Table of Contents

Chapter One. Introduction

1. Introduction

1.1 **Introduction to the solar panel** 4

1.2 Introduction to the smart home 6

1.3 Purpose 7

1.4 Problem statement 8

1.5 Scope 8

Chapter Two. Technologies and used Tools

1.Technologies and used Tools

1.1 Hardware Components.....10

1.2 Software Tools 35

Chapter Three. Analysis

1.Diagrams 39

1.1 Use Case Mane Diagram.....39

1.2 Context Diagram.....40

1.3 Sub Use Case.....41

1.4 Component Diagram.....42

Chapter Four. Design

1.Smart Home Application 43

1.1 Page one 43

1.2 Page Two 45

1.3 Page Three and Main Page 47

1.4 Page Four.....49

1.5 Page Five 51

1.6 Page Six 52

1.7 Page Seven 53

1.8 Page Eight and The Last Page 54

Chapter Five. Implementation and Tools

1.Project definition 55

2.Code 55

3.Code link on GitHub.....56

3.1 The code of Arduino.....56

3.2 The purpose of the application.....59

3.3 The code of application 60

Chapter Six. Application Sector

1. Flutter 77
2. SDK and Flutter SDK Components 79
3. Revolutionizing Cross-Platform Development.....81

Chapter One. Introduction

Introduction to Solar Tracking

Solar tracking is the process of orienting a solar panel or array towards the sun's position throughout the day in order to maximize its solar energy output. This is achieved through the use of a tracking system that adjusts the angle and orientation of the solar panel or array in response to the changing position of the sun in the sky.

There are two main types of solar tracking systems: single-axis and dual-axis. Single-axis trackers rotate the solar panel or array around a single axis to follow the sun's daily east-to-west movement. Dual-axis trackers, on the other hand, can rotate the panel or array around both a horizontal and a vertical axis to track the sun's movement in both the east-west and north-south directions.

Solar tracking systems can significantly increase the efficiency of a solar panel or array, as they allow the panel to capture more sunlight throughout the day. This is particularly important in regions with high levels of solar insolation, where the added energy output can greatly increase the viability and economic feasibility of solar power installations.

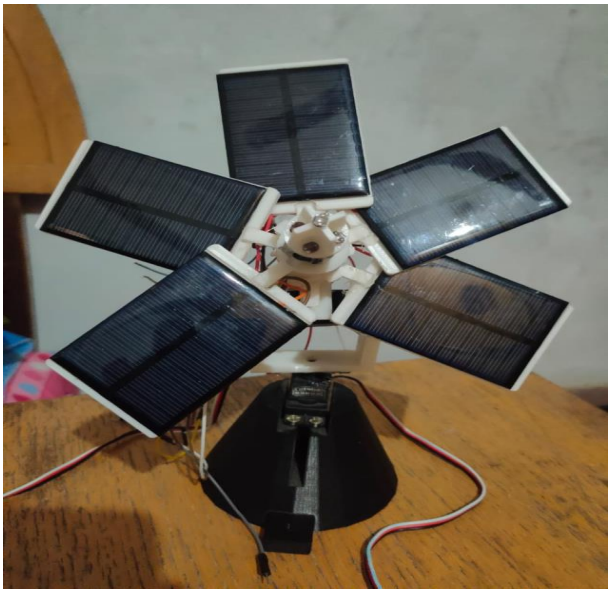
Solar tracking systems are designed to optimize the amount of solar energy that a solar panel or array can generate. By keeping the panel or array oriented towards the sun at all times, solar tracking can increase the energy output of a solar installation by up to 40% compared to a fixed configuration.

There are a few different types of solar tracking systems that are commonly used:

1. Active tracking systems: These systems use motors or actuators to move the solar panel or array to follow the sun's movement. They require a power source to operate and can be more expensive to install and maintain.

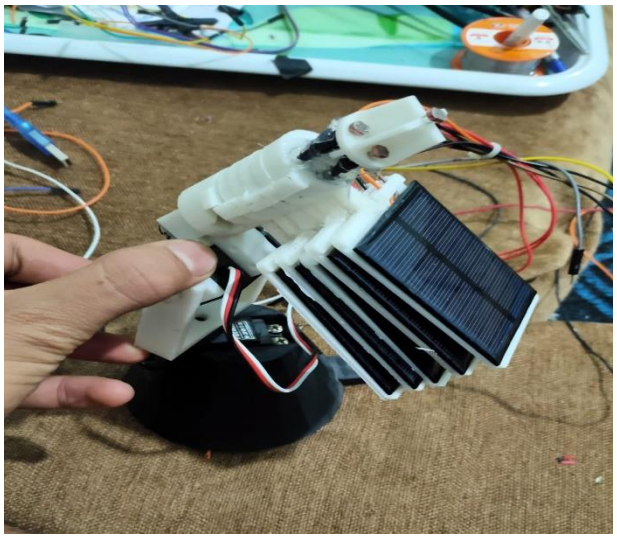
2. Passive tracking systems: These systems use mechanical or gravity-based methods to track the sun's movement without requiring any external power source. They are typically more affordable and simpler to maintain than active tracking systems.

3. Concentrated solar power (CSP) tracking systems: These systems are used in large-scale solar power plants that use mirrors or lenses to focus sunlight onto a central receiver. CSP tracking systems are typically dual-axis and require precise tracking to maintain maximum efficiency.



The choice of solar tracking system depends on a number of factors, including the size and location of the solar installation, the available budget, and the desired level of energy output.

While solar tracking systems can significantly increase the efficiency of a solar installation, they also come with some drawbacks. They can be more expensive to install and maintain than fixed solar systems, and they require additional space to accommodate the tracking mechanism. Additionally, tracking systems are more susceptible to damage from wind, hail, and other weather-related events, which can increase maintenance costs over time.



Introduction to the Smart Home

A smart home is a residence that uses internet-connected devices to remotely monitor and manage various systems and appliances, such as lighting, heating, cooling, security, and entertainment. These devices are typically controlled through a central hub or smartphone app, allowing homeowners to adjust settings or receive notifications from anywhere with an internet connection.

Smart homes are designed to increase convenience, energy efficiency, and security, while also providing a more personalized and comfortable living experience. They can be customized to meet the specific needs and preferences of individual homeowners, and can even learn and adapt to their behavior over time.

Some examples of smart home devices include smart thermostats that can be programmed to adjust the temperature based on occupancy patterns, smart lighting systems that can be controlled remotely and set to different moods or schedules, and smart security systems that can alert homeowners to potential intruders or other threats.

Smart home technology is becoming increasingly popular as the cost of devices decreases, and more homeowners seek to simplify their lives and reduce their environmental impact. However, there are also concerns about data privacy and security, as well as the potential for technical issues or compatibility problems between different devices.

Purpose

The purpose of solar panels in a smart home is to generate renewable energy from sunlight, which can be used to power various devices and appliances within the home. Solar panels are typically installed on the roof or a nearby location that receives ample sunlight throughout the day.

When a smart home is equipped with solar panels, the energy generated by the panels can be used to power the home's lighting, heating and cooling systems, and other electrical appliances. Any excess energy generated can be stored in batteries or sent back to the grid for credit.

Smart home technology can be used to optimize energy use and reduce energy waste. By using sensors and automation, smart homes can adjust the temperature, lighting, and other settings based on occupancy, time of day, and weather conditions. This can help to reduce energy consumption and save money on energy bills.

In addition, smart homes can be equipped with energy monitoring systems that track energy usage and provide insights into how energy is being used throughout the home. This can help homeowners identify areas where energy use can be reduced and make changes to improve energy efficiency.

Overall, the combination of solar panels and smart home technology can help to reduce energy costs, increase energy efficiency, and reduce carbon emissions by generating and using renewable energy.

Problem statement

The problem statement for a smart home with a solar panel can be framed as follows:

As the world moves towards sustainable living, many homeowners are opting for solar panels to power their homes. However, the challenge lies in optimizing the usage of solar energy to minimize grid dependency while maintaining the comfort and convenience of a modern home. A smart home with a solar panel system must be designed to intelligently manage energy consumption throughout the day, taking into account the varying availability of solar energy. This requires the integration of various technologies such as energy storage systems, smart home devices, and energy management software. The goal is to achieve maximum energy efficiency, cost savings, and environmental sustainability without compromising on the quality of life. The problem, therefore, is to develop a comprehensive, reliable, and user-friendly solution that seamlessly integrates all these components to create a truly smart and sustainable home.

Scope

The scope of solar panels and smart home technology is vast and growing. As the world transitions towards cleaner and more sustainable energy sources, solar panels and smart home systems are becoming increasingly popular and widely adopted.

In terms of the scope of solar panels, they can be used for a wide range of applications beyond just powering homes. Solar panels can be used to power commercial and industrial buildings, streetlights, electric vehicles, and even entire communities. They can also be used in remote areas where access to traditional power grids is limited or nonexistent.

Similarly, the scope of smart home technology is expanding rapidly. Smart home systems can be used to control and automate a wide range

of devices and appliances, including lighting, heating and cooling systems, security systems, and entertainment systems. The integration of artificial intelligence and machine learning capabilities into smart home technology is also opening up new possibilities for energy optimization and efficiency.

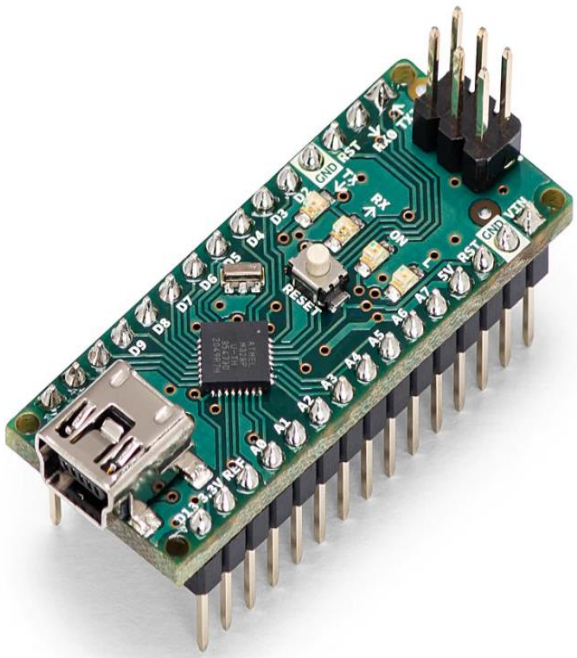
As the technology continues to evolve, the scope of solar panels and smart home systems is likely to grow even further. For example, advancements in battery storage technology are making it possible to store excess solar energy for later use, while innovations in energy management systems are allowing homeowners to better monitor and manage their energy usage in real-time. These developments will make solar panels and smart home systems an even more attractive option for those looking to reduce their energy costs and carbon footprint.

Chapter Two. Technologies and used Tools

Hardware Requirements

- The Arduino Nano

The Arduino Nano is a small, complete, and breadboard-friendly board based on the ATmega328 (Arduino Nano 3.x). It has more or less the same functionality of the Arduino Duemilanove, but in a different package. It lacks only a DC power jack, and works with a Mini-B USB cable instead of a standard one.



Power

The Arduino Nano can be powered via the Mini-B USB connection, 6-20V unregulated external power supply (pin 30), or 5V regulated external power supply (pin 27). The power source is automatically selected to the highest voltage source.

Memory

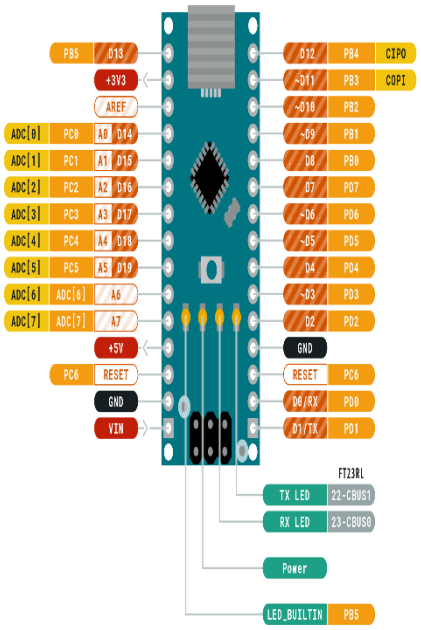
The ATmega328 has 32 KB, (also with 2 KB used for the bootloader. The ATmega328 has 2 KB of SRAM and 1 KB of EEPROM.

Input and Output

Each of the 14 digital pins on the Nano can be used as an input or output, using pinMode (), digitalWrite (), and digitalRead () functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the FTDI USB-to-TTL Serial chip.

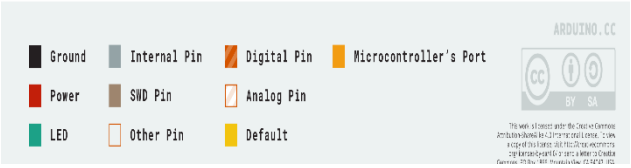
- External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt () function for details.



- PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite () function.

- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.

- LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.



The Nano has 8 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default, they measure from ground to 5 volts, though is it possible to change the upper end of their range using the analogReference () function. Analog pins 6 and 7 cannot be used as digital pins. Additionally, some pins have specialized functionality:

- I2C: A4 (SDA) and A5 (SCL). Support I2C (TWI) communication using the Wire library (documentation on the Wiring website).

There are a couple of other pins on the board:

- AREF. Reference voltage for the analog inputs. Used with analogReference ().

- Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Communication

The Arduino Nano has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provide UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An FTDI FT232RL on the board channels this serial communication over USB and the FTDI drivers (included with the Arduino software) provide a virtual com port to software on the computer. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the FTDI chip and USB connection to the computer (but not for serial communication on pins 0 and 1). A SoftwareSerial library allows for serial communication on any of the Nano's digital pins. The ATmega328 also support I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus. To use the SPI communication, please see ATmega328 datasheet.

Programming

The Arduino Nano can be programmed with the Arduino software (download). Select "Arduino Duemilanove or Nano w/ ATmega328" from the Tools > Board menu (according to the microcontroller on your board). The ATmega328 on the Arduino Nano comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol. You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header using Arduino ISP or similar.

Automatic (Software) Reset

Rather then requiring a physical press of the reset button before an upload, the Arduino Nano is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the FT232RL is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this

capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload. This setup has other implications. When the Nano is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Nano. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

Servo Motor s3003

This servo can produce high-current draw from your batteries but if NiMH and LiPo batteries are used, you need to make sure that they are capable of delivering sufficient amps. This is widely used in applications in which accuracy is required. It's also a great tool for building your own 3D printer or CNC.

The information presented in the datasheet below can be used to calculate what signal it takes to rotate the motor a certain number of degrees. The advantage of servos to stepper motor is when they are empty, servo motors neither consume nor keep shafts blocked.

Servo Moteur sg90

The Micro Servo Module has a standard connection 3 points.

It is an ideal choice for your Arduino-driven project of robototique and Mechatronics.

The Micro Servo Module consists of an electric motor mechanically linked to a potentiometer. Using the library Servo Library to be able to fly easily this module with an Arduino kit.
The electronics inside the servo-motor turns a PWM pulse in physical position width: when the servo is controlled, the engine will be operated up to the corresponding value of the potentiometer at the required position.

A kit of 3 ends in plastic and a screw is supplied with the sero-engine microphone so you can easily connect it to the mechanical universe.

Small Geared DC Motor 6Vdc (100 RPM)

This N20-6V-100 Rpm Micro Metal Gear Motor has small volume, torsion big, all metal gear, durable, not easy to wear. Great replacement for the rusty or damaged DC geared speed reduce motor on the machine. Widely used on Boat, Car, Electric Bicycle, Fan, Home Appliance.

The N20 Micro Gear 6V 100 RPM DC Motor (High Torque) is lightweight, high torque, and low RPM motor. It is equipped with gearbox assembly so as to increase the torque of the motor. It has a cross-section of 10 × 12 mm, and the D-shaped gearbox output shaft is 9 mm long and 3 mm in diameter. It has a very small size so as fit in complex spaces of small-scale application. One can connect this Micro Gear Motor to wheels to drive them from one place to other while carrying high loads.

This N20 Micro Gear 100RPM 6V DC Motor (High Torque) has small volume, torsion big, all metal gear, durable, not easy to wear. Great replacement for the rusty or damaged DC geared speed reduce motor on the machine. Widely used on Boat, Car, Electric Bicycle, Fan, Home Appliance.

Solar panal

A solar panel with specifications 5.5V and 0.5W means that it can generate a maximum voltage of 5.5 volts and a maximum power output of 0.5 watts under ideal conditions. The physical dimensions of the panel are 70mm by 50mm.

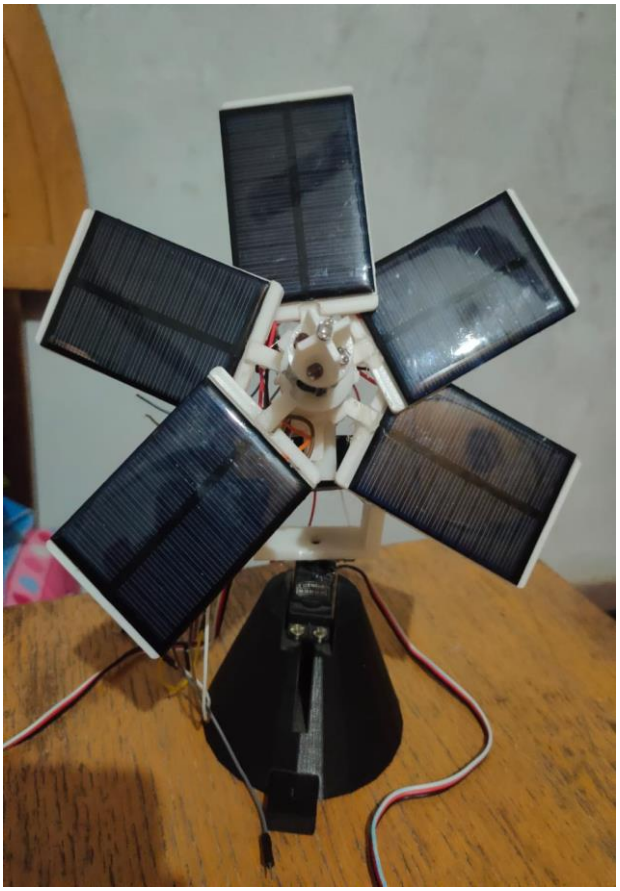
The voltage and power output of the panel are dependent on various factors, such as the intensity of the sunlight, the angle of the panel, and any shading or obstructions that may be present. Therefore, the actual voltage and power output of the panel may vary based on the environmental conditions.

The panel can be used to generate electricity and charge batteries in various applications, such as in portable electronics, outdoor lighting, and small-scale projects. It is important to note that the panel should be connected to a suitable charge controller to ensure that it does not overcharge the batteries or damage the devices being charged.

- Solar panels are devices that convert sunlight into electricity using photovoltaic (PV) cells. When sunlight hits the PV cells, it creates an electric field that causes electrons to flow, generating a direct current (DC) electricity.

- Solar panels come in different sizes and power ratings, depending on the number and type of PV cells used. The power rating is the maximum power output that the panel can produce under standard test conditions (STC), which is 1000 watts per square meter of sunlight, at a temperature of 25°C and with the panel facing directly at the sun.

- The physical dimensions of a solar panel do not necessarily determine its power output. A larger panel may have more PV cells and a higher power output, but it may also be more expensive and less portable. A smaller panel may be more affordable and portable, but it may produce less power and take longer to charge a battery.



- When using a solar panel to charge a battery, it is important to use a charge controller to regulate the voltage and prevent overcharging. A charge controller is a device that sits between the solar panel and the battery, and it ensures that the battery is charged safely and efficiently.

- Solar panels can be used in a variety of applications, such as in off-grid cabins, boats, RVs, and remote locations where grid power is not available. They can also be used in urban settings to supplement grid power, reduce energy bills, and lower carbon emissions.

- Solar panels have become more affordable and efficient in recent years, making them a popular choice for renewable energy. They require little maintenance and can last for decades, making them a reliable investment for homeowners and businesses alike.

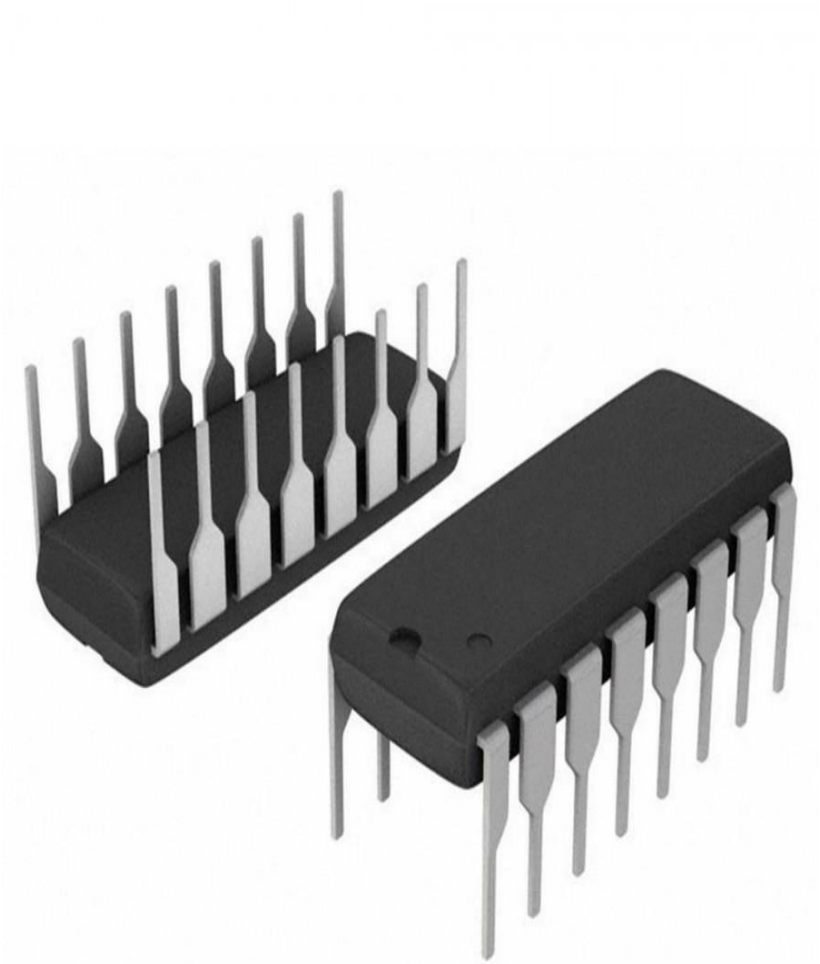
L7806

L78 Datasheet Positive voltage regulator ICs TO-220 DPAK TO-220FP D²PAK
Features
• Output current up to 1.
5 A
• Output voltages of 5; 6; 8; 8.
5; 9; 12; 15; 18; 24 V
• Thermal overload protection
• Short circuit protection
• Output transition SOA protection
• 2 % output voltage tolerance (A version)
• Guaranteed in extended temperature range (A version)
Description The L78 series of three-terminal positive regulators is available in TO-220, TO-220FP, D²PAK and DPAK packages and several fixed output voltages, making it useful in a wide range of applications.
These regulators can provide local.

L293D

The L293 and L293D devices are quadruple high-current half-H drivers. The L293 is designed to provide bidirectional drive currents of up to 1 A at voltages from 4.5 V to 36 V. The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. Both devices are designed to drive inductive loads such as relays, solenoids, DC and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply
applications.

Each output is a complete totem-pole drive circuit, with a Darlington transistor sink and a pseudo-Darlington source. Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN.



The L293 and L293D are characterized for operation from 0°C to 70°C.

1N4007

Diodes Incorporated Makes No Warranty Of Any Kind, Express Or Implied, With Regards To This Document, Including, But Not Limited To, The Implied Warranties Of Merchantability And Fitness For A Particular Purpose (And Their Equivalents Under The Laws Of Any Jurisdiction). Diodes Incorporated and its subsidiaries reserve the right to make modifications, enhancements, improvements, corrections or other changes without further notice to this document and any product described herein. Diodes Incorporated does not assume any liability arising out of the application or use of this document or any product described herein; neither does Diodes Incorporated convey any license under its patent or trademark rights, nor the rights of others. Any Customer or user of this document or products described herein in such applications shall assume all risks of such use and will agree to hold Diodes Incorporated and all the companies whose products are represented on Diodes Incorporated website, harmless against all damages. Diodes Incorporated does not warrant or accept any liability whatsoever in respect of any products purchased through unauthorized sales channel. Should Customers purchase or use Diodes Incorporated products for any unintended or unauthorized application, Customers shall indemnify and hold Diodes Incorporated and its representatives harmless against all claims, damages, expenses, and attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized application. Products described herein may be covered by one or more United States, international or foreign patents pending. Product names and markings noted herein may also be covered by one or more United States, international or foreign trademarks. This document is written in English but may be translated

into multiple languages for reference. Only the English version of this document is the final and determinative format released by Diodes Incorporated.

LDR

What is a Light Dependent Resistor (LDR) or Photoresistor?

A **Light Dependent Resistor** (also known as a photoresistor or LDR) is a device whose resistivity is a function of the incident electromagnetic radiation. Hence, they are light-sensitive devices. They are also called as photoconductors, photoconductive cells or simply photocells.

They are made up of semiconductor materials that have high resistance. There are many different symbols used to indicate a photoresistor or LDR, one of the most commonly used symbol is shown in the figure below. The arrow indicates light falling on it.

Working Principle of Photoresistor (LDR)

So how exactly does a photoresistor (i.e. a light dependent resistor or LDR) work? Photoresistors work based off of the principle of photoconductivity.

Photoconductivity is an optical phenomenon in which the material's conductivity is increased when light is absorbed by the material.

When light falls i.e. when the photons fall on the device, the electrons in the valence band of the semiconductor material are excited to the conduction band. These photons in the incident light should have energy greater than the bandgap of the semiconductor material to make the electrons jump from the valence band to the conduction band.

Hence when light having enough energy strikes on the device, more and more electrons are excited to the conduction band which results in a large number of charge carriers. The result of this process is more and more current starts flowing through the device when the circuit is closed and hence it is said that the resistance of the device has been decreased. This is the most common **working principle of LDR**.

Characteristics of Photoresistor (LDR)

Photoresistor LDR's are light-dependent devices whose resistance is decreased when light falls on them and that is increased in the dark. When a light dependent resistor is kept in dark, its resistance is very high. This resistance is called as dark resistance. It can be as high as 1012 Ω and if the device is allowed to absorb light its resistance will be decreased drastically. If a constant voltage is applied to it and the intensity of light is increased the current starts increasing. The figure below shows the resistance vs. illumination curve for a particular **LDR**.

18650 battery

What is an 18650 battery?

The 18650 battery is a lithium-ion cell classified by its 18mm x 65mm size, which is slightly larger than a AA battery. They're often used in flashlights, laptops, and high-drain devices due to their superior capacity and discharge rates.

18650s come in both flat and button top styles, and usually boast 300-500 charge cycles. You can learn more about 18650 batteries

Resistor 10K

A resistor is a passive two-terminal electrical component that implements electrical resistance as a circuit element. In electronic circuits, resistors are used to reduce current flow, adjust signal levels, to divide voltages, bias active elements, and terminate transmission lines, among other uses. High-power resistors that can dissipate many watts of electrical power as heat may be used as part of motor controls, in power distribution systems, or as test loads for generators. Fixed resistors have resistances that only change slightly with temperature, time or operating voltage. Variable resistors can be used to adjust circuit elements (such as a volume control or a lamp dimmer), or as sensing devices for heat, light, humidity, force, or chemical activity.

Resistors are common elements of electrical networks and electronic circuits and are ubiquitous in electronic equipment. Practical resistors as discrete components can be composed of various compounds and forms. Resistors are also implemented within integrated circuits.

The electrical function of a resistor is specified by its resistance: common commercial resistors are manufactured over a range of more than nine orders of magnitude. The nominal value of the resistance falls within the manufacturing tolerance, indicated on the component.

LED

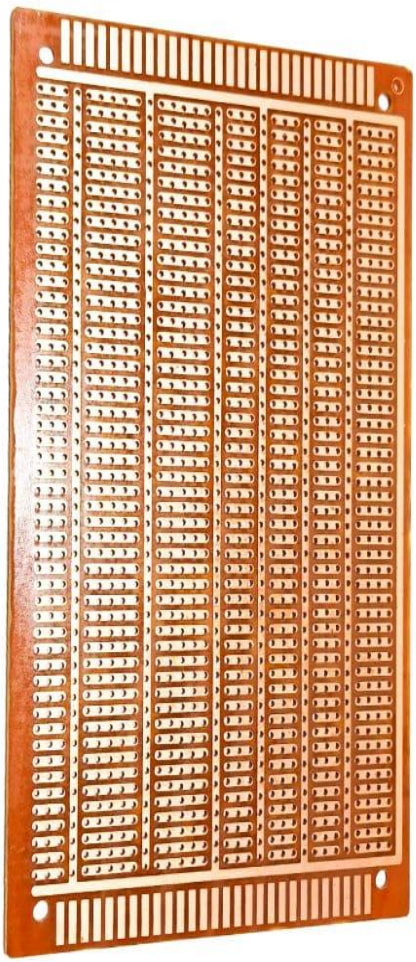
LED Overview

A Super Bright 5mm LED is exceptionally bright with a wide beam angle, so they’re suitable for use in your projects, illuminations, headlamps, spotlights, car lighting, and models. The 5mm LED can be used anywhere where you need low power, high-intensity reliable light, or indication. They go quickly into a breadboard and will add that extra zing to your project.

Breadboard

PCB prototype is an important process to your project's development, which is a trial manufacturing for the printed circuit board before mass production, electronic engineer is designing the circuit and that is mainly in doing small volume trail manufacturing for PCB manufacturer after designing the circuit and finishing PCB layout for the electronic engineer. However, there is no limit for PCB manufacturing quantity, generally speaking, PCB prototype is the best practice method to verify the quality of a design before proceeding.
undefined

PCB is the support body of electronic components and the carrier of electrical connection of electronic components, which mainly plays the role of support and interconnection. What's more, the PCB will be your finished product. Now printed circuit board (PCB) is widely used in various electronic and related products.



Where does the name "breadboard" come from?
You might be wondering what any of this has to do with bread. The term breadboard comes from the early days of electronics, when people would literally drive nails or screws into wooden boards on which they cut bread in order to connect their circuits. Luckily, since you probably do not want to ruin all your cutting boards for the sake of an electronics project, today there are better options.

How does a breadboard work?

You need to make a breadboard before creating the permanent circuit board because it can remove and change components on a breadboard. What's more, you will draw out the schematics and connect the wires accordingly.

As you know, the centre of PCB is the prototyping region, which is made up of two rows of five holes. There is a channel between the two rows, you can place a chip with pins on either side so that prevent them from connecting together. Also, you can seek out power busses (either one or two) on the side of the breadboard for working power and grounding. In fact, designing breadboards are used for integrated circuit (ICs). you can place an IC chip over the channel, which you can access to the pins on either side of the existing chip. What's more, connecting a resistor to the power bus into the channel, at the same time taking an LED from the ground bus to create the whole circuit.
undefined
Breadboard and Printed circuit board
On one hand, a breadboard is usually used as the first step before creating a printed circuit board. You can change and move circuits that are otherwise permanent on a PCB with a breadboard. On the other hand, breadboards are used for design and investigation, while the boards are for your finished products.

The advantages of breadboard:

- You can rapidly change connections and test various plans in a development phase.
- It's easy and fast to assemble as there are no permanent solder connections.
- You can also change various components such as the capacitor or resistor value.

- You can add an ammeter anywhere with shifting wires (breaking into) any branch of your circuit. What's more, the current measurement on PCBs require you to break tracks or add extra resistors in your design.

undefined

The advantages of a printed circuit board:

- The board is permanent to have an electronic device worked.
- PCB has a better current carrying capacity comparing to a breadboard, you can make your traces wider to take more current so that work well.

- You can add terminals to your printed circuit board for external connections.
- You can mount heat-sinks to the board so that have them rigid.
- There is widely used in electronic devices.
- A PCB has a cleaner look than a breadboard (when manufactured correctly).
- It is normally easier to understand the circuit on the board. None of those looping wires going everywhere.
- Nobody is going to buy your great, fantastic, electronic design (product) on a breadboard.

Should you use a PCB or a breadboard?

There is a time when you would use a breadboard over a PCB, and the other way around. According to what you are making and the stage you are at, which will help you decide when to use either a breadboard or a PCB.

When to use a breadboard?

In normal, the breadboard can be used in testing with connections and circuits. So you can move circuits around without damaging the printed circuit boards because the board is not permanent. But, there is a minimal current capacity and you'd better prepare for work before developing the actual board as it is not a permanent board.

undefined

When to use a PCB?

As you know, the printed circuit board can be used for the actual electronic device. What's more, you can develop that into a printed circuit board after testing out the breadboard and finding the perfect design for your project. Needless to say, PCB is a permanent device in electronics as it needs to solder, so the board has been widely used in your electronic projects

Buzzer

Description

Apply 3V to 5V to this piezo buzzer module and you'll be rewarded with a loud 2KHz BEEEEEEEEEEEEEEEEP. Unlike a plain piezo, this buzzer does not need an AC signal. Inside is a piezo element plus the driver circuitry that makes it oscillate at 2KHz. The piezo buzzer is 5V TTL logic compatible and Breadboard friendly pin spacing.

This buzzer is ideal when you need to fit a buzzer in a small place. It has its own built-in drive circuit. It offers low current consumption. Used in manufacturing applications such as laptops, alarms, pagers, etc.

Great for use as part of a Code Practice Oscillator.

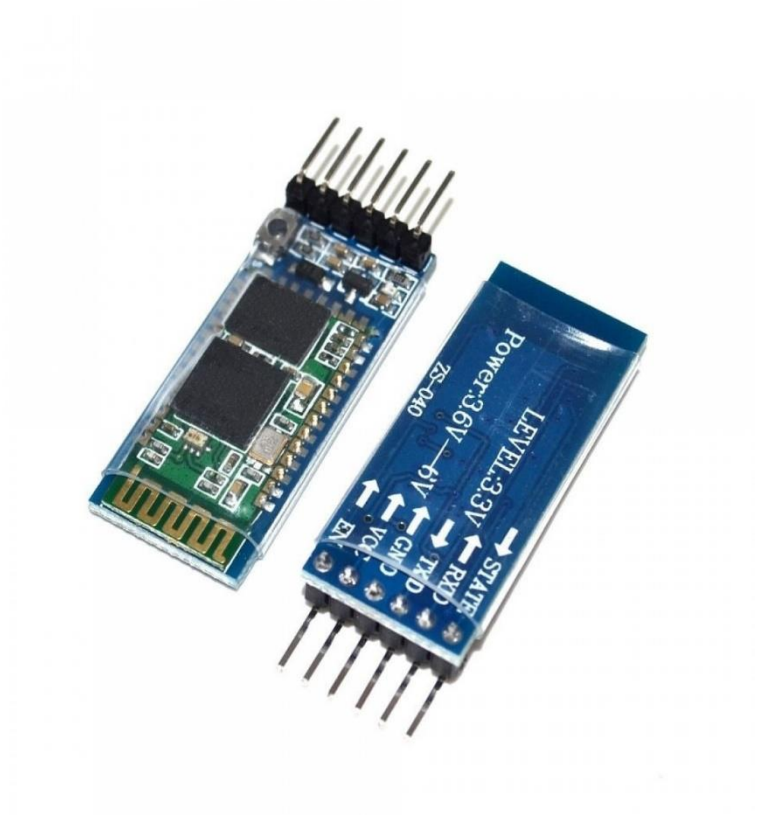
Specifications:

- Operation Voltage: 3-5V DC
- Current: <25mA
- SPL: 85dBA/10cm
- Frequency: 2,300Hz
- Color: Black
- Operating Temperature: – 20° to 65°C
- Weight: 2.4 gram
- Size: 1.2cm diameter x 1cm tall (0.47" x 0.39")
- Pin Spacing: 7.6mm

Bluetooth module HC - 05 (6 pin +button)

DESCRIPTION

HC-05 6 Pin Wireless Serial Bluetooth Module with TTL is a Bluetooth module for use with any microcontroller. It uses the UART protocol to make it easy to send and receive data wirelessly.



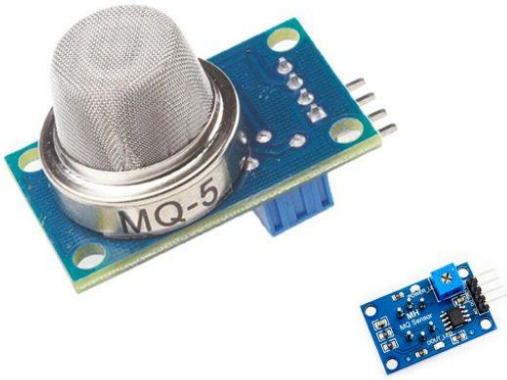
The HC-06 module is a slave only device. This means that it can connect to most phones and computers with Bluetooth but it cannot connect to another slave-only device such as keyboards and other HC-06 modules. To connect with other slave devices a master module would be necessary such as the HC-05 version which can do both master and slave.

Features:

- Working current: matching for 30 mA, matching the communication for 8 mA.
- Dormancy current: no dormancy.
- Used for a GPS navigation system, water, and electricity gas meter reading system.
- With the computer and Bluetooth adapter, PDA, seamless connection equipment.
- Bluetooth module HC-08 Master and slave Two in one module.
- Use the CSR mainstream Bluetooth chip, Bluetooth V2.0 protocol standards.
- Potter default rate of 9600, the user can be set up.
- Bluetooth protocol: Bluetooth Specification v2.0+EDR
- Speed: Asynchronous: 2.1Mbps(Max) / 160 kbps, Synchronous: 1Mbps/1Mbps.
- Security: Authentication and encryption.

Mq-5 liquofied gas methane gas sensor module

description:

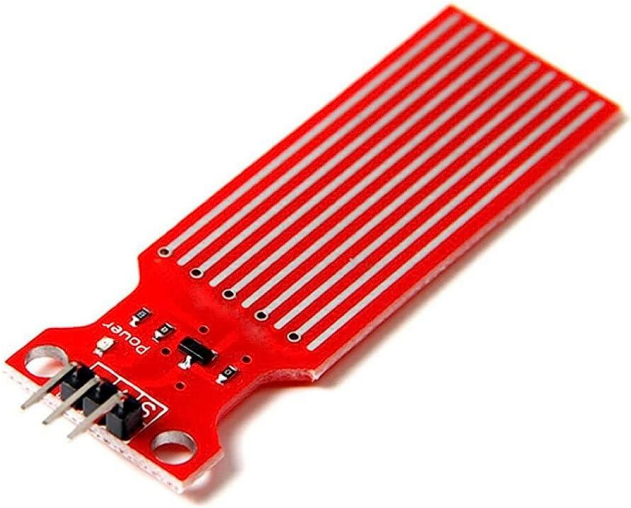


- Sensitive material of MQ-5 gas sensor is SnO2, which with lower conductivity in clean air. When the target combustible gas exists, the sensors conductivity is higher along with the gas concentration rising. Please use simple electrocircuit, convert change of conductivity to correspond output signal of gas concentration.
    - MQ-5 gas sensor has high sensitity to Methane, Propane and Butane, and could be used to detect both Methane and Propane. The sensor could be used to detect different combustible gas especially Methane, it is with low cost and suitable for different application.

Features:

- Indication of the signal output.
- Dual signal output (analog output, and TTL level output)
- TTL output signal is low. (When low output signal light, and can be connected directly to the microcontroller)
- 0 ~ 5V analog output voltage, the higher the concentration, the higher the voltage.
- Liquefied petroleum gas, natural gas, city gas, better sensitivity.
- Has a long life and reliable stability?
- Fast response and recovery characteristics
- Operating voltage: DC 5 V

Rain water level detection sensor module



Description

Water Sensor water level sensor is an easy-to-use, cost-effective high level/drop recognition sensor, which is obtained by having a series of parallel wires exposed traces measured droplets/water volume in order to determine the water level. Easy to complete water to analog signal conversion and output analog values can be directly read development board of FOR to achieve the level alarm effect.

Specifications:

Product Name: water level sensor

Operating voltage: DC3-5V

Operating current: less than 20mA

Sensor Type: Analog

Detection Area: 40mmx16mm

Production process: FR4 double-sided HASL

Operating temperature:10? -30?

Humidity: 10% -90% non-condensing

Product Dimensions: 62mmx20mmx8mm

Heatsink 2 big to -220 (25 x 14 x 9mm)

This Aluminum Heatsink Cooler Cooling Kit for Raspberry Pi 3/2 Model 3B+ designed explicitly for cooling the chips on Raspberry Pi. These are attached with thermally conductive adhesive tape, allowing the Raspberry Pi to cool more efficiently, compared with the traditional heatsinks.

This heat sink uses Aluminium alloy 1050A as the metal. Therefore, this high-quality aluminum is the best alloy for the highest heat dissipation providing adhesive thermal tape for efficient thermal transfer to the heat sink providing additional cooling to the IC. Thus the usage of high-quality aluminum ensures the best fin efficiency and fin effectiveness. It uses a thermally conductive 3M sticker for better adhesiveness to the silicon. These heat sinks designed short enough to fit with standard Raspberry Pi cases. Thermally Conductive Epoxy Adhesive offers maximum heat dissipation, thereby ensures maximum heat transfer from the silicon surface. Moreover, these heat sinks also used in other devices such as the BeagleBone to cool processors, regulators, etc.

press 4 pin (6 x 6)

Description

Features:

- Size:6x6x5mm
- Withstand Voltage:AC250V
- Rated Load: DC12V 50mA
- Used in the fields of electronic products, household or office appliances, sound equipment, digital products, and more
- Feature momentary contact, 4 pins, round black push button, SMD, 6 x 6 x 5mm size, etc.
- Used in the fields of electronic products, household appliances and more.
- High precision mechanism design offers acute operation and long service life.

Specification:

- Type: SMD
- Action Type: Momentary
- Pin Number: 4
- Dimension: 6x6x5mm
- Button Diameter: 3.5mm
- Material: Plastic, Metal, And Electric Parts
- Size: 6x6x5mm 4 pin
- Widely used in the fields of electronic product, computer mouse, instrument and meter, household appliance and more.

Jumper Wire

Arduino Jumper Wire Set (40 Jumper) Handy for making wire harnesses or umpiring between headers on PCB's. These premium jumper wires are 20cm long and come in a strip of 40. They have 0.1" male header contacts on one end and female header on the other. They fit cleanly next to each other on standard-pitch 0.1" (2.54mm) header.

Features

- Suitable for Male & Female Headers (Such as in Arduino Board)
- Current Rating up to 1A
- Mixed Colors
- Length 20 cm

Software Tools

Smart homes are becoming increasingly popular, with homeowners looking for ways to automate their houses for added convenience, energy efficiency, and security. Arduino is a popular platform for building smart home devices, and with the development of mobile apps, controlling smart homes has never been easier. In this topic, we will explore the software tools used in smart homes that work with Arduino and how Flutter can be used to develop the controlling app.

Arduino is an open-source electronics platform that allows anyone to create interactive projects. It consists of a microcontroller, which can be programmed to control various electronic devices. Arduino boards are inexpensive, widely available, and easy to use, making them an excellent choice for building smart home devices.

To program Arduino boards, developers use the Arduino IDE, an open-source software tool that runs on Windows, macOS, and Linux. The IDE is used to write and upload code to the board, which controls the smart device. Arduino IDE has a simple and easy-to-use interface, and it supports a wide range of libraries that simplify programming.

In addition to the Arduino IDE, developers can also use other software tools, such as the Platform IO IDE, to program Arduino boards. Platform IO is an open-source ecosystem for IoT development, which supports more than 600 development boards and over 20 different platforms. Platform IO also provides a built-in library manager that simplifies the process of installing and managing libraries.

To control smart homes, mobile apps can be developed using various frameworks, including Flutter. Flutter is an open-source mobile application development framework that allows developers to build high-performance, cross-platform apps for Android, iOS, and the web. Flutter uses the Dart programming language, which is easy to learn and provides features such as hot reload, which allows developers to see changes instantly.

Using Flutter, developers can create intuitive and user-friendly mobile apps that allow homeowners to control their smart homes remotely. Flutter provides a wide range of widgets and tools that simplify app development, including customizable buttons, sliders, and switches. Flutter also has an active community that provides plugins and packages that can be used to add more features to the app.

When developing mobile apps using the Flutter framework, developers use an integrated development environment (IDE) to write, test, and debug code. There are several popular IDEs that support Flutter, including:

Android Studio: Android Studio is the official IDE for Android app development and is fully integrated with Flutter. It provides a comprehensive development environment that includes code editing, debugging, and performance profiling tools. Android Studio also provides built-in support for the Dart programming language used by Flutter.

Visual Studio Code: Visual Studio Code is a lightweight, cross-platform code editor that supports Flutter. It provides features such as code highlighting, auto-completion, and debugging tools, making it a popular choice for Flutter developers. Visual Studio Code also has a large library of plugins and extensions that can be used to customize the editor and add more features.

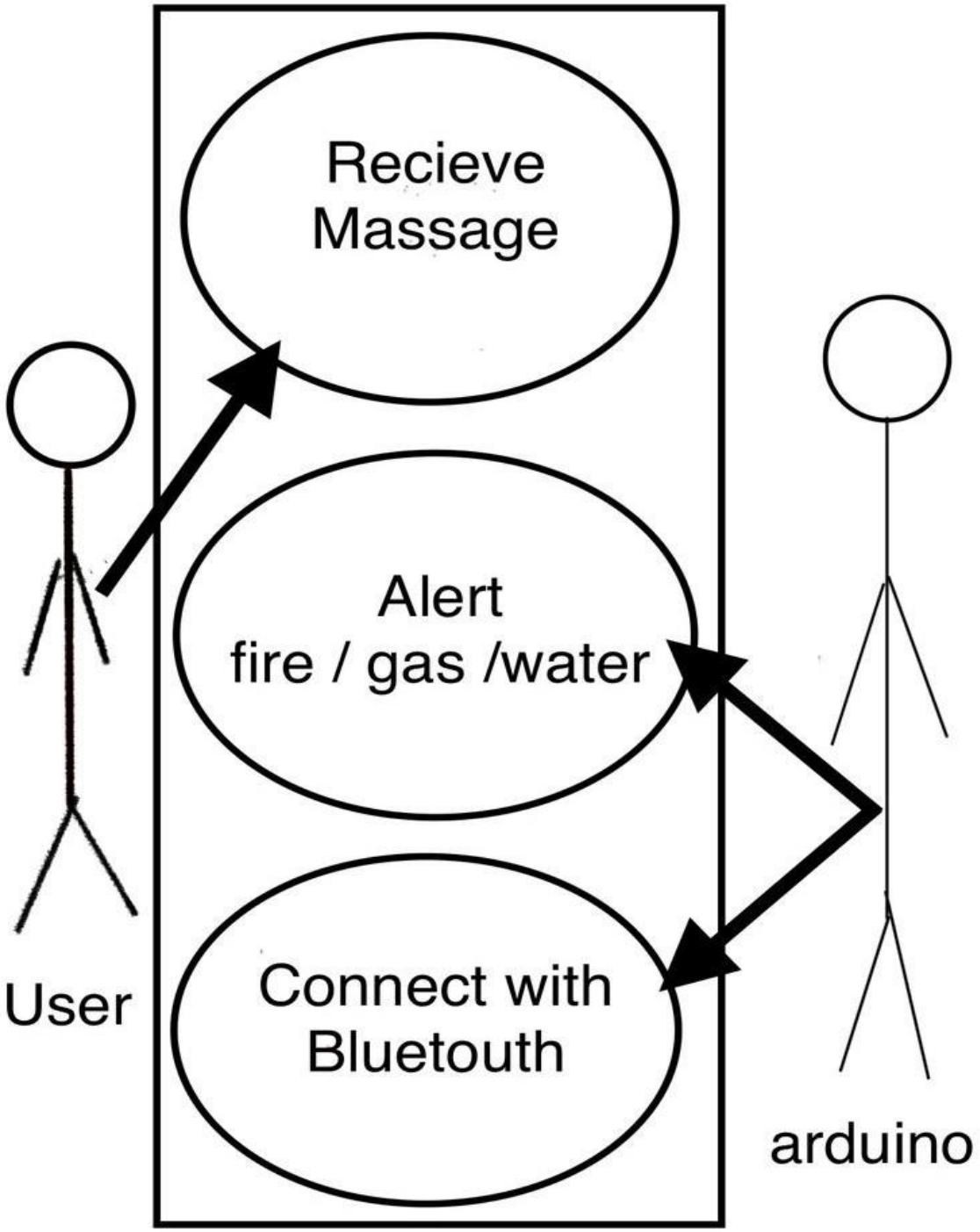
When developing mobile apps using Flutter, developers can choose from several popular IDEs, including Android Studio and Visual Studio Code. Each IDE provides a comprehensive set of tools and features that make it easier to write, test, and debug Flutter code. Choosing the right IDE is a matter of personal preference and depends on the developer's needs and expertise.

In conclusion, building smart homes using Arduino is becoming increasingly popular, and with the development of mobile apps, controlling smart homes has never been easier. Arduino IDE and PlatformIO are software tools that simplify programming Arduino boards, while Flutter is an excellent framework for developing mobile apps that control smart homes. With these tools, developers can create smart homes that are energy-efficient, secure, and convenient.

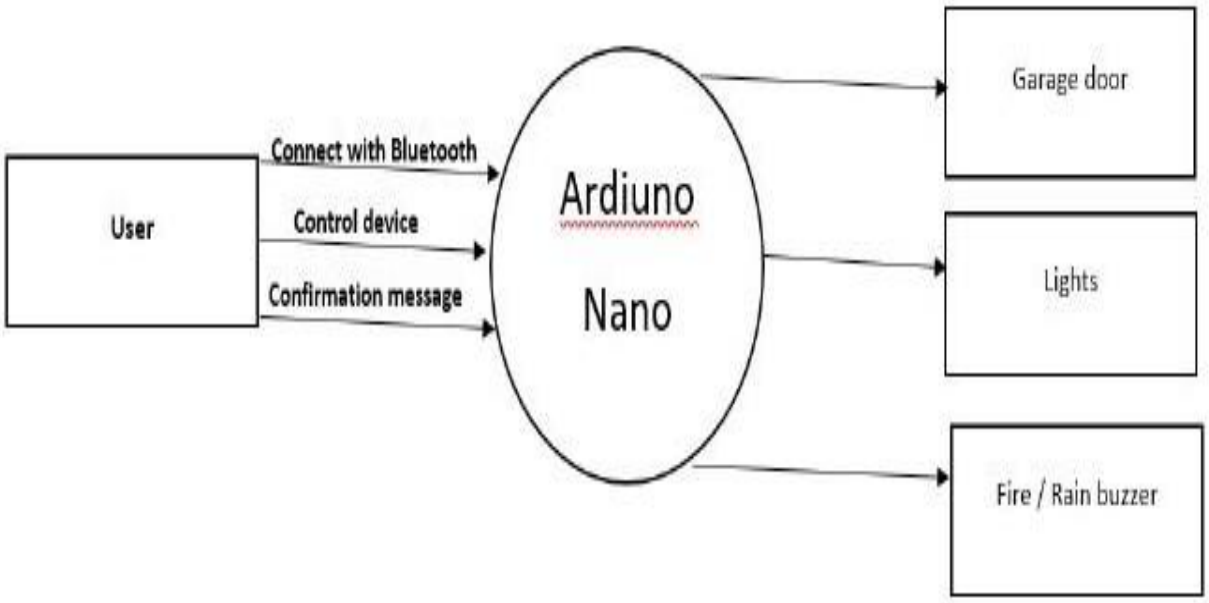
Chapter Three. Analysis

1.Diagrams

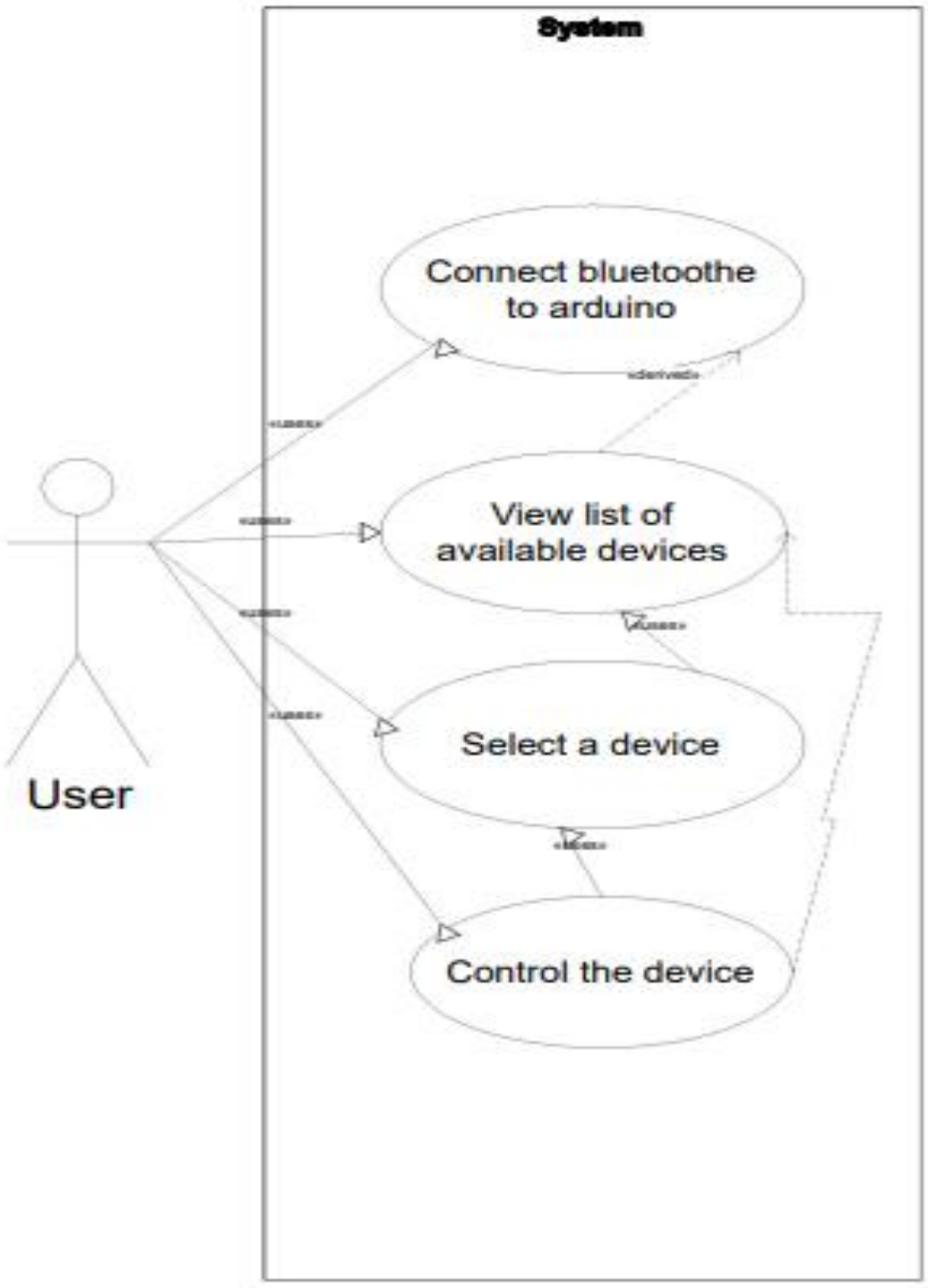
1. Main Use Case Diagram



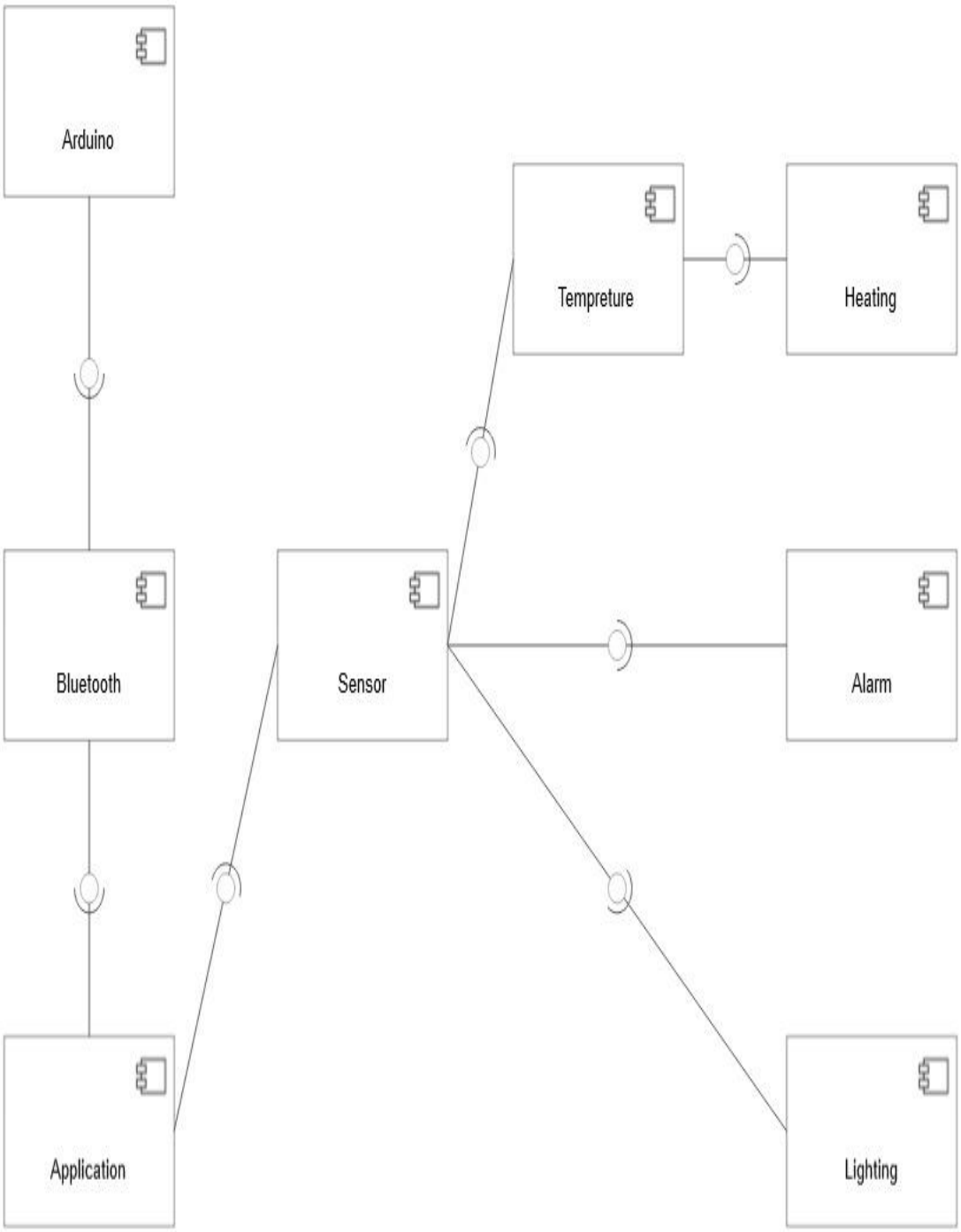
2.Context Diagram



3. Sub Use Case



4.Component Diagram



Chapter Four. Design

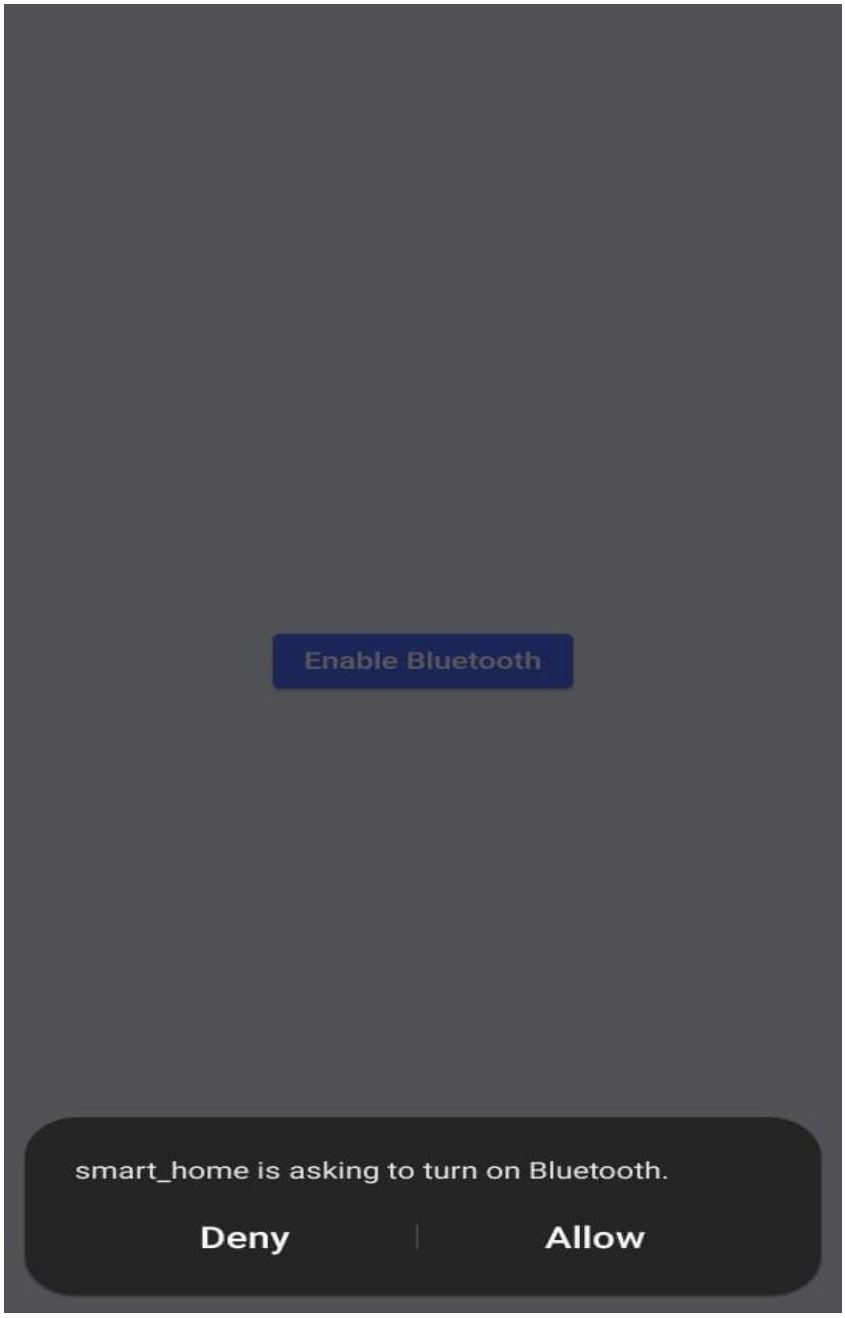
Smart Home Application

The Smart Home Application allows you to connect your Bluetooth-enabled home to your smartphone, giving you the ability to control your home's lighting and give voice commands using features similar to Siri and Google Assistant. Additionally, the app can display Alarm screens automatically for, rain, or fire alerts through the use of sensors

Page One

This is first page where we can open Bluetooth to search for devices, and if any permissions are required, firstly, we need to accept them.

After pressing the button “Enable Bluetooth”, the message to accept permissions and enable bluetooth will appear.



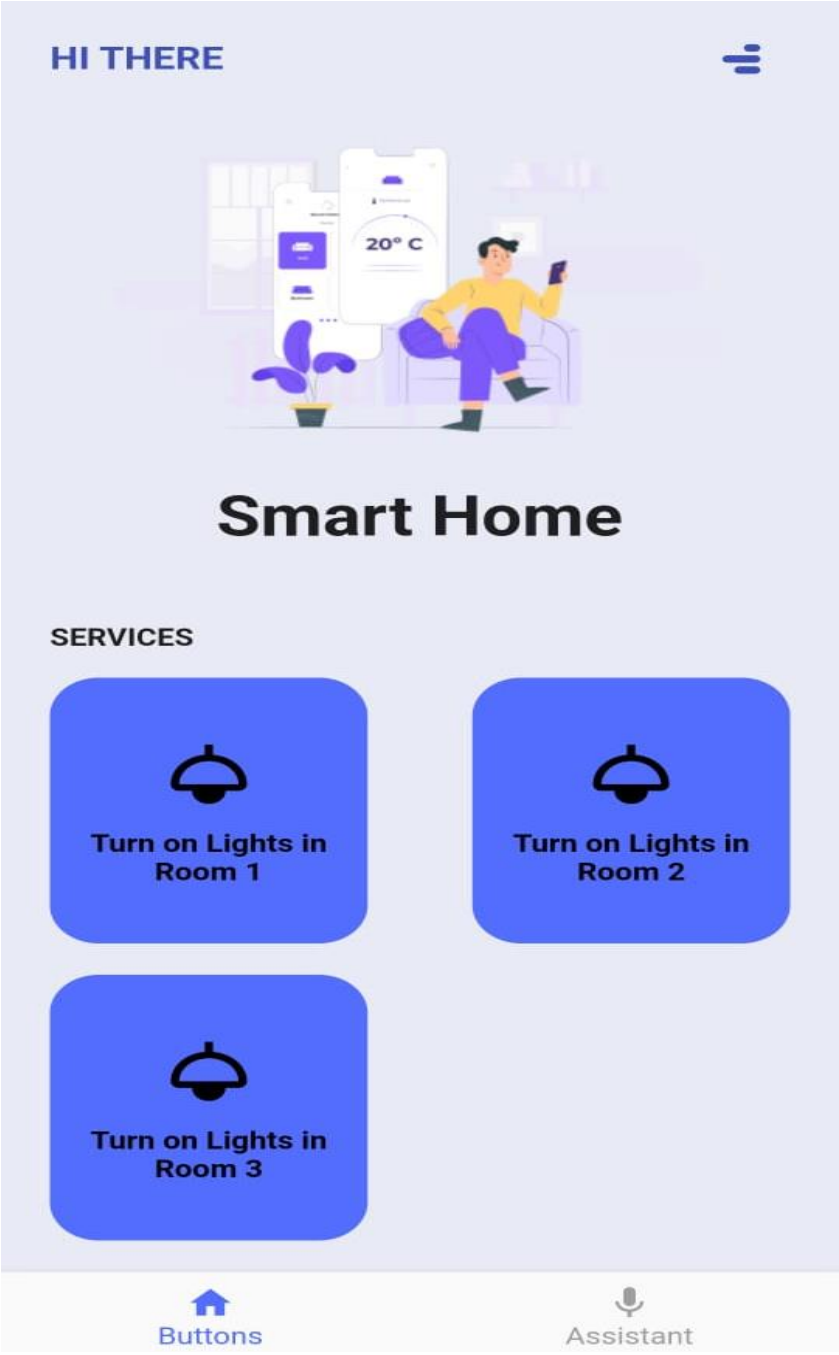
Page Two

On this page we choose the Bluetooth of the Arduino

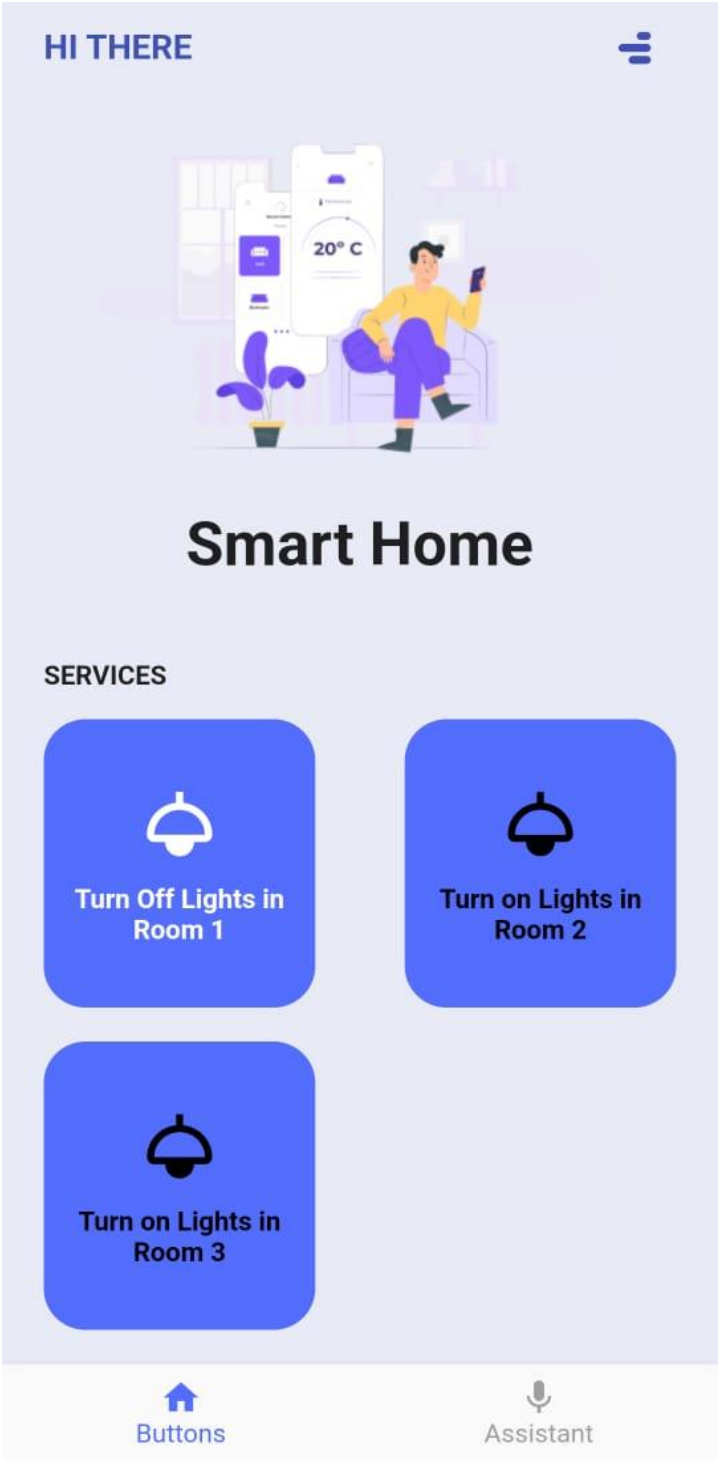
After choosing the Bluetooth of the Arduino, this page will appear and Bluetooth is trying to connect to it

Page Three and main Page

We have three buttons, when pressing button for specific room, the light will be turned on or off.

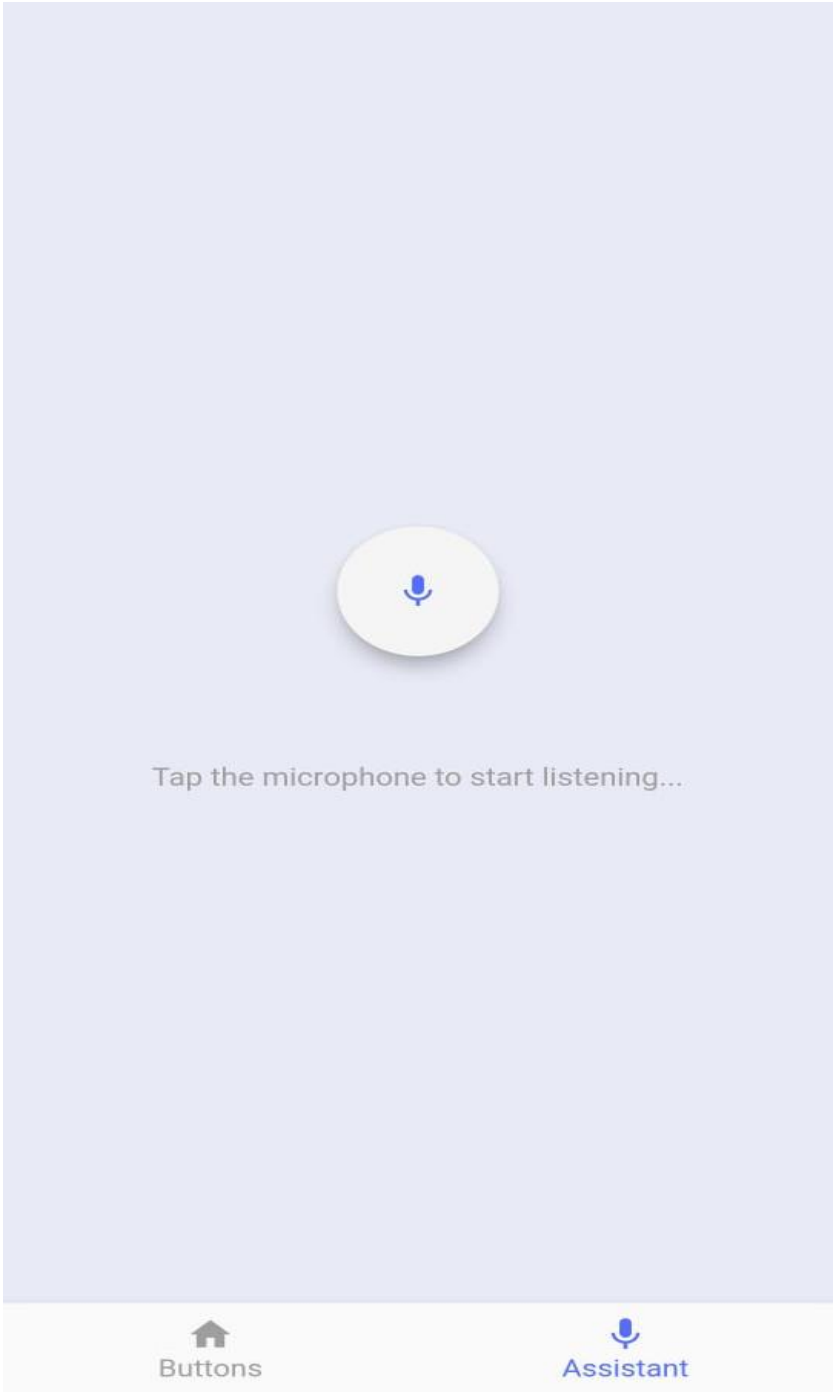


This is what the page looks like when room number one is pressed.

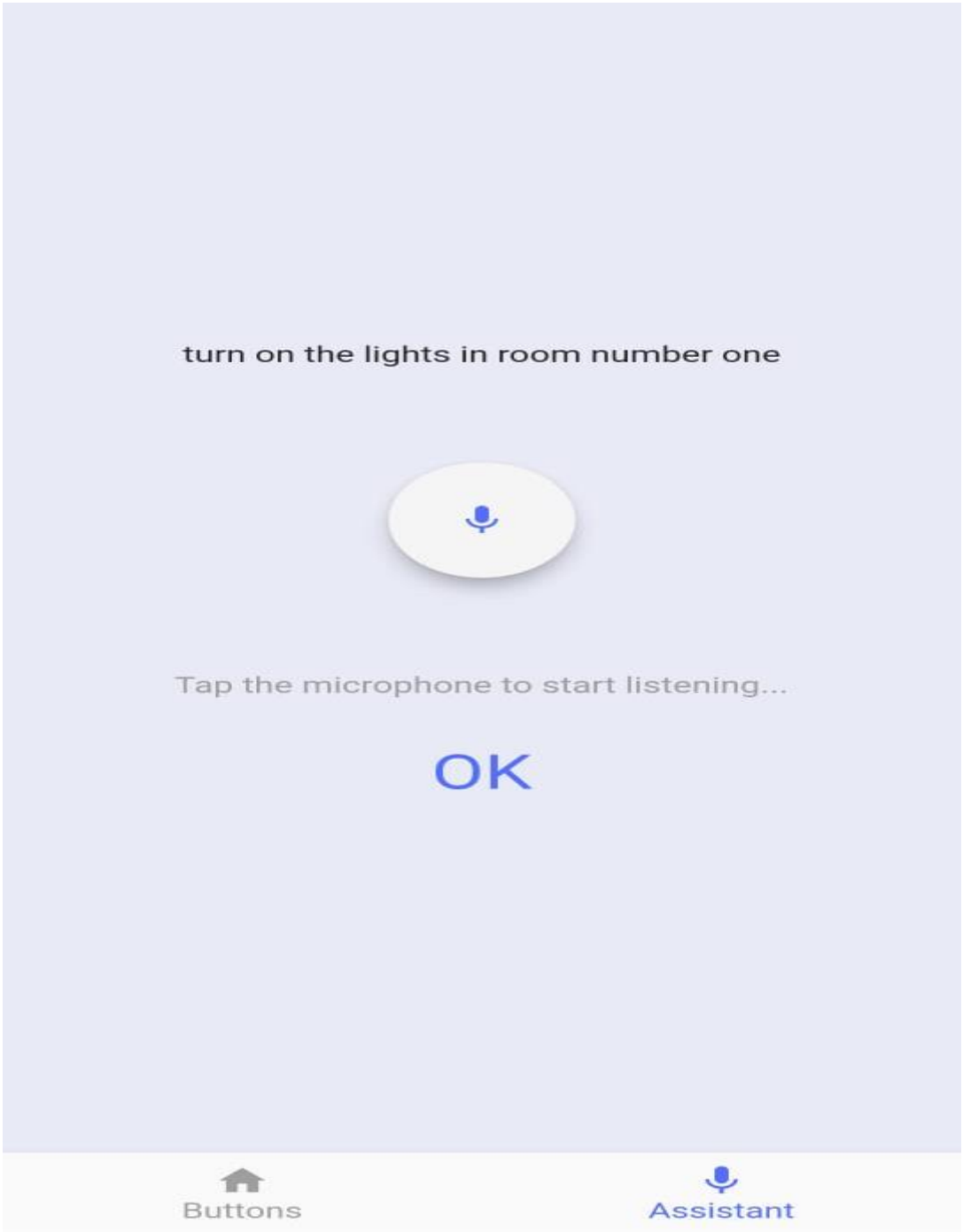


Page Four

This is a voice assistant instead of buttons, we can give any command, and the assistant will do it.



Here's an example: If we pressed the microphone button and try to turn on room number one, the assistant will do it.



Page Five

On this page we can find Assistant instructions, we open it and tell us what we can say to the assistant and what it can do.

Page Six

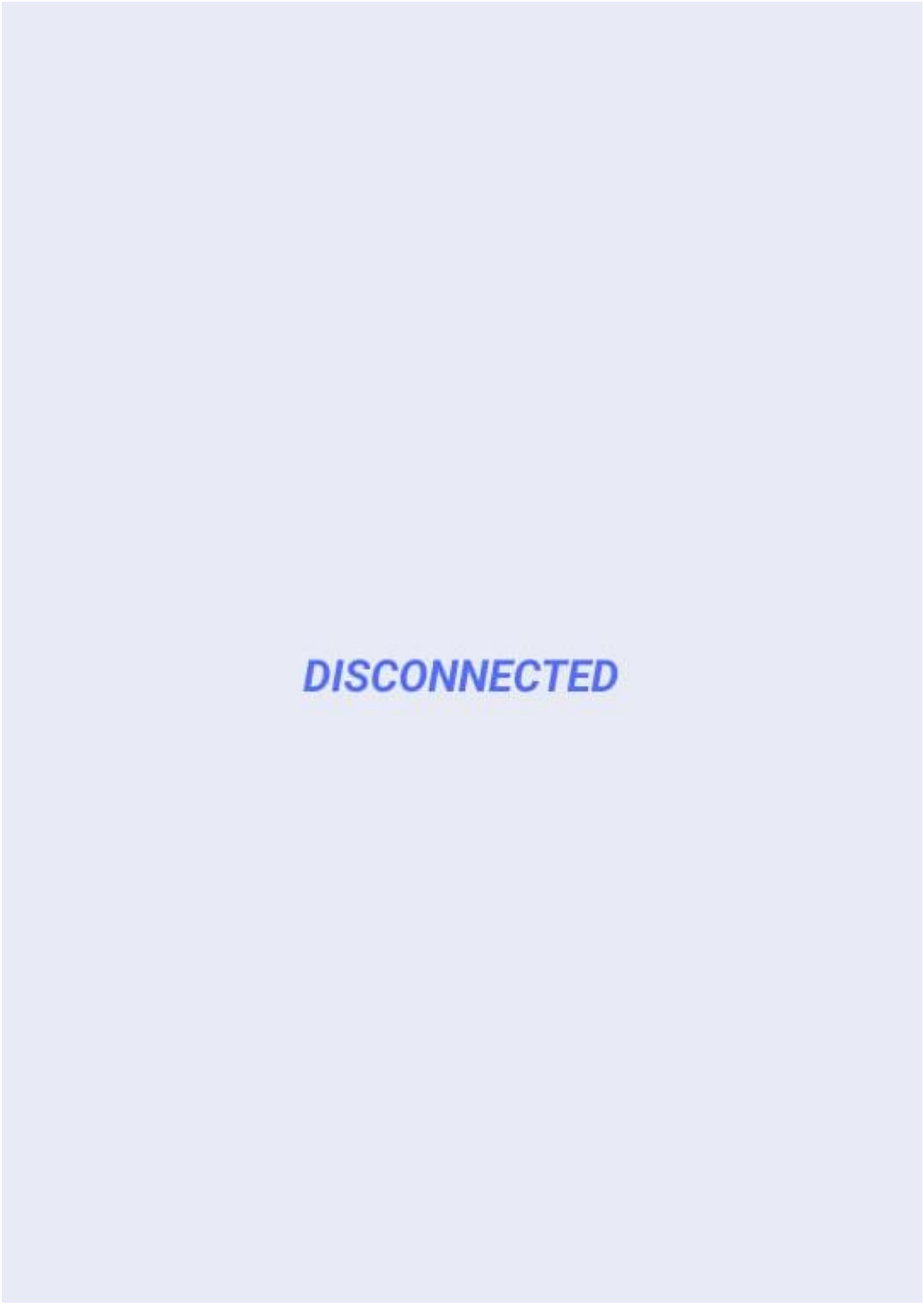
This page opens automatically if there is water on the water sensor and it makes a rain sound.

Page Seven

This page opens automatically if there is a fire on the fire sensor and it makes an alarm sound for fire.

Page Eight and the last Page

This page is displayed if the Bluetooth connection is lost.



Chapter Five. Implementation and Tools

The project involves a solar tracking system that tracks the sun's movement throughout the day and stores the generated solar energy in batteries. These batteries are then used to power a smart home system that utilizes Bluetooth technology to control lighting and other devices.

Arduino:

The Arduino board can be used to design a smart lighting system that controls LEDs and other lighting devices. It can be programmed to respond to various inputs, such as fire sensors or light and water sensors. Additionally, the system can be connected to Bluetooth and remotely controlled using a smartphone or other device. Due to its flexibility and ease of use, Arduino is a popular choice for DIY smart home projects, including smart lighting.

The code of Arduino

```
#include <Servo.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <SoftwareSerial.h>

#define THRESHOLD 100
#define DTIME 50
#define SENSOR_TL A0  //Top Left sensor pin
#define SENSOR_TR A3  //Top Right sensor pin
#define SENSOR_BL A1  //Bottom Left sensor pin
#define SENSOR_BR A2  //Bottom Right sensor pin

#define HORIZ_LIMIT 180 //Horizontal limit should be the full rotation of a servo
#define VERT_LIMIT 90   //The value is determined depending on the mechanical setup (the angle at which the vertical servo keeps the panel parallel to ground)

Servo horiz;        //Servo for horizontal movement
Servo vert;         //Servo for vertical movement
int horizpos = 90;
int vertpos = 90;

//Smart Home
SoftwareSerial myserial(0, 1); // Bluetooth: Tx = 5, Rx = 6
LiquidCrystal_I2C lcd(0x27, 16, 2);
// rain
#define ra A6
int value;

//garage
#include <Servo.h>

const uint8_t sensor1_pin = 5;
const uint8_t sensor2_pin = 6;
const uint8_t servo_pin = 11;

const uint8_t sensor_threshold = 1;

Servo myServo;


void setup()
{
  Serial.begin(9600);
  horiz.attach(9);
  vert.attach(10);
  vert.write(90);
  horiz.write(90);
  delay(1000);


//Smart Home

  myserial.begin(9600);
  Serial.begin(9600);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(11, OUTPUT);
  myserial.write("Hello in Bluetooth");

  lcd.init();
  lcd.backlight();

  pinMode(A6, INPUT);

  // GAS
  pinMode(A7, INPUT);

  //garage
  pinMode(sensor1_pin, INPUT);
  pinMode(sensor2_pin, INPUT);
  myServo.attach(servo_pin);
}

void loop() {
   track();
   gas();
   rain();
   Lighting();
   garage();
}

void track() {
  int tl = analogRead(SENSOR_TL); //Read the value of the TL sensor
  int tr = analogRead(SENSOR_TR); //Read the value of the TR sensor
  int bl = analogRead(SENSOR_BL); //Read the value of the BL sensor
  int br = analogRead(SENSOR_BR); //Read the value of the BR sensor

  int average_top = (tl + tr) / 2;
  int average_down = (bl + br) / 2;
  int average_left = (tl + bl) / 2;
  int average_right = (tr + br) / 2;

  int dif_vert = average_top - average_down;
  int dif_horz = average_left - average_right;

```

```
  if ( ((-1 * THRESHOLD) <= dif_vert) && (dif_vert <= THRESHOLD) ) {
    vert.write(90);
  }
  else {
    vert.attach(10);

    if (average_top > average_down)
    {
      vertpos = --vertpos;
      if (vertpos > VERT_LIMIT)
      {
        vert.detach();
        vertpos = VERT_LIMIT;
      }
    }
    else if (average_top < average_down)
    {
      vertpos = ++vertpos;
      if (vertpos < 0)
      {
        vert.detach();
        vertpos = 0;
      }
    }
    vert.write(vertpos);
  }

  if ( ((-1 * THRESHOLD) <= dif_horz) && (dif_horz <= THRESHOLD) ) {
    horiz.write(90);
  }
  else {
    horiz.attach(9);

    if (average_left > average_right)
    {
      horizpos = --horizpos;
      if (horizpos < 0)
      {
        horiz.detach();
        horizpos = 0;
      }
    }
    else if (average_left < average_right)
    {
      horizpos = ++horizpos;
      if (horizpos > HORIZ_LIMIT)
      {
         horiz.detach();
         horizpos = HORIZ_LIMIT;
       }
     }
     horiz.write(horizpos);
   }

   delay(DTIME);
 }

 ////////////////////////////////////////////////////////////////////////////////////////////////////

 //Smart Home

 // GAS
 void gas()
 {
   int val = analogRead(A7);
   if (val > 503) {
     digitalWrite(3, HIGH);
     lcd.setCursor(0,0);
     lcd.print("Fire  on");
     myserial.write("15");
     Serial.println(15);
     delay(1000);

   } else {
     digitalWrite(3, LOW);
     lcd.setCursor(0,0);
    lcd.print("          ");
   }

  //Serial.println(val);
   //delay(200);
 }


 // RAIN
 void rain()
 {
   value = analogRead(ra);
   if (value >= 200) {
     lcd.setCursor(0,1);
     lcd.print("Rain on ");
     Serial.println("16");
     myserial.write("16");
   }
```

```
190
191    else {
192      digitalWrite(11, LOW);
193      lcd.setCursor(0,1);
194      lcd.print("          ");
195      //myserial.write("0");
196    }
197
198    //Serial.println("RAIN = " + String(value));
199    //delay(100);
200  }
201
202  //Lighting
203
204   void Lighting()
205   {
206
207    if (myserial.available()) {
208      int x = myserial.read();
209      Serial.println(x);
210
211      // Room 1 (living room)
212      if (x == '1') {
213        digitalWrite(4, HIGH);
214      } else if (x == '2') {
215        digitalWrite(4, LOW);
216      }
217
218      // Room 2 (crush)
219      if (x == '3') {
220        digitalWrite(7, HIGH);
221      } else if (x == '4') {
222        digitalWrite(7, LOW);
223      }
224
225      // Room 3 (kitchen)
226      if (x == '5') {
227        digitalWrite(8, HIGH);
228      } else if (x == '6') {
229        digitalWrite(8, LOW);
230      }
231    }
232  /*
233    if (Serial.available()) {
234      myserial.write(Serial.read());
235    }

236     */
237   }
238
239   void garage() {
240       bool sensor1_value = digitalRead(sensor1_pin);
241     bool sensor2_value = digitalRead(sensor2_pin);
242
243     if (sensor1_value == LOW) {
244       // Open the door
245       myServo.write(180); // Rotate the servo to open the door
246     }
247
248     if (sensor2_value == LOW) {
249       // Close the door
250       myServo.write(0); // Rotate the servo to close the door
251     }
252   }
```

Application:

This application is designed to provide users with the ability to control their lights and open alarm screen in case of fire or rain. It was built using various technologies and libraries, including..

(flutter_bluetooth_serial, permission_handler, speech_to_text, flutter_tts, avatar_glow, and audioplayers.)

- Flutter_bluetooth_serial: to Bluetooth connectivity to control the lights, allowing users to turn them on and off remotely. Additionally, and the App can open alarm screen Automatic in case of fire or rain This makes sure that users know about any possible dangers.

-Permission_handler: Used to manage access permissions in mobile applications in an easy and secure way.

-Audioplayers: Used to plays Rain sound when the rain sensor is activated, and plays Alarm sound when the fire sensor is activated.

-Speech_to_text and flutter_tts : Libraries are used to enable the app to convert speech to text and vice versa in the assistant screen,

-avatar_glow: Used to make some animation in assistant screen.

The code of application:

1- Main.dart:

```dart
import 'package:flutter/material.dart';
import 'package:smart_home/screens/connection_screen.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      debugShowCheckedModeBanner: false,
      home: ConnectionScreen(),
    );
  }
}
```

2- connection_screen.dart:

```dart
import 'package:flutter/material.dart';
import 'package:flutter_bluetooth_serial/flutter_bluetooth_serial.dart';
import 'package:permission_handler/permission_handler.dart';

import '../widgets/scan_widget.dart';

class ConnectionScreen extends StatefulWidget {
  const ConnectionScreen({super.key});

  @override
  State<ConnectionScreen> createState() => _ConnectionScreenState();
}

class _ConnectionScreenState extends State<ConnectionScreen> {
  Future<bool> bluetoothGranted() async {
    bool pgranted = await Permission.bluetooth.isGranted;
    if (pgranted) {
      bool? bgranted = await FlutterBluetoothSerial.instance.isEnabled;
      if (bgranted != null && bgranted) {
        setState(() {});
        return true;
      } else {
        setState(() {});
        return false;
      }
    } else {
      setState(() {});
      return false;
    }
  }

  @override
  Widget build(BuildContext context) {
```

```dart
    return Scaffold(
      backgroundColor: Colors.indigo.shade50,
      body: SafeArea(
        child: FutureBuilder(
            future: bluetoothGranted(),
            builder: (context, snapshot) {
              if (snapshot.data == true) {
                return const ScanWidget();
              } else {
                return Center(
                  child: ElevatedButton(
                    style: ButtonStyle(
                        backgroundColor:
                            MaterialStateProperty.all(Colors.indigoAccent)),
                    onPressed: () async {
                      await Permission.bluetooth.request();
                      await Permission.bluetoothConnect.request();
                      await Permission.bluetoothScan.request();
                      await Permission.bluetoothAdvertise.request();

                      bool isBluetoothEnabled =
                          await Permission.bluetooth.isGranted;
                      if (isBluetoothEnabled) {
                        await FlutterBluetoothSerial.instance.requestEnable();
                        setState(() {});
                      }
                    },
                    child: const Text("Enable Bluetooth"),
                  ),
                );
              }
            }),
      ),
    );
  }
}
```

3- scan_widget.dart:

```dart
import 'dart:async';

import 'package:flutter/material.dart';
import 'package:flutter_bluetooth_serial/flutter_bluetooth_serial.dart';

import '../screens/smart_home_screen.dart';
import 'BluetoothDeviceListEntry.dart';

class ScanWidget extends StatefulWidget {
  final bool start;
  const ScanWidget({super.key, this.start = true});

  @override
  State<ScanWidget> createState() => _ScanWidgetState();
}

class _ScanWidgetState extends State<ScanWidget> {
  late StreamSubscription<BluetoothDiscoveryResult> _streamSubscription;
  List<BluetoothDiscoveryResult> results = [];
  late bool isDiscovering;

  @override
  void initState() {
    super.initState();

    isDiscovering = widget.start;
    if (isDiscovering) {
      _startDiscovery();
    }
  }

  void _restartDiscovery() {
    if (!isDiscovering) {
      setState(() {
```

```dart
        results.clear();
        isDiscovering = true;
      });

      _startDiscovery();
    } else {
      ScaffoldMessenger.of(context).showSnackBar(const SnackBar(
          content: Text("You can't refresh when still scanning")));
    }
  }

  void _startDiscovery() {
    _streamSubscription =
        FlutterBluetoothSerial.instance.startDiscovery().listen((r) {
      setState(() {
        bool inResult = false;
        for (var element in results) {
          if (element.device == r.device) {
            inResult = true;
          }
        }
        if (!inResult) {
          results.add(r);
        }
      });
    });

    _streamSubscription.onDone(() {
      setState(() {
        isDiscovering = false;
      });
    });
  }

  // @TODO . One day there should be `_pairDevice` on long tap on something... ;)
```

```
70
71    @override
72    void dispose() {
73      // Avoid memory leak (`setState` after dispose) and cancel discovery
74      _streamSubscription.cancel();
75
76      super.dispose();
77    }
78
79    @override
80    Widget build(BuildContext context) {
81      return RefreshIndicator(
82        onRefresh: () async => _restartDiscovery(),
83        child: ListView.builder(
84          itemCount: results.length,
85          itemBuilder: (BuildContext context, index) {
86            BluetoothDiscoveryResult result = results[index];
87            return BluetoothDeviceListEntry(
88              device: result.device,
89              rssi: result.rssi,
90              onTap: () {
91                Navigator.of(context).push(
92                  MaterialPageRoute(
93                    builder: (context) {
94                      return SmartHomeScreen(server: result.device);
95                    },
96                  ),
97                );
98              },
99            );
100         },
101       ),
102     );
103   }
104 }
```

4-BluetoothDeviceListEntry.dart:

```dart

import 'package:flutter/material.dart';
import 'package:flutter_bluetooth_serial/flutter_bluetooth_serial.dart';

class BluetoothDeviceListEntry extends ListTile {
  BluetoothDeviceListEntry({super.key,
    required BluetoothDevice device,
    required rssi,
    required GestureTapCallback onTap,
    bool enabled = true,
  }) : super(
        onTap: onTap,
        enabled: enabled,
        leading: const Icon(Icons.devices),
        // @TODO . !BluetoothClass! class aware icon
        title: Text(device.name ?? "Unknown device"),
        subtitle: Text(device.address.toString()),
        trailing: Row(
          mainAxisSize: MainAxisSize.min,
          children: <Widget>[
            rssi != null
                ? Container(
                    margin: const EdgeInsets.all(8.0),
                    child: DefaultTextStyle(
                      style: _computeTextStyle(rssi),
                      child: Column(
                        mainAxisSize: MainAxisSize.min,
                        children: <Widget>[
                          Text(rssi.toString()),
                          const Text('dBm'),
                        ],
                      ),
                    ),
                  )
                : const SizedBox(width: 0, height: 0),
            device.isConnected
                ? const Icon(Icons.import_export)
                : const SizedBox(width: 0, height: 0),
```

```dart
                device.isBonded
                    ? const Icon(Icons.link)
                    : const SizedBox(width: 0, height: 0),
              ],
            ),
          );

    static TextStyle _computeTextStyle(int rssi) {
      /**/ if (rssi >= -35) {
        return TextStyle(color: Colors.greenAccent[700]);
      } else if (rssi >= -45) {
        return TextStyle(
            color: Color.lerp(
                Colors.greenAccent[700], Colors.lightGreen, -(rssi + 35) / 10));
      } else if (rssi >= -55) {
        return TextStyle(
            color: Color.lerp(
                Colors.lightGreen, Colors.lime[600], -(rssi + 45) / 10));
      } else if (rssi >= -65) {
        return TextStyle(
            color: Color.lerp(Colors.lime[600], Colors.amber, -(rssi + 55) / 10));
      } else if (rssi >= -75) {
        return TextStyle(
            color: Color.lerp(
                Colors.amber, Colors.deepOrangeAccent, -(rssi + 65) / 10));
      } else if (rssi >= -85) {
        return TextStyle(
            color: Color.lerp(
                Colors.deepOrangeAccent, Colors.redAccent, -(rssi + 75) / 10));
      } else {
        /*code symetry*/
        return const TextStyle(color: Colors.redAccent);
      }
    }
  }
```

5-smart_home_screen.dart:

```dart
import 'dart:convert';
import 'dart:typed_data';

import 'package:avatar_glow/avatar_glow.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bluetooth_serial/flutter_bluetooth_serial.dart';
import 'package:flutter_tts/flutter_tts.dart';
import 'package:speech_to_text/speech_recognition_result.dart';
import 'package:speech_to_text/speech_to_text.dart';

import 'assistant_instructions_screen.dart';
import 'sensor_running_screen.dart';

class SmartHomeScreen extends StatefulWidget {
  final BluetoothDevice server;

  const SmartHomeScreen({super.key, required this.server});

  @override
  State<SmartHomeScreen> createState() => _SmartHomeScreen();
}

class _Message {
  int whom;
  String text;

  _Message(this.whom, this.text);
}

class _SmartHomeScreen extends State<SmartHomeScreen> {
  var connection; //BluetoothConnection

  bool isConnecting = true;
  bool isDisconnecting = false;

  // bottom navigation bar
  int selectedIndex = 0;
  void _onItemTapped(int index) {
    setState(() {
      selectedIndex = index;
    });
  }

  // tts
  FlutterTts flutterTts = FlutterTts();

  // stt
  final SpeechToText _speechToText = SpeechToText();
  bool _speechEnabled = false;
  String _lastWords = '';
```

```
51
52      void _initSpeech() async {
53        _speechEnabled = await _speechToText.initialize();
54        setState(() {});
55      }
56
57      void _startListening() async {
58        await _speechToText.listen(onResult: _onSpeechResult);
59        setState(() {
60          _indexOfResponses = null;
61        });
62      }
63
64      void _stopListening() async {
65        await _speechToText.stop();
66        setState(() {});
67      }
68
69      final List<String> _responses = [
70        "OK",
71        "Hello",
72        "My name is Makkah",
73        "I\'m your assistant in your smart home",
74        "Thank You",
75      ];
76
77      int? _indexOfResponses;
78
79      _doByVoice(words) async {
80        if (words.toLowerCase() == 'turn on the light in room number one' ||
81            words.toLowerCase() == 'turn on the lights in room number one' ||
82            words.toLowerCase() == 'turn on the lights in first room' ||
83            words.toLowerCase() == 'turn on the light in first room') {
84          await _sendMessage('1').then((value) async {
85            setState(() {
86              _toggle_lights_room1 = true;
87              _indexOfResponses = 0;
88            });
89            await flutterTts.setLanguage("en-US");
90
91            await flutterTts.setSpeechRate(0.8);
92
93            await flutterTts.setVolume(1.0);
94
95            await flutterTts.speak(_responses[_indexOfResponses!]);
96          });
97        } else if (words.toLowerCase() == 'turn off the light in room number one' ||
98            words.toLowerCase() == 'turn off the lights in room number one' ||
99            words.toLowerCase() == 'turn off the lights in first room' ||
100           words.toLowerCase() == 'turn off the light in first room') {
```

```
101       await _sendMessage('2').then((value) async {
102         setState(() {
103           _toggle_lights_room1 = false;
104           _indexOfResponses = 0;
105         });
106         await flutterTts.setLanguage("en-US");
107
108         await flutterTts.setSpeechRate(0.8);
109
110         await flutterTts.setVolume(1.0);
111
112         await flutterTts.speak(_responses[_indexOfResponses!]);
113       });
114     } else if (words.toLowerCase() == 'turn on the light in room number two' ||
115         words.toLowerCase() == 'turn on the lights in room number two' ||
116         words.toLowerCase() == 'turn on the lights in second room' ||
117         words.toLowerCase() == 'turn on the light in second room') {
118       await _sendMessage('3').then((value) async {
119         setState(() {
120           _toggle_lights_room2 = true;
121           _indexOfResponses = 0;
122         });
123         await flutterTts.setLanguage("en-US");
124
125         await flutterTts.setSpeechRate(0.8);
126
127         await flutterTts.setVolume(1.0);
128
129         await flutterTts.speak(_responses[_indexOfResponses!]);
130       });
131     } else if (words.toLowerCase() == 'turn off the light in room number two' ||
132         words.toLowerCase() == 'turn off the lights in room number two' ||
133         words.toLowerCase() == 'turn off the lights in second room' ||
134         words.toLowerCase() == 'turn off the light in second room') {
135       await _sendMessage('4').then((value) async {
136         setState(() {
137           _toggle_lights_room2 = false;
138           _indexOfResponses = 0;
139         });
140         await flutterTts.setLanguage("en-US");
141
142         await flutterTts.setSpeechRate(0.8);
143
144         await flutterTts.setVolume(1.0);
145
146         await flutterTts.speak(_responses[_indexOfResponses!]);
147       });
148     } else if (words.toLowerCase() ==
149           'turn on the light in room number three' ||
150         words.toLowerCase() == 'turn on the lights in room number three' ||
```

```
        words.toLowerCase() == 'turn on the lights in third room' ||
        words.toLowerCase() == 'turn on the light in third room') {
      await _sendMessage('5').then((value) async {
        setState(() {
          _toggle_lights_room3 = true;
          _indexOfResponses = 0;
        });
        await flutterTts.setLanguage("en-US");

        await flutterTts.setSpeechRate(0.8);

        await flutterTts.setVolume(1.0);

        await flutterTts.speak(_responses[_indexOfResponses!]);
      });
    } else if (words.toLowerCase() ==
            'turn off the light in room number three' ||
        words.toLowerCase() == 'turn off the lights in room number three' ||
        words.toLowerCase() == 'turn off the lights in third room' ||
        words.toLowerCase() == 'turn off the light in third room') {
      await _sendMessage('6').then((value) async {
        setState(() {
          _toggle_lights_room3 = false;
          _indexOfResponses = 0;
        });
        await flutterTts.setLanguage("en-US");

        await flutterTts.setSpeechRate(0.8);

        await flutterTts.setVolume(1.0);

        await flutterTts.speak(_responses[_indexOfResponses!]);
      });
    } else if (words.toLowerCase() == 'hello') {
      await flutterTts.setLanguage("en-US");

      await flutterTts.setSpeechRate(0.4);

      await flutterTts.setVolume(1.0);
      setState(() {
        _indexOfResponses = 1;
      });
      await flutterTts.speak(_responses[_indexOfResponses!]);
    } else if (words.toLowerCase() == "what's your name" ||
        words.toLowerCase() == 'what is your name') {
      await flutterTts.setLanguage("en-US");

      await flutterTts.setSpeechRate(0.4);

      await flutterTts.setVolume(1.0);
```

```
        setState(() {
          _indexOfResponses = 2;
        });
        await flutterTts.speak(_responses[_indexOfResponses!]);
      } else if (words.toLowerCase() == "what are you doing" ||
          words.toLowerCase() == 'who are you') {
        await flutterTts.setLanguage("en-US");

        await flutterTts.setSpeechRate(0.4);

        await flutterTts.setVolume(1.0);
        setState(() {
          _indexOfResponses = 3;
        });
        await flutterTts.speak(_responses[_indexOfResponses!]);
      } else if (words.toLowerCase() == "you are great" ||
          words.toLowerCase() == 'good job' ||
          words.toLowerCase() == "you're great") {
        await flutterTts.setLanguage("en-US");

        await flutterTts.setSpeechRate(0.4);

        await flutterTts.setVolume(1.0);
        setState(() {
          _indexOfResponses = 4;
        });
        await flutterTts.speak(_responses[_indexOfResponses!]);
      }
    }

    Future<void> _onSpeechResult(SpeechRecognitionResult result) async {
      setState(() {
        _lastWords = result.recognizedWords;
      });
      await _doByVoice(_lastWords);
    }

    @override
    void initState() {
      super.initState();
      _initSpeech();

      BluetoothConnection.toAddress(widget.server.address).then((_connection) {
        print('Connected to the device');
        connection = _connection;
        setState(() {
          isConnecting = false;
          isDisconnecting = false;
        });

```

```
251         connection.input.listen(_onDataReceived).onDone(() {
252           // Example: Detect which side closed the connection
253           // There should be `isDisconnecting` flag to show are we are (locally)
254           // in middle of disconnecting process, should be set before calling
255           // `dispose`, `finish` or `close`, which all causes to disconnect.
256           // If we except the disconnection, `onDone` should be fired as result.
257           // If we didn't except this (no flag set), it means closing by remote.
258           if (isDisconnecting) {
259             print('Disconnecting locally!');
260           } else {
261             print('Disconnected remotely!');
262           }
263           if (mounted) {
264             setState(() {});
265           }
266         });
267       }).catchError((error) {
268         print('Cannot connect, exception occured');
269         print(error);
270       });
271     }
272
273     @override
274     void dispose() {
275       // Avoid memory leak (`setState` after dispose) and disconnect
276       if (isConnected()) {
277         isDisconnecting = true;
278         connection.dispose();
279         connection = null;
280       }
281
282       super.dispose();
283     }
284
285     bool _toggle_lights_room1 = false;
286     bool _toggle_lights_room2 = false;
287     bool _toggle_lights_room3 = false;
288     //bool _toggle_window = false;
289
290     @override
291     Widget build(BuildContext context) {
292       // final List<Row> list = messages.map((message) {
293       //   return Row(
294       //     mainAxisAlignment: message.whom == clientID
295       //         ? MainAxisAlignment.end
296       //         : MainAxisAlignment.start,
297       //     children: <Widget>[
298       //       Container(
299       //         padding: const EdgeInsets.all(12.0),
300       //         margin: const EdgeInsets.only(bottom: 8.0, left: 8.0, right: 8.0),
```

```
//          width: 222.0,
//          decoration: BoxDecoration(
//              color:
//                  message.whom == clientID ? Colors.blueAccent : Colors.grey,
//              borderRadius: BorderRadius.circular(7.0)),
//          child: Text(
//              (text) {
//                return text == '/shrug' ? '¯\\_(ツ)_/¯' : text;
//              }(message.text.trim()),
//              style: const TextStyle(color: Colors.white)),
//        ),
//      ],
//    );
// }).toList();

return Scaffold(
  backgroundColor: Colors.indigo.shade50,
  body: SafeArea(
    child: isConnecting
        ? const Center(
            child: Text(
              "WAIT UNTIL CONNECTED ...",
              style: TextStyle(
                fontSize: 18,
                color: Colors.indigoAccent,
                fontWeight: FontWeight.bold,
                fontStyle: FontStyle.italic,
              ),
            ),
          )
        : isConnected()
            ? selectedIndex == 0
                ? GestureDetector(
                    onTap: () {},
                    child: Container(
                      margin: const EdgeInsets.only(
                          top: 18, left: 24, right: 24),
                      child: Column(
                        children: <Widget>[
                          Row(
                            mainAxisAlignment:
                                MainAxisAlignment.spaceBetween,
                            children: [
                              const Text(
                                "HI THERE",
                                style: TextStyle(
                                  fontSize: 18,
                                  color: Colors.indigo,
                                  fontWeight: FontWeight.bold,
                                ),
```

```
                              ),
                              IconButton(
                                onPressed: () {
                                  Navigator.push(
                                      context,
                                      MaterialPageRoute(
                                          builder: (conte) =>
                                              const AssistantInstructionsScreen()));
                                },
                                icon: const RotatedBox(
                                  quarterTurns: 135,
                                  child: Icon(
                                    Icons.bar_chart_rounded,
                                    size: 28,
                                    color: Colors.indigo,
                                  ),
                                ),
                              ),
                            ],
                          ),
                          Expanded(
                            child: ListView(
                              physics: const BouncingScrollPhysics(),
                              children: [
                                const SizedBox(
                                  height: 32,
                                ),
                                Center(
                                  child: Image.asset(
                                    "assets/images/banner.png",
                                    scale: 1.2,
                                  ),
                                ),
                                const SizedBox(
                                  height: 32,
                                ),
                                const Center(
                                  child: Text(
                                    "Smart Home",
                                    style: TextStyle(
                                      fontSize: 32,
                                      fontWeight: FontWeight.bold,
                                    ),
                                  ),
                                ),
                                const SizedBox(height: 48),
                                const Text(
                                  "SERVICES",
                                  style: TextStyle(
                                      fontSize: 14,
```

```
                    fontWeight: FontWeight.bold),
              ),
              const SizedBox(
                height: 16,
              ),
              Row(
                mainAxisAlignment:
                    MainAxisAlignment.spaceBetween,
                children: [
                  _cardMenu(
                    title: !_toggle_lights_room1
                        ? "Turn on Lights in Room 1"
                        : "Turn Off Lights in Room 1",
                    icon: Icons.light,
                    color: Colors.indigoAccent,
                    fontColor: !_toggle_lights_room1
                        ? Colors.black
                        : Colors.white,
                    onTap: isConnected()
                        ? () async {
                            await _sendMessage(
                                    _toggle_lights_room1
                                        ? "2"
                                        : "1")
                                .then((value) {
                              setState(() {
                                _toggle_lights_room1 =
                                    !_toggle_lights_room1;
                              });
                            });
                          }
                        : null,
                  ),
                  _cardMenu(
                    title: !_toggle_lights_room2
                        ? "Turn on Lights in Room 2"
                        : "Turn Off Lights in Room 2",
                    icon: Icons.light,
                    color: Colors.indigoAccent,
                    fontColor: !_toggle_lights_room2
                        ? Colors.black
                        : Colors.white,
                    onTap: isConnected()
                        ? () async {
                            await _sendMessage(
                                    _toggle_lights_room2
                                        ? "4"
                                        : "3")
                                .then((value) {
                              setState(() {
```

```
                                _toggle_lights_room2 =
                                    !_toggle_lights_room2;
                              });
                            });
                          }
                        : null,
                  ),

                  // _cardMenu(
                  //   title: !_toggle_window
                  //       ? "Open The Window"
                  //       : "Close The Window",
                  //   icon: !_toggle_window
                  //       ? Icons.square
                  //       : Icons.crop_square_sharp,
                  //   color: Colors.indigoAccent,
                  //   fontColor: !_toggle_window
                  //       ? Colors.black
                  //       : Colors.white,
                  //   onTap: isConnected()
                  //       ? () async {
                  //           await _sendMessage(
                  //                   _toggle_window ? "-2" : "2")
                  //               .then((value) {
                  //             setState(() {
                  //               _toggle_window =
                  //                   !_toggle_window;
                  //             });
                  //           });
                  //         }
                  //       : null,
                  // ),
                ],
              ),
              const SizedBox(height: 20),
              Row(
                children: [
                  _cardMenu(
                    title: !_toggle_lights_room3
                        ? "Turn on Lights in Room 3"
                        : "Turn Off Lights in Room 3",
                    icon: Icons.light,
                    color: Colors.indigoAccent,
                    fontColor: !_toggle_lights_room3
                        ? Colors.black
                        : Colors.white,
                    onTap: isConnected()
                        ? () async {
                            await _sendMessage(
                                    _toggle_lights_room3
```

```
                                                    ? "6"
                                                    : "5")
                                                .then((value) {
                                              setState(() {
                                                _toggle_lights_room3 =
                                                    !_toggle_lights_room3;
                                              });
                                            });
                                          }
                                        : null,
                                    ),
                                  ],
                                )
                              ],
                            ),
                          )
                        ],
                      ),
                    ),
                  )
                : Center(
                    child: Column(
                      mainAxisAlignment: MainAxisAlignment.center,
                      children: <Widget>[
                        Padding(
                          padding: const EdgeInsets.all(16),
                          child: Text(_lastWords),
                        ),
                        GestureDetector(
                          onTap: _speechToText.isNotListening
                              ? _startListening
                              : _stopListening,
                          child: AvatarGlow(
                            animate: _speechToText.isListening,
                            glowColor: Colors.blue,
                            endRadius: 90.0,
                            duration: const Duration(milliseconds: 2000),
                            repeat: true,
                            showTwoGlows: true,
                            repeatPauseDuration:
                                const Duration(milliseconds: 100),
                            child: Material(
                              // Replace this child with your own
                              elevation: 8.0,
                              shape: const CircleBorder(),
                              child: CircleAvatar(
                                backgroundColor: Colors.grey[100],
                                radius: 40.0,
                                child: const Icon(
                                  Icons.mic,
```

```
                                  color: Colors.indigoAccent,
                                ),
                              ),
                            ),
                          ),
                        ),
                        Padding(
                          padding: const EdgeInsets.all(16),
                          child: Text(
                            _speechToText.isListening
                                ? ''
                                : _speechEnabled
                                    ? 'Tap the microphone to start listening...'
                                    : 'Speech not available',
                            style: const TextStyle(color: Colors.grey),
                          ),
                        ),
                        Padding(
                          padding: const EdgeInsets.all(16),
                          child: Text(
                            _indexOfResponses != null
                                ? _responses[_indexOfResponses!]
                                : '',
                            textAlign: TextAlign.center,
                            style: const TextStyle(
                                color: Colors.indigoAccent, fontSize: 30),
                          ),
                        ),
                      ],
                    ),
                  )
              : const Center(
                  child: Text(
                    "DISCONNECTED",
                    style: TextStyle(
                      fontSize: 18,
                      color: Colors.indigoAccent,
                      fontWeight: FontWeight.bold,
                      fontStyle: FontStyle.italic,
                    ),
                  ),
                ),
            ),
      ),
      bottomNavigationBar: isConnected()
          ? BottomNavigationBar(
              items: const <BottomNavigationBarItem>[
                BottomNavigationBarItem(
                  icon: Icon(Icons.home),
                  label: 'Buttons',
                ),
```

```
                  BottomNavigationBarItem(
                    icon: Icon(Icons.mic),
                    label: 'Assistant',
                  ),
                ],
                unselectedItemColor: Colors.grey,
                showUnselectedLabels: true,
                unselectedLabelStyle: const TextStyle(
                    color: Colors.grey, fontFamily: "Cairo", fontSize: 15),
                currentIndex: selectedIndex,
                selectedItemColor: Colors.indigoAccent,
                selectedLabelStyle: const TextStyle(
                    color: Colors.grey, fontFamily: "Cairo", fontSize: 15),
                onTap: _onItemTapped,
              )
            : null,
    );
  }

  void _onDataReceived(Uint8List data) {
    String message = String.fromCharCodes(data);

    if (int.tryParse(message) == 15 || message == "15") {
      Navigator.of(context).push(MaterialPageRoute(
          builder: (ctx) => const SensorRunningScreen(isFire: true)));
    } else if (int.tryParse(message) == 16 || message == "16") {
      Navigator.of(context).push(MaterialPageRoute(
          builder: (ctx) => const SensorRunningScreen(isFire: false)));
    }
  }

  Future<void> _sendMessage(String text) async {
    text = text.trim();
    //textEditingController.clear();
    if (text.isNotEmpty) {
      try {
        connection.output.add(utf8.encode("$text\r\n"));
        await connection.output.allSent;
      } catch (e) {
        setState(() {});

        rethrow;

        // Ignore error, but notify state
      }
```

```
646       }
647     }
648
649     bool isConnected() {
650       return connection != null && connection.isConnected;
651     }
652
653     Widget _cardMenu({
654       required String title,
655       required IconData icon,
656       required void Function()? onTap,
657       Color color = Colors.white,
658       Color fontColor = Colors.grey,
659     }) {
660       return GestureDetector(
661         onTap: onTap,
662         child: Container(
663           padding: const EdgeInsets.symmetric(
664             vertical: 36,
665           ),
666           width: 156,
667           decoration: BoxDecoration(
668             color: color,
669             borderRadius: BorderRadius.circular(24),
670           ),
671           child: Column(
672             children: [
673               Icon(
674                 icon,
675                 color: fontColor,
676                 size: 50,
677               ),
678               const SizedBox(height: 10),
679               Text(
680                 title,
681                 textAlign: TextAlign.center,
682                 style: TextStyle(fontWeight: FontWeight.bold, color: fontColor),
683               )
684             ],
685           ),
686         ),
687       );
688     }
689   }
```

6-assistant_instructions_screen.dart:

```dart
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';

class AssistantInstructionsScreen extends StatelessWidget {
  const AssistantInstructionsScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        centerTitle: true,
        title: const Text(
          "Assistant Instructions",
          style: TextStyle(
            fontSize: 18,
            color: Colors.indigo,
            fontWeight: FontWeight.bold,
          ),
        ),
        leading: IconButton(
          onPressed: () {
            Navigator.of(context).pop();
          },
          icon: const Icon(CupertinoIcons.left_chevron),
          color: Colors.indigo,
        ),
        elevation: 0,
        backgroundColor: Colors.indigo.shade50,
      ),
      backgroundColor: Colors.indigo.shade50,
      body: SafeArea(
        child: Padding(
          padding: const EdgeInsets.all(8.0),
          child: SingleChildScrollView(
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Center(
                  child: Image.asset(
                    "assets/images/banner.png",
                    scale: 1.2,
                  ),
                ),
```

```dart
                const SizedBox(
                  height: 32,
                ),
                const Center(
                  child: Text(
                    "Smart Home",
                    style: TextStyle(
                      fontSize: 32,
                      fontWeight: FontWeight.bold,
                    ),
                  ),
                ),
                const SizedBox(height: 48),
                _roomInstructions(1),
                _roomInstructions(2),
                _roomInstructions(3),
              ],
            ),
          ),
        ),
      ),
    );
  }

  _roomInstructions(int roomNumber) {
    late String room;

    if (roomNumber == 1) {
      room = "First";
    } else if (roomNumber == 2) {
      room = "Second";
    } else if (roomNumber == 3) {
      room = "Third";
    }
    return Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Text(
          "${room.toUpperCase()} ROOM",
          style: const TextStyle(
            fontSize: 14,
            fontWeight: FontWeight.bold,
          ),
```

```
        ),
        const SizedBox(height: 10),
        _containerTitle("Turn On The Light/s In Room Number $roomNumber"),
        const SizedBox(height: 10),
        _containerTitle(" Turn On The Light/s In $room Room"),
        const SizedBox(height: 10),
        const Divider(),
        const SizedBox(height: 10),
      ],
    );
  }

  _containerTitle(String title) {
    return Container(
      padding: const EdgeInsets.symmetric(
        vertical: 36,
      ),
      margin: const EdgeInsets.symmetric(
        horizontal: 10,
      ),
      decoration: BoxDecoration(
        color: Colors.indigoAccent,
        borderRadius: BorderRadius.circular(24),
      ),
      child: Padding(
        padding: const EdgeInsets.only(left: 17.0),
        child: Row(
          children: [
            const Icon(
              Icons.mic,
              color: Colors.white,
              size: 50,
            ),
            const SizedBox(height: 10),
            Text(
              title,
              textAlign: TextAlign.center,
              style: const TextStyle(
                fontWeight: FontWeight.bold,
                color: Colors.white,
              ),
            )
          ],
        ),
      ),
    );
  }
}
```

7-sensor_running_screen.dart:

```dart
import 'package:audioplayers/audioplayers.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';

class SensorRunningScreen extends StatefulWidget {
  final bool isFire;
  const SensorRunningScreen({super.key, required this.isFire});

  @override
  State<SensorRunningScreen> createState() => _SensorRunningScreenState();
}

class _SensorRunningScreenState extends State<SensorRunningScreen> {
  final audioPlayer = AudioPlayer();

  Future setAudio() async {
    audioPlayer
        .setReleaseMode(widget.isFire ? ReleaseMode.loop : ReleaseMode.stop);
    audioPlayer
        .play(AssetSource(widget.isFire ? 'fire_alarm.mp3' : 'rain_alarm.mp3'));
  }

  @override
  void initState() {
    super.initState();
    setAudio();
  }

  @override
  void dispose() {
    audioPlayer.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: widget.isFire ? Colors.red : Colors.blueAccent,
        elevation: 0,
        leading: IconButton(
          onPressed: () {
            Navigator.pop(context);
          },
          icon: const Icon(CupertinoIcons.left_chevron, color: Colors.white,),
        ),
      ),
      backgroundColor: widget.isFire ? Colors.red : Colors.blueAccent,
      body: SafeArea(
        child: Center(
          child: Text(
            widget.isFire ? "Fire Sensor is Running" : "Rain Sensor is Running",
            textAlign: TextAlign.center,
            style: const TextStyle(
                color: Colors.white, fontSize: 25, fontWeight: FontWeight.bold),
          ),
        ),
      ),
    );
  }
}
```

Implementation :

-The code of Arduino and Flutter App:

https://github.com/10YousefTarek/The-Final-Project

-The Arduino file called arduin.ino:

https://github.com/10YousefTarek/The-Final-Project/blob/main/arduino.ino

-To download the Flutter Application:

https://github.com/10YousefTarek/The-Final-Project/blob/main/app-release.apk

Chapter Six. Application Sector

Introduction to Flutter:

In the realm of app development, creating applications that work seamlessly across multiple platforms has always been a challenge. However, with the introduction of Flutter, developers now have a powerful framework at their disposal that enables them to build stunning, high-performance apps for various platforms using a single codebase. Flutter, developed by Google, has taken the development community by storm, revolutionizing cross-platform app development and redefining user experiences.

1. What is Flutter?

Flutter is an open-source UI toolkit that allows developers to create natively compiled applications for mobile, web, desktop, and even embedded systems from a single codebase. It provides a comprehensive set of tools, widgets, and libraries that streamline the app development process, empowering developers to create beautiful, responsive, and fast applications across different platforms.

2. The Dart Programming Language:

At the core of Flutter is the Dart programming language. Dart is a modern and object-oriented language specifically designed for building user interfaces. It combines the best features of languages like JavaScript, Java, and C++, offering developers a familiar and efficient programming experience. Dart's just-in-time (JIT) compilation allows for fast development cycles, while ahead-of-time (AOT) compilation ensures optimal performance and efficiency in the final deployed app.

3. Widgets: The Building Blocks of Flutter:

One of the standout features of Flutter is its rich and customizable set of widgets. Widgets are the building blocks of Flutter applications, representing everything from basic UI elements to complex layouts. Flutter offers an extensive collection of pre-designed and adaptive widgets that conform to the platform's design guidelines, ensuring a native-like look and feel across different devices.

4. Hot Reload: Rapid Iteration and Faster Development:

Flutter's hot reload feature has become a developer favorite. It allows developers to see the changes made in the code almost instantly

without losing the current app state. This rapid iteration capability significantly speeds up the development process, enabling developers to experiment, fix issues, and fine-tune the user interface in real-time. Hot reload eliminates the need for time-consuming recompilations and accelerates the debugging and testing phases.

5. Beautiful UI and Customization:

Flutter emphasizes the creation of visually stunning and engaging user interfaces. With its customizable widgets and rich animation support, developers can bring their app designs to life with smooth transitions, fluid motion, and delightful interactions. Flutter's powerful graphics engine enables the creation of beautiful UIs that can be tailored to match specific branding or design requirements, providing a unique and captivating user experience.

6. Performance and Native Integration:

Flutter's performance is another compelling aspect of the framework. By using the Flutter engine, which leverages the Skia graphics library, Flutter applications can achieve high rendering speeds, resulting in smooth animations and responsive user interfaces. Additionally, Flutter provides native-like performance by compiling the Dart code to native machine code, allowing applications to take full advantage of the underlying platform's capabilities.

Conclusion:

Flutter has emerged as a game-changer in the world of cross-platform app development. With its versatile toolkit, efficient programming language, extensive widget library, and emphasis on stunning user interfaces, Flutter empowers developers to build applications that run seamlessly on multiple platforms with impressive speed and visual appeal. The framework's flexibility, ease of use, and robust community support have contributed to its rapid adoption by developers worldwide. As Flutter continues to evolve, we can anticipate even greater advancements in cross-platform development, setting new standards for app performance, aesthetics, and user experience.

Exploring the Application Sector for SDK and Flutter SDK Components

Introduction:

In the world of software development, SDKs (Software Development Kits) play a crucial role in providing developers with the tools, libraries, and frameworks necessary to create robust and feature-rich applications. Among the prominent SDKs, Flutter SDK has gained considerable popularity due to its ability to enable cross-platform development. In this topic, we delve into the application sector for SDKs in general and explore the various components of the Flutter SDK that make it a versatile and powerful choice for developers.

1. Understanding SDKs:

SDKs are comprehensive packages that consist of various software tools and resources. They serve as a foundation for developers, offering pre-built components, libraries, and APIs that simplify the development process and enhance productivity. SDKs cater to specific platforms, such as mobile, web, desktop, or IoT, providing developers with the necessary tools to build applications tailored to those platforms.

2. The Application Sector for SDKs:

The application sector for SDKs is vast and encompasses a wide range of industries and domains. From mobile app development to game development, e-commerce platforms to IoT applications, SDKs are instrumental in accelerating the development process and ensuring the creation of high-quality applications. Developers across various sectors rely on SDKs to access platform-specific functionalities, integrate APIs, handle complex tasks, and deliver exceptional user experiences.

3. Introduction to Flutter SDK:

Flutter SDK, developed by Google, has gained significant attention in recent years for its cross-platform capabilities. It allows developers to write code once and deploy it seamlessly on multiple platforms, including iOS, Android, web, desktop, and even embedded systems.

Flutter SDK consists of several key components that contribute to its success in the application sector.

4. Dart Programming Language:

At the heart of Flutter SDK is the Dart programming language. Dart is a modern and efficient language that provides developers with the necessary tools for building high-performance applications. Dart's strong typing, just-in-time compilation, and garbage collection make it suitable for developing complex and responsive applications across different platforms.

5. Flutter Framework:

The Flutter framework is a key component of the Flutter SDK. It offers a rich set of customizable widgets that enable developers to create visually appealing and responsive user interfaces. Flutter's "hot reload" feature allows for real-time testing and rapid iteration, streamlining the development process and reducing time-to-market.

6. Widget Library:

Flutter SDK includes an extensive widget library, offering a wide range of pre-built UI components. These widgets can be easily customized to match the application's design requirements, ensuring a consistent user experience across platforms. The widget library also includes support for animations, gestures, and platform-specific functionalities, empowering developers to create engaging and interactive applications.

7. Flutter Engine:

The Flutter engine is responsible for rendering the user interface and handling interactions. It leverages Skia, a powerful graphics engine, to deliver fast and fluid animations and visuals. The Flutter engine ensures excellent performance on different platforms, providing a native-like experience to end-users.

8. Packages and Plugins:

Flutter SDK provides access to a vast ecosystem of packages and plugins contributed by the Flutter community. These packages extend the capabilities of Flutter, enabling developers to integrate various

functionalities such as networking, databases, authentication, and more. The availability of numerous packages significantly enhances productivity and allows developers to leverage existing solutions for common development challenges.

Conclusion:

SDKs are essential tools for developers, providing them with the necessary resources to build applications across diverse sectors. Flutter SDK, with its comprehensive set of components, has revolutionized cross-platform development. The combination of Dart, the Flutter framework, the widget library, the Flutter engine, and the extensive package ecosystem has made Flutter SDK a popular choice among developers looking for efficient and scalable solutions.

The application sector for SDKs, including Flutter SDK, continues to grow rapidly. From mobile apps to web applications, desktop software to IoT devices, the versatility of SDKs enables developers to create innovative solutions that meet the specific needs of different platforms and industries. As SDKs evolve and new technologies emerge, we can expect even more advancements and opportunities within the application sector, further enhancing the capabilities of developers and pushing the boundaries of software development.

The Expanding Application Sector for Flutter: Revolutionizing Cross-Platform Development

Introduction:

In recent years, Flutter has emerged as a leading cross-platform development framework, transforming the way developers create mobile, web, and desktop applications. Developed by Google, Flutter offers a seamless and efficient development experience, enabling developers to build high-quality, native-like applications for multiple platforms simultaneously. As a result, the application sector for Flutter has expanded rapidly, with developers and businesses leveraging its capabilities to create stunning and feature-rich apps. In this topic, we explore the diverse application sector for Flutter and the ways it is revolutionizing cross-platform development.

1. Mobile Applications:

Flutter has gained significant traction in the mobile application space, allowing developers to create visually appealing, responsive, and performant apps for both iOS and Android platforms. The framework's rich set of pre-built widgets, along with its hot-reload feature for rapid testing and development, empowers developers to iterate quickly and efficiently. From social media apps to e-commerce platforms, Flutter has proven its worth in a wide range of mobile applications, delivering smooth animations, delightful user experiences, and native performance.

2. Web Applications:

With the introduction of Flutter for web, developers can now create beautiful and interactive web applications using the same codebase used for mobile apps. Flutter's reactive framework and customizable widgets enable developers to design web interfaces that are responsive, visually consistent, and compatible across different browsers. Whether it's a single-page application or a complex web portal, Flutter's versatility allows for seamless deployment and maintenance of web applications.

3. Desktop Applications:

Flutter's expansion into desktop application development has opened up new possibilities for creating cross-platform software. By utilizing Flutter's framework, developers can build native-like desktop applications for Windows, macOS, and Linux operating systems. Whether it's productivity tools, media players, or creative software, Flutter's performance and native integration capabilities make it a viable choice for desktop app development, enabling developers to reach a broader audience with a single codebase.

4. IoT Applications:

The Internet of Things (IoT) sector has seen a surge in demand for applications that integrate with connected devices. Flutter's flexibility and adaptability make it an excellent choice for building IoT applications. By leveraging Flutter's rich ecosystem of libraries and plugins, developers can create user-friendly interfaces that interact seamlessly with IoT devices, enabling users to control and monitor their smart devices from a single application. Flutter's ability to connect to various hardware devices via Bluetooth, Wi-Fi, or other protocols makes it a versatile framework for IoT app development.

5. Cross-Platform Game Development:

Flutter's fast rendering engine and support for game development frameworks, such as Flame, have made it an appealing choice for cross-platform game development. Developers can create engaging 2D games with smooth animations and fluid user interactions. Flutter's performance optimization capabilities ensure games run seamlessly on different platforms, offering an immersive gaming experience to users regardless of their preferred device.

Conclusion:

Flutter has emerged as a powerhouse in the cross-platform development landscape, empowering developers to build high-quality applications across mobile, web, desktop, IoT, and gaming platforms. Its flexibility, hot-reload feature, rich widget library, and extensive ecosystem have made it a go-to choice for developers looking to streamline the app development process and reduce time-to-market. As the application sector for Flutter continues to expand, we can expect to see even more innovative and diverse applications taking advantage of this robust framework, driving the future of cross-platform development forward.

6. Emerging Trends and Future Outlook:

The application sector for Flutter is evolving rapidly, with new trends and opportunities emerging. One such trend is the integration of Flutter

with augmented reality (AR) and virtual reality (VR) technologies. By combining Flutter's cross-platform capabilities

With AR/VR, developers can create immersive experiences that bridge the gap between physical and digital worlds. This opens up avenues for applications in various industries, such as gaming, education, architecture, and training.

Another promising area is Flutter's integration with machine learning (ML) and artificial intelligence (AI) frameworks. By leveraging Flutter's flexibility and ML/AI capabilities, developers can create intelligent applications that offer personalized experiences, predictive analytics, and automation. This paves the way for innovative applications in areas like healthcare, finance, and smart home automation.

Conclusion (continued):

The future of Flutter's application sector looks promising, with continuous advancements and enhancements in the framework. The Flutter community is active and vibrant, with developers contributing new packages, plugins, and libraries regularly. This collaborative environment ensures that the framework remains up-to-date with the latest technologies and trends, further expanding its potential application sectors.

In conclusion, Flutter has revolutionized cross-platform development, enabling developers to build high-quality applications for mobile, web, desktop, IoT, and gaming platforms. Its versatility, performance, and extensive ecosystem make it a preferred choice for developers and businesses alike. As the application sector for Flutter continues to grow, we can expect to see innovative and impactful applications that push the boundaries of what is possible with cross-platform development.