International Journal of Research and Review
Volume 11; Issue: 11; November 2024
Website: www.ijrrjournal.com
E-ISSN: 2349-9788; P-ISSN: 2454-2237

Review Paper

Arduino-Based Automatic School Bell System: A Simple and Cost-Effective Solution

Siddhartha Subedi[1], Ajay Raj Chalise[2]

[1]Department of Computer Science, Shree Amarsingh Secondary School, Pokhara, Nepal.
[2]Department of Computer Science, School of Engineering, Pokhara University, Pokhara, Nepal.

Corresponding Author: Siddhartha Subedi

DOI: https://doi.org/10.52403/ijrr.20241127

ABSTRACT

In schools, the bell plays a crucial role in maintaining the schedule. Ringing the bell manually is often ineffective due to issues with accuracy. Therefore, an automatic school bell system is needed to save the workforce and ensure precision. This project utilizes an Arduino UNO microcontroller to read the time from a Real-time clock (RTC DS3231) and display it on a 16 × 2 LCD display. At specific times, the Arduino activates a 5-volt relay for a predetermined duration to ring the electric bell. An alternative switch for the electric bell is included for use during examinations and emergencies. The bell is powered by direct AC and controlled by the relay module. The Arduino UNO is powered using a 5-volt output mobile charging adapter. Thus, the Arduino-based automatic school bell system is easy to design, develop, cost-effective, and operates at predefined times.

Keywords: Automatic School Bell System, Arduino UNO, Accuracy, DS3231, LCD, Relay.

INTRODUCTION

Time is a crucial aspect of students' lives, as it directly impacts their learning and development. Therefore, it is essential to utilize time efficiently and ensure that all activities are performed timely and accurately. However, many educational institutions still rely on manually operated traditional school bell systems, which often suffer from significant inaccuracies and inefficiencies. To eradicate this problem and save manpower, we should use an automatic school bell system. Such a system can ensure precise timing, reduce human error, and improve the overall efficiency of school operations.
The automatic school bell is a digital circuit programmed to ring at specific times every day using an Arduino UNO, which is an easy-to-program and low-cost microcontroller compared to others. The RTC module is connected to the microcontroller using jumper wires to provide accurate time, which is then displayed on a 16 × 2 LCD display. The main objective of automation is to save labor and provide a solution that is much better, more accurate, consistent, and useful compared to traditional bell systems.

MATERIALS & METHODS

System Overview

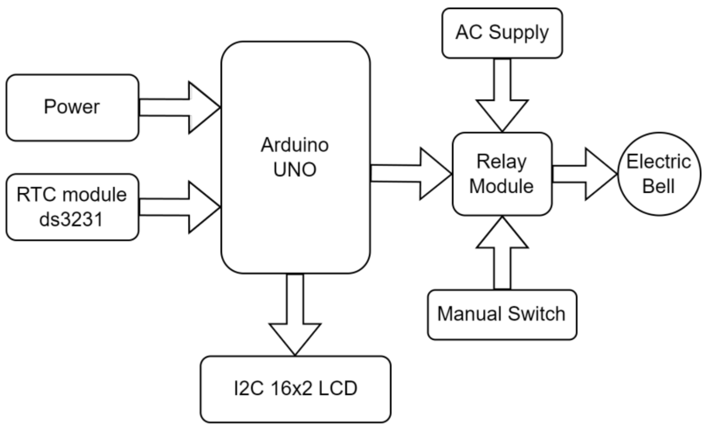


Figure 1: System Block Diagram of Automatic School Bell System

The system block diagram of the automatic school-bell system is shown in Figure 1. The Arduino UNO microcontroller takes real-time data from the RTC DS3231 module and receives 5V of DC power as input. After processing these data, the Arduino outputs signals to both the relay module and 16 × 2 LCD display. The relay module was also connected to a manual switch and an external power supply for the electric bell. When the Arduino sends a 5V signal to the relay, the circuit is closed and the bell rings.

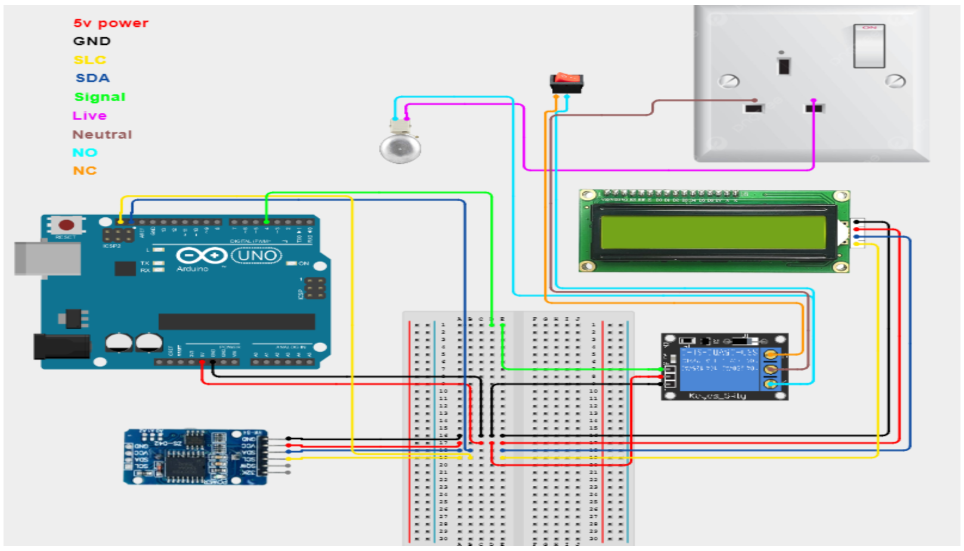


Figure 2: Circuit Diagram for Automatic School Bell System

Arduino

The Arduino UNO serves as the processing unit in this project, featuring the ATmega328 microcontroller. "UNO," meaning "one" in Italian, signifies the release of Arduino Software (IDE) version 1.0. As the first USB-compatible board, the Arduino UNO has become the reference model for the entire Arduino platform. Operating at 5V, it included 32 KB of flash memory, 2 KB of SRAM, and 1 KB of EEPROM. The board also provides 14 digital input/output pins (6 of which support PWM outputs) and 6 analog inputs. It maintains a USB connection for both programming and power, thus facilitating easy integration and development.

To power the Arduino, a transformer was used to step down the 240V AC to a lower voltage, which was then converted to 5V DC using bridge rectification. This 5V DC supply is connected to the Arduino's 5V pin and GND through a breadboard, ensuring a stable power connection, and the SCL and SDA pins are connected to the breadboard, which in turn connects to a 16 × 2 LCD with an I2C module and the RTC DS3231, as shown in Figure 2. Additionally, digital pin 4 was utilized as the output, linking it to the signal pin of the relay module.

The Arduino UNO receives real-time data from the RTC DS3231 module and displays this information on an LCD. It is programmed to compare the current time with the pre-set schedule; when a match is detected, the Arduino sends a 5V signal to the relay for a specific duration, activating the connected bell.

Figure 3: Arduino

DS3231 RTC

The DS3231 is a highly accurate, low-cost I2C real-time clock with an integrated temperature-compensated crystal oscillator (TCXO) and crystal. The Real-Time Clock (RTC) tracks seconds, minutes, hours, dates, months, days of the week, and years, with leap-year adjustments accurate through 2100.The DS3231 Real-Time Clock (RTC) module operates at a low voltage of 3.3V. The DS3231 Real-Time Clock (RTC) module also supports an external coin cell battery (like a CR2032), ensuring that timekeeping continues even when the main power supply is unavailable. It typically consists of six pins: VCC for power, GND for ground, SDA and SCL for I2C communication, an optional 32.768 kHz output pin (32K), and a square wave output pin (SQW). The VCC, GND, SDA, and SCL pins are connected from the breadboard to the DS3231 module, ensuring proper power and communication links.

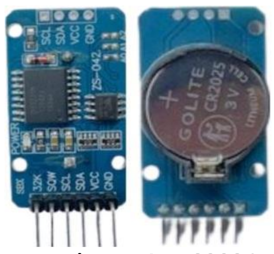


Figure 4: DS3231

I2C 16x2 Liquid Crystal Display

The 16x2 character LCD, which is attached to an I2C module, features two rows for displaying text, with each row capable of showing 16 characters. It operates at 5V, and the display contrast can be manually adjusted using a potentiometer. Although I2C communication simplifies wiring, only four pins are required to connect the LCD: VCC, GND, SDA, and SCL, all sourced from the breadboard. This screen displays the current date and time provided by the RTC module.

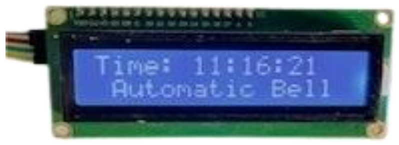


Figure 5: 16x2 LCD

Relay Module

The relay is an electromagnetic component that functions as an automatic switch in automation circuits. This allows the control of high-current loads using a low-current signal and supports both AC and DC. The relay module typically consists of six pins: three for the input (VCC, GND, and Signal) and three for the output (NO, COM, and NC).

To power the relay module, the VCC and GND pins were connected to a 5V supply and ground through the breadboard. The Signal pin is used to control the relay; it can operate in either the active high or active low mode. In this project, the signal pin was connected to digital pin 4 on the Arduino through the breadboard. The relay we are using operates in active low mode, meaning that the signal pin continuously receives 5V from the Arduino. When it is time to ring the bell, the 5V signal is interrupted, activating the relay. The opposite occurs for an active high relay.

Regarding the output pins, the Normally Open (NO) pin remains open unless a signal is sent to the relay module that closes the circuit. The Common (COM) pin is the central connection point in the relay that connects the load (e.g., electric bell) to either normally open (NO) or Normally Closed (NC), depending on whether the relay is activated or not. The Normally Closed (NC) pin, on the other hand, maintains a closed circuit with the COM pin by default and disconnects when the relay receives a signal, breaking the circuit.

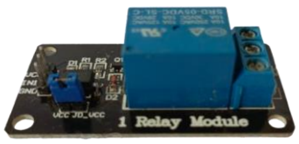


Figure 6: 5V Relay

Breadboard

A breadboard is a rectangular board with small holes that simplify wiring and connections during circuit construction. It is reusable and does not require soldering.

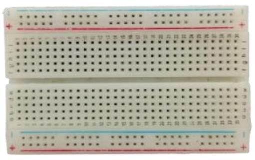


Figure 7: Breadboard

Jumper Wires

Jumper wires were used for the entire connection in this project. These wires are essential for connecting components such as DS3231, LCD, and relays. These are of various types, including male-to-male, male-to-female, female-to-female, and U-shaped configurations.

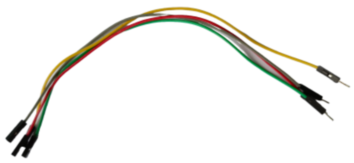


Figure 8: Jumper Wire

Adapter

Jumper wires were used for the entire connection in this project. These wires are essential for connecting components such as DS3231, LCD, and relays. These are of various types, including male-to-male, male-to-female, female-to-female, and U-shaped configurations.

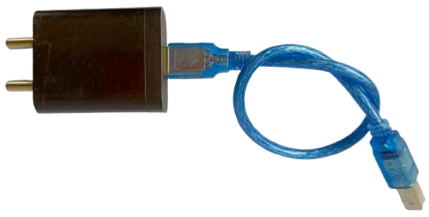


Figure 9: Adapter with USB Cable

Electric Gong Bell

The Electric Gong bell operates directly on a 220V 50 Hz AC supply, simplifying the automation process. The positive terminal of the bell connects to the common (COM) pin of the relay, whereas the negative terminal is connected directly to the ground. The Normally Open (NO) pin of the relay was connected to the positive terminal of the power supply. When the relay is activated by the Arduino, the circuit between the COM and NO pins closes, allowing current to flow and causing the bell to ring.

Manual Switch

The switch is used to ring the bell in emergency situations, during exams, or for programs in which the time is not scheduled in the Arduino. One terminal of the switch was connected to the NO pin, and the other terminal was connected to the COM pin. When the switch is pressed, the circuit closes, allowing the bell to ring even without Arduino activation.

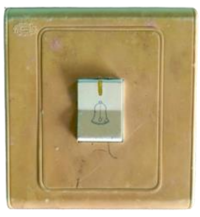


Figure 10: Manual Switch

Arduino IDE

The Arduino IDE is an open-source software platform used for writing and uploading code to any Arduino board. It simplifies the coding using a simplified version of C++. The software includes a Serial Monitor tool for debugging, testing, and direct communication with the Arduino board during the project development. In addition, the IDE provides built-in libraries and supports the addition of external libraries to various modules and sensors.

Library Used

Libraries simplify project development by providing prewritten functions and codes, making interactions with modules and sensors more convenient. The following libraries were used in this project:

- **DS3231 Library:** Developed by Henning Karlsen, this library simplifies the process of setting and retrieving real-time data from the DS3231 RTC module. It provides easy-to-use functions for managing time and date. This library can be found in RinkyDink Electronics.
- **LiquidCrystal-I2C Library:** This library facilitates communication between the Arduino and the I2C-based LCD display, is used to show the time and other messages, and is available on GitHub.

Setting Time in the RTC

To set the time in the DS3231 RTC module, the following steps:

Step 1: The DS3231 library was downloaded and installed on the Arduino IDE.

Step 2: Navigate to File > Examples > DS3231 >Arduino> DS3231_Serial_Easy.

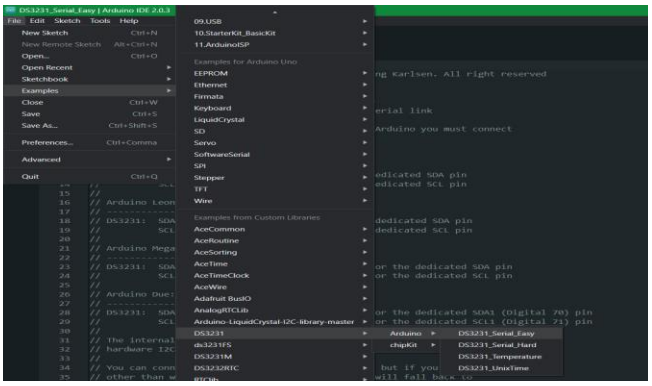
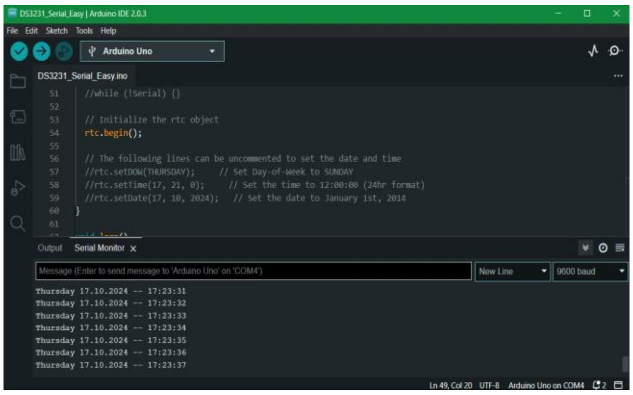


Figure 11: Navigation to set time in DS3231

Step 3: Open the DS3231_Serial_Easy example code. Uncomment the lines where the actual day, time, and date need to be set, and the correct values are input in the specified format.

Step 4: The modified code is uploaded to Arduino.


DS3231_Serial_Easy | Arduino IDE 2.0.3
File Edit Sketch Tools Help
Arduino Uno
DS3231_Serial_Easy.ino
Output
Serial Monitor
New Line
9600 baud
Ln 49, Col 20 UTF-8 Arduino Uno on COM4

```
  //while (!Serial) {}

  // Initialize the rtc object
  rtc.begin();

  // The following lines can be uncommented to set the date and time
  //rtc.setDOW(THURSDAY);     // Set Day-of-Week to SUNDAY
  //rtc.setTime(17, 21, 0);     // Set the time to 12:00:00 (24hr format)
  //rtc.setDate(17, 10, 2024);   // Set the date to January 1st, 2014
}
```

Message (Enter to send message to 'Arduino Uno' on 'COM4')

```
Thursday 17.10.2024 -- 17:23:31
Thursday 17.10.2024 -- 17:23:32
Thursday 17.10.2024 -- 17:23:33
Thursday 17.10.2024 -- 17:23:34
Thursday 17.10.2024 -- 17:23:35
Thursday 17.10.2024 -- 17:23:36
Thursday 17.10.2024 -- 17:23:37
```

Figure 12: Setting and Uploading time, date and day with Serial Monitor output.

Step 5: After uploading, comment on the time-setting lines again and reupload the code. The current date and time will be displayed on a Serial Monitor.
Step 6: Even if the Arduino is powered off, the RTC will continue to keep track of time because of its onboard battery. The RTC is ready for use in this project.

Project Source Codes
To implement all the functionalities of this project, such as displaying time on the LCD and sending signals to the relay at specific times, the necessary code has been written and uploaded. The code includes fetching the time from the RTC module, comparing the current time with preset times, and sending signals to the relay to ring the bell.

The complete source code, along with detailed descriptions and explanations of each section, can be found on GitHub.

RESULT & DISCUSSION
The bell was programmed to ring according to the timetable shown in Figure X, with a total of 13 rings, including short and long breaks. Each bell rings for a duration of 10 seconds. The schedule can easily be modified by updating the time settings or adjusting the bell duration. Additionally, an emergency switch is provided for ringing the bell during unscheduled events such as programs or exams. The system also displays the current time and date on the LCD screen. This automatic school-bell system eliminates the need for manual operation, ensuring precise bell ringing without delay.

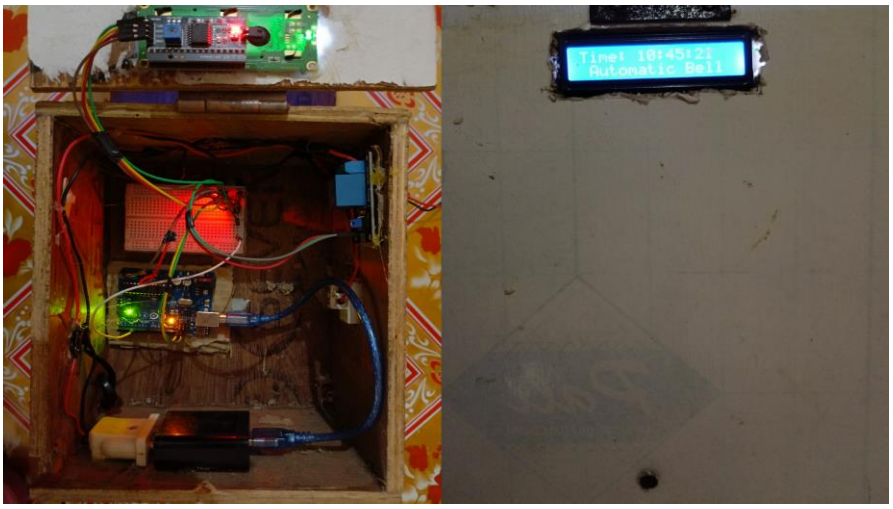


Figure 13: Final Implementation of the Arduino-Based School Bell System

CONCLUSION
During school hours, it is often frustrating when classes run beyond their scheduled time, especially when the bell rings manually. This project addresses this issue by offering an affordable, simple, and efficient automatic school-bell system that rings accurately on time, ensuring no unnecessary delays or extensions. This system helps improve the overall punctuality of school activities.

Declaration by Authors
Acknowledgement: None
Source of Funding: None
Conflict of Interest: The authors declare no conflict of interest.

REFERENCES
1. Mahmood RZ. High Accurate Automatic School Bell Controller Based On ESP-32 Wi-Fi.
2. Dinda RA, Sadrina S, Mursyidin M. The High Accurate Automatic School Bell Controller Based On Arduino Uno DS1307

I2C Real-time Clock. Jurnal Teknik Mesin Mechanical Xplore. 2023 Jul 24;4(1):17-26.

3. Reddy AV, Sharmila V, Chandrasekhar S. Timer Based Automatic College Bell System.
4. Ayson R. Microcontroller-based school bell system for the College of Engineering, DMMMSU-MLUC. DMMMSU Research and Extension Journal. 2021 Dec 1;5:47-66.
5. Madakam S, Ramaswamy R, Tripathi S. Internet of Things (IoT): A literature review. Journal of Computer and Communications. 2015 May 25;3(5):164-73.
6. Khanna A, Kaur S. Internet of things (IoT), applications and challenges: a comprehensive review. Wireless Personal Communications. 2020 Sep; 114:1687-762.
7. Shah DK, Gunjal N, Shinde D, Mandave S. Automatic Wireless College Bell System Using Zigbee. Journal of Emerging Technologies and Innovative Research. 2019 May;6(5):525-529.
8. http://www.rinkydinkelectronics.com/
9. https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library
10. https://github.com/siddharthayas/Automatic-School-Bell-System-Using-Arduino

How to cite this article: Siddhartha Subedi, Ajay Raj Chalise. Arduino-based automatic school bell system: a simple and cost-effective solution. *International Journal of Research and Review*. 2024; 11(11): 314-320. DOI: *https://doi.org/10.52403/ijrr.20241127*

******