SABERES DIGITALES

AUTORIDADES

Presidente de la Nación
Mauricio Macri

Vicepresidenta de la Nación
Marta Gabriela Michetti

Jefe de Gabinete de Ministros
Marcos Peña

Ministro de Educación, Cultura, Ciencia y Tecnología
Alejandro Finocchiaro

Titular de la Unidad de Coordinación General del Ministerio de Educación, Cultura, Ciencia y Tecnología
Manuel Vidal

Subsecretario de Coordinación Administrativa
Javier Mezzamico

Director Ejecutivo INET
Leandro Goroyesky

Gerenta General de EDUCAR Sociedad del Estado
Liliana Casaleggio

Directora Nacional de Asuntos Federales
María José Licio Rinaldi

Director Nacional de Educación Técnico - Profesional
Fabián Prieto

Coordinador de Secundaria Técnica
Alejandro Anchava

Responsable de Formación Docente Inicial y Continua INET
Judit Schneider

Coordinador General En FoCo
Pablo Trangone

DRON ACUÁTICO

Ficha técnica

Nivel educativo	Secundario. Ciclo básico.

Descripción general	Construcción y programación de un dron que pueda desplazarse por la superficie del agua.
Niveles de complejidad	Nivel inicial: Construir con **impresión 3D** y una **Placa Arduino** un dron acuático capaz de ser programado para recorrer trayectorias previamente determinadas. Nivel avanzado: Complejizar el sistema agregando **sensores ultrasónicos** que le permitan al dron reconocer obstáculos en su camino y recalcular su trayectoria.

Listado de componentes:

Insumos	● 1 x Arduino UNO R3 ● 1 x Cable usb tipo B ● 1 x Protoboard ● 20 cables dupont macho hembra ● 20 cables dupont macho macho ● 3 x Sensor ultrasónico ● 1x Módulo motor (ULN2003) ● 2 x Motor DC 5v 1500 rpm ● 6 x Pila AA + Portapila ● 1x PLA
Equipamiento	● Soldador ● Estaño ● Alicate ● Pinza de punta ● Brusela
Otros requisitos	● Conexión a internet ● Descargar el programa “mblock3”

	http://www.mblock.cc/software-1/mblock/mblock3/

Presentación

Descripción ampliada del proyecto

El proyecto propone la construcción y programación de un dron que pueda desplazarse por la superficie del agua. Este será capaz de realizar distintos recorridos de manera automática. En primera instancia estará programado para recorrer un área previamente determinada de la manera más eficaz posible. En el nivel intermedio se incorporarán sensores para que el dron pueda esquivar obstáculos en su camino.

Al final de esta guía se puede encontrar un glosario donde se provee la información técnica necesaria para poder poner el proyecto en funcionamiento. El mismo cuenta con aclaraciones sobre los diversos elementos electrónicos involucrados así como también conceptos claves.

Objetivos

• Introducirse en algunas nociones de mecánica e hidráulica mediante el armado de un dron acuático.
• Comprender la lógica de los sistemas de movimiento.
• Comprender la lógica de la programación secuencial.
• Resolver el ensamble de piezas según las necesidades de diseño.

Desarrollo

Nivel Inicial

Un equipo de investigadores científicos desea poder examinar las impurezas que posee el agua del lago Traful. Para ello, en primera instancia decidieron desarrollar un dron acuático que pueda barrer toda la superficie del lago, para luego incorporar un dispositivo que pueda tomar las mediciones necesarias.

Dado que la superficie del lago es de 76 km^2, resolvieron que la mejor forma de conseguir que el dron la abarque por completo es programándolo para que realice trayectorias en las que recorra pequeñas áreas cada vez, hasta completar el área total.

Se construirá y programará un dron acuático capaz de realizar una trayectoria (previamente programada en la placa Arduino) que abarque un área cerrada.

Paso 1 - Impresión de las piezas.

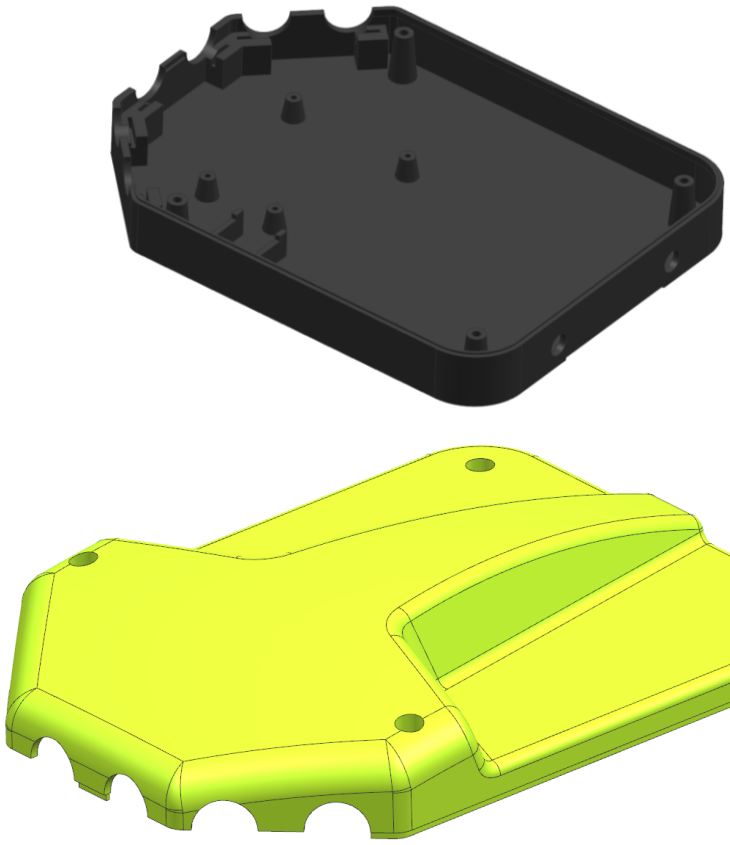
Para armar la maqueta del semáforo, se deben imprimir las piezas en la impresora 3D. El modelo 3D del semáforo se puede descargar de forma libre y gratuita en el siguiente enlace: https://enfoco.net.ar/sd

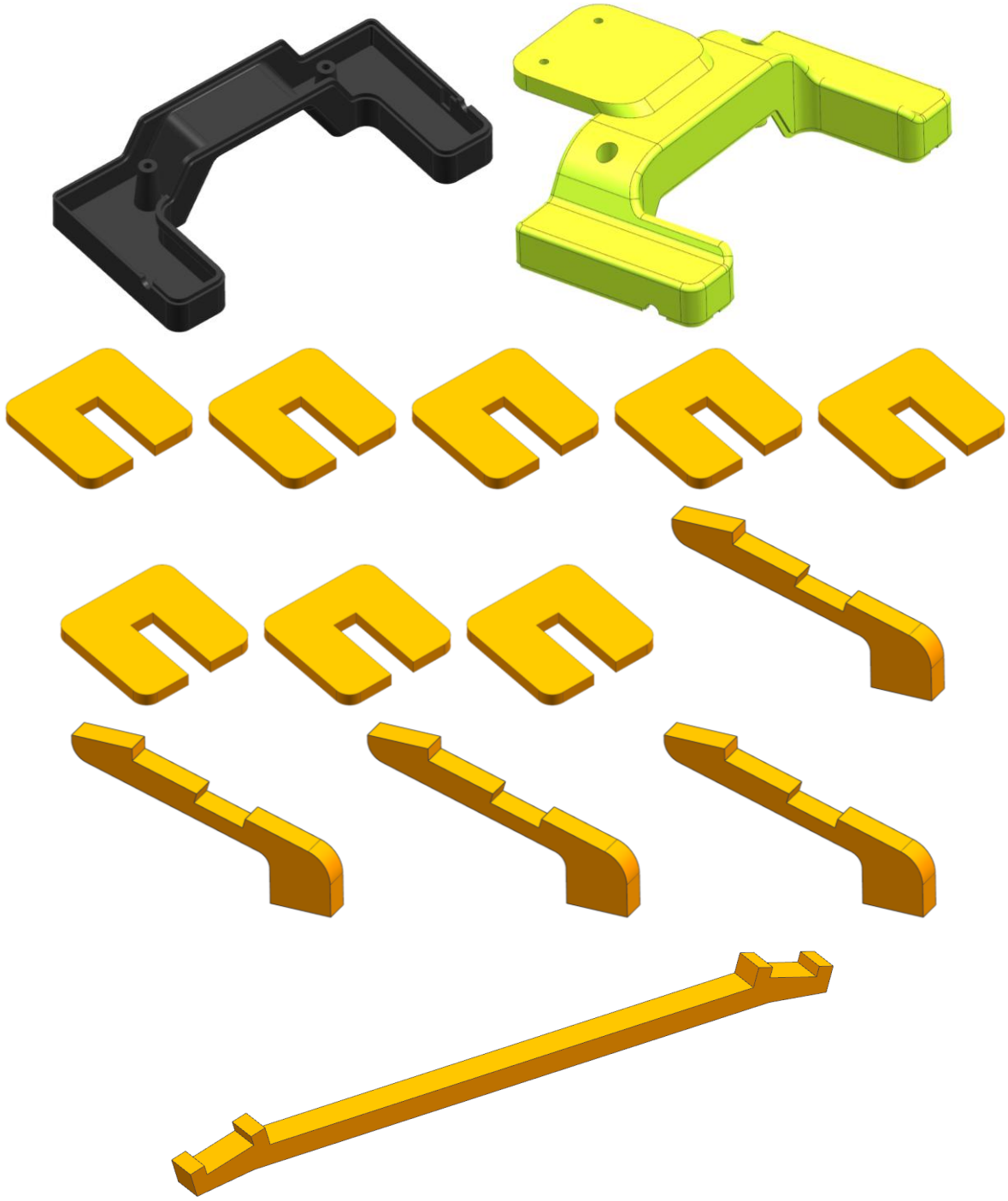
Una vez descargado el modelo, lo imprimimos con la impresora 3D. Cuando estén listas todas las piezas, las ensamblamos para construir el sistema mecánico del dron.

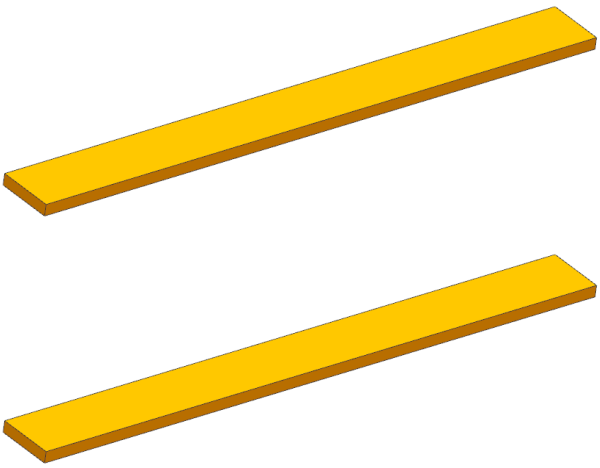
En caso de querer realizar una modificación en el modelo, independientemente del programa de modelado que utilicemos, debemos exportar nuestras piezas en formato .stl.

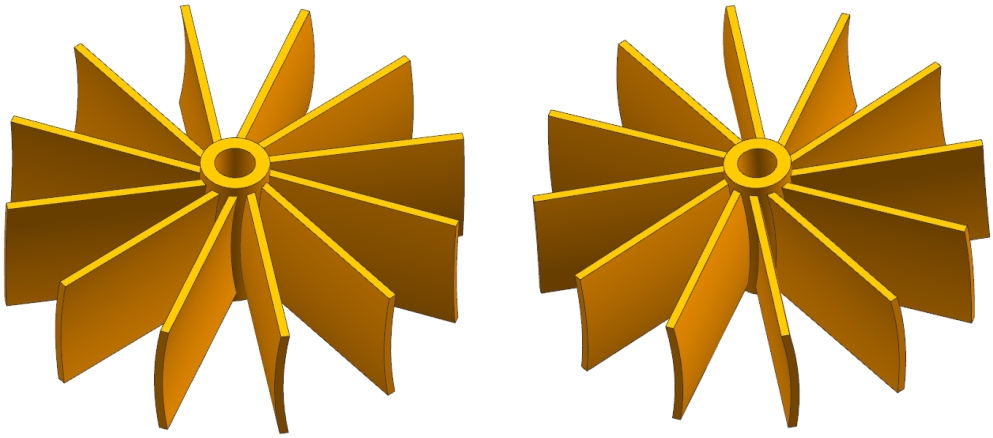
El .stl es un formato de archivo informático de diseño asistido por computadora (CAD) que define la geometría de objetos sólidos 3D. Es el formato más popular a la hora de intercambiar digitalmente modelos de objetos para ser impresos en 3D.

Las piezas son las siguientes:





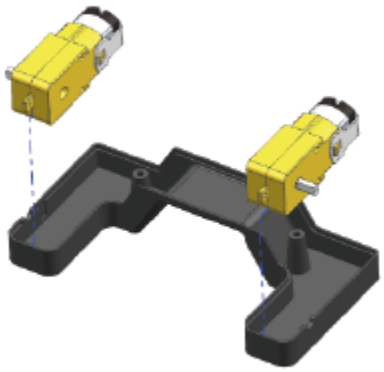




- Carcasa principal inferior (x1)
- Caracasa principal superior (x1)
- Carcasa motores inferior (x1)
- Carcasa motores superior (x1)
- Placa (x8)
- Soporte (x4)
- Vínculo lateral (x2)
- Vínculo transversal (x1)
- Rotor de paletas derecho (x1)
- Rotor de paletas izquierdo (x1)

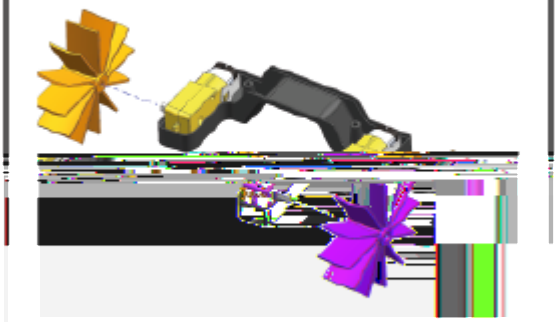
Paso 2 - Ensamblar las piezas

1

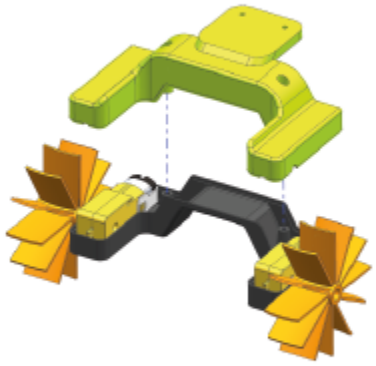


Colocar los motores amarillos en la carcasa inferior.

2



3

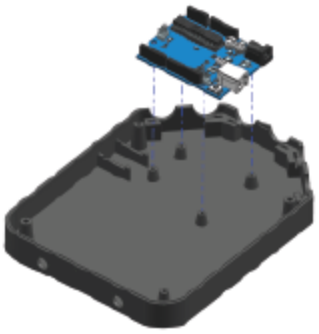


Vincular la carcasa superior de los motores con la carcasa inferior.

4

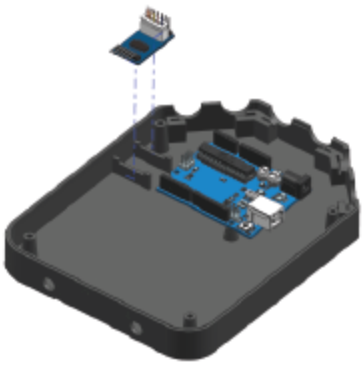
Así debe verse el conjunto de motores ya ensamblado.

5



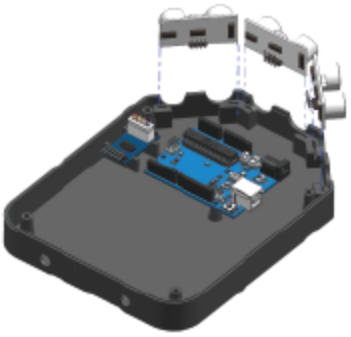
Montar la placa Arduino Uno en la carcasa inferior.

6



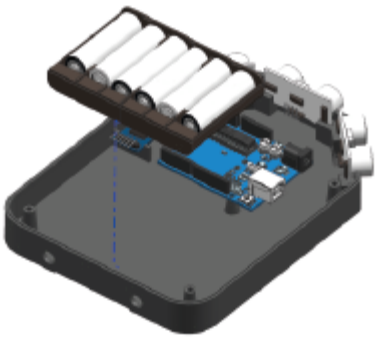
Montar la plaqueta UNL2003 en la carcasa inferior.

7



Colocar los sensores ultrasónicos en los alojamientos de la carcasa inferior.

8



Pegar con un adhesivo el porta pilas en la carcasa inferior.

9

Así debe verse el conjunto electrónico ya montado.

10

Vincular el conjunto de motores con el conjunto electrónico.

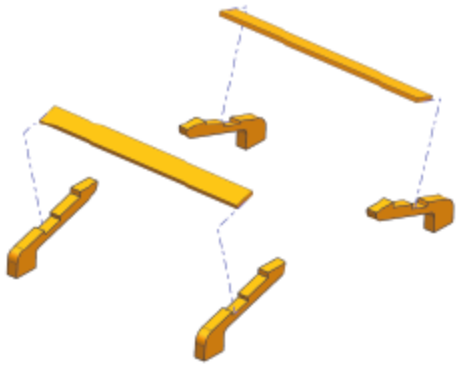
11

Así debe verse el conjunto de motores ya montado con el conjunto electrónico.

12

Montar la carcasa superior con la inferior.

13



Encastrar los vínculos laterales de la estructura con los soportes.

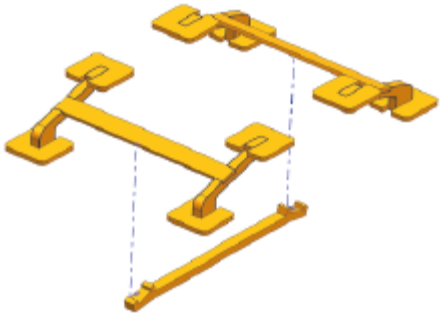
14

Encastrar las placas inferiores con los soportes.

15

Encastrar las placas superiores con los soportes.

16



Encastrar el vínculo transversal con ambas estructuras.

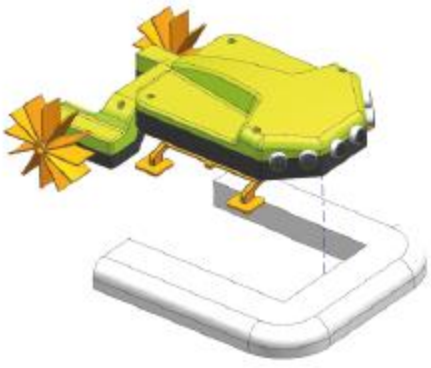
17

Así debe verse la estructura ensamblada.

18

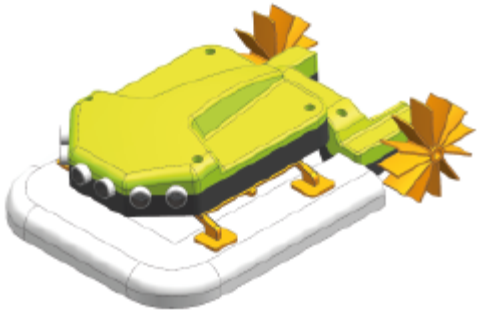
Pegar la estructura con un adhesivo a la carcasa inferior.

19



Pegar la estructura con un adhesivo a un flotador de telgopor.

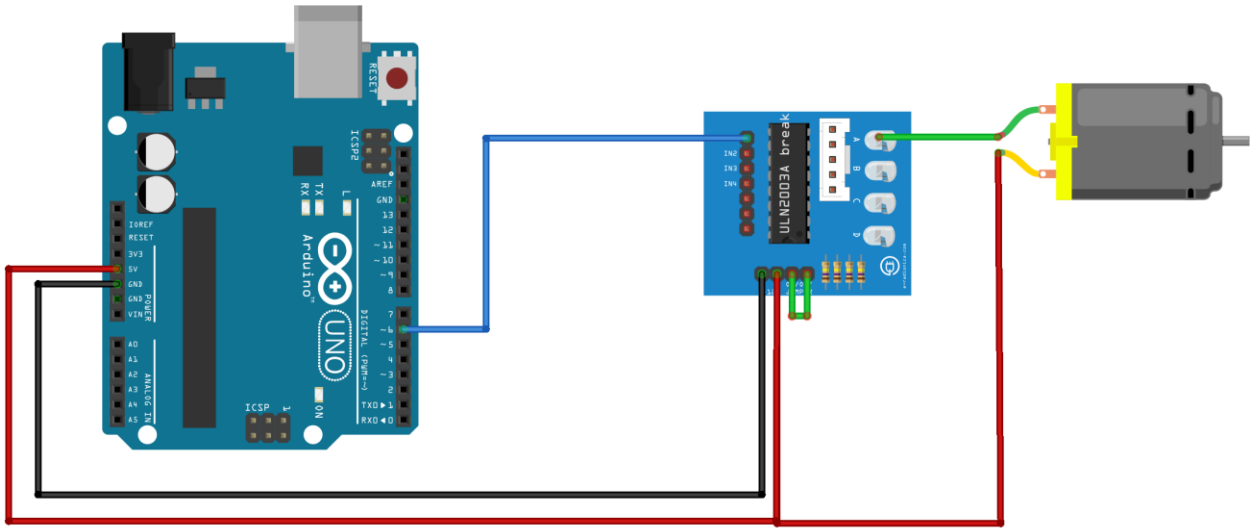
20



Así se ve el dron acuático ya ensamblado.

Paso 3 - Conectar un motor DC

Para poder hacer que los motores del dron acuático se muevan, necesitamos utilizar un controlador para motores ULN2003. Este nos permitirá hacer girar el motor hacia adelante y controlar la velocidad del movimiento del dron. Primero, se probará el funcionamiento del dispositivo con un solo motor, para entender cómo funciona. Comenzamos realizando las conexiones como se ve en la siguiente imagen:

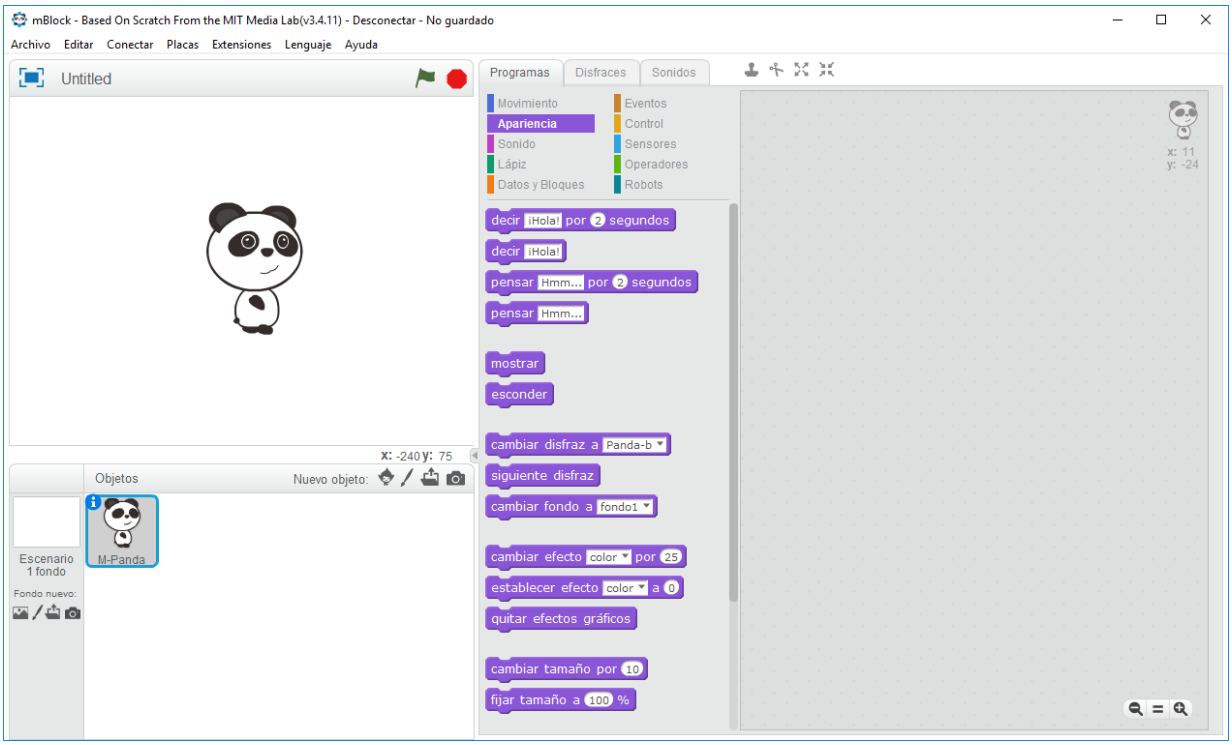


El ULN2003 es un circuito integrado que nos servirá para realizar la conversión de una señal de control de baja potencia (salida digital del Arduino) a un señal con la potencia necesaria para poder hacer se mueva el motor.
No es posible conectar motores DC directamente a las salidas de Arduino ya que los mismos requieren mayor nivel de potencia.

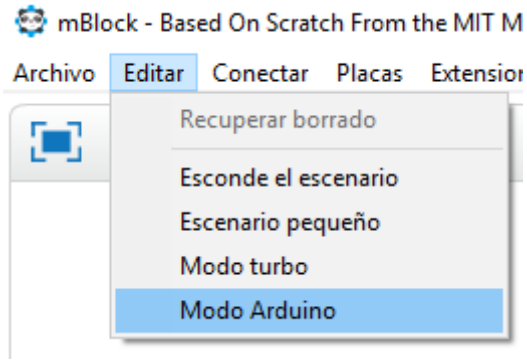
Paso 4 - Programar el funcionamiento del motor

La programación la realizaremos con mBlock3, un entorno de programación basado en Scratch2 que permite programar proyectos de Arduino utilizando bloques. Se puede descargar siguiendo este enlace: http://www.mblock.cc/software-1/mblock/mblock3/

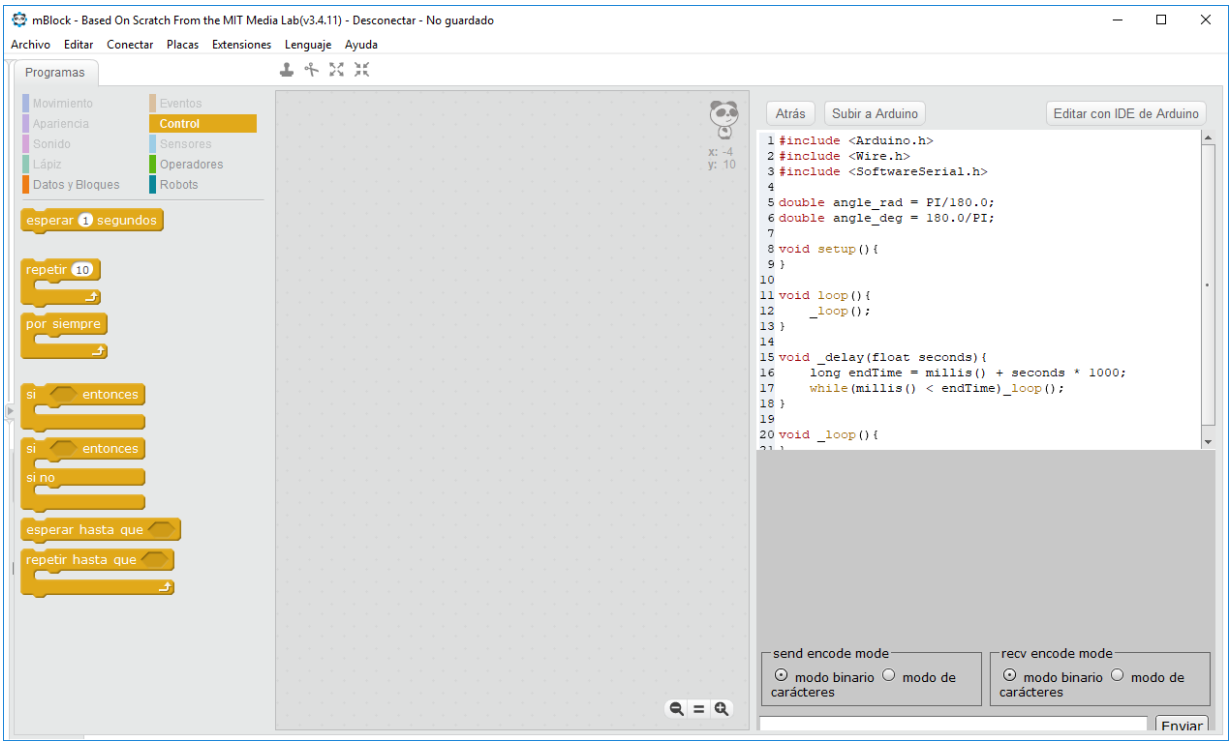
Cuando abrimos mBlock3, vemos una pantalla como la siguiente:



Para programar un proyecto de Arduino con mBlock3 debemos seleccionar el “Modo Arduino” desde el menú.

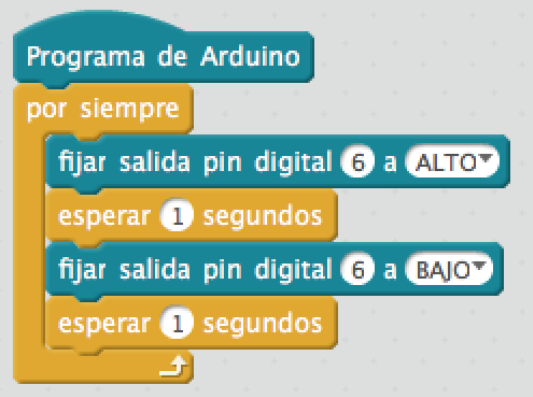


Al seleccionar este modo, el programa cambiará de aspecto. Se verá un área en el centro que es la que utilizaremos para programar con bloques. A la derecha se verá un campo donde aparecerá el código escrito que le corresponde a los bloques que están en el centro. Este código se irá escribiendo automáticamente a medida que se vaya armando el programa con los bloques.



Los bloques están agrupados por categorías. En este caso, se usarán bloques de las categorías “**Robots”**, “**Control”**, “**Operadores”** y “**Datos y Bloques”**. Cuando seleccionamos una de estas categorías, se pueden visualizar todos los bloques que pertenecen a ese grupo.

Después de familiarizarnos con el sistema, vamos a empezar escribir un programa que permita encender y apagar el motor por un determinado tiempo. Debería ser similar al siguiente modelo:



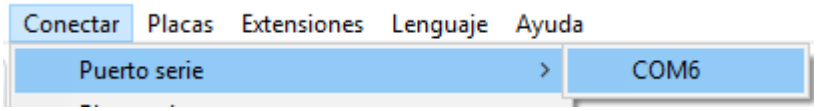
Veremos que a la derecha se muestra el código escrito que corresponde a este programa:

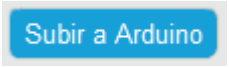
```
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>

double angle_rad = PI/180.0;
double angle_deg = 180.0/PI;

void setup(){
    pinMode(6,OUTPUT);
}

void loop(){
    digitalWrite(6,1);
    _delay(1);
    digitalWrite(6,0);
    _delay(1);
    _loop();
}
void _delay(float seconds){
    long endTime = millis() + seconds * 1000;
    while(millis() < endTime)_loop();
}
void _loop()
{

}
```

Paso 5 - Subir el código a la placa Arduino

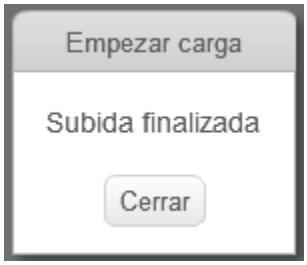
Para subir el código de nuestro programa a la placa Arduino, necesitamos:

1. Conectar la placa Arduino a la entrada USB.
2. Chequear que en el menú “Placas” esté seleccionado “Arduino Uno”.
3. Seleccionar el puerto serie al que está conectada la placa.



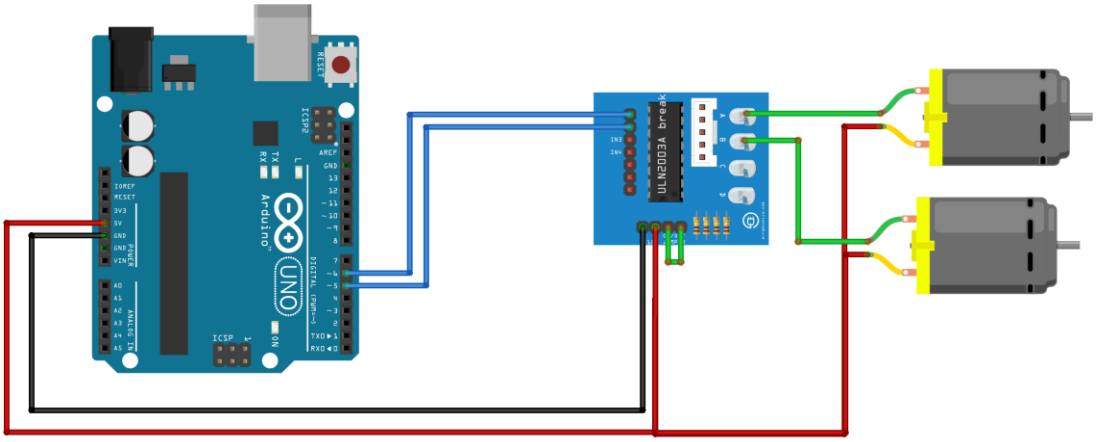
4. Clickear el botón 

Al terminar de subir nuestro código, veremos este mensaje



Paso 6 - Conectar ambos motores

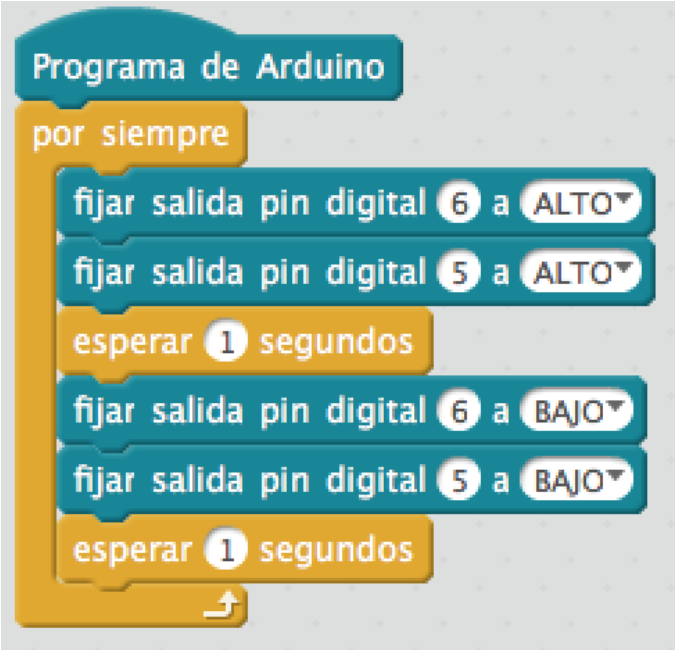
Ahora vamos a realizar la conexión del segundo motor y conocer cómo debe programarse el movimiento de cada motor para desplazar el dron acuático.



Paso 7 - Controlar ambos motores

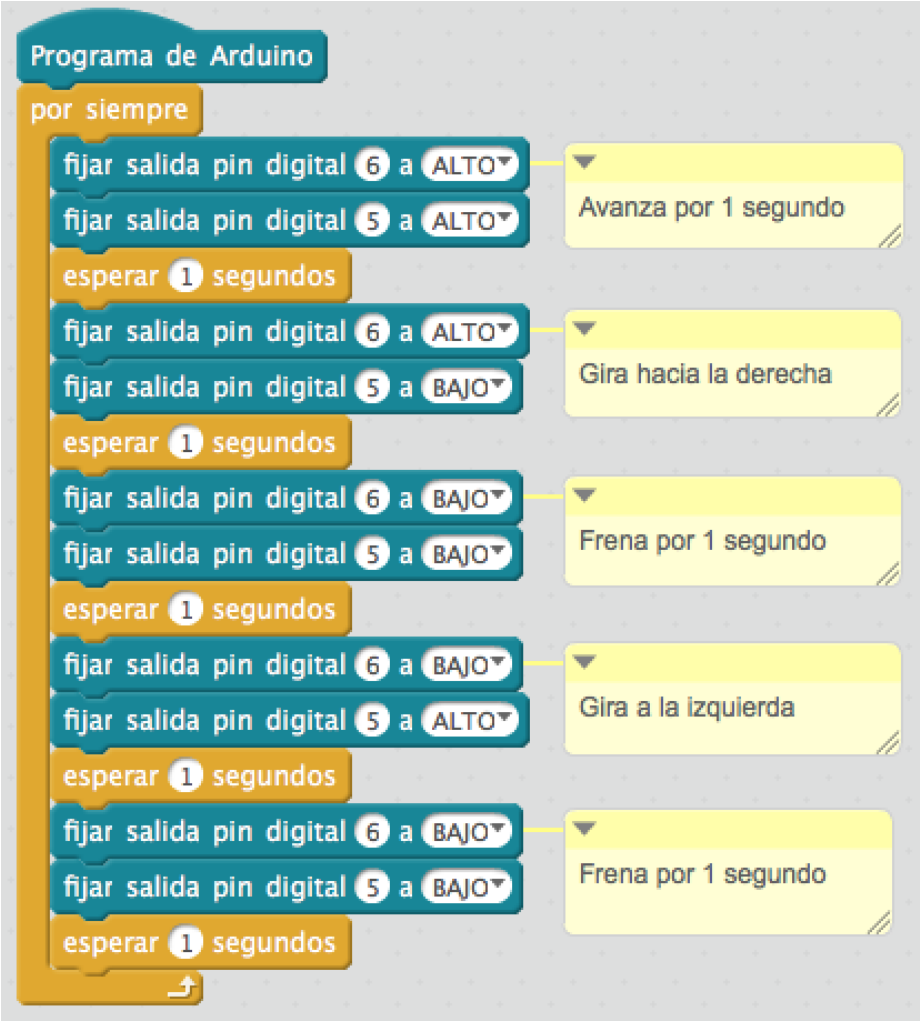
Para que el dron acuático vaya hacia adelante debemos prender ambos motores. Al accionarlos, siempre debemos tener en cuenta la dirección en la que girará el motor, que está determinada por el modo en que están conectados sus cables.

Es importante saber que si necesitamos que el motor gire en el sentido contrario debemos invertir la conexión de los cables. Un programa que accione ambos motores de forma conjunta debería verse similar al siguiente modelo:



Paso 8 - Programar los giros del dron

Para que el dron gire hacia la derecha, el motor de la izquierda debe girar y el de la derecha debe frenar. Para que gire hacia la izquierda se debe programar un accionar exactamente opuesto: el motor de la derecha debe girar mientras el de la izquierda debe frenar. Vamos a realizar un programa que nos permita hacer que el dron vaya hacia adelante, gire a la derecha, frene, gire hacia la izquierda y frene. Este quedaría como se ve a continuación:



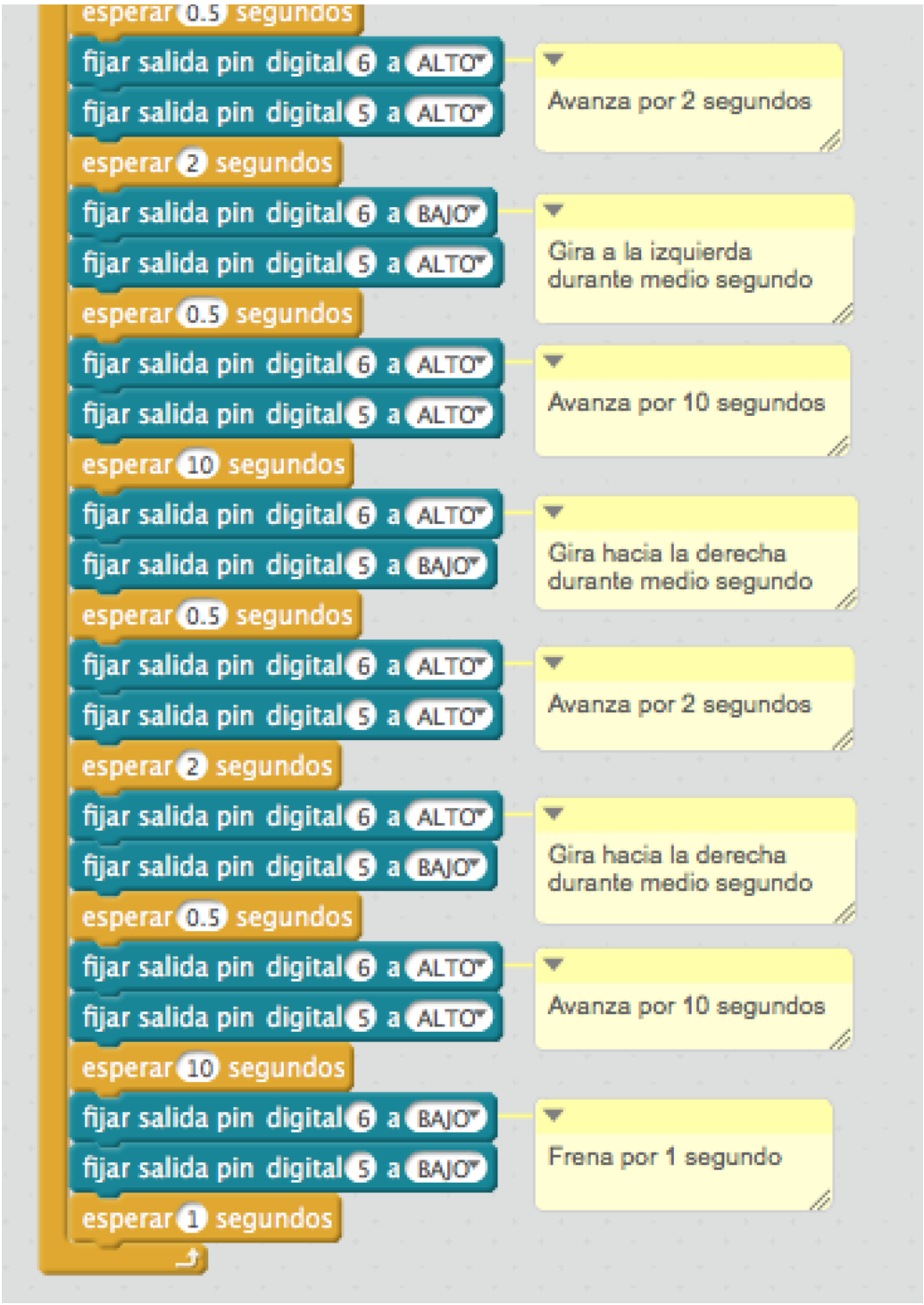
Paso 9 - Programar el movimiento del dron acuático

Para que el dron pueda recolectar las muestras del lago, vamos a establecer el recorrido de manera tal que le permita abarcar la mayor cantidad de superficie posible. Por ejemplo:

La distancia total que recorrerá el dron estará determinada por el tiempo de recorrido que le asignemos para moverse entre cada uno de los bloques. El recorrido estará planteado en función de cuánto tiempo debe recorrer en dron en cada dirección antes de cambiar a otra.
Vamos plantear las acciones principales que debe realizar el dron acuático para llegar a su destino como se muestra en el croquis:

1. Avanzar
2. Girar a la derecha
3. Avanzar
4. Girar a la derecha
5. Avanzar
6. Girar a la izquierda
7. Avanzar
8. Girar a la izquierda
9. Avanzar
10. Girar a la derecha
11. Avanzar
12. Girar a la derecha
13. Avanzar
14. Frenar

El programa que indique la realización de esta secuencia de movimiento nos debería quedar de la siguiente forma:


esperar 0.5 segundos
fijar salida pin digital 6 a ALTO
fijar salida pin digital 5 a ALTO
esperar 2 segundos
Avanza por 2 segundos
fijar salida pin digital 6 a BAJO
fijar salida pin digital 5 a ALTO
esperar 0.5 segundos
Gira a la izquierda durante medio segundo
fijar salida pin digital 6 a ALTO
fijar salida pin digital 5 a ALTO
esperar 10 segundos
Avanza por 10 segundos
fijar salida pin digital 6 a ALTO
fijar salida pin digital 5 a BAJO
esperar 0.5 segundos
Gira hacia la derecha durante medio segundo
fijar salida pin digital 6 a ALTO
fijar salida pin digital 5 a ALTO
esperar 2 segundos
Avanza por 2 segundos
fijar salida pin digital 6 a ALTO
fijar salida pin digital 5 a BAJO
esperar 0.5 segundos
Gira hacia la derecha durante medio segundo
fijar salida pin digital 6 a ALTO
fijar salida pin digital 5 a ALTO
esperar 10 segundos
Avanza por 10 segundos
fijar salida pin digital 6 a BAJO
fijar salida pin digital 5 a BAJO
esperar 1 segundos
Frena por 1 segundo

Veremos que a la derecha se muestra el código escrito que corresponde a este programa:

```
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>

double angle_rad = PI/180.0;
double angle_deg = 180.0/PI;

void setup(){
    pinMode(6,OUTPUT);
    pinMode(5,OUTPUT);
}

void loop(){
    digitalWrite(6,1);
    digitalWrite(5,1);
    _delay(10);
    digitalWrite(6,1);
    digitalWrite(5,0);
    _delay(0.5);
    digitalWrite(6,1);
    digitalWrite(5,1);
    _delay(2);
    digitalWrite(6,1);
    digitalWrite(5,0);
    _delay(0.5);
    digitalWrite(6,1);
    digitalWrite(5,1);
    _delay(10);
    digitalWrite(6,0);
    digitalWrite(5,1);
    _delay(0.5);
    digitalWrite(6,1);
    digitalWrite(5,1);
    _delay(2);
    digitalWrite(6,0);
    digitalWrite(5,1);
    _delay(0.5);
```

```
    digitalWrite(6,1);
    digitalWrite(5,1);
    _delay(10);
    digitalWrite(6,1);
    digitalWrite(5,0);
    _delay(0.5);
    digitalWrite(6,1);
    digitalWrite(5,1);
    _delay(2);
    digitalWrite(6,1);
    digitalWrite(5,0);
    _delay(0.5);
    digitalWrite(6,1);
    digitalWrite(5,1);
    _delay(10);
    digitalWrite(6,0);
    digitalWrite(5,0);
    _delay(1);
    _loop();
}

void _delay(float seconds){
    long endTime = millis() + seconds * 1000;
    while(millis() < endTime)_loop();
}

void _loop(){
}
```

Nivel Avanzado

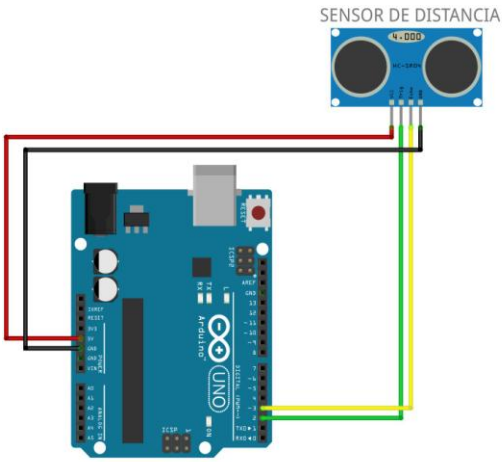
Luego de realizar algunos recorridos de prueba con el dron, los científicos notaron que, en algunos casos, sus trayectorias se veían interrumpidas. Descubrieron que esto se debía a que los guardaparques depositan boyas para el control del nivel del lago en ciertos puntos. Es por eso que decidieron desarrollar una nueva versión del dron acuático que sea capaz de percibir los objetos que se encuentran en su camino a una distancia considerable, pueda cambiar su rumbo para esquivarlos y luego volver a su trayectoria inicial.

Se implementará el uso de sensores ultrasónicos para que el dron pueda detectar objetos a la proximidad. La información que obtengan los sensores se enviará a la placa Arduino para que active la programación de la nueva trayectoria. Luego de ser esquivado el objeto en cuestión el dron retomará la trayectoria inicial.

Paso 1 - Conectar el sensor ultrasónico

Se incorpora un sensor de distancia ultrasónico que le permita al dron acuático cambiar el rumbo y volver a la trayectoria inicial. Conectaremos el *pin Trigger* del sensor al *pin* 2 del arduino y el *pin echo del sensor* al *pin* 3 del arduino. Todo con su correspondiente alimentación de energía (5 Volt y *GND*).

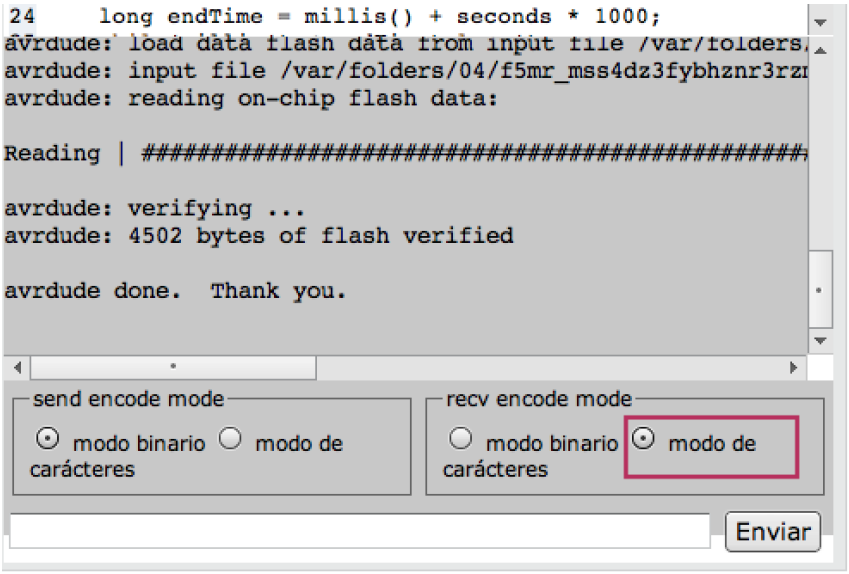
Sensor de distancia ultrasónico: Es un tipo de sensor que se utiliza para medir distancias. El principio de funcionamiento del mismo se basa en emitir un pulso de sonido ultrasónico y medir el tiempo que pasa hasta registrar que dicho pulso regresó a la fuente tras rebotar en un obstáculo. El tiempo transcurrido es directamente proporcional a la distancia que se encuentra el objeto.



Paso 2 - Obtener la distancia del sensor

En primer lugar, debemos familiarizarnos con la forma de medición del sensor ultrasónico.

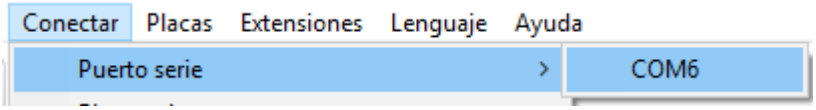
Podemos visualizar los valores de registro del sensor en la consola del programa, que se encuentra en la esquina inferior derecha de la pantalla. Es importante seleccionar “modo de caracteres” para el modo de recepción de los datos, como se muestra en la imagen:



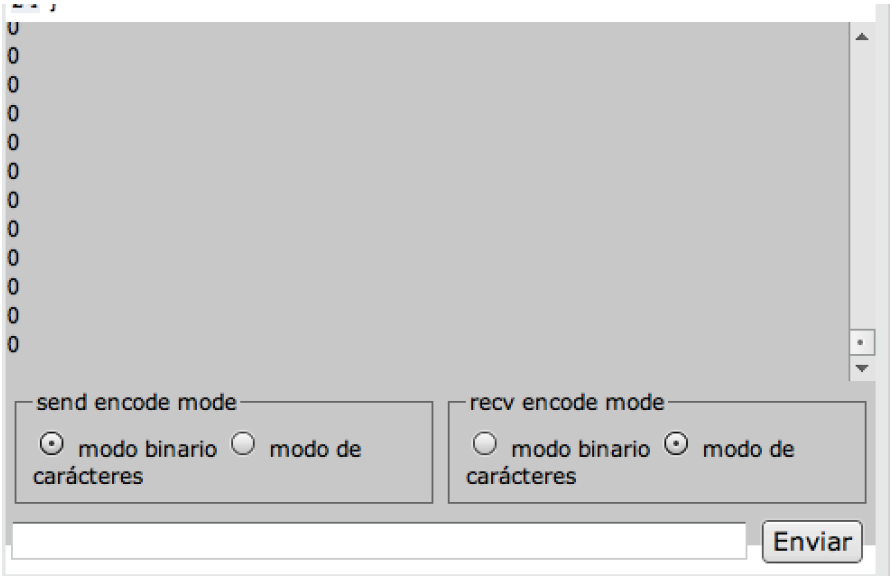
Para enviar los datos que toma el sensor a la consola se utiliza el bloque “Escribir en el serial el texto”. A este bloque se le agrega el bloque “Leer el sensor ultrasónico”. Es importante que el bloque “Por siempre” contenga a los dos bloques anteriores ya que necesitamos que el sistema actualice constantemente el valor de lectura del sensor.

Paso 3 - Activar el envío de datos a la consola

Una vez que nuestro programa esté cargado, debemos volver a conectar nuestra placa para que se envíen los datos a la consola.

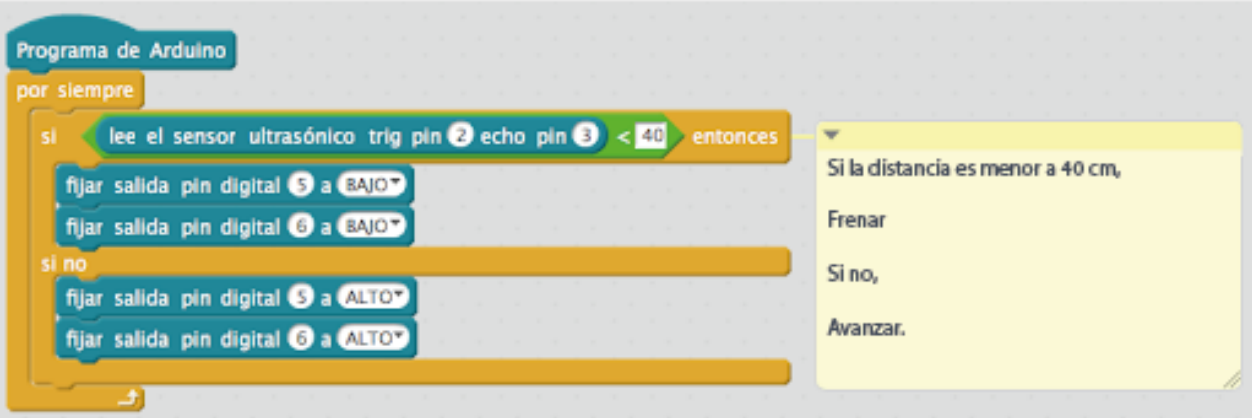


Finalmente podemos visualizar los valores del sensor en el programa. Con los potenciómetros de ajuste podemos calibrar la sensibilidad de medición y el tiempo que habrá entre cada lectura hasta los parámetros se ajusten a nuestras necesidades.



Paso 4 - Controlar los motores según la distancia

Vamos a controlar el avance del dron (frenando o manteniendo activos sus motores) en función de la distancia a la que se encuentran las boyas. En el ejemplo que se ve a continuación hacemos que frenen los motores si la distancia de medición del ultrasonido es menor a 40 cm. Si no, hacemos que los motores avancen.

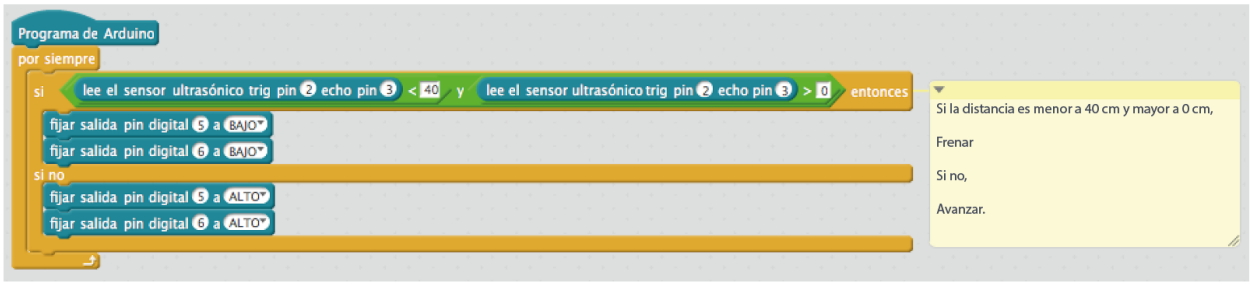


El dron debe frenar sólo si la distancia que mide el sensor es menor, en este caso, a 40 cm. En caso contrario, deberá avanzar. El bloque que nos permite evaluar si la distancia es “menor que” un valor lo encontraremos dentro de “Operadores”.

Paso 5 - Ajustar la medición

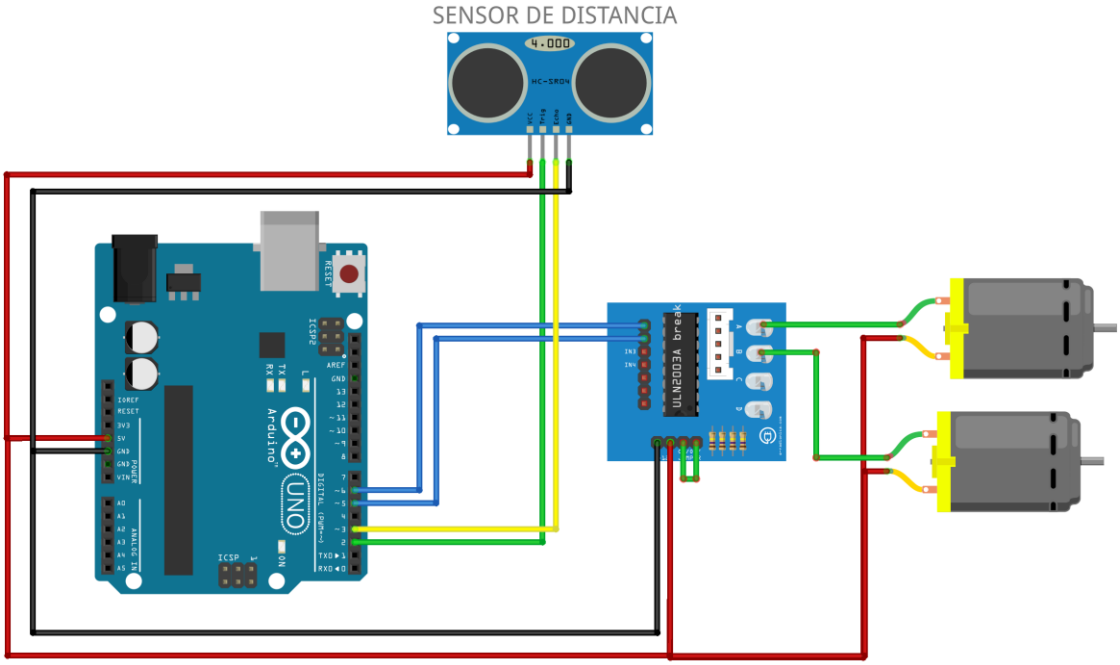
Debemos tener en cuenta que la mayoría de los ultrasonidos nos arrojan un valor de distancia igual a 0 (cero) cuando la medición se pasa de su máximo alcance. Esto quiere decir que si nuestro ultrasónico tiene una medición máxima de 400 cm y el objeto cuya distancia queremos medir se encuentra a 401 cm, la medición que veremos será de 0 cm.

Esto no quiere decir que el dron se encuentra a 0 cm de la boya, sino que la boya está “fuera de rango”, es decir, fuera del rango de medición del sensor. Pero este valor se puede dar a confusión, dado que, en algunas ocasiones, efectivamente habrá boyas a 0 cm de nuestro dron. Para solucionar esto, utilizaremos una condición en nuestro programa para que no tenga en cuenta al valor 0 (cero). En este caso, lo programaremos para que el dron frene si la distancia entre este y el objeto es menor a 40 cm y mayor a 0 cm, como se ve a continuación:



Paso 6 - Agregar los motores

Utilizaremos el sensor ultrasónico para que el dron pueda detectar las boyas y esquivarlas. Agregaremos ahora el circuito de los motores al circuito del sensor ultrasónico. Nuestro circuito completo quedará, entonces, de la siguiente manera:



Paso 6 - Programar el movimiento

Para que el dron pueda realizar su recorrido sin chocar con objetos presentes en su trayectoria, vamos a establecer el recorrido de manera que pueda abarcar la mayor cantidad de superficie posible y también esquivar las boyas. El recorrido podría verse como en el siguiente ejemplo:

La distancia total que recorrerá el dron estará determinada por el tiempo de recorrido que le asignemos para moverse entre cada uno de los bloques. Como antes, el recorrido estará planteado en función de cuánto tiempo debe recorrer en dron en cada dirección antes de cambiar a otra.
Vamos plantear las acciones principales que debe realizar el dron acuático para llegar a su destino como se muestra en el croquis esquivando las boyas:

15. Avanzar
16. Boya!
17. Girar a la izquierda
18. Avanzar
19. Girar a la izquierda
20. Avanzar
21. Boya!
22. Girar a la derecha
23. Avanzar
24. Girar a la derecha
25. Avanzar
26. Boya!
27. Girar a la izquierda
28. Avanzar
29. Girar a la izquierda
30. Avanzar
31. Frenar

La programación nos quedaría como se ve a continuación:

Veremos que a la derecha se muestra el código escrito que corresponde a este programa:

```
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>

double angle_rad = PI/180.0;
```

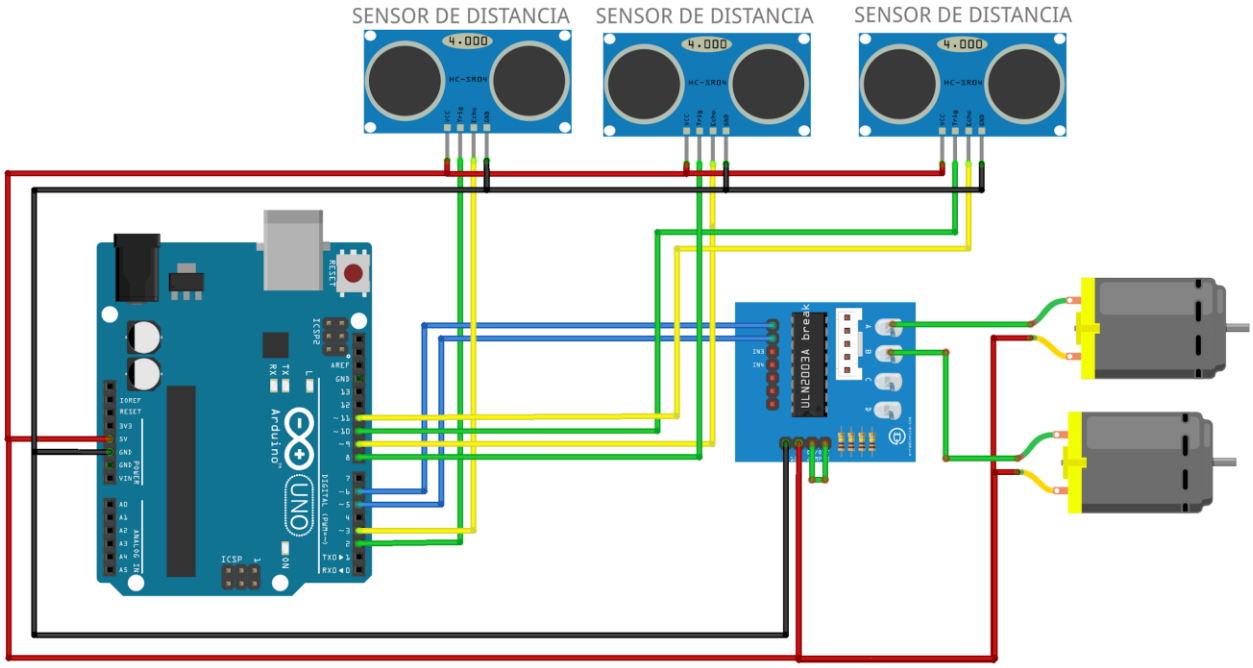
```
double angle_deg = 180.0/PI;
float getDistance(int trig,int echo){
    pinMode(trig,OUTPUT);
    digitalWrite(trig,LOW);
    delayMicroseconds(2);
    digitalWrite(trig,HIGH);
    delayMicroseconds(10);
    digitalWrite(trig,LOW);
    pinMode(echo, INPUT);
    return pulseIn(echo,HIGH,30000)/58.0;
}

void setup(){
    pinMode(5,OUTPUT);
    pinMode(6,OUTPUT);
    digitalWrite(5,1);
    digitalWrite(6,1);
    while(!(((getDistance(2,3)) < (40)) && ((getDistance(2,3)) >
(0))))
    {
        _loop();
    }
    digitalWrite(5,1);
    digitalWrite(6,0);
    _delay(0.5);
    digitalWrite(5,1);
    digitalWrite(6,1);
    _delay(2);
    digitalWrite(5,1);
    digitalWrite(6,0);
    _delay(0.5);
    digitalWrite(5,1);
    digitalWrite(6,1);
    while(!(((getDistance(2,3)) < (40)) && ((getDistance(2,3)) >
(0))))
    {
        _loop();
    }
    digitalWrite(5,0);
```

```
    digitalWrite(6,1);
    _delay(0.5);
    digitalWrite(5,1);
    digitalWrite(6,1);
    _delay(2);
    digitalWrite(5,0);
    digitalWrite(6,1);
    _delay(0.5);
    digitalWrite(5,1);
    digitalWrite(6,1);
    while(!(((getDistance(2,3)) < (40)) && ((getDistance(2,3)) >
(0))))
    {
        _loop();
    }
    digitalWrite(5,1);
    digitalWrite(6,0);
    _delay(0.5);
    digitalWrite(5,1);
    digitalWrite(6,1);
    _delay(2);
    digitalWrite(5,1);
    digitalWrite(6,0);
    _delay(0.5);
    digitalWrite(5,1);
    digitalWrite(6,1);
    _delay(10);
    digitalWrite(5,0);
    digitalWrite(6,0);
}

void loop(){
    _loop();
}

void _delay(float seconds){
    long endTime = millis() + seconds * 1000;
    while(millis() < endTime)_loop();
}
```

```
void _loop(){
}
```

Paso 7 - Incorporar dos sensores ultrasónicos más

Podemos haber notado que el sensor sólo registra objetos que se encuentren frente a él . Eso puede traer algunos problemas a la hora de hacer navegar a nuestro dron acuático. Por eso, agregaremos 2 sensores más para poder abarcar un rango más amplio de espacio. Ahora nos quedará el primer sensor en los pines 2 y 3, el segundo en los pines 6 y 7 y el tercero en los pines 10 y 11.



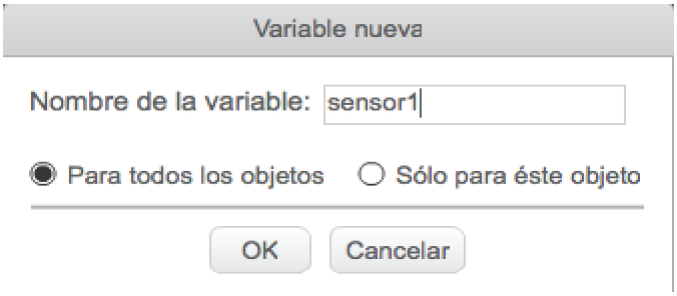
Paso 8 - Agregar la lectura de los ultrasonidos

El programa ahora deberá esperar hasta que se detecte un obstáculo en cualquiera de los tres sensores (el sensor 1 o el sensor 2 o el sensor 3) para realizar sus movimientos. Para programar este cambio utilizamos el bloque “o”, que indica que para accionar se debe evaluar si sucede una cosa u otra.

Paso 9 - Crear variables

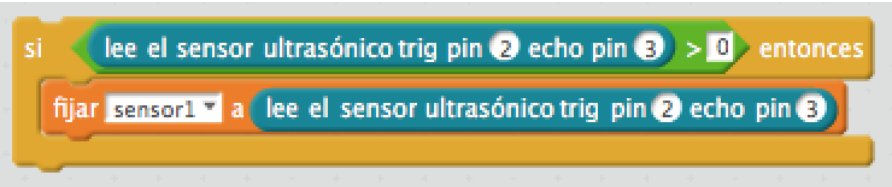
Para armar el bloque que indica que se deben considerar todos los sensores, utilizaremos variables. Vamos a crear 3 variables, una por cada sensor.

A la primera le pondremos como nombre, por ejemplo sensor1. Luego realizaremos dos variables más (sensor2 y sensor3).



Una variable es un espacio en la memoria que nos permite guardar un dato para ser leído en otra instancia del programa. Es importante tener en cuenta al momento de crear una variable que su nombre no puede comenzar con un número ni contener espacios.

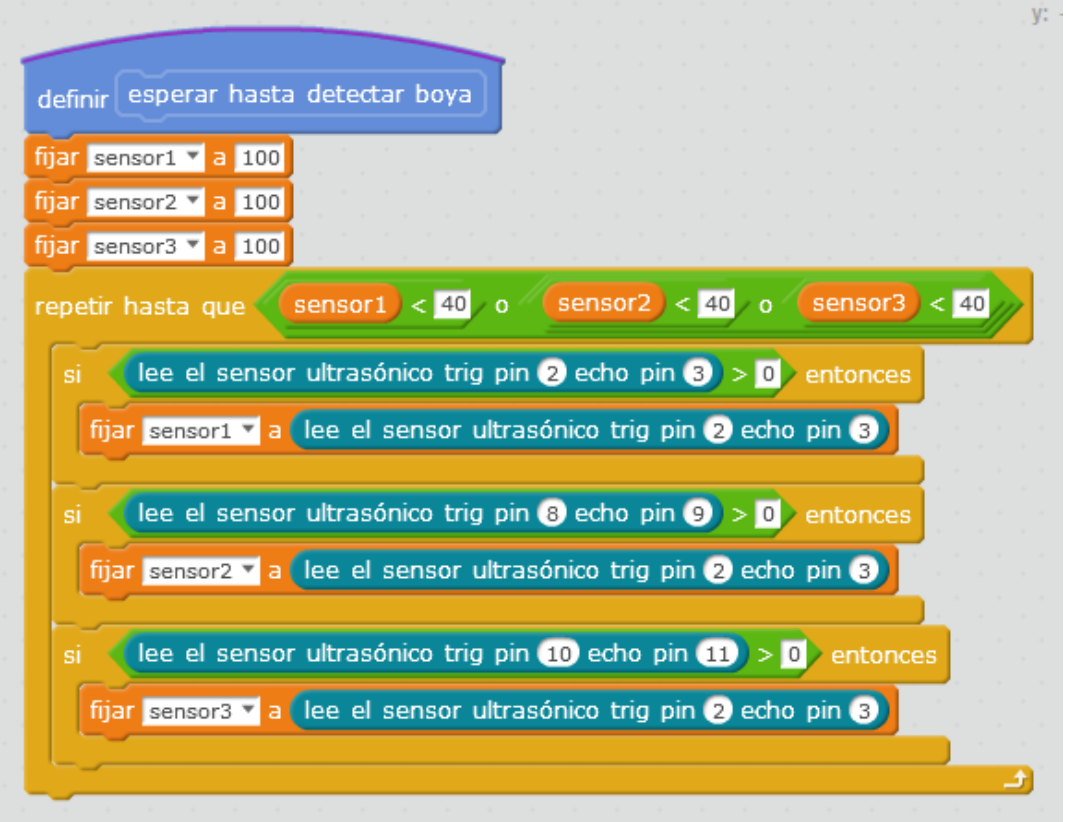
A esta variable le asignaremos el valor de la medición del ultrasonido, pero indicaremos que la considere sólo si es mayor a 0. De esta forma evitamos que la distancia del sensor nos de 0 cuando no tiene ningún obstáculo a la vista. La programación de esta parte debería quedar como se ve a continuación:



Paso 10 - Programar los tres sensores

Realizamos lo mismo para cada uno de los sensores. Luego haremos que el sistema revise si el valor de cada una de estas variables es menor a 40 cm. para ver si el dron debe cambiar o no su dirección. Notamos que al comienzo se incializan las tres variables para evitar errores en caso de que hayan quedado cargadas con los valores de una lectura anterior.

Debido a la extensión de este segmento de código y el hecho de que lo vamos a utilizar repetidas veces a lo largo de nuestro programa podemos crear un nuevo bloque o procedimiento que integre este código y luego lo podamos invocar de manera sencilla. El botón para crear bloques se encuentra en el mismo espacio que el de crear variables. En este caso lo hemos llamado “esperar hasta detectar boya”.



Finalmente, el programa completo con todos los bloques quedará de la siguiente manera:


Programade Arduino
fijar salida pin digital 5 a ALTO
avanzar
fijar salida pin digital 6 a ALTO
esperar hasta detectar boya
detectar boya
fijar salida pin digital 5 a ALTO
girar a la izquierda
fijar salida pin digital 6 a BAJO
esperar 0.5 segundos
fijar salida pin digital 5 a ALTO
avanzar por 2 segundos
fijar salida pin digital 6 a ALTO
esperar 2 segundos
fijar salida pin digital 5 a ALTO
girar a la izquierda
fijar salida pin digital 6 a BAJO
esperar 0.5 segundos
fijar salida pin digital 5 a ALTO
avanzar
fijar salida pin digital 6 a ALTO
esperar hasta detectar boya
detectar boya
fijar salida pin digital 5 a BAJO
girar a la derecha
fijar salida pin digital 6 a ALTO
esperar 0.5 segundos
fijar salida pin digital 5 a ALTO
avanzar
fijar salida pin digital 6 a ALTO
esperar 2 segundos
fijar salida pin digital 5 a BAJO
girar a la derecha
fijar salida pin digital 6 a ALTO
esperar 0.5 segundos
fijar salida pin digital 5 a ALTO
avanzar
fijar salida pin digital 6 a ALTO
esperar hasta detectar boya
detectar boya
fijar salida pin digital 5 a ALTO
girar a la izquierda
fijar salida pin digital 6 a BAJO
esperar 0.5 segundos
fijar salida pin digital 5 a ALTO
avanzar
fijar salida pin digital 6 a ALTO
esperar 2 segundos
fijar salida pin digital 5 a ALTO
girar a la izquierda
fijar salida pin digital 6 a BAJO
esperar 0.5 segundos
fijar salida pin digital 5 a ALTO
avanzar
fijar salida pin digital 6 a ALTO
esperar 10 segundos
fijar salida pin digital 5 a BAJO
frenar
fijar salida pin digital 6 a BAJO

Veremos que en la ventana de mBlock a la derecha se muestra el código escrito que corresponde a este programa:

```
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>

double angle_rad = PI/180.0;
double angle_deg = 180.0/PI;
void esperar();
double sensor1;
double sensor2;
double sensor3;
float getDistance(int trig,int echo){
    pinMode(trig,OUTPUT);
    digitalWrite(trig,LOW);
    delayMicroseconds(2);
    digitalWrite(trig,HIGH);
    delayMicroseconds(10);
    digitalWrite(trig,LOW);
    pinMode(echo, INPUT);
    return pulseIn(echo,HIGH,30000)/58.0;
}

void esperar()
{
    sensor1 = 100;
    sensor2 = 100;
    sensor3 = 100;
    while(!(((sensor1) < (40)) || (((sensor2) < (40)) || ((sensor3) <
(40)))))
    {
        _loop();
        if((getDistance(2,3)) > (0)){
            sensor1 = getDistance(2,3);
        }
        if((getDistance(8,9)) > (0)){
            sensor2 = getDistance(2,3);
        }
```

```
        if((getDistance(10,11)) > (0)){
            sensor3 = getDistance(2,3);
        }
    }
}

void setup(){
    pinMode(5,OUTPUT);
    pinMode(6,OUTPUT);
    digitalWrite(5,1);
    digitalWrite(6,1);
    esperar();
    digitalWrite(5,1);
    digitalWrite(6,0);
    _delay(0.5);
    digitalWrite(5,1);
    digitalWrite(6,1);
    _delay(2);
    digitalWrite(5,1);
    digitalWrite(6,0);
    _delay(0.5);
    digitalWrite(5,1);
    digitalWrite(6,1);
    esperar();
    digitalWrite(5,0);
    digitalWrite(6,1);
    _delay(0.5);
    digitalWrite(5,1);
    digitalWrite(6,1);
    _delay(2);
    digitalWrite(5,0);
    digitalWrite(6,1);
    _delay(0.5);
    digitalWrite(5,1);
    digitalWrite(6,1);
    esperar();
    digitalWrite(5,1);
    digitalWrite(6,0);
    _delay(0.5);
```

```
    digitalWrite(5,1);
    digitalWrite(6,1);
    _delay(2);
    digitalWrite(5,1);
    digitalWrite(6,0);
    _delay(0.5);
    digitalWrite(5,1);
    digitalWrite(6,1);
    _delay(10);
    digitalWrite(5,0);
    digitalWrite(6,0);
}

void loop(){
    _loop();
}

void _delay(float seconds){
    long endTime = millis() + seconds * 1000;
    while(millis() < endTime)_loop();
}

void _loop(){
}
```

Cierre

Una vez finalizado este proyecto, es posible extenderlo si se quiere continuar. Estas son algunas opciones sugeridas:

- ➢ Programar trayectorias triangulares de recorrido para optimizar el barrido total deseado.
- ➢ Programar diferentes trayectorias alternativas para esquivar un mismo obstáculo.
- ➢ Incluir en la electrónica planteada un doble puente H para poder invertir el sentido de las trayectorias en las que se mueve el dron acuático.

El proceso de resolución de problemas como los que se han planteado aquí permite la movilización y la integración de distintos saberes en la búsqueda de soluciones posibles a una situación dada. Si bien la información aquí fue presentada a modo de instructivo, se espera que sean los estudiantes organizados en pequeños grupos quienes vayan encontrando las mejores formas para construir los dispositivos. Esto implica preparar los materiales para que cada grupo cuente con todo lo necesario para la construcción del proyecto. Además, al interior de cada grupo, los estudiantes deben distribuirse los roles y las tareas de acuerdo a las demandas que van teniendo en las actividades.

Es importante que los docentes acompañen las producciones de cada grupo monitoreando los avances de todos los estudiantes y presentando la información que se considere necesaria para continuar la tarea. Pero, al mismo tiempo, es necesario que habiliten espacios para que los alumnos realicen hipótesis, planteen interrogantes, indaguen, prueben y realicen ajustes de acuerdo a lo que ellos mismo van pensando sobre cómo llevar a cabo el proyecto.

En este sentido, registrar lo que se va haciendo, las preguntas de los alumnos, las pruebas, los errores y cómo se fueron construyendo los dispositivos, permite reflexionar sobre la propia práctica, reforzar los aprendizajes construidos a lo largo de este proceso y poder volver a ese material disponible para próximos proyectos que se realicen.

Una vez terminado el proyecto, se sugiere reunir y organizar con el grupo el registro que se hizo del proceso realizado. Esta instancia de sistematización también permite movilizar capacidades vinculadas a la comunicación porque implica tomar decisiones respecto a cómo se quiere mostrar el proyecto a otros (otros grupos, otras escuelas, otros docentes, a la comunidad, etc.).

Glosario

Electrónica y arduino

Arduino: Placa electrónica que contiene un microcontrolador programable y sistema de comunicación (USB y serial) que permite al usuario cargarle diversos programas así como también comunicarse con la misma. Del lado de la computadora se utiliza un IDE (entorno de desarrollo integrado) para generar el código, compilarlo y quemarlo en la placa. Existen múltiples IDE compatibles con las placas Arduino.

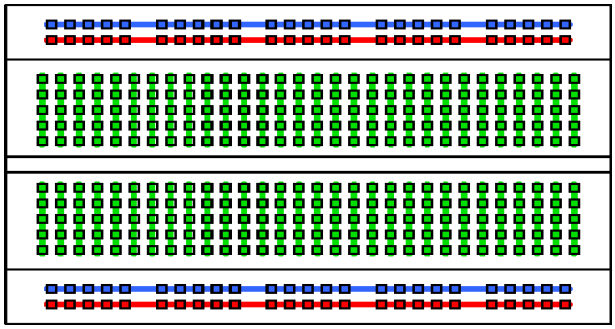
El microcontrolador posee entradas analógicas y digitales así como salidas digitales, PWM y servo. Las entradas y salidas digitales son las que permiten leer o escribir estados del tipo binarios. Pueden adoptar la forma de 0 ó 1, alto o bajo, verdadero o falso. Para prender y apagar los LED del semáforo utilizamos salidas digitales, las mismas están nomencladas con números desde el 0 al 13.

Las entradas analógicas permiten leer información que puede adoptar diferentes niveles de tensión, tal como la lectura de un termómetro analógico, la posición de un potenciómetro, etc. Las mismas están identificadas en la placa como A0 a A5.

Puerto COM: Es el puerto de comunicaciones a través del cual un sistema operativo informático se comunica con un dispositivo externo tal como una placa Arduino. La asignación de los mismos suele realizarse de forma automática al conectar la placa via USB. Dicha asignación suele ser dinámica, lo que significa que a veces cambia el número al conectar una misma placa en otro puerto USB o al conectar varias placas. En todos los IDE de programación es necesario especificar el puerto COM a través del cual nos comunicaremos con la placa Arduino.

Protoboard: Es una placa experimental que permite el prototipado rápido de circuitos electrónicos. Tiene orificios para insertar las patas de los componentes permitiendo que se conecten sin tener que recurrir a la soldadura.

El mismo posee una grilla de orificios que se encuentran conectados entre sí siguiendo el esquema de la imágen. Las líneas de conexión superior e inferior recorren la placa de punta a punta y suelen utilizarse para la alimentación del circuito, mientras que las líneas verdes se suelen utilizar para interconectar componentes. Tomar en cuenta que las líneas verdes se interrumpen en el centro de la placa. Generalmente se utilizan cables del tipo dupont para realizar conexiones en la protoboard.



Sensor de distancia ultrasónico: Es un tipo de sensor que se utiliza para medir distancias. El principio de funcionamiento del mismo se basa en emitir un pulso de sonido ultrasónico y medir el tiempo que pasa hasta registrar que dicho pulso regresó a la fuente tras rebotar en un

obstáculo. El tiempo transcurrido es directamente proporcional a la distancia que se encuentra el objeto, esto se debe a que la velocidad de propagación del sonido en el aire es lineal. Este tipo de sensor tiene cuatro pines de conexión, de estos se utilizan dos para la alimentación eléctrica del mismo (VCC y GND). Los dos pines restantes se utilizan para generar el pulso (Trigger) y para detectar la llegada del mismo (Echo).
La utilización del mismo requiere programar el mecanismo de disparo, medición de tiempo y sensado de la llegada del rebote. En general la mayoría de los entornos de programación facilitan alguna librería que resuelva de manera simple la gestión de estos procesos, pudiendo acceder como usuario directamente a la información en Centímetros de la medición realizada.

Motor DC: es el tipo de motor más popular entre los dispositivos que nos rodean. Estos motores pueden girar en ambos sentidos libremente. La velocidad de giro será directamente proporcional a la potencia entregada al mismo. Sirven para mover ruedas de vehículos, aspas de ventiladores, etc. Los mismos no permiten un control preciso de la posición del eje o cantidad de vueltas recorridas. Existen motores DC que traen integrada una caja de engranajes para ajustar el rango de velocidad y fuerza otorgados a nuestro sistema mecánico. Se conectan mediante dos pines (+ y -), controlando la señal que proveemos a los mismos podemos definir la velocidad y el sentido de giro del motor.
Debido a que los motores requieren mayor nivel de potencia del que una placa Arduino es capaz de manejar, siempre se va a requerir un circuito electrónico intermediario que se encargue de “amplificar” dicha señal para que el motor reciba la energía necesaria.

ULN2003: El ULN2003 es un circuito integrado que nos servirá para realizar la conversión de una señal de control de baja potencia (salida digital del Arduino) a un señal con la potencia necesaria para poder hacer se mueva el motor.
No es posible conectar motores DC directamente a las salidas de Arduino ya que los mismos requieren mayor nivel de potencia.

Impresión 3D

Formato .stl: El .stl es un formato de archivo que contiene la forma de un objeto sólido. Este formato de archivo no contiene información tal como color, texturas o propiedades físicas. Los archivos STL son objetos ya consolidados por lo que resulta útil para el intercambio e impresión de los mismos. Este formato de archivo no resulta práctico en caso de necesitar editar la geometría del objeto. Esto se debe a que no contiene información paramétrica sobre la generación de las diversas formas, curvas o capas que se utilizan a la hora de diseñar. Se lo puede considerar como la bajada o exportación final de un diseño, en ciertos aspectos equivalente a pensar en exportar un documento PDF partir de un documento de texto. Es posible generar archivos STL partiendo desde distintos tipos de entornos de diseño y modelado en 3D.

Código G (GCODE): Es el nombre que recibe el conjunto de acciones que va a tener que realizar la impresora 3D, o cualquier otro tipo de máquina CNC, para completar el trabajo solicitado. Estas instrucciones se generan partiendo del análisis de un archivo STL y realizando el cálculo de todos los movimientos y trayectorias que realizará cada componente de la impresora (motores, avance de filamento, calentador de extrusor, calentador de la base, etc) para materializar el objeto en cuestión. Debido a que cada marca y modelo de impresora 3D

tiene diferentes características mecánicas, el código G generado para imprimir cierto objeto solo va a servir para ejecutarse en un modelo de impresora específico.

Reconocimientos

Este trabajo es fruto del esfuerzo creativo de un enorme equipo de entusiastas y visionarios de la pedagogía de la innovación, la formación docente, la robótica, la programación, el diseño y la impresión 3D. Les agradecemos por el trabajo en equipo inspirador para traer a la realidad la obra que, en forma conjunta, realizamos INET y EDUCAR del Ministerio de Educación, Cultura, Ciencia y Tecnología de la Nación Argentina.

Contenidos
Equipo INET
Alejandro Anchava
Joreliz Andreyna Aguilera Barragán
Omar Leandro Bobrow
Alejandro Cesar Cáceres
Ezequiel Luberto
Gustavo Roberto Mesiti
Alejandro Palestrini
Judit Schneider
Pablo Trangone

Equipo Educar:
Pablo Aristide
Mayra Botta
Anabela Cathcarth
Eduardo Chiarella
María Laura Costilla
Diego Dorado
Facundo Dyszel
Federico Frydman
Matías Rinaldi
Uriel Rubilar
Camila Stecher
Carolina Sokolowicz
Nicolás Uccello

Para la confección de esta obra se contó con el el apoyo de la Universidad Pedagógica Nacional "UNIPE". En particular en el desarrollo de los capítulos 1 y 2, los cuales estuvieron a cargo de los profesores Fernando Raúl Alfredo Bordignon y Alejandro Adrián Iglesias.

Producción y comunicación
Juliana Zugasti

Diseño y edición
Leonardo Frino
Mario Marrazzo

Corrección de estilo
María Cecilia Alegre

Agradecimientos especiales
Mariano Consalvo. Equipo ABP
Damián Olive. Equipo de ABP
María José Licio Rinaldi, Directora Nacional de Asuntos Federales INET, quien siempre acompañó a este equipo en todas las gestiones para su implementación

Estamos comprometidos en instalar la innovación en la escuela secundaria técnica: la robótica, la programación, el pensamiento computacional, los proyectos tecnológicos, el ABP, la impresión 3D, de manera más accesible para todos.
Agradecemos enormemente, docente, tu continua dedicación y compromiso con el futuro de tus estudiantes.

¡Estamos ansiosos por saber qué es lo que vamos a crear juntos!