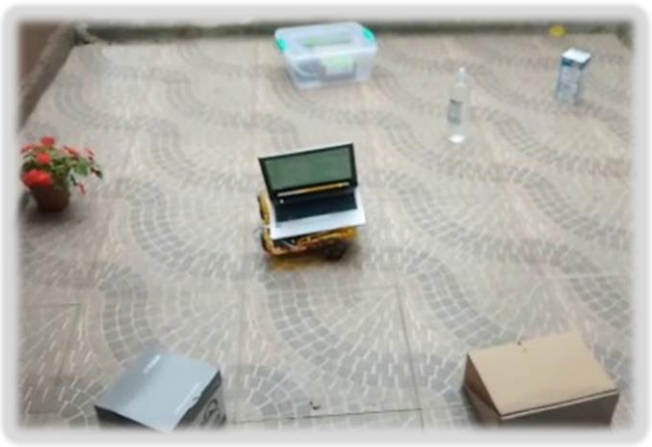
Grupo MINA - Instituto de Computación
Facultad de Ingeniería, Universidad de la República

Taller de Robótica Educativa con el Robot Butiá 2020

ROBOT ESPÍA



Estudiante:
Veronica Bentancor

Docentes:
Guillermo Trinidad
Gonzalo Tejera

Índice

Introducción

El presente proyecto surge como t final del curso *Taller de Robótica Educativa con el Robot Butiá,* de la *Facultad de Ingeniería*, *UdelaR*, con el objetivo de brindar un primer acercamiento a la programación y a la robótica, se pretende presentar un proyecto modular, donde las partes se puedan reutilizar en otros proyectos.

La motivación de este proyecto surge por el robot *Opportunity* creado por la NASA para la exploración de planeta Marte. Opportunity permitió explorar el planeta rojo, entre los descubrimientos se destaca que Marte fue alguna vez un planeta cubierto de agua y, por lo tanto, pudo haber sido un mundo habitable. (Opportunity, 2020)

Se decidió trabajar con TurtleBots apuntando a un público de educación primaria, de todos modos, este proyecto puede ser realizado por cualquier persona que disponga del kit butiá.

Descripción

El robot cumplirá el rol de un espía, o un explorador, que deberá recorrer distintos lugares, tomando fotografías de los objetos con los que se encuentra, para permitirnos espiar el lugar.

El robot recorre distintos lugares continuamente, cuando detecta algún obstáculo (“espía”) captura una imagen de este que se guarda en la computadora.

Imagen 1. El robot butiá fotografiando el objeto que detectó

Teniendo en cuenta que podría haber poca luz en el lugar donde el robot esté recorriendo y las imágenes podrían salir oscuras, se evita capturar imágenes cuando no hay suficiente iluminación.

Una vez que se termina el espionaje del butiá, desde la computadora utilizada se puede ver las imágenes capturadas por el robot.

Video

Aquí podemos ver el comportamiento del robot espía: **https://youtu.be/xmQA4J6Plpk**

Planificación del proyecto

Se va a dividir el proyecto en tres partes, primero probaremos los sensores a utilizar para luego desarrollar el código para recorrer el lugar, y por último nos encargamos de capturar las imágenes según la luz de ambiente.

Cabe aclarar que cada una de las partes, se presentará más de una versión para resolver el problema, las cuales difieren en dificultad. Puede resolverse el robot espía de una forma simple o utilizar las versiones avanzadas.

Se proveerán cuadros explicativos acerca del funcionamiento de las partes relevantes del robot, así como se mencionan errores comunes que se puedan encontrar y cómo solucionarlos. Además de brindar información extra referida a los temas de interés.

Primera parte

En este proyecto trabajaremos con el sensor de distancia y el sensor de luz, explicaremos como colocar los mismos y cómo trabajar con ellos.

En primer lugar debemos activar las extensiones o *plug-in*, en este proyecto utilizaremos las siguientes: *butia* y *turtleblocksextras*. Las habilitamos desde el menú en la opción extensiones como se puede ver en la siguiente imagen:

OBSERVACIÓN:

Se deberá reiniciar el TurtleBots para que los cambios tengan efecto!

Comenzaremos utilizando los sensores, en las imágenes 2 y 3 vemos cómo se deben colocar en particular el sensor de luz, pero todos se colocan del mismo modo.

Primero sujetamos (con el tornillo y la tuerca) la placa sensor de forma perpendicular al chasis, luego colocamos el sensor en la placa sensor. Conectamos con el cable RJ45 desde el sensor a algún puerto de la placa USB4Butia. (Consultar nombre de las partes y armado del kit (Armado del Kit butiá, 2020)). La placa USB4Butiá, dispone de seis puertos RJ45 que están enumerados del 1 al 6 en el chasis.

Imagen 2

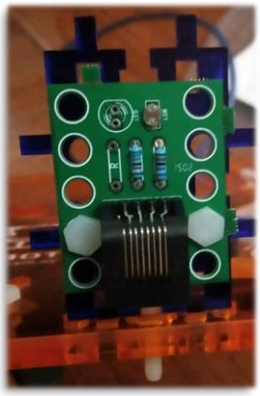


Imagen 3

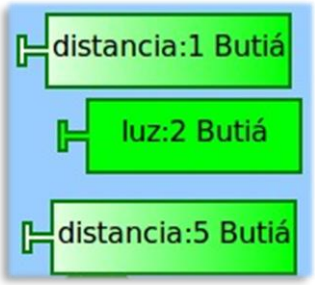


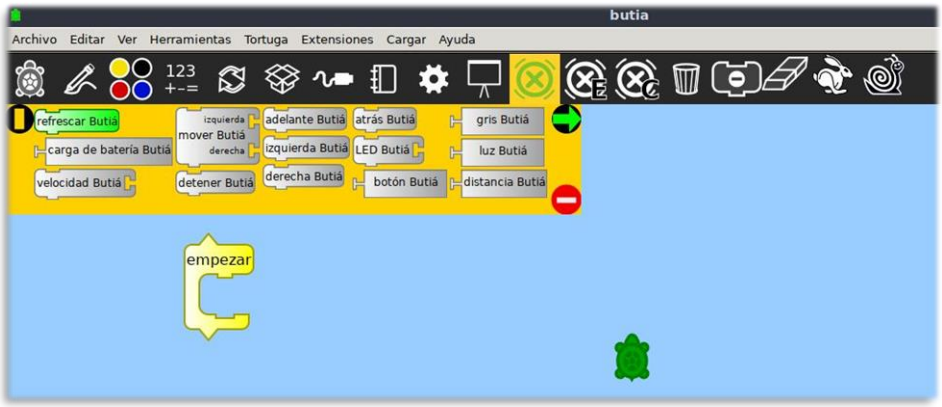
Imagen 4

En este proyecto utilizaremos los sensores ubicados de forma perpendicular porque queremos que el butiá “mire” al frente, pero si quisiéramos observar algo en el piso, conectaremos las placa sensores paralelas al chasis.

Tener en cuenta que la placa USB4Butia al ser conectada a la computadora deberá encender el led.

Imagen 4. Placa USB4Butiá

Para probar los sensores utilizamos las funciones que se encuentran en la extensión/*plug-in* denominada: “*butiá*”.



Como vemos en la tercera imagen el sensor aparecerá en el TurtleBots seguido del número del puerto en el que está colocado el sensor. En este caso el sensor de luz está conectado en el puerto 2, hay un sensor de distancia conectado en el puerto 1 y el otro en el 5. Una observación es que todos los sensores pueden ser conectados en cualquier puerto USB de la placa USB4Butiá.

Imagen 5. La placa USB4Butiá conectada a la computadora

OBSERVACIÓN:

Para probar los sensores solo necesitamos hacer uso de la placa USB4Butiá.

No tenemos que tener el butiá totalmente armado ni las pilas colocadas, alcanza con conectar el USB desde la placa USB4Butiá a la computadora.

Sensor de distancia

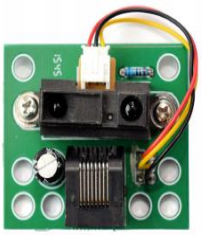


Imagen 6. Sensor de distancia

Comencemos por probar el sensor de distancia. Este sensor nos permite reconocer cuán cerca se está del objeto al que se apunta.

Esperaríamos que nos devuelva exactamente la distancia al objeto en metros o centímetros, esto no sucede ya que el robot Butiá es un robot de hardware libre y no se encuentra dentro de sus objetivos realizar mediciones exactas, se busca que al ser un robot educativo accesible permite al usuario ser también desarrollador. (Manual de uso básico de TurtleBots y robot Butiá 2.0, 2020).

El sensor de distancia retorna valores entre 0 y 65535. Los valores crecerán a medida que alejemos el sensor de dicho objeto (y se reducirán a medida que lo acerquemos), este

sensor tiene un alcance de 10 cm a 80 cm (Sensor de distancia: gp2y0a21yk0F, 2020), hay un punto cercano al sensor en el cual desde dicho punto o más cerca, nos retorna valores aleatorios (esto sucede a distancias menores a 10 cm) porque ya no está midiendo correctamente, se puede pensar como un punto ciego al sensor.

Por otro lado, existe otro punto en el cual a partir de dicho punto o más lejos, el sensor devuelve valores altos (cercanos a 65535) porque el objeto próximo está demasiado lejos (esto sucede a partir de los 80 cm).

Teniendo en cuenta el alcance del sensor, podemos realizar pruebas colocando distintos objetos a determinadas medidas para conocer qué valor arroja el sensor según las distancias. Conectamos el sensor como fue explicado anteriormente y utilizamos el bloque “distancia butiá” para testear el sensor.

Ejemplo

Distancia desde el sensor al objeto (cm)	Promedio del medidas del sensor distancia
10	36224
20	49408
30	53376
40	56371

Debido a que los valores arrojados pueden variar entre sensores, es importante hacer pruebas con el sensor para determinar los valores que arrojará, se recomienda hacer varias mediciones y tomar un promedio de ellas para obtener un resultado más preciso.

OBSERVACIÓN

Suele ocurrir que para una misma distancia, los valores arrojados por el sensor cambian continuamente, esto se debe a la desviación del sensor, pero estos cambios son mínimos… Por ejemplo podría pasar que el sensor de una serie de valores como: 3080, 3200, 2930, 3150...

Ahora deberíamos reconocer qué rango de valores devuelve el sensor de distancia cuando encuentra algún objeto cerca, recordemos que como buscamos sacar fotos de estos objetos, no deberíamos estar muy cerca ya que sino en la foto no se va a observar bien, un buen número puede ser cerca de unos 15cm aproximadamente, pero puedes experimentar con tus propias medidas.

A partir de los valores obtenidos al probar el sensor, podemos ver para el nuestro que los valores que nos sirven son menores a 45000, este dato lo utilizaremos en la segunda parte.

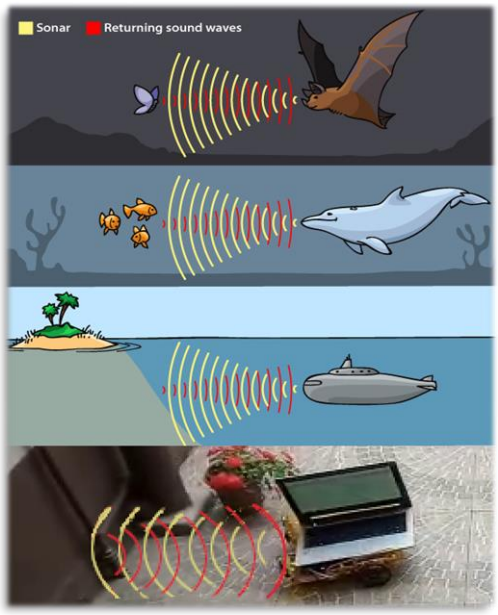


Imagen 7. Proceso de detección de objetos.

SABÍAS QUÉ…

El sensor de distancia funciona con un sensor infrarrojo y su método de detección se asemeja a la de otros animales en la naturaleza (o incluso algunos submarinos), los cuales en vez de tener sensores infrarrojos utilizan el sonido para ubicarse en el espacio, en un proceso llamado ecolocalización.

(science behind super powers, 2020)

Sensor de luz

Pasemos a conocer el sensor de luz, este mide la intensidad de la luz, es decir que nos da una noción de cuánta luz hay en el entorno del robot.

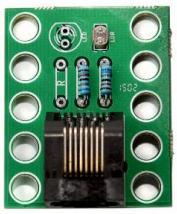


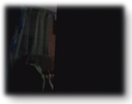
Imagen 8. Sensor de luz

Retorna valores entre 0 y 65535 (igual que el sensor de distancia). Mientras más luz recibe el sensor más alto es el valor que retorna, 0 significa oscuridad total y luego aumenta a medida que sube la intensidad de la luz a la que está expuesto.

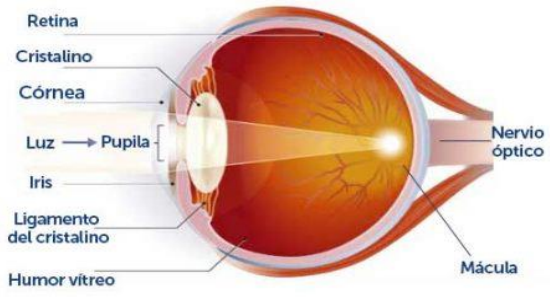
Colocamos el sensor como ya mencionamos anteriormente y lo conectamos en cualquier puerto de la placa USB4Butiá. Para probar este sensor utilizamos el bloque “luz butiá”. Se recomienda hacer varias pruebas en distintos lugares para ver los valores arrojados, por ejemplo a la luz del día, a la luz de habitación, de noche, a la sombra, etc. En particular en este proyecto nos interesa saber cuándo una imagen se ve bien, eso sería cuando no saliese oscura la foto, para esto lo que haremos es probar el sensor de luz en varios ambientes y desde la cámara de la computadora sacar imágenes.

Recordar que los valores dependen del sensor utilizado y las imágenes de la cámara de la computadora, a modo de ejemplo se plantea la siguiente tabla:

Ejemplo

Ambiente	Valor del sensor de luz	Fotografía
Habitación con la luz encendida	13120	
Poca luz o a la sombra	10024	
Habitación con poca luz	8064	
Habitación de noche sin luces encendidas	6080	

Según estos datos, podemos concluir con nuestro sensor y con la cámara que tenemos, que a partir de valores mayores a 10000 la luz es suficiente para que la imagen se vea bien. Este dato lo utilizaremos en la segunda parte.



SABÍAS QUÉ…

El sensor de luz de butiá funciona convirtiendo ondas de luz en electricidad, estos no son los únicos, ya que los humanos tenemos nuestro propio sensor, el cual dilata las pupilas cuando hay poca luz, dejando que entre la mayor cantidad posible para poder ver a oscuras y las contrae cuando hay demasiada, para proteger el ojo y no cegarnos. (oftalvist, 2020)

Los piratas sabían de esto, por eso llevaban un parche en el ojo (no es que fueran todos tuertos!), cuando debían abandonar la soleada cubierta de barco para luchar en su oscuro interior, se retiraban el parche.

(Ciencia Historica, 2020)

Posibles problemas qué pueden surgir al testear un sensor

Estos problemas aplican para todos los sensores, no solo los que utilizamos en este proyecto:

- ➔ retorna un valor -1.
- ➔ retorna un valor fijo, no varía (excepto el sensor botón que retorna 0 ó 1).
- ➔ en TurtleBots, el bloque del sensor no se pone en verde aunque esté todo correctamente conectado.
- ➔ para una misma prueba, retorna valores que varían demasiado entre sí (por ejemplo 1200 y luego 49500).
- ➔ no funciona correctamente, por ejemplo un sensor de distancia que retorna valores más pequeños cuando un objeto está lejos y estos aumenten al acercar el objeto o un sensor de luz que retorna valores más bajos cuando se le acerca una linterna.

Soluciones posibles

Los problemas mencionados pueden deberse a que el sensor o el cable con el que se conectó esté roto o a una falla de TurtleBots.

Se sugiere:

- ➔ conectar el sensor utilizando otro cable.
- ➔ conectar en otro puerto el sensor.
- ➔ testear otro sensor para corroborar que lo que falla es el sensor actual.
- ➔ reiniciar TurtleBots.
- ➔ reiniciar la computadora con el butiá conectado a la misma.
- ➔ corroborar que la luz led de la placa USB4Butia está encendida.
- ➔ conectar utilizando otro cable USB desde la placa a la computadora.
- ➔ probar utilizando otra placa USB4Butia

Segunda parte

El desafío en esta parte es desarrollar los bloques para recorrer donde vamos a espiar, puede ser una habitación, un patio, el salón de clases, cualquier lugar.

Haremos dos versiones de esta parte, la primer versión será girar siempre hacia el mismo lado cuando se detecta un objeto, por ejemplo girar siempre a la derecha. En la segunda versión veremos qué podemos hacer para que el sentido de giro sea aleatorio.

Antes que nada, deberíamos hacer pruebas con los bloques “adelante butia”, “derecha butia”, “izquierda butiá”, etc. para corroborar los movimientos. Puede suceder que los motores estén soldados al revés y que el bloque “adelante butiá” realice un giro, esto no es un error en el armado del butiá sino en la fabricación de los motores, de ser así; podría pasar que: “adelante butiá” realiza un giro hacia la derecha, “atrás butiá” realiza un giro hacia la izquierda, con “derecha butiá” se mueve hacia adelante y con “izquierda butiá” se mueve hacia atrás.

Primera versión

El robot espía debe avanzar siempre que sea posible, hasta detectar un objeto que le impida seguir, así el espía va a poder recorrer por todas partes el lugar a explorar. Cuando el robot detecte que hay un objeto cerca debemos detenernos, retroceder (hasta el lugar donde estábamos al detectarlo) y girar hacia la derecha, continuando en un nuevo rumbo.

Tenemos que hacer que el butiá retroceda por dos motivos, primero, queremos evitar que choque al objeto al intentar girar, y segundo, en la parte 3 vamos a necesitar estar lejos del objeto para sacarle una buena foto.

Procedimiento

Primero vamos a comentar las estructuras de control a utilizar y cómo definir acciones.

Debemos tener en cuenta que para estar continuamente buscando si hay obstáculos presentes necesitaremos hacer uso de alguna estructura de control, en este caso debemos tener el código dentro del bloque “por siempre”, esto permite que continuamente el robot espía avance y verifique la presencia de objetos en su camino.

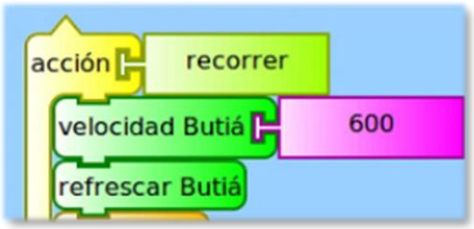
Para simplificar, vamos a definir el código dentro de una acción a la cual llamaremos “recorrer”, las acciones las encontramos dentro de la paleta “blocks”.

Ahora pasamos a colocar los sensores, colocaremos en cada borde del chasis un sensor de distancia, así en caso de haber un obstáculo cerca de los “bordes” del robot podrá capturarlos.

Imagen 9. Ubicación de los sensores distancia

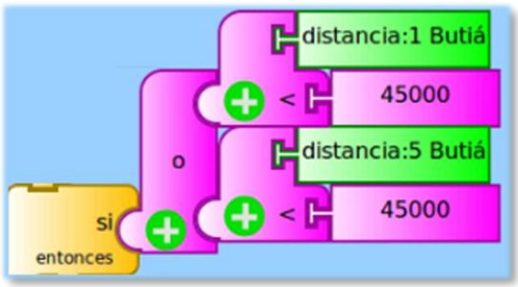
También se podría llegar a utilizar un solo sensor de distancia, el problema es que si utilizamos un sólo sensor, lo deberíamos ubicar en el centro del chasis y si el robot en algún momento tiene un objeto cerca de sus extremos posiblemente no lo capte porque está muy lejos de él, y nos terminaríamos chocando.

Empezamos por fijar la velocidad del butiá en un valor, por ejemplo 600. La velocidad del butiá puede ir de 0 a 1023 (cuanto más grande el número, más rápido se moverá), pero si utilizamos los valores cercanos al máximo el robot no llega a detectar los obstáculos a tiempo y termina chocando contra ellos.



Para detectar si hay algún objeto a cerca debemos hacer uso de la estructura de control “si entonces” (corresponde a la estructura *if*) si observamos las forma de este bloque podemos ver que tiene un “hueco curvo” es donde se debe colocar las condiciones lógicas que se pueden elaborar con la paleta de operadores numéricos.

Hacemos que el butiá avance hacia adelante hasta detectar con alguno de sus dos sensores (por eso utilizamos el operador lógico *or*) un objeto por delante. Gracias a las medidas que tomamos en la parte anterior, sabemos que para nuestro sensor una distancia mayor a 45000 representa objetos desde 15-20cm en adelante.

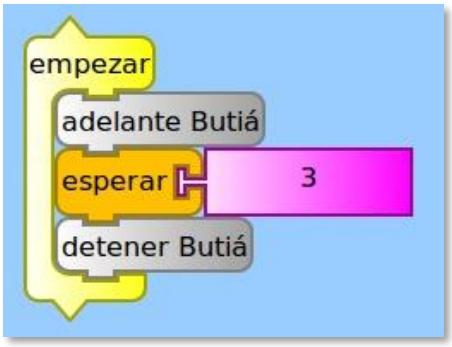


Si el butiá logra detectar un objeto con alguno de los dos sensores, entonces le ordenamos retroceder durante aproximadamente un segundo, para luego indicarle que gire hacia la derecha, buscando un nuevo rumbo.

Bloque esperar

Con este programa si probamos el código el butiá no se mueve, esto se debe a que si bien se ejecutó la instrucción “adelante butia”, inmediatamente a una velocidad altísima imperceptible para el ojo humano se ejecutó la instrucción “detener butiá” entonces el butiá no se mueve.

Esto lo solucionamos utilizando el bloque “esperar”, en este programa el butiá avanza tres segundos y luego inmediatamente se detiene.

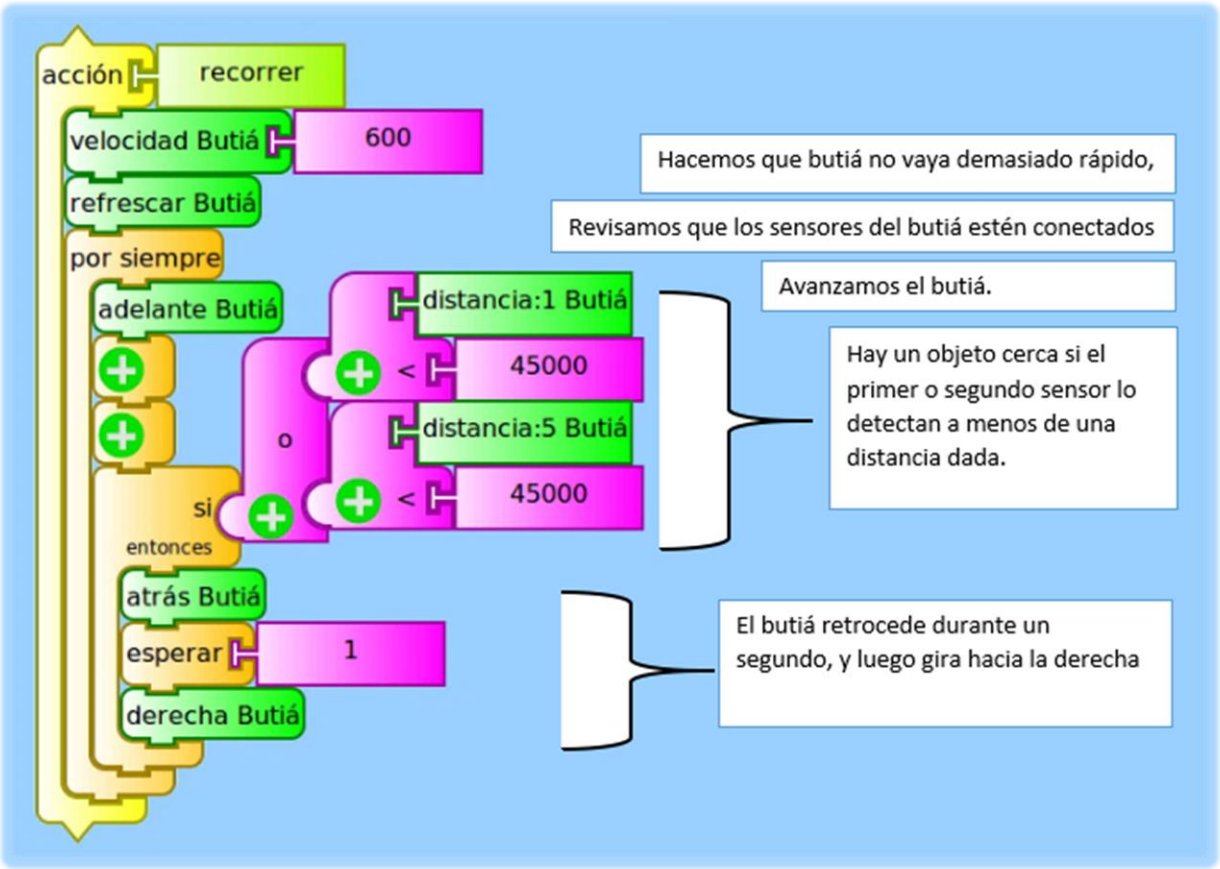


Por lo tanto, el bloque esperar es una orden que permite detener la ejecución de nuestro programa (pero no los motores del butiá) durante el tiempo deseado. El bloque esperar sirve mucho cuando le mandamos instrucciones al robot.

Al ejecutar por ejemplo la instrucción “adelante butiá”, se les envía a los motores la orden de girar para que el robot se mueva hacia adelante; y hasta que no se envíen nuevas órdenes a los motores, estos girarán hasta que se quede sin batería o sea desconectado. Entonces con el bloque esperar podemos definir durante cuánto tiempo puede ejecutarse una orden hacia los motores.

Volviendo a nuestro desafío, haremos que el robot avance siempre que sea posible y en caso de haber un obstáculo que retrocede durante 1 segundo, luego gira hacia la derecha y nuevamente busca obstáculos (en el código lo vemos por estar dentro del bloque “por siempre”).

Juntamos todas las partes y nuestros bloques deben lucir algo así:



Segunda versión

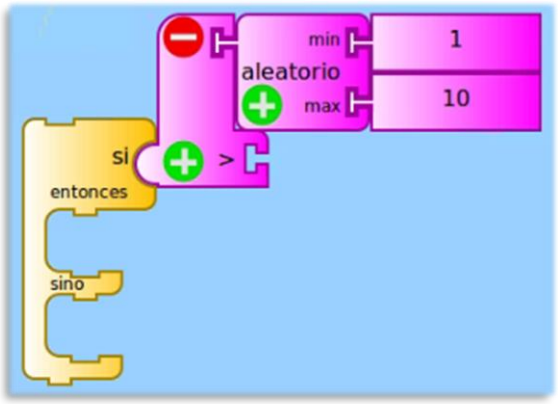
Para esta versión solo cambiaremos el bloque girar derecha por una acción de giro aleatorio

Podemos hacer que el butiá gire de manera aleatoria, es decir, hacia la derecha o hacia la izquierda luego de detectar un obstáculo (en vez de seguir siempre por la derecha). Para eso generamos con el bloque aleatorio de la paleta de operadores numéricos, un valor aleatorio entre 1 y 10.

El bloque aleatorio

Bloque aleatorio

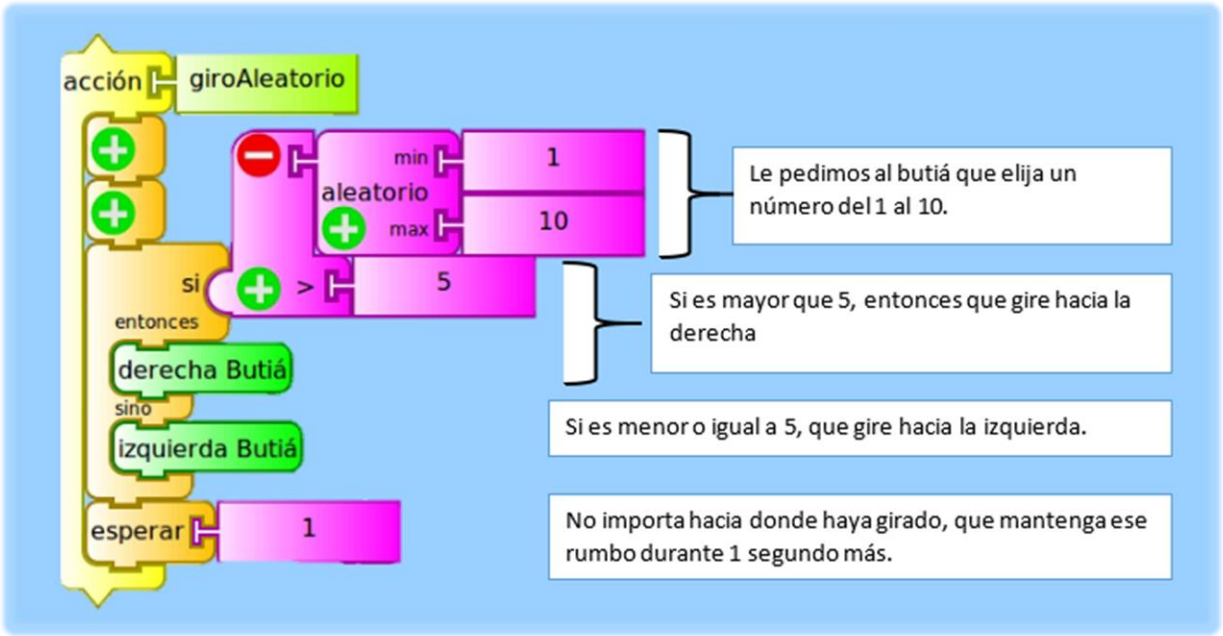
Si el valor generador es mayor o igual a 5, entonces movemos el butiá hacia la izquierda, de lo contrario, lo hacemos hacia la derecha.



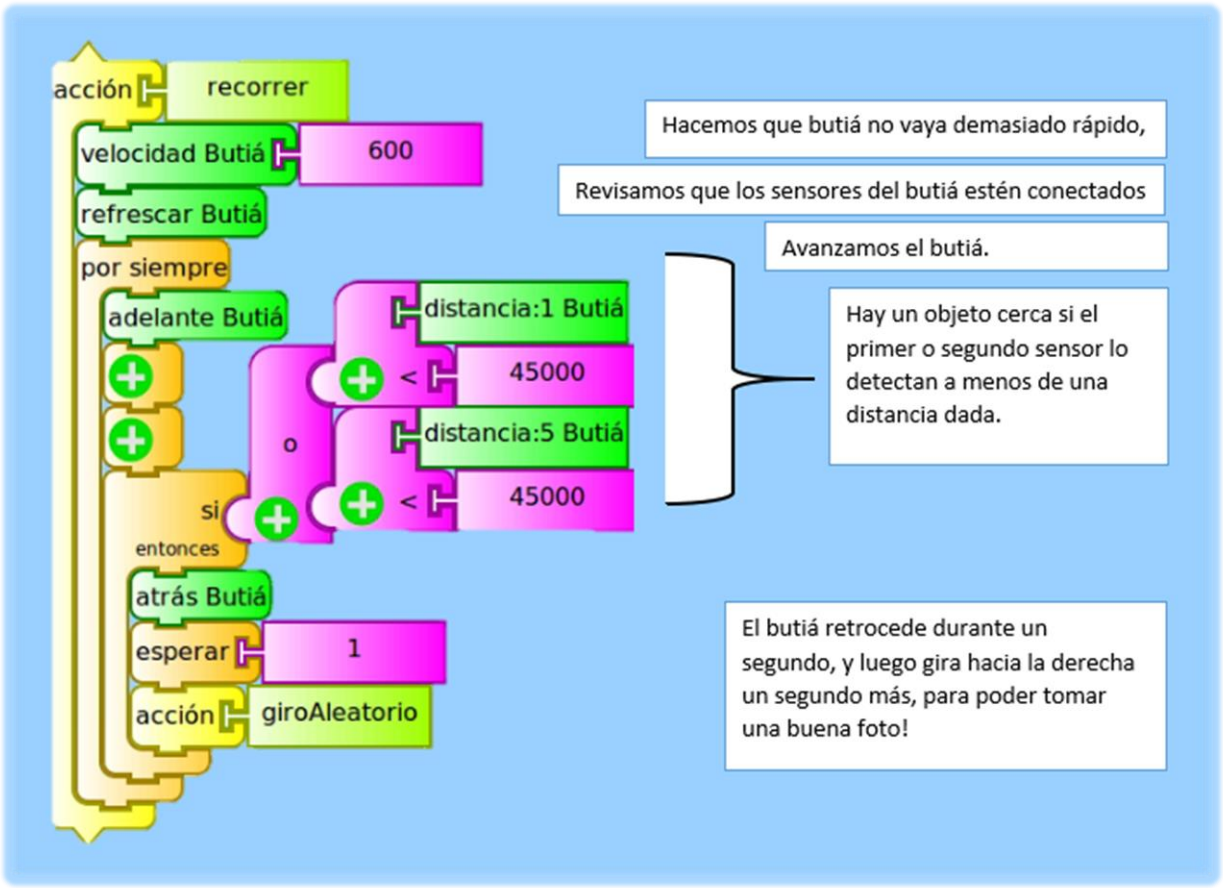
INFORMACIÓN:

Cuando decimos al butia que haga un giro aleatorio, es como si le estuviéramos pidiendo que tire un dado de 10 caras, si saca 5 o menos, le decimos que gire en un rumbo, y si saca más, que lo haga en el otro!

La acción de giro aleatorio nos debe quedar así:



Actualizamos el bloque de recorrer:



Tercer parte

En esta parte el nuevo desafío es cómo trabajar con la cámara de la computadora, nuevamente realizaremos dos versiones.

Imagen 10. El robot tendría que girar hacia la derecha para capturar la planta.

La primera versión (más sencilla) siempre capturará imágenes, en la segunda versión haremos un espía aún más inteligente, tendremos la precaución de capturar imágenes solo cuando haya suficiente luz, de forma de evitar sacar muchas fotos que salgan oscuras y luego deban ser eliminadas.

Se debe tener en cuenta es que si la computadora es colocada en el butiá "de costado" (como se ve en la imagen) cuando el sensor de distancia detecta el objeto debemos realizar un giro para quedar enfrente al él y recién ahí capturar la imagen.

En caso de que la computadora pueda ser colocada correctamente (de frente), al momento de detectar el objeto se captura la imagen inmediatamente.

Como se indicó en la primer parte, una vez que tenemos la extensión, vamos a *paleta de opciones adicionales*, veremos un bloque que dice *ejecuta el bloque tamyblock.py que se encuentra en el diario,* lo arrastramos y haciendo doble clic del lado de la *viborita* podemos cargar un archivo *Python* (.py). El archivo que debemos cargar es main.py[1]. Ignoramos el parámetro 100 y ya tendremos todo pronto.

De esta forma podemos importar cualquier otro código Python desde TurtleBots.

OBSERVACIÓN

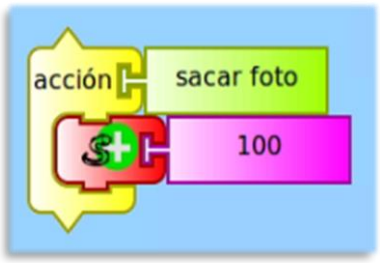
Este código utilizará la cámara de la computadora para sacar fotos y las guardará en el escritorio.

En caso de querer guardar en otra carpeta las instrucciones se encuentran en el anexo.

Lo añadimos dentro de un bloque acción y así podemos luego usarlo para sacar las fotos.

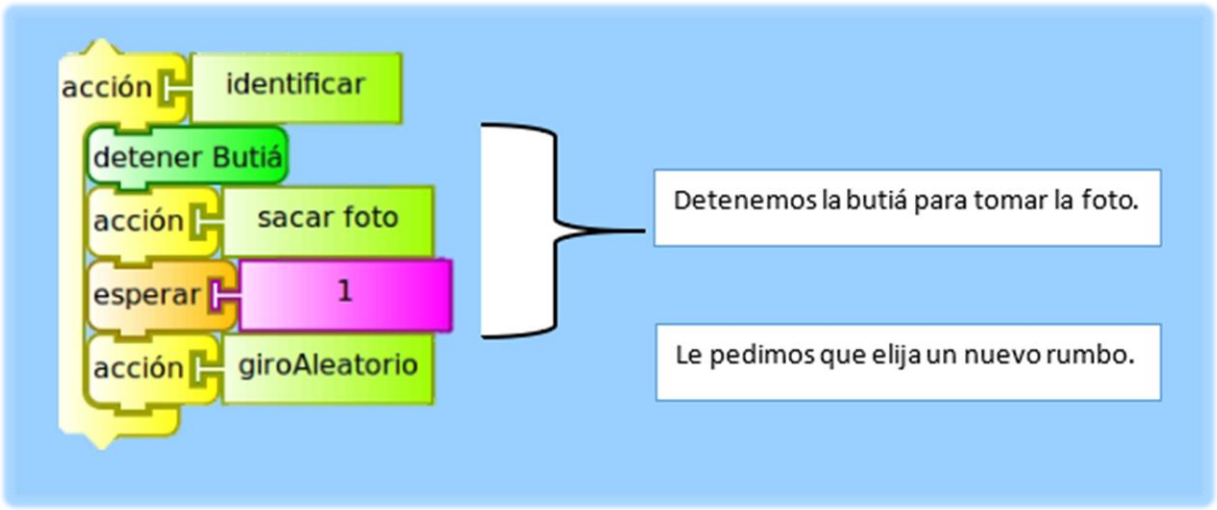
Como ya hemos mencionado, recomendamos previamente ir probando las acciones por separado antes de probar todo el código junto.

[1] Al final del documento en la sección Anexo se encuentra la explicación detallada, debido a que es muy técnica!.



En este caso probar la acción “sacar foto”, si la cámara de la computadora enciende alguna luz al iniciar la acción, significa que la cámara está siendo utilizada. Si esto no sucede, debemos probar primero sacar fotos con alguna aplicación de escritorio o página web. También se puede ir al escritorio para visualizar las imágenes que se van capturando.

Vamos a definir una acción para tomar fotos, lo que nos va a facilitar si luego queremos hacer mejoras, la llamaremos “*identificar*” porque identificará a los objetos.



Podemos mejorar la parte 2 de dos formas distintas modificando solamente dentro del bloque “si entonces”:

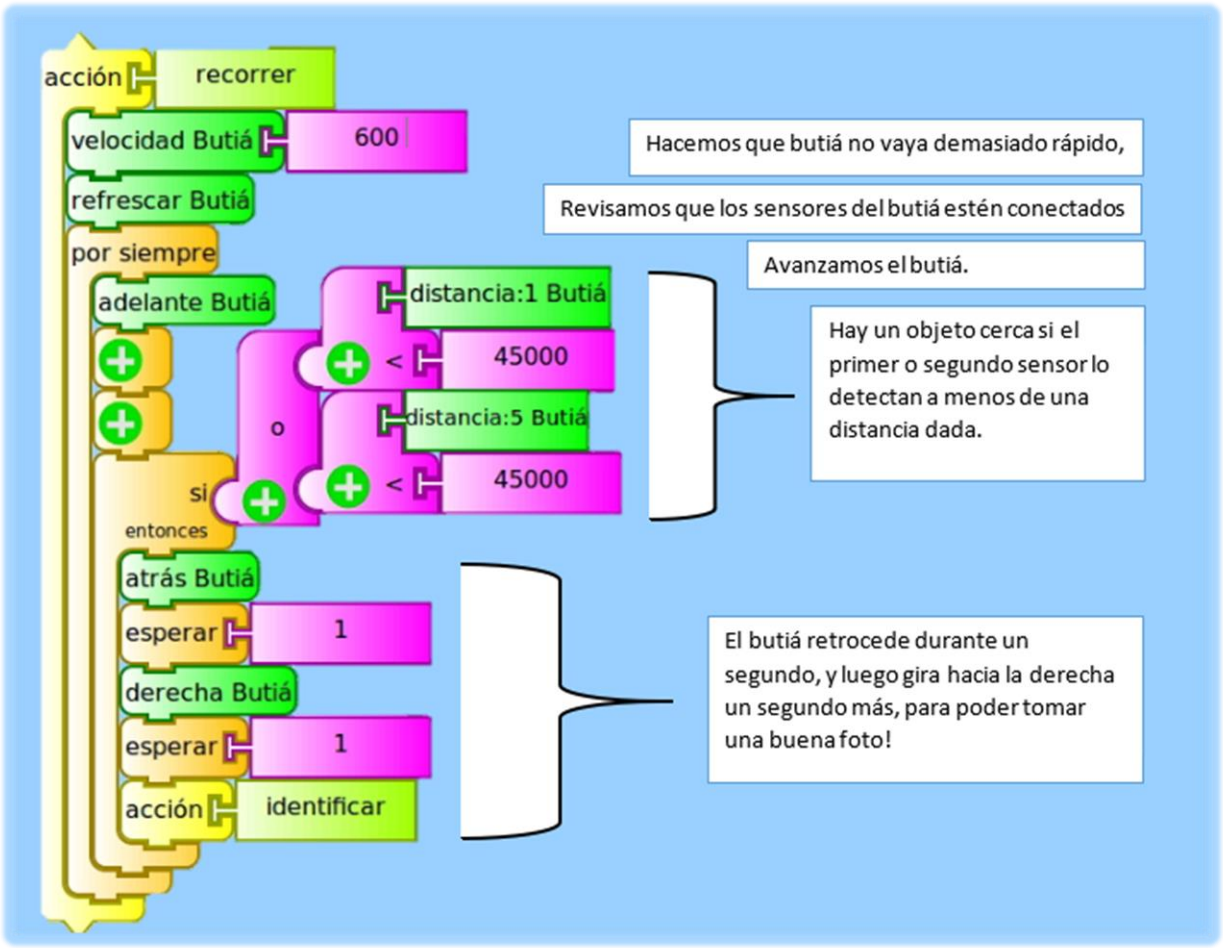
Primera versión

Mejoraremos lo que teníamos hasta el momento haciendo que cuando el robot detecte que hay un objeto cerca, se detenga para sacar una foto, esto evitará que las imágenes salgan movidas o borrosas. Para esto, se debe retroceder y girar hacia la derecha, para “detenerse” cuando la cámara esté frente al obstáculo.

Procedimiento

Haremos que luego de detectar un objeto con alguno de los dos sensores, retroceda como lo hacía antes durante aproximadamente un segundo, para decirle luego que gire hacia la derecha. Luego de posicionarse frente al objeto, le ordenamos al butiá detenerse (a nadie le gusta las fotos borrosas!) y ejecutamos la acción sacar foto. Hay que recordar que la cámara necesita iniciarse y esto puede llevarle unos instantes, nuestro bloque serpiente (que cargaba un archivo .py) espera el tiempo necesario.

Actualizando la acción “recorrer” ahora el código queda como en la siguiente imagen:



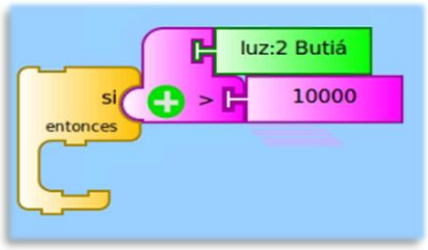
Segunda versión

Para esta versión solo modificaremos la acción identificar.

Como ya hemos visto conectamos el sensor de luz en cualquiera de los puertos, lo ideal es ubicar el sensor adelante ya que al momento de capturar la imagen nos posicionamos frente al objeto entonces el sensor debería informarnos si hay suficiente luz frente al objeto.

Imagen 11. Dos sensores distancias en los costados, y un sensor de luz en el medio.

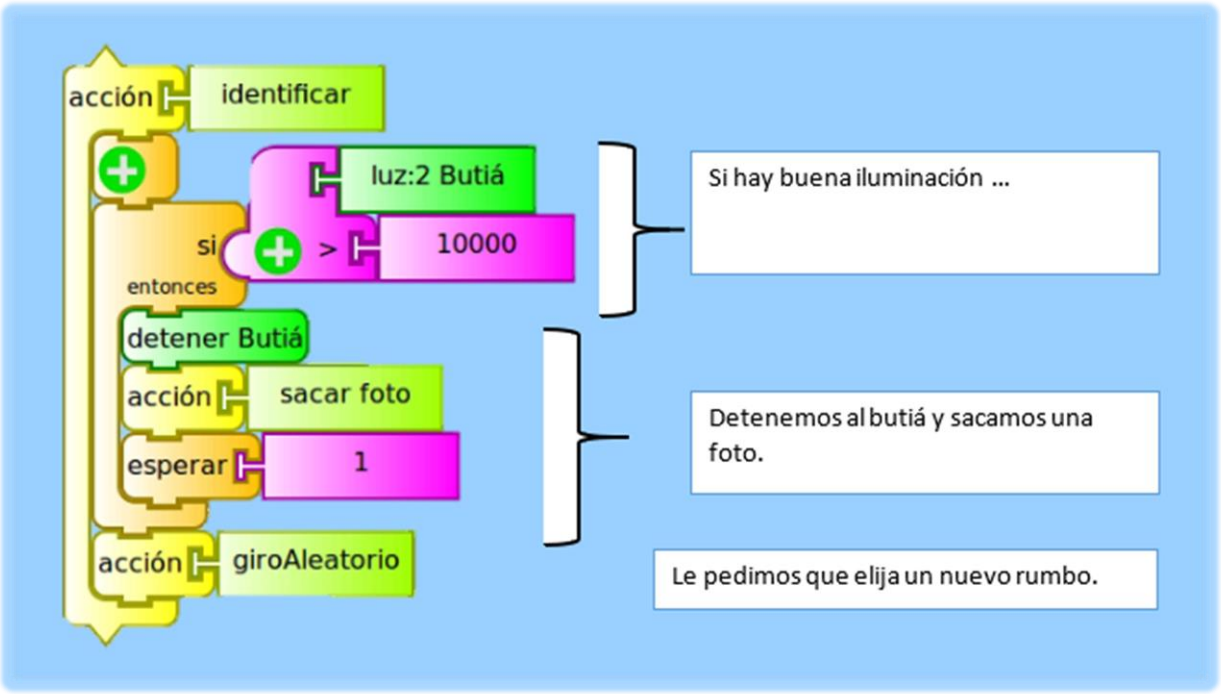
Así como en la parte dos utilizamos el bloque condicional “*si entonces*” para controlar la distancia, ahora lo utilizaremos para agregar la condición de la luz. Recordemos que en la estructura de control se debe agregar un bloque que permite escribir la condición lógica, estos están en la paleta: *operadores numéricos.*



En la parte uno, luego de hacer pruebas con nuestro sensor, dedujimos que en un ambiente donde este arrojaba valores superiores a 10000 las imágenes se veían bien. Así que nosotros utilizaremos este valor.

Resumiendo, hay que tener en cuenta que el objeto puede estar mal iluminado, y no tiene sentido sacar fotos muy oscuras, por lo que usando el sensor de luz, pedimos que tenga un valor lo suficientemente alto como para poder ver algo en las fotos. Si la luz finalmente lo permite, ordenamos al butiá detenerse y ejecutamos la acción sacar foto. Finalmente, ya sea que pudimos sacar la foto o no, seguimos hacia delante explorando nuevos objetos.

Con estas nuevas consideraciones modificamos la acción identificar:



Al finalizar la parte 3 finalmente tenemos cada acción implementada por separado, tenemos la acción “recorrer” (dentro de ella giro aleatorio e identificar que ya están terminadas en la parte tres) y “sacar foto”, por lo tanto el programa principal está completo.

Cuando queremos que el robot termine de espiar, podemos ir y apagar el interruptor o detenerlo agregando el bloque “detener butia”.

Finalmente guardamos nuestro programa, desde el menú de la barra superior vamos a archivo y luego guardar, seleccionamos la carpeta donde queremos guardar el mismo y el nombre del programa.

Anexo

Código Python para fotos

Creamos un archivo nuevo, main.py y escribimos:

```
1. import pygame
2. import pygame.camera
3. import time
4. import getpass
5. import sys
6. import datetime
7.
8. pygame.camera.init()
9. pygame.camera.list_cameras()
10. cam = pygame.camera.Camera("/dev/video0", (640, 480))
11. cam.start()
12. time.sleep(0.5)
13. img = cam.get_image()
14. pygame.image.save(img, "/home/"+getpass.getuser()+"/Escritorio/"+
    str(datetime.datetime.now()) +".png")
15. cam.stop()
```

Primero importamos las librerías necesarias, *pygame* y *pygame.camera* ya vienen instaladas al instalar el TurtleBots. Las librerías *time* y *datetime* nos permiten obtener la fecha y hora de sistema, para identificar en qué momento fueron sacar las fotos. Por último, *getpass* y *sys* son librerías que nos permiten utilizar funciones del sistema operativo, como conocer el nombre de usuario (para poder dejarle las fotos en su escritorio!).

Con las líneas 8,9 conseguimos la lista de cámaras de la computadora.

En la línea 10 y 11 elegimos la cámara (siendo *video0* la primera) y la resolución de las fotos.

Con la línea 11,12,13 obtenemos la foto. Es importante el *sleep(0.5)*, ya que la cámara puede demorar en iniciar y si nos apuramos la imagen puede salir oscura, movida o toda negra.

Finalmente en la línea 14 guardamos la foto en el escritorio del usuario, *getpass.getuser()* nos da el nombre. La función *str(datetime.datetime.now())* nos da un nombre único a cada foto, así que asegúrate que si lo cambias, lo hagas con otra función (hecha por ti o perteneciente al lenguaje) que varíe!.

Al terminar, detenemos la cámara con *cam.stop()*.

Si la carpeta elegida da error o prefieres otra ubicación, puedes cambiar “escritorio” por otra carpetas u otra ruta, por ejemplo, para guardar en una carpeta *imágenes* dentro de otra carpeta *butía,* en tu carpeta *home,* la línea 14 quedaría así:

pygame.image.save(img, "/home/"+getpass.getuser()+"/butia/imágenes"+ str(datetime.datetime.now()) +".png").

Se puede probar este código sin siquiera tener el turtlebots, y se puede reutilizar para otros proyectos. Para probarlo de manera independiente se necesita *python 2.7* y las librerías *pygame*. En una terminal, se puede ejecutar *python main.py* y la foto quedará en el escritorio.

OBSERVACIÓN:

Si las fotos no están apareciendo en el escritorio, asegúrate de tener la librería *pygame*, verifica también que la carpeta escritorio exista dentro de tu *home*, y por último prueba con otras resoluciones cambiando la línea 10 (por ejemplo 320x240).

Enlaces

Código Python y TurtleBots:
https://www.fing.edu.uy/inco/proyectos/butia/mediawiki/index.php/Archivo:Robot_espia.zip

Referencias

Armado del Kit butiá. (2020). Obtenido de https://docs.google.com/document/d/1VzC4wt_N7Fd2oPCEuOcGK96PS3KvVr-qzV8Gn01ISpU/edit#heading=h.nhtc8zalczai

Ciencia Historica. (2020). Obtenido de https://www.cienciahistorica.com/2017/01/03/los-piratas-llevaban-parche-en-el-ojo/

Manual de uso básico de TurtleBots y robot Butiá 2.0. (2020). Obtenido de https://www.centrosmec.gub.uy/innovaportal/file/823/1/manual-turtlebots---butia-2.0_05-11-18-de-a-una-pag.pdf

oftalvist. (2020). Obtenido de https://www.oftalvist.es/blog/punto-ciego-del-ojo-definicion-y-funcion/

Opportunity. (2020). Obtenido de https://elcomercio.pe/tecnologia/ciencias/opportunity-nasa-hazanas-robot-exploro-marte-15-anos-fotos-noticia-607738-noticia/?foto=8

science behind super powers. (2020). Obtenido de https://sciencebehindsuperpowers.weebly.com/echolocation.html

Sensor de distancia: gp2y0a21yk0F. (2020). Obtenido de (https://global.sharp/products/device/lineup/data/pdf/datasheet/gp2y0a21yk_e.pdf)