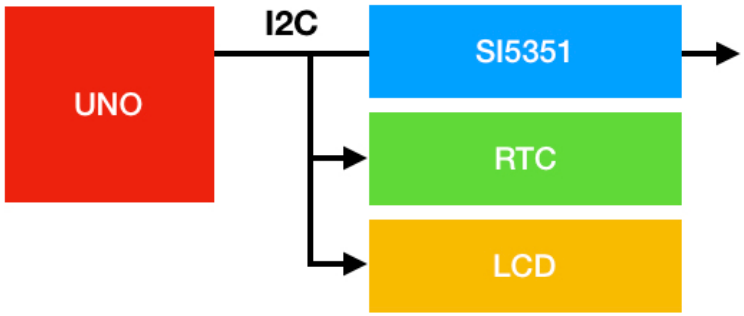
26. Application - RTC

First build the RTC part of the circuit on your breadboard. Arduino UNO + RTC + LCD



RTC and LCD connections

SCL	A5
SDA	A4
+5V	
GND	

RTC_SET_TIME

The first sketch is used to set the RTC date and time to the nearest second. when it is started, or the UNO is reset you can enter the date and time:

"YYMMDDwHHMMSS" - year, month, day, day-of-week, hour, minute, second. Note that day_of_week starts with Sunday = 0, Monday = 1, etc, 24 hour clock

This is entered on the IDE monitor wndow, and the RETURN key or SEND is pressed exactly on the second. This will program the DS3231 and the Date and Time will be displayed.

```
#include "Wire.h"
#include "LiquidCrystal_I2C.h"

#define LCDADDR 0X3F                    // I2C addresses
#define RTCADDR 0x68

byte secOld, sec, mns, hrs;             // data & time
byte dow;
byte dy, mth, yr;

char rtcBuf[20];                        // input time
bool inString;

LiquidCrystal_I2C lcd(LCDADDR, 16, 2);  // lcd object
```

The sketch uses two libraries, “Wire.h” is a serial communication library for TWI (Two Wire Interface, and example of which is the I2C bus), the other is the Liquid Crystal I2C library for driving the display. The I2C addresses of the LCD and the RTC are defined.

Next is a set of variables to store the time and date. Followed by a character buffer rtcBuf, an array able to hold up to 19 characters (and the terminator ‘/0’). The boolean flag inString is set when an input string has been received, it is used to display the correct information, either “-> YYMMDDwHHMMSS” or the date and time from the RTC on two lines.

The object “lcd” is created from its library.

setup()

```
void setup() {
  Serial.begin(9600);                    // start serial comms
  lcd.begin();

  strcpy(rtcBuf, "");                    // empty buffer
  inString = false;                      // input string?

  dispUpdate();
}
```

The sketch communicates by USB with the IDE monitor window to input the date and time, serial comms is initialised here, as is the lcd. The function strcpy(out, in) is a C library function to copy the string ’in’ to the string ‘out’ thus clearing the buffer. The starting display is updated on the LCD.

loop()

```
void loop() {
  while (getMsg(rtcBuf)) {               // check and get input
    inString = true;
    asciiToByte();                       // convert to bytes
    setRTC();                            // set the RTC
  }
  readRTC();                             // read the data/time
  if(secOld != sec)
     dispUpdate();
  secOld = sec;
}
```

As is often the case, the loop() function is simple. It calls getMsg to see if a message has been entered, and if so it converts it to byte data and sets the RTC. It then reads the RTC,

saves the previous “sec” value and displays the date and time. The next time round the loop if sec not been updated it skips the update of the display. This stops the display flickering.

Now we get to the complicated bit

A number of conversions must be made. The input data in rtcBuf is in ASCII. The date and time are variables are decimal bytes (0-255). So the first conversion we have to make is to read the character in the ASCII buffer rtcBuf[0] to rtcBuf[12] and convert them in pairs to decimal yr, mth, dy etc.

```
void asciiToByte() {                              // convert to bytes
  // convert ASCII codes to bytes
  yr = ((byte)rtcBuf[0] - 48) * 10 + (byte)rtcBuf[1] - 48;
  mth = ((byte)rtcBuf[2] - 48) * 10 + (byte)rtcBuf[3] - 48;
  dy = ((byte)rtcBuf[4] - 48) * 10 + (byte)rtcBuf[5] - 48;
  dow = ((byte)rtcBuf[6] - 48);
  hrs = ((byte)rtcBuf[7] - 48) * 10 + (byte)rtcBuf[8] - 48;
  mns = ((byte)rtcBuf[9] - 48) * 10 + (byte)rtcBuf[10] - 48;
  sec = ((byte)rtcBuf[11] - 48) * 10 + (byte)rtcBuf[12] - 48;
}
```

ASCII ‘0’ (zero) is decimal 48. The function reads a character as two bytes (e.g. March is month ’03’), subtracting 48 (ASCII for zero) for each, multiplies the MSB by x10 and adds them together to get the values.

The RTC itself uses BCD input/output. This is two nibbles (4 bits) of an 8 bit byte. XXXXYYYY. where XXXX & YYYY are 0-15, so March 03 is 0000 0011 (0 3) in BCD. We need to convert both ways so decimal to BCD and BCD to decimal, like this

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

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

Lastly we need to set and read the RTC (in BCD). This is done in the order sec, mns....yr.

```
void setRTC() {                                 // program RTC
  Wire.beginTransmission(RTCADDR);
  Wire.write(0);                                // next input at sec register

  Wire.write(decToBcd(sec));                    // set seconds
  Wire.write(decToBcd(mns));                    // set minutes
  Wire.write(decToBcd(hrs));                    // set hours (0-23)
  Wire.write(decToBcd(dow));                    // set day of week
  Wire.write(decToBcd(dy));                     // set date (1 to 31)
  Wire.write(decToBcd(mth));                    // set month (1-12)
  Wire.write(decToBcd(yr));                     // set year (0 to 99)
  Wire.endTransmission();
}
```

```
void readRTC() {                                // read RTC
  Wire.beginTransmission(RTCADDR);              // Reset the RTC register pointer
  Wire.write(0x00);
  Wire.endTransmission();

  Wire.requestFrom(RTCADDR, 7);                 // request 7 bytes from the RTC address


  sec = bcdToDec(Wire.read());                  // 0 - 59
  mns = bcdToDec(Wire.read());                  // 0 - 59
  hrs = bcdToDec(Wire.read() & 0b111111);       // mask 12/24 bit
  dow = bcdToDec(Wire.read());                  // 0 = Sunday
  dy  = bcdToDec(Wire.read());                  // 1 - 31
  mth = bcdToDec(Wire.read());                  // 0 = jan
  yr  = bcdToDec(Wire.read());                  // ..yy
}
```

Phew! Who knew data and time would be so complicated to handle? But it is important to understand the different coding systems (ASCII, Decimal and Binary Coded Decimal).

Display

For the display we need to convert 01, 02, etc to Jan, Feb, etc. And the Time to 10:30:55 format. This is done with two functions dispDate(...) and dispTime(...) which you can see in the sketch, nothing complicated here.

Lastly we need a display function to show the date and time on the LCD display

```
void dispUpdate() {                          // YYDD... and date time
  lcd.clear();

  if (inString == false) {
    lcd.setCursor(0, 0);
    lcd.print("Enter YYMMDDwHHSSMM");
  }
  else {
    lcd.setCursor(0, 0);
    dispDate(dow, dy, mth, yr);
    lcd.setCursor(0, 1);
    dispTime(hrs, mns, sec);
  }
}
```

That's it. When you run the sketch, or reset the UNO it will remind you of the format to enter the date and time. After that the RTC will be set and it will display the date and time.

You only need to set it occasionally, but you must get it right to within the first second of an even minute for digital applications like WSPR (Whisper).