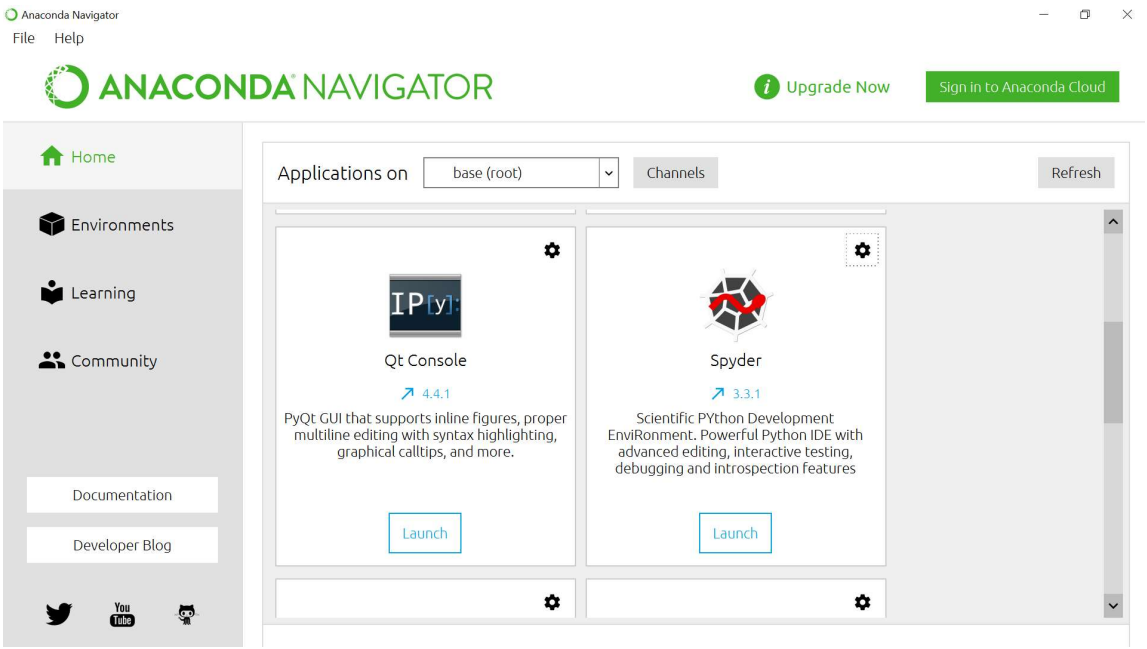
Tutorial de Python para controlar los GPIO de la Raspberry

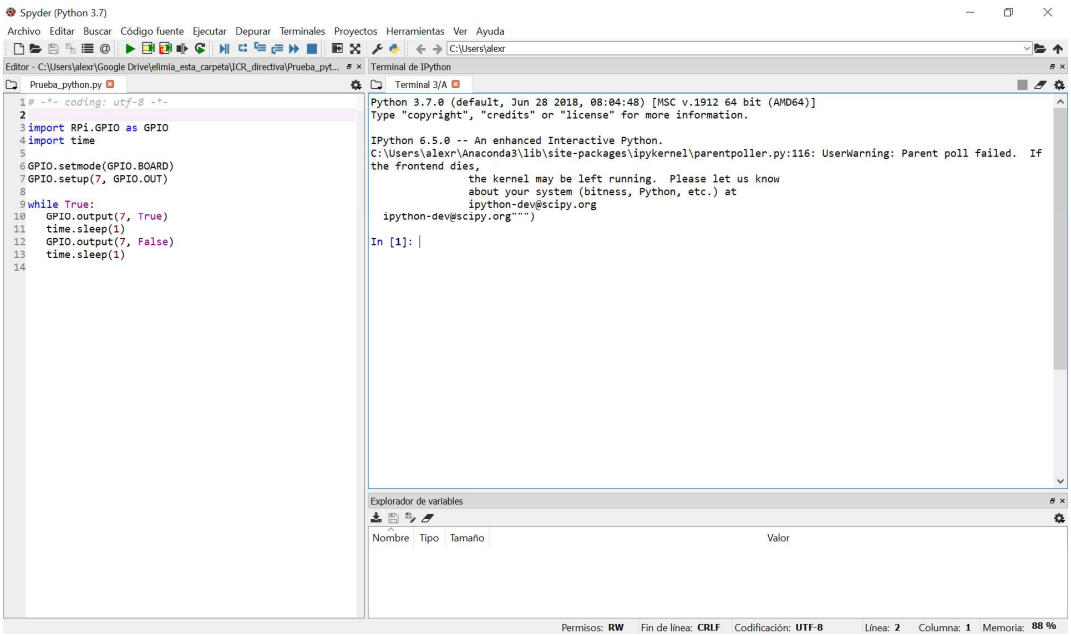
En este documento se explica cómo preparar el entorno de programación que necesitamos para probar el código en nuestros pc antes de cargarlo en la Raspberry. No es obligatorio que uséis este, pero si aún no tenéis ninguno este es sencillo de instalar, no da problemas y es muy como de usar.

Primero vamos a descargarnos *Anaconda*, un entorno de programación gráfico para Python y R que incluye una serie de paquetes muy útiles. Descargaros la versión hecha para Python 3.7.

https://www.anaconda.com/distribution/#download-section



Una vez instalado, dentro de la pestaña de “Home” nos saldrán varios programas, nosotros usaremos Spyder. Una vez arrancado nos aparecerá la siguiente pantalla:



Este será el entorno donde trabajemos, en el lado izquierdo tendremos nuestro archivo . *py* el cual editaremos y en el lado derecho el terminal de Python. Para ejecutar el programa tenemos que pulsar el triángulo verde que aparece en la barra de opciones superior. Además, *Spyder* nos permite debugear el código usando los símbolos de color azul que aparecen en la barra de opciones.

Para iniciarse en la sintaxis de los pines GPIO(muy básico):

https://www.programoergosum.com/cursos-online/raspberry-pi/238-control-de-gpio-con-python-en-raspberry-pi/introduccion

Como vamos a trabajar

Necesitáis saber que un archivo de Python *.py* puede tener dos funcionalidades, como un programa que ejecutamos o como un paquete de Python el cual ejecutamos como si fuese una librería (con la sentencia *import*). Para hacer más fácil la programación y la detección de errores en nuestro código vamos a trabajar de la siguiente manera: tendremos nuestro programa principal que desde el punto de vista de C será nuestro *main*. En este programa principal vamos a importar dos paquetes creados por nosotros, *logica* y *comunicacion* . En *logica* se van a implementar todas las funciones que tengan que ver con el funciona miento de los robots como por ejemplo logica.Mover() o logica.Golpear() y en *comunicacion* tendremos las funciones que conecten nuestro código en Python con los pines GPIO.

La primera razón de hacerlo así es que nos permite dividir el trabajo de forma más fácil y sin necesidad de trabajar todos sobre el mismo código. Lo segundo y más importante es que cuando queramos probar el código en nuestro ordenador solo tendremos que cambi*ar conexion* por un paquete en el que le metamos directamente los valores de los pines GPIO y por lo tanto no estaremos usando la biblioteca *RPi.GIPO*.

Este sería nuestro programa principal, el que vamos a ejecutar.

```
import conexion
import logica

def setup():
        pass
def loop():
        while True:
                #Leer entrada
                s1,s2 = conexion.read()

                #Ejecutar algoritmo
                m1,m2=logica.avanzar(s1,s2)

                #Enviar señal
                conexion.output(m1,m2)

#Ejecutar las funciones
setup()
loop()
```

Paquete *logica* (archivo *logica.py*):

```
def avanzar(s1,s2):
   #algoritmo
def girar(a, b):
   #algoritmo
```