# Arduino Leonardo

The Arduino Leonardo is a microcontroller board based on the ATmega32u4 MCU. It has 20 digital input/output pins (of which 7 can be used as PWM outputs and 12 as analog inputs), a 16 MHz crystal oscillator, a micro USB connection, a power jack, an ICSP header, and a reset button.

## Power

The Arduino Leonardo can be powered via the micro USB connection or with an external power supply. The power source is selected automatically.

The power pins are as follows:

* VIN: The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). Voltage can be supplied through this pin, or, if supplying voltage via the power jack, it can be accessed through this pin.
* 5V: The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
* 3V3: A 3.3 Volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
* GND: Ground pins.
* IOREF: The voltage at which the i/o pins of the board are operating (i.e. VCC for the board). This is 5V on the Leonardo.

## Memory The ATmega32u4 has 32 KB (with 4 KB used for the bootloader). It also has 2.5 KB of SRAM and 1 KB of EEPROM.

## Input and Output Each of the 20 digital i/o pins on the Leonardo can be used as an input or output, operating at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 k.

In addition, some pins have specialized functions:

* Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data using the ATmega32U4 hardware serial capability.
* TWI (I2C equivalent): 2 (SDA) and 3 (SCL). Support TWI communication using the Wire library.
* External Interrupts: 3 (interrupt 0), 2 (interrupt 1), 0 (interrupt 2), 1 (interrupt 3) and 7 (interrupt 4). These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
* PWM: 3, 5, 6, 9, 10, 11, and 13. Provide 8-bit PWM output with the [analogWrite()](https://www.arduino.cc/en/Reference/AnalogWrite) function.
* SPI: on the ICSP header. These pins support SPI communication using the [SPI library](https://www.arduino.cc/en/Reference/SPI).
* LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
* Analog Inputs: A0-A5, A6 - A11 (on digital pins 4, 6, 8, 9, 10, and 12). The Leonardo has 12 analog inputs, labeled A0 through A11, all of which can also be used as digital i/o. Each analog input provide 10 bits of resolution (i.e. 1024 different values). By default the analog inputs measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the [analogReference](https://www.arduino.cc/en/Reference/AnalogReference)() function.
* AREF: Reference voltage for the analog inputs. Used with [analogReference](https://www.arduino.cc/en/Reference/AnalogReference)().
* Reset: Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

## Communication The Leonardo has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega32U4 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). The 32U4 also allows for serial (CDC) communication over USB and appears as a virtual com port to software on the computer. The chip also acts as a full speed USB 2.0 device, using standard USB COM drivers. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB connection to the computer. A [SoftwareSerial library](http://www.arduino.cc/en/Reference/SoftwareSerial) allows for serial communication on any of the Leonardo's digital pins. The ATmega32U4 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus. The Leonardo appears as a generic keyboard and mouse, and can be programmed to control these input devices using the [Keyboard and Mouse](https://www.arduino.cc/en/Reference/MouseKeyboard) classes.

## Programming The Leonardo can be programmed with the Arduino software. The ATmega32U4 on the Arduino Leonardo comes preburned with a [bootloader](https://www.arduino.cc/en/Tutorial/Bootloader) that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the AVR109 protocol. You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header using [Arduino ISP](https://www.arduino.cc/en/Main/ArduinoISP) or similar.

## Automatic (Software) Reset and Bootloader Initiation

Rather than requiring a physical press of the reset button before an upload, the Leonardo is designed in a way that allows it to be reset by software running on a connected computer. The reset is triggered when the Leonardo's virtual (CDC) serial / COM port is opened at 1200 baud and then closed. When this happens, the processor will reset, breaking the USB connection to the computer (meaning that the virtual serial / COM port will disappear). After the processor resets, the bootloader starts, remaining active for about 8 seconds. The bootloader can also be initiated by pressing the reset button on the Leonardo. Note that when the board first powers up, it will jump straight to the user sketch, if present, rather than initiating the bootloader. 

## USB Overcurrent Protection

The Leonardo has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed. 

