Robótica Con Raspberry Y Python: Alphabot

Introduction

... y ALPHABOT

- El objetivo de este curso es proporcionar el lado robótico de Python programando en la misma Raspberry.
- Para ello utilizaremos este robot que dota a la Raspberry de chasis, motor y sensores para trabajar con ellos ¡Hasta una cámara web!!

Requisitos conocimientos

- Los expuestos en el **Curso Python básico**.
- Los expuestos en **RASPBERRY MUY BÁSICO** donde aprenderás a:
- instalar el sistema operativo
- comunicarte con la Raspberry ### Requisitos materiales
- Wifi
- Ordenador
- Alphabot con raspberry

Guía orientativa

https://docs.google.com/presentation/d/e/2PACX-1vQHiZvv1cGHet7eXVy-QcECY4Lj0k0l7ntDi8MevRWHQX-9myA0bfR5IofMeuGZkWD0Hw-Ob-MGoco_/embed?start=trueloop=true&delayms=3000

Tenemos un **grupo Telegram Robótica Educativa en Aragón**,
https://t.me/roboticaeducativaaragon

ALPHABOT

ALPHABOT

1 ALPHABOT

Elegimos este robot como una buena opción para dar a la Raspberry la movilidad y sensores que se espera en un robot. Hay otras alternativas por supuesto, basta con poner las palabras ***Raspberry*** y ***Robot*** en cualquier buscador.

Veremos en VENTAJAS que sirve tanto para Raspberry y Arduino y tiene una buena dotación de sensores, en contra tiene importantes defectos de diseño, esto lo veremos en DESVENTAJAS.

¿Qué incluye este robot?

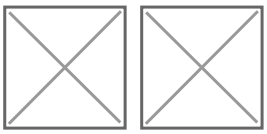
- **Raspberry PI3+** con la opción de añadir un Arduino. Puede ir con uno de los dos o ambos. En este curso sólo trabaremos con la Raspberry.
- **Dos motores** con el L298P driver (¿Qué es eso? Pues parecido al L293 míralo aquí) que proporciona 2A a los motores y tienen diodo de protección para manejarlos con seguridad.
- **Dos sensores de IR de proximidad** no tienen tanta precisión como los sensores de ultrasonidos, pero hacen su función para evitar obstáculos. Hay posibilidad de añadir un sensor de Ultrasonidos (no incorporado pero lo veremos aquí)
- **Sensores de paso** en los motores por lo tanto control de velocidad y de recorrido.
- **Control remoto por IR** con su mando, lo que aumenta nuestra posibilidad creativa.
- **Módulo con 5 sensores sigue-líneas** con un TLC1543 conversor Analógico Digital que lo veremos detenidamente.
- **Brazo robótico** con dos servos que permiten trabajar didácticamente con este importante elemento.
- **Cámara web** que añade una importante gamificación al kit, y al trabajar con la Raspberry en vez de con el Arduino, su control vía web es fácil.

ALPHABOT

1.1 Ventajas

Raspberry, Arduino o ambos

Lo primero que nos gustó es su versatilidad de que sirve tanto para **ARDUINO**, como para la **RASPBERRY**. Tiene un regulador LM2596 que proporciona una tensión estable de 5V para las dos placas. En la figura puedes ver que simplemente cambiando los jumpers amarillos de lugar decides quien actúa el Arduino o la Rasbperry, incluso los dos a la vez !! puedes hacer que los motores vayan con Arduino y los sensores con Raspberry.



Opción Arduino

Para trabajar con el Arduino tiene **un interruptor UART SWITCH** que permite establecer una comunicación serie entre Raspberry y Arduino. Para trabajar con el Arduino sólo tiene en la parte trasera dos conectorres: * 11=Conexión por UART para poner un módulo Bluetooth por ejemplo un JY-MCU HC-06 ¿Qué es eso? mira aquí * 12=Una interface SPI para conectar un módulo wifi NRF24L01

ATENCIÓN: En este curso SÓLO VAMOS A TRABAJAR CON LA RASPBERRY por lo tanto no trabajaremos estos puntos específicos para Arduino. (Una de las ventajas de trabajar con la Raspberry es que ya tiene Bluetooth y Wifi integrados).

Al grano: ¿cuanto cuesta?

Nosotros no somos comerciales, ni intermediarios, sólo somos formadores. Pero para informaros, cuesta aproximadamente unos 100€ se puede conseguir en: * Aliexpres (ojo, que hay modelos **sin** Raspberry o **con** Raspberry) * En la web del fabricante Waveshare

ALPHABOT

1.2 Desventajas

Es importante que las conozcas:

Primera desventaja: LAS PILAS son especiales

- Son del tipo **18650** no son las "*normales AA o AAA*" pero proporcionan 3.7V y más de 1.000mAh cada una lo que asegura la alimentación del robot+raspberry de forma autónoma. Se pueden encontrar en tiendas online por 10€ con cargador incluido. (ojo,hay dos versiones, elegir la de 65mm).
- Encima para complicar las cosas, hay algunas que son falsas.
- **OJO ESTAS PILAS SON PELIGROSAS SI SE CORTOCIRCUITAN O NO SE RESPETA LA POLARIDAD, PUEDEN LLEGAR INCLUSO A EXPLOTAR**. Y para complicarlo, no se ve bien (los símbolos + y - de las 18650 soy muy pequeños) y en Alphabot hay una contradicción, los símbolos de fuera en la placa no coinciden con los símbolos de dentro grabados en el portapilas ¿cuales son los verdaderos?: Los de fuera. Para que quede claro aquí tienes un dibujo:

pub?w=996&h=849

- Algunas están protegidas, pero lo normal es que no. Aquí para ver si la pila es protegida o no.
- *Curiosamente estas baterías forman parte de las baterías de los portátiles, pero manipularlas tiene riesgos ver*

Segunda desventaja: No se puede utilizar la fuente de alimentación de la Raspberry con el chasis de abajo montado

Esto es importante mientras estamos programando este robot, hacer pruebas y depuraciones **sin utilizar las pilas** (son un engorro, sólo hay que ponerlas cuando ya lo tenemos todo depurado).

Se puede utilizar la fuente de alimentación de la Raspberry (output 3.000 mA) pero para conectarlo hay que quitar la placa de abajo

Y por supuesto levantar el robot para que no salga disparado conectado con el cable, que los motores trabajen en vacío y entonces sí que la fuente de alimentación lo puede soportar:

Tercera desventaja: FALOS EN EL DISEÑO, MONTAJE:

```
* Del brazo de robot, el pie no se ajusta bien a la placa y tampoco a la cámara web (en las fotos las flechas amarillas) Ver Chapuzas nº 1, 2 y 3 de [DIY](/diy.md).
* El brazo robot está situado demasiado hacia delante, lo que dificulta la posibilidad de colocar un sensor de Ultrasonidos en la parte delantera, esto lo hablaremos en [este punto](/45-posibilidad-ultrasonidos.md).
* El acceso a la tarjeta microSD es difícil, una manera es utilizando unas pinzas de depilar (ver foto) o desmontando la tapa inferior.
* Como hemos visto anteriormente, no se puede acceder a la alimentación por USB con la tapa
```

```
inferior luego tenemos dos opciones:
    * Alimentar Alphabot con las pilas. (única opción cuando está en movimiento).
    * Desmontar la tapa inferior y alimentarlo por USB. Si elegimos esta opción hay que dejar las ruedas en alto para que los motores trabajen en vacío.
```

Cuarta desventaja: VARIOS

- **La información que hay en Internet** no es muy buena, pero al menos hay una wiki más o menos útil: https://www.waveshare.com/wiki/AlphaBot

- Otro defecto es **la colocación del siguelíneas atrás del sentido de la marcha**, esto lo veremos en el capítulo correspondiente y lo solucionaremos haciendo que vaya hacia atrás, pero claro, la cámara enfoca a la parte trasera y pierde su gracia.

ALPHABOT

1.3 DIY

Este robot es delicado y difícil de montar, intentamos con este manual ayudarte a montarlo si te convence comprarlo pero nosotros no somos comerciales de este robot. O sea, ésto mejor que no:

Pero te queremos animar:

https://giphy.com/embed/xjUGCnG53aCBbfokdS

via GIPHY

Nomenclatura:

- Parte delantera: La que tiene la cámara.
- Parte trasera la que tiene el siguelíneas.

El Paquete de piezas

Encontramos todas estas piezas, destacamos: * Placa de raspberry con microSD, pincho adaptador y fuente de alimentación (no fotografiado) * Tornillería abundante algunos tornillos son minúsculos.

Vamos a por ello (Advertencia: empieza si tienes tiempo por delante):

Los motores

Ponemos primero los dos soportes:

Atornillamos el motor con los tornillos largos

conectamos el motor

repetimos los mismos pasos con el otro motor.

Medidor de velocidad

Ponemos la rueda de agujeros para el medidor de velocidad:

Tiene que ir en este agujero **van muy ajustados** luego no es necesario atornillarlos.
ACONSEJAMOS NO METERLOS AÚN sino después de colocar el brazo robótico

conectamos el sensor:

con la placa

Sensor de siguelíneas

Hay tres tipos de barras, elegimos **siempre las largas** *(no sé para que sirven las pequeñas)*

atornillamos en la parte trasera del robot

Conectamos

y aún no atornillamos, tiene que ir atornillado al final, cuando pongamos la tapa inferior. **El sensor tiene que estar atrapado con el tornillo entre la barra y la tapa inferior**. Esta foto es para que veas cómo tiene que quedar al final, pero aún no lo hagas:

Sensor distancia IR

Colocamos un tornillo de plástico que servirá de arandela aislante pues si no se hace, al atornillar hace un cortocircuito y el sensor no funciona bien:

utilizando los tornillos un poco más largos:

Atornillamos en la parte delantera en los agujeros extremos :

Conectamos

Y por abajo también:

Arduino (opcional)

Si decidimos conectar un Arduino ahora es el momento:

Raspberry

Antes de colocarlo:

Pasamos el cable de la cámara por la ranura de la placa del robot para que salga al exterior:

Ponemos el cable de la cámara, levantamos el plástico negro sin arrancarlo:

Y colocamos el cable, con el lado azul tal como está en la foto y volvemos a apretar el plástico negro para que fije el cable a la Raspberry:

Ahora ya podemos colocar la Raspberry en el zócalo de los GPIO: *(si además tienes puesto un Arduino, queda el Arduino entre la Raspberry y la placa).*

Aprovechamos y ponemos las barras largas para proteger los distintos elementos *(Las 2 barras de la parte delantera pueden ir en esa posición o en los otros dos agujeros más adelantados).*

Brazo robótico

Esta parte es la más difícil !!!

https://giphy.com/embed/3o7abrH8o4HMgEAV9e

via GIPHY

De momento algo sencillote: Conectar los servos **CABLE MARRÓN A GND** pasando los cables por el mismo agujero que están los cables de conexión del sensor de velocidad y el de proximidad:

El servo de abajo tiene que colocarse en esta pieza

entra ajustado pero entra:

colocamos la otra pieza:

Utilizamos los tornillos largos pero no los más largos y estrechos sino este:

Atornillamos:

Para el servo de arriba utilizamos un tornillo de punta pequeño:

Lo atornillamos en los dos lados del servo:

Y lo colocamos con la otra pieza:

CHAPUZA ahora vemos que la pieza de brazo del servo no se ajusta al hueco

La palabra "chapuza" es típico española, y también estos cuchillos albaceteños:

Los españoles estamos entrenados a resolver situaciones chapuzas:

El tornillo es uno en punta con arandela soldada:

CHAPUZA2 la parte que tiene que unir el servo de abajo con la plataforma de la placa tiene que ser con un brazo de servo QUE NO ENTRA:

Pero los maños no nos rendimos:

Esto no sé si está en los libros de ingeniería !!

Aún así **hay que rebajar un poco más en los lados** para que entre bien el brazo del servo blanco, puedes ver en la foto como con el cuchillo se ha rebajado un poco más a los lados para que **la pieza blanca esté lo más prieta a la negra**:.

Bien atornillado por la parte reversa (*Nota: tendrás que agrandar los agujeros de la pieza blanca, un truco es utilizar un tornillo de punta, atornillarlo y destornillarlo*):

Utilizando los tornillos finos y cortos:

Es un buen momento para colocar la cámara. Levantamos la pieza negra sin arrancarla:

Ponemos el cable con el lado azul mirando hacia la cámara como en la foto y volvemos a colocar la pieza negra:

Truco: Al encender el robot, tiene que encenderse un led rojo de la cámara. Si no es así es que has conectado la cinta mal.

CHAPUZA3 Mete la cámara a presión y verás: ¡¡ Es más grande la cámara que el soporte !!. Queda torcido, no es muy estético pero está bien sujeto:

Ahora viene el **PUNTO DÉBIL DE ESTE ROBOT** la unión del brazo robótico con la placa. Ponemos el servo de abajo con la plataforma:

Utilizaremos un tornillo con arandela soldada:

Y bien apretado pero sin reventar el servo, ojo:

Ahora utilizaremos los tornillos más largos con tuerca que lo utilizaremos de arandela de plástico y tuerca

CHAPUZA4: ¿Por qué utilizar la arandela de plástico? porque si no se utiliza, las tuercas hacen cortocircuitos con las soldaduras de la placa, luego necesitamos levantar un poco la plataforma del brazo robótico de la placa:

Atornillamos los 4 (por eso decíamos que no había que poner los sensores de velocidad aún) :

Y ponemos las 4 tuercas por la parte de atrás bien prietas *OJO SE NECESITA* ***MAÑA*** *abstenerse los que no tengan uñas y dedos gordos*:

Ahora ya podemos colocar los sensores de velocidad, que no hace falta atornillarlos pues entran muy ajustados y prietos:

Tiene que quedar que vean bien los agujeros de las ruedas:

Ruedas

Ponemos la rueda loca en la parte trasera:

Atornillamos

Ponemos las ruedas traseras, las pilas:

Acuérdate de poner bien los jumpers amarillos !!:

Y fin !!

https://giphy.com/embed/3o7TKEP6YngkCKFofC

via GIPHY

ALPHABOT

1.4 Configuración

1. Tienes que instalar el sistema operativo Raspbian en la micro tarjeta SD (que ya tiene **Python**) para ello tienes que seguir los pasos de los apuntes de **los apuntes Raspberry muy básico**. Concrétamente el capítulo 3.
2. Una vez instalado tienes que conectar la Raspberry a la wifi, para ello sigue los pasos marcados en el capítulo 4.
3. Recomendable pero no obligatorio, es aprender a configurar el sistema operativo, comunicarte con él via texto por SSH, cambiar el usuario, contraseña, esto sí que es obligatorio: tienes que aprender a apagar ;).
4. Luego tienes que comunicarte en este curso VIA GRÁFICAMENTE por VNC lo tienes explicado en el capítulo 8.

Cómo ejecuto un programa

Vía VNC de forma gráfica, creas un fichero con extensión .py le das dos cliks y ya está !! Se abre el editor de Python para que escribas tus programas. (pues Raspbian tiene Python de forma nativa). Se ejecuta con el botón Play (redondo verde de la figura) y se para con el rojo. Esta será la forma de trabajar en este curso.

UNA VEZ REALIZADO ESTOS PASOS YA PODEMOS PASAR A REALIZAR NUESTRO PRIMER PROGRAMA.

> Nota: También se puede hacer de forma textual con el protocolo SSH ejecutando la orden python. Por ejemplo: Tenemos un programa llamado miprograma.py en la carpeta Aphpabot de la Raspberry luego las instrucciones serían en el terminal ssh: cd ~/AlphaBot/
> sudo python miprograma.py

Curiosidad ¿se puede hacer desde Internet?

Totos estos pasos se entiende que lo haces desde **LA RED LOCAL** accediendo a la Raspberry por una IP fija tal y como hemos explicado en los enlaces del principio de esta misma página, pero también lo podemos hacer desde Internet siguiendo estos pasos.

O sea: ¿que puedo manejar mi Alphabot desde cualquier lugar del mundo? Si

https://giphy.com/embed/3o7TKME5YAAn6LKQjm

via GIPHY

ALPHABOT

1.5 Pensamiento computacional

¿Dónde se encaja este robot?¿se puede comparar este robot con otros robots de otros cursos que hacemos desde CATEDU?

Esta es la hoja de ruta, no se tiene que tomar al pie de la letra, pero intenta ayudar al profesorado que tenga una visión global de tanta oferta:

https://view.genial.ly/5c546dc28805472c3451861a

Como se puede ver ALPHABOT es LA ALTERNATIVA ROBÓTICA A PYTHON. En programación con código (es decir, no gráfica con bloques) tenemos Python y Java, y en robótica Arduino y Alphabot.

El precio dada las características que tiene (rasperry, cámara, brazo robótico, motores ...) está razonablemente bien:

ALPHABOT

1.6 Vaya programación cutre!

Si eres un programador, te recomiendo que no sigas el curso, yo no soy un experto y seguro que estoy cometiendo muchos errores.

Hay dos formas de programar: sencilla pero no profesional y profesional pero no sencilla,igual que los coches, si funcionan bien, los dos llegan al destino.

Como este curso está orientado para ayudar a los docentes a aplicar la robótica en el aula, preferimos ser sencillos pero que se entienda. Se admite comentarios, propuestas y los programas están en este Github para mejorar depuraciones.

Puedes optar por la programación profesional, en la página https://www.waveshare.com/wiki/AlphaBot tienes Software demo que te puedes descargar y puedes hacer las mismas propuestas, está programado en Python y utilizando programación orientada a objetos:

ALPHABOT

1.7 GPIO

GPIO

Vamos a recordar lo que vimos aquí, dos cosas:

- Estos son los pines GPIO con la numeración BCM:

- Y sobre todo **RECUERDA** : Están diseñados para 3.3V sólo proporcionan 3mA cada pin luego NO conectes diréctamente componentes de 5V ni que consuman más corriente o de lo contrario ESTROPEARÁS LA RASPBERRY DE FORMA IRREVERSIBLE, o sea, directamente sólo LEDs con una resistencia de mínimo 1.1K tal y como vimos aqui, todo lo demás a través de chips drivers.

Librería RPI.GPIO

Necesitamos una librería GPIO que Raspbian lo tiene por defecto, pero por si acaso ejecuta estas instrucciones:

> sudo apt-get install python-dev
>
> sudo apt-get install pyton-rpi.gpio

Normalmente te dirá que las tienes instaladas en su última versión.
Para utlizar la librería, simplemente tenemos que poner esta instrucción:
import RPi.GPIO as GPIO

GPIO.setmode y GPI.setup

Hay dos formas de utilizar la numeración de las GPIO, respetando la misma numeración que los pines de la placa, entonces la instrucción que tenemos que poner en nuestros programas es:
GPIO.setmode(GPIO.BOARD)
o utilización de la numeración BCM:
GPIO.setmode(GPIO.BCM)
nosotros elegiremos esta última por ser más sencilla, aunque tiene la desventaja de que si cambian en el futuro la numeraciones en los BCM nuestro programa no servirá.

Una vez definido qué numeración usamos, tenemos que especificar en nuestro programa si tal GPIO es entrada o salida, por ejemplo la siguiente instrucción define el GPIO número 4 como entrada (7 en numeración BOARD):
GPIO.setup(4, GPIO.IN)

Ejemplo de utilización de la librería RPI.GPIO

El siguiente ejemplo enciende un LED puesto en el GPIO 4, durante 2 segundos

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(4, GPIO.OUT) ## GPIO 4 como salida
GPIO.output(4,True) ##encendemos
time.sleep(2)        ## espera 2 segundos
GPIO.output(4,False)  ##APAGAMOS
```

ALPHABOT

1.8 PWM

¿Qué es?

Para entender el funcionamiento de los motores, primero tenemos que hablar de esta señal especial.

La RASPBERRY igual que el ARDUINO (ver cap 2.4 curso Arduino) no es capaz de generar señales ANALÓGICAS. Un truco es generar una señal cuadrada de pulsación variable PWM (Pulse Width Modulation, Modulación de Ancho de Pulso) de esta forma "simula" una señal analógica.

¿Cómo se genera utilizando PYTHON y los pines GPIO?

Se realiza primero creando una variable especial PWM con la instrucción:

p = GPIO.PWM(canal, frecuencia) donde canal es el número de pin GPIO donde queremos generar la señal PWM de frecuencia dada en Hz

Con esto está creado pero no genera los pulsos, para eso se hace con la instrucción:

p.start(dc) donde dc=duty cycle en % es decir desde 0.0 hasta 100.0, por ejemplo en la figura anterior, la a) sería dc=25 el b) dc=50 y la gráfica c) sería un dc=75.

Para parar **p.stop()**

Please! ¿Un ejemplo?

Claro, vamos a ver un ejemplo sencillo que es encender un LED cada 2 segundos en el GPIO número 12:

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(12, GPIO.OUT)  #definimos el GPIO12 como salida

p = GPIO.PWM(12, 0.5)
p.start(50)
```

ALPHABOT

1.9 Kit de prestamo

Si haces el curso, está disponible este préstamo para que lo puedas hacer: pub?w=960&h=720

Movimiento

Movimiento

2 Movimiento

El primer contacto con este robot va a ser controlar el movimiento.

> Nomenclatura: Hacia delante es donde está la cámara.

https://giphy.com/embed/J3eLH0ybAZy8g

via GIPHY

Movimiento

2.1 Motores

Evidentemente los sensores, motores, etc... estarán conectados de alguna manera a algún pin de la GPIO (¿qué es eso de GPIO? Pues eso es que no te has leido esto).

| Interfaces | Puertos GPIO de la Raspberry Pi | |------------|-------------| | IN1 | P12 | | IN2 | P13 | | ENA | P6 | | IN3 | P20 | | IN4 | P21 | | ENB | P26 |

Luego una de las primeras líneas que hay que poner en nuestros programas es traducir esos números a letras para que sea más facil utilizarlos en el código, y definir esos pines como pines de salida que van a gobernar a los motores:

```
import RPi.GPIO as GPIO

IN1=12;IN2=13;ENA=6;IN3=20;IN4=21;ENB=26

GPIO.setmode(GPIO.BCM);GPIO.setwarnings(False)
GPIO.setup(IN1,GPIO.OUT);GPIO.setup(IN2,GPIO.OUT);GPIO.setup(IN3,GPIO.OUT);GPIO.setup(IN4,GPIO
.OUT)
GPIO.setup(ENA,GPIO.OUT);GPIO.setup(ENB,GPIO.OUT)
```

¿Qué significan IN1 IN2 IN3 IN4 ?

|IN1| IN2| IN3| IN4| Descripción| |---|----|----|----|------------| |1 |0 |0 |1 |Motores hacia delante| |0 |1 |1 |0 |Motores hacia atrás| |0 |0 |0 |1 |Giro derecha| |1 |0 |0 |0 |Giro izquierda| |0 |0 |0 |0 |Stop|

¿Y qué significa ENA ENB?

ENA y ENB es la velocidad de los motores A y B respectivamente.

Su valor tiene que ser analógico pero los GPIO son digitales, así que tienen que ser señales PWM.

Si vamos a poner una frecuencia de 500Hz y una velocidad media, el código que tenemos que poner al principio de nuestro programa es:

```
PWMA = GPIO.PWM(ENA,500);PWMB = GPIO.PWM(ENB,500)
PWMA.start(50);PWMB.start(50)
```

Bueno, pero ... ¿cómo son las conexiones?

En el AlphaBot están conectados los pines IN1 IN2 IN3 IN4 ENA ENB en los pines de un chip L298P que hace de driver a los motores (nunca conectes un motor a un GPIO de la Raspberry ya lo sabes)

Vale... ¿Y cómo se utiliza?

Podemos definir en nuestros programas unas funciones para simplificar código para utilizar los motores hacia delante, detrás y giros:

```
def FORDWARD():

GPIO.output(IN1,GPIO.HIGH);GPIO.output(IN2,GPIO.LOW);GPIO.output(IN3,GPIO.LOW);GPIO.output(IN4
,GPIO.HIGH)
def BACKWARD():

GPIO.output(IN1,GPIO.LOW);GPIO.output(IN2,GPIO.HIGH);GPIO.output(IN3,GPIO.HIGH);GPIO.output(IN
4,GPIO.LOW)
def LEFT():

GPIO.output(IN1,GPIO.LOW);GPIO.output(IN2,GPIO.LOW);GPIO.output(IN3,GPIO.LOW);GPIO.output(IN4,
GPIO.HIGH)
def RIGHT():

GPIO.output(IN1,GPIO.HIGH);GPIO.output(IN2,GPIO.LOW);GPIO.output(IN3,GPIO.LOW);GPIO.output(IN4
,GPIO.LOW)
def STOP():
```

```
GPIO.output(IN1,GPIO.LOW);GPIO.output(IN2,GPIO.LOW);GPIO.output(IN3,GPIO.LOW);GPIO.output(IN4,
GPIO.LOW)
```

Movimiento

2.2 Fichero VARIABLES.py

Debido a que vamos a utilizar varias variables que serán comunes a varias librerías que también vamos a crear, vamos a crear un fichero común a todos, de momento será este:

Cuando queramos incorporar estas variables pondremos esta instrucciónde Python **from VARIABLES import ***

VARIABLES.py

```
import RPi.GPIO as GPIO

DataMotorR = 7
DataMotorL = 8

IN1=12
IN2=13
ENA=6
IN3=20
IN4=21
ENB=26

##############CONFIGURACION GPIO ENTRADAS SALIDAS ####
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(IN1,GPIO.OUT)
GPIO.setup(IN2,GPIO.OUT)
GPIO.setup(IN3,GPIO.OUT)
GPIO.setup(IN4,GPIO.OUT)
GPIO.setup(ENA,GPIO.OUT)
GPIO.setup(ENB,GPIO.OUT)

GPIO.setup(DataMotorR,GPIO.IN)
```

```
GPIO.setup(DataMotorL,GPIO.IN)



########################## VELOCIDAD DE LOS MOTORES
PWMA = GPIO.PWM(ENA,500)
PWMB = GPIO.PWM(ENB,500)
```

Movimiento

2.3 Libreria MOVIMIENTOS.py

Para simplificar nuestros programas podemos hacer una librería propia.

Esta librería la vamos a llamar **MOVIMIENTOS.py** y su contenido sería lo visto en las páginas anteriores, añadiendo las variables definidas en **VARIABLES.py**:

```cpp+lineNumbers:true import RPi.GPIO as GPIO

from VARIABLES import *

####################FUNCIONES
def FORDWARD(vel): GPIO.output(IN1,GPIO.HIGH) GPIO.output(IN2,GPIO.LOW) GPIO.output(IN3,GPIO.LOW) GPIO.output(IN4,GPIO.HIGH) PWMA.start(vel) PWMB.start(vel)

def BACKWARD(vel): GPIO.output(IN1,GPIO.LOW) GPIO.output(IN2,GPIO.HIGH) GPIO.output(IN3,GPIO.HIGH) GPIO.output(IN4,GPIO.LOW) PWMA.start(vel) PWMB.start(vel)

def LEFT(vel): GPIO.output(IN1,GPIO.LOW) GPIO.output(IN2,GPIO.LOW) GPIO.output(IN3,GPIO.LOW) GPIO.output(IN4,GPIO.HIGH) PWMA.start(vel) PWMB.start(vel)

def RIGHT(vel): GPIO.output(IN1,GPIO.HIGH) GPIO.output(IN2,GPIO.LOW) GPIO.output(IN3,GPIO.LOW) GPIO.output(IN4,GPIO.LOW) PWMA.start(vel) PWMB.start(vel)

def STOP(): GPIO.output(IN1,GPIO.LOW) GPIO.output(IN2,GPIO.LOW) GPIO.output(IN3,GPIO.LOW) GPIO.output(IN4,GPIO.LOW) ```
```

Movimiento

2.4 Baile

Vamos a realizar un sencillo programa para romper el hielo, unos movimientos delante, atrás, derecha, izquierda y paro utilizando la librería anterior:

https://www.youtube.com/embed/WAycuDaKB0Q

Solución

- Ponemos el fichero MOVIMIENTOS.py que hemos visto en la misma carpeta que vamos a crear este programa.
- En este programa importamos la librería de MOVIMIENTOS.py.
- Vamos llamando a las distintas funciones de movimientos, fijamos la velocidad al 50%, insertando entre ellas un tiempo de retraso de la librería time de 1 segundo.

¿Te atreves? Sino, mira la solución:

%accordion%Solución%accordion%

Fichero 2-6-Baile.py

```
import RPi.GPIO as GPIO
import time

import MOVIMIENTOS

MOVIMIENTOS.FORDWARD(50)
time.sleep(1)
MOVIMIENTOS.BACKWARD(50)
time.sleep(1)
MOVIMIENTOS.LEFT(50)
time.sleep(1)
MOVIMIENTOS.RIGHT(50)
```

```
time.sleep(1)
MOVIMIENTOS.STOP()
```

%/accordion%

Movimiento

2.5 Movimientos con tecla

Ahora vamos a hacer lo mismo, pero gobernado por el teclado:

- PARAR = tecla ESPACIO
- ADELANTE=FORDWARD = f
- ATRAS=BACKWARD = b
- DERECHA=RIGHT = r
- IZQUIERDA=LEFT = l

https://www.youtube.com/embed/fb6w5yQB_AM

Solución

- Ponemos el fichero MOVIMIENTOS.py que hemos visto en la misma carpeta que vamos a crear este programa.
- En este programa importamos la librería de MOVIMIENTOS.py.
- Vamos llamando a las distintas funciones de movimientos según la tecla pulsada, fijamos la velocidad al 30% para que nos de tiempo de gobernarlo, por pantalla va saliendo el mensaje del estado.
- Todo dentro de un bucle de manera que si pulsamos la tecla espacio sale del buble no sin antes parar el robot.

¿Te atreves a hacerlo tú solo? sino, mira la solución:

%accordion%Solución%accordion%

Fichero 2-7-Movimientos-Teclas.py

```
import RPi.GPIO as GPIO
import time

import MOVIMIENTOS
```

```
print ('TECLAS ¡en minúscula!:\nPARAR = tecla ESPACIO\nADELANTE=FORDWARD = f\nATRAS=BACKWARD =
b\nDERECHA=RIGHT = r\nIZQUIERDA=LEFT = l')
tecla='x'
while tecla!=' ':
    tecla = input('\nPresiona una tecla y después enter : ')
    if tecla != ' ':
        print ('\nHas presionado ', tecla)
        if tecla=='f':
            print ('\nadelante')
            MOVIMIENTOS.FORDWARD(30)
        if tecla=='b':
            print ('\natrás')
            MOVIMIENTOS.BACKWARD(30)
        if tecla=='r':
            print ('\nderecha')
            MOVIMIENTOS.RIGHT(30)
        if tecla=='l':
            print ('\nizquierda')
            MOVIMIENTOS.LEFT(30)

    else:
        print ('\nFin, has apretado STOP')
        MOVIMIENTOS.STOP()
```

%/accordion%

Control velocidad

3 Control velocidad

Control velocidad

3.1 ¿Cómo funciona?

Las ruedas tienen un disco con agujeros, una parte es un diodo emisor de IR y el otro es un sensor fotoeléctrico tipo WYC-H206 que detecta los agujeros:

Están conectados a los siguientes GPIO: * Motor derecha GPIO7 * Motor izquierda GPIO8

Si te fijas en el esquema anterior, las resistencias, están con la configuración **PULL-UP** (aquí para saber + en el curso Arduino) ¿qué significa esto? pues que van al revés, cuando el circuito está encendido, o sea detecta agujero, estado ON transmite un 0 lógico, y al revés, cuando está apagado OFF transmite un 1 lógico, lo puedes ver mejor en estas fotografías:


Control velocidad

3.2 Prueba velocidad

En el siguiente vídeo vemos como cuando el sensor está encendido, el programa detecta un 0 y si el sensor está apagado, el programa detecta un 1:

https://www.youtube.com/embed/anWM7Y54u98

Fichero Pruebasensorvelocidad.py

El programa es el siguiente:

```
import RPi.GPIO as GPIO

DataMotorR = 7
DataMotorL = 8

GPIO.setmode(GPIO.BCM)

GPIO.setup(DataMotorR,GPIO.IN)
GPIO.setup(DataMotorL,GPIO.IN)



for i in range(100000):
    print('\nMotor derecha   :',GPIO.input(DataMotorR))
    print('\nMotor izquierda :',GPIO.input(DataMotorL))
```

Segundo test de contador

En el segundo vídeo vídeo vemos como un simple contador puede detectar el paso del 1 al 0:

https://www.youtube.com/embed/Xm1aZvEp9fE

El programa es el siguiente:

Fichero Pruebasensorvelocidad-2.py

```cpp+lineNumbers:true

import RPi.GPIO as GPIO

DataMotorR = 7 DataMotorL = 8

GPIO.setmode(GPIO.BCM)

GPIO.setup(DataMotorR,GPIO.IN) GPIO.setup(DataMotorL,GPIO.IN)

contador=0 repetido=0 num = 100 while (contador<num):
if((GPIO.input(DataMotorR)==1)and(repetido==0)): contador=contador+1 print('\nContador
:',contador) repetido=1 if(GPIO.input(DataMotorR)==0): repetido=0 ```
```

Control velocidad

3.3 Variables.py

Añadimos ahora las varialbes de paso siguientes a este fichero

VARIABLES.py

import RPi.GPIO as GPIO

DataMotorR = 7 DataMotorL = 8

IN1=12 IN2=13 ENA=6 IN3=20 IN4=21 ENB=26

########CONFIGURACION GPIO ENTRADAS SALIDAS

GPIO.setmode(GPIO.BCM) GPIO.setwarnings(False) GPIO.setup(IN1,GPIO.OUT) GPIO.setup(IN2,GPIO.OUT) GPIO.setup(IN3,GPIO.OUT) GPIO.setup(IN4,GPIO.OUT) GPIO.setup(ENA,GPIO.OUT) GPIO.setup(ENB,GPIO.OUT)

GPIO.setup(DataMotorR,GPIO.IN) GPIO.setup(DataMotorL,GPIO.IN)

########################## VELOCIDAD DE LOS MOTORES

PWMA = GPIO.PWM(ENA,500) PWMB = GPIO.PWM(ENB,500) ```

Control velocidad

3.4 MOVIMIENTOSPASO.py

Vamos a hacer una pequeña función donde le pasemos dos argumentos por cada motor (en total 4 argumentos): velocidad y número de pasos. Tiene que hacer:

- Si el número de pasos es positivo va hacia delante el motor.
- Si el paso es negativo es que el motor va hacia atrás.
- Los motores funcionarán con la velocidades dadas en los argumentos.
- En total 4 argumentos tiene la función, dos para cada motor R y L: *velR,numR,velL,numL* donde *vel* es la velocidad del motor y *num* el número de pasos.

¿Te atreves? Sino, mira la solución:

%accordion%Solución%accordion%

Fichero MOVIMIENTOSPASO.py

```
import RPi.GPIO as GPIO
import time
import MOVIMIENTOS
from VARIABLES import *

######################################################################
####################FUNCIóN AMBOS#####################################
######################################################################
def BOTH(velR,numR,velL,numL):
    repetidoR=0
    repetidoL=0
    if (numR>0):
        GPIO.output(IN1,GPIO.HIGH)
        GPIO.output(IN2,GPIO.LOW)
    else:
        numR=-numR
        GPIO.output(IN1,GPIO.LOW)
        GPIO.output(IN2,GPIO.HIGH)
```

```
    if (numL>0):
         GPIO.output(IN4,GPIO.HIGH)
         GPIO.output(IN3,GPIO.LOW)
    else:
        numL=-numL
        GPIO.output(IN4,GPIO.LOW)
        GPIO.output(IN3,GPIO.HIGH)
    contadorR=0
    contadorL=0
    while ((contadorR
```

Control velocidad

3.5 Movimientos con paso

Vamos a hacer un programa para utilizar la librería anterior MOVIMIENTOSPASO.py y gobernado por el teclado *numérico*:

- PARAR = tecla 5
- ADELANTE=FORDWARD = 8
- ATRAS=BACKWARD = 2
- DERECHA=RIGHT = 6
- IZQUIERDA=LEFT = 4

Fijaremos de antemano las velocidades y el paso a 50 y 10 por ejemplo.

https://www.youtube.com/embed/n0tOHIHl0Rs

Solución

- Ponemos las librerías fichero MOVIMIENTOS.py y MOVIMIENTOSPASO.py en la misma carpeta que vamos a crear este programa y las incorporamos en el programa con **import**.
- Vamos llamando a las distintas funciones de movimientos según la tecla pulsada, fijamos la velocidad al 50%, por pantalla va saliendo el estado de los contadores.
- Todo dentro de un bucle de manera que si pulsamos la tecla espacio sale del buble no sin antes parar el robot.

%accordion%Solución%accordion%

Fichero 3-5-Movimientos-paso.py

```
import RPi.GPIO as GPIO
import time

import MOVIMIENTOS
import MOVIMIENTOSPASO

velR=50
numR=10
```

```
velL=50

numL=10

print ('TECLAS ¡en minúscula!:\nPARAR = tecla 5\nADELANTE=FORDWARD = 8\nATRAS=BACKWARD =
2\nDERECHA=RIGHT = 6\nIZQUIERDA=LEFT = 4')
tecla='x'
while tecla!='5':
    tecla = input('\nPresiona una tecla y después enter : ')
    if tecla != '5':
        print ('\nHas presionado ', tecla)
        if tecla=='8':
            print ('\nadelante')
            MOVIMIENTOSPASO.BOTH(velR,numR,velL,numL)
        if tecla=='2':
            print ('\natrás')
            MOVIMIENTOSPASO.BOTH(velR,-numR,velL,-numL)
        if tecla=='6':
            print ('\nderecha')
            MOVIMIENTOSPASO.BOTH(velR,-numR,velL,numL)
        if tecla=='4':
            print ('\nizquierda')
            MOVIMIENTOSPASO.BOTH(velR,numR,velL,-numL)

    else:
        print ('\nFin, has apretado STOP')
        MOVIMIENTOS.STOP()
```

%/accordion%

Sensor obstáculos IR

Sensor obstáculos IR

4 Sensor obstáculos IR

Sensor obstáculos IR

4.1 ¿Cómo funciona?

Lo primero que tenemos que hacer es ajustar su sensibilidad con el potenciómetro que tiene de tal manera que detecte de forma correcta un obstáculo:

https://www.youtube.com/embed/_89J0WtOqgQ

ES MUY SENSIBLE el punto está justo cuando se apaga:

https://www.youtube.com/embed/aJonvDOttgU

Tiene un led de infrarrojos que cuando hay un obstáculo delante del sensor, refleja esta radiacción y el otro LED (realmente no se podría decir que es LED pues no emite, se tendría que llamar unión PN semiconductor), es un sensor receptor que lo detecta. Un chip comparador LM393 detecta la señal y su sensibilidad se ajusta con el potenciómetro:

La salida DOUT de cada LM393 están conectados a los siguientes puertos GPIO:

- Derecha GPIO 16
- Izquierda GPIO 19

Una vez más vemos que la resistencias del sensor están tipo PULL-UP luego cuando detecta emitirá un 0 lógico y cuando no detecta emitirá un 1 lógico.

Sensor obstáculos IR

4.2 Test

Ejecutamos este pequeño programa:

Fichero 4-2-TestObstaculoIR.py

```
import RPi.GPIO as GPIO
import time

DR = 16
DL = 19

GPIO.setmode(GPIO.BCM)

GPIO.setup(DR,GPIO.IN)
GPIO.setup(DL,GPIO.IN)

for i in range(10000):
     print('\nSensor derecha :',GPIO.input(DR))
     print('\nSensor izquier :',GPIO.input(DL))
```

Y podemos ver en el vídeo que emite un 0 cuando detecta un obstáculo:

https://www.youtube.com/embed/oehMTYNSPHA

Sensor obstáculos IR

4.3 Variables.py

Añadimos más variables a VARIABLES.py

Los marcados en negrita:

---

import RPi.GPIO as GPIO

#####MOTORES

IN1=12 IN2=13 ENA=6 IN3=20 IN4=21 ENB=26

##SENSOR VELOCIDAD MOTORES

DataMotorR = 7 DataMotorL = 8

##########SENSOR OBSTACULOS IR DR = 16 DL = 19

########CONFIGURACION GPIO ENTRADAS SALIDAS

GPIO.setmode(GPIO.BCM) GPIO.setwarnings(False)

####################MOTORES

GPIO.setup(IN1,GPIO.OUT) GPIO.setup(IN2,GPIO.OUT) GPIO.setup(IN3,GPIO.OUT)
GPIO.setup(IN4,GPIO.OUT) GPIO.setup(ENA,GPIO.OUT) GPIO.setup(ENB,GPIO.OUT)

############## VELOCIDAD DE LOS MOTORES

PWMA = GPIO.PWM(ENA,500) PWMB = GPIO.PWM(ENB,500)

############SENSOR VELOCIDAD MOTORES

GPIO.setup(DataMotorR,GPIO.IN) GPIO.setup(DataMotorL,GPIO.IN)

####################SENSORES OBSTACULOS IR GPIO.setup(DR,GPIO.IN) GPIO.setup(DL,GPIO.IN)

Sensor obstáculos IR

4.4 Roomba

Bueno ahora hay que hacer el típico programa que evite los obstáculos. Ya sabes detectar los obstáculos, ya sabes controlar el movimiento ¿a qué esperas?

https://giphy.com/embed/kviKlWqjoBdCw

via GIPHY

https://www.youtube.com/embed/5LvBqHv1wM4

Solución

La solución no es única, una propuesta es hacerlo con las librerías que hemos aprendido: *

Ponemos las librerías fichero MOVIMIENTOS.py y MOVIMIENTOSPASO.py en la misma carpeta que vamos a crear este programa y las incorporamos en el programa con **import**. * También incorporamos las variables definidas en **VARIABLES.py** * Si no detecta nada, que sigua hacia delante. * Si detecta algo, según los dos o uno, que de unos pasos atrás y que gire.

%accordion%Solución%accordion%

Fichero Roomba.py

```
import RPi.GPIO as GPIO
import time
from VARIABLES import *
import MOVIMIENTOS
import MOVIMIENTOSPASO

while True:
    sensorR= not (GPIO.input(DR))
    sensorL= not (GPIO.input(DR))
    if not(sensorR and sensorL):
         MOVIMIENTOS.FORDWARD(50)
```

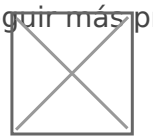
```
if (sensorR and sensorL):
    MOVIMIENTOSPASO.BOTH(50,-10,50,-10)
if (sensorR and not(sensorL)):
    MOVIMIENTOSPASO.BOTH(50,-5,50,-5)
    MOVIMIENTOSPASO.BOTH(40,-5,40,5)
if (not(sensorR) and (sensorL)):
    MOVIMIENTOSPASO.BOTH(50,-5,50,-5)
    MOVIMIENTOSPASO.BOTH(40,5,40,-5)
```

%/accordion%

Sensor obstáculos IR

4.5 Posibilidad ultrasonidos

Se puede conseguir más precisión añadiendo un tercer sensor y mucho más preciso: El **sensor de Ultrasonidos.**



Mira en esta página para saber cómo se utiliza con el Arduino.

Básicamente se emite un pulso por el pin **Trigger**, él emite una señal de 40kHz y según el eco recibido saca por **Output** un pulso cuyo ancho es proporcional a la distancia:

Conexión en alphabot

En Alphabot se conectaría los cables en el conector blanco de abajo y el ultrasonidos con unos tornillos en los dos agujeros de la parte delantera:

El sensor de ultrasonidos tiene que estar adaptado a los cables compatibles con la Shield Grove de Arduino. Por ejemplo este:

CHAPUZA : No tiene su orden standard GND-Vcc-DATA1-DATA2 sino es GND-DATA1-DATA2-VCC o sea GND-Trg-Echo-5V luego habría que hacer alguna chapucilla de intercambiar cables, habría que elegir unos cables largos, cortarlos e intercambiarlos:

Para sujetar el sensor ultrasonidos al chasis habría que comprar un soporte:

Otra opción es quitar la cámara y poner el sensor de ultrasonidos:

EL KIT DE CATEDU NO PROPORCIONA EL SENSOR ULTRASONIDOS

Bueno, si aún así me decido ponerlo ¿cómo se programa?

Muy fácil, el conector blanco de abajo está conectado con los siguientes GPIO:

- Echo en el GPIO 5
- Trigger en el GPIO 17

Por lo tanto, viendo la teoría, una posible función en código Python para utilizarlo sería:

- Emitir un pulso alto por TRIG durante 15 microsegundos.
- Esperar el pulso alto de ECHO
- Cronometrar el pulso alto de ECHO
- La distancia será velocidad por tiempo o sea: la diferencia el tiempo del pulso ECHO multiplicado por la velocidad del sonido y dividido por 2 pues es el recorrido del sonido ida y vuelta.

```
TRIG = 17
ECHO = 5

GPIO.setup(TRIG,GPIO.OUT,initial=GPIO.LOW)
GPIO.setup(ECHO,GPIO.IN)
```

```
def Distance():
    GPIO.output(TRIG,GPIO.HIGH)
    time.sleep(0.000015)
    GPIO.output(TRIG,GPIO.LOW)
    while not GPIO.input(ECHO):
        pass
    t1 = time.time()
    while GPIO.input(ECHO):
        pass
    t2 = time.time()
    return (t2-t1)*34000/2
```

Control remoto

5 Control remoto

Control remoto

5.1 NEC

Alphabot tiene un receptor LFN0038K compatible con el protocolo estandard NEC y el mando emisor que acompaña a este kit también es compatible con él. Encima, para complicarnos más las cosas, tiene configuración PULL-UP como el resto de sensores de este robot, por lo que si recibe del mando a distancia un 1 lógico, él transmite un 0 y viceversa.

Una señal con el protocolo NEC es una especie de señal PWM donde un '0' es es un ciclo de 1.125ms la emisión de una señal de 0.565ms y un '1' lógico o mismo pero en un intervalo de 2.25ms, se ve mejor con un dibujo:

El protocolo establece que primero se emite 9ms de una señal alta, y luego 4.5ms de señal baja para empezar el envío de una señal de 8 bits, empezando por el bit más bajo LSB hasta el más alto MSB y luego su complemento. Todo en una señal de 38kHz donde se modula primero la dirección y luego el comando. Mejor con un dibujo ¿no?

Cada comando se envía una sola vez al menos que mantengas el botón pulsado más de 110ms. El formato de duplicación es según este dibujo:

Control remoto

5.2 VARIABLES.py y NEC.py

El sensor IR está unido al GPIO número 18 luego añadimos en el fichero variables.py las siguientes líneas:

> IR = 18
>
> GPIO.setup(IR,GPIO.IN,GPIO.PUD_UP)

lo de PUD_UP es porque su configuración es en PULL-UP

LIBRERIA NEC.py

Creamos este fichero que lo ponemos en la misma carpeta que nuestros ejercicios, el código es complejo, sigue los pasos explicados en el protocolo NEC y lo hemos sacado del código demo de la página https://www.waveshare.com/wiki/AlphaBot :

```cpp+lineNumbers:true
import RPi.GPIO as GPIO from VARIABLES import *

def getkey(): if GPIO.input(IR) == 0: count = 0 while GPIO.input(IR) == 0 and count < 200: #9ms count += 1 time.sleep(0.00006)

    count = 0
    while GPIO.input(IR) == 1 and count < 80:  #4.5ms
        count += 1
        time.sleep(0.00006)

    idx = 0
    cnt = 0
    data = [0,0,0,0]
```

```
    for i in range(0,32):
        count = 0
        while GPIO.input(IR) == 0 and count < 15:    #0.56ms
            count += 1
            time.sleep(0.00006)

        count = 0
        while GPIO.input(IR) == 1 and count < 40:   #0: 0.56mx
            count += 1                               #1: 1.69ms
            time.sleep(0.00006)

        if count > 8:
            data[idx] |= 1<<cnt
        if cnt == 7:
            cnt = 0
            idx += 1
        else:
            cnt += 1
```

print (data)

```
    if data[0]+data[1] == 0xFF and data[2]+data[3] == 0xFF:  #check
        return data[2]

    if data[0] == 255 and data[1] == 255 and data[2] == 15 and data[3] == 255:
        return "repeat"
```

```
```

Control remoto

5.3 Test Control Remoto IR

Vamos a ejecutar un sencillo programa que nos vaya diciendo los códigos que lee las diferentes teclas utilizando la función key de la librería NEC.py

Fichero Test-ControlRemoto-IR.py

```
import RPi.GPIO as GPIO
import time

from VARIABLES import *

import MOVIMIENTOS
import MOVIMIENTOSPASO
import NEC

while True:
    key=NEC.getkey()
    if (key != None):
        print (key)
```

Lo que nos sale por pantalla son los siguiente códigos de las diferentes teclas del mando siguiente:

Estos son los valores (ordenados tal y como están en el mando)

|Tecla|Valor|Tecla|Valor|Tecla|Valor| |-----|-----|-----|-----|-----|-----| |CH-|69|CH|70|CH+|71| |<<|68|>>|64|>|67| |-|7|+|21|EQ|9| |0|22|100+|25|200+|13| |1|12|2|24|3|94| |4|8|5|28|6|90| |7|66|8|82|9|74|

Control remoto

5.4 Control remoto

¿Qué esperas? te lo pide el cuerpo !!! vamos a hacer un control remoto del robot !!

Vamos a definir las siguientes teclas

gobernado por el teclado *numérico*:

- PARAR = tecla 5
- ADELANTE=FORDWARD = 8
- ATRAS=BACKWARD = 2
- DERECHA=RIGHT = 6
- IZQUIERDA=LEFT = 4

https://www.youtube.com/embed/PfoVh2BTlLY

Solución

La solución es fácil con las librerías que hemos aprendido: * Ponemos las librerías fichero MOVIMIENTOS.py y ahora esta nueva NEC.py en la misma carpeta que vamos a crear este programa y las incorporamos en el programa con **import**. * También incorporamos las variables definidas en **VARIABLES.py** * Utilizaremos los códigos que hemos optenido en Test Control Remoto IR. * Un bucle, si no detecta la tecla 5 que haga los movimientos según las teclas del mando IR.

%accordion%Solución%accordion%

Fichero Control-Remoto-IR.py

```
import RPi.GPIO as GPIO
import time

from VARIABLES import *
```

```
import MOVIMIENTOS
import NEC

vel=50

print ('TECLAS :\nPARAR = tecla 5\nADELANTE=FORDWARD = 2\nATRAS=BACKWARD = 8\nDERECHA=RIGHT =
6\nIZQUIERDA=LEFT = 4')

key=0
while key!=28:
    key=NEC.getkey()
    if (key != None):
        if key==24:
            print ('\nadelante')
            MOVIMIENTOS.FORDWARD(vel)
        if key==82:
            print ('\natrás')
            MOVIMIENTOS.BACKWARD(vel)
        if key==90:
            print ('\nderecha')
            MOVIMIENTOS.RIGHT(vel)
        if key==8:
            print ('\nizquierda')
            MOVIMIENTOS.LEFT(vel)
        if key==28:
            print ('\nFin, has apretado STOP')
            MOVIMIENTOS.STOP()
```

%/accordion%

Módulo siguelíneas

Módulo siguelíneas

6 Módulo siguelíneas

Tiene un funcionamiento similar al Sensor obstáculos IR. El receptor tiene un sensor de reflexión de infrarrojos ITR20001/T. Un led emite luz IR contínuamente, la luz infraroja es reflejado por un obstáculo y lo recibe el receptor.

La salida del sensor es analógica y es sensible al color y la distancia del objeto detectado.

Tiene 5 canales de sensores. Chequeandolos se puede juzgar la posición de la línea oscura que esté en el suelo.

Módulo siguelíneas

6.2 TLC1543

Este robot no nos lo pone fácil con el siguelíneas ¿Por qué? Porque los 5 sensores (IR1..IR5) están conectados **a un conversor analógico digital TLC1543** tal y como puedes ver en su esquema eléctrico:

¿Cómo está conectado con GPIO?

Pues con estos números: * CS en GPIO 5 * Clock en GPIO 25 * Address en GPIO 24 * DataOut en GPIO 23

¿Cómo funciona este chip?

Pues léete su manual de instrucciones, si quieres te lo comentamos brevemente:

1. Este chip se activa por nivel bajo del **CS**.
2. En ese momento LEE LA DIRECCIÓN definida por **ADDRESS** (*empezando por el bit más alto MSB)* donde ADDRESS es un número entre 0 y 4 en binario que corresponde canal o sensor infrarrojo que se quiere leer A0 hasta A4 (que corresponden a los sensores IR1 hasta IR5).
3. Los primeros 4 pulsos de **CLOCK** son para leer ADDRESS (*en el flanco de subida*).
4. Los otros 6 *no valen para nada*
5. Los siguientes 10 pulsos se emite por **DATAOUT** (*empezando por el bit más alto MSB)* el valor leído del sensor de infrarojos que has seleccionado en ADDRESS.

> Mentirijillas: Realmente el punto 4 no es verdad, lo que pasa es que si has leido el punto 5 durante los 10 pulsos de reloj está sacando la lectura del IR definido

> por ADDRESS de los 10 anteriores, pero como son los 10 primeros, no valen.

Esto se ve mejor con un dibujo de su manual de instrucciones:

- Los 4 primeros pulsos de reloj por flanco de subida leen ADDRESS: B3 B2 B1 y B0 mientras tanto por DATAOUT está sacando los valores A9..A0 definido en una anterior ADDRESS que no vemos (los 10 primeros pulsos de CLOCK no tiene sentido esos valores de DOUT).
- Los rayados significan que igual da lo que haya pues no lo lee. Por ejemplo entre el pulso 5 y 10 del rejoj, no se está leyendo ADDRESS.
- Si te fijas, en los siguientes 10 pulsos de CLOCK (sólo aparece el primer pulso) ya empieza a salir por DATAOUT los valores del sensor definidos en ADDRESS como B3B2B1B0.
- Entre 10 pulsos de reloj y los otros 10 se necesita un tiempo de conversión A/D Conversion interval.

Módulo siguelíneas

6.3 TLC1543.py y VARIABLES.py

Tal y como hemos visto en la teoría del TLC1543 ¿Cómo está conectado? añadimos estas líneas al archivo **VARIABLES.py**

##SENSOR SIGUELINEAS

CS = 5 Clock = 25 Address = 24 DataOut = 23

##SENSOR SIGUELINEAS

CS = 5 Clock = 25 Address = 24 DataOut = 23

Script Damebit

En la teoría del TLC1543 ¿Cómo funciona? tenemos que obtener el bit de una posición dada de un número dado. Aquí hay un pequeño script para hacerlo (dale al play para ejecutarlo):

https://repl.it/@deleyva/ObtenerBitEntero?lite=true

TLC1543.py

Tal y como hemos visto en la teoría del TLC1543 ¿Cómo funciona? podemos hacer una librería que tenga una función **SENSORLINEA(cual)** que nos devuelva el valor que lee el sensor *cual*: * Importamos las variables de **VARIABLES.py** * Luego realizamos una función **SACADIRECCION** que active la salida ADDRESS según sus bits basándonos en la función **Damebit** que hemos visto. * Activamos 4 golpes de reloj sacando la dirección **ADDRESS** con la función SACADIRECCION * Hacemos 6 pulsos de **CLOCK** perdidos * Hacemos 10 pulsos de CLOCK pero leyendo el valor **DATAOUT** y convirtiendo esos bits en un número decimal, ese será el valor que devolverá la función **SENSORLINEA(cual)** * Grabamos esto en un archivo TLC1543.py

```
import RPi.GPIO as GPIO
import time

from VARIABLES import *

########################################################
#función de manipulación de bits
#ver https://repl.it/@javierquintana/ObtenerBitEntero
########################################################
def SACADIRECCION(x,n):
  if (x & (1
```

Módulo siguelíneas

6.4 Test-Sigue-lineas

Podemos hacer un test para ver cómo funciona este siguelíneas y cómo funciona la librería

- Importamos la librería TLC1543 creada anteriormente
- Vamos leyendo cada uno de los sensores.
- Que salga por pantalla los valores leídos.

https://www.youtube.com/embed/MFWnOyL_nzU

¿Te atreves?:

%accordion%Solución%accordion%

Fichero Testsiguelineas.py

```
import RPi.GPIO as GPIO
import time

import TLC1543

while True:
    for i in range(5):
        x=TLC1543.SENSORLINEA(i)
        print (" Sensor",i,"= ",x,end="")
    print(" ");
    time.sleep(0.5)
```

%/accordion%

Módulo siguelíneas

6.5 Siguelineas

Vamos a realizar un programa que siga la línea: * Pero **¡¡AL REVÉS!!!** ¿por qué marcha hacia atrás? (o sea, la cámara mira hacia la parte trasera del sentido de la marcha): mira al final de la página.

Fijaremos de antemano una velocidad pequeña de 25% y un incremento de velocidad de diferencia en los motores de 10% cuando no está centrado para que gire hasta que se centre. Veamoslo con un vídeo:

https://www.youtube.com/embed/rsdQ9fGOl-I

Solución

La solución es fácil con la librería TLC1543.py donde la función **SENSORLINEA(cual)** nos da el valor que lee los sensores IR. Recuerda que TLC1543.py en la misma carpeta que vamos a crear este programa y las incorporamos en el programa con **import**. * También incorporamos las variables definidas en **VARIABLES.py**

¿Te atreves?

%accordion%Solución%accordion%

- Leemos la lectura de los 5 sensores
- Ajustamos la velocidad de los motores según si hay lectura de línea negra y donde
- La potencia PWM no puede pasar de 0 a 100 por eso limitamos los valores
- Si no hay linea negra que vuelva hasta que recupera la línea negra:

Fichero Siguelineas.py

```
import RPi.GPIO as GPIO
import time


import TLC1543
```

```
from VARIABLES import *
from random import randint

x=[0,0,0,0,0]
vel=25
blanco=14
incremento=10
tiempo = 0.2


##hacia DELANTE
GPIO.output(IN1,GPIO.LOW)
GPIO.output(IN2,GPIO.HIGH)
GPIO.output(IN3,GPIO.HIGH)
GPIO.output(IN4,GPIO.LOW)

while True:
    lineanegra=0
    #Leemos los siguelíneas
    for i in range(5):
        x[i]=TLC1543.SENSORLINEA(i)
        if (x[i]90):
            velA=90
        if (velB90):
            velB=90
        ##activamos motores
        print ('con linea negra SENSORES=',x[0],'-',x[1],'-',x[2],'-',x[3],'-',x[4],'-
VELA=',velA,'VELB=',velB);
        PWMA.start(velA)
        PWMB.start(velB)
        #time.sleep(tiempo)
    else:           ##no tiene linea negra pues marcha atrás y que lo busque
        GPIO.output(IN1,GPIO.HIGH)
        GPIO.output(IN2,GPIO.LOW)
        GPIO.output(IN3,GPIO.LOW)
        GPIO.output(IN4,GPIO.HIGH)
```

```
        while (lineanegra==0):
            velA=randint(vel-incremento,vel+incremento)
            velB=vel-(velA-vel)  #el opuesto
            print ('SIN linea negra VELA=',velA,'VELB=',velB);
            PWMA.start(velA)
            PWMB.start(velB)
            time.sleep(tiempo)
            for i in range(5):
                x[i]=TLC1543.SENSORLINEA(i)
                if (x[i]
```

%/accordion%

¿Por qué en este ejercicio ALPHABOT va al revés?

Por que los sensores siguelineas por la parte de atrás del sentido de la marcha **PRODUCE UNA REALIMENTACIÓN POSITIVA** es decir, cuando detecta que hay que girar, gira, pero la cola se mueve demasiado deprisa que produce que pierda la línea. Controlarlo **es posible pero es difícil** la demo de Alphabot lleva el software para hacerlo.

No seas cobarde!! Pruébalo.

- Cambia el código anterior las marchas es decir los GPIO.output los HIGH por LOW y viceversa
- Y también el control de giro, es decir en vez de +incremento pon -incremento y viceversa
- ¿Funciona?

Chocheando un poco.. esto me recuerda a una vieja historia..

Los pioneros de la aviación lo tuvieron difícil. Se lo podríamos preguntar al **pastor** del pueblo Coruña del Conde: **Diego Marín Aquilera** que en 1793 inventó un artilugio que volaba de forma controlada... la pena es que la Inquisición, el cura del pueblo junto con los lugareños no tenían ni

idea que este español hubiera hecho historia, y que la aviación hubiera adelantado más de 100 años. Pensaban que eso era obra del demonio por lo tanto quemaron todos sus inventos.

Esa mala suerte no lo tuvieron **los hermanos Wright** que en 1903 volaron su primer artilugio:

[Fuente Wikipedia]

Pero... pusieron **el timón delante**, esto provocaba también una **realimentación positiva** (al levantar el timón, levantaba el morro, y esto provocaba que se levantase aún más, y viceversa a la hora de bajarlo). Los hermanos Wright patentaron su invento y gastaron todo su dinero en abogados para defender que nadie copiase su control, pero la verdad es ... que nadie lo hizo: La industria de la aviación detectó el fallo y los elementos de control van por detrás del ala principal, esto crea una realimentación negativa, por lo tanto mayor estabilidad en el vuelo y ... la ruina de los hermanos Wright.

> ¿En el diseño de este Alphabot habrá participado algún descendiente de los hermanos Wright?

Servos

Servos

7 Servos

Los servos son motores con una reductora, un sensor de posición y un circuito de control del ángulo de giro. Para saber más de servos te recomendamos el capítulo de servomotores del curso de Arduino de Aularagón..

Tiene 3 cables: * Marrón a GND * Naranja a 5V * Rojo la señal de control

La señal de control tiene que emitir un pulso alto durante un intervalo de al menos 20mseg, que según su duración en estado alto, se traduce en un ángulo de rotación del servo:

Servos

7.1 BRAZO.py y VARIABLES.py

En Alphabot el servo de abajo del brazo robot (lo llamaremos eje z por ser el responsable del giro del eje vertical) está conectado al GPIO 27 y el servo de arriba (lo llamaremos x) al GPIO 22 luego añadiremos estas líneas en nuestro fichero VARIABLES.py. Lo configuramos como salida y que inicialmente esten no activos para que no se mueva el brazo en el comienzo:

SERVOS BRAZO ROBOT

SERVOEJEX = 22

SERVOEJEZ = 27

####### SERVOS BRAZO ROBOT

GPIO.setup(SERVOEJEX, GPIO.OUT, initial=GPIO.LOW)

GPIO.setup(SERVOEJEZ, GPIO.OUT, initial=GPIO.LOW)

BRAZO.py

Realmente el control de un servo se hace con una modulación PWM que ya hemos visto. La función que modula la señal PWM es ChangeDutyCycle y se le da el argumento en % entre 0 y 100. Si queremos 180º necesitamos un pulso de 2.5ms por lo que en 20ms corresponde a 12.5% por lo tanto la fórmula es % = 2.5+10*(angulo/180):

```
import RPi.GPIO as GPIO
import time

from VARIABLES import *

servox = GPIO.PWM(SERVOEJEX,40)
servoz = GPIO.PWM(SERVOEJEZ,40)
servox.start(0)
servoz.start(0)
```

```
def ANGULO(angle,x):
    if (x):
        servox.ChangeDutyCycle(2.5 + 10.0 * angle / 180)
    else:
        servoz.ChangeDutyCycle(2.5 + 10.0 * angle / 180)
```

> Nota: Los servos tiemblan algo, es normal, no pienses que un robot barato esté bien calibrado.

Servos

7.2 Test Brazo

Vamos a realizar una función que controle con el teclado el brazo robótico: * Teclas 8 y 2 mueven el brazo robot en el eje x arriba y abajo * Teclas 4 y 6 mueven el brazo robot en el eje z derecha e izquierda.

Fijaremos de antemano un incremento de 10ª cada vez que pulsamos la tecla. Veamoslo con un vídeo:

https://www.youtube.com/embed/S3Z9vRjPtQo

Solución

- Ponemos la librería del fichero BRAZO.py en la misma carpeta que vamos a crear este programa y la incorporamos en el programa con **import**.
- Importamos las variables también con import * from VARIABLES
- Vamos incrementando los ángulos eje x y eje z según la tecla pulsada.
- Todo dentro de un bucle.

¿Te atreves? :

%accordion%Solución%accordion%

Fichero Test-Brazo.py

```
import RPi.GPIO as GPIO
import time

from VARIABLES import *

import BRAZO

angulox=90
anguloz=90
incremento=20
print("Teclas 8 y 2 SERVOX\n Teclas 4 y 6 SERVOZ")
```

```
while True:
    BRAZO.ANGULO(angulox,1)
    BRAZO.ANGULO(anguloz,0)
    tecla=input("Mueve el brazo : ")
    if (tecla=="8"):
        angulox=angulox-incremento
    if (tecla=="2"):
        angulox=angulox+incremento
    if (tecla=="4"):
        anguloz=anguloz+incremento
    if (tecla=="6"):
        anguloz=anguloz-incremento
```

%/accordion%

Cámara

Cámara

8 Cámara

Cámara

8.1 ¿Qué vamos a hacer?

Manejar la cámara web es fácil si queremos que salga por la salida HDMI de la Raspberry, simplemente ejecutando el siguiente código Python sale, en este caso durante 10 segundos **Pero no sale por VNC ni por SSH**:

```
from picamera import PiCamera
from time import sleep
camera = PiCamera()
camera.start_preview()
sleep(10)
camera.stop_preview()
```

Pero nosotros necesitamos que **retransmita = streaming** las imágenes, pues el robot se mueve, no tiene instalado un monitor.

Encontrarás en Internet varias formas de hacerlo:

1. Utilizando un programa en Python
2. Utilizando MJPG STREAMER bajo un programa servidor WEBIOPI
3. Utilizando **Motion** (recomendamos)

La primera opción video o este tutorial dependes de tener todas las librerías intaladas, por ejemplo *limmal* ...

La segunda opción WEBIOPI (http://webiopi.trouch.com/) siguen con la versión 0.7.1 sin actualizar luego no lo recomendamos

Vamos a usar **Motion**, un programa diseñado para manejar la cámara en estos contextos y sí que está actualizado (actualmente por la 4.3.1) y muy extendido en el uso de cámaras web, lo que nos da unas garantías de no tener problemas, su página web oficial https://motion-project.github.io/index.html .

Cámara

8.2 Configuracion

Lo primero que tienes que hacer es activar la cámara y Remote GPIO

> ¿Recuerdas? En el capítulo 8.1 de Raspberry básico ya activamos **VNC Server y SSH**, ahí también están la opciones de **activar la cámara y Remote GPIO**.

Si estás utilizando la Raspberry pero no de forma gráfica con VNC sino textual con SSH el comando a utilizar es

> **sudo raspi-config**

entra en la opción 5

Y activas la cámara y Remote SSH en las opciones P1 y P8 :

Cámara

8.3 Motion

Esta librería open-source muy utilizado en sistemas de alarma con la Raspberry (□□□□□□□ hacer un sistema de videovigilancia□□ de mi casa □□a distanci□□ y monitorizar □□ por muy bajo coste... □□), se pueden encontrar proyectos interesantes como :

- Que grabe en un vídeo cuando detecta un movimiento
- Que nos envíe un email con una foto cuando detecta uno movimiento

Pero nosotros NO nos interesa que detecte movimiento, sino que simplemente haga streaming.

Para esto, simplemente modificaremos el fichero de configuración de la librería **motion.conf**. Es muy típico modificar ficheros de configuración tipo texto en los softwares abiertos, lo que muestra su versatilidad y potencialidad. Puedes ver las diferentes posibilidades de configuración de Motion aquí

Cómo hacerlo

Abrimos una ventana de comandos, en SSH, ya sabes cómo y ejecutamos estas órdenes:

Instalamos MOTION :

sudo apt-get install motion

Editamos el fichero de configuración motion.conf con el editor nano

sudo nano /etc/motion/motion.conf

Buscamos estas líneas y las modificamos :

- stream_localhost on lo cambiamos por off si es on sólo localhost puede abrirlo, si es off pueden todos:
- **stream_localhost off**
- si vemos #stream_port 8081 y como queremos abrirlo por ese puerto, le quitamos el hastag, o sea lo dejamos así,:
- **stream_port 8081**
- Si vemos daemon off lo cambiamos por
- **daemon on**

Si estuvieran estas líneas webcam_localhost on y webcam_port 8080 las borramos, o mejor las dejamos como comentarios poniendo delante un hastag # así #webcam_localhost on #webcam_port 8080.

En esta página podemos ver otra configuración de motion.conf para el mismo propósito de streaming.

Grabamos el fichero: pulsando Ctrl+X se sale pero preguntará si queremos grabar el fichero con el mismo nombre, le decimos que sí

Finalmente ejecutamos motion con esta orden

sudo motion

Si queremos que se ejecute de forma automática cuando arranque la raspberrypi editamos el fichero /etc/rc.local y al final le ponemos esa instrucción. Es decir

sudo nano /etc/rc.local y añadimos sudo motion al final (he puesto un comentario My script optativo)

¿Cómo se ve desde la red local?

Pues abrimos un navegador y ponemos la dirección a través del puerto que le hemos dicho en stream_port o sea 8081

http://---LA-DIRECCION-DE-LA-RASPBERRY--:8081

es decir si la dirección es 192.168.1.25 entonces tecleamos http://192.168.1.25:8081

Si queremos un protocolo seguro https mirar esta página

¿Y desde Internet?

Opción instalar un nuevo servicio

No se puede hacer gráficamente en la página Remote.it no sabemos por qué. Hay que hacerlo con comandos con SSH.

sudo remoteit add -h

Nos sale una lista de servicios que podemos añadir, tecleamos el ID del servicio que queremos añadir en este caso vemos en la figura que el 7 es HTTP.

Nos pide el puerto, ponemos **8081** el mismo que en Motion

Un nombre para el servicio, le hemos puesto webcam, y hecho lo que tienes que hacer en la Raspberry.

Entramos ahora en un ordenador a remote.it en nuestros "Devices" y pinchamos en el servicio que hemos creado:

Y automáticamente nos abre el navegador con la webcam funcionando

(en este caso he utilizado el robot para vigilar la impresora 3D)

Vale, pero .. ¿y cómo se quita un servicio de Remoteit?

Entramos en al página web en el device en cuestión

Y copiamos el ID del servicio que queremos borrar:

Y ejecutamos la orden **sudo remoteit remove --id y la ID que queremos borrar** es decir en mi caso :

sudo remoteit remove --id 80:00:00:00:01:0A:18:DF

Opción cutre con VNC

Si lo anterior por alguna razón fallara o remote.it quita el servicio HTTP, puedes ver la cámara por VNC.

Tienes que acceder a la Raspberry desde Internet con VNC mira estos apuntes

Una vez accedido por VNC remotamente podemos abrir el navegador de la misma Raspberry y la IP de él mismo es 127.0.0.1 luego abrir

http://127.0.0.1:8081

ojo en el navegador de la Raspberry no en tu ordenador.

Bajos del coche

Bajos del coche

9 Bajos del coche

Para finalizar, vamos a ver un ejemplo de proyecto:

Hemos visto varios elementos, son las piezas del puzzle de tu creatividad, vamos a ver un ejemplo de juntar varias.

Proyecto

Este proyecto intenta hacer un robot que nos visualice de forma remota los bajos de un coche, así pues juntamos 3 piezas del puzzle: * Movimientos paso a paso con las teclas * Movimiento brazo robótico * Cámara

https://www.youtube.com/embed/plpvaGh7otw

¿Te atreves?

%accordion%Solución%accordion%

- El proyecto es fácil pues es la unión de Movimientos paso a paso con las teclas y Movimiento brazo robótico
- Ver la cámara no implica ningún código Python especial, sólo es un comando Linux si está bien configurado.

Fichero BajosCoche.py

```
import RPi.GPIO as GPIO
import time

from VARIABLES import *

import BRAZO
import MOVIMIENTOS
```

```
import MOVIMIENTOSPASO



velR=30
numR=10
velL=30
numL=10

angulox=90
anguloz=90
incremento=20
print("Teclas 8 y 2 SERVOX\n Teclas 4 y 6 SERVOZ")
print ('TECLAS ¡en minúscula!:\nADELANTE=FORDWARD = f\nATRAS=BACKWARD = b\nDERECHA=RIGHT =
r\nIZQUIERDA=LEFT = l')
tecla='x'

print ('Mira la cámara en http://192.168.1.25:8080')

while True:
    BRAZO.ANGULO(angulox,1)
    BRAZO.ANGULO(anguloz,0)
    tecla=input("Mueve el brazo o movimiento: ")
    if (tecla=="8"):
        angulox=angulox-incremento
    if (tecla=="2"):
        angulox=angulox+incremento
    if (tecla=="4"):
        anguloz=anguloz+incremento
    if (tecla=="6"):
        anguloz=anguloz-incremento
    if tecla=='f':
        print ('\nadelante')
        MOVIMIENTOSPASO.BOTH(velR,numR,velL,numL)
    if tecla=='b':
        print ('\natrás')
        MOVIMIENTOSPASO.BOTH(velR,-numR,velL,-numL)
```

```
    if tecla=='r':
        print ('\nderecha')
        MOVIMIENTOSPASO.BOTH(velR,-numR,velL,numL)
    if tecla=='l':
        print ('\nizquierda')
        MOVIMIENTOSPASO.BOTH(velR,numR,velL,-numL)
```

%/accordion%

Bajos del coche

Grupo Robotica Educativa Aragon

Bajos del coche

Muro

https://padlet.com/CATEDU/alphabot

https://padlet.com/embed/w64rih545vkj

Hecho con Padlet

Bajos del coche

Créditos

Autoría

- {{ book.author }}

Colaboradores:

Cualquier observación o detección de error en soporte.catedu.es

image-1648462225402.gif

image-1648462299882.png

image-1648462361893.png