

Estás aquí: [Tienda PJRC](#) ► [Teensy 32 bit](#) ► Teensy 4.1

## Tienda PJRC

- [Lista completa de productos](#)

- **Teensy 32 bit**

- [Restos](#)

- **Teensy 4.1**

- [Teensy 4.1 Pines](#)
    - [Teensy 4.0](#)
    - [Teensy 4.0 Pines](#)
    - [Pantalla táctil a color](#)
    - [Pines 14x1](#)
    - [Pines 24x1](#)
    - [Zócalo 14x1](#)
    - [Zócalo 24x1](#)
    - [Cable host USB](#)
    - [Adaptador de audio](#)
    - [Micrófono](#)
    - [Tutorial de audio](#)
    - [Piezas del tutorial de audio](#)
    - [Aislador de tierra](#)
    - [de audio Kit de audio PT8211](#)
    - [Kit Ethernet](#)
    - [Pines + Zócalo 3x2](#)
    - [Adaptador OctoWS2811](#)
    - [Chip MKL02 T4.x](#)
    - [Chip PSRAM](#)
    - [Tutorial Kit Potenciómetro](#)
    - [LED RGB, 25K](#)
    - [Pines 14x1-D](#)

- [Teensy 3.x \(legado\)](#)

- [Reproductor MP3](#)





- [Teensy de 8 bits Placa de desarrollo 8051](#)

- [Descontinuado](#)
    - [Opciones de pago](#)
    - [Política de privacidad](#)



# Placa de desarrollo Teensy® 4.1

SparkFun ya fabrica productos Teensy. Puedes comprarlos directamente en SparkFun.

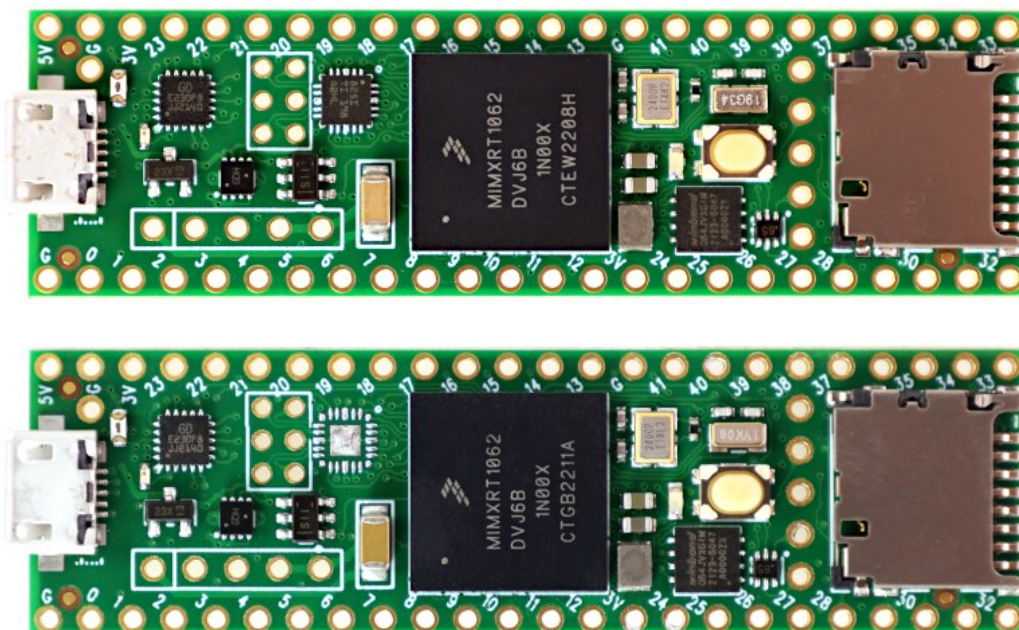
	TEENSY41	Placa USB Teensy, versión 4.1, con chip Ethernet ( <a href="#">el kit de conector</a> Ethernet se vende por separado) Para prototipos, experimentación y aprendizaje
	TEENSY41_NE	Placa USB Teensy, versión 4.1, sin chip Ethernet Para prototipos, experimentación y aprendizaje
	TEENSY41_LOCK	Placa USB Teensy con cerradura, versión 4.1, con chip Ethernet ( <a href="#">el kit de conector</a> Ethernet se vende por separado) Para productos comerciales y aplicaciones seguras, <a href="#">consulte Code Security para obtener detalles sobre Teensy con cerradura</a> .
	TEENSY41_NE_LOCK	Placa USB Teensy bloqueable, versión 4.1, sin chip Ethernet Para productos comerciales y aplicaciones seguras, <a href="#">consulte Code Security para obtener detalles sobre Teensy bloqueable</a> .

Accesorios recomendados: [Cable USB](#) , [pines 24x1](#) (2), [8 MB PSRAM](#) , [cable host USB](#) , [kit Ethernet](#)

## Secciones de esta página:

[Fotos](#) – [Especificaciones](#) – [Software](#) – [Procesador](#) – [Pines](#) – [Pines digitales](#) – [Pines analógicos](#) – [Comunicación](#) – [Pantallas](#) – [Audio](#) – [Luces y LED](#) – [Temporización](#) – [Alimentación](#) – [Memoria](#) – [Programación](#) – [Código de seguridad](#) – [Características especiales](#) – [Información técnica](#)

## Fotos



## Presupuesto

Característica	Teensy 4.1	Teensy 4.0
Ethernet	10/100 Mbit <a href="#">DP83825 PHY</a> (6 pines)	-ninguno-
Host USB	5 pines con gestión de energía	2 almohadillas SMT
SDIO (datos de 4 bits)	Ranura para micro SD	8 almohadillas SMT
Pines PWM	35	31
Entradas analógicas	18	14
Puertos serie	8	7
Memoria Flash	8 MB	2 Mbyte
Memoria QSPI	2 chips + memoria de programa	Memoria del programa
E/S de placa de pruebas	42	24
Almohadillas SMT inferiores	7	16
Señales de la tarjeta SD	6	0
Pines de E/S totales	55	40

### Diferencias entre Teensy 4.1 y Teensy 4.0

- ARM Cortex-M7 a 600 MHz
- Unidad matemática de punto flotante, 64 y 32 bits
- 7936K Flash, 1024K RAM (512K estrechamente acoplados), 4K EEPROM (emulado)
- Expansión de memoria QSPI, ubicaciones para 2 chips RAM o Flash adicionales
- Dispositivo USB 480 Mbit/seg y host USB 480 Mbit/seg
- 55 pines de entrada/salida digitales, 35 pines de salida PWM
- 18 pines de entrada analógica

- 8 puertos serie, 3 SPI, 3 I2C
- 2 puertos de audio digital I2S/TDM y 1 S/PDIF
- 3 buses CAN (1 con CAN FD)
- 1 puerto de tarjeta SD nativo SDIO (4 bits)
- Ethernet 10/100 Mbit con [DP83825 PHY](#)
- 32 canales DMA de propósito general
- Aceleración criptográfica y generador de números aleatorios
- RTC para fecha/hora
- FlexIO programable
- Canal de procesamiento de píxeles
- Disparo cruzado periférico
- Gestión de encendido y apagado

[Compare las especificaciones detalladas de todos los modelos Teensy .](#)

## Software

### • IDE de Arduino + Teensyduino

[El software IDE de Arduino con el complemento Teensyduino](#) es el entorno de programación principal para Teensy. En Windows, Linux y Macs antiguos, primero se instala Arduino y, a continuación, el instalador de Teensyduino añade la compatibilidad con Teensy al IDE de Arduino. En Macs más recientes, se proporciona una descarga completa. Teensyduino incluye una amplia colección de bibliotecas probadas y optimizadas para Teensy. Otras bibliotecas se pueden instalar manualmente o mediante el administrador de bibliotecas de Arduino.

### • Visual Micro

[Visual Micro](#) permite usar Microsoft Visual Studio para programar placas compatibles con Arduino, incluyendo Teensy. Solo es compatible con Windows. Visual Micro es un software comercial de pago.

### • Plataforma IO

[PlatformIO IDE](#) es un entorno de desarrollo multiplataforma con numerosas funciones avanzadas. Compatible con Windows, Linux y Macintosh.

### • CircuitPython

[CircuitPython](#) proporciona un archivo .HEX que se programa en Teensy 4.1 con [Teensy Loader](#) . Teensy se muestra en el ordenador como un disco USB, donde se copia o guarda el código Python. CircuitPython no es totalmente compatible con todo el hardware de Teensy 4.1.

### • Línea de comandos con Makefile

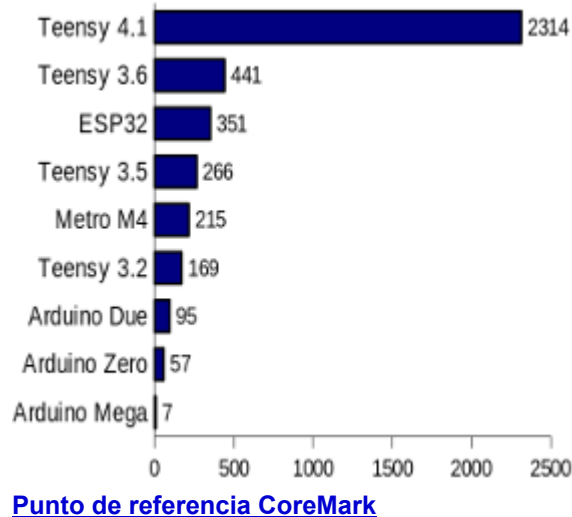
Los archivos Makefiles para uso no gráfico se proporcionan con el instalador de Teensyduino.

- **Teensy 4.x:**  
{Arduino}/hardware/teensy/avr/cores/teensy4/Makefile
- **Teensy LC y 3.x:**  
{Arduino}/hardware/teensy/avr/cores/teensy3/Makefile

# Procesador

## • Actuación

ARM Cortex-M7 incorpora numerosas y potentes funciones de CPU a una auténtica plataforma de



microcontroladores en tiempo real. El rendimiento de la CPU es mucho más rápido que el de los microcontroladores típicos de 32 bits.

## • Arquitectura de superescalador de doble emisión

Cortex-M7 es un procesador superescalador de doble emisión, lo que significa que el M7 puede ejecutar dos instrucciones por ciclo de reloj a 600 MHz. Por supuesto, la ejecución simultánea de dos instrucciones depende de la ordenación de las instrucciones y los registros por parte del compilador. Las primeras pruebas de rendimiento han demostrado que el código C++ compilado por Arduino tiende a ejecutar dos instrucciones entre el 40 % y el 50 % del tiempo al realizar tareas numéricas intensivas con enteros y punteros.

## • Unidad de punto flotante

La FPU realiza cálculos matemáticos de precisión doble de 64 bits y de punto flotante de 32 bits en hardware. La velocidad del punto flotante de 32 bits es aproximadamente la misma que la del cálculo de números enteros. La precisión doble de 64 bits se ejecuta a la mitad de la velocidad del punto flotante de 32 bits.

## • Memoria estrechamente acoplada

La memoria estrechamente acoplada (MTM) es una característica especial que permite a Cortex-M7 acceder rápidamente a la memoria en un solo ciclo mediante un par de buses de 64 bits. El bus ITCM proporciona una ruta de 64 bits para obtener instrucciones. El bus DTCM consiste en un par de rutas de 32 bits, lo que permite a M7 realizar hasta dos accesos a memoria independientes en el mismo ciclo. Estos buses de alta velocidad son independientes del bus AXI principal de M7, que accede a la memoria y a otros periféricos.

## • Cache

Se utilizan dos cachés de 32 K, uno para instrucciones y otro para datos, para acelerar el acceso repetitivo a la memoria que

no es TCM.

- **Predicción de rama**

Cortex-M7 es el primer microcontrolador ARM que utiliza predicción de bifurcación. En Cortex-M4 y versiones anteriores, los bucles y otro código con mucha bifurcación requieren tres ciclos de reloj. Con M7, tras la ejecución de un bucle varias veces, la predicción de bifurcación elimina esta sobrecarga, permitiendo que la instrucción de bifurcación se ejecute en un solo ciclo de reloj.

- **Procesamiento de señales digitales**

Las instrucciones de extensión DSP aceleran el procesamiento de señales, los filtros y la transformada de Fourier. La [biblioteca de audio](#) utiliza automáticamente estas instrucciones DSP.

## Patatas

Teensy 4.1 tiene un total de 55 pines de señal de entrada/salida. 42 son fácilmente accesibles cuando se utiliza con una placa de pruebas sin soldadura.

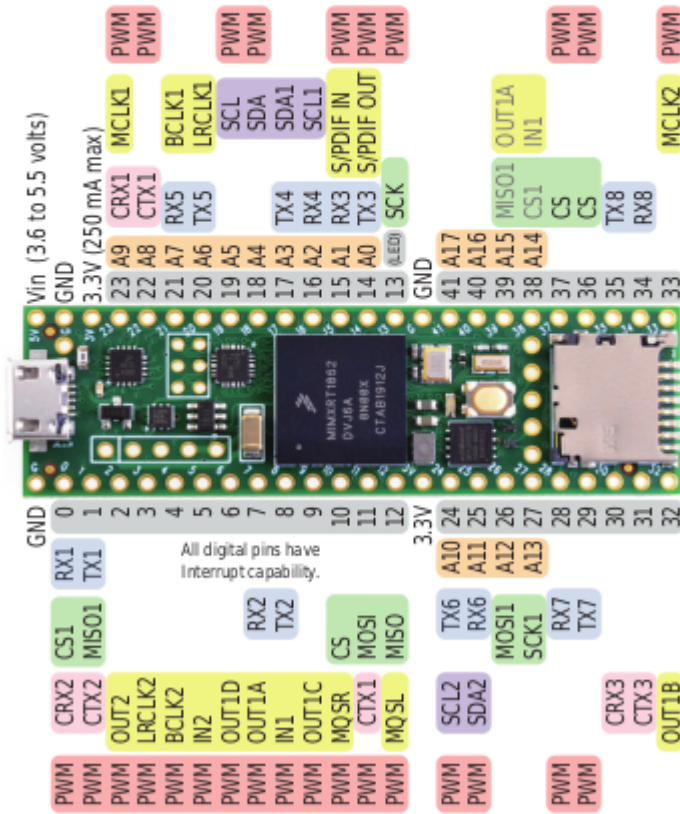
Esta tarjeta de referencia de distribución de pines viene con Teensy 4.1.

# Welcome to Teensy® 4.1

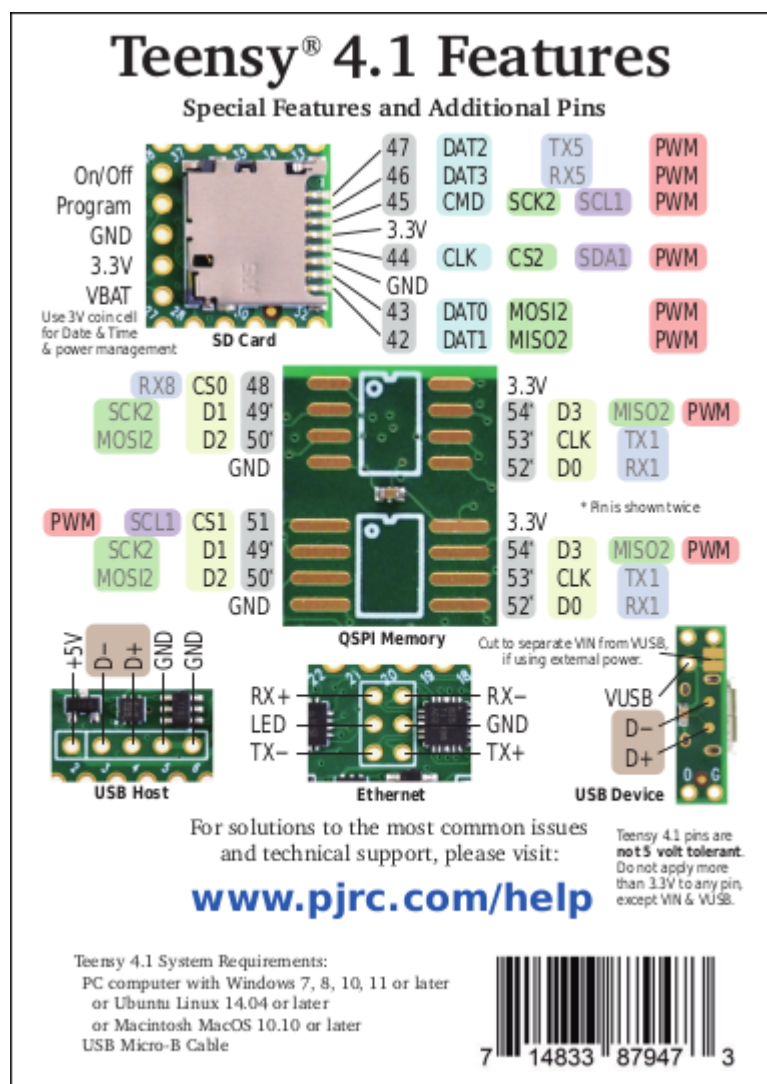
32 Bit Arduino-Compatible Microcontroller

To begin using Teensy, please visit the website & click [Getting Started](#).

[www.pjrc.com/teensy](http://www.pjrc.com/teensy)







Archivos de tarjetas de distribución de pines: [Anverso](#) (PDF) / [Reverso](#) (PDF)

Las tarjetas impresas antes de septiembre de 2021 mostraban incorrectamente el pin 53 con PWM.

También está disponible en el foro [un diagrama de pines más grande y detallado de KurtE.](#)

## Pines digitales

- Pines de entrada digital**

[Se pueden usar pines digitales](#) para recibir señales. Los pines del Teensy 4.1 están configurados por defecto como ENTRADA, la mayoría con una resistencia de retención. Los pines del Teensy 4.1 aceptan señales de 0 a 3,3 V. No toleran 5 V. No exceda ningún pin digital de 3,3 V.

- Resistencias de entrada pullup/pulldown/keeper**

Todos los pines digitales tienen resistencias pullup, pulldown o de mantenimiento opcionales. Estas se utilizan para mantener el pin en nivel lógico ALTO o BAJO, o en el mismo nivel lógico, cuando no está siendo controlado activamente por circuitos externos. Normalmente, estas resistencias se utilizan con pulsadores e interruptores. La función pinMode con INPUT\_PULLUP o INPUT\_PULLDOWN debe utilizarse para

configurar estos pines en modo de entrada con la resistencia integrada.

- **Interrupciones por cambio de pin**

All digital pins can detect changes. Use `attachInterrupt` to cause a function to be run automatically. Interrupts should only be used for clean signals. The [Bounce library](#) is recommended for detecting changes on pushbuttons, switches, and signals with noise or mechanical chatter.

- **Digital Output Pins**

All digital pins can act at output. The `pinMode` function with `OUTPUT` or `OUTPUT_OPEN_DRAIN` must be used to configure these pins to output mode. The `digitalWrite` and `digitalToggle` functions are used to control the pin while in output mode. Output HIGH is 3.3V. The recommended maximum output current is 4mA.

- **Pulse Width Modulation (PWM)**

35 of the digital pins support [Pulse Width Modulation \(PWM\)](#), which can be used to control motor speed, dim lights & LEDs, or other uses where rapid pulsing can control average power. PWM is controlled by the `analogWrite` function. 22 groups of PWM can have distinct frequencies, controlled by the [analogWriteFrequency](#) function.

- **Slew Rate Limiting**

This optional feature greatly reduces high frequency noise when long wires are connected to digital output pins. The rate of voltage change on the pin is slowed. The extra time is only nanoseconds, which is enough to lower undesirable high frequency effects which can cause trouble with long wires.

- **Variable Drive Strength**

The output impedance of each digital output may be controlled in 7 steps, ranging from 150 ohms (weakest) up to about 21 ohms (strongest).

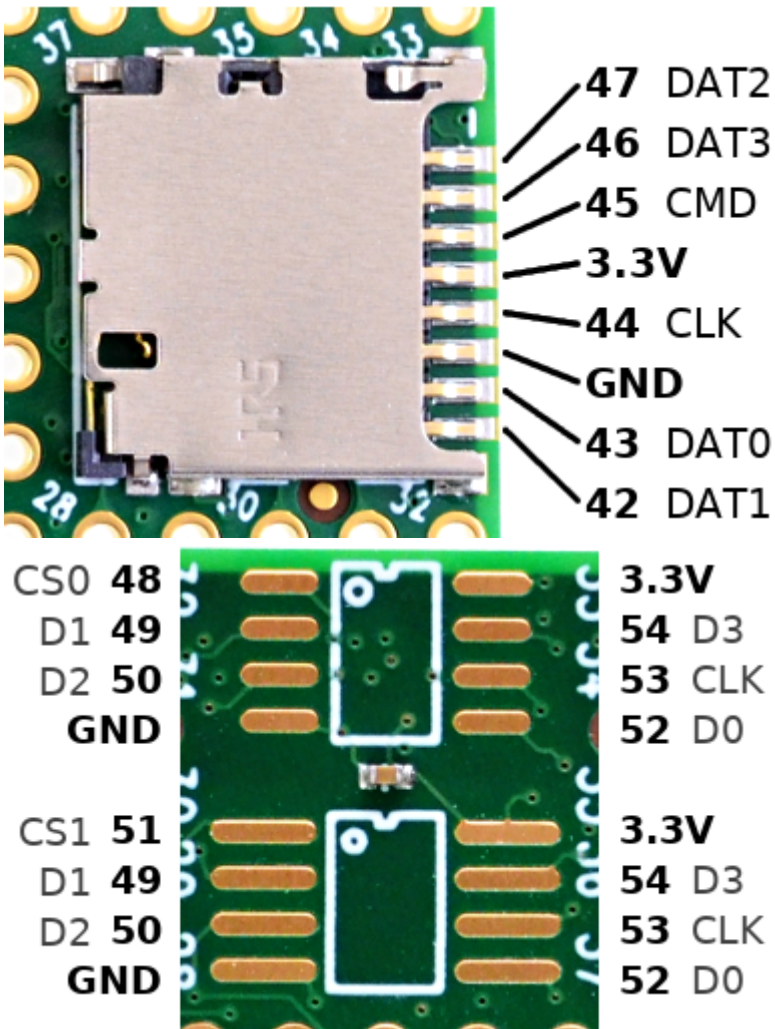
- **Adjustable Output Bandwidth**

Digital output bandwidth is also programmable, in 4 steps: 50, 100, 150 and 200 MHz.

- **LED Pin**

Pin 13 has an orange LED connected. The LED can be very convenient to show status info. When pin 13 is used as an input, the external signal must be able to drive the LED when logic HIGH. `pinMode INPUT_PULLUP` should not be used with pin 13.





## Analog Pins

- **Analog Inputs**

18 pins can be used as analog inputs, for reading sensors or other analog signals. Basic analog input is done with the `analogRead` function. The default resolution is 10 bits (input range 0 to 1023), but can be adjusted with `analogReadResolution`. The hardware allows up to 12 bits of resolution, but in practice only up to 10 bits are normally usable due to noise. More advanced use is possible with the [ADC library](#).

- **Analog Range**

The analog input range is fixed at 0 to 3.3V. On Teensy 4.1, the `analogReference()` function has no effect. The analog pins are not 5V tolerant. Do not drive any analog pin higher than 3.3 volts.

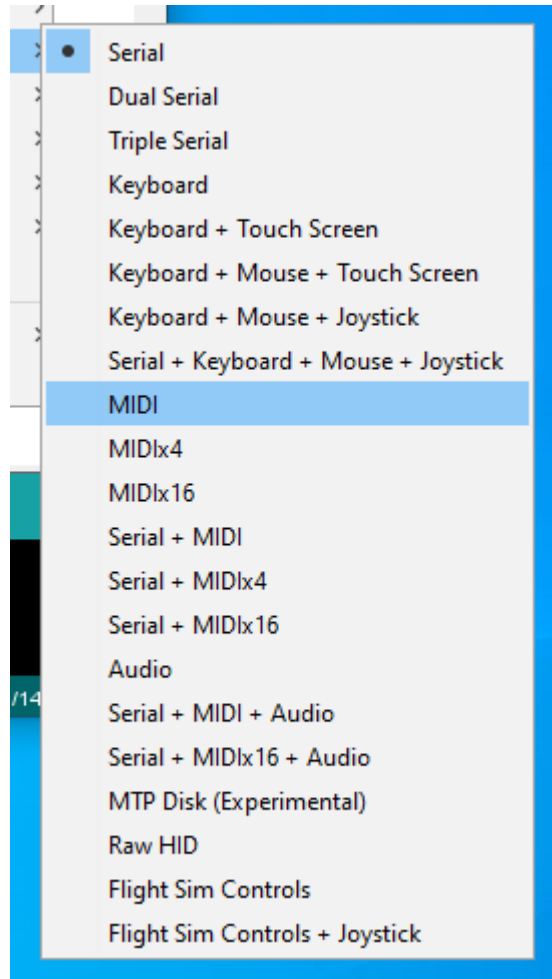
- **Analog Comparators**

These comparators allow an analog signal to be converted to digital, with a precisely defined voltage threshold for logic low versus high.

## Communication

## • USB Device

Teensy's  
primary



**Tools > USB Type menu configures the type of USB device Teensy will implement.**

communication is its main USB port, which operates in USB device / peripheral mode at 480 Mbit/sec speed. The Teensyduino software supports many different types of USB communication to your PC or Mac, selected by the Tools > USB Type menu. Several of these device types may be used simultaneously.

- **Serial** - Seen by your computer as a COM port (Windows) or serial device (Mac, Linux), Serial is the default and most commonly used communication type. Bytes are transferred in both directions at maximum USB speed (baud rate settings are ignored). Teensyduino has highly optimized code to allow fast USB serial data transfer. While normally used with the Arduino Serial Monitor, Teensy's USB Serial mode is compatible with software designed for serial ports, like CoolTerm. On Teensy, the serial device is accessed as "Serial". In the Dual & Triple Serial modes, the additional serial devices are "SerialUSB1" and "SerialUSB2".
- **Emulated Serial** - The USB Type settings lacking Serial use a HID interface to emulate serial. In these modes, your PC or Mac will not detect a COM port or serial device, but you can still use Serial.print() to send text to the Arduino Serial Monitor.
- **MIDI** - Musical Instrument Device. MIDI is often used to interface knobs, sliders and buttons to music & sound control software. MIDI messages may be sent in both directions. Teensyduino's MIDI is "class compliant" for compatibility with Macintosh, Linux, and Windows using only built-in drivers. The MIDIx4 & MIDIx16 modes provide 4 or 16 virtual MIDI ports / cables. The MIDI device name seen by your computer may be customized.

- **Audio** - Bi-directional stereo audio streaming, seen by your computer as a USB sound card. Using your computer's sound preferences, programs which play sound can stream to Teensy, and programs which record or process sound can receive, as if you were using a USB microphone. USB Audio is meant to be used together with the [Teensy Audio Library](#), allowing your computer's sound to integrate with any audio processing system you design on Teensy.
- **Keyboard** - Standard 104 key USB keyboard. Programs can transmit keystrokes to your computer, allowing control of nearly any software. Media control keys (play, pause, volume, etc) may also be used. Many non-US keyboard layouts are supported, using the Tools > Keyboard Layout menu.
- **Mouse** - A special USB mouse is emulated. Both relative motion of a normal mouse, and absolute screen position similar to a digitizer pen can be sent to your computer. Mouse buttons and scroll wheel are also supported.
- **Joystick** - A joystick / game controller with 6 axes (X, Y, Z, Zr, Slider1, Slider2), 32 buttons, and 1 hat switch are supported. The Joystick type is useful for controlling games or other software which responds to a joystick.
- **Touchscreen** - Emulates a touchscreen capable of detecting up to 10 finger positions.
- **MTP Disk** - Media Transfer, seen by your computer as a phone or camera which shares files.
- **Flight Sim** - Allows integration with the X-Plane flight simulator software. Variables and controls within the simulator are linked to variables in your code running on Teensy.
- **Raw HID** - Allows communicating 64 byte messages with custom written software on your computer.

## • USB Host

A second USB port operates in host mode, which allows you to connect USB devices to Teensy 4.1. It is fully independent of the main USB device port, so USB devices you connect on



USB Keyboard connected to USB Host port

the host port can simultaneously communicate with Teensy while Teensy communicates with your computer via the USB device port. This USB host port runs at 480, 12 or 1.5 Mbit/sec, depending on the speed of the device you connect. USB hubs may be used to connect many USB devices. The [USBHost\\_t36 library](#) is used for the USB host port. This USB host cable is normally used to connect a USB device or hub.

## • Serial

[8 serial ports](#) allow you to connect serial devices, such as MIDI, GPS receivers, DMX lighting, ESP wireless modules, etc. All 8

serial ports are fully independent and can transfer data simultaneously. None are shared with USB (as is done on some Arduino boards). All 8 ports include FIFOs for better performance at high speed baud rates.

## • I2C

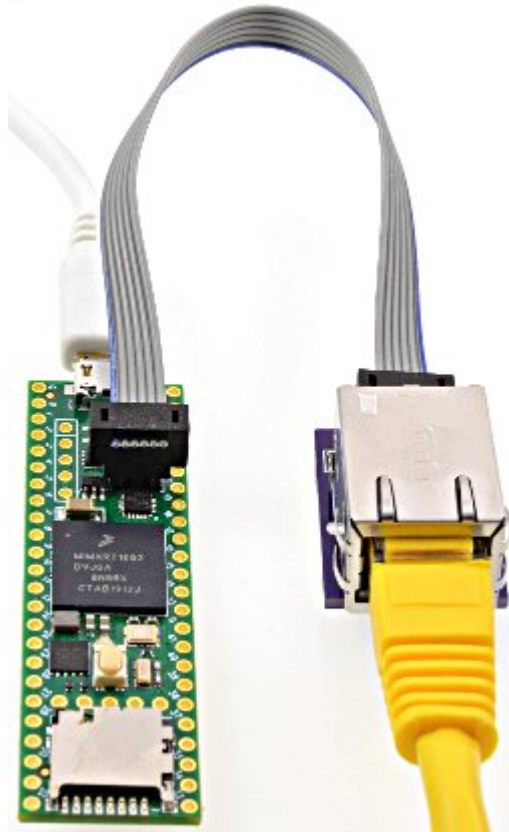
3 ports for I2C (signals SDA & SCL) allow connecting a wide variety of chips which use I2C communication. The [Wire library](#) is used for I2C. Each I2C chip connected to the same SDA/SCL wires needs a unique address. Multiple I2C ports allow you to easily use more than 1 chip with the same address. All I2C ports support 100, 400, and 1000 kbit/sec speeds.

## • SPI

3 ports for SPI (signals MOSI, MISO, SCK) allow connecting higher speed chips, SD cards, and displays which use SPI communication.

The [SPI library](#) provides software support for SPI.

The first SPI port features a FIFO for higher sustained speed transfers. Each SPI chip requires a chip select (CS) signal. Most libraries using SPI can use any digital pin. The SPI ports provide special hardware controlled CS pins, which are used by specially optimized libraries for higher performance.



[Ethernet Kit](#) on Teensy 4.1

## • CAN

3 ports for CAN bus allow connecting to automotive & industrial control systems which use CAN communication. A CAN transceiver chip must be added to complete the electrical interface between Teensy 4.1 and the CAN bus.

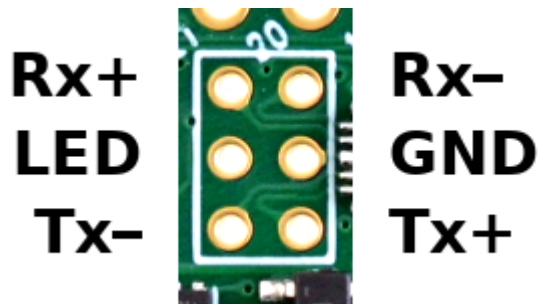
## • FlexIO

FlexIO is a highly configurable peripheral, with a sort of build-your-own ports from a collection of shift registers, timers, logic and state machines. FlexIO can implement UARTs (serial), I2C, SPI, I2S audio, PWM. Unique interfaces can also be built, such as the [TriantaduoWS2811 library](#).

## • Ethernet



Teensy 4.1 contains an Ethernet controller and Ethernet PHY chip. To connect an Ethernet cable, only this [RJ45 magjack kit](#) is needed. Ethernet can also be implemented using the [Wiznet W5500 and Wiz820 shield](#), connected to the SPI port.



## Displays



[ILI9341 Color TFT Display](#) The best supported display for Teensy 4.1

- **ILI9341 320x240 Color TFT**

These displays are the best supported on Teensy 4.1, with multiple high performance libraries for fast updates speed. [ILI9341](#) is usually the best display to use, due to superior software support.

- **ST7735 Color TFT**

These displays are slightly smaller and lower resolution than ILI9341. Highly optimized libraries for ST7735 & ST7789 allow these to also perform very well.

- **SSD1306 Monochrome OLED**

These small displays are very popular and well supported.

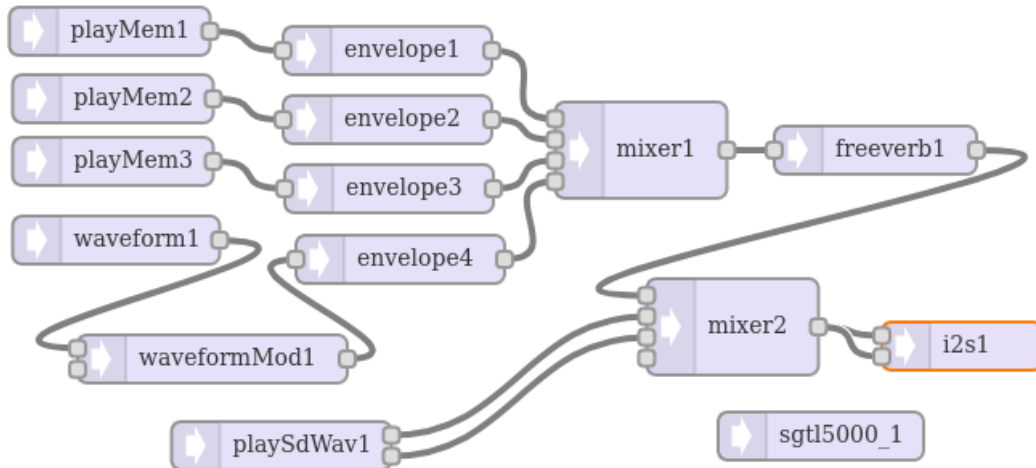
- **Other Displays**

Almost all displays with Arduino libraries work on Teensy 4.1.

- **Pixel Pipeline**

A special graphic engine can perform color space transformation, alpha blending and chroma keying, bilinear resize and other operations one frame buffers. Software support is [still very experimental](#).

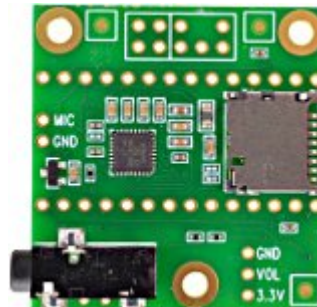
# Audio



[Audio Design Tool](#) makes it easy to create an audio processing system which streams sound while your program runs.

## • I2S / TDM

Most commonly used with the [audio shield](#), 2 digital audio ports can simultaneously transmit and receive up to 8 audio channels using I2S protocol, or up to 16 channels using [TDM](#). Alternately, a special format used by inexpensive [PT8211 DAC](#) chips can be used.



[Audio Shield](#) converts I2S digital audio to analog stereo input & output.

- **I2S1** - 1 stereo output pin, 1 stereo input pin, 3 stereo input or output pins
- **I2S2** - 1 stereo output pin, 1 stereo input pin

## • S/PDIF

The I2S port may be used to receive and transmit S/PDIF. The S/PDIF is independent of both I2S/TDM ports and can be used simultaneously. Either or both of the I2S ports may also be used to transmit S/PDIF.

## • Analog Input (ADC)

1 analog input pins may be used for audio inputs. Using an ADC pin for audio input currently has only "experimental" software support.

## • MQS Output

This pulsed digital output which blends PWM with oversampling & noise shaping can be used to drive small speakers. Or the output may be low-pass filtered to give analog signals. While called "Medium Quality Sound", the performance is surprisingly good.





**PT8211** is the least expensive DAC for good quality stereo signal output

## Lights & LEDs



**OctoWS2811 Library** controlling 1920 WS2812B RGB LEDs at 30 Hz refresh rate

- **WS2812B / NeoPixel**

Two high performance non-blocking libraries support use of WS2812B LEDs. [OctoWS2811](#) transmits any number of outputs in parallel, allowing almost any number LEDs to be refreshed at up to 30 Hz video rate. On Teensy 4.1, OctoWS2811 supports use of any number of digital pins, not limited to only 8 pins as on Teensy 3.x. [WS2812Serial](#) transmits a single output, but up to 8 instances may be used. Non-blocking transmission uses DMA to transmit automatically, while your code is able to continue running. This much more allows complex animations or efficient communication than traditional blocking.

- **SmartMatrix & SmartLED Shield for HUB75 RGB LED Panels**

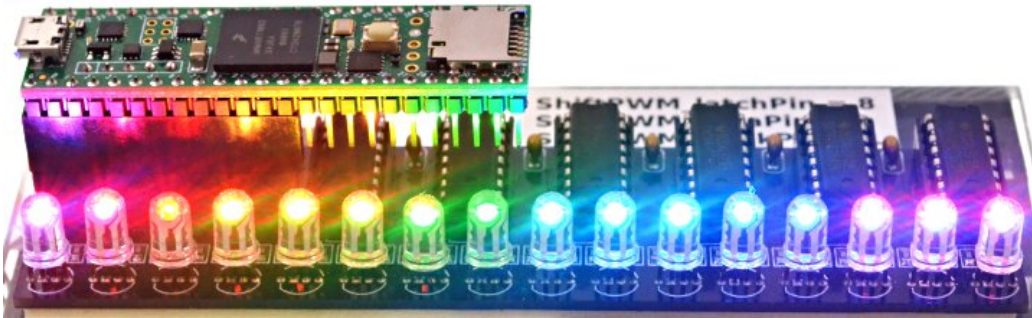
[SmartLED Shield](#) (version 5) enables Teensy 4.1 to drive high-quality graphics to large HUB75 RGB LED panel arrays (from 32x16 up to 128x64 pixels). [The SmartMatrix library](#) makes it easy to draw basic graphics, create scrolling and static text, draw beautiful patterns using FastLED, and play animated GIFs on the panel. SmartMatrix uses Teensy 4.1's special features to send graphics data with minimal CPU usage, so you can use the processor to do other tasks in parallel such as SPI communication, file decoding, or complex rendering.

- **DMX Lighting Control**

Any of the 8 serial ports may be used for efficient communication with [DMX lighting](#) controllers.

## • RGB LEDs

Ordinary LEDs may be variable-brightness controlled by PWM, or the [SoftPWM](#) & [ShiftPWM](#) libraries.

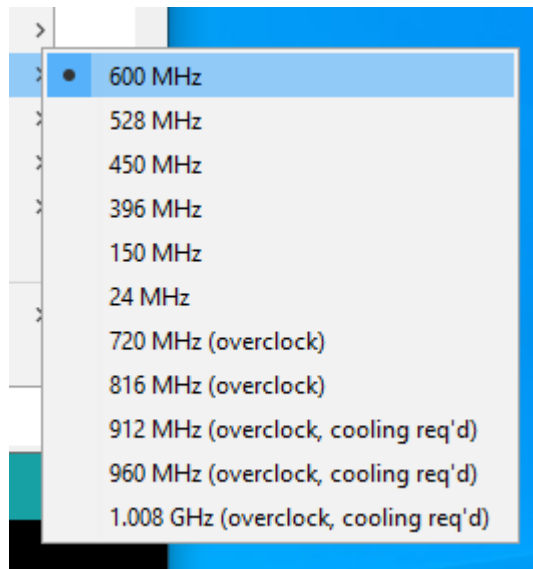


ShiftPWM controlling 16 RGB LEDs using six 74HCT595 chips

## Timing

### • Crystals & Clock Generation

Two crystals provide accurate timing. A 24 MHz crystal is the basis for the system clock and most peripherals. A phase locked loop (PLL) increases the 24 MHz up to the system clock speed. A separate 32.768 kHz crystal is used for the Real Time Clock (RTC). If a coin cell is added to VBAT, the 32.768 kHz oscillator continues keeping date/time while main power is off.



Tools > USB Speed menu configures the speed Teensy 4.1 will run your code.

### • Interval Timers

4 timers are dedicated to running a function at precisely timed intervals. These are configured using the [IntervalTimer class](#).

### • PWM Timers

32 timers [control PWM pins](#), or may be used for other timing functions. Normally these timers are accessed with analogWrite or libraries, but they have many very advanced features which may be accessed by direct hardware register use.

- **FlexPWM1 Module0** - Controls PWM pins 1, 44, 45.
- **FlexPWM1 Module1** - Controls PWM pins 0, 42, 43.
- **FlexPWM1 Module2** - Controls PWM pins 24, 46, 47.

- **FlexPWM1 Module3** - Controls PWM pins 7, 8, 25.
- **FlexPWM2 Module0** - Controls PWM pins 4, 33.
- **FlexPWM2 Module1** - Controls PWM pin 5.
- **FlexPWM2 Module2** - Controls PWM pins 6, 9.
- **FlexPWM2 Module3** - Controls PWM pins 36, 37.
- **FlexPWM3 Module0** - Controls PWM pin 53.
- **FlexPWM3 Module1** - Controls PWM pins 28, 29.
- **FlexPWM3 Module2** - No pins accessible.
- **FlexPWM3 Module3** - Controls PWM pin 41.
- **FlexPWM4 Module0** - Controls PWM pin 22.
- **FlexPWM4 Module1** - Controls PWM pin 23.
- **FlexPWM4 Module2** - Controls PWM pins 2, 3.
- **FlexPWM4 Module3** - No pins accessible.
- **QuadTimer1 Module0** - Controls PWM pin 10.
- **QuadTimer1 Module1** - Controls PWM pin 12.
- **QuadTimer1 Module2** - Controls PWM pin 11.
- **QuadTimer1 Module3** - No pins accessible.
- **QuadTimer2 Module0** - Controls PWM pin 13.
- **QuadTimer2 Module1** - No pins accessible.
- **QuadTimer2 Module2** - No pins accessible.
- **QuadTimer2 Module3** - No pins accessible.
- **QuadTimer3 Module0** - Controls PWM pin 19.
- **QuadTimer3 Module1** - Controls PWM pin 18.
- **QuadTimer3 Module2** - Controls PWM pin 14.
- **QuadTimer3 Module3** - Controls PWM pin 15.
- **QuadTimer4 Module0** - No pins accessible. Used by [OctoWS2811 library](#), ADC Library
- **QuadTimer4 Module1** - No pins accessible. Used by [OctoWS2811 library](#)
- **QuadTimer4 Module2** - No pins accessible. Used by [OctoWS2811 library](#)
- **QuadTimer4 Module3** - No pins accessible. Used by [Audio](#) for ADC timing, and ADC Library

## • Watchdog Timer

3 separate watchdog timers are meant to reboot Teensy if your software crashes or gets stuck. Once started, the watchdog timer must be periodically reset. If the software stops resetting the timer for too long, Teensy reboots.

## • Special Timers

These extra timers allow delays, analog sample rate timing, carrier modulation, and other special timing tasks to be performed, without consuming any of the normal PWM-oriented timers.

- **GPT1** - Generic 32 bit timer
- **GPT2** - Generic Generic 32 bit Timer
- **Quadrature Encoders** - 4 special timers are meant for decoding quadrature signals.

## • Cycle Counter

A 32 bit counter increments every CPU clock cycle (600 MHz). ARM\_DWT\_CYCCNT may be read by programs to precisely measure short time duration time.

- **SysTick**

This system timer generates an interrupt every millisecond. Most of the software timing features use this SysTick timer.

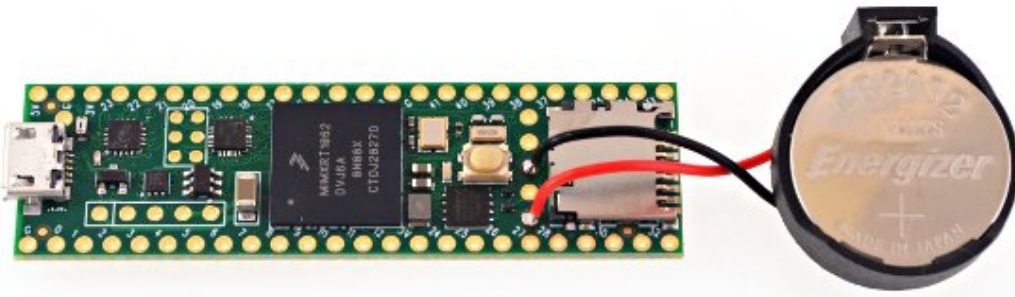
- **Software Timing**

Many common timing requirements can be met using the software timing features.

- [delay\(\)](#), [delayMicroseconds\(\)](#), [delayNanoseconds\(\)](#) - Simple delay for milliseconds, microseconds, or nanoseconds.
- [elapsedMillis](#), [elapsedMicros](#) - These C++ classes act as a variable which automatically increments every millisecond or microsecond. These can be written or modified as needed, which greatly simplifies implementation of repetitive tasks, measuring elapsed time, inactivity timeouts, and so on. The number of these variable is only limited to the available memory.
- [millis\(\)](#), [micros\(\)](#) - Standard Arduino functions for the system time in milliseconds and microseconds.

- **Real Time Clock - Date & Time**

The RTC keeps track of date / time. The [Time library](#) is typically used together with the RTC. Teensy Loader automatically initializes the RTC to your PC's time while uploading. If a coin cell is connected to VBAT, the RTC will continue keeping time while power is turned off.



CR2032 Coin Cell connected to VBAT allows Teensy 4.1 to keep date / time while power is off

## Power

- **USB Power**

Normally Teensy is powered by your PC or USB hub, through a USB cable. The USB power arrives at the VUSB pin, which is connected VIN and powers the entire board.

- **VIN Pin**

When USB power is not used, 5V power may be applied to the VIN pin. Because VIN & VUSB are connected, power should not be applied to VIN while a USB cable is used, to prevent the possibility of power flowing back into your computer. Alternately, a pair of pads on the bottom side may be cut apart, to separate VUSB from VIN, allowing power to be safely applied while USB is in use. (TODO: VUSB-VIN pads photo, right side)

- **3.3V Power**

Teensy 4.1 has a voltage regulator which reduces the 5V VUSB / VIN power to 3.3V for use by the main processor and most



other parts. Additional circuitry may be powered from the 3.3V pin. The recommended maximum for external 3.3V usage is 250mA. Teensy 4.1 is not meant to receive power on its 3.3V pin, but this can be done with [special modificaton](#).

- **USB Host Hot Plugging**

Power to USB devices connected on the USB host port is provided through a current limited switch and a large capacitor. The current limit lessens the disruption to Teensy's power when a USB device is hot plugged and needs a sudden inrush current to charge up all its capacitors.

- **Power Consumption**

When running at 600 MHz, Teensy 4.0 consumes approximately 100 mA current. Reducing CPU speed to 528 MHz or lower reduces power consumption.

- **Low Power Features**

[Snooze library](#) (TODO: more info here...)

- **CPU Voltage Control**

A DC-DC buck converter creates lower voltage needed for the CPU. Software can control this voltage, in 50 mV steps. At 600 MHz, the CPU runs on 1.25V. For 528 MHz and lower, 1.15V is used. At 24 MHz, the CPU runs on only 0.95 volts. When overclocking, higher voltages are automatically used.

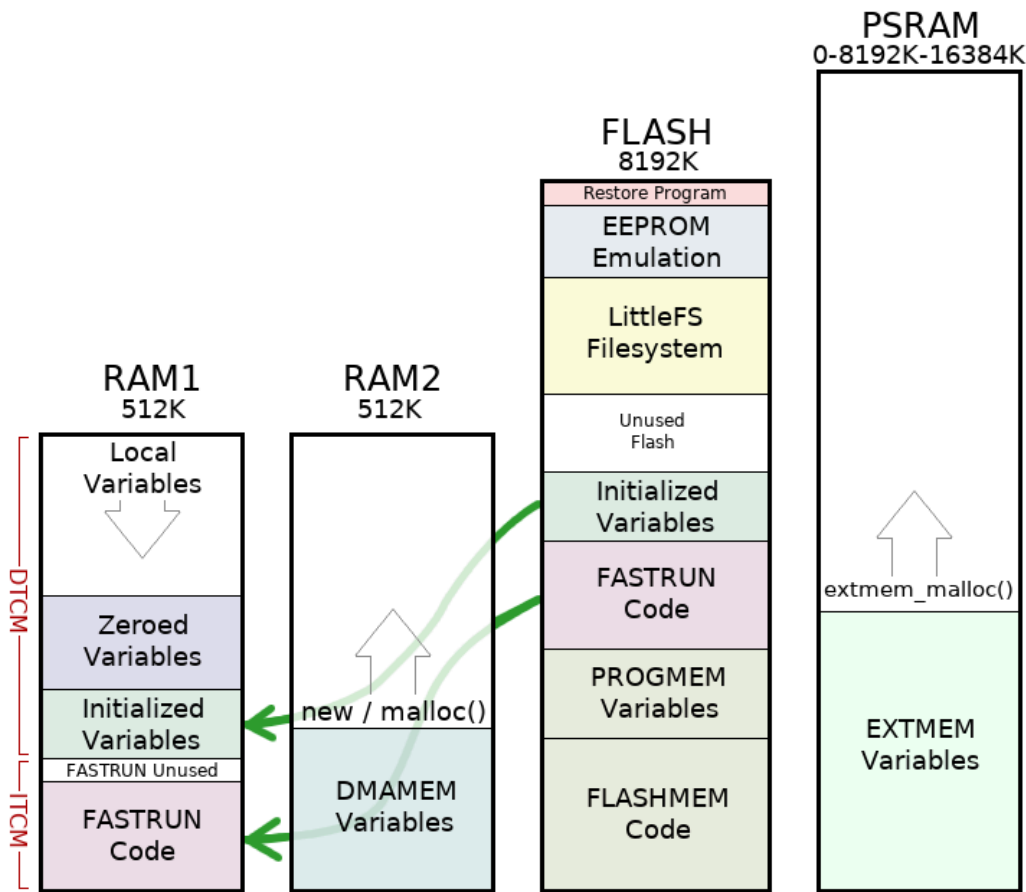
- **VBAT**

A 3 volt coin cell may be connected to VBAT & GND to allow the RTC to keep track of date / time while power is off. A CR2032 type battery is recommended, though other 3V coin cells may also be used.

- **On / Off Pin and Power Control**

A special low power state which turns off the 3.3V power can be controlled by the On/Off pin. A pushbutton is meant to be connected between On/Off and GND. While running, holding the button for 4 seconds turns off power. Pressing for 0.5 seconds while power is off turns the 3.3V power back on and reboots the processor. If a coin cell is connected to BVAT, the power state is retained when main power is disconnected. Without VBAT, the power state always defaults to 3.3V power on, even if the On/Off button had been used to turn off 3.3V before main VIN/USB power was removed.

## Memory



Four Memory Regions On Teensy 4.1

## • Program / Flash Memory

Teensy 4.1 has 8 Mbyte of flash memory intended for storing your code. The flash memory can also store read-only variables and arrays. A portion of the flash memory may be used for file storage using the LittleFS library. The top 256K of this memory is reserved for EEPROM emulation data and the LED blink restore program.

## • RAM

1024K of memory is available for variables and data. Half of this memory (RAM1) is accessed as tightly coupled memory for maximum performance. The other half (RAM2) is optimized for access by DMA. Normally large arrays & data buffers are placed in RAM2, to save the ultra-fast RAM1 for normal variables.

## • EEPROM

4284 bytes of emulated EEPROM memory is supported. Writing to this memory temporarily stalls code execution from flash. The [EEPROM library](#) is typically used to access this memory. AVR libc functions may also be used.

QSPI [PSRAM](#) & Flash



## • QSPI Memory Expansion

Teensy 4.1 has 2 locations to add 8 pin QSPI memory chips. Both locations support [8MB PSRAM chips](#). If only 1 PSRAM chip is used, it must be soldered to the smaller pads. The larger pads may be used with certain flash memory chips supported by the LittleFS library.

## • Static Allocation Keywords

When the compiler builds your program, all global variables, static variables, and compiled code is assigned to dedicated locations in memory. This is called static allocation, because the memory addresses are fixed. By default, allocation tries to use the ultra-fast DTCM & ITCM memory. The following keywords allow control over where the compiler will place your variables and code within the memory.

- **DMAMEM** - Variables defined with DMAMEM are placed at the beginning of RAM2. Normally buffers and large arrays are placed here. These variables can not be initialized, your program must write their initial values, if needed.
- **EXTMEM** - Variables defined with EXTMEM are placed in the optional PSRAM memory chip soldered to the QSPI memory expansion area on bottom side of Teensy 4.1. These variables can not be initialized, your program must write their initial values, if needed.
- **PROGMEM & F()** - Variables defined with PROGMEM, and strings surrounded by F() are placed only in the flash memory. They can be accessed normally, special functions normally used on 8 bit boards are not required to read PROGMEM variables.
- **FASTRUN** - Functions defined with "FASTRUN" are allocated in the beginning of RAM1. A copy is also stored in Flash and copied to RAM1 at startup. These functions are accessed by the Cortex-M7 ITCM bus, for the fastest possible performance. By default, functions without any memory type defined are treated as FASTRUN. A small amount of memory is typically unused, because the ITCM bus must access a memory region which is a multiple of 32K.
- **FLASHMEM** - Functions defined with "FLASHMEM" executed directly from Flash. If the Cortex-M7 cache is not already holding a copy of the function, a delay results while the Flash memory is read into the M7's cache. FLASHMEM should be used on startup code and other functions where speed is not important.

## • Dynamic Allocation

As your program runs, it may use all of the RAM which was not reserved by static allocation. Because the specific memory address for each variable is computed as your program runs, this is called dynamic memory allocation.

- **Local Variables** - Local variables, and also return addresses from function calls and the saved state from interrupts are placed on a stack which starts from the top of RAM1 and grown downward. The amount of space for local variable is the portion of RAM1 not used by FASTRUN code and the initialized and zeroed variables.
- **Heap** - Memory allocated by C++ "new" and C malloc(), and Arduino String variables are placed in RAM2, starting immediately after the DMAMEM variables.
- **Heap externo** : si se ha añadido PSRAM, se puede usar extmem\_malloc() para asignar esta memoria, comenzando inmediatamente después de las variables EXTMEM. Si no hay

PSRAM, `extmem_malloc()` asigna automáticamente memoria del heap normal en RAM2.

- **RAM del reloj de tiempo real**

El RTC cuenta con 16 bytes de memoria. Si se conecta una pila de botón a VBAT, el contenido de esta memoria se conserva incluso si la alimentación está apagada. Se accede a esta memoria mediante los registros de 32 bits `LPGPR0-LPGPR3`.

- **Tarjeta SD**

Un zócalo SD integrado permite usar tarjetas SD para almacenar grandes cantidades de datos. La biblioteca SD de Arduino se utiliza para acceder a la tarjeta mediante `SD.begin(BUILTIN_SDCARD)`. Este zócalo SD integrado utiliza una interfaz SDIO nativa de 4 bits para acceder a la tarjeta. También se pueden usar tarjetas SD a través de los pines SPI, con `SD.begin(cspin)`, utilizando el protocolo SPI de un solo bit, más lento.

- **Flash SPI**

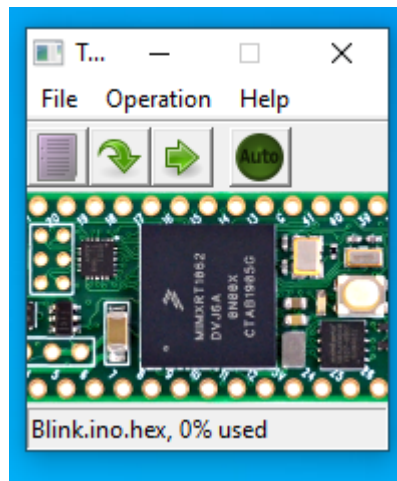
Se pueden añadir chips de memoria flash mediante los pines SPI. Estos son compatibles con las bibliotecas `SerialFlash` y `LittleFS`.

TODO: diagrama de la estructura interna del bus

## Programación

- **Cargador Teensy**

La memoria flash de Teensy se programa mediante la [aplicación Teensy Loader](#). Normalmente, se utiliza el IDE de Arduino u otro software para componer el código, y Teensy Loader se ejecuta automáticamente según sea necesario. Si ha compilado código en formato de archivo HEX, Teensy Loader puede usarse de forma independiente para escribir su archivo HEX en la memoria flash de Teensy.



Aplicación Teensy Loader

- **Entrada automática de software al modo de programa**

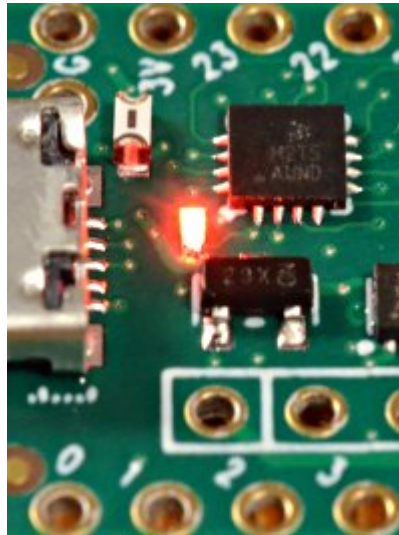
Al desarrollar con Teensy, la carga se realiza automáticamente tras compilar el programa. La utilidad `"teensy_reboot"` busca el Teensy en todos los puertos USB y envía una solicitud (velocidad de transmisión en serie o informe de características HID) para cambiar automáticamente al modo de programación.

- **Programa Pulsador / Pin**

Si el código previamente escrito en Teensy no detecta la comunicación USB, no es posible acceder automáticamente al modo de programación. Se incluye un botón físico para recuperarse de código erróneo. Al presionarlo, Teensy entra en modo de programación. No es un botón de reinicio que reinicia el programa. El botón está dedicado a la recuperación de código erróneo. Un pin de programa también permite que un hardware externo fuerce el acceso al modo de programación.

- **LED rojo: cargador de arranque activo y escritura flash**

Un LED rojo muestra el estado del cargador de arranque. Cuando el cargador de arranque está activo, pero esperando la comunicación con la PC, este LED rojo se atenúa. Mientras se escribe en la memoria flash, permanece encendido.



LED rojo: estado del cargador de arranque

- **Reiniciar**

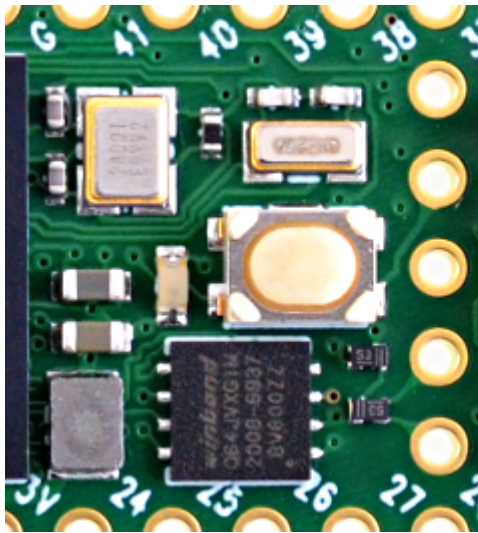
Teensy 4.1 *no* dispone de una señal de reinicio por hardware. El reinicio se puede realizar mediante software mediante los temporizadores de vigilancia o el registro SCB\_AIRCR.

- **Borrado de memoria y restauración de parpadeo de LED**

Teensy 4.1 will fully erase its non-volatile memory and return the flash memory to a simple LED blink program if the program button is held between 13 to 17 seconds. The red LED flashes briefly at the beginning of this time window. During flash erase, the red LED is on bright. When completed, Teensy 4.1 will automatically reboot and run the LED blink program, causing the orange LED to blink slowly.

- **Bootloader Chip**

Teensy 4.1's bootloader is stored in a dedicated chip. All of the main chip's memory is available to your program. Upon power up, your program runs immediately. The bootloader does not run automatically at startup, as is done with most Arduino compatible boards. The physically separate chip keeps Teensy's bootloader separate from your code and prevents flash programming from being able to damage or erase the bootloader.



**On/Off**  
**Program**  
**GND**  
**3.3V**  
**VBAT**

unencrypted programs can run. Secure mode permanently disables the ability to run unencrypted code, and activates hardware security features.

- **Lockable Teensy**

Secure mode can only be activated on Lockable



**Lockable Teensy - White Lock Stamp**

Teensy. While Standard and Lockable Teensy are identical hardware, the permanent fuse configuration differs. Standard Teensy does not allow changes to fuses affecting boot or other critical configuration. Standard Teensy is meant to safe from "bricking" by programs which could write to fuse memory, but this safety means secure mode can not be activated. Standard Teensy can have a key written and can run encrypted code, but encryption alone is not fully secure. **Only Lockable Teensy provides proper code security**, and only when a key is written and secure mode is locked.

- **Authentication**

The encryption process includes digital signature authentication. In secure mode, this signature is checked before any code can be decrypted.

- **JTAG Disable**

Secure mode permanently disables the JTAG port. To enter programming mode without JTAG, Teensy Loader and the EHEX file automatically utilize a loader utility which is authenticated by your key's digital signature, and in turn uses secure hash checks to fully authenticate all components of the programming process.

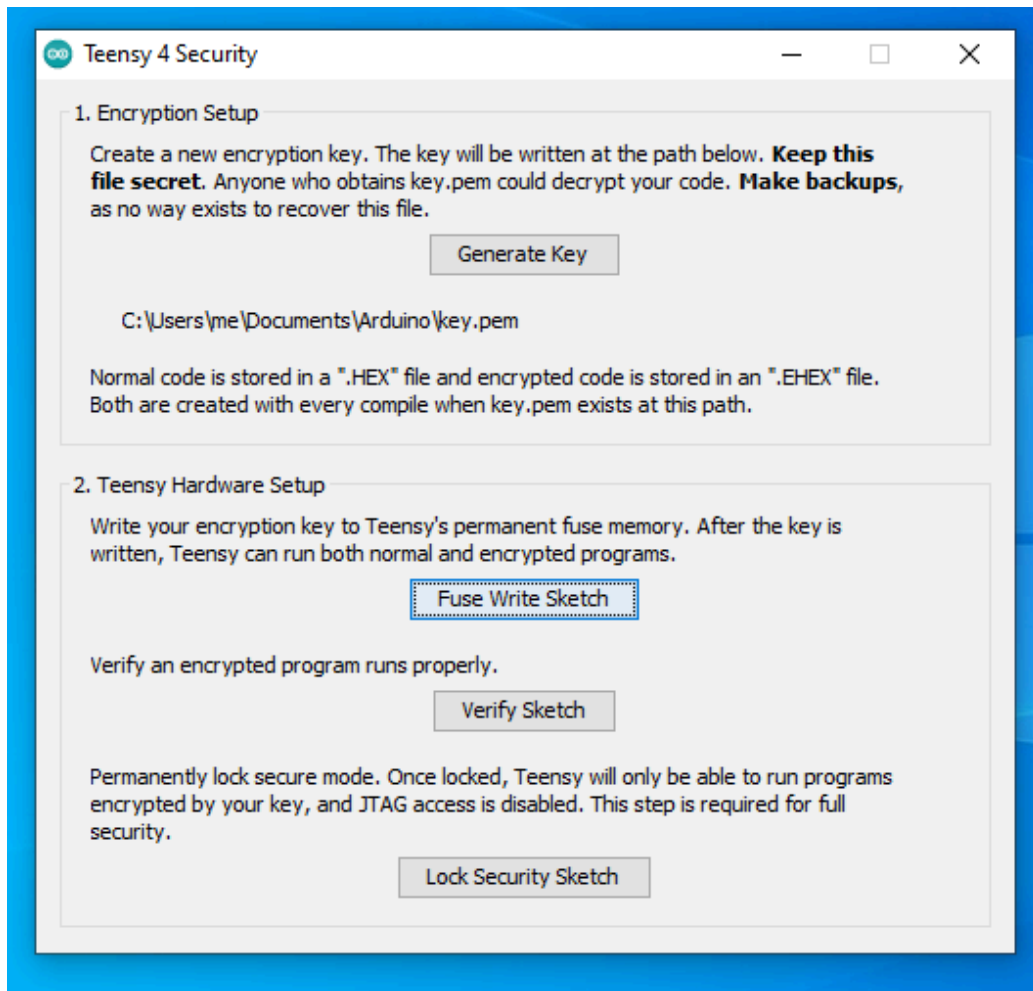
- **EHEX File Format**

Teensyduino packages your encrypted code, metadata, a startup shim, loader utility, digital signatures and other essential details into a single EHEX file. This EHEX may be given to customers or untrusted parties to perform code updates with the convenience of a single file. The EHEX format and encryption details are [documented on the code security page](#)

- **Key Management**

To make creating and using your key simple, Teensyduino adds a "Teensy 4 Security" window to the Arduino Tools menu. These functions can also be accessed from a command line utility for use from non-Arduino tools or automated scripts.





Tools > Teensy 4 Security - Create Your Key and Write to Fuse Memory

## Special Features

- **Nested Interrupt Controller**

Priority nesting allows low latency for critical interrupts while lower priority interrupts are in use. Teensyduino's libraries utilize interrupt nesting with priority level defaults which allow many types of libraries to work well when used together.

- **Direct Memory Access (DMA)**

Teensy 4.1 has a general purpose 32 channel DMA controller. Optimized Audio, LED and display libraries make use of these DMA channels. A DMACHannel.h abstraction layer is provided. The USB device, USB host, SD card and Ethernet peripherals also have specialized DMA engines built in.

- **Random Number Generator**

True random number hardware is capable of generating random data at (TBD) rate. The Entropy library is used to access the random number generator.

- **Cryptographic Acceleration**

Symmetric ciphers and one-way hash can be computed by hardware, but currently no library support exists to utilize this hardware.

- **Temperature Sensor**

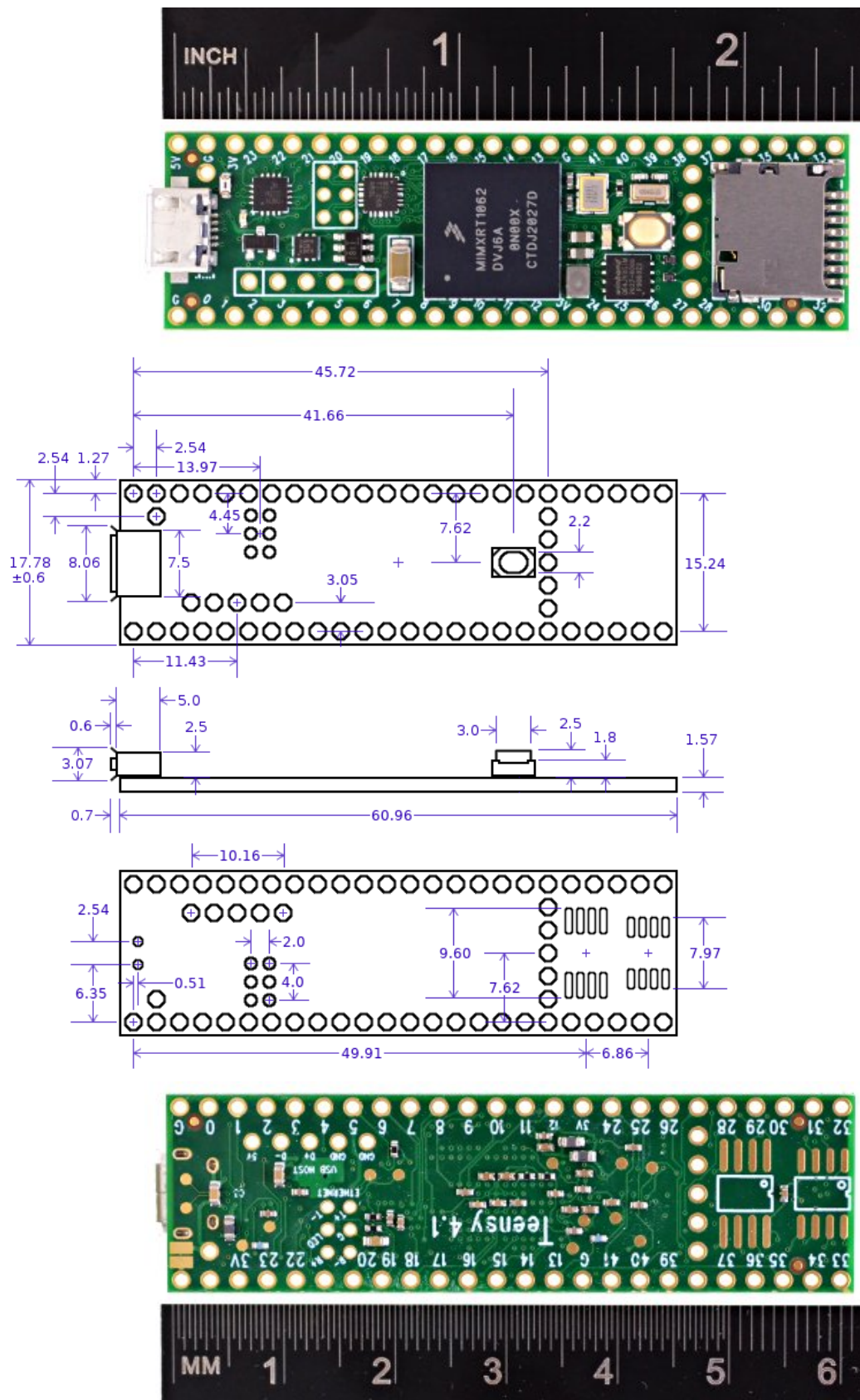


Un sensor de temperatura integrado permite leer la temperatura dentro del chip principal. La [biblioteca InternalTemperature](#) permite acceder a este sensor.

## Información técnica

- [Manual del IMXRT1060](#) : Toda la información útil sobre programación de periféricos (con anotaciones para Teensy)
- [Hoja de datos IMXRT1060](#) : solo las especificaciones eléctricas
- [IMXRT1060 Errata Rev B](#) : Aplicable a Teensy 4.1 después de julio de 2021
- [IMXRT1060 Errata Rev A](#) : Aplicable a Teensy 4.1 antes de abril de 2021
- Hoja de datos del chip de memoria flash [W25Q64JV-DTR](#)
- [Hoja de datos del DP83825j](#) : chip Ethernet PHY
- [Manual de referencia de ARM Cortex-M7](#)
- [Manual de referencia de la arquitectura ARM v7-M](#) : detalles de bajo nivel del procesador ARM (gratuito, pero difícil de leer)
- [Guía definitiva de ARM Cortex-M3 y Cortex-M4](#) (libro): detalles de bajo nivel del procesador ARM (más fácil de leer)
- [Declaración de la Sección 889 de EE. UU.](#)
- [Certificado de conformidad con RoHS](#)
- [Declaración de conformidad REACH](#)

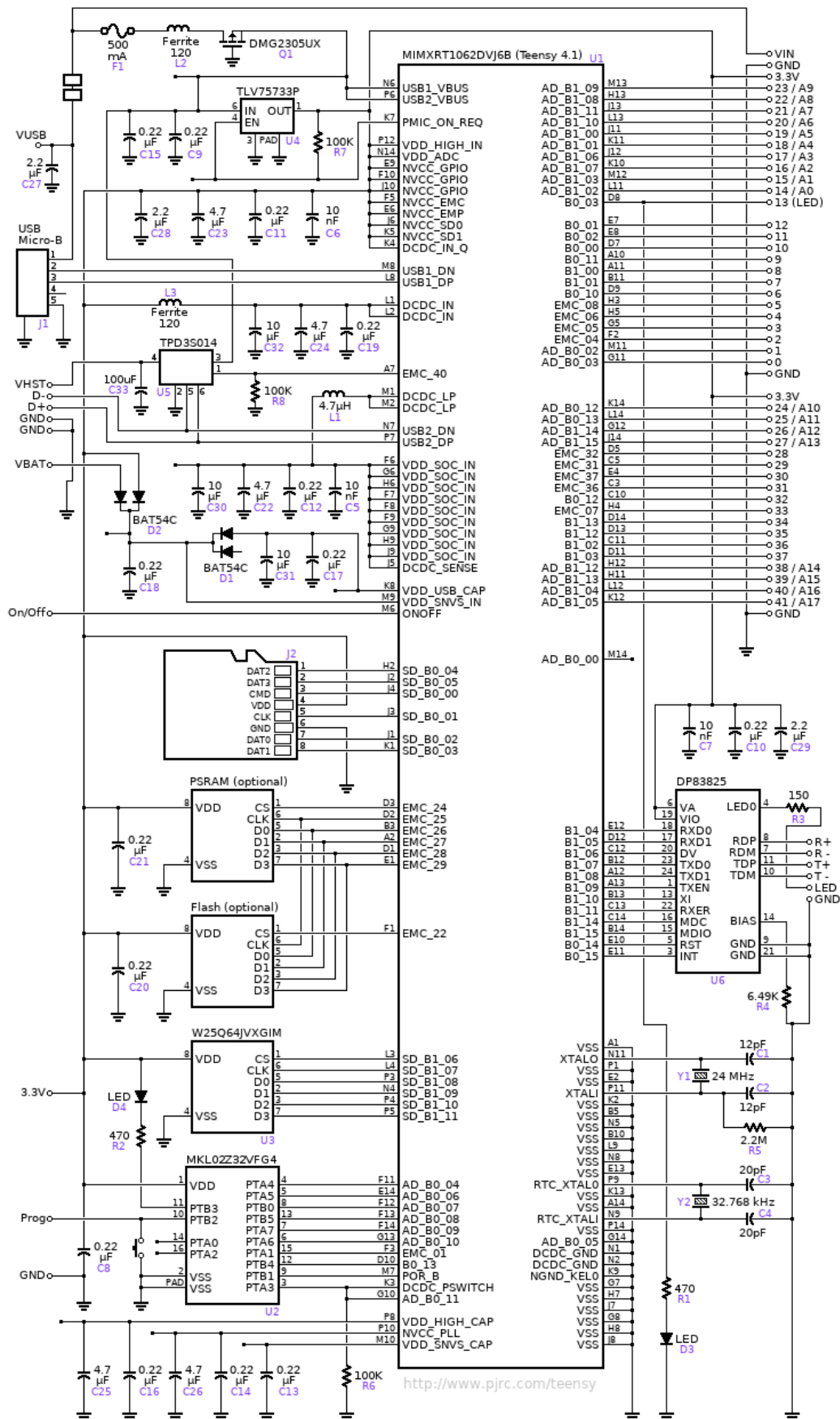
## Dimensiones



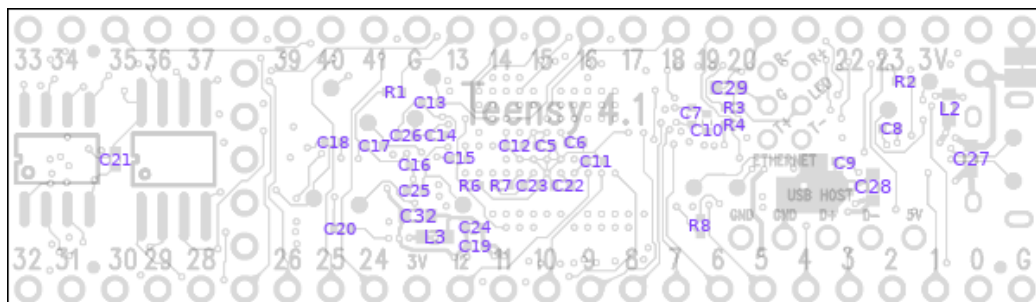
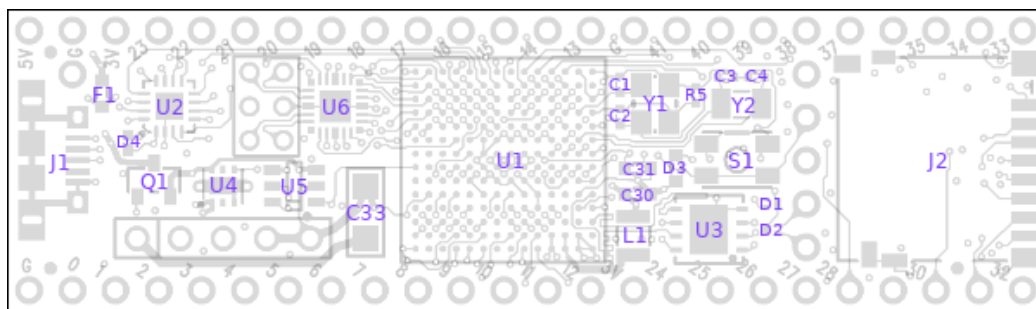
El diámetro del orificio terminado es de 0,965 mm (59 ubicaciones) y 0,762 mm (6 ubicaciones).

Es posible que los modelos CAD 3D aportados por los usuarios estén disponibles en el foro.

## Esquemático



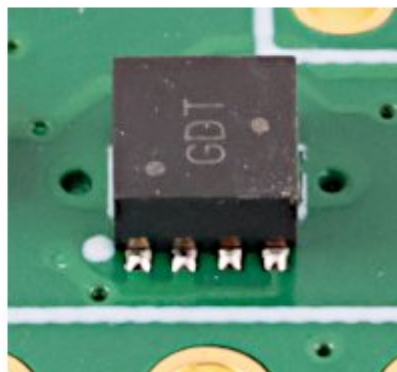
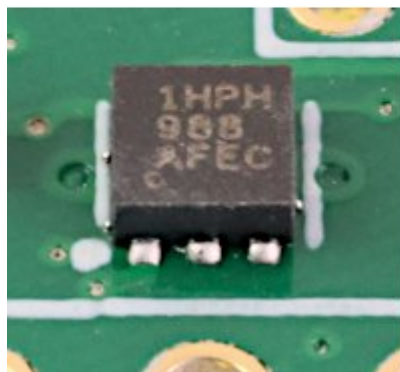
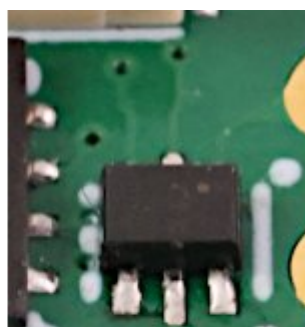
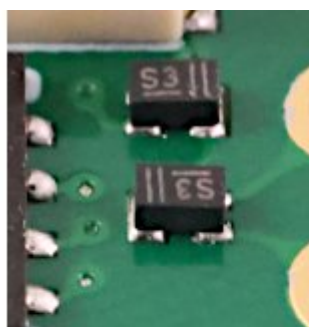
## Ubicación de los componentes



## Cambios de diseño

El Teensy 4.1 fabricado después de julio de 2021 tiene el número de pieza U1 MIMXRT1062DVJ6B. Las placas anteriores tenían la versión anterior terminada en "A". La versión "B" corrige errores de hardware poco conocidos con el bus CAN y el modo isócrono de dispositivos USB.

Los Teensy 4.1 fabricados después de marzo de 2022 tienen TLV75733P (U4) reemplazado por NCV8186AMN330TAG y BAT54C (D1, D2) reemplazado por BAS40-05V.



En el Teensy 4.1 fabricado después de junio de 2022, el U2 se reemplazó por el GD32E230F8. Consulte este [hilo del foro](#) para conocer las nuevas ubicaciones de los pines JTAG.

Teensy 4.1 fabricado después de agosto de 2022 tiene R7 cambiado a 470K, lo que ayuda con el arranque con VUSB-VIN inferior a 3,8 V.



Los Teensy 4.1 fabricados después de marzo de 2023 tienen U4 reemplazado por TLV75533P, D1-D2 reemplazado por BAT54CTB6 y un [zócalo SD diferente](#) (J2).

## Puntos de prueba

