System Design and Performance

Over the past week, we've made significant progress on our alarm clock project. We've successfully integrated multiple libraries to manage the IR sensors and the LCD display, ensuring seamless operation. The core functionality is powered by robust C code, which we've meticulously debugged and optimized for efficiency. The result is a fully operational alarm clock that stands as a testament to our collaborative effort and technical skill. It's been a rewarding journey from concept to creation, and the clock now serves as a functional piece of technology crafted with precision and care. A recording of the artifact can is viewable at the link located in the footnote.[1]

To create the clock, we utilized the materials listed in Table 1. The total cost of the materials was around $44, but this value will be lower in a production situation because this is only a prototype and our materials are not purchased in bulk. The system's schematic is shown in Figure 1. Figure 2 is an image of the constructed system. The LCD used in the physical circuit has an extra ground and power pin for controlling the contrast of the display – these pins are not shown in Figure 1.

To program the Arduino Uno, we used the avr-gcc compiler along with two libraries – one for the LCD interface and another for the IR sensor. The code for the IR sensor was found on Github.[2] The code written for the alarm clock is hosted on Github.[3]

Using a multimeter, we found that the clock used 50mA of current when connected to a 9V battery. One battery provides around 550 mAh, so this system can only run for 11 hours on a single battery.

Item	Quantity	Unit Cost	Total Cost
Arduino Uno R3	1	$27.50	$27.50
HD44780 LCD	1	$10.00	$10.00
IR Remote + Sensor	1	$5.00	$5.00
Piezo Speaker	1	$0.20	$0.20
10kΩ Potentiometer	1	$1.00	$1.00
220Ω Resistor	2	$0.06	$0.12
			$43.82

Table 1 Bill of materials.

[1] https://drive.google.com/file/d/1nEkJ9h8Lekg8FIJa-DfSSFg-iaVfNQxE/view?usp=sharing
[2] https://github.com/sparkletron/AVR-LIBRARY-IR_REMOTE_RECV?tab=GPL-2.0-1-ov-file
[3] https://github.com/Mikaburrie/ECE484-Alarm-Clock

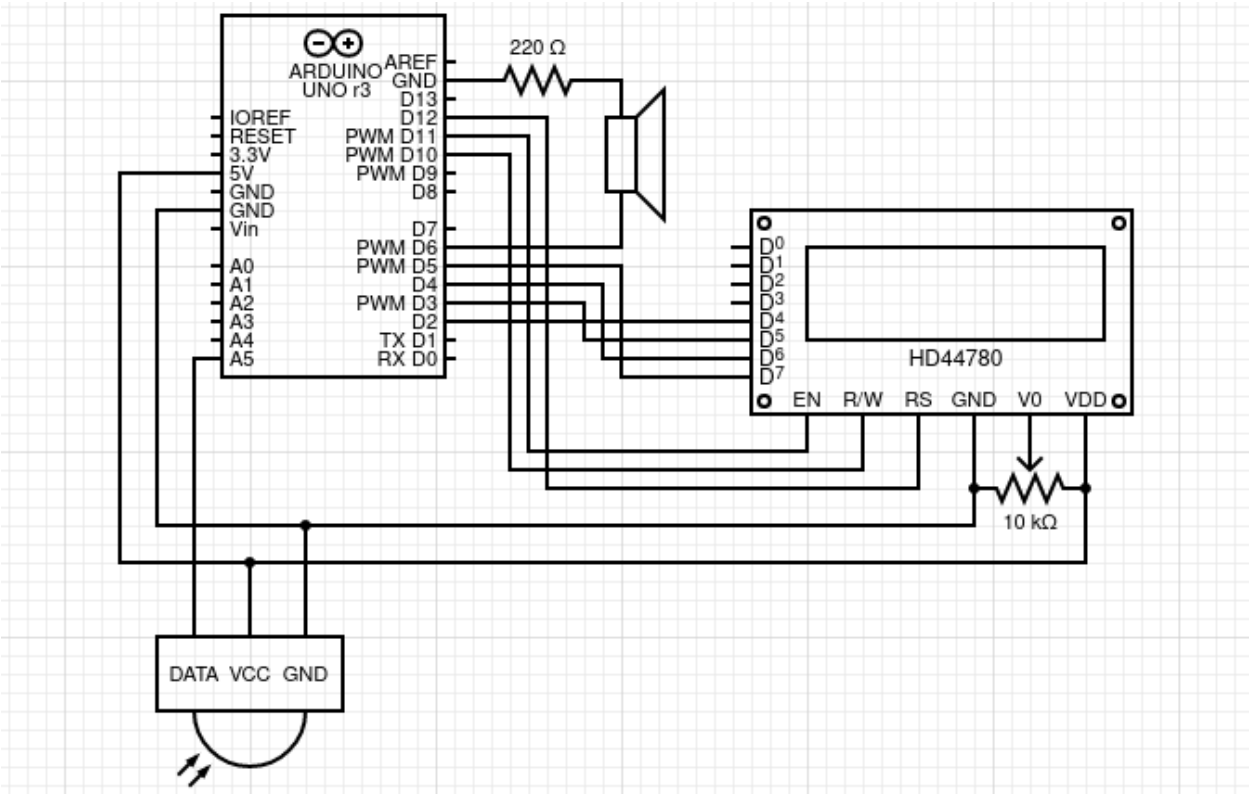

Figure 1 Schematic representation of the finished design.

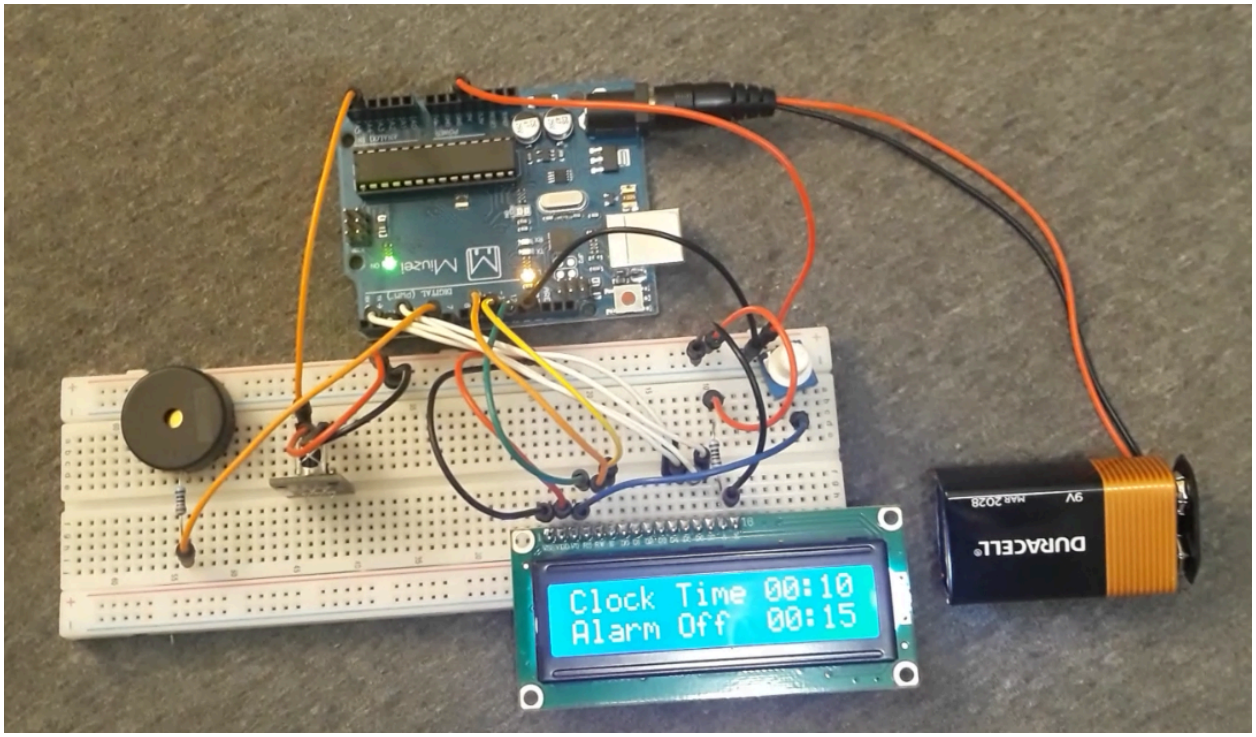

Figure 2 The constructed system.

Group Organization and AI Usage

Our project group is composed of three dedicated members, each with a specialized role that contributed to the successful completion of our alarm clock project. Erik was the programming expert, meticulously crafting the C code that serves as the backbone of our alarm clock's functionality. Mika took charge of the physical assembly, piecing together the hardware components with precision to ensure everything worked in harmony. Sam played a crucial role in library research and the application of AI tools, which were instrumental in generating this report. Together, our group's structured approach and clear division of labor have been pivotal in achieving our project goals.

Conclusion

In summary, our project group has successfully developed an operational alarm clock over the past week, leveraging each member's unique skills. Erik's programming prowess, Mika's assembly expertise, and Sam's research and AI tool application have been instrumental in our project's success. The clock is a product of our combined efforts, integrating C code with IR sensors and LCDs through the use of multiple libraries. When it comes to C programming in embedded system design, it's fascinating to consider the potential of modular programming. This approach allows for the separation of functionality into independent, interchangeable modules, which can simplify debugging and enhance system reliability. Additionally, the use of interrupt-driven programming can greatly improve the efficiency of embedded systems, allowing them to respond to real-time events with minimal CPU usage. These concepts, when applied thoughtfully, can lead to the creation of robust and efficient embedded systems that are well-suited to the demands of modern technology.