JUNTA DE ANDALUCIA

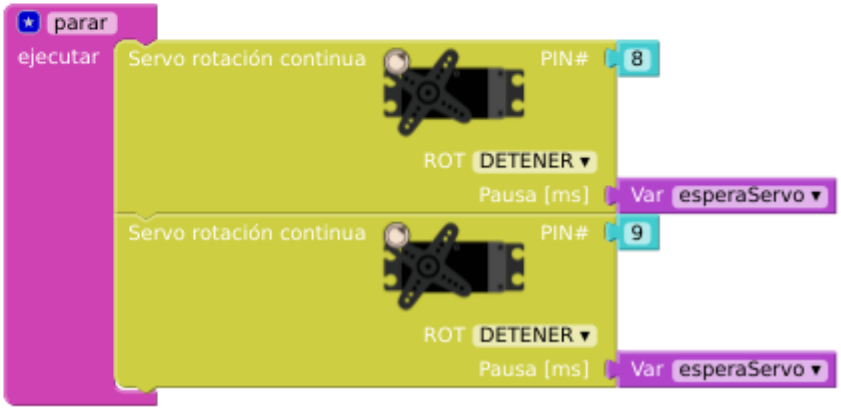
# CENTRO DEL PROFESORADO DE MÁLAGA

## Una experiencia personal de uso de ARDUINO con VISUALINO

Nombre: Aurelio Gallardo Rodríguez
Fecha: 4 – feb - 2017

Empezar con una demostración de VISUALINO. Robot PRINTBOT Evolution y Servo motores de rotación continua.





# ¿Por qué ARDUINO? ¿Por qué VISUALINO? Contexto.

- Sep 2015: último Proyecto Integrado de 4º ESO del Dpto. Tecnología del IES Seritium → Programación y Robótica.

- Situación:
  - Robots LEGO: problemas en los sensores. Problemas en alguna pantalla. Problemas de piecerío. Problemas de software (de siempre, con programas no libres).
  - Se "***arriesga***" en la compra de 5 robots PRINTBOT BQ EVOLUTION.

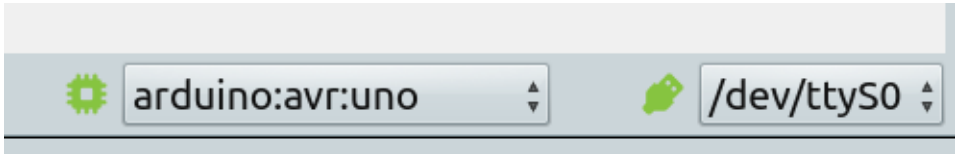
- Posteriormente:
  - A mitad de curso, y a petición del CEP de Jerez, se imparte curso sobre PRINTBOT EVOLUTION BQ.

- Ponencia presentada en el marco del curso del CEP DE JEREZ: ROBOTEXPERIMENTANDO EN EL AULA DE TECNOLOGÍA CON EL ROBOT PRINTBOT BASADO EN ARDUINO (161105ES070)

# Instalando VISUALINO

- La instalación de VISUALINO es sencilla. Entras en la página del programa, descargas el paquete correspondiente (ejemplo, Ubuntu 16.04 32 bits - visualino_0.7_i386.deb del 08-Feb-2016) y lo ejecutas.
- Importante: hay que tener el IDE de Arduino instalado y enlazarlo desde las preferencias de VISUALINO. Elegir, también, el idioma.


- Elegir puerto y modelo de placa (uno, bt, nano, mega, diecimila…) → mejora en la versión 7. Parte inferior derecha del programa.



# ¿Algo más con VISUALINO? Contexto.

- Participación en CTC de Granada. Proyecto para FANTEC 2016. 2 cursos de informática 4º ESO. ¿Qué hago? Implico aproximadamente a 15 alumnos muy interesados en electrónica recreativa y programación para llevar a cabo algunos proyectos. Algunos/as coinciden en la clase de informática y en la de proyecto integrado

  - Programamos en informática con VISUALINO y en proyecto integrado con BITBLOQ1. Ambos se basan en las bibliotecas javascript blockly
  - Durante aproximadamente dos – tres meses aprendemos lo básico del programa y de las posibilidades de la placa ARDUINO UNO.
  - Apuntes VISUALINO parte I
  - Trabajamos después los proyectos: torre de LEDs y POV (Persistance of vision).
    - Trabajos de los alumnos: torre1, torre2 y POV
  - Ampliación VISUALINO parte II

## Algunos proyectos interesantes con VISUALINO

- El experimento del Péndulo. Una iniciación al data logging
- Documento Péndulo
- Pequeño ejemplo de comunicaciones con Bluetooth

## Modificando VISUALINO

- Es posible modificar el código de VISUALINO, es un programa opensource, disponible en github
- Lo mantiene Victor R. Ruiz, de canarias: https://github.com/vrruiz

- Es posible añadir y modificar bloques.

# Añadir bloques a VISUALINO

- VISUALINO es un programa OPEN SOURCE basado en ROBOBLOCKS. Y ROBOBLOCKS es el repositorio de bloques de blockly que usa bq en bitbloq.

- Blockly, tal como dice en la introducción de su página: “La biblioteca Blockly añade un editor a tu aplicación que muestra elementos de código como bloques interconectables. Genera una salida sintácticamente correcta en el lenguaje de tu elección. Puedes crear bloques propios para conectar con tu propia aplicación”. ROBOBLOCK, por tanto, tiene bloques propios que genera una salida adaptada al IDE de ARDUINO.

# Procedimiento (en UBUNTU)…

- El objetivo final para añadir bloques a VISUALINO es conseguir un fichero **roboblocks.js** en el que se integra la información de los bloques que maneja el programa, con nuestras incorporaciones. Toda esta información se encuentra en README.md de ROBOBLOCKS.

# Procedimiento (en UBUNTU)…

- Conecto la carpeta al repositorio de github de roboblock con el comando git clone (entro en la carpeta de usuario y ejecuto como root **git clone http://github.com/bq/roboblock.git**) → NOTA: parece que ha cambiado a https://github.com/bq/roboblocks**.** *No hacerlo como root sino como un usuario del sistema.*
- Ejecutar **npm install** (como superusuario o no, da igual).
- Ejecutar **bower install** (quizás necesitéis hacerlo con la opción ***- - allow-root*** y siendo superusuario).
  - NOTA: la instalación de estos componentes me ha dado problemas cuando lo he vuelto a intentar al cabo del tiempo. Ambos paquetes han cambiado.
- Una vez instalado todo, en la carpeta **roboblocks**, la ejecución del comando ***grunt*** creará en el directorio de distribución **dist** una versión de ***roboblocks.js*** (y de roboblocks.min.js y la carpeta img con imágenes) que después podremos incorporar a directorio de VISUALINO (normalmente en /usr/share/visualino/html).
- Tal como dice la documentación, para crear un nuevo bloque, hay que escribir una serie de ficheros en el directorio fuente de la aplicación ***/src/blocks***

```
src
├── blocks                      // blocks folder
│   └── servo_move              // block name
│       ├── img                 // block image
│       │   └── blocks
│       │       └── *.png
│       ├── servo_move.c.tpl    // c code template
│       ├── servo_move.js       // block definition & code generation
│       └── README.md           // block documentation
├── profiles.js                 // supported profiles
└── utils.js                    // some utils and Blockly extensions
```

# Creando un bloque…

- Lo probé creando un bloque que aceptaba un valor numérico para escribir en un pin, ya que sólo lo hacía en función de los valores lógicos ALTO o BAJO, no en función de una variable numérica 0 ó 1.
  - Escojo el nombre: **inout_digital_write_var** (función de entrada salida, digital, escribir, con variable).
  - Crear la carpeta con ese nombre en el directorio **roboblocks/src/blocks**
  - A partir de aquí hay que programar. Es buena idea, quizás, ir fijándose en el código de programación de otros bloques. Lo primero que podemos hacer, para aclarar las cosas, es construir un README.md que me defina bien las entradas de dicho bloque (en este caso el valor del **PIN** y el valor a escribir **Value**)

README.md

```
inout_digital_write_var
=======================

Writes the value 0/1 - Variable to the specified PIN.

Parameters
----------

| Param name | Description | Type      |
 ------------|-------------|----------
| PIN        | Pin to which we are writing | `Number` |
| Value      | Int between 0 - 1 | `Number` |
```

# Creando un bloque…

- El siguiente paso es crear el código de programación en javascript: ***[nombre].js***
- No hay documentación precisa, salvo el **README.md** del proyecto ROBOBLOCKS. Lo mejor es analizar otros bloques hechos encontrados en la carpeta /src/blocks.
- Básicamente hay que hacer:
  - Definir una función (Blockly.arduino.[nombre]) con las entradas que sea y que retorna una variable ***code***. Esta variable es alfanumérica (de texto), por lo que lo que lo que se le añada debe ser del mismo tipo. A ***code*** se le van añadiendo objetos que posteriormente irá generando el código del IDE de ARDUINO en sus distintas partes: definición, setup y loop.
  - Definir cómo será el diseño del  bloque (Blockly.Blocks.[nombre]). Se van añadiendo los valores, los textos, si está en línea, si es un bloque que tiene un previo y un siguiente...
  - Para saber más hay que leer atentamente, y adaptar, lo que se escribe en https://developers.google.com/blockly/
  - El fichero de ejemplo: INOUT_VAR

## Creando un bloque…

- Y el último paso es crear los ficheros de “plantilla de código” (code template). Se llaman desde el fichero js del paso anterior e indican cómo es el código ARDUINO a implementar. Siguen una nomenclatura proporcionada por la biblioteca ***underscore.js***
- En mi ejemplo necesito dos ficheros de éstos. Uno para lo que inserto en el código ***setup*** y otro en el ***loop***.

| **SETUP: inout_digital_write_var_setups.c.tpl** | **LOOP: inout_digital_write_var.c.tpl** |
|---|---|
| // {{comentar}};<br>pinMode({{dropdown_pin}},OUTPUT); | digitalWrite({{dropdown_pin}},<br>{{value_num}}); |

## Terminando…

- Para terminar, debo, en la carpeta del proyecto ROBOBLOCK ejecutar la orden ***grunt***.
- Se ha generado en ***dist*** un fichero ***roboblock.js*** (y opcionalmente una imagen en dist/img/blocks). Este código se copia a la carpeta de VISUALINO (/usr/share/visualino/html) y ¡¡ya puedo ejecutar VISUALINO con el nuevo bloque!!

## Instalando desde GITHUB

- Puedo bajar con **git clone** el repositorio donde está visualino a una carpeta y compilarlo. Para eso necesito instalar Qt5 (con qtcreator y las librerías libqt5serialport5). Quizás se necesita desinstalar Qt4. Pero el proceso está en estudio, por un cambio en la API.
- Creo que funcionaría si instalo el último paquete debian de Visualino, y de la carpeta creada con git clone extraigo **roboblocks.js** y la carpeta **/img/blocks** copiándola a **/usr/share/visualino/html**

## ¿Por dónde va el proyecto?

- Teóricamente, si creáis un bloque válido podréis subirlo al proyecto VISUALINO en la página github del mismo: https://github.com/vrruiz ; escribid a Victor R. Ruiz antes , a través de su foro: http://www.visualino.net/forum/
- Parece que hay modificaciones hechas (entre ellas una adaptación del ejemplo anterior) en el master del github; sin embargo no se genera un paquete debian nuevo desde la versión 0.7 para ubuntu (08/02/2016) y el ejecutable de Windows versión 0.7.1 (13/05/2016), aunque hay una beta para windows del 25-Nov-2016 (vs. 0.7.2).