Màster universitari en **Formació del Professorat d'Educació Secundària Obligatòria i Batxillerat, Formació Professional i Ensenyament d'Idiomes**

Trabajo Final de Máster

Título: Retos PBL

Apellidos: *Corbera Moreno*
Nombre: *Bernat*
Titulación: *Màster en Formació del Professorat d'Educació Secundària Obligatòria i Batxillerat, Formació Professional i Ensenyament d'Idiomes.*
Especialitat: *Formació Professional.*

Director/a: *Francesc Farre Calpe*
Data de Lectura: *19 de Junio*

En el presente documento, los aspectos y puntos importantes serán definidos con las siguientes etiquetas:

Los puntos definidos en este apartado pueden causar daños en el equipo, el arduino, en el diseño o parte de este…

Los puntos definidos en este apartado pueden ser causa de riesgo por descarga eléctrica para el alumno.
El procedimiento para evitar todo daño personal será el de desconectar la alimentación para manipular esta parte del sistema.

Cuestionario o pregunta a resolver por alumno…

Las observaciones definidas en este apartado son importantes……

ÍNDICE MEMORIA

Contenido

Introducción 5
Marco teórico 7
Partes del diseño. 11
Diseño 12
Alimentación y electrónica 12
Programación Arduino 28
Diagramas de flujos o pseudocódigo 28
Código fuente de la aplicación comentado. 31
Construcción maqueta 46
Programación didáctica y temporización 49
Proceso de trabajo, metodología, evaluación y temporización. 53
Prototipo y lecciones aprendidas 64
Conclusiones 64
ANEXO1: Materiales 66
ANEXO2: Ficha del Alumno 78
Información para el alumnado 80
ANEXO3: Rúbricas de Evaluación según Criterios de Evaluación (CA) 89
ANEXO4: Manual de Programación Arduino 92
Usos de este Manual 92
1. Objetivos 92
2. ¿A quién va dirigido? 92
3. La Comisión 'Arduino'. 92
CRITERIOS BÁSICOS DE PROGRAMACIÓN 94
Instalación entorno de desarrollo 'ARDUINO' 94
Conexión de un 'Arduino' 96
Explorar el entorno de desarrollo 'Arduino' 98
Declaración y tipos de variables en 'Arduino' 102
Expresión lógica en 'Arduino'. Estructuras de control 106
- Condición If/Else 106
- Condición Switch case 107
- Bucles While 107

- Bucles do/While 108
- Bucles For 108

Máquina de estados 109

Lógica aplicada 110

Funciones 120

1. Funciones creadas por el usuario 120
2. Funciones proporcionadas por 'Arduino' 122
3. Librerías proporcionadas por 'Arduino' 122

Aplicaciones Básicas de programación con 'Arduino' 124

Definición Hardware 'Arduino'. Pins de conexión 124

1. Módulo PcB programable 'Arduino' UNO. A000066 124

Módulos de entradas/salidas digitales 125

Módulos de entradas/salidas analógicas 130

Control de los pins PWM 133

1. Programar Leds IR infrarrojos a 33 Khz 133
2. Programar el control de motores DC por Módulos de control PWM 133

Librería TIMER 133

1. Función Delay() 133
2. Función millis() 133

Librería SD. 134

Leer y grabar tarjeta de Memoria SD 134

Pantalla LCD. **Modelo A000096** 134

1. Escribir texto en la pantalla TFT 134
2. Importar archivo.bmp desde tarjeta SD 134

Librería Speaker 134

1. Enviar melodías al speaker 134

Comunicaciones 136

Módulos comunicación Serial 136

Módulos de comunicación $I2C$ 136

ANEXO5: Datasheets de dispositivos electrónicos utilizados 137

ANEXO6: Código fuente de la aplicación 137

ANEXO7: Documentación complementaria para los docentes y bibliografía 138

Introducción

El método de aprendizaje basado en resolución de problemas o Retos es-probablemente- el método que más se asemeja al aprendizaje laboral. Es una forma innovadora de estimular al auto aprendizaje y al autoconocimiento al alumno, presionarlo a experimentar, y cumplir plazos de entrega…
La finalidad de este trabajo es diseñar una actividad de innovación pedagógica relacionando las competencias del módulo de electrónica del CFGM: *Instal·lacions elèctriques i automàtiques*.
Se plantea crear una actividad para el alumnado de este CFGM, integrando las dos UFs de electrónica analógica y digital que se aplican al propio módulo y que culminaría con el diseño, montaje y programación de la maqueta a escala de un grupo de ventiladores y un sistema automatizado de control para estos y las luces de la sala de actos del centro educativo. Los ventiladores son más eco sostenibles que los equipos de climatización estándar. La finalidad es que el alumnado consiga desarrollar el sistema de control, mediante 'Arduino', y elabore una maqueta para presentar su idea al resto de grupos del curso. A través de esta actividad se integran competencias trabajando a partir de la idea de la concienciación ambiental. Por este motivo, se propone al alumnado trabajar un producto industrial y tecnológico bajo una responsabilidad medioambiental; trabajar sus competencias digitales (programación) y sus habilidades profesionales y sociales (práctica en equipo, gestión de la innovación empresarial y conocimiento del entorno).
A su vez, con el proyecto se da libertad creativa al alumnado por lo que corresponde al diseño y la justificación de la elección de materiales y funcionalidades.
Por todos estos motivos, se ha decidido organizar el Módulo mediante la metodología de Resolución de Proyectos.

Se ha planteado este TFM con un formato claro de Manual de Arduino, porqué:

1. Arduino es una herramienta adecuada para desarrollar competencias en la electrónica, si bien es relativamente económica y eficiente.
2. Hace dos meses no sabía programar en Arduino.

El autor ha tardado dos meses en aprender a programar Arduino, desarrollar el prototipo modelo del TFM y redactar este documento.
Tenemos la teoría, y para ello se ha planteado como manual, de que cualquiera que siga atentamente este documento, podría aprender a programar en Arduino, y desarrollar un PBL en tan solo dos meses.

3. **Nuestro objetivo final es poner al alcance de cualquier profesor y de todos los alumnos la programación de micro controladores, y hacerlo de una forma sencilla y accesible.**

EL TEMA

Nuestro tema es trabajar un proyecto de especialidad e instalaciones eléctricas con el uso de electrónica aplicada, para el alumnado de CFGM en el contexto de un aula-taller.
Como se ha comentado con anterioridad, la idea es que el proyecto sea una actividad vinculada a la zona geográfica donde se encuentra el centro. De esta forma es un pequeño proyecto real que tiene un encaje dentro de la comunidad educativa y el entorno.

ORGANIZACIÓN Y ASIGNACIÓN DE TAREAS

Para la coordinación y correcta elaboración del trabajo hemos distinguido las tareas y responsabilidades de cada miembro del grupo.
De manera organizativa hemos decidido que uno de nosotros era el *team leader* sobre alguna área dónde había asignadas tareas. De esta forma habrá un control y seguimiento sobre cada objetivo del grupo. La planificación será flexible y se irán adaptando a las necesidades que vayan surgiendo a medida que avance el proyecto.
Cada semana se harán reuniones fuera del horario de clase con todos los miembros del grupo, así como trabajo individual. La coordinación será virtual a través del correo electrónico y las plataformas de whatsapp y Skype. En las reuniones servirán para revisar el retro timing, la calendarización y para tratar de prever posibles nuevas tareas o corregir imprevistos; como el hecho de comprar algún componente electrónico necesario antes de las sesiones de taller.
Las tareas se han asignado teniendo en cuenta la formación individual de cada miembro del grupo, así como su experiencia y habilidades. Cada alumno dedicará 10-12 horas semanales de trabajo personal para desarrollar las tareas, durante 11 semanas. Los hitos detallados de este trabajo van en acorde al tiempo dedicado concretado en la planificación del proyecto.

La composición de los grupos

Se forman grupos de 6 personas, siendo dos miembros del grupo de cada CF previamente mencionado. El valor de este proyecto es juntar en un mismo equipo de trabajo alumnado de disciplinas distintas, para que intercambien información, superen el reto y colaboren con los objetivos que tienen. Cuanto más diferente es el grupo, desde un punto de vista social, profesional e intelectual se conseguirá un mayor número de competencias relacionadas con el aprendizaje.
Cada profesor de cada CF valorará la posibilidad de compatibilizar 2 compañeros de un mismo grupo que se complementen y se valorará tener grupos mixtos de chicos y chicas. La idea que perseguimos como docentes es crear un grupo que represente la realidad y variedad dentro de la clase, pero también que represente la realidad profesional y ponga a prueba las competencias sociales y personales del alumnado.

Marco teórico

Para iniciar este trabajo hemos formado grupos de cuatro siguiendo la metodología PBL o Aprendizaje Basado en la resolución de Problemas o Proyectos. Actualmente “la idea de trabajar con este procedimiento funciona porque motiva la activación del aprendizaje previo en el entorno de grupos pequeños. Además existe evidencia que las actividades PBL facilitan la comprensión de nueva información relacionada con el problema, mejoran la memoria a largo plazo y protege contra el abandono escolar motivando el alumnado a estudiar con regularidad” (Henk G Schmidt 2011, traducido del inglés).

Para situar este trabajo de innovación y mejora nos basamos en parte en el análisis del entorno, estudio de las necesidades del centro y de la comunidad. Por este motivo, como hemos avanzado en la introducción, este proyecto de innovación está formado por el conjunto de los alumnos que constituyen el curso de un CFGM en: *Instal·lacions elèctriques i automàtiques*. Si bien, en su primera fase del mismo, los alumnos se distribuirán en grupos de 4 alumnos.

Para realizar el proyecto tecnológico y diseñar la actividad hemos tenido en cuenta las fases:

- Idea
- Diseño
- Ejecución
- Evaluación

Partiendo de la necesidad de crear un proyecto tecnológico vinculado con el medioambiente, hemos iniciado la concreción de soluciones e ideas. En la búsqueda de una opción para trabajar por cambio a un modelo más ecológico. Las dimensiones de esta tarea son tanto personales como sociales. El potencial para la acción personal, o basada en la comunidad es claro. El cambio se puede experimentar en la forma en que vivimos, las cosas que compramos, la forma de energía que consumimos…

El proyecto PBL se estructura con un total de 5 sesiones para el alumnado incluyendo 2 sesiones de taller, con la propuesta de vincular disciplinas tecnológicas, creativas y sociales. La propuesta busca innovar en la creación de un proyecto que explore campos como el conocimiento del entorno y del concepto de eco sostenibilidad responsable y la aplicación de la tecnología para un proyecto real.

Para situar nuestro objetivo docente hemos reunido nuestras habilidades y expresado nuestras motivaciones. Hemos tenido en cuenta nuestra formación y experiencia y también hemos revisado algunos puntos en común en el currículum de los CFGM como:

1) Aplicar y analizar las técnicas necesarias para mejorar los procedimientos de calidad del trabajo en el proceso de aprendizaje y del sector productivo de referencia.
2) Analizar y utilizar los recursos existentes para el aprendizaje a lo largo de la vida y de las tecnologías de la información y la comunicación para aprender y actualizar los conocimientos, reconociendo las posibilidades de mejora profesional y personal, para adaptarse a distintas situaciones profesionales y laborales.
3) Analizar riesgos ambientales y laborales asociados a la actividad profesional, relacionándolos con las causas que los producen, con el fin de fundamentar las medidas preventivas que se adoptaran, y aplicar los protocolos correspondientes para evitar daños en uno mismo, en las otras personas, en el entorno y en el medio ambiente.

Como objetivos nos proponemos:

- Desarrollar un proyecto que promueva aplicar técnicas de diseño y construcción con el uso de materiales eléctricos y electrónicos.
- El análisis y aplicación de criterios tecnológicos, estilísticos, organizativos y presupuestarios.
- La determinación y correcta aplicación de los medios necesarios.
- El conocimiento del entorno y el contexto.
- Desarrollo de capacidad crítica y de trabajo en equipo.

A su vez, teniendo en cuenta que se trata de alumnado de ciclo formativo de grado medio, para desarrollar el proyecto se seguirán objetivos propios de la disciplina como: seguir procesos de documentación y uso de manuales técnicos, manipular material y herramientas técnicas propias para la actividad, dominar el uso de programas técnicos de programación y el uso y control de herramientas de trabajo.

La metodología

El PBL o Aprendizaje Basado en Problemas permite que los docentes efectúen el seguimiento de un proceso creativo y de las tomas de decisiones que sigue el alumnado hasta desarrollar el proyecto. Esta actividad está planteada para que el problema sea el punto de partida que estimule al alumnado a diseñar, construir, conectar, y programar el sistema de control de la Sala de Actos en una maqueta. Y siempre teniendo en cuenta todos los objetivos descritos con anterioridad.
Es una forma completa de aprendizaje donde los alumnos con el trabajo en grupo, enriquecen sus conocimientos y participan activamente en todo el proceso productivo.
Como docentes, para empezar con un procedimiento de diseño de innovación, se deben tener en cuenta:

1) Los procedimientos: se estudia cómo mejorar el procesos de Enseñanza/Aprendizaje de los Módulos involucrados en la actividad.
2) Las personas: Prever el nivel educativo de los alumnos y la formación del Profesorado; y también las actividades que se introducen.
3) Conocimiento: Con qué conocimientos trabajaremos el proceso E/A. ¿Qué experiencias previas existen? Qué conocimiento tengo? ¿Hace falta formación?
4) Tecnología: En función de lo que se quiere mejorar en el proceso E/A se escoge una tecnología que se adapte al proceso, personas y conocimiento.

En el caso que nos ocupa se parte del Decreto que define el currículo en el siguiente Módulo Formativo:

CFGM	MP	UF	HORAS PBL	HORAS UF	HORA MP
CFGM Instal·lacions elèctriques i automàtiques.	*MP 9: Electrónica*	*UF1: Electrónica Digital* *UF 2: Electrónica Analógica*	**32h**	**UF1: 26h** **UF2: 40h**	**66h**

Para iniciar el diseño del objeto así como para poder plantearlo al alumnado, respondemos a diferentes fases:

1. **Búsqueda del problema.** Primero: El calentamiento global esta causando daños irreparables en nuestro planeta. Son parte de este problema los equipos de refrigeración, climatización y aire acondicionado. Los primeros porqué consumen energía eléctrica de forma masiva i porqué emiten grandes cantidades de gases de origen fluorado, resultantes del proceso químico que utilizan para producir frío a nivel industrial.[1]

   Los últimos, por la gran cantidad de energía eléctrica que consumen en su funcionamiento. Generamos corriente eléctrica de forma masiva en centrales térmicas convencionales, que como residuo emiten CO2 a nuestra atmósfera. Para reducir la emisión de CO2 deberemos reducir el consumo de energía eléctrica.

2. **Definición del problema:** Con este fin, substituiríamos los equipos de aire acondicionado del centro por ventiladores y controlaríamos las luces de las aulas de forma inteligente y conforme a la normativa de instaladores. El objetivo de este Reto PBL es el de invitar a los alumnos a proponer el diseño de la maqueta a escala de nuestro sistema de control de la Sala de Actos. **El cuál se gestionará automáticamente según el número de usuarios que se encuentren en ella.**

3. **Ideación y resolución proyectiva:** La construcción de la maqueta a escala de la Sala de Actos, el diseño de la instalación a escala, del control electrónico, y sus componentes eléctricos, **y la programación del software del micro controlador 'Arduino'**.

4. **Ejecución del diseño:** La construcción de una maqueta del aula a escala en la primera sesión de taller, para simular el correcto funcionamiento de la aplicación electrónica desarrollada sobre una 'protoboard'.

   La programación del sistema, pruebas y correcciones del sistema. Esta instalación se hará durante la segunda sesión de taller.

5. **Evaluación del diseño:** El diseño se evaluará en función del producto final, tanto por su correcto funcionamiento como por su utilidad para reducir el consumo eléctrico del centro educativo.

[1] https://www.miteco.gob.es/es/calidad-y-evaluacion-ambiental/temas/atmosfera-y-calidad-del-aire/emisiones/prob-amb/gases_fluorados.aspx

Partes del diseño.

Uno de los objetivos de este proyecto es de dar a conocer la importancia del cuidado de nuestro ecosistema para paliar el calentamiento global de nuestro planeta. Para ello, deberemos escoger los ventiladores y las lámparas más adecuadas para nuestra instalación, fomentando aquellos cuyo consumo eléctrico sea menor y su funcionamiento más eficiente, y adaptando el diseño a los recursos del centro.

El diseño básico está formado por las siguientes partes, que forman la estructura del sistema. Las enumeramos a continuación, añadiendo diferentes posibilidades (en primer lugar aparecen los relativos a la maqueta).

Parte	Requisitos	Opciones
Diseño del sistema electrónico de control	Debía ser barato, y para ello, no se comprará dispositivos desarrollados que encarecerían el proyecto, sino que se evaluará la creatividad del alumno, y la puesta en práctica de sus conocimientos.	Existían dos opciones: 1. Desarrollar 2 detectores independientes con salida digital con dos arduinos, a conectar a un tercer arduino que habría recibido la señal digital de ambos, o bien, mucho más económico: **2. Incorporar el sensor en el Arduino, y utilizar un Potenciómetro como señal en la puerta…Se ha optado por este sistema**
Programación Arduino	Existen gran multitud de ejemplos y documentos sobre la base de Arduino en la red. Se ha intentado poner orden en este documento. Se ha verificado el funcionamiento de todas las fuentes consultadas, antes de incorporarlas al presente documento.	www.arduino.cc y múltiples proveedores independientes, como por ejemplo, bloggers o distribuidores y fabricantes de dispositivos compatibles con Arduino.
Construcción de la maqueta	Trabajo de artesanía en madera fina.	Se ha optado por construir una maqueta en madera para escenificar el proyecto.

Diseño

Si el lector no tiene conocimientos previos de programación, se recomienda, debería empezar por el anexo3 o manual de Arduino…

Se ha desarrollado una breve presentación del modelo a escala de este proyecto con algunas imágenes fotográficas y vídeos en el link siguiente:

http://elgaratgedelselectrons.blogspot.com/2019/06/maqueta-escala-de-una-sala-de-actos.html

El diseño del producto final tiene una única dimensión diferenciada. El **diseño electrónico** tiene que considerar las características de los materiales escogidos. Los alumnos del ciclo de *Instal·lacions eléctriques i automàtiques* serán los responsables de llevar a cabo estos diseños.

Alimentación y electrónica

La mayor parte de los contenidos didácticos y conocimientos técnicos a aprender por parte del alumnado del CFGM de *Instal·lacions elèctriques i automàtiques* está comprendido en el diseño y construcción de los circuitos electrónicos y su alimentación. En los siguientes puntos se describen los pasos necesarios seguidos en cada una de las partes del primer prototipo construido y enlaces a la documentación y foros consultados.

1. Infrared Sensors IR (Receptor)

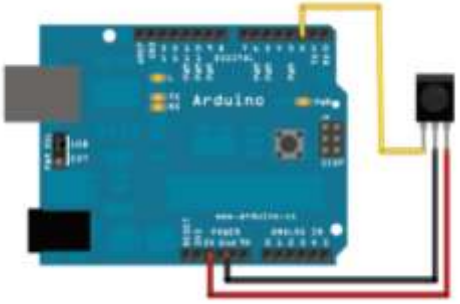
En la lista de materiales electricos y electrónicos, podemos encontrar en el ítem 17 a la derecha el receptor IR y a la izquierda el emisor IR utilizado a 38 Khz.

https://playground.arduino.cc/Code/InfraredReceivers

http://forum.arduino.cc/index.php/topic,17965.html

https://www.arduino.cc/reference/en/language/functions/advanced-io/pulsein/

Conectar el receptor a l'Arduino: https://learn.adafruit.com/ir-sensor/using-an-ir-sensor



Conexión receptor IR definido en la web de Arduino.

Según el modelo de receptor IR, la tensión Vs y la GND dependerán en su patillaje o pin.
Comprueba el datasheet de tu modelo antes de conectarlo.
Conectarlos al revés causará daños irreparables en tu microcontrolador Atmel.(El que lleva la placa Arduino).
Para cambiar el microcontrolador Atmel deberás instalar el Bootloader desde otra placa Arduino, y si no dispones de ella, eso será un problema…

Según el modelo de receptor la tensión de alimentación Vs, y la GND dependerán del Pin:

Ver Datasheet TSOP2438 en anexos.

Este ejemplo muestra la conexión a Arduino de un receptor IR así como la programación de una aplicación para gestionar, en la ventana de 'Serie Monitor', las datos que recibe este. Si, por ejemplo, le enviamos la señal IR de un emisor, que podría ser el mando a distancia de un televisor:

Ventana 'Serie Monitor de la aplicación. Señal recibida por el receptor de la señal enviada por un mando a distancia de un televisor:

a. Delaymicroseconds: El tiempo que esta señal está accionada.
   Puede servir, por ejemplo para determinar el tamaño de un objeto, según pase por delante de una barrera de detección.
b. PulseIR: El código enviado por el emisor, el 500 podría ser TV3, 2820 sería el canal super3..

2. El diodo emisor IR –LD271
https://www.arduino.cc/en/Tutorial/StepperSpeedControl

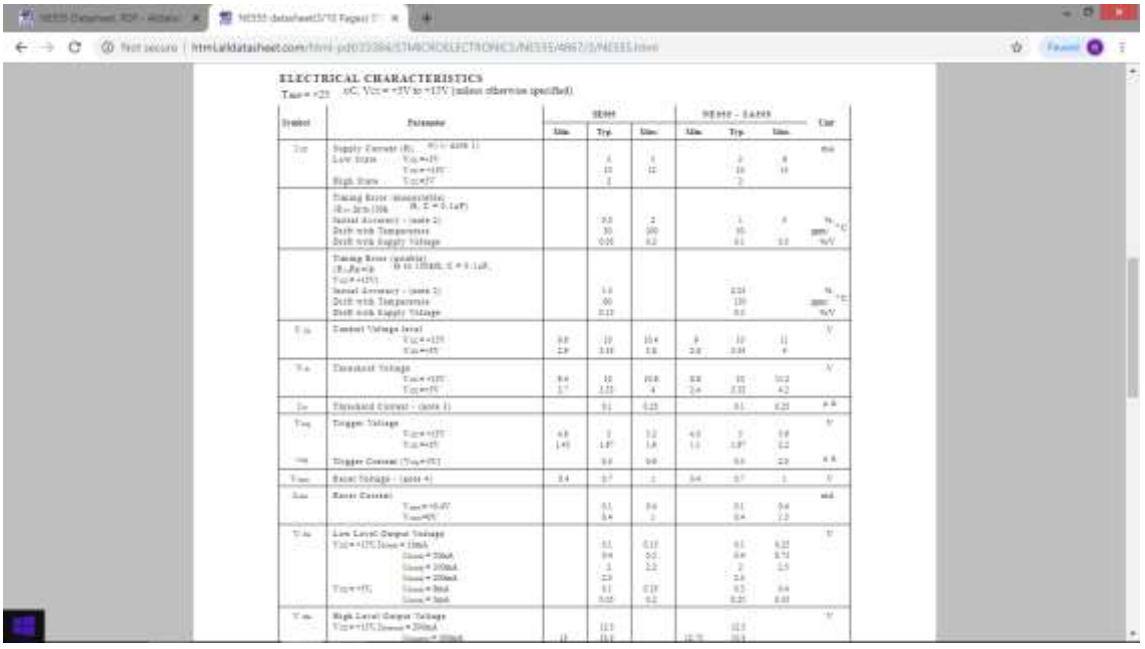
Diode Led emisor 38 Khz de frecuencia.
Ver Datasheet NE555 STelectronics en documentos anexos.

En su configuración de oscilador Astable:

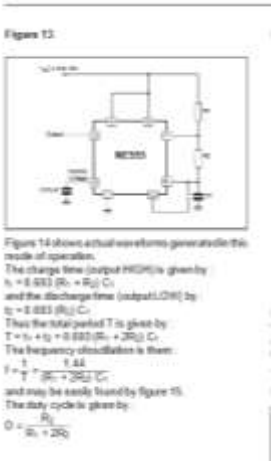
En la página 3 del Datasheet nos dice que:
En su condición de oscilador ‘Astable’, con igual tiempo de carga(t_{ON}) que tiempo de descarga(t_{OFF}), R_a será igual R_b…
…i que I_{cc} será < 6mA

Para que la corriente en los pines del EN555 sea de apenas unos mA, definiremos:
1kΩ<< R_b<<100kΩ.
Para simplificar los cálculos asignamos que

$$\boldsymbol{R_b = 10k\Omega}$$



En la página 7 del Datasheet, se definen las fórmulas necesarias para calcular C1, R_a = R_b…



Si nuestro objetivo es que el tiempo (t_{ON}) sea al (t_{OFF}) , y el período es igual a T=t_{ON} + t_{OFF}:

$$\begin{cases} \boldsymbol{t_{ON} = 0.693\ (R_a + R_b)C_1} \\ \mathbf{t_{OFF} = 0.693\ (R_b)C_1} \\ \\ \ln(2) = 0.693 \\ \dfrac{1}{\ln(2)} = 1.443 \end{cases}$$

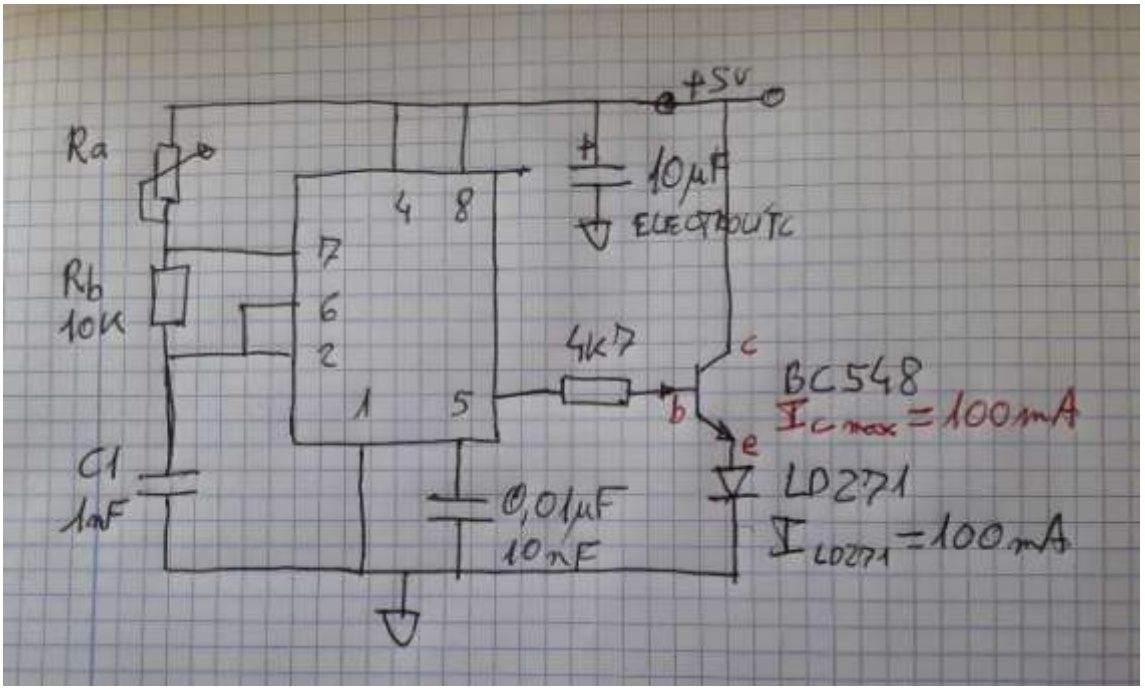
$$\begin{cases} \boldsymbol{f = \dfrac{1}{\ln(2) * (R_a + 2R_b) * C_1}} \\ f = \dfrac{1}{\ln(2) * (R_b + 2R_b) * C_1} \\ C_1 = \dfrac{1}{\ln(2) * (3R_b) * f} \\ C_1 = \dfrac{1}{\ln(2) * (3 * 10k\Omega) * 38000Hz} \end{cases}$$

$$C_1 = 1.266 * 10^{-9} Faradios$$

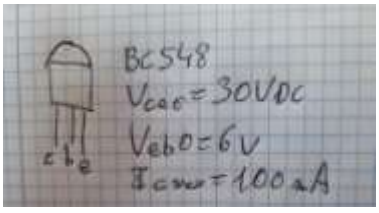
Será difícil encontrar un condensador de ese valor. Consideramos un condensador:
$C_1 = 0.001 * 10^{-6} Faradios$=1*$10^{-9}$Faradios=1 nanoFaradio(nF), que es el valor que todos los fabricantes proporcionan:
Y calculamos R_b, para el nuevo condensador:

$$\begin{cases} \boldsymbol{f = \dfrac{1}{\ln(2) * (R_a + 2R_b) * C_1}} \\ C_1 * f * ln(2) = \dfrac{1}{(\mathrm{R_a} + 2R_b)} \\ (R_a + 2R_b) = \dfrac{1}{C_1 * f * ln(2)} \\ \mathrm{R_a} = \dfrac{1}{\mathrm{C_1 * f * \ln(2)}} - 2\mathrm{R_b} \\ \mathrm{R_a} = \dfrac{1}{0.001 * 10^{-6}\mathrm{F} * 38000\mathrm{Hz} * \ln(2)} - 20000\Omega \\ \\ \mathrm{R_a} = \mathbf{17970}\ \Omega \end{cases}$$

Esto significa que Ra sea variable, entre 10kΩ y 18kΩ. Para ello deberemos usar un potenciómetro que defina estos valores.



El Condensador electrolítico de 10µF en la alimentación estabiliza la alimentación del EN555.
Sabiendo cuál será el consumo de la carga a conectar en el pin emisor del transistor; la corriente del **diodo IR** LD271 es de unos 100 mA, escogemos el transistor, conforme a su datasheet, de corriente máxima de colector Icmax=100mA.
El transistor BC548 tiene los siguientes datos:



3. Programar el control de motores DC por Módulos de control PWM

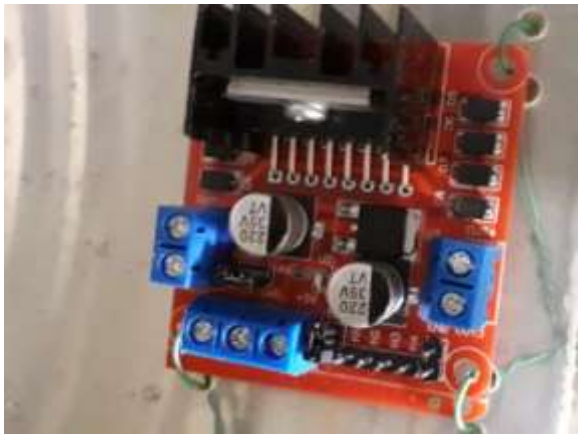
Imagen del motor eléctrico que deberemos controlar a +12 voltios corriente continua para desarrollar nuestro ventilador.

A menudo, los fabricantes y distribuidores de material electrónico desarrollan manuales y programas matriz para facilitar la integración de sus dispositivos a sus clientes. Muchos de estos documentos son claros, sencillos y flexibles de forma profesional.

El siguiente es uno de estos casos. Lo recomendamos.

En este manual aprenderemos a configurar el puente H 'LM298' que se ha optado por utilizar en esta aplicación. Se trata de un dispositivo económico y sencillo de controlar.

https://electronilab.co/tutoriales/tutorial-de-uso-driver-dual-l298n-para-motores-dc-y-paso-a-paso-con-arduino/



Puente H. Es un controlador de motores Paso a Paso y motores DC. En la imagen el LM298.

4. Conexión del motor DC al equipo

Los circuitos con emisores y receptores Led IR, con emisores /receptores radiofrecuencia…son circuitos muy sensibles a los picos de corriente emitidos por motores.

Al ponerse en marcha se observó que el equipo dejaba de funcionar correctamente. En concreto, el contador de Usuarios de la Sala se incrementaba automáticamente.

Para ellos se buscó la solución en los siguientes webs:

La mejor de las soluciones era montar un diodo en los polos del motor, según indican los links:

https://www.youtube.com/watch?v=0ujOHSHYA1w

https://forum.arduino.cc/index.php?topic=32257.0

En este fórum se indicaba se debía utilizar el diodo 1N4007, del que yo disponía. Solo disponía de este modelo y de otro del tipo SB360, para regular circuitos eléctricos a 3 amperios de corriente.

El modelo 1N4007 funcionó durante 2 horas. En un arranque del motor dejó de hacerlo. Probablemente, era demasiado pequeño.

Utilizar el diodo regulador modelo SB360 podría dañar seriamente el puente H …

La siguiente opción fue probar las velocidades a las que se podía configurar el motor (de 0 a 255 puntos) hasta encontrar la única a la que el ruido era mucho menor. A la velocidad de 30 puntos, solo se percibía una subida ocasional de la frecuencia de contaje de usuarios. Este fallo desapareció al trenzar los dos cables monofilares, tal y como se aprecia en la imagen:

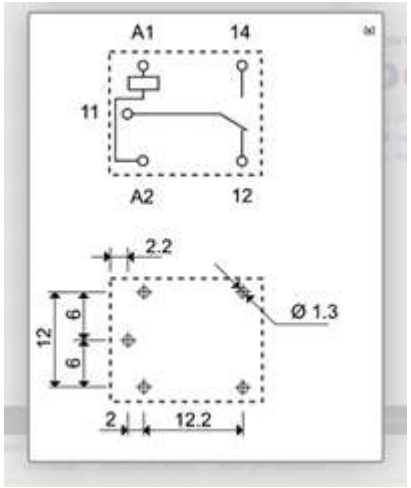
Esta solución no pasaría los controles de Emisiones Radiadas y inmunidad radiada en un laboratorio de ensayos. La opción correcta es buscar el diodo adecuado.

5. **Conexión pin salida 'Arduino' a Relé.**

La alimentación de la bombilla utilizada en el modelo, así como los cables que conectan ésta a los pines del relé donde está el contacto conmutado se alimentan a 240 Vca.
El procedimiento para evitar todo daño personal será el de desconectar la alimentación para manipular esta parte del sistema.

Datasheet del Relé que vamos a utilizar.

https://www.tme.eu/en/details/36.11.9.005.4001/miniature-electromagnetic-relays/finder/361190054011/?utm_source=octopart.com&utm_medium=cpc&utm_campaign=compare-2019-03



Conexionado del Relé de potencia con bobina a 5 VDC que vamos a usar.

LA Bobina corresponde a A1, A2.

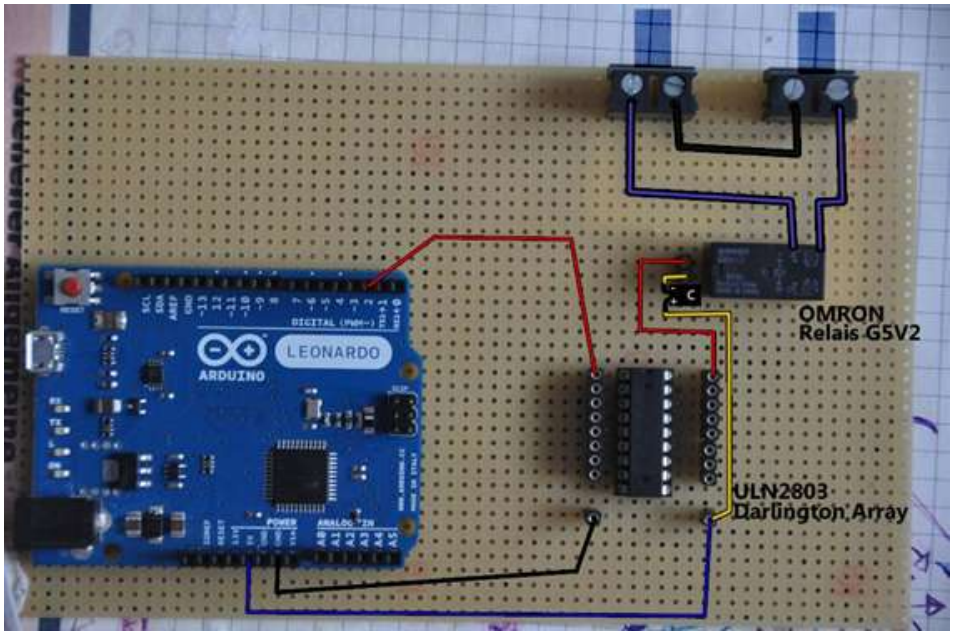
EL contacto Conmutado, normalmente abierto(NO) es 11,14.

El Contacto conmutado , normalmente cerrado (NC) es 11, 12.

Datasheet del Toshiba ULN2804A para controlar relé:

https://html.alldatasheet.com/html-pdf/182616/TOSHIBA/ULN2804A/295/1/ULN2804A.html

Existen múltiples modelos de circuitos integrados ULN. Són amplificadores de los conocidos como Darlington. Su conexionado puede variar según el modelo.
Se deberá consultar al datasheet antes de proceder a su conexión. Una mala conexión puede dañar el circuito integrado.



Puedes encontrar información sobre la conexión de un relé a ‘Arduino’ mediante un circuito integrado ULN2804A en los siguientes links:

https://forum.arduino.cc/index.php?topic=157018.0 Conexión ARDUINO Forum

https://www.google.com/search?q=connection+Toshiba+ULN2804A+examples&tbm=isch&source=iu&ictx=1&fir=hNMDFy54LT2FgM%253A%252CDBXaOsq4e-lmfM%252C_&vet=1&usg=AI4_-kQfz6SdtLpIh0H_WlXisposvxpArQ&sa=X&ved=2ahUKEwiR_9GSgpjhAhVCTBoKHb46DlAQ9QEwA3oECAkQCA#imgrc=JyhtBTUCHJNivM:&vet=1

https://www.youtube.com/watch?v=5k5Li4XRUb4

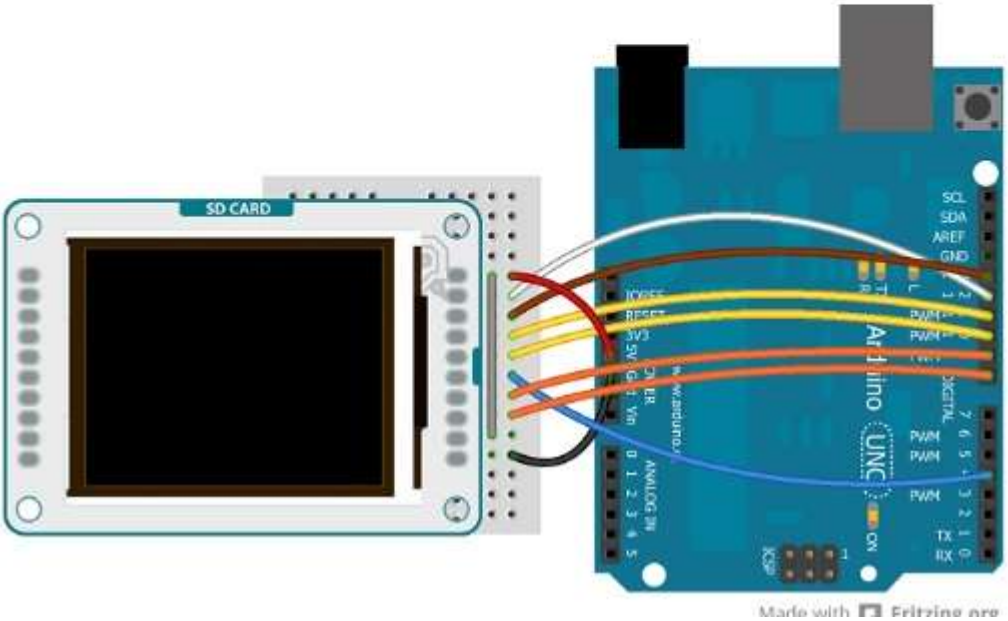
6. Conexión y configuración de una pantalla TFT

Se puede encontrar información a propósito sobre la pantalla TFT utilizada con referencia A00096 de Arduino en los Links:

https://www.arduino.cc/en/Guide/TFTtoBoards
https://www.arduino.cc/en/Reference/TFTLibrary

El patillaje de conexión con el Arduino UNO es el de la siguiente imagen:

+5V:	+5V
MISO:	pin 12
SCK:	pin 13
MOSI:	pin 11
LCD CS:	pin 10
SD CS:	pin 4
D/C:	pin 9
RESET:	pin 8
BL:	+5V
GND:	GND



Librería de ejemplos para pantalla TFT.

En el archivo TFTtoBoards el pin SD_CS de la pantalla se conecta a pin 5, en el ejemplo TFTLibrary, el pin SD_CS de la pantalla se conecta al pin 4.

Los archivos solo pueden ser en formato *.bmp, y de 160x128 píxels.

```
// incluye las librerías necesarias
#include <SPI.h>
#include <SD.h>
#include <TFT.h>  // Arduino LCD library

// definición de los pin para 'Arduino' UNO. La conexión es diferente para todos los modelos
#define sd_cs  4
#define lcd_cs 10
#define dc     9
```

```
#define rst    8
TFT TFTscreen = TFT(lcd_cs, dc, rst);

// estas son las variables que etiquetan cada una de las imágenes en la tarjeta SD
PImage logo;
PImage logo2;
PImage logo3;


void setup() {     //Esta parte del programa solo se ejecutará una vez
  // inicialitza GLCD y abrimos el puerto serie
  TFTscreen.begin();
  TFTscreen.background(255, 255, 255);

  TFTscreen.stroke(0, 0, 255);
  TFTscreen.println();
  TFTscreen.println(F("Arduino TFT Bitmap Example"));
  TFTscreen.stroke(0, 0, 0);
  TFTscreen.println(F("Open serial monitor"));
  // inicializa el puerto serie: lo vamos a usar para mostrar la información de
diagnóstico
  Serial.begin(9600);
  while (!Serial) {
  }

  // Limpia el GLCD después de reiniciarse
  TFTscreen.background(255, 255, 255);
//Intentamos abrir la tarjeta SD. Si falla, el proceso se detendrá…
  Serial.print(F("Initializing SD card..."));
  if (!SD.begin(sd_cs)) {
    Serial.println(F("failed!"));
    return;
  }
  Serial.println(F("OK!"));

  // inicializa y limpia la pantalla GLCD
  TFTscreen.begin();
  TFTscreen.background(255, 255, 255);

//Ahora ya tenemos acceso a la tarjeta SD, intentamos cargar las imágenes.
    logo = TFTscreen.loadImage("scrooge1.bmp");
  if (!logo.isValid()) {
    Serial.println(F("Ha habido un error cargando scrooge1.bmp"));
  }
    logo2 = TFTscreen.loadImage("scrooge2.bmp");
  if (!logo2.isValid()) {
    Serial.println(F("Ha habido un error cargando scrooge2.bmp "));
```

```
  }
    logo3 = TFTscreen.loadImage("scrooge3.bmp");
  if (!logo3.isValid()) {
    Serial.println(F("Ha habido un error cargando scrooge3.bmp "));
  }
}
//Esta parte del programa se repetirá de forma indefinida.
void loop() {
  // Si la imagen no se carga correctamente, no sigas cargándolas
  if (logo.isValid() == false) {
    return;
  }
    if (logo2.isValid() == false) {
    return;
  }
    if (logo3.isValid() == false) {
    return;
  }
  Serial.println(F("dibujando imágenes"));
  int x=0;
  int y=0;

  TFTscreen.image(logo, x, y);            // dibuja la imagen 1 en la pantalla



  delay(2000);          // espera 2 segundos

  TFTscreen.image(logo2, x, y);     // dibuja la imagen 2 en la pantalla

  delay(2000);            // espera 2 segundos
  TFTscreen.image(logo3, x, y);     // dibuja la imagen 3 en la pantalla

  delay(2000);            // espera 2 segundos
}
```

7. **Sensor de luz LDR**

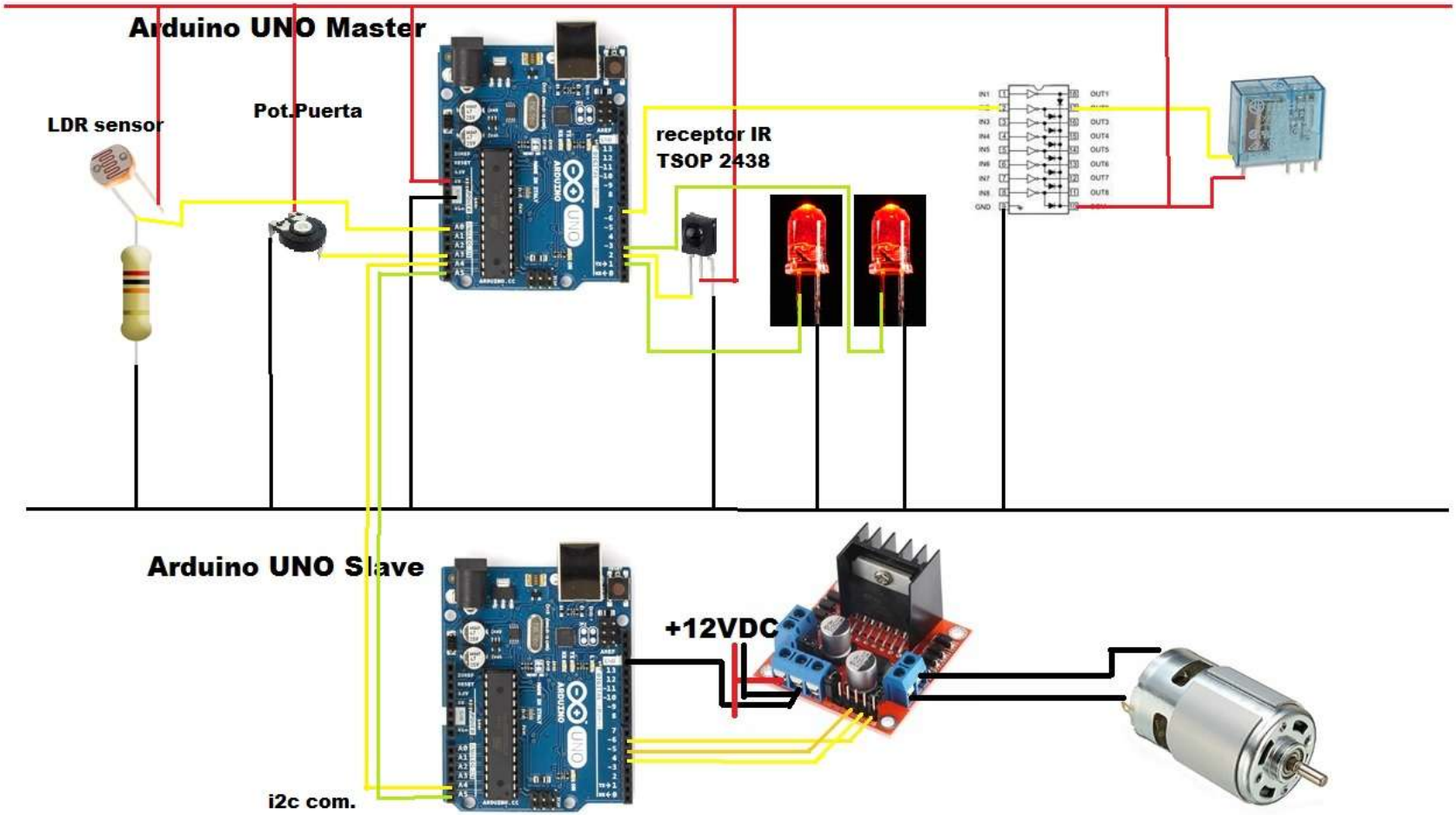
Podemos encontrar un buen tutorial sobre las células LDR, su conexión, su funcionamiento, y el programa de Arduino a desarrollar en el siguiente Link:

https://www.luisllamas.es/medir-nivel-luz-con-arduino-y-fotoresistencia-ldr/

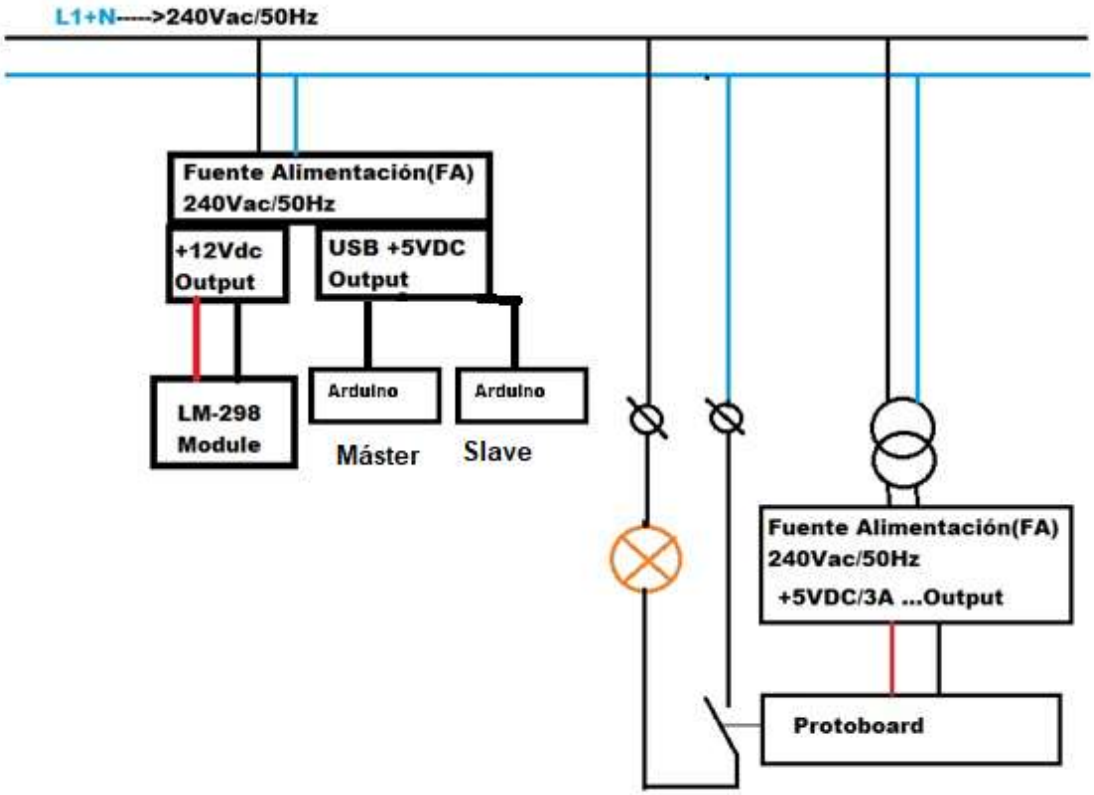
En el siguiente esquema se puede apreciar su conexión al Arduino.

Se puede también estudiar el uso que se ha hecho de ésta en nuestra aplicación Modelo.

8. Conexionado Arduino Máster con Arduino Slave y otros dispositivos(Excepto pantalla TFT)



9. Diagrama de bloques de conexión Fuentes de alimentación.



El sistema está alimentado a 240 Voltios, 50 Hz de corriente alterna. Se ha utilizado dos fuentes de alimentación independientes por motivos de presupuesto.

La primera con dos salidas, una regulable a varias tensiones por un selector, ha sido configurada a +12 voltios corriente continua, para alimentar el motor, y el driver LM-289 que lo controla.

La segunda salida es un conector USB a +5VDC, y nos sirve para alimentar el arduino.

La segunda fuente de alimentación, con salida a +5VDC nos sirve para alimentar el circuito de mayor consumo de la 'Protoboard'. Puede suministrar hasta 3 amperios de corriente contínua.

En el interior de la caja de plástico donde estan los dispositivos electrónicos podemos encontrar también 2 bornes a 240 voltios corriente alterna. Se utilizan para alimentar la bombilla luminosa controlada por arduino, a través de un relé.

Programación Arduino

Diagramas de flujos o pseudocódigo

El diagrama de flujos o el pseudocódigo sirven para estudiar el programa antes de empezar a desarrollarlo y para compartir los puntos desarrollo, delegándolos por subestructuras llamadas funciones, cuando se trabaja en grupo. En este caso, ambas técnicas son imprescindibles…

Tiende a ser más complejo, cuánto mayor lo es el programa.
Yo personalmente prefiero el pseudocódigo. Pero es una elección personal.

Pseudocódigo Arduino Máster de la Aplicación

El receptor IR Infrarrojo del acceso de la puerta recibe una señal
    Enciende el Led Indicador, se apagará al cabo de 2,6 seg., para avisar se ha recibido una señal.
    Cuentas el número de Usuarios.
    ¿La puerta está abierta?
        El usuario está saliendo.
        ¿El Número de usuario es mayor a 0?
            Resta un Usuario al contador
        ¿El Número de usuario NO es mayor a 0?
            Ya NO sigas restando usuarios.
    ¿La puerta está cerrada?
        El usuario está entrando.
        ¿El Número de usuario es mayor a 32?
            Ya no sigas contando los usuarios. Este usuario ya no debería entrar en el aula. Se ha superado el aforo máximo
        ¿El Número de usuario NO es mayor a 32?
            Incrementa el Número de Usuarios.
    Mostrar el Número de Usuarios en la Pantalla TFT.
    Enviar el Número de Usuarios al Arduino Slave.
    ¿Si el usuario es ‘0’?

Se apagarán las luces de la Sala de Actos.
¿Si el usuario es ‘1’?
¿El sensor LDR es inferior a parámetro definido?
Encender las luces de la Sala de Actos.
¿El sensor LDR NO es inferior a parámetro definido?
NO encender las luces de la Sala de Actos.

El receptor IR Infrarrojo del acceso de la puerta NO recibe una señal
No hagas NADA

Pseudocódigo Arduino Slave de la Aplicación

¿Se ha ejecutado la instrucción EndTransmission en el Arduino Máster?
Leemos el valor de la variable que ha enviado el Arduino Máster. “comptUsuarios”
¿El valor de comptUsuarios es de 0 a 5?
El motor permanecerá cerrado
¿El valor de comptUsuarios es de 6 a 34?
El motor se pondrá en marcha.
¿El valor de comptUsuarios es cualquier otro?
El motor permanecerá apagado

¿NO se ha ejecutado la instrucción EndTransmission en el Arduino Master?
No hagas NADA.

Antes de programar Arduino, sino has programado nunca, debes saber que:

1. **Las dos contrabarras (//) definen anotaciones de texto que sirven al programador y que no afectan a la ejecución del programa.**
2. **Al final de cada línea de órdenes se define (;) punto y coma para separla de la siguiente línea de órdenes.**
3. **Los textos en Serial.println() se escriben entre comillas(“ “). A menudo, usamos esta instrucción y el ‘Monitor Serie’ para evaluar la ejecución del programa, y como ‘baliza de señalización’ - `por llamarlo de algún modo- en el momento de la depuración.**

4. La expresión ‘int variableNombre = 5;’ un solo igual significa que el número ‘5’ se cargará a ‘variableNombre’.
   En los condicionales if se emplea ‘==’ para comparar el estado o valor de una variablecon una constante o el valor de otra variable…
   ‘if(variableNombre==5){‘
5. Recomendamos que:
   …gestiones las tabulaciones. Con un nuevo condicional o bucle anidado, desplaza el texto del código fuente a la derecha…
   …etiqueta los corchetes finales de los condicionales y bucles. Te servirá para depurar y seguir programas muy anidados sin perder el tiempo.

Código fuente de la aplicación comentado.

Código fuente del Arduino Máster

```
//Las instrucciones y la estructura de comunicación de dos Arduinos se pueden consultar en las webs:
// https://programarfacil.com/blog/arduino-blog/conectar-dos-arduinos-i2c/
// https://www.arduino.cc/en/Tutorial/MasterReader
//Añadimos la librería para la comunicación I2C
#include <Wire.h>

//Añadimos la librería para gestionar la pantalla TFT
#include <TFT.h>
#include <SPI.h>

//Definimos que Arduino UNO utiliza para la comunicación con la pantalla TFT
#define cs   10
#define dc   9
#define rst  8

//Creamos una instancia para la comunicación con la pantalla TFT
TFT TFTscreen = TFT(cs, dc, rst);
```

```
//La pantalla TFT solo muestra cadenas de caracteres, y nosotros tenemos que mostrar un
número integer //de dos dígitos. Habrá que pasarlos a string. Para posteriormente, pasarlos a una
array de cadena de //caracteres.
char UsuariosPrintout[4];

// el máximo tiempo de lectura que recibiremos es de 10 pulsos….podría ser de 65000 pulsos,
pero no necesitamos una señal tan larga. Solo es una barrera de paso.
#define MAXPULSE 10

// la resolución podría ser de 20 puntos, pero no lo necesitamos. Esa resolución nos perrmitiría
localizar en un eje de 2 coordenadas 'x' e y' donde están el emisor, el objeto que ha pasado por la
barrera, y lo grande que es, pero no nos interesa. Nos basta con saber que ha pasado.

#define RESOLUTION 2

uint8_t currentpulse = 0; // este será el índice de nuestros pulso

//Definimos los pines del Arduino.
const int LedEntrant = 1;
const int LeDetectorPorta=3;
const int  SensorPorta = A3;   //Potenciometro que representa el estado de la puerta:'abierto' o 'cerrado'.

//Lo mismo para la parte de la iluminación, el relé de la cuál se activa a través del pin 7 del
Arduino, y el sensor LDR recibe la señal analógica en el pin A0.
const int EncendreLLums = 7;
int LDRPinSensor = A0;
```

```
const int LimitActLlum = 900;
```

//En la programación de microcontroladores, en el que no existen debuggers o software de compilación complejos, tenemos que interactuar habitualmente con el dispositivo para saber si nuestra aplicación se esta ejecutando correctamente. Para ello se utilizan pantallas TFT, Leds, o el 'Monitor Serie'. En este caso, un Led indicador utiliza las siguiientes variables para mostrarnos que alguién ha pasado por delante de la barrera, y pasado un tiempo de 2 segundos y medio apagarse.

```
unsigned long tempsActual = 0;
unsigned long tempsAnterior = 0;
const long tempsActiuEntrant=2600;

int usuarioEntraOSale = 0;    //Variable a donde se transfiere el valor del sensor para ser comparado
int  comptUsuarios = 0;       //El número de usuarios que han entrado en la sala

int EstatDetectorPorta=0;

void setup(void) {                //Esta parte del programa solo se ejecutará una vez, al recibir alimentación el Arduino.
 Wire.begin();                     //Inicializas el puerto I2C

  Serial.begin(9600);             //Inicializas el 'Monitor Serie'
    pinMode(LedEntrant, OUTPUT);
    pinMode(LeDetectorPorta, OUTPUT);
```

//Configuramos el estado de los pines del Arduino que vamos a usar para controlar el motor como pin de SALIDA. Los relativos a la luz

```
  pinMode(EncendreLLums, OUTPUT);
```

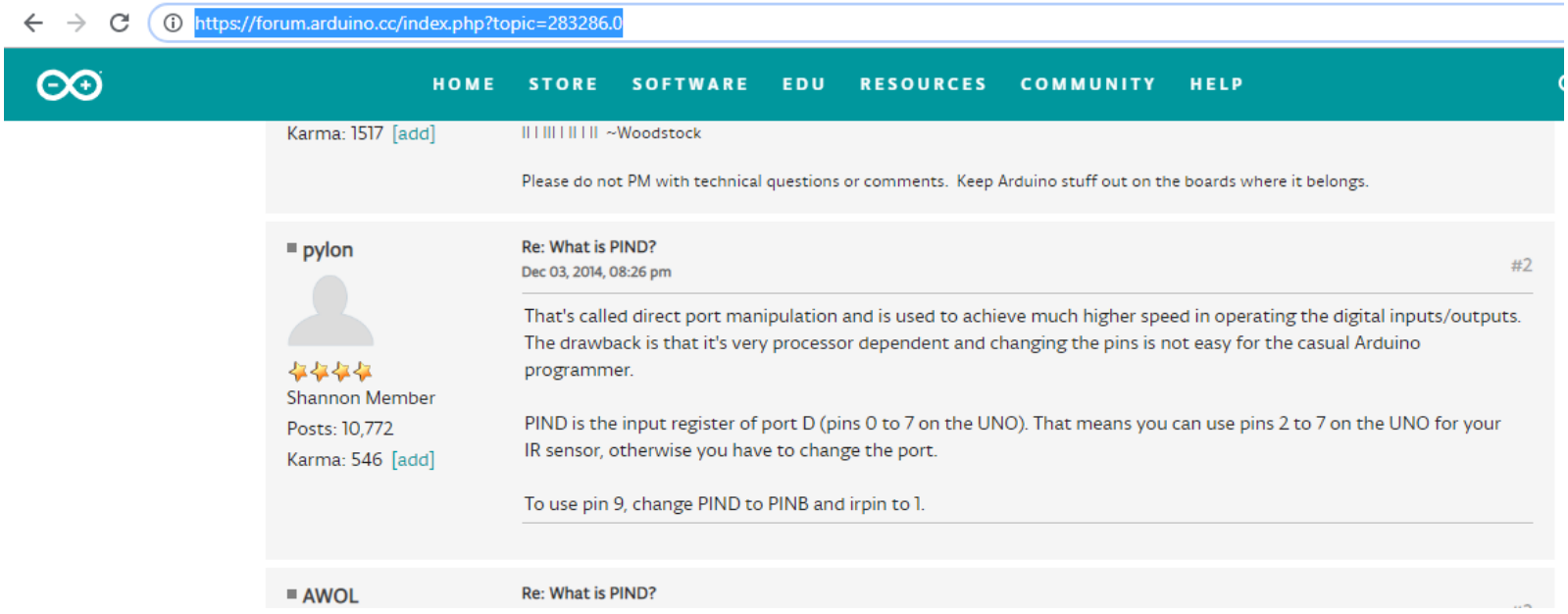
```
pinMode(LDRPinSensor, INPUT);

TFTscreen.begin();            //Inicializas la pantalla TFT a través del GLCD

TFTscreen.background(0, 0, 0);      //Borramos la pantalla TFT

TFTscreen.stroke(255, 255, 255); //Para escribir el texto configuramos el color del mismo a 'Blanco'
TFTscreen.setTextSize(2);        //Y definimos el tamaño de la letra  a (2) para el texto estático "Num.Usuarios"
// Empezamos a escribir desde la esquina superior izquierda.
TFTscreen.text("Num.usuarios:\n ", 0, 0);
//Definimos el tamaño de la letra para el texto relativo al número de Usuarios de la Sala de Actos.
TFTscreen.setTextSize(5);

  digitalWrite(LedEntrant,LOW);   //Inicializamos las salidas del Arduino a '0'
```

El siguiente delay() de 4 segundos es importante. ¿Para qué sirve?

```
delay(4000);
}  //Final del void setup(void)


void loop(void) {        //Esta parte del programa se ejecutará de forma continua e indefinida hasta que el Arduino pierda la alimentación.
```

```
    LlegirSensorEntrant();   //Llamamos a la Función que cuenta los flancos del sensor IR Infrarrojo, y activa luces, communica por I2C con el Slave…


}//Final del void loop(void)

//Función que cuenta los flancos del sensor IR Infrarrojo, y activa luces, communica por I2C con el Slave…
void LlegirSensorEntrant(void){

/* La parte del código del Receptor IR Infrarrojo se ha modificado a partir del código fuente público y gratuito de la Web www.ladyada.net y del suministrador de código didáctico adafruit.com.

 En su desarrollo se ha utilizado para un Arduino UNO un emisor IR Infrarrojo a 38 KHz, y un receptor TSOP de señal IR Infrarroja. El emisor IR Infrarrojo no genera una señal cuadrada, de niveles, sino que genera señales de flanco, que pueden ser vistas solo con la ayuda de la cámara del teléfono móbil.
El programa gestiona la señal del receptor cuando su señal esta a LOW, y cuando está en HIGH, cuenta los flancos que se producen en durada y resolución definidas.
Para poder manipular el contaje de los flancos utilizaremos un mapeado 'raw' de la lectura del pin.
Esta técnica viene definida en un ejemplo sencillo en la página:
//                http://arduino.cc/en/Hacking/PinMapping168
*/

//A continuación definimos el pin del receptor IR Infrarrojos. No es un pin de entrada que se lea con la instrucción digitalRead(), sino que se trata de un acceso directo al registro de la entrada. ¡¡Es mucho más Rápido!! Pero no podemos acceder así a todos los pins del equipo. Su definición vienen definida por los siguientes expresiones:
```

```
//#define IRpin_PIN      PIND      //que se conecta  al pin (2) del Arduino UNO
//#define IRpinSortint       2
//#define IRpin_PIN     PINB      //que se conecta  al pin (9) del Arduino UNO
//#define IRpinSortint       1
//No pueden ser ambos. Escogemos el primero porqué el pin 9 lo utiizaremos para otra cosa…
//La información la podemos encontrar en la siguiente página:
/*      https://www.arduino.cc/en/Reference/PortManipulation
        https://forum.arduino.cc/index.php?topic=283286.0
```



```
*/

#define IRpin_PIN      PIND
#define IRpinEntrant         2

uint16_t highpulse, lowpulse;           // timing de la memorización temporal
  highpulse = lowpulse = 0;     // Empieza con el pulso a 0. sin dar señal.


//  while (digitalRead(IRpin)) { // Así es demasiado lento! Hay que acceder directamente al registro PIND
   while (IRpin_PIN & (1 << IRpinEntrant)) {                // cuando el pin permanezca en HIGH

          highpulse++;
          delayMicroseconds(RESOLUTION);

          // Si el pulse es demasiado largo, nosotros 'timed out' – y nos preparamos para medir el siguiente
          if ((highpulse >= MAXPULSE) && (currentpulse != 0)) {
               digitalWrite(LedEntrant,HIGH);

               tempsActual= millis();
               tempsAnterior = millis();
         }

     //}
  ////////////Mostrar el estado actual actual del led que solo se enciende cuando recibe una señal del receptor IR Infrarrojo
  if (digitalRead(LedEntrant)==HIGH){
```

```
        tempsActual = millis();
            if (tempsActual- tempsAnterior >= tempsActiuEntrant){     //Si el tiempo que lleva encendido el LED es mayor a 2,6 segundos, el Led
se apaga.
                digitalWrite(LedEntrant,LOW);

            }
            else
            {
               digitalWrite(LedEntrant,HIGH);              }

     currentpulse=0;
      return;
    }

 ////////////////////////////////////Final de estado actual de ledEntrant

 }                //Final While del flanco a HIGH
    ////////////////////////////////////////////A partir de aquí, el programa cuenta el Número de Usuarios de la Sala de Actos.
   usuarioEntraOSale = analogRead( SensorPorta );     //El potenciómetro nos dice si la puerta esá abierta o cerrada.
   Serial.println("valor puerta:  ");
   Serial.println(usuarioEntraOSale);
   if (usuarioEntraOSale > 700){
     //La puerta estaba abierta. Por lo tanto, el usuario estaba saliendo
          if(comptUsuarios > 0 ){
               comptUsuarios= comptUsuarios -1;
               Serial.println("La porta estaba abierta.");
          }
          else       //Cuando el valor de Uuarios se igual a '0' no seguirá descontando.
          {
```

```
        }


    }
    else
    {
      //La puerta estaba cerrada. Por lo tanto, si el usuario no ha podido abrir la puerta antes de
acercarse a la fotocelula, es porqué el usuario estaba entrando.
        if(comptUsuarios <= 32 ){      //Si el nombre d'usuaris de la Sala d'actes es major a '32'
            comptUsuarios= comptUsuarios + 1;
            Serial.println("La porta estaba cerrada.");
        }
        else
        {   //Si el Número de Usuarios de la Sala de Actos es mayor a 32, ya no sigues incrementando, porqué ya no caben más…

        }

    }

      Serial.println("El numero usuario es:");    //Y Mostramos el valor en el Monitor Serie' para poder depurar el programa.
      Serial.println(comptUsuarios);
 ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
 // A continuación, seleccionamos lo que hará la máquina según el número de usuarios de la sala de
Actos
 ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
 ////////////////////////////Mostrar número de usuarios por la pantalla TFT
//Pasamos el valor en integer a un string
String ComptUsuariosPAssatAString = String(comptUsuarios);
```

```
// convert the reading to a char arrayConvertimos el valor de String a un array de caracteres….
ComptUsuariosPAssatAString.toCharArray(UsuariosPrintout, 4);

  TFTscreen.stroke(255, 255, 255);   //Configura el color de la fuente con que se va a mostrar el Número a 'Blanco'
  TFTscreen.text(UsuariosPrintout, 0, 20);            //Muestra el valor de los Usuarios 20 líneas por debajo del text estático de Número de
Usuarios
  delay(250);                        //Espera 250 mseg.
  TFTscreen.stroke(0, 0, 0);           //Borra el texto que habías escrito. Solo queríamos que los mostrase en el momento de cambiar.


////////Nos comunicamos con el Arduino Slave para transmitirle elnuevo Número de Usuarios
de la Sala.

  Wire.beginTransmission(1);              // Comenzamos la transmisión al dispositivo 1

  Wire.write(comptUsuarios);             // Enviamos un byte, será el pin a encender

  Wire.endTransmission();                // Paramos la transmisión

  delay(1000);             // Esperamos 1 segundo para estabilizar el sistema
////////////////////////////////////////////////Control de las luces

if (comptUsuarios==0){        //Cuando el Número de Usuarios sea igual a '0' las luces permanecerán apagadas.

          digitalWrite(EncendreLLums, LOW);
      }
      else
      {
```

```
        if (comptUsuarios >=1){                    //Si el número de usuarios es igual a ‘’1’.
            int input = analogRead(LDRPinSensor);          //Leemos el estado del sensor LDR
                Serial.println("El LDR llegeix:");
                Serial.println(input);

            if (input < LimitActLlum) {
              digitalWrite(EncendreLLums, HIGH);
            }
            else {
              digitalWrite(EncendreLLums, LOW);
            }
        }
    }

////////////////////////////////////////
 while (! (IRpin_PIN & _BV(IRpinEntrant))) {       // cuando el pin permanezca en LOW

  if(! (IRpin_PIN & _BV(IRpinEntrant))){
      lowpulse++;
      delayMicroseconds(RESOLUTION);
      if ((lowpulse >= MAXPULSE)  && (currentpulse != 0)) {
           digitalWrite(LedEntrant,LOW);    //Si el pulso esta LOW, el LEd indicador permanece LOW

          currentpulse=0;
          return;
      }
           // Aquí ya hemos leído los flancos HiGH-LOW de una señal
       currentpulse++;
  }
```

```
  }//Final posa a LOW les sortides dels LedEntrant quan NO estiguin activa

}//Final de la Función:   void LlegirSensorEntrant(void){
```

Código fuente del Arduino Slave

```
//Las instrucciones y la estructura de comunicación de dos Arduinos se pueden consultar en las webs:
// https://programarfacil.com/blog/arduino-blog/conectar-dos-arduinos-i2c/
// https://www.arduino.cc/en/Tutorial/MasterReader
//Añadimos la librería para la comunicación I2C
#include <Wire.h>

//Declaramos los Pines del Arduino que vamos a usar para controlar el  motor.
const int IN3 = 5;    // Input3 conectada al pin 5
const int IN4 = 4;    // Input4 conectada al pin 4
const int ENB = 6;    // ENB conectada al pin 6 de Arduino

void setup() {                    //Esta parte solo se va a ejecutar al iniciarse el arranque.
//Configuramos el estado de los pines del Arduino que vamos a usar para controlar el motor como pin de SALIDA
  pinMode (ENB, OUTPUT);
  pinMode (IN3, OUTPUT);
  pinMode (IN4, OUTPUT);
```

```
 // Unimos este dispositivo al bus I2C a la dirección 1
 Wire.begin(1);

 // Registramos el evento al recibir datos
 Wire.onReceive(receiveEvent);

 // Iniciamos el 'Monitor serie' para monitorizar la comunicación
 Serial.begin(9600);

   digitalWrite (IN3, LOW); //Inicializamos el motor a '0' para que no arranque por su cuenta al alimentar la placa...
   digitalWrite (IN4, LOW);
   analogWrite(ENB,0);
}

void loop() {            //Esta parte del programa se ejecutará de forma continua e indefinida hasta que el Arduino pierda la alimentación.
//Al loop() le damos 300 milisegundos entre ciclo y ciclo para que se estabilice.
 delay(300);
}

//La función receiveEvent() se ejecutará en el Slave para leer cada vez que en el Máster se
ejecute la sentencia "endTransmission".
void receiveEvent(int howMany) {

//Contador de número de Usuarios que ha enviado a la función del Slave el Máster
int comptUsuarios=0;
// Cuando la información enviada se podría almacenar en un solo registro, como por ejemplo, una
variable...
```

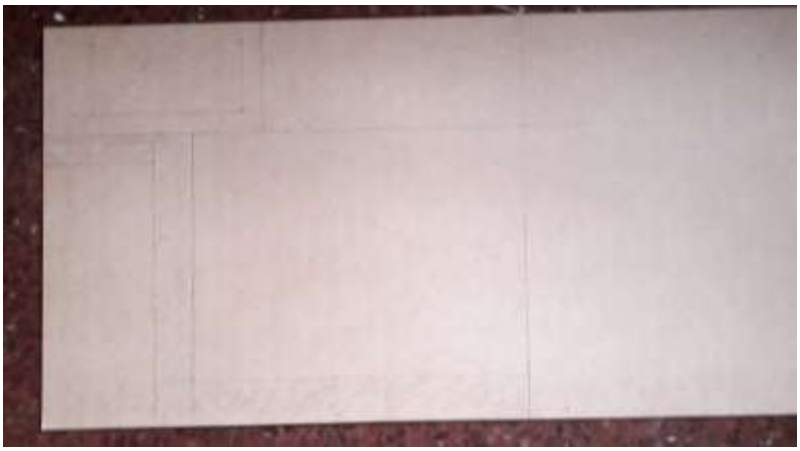
```
 if (Wire.available() == 1)
 {
//Leemos el dato y lo almacenamos en la variable 'comptUsuarios'
   comptUsuarios = Wire.read();
//Mostramos el dato recibido en el 'Monitor Serie' para evaluar se ejecuta correctamente el
programa y las comunicaciones

   Serial.print("Numero de usuarios: ");
   Serial.println(comptUsuarios);
 }

  // A partir de aquí comenzamos a gestionar el código con el dato recibido...
//Si comptUsuarios es inferior o igual a 5, el motor permanecerá PARADO
 if (comptUsuarios<=5){

   digitalWrite (IN3, LOW);
   digitalWrite (IN4, LOW);
   analogWrite(ENB,0);
   Serial.println("Arranque en motor <= 5");
     delay(2000);
 }
 else
 {
//Si comptUsuarios es superior a 5 e inferior  34, el motor se pondrá en marcha
     if((comptUsuarios>=6)&&(comptUsuarios<=33)){
         digitalWrite (IN3, LOW);
         digitalWrite (IN4, HIGH);
         analogWrite(ENB,30);
```

```
        Serial.println("Arranque en motor:entre 6 i 10");
        delay(2000);
    }
    else
    {
//En cualquier otro caso, que NO existe, el motor permanecerá apagado. Es una condición
necesaria en las puestas en marcha que NO sabes nunca lo que le puedes pasar al sistema
hasta que lo has probado…

                    digitalWrite (IN3, LOW);
                        digitalWrite (IN4, LOW);
                        analogWrite(ENB,0);
                        Serial.println("Arranque en motor:en ningún caso todo a 0");
                        delay(2000);

    }//Final if entre 6 y 33
 }//Final if entre 0 i 5
 // FINAL DE CONTROL DEL MOTOR
 ////////////////////////////////////////
}
//Final  del programa del Arduino SLAVE
```

Construcción maqueta

Se define en la parte de materiales el material y las herramientas utilizadas para construir el escenario de la maqueta.
La parte de las paredes y el suelo se ha montado sobre lámina DM rígida. Se necesita una sierra de arco para manipularla, y debe ser perforada con un taladro.
Se han unido entre sí, mediante un listón cuadrado de 9 x 9 milímetros y tornillos de carpintería de 3mm x 10 mm de longitud.
La parte de los muebles se ha montado sobre lámina fina de madera de Tilo que debe ser manipulada con cutter de marquetería.
Se han encolado las partes de las escaleras y de las mesas con cola blanca de carpintero.

La siguiente secuencia de imágenes lo muestra:



Boceto de las paredes y suelo a recortar en la lámina DM.

Secuencia del montaje de las mesas escaleras y pizarra de la maqueta.

A bajo. Detalle de los tornillos usado para fijar paredes y suelo a los listones de 9x9 mm.

Acabado final.

Programación didáctica y temporización

- Sesión 1 - Presentación del proyecto Interciclos (2h)
    - Convocatoria a los alumnos de todos los ciclos participantes
    - Los profesores de todos los ciclos presentan el proyecto
    - Explicación metodología trabajo
    - Sistema de evaluación
    - Objetivos del proyecto para cada ciclo
    - Dinámica de grupos para romper el hielo entre ciclos (warm up)
    - Formación de los grupos de trabajo interciclos
        - Orden de lista
        - 1 o 2 personas por ciclo
        - Test Belbin y dinámica para identificar el perfil de cada alumno
    - Trabajo fuera de clase para los grupos:
        - Elección del diseño a desarrollar
        - Borrador de la idea general
        - Listado materiales

- Sesión 2 – Debate sobre las emisiones de CO2 a la atmosfera (3h)
    - Exponer en grupo los daños que causan a nuestro planeta las emisiones de CO2 y de gases fluorados producidos por el uso abusivo de equipos de aire acondicionado.
    - Los grupos recogerán material para sus proyectos
    - Realizar la lista del material recogido, incluyendo foto, descripción, el precio estimado si lo hubierais comprado y la función que tendrá en vuestro proyecto.
    - Trabajo fuera de clase para los grupos:
        - Ajuste de la idea según materiales disponibles
        - Análisis de material según la tecnología a usar
        - Elaboración del diseño del proyecto y bocetos para el prototipo.

- Sesión 3 - Concreción del Diseño (2h)
    - Por grupos, trabajan en parejas en el diseño de su especialidad
    - Al final de la clase se compartirá rápidamente la idea final
    - El resto de grupos realizarán las aportaciones o críticas constructivas que consideren para poder revisar la idea inicial del proyecto.
    - Trabajo fuera de clase para los grupos:
        - Antes de la siguiente sesión, colgar en el moodle un documento resumen (briefing) del proyecto con los apartados de los tres CFGM.
        - Y para los alumnos de Instalaciones elèctriques:
            - Simulación de los esquemas electrónicos en aplicación ‘orCAD PsPice’.

- Sesión 4 - Taller 1 (11h)
    - CFGM Instal·lacions elèctriques i automàtiques
        - OBJETIVO: Montaje y testeo de los circuitos electrónicos en la placa de pruebas o ‘protoboard’.
    - Elaboración del prototipo en el taller

        - Las 3 personas del grupo del ciclo de electricidad trabajarán en esta sesión en el montaje y testeo de los circuitos eléctricos y electrónicos y su alimentación.
        - Instrucciones:
            - 1) Hacer el conexionado de los componentes electrónicos tal como se diseñó previamente y se comprobó mediante simulaciones. Para esto se utilizará la placa de pruebas o protoboard.
            - 2) Comprobar el correcto funcionamiento de cada parte individual:
                - Fuente de alimentación
                - Sistema de detección de monedas
                - Control del motor para el movimiento del brazo
                - Señales acústicas y lumínicas
            - 3) Una vez testeado de forma satisfactoria se ensamblarán de forma conjunta, formando el conjunto del automatismo.
            - 4) Comprobar el correcto funcionamiento de todo el sistema de forma que quede listo para ser montado en la maqueta a escala en la segunda sesión de taller.

    - Realización de un diario del proyecto
    - Trabajo fuera de clase para los grupos:
        - Avanzar el resto de tareas posibles fuera de taller

- <u>Sesión 5 - Taller 2</u> (11h)
    - CFGM Instal·lacions elèctriques i automàtiques
        - OBJETIVOS: Montaje de la maqueta a escala, Instalación de la electrónica preparada en la sesión anterior, programación del micro controlador ‘Arduino’, prueba del dispositivo y modificaciones del sistema.
    - Finalización del prototipo a escala.
        - En esta última sesión de taller se tendrá que terminar el montaje y programación de la maqueta.
        - Instrucciones
            - 1) Preparar el material electrónico que formará el sistema de la maqueta. Utilizar las herramientas apropiadas para cortar, doblar o unir.
            - 2) Hacer el montaje de todos los componentes electrónicos.
            - 3) Programar el ‘Arduino’ conforme a las condiciones de funcionamiento definidas previamente.
            - 4) Comprobar el funcionamiento de la maqueta y todas sus funciones diseñadas.
            - 5) Añadir las modificaciones y mejoras al sistema.
            - 5) Añadir los acabados y detalles estéticos.

- Sesión 6 - Presentación final proyectos (3h)
    - Realizar la presentación final de los proyectos. Se dispondrá de 5 min por grupo+ la exposición de la maqueta.

CFGM Instal·lacions elèctriques i automàtiques
MP 9: Electrònica
UF1: Electrònica digital.

RA1. Reconeix circuits lògics combinacionals determinant-ne les característiques i aplicacions.
1.1 Utilitza diferents sistemes de numeració i codis.
1.2 Descriu les funcions lògiques fonamentals utilitzades en els circuits electrònics digitals.
1.3 Representa els circuits lògics mitjançant la simbologia adequada.
1.4 Interpreta les funcions combinacionals bàsiques.
1.5 Identifica els components i blocs funcionals.
1.6 Munta o simula circuits.
1.7 Verifica el funcionament dels circuits.
1.8 Identifica les diferents famílies d'integrats i la seva aplicació.
1.9 Realitza les tasques que cal fer individualment amb autosuficiència i seguretat.

RA2. Reconeix circuits lògics seqüencials determinant-ne les característiques i aplicacions.
2.1 Descriu diferències entre circuits combinacionals i seqüencials.
2.2 Descriu diferències entre sistemes síncrons i asíncrons.
2.3 Identifica els components i blocs funcionals.
2.4 Utilitza els instruments lògics de mesura adequats.
2.5 Munta o simula circuits.
2.6 Verifica el funcionament de circuits bàsics seqüencials.
2.7 Descriu aplicacions reals dels circuits amb dispositius lògics seqüencials.
2.8 Realitza les tasques que cal fer individualment amb autosuficiència i seguretat.

UF2: electrònica analògica.

RA1. Reconeix circuits de rectificació i filtratge determinant-ne les característiques i aplicacions.
1.1 Reconeix els diferents components.
1.2 Descriu els paràmetres i magnituds que caracteritzen els circuits amb components passius.
1.3 Utilitza els instruments de mesura adients (multímetre i oscil·loscopi, entre d'altres).
1.4 Relaciona els components amb els símbols que apareixen als esquemes.
1.5 Descriu els tipus de rectificadors i filtres.
1.6 Munta o simula circuits.
1.7 Obté els paràmetres i característiques elèctriques dels components dels sistemes.
1.8 Descriu les aplicacions reals d'aquest tipus de circuits.
1.9 Realitza les tasques que cal fer individualment amb autosuficiència i seguretat.

RA2. Reconeix fonts d'alimentació determinant-ne les característiques i aplicacions.
2.1 Descriu les diferències entre fonts commutades i no commutades.
2.2 Descriu el funcionament dels diferents blocs que componen els sistemes complets d'alimentació.
2.3 Identifica les característiques més rellevants proporcionades pels fabricants.
2.4 Descriu les diferents configuracions de circuits reguladors integrats.
2.5 Munta o simula circuits.

2.6 Utilitza els instruments de mesura adients (multímetre i oscil·loscopi, entre d'altres).
2.7 Descriu aplicacions reals de les fonts commutades.
2.8 Verifica el funcionament de les fonts commutades.
2.9 Realitza les tasques que cal fer individualment amb autosuficiència i seguretat.

RA5. Reconeix circuits de temporització i oscil·lació verificant-ne les característiques i el funcionament.
5.l Reconeix els components dels circuits de temporització i oscil·lació amb dispositius integrats.
5.2 Descriu el funcionament de temporitzadors i oscil·ladors.
5.3 Verifica el funcionament dels circuits de temporització.
5.4 Verifica el funcionament dels circuits oscil·ladors.
5.5 Utilitza els instruments de mesura adients.
5.6 Munta o simula circuits.
5.7 Visualitza els senyals més significatius.
5.8 Descriu aplicacions reals dels circuits amb dispositius integrats de temporització i oscil·lació.
5.9 Realitza les tasques que cal fer individualment amb autosuficiència i seguretat.

●

Activitats d'Ensenyament i Aprenentatge		**RA**	**Continguts**	**Avaluació**	
				CA	**Instruments d'Avaluació**
A1 - Hucha automatizada	**16 h**	**UF1 RA1 RA2** **UF2 RA1 RA2 RA5**	1.1 a 1.8 2.1 a 2.4 3.1 a 3.4 4.1 a 4.2 5.1 a 5.2	UF1 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 UF2 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.1	- Rúbricas de evaluación para cada fase/entrega establecida.
Descripció	** Presentación del proyecto (2h)* ** VIsita a desguace y recogida de materiales (3h)* ** Diseño de la hucha y sus circuitos (2h)* ** Montaje y testeo del automatismo (11h)* ** Montaje de la hucha y acabados (11h)* ** Presentación del proyecto (3h)*				

				2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 5.1 5.2 5.3 5.4 5.6 5.7 5.8 5.9 5.10	

Proceso de trabajo, metodología, evaluación y temporización.

Proceso de trabajo, metodología, evaluación y temporización	
Sesiones: 6	***Dedicación alumnado:***
Semanas: 9	Presencial 16 / No presencial: 10
Proceso de trabajo y objetivos	
En esta actividad realizaremos un trabajo utilizando la metodología PBL que hemos estructurado en seis sesiones: ● En la primera sesión los profesores presentarán el proyecto a los alumnos, explicando la metodología de trabajo, cómo se realizará la coordinación entre grupos, el sistema de evaluación y los objetivos del proyecto. ● En la segunda sesión, se seleccionará y comprará el material necesario, dependiendo de la instalación prediseñada, revisando el concepto de emisiones de CO2 por consumo energético y uso de sistemas de refrigeración, y recogiendo material para los proyectos a desarrollar. ● En la tercera sesión, antes de comenzar con las sesiones de taller, concretaremos el diseño a realizar. Es el momento de aclarar las dudas de las ideas y diseños planteados.	

- La cuarta y quinta sesión serán sesiones de taller. Los alumnos tendrán que realizar la instalación, elaborar el prototipo de automatización, y realizar un diario del proyecto.

- Por último, en la sexta sesión se realizará la presentación final de los proyectos donde veremos el acabado y funcionamiento de la instalación y los ventiladores.

Metodología (PBL)

El PBL o Aprendizaje Basado en Problemas permite que los docentes efectúen el seguimiento de un proceso creativo y de las tomas de decisiones que efectúa el alumnado hasta desarrollar el proyecto. Esta actividad está planteada para que el problema sea el punto de partida que estimule al alumnado a diseñar y automatizar una instalación.
Es una forma completa de aprendizaje donde los alumnos con el trabajo en grupo, enriquecen sus conocimientos y participan activamente en todo el proceso productivo.

Evaluación

Evidencias recogidas:
a) PROTOTIPO
b) DIARIO/MEMORIA .PDF
c) PITCH: Exposición oral .PPT

Tipo de evaluación
Coevaluación (entre alumnado)
Heteroevaluación (por parte del docente)

Estrategia utilizada
Observación de les aportaciones
Criterial

Retroacción
Inmediata o a través del *Moodle* (coevaluación)
La semana siguiente a la sesión de entrega (heteroevaluación)

ANÁLISIS Y MEJORA DEL PROYECTO PBL
Se solicitará la opinión del alumnado a lo largo del proyecto y se observará su evolución.
Al finalizar se realizarán encuestas.

- Sesión 1 - Presentación del proyecto (2h)
  - Convocatoria a los alumnos participantes
  - Los profesores presentan el proyecto
  - Formación de los grupos de trabajo
  - Trabajo fuera de clase para los grupos
- Sesión 2 – Selección y compra de los elementos y material necesario(3h)
  - Charla sobre el consumo abusivo de energía por los equipos de refrigeración y aire acondicionado, y de sus emisiones de CO_2

    - Los grupos recogerán material para sus proyectos
    - Trabajo fuera de clase para los grupos
- Sesión 3 - Concreción del Diseño (2h)
    - Por grupos, trabajan en parejas en el diseño de su especialidad
    - Al final de la clase se compartirá rápidamente la idea final
    - El resto de grupos realizarán las aportaciones o críticas constructivas que consideren para poder revisar la idea inicial del proyecto.
    - Trabajo fuera de clase para los grupos
- Sesión 4 - Taller 1 (11h)
    - CFGM Instal·lacions elèctriques i automàtiques
        - OBJETIVO: Montaje y testeo de los circuitos electrónicos en la placa de pruebas o 'Protoboard'.
    - Elaboración del prototipo en el taller sobre una maqueta a escala.
    - Realización de un diario del proyecto
    - Trabajo fuera de clase para los grupos
- Sesión 5 - Taller 2 (11h)
    - CFGM Instal·lacions elèctriques i automàtiques
    - OBJETIVOS: Montaje de la maqueta a escala, instalación de la electrónica preparada en la sesión anterior, programación del micro controlador 'Arduino', prueba del dispositivo y modificaciones del sistema.
- Sesión 6 - Presentación final proyectos (3h)
    - Realizar la presentación final de los proyectos.

RESUMEN EVALUACIÓN POR MÓDULOS

MÓDULOS	UF	PESO UF	Elemento	PESO Rúbrica	Entregables	Horas PBL	Horas restantes a evaluar
MP9 del CFGM de Instalaciones eléctricas y automáticas	UF1	30,77 %	Rúbrica	100%	(PDF Memoria con documentación técnica, cálculos…)	16	10
	UF2	20%				16	24

A continuación se detallan los Resultados de aprendizaje vinculados al PBL y los que no lo están(marcados con una cruz en rojo):

MF	UF	RA	
MP 9: Electrònica	*UF1: Electrònica Digital*	*RA1. Reconeix circuits lògics combinacionals determinant-ne les característiques i aplicacions.*	✓
		RA2. Reconeix circuits lògics seqüencials determinant-ne les característiques i aplicacions.	✓
	UF2: Electrònica	*RA1. Reconeix circuits de rectificació i filtratge determinant-ne les característiques i aplicacions.*	✓

	Analògica	*RA2. Reconeix fonts d'alimentació determinant-ne les característiques i aplicacions.*	✓
		RA3. Reconeix circuits amplificadors determinant-ne les característiques i aplicacions.	x
		RA4. Reconeix sistemes electrònics de potència verificant-ne les característiques i funcionament.	x
		RA5. Reconeix circuits de temporització i oscil·lació verificant-ne les característiques i el funcionament.	✓

Cómo se evaluarán las horas restantes de cada Módulo (fuera del PBL):

MP	UF	Horas pendientes	PESO Apartado	Enunciado	Elementos	Entregables
MP9 del CFGM de Instalaciones eléctricas y automáticas	UF 1	10	69,23%	Temario teórico electrónica digital. Ejercicios teóricos y prueba escrita.	Rúbrica Rúbrica	Prueba escrita Colección ejercicios resueltos
	UF 2	24	80%	Temario teórico electrónica analógico. Ejercicios teóricos y prueba escrita.	Rúbrica Rúbrica	Prueba escrita Colección ejercicios resueltos

ENTREGAS Y EVALUACIÓN

Evidencia	Participa	Tipo	Agente	Instrumento	Formato	Fecha entrega
Proyecto	Grupo	Sumativa	Profesor	Rúbrica	Aula (feedback profesor)	Sesión 6
Diario	Grupo	Sumativa	Profesor	Rúbrica	Moodle (.pdf)	5 entregas: el apartado a) Memoria de la sesión se debe entregar antes de la siguiente sesión). Entrega final Dietario completo: sesión 6.
Exposición oral	Grupo	Sumativa	Coevaluación	Rúbrica	Moodle (.pdf)	Sesión 6

1. <u>Los distintos apartados que tendría que tener el trabajo.</u>

CFGM	MP	UF	APARTADOS	RA
CFGM Instal·lacions elèctriques i automàtiques.	*MP 9: Electrònica*	*UF1: Electrònica Digital*	**NF1:** Definir funcionamiento del equipo. Definir elementos del circuito.(3h) **NF2:** Montaje del circuito sobre 'protoboard'. Montaje mecánico de la hucha o ScrOoge(5h) **NF3:** Programación del sistema en 'Arduino' Test de funcionamiento del sistema. Modificaciones y mejoras del mismo.(8h)	**RA1** **RA2**
		UF 2: Electrónica Analògica		**RA1,RA2, RA5**

2. <u>Cómo se evaluarían estos apartados (y ejemplo de alguna actividad). 1 sesión del CFGM en el proyecto PBL:</u>

CFGM de Instalaciones Eléctricas y Automáticas			
NF	**Descripción apartados**	**RA**	**C.A**
NF1	NF1: Definir funcionamiento del equipo. Definir elementos del circuito.(3h)	UF1: Electrónica digital RA1, RA2	UF1: Electrónica digital CA1.1 al CA1.9 CA2.1 al CA2.8 UF2: Electrónica analógica CA1.1 al CA1.9 CA2.1 al CA2.9 CA5.1 al CA5.10*
NF2	NF2: Montaje del circuito sobre 'protoboard'. Montaje mecánico del prototipo .(5h)		UF1: Electrónica digital CA1.1 al CA1.9 CA2.1 al CA2.8 UF2: Electrónica analógica CA1.1 al CA1.9 CA2.1 al CA2.9 CA5.1 al CA5.10*
NF3	NF3: Programación del sistema en 'Arduino' Test de funcionamiento del sistema.	UF2: Electrónica analógica RA1, RA2, RA5	UF2: Electrónica analógica CA1.1 al CA1.9 CA2.1 al CA2.9

	Modificaciones y mejoras del mismo.(8h)		CA5.1 al CA5.10*

*Se añaden 1 C.A:.5.10 Hace la presentación del proyecto

Ejemplo actividad CFGM *de Instalaciones Eléctricas y Automatismos:*

ACTIVIDAD *D'ENSENYAMENT-APRENENTATGE*			**JUSTIFICACIÓN EN RELACIÓN AL PERFIL PROFESSIONAL**
UF 2: Electrónica analógica			**NF3: Programación del sistema en 'Arduino'** **Test de funcionamiento del sistema.** **Modificaciones y mejoras del mismo.**
A2 Puesta en marcha del sistema electrónico de control de la hucha electrónica.			
RA	**Contenidos**	**CA**	**Capacidades Clave**
RA1. Reconeix circuits de rectificació i filtratge determinant-ne les característiques i aplicacions. *RA2. Reconeix fonts d'alimentació determinant-ne les característiques i aplicacions.* *RA5. Reconeix circuits de temporització i oscil·lació verificant-ne les característiques i el funcionament.*	1.1 a 1.8 2.1 a 2.4 3.1 a 3.4 4.1 a 4.2 5.1 a 5.2	1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.1	Autonomía, innovación, organización del trabajo, relación interpersonal, trabajo en equipo, resolución de problemas.

		2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 5.1 5.2 5.3 5.4 5.6 5.7 5.8 5.9 5.10	

DESCRIPCIÓN DE LA ACTIVIDAD

Desarrollo	Duración	Recursos y TIC	Espacio	Actividades de evaluación
Gamificación con Kahoot (se divide la clase en los equipo ABP formados). Se responde un cuestionario sobre proyectos vinculado con la economía circular.	15 min	Proyector y un ordenador por alumno.	Aula -taller	Kahoot (resultados y puntuación). (In1)
Programación del sistema en 'Arduino' **Test de funcionamiento del sistema.**	2h 15 min	Taller y un ordenador por alumno. (Herramientas y material).	Aula -taller	Rúbrica de evaluación
Modificaciones y mejoras del mismo.	30 min	Taller y un ordenador por alumno. (Herramientas y	Aula -taller	Rúbrica de evaluación

		material).		
AVALUACIÓ DE L'ACTIVITAT				
Instruments d'avaluació	**RA i CA**		**Criteris de correcció**	
Resultados Kahoot	CA 0.1		Puntuación (Los 3 grupos con más aciertos obtienen 1 punto extra en la nota final y los demás 0,25).	
Diario del taller	RA1, RA2, RA3 y RA5 (ver tabla anterior para consultar CA)		Rúbrica del diario	
Entrega y prototipo final	RA1, RA2, RA3 y RA5 (ver tabla anterior)		Rúbrica de evaluación final	

3. Cómo se evaluarían finalmente cada uno de los módulos

MÓDULOS	UF	Apartados	PESO Apartado	Elemento	PESO Rúbrica	Entregables	Horas ABP	Horas restantes a evaluar
MP9 del CFGM de Instalaciones eléctricas y automáticas	UF1	NF1 NF2 NF3	30,77%	Rúbrica	100%	(PDF Memoria con documentación técnica, cálculos…)	16	10
	UF2		20%				16	24

ii7.1 Cómo se evaluarán las horas restantes de cada Módulo

MÓDULOS	UF	Horas pendientes	PESO Apartado	Enunciado	Elementos	PESO Rúbrica	Entregables
MP9 del CFGM de Instalaciones eléctricas y automáticas	UF1	10	69,23%	Temario teórico electrónica digital. Ejercicios teóricos y prueba escrita.	Rúbrica Rúbrica	50% 50%	Prueba escrita Colección ejercicios resueltos
	UF2	24	80%	Temario teórico electrónica analógico. Ejercicios teóricos y prueba escrita.	Rúbrica Rúbrica	50% 50%	Prueba escrita Colección ejercicios resueltos

- Cada apartado tiene una importancia en función de los RA, los contenidos que se entregan y las horas dedicadas.
- La evaluación forma parte de las UF previamente indicadas. La puntuación va del 0 al 10. Se aprueba con un mínimo de 5.
- La participación y la asistencia al tratarse de una metodología ABP y un módulo presencial son esenciales para una evaluación contínua.
- El alumnado debe asistir como mínimo al 80% de las horas lectivas dedicadas al ABP y al 70% de las horas respectivas de cada UF. En caso contrario deberá ir a recuperación.
- Antes de iniciar el módulo ABP se concretará cómo acceder a la segunda convocatoria (si es a partir de un trabajo individual o con un examen).
- La nota final que se obtiene de un Módulo Formativo se calcula haciendo la media teniendo en cuenta:
  > El número de horas de cada NF de la UF. Las notas de las UF tienen que ser superiores a 5; siendo así el alumno habrá aprobado todo el Módulo.
  > Los criterios de evaluación van en acorde a si se trabajan conocimientos prácticos y teóricos, y atendiendo a las competencias y a los RA que se tienen que completar.
  > La Cualificación de los Módulos y de sus UF se calculará a partir de las notas de los Resultados de Aprendizaje y su ponderación tal como se especifica en el siguiente apartado.

Prototipo y lecciones aprendidas

En el diseño y el montaje del primer prototipo nos hemos encontrado con ciertas dificultades, habituales la primera vez que se lleva a cabo un proyecto de esta complejidad.

Desarrollar la maqueta previamente, antes de encargar la actividad a los alumnos nos ayuda a prever los posibles fallos y problemas que nuestros alumnos podrían tener. Una vez identificados estos, podemos plantear un proyecto PBL con la experiencia suficiente para que no suceda algo parecido cuando lo realicen los alumnos del ciclo de grado medio de instalaciones eléctricas.

Un factor clave que ha afectado a la temporización ha sido la disponibilidad de espacios y de herramientas. Este es un problema que, conociendo qué se necesita con antelación en cada sesión, se puede evitar con facilidad en el contexto del aula.

Uno de los aspectos más importantes en la planificación de un proyecto es la comprobación previa del correcto funcionamiento de cada parte crítica del sistema. Existe una tendencia a omitir esta clase de comprobaciones por falta de tiempo. Pero es precisamente en esta circunstancia cuando más necesarios son y más problemático es no hacerlas.

En definitiva, el aprendizaje específico que hemos obtenido con este proyecto ha sido este: la necesidad de elegir los componentes apropiados antes de continuar con el siguiente paso, la importancia de testear cada parte de forma independiente y una buena planificación en cuanto al tiempo y las herramientas que se necesitan en cada sesión. Además, a nivel docente hemos podido ver como, aún planeando con tiempo lo que haríamos y lo que necesitaríamos, surgían contratiempos inesperados. Un proyecto de este tipo se tiene que realizar antes para experimentar estas dificultades y poderlas prever a tiempo cuando se lleve a cabo con el alumnado.

Conclusiones

Con este trabajo hemos podido poner a prueba nuestra capacidad creativa para integrar contenidos del Módulo de electrónica del CFGM de instalaciones eléctricas y automáticas. También hemos desarrollado el rol de flexibilizar y aplicar nuevas formas de aprendizaje al aula, además de modificar la metodología para poder prever la integración de alumnado en una actitud proactiva y creativa. La necesidad de integrar información, tecnología y comunicación en procesos de innovación pedagógica es clave para dotar de competencias al alumnado.

En una primera fase, concretamos la necesidad o el problema desde donde partir y pudimos pensar en proyectos que fomentaran el respeto al medioambiente y que, de una forma didáctica, fuesen útiles para el entorno. Decidimos fijarnos el objetivo de trabajar con materiales baratos y darles un uso tecnológico y social, para concienciarnos sobre nuestra responsabilidad docente y de centro con el alumnado, y para hacer el proyecto accesible a todo el conjunto de alumnos en general, cuál fuese su situación económica.

Aprovechamos el feedback recibido en las primeras fases de explicación de la idea para acabar de sustentar la base de nuestro Reto PBL. Se nos aconsejó que éste estuviese íntegramente relacionado al temario, criterios de evaluación, Resultados de aprendizaje y contenidos de un módulo del ciclo a trabajar, en cuanto no existe en el currículum oficial de

ningún ciclo formativo la competencia alguna de aprender a programar en ‘Arduino’. Con este fin se escogió el Módulo de electrónica, para su aplicación y demostración. No existe, sin embargo, limitación para usar el planteamiento de este TFM y del manual que lo acompaña en otros ciclo formativo que no sea el ya citado de Instalaciones eléctricas y Automáticas, o bien, en módulos incluso del mismo, como el de electrotecnia, para desarrollar sensores magnéticos, por ejemplo…

A medida que hemos avanzado hemos comprobado como la temporización a nivel experimental nos ha llevado mucho tiempo de preparación, diseño y comprobación mecánica, aún y tenerlo todo calendarizado y con un retrotiming de previsión hecho. Cabe destacar que siendo un proceso de desarrollo desde cero de un proyecto de innovación el rol y distribución de las tareas tiene muchas horas asignadas para poder dar respuesta a todas las necesidades. Todo eso ha implicado un número elevado de horas personales.

El desarrollo y construcción de la maqueta a escala nos ha permitido crear una temporización y una ficha del estudiante que estuviera en acorde con los cánones de enseñanza y aprendizaje. Teniendo en cuenta que tras nuestro diseño y experimentación podríamos crear una programación adecuada para el alumno, que ya partiría de un enunciado y unas instrucciones para desarrollar su propia maqueta.

Como docentes en activo en la actualidad entendemos la necesidad de aplicar metodologías de innovación en el aula. El efecto y las posibilidades de crear proyectos interciclo tiene un potencial elevado ya que implica la necesidad de adaptación y trabajo en equipo. El modelo de aprendizaje trabajando competencias multidisciplinares hace que el alumno se acostumbre al mundo laboral, donde debe estar preparado para un entorno cambiante y dinámico. Y creo que ese debe ser el objetivo final de los retos PBL, y por lo tanto, de este manual.

A nivel personal, hemos experimentado cómo la innovación aporta procesos de creación metodológica variados, nos ha permitido explorar la integración de conceptos y contenidos multidisciplinares y ha conseguido probar nuestra capacidad y competencias, como futuros docentes.

ANEXO1: Materiales

Material necesario para desarrollar la maqueta del simulador:

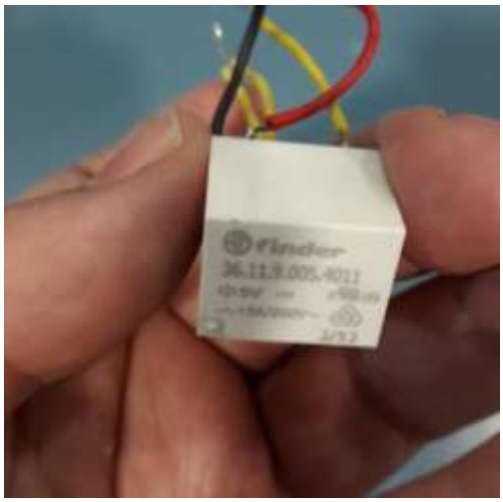
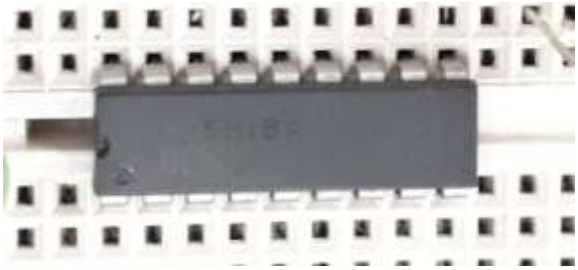
1.	**Plancha DM 3mm** **30x60 cm** Plancha DM de 3 mm de Grosor	
2.	**Plancha madera Tilo 1,5 mm** **25x100cm Paq.2Unid**	
3.	**Listón madera de pino, de 2,4 m** **9x9mm,**	
4.	**Tornillo avellanado TORX para madera, a 3mm diámetro, 13mm de largo. 100 tornillos**	

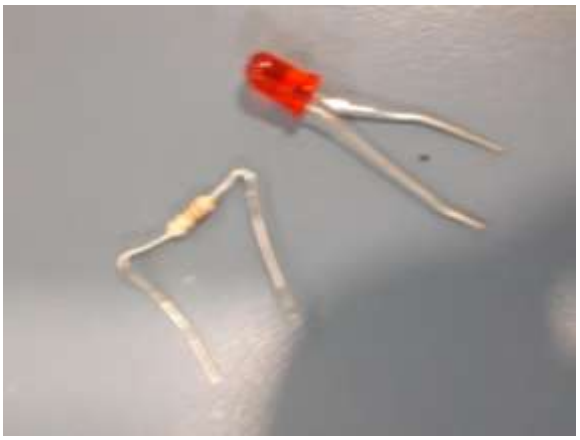
En cuanto a los elementos eléctricos y electrónicos, el centro también intentará reducir el coste de inversión de los alumnos, proporcionando herramientas, y algunos de los dispositivos, los más caros, como la placa Arduino; otros, es posible que haya que comprarlos. El cualquier caso el presupuesto no superará los 30€, y este dependerá de las elecciones de cada grupo; tanto por el diseño inicial como por el trabajo de búsqueda y desarrollo que hagan..

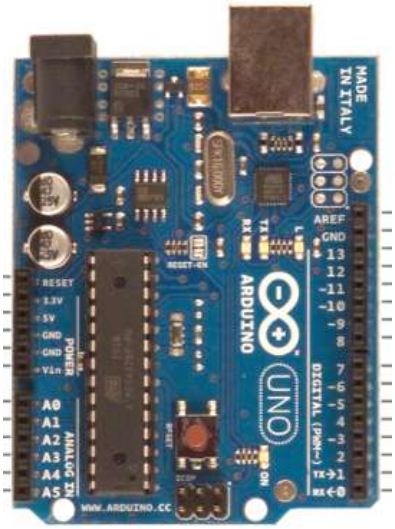
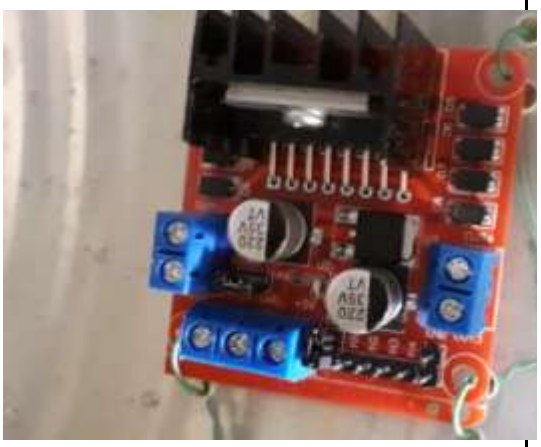
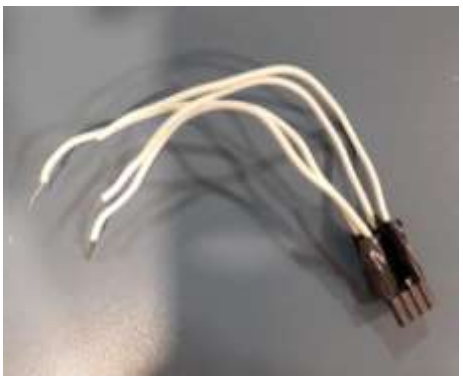
En la siguiente tabla se hace una relación de estos materiales, y se añade el modelo concreto utilizado en el montaje del modelo a escala controlado por ‘Arduino’.

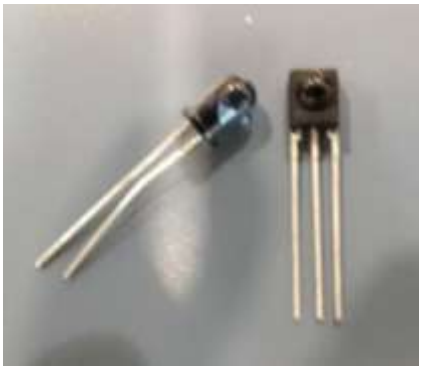
5.	**Fuente alimentación CEBEK 0 a +5VDC, 3 Amperios.** La usaremos para alimentar el circuito de los leds emisores IR a 38 Khz.	
6.	**Fuente de alimentación TALUS, configurada a +12VDC.** Esta Fuente de alimentación dispone de una salida de alimentación exterior de USB, con la que alimentaremos ‘Arduino’.	
7.	**Cable USB-Mini-USB** alimentación ‘Arduino’, Cable conexión ‘Arduino’-PC programación.	

8.	**USB 4Port HUB**	
9.	**Cable General de alimentación monofásica**	
10.	**Bornes de conexión.**	

11.	**Pasa muros. Protege los cables.**	
12.	**Relé finder bobina a 5VDC, 1 contacto conmutado 240 Vac, 10A corriente máxima.**	
13.	**National ULN2805A.** 8-channel Darlington amplifier.	
14.	**Porta-lámparas roscado E-14, paso 10x1, Blanco**	

15.	**Bombilla LED E14**	
16.	**Cables de electrónica de 0.5 mm2**	
	Resistencias 1/4w varias. **Leds de señalización.**	
18.	**Cinta aislante**	

19.	**Módulo PcB programable ‘’Arduino’ UNO A000066**	
20.	**LM298. Controlador de 2 canales para motores DC y motores Paso a Paso.**	
21.	**Motor 12VDC 3.4 Amperios. Magnético-reversible.**	
22.	**Cable conexión ‘Arduino’ a L-298.**	

23.	A la izquierda el **emisor IR** utilizado a 38 Khz. A la derecha **receptor IR**.	
24.	**Células Fotoconductoras 4-10 100MW 150VDC GL7516**	
25.	**Transistor PNP BC548**	
26.	**Potenciómetros.**	

A continuación las referencias de los componentes utilizados incluyendo la referencia, el precio y su procedencia:

Nº Ítem	**Descripción**	**Referencia**	**Precio(Euros)**	**Procedencia**
1	Transistor ROHS NPN. UNI 30V 0'1A 0'5W 300MHZ. TO-92	**Código: BC548B**	0,18	onda Radio
2	ROHS 8-CHANNEL HIGH-CURRENT DARLINGTON ARRAYS DIP-18	**Código: ULN2804**	1	onda Radio

3	Relé finder 5VDC	3611.9005	2.87	Diotronic
4	Emisor IR	LD271	0.37	Diotronic
5	Receptor IR	TSOP2438	2.06	Diotronic
6	Arduino UNO	A000066	18,73	onda Radio
7	Control Driver LM298		2,59	amazon.es
8	Piezo Speaker 8 ohmios	**Código: HPE200H**	1,8	onda Radio
9	ROHS POT 2'5 22K PT-10LH01-223A2020-S AJUSTE HORIZONTAL	**Código: PT10H22K**	0,61	onda Radio
10	Condensador electrolítico 10 microfaradios	**Código: 10450PHT** EAN: 5412810204977	1	onda Radio
11	Condensadores cerámicos 0.01 microfaradios		0,30	onda Radio
12	Condensadores cerámicos 0.001 microfaradios		0,30	onda Radio
13	Resistencias varios 1/4W		5	onda Radio
14	Fuente Alimentación CEBEK FE-3	EAN: 8436544281841	20,31	Onda Radio
15	ROHS TRAFO 220V-9V 0'5A	Código: 10014	7,07	Onda Radio
16	Porta-lámparas roscado E-14, paso 10x1, Blanco	159768	2,90	Servei Estació
17	Led esférica E14 de 5W 4000K LEXMAN 220V	3276000389989	4,98	Leroy Merlin
18	ROHS BLS/10	CC1072V	1,43	Onda Radio

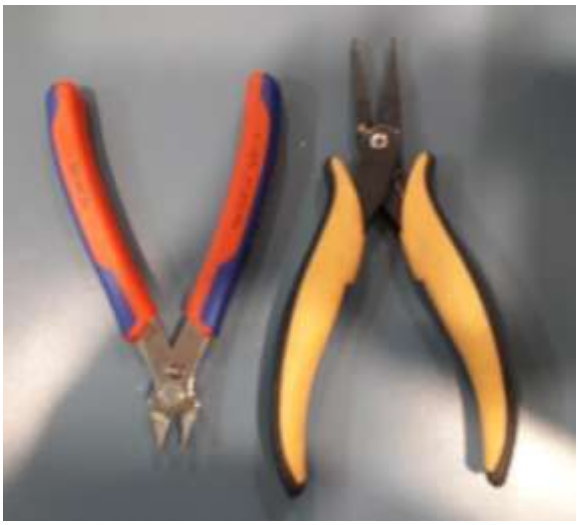
	Cable Conexión 0,07			
19	ROHS BLS/10 Cable Conexión 0,14	CC1142N	1,58	Onda Radio
20	ROHS BLS/10 Cable Conexión 0,14	CC1142R	1,65	Onda Radio
21	Células Fotoconductoras 4-10 100MW 150VDC	GL7516	1,20	Onda Radio
22.	USB 4PortsHUB	021211	6,49	Tienda Cables

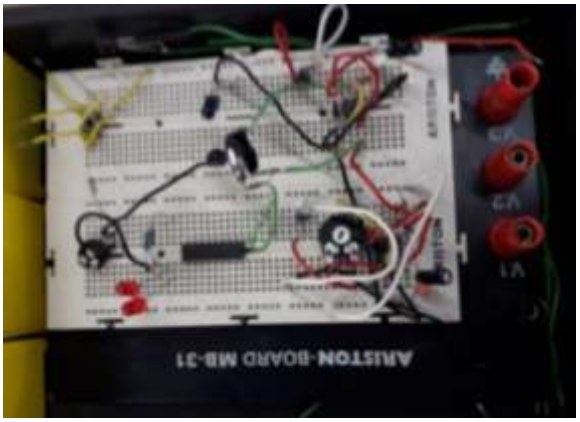
A continuación las referencias de los materiales utilizados en la construcción de la maqueta a escala:

Nº Ítem	**Descripción**	**Referencia**	**Precio(Euros)**	**Procedencia**
1b	**Plancha DM 3mm** **30x60 cm** Plancha DM de 3 mm de Grosor		1 euro	Servei Estació
2b	**Plancha madera Tilo 1,5 mm** **25x100cm Paq.2Unid**	166627	15,50	Servei Estació
3b	**Listón madera de pino, de 2,4 m** **9x9mm,**	**Ref.17371445**	1,69	Bricomart
4b	**Tornillo avellanado TORX para madera, a 3mm diámetro, 13mm**		1,59	Bricomart

	de largo. 40 tornillos a granel.			
5b	Transistor ROHS NPN. UNI 30V 0'1A 0'5W 300MHZ. TO-92	**Código: BC548B**	0,18	onda Radio

En la siguiente tabla se hace una relación de las **herramientas** básicas necesarias para el montaje del sistema ‘Arduino’ en el ‘protoboard’:

1.	**Juego de destornilladores** a)Destornillador fino de precisión para ajustar los potenciómetros. b) Destornillador fino plano para bornes. c) Juego de destornilladores de electricista.	
2.	● **Alicates de corte.** ● **Alicates de electrónica.**	

3.	**Soldador estaño JBC 30S**	
4.	**Estaño Ariston para soldadura.**	
5.	**Protoboard Ariston MB-31**	
6.	**Multímetro. MASTECH my-62**	

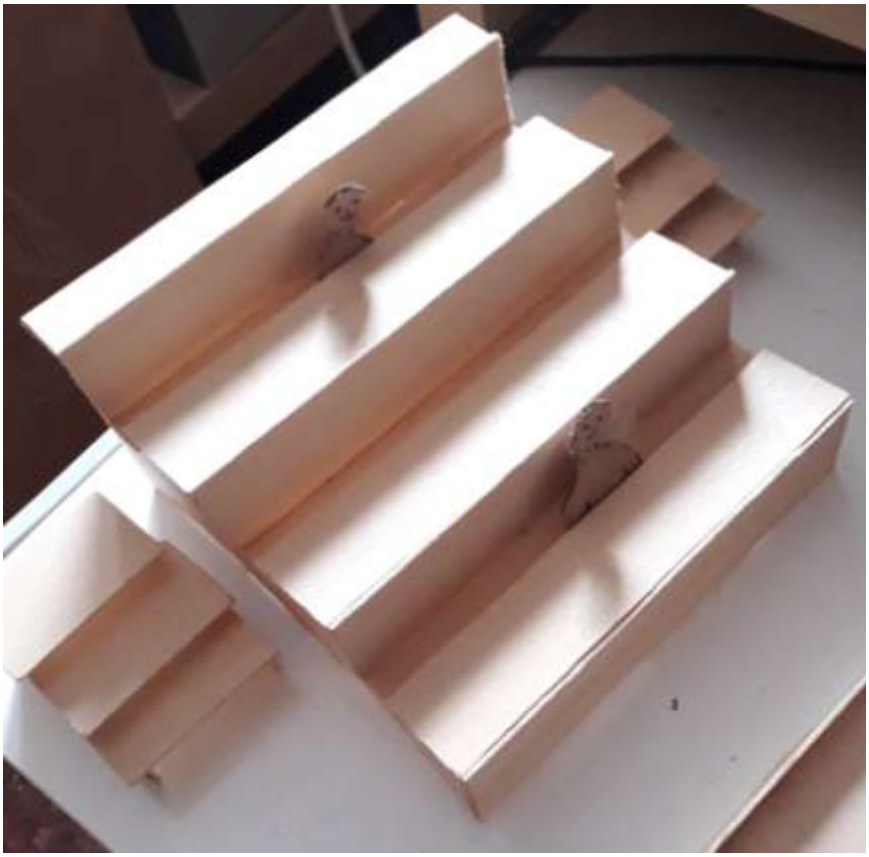
7.	**Fuente de alimentación Blausonic EP-613A** ● Fuente de alimentación de laboratorio ● Salida fija 0 a +5VDC, +15VDC, -15VDC, ● Salida regulable 0 a 24VDC, a 3 Amperios.	

En la tabla se hace una relación de las **herramientas** básicas necesarias para el montaje de la maqueta a escala de la Sala de Actos:

1.	**Sierra de arco con hoja**	
2.	**cutter**	
3.	**Cutter precisión de marqueteria**	
4.	**Martillo**	

ANEXO2: Ficha del Alumno

Guión de trabajo para el alumnado y material complementario para docentes



FICHA DEL ALUMNO

METODOLOGÍA PBL

SOSTENIBILIDAD
TECNOLOGÍA
CREATIVIDAD

Trabaja el desarrollo de entornos medioambientalmente sostenibles

¿Aceptas el reto?

CONTACTO PROFESORADO:
bernat.corbera@est.fib.upc.edu
bernatcorb@gmail.com

AUTOMATIZACIÓN DE LA SALA DE ACTOS DE UN CENTRO EDUCATIVO

PROYECTO DE INNOVACIÓN

A lo largo de 6 sesiones desarrollaremos una actividad como reto PBL, con el fin de crear un proyecto sostenible y tecnológico. En este documento quedan definidos la parte del control electrónico de la instalación, así como sus criterios de desarrollo.

Información para el alumnado

1A SESIÓN:

PRESENTACIÓN DEL PROYECTO INTERCICLOS (2H)

- **CFGM *Instal·lacions elèctriques i automàtiques***
  - **MP 9: Electrònica**
  - **UF1: Electrònica Digital, UF2: Electrònica Analògica**

OBJETIVO: Diseñar y crear un proyecto tecnológico. En concreto, los alumnos del centro desarrollarán la maqueta a escala, controlada por un micro controlador 'Arduino', que hará a la vez la gestión de luces y de puesta en marcha de ventiladores en una teórica Sala de Actos de un centro educativo.

.

01 PRESENTACIÓN DEL PROYECTO

Actividad: Los profesores de todos los ciclos presentan el proyecto a los alumnos de los ciclos participantes (90 min)

- **Explicación metodología de trabajo PBL**
- **Sistema de evaluación**
- **Objetivos del proyecto en común**
- **Dinámica de grupos para romper el hielo (*warm-up*)**

02 Formación grupos

Actividad: Formación de los grupos de trabajo. Explicación Test de Belbin (30 min)

- **Test y dinámica para identificar el perfil de cada alumno: deberéis contestar el Test de Belbin: https://www.belbin.es/test-belbin-gratis/**
- **Participaran preferiblemente 2 personas por ciclo en cada grupo (6 total)**
- **Se forman grupos heterogéneos (teniendo en cuenta el perfil del Test de Belbin)**

ANTES DE LA SESIÓN 2….

Trabajo **fuera de clase** para los grupos:

- **Idea: elección de temática del Proyecto (incluir en el Diario del proyecto)***
- **Borrador de la idea general (incluir en el Diario del proyecto)***
- **Listado materiales (incluir en el Diario del proyecto)***

* Las instrucciones para la entrega del Dietario están especificadas en el apartado de Instrucciones de entregas al final de esta ficha.

2A SESIÓN: DEBATE SOBRE LAS EMISIONES DE CO2 A LA ATMOSFERA (3H)

OBJETIVO: Exponer en grupo los daños que producen a nuestro planeta las emisiones de CO2 y de gases fluorados producidos por el uso abusivo de equipos de aire acondicionado.

01 CHARLA SOBRE MEDIOAMBIENTE Y SOSTENIBILIDAD.

Actividad: Debate sobre la Gestión del CO2 generado por los equipos de climatización en edificios e infraestructuras públicas, así como en grandes edificios de oficinas.

https://noticias.eltiempo.es/aire-acondicionado-y-contaminacion-como-esta-aumentando/
http://www.ecoavant.com/es/notices/2018/07/el-aire-acondicionado-calienta-cada-vez-mas-el-planeta-3962.php

- **Tarea en grupo: Comparación de los consumos de electricidad generados, según facturas del centro o dispositivos de medida y monitorización, cuando se disponga de ellos. El análisis se llevará a cabo con los datos del último año, y en relación a los meses en que los equipos a substituir permanecen apagados, y aquellos en que inevitablemente se ponen en marcha todos los días.**

02 Recogida material

Actividad: Los grupos recogerán material para sus trabajos (120 min)

- **Los grupos recogerán material para sus proyectos en función del diseño previo que hayan ideado.**
- **Realizar la lista del material adquirido, incluyendo foto, descripción, el precio estimado si lo hubierais comprado y la función que tendrá en vuestro proyecto.**

CONSEJO…
Es importante recoger material extra por si en las sesiones de taller surgen imprevistos. Hace falta pensar en las posibilidades mecánicas del material, en relación con la parte de diseño y de programación posterior.

ANTES DE LA SESIÓN 3…

Trabajo **fuera de clase** para los grupos:

- **Ajuste de la idea según materiales disponibles**
- **Análisis de material según la tecnología a usar**
- **Elaboración del diseño del proyecto y bocetos para el prototipo**

3A SESIÓN: CONCRECIÓN DEL DISEÑO (2H)

OBJETIVO: Con la asistencia del profesorado en el aula concretar el diseño a realizar antes de las sesiones de taller y exponerlo al resto de compañeros.

- **Por grupos: trabajar en parejas en el diseño:**

CFGM *Instal·lacions elèctriques i automàtiques*

- Para concretar el diseño de la parte electrónica, los alumnos utilizarán la herramienta del orCAD PsPice. Un simulador de circuitos electrónicos del desarrollador CADEnCE, y gratuito para estudiantes.
  Cuando este diseño les funcione en el software de simulación, a partir de los cálculos, pasarán a representarlo sobre una placa 'protoboard', sobre la que ensamblarán todos y cada uno de los dispositivos del mismo.
  Estas tres fases creativas se incluyen en el Dietario (consultar Instrucciones entregas al final de este documento).

Pitch Elevator: Por indicación del profesorado al final de esta sesión cada grupo PBL intentará "vender" al resto de compañeros su idea de proyecto en 1 minuto.

- **Feedback en el aula:** el resto de grupos realizará las aportaciones o críticas constructivas que consideren oportunas.

ANTES DE LA SESIÓN 4…

Y para los alumnos de *Instalaciones elèctriques:*

- **Simulación de los esquemas eléctricos en orCAD:**
  https://www.orcad.com/buy/orcad-licensing-options
  Las versiones para estudiantes son gratuitas, aunque limitadas en recursos.

4A SESIÓN: TALLER I (11H)

- **CFGM Instal·lacions elèctriques i automàtiques**
  - OBJETIVO: Montaje y testeo de los circuitos electrónicos en la placa de pruebas.

01 ELABORACIÓN DEL PROTOTIPO EN EL TALLER

Los alumnos trabajarán en grupos de 4 los conocimientos adquiridos en el ciclo de electricidad durante la sesión de montaje y testeo de los circuitos eléctricos y electrónicos y su alimentación.

Instrucciones

1) Hacer el conexionado de los componentes electrónicos tal como se diseñó previamente y se comprobará mediante simulaciones. Para esto se utilizará la placa de pruebas o protoboard, y se montará en una maqueta.
2) Comprobar el correcto funcionamiento de cada parte individual:
   - Fuente de alimentación
   - Sistema de detección de monedas
   - Control del motor para el movimiento del brazo
   - Señales acústicas y lumínicas
3) Una vez testeado de forma satisfactoria se ensamblarán de forma conjunta, formando el conjunto del automatismo.
4) Comprobar el correcto funcionamiento de todo el sistema de forma que quede listo para ser montado en el aula, en la segunda sesión de taller.

RECUERDA...
Es importante repartir las tareas de forma eficiente para aprovechar el tiempo disponible en el taller.

02 REALIZACIÓN DE UN DIARIO DEL PROYECTO

Instrucciones

Avanzar en la redacción del Diario.

ANTES DE LA SESIÓN 5…

- **Avanzar el resto de tareas posibles (prever necesidades: materiales, etc) para la siguiente sesión de taller.**

5A SESIÓN: TALLER II (11H)

- **CFGM Instal·lacions elèctriques i automàtiques**
  - OBJETIVOS: Montaje de la maqueta a escala, instalación de la electrónica preparada en la sesión anterior, programación del micro controlador 'Arduino', prueba del dispositivo y modificaciones del sistema.

01 MONTAJE DE LA MAQUETA CON LOS DISPOSITIVOS ELECTRÓNICOS DEFINIDOS PREVIAMENTE. PROGRAMACIÓN DEL SISTEMA.

En esta última sesión de taller se tendrá que cablear y conectar todos los dispositivos y sensores del sistema y programar el micro controlador 'Arduino'.

Instrucciones

1) Preparar el material que formará el sistema. Utilizar las herramientas apropiadas para cortar, doblar o unir.
2) Hacer el montaje de todos los componentes eléctricos y electrónicos.

3)Programar el mico controlador 'Arduino'conforme a condiciones de funcionamiento pre-definidas.
4) Comprobar el funcionamiento de la Instalación y todas sus funciones diseñadas.
5)Añadir modificaciones y mejoras al sistema.
6) Añadir los acabados y detalles estéticos.

04 CONTINUACIÓN DEL DIARIO DEL PROYECTO

PARA LA SESIÓN 6…

Trabajo **fuera de clase** para los grupos:

- **Publicación del diario del proyecto en el moodle del centro.**
- **Preparación de la presentación del proyecto con todos los miembros del proyecto: Pitch elevator de 5 minutos.**

6A SESIÓN:

PRESENTACIÓN FINAL PROYECTOS (3H)

OBJETIVO: Realizar la presentación final de los proyectos. Se dispondrá de 5 min por grupo + la exposición de la maqueta prototipo.

01 PRESENTACIÓN POR PARTE DE LOS ALUMNOS DE LOS PROYECTOS ELABORADOS

INSTRUCCIONES ENTREGAS:

a) ENTREGA COMÚN:

DIETARIO + PROYECTO Y PITCH ELEVATOR

DIETARIO: ¿Qué apartados debe incluir el diario del proyecto? (Extensión 15-20 pág + Anexos).

a) Memoria de cada sesión: detallar qué se ha hecho a cada sesión tanto a nivel PBL como a nivel de cada especialidad.*
b) Idea: explicar el proceso de elección de la idea (*brainstorming).*
c) Material necesario: listado de materiales elegidos.
d) Implementación de la idea y diseño: desarrollo del diseño.
e) Prototipo y esbozos: acotaciones de las piezas utilizadas.
f) Evaluación: análisis de los resultados.
g) Aprendizaje: reflexión sobre la utilidad pedagógica del trabajo.
h) Bibliografía.

Entrega dietario:

En la sesión 2, 3, 4 y 5 (se deben hacer entregas del apartado *a)* con la Memoria de cada sesión.
La entrega final es en la sesión 6.

PITCH ELEVATOR: ¿Qué apartados debe incluir el Pitch?

- El pitch elevator de debe extender un máximo de 5 minutos. Se puede utilizar como soporte un .PTT.

ANEXO3: Rúbricas de Evaluación según Criterios de Evaluación (CA)

CFGM de Instalaciones eléctricas y automáticas
M9: Electrónica
UF 1: Electrónica digital
UF2: Electrónica analógica
NF3
Rúbrica sobre el proyecto (en la página siguiente)

				INDICADORS	ASSOLIT EXCEL·LENT 3	ASSOLIT NOTABLE 2	ASSOLIT 1	NO ASSOLIT 0	PONDERACIÓ	PUNTUACIÓ	NOTA FINAL
UF 1 electrònica digital	1. Reconeix circuits lògics combinacionals determinant-ne les característiques i aplicacions.	33.33%	CAL1.1	1.1 Utilitza diferents sistemes de numeració i codis.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...			
			CAL1.2	1.2 Descriu les funcions lògiques fonamentals utilitzades en els circuits electrònics digitals.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...			
			CAL1.3	1.3 Representa els circuits lògics mitjançant la simbologia adequada.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...			
			CAL1.4	1.4 Interpreta les funcions combinacionals bàsiques.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...			
			CAL1.5	1.5 Identifica els components i blocs funcionals.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...			
			CAL1.6	1.6 Munta o simula circuits.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...			
			CAL1.7	1.7 Verifica el funcionament dels circuits.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...			
			CAL1.8	1.8 Identifica les diferents famílies d'integrats i la seva aplicació.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...			
			CAL1.9	1.9 Realitza les tasques que cal fer individualment amb autosuficiència i seguretat.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...			
	2. Reconeix circuits lògics seqüencials determinant-ne les característiques i aplicacions.	33.33%	CAL2.1	2.1 Descriu diferències entre circuits combinacionals i seqüencials.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...			
			CAL2.2	2.2 Descriu diferències entre sistemes síncrons i asíncrons.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...			
			CAL2.3	2.3 Identifica els components i blocs funcionals.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...			
			CAL2.4	2.4 Utilitza els instruments lògics de mesura adequats.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...			
			CAL2.5	2.5 Munta o simula circuits.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...			
			CAL2.6	2.6 Verifica el funcionament de circuits bàsics seqüencials.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...			
			CAL2.7	2.7 Descriu aplicacions bàsiques dels circuits amb elements lògics seqüencials.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...			
			CAL2.8	2.8 Realitza les tasques que cal fer individualment amb autosuficiència i seguretat.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...			

				INDICADORS	ASSOLIT EXCEL·LENT 3	ASSOLIT NOTABLE 2	ASSOLIT 1	NO ASSOLIT 0	PONDERACIÓ	PONDERACIÓ	NOTA FINAL
UT2. electrònica analògica	1. Reconeix circuits de rectificació i filtratge determinant-ne les característiques i aplicacions.	16,90%	CA1.1	1.1 Descriu els diferents components.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...	2		
			CA1.2	1.2 Descriu els paràmetres i magnituds que caracteritzen els circuits amb components passius.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...	2		
			CA1.3	1.3 Utilitza els instruments de mesura adients (multímetre i oscil·loscop, entre d'altres).	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...	2		
			CA1.4	1.4 Relaciona els components amb els símbols que apareixen als esquemes.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...	1		
			CA1.5	1.5 Descriu els tipus de rectificadors i filtres.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...	1		
			CA1.6	1.6 Munta o simula circuits.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...	1		
			CA1.7	1.7 Obté els paràmetres i característiques elèctriques dels components dels circuits.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...	1		
			CA1.8	1.8 Descriu les aplicacions reals d'aquest tipus de circuits.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...	1		
			CA1.9	1.9 Realitza les tasques que cal fer individualment amb autoavaluació i cogavaluació.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...	1		
	2. Reconeix fonts d'alimentació determinant-ne les característiques i aplicacions.	16,90%	CA2.1	2.1 Descriu les diferències entre fonts commutades i no commutades.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...	10		
			CA2.2	2.2 Descriu el funcionament dels diferents blocs que composen els sistemes complets d'alimentació.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...	3		
			CA2.3	2.3 Identifica les característiques més rellevants proporcionades pels fabricants.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...	2		
			CA2.4	2.4 Descriu les diferents configuracions de circuits reguladors integrats.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...	2		
			CA2.5	2.5 Munta o simula circuits.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...	0		
			CA2.6	2.6 Utilitza els instruments de mesura adients (multímetre i oscil·loscop, entre d'altres).	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...	1		
			CA2.7	2.7 Descriu aplicacions reals de les fonts commutades.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...	1		
			CA2.8	2.8 Verifica el funcionament de les fonts commutades.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...	1		
			CA2.9	2.9 Realitza les tasques que cal fer individualment amb autoavaluació i cogavaluació.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...	0		
	3. Reconeix circuits de temporització i oscil·lació verificant-ne les característiques i el funcionament.	16,90%	CA3.1	3.1 Reconeix els components dels circuits de temporització i oscil·lació amb dispositius integrats.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...	1		
			CA3.2	3.2 Descriu el funcionament de temporitzadors i oscil·ladors.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...	1		
			CA3.3	3.3 Verifica el funcionament dels circuits de temporització.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...	1		
			CA3.4	3.4 Verifica el funcionament dels circuits oscil·ladors.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...	1		
			CA3.5	3.5 Utilitza els instruments de mesura adients.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...	0		
			CA3.6	3.6 Munta o simula circuits.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...	1		
			CA3.7	3.7 Visualitza els senyals més significatius.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...	1		
			CA3.8	3.8 Descriu aplicacions reals dels circuits amb dispositius integrats de temporització i oscil·lació.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...	0		
			CA3.9	3.9 Realitza les tasques que cal fer individualment amb autoavaluació i cogavaluació.	Aquest indicador ha estat definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	No està definit aquest indicador de la rúbrica...	1		
			CA3.10	3.10 Presentació del Projecte	Definit correctament i amb detall...	Aquest indicador ha estat definit correctament...	Defineix de forma parcial aquest indicador...	Indicador de la rúbrica...	0		

ANEXO4: Manual de Programación Arduino

Usos de este Manual

1. Objetivos

a. El objetivo de este Manual de Arduino es que los alumnos, a quienes a partir de ahora definiremos como **los jugadores,** aprendan a desarrollar aplicaciones en el entorno de 'Arduino' para integrarlo en un Reto PBL, en el que deberán aplicar todos los conocimientos adquiridos en los Módulos listados en el punto siguiente del presente documento.
b. La evaluación del reto dependerá en concreto de **la calidad y de la dificultad de las soluciones propuestas por el alumno**, y en menor medida, de las soluciones elaboradas por el fabricante de Hardware y dispositivos de 'Arduino' y sus proveedores.

2. ¿A quién va dirigido?

El reto iría dirigido a alumnos de la familia de la electricidad-electrónica, y en concreto a aquellos en los que se imparte módulos de:
CFGM de instalaciones eléctricas y automáticas DUAL, y en sus módulos:

MP07 de Máquinas Eléctricas. UF1 Transformadores
UF2. Máquinas Rotativas de corriente continua
UF3. Máquinas Rotativas de corriente alterna

MP09 de Electrónica. UF1 Electrónica Digital no Programable
UF2. Electrónica Analógica
MP10 de Electrotecnia. UF1 Corriente Continua y electromagnetismo
UF2. Corriente Alterna
UF3. Máquinas eléctricas

Aunque para el presente ejemplo de 'Maqueta de la Sala de Actos' nos hemos centrado específicamente a trabajar los criterios de Evaluación (CA) del Módulo 09 de Electrónica.
Los **'jugadores'** escogerían el proyecto a desarrollar, definiendo las áreas de desarrollo, y las condiciones del propio.

3. La Comisión 'Arduino'.

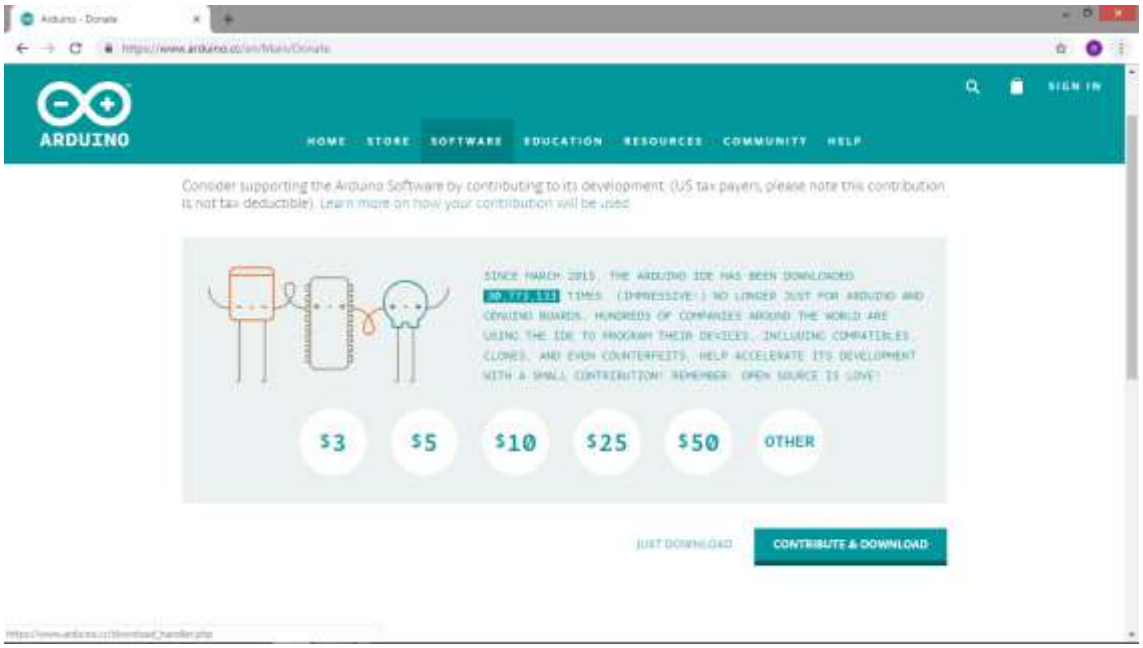
Definimos la **Comisión 'Arduino'** como un Comisión organizativa, a incorporar en todos los centros educativos, en los que este proyecto interdisciplinar sea de aplicación.
Sus competencias como Comisión serían las siguientes:

El presente documento es un modelo a considerar de inicio, si bien, en ningún caso será definitivo. Los centros deberán considerar la incorporación de nuevo material, al documento, tras contrastar de su veracidad.
El organismo a decidir, tras su evaluación, la modificación del manual, incorporando nueva información o eliminando la existente, será **la comisión 'Arduino'.**
La **Comisión 'Arduino'** se formará a principios de curso, y con la representación de dos alumnos escogidos por sus compañeros, y tres representantes del claustro de profesores.
Los miembros de la **Comisión 'Arduino'** no podrán proponer modificaciones propias sobre el manual ni sobre la rúbrica de evaluación. Las modificaciones serán de origen del resto de la comunidad del centro. Y, relativas a la temática, o a los ejemplos de proyectos desarrollados en el presente curso, así como al contexto de aclaraciones y observaciones que pudieran ser de utilidad a 'jugadores' de posteriores curso.

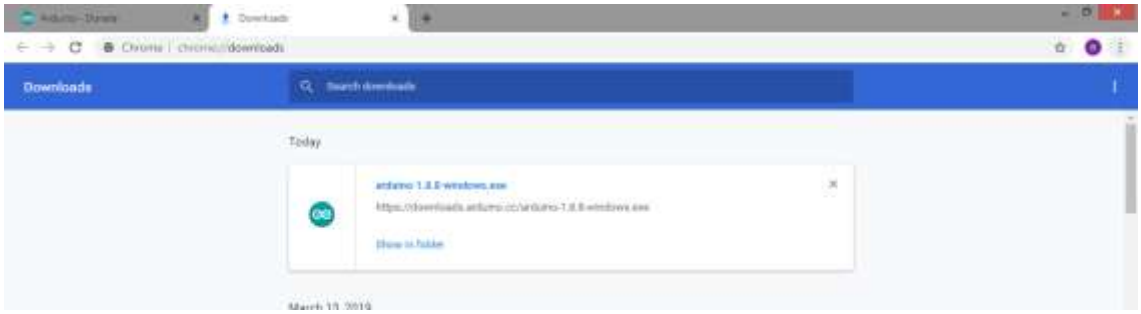
CRITERIOS BÁSICOS DE PROGRAMACIÓN

Instalación entorno de desarrollo ‘ARDUINO’

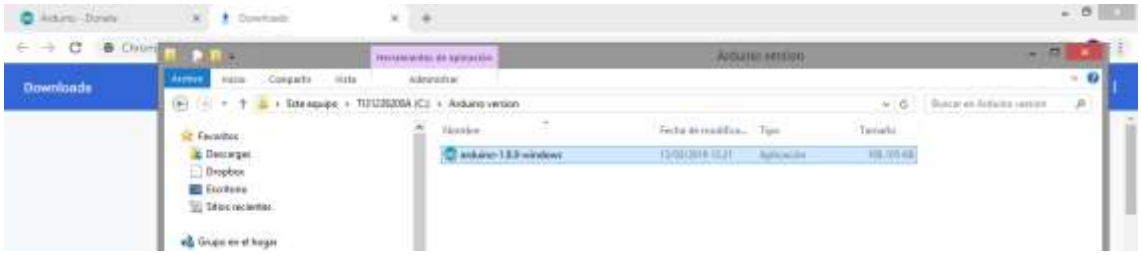
Arduino proporciona en su web oficial su entorno de Desarrollo libre y gratuito. Este entorno de desarrollo es compatible con los sistemas Operativos de Windows, Mac, y Linux.
Nos lo descargamos. En este caso será Windows Installer for Windows XP and UP.



‘Just download’. Se nos descargará en la carpeta de descargas. La abrimos en ‘Show ALL’ abajo a la derecha…



Ya nos lo hemos bajado…Si abrimos ‘Show in Folder’, podremos gestionar el ejecutable.



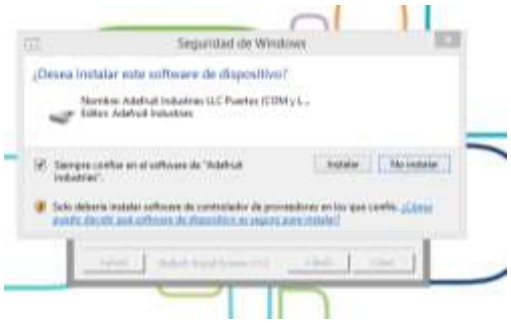
Le decimos que ‘I Agree’.(Yo estoy de acuerdo)

Ahora le decimos que…’Next’

Le decimos que …‘Install’

Ya se está instalando…

¡Paciencia! ¡Paciencia! ¡Que ya acaba!!



Es posible que el sistema te pida permiso para instalar algún ‘driver’ en tu ordenador…Yo siempre confío en el software de ‘Adafruit Industries’ porque son unos tíos que molan mucho…

Ya has finalizado la instalación en tu ordenador…

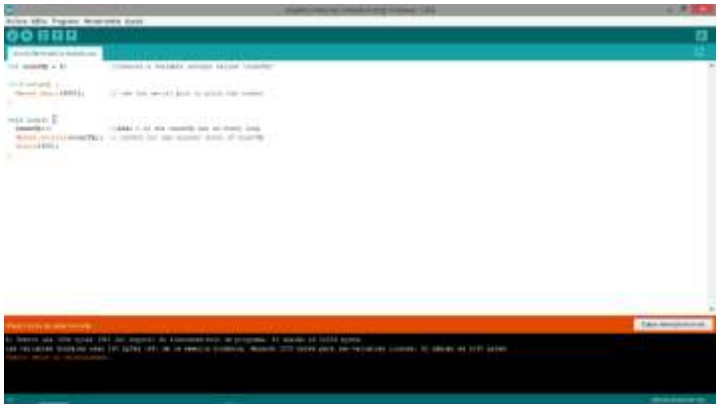
Te aparecerá el logo infinito de ‘Arduino’ en el escritorio de tu PC…¡Dále! ¿ A qué esperas?

Ya puedes empezar tu nueva aventura ‘Arduino’…¡Bienvenido!

Conexión de un ‘Arduino’

Al conectar por primera vez el ‘arduino’ a nuestro ordenador, es posible que tengamos que esperar a que se configuren los drivers…

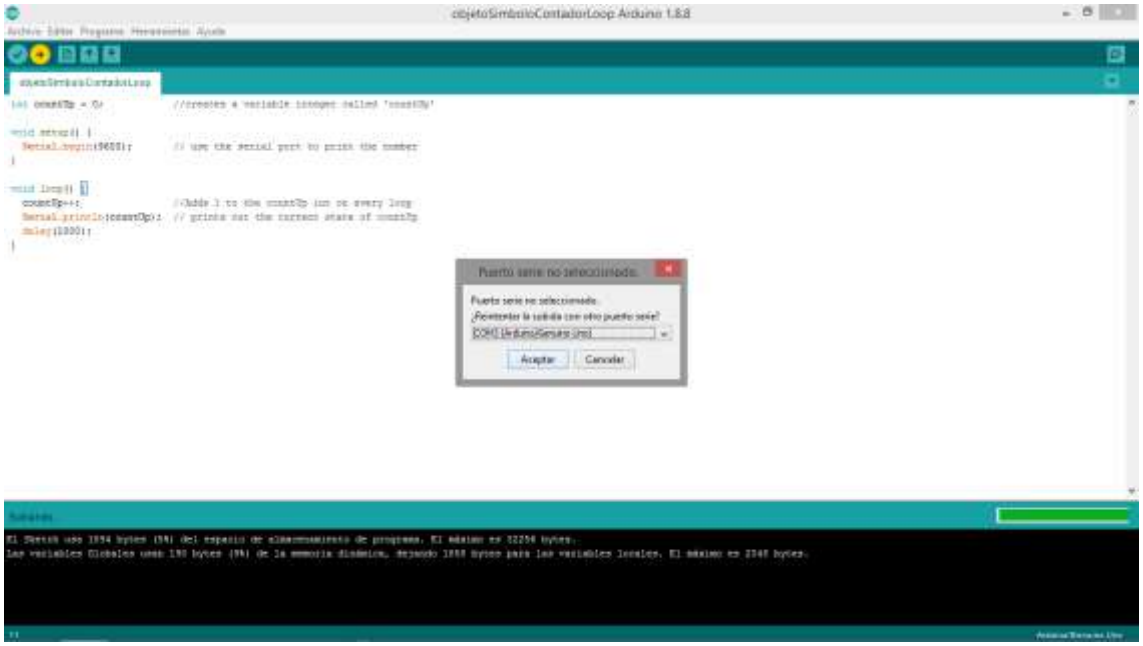
Y que al Compilar nos salga este mensaje:



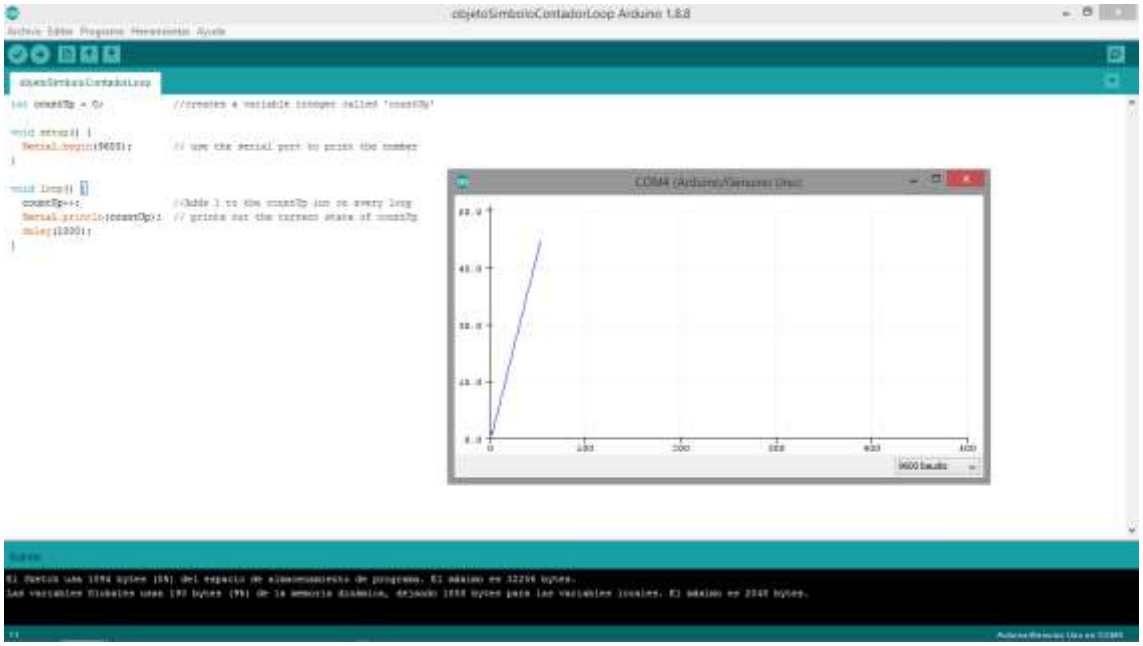
Puerto Serie no seleccionado. Podemos seleccionar el puerto ‘Com’ en el submenú ‘Puerto’ del Menú ‘Herramientas’.

Es el puerto Usb que vamos a usar para programar el ‘Arduino’…
Si no está seleccionable, conecta en Arduino a otro puerto Usb de tu Ordenador…

Volvemos a compilar. Nos tiene que salir este mensaje:

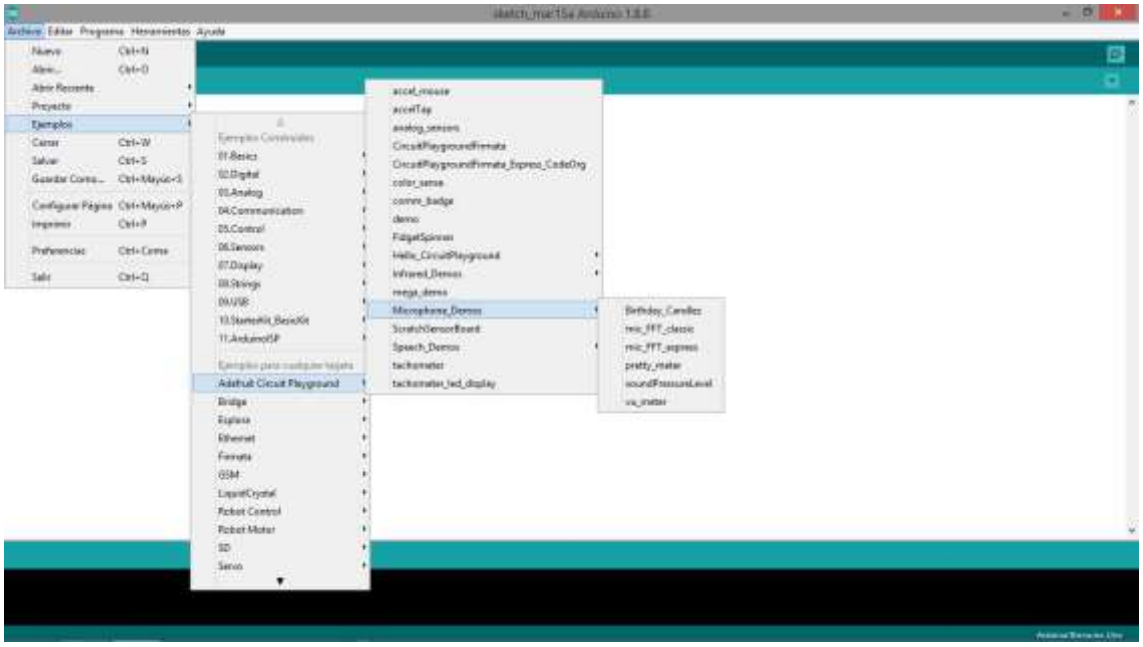


Escogemos la aplicación de ejemplo el programa ‘ObjetoSimboloContadorLoop’ para comprobar que todo funciona. Se tiene que poder visualizar su ejecución en el **Serial Plotter** del Menú ‘Herramientas’:



Explorar el entorno de desarrollo ‘Arduino’

En el menú ‘Archivo’ encontraremos los submenús que son habituales en todas las aplicaciones de software: **Nuevo, Abrir, Abrir Reciente**(listado de archivos con los que has estado trabajando recientemente), **Proyecto, Cerrar, Salvar**(las modificaciones que has hecho en un archivo. Recomiendo ‘darle a este submenú’ antes de hacer el ‘debugging’ y compilarlo), **Guardar Como…**(para guardar el nuevo proyecto), **Configurar Página** (Configura la página que vas a Imprimir), **Imprimir, Preferencias, Salir**
Además, podemos encontrar en el menú ‘Archivo’ un práctico submenú de **ejemplos** para empezar a aprender…
Trabajaremos con algunos de estos ejemplos para desarrollar nuestro tutorial…



En el Menú ‘Editar’ podemos encontrar los submenús o instrucciones:
Deshacer (instrucción), **Rehacer**(por si has hecho deshacer, y después te arrepientes…), **Cortar, Copiar, Copiar al foro**(https://forum.arduino.cc/index.php?board=32.0 ‘Arduino’ no es solo un entorno de desarrollo, es también un grupo social al que recurrir cuando no sabes que hacer), **Copiar como HTML** (para colgar en una Web), **Pegar, Seleccionar Todo, Ir a línea…**(Buscar una línea en concreto en el código que has desarrollado), **Comentar/Descomentar, Aumentar Sangría o Disminuir Sangría**(para desplazar las líneas horizontalmente. Tener las líneas de órdenes diferenciadas por instrucción, a veces hace el código más entendedor, sino es así, es porqué el código es muy difícil…)
También podemos encontrar los submenús **Incrementa o Reducir Tamaño Fuente**, y **Buscar…** (Buscar instrucción en el código fuente del programa), **Buscar Siguiente, o Buscar Anterior**(o sea, buscar hacia abajo, o hacia arriba)

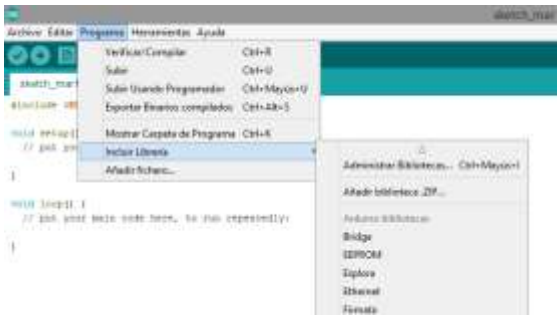
En el Menú ‘Programa’ podemos encontrar los submenús **Verificar/ Compilar** (la verificación o debugging comprueba que no haya errores significativos en las expresiones del código fuente que has desarrollado

Podemos encontrar en la barra de herramientas superior los accesos directos a:

Para comprobar el código programado, primero se verifica,

Para, a continuación, compilarlo, generando un archivo binario que el software envía al micro controlador, y que es el que este interpreta.

Otros submenús son **Subir, Subir Usando Programador, Exportar Binarios Compilados, Mostrar Carpeta de Programa, Incluir Librería, Añadir Fichero**.
Los primeros son para compartir, gestionar, copiar o enviar los ficheros Binarios Compilados.
Incluir carpeta permite agregar las librerías de funciones o procedimientos desarrollados para ‘Arduino’, para simplificar el desarrollo de nuestros programas.



Pongamos por ejemplo la librería EEPROM…la seleccionamos, y queda incorporada automáticamente en la cabecera de nuestra aplicación.
A partir de entonces, podremos utilizar todos los elementos de esta librería.

El entorno de desarrollo de ‘Arduino’ ha añadido en la cabecera de la aplicación la librería <EEPROM.h>.

En el Menú ‘Herramientas’ encontraremos los siguientes submenús:
Auto Formato, Archivos de Programa, Reparar codificación&Recargar, **Administrar Bibliotecas…** (Permite actualizar Bibliotecas específicas, como las que se utilizan en el protocolo de comunicación ModBus…),

Monitor serie

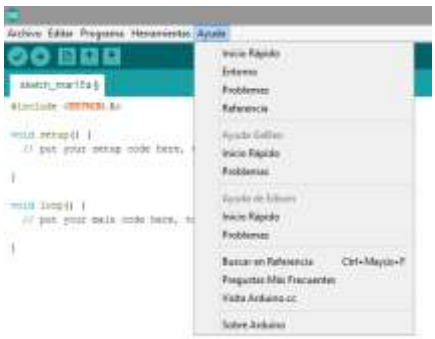
Serial Plotter(Muestra en una pantalla virtual de símbolo de Sistema, antigua pantalla de MsDos, las instrucciones que han sido declaradas como Serial.printf().

```
lcd.printf("Hello World");
Serial.printf("Hello World\n");
SoftSerial.printf("Hello World\n");
Serial.printf("Value is:%3d\n"), value);
```

Se utiliza habitualmente para testear la ejecución de una aplicación que se esta ejecutando...

Placa(Modelo de placa que vamos a utilizar en nuestra aplicación. Debemos seleccionarla al empezar el proyecto), **el Puerto**(USB desde el que nuestro PC se comunica al 'Arduino'. Se configura automáticamente. En el caso que no sea así deberemos ir a nuestro administrador de Hardware, a los puertos USB, a comprobar que estos han sido configurados correctamente, y a identificarlos…pero esto es especialmente raro en las últimas versiones de Software de desarrollo. Lo normal debería bastar con cambiar de puerto Usb del ordenador y volver a probar).
Obtén información de la placa(de la placa del Arduino), **Programador,**

Quemar Bootloader.
No es difícil tener un mal día y quemar el Atmel. (el micro controlador)
Cuando vayas a la tienda a comprarlo, te lo venderán sin la instalación del Bootloader.
El Bootloader es el ejecutable que se encarga de establecer comunicación entre 'Arduino' y ordenador al alimentar la placa de Arduino.
Para instalárselo necesitarás otro dispositivo Arduino. 'Quemar Bootloader' es el menú que se encarga de ello.



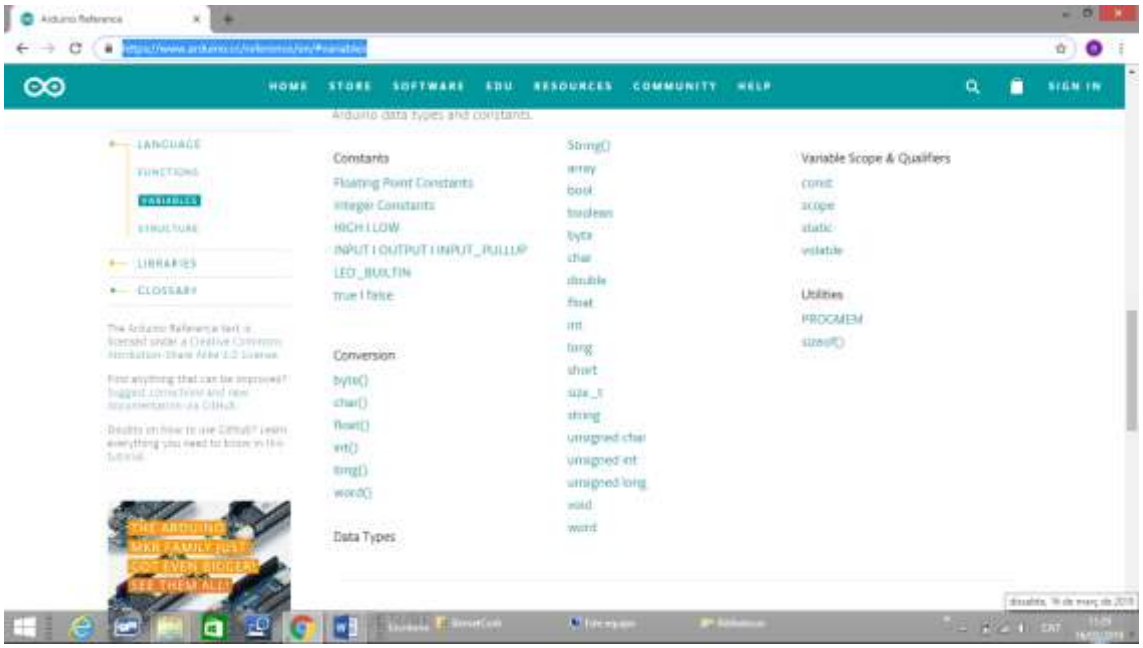
El menú 'Ayuda' da acceso directo a la comunidad 'Arduino', ya sea bien para consultar ejemplos, acceder al Fórum de la comunidad, o buscar glosarios de Referencia concretos…

En la parte superior del cuadro de texto, bajo los menús, encontramos las cabeceras del programa. Es la parte del código fuente donde vamos a declarar **librerías** (p.ej.#include<EEPROM.h> que se ve en la imagen), **variables globales** (las variables cuyo ámbito de trabajo es todo el programa), y archivos con librerías propias que adjuntaremos…

Bajo la cabecera programa encontramos el Procedimiento **'Setup'**. En inglés nos indican que esta parte del programa solo se ejecutará una vez, al iniciar el sistema.
Y a continuación, el procedimiento **'Loop'**, donde crearemos el código fuente para una aplicación indefinida en el tiempo.

Declaración y tipos de variables en 'Arduino'

https://www.arduino.cc/reference/en/#variables



- En 'Arduino' podemos declarar datos de tipo **Constantes:**
  Las constantes **Input/output/Input_Pullup**, por ejemplo las utilizaremos para definir el modo de funcionamiento de las patillas de Entrada/Salida del micro controlador 'Arduino'.
  A diferencia de Input/output, Input_Pullup ha sido configurado para añadir una resistencia interna de 20kΩ que protege el pin del micro controlador, de tensiones de entrada negativas, o superiores a 5 voltios dc, que podrían dañarlo.
  Las constantes **HIGH/LOW** se utilizan para cambiar el estado de una salida de forma rápida, por ejemplo.
  True/False es una constante definida inicialmente como Booleana. Una Booleana solo podrá tener 2 estados, que son equivalentes a 1(true), y a 0(false).
  Podemos también en 'Arduino', declarar **constantes numéricas**.

Las constantes Numéricas pueden ser:

BASE	EXAMPLE	FORMATTER	COMMENT
10 (decimal)	123	none	
2 (binary)	B11001	leading 'B'	only works with 8 bit values (0 to 255) characters 0&1 valid
8 (octal)	0173	leading "0"	characters 0-7 valid
16 (hexadecimal)	0x7B	leading "0x"	characters 0-9, A-F, a-f valid

P.ej.El número binario declarado como:

```
n = B101;
```

// es equivalente a 5 en decimal ((1 * 2^2) + (0 * 2^1) + 1)

Solo es compatible con números de 8 bits.

Lo recordáis de la UF de electrónica digital, verdad?

Las constantes numéricas hexadecimales se declaran:

```
n = 0x101;
```

// same as 257 decimal ((1 * 16^2) + (0 * 16^1) + 1)

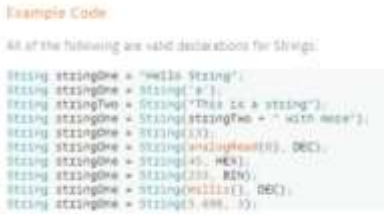
- En 'Arduino' podemos declarar datos de tipo **Variables:**

Las variables pueden ser de tipo:
https://aprendiendoarduino.wordpress.com/2015/03/26/tipos-de-datos/

Las constantes se declaran usando los calificadores de constantes 'const' o 'static':

Las variables se declaran:

1. Variables de cadenas de caracteres 'string'



‘stringOne’ es el nombre de la variable que el programador ha decidido, ‘String’-es el tipo de variable que hay que definir.

2. Variables numéricas

https://www.arduino.cc/reference/en/language/variables/data-types/int/
Podemos encontrar los identificadores de todos los tipos de variables en este link.

```
int NomVariableInteger=0;
int NomVariableInteger2;
float NomVariableFloat;
```

Las variables declaradas tienen su ámbito de uso en la parte del programa en la que han sido creadas, ya sea función, procedimiento o ámbito global.
Tras ser declaradas, se hará el uso:

```
NomVariableInteger=NomVariableInteger+NomVariableInteger2
```

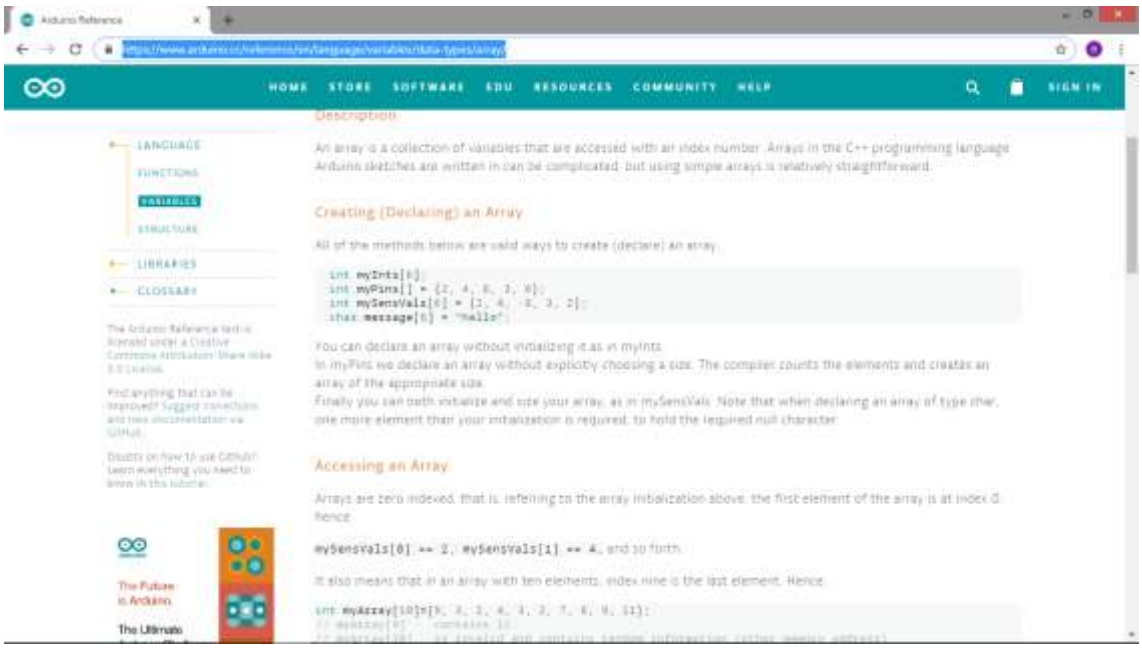
//A la izquierda está la solución de la operación, y a la derecha, la operación de origen. //A NomVariableInteger le sumaremos su valor inicial, y el valor de NomVariableInteger2

3. Las variables también se pueden Convertir de Tipo, utilizando las funciones proporcionadas por ‘Arduino’

La definición de estas funciones de Conversión se puede resumir:
Char(data);
Por ejemplo convierte cualquier valor al tipo de dato Char(carácter).
Cuando data ha sido declarado:

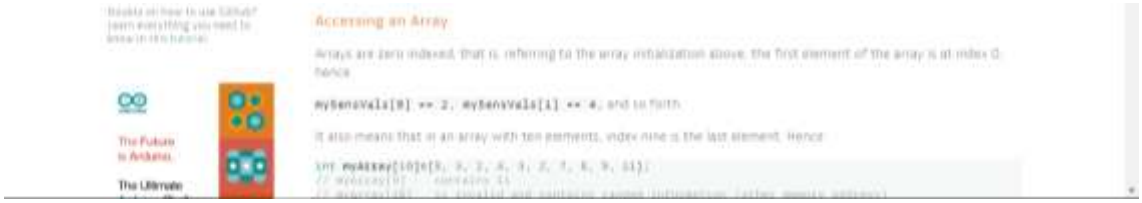
```
Int data=5;
Char letra;
Letra=Char(data);
```

4. Declaración de Arrays
   https://www.arduino.cc/reference/en/language/variables/data-types/array/



Un array es una colección de variables a las que se puede acceder a través de un número índice.
Las declaramos:

Y accedemos a ellas:



Observamos que el índice 0 del array mySensVals es igual a 2. El menor índice de un array es el 0.
'myArray' puede indexar 11 elementos, a los que se accede uno por uno por la expresión:

```
myArray[numIndice];
```

Bien sea para escribir el valor en otra variable numérica,

```
VarNumericaNueva=myArray[numIndice];
```

O para leer el valor de una variable numérica antigua, y memorizarla en el Array:

```
myArray[numIndice]=VarNumericaAntigua;
```

Expresión lógica en ‘Arduino’. Estructuras de control

https://www.arduino.cc/reference/en/#structure

A partir de los operadores de comparación…

```
x == y (x es igual a y)
x != y (x es diferente a y)
x <  y (x es menor a y)
x >  y (x es mayor a y)
x <= y (x es menor o igual a y)
x >= y (x es mayor o igual a y)
```

…se definen las instrucciones que se ejecutarán en las diferentes estructuras de control.

➢ Condición If/Else

La estructura de control if/else define que:

```
if (condition1) {
//instrucción1
}
Else(condition2){
//instrucción2
}
```

Cuando se cumpla la condition1, solo se ejecutará la instrucción1.
Y que cuando se cumpla la condition2, solo se ejecutará la instrucción2.
La estructura de control **if/else** solo se ejecutará una vez por ciclo.

➢ Condición Switch case

```
switch (NombreVariable) {
  case label1:
    //instrucción1
    break;
  case label2:
    //instrucción2
    break;
  default:
    //instrucción3
    break;
}
```

Labels pueden ser cadenas de texto, valores integer, o conditions. Depende de lo que necesitas en ese punto de la aplicación…
Cuando se cumpla la label1, solo se ejecutará la instrucción1.
Y que cuando se cumpla label2, solo se ejecutará la instrucción2.
Cuando ya no haya más labels con que comparar (default o por defecto), y todas las comparaciones previas hayan sido rechazadas, se ejecutará la instrucción3.
Tras cada instrucción se utiliza un **break** para salir de la instrucción. Si bien se debe evitar que la aplicación pueda entrar en varias labels en un mismo ciclo de ejecución.
La estructura de control **Switch** solo se ejecutará una vez por ciclo.

➢ Bucles While

```
while (condition) {
  //instrucción1
incrementaVariableCondition+1
}
```

Cuando se cumpla la condition, el programa ejecutará la instrucción1, y lo hará tantas veces como condition se cumpla. Si NO añadimos **'incrementaVariableCondition+1'** la instrucción se repetirá indefinidamente, hasta provocar el fallo de la aplicación. Si el tiempo de ciclo es superado, el sistema detiene la aplicación y resetea el micro controlador automáticamente…

```
incrementaVariableCondition=0;
while (incrementaVariableCondition<200) {
  //instrucción1
incrementaVariableCondition= incrementaVariableCondition+1;
}
```

a. ¿Cuántas veces se ejecutará la instrucción1?
b. En el caso la declaración inicial de 'incrementaVariableCondition = 200', **¿Se llegaría a ejecutar instrucción1?**

- Bucles do/While

```
do {
  //instrucción1
incrementaVariableCondition+1
} while (condition);
```

El programa ejecutará la instrucción1 al menos la primera vez en la ejecución del ciclo. Si al llegar la while, la condición no se cumple, ejecutará la instrucción1 tantas veces como condition NO se cumpla. Si NO añadimos **'incrementaVariableCondition+1'** la instrucción se repetirá indefinidamente, hasta provocar el fallo de la aplicación. Si el tiempo de ciclo es superado, el sistema detiene la aplicación y resetea el microcontrolador automáticamente…

- Bucles For

```
for (initialization; condition; incrementaVariableCondition+1) {
  //instrucción1
}
```

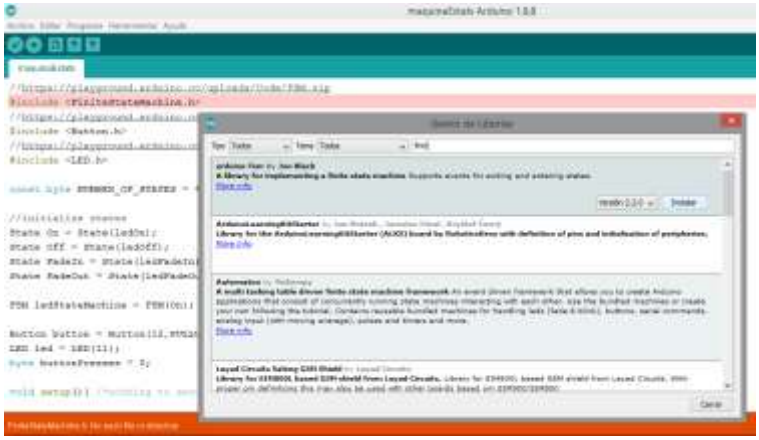
Cuando se cumpla la **initialization**, el programa ejecutará la instrucción1, y lo hará tantas veces como **condition** NO se cumpla.
Si NO añadimos **'incrementaVariableCondition+1'** la instrucción se repetirá indefinidamente, hasta provocar el fallo de la aplicación. Si el tiempo de ciclo es superado, el sistema detiene la aplicación y resetea el micro controlador automáticamente…

```
for (int i = 0; i <= 255; i++) {
  analogWrite(PWMpin, i);
  delay(10);
}
```

En este ejemplo, la variable numérica 'i' se inicializará a '0' en la propia instrucción, para posteriormente incrementarse cada vez que se ejecute la línea en la que está la instrucción **'for'.**
El valor de la i se escribirá en la salida analógica definida en la variable PWMpin, se esperará 10 milisegundos, y volverá a ejecutarse el '**for**'. Se repetirá mientras 'i' sea inferior o igual a 255.

Máquina de estados

https://playground.arduino.cc/Code/FiniteStateMachine/
La máquina de estados es una técnica de programación que se aplica solo para sistemas **secuenciales**…
Cada fabricante tiene sus librerías, y su estructura, pero en esencia, se trata de un switch case que va habilitando ordenadamente los pasos según se cumplen las condiciones.



Para ejecutar la aplicación ejemplo deberéis instalar la librería 'finiteStateMachine'…

El cambio de estados se puede motivar a partir de teclas del ordenador, como en esta aplicación, o de pulsadores en el Arduino.

Lógica aplicada

Quiero empezar este apartado recomendando el libro, del cuál he extraído además de algunos ejemplos, el método del que me voy a servir para explicar todo este apartado:
Iniciación a la programación. Lógica y diseño. Joyce Farrel. Editorial Paraninfo. 1999
La Programación es el conjunto de estrategias, previsiones y perspectivas que vamos a necesitar para resolver un problema. Es una técnica ordenada, eficiente que se puede aplicar en la resolución de un problema matemático, como en cualquier otra circunstancia en la vida.
La perspectiva podría definir la cantidad de puntos de vista, u observaciones que hemos estudiado antes de ponernos a resolver el problema. Son perspectivas, por ejemplo, los diferentes puntos de vista de cómo debería funcionar la máquina de los 12 ingenieros de una empresa, así como la opinión del cliente. ¿Cuál de estas perspectivas es la correcta? Eso depende. Con los años, aprendes a identificarla. A eso se le llama experiencia. Lo mismo pasa con la programación.
Cuantas más perspectivas hayas considerado, más fácil te será identificar la más correcta. Abre la mente a propuestas y comentarios ajenos, aunque a priori puedas considerar que son verdaderas barbaridades o locuras. La perspectiva más correcta puede ser una síntesis de varias propuestas. Intenta racionalizarlas todas. Nadie tiene la verdad absoluta. Ni siquiera tú.

La previsión, podemos definir cuando los puntos concretos son tan evidentes que son previsibles. Los vamos a poder intuir antes de que se produzca el fallo. Cuanto más estudiemos el programa, antes de subirnos al teclado del ordenador, menos tiempo perderemos en la evaluación del mismo. El ensayo de 'fallo-corrección', acaba siempre con la paciencia del cliente.
El mismo concepto se puede emplear en la vida. Cuanto más previsores seamos, más planifiquemos nuestros viajes en vacaciones, menos problemas tendremos. Y cuando sean evitables, puesto que los habremos previsto, tendremos más margen de maniobra. Habremos desarrollado un programa más flexible o adaptable.

Estrategias. Las estrategias lo son todo. Has definido ya la perspectiva, hemos previsto los posibles problemas con que nos vamos a encontrar, ahora deberás escoger el camino a seguir…Recuerda **para ello** *los Bucles y Condiciones* que hemos explicado en el punto anterior.
Un camino equivocado puede hacerte perder el tiempo y exaspera al cliente.
¡¡ Ejemplo !!
Nos acabamos de despertar. Es un domingo festivo. Decidimos bajar a la panadería a por croissants para desayunar un suizo…
1. Para empezar, si nuestra sociedad NO ve adecuado que los vecinos paseen por la calle en pijama a las 8 de la mañana, ¿ deberemos cambiarnos de ropa ?
Para tomar una decisión deberemos utilizar **la condición if/else/elseif…**
Si es que sí, deberemos cambiarnos de ropa…

1.1.Aquí decidimos la ropa que nos vamos a poner.
Nos vamos a poner los tejanos azules y una camisa verde…

Nos vamos a poner los tejanos azules y una camisa azul…
Nos vamos a poner los tejanos azules y una camisa blanca…
Nos vamos a poner los tejanos negros y una camisa verde…
Nos vamos a poner los tejanos negros y una camisa azul…
Nos vamos a poner los tejanos negros y una camisa blanca…

En el caso de que tengamos toda la ropa sucia, o no sepamos que ponernos…**(default case)**

Vamos a salir en chándal.

Cuando las opciones son infinitas o múltiples deberemos utilizar una condición **Switch case…**

Si es que no, podemos salir a la calle sin cambiarnos de ropa…
¿Deberemos ponernos calzado?
Si es que si, ¿ nos hemos puesto los tejanos ?

Nos vamos a poner zapatos.
¿ Nos hemos puesto el chándal?
Nos vamos a poner unas zapatillas de deporte.

Si es que no, vamos a salir con zapatillas de ir por casa…
Esto son dos **condiciones if /else/elseif** anidadas.
A continuación, vamos a definir la rutina que supone salir a la calle:
Nos ponemos las llaves en el bolsillo… ¿Qué ocurriría si las olvidásemos en casa? Ya no podríamos entrar en nuestro piso, sin la ayuda de un cerrajero, y nos costaría muy caro…Hay que ser previsor…
Cogemos el dinero para pagar los croissants. Si lo olvidásemos, tendríamos que volver a casa a por él…Tendríamos que hacer el viaje por segunda vez. Hay que ser previsor…
Abrir la puerta…
Salir del apartamento.
En el rellano del edificio, cerramos la puerta de nuestro apartamento,
Nos damos la vuelta, y nos dirigimos al ascensor.
¿Durante cuánto tiempo pulsaremos el botón del ascensor?
Pulsamos el pulsador del ascensor durante 3 segundos.
¿Cuántas veces pulsaremos el botón antes de llegar a la conclusión de que alguien está utilizando el ascensor o que está estropeado?
Pulsaremos el botón seis veces…Esto son dos Bucles anidados. En programas de desarrollo como BorlandC++, o Visual C++, se utiliza habitualmente el Bucle **For(),** porqué permite declarar varias condiciones, inicializar varias variables, e incrementar varios contadores simultáneamente, pero en entornos de desarrollo como ‘Arduino’ o Visual Basic. NET nos acostumbraremos a utilizar **while()** o **do…while()**
La expresión sería:

```
While(NúmeroVecesQueHePulsado<= 6) {
        While(TiempoQuePulsoCadaVez<=3 segundos){
                El tiempo no tenemos que incrementarlo. Lo hace por sí mismo. O al menos
                eso decía en su libro “Momo” Michael Ende.
                Para capturarlo, eso sí, utilizaremos la función millis(); cada vez que
                entremos en esta parte del programa…
                ( Ver definición en su apartado y en el ejemplo del robot)
        }
Incrementa NúmeroVecesQueHePulsado+1
}
```

El bucle que se encuentra en el interior, se ejecutará tantas veces como el bucle exterior lo solicite…
El programa no saldrá de la estructura anidada hasta que se haya llamado al ascensor 6 veces…
A continuación, decidiremos:
Si ha llegado el ascensor, ¿nos montaremos en él para bajar a la calle?
Si ha llegado el ascensor, ¿bajaremos a la calle por las escaleras?
Si no ha llegado el ascensor, ¿bajaremos a la calle por las escaleras?
Si no ha llegado el ascensor, ¿ Volveremos a entrar en casa?
¿Qué condición utilizarías? ¿Quizás un **if/else/elseif**? ¿Quizás un **a**?
Hemos llegado al vestíbulo del bloque. ¿La puerta está abierta?
Si es que sí, ¿Salimos a la calle a comprar los croissants?
Si es que sí, ¿Nos quedamos en el vestíbulo?
Si es que No, ¿Abrimos la puerta para salir a la calle?
Si es que No, ¿Atravesamos el cristal de la puerta para salir a la calle?
Si es que No, ¿Nos quedamos en el vestíbulo?
¿Qué condición o bucle utilizamos para tomar esta decisión?
Y, una vez en la calle, ¿ … a cuál de las 10 panaderías que hay en el barrio iremos?
¿Qué condición o bucle utilizamos para tomar esta decisión?

¿Qué camino cogeremos para ir a la panadería?

Se puede observar que nos dedicamos instintivamente a programar nuestras decisiones de la vida diaria. Programar software o micro controladores consiste en aplicar y desarrollar esa habilidad en el campo de la electrónica.
Para ello nos servimos de pseudocódigo y diagramas de flujos, que componen el algoritmo de un programa. Cuanto mayor sea el programa, más detallados deberán ser estos algoritmos, y por lo tanto, los diagramas de códigos.
Podemos y debemos también dividir en bloques o Funciones las partes del programa que más se repiten. Reutilizándolas

¡¡Ejercicios!!

1. Realizar el algoritmo de un programa que nos pida introducir dos números por teclado, y si la suma de ambos es mayor a 100, nos calcule la media aritmética. Sino lo son, calcule el producto.

2.Realizar un algoritmo de un programa que visualice la tabla de multiplicar de un número introducido por teclado(la tabla entera hasta el 10)

3.Realizar el algoritmo de un programa que tras introducir por teclado 50 números, los muestre por el orden inverso al que fueron introducidos…
p.ej. Introduces: 4,7,6,2…y visualizará: 2, 6,7, 4…

4.Realizar el algoritmo de un programa que tras introducir por teclado 100 números enteros, muestre al final cuántos de estos números eran pares…

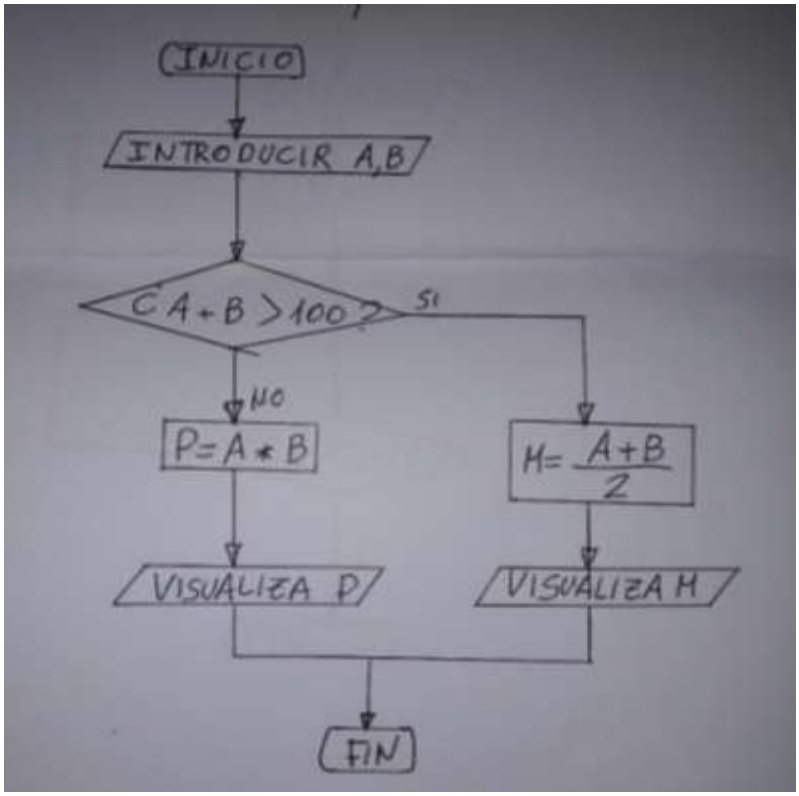
5.Realizar el algoritmo de un programa en el que introducimos por teclado 2 números enteros, y positivos. Por pantalla mostraremos el producto de estos, pero sin utilizar el operador de multiplicación.

6.Realizar el algoritmo de un programa en el que introducimos por teclado 2 números enteros, y positivos. El primero será el Dividendo y el segundo el divisor. Por pantalla mostraremos el cociente de estos y el resto de la división, pero sin utilizar el operador de división.

7.Realizar el algoritmo de un programa en el que introducimos por teclado 2 números enteros, y positivos. Por pantalla mostraremos la potencia sin utilizar el operador de potencia. El primer número será la base.

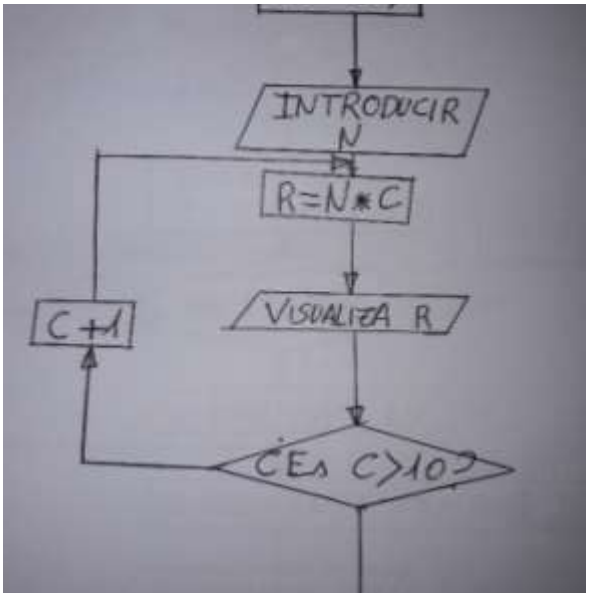
Resolución Problemas

1. Realizar el algoritmo de un programa que nos pida introducir dos números por teclado, y si la suma de ambos es mayor a 100, nos calcule la media aritmética. Sino lo son, calcule el producto.



Esto es un **Diagrama de Flujos…**

2.Realizar un algoritmo de un programa que visualice la tabla de multiplicar de un número introducido por teclado(la tabla entera hasta el 10)



Si el Bucle se repite mientras el contador 'C' sea inferior, el 'NO' se etiquetará en la rama que vuelve al inicio, y el 'SI' se etiquetará en la rama que va al FIN del programa.
Pero entonces, mostraríamos también la tabla del 11. Estaríamos cometiendo un error…
¿Cuál habría sido la expresión correcta?

```
While(C!=10){
        //Mientras C sea DIFERENTE a 10, Incrementa el contador C y continua haciendo la operación.
}
```

3.Realizar el algoritmo de un programa que tras introducir por teclado 50 números, los muestre por el orden inverso al que fueron introducidos…
p.ej. Introduces: 4,7,6,2…y visualizará: 2, 6,7, 4…

Este ejercicio lo podemos resolver de dos maneras. Podemos crear 50 variables de diferente nombre, y guardar los números que vas introduciendo en cada una de las variables, una por una. Mucho trabajo, y un código difícil de mantener (reparar o modificar).
La segunda es mucho mejor.
Vamos a declarar una array de tipo[] int (integer o numérica) de 50 elementos. Una array reserva la memoria que va a necesitar para todos los elementos que hemos solicitado.
Por ejemplo: int Pepito[49], que deberemos inicializar: int Pepito[49]={0};
Y a la que podremos vincular un contador: p.ej. ContArray de tipo int (integer o numérica). **El primer índice de un array es el 0.**
Con la expresión: int Pepito[ContArray] nos podremos desplazar en cada uno de los índices o espacios de memoria del array.
Hay Arrays de una dimensión o índice. Pero las hay también de dos y tres dimensiones…Se utilizan en BorlandC++, en VisualC++ para gestionar valores asignados al mismo objeto, así como registros de texto, a los que se puede acceder a los caracteres individualmente, a palabras, o al texto en conjunto. Sea cual sea su dimensión de un array, todos los datos serán del mismo tipo…

Pseudocódigo del ejercicio…

```
INICIO
Inicializas las variables, el contador a 0, el array int Pepito[ContArray]  con todos sus
índices a 0.

Mientras el índice sea inferior a 50,
        Introduce un número en teclado
        Guárdalo en int Pepito[ContArray]
        Incrementa contador ContArray
Ahora el índice es igual a 50, pero en este índice no hay nada. Por lo tanto:
Mientras el índice sea mayor o igual 0,
        Decrementa el contador
        Muestra en pantalla el número del registro
Ahora, el índice es -1, y ya hemos mostrado en pantalla todos los números del Array.
FIN
```

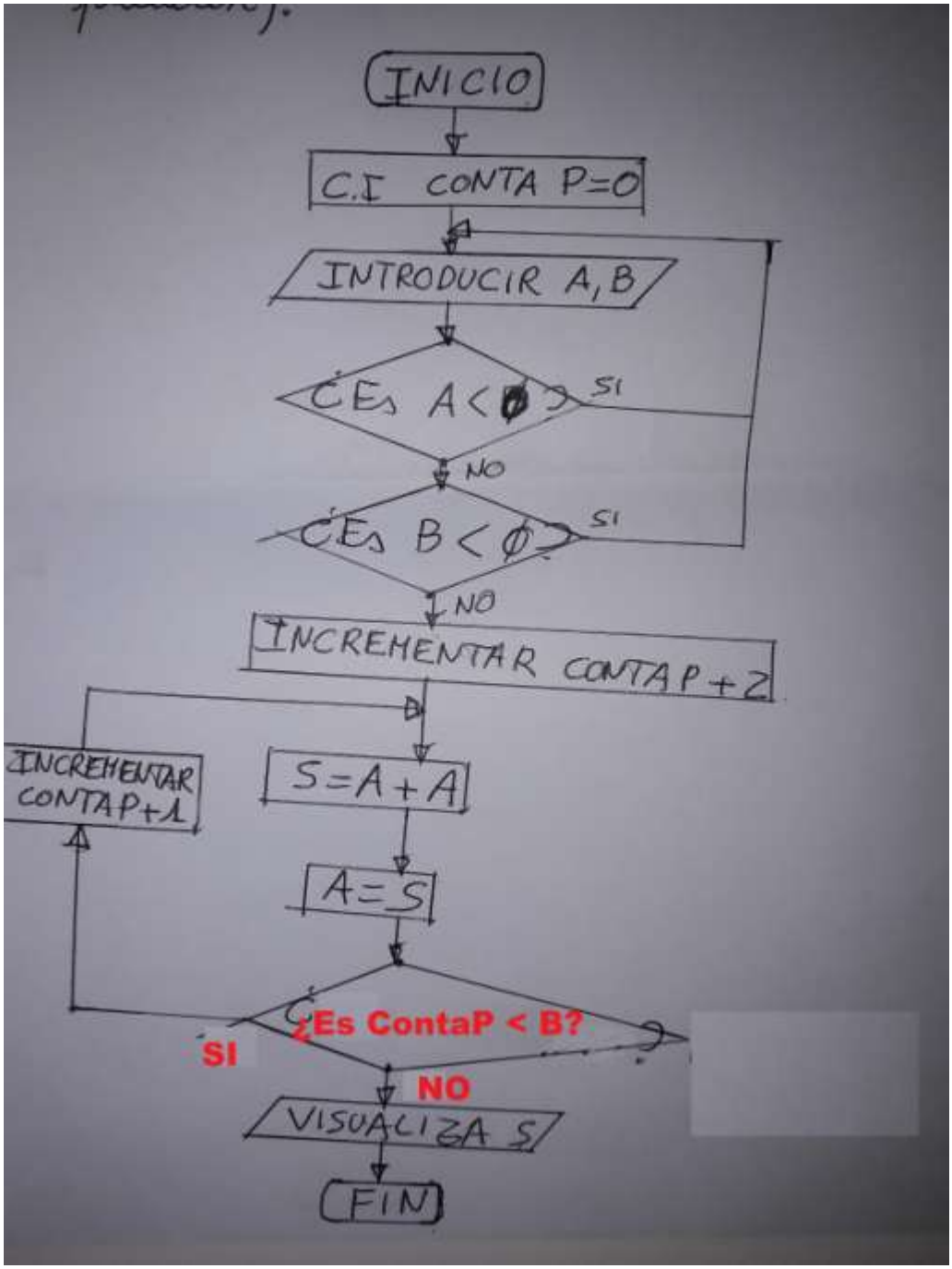
4.Realizar el algoritmo de un programa que tras introducir por teclado 100 números enteros, muestre al final cuántos de estos números eran pares…

Pseudocódigo:

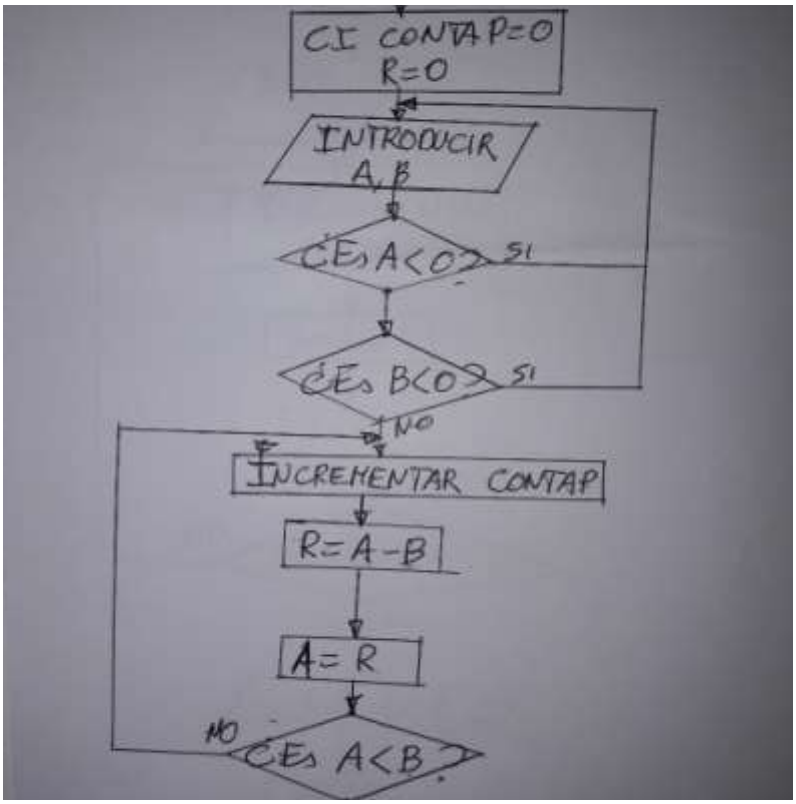
INICIO
Inicializas las variables, ’ContPares’, ‘ContArray’ a 0, el array int Pepito[ContArray] con todos sus índices a 0.
Mientras el índice sea inferior a 100,
    Introduce un número en teclado
    Guárdalo en int Pepito[ContArray]
    Incrementa contador ContArray
Ponemos ContArray=0 . Empezaremos a comprobar desde ‘0’.
Mientras el índice sea inferior a 100,
    Coge el número que esta en el registro asociado a este índice.
    ¿El módulo de este número dividido por 2 es diferente a 0?
    En ‘Arduino’ se representa con el símbolo ‘%’.
    Si es que sí, no hagas nada
    Si es que No, incrementa el contador que incrementa los números divisibles por dos o pares.’ContPares’

Muestra ’ContPares’ en pantalla.
FIN

5.Realizar el algoritmo de un programa en el que introducimos por teclado 2 números enteros, y positivos. Por pantalla mostraremos el producto de estos, pero sin utilizar el operador de multiplicación.



6.Realizar el algoritmo de un programa en el que introducimos por teclado 2 números enteros, y positivos. El primero será el Dividendo y el segundo el divisor. Por pantalla mostraremos el cociente de estos y el resto de la división, pero sin utilizar el operador de división.



En la primera parte del programa, este nos pregunta si alguno de los dos números es negativo.

En la segunda parte, va restando el divisor al dividendo hasta que este es menor al divisor, y por lo tanto, no podemos seguir restando.

7.Realizar el algoritmo de un programa en el que introducimos por teclado 2 números enteros, y positivos. Por pantalla mostraremos la potencia sin utilizar el operador de potencia. El primer número será la base.

Resolvemos por pseudocódigo

```
INICIO
Inicializas las variables, introducidoA, introducidoB, contaP, ValorPot a 0,
¿El número introducidoA(la base) es mayor a 0.?
        Si es que Si, introduce el número introducidoB.
        Si es que NO, vuelve a introducir el número introducidoA
¿El número introducidoB es mayor a 0.?
        Si es que Si, introduce continua los cálculos.
                contaP=2
                ValorPot= introducidoA* introducidoA;
                  Multiplica introducidoA por sí mismo, y guardarlo en una variable 'ValorPot'
                Mientras el contaP sea inferior o igual al número introducidoB
                  ValorPot= ValorPot* introducidoA;
                  Incrementa contaP+1
              Muestra el resultado ValorP en pantalla.
Si es que NO, vuelve a introducir los dos números.
FIN
```

Funciones

1. Funciones creadas por el usuario.

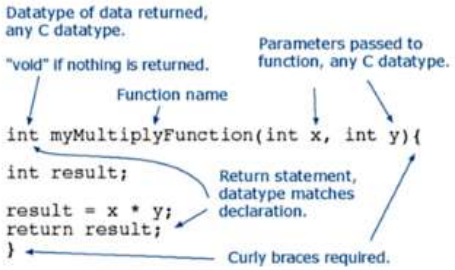
https://www.arduino.cc/en/Reference/FunctionDeclaration

¿Para qué sirven las funciones?

Fraccionar el código Fuente en funciones facilita al programador la tarea de crear modularmente piezas en código que podrá:

1. Un programa modular es más claro, y estructurado.
2. Su modulado también simplifica el mantenimiento y modificación
3. Sus Módulos pueden ser reutilizados en aplicaciones semejantes posteriores.
4. Los módulos facilitan la comprensión en aquellos fragmentos del código fuente en que se requiere una gran repetición.
   Se repite la llamada de la Función o módulo, pero el código de esta, es siempre el mismo, aunque el nombre de las variables en fragmento de código en el que se realiza la llamada, sea siempre diferente.
   Para tras ejecutar la función, devolver su resultado al área de código que hizo la llamada a la función.

5. En el **sketch() o escritorio** de programación existen dos funciones: **setup()** y **loop().** La primera solo se ejecuta en el momento de reiniciar el sistema, la segunda, se ejecuta de forma indefinida hasta que se interrumpe la alimentación.
   El resto de las funciones irán anidadas a estas dos funciones, a medida que el programador las desarrolle.



Anatomía de una función C

Nombre de la Función(Function Name).
El nombre que se le ha asignado a la función en este caso:
'myMultiplyFunction()'

Parámetros que pasamos a la función: int x, int y…tantos como necesitemos. Pueden ser de cualquier tipo. También arrays, de tipo puntero, o cadenas de caracteres o Strings…
El tipo de data devuelto (Datatype of data returned, any C datatype) es el dato devuelto al código fuente que llamó a la función y puede ser de todo tipo:
Void myMultiplyFunction(int x, int y) =>Cuando no se retorna ningún valor…

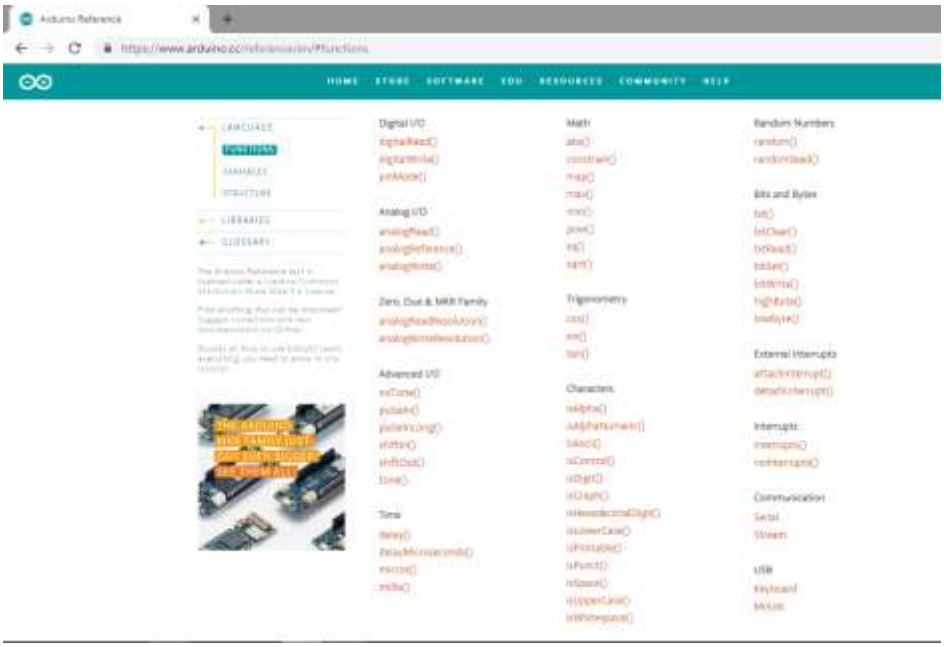
Char myMultiplyFunction(int x, int y) =>Cuando se retorna un carácter…
Float myMultiplyFunction(int x, int y) =>Cuando se retorna un valor float…

En estos casos, el valor devuelto por la función vendrá definido por la instrucción 'return()', y solo se puede retornar un valor por función ejecutada…

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

void loop() {
  int i = 2;
  int j = 3;
  int k;

  k = myMultiplyFunction(i, j); // Llamada a la Función; introducimos el valor de las variables i, j
  Serial.println(k);            //Al salir de la función, esta, carga la solución de su respuesta en k.
  delay(500);                   //Se muestra el resultado memorizado en la variable k en el Serial
Plotter
}                               // Esperamos medio segundo y se repite la función loop()

int myMultiplyFunction(int x, int y){   //Declaración de la Función
  int result;
  result = x * y;
  return result;   //Retornar a la Función loop() el resultado de la multiplicación de la
multiplicación
```

2. Funciones proporcionadas por ‘Arduino’

https://www.arduino.cc/reference/en/#variables



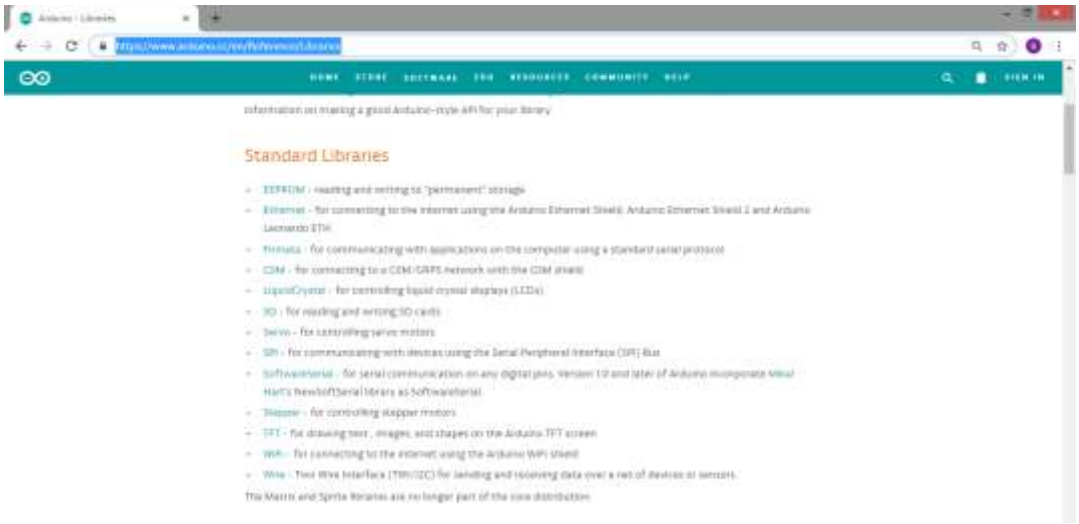
Las funciones en la lista superior ya vienen declaradas por el sistema. Son funciones propias del sistema, y no hay que añadir librerías:
Son las funciones **Time, digital i/O, analog i/O…**
Para otras de estas funciones hay que incluir la librería…por ejemplo **EEPROM**…¡¡Hay más!!
De las funciones de la lista superior, busca:

1. ¿Cuál es la función que genera un número aleatorio?
2. ¿Para qué sirve la función ‘pinMode’ de la librería de sistema **Digital i/O**?
3. El parámetro de entrada, entre paréntesis, de la función ‘delay()’ en la librería **Time**, ¿Se configura en segundos?
4. ¿Cuál es la función de la librería **Math** que eleva un número al cuadrado?
5. ¿Cuál es la función de la librería **Math** que eleva un número al cubo?
6. ¿Cuál es la función de la librería Digital i/O que lee el estado de un pin, de entrada o de salida, digital?
7. ¿Se puede utilizar una interrupción cómo temporizador?
8. ¿Qué es una interrupción exterior?

3. Librerías proporcionadas por ‘Arduino’

https://www.arduino.cc/en/Reference/Libraries
Hemos comentado como se incluía una librería en la cabecera del programa:
Menú ‘Programa’, incluir ‘Librería’…

Estudia las librerías y las funciones que forman parte de estas…
Muchas de estas librerías vienen definidas por el dispositivo ‘Arduino’ que se ha adquirido, y su definición viene detallada en las instrucciones del propio Hardware.
Otras son definidas como las **Standard Libraries**, etc…y son comunes a todos los sistemas, pero hay que definirlas en su uso.

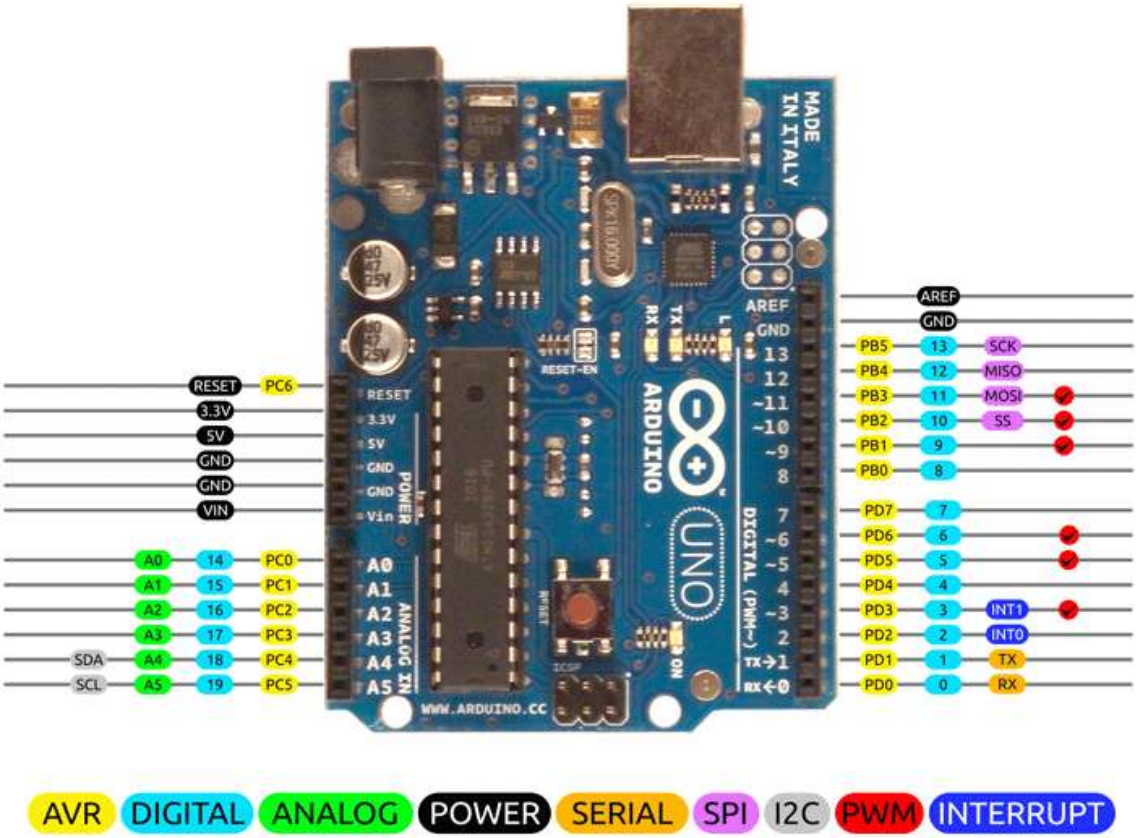


Aplicaciones Básicas de programación con ‘Arduino’

Definición Hardware ‘Arduino’. Pins de conexión

1. Módulo PcB programable ‘Arduino’ UNO. A000066

https://www.arduino.cc/en/main/arduinoBoardUno/
https://www.arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf
Esquemas placa ‘Arduino’ UNO



2014 by Bouni
Photo by Arduino.cc

Conexión pins Arduino UNO.
En esta Pcb tenemos:

1. Entradas digitales(19)
2. Entradas analógicas(5, que pueden ser digitales)
3. Comunicación I2C, pines A4(SDA) y A5.(SCL). En Arduino, la comunicación I2C requiere de un dispositivo MASTER, o maestro, y otro SLAVE o esclavo. El esclavo SOLO puede recibir información. No se la puede enviar al Máster. Los Ordenadores o PC utilizan este protocolo para comunicar los microprocesadores a la tarjeta gráfica, por ejemplo.

4. Comunicación SPI, o paralelo. Antiguamente se utilizaba para comunicar los ordenadores a las impresoras o algunos periféricos.
   La tecla 10 es SS
   La tecla 11 es MOSI
   La tecla 12 es MISO
   La tecla 13 es SCK.
   La tarjeta con LCD TFT A000096 a la que se refiere el Manual, se comunica mediante este protocolo. Arduino gestiona el protocolo mediante la librería.
5. Pines PWM. Pueden ser configuradas con señales cuadradas, de Duty(o tiempo ON) variable, que utilizamos habitualmente para controlar, por ejemplo, Mosfet, Drivers de motores DC, Paso a Paso, etc…Han sido identificadas en el gráfico con 6 señales en rojo, y un símbolo [~] en la tarjeta Pcb…
6. Las entradas ‘2’ y ‘3’ también pueden ser configuradas como entradas de Interrupción. Este tipo de entradas son checkeadas por la Pcb de forma independiente a la ejecución del resto del programa. Cuando se actúa sobre ellas, la Pcb, internamente, deja de leer y ejecutar el programa, y ejecuta solo el código fuente relativo a estas. Son las teclas F1, F2, F3…de nuestros ordenadores o laptops…

Módulos de entradas/salidas digitales

https://www.arduino.cc/en/Tutorial/BuiltInExamples
Link al Tutorial de ‘Arduino’ con todos los ejemplos, código de programación, y conexionado del Hardware a dispositivos electrónicos.

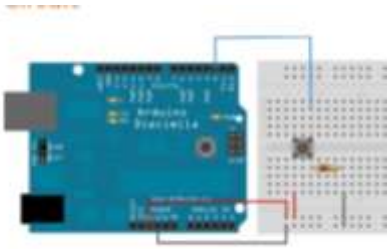
El primero de los ejemplos es el más fácil. El ejemplo **‘blink’** del tutorial te va a servir para empezar a conocer las funciones delay() de la librería TIMER, y la función digitalWrite() de la librería DIGITAL I/O. No hay que declarar ninguna de las dos librerías.
No has que conectar ningún elemento al ‘Arduino’

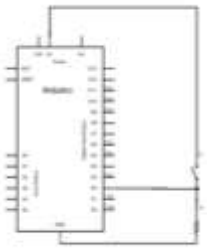
```
// la función setup()  se ejecuta solo una vez, al pulsar el pulsador reset que lleva incorporado 'Arduino'
o al //alimentar la Pcb
void setup() {
 // Inicializa el pin digital LED_BUILTIN como a una output o salida. Este LEd está incorporado en la
Pcb del //'Arduino'. Viene definido en el diagrama de la figura 1 con la etiqueta 'L'
  pinMode(LED_BUILTIN, OUTPUT);
}

// la función Loop() funcionará de forma indefinida
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // pon el LED a '1' (HIGH es el nivel de voltaje)
  delay(1000);                       // espera 1 segundo
  digitalWrite(LED_BUILTIN, LOW);    // pon el LED a '0' poniendo el nivel de voltaje a LOW
  delay(1000);                       // espera 1 segundo
}
```

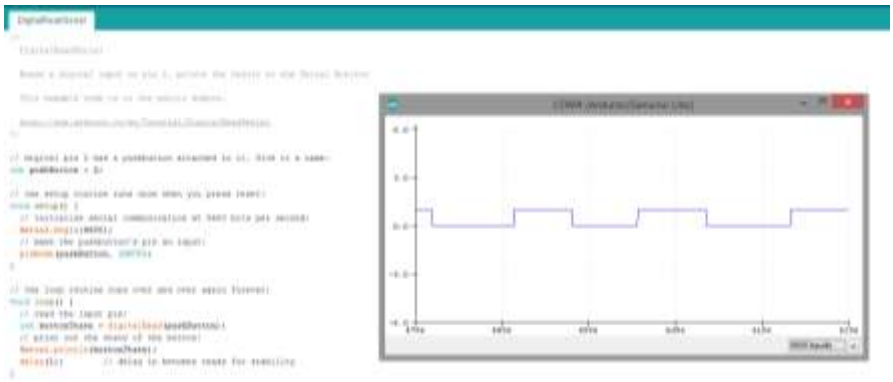
https://www.arduino.cc/en/Tutorial/DigitalReadSerial
Link al Tutorial de 'Arduino' con todos los ejemplos, código de programación, y conexionado del Hardware a dispositivos electrónicos.

Con el ejemplo **'DigitalReadSerial'** aprendes a leer el estado de un pulsador, conectado al Arduino', con una Resistencia de 10KΩ a masa o Gnd.

```
// Definimos como global una etiqueta para la entrada digital '2'
int pushButton = 2;
// la función setup()  se ejecuta solo una vez, al pulsar el pulsador reset que lleva incorporado 'Arduino'
o al //alimentar la Pcb
void setup() {
    // Inicializa la comunicación Serie a  9600 bits por segundo, o baudios
  Serial.begin(9600);
  // define el modo de entrada, INpuT en la entrada '2'
  pinMode(pushButton, INPUT);
}

// la función Loop() funcionarà de forma indefinida
void loop() {
  // Lee el estado de la entrada digital:
  int buttonState = digitalRead(pushButton);
  // imprime el estado de l'entrada digital '2' que has leído en el Serial Plotter o en el Monitor Serial
  Serial.println(buttonState);
  delay(1);        // espera 1 milisegundo  para estabilitzar la lectura de la entrada digital
}
```







https://www.arduino.cc/en/Tutorial/BlinkWithoutDelay
Link al Tutorial ‘Arduino con todos los ejemplos,, códigos de programación, y conexionada del hardware a dispositivos electrónicos.

Con el ejemplo **‘BlinkWhithoutDelay’**, aprenderás a apagar y encender un led, usando un pin digital de ‘Arduino’ pero sin utilizar la función delay() de la librería TIMER. Para ello usaremos la función:
Millis()
Entre el led el pin del micro controlador se ha ubicado una resistencia de 220 Ω.

```
//Definimos la etiqueta del pin que va a controlar el LED. Será el pin 2
const int ledPin =  2;

// El estado de la variable que encenderá y apagarà el LED se inicializa como LOW('0'):
int ledState = LOW;

// Siempre usaremos "unsigned long" para variables que definen un largo tiempo...
unsigned long previousMillis = 0;        // Y memorizaremos este tiempo en una variable

// Las constantes no cambiarán
const long interval = 1000;           // intervalo de tiempo entre blink (millisegundos) y el siguiente

void setup() {
  // Configura el pin digital com a sortida:
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // Aquí escribes tu código

  // Memorizas el tiempo actual
  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval) {
    // Memoritza el último tiempo en el que hiciste un blink
    previousMillis = currentMillis;

    // Haz el blink
    if (ledState == LOW) {                          //¡¡Fíjate de que es ==!!!
      ledState = HIGH;
    } else {
      ledState = LOW;
    }

    // Pon el LED al estado de la variable
    digitalWrite(ledPin, ledState);
  }
}
```

https://www.arduino.cc/en/Tutorial/Button

Link al Tutorial ‘Arduino con todos los ejemplos,, códigos de programación, y conexionada del hardware a dispositivos electrónicos.

Con el ejemplo **‘Button’**, aprenderás a apagar y encender un led, usando un pin digital de ‘Arduino’ para un pulsador(el ‘4’) y otro pin digital para el led(el ‘2’)

```
const int pulsador = 4;   //Creas las etiquetas para los pines de entrada y salida
const int LED = 2;

int EstatButo =0;         //Creas una variable que cambiará de estado cuando pulses

void setup() {
  // Esta parte del programa se ejecutará solo una vez, tres polsar el reset del ‘Arduino’ o al alimentar la PcB:
pinMode(pulsador, INPUT); //Configuras los pines que serán entrada y salida
pinMode(LED,OUTPUT);

}

void loop() {
          // Esta es la parte dels programa que se ejecuta de forma indefinida
        EstatButo = digitalRead(pulsador);  //Lees el estado del pulsador y lo guardas en la variable creada
        if(EstatButo == HIGH)     //Recuerda que comparas si '==', y que si solo es un '=' no comparas...
        {
                digitalWrite(LED,HIGH); //Escribes en el pin de salida, el estado HIGH o '1'
        }
        else
        {
                digitalWrite(LED,LOW); //Escribes en el pin de salida, el estado LOW o '0'
        }
```

Las conexiones de Hardware vienen definidas en los ejemplos:

https://www.arduino.cc/en/Tutorial/DigitalReadSerial
https://www.arduino.cc/en/Tutorial/Fade

https://www.arduino.cc/en/Tutorial/Debounce
Link al Tutorial ‘Arduino’ con todos los ejemplos, código de programación y conexionado del hardware a dispositivos electrónicos.

Con el ejemplo ‘Debounce’’ aprenderás a programar un pulsador para que al accionarlo, no tenga anti rebote, el estado de la entrada solo se lea una vez..
Para ello, no utilizaremos la función delay(), sino que utilizaremos la instrucción milli().

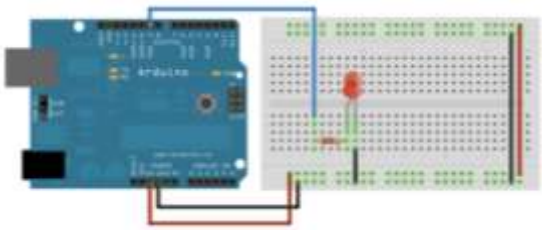
```
const int pulsador = 4;   //Creas las etiquetas para los pines de entrada y salida
const int LED = 2;

int EstatButo =0;        //Creas una variable que cambiará de estado cuando pulses

void setup() {
  // Esta parte del programa se ejecutará solo una vez, tres polsar el reset del 'Arduino' o al alimentar la PcB:
pinMode(pulsador, INPUT); //Configuras los pines que serán entrada y salida
pinMode(LED,OUTPUT);

}

void loop() {
          // Esta es la parte dels programa que se ejecuta de forma indefinida
        EstatButo = digitalRead(pulsador);  //Lees el estado del pulsador y lo guardas en la variable creada
        if(EstatButo == HIGH)     //Recuerda que comparas si '==', y que si solo es un '=' no comparas...
        {
                digitalWrite(LED,HIGH); //Escribes en el pin de salida, el estado HIGH o '1'
        }
        else
        {
                digitalWrite(LED,LOW); //Escribes en el pin de salida, el estado LOW o '0'
        }
```

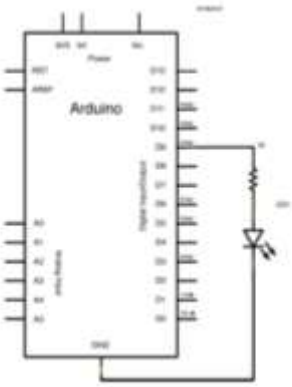
Módulos de entradas/salidas analógicas

https://www.arduino.cc/en/Tutorial/Fade

Link al Tutorial 'Arduino' con todos los ejemplos, código de programación y conexionado del hardware a dispositivos electrónicos.

Con el ejemplo **'Fade'**, aprenderás a apagar y encender un led progresivamente, usando un pin PWM de 'Arduino' como si se tratara de un pin analógico, con las instrucciones analógicas. El Led va conectado a una de las patillas PWM, la '9', y entre el led el pin del micro controlador se ha ubicado una resistencia de 220 Ω.





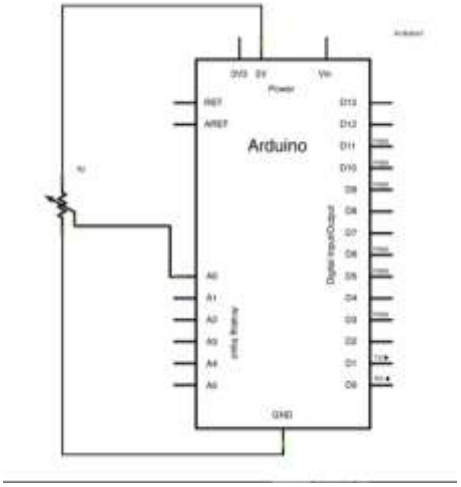
```
int led = 9;          // Definimos como global una etiqueta para la entrada PWM '9'
int brightness = 0;    // Como de brillante està el LED
int fadeAmount = 5;    // Los puntos de diferencia que habrá entre brightness y brightness

// la función setup()  se ejecuta solo una vez, al pulsar el pulsador reset que lleva incorporado 'Arduino'
o al //alimentar la Pcb
void setup() {
  // define el  pin '9' como salida
  pinMode(led, OUTPUT);
}

// la función Loop() funcionarà de forma indefinida
void loop() {
  // Escribe el nuevo valor de brightness en el pin '9'
  analogWrite(led, brightness);

  // Cambia el brightness para el próximo ciclo de ejecución
  brightness = brightness + fadeAmount;

  // Si llegas a los 255 puntos máximos, invertiremos el sentido del fade
  if (brightness <= 0 || brightness >= 255) {
    fadeAmount = -fadeAmount;
  }
  // Espera 30 milisegundos para estabilitzar la lectura del pin de entrada
  delay(30);
}
```

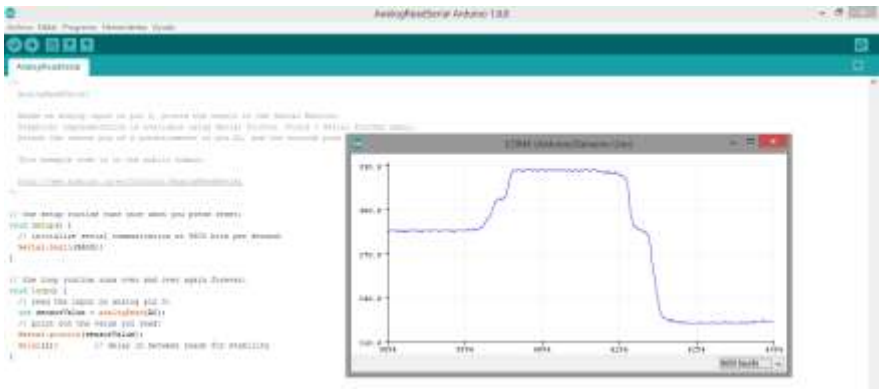
https://www.arduino.cc/en/Tutorial/AnalogReadSerial
Link al Tutorial de 'Arduino' con todos los ejemplos, código de programación, y conexionado del Hardware a dispositivos electrónicos.

Con el ejemplo ‘AnalogReadSerial’ aprendes a conectar un potenciómetro de 10KΩ a una de las entradas/salidas analógicas, etiquetada como ‘A0’ en la PcB. El ejemplo también proporciona el código fuente…

```
// la función setup()  se ejecuta solo una vez, al polsar el polsador reset que lleva incorporado ‘Arduino’
o al //alimentar la Pcb
void setup() {
    // Inicializa la comunicación Serie a  9600 bits por segundo, o baudios
  Serial.begin(9600);
}

// la función Loop() funcionarà de forma indefinida
void loop() {
  // lee el pin de la entrada analògica A0
  int sensorValue = analogRead(A0);
  // Imprime el valor que has leído de la entrada analògica A0 en el Serial Plotter o en el Monitor Serial
  Serial.println(sensorValue);
  delay(1);        // espera 1 milisegundo  para estabilitzar la lectura de la entrada analógica
}
```

Conexionado del Potenciómetro.





Demostración gráfica en el Serial Plotter.

Control de los pins PWM

¿ Qué es el PWM?
https://www.youtube.com/watch?v=Y1QraI5i_XM

1. Programar Leds IR infrarrojos a 33 Khz

   Este punto ha sido definido en el diseño de nuestra Sala de Actos controlada por Arduino.

2. Programar el control de motores DC por Módulos de control PWM

   Este punto ha sido definido en el diseño de nuestra Sala de Actos controlada por Arduino.

Librería TIMER

1. Función Delay()

   El delay detiene la ejecución del resto del programa hasta que ha pasado el período de tiempo.
   Es una función propia del sistema, a la que se llama aportándole el tiempo de espera en **milisegundos.**

2. Función millis()

   La función millis() es un temporizador que no interrumpe la ejecución del programa mientras espera.

   Este punto ha sido definido en el diseño de nuestra Sala de Actos controlada por Arduino.
   El Led indicador que notifica que un usuario está entrando en la sala está controlado con la función millis().

Librería SD.

Leer y grabar tarjeta de Memoria SD

Este punto ha sido definido en el diseño de nuestra Sala de Actos controlada por Arduino.

Pantalla LCD. **Modelo A000096**

https://www.arduino.cc/en/Guide/TFTtoBoards
Datasheet adjuntado al final del Tutorial
Esquema del conexionado de la pantalla TFT con 'Arduino' **UNO. LA PANTALLA SE CONECTA, Y POR LO TANTO SE PROGRAMA DIFERENTE PARA CADA MODELO DE 'ARDUINO'**

1. Escribir texto en la pantalla TFT

Este punto ha sido definido en el diseño de nuestra Sala de Actos controlada por Arduino.

2. Importar archivo.bmp desde tarjeta SD

Este punto ha sido definido en el diseño de nuestra Sala de Actos controlada por Arduino.

Librería Speaker

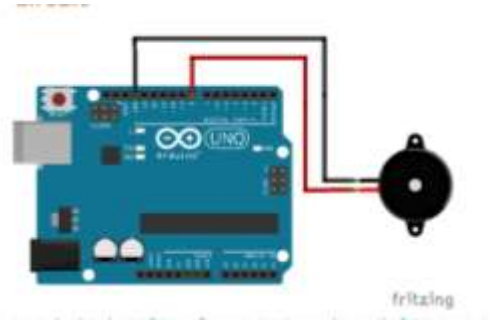
1. Enviar melodías al speaker

https://www.arduino.cc/en/Tutorial/ToneMelody
Link al Tutorial de'Arduino' con todos los ejemplos, código de programación, conexionado del Hardware a dispositivos electrónicos.

Con el ejemplo 'ToneMelody' aprendes a gestionar arrays, los bucles for() y a crear sonidos que se reproducen en un speaker piezoeléctrico simple, que has conectado directamente a

```
//Declaracion de las notas
/*************************************************
 * Public Constants
 *************************************************/

#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
//(…..)Las CONSTANTES DE TODAS LAS NOTAS ESTAN DEFINIDAS AL FINAL DE LA PÁGINA DEL Link
adjuntado
//NOTE_B0 es la etiqueta, 31 es el valor digital que le corresponde a cada nota. La tabla de definiciones es
inmensa…

int melody[] = {   //Declaración de la array de notas. Son 8 notas…
  NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4
};

// Array de la duración de las notas: 4 = un cuarto de tiempo, 8 = un octavo, etc.:
int noteDurations[] = {
  4, 8, 8, 4, 4, 4, 4, 4
};

void setup() {              //Solo lo ejecutaremos una vez
  // bucle para leer una por una todas las notas, con su duración con la función tone()
  for (int thisNote = 0; thisNote < 8; thisNote++) {

    // Para calcular la duración real de la nota cogemos un segundo(1000mSeg.) y lo dividimos por el tipo..

    int noteDuration = 1000 / noteDurations[thisNote];
    tone(8, melody[thisNote], noteDuration);

    // Para distinguir las notas, dejaremos un tiempo minimo entre ellas.
    // La duración de la nota y un 30%
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    // Para detener el tono que estaba sonando
    noTone(8);
  }
}

void loop() {
  // no need to repeat the melody.
}
```

la salida 8 del Pcb, y que gestionas con la instrucción tone()



Comunicaciones

Módulos comunicación Serial

Comunicación con el Puerto serie. Mostrar en Monitor serie cuando escribes una palabra en el teclado del ordenador…

https://www.youtube.com/watch?v=jrFQlspRgWw

Módulos de comunicación I^2C

Definición muy clara del procedimiento a seguir en una comunicación Máster-Slave con I^2C
https://programarfacil.com/blog/arduino-blog/conectar-dos-arduinos-i2c/

Este punto ha sido definido en el diseño de nuestra Sala de Actos controlada por Arduino.
Se encuentra definida también en detalle en el código fuente de la aplicación adjunta:

ProjecteMaquetaAula.ino
ProjecteMaquetaAulaSlave.ino

ANEXO5: Datasheets de dispositivos electrónicos utilizados

Los Datasheets de los dispositivos electrónicos utilizados en la Maqueta a escala de la Sala de Actos vienen adjuntos en una carpeta winrar comprimida ‘**Anexo4Datasheets’**

ANEXO6: Código fuente de la aplicación

Los códigos fuente utilizados en la Maqueta a escala de la Sala de Actos vienen adjuntos en una carpeta winrar comprimida ‘**CodigoFuenteAplicacion’**

En esta carpeta encontraremos los programas:

ProjecteMaquetaAula.ino
ProjecteMaquetaAulaSlave.ino

ANEXO7: Documentación complementaria para los docentes y bibliografía

Bibliografía propia al reto PBL y programación formativa.

Libros:

Marta Orts Alís. *L'aprenentatge basat en problemes (ABP): De la teoria a la pràctica: una experiència amb un grup nombrós d'estudiants*. Barcelona, 2011. ISBN: 978-84-9980-200-8.

Hernández-Fernández, A. (coord.), Luján de Francisco, J., Muntada, M., Muñoz, I. y Siscart, B. (2015). *Tecnología, programación y robótica 3.* Barcelona: Editorial Casals. ISBN: 978-84-218-6026-7.

Tim Jackson. *Prosperity without growth. Economics for a Finite Planet.* UK y USA, 2009. ISBN: 978-1-84977-000-2.

Manuel Serrano Ortega, Pilar Blázquez Ceballos. ESIC, 2015. *Design thinking: Lidera el presente. Crea el futuro*: https://books.google.es/munari

Iniciación a la programación. Lógica y diseño. Joyce Farrel. Editorial Paraninfo. 1999

Artículos:

Carme Arpí Miró et al. *El ABP: origen, modelos y técnicas afines. Revista GRAO: Aula de Innovación Educativa | núm. 216 | noviembre 2012. https://dugi-doc.udg.edu/handle/10256/8680?show=full*

Henk G Schmidt Jerome I Rotgans Elaine HJ Yew. *The process of problem- based learning: what works and why | Medical education | Volumen 45 | agosto 2011. https://doi.org/10.1111/j.1365-2923.2011.04035.x*

Apuntes:

Antoni Hernández-Fernández. *Innovació docent i iniciació a la recerca educativa.* Apuntes sesión 1,2,3,5,9. [en línea] https://atenea.upc.edu/pluginfile.php/2748481/mod_resource/content/1/sessio3_innovacio1819_FP.pdf

Web:

UPC. *Citar i elaborar la bibliografia* [en línea] Versió online. Data de la darrera actualització: 27 de març de 2019. Barcelona.
[consulta: 12-04-2019] Disponible a:
https://bibliotecnica.upc.edu/investigadors/citar-elaborar-bibliografia

Bruno Munari. *Linealidad "Cómo nacen los objetos". Editorial GG_Diseño. Disponible a: https://alzado.org/articulo.php?id_art=354*

Departament d'Ensenyament. [en línea] *DECRET 125/2012, de 9 d'octubre. Currículum del cicle formatiu de grau mitjà d'instal·lacions elèctriques i automàtiques.* [consulta: 1-05-2019] Disponible a: http://xtec.gencat.cat/web/.content/alfresco/d/d/workspace/SpacesStore/0038/215fe7f1-05a9-484e-9f78-ff04dc86f387/DOGC_T_instal_electriques_automatiques.pdf

- Comunidad 'Arduino'. Fórum de la comunidad Arduino al que el alumno puede acudir para hacer preguntas.

https://playground.arduino.cc/uploads/Code/FSM.zip

- Recursos sobre innovación PBL:
  - http://xtec.gencat.cat/ca/innovacio/cesire/
  - http://srvcnpbs.xtec.cat/cdec/
- Software de simulación de circuitos **OrCad PsPice. LA versión educativa es gratuita.**
  - https://www.pspice.com/academic-user