07. Programming Onboard Devices

This tutorial will walk you through how to connect to devices on the multifunction Arduino board, including the real-time clock, temperature sensor and 4-digit display.

Introduction

In Chapter 5 we learned how to use some of the onboard devices on the Arduino Rich UNO R3 board, including the onboard LED, piezoelectric buzzer, knob potentiometer and touch buttons. In this chapter, we will learn how to use some of the remaining onboard devices, including the real-time clock, temperature sensor and 4-digit display. These more advanced features are often very useful when building prototypes and custom instruments for biological applications. By the end of this chapter you will be able to connect to, and make use of, the majority of Arduino Rich UNO board’s inbuilt devices.

Objectives

- Write the current date-time to the onboard real time clock.
- Write a program to read the current date-time from the board and display it on an LCD screen.
- Read the temperature from the onboard LM75 temperature sensor.
- Format the raw output of the temperature sensor using a join node.
- Display readings from the onboard temperature sensor on the onboard 4-digit display.
- Compare the utility of the onboard 4-digit, external 4-digit, and LCD displays.

Requirements

- Computer running MacOS, Windows or Linux
- Arduino Rich UNO R3 board
- Expansion shield
- I2C 1602 LCD display module
- External 4-digit display module
- Hook-up wires/Dupont line

Step-by-Step Instructions

Step 1: Downloading Tutorial Software

There are no specific XOD files required for this tutorial. However, you will need to download some additional libraries. Cesar Sosa has created several useful XOD libraries for controlling the Rich UNO R3 board. In particular, you should download the library cesars/tm1637, which contains a number of nodes for controlling the onboard 4-digit display. When using nodes from this library for the first time, you may be notified that you need to install required dependencies. These are additional libraries that are required to make these nodes work, and you should install them when prompted. You will also need to download the gst/lm75atempsensor library, for access to the lm75a-temp-sensor node.
**Step 2: Real Time Clock Tutorial**

The Arduino Rich UNO R3 board has a built-in clock and back-up battery, located to the right of the piezoelectric buzzer. This clock is based on a DS1307 high-precision real-time clock module with I2C serial interface, and address 68h.

![Real Time Clock (RTC) Module on the Arduino Rich UNO R3 Board](image)

There is an excellent tutorial and guide to building and using the real time clock module available at: [https://xod.io/docs/guide/rtc-example/](https://xod.io/docs/guide/rtc-example/). The guide works through an example for how to use the datetime ([https://xod.io/libs/xod/datetime/](https://xod.io/libs/xod/datetime/)) and ds-rtc ([https://xod.io/libs/xod-dev/ds-rtc/](https://xod.io/libs/xod-dev/ds-rtc/)) libraries. You will learn how to make a program to write the current date-time to the memory of the RTC module, followed by how to read this information from the RTC module and display it on a screen. To adapt the guide for use with the Rich UNO R3 board, you will need to wire the LCD screen to the Arduino board as you did in Chapter 6, and change the address of the screen from 27h to 38h. Follow the tutorial available on the XOD website to complete the guide and learn how to create a simple digital clock.

The tutorial information can be found [here](https://xod.io/docs/guide/rtc-example/).
Download the completed XOD code for this tutorial [here](https://xod.io/docs/guide/rtc-example/).

Once you have completed this tutorial, you can try to adapt and experiment with the real-time clock. For example, by reformatting the time and date.

![Illustration of a Digital Clock Displayed on an I2C 16x2 Display](image)
If you do not have hardware available, you can simulate some of the material in this tutorial by replacing the rtc node with a datetime node and the text-lcd-i2c-16x2 node with two watch nodes, as below.*
Step 3: Reading from the Temperature Sensor

An LM75 temperature sensor is mounted on the board next to the real time clock. The microcontroller communicates with it via an I2C interface, at the address 48h. The temperature sensor is internally calibrated and measures to ~0.1 degree C accuracy. The XOD `lm75a-temp-sensor` node can be found in the `gst/lm75atempsensor` XOD library, and allows simple access to the on-board sensor.

![LM75 Temperature Sensor on the Arduino Rich UNO R3 Board](image)

Create a new patch called ‘tempsensor’ and add an `lm75a-temp-sensor` node. The numerical output which comes from the `lm75a-temp-sensor` node can be reformatted using the `format-number` node. This allows you to define the number of significant digits. Add a `format-number` node and set DIG to 1. The output of the `format-number` node is a string, which can be fed into a `join` node to include additional text before sending the information to a display. Add a `join` node to your patch. Join the string pin (STR) from the `format-number` node to the S1 pin of the join node. Set the S2 pin to ‘oC’. The D pin of the `join` node defines how the strings are separated, this could be a comma, decimal point, colon etc. but is automatically set as a space. To see how the format of the number changes at each stage, try adding watch nodes to each output. Use the ‘Upload and Debug’ button (looks like a ladybird) in the bottom right hand corner of XOD to upload the program and watch each output at the same time.
XOD Patch to Format the Output of the LM75 Temperature Sensor

*If you do not have a hardware kit, you can practice formatting numbers using a tweak-number node instead of the lm75a-temp-sensor node.*

**Step 4: Writing to the 4-Digit Display**

Now that we can read the temperature, we will learn how to display the output using the onboard 4-digit display. Start by opening a new patch and calling it 'tempdisplay'. Add a tm1637-dev node from the cesars/tm1637 library. This node inputs the correct ports for the CLK and DIO pins, set these to D10 and D11. Now add a tm1637-bright node and connect it. This node sets the brightness of the LEDs (0 dimmest - 7 brightest). Make sure that the boolean pin is set to ‘True’ so that the display turns on, and the SET pin is on ‘Boot’ so that the brightness is set when the program first loads. Finally add a tm1637-autorange node and connect it. This node takes an input (Value) and uploads it to the display. To test this patch, set the UPD pin to ‘Loop’ so that it updates continuously, add a tweak-number node to the Value pin and press ‘Upload and Debug’. Have a play around entering different values into the tweak-number node and seeing how they’re displayed.
*Please note that the four-digit display cannot be simulated without hardware.*

**Step 5: Using the 4-Digit Display to Show Temperature Readings**

Now let's display some real data on the 4-digit display. Add an `lm75a-temp-sensor` node to your patch and connect it to the Value pin of the `tm1637-autorange` node. Upload the code and see how the number is displayed.

![XOD Patch to Show the Temperature on the 4-Digit Display](image)

Note that the onboard 4-digit display uses a colon rather than a decimal point. However, the additional 4-digit display included in the Biomaker starter kit has a slightly different format, with a point after each digit. Try connecting the external 4-digit display to the board and see how this changes the format. Using pins CLK-D10, DIO-D11, VCC-VCC and GND-GND will allow you to mirror what is happening on the onboard display.

![Comparison of Onboard and External 4-Digit Displays](image)

You could also use the kit’s 16x2 LCD screen to display this information. Try connecting the LCD screen (as in Chapter 6) and using it to display the temperature. Using the LCD screen allows you to add additional text and format the display of the number as we did in Step 3.
Step 6: Using the 4-Digit Display to Show the Real Time Clock

Now that you can work the 4-digit display, try combining the real time clock tutorial with the four digit display to make an onboard clock display. You will also need to use another node from the cesars/tm1637 library: tm1637-decx.

The tm1637-decx node

This node is used instead of the tm1637-autorange node and allows you to play around with which digits are displayed in which position. The pins perform the following tasks:

- Number takes your input value.
- Dots determines the type of display, this is explained in the cesars/tm1637 library under 00-list-readme, and should be set to 2 for the onboard display.
- Zeros determines whether zeros should be added to empty spaces in the display, e.g. if TRUE it will display 0542 and if FALSE it will display 542.
- Length sets the length of the number displayed. If the value input is longer than the length it will take the last two integer digits.
- Position determines where the first digit of the input number should be displayed. The left-most position on the display is 0 and the right-most position is 3.
- UPD determines when the display should be updated, set this to ‘continuously’ to update constantly.

Using this information, try to display the current time in hours and minutes on the onboard 4-digit display. Hint: you will need to use the unpack-datetime node, and two tm1637-decx nodes. Once you have done this, it should be easy to display other data from the real time clock, for example, the year, or the time in minutes and seconds. If you are having trouble, try referring to the XOD patch diagram displayed on the final page of this chapter.
Step 7: Displaying Text on the 4-Digit Display

The cesars/tm1637 library also contains a useful node for displaying letters on the 4-digit display: tm1637-digits. This node takes inputs in the form of a hexadecimal byte for each digit on the display. Working out the correct hexadecimal to display the letter you want is tricky, but there is a free app which makes this a lot easier. The 7 Segment Editor app for Android allows you to highlight the segments you want to light up, and spits out a two-digit code to use. The onboard display is a common cathode display, so use the common cathode code and add a ‘h’. This is the hexadecimal to use in your tm1637-digits node.

For example, the common cathode code for the letter ‘t’ is 78h, for the letter ‘e’ is 7Bh and for the letter ‘s’ is 6Dh. We can input these hexadecimal codes into the pins DIG0, DIG1, DIG2 and DIG3 to display the word ‘test’ on our 4-digit display.
Try combining the temperature display from Step 5 with a text display to show the temperature in °C. The hexadecimal code for ‘°’ is 63h and for ‘C’ is 39h. If you are having trouble, try referring to the XOD patch diagram displayed on the final page of this chapter.

Arduino Rich Uno R3 Board Displaying the Temperature in °C

**Further Information**

**Next Chapter**

In the next chapter we will walk you through how to use the skills you have learned so far to make your own data logger. You will use the temperature and ultrasonic sensors to collect data, and write a program to log this data for later analysis. Next chapter available soon.
Help: XOD Patch Diagrams

XOD Patch to Display the Current Time on the Onboard 4-Digit Display

XOD Patch to Display the Temperature in °C