Automatic School Bell by using Arduino UNO

Khin Thet Mar
Faculty of Computer System and Technology
University of Computer Studies (Lashio)
khinthetmar23@gmail.com, moenaychikhin@gmail.com

Abstract

This paper attempts to achieve automatic school bell by using an Arduino UNO as a main processor. Automatic School Bell has been designed and constructed based on Real time clock ***RTC IC DS1307*** *which works on I2C protocol. The design is in four modules; power supply, RTC IC DS1307, LCD device and Arduino UNO modules. While the Arduino UNO forms the main control element, Real time clock means it runs even after power failure. When power is reconnected, it displays the real time respective to the time and duration it was in off state.16x2 LCD module to display the time in -(hour, minute, seconds, date, month and year) format. An Alarm option is also added and we can set up the alarm time. Once alarm time it saved in internal EEPROM of arduino, it remains saved even after reset or electricity failure. Real time clocks are commonly used in our computers, houses, offices and electronics device for keeping them updated with real time.*

Keywords*:* Arduino UNO, DS1307, LCD, Relay, Buzzer

1. Introduction

In today's life, everyone gives importance to time. Time does not wait for anybody. Everything should be performed in time and accurately. Now a day's school or college bells are manually operated. Hence there is a big question of accuracy. Also there is necessity of manpower and money. Hence here we should use automatic control system, which saves our manpower and money and also gives highest accuracy.

A bell is a percussion instrument used in schools and colleges that indicates the students when it is time to go to the class in the morning and when it is time to change classes during the day. No other instrument can do such a work. So it is an important instrument in both primary and secondary schools and even in the industries and other businesses where the bell timer plays a critical role throughout the day.

Present bells for periods in schools are operated manually. This design takes over the task of ringing the bell in schools as the bell would ring automatically at the schedule time. It has a Real Time Clock (DS1307) which tracks over the real time. When this time equals to the bell ringing time, then the relay for the bell is switched on. The real time clock is displayed on LCD display.

2. Background Theory

School Bell finds a tremendous use at primary and secondary school levels as well as in colleges where the teaching sections can span over eight periods including breaks. The advantage of this design is that the bell rings at the start of each period without any human intervention to a great degree of accuracy and hence takes over the manual task of switching on/off the college bell with respect to time. It uses Real Time Clock (DS1307) which tracks the real time.

The scheduled time results are compared with that of a clock, however, some drift is noticed, which is negligible. The microcontroller AT328 is used to control all the functions, it gets the time through the switch and stores it in its memory. When this programmed time equals the real time then the bell is switched on via a relay for a predetermined time. The bell ringing time can be edited at any time, so that it can be reused again and again at normal class timings as well as at exam times. For this a microcontroller has to be programmed using the C language or assembly language for controlling the circuit.

3. The Proposed Design

Automatic school bell was designed and constructed using Arduino microcontroller device. The block diagram for the School Bell is as shown in Figure 1. The design is in four modules; Buzzer, RTC chip, LCD device and Arduino UNO modules. Here arduino is used for reading time from DS1307 and display it on 16x2 LCD. DS1307 sends time/date using 2 lines to arduino.

A buzzer is also used for alarm indication, which beeps when alarm is activated. The Real time clock module keeps the track of the time even after long power-cut. A 5V relay is provided for switching the bell on and off. A block diagram is shown below to understand the working of this Real Time Clock.

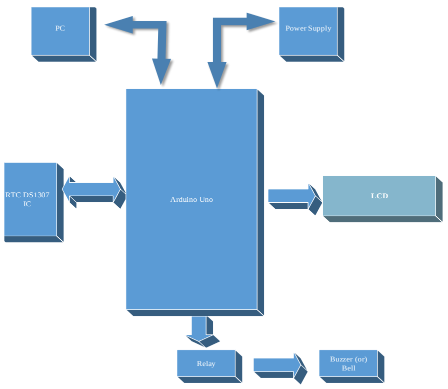


Figure 1. System Block Diagram of Automatic School Bell

4. Hardware Implementation Process

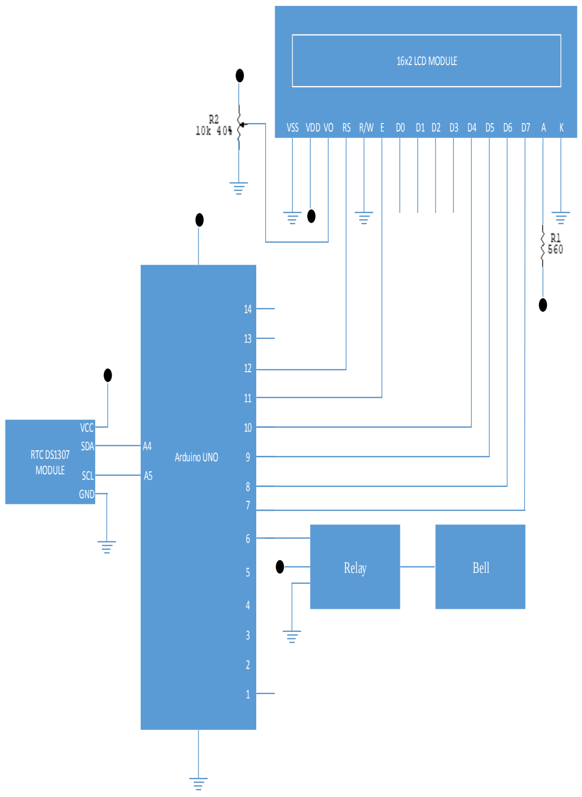
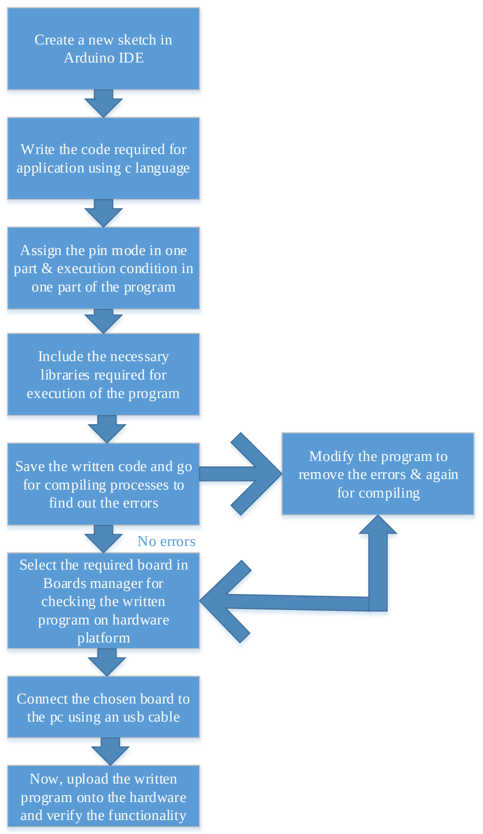


Figure 2. Overall Circuit Diagram

A4 and A5 pin of Arduino UNO is configured by input pin and 6, 7, 8, 9, 10, and 11 are configured as digital output to send data to 16x2 character dot matrix LCD (Liquid Crystal Display). Pin 6 configured relay to operate buzzer (or) bell.

4.1. Flow Chart


Create a new sketch in Arduino IDE
Write the code required for application using c language
Assign the pin mode in one part & execution condition in one part of the program
Include the necessary libraries required for execution of the program
Save the written code and go for compiling processes to find out the errors
Modify the program to remove the errors & again for compiling
No errors
Select the required board in Boards manager for checking the written program on hardware platform
Connect the chosen board to the pc using an usb cable
Now, upload the written program onto the hardware and verify the functionality

4.2. Arduino UNO

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

Uno means "One" in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

Arduino UNO is used as a main processing unit. A4 and A5 of Arduino UNO connected SCL and SDATA of DS1307. Pin 5, 4, 3, 2, 11 & 12 of Arduino are configured as digital output pins to send data to 16x2 character dot matrix LCD (Liquid Crystal Display).

.

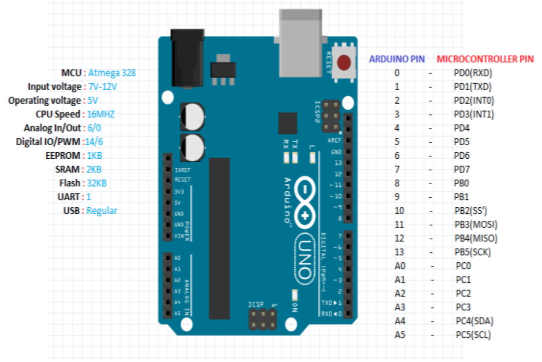


Figure 3. Arduino UNO

4.3. Liquid Crystal Display (LCD)

The 2 line 16 characters LCD, is used to display current date and time. It is also used to display current date and time to see easily.

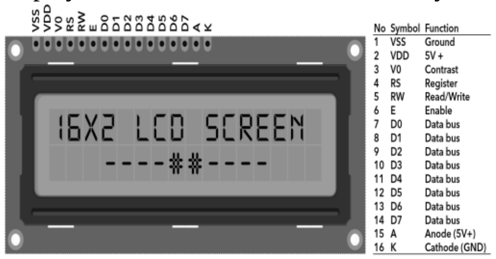


Figure 4. Liquid Crystal Display (LCD)

The LCD is configured as 4-bit data interface D4 to D7 pins are used for data inputs. R/W (5) pin and VSS (1) pin of LCD is pulled to ground. The E (6) pin of LCD is controlled by Arduino to store the data. The RS (4) pin of LCD is controlled by Arduino to select whether the input data is command or data. V0 (3) pin is used to adjust LCD brightness by using variable resistor, R2.

The LCD employed is a 16 x 2 type capable of displaying 32 characters in alphanumeric form. It has a wide range of LCD driver power from - 3 to 1V with high speed MPU bus interface of 2MHZ when the supply voltage is Vcc = 5V. It can also be configured as 4 bit or 8 bit interface enabled to transmit or receive data in either 4 bits or 8 bits. It consumes very small power with automatic reset circuit that initializes the controller/driver after power on. Internally there is an oscillator that has external resistors (LCD Data book).

The LCD was configured to drive its dot - matrix under the control of 4-bit output of the microcontroller. A regulated supply of 5V was used to supply the chip which is within the recommended supply voltage of the chip. A 560Ω resistor was included as a current limiting resistor. The pin 16 of the chip is the K (16), VSS (1) and R/W (5), are the ground and was connected to the 0 line of the supply.

While pin 11 to pin 14 were connected to receive the 4 - bit data from the main microcontroller. A variable resistor is provided to adjust the brilliance of the LCD. The value as recommended in the datasheet is from 10k to 30k. For this project, a 10k variable resistor was used to vary the brightness of the LCD. Pin 4 is the reset pin that is used to clear the registers of the LCD.

4.3. RTC DS1307

DS1307 is the frequently used real time clock (RTC) IC for clock and calendar. The clock function provides seconds, minutes and hours while the calendar function provides day, date, month and year values.

The clock can operate in either 12 hour with AM/PM indication or 24 hour format. A 3V backup battery must be connected to the RTC so that the IC can automatically switch to the backup supply in case of power failure. A 32.768 KHz crystal is connected to the oscillator terminal of DS1307 for 1 Hz oscillations.

A Real Time Clock or RTC is a battery powered clock that measures time even when there is no external power, or the microcontroller is reprogrammed. An RTC displays clock and calendar with all timekeeping functions. The battery, which is connected to the RTC is a separate one and is not related or connected to the main power supply. When the power is restored, RTC displays the real time irrespective of the duration for which the power is off. Such Real Time Clocks are commonly found in computers and are often referred to as just CMOS (Complementary Metal Oxide Semiconductor).

Most microcontrollers and microprocessors have built in timers for keeping time. But they work only when the microcontroller is connected to power supply. When the power is turned on, the internal timers reset to 0. Hence, a separate RTC chip is included in applications like data loggers for example, which doesn't reset to 0 when the power is turned off or reset.

Real Time Clocks are often useful in data logging applications, time stamps, alarms, timers, clock builds etc. In this project, a Real Time Clock, which displays accurate time and date along with an alarm feature is designed.

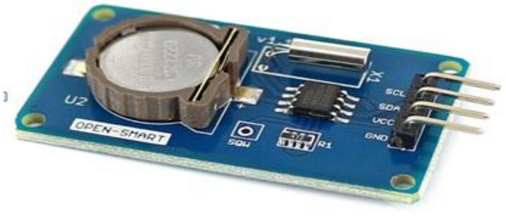


Figure 5. RTC DS1307

The circuitry involved in the design of an automatic college bell. Here, we are making use of ARDUINO UNO board for dumping

5. Software Procedure for Controlling the Whole System Function

the code written in ARDUINO IDE 1.8.5 software using Python coding and then we can check the required output of bell by interfacing it to the ARDUINO UNO board. The heart of the circuit is the ATMEGA microcontroller. The microcontroller we have used is ATMEGA328 which is the master device.

The slave device is the RTC IC DS1307, which automatically counts every second, once enabled. The intervals of time after which the bell should ring is already programmed and loaded into the microcontroller. Once the time that is fixed matches with the time in the RTC clock, the bell rings. The bell rings continuously for a fixed time (50 seconds in our implementation) which is also mentioned at the time of programming.

The circuit is implemented by interface the DS1307 with the ATMEGA328 microcontroller. It is through this serial interface that the exact time is read into the ATMEGA328 microcontroller and is compared against the set of time in the code. If the present time matches with the time that is set in the program, that is when the bell should ring, logic HIGH is driven to the output port of the microcontroller.

This small voltage (5V) acts as the enable to the relay circuit, which turns on the 230V to the bell and the bell rings. Another part of the system is the time display. The time value read into the microcontroller from RTC is also given as output through its port pins every instant to be displayed, along with comparing the values internally. The output value from the microcontroller pins are displayed in the 16X2 LCD display, which gets automatically updated every minute. In the application of the automatic bell that we have used, the microcontroller is configured as the master device. Microcontroller serially communicates with the RTC (DS1307), which is the slave device.

6. Overall Discussion

The project is an Arduino based automatic bell system which can be configured for every class of the school. It is assume that the school has seven periods organized in a day for different subjects and have lunch break in between. The break occurs after third time.

The program has been written in such a fashion that, the bell should ring simultaneously for every 50 minutes along with the display of the date and time on the LCD screen indicating the completion of a particular session and beginning of another session exactly at that instant of time at which the bell rings continuously for 50 seconds from the movement it is activated. Here, in our implementation, the bell rings at 9 instants of time in a day's schedule which is according to the program we have assigned. Manually operate the bell during some situation push button are provided. The project allows to set duration for each period and assign subject from a list of subjects to each period. The user can set time-table for five days of week from Monday to Friday. The user can also set the duration of both periods.

The time instants at which the bell rings in a day's schedule at UCS (Lashio) for 2017/2018 Academic Year as follows:

Table 1. Time Schedule for 2017/2018 Academic Year

TIME	**DATA DISPLAYED**	**STAT OF THE BELL**
8.45	DISPLAYS DATE	OF STATE (INACTIVE)
9:00	FIRST HOUR BEGINS	CTIVAT (From 9:00:00 to 9:00:50)
9:49	DISPLA DATE AN TIM	INACTIVE
9:50	FIRST HOU COMPLETED	ACTIVATED (From 9:50:00 to 9:50:50)
10:45	SECO HO COMPLETED	ACTIVATE

(From 10:45:00 to 10:45:50)
11:40 THIRD COMPLETED ACTIVATE
(From 11:40:00 to 11:40:50)
12:25 LUNCH HO STARTED ACTIVATE
(From 12:25:00 to 12:25:50)
13:15 LUNCH HO COMPLETED ACTIVATE
(From 13:15:00 to 13:15:50)
14:10 FIFTH HOU COMPLETED ACTIVATED
(From 14:10:00 to 14:10:50)
15:05 SIXTH HO COMPLETED ACTIVATE
(From 15:05:00 to 15:05:50)
16:0 END TODAY
(From 16:00:00 to 16:00:50)

6.1. Result and Discussion

I tested both **RTC DS1307** and **DS3231**. There are two main differences between the ICs on the real-time clock modules, which is the accuracy of the time-keeping. The **DS1307** used in the first module works very well, however the external temperature can effect the frequency of the oscillator circuit which drives the DS1307's internal counter. There is a problem, however will usually result with the clock being off by around five or so minutes per month.

The **DS3231** is much more accurate, as it has an internal oscillator which isn't affected by external factor. Along with keeping track of the time and date, these modules also have a small EEPROM, an alarm function.

6.2. Conclusion and Future Work

Using an Arduino simplifies the amount of Hardware and software development you need to do in order to get a system running. On the software side, Arduino provides a number of libraries to make programming the microcontroller easier. The simplest of these are functions to control and read the I/O pins rather than having to fiddle with the bus/bit masks normally used to interface with the Atmega I/O (This is a fairly minor inconvenience). If you operate manually the bell the push buttons are provided for manually operating the bell during user define time.

More useful are things such as being able to set I/O pins to PWM at a certain duty cycle using a single command or doing Serial communication. Personally, I think the greatest advantage is having the hardware platform set up already, especially the fact that it allows programming and serial communication over USB. This saves me the trouble of having to do my own PCB (which can cost more than an Arduino) or bread boarding (which most people won't like doing).

References

[1] https://store.arduino.cc/usa/arduino-uno-rev3
[2]https://classes.ucs.edu/ee459/library/datasheets/DS1307.pdf
[3]https://www.sparkfun.com/datasheets/LCD/ADM1602K-NSW-FBS-3.3v.pdf
[3] http://www.rinkydinkelectronics.com/library.php
[4]https://www.element14.com/community/groups/embedded/blog/2017/01/18/new-proteus-libraries-for-engineering-students
[5]https://www.elprocus.com/different-types-of-arduino-boards/
[6] https://github.com/rodan/ds3231
[7]https://datasheets.maximintegrated.com/en/ds/DS3231-DS3231S.pdf