03. 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. Plugging into these pins will allow you to send information to, or receive information from these components.

<table>
<thead>
<tr>
<th>Pin</th>
<th>Connected Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0</td>
<td>KNOB potentiometer</td>
</tr>
<tr>
<td>A4</td>
<td>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).</td>
</tr>
<tr>
<td>A5</td>
<td>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).</td>
</tr>
<tr>
<td>D2</td>
<td>Infrared receiver</td>
</tr>
<tr>
<td>D3</td>
<td>Touch sensor channel 1 (TCH1)</td>
</tr>
<tr>
<td>D4</td>
<td>Touch sensor channel 2 (TCH2)</td>
</tr>
<tr>
<td>D5</td>
<td>Touch sensor channel 3 (TCH3)</td>
</tr>
<tr>
<td>D6</td>
<td>Touch sensor channel 4 (TCH4)</td>
</tr>
<tr>
<td>D7</td>
<td>MP3 player</td>
</tr>
<tr>
<td>D8</td>
<td>MP3 player</td>
</tr>
<tr>
<td>D9</td>
<td>Buzzer</td>
</tr>
<tr>
<td>D10</td>
<td>4-digit display</td>
</tr>
<tr>
<td>D11</td>
<td>4-digit display</td>
</tr>
</tbody>
</table>

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 website. Several tutorials relevant for this chapter are listed below.

**Introduction to serial data transfer:** [https://learn.sparkfun.com/tutorials/serial-communication](https://learn.sparkfun.com/tutorials/serial-communication)

**Introduction to I2C:** [https://learn.sparkfun.com/tutorials/i2c/all](https://learn.sparkfun.com/tutorials/i2c/all)

**Introduction to SPI:** [https://learn.sparkfun.com/tutorials/serial-peripheral-interface-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](https://learn.sparkfun.com/tutorials/how-to-use-a-breadboard?_ga=2.267526030.1277465979.1584962875-1288494135.1584962875)

Next Chapter

In the next chapter we will introduce you to XOD, the no-code programming software we will use to program your board. [Go to next chapter >>](#)