

Universidad Nacional Autónoma de México

Facultad de Ingeniería

Laboratorio de Fundamentos Embebidos

Práctica 4:

Manejo remoto de raspberry pi mediante bots de Telegram

Integrantes:

Cárdenas Cárdenas Jorge

Garrido Sánchez Samuel Arturo

Murrieta Villegas Alfonso

Reza Chavarría Sergio Gabriel

Valdespino Mendieta Joaquín

Práctica 4: Manejo remoto de raspberry pi mediante bots de telegram

Objetivo:

Manejo y monitoreo remoto de Raspberry Pi y sus puertos GPIO mediante la programación y aplicación de Bots de Telegram

Introducción:

Una API es un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones. API significa interfaz de programación de aplicaciones.

Las API permiten que sus productos y servicios se comuniquen con otros, sin necesidad de saber cómo están implementados. Esto simplifica el desarrollo de las aplicaciones y permite ahorrar tiempo y dinero. Las API le otorgan flexibilidad; simplifican el diseño, la administración y el uso de las aplicaciones, y proporcionan oportunidades de innovación.

En la presente práctica, emplearemos la API de Telegram para poder usar uno de los bots más conocidos dentro de esta comunidad que es BotFather.

Por otro lado, un sensor ultrasónico es un dispositivo electrónico que mide la distancia de un objeto objetivo mediante la emisión de ondas sonoras ultrasónicas y convierte el sonido reflejado en una señal eléctrica.

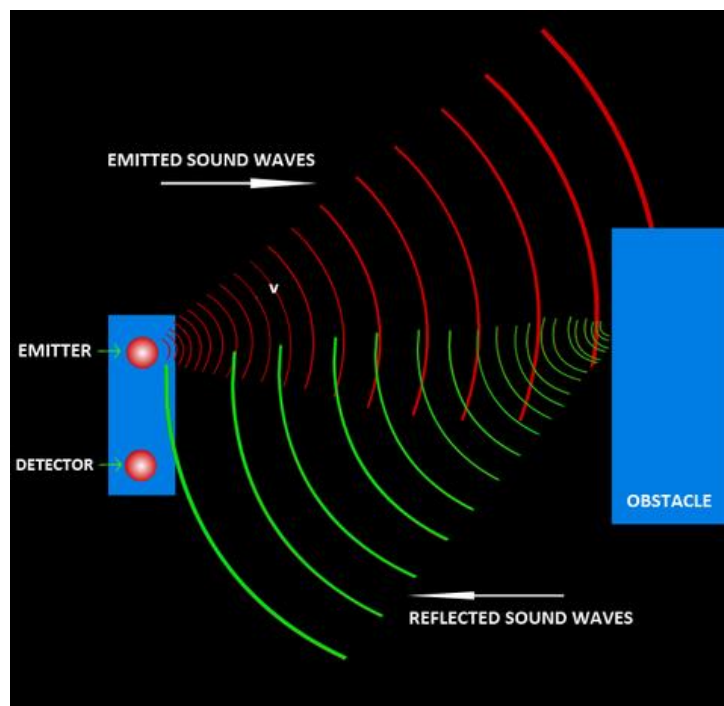


Imagen 1: Funcionamiento del sensor ultrasónico

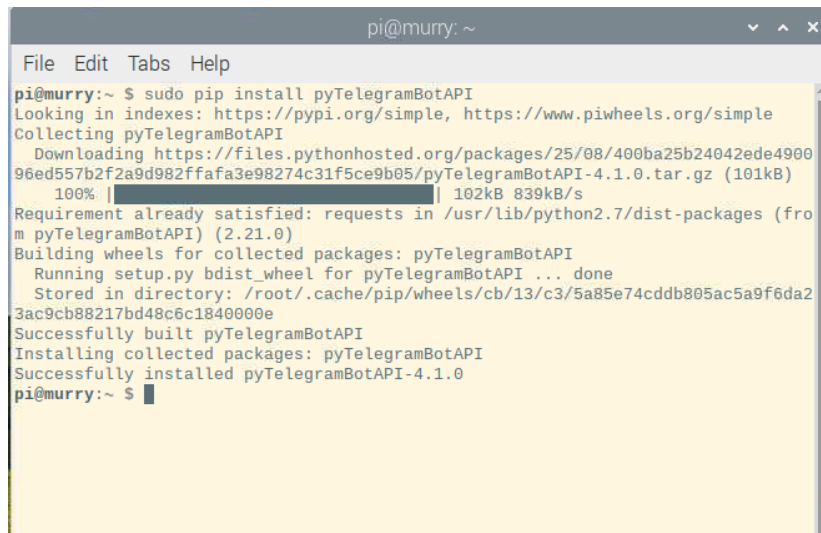
Las ondas ultrasónicas viajan más rápido que la velocidad del sonido audible (es decir, el sonido que los humanos pueden escuchar). Los sensores ultrasónicos tienen dos componentes principales: el transmisor (que emite el sonido mediante cristales

piezoeléctricos) y el receptor (que encuentra el sonido después de que ha viajado hacia y desde el objetivo).

Desarrollo y resultados:

1. Instalación de bibliotecas y paquetes

Como parte esencial para el desarrollo de esta práctica se tuvo que instalar las bibliotecas necesarias para poder usar el bot de Telegram, además de crear un token para conexión mediante Api con el BotFather.



```
pi@murry: ~  
File Edit Tabs Help  
pi@murry:~ $ sudo pip install pyTelegramBotAPI  
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple  
Collecting pyTelegramBotAPI  
  Downloading https://files.pythonhosted.org/packages/25/08/400ba25b24042ede490096ed557b2f2a9d982ffafa3e98274c31f5ce9b05/pyTelegramBotAPI-4.1.0.tar.gz (101kB)  
    100% |#####| 102kB 839kB/s  
Requirement already satisfied: requests in /usr/lib/python2.7/dist-packages (from pyTelegramBotAPI) (2.21.0)  
Building wheels for collected packages: pyTelegramBotAPI  
  Running setup.py bdist_wheel for pyTelegramBotAPI ... done  
  Stored in directory: /root/.cache/pip/wheels/cb/13/c3/5a85e74cddb805ac5a9f6da23ac9cb88217bd48c6c1840000e  
Successfully built pyTelegramBotAPI  
Installing collected packages: pyTelegramBotAPI  
Successfully installed pyTelegramBotAPI-4.1.0  
pi@murry:~ $
```

Imagen 2: Instalación de la biblioteca para el uso de Bots de telegram



Imagen 3: Generación de token para conexión mediante un API a nuestro Murry - Bot

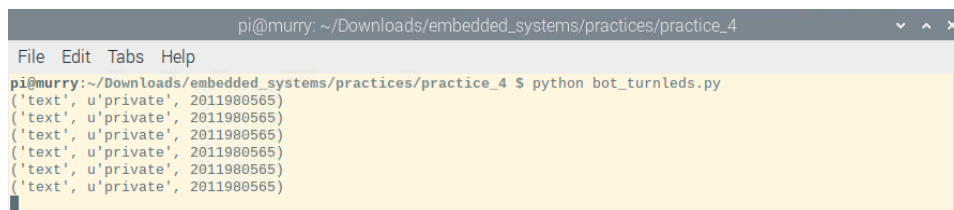
2. Validación de conexión con el bot

Debido a problemas que tuvo el equipo al momento de emplear la forma sugerida de conexión del API, tuvimos omitir la validación mediante el archivo *echo* que se nos sugería en la práctica (Debido a que la carpeta nunca se creaba al momento de instalar mediante pip la biblioteca del bot), sin embargo, la solución que encontramos que realmente es la forma recomendada de emplear el API, es solamente importar la biblioteca en cualquier código, donde a través de unas líneas podemos validar el echo del bot:

```
20 #Main bot function
21 def handleCommand(msg):
22     contentType, chatType, chatId = telepot.glance(msg)
23     print(contentType, chatType, chatId)
```

Imagen 4: Validación de conexión con el bot

A continuación, un ejemplo de conexión una vez ejecutado el código de python:



```
pi@murry: ~/Downloads/embedded_systems/practices/practice_4 $ python bot_turnleds.py
('text', u'private', 2011980565)
('text', u'private', 2011980565)
('text', u'private', 2011980565)
('text', u'private', 2011980565)
('text', u'private', 2011980565)
('text', u'private', 2011980565)
```

Imagen 5: Validación de conexión una vez ejecutado el código en nuestra raspberry pi

3. Ejercicio de prender un led mediante el bot de Telegram

Para este ejercicio lo primero que se realizó fue importar varias bibliotecas asociadas a los puertos de nuestra Raspberry además de la asociada con el bot de Telegram. Cabe destacar que en el caso de la biblioteca Time fue exclusivamente para poder realizar lapsos no tan instantáneos:

A continuación, se muestra la importación de las bibliotecas además del uso del token para la implementación del bot

```
1 import RPi.GPIO as GPIO
2 import time
3 import telepot
4 from telepot.loop import MessageLoop
5
6 GPIO.setmode(GPIO.BCM)
7 LED_PIN = 17
8 GPIO.setup(LED_PIN, GPIO.OUT)
9 bot = telepot.Bot('1AYTCVkJmCDiciEJelw')
```

Imagen 6: Bibliotecas y token

Posteriormente observamos las 2 funciones para el prendido y apagado de los leds además de mandar mensaje a través del bot

```
12 def turnOn(chatId):
13     GPIO.output(LED_PIN, True)
14     bot.sendMessage(chatId, 'The led is ON now!')
15
16 def turnOff(chatId):
17     GPIO.output(LED_PIN, False)
18     bot.sendMessage(chatId, 'The led is OFF now!')
```

Imagen 7: Funciones para prendido y apagado de leds

Por último, observamos la función handlecommand que es la encargada de llamar a cada función anterior en caso de que el usuario le mande un mensaje al bot, además de usar un pre-procesado del mensaje con el objetivo de que no afecte las mayúsculas y minúsculas en el programa.

```
21 def handleCommand(msg):
22     contentType, chatType, chatId = telepot.glance(msg)
23     print(contentType, chatType, chatId)
24
25     #Checking command with the "separator"
26     if contentType != 'text':
27         return
28     message = msg['text']
29
30     if not message.startswith('!'):
31         return
32
33     #Call previous functions
34     command = message[1:].lower()
35     if command == 'on':
36         turnOn(chatId)
37     elif command == 'off':
38         turnOff(chatId)
```

Imagen 8: Función principal para llamada de las funciones de alteración de estados

```
40 if __name__ == '__main__':
41     try:
42         MessageLoop(bot, handleCommand).run_as_thread()
43
44         while 1:
45             time.sleep(10)
46     except KeyboardInterrupt:
47         print("Interrupted by user")
48         pass
49     finally:
50         print("Program stopped")
51         GPIO.cleanup()
```

Imagen 9: Instancia del messageLoop para el uso del bot

Como parte final del ejercicio, a continuación, se muestran los resultados obtenidos usando telegram y el bot:

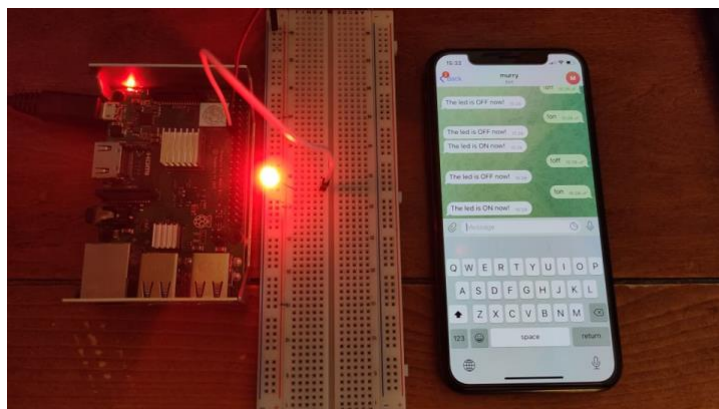


Imagen 10: Prendido del led

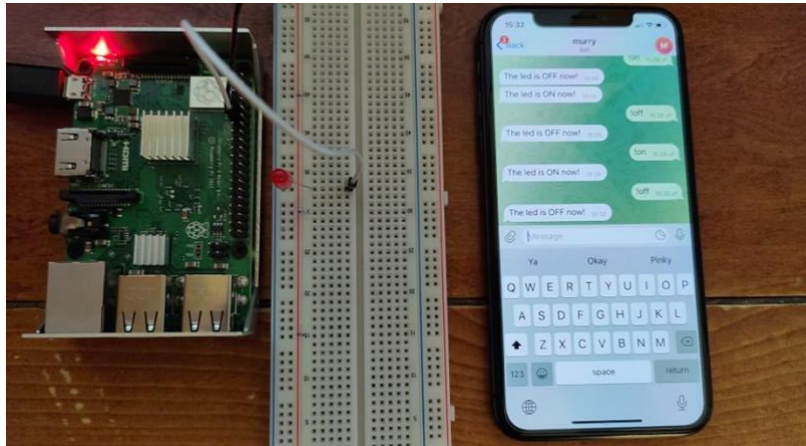


Imagen 11: Apagado del led

4. Ejercicio empleando sensor ultrasónico para alarma mediante Telegram bot

Para este ejercicio lo primero que se realizó fue importar varias bibliotecas asociadas a los puertos de nuestra Raspberry además de la asociada con el bot de Telegram. Cabe destacar que en el caso de la biblioteca Time fue exclusivamente para poder realizar lapsos no tan instantáneos:

A continuación, se muestra la importación de las bibliotecas además del uso del token para la implementación del bot

```

1  import RPi.GPIO as GPIO
2  import telepot
3  import time
4  from telepot.loop import MessageLoop
5
6  #GPIO Mode (BOARD / BCM)
7  GPIO.setmode(GPIO.BCM)
8
9  #set GPIO Pins
10 GPIO_TRIGGER = 17
11 GPIO_ECHO = 27
12
13 #set GPIO direction (IN / OUT)
14 GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
15 GPIO.setup(GPIO_ECHO, GPIO.IN)
16
17
18 bot = telepot.Bot('VkXmCDiciEJelw')
```

Imagen 12: Importación de bibliotecas e instancia del bot mediante el token

Posteriormente y como parte principal de nuestra aplicación tenemos una función destinada a la obtención de la distancia empleando nuestro sensor ultrasónico HCSR04.

A continuación, se muestra el diagrama de conexión con nuestra Raspberry Pi, además del código encargado de llevar a cabo la obtención de la distancia mediante el trigger y el echo de nuestro sensor:

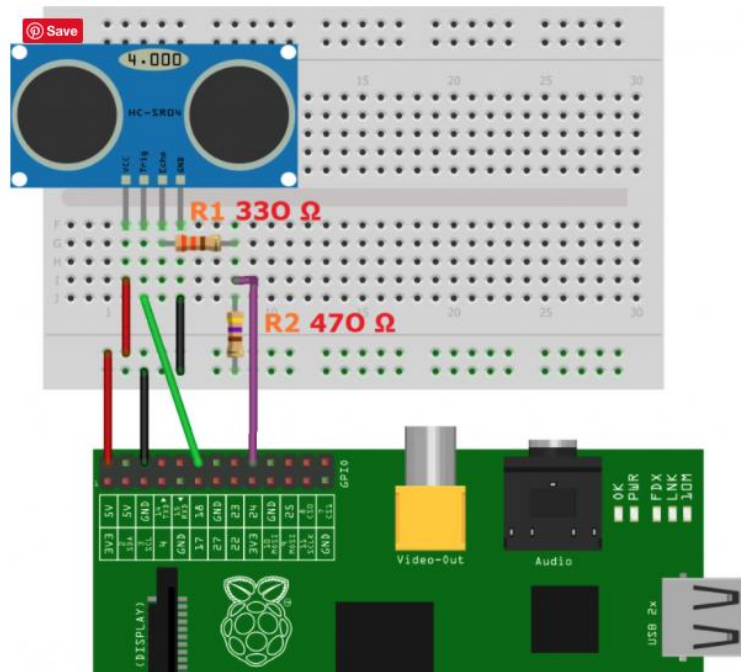


Imagen 13: Diagrama de conexiones para la implementación de nuestro sensor ultrasónico

A continuación, la función de obtención de distancia:

```

21 def getDistance():
22     # set Trigger to HIGH
23     GPIO.output(GPIO_TRIGGER, True)
24
25     # set Trigger after 0.01ms to LOW
26     time.sleep(0.00001)
27     GPIO.output(GPIO_TRIGGER, False)
28
29     StartTime = time.time()
30     StopTime = time.time()
31
32     # save StartTime
33     while GPIO.input(GPIO_ECHO) == 0:
34         StartTime = time.time()
35
36     # save time of arrival
37     while GPIO.input(GPIO_ECHO) == 1:
38         StopTime = time.time()
39
40     # time difference between start and arrival
41     TimeElapsed = StopTime - StartTime
42     distance = (TimeElapsed * 34300) / 2
43
44     return distance

```

Imagen 13: Función de obtención de distancia mediante un sensor ultrasónico

Por último y como parte de comunicación con nuestro bot en telegram, a continuación, se muestra la función principal de la relación entre la distancia y el mandar una “alerta” a nuestro usuario mediante Telegram, cabe destacar que la lógica detrás de esta alerta es que si el sensor detecta algo cerca (Menos de 20 cm) manda una alerta de que hay alguien en la casa


```

48 #Main bot function
49 def handleCommand(msg):
50     contentType, chatType, chatId = telepot.glance(msg)
51     print(contentType, chatType, chatId)
52
53     #Checking command with the "separator"
54     if contentType != 'text':
55         return
56     message = msg['text']
57
58     if not message.startswith('!'):
59         return
60
61     #Call previous functions
62     command = message[1:].lower()
63     if command == 'start':
64         bot.sendMessage(chatId, 'Checking your home!')
65         while True:
66             dist = getDistance()
67             if dist < 20:
68                 bot.sendMessage(chatId, 'Someone is in your home!')
69                 time.sleep(2)
70             #else:
71             #message = 'None in your home, distance ' + str(dist)
72             #bot.sendMessage(chatId, message)
73             #time.sleep(10)
74
75
76 if __name__ == '__main__':
77     try:
78         MessageLoop(bot, handleCommand).run_as_thread()
79
80         while 1:
81             time.sleep(10)
82     except KeyboardInterrupt:
83         print("Interrupted by user")
84         pass
85     finally:
86         print("Program stopped")
87         GPIO.cleanup()

```

Imagen 14: Instancia de nuestro bot mediante un messageLoop el cual a su vez llama a la función handleCommand quien se encarga de llamar cada cierto tiempo a la función de obtención de la distancia

NOTA: es importante mencionar que para hacer funcionar este código es necesario mandar un mensaje a nuestro bot para que empiece a censar, el cual es **!start**

Conclusiones:

Cárdenas Cárdenas Jorge

Hasta las prácticas anteriores habíamos podido interactuar con nuestra Raspberry Pi mediante sus puertos o mediante su interfaz Bluetooth, sin embargo, con ambas implementaciones existe la limitación de que el control no puede ser del todo remoto. La implementación del bot en Telegram permite justamente lograr ese control remoto de la Raspberry sin importar la distancia a la que nos encontremos (como en el caso del control por Bluetooth, donde esta es un factor determinante). No se puede decir que una forma de controlar la Raspberry es mejor o peor que otra, sino que entre ellas se complementan para formar un sistema aún más complejo. La implementación del bot me pareció más sencilla de lo que hubiera creído, y los ejemplos, aunque hasta cierto punto sencillos, me permitieron entender cómo funciona la API de Telegram y como poderla integrar dentro de un proyecto más grande, así como poder seguir interactuando con algunos otros sensores e interfaces.

Garrido Sánchez Samuel Arturo

Para controlar nuestra Raspberry no es necesario manipularla manualmente con instrucciones en Python u otros lenguajes que hagan uso de las interfaces GPIO, sino que además podemos controlarlo remotamente haciendo el uso de un servicio de backend que está proporcionado por Telegram bot. Para hacer uso de esto agregamos la biblioteca teleport. Con ello podemos enviar mensajes de texto al contacto BotFather y este se encargará de mandar la petición http a nuestro servicio y si se cumple las condiciones mandará mensajes o con los mensajes recibidos haremos acciones.

Murrieta Villegas Alfonso

En la presente práctica aprendimos a emplear uno de los API's más conocidos que es el de Telegram para desarrollar o emplear bots, precisamente a lo largo de los 2 ejercicios anteriores es como conocimos diferentes métodos-funciones asociados al bot para obtener o mostrar información de los pequeños sistemas que se hicieron.

Además, y como parte principal de esta práctica aprendimos a usar un API en nuestra Raspberry Pi para de esa forma expandir las funcionalidades de nuestra tarjeta, pues además de disponer de sistemas completos a su vez también podemos ahorrar ese tiempo de desarrollo.

Reza Chavarria Sergio Gabriel

Mediante el manejo de la implementación del bot de telegram nos permite tener un control de los sistemas a partir de un asistente programado. La comunicación con este elemento nos permite el control remoto de diferentes funcionalidades, como en los ejercicios implementados en la práctica.

Mediante la programación y los requerimientos propuestos se pueden manejar diferentes funcionalidades con el manejo de los bots de Telegram.

Valdespino Mendieta Joaquín

En la presente practica pudimos dar paso a una implementación bastante útil, relacionada con IoT, unimos una red social con un elemento de hardware programable, la ventaja de esta red es la capacidad para desarrollar un aplicativo utilizando chatbots, que, mediante una API, podemos entablar una comunicación o canal entre dicho Bot y la raspberry, ejemplificado con los ejercicios realizados, podemos darle escalabilidad o funcionalidad extra a nuestros proyectos actuales o futuros, así como nuevas implementaciones o cambios.

Referencias:

1. GPIO ZERO Docs. *gpiozero*. Recuperado el 3 de Octubre de 2021, de <https://gpiozero.readthedocs.io/en/stable/>
2. Telegram. *Telegram Bot API*. Recuperado el 3 de Octubre de 2021, de <https://core.telegram.org/bots/api>
3. The Humble Code. *How to turn on and off a led using a telegram bot*. Recuperado el 3 de octubre de 2021, <http://thehumblecode.com/blog/how-to-turn-on-and-off-a-led-using-a-telegram-bot-and-raspberry-pi/>
4. Raspberry Pi tutorials. *Using a RaspberrycPi Distance sensor*. Recuperado el 3 de octubre de 2021, <https://tutorials-raspberrypi.com/raspberry-pi-ultrasonic-sensor-hc-sr04/>