in Java and will work on a variety of platforms: Windows, Mac, and Linux.

The Arduino IDE is a software application that allows you to program the Arduino board. It is a simple and easy-to-use program that can be used to create a variety of projects.

The Arduino is a versatile and powerful tool that can be used for a variety of projects. It is a great choice for beginners and experienced makers alike.

Here are some of the things you can do with Arduino:

- Control LEDs, motors, and other physical objects
- Create interactive projects, such as games and robots
- Collect data from sensors

- Communicate with other devices
- Build your own projects

2.7.1.2 What makes up an Arduino?

Arduinos contain several different parts and interfaces together on a single circuit board.

Several pins, which are used to connect with various components you might want to use with the Arduino. These pins come in two varieties:

- Digital pins, which can read and write a single state, on or off. Most Arduinos have 14 digital I/O pins.
- Analog pins, which can read a range of values, and are useful for more fine-grained control. Most Arduinos have six of these analog pins.
- A power connector, which provides power to both the device itself, and provides a low voltage which can power connected components like LEDs and various sensors, provided their power needs are reasonably low.
- A microcontroller, the primary chip, allows you to program the Arduino for it to be able to execute commands and make decisions based on various inputs. The exact chip varies depending on what type of Arduino you buy, but they are generally Atmel controllers, usually an ATmega8, ATmega168, ATmega328, ATmega1280, or ATmega2560.
- A serial connector, which on most newer boards is implemented through a standard USB port. This connector allows you to communicate to the board

from your computer, as well as load new programs onto the device. Often Arduinos can also be powered through the USB port, removing the need for a separate power connection.

- A variety of other small components, like an oscillator and/or a voltage regulator, which provide important capabilities to the board, although you typically don’t interact with these directly.

2.7.1.3 Types of Arduino boards

2.7.1.3.1 Arduino Uno R3

Arduino UNO is the best microcontroller board to get started with electronics and coding based on the **ATmega328P**. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button.

Arduino Uno Pros:

- **Easy Connection:** You can effortlessly join the board to your computer via USB.
- **Changeable Microcontroller:** If it gets damaged, you can swap the heart of the board, the ATmega328 microcontroller. Not all Arduino boards provide this feature.
- **Steady Voltage:** The board has pins that provide a constant 5V power supply, which is helpful for various components. You can adjust the voltage using digital and analog pins.

Figure 7.Arduino Uno 1

- **User-Friendly Design:** The community well-supports its straightforward layout, making it user-friendly.
- **Hardware Interaction:** The UNO can connect with Bluetooth, the internet, and motors.

Arduino Uno Cons:

- It cannot run more than one program at the same time.
- It has no memory safety checks.
- It is expensive for the CPU power and memory.

Figure 8.Arduino Uno 2

2.7.1.3.2 Arduino Nano

Arduino Nano is a type of microcontroller board, and it is designed by Arduino.cc. It can be built with a microcontroller like Atmega168 or Atmega328. This microcontroller is also used in Arduino UNO. It is a small size board and flexible with a wide variety of applications.

Arduino Nano Pros:

- **Compact Design:** The Arduino Nano is known for its small and compact design. Measuring just 18mm x 45mm, it fits easily into various projects, making it an ideal choice for applications where space is a constraint.

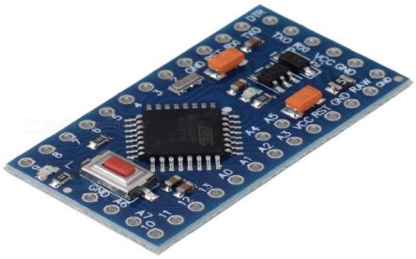


Figure 9.Arduino Nano 1

- **Ease of Use:** With compatibility with the Arduino IDE, the Nano ensures a user-friendly experience. Thanks to extensive support and community-driven resources, even those new to programming can quickly get started.
- **Power Efficiency:** The Arduino Nano is designed with power efficiency in mind. It operates at a lower voltage, consuming less power than other boards.

Arduino Nano Cons:

- **Lack of Native Connectivity:** The Arduino Nano does not come with onboard Wi-Fi or Bluetooth capabilities. This lack of native connectivity can limit its use in Internet of Things (IOT) applications.
- **Limited On-Board Memory:** The Nano's limited onboard memory can make complex programs and applications challenging. With a 32K program memory limit, complex projects like robotics or builds with intricate user interfaces might face constraints.

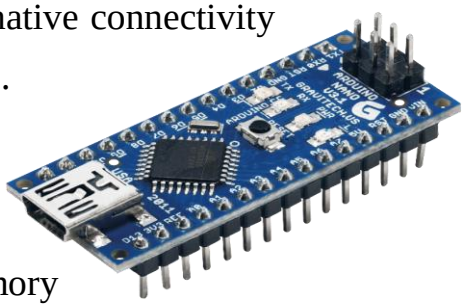


Figure 10.Arduino Nano 2

2.7.1.3.3 Arduino Micro

The Micro is a microcontroller board based on the ATmega32U4. It has 20 digital input/output pins (of which 7 can be used as PWM outputs and 12 as analog inputs), a 16 MHz crystal oscillator, a micro-USB connection, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a micro-USB cable to get started.

Arduino Micro Pros:

- It has 20 digital input/output pins (of which 7 can be used as PWM outputs and 12 as analog inputs)
- a 16 MHz crystal oscillator, a micro-USB connection, an ICSP header, and a reset button.
- Built-in USB communication: The ATmega32U4 has built-in USB communication that allows the Micro to appear as a mouse/keyboard on your machine.
- More sustainable than the Nano

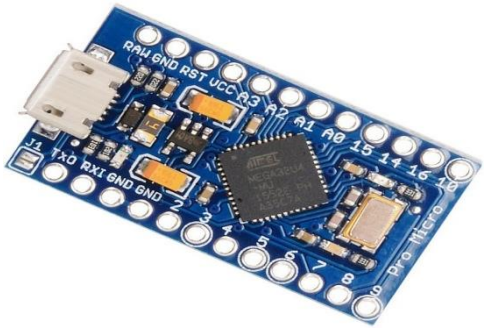


Figure 11.Arduino Micro 1

Arduino Micro Cons:

- More expensive than the Nano

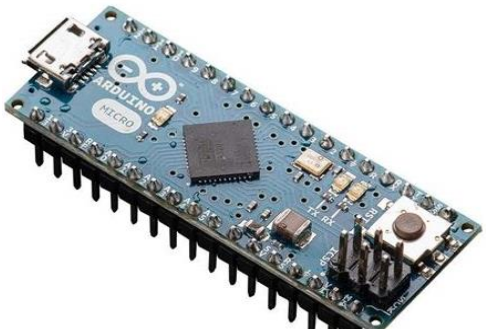


Figure 12.Arduino Micro 2

2.7.2 ATmega328 Microcontroller Chip

Digital I/O Pins	14
Analog I/O Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
SRAM	2 KB
EEPROM	1 KB
Flash Memory	32 KB
Clock Speed	16 MHz
Operating Voltage	5V
Input Voltage	7-12V

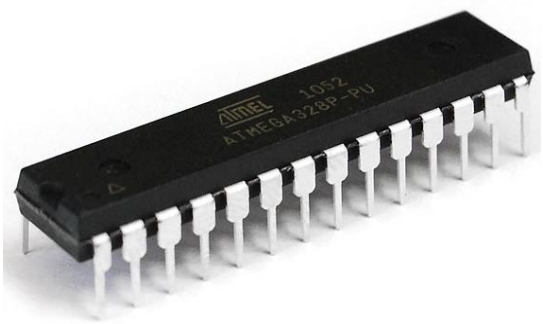


Figure 13.ATmega328 Microcontroller Chip

2.7.3 ATmega32U4 Microcontroller Chip

CPU	8-Bit
Digital I/O Pins	24
PWM Output	7
Analog Input	12

Flash Memory (Program Memory)	32 KB
SRAM	2.5 KB
EEPROM	1 KB
Input Voltage	7-12 V
Operating Voltage	5 V
Oscillator	up to 16 MHz
Software Used	Arduino Software (IDE)
Reset Button	Yes
ICSP Header	Yes
USB Port	1
UART (Serial Communication)	Yes

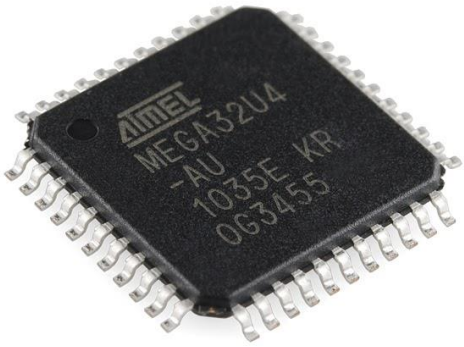


Figure 14.ATmega32U4 Microcontroller Chip

2.7.4 Raspberry Pi

2.7.4.1 Introduction

Raspberry Pi is a project that uses a Raspberry Pi to control devices in a home using the Internet of Things (IOT). The Raspberry Pi is a small, single-board computer that can be programmed to control a variety of devices, including lights, switches, and thermostats. The IOT allows devices to communicate with each other and with the internet, which makes it possible to control them remotely.

The Raspberry Pi is a low-cost, credit-card-sized computer that can be used to create a variety of projects, including IOT home automation systems. The Pi can

be programmed using a variety of programming languages, including Python, C++, and Java.

2.7.4.2 Execution

- To create an IOT home automation system using a Raspberry Pi, you will need the following:
- A Raspberry Pi
- A power supply for the Raspberry Pi
- A micro SD card with a Linux operating system installed on it
- A case for the Raspberry Pi
- A variety of sensors and actuators to control your devices
- IOT home automation systems can provide a variety of benefits, including:
- **Convenience:** You can control your devices from anywhere in the world using a mobile app or web interface.
- **Security:** You can monitor your home remotely and receive alerts if there is an issue.
- **Energy efficiency:** You can automate your devices to save energy
- **Comfort:** You can create a more comfortable home environment by automating your devices.

2.7.4.3 Types of Raspberry Pi modules

2.7.4.3.1 Raspberry Pi zero

- **Features:**

1. Tiny
2. Built-in Wi-Fi / Bluetooth
3. Requires extra adapter
4. Great for small projects
5. Cost 5$ to 10$

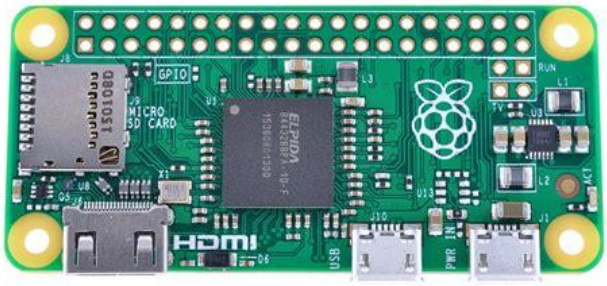


Figure 15.Raspberry Pi zero

2.7.4.3.2 Raspberry Pi 3 Model A+

- **Features:**

1. Faster than The Pi zero
2. Full size USB audio, HDMI
3. Requires extra adapter
4. Great for small project that need more power
5. Cost 25$

Figure 16.Raspberry Pi 3 Model A+

2.7.4.3.3 Raspberry Pi 3 Model B

- **Features:**

1. Four full size USB ports
2. Full size Ethernet
3. Up to 4gb of memory
4. Quad-core processor
5. Can support dual monitors
6. Built in Wi-Fi and Bluetooth
7. Starts at 35$

Figure 17.Raspberry Pi 3 Model B

2.7.4.3.4 Raspberry Pi 4 Model B

- **Features:**

1. Processor: Quad-core ARM Cortex-A72 CPU
2. RAM: Up to 8GB of RAM
3. Ports:
   - 2 USB 3.0 ports
   - 2 USB 2.0 ports
   - 2 micro HDMI ports
   - Gigabit Ethernet port
   - 3.5mm audio jack
   - micro SD card slot
4. Power: USB-C for power supply
5. Network:
   - Built-in dual-band Wi-Fi (802.11ac)
   - Bluetooth 5.0

Figure 18.Raspberry Pi 4 Model B

2.7.4.4 Difference between Raspberry Pi 3 and 4

	Raspberry PI 4	Raspberry PI 3
1. Processor:	It is powered by a quad-core ARM Cortex-A72 CPU	It features a quad-core ARM Cortex-A53 CPU.
2. RAM:	It comes with 2GB, 4GB, or 8GB of LPDDR4 RAM	It typically has 1GB of LPDDR2 RAM.
3. USB Ports:	It features two USB 3.0 ports and two USB 2.0 ports	It has four USB 2.0 ports.
4. HDMI Outputs:	It has two micro-HDMI ports, capable of supporting dual displays at up to 4K resolution.	It has one HDMI port that supports a single 1080p display.
5. Ethernet:	It comes with Gigabit Ethernet	It has 10/100 Mbps Ethernet.
6. Wireless Connectivity:	Built-in dual-band Wi-Fi (802.11ac) and Bluetooth 5.0.	Built-in Wi-Fi (802.11n) and Bluetooth 4.2.
7. Power Supply:	It requires a USB-C power supply.	It uses a micro-USB power supply.
8. GPIO Pins:	Both models have a 40-pin GPIO header for connecting to external hardware and sensors.	
9. USB-C Port:	**Raspberry Pi 4** uses a USB-C port for power, which is more robust than the micro-USB port on the **Raspberry Pi 3**.	

2.8 Sensors

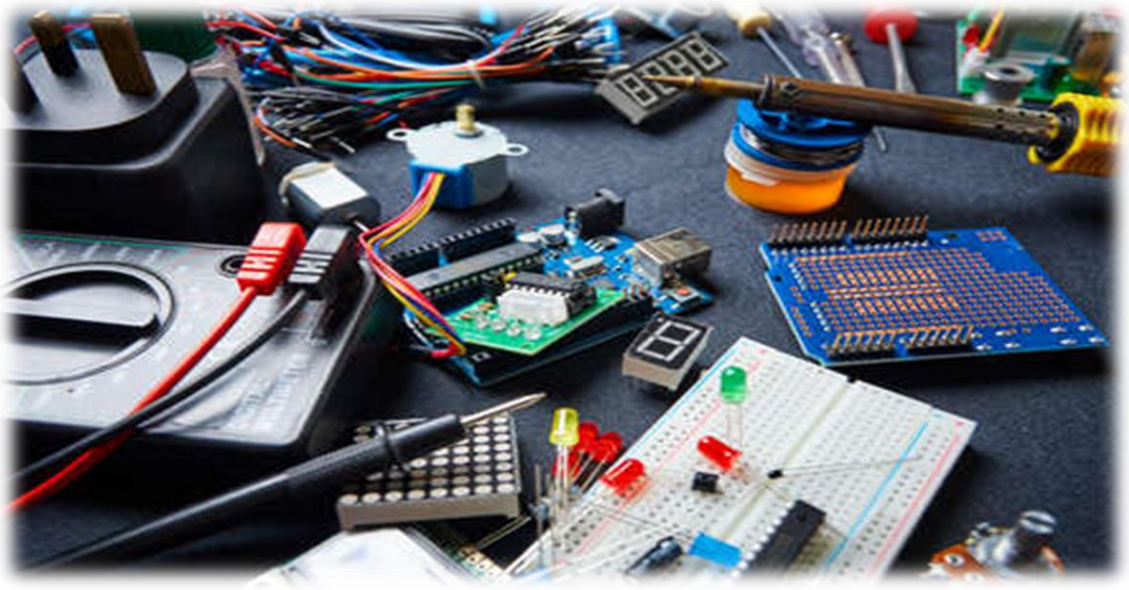


Figure 19.Sensors

2.8.1 Smart Camera

2.8.1.1 Introduction

A smart home camera is an electronic device that allows you to see and record what is happening in and around your home. Some models also transmit and record two-way audio, so you can talk to people near the camera. Smart cameras connect to your smart home hub via Wi-Fi, so you can control them from anywhere. You can also capture, download, and store the footage they collect.

Smart cameras are a valuable part of your home automation system, as they can provide you with peace of mind by allowing you to keep an eye on your home even when you're not there. They can also be used to monitor your pets, check on elderly relatives, or deter burglars.

➢ **Most common smart camera features:**

- Cloud storage
- Built-in storage
- Video capture
- Motion detection
- Two-way audio
- Solar charging and battery back-up

➢ **Examples of where you can use smart cameras at your home**

a) Connection methods:

- Wired
- Wireless

b) Indoor smart home cameras:

- Smart baby monitors and nanny cams
- Smart garage cameras
- Smart home security cameras

c) Outdoor smart home cameras:

- Smart video doorbells
- Smart door locks with video capture
- Smart floodlight camera

2.8.1.2 Most common Raspberry Pi smart cameras:

- **ZeroCam Noir**

The ZeroCam Noir is a camera module for the Raspberry Pi Zero or Raspberry Pi Zero W, so if you want to use it in a Raspberry Pi 3 or 2, you need to use an adapter cable.

This camera doesn't have an infrared filter on the lens, so it is perfect to take pictures in low light.

Figure 20.ZeroCam Noir

- **ZeroCam Fisheye**

This is the Fisheye version of the ZeroCam, which means it has a wide angle image read the "Field of View Comparison" section to compare the pictures.

This camera is also made for the Pi Zero or Pi Zero W, so to use it with another Pi board, you need an adapter cable.

Figure 21.ZeroCam Fisheye

- **Raspberry Pi Camera V2**

The Raspberry Pi Camera V2 features an 8 megapixel Sony IMX219 image sensor with fixed focus lens, it is capable of 3280×2464 pixel static images and supports 1080p30, 720p60, and 640×480p90 video.

The camera is compatible with all Raspberry Pi boards, but if you want to use it with the Pi Zero, you need an adapter cable.

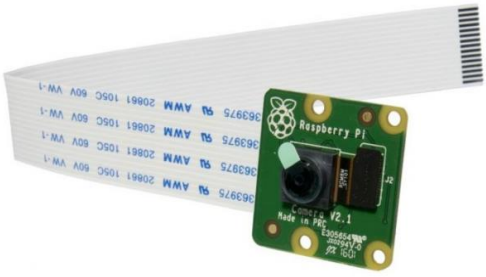


Figure 22.Raspberry Pi Camera V2

- **Raspberry Pi Camera V2 Noir**

The Raspberry Pi Camera V2 Noir has all the features of the V2 module, but it has no infrared filter. This means it is the perfect camera to see in the dark with infrared light.

Figure 23.Raspberry Pi Camera V2 Noir

- **Raspberry Pi Camera 1.3**

The Raspberry Pi Camera 1.3 is the predecessor of the V2 module. It features a 5 MP OmniVision OV5647 sensor.

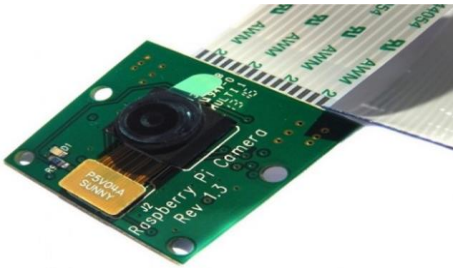


Figure 24.Raspberry Pi Camera 1.3

2.8.1.3 Most common Arduino smart cameras:

- **Esp32:** connected by WIFI and Bluetooth

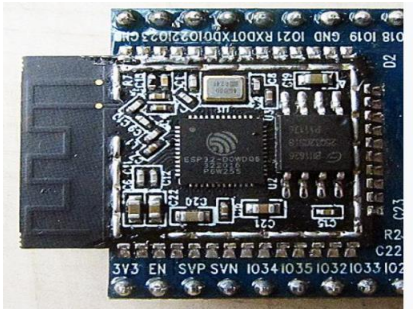


Figure 25.Esp32

- **Esp8266(Mega camera):** can be controlled in colors

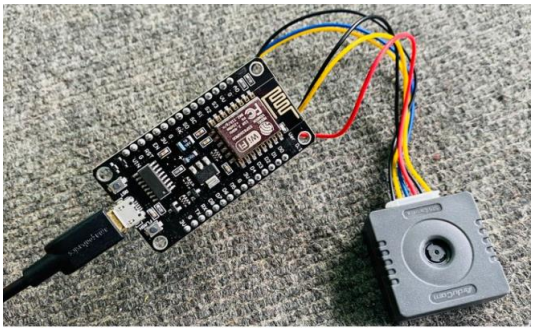


Figure 26.Esp8266(Mega camera)

- **FTDI:** Know how to let us write code for the camera

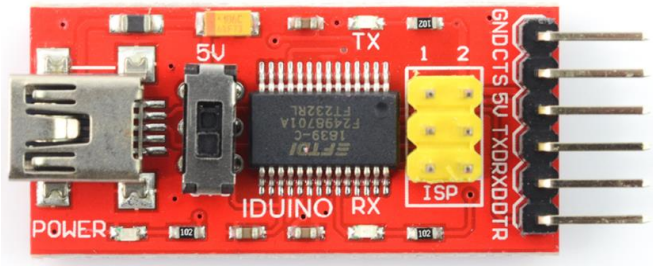


Figure 27.FTDI

2.8.2 Flame sensor

2.8.2.1 Introduction

The infrared flame sensor serves a dual purpose: it can identify the presence of a flame and measure the emitted infrared levels from the flame. This versatility makes the flame sensor a valuable tool for fire detection. It offers two output options through a digital output pin and an analog output pin.

- **The flame sensor includes four pins:**
  - **VCC pin:** It needs to be connected to VCC (3.3V to 5V).
  - **GND pin:** It needs to be connected to GND (0V).
  - **DO pin:** It is a digital output pin. It is HIGH if the flame is not detected and LOW if detected. The threshold value for flame detection can be adjusted using built-in potentiometer.
  - **AO pin:** It is an analog output pin. The output value decreases as the infrared level is decreased, and it increases as infrared level is increased.

Figure 28.Flame sensor 1

2.8.2.2 How it works:

- **For the DO pin:**
  - The module has a built-in potentiometer for setting the infrared threshold (sensitivity).
  - When the infrared intensity is above the threshold value, the flame is detected, the output pin of the sensor is LOW, and the DO-LED is on.
  - When the infrared intensity is below the threshold value, the flame is NOT detected, the output pin of the sensor is HIGH, and the DO-LED is off

- **For the AO pin:**
  - The higher the infrared intensity in the surrounding environment, the higher the value read from the AO pin.
  - The lower the infrared intensity in the surrounding environment, the lower the value read from the AO pin.

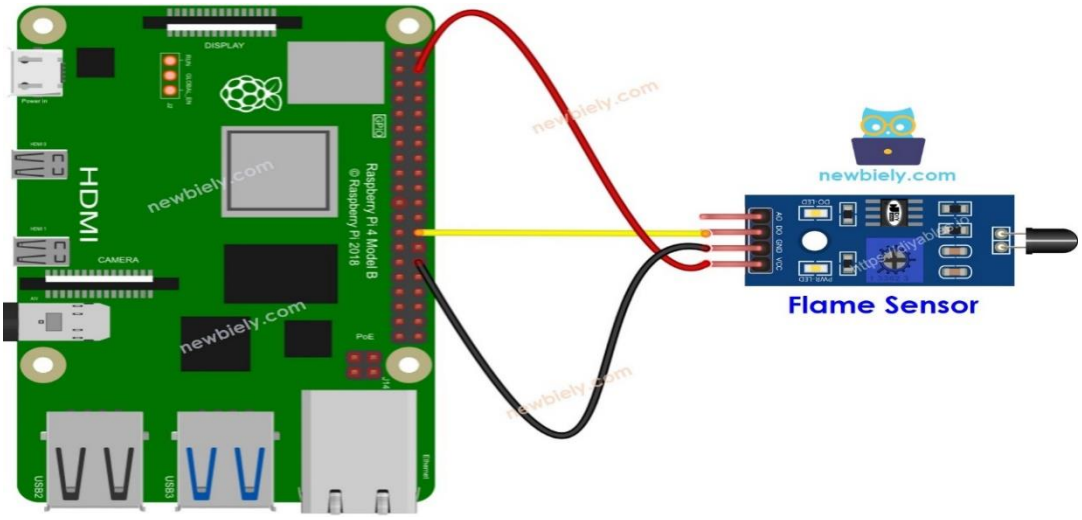


Figure 29.Flame sensor

2.8.3 Gas Sensor

2.8.3.1 Introduction

The MQ2 gas sensor is capable of detecting the presence of various gasses such as LPG, smoke, alcohol, propane, hydrogen, methane, and carbon monoxide in the surrounding environment. It provides two options for output: a digital output pin and an analog output pin.

It's important to note that the MO2 gas sensor does not provide information about each gas individually. Instead, provide Information about the combination of gasses or the presence of gasses as a whole.

By using the MQ2 sensor, we can determine if there is a gas leak or if the air quality is poor. This information is valuable in helping us take appropriate actions to ensure our safety, such as triggering an alarm or activating ventilation systems.

- **The MQ2 gas sensor has four pins that serve different purposes:**
  - **VCC pin:** This pin needs to be connected to a 5V power source (VCC).
  - **GND pin:** This pin needs to be connected to the ground (OV) for proper circuit operation.
  - **DO pin:** It is a digital output pin that indicates the presence of flammable gasses. When detected, the pin is set to LOW. Conversely, if no gasses are detected, the pin is set to HIGH. The threshold for detecting gas concentration can be adjusted by using the built-in potentiometer.
  - **AO pin:** This is an analog output pin that generates a voltage proportional to the gas concentration. As the gas concentration increases, the voltage on this pin also rises. Similarly, when the gas concentration decreases, the voltage decreases accordingly.

2.8.3.2 How it works:

- **Regarding the DO pin:**
  - The MQ2 module includes a built-in potentiometer that allows you to adjust the sensitivity or threshold for gas concentration.
  - When the concentration of gasses in the surrounding environment surpasses the set threshold value, the output pin of the sensor becomes LOW. and the DO-LED turns on.
  - Conversely, when the concentration of gasses in the surrounding environment falls below the threshold value, the output pin of the sensor becomes HIGH, and the DO-LED turns off.

- **Regarding the AO pin:**
  - As the gas concentration increases, the voltage on the AO pin also increases proportionally.
  - Conversely, as the gas concentration decreases, the voltage on the AO pin decreases accordingly.

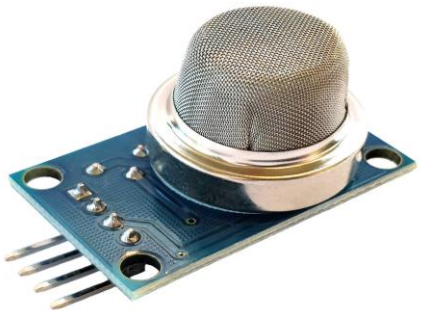


Figure 31.Gas Sensor 1

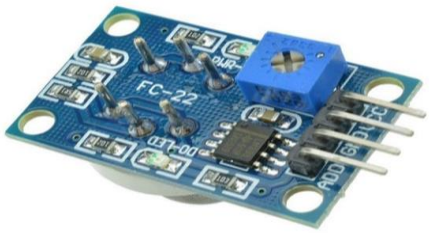


Figure 30.Gas Sensor 2

2.8.3.3 The MQ2 Sensor Warm-up

- **The MQ2 gas sensor requires a warming-up process before it can be used effectively.**
  - If the sensor has been stored for a long time (approximately a month or more) and you are using it for the first time, it needs to be warmed up for 24-48 hours. This extended warm-up time ensures accurate readings.
  - However, if the sensor has been recently used, it only takes around 5-10 minutes to reach its fully warmed-up state. During this warm-up period, the sensor may initially provide high readings, but they will gradually decrease until the sensor stabilizes.

To warm up the MQ2 sensor, simply connect its VCC and GND pins to a power supply or connect them to the VCC and GND pins of Raspberry Pi. Then, allow the sensor to remain in this connected state for the required period of time.

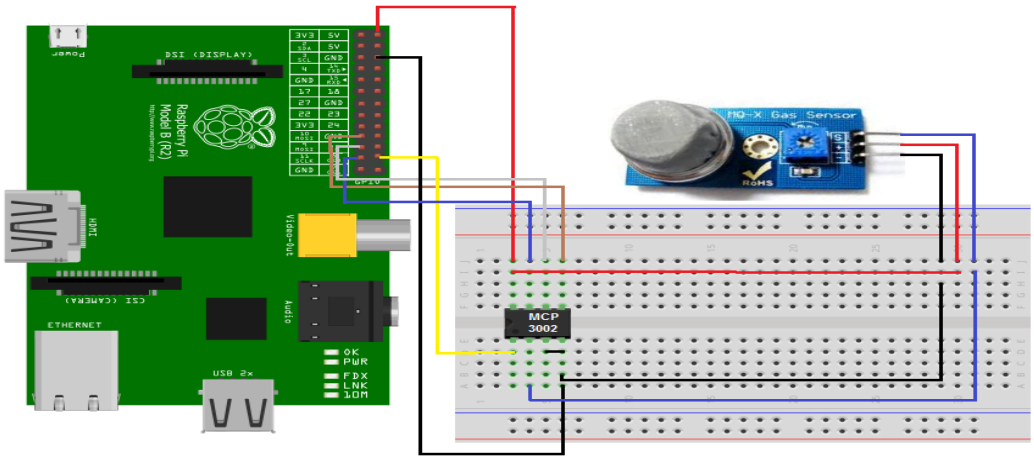


Figure 32. MQ2 Sensor Warm-up

2.8.4 Ultrasonic sensor

2.8.4.1 Introduction

An ultrasonic sensor is an instrument that measures the distance to an object using ultrasonic sound waves.

An ultrasonic sensor uses a transducer to send and receive ultrasonic pulses that relay back information about an object's proximity.

High-frequency sound waves reflect across boundaries to produce distinct echo patterns.

The sensor has 4 pins: **VCC**, **GND**, **Trig** and **Echo**.

- **VCC (voltage)** and **GND (ground): are** power pins.
- **Trig pin:** send the ultrasound wave from the transmitter.
- **Echo pin:** listen for the reflected signal.

Figure 33.ultrasonic sensor

2.8.4.2 How it works:

Ultrasonic sensors work by sending out a sound wave at a frequency above the range of human hearing. The transducer of the sensor acts as a microphone to receive and send the ultrasonic sound. Our ultrasonic sensors, like many others, use a single transducer to send a pulse and to receive the echo. The sensor determines the distance to a target by measuring time lapses between the sending and receiving of the ultrasonic pulse.

➢ **To connect your ultrasonic sensor with Raspberry Pi:**

As discussed earlier, ultrasonic sensors have a transmitter (Trigger) that can be used to transmit infrared sound waves and have a receiver that receives reflected sound waves (ECHO pin).

And it has power pins which are VCC and GND. The following picture shows the pinout of the ultrasonic sensor and its interface with the Raspberry Pi.

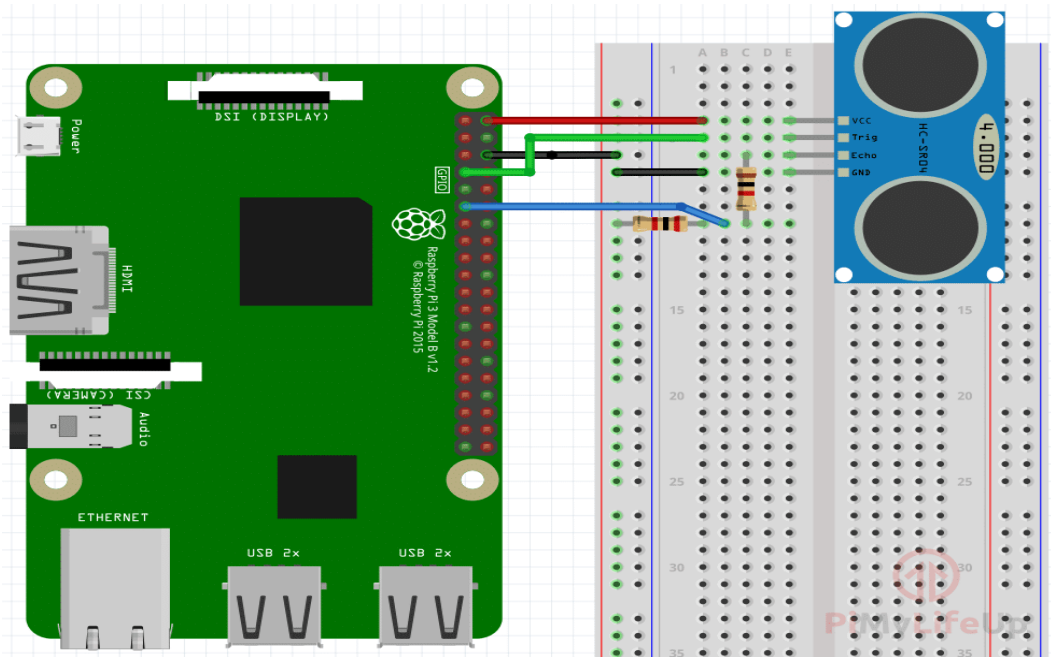


Figure 34.ultrasonic sensor with Raspberry Pi

2.8.5 Relay Module

2.8.5.1 Introduction

The relay is a simple electromechanical switch. While we use regular switches to manually close or open a circuit, a relay is also a switch that connects or disconnects two circuits. But instead of manual operation, the relay uses an electrical signal to control an electromagnet, which in turn connects or disconnects another circuit.

- **To connect a relay module to an Arduino, you will need to connect the following pins:**

  - VCC: Connect this pin to the Arduino's 5V power supply.
  - GND: Connect this pin to the Arduino's ground.
  - IN: Connect this pin to an Arduino digital output pin.
  - NO: Connect this pin to the load that you want to control.
  - COM: Connect this pin to the ground.

Once you have connected the relay module to the Arduino, you can use the Arduino's digital output pin to control the load. For example, if you want to turn on a light, you would set the digital output pin to HIGH. This would send a signal to the relay module, which would close the switch and turn on the light.

⇨Here is how to connect a relay module to an Arduino:

The Relay module has six pins. The six pins on the two cubes are connected to a high voltage, and the pins at the bottom connect the module to the Arduino pins.

✓ Why is the carryover used?

We cannot connect a continuous current engine directly with the Arduino, because the engine needs a high current that the Arduino cannot give it. So we use the relay as an interface between the Arduino, which operates in small currents, and the engine that needs high currents.

2.8.5.2 Types of relay modules:

1. **Single channel:**

The 3.3V relay module is a less common type of the switching device. It's designed for use with low-voltage circuits, such as those using 3.3V logic levels. These include ESP8266 and ESP32-based microcontrollers.

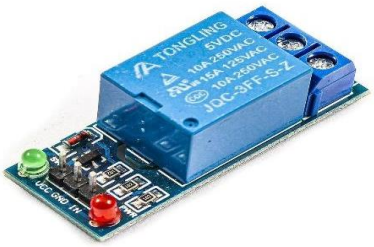


Figure 35.Single channel

2. **Dual channel:**

The 5V relay module is designed for use in circuits that use 5V logic levels. This type of voltage is common in Arduino and Raspberry Pi applications. These provide 5V control signals, which is high enough to activate the 5volt relay module.

Figure 36.Dual channel

3. **Quad channel:**

The 12V relay module is designed for use with circuits that use 12V logic levels, such as those used in automotive applications. Just like with other relays, the 12volt relay module can be used to switch both AC and DC voltages.

Figure 37.Quad channel

4. **Octal channel:**

The 24V relay module is designed for use with higher voltage circuits, such as those that use 24V logic levels. This type of voltage is often used in industrial applications. These include process control, machine automation, and power control systems.

Figure 38.Octal channel

3. IOT Home Automation(IHomA)

3. IOT Home Automation (IHomA)

3.1 Introduction

This chapter is about the design phase of the proposed system and describes the design methods and techniques used to develop the system. This chapter contains the design of systems from many contexts using his UML diagrams such as class diagrams, use case diagrams, and sequence diagrams. This chapter provides an overview of the implementation details of the system, including the implementation environment, hardware and software requirements, development methodology, tools, and technologies used to develop the application, and the software architecture of the proposed system.

3.2 Requirement Gathering and Analysis

3.2.1 Functional requirements:

1. **Open doors using** passcode.
2. **Temperature measurement**
3. **Appliance controlling Like (**Air Conditioner**,** Curtains**,** Boiler**,** Sound system **and** Lighting system**).**
4. **Safety assurance (**Gas **and** Flame**).**
5. **Cameras and surveillance system (**Outdoor **and** Indoor surveillance).

3.2.2 Non Functional requirements

1. Accessibility:

is a crucial aspect of IOT home automation, ensuring that smart home technologies are usable by all individuals, including those with disabilities, the following points highlight the key considerations and features for making IOT home automation systems accessible:

1.1 Voice Control Integration:

Incorporate voice assistants like Amazon Alexa, Google Assistant, and Apple Siri to allow hands-free control of home devices, benefiting users with mobility or visual impairments.

1.2 Remote Control and Monitoring:

Enable remote access to home automation systems via smartphones, tablets, and computers, allowing users to control their home environment from anywhere.

1.3 Customizable Alerts and Notifications:

Provide real-time alerts and notifications through various channels to keep users informed of important events like security breaches or system malfunctions.

2. Availability:

is a critical aspect of IOT home automation systems, ensuring that these systems are consistently operational and accessible to users whenever needed. The following points highlight key considerations and strategies for maintaining high availability in IOT home automation:

2.1 Reliable Hardware:

Use high-quality sensors, microcontrollers, and other hardware components that are durable and have a low failure rate.

2.2 Cloud Services and Data Storage:

Utilize reputable cloud service providers with high uptime guarantees for data storage and remote access.

3. Durability:

is essential for IOT home automation systems to ensure that they remain functional and reliable over time. Durable systems reduce maintenance costs, improve user satisfaction, and provide long-term value. The following points highlight key considerations for enhancing the durability of IOT home automation systems:

3.1 High-Quality Components:

Use robust and high-quality sensors, microcontrollers, and other electronic components that are designed for longevity.

3.2 Environmental Resistance:

Ensure that devices are designed to withstand various environmental conditions, including temperature fluctuations, humidity, dust, and exposure to sunlight.

3.3 Protective Housing:

Design devices with durable housings that can resist physical damage from impacts, drops, and daily wear and tear.

4. Efficiency:

is crucial for maximizing the benefits of smart home technology. It encompasses energy efficiency, system performance, cost-effectiveness, and user convenience. The following points highlight key strategies for enhancing efficiency in IOT home automation systems:

4.1 Energy Efficiency:

- Smart Lighting
- Thermostat Control
- Appliance Management

4.2 Optimized Resource Usage:

- Water Management
- Energy Monitoring

4.3 Data Efficiency:

- Smart Data Management
- Machine Learning

5. Fault Tolerance:

is essential for ensuring continuous and reliable operation even in the face of hardware or software failures. The following points highlight key strategies for enhancing fault tolerance in IOT home automation systems:

5.1 Redundancy:

- Redundant Devices
- Backup Systems

5.2 Testing and Simulation:

- Regular Testing
- Simulation of Failures

➢ **Features:**

1. **Garage Door Control:** Connect a servo motor to the Raspberry Pi to control the garage door. You can use the Raspberry Pi's GPIO pins to interface with the servo motor and control its movements.

2. **Wi-Fi Connectivity:** The Raspberry Pi can be connected to your home Wi-Fi network to enable remote control of the smart home system through an application.

3. **Mobile Application:** Develop a mobile application that communicates with the Raspberry Pi over Wi-Fi. The app should provide controls for the garage door, windows, lights, curtains, and other smart gadgets. It should also include features like password input, gesture controls, and biometric authentication for added security.

4. **Safety Sensors:** Install fire, gas, and crash sensors throughout the house. Connect these sensors to the Raspberry Pi to monitor and detect any potential hazards. You can use sensors like smoke detectors, gas sensors, and impact sensors to ensure safety.

5. **Remote Surveillance:** Set up cameras around the house for remote surveillance. Connect the cameras to the Raspberry Pi or use IP cameras that can be accessed through the Wi-Fi network. The mobile application can provide live video streaming and control of the cameras.

6. **Fan and AC Control:** Use a 5V relay connected to the Raspberry Pi to control the fan and air conditioning units. Connect a temperature sensor, such as a DHT11 or DHT22, to the Raspberry Pi to monitor the temperature and control the fan and AC accordingly.

7. **Time-Oriented Tasks:** Utilize the Raspberry Pi's internal clock or connect an external real-time clock (RTC) module to schedule time-oriented tasks and actions. This will allow you to automate certain actions based on specific times or schedules.

8. **Additional Sensors:** Incorporate various sensors into the system based on your specific needs. This includes an accelerometer sensor for motion detection, an OLED LCD for displaying information, infrared sensors for presence detection, ultrasonic sensors for distance measurement, and moisture sensors for detecting water level in the soil leaks.

3.3 IOT HOME Automation Design

Figure 39.Design

3.3.1 Hardware Design

3.3.1.1 Selected Hardware component

- **Raspberry Pi 4 Model B**
- **Key pad**
- **Flame sensor**
- **MQ2 gas**
- **Ultrasonic sensor**
- **Relay Module (Octal channel)**
- **Female/Male wires**
- **LEDs**
- **Servo motor**
- **LCD**
- **DHT Sensor**

- **Raspberry Pi 4 Model B** is ideal due to its powerful quad-core ARM Cortex-A72 processor, versatile connectivity options (Gigabit Ethernet, Wi-Fi, Bluetooth 5.0), and multiple memory configurations (2GB, 4GB, 8GB RAM). It features multiple I/O ports (USB 3.0, USB 2.0, HDMI, GPIO pins) for easy integration with sensors and devices. Its compact size, energy efficiency, and strong community support, along with extensive educational resources, make it a practical and cost-effective choice for developing smart home solutions.

Figure 40.Raspberry pi 4 Model B

- **Keypad** is selected essential for secure access control and managing smart devices. Selection criteria include compatibility, security features, ease of use, and durability. Popular options include Z-Wave, Wi-Fi, and Bluetooth keypads. They can control smart locks, lighting, alarm systems, and activate automation scenes efficiently.

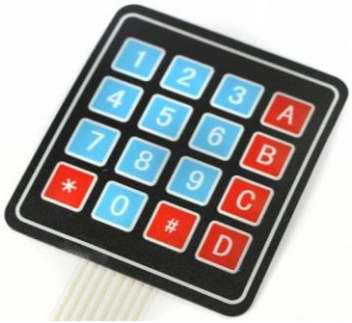


Figure 41.Keypad

- **Flame sensor** is ideal due to their ability to detect fire quickly and accurately through high sensitivity to infrared light. They offer fast response times, low power consumption, and easy integration with other IOT devices for centralized monitoring.

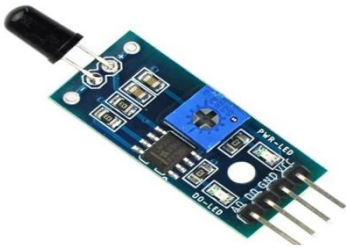


Figure 42.Flame sensor

- **MQ2 gas** is ideal due to its ability to detect multiple gases like methane, propane, and smoke with high sensitivity and fast response times. It offers easy integration with various microcontrollers, both analog and digital outputs, and is cost-effective. Its compact size and extensive community support further enhance its practicality for comprehensive home safety monitoring.

Figure 43.MQ2 gas

- **Ultrasonic sensor** is ideal due to their accurate distance measurement and non-contact sensing capabilities. These sensors are easy to integrate with microcontrollers, energy-efficient, cost-effective, and versatile

Figure 44.Ultrasonic Sensor

- **Relay Module (Octal channel)** is ideal due to its ability to control multiple devices simultaneously with eight channels. It handles high current loads, offers electrical isolation for safety, and is compatible with various microcontrollers like Arduino and Raspberry Pi. It's cost-effective, easy to use, and suitable for applications such as lighting control, motor control, and home security systems.

Figure 45.Relay Module Octal Channel

- **Female/Male wires** is selected because they are designed to provide a secure and reliable electrical connection, ensuring that power can be transmitted safely from the power source to the device.

Figure 46.Female/Male wires

- **LEDs** is selected due to their energy efficiency, long lifespan, and low heat emission. They offer instant lighting, color versatility, and immobility, enhancing ambiance and control. LEDs integrate seamlessly with IOT systems for smart control and automation.

Figure 47.LEDs

- **Servo motor** is selected control of mechanical systems like door locks, window blinds, and robotic arms. They offer accurate positioning and smooth operation, essential for tasks requiring fine adjustments.

Figure 48.Servo motor

- **LCD** is selected for their ability to provide clear visual feedback and control interfaces for various smart devices. They display important information like system status, sensor readings, and user options.

Figure 49.LCD

- **DHT Sensor** is selected due to its ability to accurately monitor temperature and humidity. It is cost-effective, energy-efficient, and easy to integrate with microcontrollers like Arduino and Raspberry Pi. Its compact size allows for flexible installation, and it offers reliable performance with extensive community support and documentation.

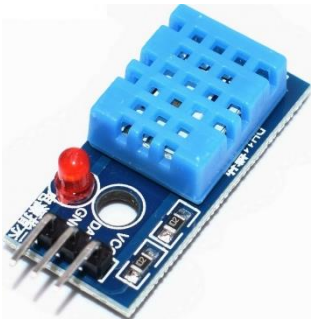


Figure 50.DHT Sensor

3.3.2 Software Design

3.3.2.1 ERD

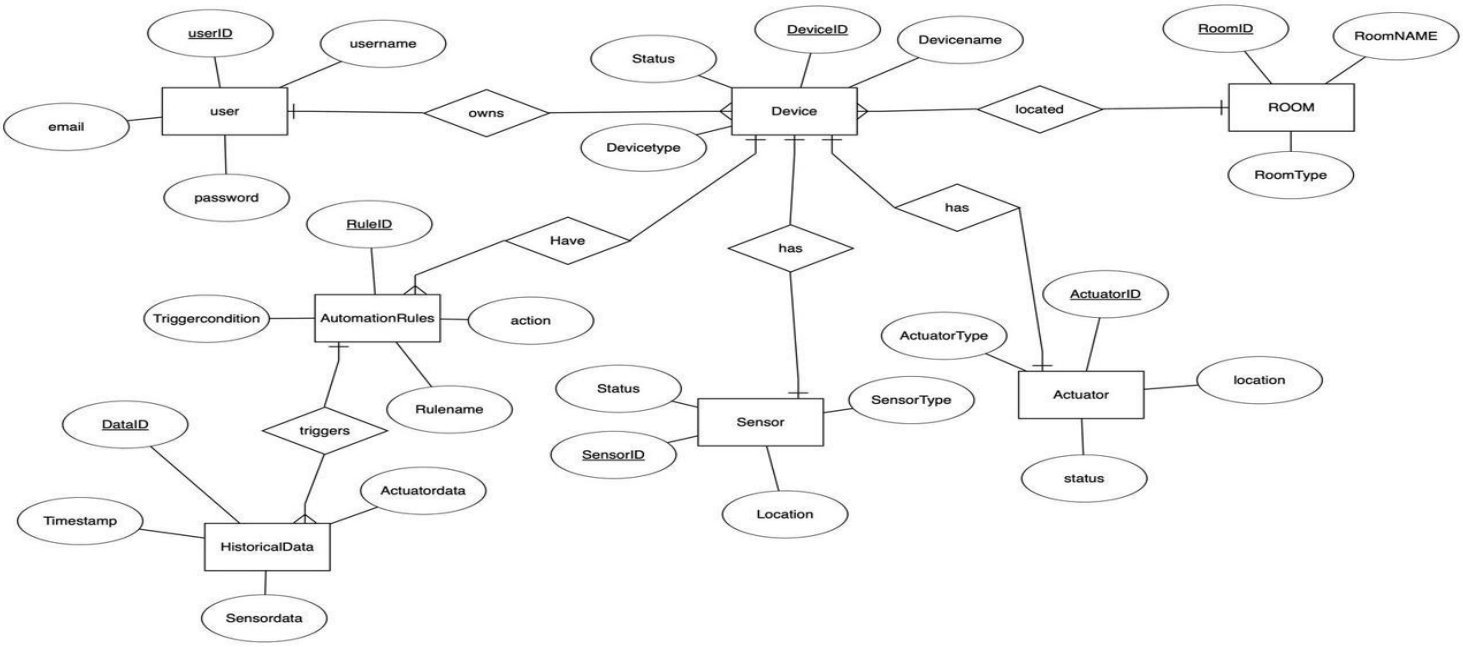


Figure 51.ERD

3.3.2.2 Schema

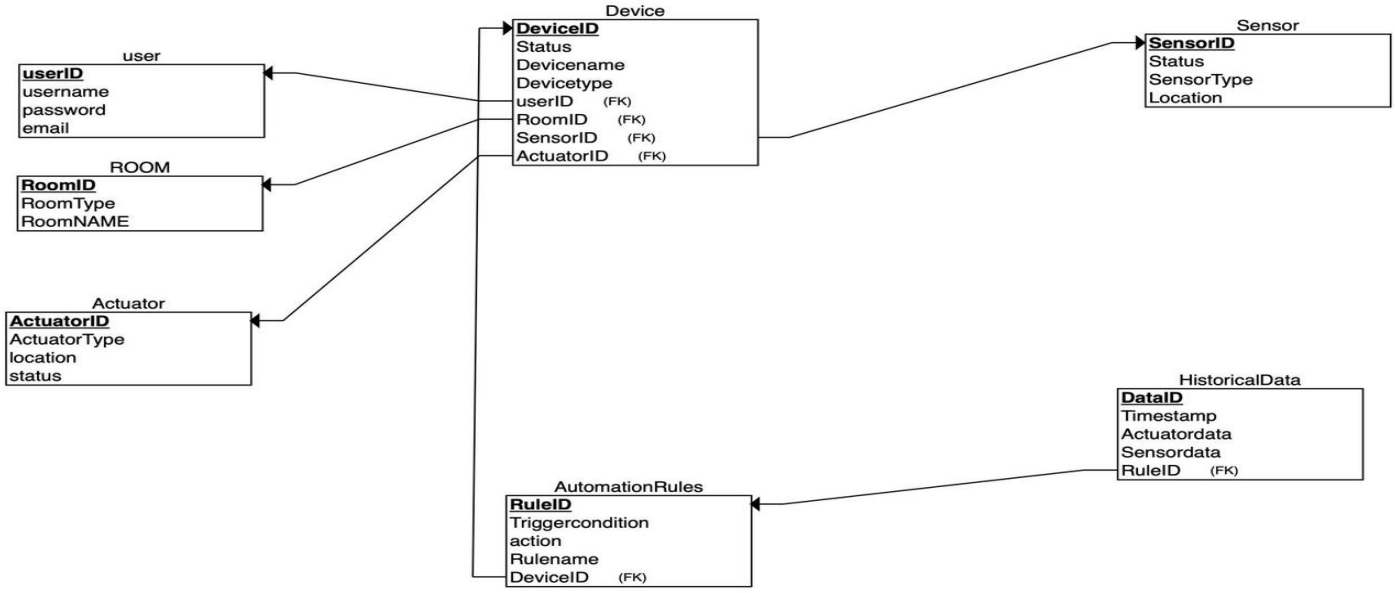


Figure 52.Schema

3.3.2.3 Use Case Diagram

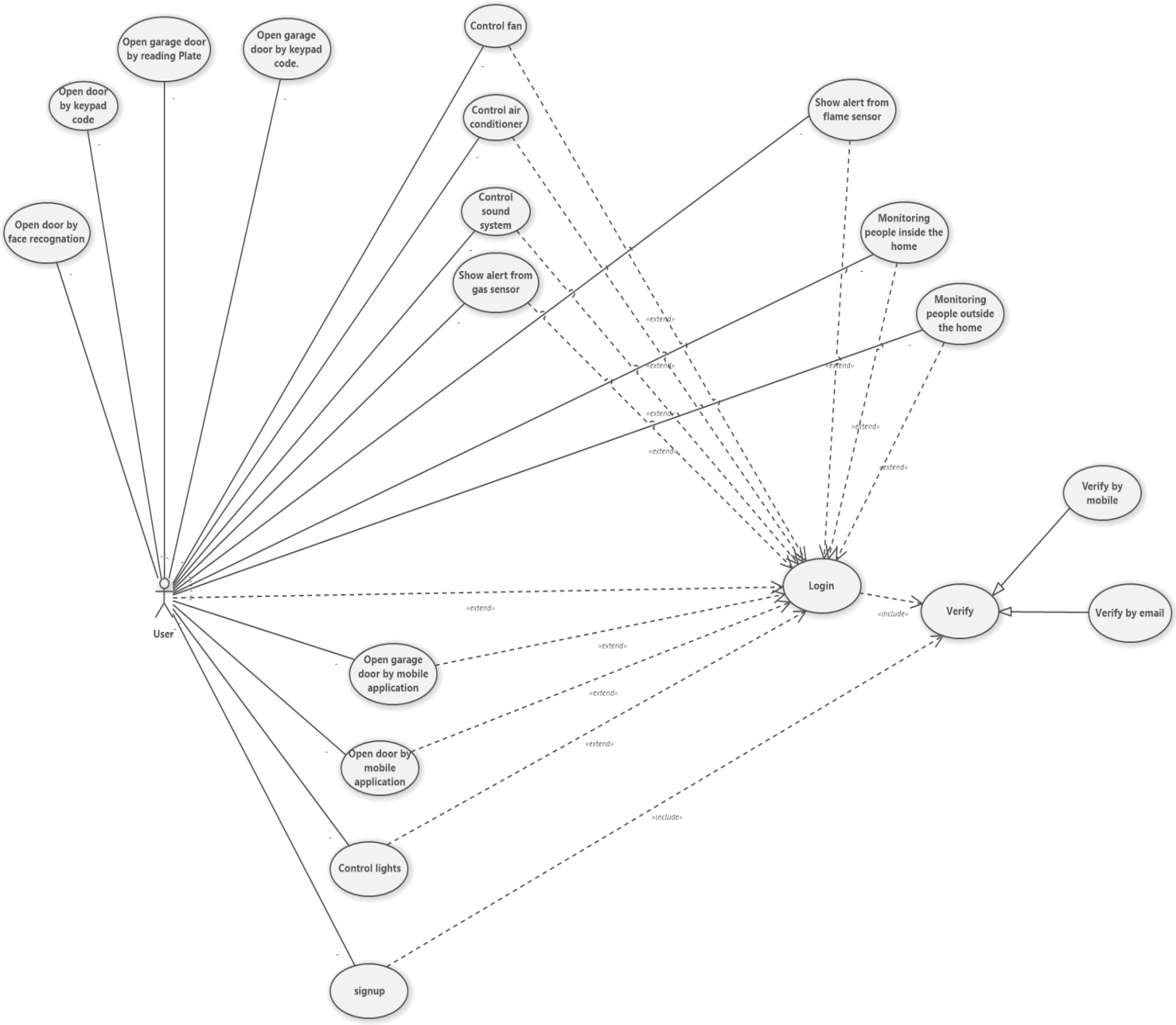


Figure 53.Use case diagram

3.3.2.4 Class diagram

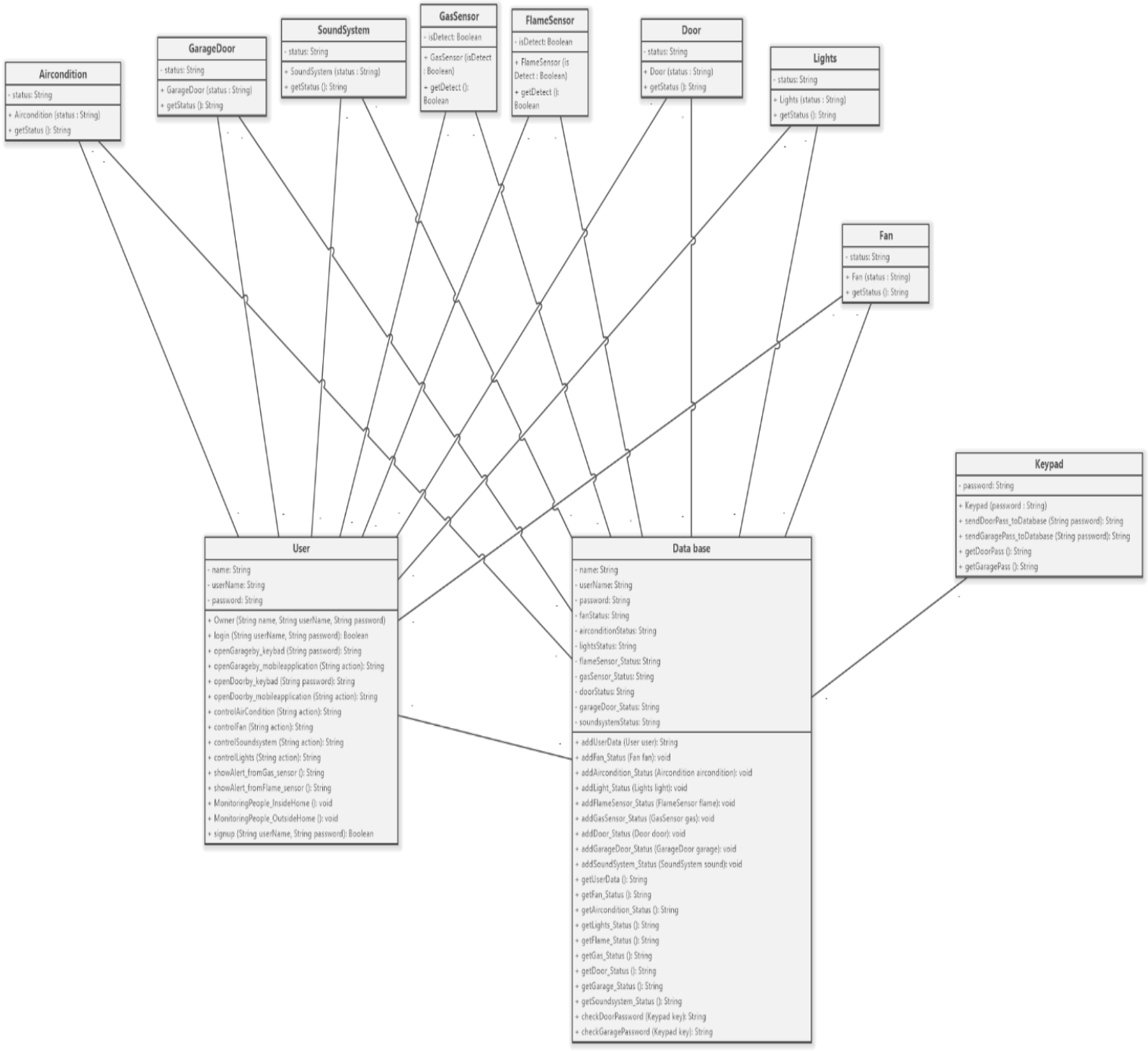


Figure 54.Class diagram

3.3.2.5 Sequence Diagram

- **Signup:**

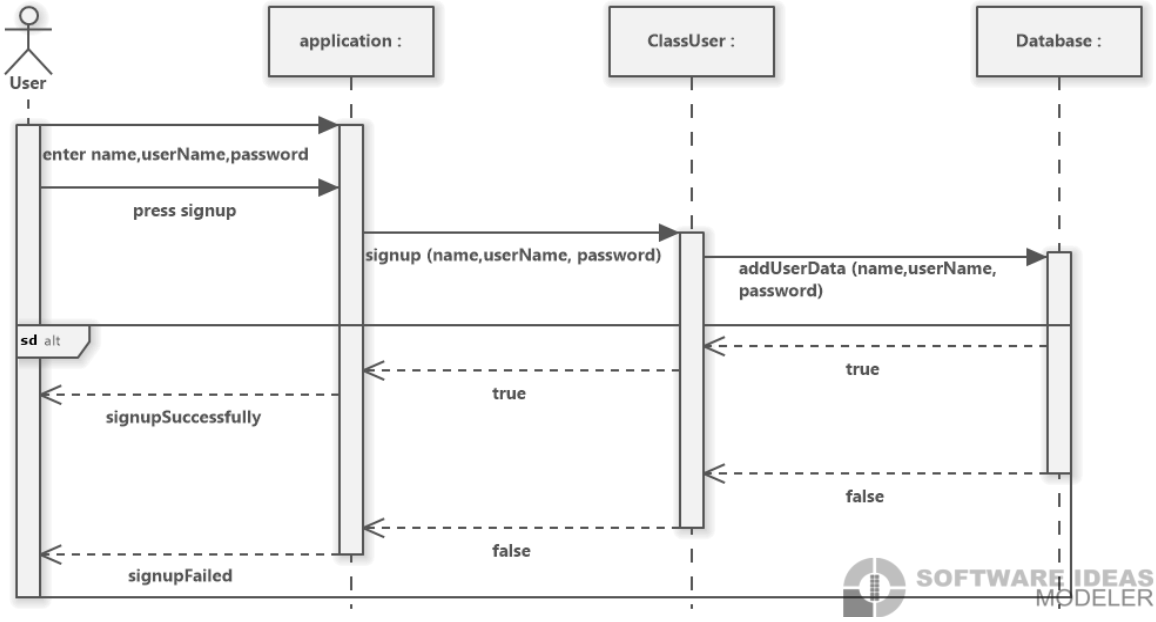


Figure 55.Signup

- **Login:**

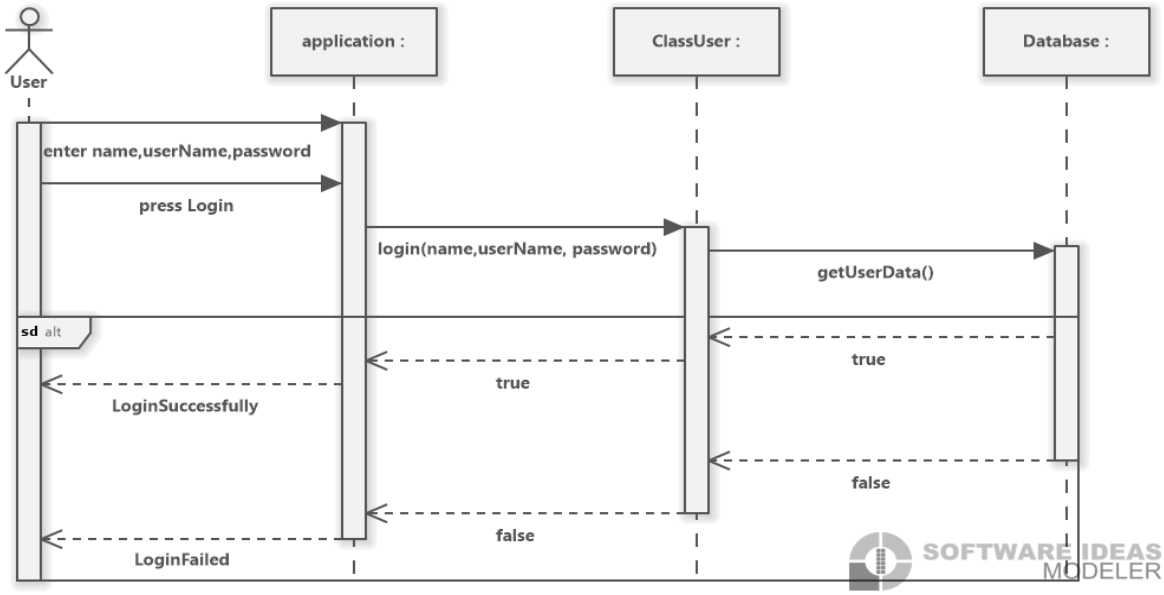


Figure 56.Login

Open garage door by keypad

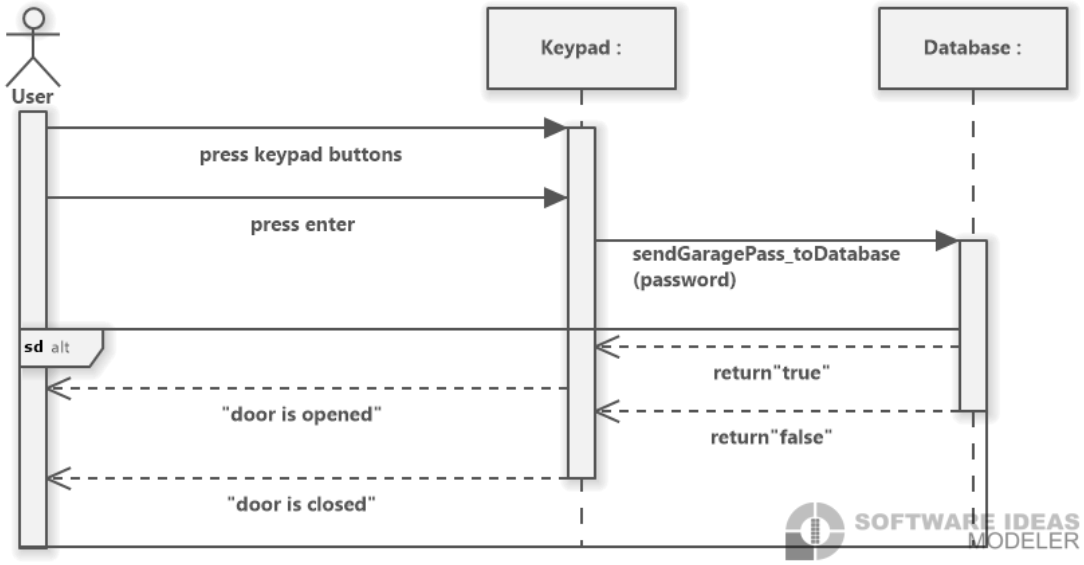


Figure 57.Open garage door by keypad

Open garage door by mobile application

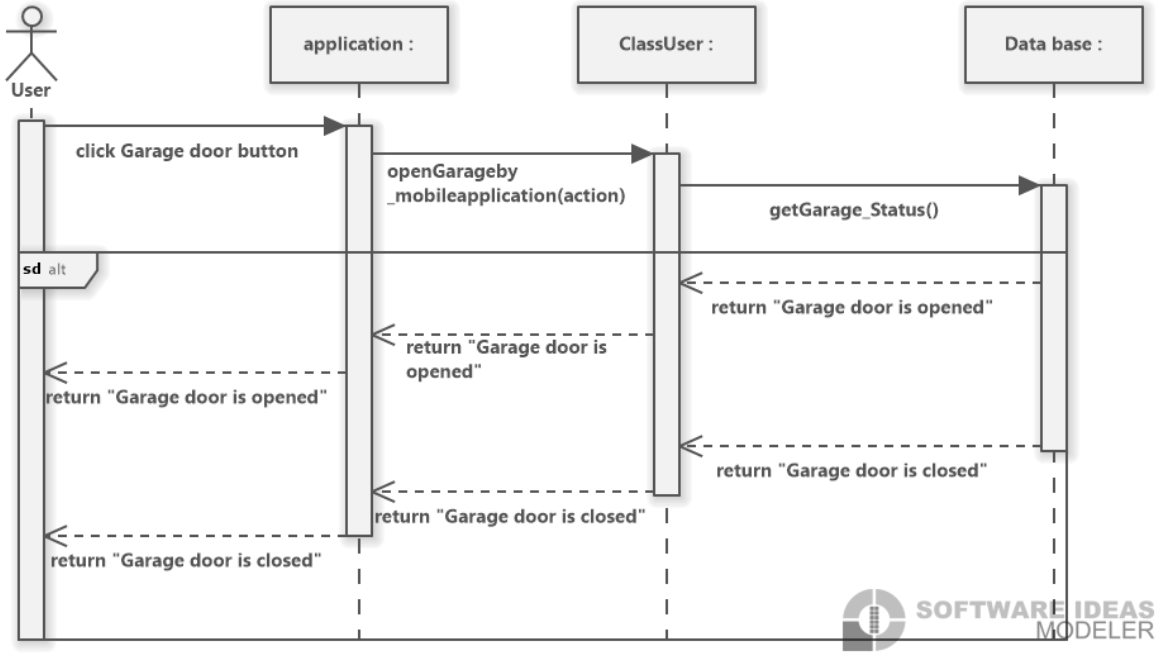


Figure 58. Open garage door by mobile application

Open door by keypad

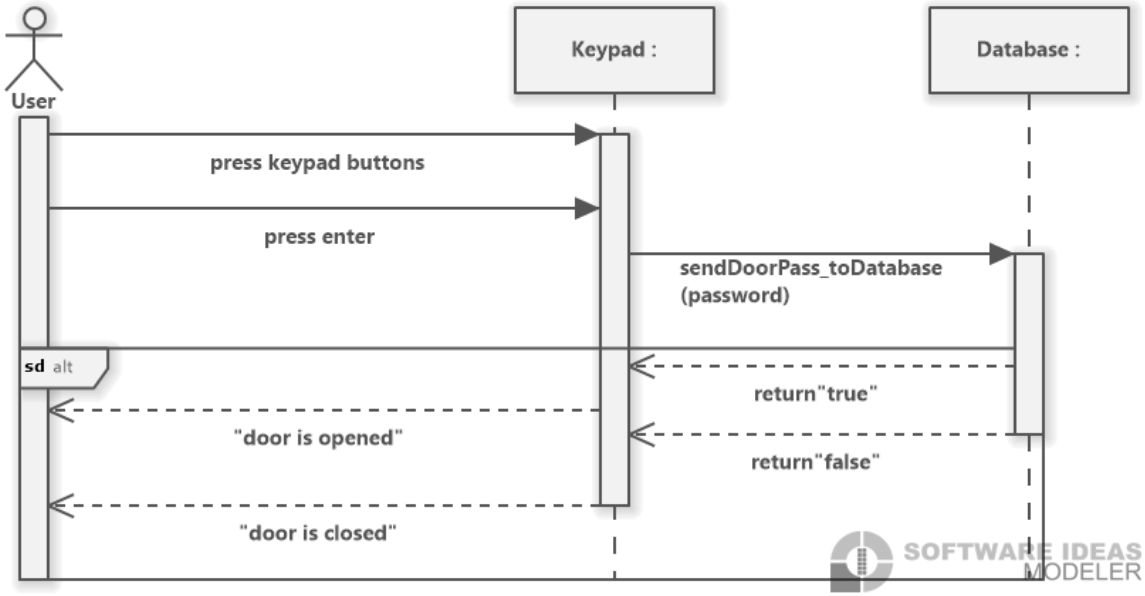


Figure 59. Open door by keypad

Open door by mobile application

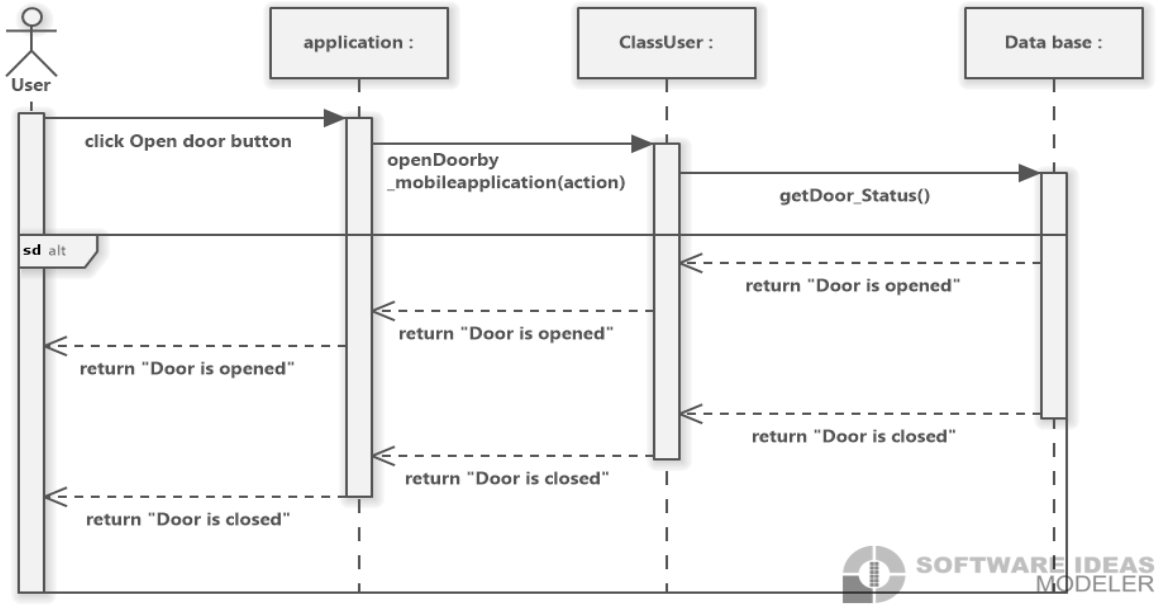


Figure 60. Open door by mobile application

Control light

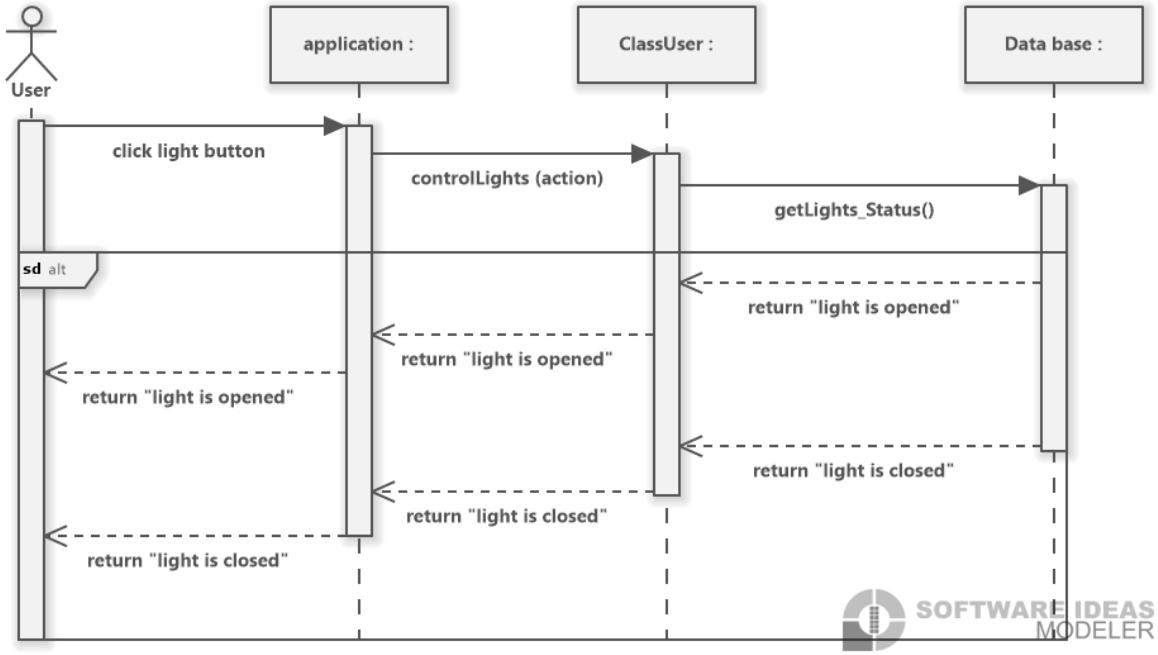


Figure 61. Control light

Control fan

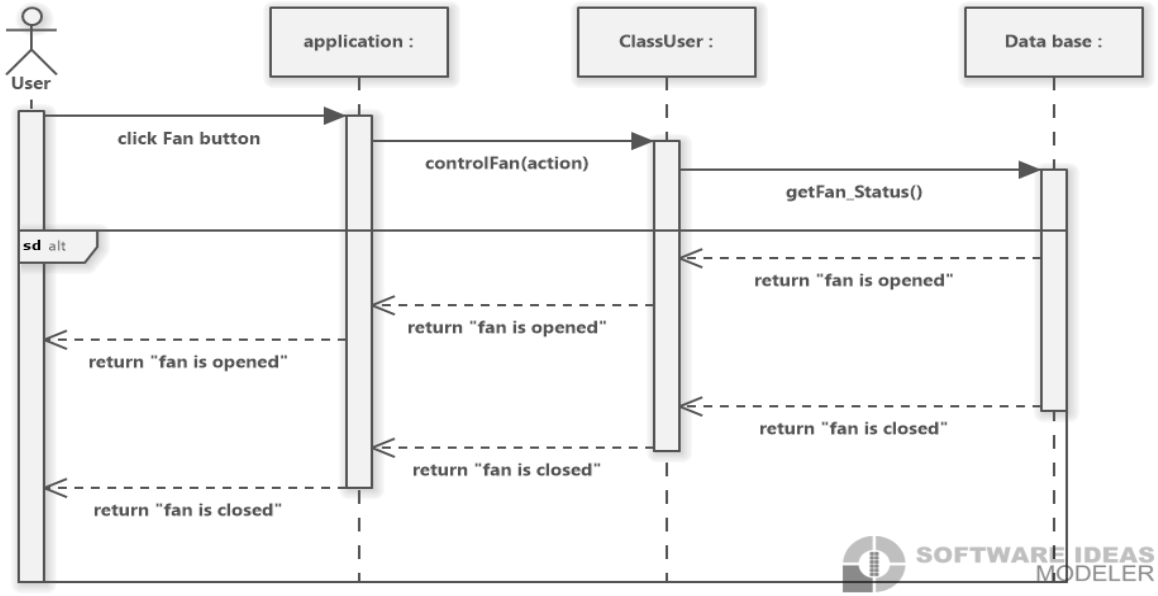


Figure 62. Control fan

Control air-condition

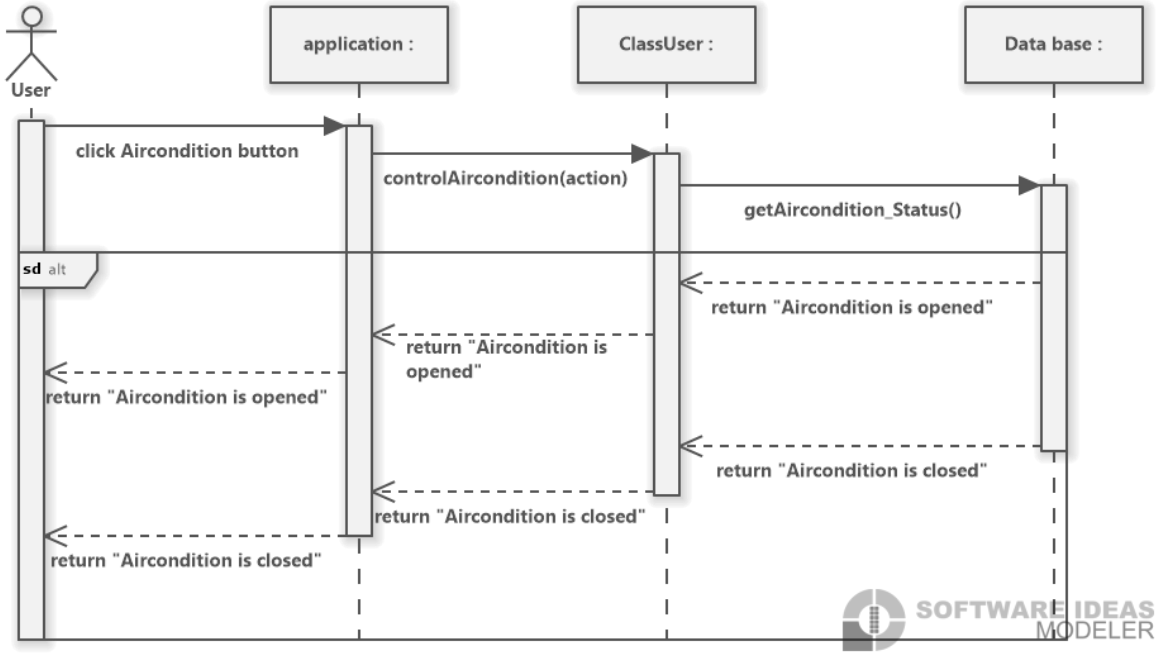


Figure 63. Control air-condition

Control sound system

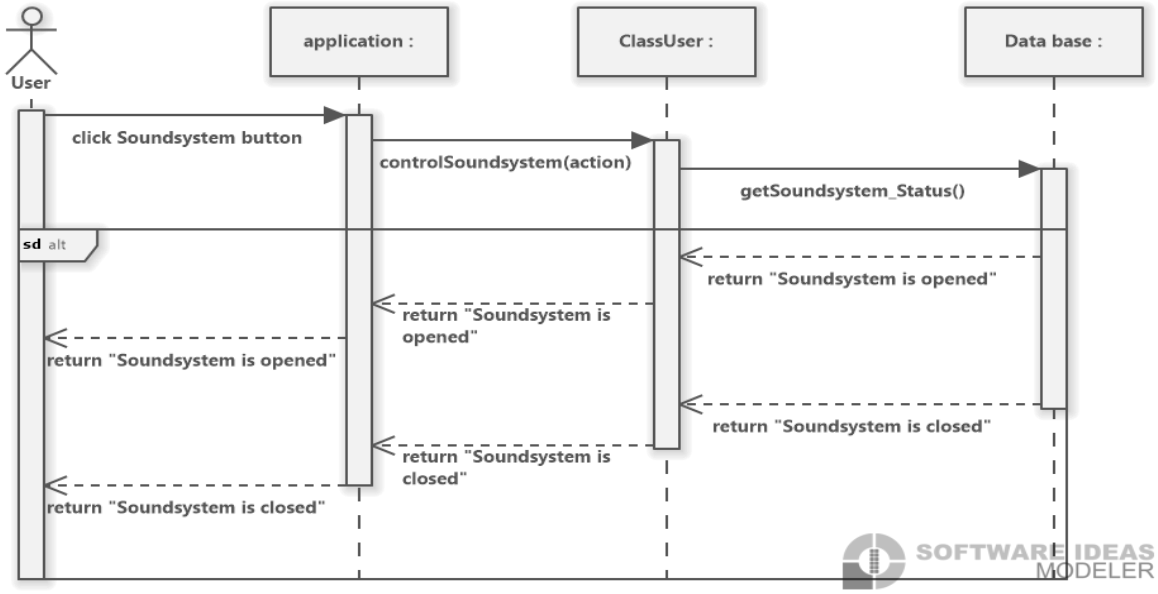


Figure 64. Control sound system

Indoor camera

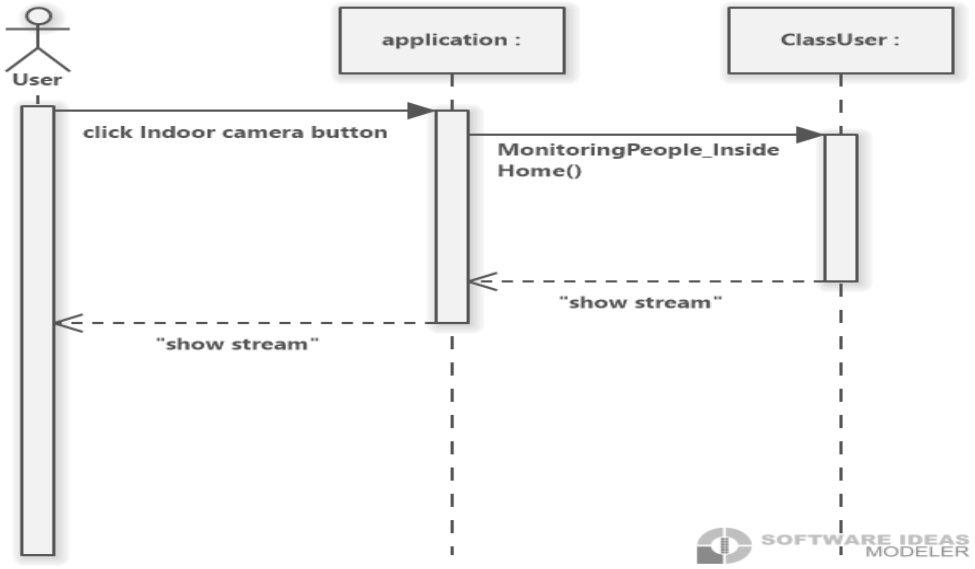


Figure 65. Indoor camera

Outdoor camera

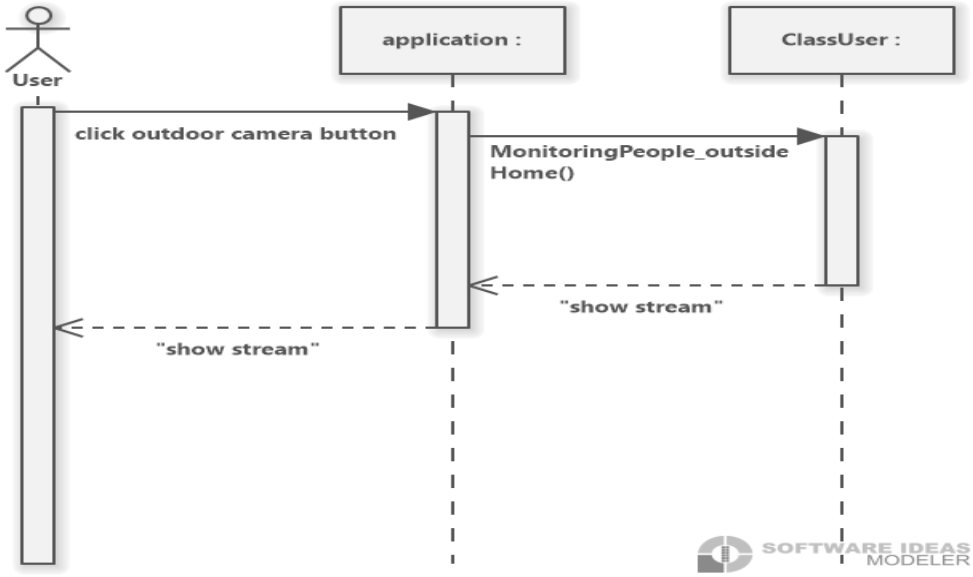


Figure 66. Outdoor camera

3.3.2.6 Flowchart

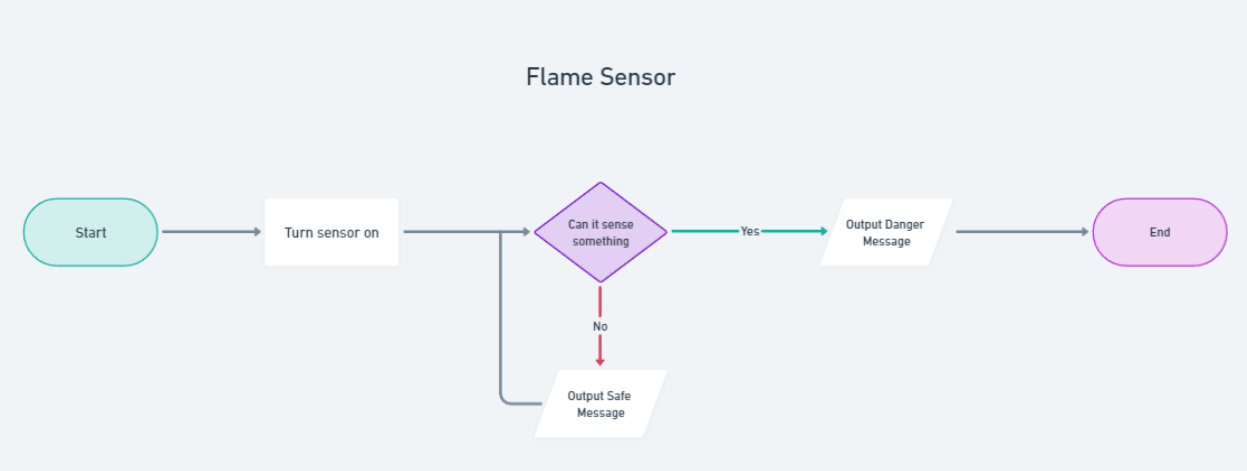


Figure 67.Flame Sensor

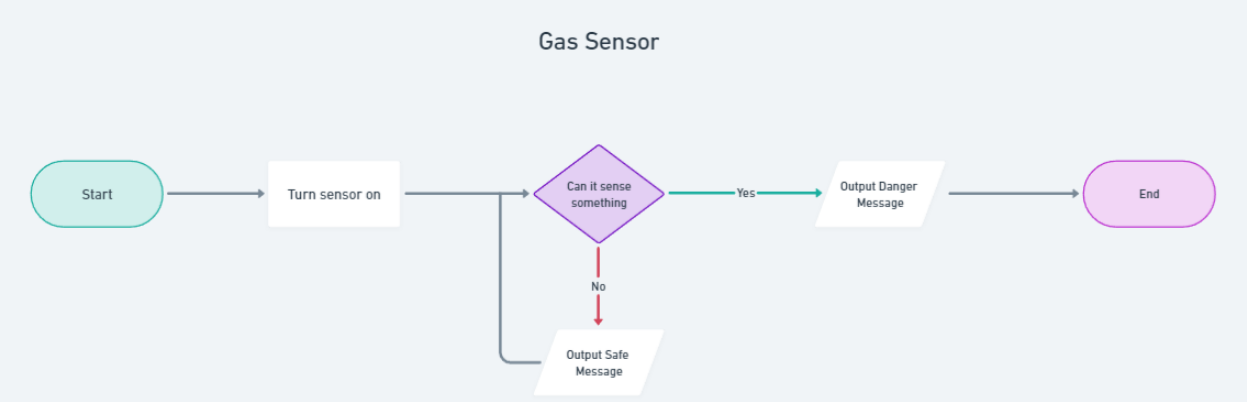


Figure 68.Gas Sensor

3.3.2.7 Block Diagram

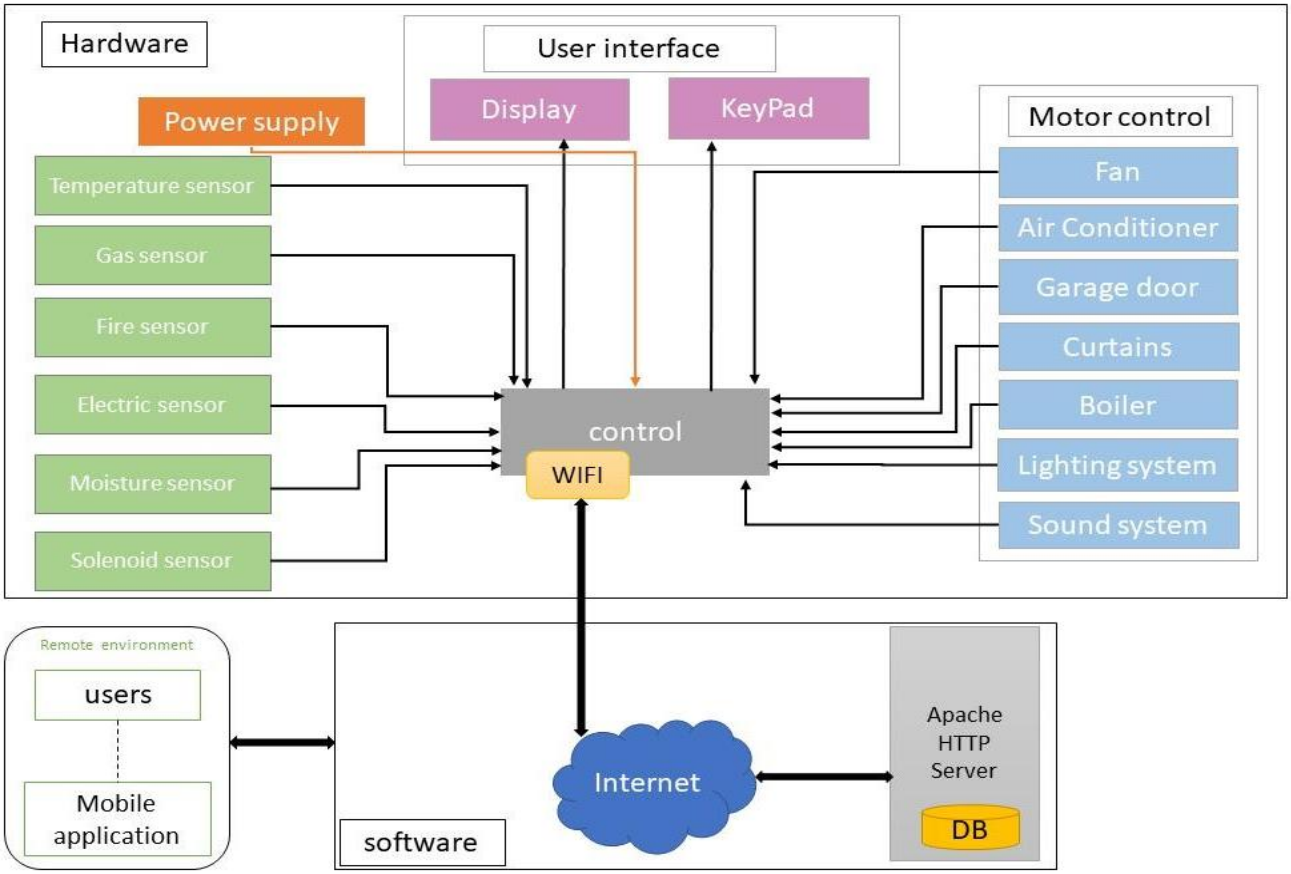


Figure 69.Block Diagram

3.4 Implementation

3.4.1 Hardware implementation

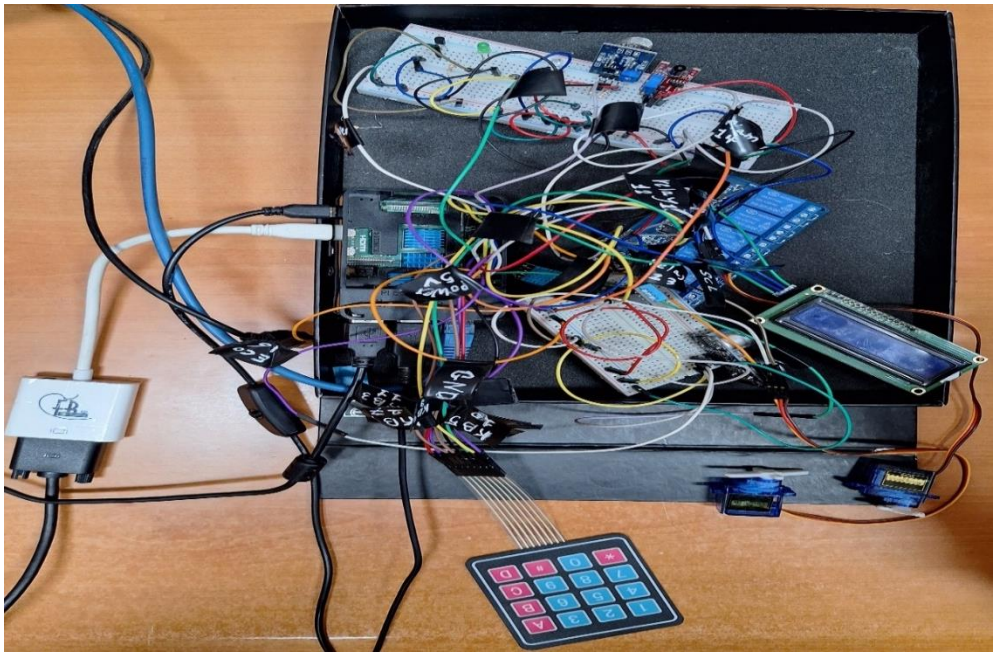


Figure 70.Hardware 1

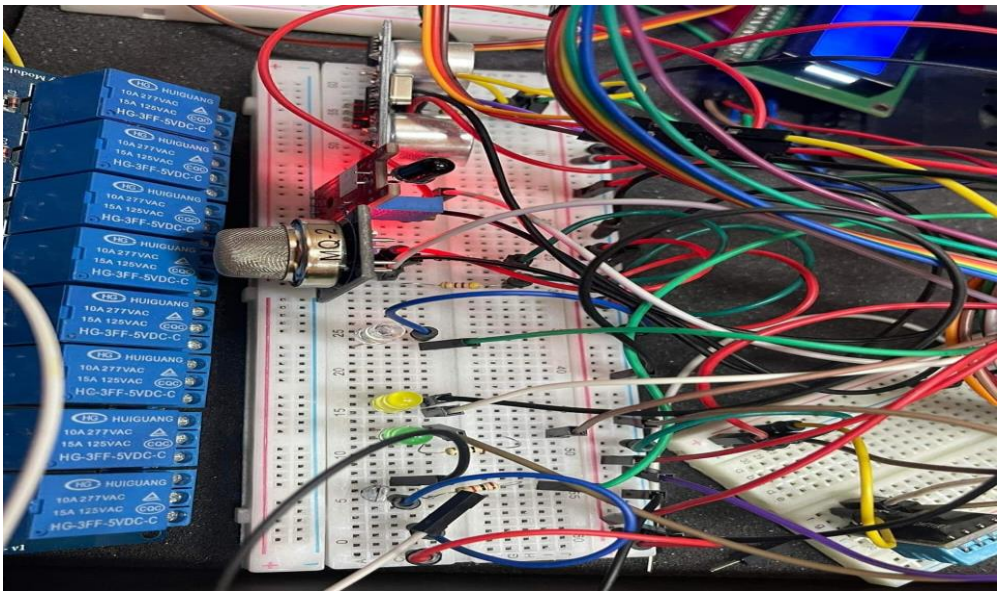


Figure 71.Hardware 2

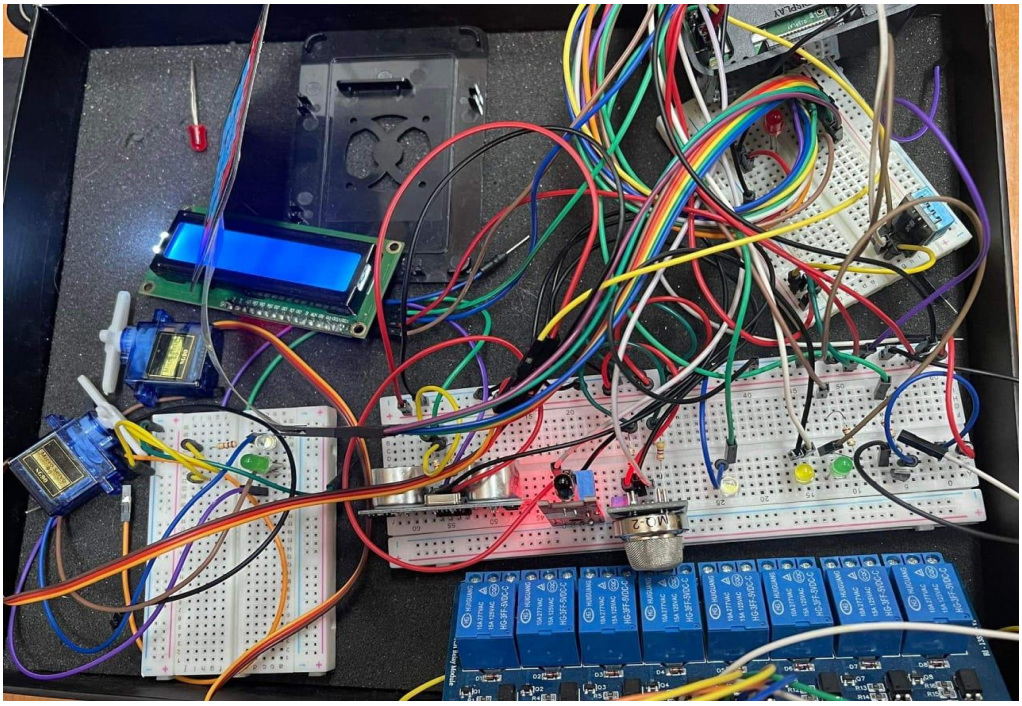


Figure 72.Hardware 3

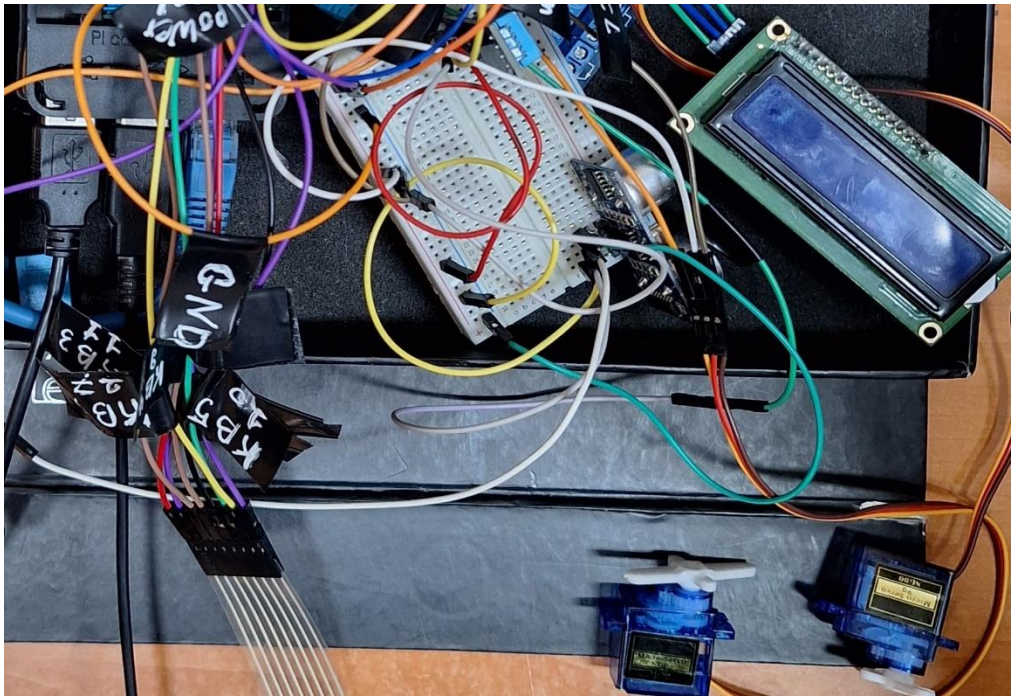


Figure 73.Hardware 4

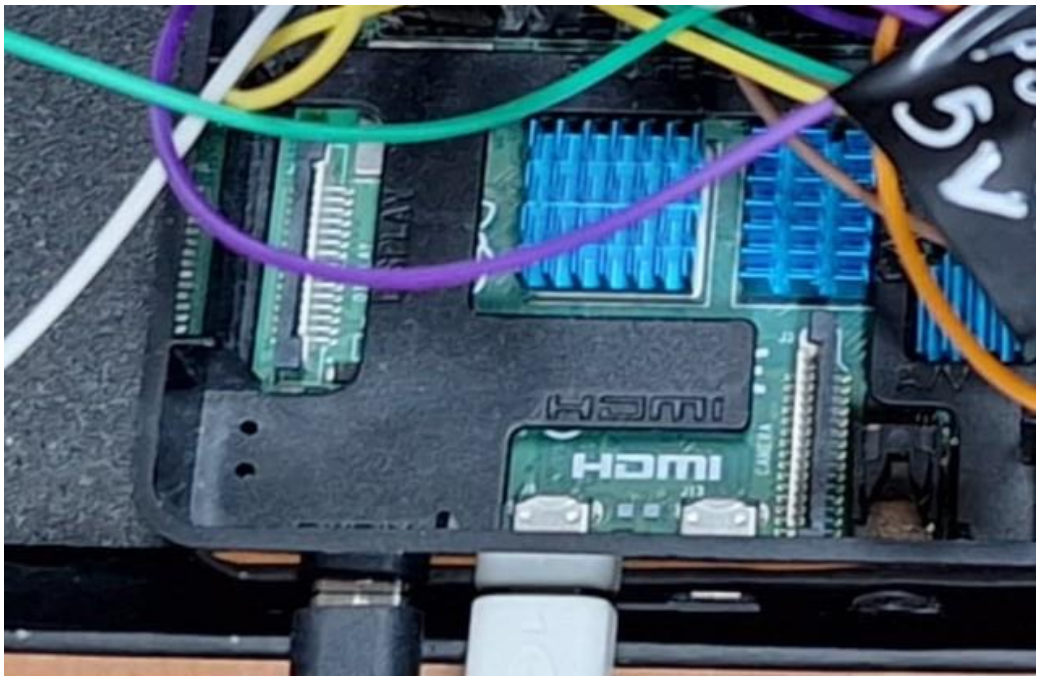


Figure 74.Hardware 5

Figure 75.Hardware 6

3.4.2 Software implementation

⇨ **Wire Temperature Sensor - S18B20:**

a) Code:

```
#! /bin/bash

# Read Temperature
tempread=`cat /sys/bus/w1/devices/10-000802b4ba0e/w1_slave`
# Format
temp=`echo "scale=2; "\`echo ${tempread##*=}\`" / 1000" | bc`

# Output
echo "The measured temperature is " $temp "°C"
```

b) Schematic trial:

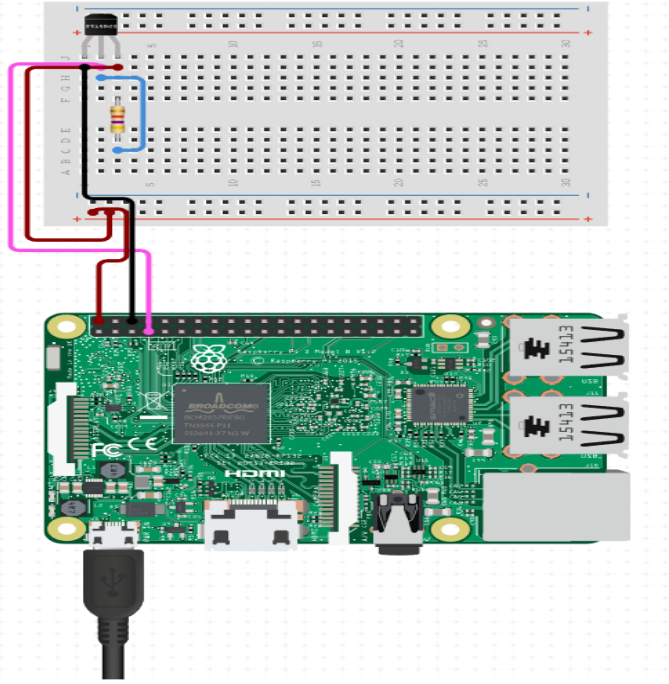


Figure 76.Wire Temperature Sensor - S18B20

⇨ Ultrasonic Sensor - HC-SR04:

a) Code:

```
#Libraries
import RPi.GPIO as GPIO
import time

#GPIO Mode (BOARD / BCM)
GPIO.setmode(GPIO.BCM)

#set GPIO Pins
GPIO_TRIGGER = 18
GPIO_ECHO = 24

#set GPIO direction (IN / OUT)
GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
GPIO.setup(GPIO_ECHO, GPIO.IN)

def distance():
    # set Trigger to HIGH
    GPIO.output(GPIO_TRIGGER, True)

    # set Trigger after 0.01ms to LOW
    time.sleep(0.00001)
    GPIO.output(GPIO_TRIGGER, False)

    StartTime = time.time()
    StopTime = time.time()

    # save StartTime
    while GPIO.input(GPIO_ECHO) == 0:
        StartTime = time.time()

    # save time of arrival
    while GPIO.input(GPIO_ECHO) == 1:
        StopTime = time.time()

    # time difference between start and arrival
    TimeElapsed = StopTime - StartTime
    # multiply with the sonic speed (34300 cm/s)
    # and divide by 2, because there and back
    distance = (TimeElapsed * 34300) / 2
```

```
    return distance

if __name__ == '__main__':
    try:
        while True:
            dist = distance()
            print ("Measured Distance = %.1f cm" % dist)
            time.sleep(1)

        # Reset by pressing CTRL + C
    except KeyboardInterrupt:
        print("Measurement stopped by User")
        GPIO.cleanup()
```

b) Schematic trial:

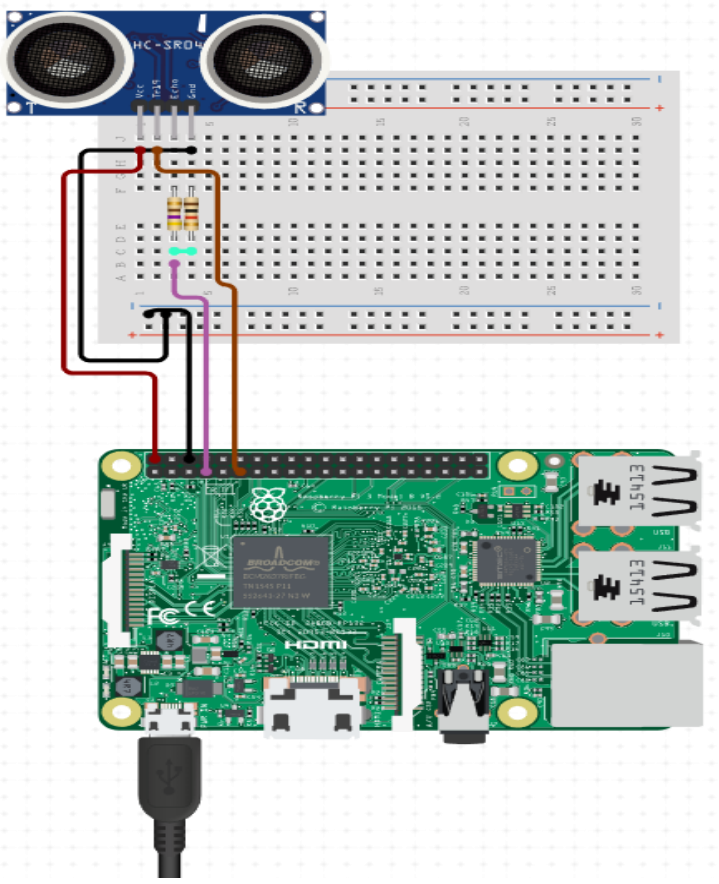


Figure 77.Ultrasonic Sensor - HC-SR04

⇨ Infrared Flame Detection Sensor:

a) Code:

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time

#GPIO SETUP
channel = 21
GPIO.setmode(GPIO.BCM)
GPIO.setup(channel, GPIO.IN)

def callback(channel):
    print("flame detected")

GPIO.add_event_detect(channel, GPIO.BOTH, bouncetime=300)  # let
us know when the pin goes HIGH or LOW
GPIO.add_event_callback(channel, callback)  # assign function to
GPIO PIN, Run function on change

# infinite loop
while True:
        time.sleep(1)
```

b) Schematic trial:

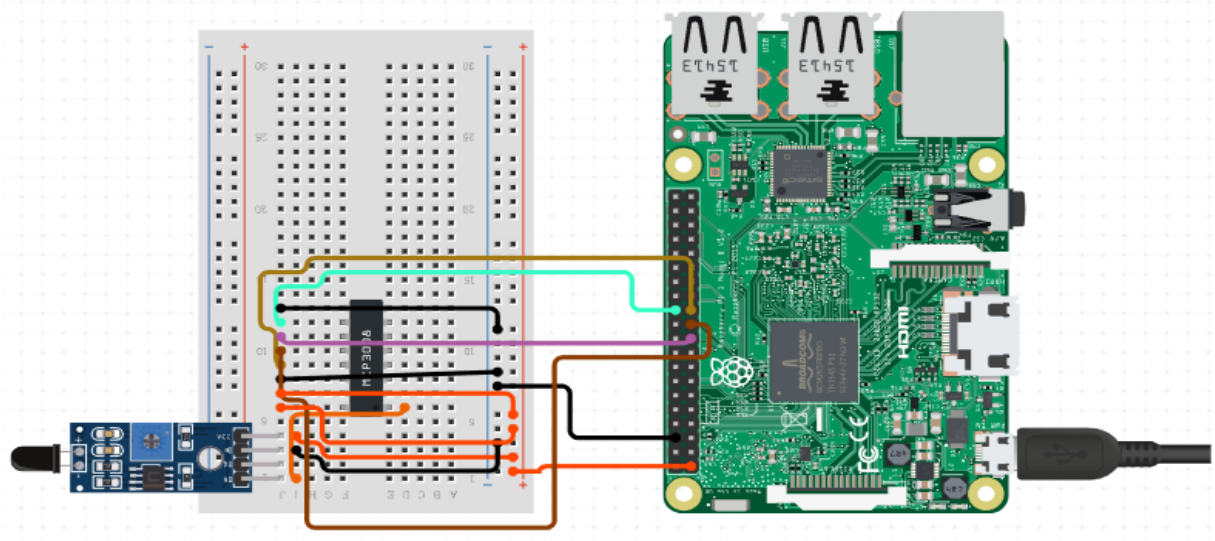


Figure 78.Infrared Flame Detection Sensor

⇨ Gas Sensor - MQ-2:

a) Code:

/*File on system*/

b) Schematic trial:

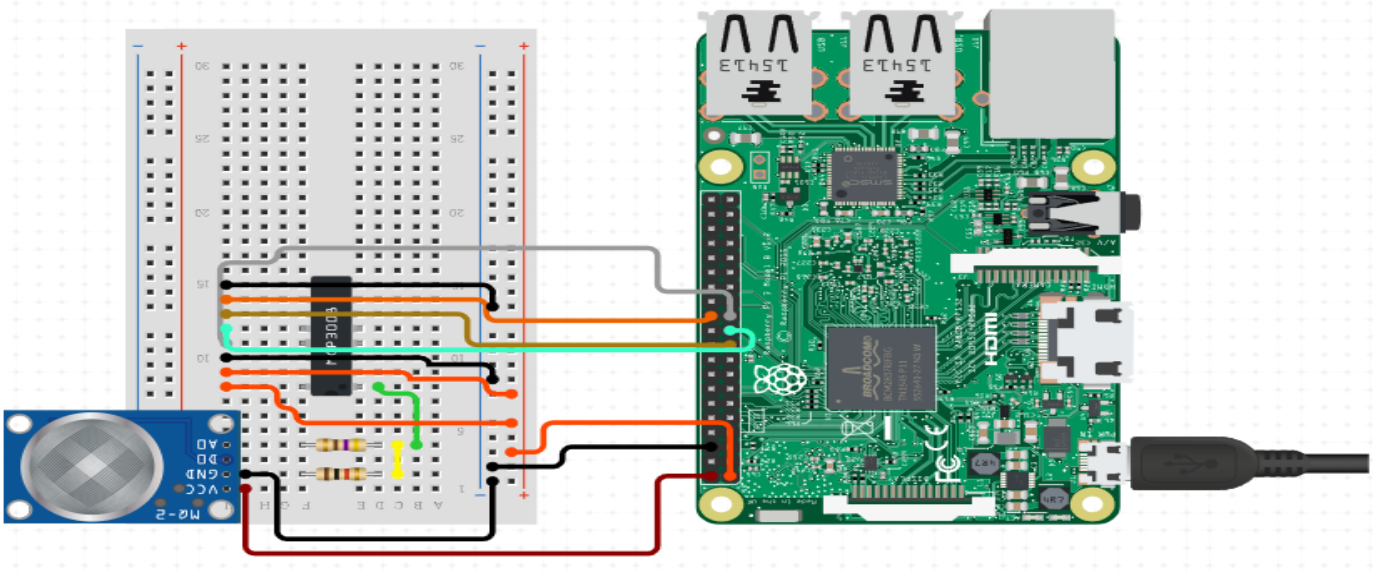


Figure 79.Gas Sensor - MQ-2

⇨ Membrane 3x4 Matrix Keypad:

a) Code:

```
import time
import RPi.GPIO as GPIO
from keypad import keypad

GPIO.setwarnings(False)

if __name__ == '__main__':
    # Initialize
    kp = keypad(columnCount = 3)

    # waiting for a keypress
    digit = None
```

```
while digit == None:
    digit = kp.getKey()
# Print result
print digit
time.sleep(0.5)

###### 4 Digit wait ######
seq = []
for i in range(4):
    digit = None
    while digit == None:
        digit = kp.getKey()
    seq.append(digit)
    time.sleep(0.4)
# Check digit code
print(seq)
if seq == [1, 2, 3, '#']:
    print "Code accepted"
```

b) Schematic trial:

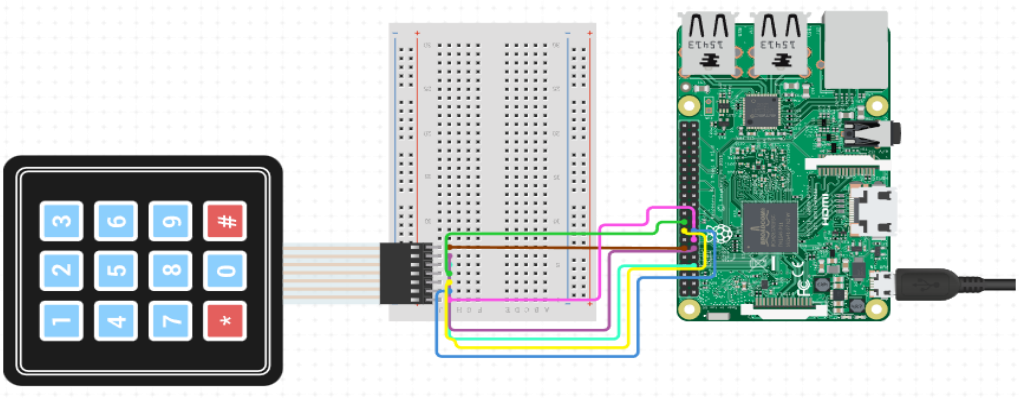


Figure 80.Membrane 3x4 Matrix Keypad

⇨ LCD 16x2 I2C:

a) Code:

```
from signal import signal, SIGTERM, SIGHUP, pause
from rpi_lcd import LCD
lcd = LCD()
def safe_exit(signum, frame):
    exit(1)
try:
    signal(SIGTERM, safe_exit)
    signal(SIGHUP, safe_exit)
    lcd.text("Hello,", 1)
    lcd.text("Raspberry Pi!", 2)
    pause()
except KeyboardInterrupt:
    pass
finally:
    lcd.clear()
```

b) Schematic trial:

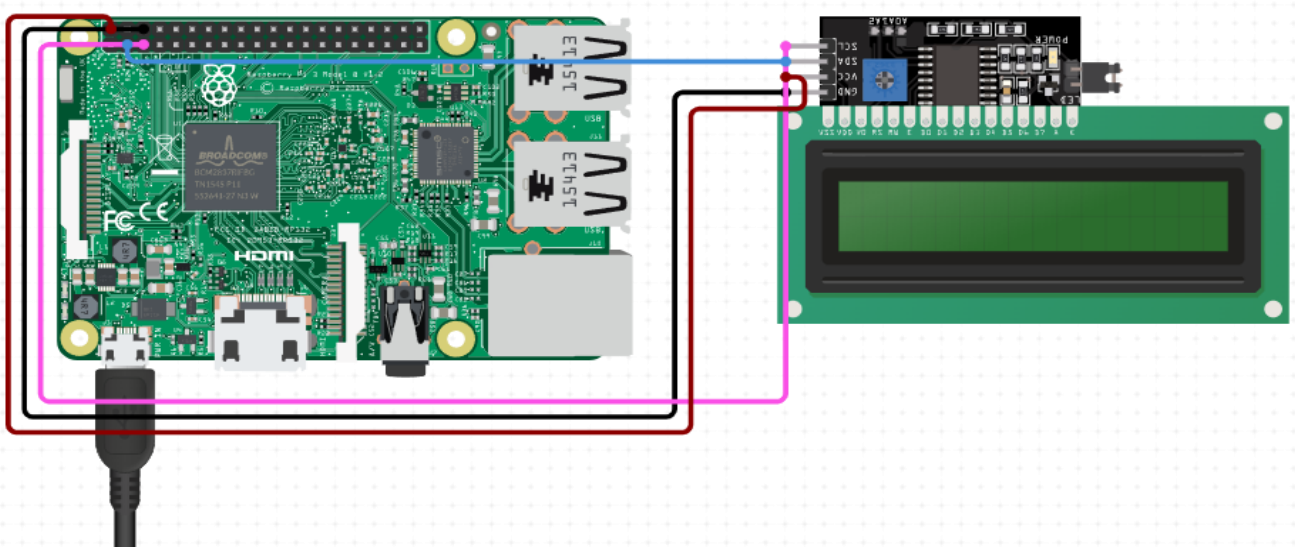


Figure 81.LCD 16x2 I2C

⇨ 9G Micro Servo Motor:

a) Code:

```
# We imports the GPIO module
import RPi.GPIO as GPIO
# We import the command sleep from time
from time import sleep

# Stops all warnings from appearing
GPIO.setwarnings(False)

# We name all the pins on BOARD mode
GPIO.setmode(GPIO.BOARD)
# Set an output for the PWM Signal
GPIO.setup(16, GPIO.OUT)

# Set up the PWM on pin #16 at 50Hz
pwm = GPIO.PWM(16, 50)
pwm.start(0) # Start the servo with 0 duty cycle ( at 0 deg position )
pwm.ChangeDutyCycle(5) # Tells the servo to turn to the left ( -90 deg
position )
sleep(0.5) # Tells the servo to Delay for 5 sec
pwm.ChangeDutyCycle(7.5) # Tells the servo to turn to the neutral
position ( at 0 deg position )
sleep(0.5) # Tells the servo to Delay for 5 sec
pwm.ChangeDutyCycle(10) # Tells the servo to turn to the right ( +90
deg position )
sleep(0.5) # Tells the servo to Delay for 5 sec
pwm.stop(0) # Stop the servo with 0 duty cycle ( at 0 deg position )
GPIO.cleanup() # Clean up all the ports we've used.
```

b) Schematic trial:

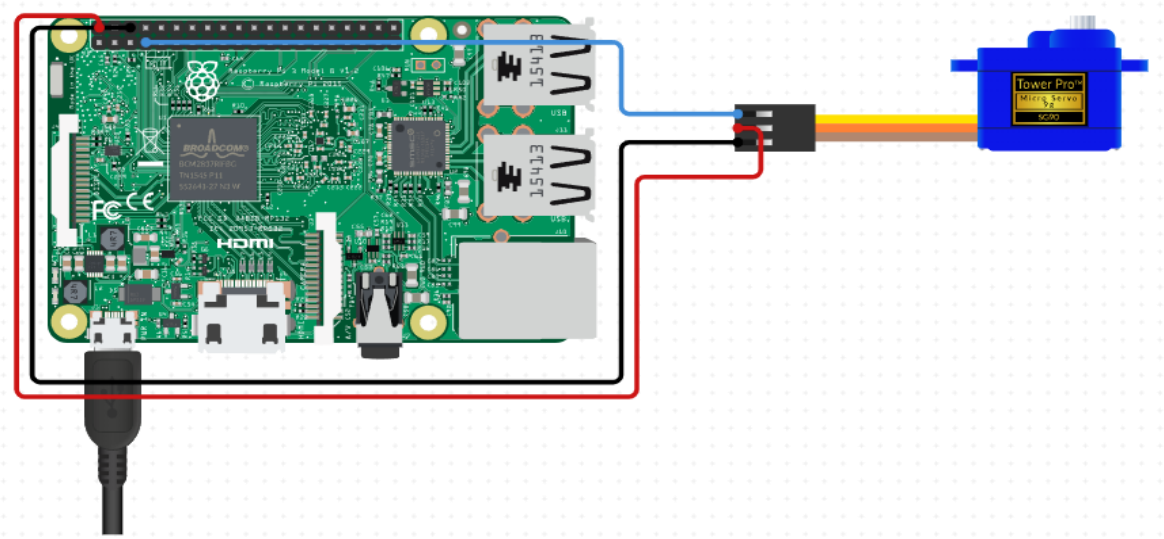


Figure 82.9G Micro Servo Motor

⇨ **Lock-style Solenoid 12VDC:**

a) Code:

```
import RPi.GPIO as GPIO
import time
BUTTON_PIN = 18 # GPIO pin connected to the button
RELAY_PIN = 16 # GPIO pin controlled the solenoid lock via the relay
module
# Set up the GPIO pins
GPIO.setmode(GPIO.BCM)
GPIO.setup(BUTTON_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(RELAY_PIN, GPIO.OUT)
prev_button_state = GPIO.HIGH  # HIGH means the button is not
pressed initially
try:
    # Lock the door initially
```

```
    GPIO.output(RELAY_PIN, GPIO.HIGH)
    while True:
        button_state = GPIO.input(BUTTON_PIN)
        if button_state == GPIO.LOW and prev_button_state ==
GPIO.HIGH:
            # Button is pressed (LOW means pressed due to pull-up
resistor)
            print("The button is pressed")
            GPIO.output(RELAY_PIN, GPIO.LOW)  # Unlock the door
            print("The door is unlocked")
            time.sleep(5)  # Wait for 5 seconds
            GPIO.output(RELAY_PIN, GPIO.HIGH)  # Lock the door again
            print("The door is locked again")
        # Update the previous button state
        prev_button_state = button_state
except KeyboardInterrupt:
    print("Exiting...")
    GPIO.cleanup()
```

b) Schematic trial:

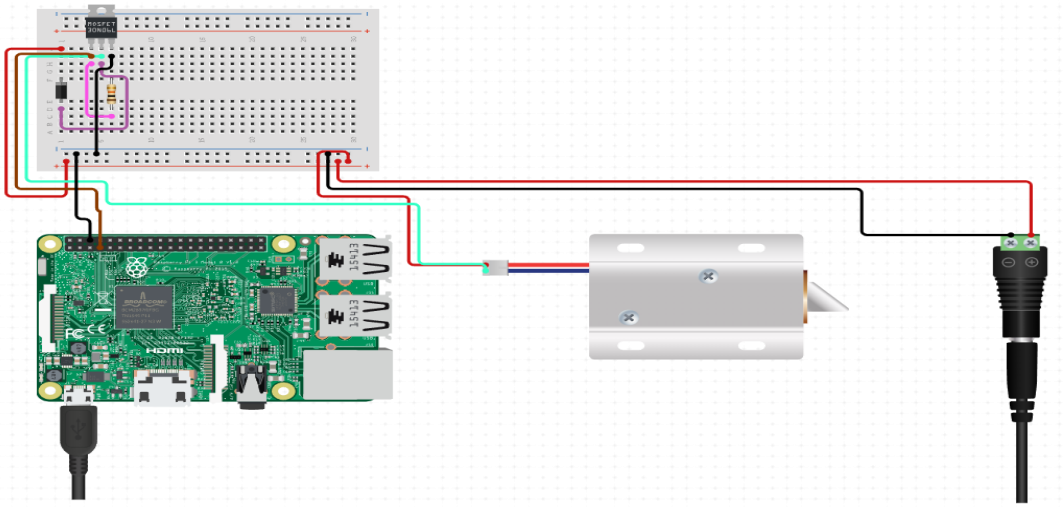


Figure 83.Lock-style Solenoid 12VDC

⇨ **Relay Module 8-Ch:**

a) Code:

b) Schematic trial:

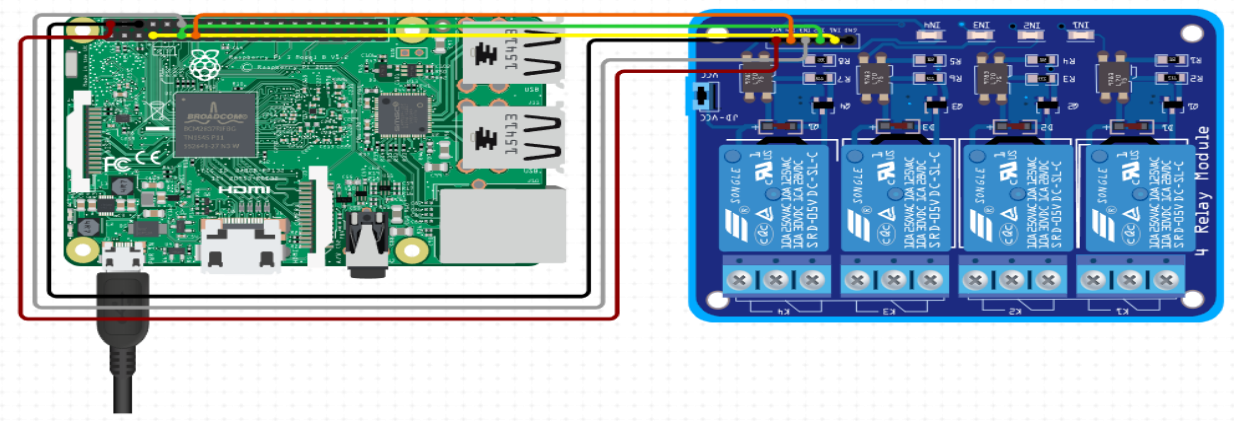
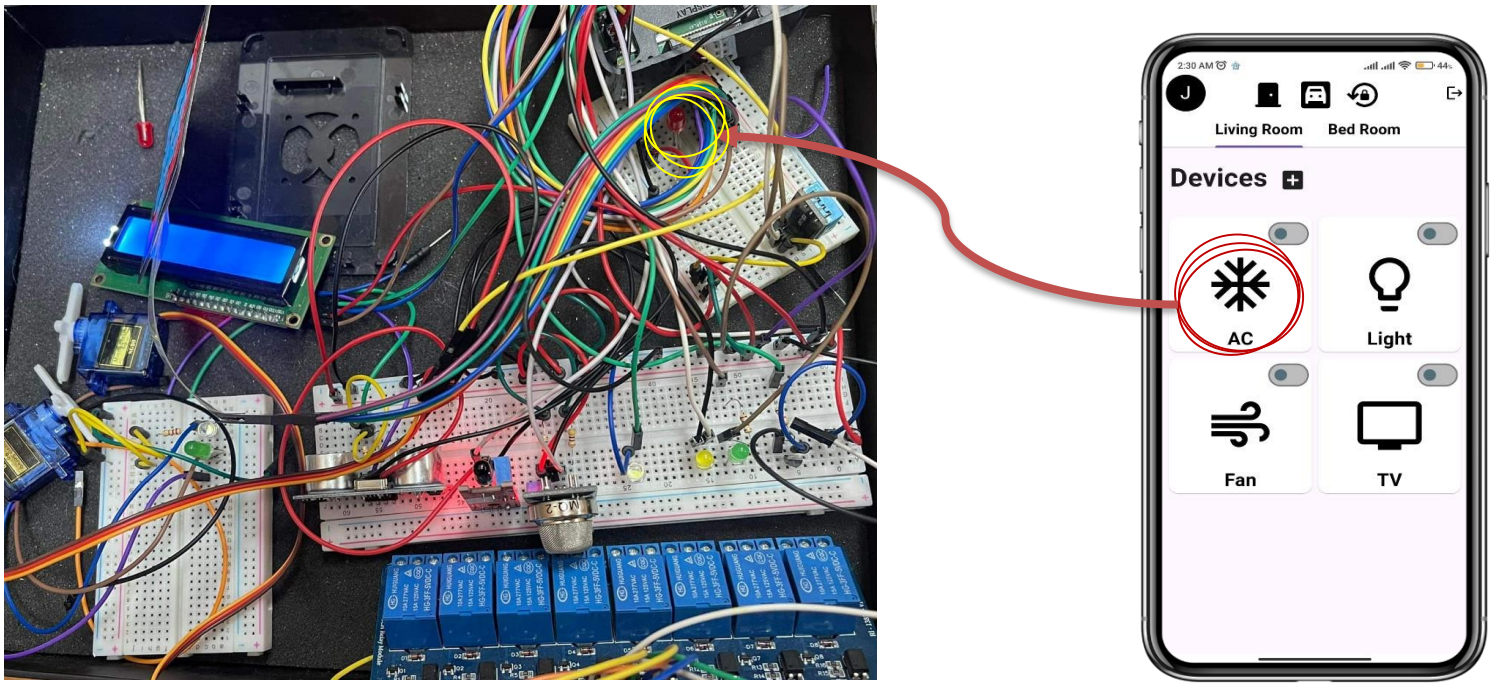


Figure 84.Relay Module 8-Ch

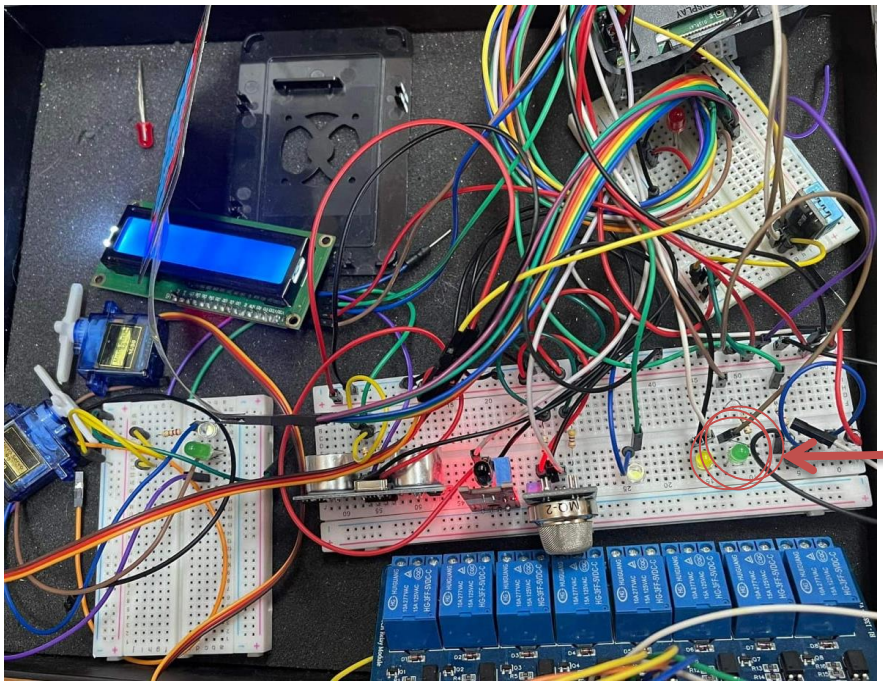
3.4.3 Hardware / Software implementation



⇨ **Code:**

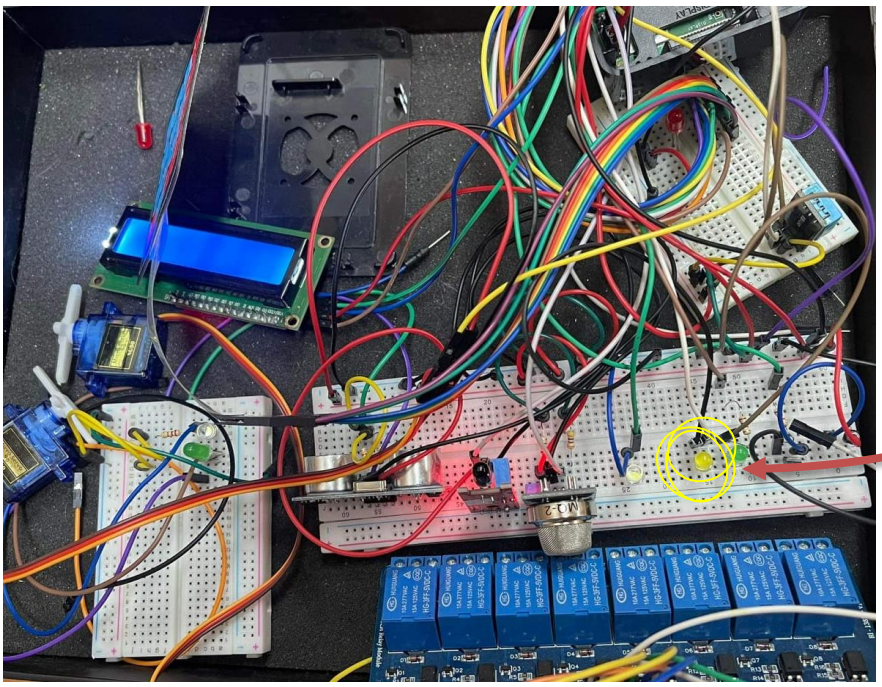
```
import RPi.GPIO as GPIO

import time

from collections.abc import MutableMapping

from firebase import firebase

from time import sleep

url="https://mothakem-8a457-default-rtdb.firebaseio.com/"

firebase = firebase.FirebaseApplication(url)

red_pin = 26

GPIO.setmode(GPIO.BCM)
```

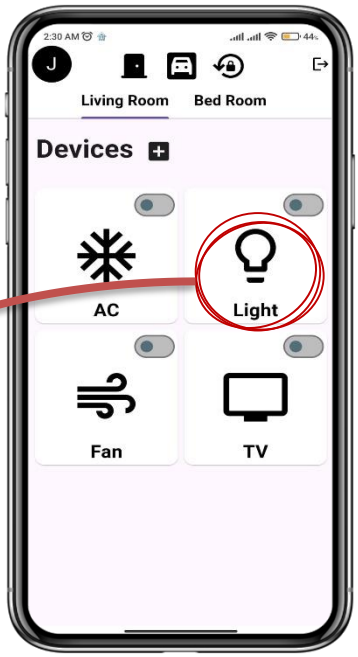
```
GPIO.setup(red_pin, GPIO.OUT)

#======================Servo & Led

d2=firebase.get("/37/servo/servo_action",None)

red=firebase.get("/37/0/Ac_action",None)

if red==True:

print("ai")

GPIO.output(red_pin, GPIO.HIGH)

elif red==False:

 GPIO.output(red_pin, GPIO.LOW)

 print(red)
```



⇨ **Code:**

```
import RPi.GPIO as GPIO

import time

from collections.abc import MutableMapping

from firebase import firebase

from time import sleep

url="https://mothakem-8a457-default-rtdb.firebaseio.com/"

firebase = firebase.FirebaseApplication(url)

green_pin = 21

GPIO.setmode(GPIO.BCM)

GPIO.setup(green_pin, GPIO.OUT)

#======================Servo & Led

d2=firebase.get("/37/servo/servo_action",None)

green=firebase.get("/37/2/Fan_action",None)

if green==True:

print("ai")

GPIO.output(green_pin, GPIO.HIGH)

elif green==False:

GPIO.output(green_pin, GPIO.LOW)

print(green)
```

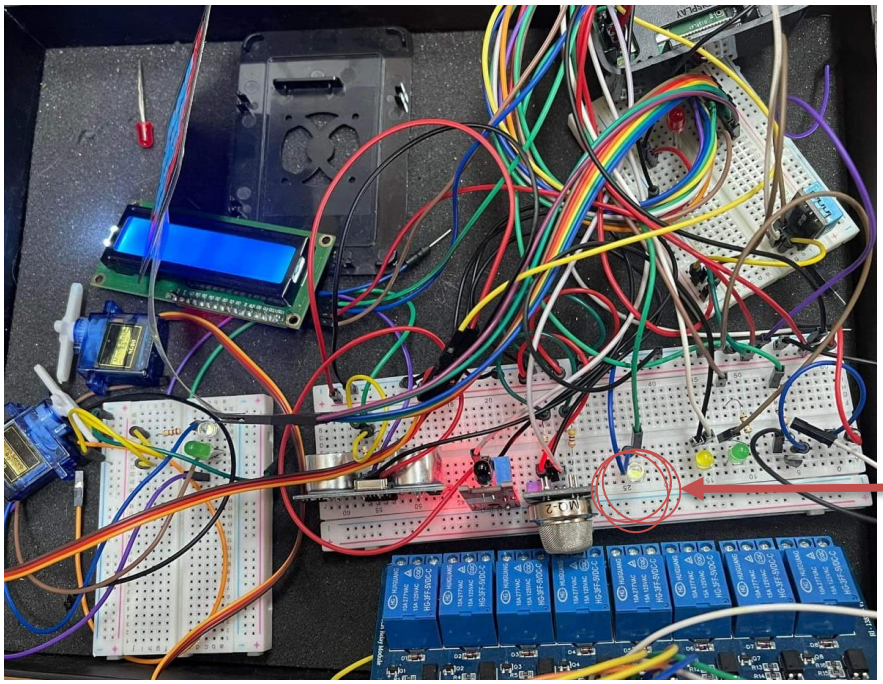


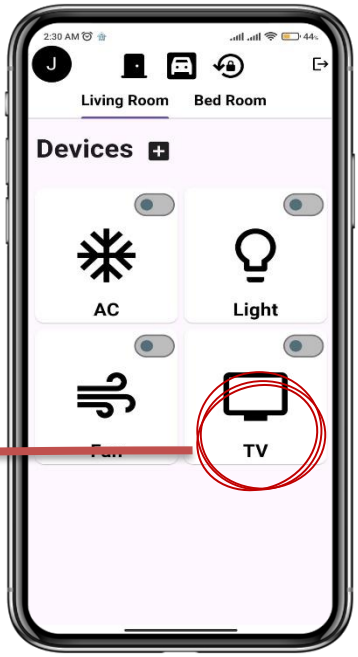


⇨ **Code:**

```
import RPi.GPIO as GPIO

import time

from collections.abc import MutableMapping

from firebase import firebase

from time import sleep

url="https://mothakem-8a457-default-rtdb.firebaseio.com/"

firebase = firebase.FirebaseApplication(url)

yellow_pin = 6

GPIO.setmode(GPIO.BCM)

GPIO.setup(yellow_pin, GPIO.OUT)
```

```
#=======================Servo & Led
d2=firebase.get("/37/servo/servo_action",None)
yellow=firebase.get("/37/1/Led_action",None)
if yellow==True:
print("ai")
GPIO.output(yellow_pin, GPIO.HIGH)
elif yellow==False:
GPIO.output(yellow_pin, GPIO.LOW)
print(yellow)
```





⇨ **Code:**

```
import RPi.GPIO as GPIO

import time

from collections.abc import MutableMapping

from firebase import firebase

from time import sleep

url="https://mothakem-8a457-default-rtdb.firebaseio.com/"

firebase = firebase.FirebaseApplication(url)

white_pin = 20

GPIO.setmode(GPIO.BCM)

GPIO.setup(white_pin, GPIO.OUT)

#======================Servo & Led

d2=firebase.get("/37/servo/servo_action",None)

white=firebase.get("/37/3/TV_action",None)

if white==True:

print("ai")

GPIO.output(white_pin, GPIO.HIGH)

elif white==False:

GPIO.output(white_pin, GPIO.LOW)

print(white)
```

3.5 Mobile Application

- Welcome Page:

Figure 85.Welcome Page

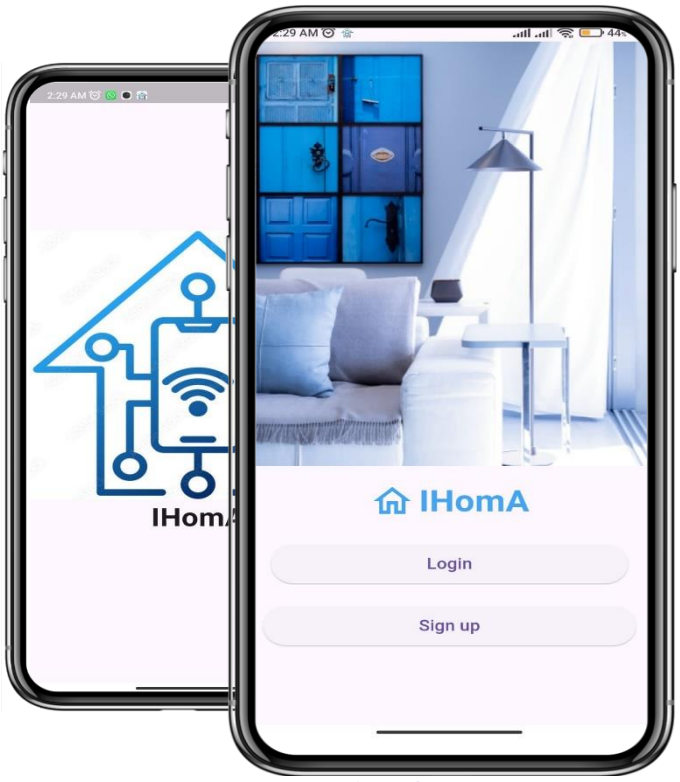


Figure 86.Home Page

Home Page

Login Page

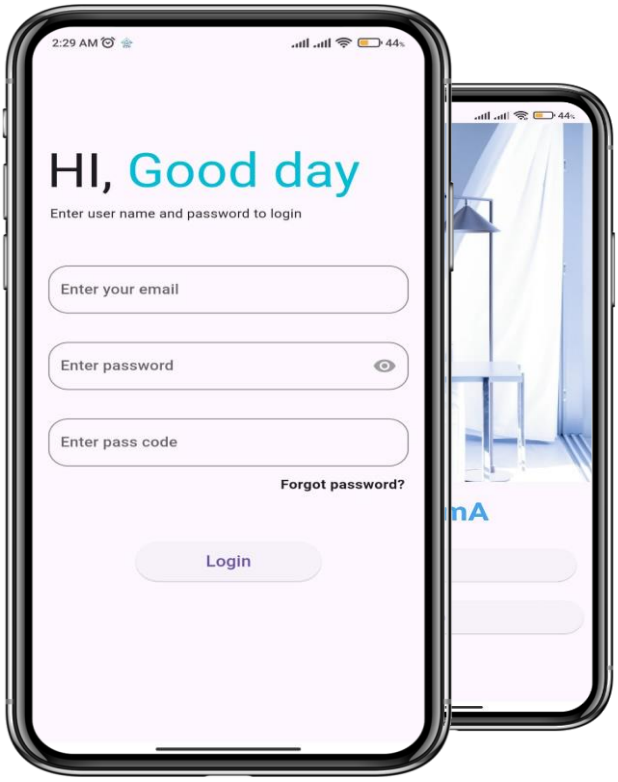


Figure 87.Login Page

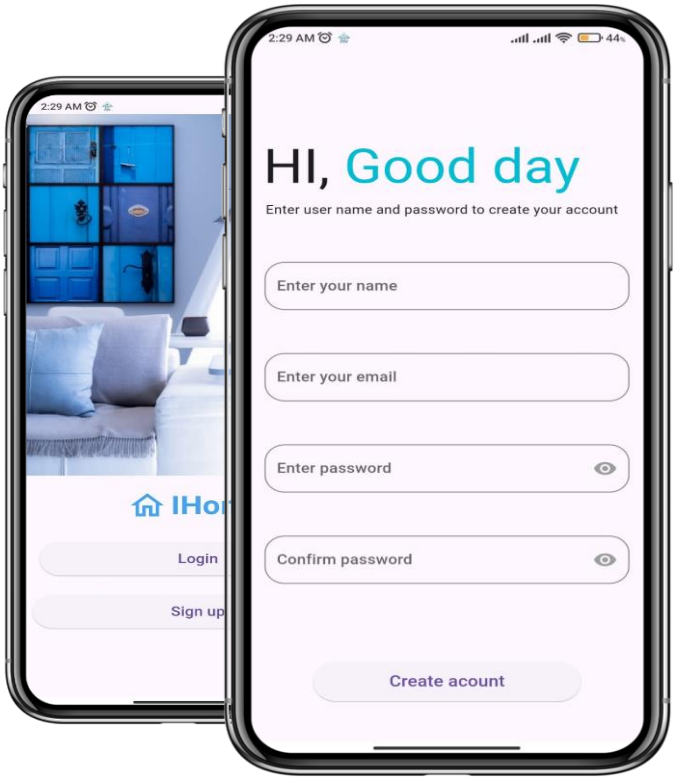


Figure 88.Sign up

Sign UP

Figure 89.Living Room

Living Room

Figure 90.Bed Room

Bed Room

Add New Device

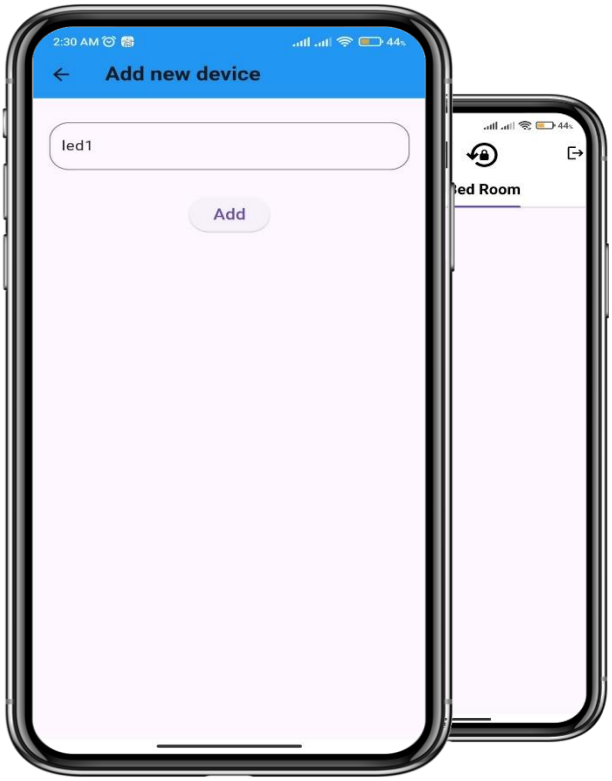


Figure 91.Add New Device

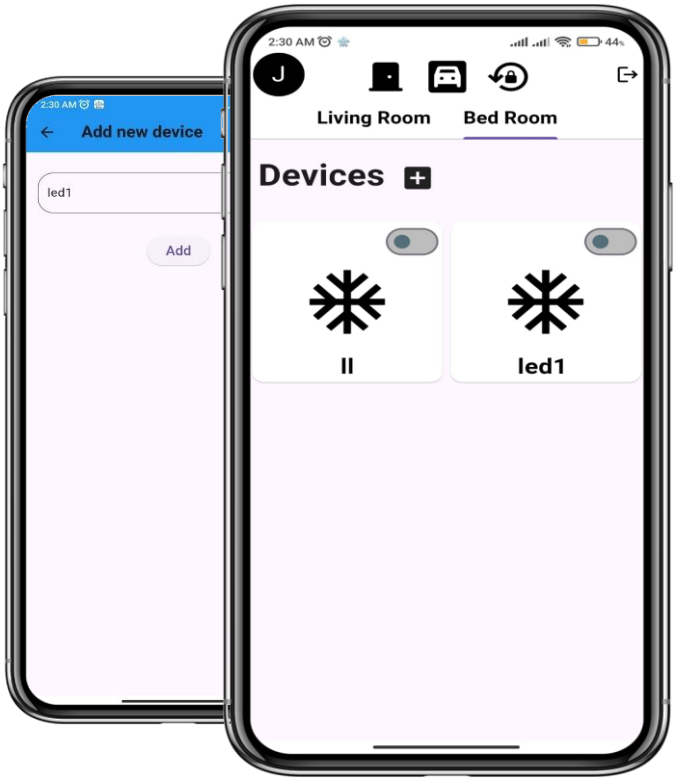


Figure 92.Add New Led Icon

Add New Led Icon

3.6 Test scenario

Test scenario		
Module	Test scenario ID	Test scenario description
Login	ID_1	The user with username 20204990@fue.edu.eg and password 20204990 tried to login with email 20205070@fue.edu.eg and password 20204990 and alert message appears with “Incorrect username or password”
Login	ID_2	The user with username 20204990@fue.edu.eg and password 20204990 tried to login with email 20204990@fue.edu.eg and password 20204990 and alert message appears with “Login successfully”

Test scenario		
Module	Test scenario ID	Test scenario description
Signup	ID_3	The user tried to sign up through the application with username 20204990@fue.edu.eg, name Khaled , phone 01154433898, password 20204990 and there’s alert message appeared with “Registration successful”

Test scenario		
Module	Test scenario ID	Test scenario description
Servo motor	ID_4	After the user enter username and password and the camera capture the visitor in front of the gate, the user can open the gate from application and the servo motor will open the gate
Servo motor	ID_5	When the visitor stands front of the camera on the gate the user will get alert on his application that someone stand front of the gate and chooses from app to “open” or “denied” if he presses “open” the gate will be open and if he presses “denied” the gate will not open

Test scenario		
Module	Test scenario ID	Test scenario description
Flame sensor	ID_6	When the sensor senses the presence of a fire in the house, it will send a message to the user that “there is a fire,” and the camera will open so that the user can confirm that there are fires. The user will choose from two options: “Turn on the water” and “Reject.”.
Gas sensor	ID_7	When the sensor senses the presence of a gas in the house, it will send a message to the user that “there is a gas leakage” and the camera will open so that the user can confirm that there is gas. The user will choose from two options: “call 122” and “Reject.”.

Test scenario		
Module	Test scenario ID	Test scenario description
Ultrasonic sensor	ID_8	When the visitor stands front of the ultrasonic sensor, It measures the distance using ultrasonic sound waves to open the gate.
Temperature sensor	ID_9	It responsible for detect and measure coldness and heat and convert it into an electrical signal. when sensor senses the presence of high temperature, it calls buzzer to make a sound and send an alert to the user that “the temperature is high”

Test scenario		
Module	Test scenario ID	Test scenario description
Raspberry pi	ID_10	When the user press on the "AC" button in the mobile application. the Raspberry pi sends a signal to the transistor, activating the relay connected to the air condition. The air condition open.
Raspberry pi	ID_11	When the user press on the " Light " button in the mobile application. the Raspberry pi sends a signal to the transistor, activating the relay connected to the light. The light open.

Raspberry pi	ID_12	When the user press on the " Fan " button in the mobile application. the Raspberry pi sends a signal to the transistor, activating the relay connected to the Fan. The Fan open.
Raspberry pi	ID_13	When the user press on the " TV " button in the mobile application. the Raspberry pi sends a signal to the transistor, activating the relay connected to the TV. The TV open.
Raspberry pi	ID_14	When the user press on the " Add Devices " button in the mobile application. the Raspberry pi sends a signal to the transistor that there are new devices is added in mobile application. And send signal to relay to take its place that the user can control this device by using mobile application. The new device is added.

4. Experiment work

4. Experiment work

4.1 Introduction

In this chapter, we will discuss Experimental work, specifically focusing on IOT home automation. These features are still in the development phase and not ready for deployment into the actual product or system. Regular experimentation is crucial for IOT home automation systems to ensure continuous improvement. This process helps validate new developments and changes, demonstrating their potential positive impact on user experience and system functionality. By conducting regular tests, developers can refine IOT devices and infrastructure, ensuring they meet user needs and expectations effectively.

4.2 Evaluation Scenarios

Evaluation scenario			
Module	Evaluation scenario ID	Evaluation scenario description	Result
Login (sign in)	1	Valid user id	“Login Successfully”
Login (sign up)	2	Invalid user id	“fail login”
Sign up (registration)	3	Register new user	“admin permit for new user”
Light	4	Turn on / off	“successfully”
Fan /AC	5	Turn on / off	“successfully”
TV	6	Turn on / off	“ successfully”
Add Devices	7	Adding new devices	“add successfully”

Open doors (main door)	8	using keypad to enter passcode to open the door.	“open Successfully with valid passcode”
Open doors (garage)	9	open garage doors by using mobile application	“successful”
Temperature measure	10	Detect high / low temperature	“Detected Successfully”
Appliance controlling	11	Control devices using mobile application	“control successfully”
Safety assurance (Gas Sensor)	12	Detect gas emissions	“Detected Successfully”
Safety assurance (fire Sensor)	13	Detect fire emissions	“Detected Successfully”

5. Conclusion and Future Work

5. Conclusion and Future Work

5.1 Conclusion

The IOT Home Automation project aimed to retrofit existing homes with modern functionalities, providing users with enhanced security features, comprehensive remote-control options, and real-time environmental monitoring. The project followed a systematic Software Development Life Cycle (SDLC) approach, integrating IOT devices, cloud-based servers, and mobile applications to achieve its objectives. Through careful planning, development, testing, and deployment, the project successfully enhanced the convenience, safety, and efficiency of home environments.

5.2 Future Work

- **City Automation:** involves using technologies like IOT, AI, and data analytics to enhance urban management and services. This includes automated traffic control, smart utilities, and efficient waste management systems. this improve quality of life, sustainability, and operational efficiency in urban environment.
- **Traffic Automation:** involves using smart sensors and IOT to manage and optimize traffic flow. it includes systems such as real time traffic monitoring, and autonomous vehicles. These innovations aim to reduce congestion, enhance safety, and improve overall transportation efficiency.
- **IOT Automation in water management:** uses connected sensors and devices to monitor and control water distribution and quality in real time. This enhances efficiency, reduces waste, and promotes sustainable water use.
- **IOT Automation in electricity management:** uses smart devices and sensors to monitor and control energy consumption in real time. this enhances efficiency, reduces costs, and supports sustainable energy use.

References:

- ✓ (PDF) IoT-Based Integrated Smart Home Automation System (researchgate.net)
- ✓ IoT-Based Smart Home Automation (researchgate.net)
- ✓ (PDF) SMART HOME AUTOMATION SYSTEM BASED ON IoT (researchgate.net)
- ✓ Smart Home Automation Using IoT by Dr. Lakshmi B N, Dr. Ashwini N, Mr.S Satish Kumar Reddy (Final Year Student) :: SSRN
- ✓ Smart Home Automation System | International Research Journal on Advanced Engineering Hub (IRJAEH)
- ✓ What is IoT? - Internet of Things Explained - AWS (amazon.com)
- ✓ What is SDLC? - Software Development Lifecycle Explained - AWS (amazon.com)
- ✓ (PDF) Software Development Life Cycle Models-A Comparative Study (researchgate.net)
- ✓ A Complete Guide to SDLC Models | Types, Phases, When to use (geeksforgeeks.org)
- ✓ What is IoT (Internet of Things) and How Does it Work? | Definition from TechTarget
- ✓ Artificial intelligence: A powerful paradigm for scientific research - PMC (nih.gov)
- ✓ Artificial intelligence - Wikipedia
- ✓ What is a Microcontroller and How Does it Work? (techtarget.com)
- ✓ Microcontrollers Types : Advantages, Disadvantages & Their Applications (elprocus.com)
- ✓ 20 Different Types of Arduino Boards And Their Features (internshala.com)

- ✓ #1 Raspberry Pi Course-7 Different Types-Models (embeddedschool.in)
- ✓ Types of Smart Home Cameras | Constellation
- ✓ Best Raspberry Pi Camera For Your Project | Random Nerd Tutorials
- ✓ What is a Flame Detector and How Does it Work? (crowcon.com)
- ✓ Gas Sensor - an overview | ScienceDirect Topics
- ✓ What is an Ultrasonic Sensor? | Sensor Basics: Introductory Guide to Sensors | KEYENCE
- ✓ Different Types of Relay Module Configurations (geya.net)
- ✓ Temperature Humidity Sensor Module(DHT11) – Future Electronics Egypt (fut-electronics.com)
- ✓ Non-functional requirement - Wikipedia