MANUAL TÉCNICO

# VERSIÓN 01

RECURSOS DE HARDWARE

# Raspberry Pi 3 model b

Es un minicomputador de alto desempeño y bajo costo, perteneciente a la familia de las plataformas de prototipado y muy popular debido a sus cualidades.

## Características:

* Puertos GPIO 40 pines, HDMI, 4 USB 2.0, CSI, DSI, Audio, Micro SD, Micro USB.
* Conectividad Fast Ethernet 10 – 100 Gbps.
* Bluetooth 4.1
* Conectividad 2.4GHz IEEE 802.11.b/g/n
* Frecuencia de clock: 1.2 GHz
* Unidad de procesamiento gráfico VideoCore IV 400 MHz
* 1GB de memoria RAM

# Módulo GPS (GY-GPS6MV2)

Este módulo es capaz de reconocer la posición del dispositivo en términos de longitud y latitud; también la altura tomando como referencia el nivel del mar.

## Características:

* Rango de alimentación: 3V – 5V
* Antena cerámica
* Posee memoria EEPROM
* Batería de respaldo
* Indicador LED de señal
* Dimensiones de la antena: 25 x 25 mm
* Dimensiones del módulo: 25 x 35 mm
* Diámetro del orificio de montaje: 3 mm
* Tasa por defecto: 9600 bps

# Tarjeta Micro SD

Este dispositivo es necesario porque actuará como almacenamiento principal de las Raspberry y en él se instalará el sistema operativo (que será una distribución de Linux). Puede ser mínimo 8GB.

# Batería LiPo

Según los requerimientos de alimentación de la Raspberry Pi, la batería LiPo de 2500mAh puede suplir lo solicitado.

# Sensor Ultrasónico (HC-SR04)

Su función será medir distancia de un objeto próximo al frente del sensor mediante el uso de ondas de ultrasonido.

## Características:

* Puede reflejarse en vidrio o líquido.
* El polvo no afecta la detección.
* Alimentación de 5V DC.
* Rango de distancias: 2cm a 400cm

# Motores de corriente continua y Buzzers

Estos dispositivos serán utilizados para dirigir mediante estímulos al usuario.

RECURSOS DE SOFTWARE

# Raspbian OS

Esta distribución de Linux es uno de los sistemas operativos con los que la Raspberry Pi es compatible. Como proviene de Linux podemos contar con la útil interfaz de línea de comandos para programar.

# Python IDE

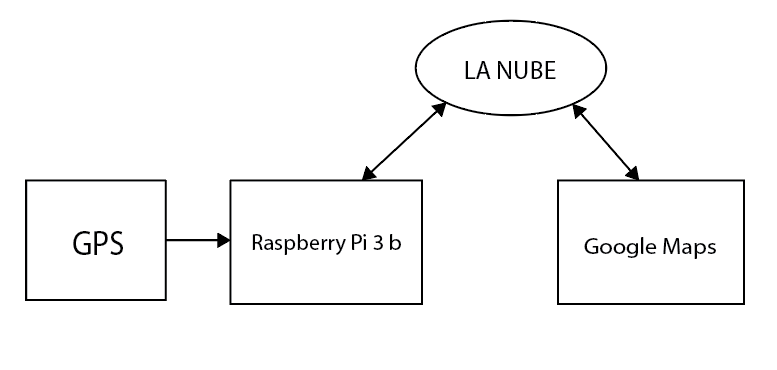
Ofrece un entorno de trabajo amigable para poder compilar y ejecutar el código correspondiente. Python es un lenguaje de alto nivel e interpretado, lo que lo hace una buena opción para cualquier tipo de programación como, la programación orientada a objetos.

# Google Maps API

Ofrece a los usuarios la posibilidad de desarrollar aplicaciones web que permiten mostrar instrucciones para llegar a determinada calle, entre otras características. Se usará esta API para interactuar con los datos de localización.

PROCESO GENERAL DE DESARROLLO

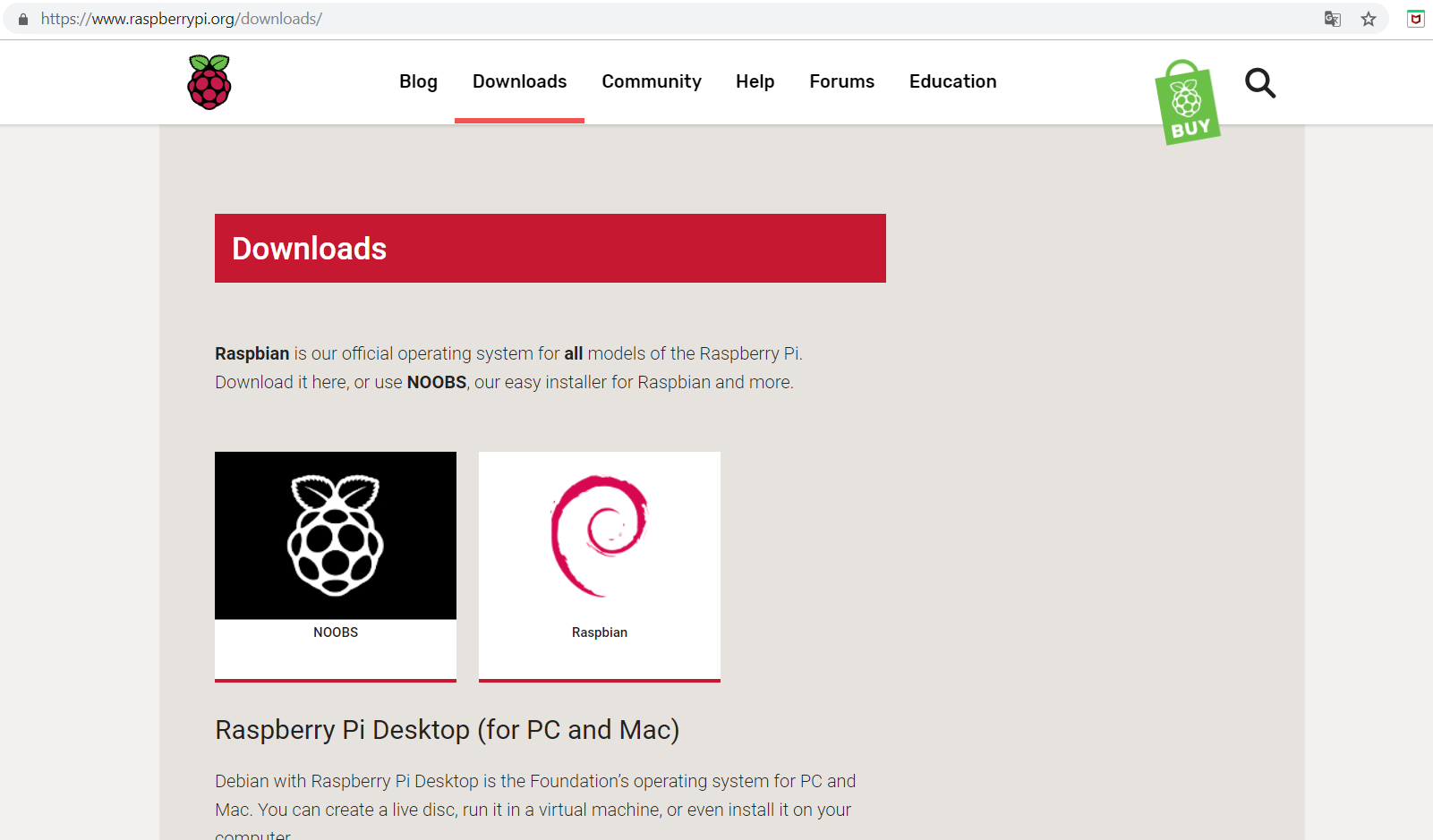
1. Descargar el sistema operativo Raspbian e instalarlo en la tarjeta micro SD.
2. Descargar e instalar Python IDE.
3. Programar la Raspberry Pi para que obtenga su ubicación del GPS y lo envíe a la nube para así el usuario puede dar seguimiento al posicionamiento.
4. Google Maps se usará para mostrar y manejar los datos de ubicación del dispositivo.



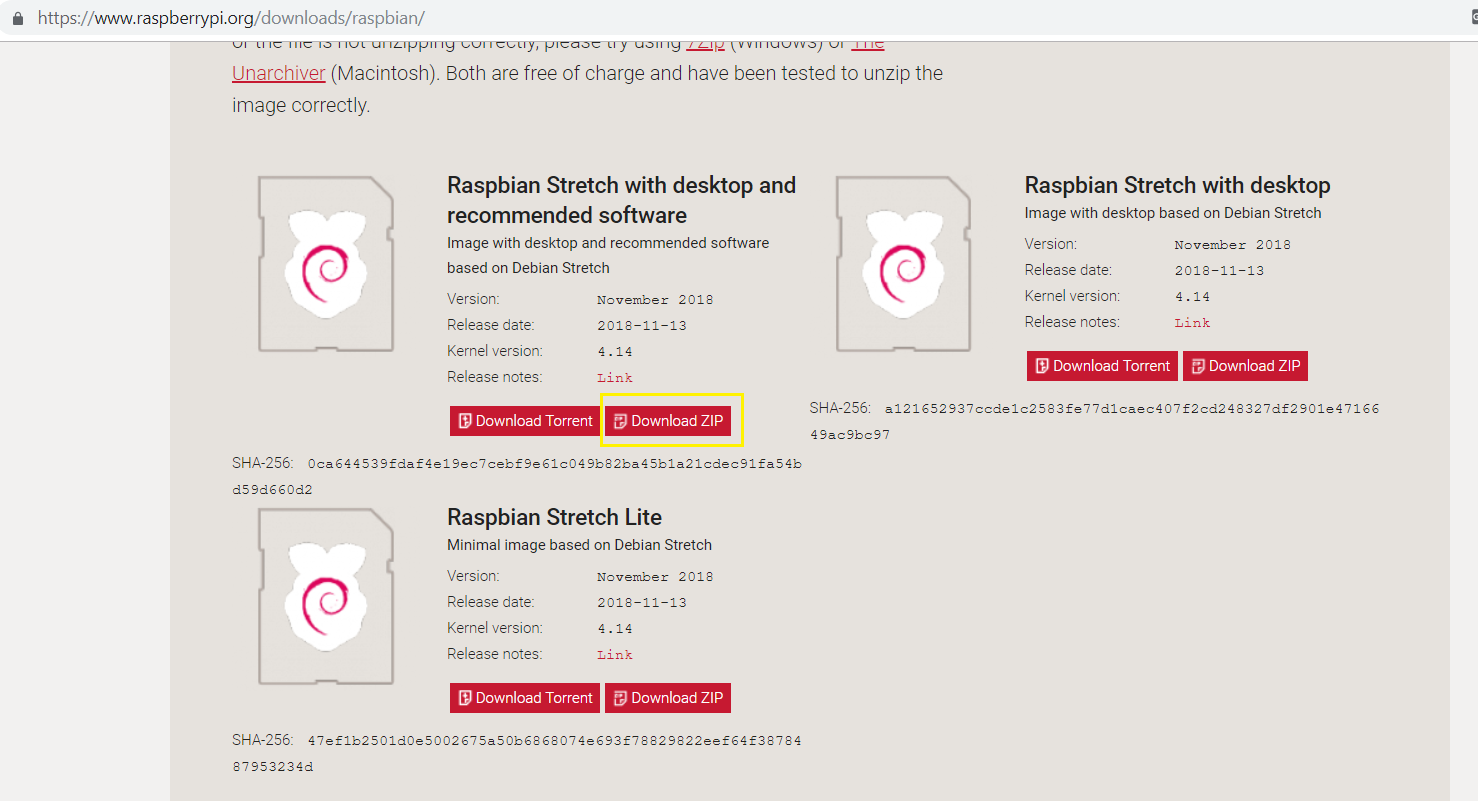
PROCESO DE IMPLEMENTACIÓN

# Instalación de Raspbian OS

1. Acceder al siguiente enlace: <https://www.raspberrypi.org/downloads/>

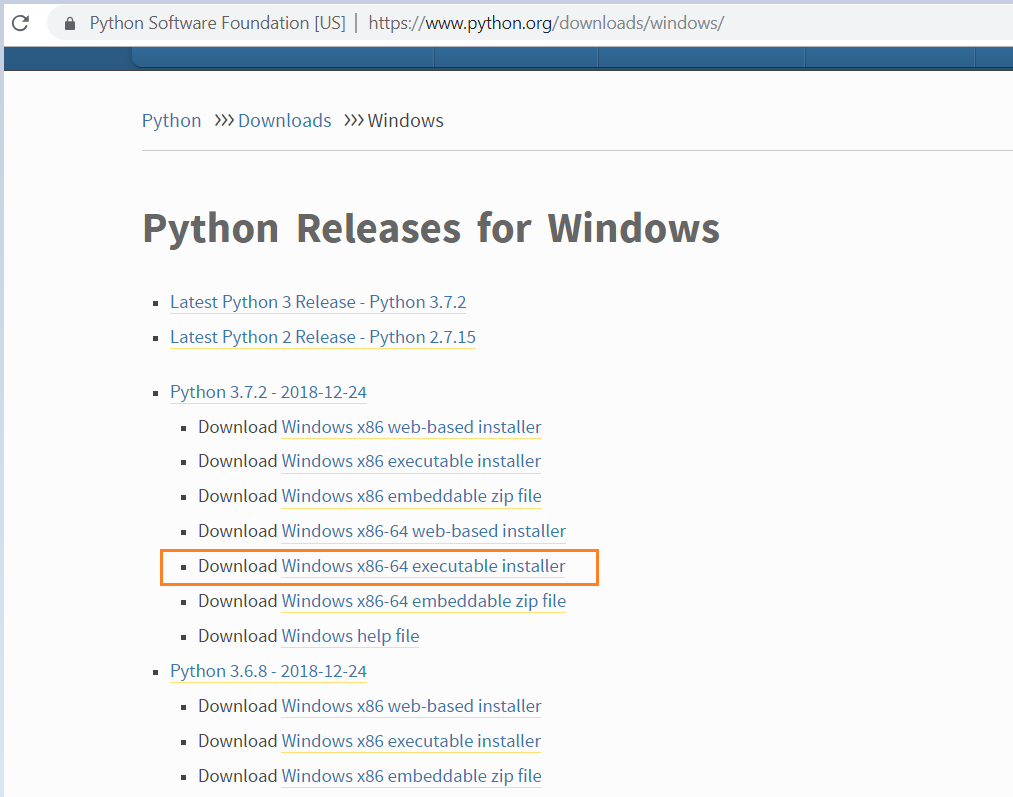


1. Accedemos a ‘Raspbian’. Y descargamos el archivo ZIP siguiente.

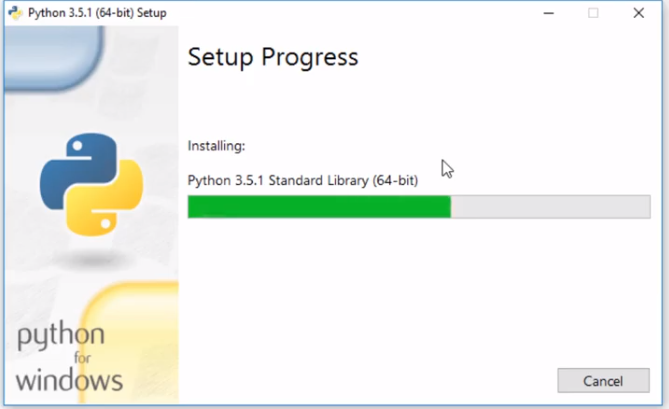


# Instalación de Python IDLE

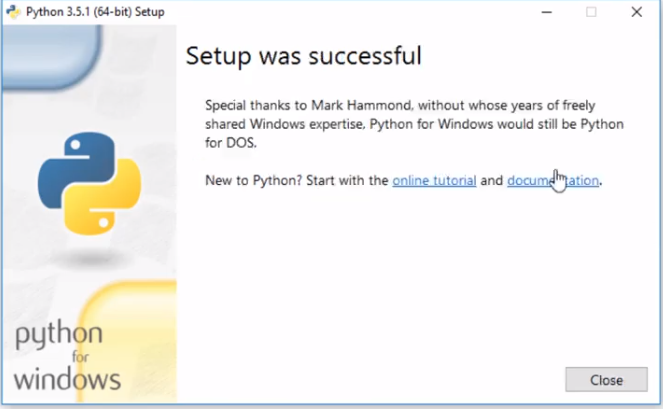
1. Acceder al siguiente enlace: <https://www.python.org/downloads/windows/>
2. Escoger la edición más apropiada.



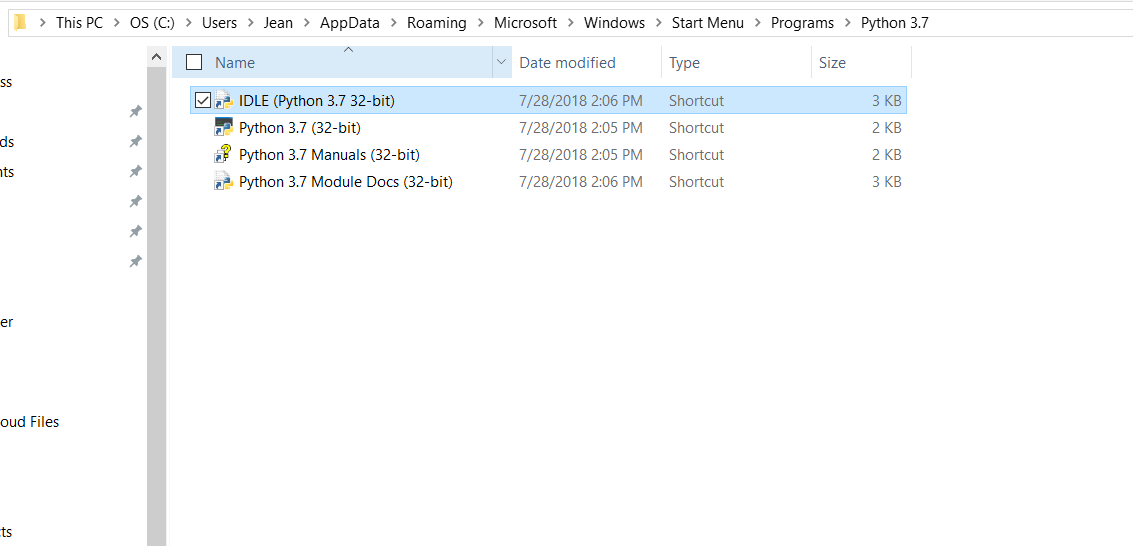
1. Abrimos dicho archivo descargado y procedemos a ejecutar la instalación.



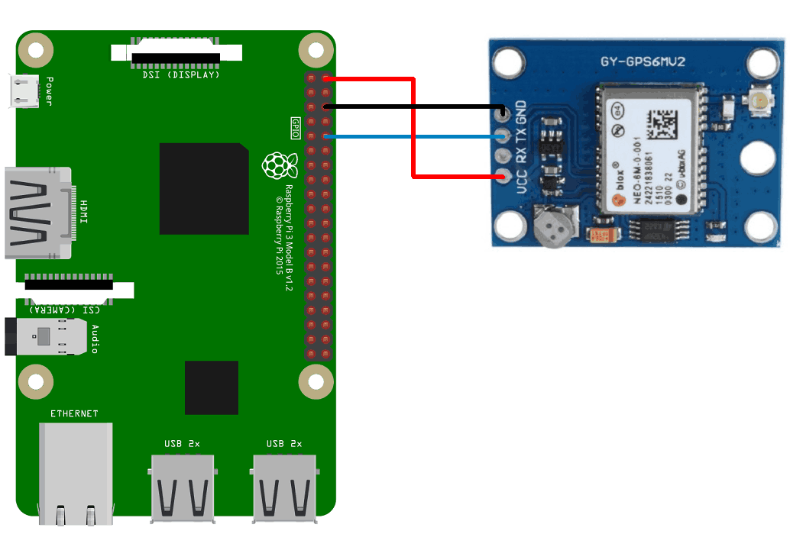
1. Una vez instalado, procedemos a dar clic en Close.



1. Ya podemos ver instalado el entorno de desarrollo para Python.



# Programación Raspberry Pi y GPS



import serial #import serial pacakge

from time import sleep

import webbrowser #import package for opening link in browser

import sys #import system package

def GPS\_Info():

global NMEA\_buff

global lat\_in\_degrees

global long\_in\_degrees

nmea\_time = []

nmea\_latitude = []

nmea\_longitude = []

nmea\_time = NMEA\_buff[0] #extract time from GPGGA string

nmea\_latitude = NMEA\_buff[1] #extract latitude from GPGGA string

nmea\_longitude = NMEA\_buff[3] #extract longitude from GPGGA string

print("NMEA Time: ", nmea\_time,'\n')

print ("NMEA Latitude:", nmea\_latitude,"NMEA Longitude:", nmea\_longitude,'\n')

lat = float(nmea\_latitude) #convert string into float for calculation

longi = float(nmea\_longitude) #convertr string into float for calculation

lat\_in\_degrees = convert\_to\_degrees(lat) #get latitude in degree decimal format

long\_in\_degrees = convert\_to\_degrees(longi) #get longitude in degree decimal format

#convert raw NMEA string into degree decimal format

def convert\_to\_degrees(raw\_value):

decimal\_value = raw\_value/100.00

degrees = int(decimal\_value)

mm\_mmmm = (decimal\_value - int(decimal\_value))/0.6

position = degrees + mm\_mmmm

position = "%.4f" %(position)

return position

gpgga\_info = "$GPGGA,"

ser = serial.Serial ("/dev/ttyS0") #Open port with baud rate

GPGGA\_buffer = 0

NMEA\_buff = 0

lat\_in\_degrees = 0

long\_in\_degrees = 0

try:

while True:

received\_data = (str)(ser.readline()) #read NMEA string received

GPGGA\_data\_available = received\_data.find(gpgga\_info) #check for NMEA GPGGA string

if (GPGGA\_data\_available>0):

GPGGA\_buffer = received\_data.split("$GPGGA,",1)[1] #store data coming after "$GPGGA," string

NMEA\_buff = (GPGGA\_buffer.split(',')) #store comma separated data in buffer

GPS\_Info() #get time, latitude, longitude

print("lat in degrees:", lat\_in\_degrees," long in degree: ", long\_in\_degrees, '\n')

map\_link = 'http://maps.google.com/?q=' + lat\_in\_degrees + ',' + long\_in\_degrees #create link to plot location on Google map

print("<<<<<<<<press ctrl+c to plot location on google maps>>>>>>\n") #press ctrl+c to plot on map and exit

print("------------------------------------------------------------\n")

except KeyboardInterrupt:

webbrowser.open(map\_link) #open current position information in google map

sys.exit(0)

