



**Universidad**  
Zaragoza

## Proyecto Fin de Carrera

Integración de los contenidos de la asignatura de  
tecnología de 4ºESO para ser impartidos con Arduino y  
S4A

Autor/es

Juan Fco Sangüesa Ortiz

Director/es y/o ponente

Antonio Montañes Espinosa

Facultad / Escuela

Escuela de Ingeniería y Arquitectura

Año

2017

## INTEGRACIÓN DE LOS CONTENIDOS DE LA ASIGNATURA DE TECNOLOGÍA DE 4º E.S.O. PARA SER IMPARTIDOS CON ARDUINO Y S4A

### RESUMEN

El objetivo del proyecto es utilizar una de las placas Arduino, concretamente la Arduino Uno Rev3, como controladora junto con el lenguaje de programación S4A, basado en Scratch, para facilitar al profesorado de la asignatura de tecnología de 4º ESO, herramientas para impartir de forma integrada y amena para los alumnos, los conocimientos, las habilidades, destrezas y actitudes relacionados con la electricidad, electrónica, control y robótica.

Partiendo del Real Decreto de enseñanzas mínimas y de la orden por la que se aprueba el currículo aragonés para enseñanza secundaria, integrar los contenidos anteriormente citados, trabajar las competencias clave y llegar a proporcionar a los docentes de la especialidad de tecnología una serie de herramientas que les ayuden a impartir la materia.

Las herramientas que se pretende proporcionar son:

- Manuales de uso de Arduino, Scratch y S4A.
- Listado de material necesario y presupuesto.
- Colección de prácticas para trabajar en el aula taller.
- Colección de proyectos para proponer al alumnado.

Otro objetivo es fomentar entre el alumnado el uso de hardware y software libre.

El objetivo final es que los alumnos se aproximen lo máximo posible a los objetivos de la materia al acabar la etapa de la ESO y que adquieran un grado de desarrollo de las competencias clave lo más alto posible, con el fin de que puedan continuar sus estudios con solvencia en ciclos formativos de grado medio, grado superior, bachillerato y universidad.

Conforme se desarrolle el proyecto, se va a ir poniendo en práctica en el aula.

## ÍNDICE

1. INTRODUCCIÓN .....	3
2. ¿QUÉ ES ARDUINO? .....	6
3. ¿POR QUÉ ARDUINO? .....	7
4. ELECCIÓN DE LA PLACA ARDUINO .....	8
5. ¿POR QUÉ NO USAR EL LENGUAJE DE PROGRAMACIÓN ARDUINO APL? .....	10
6. LENGUAJES GRÁFICOS DE PROGRAMACIÓN .....	11
7. ELECCIÓN DEL LENGUAJE GRÁFICO DE PROGRAMACIÓN .....	11
8. SCRATCH .....	12
9. S4A .....	13
10. RESULTADOS OBTENIDOS .....	16
11. CONCLUSIONES .....	20
12. BIBLIOGRAFÍA Y WEBGRAFÍA .....	22
13. ÍNDICE DE FIGURAS .....	23
14. ÍNDICE DE TABLAS .....	23
ANEXO I: GLOSARIO DE BLOQUES DE S4A A USAR EN EL CUADERNO DE PRÁCTICAS	24
ANEXO II: CUADERNO CON COLECCIÓN DE PRÁCTICAS .....	29
ANEXO III: COLECCIÓN DE PROPUESTAS DE PROYECTO .....	175
ANEXO IV: LISTA DE MATERIALES Y PRESUPUESTO .....	181
ANEXO V: PRÁCTICA COMPLETA A DESARROLLAR EN LA EXPOSICIÓN .....	183

## 1. INTRODUCCIÓN

Desde el año 2002 soy profesor de la materia de Tecnología para la etapa de E.S.O. y Bachillerato. Desde el año 2003 tengo destino definitivo en el I.E.S. Damián Forment de Alcorisa, donde sólo se imparte la etapa de E.S.O., además de un programa de F.P. Básica en la modalidad de operario de carpintería y mueble y un ciclo de grado medio de atención a personas en situación de dependencia.

En cuanto a la materia de tecnología, desde mi primer año de trabajo en la enseñanza, observé la forma en la que los alumnos van evolucionando en sus conocimientos, destrezas, habilidades y actitudes. Nosotros, los profesores, somos los encargados de guiar este proceso, ayudados por los reales decretos de enseñanzas mínimas y currículos. Por desgracia, todo este proceso está claramente influenciado por cambios políticos y económicos.

Por suerte para la materia de Tecnología, todo este proceso se ve favorecido por los grandes avances tecnológicos que se producen constantemente. Esto ha hecho, que todos estos avances aplicados a la forma de impartir la asignatura, hagan más fácil y motivador el aprendizaje de los alumnos.

La experiencia me dice, que cuando los alumnos llegan a 4º E.S.O., tienen las herramientas para poder avanzar tal y como indica el currículo de la materia para ese nivel, con contenidos de electricidad, electrónica, control, robótica, neumática e informática (ahora llamada tecnologías de la información y de la comunicación T.I.C.). Pero hasta no hace mucho tiempo, nos encontrábamos con una limitación muy importante, la limitación económica. Los equipos y materiales para impartir todos estos contenidos de forma lo más práctica posible, eran demasiado costosos, lo que hacía inviable cualquier intento de compra.

La aparición y rápido desarrollo del hardware y software libre, ha abierto una importante ventana a nuestra materia, pasando de equipos desarrollados por los pocos fabricantes que se dedicaban a equipamientos didácticos, cuyo coste era inasumible para los centros pequeños como en el que trabajo, a tener en internet una enorme oferta de equipos, kits y complementos basados en hardware y software libre, a unos precios muy competitivos, que casi cualquier centro se puede permitir, y con una flexibilidad de uso que para nada dan los kits que desarrollan las empresas de equipamientos didácticos.

Todo esto me llevó a plantearme, hace 5 años, la elección de un hardware libre para utilizar como controladora, la elección de un software libre para programar la controladora de forma sencilla para el alumnado de esta etapa educativa y la redacción de un cuaderno de prácticas guiadas que permitieran trabajar de forma integrada los contenidos de electricidad, electrónica, control y robótica.

En cuanto al hardware no tuve ninguna duda nada más conocer Arduino. Cumplía con los requisitos que me había planteado. Es versátil, manejable, sencillo de controlar y económico.

La parte de software fue un poco más complicada. Lo que tenía claro era que descartaba el software Arduino IDE y su lenguaje de programación Arduino APL. Tenía que recurrir a un software sencillo de utilizar para el alumnado. Un software para programar de forma gráfica. Me fue de mucha ayuda en la elección, un cuaderno de José Manuel Ruiz Gutiérrez, llamado “Entornos Gráficos para la Programación, Plataforma Open Hardware Arduino”<sup>3</sup>. Dentro de los diferentes entornos que se planteaban en este cuaderno, me llamó poderosamente la atención Scratch y su adaptación S4A para poder controlar una placa Arduino Uno. La principal ventaja es que desde 2º E.S.O., los alumnos pueden empezar a trabajar con Scratch y cuando llegan a 4º E.S.O., ya conocen la forma de trabajar y entonces se pasa a utilizar S4A para programar la placa Arduino Uno. El principal inconveniente es que S4A es un entorno de programación esclavo, es decir, la placa Arduino Uno tiene que estar conectada de forma permanente al ordenador mediante el puerto USB para su funcionamiento. Este inconveniente nos da también la ventaja de que al estar permanente conectados al ordenador, podemos monitorizar los procesos.

Dentro de los entornos de programación gráfica autónomos (aquellos en los que no se necesita estar conectado a Arduino para poder ejecutar el programa, solamente tienes que estar conectado para transferirlo) que se describen en el cuaderno de José Manuel Ruiz Gutiérrez, me llamó la atención Ardublock, por su sencilla integración con el IDE Arduino, fácil programación, muy similar a S4A y fácil generación del código a transferir a la placa Arduino Uno.

Desde el curso 11/12, el proyecto se ha ido poniendo en práctica en el aula. Inicialmente se comenzó comprando 4 placas Arduino y utilizando el material del que se dispone en un taller de tecnología (diodos, resistencias, pulsadores, transistores). Las explicaciones se daban a los alumnos mediante la pizarra de tiza y esquemas eléctricos que los alumnos tenían que copiar en el cuaderno.

Curso a curso se ha ido avanzando, mejorando la composición, organización y número de prácticas del cuaderno, la creación de esquemas eléctricos y de conexionado a través de formato informático (trasmitidos a los alumnos mediante proyector, fotocopias y página web) y creados con software libre, adquisición de más material (más placas Arduino Uno, diodos led bicolor, diodos led RGB, displays de 7 segmentos, servomotores de 180º y de rotación continua, drivers para control de motores de corriente continua,...)

Durante el curso 15/16 participamos con los alumnos de 4º E.S.O. en el taller básico de construcción y puesta en funcionamiento de una impresora 3D a partir de un kit. Este taller estuvo organizado por el vicerrectorado de divulgación científica de la Facultad de Ciencias de la Universidad de Zaragoza. En este taller se pusieron en práctica todos los conocimientos adquiridos durante toda la etapa de la E.S.O. El centro consiguió el primer premio en la jornada de presentación de proyectos y clausura del taller.

La participación en este taller supuso la puesta en práctica de la totalidad de los contenidos de los que se compone este proyecto, permitiendo incluso ampliarlo al control de motores paso a paso, con los que diseñó alguna de las últimas prácticas.

También se diseñó una práctica para combinar Arduino con electrónica digital, en la que utilizando 4 salidas digitales se programa con S4A una cuenta de 0 a 9 en binario y utilizando un decodificador BCD a 7 segmentos, se visualiza la cuenta en un display.

Durante este curso 16/17 se va a acabar de implantar totalmente el proyecto, con una parte final del curso donde a los alumnos se les dará una propuesta de proyecto con un problema a solucionar y deberán realizar el diseño en 3D de la maqueta que resuelva el problema, la impresión de las piezas necesarias, la construcción y montaje, el cableado, la programación y puesta en funcionamiento utilizando diferentes tecnologías (servomotores, motores de corriente continua y tecnología neumática).

Hay que hacer notar que durante el curso pasado y durante el actual, tanto los alumnos como el profesor, estamos realizando todas las semanas 2 horas de actividades extraescolares, en horario de tarde, para poder realizar todas las tareas que se han comentado.

A continuación, en los diferentes apartados que forman esta memoria, se va a comentar brevemente qué es Arduino y la situación actual del proyecto. Posteriormente pasaremos a ver los motivos que nos han llevado a elegir Arduino como hardware de este proyecto.

Una vez elegido el hardware, el siguiente paso que se va a detallar es la elección del modelo concreto de placa Arduino. Los motivos para la no elección del Arduino IDE y Arduino APL para programar la placa y la búsqueda de un lenguaje gráfico de programación para programar de forma sencilla Arduino Uno.

Una vez elegido el lenguaje de programación, comentaremos en qué consiste Scratch y pasaremos a explicar S4A que es el software elegido, justificando de forma muy clara esta elección.

Seguidamente y como cuerpo principal de esta memoria, mostraremos un ejemplo de los resultados obtenidos, es decir, una de las prácticas del cuaderno de prácticas, desarrollada con detalle para que se pueda ver claramente la forma en la que se ha trabajado.

En el apartado de conclusiones se valorará el cumplimiento de los objetivos planteados, las posibilidades de continuación, ampliación o mejora del proyecto, las incidencias encontradas en el desarrollo del proyecto y una valoración crítica del conjunto del proyecto.

El siguiente apartado con la bibliografía y webgrafía utilizada.

Y por último los resultados obtenidos que aparecerán en su totalidad en los Anexos, haciendo especial hincapié en el Anexo II: cuaderno con colección de prácticas, ya que es el núcleo central de este proyecto.

## 2. ¿QUÉ ES ARDUINO?

Arduino es una plataforma de electrónica abierta para la creación de prototipos, basada en hardware y software libres, flexibles y fáciles de usar.

Todo surgió en el año 2003, cuando Massimo Banzi, profesor del instituto Ivrea (Italia), desarrolló una herramienta de programación de microcontroladores. Esta herramienta y el desarrollo de Processing (lenguaje de programación basado en Java) sirvieron como base al estudiante colombiano Hernando Barragán, para desarrollar la tarjeta Wiring y el lenguaje de programación.

Basándose en Wiring, Massimo Banzi, el investigador del instituto David Cuartielles y el desarrollador de hardware Gianluca Martino desarrollaron una tarjeta más pequeña y económica a la que llamaron Arduino.

La unión al grupo de trabajo de los estudiantes Mellis y Zambetti impulsó la construcción de una tarjeta básica y un ambiente de desarrollo completo.

En el año 2005 se une a este equipo de trabajo Tom Igoe, quien se encarga de las pruebas del sistema con estudiantes estadounidenses y de la distribución de la tarjeta en territorio americano.

Arduino puede tomar información del entorno a través de sus pines de entrada de toda una gama de sensores, captadores o accionadores y puede actuar sobre aquello que le rodea, controlando órganos de mando, actuadores o directamente receptores de muy pequeña potencia. El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino APL (basado en Wiring) y el entorno de desarrollo integrado Arduino IDE (basado en Processing).

En 2009, Massimo Banzi, David Cuartielles, David Mellis, Tom Igoe y Gianluca Martino fundaron Arduino LLC (<http://www.arduino.cc/>)<sup>1</sup>. Esta empresa se encarga del desarrollo de las placas Arduino, de la gestión de la comunidad y de la gestión de los proyectos libres que estén relacionados con Arduino. Arduino LLC quería permitir que otras empresas pudieran desarrollar, fabricar y distribuir placas originales de Arduino, ya fueran placas ya existentes o nuevas placas desarrolladas por estas empresas, que permitieran hacer crecer el proyecto Arduino.

Gianluca Martino, disconforme con esta política de apertura a otras empresas, fundó Arduino Srl (<http://www.arduino.org/>)<sup>2</sup> y registró la marca Arduino en más de 40 países, fundamentalmente de Europa y Asia.

Esto hace que Arduino LLC fabrique bajo la marca Arduino en EEUU, que es donde tiene registrada la marca, y bajo la marca Genuino en el resto del mundo. Actualmente, la única diferencia entre Arduino y Genuino es el lugar dónde se fabrica la placa. En un futuro, con el desarrollo de nuevas placas, la diferencia estará en el equipo que las haya desarrollado. En la figura 1 podemos ver el logotipo de ambas marcas.



Figura 1: logotipo de Arduino y Genuino.

Las placas se pueden construir manualmente o comprarse de fábrica; el software se puede descargar de forma gratuita. Los ficheros de diseño de referencia (CAD) están disponibles bajo una licencia abierta (open-source), así que se pueden adaptar de forma libre a nuestras necesidades.

### 3. ¿POR QUÉ ARDUINO?

Hay otros muchos microcontroladores disponibles para realizar la misma función que Arduino (Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, ...) Todas estas herramientas toman los desordenados y complejos detalles de la programación de un microcontrolador y la encierran en un paquete más o menos fácil de usar. Arduino simplifica el proceso de trabajo con microcontroladores, pero además ofrece algunas ventajas para cualquier usuario, al compararlo con el resto de sistemas:

- Es barato: las placas Arduino son relativamente baratas comparadas con otras plataformas microcontroladoras. La versión más barata del módulo Arduino puede ser ensamblada a mano, e incluso los módulos de Arduino ya ensamblados cuestan sobre 20€. Y actualmente, con la competencia del mercado chino, podemos encontrarlas a precios mucho más baratos.
- Multiplataforma: el software del entorno de desarrollo integrado de Arduino IDE se ejecuta en sistemas operativos Windows, Macintosh OSX y GNU/Linux a diferencia de la mayoría de los sistemas microcontroladores que están limitados a Windows.
- Entorno de programación simple y claro: el entorno de programación de Arduino es fácil de usar para principiantes, pero suficientemente flexible para que pueda ser utilizado por usuarios avanzados. En este punto se centrará este proyecto, ya que considero que para alumnos de 15-16 años, el entorno de programación no es simple, e intentaremos seleccionar un entorno de programación gráfico.
- Código abierto y hardware extensible: Arduino está basado en microcontroladores ATMEGA8 y ATMEGA168 de Atmel. Los planos para los módulos están publicados bajo licencia Creative Commons, por lo que diseñadores experimentados de circuitos pueden hacer su propia versión del módulo, ampliándolo y mejorándolo. Incluso usuarios relativamente inexpertos pueden construir la versión de la placa del módulo para entender cómo funciona y ahorrar dinero.
- Código abierto y software extensible: el software Arduino está publicado como herramientas de código abierto (open-source), disponible para la modificación y adaptación por programadores experimentados. El lenguaje puede ser expandido mediante librerías C++, y la gente que quiera entender los detalles técnicos pueden hacer el salto desde Arduino a la programación en lenguaje AVR C en el cual está basado.



## 4. ELECCIÓN DE LA PLACA ARDUINO

Para la elección de la placa Arduino a usar en el proyecto, aunque ya se tenía bastante clara la elección, hubo que recurrir a un proceso de recopilar y analizar información sobre las diferentes placas que Arduino ha desarrollado. Para ello se recurrió a sacar esa información de las páginas web de las dos empresas en las que actualmente se encuentra dividido el proyecto.

En estas páginas la información es bastante clara, está bien detallada y es totalmente fiable, al ser información proporcionada por los desarrolladores del proyecto. A continuación se proporcionan los enlaces a la sección de productos de las páginas de ambos desarrolladores:

- <https://www.arduino.cc/en/Main/Products?from=Main.GenuinoProducts> <sup>1</sup>
- <http://www.arduino.org/products/boards> <sup>2</sup>

En la figura 2, extraída de una de las páginas, podemos ver el conjunto de todas las placas que se han desarrollado hasta el momento:

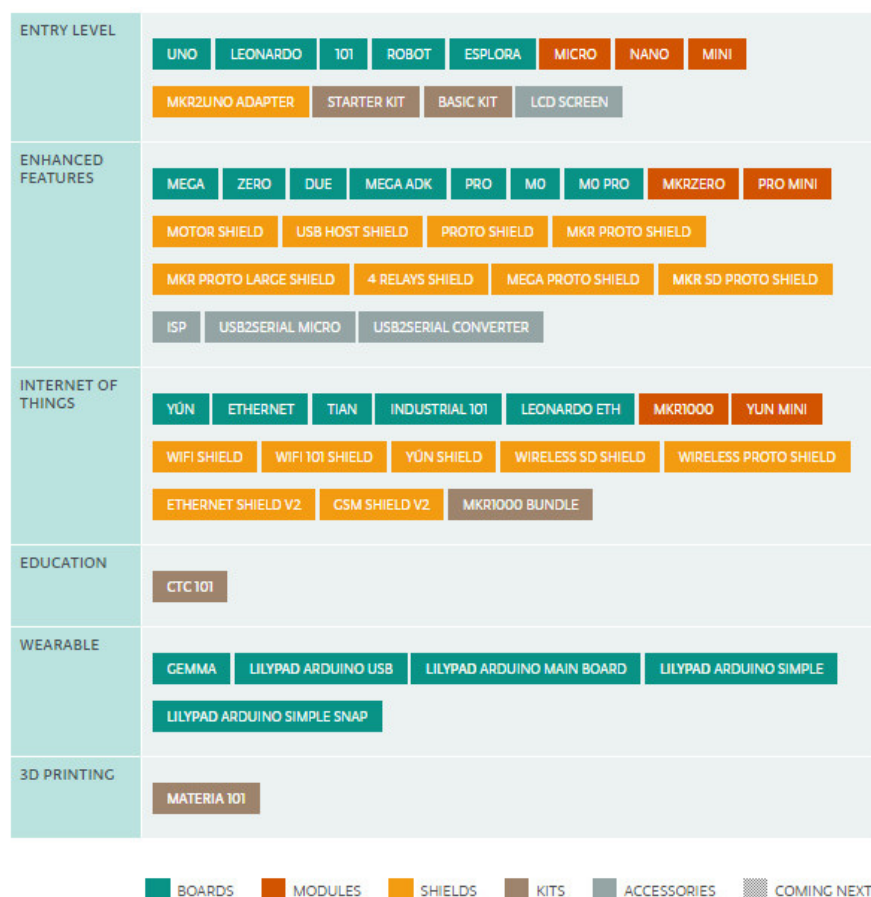


Figura 2: clasificación del conjunto de placas desarrolladas por Arduino.

Dentro de los diferentes modelos de placas Arduino ya ensamblados y atendiendo a las características que nos ofrecen, necesidades de nuestro proyecto, compatibilidad con el software de programación que vamos a usar y coste de la placa, elijo la Placa Arduino Uno Rev 3, cuyas características detalladas podemos ver en los siguientes enlaces:



## 5. ¿POR QUÉ NO USAR EL LENGUAJE DE PROGRAMACIÓN ARDUINO APL?

El microcontrolador de la placa Arduino se programa mediante el lenguaje de programación Arduino APL (Arduino Programming Language basado en Wiring) y el entorno de desarrollo integrado Arduino IDE (Integrated Development Environment basado en Processing). Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de estar conectado a un ordenador, solamente es necesaria la conexión para transferir el programa, si bien existe la posibilidad de estar conectado y comunicar con diferentes tipos de software.

El lenguaje de programación Arduino está basado en C y soporta todas las funciones del estándar de programación C y algunas funciones de C++. En el siguiente enlace podemos ver el glosario de las diferentes instrucciones de programación en lenguaje Arduino: <https://www.arduino.cc/en/Reference/HomePage><sup>1</sup>

Como ventaja, se ha comentado que el entorno de desarrollo de Arduino es fácil de usar para principiantes, y no lo negamos, como podemos observar en la figura 4, el Arduino IDE es sencillo de utilizar. Cualquier alumno de 12 años, tras una pequeña explicación, sería capaz de transferir un programa a una placa Arduino y ponerlo en funcionamiento. El problema es la confección del programa con APL, ya que sinceramente, no veo a mis alumnos de 4º ESO, con edades comprendidas entre los 15 y 17 años, programando en C. Es más, pienso que la complicación que esto supone perjudicaría los fines que queremos conseguir y desmotivaría al alumnado.

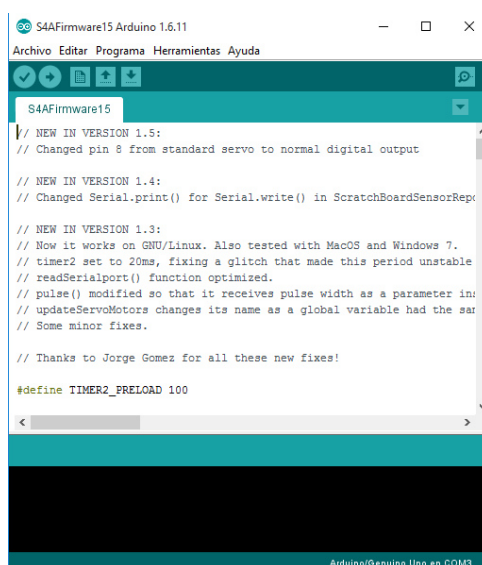


Figura 4: entorno gráfico del Arduino IDE.

Se podría empezar a trabajar en lenguaje Arduino en 2º de Bachillerato, en la asignatura TIC 2 y continuar a nivel universitario.

Por lo tanto, la idea es utilizar el hardware Arduino y buscar un entorno de programación que sea compatible, sencillo de comprender y de usar, e incluso que permita introducirlo a edades más tempranas, para ir evolucionando durante toda la etapa de secundaria.

## 6. LENGUAJES GRÁFICOS DE PROGRAMACIÓN

Con la inestimable ayuda del cuaderno de José Manuel Ruiz Gutiérrez, “Entornos Gráficos para la Programación, Plataforma Open Hardware Arduino”, al que podemos acceder desde el siguiente enlace: <https://www.dropbox.com/s/aywqux80kmigqsn/Programacion%20Grafica%20de%20Arduino.pdf><sup>3</sup>, vemos que los entornos gráficos de programación de Arduino se pueden clasificar en entornos de programación esclavos, que son aquellos que para poder ejecutar el programa, es necesario estar conectado a la placa Arduino y entornos de programación autónomos, que son aquellos en los que para ejecutar el programa no es necesario estar conectado al ordenador, solo es necesario estar conectado para transferir el programa desde el ordenador a la placa Arduino.

El funcionamiento de los lenguajes gráficos de programación autónomos, consiste en programar de forma gráfica mediante bloques, y una vez terminado el programa tenemos la opción de generar el código del programa en el lenguaje de Arduino, y bien mediante la interface del propio entorno de programación, o bien mediante el IDE Arduino, transferirlo a la placa. Estos entornos de programación no suelen ser reversibles, es decir, no pueden transformar un programa hecho en lenguaje Arduino, en el programa en lenguaje de bloques. Ejemplos de este tipo de lenguajes son: Minibloq, Ardublock, Amici, Modkit, VirtualBreadBoard,...

En los lenguajes de programación de Arduino en entornos esclavos, también realizamos el programa mediante bloques y para ejecutar el programa, tienen el inconveniente de tener que estar conectados continuamente al ordenador mediante un cable USB, lo que puede resultar bastante latoso cuando se trabaja con dispositivos o maquetas móviles. Por el contrario aportan una gran ventaja, y es que al estar conectado continuamente al ordenador, vamos a poder monitorizar en la pantalla del ordenador, todo lo que ocurre en el dispositivo o maqueta que estamos controlando. Etoys, S4A, Labview, Firefly, MyOpenLab son ejemplos de este tipo de lenguajes.

Últimamente han surgido multitud de entornos gráficos de programación para Arduino de ambos tipos (MindPlus, Snap, Calico, Blockly, Visualino, 12Blocks, Mixly, EasyArduino). Queda pendiente el probar todos estos entornos surgidos en los dos últimos años.

En el blog de José Manuel Ruiz Gutierrez podemos encontrar información muy completa y detallada sobre programación de Arduino con entornos gráficos: [http://josemanuelruizgutierrez.blogspot.com/es/](http://josemanuelruizgutierrez.blogspot.com.es/)<sup>4</sup>.

## 7. ELECCIÓN DEL LENGUAJE GRÁFICO DE PROGRAMACIÓN

El siguiente paso en el desarrollo de este proyecto fue probar todos los lenguajes gráficos de programación vistos en el apartado anterior y elegir aquel que se ha adoptado para trabajar en el aula con los alumnos y confeccionar este proyecto.

De todos ellos, el que más me llamó la atención fue Scratch y su adaptación S4A para Arduino. Los principales motivos son los siguientes:

- Con Scratch se podría empezar a trabajar en 2ºESO, de forma que los alumnos cuando lleguen a 4ºESO ya conozcan el entorno y la forma de trabajar.
- Es software multiplataforma, es decir, podemos usarlo en diferentes sistemas operativos.
- El entorno de programación y la forma de trabajar son muy sencillos.
- Es software libre, por lo que los alumnos lo pueden instalar y usar en sus ordenadores de casa.
- No se necesita un ordenador muy potente y sobre todo, funciona en los miniordenadores que tenemos en el centro.
- En las pruebas realizadas, el funcionamiento ha sido totalmente correcto.

Como inconveniente nos encontramos con que es un entorno gráfico de programación esclavo, es decir, para ejecutar el programa tenemos que tener conectado el ordenador a la placa Arduino. Aunque este inconveniente lo vamos a transformar en una ventaja, ya que al tener continuamente conectada la placa al ordenador, vamos a poder monitorizar las diferentes prácticas y proyectos.

## 8. SCRATCH

Programar ordenadores, controladoras u otros sistemas puede llegar a ser maravilloso y divertido, pero no utilizando los lenguajes clásicos, sino utilizando lenguajes gráficos de programación. Al hacerlo, pasamos de utilizar programas o juegos de ordenador elaborados por otras personas a nuestros propios programas.

Scratch es un entorno de programación, desarrollado a partir del año 2003, por un grupo de investigadores del Instituto Tecnológico de Massachusetts (MIT), una de las Universidades más importantes del mundo, bajo la dirección del Dr. Mitchel Resnick.

Cuando utilizamos el entorno de programación Scratch, aprendemos a crear, manejar e integrar textos, imágenes y grabaciones de audio. Además, al tiempo que nos divertimos, podemos realizar actividades de programación de ordenadores que nos ayuden a mejorar nuestra comprensión de diferentes temas interdisciplinares de Matemáticas, Ciencias Naturales, Ciencias Sociales, Lenguaje, etc.

Scratch hace que la programación sea más divertida para todo aquel que se enfrente por primera vez a aprender a programar. Según sus creadores, fue diseñado como medio de expresión para ayudar a niños y jóvenes a expresar sus ideas de forma creativa, al tiempo que desarrollan habilidades de pensamiento lógico y de aprendizaje del Siglo XXI.

La gramática de Scratch se basa en un conjunto de “bloques gráficos de programación”, clasificados por categorías, que se pueden ensamblar para crear programas. Tal como con las fichas de LEGO, conectores en los bloques sugieren de qué manera se pueden ensamblar. Comenzar simplemente a arrastrar bloques al área de programas y experimentar con ellos, ensamblándolos en diferente orden y observando qué pasa.

En la figura 5 podemos observar la interface del programa.

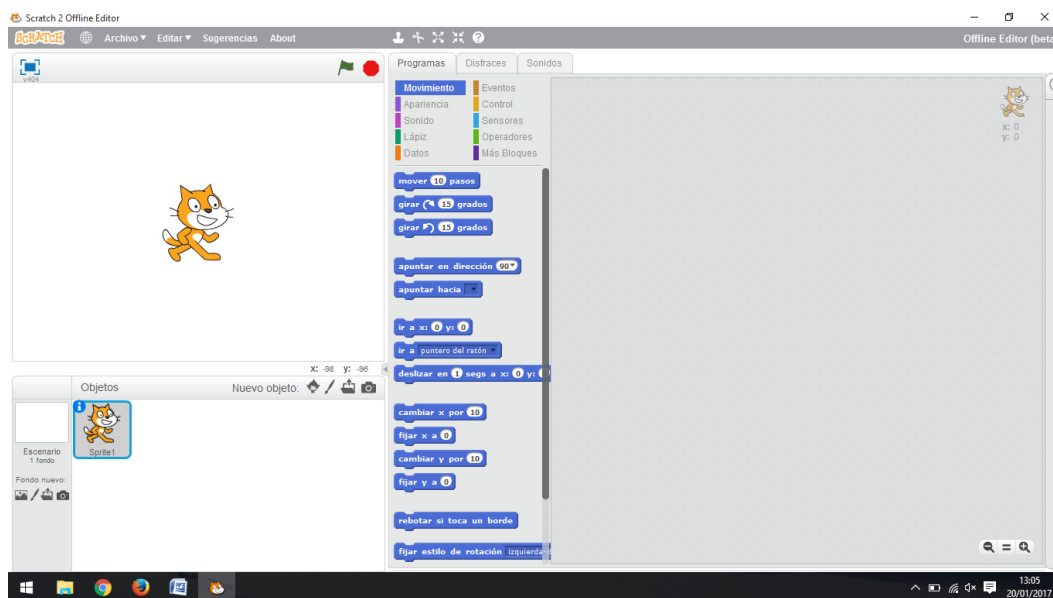


Figura 5: interface gráfica de Scratch.

Desde el siguiente enlace podemos acceder a la web oficial de Scratch: <https://scratch.mit.edu/><sup>5</sup>

Aquí podemos encontrar una guía de referencia de Scratch: <http://eduteka.icesi.edu.co/pdfdir/ScratchGuiaReferencia.pdf><sup>6</sup>

Y un cuaderno de trabajo y una guía para el profesorado: <http://eduteka.icesi.edu.co/articulos/ScratchCuadernoTrabajo1><sup>6</sup>

## 9. S4A

S4A es una modificación de Scratch que proporciona una programación sencilla de la plataforma abierta de hardware Arduino. Incluye bloques para recibir señales de sensores y controlar actuadores conectados a Arduino. También hay una tabla que informa del estado de los sensores en la pantalla.

Ha sido desarrollada para atraer a la gente al mundo de la programación. Su objetivo es proporcionar una interfaz de nivel alto para programadores de Arduino con funcionalidades como la interacción de varias placas a través de eventos de usuario.

S4A ha sido desarrollado en el Citilab por el equipo de investigación Edutech (Marina Conde, Víctor Casado, Joan Güell, José García y Bernat Romagosa), con la ayuda del grupo de programación smalltalk.cat y de colaboradores como Jorge Gómez, jefe del proyecto de educación en software y hardware de Miscela.

El enlace a la página de S4A es: [http://s4a.cat/index\\_es.html](http://s4a.cat/index_es.html)<sup>7</sup>

En la figura 6 podemos ver la interfaz de S4A:

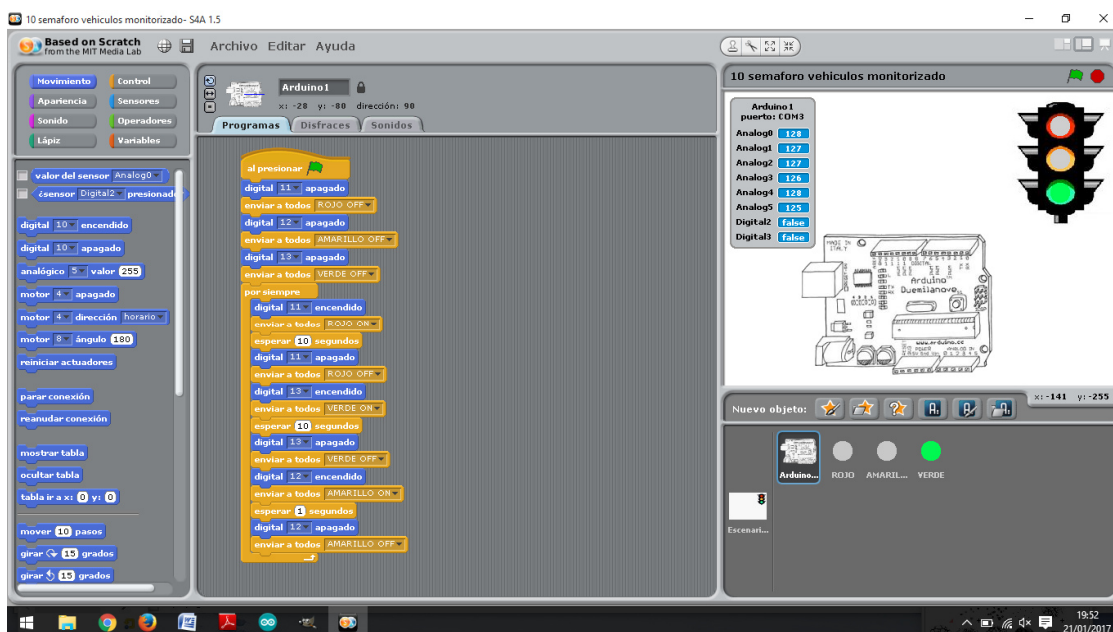


Figura 6: interface gráfica de S4A.

Las especificaciones técnicas de S4A son las siguientes:

- Placas compatibles: Arduino Diecemila, Arduino Duemilanove y Arduino Uno. Por lo tanto es compatible con la Arduino Uno Rev 3 seleccionada.
- Los objetos de la librería Arduino ofrecen bloques de programación para las funcionalidades básicas del microcontrolador, lecturas y escrituras tanto analógicas como digitales, control de servomotores estándar y de rotación continua y otras funcionalidades de más alto nivel. La conectividad de las diferentes entradas y salidas podemos verla en la figura 7:

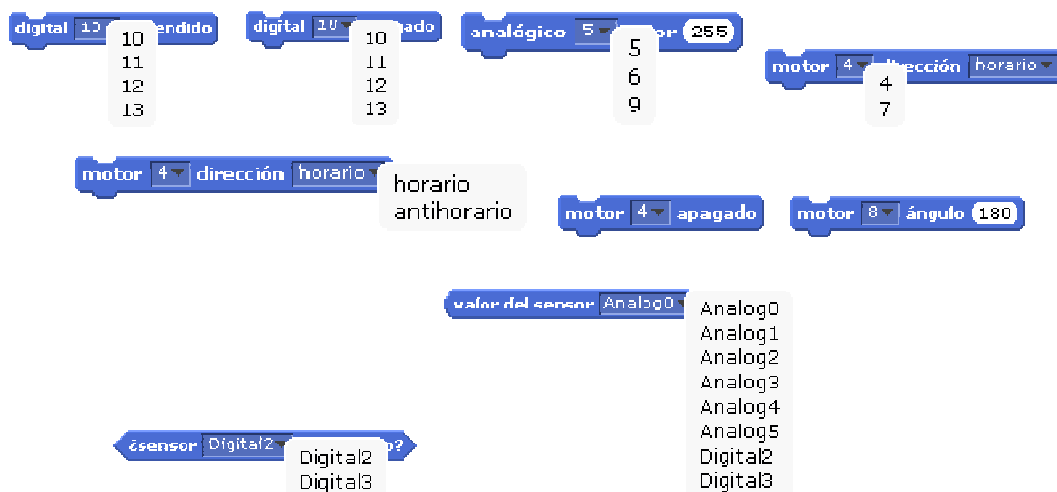


Figura 7: bloques de movimiento de S4A con las diferentes acciones y asignación de entradas y salidas.

- En S4A, una placa Arduino se representa con un tipo especial de objeto. El objeto Arduino 1 encontrará automáticamente el puerto USB en que se haya conectado la placa. Al crear un nuevo objeto Arduino puedes elegir entre crear una nueva conexión o usar una ya existente. Si se crea una nueva conexión podrás trabajar con varias placas conectadas a la vez. Esta característica permite a los objetos virtuales Arduino funcionar de forma colaborativa usando la misma conexión (el objeto físico). El objeto Arduino encontrará automáticamente el puerto USB donde esté conectada cada una de las placas.
- Protocolo de comunicación: S4A interactúa con Arduino enviando el estado de los actuadores y recibiendo el de los sensores cada 75 ms, por lo tanto, el ancho de pulso ha de ser mayor que este período.
- Compatibilidad: S4A es compatible con Scratch, es posible abrir proyectos de Scratch en S4A, pero no a la inversa. Tampoco es posible compartir proyectos de S4A en la web de la comunidad Scratch, ya que esto va en contra de los términos de uso de Scratch.

Instalación y puesta en funcionamiento del conjunto S4A – Arduino Uno Rev3 en un ordenador con Windows:

- Descargar e instalar el software S4A desde la web de S4A en el apartado de descargas: [http://s4a.cat/index\\_es.html](http://s4a.cat/index_es.html)<sup>7</sup>. Elegimos el sistema operativo para el que queremos instalarlo y una vez descargado, lo instalamos.
- Descargar e instalar el entorno de desarrollo integrado Arduino IDE desde el siguiente enlace: <https://www.arduino.cc/en/Main/Software><sup>1</sup>. Elegimos el sistema operativo para el que queremos instalarlo y una vez descargado, lo instalamos. En Windows tenemos la opción de descargar el archivo exe ejecutable para la instalación o el archivo zip comprimido para usarlo como versión portable del IDE.
- Descargar el firmware de S4A desde el siguiente enlace: <http://vps34736.ovh.net/S4A/S4AFirmware15.ino><sup>7</sup>. Este firmware al cargarlo en la placa Arduino facilita la comunicación con S4A y permite el funcionamiento del conjunto.
- Conectar la placa Arduino Uno Rev3 a nuestro ordenador e instalar los drivers si fuera necesario. Los drivers se encuentran dentro de la carpeta Arduino/drivers.
- Abrir el archivo de firmware de arduino (S4AFirmware15.ino) descargado anteriormente, desde el entorno de desarrollo integrado Arduino IDE.
- En el menú herramientas seleccionar el modelo de la placa (Arduino Uno en nuestro caso) y el puerto serie al que está conectada la placa (se conecta por USB pero el sistema operativo crea un puerto serie COM virtual). Ver figura 8.



- Cargar el firmware a la placa Arduino Uno mediante el menú programa/subir o directamente desde el icono con forma de flecha hacia la derecha. Si todo ha funcionado correctamente el Arduino IDE nos indica que se ha subido el programa. Ver figura 8.

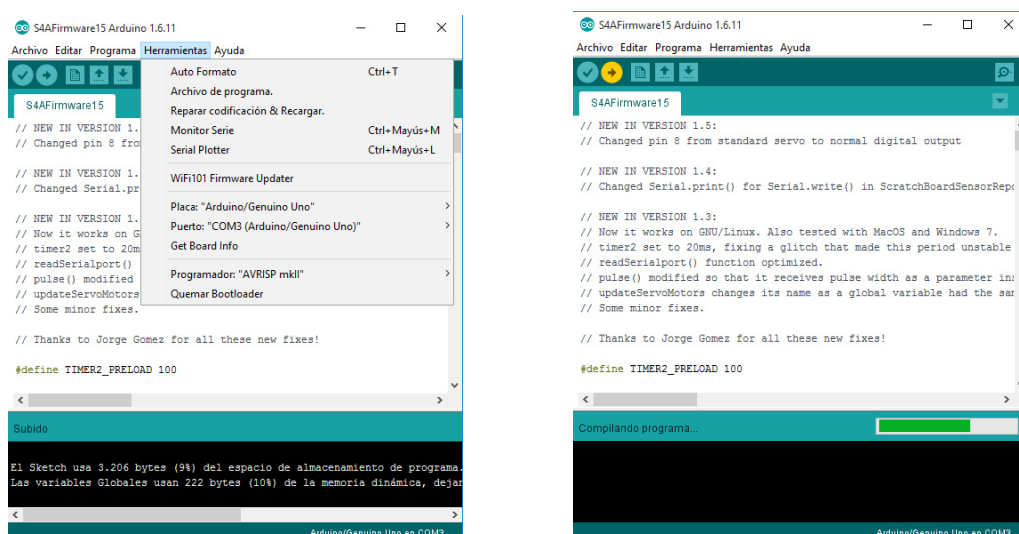


Figura 8: acciones sobre Arduino IDE cargando el firmware de S4A.

Si el sistema operativo usado es Windows XP o inferior, en el siguiente enlace disponemos de un manual de usuario de Arduino, con las instrucciones detalladas para la instalación de los drivers y la carga de un programa desde Arduino IDE: [http://www.uco.es/aulasoftwarelibre/wp-content/uploads/2010/05/Arduino\\_user\\_manual\\_es.pdf](http://www.uco.es/aulasoftwarelibre/wp-content/uploads/2010/05/Arduino_user_manual_es.pdf)<sup>8</sup>

- Ahora sólo quedará comprobar el correcto funcionamiento del conjunto. Para ello, con la placa Arduino Uno conectada a un puerto USB del ordenador, ejecutaremos S4A, de forma que si todo funciona correctamente, desaparecerá del escenario la ventana “Arduino Uno Buscando placa” y veremos que en la placa Arduino se mantiene encendido el led TX y el led RX parpadea continuamente. Ya estamos preparados para montar el sistema a controlar, programarlo y ponerlo en funcionamiento.

## 10.RESULTADOS OBTENIDOS

Una vez citados los conceptos clave del proyecto, seleccionada la placa Arduino que vamos a utilizar (Arduino Uno Rev 3) y el lenguaje de programación (S4A), el siguiente paso y núcleo central de este proyecto es trabajar en el diseño y desarrollo de un cuaderno de prácticas, que permita al profesorado de 4º ESO en la asignatura de Tecnología, trabajar de forma integrada y amena para los alumnos, los conocimientos, las habilidades, destrezas y actitudes relacionados con la electricidad, electrónica, control y robótica.

Antes de empezar a trabajar con el cuaderno de prácticas es necesario tener instalado S4A y el entorno de desarrollo Arduino IDE, cargado en la placa Arduino Uno Rev3 el firmware de S4A y comprobado el correcto funcionamiento del conjunto. Todo este proceso se ha explicado detalladamente en el apartado anterior.

El cuaderno consta de 76 prácticas y se va a estructurar de forma secuencial para que se vaya avanzando práctica a práctica. En algunos momentos, será el profesor el que decida trabajar unas prácticas u otras alternativas, en función de lo que interese y se quiera hacer más hincapié. Un ejemplo de esto es cuando se llega a la práctica 37 y posteriores hasta la 50. En estas prácticas se realiza el control del accionamiento de un relé mediante transistor NPN o PNP y en función de la luminosidad o de la temperatura. De forma que el profesor puede decidir realizar todas las prácticas o seleccionar aquellas que más interés presenten para los alumnos en cada momento. Algo similar ocurre entre las prácticas 54 y 67, en las que se trabaja el control de la intensidad luminosa de una bombilla y la regulación de la velocidad de un motor de corriente continua.

El cuaderno completo aparece en el **Anexo II**. En este apartado de la memoria se va a seleccionar una de las prácticas propuestas en el cuaderno y se va a desarrollar paso a paso, para que se pueda observar la forma de trabajo.

El cuaderno de prácticas del Anexo II, va acompañado por el **Anexo I**, donde aparece un glosario de todos los bloques de S4A que se van a usar en el cuaderno de prácticas, la categoría en la que aparecen y su explicación. En el **Anexo III** aparece una colección de 6 propuestas de proyecto para que los alumnos apliquen todo lo aprendido en el desarrollo completo del correspondiente proyecto a final de curso. En el **Anexo IV** podemos ver una lista con los materiales y componentes necesarios para realizar las prácticas del cuaderno, junto con el presupuesto. Y por último, en el **Anexo V**, podemos ver la práctica completa que se desarrollará el día de la exposición y defensa de este proyecto.

Para el desarrollo del cuaderno se ha utilizado <http://fritzing.org/home/><sup>9</sup>. Fritzing es un software libre que nos permite automatizar el diseño electrónico y que nosotros vamos a utilizar para documentar cada una de las prácticas con los esquemas funcionales y de conexionado de los elementos necesarios en cada una de las prácticas. En los siguientes enlaces encontramos tutoriales para el manejo del programa: <http://fritzing.org/media/uploads/learning/translations/Fritzing-PrimerosPasos.pdf><sup>9</sup>, <http://fritzing.org/media/uploads/learning/translations/Fritzing-ConstruyendoCircuito.pdf><sup>9</sup> y <http://tutorialfritzing.blogspot.com.es/><sup>10</sup>.

Hay que hacer notar que algunos de los operadores utilizados no estaban en la librería del programa (final de carrera, relé, led bicolor, portapilas, ...) y se han tenido que diseñar, bien totalmente, o bien partiendo de algún operador ya diseñado. Para el diseño de estos componentes se ha utilizado el software libre de diseño vectorial <https://inkscape.org/es/><sup>11</sup>, ya que es el formato de salida de este programa (svg) el que utiliza Fritzing para importar los diferentes componentes.

La idea es proporcionar al profesorado los esquemas funcionales y de conexionado, de forma que el profesor, cuando se empiece a trabajar con el cuaderno, pueda mostrar a los alumnos el esquema funcional y a continuación el esquema de

conexión de los diferentes elementos, para que estos puedan entender cómo se realiza la conexión. Una vez entendido, los esquemas de conexión se utilizarán en ocasiones contadas (cuando se trabaje con elementos con los que todavía no se ha trabajado), ya que se facilitará a los alumnos el esquema funcional y serán ellos los que deberán montar las prácticas. Con esto conseguimos que el alumno entienda la función de un esquema eléctrico y sepa usarlo para montar el circuito necesario para la práctica, acercando al alumno a la forma de trabajar en el mundo laboral.

Posteriormente el alumno programará la práctica en S4A, según las instrucciones y secuencia de funcionamiento que se facilitarán en el cuaderno. Estas instrucciones de funcionamiento podrán ser por escrito, o mediante gráficos etapa-transición (GRAF CET). El último paso será comprobar el funcionamiento y si no es correcto, realizar las comprobaciones y modificaciones que sean necesarias, tanto a nivel de montaje eléctrico y electrónico, como a nivel de programación. También se podrá realizar la monitorización de la práctica en la pantalla del ordenador.

A continuación y a modo de ejemplo, podemos ver el desarrollo de la práctica 1 del cuaderno de prácticas, el cual aparece completo en el **Anexo II**:

### Encendido y apagado de un diodo led mediante teclado.

Antes de comenzar la práctica se trabajará con el alumnado las resistencias y el funcionamiento general de un diodo, para acabar concretando en un diodo led. Para ello existen dos opciones, una explicación detallada por parte del profesor o un trabajo de autoaprendizaje del alumno, usando, por ejemplo, el libro multimedia de la página <http://www.tecno12-18.com/><sup>12</sup>, dentro de la unidad de electrónica analógica, apartado de resistencias y apartado del diodo. Se recomienda usar esta segunda opción, de forma que el alumno trabaje con la versión dinámica, vaya aprendiendo poco a poco, haciendo un esquema resumen en el cuaderno y realizando los test de autoevaluación, de forma que es la propia página la que si los test no se realizan correctamente obliga a repasar la parte de teoría relacionada con el test. También se dispone de la versión libre, que puede usar el alumno para repasar, estudiar o concretar y el profesor para explicar los aspectos que quiera recalcar o solucionar las dudas que puedan surgir.

El esquema eléctrico del montaje a realizar para la práctica podemos verlo en la figura 9:

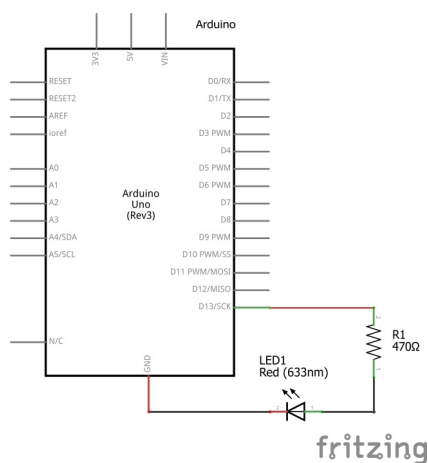


Figura 9: esquema eléctrico funcional de la práctica 1.

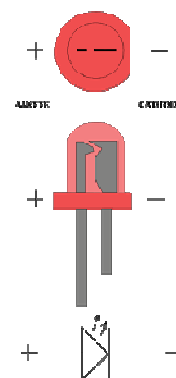


Figura 10: esquema de patillaje de un diodo led.

Hay dos puntos importantes que debemos recalcar al alumnado. El primero es la necesidad de limitar la corriente del diodo led mediante una resistencia (vamos a usar resistencias de 470Ω). El segundo es la polaridad de las patillas del diodo led <sup>18</sup> y que podemos ver en la figura 10.

El montaje se realizará como podemos ver en la figura 11:

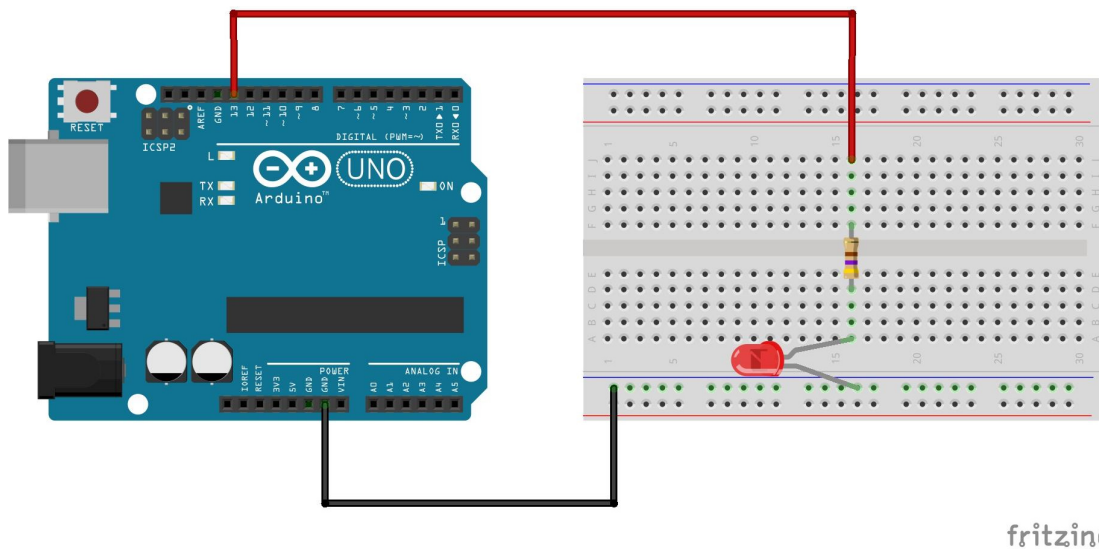


Figura 11: esquema de montaje y conexionado de la práctica 1.

A continuación se realizará la programación en S4A, del funcionamiento de la práctica. Para ello se utilizarán los bloques “al presionar tecla \_” de la librería control asociadas a las letras E (para encender) y A (para apagar) y dos funciones de la librería movimiento “digital \_ encendido” (salen 5V por la correspondiente salida), “digital \_ apagado” (salen 0V por la correspondiente salida) asociadas a la salida digital 13. Estos bloques podemos verlos en la figura 12.

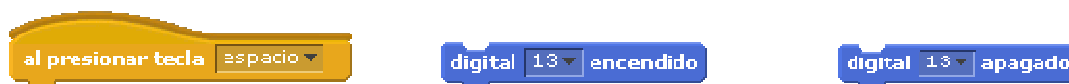


Figura 12: bloques de S4A a utilizar en la práctica 1.

Indicar que presionando sobre la flecha se pueden seleccionar los valores de tecla o salida digital que nos interesen.

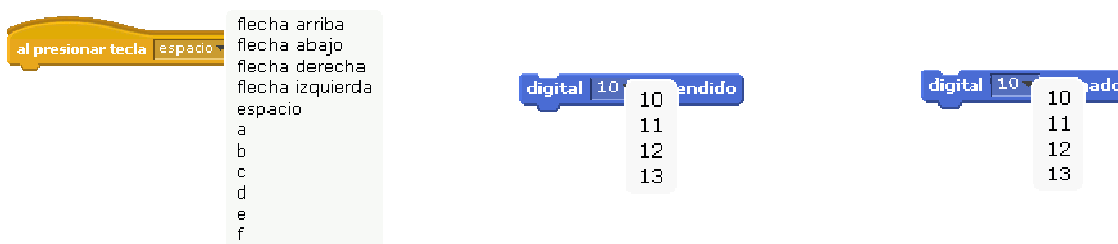


Figura 13: forma de usar los bloques necesarios en la práctica 1.

Una posible solución para nuestra práctica es:



Figura 14: programa en S4A de la práctica 1.

Para que los alumnos se vayan habituando a trabajar con la placa Arduino sería muy interesante hacer la práctica con el resto de salidas digitales (10, 11 y 12) y con diodos de color verde, amarillo, multicolor de cambio lento y multicolor de cambio rápido.

## 11. CONCLUSIONES

El objetivo del proyecto ha sido utilizar una de las placas Arduino, concretamente la Arduino Uno Rev3, como controladora junto con el lenguaje de programación S4A, basado en Scratch, para facilitar al profesorado de la asignatura de tecnología de 4º ESO, herramientas para impartir de forma integrada y amena para los alumnos, los conocimientos, las habilidades, destrezas y actitudes relacionados con la electricidad, electrónica, control y robótica.

Estas herramientas, que han sido el punto central de este proyecto, han consistido en la confección de un cuaderno de prácticas, que podemos ver en el **ANEXO II**, estructurado de forma secuencial para que se vaya avanzando práctica a práctica, asimilando poco a poco los diferentes conocimientos, habilidades, destrezas y actitudes. A este cuaderno de prácticas, también le acompaña un glosario de los bloques de programación a usar en el cuaderno de prácticas en el **ANEXO I**, varias propuestas de proyecto para el trabajo de los alumnos en el **ANEXO III**, una lista de materiales necesarios para llevar a la práctica este proyecto y el presupuesto con el coste que supone en el **ANEXO IV** y una práctica completa que se usará en la exposición de este proyecto en el **ANEXO V**.

En cuanto a otros materiales que se han utilizado y recopilado en este proyecto y que no son de elaboración propia, pero que han sido de mucha utilidad para llevarlo a cabo y que aparecen recopilados en la bibliografía y webgrafía, tenemos un manual de usuario de Arduino, un manual de diferentes entornos de programación gráfica de Arduino y varios manuales y cuadernos de Scratch.

Bajo mi punto de vista y basándome en los resultados que he ido observando al poner en práctica el proyecto en el aula, durante varios cursos, con alumnos de 4º ESO, los resultados han sido muy positivos y conforme me acerco a la implantación total del proyecto, veo que cada vez los resultados son más positivos.

Las posibilidades de aplicación del proyecto en el nivel de 4ºESO son totales e incluso existe la posibilidad de extenderlo a 1º y 2º de bachillerato, bien en las asignaturas de Tecnología Industrial I y II o bien en las asignaturas de TIC I y TIC II, ya que con la última modificación del currículo de las asignaturas de TIC, hay mucha carga lectiva de programación.

Una vez acabado todo el proceso de depósito, presentación y defensa del proyecto, me gustaría pensar la forma de poder hacer llegar este trabajo a los compañeros de tecnología del resto de los centros de Aragón, con la idea de que lo puedan aplicar en sus actividades lectivas.

El cuaderno de prácticas está en constante ampliación con las nuevas prácticas que van surgiendo. Las últimas prácticas del cuaderno han ido surgiendo durante los dos últimos cursos. Y durante los próximos cursos voy a continuar ampliando el proyecto con nuevos sensores (diferentes gases, detección de fugas de agua, humedad, ultrasonidos, ...)

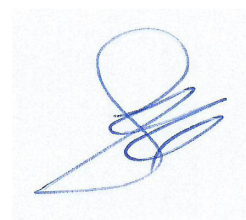
En cuanto a las posibilidades de ampliación y mejora del proyecto, si hay varias ideas que quedan pendientes:

- Aumento del número de propuestas de proyecto para el trabajo final de los alumnos. Se irán aumentando de cara a los próximos cursos.
- Traducción del cuaderno de prácticas para trabajarlo con el entorno de programación gráfica autónomo Ardublock.
- Estudio y prueba de los nuevos lenguajes gráficos de programación que se han creado durante los dos últimos años.

En cuanto a las incidencias que se han presentado en el desarrollo de este proyecto, la principal ha sido la falta de tiempo que he tenido para dedicarle, al estar trabajando. Por este motivo se ha alargado su realización. El pensar las diferentes prácticas era rápido, iban surgiendo conforme estabas trabajando en clase con los alumnos. El problema ha sido organizarlas y fundamentalmente preparar todo el material complementario para la presentación de las prácticas, es decir, esquemas funcionales de conexionado, esquemas de montaje, programación de las prácticas en S4A y captura y edición gráfica de los programas para mostrarlos en el cuaderno de prácticas. Todo este proceso lo he tenido que hacer en los meses de verano de estos últimos años, ya que durante el curso, es bastante difícil sacar el tiempo necesario.

Personalmente me ha gustado mucho este proyecto, ya que yo buscaba algún proyecto que pudiera aplicar a mi actividad profesional y después de darle muchas vueltas, me encontré con Arduino y la verdad es que ha merecido la pena.

Aunque desde el punto de vista personal he sacado una conclusión muy importante que me gustaría que conocieran todos los alumnos universitarios, y es que primero hay que acabar los estudios y después ir al mercado laboral, porque el esfuerzo de retomar algo que ha quedado pendiente en los estudios, es enorme y va aumentando conforme va pasando el tiempo.



Fdo.: Juan Fco Sangüesa Ortiz

## 12. BIBLIOGRAFÍA Y WEBGRAFÍA

1. Página web de Arduino LLC: <https://www.arduino.cc/>
2. Página web de Arduino Srl: <http://www.arduino.org/>
3. José Manuel Ruiz Gutiérrez, Entornos gráficos de programación Arduino, web, <https://www.dropbox.com/s/aywqux80kmiqsn/Programacion%20Grafica%20de%20Arduino.pdf>, Noviembre 2011
4. Blog de José Manuel Ruiz Gutiérrez: <http://josemanuelruizgutierrez.blogspot.com.es/>
5. Página web de Scratch: <https://scratch.mit.edu/>
6. Página web eduteka: <http://eduteka.icesi.edu.co/articulos/ScratchCuadernoTrabajo1>
7. Página web de S4A: [http://s4a.cat/index\\_es.html](http://s4a.cat/index_es.html)
8. Rafael Enriquez Herrador, Guía de usuario de Arduino, web, [http://www.uco.es/aulasoftwarelibre/wp-content/uploads/2010/05/Arduino\\_user\\_manual\\_es.pdf](http://www.uco.es/aulasoftwarelibre/wp-content/uploads/2010/05/Arduino_user_manual_es.pdf), 13 de Noviembre del 2009
9. Página web de fritzing: <http://fritzing.org/home/>
10. Tutorial de fritzing: <http://tutorialfritzing.blogspot.com.es/>
11. Página web de inkscape: <https://inkscape.org/es/>
12. Página web de tecno12-18.com: <http://www.tecno12-18.com/>
13. Página de areatecnología: <http://www.areatecnologia.com/>
14. Video de youtube: <https://www.youtube.com/watch?v=84mxq41zdwE>
15. Control de motores paso a paso: <http://server-die.alc.upv.es/asignaturas/lсед/2002-03/MotoresPasoapaso/Motorespasoapaso.pdf>
16. Motores paso a paso: <https://coscomantauni.files.wordpress.com/2014/01/motor-paso-a-paso.pdf>
17. Página web diymakers: <http://diymakers.es/mover-motores-paso-paso-con-arduino/>
18. Características de componentes electrónicos: <http://www.datasheetcatalog.net/es/>
19. Página web robots Argentina: [http://robots-argentina.com.ar/MotorCC\\_PuenteH.htm](http://robots-argentina.com.ar/MotorCC_PuenteH.htm)
20. Página web micropingüino: <http://micropinguino.blogspot.com.es/2011/04/puente-h-con-el-lm386.html>
21. Aprendiendo Arduino: <https://aprendiendoarduino.wordpress.com/tag/microcontroladores/>

## 13. ÍNDICE DE FIGURAS

*Figura 1: logotipo de Arduino y Genuino.*

*Figura 2: clasificación del conjunto de placas desarrolladas por Arduino.*

*Figura 3: funciones de los diferentes pines y elementos de la placa Arduino Uno Rev3.*

*Figura 4: entorno gráfico del Arduino IDE.*

*Figura 5: interface gráfica de Scratch.*

*Figura 6: interface gráfica de S4A.*

*Figura 7: bloques de movimiento de S4A con las diferentes acciones y asignación de entradas y salidas.*

*Figura 8: acciones sobre Arduino IDE cargando el firmware de S4A.*

*Figura 9: esquema eléctrico funcional de la práctica 1.*

*Figura 10: esquema de patillaje de un diodo led.*

*Figura 11: esquema de montaje y conexionado de la práctica 1.*

*Figura 12: bloques de S4A a utilizar en la práctica 1.*

*Figura 13: forma de usar los bloques necesarios en la práctica 1.*

*Figura 14: programa en S4A de la práctica 1.*









## 14. ÍNDICE DE TABLAS

*Tabla 1: principales características de Arduino Uno Rev3.*





## ANEXO I: GLOSARIO DE BLOQUES DE S4A A USAR EN EL CUADERNO DE PRÁCTICAS



CATEGORÍA: Movimiento

BLOQUE	FUNCIÓN
	Informa del valor del sensor analógico (0 – 1023) seleccionado y conectado a la correspondiente entrada analógica de Arduino (A0, ..., A5).
	Informa del valor del sensor digital (0 = false, 1 = true) seleccionado y conectado a la correspondiente entrada digital (2, 3).
	Enciende o activa (5V) la salida digital seleccionada (10, 11, 12 ó 13).
	Apaga o desactiva (0V) la salida digital seleccionada (10, 11, 12, 13).
	Fija la salida analógica seleccionada (5, 6, 9) al valor elegido con un convertor digital de 256 valores (0-255) correspondiendo 0 a 0V y 255 a 5V.
	Apaga o desactiva la salida PWM seleccionada (4, 7).
	Enciende o activa la salida PWM seleccionada (4, 7) para hacer girar un servomotor de rotación continua en el sentido elegido.
	Activa la salida PWM seleccionada (8) para situar un servomotor angular en el ángulo elegido.









CATEGORÍA: Apariencia

BLOQUE	FUNCIÓN
	Modifica la apariencia del objeto cambiando de disfraz al disfraz elegido.

CATEGORÍA: 

BLOQUE	FUNCIÓN
	Comienza la reproducción del sonido elegido, e inmediatamente pasa al siguiente bloque aunque el sonido no haya acabado de reproducirse.
	Detiene todos los sonidos.

CATEGORÍA: 













BLOQUE	FUNCIÓN
	Ejecuta el programa que tiene debajo al hacer clic en la bandera verde
	Ejecuta el programa que tiene debajo al presionar la tecla seleccionada
	Ejecuta el programa que tiene debajo al hacer clic en el objeto Arduino 1
	Espera el número de segundos seleccionado y luego continúa con el siguiente bloque del programa.
	Ejecuta continuamente los bloques de su interior.
	Ejecuta, el número de veces seleccionadas, los bloques de su interior.
	Envía el mensaje seleccionado a todos los objetos y luego continúa con el bloque siguiente sin esperar a que se realicen las acciones de los objetos activados.
	Envía el mensaje seleccionado a todos los objetos, activándolos para que hagan lo que tengan programado y espera a que todos terminen antes de continuar con el siguiente bloque.

	Ejecuta el programa que tiene debajo cuando recibe el mensaje seleccionado desde el bloque "enviar a todos".
	Comprueba continuamente si una condición es verdadera; cada vez que es verdadera, ejecuta los bloques de su interior.
	Si la condición es verdadera, ejecuta los bloques de su interior.
	Si la condición es verdadera, ejecuta los bloques dentro de la porción sí; si no es verdadera, ejecuta los bloques dentro de la porción si no.
	Espera hasta que la condición sea verdadera para ejecutar los bloques siguientes.
	Comprueba si la condición es falsa; si lo es ejecuta los bloques de su interior y vuelve a comprobar la condición. Si la condición es verdadera, pasa a ejecutar los bloques siguientes.
	Detiene el programa que se está ejecutando dentro de un objeto.
	Detiene todos los programas de todos los objetos.

CATEGORÍA: Sensores















BLOQUE	FUNCIÓN
	Informa verdadero, si la tecla seleccionada está pulsada.
	Pone el cronómetro a 0.
	Usa como dato el tiempo que marca el cronómetro. Si el tic está activado, muestra el valor del tiempo en la pantalla.

CATEGORÍA: **Operadores**

BLOQUE	FUNCIÓN
	Suma dos números constantes o variables.
	Resta el segundo número del primero. Ambos números pueden ser constantes o variables.
	Multiplica dos números constantes o variables.
	Divide el primer número entre el segundo. Ambos números pueden ser constantes o variables.
	Informa verdadero, si el valor del primer número es menor que el del segundo. Ambos números pueden ser constantes o variables.
	Informa verdadero, si los dos valores son iguales. Ambos números pueden ser constantes o variables.
	Informa verdadero, si el valor del primer número es mayor que el del segundo. Ambos números pueden ser constantes o variables.
	Informa verdadero, si ambas condiciones son verdaderas.
	Informa verdadero, si al menos una de las condiciones es verdadera.
	Informa verdadero, si la condición es falsa. E informa falso, si la condición es verdadera.
	Informa del número entero más cercano al número constante o variable.
	Informa del resultado de aplicar la función elegida (raíz cuadrada, funciones trigonométricas, funciones logarítmicas y funciones exponenciales) a un número constante o variable.

CATEGORÍA:

Variables

BLOQUE	FUNCIÓN
	Crea una nueva variable con el nombre que se le ponga. El resto de funciones aparecen una vez creada al menos una variable.
	Borra todos los bloques asociados a una variable, incluida la propia variable.
<input checked="" type="checkbox"/> 	Si el tic está marcado, informa en el escenario del valor de la variable.
  a 	Fija la variable al valor elegido.
  a 	Suma la cantidad seleccionada (puede ser positiva o negativa) a la variable elegida.
 	Muestra el monitor de la variable en el escenario.
 	Esconde el valor de la variable para que no se muestre en el escenario.
	Permite crear y nombrar una nueva lista para clasificar las variables.

## ANEXO II: CUADERNO CON COLECCIÓN DE PRÁCTICAS

### ÍNDICE

1. Encendido y apagado de un diodo led mediante teclado.
2. Encendido y apagado de un diodo led mediante teclado y monitorización de su estado en pantalla.
3. Encendido y apagado de un diodo led mediante teclado, monitorización de su estado en pantalla y control de un sonido.
4. Encendido y apagado de un diodo led mediante teclado y control de su intensidad luminosa (salida analógica 0-255, 0V-5V).
5. Parpadeo de un diodo led con diferente tiempo de encendido y apagado.
6. Parpadeo de dos diodos led con diferentes tiempos de encendido y apagado cada uno.
7. Control del encendido y apagado de un diodo bicolor-tricolor.
8. Control de un diodo RGB mediante salidas digitales.
9. Control de un diodo RGB mediante salidas analógicas.
10. Control de un semáforo de vehículos usando salidas digitales.
11. Control de un semáforo de vehículos usando salidas analógicas.
12. Control de un semáforo de peatones.
13. Control conjunto de un semáforo de vehículos y de un semáforo de peatones.
14. Control del encendido y apagado de un diodo led mediante un cronómetro.
15. Control del encendido y apagado de un diodo led mediante un pulsador usando una entrada digital.
16. Control del encendido y apagado de un diodo led mediante un pulsador con retardo a la desconexión.
17. Control del encendido y apagado de un diodo led mediante un pulsador con retardo a la conexión.
18. Control del encendido y apagado de un diodo led mediante un pulsador con retardo a la conexión y a la desconexión.
19. Control de un semáforo de peatones con pulsador.
20. Control del encendido y apagado de un diodo led mediante un pulsador usando una entrada analógica.
21. Control del encendido y apagado de un diodo led mediante un pulsador usando una entrada analógica asignada a una variable.
22. Control del encendido y apagado de un diodo led mediante un cronómetro o el accionamiento de un pulsador.
23. Control del encendido y apagado de un diodo led mediante un cronómetro y el accionamiento de un pulsador.

24. Control de un semáforo de vehículos y de un semáforo de peatones con pulsador.
25. Control del parpadeo de un diodo led con tiempo de encendido fijo y tiempo de apagado seleccionado mediante una resistencia variable.
26. Control del parpadeo de un diodo led con tiempo de encendido fijo y tiempo de apagado seleccionado mediante una resistencia variable (entrada analógica asignada a variable).
27. Control del parpadeo de un diodo led con tiempo de encendido fijo y tiempo de apagado seleccionado mediante una resistencia variable (entrada analógica asignada a una variable y otra variable para el cálculo).
28. Control del parpadeo de un diodo led con regulación independiente del tiempo de encendido y de apagado seleccionado mediante una resistencia variable asignada a una variable para el cálculo.
29. Regulación de la intensidad luminosa de un diodo led mediante una resistencia variable.
30. Regulación de la intensidad luminosa de un diodo led mediante un deslizador en pantalla.
31. Regulación de la intensidad luminosa de un diodo led mediante una resistencia variable, mostrando el valor analógico de salida y el valor analógico de salida en voltios.
32. Control del encendido y apagado de un diodo led en función de la luminosidad ambiental (LDR).
33. Control del encendido con retardo y apagado de un diodo led en función de la luminosidad ambiental.
34. Control del encendido y apagado con retardo a la conexión y a la desconexión de un diodo led en función de la luminosidad ambiental.
35. Regulación y control de la intensidad luminosa de un diodo led y de su encendido y apagado en función de la luminosidad ambiental.
36. Detector de vibraciones.
37. Control del accionamiento mediante pulsador de un relé, de forma indirecta a través de un transistor NPN BD135 ( $I_c \text{ max} = 1,5 \text{ A}$ ).
38. Control del accionamiento mediante pulsador de un relé, de forma indirecta a través de un transistor PNP BD138 ( $I_c \text{ max} = -1,5 \text{ A}$ ).
39. Control del accionamiento, en función de la luminosidad ambiental, de un relé, de forma indirecta a través de un transistor NPN BD135.
40. Control del accionamiento, en función de la luminosidad ambiental, de un relé, de forma indirecta a través de un transistor PNP BD138.
41. Control del accionamiento, en función de la luminosidad ambiental, de un relé, de forma indirecta a través de un transistor NPN BD135, con posibilidad de elegir y modificar el nivel de referencia de luminosidad.

42. Control del accionamiento, en función de la luminosidad ambiental, de un relé, de forma indirecta a través de un transistor PNP BD138, con posibilidad de elegir y modificar el nivel de referencia de luminosidad.
43. Control del encendido y apagado de un diodo led y de un zumbador en función de una temperatura máxima.
44. Control del encendido y apagado de un diodo led y de un zumbador en función de una temperatura mínima.
45. Termómetro digital con LM35 con indicación de la temperatura en pantalla.
46. Termostato básico con LM35.
47. Control del accionamiento, en función de la temperatura ambiental, de un relé, de forma indirecta a través de un transistor NPN BD135 (termostato avanzado).
48. Control del accionamiento, en función de la temperatura ambiental, de un relé, de forma indirecta a través de un transistor PNP BD138 (termostato avanzado).
49. Control del accionamiento, en función de la temperatura ambiental, de un relé, de forma indirecta a través de un transistor NPN BD135, con posibilidad de elegir y modificar el nivel de referencia de temperatura (termostato completo).
50. Control del accionamiento, en función de la temperatura ambiental, de un relé, de forma indirecta a través de un transistor PNP BD138, con posibilidad de elegir y modificar el nivel de referencia de temperatura (termostato completo).
51. Contador +1.
52. Contador +1 -1.
53. Contador +1 -1 reset.
54. Control del encendido y apagado de una lámpara incandescente mediante un transistor NPN BC547 ( $I_c \text{ max} = 100 \text{ mA}$ ).
55. Regulación y control de la intensidad luminosa de una lámpara incandescente y de su encendido y apagado, de forma indirecta, a través de un transistor NPN BC547.
56. Control del encendido y apagado de una lámpara incandescente mediante un transistor PNP BC557.
57. Regulación y control de la intensidad luminosa de una lámpara incandescente y de su encendido y apagado, de forma indirecta, a través de un transistor PNP BC557.
58. Control del encendido y apagado de un motor, de forma indirecta, a través de un transistor NPN BD135.
59. Regulación y control de la velocidad y del encendido y apagado de un motor, de forma indirecta, a través de un transistor NPN BD135.
60. Control del encendido y apagado de un motor, de forma indirecta, a través de un transistor PNP BD138.
61. Regulación y control de la velocidad y del encendido y apagado de un motor, de forma indirecta, a través de un transistor PNP BD138.
62. Marcha-paro y control de la velocidad de un motor, de forma indirecta, a través de un transistor NPN BD135.

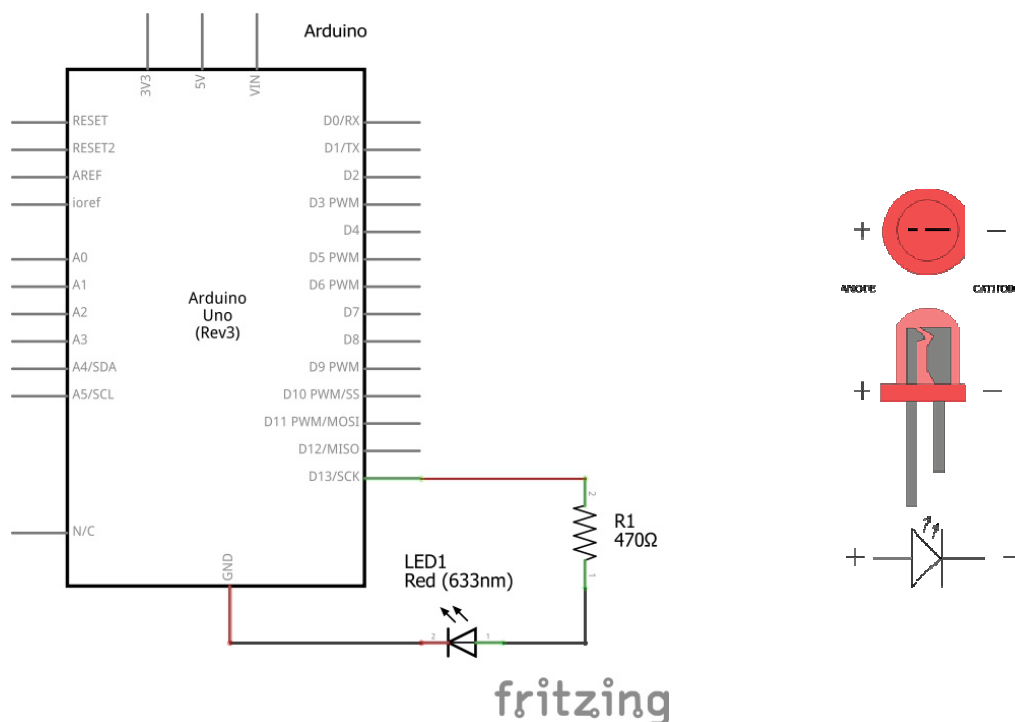


- 63.** Marcha-paro con varios pulsadores y función OR, control de la velocidad de un motor, de forma indirecta, a través de un transistor NPN BD135.
- 64.** Marcha-paro con varios pulsadores y función AND, control de la velocidad de un motor, de forma indirecta, a través de un transistor NPN BD135.
- 65.** Marcha-paro con prioridad al paro y control de la velocidad de un motor, de forma indirecta, a través de un transistor NPN BD135.
- 66.** Marcha-paro con prioridad a la marcha y control de la velocidad de un motor, de forma indirecta, a través de un transistor NPN BD135.
- 67.** Marcha paro con prioridad al paro de un diodo led, desde pulsadores y pulsadores virtuales en pantalla, con monitorización del estado del diodo led.
- 68.** Control total (encendido, apagado, sentido de giro y velocidad) de un motor mediante un puente H con transistores PNP BC138 y NPN BC135, controlados por otros dos transistores NPN BC547.
- 69.** Control total (encendido, apagado, sentido de giro y velocidad) de un motor mediante un puente H confeccionado con dos amplificadores operacionales LM386.
- 70.** Control total (encendido, apagado, sentido de giro y velocidad) de un motor mediante el driver L298N.
- 71.** Control de un motor miniservo ( $0^{\circ}$ - $180^{\circ}$ ) mediante una resistencia variable.
- 72.** Control de un servo de rotación continua.
- 73.** Cuenta atrás en un display numérico de 7 segmentos.
- 74.** Cuenta adelante programada en BCD, decodificada y mostrada en un display de 7 segmentos.
- 75.** Control de un motor paso a paso bipolar mediante el driver L298N.
- 76.** Control de un diodo led, en función del estado de una barrera óptica formada por un diodo de infrarrojos TSUS 5400 y un fototransistor BPW96.

## 1. Encendido y apagado de un diodo led mediante teclado.

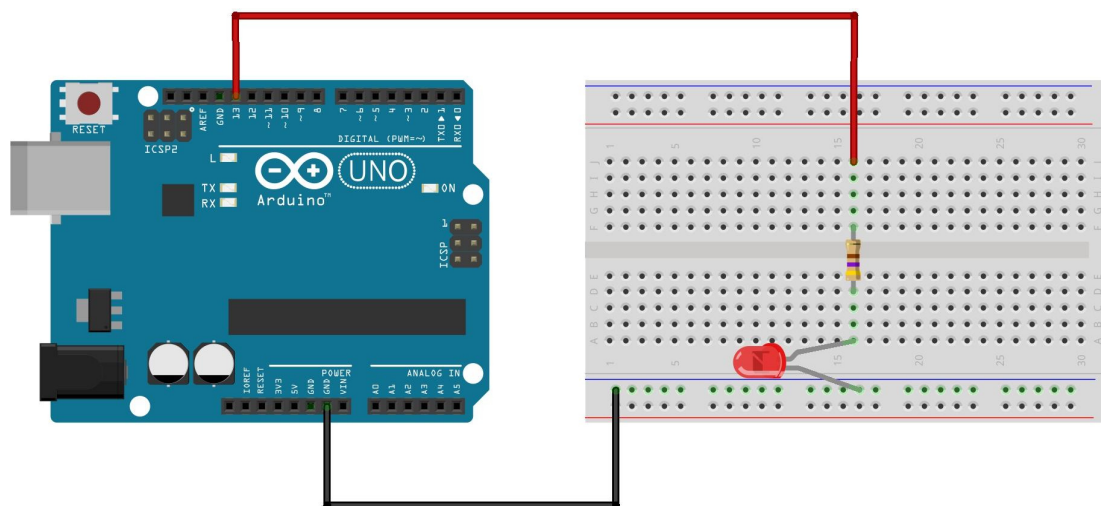
Antes de comenzar la práctica se trabajará con el alumnado las resistencias y el funcionamiento general de un diodo, para acabar concretando en un diodo led. Para ello existen dos opciones, una explicación detallada por parte del profesor o un trabajo de autoaprendizaje del alumno, usando, por ejemplo, el libro multimedia de la página <http://www.tecno12-18.com/><sup>12</sup>, dentro de la unidad de electrónica analógica, apartado de resistencias y apartado del diodo. Se recomienda usar esta segunda opción, de forma que el alumno trabaje con la versión dinámica, vaya aprendiendo poco a poco, haciendo un esquema resumen en el cuaderno y realizando los test de autoevaluación, de forma que es la propia página la que si los test no se realizan correctamente obliga a repasar la parte de teoría relacionada con el test. También se dispone de la versión libre, que puede usar el alumno para repasar, estudiar o concretar y el profesor para explicar los aspectos que quiera recalcar o solucionar las dudas que puedan surgir.

El esquema eléctrico del montaje a realizar para la práctica es el siguiente:



Hay dos puntos importantes que debemos recalcar al alumnado. El primero es la necesidad de limitar la corriente del diodo led mediante una resistencia (vamos a usar resistencias de 470Ω). El segundo es la polaridad de las patillas del diodo led<sup>(18)</sup> y que podemos ver en la figura anterior.

El montaje se realizará como podemos ver en la siguiente figura:

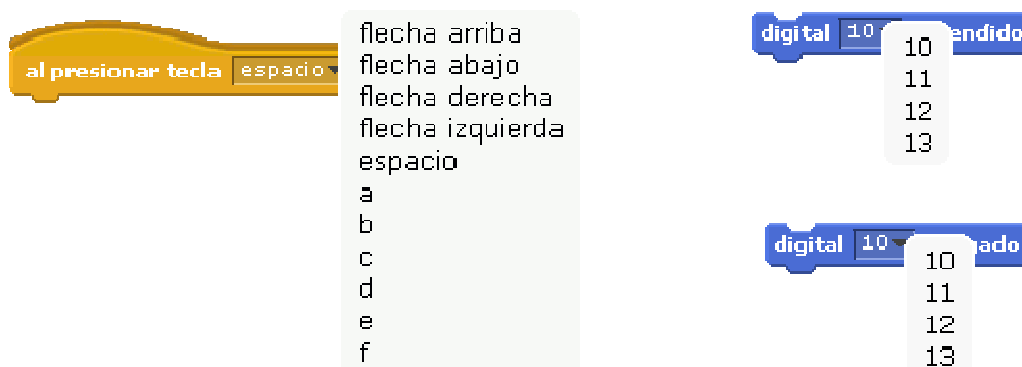


fritzing

A continuación se realizará la programación en S4A, del funcionamiento de la práctica. Para ello se utilizarán los bloques “al presionar tecla \_” de la librería control asociadas a las letras E (para encender) y A (para apagar) y dos funciones de la librería movimiento “digital \_ encendido” (salen 5V por la correspondiente salida), “digital \_ apagado” (salen 0V por la correspondiente salida) asociadas a la salida digital 13.



Recordar que presionando sobre la flecha se pueden seleccionar los valores de tecla o salida digital que nos interesen.



Una posible solución para nuestra práctica es:

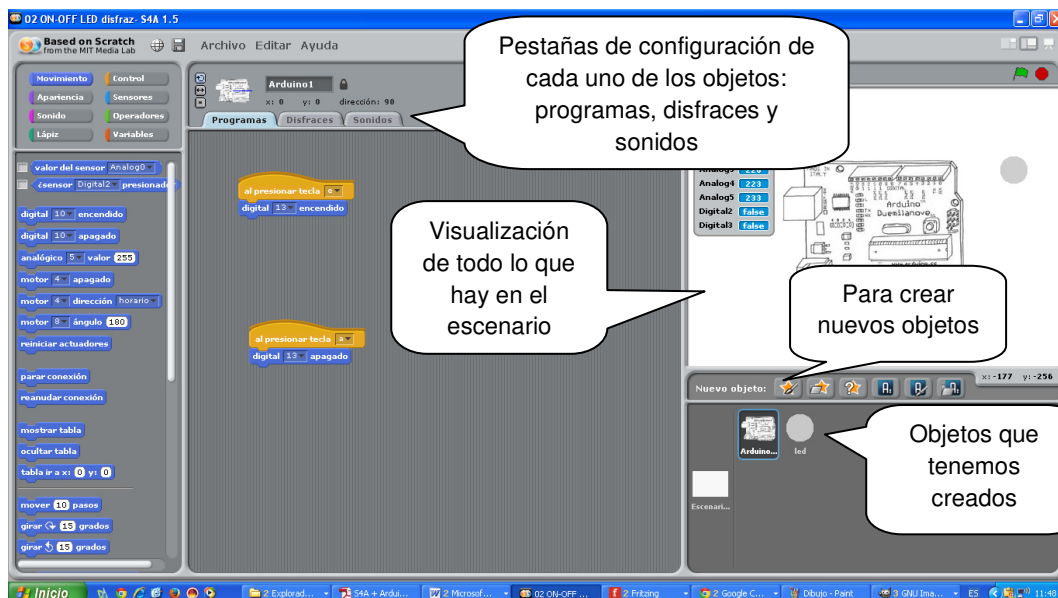


Para que los alumnos se vayan habituando a trabajar con la placa Arduino sería muy interesante hacer la práctica con el resto de salidas digitales (10, 11 y 12) y con diodos de color verde, amarillo, multicolor de cambio lento y multicolor de cambio rápido.

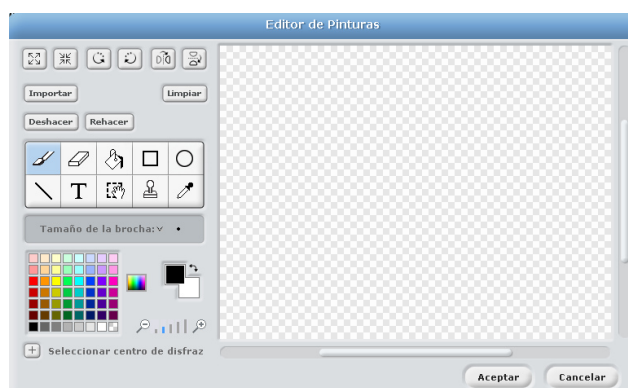
## 2. Encendido y apagado de un diodo led mediante teclado y monitorización de su estado en pantalla.

Los esquemas eléctricos de montaje son los mismos que los de la práctica 1. La única diferencia es que vamos a ampliar la práctica añadiendo la monitorización del estado del diodo led en la pantalla del ordenador.

Para ello hay que tener claro que S4A trabaja con un escenario. Que nosotros podemos crear o importar todos los objetos que nos sean necesarios y que los colocaremos sobre el escenario. Que cada objeto puede tener diferentes disfraces, es decir, diferentes formas de visualizarse según circunstancias. Y que cada objeto se puede programar de forma individual.



La creación de objetos es muy sencilla. Se basa en un editor de dibujo muy básico y con herramientas muy sencillas. Los alumnos no tendrán ningún problema para manejarlo con soltura.



Dibujaremos el diodo led de forma muy sencilla, círculo relleno de gris para representar el estado de diodo apagado. Al aceptar veremos que se ha creado un nuevo objeto al que le pondremos nombre (led). De este objeto tendremos, como de cualquier objeto, las pestañas programas, disfraces y sonidos.



En la pestaña disfraces, copiaremos el disfraz original para duplicarlo. Editaremos el disfraz duplicado, para que su apariencia sea la del diodo led encendido, rellenándolo de color rojo y aceptaremos para guardar los cambios.

A continuación pondremos nombre a los disfraces. Led on para el disfraz que usaremos cuando el led esté encendido y led off para el disfraz que usaremos cuando el led esté apagado.

Ahora solo nos queda programar el funcionamiento. La programación del objeto Arduino 1 es la misma que en la práctica anterior, ya que el funcionamiento es el mismo.

Para la programación del objeto led utilizaremos el bloque “al presionar tecla \_” de la librería control asociadas a las letras E (para encender) y A (para apagar) y el bloque “cambiar el disfraz a \_\_” de la librería apariencia (recordar que al pinchar sobre la flecha nos dejará elegir el disfraz con el que queremos trabajar).



Una posible solución a nuestra práctica es:



### 3. Encendido y apagado de un diodo led mediante teclado, monitorización de su estado en pantalla y control de un sonido.

Los esquemas eléctricos de montaje son los mismos que los de la práctica 1 y 2. La única diferencia es que vamos a ampliar la práctica añadiendo la reproducción de un sonido para completar la monitorización realizada en la práctica 2.

Lo primero que hay que hacer es grabar o importar el sonido deseado (formatos wav o mp3) dentro del objeto en el que lo queremos reproducir. En nuestro caso dentro del objeto led. Para grabarlo nos aparece un grabador de sonidos muy básico, con el que ningún alumno tendrá problemas para trabajar. Si lo queremos importar lo podemos hacer desde la colección que lleva el programa o desde alguna carpeta donde lo tengamos guardado.



Una vez tenemos el sonido, el siguiente paso es realizar el programa. Nos centraremos en la programación del objeto led, ya que la programación del objeto Arduino 1 es la misma que en la práctica 1, ya que su funcionamiento es el mismo.

Para la programación del objeto led, aparte de los bloques ya utilizados en las anteriores prácticas, también utilizaremos los bloques “tocar sonido\_\_” y “detener todos los sonidos” de la librería sonido.



Recordar que si trabajamos con varios sonidos, al pinchar sobre la flecha podremos elegir aquel sonido con el que queremos trabajar.

Una posible solución a nuestra práctica es:



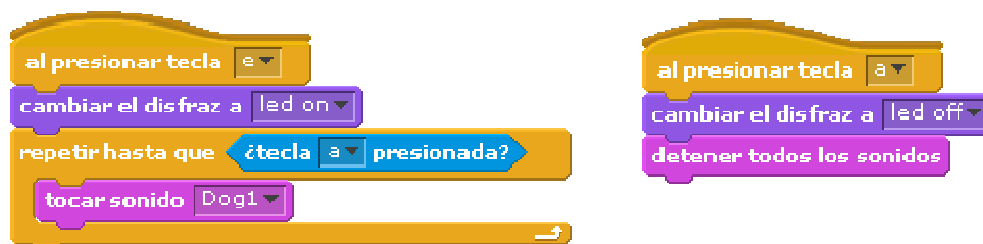
La práctica la podemos complicar un poco más haciendo que el sonido se repita mientras el diodo led esté encendido.

Aparte de los bloques ya utilizados anteriormente, también utilizaremos los bloques “repetir hasta que \_\_” de la librería control y “¿Tecla \_\_ presionada?” (al presionar sobre la flecha podemos elegir la tecla deseada) de la librería sensores. Para

colocar el bloque “¿Tecla \_\_ presionada?” tenemos que cogerlo con el ratón y desplazarlo encima del hexágono irregular que hay en el bloque “repetir hasta que \_\_”. Cuando nos acercamos a la zona del hexágono vemos que se activa y se resalta en color blanco, momento en el que podemos soltar el bloque.



El posible programa podría ser:

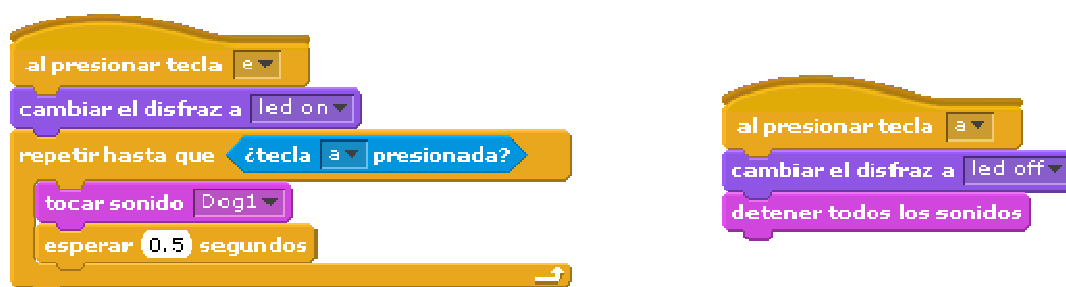


Escucharemos que el sonido se repite con demasiada rapidez. Para corregirlo es interesante dejar un tiempo entre repetición y repetición.

Necesitaremos utilizar el bloque “esperar \_\_ segundos” (al pinchar sobre el número podemos modificarlo según nuestras necesidades) de la librería control.



La solución podría ser:

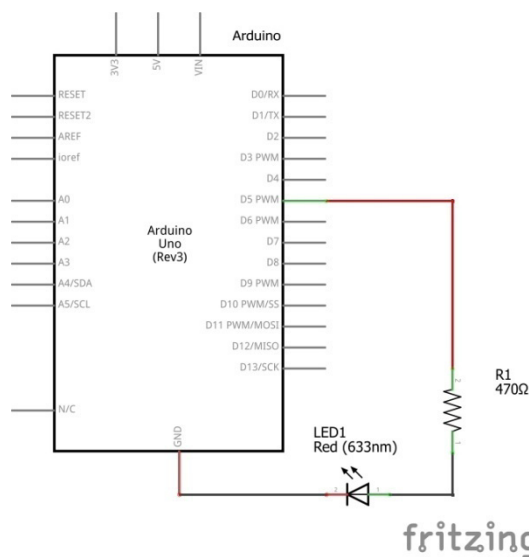


La librería sonido da mucho juego y se puede dejar a los alumnos durante un tiempo que practiquen con diferentes notas, tambores, instrumentos, volumen y tempo.

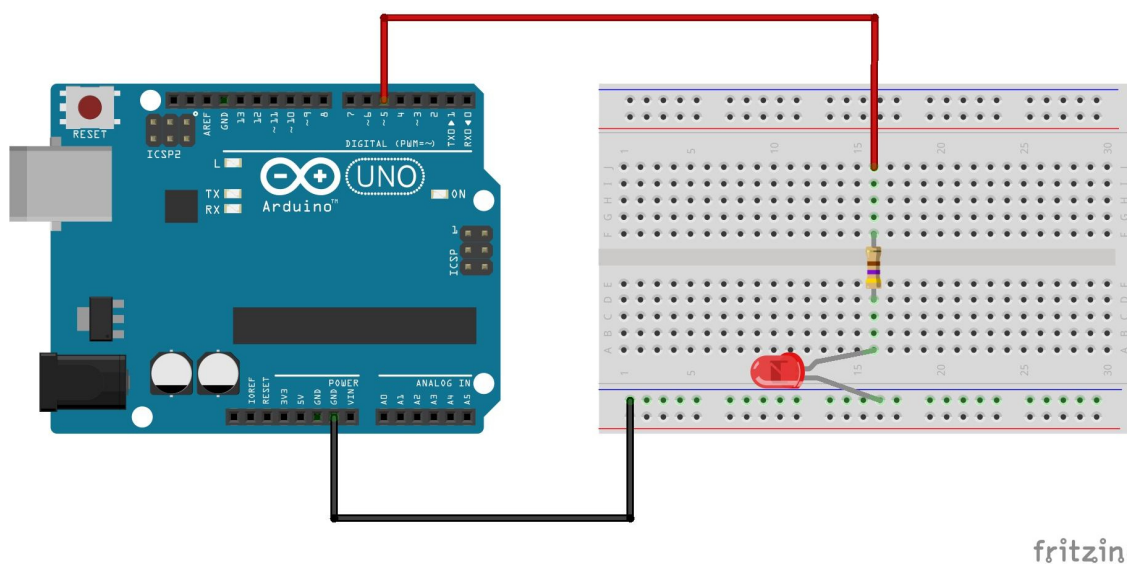
#### 4. Encendido y apagado de un diodo led mediante teclado y control de su intensidad luminosa (salida analógica 0-255, 0V-5V).

Esta práctica es la misma que la práctica 1 con la diferencia de que vamos a utilizar una salida analógica. Las salidas analógicas de la placa Arduino Uno trabajan con 8 bits, oscilando sus valores entre 0 y 255 ( $2^8 = 256$  valores diferentes), correspondiéndose la salida 0 con 0V y la salida 255 con 5V.

El esquema eléctrico del montaje a realizar para la práctica es el siguiente:



El montaje se realizará como podemos ver en la siguiente figura:

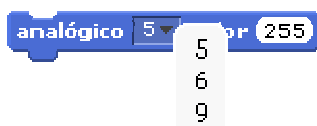


A continuación se realizará la programación en S4A, del funcionamiento de la práctica. Para ello se utilizarán los bloques “al presionar tecla \_\_\_” de la librería control asociadas a las letras E (para encender) y A (para apagar) y la función de la librería movimiento “analógico \_\_\_ valor\_\_\_” asociada a la salida 5 y con un valor comprendido entre 0 y 255 en función de la luminosidad con la que queremos que luzca el diodo led.





Recordar que presionando sobre la flecha se pueden seleccionar los valores de tecla o salida analógica que nos interesen y que al pinchar sobre el número del valor podemos modificarlo según nuestras necesidades, siempre que sea un número entero positivo comprendido entre 0 y 255.



Una posible solución para nuestra práctica es:



De esta forma, si nos damos cuenta, utilizamos las salidas analógicas como si fueran salidas digitales, por lo que en caso de necesitar más de 4 salidas digitales, podemos usar las analógicas para este fin.

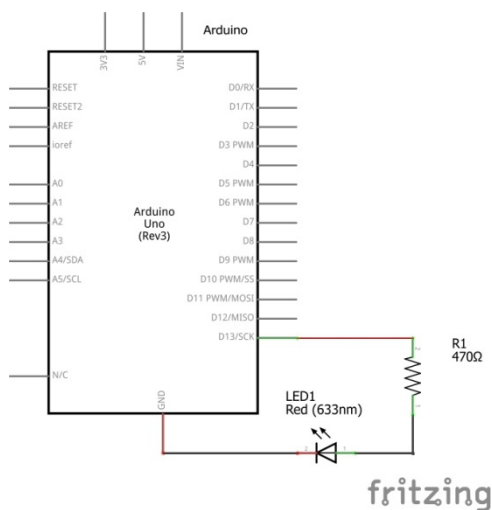
Para que los alumnos se vayan habituando a trabajar con la placa Arduino sería muy interesante hacer la práctica con el resto de salidas analógicas (6 y 9) y que modifiquen el valor de la salida analógica correspondiente, para poder observar cómo se modifica la intensidad luminosa del diodo led.



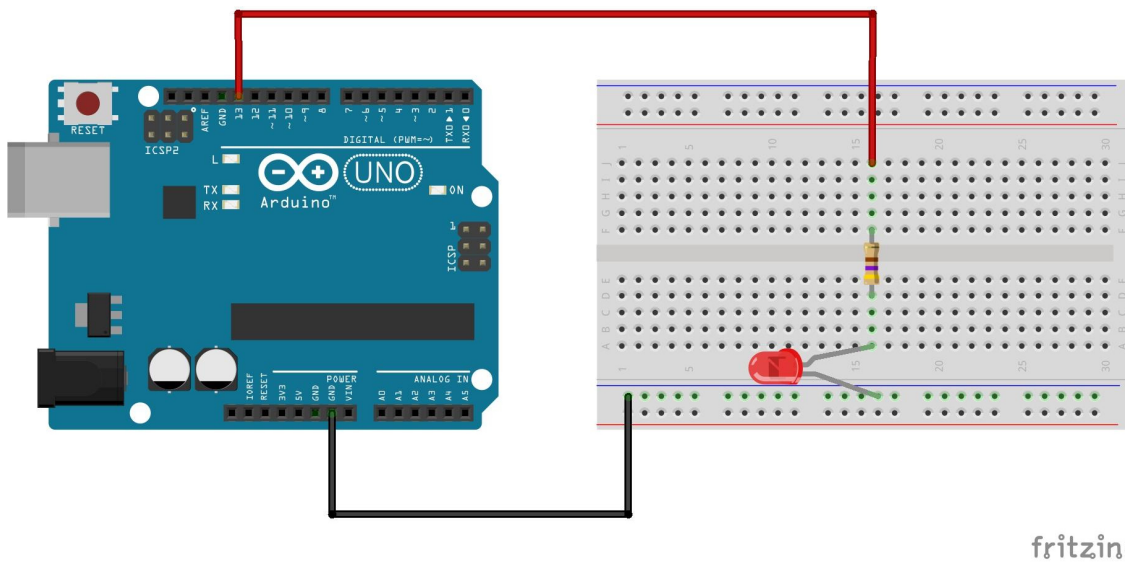
Más adelante, en otras prácticas, veremos cómo podemos regular la intensidad luminosa actuando externamente, es decir, sin tener que modificarla desde el ordenador a través de S4A.

### 5. Parpadeo de un diodo led con diferente tiempo de encendido y apagado.

El esquema eléctrico del montaje a realizar para la práctica es el siguiente:



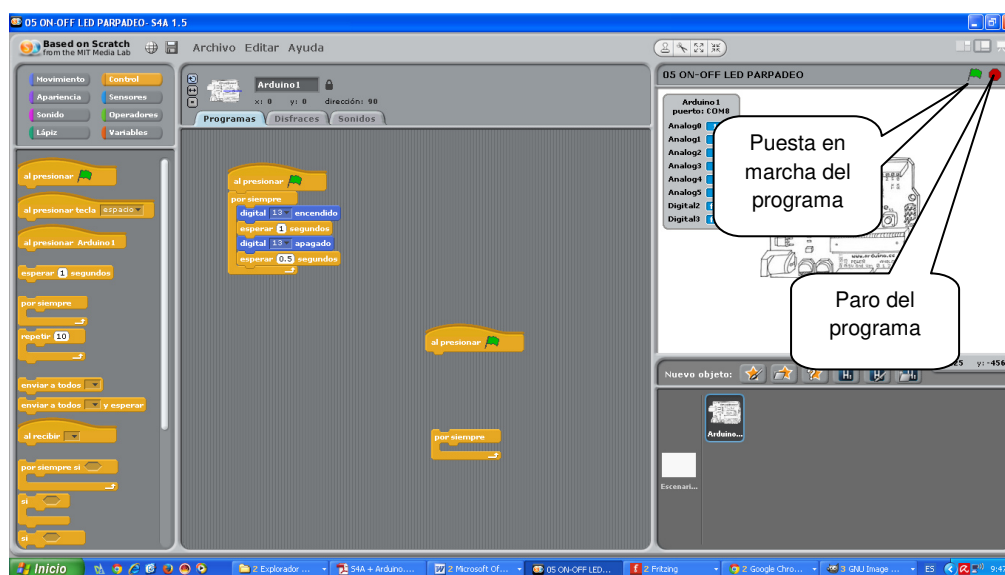
El montaje se realizará como podemos ver en la siguiente figura:



A continuación se realizará la programación en S4A, del funcionamiento de la práctica. Para ello, como bloques no vistos en las prácticas anteriores, se utilizarán los bloques “al presionar bandera verde” para poner en marcha el programa y “por siempre” para que el programa se repita hasta que lo paremos. Ambos bloques son de la librería control.



Inicialmente les indicaremos a los alumnos que lo programen para que el diodo led esté encendido durante 1 segundo y apagado durante 0,5 segundos.



Una posible solución para nuestra práctica es:



Hay que hacer notar, que una vez puesto en marcha el programa y antes del bloque “por siempre” se recomienda fijar los valores iniciales de las salidas con las que vayamos a trabajar.

Se pueden ir modificando los tiempos de encendido y apagado para ver como se modifica el funcionamiento del diodo.

Recordar que también es muy interesante, para que los alumnos se vayan habituando a trabajar con la placa Arduino, hacer la práctica con el resto de salidas digitales (10, 11 y 12) y con diodos de color verde y amarillo.

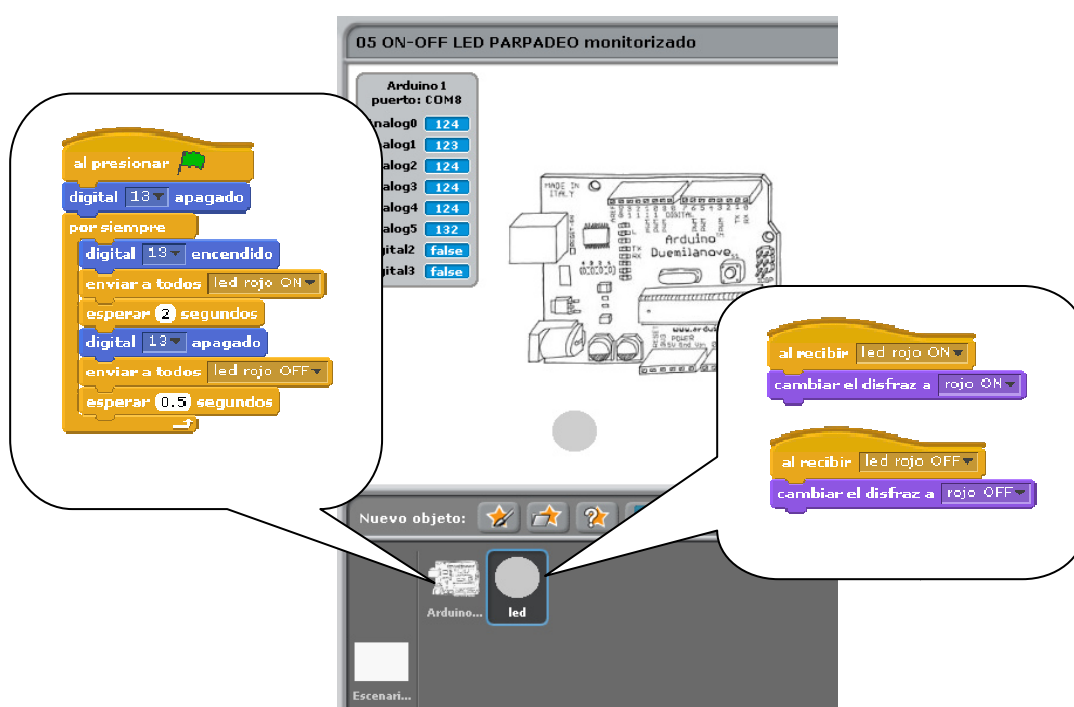
Para seguir avanzando y a la vez repasar los contenidos de las prácticas anteriores, se puede monitorizar esta práctica. Se puede mandar esta tarea para que los alumnos la trabajen individualmente y en casa.

Al día siguiente, es muy interesante comentarles el uso de los bloques “enviar a todos \_\_\_” y “al recibir” de la librería de control. Estos bloques nos simplifican la programación de los objetos, ya que el programa principal lo hacemos en el objeto Arduino 1 y desde este enviamos mensajes para modificar el disfraz de otros objetos cuando reciban el correspondiente mensaje. Ambos bloques nos permiten elegir mensajes ya escritos o escribir mensajes nuevos.



En el menú que nos aparece al hacer clic sobre la flecha, y seleccionando nuevo, podemos crear los mensajes que sean necesarios.

La programación en S4A:



El bloque “enviar a todos \_\_” envía el mensaje seleccionado y continúa ejecutando el siguiente bloque del programa, sin esperar a que se realicen las acciones en los objetos que han recibido el mensaje. Si necesitáramos detener la ejecución del programa, hasta que las acciones en los objetos que han recibido el mensaje hayan terminado de realizarse, podríamos utilizar el bloque “enviar a todos \_\_ y esperar”.



Otra forma de poner en funcionamiento los programas es utilizando el bloque de la librería de control “al presionar Arduino 1”, de forma que cuando se haga clic sobre el objeto Arduino, el programa se pondrá en funcionamiento.



Para detener los programas podríamos utilizar los bloques de la librería de control “detener programa” para detener el programa del objeto en el que esté colocado el bloque y “detener todo” para detener todos los programas de todos los objetos. Es bueno tener un bloque que funcione en paralelo con el resto de programas para poder parar todo.

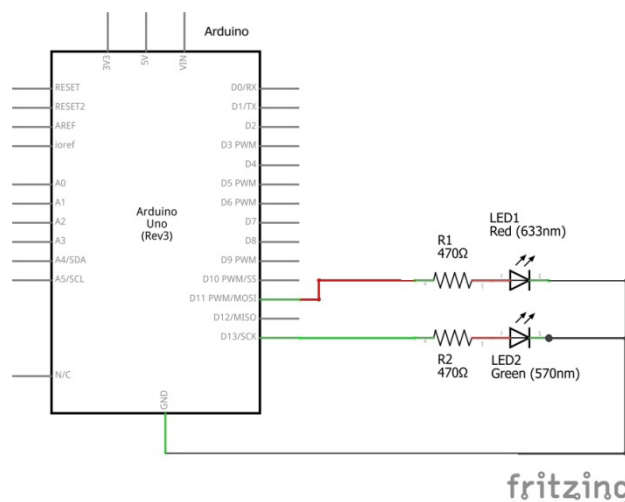


El programa en S4A y que nos serviría para todas las prácticas podría ser el siguiente:

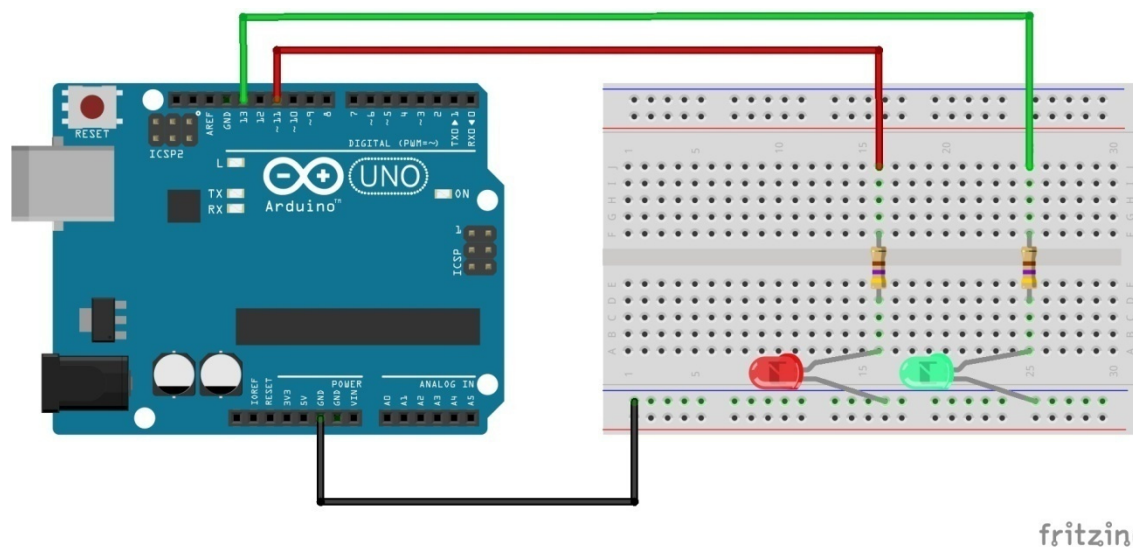


**6. Parpadeo de dos diodos led con diferentes tiempos de encendido y apagado cada uno.**

El esquema eléctrico del montaje a realizar para la práctica es el siguiente:



El montaje se realizará como podemos ver en la siguiente figura:



Todos los bloques necesarios para la programación en S4A se han visto en las prácticas anteriores. Es importante hacer ver a los alumnos la importancia de la programación en bucles separados, para poder controlar de forma totalmente independiente ambos diodos led. El programa es el siguiente:



Y la programación en S4A con la monitorización para los tres objetos:

Objeto Arduino:



Objeto led rojo:

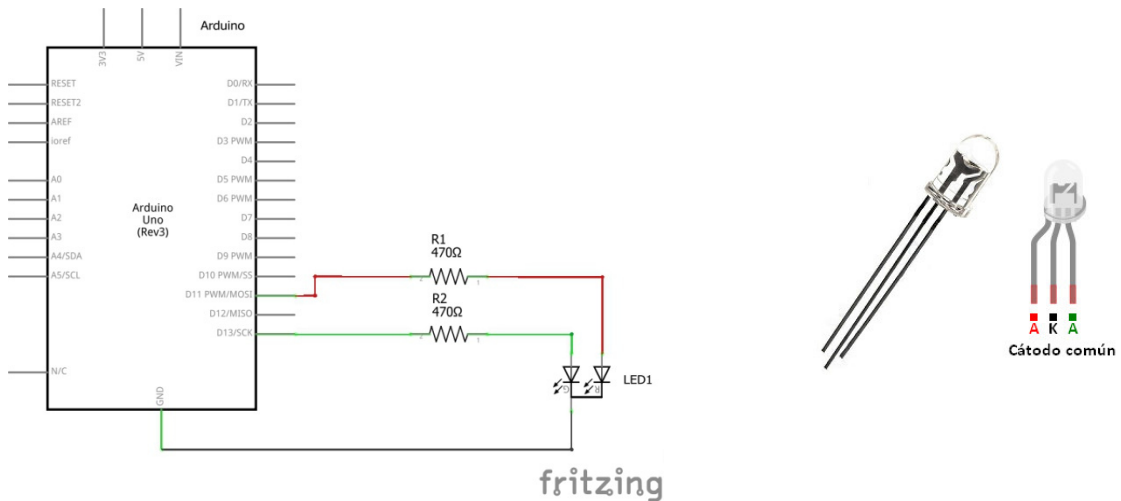


Objeto led verde:

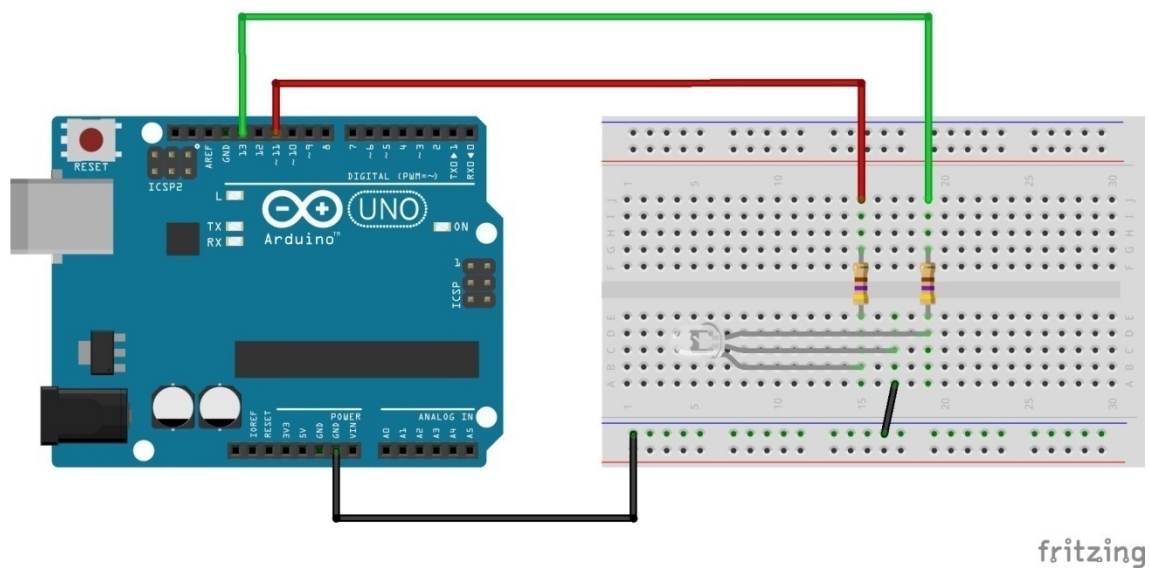


### 7. Control del encendido y apagado de un diodo bicolor-tricolor.

El esquema eléctrico del montaje a realizar para la práctica junto con el esquema de las patillas del diodo bicolor <sup>18</sup> son los siguientes:



El montaje se realizará como podemos ver en la siguiente figura:



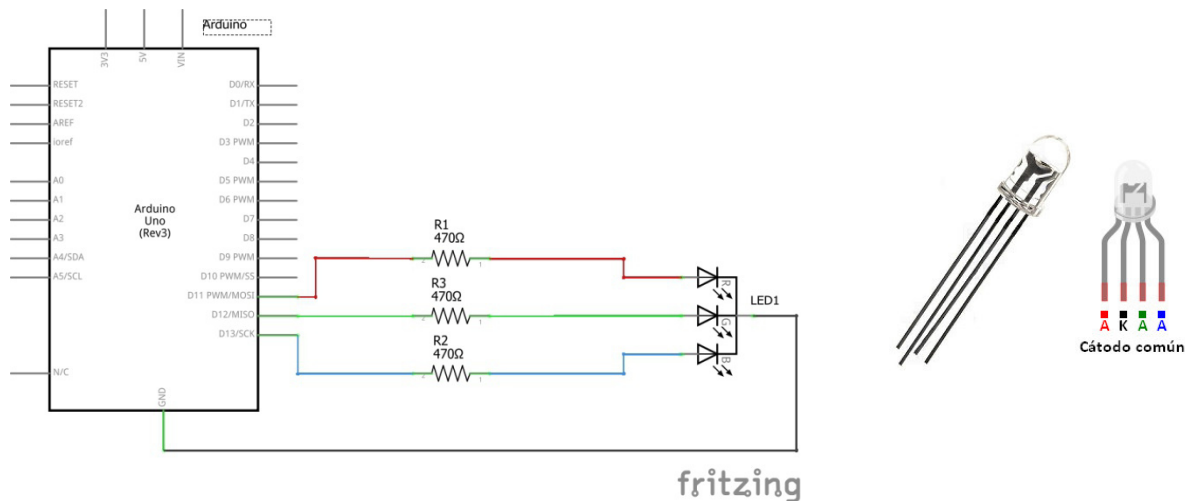
En la programación en S4A se va a realizar un programa en el que activaremos la salida 13 durante 3 segundos (diodo lucirá verde), todo apagado durante 1 segundo, activaremos la salida 11 durante 3 segundos (diodo lucirá rojo), todo apagado durante 1 segundo, activaremos ambas salidas durante 3 segundos (diodo lucirá amarillo), todo apagado durante 1 segundo y se volverá a repetir el ciclo por siempre. Como vemos, podemos conseguir tres colores diferentes (diodo tricolor), dos de ellos directamente (verde y rojo) y el otro por combinación de los dos colores anteriores (amarillo).





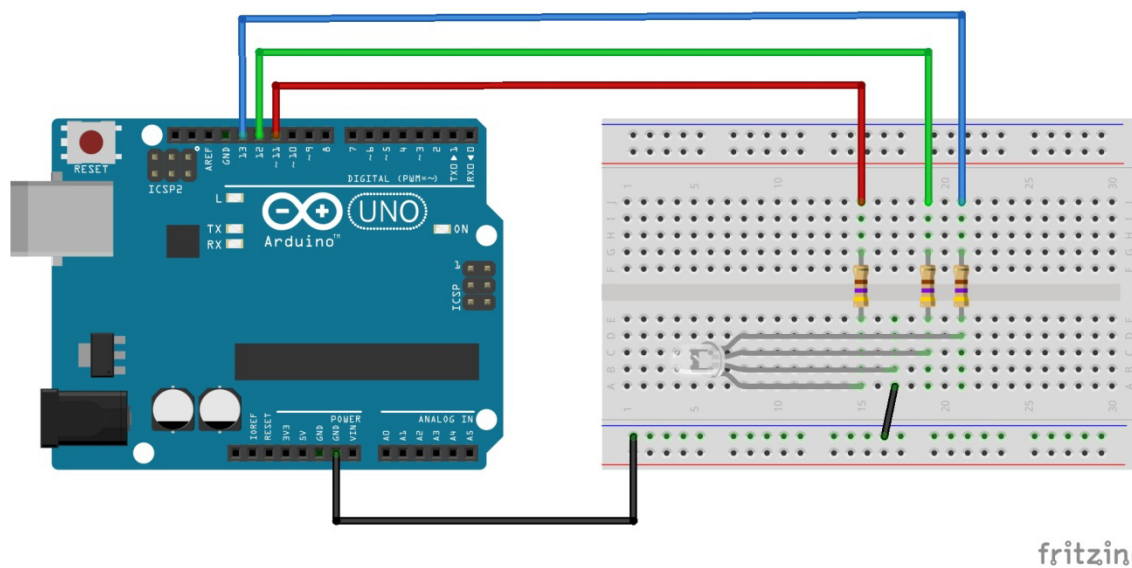
### 8. Control de un diodo RGB mediante salidas digitales.

El esquema eléctrico del montaje a realizar para la práctica junto con el esquema de las patillas del diodo RGB (Red – Green – Blue) <sup>18</sup> son los siguientes:



Aprovecharemos para explicar la diferencia entre ánodo común y cátodo común, tanto en un diodo bicolor, en un diodo RGB y en un display de 7 segmentos.

El montaje se realizará como podemos ver en la siguiente figura:



La programación en S4A se va a basar en un programa en el que vamos a jugar con las tres salidas que alimentan al diodo RGB, activándolas de forma secuencial siguiendo una tabla de verdad hasta completar las 8 combinaciones posibles ( $2^3 = 8$ ). Las salidas las mantendremos activadas durante 3 segundos y las desactivaremos durante 1 segundo, con el fin de observar los diferentes colores de la luz que obtenemos en el diodo RGB.

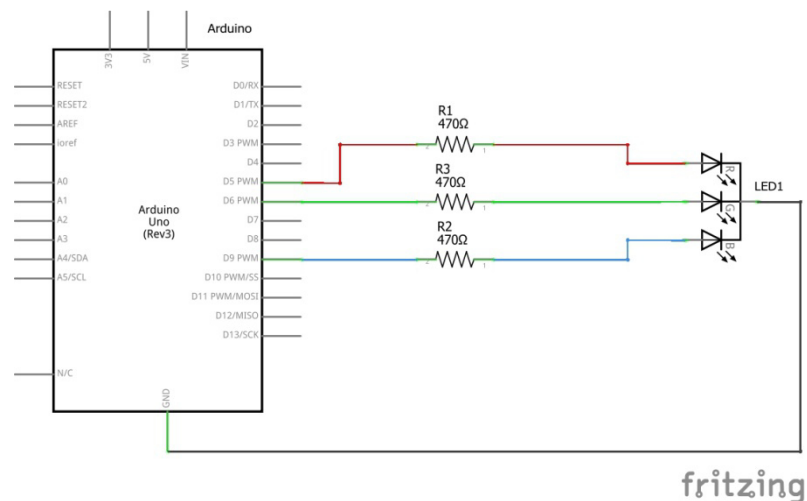


Una vez programado los alumnos observarán el funcionamiento y completarán la tabla de la verdad con los colores de la luz que ellos consideren que se obtienen.

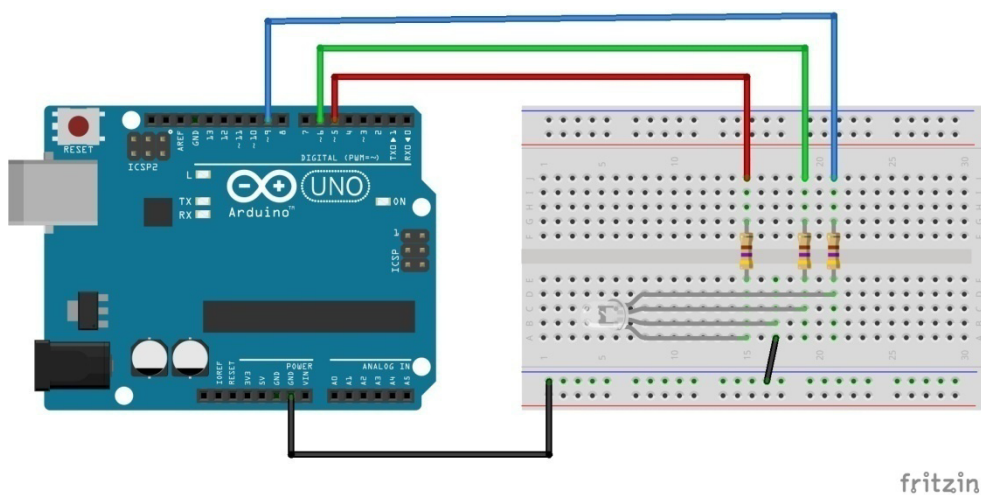
R (11)	G (12)	B (13)	Color RGB
0	0	0	Negro
0	0	1	Azul
0	1	0	Verde
0	1	1	Cian
1	0	0	Rojo
1	0	1	Magenta
1	1	0	Amarillo
1	1	1	Blanco

**9. Control de un diodo RGB mediante salidas analógicas.**

El esquema eléctrico del montaje a realizar para la práctica es el siguiente:



El montaje se realizará como podemos ver en la siguiente figura:



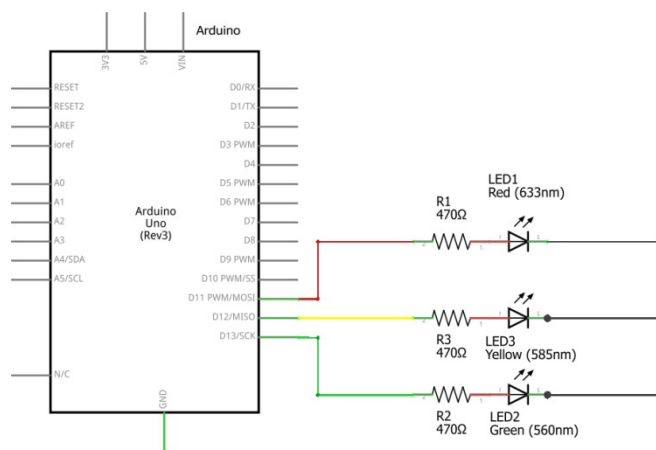
La programación en S4A es muy sencilla y consiste en activar las salidas analógicas para ir detectando la variación en el color de la luz que emite el diodo RGB

en función de las salidas que activemos, y la intensidad de la luz en función del valor de la salida analógica.



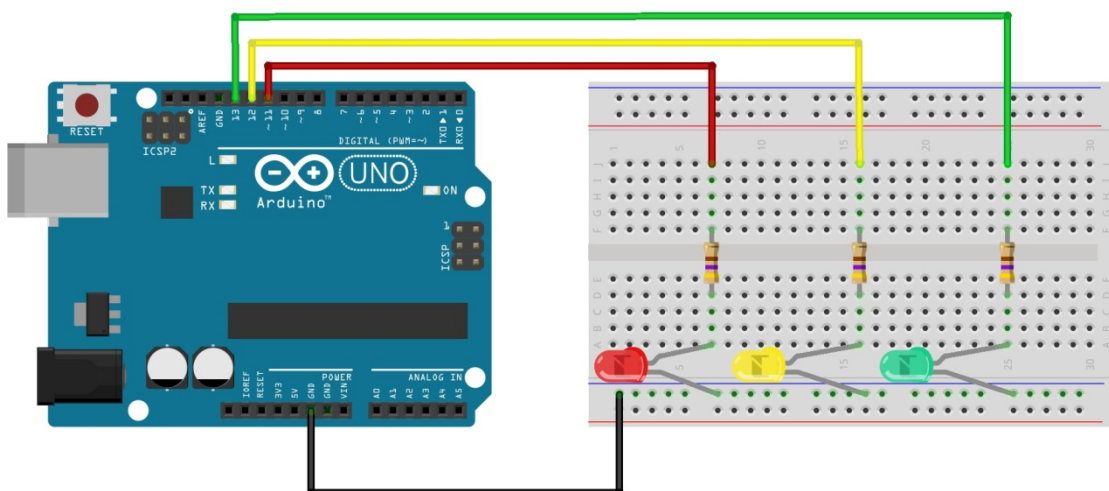
**10. Control de un semáforo de vehículos usando salidas digitales.**

El esquema eléctrico del montaje a realizar para la práctica es el siguiente:



fritzing

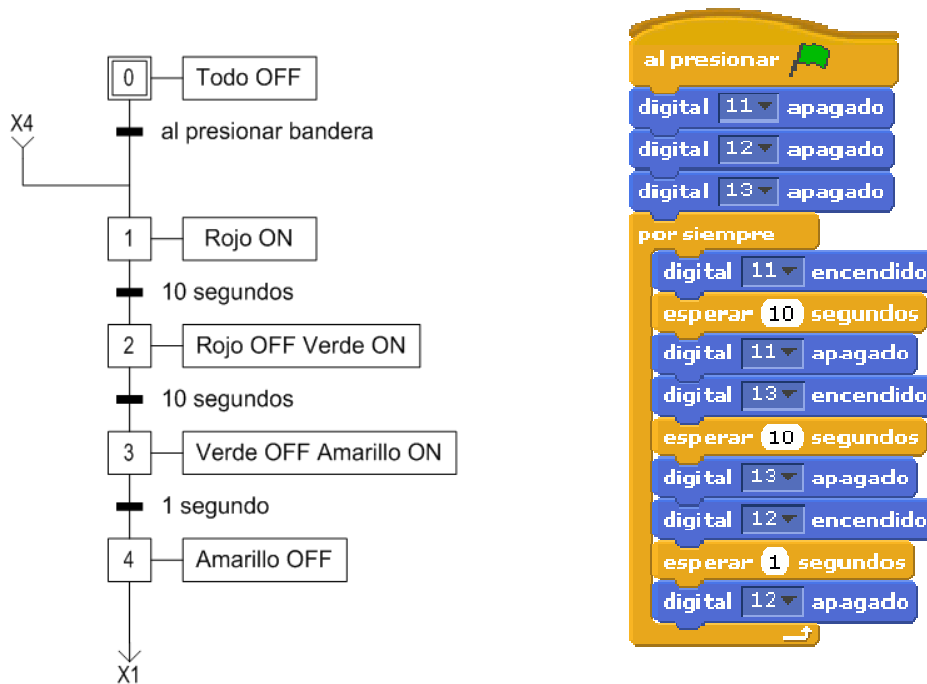
El montaje se realizará como podemos ver en la siguiente figura:



fritzing

A la hora de definir el funcionamiento de nuestro semáforo recurriremos a gráficos en grafcet, ya que se considera una forma muy sencilla y gráfica de explicar y

definir el funcionamiento de sistemas secuenciales. A continuación se muestra el gracet explicativo del funcionamiento y el programa en S4A.



Para la monitorización:

Objeto Arduino:



Objeto led rojo:



Objeto led amarillo:

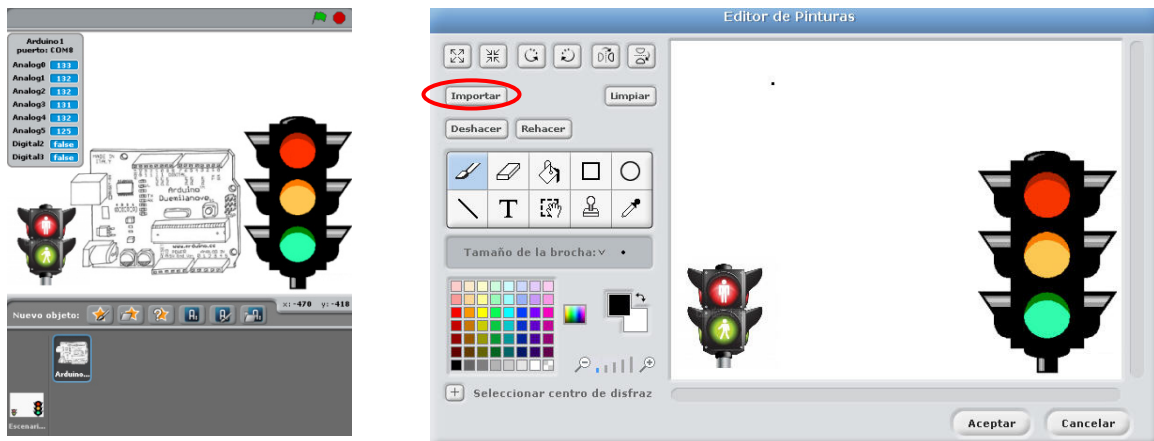


Objeto led verde:



Es interesante saber que el escenario se puede editar exactamente igual que cualquier objeto o disfraz de objeto. De forma que partiendo de imágenes en formato

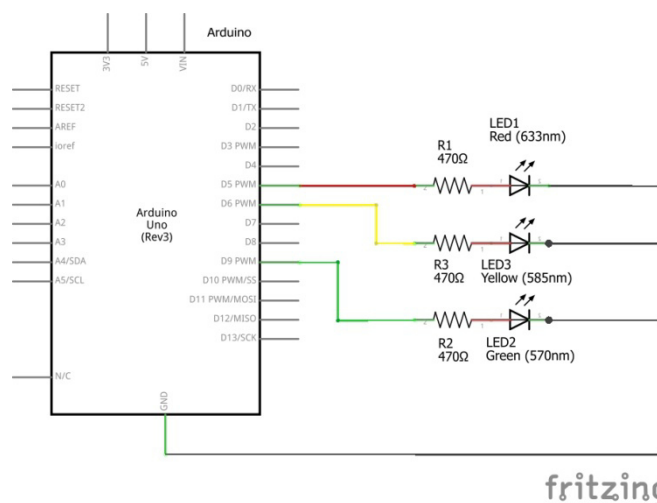
gif, jpg, ... podemos personalizar de forma rápida y sencilla el escenario, usando el botón importar dentro del editor de pinturas.



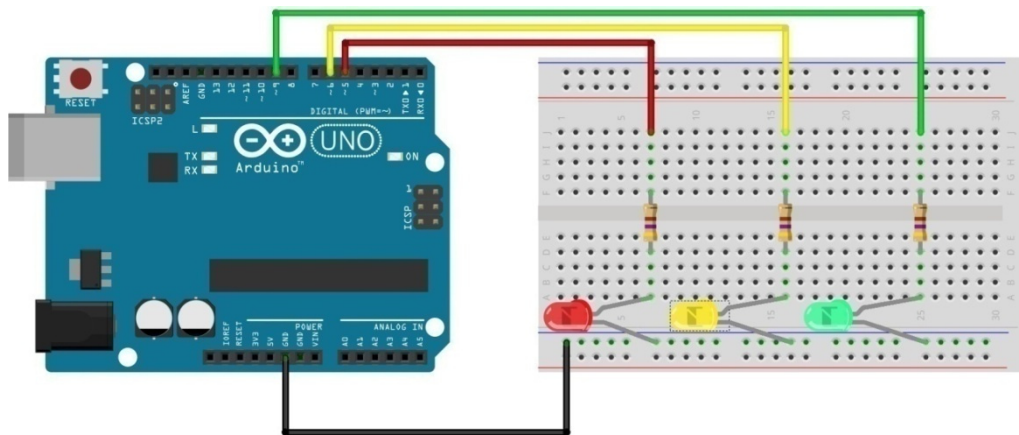
### 11. Control de un semáforo de vehículos usando salidas analógicas.

La práctica es la misma que la anterior pero usando salidas analógicas para alimentar los diodos led que simulan el semáforo. Para ello fundamentalmente hay que aplicar lo visto en la práctica 4.

El esquema eléctrico del montaje a realizar para la práctica es el siguiente:

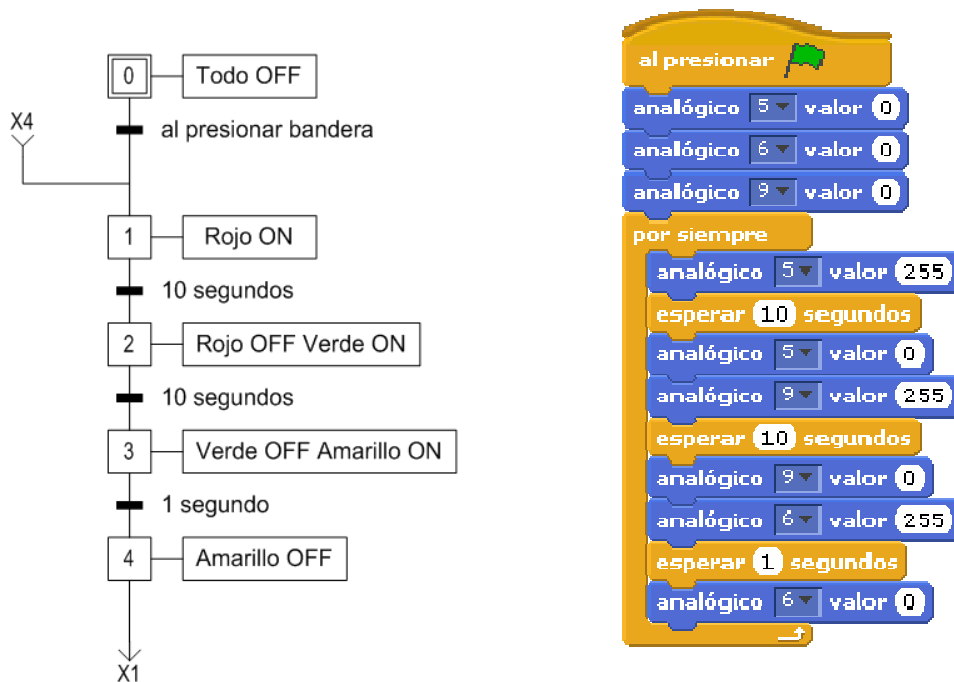


El montaje se realizará como podemos ver en la siguiente figura:



fritzing

El graficet explicativo del funcionamiento y el programa en S4A son los siguientes:

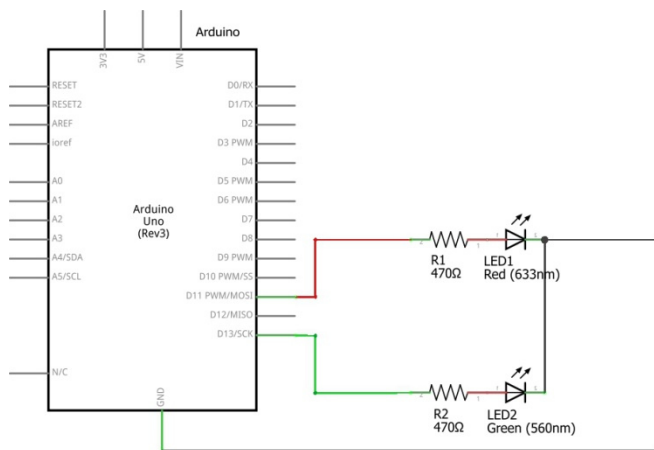


La monitorización se realiza exactamente igual que la de la práctica anterior, adaptando el programa del objeto Arduino para trabajar con salidas analógicas, tal y como podemos ver en la parte superior.

### 12. Control de un semáforo de peatones.

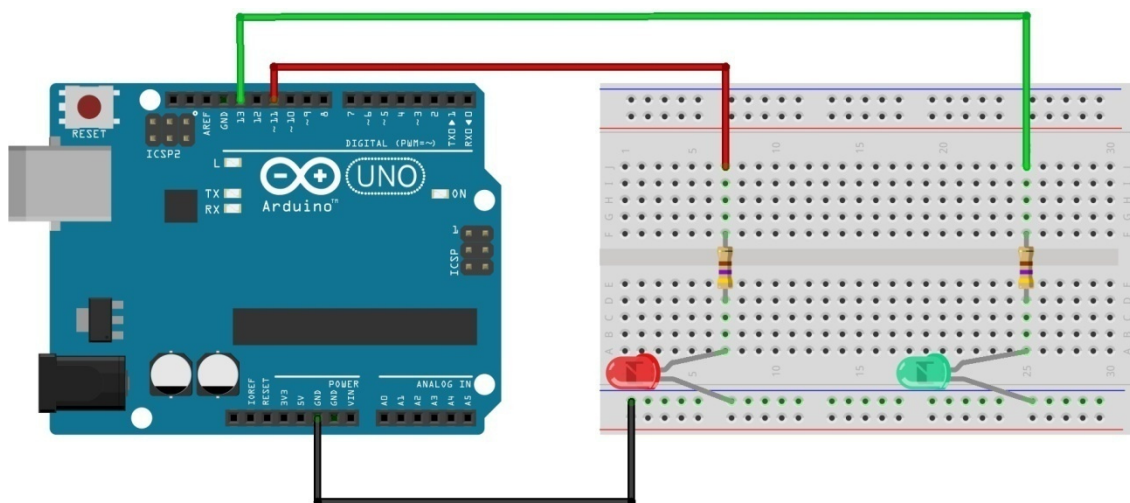
Esta práctica, exactamente igual que la del semáforo de vehículos, la podemos resolver usando salidas digitales o analógicas. La vamos a resolver en primer lugar con salidas digitales y posteriormente con analógicas.

El esquema eléctrico es el siguiente:



fritzing

El montaje se realizará como podemos ver en la siguiente figura:



fritzing

A continuación se realizará la programación en S4A, del funcionamiento de la práctica, basada en el grafcet explicativo del funcionamiento y que podemos ver a continuación. Para ello, como bloque no visto en las prácticas anteriores, se utilizará el bloque “repetir \_\_\_” para repetir la acción del parpadeo del diodo led verde el número de veces necesario, acortando la extensión del programa. El bloque es de la librería control.

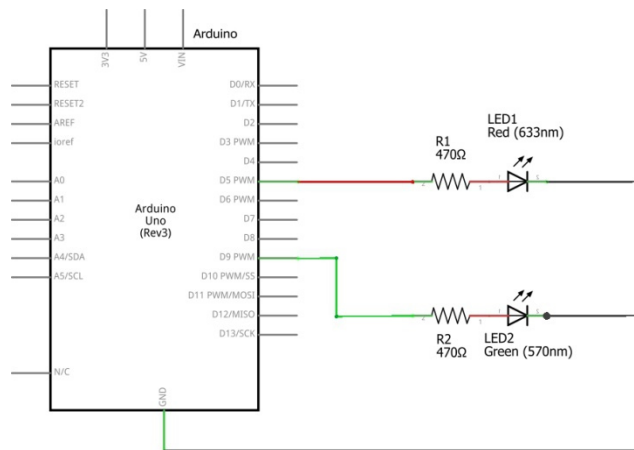




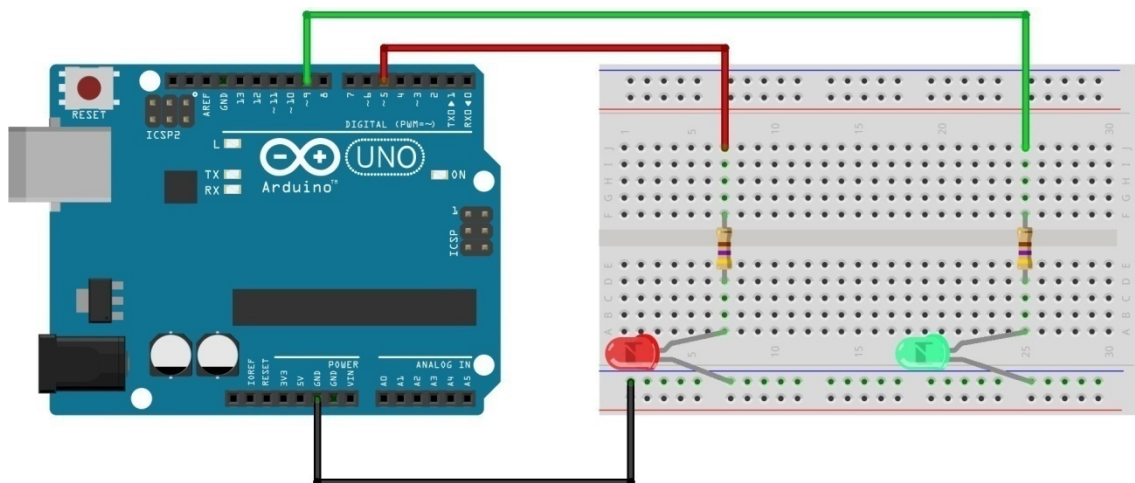
Recordar que al pinchar sobre el número de repetir podemos modificarlo según nuestras necesidades. El graficet de funcionamiento y la programación en S4A:



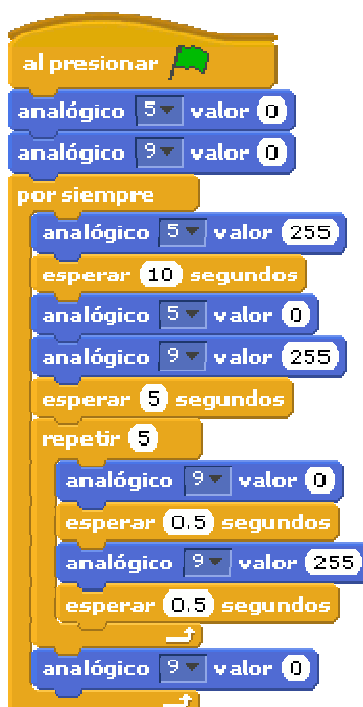
Usando las salidas analógicas:



fritzing



fritzing

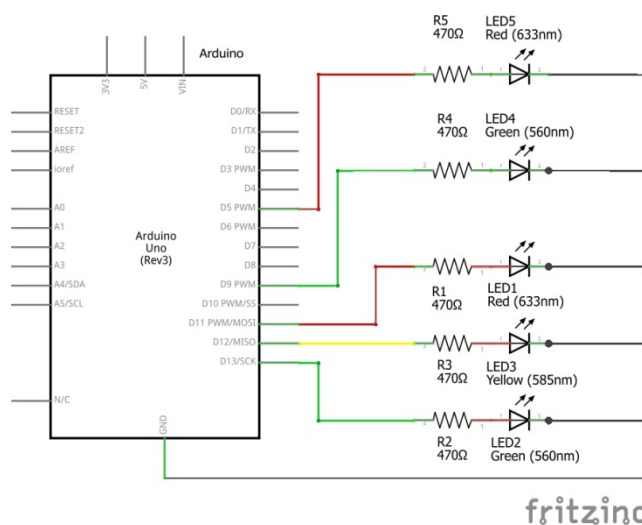


Sería muy interesante que los alumnos realizaran la monitorización para ir afianzando los conocimientos. Se realiza exactamente igual que en las prácticas anteriores.

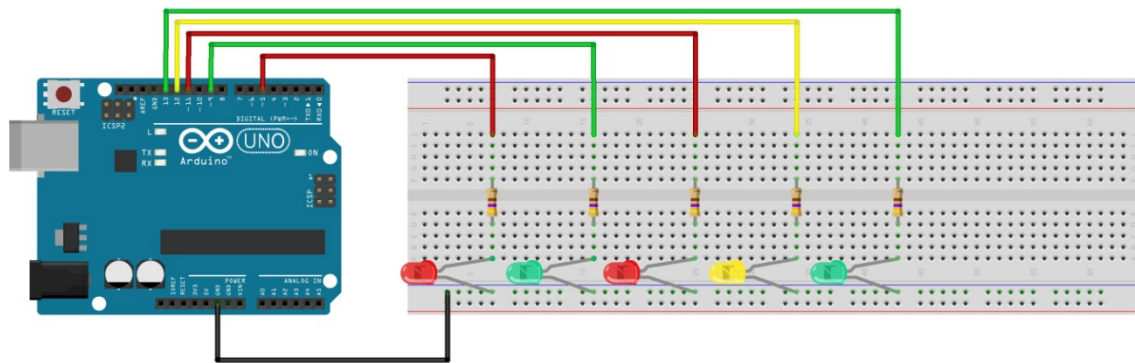
### 13. Control conjunto de un semáforo de vehículos y de un semáforo de peatones.

Como la placa Arduino Uno Rev3 programada a través de S4A solo dispone de 4 salidas digitales y necesitamos 5 salidas, controlaremos el semáforo de vehículos con salidas digitales y el de peatones con salidas analógicas.

El esquema eléctrico es el siguiente:

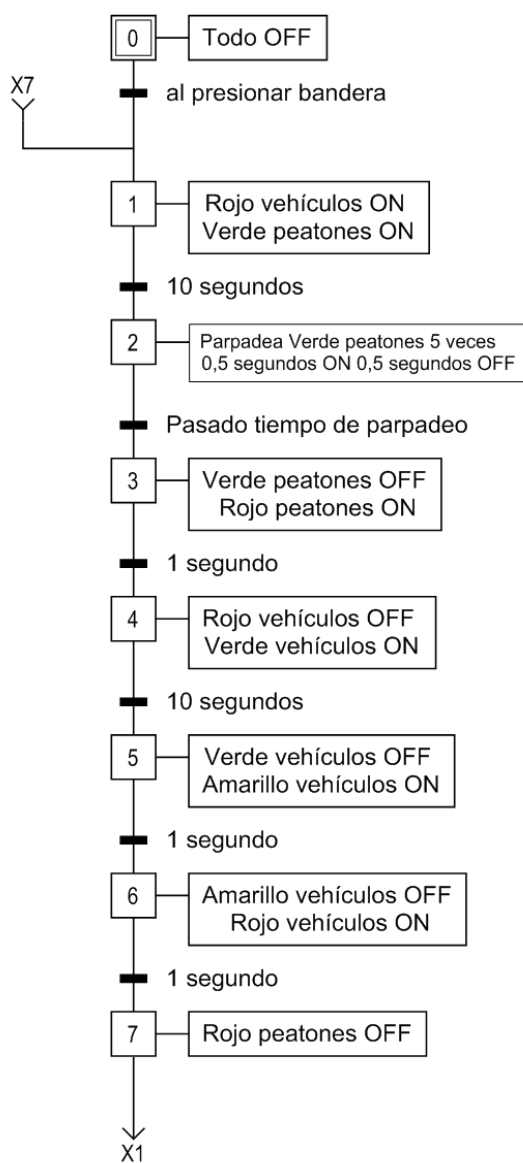


El montaje se realizará como podemos ver en la siguiente figura:



fritzing

El graficet explicativo del funcionamiento y el programa en S4A son los siguientes:



```

al presionar
digital 11 apagado
digital 12 apagado
digital 13 apagado
analógico 5 valor 0
analógico 9 valor 0

por siempre
digital 11 encendido
analógico 9 valor 255
esperar 10 segundos
repetir 5
analógico 9 valor 0
esperar 0,5 segundos
analógico 9 valor 255
esperar 0,5 segundos
analógico 9 valor 0
analógico 5 valor 255
esperar 1 segundos
digital 11 apagado
digital 13 encendido
esperar 10 segundos
digital 13 apagado
digital 12 encendido
esperar 1 segundos
digital 12 apagado
digital 11 encendido
esperar 1 segundos
analógico 5 valor 0
    
```

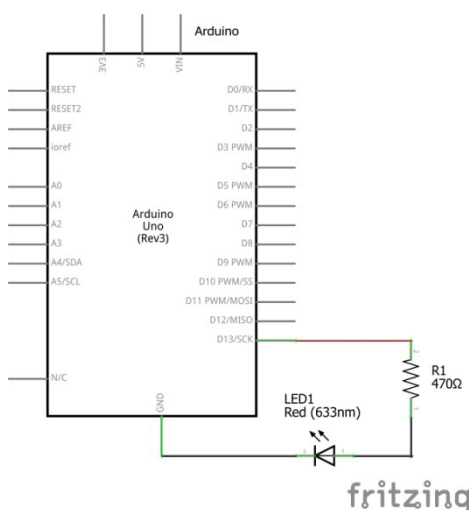
Posteriormente los alumnos realizarán la monitorización completa con escenario personalizado, trabajando tal y como hemos visto en las prácticas anteriores.

Esta práctica se puede completar modificando el graficet de funcionamiento, de forma que inicialmente, cuando se pone en marcha el programa, todos los led rojos (rojo de vehículos y rojo de peatones) estén encendidos durante 1 segundo, con el fin de que durante un tiempo todo el tráfico, tanto de vehículos como de peatones, se detenga con el fin de evitar accidentes al poner en marcha el programa.

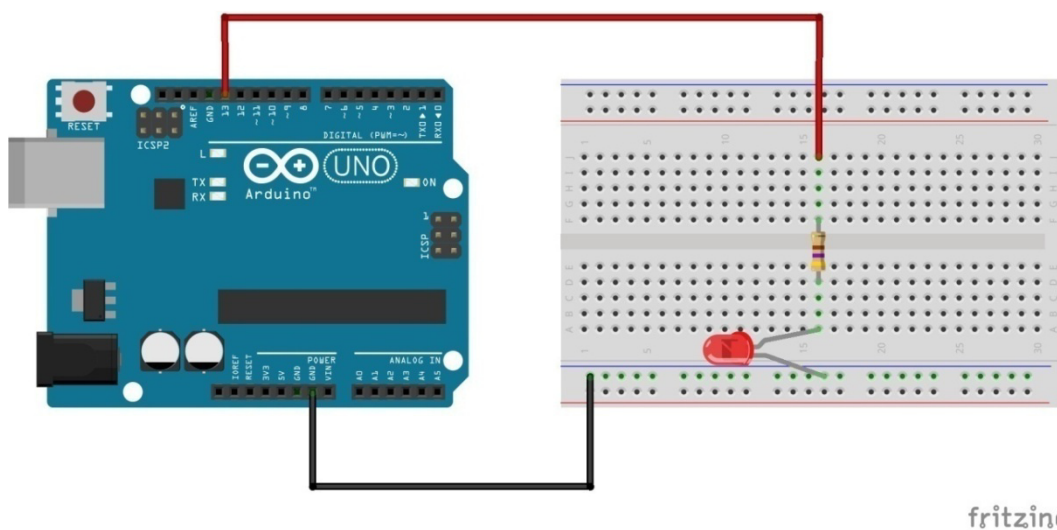
También podemos completar la práctica adaptando el semáforo para personas invidentes, de forma que cuando el semáforo de peatones está verde, sonará de forma rápida el sonido de un ave piando, cuando el semáforo de peatones se ponga intermitente, sonará de forma más lenta el mismo sonido de ave piando y cuando el semáforo de peatones esté rojo, no sonará nada.

### 14. Control del encendido y apagado de un diodo led mediante un cronómetro.

El esquema eléctrico es el siguiente:



El esquema de montaje:



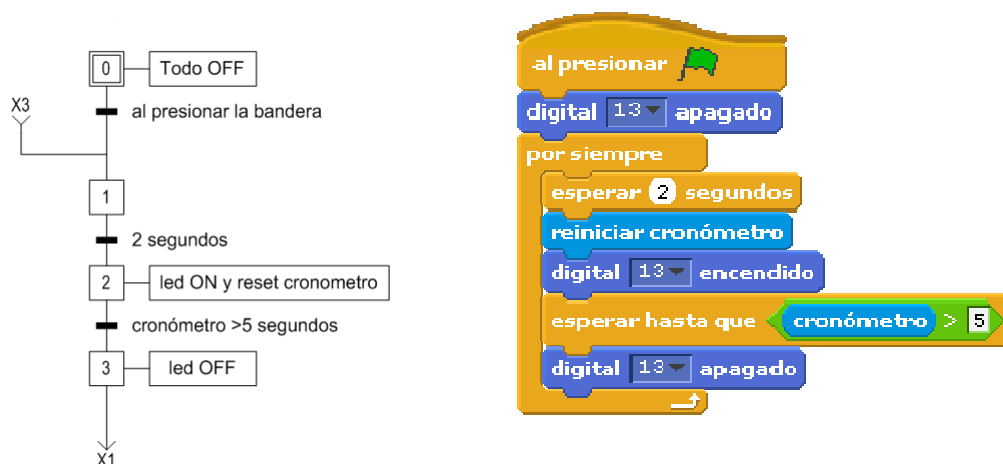
En la programación en S4A como bloques no vistos en las prácticas anteriores, se utilizará el bloque perteneciente a la librería de control “esperar hasta que \_\_\_” para mantener detenido el programa hasta que ocurra algo. También vamos a utilizar dos nuevos bloques de la biblioteca de sensores. Estos son “cronómetro” para usarlo cuando y donde sea necesario y “reiniciar cronómetro” para comenzar la cuenta. También se utilizará el bloque perteneciente a la librería de operadores “\_\_\_ > \_\_\_” (mayor que) para poder comparar un valor con otro. En este caso irá acompañado del bloque “cronómetro”.



Para colocar el bloque “cronómetro” tenemos que cogerlo con el ratón y desplazarlo encima de uno de los rectángulos que hay en el bloque “\_\_\_ > \_\_\_”. Cuando nos acercamos a la zona del rectángulo vemos que se activa y se resalta en color blanco, momento en el que podemos soltar el bloque.



A continuación podemos ver el graficet explicativo del funcionamiento y el programa en S4A:

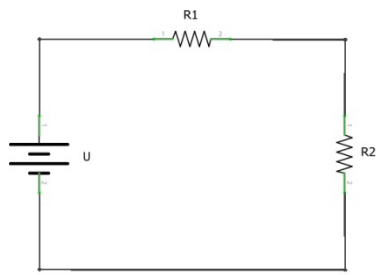


### 15. Control del encendido y apagado de un diodo led mediante un pulsador usando una entrada digital.

Hasta ahora solo habíamos sacado señales de la placa Arduino Uno. A partir de esta práctica vamos a aprender a introducir señales. Primero empezaremos con señales digitales y luego pasaremos a señales analógicas.

En primer lugar hay que explicar una serie de conocimientos muy básicos y sencillos, que para los alumnos serán de repaso y aplicación de lo visto en la asignatura de Tecnología en otros cursos de la etapa.

El primer concepto es el de divisor de tensión:



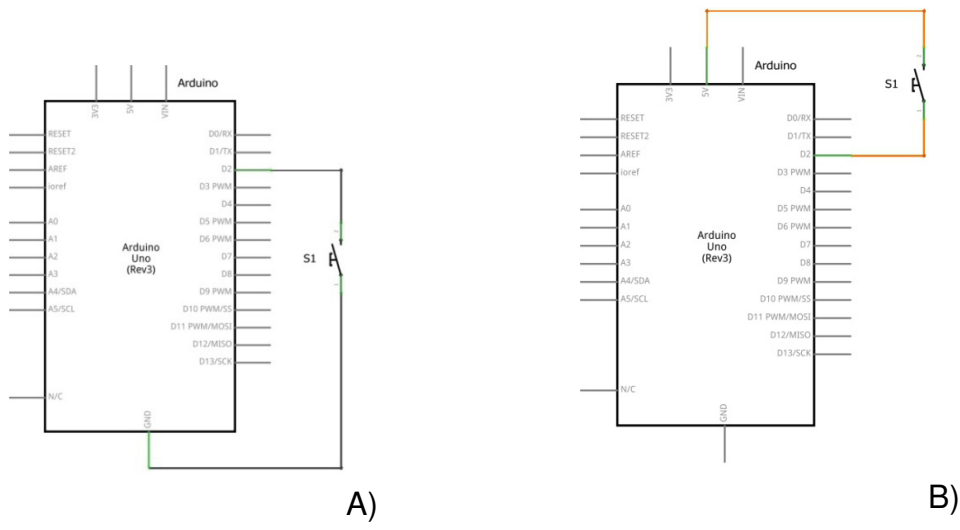
$$R_T = R_1 + R_2$$

$$I_T = \frac{U}{R_T} ; I_T = I_1 = I_2$$

$$U_1 = R_1 \cdot I_1 = R_1 \cdot \frac{U}{R_T} = U \cdot \frac{R_1}{R_1 + R_2}$$

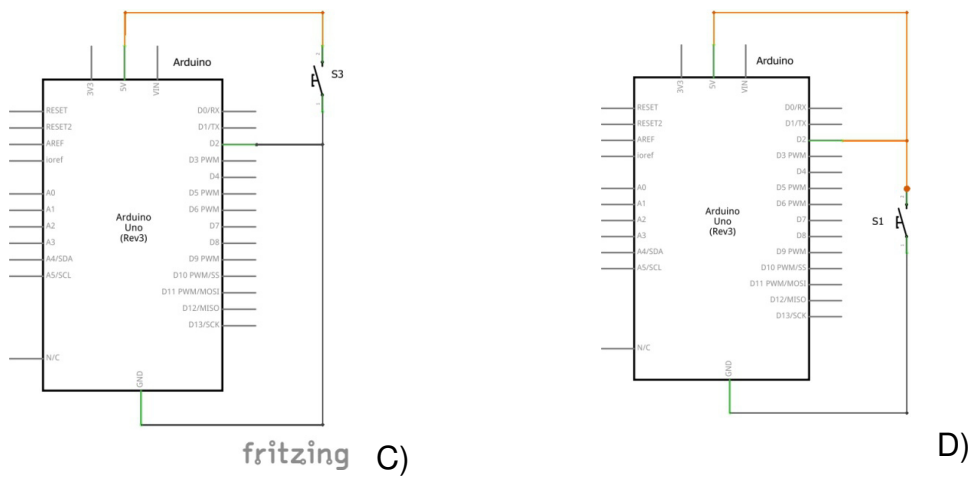
$$U_2 = R_2 \cdot I_2 = R_2 \cdot \frac{U}{R_T} = U \cdot \frac{R_2}{R_1 + R_2}$$

Repasados estos sencillos conceptos de la simplificación y cálculo de un circuito de resistencias en serie, pasamos a ver cuatro formas incorrectas de introducir señales digitales.



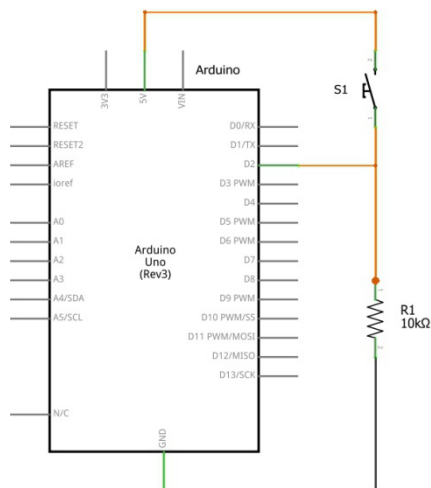
En el caso A), cuando pulsemos el pulsador S<sub>1</sub> llegarán 0V a la entrada D2, pero cuando lo soltemos la tensión que llegará a la entrada D2 será indeterminada y por lo tanto se producirán errores.

En el caso B), cuando pulsemos el pulsador S<sub>1</sub> llegarán 5V a la entrada D2, pero cuando lo soltemos la tensión que llegará a la entrada D2 será indeterminada y por lo tanto se producirán errores.

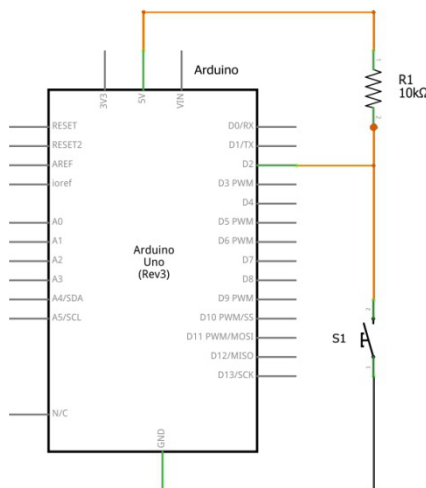


En el caso C), sin pulsar llegarán 0V a la entrada D2 y pulsando cortocircuitaremos la salida de 5V de la fuente de alimentación. En el D), cuando no pulsemos el pulsador S<sub>1</sub> llegarán 5V a la entrada D2, pero cuando lo pulsemos cortocircuitaremos la salida de 5V de la fuente de alimentación de Arduino Uno.

La solución es usar estos dos últimos circuitos pero limitando la corriente con una resistencia de 10kΩ. A continuación podemos ver la forma correcta de introducir señales digitales en Arduino Uno.



Pull-down o lógica directa



Pull-up o lógica inversa

En pull-down o lógica directa, cuando el pulsador está sin pulsar a la entrada D2 llegan exactamente 0V (no pulsado → 0 lógico). Cuando el pulsador está pulsado, a la entrada D2 llegan exactamente 5V (pulsado → 1 lógico) y R<sub>1</sub> limita la corriente para evitar el cortocircuito, reduciéndola a una corriente muy pequeña, casi despreciable.

Hay que hacer ver a los alumnos que esto es una aplicación del divisor de tensión. Suponiendo que R<sub>2</sub> es la resistencia del pulsador (Pulsador sin pulsar → R<sub>2</sub> = ∞ Ω, pulsador pulsado → R<sub>2</sub> = 0 Ω) y que estamos introduciendo a través de la entrada D2 la tensión en la resistencia R<sub>1</sub>.

$$\text{Pulsador sin pulsar: } U_1 = U \cdot \frac{R_1}{R_1 + R_2} = 5V \cdot \frac{10k\Omega}{10k\Omega + \infty k\Omega} \cong 0V$$

$$\text{Pulsador pulsado: } U_1 = U \cdot \frac{R_1}{R_1 + R_2} = 5V \cdot \frac{10k\Omega}{10k\Omega + 0\Omega} = 5V$$

En pull-up o lógica inversa, cuando el pulsador está sin pulsar a la entrada D2 llegan exactamente 5V (no pulsado → 1 lógico). Cuando el pulsador está pulsado, a la entrada D2 llegan exactamente 0V (pulsado → 0 lógico) y R<sub>1</sub> limita la corriente para evitar el cortocircuito, reduciéndola a una corriente muy pequeña, casi despreciable.

Hay que hacer ver a los alumnos que esto es una aplicación del divisor de tensión. Suponiendo que R<sub>2</sub> es la resistencia del pulsador (Pulsador sin pulsar → R<sub>2</sub> =

$\infty \Omega$ , pulsador pulsado  $\rightarrow R_2 = 0 \Omega$ ) y que estamos introduciendo a través de la entrada D2 la tensión en el pulsador  $R_2$ .

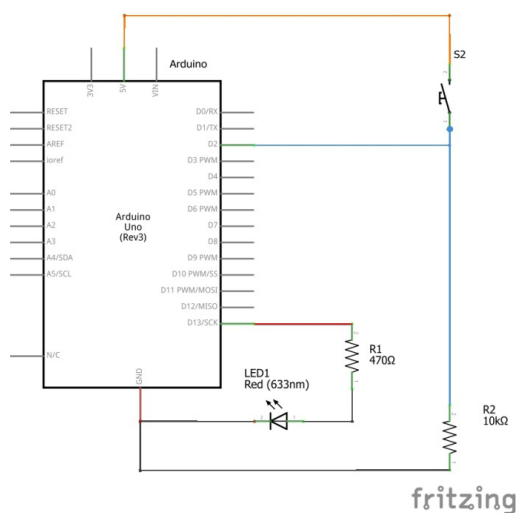
$$\text{Pulsador sin pulsar: } U_2 = U \cdot \frac{R_2}{R_1 + R_2} = 5V \cdot \frac{\infty k\Omega}{10k\Omega + \infty k\Omega} \cong 5V$$

$$\text{Pulsador pulsado: } U_2 = U \cdot \frac{R_2}{R_1 + R_2} = 5V \cdot \frac{0k\Omega}{10k\Omega + 0\Omega} = 0V$$

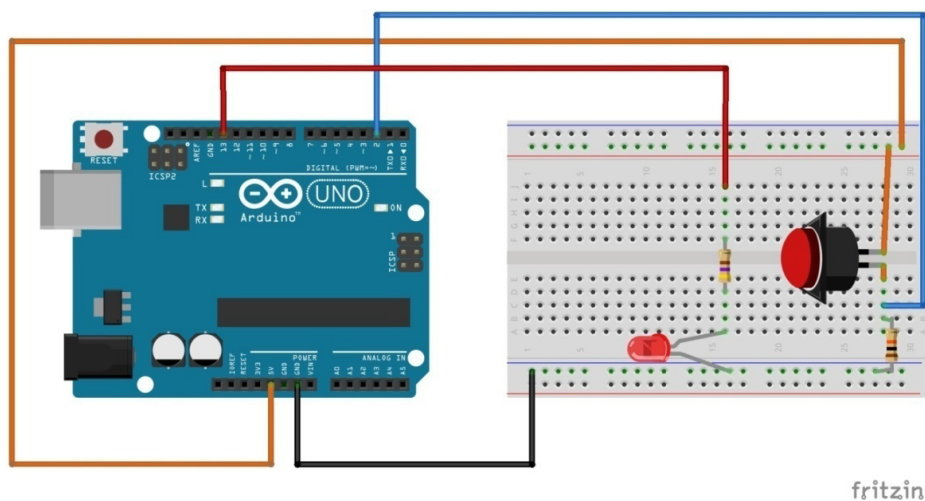
Como hemos podido observar, para introducir señales, tanto analógicas como posteriormente digitales y para alimentar pequeñas cargas, podemos utilizar la salida de 5V que tiene la fuente de alimentación de la placa Arduino Uno.

En esta práctica, vamos a montar varios elementos accionadores con los que podemos introducir señales digitales a Arduino Uno. Estos elementos van a ser pulsadores con contactos normalmente abiertos o normalmente cerrados, finales de carrera con contacto conmutado e interruptor reed.

Una vez aclarados estos conceptos, el esquema eléctrico de la práctica con **pulsador normalmente abierto (NO) y sistema pull-down o lógica directa:**



El montaje se realizará como podemos ver en la siguiente figura:





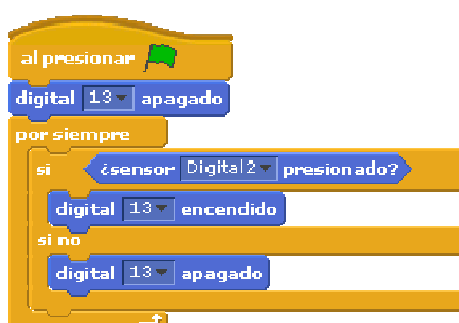
A continuación se realizará la programación en S4A. Para ello, como bloques no vistos en las prácticas anteriores, se utilizará el bloque de la librería control “si \_\_ si no” para realizar diferentes acciones en función de lo que nos diga el complemento que coloquemos para controlar el bloque. Y de la librería movimiento utilizaremos el bloque “¿sensor digital \_\_ presionado?” (al presionar sobre la flecha podemos elegir la entrada digital deseada).



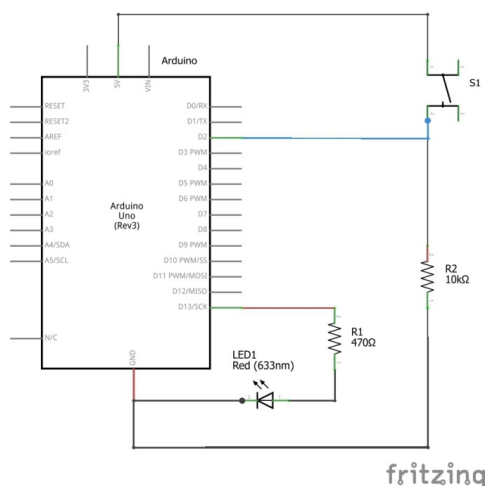
Para colocar el bloque “¿sensor digital \_\_ presionado?” tenemos que cogerlo con el ratón y desplazarlo encima del hexágono irregular que hay en el bloque “si \_\_ si no”. Cuando nos acercamos a la zona del hexágono vemos que se activa y se resalta en color blanco, momento en el que podemos soltar el bloque.



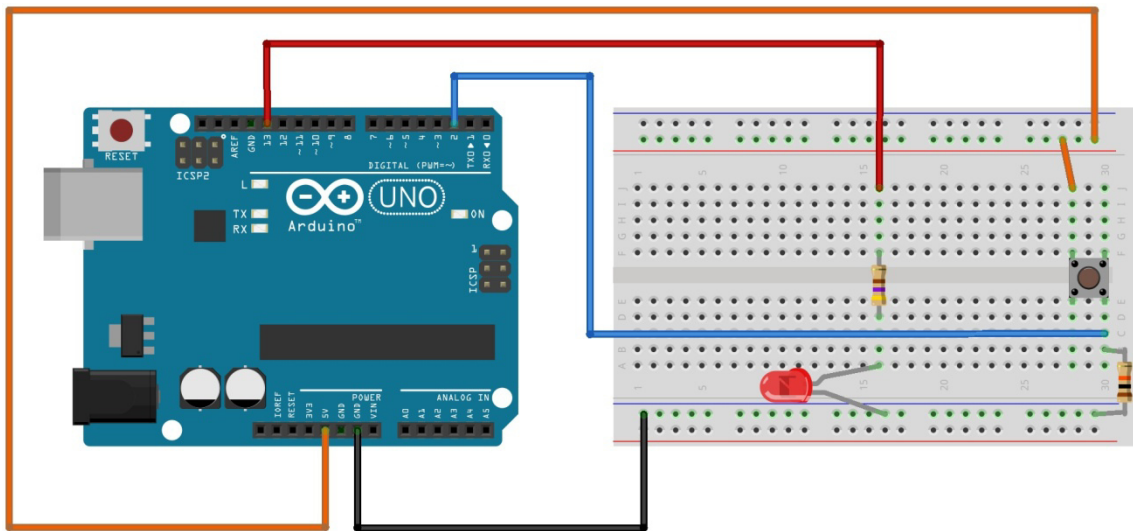
El programa en S4A es el siguiente:



Sería muy interesante hacer la práctica utilizando también la entrada digital 3. Con un pulsador para circuito integrado, el esquema eléctrico es el siguiente:

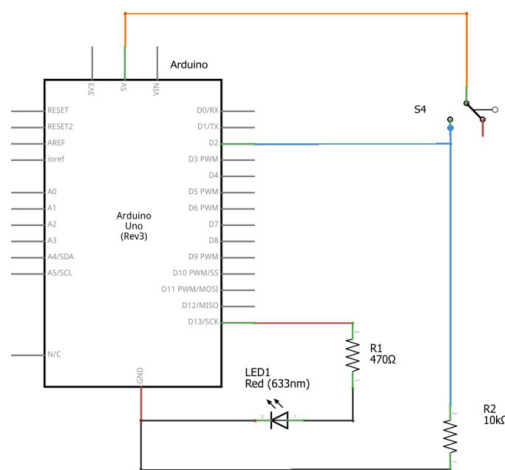


Y el esquema de conexionado y montaje:

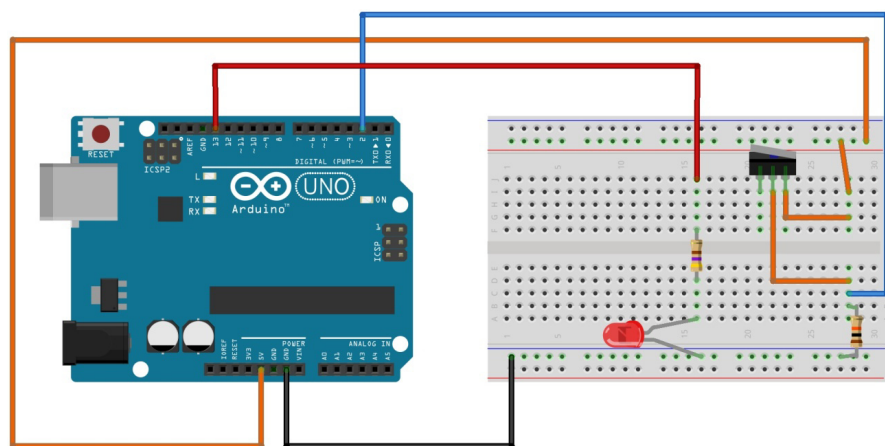


fritzing

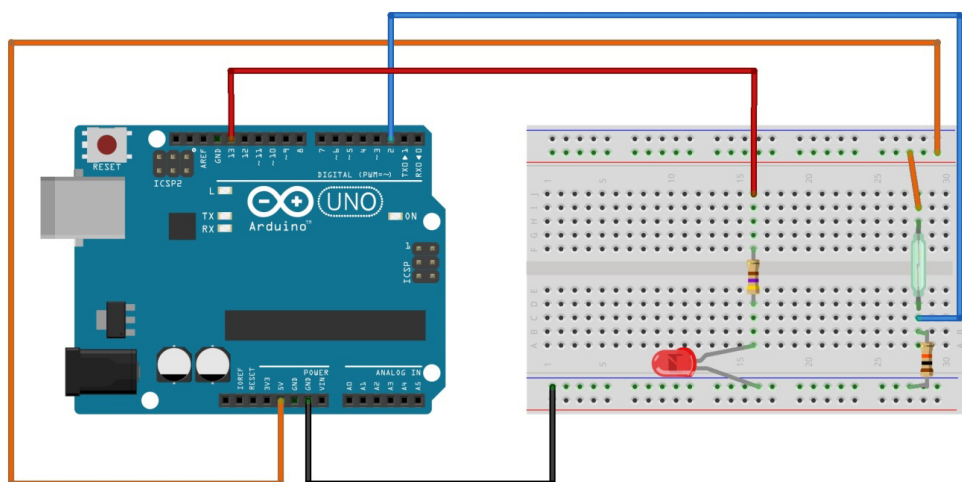
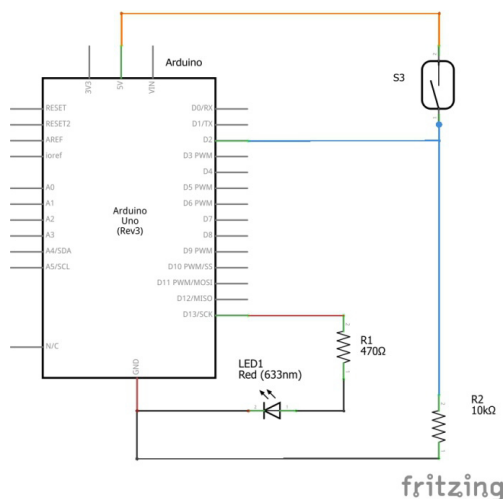
A continuación y como hemos comentado antes, se van a poner los esquemas eléctricos y de montaje o conexionado de la misma práctica pero con señales procedentes de un **final de carrera** y de un **interruptor reed**, con **contactos normalmente abiertos (NO)** y **pull-down** o **lógica directa**.



fritzing



fritzing

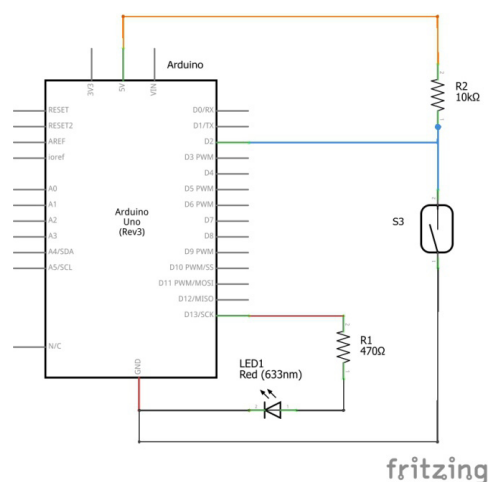


fritzing

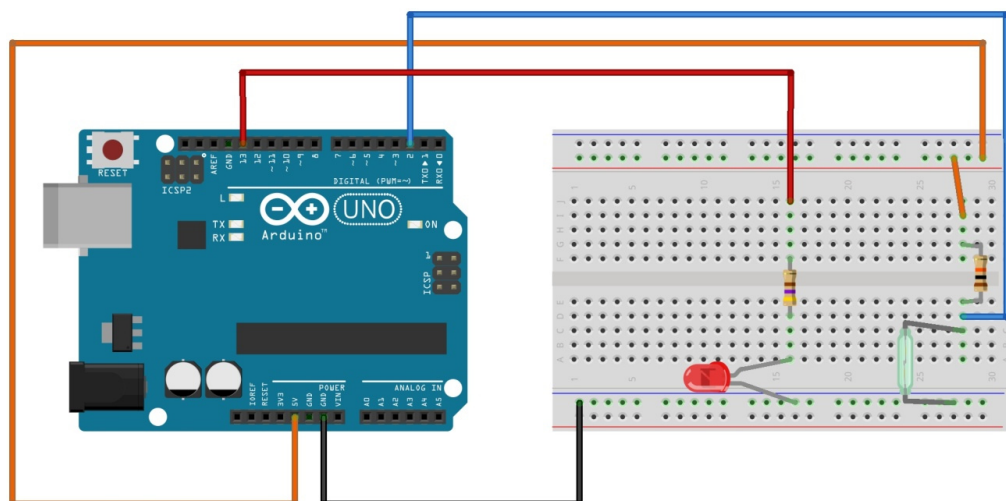
Para todos ellos, la programación en S4A es la misma que para el del pulsador.

Si observamos, con contactos normalmente abiertos (NO) y pull-down o lógica directa, cuando el contacto está sin accionar (abierto) a la entrada D2 llegan exactamente 0V (no pulsado → 0 lógico). Cuando el contacto está accionado, a la entrada D2 llegan exactamente 5V (pulsado → 1 lógico) y R<sub>1</sub> limita la corriente para evitar el cortocircuito, reduciéndola a una corriente muy pequeña, casi despreciable.

**Con lógica pull-up o lógica inversa para el interruptor reed (NO):**

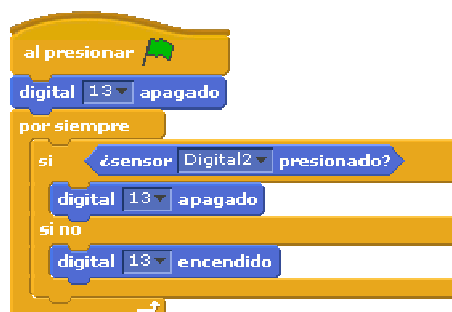


fritzing

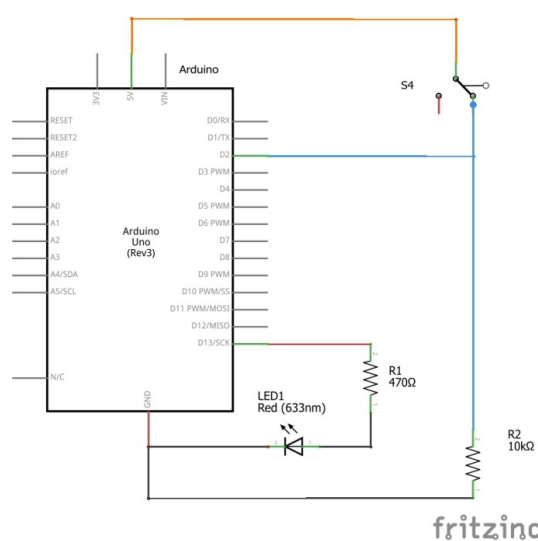


fritzing

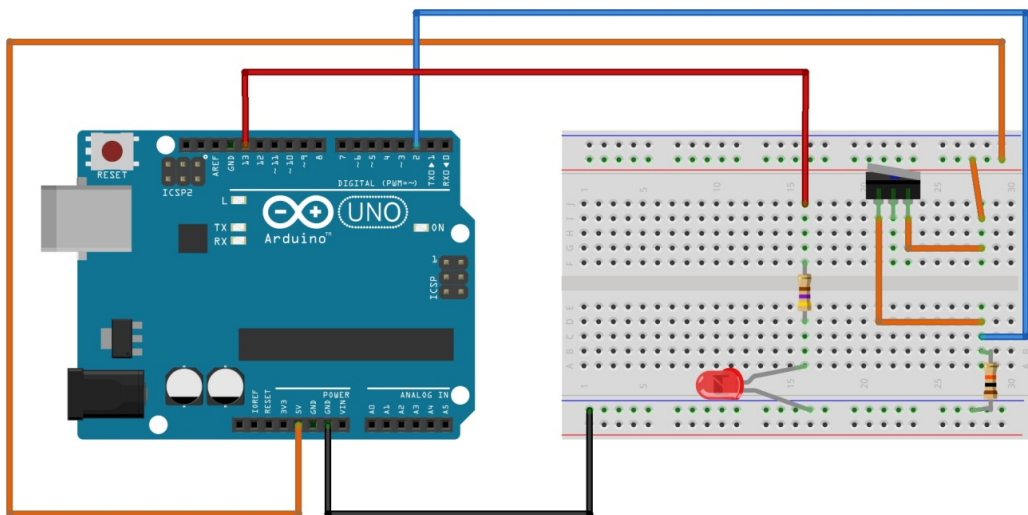
En pull-up o lógica inversa y contacto NO, cuando el interruptor reed está sin accionar por un campo magnético, a la entrada D2 llegan exactamente 5V (no accionado → 1 lógico). Cuando interruptor reed está accionado por un campo magnético, a la entrada D2 llegan exactamente 0V (accionado → 0 lógico) y R<sub>2</sub> limita la corriente para evitar el cortocircuito, reduciéndola a una corriente muy pequeña, casi despreciable.



Si se utilizara un final de carrera con **contactos normalmente cerrados y pull-down o lógica directa:**

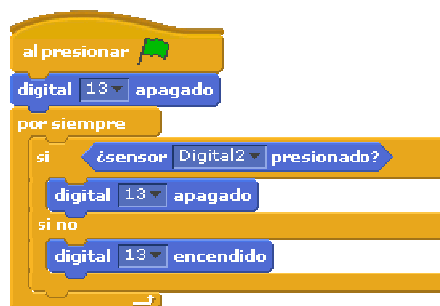


fritzing

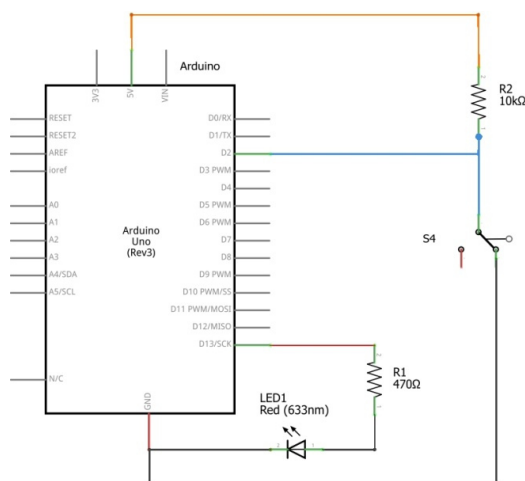


fritzing

Con contactos normalmente cerrados (NC) y pull-down o lógica directa, cuando el contacto está sin accionar (cerrado) a la entrada D2 llegan exactamente 5V (no pulsado → 1 lógico) y R<sub>1</sub> limita la corriente para evitar el cortocircuito, reduciéndola a una corriente muy pequeña, casi despreciable. Cuando el contacto está accionado, a la entrada D2 llegan exactamente 0V (pulsado → 0 lógico). Es por ello por lo que hay que cambiar la lógica del programa en S4A:

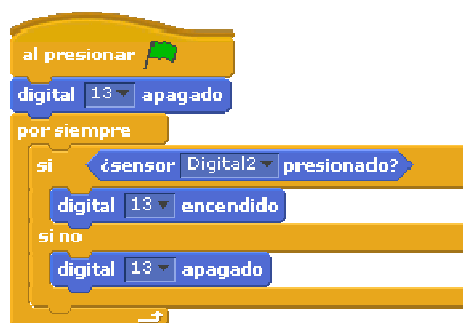
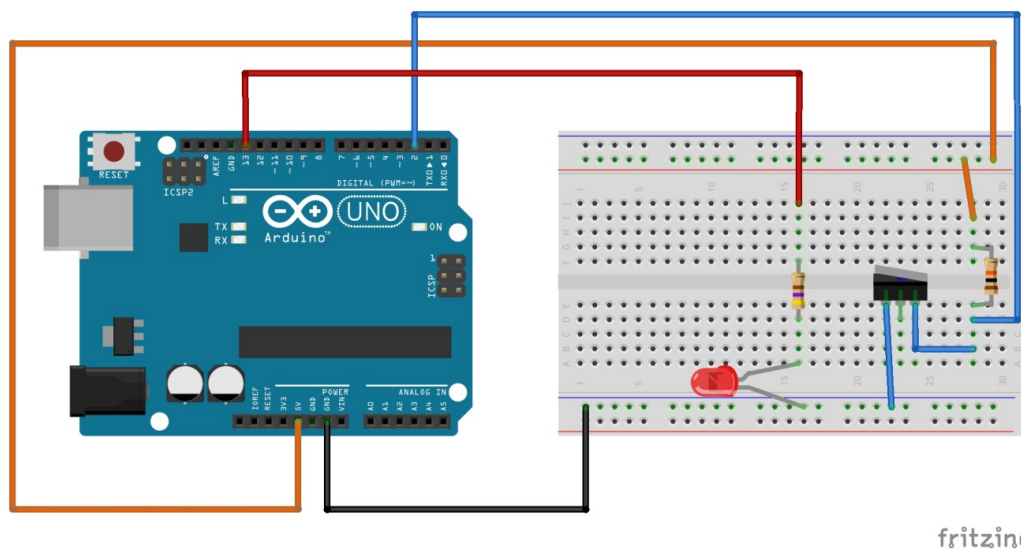


Y por último nos queda el caso de final de carrera con **contactos normalmente cerrados y pull-up o lógica inversa**:



fritzing

Con contactos normalmente cerrados (NC) y pull-up o lógica inversa, cuando el contacto está sin accionar (cerrado) a la entrada D2 llegan exactamente 0V (no accionado → 0 lógico) y  $R_1$  limita la corriente para evitar el cortocircuito, reduciéndola a una corriente muy pequeña, casi despreciable. Cuando el contacto está accionado, a la entrada D2 llegan exactamente 5V (accionado → 1 lógico).



También hay que recordar que S4A interactúa con Arduino enviando el estado de los actuadores y recibiendo el de los sensores cada 75 ms, por lo tanto el ancho de pulso ha de ser mayor que este período. Si el pulso es menor de 75 ms, puede ser que S4A no se entere de que ha habido una pulsación.

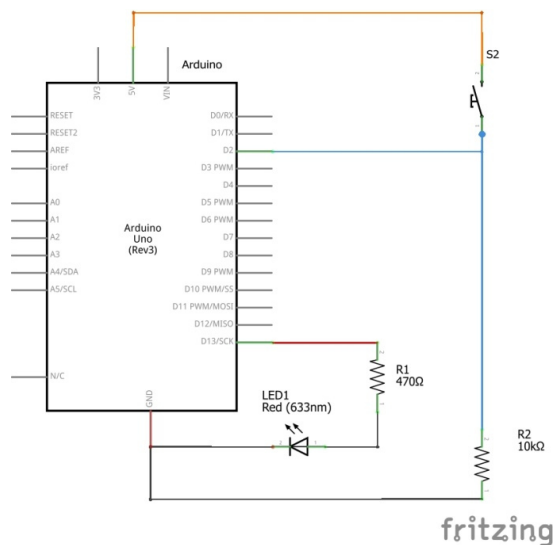
## 16. Control del encendido y apagado de un diodo led mediante un pulsador con retardo a la desconexión.

A partir de este momento y si no se indica lo contrario, utilizaremos pulsadores con el contacto normalmente abierto (NO) y sistema pull-down o lógica directa.

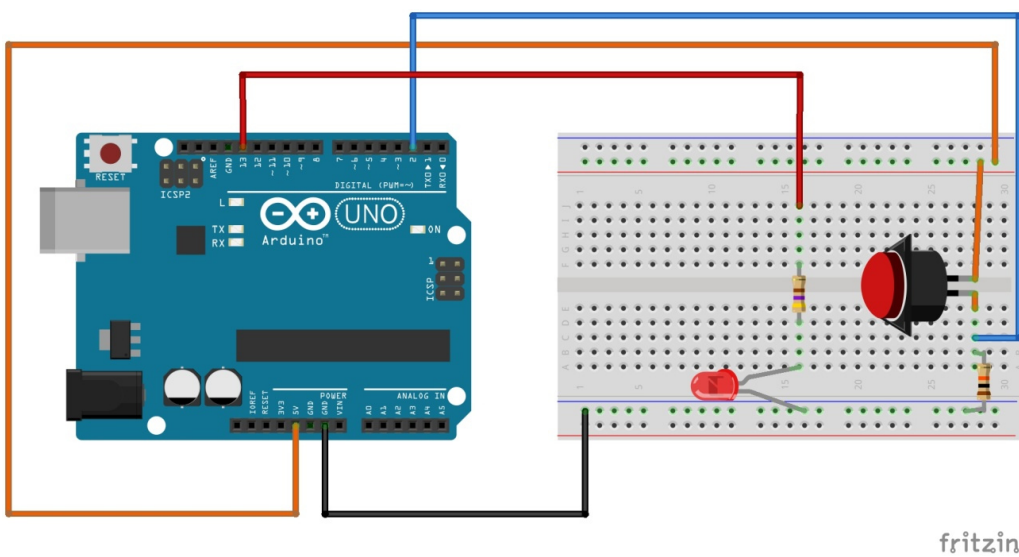
La práctica consiste en programar un temporizador con retardo a la desconexión. Recordemos cómo funciona un temporizador a la desconexión. Al pulsar el pulsador, de forma inmediata se activa la salida (led encendido). Al dejar de pulsar el pulsador,

comienza la temporización (según el tiempo programado 3 segundos), manteniéndose activa la salida, hasta que transcurre el tiempo de temporización, momento en el cual se desactiva la salida (led apagado). Si antes de que concluya el tiempo de temporización se vuelve a pulsar el pulsador, al soltarlo, se inicia una nueva temporización fijada al tiempo programado (3 segundos).

El esquema eléctrico del montaje es el siguiente:



El esquema eléctrico del montaje:

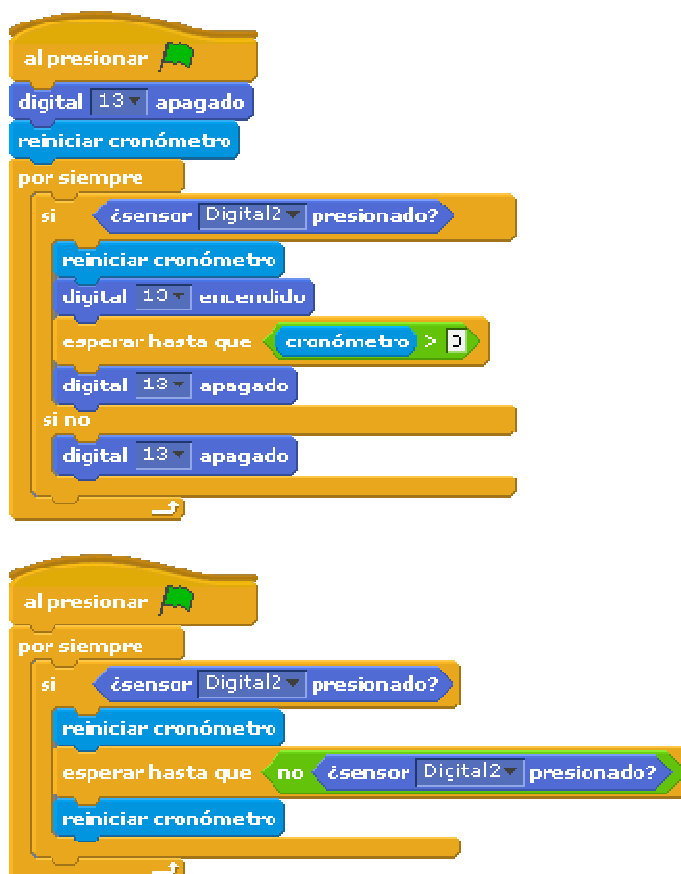


En la programación en S4A, se utilizará el bloque perteneciente a la librería de control “esperar hasta que \_\_” para mantener detenido el programa hasta que ocurra algo. Irá acompañado del bloque “¿sensor digital \_\_ presionado?” ya visto en la práctica anterior. Aunque en este caso hay que hacer notar que nos interesa saber cuándo se ha soltado el pulsador, para iniciar la cuenta del tiempo de temporización. Para ello utilizaremos también el bloque “no” de la librería operadores.





Hay que hacerles ver a los alumnos que con este programa no funciona bien, ya que una vez soltado el pulsador e iniciada la cuenta, si se vuelve a pulsar el pulsador, la cuenta no se reinicia. La solución podemos verla en la siguiente imagen:



También hay que recordar que S4A interactúa con Arduino enviando el estado de los actuadores y recibiendo el de los sensores cada 75 ms, por lo tanto el ancho de pulso ha de ser mayor que este período. Si el pulso es menor de 75 ms, puede ser que S4A no se entere de que ha habido una pulsación.



## 17. Control del encendido y apagado de un diodo led mediante un pulsador con retardo a la conexión.

Los esquemas eléctrico y de montaje o conexionado son los mismos que los de la práctica anterior.

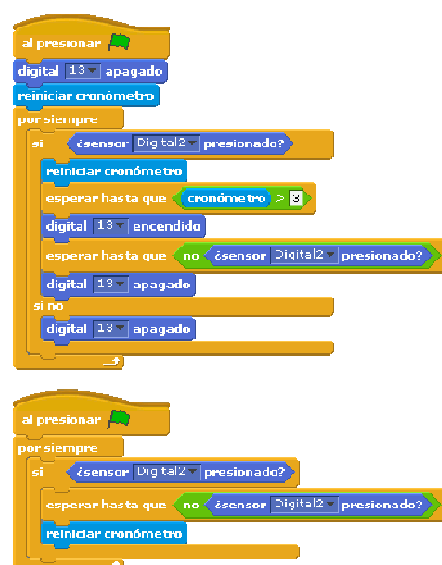
Esta práctica consiste en programar un temporizador con retardo a la conexión. Recordemos su funcionamiento. Al pulsar el pulsador, comienza de forma inmediata la temporización (según el tiempo programado 3 segundos) pero la salida se mantiene desactivada (led apagado). Pasado el tiempo de temporización, se activa la salida (led encendido). Al dejar de pulsar el pulsador la salida se desactiva (led apagado). Si dejamos de pulsar el pulsador antes de que trascorra el tiempo de temporización la cuenta se interrumpe y se anula.

La programación en S4A:



Hay que hacer ver a los alumnos que tiene un fallo. Y es que no cumple uno de los requisitos de un temporizador con retardo a la conexión. Ese requisito es “si dejamos de pulsar el pulsador antes de que trascorra el tiempo de temporización la cuenta se interrumpe y se anula”. Es decir, pulsamos y antes de que trascorra la cuenta soltamos el pulsador y lo volvemos a pulsar antes del tiempo de cuenta; veremos que la cuenta no se ha interrumpido ni anulado. Deben de entenderlo e intentar solucionarlo.

La programación correcta en S4A:



### 18. Control del encendido y apagado de un diodo led mediante un pulsador con retardo a la conexión y a la desconexión.

Los esquemas eléctrico y de montaje o conexionado son los mismos que los de la práctica 16.

La programación en S4A:



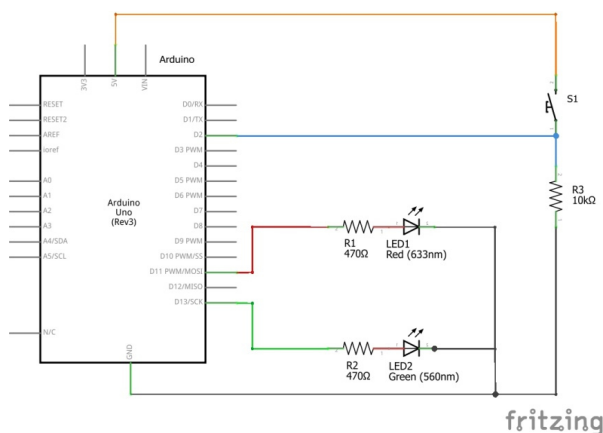
Recordemos que en la parte de temporización a la conexión seguimos teniendo el mismo problema que en la práctica anterior.

La programación correcta en S4A:

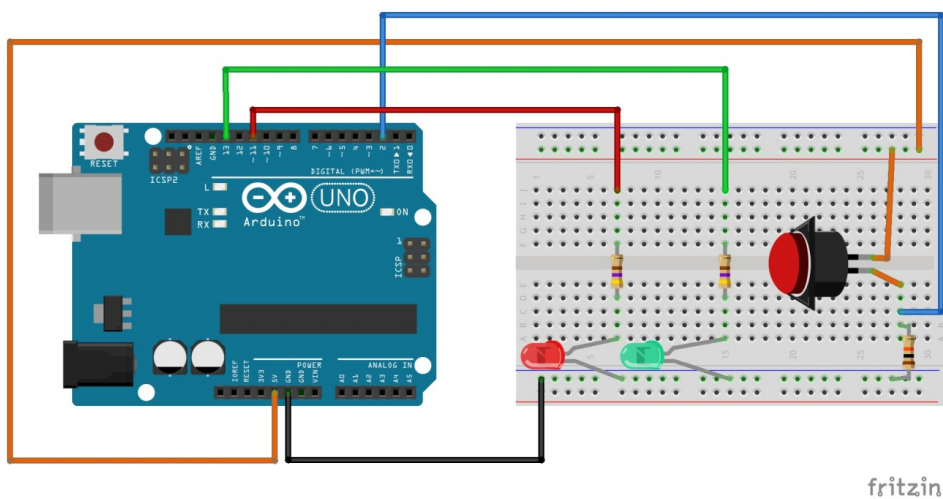


### 19. Control de un semáforo de peatones con pulsador.

En primer lugar lo diseñaremos utilizando salidas digitales. El esquema eléctrico es el siguiente:



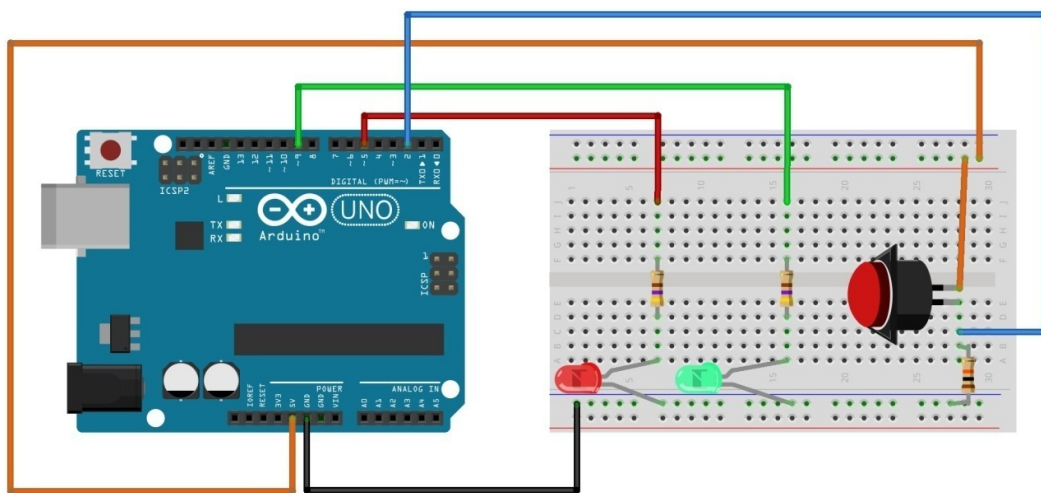
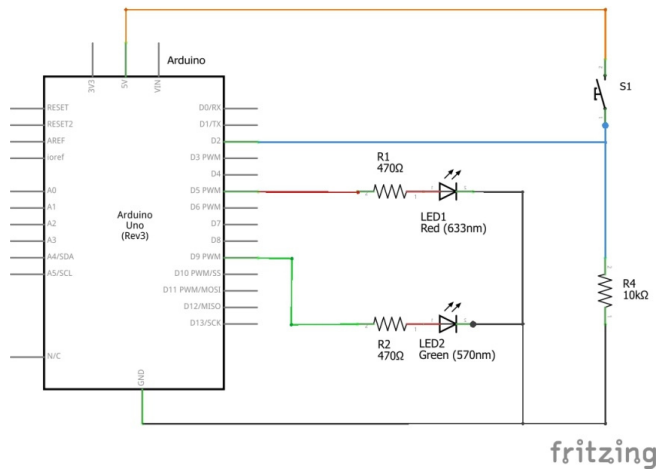
El montaje se realizará como podemos ver en la siguiente figura:



A continuación se realizará la programación en S4A, del funcionamiento de la práctica, basada en el graficet explicativo del funcionamiento y que podemos ver a continuación.



Con salidas analógicas:



```

al presionar [botón]
analógico [5] valor 0
analógico [9] valor 1
por siempre
analógico [5] valor 255
esperar hasta que [¿sensor Digital2 presionado?]
esperar 2 segundos
analógico [5] valor 0
analógico [9] valor 255
esperar 5 segundos
repetir 5
analógico [5] valor 0
esperar 0.5 segundos
analógico [5] valor 255
esperar 0.5 segundos
analógico [9] valor 0
    
```

Se puede completar con la monitorización del funcionamiento del semáforo. Es aconsejable el ir alternando en las prácticas el trabajo con salidas digitales, con salidas analógicas y la monitorización, con el fin de que las prácticas no sean muy repetitivas.

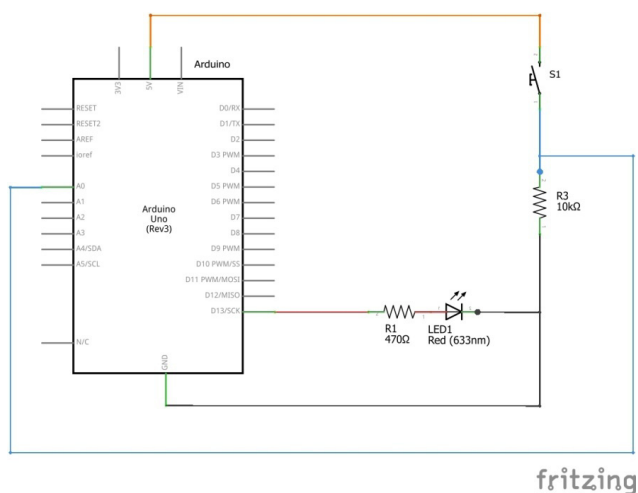
En este documento se plantean todas las vertientes posibles, pero luego es el correspondiente profesor el que decide que partes hacer y cuáles no.

## 20. Control del encendido y apagado de un diodo led mediante un pulsador usando una entrada analógica.

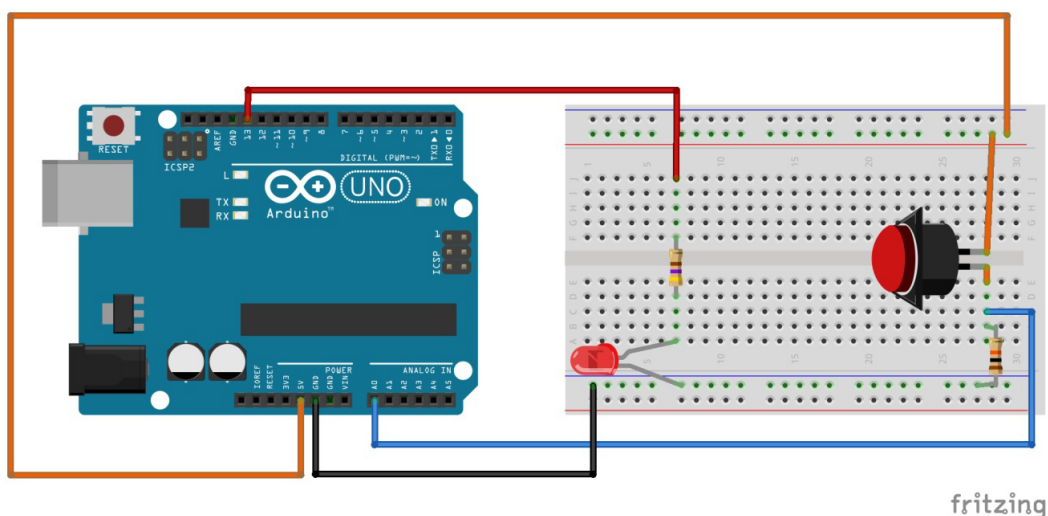
Como nos ha ocurrido con las salidas digitales, en ocasiones no son suficientes y tenemos que utilizar las salidas analógicas como salidas digitales. Lo mismo ocurre con las entradas digitales de las que solo disponemos 2. Por ello vamos a aprender a utilizar las 6 entradas analógicas como si fueran digitales.

Las entradas analógicas de la placa Arduino Uno trabajan con un convertor analógico-digital de 10 bits, oscilando sus valores entre 0 y 1023 ( $2^{10} = 1024$  valores diferentes), correspondiendo un valor de entrada de 0V con conversión 0 y un valor de entrada de 5V con conversión 1023.

El esquema eléctrico es el siguiente:



El conexionado:



A continuación se realizará la programación en S4A. Para ello, como bloques no vistos en las prácticas anteriores, podríamos utilizar el bloque de la librería de operadores “ $\_ = \_$ ”, pero podríamos tener el problema de una pequeña pérdida de tensión, por mal contacto o por caída de tensión por la longitud de los cables y en este caso el valor de la entrada analógica utilizada no llegaría a 1023, por lo que nunca se cumpliría la condición y no funcionaría. Para evitar esto, se utilizará el bloque perteneciente a la librería de operadores “ $\_ > \_$ ” (mayor que) para poder comparar un valor con otro. En este caso irá acompañado del bloque “valor del sensor  $\_$ ” de la biblioteca movimiento.



Como ya hemos visto en otras prácticas, el bloque “ $\_ > \_$ ” lo podemos arrastrar y colocar dentro del bloque “si  $\_$  si no”, al igual que el bloque “valor del sensor  $\_$ ”, del cual pinchando sobre la flecha podemos elegir la entrada con la que queremos trabajar. Y sobre el bloque “ $\_ > \_$ ” podemos introducir valores constantes mediante teclado.



Si observamos la comparación la hacemos con el valor constante 1000, no con el máximo que puede alcanzar la entrada (1023). Esto es debido a que puede haber fallos en conexiones, caídas de tensión en cables, ... y lo que es seguro es que si desde el pulsador nos llega un valor superior a 1000 es porque el pulsador está pulsado y su contacto cerrado.

Para completar la práctica se puede probar con el resto de entradas analógicas.

### 21. Control del encendido y apagado de un diodo led mediante un pulsador usando una entrada analógica asignada a una variable.

La práctica es la misma que la anterior (con los mismos esquemas) con la diferencia de que en vez de trabajar directamente con el valor de la entrada analógica, ésta la asignamos a una variable y trabajamos con la variable. Esto es así porque a la hora de prácticas más complicadas, el trabajar con variables nos da una mayor amplitud de posibilidades.

El trabajo con variables se realiza desde la biblioteca variables que podemos ver en la imagen siguiente. Desde esa biblioteca podemos crear nuevas variables con el nombre que nos interese, borrar variables ya creadas, visualizar la variable en el escenario y fijar la variable a un valor constante o a una entrada, entre otras cosas.



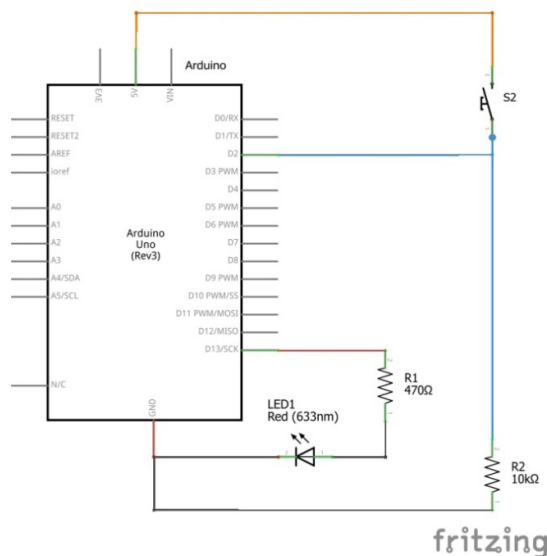
Como bloque nuevo en esta práctica, desde la biblioteca de variables utilizamos “fijar variable \_\_ a \_\_”. Pinchando sobre la flecha podemos elegir la variable con la que queremos trabajar dentro de las variable que hayamos creado; el valor puede ser un valor constante que podemos introducir a través de teclado o asignarlo a una entrada analógica con el bloque “valor del sensor \_\_” arrastrando y colocando sobre el bloque.

El programa en S4A sería el siguiente:

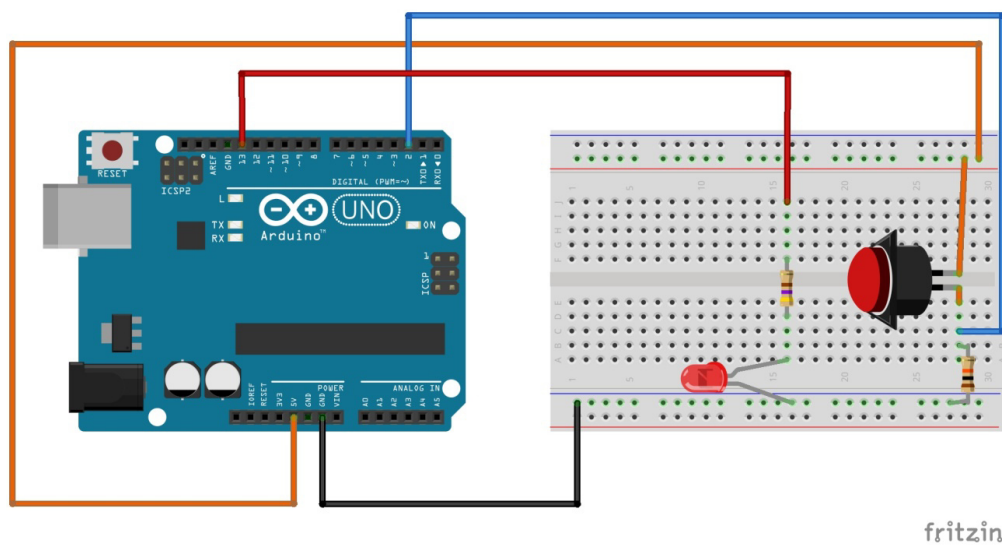


## 22. Control del encendido y apagado de un diodo led mediante un cronómetro o el accionamiento de un pulsador.

El esquema eléctrico es el siguiente:



El esquema de conexionado y montaje:



Como bloque nuevo en esta práctica, desde la biblioteca de operadores utilizamos “`||`” o “`|||`”. Es el bloque que nos permite programar la función lógica O (función OR).



En este caso, se deberá de cumplir **al menos** una de las dos condiciones, es decir, para que el diodo led se apague, habrán tenido que haber pasado 5 segundos **ó** se habrá tenido que pulsar el pulsador. Si se cumplen las dos condiciones también se apagará.



El graficet de explicación del funcionamiento y el programa en S4A son los siguientes:



### 23. Control del encendido y apagado de un diodo led mediante un cronómetro y el accionamiento de un pulsador.

Los esquemas eléctrico y de conexionado son los mismos que los de la práctica anterior.

Como bloque nuevo en esta práctica, desde la biblioteca de operadores utilizamos “\_ y \_”. Es el bloque que nos permite programar la función lógica Y (función AND).



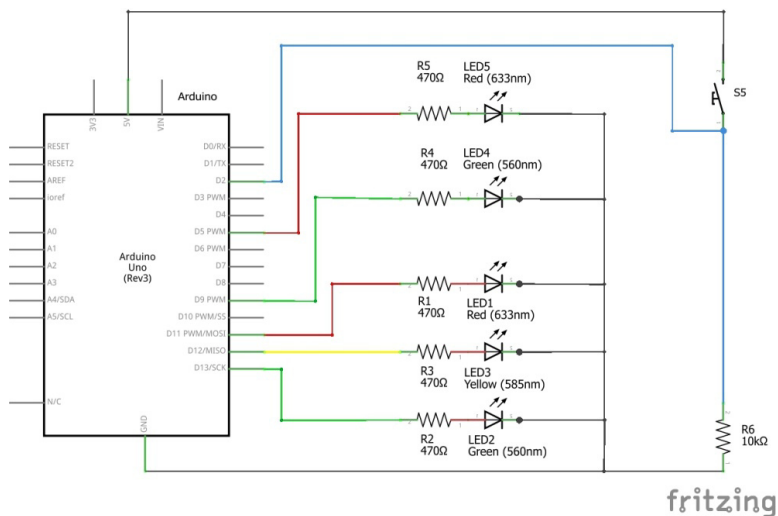
El graficet de explicación del funcionamiento y el programa en S4A son los siguientes:



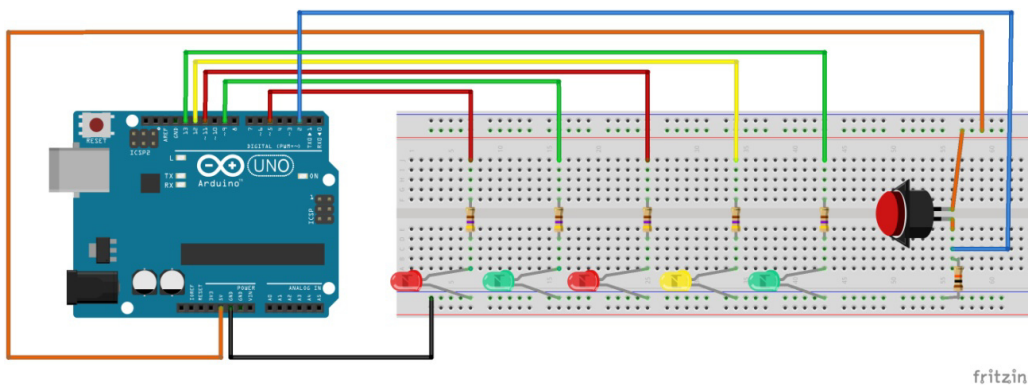
En este caso y a diferencia de la práctica anterior, se deberán de cumplir las dos condiciones **simultáneamente**, es decir, para que el diodo led se apague, habrán tenido que haber pasado 5 segundos **y** se habrá tenido que pulsar el pulsador.

### 24. Control de un semáforo de vehículos y de un semáforo de peatones con pulsador.

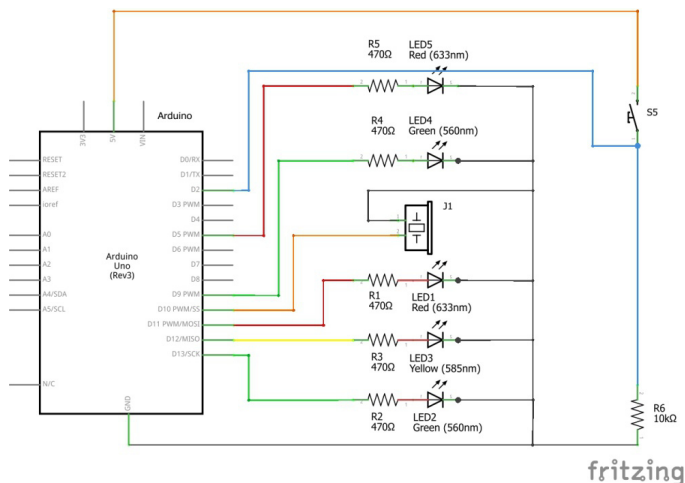
El esquema eléctrico es el siguiente:

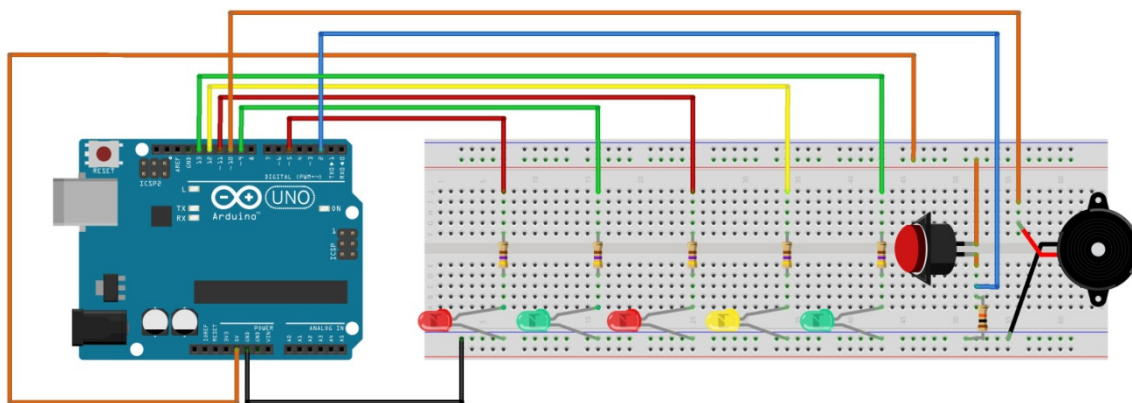


El esquema de conexionado y montaje:



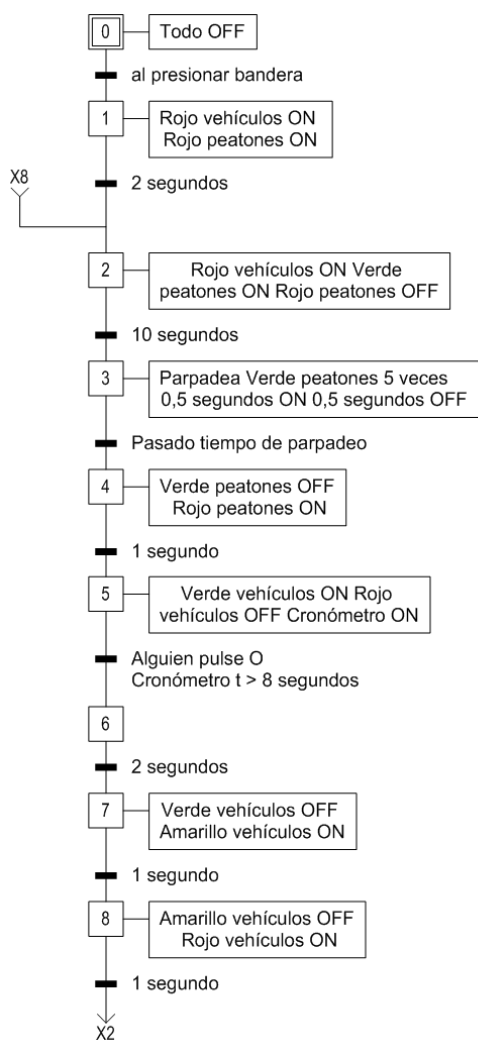
Esta práctica la podremos completar con la conexión de un zumbador piezoeléctrico en la salida digital 10, para utilizarlo como sonido avisador para peatones invidentes y la correspondiente programación de esa salida. También se podrá pedir a los alumnos la monitorización completa de la práctica. En este caso, los esquemas eléctricos y de conexionado y montaje son los siguientes:





fritzing

El graficet de funcionamiento y la programación en S4A para el montaje sin zumbador:



```

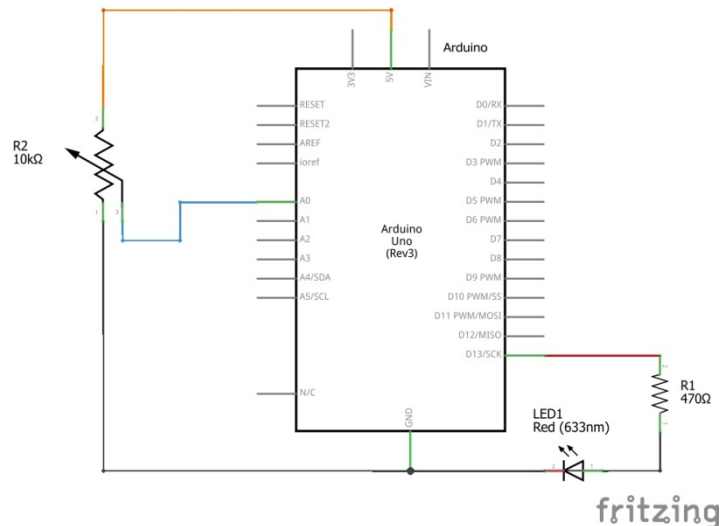
al presionar
digital 11 apagado
digital 12 apagado
digital 13 apagado
analógico 5 valor 0
analógico 9 valor 0
reiniciar cronómetro
esperar 2 segundos
digital 11 encendido
analógico 5 valor 255
esperar 2 segundos
analógico 5 valor 0
por siempre
digital 11 encendido
analógico 9 valor 255
esperar 10 segundos
repetir 5
analógico 9 valor 0
esperar 0,5 segundos
analógico 9 valor 255
esperar 0,5 segundos
analógico 9 valor 0
analógico 5 valor 255
esperar 1 segundos
digital 11 apagado
digital 13 encendido
reiniciar cronómetro
esperar hasta que ¿sensor Digital2 presionado? o cronómetro > 8
esperar 2 segundos
digital 13 apagado
digital 12 encendido
esperar 1 segundos
digital 12 apagado
digital 11 encendido
esperar 1 segundos
analógico 5 valor 0
analógico 9 valor 255
    
```

## 25. Control del parpadeo de un diodo led con tiempo de encendido fijo y tiempo de apagado seleccionado mediante una resistencia variable.

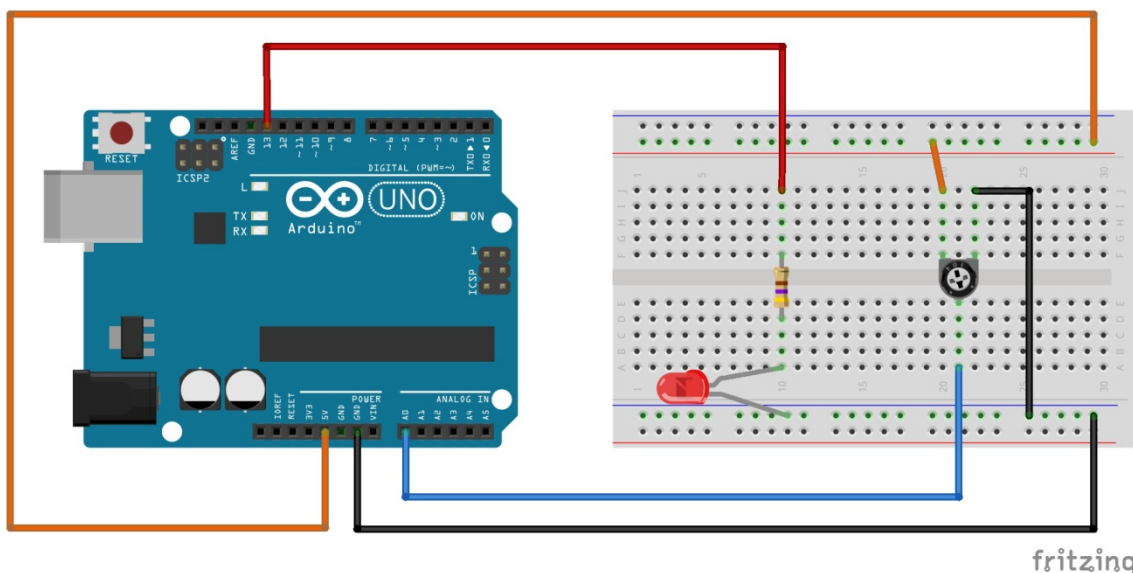
En primer lugar hay que repasar el funcionamiento de una resistencia variable.

Recordemos que cada una de las 6 entradas analógicas (A0, A1, ..., A5) de la placa Arduino Uno trabajan con un convertor analógico-digital de 10 bits, oscilando sus valores entre 0 y 1023 ( $2^{10} = 1024$  valores diferentes), correspondiendo un valor de entrada de 0V con conversión 0 y un valor de entrada de 5V con conversión 1023.

El esquema eléctrico es el siguiente:



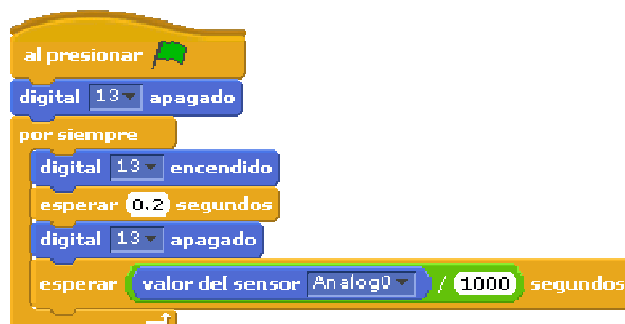
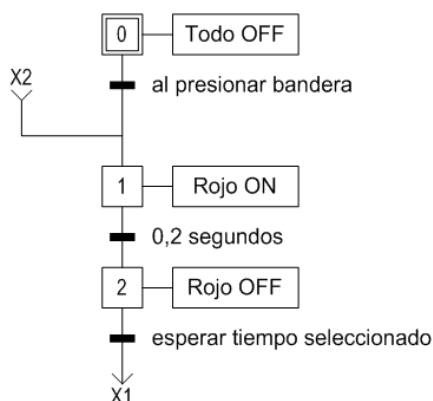
El esquema de conexionado y montaje:



Como bloque nuevo, de la librería de operadores, utilizamos “`__ / __`” (dividir) para hacer el cálculo del tiempo que queremos tener el diodo led apagado. En el bloque podemos hacer cálculos con los valores de entradas analógicas y con constantes introducidas por teclado.



El graficet de funcionamiento y la programación en S4A:



Sería interesante que los alumnos prueben diferentes valores constantes con los que realizar la división (100, 10), para ver como modificamos el tiempo de apagado y modificarlo también desde la resistencia variable.

## 26. Control del parpadeo de un diodo led con tiempo de encendido fijo y tiempo de apagado seleccionado mediante una resistencia variable (entrada analógica asignada a variable).

La práctica es la misma que la anterior con la diferencia de que ahora los cálculos los vamos a realizar por medio de una variable a la que le asignaremos la entrada analógica correspondiente.



Los alumnos tienen que observar que la actualización de la variable nivel, no es constante, es decir, solo se realiza una vez al principio del ciclo. Esto lo podríamos solucionar, quitando al asignación de la entrada analógica a la variable nivel del bucle y creando un nuevo bucle de forma que siempre esté realizando la asignación de la entrada analógica a la variable nivel.

**27. Control del parpadeo de un diodo led con tiempo de encendido fijo y tiempo de apagado seleccionado mediante una resistencia variable (entrada analógica asignada a una variable y otra variable para el cálculo).**

```

al presionar
digital 13 apagado
por siempre
  fijar nivel a valor del sensor Analog0
  fijar tiempo apagado a nivel / 1000
  digital 13 encendido
  esperar 0.2 segundos
  digital 13 apagado
  esperar tiempo apagado segundos
  
```

Hay que hacer notar que el programa lo podríamos haber hecho exactamente igual con solo una variable y cálculo constante del tiempo de apagado.

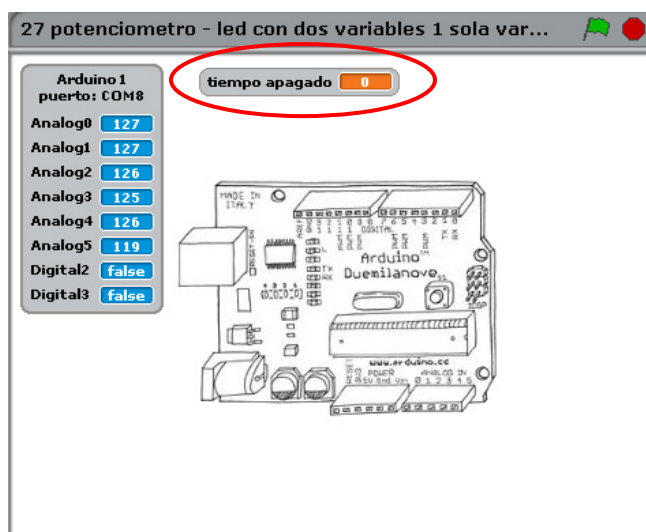
```

al presionar
por siempre
  fijar tiempo apagado a valor del sensor Analog0 / 1000
  
```

```

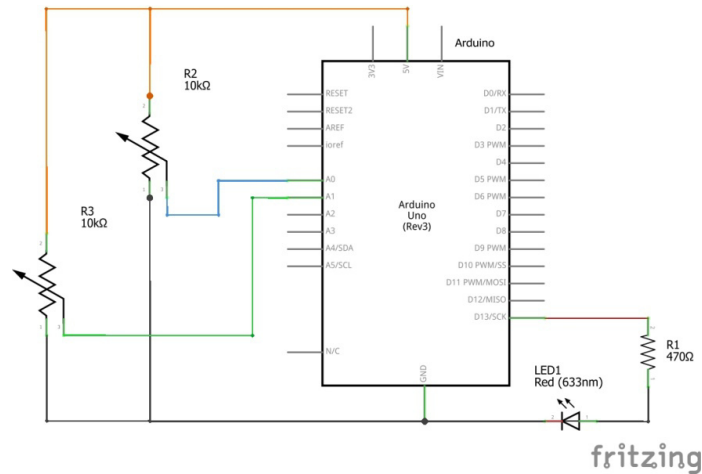
al presionar
digital 13 apagado
por siempre
  digital 13 encendido
  esperar 0.2 segundos
  digital 13 apagado
  esperar tiempo apagado segundos
  
```

La ventaja de la forma de realizar el programa en esta práctica es que tenemos la posibilidad de mostrar en el escenario los cálculos realizados (variable tiempo apagado)

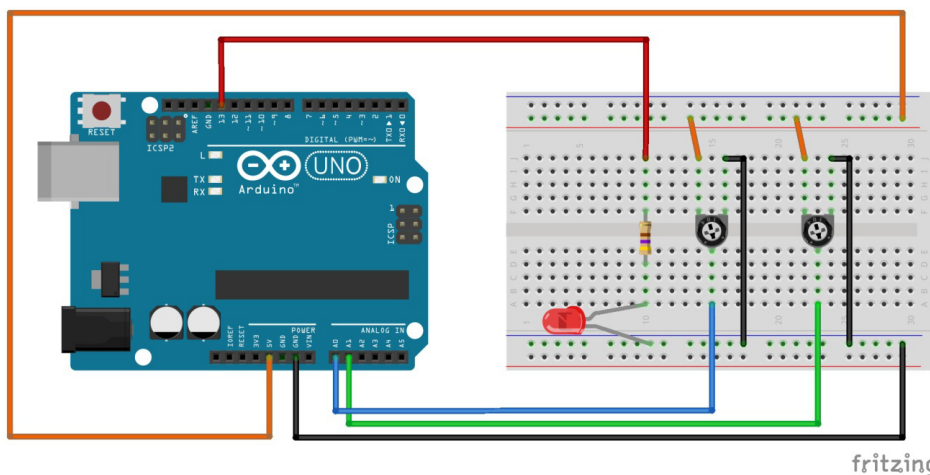


**28. Control del parpadeo de un diodo led con regulación independiente del tiempo de encendido y de apagado seleccionado mediante una resistencia variable asignada a una variable para el cálculo.**

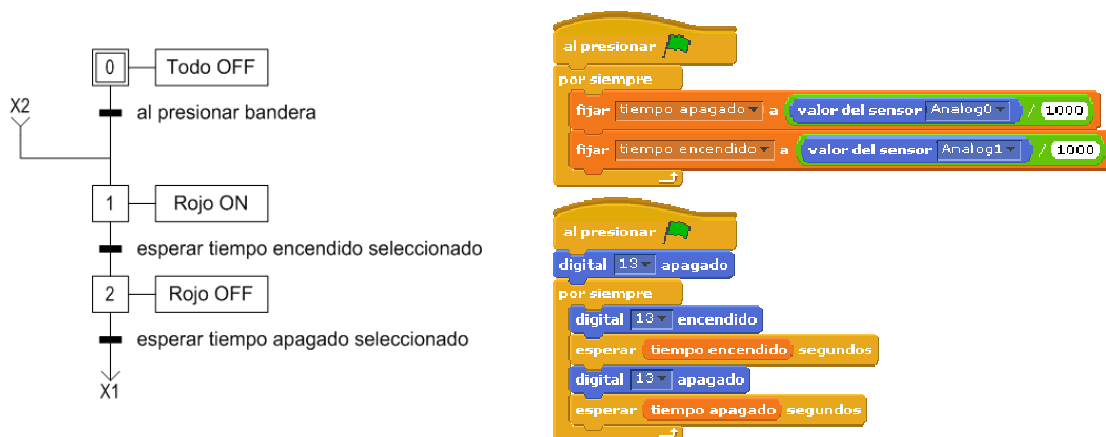
El esquema eléctrico es el siguiente:



El esquema de conexionado y montaje:



El graficet de funcionamiento y la programación en S4A:



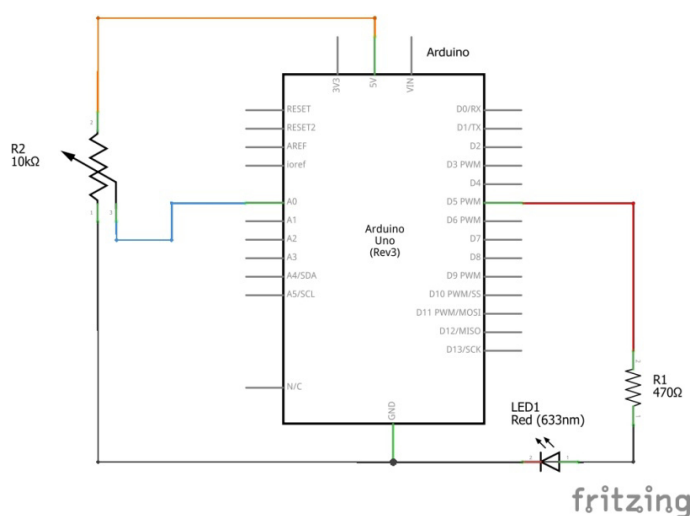
Hemos realizado el programa trabajando con dos variables en las que hacemos los cálculos constantemente, para mostrarlo en pantalla y actuar de forma inmediata.

## 29.Regulación de la intensidad luminosa de un diodo led mediante una resistencia variable.

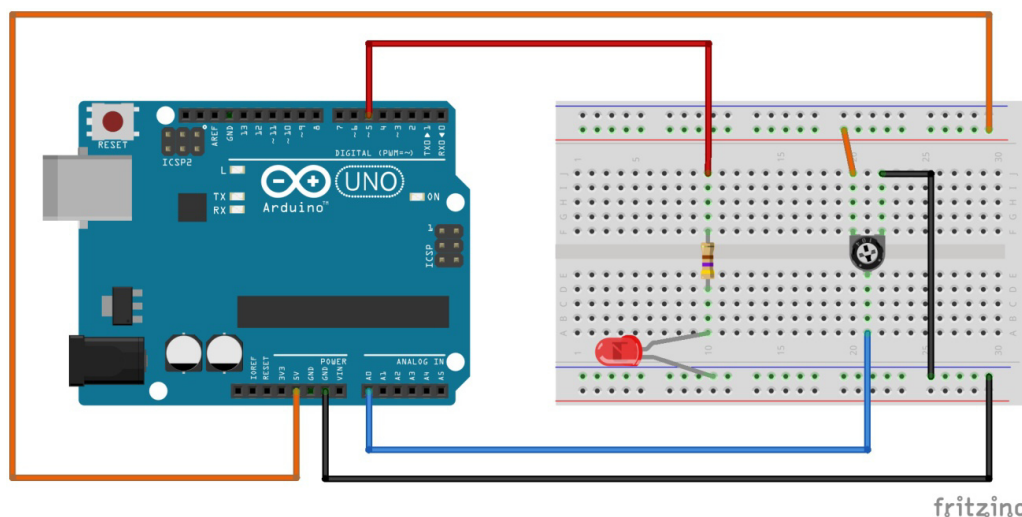
En primer lugar vamos a hacer un breve recordatorio de cómo funcionan tanto las entradas analógica como las salidas analógicas de Arduino. Cada una de las 6 entradas analógicas (A0, A1, ..., A5) de la placa Arduino Uno trabajan con un convertor analógico-digital de 10 bits, oscilando sus valores entre 0 y 1023 ( $2^{10} = 1024$  valores diferentes), correspondiendo un valor de entrada de 0V con conversión 0 y un valor de entrada de 5V con conversión 1023.

Las salidas analógicas (5, 6 y 9) de la placa Arduino Uno trabajan con 8 bits, oscilando sus valores entre 0 y 255 ( $2^8 = 256$  valores diferentes), correspondiéndose la salida 0 con 0V y la salida 255 con 5V.

El esquema eléctrico es el siguiente:

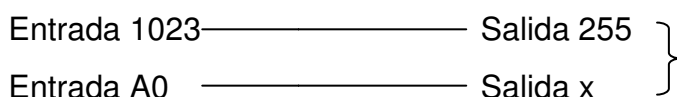


El esquema de conexionado y montaje:



Lo primero que hay que recordar a los alumnos es que podemos trabajar con una escala directamente proporcional, es decir, cuanto mayor sea la entrada analógica A0 mayor será la salida analógica 5. Para ello se debe de resolver la siguiente regla de tres:



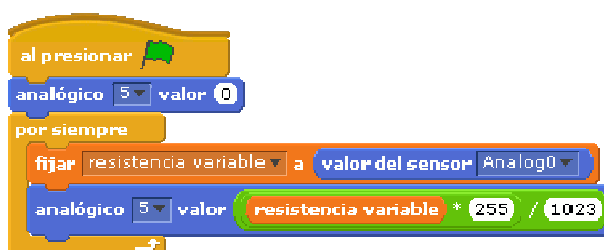


$$\frac{1023}{A0} = \frac{255}{x} \rightarrow x = \frac{A0 \cdot 255}{1023}$$

Vamos a necesitar un nuevo bloque de la biblioteca operadores para poder realizar multiplicaciones, el bloque “\_\_ \* \_\_”:



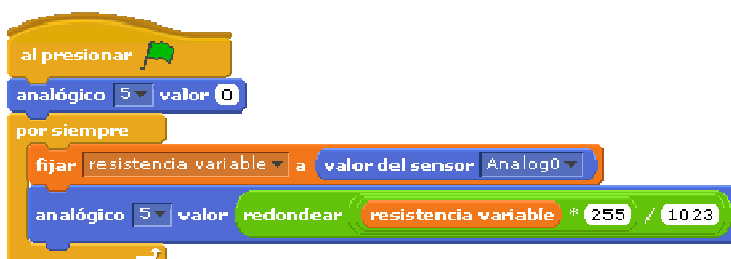
La programación en S4A:



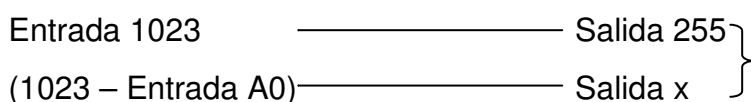
Cuando los alumnos lo prueben observarán que no funciona. Que da un error que indica que las salidas analógicas no pueden trabajar con decimales, solo con números enteros. Por lo tanto, nos va a hacer falta un nuevo bloque de la biblioteca operadores, el bloque “redondear \_\_”, para redondear a un número entero el cálculo realizado y que podamos sacarlo a través de la salida analógica 5.



El programa en S4A correcto es el siguiente:



Con escala inversamente proporcional, es decir, cuanto mayor sea la entrada analógica A0 menor será la salida analógica 5, la regla de tres a plantear es la siguiente:



$$\frac{1023}{1023 - A0} = \frac{255}{x} \rightarrow x = \frac{(1023 - A0) \cdot 255}{1023}$$

El programa en S4A es el siguiente:



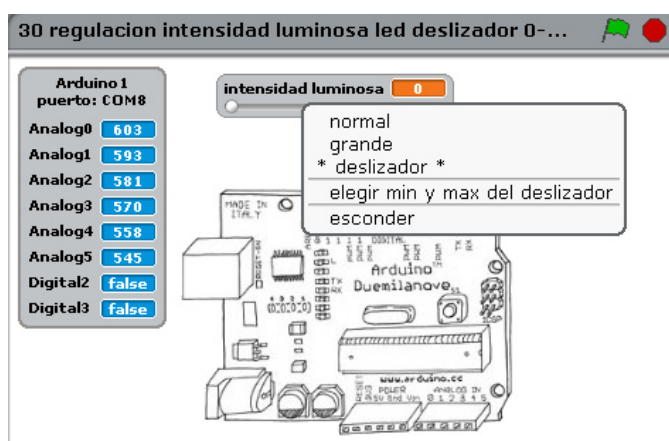
Para realizarlo hemos utilizado un nuevo bloque que utilizamos para invertir la escala. El bloque es restar “\_\_ - \_\_” de la biblioteca de operadores. También vamos a introducir el bloque sumar “\_\_ + \_\_” para utilizarlo cuando sea necesario.



### 30.Regulación de la intensidad luminosa de un diodo led mediante un deslizador en pantalla.

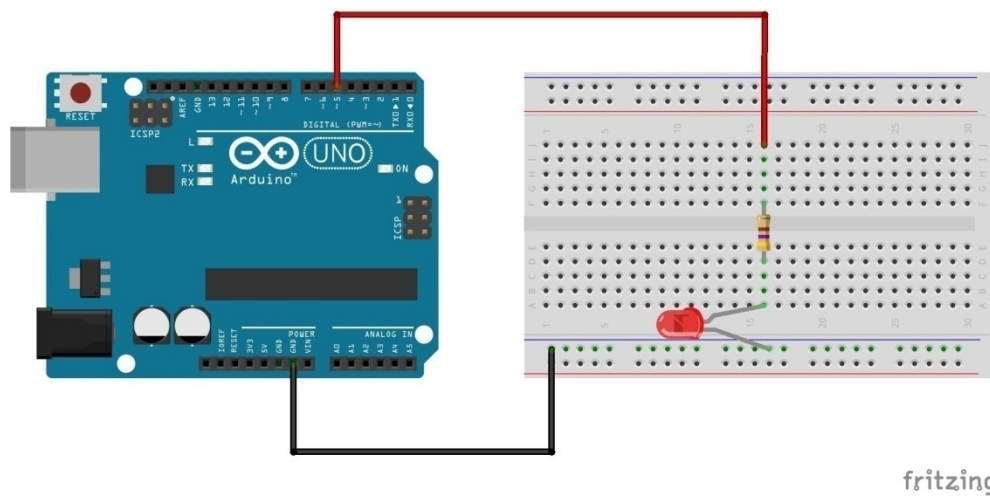
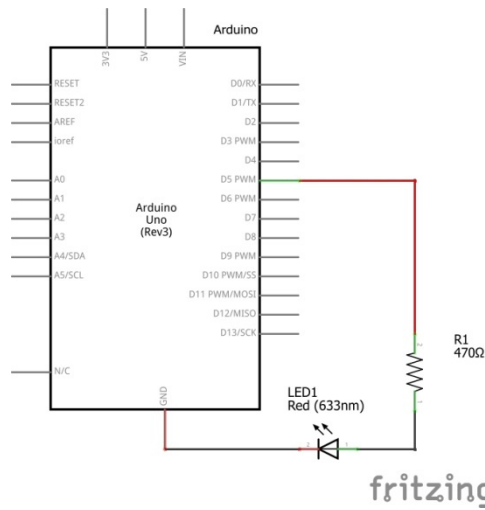
La práctica la podemos realizar introduciendo los valores a través de la pantalla, utilizando “deslizadores” en la variable correspondiente.

Cuando tenemos creada una variable y la estamos visualizando en el escenario (biblioteca variables, tic marcado en la variable), al hacer clic con el botón derecho del ratón sobre la variable visualizada en el escenario, nos aparece el menú que podemos ver en la siguiente imagen, donde podemos cambiar el tamaño de visualización de la variable en el escenario o escoger deslizador, para introducir los valores de esa variable a través de la pantalla. Si elegimos deslizador se nos activa la opción elegir min y max del deslizador, desde donde podemos modificar los valores entre los que va a oscilar la variable en pantalla.



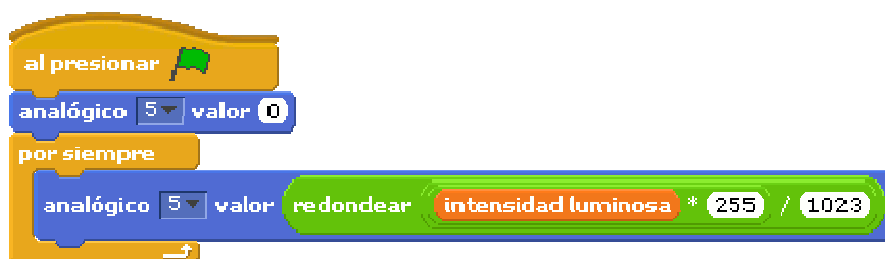
Vamos a plantear tres casos diferentes, de forma que los alumnos tengan que realizar los cálculos correspondientes.

Los esquemas eléctricos y de conexionado son los siguientes:



Sus programas en S4A son los siguientes:

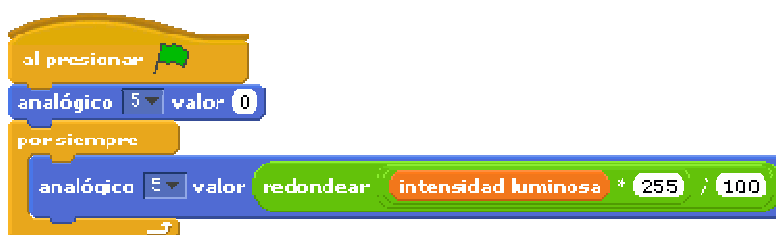
- a) El deslizador de la variable intensidad luminosa oscila entre 0 y 1023, exactamente igual que si fuera una entrada analógica:



- b) El deslizador oscila entre 0 y 255:



c) El deslizador oscila entre 0 y 100 (deslizador porcentual)

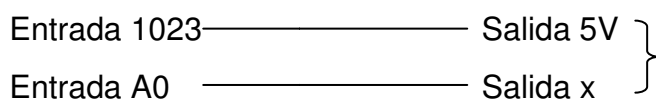


Se pueden plantear nuevos ejercicios jugando con diferentes valores de escala, escalas sin que tengan 0 y sean sus valores extremos positivos, escalas con valores positivos y negativos, escalas con valores negativos e incluso escalas sin que tengan 0 y sus valores extremos sean negativos.

**31.Regulación de la intensidad luminosa de un diodo led mediante una resistencia variable, mostrando el valor analógico de salida y el valor analógico de salida en voltios.**

Los esquemas eléctrico y de conexionado son los mismos que los de la práctica 29. Es más, la práctica es la misma, con la diferencia de que queremos mostrar en pantalla los valores de la salida analógica tanto en valor como en voltaje.

Trabajando con una escala proporcional, en la práctica 29 ya hemos visto como calcular el valor de salida analógica. Para calcular el valor de la salida en voltios planteamos la siguiente regla de tres:



$$\frac{1023}{A0} = \frac{5V}{x} \rightarrow x = \frac{A0 \cdot 5V}{1023}$$

El programa en S4A es el siguiente:



Hay que hacer constar que cuando un valor lo vamos a sacar por una salida analógica sí que debemos redondearlo. Si es para mostrarlo en pantalla no es necesario, ya que si lo redondeamos no veremos la parte decimal.

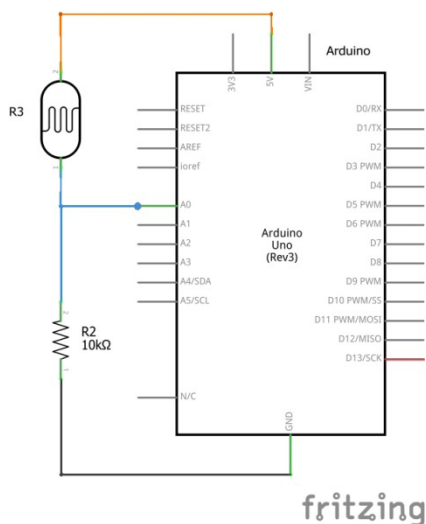
### 32. Control del encendido y apagado de un diodo led en función de la luminosidad ambiental (LDR).

Antes de empezar a trabajar en esta práctica sería muy interesante repasar los contenidos de la práctica 14, concretamente el divisor de tensión y los sistemas de introducir señales Pull-down o lógica directa y Pull-up o lógica inversa.

También se trabajará el funcionamiento de una LDR y se aplicará a los sistemas anteriormente indicados para introducir señales (tecno12-18.com <sup>12</sup> apartado LDR).

Una LDR (light dependent resistor), es una resistencia cuyo valor varía con la intensidad luminosa que incide sobre ella. De forma que si la intensidad luminosa que incide sobre ella es muy alta (día) el valor de su resistencia es muy bajo. Y si la intensidad luminosa es muy baja (noche) el valor de su resistencia es muy alto.

Trabajando con el sistema Pull-down o lógica directa:



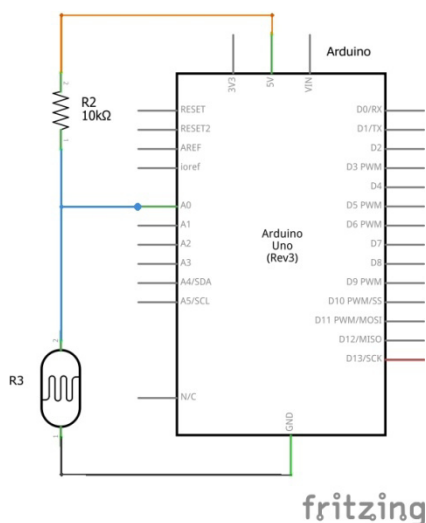
$$U_{A0} = U \cdot \frac{R_2}{R_{LDR} + R_2}$$

Día:  $R_{LDR} \downarrow \rightarrow U_{A0} \uparrow \approx U=5V$

Noche:  $R_{LDR} \uparrow \rightarrow U_{A0} \downarrow \approx 0V$

Con este sistema si observamos, de día mucha luminosidad y mucha tensión en la entrada (lógica directa). Conforme anochece la tensión de la entrada disminuye.

Trabajando con el sistema Pull-up o lógica inversa:



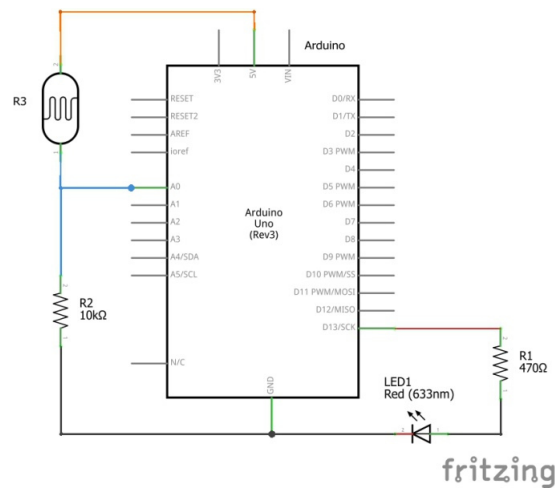
$$U_{A0} = U \cdot \frac{R_{LDR}}{R_{LDR} + R_2}$$

Día:  $R_{LDR} \downarrow \rightarrow U_{A0} \downarrow \approx 0$

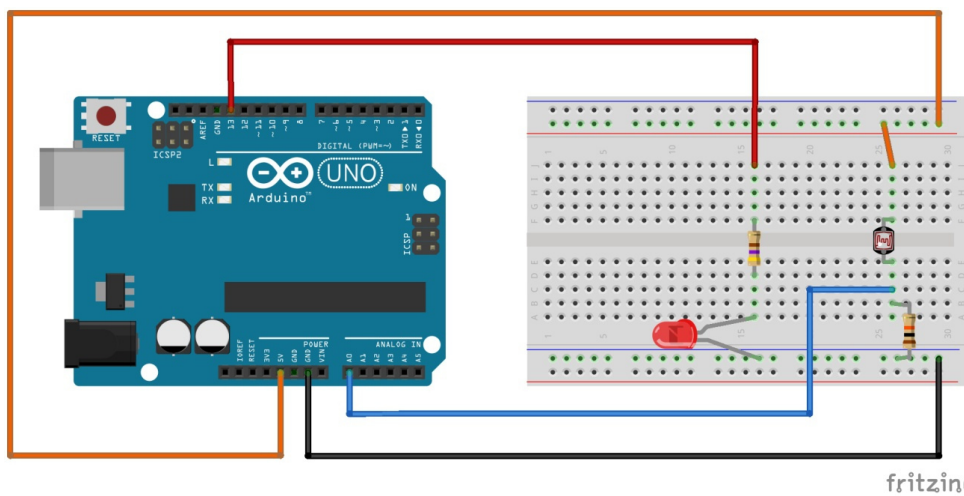
Noche:  $R_{LDR} \uparrow \rightarrow U_{A0} \uparrow \approx U=5V$

Con este sistema de día mucha luminosidad pero poca tensión (lógica inversa). Conforme anochece la tensión de la entrada aumenta.

Una vez entendido el funcionamiento de la LDR y del sistema, el esquema eléctrico de la práctica usando lógica directa es el siguiente:



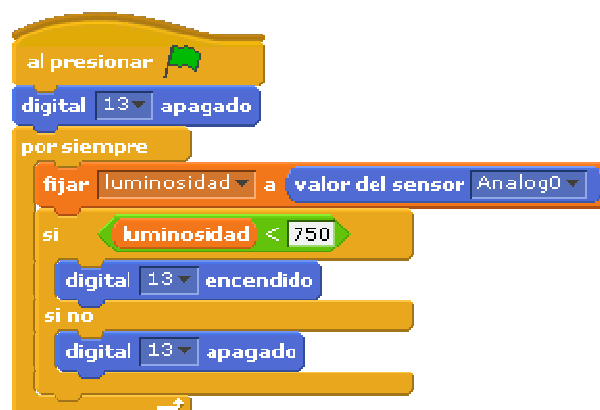
El esquema de conexionado:



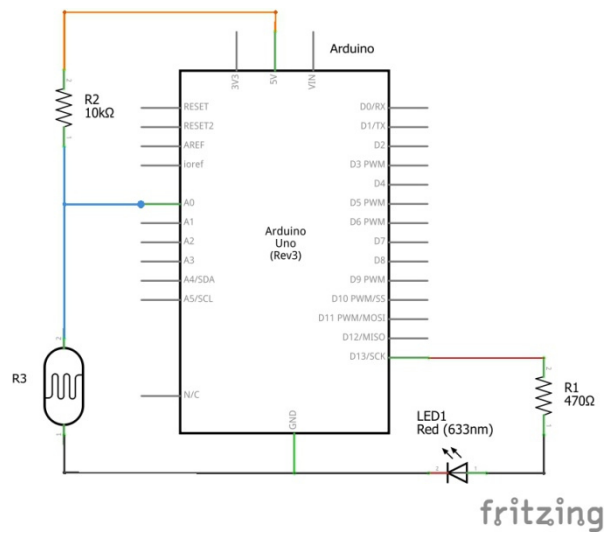
Como bloque nuevo, de la librería de operadores, utilizamos “\_\_ < \_\_” (menor que) para detectar cuando el límite de luminosidad ha descendido por debajo del valor que consideramos para encender el diodo led. En el bloque podemos comparar con otros valores de entradas analógicas y con constantes introducidas por teclado.



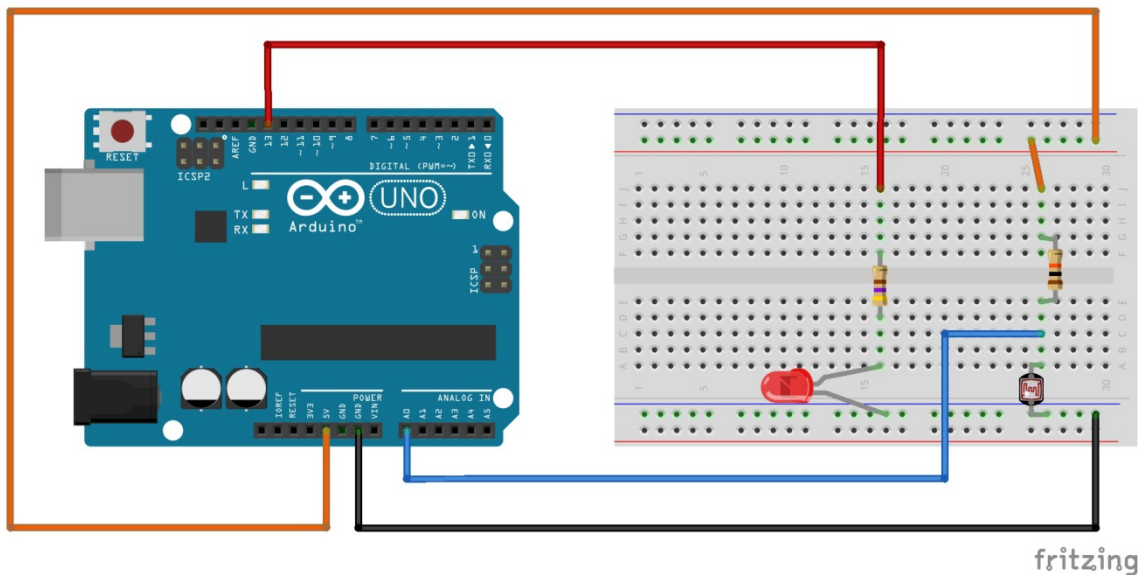
El programa en S4A:



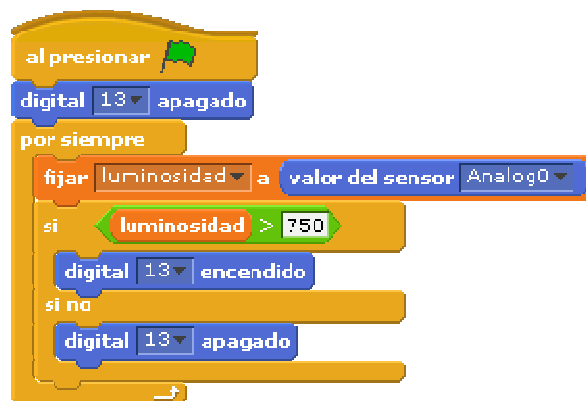
Para el caso de utilizar lógica inversa el esquema eléctrico es el siguiente:



El esquema de conexionado:



La programación en S4A:



### 33. Control del encendido con retardo y apagado de un diodo led en función de la luminosidad ambiental.

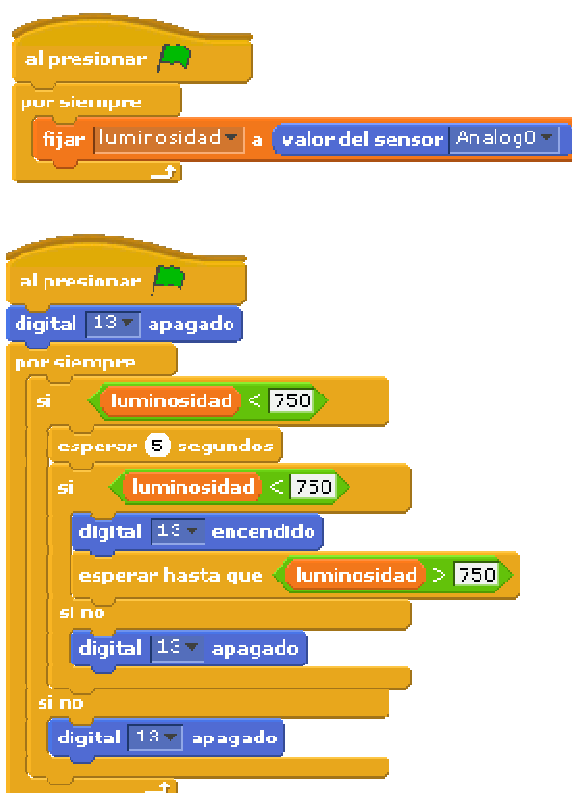
En las siguientes prácticas, a no ser que se indique lo contrario, utilizaremos el sistema de lógica directa.

Los esquemas de montaje y conexionado son los mismos que los de la práctica anterior.

Antes de comenzar con el desarrollo de la práctica, es necesario repasar las prácticas 16, 17 y 18 y recordar que teníamos algunos problemas de funcionamiento en la parte de temporizador a la conexión y a la desconexión. Ahora ya tenemos los conocimientos necesarios para solucionarlo.

En la práctica anterior, el funcionamiento es correcto, pero con el pequeño inconveniente de que el sistema actúa de forma inmediata ante los cambios de luminosidad. En determinadas aplicaciones, como puede ser el encendido de un alumbrado, se requiere un tiempo de retardo y un mantenimiento o disminución de las condiciones de luminosidad para que el sistema actúe.

El programa en S4A es el siguiente:



Al colocar dos bloques de inicio de programa conseguimos actualizar constantemente el valor de la variable luminosidad asignada al sensor analógico 0, evitando la actualización solamente cuando pasamos por un punto determinado del programa.



### 34. Control del encendido y apagado con retardo a la conexión y a la desconexión de un diodo led en función de la luminosidad ambiental.

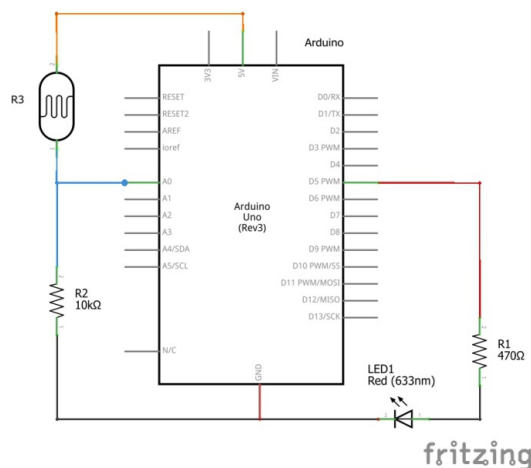
Los esquemas de montaje y conexionado son los mismos que los de la práctica número 32.

La programación en S4A:

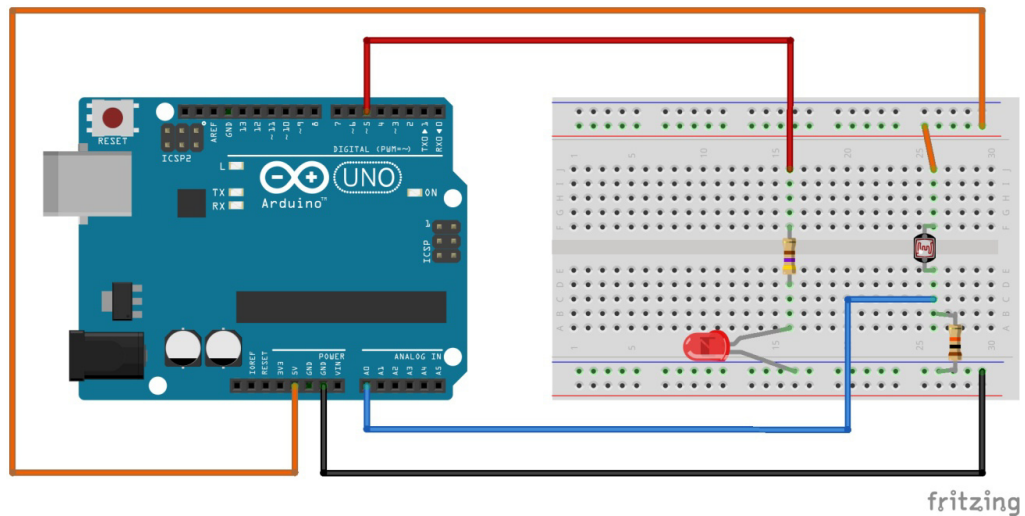


### 35. Regulación y control de la intensidad luminosa de un diodo led y de su encendido y apagado en función de la luminosidad ambiental.

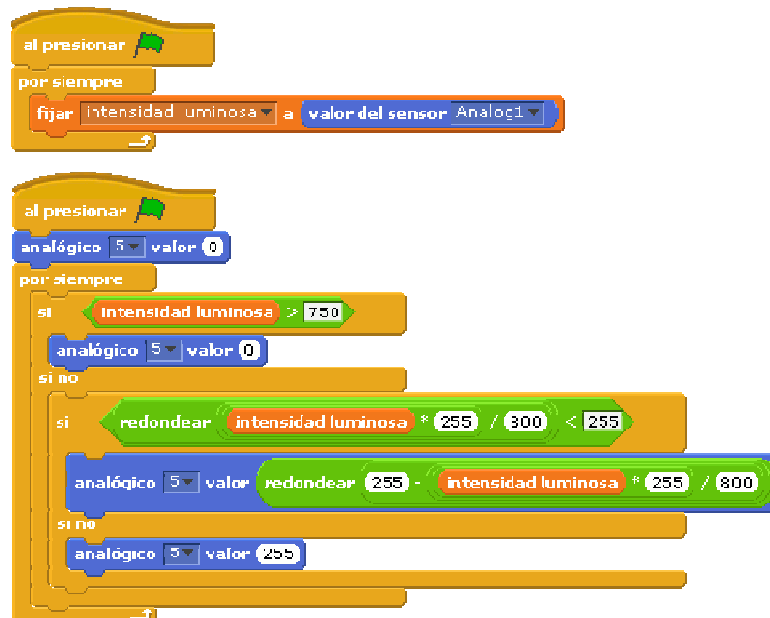
El esquema eléctrico de la práctica es el siguiente:



El esquema de conexionado y montaje:



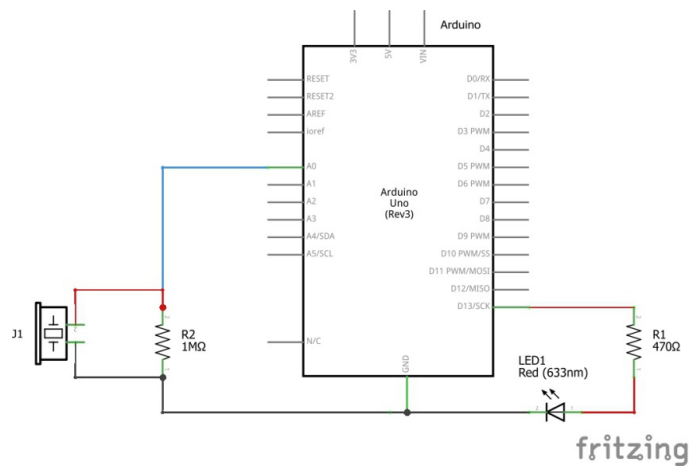
Un ejemplo de programación en S4A:



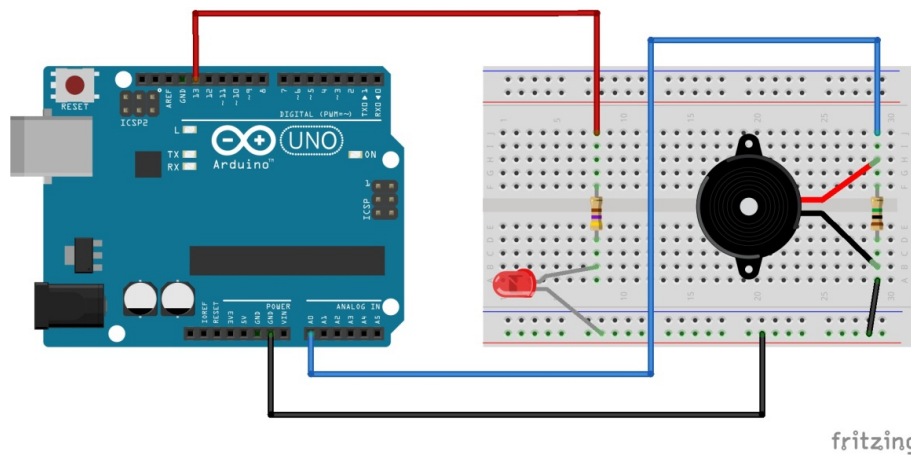
### 36. Detector de vibraciones.

En esta práctica vamos a utilizar como detector de vibraciones un zumbador piezoeléctrico en su funcionamiento inverso, es decir, al alimentarlo con una tensión eléctrica genera una vibración que es la que produce el sonido. Pero si no lo alimentamos y lo hacemos vibrar, esa vibración genera una pequeña tensión eléctrica, que es la que podemos utilizar para detectar esas vibraciones.

El esquema eléctrico es el siguiente:



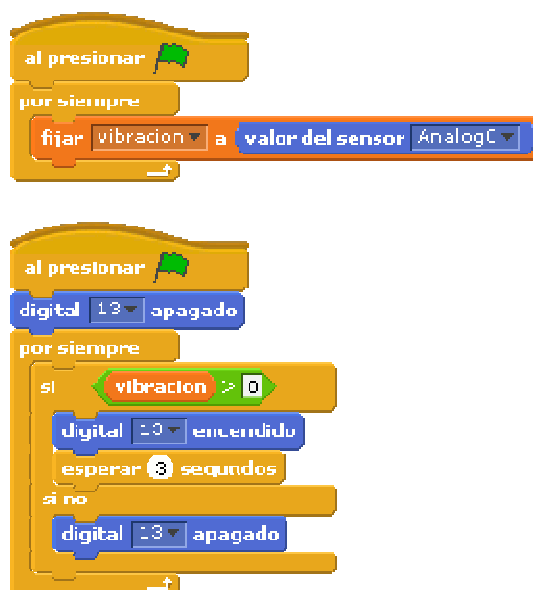
El esquema de conexionado y montaje:



La programación en S4A:



Si observamos detalladamente, con esta programación las lecturas no son continuas, de forma que si ha detectado vibración y ha actuado, la siguiente lectura se realizará a los 3 segundos. Esto podríamos solucionarlo con esta otra programación:

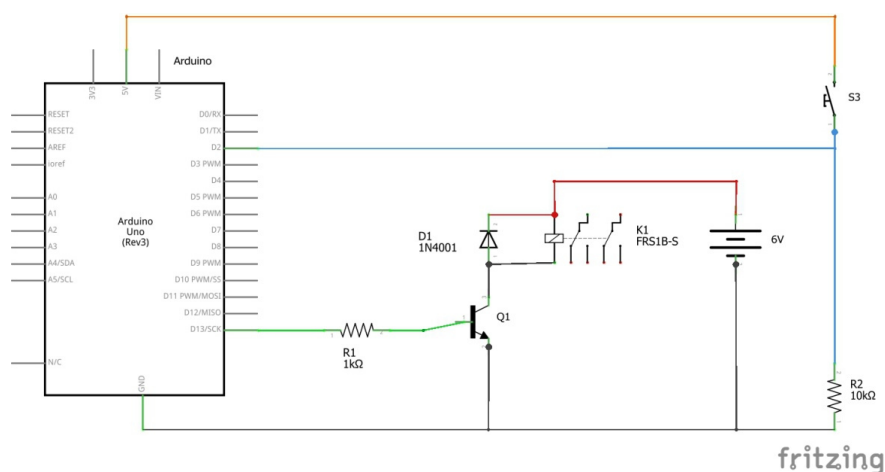


### 37. Control del accionamiento mediante pulsador de un relé, de forma indirecta a través de un transistor NPN BD135 (Ic max = 1,5 A).

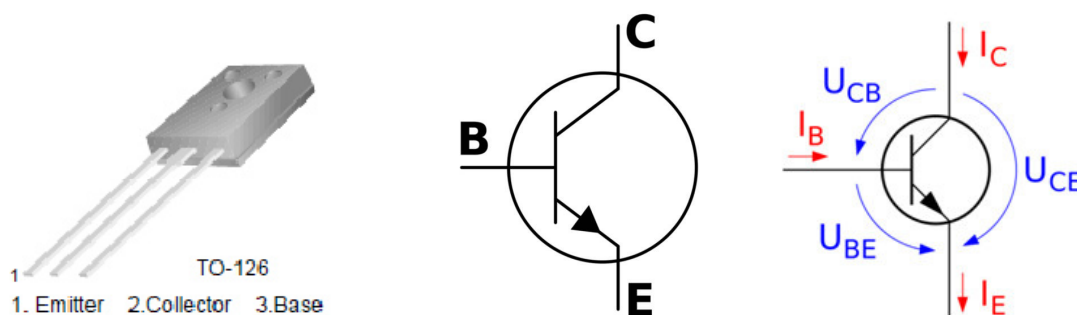
Una de las características de la placa Arduino es que la corriente máxima que puede proporcionar por sus pines utilizados como salidas son 40 mA. Esa corriente es suficiente para alimentar un diodo led o un pequeño zumbador, como hemos estado usando hasta ahora, pero no es suficiente para alimentar el resto de receptores, fundamentalmente lámparas incandescentes, motores y resistencias. Por este motivo, a partir de esta práctica, vamos a ver diferentes formas de construir una etapa de salida para poder alimentar todos estos receptores, siempre contando con una fuente de alimentación externa (pilas) que nos permita amplificar la corriente de salida.

Para ello se recomienda trabajar el capítulo del transistor en el libro digital de tecno12-18.com <sup>12</sup>, situado dentro de la unidad electrónica analógica y repasar el capítulo del relé.

El esquema eléctrico es el siguiente:

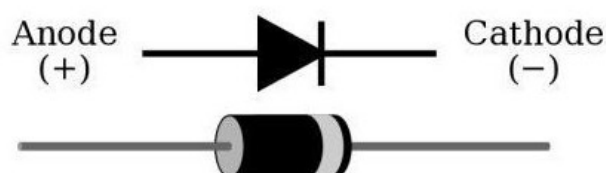


A continuación podemos observar el esquema de patillaje y tensiones e intensidades en el transistor <sup>18</sup> :

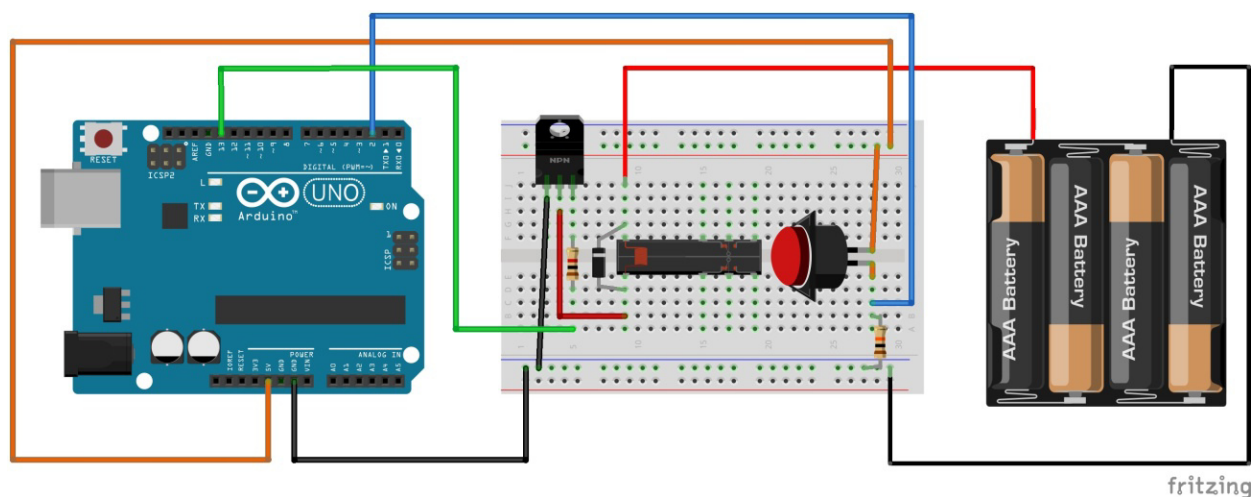


También podemos observar que en paralelo con la bobina del relé hay un diodo rectificador en antiparalelo cuya función es la de evitar las sobretensiones que se originan al interrumpir bruscamente la corriente que circula por la bobina del relé. Estas sobretensiones podrían dañar el transistor e incluso la propia placa Arduino.

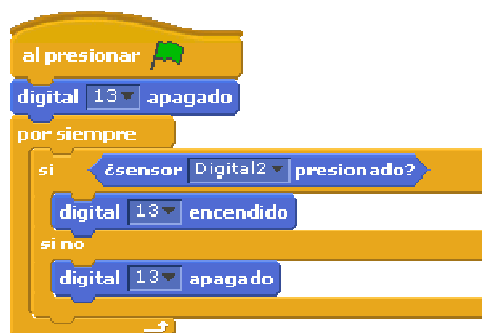
El esquema de patillaje del diodo rectificador es el siguiente <sup>18</sup> :



El esquema de conexionado y montaje:



La programación la vamos a hacer muy sencilla, de forma que cuando pulsemos el pulsador NA, la correspondiente salida de la placa active el relé y cuando dejemos de pulsarlo lo desactive. Con los contactos del relé podríamos accionar lo que nos interese, alimentado con corriente continua o alterna y a la tensión necesaria, respetando siempre las características nominales del relé utilizado (8A 250V AC). Un ejemplo de programación:

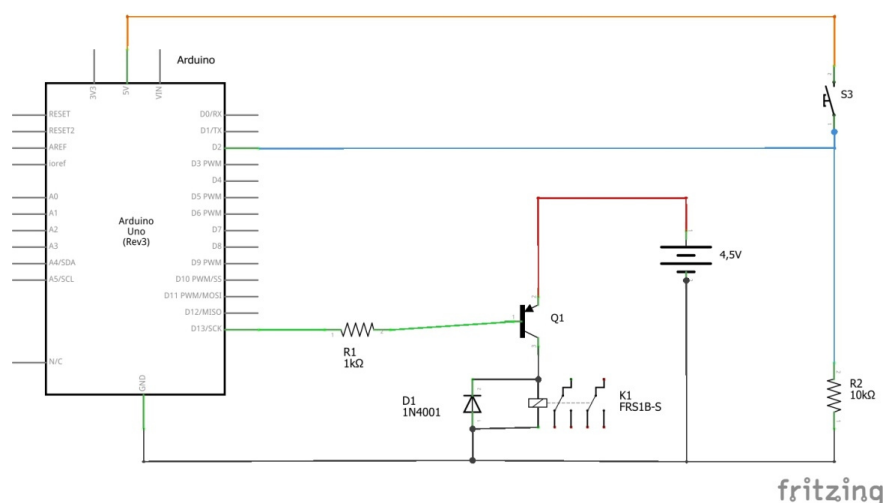


### 38. Control del accionamiento mediante pulsador de un relé, de forma indirecta a través de un transistor PNP BD138 ( $I_c \text{ max} = -1,5 \text{ A}$ ).

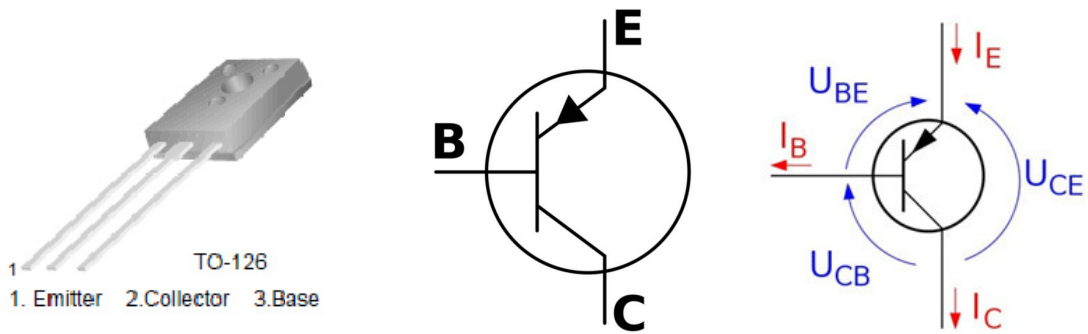
La práctica es similar a la anterior con la diferencia de que usamos un transistor PNP. Hay que fijarse que con un transistor NPN, la carga (bobina del relé) la tenemos conectada al positivo de la fuente de alimentación externa (portapilas) y la corriente que sale por la salida de arduino entra por la base del transistor. Con un transistor PNP la carga la tenemos conectada al negativo de la fuente de alimentación externa (referenciada siempre a masa) y la corriente sale por la base del transistor y entra por la salida de la placa Arduino.

También hay que hacer notar que la fuente de alimentación externa (portapilas) que vamos a usar es de 4,5V a diferencia de la práctica anterior que era de 6V. El motivo es que como la salida de la placa Arduino nos da 5V, con una tensión de 6V el transistor en ocasiones entraba en zona activa cosa que con una fuente de alimentación de 4,5V es imposible.

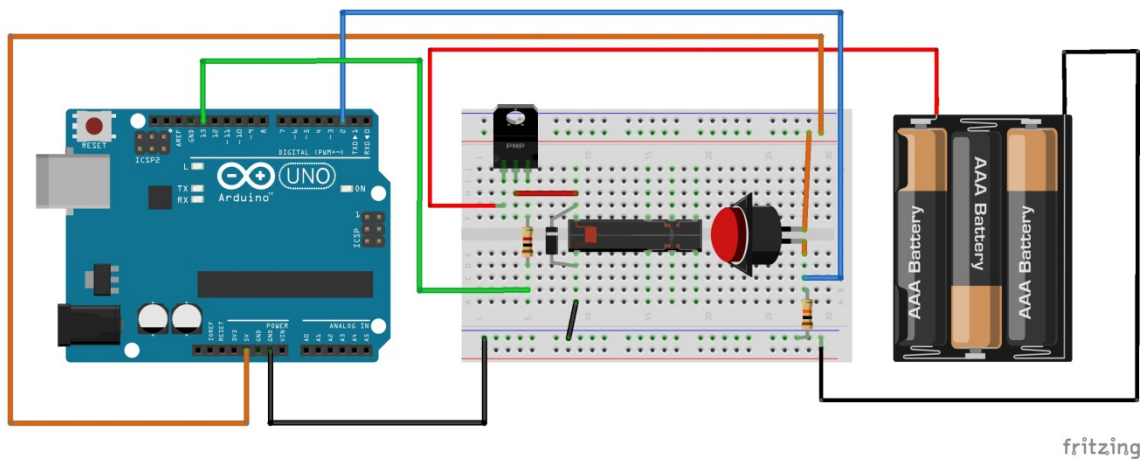
El esquema eléctrico es el siguiente:



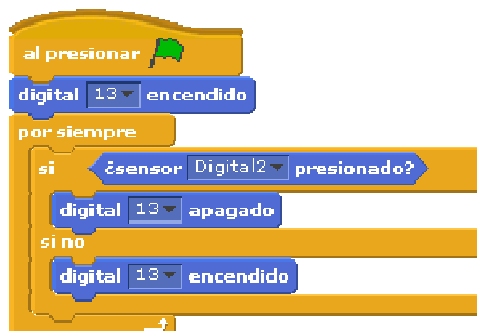
A continuación podemos observar el esquema de patillaje y tensiones e intensidades en el transistor <sup>18</sup>:



El esquema de conexionado y montaje:

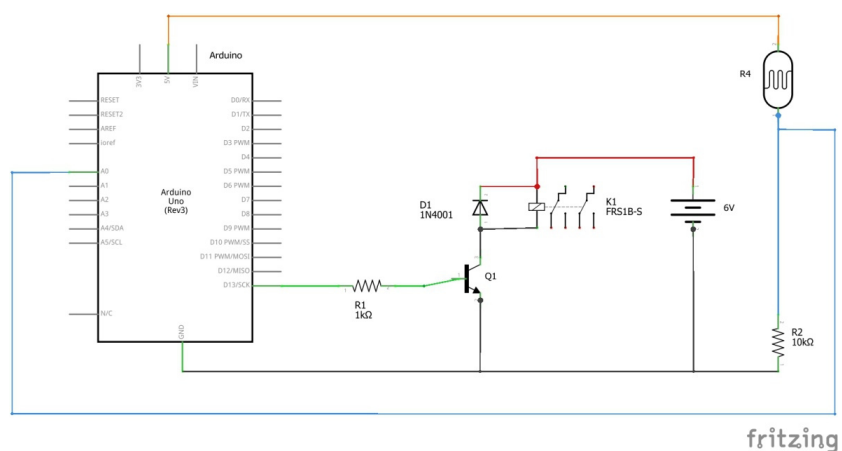


Un ejemplo de programación en S4A:

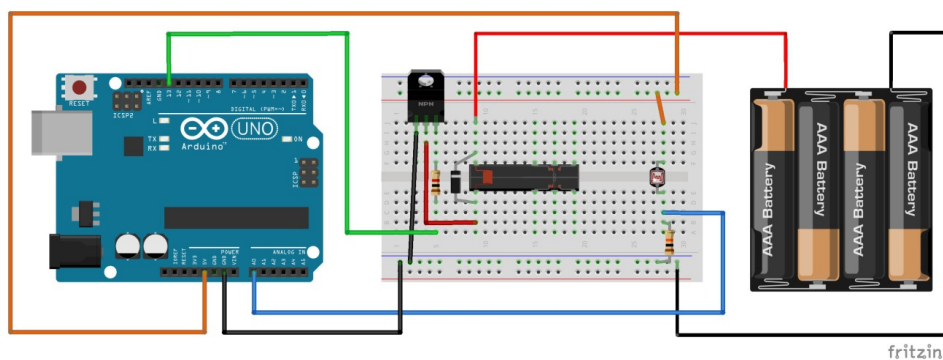


### 39. Control del accionamiento, en función de la luminosidad ambiental, de un relé, de forma indirecta a través de un transistor NPN BD135.

El esquema eléctrico es el siguiente:



El esquema de conexionado y montaje:



Como bloque nuevo, de la librería de control, utilizamos “repetir hasta que \_\_\_” para repetir una acción o serie de acciones de forma continua hasta que ocurra el evento que añadimos al bloque.

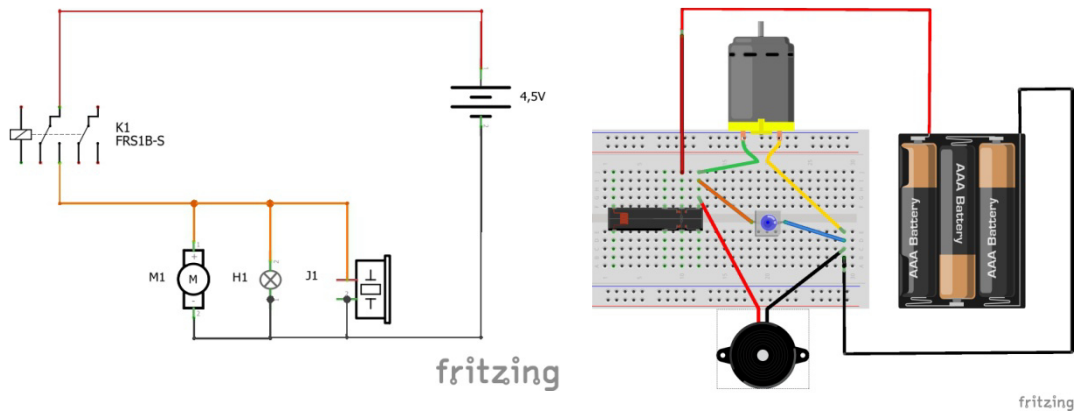


Un ejemplo de programación en S4A:



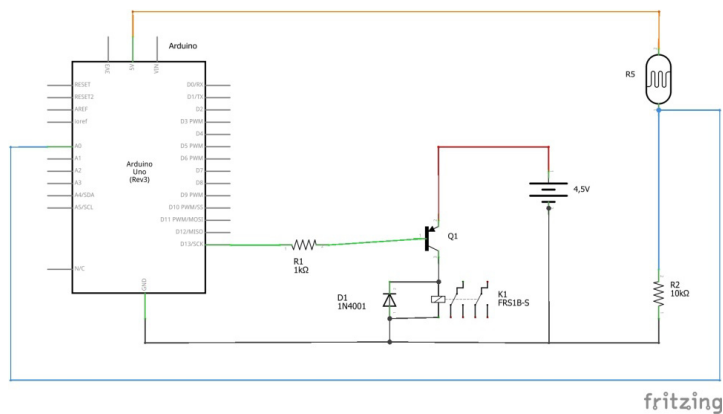


El fin principal de esta práctica y de todas que vienen a continuación hasta la 42, es el accionamiento automático indirecto a través de un relé de un receptor en función de la luminosidad ambiental (encendido automático de alumbrado público, en automóviles, dispositivos que solo queramos que trabajen de noche, ...) y afianzar el montaje y comprensión de los circuitos con transistores. Para no complicar demasiado los circuitos y como ya se ha trabajado con relés, no se representan los circuitos de potencia o circuitos a controlar, que podrían ser los necesarios para controlar el o los receptores que nos interese y de los que se pone un ejemplo a continuación:

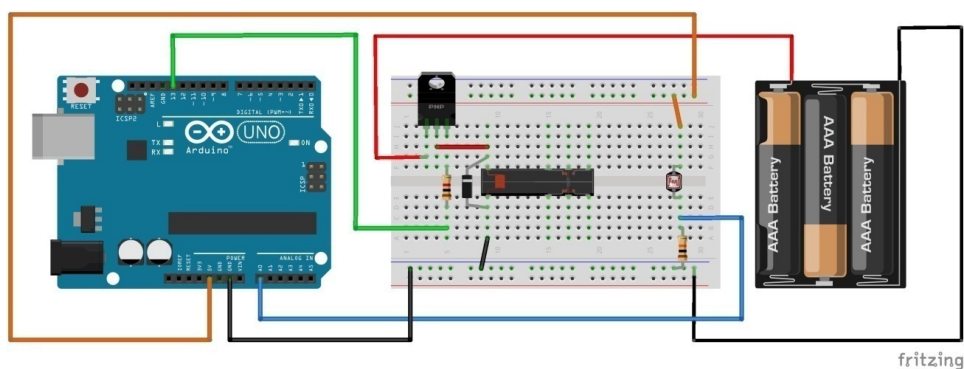


**40. Control del accionamiento, en función de la luminosidad ambiental, de un relé, de forma indirecta a través de un transistor PNP BD138.**

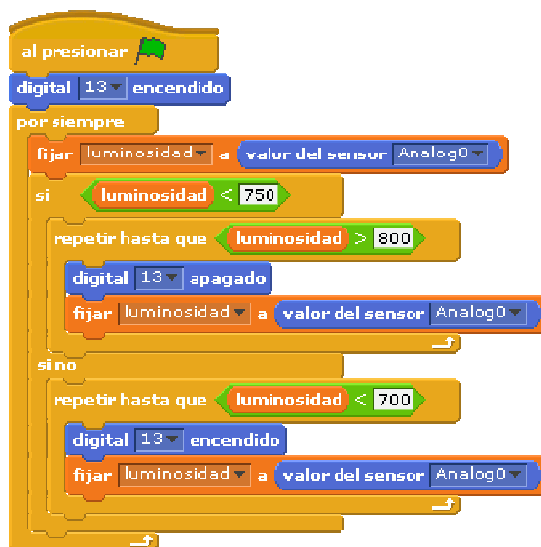
El esquema eléctrico es el siguiente:



El esquema de conexionado y montaje:

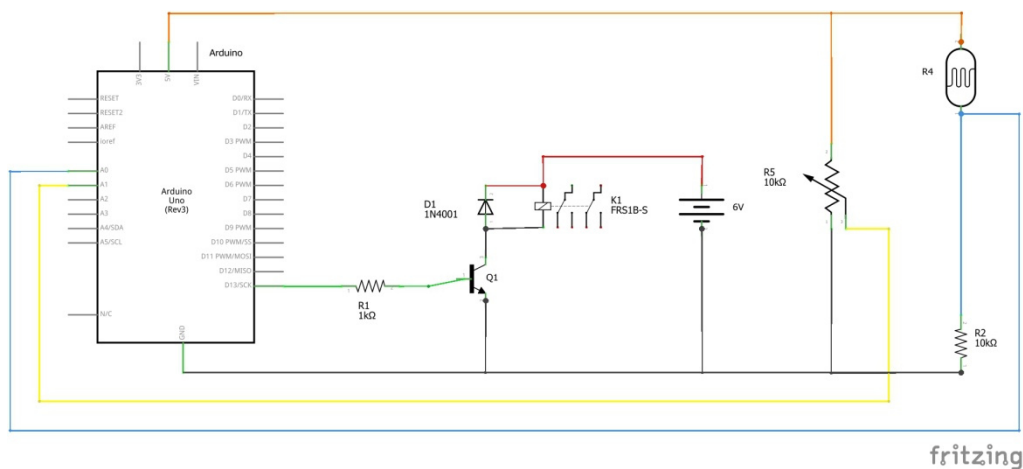


Un ejemplo de programación en S4A:

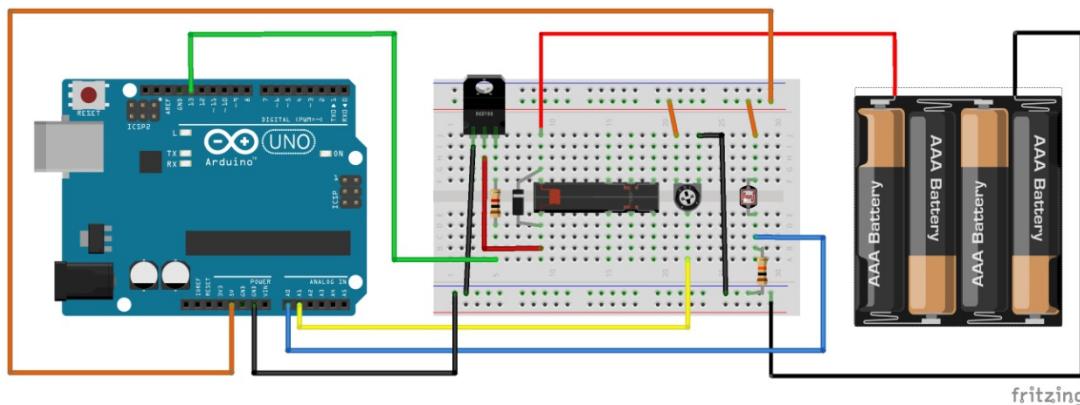


**41. Control del accionamiento, en función de la luminosidad ambiental, de un relé, de forma indirecta a través de un transistor NPN BD135, con posibilidad de elegir y modificar el nivel de referencia de luminosidad.**

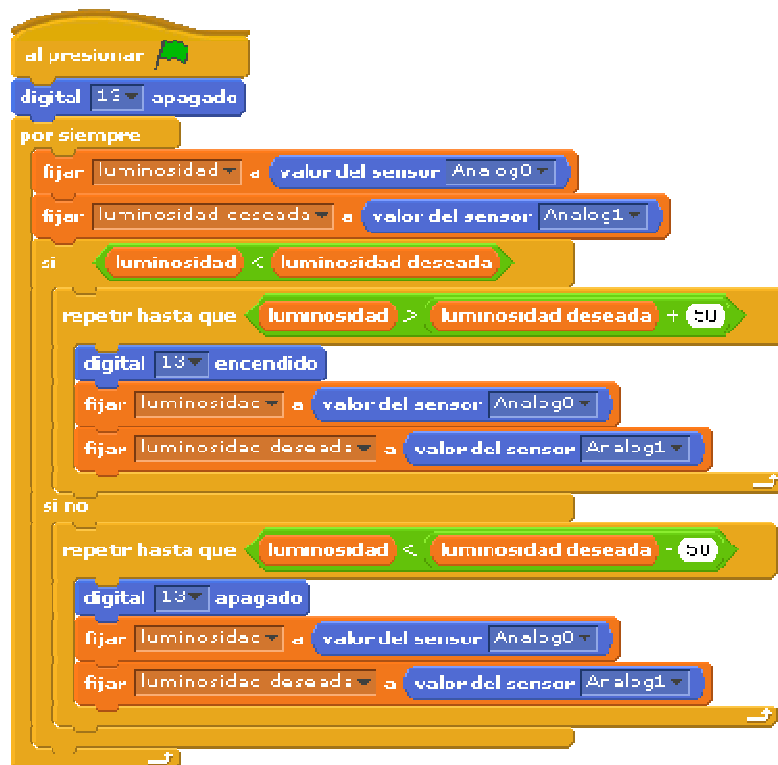
El esquema eléctrico es el siguiente:



El esquema de conexionado y montaje:

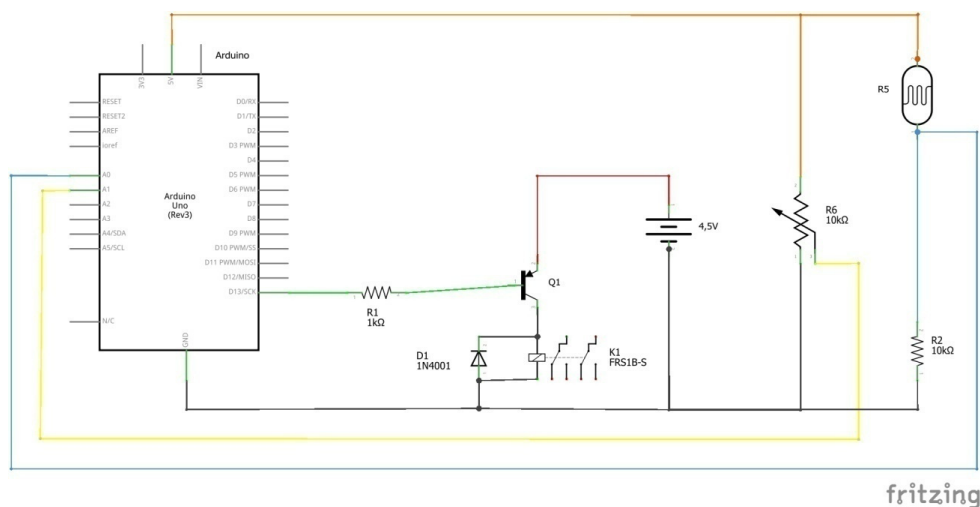


Un ejemplo de programación en S4A:

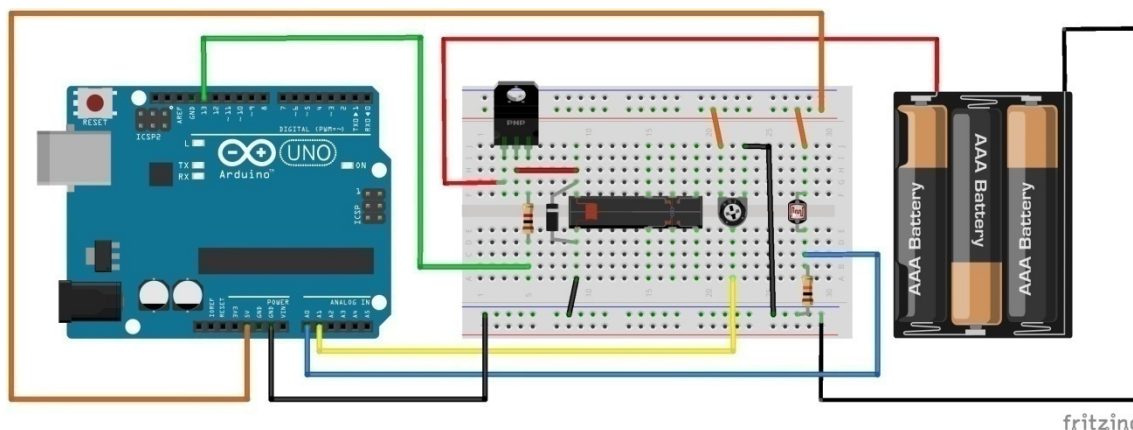


42. Control del accionamiento, en función de la luminosidad ambiental, de un relé, de forma indirecta a través de un transistor PNP BD138, con posibilidad de elegir y modificar el nivel de referencia de luminosidad.

El esquema eléctrico es el siguiente:



El esquema de conexionado y montaje:



Un ejemplo de programación en S4A:

```

al presionar [bandera]
digital 13 encendido
por siempre
  fijar luminosidad a valor del sensor Analog0
  fijar luminosidad deseada a valor del sensor Analog1
  si < luminosidad < luminosidad deseada
    repetir hasta que luminosidad > luminosidad deseada + 50
    digital 13 apagado
    fijar luminosidad a valor del sensor Analog0
    fijar luminosidad deseada a valor del sensor Analog1
  si no
    repetir hasta que luminosidad < luminosidad deseada - 50
    digital 13 encendido
    fijar luminosidad a valor del sensor Analog0
    fijar luminosidad deseada a valor del sensor Analog1
  
```

**43. Control del encendido y apagado de un diodo led y de un zumbador en función de una temperatura máxima.**

Antes de empezar a trabajar esta práctica sería muy interesante repasar los contenidos de la práctica 14, concretamente el divisor de tensión y los sistemas de introducir señales Pull-down o lógica directa y Pull-up o lógica inversa.

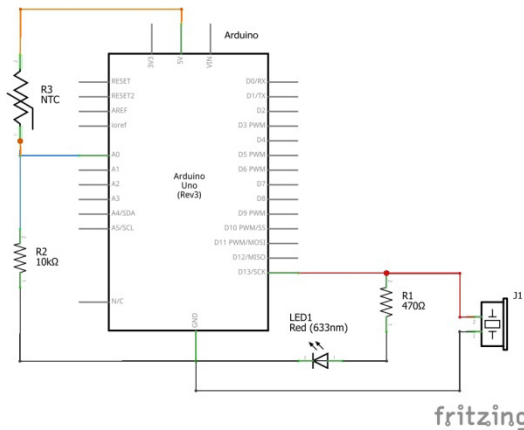
También se trabajará el funcionamiento de una resistencia NTC y PTC mediante el libro digital [tecno12-18.com](http://tecno12-18.com) <sup>12</sup> apartado termistor.

Un termistor o resistencia variable con la temperatura NTC (Negative Temperature Coefficient) es una resistencia la cual, si la temperatura aumenta su valor de resistencia disminuye y a la inversa, si la temperatura disminuye su valor de resistencia aumenta.

Un termistor o resistencia variable con la temperatura PTC (Positive Temperature Coefficient) es una resistencia la cual, si la temperatura aumenta su valor

de resistencia aumenta y a la inversa, si la temperatura disminuye su valor de resistencia disminuye.

Combinando los dos tipos de termistores y las dos lógicas de introducción de señales, podríamos hacer 4 montajes diferentes. Hay que hacer mucho hincapié en que los alumnos entiendan perfectamente el funcionamiento en los 4 casos ya que de ello va a depender la correcta programación. A continuación se muestran los 4 montajes y la correspondiente explicación:



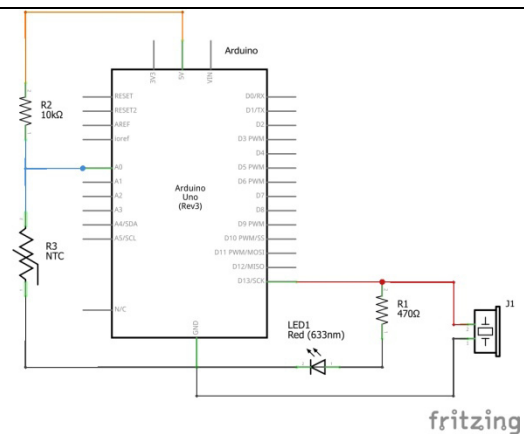
$$U_{A0} = U \cdot \frac{R_2}{R_{NTC} + R_2}$$

$$T^a \downarrow \rightarrow R_{NTC} \uparrow \rightarrow U_{A0} \downarrow \approx 0V$$

$$T^a \uparrow \rightarrow R_{NTC} \downarrow \rightarrow U_{A0} \uparrow \approx U=5V$$

Con este sistema observamos que conforme la temperatura va aumentando, la tensión que llega a la entrada analógica también va aumentando (lógica directa)

NTC lógica directa o Pull-down



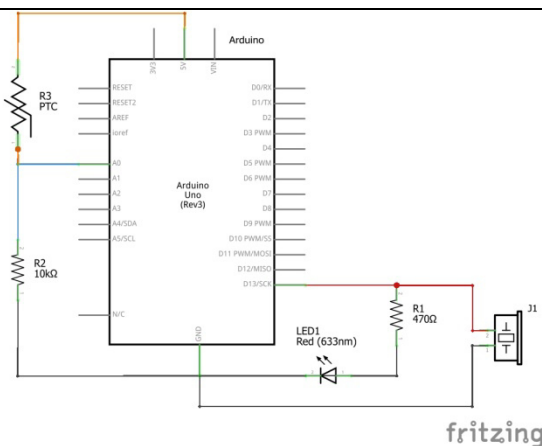
$$U_{A0} = U \cdot \frac{R_{NTC}}{R_{NTC} + R_2}$$

$$T^a \downarrow \rightarrow R_{NTC} \uparrow \rightarrow U_{A0} \uparrow \approx U=5V$$

$$T^a \uparrow \rightarrow R_{NTC} \downarrow \rightarrow U_{A0} \downarrow \approx 0V$$

Con este sistema observamos que conforme la temperatura va aumentando, la tensión que llega a la entrada analógica va disminuyendo (lógica inversa)

NTC lógica inversa o Pull-up



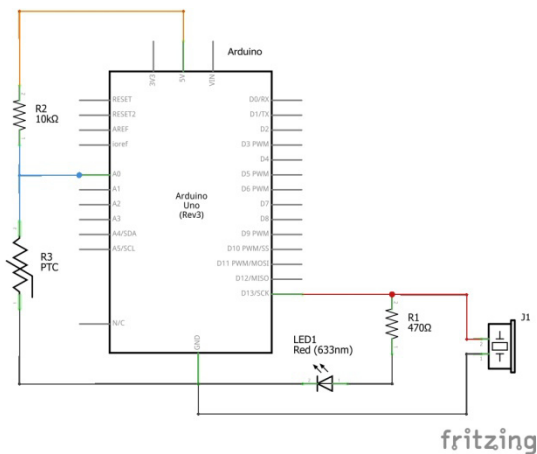
$$U_{A0} = U \cdot \frac{R_2}{R_{PTC} + R_2}$$

$$T^a \downarrow \rightarrow R_{PTC} \downarrow \rightarrow U_{A0} \uparrow \approx U=5V$$

$$T^a \uparrow \rightarrow R_{PTC} \uparrow \rightarrow U_{A0} \downarrow \approx 0V$$

Con este sistema observamos que conforme la temperatura va aumentando, la tensión que llega a la entrada analógica va disminuyendo (cambio a lógica inversa por el comportamiento de la PTC)

PTC lógica directa Pull-down



$$U_{A0} = U \cdot \frac{R_{PTC}}{R_{PTC} + R_2}$$

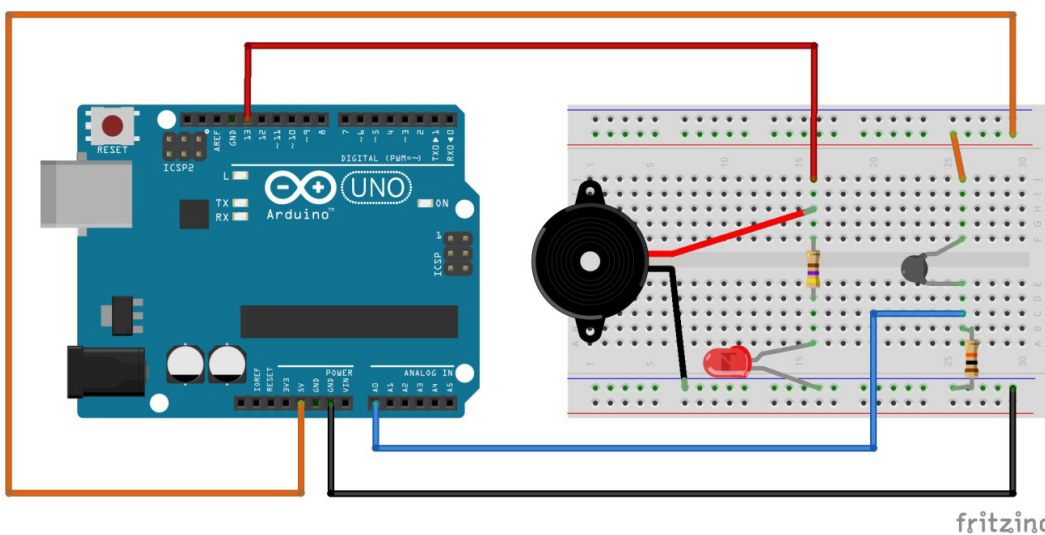
$$T^a \downarrow \rightarrow R_{PTC} \downarrow \rightarrow U_{A0} \downarrow \approx 0V$$

$$T^a \uparrow \rightarrow R_{PTC} \uparrow \rightarrow U_{A0} \uparrow \approx U=5V$$

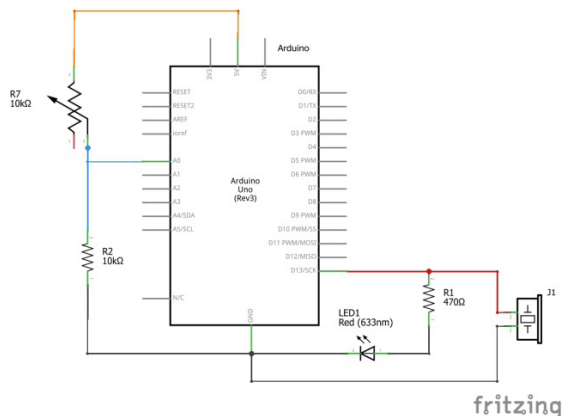
Con este sistema observamos que conforme la temperatura va aumentando, la tensión que llega a la entrada analógica también va aumentando (cambio a lógica directa por el comportamiento de la PTC)

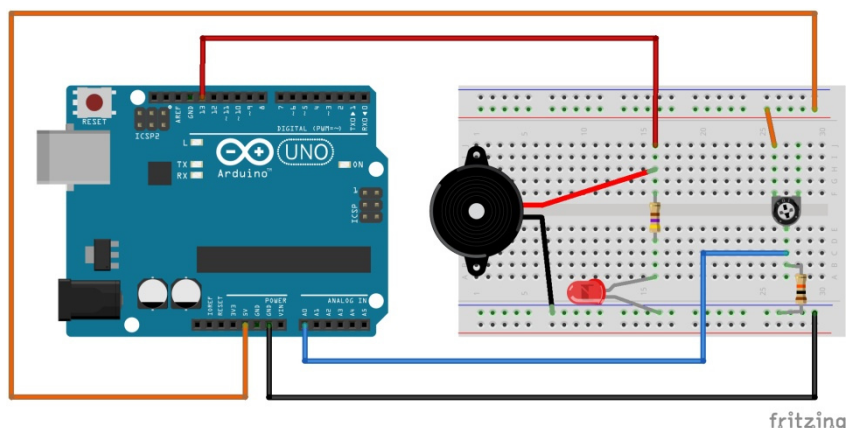
PTC lógica inversa o Pull-up

A continuación podemos ver el esquema de conexionado y montaje de la práctica con NTC y lógica directa o Pull-down:



Para trabajar la práctica tenemos la dificultad de elevar la temperatura del termistor. Suele hacerse con un mechero o con un secador de pelo, pero es muy difícil acercarlo al termistor y trabajar sin quemar cables o la placa board. Para solucionar este problema y si el alumno ha entendido bien cómo funcionan los dos tipos de termistores, podemos reemplazarlo por una resistencia variable. Con esta solución el esquema eléctrico y el de conexionado y montaje serían los siguientes:

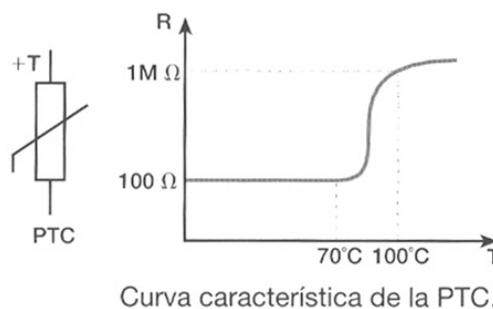
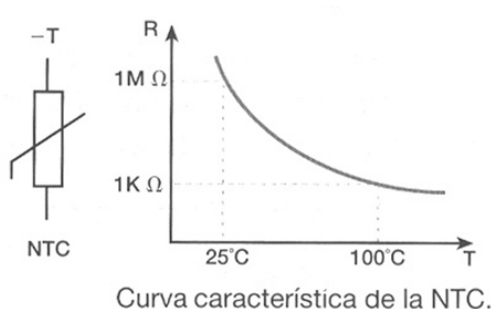




Un ejemplo de programación en S4A:

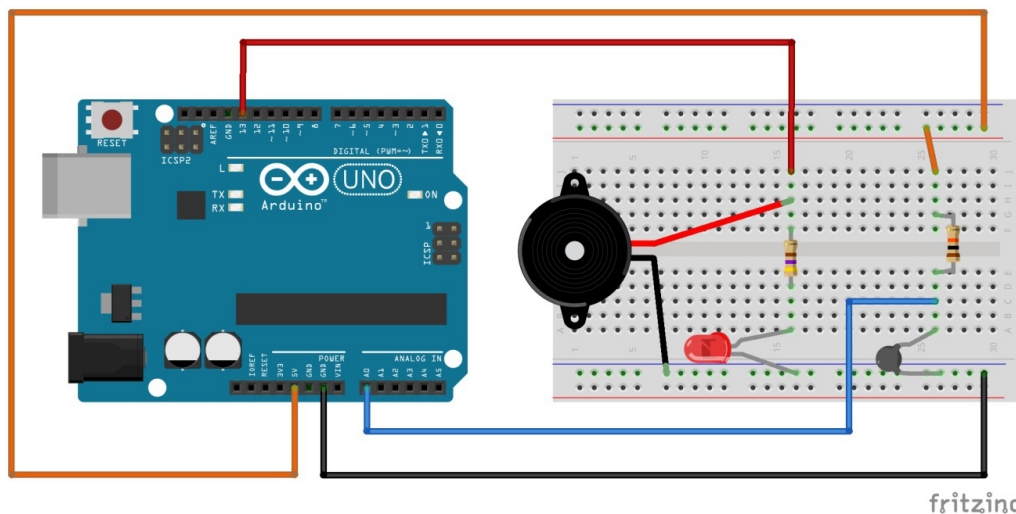


El valor de 750 con el que comparamos es un valor práctico y orientativo. Para cada tipo de resistencia el valor límite en el que queremos que dispare será diferente. Hay que tener en cuenta que si conociéramos la curva característica o la ecuación característica del termistor utilizado, para una temperatura máxima fijada, podríamos calcular el valor exacto de la resistencia del termistor y a continuación el valor de tensión analógica que se correspondería con la temperatura máxima fijada.



Si observamos las gráficas, en un termistor NTC la variación de resistencia con la temperatura es bastante suave en todo el rango de temperaturas mientras que en el termistor PTC esta variación es muy brusca en un escaso margen de temperatura. Por lo tanto utilizaremos los termistores NTC como sensores de temperatura mientras que los termistores PTC los usaremos como elementos de protección contra sobrecalentamientos en dispositivos eléctricos y electrónicos.

El esquema de conexionado y montaje de la práctica con NTC y lógica inversa o Pull-up:



fritzing

Recordamos que para poder realizar la simulación es suficiente con sustituir el termistor NTC por una resistencia variable.

La programación en S4A:



#### 44. Control del encendido y apagado de un diodo led y de un zumbador en función de una temperatura mínima.

Los esquemas eléctricos, de conexionado y montaje son los mismos que los de la práctica anterior.

La programación en S4A para NTC y lógica directa o Pull-down:



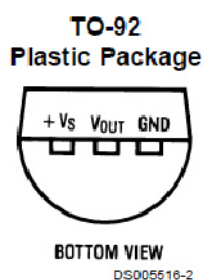


La programación en S4A para NTC y lógica inversa o Pull-up:



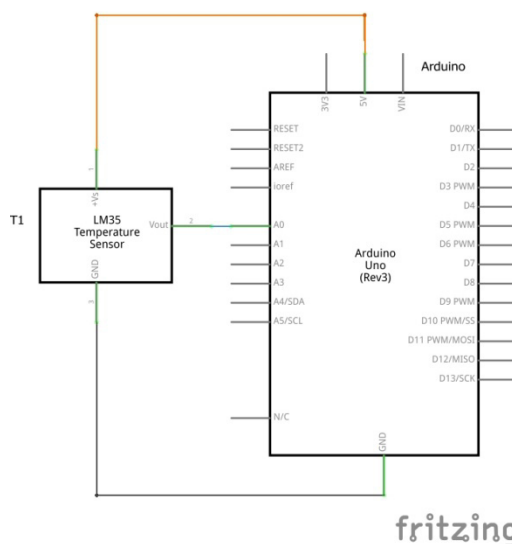
**45. Termómetro digital con LM35 con indicación de la temperatura en pantalla.**

El LM35 es un sensor de temperatura muy sencillo de utilizar. Se alimenta entre 0 y 5V y nos devuelve una tensión proporcional y lineal con la temperatura, en un rango entre 2 y 150°C. A continuación mostramos su esquema de patillaje y la ecuación de funcionamiento <sup>18</sup>:



$$U_{\text{SENSOR}} = 10\text{mV}/^{\circ}\text{C} \cdot T (^{\circ}\text{C}) \text{ (de } 2^{\circ}\text{C a } 150^{\circ}\text{C)}$$

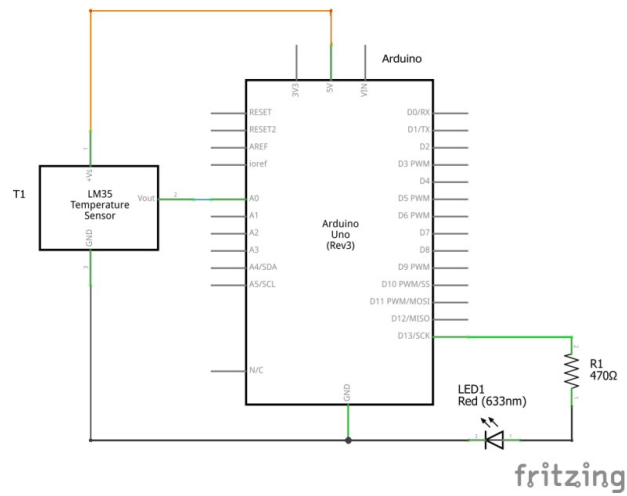
El esquema eléctrico es el siguiente:



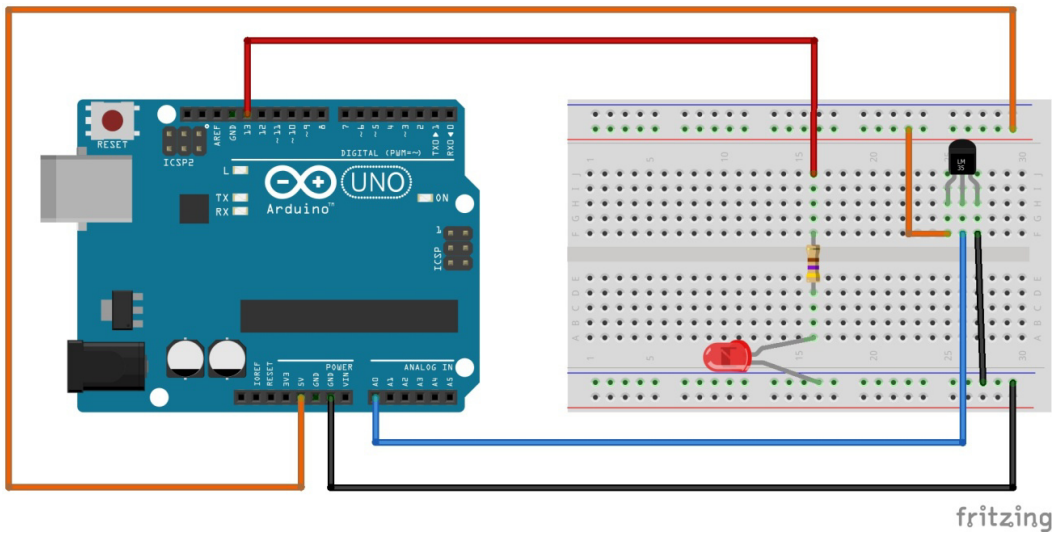


### 46. Termostato básico con LM35.

El esquema eléctrico es el siguiente:



El esquema de conexionado y montaje:

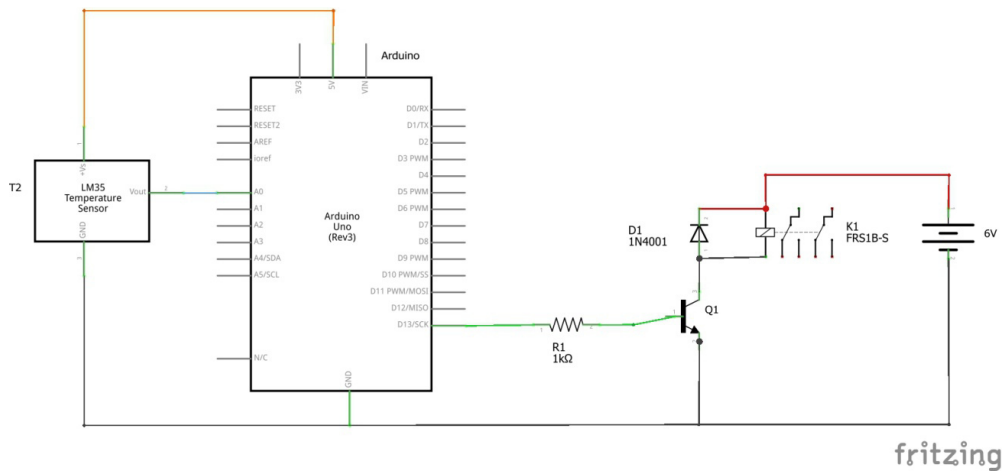


Un ejemplo de programación en S4A:

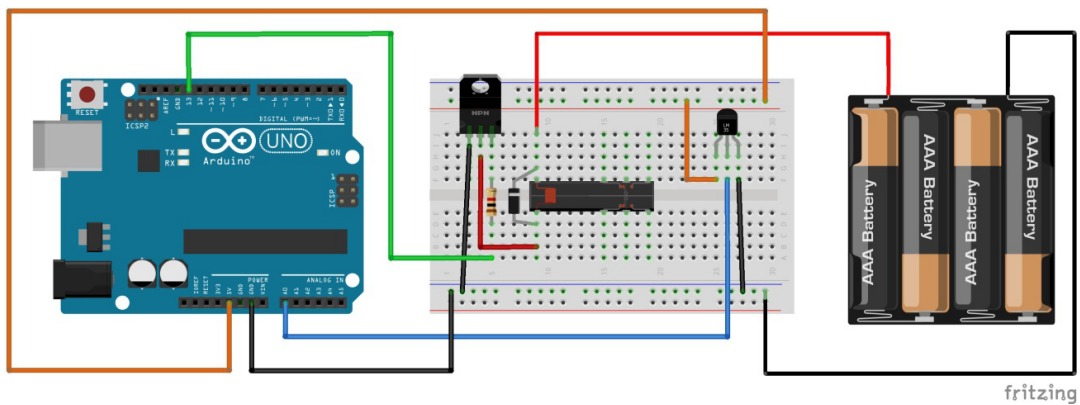


**47. Control del accionamiento, en función de la temperatura ambiental, de un relé, de forma indirecta a través de un transistor NPN BD135 (termostato avanzado).**

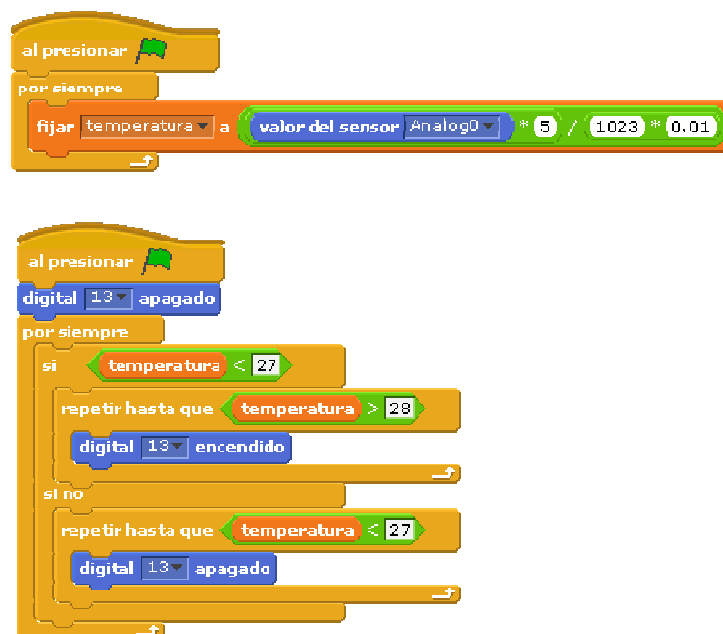
El esquema eléctrico es el siguiente:



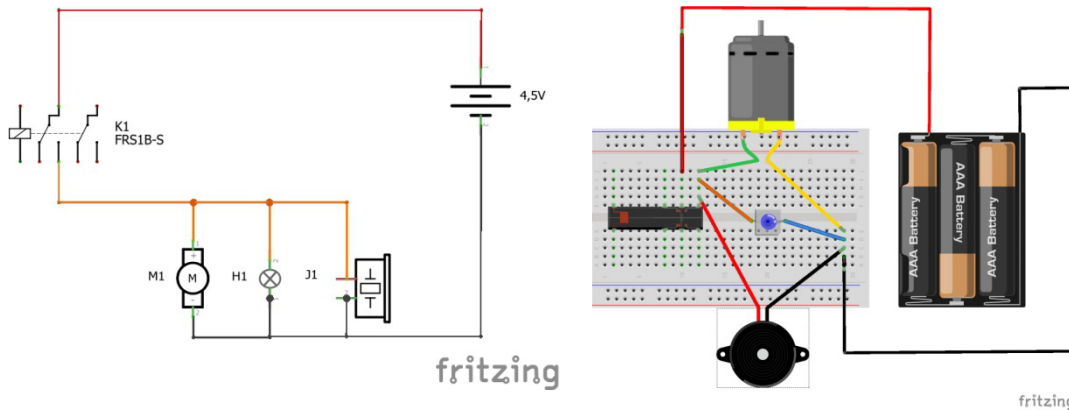
El esquema de conexionado y montaje:



Un ejemplo de programación en S4A:

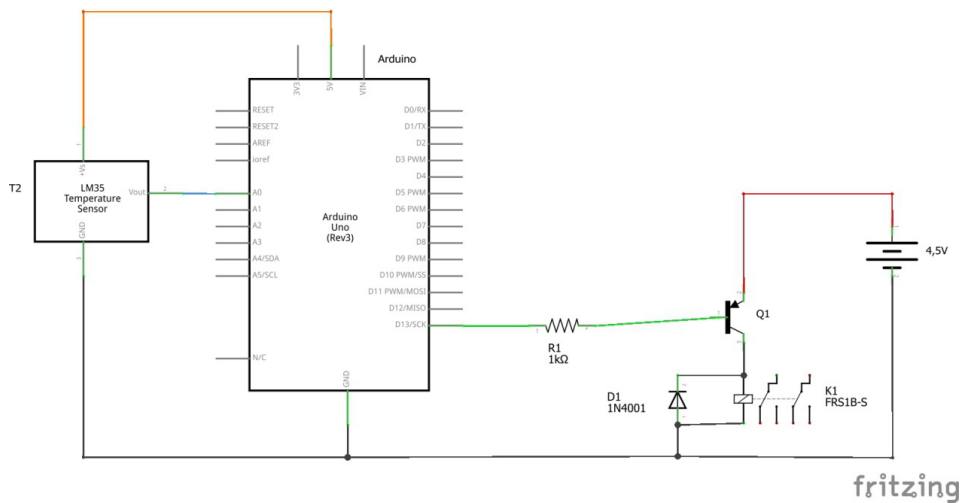


En estas prácticas y todas las que vienen a continuación hasta la práctica 50, con el fin de no complicar en exceso la representación de los esquemas eléctricos, no se representan los circuitos de potencia o circuitos a controlar, es decir, caldera de calefacción, aire acondicionado, circuitos de refrigeración, ... A continuación se representa como podría ser uno de ellos.

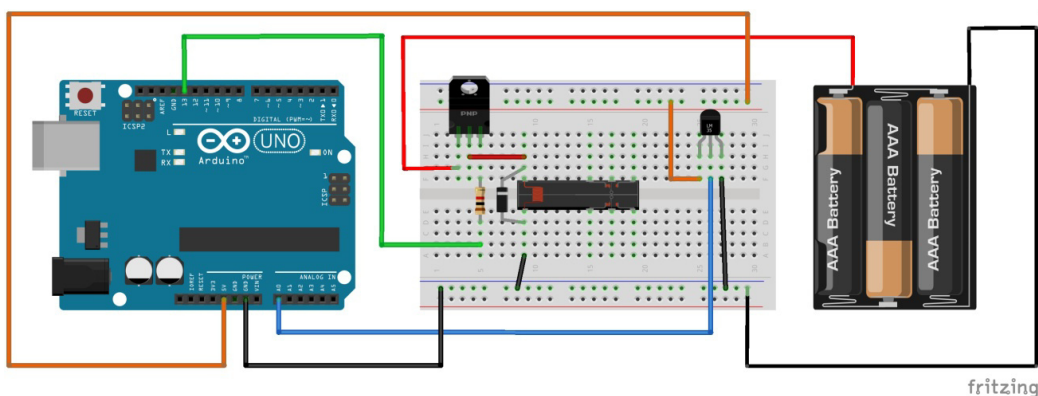


**48. Control del accionamiento, en función de la temperatura ambiental, de un relé, de forma indirecta a través de un transistor PNP BD138 (termostato avanzado).**

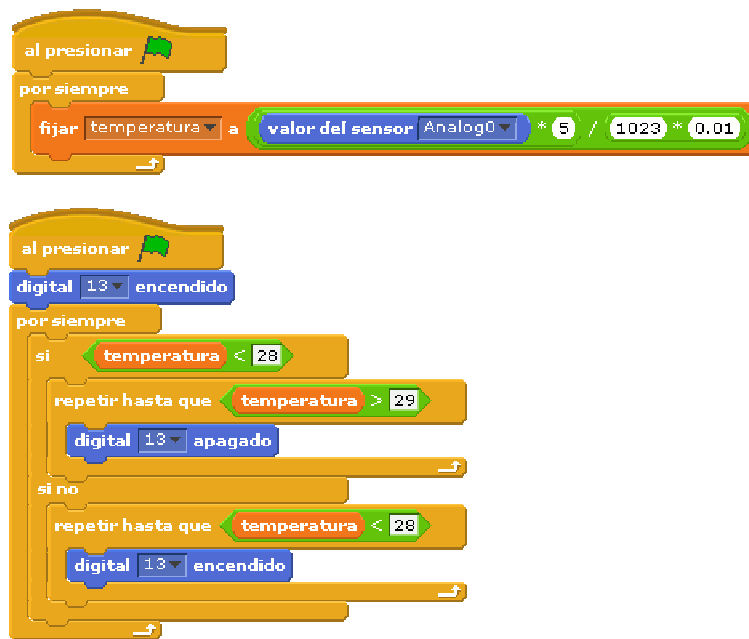
El esquema eléctrico es el siguiente:



El esquema de conexionado y montaje:

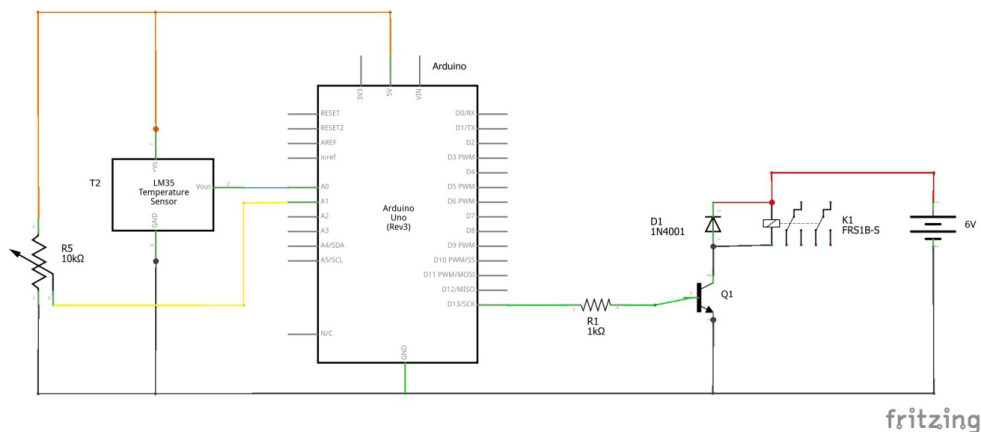


Un ejemplo de programación en S4A:

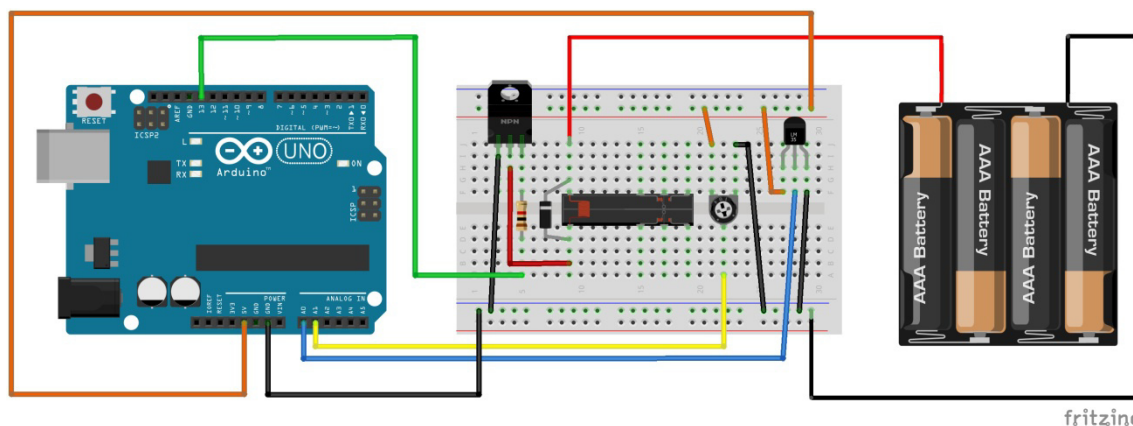


**49. Control del accionamiento, en función de la temperatura ambiental, de un relé, de forma indirecta a través de un transistor NPN BD135, con posibilidad de elegir y modificar el nivel de referencia de temperatura (termostato completo).**

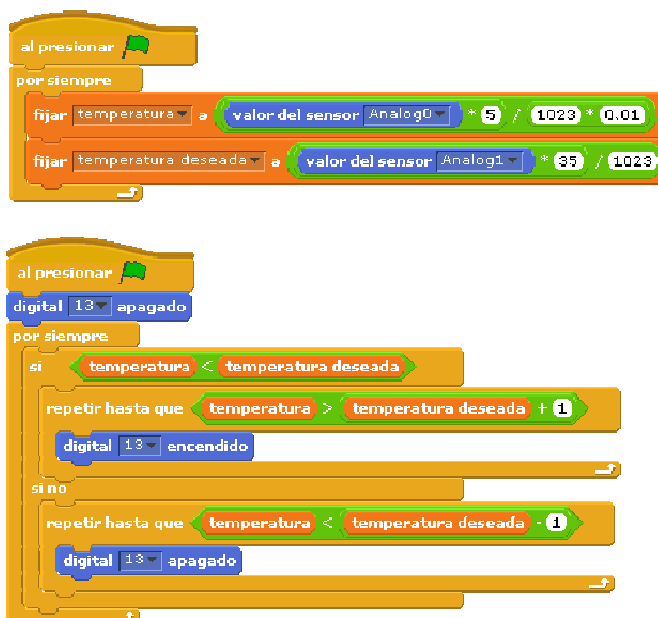
El esquema eléctrico es el siguiente:



El esquema de conexionado y montaje:



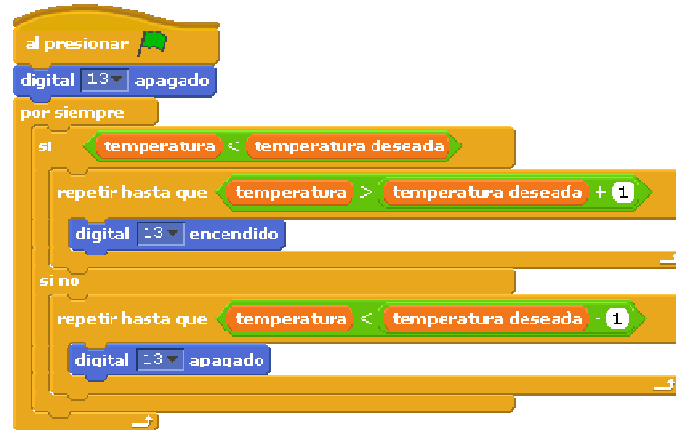
Un ejemplo de programación en S4A para una selección de temperatura entre 0 y 35°C:



Un ejemplo de programación en S4A para una selección de temperatura entre 5 y 35°C. Para ello, al no ser el origen 0°C, se tiene que realizar la interpolación lineal que se explica a continuación:

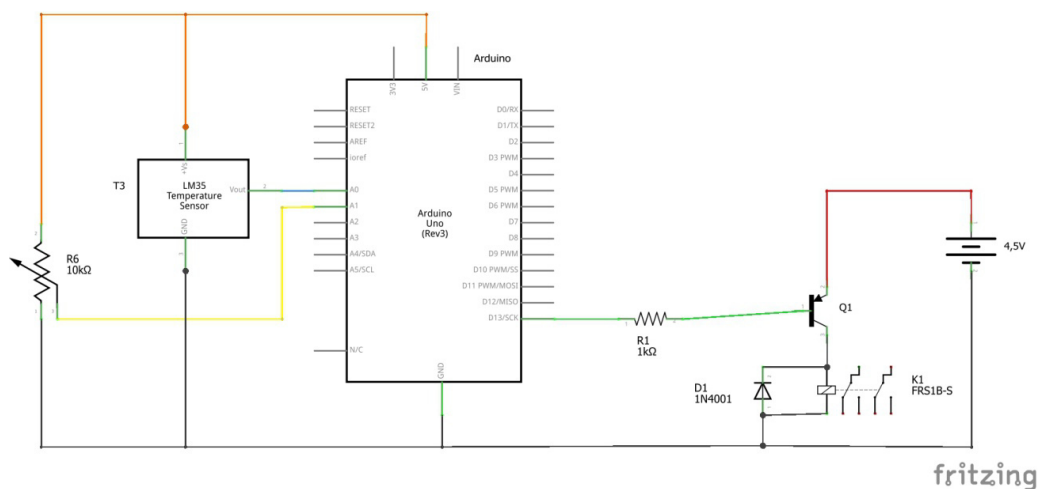
0	—————	5°C	}	$\frac{x - x_0}{x_1 - x_0} = \frac{y - y_0}{y_1 - y_0} \rightarrow \frac{Vana \log}{1023} = \frac{T^a - 5}{30}$
Val.analog	—————	T <sup>a</sup>		
1023	—————	35°C		

$$T^a = \frac{30}{1023} Vana \log + 5$$



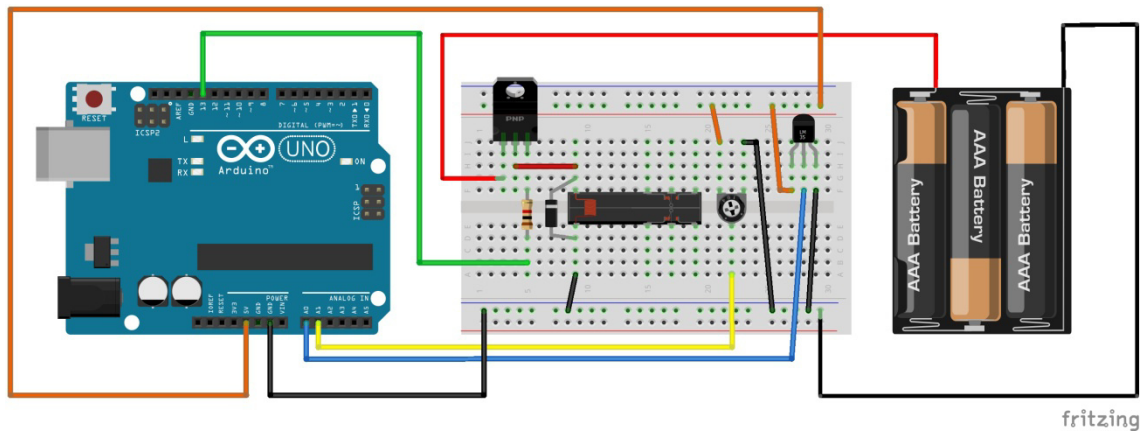
50. Control del accionamiento, en función de la temperatura ambiental, de un relé, de forma indirecta a través de un transistor PNP BD138, con posibilidad de elegir y modificar el nivel de referencia de temperatura (termostato completo).

El esquema eléctrico es el siguiente:

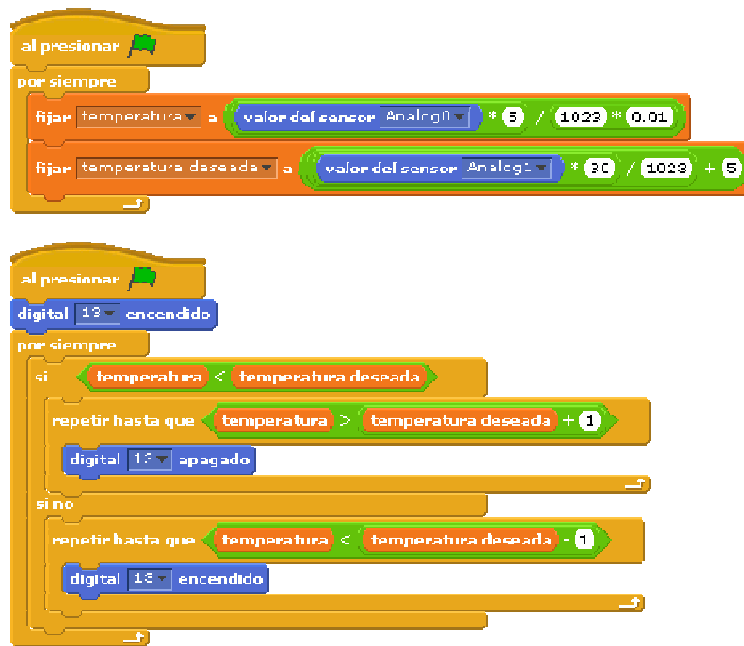


El esquema de montaje y conexionado:



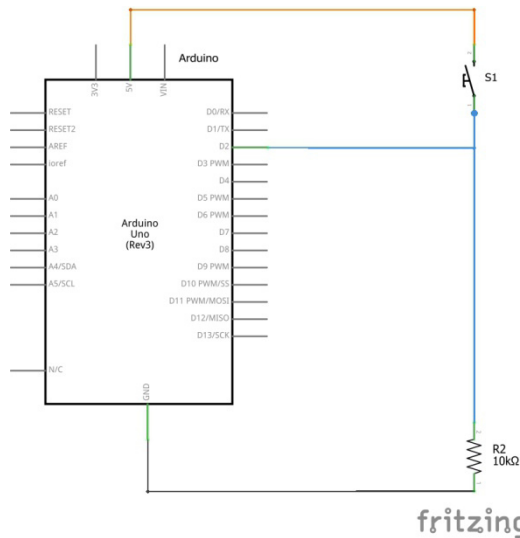


Un ejemplo de programación en S4A:



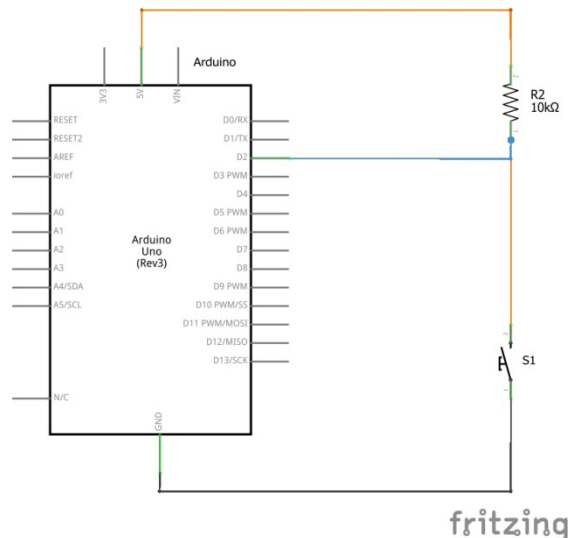
### 51. Contador +1.

Esta práctica la vamos a realizar con varios elementos diferentes. El primero con un pulsador normalmente abierto. A continuación representamos los esquemas eléctricos, de conexionado y montaje y un ejemplo de programación en S4A:



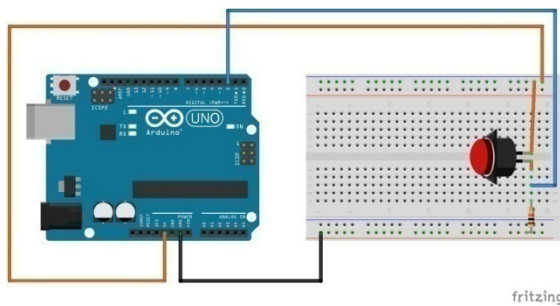
fritzing

Lógica directa o pull-down

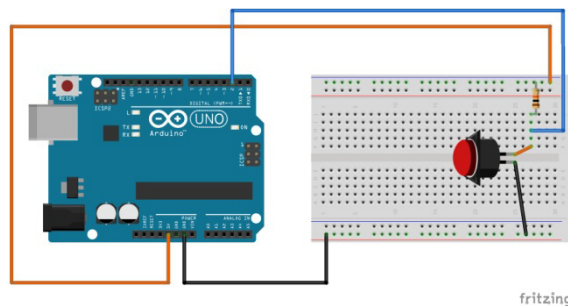


fritzing

Lógica inversa o pull-up



fritzing



fritzing

```

al presionar
  fijar contador a 0
  por siempre
    si ¿sensor Digital2 presionado?
      fijar contador a contador + 1
      esperar hasta que no ¿sensor Digital2 presionado?
  
```

```

al presionar
  fijar contador a 0
  por siempre
    si no ¿sensor Digital2 presionado?
      fijar contador a contador + 1
      esperar hasta que ¿sensor Digital2 presionado?
  
```

En la programación en S4A estamos utilizando un nuevo bloque de la librería control “si \_\_\_”, que nos sirve para ejecutar los bloques de su interior solamente cuando se cumple una condición. También habríamos podido utilizar el bloque “por siempre si \_\_\_”

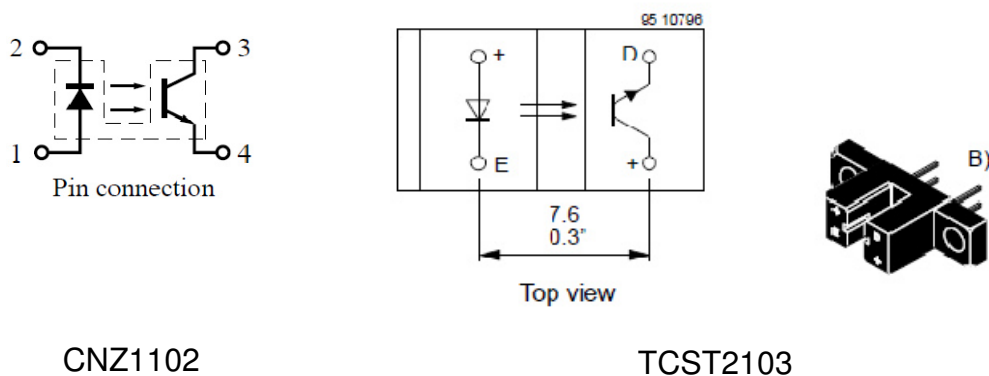


Hay que recordar que ¿sensor digital\_ presionado? no se corresponde con que el pulsador o el accionador que sea estén presionados, sino que se corresponde con la llegada a la correspondiente entrada de 0V o de 5V. Eso cambia en función de la lógica utilizada y en función del tipo de contacto del accionador (normalmente abierto (NA) o normalmente cerrado (NC)).

Si observamos detenidamente, en el sistema de lógica directa o pull-down, el contador va a realizar la cuenta cuando la tensión en la entrada digital 2 pase de 0V a 5V (pulsador pulsado), es decir, en el flanco de subida, y se va a preparar para una nueva cuenta justo a continuación del flanco de bajada, paso de 5V a 0V (pulsador dejado de pulsar). En cambio, en el sistema de lógica inversa o pull-up, el contador va a realizar la cuenta cuando la tensión en la entrada digital 2 pase de 5V a 0V (pulsador pulsado), es decir, en el flanco de bajada, y se va a preparar para una nueva cuenta justo a continuación del flanco de subida, paso de 0V a 5V (pulsador dejado de pulsar).

Como podemos observar, la programación de un contador es muy sencilla. Simplemente se crea una variable con el nombre que nos interese y la modificamos sumando o restando la cantidad que nos interese a su valor actual.

A continuación vamos a realizar la misma práctica pero con un optoacoplador CNZ1102<sup>18</sup> o un optoacoplador TCST2103<sup>18</sup>. El esquema de patillaje es el siguiente:



CNZ1102

TCST2103

Esta práctica la vamos a realizar con el optoacoplador CNZ1102. Se aporta la información del optoacoplador TCST2103 porque es bastante habitual encontrarlo. Hay que tener en cuenta que los terminales de emisor y receptor van completamente al revés, por lo que habrá que tener mucho cuidado a la hora de identificarlos y usarlos.

Como podemos observar, un optoacoplador es un elemento electrónico que nos permite acoplar señales sin que haya contacto mecánico ni movimiento. Consta de un diodo led de infrarrojos y de un fototransistor. En este caso cuando el diodo led emite luz infrarroja, si no hay ningún elemento que interrumpa el paso de luz hacia el fototransistor, este recibe luz infrarroja por su base y trabaja en zona activa, es decir, permite el paso de corriente entre colector y emisor. Si el diodo led no emite luz o hay algún obstáculo que impide que la luz infrarroja llegue a la base del fototransistor, este se encuentra en zona de corte.

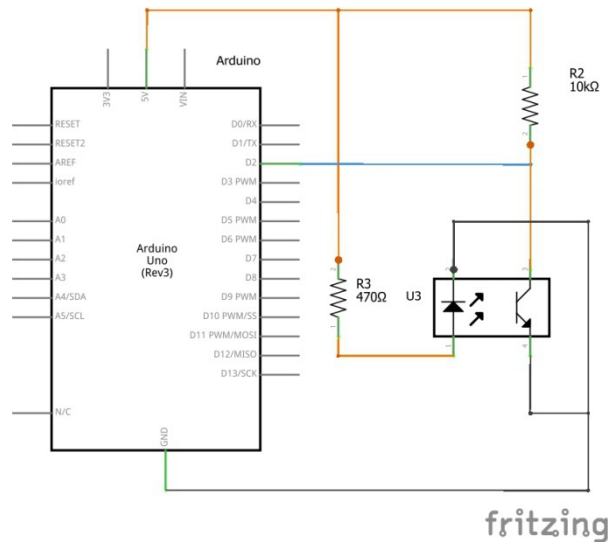
Los relés de estado sólido están basados en el funcionamiento del optoacoplador. Podríamos decir que son un optoacoplador, con la única diferencia que en función de la potencia y del tipo de corriente que se quiere controlar, la parte de potencia puede ser un fototransistor, un tiristor o un triac.

El CNZ1102 o el TCST2103 los podemos usar como relés de estado sólido, pero sobre todo como detectores de algún obstáculo entre emisor y receptor, para contar

objetos que pasan por delante, velocidad de una rueda (velocidad circular de un motor), detector de si una rueda gira o no (ABS), ...

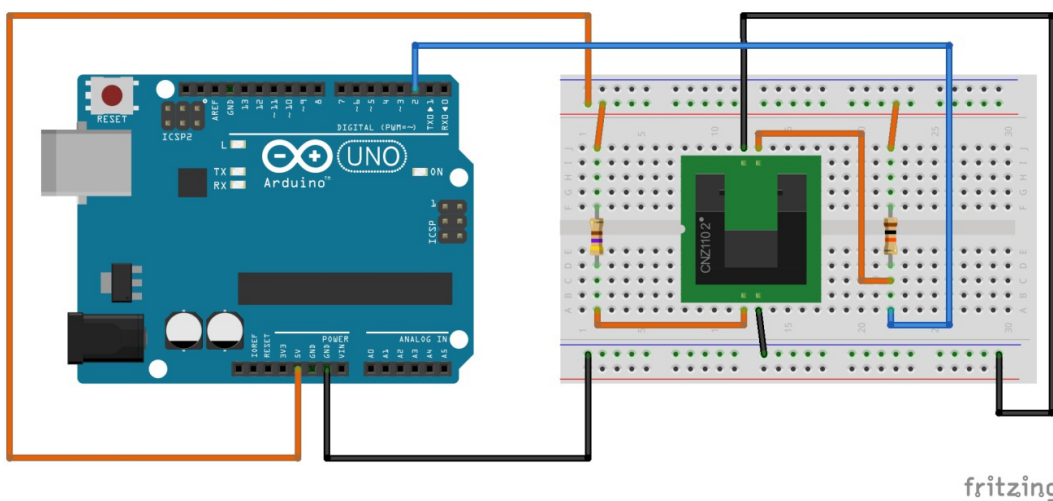
Construyendo el sistema con un diodo led de infrarrojos y un fototransistor por separado obtenemos un sensor de barrera. El emisor y receptor pueden estar colocados uno en cada extremo o los dos en el mismo extremo, utilizando un reflector para desviar el haz infrarrojo desde el emisor al receptor (ver práctica 76).

El esquema eléctrico de la práctica es el siguiente:



Como podemos observar, lo que nos interesa es detectar y contar los obstáculos o las interrupciones que pasan entre el fotodiodo y el fototransistor. Colocando la resistencia de 10K como podemos observar en el esquema, conseguimos que la tensión en la entrada digital 2 sea de 0V si no hay obstáculo (visión directa entre diodo led y fototransistor) y sea de 5V cuando se detecta un obstáculo (no hay visión directa entre diodo led y fototransistor).

El esquema de conexionado y montaje:

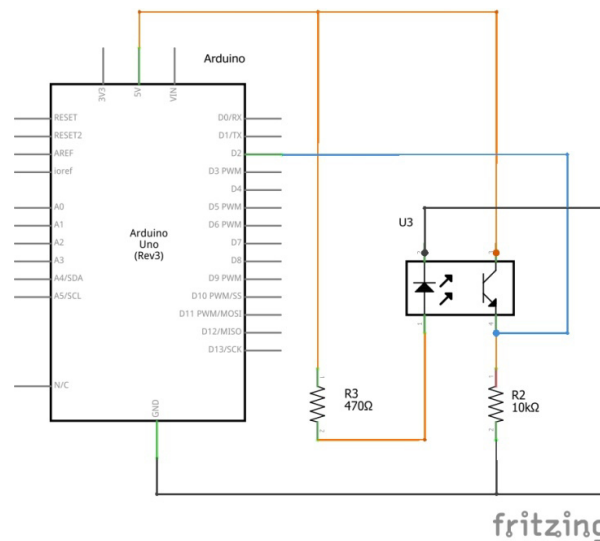


Un ejemplo de programación en S4A:

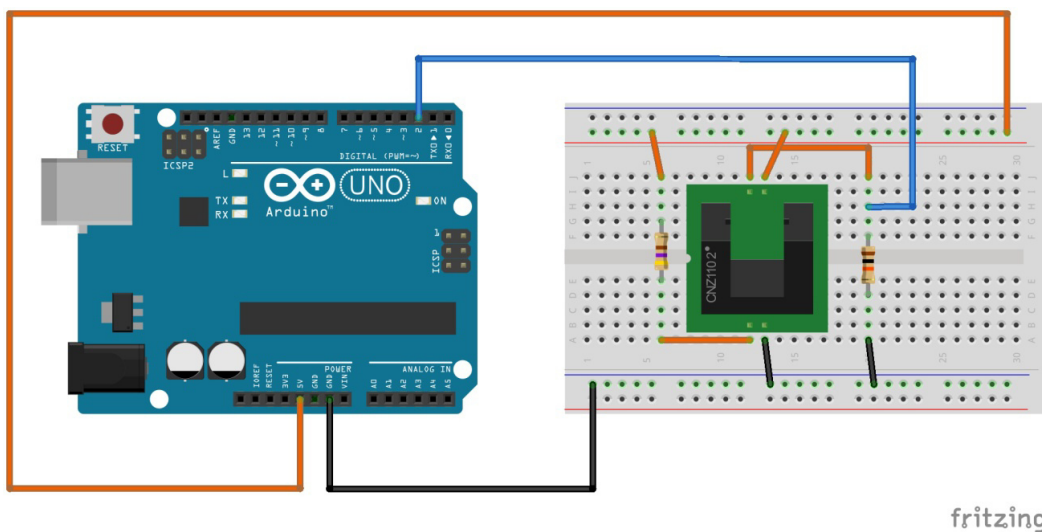


Este montaje podríamos decir que es el de lógica directa. Cuando detecta un obstáculo devuelve 5V e incrementa la cuenta y cuando deja de detectarlo devuelve 0V y se prepara para detectar otro y volver a aumentar la cuenta.

En caso de necesitar la lógica inversa podríamos utilizar el siguiente esquema eléctrico:



El esquema de conexionado y montaje:

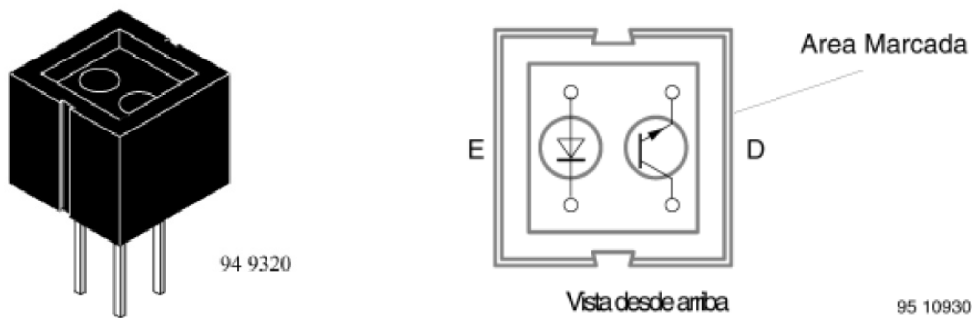


Un ejemplo de programación en S4A:



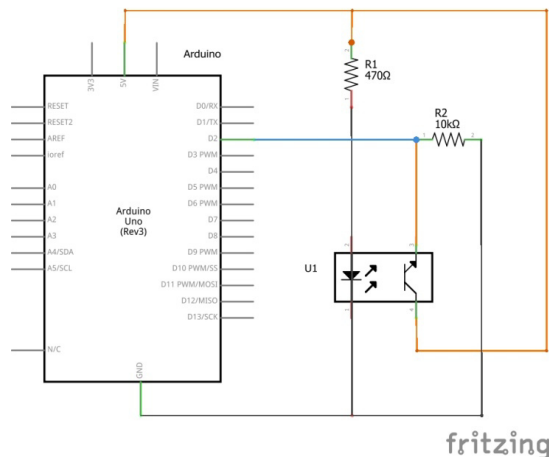
En este caso cuando detecta un obstáculo nos devuelve 0V e incrementa la cuenta y cuando deja de detectarlo devuelve 5V y se prepara para detectar otro y volver a aumentar la cuenta.

Por último vamos a realizar esta misma práctica pero con un optoacoplador CNY70. El esquema de patillaje es el siguiente:

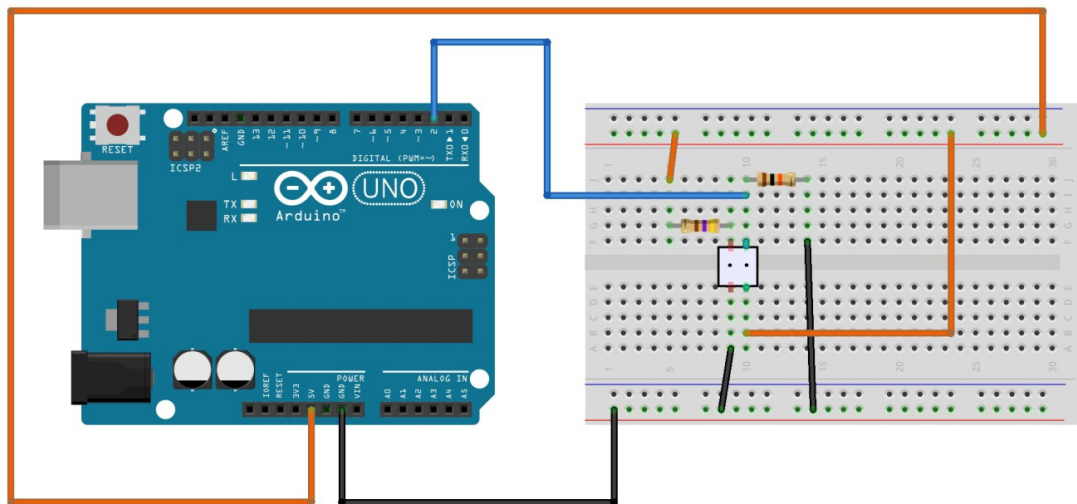


El CNY70 lo podemos usar sobre todo como detector de algún elemento que refleje la luz, ya que el emisor (diodo led de infrarrojos) y el receptor (fototransistor) están colocados de forma paralela y tiene que ser un reflector o el propio elemento a detectar quien refleje el haz infrarrojo; también lo podemos utilizar como detector o clasificador de elementos de diferente color (blanco refleja la luz y negro la absorbe), para un seguidor de trayectoria blanca o negra, ...

El esquema eléctrico de la práctica es el siguiente:



El esquema de conexionado y montaje:



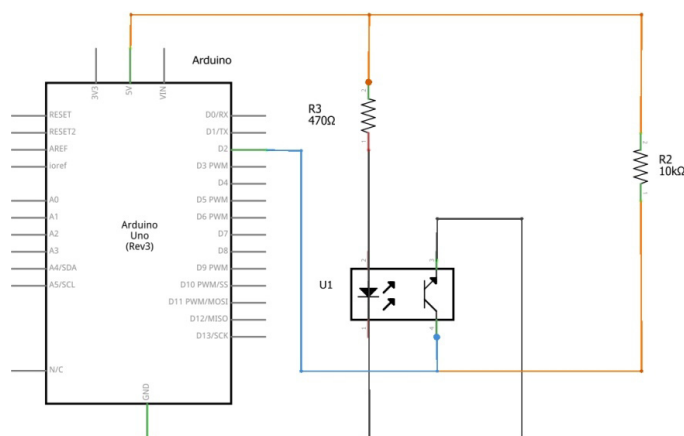
fritzing

Un ejemplo de programación en S4A:



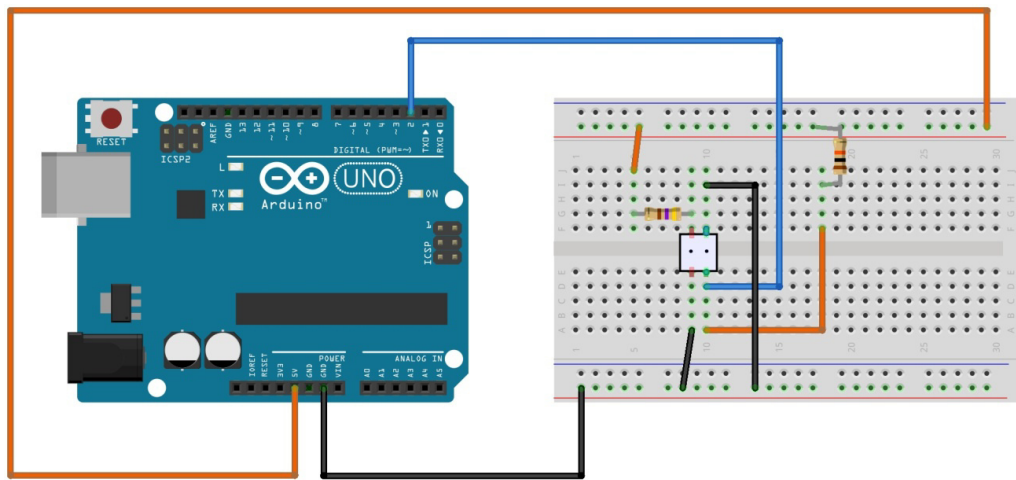
Este montaje podríamos decir que es el de lógica directa. Cuando detecta un objeto que refleja la luz infrarroja devuelve 5V e incrementa la cuenta y cuando deja de detectarlo devuelve 0V y se prepara para detectar otro y volver a aumentar la cuenta.

En caso de necesitar la lógica inversa podríamos utilizar el siguiente esquema eléctrico:



fritzing

El esquema de conexionado y montaje:



fritzing

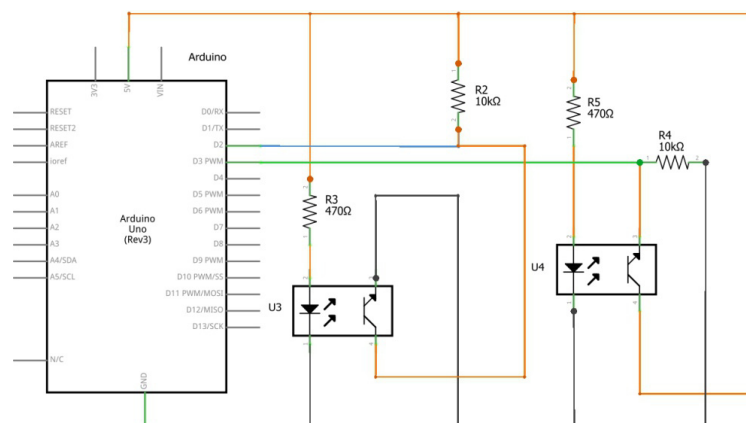
Un ejemplo de programación en S4A:



En este caso cuando detecta un objeto que refleja la luz infrarroja nos devuelve 0V e incrementa la cuenta y cuando deja de detectarlo devuelve 5V y se prepara para detectar otro y volver a aumentar la cuenta.

### 52. Contador +1 -1.

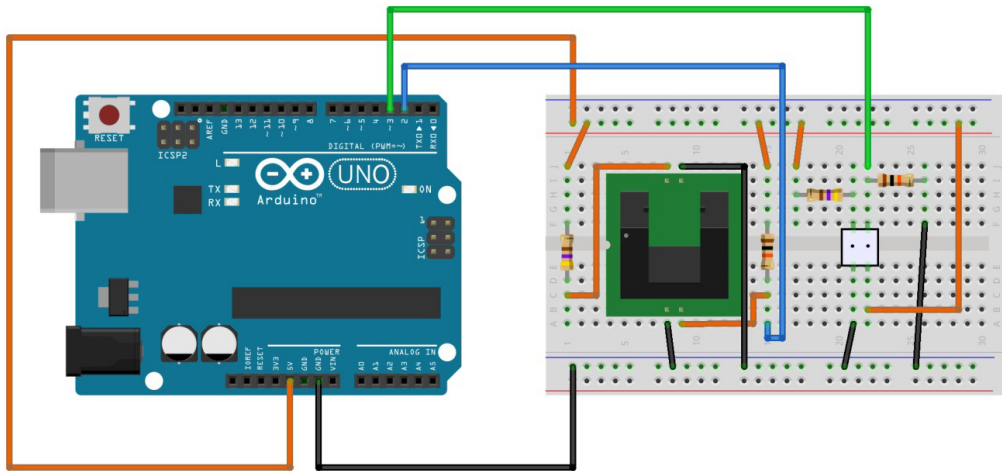
El esquema eléctrico de la práctica es el siguiente:



fritzing



El esquema de conexionado y montaje:



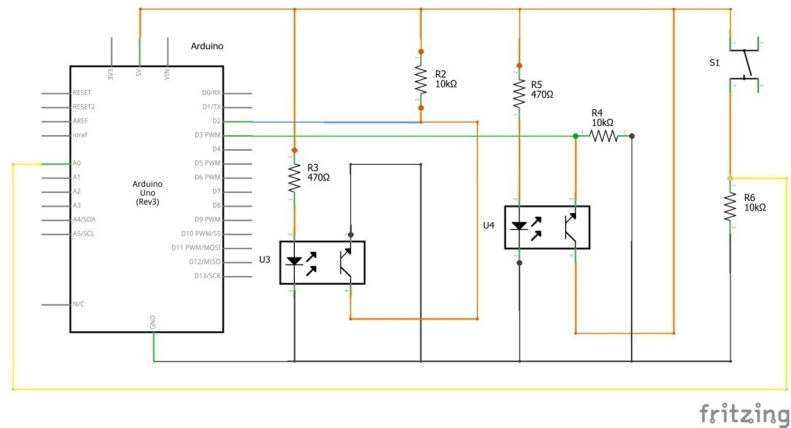
fritzing

Un ejemplo de programación en S4A:

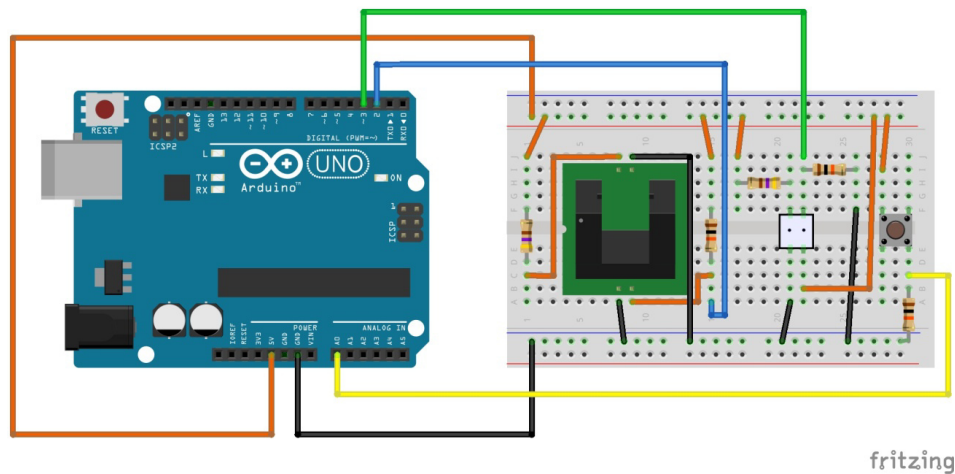


### 53. Contador +1 -1 reset.

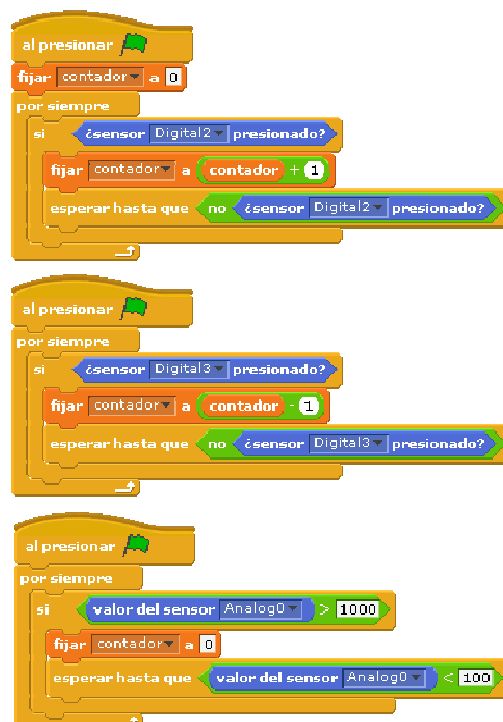
El esquema eléctrico de la práctica es el siguiente:



El esquema de conexionado y montaje:



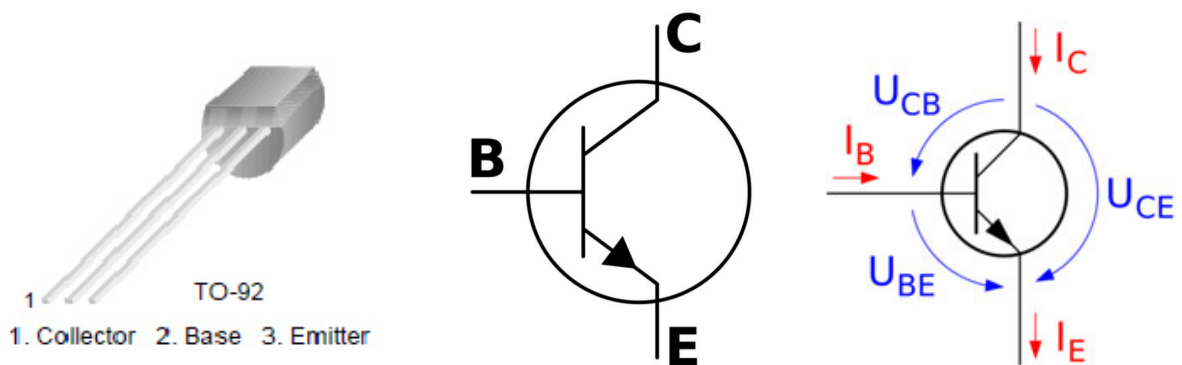
Un ejemplo de programación en S4A:



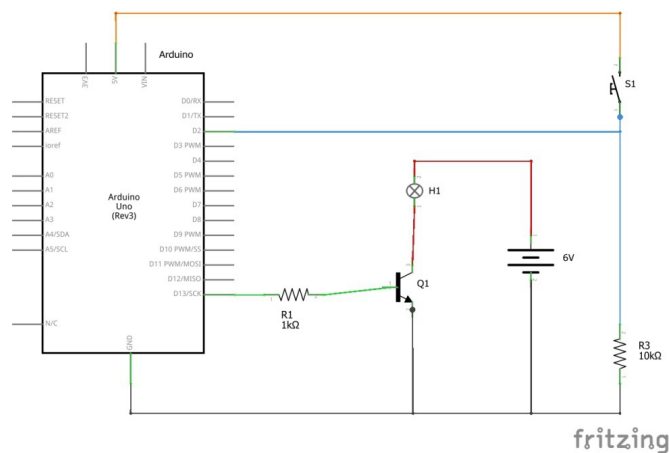
### 54. Control del encendido y apagado de una lámpara incandescente mediante un transistor NPN BC547 (Ic max = 100 mA).

Una vez visto el funcionamiento del transistor, estas prácticas surgen con el fin de ver y trabajar con un encapsulamiento diferente para necesidades de potencia menores y para ver cómo podemos trabajar en la regulación de la intensidad y por lo tanto en la regulación de potencia de un receptor, lo que físicamente significará la regulación de la intensidad luminosa de una lámpara incandescente o la regulación de la velocidad de un motor.

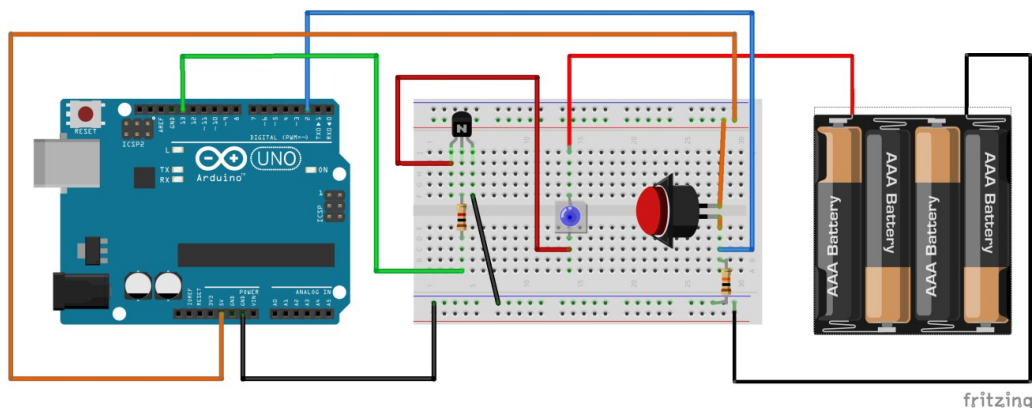
A continuación podemos ver los esquemas de patillaje, terminales, tensiones y corrientes de un transistor NPN BC547 <sup>18</sup> :



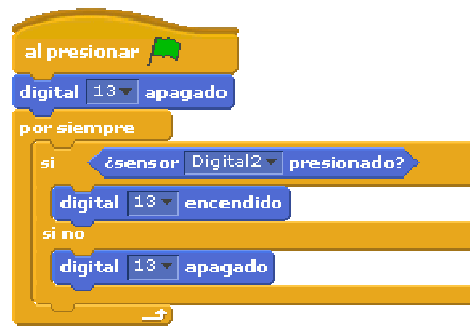
El esquema eléctrico de la práctica es el siguiente:



El esquema de conexionado y montaje:

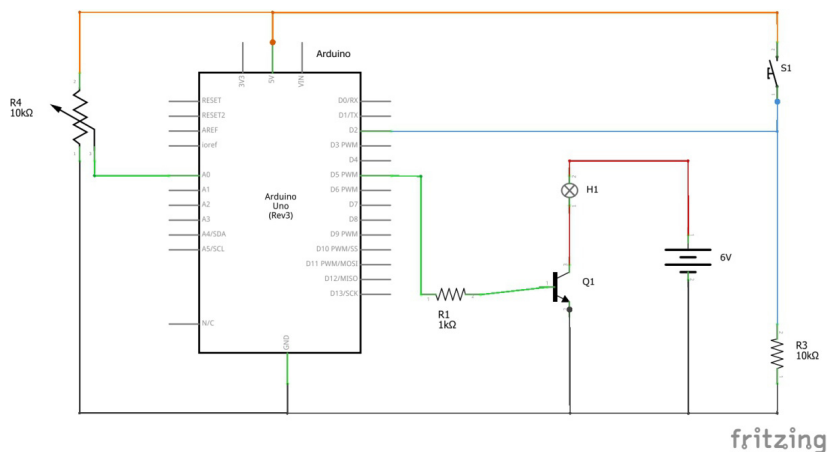


Un ejemplo de programación en S4A:

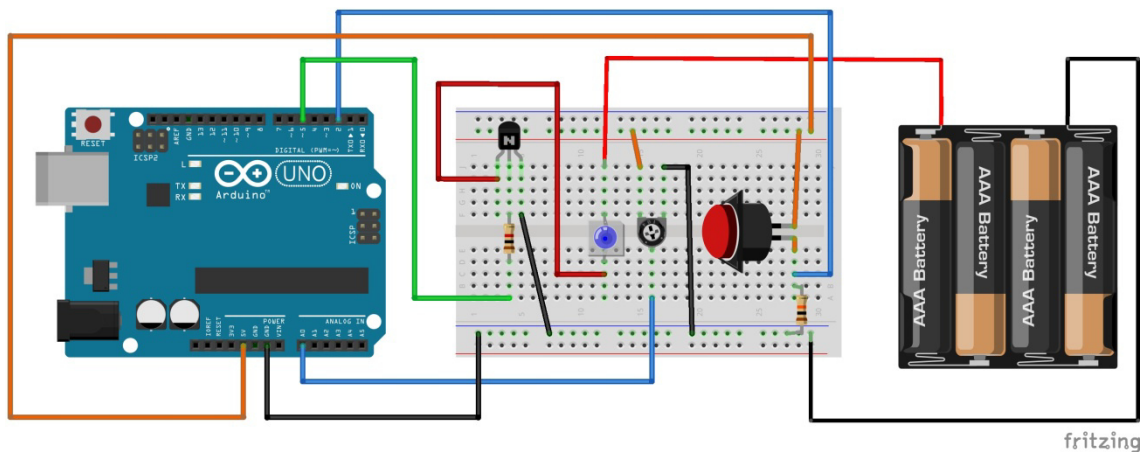


**55.Regulación y control de la intensidad luminosa de una lámpara incandescente y de su encendido y apagado, de forma indirecta, a través de un transistor NPN BC547.**

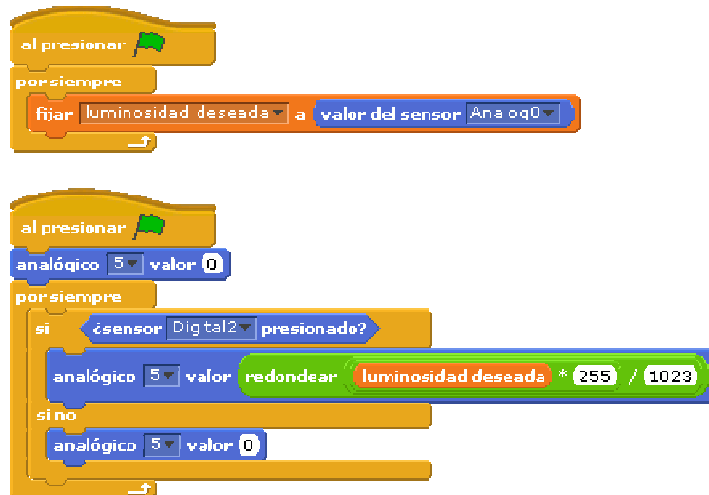
El esquema eléctrico de la práctica es el siguiente:



El esquema de conexionado y montaje:

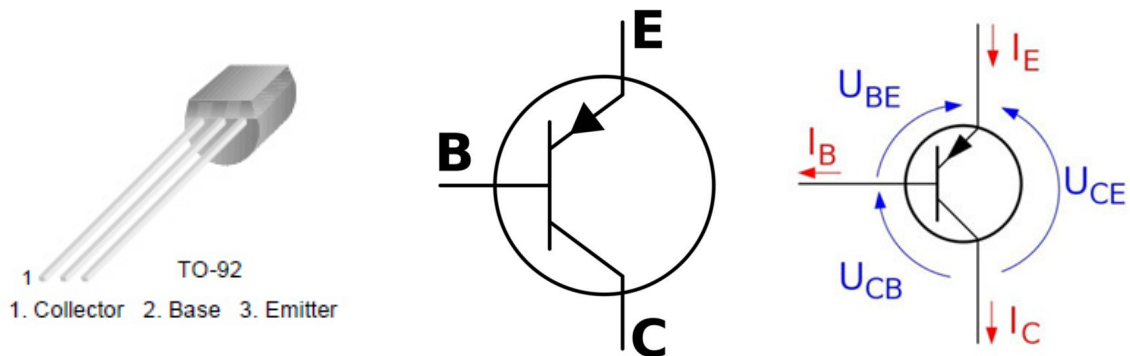


Un ejemplo de programación en S4A:

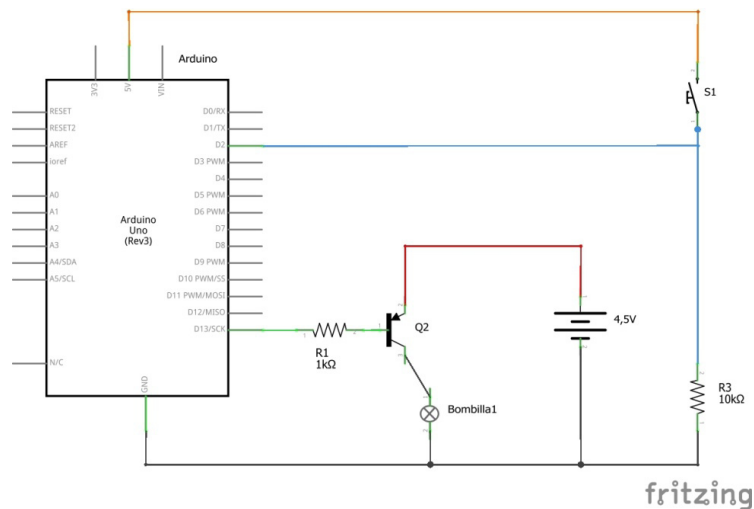


### 56. Control del encendido y apagado de una lámpara incandescente mediante un transistor PNP BC557.

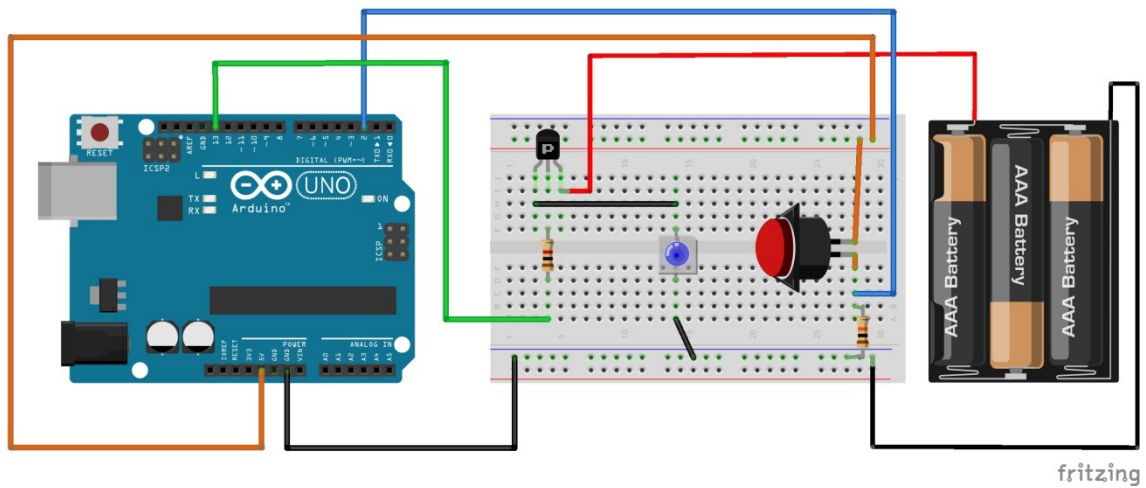
A continuación podemos ver el esquema de patillaje, terminales, tensiones y corrientes y de un transistor PNP BC557 <sup>18</sup> :



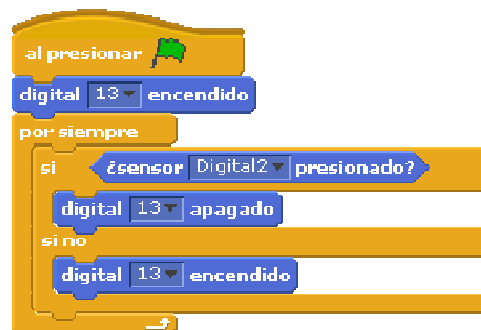
El esquema eléctrico de la práctica es el siguiente:



El esquema de conexionado y montaje:

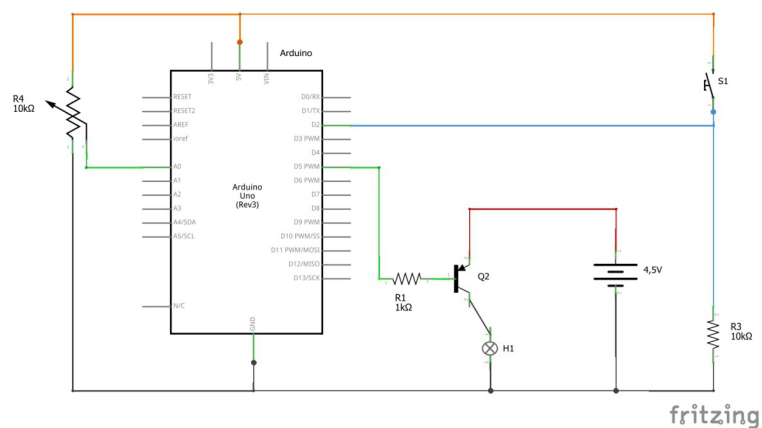


Un ejemplo de programación en S4A:

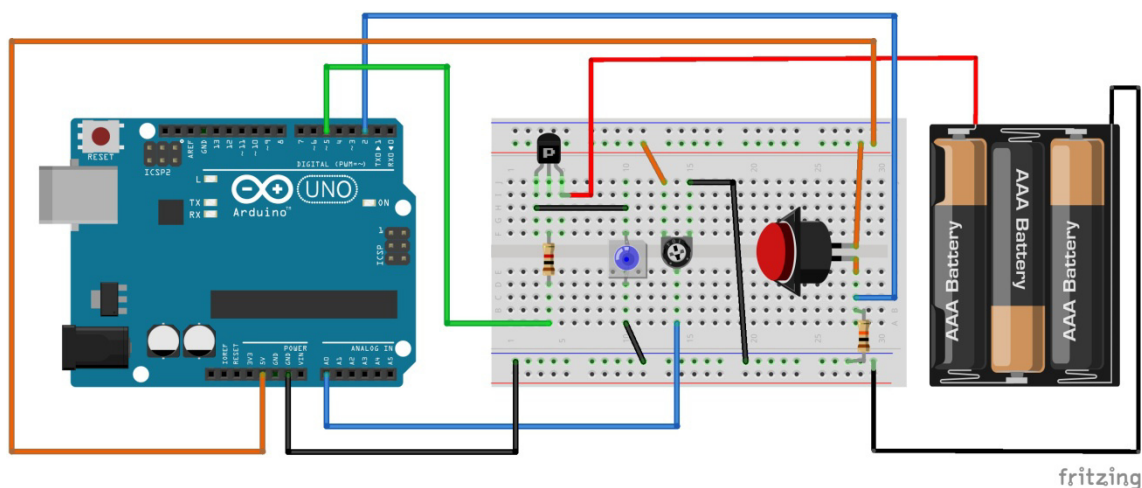


**57.Regulación y control de la intensidad luminosa de una lámpara incandescente y de su encendido y apagado, de forma indirecta, a través de un transistor PNP BC557.**

El esquema eléctrico de la práctica es el siguiente:



El esquema de conexionado y montaje:



Un ejemplo de programación en S4A:

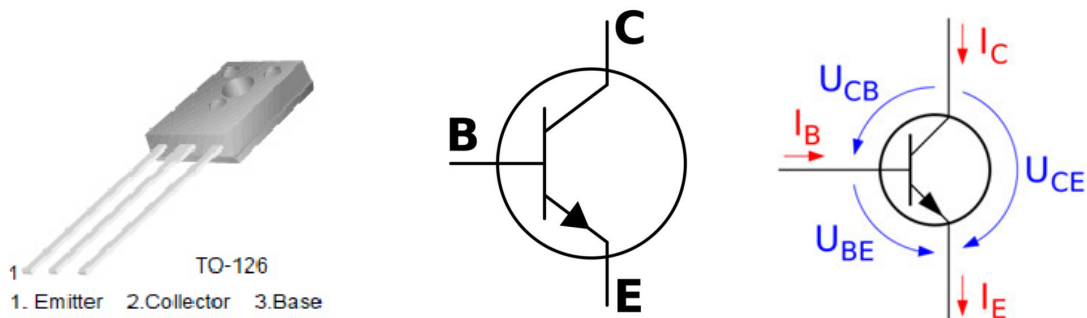


Es importante hacer ver a los alumnos que la luminosidad de la lámpara incandescente va al revés que la indicada en pantalla, ya que en un transistor PNP, con una entrada de luminosidad deseada máxima de 1023 que se corresponde con 5V, la salida analógica será de 255 que también corresponde con 5V, de forma que el transistor estará en corte y la luminosidad real de la lámpara incandescente será 0 (apagada). Lo mismo sucede para una entrada de luminosidad desea mínima. Los alumnos deberán intentar corregir este desajuste. Un ejemplo de programación en S4A para la corrección:

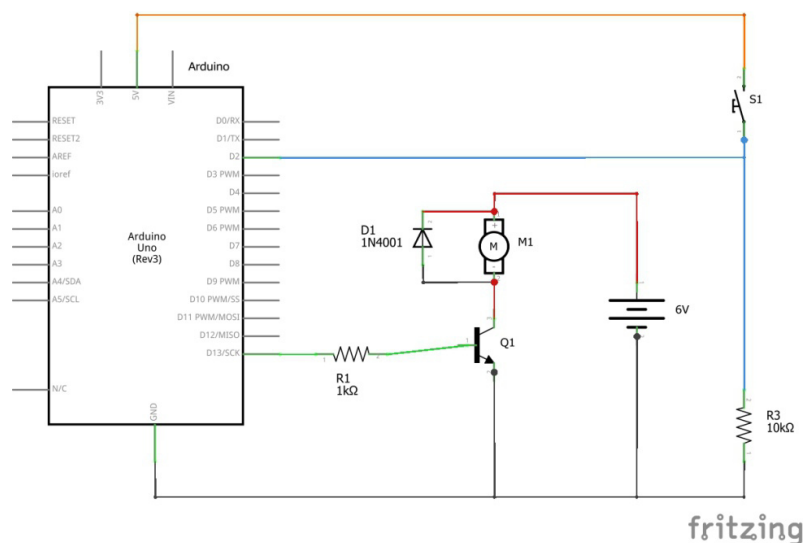


**58. Control del encendido y apagado de un motor, de forma indirecta, a través de un transistor NPN BD135.**

En primer lugar repasaremos el esquema de patillaje, terminales, tensiones y corrientes del transistor NPN BD135 <sup>18</sup> :

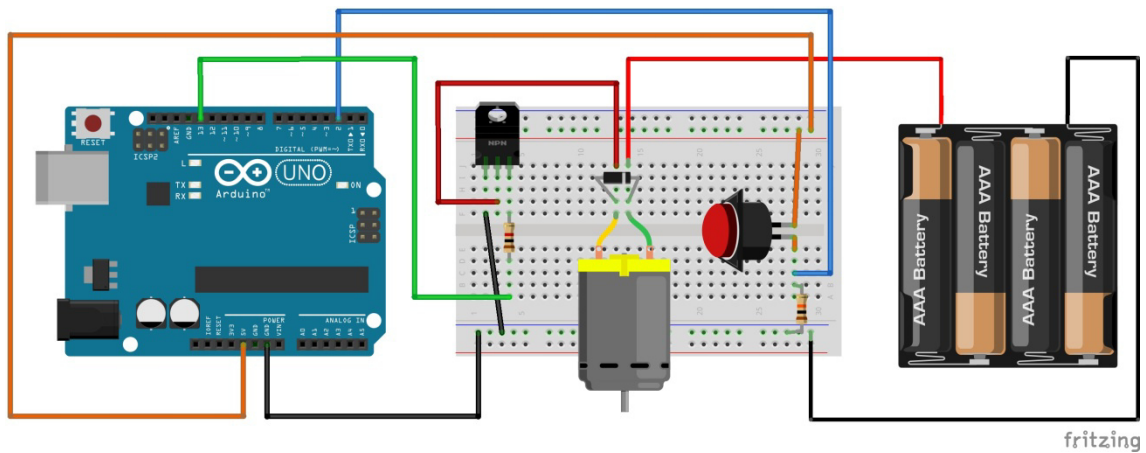


El esquema eléctrico de la práctica es el siguiente:





El esquema de conexionado y montaje:

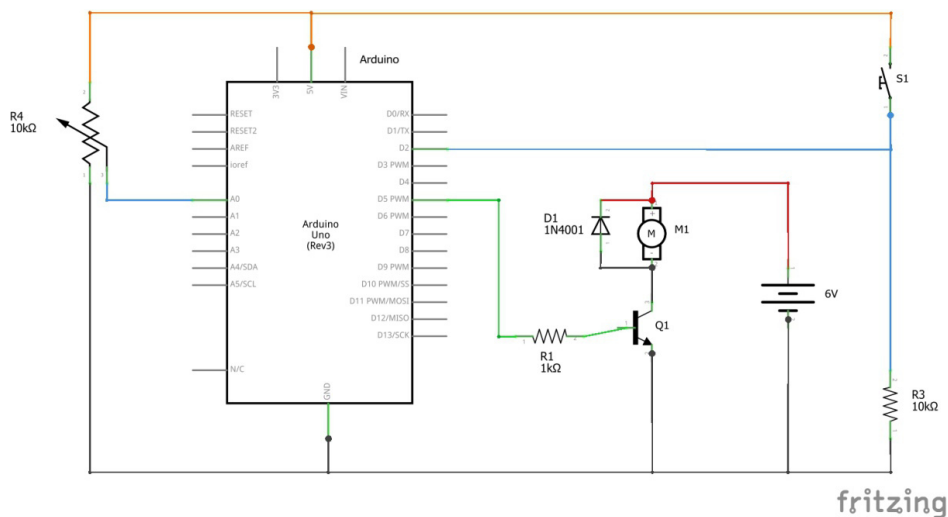


Un ejemplo de programación en S4A:

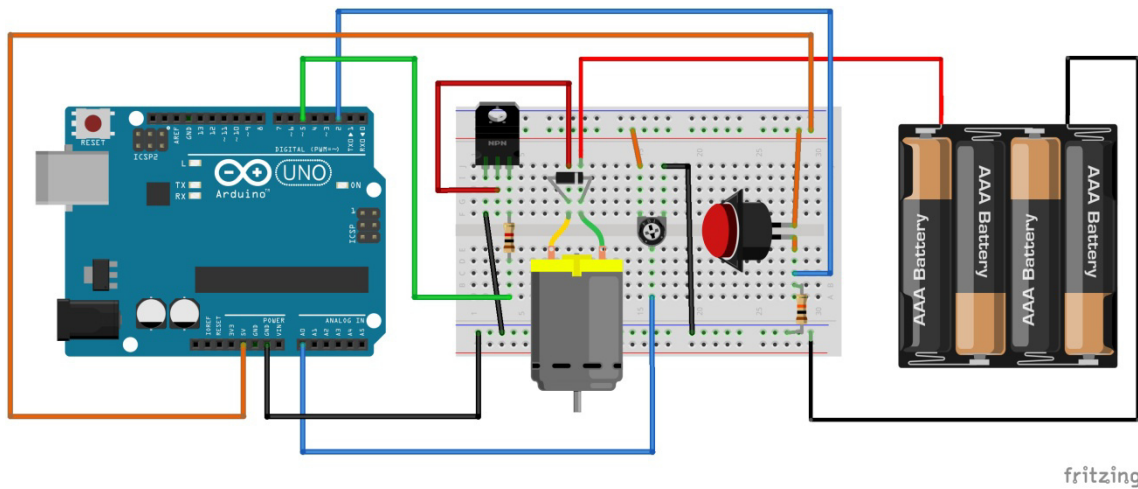


**59.Regulación y control de la velocidad y del encendido y apagado de un motor, de forma indirecta, a través de un transistor NPN BD135.**

El esquema eléctrico de la práctica es el siguiente:



El esquema de conexionado y montaje:



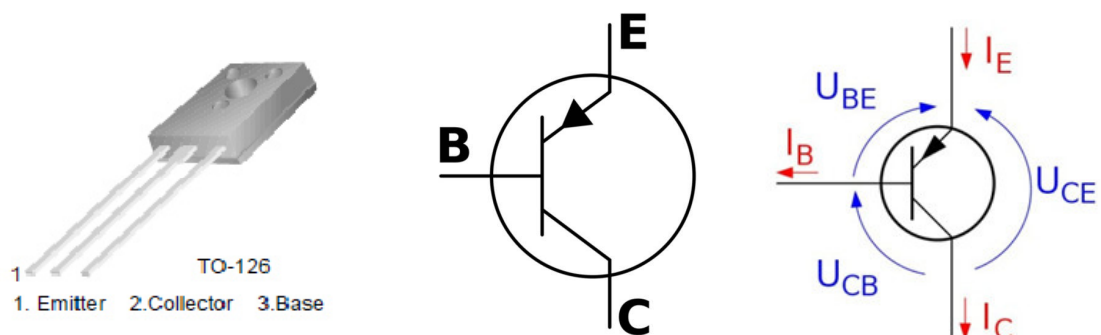
fritzing

Un ejemplo de programación en S4A:

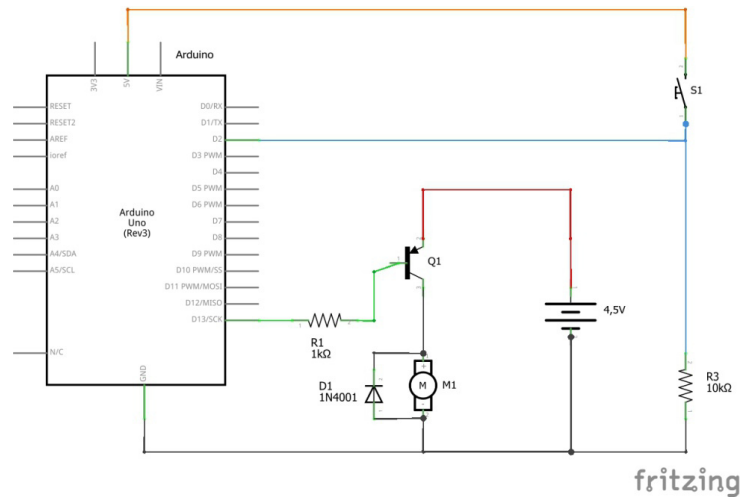


**60. Control del encendido y apagado de un motor, de forma indirecta, a través de un transistor PNP BD138.**

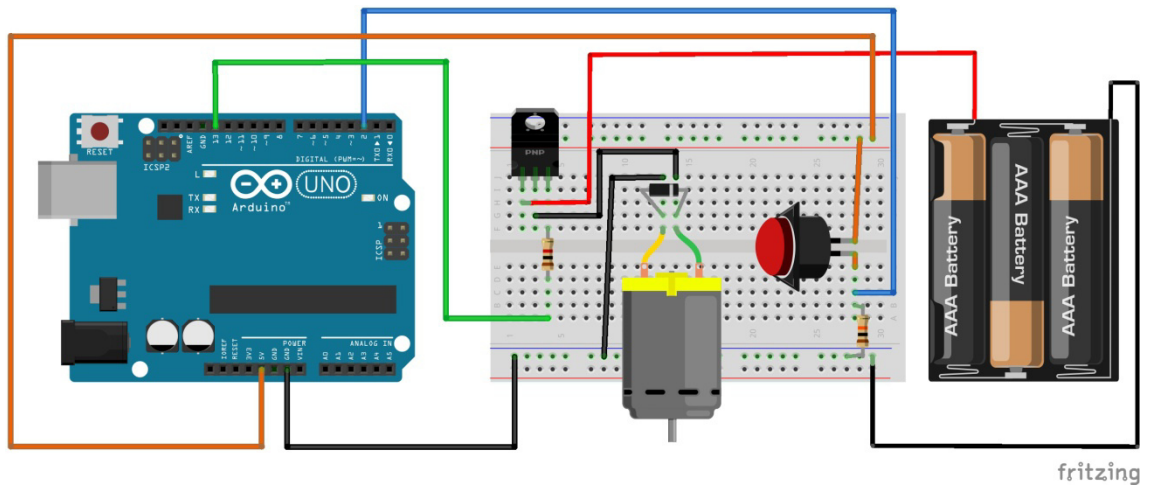
En primer lugar repasaremos el esquema de patillaje, terminales, tensiones y corrientes del transistor NPN BD138 <sup>18</sup> :



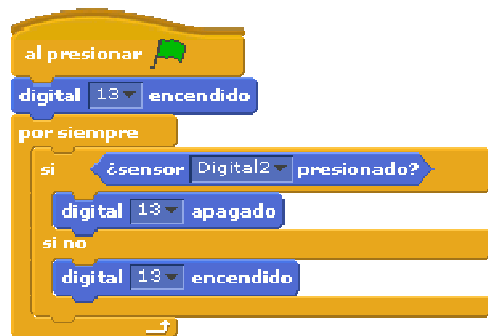
El esquema eléctrico de la práctica es el siguiente:



El esquema de conexionado y montaje:

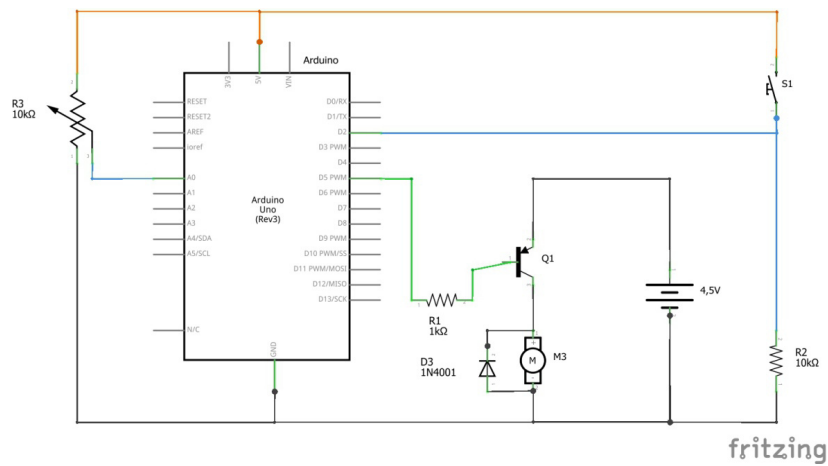


Un ejemplo de programación en S4A:

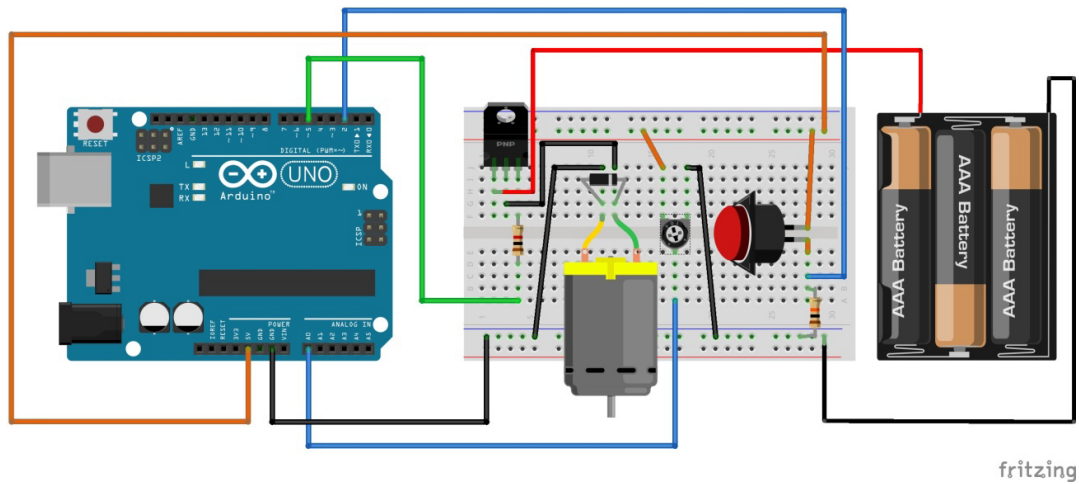


### 61.Regulación y control de la velocidad y del encendido y apagado de un motor, de forma indirecta, a través de un transistor PNP BD138.

El esquema eléctrico de la práctica es el siguiente:



El esquema de conexionado y montaje:



Un ejemplo de programación en S4A:

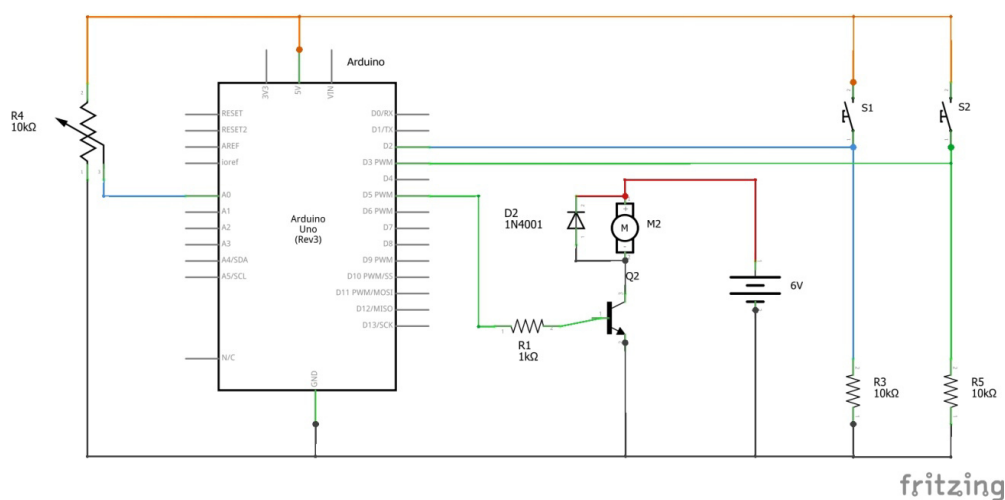


Hay que recordar a los alumnos que se debe de realizar una pequeña corrección en la entrada analógica de la resistencia variable con la que seleccionamos la velocidad deseada, ya que sin esa corrección, velocidad y selección de velocidad se comportan de forma opuesta.

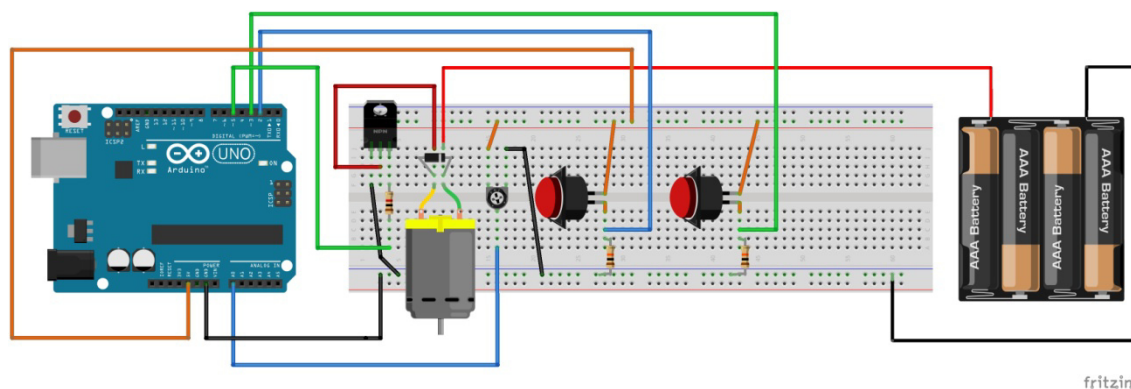
## 62. Marcha-paro y control de la velocidad de un motor, de forma indirecta, a través de un transistor NPN BD135.

A partir de esta práctica y hasta la práctica 67 vamos a trabajar los diferentes circuitos de mando o control que se utilizan en la puesta en marcha y parada de motores. Hay que resaltar que en todos ellos el motor va a ser controlado de forma indirecta por un transistor NPN BD135. El profesor puede decidir si quiere que los alumnos desmonten la práctica completamente y vuelvan a montarla (si todavía hay dudas sobre el funcionamiento, montaje y conexionado del transistor) o prefiere ir modificando esta practica 62 como montaje base. También se pueden plantear todas ellas con un transistor PNP BD138 (ver prácticas 60 y 61).

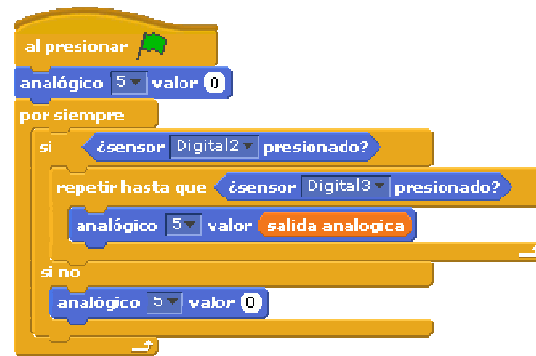
El esquema eléctrico de la práctica es el siguiente:



El esquema de conexionado y montaje:

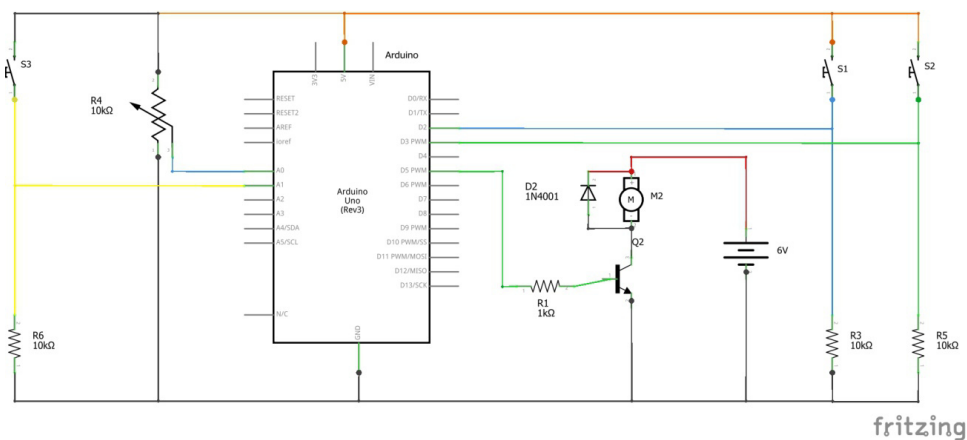


Un ejemplo de programación en S4A:

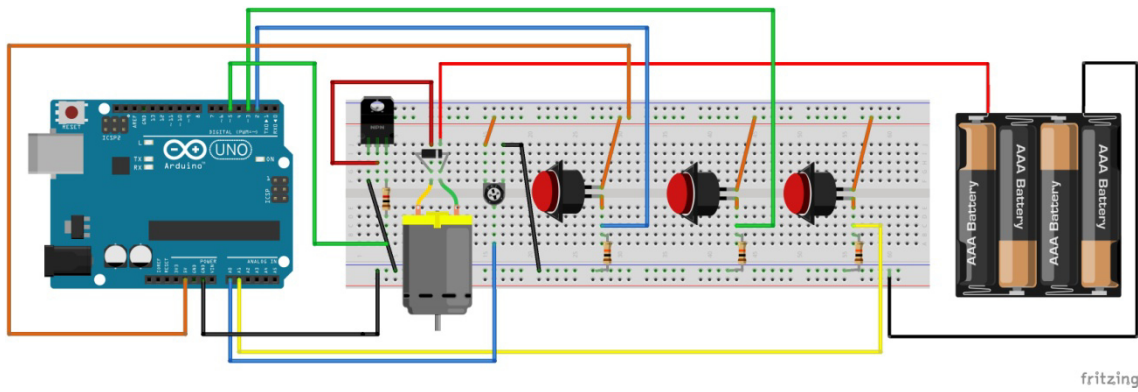


**63. Marcha-paro con varios pulsadores y función OR, control de la velocidad de un motor, de forma indirecta, a través de un transistor NPN BD135.**

El esquema eléctrico de la práctica es el siguiente:



El esquema de conexionado y montaje:



Un ejemplo de programación en S4A para dos pulsadores de paro y función OR:

```

al presionar
por siempre
  fijar velocidad deseada a valor del sensor Analog0
  fijar salida analogica a redondear velocidad deseada * 255 / 1023

```

```

al presionar
analógico 5 valor 0
por siempre
  si ¿sensor Digital2 presionado?
  repetir hasta que ¿sensor Digital3 presionado? o valor del sensor Analog1 > 1000
  analógico 5 valor salida analogica
  si no
  analógico 5 valor 0

```

Un ejemplo de programación en S4A para dos pulsadores de marcha y función OR:

```

al presionar
por siempre
  fijar velocidad deseada a valor del sensor Analog0
  fijar salida analogica a redondear velocidad deseada * 255 / 1023

```

```

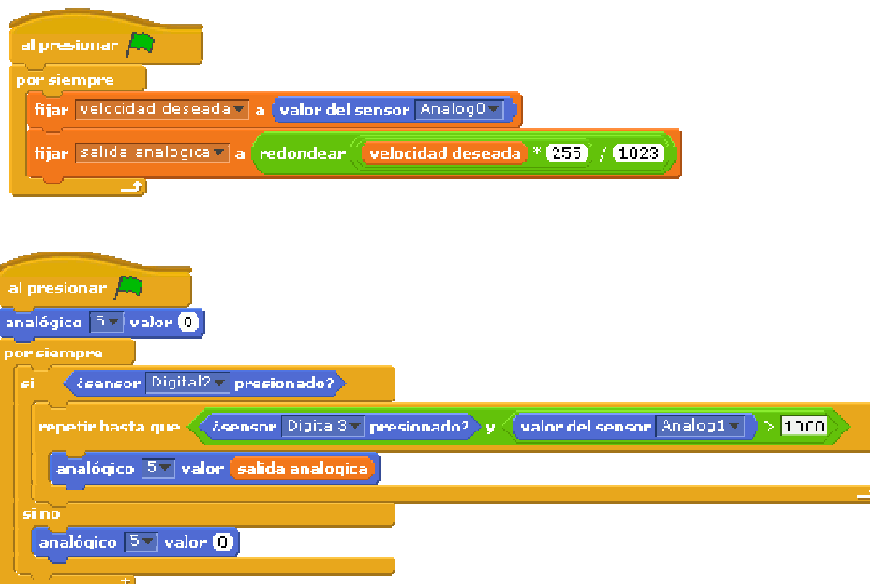
al presionar
analógico 5 valor 0
por siempre
  si ¿sensor Digital3 presionado? o ¿sensor Digital2 presionado?
  repetir hasta que valor del sensor Analog1 > 1000
  analógico 5 valor salida analogica
  si no
  analógico 5 valor 0

```

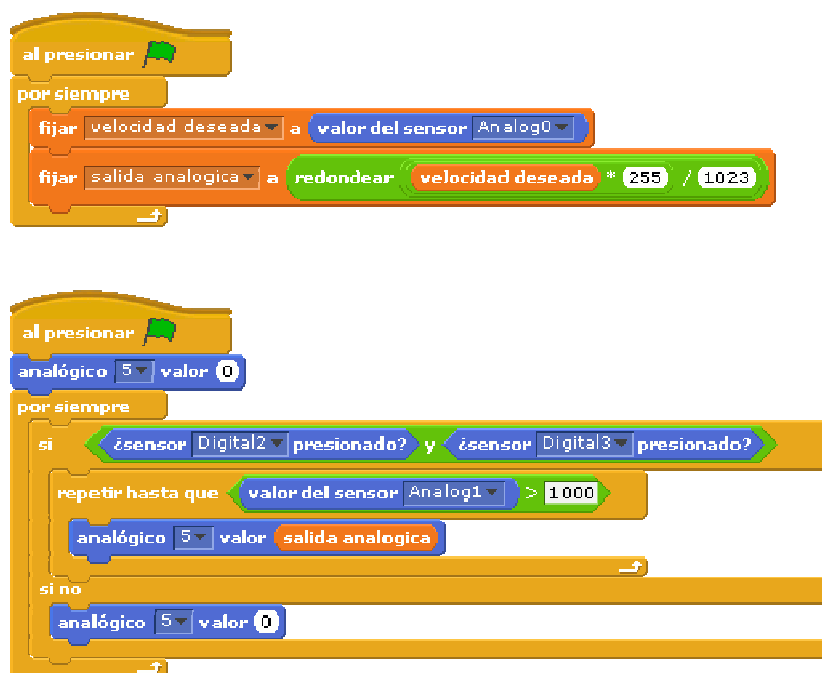
#### 64. Marcha-paro con varios pulsadores y función AND, control de la velocidad de un motor, de forma indirecta, a través de un transistor NPN BD135.

Los esquemas eléctricos, de conexionado y montaje son los mismos que los de la práctica anterior.

Un ejemplo de programación en S4A para dos pulsadores de paro y función AND:



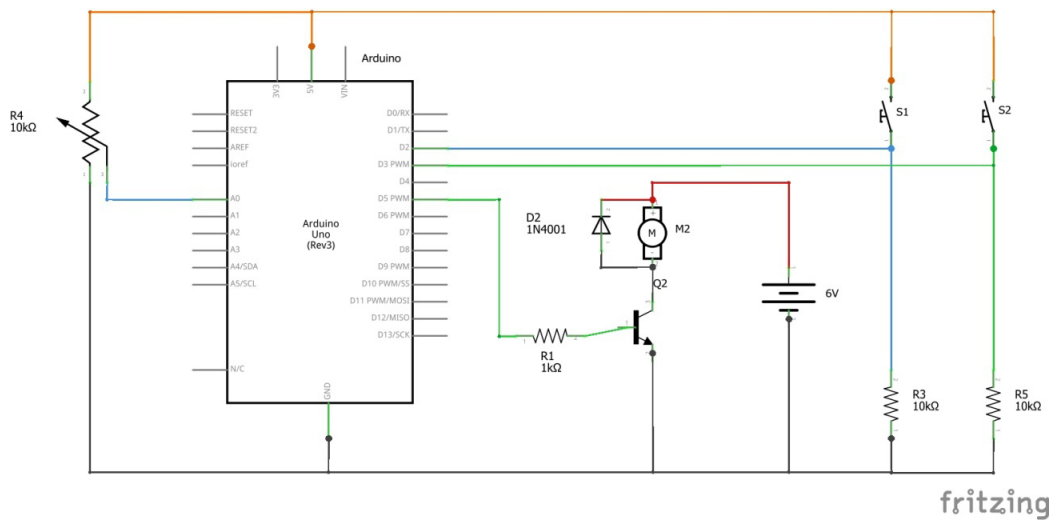
Un ejemplo de programación en S4A para dos pulsadores de marcha y función AND:



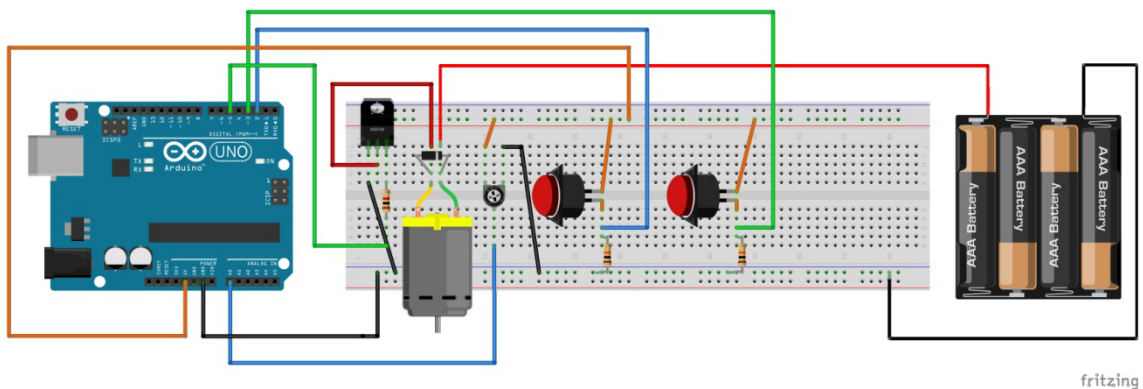


**65. Marcha-paro con prioridad al paro y control de la velocidad de un motor, de forma indirecta, a través de un transistor NPN BD135.**

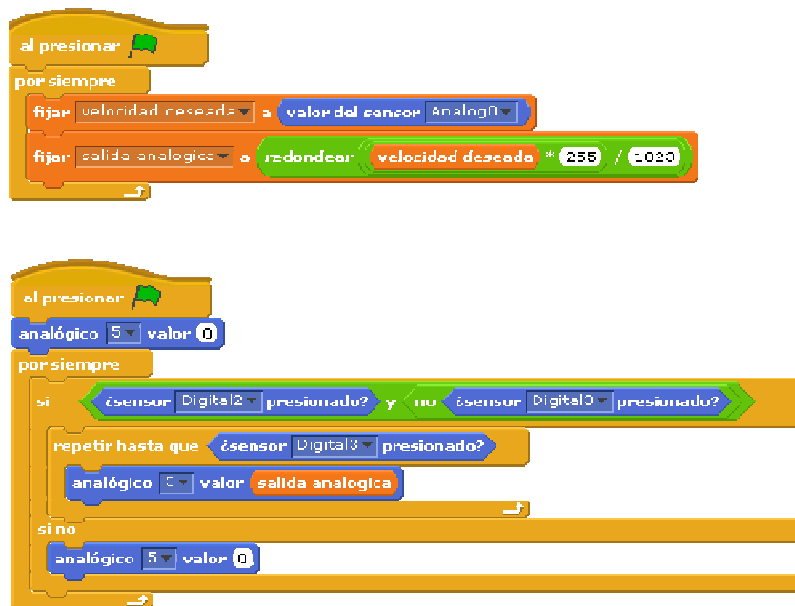
El esquema eléctrico de la práctica es el siguiente:



El esquema de conexionado y montaje:



Un ejemplo de programación en S4A:

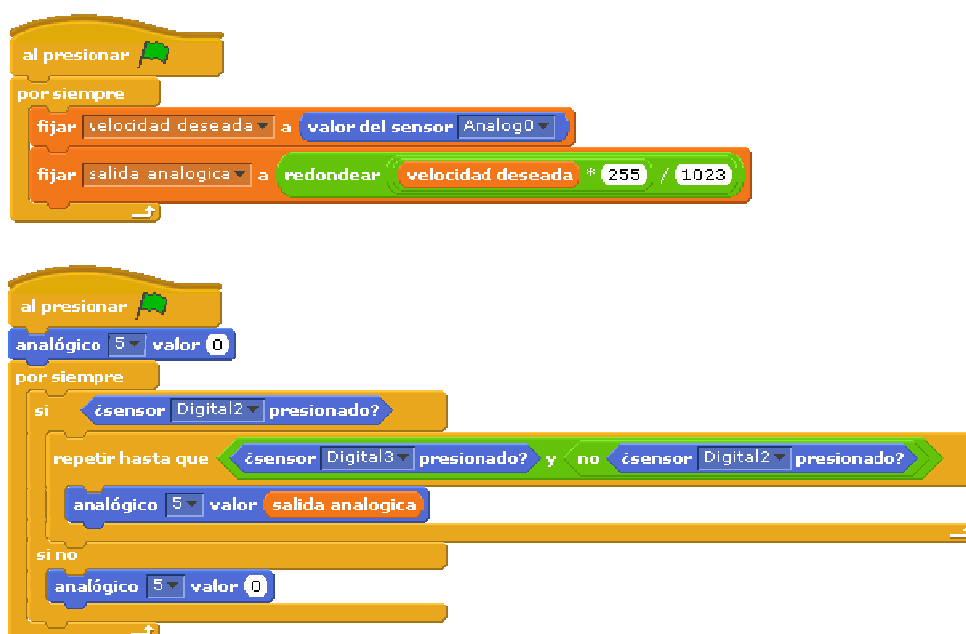


Se podría proponer a los alumnos el mismo ejercicio con varios pulsadores de marcha y/o varios pulsadores de paro y que cumplieran la condición de prioridad al paro. Para añadir más pulsadores, se utilizarían las entradas analógicas tal y como hemos visto en prácticas anteriores, tanto para el montaje y conexionado, como para la programación.

### 66. Marcha-paro con prioridad a la marcha y control de la velocidad de un motor, de forma indirecta, a través de un transistor NPN BD135.

Los esquemas eléctricos, de conexionado y montaje son los mismos que los de la práctica anterior.

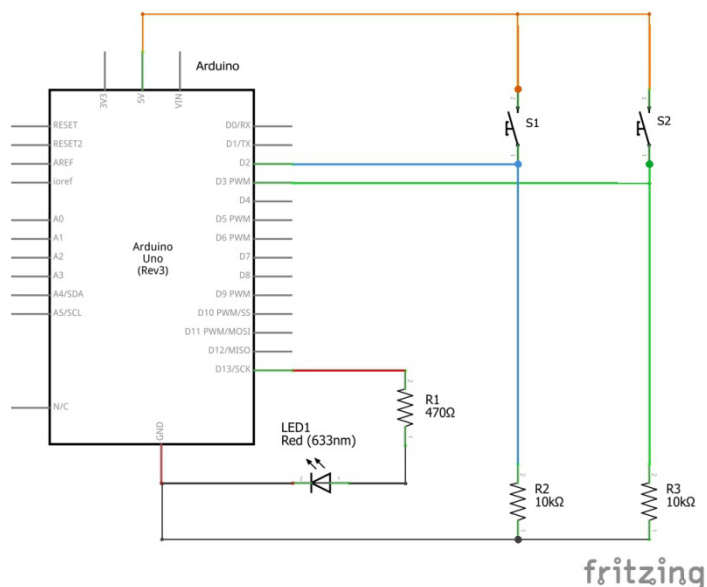
Un ejemplo de programación en S4A:



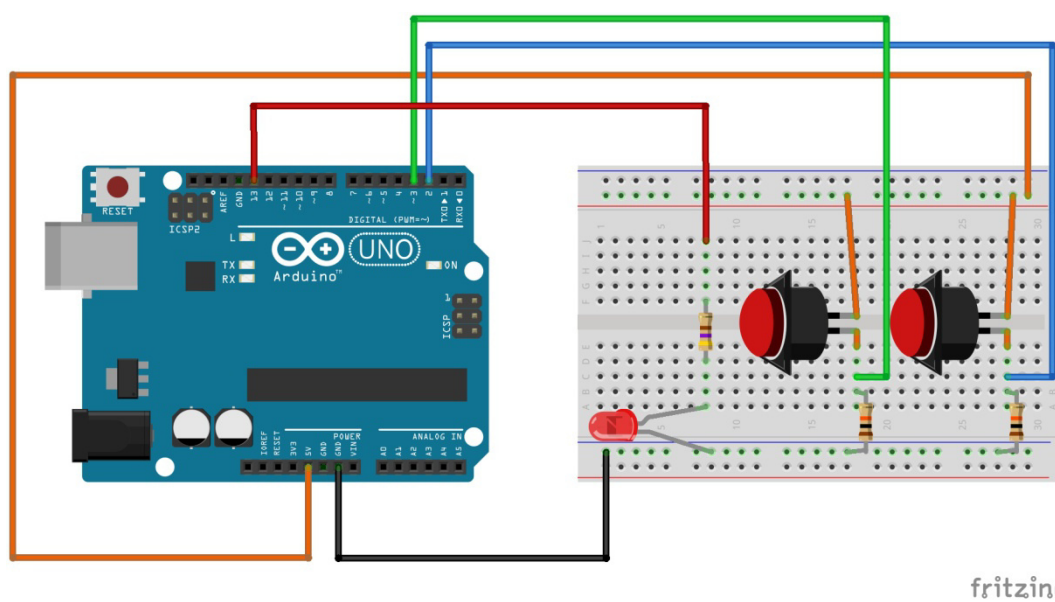
Hay que recordar que la lógica que se suele utilizar y que nos interesa es la de prioridad al paro. Por motivos de seguridad, si hay un elemento de paro accionado (pulsador, final de carrera, interruptor reed, ...), nunca debe de poder ponerse en marcha ningún receptor que produzca movimiento (motor eléctrico, cilindro neumático o hidráulico, motor neumático o hidráulico, ...).

## 67. Marcha paro con prioridad al paro de un diodo led, desde pulsadores y pulsadores virtuales en pantalla, con monitorización del estado del diodo led.

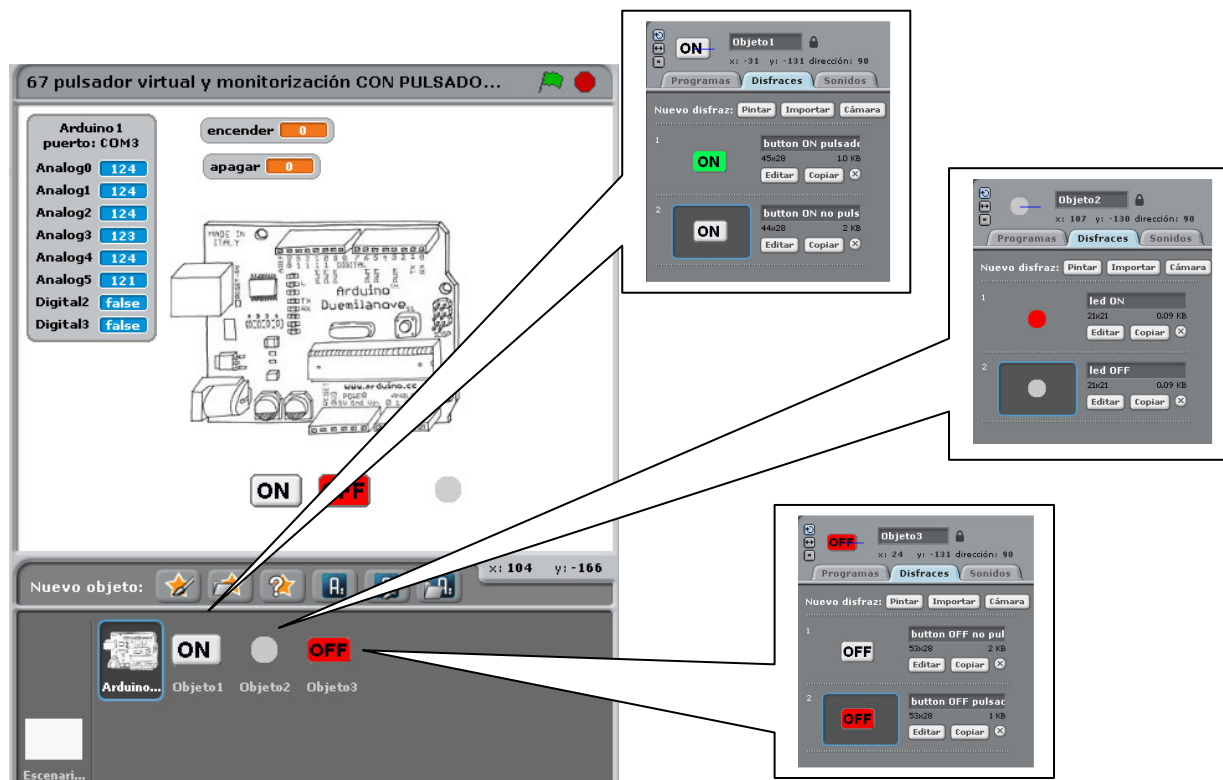
El esquema eléctrico de la práctica es el siguiente:



El esquema de conexionado y montaje:



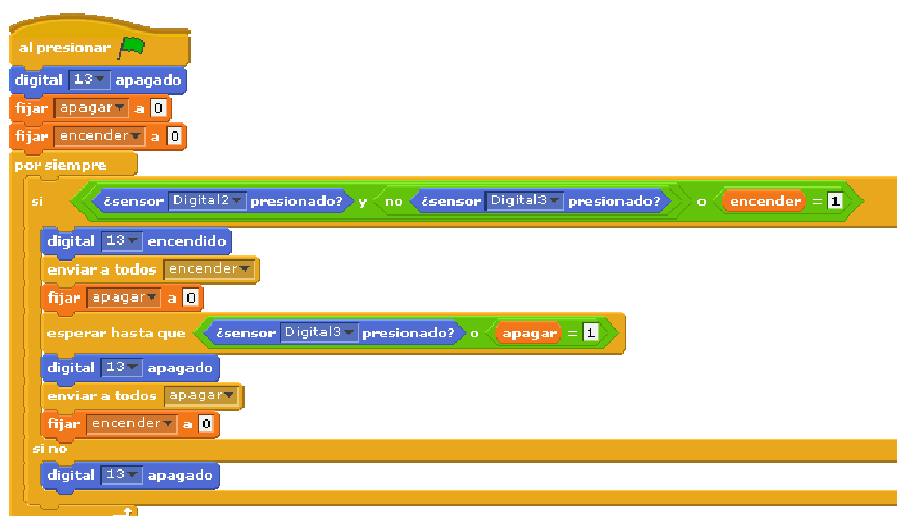
Para la confección de los pulsadores virtuales en pantalla, en este caso pulsador de marcha y pulsador de paro, necesitaremos dos objetos, uno para cada pulsador y otro objeto para la monitorización del diodo led. En la siguiente imagen podemos observar los diferentes objetos y sus disfraces:



En la programación en S4A como bloques no vistos en las prácticas anteriores, se utilizará el bloque perteneciente a la librería de control “al presionar Objeto \_\_” para iniciar una serie de acciones al presionar sobre el correspondiente objeto en pantalla. Este bloque aparece solo en la programación del correspondiente objeto, es decir, si vamos a programar el objeto 1 nos aparece el bloque “al presionar Objeto 1” y si vamos a programar el objeto 3 nos aparecerá el bloque “al presionar Objeto 3”.

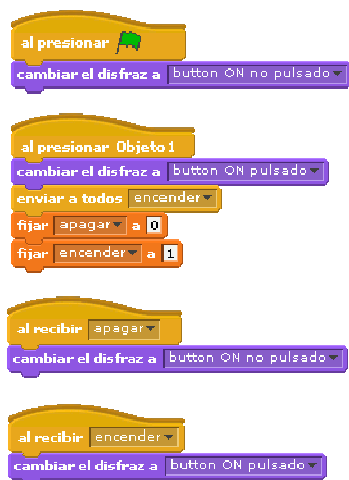


Un ejemplo de programación en S4A para el bloque Arduino 1:

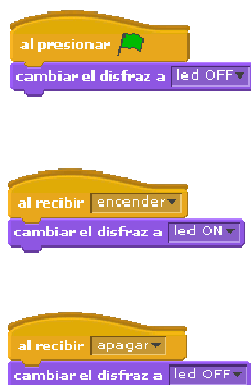


Y para el resto de los objetos:

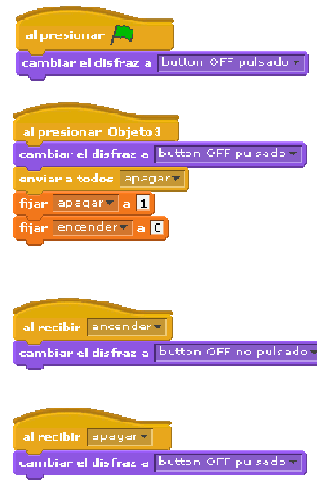
Objeto 1 (pulsador de marcha)



Objeto 2 (diodo led)



Objeto 3 (pulsador de paro)

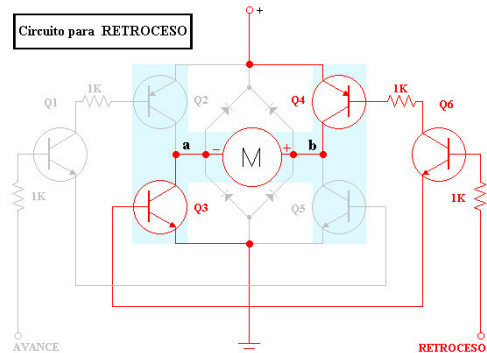
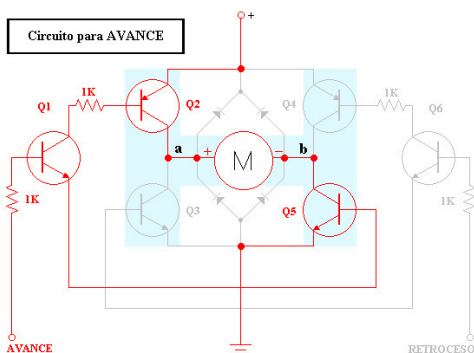
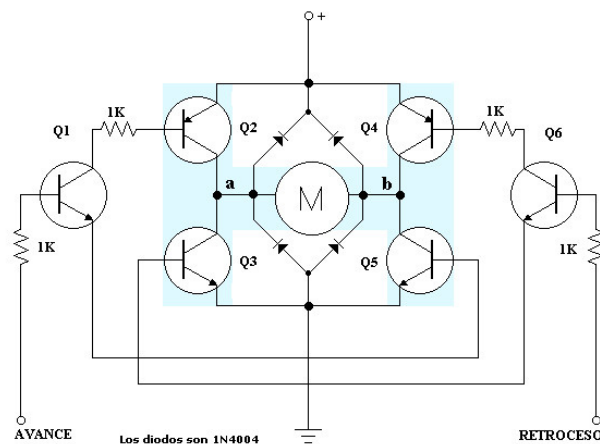


Una vez probado, hay que hacer ver a los alumnos que tenemos un pequeño problema y es que cuando uno de los dos pulsadores virtuales lo tenemos pulsado, no debemos de tener la opción de pulsarlo, ya que el motor o ya está en marcha (pulsador ON) o ya está parado (pulsador OFF). En pantalla nos aparece el correspondiente disfraz en color gris, lo que puede llevar a engaño al operador que trabaja desde el ordenador. Esto lo podemos solucionar haciendo que los disfraces del pulsador virtual cuando no está pulsado, o no tiene que haber la opción de pulsarlo, del mismo color que el fondo del escenario o transparentes. En la siguiente imagen podemos ver las modificaciones:

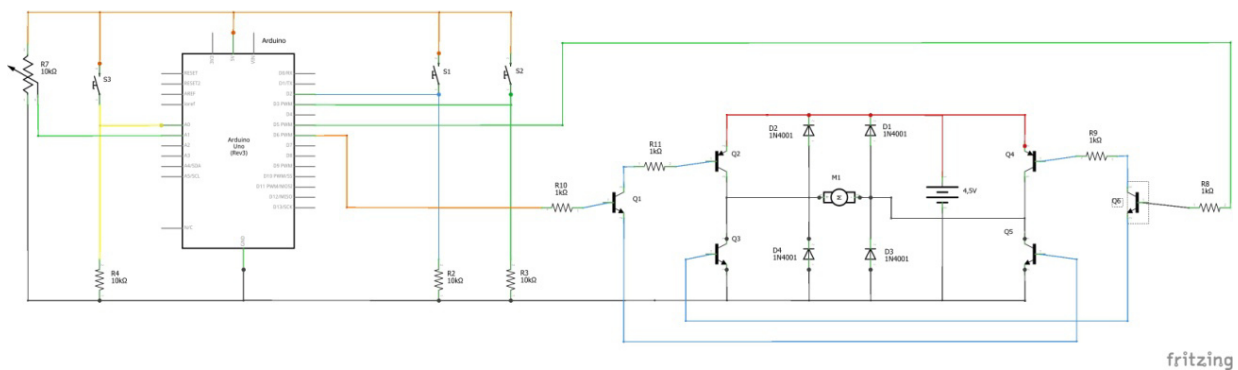


**68. Control total (encendido, apagado, sentido de giro y velocidad) de un motor mediante un puente H con transistores PNP BC138 y NPN BC135, controlados por otros dos transistores NPN BC547.**

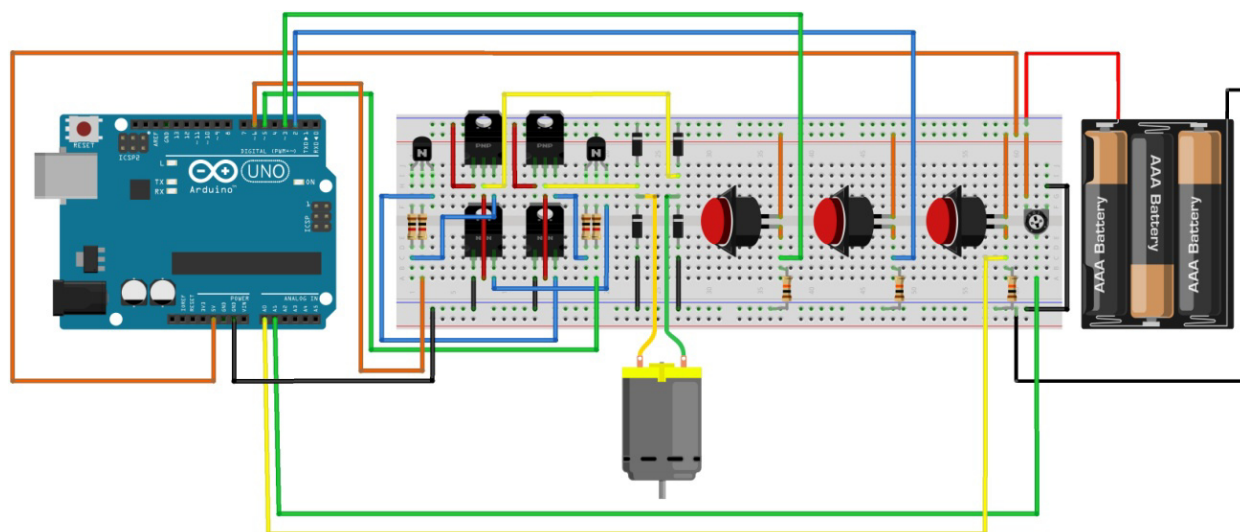
En primer lugar habrá que hacer una explicación detallada del esquema eléctrico correspondiente y de su funcionamiento. Hay que recordar a los alumnos que para cambiar el sentido de giro de un motor de corriente continua hay que invertir el sentido de la corriente que circula a través de sus bobinas. También habrá que hacer especial hincapié en la función de los diodos de protección o snubber cuando se trabaja con elementos inductivos (bobinas). Para ello el profesor se podrá ayudar de los siguientes esquemas obtenidos de la web [http://robots-argentina.com.ar/MotorCC\\_PuenteH.htm](http://robots-argentina.com.ar/MotorCC_PuenteH.htm)<sup>19</sup>:



El esquema eléctrico de la práctica es el siguiente:

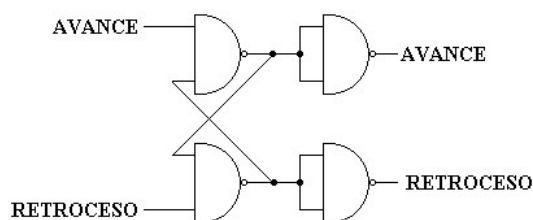


El esquema de conexionado y montaje:



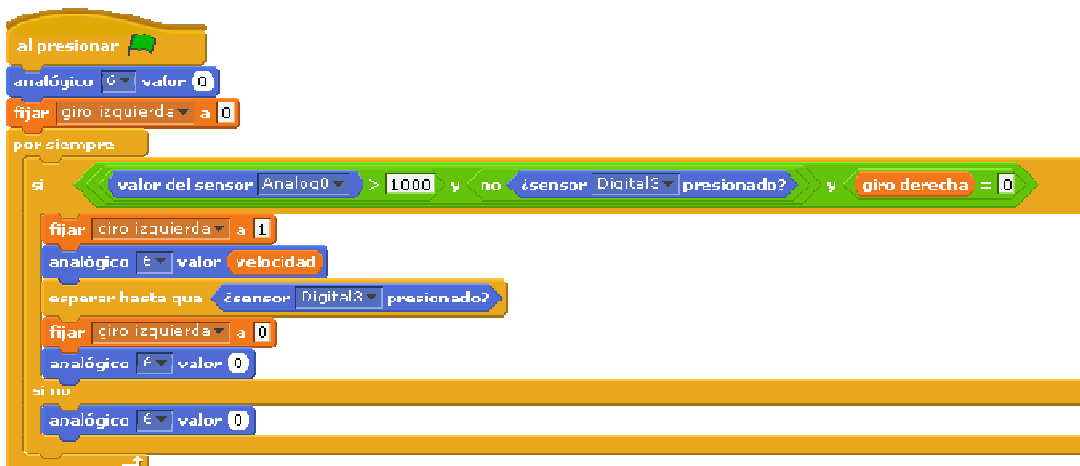
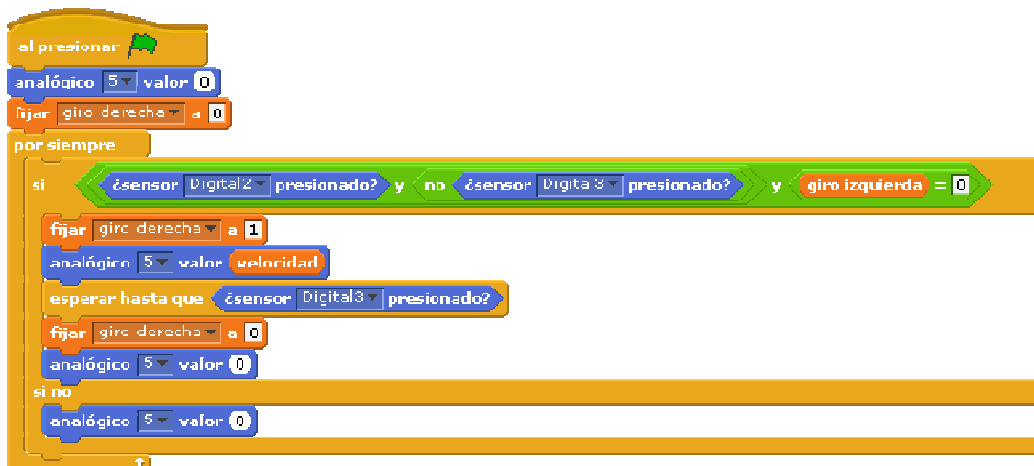
fritzing

Hay que hacer especial énfasis en que al puente H nunca le debe de llegar de forma simultánea alimentación de avance (giro sentido horario pulsador S1) y de retroceso (giro sentido antihorario pulsador S3) ya que esto originaría un importante cortocircuito, dañando como mínimo y de forma permanente los 4 transistores que forman el puente H. Para evitar esto se puede complementar la práctica con electrónica digital mediante el siguiente circuito de puertas lógicas:



Hay que tener en cuenta que si usamos este circuito perdemos la capacidad de regular la velocidad del motor, ya que las salidas de avance y retroceso del circuito de puertas lógicas son digitales, es decir, todo (5V) o nada (0V). Por lo tanto, si queremos poder regular la velocidad del motor en ambos sentidos de giro, deberemos de hacer el enclavamiento mediante la programación en S4A.

Un ejemplo de programación en S4A:

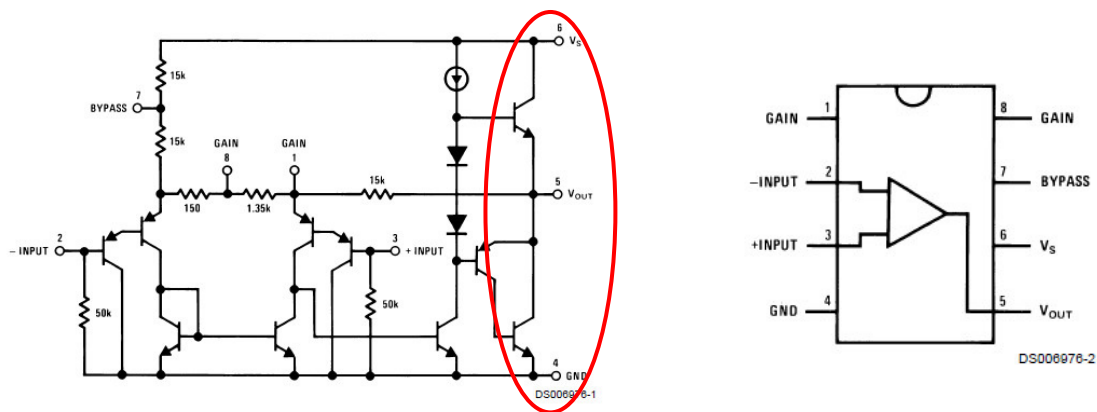


Hay que hacer ver a los alumnos que con esta programación en S4A podemos ajustar la velocidad del motor en ambos sentidos de forma constante, durante la marcha, ya que el programa, de forma independiente en un bucle aparte, fija la variable velocidad.

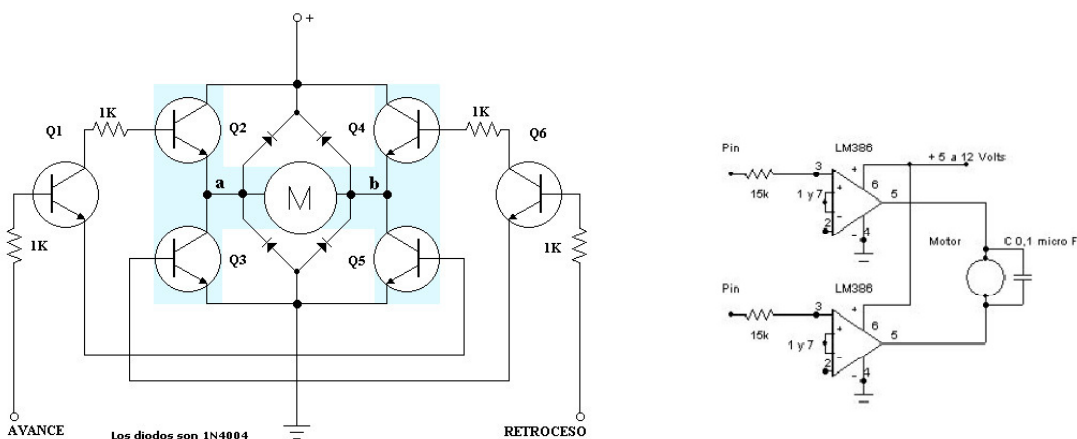
### 69. Control total (encendido, apagado, sentido de giro y velocidad) de un motor mediante un puente H confeccionado con dos amplificadores operacionales LM386.

En primer lugar es necesario conocer el esquema de patillaje y el esquema equivalente de todos los elementos que componen el amplificador operacional LM386<sup>18</sup> y que podemos ver a continuación:



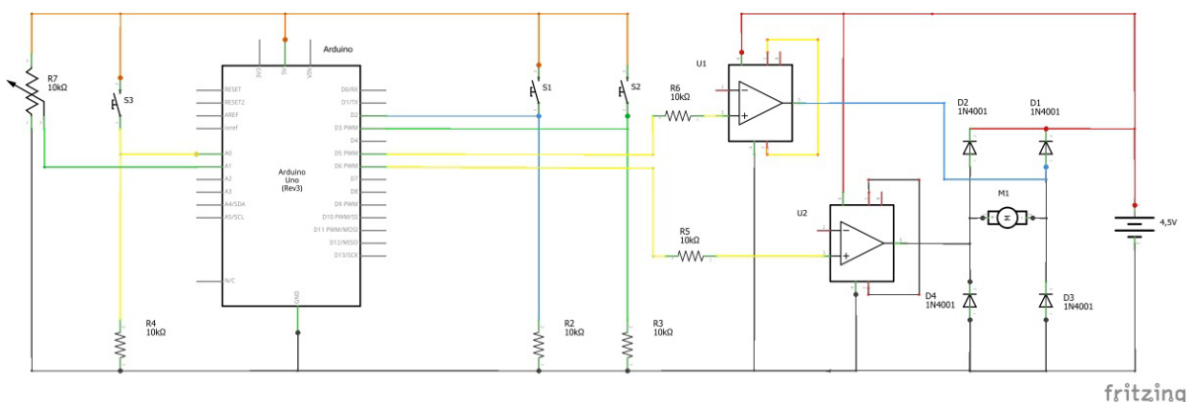


Hay que hacer ver a los alumnos que con el LM386 tenemos un semipunto H con dos transistores NPN (parte derecha del esquema equivalente de la figura anterior y señalada en rojo). Podemos encontrar información en <http://micropinguino.blogspot.com.es/2011/04/puente-h-con-el-lm386.html><sup>20</sup>. A continuación podemos ver el esquema de la adaptación de la práctica anterior para poder comprender el funcionamiento de circuito y realizar su montaje:

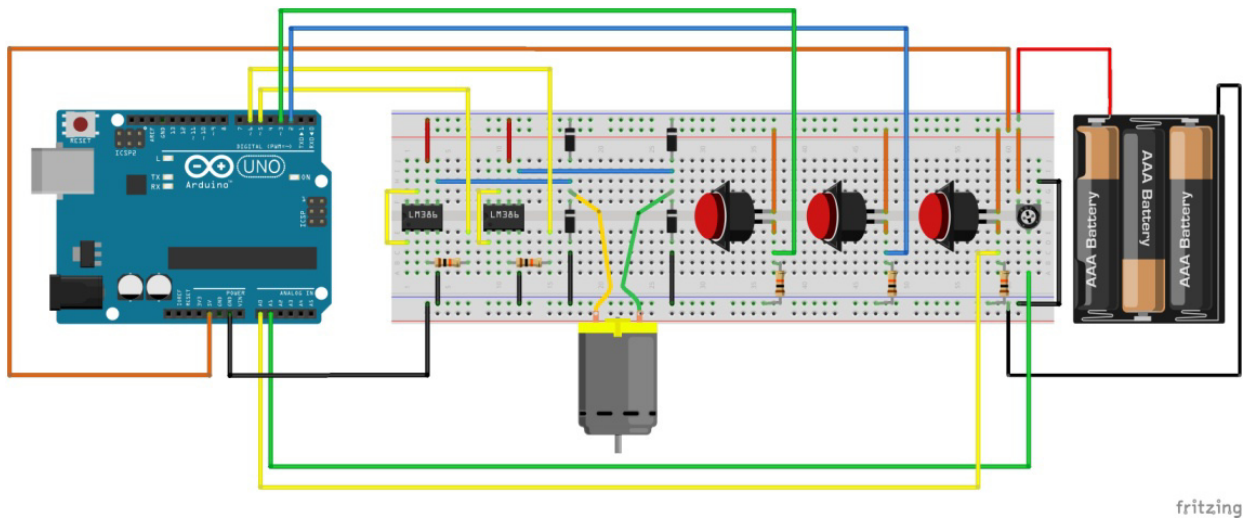


Necesitaremos dos amplificadores operacionales LM386 para poder construir el puente H de 4 transistores NPN.

El esquema eléctrico de la práctica es el siguiente:



El esquema de conexionado y montaje:



Un ejemplo de programación en S4A:

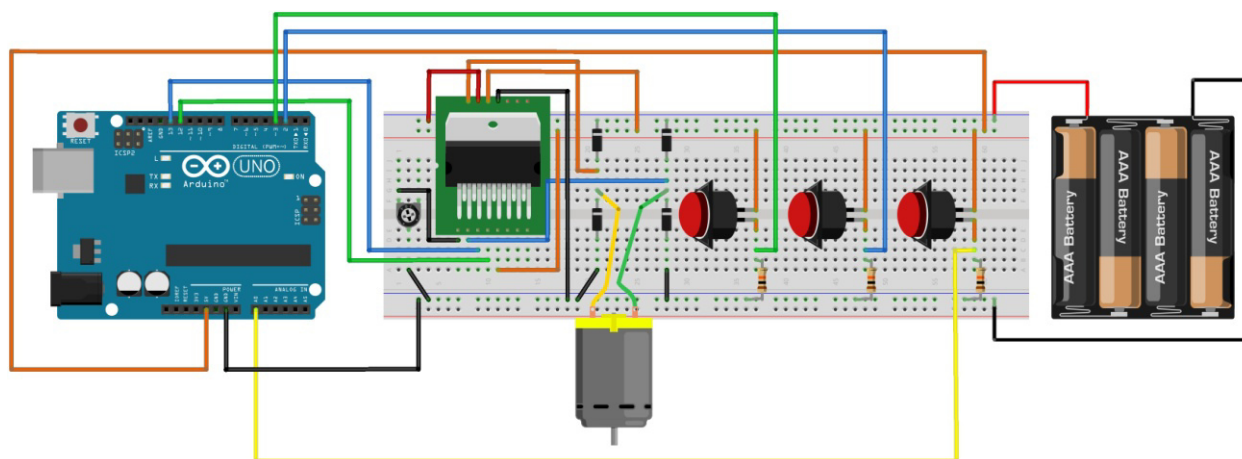
```

al presionar
por siempre
  fijar velocidad a redondear valor del sensor Analog1 * 255 / 1023

al presionar
analógico 5 valor 0
fijar giro derecha a 0
por siempre
  si ¿sensor Digital2 presionado? y no ¿sensor Digital3 presionado? y giro izquierda = 0
    fijar giro derecha a 1
    analógico 5 valor velocidad
    esperar hasta que ¿sensor Digital3 presionado?
    fijar giro derecha a 0
    analógico 5 valor 0
  si no
    analógico 5 valor 0

al presionar
analógico 6 valor 0
fijar giro izquierda a 0
por siempre
  si valor del sensor Analog0 > 1000 y no ¿sensor Digital3 presionado? y giro derecha = 0
    fijar giro izquierda a 1
    analógico 6 valor velocidad
    esperar hasta que ¿sensor Digital3 presionado?
    fijar giro izquierda a 0
    analógico 6 valor 0
  si no
    analógico 6 valor 0
    
```





fritzing

Un ejemplo de programación en S4A:

```

al presionar
digital 13 apagado
fijar giro derecha a 0
por siempre
si << sensor Digital 2 presionado? y no << sensor Digital 3 presionado? y giro izquierda = 0
    fijar giro derecha a 1
    digital 13 encendido
    esperar hasta que << sensor Digital 3 presionado?
    fijar giro derecha a 0
    digital 13 apagado
si no
    digital 13 apagado
    
```

```

al presionar
digital 12 apagado
fijar giro izquierda a 0
por siempre
si << valor del sensor Analógico > 1000 y no << sensor Digital 0 presionado? y giro derecha = 0
    fijar giro izquierda a 1
    digital 12 encendido
    esperar hasta que << sensor Digital 0 presionado?
    fijar giro izquierda a 0
    digital 12 apagado
si no
    digital 12 apagado
    
```

### 71. Control de un motor miniservo (0º-180º) mediante una resistencia variable.

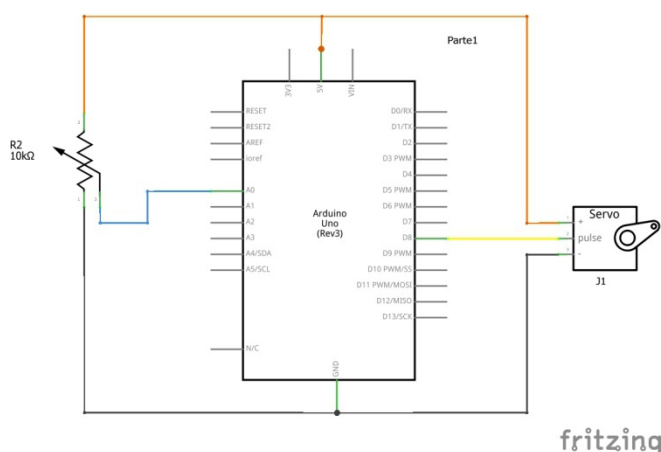
El primer paso es entender el funcionamiento de un servomotor. En este caso, a diferencia del resto del libro, en el que hemos utilizado el libro digital de tecno12-18.com <sup>12</sup>, al carecer de la explicación de este motor, vamos a recurrir a la página <http://www.areatecnologia.com/electricidad/servomotor.html> <sup>13</sup> y a un vídeo de youtube

sobre la <https://www.youtube.com/watch?v=84mxq41zdwE><sup>14</sup>, con lo que los alumnos pueden trabajar de forma autónoma y entender perfectamente el funcionamiento y principios de control de este tipo de motores.

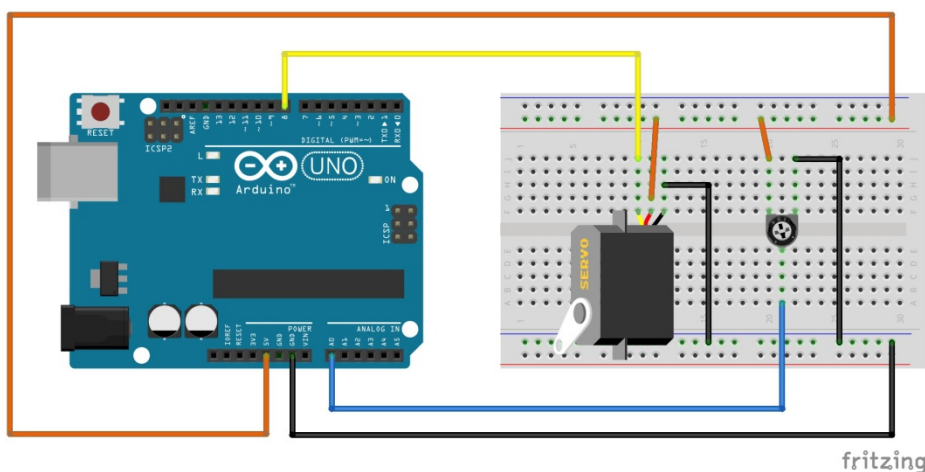
De cara al conexionado del motor, hay que tener en cuenta que no se han estandarizado los colores del cableado. En la siguiente tabla podemos ver un resumen en el que se muestran los colores del cableado de varios de los principales fabricantes de servomotores:

R/C Servo Wire Color Code			
Servo	Positive (+)	Signal (S)	Negative (-)
Futaba - J	Red	White	Black
JR	Red	Orange	Brown
Hitec	Red	Yellow	Black
Airtronics	Red	White	Black
	Red	Black	Black
Airtronics - Z	Red	Blue	Black
Fleet	Red	White	Black
KO	Red	White	Black

El esquema eléctrico de la práctica es el siguiente:



El esquema de conexionado y montaje:



A continuación se realizará la programación en S4A, del funcionamiento de la práctica. Para ello, como bloque no visto en las prácticas anteriores, se utilizará el bloque “motor \_\_ ángulo \_\_” para controlar mediante PWM el giro del servomotor. Este bloque podemos encontrarlo dentro de la librería movimiento.



Hay que hacer notar, que la única salida específica para controlar un servomotor es la salida 8. Podríamos introducir por teclado el ángulo de giro que deseamos o como es nuestro caso, calcularlo a partir de la señal de una resistencia variable.

En primer lugar y después de la explicación anterior, se puede dejar a los alumnos para que hagan un programa en el que modifiquen mediante teclado el ángulo del servomotor.

Posteriormente podemos pasar directamente a controlar el servomotor mediante una resistencia variable. Los cálculos a realizar serían los siguientes:

$$\left. \begin{array}{l} 1023 \text{ ————— } 180^\circ \\ A0 \text{ ————— } x \end{array} \right\} x = \frac{A0 \cdot 180^\circ}{1023}$$

Un ejemplo de programación en S4A:



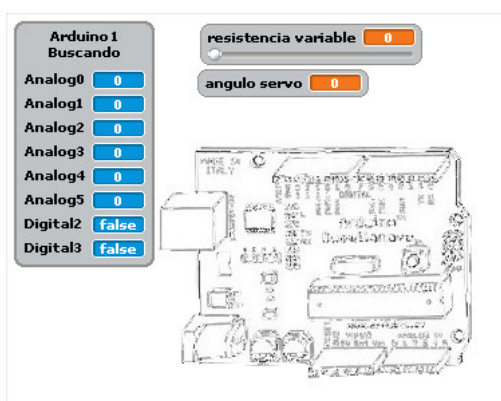
La práctica se podría completar utilizando la resistencia variable con deslizador en pantalla. Si elegimos un deslizador entre 0 y 1023, un posible ejemplo de programación en S4A:



Si el deslizador elegido es porcentual, un posible ejemplo de programación en S4A:



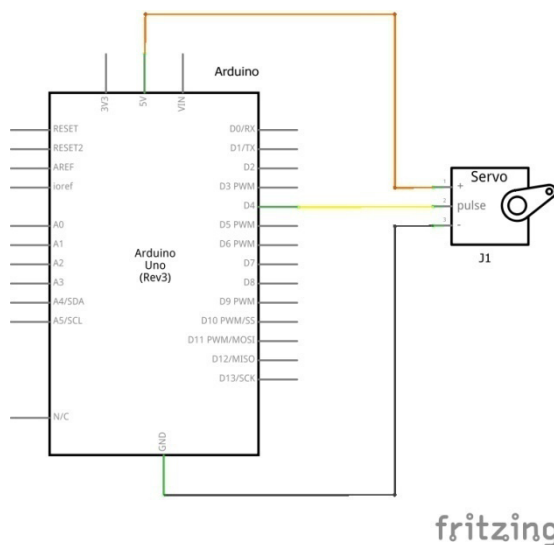
En la siguiente imagen podemos ver una muestra del escenario para los dos últimos casos:



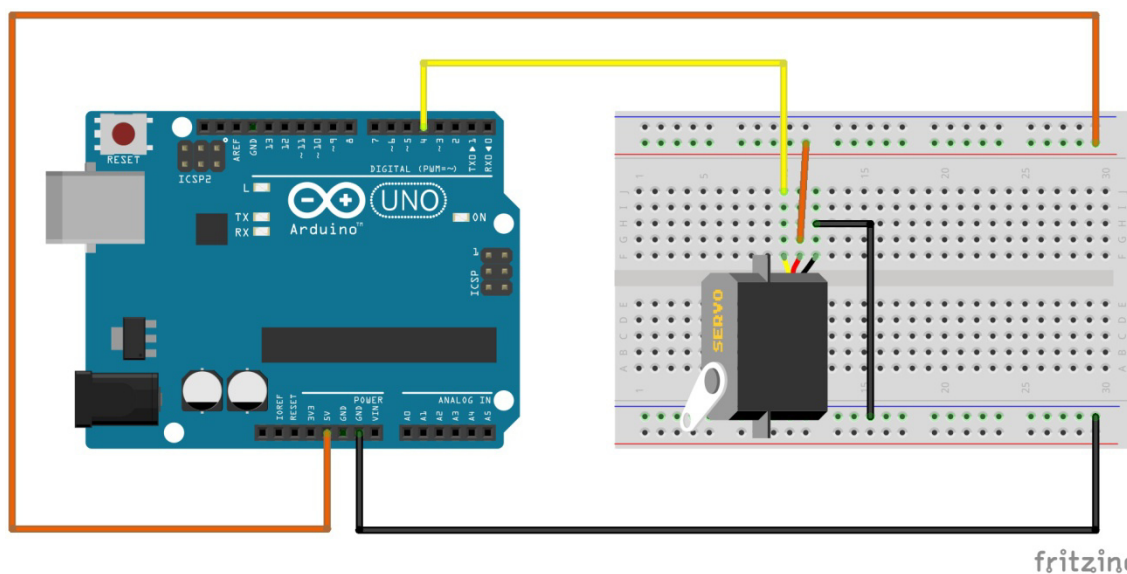
### 72. Control de un servo de rotación continua.

Como continuación de la práctica anterior, un servomotor de rotación continua funciona y se controla exactamente igual que el miniservo de 180º, con la única diferencia que puede girar 360º de forma indefinida.

El esquema eléctrico de la práctica es el siguiente:



El esquema de conexionado y montaje:



A continuación se realizará la programación en S4A, del funcionamiento de la práctica. Para ello, como bloques no vistos en las prácticas anteriores, se utilizarán el bloque “motor \_\_ dirección \_\_” para elegir la salida PWM (4 ó 7) con la que controlar el giro del servomotor y el sentido en el que queremos que gire el motor (horario o antihorario) y el bloque “motor \_\_ apagado” para poner a cero la salida elegida. Estos bloques podemos encontrarlos dentro de la librería movimiento.



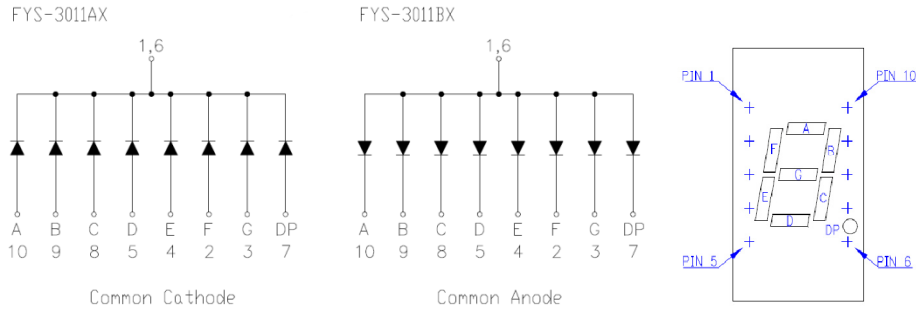
Un ejemplo de programación en S4A de forma que hagamos girar el motor durante 4 segundos en sentido horario, lo paremos durante un segundo, lo hagamos girar durante 4 segundos en sentido antihorario, lo volvamos a parar durante 1 segundo y repitamos el ciclo de forma indefinida:





### 73. Cuenta atrás en un display numérico de 7 segmentos.

Para esta práctica vamos a usar un nuevo elemento, el circuito integrado FYS-3011A, un display de 7 segmentos de cátodo común. Aprovecharemos para repasar las diferencias entre ánodo común y cátodo común, ayudándonos de la siguiente imagen<sup>18</sup>:

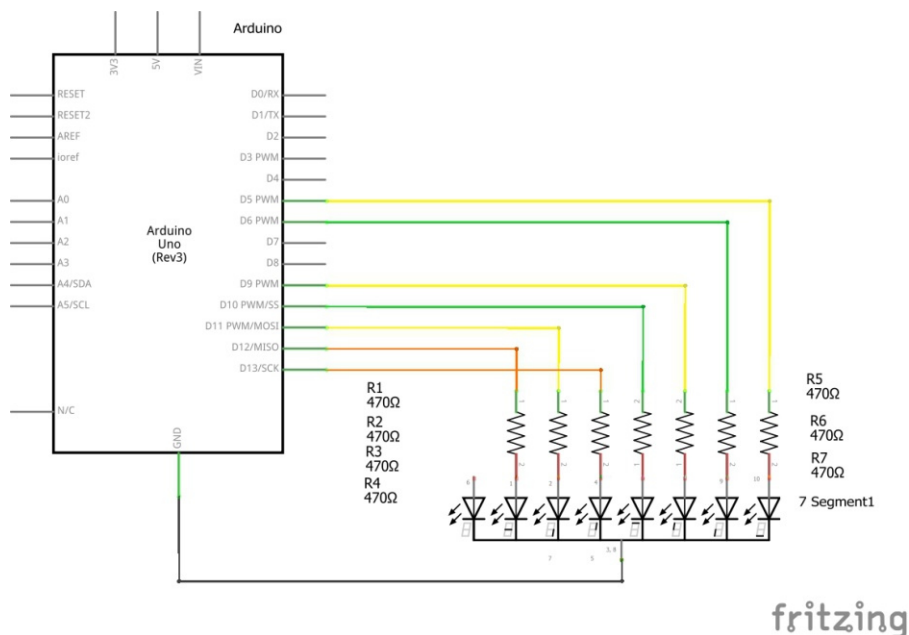


Necesitaremos también el esquema de patillaje del display <sup>18</sup> que podemos ver en la imagen anterior.

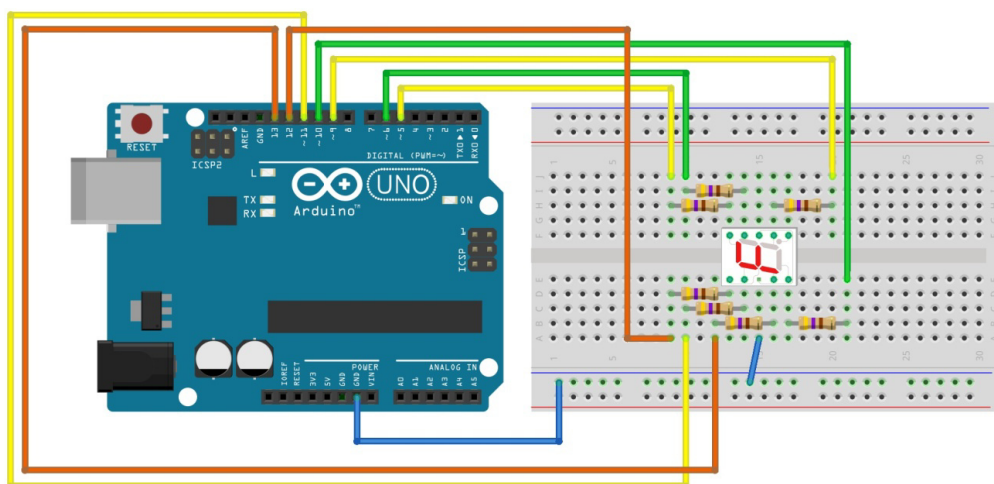
En la siguiente tabla se muestra la asignación de las salidas de Arduino que se van a usar para controlar los diferentes segmentos del display:

SEGMENTO	SALIDA ARDUINO	PIN
A	5	10
B	6	9
C	9	8
D	10	5
E	11	4
F	12	2
G	13	3

El esquema eléctrico de la práctica es el siguiente:

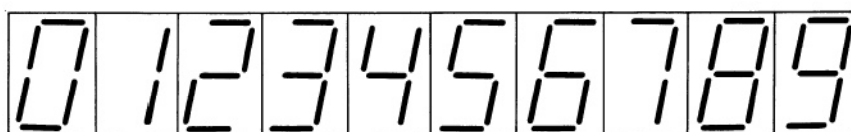


El esquema de conexionado y montaje:



fritzing

Los alumnos deberán diseñar un programa con S4A para controlar una cuenta atrás de 1 segundo entre cada dígito de la cuenta, partiendo de 9 y llegando a 0, de forma que se visualice la cuenta en el LCD de 7 segmentos. Cuando llegue a cero la cuenta debe de reiniciarse (repetirse de forma indefinida). Para ayudar en la programación y evitar errores, sería muy interesante que los alumnos completaran la siguiente tabla ayudados por la imagen con la representación de los diferentes dígitos del sistema decimal en código de 7 segmentos:



NÚMERO DECIMAL	SEGMENTOS ACTIVOS	SALIDAS A ACTIVAR RESPECTO AL ANTERIOR	SALIDAS A DESACTIVAR RESPECTO AL ANTERIOR
9	A, B, C, D, F, G	5, 6, 9, 10, 12, 13	11
8	A, B, C, D, E, F, G	11	
7	A, B, C		10, 11, 12, 13
6	A, C, D, E, F, G	10, 11, 12, 13	6
5	A, C, D, F, G		11
4	B, C, F, G	6	5, 10
3	A, B, C, D, G	5, 10	12
2	A, B, D, E, G	11	9
1	B,C	9	5, 10, 11, 13
0	A, B, C, D, E, F	5, 10, 11, 12	

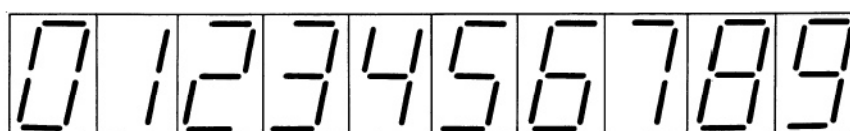
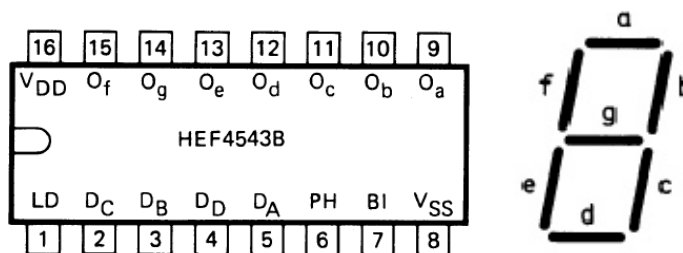
El programa en S4A es el siguiente:

```

al presionar [bandera]
analógico 5 valor 0
analógico 6 valor 0
analógico 9 valor 0
digital 10 apagado
digital 11 apagado
digital 12 apagado
digital 13 apagado
esperar 1 segundos
por siempre
analógico 5 valor 255
analógico 6 valor 255
analógico 9 valor 255
digital 10 encendido
digital 12 encendido
digital 13 encendido
esperar 1 segundos
digital 11 encendido
esperar 1 segundos
digital 10 apagado
digital 11 apagado
digital 12 apagado
digital 13 apagado
esperar 1 segundos
analógico 6 valor 0
digital 10 encendido
digital 11 encendido
digital 12 encendido
digital 13 encendido
esperar 1 segundos
digital 11 apagado
esperar 1 segundos
analógico 5 valor 0
digital 10 apagado
analógico 6 valor 255
esperar 1 segundos
digital 12 apagado
analógico 5 valor 255
digital 10 encendido
esperar 1 segundos
analógico 9 valor 0
digital 11 encendido
esperar 1 segundos
analógico 5 valor 0
digital 10 apagado
digital 11 apagado
digital 13 apagado
analógico 9 valor 255
esperar 1 segundos
analógico 5 valor 255
digital 10 encendido
digital 11 encendido
digital 12 encendido
esperar 1 segundos
digital 11 apagado
    
```

### 74. Cuenta adelante programada en BCD, decodificada y mostrada en un display de 7 segmentos.

En esta práctica vamos a utilizar el circuito integrado 4511, un decodificador de lenguaje BCD a 7 segmentos. En la siguiente imagen podemos ver el esquema de patillaje del integrado <sup>18</sup>, la representación de los dígitos del sistema decimal en 7 segmentos, la asignación del lugar que ocupa cada segmento y la tabla de la verdad o de funcionamiento.



INPUTS								OUTPUTS							
LD	BI	PH <sup>(4)</sup>	D <sub>D</sub>	D <sub>C</sub>	D <sub>B</sub>	D <sub>A</sub>	O <sub>a</sub>	O <sub>b</sub>	O <sub>c</sub>	O <sub>d</sub>	O <sub>e</sub>	O <sub>f</sub>	O <sub>g</sub>	DISPLAY	
X	H	L	X	X	X	X	L	L	L	L	L	L	L	blank	
H	L	L	L	L	L	L	H	H	H	H	H	H	L	0	
H	L	L	L	L	L	H	L	H	H	L	L	L	L	1	
H	L	L	L	L	H	L	H	H	L	H	H	L	H	2	
H	L	L	L	L	H	H	H	H	H	H	L	L	H	3	
H	L	L	L	H	L	L	L	H	H	L	L	H	H	4	
H	L	L	L	H	L	H	H	L	H	H	L	H	H	5	
H	L	L	L	H	H	L	H	L	H	H	H	H	H	6	
H	L	L	L	H	H	H	H	H	H	L	L	L	L	7	
H	L	L	H	L	L	L	H	H	H	H	H	H	H	8	
H	L	L	H	L	L	H	H	H	H	H	L	H	H	9	
H	L	L	H	L	H	L	L	L	L	L	L	L	L	blank	
H	L	L	H	L	H	H	L	L	L	L	L	L	L	blank	
H	L	L	H	H	L	L	L	L	L	L	L	L	L	blank	
H	L	L	H	H	H	L	L	L	L	L	L	L	L	blank	
L	L	L	X	X	X	X	(5)							(5)	
as above		H	as above				inverse of above							as above	

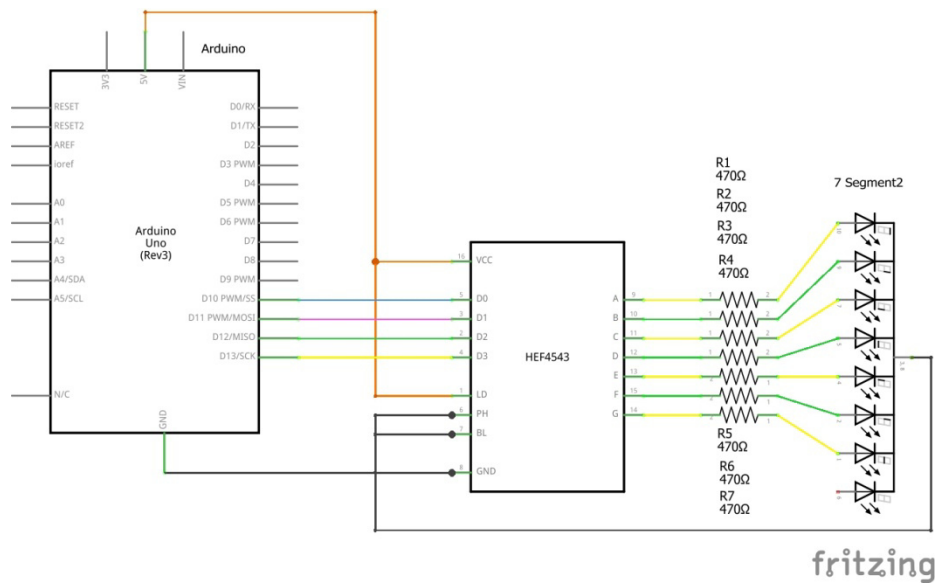
**Notes**

1. H = HIGH state (the more positive voltage)
2. L = LOW state (the less positive voltage)
3. X = state is immaterial
4. For liquid crystal displays, apply a square-wave to PH.  
For common cathode LED displays, select PH = LOW.  
For common anode LED displays, select PH = HIGH.
5. Depends upon the BCD-code previously applied when LD = HIGH.

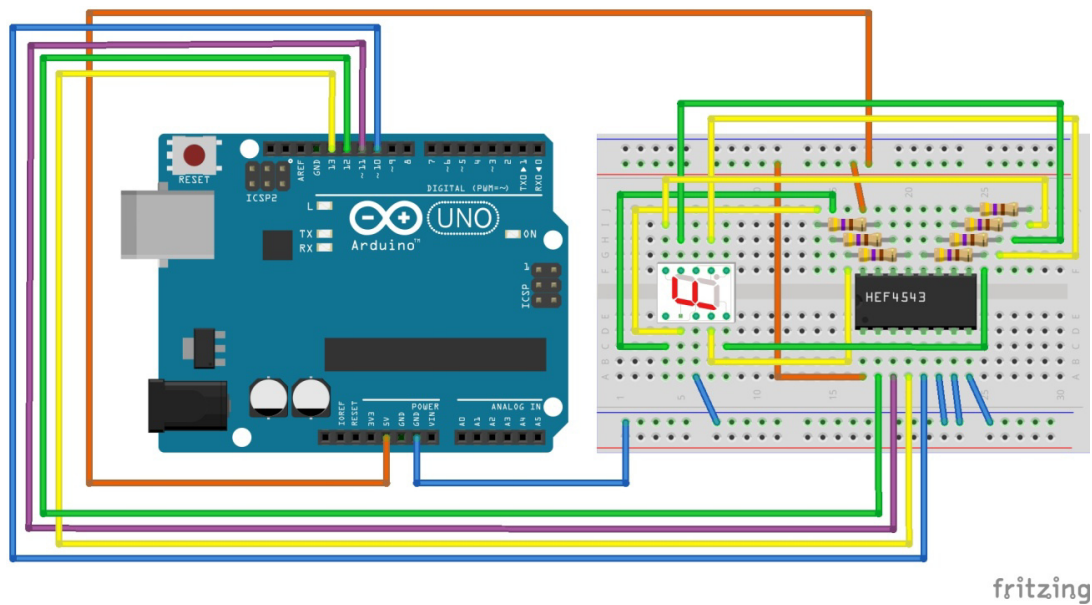
En la siguiente tabla podemos ver la asignación que vamos a realizar de las salidas digitales de Arduino a los diferentes dígitos BCD:

BCD	D <sub>D</sub> (2 <sup>3</sup> =8)	D <sub>C</sub> (2 <sup>2</sup> =4)	D <sub>B</sub> (2 <sup>1</sup> =2)	D <sub>A</sub> (2 <sup>0</sup> =1)
Salidas Arduino	13	12	11	10

El esquema eléctrico de la práctica es el siguiente:



El esquema de conexionado y montaje:



Los alumnos deberán de diseñar un programa con S4A en el que se realice una cuenta de 0 a 9 en binario y la cuenta se visualice en un display de 7 segmentos de cátodo común, de forma que la cuenta siempre comience en 0, sea ascendente y se repita constantemente hasta que detengamos el programa.

El programa en S4A es el siguiente:

```

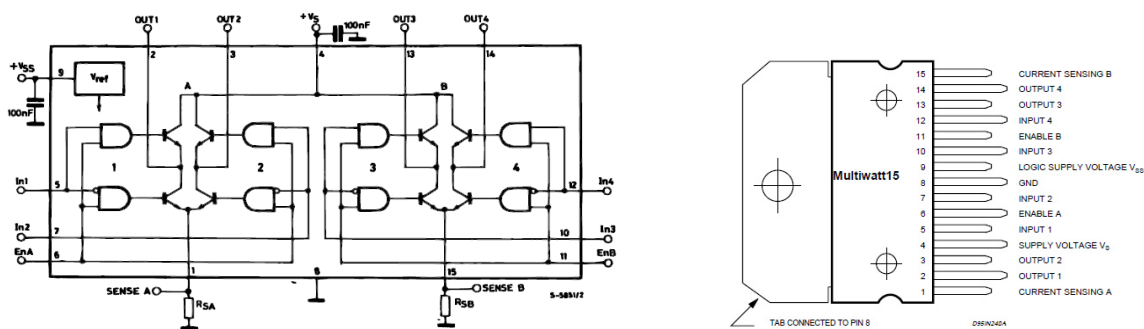
al presionar
digital 10 apagado
digital 11 apagado
digital 12 apagado
digital 13 apagado
esperar 1 segundos
por siempre
digital 10 apagado
digital 11 apagado
digital 12 apagado
digital 13 apagado
esperar 1 segundos
digital 10 encendido
esperar 1 segundos
digital 10 apagado
digital 11 encendido
esperar 1 segundos
digital 10 encendido
esperar 1 segundos
digital 10 apagado
digital 11 apagado
digital 12 encendido
esperar 1 segundos
digital 10 encendido
esperar 1 segundos
digital 10 apagado
digital 11 encendido
esperar 1 segundos
digital 10 encendido
esperar 1 segundos
digital 10 apagado
digital 11 apagado
digital 12 apagado
digital 13 encendido
esperar 1 segundos
digital 10 encendido
esperar 1 segundos
    
```

### 75. Control de un motor paso a paso bipolar mediante el driver L298N.

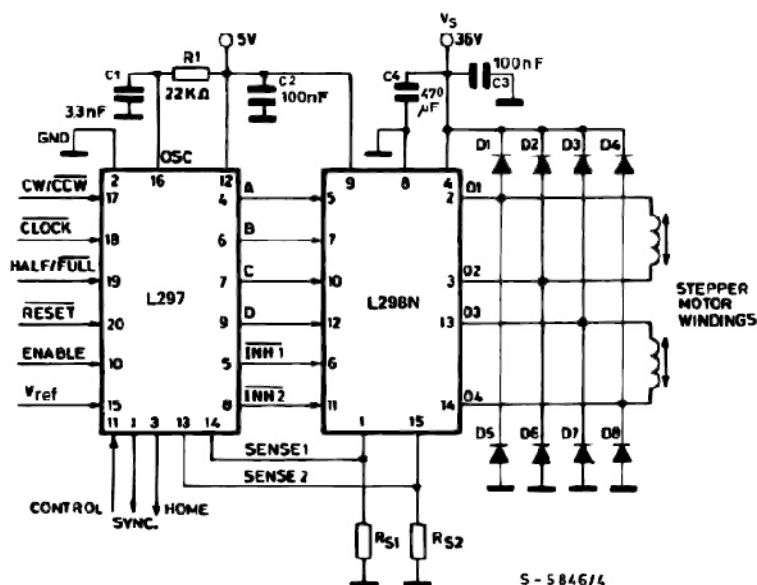
En primer lugar hay que explicar a los alumnos qué es un motor paso a paso, su principio de funcionamiento, los diferentes tipos de motores paso a paso y la forma de controlarlos. Para ello vamos a utilizar diferentes páginas con información sobre este tipo de motores:

- <http://server-die.alc.upv.es/asignaturas/lased/2002-03/MotoresPasoPaso/Motorespasoapaso.pdf><sup>15</sup>
- <https://coscomantauni.files.wordpress.com/2014/01/motor-paso-a-paso.pdf><sup>16</sup>
- <http://diymakers.es/mover-motores-paso-paso-con-arduino/><sup>17</sup>

El circuito integrado L298N es un doble puente H con el que podríamos controlar un motor paso a paso bipolar. A continuación podemos ver su esquema equivalente y el de patillaje<sup>18</sup>:



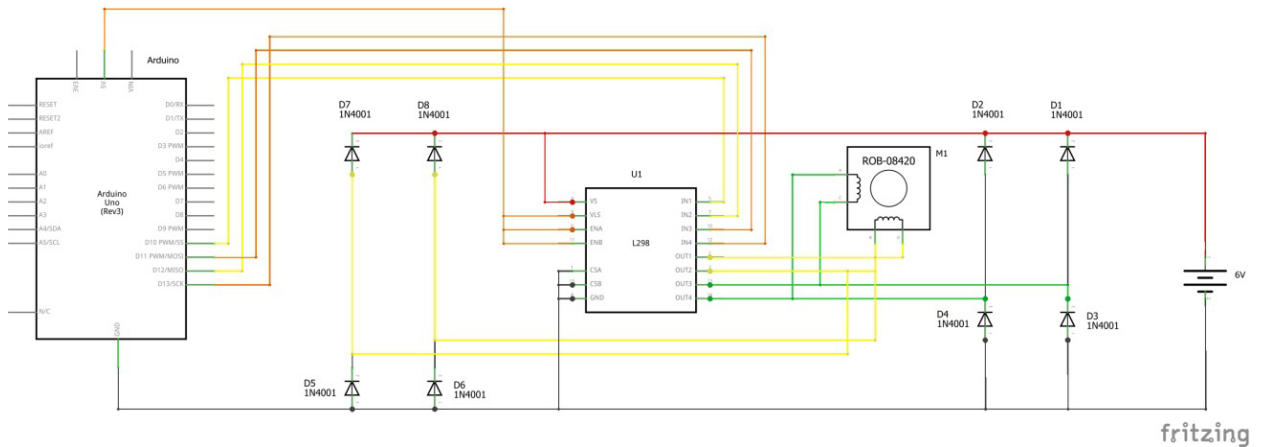
Para el control de un motor paso a paso bipolar, el fabricante nos proporciona la siguiente información:



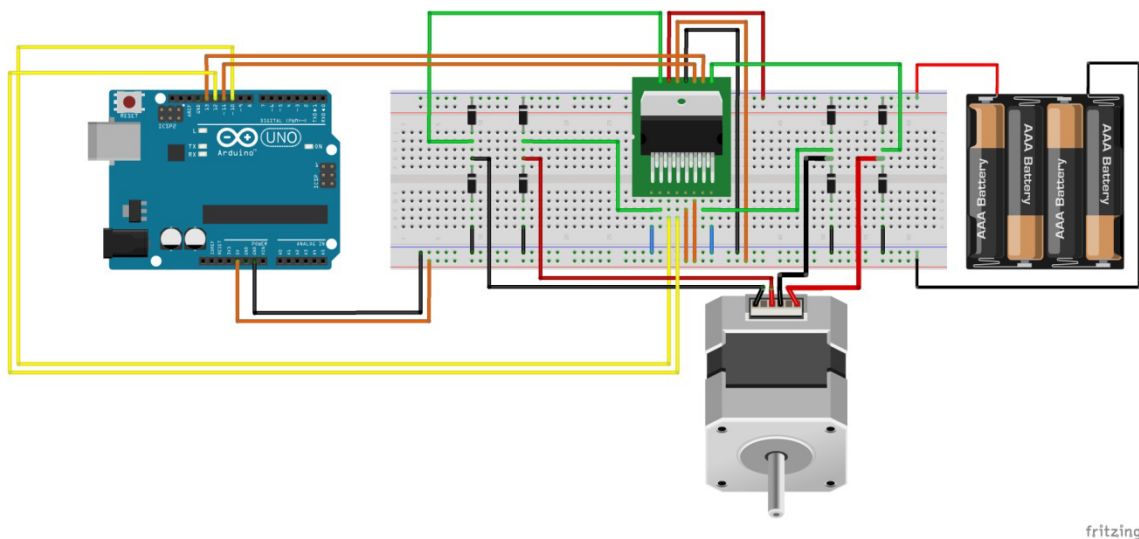
Hay que tener en cuenta que el control lo vamos a realizar mediante Arduino Uno programado con S4A, de forma que no utilizaremos el integrado L297 para el control, actuando directamente sobre el driver L298N.

Para la práctica vamos a utilizar un motor paso a paso bipolar de 200 pasos, sacado de una impresora de inyección de tinta. Este componente no se ha valorado en el presupuesto de material que aparece en el anexo IV.

El esquema eléctrico de la práctica es el siguiente:



El esquema de conexionado y montaje:



Paso	Bobina 1A	Bobina 1B	Bobina 2A	Bobina 2B
Salida Arduino	10	12	11	13
Paso 1	1	0	1	0
Paso 2	1	0	0	1
Paso 3	0	1	0	1
Paso 4	0	1	1	0

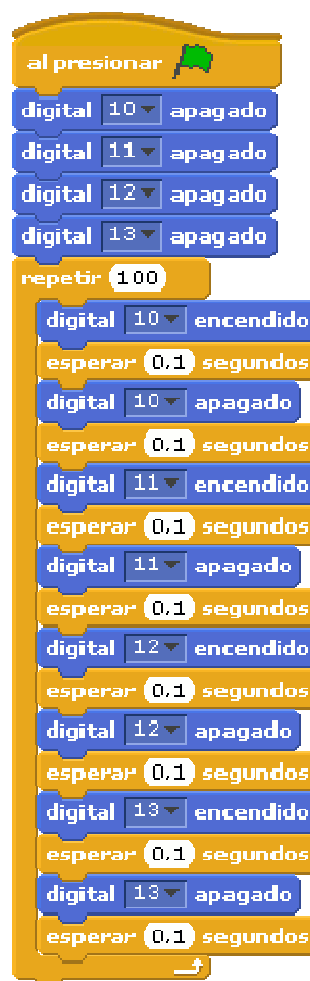
En la tabla superior, podemos ver la secuencia recomendada para alimentar los motores paso a paso bipolares. Hay que tener en cuenta que no se alimenta directamente el motor con las salidas digitales indicadas, sino que esas salidas son las que se utilizan en las entradas del L298N para controlarlo.



De cara a la programación vamos a realizar varios programas para poder ver el funcionamiento y precisión de este tipo de motores.

La primera, tal y como podemos ver en la siguiente tabla, en cada uno de los pasos vamos a alimentar solo una bobina, de forma que los alumnos podrán observar que el motor tiene poca fuerza.

Paso	Bobina 1A	Bobina 1B	Bobina 2A	Bobina 2B
Salida Arduino	10	12	11	13
Paso 1	1	0	0	0
Paso 2	0	0	1	0
Paso 3	0	1	0	0
Paso 4	0	0	0	1



El siguiente programa lo realizaremos utilizando la secuencia recomendada para alimentar los motores bipolares. En esta secuencia se alimentan dos bobinas a la vez y se juega con la dirección de la corriente por las bobinas para cambiar la polaridad de esta y obtener los movimientos. Con esta secuencia obtendremos un giro en sentido horario.

Paso	Bobina 1A	Bobina 1B	Bobina 2A	Bobina 2B
Salida Arduino	10	12	11	13
Paso 1	1	0	1	0
Paso 2	1	0	0	1
Paso 3	0	1	0	1
Paso 4	0	1	1	0



Si invertimos el orden de esa secuencia obtendremos un movimiento de giro en sentido antihorario. En la siguiente tabla podemos ver la secuencia:

Paso	Bobina 1A	Bobina 1B	Bobina 2A	Bobina 2B
Salida Arduino	10	12	11	13
Paso 1	0	1	1	0
Paso 2	0	1	0	1
Paso 3	1	0	0	1
Paso 4	1	0	1	0



Si a los pasos anteriores añadimos 4 estados intermedios en los que una de las bobinas no estará alimentada, conseguiremos movimientos de medio paso, de forma que obtendremos mayor precisión y los movimientos serán más suaves. Como contrapartida perderemos par en el motor.

Paso	Bobina 1A	Bobina 1B	Bobina 2A	Bobina 2B
Salida Arduino	10	12	11	13
Paso 1	1	0	1	0
	1	0	0	0
Paso 2	1	0	0	1
	0	0	0	1
Paso 3	0	1	0	1
	0	1	0	0
Paso 4	0	1	1	0
	0	0	1	0



Si con el sentido de giro antihorario añadimos 4 pasos intermedios, obtenemos 8 movimientos de medio paso.

Paso	Bobina 1A	Bobina 1B	Bobina 2A	Bobina 2B
Salida Arduino	10	12	11	13
Paso 1	0	1	1	0
	0	1	0	0
Paso 2	0	1	0	1
	0	0	0	1
Paso 3	1	0	0	1
	1	0	0	0
Paso 4	1	0	1	0
	0	0	1	0

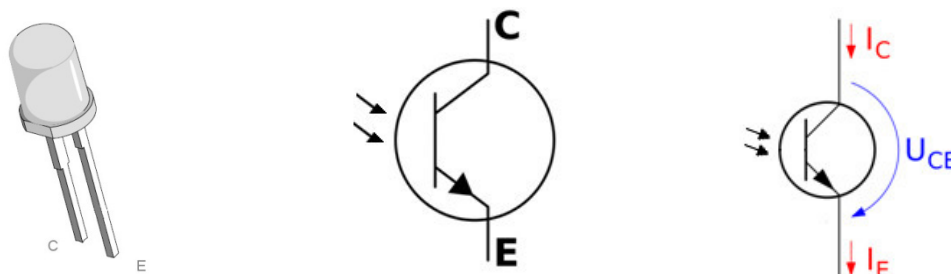
```

al presionar [bandera]
digitalWrite(10, LOW)
digitalWrite(11, LOW)
digitalWrite(12, LOW)
digitalWrite(13, LOW)
repetir (50)
  digitalWrite(11, HIGH)
  digitalWrite(12, HIGH)
  esperar (0.05) segundos
  digitalWrite(11, LOW)
  esperar (0.05) segundos
  digitalWrite(13, HIGH)
  esperar (0.05) segundos
  digitalWrite(12, LOW)
  esperar (0.05) segundos
  digitalWrite(10, HIGH)
  esperar (0.05) segundos
  digitalWrite(13, LOW)
  esperar (0.05) segundos
  digitalWrite(11, HIGH)
  esperar (0.05) segundos
  digitalWrite(10, LOW)
  esperar (0.05) segundos
  
```

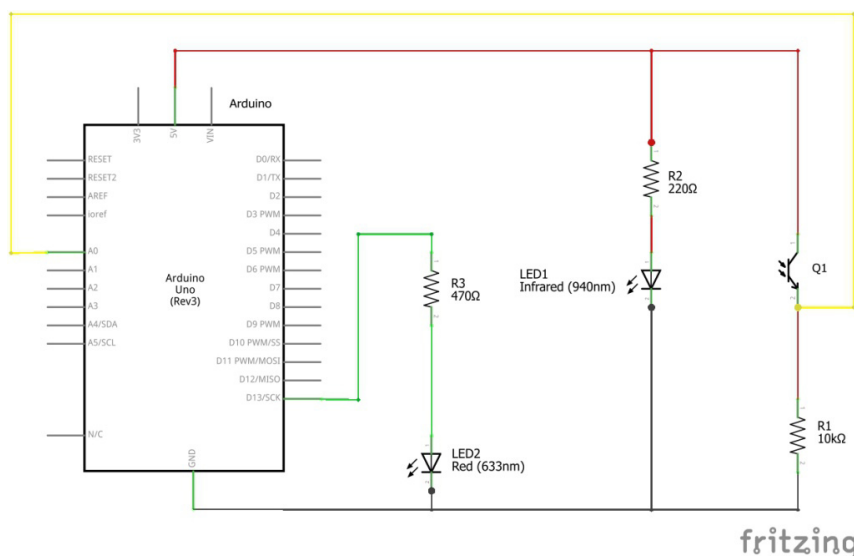
**76. Control de un diodo led, en función del estado de una barrera óptica formada por un diodo de infrarrojos TSUS 5400 y un fototransistor BPW96.**

Para esta práctica utilizaremos dos componentes nuevos. El primero de ellos es un diodo de infrarrojos TSUS 5400, el cual trabaja y se utiliza exactamente igual que los diodos vistos hasta ahora, con la única diferencia de que la luz que emite está en el espectro infrarrojo no visible.

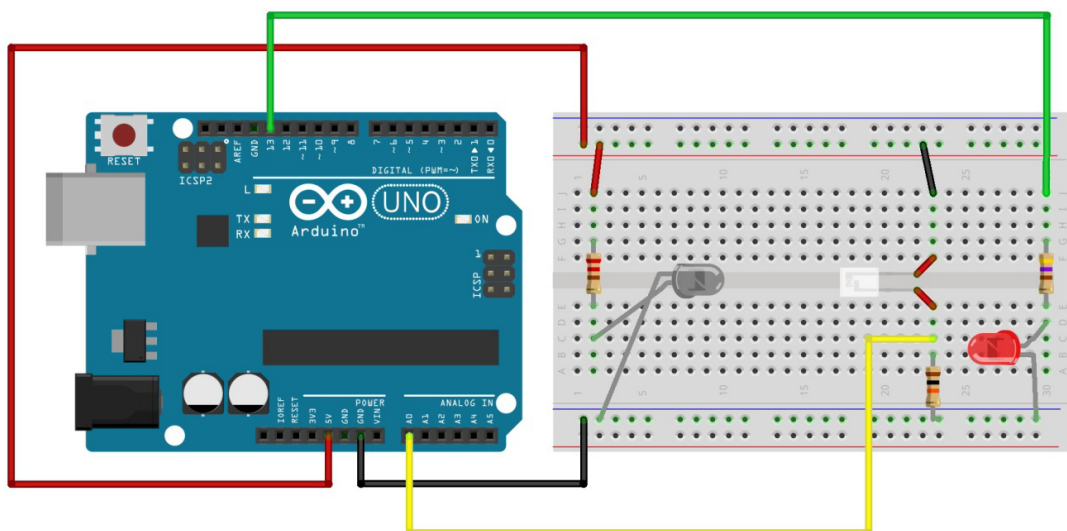
El segundo es el BPW96, un fototransistor tipo NPN sin terminal de base, el cual se encuentra en estado de corte si no recibe luz infrarroja en cantidad suficiente y si la recibe, entra en estado de conducción, permitiendo el paso de la corriente entre sus bornes de colector y emisor. A continuación podemos ver el esquema de patillaje y su símbolo eléctrico <sup>18</sup> :



El esquema eléctrico de la práctica es el siguiente:



El esquema de conexionado y montaje:



fritzing

Realizaremos un programa en S4A en el que encenderemos un diodo led si no hay ningún obstáculo entre el diodo led de infrarrojos y el fotodiodo. Si hay algún obstáculo lo apagaremos. Un ejemplo de la programación en S4A:



En esta práctica, para un correcto funcionamiento, hay que inculcar a los alumnos la importancia de una correcta alineación entre el diodo emisor de infrarrojos y el fototransistor. Será muy interesante hacer pruebas con diferentes valores de luminosidad ambiental y llegar a elegir el valor de comparación para detectar un obstáculo en cualquier condición ambiental.

Se podrán hacer pruebas con los dos tipos de barreras ópticas que podemos encontrar, es decir, emisor en un extremo y receptor en el otro extremo, y emisor y receptor en el mismo extremo y en el otro extremo un espejo o reflector, que refleje la señal del diodo emisor y la dirija hacia el fototransistor receptor. En este último caso se podría llegar a hacer pruebas colocando ambos elementos a la misma altura o a diferentes alturas, de forma que habría que modificar el ángulo del reflector.

## ANEXO III: COLECCIÓN DE PROPUESTAS DE PROYECTO

### PROYECTO 1 EL ASCENSOR PROPUESTA DE TRABAJO

Diseñar y construir un ascensor de tres plantas que se mueva mediante un motor eléctrico.

#### CONDICIONES PARTE A:

- El diseño de las piezas estructurales y mecanismos que lo formen estará realizado con Sketchup e impreso en la impresora 3D con cualquiera de los diferentes tipos de plásticos utilizados.
- Estará provisto de un servomotor de rotación continua.
- Tendrá los dispositivos necesarios para controlar la posición del ascensor y para realizar las llamadas desde cada una de las plantas.
- El control se realizará mediante una placa Arduino Uno programada con S4A y se monitorizarán las acciones en la pantalla del ordenador.
- Su tamaño no debe exceder de 200 x 280 x 400 mm.
- Los materiales y herramientas serán los definidos por el alumno en el apartado de planificación, siempre que estén disponibles en el taller o se avise con tiempo suficiente para comprarlo.

#### CONDICIONES PARTE B:

- Se realizará una adaptación del diseño de la parte A, para controlar el ascensor con un motor de corriente continua con reductora.
- Para el control del motor de corriente continua se podrán utilizar los diferentes métodos vistos durante el curso (un puente H con transistores PNP BC138 y NPN BC135, puente H confeccionado con dos amplificadores operacionales LM386 o mediante el driver L298N)
- Para la alimentación de la parte de potencia del motor de corriente continua se utilizará una pila de 4,5V.

#### PLAZO DE ENTREGA:

Para poder comenzar la construcción se debe realizar el proyecto técnico por escrito hasta el apartado de planificación. La entrega del proyecto y la construcción finaliza en Junio, la semana anterior a la evaluación, no valorando los trabajos entregados con fecha posterior.



## **PROYECTO 2**

### **LA BARRERA DE PARKING**

### **PROPUESTA DE TRABAJO**

Diseñar y construir una barrera de parking que se mueva mediante un motor eléctrico.

#### **CONDICIONES PARTE A:**

- El diseño de las piezas estructurales y mecanismos que lo formen estará realizado con Sketchup e impreso en la impresora 3D con cualquiera de los diferentes tipos de plásticos utilizados.
- Estará provisto de un miniservomotor de 180º.
- Tendrá los dispositivos necesarios para controlar la apertura de la barrera tanto en la entrada como en la salida, para evitar el cierre de la barrera si el vehículo no ha acabado de entrar o de salir, contabilidad de los coches que entran, que salen y que hay dentro e indicación con un diodo led verde si hay plazas libres y con un diodo rojo si el parking está lleno.
- El control se realizará mediante una placa Arduino Uno programada con S4A y se monitorizarán las acciones en la pantalla del ordenador.
- Su tamaño no debe exceder de 200 x 280 x 200 mm.
- Los materiales y herramientas serán los definidos por el alumno en el apartado de planificación, siempre que estén disponibles en el taller o se avise con tiempo suficiente para comprarlo.

#### **CONDICIONES PARTE B:**

- Se realizará una adaptación del diseño de la parte A, para controlar la barrera con un motor de corriente continua con reductora.
- Para el control del motor de corriente continua se podrán utilizar los diferentes métodos vistos durante el curso (un puente H con transistores PNP BC138 y NPN BC135, puente H confeccionado con dos amplificadores operacionales LM386 o mediante el driver L298N)
- Para la alimentación de la parte de potencia del motor de corriente continua se utilizará una pila de 4,5V.

#### **PLAZO DE ENTREGA:**

Para poder comenzar la construcción se debe realizar el proyecto técnico por escrito hasta el apartado de planificación. La entrega del proyecto y la construcción finaliza en Junio, la semana anterior a la evaluación, no valorando los trabajos entregados con fecha posterior.

### PROYECTO 3

## EL COHECITO ELÉCTRICO SEGUIDOR DE LÍNEA

### PROPUESTA DE TRABAJO

Diseñar y construir un vehículo que se mueva mediante dos motores eléctricos siguiendo una trayectoria negra marcada en el suelo.

#### CONDICIONES PARTE A:

- El diseño de las piezas estructurales y mecanismos que lo formen estará realizado con Sketchup e impreso en la impresora 3D con cualquiera de los diferentes tipos de plásticos utilizados.
- Estará provisto de dos servomotores eléctricos de rotación continua para poder controlar el giro del cochecito.
- Tendrá dos dispositivos para controlar el seguimiento de la trayectoria.
- El control se realizará mediante una placa Arduino Uno programada con S4A y se monitorizarán las acciones en la pantalla del ordenador.
- Su tamaño no debe exceder de 200 x 280 x 200 mm.
- Los materiales y herramientas serán los definidos por el alumno en el apartado de planificación, siempre que estén disponibles en el taller o se avise con tiempo suficiente para comprarlo.

#### CONDICIONES PARTE B:

- Se realizará una adaptación del diseño de la parte A, para controlar el cochecito con dos motores de corriente continua con reductora.
- Para el control de los motores se podrán utilizar los diferentes métodos vistos durante el curso (un puente H con transistores PNP BC138 y NPN BC135, puente H confeccionado con dos amplificadores operacionales LM386 o mediante el driver L298N)
- Para la alimentación de la parte de potencia de los motores se utilizará una pila de 4,5V.

#### CONDICIONES OPCIONALES:

- Se podrá realizar la conexión al ordenador utilizando un módulo bluetooth de Arduino o usando para el control una Raspberry.

#### PLAZO DE ENTREGA:

Para poder comenzar la construcción se debe realizar el proyecto técnico por escrito hasta el apartado de planificación. La entrega del proyecto y la construcción finaliza en Junio, la semana anterior a la evaluación, no valorando los trabajos entregados con fecha posterior.

## **PROYECTO 4**

### **EL COHECITO ELÉCTRICO SEGUIDOR DE LUZ**

#### **PROPUESTA DE TRABAJO**

Diseñar y construir un vehículo que se mueva mediante dos motores eléctricos siguiendo un haz de luz y acercándose hacia él.

#### **CONDICIONES PARTE A:**

- El diseño de las piezas estructurales y mecanismos que lo formen estará realizado con Sketchup e impreso en la impresora 3D con cualquiera de los diferentes tipos de plásticos utilizados.
- Estará provisto de dos servomotores eléctricos de rotación continua para poder controlar el giro del cochecito.
- Tendrá los dispositivos para controlar el seguimiento de la trayectoria.
- El control se realizará mediante una placa Arduino Uno programada con S4A y se monitorizarán las acciones en la pantalla del ordenador.
- Su tamaño no debe exceder de 200 x 280 x 200 mm.
- Los materiales y herramientas serán los definidos por el alumno en el apartado de planificación, siempre que estén disponibles en el taller o se avise con tiempo suficiente para comprarlo.

#### **CONDICIONES PARTE B:**

- Se realizará una adaptación del diseño de la parte A, para controlar el cochecito con dos motores de corriente continua con reductora.
- Para el control de los motores se podrán utilizar los diferentes métodos vistos durante el curso (un puente H con transistores PNP BC138 y NPN BC135, puente H confeccionado con dos amplificadores operacionales LM386 o mediante el driver L298N)
- Para la alimentación de la parte de potencia de los motores se utilizará una pila de 4,5V.

#### **CONDICIONES OPCIONALES:**

- Se podrá realizar la conexión al ordenador utilizando un módulo bluetooth de Arduino o usando para el control una Raspberry.

#### **PLAZO DE ENTREGA:**

Para poder comenzar la construcción se debe realizar el proyecto técnico por escrito hasta el apartado de planificación. La entrega del proyecto y la construcción finaliza en Junio, la semana anterior a la evaluación, no valorando los trabajos entregados con fecha posterior.

## **PROYECTO 5**

### **LA PUERTA AUTOMÁTICA DE GARAJE**

#### **PROPUESTA DE TRABAJO**

Diseñar y construir una puerta automática de garaje que se mueva mediante un motor eléctrico.

#### **CONDICIONES PARTE A:**

- El diseño de las piezas estructurales y mecanismos que lo formen estará realizado con Sketchup e impreso en la impresora 3D con cualquiera de los diferentes tipos de plásticos utilizados.
- Estará provisto de un servomotor eléctrico de rotación continua para poder mover el mecanismo de la puerta.
- Tendrá los dispositivos necesarios para controlar la apertura de la puerta tanto en la entrada como en la salida, para evitar el cierre de la puerta si el vehículo no ha acabado de entrar o de salir o también si alguna persona se interpone en la trayectoria de la puerta en movimiento (si la puerta se está cerrando, se detendrá y se volverá a abrir), e indicación con un diodo led ambar con intermitencia cuando hay movimiento de la puerta.
- El control se realizará mediante una placa Arduino Uno programada con S4A y se monitorizarán las acciones en la pantalla del ordenador.
- Su tamaño no debe exceder de 200 x 280 x 200 mm.
- Los materiales y herramientas serán los definidos por el alumno en el apartado de planificación, siempre que estén disponibles en el taller o se avise con tiempo suficiente para comprarlo.

#### **CONDICIONES PARTE B:**

- Se realizará una adaptación del diseño de la parte A, para controlar la puerta con un motor de corriente continua con reductora.
- Para el control del motor de corriente continua se podrán utilizar los diferentes métodos vistos durante el curso (un puente H con transistores PNP BC138 y NPN BC135, puente H confeccionado con dos amplificadores operacionales LM386 o mediante el driver L298N)
- Para la alimentación de la parte de potencia del motor de corriente continua se utilizará una pila de 4,5V.

#### **PLAZO DE ENTREGA:**

Para poder comenzar la construcción se debe realizar el proyecto técnico por escrito hasta el apartado de planificación. La entrega del proyecto y la construcción finaliza en Junio, la semana anterior a la evaluación, no valorando los trabajos entregados con fecha posterior.

**PROYECTO**  
**EL SEGUIDOR SOLAR**  
**PROPUESTA DE TRABAJO**

Diseñar y construir un seguidor solar que se mueva mediante dos motores eléctricos, siguiendo la trayectoria del sol, de forma que los rayos actúen de la forma más perpendicular posible.

**CONDICIONES PARTE A:**

- El diseño de las piezas estructurales y mecanismos que lo formen estará realizado con Sketchup e impreso en la impresora 3D con cualquiera de los diferentes tipos de plásticos utilizados.
- Estará provisto de un servomotor eléctrico de rotación continua y otro miniservomotor de 180º.
- Tendrá los dispositivos necesarios para controlar el seguimiento de la trayectoria.
- El control se realizará mediante una placa Arduino Uno programada con S4A y se monitorizarán las acciones en la pantalla del ordenador.
- Su tamaño no debe exceder de 200 x 280 x 200 mm.
- Los materiales y herramientas serán los definidos por el alumno en el apartado de planificación, siempre que estén disponibles en el taller o se avise con tiempo suficiente para comprarlo.

**CONDICIONES PARTE B:**

- Se realizará una adaptación del diseño de la parte A, para controlar el cochecito con un miniservomotor de 180º y un motor de corriente continua con reductora.
- Para el control del motor de corriente continua se podrán utilizar los diferentes métodos vistos durante el curso (un puente H con transistores PNP BC138 y NPN BC135, puente H confeccionado con dos amplificadores operacionales LM386 o mediante el driver L298N)
- Para la alimentación de la parte de potencia del motor de corriente continua se utilizará una pila de 4,5V.

**PLAZO DE ENTREGA:**

Para poder comenzar la construcción se debe realizar el proyecto técnico por escrito hasta el apartado de planificación. La entrega del proyecto y la construcción finaliza en Junio, la semana anterior a la evaluación, no valorando los trabajos entregados con fecha posterior.

## ANEXO IV: LISTA DE MATERIALES Y PRESUPUESTO

Designación	Cantidad	Precio unitario	Precio total
Placa Arduino Uno Rev3	1	24,00 €	24,00 €
Cable USB tipo A-B	1	2,25 €	2,25 €
Placa board mediana	1	6,00 €	6,00 €
Placa board grande	1	8,40 €	8,40 €
Cables de conexión con jumpers	1	6,00 €	6,00 €
Diodo led rojo	5	0,18 €	0,90 €
Diodo led verde	5	0,18 €	0,90 €
Diodo led ambar	5	0,18 €	0,90 €
Diodo led multicolor cambio lento	1	1,70 €	1,70 €
Diodo led multicolor cambio rápido	1	1,60 €	1,60 €
Diodo led bicolor rojo-verde	1	0,59 €	0,59 €
Diodo led RGB cátodo común	1	1,21 €	1,21 €
Pulsador normalmente abierto	2	1,03 €	2,06 €
Pulsador normalmente cerrado	2	1,03 €	2,06 €
Pulsador para circuito integrado	2	0,44 €	0,88 €
Final de carrera	2	1,73 €	3,46 €
Ampolla relé reed	2	1,56 €	3,12 €
Imán rectangular	1	0,29 €	0,29 €
Potenciómetro 100 ohmios	2	0,58 €	1,16 €
Potenciómetro 1 Kohmio	2	0,58 €	1,16 €
Potenciómetro 10 Kohmios	2	0,58 €	1,16 €
Potenciómetro 100 Kohmios	2	0,58 €	1,16 €
Potenciómetro 1 Mohmio	2	0,58 €	1,16 €
LDR	1	0,95 €	0,95 €
Zumbador electrónico	1	3,00 €	3,00 €
Zumbador piezo-eléctrico	1	4,00 €	4,00 €
Relé simple 4,5-9 V 2 circuitos de conmutación	4	3,36 €	13,44 €
Transistor NPN BD135	2	3,63 €	7,26 €
Diodo de silicio 1N4007	10	0,08 €	0,80 €
Puente de 4 diodos	1	0,32 €	0,32 €
Portapilas 4xR6 (4 pilas de 1,5V AA)	1	0,59 €	0,59 €
Pilas de 1,5V AA (Paquete de 4)	1	2,68 €	2,68 €
Transistor PNP BD138	2	3,63 €	7,26 €
Pila de petaca 4,5V	1	1,68 €	1,68 €
Termistor NTC 1 Kohmios	1	0,99 €	0,99 €
Termistor NTC 10 Kohmios	1	0,99 €	0,99 €
Termistor PTC 600 ohmios	1	0,90 €	0,90 €
Sensor de temperatura LM35	1	2,60 €	2,60 €
Optoacoplador CNZ1102	1	2,10 €	2,10 €
Optoacoplador CNY70 por reflexión	1	2,18 €	2,18 €
Transistor NPN BC547	2	0,08 €	0,16 €
Portalámparas E-10 lengüeta metálica conexión ambos lados	2	0,68 €	1,36 €

Designación	Cantidad	Precio unitario	Precio total
Bombillita esférica rosca E-10 5V-0,3A	2	2,74 €	5,48 €
Transistor PNP BC557	2	0,95 €	1,90 €
Motor con reductora 17:1	1	9,28 €	9,28 €
Amplificador operacional de potencia LM386	2	1,06 €	2,12 €
Driver 298N	1	3,00 €	3,00 €
Miniservomotor angular 180º	1	8,40 €	8,40 €
Servomotor de rotación continua	1	19,65 €	19,65 €
Display numérico FYS-3011A	1	1,87 €	1,87 €
Decodificador BCD a 7 segmentos HEF4543B	1	0,73 €	0,73 €
Resistencia 470 ohmios (Paquete de 10)	2	0,41 €	0,82 €
Resistencia 1 Kohmio (Paquete de 10)	1	0,41 €	0,41 €
Resistencia 10 Kohmio (Paquete de 10)	2	0,41 €	0,82 €
Resistencia 100Kohmio (Paquete de 10)	1	0,41 €	0,41 €
Resistencia 1 Mohmio (Paquete de 10)	1	0,41 €	0,41 €
Led de infrarrojos TSUS 5400	1	0,59 €	0,59 €
Fototransistor BPW 965	1	1,20 €	1,20 €
Conector para pila de 9V	1	0,42 €	0,42 €
Maletín con compartimentos	1	3,40 €	3,40 €

<b>TOTAL</b>	<b>186,29 €</b>
--------------	-----------------

- Los precios unitarios se han tomado de catálogos de fabricantes de material educativo con la tarifa del curso 16/17 (Microlog, Opitec, Libelium), tomando el precio más económico para cada uno de los componentes.
- Los precios llevan el I.V.A. incluido.
- Los precios se han tomado para un componente. Hay fabricantes que hacen descuentos si se compran paquetes de 10 componentes.
- De cara al proyecto, hay que tener en cuenta que gran parte del material usado ya se tiene en un taller de tecnología, por lo que el coste de este proyecto disminuye considerablemente.

## ANEXO V: PRÁCTICA COMPLETA A DESARROLLAR EN LA EXPOSICIÓN

La práctica que vamos a utilizar es la práctica 68, debido a que es una práctica muy completa en la que se usan la mayoría de los operadores que se han de ver en la asignatura de Tecnología en 4º ESO. El título de la práctica es: control total (encendido, apagado, sentido de giro y velocidad) de un motor de corriente continua mediante un puente H con transistores PNP BC138 y NPN BC135, controlados por otros dos transistores NPN BC547. La práctica, para que se pueda ver claramente y de forma global el trabajo realizado y el potencial de la placa Arduino Uno, la vamos a completar con la monitorización del motor (explicado en la práctica 2), led indicativos del estado del motor (explicado en varias prácticas del comienzo del cuaderno) y pulsadores virtuales de marcha en ambos sentidos y paro (explicado en práctica 67) que funcionen de forma simultánea con los pulsadores reales. Y le vamos a añadir la práctica 41, control del accionamiento, en función de la luminosidad ambiental, de un relé, de forma indirecta a través de un transistor NPN BD135, con posibilidad de elegir y modificar el nivel de referencia de luminosidad, la práctica 45, termómetro digital con LM35 con indicación de la temperatura en pantalla, la práctica 71, control de un miniservo (180º) mediante un deslizador en pantalla y la práctica 72, control de un servo de rotación continua.

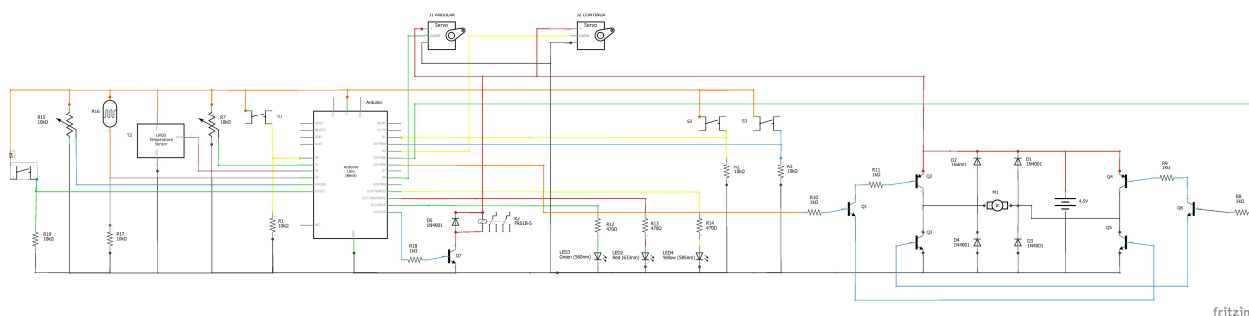
En primer lugar, en el cuaderno de prácticas, se van a dar una serie de consejos y referencias sobre como compatibilizar la parte teórica de la asignatura con la parte práctica. Fundamentalmente se va a centrar en el uso del libro multimedia <http://www.tecno12-18.com/><sup>12</sup>, el cual facilita el trabajo de autoaprendizaje por parte del alumno. El libro, mediante elección y configuración por parte del profesor, cuenta con varias unidades o capítulos. Los contenidos directamente relacionados con este proyecto se encuentran dentro de los capítulos de electrónica analógica, electrónica digital, control y robótica. Dentro de cada capítulo, los contenidos se encuentran distribuidos en miniunidades o apartados, los cuales, tienen versión dinámica y versión libre. Se recomienda usar la primera opción, de forma que el alumno trabaje con la versión dinámica, vaya aprendiendo poco a poco, haciendo un esquema resumen en el cuaderno y realizando los test de autoevaluación, de forma que es la propia página, la que si no se realizan correctamente los test, obliga a repasar la parte de teoría relacionada con el test para volver a hacerlo. La versión libre la puede usar el alumno para repasar, estudiar o concretar y el profesor para explicar los aspectos que quiera recalcar o solucionar las dudas que puedan surgir. En ocasiones, utilizamos la versión libre para trabajar el cuestionario final en el cuaderno digital.

En otras ocasiones, en las que los contenidos a tratar no aparecen en el libro multimedia anteriormente comentado, se proporcionará el material necesario, normalmente a partir de explicaciones en el propio cuaderno o a partir de páginas web externas con videos incluidos. Este es el caso de los servomotores y de los motores paso a paso.

A continuación, se presentará el esquema eléctrico funcional del montaje a realizar para la correspondiente práctica. Hay que tener en cuenta, que van a ser esquemas sencillos y que se van a ir proporcionando para cada una de las prácticas, con la idea de ir avanzando poco a poco y con la seguridad de que los alumnos están comprendiendo perfectamente lo que se está trabajando. En la práctica propuesta para la defensa del proyecto, vamos a ver que el esquema es complejo. Esto se debe a que es una práctica ya de por si compleja y además se ha completado con partes de otras

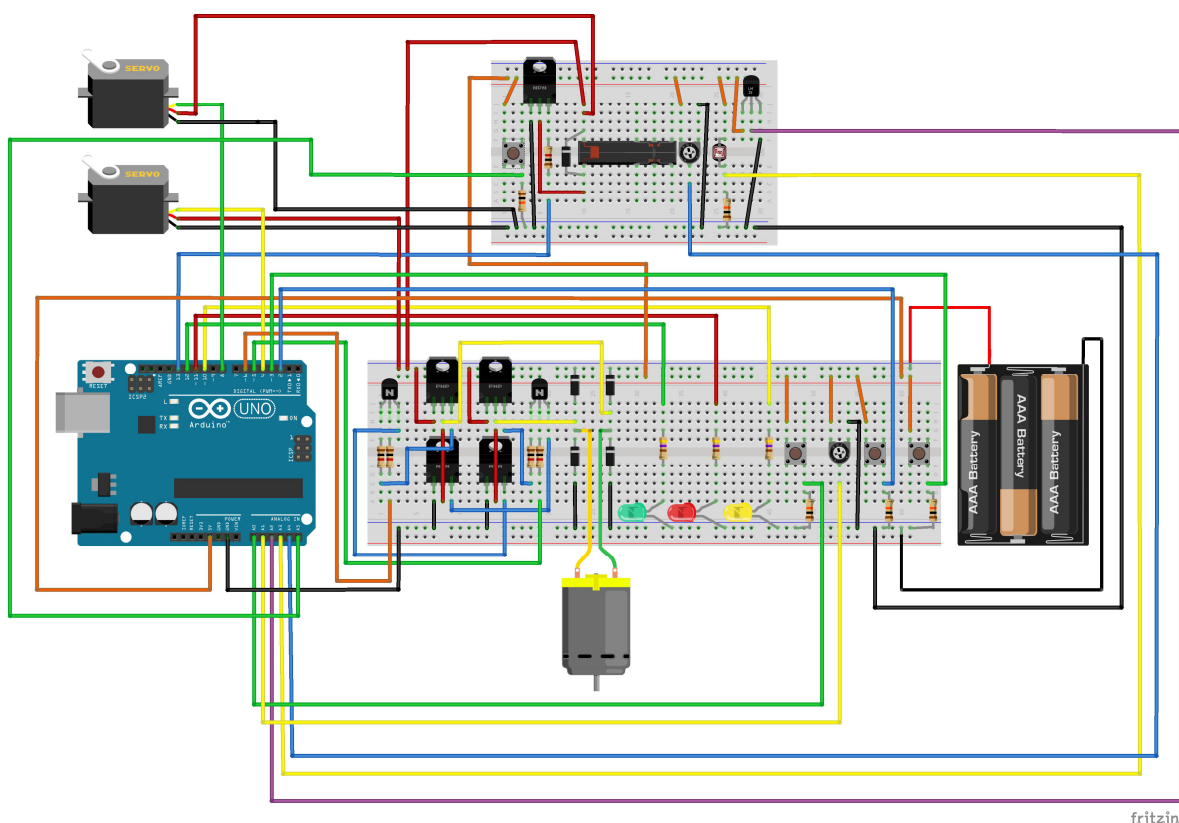


prácticas. En la explicación de la práctica, la vamos a ir dividiendo en partes para poder mostrar los esquemas de forma completa y poder ser vistos y explicados con más comodidad.



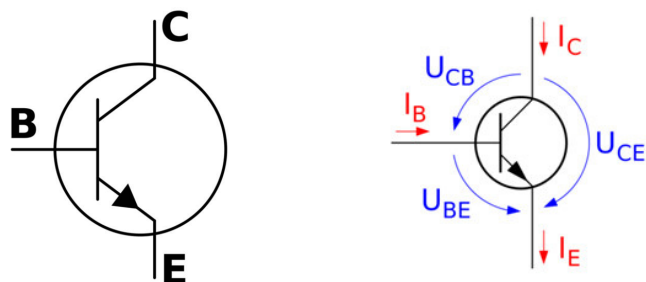
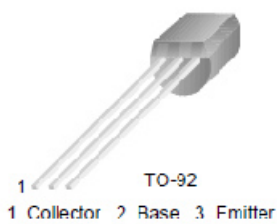
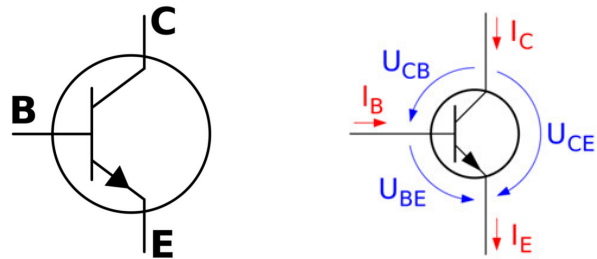
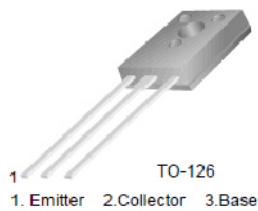
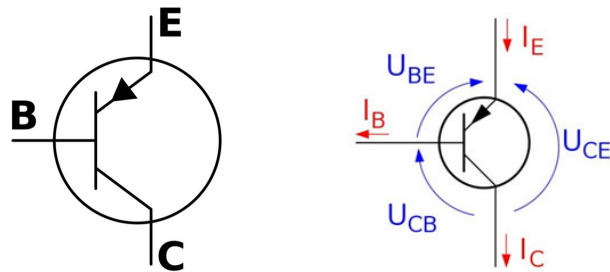
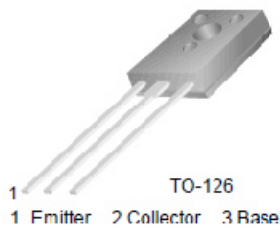
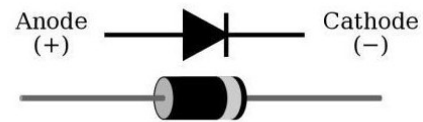
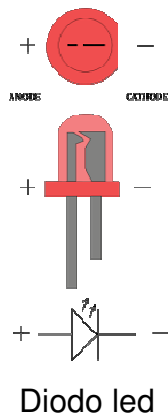
Posteriormente y para continuar avanzando, se proporcionará información detallada sobre los componentes necesarios y los esquemas de patillaje para el correcto conexionado de los mismos en el circuito.

También se proporcionará el esquema de conexionado y montaje de los diferentes componentes, tanto a la placa board como a la placa Arduino Uno Rev 3. Este esquema no se facilitará a los alumnos, ya que uno de los objetivos es que aprendan a montarlo a partir del esquema funcional. Solo se usará para explicar la conexión de componentes no vistos hasta el momento.



Una vez montado el sistema a controlar, pasaremos a explicar a los alumnos la forma en la que se realiza la programación, utilizando explicaciones orales, explicaciones escritas o explicaciones gráficas mediante grafcet. A esto añadiremos la explicación de los bloques de programación que no se hayan visto todavía en las prácticas anteriores y dejaremos a los alumnos autonomía para que intenten programar ellos solos la solución.

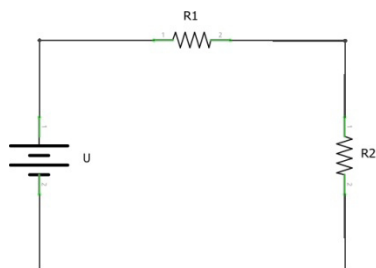
Comenzando por la práctica 68, que como hemos comentado es la base de esta presentación, lo primero que vamos a mostrar son los esquemas de patillaje de aquellos componentes que se han explicado en prácticas anteriores y que son necesarios para esta práctica <sup>18</sup> :



A continuación será necesario que los alumnos conozcan la forma correcta de introducir señales digitales a la placa Arduino.

En primer lugar hay que explicar una serie de conocimientos muy básicos y sencillos, que para los alumnos serán de repaso y aplicación de lo visto en la asignatura de Tecnología en los cursos anteriores de la etapa.

El primer concepto es el de divisor de tensión:



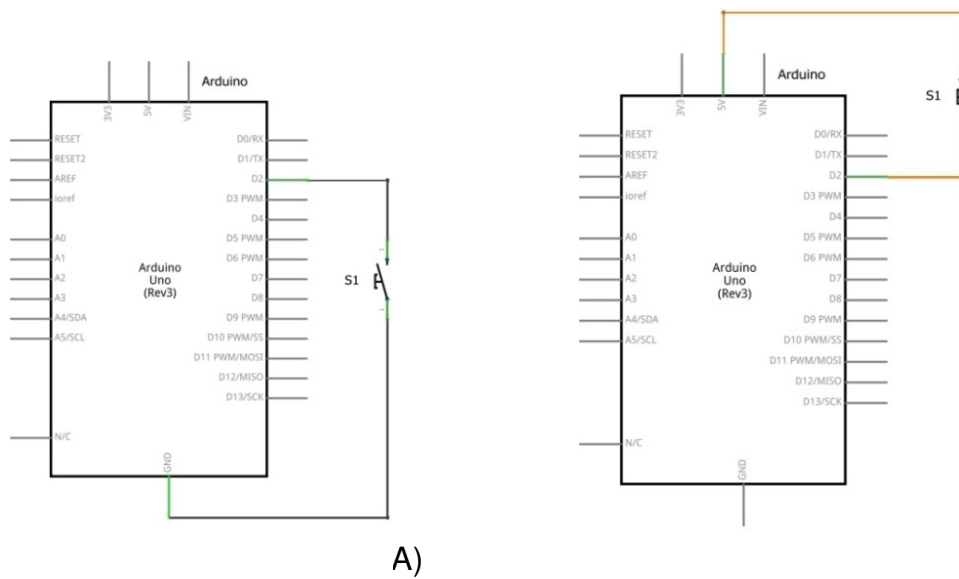
$$R_T = R_1 + R_2$$

$$I_T = \frac{U}{R_T}; I_T = I_1 = I_2$$

$$U_1 = R_1 \cdot I_1 = R_1 \cdot \frac{U}{R_T} = U \cdot \frac{R_1}{R_1 + R_2}$$

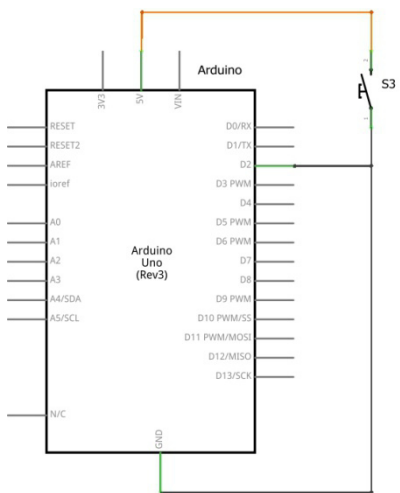
$$U_2 = R_2 \cdot I_2 = R_2 \cdot \frac{U}{R_T} = U \cdot \frac{R_2}{R_1 + R_2}$$

Repasados estos sencillos conceptos de la simplificación y cálculo de un circuito de resistencias en serie, pasamos a ver cuatro formas incorrectas de introducir señales digitales.

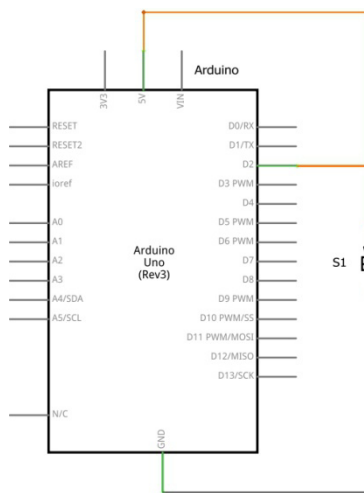


En el caso A, cuando pulsemos el pulsador S<sub>1</sub> llegarán 0V a la entrada D2, pero cuando lo soltemos la tensión que llegará a la entrada D2 será indeterminada y por lo tanto se producirán errores.

En el caso B), cuando pulsemos el pulsador S<sub>1</sub> llegarán 5V a la entrada D2, pero cuando lo soltemos la tensión que llegará a la entrada D2 será indeterminada y por lo tanto se producirán errores.



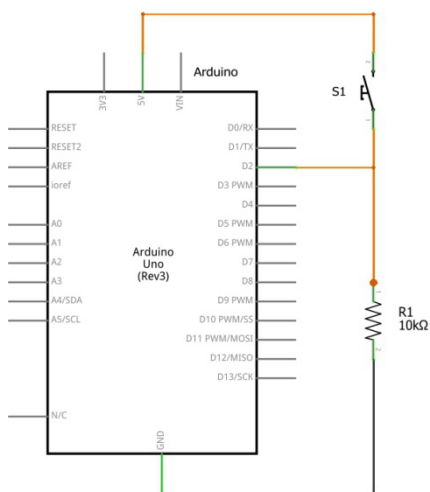
fritzing C)



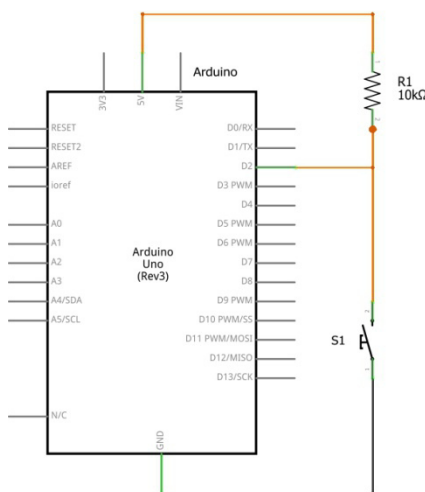
D)

En el caso C), sin pulsar llegarán 0V a la entrada D2 y pulsando cortocircuitaremos la salida de 5V de la fuente de alimentación. En el D), cuando no pulsemos el pulsador S<sub>1</sub> llegarán 5V a la entrada D2, pero cuando lo pulsemos cortocircuitaremos la salida de 5V de la fuente de alimentación de Arduino Uno.

La solución es usar estos dos últimos circuitos pero limitando la corriente con una resistencia de 10kΩ. A continuación podemos ver la forma correcta de introducir señales digitales en Arduino Uno.



Pull-down o lógica directa



Pull-up o lógica inversa

En pull-down o lógica directa, cuando el pulsador está sin pulsar a la entrada D2 llegan exactamente 0V (no pulsado → 0 lógico). Cuando el pulsador está pulsado, a la entrada D2 llegan exactamente 5V (pulsado → 1 lógico) y R<sub>1</sub> limita la corriente para evitar el cortocircuito, reduciéndola a una corriente muy pequeña, casi despreciable.

Hay que hacer ver a los alumnos que esto es una aplicación del divisor de tensión. Suponiendo que R<sub>2</sub> es la resistencia del pulsador (Pulsador sin pulsar → R<sub>2</sub> = ∞ Ω, pulsador pulsado → R<sub>2</sub> = 0 Ω) y que estamos introduciendo a través de la entrada D2 la tensión en la resistencia R<sub>1</sub>.

$$\text{Pulsador sin pulsar: } U_1 = U \cdot \frac{R_1}{R_1 + R_2} = 5V \cdot \frac{10k\Omega}{10k\Omega + \infty k\Omega} \cong 0V$$

$$\text{Pulsador pulsado: } U_1 = U \cdot \frac{R_1}{R_1 + R_2} = 5V \cdot \frac{10k\Omega}{10k\Omega + 0\Omega} = 5V$$

En pull-up o lógica inversa, cuando el pulsador está sin pulsar a la entrada D2 llegan exactamente 5V (no pulsado  $\rightarrow$  1 lógico). Cuando el pulsador está pulsado, a la entrada D2 llegan exactamente 0V (pulsado  $\rightarrow$  0 lógico) y  $R_1$  limita la corriente para evitar el cortocircuito, reduciéndola a una corriente muy pequeña, casi despreciable.

Hay que hacer ver a los alumnos que esto es una aplicación del divisor de tensión. Suponiendo que  $R_2$  es la resistencia del pulsador (Pulsador sin pulsar  $\rightarrow R_2 = \infty \Omega$ , pulsador pulsado  $\rightarrow R_2 = 0 \Omega$ ) y que estamos introduciendo a través de la entrada D2 la tensión en el pulsador  $R_2$ .

$$\text{Pulsador sin pulsar: } U_2 = U \cdot \frac{R_2}{R_1 + R_2} = 5V \cdot \frac{\infty k\Omega}{10k\Omega + \infty k\Omega} \cong 5V$$

$$\text{Pulsador pulsado: } U_2 = U \cdot \frac{R_2}{R_1 + R_2} = 5V \cdot \frac{0k\Omega}{10k\Omega + 0\Omega} = 0V$$

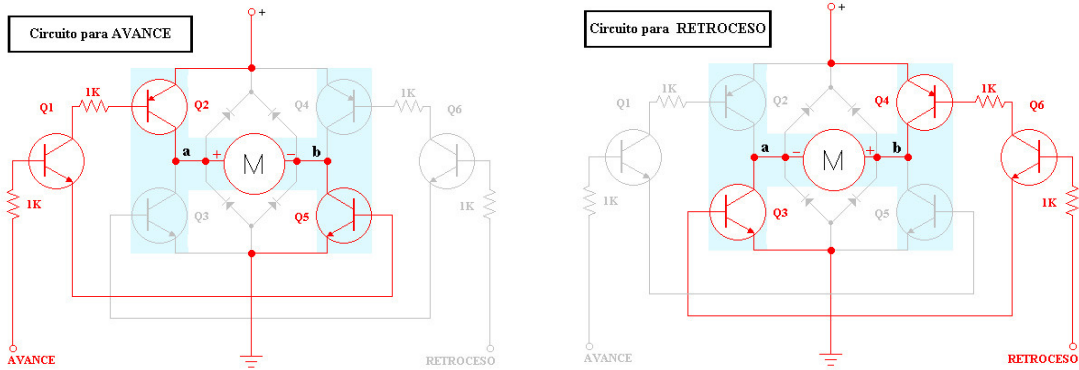
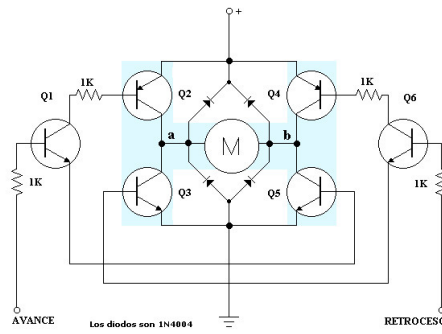
Como hemos podido observar, para introducir señales, tanto digitales como posteriormente analógicas y para alimentar pequeñas cargas, podemos utilizar la salida de 5V que tiene la fuente de alimentación de la placa Arduino Uno.

Como nos ocurre con las salidas digitales, en ocasiones no son suficientes y tenemos que utilizar las salidas analógicas como salidas digitales. Lo mismo ocurre con las entradas digitales de las que solo disponemos 2. Por ello vamos a aprender a utilizar las 6 entradas analógicas como si fueran digitales.

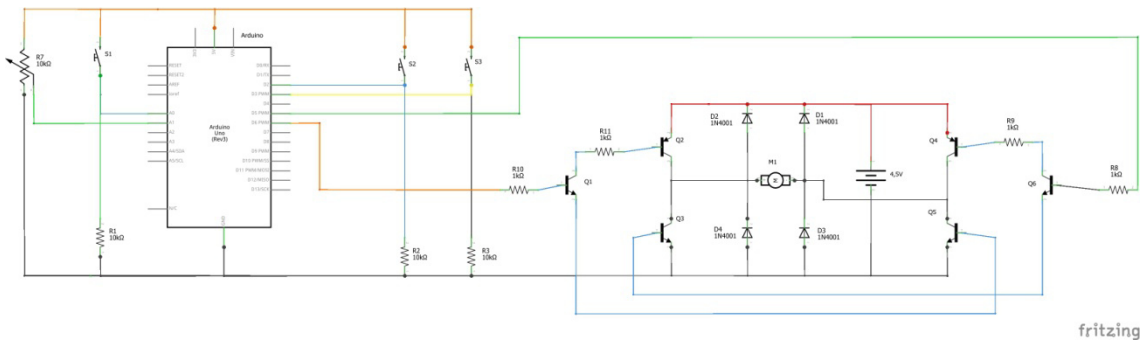
Las entradas analógicas de la placa Arduino Uno trabajan con un conversor analógico-digital de 10 bits, oscilando sus valores entre 0 y 1023 ( $2^{10} = 1024$  valores diferentes), correspondiendo un valor de entrada de 0V con conversión 0 y un valor de entrada de 5V con conversión 1023.

Para poder usar las entradas analógicas como entradas digitales, haremos una comparación con el valor constante 1000, no con el máximo que puede alcanzar la entrada (1023). Esto es debido a que puede haber fallos en conexiones, caídas de tensión en cables, ... y lo que es seguro es que si desde el pulsador nos llega un valor superior a 1000, es porque el pulsador está pulsado y su contacto cerrado.

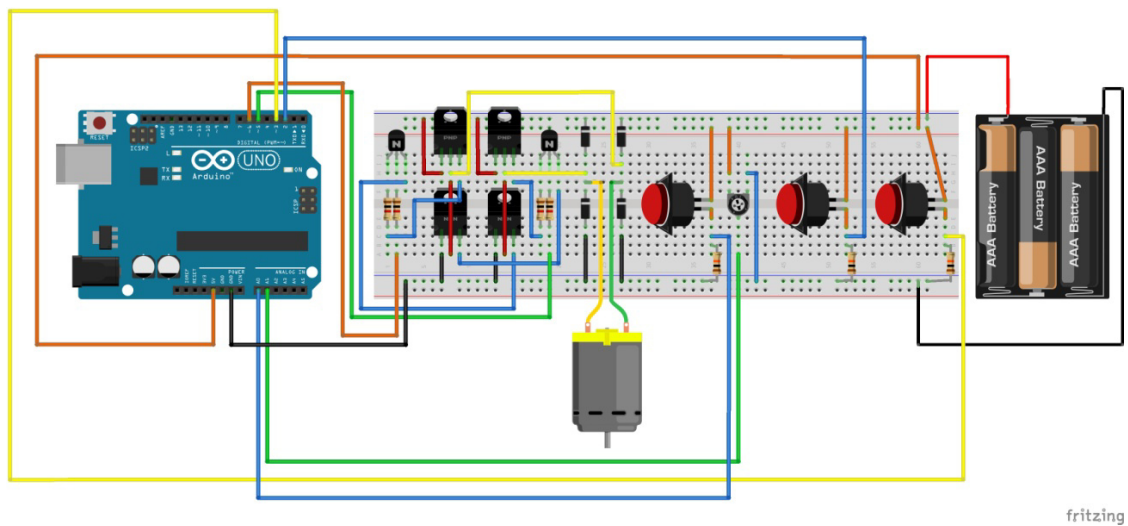
Posteriormente habrá que hacer una explicación detallada del puente H, del esquema eléctrico correspondiente y de su funcionamiento. Hay que recordar a los alumnos, que para cambiar el sentido de giro de un motor de corriente continua, hay que invertir el sentido de la corriente que circula a través de sus bobinas. También habrá que hacer especial hincapié en la función de los diodos de protección o snubber cuando se trabaja con elementos inductivos (bobinas). Para ello el profesor se podrá ayudar de los siguientes esquemas obtenidos de la web [http://robots-argentina.com.ar/MotorCC\\_PuenteH.htm](http://robots-argentina.com.ar/MotorCC_PuenteH.htm)<sup>19</sup>:



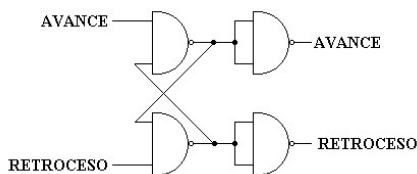
El esquema eléctrico de esta parte de la práctica es el siguiente:



El esquema de conexionado y montaje de esta parte de la práctica:



Hay que hacer especial énfasis en que al puente H nunca le debe de llegar de forma simultánea alimentación de avance (giro sentido horario pulsador S2) y de retroceso (giro sentido antihorario pulsador S1) ya que esto originaría un importante cortocircuito, dañando como mínimo y de forma permanente los 4 transistores que forman el puente H. Para evitar esto se puede complementar la práctica con electrónica digital mediante el siguiente circuito de puertas lógicas:



Hay que tener en cuenta que si usamos este circuito perdemos la capacidad de regular la velocidad del motor, ya que las salidas de avance y retroceso del circuito de puertas lógicas son digitales, es decir, todo (5V) o nada (0V). Por lo tanto, si queremos poder regular la velocidad del motor (resistencia variable R7) en ambos sentidos de giro, deberemos de hacer el enclavamiento mediante la programación en S4A.

Un ejemplo de programación en S4A para el control del giro en sentido horario y antihorario:

```

al presionar
por siempre
  fijar velocidad a redondear valor del sensor Analógico1 * 255 / 1023

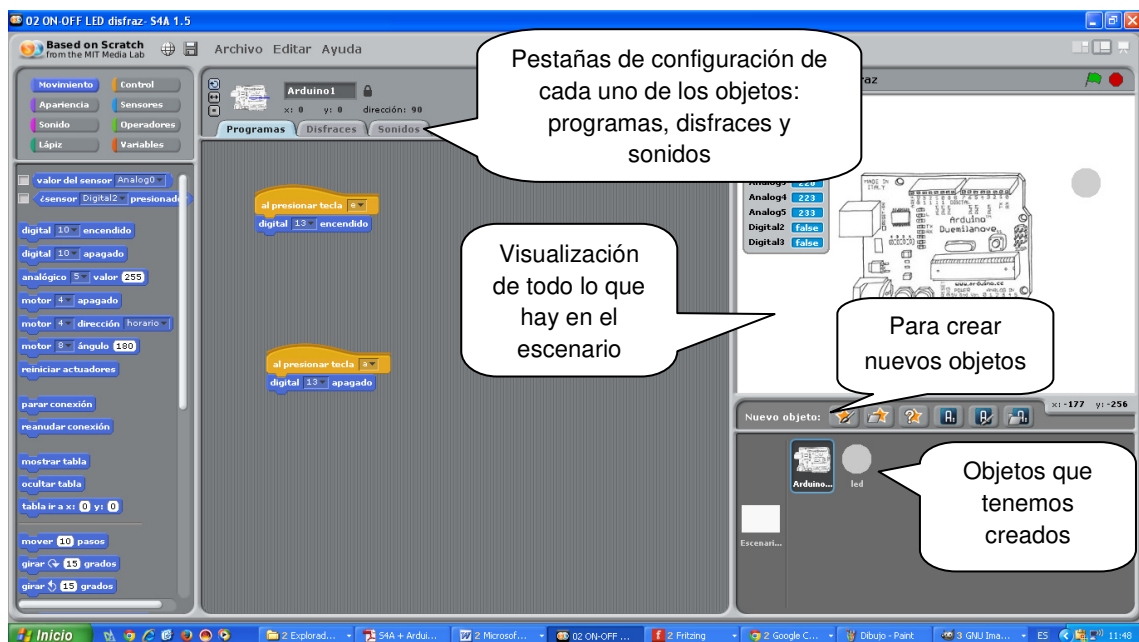
al presionar
analógico 5 valor 0
fijar giro derecha a 0
fijar DERECHA a 0
digital 10 apagado
enviar a todos Motor parado
por siempre
  si <<sensor Digital2 presionado? >> no <<sensor Digital3 presionado? >> o DERECHA = 1 y giro izquierda = 0
    fijar STOP a 0
    enviar a todos PULS DER ON
    enviar a todos PULS IZQ ON
    repetir hasta que <<sensor Digital3 presionado? >> o STOP = 1
      fijar giro derecha a 1
      fijar DERECHA a 1
      analógico 5 valor velocidad
      digital 10 encendido
      enviar a todos Motor derecha
    fijar giro derecha a 0
    fijar DERECHA a 0
    digital 10 apagado
    analógico 5 valor 0
    enviar a todos Motor parado
    enviar a todos PULSADOR DERECHA OFF
    enviar a todos PULSADOR IZQUIERDA OFF
    enviar a todos STOP
  si no
    analógico 5 valor 0
    digital 10 apagado
    enviar a todos Motor parado
  
```

```

al presionar
analógico 6 valor 0
fijar giro izquierda a 0
fijar IZQUIERDA a 0
digital 12 apagado
enviar a todos Motor parado

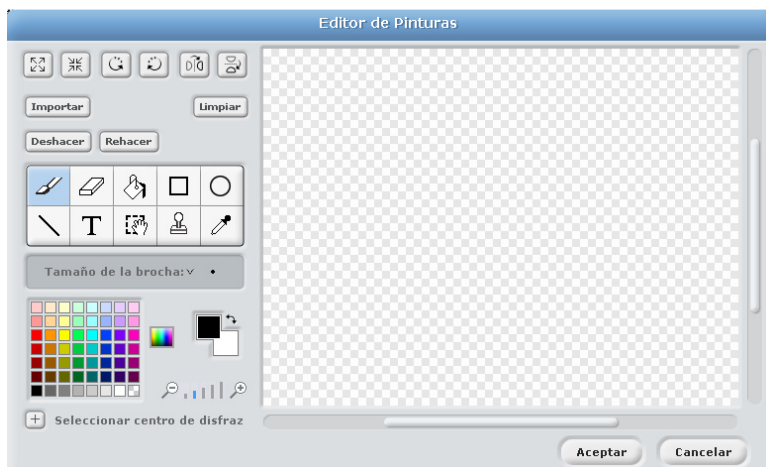
por siempre
si
  valor del sensor Analógico > 1000 y no sensor Digital3 presionado? o IZQUIERDA = 1 y giro derecha = 0
  fijar STOP a 0
  enviar a todos PULS DER ON
  enviar a todos PULS IZQ ON
  repetir hasta que sensor Digital3 presionado? o STOP = 1
  fijar giro izquierda a 1
  fijar IZQUIERDA a 1
  analógico 6 valor velocidad
  digital 12 encendido
  enviar a todos Motor izquierda
  fijar giro izquierda a 0
  fijar IZQUIERDA a 0
  analógico 6 valor 0
  digital 12 apagado
  enviar a todos Motor parado
  enviar a todos PULSADOR DERECHA OFF
  enviar a todos PULSADOR IZQUIERDA OFF
  enviar a todos STOP
si no
  analógico 6 valor 0
  digital 12 apagado
  enviar a todos Motor parado
  
```

Para la monitorización hay que tener claro que S4A trabaja con un escenario. Que nosotros podemos crear o importar todos los objetos que nos sean necesarios y que colocaremos sobre el escenario. Que cada objeto puede tener diferentes disfraces, es decir, diferentes formas de visualizarse según circunstancias. Y que cada objeto se puede programar de forma individual.





La creación de objetos es muy sencilla. Se basa en un editor de dibujo muy básico y con herramientas muy sencillas. Los alumnos no tendrán ningún problema para manejarlo con soltura.



Si queremos monitorizar el motor, lo dibujaremos de forma muy sencilla, círculo con una M mayúscula en el centro y relleno de color rojo para representar el estado de motor apagado. Al aceptar veremos que se ha creado un nuevo objeto al que le pondremos nombre (Motor). De este objeto tendremos, como de cualquier objeto, las pestañas programas, disfraces y sonidos.

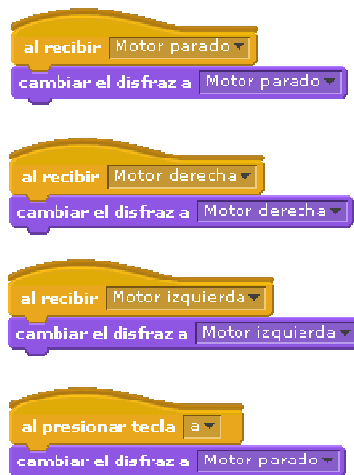
En la pestaña disfraces, copiaremos el disfraz original para duplicarlo dos veces. Editaremos uno de los disfraces duplicados, para que su apariencia sea la del motor girando en sentido antihorario, rellenándolo de color verde y con una flecha que indica ese sentido de giro y aceptaremos para guardar los cambios. Editaremos el otro disfraz duplicado para representar el sentido de giro horario, rellenándolo de color amarillo y con una flecha que indica ese sentido de giro.



A continuación pondremos nombre a los disfraces. Motor parado para el disfraz que representa ese estado, motor izquierda para el disfraz que representa sentido de giro antihorario y motor derecha para el disfraz que representa sentido de giro horario.

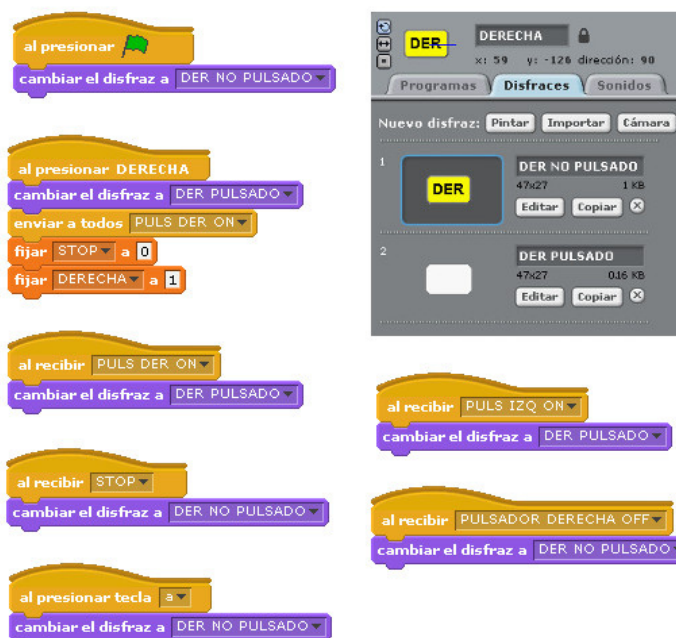
La programación del objeto Arduino 1 será la adecuada para controlar el funcionamiento del motor. Desde la programación del objeto Arduino 1 enviaremos

mensajes al resto de objetos con las acciones a realizar. Programaremos el objeto motor para monitorizarlo, cambiando su disfraz cuando reciba los correspondientes mensajes. La programación del objeto motor es la siguiente:



En pantalla también se realizará la monitorización de los 3 pulsadores virtuales para poner en marcha el motor en sentido horario, en sentido antihorario y para poder pararlo. Como podemos observar, utilizamos el bloque “al presionar \_\_\_” para poner una variable a 1 y con esa variable actuar en la programación del bloque Arduino 1. Con mensajes interactuamos con los otros objetos. La programación de los 4 objetos en S4A es la siguiente:

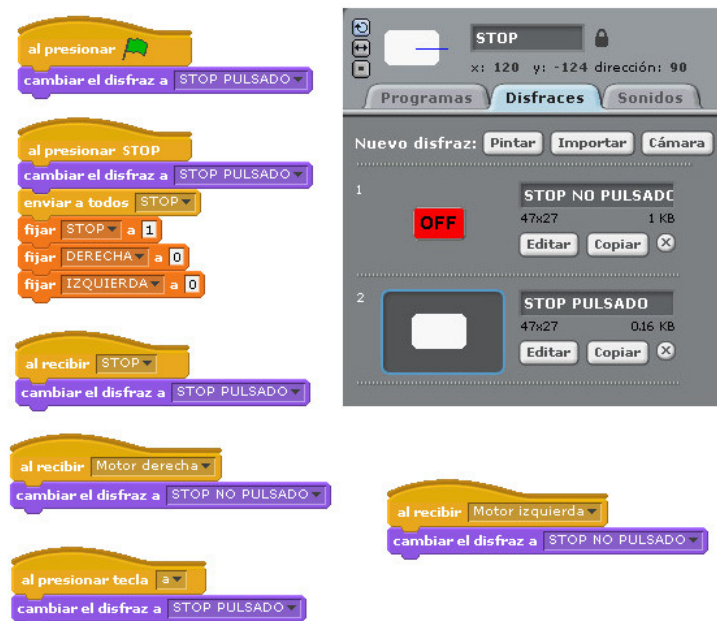
Pulsador virtual de marcha en sentido horario:



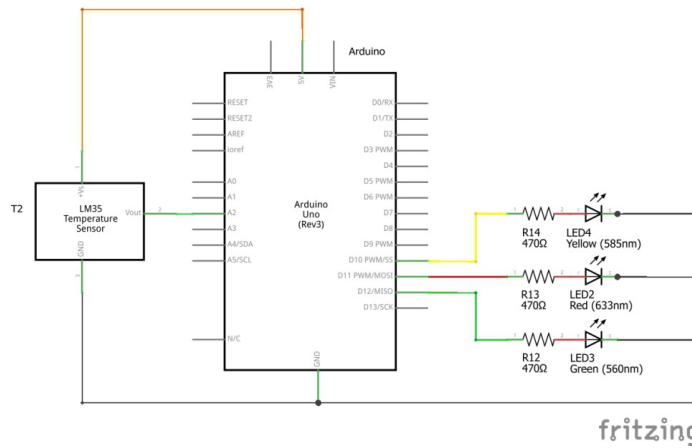
Pulsador virtual de marcha en sentido antihorario:



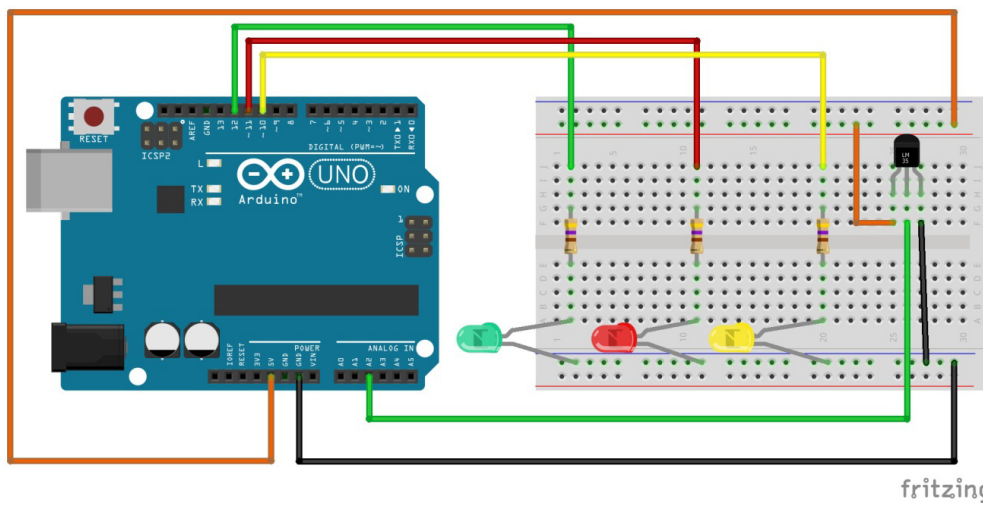
Pulsador virtual de paro:



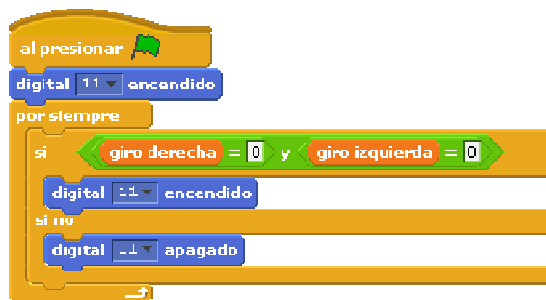
Desde la programación en el elemento Arduino 1, en los bucles que controlan el giro a derecha e izquierda, podemos observar que también controlamos las salidas digitales 10 y 12 respectivamente, las cuales utilizamos para controlar los diodos indicadores del sentido de giro. En el siguiente esquema funcional podemos observar la conexión de estos diodos junto con el diodo indicador del motor parado y el sensor de temperatura LM35 para usarlo como termómetro digital con indicación de la temperatura en pantalla.



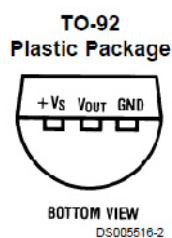
El esquema de conexionado:



La programación en S4A del diodo rojo indicador de que el motor está parado:



El esquema de patillaje del sensor de temperatura LM35 y su ecuación característica son los siguientes <sup>18</sup> :



$$U_{\text{SENSOR}} = 10\text{mV}/^{\circ}\text{C} \cdot T (^{\circ}\text{C}) \text{ (de } 2^{\circ}\text{C a } 150^{\circ}\text{C)}$$

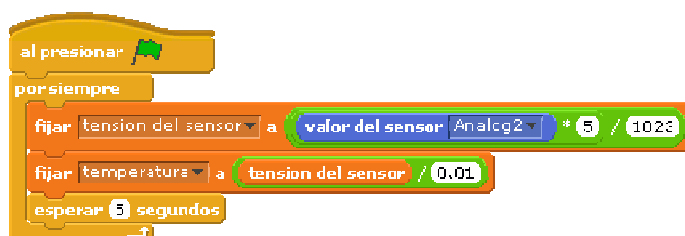
Para la programación hay que tener en cuenta una serie de cosas. La primera es que el sensor LM35 nos devuelve una tensión proporcional a la temperatura, que al introducirla a través de una entrada analógica es digitalizada con un valor entre 0 y 1023. Como necesitamos conocer el valor de tensión que nos devuelve, deberemos de calcularlo con la siguiente regla de tres:

$$\left. \begin{array}{l} 1023 \text{ ————— } 5V \\ U_{\text{SENSOR}} \text{ ————— } Xv \end{array} \right\} Xv = \frac{U_{\text{SENSOR}} \cdot 5V}{1023}$$

Una vez conocido el valor de la tensión en voltios despejaremos de la ecuación de funcionamiento del sensor LM35, adaptada para trabajar en voltios:

$$Xv = 0,01V/^{\circ}C \cdot T (^{\circ}C) \text{ (de } 2^{\circ}C \text{ a } 150^{\circ}C) \rightarrow T (^{\circ}C) = \frac{Xv}{0,01v/^{\circ}C}$$

Vistos los cálculos, la programación en S4A de la entrada analógica A2 para mostrar la temperatura en pantalla es la siguiente:

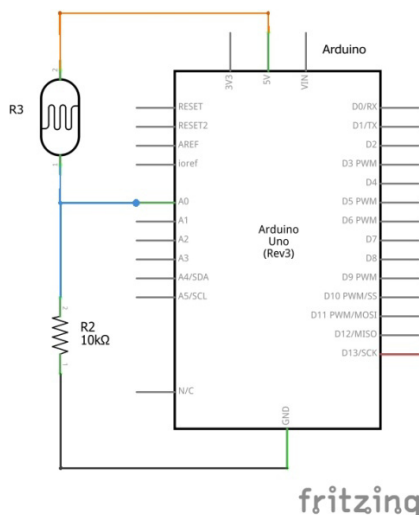


Observamos que en este caso las lecturas del sensor no las realizamos de forma continua, sino que las realizamos cada 5 segundos. Esto es suficiente ya que los cambios en la temperatura ambiente no son bruscos. Muchas estaciones meteorológicas realizan la lectura de los sensores cada 30 segundos.

El siguiente paso es integrar la práctica 41, control del accionamiento, en función de la luminosidad ambiental, de un relé, de forma indirecta a través de un transistor NPN BD135, con posibilidad de elegir y modificar el nivel de referencia de luminosidad.

Una LDR (light dependent resistor), es una resistencia cuyo valor varía con la intensidad luminosa que incide sobre ella. De forma que si la intensidad luminosa que incide sobre ella es muy alta (día) el valor de su resistencia es muy bajo. Y si la intensidad luminosa es muy baja (noche) el valor de su resistencia es muy alto.

Trabajando con el sistema Pull-down o lógica directa:



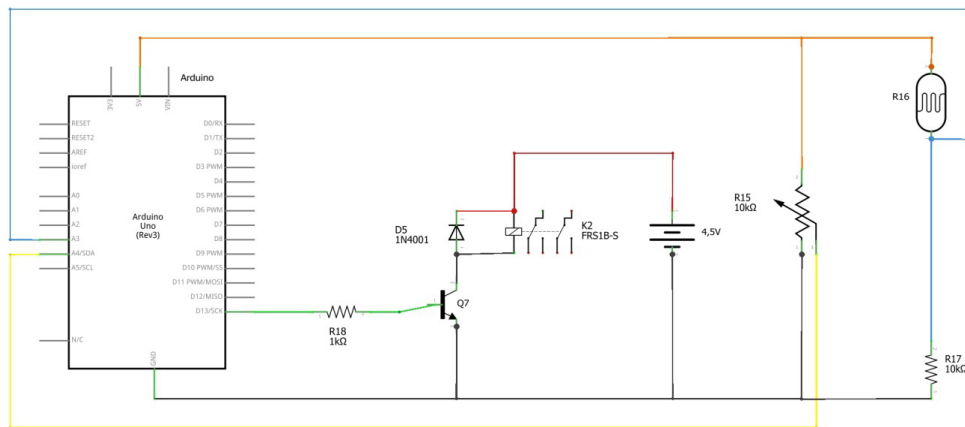
$$U_{A0} = U \cdot \frac{R_2}{R_{LDR} + R_2}$$

Día:  $R_{LDR} \downarrow \rightarrow U_{A0} \uparrow \approx U = 5V$

Noche:  $R_{LDR} \uparrow \rightarrow U_{A0} \downarrow \approx 0V$

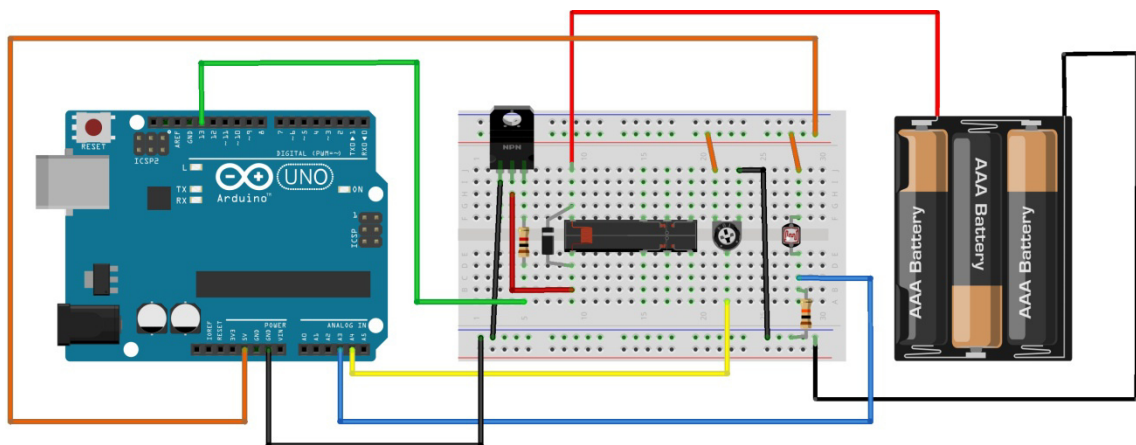
Con este sistema si observamos, de día mucha luminosidad y mucha tensión en la entrada (lógica directa). Conforme anochece la tensión de la entrada disminuye.

El esquema eléctrico funcional de esta parte de la práctica es el siguiente:



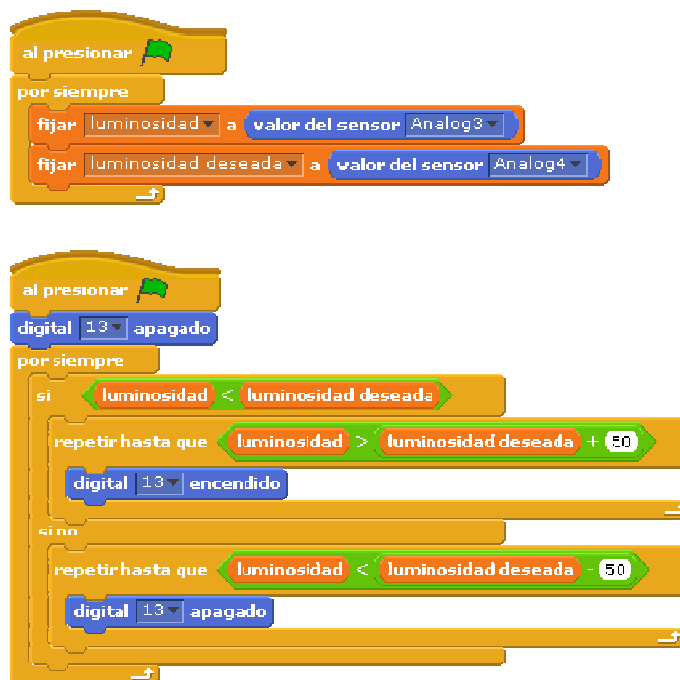
fritzing

El esquema de montaje y conexionado:



fritzing

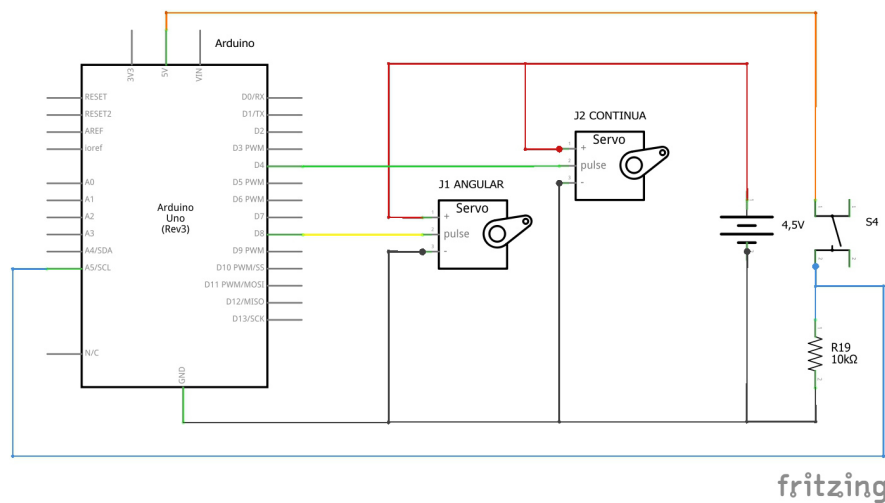
La programación de esta parte de la práctica en S4A:



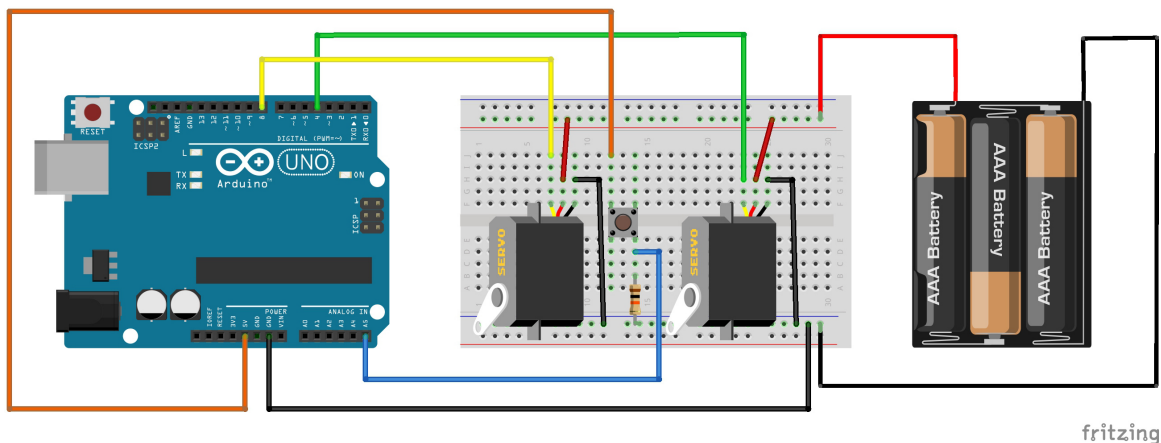
Por último queda añadir la parte de las prácticas 71 y 72 para controlar un servomotor angular desde un deslizador en pantalla y un servomotor de rotación continua.

El primer paso es entender el funcionamiento de un servomotor. En este caso, a diferencia del resto del libro, en el que hemos utilizado el libro digital de <http://tecno12-18.com><sup>12</sup>, al carecer de la explicación de este motor, vamos a recurrir a la página <http://www.areatecnologia.com/electricidad/servomotor.html><sup>13</sup> y a un vídeo de youtube sobre la <https://www.youtube.com/watch?v=84mxq41zdwE><sup>14</sup>, con lo que los alumnos pueden trabajar de forma autónoma y entender perfectamente el funcionamiento y principios de control de este tipo de motores.

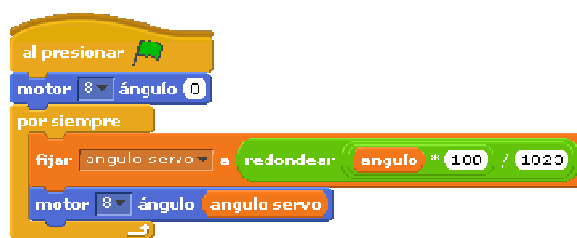
El esquema funcional eléctrico es el siguiente:



El esquema de montaje y conexionado:



Y por último, la programación en S4A:





También programaremos un último bucle que activaremos desde el teclado, para poner a cero todas las salidas y detener todos los bucles del programa.

