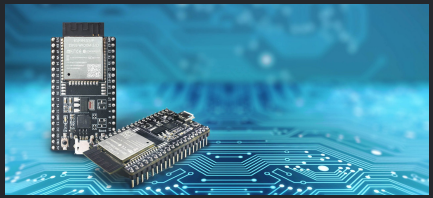
Sensores

Introducción al Internet de las Cosas



Marco Teran
Universidad Sergio Arboleda

2023

Contenido

Introducción al IoT y los sensores

Introducción al IoT y los sensores

- El IoT es un sistema de dispositivos interconectados que recopilan y comparten datos a través de Internet.
- Los sensores son dispositivos que se utilizan para medir y detectar cambios en el entorno físico, como la temperatura, la humedad, la presión, el movimiento, la luz y el sonido.
- Los sensores son esenciales para el IoT, ya que permiten a los dispositivos recopilar datos del mundo físico y transmitirlos a través de la red.

Sistemas embebidos y sensores

Sistemas embebidos y sensores

- Los sistemas embebidos son sistemas informáticos integrados en dispositivos físicos, como automóviles, electrodomésticos, cámaras y dispositivos médicos.
- Los sensores se utilizan comúnmente en los sistemas embebidos para medir variables como la temperatura, la humedad, la presión, el movimiento y la luz.
- Los sistemas embebidos a menudo requieren sensores altamente precisos y confiables para garantizar un funcionamiento seguro y efectivo.

Sistemas embebidos y sensores

- El procesamiento en sistemas embebidos simples consiste en leer entradas, computar salidas, escribir salidas y esperar hasta el próximo ciclo de procesamiento o evento de activación.
- En sistemas embebidos, las entradas generalmente consisten en comandos ingresados por un usuario, comandos recibidos de otras fuentes como un servidor de red, y mediciones de sensores.

Sistemas embebidos y sensores

Sensores

Los sensores son componentes eléctricos o electrónicos sensibles a alguna propiedad de su entorno y producen una salida correspondiente a la propiedad medida.

- Los sensores pasivos miden aspectos del entorno como temperatura o intensidad de luz respondiendo directamente al parámetro medido sin perturbar el entorno.
- Los sensores activos generan algún tipo de estímulo que aplican al entorno y miden la respuesta.

Sensores

- Los sensores inteligentes integran la función de medición y conversión en una sola unidad, lo que simplifica el diseño de hardware.
- Los circuitos básicos interfazan los tipos comunes de sensores con los procesadores embebidos.
- Algunos tipos de sensores pueden operar en ambos modos, pasivo y activo.
- Ejemplos de sensores activos incluyen el sensor de distancia ultrasónico, radar y sensores LIDAR.
- La elección de un sensor inteligente en aplicaciones específicas dependerá de factores como el costo del sensor, el volumen de producción anticipado y la presión del tiempo de mercado.

Tipos de sensores para el IoT

Tipos de sensores para el IoT

- Los sensores utilizados en el IoT incluyen sensores de temperatura, humedad, presión, movimiento, luz, sonido, posición y gas.
- Los sensores de temperatura miden la temperatura ambiental o de un objeto en particular.
- Los sensores de humedad miden la cantidad de humedad en el aire o en un objeto.
- Los sensores de presión miden la presión atmosférica o la presión en un objeto.
- Los sensores de movimiento detectan cambios en la posición, velocidad o aceleración de un objeto.

Sensores de luz

Sensores de luz

- Los sensores de luz en sistemas embebidos pueden variar desde fotoresistores simples hasta complejos arreglos de sensores de múltiples bandas utilizados en dispositivos como cámaras de video, microscopios y telescopios astronómicos.
- Los fotodiodos y fototransistores son dispositivos semiconductores que convierten la luz en corriente eléctrica, son más sensibles y consistentes que los fotoresistores.
- Los sensores de video contienen una matriz bidimensional de elementos sensibles a la luz, con filtrado que limita la entrada de cada elemento a un rango de frecuencia específico dentro del espectro de luz.

Sensores de temperatura

Sensores de temperatura

- Un termistor es un elemento resistivo que cambia la resistencia con los cambios de temperatura y se puede medir su resistencia a través de un circuito divisor de voltaje.
- Un termopar es un dispositivo de detección de temperatura que se conecta a dos metales diferentes en un solo punto para medir la temperatura en ese lugar.
- Los termistores son más adecuados para rangos de temperatura de -50 a 250 °C, mientras que los termopares soportan un rango de temperatura más amplio de aproximadamente -200 a 1,250 °C.

Sensores de presión

Sensores de presión

- Los sensores de presión miden la presión de líquidos y gases, ya sea de forma absoluta o relativa a algún punto de referencia como la presión atmosférica.
- Los sensores de presión diferencial miden la diferencia de presión entre dos ubicaciones, y se usan en aplicaciones como medir la caída de presión a través de un filtro de fluido.
- Los sensores de presión de bajo costo suelen ser construidos con un material piezorresistivo que cambia en resistencia en respuesta a la presión medida, y el sistema integrado mide la resistencia del sensor para determinar la lectura de presión.

Sensores de humedad

Sensores de humedad

- Los sensores de humedad miden la presión parcial del vapor de agua en el ambiente y la expresan en un porcentaje relativo a la presión de saturación.
- Son útiles en aplicaciones de monitoreo y control ambiental para mantener la humedad dentro de límites deseados para equipos electrónicos o confort humano, y a menudo se combinan con sensores de temperatura.
- Los sensores de humedad de bajo costo se construyen con elementos sensibles que varían su capacidad según la humedad y miden la capacitancia del elemento para calcular la humedad relativa, y algunos son "smart sensorsçon capacidad de procesamiento y comunicación digital.

Sensores de flujo

Sensores de flujo

- Los sensores de flujo miden la cantidad de gas o líquido que pasa por una región activa del sensor, generalmente mediante la colocación de algún tipo de restricción en el fluido y midiendo el efecto de la restricción.
- Algunos diseños de sensores de flujo utilizan tecnologías de medición que no requieren restricción de flujo, como ultrasonido y láser. Estos sensores pueden tener salida analógica o interfaz digital.
- Los sensores de flujo se usan en una amplia variedad de aplicaciones, desde monitorear el consumo de combustible en automóviles y aviones hasta medir el flujo de anestésicos en aplicaciones médicas, así como en medidores de agua y gas en hogares y edificios comerciales.

Sensores de fuerza

Sensores de fuerza

- Un sensor de fuerza mide la cantidad de fuerza aplicada a un objeto, como una báscula de baño.
- Los sensores de fuerza comúnmente utilizan resistores de fuerza, piezoeléctricos o fluidos/gases bajo presión para medir la fuerza aplicada.
- Los sensores de fuerza se utilizan en una amplia variedad de aplicaciones, desde medir el peso de una persona en una báscula hasta controlar la fuerza en maquinaria y robots industriales.

Sensores ultrasónicos

Sensores ultrasónicos

- Un sensor ultrasónico genera una onda de sonido de alta frecuencia que se transmite a una región de medición y escucha los ecos de los objetos en el rango.
- El tiempo entre la transmisión y recepción de la señal, multiplicado por la velocidad del sonido en el medio de medición, representa la distancia de ida y vuelta del transmisor al objeto y de vuelta al receptor.
- Los sensores ultrasónicos simples utilizan dos señales digitales para controlar la operación del sensor y leer la salida de medición. El pin de disparo es una señal de entrada del sensor que inicia un ciclo de medición en respuesta a un borde de pulso ascendente. El pin Echo es una salida del sensor que se activa cuando se transmite el pulso y vuelve a desactivarse cuando se recibe el eco.

Sensores de audio

Sensores de audio

- Los sensores de audio reciben señales sonoras en el rango de audición humana y producen una salida eléctrica en respuesta a la señal recibida. Un micrófono estándar es un ejemplo de sensor de audio.
- Los dispositivos de asistente inteligente escuchan continuamente el sonido en su entorno y, cuando se detecta una secuencia de sílabas de activación (como .^Alexa.° "Hey Google"), el dispositivo graba los sonidos siguientes e intenta interpretar el comando proporcionado como una secuencia de palabras.
- En aplicaciones más simples, un sensor de audio puede simplemente monitorear la intensidad del sonido en su entorno y producir una salida cuando el nivel de sonido supera un umbral. Las aplicaciones más sofisticadas del monitoreo de audio incluyen la detección de rotura de

Sensores magneticos

Sensores magneticos

- Un sensor de campo magnético, o magnetómetro, detecta la presencia de un campo magnético en la cercanía del sensor. Los sensores simples responden solo a la intensidad del campo.
- Los sensores más sofisticados miden los componentes vectoriales tridimensionales del campo magnético.
- Un ejemplo de uso es el magnetómetro que detecta el campo magnético terrestre, la cual puede ser calibrada para obtener una orientación más precisa basada en la ubicación en la que se encuentra. También se utilizan sensores magnéticos en sistemas de seguridad para detectar la apertura de puertas mediante la presencia de un imán que genera un campo magnético local.

Sensores químicos

Sensores químicos

- Los sensores químicos miden atributos de componentes químicos en la cercanía del sensor, siendo sensibles a la presencia de un elemento o compuesto particular en el gas o líquido que lo rodea.
- Ejemplos comunes de sensores químicos incluyen detectores de monóxido de carbono y radón en hogares, y los motores de gasolina modernos contienen sensores de oxígeno en el sistema de escape.
- Sensores de bajo costo y un solo chip están disponibles para medir la calidad del aire, reportando los niveles de dióxido de carbono y compuestos orgánicos volátiles en el aire circundante, y la nanotecnología está permitiendo la producción de una variedad de dispositivos de detección química.

Sensores ionizantes

Sensores ionizantes

- La radiación ionizante se compone de partículas electromagnéticas capaces de ionizar átomos y moléculas, y es perjudicial para el tejido vivo a niveles bajos.
- Tanto la radiación natural como la generada por el hombre se puede medir con sensores de radiación ionizante que miden los cambios en un material sensible debido a la radiación incidente.
- Los sensores de radiación ionizante pueden informar eventos individuales o acumular la dosis de radiación en un período de tiempo.

Sensores RADAR

Sensores RADAR

- RADAR es un acrónimo para Radio Detection and Ranging, que se refiere a un sistema que transmite señales de radio y recibe eco de los objetos encontrados por los pulsos.
- Los sensores de radar de un solo chip están disponibles para la detección de vehículos cercanos en aplicaciones como el control de crucero adaptativo en automóviles.
- Los sensores de radar automotriz miden la distancia y la velocidad relativa del vehículo por delante del vehículo anfitrión y permiten mantener la distancia adecuada y responder a eventos como frenadas repentinas del vehículo delantero.

Sensores LIDAR

Sensores LIDAR

- LIDAR es un acrónimo de Light Detection and Ranging, que utiliza luz láser o infrarroja para detectar objetos en la proximidad del sensor, en lugar de señales de radiofrecuencia como el radar.
- El sensor LIDAR emite un pulso de luz enfocado y mide la distancia a un obstáculo en la dirección del haz mediante la detección del tiempo de recepción de la reflexión del objeto. Repite este proceso miles de veces para crear un mapa tridimensional de la zona.
- Aunque los componentes ópticos necesarios para los sensores LIDAR de precisión son costosos, su precio ha disminuido en los últimos años gracias al desarrollo de la tecnología de vehículos autónomos.

Sensores de vídeo e infrarrojos

Sensores de vídeo e infrarrojos

- Cámaras de video digital color tradicionales producen una secuencia de imágenes compuestas de píxeles que contienen tres intensidades de color (rojo, verde y azul).
- Sistemas integrados pueden utilizar cámaras de video para capturar escenas para consumo humano, como en una aplicación de videoportero.
- Algunas áreas de aplicación requieren el procesamiento de flujos de datos de video por máquinas, como en un sistema de seguridad que monitorea una zona y detecta movimiento comparando imágenes secuenciales.

Sensores de vídeo e infrarrojos

- Vehículos autónomos necesitan un sistema de procesamiento de video más sofisticado que el simple diferencial de imágenes, integrando varios tipos de datos de sensores incluyendo cámaras de video.
- Cámaras infrarrojas son similares a las de video, pero sensibles a longitudes de onda de luz más largas que el rojo visible, respondiendo al calor radiado en el infrarrojo, útiles en aplicaciones como monitoreo de temperaturas en circuitos eléctricos.
- El procesamiento de los datos de video en vehículos autónomos requiere algoritmos de inteligencia artificial sofisticados y procesamiento de alta velocidad y en tiempo real.

Sensores inerciales

Sensores inerciales

- Un sensor inercial detecta cambios en el movimiento del cuerpo al que está unido.
- Para caracterizar completamente la aceleración y la rotación de un objeto en el espacio tridimensional se requieren tres acelerómetros y tres sensores de velocidad de rotación.
- Los sensores inerciales se utilizan en aeronaves, naves espaciales y barcos para rastrear con precisión el movimiento en presencia de perturbaciones y deben tener errores de medición lo suficientemente pequeños para no degradar la medición de posición a un grado inaceptable.

Sensores GNSS

Sensores GNSS

- Los receptores del Sistema de Posicionamiento Global (GPS) recopilan señales de satélites en órbita alrededor de la Tierra para calcular la posición del receptor y proporcionar la hora actual.
- Varios sistemas de navegación por satélite constelaciones están en funcionamiento en todo el mundo, incluyendo Galileo, Beidou y Glonass.
- Los receptores modernos de navegación por satélite son capaces de recibir señales de una o varias constelaciones simultáneamente, lo que aumenta la precisión y la velocidad de la medición de la posición.

Sensores GNSS

Sensores GNSS

- Los receptores GNSS son comunes en aeronaves, automóviles, dispositivos militares, teléfonos inteligentes y otros equipos que operan en exteriores.
- La integración de sensores inerciales con un receptor GNSS crea un sistema GPS/INS, que se utiliza ampliamente en la navegación aérea, naval y militar.
- El siguiente tema tratará sobre las tecnologías de comunicación más útiles para conectar sensores a procesadores integrados.

Procesamiento de los datos de sensor

Procesamiento de los datos de sensor

- Los errores de medición en los sensores pueden tener diversas causas.
- En algunas aplicaciones, no es necesario corregir el error de medición, pero en otras sí es crítico eliminarlo tanto como sea posible.
- Los sensores de precisión suelen ser más costosos, pero pueden incluir técnicas de diseño que mejoran la calidad de la medición.

Procesamiento de los datos de sensor

- En algunos casos, es necesario realizar una calibración adicional para mejorar la precisión de la medición.
- Para reducir el ruido aleatorio en las mediciones de los sensores, se pueden tomar varias lecturas y promediarlas.
- En situaciones con señales que varían rápidamente, puede ser útil construir un filtro digital para procesar la señal de entrada.

Métricas de desempeño de los sensores

Métricas de desempeño de los sensores

Métricas de desempeño

Las métricas de desempeño de los sensores son medidas que se utilizan para evaluar la calidad y precisión de los datos que se recopilan.

- La resolución se refiere a la capacidad de un sensor para detectar cambios pequeños en una variable medida.
- La precisión se refiere a la medida en que los datos del sensor son comparables con un valor de referencia.
- La linealidad se refiere a la relación entre la entrada del sensor y la salida del sensor.
- La estabilidad se refiere a la capacidad de un sensor para mantener su precisión a lo largo del tiempo.

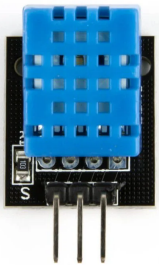
Desafíos en la implementación de sensores para el IoT

Sistemas embebidos y sensores

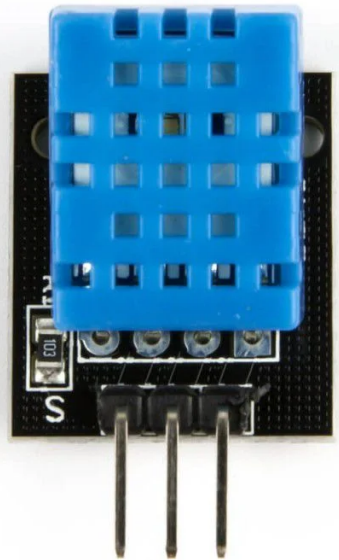
- Seleccionar los sensores adecuados para una tarea específica puede ser un desafío, ya que se requiere una comprensión detallada de los parámetros de rendimiento del sensor y las condiciones del entorno.
- La calibración y la integración de los sensores en el sistema embebido también pueden presentar desafíos, ya que es importante asegurarse de que los sensores sean precisos y funcionen correctamente.
- Además, los desafíos en la transmisión de datos pueden incluir problemas de seguridad y privacidad de los datos transmitidos.
- Otros desafíos incluyen la conservación de energía de los sensores, el procesamiento de datos en tiempo real y la compatibilidad con diferentes plataformas y protocolos de red.

Laboratorio de sensores

Introducción al DHT11

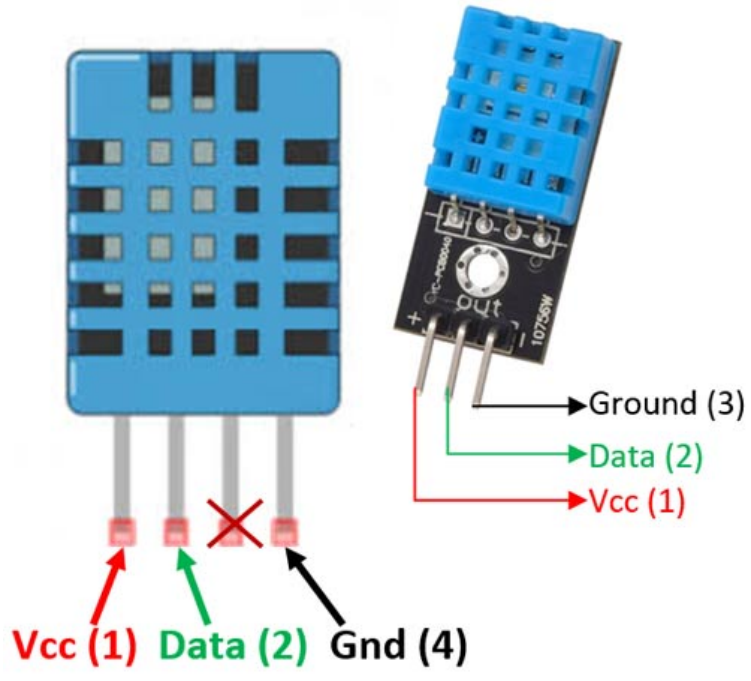


- El DHT11 es un sensor de humedad y temperatura de bajo costo.
- Es fácil de usar y se puede conectar directamente a una placa Arduino o Raspberry Pi.
- Proporciona mediciones precisas de la humedad relativa y la temperatura ambiente.


S

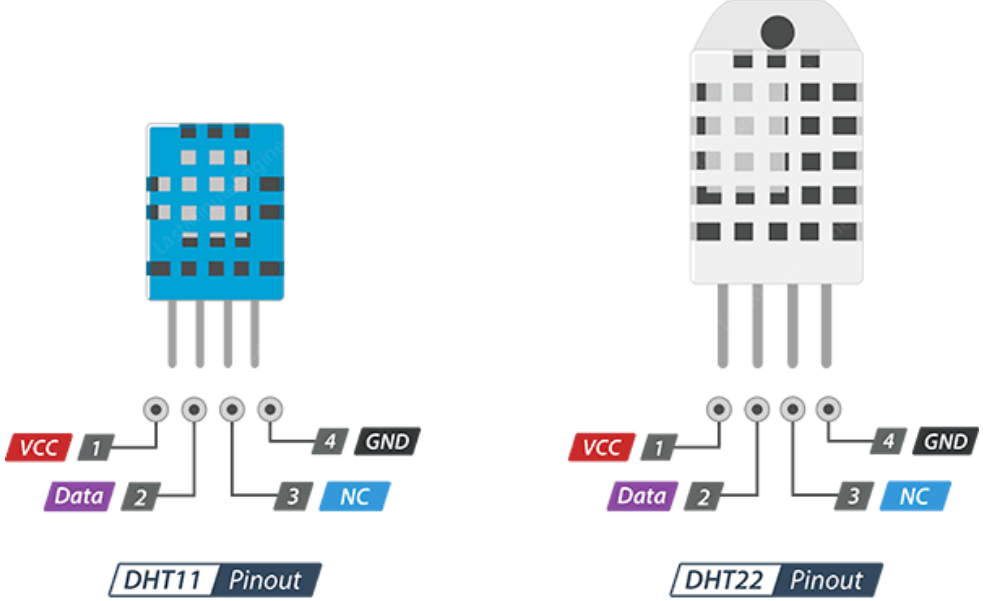
Importancia del DHT11 en el IoT

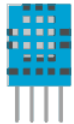
- El DHT11 es un sensor esencial en aplicaciones de IoT relacionadas con el monitoreo ambiental y control de clima.
- Puede utilizarse en combinación con otros sensores para mejorar la precisión de los datos recopilados.
- La facilidad de uso y bajo costo lo hacen ideal para proyectos de IoT de bajo presupuesto.

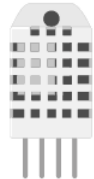

Ground (3)
Data (2)
Vcc (1)
Vcc (1)
Data (2)
Gnd (4)

Principales características del DHT11

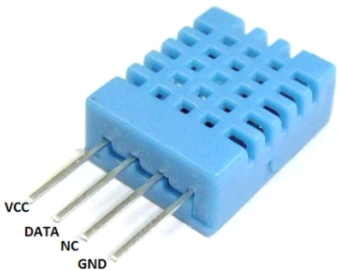
- Rango de medición de humedad: 20 % a 90 % HR con una precisión de ± 5 %.
- Rango de medición de temperatura: 0 a 50 grados Celsius con una precisión de ± 2 grados.
- Voltaje de funcionamiento: 3.3V a 5V.
- Salida digital con una resolución de 1 grado Celsius y 1 % de humedad relativa.
- Tiempo de respuesta rápido: 2 segundos para la humedad y 5 segundos para la temperatura.
- Bajo consumo de energía: menos de 1mA en promedio.

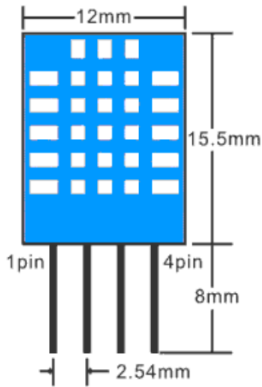

VCC
1
Data
2
3
NC
4
GND
DHT11 Pinout
VCC
1
Data
2
3
NC
4
GND
DHT22 Pinout
Last Minute
ENGINEERS.com

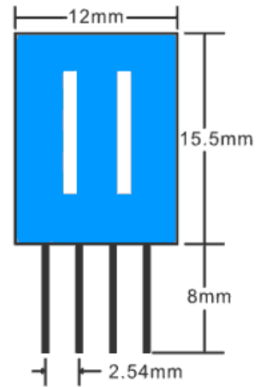


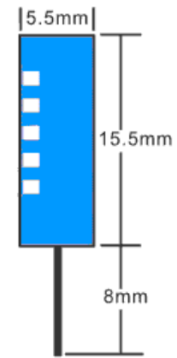


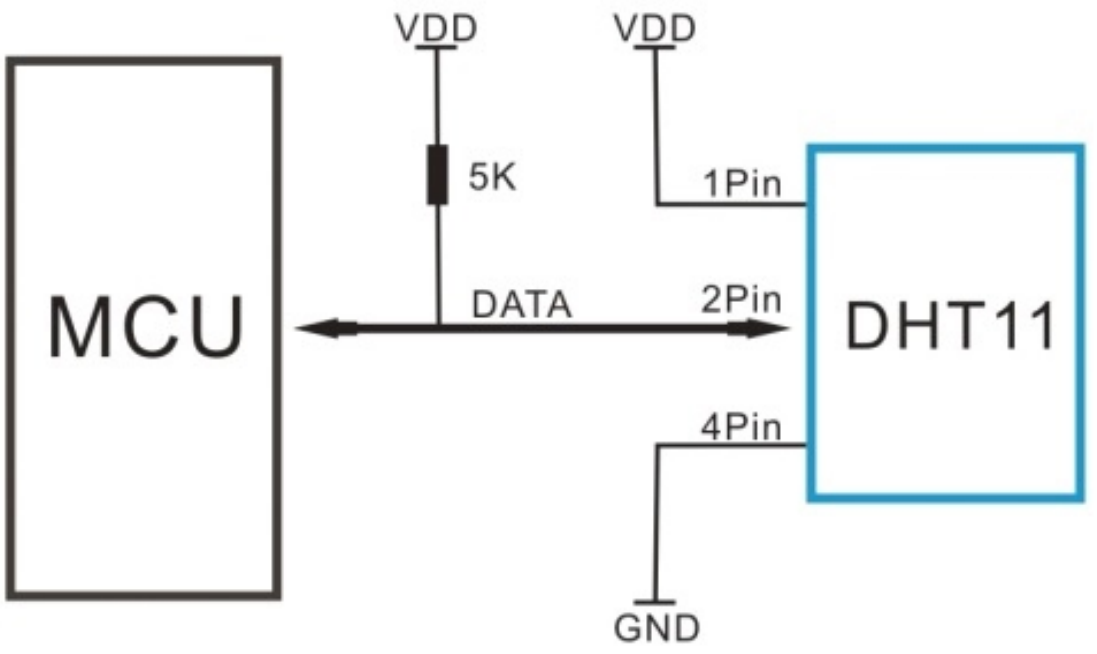
	DHT11	DHT22
Operating Voltage	3 to 5V	3 to 5V
Max Operating Current	2.5mA max	2.5mA max
Humidity Range	20-80% / 5%	0-100% / 2-5%
Temperature Range	0-50°C / ± 2°C	-40 to 80°C / ± 0.5°C
Sampling Rate	1 Hz (reading every second)	0.5 Hz (reading every 2 seconds)
Body size	15.5mm x 12mm x 5.5mm	15.1mm x 25mm x 7.7mm
Advantage	Ultra low cost	More Accurate

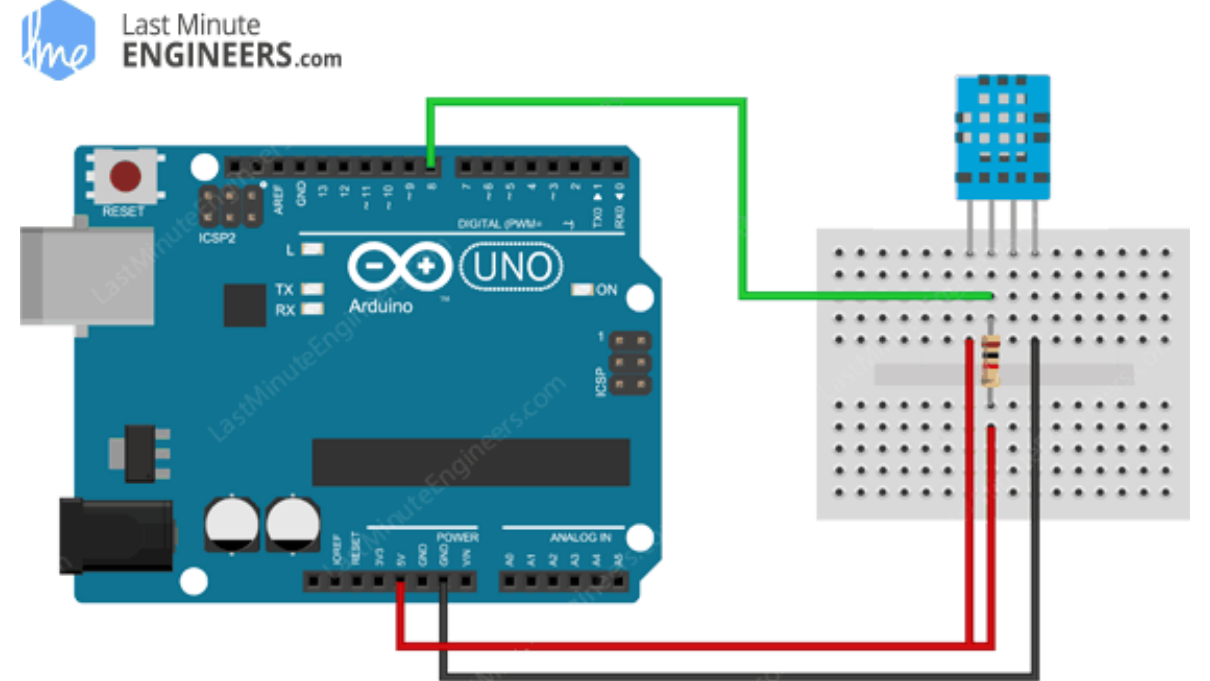

VCC
DATA
NC
GND

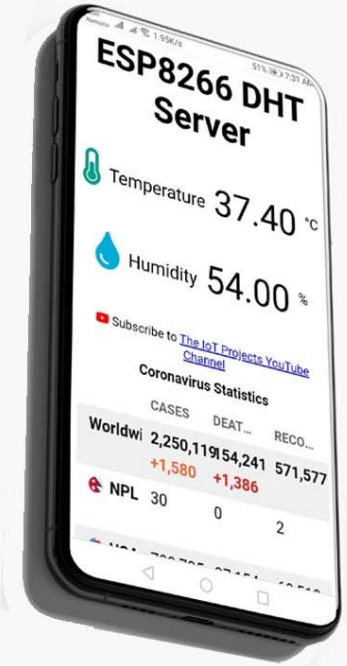

12mm
15.5mm
1pin
4pin
8mm
2.54mm

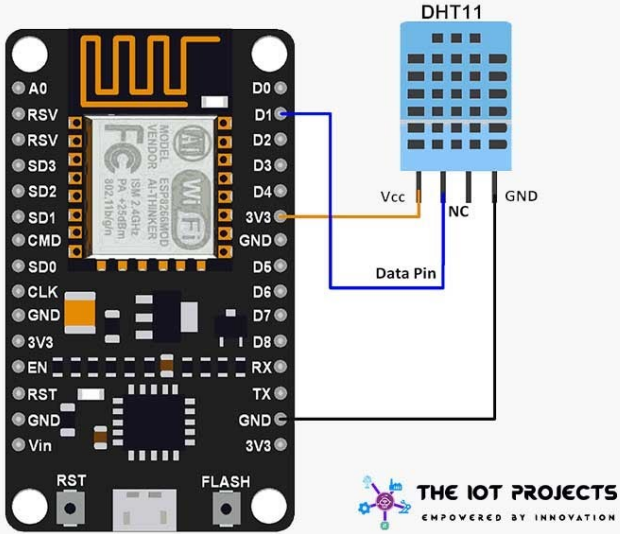

12mm
15.5mm
8mm
2.54mm


5.5mm
15.5mm
8mm


VDD
VDD
5K
1Pin
MCU
DATA
2Pin
DHT11
4Pin
GND

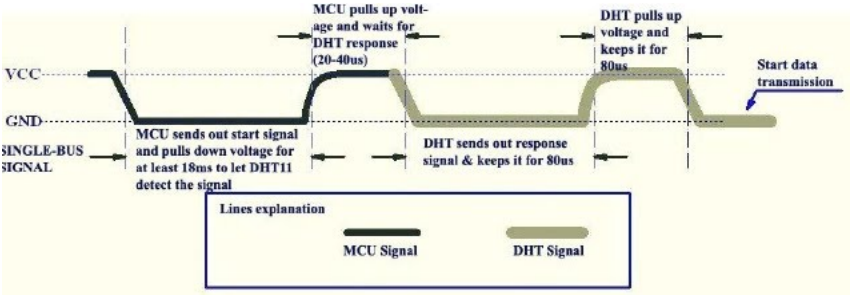

Last Minute
ENGINEERS.com
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
L
TX
RX
Arduino
UNO
ON
ICSP
POWER
ANALOG IN
IOREF
RESET
3V3
5V
GND
GND
VIN
A0
A1
A2
A3
A4
A5


ESP8266 DHT Server
Temperature 37.40 °C
Humidity 54.00 %
Subscribe to The IoT Projects YouTube Channel
Coronavirus Statistics
CASES
DEAT...
RECO...
Worldwi
2,250,119
54,241
571,577
+1,580
+1,386
NPL
30
0
2


DHT11
Vcc
NC
GND
Data Pin
A0
RSV
RSV
SD3
SD2
SD1
CMD
SD0
CLK
GND
3V3
EN
RST
GND
Vin
D0
D1
D2
D3
D4
3V3
GND
D5
D6
D7
D8
RX
TX
GND
3V3
RST
FLASH
THE IOT PROJECTS
EMPOWERED BY INNOVATION

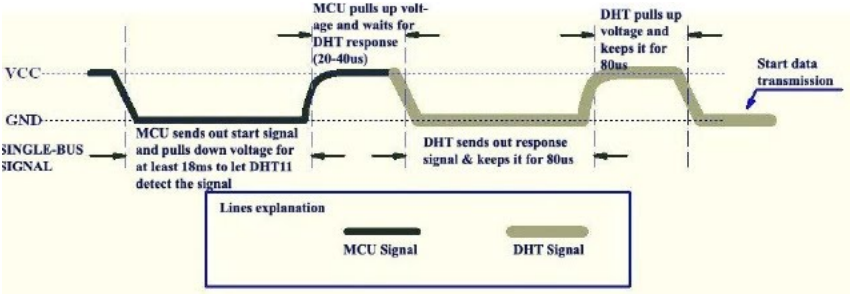
DHT11: protocolo OneWire

- Para comunicarnos con el sensor debemos de implementar en nuestra MCU el protocolo que el fabricante nos proporciona el datasheet es el siguiente:
- La MCU se comunicará con el DHT y éste le enviará 5 bytes de datos donde los 2 primeros corresponderán a la Humedad Relativa, los 2 siguientes a la Temperatura y el último byte para el Checksum.



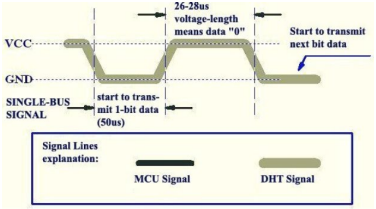
DHT11: protocolo OneWire

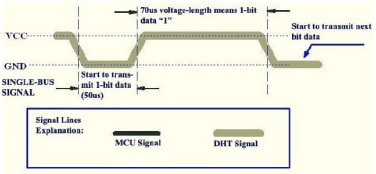
- Pimero de todo MCU mandará una secuencia de START al DHT y después el mismo esperará a que el DHT responda. Después el DHT mandará una señal al MCU para indicar de que está preparado para inicial la trasmisión bit a bit.

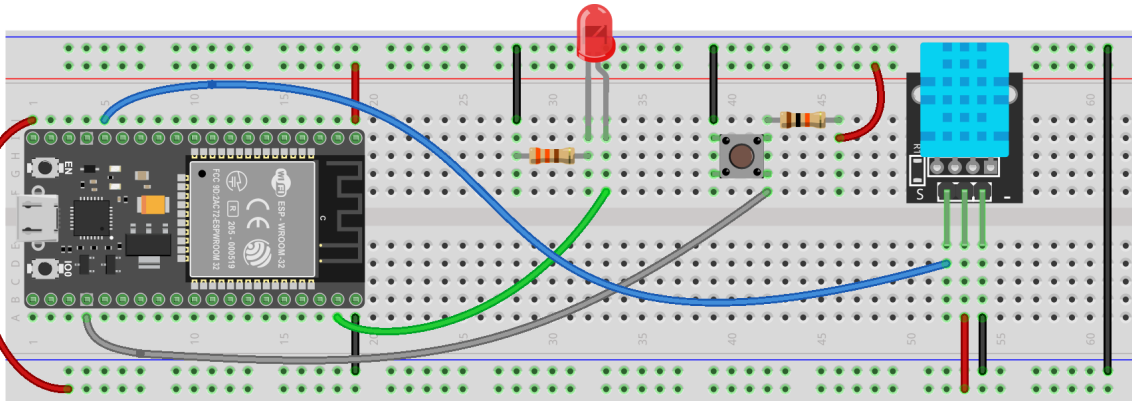


DHT11: protocolo OneWire

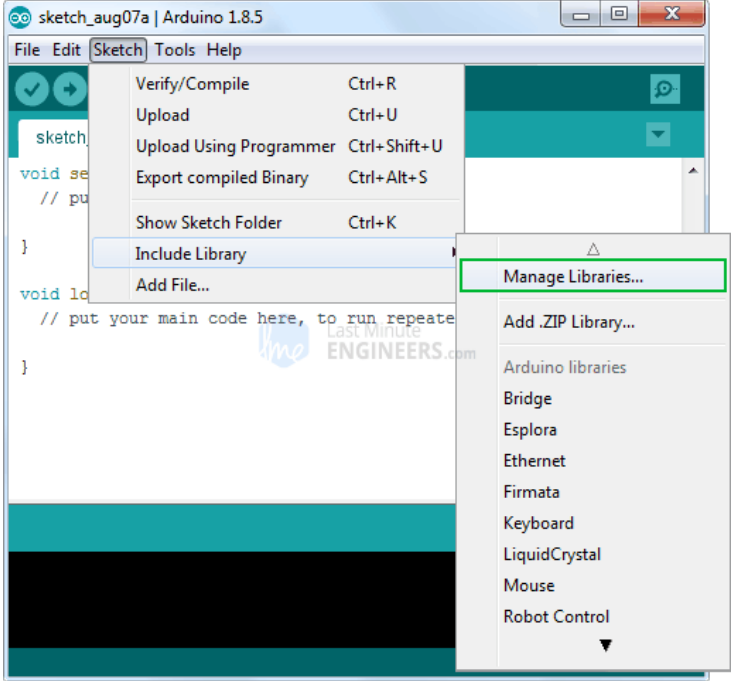
- Siempre que inicia la trasmisión de un bit el sensor, mandará un pulso de bajo voltaje de 50us y después enviará un pulso de alto voltaje que según la duración del mismo el pulso significará que es un "0.º un "1".

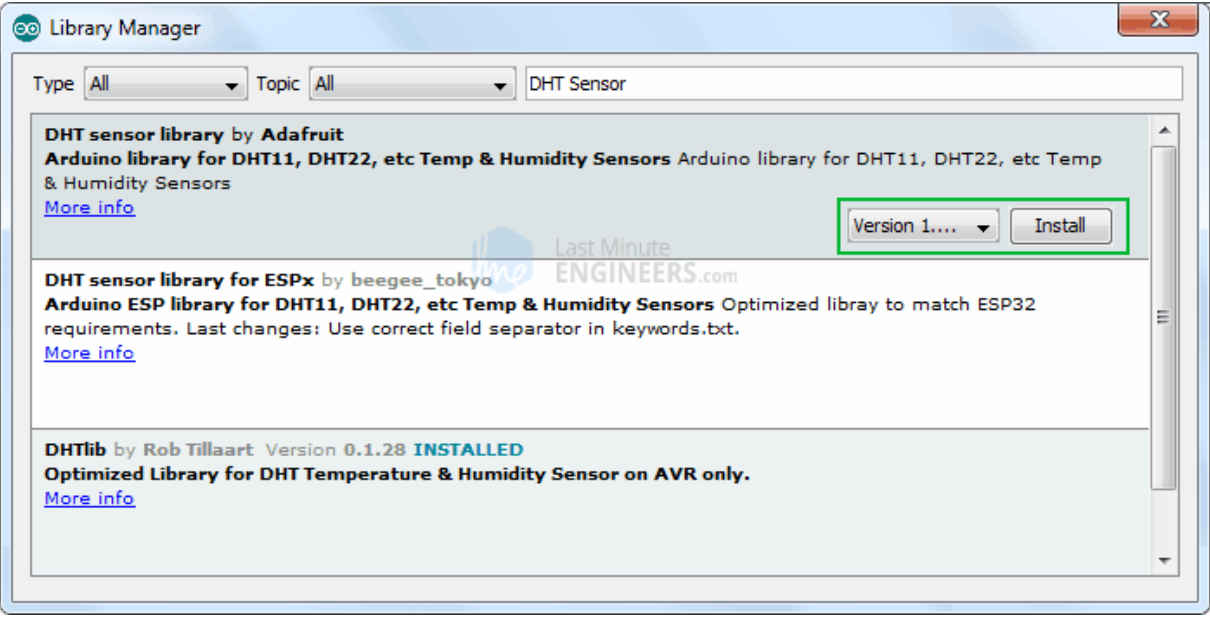


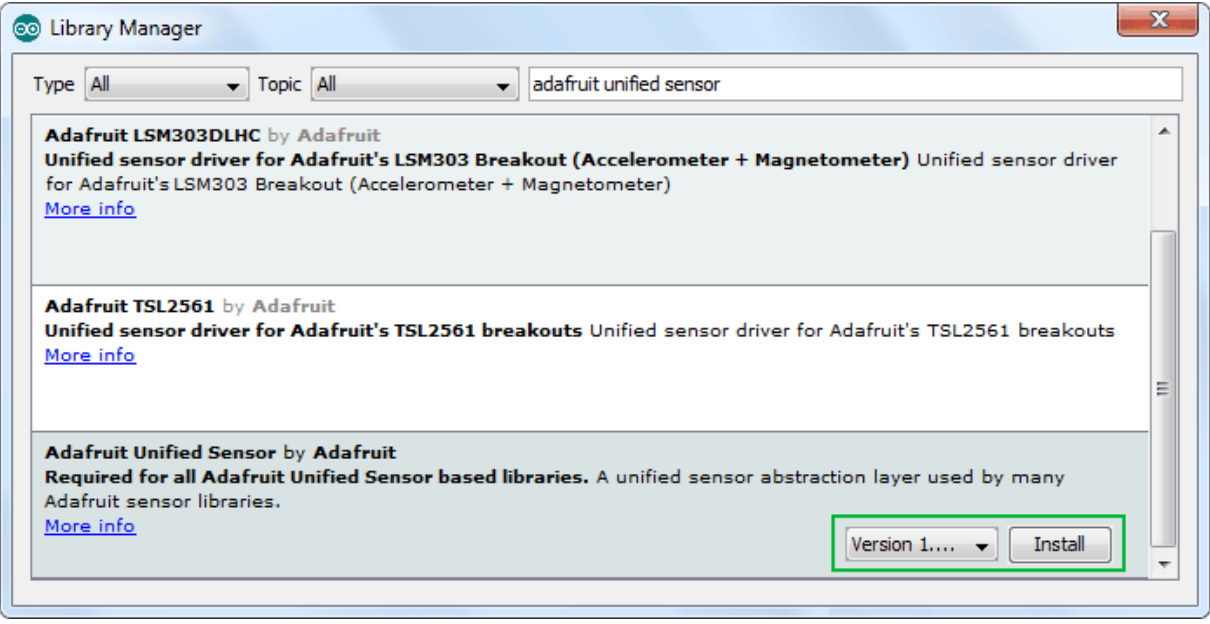



EN
IO0
ESP-WROOM-32
fritzing

Installing DHT Sensor Library


sketch_aug07a | Arduino 1.8.5
File Edit Sketch Tools Help
Verify/Compile Ctrl+R
Upload Ctrl+U
Upload Using Programmer Ctrl+Shift+U
Export compiled Binary Ctrl+Alt+S
Show Sketch Folder Ctrl+K
Include Library
Add File...
Manage Libraries...
Add .ZIP Library...
Arduino libraries
Bridge
Esplora
Ethernet
Firmata
Keyboard
LiquidCrystal
Mouse
Robot Control
// put your main code here, to run repeate
Last Minute ENGINEERS.com


Library Manager
Type All
Topic All
DHT Sensor
DHT sensor library by Adafruit
Arduino library for DHT11, DHT22, etc Temp & Humidity Sensors Arduino library for DHT11, DHT22, etc Temp & Humidity Sensors
More info
Version 1....
Install
Last Minute ENGINEERS.com
DHT sensor library for ESPx by beegee_tokyo
Arduino ESP library for DHT11, DHT22, etc Temp & Humidity Sensors Optimized libray to match ESP32 requirements. Last changes: Use correct field separator in keywords.txt.
More info
DHTlib by Rob Tillaart Version 0.1.28 INSTALLED
Optimized Library for DHT Temperature & Humidity Sensor on AVR only.
More info


Library Manager
Type All
Topic All
adafruit unified sensor
Adafruit LSM303DLHC by Adafruit
Unified sensor driver for Adafruit's LSM303 Breakout (Accelerometer + Magnetometer) Unified sensor driver for Adafruit's LSM303 Breakout (Accelerometer + Magnetometer)
More info
Adafruit TSL2561 by Adafruit
Unified sensor driver for Adafruit's TSL2561 breakouts Unified sensor driver for Adafruit's TSL2561 breakouts
More info
Adafruit Unified Sensor by Adafruit
Required for all Adafruit Unified Sensor based libraries. A unified sensor abstraction layer used by many Adafruit sensor libraries.
More info
Version 1....
Install

Funciones en Arduino

Funciones en Arduino

Definición

Las funciones son bloques de código reutilizable.

- Ayudan a dividir un programa en tareas más pequeñas y manejables.
- Ayudan a mantener un código más ordenado y legible.
- Son esenciales para la programación estructurada en Arduino.
- Se pueden crear funciones personalizadas o utilizar las funciones predefinidas de la librería de Arduino.

Características de las funciones en Arduino

- Deben tener un nombre descriptivo.
- Pueden tener parámetros que se utilizan para pasar información a la función.
- Pueden devolver valores utilizando la palabra clave return".
- Pueden ser llamadas desde cualquier parte del código.
- Se definen fuera de la función "void setupz antes de la función "void loop".

Aplicaciones de las funciones en Arduino

- Lectura de sensores: se pueden crear funciones para leer diferentes tipos de sensores como temperatura, humedad, luz, etc.
- Control de actuadores: se pueden crear funciones para controlar motores, luces, relés, etc.
- Realización de cálculos complejos: se pueden crear funciones para realizar cálculos matemáticos o conversiones de unidades.
- Comunicación con otros dispositivos: se pueden crear funciones para enviar y recibir datos a través de diferentes protocolos como I2C, SPI, Bluetooth, etc.

Tipos de funciones en Arduino

1. Funciones de retorno de valor
2. Funciones sin retorno de valor
3. Funciones con parámetros
4. Funciones con valores predeterminados

Funciones de retorno de valor en Arduino

Definición

Las funciones de retorno de valor en Arduino son aquellas que devuelven un valor específico después de que se ejecuta el bloque de código. Estas funciones se definen con el tipo de dato que se va a devolver y deben utilizar la palabra clave return"para devolver el valor.

- Las funciones de retorno de valor pueden recibir parámetros para utilizar en el bloque de código.
- El tipo de dato que se devuelve puede ser cualquiera que se pueda manejar en Arduino (enteros, flotantes, caracteres, etc.).
- El valor que se devuelve debe ser compatible con el tipo de dato que se define en la definición de la función.

Funciones de retorno de valor en Arduino

```
int sumar(int a, int b) {
    int resultado = a + b;
    return resultado;
}

int resultadoSuma = sumar(2, 3);
```

Funciones sin retorno de valor en Arduino

Definición

Las funciones sin retorno de valor en Arduino son aquellas que no devuelven ningún valor específico después de que se ejecuta el bloque de código. Estas funciones se definen con el tipo de dato "voidz no utilizan la palabra clave return".

- Las funciones sin retorno de valor también pueden recibir parámetros para utilizar en el bloque de código.
- Estas funciones suelen utilizarse para realizar una tarea específica en el programa sin necesidad de devolver un valor.
- Al no devolver un valor, estas funciones no pueden ser utilizadas en una expresión o asignación de variable.

Funciones sin retorno de valor en Arduino

```
void prenderLed(int pin) {
    digitalWrite(pin, HIGH);
}

prenderLed(13);
```

Funciones en Arduino: Funciones en bloque

Definición

Las funciones en bloque son un tipo de función que permite agrupar varias instrucciones juntas para realizar una tarea específica.

- Pueden tener parámetros que se utilizan para pasar información a la función.
- No devuelven valores utilizando la palabra clave return".
- Pueden ser llamadas desde cualquier parte del código.

Funciones en Arduino: Funciones en bloque

```
void encenderLed(int pinLed) {
    digitalWrite(pinLed, HIGH);
    delay(1000);
    digitalWrite(pinLed, LOW);
    delay(1000);
}
```

Funciones en Arduino: Funciones que devuelven valores

Definición

Las funciones que devuelven valores son un tipo de función que permite realizar cálculos y devolver un valor como resultado de la función.

- Tienen un tipo de dato de retorno que indica el tipo de valor que devuelve la función.
- Pueden tener parámetros que se utilizan para pasar información a la función.
- Devuelven valores utilizando la palabra clave return".
- Pueden ser llamadas desde cualquier parte del código.

Funciones en Arduino: Funciones que devuelven valores

```
int sumar(int num1, int num2) {
    int resultado = num1 + num2;
    return resultado;
}
```

Funciones con parámetros por referencia en Arduino

Definición

Las funciones con parámetros por referencia en Arduino son aquellas que utilizan la dirección de memoria de una variable existente en el programa como parámetro de entrada. Esto permite que los cambios realizados en la función afecten a la variable original fuera de la función.

- Para indicar que un parámetro es por referencia, se utiliza el símbolo "&.ªntes del nombre del parámetro en la definición de la función.
- Estas funciones permiten reducir el uso de memoria en el programa al evitar la creación de nuevas variables en la función.
- Es importante tener en cuenta que al utilizar parámetros por referencia, se deben tomar precauciones para evitar cambios no deseados en la variable original.

Funciones con parámetros por referencia en Arduino

```
void incrementar(int &x) {
    x++;
}

int numero = 5;
incrementar(numero);
```

Funciones en Arduino: Funciones con parámetros por referencia

Definición

Las funciones con parámetros por referencia son un tipo de función que permite modificar los valores de una variable pasada como parámetro dentro de la función.

- Los parámetros se definen con el símbolo "&.ª final del tipo de dato.
- Los parámetros se pasan sin el símbolo "&.ª llamar la función.
- No devuelven valores utilizando la palabra clave return".
- Pueden ser llamadas desde cualquier parte del código.

Funciones en Arduino: Funciones con parámetros por referencia

```
void duplicar(int& num) {
    num = num * 2;
}
```

Funciones en Arduino: Funciones con parámetros por valor

Definición

Las funciones con parámetros por valor son un tipo de función que permite utilizar el valor de una variable pasada como parámetro dentro de la función, sin modificar la variable original.

Ejemplo

Resumen

En este ejemplo se utilizan tres funciones diferentes:

- Una función para encender y apagar un LED.
- Una función para leer el valor del sensor LDR utilizando una función de retorno de valor.
- Una función para leer los valores de temperatura y humedad del sensor DHT11 utilizando parámetros con valores predeterminados.

Cada función tiene valores predeterminados para algunos de sus parámetros, lo que permite su uso sin necesidad de proporcionar valores para todos los parámetros. Además, se utilizan funciones de retorno de valor (en el caso de la función para leer el sensor LDR) y funciones void (en el caso de las otras dos funciones). Los valores de los sensores y el estado del LED se imprimen por comunicación serial para su posterior análisis o visualización.

Ejemplos de código para Arduino

```
// Ejemplo de función para encender y apagar un LED
void parpadearLed(int pin = 13, int tiempoEncendido = 500, int tiempoApagado =
    500) {
    digitalWrite(pin, HIGH);
    delay(tiempoEncendido);
    digitalWrite(pin, LOW);
    delay(tiempoApagado);
}

// Ejemplo de función para leer el sensor LDR
int leerLDR(int pin) {
    int lectura = analogRead(pin);
    return lectura;
}
```

Ejemplos de código para Arduino

```
// Ejemplo de función para leer el sensor DHT11
void leerDHT11(int pin = 2, int tiempoEspera = 2000) {
   DHT dht(pin, DHT11);
   delay(tiempoEspera);
   float temperatura = dht.readTemperature();
   float humedad = dht.readHumidity();
   Serial.print("Temperatura: ");
   Serial.print(temperatura);
   Serial.print(" grados Celsius, Humedad: ");
   Serial.print(humedad);
   Serial.println(" %");
}

void setup() {
   pinMode(13, OUTPUT); // configurar el pin 13 como salida
Serial.begin(115200); // iniciar comunicación serial
}
```

Ejemplos de código para Arduino

```
void loop() {
    // Ejecutar la función para encender y apagar el LED
    parpadearLed();

    // Leer el valor del sensor LDR y mostrarlo por serial
    int valorLDR = leerLDR(A0);
    Serial.print("Valor LDR: ");
    Serial.println(valorLDR);

    // Leer los valores del sensor DHT11 y mostrarlos por serial
    leerDHT11();
}
```

Lectura de un sensor LDR mediante una función

```
int leerLDR(int pinLDR) {
  int valorLDR = analogRead(pinLDR);
  return valorLDR;
}

int pinSensorLDR = A0;
int valorSensorLDR = leerLDR(pinSensorLDR);

void setup() {
  // Configuración del pin del sensor LDR
  pinMode(pinSensorLDR, INPUT);
}

void loop() {
  // Realizar acciones con el valor del sensor LDR
}
```

Interrupciones

Introducción a las interrupciones

Interrupciones en sistemas embebidos

- Los eventos en tiempo real son de suma importancia para las aplicaciones de Internet de las Cosas (IoT).
- En sistemas embebidos, las interrupciones son una **técnica importante** para manejar eventos en **tiempo real**.
- Las interrupciones se utilizan en sistemas embebidos para manejar eventos en tiempo real y mejorar la eficiencia y el rendimiento del sistema.
- Esto resulta en un mejor rendimiento del sistema y un menor consumo de recursos, lo que es especialmente importante en el Internet de las Cosas (IoT).

Ejemplo de aplicación

- A menudo en un proyecto necesitas el ESP32 para ejecutar tu programa normal, mientras que continuamente necesitas que se monitorize para algún tipo de evento
  - Una solución ampliamente adoptada es el uso de una interrupción.
- El ESP32 ofrece hasta 32 pines de interrupción para cada núcleo.

Interrupciones

Definición de interrupción

Definición de interrupción

Una interrupción es un **mecanismo** que permite a un dispositivo externo o evento **interrumpir** temporalmente el flujo normal de ejecución del programa en un microcontrolador.

Definición de interrupción

Las interrupciones:

- Permiten que un microcontrolador **suspenda temporalmente** la ejecución del programa principal y atienda un evento específico.
- Permiten a los microcontroladores responder a eventos sin tener que que sondear continuamente para ver si algo ha cambiado.
- Cuando se produce una interrupción, el microcontrolador pausa el programa principal y ejecuta una rutina de interrupción específica.

Ventajas de utilizar interrupciones en sistemas embebidos

Ventajas de utilizar interrupciones en sistemas embebidos

Las interrupciones:

- Permiten un manejo más rápido y eficiente de eventos en tiempo real, como:
  - Lectura de sensores
  - Recepción de datos
  - Ejecución de tareas críticas
- El uso de condicionales en el bucle principal para detectar eventos es común, pero esto podría ocasionar que se pierdan eventos muy rápidos o críticos si el microcontrolador está realizando otras tareas.

Ventajas de utilizar interrupciones en sistemas embebidos

- Pueden reducir la complejidad del programa principal
  - **Evitan** la necesidad de revisar constantemente el estado de los dispositivos periféricos.

Ventajas de utilizar interrupciones en sistemas embebidos

- El uso adecuado de interrupciones puede mejorar la confiabilidad y la seguridad del sistema: garantizan que los eventos críticos sean atendidos inmediatamente.

Ahorro de recursos del microcontrolador

Permiten que el microcontrolador:

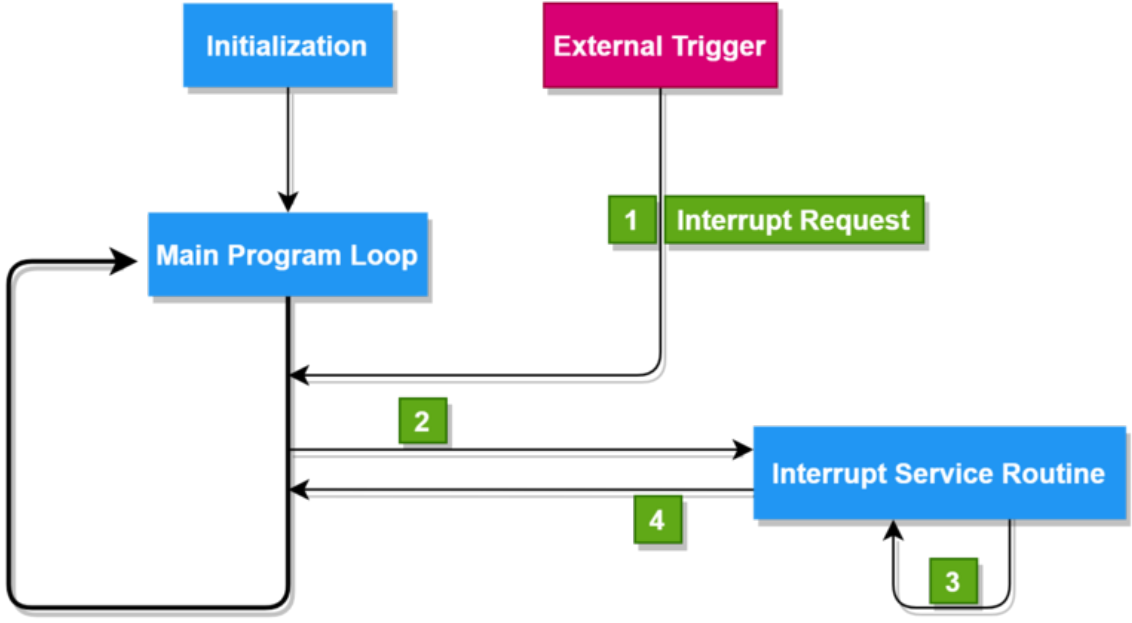
- Se concentre en las tareas principales, en lugar de estar constantemente buscando eventos externos.
- Pueda ahorrar recursos como la memoria y el tiempo de procesamiento para las tareas principales: aumenta la eficiencia general del sistema.
- Ayuda a evitar el consumo excesivo de energía, lo que es especialmente importante en aplicaciones con baterías.
- Responda inmediatamente a eventos externos, lo que es crucial en **aplicaciones en tiempo real**: control de motores, sensores de movimiento, etc.

Rutina de interrupción

Rutina de interrupción

1. Cuando se activa una interrupción, el microcontrolador **detiene temporalmente** la ejecución del **programa principal** y salta a una función llamada **rutina de interrupción**.
2. La **rutina de interrupción** es un conjunto de instrucciones específicas que manejan el evento que generó la interrupción.
3. Después de que se ha **completado** la rutina de interrupción, el microcontrolador vuelve a la ejecución del **programa principal** desde donde se detuvo.


Main program
Interrupt
Interrupt handling routine
End of handling routine
Main program


Initialization
External Trigger
1
Interrupt Request
Main Program Loop
2
Interrupt Service Routine
4
3

Continuidad de la ejecución del programa principal

- Las interrupciones no detienen la ejecución del programa principal, solo lo **suspenden temporalmente**.
- El programa principal continúa su ejecución desde **donde se detuvo** después de que se ha completado la rutina de interrupción.
- Es posible que ocurran varias interrupciones mientras el programa principal se está ejecutando.

Características de las interrupciones

Características de las interrupciones

- Tienen un **nivel de prioridad**: las interrupciones de mayor prioridad se atienden antes que las de menor prioridad.
- Las interrupciones pueden ser habilitadas o deshabilitadas en cualquier momento durante la ejecución del programa.
  - La función `cli()` *(Clear Interrupts)* deshabilita todas las interrupciones.
  - `cli()` garantiza que la ejecución del programa principal no se vea interrumpida por interrupciones no deseadas
- **Tener cuidado:** una mala implementación puede causar problemas de estabilidad y rendimiento.

Clases de interrupciones

Clases de interrupciones en Arduino

Existen dos clases de interrupciones:

- Interrupciones externas (*hardware*)
- Temporizadores (*software*)

Interrupciones externas

Interrupciones externas

Interrupciones Externas

Ocurren en respuesta a un evento externo. Señales eléctricas generadas por **hardware externo** al microcontrolador activan la interrupción.

- **Ejemplo:** Una interrupción de GPIO (cuando se pulsa una tecla) o una interrupción de toque o pulsación (cuando se detecta el pulsar)

Características de las Interrupciones Externas

- Se activan cuando se detecta un cambio en el estado de una entrada, independientemente de lo que el microcontrolador esté haciendo.
- La interrupción es generada por una señal eléctrica externa (*trigger*).
- Son generadas por dispositivos como botones, sensores, entre otros.
- El microcontrolador mantiene **monitoreo** de la señal de interrupción, mientras se ejecuta el programa principal.
- Cuando se genera la señal externa, el controlador detiene la ejecución del programa principal y comienza la ejecución de la rutina de interrupción.
- Las interrupciones de hardware son capaces de detectar eventos muy rápidos, como pulsos de sensores que solo duran unos pocos microsegundos.

Casos de uo de Interrupciones Externas

- **Contadores de eventos:** Se puede utilizar interrupciones para contar el número de veces que un evento ha ocurrido, como por ejemplo el número de veces que se ha pulsado un botón.
- **Sistemas de alarma:** Las interrupciones externas pueden ser utilizadas para activar sistemas de alarma, por ejemplo cuando una puerta se abre o se cierra.
- **Control de motores:** En sistemas de control de motores, se pueden utilizar interrupciones externas para indicar cuándo un motor ha alcanzado una determinada posición.

Cómo utilizar interrupciones en Arduino

Cómo utilizar interrupciones en Arduino

- En el Arduino Uno, solo se pueden usar dos pines como interrupciones de hardware, lo que limita su uso.
- El Arduino Leonardo tiene cuatro pines interrupciones-capaces, mientras que el Mega2560 tiene muchos más, y el Due permite interrupciones en todos sus pines.
- Si se utiliza un microcontrolador compatible con Arduino, se debe verificar la documentación del núcleo de Arduino para conocer qué pines están disponibles para las interrupciones.
- La biblioteca del núcleo de Arduino para ESP8266/ESP32 es muy buena para las interrupciones, ya que permite interrupciones en cualquier pin y utiliza el nombre del pin en lugar del número de interrupción.

Cómo utilizar interrupciones en Arduino

- Se deben definir funciones que se ejecuten cuando se detecta una interrupción.
- Estas funciones se deben registrar con la función `attachInterrupt()`, que toma tres argumentos:
  - El número de pin
  - La función que se ejecutará
  - El modo de activación de la interrupción.

Cómo utilizar interrupciones en Arduino

El modo de activación de la interrupción puede ser `CHANGE`, `RISING` o `FALLING`, que especifica si la interrupción debe activarse en **cualquier** cambio, en un cambio **ascendente** o en un cambio **descendente**.

Modos de interrupción

Modo	Operación	Descripción
LOW	Interrumpe cuando el pin está en nivel BAJO	Este modo hace que la ISR se ejecute continuamente mientras el pin esté en nivel BAJO.
RISING	Interrumpe cuando el pin cambia de nivel BAJO a nivel ALTO	
FALLING	Interrumpe cuando el pin cambia de nivel ALTO a nivel BAJO	
CHANGE	Interrumpe cuando el pin cambia de nivel en cualquier dirección	
HIGH	Interrumpe cuando el pin está en nivel ALTO	Este modo solo está disponible en el Due y, al igual que LOW, se usa raramente.

Cómo utilizar interrupciones en Arduino

- **Importante:** las funciones que se ejecutan como parte de una interrupción deben ser **muy breves y no deben bloquear el flujo del programa**.
- Si se necesita realizar operaciones más complejas dentro de una interrupción, se puede utilizar una variable global para indicar que se ha producido una interrupción, y luego procesar esa variable en el bucle principal.

Ejemplo de interrupción en Arduino

- Se muestra cómo funcionan las interrupciones de hardware.
- Se requiere un protoboard, un botón táctil, una resistencia de $1mega$ y cables de puente.
- La resistencia mantiene el pin de interrupción (D2) en **ALTO** hasta que se presiona el botón, momento en el que D2 se conecta a tierra y se va a **BAJO**.

SCL
SDA
AREF
GND
13 12 11
L
TX
RX
PWM
PWM
PWM
DIGITAL
IOREF
RESET
3V3
POWER
Gnd Vin
www.arduino.cc
Arduino
UNO
PWM
PWM
PWM
TX
RX
ON
ICSP
ANALOG IN
0 1 2 3 4 5

Ejemplo de interrupción en Arduino

```
const int ledPin = 13;
const int interruptPin = 2;

void setup()
{
   pinMode(ledPin, OUTPUT);
   attachInterrupt(digitalPinToInterrupt(interruptPin), stuffHapenned, FALLING);
}

void loop()
{
}

void stuffHappened()
{
   digitalWrite(ledPin, HIGH);
}
```

Ejemplo de interrupción en Arduino

- La función de configuración establece el pin del LED como una salida y asocia una función con una interrupción.
- La función `stuffHappened` se ejecuta cada vez que ocurre la interrupción.
- Los nombres de los pines de interrupción no corresponden con los nombres de los pines en una placa Arduino.
- Para obtener el número de interrupción correcto para un pin, utiliza la función `digitalPinToInterrupt(pin)`.

Ejemplo de interrupción en Arduino

- El segundo argumento de `attachInterrupt` es el nombre de la función que se ejecutará al ocurrir la interrupción.
- Esta función se llama **rutina de servicio de interrupción (ISR)**.
- Las **ISRs** no pueden tener parámetros ni devolver valores.
- El tercer argumento de `attachInterrupt` es una constante que indica el tipo de interrupción.

Condiciones para el uso de ISR (Interrupt Service Routines)

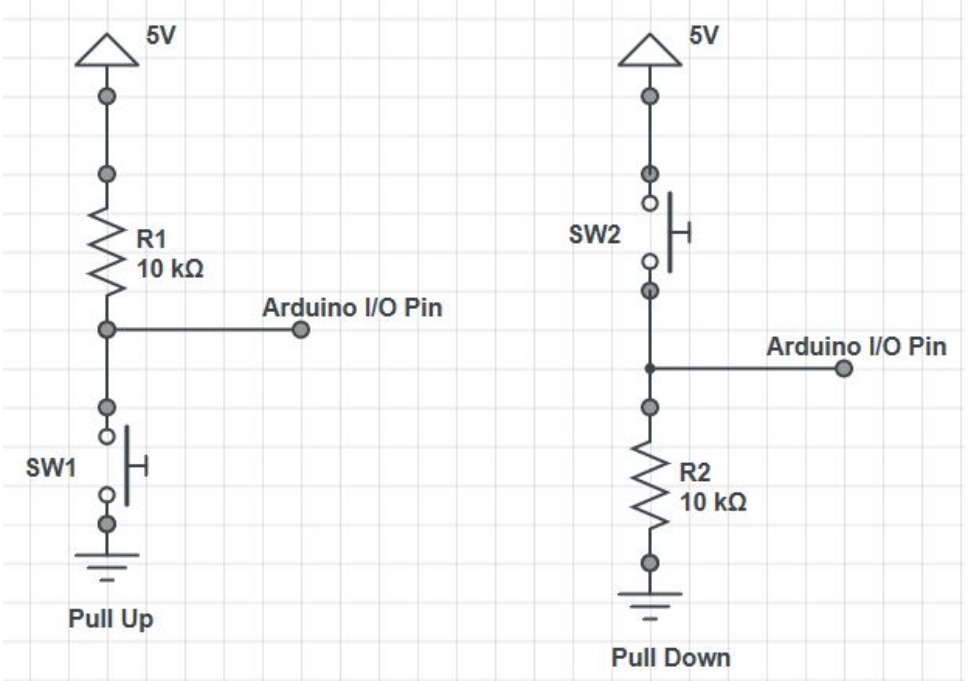
- ISR debe ser lo más corta y rápida posible.
- Si se produce otra interrupción mientras se está ejecutando la ISR, ésta no será interrumpida y la señal de interrupción se ignorará hasta que la ISR haya terminado.
- Nada sucede con el código en la función `loop` mientras se ejecuta una ISR.
- Las interrupciones se desactivan automáticamente dentro de una ISR.

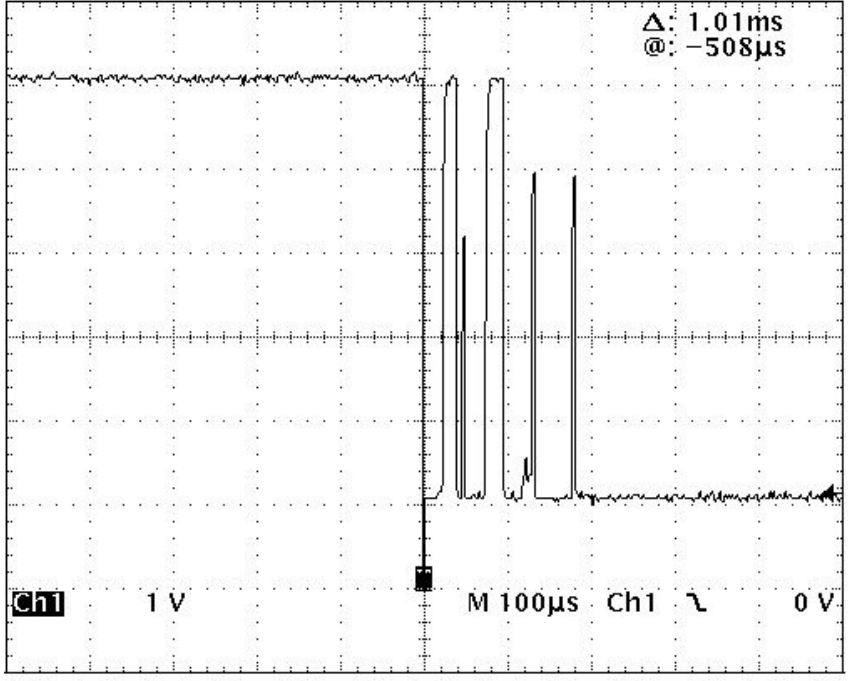
Ejemplo de interrupción en Arduino

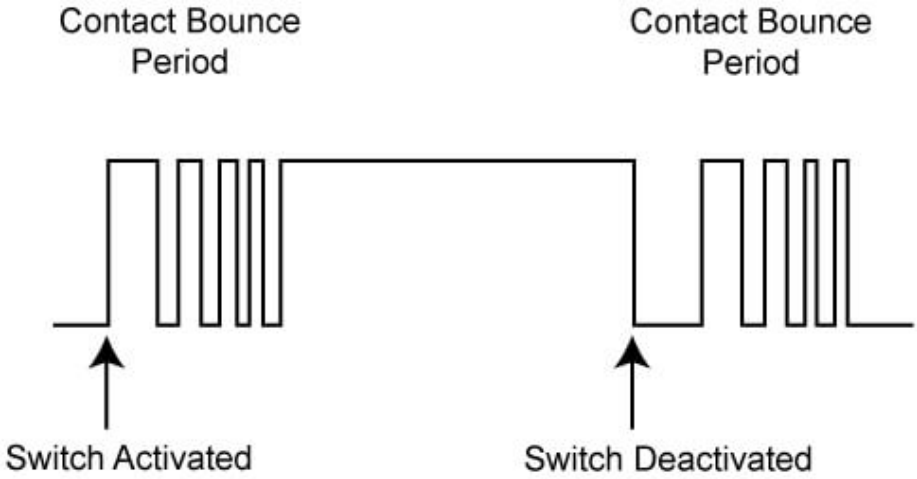
- El *loop* del *sketch* está vacío porque la **ISR** se encarga de encender el LED.
- Si cambias el tercer argumento de attachInterrupt a RISING, la ISR se ejecutará cuando D2 cambie de BAJO a ALTO.
- Si mantienes presionado el botón, la ISR no se ejecutará hasta que lo sueltes porque D2 solo cambia de ALTO a BAJO cuando lo sueltas.

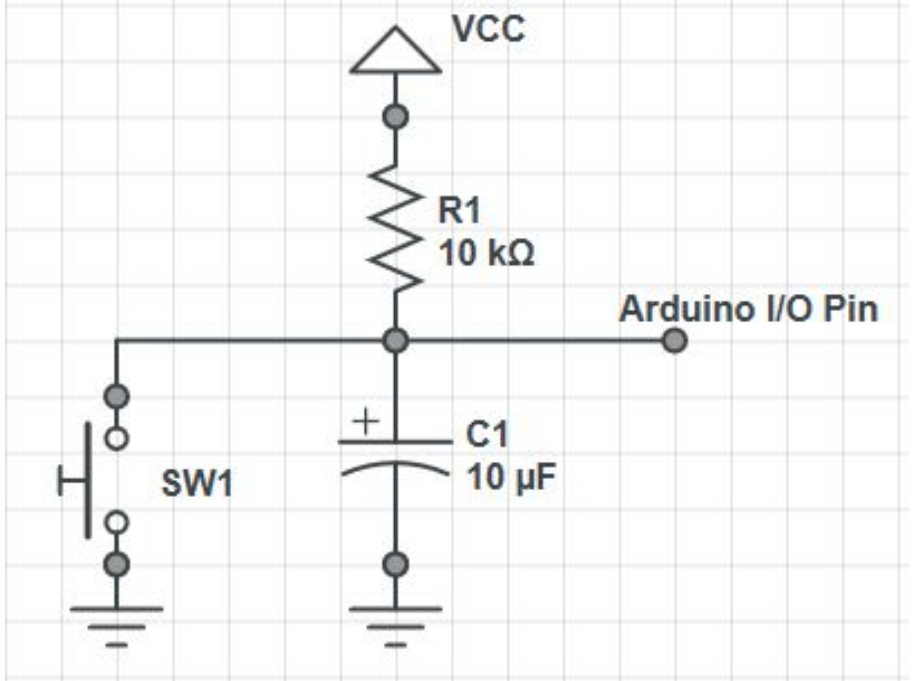
Ejemplo de interrupción en Arduino

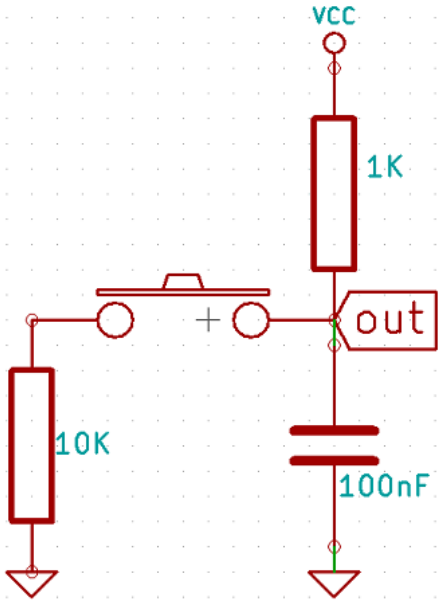
- Si el botón no funciona correctamente, puede ser debido al **efecto rebote**.
- Para probar si hay efecto rebote, mantén presionado el botón mientras presionas el botón de reset en el Arduino.
- Cuando sueltes el botón de prueba, el LED se encenderá.


5V
R1
10 kΩ
Arduino I/O Pin
SW1
Pull Up
5V
SW2
Arduino I/O Pin
R2
10 kΩ
Pull Down


Δ: 1.01ms
@: −508µs
Ch1 1 V
M 100µs Ch1 ↘ 0 V


Contact Bounce Period
Contact Bounce Period
Switch Activated
Switch Deactivated


VCC
R1
10 kΩ
Arduino I/O Pin
SW1
C1
10 µF


VCC
1K
out
10K
100nF

Pines de interrupción de diferentes placas Arduino

	Número de la interrupción						
Placa	**0**	**1**	**2**	**3**	**4**	**5**	**Notas**
Uno/Pro Mini	D2	D3	–	–	–	–	
Leonardo	D3	D2	D0	D1	–	–	Sí, los números de las interrupcione son opuestos a los del Uno.
Mega2560	D2	D3	D21	D20	D19	D18	
Due/ESP32/ESP8266							Se utilizan números de pin en lugar de los números de interrupción.

```
attachInterrupt(GPIOPin, ISR, Mode);
```

Esta función toma tres parámetros:

- **GPIOPin:** Establece la clavija GPIO como una clavija de interrupción, que le dice al ESP32 qué clavija debe monitorear.
- **ISR**: Es el nombre de la función que se llamará cada vez que se dispare la interrupción.
- **Mode**: Define cuándo se debe disparar la interrupción. Cinco constantes están predefinidas como valores válidos:

LOW	Los disparadores interrumpen cuando el pin está LOW
HIGH	Los disparadores interrumpen cuando el pin es HIGH
CHANGE	Los disparadores interrumpen cuando el pin cambia de valor, de HIGH a LOW o LOW a HIGH
FALLING	Los disparadores interrumpen cuando el pin va de HIGH a LOW
RISING	Los disparadores interrumpen cuando el pin va de LOW a HIGH

Ejemplo de Código para la Interrupción Utilizando un Sensor

```
const int SENSOR_PIN = A0;
volatile int sensorValue = 0;

void setup() {
   pinMode(SENSOR_PIN, INPUT);
   attachInterrupt(digitalPinToInterrupt(SENSOR_PIN), sensorISR, CHANGE);
}

void loop(){
   // Realiza alguna accion basada en el valor del sensor
}

void sensorISR(){
   sensorValue = analogRead(SENSOR_PIN);
}
```

Ejemplo de interrupción en Arduino que utiliza las funciones cli() y sei()

```
const byte interruptPin = 2;
volatile boolean state = LOW;

void setup() {
    pinMode(interruptPin, INPUT_PULLUP);
    attachInterrupt(digitalPinToInterrupt(interruptPin), interruptFunction, RISING
        );
    Serial.begin(9600);
}

void loop() {
    Serial.println(state);
    delay(1000);
}

void interruptFunction() {
    cli();
    state = !state;
    sei();
}
```

Pasos para hacer una interrupción en Arduino

- Utiliza una interrupción externa para detectar el flanco de subida en un pin digital determinado
- La función `cli()` (*Clear Enable Interrupts*, Limpiar de Interrupciones) se utiliza para deshabilitar temporalmente todas las interrupciones del microcontrolador mientras se actualiza la variable `state`
- Asegura que no se produzcan interrupciones adicionales mientras se está en medio de una interrupción
- Una vez que se actualiza `state`, la función `sei()` (*Set Enable Interrupts*, Habilitar Interrupciones) se utiliza para habilitar nuevamente las interrupciones
- Esto asegura que las interrupciones externas que puedan ocurrir durante la interrupción actual no se pierdan.

Interrupciones de software

Interrupciones de software

Interrupciones de software

Las interrupciones de software son aquellas que son generadas por el propio programa en ejecución. Estas interrupciones permiten que una tarea de baja prioridad sea interrumpida por una tarea de alta prioridad.

Interrupciones de software

Las interrupciones de software son especialmente útiles cuando:

- Se trabaja con tareas en segundo plano
- Se requiere una rápida respuesta a un evento específico.
- Estas ocurren en respuesta a una instrucción de software. Por ejemplo, una simple interrupción de temporizador o una interrupción de temporizador de vigilancia (cuando el temporizador se agota)

Características de las interrupciones de software

- Son activadas por **instrucciones** en el programa, en lugar de por una señal externa.
- Se implementan utilizando una función de **interrupción personalizada**, que es llamada por el programa principal cuando se requiere una interrupción.
- **No son inmediatas**, sino que se activan en el momento en que se ejecuta la instrucción de activación de interrupción en el programa principal.

Ejemplos de uso de interrupciones de software

- Implementación de **temporizadores** para *tareas en segundo plano*, como la medición de temperatura y humedad.
- Control de servomotores para la automatización de procesos.
- Lectura de sensores que requieren una **respuesta rápida**, como los sensores de proximidad.
- Implementación de un sistema de seguridad que requiere una respuesta inmediata a eventos específicos.

Pasos para hacer una interrupción temporal en Arduino

1. Selecciona el temporizador adecuado para tu proyecto: Arduino tiene varios temporizadores que se pueden utilizar para diferentes tareas.
2. **Configura el modo del temporizador:** Esto determinará la forma en que el temporizador funcionará, como el modo normal o el modo PWM.
3. **Establece el preescalador del temporizador:** Esto ajustará la frecuencia del temporizador, lo que afectará la precisión del temporizador.
4. **Establece el valor inicial del contador:** Esto determinará el valor inicial del temporizador y se puede ajustar según la necesidad del proyecto.
5. **Habilita la interrupción del temporizador si es necesario:** Esto permitirá que el procesador de Arduino sepa cuando el temporizador ha alcanzado el valor deseado y ejecutar la interrupción correspondiente.

Pasos para hacer una interrupción temporal en Arduino

7. **Configura el control de comparación de salida (OC) si se utiliza el modo PWM:** Esto permitirá que el temporizador genere una señal PWM en los pines adecuados.
8. **Configura el registro de control de salida (OCR):** Esto determinará el valor de comparación para la salida del temporizador.
9. **Inicia el temporizador:** Esto permitirá que el temporizador comience a contar y a realizar su función según su configuración.
10. **Verifica el valor del temporizador cuando sea necesario:** Esto permitirá que se realicen acciones específicas cuando el temporizador alcance ciertos valores.
11. **Detén el temporizador cuando sea necesario:** Esto permitirá que se detenga la función del temporizador y se conserve la energía y recursos de Arduino.

Temporizadores y retardos en Arduino

Temporizadores y retardos en Arduino

```
int led = 13;
bool ledOn;

void setup() {
   Serial.begin(115200);
   pinMode(led, OUTPUT);   // initialize the digital pin as an output.
   digitalWrite(led, HIGH); // turn led on
   ledOn = true; // led is on
}

void loop() {
    if (ledOn) {
        delay(10000);
        digitalWrite(led, LOW); // turn led off
        ledOn = false;  // prevent this code being run more then once
        Serial.println("Turned LED Off");
    }
    //  Other loop code here . . .
    Serial.println("Run Other Code");
}
```

Código en Arduino

```
int led = 13;
unsigned long DELAY_TIME = 10000;
unsigned long delayStart = 0;
bool delayRunning = false;

void setup() {
    pinMode(led, OUTPUT);
    digitalWrite(led, HIGH);
    delayStart = millis();
    delayRunning = true;
}

void loop() {
    if (delayRunning && ((millis() - delayStart) >= DELAY_TIME)) {
        delayRunning = false;
        digitalWrite(led, LOW);
    }
}
```

Conclusión

Conclusión

- Las interrupciones de software son una herramienta importante en la programación de sistemas embebidos, permitiendo la ejecución de tareas de alta prioridad en respuesta a eventos específicos.
- Los programadores de IoT deben conocer las características y ejemplos de uso de interrupciones de software para aprovechar al máximo las capacidades de sus dispositivos.

Conclusión

- Las interrupciones en Arduino pueden ser activadas por hardware o software.
- Las interrupciones por hardware son generadas por señales externas, como un cambio de nivel en un pin, mientras que las interrupciones por software son generadas por instrucciones específicas del programa.
- Las interrupciones por hardware son más rápidas y eficientes que las interrupciones por software, pero son menos flexibles.

¡Muchas gracias por su atención!

¿Preguntas?

Contacto: Marco Teran
webpage: marcoteran.github.io/
e-mail: marco.teran@usa.edu.co

UNIVERSIDAD
SERGIO ARBOLEDA