**VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY**

**UNIVERSITY OF INFORMATION TECHNOLOGY**

**FACULTY OF COMPUTER ENGINEERING**

**Full name: Đỗ Thanh Sơn Full name: Trương Hữu Trường Sơn**

**Student ID: 21522551 Student ID: 21522559**

**Class: CE103.N22.MTCL Class: CE103.N22.MTCL**

**ELECTRONIC PROJECT CLOCK**

**DESIGN ELECTRONIC CLOCK DISPLAY ON 7 SEGMENT LED USING 8051 MICROCONTROLLER**

**THIẾT KẾ ĐỒNG HỒ ĐIỆN TỬ HIỂN THỊ TRÊN LED 7 ĐOẠN SỬ DỤNG VI ĐIỀU KHIỂN 8051**

**MID-TERM ELECTRONIC CLOCK PROJECT REPORT**

**MENTOR**

**PhD. TRI NHUT DO**

**VIETNAM NATIONAL UNIVERSITY SOCIALIST REPUBLIC OF VIETNAM**

**HO CHI MINH CITY Independence – Freedom – Happiness**

**UIVERSITY OF INFORMATION**

**TECHNOLOGY**

**ELECTRONIC PROJECT CLOCK**

|  |  |
| --- | --- |
| **VIETNAMESE PROJECT NAME: Thiết kế đồng hồ điện tử hiển thị trên led 7 đoạn** | |
| **ENGLISH PROJECT NAME: Design an electronic watch display on 7-Segment LED** | |
| **Instructor PhD. Tri Nhut Do, Faculty of Computer Engineering** | |
| **Implementation time: From: 01/04/2023 To: 28/04/2023** | |
| **Student Perform: Do Thanh Son – 21522551 and Truong Huu Truong Son – 21522559** | |
| **Overview the topic: The invention of the clock displayed on a 7-Segment LED**  **The goal of the subject: 8**  **Methods of implementation: Simulation on proteus and realistic design**  **Main contents of the topic: Provide a visual representation of time through the use of a digital display.** | |
| **Certification of Instructor** | **HCM city, 2023 May 5th**  **Student** |

**TABLE OF CONTENTS Page**

**Chapter 1. INTRODUCTION 4**

**Chapter 2. AN OVERVIEW 5**

2.1 Brief analysis **5**

**2.2 Block diagram 5**

**2.3 Flow chart 6**

**Chapter 3. IMPLEMENTATION 7**

**3.1 Describe components. 7**

**3.2 Circuit on breadboard 8**

**3.3 Code for the program and explanation 9**

**Chapter 4. EXPERIMENT 19**

**4.1 Setup scenarios 19**

**4.2 Evaluation 19**

**Chapter 5. CONCLUSION 20**

**5.1 Comparision 20**

**5.2 Development 20**

**REFERENCES: 21**

Microproccesor-Microcontroller

Project

**CHAPTER 1. INTRODUCTION**

The idea behind displaying a clock on a 7-segment LED is to provide a simple and easy-to-read visual representation of time. The 7-segment LED is a popular choice for displaying digits because it is compact, easy to interface with digital electronics, and can display all the digits from 0 to 9. By using a clock circuit to generate the time and a microcontroller or other digital logic to drive the 7-segment LED display, it is possible to create a simple, reliable, and low-cost timekeeping solution that can be used in a wide range of applications, such as watches, clocks, and other timekeeping devices.



*Figure 1. Clock displaying with large LED 7-Segment 6 digits.*

**CHAPTER 2. AN OVERVIEW**

**2.1 Brief analysis**

The topic of designing an electronic watch display on a 7-segment LED is an interesting and challenging project for electronics enthusiasts. It involves the use of microcontrollers, programming languages, and circuit design to create a functional and accurate timekeeping device.

The project requires a solid understanding of digital electronics, particularly in the areas of microcontrollers and interfacing with 7-segment displays. It also requires proficiency in programming languages such as assembly to write the necessary code to control the display and implement timekeeping functions.

Overall, the project provides a great opportunity to learn and apply practical skills in electronics and programming, while also resulting in a useful and functional product.

**2.2 Block diagram**

Diagram

Description automatically generated

*Figure 2. Visual block diagram illustrating the function associated with the microcontroller.*

* 1. **Flow chart**

Diagram

Description automatically generated

*Figure 3. Flowchart Illustrating the Operation of LED 7-Segment Display*

**CHAPTER 3. THEORY BASIS**

**3.1 Introduction to the 8051 Family and AT89S52 microcontroller**

**3.1.1 Overview of the 8051 Family**

The 8051 family is a popular family of microcontrollers that was introduced by Intel in 1980. It is a widely used 8-bit microcontroller family that is based on a Harvard architecture, which means it has separate memory spaces for program and data. The 8051 family has several variants, each with its own set of features, but they all share the same basic architecture.

An 8051 microcontroller typically includes the following features:

* Central Processing Unit (CPU)
* On-chip memory
* Input/output (I/O) ports
* Timers/counters
* Interrupts
* Serial communication ports
* Analog-to-digital converter (ADC)

**A picture containing text, screenshot, font, number

Description automatically generated**

*Table 1. Comparisionn of 8051 Family members*

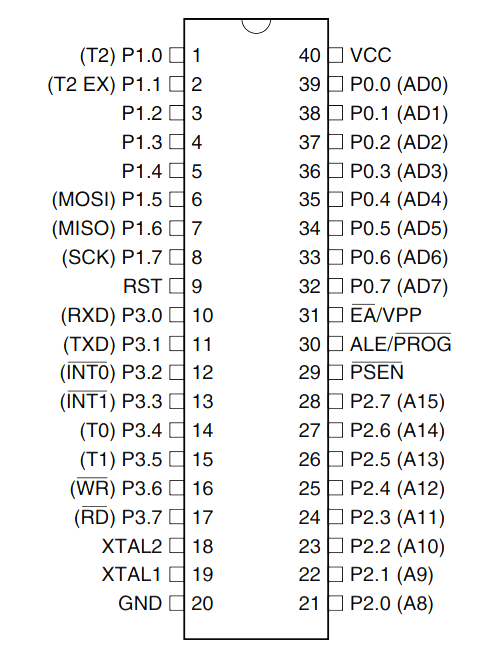
**A diagram of a computer system

Description automatically generated with low confidence**

*Figure 4. Block Diagram of 8051 microcontroller*

**3.1.2 AT89S2 microcontroller**

The AT89S2 is a low-power, high-performance 8-bit microcontroller that belongs to the 8051 family. It was developed by Atmel Corporation, now part of Microchip Technology. The AT89S2 microcontroller is a popular choice for a wide range of applications, including automotive, industrial, and consumer electronics.



*Figure 5. AT89S52 Microcontroller Pin out*

Here are some key features of the AT89S2 microcontroller:

* CPU: The AT89S2 microcontroller has an 8-bit CPU that can operate at up to 24 MHz.
* On-chip memory: The AT89S2 microcontroller has 32 KB of flash memory for program storage and 2 KB of SRAM for data storage.
* I/O ports: The AT89S2 microcontroller has four 8-bit I/O ports that can be used to interface with external devices.
* Timers/counters: The AT89S2 microcontroller has two 16-bit timers/counters that can be used for timing and counting operations.
* Interrupts: The AT89S2 microcontroller supports both external and internal interrupts, allowing it to respond to external events in real-time.
* Serial communication ports: The AT89S2 microcontroller has a built-in UART for serial communication, which can be used to communicate with other devices using standard protocols such as RS-232 or SPI.
* Analog-to-digital converter (ADC): The AT89S2 microcontroller includes an on-chip 10-bit ADC that can be used to convert analog signals into digital values for processing.
* Low-power modes: The AT89S2 microcontroller includes several low-power modes that allow it to conserve power when it is not in use.

**3.2 Kit 8051 V1 Socket**

The Kit 8051 V1 is a development board designed for use with 8051 microcontrollers. It provides an easy and convenient way to develop and test 8051-based projects without the need for complex hardware configurations.

The following features provided by the Kit 8051 V1 Socket:

* Compatible with chip models: The Kit 8051 V1 Socket is designed to work with a variety of 8051 microcontroller models, including the AT89S52 and AT89S51, among others.
* Dimensions: The board measures 45mm x 92mm, making it compact and easy to use in a variety of projects.
* Power supply: The board can be powered by a 5V DC supply, which can be connected via a DC jack or through the ISP interface.
* Crystal oscillator: The Kit 8051 V1 Socket includes an 11.0592 MHz crystal oscillator, which provides a stable clock signal for the microcontroller.
* Weight: The board weighs just 45g, making it lightweight and easy to transport.

A picture containing electronic component, electronic engineering, circuit component, passive circuit component

Description automatically generated

*Figure 6. Kit8051 V1 Socket*

Diagram, schematic

Description automatically generated

*Figure 7. AT89S52 Minimum development board*

**3.3 Overview of LED 7-Segment 2 digits 2821AH**

A picture containing black and white, design

Description automatically generated with medium confidence

Figure 6.1 LED 7-Segment Display 2821AH

The LED 7-Segment Display 2821AH is a specific model of the LED 7-Segment display family. Here's an overview of its features:

* Display: The 2821AH display consists of one 7-segment display, which can display numerical values from 0 to 9 and some letters like A, B, C, D, E, F, H, L, and P.
* LED technology: The display is made up of individual light-emitting diodes (LEDs), which provide bright and clear visibility even in low-light conditions.
* Pin configuration: The display has 12 pins, with one pin for each of the seven segments (a to g), one pin for the decimal point (DP), and four pins for the common cathode.
* Dimensions: The dimensions of the display are 19mm x 12.6mm x 7.9mm.
* Color: The 2821AH display emits red light.
* Forward voltage and current: The forward voltage of the LED segments is typically around 2.0V and the recommended current is around 20mA.
* Driver circuit: To drive the 2821AH display, a driver circuit is needed to provide the appropriate current to each LED segment. This can be done using a simple transistor-based circuit or with specialized driver ICs.

Overall, the LED 7-Segment Display 2821AH is a small and versatile component for displaying numerical values and some letters in digital circuits and microcontroller projects. Its bright and clear visibility, along with its small size, makes it suitable for a wide range of applications, such as digital clocks, timers, and counters.

Diagram, engineering drawing

Description automatically generated

*Figure 6.2 Dimension and Diagram of the 2821AH*

**3.4 Overview of USBASP USBISP+**

A close-up of a usb cable

Description automatically generated with low confidence

*Figure 7.1 Charging circuit 89/AVR USBASP USBISP+*

The Charging Circuit 89/AVR USBASP USBISP+ is a device used for charging rechargeable batteries and programming microcontrollers like 89/AVR via USB. Here's an overview of its features:

* Charging: The charging circuit is designed to charge a variety of rechargeable batteries, including Li-ion, NiMH, and NiCd batteries. It has a built-in charging management system that protects the battery from overcharging, over-discharging, and short-circuiting.
* Programming: The circuit supports programming of microcontrollers like 89/AVR using a USBASP or USBISP programmer. It has a 10-pin programming header that connects to the programmer and the microcontroller.
* USB interface: The circuit is powered and controlled via USB, with a USB Type B port for connection to a computer or USB charger.
* Battery status indicator: The circuit has a built-in battery status indicator that shows the charging status of the battery, with a red LED indicating charging and a green LED indicating fully charged.
* Power supply: The circuit can be powered via USB or an external power supply, with a voltage range of 5V to 12V.
* Protection: The circuit has a built-in reverse polarity protection that prevents damage to the circuit if the battery is inserted incorrectly.

Overall, the Charging Circuit 89/AVR USBASP USBISP+ is a versatile device that combines battery charging and microcontroller programming in one compact package. Its built-in safety features protect the battery and the circuit from damage, while its USB interface provides convenient and easy-to-use connectivity.

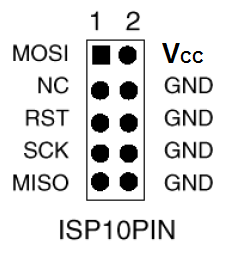


Figure 7.2 USBASP Pinout

**CHAPTER 4. IMPLEMENTATION**

* 1. **Describe components.**

**4.1.1 Selected Components**

1. Breadboard Jumper Wire Set
2. Bus wire, male-female sheath 15 cm
3. Bus wire, female double-ended sheath 15 cm
4. Bus wire, male double-ended sheath 15 cm
5. Breadboard 1660 holes ZY-204
6. Resistor 10K 1/4W 1%
7. Button 6X6X5MM 2 pins
8. 2SD965-R D965 2SD965 NPN Transistor 20V 5A 1W
9. Buzzer 12095 5V
10. AT 89S52-24PU
11. Charging circuit 89/AVR USBASP USBISP+
12. Development KIT 8051
13. 7-segment leds 2 numbers 0.36 inch common cathode

**4.1.2 Schematic**

Diagram, schematic

Description automatically generated

*Figure 8. Schematic Design in Quartus*

*(note: The original Quartus library does not include the AT89S52 microcontroller. However, since the AT89C51 has similar functions, it can be used as a substitute for the AT89S52, especially for simulation purposes.)*

* 1. **Circuit on breadboard**

**Before conducting the simulation of clock display in real life, a series of tests were performed to ensure that all hardware and software components worked properly. This included testing each component separately. The AT89S52 was then placed on the kit 8051 v1 socket, which was connected to a breadboard that also contained LED lights and a 7-segment display. The connections were checked against the schematic design to ensure their accuracy and integrity.**

**To implement the AT89S52, the code was compiled in Assembly language using Keil C µVision. After compilation, the code was uploaded onto the AT89S52 through USBISP. The code was also thoroughly tested for bugs and errors, and necessary adjustments were made to improve its functionality.**

****

***Figure 9. Fully function circuit based on Schematic.***

* 1. **Code for the program and explanation**
* Demonstration:



*Figure 10. Locate where the actual program code is located.*

* Explanation:

These two lines set the starting address for the main program code in the microcontroller's memory and then jump to the main subroutine to start executing the program.

* Demonstration:

Text

Description automatically generated

*Figure 11. Sets an alarm to trigger when the time reaches 59 minutes and 50 seconds.*

* Explanation:

It checks if the clock has reached 59 minutes and 50 seconds and sets Port 0.5 to activate the alarm. It then waits for a delay using the ACALL DELAY instruction and jumps back to the main time-keeping loop.

* Demonstration:

Table

Description automatically generated

*Figure 12. Setting Up Clock Time: Main Program Code*

* Explanation:

The "SETUP\_TIME" section initializes the clock time registers to 12:58:00 (hours:minutes:seconds).

The "RESTART\_TIME" section resets the clock time registers to 00:00:00 (hours:minutes:seconds) when the clock time reaches 23:59:59.

The "RESTART\_TIME\_2" section decrements the hours register and resets the clock time to 23:##:## if it reaches 00:##:##. ( note: “#” don’t care about that value )

* Demonstration:

Text

Description automatically generated

*Figure 13. Displaying Numbers and Receiving Signals in 7-Segment LED Clock*

* Explanation:

The code first calls a subroutine called SHOW\_NUM to display a number on a 7-segment LED display. It then enters a loop to wait for signals from external sources, which are received through the input pins of the microcontroller (P0.0 to P0.4).

If a signal is received, the code branches to the appropriate label (SIG\_H\_I, SIG\_M\_I, SIG\_H\_D, or SIG\_M\_D) to adjust the displayed time accordingly. If the signal received is for restarting the clock (P0.4), the code jumps to the RESTART\_TIME label.

If no signal is received, the code jumps to the ALARM\_ST label to check if the alarm needs to be activated.

If the time displayed on the LED needs to be cleared, the code jumps to the L1 label to turn off the LED display. Finally, the code calls a subroutine called DELAY to introduce a delay before repeating the loop.

* Demonstration:

Graphical user interface, text, table

Description automatically generated

*Figure 14.* Clock Time Conditions for Seconds and Minutes Increment

* Explanation:

The code block labeled "Time\_condition" increments the seconds value of the clock and checks if it has reached 10 seconds. If it hasn't, it jumps to the "DISPLAY" label to display the updated time. If it has reached 10 seconds, it resets the seconds value to 0 and increments the minutes value. It then checks if the minutes value has reached 6 (indicating that an hour has passed) and updates the minutes accordingly.

The code block labeled "SIG\_M\_I" increments the minutes value of the clock and checks if it has reached 10 minutes. If it hasn't, it jumps to the "DISPLAY" label to display the updated time. If it has reached 10 minutes, it resets the minutes value to 0 and increments the hours value. It then checks if the hours value has reached 6 (indicating that a day has passed) and updates the hours accordingly.

* Demonstration:

Text

Description automatically generated

*Figure 15.* Increase Hours of the Clock

* Explanation:

This block of code is responsible for increasing the hours value of the clock.

It checks the current value of the hours in register R5, and XORs it with 2D (which is the value of 45 in decimal) to determine if the value is 45 or not.

+ If it is 45, the code jumps to INCRE\_H1\_2 to increment the tens digit of the hours value.

+ If it is not 45, the code jumps to INCRE\_H2 to increment the units digit of the hours value.

The code also includes checks to ensure that the hours value does not exceed 23, as well as to reset the hours value to 0 and the tens digit to 2 if the value reaches 24. Finally, the code jumps to the RESTART\_TIME label to continue the clock cycle.

* Demonstration:

Text

Description automatically generated

*Figure 16.* Clock Time Conditions for Minutes Decrement

* Explanation:

This code is a clock time condition for decrementing minutes. It starts with the label "SIG\_M\_D". It decreases the value in register R2, which holds the current value of the minutes. It then checks if the value of R2 is equal to -1D (which is equivalent to 29 in decimal). If R2 is not equal to -1D, it jumps to the label DISPLAY. If R2 is equal to -1D, it moves the value 9D into R2 (which is equivalent to 59 in decimal).

Then it decreases the value in register R3, which holds the current value of the hours. It checks if the value of R3 is equal to -1D. If R3 is not equal to -1D, it jumps to the label DISPLAY. If R3 is equal to -1D, it moves the value 5D into R3 (which is equivalent to 23 in decimal).

Overall, this code checks if the minutes have reached their minimum value of 0, and if so, it decreases the hours and sets the minutes to their maximum value of 59.

* Demonstration:

Text

Description automatically generated

*Figure 17.* Decrease Hours of the Clock

* Explanation:

This code is responsible for decreasing the hours of the clock. The program first checks the value of the hours in register R5, and if the value is 2D (indicating that it is 23 hours), it jumps to DECRE\_TH0 to decrease the tens digit of the hours. If the value is not 2D, the program jumps to DECRE\_TH1 to decrease the units digit of the hours.

After decrementing the hours, the program checks if the value is not equal to -1D (indicating that it is still a valid hour value). If it is a valid value, it jumps to DISPLAY to update the display. If the value is not valid, the program sets the hours to 9D (indicating that it is 9 hours) and jumps to DECRE\_H1\_1 to decrease the tens digit of the hours.

After decrementing the tens digit of the hours, the program checks if the value is still a valid hour value. If it is, it jumps to DISPLAY to update the display. If the value is not valid, the program sets the hours to 9D and jumps to RESTART\_TIME\_2 to reset the clock.

* Demonstration:

Text, table

Description automatically generated with medium confidence

Text

Description automatically generated

*Figure 18.* Displaying a number on multiple 7-segment LED displays using Port 1 and Port 2

* Explanation:

It uses the ports P1 and P2 for transmitting the signal to the display. Port P1 is used to send the data to the display and port P2 is used to control which segment should be on at any given time.

The code initializes the loop counter R6 to 74 and loads the memory pointer DPTR with the address of MYDATA. It then enters a loop where it reads data from memory and sends it to the LED display using port P1. The data is sent to each segment of the display in turn, with port P2 used to control which segment is on.

The delay between each segment is controlled by the subroutine DELAY, which is not shown in the code snippet. The loop counter is decremented with each iteration, and the loop continues until the counter reaches zero. The program then returns to its calling function.

* Caculation of the displaying time of LED 7 segment display:

The label "DELAY" in this code represents a time delay of 2227,405 microseconds. When we add the time taken by the five instructions outside of the "DELAY" label, which is 2227.405 us + 6 x 1,085us = 2233.915 us, we get a total delay of 2233,915 us. This means that a single LED 7 Segment display requires a delay of 2233,915 microseconds.

Since we have six 7 seg LEDs in this project, the total delay required for all of them is 2233.915 microseconds x 6 = 13403,49 microseconds, which is equivalent to 0.013 seconds.

For the "LOOP" loop, which repeats 74 times, the total delay is 74 x 13403.49 microseconds = 999,042.26 microseconds, or 0.99904226 seconds.

* Demonstration:

Text

Description automatically generated

*Figure 19. Large Delay using nested loop which this delay approximates to 0,0022s.*

* Explanation:

For the AGAIN1 loop, the code executes the DJNZ (Decrement and Jump if Not Zero) instruction 100 times, which takes 1,085 microseconds (us) per iteration. Since the loop runs twice, the total time for the AGAIN1 loop is 2 x 100 x 1,085 us = 217 us.

The AGAIN2 loop repeats the AGAIN1 loop 10 times, so the total time for the AGAIN2 loop is 10 x 217 us = 2170 us. However, there are additional instructions at the start and end of the AGAIN2 loop - MOV R3,#100 and DJNZ R2,AGAIN2 - that also take time to execute. These instructions take 3 x 10 x 1,805 us = 54,15 us.

In addition, there are three instructions - SETB PSW.4, MOV R2,#10, and CLR PSW.4 - that take a total of 3 x 1,085 us = 3,255 us.

Therefore, the total delay time for the code is 2170 us + 54,15 us + 3,255 us = 2227,405 us, which is approximately 0,0022 seconds.

* Demonstration:

Logo, company name

Description automatically generated

*Figure 20. Hexadecimal Lookup Table for 7-Segment Display*

* Explanation:

This code defines a lookup table of hexadecimal values for use with a 7-segment display. Each byte of the table represents a hexadecimal digit (0-9, A-F) and the corresponding binary pattern to display that digit on a 7-segment display.

This table can be used to easily display numbers or other data on a 7-segment display by indexing into the table with the desired value and outputting the corresponding binary pattern to the display.

* Look up table:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Digit** | **P/h P1.7** | **G**  **P1.6** | **F**  **P1.5** | **E P1.4** | **D P1.3** | **C P1.2** | **B P1.1** | **A P1.0** | **Hex Port 1** |
| **0** | **0** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **3FH** |
| **1** | **0** | **0** | **0** | **0** | **0** | **1** | **1** | **0** | **06H** |
| **2** | **0** | **1** | **0** | **1** | **1** | **0** | **1** | **1** | **5BH** |
| **3** | **0** | **1** | **0** | **0** | **1** | **1** | **1** | **1** | **4FH** |
| **4** | **0** | **1** | **1** | **0** | **0** | **1** | **1** | **0** | **66H** |
| **5** | **0** | **1** | **1** | **0** | **1** | **1** | **0** | **1** | **6DH** |
| **6** | **0** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **7DH** |
| **7** | **0** | **0** | **0** | **0** | **0** | **1** | **1** | **1** | **07H** |
| **8** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **7FH** |
| **9** | **0** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **6FH** |

***Table 2.*** *lookup table of hexadecimal values 0-9*

**CHAPTER 5. EXPERIMENT**

**5.1 Setup scenarios**

1. **Purpose: Set up an electronic clock to display the current time on a 7-segment LED display.**
2. **Audience and objective: The audience is anyone who wants to know the current time. The objective is to provide accurate information about the current time on the 7-segment LED display.**
3. **Situation and context: The electronic clock is set to display the current time in hours, minutes, and seconds format. Users can set the time or change the system clock.**
4. **Options and actions:**

* **Users can set a new time or adjust the current time.**
* **The electronic clock will automatically display the time in hours, minutes, and seconds format.**
* **If no action is taken, the electronic clock will continue to display the current time.**

1. **Description of the result:**

* **When the clock is activated, it will automatically update the current time and display it on the 7-segment LED display.**
* **If the user sets a new time or adjusts the current time, the clock will update and display the new time.**
* **If no action is taken, the clock will continue to display the current time.**

**5.2 Evaluation**

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Score** | **Description** |
| **Accuracy** | **9** | **The actual time with the time displayed on the watch is relatively consistent (Insignificant error ).** |
| **Readability and clarity** | **8** | **There are occasional blurs or errors showing a few characters incorrectly.** |
| **User-friendliness and convenience** | **8** | **Simple design, easy to use but quite huge to carry around.** |
| **Overall performance and reliability** | **8** | **Low power consumption, high stability, not high durability because it is only installed on the board.** |

**CHAPTER 6. CONCLUSION**

* 1. **Comparision**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Components** | **Degree of completion** | **Functionality** | | **Note** |
| **Real life circuit** | **Simulation on Quartus** |
| **Button** | **7** | **Yes** | **Yes** | **The performance is relatively good, but due to the short delay time for signal reception, it only works when the button is held down.** |
| **Buzzer** | **10** | **Yes** | **No** | **The operation is excellent with a loud sound output.** |
| **LED 7-Segment** | **8** | **Yes** | **Yes** | **The performance is generally good, with clear and legible display most of the time. However, there are occasional blurs or errors showing a few characters incorrectly.** |
| **AT89S52** | **10** | **Yes** | **Yes** | **The performance is excellent. No error in executing code** |

|  |  |  |
| --- | --- | --- |
| **Testing criteria** | **Degree of completion** | **Note** |
| **Performance** | **8** | **The system performs well with an accurate delay as simulated.** |
| **Durability** | **7** | **Some components are not very secure.** |
| **Aesthetics** | **4** | **Slightly large and quite bulky.** |

* 1. **Development**
* Change the color of the LED lights: Change the color of the LED lights to create a different effect on the clock. For example, use red LED lights for the hour digits and green LED lights for the minute digits.
* Change the size of the clock: Change the size of the clock to suit user needs. If the client want to create a compact clock, for example using mini 7-segment LED displays and shrink other components.
* Add an automatic clock-off feature: Add an automatic clock-off feature after a certain period to save energy.
* Add a date display feature: Add a date display feature to the clock which help tracking time more accurately.
* Add a notification display feature: Add a simple notification display feature on the clock. For example, "Worktime!",”Breakfast time!”

**REFERENCES.**

[1]:<https://hacoled.com/led-7-doan-la-gi/> ( What is 7 segment LED? Working principle of 7-segment LED, Control methods )

[2]:<https://rmkcet.ac.in/eee/Notes/Micro_Processor_Based_System_Design_EE6502/UNIT%20III%20Introduction%20to%20Microcontroller.pdf>( Pin Description of the 8051 )

[3]:<https://www.youtube.com/watch?v=UFQ0R97q_Jc>( Clock simulation )

[4]:<https://drive.google.com/file/d/1570YwGT7Rt7xnVgs6V6nglDo2fNDHdOm/view> ( I/O Port Programming )

[5]:<https://www.youtube.com/watch?v=A4X_puCst4k&t=827s>( Minimum configuration of 8051)

[6]:<https://www.youtube.com/watch?v=L6gwURPOxGY> ( Minimum circuit diagram of 8051 )