Component	Connected to	Arduino pins/Breadboard
LCD	GND	GND
LCD	VCC	5V
LCD	SDA	A4
LCD	SCL	A5
BUTTON 1	Upper Right	GND
BUTTON 1	Upper left	Pin 2
BUTTON 2	Upper Right	GND
BUTTON 2	Upper left	Pin 3
Piezo Buzzer	+	Pin 4
Piezo Buzzer	-	GND
DS1302 RTC Module	VCC	5V
DS1302 RTC Module	GND	GND
DS1302 RTC Module	CLK	Pin 5
DS1302 RTC Module	DAT	Pin 6
DS1302 RTC Module	RST	Pin 7
SD Card Module	CS	Pin 10
SD Card Module	SCK	Pin 13
SD Card Module	MOSI	Pin 11
SD Card Module	MISO	Pin 12
SD Card Module	VCC	5V
SD Card Module	GND	GND

RAW CODE:

```
/////
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <RtcDS1302.h>
#include <ThreeWire.h>
#include <TimeLib.h>
#include <SPI.h>
#include <SD.h>

// Pin Assignments
const int buttonPin = 2;
const int resetAllPin = 3;
const int buzzerPin = 4;
const int chipSelect = 10;  // SD card CS pin (change to match your SD card module)

// I2C LCD Setup (address, columns, rows)
LiquidCrystal_I2C lcd(0x27, 16, 4);

// RTC DS1302 Pins
ThreeWire myWire(6, 5, 7);
RtcDS1302<ThreeWire> rtc(myWire);

// Variables
int count = 0;
char lastPressTime[9] = "--:--:--";
unsigned long lastUpdate = 0;
bool displayData = false;
char dateStr[11];
char timeStr[9];
File logFile;
bool sdCardAvailable = false;
static unsigned long displayDataStartTime = 0;  // NEW: timer to track display timeout

// Custom function for HD44780 LCD address mapping
void lcdSetCursorRaw(uint8_t col, uint8_t row) {
  const uint8_t row_map[] = {0x00, 0x40, 0x10, 0x50}; // HD44780 layout for 4x16
```

```
  lcd.command(0x80 | (col + row_map[row]));
}

void setup() {
  // Initialize Serial for debugging
  Serial.begin(9600);

  // Initialize LCD
  lcd.init();
  lcd.backlight();

  // Initialize pins
  pinMode(buttonPin, INPUT_PULLUP);
  pinMode(resetAllPin, INPUT_PULLUP);
  pinMode(buzzerPin, OUTPUT);

  // Initialize RTC
  rtc.Begin();
  if (!rtc.IsDateTimeValid()) {
    rtc.SetDateTime(RtcDateTime(__DATE__, __TIME__));
  }

  // Initialize SD card
  lcd.clear();
  lcdSetCursorRaw(0, 1);
  lcd.print("  Initializing");
  lcdSetCursorRaw(0, 2);
  lcd.print("   SD Card....");
  delay(2000);

  if (!SD.begin(chipSelect)) {
    lcd.clear();
    lcdSetCursorRaw(0, 1);
    lcd.print(" SD Card Failed");
    lcdSetCursorRaw(0, 2);
    lcd.print(" Continuing w/o");
    lcdSetCursorRaw(0, 3);
    lcd.print("    SD CARD.");
```

```
    delay(7000);

    sdCardAvailable = false;
  } else {
    lcd.clear();
    lcdSetCursorRaw(0, 1);
    lcd.print(" SD Card Ready.");
    delay(2000);
    sdCardAvailable = true;

    // Write header to log file if it doesn't exist
    if (!SD.exists("Doorbell.txt")) {
      logFile = SD.open("Doorbell.txt", FILE_WRITE);
      if (logFile) {
        logFile.println("Date   ||   Time Press   ||   PRESS Count   ||   Last Press Time");
        logFile.close();
      }
    }
  }

  lcd.clear();
  lcdSetCursorRaw(0, 1);
  lcd.print("      Ring");
  lcdSetCursorRaw(0, 2);
  lcd.print("    Doorbell");
}

void loop() {
  unsigned long nowMillis = millis();

  // Update only if button was pressed before
  if (displayData && (nowMillis - lastUpdate >= 1000)) {
    lastUpdate = nowMillis;
    RtcDateTime now = rtc.GetDateTime();

    // Format date and time strings
    sprintf(dateStr, "%04u-%02u-%02u", now.Year(), now.Month(), now.Day());
    sprintf(timeStr, "%02u:%02u:%02u", now.Hour(), now.Minute(), now.Second());
```

```
  lcdSetCursorRaw(0, 0);
  lcd.print("Date: ");
  lcd.print(dateStr);

  lcdSetCursorRaw(0, 1);
  lcd.print("Time: ");
  lcd.print(timeStr);

  lcdSetCursorRaw(0, 2);
  lcd.print("Press Count: ");
  lcd.print(count);
  if (count < 10) lcd.print("  "); // clear extra chars

  lcdSetCursorRaw(0, 3);
  lcd.print("Record: ");
  lcd.print(lastPressTime);

  // Check if 20 seconds passed since last button press
  if (nowMillis - displayDataStartTime >= 20000) { // 20 seconds = 20000 ms
    displayData = false;  // Stop showing detailed data
    lcd.clear();
    lcdSetCursorRaw(0, 1);
    lcd.print("      Ring");
    lcdSetCursorRaw(0, 2);
    lcd.print("    Doorbell");
  }
}

// Button press handling
if (digitalRead(buttonPin) == LOW) {
  delay(50); // debounce
  if (digitalRead(buttonPin) == LOW) {
    count++;
    RtcDateTime now = rtc.GetDateTime();
    sprintf(lastPressTime, "%02u:%02u:%02u", now.Hour(), now.Minute(), now.Second());
    sprintf(dateStr, "%04u-%02u-%02u", now.Year(), now.Month(), now.Day());
    sprintf(timeStr, "%02u:%02u:%02u", now.Hour(), now.Minute(), now.Second());
```

```
    // Save to SD card
    if (sdCardAvailable) {
      saveToSD(dateStr, timeStr, count, lastPressTime);
    }

    playDingDong();
    displayData = true;          // Activate full display
    lastUpdate = 0;              // Force immediate update
    displayDataStartTime = millis(); // Start 20-sec timeout timer
    delay(250); // debounce delay
  }
 }

 // Reset handling
 if (digitalRead(resetAllPin) == LOW) {
  delay(50); // debounce
  if (digitalRead(resetAllPin) == LOW) {
    count = 0;
    displayData = false;
    strcpy(lastPressTime, "--:--:--");
    lcd.clear();
    lcdSetCursorRaw(0, 1);
    lcd.print("  Resetting...");
    delay(2000);
    lcd.clear();
    lcdSetCursorRaw(0, 1);
    lcd.print("      Ring");
    lcdSetCursorRaw(0, 2);
    lcd.print("    Doorbell");
  }
 }
}

// Function to save display data to SD card
void saveToSD(const char* date, const char* time, int pressCount, const char* recordTime) {
 logFile = SD.open("Doorbell.txt", FILE_WRITE);
```

```
  if (logFile) {
    // Write data in TXT format
    logFile.print(date);
    logFile.print("   ||   ");
    logFile.print(time);
    logFile.print("   ||   ");
    logFile.print(pressCount);
    logFile.print("   ||   ");
    logFile.println(recordTime);

    logFile.close();

    // Briefly indicate logging was successful
    lcdSetCursorRaw(15, 3);
    lcd.print("*");
    delay(300);
    lcdSetCursorRaw(15, 3);
    lcd.print(" ");
  } else {
    // Indicate SD write error
    lcdSetCursorRaw(15, 3);
    lcd.print("!");
    delay(300);
    lcdSetCursorRaw(15, 3);
    lcd.print(" ");

    // Mark SD card as unavailable if write fails
    sdCardAvailable = false;
  }
}

// Function to play "Ding Dong" sound
void playDingDong() {
  // Play the "Ding" sound (short high-pitched tone)
  tone(buzzerPin, 2000, 2000);  // 1000Hz for 1000ms
  delay(1000);                  // Pause between tones
  // Play the "Dong" sound (longer, lower-pitched tone)
  tone(buzzerPin, 1000, 1500);    // 500Hz for 500ms
```

```
  delay(700);                   // Pause after sound
}

void printDate(RtcDateTime now) {
  sprintf(dateStr, "%04u-%02u-%02u", now.Year(), now.Month(), now.Day());
  lcd.print(dateStr);
}

void printTime(RtcDateTime now) {
  sprintf(timeStr, "%02u:%02u:%02u", now.Hour(), now.Minute(), now.Second());
  lcd.print(timeStr);
}
```