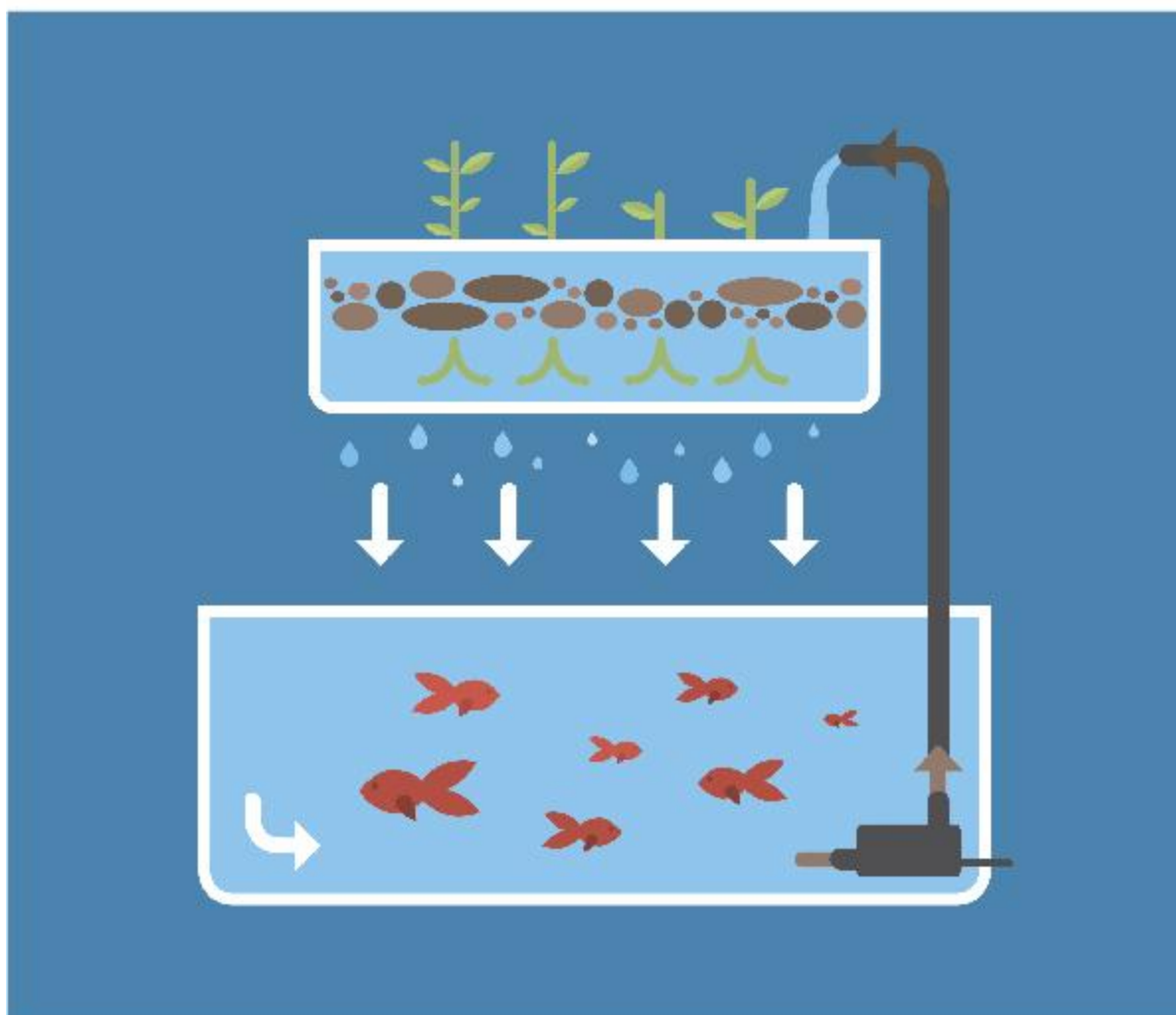


SABERES DIGITALES



SISTEMA ACUAPÓNICO AUTOMATIZADO

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

AUTORIDADES	¡Error! Marcador no definido.
SISTEMA ACUAPÓNICO AUTOMATIZADO	4
Ficha Técnica	4
Presentación	5
Desarrollo	7
Nivel Inicial	7
Paso 1 - Conectar el aireador al módulo relé	7
Paso 2 - Programar los tiempos de encendido del aireador	8
Paso 3 - Subir el código a la placa Arduino	12
Paso 4 - Incluir la bomba de agua	12
Paso 5 - Programar sin código bloqueante	13
Nivel Intermedio	17
Paso 1 - Imprimir las partes	17
Paso 2 - Armar el mecanismo	19
Paso 3 - Conectar el motor del dosificador a la placa Arduino	20
Nivel Avanzado	23
Paso 1 - Introducción a Internet de las Cosas (IoT)	23
Paso 2 - Crear un Panel de Control	24
Paso 3 - Conectar módulo Obloq	30
Paso 4 - Arduino IDE	31
Paso 5 - Programación IoT.	33
Cierre	38
Glosario	39
Reconocimientos	¡Error! Marcador no definido.

SISTEMA ACUAPÓNICO AUTOMATIZADO

Ficha Técnica

Nivel educativo	Secundario. Ciclo Básico.
Descripción Gral	Automatización de un sistema acuapónico.
Niveles de complejidad	<p>Nivel inicial: Automatizar un sistema acuapónico mediante una Placa Arduino. Se programará el tiempo en el que se activarán el aireador de la pecera y la bomba que lleva el agua a las plantas.</p> <p>Nivel intermedio: Agregar un dosificador hecho con impresión 3D, automatizado para alimentar a los peces.</p> <p>Nivel avanzado: A través de Internet de las cosas (IoT), monitorear a distancia el funcionamiento de la bomba de agua, del aireador y del dosificador de alimento.</p>
Insumos	<ul style="list-style-type: none">• 1 x Arduino UNO R3• 1 x Cable usb tipo B• 1 x Fuente de 9v 1 A (plug centro positivo, 5.5x2.1mm)• 2 x Modulo relé• 1 x Protoboard• 1 x Motor DC con reducción• 1 x Módulo motor (uln2003)• 1 x Oxigenador para peceras• 1 x Bomba de agua• 20 cables Dupont macho-hembra• 20 cables Dupont macho-hembra• 1 x rollo PLA
Equipamiento	<ul style="list-style-type: none">• Computadora
Otros requisitos	<ul style="list-style-type: none">• Conexión a internet• Descargar el programa "mblock3" http://www.mblock.cc/software-1/mblock/mblock3/• Descargar el programa "Arduino"

Presentación

Descripción ampliada del proyecto

En este proyecto, se propone realizar una automatización de un sistema de acuaponia, considerando que los aspectos biológicos del mismo se trabajarán en otras instancias. Para esto, en el sistema acuapónico montado previamente, es necesario instalar un aireador del agua del estanque de los peces, una bomba para que circule el agua y un dosificador de alimento para los peces. Los sistemas acuapónicos se basan en la combinación de la hidroponia (cultivo de plantas en agua) con la acuicultura (cría de organismos acuáticos, en este caso de peces) a través de la recirculación de agua y nutrientes de un sistema a otro.

El agua proveniente del estanque de los peces posee (luego de un proceso de “biofiltrado”) nutrientes que pueden ser aprovechados por las plantas. Los desechos generados por los peces son descompuestos en nitritos y posteriormente en nitratos por las bacterias de nitrificación presentes en el sistema. Este proceso es llamado “biofiltrado”. Esos desechos en altas concentraciones son tóxicos para los peces y son contaminantes si se vierten en el ambiente. Las plantas aprovechan los nitratos y los usan como nutrientes, reduciendo su concentración y haciendo posible la recirculación del agua al subsistema de acuicultura.

La automatización del dispositivo se hará partiendo de un sistema acuapónico estándar de referencia (ver figura 1). Este cuenta con un estanque para los peces y una batea superior para el cultivo de plantas. El estanque y la batea estarán vinculados por una manguera conectada a una bomba de agua ubicada dentro del estanque y por un colector que conduce el agua sobrante de la batea hacia el estanque.

La automatización se hará utilizando una placa Arduino y una serie de sensores y actuadores. Inicialmente, el sistema será programado para regular la activación de la bomba de agua y el aireador de la pecera durante un intervalo de tiempo determinado.

En el nivel de complejidad intermedio, se propone construir un dosificador de alimento para peces impreso en 3D y automatizar su funcionamiento. En el nivel avanzado, se propone incorporar Internet de la Cosas (IoT) para monitorear, de manera remota, el funcionamiento (apagado y encendido) de la bomba de recirculación del agua, del aireador y del dosificador de alimento.

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

- Aproximarse al conocimiento y el manejo de distintos componentes electrónicos mediante la automatización de un sistema acuapónico.
- Conocer los componentes de la interfaz de Arduino.

- Analizar y desarrollar la programación de estructura secuencial de un programa que controle el tiempo de encendido y apagado de los componentes eléctricos.
- Aproximarse a la impresión 3D (nivel intermedio).
- Utilizar IoT (Internet de las Cosas) para registrar y monitorear el funcionamiento del sistema acuapónico (nivel avanzado).

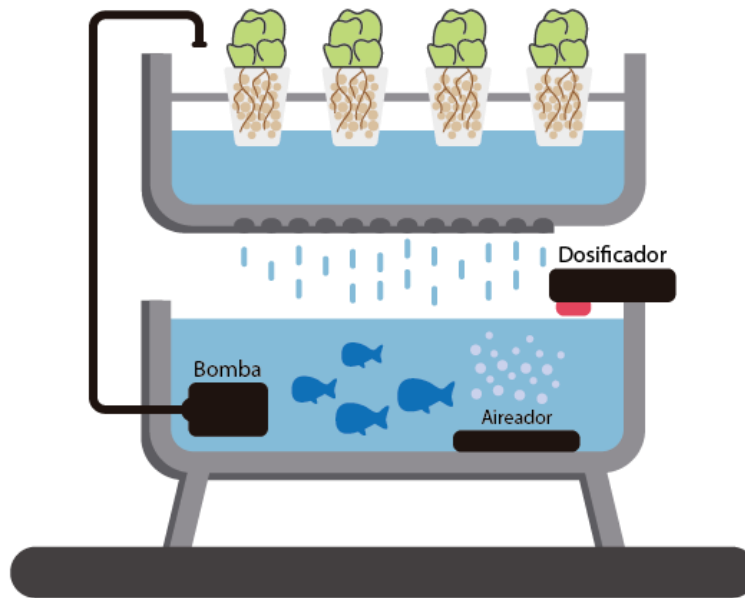


Figura 1. Sistema acuapónico estándar de referencia.

Desarrollo

Nivel Inicial

Los sistemas acuapónicos se basan en la combinación de la hidroponia (cultivo de plantas en agua) con la acuicultura (cría de organismos acuáticos, en este caso de peces). Uno de los elementos básicos de los sistemas acuapónicos es la recirculación de agua entre el compartimiento de los peces y el de las plantas, que permite aprovechar los nutrientes presentes en el agua de la pecera. Dado que este es un procedimiento que tiene que ser realizado a diario, es conveniente automatizarlo.

Al mismo tiempo, es necesario mantener un nivel determinado de oxígeno en la pecera, por lo que es conveniente incorporar un sistema aireador que funcione a intervalos de tiempo adecuados según las necesidades de los peces.

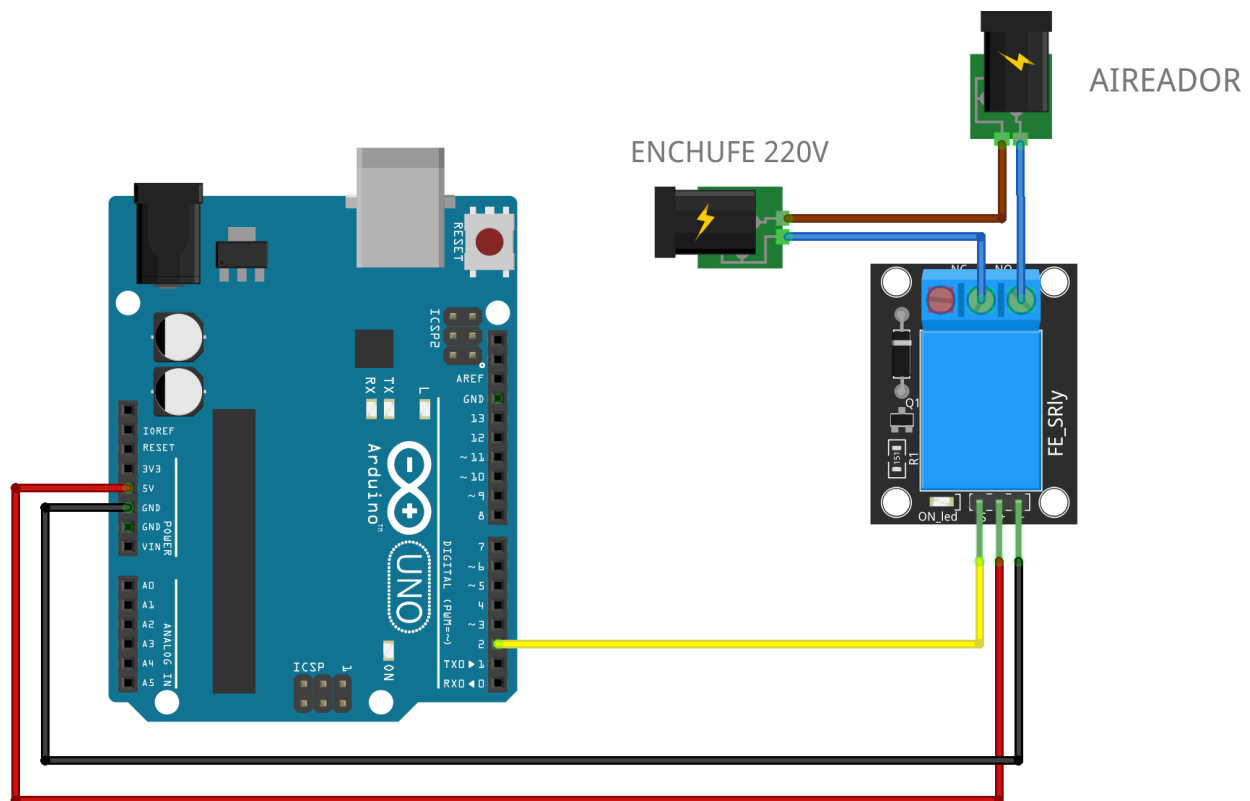
Un grupo de estudiantes de una escuela secundaria eligió construir un sistema acuapónico como proyecto para la materia Biología. Estudiaron los principios químicos y biológicos básicos de la acuaponia (como por ejemplo, los requerimientos de distintas especies de plantas y peces), diseñaron el sistema y consiguieron los materiales para su construcción. Sin embargo, tuvieron dificultades a la hora de automatizar el sistema. Por eso, se contactaron con distintas escuelas técnicas buscando ayuda para resolver este problema.

Instalar un sistema de recirculación del agua que funcione mediante una Placa Arduino. Se programará el tiempo en el que se activará la bomba que lleva el agua a las plantas y el aireador de la pecera.

Paso 1 - Conectar el aireador al módulo relé

Para que el agua de la pecera se mantenga aireada, encenderemos y apagaremos el aireador mediante un relé en un intervalo de tiempo determinado.

Para poder controlarlo electrónicamente, el aireador debe conectarse a un módulo relé y este a la placa Arduino, como se muestra en el siguiente esquema.



¡Atención! Para construir este dispositivo trabajaremos con un voltaje de 220V. En caso de utilizar protoboard, se recomienda no incluir en el mismo las conexiones a relé y 220V.

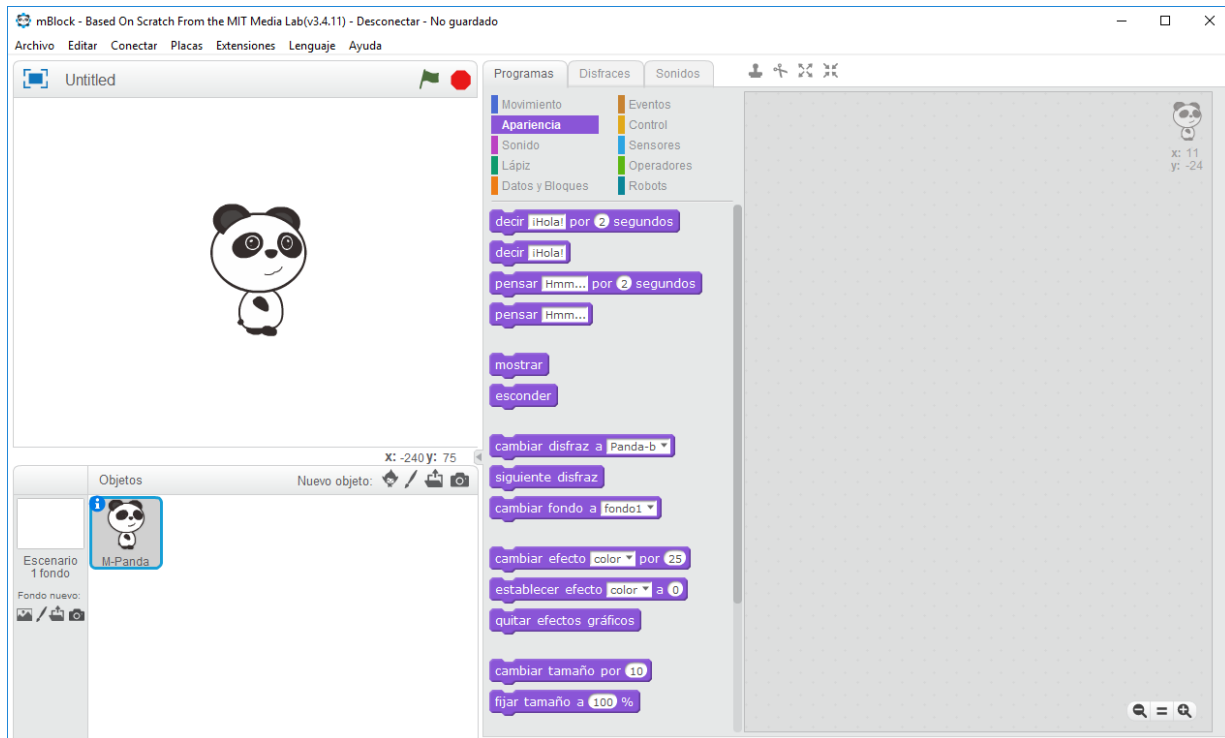
Como se puede ver en la imagen, utilizaremos un relé entre la toma de 220V y el aireador. Se conectan los dos terminales del aireador (pueden conectarse indistintamente a uno u otro ya que el aireador no posee polaridad). Donde se indica "ENCHUFE 220V", se está haciendo referencia a la conexión a la red eléctrica.

El código que subiremos a la placa Arduino es el que controlará la apertura y el cierre del relé y, en consecuencia, el encendido y el apagado del aireador.

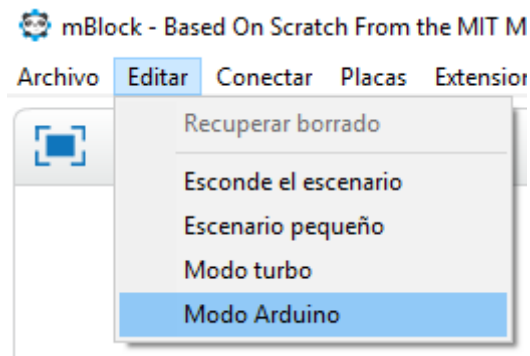
Paso 2 - Programar los tiempos de encendido del aireador

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/>

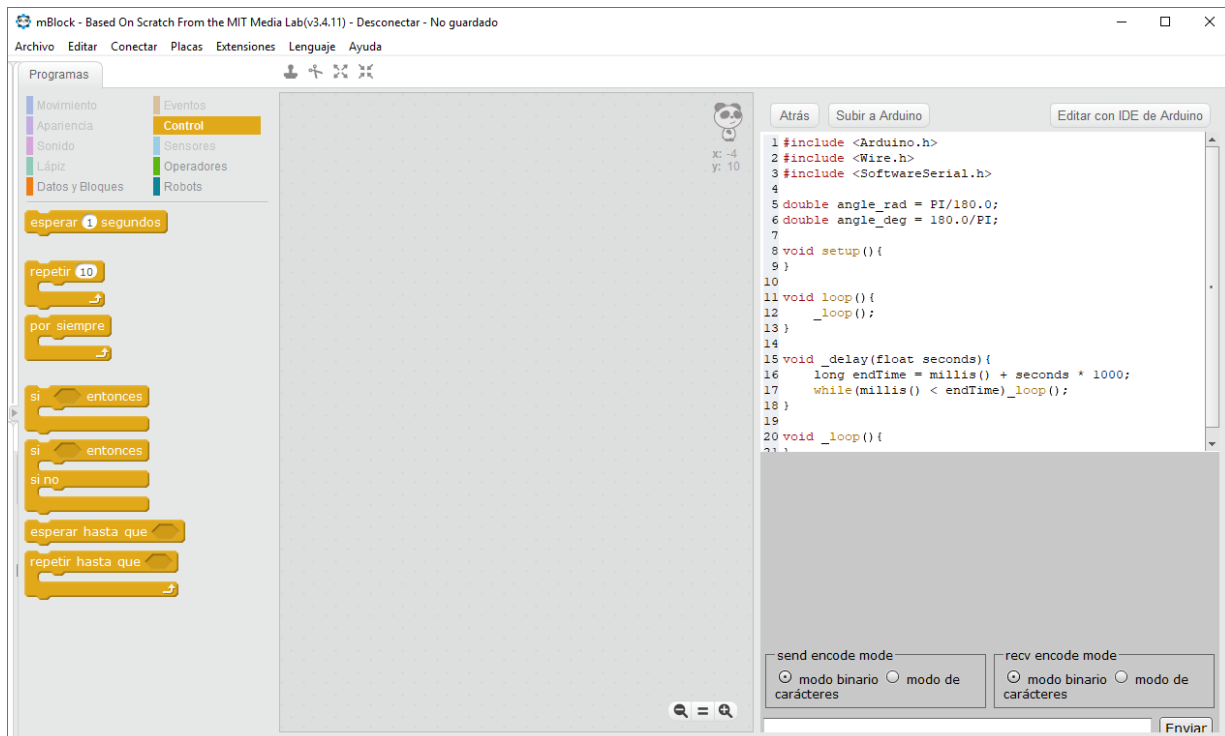
Cuando abrimos mBlock3, vemos una pantalla como la siguiente:



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



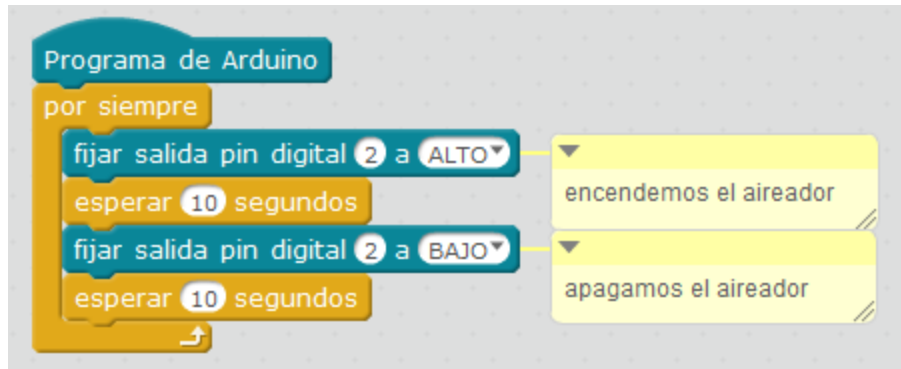
Al seleccionar este modo, el programa cambiará de aspecto. Se verá un área en el centro que es la que utilizaremos para programar con bloques. A la derecha se verá un campo donde aparecerá el código escrito que le corresponde a los bloques que están en el centro. Este código se irá escribiendo automáticamente 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 encienda el aireador por 10 segundos y que lo deje apagado por otros 10 segundos. Nuestro programa queda como sigue.



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;

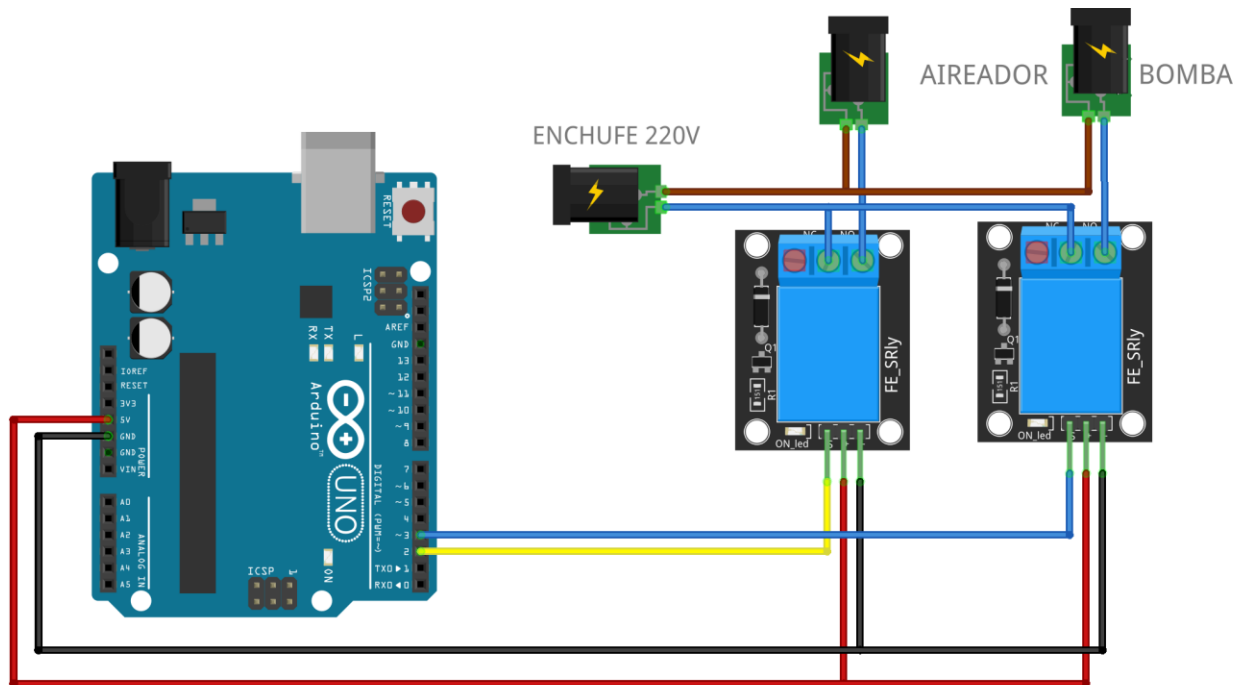
void setup(){
    pinMode(2,OUTPUT);
}

void loop(){
    digitalWrite(2,1);
    _delay(10);
    digitalWrite(2,0);
    _delay(10);
    _loop();
}

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

void _loop(){
}
```


Conectamos un segundo relé para la bomba, como indica el siguiente esquema.



Paso 5 - Programar sin código bloqueante

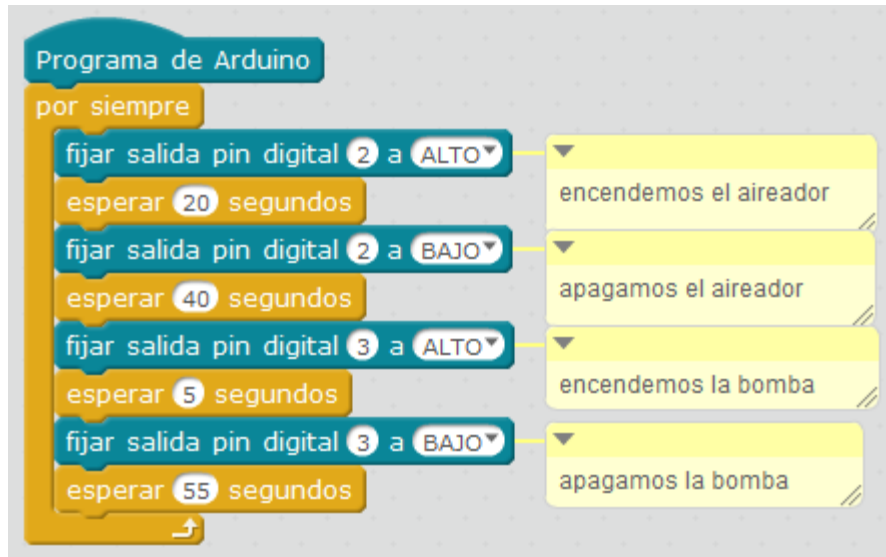
Ahora que tenemos que programar el tiempo de activación de dos relés, nos encontramos frente a un problema conocido como “código bloqueante”.

¿En qué consiste y por qué es un problema?

Vamos a suponer que queremos que nuestro programa mantenga los siguientes *regímenes de funcionamiento* para el aireador y la bomba:



- Aireador: 20 segundos encendido y 40 segundos apagado.
- Bomba: 5 segundos encendida y 55 segundos apagada.

Ambos elementos coinciden en un ciclo de encendido y apagado de 60 segundos. Si escribiéramos nuestro programa como se muestra a continuación, tendremos un problema.



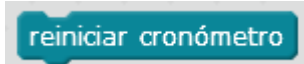
El problema es que, mientras se enciende y se apaga el aireador, la bomba permanece apagada (durante 20 + 40 segundos, según nuestro ejemplo); y mientras se enciende y se apaga la bomba, el aireador permanece apagado (durante 5 + 55 segundos, según nuestro ejemplo). Es decir, el código que indica el encendido y apagado del aireador “bloquea” el encendido y apagado de la bomba, y viceversa. Esto es lo que se conoce como “código bloqueante” y el problema es que impide realizar varias tareas en simultáneo.

¿Cómo se resuelve?

En lugar de utilizar el bloque , vamos a utilizar el bloque . A modo de ejemplo (como se ve en la imagen que está más adelante) establecemos lo siguiente:

- Si el tiempo que cuenta el cronómetro es menor a 20 segundos, se encenderá el aireador. Cuando se cumplan 20 segundos, se apagará.
- Si el tiempo que cuenta el cronómetro es menor a 5 segundos, se encenderá la bomba. Cuando se cumplan 5 segundos, se apagará.
-

Así, los *regímenes de funcionamiento* del aireador y de la bomba se ejecutan en simultáneo en función del tiempo que indica el cronómetro.

Por último, con el bloque , reiniciamos el cronómetro cada 60 segundos para que el conteo del tiempo vuelva a comenzar. De este modo, generamos código no bloqueante.

Nuestro nuevo programa queda 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;
double currentTime = 0;
double lastTime = 0;
double getLastTime(){
    return currentTime = millis()/1000.0 - lastTime;
}

void setup(){
    pinMode(2,OUTPUT);
    pinMode(3,OUTPUT);
}
```

```

void loop(){
    if((getLastTime()) < (20)){
        digitalWrite(2,1);
    }else{
        digitalWrite(2,0);
    }
    if((getLastTime()) < (5)){
        digitalWrite(3,1);
    }else{
        digitalWrite(3,0);
    }
    if((getLastTime()) > (60)){
        lastTime = millis()/1000.0;
    }
    _loop();
}

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

void _loop(){
}

```

Una vez subido a la placa Arduino, veremos que el aireador se enciende por 20 segundos y se apaga por 40 segundos. La bomba se enciende por 5 segundos y se apaga por 55 segundos. Como ya mencionamos, ambos elementos coinciden en un ciclo de 60 segundos que se va reiniciando.

Nivel Intermedio

Los alumnos notaron rápidamente que también era conveniente desarrollar un sistema automático para la alimentación de los peces. De esta manera, no tendrían que visitar el dispositivo los fines de semana ni correr el riesgo de olvidarse de proveer la comida para alguna de las ingestas de los peces. Entonces, diseñaron, construyeron y programaron un dosificador de alimento automático.

Se propone instalar, encima de la pecera, un dispositivo impreso en 3D que dosifique el alimento para los peces. Se utilizará para esto una Placa Arduino y se programará el tiempo de encendido del dosificador para que coincida con los momentos de ingesta necesarios.

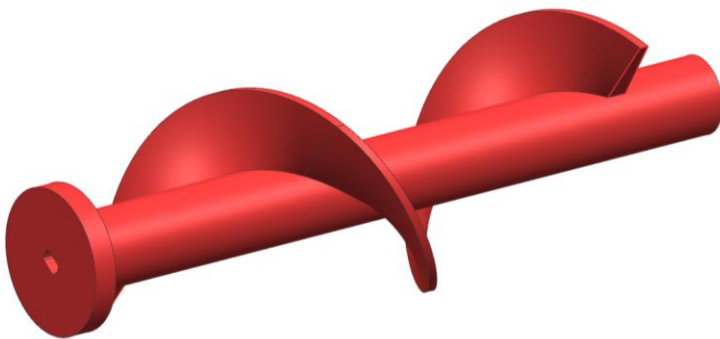
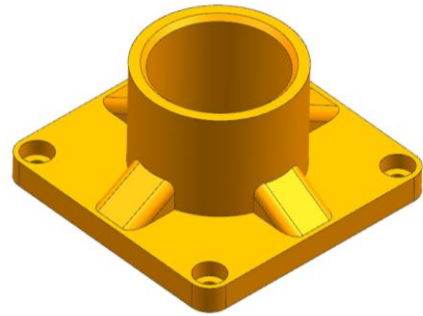
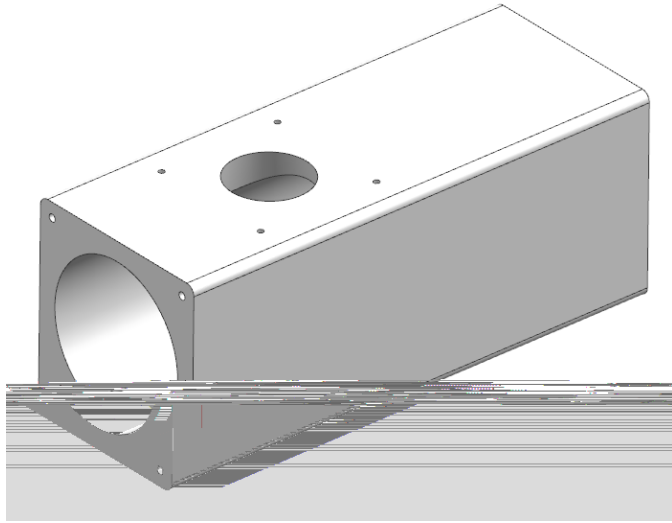
Paso 1 - Imprimir las partes

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 3D, excluyendo cierta información, como color, texturas o propiedades físicas, que se incluye en otros formatos CAD. Este es el formato que necesitamos abrir desde el software de impresión para poder desde allí exportar el modelo en código G.

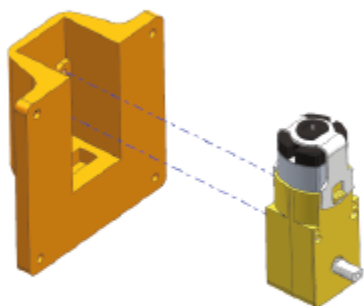
Código G es el nombre que habitualmente recibe el lenguaje de programación más usado en control numérico (CN). Es por medio de él que la impresora 3D logrará convertir el modelado en trayectorias para poder conformar la pieza que precisamos.

Las piezas son las siguientes:



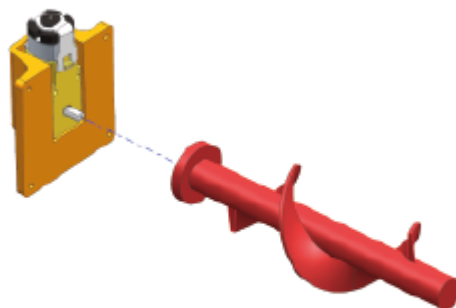
Paso 2 - Armar el mecanismo

1



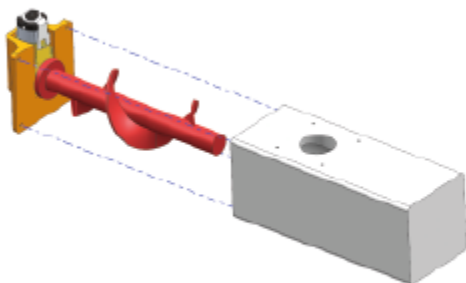
Fijar el motor amarillo al soporte motor.

2



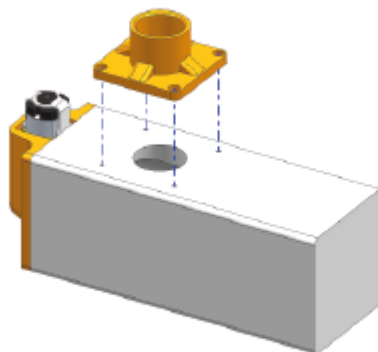
Vincular la barra tornillo al motor amarillo .

3



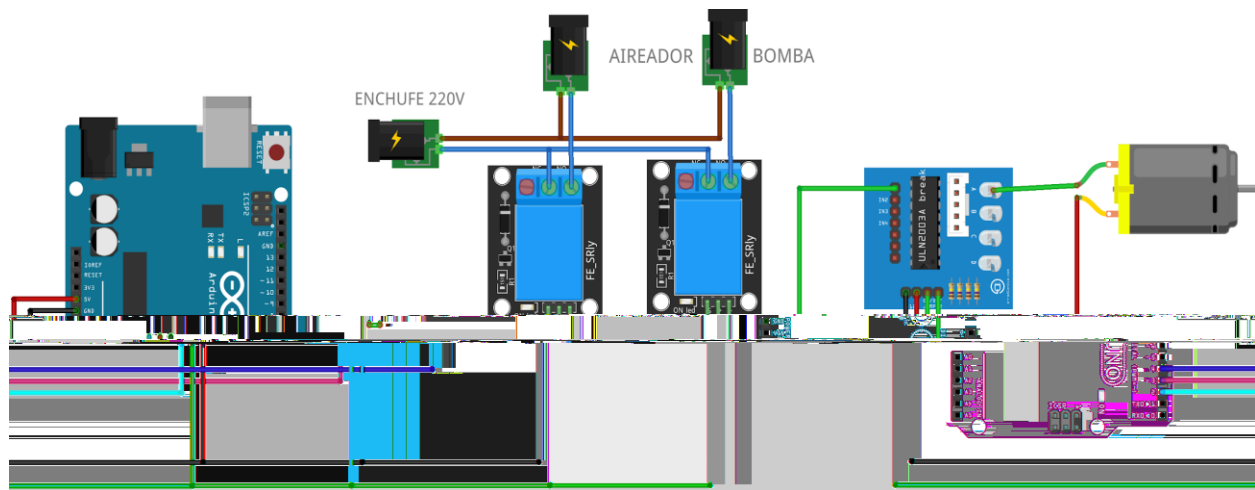
Ensamblar el conjunto soporte/motor amarillo/ barra tornillo con el cuerpo principal.

4



Vincular el soporte de la tolva (botella) con el cuerpo principal.

Paso 3 - Conectar el motor del dosificador a la placa Arduino



Paso 4 - Programar los tiempos del dosificador

Vamos a suponer que queremos que el dosificador se encienda durante 1 segundo por cada ciclo de 60 segundos y se mantenga lo que programamos en el nivel inicial. Nuestros *regímenes de funcionamiento* para el aireador, la bomba y el dosificador serían:

- Aireador: 20 segundos encendido y 40 segundos apagado.
- Bomba: 5 segundos encendida y 55 segundos apagada.
- Dosificador: 1 segundo encendido y 59 segundos apagado.

Nuestro programa queda como sigue.



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;
double currentTime = 0;
double lastTime = 0;
double getLastTime(){
    return currentTime = millis()/1000.0 - lastTime;
```

```
}

void setup(){
    pinMode(2,OUTPUT);
    pinMode(3,OUTPUT);
    pinMode(4,OUTPUT);
}

void loop(){
    if((getLastTime()) < (20)){
        digitalWrite(2,1);
    }else{
        digitalWrite(2,0);
    }
    if((getLastTime()) < (5)){
        digitalWrite(3,1);
    }else{
        digitalWrite(3,0);
    }
    if((getLastTime()) < (1)){
        digitalWrite(4,1);
    }else{
        digitalWrite(4,0);
    }
    if((getLastTime()) > (60)){
        lastTime = millis()/1000.0;
    }
    _loop();
}

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

void _loop(){
}
```

Nivel Avanzado

El proyecto de Biología ya está casi terminado. Los alumnos lograron automatizar el funcionamiento de la bomba y el aireador, así como la dosificación del alimento de los peces. Para finalizarlo, se les ocurrió que podía ser útil poder monitorear el funcionamiento de su sistema acuapónico a distancia, desde un dispositivo móvil. Así podrían asegurarse de que todo estuviera funcionando correctamente sin tener que visitar el dispositivo constantemente. Decidieron, entonces, agregarle una funcionalidad a su sistema para que comparta, utilizando internet, su estado en todo momento.

En esta actividad se programará el envío de datos obtenidos a un dispositivo móvil a través de Internet de las Cosas (IoT).

Paso 1 - Introducción a Internet de las Cosas (IoT)

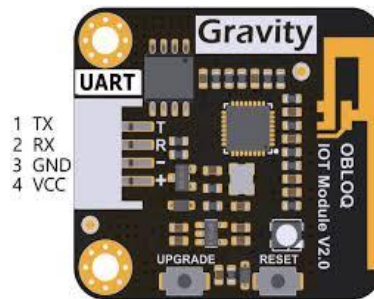
Internet de las Cosas (en inglés *Internet of Things*, abreviado IoT) es un concepto que refiere a la interconexión digital de objetos cotidianos con internet. Esta interconexión puede tener diversas funciones. Por ejemplo, puede utilizarse para monitorear la temperatura de un ambiente, enviando los datos obtenidos por un sensor a una central donde se recopile la información. De esta manera podría visualizarse en un dispositivo móvil la temperatura de un laboratorio, de un invernadero o de una sala de un hospital.

Para poder incorporar IoT a nuestro proyecto es necesario:

1. Un dispositivo capaz de conectarse a internet.
2. Un servidor que reciba y aloje los datos.

Existen diversas formas de registrar y almacenar los datos del sistema acuapónico construido. En este caso, se detallará cómo hacerlo con un módulo OBloq de DFRobot, y con los servidores de Adafruit.

El módulo UART OBLOQ es un dispositivo WiFi a serie pensado para desarrolladores no

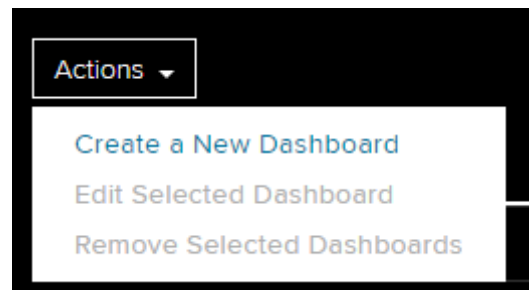


profesionales. Permite enviar y recibir datos mediante los protocolos HTTP y MQTT.

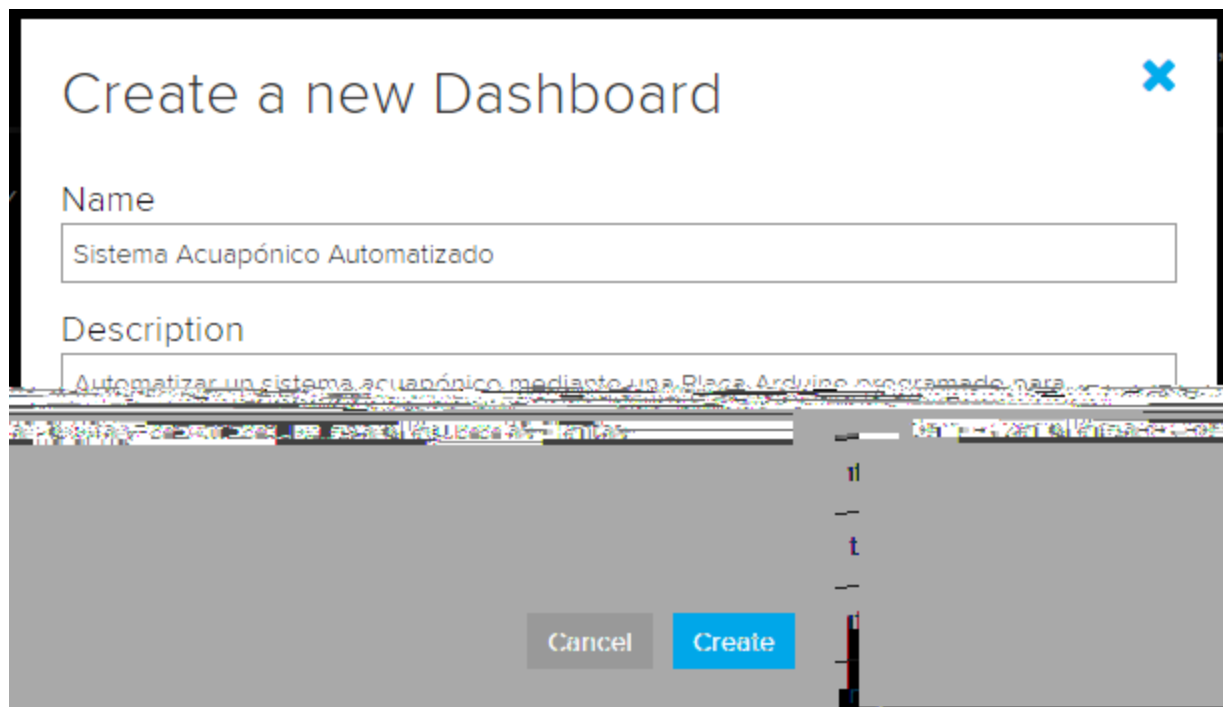
Paso 2 - Crear un Panel de Control

En primer lugar, se explicará cómo crear un Panel de Control en Adafruit. Luego, se verá cómo vincular los controles del Panel con los datos que se intercambian con el dispositivo.

Debemos crear una cuenta de usuario en io.adafruit.com. Una vez que ingresamos con nuestro usuario, creamos un nuevo panel haciendo click en “Create a New Dashboard”.



Creamos un nombre y una descripción.



Create a new Dashboard


Name

Sistema Acuapónico Automatizado

Description

Automatizar un sistema acuapónico mediante una Placa Arduino programado para...

Cancel Create

Hacemos click en el nuevo panel creado y veremos una pantalla vacía. Podemos comenzar a agregar bloques haciendo click en .

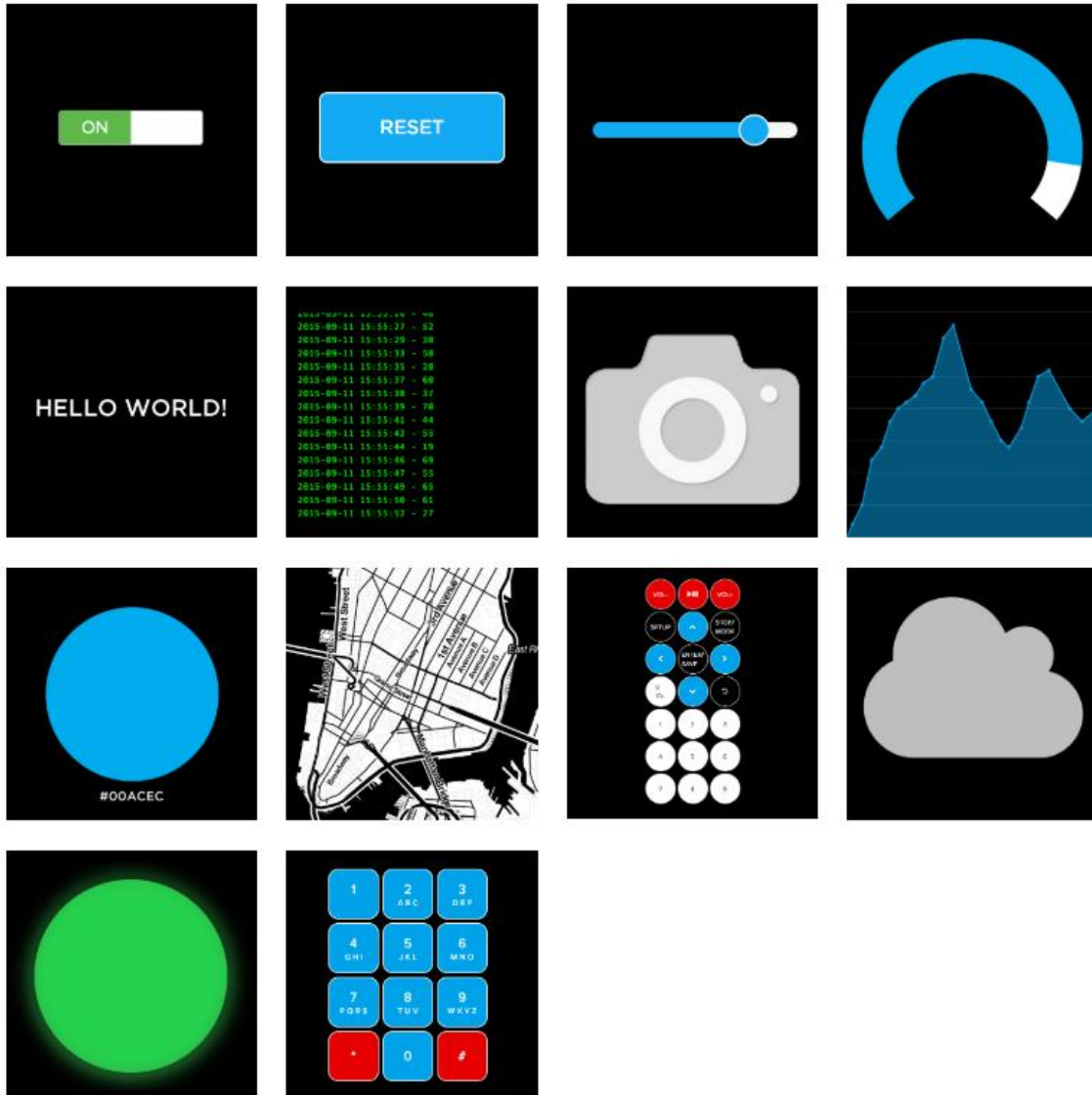


Veremos una serie de controles posibles como en la siguiente imagen:

Create a new block



Click on the block you would like to add to your dashboard. You can always come back and switch the block type later if you change your mind.



Para nuestro sistema acuapónico, podríamos ubicar tres *Indicator* (indicadores) para ver si el aireador, la bomba y el alimentador están encendidos o apagados.



Cuando agregamos un control al panel debemos asociarlo a un “*feed*”.

Un *feed* es una fuente de datos en la que uno puede publicar así como también se puede suscribir para recibir los datos de cierto feed.

En las líneas de código las palabras no pueden llevar tilde ni usar la letra Ñ.

Llamamos “aireador” al primer *feed*. En este *feed* publicaremos desde nuestro dispositivo si el aireador está encendido o apagado.

Choose feed ✕

Indicator: A simple on/off indicator lamp. Feed values are compared using the given conditions. If the conditions are true, then "On Color" is used, if false, "Off Color". All values are assumed to be numeric for comparison. If the current feed value can't be converted to a number, it will be treated as a string.

If you have lot of feeds, you may want to use the search field. You can also create a feed quickly below.

Create

Group / Feed	Last value	Recorded
<input checked="" type="checkbox"/> aireador		less than a minute

< Previous step Next step >

Luego de crearlo, hacemos click en “*Next step*” (paso siguiente) para configurar nuestro control y completamos los campos, como se ve en la imagen a continuación. Hacemos click en “Create block” (crear bloque) para completar la operación.

Block settings

In this final step, you can give your block a title and see a preview of how it will look. Customize the look and feel of your block with the remaining settings. When you are ready, click the "Create Block" button to send it to your dashboard.

Block Title (optional)

Aireador

On Color

#00ff00

Off Color

#ff0000

Conditions

=

1

Add Condition

Block Preview

Aireador

Indicator

A simple on/off indicator lamp. Feed values are compared using the given conditions. If the conditions are true, then "On Color" is used, if false, "Off Color". All values are assumed to be numeric for comparison. If the current feed value can't be converted to a number, it will be treated as a string.

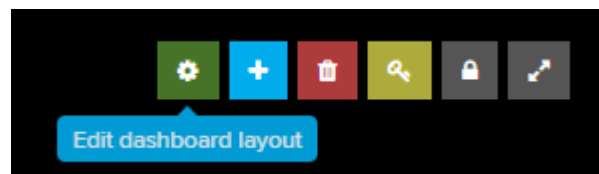
Test Value

1

Previous step

Create block

Podemos modificar el tamaño y la ubicación de los bloques haciendo click en la “rueda de configuración”.

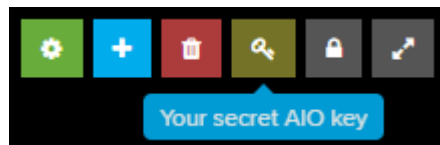


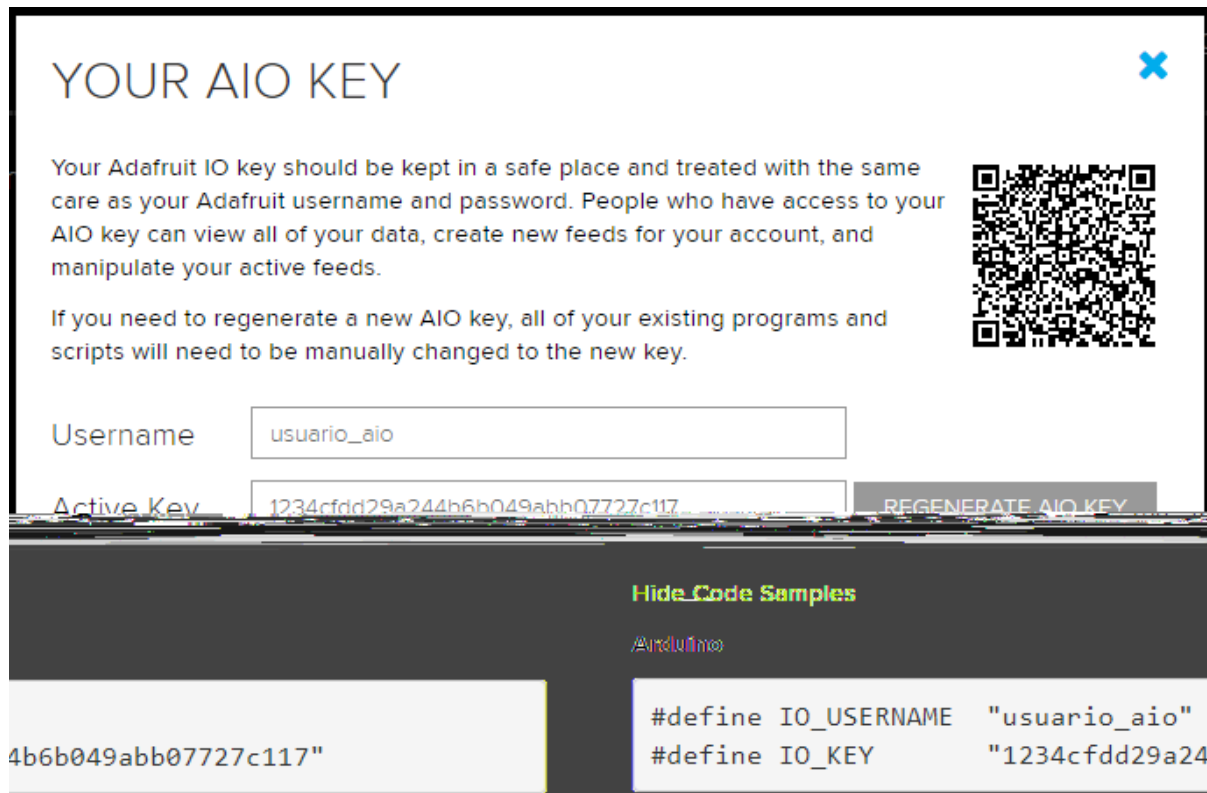
Deberíamos visualizar algo similar a lo siguiente.



Una vez realizado el Panel, publicaremos los estados del aireador, bomba y dosificador para poder monitorearlos de manera remota.

Antes de salir, debemos copiar las credenciales de acceso para poder publicar en nuestros *feeds* “aireador”, “bomba” y “dosificador”. Para ver nuestras credenciales, hacemos click en el ícono de la “llave”.





Copiamos el código que nos ofrece para Arduino, con nuestro usuario y key. Más adelante se verá que estos datos aparecen en el código de la siguiente manera:

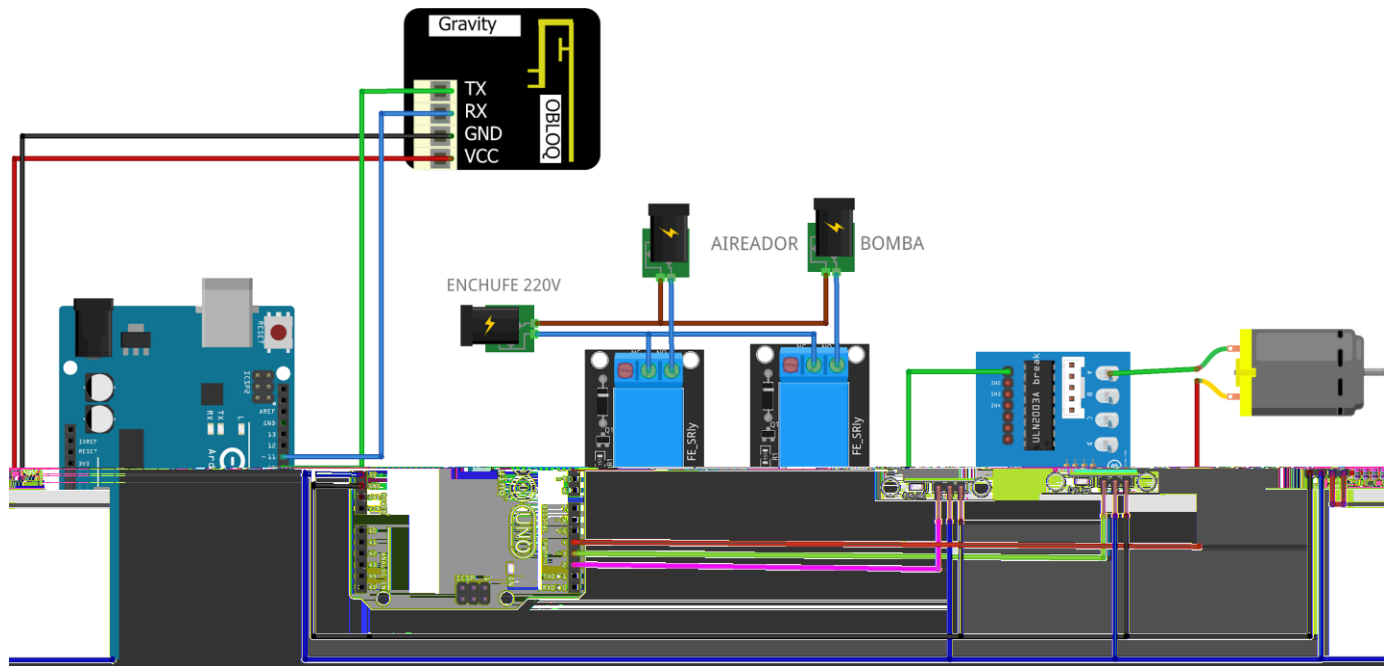
```
#define IO_USERNAME    "usuario_adafruit"
#define IO_KEY         "key_adafruit"
```

Se deberán reemplazar en esas dos líneas el usuario y key por los que se hayan obtenido en Adafruit. Por ejemplo:

```
#define IO_USERNAME    "usuario_aio"
#define IO_KEY         "1234cfdd29a244b6b049abb07727c117"
```

Paso 3 - Conectar módulo Obloq

A continuación se muestra el diagrama de conexión de Arduino UNO y OBLOQ.



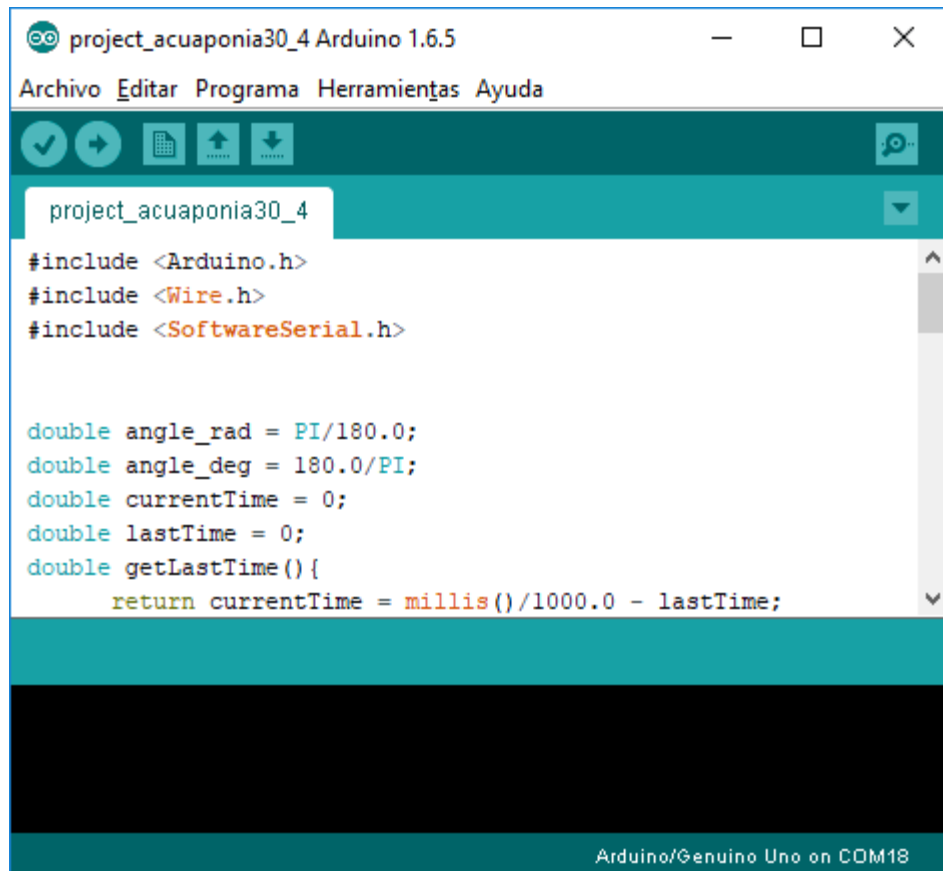
Paso 4 - Arduino IDE

La programación por bloques tiene sus ventajas desde un punto de vista didáctico, pero tiene sus límites. Cuando el programa que queremos realizar comienza a complejizarse podemos encontrarnos con el hecho de que ciertas operaciones no pueden resolverse utilizando bloques, o que hacerlo con este método resulta en algo mucho más engorroso y difícil de leer que si se utilizara el código escrito.

Hasta ahora hemos visto cómo al realizar nuestra programación en bloques, se generaba simultáneamente un código escrito en el área lateral derecha. Ahora ha llegado el momento de editar ese código escrito en otro entorno: el Arduino IDE.

Se puede editar el mismo código generado por el programa en bloques clickeando en el

botón . Veremos que se nos presenta la siguiente interfaz:



A continuación, se presenta una estructura mínima de un *sketch* (un programa) de Arduino:

```
void setup() {
    // Código de inicialización. Se ejecuta una sola vez.
}

void loop() {
    // Código principal. Se ejecuta repetidamente.
}
```

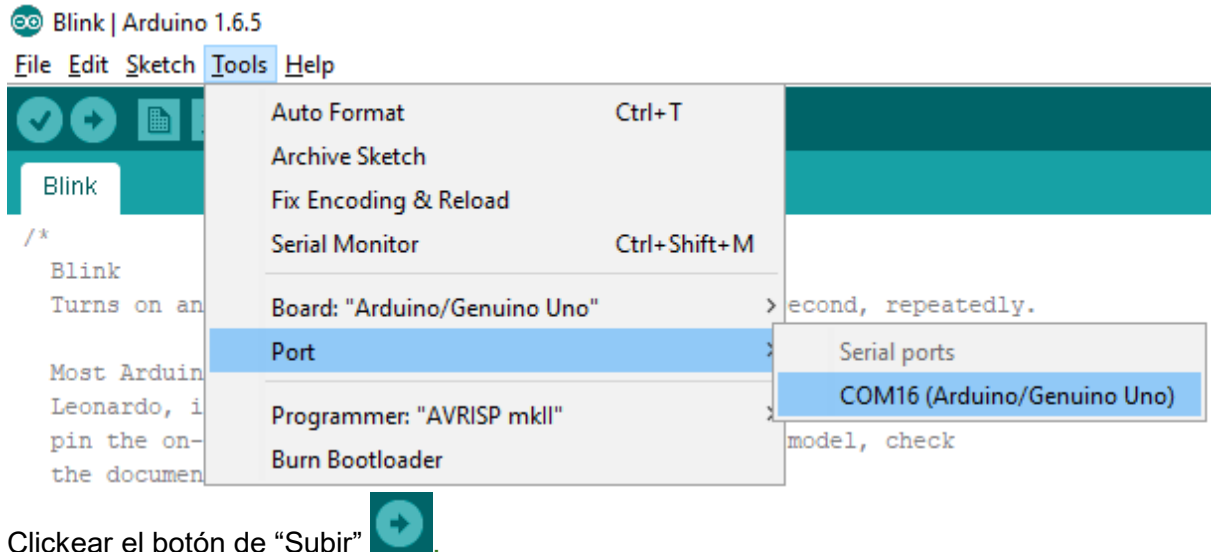
En líneas generales, un programa de Arduino es:

1. Un bloque de código que se ejecuta por única vez al inicializarse el dispositivo. Este bloque de código está contenido dentro de la función “setup” (se coloca dentro de `void setup() { y }`).
2. Un bloque de código que se ejecuta repetidamente luego de la función “setup”. Este bloque de código está contenido dentro de la función “loop” (se coloca dentro de `void loop() { y }`).

Después de `//` se incluyen comentarios para el lector que no tienen ningún efecto en el programa. Estos comentarios sirven para clarificar el código y que sea más fácil de interpretar para otras personas.

Los pasos para subir el código a través del Arduino IDE son similares a los que hemos visto para mBlock3:

1. Conectar la placa a la entrada USB.
2. Chequear que estén seleccionados la placa "Arduino/Genuino Uno" y el puerto serie al que está conectada la placa.



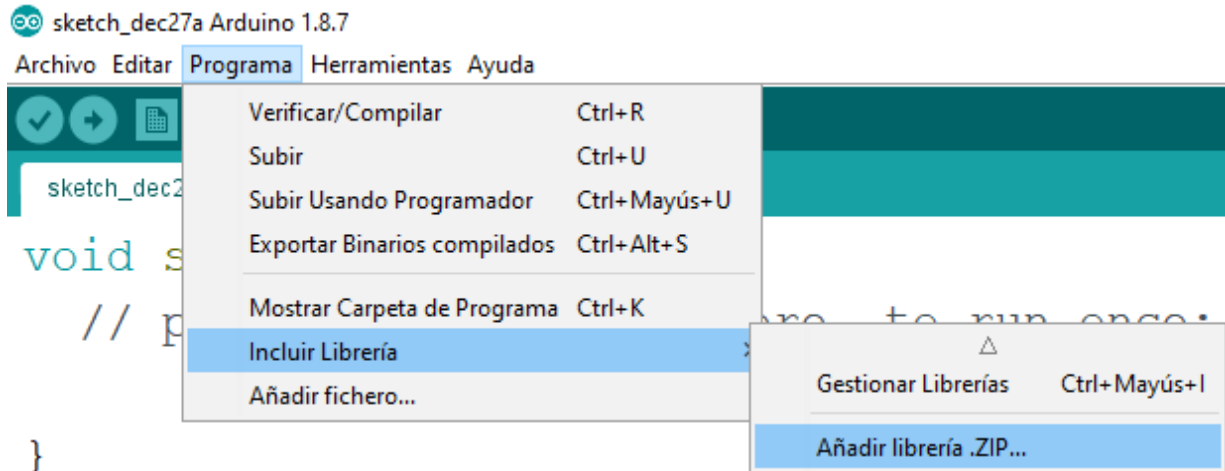
3. Clickear el botón de "Subir" .

Sabremos que nuestro código subió correctamente si en la barra de estado se escribe "Subido".

Paso 5 - Programación IoT.

Utilizaremos la librería ObloqAdafruit para informar a Adafruit el estado de las variables de nuestro sistema acuapónico. Podremos monitorearlo desde el Panel de Control que hemos creado.

En primer lugar debemos instalar la librería. Desde el menú clickeamos en "Añadir librería .ZIP" y seleccionamos el zip descargado desde <https://enfoco.net.ar/sd>.



En general, las librerías traen códigos de ejemplo como referencia. Podemos copiar la configuración del ejemplo “Publicar” de la librería.

Debemos reemplazar el SSID de la WiFi, su password, el IO_USERNAME e IO_KEY por los que copiamos de Adafruit.

```
#include "SoftwareSerial.h"
#include "ObloqAdafruit.h"

// Indicamos conexión de wifi.
#define WIFI_SSID      "SSID_de_Wifi"
#define WIFI_PASSWORD  "PWD_de_WIFI"

// Copiamos las credenciales obtenidas anteriormente en Adafruit.
#define IO_USERNAME    "usuario_adafruit"
#define IO_KEY         "key_adafruit"

SoftwareSerial softSerial(10,11);
ObloqAdafruit olq(&softSerial,WIFI_SSID,WIFI_PASSWORD,IO_USERNAME,IO_KEY);
```

El *setup* debe incluir la línea de inicialización del softwareSerial:

```
void setup()
{
    softSerial.begin(9600);
}
```

Se debe agregar también la función de actualización o “update”: `olq.update()`. Por esto es importante que nuestro código no sea bloqueante.

```
void loop()
{
    olq.update();
    // ..
    // ..
}
```

Para publicar un *feed*, utilizaremos la función `publish` del objeto `olq` :

```
olq.publish("aireador", 1); // Informar que el aireador está encendido.
```

```
olq.publish("bomba", 1); // Informar que la bomba está encendida.
```

```
olq.publish("dosificador", 1); // Informar que el dosificador está encendido.
```

Finalmente, nuestro programa de sistema acuapónico con IoT queda como sigue:

```
// Incluimos las librerías necesarias.
#include "SoftwareSerial.h"
#include "ObloqAdafruit.h"

// Indicamos conexión de wifi.
#define WIFI_SSID      "SSID_de_Wifi"
#define WIFI_PASSWORD  "PWD_de_WIFI"

// Copiamos las credenciales obtenidas anteriormente en Adafruit.
#define IO_USERNAME    "usuario_adafruit"
#define IO_KEY         "key_adafruit"

SoftwareSerial softSerial(10,11);
ObloqAdafruit olq(&softSerial,WIFI_SSID,WIFI_PASSWORD,IO_USERNAME,IO_KEY);

// Variables de estado del sistema.
// 0: apagado; 1:encendido.
int aireador = 0;
int bomba = 0;
int dosificador = 0;
```

```
double currentTime = 0;
double lastTime = 0;
double getLastTime(){
    return currentTime = millis()/1000.0 - lastTime;
}

void setup(){

    pinMode(2,OUTPUT);
    pinMode(3,OUTPUT);
    pinMode(4,OUTPUT);

    // Inicializamos la comunicación con el módulo OBloq.
    softSerial.begin(9600);

}

void loop(){

    if(getLastTime() < 20 && aireador == 0){
        digitalWrite(2,1);
        olq.publish("aireador", 1); // Publicar en adafruit.
        aireador = 1; // Cambiamos el estado.
    }
    if(getLastTime() > 20 && aireador == 1){
        digitalWrite(2,0);
        olq.publish("aireador", 0); // Publicar en adafruit.
        aireador = 0; // Cambiamos el estado.
    }

    if(getLastTime() < 5 && bomba == 0){
        digitalWrite(3,1);
        olq.publish("bomba", 1); // Publicar en adafruit.
        bomba = 1; // Cambiamos el estado.
    }
    if(getLastTime() > 5 && bomba == 1){
        digitalWrite(3,0);
        olq.publish("bomba", 0); // Publicar en adafruit.
        bomba = 0; // Cambiamos el estado.
    }
}
```

```
if(getLastTime() < 1 && dosificador == 0){
    digitalWrite(4,1);
    olq.publish("dosificador", 1); // Publicar en adafruit.
    dosificador = 1; // Cambiamos el estado.
}
if(getLastTime() > 1 && dosificador == 1){
    digitalWrite(4,0);
    olq.publish("dosificador", 0); // Publicar en adafruit.
    dosificador = 0; // Cambiamos el estado.
}

if((getLastTime()) > (60)){
    lastTime = millis()/1000.0;
}

// Llamamos a que la librería actualice lo que necesite.
olq.update();
}
```

Cierre

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

- Agregar distintos tipos de sensores para monitorear la calidad del agua y programar el envío de los datos obtenidos a un dispositivo móvil a través de Internet de las Cosas (IoT). Podríamos agregar sensores de la turbiedad del agua, de la temperatura del agua, del oxígeno disuelto, del pH, entre otros.

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 de programación para generar el código, compilarlo y quemarlo en la placa. Existen múltiples IDE compatibles con las placas Arduino.

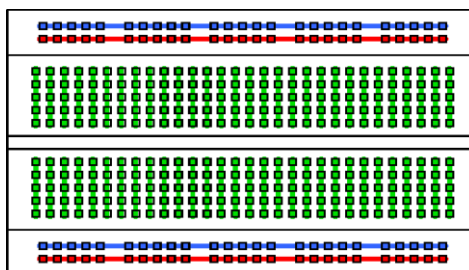
El microcontrolador posee entradas analógicas y digitales así como salidas digitales, PWM y servo. Las entradas y salidas digitales son las que permiten leer o escribir estados del tipo binarios. Pueden adoptar la forma de 0 ó 1, alto o bajo, verdadero o falso. Para prender y apagar los LED del semáforo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



Relé: Dispositivo electromagnético que funciona como un interruptor controlado por un circuito eléctrico. Por medio de un electroimán se acciona uno o varios contactos que permiten abrir o cerrar otros circuitos eléctricos independientes. Se utiliza comúnmente para aislar eléctricamente dos circuitos así como también permitir controlar con bajo voltaje a dispositivos que utilizan mayor voltaje. Arduino trabaja con señales de 5 V que mediante un relé permiten controlar la activación o desactivación de dispositivos de 220 V.

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. Este tipo de circuitos se lo llama “puente H” y pueden ser de distintos tipos de tecnologías, en esta guía utilizaremos el llamado ULN2003.

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 una 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: 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.

Internet de las cosas

Panel de Control Adafruit: Los sistemas IoT trabajan apoyándose en un servidor que se encarga de centralizar y gestionar la información que reportan los diversos sensores así como responder a las consultas de los dispositivos que buscan acceder a dicha información (mostrarla en pantalla, tomar decisiones, etc). Adafruit es una plataforma online con posibilidad de uso gratuito que ofrece el servicio de gestión de esta información. La misma ofrece un alto grado de compatibilidad con diversos estándares de trabajo IoT y se encuentra principalmente orientada al uso educativo.

Feed: fuente de datos en la que uno puede publicar y a la que puede suscribirse. Es decir, permite enviar datos, para que estos sean almacenados en el tiempo así como también leerlos, recibiendo las actualizaciones de quienes estén publicando allí. Es una forma de almacenar información en una gran base de datos de forma ordenada, utilizando el concepto de etiquetas tanto al momento de escribirla como el de leerla.

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 Andreyana 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 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 Bordinon 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!