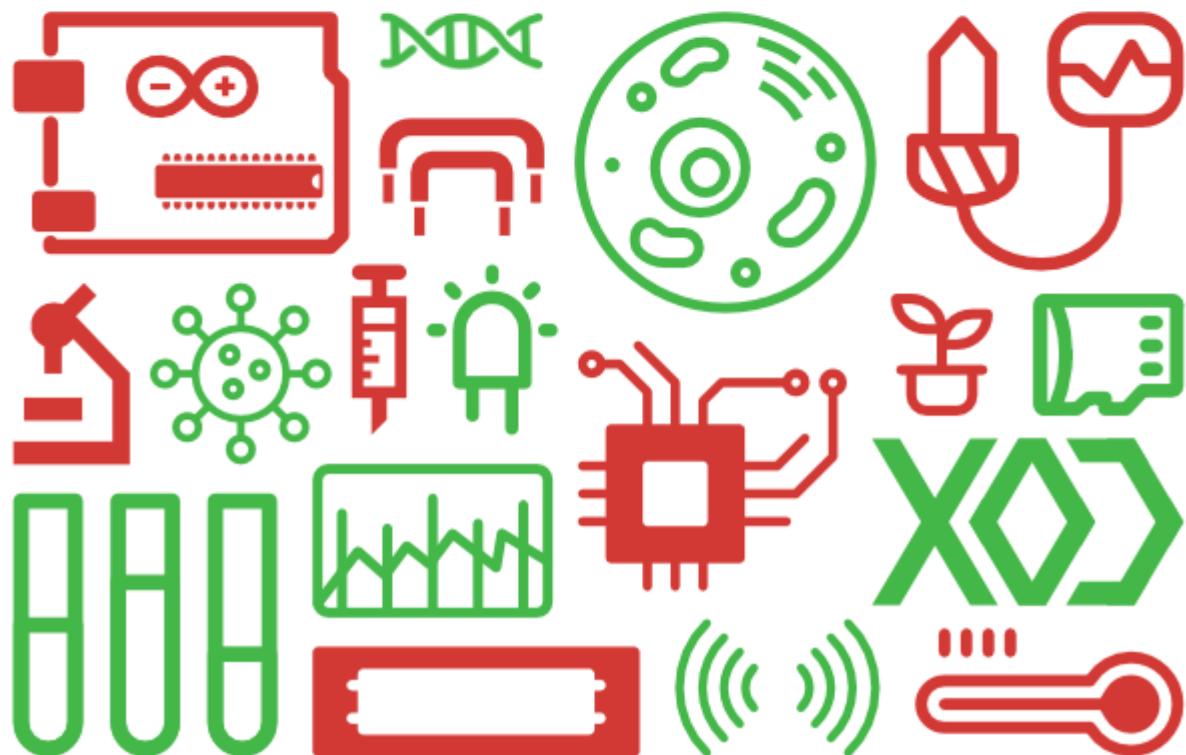


No-Code Programming for Biology



Handbook 2020



Contents

Chapter 1: Introduction to Biomaking and No-Code Programming	3
Welcome to No-Code Programming for Biology	
About Biomaker	
About this Handbook	
More Ways to Get Involved	
Useful Information	
Chapter 2: A Guided Tour of the Biomaker Starter Kit	7
The 2020 Biomaker Starter Kit	
Arduino Rich UNO R3 Board	
Kit Components	
Additional Components	
Making your own kit	
Further Information	
Chapter 3: Understanding the Arduino Board	15
Get to Know your Board!	
Power Pins	
ANALOG Pins	
DIGITAL Pins	
Pin Connections	
Further Information	
Chapter 4: XOD Visual Programming	24
XOD: Visual Programming for Biomaker	
Further Information	
Chapter 5: Getting Started	32
Introduction	
Step-by-Step Guide	
Chapter 6: Handling Simple Input Output Devices	41
Introduction	
Step-by-Step Guide	
Chapter 7: Programming Onboard Devices	50
Introduction	
Step-by-Step Instructions	
Chapter 8: DIY data logger	61
Introduction	
Step-by-Step Guide	
Further Information	
Chapter 9: Additional Tutorials and Useful Tools	71
Introduction	
Using the OLED Screen	
Using the Infrared Remote Control	
Control of Sequences: Traffic Light	
Handbook Summary	
Where to Find Additional Help	

Annex: Biomaker Starter Kit Specifications

Biomaker Starter Kit: Rich UNO R3 board & parts
Open Smart Rich UNO R3 Specifications
Open Smart Kit (B) Specifications
Additional Components
Rich UNO R3 documentation

79

Chapter 1: Introduction to Biomaking and No-Code Programming

This chapter provides an introduction to the Biomaker programme and the No-Code Programming for Biology online workshop.

Welcome to No-Code Programming for Biology

Thank you for signing up to the “No-code Programming for Biology” workshop! We have put together this virtual handbook in response to the current need for social distancing, and to help participants learn the basics of biomaking from wherever they may be. This handbook will guide you through the workshop materials, at your own pace and in your own environment. This course is designed to introduce biologists, or other scientists with little formal programming training, to the basics of biomaking, including:

- Use of Arduino-based microcontrollers
- Use of sensors, displays and actuators
- Use of XOD visual programming

Training in the use of these new tools will allow you to build a wide range of instruments and devices that can be useful for biological experiments in the lab and field. The 2020 Biomaker kit can be used to build bioinstrument prototypes for measurement and control of biological systems. These have a wide range of applications including [instrumentation](#), [microscopy](#), [microfluidics](#), [3D printing](#), [biomedical devices](#), [DNA design](#), [plant sciences](#) and [outreach and public engagement](#). You can find previous examples of Biomaker projects using these tools on the Biomaker website: www.biomaker.org

These new skills can be enabling in many ways. Scientists can gain expertise and new ways of thinking to apply to their work. Moreover, the components for this type of instrumentation are often very cheap, especially when compared with off-the-shelf commercial solutions. The use of simple hardware and software resources allow easy modification, extension and repair of custom instruments, and the use of open-source components and systems promotes sharing of information and set up of collaborative projects. This creates a growing set of resources for the community to draw from, and build upon.

We hope that you enjoy taking part in this virtual workshop, that you learn something new and that you find it useful for your future career. Most of all, we encourage anyone who is interested in developing their skills further to sign up for the 2020 Biomaker Challenge, where you can join a team of like minded scientists and engineers to build bioinstruments for real-world applications.

About Biomaker

Biomaker was set up in 2017 and is run by the University of Cambridge Synthetic Biology Interdisciplinary Research Centre ([SynBio IRC](#)) and [OpenPlant](#), one of the UK's six National Synthetic Biology Research Centres ([SBRCs](#)).

The Biomaker Challenge provides funding for interdisciplinary team-based projects at the intersection of electronics, 3D printing, sensor technology, low cost DIY instrumentation and biology, as well as workshops and outreach events. The Biomaker project aims to build open technologies and promote development of research skills and collaborations. It taps into existing open standards and a rich ecosystem of resources for microcontrollers, first established to simplify programming and physical computing for designers, artists and scientists. These tools allow biologists to program and develop real-world laboratory tools. Further, the Biomaker projects provide a direct route for physical scientists and engineers to get hands-on experience with biological systems.

An important aspect of Biomaker is the use of open source and low cost tools and hardware, which allows greater access to fundamental knowledge and technology, encourages a collaborative environment, and supports the establishment of an open, sustainable and equitable bioeconomy.

About this Handbook

This handbook will take you step by step through the basics of biomaking, starting with understanding the parts of your 2020 Biomaker toolkit, and culminating in you using your new-found no-code programming skills to start building basic instruments.

The first three chapters of the handbook will introduce you to the different components of your Biomaker kit, and the tools you will need for the rest of the course. These chapters include:

[Chapter 2: A guided tour of the Biomaker starter kit](#)

[Chapter 3: Understanding the Arduino board](#)

[Chapter 4: XOD visual programming](#)

Chapters 5-7 of the handbook will help you get started with using the kit to set up basic functions, such as flashing an LED, or using a 4-digit display. These chapters include:

[Chapter 5: Getting started](#)

[Chapter 6: Handling simple input output devices](#)

[Chapter 7: Programming onboard devices](#)

The final chapters of the handbook will show you how to use the skills you have learned to start building biological instrumentation. These tutorials will cover how to make a simple data logger to record sensor data, how to programme a sequence of events in XOD, how to use an OLED screen and how to control the board remotely. These chapters include:

[Chapter 8: DIY data logger](#)

[Chapter 9: Additional tutorials and useful tools](#)

Learning Objectives

By the end of this course you should be able to:

- Name and understand the function of the different parts of the Arduino Rich UNO R3 board
- Name and understand the function of the 2020 Biomaker kit components
- Use XOD programming to create functional software programmes
- Use XOD programming to program and debug connected hardware
- Connect, program and use external input and output devices
- Program and use onboard devices
- Use an expansion shield to extend the capacity of the Arduino board
- Use these skills to program a functional sensor and data logger

More Ways to Get Involved

If you are enjoying the course and would like to know more about other Biomaker projects, there are several ways you can get involved.

Join the No-Code Programming Community

One of the best ways to learn new skills is to work with others, share expertise, and learn from each other's difficulties. We have set up the No-Code Programming for Biology Basecamp Site as a place to meet fellow biomakers, ask questions and help each other trouble-shoot as you work through this handbook. To access the Basecamp site you will need to sign up for a free account. Once logged in you can view this latest version of this handbook in [Docs & Files](#), join discussions and ask questions on the [Message Board](#) and chat to your fellow learners around the [Campfire](#).

2020 Biomaker Challenge

We hope to run the 2020 Biomaker Challenge in the same format as previous years, however, this will depend upon how the Covid-19 coronavirus situation develops. We will keep the participants of this course updated about the Biomaker Challenge, and will provide more information on the Biomaker website when available.

Biomaker Design 2020: Low-Cost Viral Diagnostics Hardware

We are also currently running a virtual collaborative design challenge to build reaction controllers for micro-assays and general biotechnology, focusing on Covid-19 diagnosis. With the disruption caused by the encroaching incidence of coronavirus, there has been high activity around diagnostics and assay development. A number of assay systems have proven both simple and sensitive, and are applicable to a wide variety of other targets for viral diagnosis, as well as other customised laboratory use and environmental sensing. However, there is little work on cheap, portable devices to run these assays. Most of these assays are based on nucleic acid amplification and/or selective degradation, where temperature control and optical screening of the results are required. We feel that Biomaker is in an excellent position to help push forward the design of bioinstrumentation to help conduct such assays at low cost, and in low resource environments. Please note that this is intended as an exercise in instrument design and we will not be working directly with virus assays, infective material or clinical samples. This project will be running in collaboration with New England Biolabs. You can find out more about how the project is progressing [here](#).

If you are interested in finding out more please contact Prof Jim Haseloff at jh295@cam.ac.uk.

Useful Information

Websites

Biomaker: <https://www.biomaker.org>

Collection of technical information, pointers to tutorials and software resources.

Tutorials: <https://www.biomaker.org/tutorials>

List of tutorial sessions for Biomaker.

Hackster: <https://www.hackster.io/biomaker>

Biomaker community hub used for open documentation of tutorials and projects.

XOD: <https://xod.io>

Download XOD software, libraries, documentation and forum advice.

Arduino: <https://www.arduino.cc>

Official repository of Arduino information.

Arduino Create: <https://create.arduino.cc>

Integrated resource for code and project-sharing.

Open-Smart: <https://open-smart.aliexpress.com/>

Source of hardware for Biomaker starter kit.

Sparkfun: <https://www.sparkfun.com>

Good source of practical information about microcontrollers and devices.

Seeedstudio: <https://www.seeedstudio.com>

Hardware supplier for Arduino and beyond.

Adafruit: <https://www.adafruit.com>

Good source of practical information about microcontrollers and devices.

Instructables classes: <https://www.instructables.com/classes/>

Classes in many maker skills, including electronics and 3D printing.

Fritzing: <https://fritzing.org>

Open source circuit layout and illustration.

Processing: <https://processing.org>

Software sketchbook for dynamic graphics and visual arts.

Synthetic Biology IRC: <https://www.synbio.cam.ac.uk>

University of Cambridge Interdisciplinary Research Centre

OpenPlant: <https://www.openplant.org>

BBSRC-EPSRC Synthetic Biology Research Centre

Contact Information

If at point you have any questions about the course or would like to provide any feedback please get in touch with the Biomaker team at: synbio@hermes.cam.ac.uk.

For more information, contact:

Dr. Steph Norwood, san43@cam.ac.uk

Prof. Jim Haseloff, jh295@cam.ac.uk

Chapter 2: A Guided Tour of the Biomaker Starter Kit

This chapter will introduce you to the 2020 Biomaker Starter kit used in the No-Code Programming for Biology workshop. It will help you to understand the different parts of the Arduino Rich UNO R3 board, as well as the various additional components supplied in the starter kit.

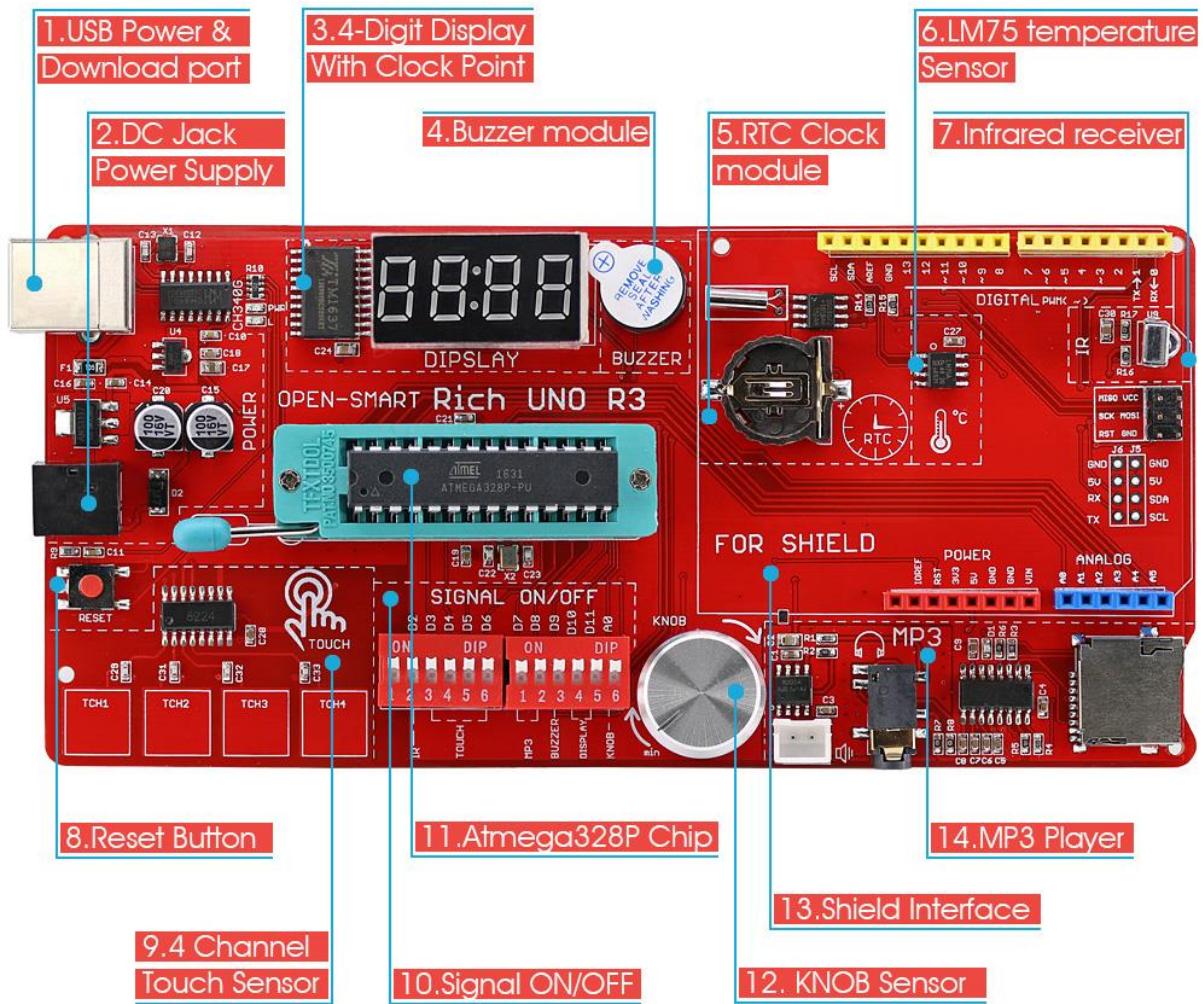
The 2020 Biomaker Starter Kit

The 2020 Biomaker starter kit consists of an extended Arduino board, a collection of electronic components, a small prototyping board and a programmable display. It is based on the Open-Smart Rich UNO R3 board, which contains a variety of embedded components, and is accompanied by a variety of electronic components and sensors which are useful for monitoring and programming of biological systems. The board is Arduino UNO compatible, and can be programmed directly from the visual programming software, XOD.

In addition to the starter kit you will also require a laptop or desktop computer with a type A USB port (or a relevant converter), and a download of the XOD programming software, which can be found on the XOD website: <https://xod.io/>. This section outlines the basics of your Biomaker starter kit. You can find more information and specifications for the Arduino Rich UNO R3 board and other kit components in the [annex](#) of this handbook.

Arduino Rich UNO R3 Board

The Rich UNO R3 board is an expansion of a simple Arduino UNO microcontroller board. Arduino is an open-source electronics company, which makes openly available programming software and low-cost hardware to allow anyone to get started making their own interactive electronics projects. In addition to the basic electronic components included in a simple Arduino board, the Rich UNO R3 board included in this kit also includes a number of integrated components and sensors, such as a four digit display (3), buzzer (4) and temperature sensor (6). Importantly, the connections to the board's embedded components can be turned off using a DIP switch (10), meaning that each part of the board can be independently switched on and off. Below is a quick explanation of each part of the board. [Chapter 3: Understanding your Arduino Board](#) will explore each of these components in more detail. You can also find specifications for the board in the [annex](#) of this handbook.



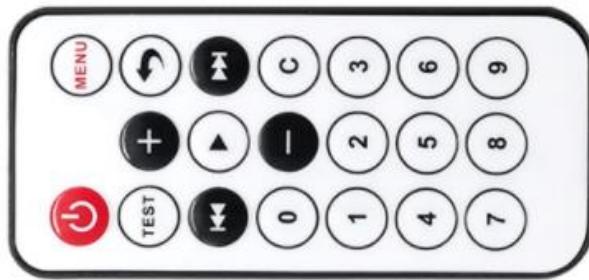
Arduino Rich UNO R3 Board

- 1. USB Power and Download Port:** type B USB port. Used to connect the board to a computer, or provide a power supply.
- 2. DC Jack Power Supply:** can be used to supply power via a DC connector rather than the USB connector. Also contains a resettable fuse to protect both the DC and USB power ports.
- 3. 4-Digit Display with Clock Point:** display screen, which can be used to display a digital clock.
- 4. Buzzer Module:** inbuilt piezoelectric buzzer.
- 5. RTC Clock Module:** high precision clock that allows the board to keep everything running at the correct time.
- 6. LM75 Temperature Sensor:** inbuilt temperature sensor. Can also be used to set a protection temperature, allowing it to monitor the temperature of the board and shut down if the board overheats.
- 7. Infrared Receiver:** can receive infrared signals from the remote control, allowing the board to be controlled remotely.
- 8. Reset Button:** Allows you to restart the programme without unplugging and plugging back in again.

9. **4-Channel Touch Sensor:** inbuilt touch sensor. Each sensor (TCH1-4) acts as a touch sensitive button.
10. **Signal ON/OFF:** onboard DIP switch. Allows you to turn each of the inbuilt components on and off independently.
11. **ATmega328P Chip:** this is the board's microcontroller. It acts as the brains of the board, instructing each of the other components what to do.
12. **KNOB Sensor:** senses the rotation angle of the knob. Can be used to adjust the volume of the MP3 player, adjust the brightness of the 4-digit display etc.
13. **Shield Interface:** used to plug in expansion shields. These allow you to expand the functionality of your board, and can be stacked on top of each other to allow endless possibilities.
14. **MP3 Player:** high-quality MP3 music chip. Allows you to plug in a microSD card with MP3 files, and play them. It can also send commands, for example to switch songs, or change the volume. Also includes a headphone port.

Kit Components

For the purpose of making biological instruments, we often need to use sensors (such as water or gas sensors) and/or actuators (moving components, e.g. motors for making robotics). The Arduino Rich UNO R3 board is available from [Open Smart](#) as part of a kit that includes a wide variety of sensors and actuators. Below is a quick explanation of each component in this kit. You can also find specifications for the kit components in the [annex](#) of this handbook.



IR Controller



Speaker with XH-2.54 Port



USB Cable



Battery

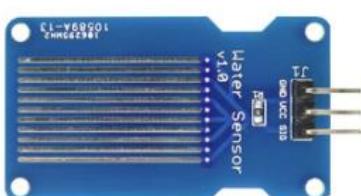


MicroSD Card

Basic Arduino Rich UNO R3 Kit Components

- **IR Controller:** infrared remote control. Used to send signals to the infrared sensor and control the board.
- **Speaker with XH-2.54 Port:** speaker that can be plugged into the MP3 module on the board, allowing you to play music and MP3 files.
- **USB Cable:** used to connect the board to a computer. This allows you to programme the board, and also supplies power.

- **Battery:** can be found plugged into the clock module. Powers the inbuilt clock.
- **MicroSD Card:** can be used to store files for the board to read or write to, e.g. MP3 files allowing the board to play music.



Water Sensor



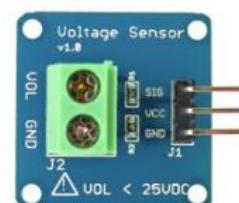
NTC Sensor Line+Adapter module



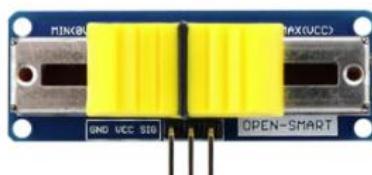
Ultrasonic Sensor



Touch Sensor (T)



Voltage Sensor



Slide Potentiometer



Rocker Switch



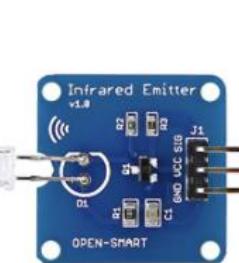
Vibration Motor



PIR motion Sensor



Light Sensor



Infrared Emitter

Commonly Used Sensor Modules

- Water Sensor: moisture sensor. Can be used to detect raindrops, or measure water levels.
- NTC Sensor Line and Adapter Module: water resistant temperature sensor. Can be used to measure the temperature indoors, outdoors, in greenhouses, in water etc.
- Ultrasonic Sensor: can be used to measure distances using an ultrasonic wave.
- Touch Sensor: touch sensor with three different modes: (i) sends signal when touched, (ii) sends signal when not touched and (iii) stops sending signal when not touched for 12 seconds.
- Voltage Sensor: can measure voltage over a wide range of values.
- Slide Potentiometer: moving the slider from side to side will change the voltage. Can be used for volume control, or as a lighting regulator etc.
- Rocker Switch: simple on/off switch.

- Vibration Motor: can act as a non-audible indicator in place of a buzzer. Will vibrate like a mobile phone.
- PIR Motion Sensor: infrared motion sensor. Often used for sensing the movement of humans or other animals.
- Light Sensor: senses the light intensity in the environment.
- Infrared Emitter: emits an infrared signal. can be used to wirelessly transfer information or act as a remote switch.



Dupont Line 1*40P



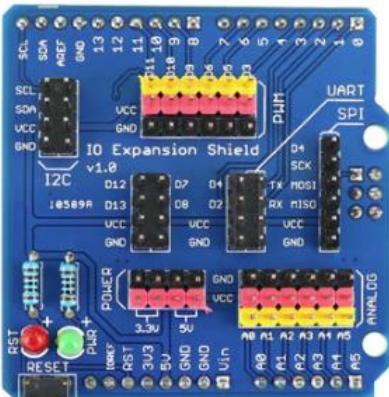
MicroSD Card Adapter



4-Digit Display



Passive Buzzer



IO Expansion Shield



LED Bar module



Eagle Eye LED Module



I2C 1602 LCD module

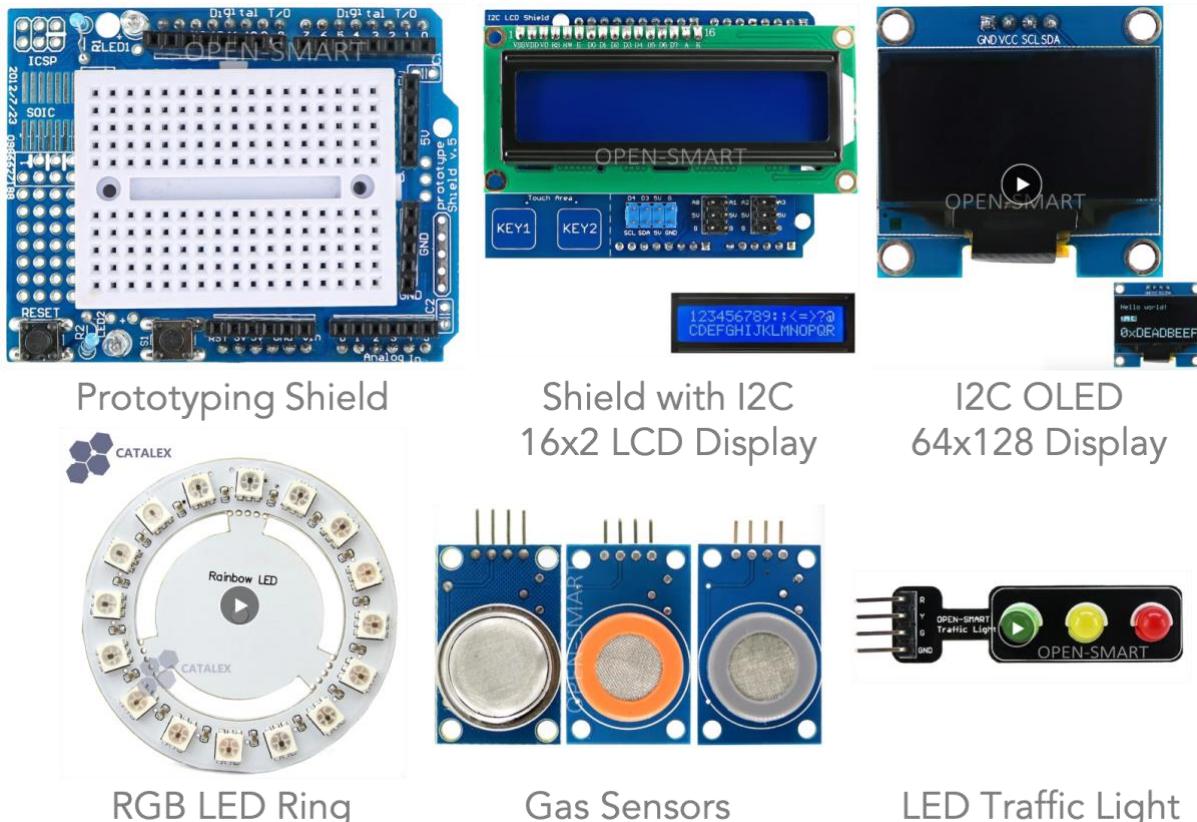
IO Expansion Shield, Displays and Adapter

- Dupont Line 1*40P: series of hook-up wires used to connect components to the board. The wires can either be separated and used individually, or left connected.
- MicroSD card Adapter: used to read and write files to a microSD card.
- 4-Digit Display: similar to the inbuilt 4-digit display. Can be used to display a clock, numbers or letters.
- Passive Buzzer: similar to the inbuilt piezoelectric buzzer.
- IO Expansion Shield: extends the board, allowing you to add additional components.
- LED Bar Module: LED board with 8 LED lights.
- Eagle Eye LED Module: very bright LED light.

- I2C 1602 LCD Module: LCD screen that can display two lines and 16 characters of text. Can be used to display read-outs etc.

Additional Components

We have also included several additional components in the 2019-2020 Biomaker kit. Below is a quick explanation of each additional component. You can also find specifications for the kit components in the [annex](#) of this handbook.



Additional Components

- Prototyping Shield: expands the capacity of the board. Can be used to assemble custom circuits for prototyping. It allows direct mounting of soldered components on the board, or connection through a mini breadboard. It also has two inbuilt LEDs and two switches.
- Shield with I2C 16x2 LCD Display: LCD screen that can display two lines and 16 characters of text. This module also included two touch-sensitive keys, and several additional pins to input extra components.
- I2C OLED 64x128 Display: high-quality OLED display screen.
- RGB LED Ring: series of 16 LEDs that can display different colours at different times.
- Gas Sensors: includes a smoke sensor that can detect liquefied petroleum gas, natural gas, coal gas and smoke; an alcohol sensor that can detect the concentration of ethanol vapor; and a carbon monoxide sensor that can detect carbon monoxide levels.
- LED Traffic Light: red, yellow and green LED lights. Allows you to simulate traffic lights.

Making your own kit

We have tried to supply as many participants as possible with Biomaker starter kits. However, if you would like to put together your own kit, all of the components are available from Open Smart. Below is a list of components and their prices (correct as of 23/03/2020). We have also noted where components are essential (E); used in some tutorials, but not essential (S); or generally useful but not required for tutorials in this course (U).

Item	Cost	Essential?	Link
Arduino Rich UNO R3 Kit (B) with MP3 and Sensors	\$ 25.93	E	https://www.aliexpress.com/item/32822090848.html
Prototyping shield	\$ 1.55	U	https://www.aliexpress.com/item/32419802380.html?spm=2114.12010615.8148356.3.64622f8cYIX97w
Shield with LCD Display	\$ 3.70	U	https://www.aliexpress.com/item/32349041767.html?spm=2114.12010615.8148356.2.540d401flklHBc
RGB LED Ring	\$ 2.43	S	https://www.aliexpress.com/item/32461822160.html?spm=2114.12010615.8148356.18.61221266in4Pdy
OLED Display	\$ 3.67	S	https://www.aliexpress.com/item/32657357817.html?spm=2114.12010615.8148356.7.77cd44ceJisJJ6
Gas Sensors	\$ 3.51	U	https://www.aliexpress.com/item/32649677074.html?spm=2114.12010615.8148356.3.1de349dbawoocu
LED Traffic Light	\$ 0.75	S	https://www.aliexpress.com/item/32820546930.html?spm=a2g0o.store_home.productList_1691794.pic_6
Total	\$ 41.54		Plus Shipping

If you do not want to purchase your own kit then some of the tutorials in this course can be completed using XOD simulation only. XOD provides “watch nodes” that will allow you to see if your program is working, without using software. Some useful information about using watch nodes to simulate in XOD can be found on the XOD website:

<https://xod.io/docs/guide/debugging/>.

This will not provide you with experience working with hardware, but will allow you to follow along with the course and start understanding the basics and underlying logic of biomaking.

Further Information

More information and specifications for the Arduino Rich UNO R3 board and kit components can be found in the [annex](#) of this handbook. You can also download the documents below from Open Smart.

Board manual:

<https://www.biomaker.org/s/Rich-UNO-R3-User-Manual.pdf> (4.9 MB PDF)

Board circuit diagram:

<https://www.biomaker.org/s/OPEN-SMART-Rich-UNO-R3-Schematic.pdf> (805 KB, PDF)

MP3 Player Manual:

<https://www.biomaker.org/s/Serial-MP3-Player-A-v10-Manual.pdf> (665 KB, PDF)

Chapter 3: Understanding the Arduino Board

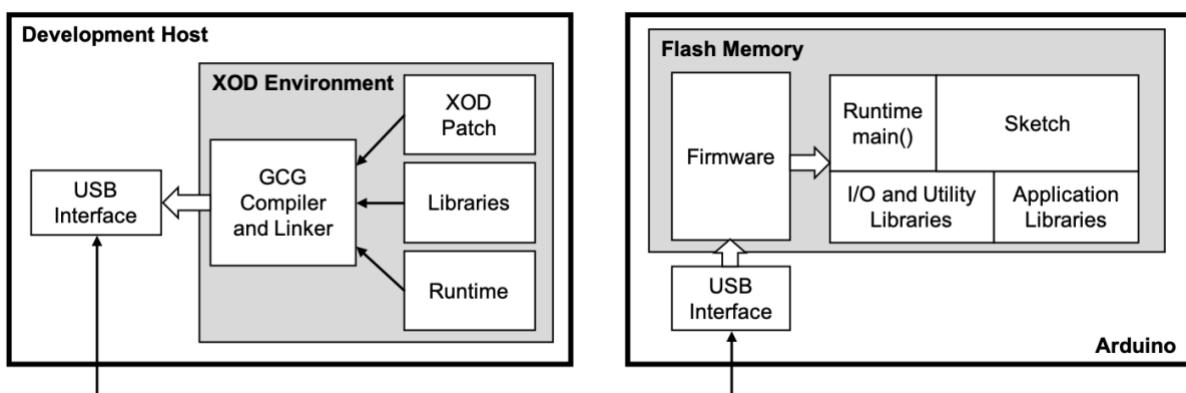
This chapter will walk you through how to use the Rich UNO Board, including how to control the board and where to plug in additional components. It will dive into some concepts of electronics.

Get to Know your Board!

In the last chapter we took you through the different parts and inbuilt components of the Arduino Rich UNO R3 board, but how do you control these components, and how do you add and connect the additional sensors and actuators supplied in the kit? In this chapter we discuss how to use your Arduino board, where to plug in your components, and the characteristics of each of the board's connecting "header sockets".

Controlling your Arduino Board

In order to tell the Arduino board what to do you will need to plug it into a computer. This is referred to as the development host, as it is where you will write and develop the programme you want to install on the board. The development host interacts with the board via a USB connection. This connection supplies power to the board and sends instructions from your computer to the memory of the board, where it is stored. Once the programme has been written on your computer and transferred to the board, the board can be disconnected and will be able to run the desired programme independently of your computer (it will need an alternative power supply). The board can be programmed to perform a multitude of different tasks depending on what components you add and what programs you install.



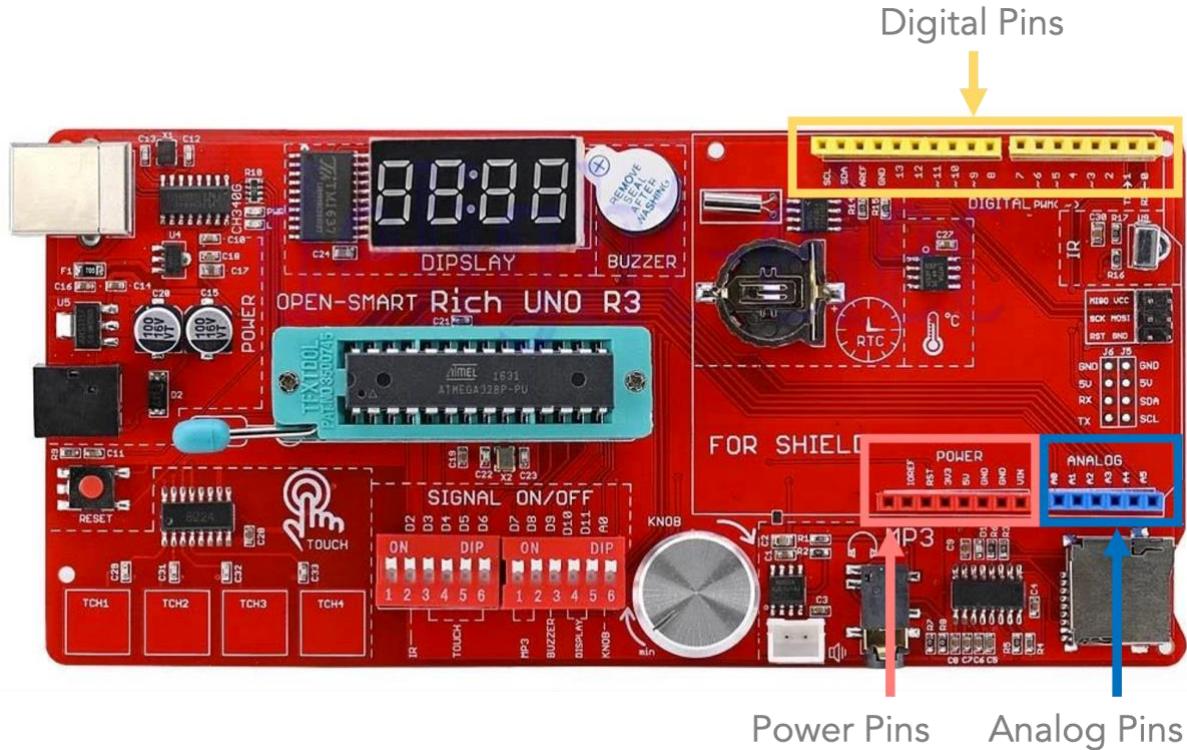
Block Diagram of the Development Environment and Arduino Board

ATmega328 Microcontroller Chip

You can think of the microcontroller chip itself as the "brains" of the board. The chip used in the Arduino Uno is the ATmega328, made by Atmel. It's the large, black component in the center of the board. This chip is known as an integrated circuit, or IC. This chip can come in different forms, referred to as packages.

Header Sockets for Connecting External Hardware

The microcontroller socket connects all the legs of the ATmega328 microcontroller chip to other sockets, referred to as header sockets, which have been arranged around the board and labeled for ease of use. They are the yellow, red and blue sockets located on the right-hand side of the board. These are divided up into three main groups: power pins (red), analog input pins (blue), and digital pins (yellow). These header sockets are where you can plug in additional components. You can also plug in an expansion shield, which allows you to increase the board's capacity for additional components. Shields can be stacked on top of one another, to create any combination of different functions.

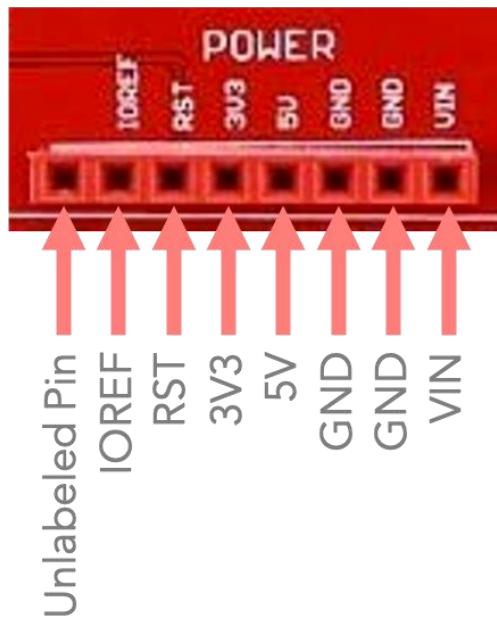


Header sockets on the Arduino Rich Uno R3 board

Each of these pins can transfer a voltage between the board and added components. This can either be sent as output or received as an input. This same process of sending and receiving electrical signals is going on inside modern computers, but difficult to access. The Arduino provides direct access to input and output signals, and a large library of open source software support is available for these kinds of interfaces with different devices. Furthermore, XOD provides a visual programming interface that allows simple access to these resources.

If you look at these pin connections you will see a few letters next to each pin. These are indications of what the pins can do, but they are not very explicit to someone who does not know much about electronics. In the rest of this chapter, we will decrypt them one by one. By the end, you should be able to understand what to plug where, and why, for your future circuits!

Power Pins



Arduino Rich UNO R3 Board Power Pins

GND, 5V and 3.3V

All of your hardware parts have a minimum of 3 pins to plug in the board. Those are VCC, GND and Signal. Basically, the board is powered by either a computer or an external power supply. Current flows through the hardware parts and makes the board work. VCC (usually 5V, but sometimes 3.3V for some components) is where the power comes from. GND is where it ends (It's basically the point of 0V). Current will always flow from VCC to GND and activate whatever is in between. The signal pin inbetween is the pin through which the board communicates with the device.

VIN

This can input power into the board when using an external power source. You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

IOREF

As its name indicates, the IOREF is the Input-Output Reference and is used to send the correct voltage to peripheral devices. On our board, this voltage is 5V, on other similar devices it may be a different voltage (it is 3.3V on the Arduino DUE for instance). This pin will not do anything if you plug a device into it.

RST

This reset pin can be connected to a button to reset the board remotely. This is particularly useful if the board is part of a device and is either not very accessible or hidden away in a box.

Unlabelled Pin

This unlabelled pin has no function, but is there for compatibility reasons. It is there to ensure that the board would stay compatible with next generation shields. It will not send or receive data.

ANALOG Pins

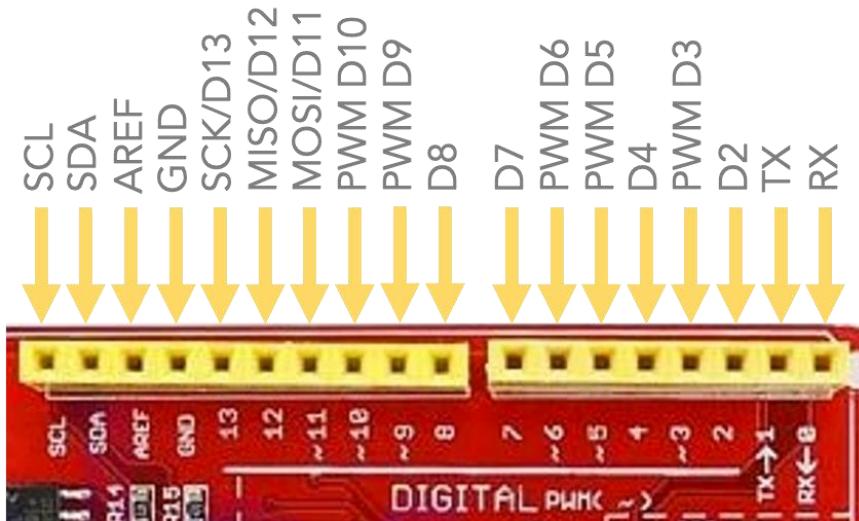


Arduino Rich UNO R3 Board Analog Pins

Pins A0 to A5

Analog pins are input pins. They transmit to the board a signal that comes from a sensor and is continuous (meaning it can have an infinite amount of values within a given range). Most sensors are analog sensors and can send an infinite amount of values to the board. The board then converts them in values ranging from 0 to 1023 to communicate with the digital pins. Most sensors in the kit are analog sensors which means that they provide measured values as continuous changes in voltage variations. You will use these pins to connect analog sensors. However, switches that go either on or off, are examples of digital inputs.

DIGITAL Pins



Pins D0 to D13

Digital pins have a discrete and finite amount of values that they can deal with. Digital pins can receive inputs and produce outputs (unlike analog pins which can only receive inputs). Digital pins are used for most non-sensor components. Digital pins can also have other abilities, which we will explore below.

PWM : Pins D3, D5, D6, D9, D10 & D11

Pulse-Width-Modulation (PWM) is a way to translate analog signals into digital signals that the microcontroller can interpret and use. This is what allows you to dim the brightness of an LED with a potentiometer, even though a potentiometer is an analog input and an LED a Digital output. These pins are marked on the board with a ~ before the number.

I2C: Pins SCL and SDA

I2C or "Inter-Integrated Circuits" are a standard to connect external devices to a microcontroller using only two lines in addition to the GND and VCC cables. The lines are the "Serial Clock Line" (SCL) and "Serial Data Line" (SDA).

To understand how these pins work, it is important to understand the concept of digital clocking. When signals are sent between devices they need to be transmitted at regular intervals to prevent information being lost. This is achieved by synchronising the transmission of data with a regular clock signal, so that data is only sent and received as the clock "ticks". There are several ways of synchronising data, including the UART and SPI protocols (discussed below), but the I2C protocol has the advantage of allowing you to connect multiple devices at once with only two pins. The board can then distinguish between these devices by giving them each a unique digital address.

SCL

Serial Clock Line, or SCL is part of the I2C bus. This line provides the clock signal by which data transfer is timed.

SDA

Serial Data Line or SDA is the other half of the I2C bus. This line sends the data in the form of “bits”. When the SCL line clock “ticks” the SDA sends either a high or low voltage signal, which represents either a 0 or a 1. This binary data is then interpreted by the board.

UART: Pins TX and RX

UART stands for Universal Asynchronous Receiver-Transmitter. It is a component that communicates with the computer without using a clock to time the data. Instead the instructions all have a start and a stop bit for the microcontroller to interpret them correctly. It is particularly useful for debugging, because it can act as a tether between your Arduino and computer, allowing you to monitor the board's functions and check for errors. UART only allows direct communication between two devices, and uses the TX and RX lines.

TX

TX is the transmitter line, which can send signals to one other device.

RX

RX is the receiver line, which can receive signals from one other device.

SPI: Pins MISO/D12, MOSI/D11 and SCK/D13

Serial Peripheral Interface (SPI), like I2C, is a synchronous protocol, which relies on a clock to organise information. It is used to communicate with peripheral devices or with another microcontroller. SPI relies on a master and slave system, meaning that the board you are using is a “master” able to send instructions to one or several “slave” devices, which will follow the master’s instructions. To connect a device via SPI you need four lines: MISO, MOSI, SCK and SS. The MISO, MOSI and SCK lines can be used to connect your master to multiple slaves, but each slave also needs its own individual Slave Select, or SS line. This line is used to identify each connected device separately.

MISO/Pin 12

Master In Slave Out. This line sends information from the slave to the master.

MOSI: Pin 11 or MOSI on the extension board

Master Out Slave In. This line sends information from the master to the slaves.

SCK: Pin 13 or SCK on the extension board

Serial Clock. This line sends the clock pulses to synchronise the data.

AREF Pin

Analog reference pin. Can be used to provide an analog reference voltage. This pin is rarely used.

GND Pin

There is also an additional GND power pin in the digital pins head socket.

Pin Connections

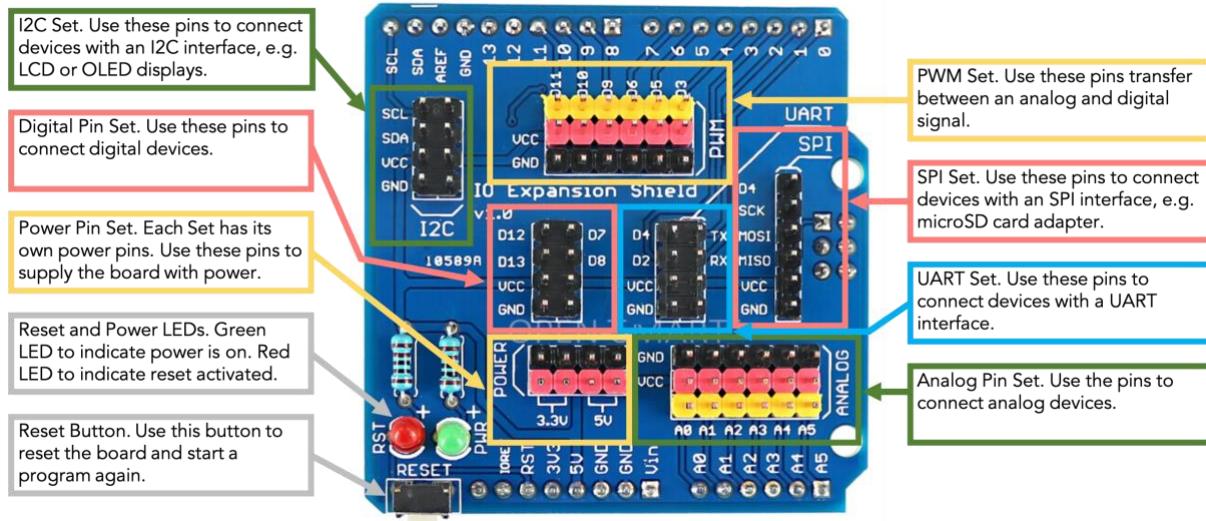
On your Arduino Rich UNO R3 board, several of the pins are already connected to components on the board. You can still plug external devices into these pins, but when you

do this you may need to turn off the onboard devices to avoid sending signals to the wrong component. You can turn any of the onboard devices off using the DIP switches (SIGNAL ON/OFF) at the bottom of the board. The onboard devices and the pins they are connected to are listed below.

Pin	Connected Device
A0	KNOB potentiometer
A4	Acts as the SDA for I2C communication with the RTC clock module (address 0x68), the LM75 temperature sensor (address 0x48), and the LCD display screen provided in the kit (address 0x27).
A5	Acts as the SCL for I2C communication with the RTC clock module (address 0x68), the LM75 temperature sensor (address 0x48), and the LCD display screen provided in the kit (address 0x27).
D2	Infrared receiver
D3	Touch sensor channel 1 (TCH1)
D4	Touch sensor channel 2 (TCH2)
D5	Touch sensor channel 3 (TCH3)
D6	Touch sensor channel 4 (TCH4)
D7	MP3 player
D8	MP3 player
D9	Buzzer
D10	4-digit display
D11	4-digit display

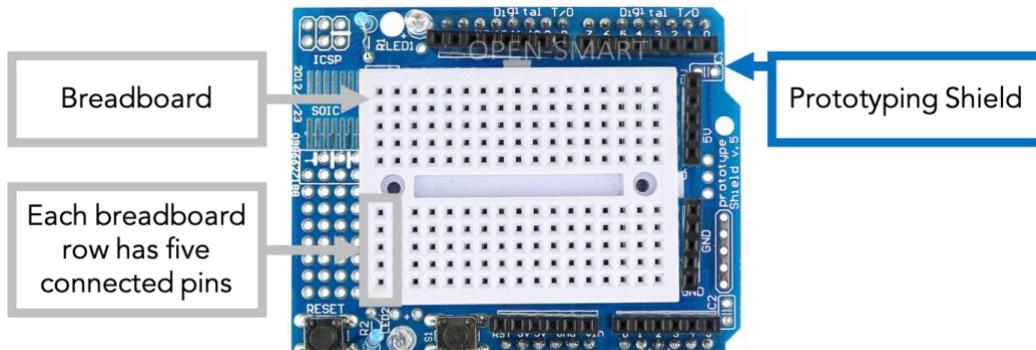
The remaining pins are available for you to connect any additional components you require. You can also use these pins to plug in shields, such as the two LCD display screen shields, the expansion shield or the prototyping shield provided in this kit. Expansion and prototyping shields can be especially useful, as they allow you to connect a wide variety of external components to the board.

The expansion shield can be used, alongside the female-to-female hook-up wires provided, to connect additional components to the board. It is split into several sets of pins based on their pin and interface types.



Open Smart Expansion Shield

The prototyping shield can be used either by soldering components to the shield, or by using a breadboard. The breadboard is the white rectangular component provided in the same package as the prototyping shield. Breadboards provide an easy way to build and test circuits, without having to solder them in place. Components can be plugged directly into the holes in the breadboard, and will be connected to any other components plugged into the same row. You will need female-to-male or male-to-male leads to connect devices on the breadboard to the Arduino (not provided).



Open Smart Prototyping Shield and Breadboard

Further Information

If you would like to understand more about the electronics concepts behind the Arduino board there is some excellent information for beginners on the [Sparkfun](https://learn.sparkfun.com/tutorials/serial-communication) website. Several tutorials relevant for this chapter are listed below.

Introduction to serial data transfer: <https://learn.sparkfun.com/tutorials/serial-communication>

Introduction to I2C: <https://learn.sparkfun.com/tutorials/i2c/all>

Introduction to SPI: <https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi>

Introduction to breadboards: https://learn.sparkfun.com/tutorials/how-to-use-a-breadboard?_ga=2.267526030.1277465979.1584962875-1288494135.1584962875

Chapter 4: XOD Visual Programming

This chapter provides an outline of the XOD graphical programming environment. This software uses a visual data flow model for programming Arduino microcontrollers, and will be used for all of the subsequent tutorials in this handbook.

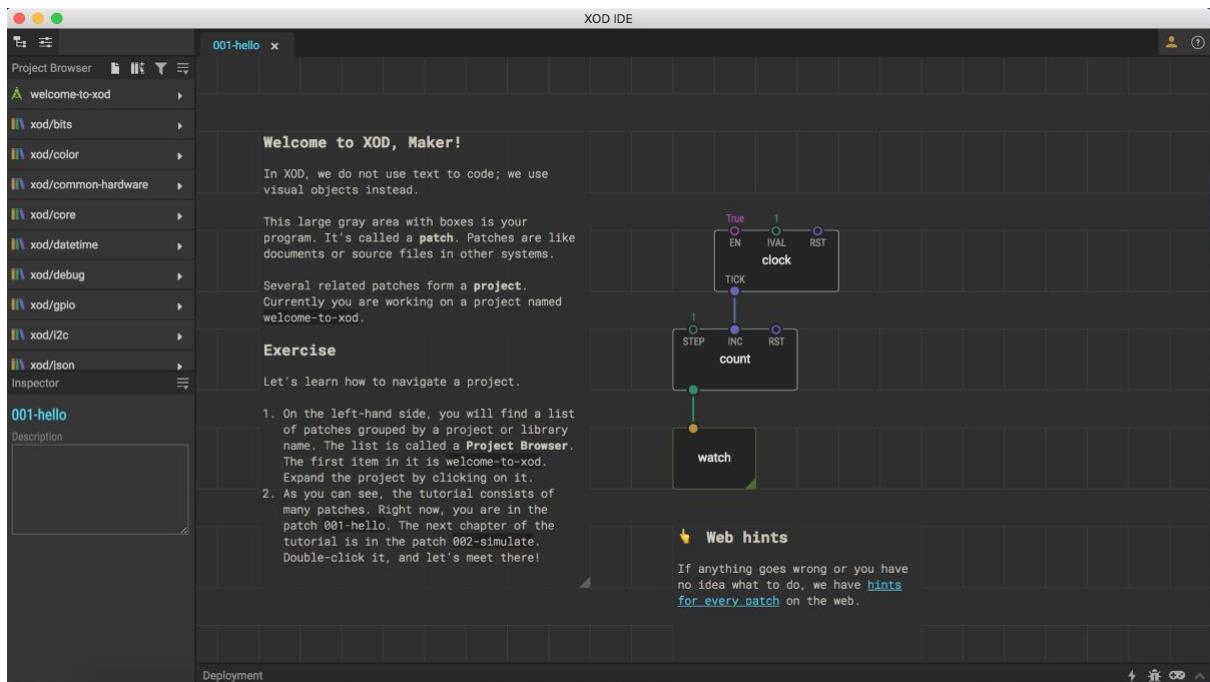
XOD: Visual Programming for Biomaker

XOD is an open source software development environment that can be used to programme the Arduino microcontroller board. It uses a visual programming tool that represents hardware and computing elements as nodes that can be wired together to allow data flow between the objects. XOD allows a hierarchical and dataflow driven approach, avoids the complexities of written code and syntax, and can be used to directly programme Arduino boards. We think that this provides a simple way for non-programmers (for example biologists, or other scientists with little formal programming training) to develop useful skills and understanding - without needing to deal with the complications of programming languages. We have chosen XOD as an accessible tool for Biomaker training and device development. The software can be downloaded from: <https://xod.io>.

The XOD IDE (Integrated Development Environment) is available in two different formats, a desktop client and a web-based client that you run in your browser. We generally use the desktop IDE. Both implementations have similar appearance and functionality, but the desktop version allows compilation on a local computer without quota limitations. There are versions of the desktop IDE for Windows, Mac OSX and Linux.

Patches, Nodes and Links

The first time you open up either the desktop or web-based IDE a default project called “welcome-to-xod” will be opened. This introductory project provides a series of tutorial “patches”. A patch is the working area for a XOD program. It is similar to a document or source file in other systems, but instead of text code the patch is built with “nodes”, which are the basic elements in a XOD program.



“Welcome to XOD” screen

A node can represent many different things: an electronic component like an LED, a sensor like the LM75 temperature sensor, a logic function like AND, OR or NOR, a pulse source like a Square Wave or Sine Wave generator, a mathematical function like multiplication or addition, a conversion function like metres to feet or a Boolean function. Each node is represented by a rectangular box that has one or more circular connections on the top and bottom. These connectors are called “pins”.

“Input pins” are located on the top side of each node, while those on the bottom are “output pins”. The pins on a node are like variables and can contain parameter values. The values can be left at their default values, or selected and set using the “Inspector”, or receive new values via connection to the output pins of other nodes.

The pins on a node can have different data types, represented by colors.

- **Green** Pins represent **numbers**.
- **Blue** pins represent **pulses**.
- **Violet** pins represent **boolean values**.
- **Orange** pins represent **strings**.

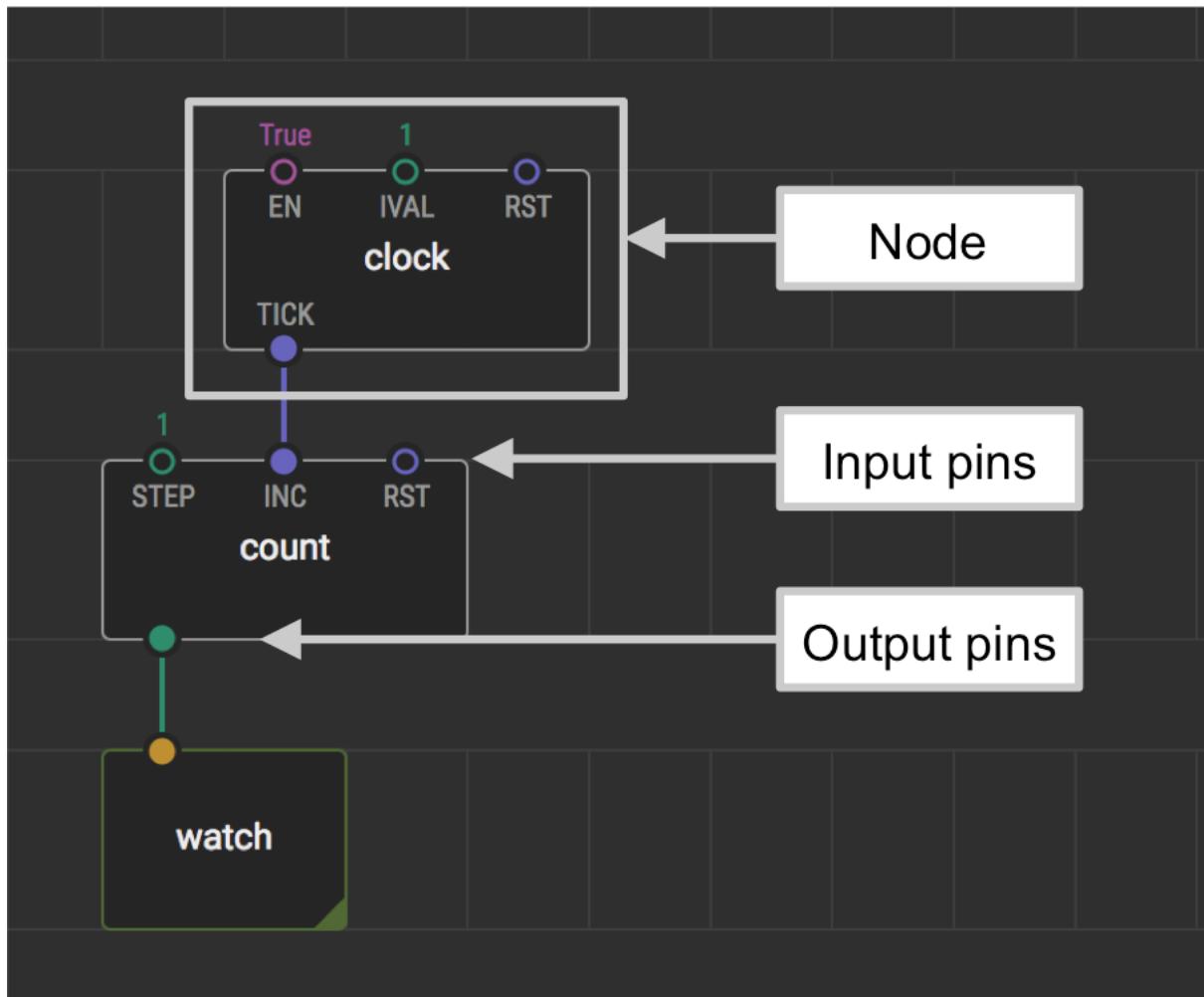
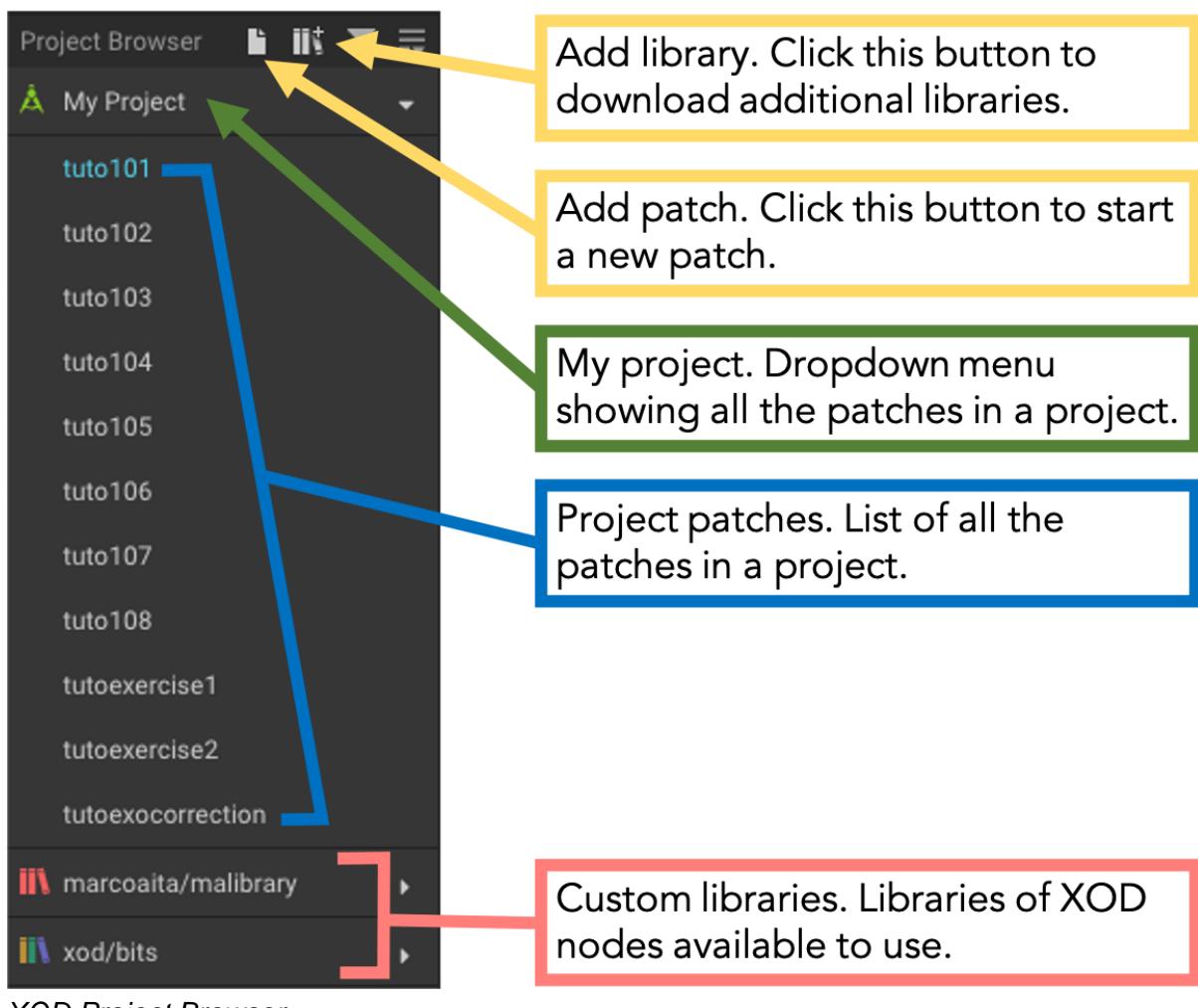


Illustration of a XOD patch

A XOD program consists of nodes connected together in one or more patches. New nodes are selected in the Project Browser, and dragged onto the patch area. If you know the name of the node you want to add, you can double-click on a blank area of the patch or press the "i" key on your keyboard. This will bring up a search box where you can type in the name of the relevant node. If the search is successful, highlight the node in the results and hit "enter" to put the node onto the patch.

In XOD “Links” are the lines used to connect nodes to each other. A link runs from an output pin on one node to the input pin on another node. You create a link by clicking on a pin on one node (this creates the starting point of the link), you then drag it to another pin on another node. You can start creating a link by clicking on either an input or an output pin. XOD is smart and won’t let you link a pin to another pin if it doesn’t make sense or if the data types are incompatible. The link color is determined by the data type of the output pin in the link. Linking nodes is a lot like wiring elements in an electronic circuit. In fact a XOD program really looks more like a wiring diagram than anything else.

Project Browser



XOD Project Browser

The Project Browser is a section on the top left side panel of the XOD IDE. It essentially allows you to manage all of the patches in your project and to add nodes. It consists of the following sections:

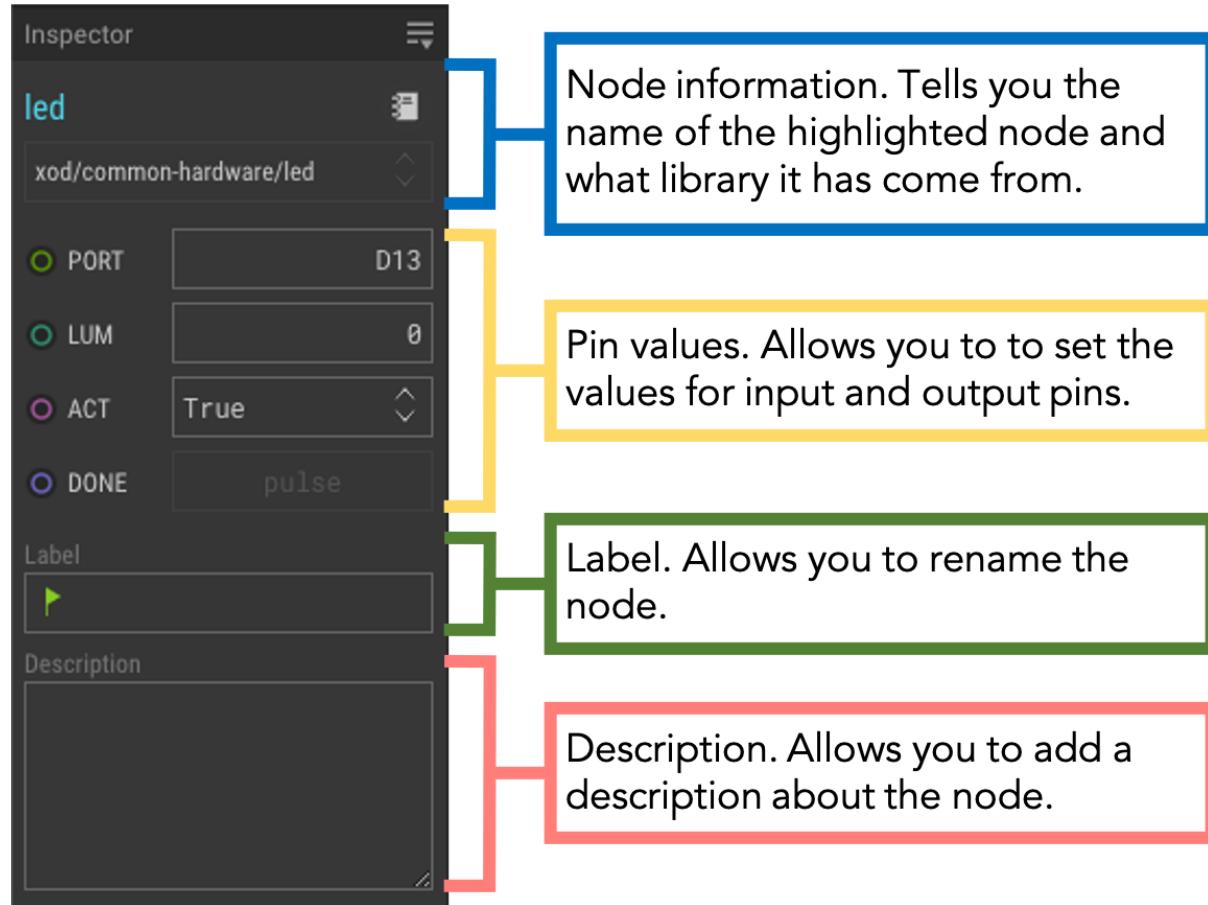
Project Patches: A list of all of the patches used in your project. You can open, rename or delete each patch or even drag it onto another patch.

Custom Libraries: These are libraries of XOD nodes available for you to use. Different libraries contain different types of nodes. For example, some libraries contain basic mathematical functions, whilst some have been built specifically to help you work with certain hardware components. The software comes with a series of useful libraries already installed, but there are a wealth of other libraries available for you to download on the XOD website: <https://xod.io/libs/>. Anyone can create and publish a library, and there is already a large number of community-generated libraries allowing you to perform almost all of the functions you could require. The pre-installed libraries and the types of nodes they contain are listed below.

- [xod/bits](#) - Low-level bits and bytes operations
- [xod/color](#) - Library to work with color

- [xod/common-hardware](#) - Hardware drivers for popular and simple peripherals
- [xod/core](#) - The very basic nodes of XOD
- [xod/datetime](#) - Date and timestamp operations
- [xod/debug](#) - Debug nodes for XOD
- [xod/gpio](#) - Nodes of XOD to deal with GPIO (hardware pins)
- [xod/i2c](#) - I²C (aka I²C, IIC, TWI) bus interfacing
- [xod/math](#) - Nodes of XOD for basic mathematical operations
- [xod/net](#) - No description
- [xod/stream](#) - No description
- [xod/uart](#) - Provides constructors and Nodes to interact with UARTs (Software, Hardware, USB) in XOD.
- [xod/units](#) - Units of measurement conversions
- [xod-dev/ds-rtc](#) - This library operates DS1302/DS1307/DS3231 based breakout RTC boards
- [xod-dev/esp8266](#) - Support for ESP8266 as a slave module
- [xod-dev/esp8266-mcu](#) - Support for ESP8266-based MCUs.
- [xod-dev/pn532-nfc](#) - Support for RFID/NFC modules based on a PN532 chip.
- [xod-dev/sharp-irm](#) - Nodes to read analog infrared range meters by Sharp (GP2Y0A) and convert the signal to distance values.
- [xod-dev/w5500](#) - Support for ethernet shields that use Wiznet W5500 chipset.

Inspector



XOD Inspector

The Inspector is located on the bottom left side of the XOD IDE. It allows you to view and modify the properties of nodes. If you highlight a node the Inspector will display its current properties. Each pin in the node will have a property. You can modify the properties of most input pins if they are unconnected. If a pin is connected to the output of another node then the pins property will be controlled by that node and you won't be able to modify it. You can rename a node so that it makes sense in your project. This is useful if you have several nodes of the same type, for example several LED nodes. You can also add a description to each node if you wish.

Quick Help

led
xod/common-hardware/led

Drives a generic single color dimmable LED. The conversion between luminance and LED power is biased so that change in brightness on range 0...1 is perceived more or less uniformly by a human. Possible errors: –
Invalid port

Inputs:

PORT port
Board port number the LED is connected to. The port should be capable of generating PWM signal.

LUM number
Luminance (brightness) value in relative units [0.0, 1.0].

ACT boolean
Updates the luminosity of the LED on incoming value change while 'ACT' is true.

Outputs:

DONE pulse
Fires on writing complete

Node information. Tells you the name of the highlighted node and what library it has come from. Also provides a brief summary of the node's function.

Input information. Provides a brief description of each input pin.

Output information. Provides a brief description of each output pin.

XOD Quick Help

The Quick Help section is on the top right of the XOD IDE. By default it is hidden but you can toggle it on by clicking on the “question mark” icon in the top right corner. Quick Help gives you information about the highlighted node and its pin functions and data types. It's very useful and We would recommend keeping it open at all times. Between the Inspector and the Quick Help you can usually get enough information to work with any XOD node.

Further Information

Introductory Lessons for XOD

If you would like to practice with XOD before starting the Biomaker tutorials, there are a variety of introductory lessons available at: <https://xod.io/docs/tutorial/>. Please note that you do not need to complete these lessons before continuing with the No-code Programming Handbook.

To get started with XOD lessons, simply install the cross-platform XOD software, assemble and plug in the hardware, and you can get started directly. Many of these introductory tutorials use the original Arduino UNO board, however, they can be easily adapted for use with the Rich UNO R3 board. In addition to these introductory exercises with minimal hardware, this handbook will provide exercises with extended componentry that we will build on throughout the course.

- [Installing and running XOD](#)
- [Required hardware](#)
- [Hello](#)
- [Upload to Arduino](#)
- [Pins, data, and the Inspector](#)
- [Fractional numbers and PWM](#)
- [Wiring configuration](#)
- [Adding nodes](#)
- [Node labels](#)
- [Constant nodes](#)
- [Input from a potentiometer](#)
- [Doing math](#)
- [Controlling servos](#)
- [Accessing help](#)
- [Mapping values](#)
- [Adjusting map range](#)
- [Buttons](#)
- [Logic nodes](#)
- [Reading lightness](#)
- [Comparing numbers](#)
- [If-else branching](#)
- [Smoother changes](#)
- [Pulses](#)
- [Clock](#)
- [Pulse counting](#)
- [Flip-flop](#)
- [Using multiple timelines](#)
- [Showing text on LCD](#)
- [Displaying sensor values on LCD](#)
- [String concatenation](#)

Chapter 5: Getting Started

This tutorial will provide a step-by-step guide to getting started with your Biomaker kit. You will learn how to connect simple parts to your board, and use the XOD programming software to control components, including the onboard buzzer and touch buttons, and a bright external LED.

Introduction

This tutorial will introduce you to the basics of adding simple hardware to the Arduino board and using XOD graphical programming to control its functions. We will show you how to program the onboard LED and buzzer, how to connect an external LED, how to use touch buttons, how to load an external library and how to simulate your program if you do not have the required hardware. After this tutorial, you should gain a basic understanding of how to use XOD to program your Arduino Rich UNO R3 board and its components.

Objectives

- Install the software and USB drivers to allow XOD graphical programming of the Biomaker starter kit.
- Test the connections and software installation by assembling simple patches for the Rich UNO R3 Arduino board.
- Enter parameter values for XOD nodes and download the code to the board to flash an onboard LED.
- Plug a bright external LED from the starter kit to the same port and test.
- Revise the graphical program to include input from an onboard touch switch.
- Use the XOD *watch* and *tweak* nodes for real-time debugging.
- Learn about conditioning signals by connecting the XOD *not* and *flip-n-times* nodes.
- Divert the output from the LEDs to the onboard piezoelectric buzzer.

Requirements

- Computer running MacOS, Windows or Linux
- Arduino Rich UNO R3 board
- Eagle Eye LED module
- Hook-up wires/Dupont line

Step-by-Step Guide

Step 1: Install the USB Driver

A USB driver is a file or a group of packets that allow the computer to interact with the USB ports. Communication between the computer programming environment (XOD) and a connected Arduino board depends on a chipset that provides an interface between USB port and serial communications with the microcontroller. Two sources of chipsets are widely used, the CH340 and FTDI families. Since you are connecting the Board with your computer to upload code on it, you will need the proper USB drivers to allow that transfer. The Rich UNO R3 board supplied in the Biomaker starter kit uses the CH340G USB interface driver

chip. Recent versions of Windows, MacOS and Linux operating systems probably contain existing compatible drivers. The website below provides information on which operating systems have the CH340G driver pre-installed, and how to install or update drivers if not: <https://learn.sparkfun.com/tutorials/how-to-install-ch340-drivers>

If you are a linux user, your library should have packages that will allow you to do the same. Type "usb driver" in the research bar of your packets manager, and select the ones that have CH34x for I2C and SPI/GPIO drivers.

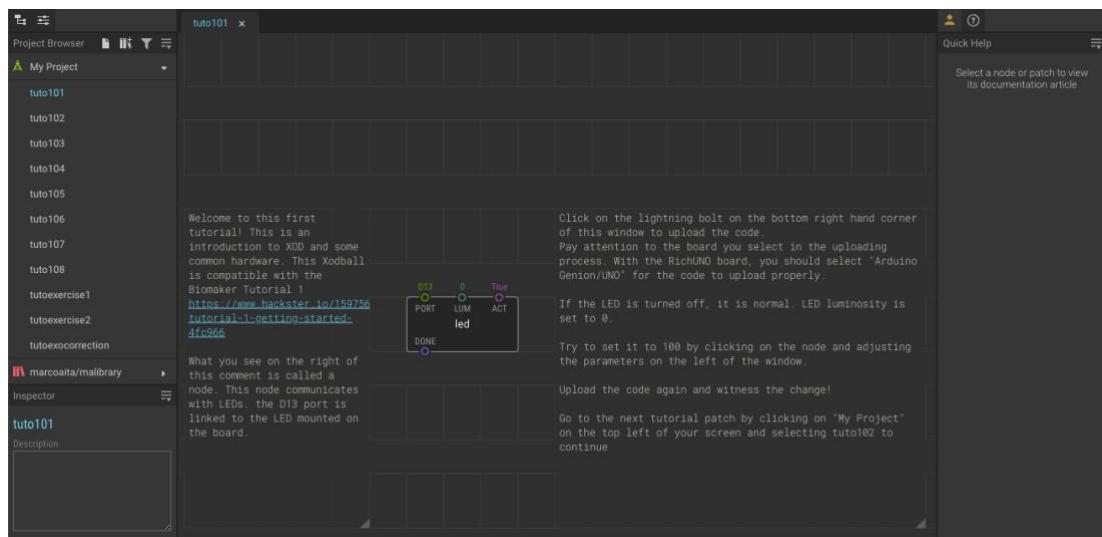
Step 2: Install XOD

The XOD development environment is an open source software that can be used to program microcontroller boards. Other open source software, such as the Arduino IDE, can also be used, however, we have chosen to use XOD as it does not require text-based coding and has a graphical interface instead. Just like the Arduino IDE, XOD uses an adapted version of the programming language C++. We recommend that you download the desktop application of the XOD development environment that is suitable for your computer from <https://xod.io/downloads/>. Note that there is also a web-browser based version also available, however you can only use this for development and simulations, not for coding hardware.

The XOD development environment is relatively new, and doesn't yet have built-in support for as wide range of hardware as the Arduino IDE. However, the level of support is increasing rapidly, and in addition, XOD provides a number of ways for building drivers for new hardware.

Step 3: Download the Tutorial Software

Download the XOD code for this tutorial [here](#). A file named tuto1_kaiRyn0QME.xodball should be downloaded to your computer. Open this file in XOD. You should see a new item appear, which should contain a list of tutorial patches. You can work through most of this tutorial directly in XOD, by following the instructions provided in each patch.



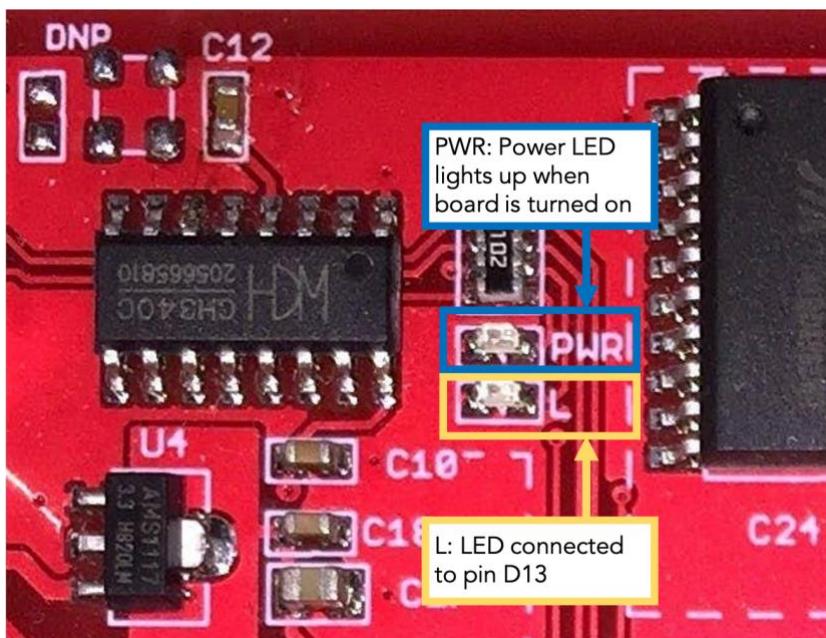
XOD Tutorial Opening Screen

Step 4: Connect the Open Smart Board

Plug in the Rich UNO R3 board to your computer with the provided USB-A to USB-B cable. (You may require an adapter if your computer lacks a USB-A connector).

Step 5: Testing the Board and Connection (*tuto101*)

There is an onboard LED that can be used to provide a visible output. It is positioned adjacent to the power LED (between the USB connector and 4-digit display), and connected to output D13. The *led* node can be used in a XOD patch to control lighting of this LED. We can use this as a simple test to check setup of the board and software. You can follow the instructions in patch *tuto101* to test the onboard LED. When it comes to uploading the code you will need to select the correct type of board for the code to work. For the Arduino UNO Rich R3 board you should select either *Arduino Genion/UNO* or *Arduino Uno*.



Onboard LEDs on the Arduino Rich UNO R3 Board

You can make the LED flash by using the *square-wave* and *flip-n-times* nodes found in the *xod/core* library. You can add these to your patch, either by dragging them from the core library, by double clicking on the patch and starting to write the node name, or by clicking the patch, pressing "i" and starting to write the node name. Connect the OUT pin of either of these nodes to the LUM pin of the *led* node. The *square-wave* node will allow you to flash the LED continuously, whilst the *flip-n-times* node will allow you to flash the LED a set number of times. These nodes will also allow you to program the frequency, duty cycle and number of flashes. Have a go at playing around with these nodes. Try to understand how they work by changing the pin values.

You can find out more about how the *square-wave* and *flip-n-times* nodes work [here](#) and [here](#). A full list of nodes available in the core library can be found at: <https://xod.io/libs/xod/core/>.

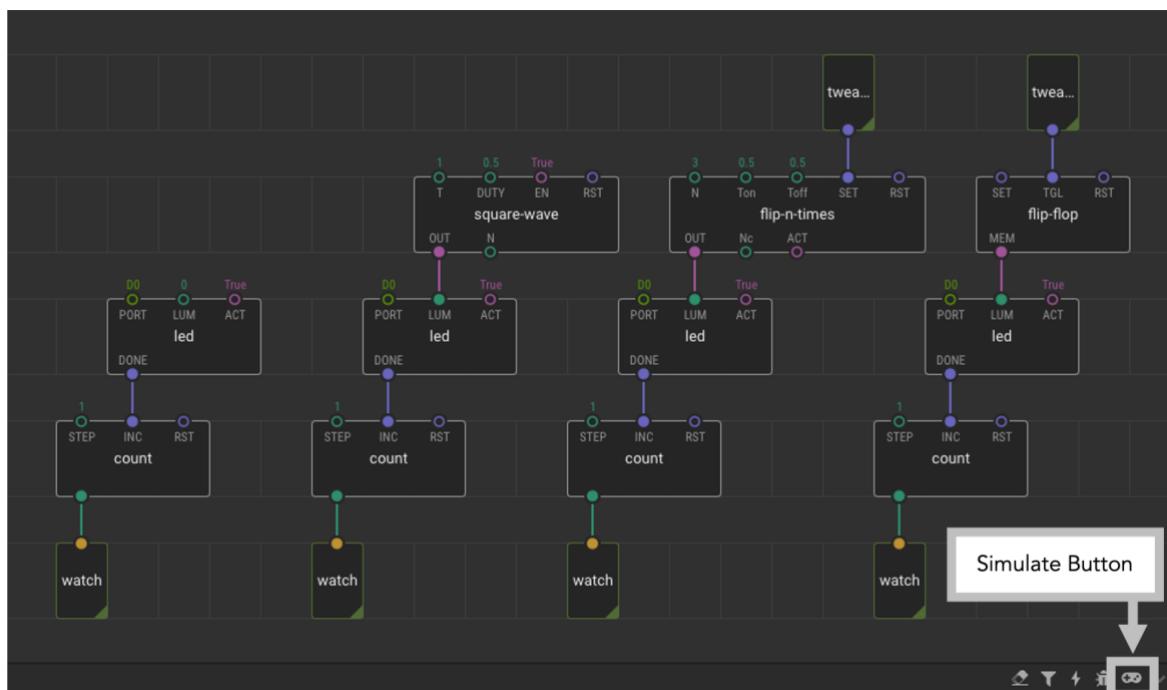
Note: Simulating XOD Patches Without Hardware

If you do not have the required hardware you can simulate this test by connecting a *count* node and a *watch* node to the LED's output pin. The *watch* node allows you to read out the output coming from a pin during simulation or debugging. However, *watch* nodes can not be connected to pulse outputs, so we add in a *count* node which will count up each time a pulse is sent from the LEDs output pin.

Press the simulate button (looks like a game controller) in the bottom right of the screen. If the *watch* node displays the value '1.00' this means that the LED node has sent 1 pulse to turn the LED on, and the patch is working.

You can also add a *square-wave* or *flip-n-times* node to simulate making the LED flash, as described above. The *square-wave* node will cause the *watch* node to count up constantly, with each count representing one cycle of the LED flashing on and off. The *flip-n-times* node will cause the counter to go up by twice the number of flashes specified, once for each switch on and once for each switch off. You can also add a *tweak-pulse* node connected to the SET pin of the *flip-n-times* node. This node will act like a button on the board. When you run the simulation, click on the *tweak-pulse* node and you will see a button labeled 'Pulse' appear in the inspector pane. Clicking this button will start the series of flashes, and the *watch* node will continue to count up each time you press it. You can reset the count to 0 by adding another *tweak-pulse* node to the RST pin of the *count* node. This will simulate a reset button.

You can also simulate a button using the *flip-flop* node. Connect the MEM pin to the LED's LUM pin, and a *tweak-pulse* node to the TGL pin. The *flip-flop* node simulates switching the LED between on and off each time you press it. This will be represented in the *watch* node by a count each time you press it.



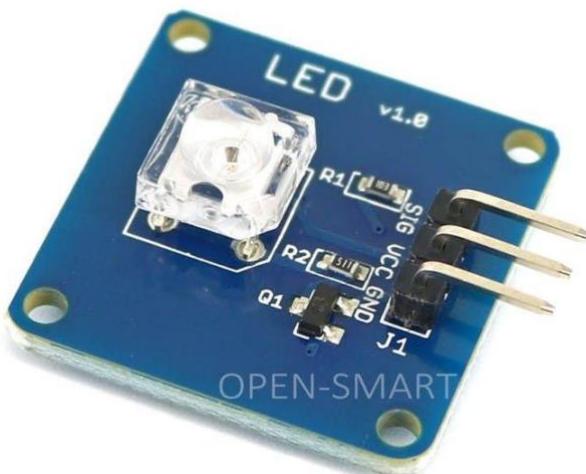
XOD Patches for Simulating an LED

You can simulate most of the early tutorials in this handbook by adding *tweak* nodes to inputs and *watch* nodes to outputs, however some patches involving specific hardware can not be simulated in XOD. Note that there are different types of *tweak* nodes for different types of input (e.g. *tweak-number* or *tweak-boolean*). Sometimes, as is the case for our LED, you will not be able to add *tweak* and *watch* nodes directly to the inputs and outputs of your patch, as the data types do not match (e.g. you can not connect a number output to a boolean input). In this case using a combination of “*tweak-pulse* and *flip-flop*” for an input and “*count* and *watch*” nodes for an output will often suffice. Where relevant, instructions for adapting each tutorial for simulations will be given in italics and marked by an asterisk at each end (e.g. **simulation instructions**).

More information about *watch* nodes, *tweak* nodes and simulations can be found on the XOD website: <https://xod.io/docs/guide/debugging/>. Note that simulations are compiled on the XOD cloud server and you have a limited number of compilations allowed per day. Signing in to a free XOD account will increase the limit to 100 compilations per day, but if you exceed this you may have to wait until the next day to continue.

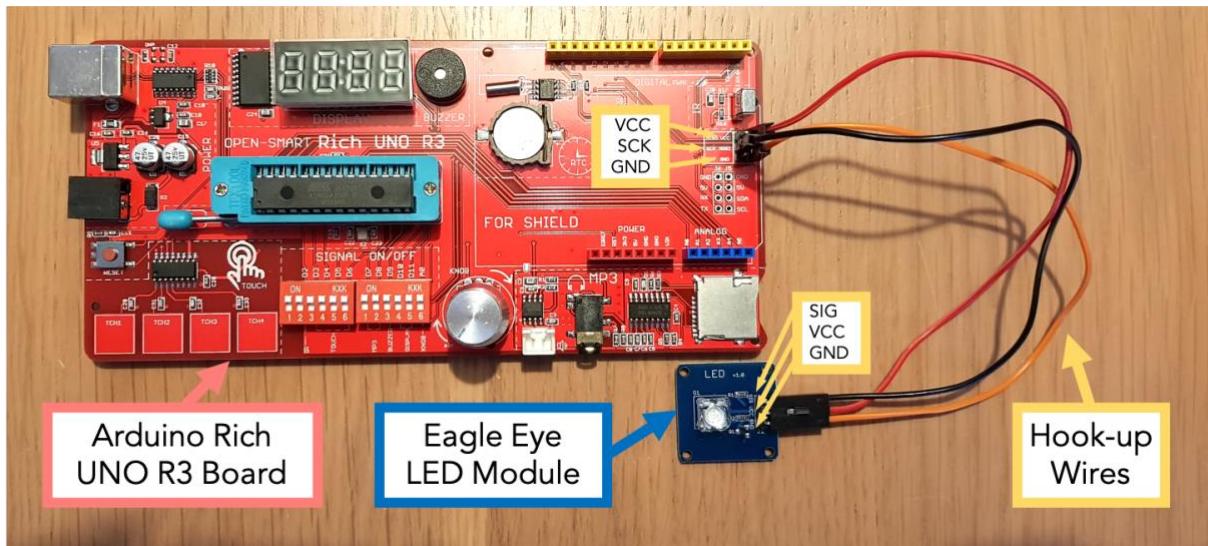
Step 6: Connecting an External LED to the Rich UNO R3 Board (*tuto102*)

A brighter LED output can be arranged by plugging in the eagle Eye LED module that comes with the Biomaker starter kit.



Eagle Eye LED Module

The LED module can be connected to GND, VCC (5V) and SCK (D13) port found as pins on the Rich UNO board, using female-female hook-up leads provided in the Biomaker starter kit. Be sure to connect GND-GND, VCC-VCC and SCK-SIG. The brighter LED will echo the behaviour of the smaller LED found on the Rich UNO R3 board, as both are connected to the same port. The instructions in *tuto102* will guide you through using an external LED, and the first stages of adding a button (Step 7, below).



Wiring the Eagle Eye LED Module to the Arduino Rich UNO R3 Board

Step 7: Reading Input from a Touch Button and Controlling the LEDs (tuto102-104)

The onboard touch buttons (TCH1-4) are connected to ports D3-D6 of the Rich UNO R3 board. In XOD, these switches can be represented using the *button* node from the *xod/common-hardware* library. The button node also provides debouncing for a switch, meaning that a single press is read as a single press, and not multiple short presses. Patches *tuto102*, *tuto103* and *tuto104* will guide you through using a touch button to control the LED.

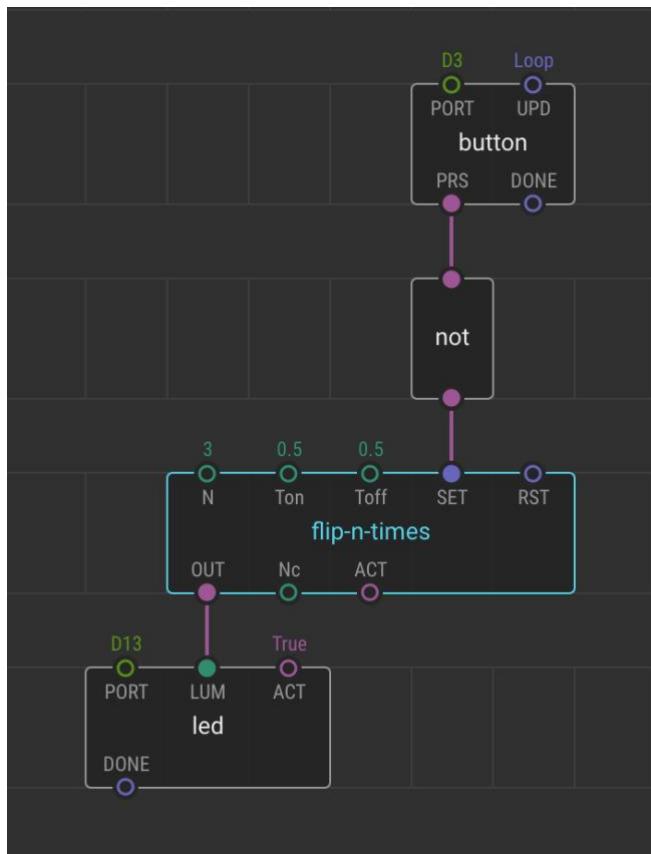
A touch button can be simulated without hardware using the method described in the note after Step 5.



Touch Buttons TCH1-4 on the Arduino Rich UNO R3 Board

Note that the touch buttons are set to the logic level HIGH and come down to the logic level LOW when pressed. This means that the device they control will be on, and will turn off when the touch button is pressed. This can be reversed by connecting the output to a *not* node from the *xod/core* library, which will switch around the state so that the logic level is set to LOW, and will switch to HIGH when pressed.

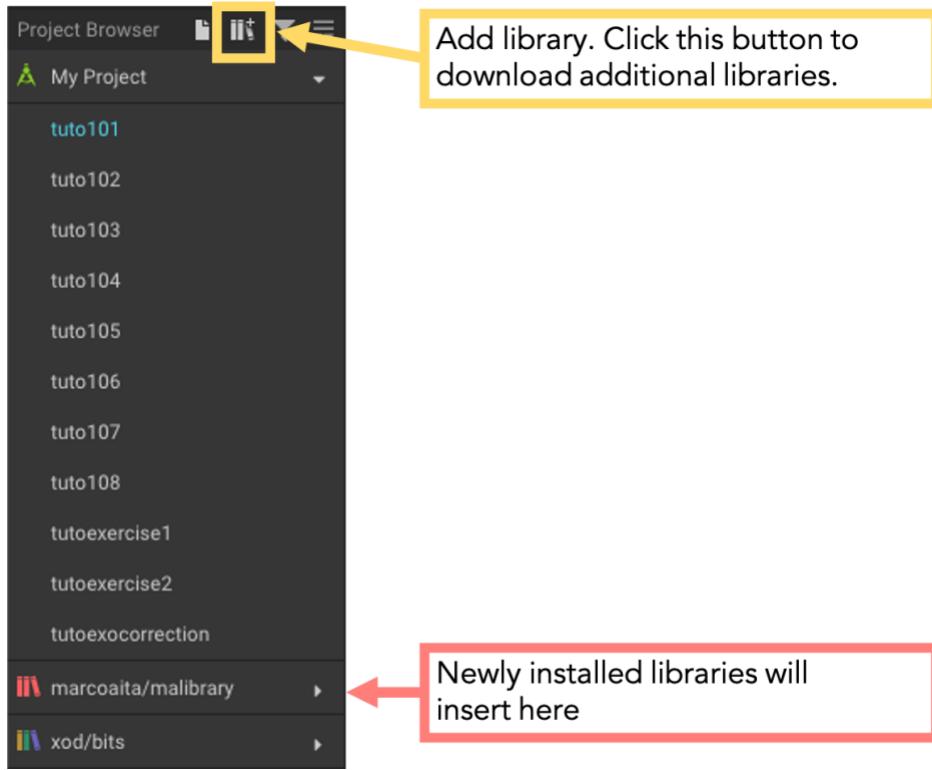
Switch signals can also be fed into a *flip-n-times* node, available in the default xod/core library. The *flip-n-times* node can be set to turn on and off a set number of times for different lengths of time.



XOD Patch to Trigger an LED to Flash a Set Number of Times

Step 8: Importing an External XOD Software Library (*tuto105*)

There is no node in the *common-hardware* library to represent a buzzer. However, Dr Marco Aita, a previous Biomaker participant, has done the work to transfer the Arduino tone library in a XOD library. This library contains a node that can be used to drive and control the pitch of the buzzer. To import it, click on the 'Add library' button at the top of the Project Browser pane, type in *marcoaita/malibrary*, and press enter to install. The patch *tuto105* will walk you through this task.



Adding a library in XOD

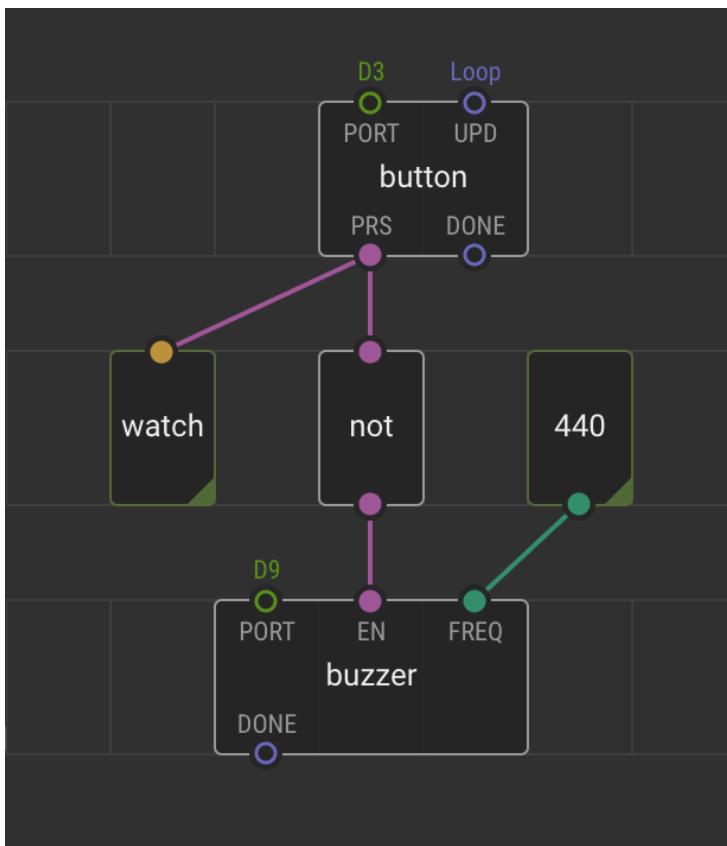
Step 9: Using the Piezoelectric Buzzer (*tuto105-108*)

A piezoelectric buzzer is provided on the Rich UNO R3 board, positioned to the right of the 4-digit display. It is connected to output D9, and can be used to create sound, for example an audible warning. The board comes with a sticker covering the buzzer, you will need to remove this to use it. Follow the instructions in patches *tuto105*, *tuto106*, *tuto107* and *tuto108* to learn about using the buzzer, as well as how to tweak input numbers and how to debug your code.

Please note that the buzzer can not be simulated without hardware.



Piezoelectric Buzzer on the Arduino Rich UNO R3 Board



XOD Patch for Controlling the Onboard Piezoelectric Buzzer with a Touch Button

Similar to the LED, the *flip-n-times* node can be inserted between the *not* node and the *buzzer* node to generate an on-off pulse for a repeating sound. This could be used as an alert or alarm.

Demo code for control of the passive buzzer c/o Marco Aita.

Step 10: Extended Exercises (*tutoexercise1*, *tutoexercise2* and *tutoexercisecorrection*)

You can now control two of the kit's hardware parts - congratulations! Endless possibilities are now open to you to build simple patches and interact with the board. To get a good handle on how XOD interacts with the board, you can try the following simple exercises:

- Change the touch button you use to interact with the LED/buzzer. Touch buttons 1-4 are connected to pins D3-D6 of the board.
- Change the pitch of the buzzer by tweaking the frequency.
- Make the buzzer and the LED connect to the same button, and make the buzzer turn on and the LED turn off when the button is pressed.
- Create a four musical notes piano by linking the buzzer at different frequencies to each of the four buttons (an example is given in *tutoexocorrection*, but try to attempt this yourself first!).
- Add a repeating blinking LED as a metronome (help can be found in the XOD tutorial on how to make loops for help <https://xod.io/docs/tutorial/205-loops/>) (an example is given in *tutoexocorrection*, but try to attempt this yourself first!).

Chapter 6: Handling Simple Input Output Devices

This tutorial will expand on the knowledge you have gained in chapter 5, and will focus on connecting and controlling external input and output devices.

Introduction

The Biomaker starter kit is an excellent way to get started with programming simple input and output devices. The XOD software provides a range of nodes for control of devices, and the microcontroller on the Arduino board has a series of digital and analogue ports that can be used to read or write to these devices. For example, in [Chapter 5](#) of this handbook, we saw how to interface with touch-sensitive key inputs and LED and buzzer outputs using standard XOD nodes. An increasing number of low cost devices are becoming available that are (i) capable of a wide range of calibrated measurements and sophisticated outputs, and (ii) use logic controllers with complex serial communication protocols. Specialised software nodes are required to use these devices in XOD, but many of these can be found in published XOD libraries, either default or user-contributed (<https://xod.io/libs/>).

In this tutorial, we will show you how to connect and control an LCD display screen to the Rich UNO R3 board via an expansion shield, how to change the brightness of an LED using a potentiometer, how to read the signal from an analog sensor and how to make your own nodes. By the end of this tutorial you gain an understanding of how to use an expansion shield and a variety of XOD nodes to expand the functions of your Arduino board, and read and write to external devices.

Please note, the tutorials in this chapter cannot be simulated in XOD without hardware. However, you can still follow along to build to required patches, and we recommend using the “Creating your own Nodes” patches (300-306) in the XOD welcome tutorial to practice building your own nodes.

Objectives

- Plug in an expansion shield and use it to connect external input and output devices.
- Send data to an LCD display.
- Use the onboard knob potentiometer to alter text on the LCD display.
- Use the onboard knob potentiometer to alter the brightness of an LED.
- Build and readout an analog sensor using the kit’s light sensor module.
- Use an analog sensor to write a simple program.
- Learn how to make new nodes in XOD.
- Plug in and program a ring of addressable LEDs.
- Experiment with writing customised programs.

Requirements

- Computer running MacOS, Windows or Linux

- XOD code for tutorial (download [here](#))
- Arduino Rich UNO R3 board
- Expansion shield
- I2C 1602 LCD display module
- Eagle eye LED module
- RGB LED ring
- Light sensor
- Hook-up wires/Dupont line

Step-by-Step Guide

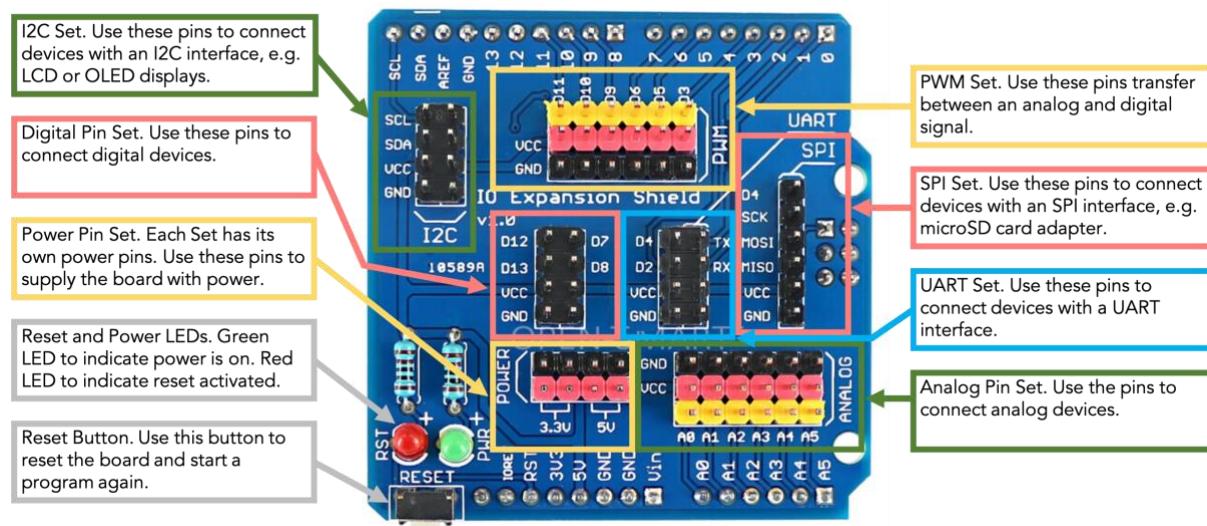
Step 1: Downloading the Tutorial Software

Download the XOD code for this tutorial [here](#), and open in the XOD IDE.

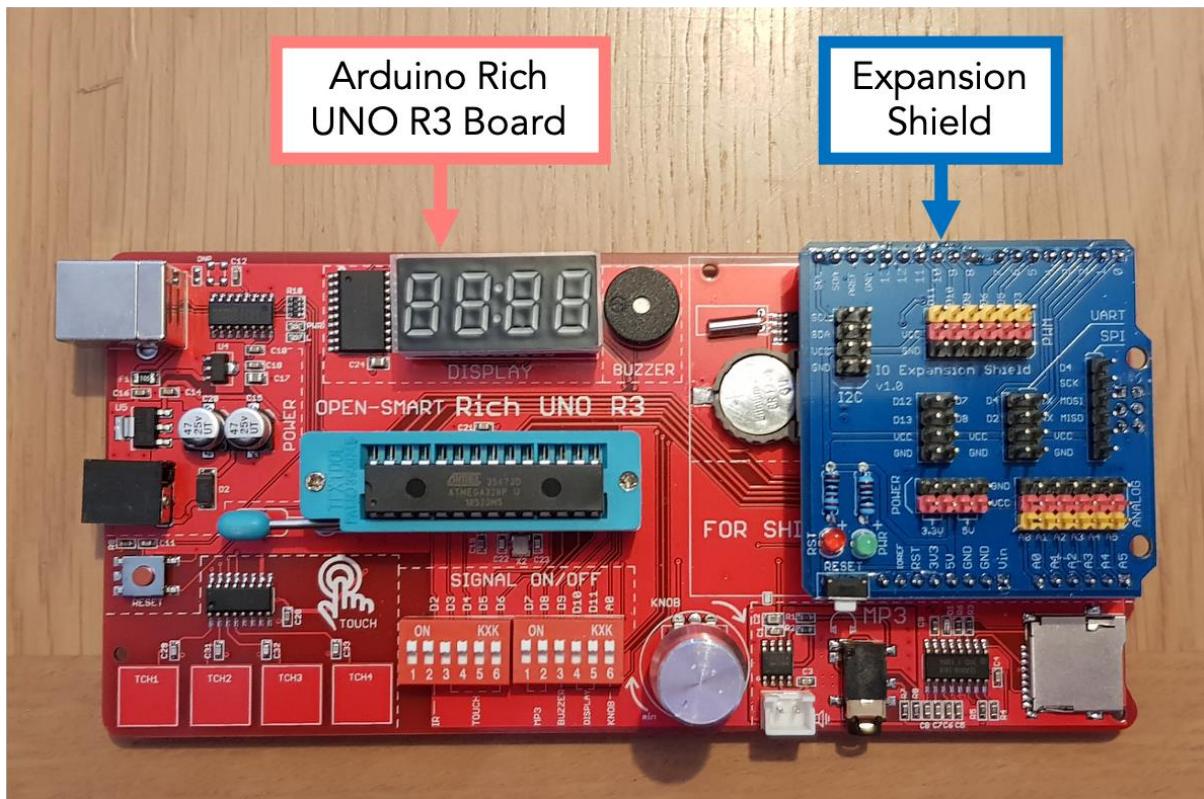
Step 2: Connecting the Expansion Shield

The header pins on the Arduino board can be used to plug in a variety of standard Arduino UNO shields. This allows simple expansion of the board's functions. Note also that the microcontroller ports are connected to on-board devices via a bank of DIP switches. This allows existing functions to be switched off, to avoid conflict with new connected devices.

The starter kit includes an IO Expansion Shield. Plug this shield into the red, yellow and blue header sockets on the top right hand side of the board to simplify the connection of new hardware devices. Use the image below to remind yourself about the different sets of pins on the expansion shield.



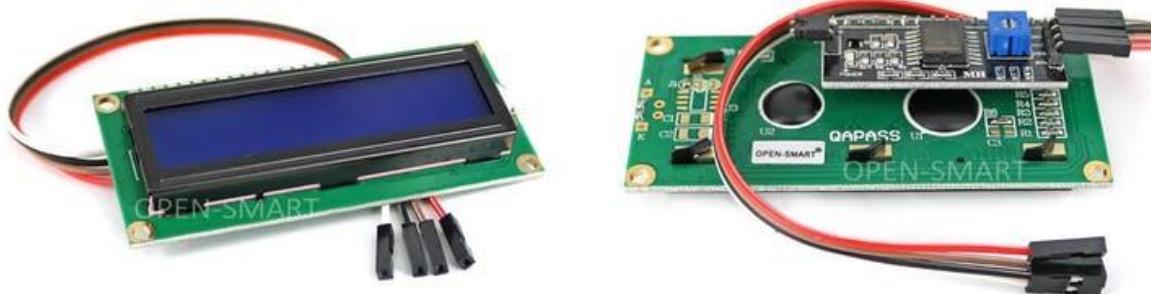
Open Smart Expansion Shield



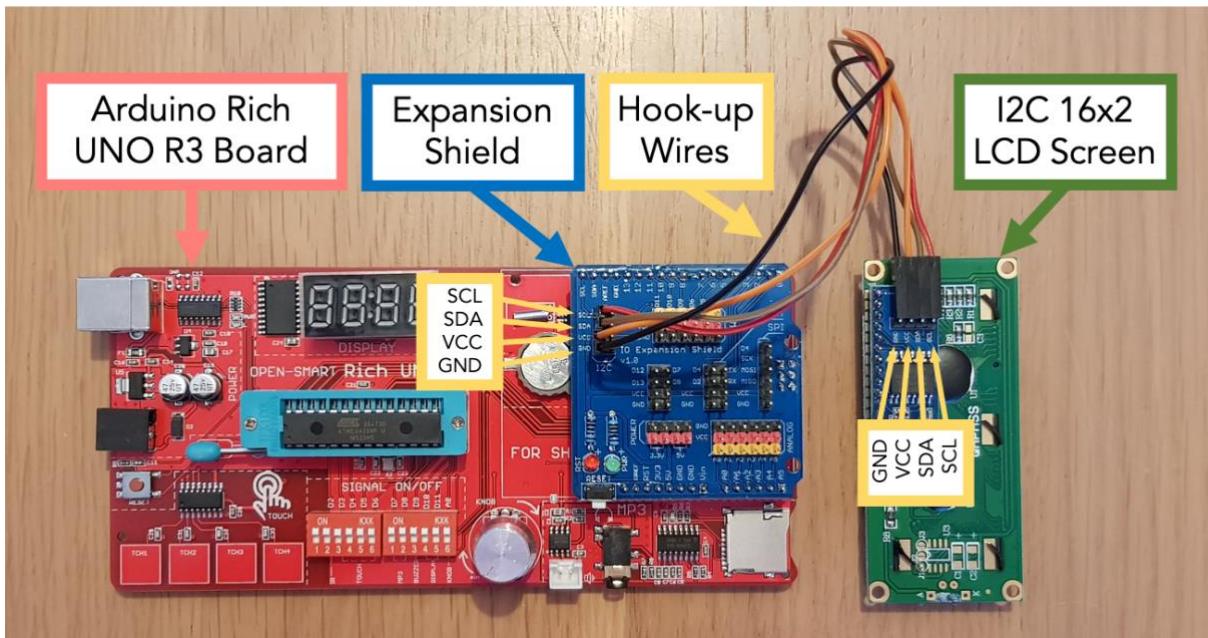
Open Smart Expansion Shield Connected to the Arduino Rich UNO R3 Board

Step 3: Connecting the 16x2 LCD Display

The starter kit contains a liquid crystal display (LCD) capable of displaying 2 lines of 16 characters. The device is equipped with an I2C interface that allows serial communication with the device. Using the hook-up wires provided in the kit, connect the pins on the back of the LCD screen to the I2C pins on the expansion shield. Make sure to connect GND-GND, VCC-VCC, SDA-SDA and SCL-SCL.



Open Smart I2C 1602 LCD Module with Hook-up Wires



Wiring for the I2C 1602 LCD Module

Step 4: Loading the XOD Node from the Library (*tuto201*)

The first patch in the XOD tutorial, *tuto201*, will introduce you to the XOD node that represents the LCD display. The patch should contain a node called *text-lcd-16x2-i2c*, which can also be found in the *xod/common-hardware* library. The LCD screen provided in the starter kit communicates with the board via an I2C interface, which means you will have to identify the device's address. This screen has an address of 0x38 (also represented as 38h). To change this, click on the node and change the ADDR pin to 38. You can use the Quick Help pane to explore the functions of the node's other pins. Follow the instructions in the tutorial to display text on the LCD screen.



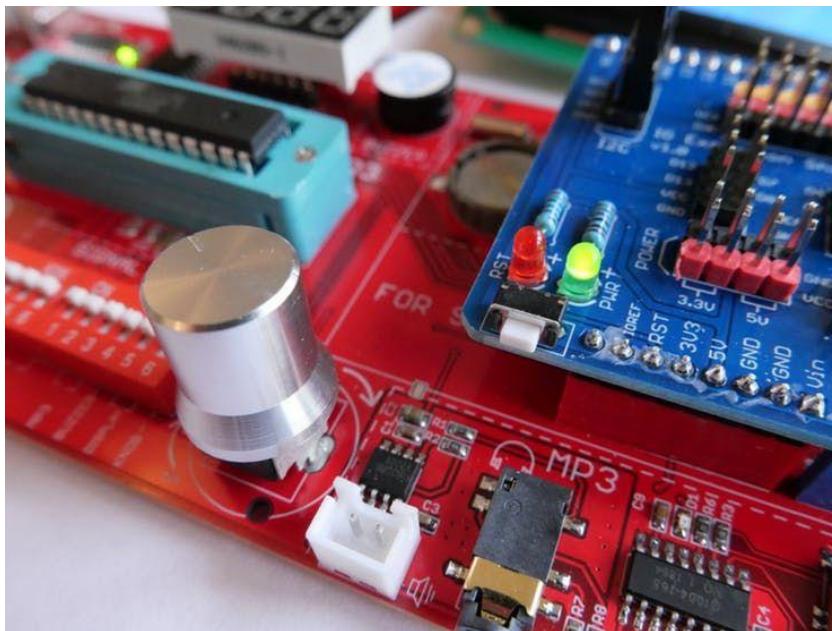
XOD Node for the I2C 1602 LCD Display Module



I2C 1602 LCD Display Module Programmed to Show Text

Step 5: Communication with the I2C Display in XOD (*tuto202*)

A knob sensor, also known as a potentiometer or variable resistor, is provided on the Rich UNO R3 board, connected to port A3. The XOD node to represent this potentiometer is called *pot*, and can be found in the *xod/common-hardware* library. Follow the instructions in *tuto202* to display the value of the potentiometer on the LCD screen.



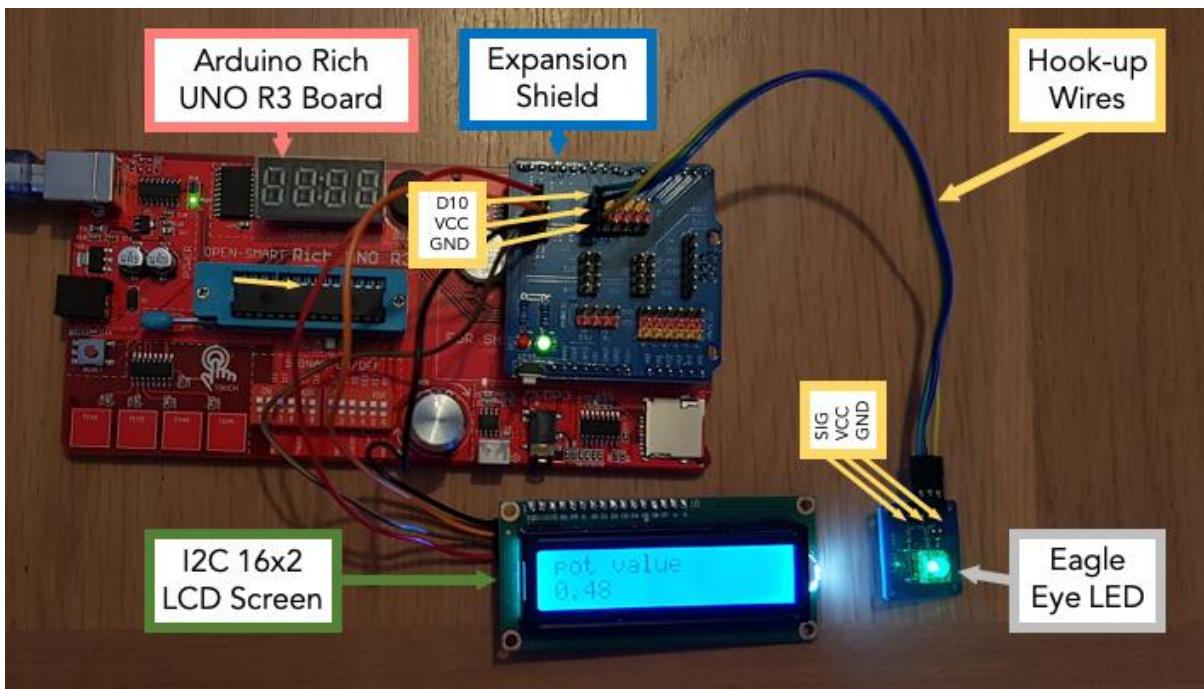
The Potentiometer Knob on the Rich UNO R3 Board

To format how the values are displayed, you can insert a *format-number* node between the *pot* node and the *text-lcd-16x2-i2c* node. This will allow you to set the number of significant digits, and convert the numerical value to a string for display on the LCD.

Step 6: Using the Potentiometer to Adjust the Brightness of an LED (*tuto203*)

Ideally, you should unplug the Rich UNO R3 board from its power source (USB cable and/or power pack) before rewiring circuits. Unplug your board, and then connect the eagle eye LED module you used in the last chapter to the shield using the PWM set of pins. Make sure to connect GND-GND VCC-VCC and SIG to one of the digital PWM pins (D3, D5, D6, D9, D10 or D11). Now reconnect your board. The patch *tuto203* will walk you through how to use the potentiometer to control the brightness of the LED. You will need to make sure that the LED's PORT pin is set to whichever digital PWM pin you used to connect the LED.

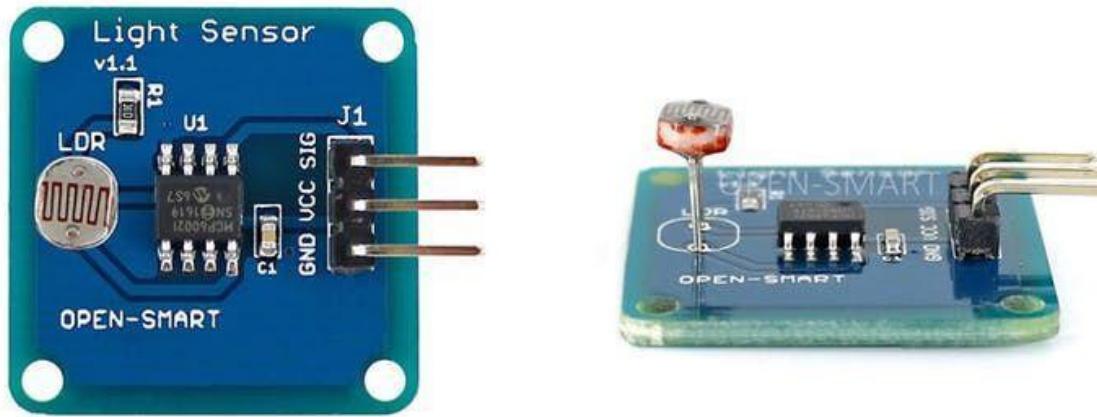
Note: PWM stands for pulse width modulation, and it provides a way for the Arduino microcontroller to produce variable average voltage outputs from digital (on-off) signals. The digital ports labelled PWM can deliver binary outputs (0 or 5V) as a series of rapid pulses. The relative length (duty cycle) of the 5V pulses will produce different average voltages, which can be used to control the output of devices like an LED - e.g. to regulate the brightness of the LED.



Arduino Rich UNO R3 Board with LCD Screen and LED Modules Attached

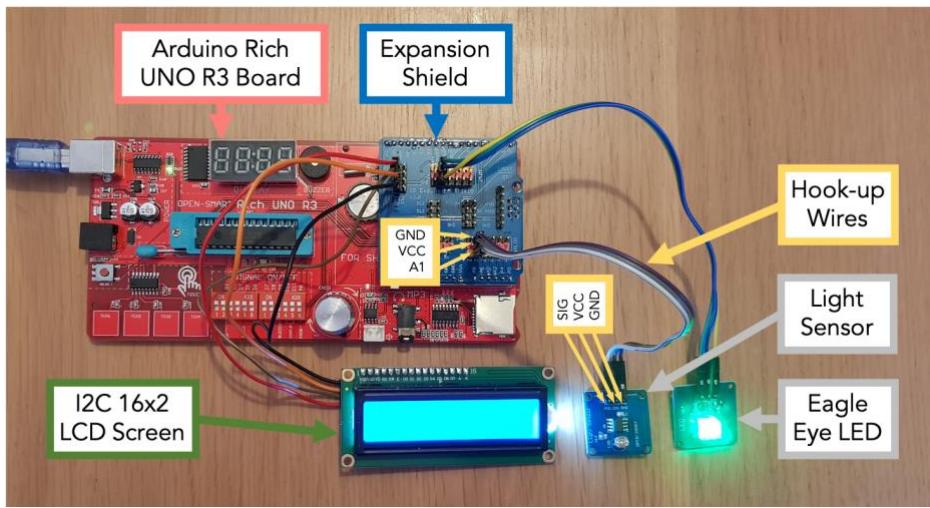
Step 7: Building a New Analog Sensor Node (*tuto204-207*)

Take the light sensor from the starter kit and use the hook-up wires to plug it into GND, VCC and A1 in the analog pins set. This sensor essentially works by allowing more current to flow through as it is exposed to more light. Exposed to 10 Lux, it will provide a resistance of 8-20K Ohms, and at 0 Lux, the resistance will be 1M Ohm.



Open Smart Light Sensor

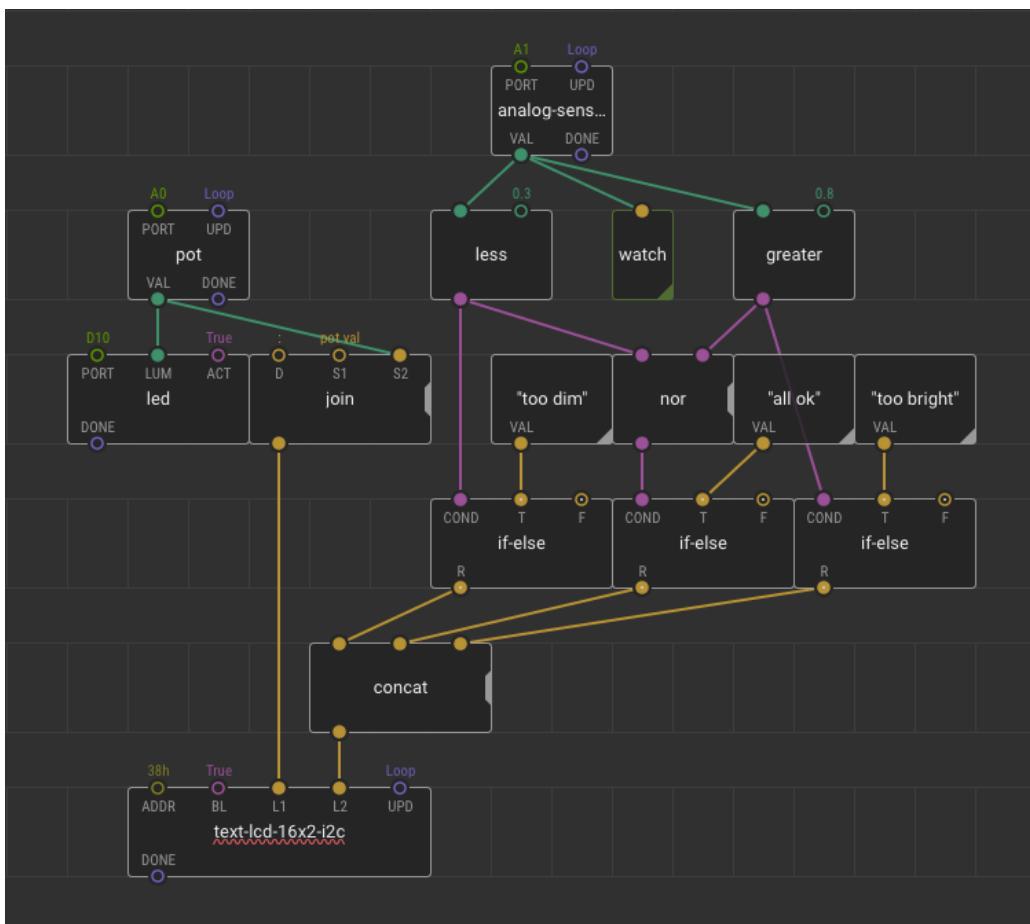
There are no dedicated nodes for the photoresistor in XOD. However, it is relatively simple to create a custom node for this, using the *analog-sensor* node from the *xod/common-hardware* library. Try plugging in an analog-sensor and a watch node to see what happens when you cover the light or expose it to your phone's flashlight. The patches *tuto204*, *tuto205*, *tuto206*, and *tuto207* will show you how to use this node in conjunction with the LCD screen and LED you have already been using.



Arduino Rich UNO R3 Board with LCD Screen, LED and Light Sensor Modules Attached

Step 8: Making a Program with the Sensor (*tuto208-211*)

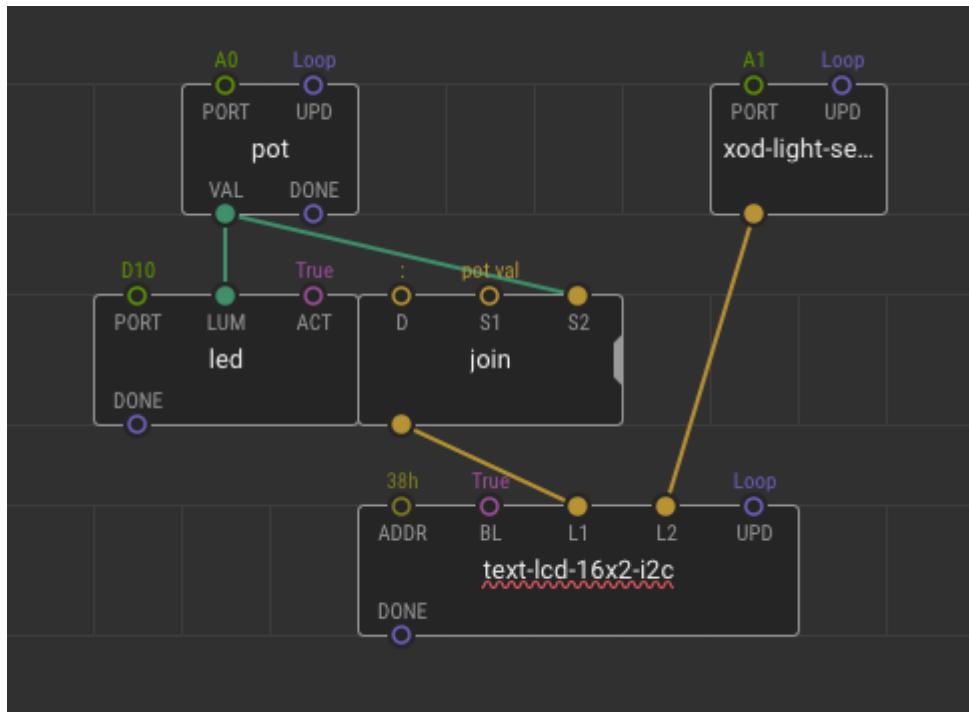
Now it's time to convert this raw analog readout into something actionable - to change the actions of other devices based on the signal from this sensor. The patches *tuto208*, *tuto209*, *tuto210* and *tuto211* will show you how to make a program that reads out when the brightness is too high, too low, or a good level.



XOD Patch to Display when an LED is the Correct Brightness

Step 9: Building your Own Node for your Device (*tuto212-213*)

Now that you have effectively made a nice program, it is time to turn it into a node that you will be able to use for future programs. We will focus on the light sensor and the cutoff value nodes that make up the right half of the program you wrote above. The patches *tuto212* and *tuto213* will show you how to do this.

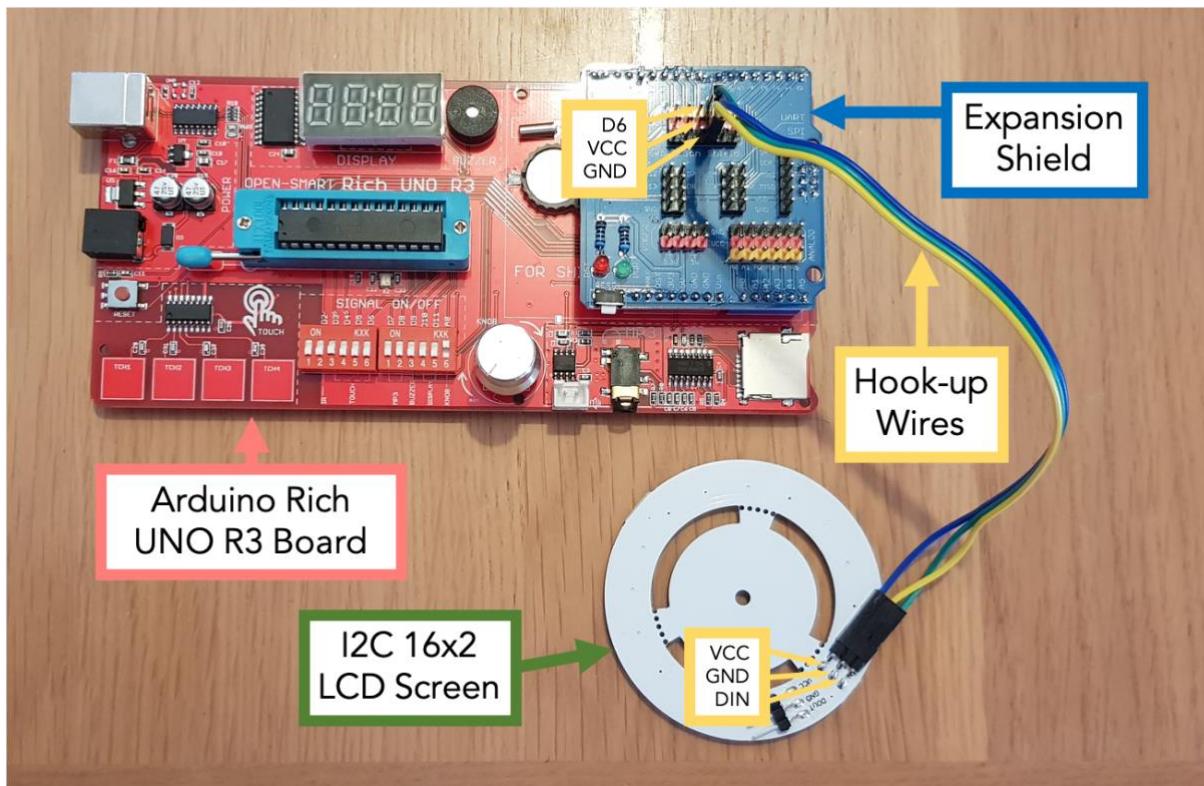


Revised XOD Patch Using the Newly-Made xod-light-sensor Node

Congratulations, you made a brand new node! You can now connect it to other nodes and build more complicated programs using this node as a building block!

Step 10: Using a Ring of Addressable RGB LEDs (*tuto214-216*)

Plug the NeoPixel RGB LED ring into the expansion shield using either the digital or PWM pin sets. The LED ring has two sets of pins: “VCC, GND, DIN” and “VCC, GND, DOUT”. You should use the first set of pins with DIN to plug in the LED ring. Make sure to connect VCC-VCC, GND-GND and DIN to one of the digital pins. You can disconnect the LCD screen, LED and light sensor. You will also need to download two new libraries for this part of the tutorial: [bradzilla84/neopixel](#) and [awgrover/adafruitneopixel](#). The patches *tuto214*, *tuto215* and *tuto216* will show you how to use the LED ring, and how to control it with the onboard potentiometer knob.



Wiring of the RGB LED Ring to the Arduino Rich UNO R3 Board

Step 12: Creating Dynamic Sequences

Take some time to look through the libraries you downloaded and see what nodes are available. Now that you know how to build small programs, try to program the LED ring do different things using these libraries. For example, you could try to:

- Make the LEDs change colour when the potentiometer passes a certain threshold.
- Change the potentiometer for another analog sensor (like the light sensor).
- Delay the rate that the LEDs turn off by adding a *delay* node.
- Make each LED light up a different colour.
- Make each LED light up in turn.

Chapter 7: 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

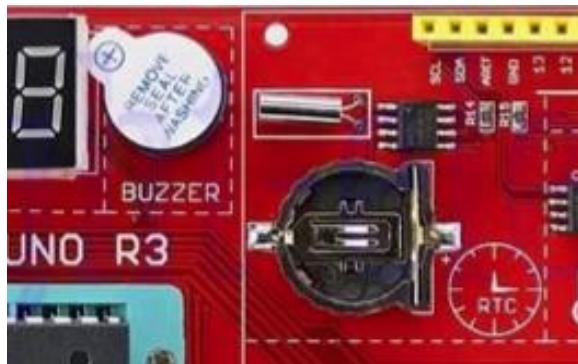
Step-by-Step Instructions

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

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/>. The guide works through an example for how to use the datetime (<https://xod.io/libs/xod/datetime/>) and 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](#).

Download the completed XOD code for this tutorial [here](#).

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.

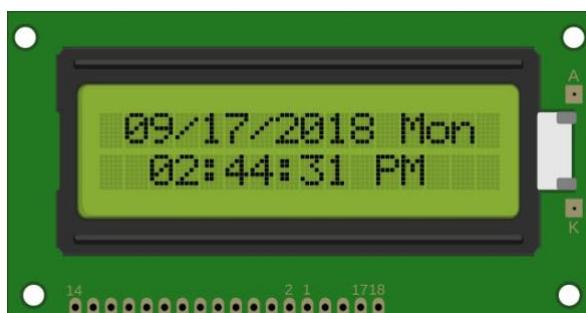
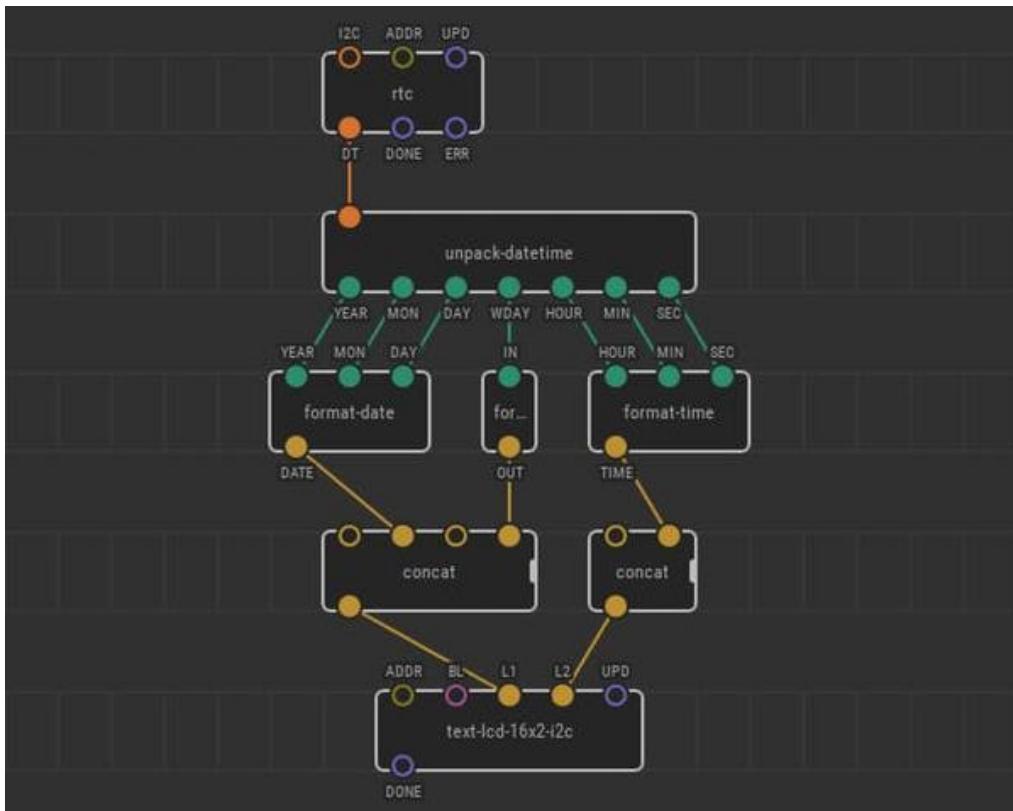
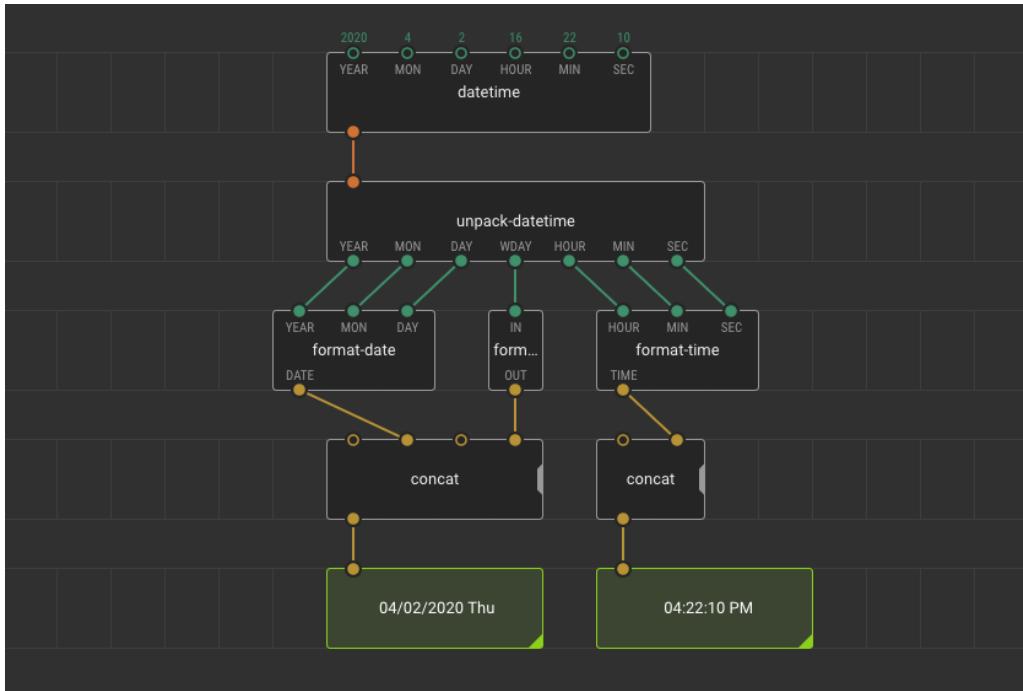


Illustration of a Digital Clock Displayed on an I2C 16x2 Display



XOD Patch Used to Program the Digital Clock

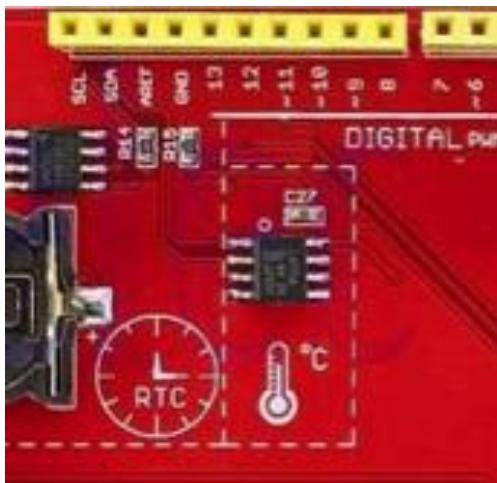
If you do not have hardware available, you can simulate some of the material in this tutorial by replacing the rtc node with a datetime node and the text-lcd-i2c-16x2 node with two watch nodes, as below.



XOD Patch Used to Simulate the Digital Clock

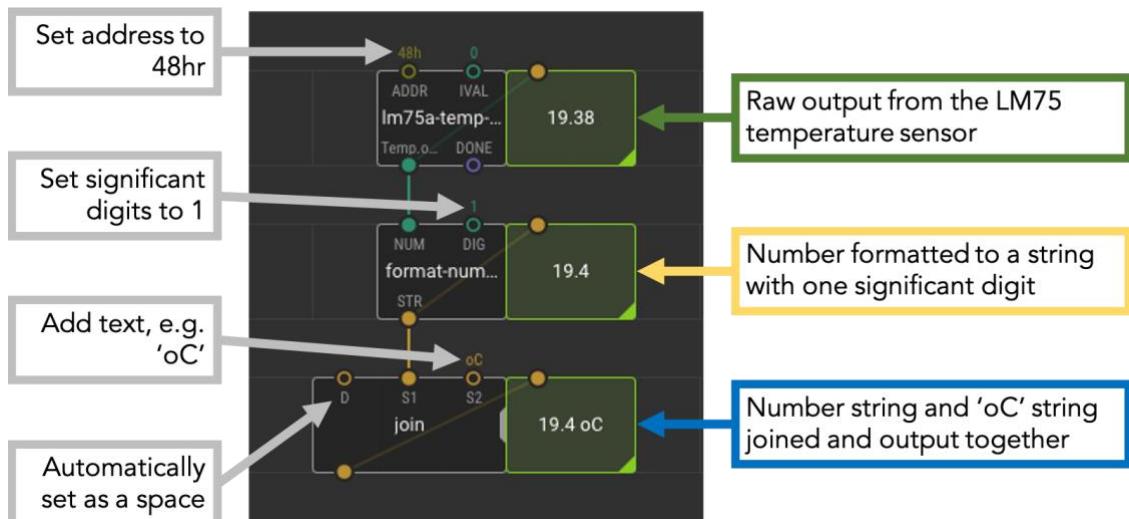
Step 3: Reading from the Temperature Sensor

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

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.

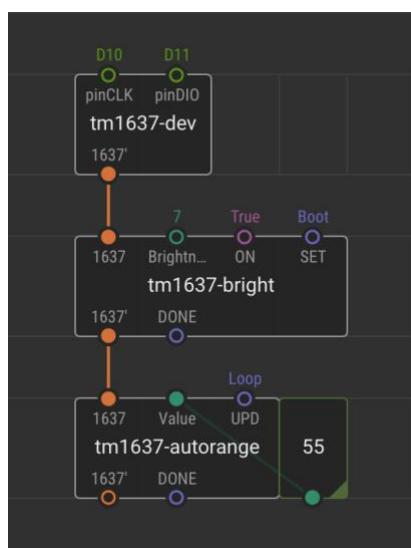


XOD Patch to Format the Output of the LM75 Temperature Sensor

*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.

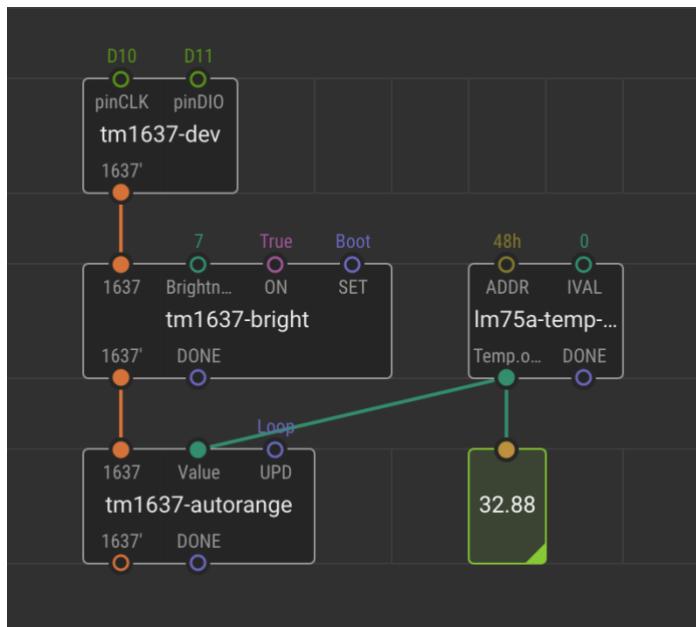


XOD Patch to Write to the 4-Digit Display

Please note that the four-digit display cannot be simulated without hardware.

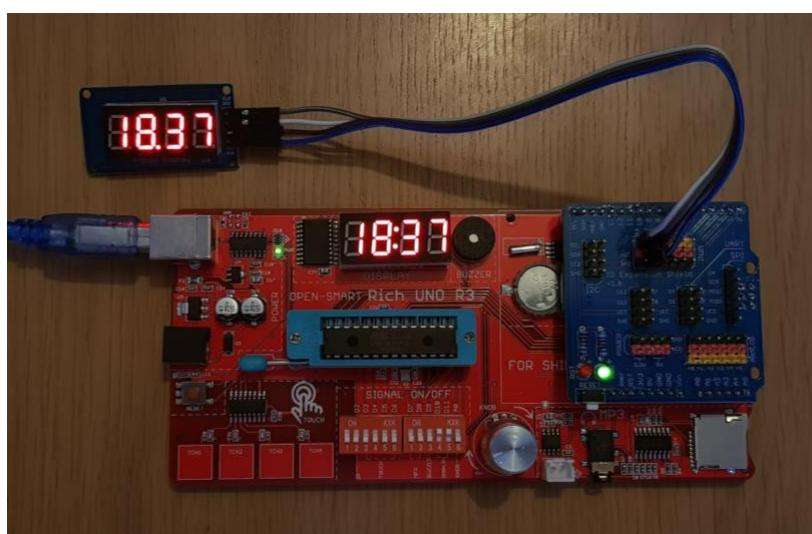
Step 5: Using the 4-Digit Display to Show Temperature Readings

Now let's display some real data on the 4-digit display. Add an *Im75a-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.



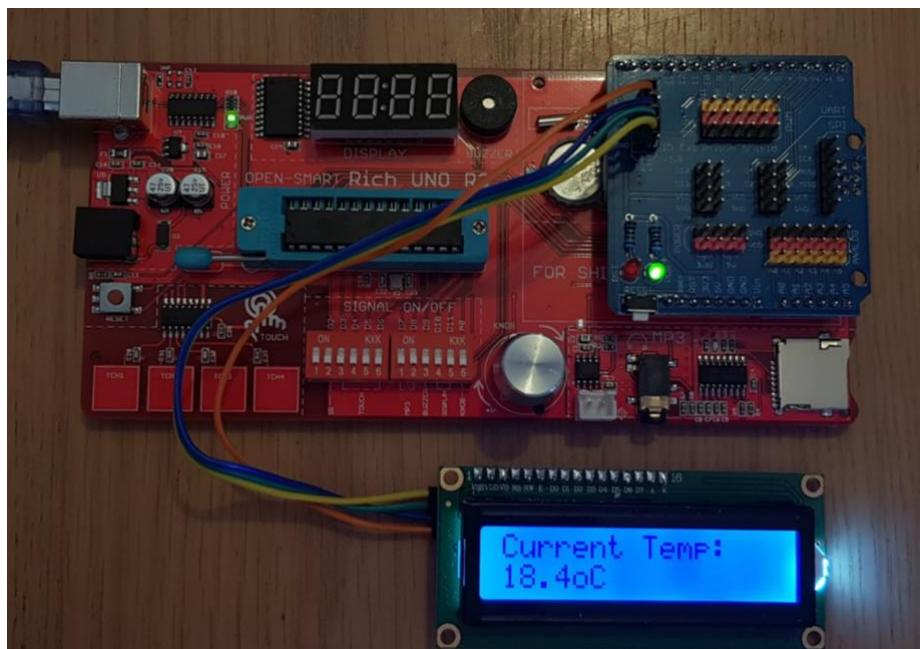
XOD Patch to Show the Temperature on the 4-Digit Display

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

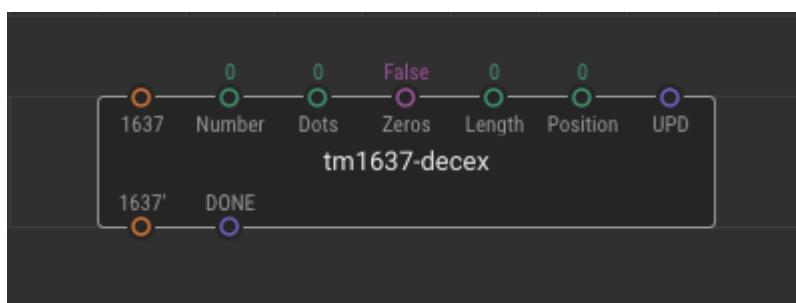
You could also use the kit's 16x2 LCD screen to display this information. Try connecting the LCD screen (as in [Chapter 6](#)) and using it to display the temperature. Using the LCD screen allows you to add additional text and format the display of the number as we did in [Step 3](#).



LCD Display of Formatted Temperature Sensor Data

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-decex*.



The *tm1637-decex* 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:

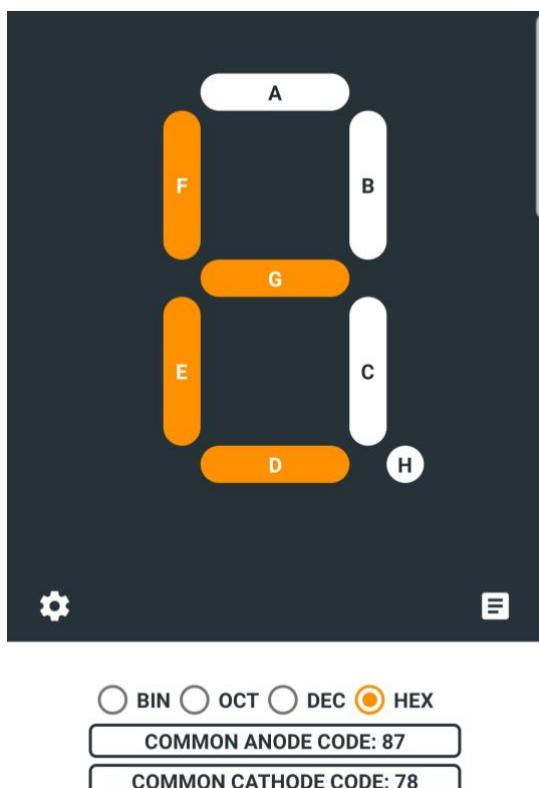
- Number takes your input value.
- Dots determines the type of display, this is explained in the *cesars/tm1637* library under *00-list-readme*, and should be set to 2 for the onboard display.
- Zeros determines whether zeros should be added to empty spaces in the display, e.g. if TRUE it will display 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-decex* 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.

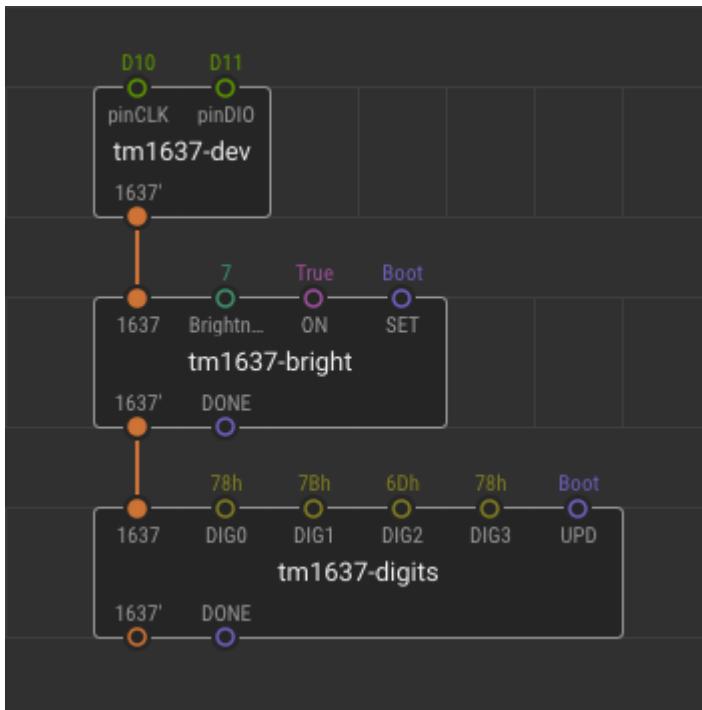
Step 7: Displaying Text on the 4-Digit Display

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.

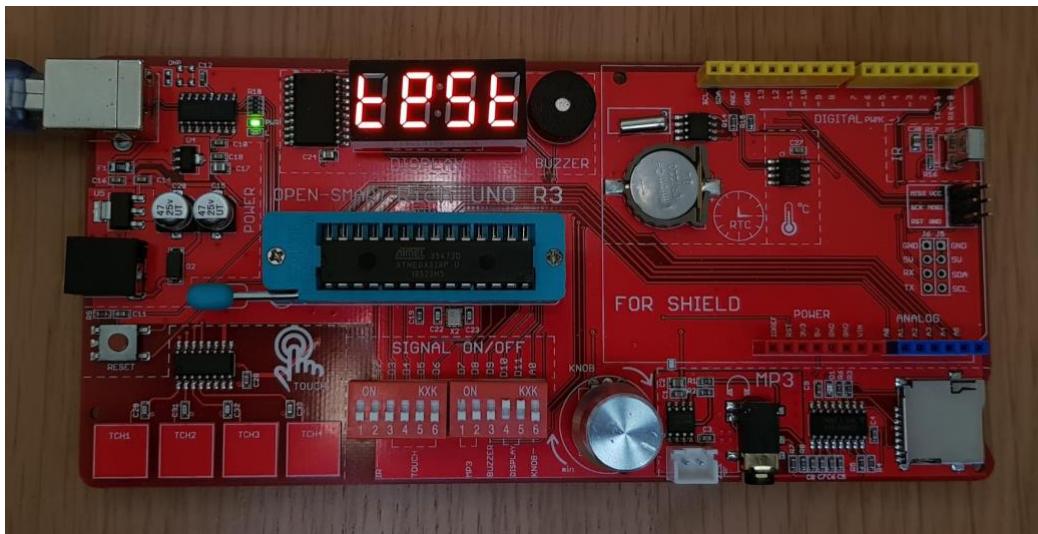


7 Segment Editor App Showing the Letter ‘t’ with the Common Cathode Code 78h

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.



XOD Patch for Displaying the Word ‘test’ on the 4-Digit Display



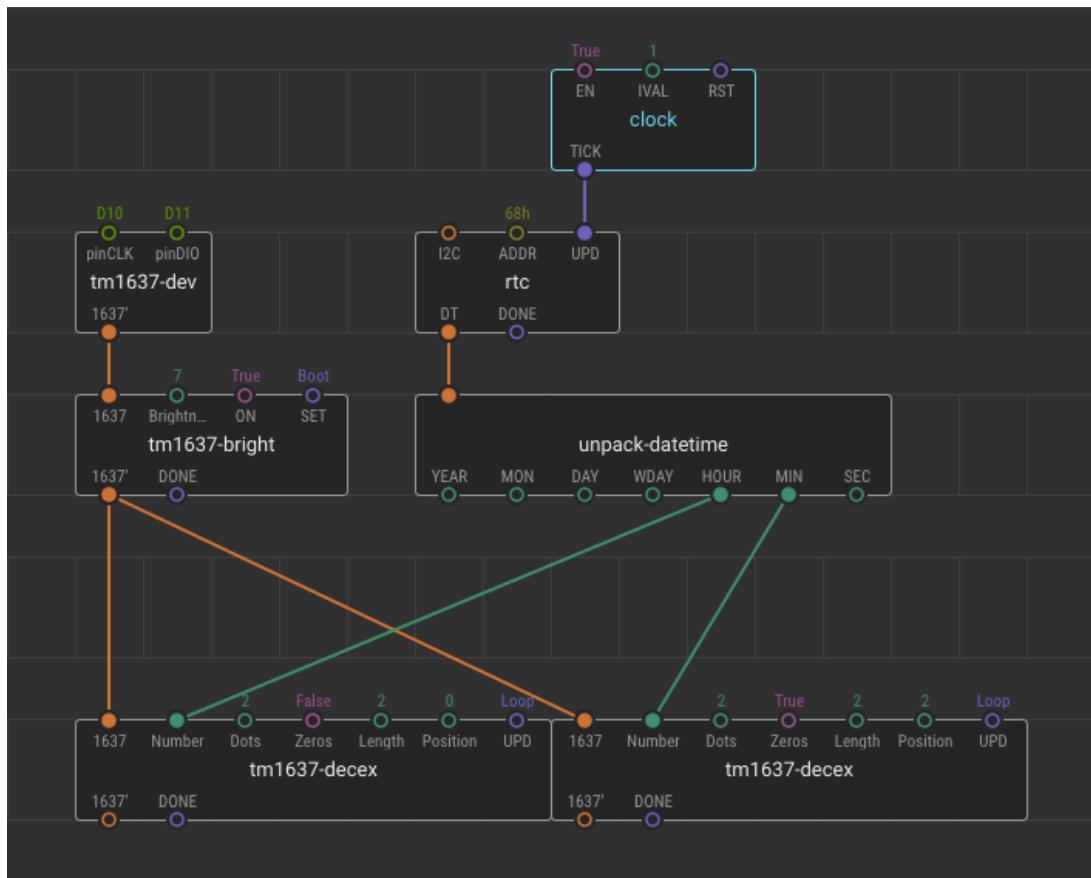
Arduino Rich Uno R3 Board Displaying the Word ‘test’

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.

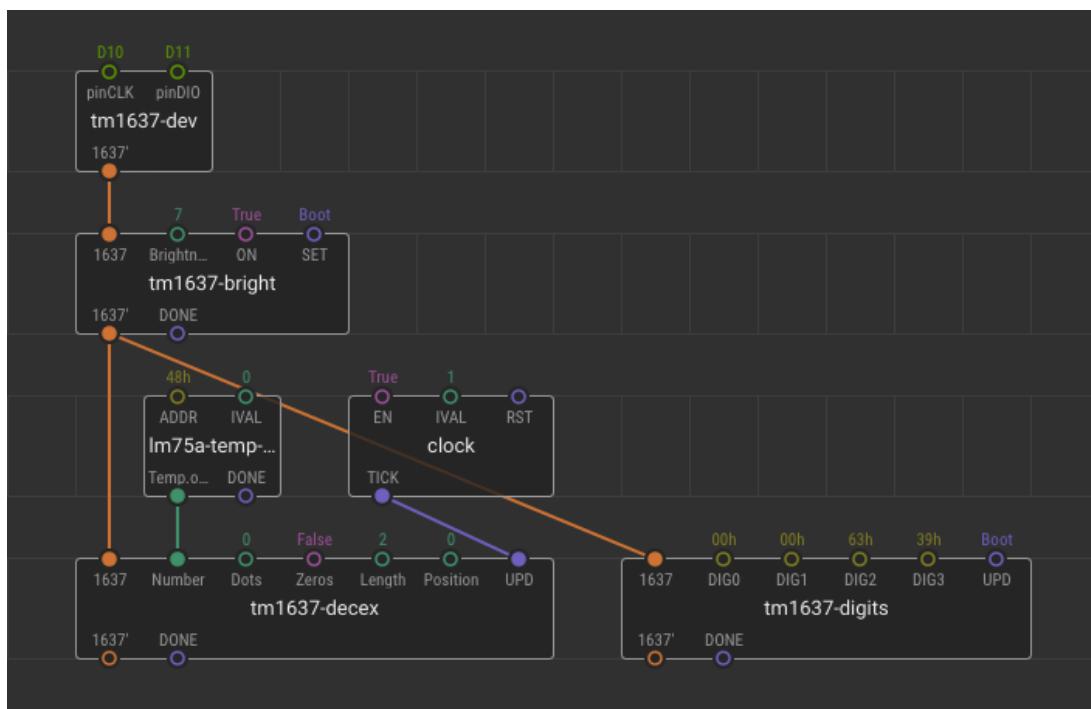


Arduino Rich Uno R3 Board Displaying the Temperature in °C

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

Chapter 8: DIY data logger

This chapter will show you how to make a DIY data logger. It will cover how to read information from sensor devices, and how to display and store data for later analysis.

Introduction

The Biomaker starter kit contains several sensor devices, which allow the construction of a variety of simple instruments for monitoring laboratory reactions, culture conditions and the environment. In this tutorial, we will examine some of the ways that a time-series of sensor data can be logged, displayed and stored on a microSD card. Storing data-point like this can be useful in many situations, and the data can later be imported from the SD card onto a computer for analysis. This tutorial is based on a guide provided on the [XOD website](#), and has been adjusted for use with the Biomaker starter kit. By the end of this tutorial you will be able to build and program your own data logger.

Objectives

- Use an ultrasonic sensor to measure distance
- Understand how to use “buses” in the XOD to tidy up patches
- Display the data from the ultrasonic sensor on the onboard 4-digit display
- Use XOD to display a running average
- Add a timestamp to data to allow data logging
- Write data to a MicroSD card for later analysis
- Understand how simple systems such as this can be used for biological applications

Requirements

- Computer running MacOS, Windows or Linux
- XOD code for tutorial (download [here](#))
- Arduino Rich UNO R3 board
- Expansion shield
- Ultrasonic sensor
- MicroSD card adapter
- MicroSD card
- Hook-up wires/Dupont line

Step-by-Step Guide

Step 1: Downloading the Tutorial Software

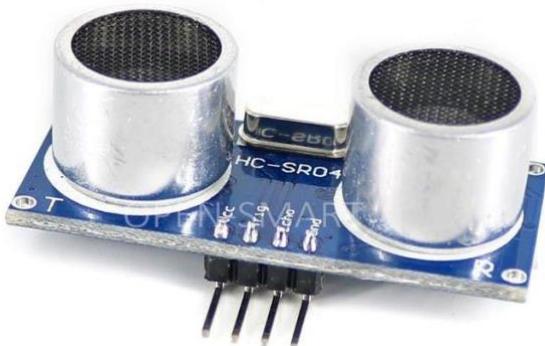
Download the XOD code for this tutorial [here](#), and open in the XOD IDE. You will also need to download some additional libraries:

- `cesars/rich-uno-r3`: contains a number of useful nodes for controlling the Rich UNO R3 board, including alternative nodes for the onboard buzzer, knob and temperature sensor.
- `cesars/tm1637-v2`: builds on the `cesars/tm1637` library that we used in [Chapter 7](#), allowing additional functionality of the 4-digit display.
- `cesars/utils`: contains nodes required by the two previous libraries.

- `gweimer/utils`: contains some useful nodes for manipulating data.

Step 2: Connecting the Ultrasonic Sensor

Your Biomaker starter kit contains an ultrasonic sensor (HC-SR04 sonar device) which is supported by XOD nodes in the `xod/common-hardware` library. The `hc-sr04-ultrasonic-range` node provides the last measured range in metres, and the `hc-sr04-ultrasonic-time` node provides the roundtrip time of the last pulse in seconds.



Ultrasonic Sonar Range-Finder Sensor

The sensor device contains both transmitter and receiver elements. The ultrasonic transmitter emits pulses of inaudible 40,000 Hz sound waves, which travel through the air and will bounce back to the module if there is an object or obstacle in the path. The module has a receiver that allows measurement of the travel time. Knowing the speed of the sound allows calculation of the distance. The device can be used for measuring water levels, movement of organisms and proximity for robots.

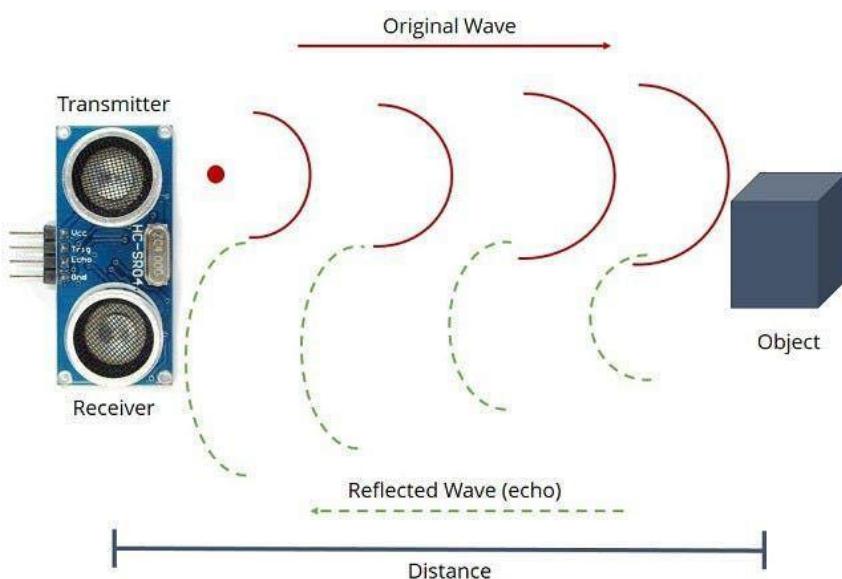
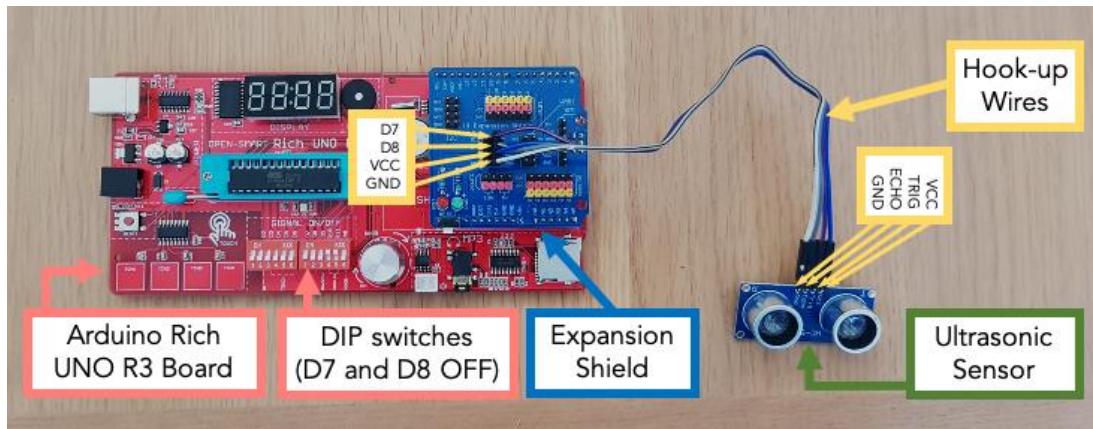


Illustration of the Ultrasonic Sensor Function

Connect the expansion shield to the Rich Arduino UNO board, then connect the ultrasonic sensor to the expansion shield using the digital pin set. Make sure to connect GND-GND, VCC-VCC and TRIG and ECHO to two digital pins. We suggest using pin D7 for TRIG and

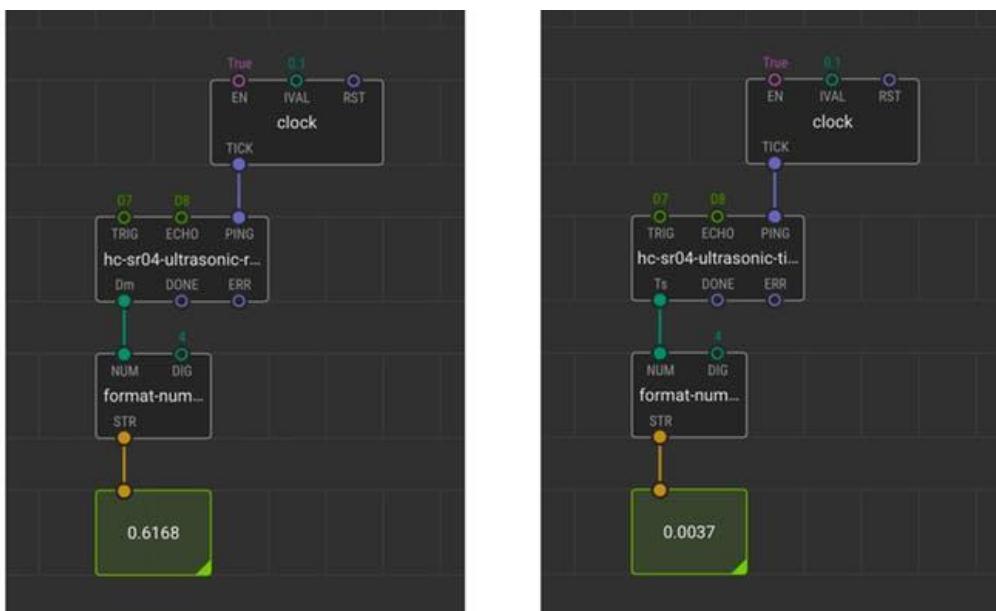
D8 for ECHO. These are usually connected to the MP3 module, but this can be switched off using the DIP switches for D7 and D8. Turning off the unneeded onboard devices with the DIP switches avoids unwanted clashes with inbuilt devices.



Wiring of the Ultrasonic Sensor to the Arduino Rich UNO R3 Board

Step 3: Reading Data from the Ultrasonic Sensor

Once you have wired up the ultrasonic sensor, you can test it using the *01-sonar-time* and *02-sonar-range* patches in this chapter's XOD file (download [here](#)). These patches include a clock node to set a 0.1 second sampling rate, and read either the range (metres) or the time (seconds) for pulse to return, as measured by the sensor. In addition, a *format-number* node and *watch* node allow display of the values with four decimal places. Try placing an object close to your sensor and see how the numbers change.

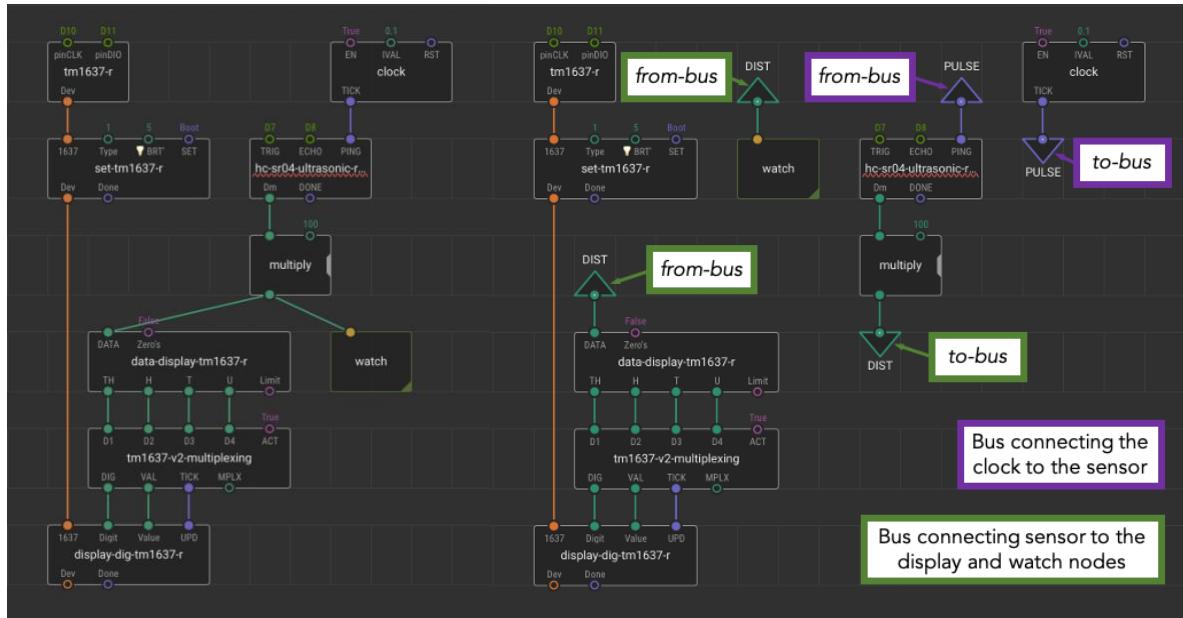


The 01-sonar-range and 02-sonar-time Patches Measuring Distance and Time to the Nearest Object

Step 4: Using Buses to Show Data on the 4-digit Display

If you have not done so already, install the libraries *cesars/rich-uno-r3*, *cesars/tm1637-v2*, *cesars/utils* and *gweimer/utils* now. Then open the next patch in the tutorial *03-sonar-to-4digit-display*. This patch takes several functions we have already used (measuring from the

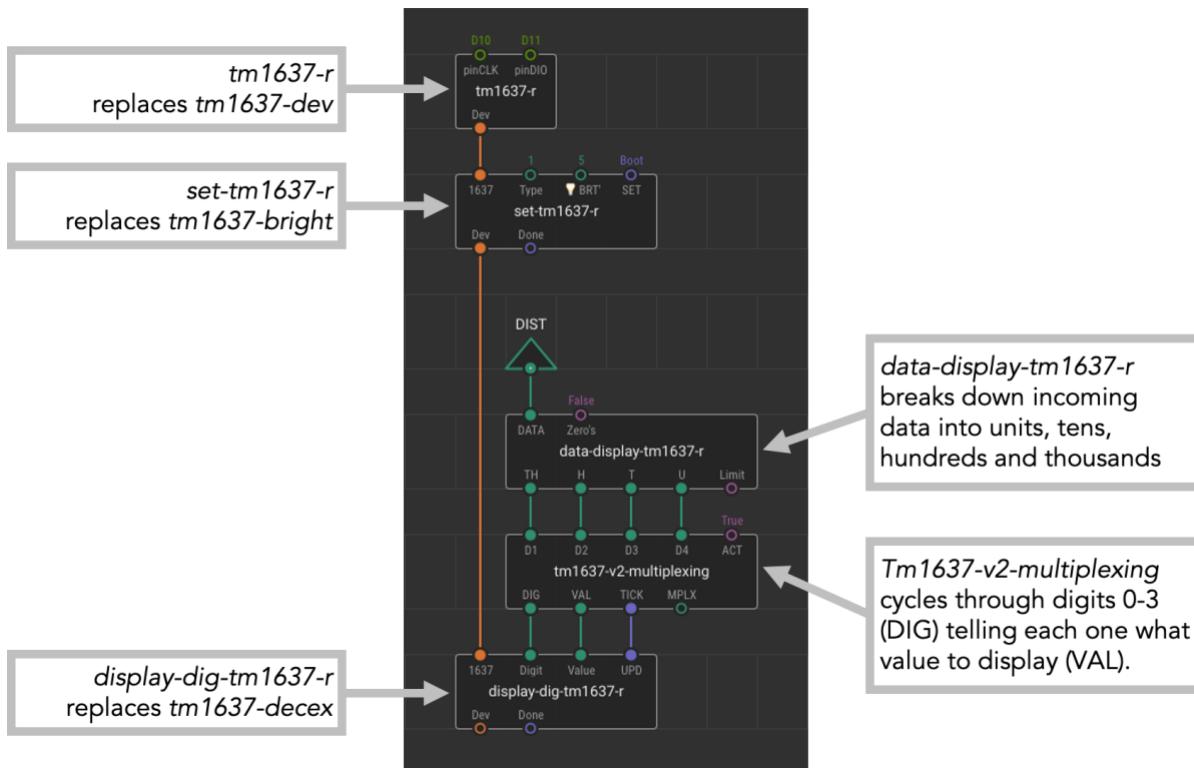
ultrasonic sensor, and displaying on the 4-digit display) and combines them using “[buses](#)”. Buses are an alternative way to link node pins which can reduce the visual complexity of the patch and make it easier to understand what is going on. This function uses two nodes from the *xod/patch-nodes* library: *to-bus* and *from-bus*. The node *to-bus* defines a new bus that stores an output. The node *from-bus* sources output data from the *to-bus* of the same name and inputs it into another node. Essentially, they provide inputs and outputs that are linked but don’t need to be physically connected, making the patch simpler. In this instance, using buses makes a relatively small difference, but in complex patches it can help to separate different modules, and use buses to connect them. The patch also uses a *multiply* node to multiply the distance output by 100, and give a readout in centimeters rather than meters. Try uploading the patch and see how the data is displayed on your screen.



Different XOD Patches to Display Distance from the Ultrasonic Sensor on the 4-Digit Display
Left: Standard XOD Patch with Each Node Connected
Right: XOD Patch Using Buses to Connect Different Modules

You may have noticed that this patch uses a slightly different set of nodes to represent the 4-digit display than the ones used in the previous chapter. In [Chapter 7](#) we used nodes from the *cesars/tm1637* library, and this patch uses nodes from the *cesars/rich-uno-r3* library. The two sets of nodes can't be combined, but are alternatives, which allow us to display data in a slightly different way. Which set of nodes you chose to use to display data is up to you, and you can play around with the different sets to see what you can achieve with each one.

There are also two new patches *data-display-tm1637-r* and *tm1637-v2-multiplexing*. The node *data-display-tm1637-r* breaks down incoming data into its constituent units, tens, hundreds and thousands. You can see what this looks like by adding *watch* nodes to each output. The node *tm1637-v2-multiplexing* uses this information to set each digit on the display one at a time. It cycles through the digits (DIG) from 0 (leftmost digit on the display) to 3 (rightmost digit on the display) and sets each one according to the values (VAL) input from the *data-display-tm1637-r* node.



Alternative XOD Patch to Show Data on the 4-Digit Display

Step 5: Data Averaging

The raw values from sensors like the HC-SR04 can be noisy, with values varying over time. The *gweimer/utils* library provides a *running-avg* node where successive values can be accumulated, and a running average established for a set number of values. This can help provide a smoothed set of values, and minimise short term fluctuations that can interfere with some forms of display. Open the next tutorial patch *04-data-averaging*. In this patch data is collected and sampled each 0.1 seconds, and both raw and 10x averaged data are fed to watch nodes. The rate of sampling can be changed by changing the interval of the *clock ticks* (IVAL), and number of accumulated values can be altered by dragging the right hand side of the *running-avg* node back and forth to display different numbers of “*dummy*” pins. Try experimenting with how the data is displayed in this patch.

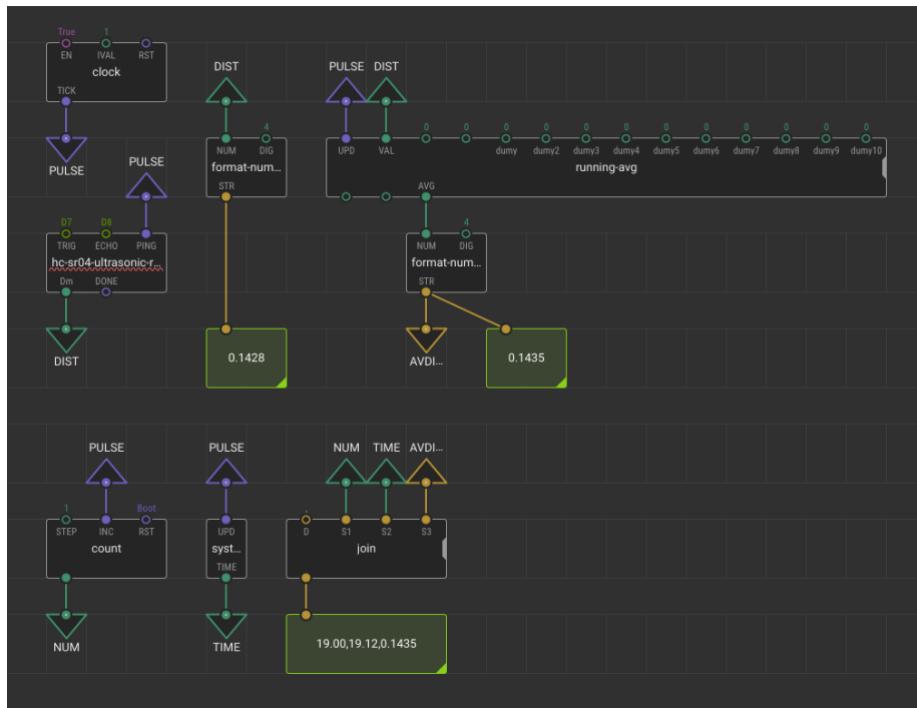
Step 6: Adding a Timestamp

In order to collect a permanent record of our data we will need to number each record, as well as log the time at which it was taken. We can use a *count* node to number records, and a *system-time* node to provide a time-stamp. This node outputs the time (in seconds) since the programme started.

Add a *count* node and a *system-time* node to the *04-data-averaging* patch. We need to set a frequency of readings to limit the data volume and to give some time for the micro SD card to flush the file after each record. Change the IVAL pin of your clock node to 1, meaning that the system will take a reading every second. Add two *from-bus* nodes and connect them to the INC pin of the *count* node and the UPD pin of the *system-time* node. Set them to PULSE. This will connect them make sure that the record number increases and the time updates each second. Now add two *to-bus* nodes and connect them to the *count* and *system-time* outputs. Name the *count* output NUM and the *system-time* output TIME. In the

current patch the DIST output comes straight from the sensor. Add a *bus-to* node to the string output that comes out after the *running-avg* and *format-number* nodes to get the averaged output. Call this AVDIST.

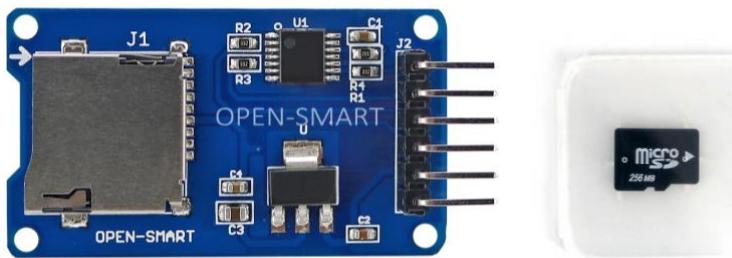
Now we have three pieces of data: the record number (NUM), the time since the program started (TIME) and the distance from the sensor averaged over 10 reads (AVDIST). Let's bind this information together using a *join* node. Add a *join* node to the patch and use three *from-bus* nodes to input the data from NUM, TIME and AVDIST. The D pin determines what character is used to separate these values. There are numerous text-based formats for storing and exchanging data. The examples of such formats are CSV (Comma-Separated Values) or TSV (Tab-separated values). These text formats are used to store tabular data and to exchange it between different computer programs. Notably, spreadsheet applications such as Google Spreadsheets, Microsoft Excel, LibreOffice Calc can easily import such files. The names suggest, CSV files use commas to separate values, whilst TSV files use tabs. To save your data in CSV format set D to a comma “,”, to save your data in TSV format set D to “\t”. Add a watch node to the join output, upload and debug your patch to see how the data is presented.



XOD Patch to Time-Stamp Data Readings

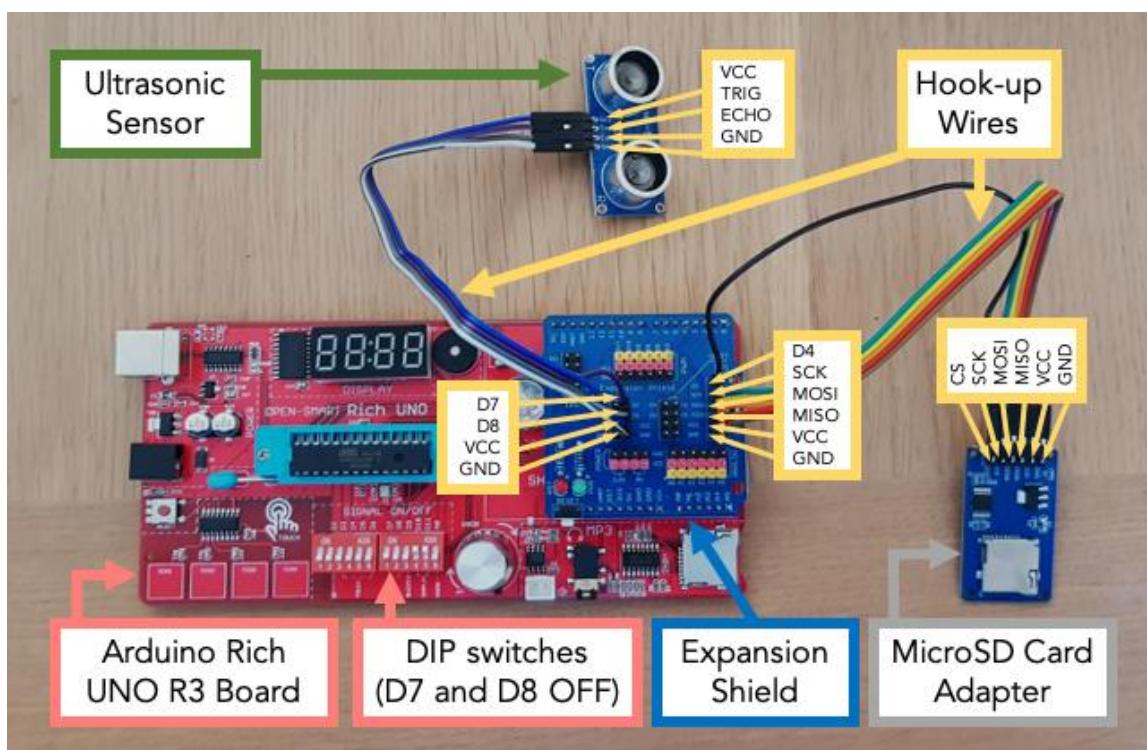
Step 7: Connect the MicroSD Card Adapter

The microSD card adapter provided in the Biomaker starter kit sends and receives data via an SPI interface (see [Chapter 3](#)). This means it has VCC and GND power lines (as with other modules we've used) and then four lines for the transmission of data: MOSI sends data from the board to the module; MISO sends data from the module to the board; the serial clock line (SCK) synchronises data; and chip select (CS) acts as the slave select line, which will select the correct pin, if more than one module is attached to the SPI pins.



Open Smart MicroSD Card Adapter and MicroSD Card

Using the SPI set on the right hand side of the expansion shield, wire VCC-VCC, GND-GND, SCK-SCK, MOSI-MOSI, MISO-MISO and CS-D4 (this is the pin we will use to identify the microSD card module). Once the adapter module is attached you can insert the microSD card provided into the microSD card slot. To remove it again, push gently inwards and the card will release.



Wiring of the Ultrasonic Sensor and MicroSD Card Adapter to the Arduino Rich UNO R3 Board

Step 8: Writing Data to the MicroSD Card

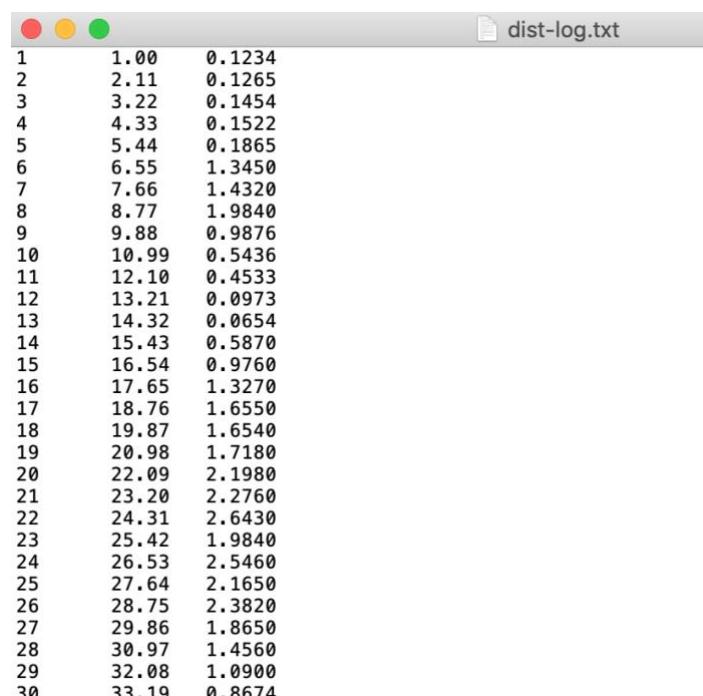
To write data to the microSD card we can use the *sd-log* node provided in the *xod/common-hardware* library. Add an *sd-log* node to the *04-data-sampling* patch. This node takes four inputs. CS determines the chip select (or slave select) line used to identify the module, in this case D4. FILE determines the file name, so enter a file name of your choice e.g. dist-log (the file will be saved as a .txt file, so in this case the final file will be called dist-log.txt). LINE determines a line of data to be written to the file. Connect the output of your *join* node to this pin, either directly or via a bus. The W pin writes to the SD card each time a pulse is sent. Connect this to the clock module output so that data is logged every second.

Try uploading this patch to your board. You may get an error as the patch is too large to load onto the Arduino board. In this case you can delete the running average module of the patch, and revert the input on the *join* node to DIST rather than AVGDIST. This way you will collect the raw data, and can create a running average later in analysis if required.

Upload this patch to your Arduino board and record data for a short amount of time (30-60sec). Try moving an object closer or further from your board to create some variation in the data. If you are having trouble with the XOD patch, a completed version is available in the tutorial patch *05-writing-data-to-sd*. You may connect and disconnect the device from the power source as many times as you wish. On every boot the log will be continued, so the data from previous sessions won't be lost.

Step 9: Accessing your Data

To access the stored data you will need a suitable microSD card reader/adapter to connect to your computer. This is not provided in the Biomaker starter kit. Open the .txt file on your computer. Your data will look something like this:

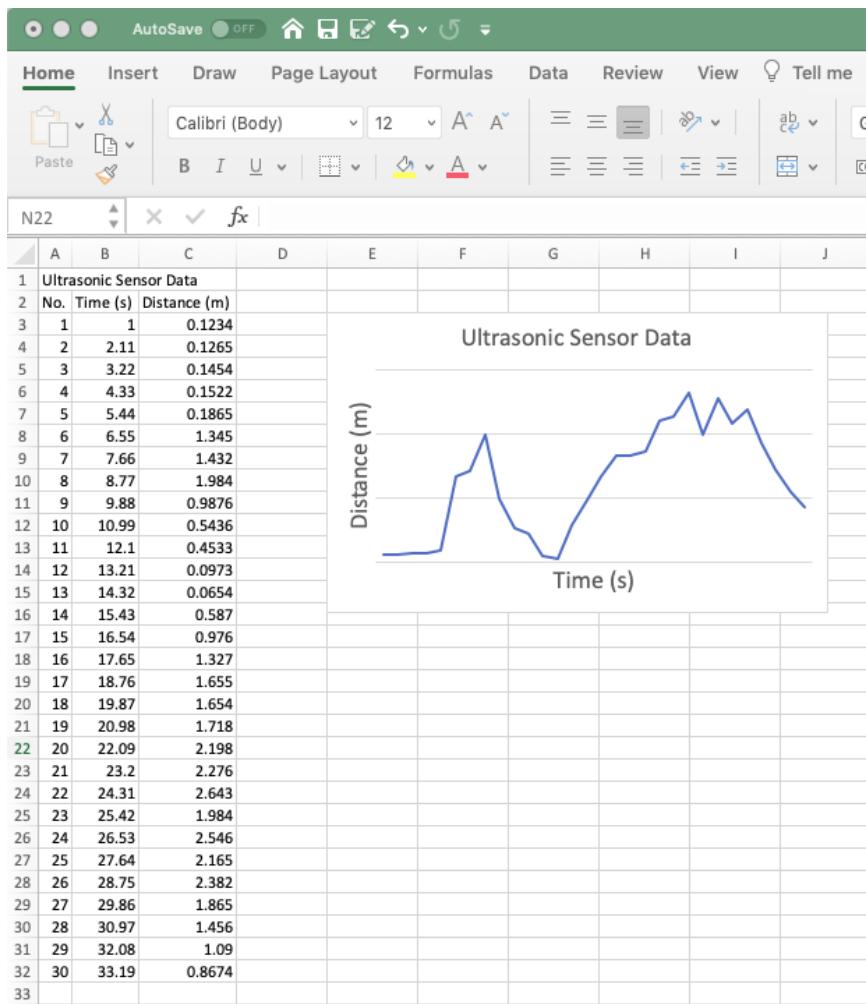


The screenshot shows a text editor window with three colored status indicators (red, yellow, green) at the top left. The title bar reads "dist-log.txt". The main area contains 30 lines of data, each consisting of three columns separated by spaces. The first column is a number from 1 to 30, the second column is a value ranging from 1.00 to 33.19, and the third column is a value ranging from 0.1234 to 0.8674. The data represents distance measurements over 30 samples.

1	1.00	0.1234
2	2.11	0.1265
3	3.22	0.1454
4	4.33	0.1522
5	5.44	0.1865
6	6.55	1.3450
7	7.66	1.4320
8	8.77	1.9840
9	9.88	0.9876
10	10.99	0.5436
11	12.10	0.4533
12	13.21	0.0973
13	14.32	0.0654
14	15.43	0.5870
15	16.54	0.9760
16	17.65	1.3270
17	18.76	1.6550
18	19.87	1.6540
19	20.98	1.7180
20	22.09	2.1980
21	23.20	2.2760
22	24.31	2.6430
23	25.42	1.9840
24	26.53	2.5460
25	27.64	2.1650
26	28.75	2.3820
27	29.86	1.8650
28	30.97	1.4560
29	32.08	1.0900
30	33.19	0.8674

Example of Sensor Data Stored as a .txt File

To access this data in a more useful way, you can import the .txt file into a program such as Excel or Numbers. You can do this by going to “Open” or “Upload” within the relevant programme, or by right-clicking the file and selecting “Open with”. When you open the file, you may be asked to select the correct format, and how the data are delimited (e.g. by commas or by tabs). This will depend on whether you set the D pin of the join node to comma (,) or tab (\t). Once uploaded you can proceed to analyse the data however necessary.



Excel File and Graph Showing Data Collected from the DIY Data Logger

Step 10: Extensions and Applications

With the microSD card, you can store a significant amount of data that can not be stored in the memory of the controller. You can use any other sensor or even several different sensors to log and observe a physical process of your choice.

Using this programme as a base, you can try to:

- Log data from a different sensor, for example, the light sensor, water sensor, or onboard temperature sensor.
- Add error nodes (e.g. `has-error`) to make an LED light up if there is an error writing to the sd card (see [Errors - XOD](#) for further guidance)
- Make an alarm system using the onboard buzzer to signal if an item gets too close to the sensor

Although this system is relatively simple it provides the basis for a wide range of devices with biological applications. Previously, biomaker projects have used sensor and data logging modules such as this for a number of applications in both the lab and the field. For example:

- [eCO-SENSE](#)

A sensor to monitor soil conditions powered by plant growth. This project uses a temperature sensor, moisture sensor and gas sensor to take measurements of soil

conditions and adds a bluetooth module to send the data wirelessly to a phone or computer.

- [Variable-Time Camera for Monitoring Plant Pollination Events](#)

A video and time-lapse monitor to record pollinators interacting with plants. This project included an environmental sensor to measure temperature, humidity and barometric pressure and wrote this data into the image file names for later analysis.

- [Low Cost Oxygen Sensor for Bioreactors](#)

This project developed a custom low-cost oxygen sensor to measure oxygen levels in cultures of *E. coli*, and logged the data via an Arduino microcontroller.

- [A Behavioural Chamber to Evaluate Rodent Forelimb Grasping](#)

This project uses a light emitter and light sensor to monitor when a rodent moves past a certain threshold, and triggers release of a sugar pellet when it does.

- [Biofab Incubator](#)

A low cost programmable incubator for growing mycelium textile. This project uses sensors to measure temperature and humidity and feeds this information back into the system to maintain constant environmental conditions.

Further Information

For more information some of the topics discussed in this tutorial, see the links below.

Buses - XOD: <https://xod.io/docs/guide/buses/>

Errors - XOD: <https://xod.io/docs/guide/errors/>

HC-SR04 Ultrasonic Sensor Data Sheet: <https://components101.com/ultrasonic-sensor-working-pinout-datasheet>

Chapter 9: Additional Tutorials and Useful Tools

This chapter provides information on additional tools and tutorials that are useful for building custom bioinstrumentation.

Introduction

Having completed the tasks in previous tutorials, including making your own data logger in [Chapter 8](#), you will now have a good understanding of the basics of programming in XOD and how to go about building low-cost instruments for use in biology. The final chapter of this handbook provides a number of additional tutorials, and highlights some tools and skills that can be very useful for developing your own hardware, including how to use an [OLED screen](#), how to use the [infrared emitter-sensor system](#), and how to [implement sequences](#) in XOD. On completion of this chapter you should be equipped with the knowledge and skills required to start building your own custom hardware, and an understanding of where to look for additional help and expertise if required.

Objectives

- Learn how to use an OLED screen to display graphics
- Use the infrared remote to switch board components on and off
- Understand how to programme sequences in XOD
- Know where to find additional tutorials and projects
- Know where to find additional help and support with no-code-programming and building bioinstrumentation

Requirements

- Computer running MacOS, Windows or Linux
- Arduino Rich UNO R3 board
- Expansion shield
- I2C OLED 128x64 display
- IR remote controller
- Hook-up wires/Dupont line

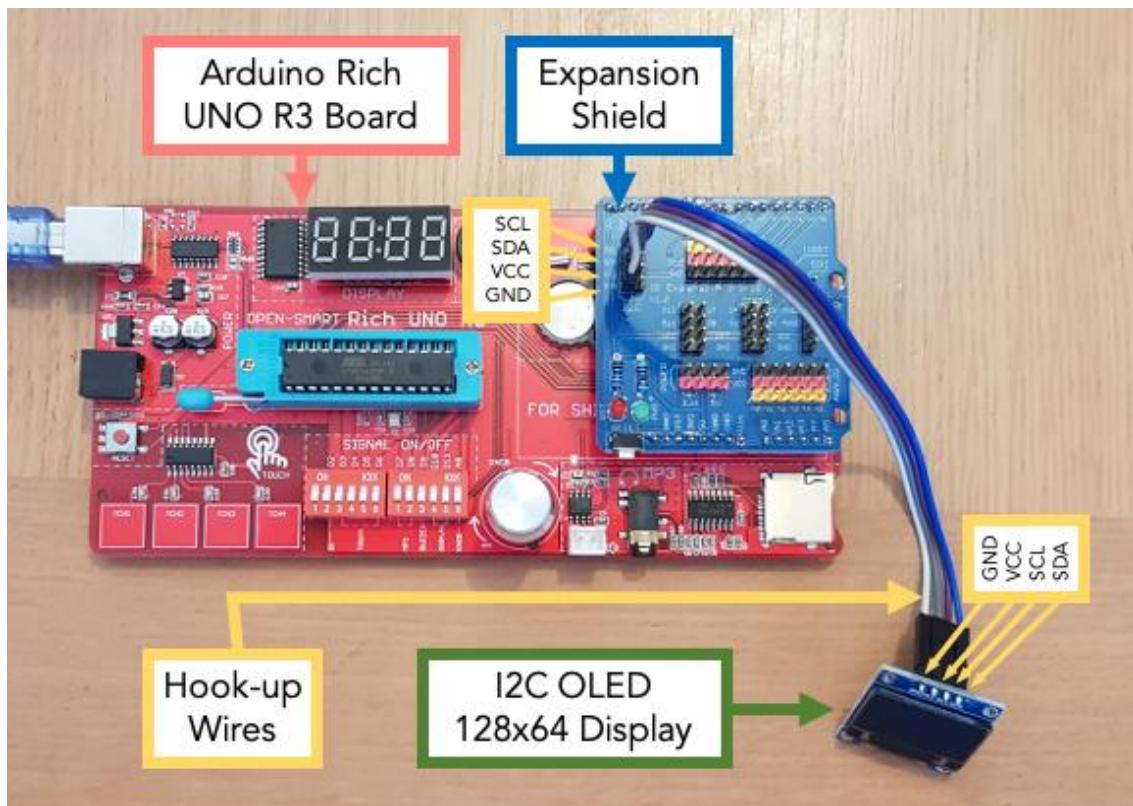
Using the OLED Screen

The Biomaker starter kit includes an I2C OLED 64x128 display, which can be useful for displaying data or images other than the simple numbers and text that can be displayed on the 4-digit and LCD screens. Matt Wayland has created a very useful XOD library for control of OLED screens driven by an SSD1306 chip, including the one provided in the Biomaker starter kit: <https://xod.io/libs/wayland/ssd1306-oled-i2c/>.



Open Smart I2C OLED 128x63 Display

Connect the OLED display to the Arduino board via the expansion shield. The screen is controlled via an I2C interface, with address 3Ch, so connect it via the I2C pin set: GND-GND, VCC-VCC, SCL-SCL and SDA-SDA.



Arduino Rich UNO R3 Board Wired to the I2C OLED 128x64 Display

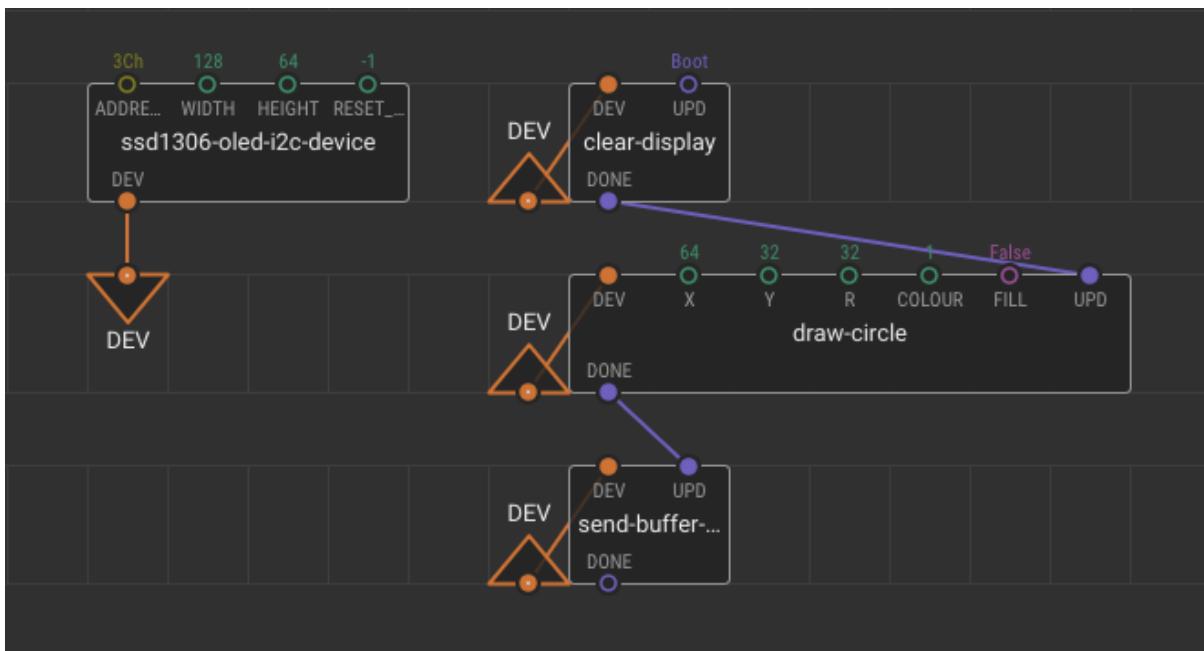
Download the library `wayland/ssd1306-oled-i2c` in XOD. This library contains a number of examples showing how to draw shapes, how to scroll and how to display the XOD logo. To understand how these nodes and patches work we will start by making a simple patch of our own.

In a new patch add the following libraries from the `wayland/ssd1306-oled-i2c` library:

- `ssd1306-oled-i2c-device`, this node represents the OLED screen, the default pin values of ADDRESS=3Ch, WIDTH=128, HEIGHT=64 and RESET_PIN=-1 are correct for our screen.
- `clear-display`, this node clears the screen before you start, otherwise the Adafruit company logo is displayed.
- `draw-circle`, this node draws a circle on the screen. It takes input in the form of coordinates for the centre of the circle (X and Y), the radius of the circle (R) and colour (1 for light pixels, 0 for dark pixels). Set the colour pin to 1. We want the circle in the middle of the screen, so set X to half the screen width (64) and Y to half the screen height (32). We want the circle to fill the screen, so set the radius to half the screen height (32). (*Note: you can also use the get-display-dimensions node in this library to output the screen dimensions, use a divide node to set a specific ratio, e.g. divide by two, and input these numbers into the draw-circle node - this is how the example-draw-shapes patch works*). If FILL is true you will draw a filled circle, if it is false you will draw the outline of a circle.
- `send-buffer-to-display`, this node sends the information above to the screen. Nothing will be visible on the screen until this function is called.

The DEV output of the `ssd1306-oled-12c-device` node will need to be connected to the DEV input of each of the other nodes. The best way to do this is using the *to-bus* and *from-bus* nodes (see [Chapter 8, Step 4](#)). Make sure the UPD pin of the `clear-display` node is set to 'Boot' so that the sequence starts when the program is booted. Next you will need to connect the OK or DONE output from each node to the UPD pin of the subsequent node, in the order `clear-display`, `draw-circle`, `send-buffer-to-display`. Your patch should now look something like the one below. Try uploading this patch to the board and see what happens. Try adding *tweak* nodes to the inputs of the `draw-circle` node. Play around with different values to see how the circle is displayed. (*Note: you will also need to set the UPD pin of the clear-display node to continuously, so that the screen updates as you tweak the values*).

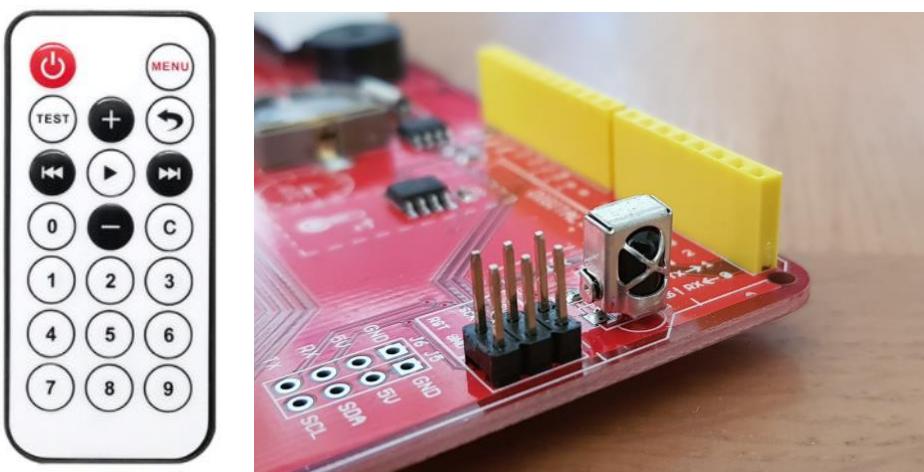
Now try exploring the library examples `example-draw-shapes`, `example-scroll` and `example-xod-logo`. See if you can work out how the patches work, then try uploading them and playing with the parameters to see what you can achieve. (*Note: in the example-scroll patch, you will need to change the DIR pin of the start-scroll node to 00h = left-handed scroll, or 01h = right-handed scroll, as the diagonal scrolls do not work with our set-up*).



XOD Patch to Draw a Circle on the I2C 128x64 OLED Display

Using the Infrared Remote Control

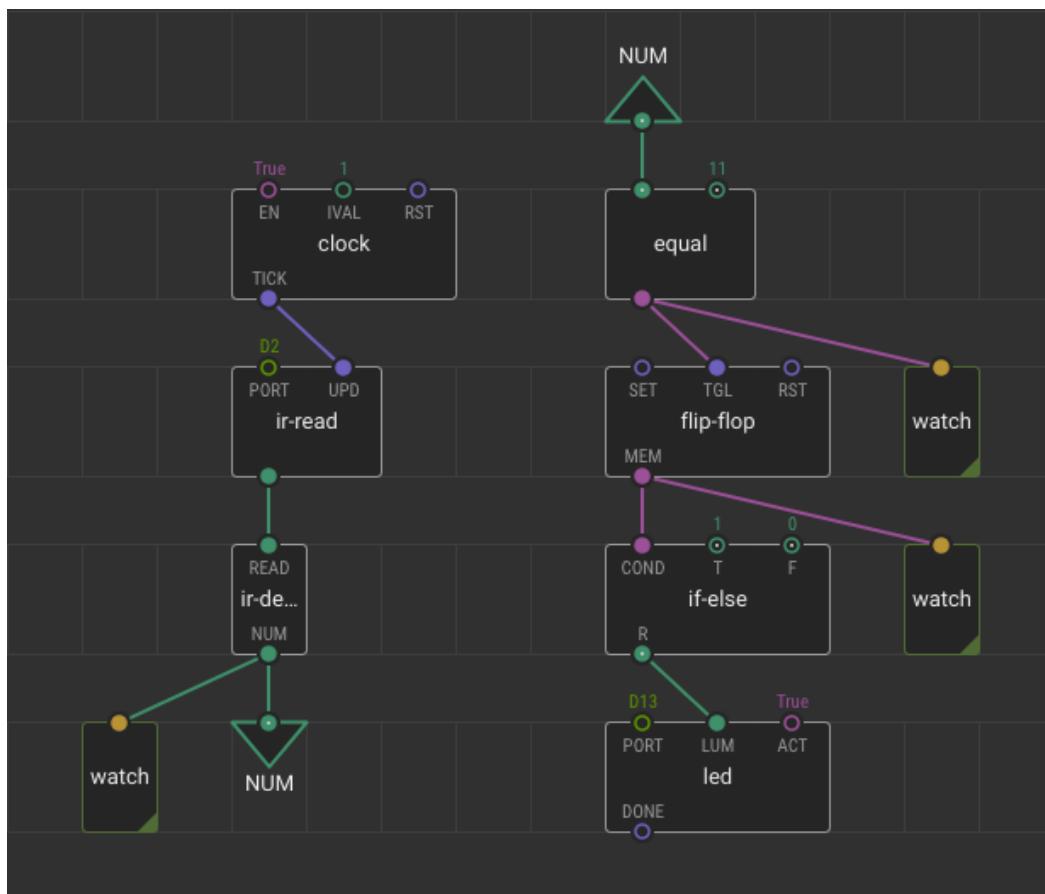
The infrared remote control and onboard sensor module can be extremely useful for controlling the Arduino board without needing to access it directly, for example, if the board is encased as part of an instrument. To use the IR remote control you will need to download the *gweimer/ir-remote* and *san43/ir-remote-opensmart* libraries in XOD.



Open Smart IR Remote and Onboard IR Sensor on the Arduino Rich UNO R3 Board

The *gweimer/ir-remote* library contains the *ir-read* node, which identifies the IR sensor and reads raw data sent from the remote. The *san43/ir-remote-opensmart* library contains the *ir-decode-opensmart* node, which decodes the raw data from the *ir-read* node and turns it into a useful code using 0-9 to represent number buttons, 11-20 to represent special character buttons, and 255 to represent no data signal. This node is specific to the layout of the Open Smart remote. You can find a key identifying which buttons are represented by which numbers in *00-ir-opensmart-key* of the *san43/ir-remote-opensmart* library.

Open *01-led-on-off-example* from the *san43/ir-remote-opensmart* library. Upload and debug this patch. Use the watch node on *ir-decode-opensmart* to see how the number output changes as you press different buttons. Notice that nothing else happens unless you press the ON/OFF button. This patch uses an *equal* node to determine whether the signal from the remote matches a certain value (in this case 11, representing the ON/OFF button). Then the information passes through a *flip-flop* node to toggle between a TRUE and FALSE state each time the button is pressed. The *if-else* node tells the board what to do when the state is TRUE and what to do when it's FALSE. In this case TRUE will turn on a LED by setting its brightness to 1 and FALSE will turn off the LED by setting its brightness to 0. This on/off mechanic is useful, and could easily be used to control an instrument from a distance, or stop and start a program at will. Alternatively different buttons could be used to control different functions on a custom built instrument.



XOD Patch to Turn and LED On and Off with an Infrared Remote

Now that you understand this patch, try making your own patch using the IR remote to control different functions on the board and added modules. For example, you could:

- Use different buttons on the remote to play different notes on the buzzer
- Use the remote to stop and start recordings on your data logger
- Use buttons on the remote to initiate different programmes on the neopixel ring

Control of Sequences: Traffic Light

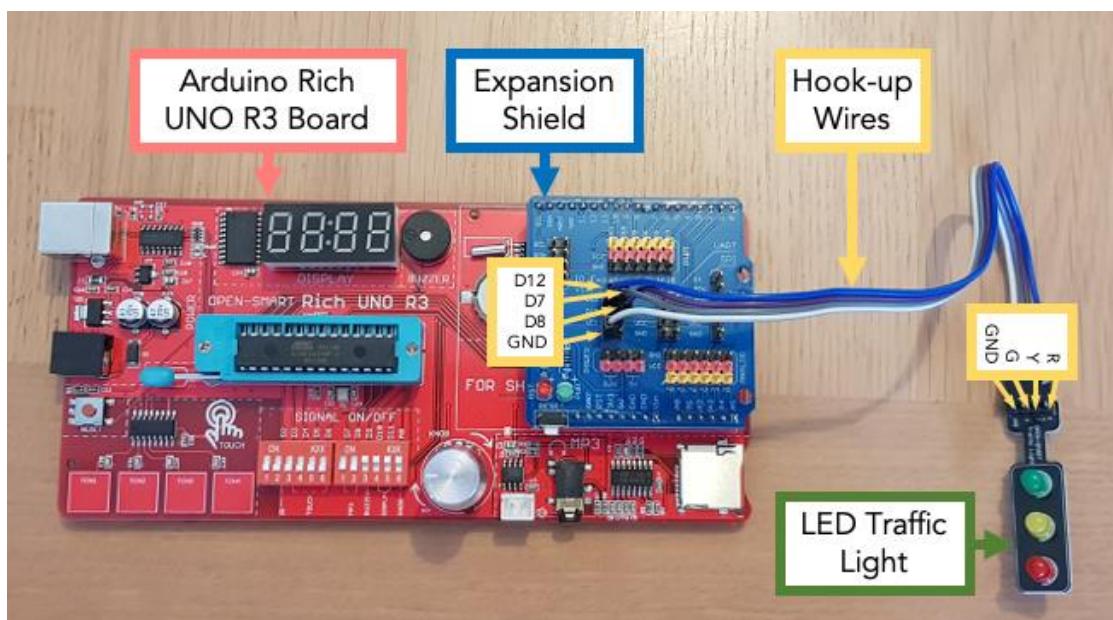
The [traffic light tutorial](#) on the XOD website provides an excellent example to introduce yourself to the basics of creating sequences in XOD. This is a very useful tool for creating functional programmes. In this example XOD uses a traffic light to show a system that works sequentially. In its simplest form a traffic light has three states: Green - go, Yellow - stop if you can and Red - stop! Each state is active for a certain time interval and when done everything starts from the beginning and repeats again and again. The tutorial describes the implementation of a state machine, which uses one patch node per state and a patch that wires each state patch together. These instructions contain important information on how to use *delay* and *defer* nodes to set up sequences. The instructions and code can be found at: <https://xod.io/docs/guide/simple-traffic-light/>

Adaptations for the Rich UNO R3 board

Rather than using three individual LED lights, the Biomaker starter kit provides an Open Smart traffic light module. Wire this to the Arduino Rich UNO R3 board, via the expansion shield. You can use either the digital pin set, or the PWM pin set. In the example below I have used the digital pin set, and wired GND-GND, Green(G)-D7, Yellow(Y)-D8 and Red(R)-D12. Make sure to set each state patch (green, yellow and red) so that the LED node has the correct corresponding pin number, and use the DIP switches to turn off any onboard devices and avoid clashes.

Once complete, try experimenting with the sequence. For example, you could:

- Change the amount of time each LED is on
- Start the sequence from a different state
- Try using a sequence to control a different board function, e.g. play a tune using different notes on the buzzer



Arduino Rich UNO R3 Board Wired the the Open Smart Traffic Light Module

Handbook Summary

This online handbook has been designed to give you the basic skills required to start building your own custom instruments for measuring and controlling biological systems.

We have covered how to use the visual programming language XOD to:

- control your hardware ([Chapter 5](#))
- make your own nodes and patches ([Chapter 6](#))
- read and write data ([Chapter 8](#))
- run sequences ([Chapter 9](#))

We have shown you how to use the majority of the Arduino Rich UNO R3 onboard devices including:

- buzzer ([Chapter 5](#))
- touch buttons ([Chapter 5](#))
- knob potentiometer ([Chapter 6](#))
- real-time clock ([Chapter 7](#))
- temperature sensor ([Chapter 7](#))
- 4-digit display ([Chapter 7](#))
- infrared receiver ([Chapter 9](#))

We have explored how to use the expansion shield to add functionality to the board, and access a variety of additional components provided in the Biomaker starter kit, including:

- eagle eye LED ([Chapter 5](#))
- I2C 1602 LCD display ([Chapter 6](#))
- light sensor ([Chapter 6](#))
- RGB LED ring ([Chapter 6](#))
- 4-digit display ([Chapter 7](#))
- ultrasonic sensor ([Chapter 8](#))
- microSD card adapter ([Chapter 8](#))
- I2C OLED 128x64 display ([Chapter 9](#))
- LED traffic light ([Chapter 9](#))

We hope that this gives you both an idea of the possibilities available when making custom instrumentation, and the basic skills required to move forward and start working on your own projects and instruments.

Where to Find Additional Help

The XOD Website

The [XOD website](#) and [forums](#) contain a wealth of information for anyone interested in exploring the possibilities of no-code programming. The website provides a series of beginners tutorials [here](#), and some excellent guides on concepts, making your own nodes and libraries, and how to use different interfaces and internet connections [here](#). The XOD community and XOD forum are also very useful. The forum contains a lot of useful information, and you can ask questions by signing up for a free account [here](#).

Hackster

If you are looking for inspiration for what to build with your Biomaker starter kit, you can find a wealth of ideas on [Hackster.io](#). There are a few [XOD-based projects](#) already available, and more are being added as the no-code community grows. There is also a wealth of [Arduino-based projects](#) available covering a wide range of topics, such as robotics, environmental sensors, and wearables. Whilst most of these projects use coding languages to program their Arduino boards, these can be easily adapted for use with XOD.

Biomaker Challenge

If you are interested in building an instrument to apply to your own work, or in learning more about “Biomaking”, the annual Biomaker challenge allows small teams to apply for funding to develop their own bioinstrumentation. You can find out more about previous Biomaker projects on our [website](#). The format of the Biomaker 2020 competition will depend largely on how the current situation and COVID-19 pandemic progress, but you can keep up to date by following the SynBio IRC on Twitter [@SynBioIRC](#), or by signing up to our [newsletter](#).

SynBio IRC

Finally, if you have any questions or queries, please feel free to get in touch by email at synbio@hermes.cam.ac.uk.

Annex: Biomaker Starter Kit Specifications

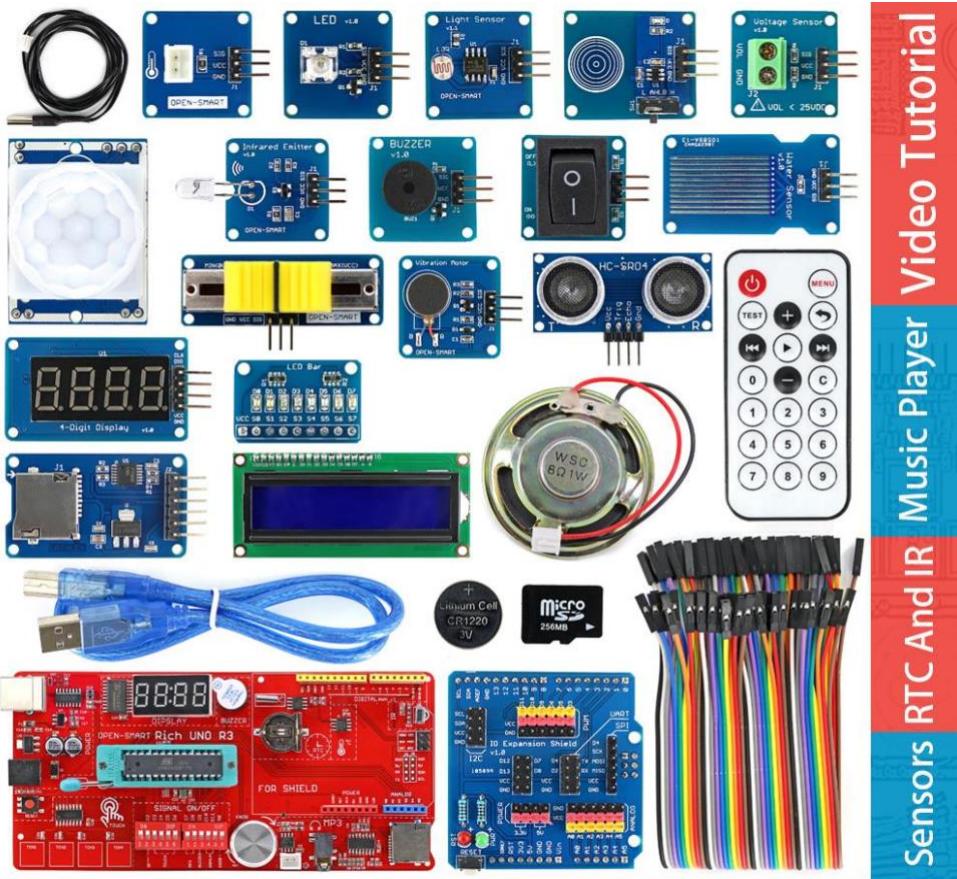
Everything you ever wanted to know about the Open Smart Rich UNO R3 Arduino board used in the 2020 Biomaker Starter Kit.

Biomaker Starter Kit: Rich UNO R3 board & parts

The 2020 Biomaker starter kit consists of an extended Arduino board, a collection of electronic components, a small prototyping board and a programmable display. It is based on the Open-Smart Rich UNO R3 board, which contains a variety of embedded components, and is accompanied by a variety of electronic components and sensors which are useful for monitoring and programming of biological systems. Importantly, the connections to the board's embedded components can be turned off using a DIP switch, meaning that information can be stored on the device even after shutdown. The board is Arduino UNO compatible, and can be programmed directly from the visual programming software, XOD.

Further, the board is available from Open Smart as a kit of components that includes a wide range of sensors and displays. The kit includes: the Arduino board and: IO Shield, Voltage sensor, Ultrasonic sensor, Touch sensor, Water sensor, PIR motion sensor, Rocker switch, NTC sensor, Light sensor, Slide Potentiometer, Vibration motor, Passive buzzer, Speaker, 8 LED bar, Eagle eyes LED, I2C 1602 LCD, 4-Digit display, microSD card (256MB), CR1220 button battery (40mAh), Infrared Remote Control (with one CR2032 battery), Micro SD card adapter, Infrared emitter, 40pin female to female cable, USB cable (50cm). The extended kit provides a wide range of sensors, actuators and displays that are programmable from XOD and can form the basis for an extended set of tutorials - as well as provide parts for applications.

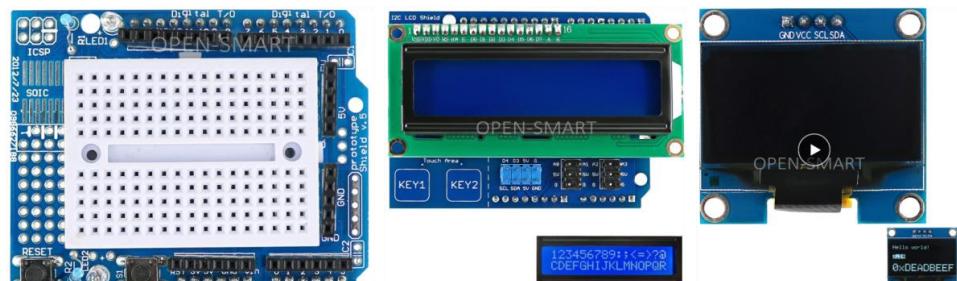
In addition to this Open Smart kit, the 2020 Biomaker kit also contains several extra components, including: a prototyping shield, an additional shield with LCD display, an RGB LED ring, an OLED display, gas sensors and an LED traffic light.



RTC And IR Music Player Video Tutorial

Sensors

Multifunctional Kit for Arduino Rich UNO R3 Kit (B) with MP3 and Sensors



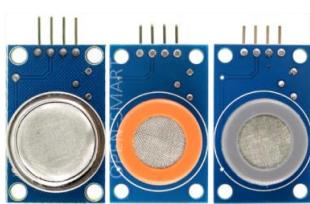
Prototyping Shield

Shield with I2C 16x2 LCD Display

I2C OLED 64x128 Display



RGB LED Ring



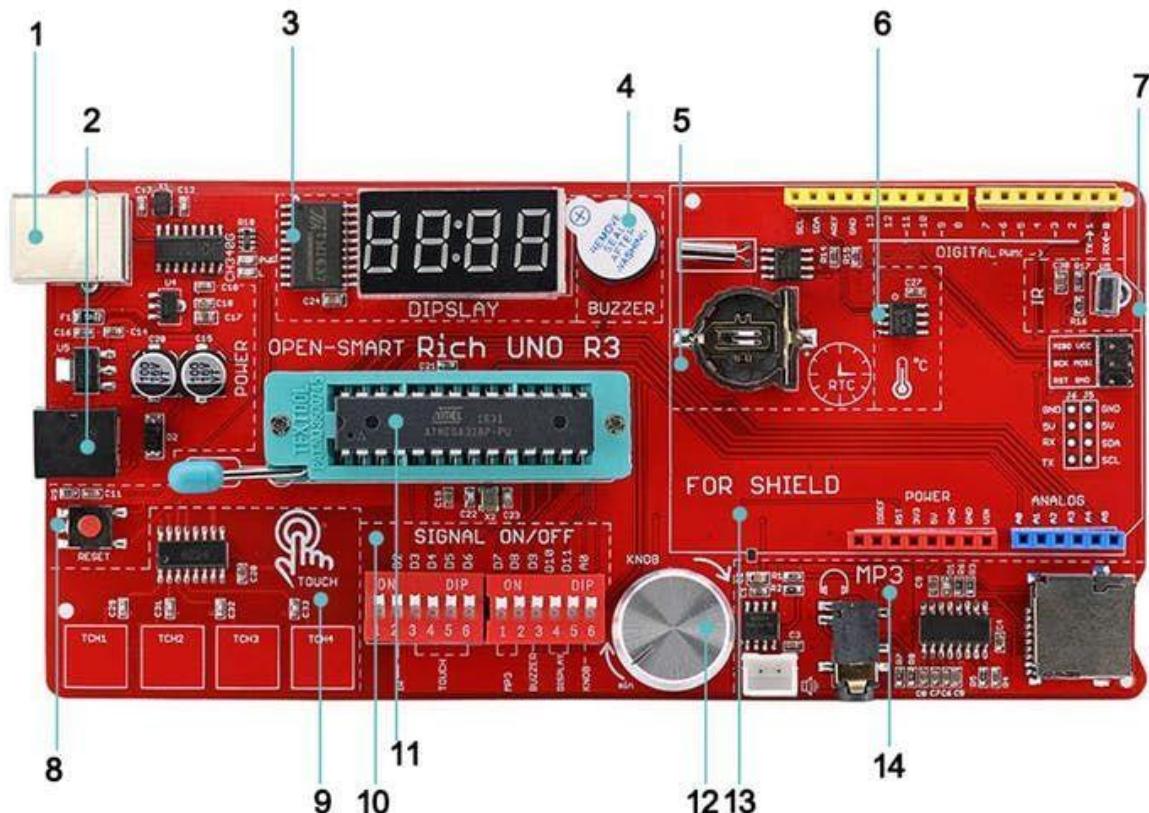
Gas Sensors



LED Traffic Light

Additional Components

Open Smart Rich UNO R3 Specifications



1. USB Power and Download Port

Type B USB connector, consistent with Arduino UNO R3, for ruggedness and long service life. USB interface driver chip: CH340G, compatible with win7, win8, linux, MAC OS. Use Arduino UNO bootloader

2. DC Jack Power Supply

Onboard 500mA resettable fuse to protect power supply from the USB port and the DC jack at the same time.

3. 4-Digit Display with Clock Point

A 4-digit tube (0.36 inches) that can display the clock point, it needs D10/D11 pins to control and display the integer, clock, stopwatch, score, etc.

4. Buzzer Module

Inbuilt piezoelectric buzzer

5. RTC Clock Module

Based on DS1307 high-precision real-time clock module, I2C interface, the address is 0x68.

6. LM75 Temperature Sensor

I2C interface temperature sensor, not only can measure the temperature, it can also set the temperature protection temperature, the address is 0x48.

7. Infrared Receiver

Use D2 pin. It can receive the modulated infrared signal of 38KHz that is sent by the IR transmitter module and demodulate it into logic level, and it can complete the infrared remote control with the codec program.

8. Reset Button

Reset switch that allows you to restart the programme without unplugging and plugging back in again.

9. 4-Channel Touch Sensor

Capacitive touch switch, only when you touch the corresponding position with your fingers, will the module corresponding pin (D3 / D4 / D5 / D6) output high level, otherwise it outputs low.

10. Signal ON/OFF

Onboard DIP switch, you can disconnect the connection between the peripheral module on the board and the Atmega328P.

11. ATmega328P Chip

Atmel ATmega328P microcontroller, working voltage: 5V, working current: 500mA (Max), IO logic voltage: 5V, 100% compatible with Arduino UNO R3 program, expansion shields, IDE.

12. KNOB Sensor

Rotation angle sensor. A 10K ohm adjustable potentiometer knob angle sensor, use A0 pin, can be used for MP3 volume adjustment, 4-digit display brightness adjustment.

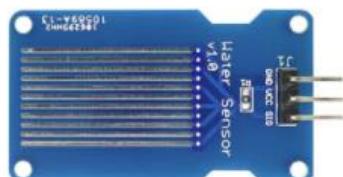
13. Shield Interface

Onboard Arduino Shield interface, can plug in compatible expansion shields.

14. MP3 Player

Serial MP3 music player module is based on high-quality MP3 music chip, use D7 / D8 pins to be software serial port, you can send commands to switch songs, change the volume and play mode and other operations. The board kit contains TF card, speaker, CR1220 battery, infrared remote control, and these are the necessary accessories for MP3, DS1307, infrared receiver.

Open Smart Kit (B) Specifications



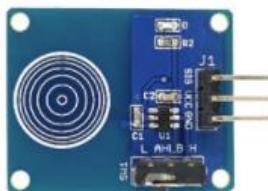
Water Sensor



NTC Sensor Line+Adapter module



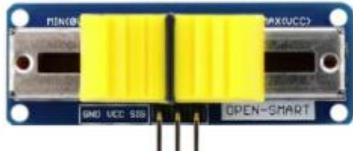
Ultrasonic Sensor



Touch Sensor (T)



Voltage Sensor



Slide Potentiometer



Rocker Switch



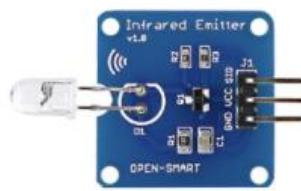
Vibration Motor



PIR motion Sensor



Light Sensor



Infrared Emitter

Commonly used Sensor modules

Water sensor

Easy to use moisture sensor module. Under normal circumstances, if the module does not touch the water line, water droplets or conductive objects, the signal pin of the module is held high by a 1M ohm pull-up resistor. When the module is exposed to water or rain droplets then that becomes low. Specifications: Operating voltage: DC3.3-5.5V. Operating current: less than 20mA. Dimension: 4.1 x 2.2cm.

NTC Thermistor sensor line + adapter module

Can be used to measure temperature directly in water. Widely used in temperature monitoring for indoor, outdoor, greenhouse, etc. You can plug the sensor onto the adapter module directly. Specifications: Working voltage: 2.2~12V DC. Working current: 0.5mA (max). Measuring range: -30~120°C. Accuracy: $\pm 2\%$ (4°C~50°C), $\pm 3\%$ (-15°C~80°C). B value: 3950K. R25: 10Kohm. Sampling resistor parameters: 10k ohm, 0.1%, 10ppm. Sensor

connector: XH2.54-2P. Lead length: 1m. Adapter output connector: standard electronic brick interface, 2.54mm-3P header.

Ultrasonic sensor

Distance measuring module for non-contact measurement. It has high measurement accuracy and its blind area is close (up to 2cm). Specifications: Arduino library ready: HCSR04 Ultrasonic. Working voltage: 3-5.5VDC. Static current: less than 2 mA. IO logic voltage: 3.3V / 5V. Induction angle: not more than 15 degrees. Detection range: 2-400 cm. Accuracy: 3mm. Adopt IO trigger by supplying at least a 10us sequence of high level signal. The module automatically sends an eight 40khz square wave and automatically detects whether to receive the returning pulse signal.

Touch sensor

This module is based on a touch detection IC (TTP223-BA6), and the touch IC is in trigger mode (Toggle Mode). On power-up, the module sends a low or high output level set by the board toggle switch. When in low or high power mode, the touch of a finger at the corresponding position will cause the module to output the opposite level (i.e. if the original output is high, it will output low after touching; if the original output is low, it will output high after touching). When in fast mode, the sensor will switch to low power mode after not being touched for 12 seconds. The module is covered by a thin paper surface (non-metallic) and can be installed into materials such as plastic, glass and non-metallic materials. As long as the touch button is in the correct location, you can hide the sensor in the walls, desktops etc. This module allows you to dispense with conventional mechanical push buttons and switches. Control Interface: A total of three pins (GND, VCC, SIG), GND to ground, VCC is the power supply, SIG is the digital signal output pin. Power light: a green LED that lights up if power is on. Touch area: A ring of concentric circles that looks similar to a fingerprint icon, this is the touch-sensitive region. Selector switch: when the switch is set to L the sensor is in low power mode; when the switch is set to H the sensor is in high power mode; when the switch is set to L the sensor is in low power mode; when the switch is set to AHBL the sensor is in fast mode. Specifications: Low power consumption. Power supply for 2 ~ 5.5V. Positive and negative can be used as a touch surface, DC can be an alternative to traditional self-locking switch.

Voltage sensor

The voltage detection module is based on the principle of a resistor divider. The measured voltage can be reduced, so that the ADC pin of the Arduino board can detect the voltage value after reduction, and the measured voltage can be calculated. The sampling resistor is a precision resistor, with precision of 0.5% and a temperature coefficient of 50PPM, in order to effectively ensure the detection accuracy. Specifications: Measurement Accuracy<= 1%. Measurement range up to 25 VDC.

Slide potentiometer

Module with 3P-2.54MM interface incorporates a linear variable resistor with a maximum resistance of 10K ohm. When you move the slider from one side to the other, its output voltage will range from 0 V to the VCC you apply. Especially suitable for volume control, lighting regulator and other DIY projects. Specifications: Working voltage: 3~12V DC. Working Current: 0.24mA. Slide stroke: 30mm.

Rocker switch

Module is based on a 2 feet 2 files rocker switch control module. The switch outputs at high level when you press the ON side, and outputs at low level when pressing the OFF side. Suitable for 3.3V and 5V.

Vibration motor

This is a mini vibration motor suitable as a non-audible indicator. When the input is high, the motor will vibrate just like your cell phone on silent mode. Specifications: Rated Voltage: 5.0VDC. Working Voltage: 3.0 - 5.3VDC. Rated Rotate Speed: Min. 9000RPM. Rated Current: Max. 60mA. Starting Current: Max. 90mA. Starting Voltage: DC 3.7V. Insulation Resistance: 10Mohm.

PIR motion sensor

HC-SR501 human infrared sensor module, based on infrared technology. High sensitivity, high reliability, low power consumption, ultra-low voltage operation mode. Widely used in various auto-sensing electrical equipment, especially battery-powered automatic control products. Infrared sensor with control circuit board, the sensitivity and holding time can be adjusted. Specifications: Working Voltage Range: DC 4.5V- 20V. Current drain:<60uA. Voltage Output: High/Low level signal:3.3V TTL output. Detection distance: 3--7M(can be adjusted). Detection range: <140°. Delay time: 5-200S (can be adjusted, default 5s +-3%). Blockade time: 2.5 S (default). Trigger: L: Non-repeatable trigger H: Repeat Trigger (default). Working temperature:-20-+80°C.

Light sensor

The light sensor module uses the GL5528 photoresistor to detect the light intensity of the environment. The resistance of the sensor decreases when the light intensity of the environment increases. The chip LM358 is used as a voltage follower to enable you to get accurate data. Specifications: low power consumption.

Infrared emitter

Module is based on the infrared emission control module 940. The infrared signal emitted by the transmitter tube is at 940nm wavelength. The signal can be acquired by an infrared receiver module a few meters away, and demodulation, encoding and decoding process can be achieved with the remote control function. Specifications: Level control interface, power supply for 5V or 3.3V



Dupont Line 1*40P



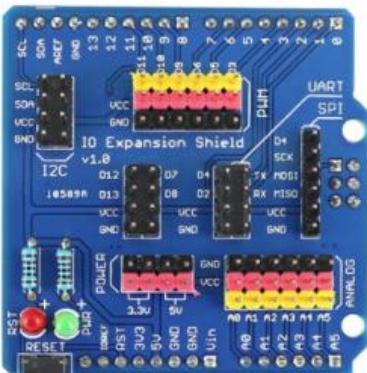
MicroSD Card Adapter



4-Digit Display



Passive Buzzer



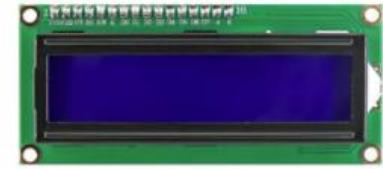
IO Expansion Shield



LED Bar module



Eagle Eye LED Module



I2C 1602 LCD module

IO Expansion Shield+Displays+Adapter

Dupont line 1*40P

Female-to-female hook-up wires.

MicroSD card adapter

The module is a Micro SD card reader module. It uses the SPI interface via the file system driver and microcontroller system to read and write files on a microSD card. The kit comes with a microSD card and the Arduino IDE can be used directly to complete the library card initialization and read-write. Supports microSD card and microSDHC card (high-speed card). The level conversion circuit board that can interface is 5V or 3.3V. Communication interface is a standard SPI interface. Control Interface: A total of six pins (GND, VCC, MISO, MOSI, SCK, CS), GND to ground, VCC is the power supply, MISO, MOSI, SCK is the SPI bus, CS is the chip select/slave select signal pin. 3.3V regulator circuit: LDO regulator output 3.3V as level converter chip. Level conversion circuit: microSD card into the direction of signals into 3.3V, microSD card toward the direction of the control interface MISO signal is also converted to 3.3V, general AVR microcontroller system can read the signal. Specifications:

4.5V - 5.5V, 3.3V voltage regulator circuit board. Positioning holes: 4 M2 screws positioning hole diameter of 2.2mm. Size: 45 x 28mm. Net weight: 6g.

4 digit display

The module is based on a decimal point display module, it displays four digital tubes (0.36 inches). The driver IC is TM1637, two signal lines can be used to make MCU control the four-digit 8-segment LED. Can be used to display a clock, decimals, letters etc.

Specifications: 4-digit red alpha-numeric display with decimal point; Working voltage: 3.3~5.5V DC; Working current: 80mA (MAX).

Passive buzzer

This is 5V passive buzzer module, and it is a piezoelectric speaker. The user can set the PWM output frequency and duration to produce different tones and beats according to the song numbered musical notation. Specifications: Operating voltage: 3.3 - 5.2VDC. Operating Current: 25mA (max).

IO expansion shield

The sensor shield (IO expansion board) is compatible with Arduino UNO, Rich UNO R3, Leonardo and Mega2560. It is a connection bridge between the electronic building block modules and the Arduino board. It extends the SPI interface, UART interface, I2C interface, PWM interface and analog interface of the Arduino board, so that DIY enthusiasts and Arduino interactive designers can quickly attach modules to the Arduino board and accelerate project development. On-board reset circuit, power and reset indicator.

LED bar module

This is an LED display module with 8 LEDs on-board. Low level signals will light the corresponding LED. Especially suitable for MCU IO tests and experiments and multi-channel wireless control indicators. Specifications: Operating voltage: 3 - 5.5VDC. Operating Current: 24mA (max). Active level: High level. Number of LEDs: 8. Display colours: yellow-green (D0 / D1), blue (D2 / D3), yellow (D4 / D5), red (D6 / D7).

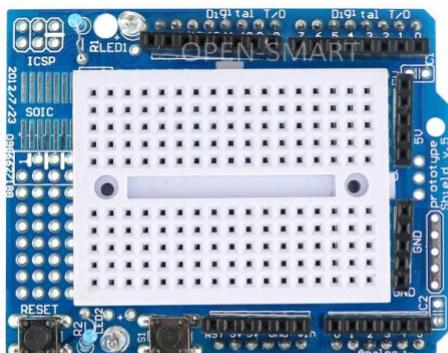
Eagle eye LED module

The module is a high efficiency green piranha LED module. Give a high signal to light up the Piranha LED. Specifications: Power supply 3.3 ~ 5.5V DC. High drive currents needed.

I2C 1602 LCD module

The module is a 2 line ,16 character LCD module with I2C interface. It is a basic character LCD screen for lots of applications. It is compatible with Arduino UNO R3 and Mega2560 and it can be used to display real time clock, temperature and humidity or any other text. Specifications: I2C Address: 0x27. Resolution: 80 * 16. Display Size: 2.6 inch. Power: 4.5~5.5V. Current: 80mA. Interface level: 5V.

Additional Components



Prototyping Shield



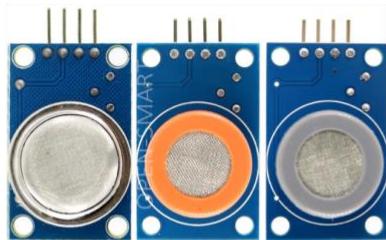
Shield with I2C
16x2 LCD Display



I2C OLED
64x128 Display



RGB LED Ring



Gas Sensors



LED Traffic Light

Additional Components

Prototyping shield

Used in conjunction with the Arduino board, the ProtoShield prototype expansion board can be used to assemble custom circuits for prototyping. It allows direct mounting of soldered components on the board, or connection through a mini breadboard. The breadboard is mounted via a double-sided adhesive pad. ProtoShield prototype expansion board includes two buttons and two LED circuits can be used directly. All of the pins and the power leads are Arduino-compatible.

Shield with I2C 16x2 LCD Display

LCD screen that can display two lines and 16 characters of text. This module also included two touch-sensitive keys, and several additional pins to input extra components. It is compatible with Arduino UNO R3 and Mega2560 and it can be used to display a real time clock, temperature and humidity or other text. Its backlight can help you to read the screen clearly in the dark environment. There are also two touch keys onboard, and an I2C interface for RTC module, a digital interface, four analog input pins that can be connected to the temperature measurement module (DS18B20), or temperature and humidity module (DHT11 or DHT22), etc. Specifications: I2C Address: 0x38. Operating voltage: 4.5~5.2V. Interface logic level: 5V. Operating current : 80mA (MAX). Backlight color: White. Char color: White. Display Size: 2.6 inch. Applications: Thermometers, Meters, DIY projects, etc.

I2C OLED 64x128 Display

This is an OLED monochrome 128x64 dot matrix display module with I2C Interface. Compared to LCD, OLED screens are way more competitive, which has a number of advantages such as high brightness, self-emission, high contrast ratio, wide viewing angle, wide temperature range, and low power consumption. Specifications: Interface: I2C interface. Resolution: 128*64. Angle of view: >160 degree. Display color: White. Display dimension: 1.3inch. Driver IC: SH1106. Working voltage: 3.3V~5VDC. Working current: 25mA (Max). Applications: smart watch, MP3, thermometer, instruments, DIY projects, etc..

RGB LED Ring

16-bit rainbow LED based on ws2811 RGB LED. It is compatible with Arduino products - just write a simple program, and it can produce different colors. Specifications: cascadable. Standard electronic building blocks interface, a single IO to control. Chipset: WS2811 (built-in LED). LED type: 5050 full color highlight RGB LED. Operating Voltage: 4.5~5.5V.

Gas Sensors

This is a commonly used gas sensor module kit that contains the most popular smoke sensor MQ-2, an alcohol sensor MQ3 and a carbon monoxide sensor.

MQ-2 Smoke Sensor

MQ-2 specifications: Type: Smoke Sensor. Detection Target: Liquefied petroleum gas, natural gas, coal gas, smoke. Effect: it can be used to detect smoke or combustible gas concentration, and in applied alarms. Operating voltage: 5V. Analog output (AO): the higher the concentration of combustible gas, the higher the output voltage. Digital output (DO): Adjustable concentration alarms by potentiometer. When the concentration exceeds the threshold you set, it outputs low level, otherwise high level output, and it can be connected to microcontroller or relay modules.

MQ-3 Alcohol Sensor

MQ-3 specifications: Type: Alcohol Sensor. Detection Target: alcohol. Effect: it can be used to detect the concentration of ethanol vapor, used in police alcohol measuring devices. Operating voltage: 5V. Digital output (DO): Adjustable concentration alarms by potentiometer, when the concentration exceeds the threshold you set, it outputs low level, otherwise high level output, and it can be connected to microcontroller or relay modules.

MQ-7 Carbon Monoxide Sensor

MQ-7 specifications: Type: Carbon Monoxide Sensor. Detection Target: carbon monoxide or other gases containing carbon monoxide. Function: used to detect the concentration of carbon monoxide, it can be applied to coal gas poisoning alarm devices. Operating voltage: 5V. Analog output AO: the higher the concentration of carbon monoxide, the higher the output voltage. Digital output DO: Adjustable concentration alarms by potentiometer, when the concentration exceeds the threshold you set, it outputs low level, otherwise high level output, and it can be connected to microcontroller or relay modules.

LED Traffic Light

This is a mini-traffic light display module, high brightness, very suitable for the production of a traffic light system model. Specifications: Number of lights: 3. Light color: red, yellow and green. Active level: high level. Operating voltage: 5V / 3.3V. Operating current: When you

use 5V high level output, the red light 13mA, yellow light 13mA, green light 25mA. Can be connected to the motherboard's PWM pin to control the brightness of the light.

Rich UNO R3 documentation

Downloads

Arduino Rich UNO R3 board manual: <https://www.biomaker.org/s/Rich-UNO-R3-User-Manual.pdf>

Arduino Rich UNO R3 board circuit diagram: <https://www.biomaker.org/s/OPEN-SMART-Rich-UNO-R3-Schematic.pdf>

Open Smart serial MP3 player manual:

<https://www.biomaker.org/s/Serial-MP3-Player-A-v10-Manual.pdf>

Websites

Arduino.cc: <https://forum.arduino.cc/index.php?topic=541297.0>

A guide to setting up the Rich UNO R3

Arduino Learning.com: <http://arduinolearning.com/hardware/look-open-smart-rich-uno-r3-atmega328p-development-kit.php>

A review of the Rich UNO R3 board

Video Tutorials

A series of video tutorials that demonstrate the use of the Open Smart Rich UNO R3 board. These tutorials use the Arduino IDE, rather than the XOD IDE, but general principles may apply.

Setup the board: (<https://www.youtube.com/watch?v=ius67kv00c4>)

Lesson 1: Touch and sound (<https://www.youtube.com/watch?v=MI7WIYSZMyc>)

Lesson 2: Number display (<https://www.youtube.com/watch?v=UMAgzA1H3Z4>)

Lesson 3: Touch and display (<https://www.youtube.com/watch?v=Ec4IL5b8wjc>)

Lesson 4: Touch and debounce (<https://youtu.be/-jydAbAgL4M>)

Lesson 5: Temperature measurement and display (<https://youtu.be/OnK0XlkGV-c>)

Lesson 6: Flashing number display (<https://youtu.be/NlpI4E6cYJE>)

Lesson 7: Knob angle (<https://youtu.be/pNXoI90HKAs>)

Lesson 8: Knob to control brightness (<https://youtu.be/TYgp9VBcYlo>)

Lesson 9: Clock input (<https://youtu.be/oK9-NryMSs4>)

Lesson 10: Clock display (<https://youtu.be/NzyGNtRFGKw>)

Lesson 11: Test for infrared keyboard (https://youtu.be/kWK1_-tZvWE)

Lesson 12: Infrared remote control and display (<https://youtu.be/n66YLmcpraA>)

Lesson 13: Play an mp3 song (https://youtu.be/_soOUNMJWkc)

Lesson 14: Play a song with its file name (<https://youtu.be/lHfYw6L4Bco>)

Lesson 15: Knob control of volume (<https://youtu.be/wgivbWiNmbM>)

Lesson 16: Touch control of mp3 playback (<https://youtu.be/-26Grg6VkJY>)

Lesson 17: Infrared remote control of mp3 playback (<https://youtu.be/JQqMmYRKRbs>)

Lesson 18: Voice playback (<https://youtu.be/XPffrH33CvU>)

Lesson 19: Speak value of Pi (https://youtu.be/_ak7TiWqoMM)

Lesson 20: Speak temperature value (<https://youtu.be/q3LMzaG2ezQ>)

Lesson 21: Speak clock time (<https://youtu.be/gscubR2FS68>)

Lesson 22: Speak time and temperature (<https://youtu.be/q2lmlQFzoiA>)

Additional tutorials can be found at:

https://www.youtube.com/channel/UCM_mzBFIDyMUnItB88c6LWQ/videos

Hardware suppliers

All components were sourced from Open-Smart: <https://open-smart.aliexpress.com>

A list of components and their costs (correct as of 23/03/2020) can be found in the table below. We have noted where components are essential (E); used in some tutorials, but not essential (S); or generally useful but not required for tutorials in this course (U).

Item	Cost	Essential?	Link
Arduino Rich UNO R3 Kit (B) with MP3 and Sensors	\$ 25.93	E	https://www.aliexpress.com/item/32822090848.html
Prototyping shield	\$ 1.55	U	https://www.aliexpress.com/item/32419802380.html?spm=2114.12010615.8148356.3.64622f8cYIX97w
Shield with LCD Display	\$ 3.70	U	https://www.aliexpress.com/item/32349041767.html?spm=2114.12010615.8148356.2.540d401flklHBc
RGB LED Ring	\$ 2.43	S	https://www.aliexpress.com/item/32461822160.html?spm=2114.12010615.8148356.18.61221266in4Pdy
OLED Display	\$ 3.67	S	https://www.aliexpress.com/item/32657357817.html?spm=2114.12010615.8148356.7.77cd44ceJisJJ6
Gas Sensors	\$ 3.51	U	https://www.aliexpress.com/item/32649677074.html?spm=2114.12010615.8148356.3.1de349dbawoocu
LED Traffic Light	\$ 0.75	S	https://www.aliexpress.com/item/32820546930.html?spm=a2g0o.store_home.productList_1691794.pic_6
Total	\$ 41.54		Plus Shipping