

HOME AUTOMATION USING HMI DISPLAY

*Project report submitted in partial fulfilment of the required Bachelor for the degree
of*

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

By

Amber Atri (191025)

Ambar S Tiwari (191039)

UNDER THE GUIDANCE OF

Asst. Prof. Munish Sood



JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

May 2022

TABLE OF CONTENTS

CAPTION	PAGE NO.
DECLARATION	i
ACKNOWLEDGEMENT	ii
LIST OF ACRONYMS AND ABBREVIATIONS	iii
LIST OF SYMBOLS	iv
LIST OF FIGURES	v
LIST OF TABLES	v
ABSTRACT	vi
CHAPTER-1: INTRODUCTION	1
1.1 Problem Statement	1
1.2 Existing Work	2
1.3 Objectives	6
1.4 Motivation	8
CHAPTER-2: OVERVIEW OF THE MODEL	9
2.1 Hardware Requirements	9
2.2 Software Requirements	9
2.2.1 Nextion Editor	9
2.2.2 Arduino IDE	10
2.3 Understanding the Components	11
2.3.1 HMI Display (Nextion Display)	11
2.3.2 ESP32	13
2.3.3 Arduino UNO	14
2.3.4 Relay Module	18
CHAPTER-3: WORKING	21
3.1 Block Diagram	21
3.2 Working	21
3.3 Arduino Code	24
3.4 ESP32 Code	32
CHAPTER-4: CONCLUSION	39
REFERENCES	
APPENDIX	
PLAGIARISM REPORT	

DECLARATION

We hereby declare that the work reported in the B.Tech Project Report entitled “Auto Temperature Detector for Entrance and Home Automation” submitted at the **Jaypee University of Information Technology, Wagnaghat, India** is an authentic record of our work carried out under the supervision of Mr Munish Sood. We have not submitted this work elsewhere for any other degree or diploma.

Amber Atri

191025

Ambar S Tiwari

191039

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Asst Prof. Munish Sood

Date:

Head of the Department/Project Coordinator

ACKNOWLEDGEMENT

We would like to express our sincere appreciation to our supervisor, Mr Munish Sood, Assistant Professor, for being pillars of support and encouragement throughout our project work. His experience, strengths, tenderness, and willfulness, have taught us valuable lessons for life, which are going to be of immense help in taking decisions for moving forward. The chain of gratitude would be complete without thanking the Almighty, the prime mover, for inspiring and guiding us to complete this task successfully.

May, 2023

Amber Atri (191025)

Ambar S Tiwari (191039)

LIST OF ACRONYMS AND ABBREVIATIONS

1. WAN- Wide Area Network
2. HMI- Human Machine Interface
3. V- volt

LIST OF SYMBOLS

1. '-': Negative Terminal/GND
2. '+': Positive Terminal/VCC

LIST OF FIGURES

1. Figure 2.1 Nextion Editor
2. Figure 2.2 Nextion HMI Display
3. Figure 2.3 ESP32 Board
4. Figure 2.4 Arduino UNO
5. Figure 2.5 Relay Module
6. Figure 3.1 Nextion Editor(With GUI)
7. Figure 3.2 Nextion Display(With GUI)
8. Figure 3.3 Working Model

LIST OF TABLES

Table 2.1: Components

ABSTRACT

With the use of home automation technology, a home's different appliances and systems can be managed remotely. This can apply to lighting, heating, cooling, security, and other factors. A microcontroller like the Arduino UNO, ESP32 and an HMI (Human-Machine Interface) display are required to build this home automation system.

Users can interact with the system using sliders, buttons, and other graphical elements on an HMI display. Usually, a touch screen or a display panel are used to present this interface. Users can effortlessly operate numerous home automation systems from a single interface by pairing an HMI display with an Arduino UNO and ESP32.

We need relays, sensors, or other components to connect the devices we need to control to the microcontroller to build a home automation system. Digital or analogue inputs and outputs on the microcontroller can be utilized for linking these devices.

CHAPTER 1

INTRODUCTION

We will use an HMI display and a few sensors to fully automate our home to make our daily-use appliances fully automatic.

1.1. Problem Statement

Conventional home automation systems can be expensive and difficult to install, and they may require significant programming skills to set up and operate. This could be the problem statement for home automation using an HMI display. Additionally, many modern home automation systems depend on unique hardware and software, which might reduce the system's flexibility and connectivity with other gadgets and systems.

A cheap and adaptable home automation system might be made using an HMI display and an ESP32 or Arduino Uno microcontroller as a solution to these problems. The system might give customers access to a user-friendly graphical interface on the HMI display that they could use to manage various household appliances, lighting, and fans.

These issues may be solved by using an HMI display as a centralized control system since it provides a user-friendly interface that allows users to manage many devices and systems from a single place. An HMI display gives the system a visual representation, making it simpler to use and traverse. The display could offer real-time information, such as a room's temperature or humidity levels, giving customers insightful knowledge of how well their house is doing.

But developing a home automation system with an HMI display necessitates fusing numerous hardware and software elements and guaranteeing that they work together. To monitor environmental factors like temperature and humidity, for instance, the system could need sensors. It might also need actuators to operate appliances like lights, fans, and air conditioners. The system must also work with other smart home appliances, like cameras, smart locks, and home assistants.

The security of the system must also be ensured since unauthorized access to the control system might have negative consequences. The safety and security of the house and its residents may be endangered if a hacked system allows an attacker to manipulate numerous pieces of equipment, such as turning off the lights or opening doors. To guarantee that the system satisfies the demands of the users while safeguarding their privacy and security, the construction of a reliable and secure home automation system using an HMI display involves careful planning, design, and execution.

In summary, the requirement for a user-friendly, interoperable, and secure centralized control system is at the centre of the issue statement for home automation utilizing an HMI display. To create such a system, it is necessary to integrate multiple hardware and software components, ensure their interoperability, and protect the system from unauthorized access. A home automation system with an HMI display can give consumers a smooth and practical method to operate and monitor the different systems and equipment in their homes by addressing these problems.

1.2 Existing work

1. "Smart Home Automation System with Touchscreen HMI" was published in the Journal of Electrical and Electronics Engineering Research in 2017.

In the research, a smart home automation system has been shown that uses a touchscreen HMI as a central control system for handling and maintaining tabs on a variety of home systems and appliances.

To give consumers a centralized control system for controlling and monitoring numerous home systems and appliances, the study's authors set out to design an affordable and user-friendly smart home automation system. The system was created to be simple to set up and use, needing no specific user expertise or understanding.

A touchscreen HMI, an Arduino Uno microcontroller, and several sensors and actuators were used to build the system. A 3.5-inch TFT touchscreen display with a 320x480 pixel resolution was used as the system's HMI. A serial interface was used to link the display to the Arduino Uno microcontroller.

Lighting, fans, air conditioners, and security systems are just a few of the systems and appliances that the system was made to monitor and function. The device may also keep a watch on environmental factors like humidity and temperature. To connect with these systems and devices, the study's authors used several sensors and actuators.

The Arduino Integrated Development Environment (IDE) and the Nextion Editor software were used to program the system. The HMI display was created and developed using the Nextion Editor programme. The Arduino IDE was used to communicate with several sensors and actuators and programme the Arduino Uno microcontroller.

The HMI display was created to be intuitive to use and simple to operate. The display was constructed from several displays that showed data about the many systems and appliances that were being managed and monitored. To illustrate the appliances and systems being tracked and managed, the study's authors employed a variety of visuals and symbols.

In addition, the system was built to be dependable and responsive. To guarantee that the system could recover from any faults or failures that could have happened, the study's authors built a variety of error handling and recovery procedures. To make sure the system was safe and secure against unauthorized access, the study's authors also added several security mechanisms.

The system underwent real-world testing and evaluation. The system was put in a house, and the study's authors kept an eye on it there for a while. The study's authors discovered that the system was dependable and simple to use. The study's authors also discovered that the technology gave consumers a centralized control system so they could manage and observe various home appliances from a single location.

The study's authors concluded that a smart home automation system could be controlled centrally by an HMI display in a way that was both affordable and simple to use. To enhance the system's usefulness and performance, the study's authors advised doing more research. The study's authors also proposed that the system may be expanded to incorporate other devices and systems, such as smart home entertainment systems and kitchen equipment.

2. “Design and Implementation of Smart Home System Based on HMI Display and Internet of Things” was published in the Journal of Information Science and Control Engineering in 2020.

The research introduces a smart home system that makes use of an HMI display and the Internet of Things (IoT) to give consumers a centralized control system for controlling and watching different home systems and appliances.

The study's authors wanted to create a smart home system that would provide consumers with an easy-to-use interface for managing and watching various systems and appliances in the home. Users may quickly add additional devices and appliances to the system as it was made to be flexible and adaptable.

An ESP32 microcontroller, an HMI display, and several sensors and actuators were used to build the system. A 7-inch TFT touchscreen display with a resolution of 1024x600 pixels was utilized as the system's HMI display. A serial interface was used to link the display to the ESP32 microcontroller.

Lighting, heating, air conditioning, and security systems were all included in the system's architecture for monitoring and control. The system may also keep an eye on environmental factors like humidity, temperature, and air quality. To connect with these devices and systems, the study's authors employed a variety of sensors and actuators.

The Arduino Integrated Development Environment (IDE) and the Nextion Editor software were used to program the system. The HMI display was created and developed using the Nextion Editor programme. The ESP32 microcontroller was programmed using the Arduino IDE, which was also used to connect with several sensors and actuators.

The HMI display was created to be intuitive to use and simple to operate. The display was made up of several shows that showed data about the many systems and appliances that were being managed and monitored. To illustrate the appliances and systems being tracked and managed, the study's authors used a wide range of symbols and images.

In addition, the system was built to be dependable and responsive. To guarantee that the system could recover from any faults or failures that could have happened, the study's authors built a variety of error handling and recovery processes. To make sure the system was safe and secure against unauthorized access, the study's authors also added several security mechanisms.

The system underwent real-world testing and evaluation. The system was put in a house, and the study's authors kept an eye on it there for a while. The study's authors found that the system was dependable and simple to use. The study's authors also discovered that the technology gave consumers a centralized control system that they could manage and observe various home appliances and systems from a single location.

The study's authors concluded that an HMI display and IoT technology were a practical and affordable choice for a central control system for a smart home automation system. The study's authors advised conducting more research to improve the value and performance of the system. The study's authors also proposed that the system be broadened to include more devices and systems, such as home entertainment systems and intelligent appliances for the kitchen.

The research paper "Design and Implementation of Smart Home System Based on HMI Display and Internet of Things" concludes by showcasing a smart home system that makes use of an HMI display and IoT technology to give users a centralized control system for managing and monitoring various home systems and appliances. The system was created to be simple to use, flexible, and extendable. An ESP32 microcontroller, an HMI display, and several sensors and actuators were used to build the system. The HMI display was created to be simple to use and simple to operate. The system was examined in a practical situation and found to be dependable and user-friendly. Additional research might be carried out, based on the study's authors, to enhance the system's performance

1.3 Objectives

Centralized control system requirement: The HMI should offer a centralized control system for controlling and keeping control of various home systems and appliances.

1. Enhance user experience: The HMI should be created with ease of use and simplicity in mind, offering an improved user interface for managing and keeping tabs on household appliances.
2. Boost user convenience and efficiency: The HMI should make controlling and monitoring household appliances easier and more convenient for users, saving them time and effort.
3. Enable remote control and monitoring: To give consumers more flexibility and convenience, the HMI should allow users to remotely control and monitor their home systems and appliances.
4. Improve the efficiency of energy use: The HMI should be created in a way that improves energy efficiency, allowing users to better control their energy use and cut expenses.
5. Real-time feedback: The HMI has to give information on all of the appliances and systems that are being managed and monitored in real time.
6. Boost home security: The HMI should be able to keep track of and manage home security systems, providing consumers more control over the security of their residences. To increase scalability and flexibility, the HMI should be developed in a way that renders it simple for users to include additional appliances and devices into the structure as required.
7. Improve the efficiency of energy use: The HMI should be created in a way that improves energy efficiency, allowing users to better control their energy use and cut expenses.

8. Real-time feedback: The HMI has to give information on all of the appliances and systems that are being managed and monitored in real time.
9. Boost home security: The HMI should be able to keep track of and manage home security systems, providing consumers more control over the security of their residences.
10. To increase scalability and flexibility, the HMI should be developed in a way that renders it simple for users to include additional appliances and devices into the structure as required.
11. Enhance fault tolerance and reliability: The HMI has to be dependable and fault-tolerant so that it can recover from mistakes and failures.
12. Enable data analysis: The HMI should be able to collect and analyse information about systems and appliances used in homes, providing users knowledge they can use to better manage their living spaces and energy use.

Overall, the goal of using home automation to build an HMI is to give consumers a centralized control system that enhances their homes' ease of use, effectiveness, and security while simultaneously lowering energy expenditures and raising energy efficiency. The HMI must allow remote control and monitoring of home systems and appliances and should be scalable, reliable, and user-friendly.

1.4 Motivation

We can easily operate your AC-related home appliances with this project, including lights, fans, air conditioners, televisions, etc. We can also keep an eye on the sensor data in real time. With the aid of a relay module, we are employing household appliances for demonstration purposes.

Here, home appliances are controlled by a widget with four control buttons. We have linked household appliances to the relay for this demonstration. With this project, we can operate and connect 220V-250VAC appliances with ease.

We don't require an active internet connection to control and monitor your home appliances because this project is locally area based. The ESP32 web server is being configured right now. We simply enter the password for the "ESP32 Smart Home" Wi-Fi access point to connect to it. Then, enter the ESP board's IP address in the browser to view real-time sensor data and remotely operate the appliances from a smartphone. Therefore, to monitor and operate your home appliances, we do not require an active wifi connection.

CHAPTER 2

OVERVIEW OF THE MODEL

2.1 Hardware Requirements

S. No	Quantity	Component Name
1	1	ESP32
2	1	HMI Display (Nextion Display)
3	1	Breadboard
4	-	Wires
5	1	Relay Module
6	1	Arduino UNO

2.2 Software Requirements

All we need is a computer system that can run the Arduino IDE software to implement the device's software. Usually, a Windows operating system with an i3 processor and 2GB of RAM suffices.

2.2.1 Nextion Editor

A software programme called Nextion Editor is used for developing graphical user interfaces (GUI) for Nextion HMI (Human Machine Interface) displays. Users using the editor can design unique user interfaces that can be used to interact with and operate a wide range of electrical components and devices.

The drag-and-drop interface of the Nextion Editor makes it simple to integrate buttons, sliders, images, text, and other graphical elements in a project. The editor also comes with several pre-designed components that are simple to adapt to the project's requirements. Users can also contribute unique scripts to the project that change how the user interface works.

The Nextion Editor's WYSIWYG (What You See Is What You Get) design interface is one of its primary features. It means that users can create a project while seeing the way it will appear on the Nextion HMI display. Users can test their projects in the editor's simulator mode without uploading them to the display.

UART, I2C, and SPI are just a few of the communication protocols that the Nextion Editor supports. It makes connecting the Nextion HMI display to a variety of electronic systems and devices simple. Debugging tools are also included in the editor to help with user interface problems or communication problems between the display and the computer.

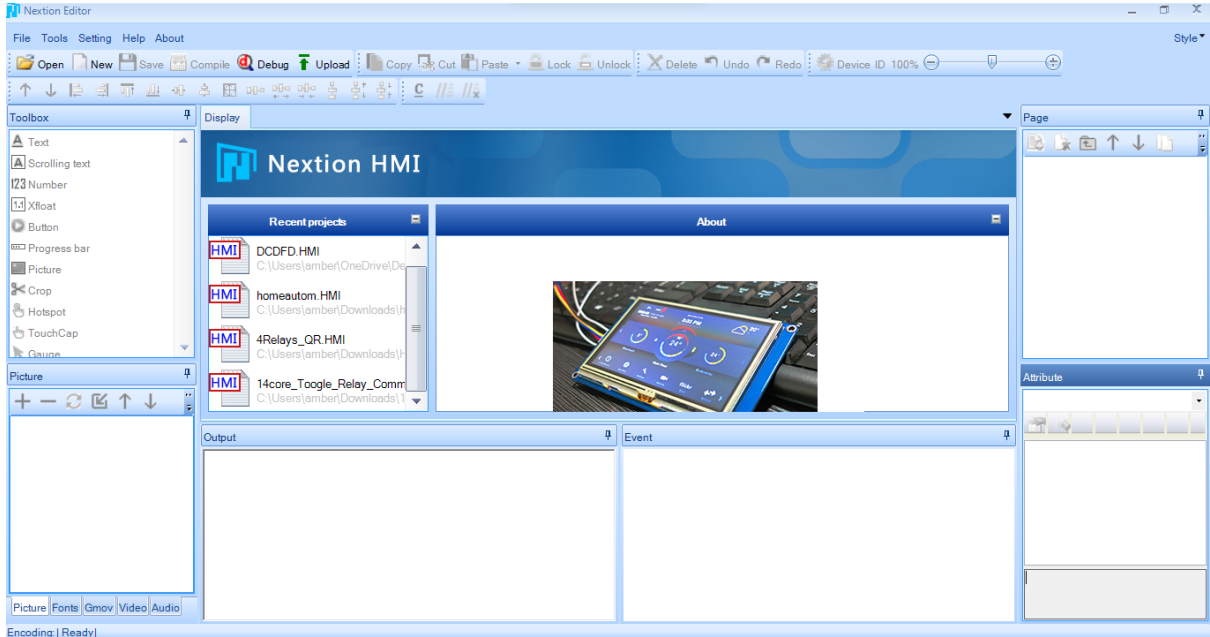


Fig 2.1 Nextion Editor

2.2.2 Arduino IDE

An open-source software programme called the Arduino Integrated Development Environment (IDE) is used to create and upload code to Arduino microcontroller boards. Even for those with little or no programming knowledge, the IDE offers a user-friendly interface that makes it simple to develop and upload code to the Arduino board.

Programming languages that are supported by the Arduino IDE include C, C++, and "Arduino Sketch," a simplified version of C++. Beginners should use the Sketch language because it is simple to learn and has a simple syntax. Users can create code quicker with the text editor included in the IDE because of features including syntax highlighting, auto-completion, and error highlighting.

A serial monitor is an additional function of the IDE which allows users to interact with the Arduino board and see the results of their code in real time. This helps with code debugging and troubleshooting as well as maintaining track of the way sensors and other linked devices behave.

The functionality of the Arduino IDE can be extended by users by downloading other libraries and tools. A library manager built within the IDE makes it simple to locate, download, and install libraries for various sensors and components. In addition, users can develop their libraries and share them throughout the Arduino community.

Windows, Linux, and macOS are only a few of the many operating systems that the Arduino IDE is compatible with. The popular Arduino Uno, Nano, and Mega boards, as well as additional boards built on the Arduino platform, are all compatible with it. It is also compatible with a broad variety of Arduino boards.

In conclusion, the Arduino IDE is a potent and simple to use programme used to create and upload code to Arduino microcontroller devices. It supports a large number of programming languages, has tools and libraries that allow for extensive customization, and has features like syntax and error highlighting. The IDE is a great option for professionals, students, and supporters alike because it is compatible with a wide range of operating systems and Arduino boards.

2.3 Understanding The Components

2.3.1 HMI Display (Nextion Display)

A graphical user interface called a human-machine interface (HMI) display enables humans to interact with machines and systems. HMIs tend to appear in consumer electronics like smartphones and tablets, medical equipment, and industrial control systems. A touch screen or display panel, along with different buttons, sliders, and other graphic components, constitute an HMI display in the majority of applications.

The capacity to connect with a system using a simple and user-friendly interface is one of the main advantages of using an HMI display. An HMI display makes it simple for users to monitor system status, alter settings, and control numerous functions with a few taps or clicks. This can be particularly useful in complex systems with many variables to control or in emergencies.

HMI's are frequently used in industrial control systems to regulate production processes as well as maintain track of equipment performance. For example, a manufacturing facility's temperature and humidity levels might be managed, and a conveyor belt system's status could be checked using an HMI display. HMI's in medical equipment can be used to monitor patient vitals in real-time while also controlling the functioning of equipment like ventilators and heart monitors. It can also be used in Home automation with the help of relay modules and microcontrollers and Home security too.

The resolution of the display is another essential aspect to take into consideration. According to the amount of information required to be provided, the display's resolution should be chosen. For systems with complex graphics or text, a higher-resolution display might be required, while a lower-resolution display might be suitable for simple systems.

When choosing an HMI display, considerations aside from size and resolution to consider include the type of touchscreen technology used, the display's brightness and contrast, and the display's endurance. Particularly in portable devices like smartphones and tablets, the power consumption of the display must be taken into account.

Overall, connecting with a machine or a system can be simplified by an HMI display. An HMI display may enhance productivity, increase efficiency, and improve user experience by offering a simple and user-friendly interface. An HMI display may help to simplify complicated procedures and improve how humans engage with technology, whether it is used in industrial control systems, medical equipment, or consumer gadgets.



Figure 2.2:Nextion HMI Display

2.3.2 ESP32

Powerful microcontrollers like the ESP32 have frequently been employed in a variety of Internet of Things (IoT) applications. It is based on the ESP32-WROOM-32 module, which has a dual-core 32-bit processor, Wi-Fi, Bluetooth, and a variety of auxiliary hardware. Here are the ESP32 microcontroller's features and specifications of the ESP32 microcontroller.

ESP32 Specifications:

- Wireless connectivity WiFi: 150.0 Mbps data rate
 - Bluetooth: BLE (Bluetooth Low Energy) and Bluetooth Classic
 - Processor: Tensilica Xtensa Dual-Core 32-bit LX6 microprocessor, running at 160 or 240 MHz
- Memory:
 - ROM: 448 KB (for booting and core functions)
 - SRAM: 520 KB (for data and instructions)
- Low Power: ensures you can still use ADC conversions, for example, during deep sleep.
- Peripheral Input/Output:
 - peripheral interface with DMA that includes a capacitive touch
 - ADCs (Analog-to-Digital Converter)
 - DACs (Digital-to-Analog Converter)
 - I to C (Inter-Integrated Circuit)
 - UART (Universal Asynchronous Receiver/Transmitter)
 - SPI (Serial Peripheral Interface)
 - RMI (Reduced Media-Independent Interface)
 - PWM (Pulse-Width Modulation)

ESP32 VS ESP8266 Board:

- The ESP32 is faster than the ESP8266;
- The ESP32 comes with more GPIOs with multiple functions;

- The ESP32 supports analogue measurements on 18 channels (analogue-enabled pins) versus just gone 10-bit ADC pins on the ESP8266;
- The ESP32 supports Bluetooth while the ESP8266 doesn't;
- The ESP32 is dual-core (most models), and the ESP8266 is single-core;
- The ESP32 is a bit more expensive than the ESP8266.

A variety of software development tools, such as the Arduino IDE and the ESP-IDF (ESP32 IoT Development Framework), support the ESP32. Several tools and libraries are included in the strong development environment known as the ESP-IDF that may be used to develop programmes for the ESP32. A wide range of third-party development tools, like MicroPython and TensorFlow Lite, are also compatible with the ESP32, enabling developers to choose those that best match their needs.

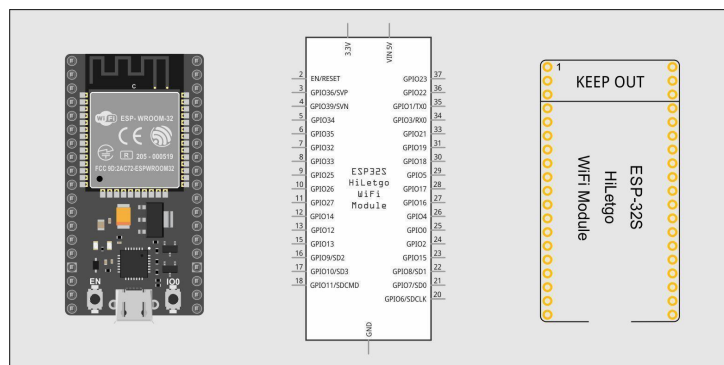


Figure 2.3: ESP32 Board

2.3.4 Arduino UNO

The open-source Arduino Platform is used in the creation of electric projects. With Arduino, we can develop programming code and upload it to an Arduino Circuit board (commonly referred to as a microcontroller) using software known as the IDE (Integrated Development Environment).

With good cause, the Arduino Platform has grown in popularity among those learning the basics of electronics. The Arduino allows uploading fresh code to the board using a USB cable as opposed to a specialized hardware item (referred to as a programmer) like the majority of earlier programmable circuit boards. A compressed version of C++ is also used by the Arduino IDE to make programming easier to understand.

Finally, Arduino provides a common form factor that divides the microcontroller features into a more practical packaging.

Arduino Board Overview

Powers(USB / Barrel Jack)

There must be a means for each Arduino board to connect to a power source. A USB device from your computer or a wall power source (like this one) that terminates in a barrel jack can be used to power an Arduino UNO. The USB connection is marked (1) in the figure below, and the barrel connection is shown (1). (2).

In addition, we can download code to the Arduino board using the USB connection. The Setup and Programming Arduino guide has more details on programming with Arduino.

NOTE: Only use power supplies with a peak power of 20 volts to prevent overloading your Arduino. It's recommended to use a voltage of between 6 and 12 volts for the bulk of Arduino devices.

Pins(5V, 3.3V, GND, Analog, Digital, PWM, AREF)

Build circuits using the ping on your Arduino, most likely in combination with a breadboard as well as some wire. They frequently have "headers" made of black plastic that allow for direct wire entry into the board.

The Arduino includes several different pin categories, one of which is noted just on board and has a specific function. Short for "Ground," GND (3) Any of the Arduino's GND pins can be utilized to ground the circuit. There are numerous GND pins.

5V (4) & 3.3V (5): The 5V pin delivers 5 volts of electricity, while the 3.3V pin delivers 3.3 volts, as one might expect. The vast majority of the simple Arduino parts happily run on 5 or 3.3 volts.

Analog (6): The UNO's "Analog In" pins(A0 through A5) are the pins that bear this designation. An analogue sensor signal, perhaps one from a temperature probe, can be read by these pins and converted into a legible digital value.

Analogue (7): The analogue pins are in front of the digital pins (0 through 13 on the UNO). Both digital output and interaction with the virtual, such as assessing if a key has been depressed, are possible with these pins.

PWM (8) There are many tildes (~) adjacent to some of the digital pins in PWM (8), as you may have observed (3, 5, 6, 9, 10, and 11 on the UNO). In Addition to serving as ordinary input pins, those pins can be used for a technique called pulse-width modulation (PWM). These ports can simulate analogue signals.

Power LED Indicator

On the circuit board, there is a little LED next to the word "ON" and beneath and to the right of the word "UNO" (11). Every time we plug the Arduino into a power source, this LED ought to turn on.

TX RX LEDs

Transmit is shortened to TX, and Receive is shortened to RX. In electronics, these markings are frequently used to identify the pins used for serial transmission. TX and RX in our case appear on the Arduino UNO twice: first by digital pins 0 and 1, and again adjacent to the TX and RX indicator LEDs (12). When Arduino is receiving or transferring data (such as when we're putting a new programme onto the board), these LEDs will provide us with some great visual cues.

Main IC

An IC, or integrated circuit, is a black object with all the metal legs (13). It can be considered as the Arduino's brain. The primary integrated circuit (IC) in the Arduino varies slightly depending on the type of board used, however, it is often from the ATMEL company's ATmega range of ICs. This can be crucial since, before loading a new programme from the Arduino software, we might need to know the IC type (along with your board type). The top side of the IC often has this information written on it. Reading the datasheets is frequently a good idea if we want to learn more about the distinctions between various ICs.

Voltage Regulator,

We should not tamper with both the voltage regulator on the Arduino (14). But it might be useful to be aware of its presence and function. In terms of regulating the voltage provided to the Arduino board, the volt regulator precisely accomplishes what it is designed to do. Think of it as a gatekeeper; it will stop extra electricity from entering the system and potentially damaging the circuit. Since Arduino plainly has limitations, avoid connecting it to anything that requires a voltage of more than 20 volts.

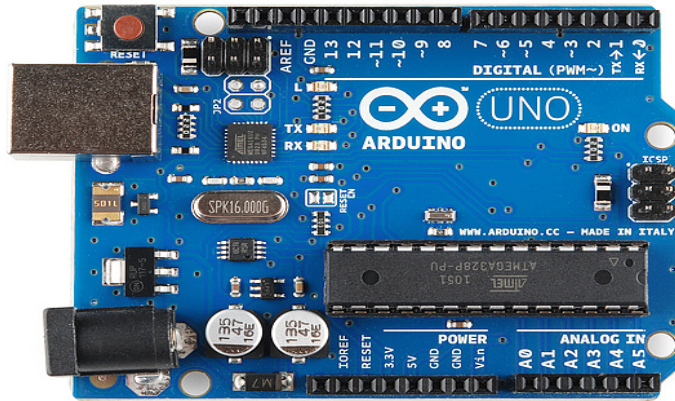


Figure 2.4 Arduino UNO

1. good for assessments of 20–80% relative humidity with 5% accuracy
2. 2°C accuracy for temperature readings

2.3.6 Relay module

Relay modules, often known as power relay modules, are common electronic parts. Any project involving home automation must include them. If we wish to utilize a low-voltage microcontroller, like an Arduino, to control motors or lighting circuits, a relay module is necessary.

Relay modules are straightforward construction elements. Mostly, they serve as switches. Two internal metal contacts make up the standard relay module. Usually, these encounters don't run into each other or interact. However, relays come with an inbuilt switch that connects these contacts to complete an electrical current that permits current passage.

Relay modules don't function the same way as conventional light switches. For example, to turn on a lamp, we need to press a button that joins the two metal layers inside the lamp. In comparison, a relay switch uses electrical pulses to turn on and off its internal switch.

An electromagnetic coil is driven by a voltage or current applied to just one side of the circuit, which squeezes the metal layers together. Because of this, current can flow through the relay's other side.

An Arduino Board or Raspberry Pi can be used to send a digital signal to the relay, as well as the relay can then provide power to any necessary applications. As we expect, there are different kinds of relay modules.

How Do Relay Modules Work?

First, we must establish the difference between a relay as well as a relay module. A relay module is made up of a group of one or even more relays. Although Purchasing individual relays in addition to the module is a possibility, we suggest purchasing them in the form of modules. This is true because it offers a few advantages.

On the input side of a conventional single-channel relay module, there are three jumper pin connectors that we can use to gain access to the relay's input. We find that the output connectors are wires that can be attached to hardware. Any single digital signal demand we may have is now easier to connect. Additionally, an LED is located at the bottom of both the majority of modules. The relay operates in two modes: on when activated and off when deactivated.

The relay's electromagnetic coil is crossed by a diode that is present. Actually, it is what is known as a bidirectional converter diode. As soon as we activate the coil and it approaches deactivation, the relay needs to release that count somewhere. The flyback diode prevents the input voltage from going back to the output pin.

Relay Module Applications

We can use relay module in the following applications:

1. Home Automation: One of the most common applications of Home Automation: One of the most common applications of a relay module is in home automation systems. The relay module can be used to control lights, fans, air conditioners, and other appliances in a home. For example, a relay module can be used to turn off the lights in a room when the room is not in use.

2. **Industrial Automation:** Relay modules are also commonly used in industrial automation systems. They can be used to control machines and equipment in a factory or production line. For example, a relay module can be used to control the motors in a conveyor belt system.
3. **Automotive Applications:** Relay modules are also used in automotive applications. They can be used to control various systems in a vehicle, such as the headlights, windshield wipers, and air conditioning. For example, a relay module can be used to turn on the headlights when the vehicle is started.
4. **Security Systems:** Relay modules are also used in security systems, such as access control systems and alarm systems. They can be used to control door locks and sensors that detect movement or intrusion. For example, a relay module can be used to activate an alarm when a motion the sensor detects movement.
5. **Energy Management Systems:** Relay modules can be used in energy management systems to control the flow of electricity in a building or facility. For example, a relay module can be used to turn off lights and appliances in a building when they are not in use, to save energy.

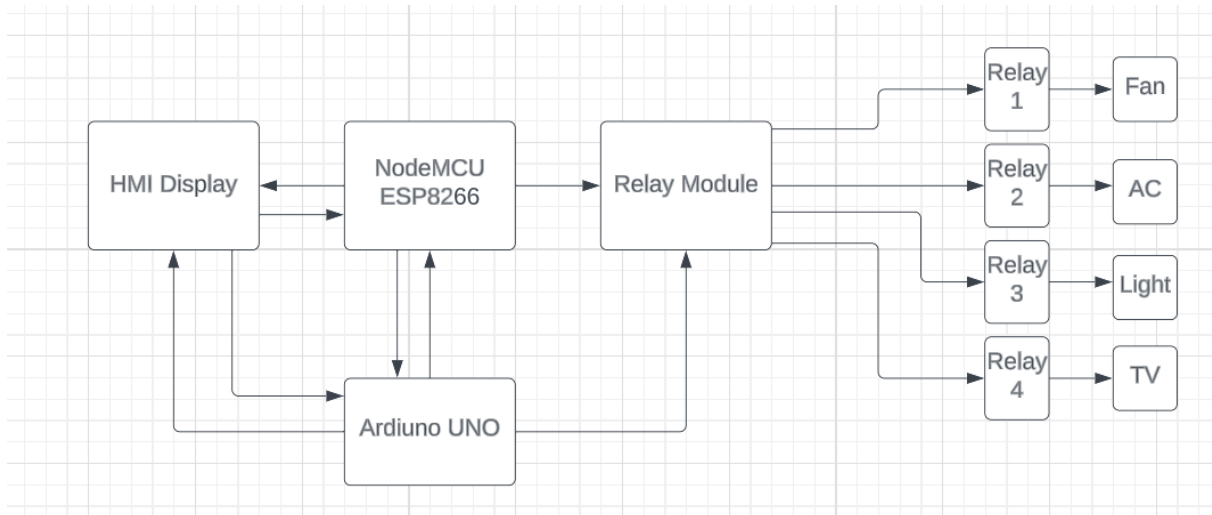


Figure 2.5 Relay Module

CHAPTER 3

WORKING

3.1 BLOCK-DIAGRAM



3.2 Working

To automate it, we're utilizing a Nextion HMI display with an Arduino UNO and Node MCU ESP 32 coupled to a Relay Module with 4 relays connecting to home appliances. Node mcu esp 32 will be connected to Nextion HMI display as shown in the block diagram above. which will provide touch-panel control of the home automation.

We are utilizing a power relay module, which is an electrical switch driven by an electromagnet. The electromagnet is turned on by a separate, low-power signal from a microcontroller. When energized, the electromagnet pulls to either open or close an electrical circuit. It is an essential part of securing home appliances.

A 2.5 Nextion HMI Touch Display is present. There are four control button widgets for our home appliances below the sensor value widgets. Toggle switches can be clicked to turn relays "ON" and "OFF." Four different AC appliances can be connected to the relay module here.

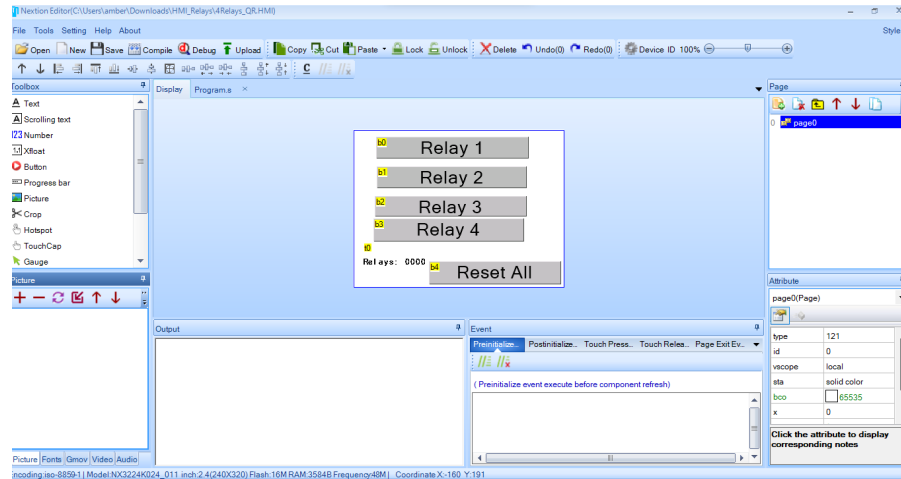


Fig 3.1 Nextion Editor(With GUI)

The Nextion display is a popular touch screen display that can be easily interfaced with an Arduino microcontroller

1. First, we connect the Nextion display to the Arduino using a serial connection. Using a USB to TTL serial converter to make this connection.
2. Next, download the Nextion Editor software from the Nextion website. This software allows us to create the user interface for display..
3. Once we have designed our user interface, We use the Nextion Instruction Set to communicate with the display from Arduino code. The instruction set allows us to set the text of text boxes, change the background color of the display, and perform other actions.
4. We use the Serial library in Arduino to send commands over the serial connection.

Connecting an Arduino with an ESP32

Using UART:

One of the simplest ways to connect an Arduino with an ESP32 is by using UART. Both the Arduino and the ESP32 have built-in UART interfaces, which can be used to send and receive serial data. To connect the two, we can use jumper wires to connect the Arduino's TX pin to the ESP32's RX pin, and the Arduino's RX pin to the ESP32's TX pin. We will also need to connect the two boards' grounds together. Once we have made the physical connections, we can use the Serial library in Arduino and the Serial object in ESP32 to send and receive data.

As this project is situated in the local region, we don't need an active internet connection to manage and watch over our household appliances. Here, we are configuring the ESP32 web server's softAP mode. Therefore, to connect to the "ESP32 Smart Home" Wi-Fi access point, we just need to enter the password. By inputting the IP address of the ESP board in your browser, you can view real-time sensor data and control your appliances from a distance. This can be done with a tablet, smartphone, or PC. Therefore, we do not need an active wifi connection in order to monitor and control your home appliances

Building a home automation system

1. Connect the Nextion display and the relay module to the Arduino. We will need to connect the display to the Arduino's serial port using jumper wires and connect the relay module to the Arduino's digital pins using additional jumper wires.
2. Write the code for the Arduino to read input from the Nextion display and control the relay module. We use the Nextion Instruction Set to define the layout of the user interface on the display and use the digitalWrite function in Arduino to turn the relay on and off.

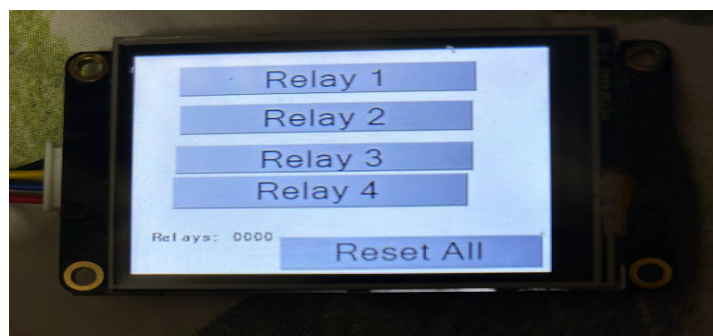


Fig 3.2 Nextion Display(With GUI)

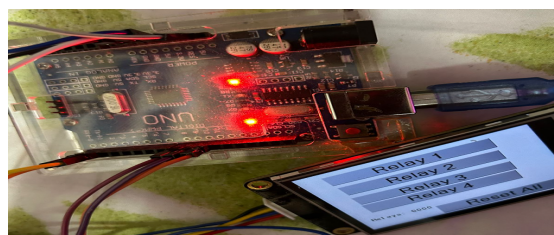


Fig 3.3 Working Model

CHAPTER 4

CONCLUSION

In conclusion, HMI displays have revolutionized the way we interact with home automation systems. By providing a user-friendly interface that can be customized to suit individual needs, HMI displays have made it possible for homeowners to easily control and monitor various home appliances and systems, increasing efficiency, convenience, and energy efficiency.

One of the main advantages of using an HMI display for home automation is the ability to provide centralized control. With a single interface, users can control and monitor all the connected devices in their homes, making it easier to manage and automate routine tasks. Moreover, the ability to customize the interface allows users to create a control system that fits their specific needs, improving the overall user experience.

Another advantage of using HMI displays for home automation is the ability to enable remote control and monitoring. This feature allows users to control and monitor their homes from anywhere, providing greater flexibility and convenience. Moreover, the ability to collect and analyze data on home appliances and systems enables users to make informed decisions that can help them reduce energy consumption and costs.

Despite the many benefits of using HMI displays for home automation, there are also some challenges that need to be addressed. One of the main challenges is ensuring compatibility with different devices and systems. As the number of devices and systems that can be connected to home automation systems increases, it is important to ensure that HMI displays can effectively communicate with all of them.

Another challenge is ensuring security and privacy. As home automation systems become more connected and data-driven, there is an increased risk of cyber-attacks and data breaches. It is therefore important to ensure that HMI displays and the underlying systems are secure and protected against potential threats.

Despite these challenges, the potential benefits of using HMI displays for home automation are significant. By providing a user-friendly and customizable interface that can be remotely controlled and monitored, HMI displays have made it possible for homeowners to easily manage their homes and automate routine tasks. Moreover, the ability to collect and analyze data on home appliances and systems enables users to make informed decisions that can help them reduce energy consumption and costs.

In conclusion, the use of HMI displays for home automation has transformed the way we interact with our homes, providing a more efficient, convenient, and customizable experience. While there are still challenges that need to be addressed, the potential benefits of using HMI displays for home automation are too significant to ignore, and are likely to continue driving innovation in this field for years to come.

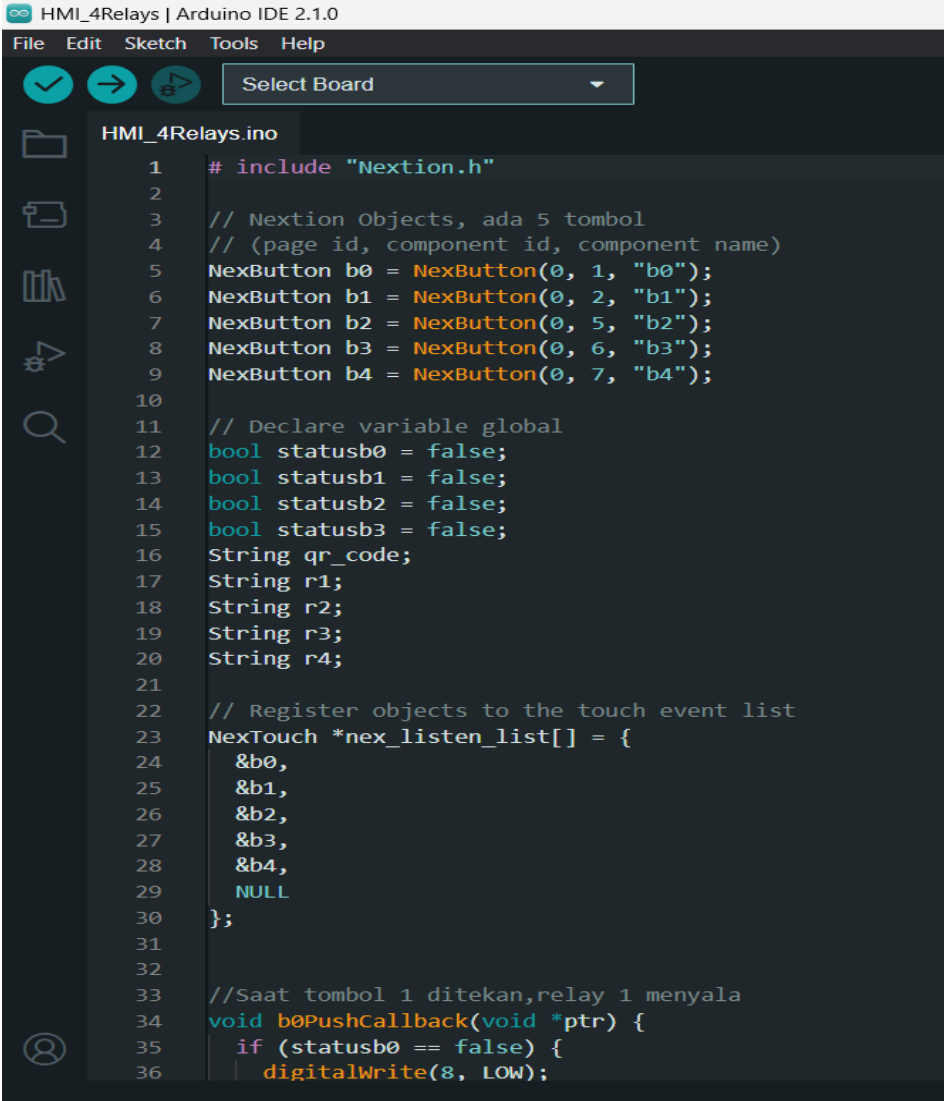
REFERENCES

1. <https://theiotprojects.com/esp32-smart-home-automation-using-dwin-hmi-display/>
2. <https://randomnerdtutorials.com/nextion-display-with-arduino-getting-started/>
3. Implementation of IoT in SmartHomeAutomation.
4. IoT Development for ESP32 and ESP8266 by Peter Hoddie.
5. Programming Arduino: Getting Started With Sketches (second edition).

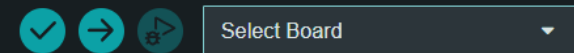
APPENDIX

In this era where everyone is leaning towards smart electronics. Mobile phones have been smart, computers have been smarter and we get to know about the new inventions and improvements in electronic devices and gadgets every day in our life.

3.3 Arduino code



```
HMI_4Relays | Arduino IDE 2.1.0
File Edit Sketch Tools Help
Select Board
HMI_4Relays.ino
1 #include "Nextion.h"
2
3 // Nextion Objects, ada 5 tombol
4 // (page id, component id, component name)
5 NexButton b0 = NexButton(0, 1, "b0");
6 NexButton b1 = NexButton(0, 2, "b1");
7 NexButton b2 = NexButton(0, 5, "b2");
8 NexButton b3 = NexButton(0, 6, "b3");
9 NexButton b4 = NexButton(0, 7, "b4");
10
11 // Declare variable global
12 bool statusb0 = false;
13 bool statusb1 = false;
14 bool statusb2 = false;
15 bool statusb3 = false;
16 String qr_code;
17 String r1;
18 String r2;
19 String r3;
20 String r4;
21
22 // Register objects to the touch event list
23 NexTouch *nex_listen_list[] = {
24     &b0,
25     &b1,
26     &b2,
27     &b3,
28     &b4,
29     NULL
30 };
31
32
33 //Saat tombol 1 ditekan, relay 1 menyala
34 void b0PushCallback(void *ptr) {
35     if (statusb0 == false) {
36         digitalWrite(8, LOW);
```



```
HMI_4Relays.ino
36     digitalWrite(8, LOW);
37     statusb0 = true;
38     //set button color RED
39     b0.Set_background_color_bco(63488);
40
41     //set for QR Code
42     r1 = "1";
43 }
44 else {
45     digitalWrite(8, HIGH);
46     statusb0 = false;
47     //reset button color
48     b0.Set_background_color_bco(48631);
49
50     //set for QR Code
51     r1 = "0";
52 }
53
54 //Set QR Code
55 qr_code = "Relays: " + r1 + r2 + r3 + r4;
56 Serial.print("qr0.txt=");
57 Serial.print("");
58 Serial.print(qr_code);
59 Serial.print("");
60 Serial.write(0xff);
61 Serial.write(0xff);
62 Serial.write(0xff);
63 delay(500);
64
65 //set qr text
66 Serial.print("t0.txt=");
67 Serial.print("");
68 Serial.print(qr_code);
69 Serial.print("");
70 Serial.write(0xff);
71 Serial.write(0xff);
```

```
HMI_4Relays | Arduino IDE 2.1.0
File Edit Sketch Tools Help
Select Board
HMI_4Relays.ino
71 Serial.write(0xff);
72 Serial.write(0xff);
73 }
74
75 //Saat tombol 2 ditekan,relay 2 menyala
76 void b1PushCallback(void *ptr) {
77     if (statusb1 == false) {
78         digitalWrite(9, LOW);
79         statusb1 = true;
80         //set button color RED
81         b1.Set_background_color_bco(63488);
82
83         //set for QR Code
84         r2 = "1";
85     }
86     else {
87         digitalWrite(9, HIGH);
88         statusb1 = false;
89         //reset button color
90         b1.Set_background_color_bco(48631);
91
92         //set for QR Code
93         r2 = "0";
94     }
95
96     //Set QR Code
97     qr_code = "Relays: " + r1 + r2 + r3 + r4;
98     Serial.print("qr0.txt=");
99     Serial.print("");
100    Serial.print(qr_code);
101    Serial.print("");
102    Serial.write(0xff);
103    Serial.write(0xff);
104    Serial.write(0xff);
105    delay(500);
106
```



Select Board

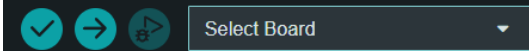


HMI_4Relays.ino

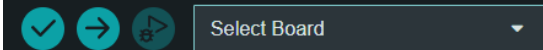


```
106
107 //set qr text
108 Serial.print("t0.txt=");
109 Serial.print('');
110 Serial.print(qr_code);
111 Serial.print('');
112 Serial.write(0xff);
113 Serial.write(0xff);
114 Serial.write(0xff);
115 }
116
117 //Saat tombol 3 ditekan,relay 3 menyala
118 void b2PushCallback(void *ptr) {
119     if (statusb2 == false) {
120         digitalWrite(10, LOW);
121         statusb2 = true;
122         //set button color RED
123         b2.Set_background_color_bco(63488);
124
125         //set for QR Code
126         r3 = "1";
127     }
128     else {
129         digitalWrite(10, HIGH);
130         statusb2 = false;
131         //reset button color
132         b2.Set_background_color_bco(48631);
133
134         //set for QR Code
135         r3 = "0";
136     }
137
138     //Set QR Code
139     qr_code = "Relays: " + r1 + r2 + r3 + r4;
140     Serial.print("qr0.txt=");
141     Serial.print('');
```

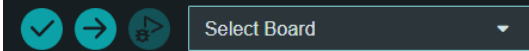
```
HMI_4Relays.ino
141 Serial.print("");
142 Serial.print(qr_code);
143 Serial.print("");
144 Serial.write(0xff);
145 Serial.write(0xff);
146 Serial.write(0xff);
147 delay(500);
148
149 //set qr text
150 Serial.print("t0.txt=");
151 Serial.print("");
152 Serial.print(qr_code);
153 Serial.print("");
154 Serial.write(0xff);
155 Serial.write(0xff);
156 Serial.write(0xff);
157 }
158
159 //Saat tombol 4 ditekan,relay 4 menyala
160 void b3PushCallback(void *ptr) {
161     if (statusb3 == false) {
162         digitalWrite(11, LOW);
163         statusb3 = true;
164         //set button color RED
165         b3.Set_background_color_bco(63488);
166
167         //set for QR Code
168         r4 = "1";
169     }
170     else {
171         digitalWrite(11, HIGH);
172         statusb3 = false;
173         //reset button color
174         b3.Set_background_color_bco(48631);
175
176         //set for OR Code
```



```
HMI_4Relays.ino
212 //reset button color
213 b0.Set_background_color_bco(48631);
214 b1.Set_background_color_bco(48631);
215 b2.Set_background_color_bco(48631);
216 b3.Set_background_color_bco(48631);
217
218 //Set QR Code
219 r1 = "0";
220 r2 = "0";
221 r3 = "0";
222 r4 = "0";
223 qr_code = "Relays: " + r1 + r2 + r3 + r4;
224 Serial.print("qr0.txt=");
225 Serial.print('');
226 Serial.print(qr_code);
227 Serial.print('');
228 Serial.write(0xff);
229 Serial.write(0xff);
230 Serial.write(0xff);
231 delay(500);
232
233 //set qr text
234 Serial.print("t0.txt=");
235 Serial.print('');
236 Serial.print(qr_code);
237 Serial.print('');
238 Serial.write(0xff);
239 Serial.write(0xff);
240 Serial.write(0xff);
241
242 }
243
244 void setup() {
245   Serial.begin(9600);
246   //Initialize Nextion Library
247   nexInit();
```

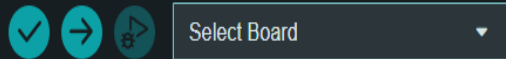
```
HMI_4Relays.ino
247     nexInit();
248
249     // Register relay (output)
250     pinMode(8, OUTPUT);
251     pinMode(9, OUTPUT);
252     pinMode(10, OUTPUT);
253     pinMode(11, OUTPUT);
254
255
256     // Register the push/pop event callback function
257     b0.attachPush(b0PushCallback, &b0);
258     b1.attachPush(b1PushCallback, &b1);
259     b2.attachPush(b2PushCallback, &b2);
260     b3.attachPush(b3PushCallback, &b3);
261     b4.attachPush(b4PushCallback, &b4);
262
263     delay(2000);
264     //Matikan semua relay
265     digitalWrite(8, HIGH);
266     digitalWrite(9, HIGH);
267     digitalWrite(10, HIGH);
268     digitalWrite(11, HIGH);
269     delay(500);
270
271     //Set QR Code
272     r1 = "0";
273     r2 = "0";
274     r3 = "0";
275     r4 = "0";
276     qr_code = "Relays: " + r1 + r2 + r3 + r4;
277     Serial.print("qr0.txt=");
278     Serial.print('');
279     Serial.print(qr_code);
280     Serial.print('');
281     Serial.write(0xff);
282     Serial.write(0xff):
```



```
HMI_4Relays.ino
272   r1 = "0";
273   r2 = "0";
274   r3 = "0";
275   r4 = "0";
276   qr_code = "Relays: " + r1 + r2 + r3 + r4;
277   Serial.print("qr0.txt=");
278   Serial.print('');
279   Serial.print(qr_code);
280   Serial.print('');
281   Serial.write(0xff);
282   Serial.write(0xff);
283   Serial.write(0xff);
284   delay(500);
285
286   //set qr text
287   Serial.print("t0.txt=");
288   Serial.print('');
289   Serial.print(qr_code);
290   Serial.print('');
291   Serial.write(0xff);
292   Serial.write(0xff);
293   Serial.write(0xff);
294   delay(500);
295
296   //set brightness
297   Serial.print("dim=5");
298   Serial.write(0xff);
299   Serial.write(0xff);
300   Serial.write(0xff);
301 }
302
303 void loop() {
304   //When push/pop event occurred execute component in touch event list
305   nexLoop(nex_listen_list);
306 }
307
```

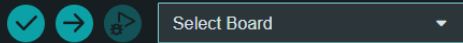
3.4 ESP32 Code

```
sketch_feb28a | Arduino IDE 2.1.0
File Edit Sketch Tools Help
Select Board
sketch_feb28a.ino
1 #include <WiFi.h>
2 #include <WebServer.h>
3 const byte rxPin = 16; //rx2
4 const byte txPin = 17; //tx2
5 // SoftwareSerial mySerial (rxPin, txPin);
6 const char* ssid = "ESP32 Smart Home"; // Enter SSID here
7 const char* password = "12345678"; //Enter Password here
8 /* Put IP Address details */
9 IPAddress local_ip(192, 168, 4, 1);
10 IPAddress gateway(192, 168, 4, 1);
11 IPAddress subnet(255, 255, 255, 0);
12 WebServer server(80);
13 uint8_t light = 23;
14 bool lightstatus = HIGH;
15 uint8_t fan = 22;
16 bool fanstatus = HIGH;
17 uint8_t tv = 21;
18 bool tvstatus = HIGH;
19 uint8_t ac = 19;
20 bool acstatus = HIGH;
21 /* Adresses of all sensors */
22 unsigned char Buffer[9];
23 void setup() {
24     Serial.begin(115200);
25     Serial2.begin(115200);
26     pinMode(light, OUTPUT);
27     pinMode(fan, OUTPUT);
28     pinMode(tv, OUTPUT);
29     pinMode(ac, OUTPUT);
30     dht.begin();
31     WiFi.softAP(ssid, password);
32     WiFi.softAPConfig(local_ip, gateway, subnet);
33     delay(100);
34     server.on("/", handle_OnConnect);
35     server.on("/lighton", handle_lighton);
36     server.on("/lightoff", handle_lightoff);
Offline
```



sketch_feb28a.ino

```
36 server.on("/lightoff", handle_lightoff);
37 server.on("/fanon", handle_fanon);
38 server.on("/fanoff", handle_fanoff);
39 server.on("/tvon", handle_tvon);
40 server.on("/tvoff", handle_tvoff);
41 server.on("/acon", handle_acon);
42 server.on("/acoff", handle_acoff);
43 server.onNotFound(handle_NotFound);
44 server.begin();
45 Serial.println("HTTP server started");
46 }
47
48 if (lightstatus)
49 {
50     digitalWrite(light, HIGH);
51 }
52 else
53 {
54     digitalWrite(light, LOW);
55 }
56 if (fanstatus)
57 {
58     digitalWrite(fan, HIGH);
59 }
60 else
61 {
62     digitalWrite(fan, LOW);
63 }
64 if (tvstatus)
65 {
66     digitalWrite(tv, HIGH);
67 }
68 else
69 {
70     digitalWrite(tv, LOW);
71 }
```



sketch_feb28a.ino

```
71     }
72     if (acstatus)
73     {
74         digitalWrite(ac, HIGH);
75     }
76     else
77     {
78         digitalWrite(ac, LOW);
79     }
80     Data_Arduino_to_Display();
81     Data_Display_to_Arduino();
82 }
83 void handle_OnConnect() {
84     lightstatus = HIGH;
85     fanstatus = HIGH;
86     tvstatus = HIGH;
87     acstatus = HIGH;
88     Serial.println("Light Status: OFF | Fan Status: OFF | TV Status: OFF | AC Status: OFF");
89     server.send(200, "text/html", SendHTML(lightstatus, fanstatus, tvstatus, acstatus));
90 }
91 void handle_lighton() {
92     lightstatus = HIGH;
93     Serial.println("Light Status: OFF");
94     server.send(200, "text/html", SendHTML(true, fanstatus, tvstatus, acstatus));
95 }
96 void handle_lightoff() {
97     lightstatus = LOW;
98     Serial.println("Light Status: ON");
99     server.send(200, "text/html", SendHTML(false, fanstatus, tvstatus, acstatus));
100 }
101 void handle_fanon() {
102     fanstatus = HIGH;
103     Serial.println("Fan Status: OFF");
104     server.send(200, "text/html", SendHTML(lightstatus, true, tvstatus, acstatus));
105 }
106 void handle_fanoff() {
```

```
sketch_feb28a | Arduino IDE 2.1.0
File Edit Sketch Tools Help
Select Board

sketch_feb28a.ino
106 void handle_fanoff() {
107     fanstatus = LOW;
108     Serial.println("Fan Status: ON");
109     server.send(200, "text/html", SendHTML(lightstatus, false, tvstatus, acstatus));
110 }
111 void handle_tvon() {
112     tvstatus = HIGH;
113     Serial.println("TV Status: OFF");
114     server.send(200, "text/html", SendHTML(lightstatus, fanstatus, true, acstatus));
115 }
116 void handle_tvoff() {
117     tvstatus = LOW;
118     Serial.println("TV Status: ON");
119     server.send(200, "text/html", SendHTML(lightstatus, fanstatus, false, acstatus));
120 }
121 void handle_acon() {
122     acstatus = HIGH;
123     Serial.println("AC Status: OFF");
124     server.send(200, "text/html", SendHTML(lightstatus, fanstatus, tvstatus, true));
125 }
126 void handle_acoff() {
127     acstatus = LOW;
128     Serial.println("AC Status: ON");
129     server.send(200, "text/html", SendHTML(lightstatus, fanstatus, tvstatus, false));
130 }
131 void handle_NotFound() {
132     server.send(404, "text/plain", "Not found");
133 }
134 String SendHTML(uint8_t lightstat, uint8_t fanstat, uint8_t tvstat, uint8_t acstat) {
135     String ptr = "<!DOCTYPE html> <html>\n";
136     ptr += "<head><meta name='viewport' content='width=device-width, initial-scale=1.0, user-scalable=no'>\n";
137     ptr += "<title>LED Control</title>\n";
138     ptr += "<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}\n";
139     ptr += "body{margin-top: 20px;} h1 {color: #444444;margin: 20px auto 20px;} h2 {color: #444444;margin-bottom: 20px;}\n";
140     ptr += ".button {display: block;width: 80px;background-color: #3498db;border: none;color: white;padding: 13px 30px;text-decoration: none;font-size: 25px;margin: 0px auto 30px;cursor:
141     ptr += ".button-on {background-color: #3498db;}\n";
Ln 12, Col 22. No board selected
```

Select Board

```
sketch_feb28a.ino
141 ptr += ".button-on {background-color: #3498db;}\n";
142 ptr += ".button-on:active {background-color: #2980b9;}\n";
143 ptr += ".button-off {background-color: #34495e;}\n";
144 ptr += ".button-off:active {background-color: #2c3e50;}\n";
145 ptr += "p {font-size: 14px;color: #888;margin-bottom: 10px;}\n";
146 ptr += "</style>\n";
147 ptr += "</head>\n";
148 ptr += "<body>\n";
149 ptr += "<h1>ESP32 Smart Home</h1>\n";
150 ptr += "<hr></h2>";
151
152 if (lightstat)
153 {
154   ptr += "<p>Light Status: OFF</p><a class=\"button button-off\" href=\"/lightoff\">OFF</a>\n";
155 }
156 else
157 {
158   ptr += "<p>Light Status: ON</p><a class=\"button button-on\" href=\"/lighton\">ON</a>\n";
159 }
160 if (fanstat)
161 {
162   ptr += "<p>Fan Status: OFF</p><a class=\"button button-off\" href=\"/fanoff\">OFF</a>\n";
163 }
164 else
165 {
166   ptr += "<p>Fan Status: ON</p><a class=\"button button-on\" href=\"/fanon\">ON</a>\n";
167 }
168 if (tvstat)
169 {
170   ptr += "<p>TV Status: OFF</p><a class=\"button button-off\" href=\"/tvoff\">OFF</a>\n";
171 }
172 else
173 {
174   ptr += "<p>TV Status: ON</p><a class=\"button button-on\" href=\"/tvon\">ON</a>\n";
175 }
176 if (acstat)
```

sketch_feb28a.ino

```
176   if (acstat)
177   {
178     ptr += "<p>AC Status: OFF</p><a class=\"button button-off\" href=\"/acoff\">OFF</a>\n";
179   }
180   else
181   {
182     ptr += "<p>AC Status: ON</p><a class=\"button button-on\" href=\"/acon\">ON</a>\n";
183   }
184   ptr += "</body>\n";
185   ptr += "</html>\n";
186   return ptr;
187 }
188 void Data_Display_to_Arduino()
189 {
190   if (Serial.available())
191   {
192     for (int i = 0; i <= 8; i++) //this loop will store whole frame in buffer array.
193     {
194       Buffer[i] = Serial.read();
195     }
196     if (Buffer[0] == 0x5A)
197     {
198       switch (Buffer[4])
199       {
200         case 0x25: //for light
201           if (Buffer[8] == 1)
202           {
203             digitalWrite(light, LOW);
204           }
205           else
206           {
207             digitalWrite(light, HIGH);
208           }
209           break;
210         case 0x30: //for fan
211           if (Buffer[8] == 1)
```


Select Board

sketch_feb28a.ino

```
211     if (Buffer[8] == 1)
212     {
213         digitalWrite(fan, LOW);
214     }
215     else
216     {
217         digitalWrite(fan, HIGH);
218     }
219     break;
220 case 0x35: //for TV
221     if (Buffer[8] == 1)
222     {
223         digitalWrite(tv, LOW);
224     }
225     else
226     {
227         digitalWrite(tv, HIGH);
228     }
229     break;
230 case 0x40: //for AC
231     if (Buffer[8] == 1)
232     {
233         digitalWrite(ac, LOW);
234     }
235     else
236     {
237         digitalWrite(ac, HIGH);
238     }
239     break;
240 default:
241     break;
242 }
243 }
244 }
245 }
246 }
```

PLAGIARISM REPORT

AMBAR S TIWARI 191039

ORIGINALITY REPORT

13% SIMILARITY INDEX	8% INTERNET SOURCES	1% PUBLICATIONS	10% STUDENT PAPERS
--------------------------------	-------------------------------	---------------------------	------------------------------

PRIMARY SOURCES

1	Submitted to Pace University Student Paper	3%
2	randomnerdtutorials.com Internet Source	2%
3	pcbknow.com Internet Source	2%
4	Submitted to University of Northampton Student Paper	1%
5	Submitted to Durban University of Technology Student Paper	1%
6	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	1%
7	Submitted to Higher Education Commission Pakistan Student Paper	1%
8	theiotprojects.com Internet Source	1%

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT
PLAGIARISM VERIFICATION REPORT

Date:

Type of Document (Tick): PhD Thesis M.Tech Dissertation/ Report B.Tech Project Report Paper

Name: _____ Department: _____ Enrolment No _____

Contact No. _____ E-mail. _____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/ revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor)

Signature of HOD

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	<ul style="list-style-type: none"> • All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String 		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature

Librarian

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com