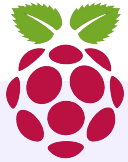


Introducción a la programación de Raspberry Pi con Python

Internet de las Cosas con Raspberry Pi y Python

Hecho por Eric Alexander

https://youtube.com/c/SoloPython

Resumen del Curso

Semana 1: Introducción a Raspberry Pi

Módulo 1.1: ¿Qué es Raspberry Pi?

Módulo 1.2: Componentes de la Raspberry Pi

Módulo 1.3: Sistemas operativos para Raspberry Pi

Módulo 1.4: Instalación y configuración de Raspberry Pi

Módulo 1.5: Conexión de Raspberry Pi a Wi-Fi

Semana 2: Introducción a Python

Módulo 2.1: Conceptos básicos de Python: Variables, tipos de datos y operadores

Módulo 2.2: Estructuras de Control: Bucles y condicionales

Módulo 2.3: Funciones y módulos en Python

Módulo 2.4: Introducción a las bibliotecas de Python

Módulo 2.5: Depuración y tratamiento de errores en Python

Semana 3: Python en Raspberry Pi

Módulo 3.1: Entorno de desarrollo integrado (IDE) de Python en Raspberry Pi

Módulo 3.2: Ejecución de scripts Python en Raspberry Pi

Módulo 3.3: Pines GPIO en Raspberry Pi

Módulo 3.4: Interfaz con Hardware usando Python

Módulo 3.5: Proyecto: Construcción de un simple programa de LED parpadeante

Semana 5: Redes y Raspberry Pi

Módulo 5.1: Fundamentos de la conexión en red

Módulo 5.2: Redes con Python

Módulo 5.3: Introducción a IoT (Internet de las cosas)

Módulo 5.4: Conexión de Raspberry Pi a Internet y Red Local

Módulo 5.5: Proyecto: Control Remoto de LED

Semana 4: Recopilación y Procesamiento de Datos

Módulo 4.1: Recopilación de datos de sensores con Python

Módulo 4.2: Introducción a las API

Módulo 4.3: Almacenamiento y recuperación de datos con Python

Módulo 4.4: Procesamiento y análisis de datos

Módulo 4.5: Proyecto: Registrador de datos

Semana 6: IA en Raspberry Pi

Módulo 6.1: Introducción a la Inteligencia Artificial

Módulo 6.2: Bibliotecas Python para IA

Módulo 6.3: Implementación de algoritmos sencillos de IA

Módulo 6.4: IA con datos de sensores

Módulo 6.5: Proyecto: IA para predecir la temperatura

Cada módulo incluirá una breve introducción, contenido en profundidad, actividades prácticas, cuestionarios y un proyecto al final de la semana para aplicar lo aprendido.

Este curso no asume ningún conocimiento previo de Raspberry Pi o Python. Todas las lecciones están diseñadas para ser fáciles de seguir y ofrecer experiencia práctica con aplicaciones del mundo real. Utilizaremos la última versión de Raspberry Pi y Python disponible en el momento del curso.

Al final de este curso, los estudiantes tendrán una base sólida en la programación de Raspberry Pi y Python y serán capaces de utilizar estas habilidades para construir sus proyectos de IA.

Materiales del Curso

Raspberry Pi

Kit Arduino - Sensores

Computadora

Pantalla HDMI

Cables Jumper

Protoboard

Módulo 1.1: Qué es Raspberry Pi

Raspberry Pi es una serie de pequeños ordenadores monoplaca desarrollados en el Reino Unido por la Fundación Raspberry Pi. Su intención inicial era promover la enseñanza de informática básica en escuelas y países en desarrollo, pero rápidamente se popularizó entre aficionados y entusiastas de la tecnología de todo el mundo.

El Raspberry Pi es un ordenador del tamaño de una tarjeta de crédito que se conecta a un monitor de ordenador o a un televisor y utiliza un teclado y un ratón estándar. Es capaz de hacer todo lo que se espera de un ordenador de sobremesa, desde navegar por Internet y reproducir vídeo de alta definición hasta hacer hojas de cálculo, procesar textos y jugar.

Lo que hace destacar a la Raspberry Pi es su versatilidad. Con una Raspberry Pi, los usuarios pueden explorar la informática y aprender lenguajes de programación como Python y Scratch. Es capaz de interactuar con el mundo exterior y se ha utilizado en una amplia gama de proyectos de creación digital, desde máquinas de música y detectores de padres hasta estaciones meteorológicas y pajareras con cámaras de infrarrojos.

Módulo 1.2: Componentes de un Raspberry Pi

- **CPU (Unidad Central de Procesamiento)**: Este es el cerebro de la Raspberry Pi. La Raspberry Pi 4 Modelo B utiliza un Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC.
- **RAM (Memoria de Acceso Aleatorio)**: Es la memoria a corto plazo donde se ejecutan los procesos activos. La Raspberry Pi 4 viene en tres versiones en cuanto a RAM: 2GB, 4GB y 8GB.
- **Pines GPIO (General Purpose Input Output)**: Estos son los pines a través de los cuales la Raspberry Pi se comunica con el mundo exterior.
- **Puertos USB**: Se utilizan para conectar periféricos como un teclado, un ratón u otros dispositivos USB. La Pi 4 tiene dos puertos USB 2.0 y dos puertos USB 3.0.
- **Puerto Ethernet**: Para la conectividad de red por cable.
- **Wi-Fi y Bluetooth**: Para redes inalámbricas y conectividad de dispositivos.
- **Ranura para tarjetas MicroSD**: Aquí se inserta la tarjeta SD que contiene el sistema operativo y el sistema de archivos de la Raspberry Pi.
- **Puertos HDMI**: La Raspberry Pi 4 tiene dos puertos micro HDMI para la salida de vídeo.
- **Conector de audio**: Este conector de 3,5 mm se puede utilizar para la salida de audio.
- **CSI (Camera Serial Interface) Puerto**: Este puerto es para conectar el módulo de la cámara Raspberry Pi.
- **DSI (Display Serial Interface) Puerto**: Este puerto es para conectar una pantalla a la Raspberry Pi.
- **Entrada de Energía**: La Raspberry Pi 4 utiliza un puerto USB-C para la alimentación.

Módulo 1.3: Sistemas Operativos

Un sistema operativo (SO) es un software de sistema que gestiona los recursos de hardware y software del ordenador y proporciona diversos servicios para los programas informáticos. Para la Raspberry Pi, existen varios sistemas operativos compatibles, pero el más común es Raspberry Pi OS (anteriormente conocido como Raspbian).

- Raspberry Pi OS: Es el sistema operativo oficial soportado por la Fundación Raspberry Pi. Está basado en Debian, optimizado para el hardware de Raspberry Pi.
- Ubuntu: Ubuntu es una opción popular para Raspberry Pi debido a su amplio soporte y facilidad de uso. Ubuntu Server, Ubuntu Core y Ubuntu Mate están disponibles para Raspberry Pi.
- LibreELEC: Si estás buscando convertir tu Raspberry Pi en un centro multimedia, LibreELEC es una buena opción. Es un sistema operativo ligero construido específicamente para ejecutar Kodi, un reproductor multimedia de código abierto.
- Retropie: Retropie es una opción popular para convertir tu Raspberry Pi en una consola de juegos retro.

Estos son sólo algunos ejemplos, y hay muchos otros sistemas operativos disponibles en función de sus necesidades.

Módulo 1.4: Instalando y Configurando RaspberryPi

Estos son los pasos para poner en marcha tu Raspberry Pi:

1) Descarga el Raspberry Pi OS: Puede descargar la última versión de Raspberry Pi OS desde el sitio web de la Fundación Raspberry Pi.
2) Flashea el Raspberry Pi OS en la tarjeta MicroSD: Para ello, necesitarás un software como BalenaEtcher. Elige la imagen del SO que acabas de descargar, selecciona tu tarjeta SD y haz clic en '¡Flash!
3) Inserta la tarjeta MicroSD en la Raspberry Pi: Una vez que el sistema operativo esté flasheado en la tarjeta MicroSD, retírala de tu ordenador e insértala en la ranura para tarjetas MicroSD de la Raspberry Pi.
4) Conecta tus periféricos: Conecte su monitor (a través de HDMI), teclado y ratón (a través de USB) a la Raspberry Pi.

Encienda la Raspberry Pi: Conecta tu fuente de alimentación a la Raspberry Pi y enchúfala a la pared. La Raspberry Pi se iniciará y debería ver el Raspberry Pi OS en la pantalla.

Módulo 1.5: Wi-Fi, SSH y VNC

1) Enciende el raspberry pi y configura el wi-fi acorde como sale en la pantalla cuando primero prendes el dispositivo.

2) Abre el terminal y escribe; sudo raspi-config

3) Habilitar SSH, VNC y Remote GPIO

Semana 2: Introduccion a Python

Python es un lenguaje excelente para principiantes por su facilidad de lectura y su sintaxis compacta. Vamos a sumergirnos en los conceptos básicos.

Crearemos un ambiente virtual, abre un terminal donde deseas trabajar y crea un ambiente virtual:

- Python -m venv env

- Para activar el ambiente escribe:

    - Mac / Linux: source ./env/bin/activate
    - Window: .\env\Scripts\activate

Módulo 2.1: Variables, tipos de datos y operadores

Variables

En Python, las variables se utilizan para almacenar información. Las variables pueden ser consideradas como contenedores que contienen datos que pueden ser cambiados posteriormente a lo largo de la programación. No necesitas declarar el tipo de variable, se infiere dinámicamente del valor que se le asigna.

```
X = 5
Name = “john”
```

En este ejemplo, x es una variable entera que almacena el valor 5 y nombre es una variable de cadena que almacena el valor "Juan".

Tipos de Datos

Python tiene varios tipos de datos estándar:

1) Números: Pueden ser enteros (1 y 2), flotantes (1,1 y 1,2), números complejos (1+2j), etc.

2) Cadenas: Cadena de texto, secuencias de datos de caracteres. El tipo string en Python se llama "str".

3) Lista: Una lista es una colección de elementos ordenados. Los elementos de una lista están indexados y el índice empieza por 0.

4) Tupla: Una tupla es similar a una lista pero es inmutable. Esto significa que, una vez creada, no se puede modificar.

5) Diccionario: Una colección desordenada de datos en un formato de par clave:valor.

Corre el programa: python tipos_de_datos.py

Operadores

Python incluye varios tipos de operadores:

Operadores Aritméticos: Se utilizan para realizar operaciones aritméticas como suma (+), resta (-), multiplicación (*), división (/), módulo (%), exponenciación (**) y división por el suelo (//).

Operadores de comparación: Se utilizan para comparar valores. Devuelven Verdadero o Falso según la condición. Ejemplos son igual a (==), no igual a (!=), mayor que (>), menor que (<), mayor o igual que (>=), y menor o igual que (<=).

Operadores de asignación: Se utilizan para asignar valores a variables. El más común es igual a (=), pero existen otros como sumar y asignar (+=), restar y asignar (-=), etc.

Operadores lógicos: Se utilizan para combinar sentencias condicionales. Incluyen y, o y no.

Corre el programa: python operadores.py

Módulo 2.2: Estructuras de control: Bucles y condicionales

Las estructuras de control son bloques de programación que analizan variables y eligen una dirección a seguir basándose en parámetros dados. En Python, las principales estructuras de control son los bucles y los condicionales.

Bucles

Los bucles se utilizan cuando se quiere repetir un bloque de código varias veces. Python tiene dos comandos de bucle primitivos:

bucles for: Un bucle for se utiliza para iterar sobre una secuencia (como una lista, tupla, diccionario, conjunto o cadena).

Ejemplo:

```
for i in range(5):
    print(i)
```

Esto imprimirá los números del 0 al 4.

Bucle While

bucles while: Un bucle while se ejecuta mientras una condición sea verdadera.

```
i = 0

while i < 5

    print(i)

    i += 1
```

Esto también imprimirá los números del 0 al 4.

Condicionales

Los condicionales se usan para ejecutar un bloque de código si se cumple una determinada condición. Python utiliza if, elif (significa else if), y else para los condicionales.

Por ejemplo

```
x = 10

If x > 0:

    print("x es positivo")

elif x < 0:

    print("x es negativo")

else:

    print("x es cero")
```


Módulo 2.3: Funciones y módulos en Python

Funciones

Una función es un bloque de código que sólo se ejecuta cuando es llamada. Puedes pasar datos, conocidos como parámetros, a una función. Una función puede devolver datos como resultado.

Definición de una función en Python:

```
def saludar(nombre):
    print("Hola, " + nombre)

saludar("Alicia")
```

En este ejemplo, *saludar* es una función que recibe un parámetro nombre. Cuando llamamos a la función y le pasamos "Alicia" como parámetro, imprimirá "Hola, Alicia".

Módulos

Un módulo es un archivo que contiene definiciones y sentencias de Python. El nombre del archivo es el nombre del módulo con el sufijo .py añadido. Puedes utilizar cualquier archivo fuente de Python como módulo ejecutando una sentencia import en algún otro archivo fuente de Python.

Por ejemplo, si tiene un archivo Python operaciones_matemáticas.py:

```
# operaciones_matemáticas.py

def sumar(x, y):

    return x + y

def restar(x, y):

    return x - y
```

Puedes importar el módulo en otro fichero Python:

```
# main.py

import operaciones_matemáticas

resultado = operaciones_matemáticas.add(5, 3)

print(resultado) # imprime 8
```

Esto cubre los fundamentos de las estructuras de control, funciones y módulos en Python. En el próximo módulo, discutiremos las bibliotecas de Python y cómo se pueden utilizar para ampliar la funcionalidad de Python.

Módulo 2.4: Introducción a las bibliotecas de Python

Una librería Python es un trozo de código reutilizable que puedes querer incluir en tus programas o proyectos para ahorrar tiempo y esfuerzo. Aquí tienes algunas librerías Python de uso común:

NumPy: NumPy, siglas de Numerical Python, es una librería para el lenguaje de programación Python, que añade soporte para matrices y arrays multidimensionales de gran tamaño, junto con una gran colección de funciones matemáticas de alto nivel para operar sobre estos arrays.

Pandas: Pandas es una biblioteca de software para la manipulación y el análisis de datos. Proporciona estructuras de datos y funciones necesarias para manipular datos estructurados.

Matplotlib: Matplotlib es una biblioteca de trazado para el lenguaje de programación Python y su extensión de matemáticas numéricas NumPy. Permite crear gráficos estáticos, animados e interactivos en Python.

SciPy: SciPy es una biblioteca Python gratuita y de código abierto utilizada para la computación científica y la computación técnica.

TensorFlow: TensorFlow es una biblioteca de software libre y de código abierto para el aprendizaje automático y la inteligencia artificial. Puede utilizarse en una amplia gama de tareas, pero se centra especialmente en el entrenamiento y la inferencia de redes neuronales profundas.

Para utilizar una biblioteca en un programa Python, primero hay que instalarla e importarla. Por ejemplo, para utilizar NumPy, primero hay que instalarlo con pip (el gestor de paquetes de Python):

pip install numpy

Puedes guardar tus paquetes en un archivo requirements.txt para poder compartirlo después

pip freeze > requirements.txt

Para instalar desde requirements puedes hacer:

pip install -r requirements.txt

Módulo 2.5: Depuración y tratamiento de errores en Python

La depuración es una parte esencial de la programación. Python proporciona varias herramientas para facilitar la depuración.

Errores de Sintaxis: Los errores de sintaxis, también conocidos como errores de análisis sintáctico, son quizás el tipo de queja más común que se obtiene mientras se está aprendiendo Python.

while True print('Hola mundo')

El analizador sintáctico repite la línea infractora y muestra una pequeña 'flecha' señalando el primer punto de la línea donde se detectó el error.

Incluso si una sentencia o expresión es sintácticamente correcta, puede causar un error cuando se intenta ejecutarla. Los errores detectados durante la ejecución se denominan excepciones.

10 * (1/0)

La última línea del mensaje de error indica lo que ha ocurrido. Las excepciones son de diferentes tipos, y el tipo se imprime como parte del mensaje.

Manejo de Excepciones

Manejo de excepciones: La sentencia try funciona de la siguiente manera:

Primero, se ejecuta la cláusula try (la(s) sentencia(s) entre las palabras clave try y except).

Si no se produce ninguna excepción, se omite la cláusula except y finaliza la ejecución de la sentencia try.

Si se produce una excepción durante la ejecución de la cláusula try, se salta el resto de la cláusula. Entonces, si su tipo coincide con la excepción nombrada después de la palabra clave except, se ejecuta la cláusula except y la ejecución continúa después de la sentencia try.

```
try:
    x = 1 / 0
except ErrorDivisiónCero:
    x = 0
```

En este ejemplo, la división por cero es manejada con gracia por la cláusula except y la ejecución puede continuar.

Semana 3: Python en Raspberry Pi

Módulo 3.1: Entorno de desarrollo integrado (IDE) de Python en Raspberry Pi

Módulo 3.2: Ejecución de scripts Python en Raspberry Pi

Módulo 3.3: Pines GPIO en Raspberry Pi

Módulo 3.4: Interfaz con Hardware usando Python

Módulo 3.5: Proyecto: Construcción de un simple programa de LED parpadeante

Módulo 3.1: Entorno de desarrollo integrado (IDE) de Python en Raspberry Pi

Un Entorno de Desarrollo Integrado (IDE) es una aplicación de software que proporciona amplias facilidades a los programadores informáticos para el desarrollo de software. Por lo general, cuentan con un editor de código, herramientas para la depuración y herramientas de automatización, entre otros.

Cuando instalas Raspberry Pi OS, viene con un IDE de Python llamado Thonny. He aquí cómo usarlo:

1) Lanzamiento de Thonny: Haga clic en el icono de Raspberry en la esquina superior izquierda, vaya a Programación y haga clic en Thonny Python IDE.
2) Escribiendo Código: Una vez que Thonny está abierto, puede empezar a escribir su código Python de inmediato en el archivo abierto.
3) Ejecutar Código: Para ejecutar su código, simplemente haga clic en el botón "Ejecutar script actual" (el botón verde con la flecha blanca apuntando a la derecha) o vaya a "Ejecutar" en el menú y haga clic en "Ejecutar script actual".
4) Depuración de código: Thonny viene con un depurador incorporado. Puede establecer puntos de interrupción en su código haciendo clic junto a un número de línea, y luego ejecutar su código en modo "Depuración" haciendo clic en "Depurar script actual" o yendo a "Ejecutar" en el menú y haciendo clic en "Depurar script actual".

Módulo 3.2: Ejecutar Scripts Python en Raspberry Pi

Los scripts de Python pueden ejecutarse directamente desde el terminal en Raspberry Pi. Aquí están los pasos para hacerlo:

1) Escribir un Script de Python: Abra un editor de texto (como Thonny, o incluso un editor simple como nano), escriba su script Python y guarde el archivo con una extensión .py. Por ejemplo, puedes escribir un script simple como este:

   ```
   print("¡Hola, Raspberry Pi!")
   ```

2) Guárdalo como hola.py.
3) Ejecutar el script: Abre una ventana de terminal. Navega al directorio donde guardaste tu script Python utilizando el comando cd (cambiar directorio). A continuación, puedes ejecutar el script escribiendo python o python3 seguido del nombre del archivo:

   ```
   python3 hola.py
   ```

Esto debería imprimir "¡Hola, Raspberry Pi!" en el terminal.

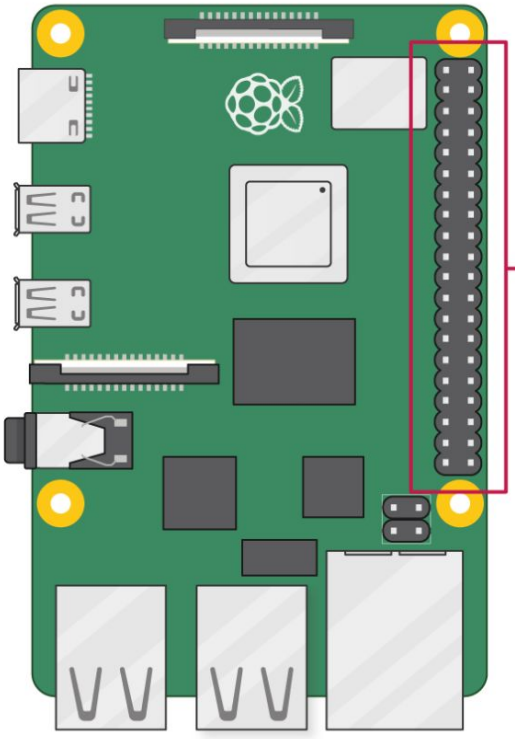
Esto cubre los conceptos básicos de la utilización de un IDE en la Raspberry Pi y la ejecución de scripts de Python. En los próximos módulos, vamos a empezar a trabajar con los pines GPIO en la Raspberry Pi, que nos permitirá interactuar con hardware externo.

Módulo 3.3: Pines GPIO en Raspberry Pi

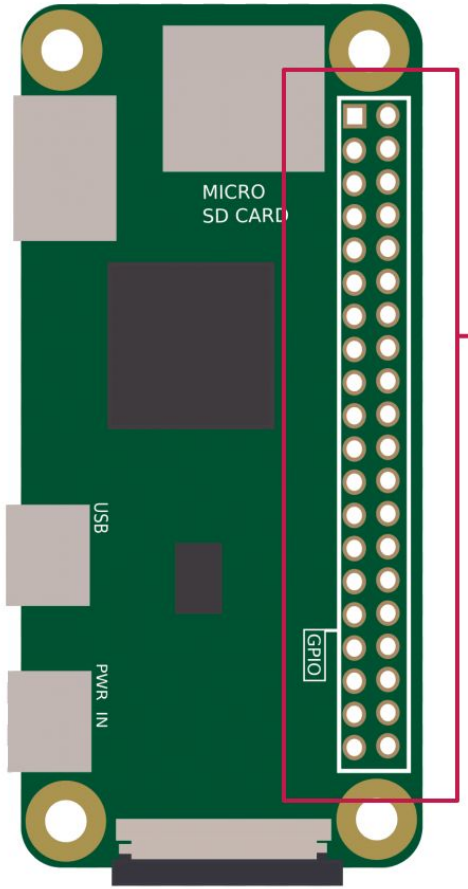
GPIO, o General-Purpose Input/Output, es una característica de la Raspberry Pi que le permite interactuar con el mundo físico. Los pines GPIO pueden ser configurados como entrada o salida y utilizados para una amplia gama de propósitos.

https://www.raspberrypi.com/documentation/computers/os.html

La Raspberry Pi tiene un conjunto de 40 pines en la placa. No todos estos pines son para GPIO, también incluyen pines de alimentación y tierra.



3V3 power	1	2		5V power
GPIO 2 (SDA)	3	4		5V power
GPIO 3 (SCL)	5	6		Ground
GPIO 4 (GPCLK0)	7	8		GPIO 14 (TXD)
Ground	9	10		GPIO 15 (RXD)
GPIO 17	11	12		GPIO 18 (PCM_CLK)
GPIO 27	13	14		Ground
GPIO 22	15	16		GPIO 23
3V3 power	17	18		GPIO 24
GPIO 10 (MOSI)	19	20		Ground
GPIO 9 (MISO)	21	22		GPIO 25
GPIO 11 (SCLK)	23	24		GPIO 8 (CE0)
Ground	25	26		GPIO 7 (CE1)
GPIO 0 (ID_SD)	27	28		GPIO 1 (ID_SC)
GPIO 5	29	30		Ground
GPIO 6	31	32		GPIO 12 (PWM0)
GPIO 13 (PWM1)	33	34		Ground
GPIO 19 (PCM_FS)	35	36		GPIO 16
GPIO 26	37	38		GPIO 20 (PCM_DIN)
Ground	39	40		GPIO 21 (PCM_DOUT)



3V3 power	1	2	5V power
GPIO 2 (SDA)	3	4	5V power
GPIO 3 (SCL)	5	6	Ground
GPIO 4 (GPCLK0)	7	8	GPIO 14 (TXD)
Ground	9	10	GPIO 15 (RXD)
GPIO 17	11	12	GPIO 18 (PCM_CLK)
GPIO 27	13	14	Ground
GPIO 22	15	16	GPIO 23
3V3 power	17	18	GPIO 24
GPIO 10 (MOSI)	19	20	Ground
GPIO 9 (MISO)	21	22	GPIO 25
GPIO 11 (SCLK)	23	24	GPIO 8 (CE0)
Ground	25	26	GPIO 7 (CE1)
GPIO 0 (ID_SD)	27	28	GPIO 1 (ID_SC)
GPIO 5	29	30	Ground
GPIO 6	31	32	GPIO 12 (PWM0)
GPIO 13 (PWM1)	33	34	Ground
GPIO 19 (PCM_FS)	35	36	GPIO 16
GPIO 26	37	38	GPIO 20 (PCM_DIN)
Ground	39	40	GPIO 21 (PCM_DOUT)

He aquí cómo utilizar los pines GPIO:

Sistemas de numeración: Los pines GPIO pueden ser referidos por su número de pin físico o por su número GPIO (también conocido como numeración BCM).

Configurar un pin: Puedes configurar un pin como entrada o como salida. Un pin de entrada se utiliza para leer un voltaje (detectar si un botón está pulsado o no), y un pin de salida se utiliza para establecer un voltaje (encender/apagar un LED).

Lectura y Escritura de Pines:

- Si has configurado un pin como entrada, puedes usar la función GPIO.input() para leer su valor.
- Si has configurado un pin como salida, puedes usar la función GPIO.output() para establecer su valor.

Módulo 3.4: Interfaz con Hardware usando Python

La interfaz con el hardware es donde comienza la verdadera diversión con Raspberry Pi. Escribiendo scripts en Python, puedes interactuar con componentes físicos como LEDs, botones, sensores y más. He aquí un ejemplo sencillo de cómo encender un LED:

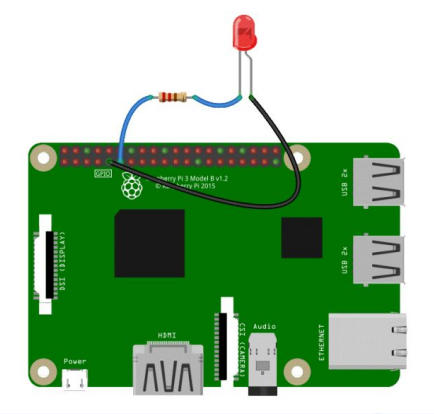
1) Conectar el LED: Conecta el LED a uno de los pines GPIO de la Raspberry Pi. La pata positiva (la más larga) del LED debe estar conectada al pin GPIO y la pata negativa debe estar conectada a un pin de tierra a través de una resistencia.

2) Configurar el pin: En tu script Python, necesitas importar la librería GPIO, configurar el modo de numeración GPIO, y configurar el pin como una salida:

import RPi.GPIO como GPIO

GPIO.setmode(GPIO.BCM)

GPIO.setup(17, GPIO.OUT)



Controlar Pines GPIO con Python

Encendiendo el LED: Para encender el LED, necesitas poner la salida del pin en HIGH:

GPIO.output(17, GPIO.HIGH)

Apagar el LED: Para apagar el LED, tienes que poner la salida del pin en LOW:

GPIO.output(17, GPIO.LOW)

Recuerda siempre limpiar al final de tus scripts para resetear los pines GPIO:

GPIO.cleanup()

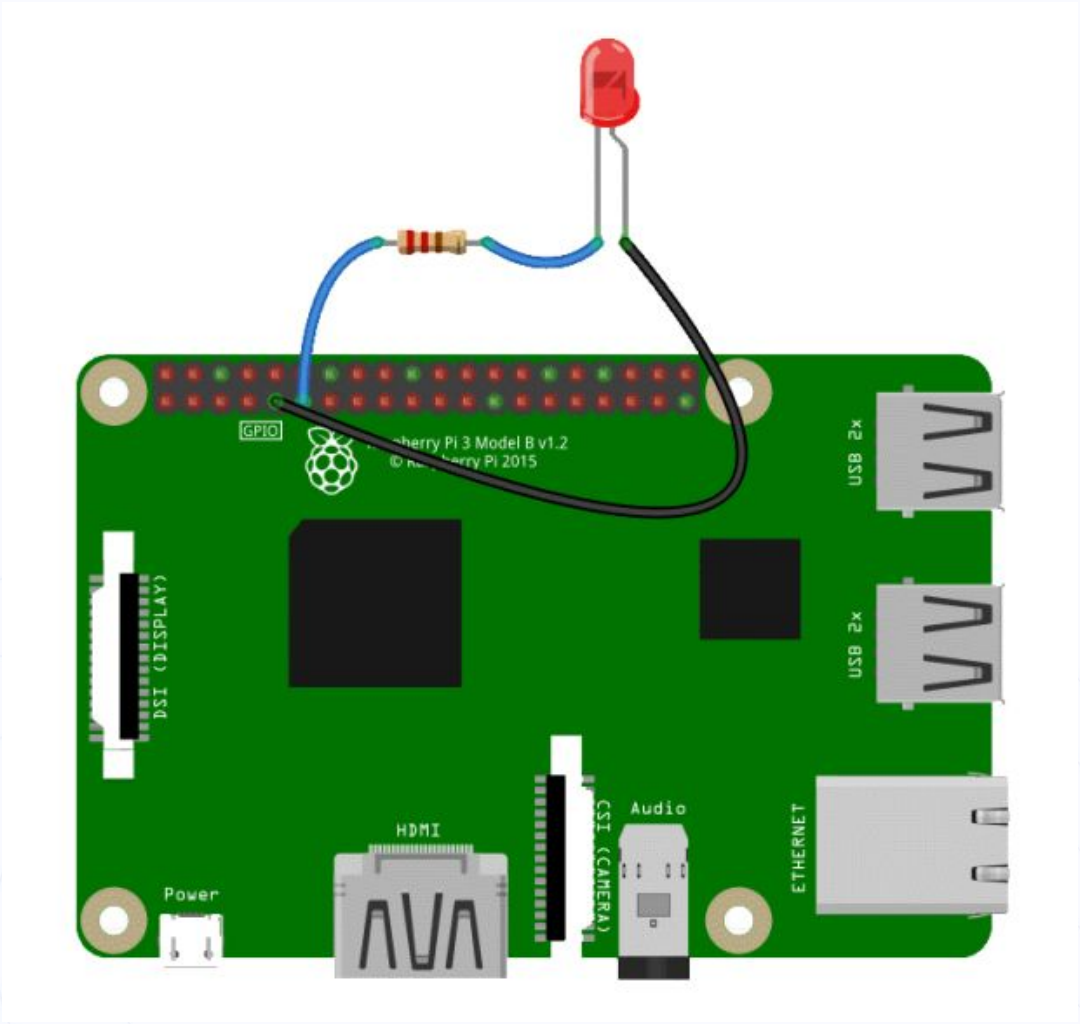
En los próximos módulos, profundizaremos en GPIO y veremos cómo trabajar con diferentes tipos de sensores y dispositivos.

Módulo 3.5: Proyecto: Creando un Simple Programa de Parpadeo de LEDs

Ahora que ya conocemos la programación en Python y la interfaz GPIO, vamos a poner en práctica nuestros conocimientos creando un sencillo programa de parpadeo de LEDs.

Materiales Requeridos:

- Raspberry Pi
- Breadboard
- LED
- Resistencia de 220 ohmios
- Cables de puente


GPIO
USB 2x
USB 2x
DSI (DISPLAY)
HDMI
CSI (CAMERA)
Audio
ETHERNET
Power

Configuración del hardware:

1. Conecta el pin GPIO17 de la Raspberry Pi a la pata positiva (la más larga) del LED de la protoboard.
2. Conecta la pata negativa (la más corta) del LED a un extremo de la resistencia.
3. Conecta el otro extremo de la resistencia al pin GND de la Raspberry Pi.

Configuración del software:

- Abre el IDE Thonny Python en tu Raspberry Pi.

Escribe el siguiente script en Python:

Ver Script led_blink.py

Ejecuta el script pulsando el botón "Run current script" en Thonny o escribiendo python3 led_blink.py en el terminal.

Ahora deberías ver tu LED parpadeando cada segundo. Este sencillo proyecto te da una idea básica de cómo controlar hardware (un LED en este caso) utilizando programación Python en la Raspberry Pi. Exploraremos proyectos más complejos en los próximos módulos.

Semana 4: Recopilación y procesamiento de datos en Raspberry Pi

Módulo 4.1: Recopilación de datos de sensores con Python

Módulo 4.2: Introducción a las API

Módulo 4.3: Almacenamiento y recuperación de datos con Python

Módulo 4.4: Procesamiento y análisis de datos

Módulo 4.5: Proyecto: Registrador de datos

Módulo 4.1: Recopilación de Datos de Sensores con Python

La recopilación de datos de los sensores es una tarea común en los proyectos de Raspberry Pi. Varios sensores se pueden conectar a la Raspberry Pi, tales como sensores de temperatura, sensores de humedad, sensores de luz, y más.

En este módulo, vamos a cubrir los conceptos básicos de cómo recopilar datos de un sensor utilizando Python. Utilizaremos el sensor de temperatura y humedad DHT11 como ejemplo.

Materiales necesarios:

- Raspberry Pi
- Sensor de temperatura y humedad DHT11
- Breadboard
- Cables de puente

Configuración del hardware:

Conecta el pin VCC del sensor DHT11 al pin 5V de la Raspberry Pi.

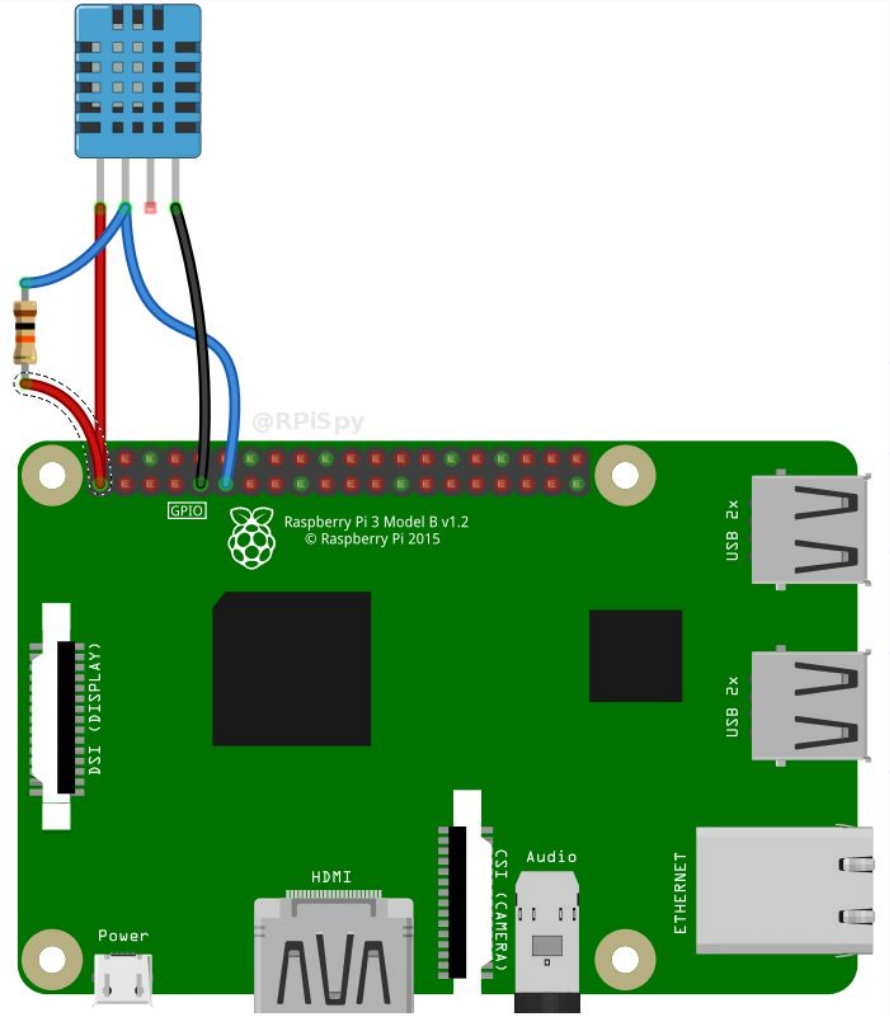
Conecta el pin GND del sensor al pin GND de la Raspberry Pi.

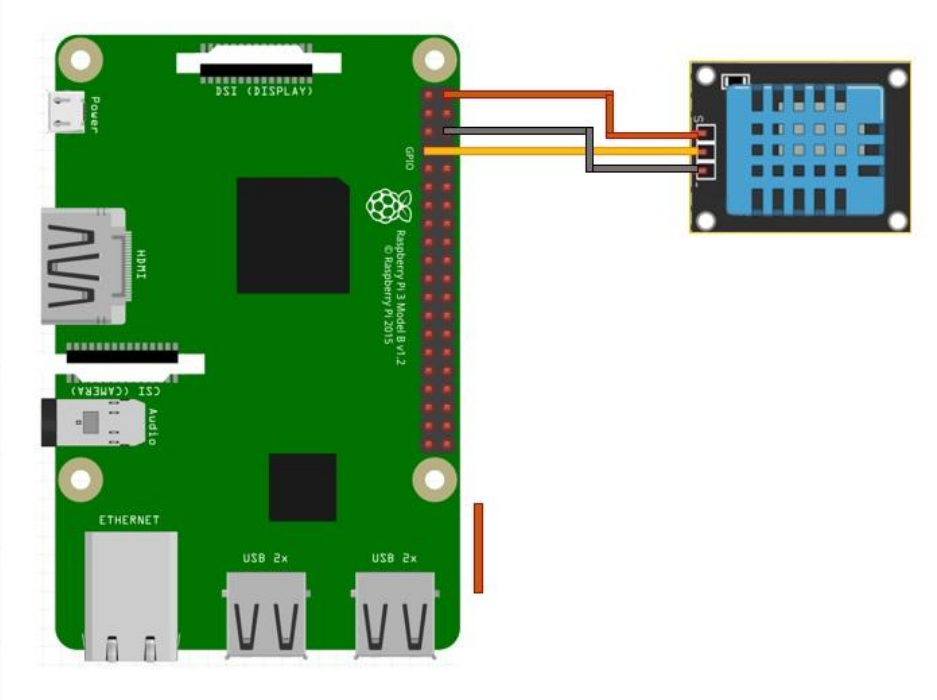
Conecta el pin DATA del sensor al GPIO4 de la Raspberry Pi.

Configuración del software:

Para leer los datos del sensor DHT11, tenemos que instalar la biblioteca Adafruit_DHT. Esto se puede hacer abriendo el terminal y escribiendo

sudo pip3 install Adafruit_DHT


@RPiSpy
GPIO
Raspberry Pi 3 Model B v1.2
© Raspberry Pi 2015
USB 2x
USB 2x
DSI (DISPLAY)
HDMI
CSI (CAMERA)
Audio
ETHERNET
Power


DSI (DISPLAY)
Power
GPIO
HDMI
CSI (CAMERA)
Audio
ETHERNET
USB 2x
USB 2x
Raspberry Pi 3 Model B v1.2
© Raspberry Pi 2015

Ahora, podemos leer los datos del sensor usando un script de Python:

Ver humidity_sensor.py

Cuando ejecutes este script, imprimirá las lecturas de temperatura y humedad del sensor.

Recuerde que cada sensor es diferente, por lo que el proceso de lectura de datos variará. Consulte siempre la hoja de datos o la guía de su sensor específico.

En los próximos módulos, aprenderemos a almacenar los datos del sensor y a crear un servidor web para mostrar los datos de nuestro sensor.

Módulo 4.2: Introducción a las API

API, o Interfaz de Programación de Aplicaciones, es un conjunto de reglas y protocolos para construir e interactuar con aplicaciones de software. Una API define la forma correcta en que un desarrollador puede solicitar servicios a una aplicación y cómo introducir y enviar datos.

En el contexto del desarrollo web, una API (a menudo denominada Web API) es un conjunto de mensajes de solicitud HTTP, junto con una definición de la estructura de los mensajes de respuesta, normalmente expresados en JSON o XML. Las API se utilizan para permitir la interacción entre distintos sistemas de software.

He aquí un ejemplo sencillo de cómo podría utilizarse una API meteorológica en Python:

```
import requests

# Haz una petición get a la API OpenWeatherMap

response = requests.get("http://api.openweathermap.org/data/2.5/weather?q=London&appid=your_api_key")

# Imprime el código de estado (200 significa que la solicitud se ha realizado correctamente)

print(response.status_code)

# Imprime los datos recibidos (estarán en formato JSON)

print(response.json())
```


Módulo 4.3: Almacenamiento de datos con Python

Django REST Framework (DRF) es un conjunto de herramientas potente y flexible para crear API web en Python. Funciona en conjunto con Django, un framework Web Python de alto nivel que fomenta el desarrollo rápido y el diseño limpio y pragmático.

Antes de comenzar, se asume que tienes Django y Django REST Framework instalados. Si no es así, se pueden instalar usando pip:

pip install django

pip install djangorestframework

Instalar Rust Compiler

python -m pip install --upgrade pip

curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh

Crear Modelo

Define tu Modelo: En tu archivo models.py (ubicado en el directorio de la app api), define un modelo llamado Reading para almacenar los datos de temperatura y humedad.

```
from django.db import models

class Reading(models.Model):

    date = models.DateTimeField(auto_now_add=True)

    temp = models.DecimalField(max_digits=5, decimal_places=2)

    humidity = models.DecimalField(max_digits=5, decimal_places=2)

    def __str__(self):

        return f"{self.date}: Temp = {self.temp}, Humidity = {self.humidity}"
```


Serializador

Migra tu Base de Datos: Ejecuta python manage.py makemigrations api y python manage.py migrate para crear la tabla en tu base de datos.

Crea un serializador: En un nuevo archivo serializers.py en el directorio api, define un serializador para tu modelo. Esto permitirá a Django REST Framework convertir tipos de datos complejos en tipos de datos Python que luego pueden ser fácilmente renderizados en JSON.

```
from rest_framework import serializers

from .models import Reading

class ReadingSerializer(serializers.ModelSerializer):

    class Meta:

        model = Reading

        fields = ['date', 'temp', 'humidity']
```


Vista

Crea una vista: En tu fichero views.py (también en el directorio api), utiliza el serializador y el modelo para crear una vista.

```
from rest_framework import viewsets

from .models import Reading

from .serializers import ReadingSerializer


class ReadingViewSet(viewsets.ModelViewSet):

    queryset = Reading.objects.all().order_by('-date')

    serializer_class = ReadingSerializer
```


Url

Registra la vista con una URL: Finalmente, en tu archivo urls.py (de nuevo en el directorio api), registra la vista.

```
from django.urls import path

From .views import *

urlpatterns = [

    path('', ReadingViewSet.as_view()),

]
```


Correr Servidor

Ahora deberías poder ejecutar tu servidor Django con python manage.py runserver y navegar a http://localhost:8000/readings/ para ver la lista de lecturas. También puedes enviar nuevas lecturas a esta URL con una petición POST.

Este ejemplo muestra cómo almacenar y recuperar datos utilizando Django y Django REST Framework. Sin embargo, DRF es muy potente y tiene muchas más características, tales como autenticación y permisos, serialización de objetos anidados, y mucho más.

Agregar Datos a Django

Para ello, primero tenemos que modificar nuestra vista Django para aceptar peticiones POST con datos de temperatura y humedad. Podemos añadir un método a nuestro ReadingViewSet que escuche las peticiones POST.

En primer lugar, vamos a instalar la biblioteca necesaria para leer desde el sensor DHT11:

sudo pip3 install Adafruit_DHT

Luego, modifica el archivo views.py en el directorio api:

import Adafruit_DHT

Ver views.py del proyecto

Módulo 4.4: Procesar y analizar datos

Ahora que hemos recogido algunos datos de nuestro sensor y los hemos almacenado en una base de datos, el siguiente paso es procesar y analizar esos datos. Esto puede implicar limpiar los datos, calcular estadísticas y visualizar los datos. Python proporciona muchas bibliotecas para el análisis de datos, como pandas, numpy y matplotlib.

1. 1. Limpieza de datos:

Antes de analizar los datos, es importante limpiarlos y tratar cualquier punto de datos que falte, sea incorrecto o irrelevante. Por ejemplo, si el sensor DHT11 ocasionalmente falla en la lectura y devuelve Ninguno, es posible que desee eliminar estas lecturas de sus datos.

```
import pandas as pd

# Cargar los datos de la base de datos en un DataFrame

df = pd.read_sql_query("SELECT * FROM readings", conn)

# Eliminar filas con datos perdidos

df = df.dropna()
```


2. 2. Cálculo de estadísticas:

Una vez que los datos están limpios, puede calcular varias estadísticas para obtener una mejor comprensión de los mismos. Por ejemplo, puede que quieras saber la temperatura y la humedad medias, o cuánto varían.

```
# Compute average temperature and humidity

avg_temp = df['temp'].mean()

avg_humidity = df['humidity'].mean()

# Compute standard deviation of temperature and humidity

std_temp = df['temp'].std()

std_humidity = df['humidity'].std()
```


3. Visualización de datos:

La visualización de datos puede ser útil para identificar patrones o tendencias. Por ejemplo, es posible que desee crear un gráfico de series temporales de la temperatura y la humedad.

```
import matplotlib.pyplot as plt

# Trazar la temperatura y la humedad a lo largo del tiempo

plt.figure(figsize=(12, 6))

plt.plot(df['date'], df['temp'], label='Temperature')

plt.plot(df['date'], df['humidity'], label='Humidity')

plt.xlabel('Date')

plt.ylabel('Value')

plt.title('Temperature and Humidity Over Time')

plt.legend()

plt.show()
```


Semana 5: Redes y Raspberry Pi

Módulo 5.1: Fundamentos de la conexión en red

Módulo 5.2: Redes con Python

Módulo 5.3: Introducción a IoT (Internet de las cosas)

Módulo 5.4: Conexión de Raspberry Pi a Internet y Red Local

Módulo 5.5: Proyecto: Control Remoto de LED

Módulo 5.1: Fundamentos de Redes

La creación de redes es un campo muy amplio, pero para los propósitos de este curso, nos centraremos en los conceptos básicos que necesita saber para obtener su Raspberry Pi conectada y comunicada.

Los conceptos clave incluyen:

Direcciones IP: Una dirección IP (Protocolo de Internet) es un identificador único para los dispositivos en una red. Las direcciones IP pueden ser estáticas (nunca cambian) o dinámicas (cambian cada vez que un dispositivo se conecta a una red).

Puertos: Los puertos son como "puertas" por las que entra y sale el tráfico de la red. Los distintos servicios utilizan números de puerto diferentes; por ejemplo, el tráfico HTTP suele pasar por el puerto 80.

Protocolos: Los protocolos son conjuntos de reglas sobre cómo se envían y reciben los datos. Los protocolos más comunes son HTTP (Protocolo de Transferencia de Hipertexto) para el tráfico web, y TCP/IP (Protocolo de Control de Transmisión/Protocolo de Internet), los protocolos fundamentales en los que se basa Internet.

DNS (Sistema de Nombres de Dominio): El DNS es el sistema que traduce los nombres de dominio legibles (como www.google.com) en direcciones IP comprensibles para los ordenadores.

Enrutadores y cortafuegos: Los routers dirigen el tráfico en una red, mientras que los cortafuegos actúan como barreras de seguridad para impedir el acceso no autorizado.

Módulo 5.2: Redes con Python

Python tiene soporte incorporado para crear, enviar y recibir peticiones de red. Las librerías clave incluyen socket para programación de red de bajo nivel, y requests para hacer peticiones HTTP.

Módulo 5.3: Introducción a IoT (Internet de las cosas)

IoT se refiere a la red de dispositivos físicos (como su Raspberry Pi) que están conectados a Internet, lo que les permite enviar, recibir y procesar datos. Los dispositivos IoT pueden incluir cualquier cosa, desde electrodomésticos hasta maquinaria industrial.

Módulo 5.4: Conexión de la Raspberry Pi a Internet y a la Red Local

Su Raspberry Pi se puede conectar a su red local y a Internet mediante un cable Ethernet o Wi-Fi. Una vez conectado, puede SSH en su Pi, lo que le permite controlarlo de forma remota desde otro ordenador en la misma red.

Módulo 5.5: Proyecto: Control Remoto de LEDs

En este proyecto, vamos a crear un servidor web simple en la Raspberry Pi que le permite encender y apagar un LED desde una página web. Esto implica conectar un LED a tu Raspberry Pi, escribir un script en Python para controlar el LED, y configurar un servidor web para aceptar peticiones de Internet.

Semana 6: IA en Raspberry Pi

Módulo 6.1: Introducción a la Inteligencia Artificial

Módulo 6.2: Bibliotecas Python para IA

Módulo 6.3: Implementación de algoritmos sencillos de IA

Módulo 6.4: IA con datos de sensores

Módulo 6.5: Proyecto: IA para predecir la temperatura

Módulo 6.1: Introducción a la Inteligencia Artificial

La Inteligencia Artificial (IA) es un amplio campo que implica la creación de máquinas y programas informáticos que puedan mostrar una inteligencia similar a la humana. Es un campo multidisciplinar que se nutre de la informática, las matemáticas, la psicología, la lingüística, la filosofía y muchas otras áreas. La IA puede clasificarse en dos tipos principales:

IA restringida: son sistemas diseñados para realizar una tarea concreta, como el reconocimiento de voz, los sistemas de recomendación o el reconocimiento de imágenes. Funcionan con un conjunto limitado de restricciones y se centran en una única tarea. Este es el tipo de IA que vemos en nuestro día a día (como Siri y Google Assistant).

IA general: Este tipo de IA puede realizar cualquier tarea intelectual que pueda realizar un ser humano. Es más compleja e involucrada que Narrow AI. A partir de mi corte de conocimiento en 2021, este tipo de IA aún no existe.

Módulo 6.2: Bibliotecas Python para la IA

Python es uno de los lenguajes más populares para la IA y el aprendizaje automático, y se han desarrollado muchas librerías para facilitar estas tareas:

NumPy: Esta librería proporciona soporte para arrays, matrices y muchas funciones matemáticas para operar sobre estas estructuras de datos.

Pandas: Esta librería proporciona estructuras de datos y herramientas de análisis de datos. Es particularmente buena para trabajar con datos etiquetados, como datos de archivos CSV o bases de datos.

Matplotlib: Esta es una biblioteca de trazado 2D para crear gráficos y diagramas. Es útil para visualizar datos, que es una parte clave de la IA y el aprendizaje automático.

Scikit-Learn: Esta biblioteca proporciona herramientas sencillas y eficientes para el aprendizaje automático, la minería de datos y el análisis de datos. Se basa en NumPy, SciPy y matplotlib.

TensorFlow: es una biblioteca de código abierto para el cálculo numérico y el aprendizaje automático a gran escala. TensorFlow agrupa muchos modelos y algoritmos de aprendizaje automático y aprendizaje profundo. Utiliza Python para proporcionar una API de front-end conveniente para construir aplicaciones con el marco, mientras ejecuta esas aplicaciones en C ++ de alto rendimiento.

Keras: Es una API de redes neuronales de alto nivel, escrita en Python y capaz de ejecutarse sobre TensorFlow, CNTK o Theano. Es fácil de usar y de crear prototipos, por lo que es una buena opción para los principiantes.

PyTorch: Se trata de una biblioteca de aprendizaje automático de código abierto basada en la biblioteca Torch, utilizada para aplicaciones como la visión por ordenador y el procesamiento del lenguaje natural, desarrollada principalmente por el laboratorio AI Research de Facebook.

Módulo 6.3: Implementación de algoritmos sencillos de IA

Comprender e implementar algoritmos de IA es una parte crucial del aprendizaje de la IA. Comience con algoritmos de IA simples como regresión lineal, árbol de decisión y k-nearest neighbors (KNN) antes de pasar a otros más complejos como redes neuronales y aprendizaje profundo. A continuación se muestra cómo se podría implementar un modelo de regresión lineal simple con scikit-learn:

```
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

import pandas as pd

# Assuming df is your DataFrame and 'target' is the column with the values you want to predict

X = df.drop('target', axis=1)

y = df['target']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

model = LinearRegression()

model.fit(X_train, y_train)

predictions = model.predict(X_test)
```


Módulo 6.4: IA con datos de sensores

Los algoritmos de IA pueden utilizarse para dar sentido a los datos de los sensores. Por ejemplo, puedes utilizar un algoritmo de agrupación para detectar anomalías en los datos de temperatura, o un modelo de regresión para predecir futuras lecturas de temperatura. El proceso suele implicar la recopilación de datos, su preprocesamiento, el entrenamiento de un modelo a partir de los datos y, a continuación, la realización de predicciones.

Módulo 6.5: Proyecto: IA para predecir la temperatura

En este proyecto, utilizarás los datos de los sensores que has estado recopilando para entrenar un modelo que pueda predecir futuras lecturas de temperatura.