Learn Arduino – Real Time Clock Interfacing

A Comprehensive Guide on RTC interfacing to Arduino Board with different code examples.

Interfacing with Real-Time-Clock (RTC) V1.0

Table of Contents

Chapter-1: What is Real Time Clock (RTC) ?

A real time clock is basically just like a watch - it runs on a battery and keeps time for you even when there is a power outage! Using an RTC, you can keep track of long timelines, even if you reprogram your microcontroller or disconnect it from USB or a power plug.A real-time clock (RTC) is a computer clock, most often in the form of an integrated circuit, that keeps track of the current time.Although the term often refers to the devices in personal computers, servers and embedded systems, RTCs are present in almost any electronic device which needs to keep accurate time.

Most microcontrollers, including the Arduino, have a built-in timekeeper called **millis()** and there are also timers built into the chip that can keep track of longer time periods like minutes or days. So why would you want to have a separate RTC chip? Well, the biggest reason is that **millis()** only keeps track of time *since the Arduino was last powered* . That means that when the power is turned on, the millisecond timer is set back to 0. The Arduino doesn't know that it's 'Tuesday' or 'March 8th', all it can tell is 'It's been 14,000 milliseconds since I was last turned on'.
OK so what if you wanted to set the time on the Arduino? You'd have to program in the date and time and you could have it count from that point on. But if it lost power, you'd have to reset the time. Much like very cheap alarm clocks: every time they lose power they blink **12:00**
While this sort of basic timekeeping is OK for some projects, some projects such as data-loggers, clocks, etc will need to have **consistent timekeeping that doesn't reset when the Arduino battery dies or is reprogrammed**. Thus, we include a separate RTC! The RTC chip is a specialized chip that just keeps track of time. It can count leap-years and knows how many days are in a month, but it doesn't take care of Daylight Savings Time (because it changes from place to place).

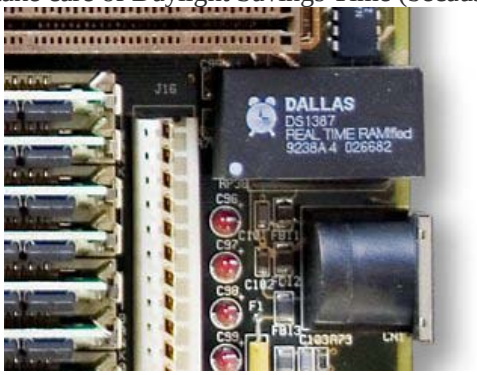


Figure 1-1: *The image above shows a computer motherboard with a Real Time Clock called the DS1387. There's a lithium battery in there which is why it's so big.*

Although keeping time can be done without an RTC, using one has benefits:

- Low power consumption (important when running from alternate power)
- Frees the main system for time-critical tasks
- Sometimes more accurate than other methods

RTCs often have an alternate source of power, so they can continue to keep time while the primary source of power is off or unavailable. This alternate source of power is normally a lithium battery in older systems, but some newer systems use a super-capacitor, because they are rechargeable and can be soldered. The alternate power source can also supply power to battery backed RAM.

Most RTCs use a crystal oscillator, but some use the power line frequency. In many cases, the oscillator's frequency is 32.768 kHz. This is the same frequency used in quartz clocks and watches, and for the same reasons, namely that the frequency is exactly 215 cycles per second, which is a convenient rate to use with simple binary counter circuits.

Chapter-2: Basic RTC Interfacing to Arduino Board

The datasheet for the DS3231 explains that this part is an "Extremely Accurate I²C-Integrated RTC/TCXO/Crystal". And, hey, it does exactly what it says on the tin! This Real Time Clock (RTC) is the most precise you can get in a small, low power package.

Most RTC's use an external 32kHz timing crystal that is used to keep time with low current draw. And that's all well and good, but those crystals have slight drift, particularly when the temperature changes (the temperature changes the oscillation frequency very veryvery slightly but it does add up!) This RTC is in a beefy package because the crystal is *inside* the chip! And right next to the integrated crystal is a temperature sensor. That sensor compensates for the frequency changes by adding or removing clock ticks so that the timekeeping stays on schedule.

This is the finest RTC you can get, and now we have it in a compact, breadboard-friendly breakout. With a coin cell plugged into the back, you can get years of precision timekeeping, even when main power is lost. Great for data logging and clocks, or anything where you need to really know the time.

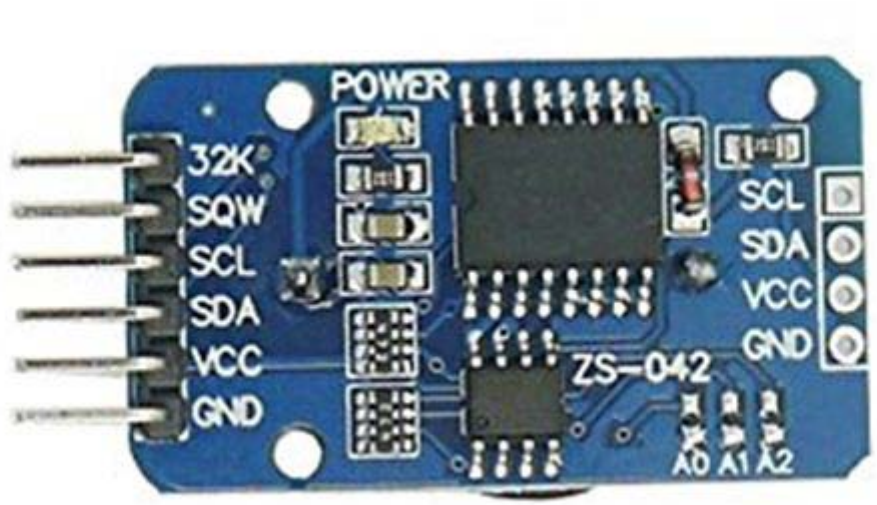


Figure 2-1: DS3231 RTC Module.

Power Pins:

- **Vcc** – this is the power pin. Since the RTC can be powered from 2.3V to 5.5V power, you do not need a regulator or level shifter for 3.3V or 5V logic/power. To power the board, give it the same power as the logic level of your microcontroller – e.g. for a 5V micro like Arduino, use 5V.
- **GND** – common ground for power and logic

I2C Logic pins:

- **SCL** – I2C clock pin, connect to your microcontrollers I2C clock line. This pin has a 10K pull-up resistor to Vcc.
- **SDA** – I2C data pin, connect to your microcontrollers I2C data line. This pin has a 10K pull-up resistor to Vcc.

Other Pins:

- **32K** – 32-KHz oscillator output. Open drain, you need to attach a pull-up to read this signal from a microcontroller pin.

- **SQW –** optional square wave or interrupt output. Open drain, you need to attach a pull-up to read this signal from a microcontroller pin.

Along with keeping track of the time and date, this modules also have a small EEPROM, an alarm function and the ability to generate a square-wave of various frequencies.

What you need for this tutorial ?

- Arduino Uno Or Arduino Mega
- DS3231 RTC breakout board
- Some jumper Wires

Connecting your module to an Arduino

This modules use the I2C bus, which makes connection very easy. First you will need to identify which pins on your Arduino or compatible boards are used for the I2C bus – these will be knows as SDA (or data) and SCL (or clock). On Arduino Uno or compatible boards, these pins are A4 and A5 for data and clock:

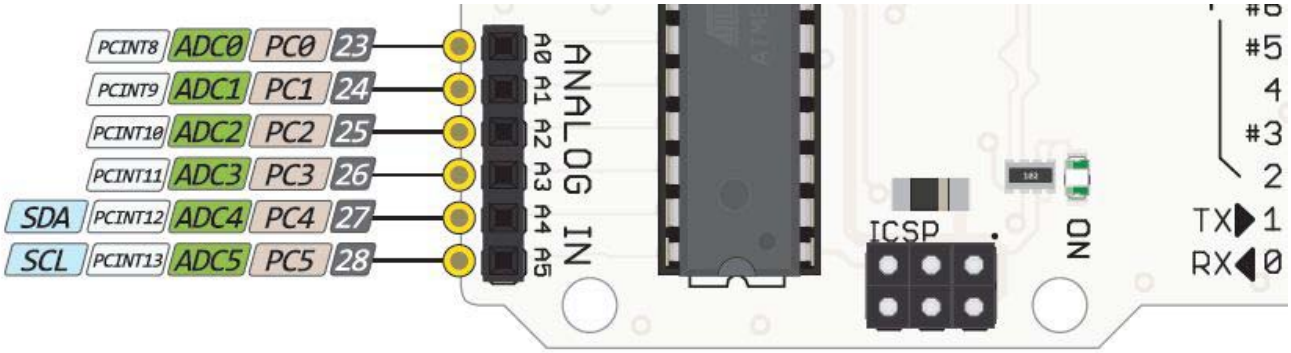


Figure 2-2: SDA & SCL pin on Arduino Uno Board

If you're using an Arduino Mega the pins are D20 and D21 for data and clock:

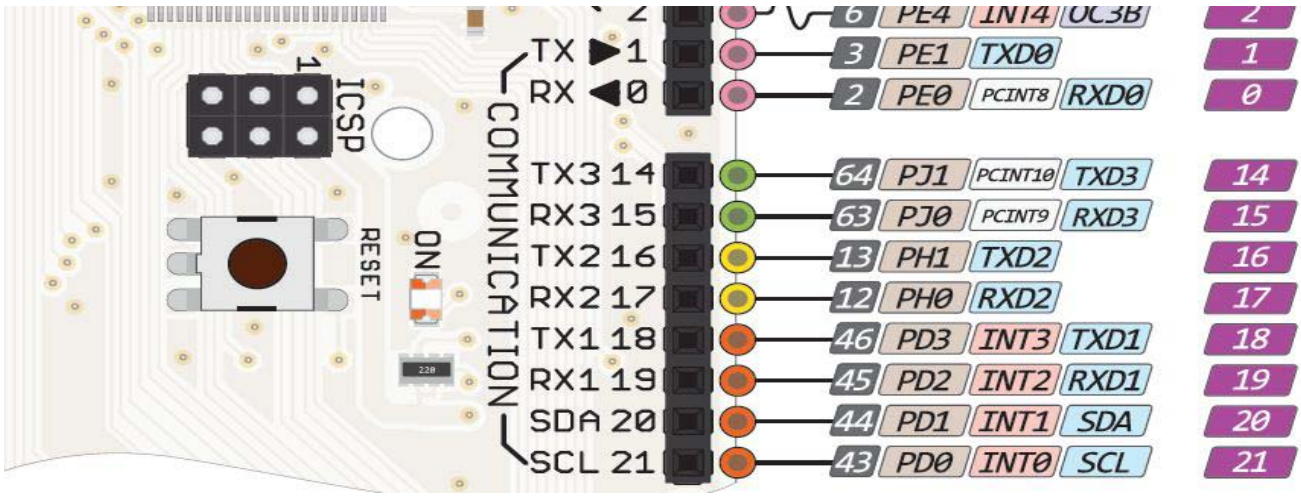


Figure 2-3: SDA & SCL pin on Arduino Mega Board

Connecting this module is easy as header pins are installed on the board at the factory. You can simply run jumper wires again from SCL and SDA to the Arduino and again from the module's Vcc and GND pins to your board's 5V or 3.3.V and GND. However these are duplicated on the other side for soldering yourown wires.
This module have the required pull-up resistors, so you don't need to add your own. Like all devices connected to the I2C bus, try and keep the length of the SDA and SCL wires to a minimum.

Reading and writing the time from your RTC Module

Once you have wired up your RTC module. Enter and upload the following sketch.

```
/*================================================================================
//  Author      : Handson Technology
//  Project     : Arduino
//  Description : DS3231 Real Time Clock
//  LiquidCrystal
//  Source-Code : RTC-Tutorial-1.ino
//================================================================================
*/

#include "Wire.h"
#define DS3231_I2C_ADDRESS 0x68

// Convert normal decimal numbers to binary coded decimal
bytedecToBcd(byte val)
{
      return((val/10*16)+(val%10));
}

// Convert binary coded decimal to normal decimal numbers
bytebcdToDec(byte val)
{
      return((val/16*10)+(val%16));
}

void setup()
{
      Wire.begin();
      Serial.begin(9600);
      // set the initial time here:
      // DS3231 seconds, minutes, hours, day, date, month, year
      setDS3231time(00,58,10,1,29,05,16);
}

void setDS3231time(byte second, byte minute, byte hour, byte dayOfWeek, byte dayOfMonth,
byte month, byte year)
{
      // sets time and date data to DS3231
      Wire.beginTransmission(DS3231_I2C_ADDRESS);
      Wire.write(0);// set next input to start at the seconds register
      Wire.write(decToBcd(second));// set seconds
      Wire.write(decToBcd(minute));// set minutes
      Wire.write(decToBcd(hour));// set hours
      Wire.write(decToBcd(dayOfWeek));// set day of week (1=Sunday, 7=Saturday)
      Wire.write(decToBcd(dayOfMonth));// set date (1 to 31)
      Wire.write(decToBcd(month));// set month
      Wire.write(decToBcd(year));// set year (0 to 99)
      Wire.endTransmission();
}

void readDS3231time(byte *second,
byte*minute,
```

```
byte*hour,
byte*dayOfWeek,
byte*dayOfMonth,
byte*month,
byte*year)

{
      Wire.beginTransmission(DS3231_I2C_ADDRESS);
      Wire.write(0);// set DS3231 register pointer to 00h
      Wire.endTransmission();
      Wire.requestFrom(DS3231_I2C_ADDRESS,7);
      // request seven bytes of data from DS3231 starting from register 00h
      *second=bcdToDec(Wire.read()&0x7f);
      *minute =bcdToDec(Wire.read());
      *hour =bcdToDec(Wire.read()&0x3f);
      *dayOfWeek=bcdToDec(Wire.read());
      *dayOfMonth=bcdToDec(Wire.read());
      *month =bcdToDec(Wire.read());
      *year =bcdToDec(Wire.read());
}

voiddisplayTime()
{
byte second, minute, hour,dayOfWeek,dayOfMonth, month, year;

// retrieve data from DS3231
readDS3231time(&second,&minute,&hour,&dayOfWeek,&dayOfMonth,&month,&year);

// send it to the serial monitor
Serial.print(hour, DEC);

// convert the byte variable to a decimal number when displayed
Serial.print(":");
if(minute<10)
{
      Serial.print("0");
}

Serial.print(minute, DEC);
Serial.print(":");

if(second<10)
{
      Serial.print("0");
}

Serial.print(second, DEC);
Serial.print(" ");
Serial.print(dayOfMonth, DEC);
Serial.print("/");
Serial.print(month, DEC);
Serial.print("/");
Serial.print(year, DEC);
Serial.print(" Day of week: ");

switch(dayOfWeek){
      case1:
            Serial.println("Sunday");
            break;
      case2:
            Serial.println("Monday");
            break;
      case3:
            Serial.println("Tuesday");
            break;
      case4:
```

```
            Serial.println("Wednesday");
            break;
      case5:
            Serial.println("Thursday");
            break;
      case6:
            Serial.println("Friday");
            break;
      case7:
            Serial.println("Saturday");
            break;
}
}

void loop()
{
      displayTime();// display the real-time clock data on the Serial Monitor,
      delay(1000);// every second
}
```

There may be a lot of code, however it breaks down well into manageable parts.

It first includes the Wire library, which is used for I2C bus communication, followed by defining the bus address for the RTC as 0x68. These are followed by two functions that convert decimal numbers to BCD (binary-coded decimal) and vice versa. These are necessary as the RTC ICs work in BCD not decimal.

The function setDS3231time() is used to set the clock. Using it is very easy, simple insert the values from year down to second, and the RTC will start from that time. For example if you want to set the following date and time – Wednesday November 26, 2014 and 9:42 pm and 30 seconds – you would use:

```
setDS3231time(30,42,21,4,26,11,14);
```

Note that the time is set using 24-hour time, and the fourth parameter is the “day of week”. This falls between 1 and 7 which is Sunday to Saturday respectively. These parameters are byte values if you are substituting your own variables.

Once you have run the function once it’s wise to prefix it with // and upload your code again, so it will not reset the time once the power has been cycled or micrcontroller reset.

Reading the time form your RTC Is just as simple, in fact the process can be followed neatly inside the function displayTime(). You will need to define seven byte variables to store the data from the RTC, and these are then inserted in the function readDS3231time().

For example if your variables are:

```
byte second, minute, hour,dayOfWeek,dayOfMonth, month, year;
```

… you would refresh them with the current data from the RTC by using:

```
readDS3232time(&second,&minute,&hour,&dayOfWeek,&dayOfMonth,&month,&year);
```

Then you can use the variables as you see fit, from sending the time and date to the serial monitor as the example sketch does – to converting the data into a suitable form for all sorts of output devices.

Just to check everything is working, enter the appropriate time and date into the demonstration sketch, upload it, comment out the *setDS3231time()* function and upload it again. Then open the serial monitor, and you should be provided with a running display of the current time and date, for example:

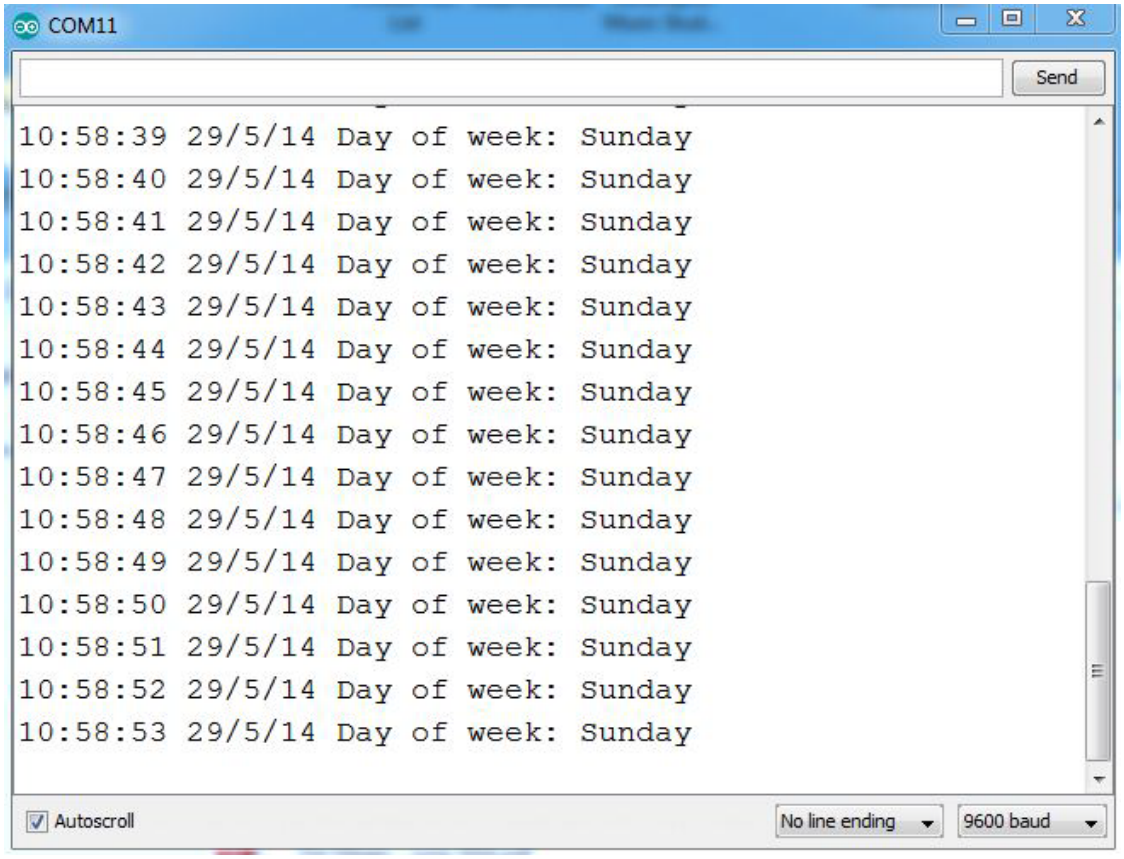


Figure 2-4: Serial Monitor Output for DS3231 RTC

From this point you now have the software tools to set data to and retrieve it from your real-time clock module, and we hope you have an understanding of how to use these inexpensive and highly accurate timing modules.

Chapter-3:DS3231 RTC with LCD Shield

How to make an accurate clock using the Real Time Clock IC DS1307. The time will be shown on a LCD display.

Figure 3-1: DS3231 RTC with LCD Shield

3.1Introduction

How do microcontrollers keep track of time and date? The regular microcontroller has a timer function that starts at 0 (zero) when power is applied, and then it starts to count. In the Arduino world, we can use the millis() function to reset how many milliseconds have passed since the power was applied. When you disconnect and reconnect the power, it starts all over again. This is not so convenient when it comes to clocks and dates.

That is where the Real Time Clockor the RTCchip comes in handy. This IC, with a 3V coin cell battery or another 3V power supply, keeps track of the time and date. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The IC corrects for months with 30/31 days and leap years. The communication is done with the I2C bus.

If the main circuit's Vcc falls below Vbat, the RTC switches automatically over to a low-power battery backup mode. The backup battery is usually a 3Vlithium coin cell battery connected to PIN 3 and GND. This way, the IC is still keeping track of time and date, and when power is applied to the main circuitry, the microcontroller gets the current time and date.

In this project we are using the DS1307. On this IC, PIN 7 is a SQW/OUT pin. You can use this pin to flash a LED, or to clock the microcontroller. We will do both. The following image, from the datasheet, helps us understand the SQW/OUT.

CONTROL REGISTER

The DS1307 control register is used to control the operation of the SQW/OUT pin.

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
OUT	0	0	SQWE	0	0	RS1	RS0

OUT (Output control): This bit controls the output level of the SQW/OUT pin when the square wave output is disabled. If SQWE = 0, the logic level on the SQW/OUT pin is 1 if OUT = 1 and is 0 if OUT = 0.

SQWE (Square Wave Enable): This bit, when set to a logic 1, will enable the oscillator output. The frequency of the square wave output depends upon the value of the RS0 and RS1 bits. With the square wave output set to 1Hz, the clock registers update on the falling edge of the square wave.

RS (Rate Select): These bits control the frequency of the square wave output when the square wave output has been enabled. Table 1 lists the square wave frequencies that can be selected with the RS bits.

SQUAREWAVE OUTPUT FREQUENCY Table 1

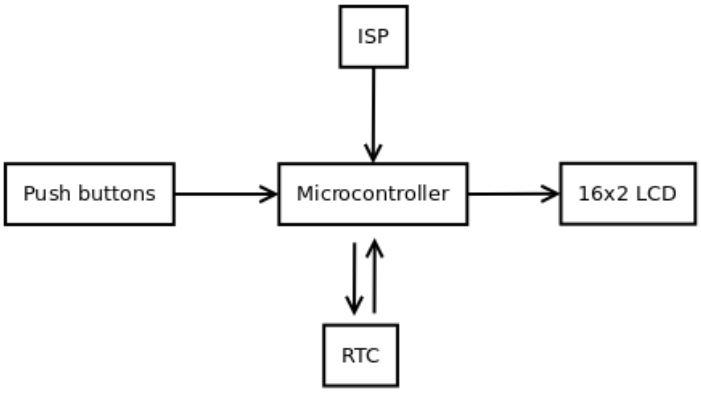
RS1	RS0	SQW OUTPUT FREQUENCY
0	0	1Hz
0	1	4.096kHz
1	0	8.192kHz
1	1	32.768kHz

This table helps you with the frequency:

Freq	BIT7 & BIT6 & BIT5	BIT4	BIT3 & BIT2	BIT1	BIT0
1Hz	0	1	0	0	0
4.096Hz	0	1	0	0	1
8.192Hz	0	1	0	1	0
32.768Hz	0	1	0	1	1

3.2 Hardware

Here's the block diagram for what we want:

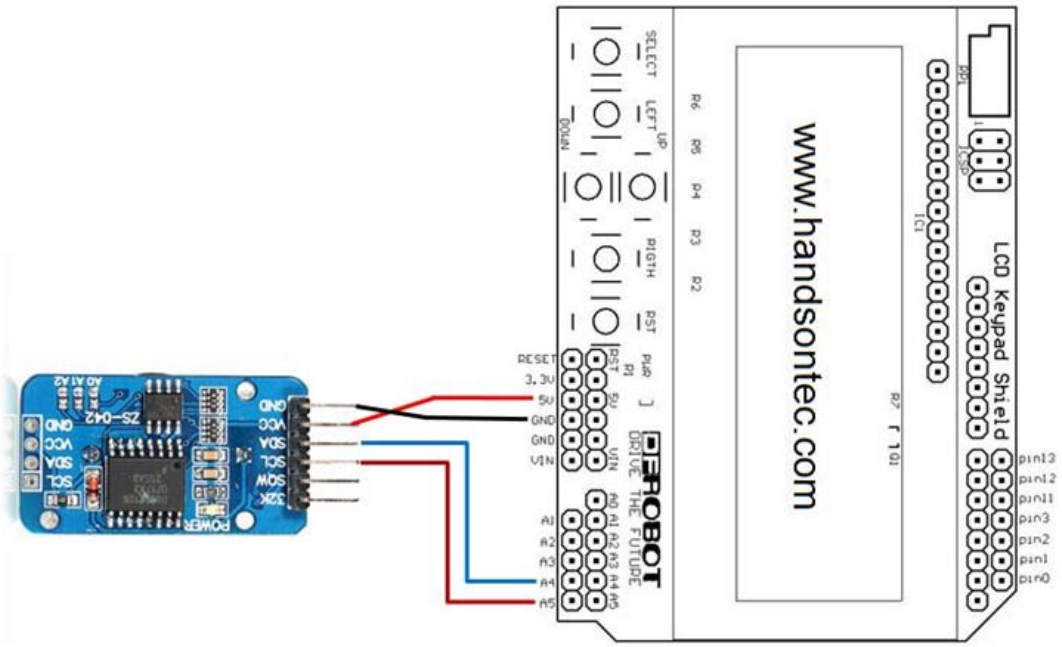


We want:

- ISP (In System Programming) to program the microcontroller
- Push buttons to set the time and date
- The microcontroller to communicate with the RTC via I^2C
- Display the time and date on the LCD

For easy implementation of the above application block diagram, we will be using LCD Shield for Arduino. This shield comes with 16x2 LCD module and 5-keypad.

The LCD shield has 5 programmable buttons on the front, so we used them to set the different features of the clock.



- Button #1 (labeled SELECT) is " Menu " button. This button displays a scrolling list of available functions (Minute Timer, Set Alarm)
- Button #2 (labeled LEFT) is " Select " button. Click to select the displayed menu option. NOTE: Also used to increment by 10s when entering hours, minutes, etc.
- Button #3 & 4 (labeled UP& DOWN) are " Increment& Decrement " buttons. Click these to increment or decrement hours and minutes while setting the timer or alarm. Also used to toggle AM and PM.
- Button #5 (labeled RIGHT) is " GO " button. Click to accept the value entered (such as minutes and hours).
- Button #6 (labeled RST) is the Reset button which reboots the Arduino

3.3The Sketch

```
/*================================================================
//  Author      : Handson Technology
//  Project     : Arduino Uno
//  Description : Motion Sensitive LCD Real-Time Clock/Alarm/Timer
//  LiquidCrystal Library
//  Source-Code : RTC-keypad.ino
//================================================================*/
```

```
#include <Wire.h>
#include <RTClib.h>
#include <LiquidCrystal.h>

RTC_DS1307 RTC;
DateTime now;
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

// define variables
int lcd_key = 0;
int adc_key_in = 0;
int lastDay = 0;
int lastMonth = 0;
int lastYear = 0;
int lastHour = 0;
int lastMinute = 0;
int movementTimer = 0;
int menuOptions = 3;
int menuOption = 0;
int alarmHours = 0;
int alarmMinutes = 0;
bool alarmPM = 0;
bool alarmSet = 0;
bool backLightOn = 1;
bool resetClock = false;

// define constants
const int backLight = 10;
const int pirPin = 16;
#define btnRIGHT  0
#define btnUP     1
#define btnDOWN   2
#define btnLEFT   3
#define btnSELECT 4
#define btnNONE   5
#define beeper A1
#define shortBeep 100
#define longBeep  500

void setup () {
  Serial.begin(57600);
  pinMode(backLight, OUTPUT);
  digitalWrite(backLight, LOW); // turn backlight off
  pinMode(beeper, OUTPUT);
  digitalWrite(beeper, LOW);
  pinMode(pirPin, INPUT);
  Wire.begin();
  RTC.begin();

  if (! RTC.isrunning()) {
    Serial.println("RTC is NOT running!");
    // following line sets the RTC to the date & time
    // this sketch was compiled
    // RTC.adjust(DateTime(__DATE__, __TIME__));
  }
}

void loop () {
  now = RTC.now();
  digitalClockDisplay( ); // update clock
  movementTimer++;
  if (movementTimer > 30) //turn off backlight after 30 cycles
  {
    digitalWrite(backLight, LOW); // turn backlight off
    movementTimer = 0;
  }
```

```
  for (int i = 0; i < 10000; i++)
  {
    button_loop(); //check for button pushed
    int val = digitalRead(pirPin); //read motion sensor
    if (val == HIGH)
    {
      //sense movement?
      digitalWrite(backLight, HIGH); // turn backlight on
      movementTimer = 0;
    }
  }
}

void printDigits(byte digits)
{
  // utility function for digital clock display:
  // prints preceding colon and leading 0
  lcd.print(":");
  if(digits < 10)
    lcd.print('0');
  lcd.print(digits,DEC);
}

void digitalClockDisplay()
{
  bool clockPM = 0;

  if (now.day() != lastDay || resetClock == true)
  {
    lcd.begin(16,2);
    lcd.setCursor(3,0);

    if(now.month() < 10)
      lcd.print('0');
    lcd.print(now.month(), DEC);
    lcd.print("/");

    if(now.day() < 10)
      lcd.print('0');
    lcd.print(now.day(), DEC);
    lcd.print("/");

    int thisYear = now.year();
    lcd.print(thisYear, DEC);
  }

  if (now.minute() != lastMinute || resetClock == true)
  {
    if(now.hour() < 10)
      lcd.setCursor(5,1);
    lcd.setCursor(4,1);

    // 7/29/2012 Fixed noon showing as 12am
    //    if(now.hour() > 12)
    if(now.hour() > 11)
    {
      // 8/19/2013 Fixed noon showing as 0:00pm
      if (now.hour() == 12)
      {
        lcd.print(now.hour(), DEC);
      }
      else{
        lcd.print(now.hour()-12, DEC);
      }
      printDigits(now.minute());
      clockPM = true;
```

```
      lcd.print(" PM ");
      // 7/29/2012 Added alarm set indicator
      if (alarmSet)
        lcd.print("*");
    }
    else
    {
      lcd.print(now.hour(), DEC);
      printDigits(now.minute());
      clockPM = false;
      lcd.print(" AM ");
      // 7/29/2012 Added alarm set indicator
      if (alarmSet)
        lcd.print("*");
    }
  }

  resetClock = false;

  lastDay = now.day();
  lastMonth = now.month();
  lastYear = now.year();
  lastHour = now.hour();
  lastMinute = now.minute();

  //check for alarm
  if (alarmSet)
  {
    if (alarmHours == lastHour && alarmMinutes == lastMinute)
    {
      //sound alarm
      setOffAlarm();
    }
  }
}

void button_loop()
{
  int button = read_LCD_buttons();
  if (button == btnSELECT)
  {
    timedBeep(shortBeep,1);
    selectMenu();
  }
}

void selectMenu()
{
  int button = 0;
  menuOption = 1;
  lcdClear();
  lcd.print("Minute Timer");

  while (menuOption <= menuOptions)
  {
    button = read_LCD_buttons();
    if (button == btnSELECT)
    {
      timedBeep(shortBeep,1);
      menuOption++;

      if (menuOption == 2)
      {
        lcdClear();
        lcd.print("Set/Clear Alarm");
```

```
      }
      if (menuOption == 3)
      {
        lcdClear();
        lcd.print("Set Date/Time");
      }
    }

    if (button == btnLEFT)
    {
      if (menuOption == 1)
      {
        timedBeep(shortBeep,1);
        minuteTimer();
        return;
      }
      if (menuOption == 2)
      {
        timedBeep(shortBeep,1);
        //check for existing alarm
        if (alarmSet)
        {
          clearAlarm();
        }
        else
        {
          setAlarm();
        }
        return;
      }
      if (menuOption == 3)
      {
        timedBeep(shortBeep,1);
        setDateTime();
        return;
      }
    }
  }
}

void clearAlarm()
{
  int button = 0;
  bool clearIt = true;
  char *ampm = "AM";

  lcdClear();
  lcd.print("Alarm Set For");
  lcd.setCursor(0,1);
  lcd.print(alarmHours);
  lcd.print(":");
  lcd.print(alarmMinutes);
  lcd.print(" ");
  if (alarmPM == 1)
    ampm = "PM";
  lcd.print(ampm);
  delay(2000);
  lcdClear();
  lcd.print("Clear Alarm?");
  lcd.setCursor(0,1);
  lcd.print("Yes");

  while (button != btnSELECT)
  {
    button = read_LCD_buttons();
    if (button == btnUP)
```

```
    {
      timedBeep(shortBeep,1);
      clearIt = !clearIt;
    }
    if (button == btnDOWN)
    {
      timedBeep(shortBeep,1);
      clearIt = !clearIt;
    }
    if (button == btnRIGHT)
    {
      timedBeep(shortBeep,1);
      alarmSet = !clearIt;
      if (clearIt)
      {
        lcdClear();
        timedBeep(shortBeep,2);
        lcd.print("Alarm Cleared!");
        delay(2000);
      }
      return;
    }
    lcd.setCursor(0,1);
    if (clearIt)
    {
      lcd.print("Yes");
    }
    else{
      lcd.print("No ");
    }
  }
}

void minuteTimer()
{
  int timerMinutes = getTimerMinutes("Set Minutes", 0, 60);
  if (timerMinutes > 0)
  {
    timedCountDown(timerMinutes*60, "Minute Timer");
  }
  else
  {
    timerCancelled("Timer");
  }
  return;
}

void setAlarm()
{
  int button = 0;
  char *ampm = "AM";
  alarmHours = getTimerMinutes("Set Alarm Hour", alarmHours, 12);
  // Validate alarm hours > 0 and < 13
  if (alarmHours > 0 && alarmHours < 13)
  {
    // 8/1/2012 Pass maxCount to getTimerMinutes
    alarmMinutes = getTimerMinutes("Set Minutes", alarmMinutes, 59);
    //if (alarmMinutes > 0)
    if (alarmMinutes < 60)
    {
      lcdClear();
      lcd.print("Toggle AM/PM");
      lcd.setCursor(0,1);
      //display alarm time
      lcd.print(alarmHours);
      lcd.print(":");
```

```
      if (alarmMinutes < 10)
        lcd.print("0");
      lcd.print(alarmMinutes);
      lcd.setCursor(6,1);
      lcd.print(ampm);
      //get AM/PM
      button = 6;
      while (button != btnSELECT && button != btnRIGHT)
      {
        button = read_LCD_buttons();
        if (button == btnUP || button == btnDOWN)
        {
          timedBeep(shortBeep,1);
          if (ampm == "AM")
          {
            ampm = "PM";
          }
          else
          {
            ampm = "AM";
          }
          lcd.setCursor(6,1);
          lcd.print(ampm);
        }
      }

      if (button == btnRIGHT)
      {
        timedBeep(shortBeep,1);
        alarmSet = true;
        // 8/1/2012 Fixed alarm set PM
        if (ampm == "PM") alarmHours += 12;
        lcd.setCursor(0,0);
        lcd.print("Alarm Set for");
        delay(1000);
        return;
      }
      else
      {
        timerCancelled("Alarm");
        return;
      }
    }
    else
    {
      timerCancelled("Alarm");
    }
  }
  else
  {
    timerCancelled("Alarm");
  }
}

void setDateTime()
{
  int button = 0;
  char *ampm = "AM";

  //get month
  int setMonth = getTimerMinutes("Set Month", lastMonth, 12);
  if (setMonth > 0 && setMonth < 13)
  {
    //get day
    // Fixed default day and hour settings on set date/time
    // Pass maxCount to getTimerMinutes
```

```
int setDay = getTimerMinutes("Set Day", lastDay, 31);
if (setDay > 0 && setDay < 32)
{
  //get year
  int setYear = getTimerMinutes("Set Year", lastYear, 2999);
  if (setYear > 2000 && setYear < 3000)
  {
    //get hour
    int thisHour = lastHour;
    if (thisHour > 12)
    {
      thisHour -= 12;
      ampm = "PM";
    }

    int setHour = getTimerMinutes("Set Hour", thisHour, 12);
    if (setHour > 0 && setHour < 13)
    {
      //get minutes
      int setMinute = getTimerMinutes("Set Minute", lastMinute, 59);

      if (setMinute < 60)
      {
        //get ampm
        lcdClear();
        lcd.print("Toggle AM/PM");
        lcd.setCursor(0,1);
        //display alarm time
        lcd.print(setHour);
        lcd.print(":");
        if (setMinute < 10)
          lcd.print("0");
        lcd.print(setMinute);
        lcd.setCursor(6,1);
        lcd.print(ampm);
        //get AM/PM
        button = 6;
        while (button != btnSELECT && button != btnRIGHT)
        {
          button = read_LCD_buttons();
          if (button == btnUP || button == btnDOWN)
          {
            timedBeep(shortBeep,1);
            if (ampm == "AM")
            {
              ampm = "PM";
            }
            else
            {
              ampm = "AM";
            }
            lcd.setCursor(6,1);
            lcd.print(ampm);
          }
        }
        if (button == btnRIGHT)
        {
          timedBeep(shortBeep,1);
          if (ampm == "PM")
            setHour = setHour + 12;
          RTC.adjust(DateTime(setYear,setMonth,setDay,setHour,setMinute));

          lcd.setCursor(0,0);
          // 8/1/2012 Fixed default day and hour settings on set date/time
          lcd.print("Saving...     ");
          delay(1000);
```

```
            return;
          }
          else
          {
            timerCancelled("");
            return;
          }
        }
        else
        {
          timerCancelled("");
        }
      }
      else
      {
        timerCancelled("");
      }
    }
    else
    {
      timerCancelled("");
    }
  }
  else
  {
    timerCancelled("");
  }

}

// read the buttons
int read_LCD_buttons()
{
  adc_key_in = analogRead(0);      // read the value from the sensor
  // my buttons when read are centered at these valies: 0, 144, 329, 504, 741
  // we add approx 50 to those values and check to see if we are close
  if (adc_key_in > 1000) return btnNONE; // We make this the 1st option for speed
                                         // reasons since it will be the most likely result
  if (adc_key_in < 50)   return btnRIGHT;
  if (adc_key_in < 195)  return btnUP;
  if (adc_key_in < 380)  return btnDOWN;
  if (adc_key_in < 555)  return btnLEFT;
  if (adc_key_in < 790)  return btnSELECT;
  return btnNONE;  // when all others fail, return this...

}

void timedCountDown(int secondCount, char countLabel[])
{
  long seconds = 0;
  long minutes = 0;

  lcdClear();
  lcd.print(countLabel);
  for (int i = secondCount; i >= 0; i--)
  {
    seconds = i;
    minutes = i / 60;
    if (minutes > 0)
    {
      seconds = seconds - (minutes * 60);
```

```
    }

    if (minutes > 0)
    {
      lcd.setCursor(0,1);
      lcd.print(minutes);
      lcd.print(" min ");
    }
    else
    {
      lcd.setCursor(0,1);
    }
    if (seconds < 10) lcd.print("0");
    lcd.print(seconds);
    lcd.print(" sec remaining");
    if (seconds > 0) delay(1000);
    if (read_LCD_buttons() == btnSELECT) //cancel
    {
      timerCancelled("Timer");
      i = 0;
      return;
    }
  }
  lcd.setCursor(6,1);
  timedBeep(longBeep,3);
}

int getTimerMinutes(char timerText[], int startNum, int maxCount)
{
  int minutes = startNum;
  int button = 0;
  lcdClear();
  lcd.print(timerText);
  lcd.setCursor(0,1);
  lcd.print(minutes);

  while (button != btnSELECT)
  {
    button = read_LCD_buttons();
    Serial.println(button);
    // 8/1/2012 Pass maxCount to getTimerMinutes
    if (button == btnLEFT)
    {
      if ((minutes + 10) <= maxCount)
      {
        timedBeep(shortBeep,1);
        minutes = minutes + 10;
      }
      else
      {
        timedBeep(shortBeep,2);
      }
    }
    // 8/1/2012 Pass maxCount to getTimerMinutes
    if (button == btnUP)
    {
      if (minutes < maxCount)
      {
        timedBeep(shortBeep,1);
        minutes++;
      }
      else
      {
        timedBeep(shortBeep,2);
      }
    }
```

```
    if (button == btnDOWN)
    {
      if (minutes > 0)
      {
        timedBeep(shortBeep,1);
        minutes--;
      }
      else
      {
        timedBeep(shortBeep,2);
      }
    }
    if (button == btnRIGHT)
    {
      timedBeep(shortBeep,1);
      return minutes;
    }
    lcd.setCursor(0,1);
    lcd.print(minutes);
    lcd.print("   ");
  }
  return 0;
}

void timedBeep(int beepTime, int beepCount)
{
  for (int i = 0; i < beepCount; i ++)
  {
    digitalWrite(beeper, HIGH);
    delay(beepTime);
    digitalWrite(beeper, LOW);
    delay(beepTime);
  }
}

void lcdClear(){
  //lastDay = 0;
  //lastMinute = 0;
  resetClock = true;
  lcd.clear();
  lcd.begin(16,2);
  lcd.setCursor(0,0);
}

void timerCancelled(char message[])
{
  lcdClear();
  lcd.print(message);
  lcd.print(" Cancelled");
  timedBeep(shortBeep,3);
}

void setOffAlarm()
{
  int button = 0;
  int i = 0;
  Serial.println(i);
  digitalWrite(backLight, HIGH); // turn backlight on
  while (button != btnSELECT)
  {
    button = read_LCD_buttons();
    lcdClear();
    i++;
    if (i > 50)
    {
      lcdClear();
```

```
      lcd.print("Alert Alert");
      lcd.setCursor(0,1);
      lcd.print("     Alert Alert");
      i = 0;
      timedBeep(shortBeep,3);
    }

  }
  timerCancelled("Alarm");
  alarmSet = false;
}
```

Chapter 4: RTC Problem & Solution

4.1: Problem

You want to use the time of day provided by a real-time clock (RTC). External boards usually have battery backup, so the time will be correct even when Arduino is reset orturned off.

4.2: Solution

The simplest way to use an RTC is with a companion library for the Time library, named*DS1307RTC.h*. This recipe is for the widely used DS1307 and DS1337 RTC chips:

```
/*
* TimeRTC sketch
* example code illustrating Time library with real-time clock.
*
*/
#include <Time.h>
#include <Wire.h>

#include <DS1307RTC.h>// a basic DS1307 library that
//returns time as a time_t

void setup(){
      Serial.begin(9600);
      setSyncProvider(RTC.get);// the function to get the time from the RTC
      if(timeStatus()!=timeSet)
            Serial.println("Unable to sync with the RTC");
      else
            Serial.println("RTC has set the system time");
}

void loop()
{
      digitalClockDisplay();
      delay(1000);
}

void digitalClockDisplay(){
// digital clock display of the time
      Serial.print(hour());
      printDigits(minute());
      printDigits(second());
      Serial.print(" ");
      Serial.print(day());
      Serial.print(" ");
      Serial.print(month());
      Serial.print(" ");
      Serial.print(year());
      Serial.println();
}

// utility function for digital clock display:
// prints preceding colon and leading 0.
//
void printDigits(int digits){
      Serial.print(":");
      if(digits <10)
      Serial.print('0');
      Serial.print(digits);
}
```

Most RTC boards for Arduino use the I2C protocol for. Connect the line marked “SCL” (or “Clock”) to Arduino analog pin 5 and “SDA” (or “Data”) to analog pin 4, as shown in Figure 4-1. (Analog pins 4 and 5 are used for I2C. Take care to ensure that you connect the +5V power line and Gnd pins correctly.

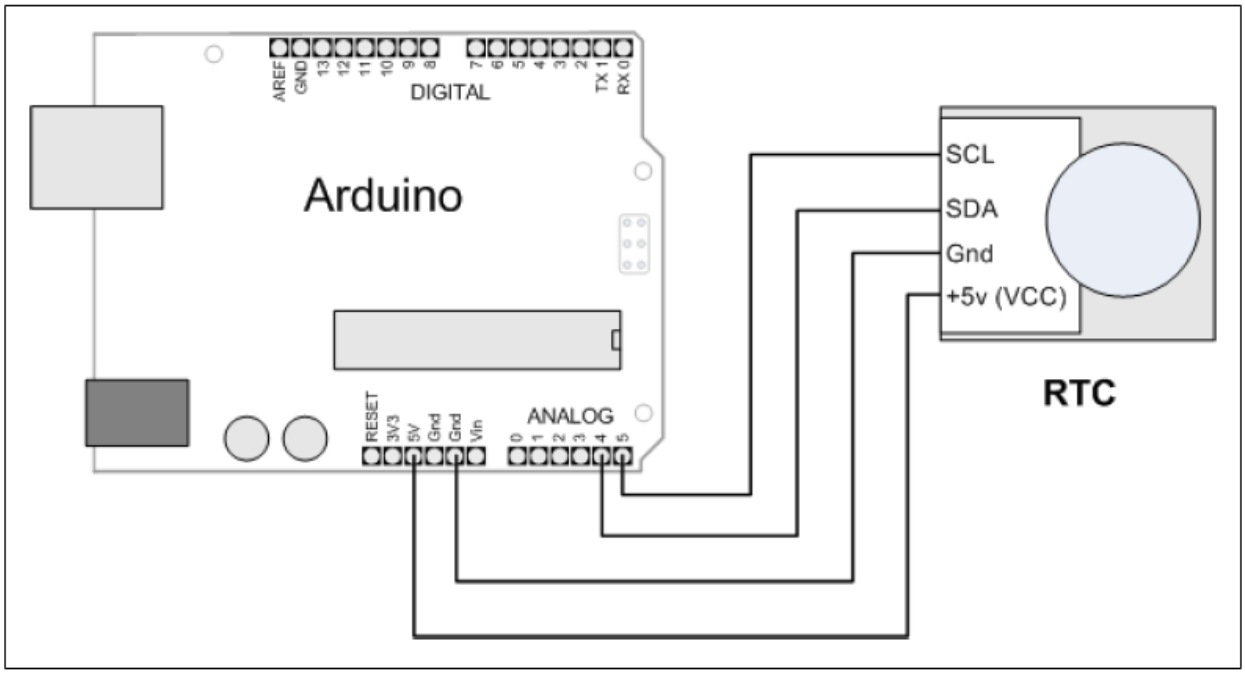


Figure 4.1: RTC Connecting to Arduino Board

4.3: Discussion

The code is similar to other recipes using the Time library, but it gets its value from theRTC rather than from the serial port or hardcoded value. The only additional lineneeded is this:

```
setSyncProvider(RTC.get);// the function to get the time from the RTC
```

The setSyncProvider function tells the Time library how it should get information forsetting (and updating)the time. RTC.get is a method within the RTC library that returnsthe current time in the format used by the Time library (Unix time).Each time Arduino starts, the setup function will call RTC.get to set the time from theRTC hardware.Before you can get the correct time from the module, you need to set its time. Here isa sketch that enables you to set the time on the RTC hardware - you only need to dothis when you first attach the battery to the RTC, when replacing the battery, or if thetime needs to be changed:

```
/*
* TimeRTCSet sketch
* example code illustrating Time library with real-time clock.
*
* RTC is set in response to serial port time message
* A Processing example sketch to set the time is included
* in the download
*/

#include <Time.h>
#include <Wire.h>
#include <DS1307RTC.h>// a basic DS1307 library that
//returns time as a time_t

void setup()
{
Serial.begin(9600);
```

```
setSyncProvider(RTC.get);// the function to get the time from the RTC

if(timeStatus()!= timeSet)
Serial.println("Unable to sync with the RTC");
else
Serial.println("RTC has set the system time");
}

void loop()
{
      if(Serial.available())
      {
                  time_t t = processSyncMessage();
                  if(t >0)
            {
                  RTC.set(t);// set the RTC and the system time to the received value
                  setTime(t);
            }
      }
      digitalClockDisplay();
      delay(1000);
}

voiddigitalClockDisplay()
{
      // digital clock display of the time

      Serial.print(hour());
      printDigits(minute());
      printDigits(second());
      Serial.print(" ");
      Serial.print(day());
      Serial.print(" ");
      Serial.print(month());
      Serial.print(" ");
      Serial.print(year());
      Serial.println();
}

// utility function for digital clock display:
// prints preceding colon and leading 0.
//
voidprintDigits(int digits)
{
      Serial.print(":");
      if(digits <10)
      Serial.print('0');
      Serial.print(digits);
}

/* code to process time sync messages from the serial port */
#define TIME_MSG_LEN 11 // time sync to PC is HEADER
//followed by Unixtime_t as ten ascii digits

#define TIME_HEADER 'T' // Header tag for serial time sync message

time_tprocessSyncMessage()
{
// return the time if a valid sync message is received on the serial port.
// time message consists of a header and ten ascii digits

while(Serial.available() >= TIME_MSG_LEN )
{
      char c = Serial.read();
      Serial.print(c);
      if( c == TIME_HEADER )
```

```
        {
        time_tpctime = 0;
        for(inti = 0; I < TIME_MSG_LEN -1; i++)
        {
              c =Serial.read();
              if( c >='0'&& c <='9'){
              pctime = (10*pctime)+(c -'0');      // convert digits to a number
              }
        }
        Return pctime;
        }
        }
        Return 0;
}
```

The following function is called when a time message is received from the computer to set the RTC:

```
RTC.set(t);// set the RTC and the system time to the received value
setTime(t);
```

Chapter 5: To Display Real Time and Temperature on Arduino LCD+Keypad Shield

Figure 5.1: Display Time/Date and Temperature

This time round we write a scratch to display Real-Time plus Temperature using Arduino LCD+Keypad shield. The sketch with comments is self explanatory. Beside display the time and date, it also shown how to read the built-in temperature sensor data and display on LCD.

Sketch 5.1: Verify and update the below sketch to Arduino board. Display shown in Figure5.1 should appear and start the clock time and measure the surround environment temperature with high accuracy.

```
/*================================================================================
//  Author      : Handson Technology
//  Project     : Arduino
//  Description : DS3231 Real Time Clock + Temperature with LCD Shield
//  LiquidCrystal Library
//  Source-Code : RTC3231.ino
```

```
//========================================================================
*/

#include <Wire.h>
#include "ds3231.h"
#include <LiquidCrystal.h>

LiquidCrystal lcd(8,9,4,5,6,7);

#define BUFF_MAX 128

uint8_t time[8];
char recv[BUFF_MAX];
unsigned int recv_size = 0;
unsigned long prev, interval = 1000;

void setup()
{
    Serial.begin(9600);
    Wire.begin();
    DS3231_init(DS3231_INTCN);
    memset(recv, 0, BUFF_MAX);
    Serial.println("GET time");
    lcd.begin(16, 2);

    // Turn On the LCD Backlight
    pinMode(10, OUTPUT);        // sets backlight pin-10 as PWM output
    analogWrite(10, 200);       // Set backlight to 50% brightness
    lcd.clear();

    // The below function is use to set the time.
    // Set the correct time "TSSMMHHWDDMMYYYY".
    // Compile and upload to Arduino once, after that comment out this 2-functions
    // compile again and upload one more time to Arduino.

    //Serial.println("Setting time");
    //parse_cmd("T151307416062016",16);
}

void loop()
{
    char in;
    char tempF[6];
    float temperature;
    char buff[BUFF_MAX];
    unsigned long now = millis();
    struct ts t;

    // show time once in a while
    if ((now - prev > interval) && (Serial.available() <= 0)) {
        DS3231_get(&t); //Get time
        parse_cmd("C",1);
        temperature = DS3231_get_treg(); //Get temperature
        dtostrf(temperature, 5, 1, tempF);

        lcd.clear();
        lcd.setCursor(1,0);

        lcd.print(t.mday);

        printMonth(t.mon);

        lcd.print(t.year);

        lcd.setCursor(0,1); //Go to second line of the LCD Screen
        lcd.print(t.hour);
```

```
        lcd.print(":");

        if(t.min<10)
        {
          lcd.print("0");
        }

        lcd.print(t.min);
        lcd.print(":");
        if(t.sec<10)
        {
          lcd.print("0");
        }
        lcd.print(t.sec);

        lcd.print(' ');
        lcd.print(tempF);
        lcd.print((char)223);
        lcd.print("C ");
        prev = now;
    }


    if (Serial.available() > 0) {
        in = Serial.read();

        if ((in == 10 || in == 13) && (recv_size > 0)) {
            parse_cmd(recv, recv_size);
            recv_size = 0;
            recv[0] = 0;
        } else if (in < 48 || in > 122) {;       // ignore ~[0-9A-Za-z]
        } else if (recv_size > BUFF_MAX - 2) {   // drop lines that are too long
            // drop
            recv_size = 0;
            recv[0] = 0;
        } else if (recv_size < BUFF_MAX - 2) {
            recv[recv_size] = in;
            recv[recv_size + 1] = 0;
            recv_size += 1;
        }

    }
}

void parse_cmd(char *cmd, int cmdsize)
{
    uint8_t i;
    uint8_t reg_val;
    char buff[BUFF_MAX];
    struct ts t;

    //snprintf(buff, BUFF_MAX, "cmd was '%s' %d\n", cmd, cmdsize);
    //Serial.print(buff);

    // TssmmhhWDDMMYYYY aka set time
    if (cmd[0] == 84 && cmdsize == 16) {
        //T355720619112011
        t.sec = inp2toi(cmd, 1);
        t.min = inp2toi(cmd, 3);
        t.hour = inp2toi(cmd, 5);
        t.wday = inp2toi(cmd, 7);
        t.mday = inp2toi(cmd, 8);
        t.mon = inp2toi(cmd, 10);
        t.year = inp2toi(cmd, 12) * 100 + inp2toi(cmd, 14);
        DS3231_set(t);
        Serial.println("OK");
```

```
    } else if (cmd[0] == 49 && cmdsize == 1) {  // "1" get alarm 1
        DS3231_get_a1(&buff[0], 59);
        Serial.println(buff);
    } else if (cmd[0] == 50 && cmdsize == 1) {  // "2" get alarm 1
        DS3231_get_a2(&buff[0], 59);
        Serial.println(buff);
    } else if (cmd[0] == 51 && cmdsize == 1) {  // "3" get aging register
        Serial.print("aging reg is ");
        Serial.println(DS3231_get_aging(), DEC);
    } else if (cmd[0] == 65 && cmdsize == 9) {  // "A" set alarm 1
        DS3231_set_creg(DS3231_INTCN | DS3231_A1IE);
        //ASSMMHHDD
        for (i = 0; i < 4; i++) {
            time[i] = (cmd[2 * i + 1] - 48) * 10 + cmd[2 * i + 2] - 48; //ss,mm,hh,dd
        }
        byte flags[5] = { 0, 0, 0, 0, 0 };
        DS3231_set_a1(time[0], time[1], time[2], time[3], flags);
        DS3231_get_a1(&buff[0], 59);
        Serial.println(buff);
    } else if (cmd[0] == 66 && cmdsize == 7) {  // "B" Set Alarm 2
        DS3231_set_creg(DS3231_INTCN | DS3231_A2IE);
        //BMMHHDD
        for (i = 0; i < 4; i++) {
            time[i] = (cmd[2 * i + 1] - 48) * 10 + cmd[2 * i + 2] - 48; // mm, hh, dd
        }
        byte flags[5] = { 0, 0, 0, 0 };
        DS3231_set_a2(time[0], time[1], time[2], flags);
        DS3231_get_a2(&buff[0], 59);
        Serial.println(buff);
    } else if (cmd[0] == 67 && cmdsize == 1) {  // "C" - get temperature register
        Serial.print("temperature reg is ");
        Serial.println(DS3231_get_treg(), DEC);
    } else if (cmd[0] == 68 && cmdsize == 1) {  // "D"-reset status reg. alarm flags
        reg_val = DS3231_get_sreg();
        reg_val &= B11111100;
        DS3231_set_sreg(reg_val);
    } else if (cmd[0] == 70 && cmdsize == 1) {  // "F" - custom fct
        reg_val = DS3231_get_addr(0x5);
        Serial.print("orig ");
        Serial.print(reg_val,DEC);
        Serial.print("month is ");
        Serial.println(bcdtodec(reg_val & 0x1F),DEC);
    } else if (cmd[0] == 71 && cmdsize == 1) {  // "G" - set aging status register
        DS3231_set_aging(0);
    } else if (cmd[0] == 83 && cmdsize == 1) {  // "S" - get status register
        Serial.print("status reg is ");
        Serial.println(DS3231_get_sreg(), DEC);
    } else {
        Serial.print("unknown command prefix ");
        Serial.println(cmd[0]);
        Serial.println(cmd[0], DEC);
    }
}

void printMonth(int month)
{
  switch(month)
  {
    case 1: lcd.print(" January ");break;
    case 2: lcd.print(" February ");break;
    case 3: lcd.print(" March ");break;
    case 4: lcd.print(" April ");break;
    case 5: lcd.print(" May ");break;
    case 6: lcd.print(" June ");break;
    case 7: lcd.print(" July ");break;
    case 8: lcd.print(" August ");break;
```

```
    case 9: lcd.print(" September ");break;
    case 10: lcd.print(" October ");break;
    case 11: lcd.print(" November ");break;
    case 12: lcd.print(" December ");break;
    default: lcd.print(" Error ");break;
  }
}
```

7. Appendix

7.1: LCD Application Manual

- Download this Manual
- LCD+KeypadShiled Schematic

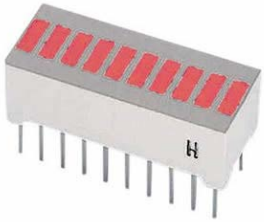
7.2: HandsOn Technology Products Quality Commitments

HandsOnTechnologywish to be perceived as simple and affordable by our customers. However the joy over a low price is never greater than the disappointment over poor quality products. All our parts are original genuine parts with proper data specifications from manufacturers. This is to ensure you always get the high quality genuine original part as stated in our products information.

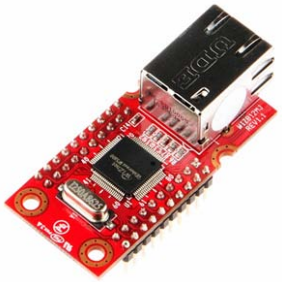
www.handsontec.com

Welcome to Handsontec Store

LCD+Keyboard Shield



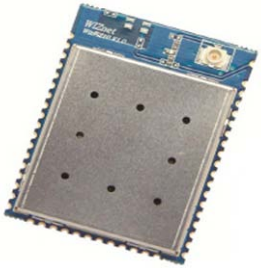
10-Segments LED Bar Display



Ethernet Module

Arduino Uno

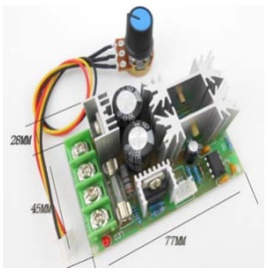
MicroSD Breakout Board



WiFi Module

20x4 LCD Display Module

Stepper Motor Driver



PWM Motor Speed Controller

Breakout Board & Modules

Integrated Circuits

Discrete Parts

Assembled Kits

Connectors

This page is intentional left blank.