

SEÑALIZACIÓN EN BICICLETAS

por Aurelio Gallardo

Mayo de 2014

II FERIA DE LA CIENCIA EN LA CALLE DE JEREZ DE LA FRONTERA



Señalización en Bicicletas by Aurelio Gallardo Rodríguez, 31667329D is licensed under a Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional License.

Índice General

1. INTRODUCCIÓN	1
2. DESCRIPCIÓN DE LA NECESIDAD	2
3. OBJETIVOS	2
4. ELEMENTOS DE HARDWARE	3
4.1. BLOQUE DELANTERO DE LUCES LARGAS Y CORTAS	3
4.1.1. DETALLANDO LA CONSTRUCCIÓN DEL MÓDULO	4
4.2. BLOQUE TRASERO LUCES DE FRENO E INTERMITENTES	5
4.2.1. DETALLANDO LA CONSTRUCCIÓN DEL MÓDULO	5
4.3. INTERMITENTES DELANTEROS	7
4.4. PULSADORES QUE ACTIVAN LOS INTERMITENTES	7
4.5. FINAL DE CARRERA DEL FRENO	7
4.6. CAJA CON LA PLACA PRINCIPAL Y CONEXIONADO DE CONTROL	8
4.7. UNIDAD DE ALIMENTACIÓN	9
4.7.1. CÁLCULOS DE CONSUMO	9
5. ELECTRÓNICA	11
5.1. CIRCUITO CON TRANSISTOR PARA ACTIVAR LOS LEDs TRASEROS	11
5.2. CIRCUITO PULSADOR O FIN DE CARRERA: INPUT DIGITAL DE ARDUINO	12
5.3. ACTIVACIÓN SIMPLE DE UN LED: OUTPUT DIGITAL DE ARDUINO	12
6. PROGRAMACIÓN Y FUNCIONAMIENTO	12
6.1. DEFINICIÓN DE VARIABLES: PINES DE CONEXIÓN	12
6.2. DEFINICIÓN DE VARIABLES: ESTADOS	13
6.3. DEFINICIÓN DE VARIABLES: TEMPORALES	13
6.4. SETUP	14
6.5. BUCLE PRINCIPAL LOOP	14
6.5.1. CONTAMOS EL TIEMPO	14
6.5.2. BUCLE DEL FRENO	14
6.5.3. BUCLE DE LAS LUCES CORTAS Y LARGAS	14
6.5.4. INTERMITENTES IZQUIERDO Y DERECHO	15
6.5.5. LUces DE EMERGENCIA	17
6.5.6. DISMINUYO/AUMENTO LA CADENCIA DE LAS LUces DE EMERGENCIA	18
7. COLABORADORES	18
REFERENCIAS	19
A. LISTADO COMPLETO DEL PROGRAMA	20



1. Introducción

Es un hecho evidente que en los últimos tiempos cada vez más personas usan la bicicleta como medio de transporte o como deporte. Muchas son las causas y muchas las ventajas de su uso. Entre las causas de su auge cabe destacar la mayor concienciación de un sector de la población por temas ambientales o de salud, la preocupación del gasto en transporte o el aumento de los tiempos de tránsito en los desplazamientos urbanos habituales. Y es que de hecho, las bicicletas en las ciudades modernas, permiten desplazamientos más rápidos en distancias a escala urbana, al ser un medio de transporte que puede sortear con éxito las congestiones del tráfico automovilístico; además, más allá de este beneficio inmediato y tangible, encontramos otros no menos importantes: hacemos ejercicio físico, no contaminamos y ahorraremos en otros tipos de transporte.

Como viene recogido en el Plan Andaluz de la Bicicleta 2014-2020¹, se considera importante el fomento de este medio de transporte en los próximos años, como forma de trasladarse a los centros de trabajo o estudios, y dando importancia también a las prácticas deportivas, de ocio y su potencial turístico. En dicho plan se tiene previsto la construcción de carriles urbanos e interurbanos que interconecten no sólo las distintas zonas urbanas de las principales ciudades andaluzas, sino también numerosas poblaciones de nuestra tierra entre sí.

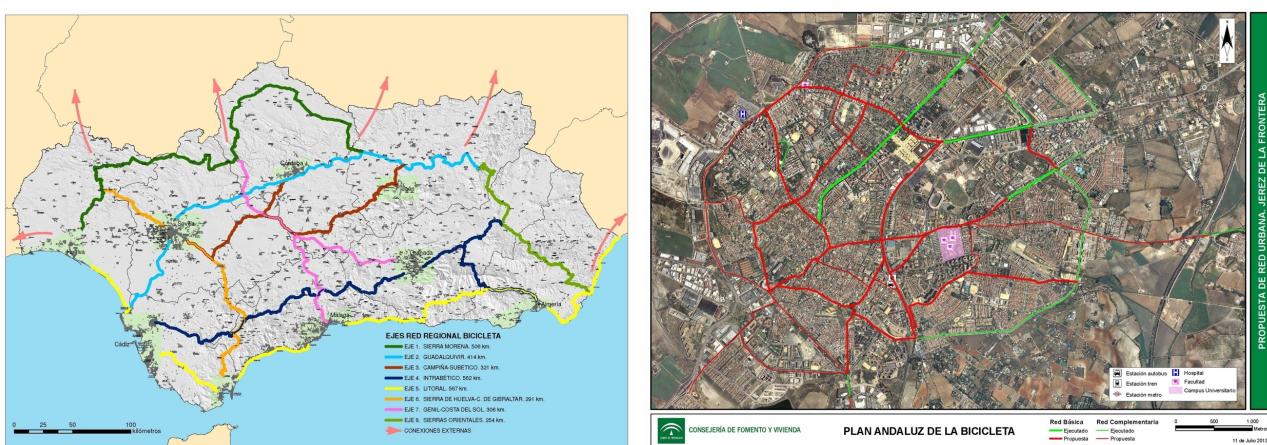


FIGURA 1: MAPAS DE CARRILES BICIS INTERURBANOS Y URBANOS EN JEREZ

Queda pues claro que la bicicleta, como medio de transporte, está adquiriendo un mayor peso específico en nuestra vida diaria y, además, cuenta con el apoyo de nuestras instituciones a nivel autonómico. Por otra parte, se reconoce de forma implícita la necesidad de mejorar o adecuar las vías de circulación; en efecto, el usuario o usuaria de la bicicleta en las ciudades y carreteras actuales es un conductor frágil y vulnerable, expuesto a frecuentes situaciones de peligro y teniendo que sortear obstáculos que otros usuarios de las vías no tienen, sobre todo cuando no se cuenta con vías específicas para uso de bicicletas.

El o la ciclista preocupado por su seguridad, se encuentra ante una disyuntiva difícil: o bien opta por usar las vías destinadas al tráfico rodado, tal como exigen las normas de circulación, arriesgándose a que los usuarios de los vehículos motorizados que circulan por la vía no cometan equivocaciones y respeten a su vez las reglas, o bien opta por incumplir la normativa y utiliza las zonas peatonales, más seguras para el ciclista.

En este último caso, no es menos cierto que el ciclista actualmente y bajo nuestra legislación pierde la razón legal, y es responsable de su conducción y de los inconvenientes que pudiesen surgir con ella. De hecho, no podemos obviar que muchos viandantes se quejan de que en las zonas peatonales hay ciclistas que molestan al paso y, a su vez, pueden provocar accidentes a los que caminan por ellas. Siendo cierto que el o la ciclista suele ser una persona respetuosa, no todo el mundo usa las vías con cuidado y atención, lo cual puede causar molestias, sobre todo si no tiene derecho a circular por ellas.

Por todo lo expuesto anteriormente, podemos concluir que la utilización de la bicicleta, sobre todo en las ciudades, necesita de nuestro respaldo y apoyo además de el de las instituciones. En este trabajo, detectamos una necesidad

¹http://www.juntadeandalucia.es/fomentoyvivienda/portal-web/web/areas/transportes/plan_bici/texto/83770544-954a-11e2-8f48-a9c3a8bb53aa



específica relacionada con el mundo del ciclismo y hemos intentado satisfacerla con el objeto de estudio de nuestra asignatura de Proyecto Integrado “Programación y Electrónica Digital”: la plataforma de electrónica abierta ARDUINO.

2. Descripción de la Necesidad

En nuestra normativa de tráfico², hay todo un capítulo, el décimo, dedicado a la señalización de vehículos. Respecto a las bicicletas sólo se obliga, según el artículo 98.3, a llevar elementos reflectantes en las mismas y en prendas que pueda llevar el ciclista. El resto de vehículos tienen elementos visuales de seguridad activa tanto para ver como para ser vistos, en circunstancias de luz diurna, o en conducción nocturna o de baja visibilidad, y la obligación/posibilidad de señalizar determinadas maniobras como frenar, girar, dar marcha atrás o parar de emergencia.

Así que... ¿por qué no dotar a una bicicleta de algunos de estos elementos de seguridad activa de señalización?

¿Acaso no es deseable que un ciclista pueda mejorar su visibilidad en conducción nocturna? ¿No sería útil que pudiese indicar de alguna forma a los otros usuarios de la vía su presencia? ¿No sería conveniente que pudiese señalizar algunas de sus maniobras como girar o frenar? O si se me detengo de emergencia en un arcén o similar... ¿no sería bueno indicar esa circunstancia con un juego de luces que parpadeen?

3. Objetivos

Los objetivos de nuestro trabajo, por tanto, serán los siguientes:

1. Dotar a nuestra bicicleta de medios de señalización e indicación luminosa: intermitentes izquierdos y derechos (delanteros y traseros), luz de freno y luces de emergencia.
2. Dotar a nuestra bicicleta de medios de iluminación: luces cortas y largas.
3. Realizar el control automático (SOFTWARE) de dicho sistema mediante ARDUINO.
4. Que los elementos de HARDWARE cumplan las siguientes condiciones:
 - a) No molesten los movimientos de la bicicleta.
 - b) Estén bien sujetos y no sean voluminosos. Que no impidan el uso de otros complementos o elementos de la bicicleta (botella de agua, pitón, transportín,...)
 - c) Los cables estén recogidos.
 - d) Los pulsadores o elementos de activación sean accesibles para el ciclista sin molestar la conducción.
5. Que los elementos de SOFTWARE cumplan las siguientes condiciones:
 - a) Respondan a las funcionalidades exigidas.
 - b) La activación de un elemento del sistema interfiera lo menos posible en el estado de funcionamiento de otro elemento cualquiera.
 - c) El control del sistema se realice con el menor gasto de energía posible³.

²Reglamento general de tráfico:

http://www.taxiasturias.com/html/3868_FEDERACION_ASTURIANA_SINDICAL_DEL_TAXI/files/5872_reglamento%20general%20de%20traf

³Un objetivo secundario que se ha planteado al finalizar este trabajo es controlar, de alguna forma, el gasto o consumo del dispositivo cuando se usen pilas. Sin embargo, el uso de las pilas puede ser sustituido por baterías recargables no demasiado caras.



4. Elementos de hardware

Antes de explicar la composición de los distintos elementos de iluminación y control que conforman el sistema hay que puntualizar que este trabajo no ha formado parte del Proyecto Integrado en el que ha sido desarrollado. El diseño e implementación del Hardware no entra dentro del currículum de la asignatura, sino tan sólo la parte de Programación en el entorno ARDUINO. Agradezco a los alumnos Luis y Juanma, de 2º de Bac. , y Christian de 2º ESO, su ayuda y colaboración en estas tareas, incluso sacando tiempo de los recreos o ratos libres que hayamos tenido.

Cabe también destacar que la construcción del prototipo ha tropezado con ciertas dificultades; algunas de diseño, otras de materiales y componentes. El prototipo es susceptible de numerosas mejoras, que se indicarán en cada apartado.

Entre los elementos hardware del sistema cabe destacar los siguientes:

- ✓ Bloque delantero de luces largas y cortas.
- ✓ Bloque trasero de luces de freno e intermitentes.
- ✓ Intermitentes delanteros: LEDs.
- ✓ Pulsadores que activan los intermitentes.
- ✓ Final de carrera del freno.
- ✓ Caja con la placa principal y conexionado de control.
- ✓ Unidad de alimentación.

4.1. Bloque delantero de luces largas y cortas

Uno de los más difíciles de implementar, es un elemento que se posiciona en el manillar y contiene el juego de luces cortas (cuatro leds blancos de alta luminosidad), el juego de luces largas (otros cuatro leds), el pulsador que activa estas luces y todo el conexionado previo del sistema de señalización delantero más el conexionado de los actuadores (pulsadores, fin de carrera...).



FIGURA 2: MÓDULO DELANTERO DEL SISTEMA

Se construye con un tubo de PVC Ø20, usado para la canalización de cables de red, pintado de negro con pintura de spray. Las sujeciones se consiguen con abrazaderas atornilladas con dos tornillos M6. A él acceden, por la izquierda, seis cables correspondientes a los intermitentes delanteros izquierdo y derecho, y al pulsador del intermitente izquierdo, y por la derecha, cuatro cables, correspondientes al pulsador del intermitente derecho y al fin de carrera que detecta que se ha activado el freno trasero de la bicicleta. Por su parte inferior central sale un conjunto de diez cables con ocho señales de control y activación, más la señal de alimentación y GND, que se conectan a un antiguo cable de comunicación en serie (9 PIN D-SUB⁴). Las conexiones y soldaduras en este cable están protegidos con funda pintada de negro y el cable se enfunda con cañitas de color negro.

⁴<http://todohard.awardspace.com/docs/ConecotorCom9/>



4.1.1. Detallando la construcción del módulo

Tras un mecanizado básico en que se corta el tubo, se practican los orificios de los leds, el orificio para alojar el pulsador que activa las luces, el del cableado de salida y los necesarios para atornillar el módulo a las abrazaderas, se sueldan los dos grupos de 4 LEDs en paralelo usando una plantilla con los orificios a la misma distancia que los practicados en el tubo. Se pone especial cuidado en que la soldadura y el cableado ocupe el mínimo espacio pues después tienen que caber en el reducido lugar donde se alojarán. Se suelda, además, una resistencia de 100Ω al negativo de los LEDs y se pega un palito de madera (pinchitos) al conjunto con silicona. Usando el palito, se colocan los LEDs en su lugar dentro del tubo y, cuando estén colocados, se rompe el palito de madera por dentro cuidando que no sobresalga por la apertura.⁵

Los comunes de los grupos de LEDs se unen entre sí y a un cable que posteriormente se soldará al GND⁶ del conjunto. Los cables de los positivos son de señal, y se hacen pasar por el orificio de salida, para conectarlos a los pines correspondientes de ARDUINO.



FIGURA 4: PULSADOR. IMAGEN DE [SHOPTRONICA.COM](#) Y DSUB9-MACHO

Se instala un pulsador NA, en el orificio practicado previamente, soldándole dos cables a cada terminal. Los cables usados son del tipo plano-paralelo de las conexiones IDE de los ordenadores: flexibles, y fáciles de pelar y de soldar, ocupando poco espacio. A continuación se colocan los tornillos y la parte superior de las abrazaderas, en previsión que el número de elementos dentro del tubo aumente y no sea fácil colocar los tornillos.

En el tapón que se usará para tapar la parte derecha del tubo, se practica un orificio, se le hace pasar una cañita protectora y se pega con silicona. Por dicha cañita, se hacen pasar cuatro cables, soldándole en los extremos exteriores del tubo conexiones hembras tipo PCB⁷. En la parte interior se sueldan las conexiones a los cables que acabamos de instalar en el tapón, que corresponden al intermitente derecho, al final de carrera del freno y al pulsador de luces largas/cortas, con sus correspondientes resistencias de $10K\Omega$ a tierra, pasando tres cables de señal por el orificio de salida.

En el lado izquierdo se opera de forma idéntica, esta vez para las señales de intermitentes delanteros izquierdo/derecho y para el pulsador del intermitente izquierdo (resistencia $10K\Omega$). Se hacen pasar los tres cables de señal a través del orificio de salida.

Se añaden los dos cables de GND y Vcc (5V) al orificio de salida, se comprueba el funcionamiento y se cierra el conjunto, cerrando los tapones. Se enfundan cañitas protectoras a los cables de salida, y se sueldan los cables a un cable reciclado DSUB-9 macho, capaz de albergar 9 señales más GND, en total las diez que necesitamos. Se comprueba el conjunto, se enfunda un tubo de goma (previamente pintado de negro) alrededor de las conexiones y se asegura la estanqueidad con cinta aislante y presillas (5).

A mejorar:

- ✓ Emplear abrazaderas y tornillos que separen más la pieza del manillar permitiría poder posicionar mejor las luces de lo que ahora están.
- ✓ Tapar con tapones de goma los orificios practicados al tubo para poder introducir el pulsador y los tornillos de las abrazaderas.

⁵ Esta operación resultó ser más dificultosa de lo que supuse. El primer conjunto de LEDs estorbó bastante la colocación del segundo con lo que sería mejor que ambas filas hubiesen estado más separadas. Por otra parte, tuve que meter un redondo para poder presionar los LEDs ya colocados de forma que sobresaliesen del tubo lo máximo posible, no consiguiéndolo del todo con los LEDs del segundo grupo.

⁶GND: ground, tierra

⁷ Sería deseable usar otro tipo de conexiones más recogidas, pero aproveché lo que ya tenía. En este trabajo he usado mucho material reciclado o de uso común en el taller

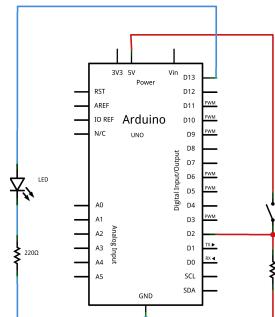


FIGURA 3: CONEXIONADO TÍPICO DE UN LED Y DE UN ACTUADOR (PULSADOR) USANDO ARDUINO



FIGURA 5: (IZQ) SOLDANDO LADO IZQUIERDO. (CENTRO) CONEXIONES AL CABLE DSUB9. (DCHA) COMPROBANDO CABLES.



4.2. Bloque trasero Luces de Freno e Intermitentes

El bloque trasero contiene las luces de freno y los intermitentes de la parte de atrás. Se construye con varias capas de madera contrachapada 4mm, dejando hueco entre ellas para incorporar los distintos elementos: un LED de alta luminosidad 5mm para cada intermitente y cinco que conforman la luz del freno.

Interiormente, y aunque la luminosidad en el rango usado de tensiones es bastante aceptable, he optado por activar los LEDs del freno mediante un transistor NPN 2N2222, lo que permite alimentarlos directamente desde la pila. El conexionado se termina con un RJ45 hembra del que sobresalen 5 señales: GND, Vin, Señal luz de freno, y señal de cada intermitente. Cada intermitente está conectado en paralelo con su par delantero.

4.2.1. Detallando la construcción del módulo

En primer lugar se recortan las distintas capas usando plantillas (8) ; éstas irán superpuestas (y atravesadas y unidas mediante dos tornillos Ø4 pasantes y cogidos con tuercas al transportín, al que previamente le he desmontado el catadióptrico trasero) desde en la que va alojado el papel de celofán rojo hasta la tapa final. En total, siete capas. Cada capa va lijada, pintada con imprimación para madera y posteriormente de negro en la zona visible.

FIGURA 6: RJ45 HEMBRA



En la segunda capa se colocan los LEDs del freno en sus orificios y se sueldan los positivos entre sí, así como los negativos, de forma que el cableado adopta la forma de la configuración definitiva de los LEDs. Se colocan y sueldan los cables a los LEDs de los intermitentes, y resistencias de 100Ω en sus negativos. Se van añadiendo capas, y en el hueco que presentan, se coloca una plaquita previamente soldada con el transistor y el conexionado a los LEDs del freno. Para terminar, y colocadas todas las capas excepto la tapa, se coloca un RJ45 hembra (6) y se conectan a ella las 5 señales (7).

FIGURA 7: DISTINTAS FASES DEL MONTAJE DEL MÓDULO TRASERO



Una vez comprobado el funcionamiento, se encolan entre sí las distintas capas, se cierran y aprietan ayudándonos de los tornillos y de algunos gatos, se encola las ranuras entre las distintas capas retirando el sobrante y se repasa la pintura negra. La unión es de tal forma que los tornillos pasantes pueden desmontarse (8).

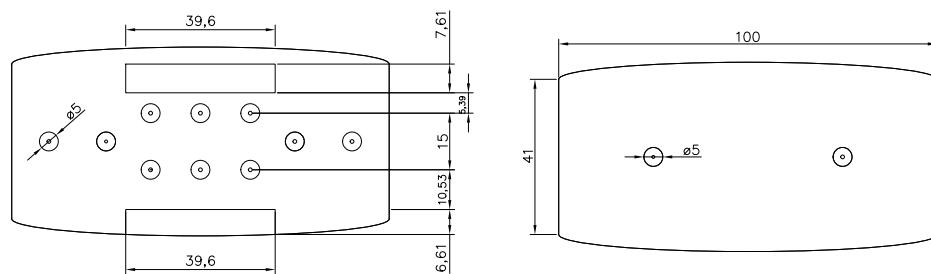
Al final, se introduce papel de celofán rojo en los huecos de las luces y se pega una capa protectora de vinilo transparente, del usado en los dossieres de fotocopiadoras.



A mejorar

- ✓ El uso de papel celofán rojo o naranja no es buena opción de catadióptrico. Hay que buscar otra solución.
- ✓ Un error de precipitación fue pegar el vinilo a la primera capa con pegamento loctite. La unión quedó fija, pero el aspecto no fue el esperado. Sin embargo no hubo ningún problema de funcionalidad.
- ✓ Los RJ45 hembras (6) están diseñados para soldar a una placa, no elementos “al aire”. Es un trabajo delicado soldar cables directamente a sus conexiones, ya que un exceso de calor puede provocar que sus contactos metálicos fundan el plástico al que van cogidos. Posteriormente, me di cuenta que es mejor soldar a los cables a tiras de zócalos hembras y éstos, al tener medidas estándar en placa, conectarlos por presión a los contactos del RJ45. Por cierto, y aunque no son caros, hay que mandarlos a pedir en tiendas online. No los pude conseguir en ninguna tienda de la zona.

FIGURA 8: FIGURAS BASE PARA LAS DISTINTAS CAPAS





4.3. Intermitentes Delanteros

Los intermitentes delanteros se implementan con dos LEDs de alta luminosidad rojos, conectados sus negativos a resistencias de 100Ω . Como el manillar es un tubo de Ø20, simplemente con dos tapones parecidos a los ya usados en el tubo de luces principales se practica en cada uno un orificio para el LED, al que se fija con silicona. Los cables, largos, se hacen pasar por el interior y se sacan por un pequeño orificio practicado en el manillar. Se fijan los tapones una vez comprobado el funcionamiento, y se sueldan conectores macho tipo PCB.



FIGURA 9: LEDs INTERMITENTES DELANTEROS

4.4. Pulsadores que activan los intermitentes

Los pulsadores (10) fueron objeto de varias elucubraciones y, al final, decidí poner algunos normales que se ajustasen al hueco entre los cambios y el manillar. Una funda de goma hace que entren en el hueco a presión y su movimiento sea mínimo. Sin embargo, es posible que este aspecto quede más estético si se pudiesen implementar de otra forma. Simplemente llevan dos cables soldados y éstos a una conexión PCB macho, tanto en el lado izquierdo como en el derecho.



FIGURA 10: PULSADORES DELANTEROS

4.5. Final de carrera del freno

Otro quebradero de cabeza fue la implementación del dispositivo que detectara cuando el freno (derecho) se activaba (13). Le dimos vueltas a la posibilidad de colocar algo en la palanca del freno que se encuentra en el cambio, o bien en algún lado del tensor o en la propia horquilla del freno (la que presiona las pastillas). Tres posibilidades parecieron ciertas:



usar un sensor de efecto Hall, un interruptor magnético reed (ambas acercando/alejando un imán a ambos dispositivos) o un final de carrera. Pronto nos dimos cuenta que lo mejor, tanto por su posición dentro de la bicicleta, como por sencillez, era usar el movimiento lineal del tensor, y en él, fijar uno de estos tres elementos.

El sensor de efecto Hall (11) no lo llegué a encargar simplemente porque tenía interruptores reed y finales de carrera, y quería aprovecharlos. Se puede comprar en tiendas como shoptronica.com o en cooking-hacks.com por alrededor de un euro. Existe una amplia documentación en la red de cómo usar y activar un sensor de este tipo con ARDUINO⁸.

FIGURA 11: SENSOR EFECTO HALL

⁸<http://minibots.wordpress.com/2013/11/05/utilizacion-de-un-sensor-de-efecto-hall/>



Me decidí en un principio por el interruptor reed (12) ([shoptronica.com reed](http://shoptronica.com/reed)). Lo uní firmemente al tensor con presillas de manera que se moviese con él y, al presionar el freno, se acercara a un imán. Efectivamente, tras un ajuste de las distancias, el interruptor reed se activaba, pero presentaba histéresis, es decir, permanecía imantado al liberar el freno y alejarse del imán. Para que se desactivase, había que alejarlo más del imán que el recorrido que permitía el tensor. Desde luego, era un gran inconveniente⁹.



FIGURA 12: INTERRUPTOR REED

Así que la siguiente opción fue usar un final de carrera. Tras varias pruebas, resultó muy satisfactorio el funcionamiento cuando lo fijé (se le pega por detrás un imán de neodimio y queda muy fijo al tubo de acero del cuadro) al principio del tensor. Sólo se necesita hacer solidario al interruptor algún elemento que hiciese de vástago que pudiese empujar la palanca del mismo. Tras darle algunas vueltas, y evitando dañar o modificar en demasiado el movimiento del tensor, una pieza de cobre de las fichas de empalme resulta perfecta. Una vez liberada de su carcasa de plástico, se le practica una ranura longitudinal para poder introducir el tensor dentro, y se atornilla firmemente a la distancia adecuada.



FIGURA 13: FIN DE CARRERA DEL FRENO

4.6. Caja con la placa principal y conexionado de control

Tanto el bloque delantero, de donde vienen el grueso de las conexiones, como el bloque trasero, tienen que comunicarse con ARDUINO. Un conector DSUB9 hembra que encontré de reciclaje en el taller me vino de perlas, pues tenía ya las conexiones soldadas a cables paralelos. Le hice un hueco en la caja de plástico que venden para alojar a ARDUINO¹⁰, lo fijé al mismo con pegamento de dos componentes, y los cables los soldé a pines de tiras de zócalos hembra(14). Tan sólo añadir que la señal GND hubo que extraerla de la protección metálica del conector, desarmándolo y soldándole un cable aparte.

FIGURA 14: TIRAS DE POSTES O ZÓCALOS HEMBRA



El siguiente paso sería alojar otro RJ45 hembra (6) , para poder tender el cable de red que conecte el bloque trasero con ARDUINO. Asimismo, y a esta conexión, se le añaden cables terminados en pines.

Se interconectan entre sí los zócalos de los intermitentes izquierdos y los derechos, y se conecta cada cable al pin correspondiente de ARDUINO (6.1).

FIGURA 15: BARREL JACK

También se le practica a la caja un orificio para poder alimentar la placa a través de una fuente externa, permitiendo el paso de un “barrel jack” (15). La alimentación externa en este modelo de ARDUINO UNO requiere la actualización del bootloader o la interconexión del pin 1 (RX) a tierra a través de una resistencia de $10K\Omega$ ¹¹.



Se fija la placa ARDUINO a la caja en su parte inferior, con sendos tornillos y tuercas Ø3. A esta parte inferior se le pega un trozo de madera al que se le han fijado tiras del velcro, de forma que sea fácil poder colocar la caja en el cuadro de la bici.

Y ya solo queda cerrar la caja, comprobar el funcionamiento y colocarla en su posición definitiva al cuadro de la bicicleta, conectando todos los elementos.

⁹Parece ser que el interruptor reed funciona mejor cuando el acercamiento al imán se realiza transversalmente a su eje, y no longitudinalmente, como lo usaba con el tensor. Pero cruzar perpendicularmente el interruptor respecto del tensor no era una opción.

¹⁰<http://www.cooking-hacks.com/shop/arduino/enclosures/plastic-case-for-small-project-88-65-36cm>

¹¹<http://arduino.cc/es/Guide/Troubleshooting#toc5>



FIGURA 16: CONTROL CENTRAL CON ARDUINO



4.7. Unidad de alimentación

ARDUINO (17) puede alimentarse o bien a través de un cable USB, que a su vez sirve para programarlo, o bien a través de una fuente externa. En la configuración actual se ha optado por una pila de 9V, y se ha construido un módulo con interruptor que permite el apagado o encendido de la unidad, y que se aloja en una de las bolsas de transporte que tiene la bicicleta, conectándose a ARDUINO a través de un cable con un barrel jack (15).

FIGURA 17: UNIDAD DE ALIMENTACIÓN Y CONECTADO



4.7.1. Cálculos de consumo

Es recomendable hacer algunas estimaciones de consumo de la unidad, sobre todo antes de lanzarse a la aventura de coger la bicicleta de noche. Para estos cálculos se ha tenido en cuenta lo siguiente

- ✓ El consumo principal sería el mantenimiento de las luces cortas y largas permanentemente encendidas. Se estudiarán los casos en el que la alimentación de los LEDs es constante y el caso en el que la activación se hace a través de una señal cuadrada con una frecuencia alta, lo que le disminuye algo el brillo pero permite la reducción del consumo (sin notar el parpadeo de los LEDs).
- ✓ La capacidad de una pila de 9V depende mucho de su tipología y marca, pero supondré que en principio entrega unos 600mAh¹², si es alcalina. También podrían usarse pilas de 9V recargables u otras pilas o baterías que no excedan de los 20V (recomendable que no excedan de 12V y que sean mayor de 6-7V)¹³
- ✓ El consumo de la placa ARDUINO no se ha medido, pero se supondrá de 100mAh¹⁴. El ajuste fino del consumo de la placa necesita de un estudio más amplio¹⁵.

¹²<http://www.afinidadelectrica.com.ar/articulo.php?IdArticulo=205>

¹³<http://es.wikipedia.org/wiki/Arduino>

¹⁴<http://forum.arduino.cc/index.php/topic,31252.0.html>

¹⁵<http://www.geekfactory.mx/arduino/mitos-de-arduino-las-baterias-duran-poco-para-alimentar-un-arduino/>



Estimaciones experimentales, para las luces cortas (4 LEDs alta luminosidad blancos en paralelo)

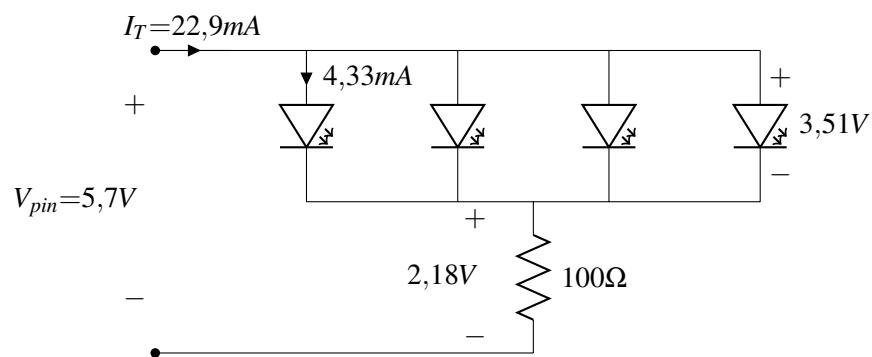
✓ Alimentación continua (18)

- ⇒ Consumo total: $I_T = 22,9mA$
- ⇒ Consumo por LED: $I_{LED} = 4,33mA$
- ⇒ Resistencia de carga: $R_{LED} = 99,2\Omega$
- ⇒ Vo pin ARDUINO: $V_o = 5,7V$
- ⇒ $V_{LED} = 3,51V$ y $V_R = 2,18V$

✓ Alimentación con pulsos

- ⇒ Cadencia 2ms
- ⇒ Período $T = 4ms$ y frecuencia $f = \frac{1}{T} = 250Hz$
- ⇒ Consumo total: $I_T = 10,5mA$
- ⇒ Consumo por LED: $I_{LED} = 2,1mA$
- ⇒ Vo pin ARDUINO: $V_o = 2,9V$
- ⇒ $V_{LED} = 1,8V$ y $V_R = 1,11V$

FIGURA 18: MEDIDAS EXPERIMENTALES, SEÑAL CONTINUA





Conclusión a los cálculos de consumo

- ✓ Para un consumo de señal continua, dos grupos de luces más el consumo de la placa ARDUINO dan un total de $Q \approx 146mAh$. Para una pila de 600mAh de carga, se puede estimar un tiempo de funcionamiento de aproximadamente 4,1 horas.
- ✓ Con una señal a pulsos, en las mismas condiciones, $Q \approx 121mAh$, lo que corresponde a un tiempo de funcionamiento de 4,96 horas.
- ✓ Es evidente que necesito o bien reducir el consumo de ARDUINO o bien usar elementos de mayor capacidad de carga. En este caso la solución de pilas recargables puede ser una opción. Otra opción, al tener ARDUINO la facilidad de alimentarse a través de una conexión USB, sería el uso de una batería externa. Con una de 2200mAh, por ejemplo, tendríamos una autonomía de entre 15 y 18,2 horas aproximadamente, números más que aceptables.

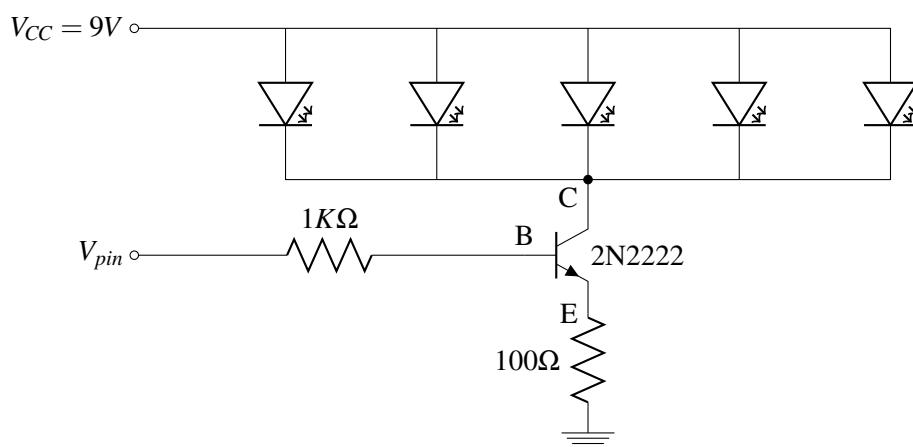
5. Electrónica

La electrónica de los circuitos es simple. A continuación se detallan los circuitos usados:

5.1. Circuito con transistor para activar los LEDs traseros

Mediante un transistor en saturación o corte, se activan o desactivan los LEDs del freno. Aunque no es necesario (experimentalmente una activación directa a ARDUINO resulta satisfactoria), es una forma de asegurarse el máximo brillo en los LEDs.

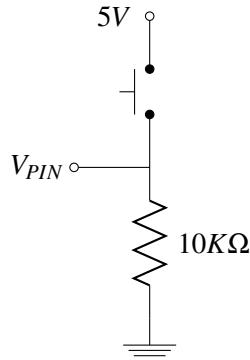
FIGURA 19: CIRCUITO ACTIVADOR CON TRANSISTOR NPN





5.2. Circuito pulsador o fin de carrera: INPUT DIGITAL de ARDUINO

FIGURA 20: PULSADOR EN ARDUINO



5.3. Activación simple de un LED: OUTPUT DIGITAL de ARDUINO

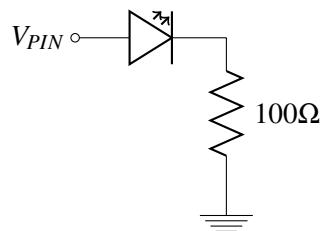


FIGURA 21: LED EN ARDUINO

6. Programación y funcionamiento

La otra parte importante del proyecto consiste en su SOFTWARE, que describiremos a continuación. Las bases del mismo ha constituido el núcleo central de la asignatura del **Proyecto Integrado de 2º BAC “Electrónica Digital y Programación”**, impartida en el **IES Seritium de Jerez de la Fra.** durante el curso 2013-2014. La funcionalidad básica fue desarrollada por los alumnos de esta asignatura, con mejoras en el código posteriormente realizadas por su profesor.

6.1. Definición de variables: pines de conexión

```
// declaracion de variables
// Freno
const int luzFreno = 3;
const int pulsadorFreno = 10;
// Intermitentes
const int intermitenteIzquierdo= 5;
const int intermitenteDerecho= 4;
const int pulsadorIzquierdo= 12;
const int pulsadorDerecho= 11;
// Luces cortas y largas
const int Largas = 6;
const int Cortas = 7;
const int pulsadorLuces = 13;
```



No hay mucho que explicar de este código. Simplemente se definen los pines que sirven para detectar (empiezan por “pulsador”) o activar las distintas señales. Se sigue, en general, la nomenclatura llamada *jorobaDeCamello*.

6.2. Definición de variables: estados

```
// variables de estado
int estadoFreno = LOW;
int estadoLuces = 0;
int estadoIIz = 0;
int estadoIDr = 0;
int estadoAlarma = 0; // controla si la bici est en posicin o no de emergencia.
int estadoEmergencia = 0; // controla el parpadeo de emergencia
```

Indican los distintos estados por los que pasará el programa. En cada estado, y según como se encuentre, funcionará de una u otra manera.

6.3. Definición de variables: temporales

```
// variables temporales
unsigned long t = 0;
unsigned long retardo = 10;
// segundos que dura el parpadeo de los intermitentes
unsigned long cadencia = 200;
// cadencia o frecuencia de parpadeo
unsigned long cadenciaEmergencia = 500;
// frecuencia de parpadeo en el estado de emergencia
// tiempo intermitente izquierdo
unsigned long tIIz = 0;
unsigned long difIIz = 0;
// tiempo intermitente derecho
unsigned long tIDr = 0;
unsigned long difIDr = 0;
// parpadeo luces largas cortas
unsigned long cadenciaPotencia = 2;
```

Las variables temporales controlan los “tiempos” del programa. En detalle:

- ✓ La variable *t* sirve para contar los milisegundos transcurridos desde que se encendió la placa.
- ✓ La variable *retardo* define la duración del parpadeo de los intermitentes.
- ✓ La variable *cadencia* controla la frecuencia de parpadeo de los intermitentes.
- ✓ La variable *cadenciaEmergencia* controla la frecuencia de parpadeo de las luces en estado de Emergencia. Podrá cambiarse posteriormente.
- ✓ La variable *tIIz* indica el tiempo, posterior a *t* en el momento de activación, en el que acabará la secuencia de activación del intermitente izquierdo (*tIDr* análogo en el derecho).
- ✓ La variable *difIIz* controla la diferencia que existe entre *tIIz* y el tiempo transcurrido. Mientras *difIIz* sea mayor que cero, el intermitente izquierdo se activará (*difIDr* análogo en el derecho).
- ✓ La variable *cadenaPotencia* controla el parpadeo rápido de las luces cortas y largas para disminuir el consumo de las mismas.



6.4. Setup

```
void setup () {
// salidas
pinMode(luzFreno,OUTPUT);
pinMode(intermiteIzquierdo,OUTPUT);
pinMode(intermiteDerecho,OUTPUT);
pinMode(Largas,OUTPUT);
pinMode(Cortas,OUTPUT);
// pulsadores
pinMode(pulsadorFreno,INPUT);
pinMode(pulsadorIzquierdo,INPUT);
pinMode(pulsadorDerecho,INPUT);
pinMode(pulsadorLuces,INPUT);
}
```

Simplemente activa los pines de salida y entrada.

6.5. Bucle principal LOOP

6.5.1. Contamos el tiempo

```
// *****
// Conteo tiempo
// *****
t = millis(); // microsegundos despues de encendido
```

6.5.2. Bucle del freno

```
// *****
// Pulso luz de freno
// *****
if (digitalRead(pulsadorFreno) == HIGH) { // 1a condicional
    delay(50);
    if (digitalRead(pulsadorFreno) == HIGH) {
        // 2a condicional
        estadoFreno=HIGH;
        estadoAlarma = LOW;
        // desconecta el estado de alarma
    } // 2a condicional FIN
} else {
    estadoFreno=LOW;
} // 1a condicional FIN

// Enciendo luz freno
digitalWrite(luzFreno,estadoFreno);
```

Al pulsar el pulsador del freno (fin de carrera), hace un doble reconocimiento para evitar activaciones espurias. Desconecta el estado de alarma (sirve para eso si la alarma está encendida) y activa el estado del freno. Si no, el estado del freno lo deja en LOW (desconectado).

Posteriormente en la salida escribe el estado del freno.

6.5.3. Bucle de las luces cortas y largas



```

// *****
// Pulso luz LARGAS / CORTAS
// *****
if (digitalRead(pulsadorLuces) == HIGH && estadoAlarma==LOW) { // 1ª
    condicional
    delay(200);
    if (digitalRead(pulsadorLuces) == HIGH && estadoAlarma==LOW) { // 2ª
        condicional
        estadoLuces = estadoLuces + 1;
        // estadoAlarma = LOW; // desconecta el estado de alarma; mejor se
        // quita el estado de alarma pulsando el freno.
    } // 2ª condicional FIN
} // 1ª condicional FIN

// desbordamiento
if (estadoLuces>=3) { estadoLuces =0; } // condicional de desbordamiento

if (estadoLuces==1 && (t/cadenciaPotencia)%2==HIGH ) {
    digitalWrite(Largas ,LOW);
    digitalWrite(Cortas ,HIGH);
} else if (estadoLuces==2 && (t/cadenciaPotencia)%2==HIGH) {
    digitalWrite(Largas ,HIGH);
    digitalWrite(Cortas ,HIGH);
} else {
    digitalWrite(Largas ,LOW);
    digitalWrite(Cortas ,LOW);
}

// Al encender las luces, pulsos de la luz de freno
// Reconocimiento de pulsos
if (estadoLuces>=1 && t%1500<=200) {
    digitalWrite(luzFreno ,HIGH);
} else {
    digitalWrite(luzFreno ,estadoFreno); // como este el feeno
}

```

Tiene tres partes diferenciadas:

1. Lee el estado del pulsador, evitando activaciones espurias, y, cada vez que detecta una detección, el estado de las luces lo aumenta en uno.
2. Controla el desbordamiento de dicho estado. Si es igual o mayor que tres lo vuelve a poner a cero. Así la variable *estadoLuces* sólo valdrá entre cero y dos. En cero, todas se apagan; en uno se encienden las cortas y en dos, las cortas y largas. La condición $(t/cadenciaPotencia)\%2==HIGH$ permite que sólo en el intervalo temporal definido por *cadenciaPotencia* se activen las luces, reduciendo el consumo.
3. Durante un tiempo definido por la expresión $t\%1500<=200$ se permite el encendido de las luces de freno (parpadeo lento) siempre que las luces cortas y/o largas estén encendidas, lo que permite la señalización de la bicicleta de noche por la parte trasera.

6.5.4. Intermitentes izquierdo y derecho

```
// *****
```



```

// Intermitentes IZQUIERDOS
// ****
if (digitalRead(pulsadorIzquierdo) == HIGH && estadoAlarma==LOW) { // 1ª
    condicional
    delay(100);
    if (digitalRead(pulsadorIzquierdo) == HIGH && estadoAlarma==LOW) { // 2ª
        condicional
        difIIz = retardo * 1000; // diferencia entre los milisegundos
        actuales mas el retardo
        tIIz = t + difIIz; // tiempo en el que acaba la secuencia
        difIDr = 0; // anula el intermitente izquierdo
    } // 2ª condicional FIN
} // 1ª condicional FIN
// } else {
//     estadoIIz = LOW;
// } // 1ª condicional FIN

if (difIIz >0) {
    estadoIIz = (difIIz/cadencia)%2;
    digitalWrite (intermitenteIzquierdo ,estadoIIz);
    difIIz = tIIz - t; // voy calculando la diferencia
} else if (difIIz <=0) {
    difIIz = 0; // fuerza a cero
    digitalWrite (intermitenteIzquierdo ,LOW); // apaga intermitente
}

// ****
// Intermitentes DERECHOS
// ****
if (digitalRead(pulsadorDerecho) == HIGH && estadoAlarma==LOW) { // 1ª
    condicional
    delay(100);
    if (digitalRead(pulsadorDerecho) == HIGH && estadoAlarma==LOW) { // 2ª
        condicional
        difIDr = retardo * 1000; // diferencia entre los milisegundos
        actuales mas el retardo
        tIDr = t + difIDr; // tiempo en el que acaba la secuencia
        difIIz = 0; // anula el intermitente izquierdo
    } // 2ª condicional FIN
} // 1ª condicional FIN
// } else {
//     estadoIIz = LOW;
// } // 1ª condicional FIN

if (difIDr >0) {
    estadoIDr = (difIDr/cadencia)%2;
    digitalWrite (intermitenteDerecho ,estadoIDr);
    difIDr = tIDr - t; // voy calculando la diferencia
} else if (difIDr <=0) {
    difIDr = 0; // fuerza a cero
    digitalWrite (intermitenteDerecho ,LOW); // apaga intermitente
}

```

Paso a explicar sólo la parte de uno de los intermitentes (izquierdo). La otra es simétrica.



1. Reconoce si se ha pulsado el intermitente izquierdo, y si el estado no es de alarma y la detección no es espuria, realiza tres operaciones
 - a) Calcula $difIIz$, según el retardo previamente establecido.
 - b) Calcula $tIIz$, tiempo máximo en el que estará activo.
 - c) Anula la $difIDr$, para desactivar el derecho caso de que estuviese encendido.
2. Si $difIIz > 0$, calcula $estadoIIz$ según la *cadencia* establecida. Escribe el estado en la salida y resta $difIIz$ del tiempo máximo entre el actual. Esta diferencia irá disminuyendo, de forma que cuando se alcance $tIIz$ será cero. Cuando lo sea, apagará el intermitente.

6.5.5. Luces de emergencia

```
// ****
// Luces de Emergencia
// ****
// Se encienden todas las luces (parpadeando) al pulsar los dos
// intermitentes a la vez
if (digitalRead(pulsadorIzquierdo) == HIGH && digitalRead(pulsadorDerecho)
    == HIGH) { // 1ª condicional
  delay(300);
  if (digitalRead(pulsadorIzquierdo) == HIGH && digitalRead(
      pulsadorDerecho) == HIGH) { // 2ª condicional
    estadoAlarma = HIGH;
    difIIz=0; // anula intermitente izquierdo
    difIDr=0; // anula intermitente derecho
    estadoLuces = LOW; // apaga las luces
    estadoFreno = LOW; // anula el freno
  } // Fin del 2º condicional
} // Fin del 1º condicional

if (estadoAlarma) {
  estadoEmergencia = (t/cadenciaEmergencia)%2;
  digitalWrite(Largas, estadoEmergencia^HIGH);
  digitalWrite(Cortas, estadoEmergencia);
  digitalWrite(intermiteDerecho, estadoEmergencia^HIGH);
  digitalWrite(intermiteIzquierdo, estadoEmergencia^HIGH);
  digitalWrite(luzFreno, estadoEmergencia);
} // condiconal de Alarma
```

Si se pulsa durante un tiempo (300 ms) los dos pulsadores de los intermitentes a la vez se entra en estado de alarma y...

1. Se anula el resto de estados. Sólo se activa el estado de alarma *estadoAlarma*.
2. En este estado, se calcula un *estadoEmergencia* con la frecuencia definida por *cadenciaEmergencia*, y en él parpadean todas las luces, algunas sincronizadas y otras desincronizadas (mediante la función booleana XOR: HIGH).

Este estado es útil si de noche se desea indicar la presencia de la bicicleta cuando esté en reposo.



6.5.6. Disminuyo/aumento la cadencia de las luces de emergencia

```

// ****
// En estado de Alarma, pulso Intermitente derecho para aumentar la
// cadencia de emergencia
// ****
if (digitalRead(pulsadorDerecho) == HIGH && estadoAlarma==HIGH) { // 1ª
    condicional
    delay(50);
    if (digitalRead(pulsadorDerecho) == HIGH && estadoAlarma==HIGH) { // 2ª
        condicional
        cadenciaEmergencia = (cadenciaEmergencia+50)*(cadenciaEmergencia
            <=1000)+1000*(cadenciaEmergencia>1000);
    } // 2ª condicional FIN
} // 1ª condicional FIN

// ****
// En estado de Alarma, pulso Intermitente izquierdo para disminuir la
// cadencia de emergencia
// ****
if (digitalRead(pulsadorIzquierdo) == HIGH && estadoAlarma==HIGH) { // 1ª
    condicional
    delay(50);
    if (digitalRead(pulsadorIzquierdo) == HIGH && estadoAlarma==HIGH) {
        // 2ª condicional
        cadenciaEmergencia = (cadenciaEmergencia -50)*(cadenciaEmergencia >50)
            +50*(cadenciaEmergencia <=50); // cadencia minima 50
    } // 2ª condicional FIN
} // 1ª condicional FIN

} // FIN DEL PROGRAMA

```

Con este código se permite, siempre que estemos en estado de alarma, aumentar o disminuir la variable *cadenciaEmergencia* de 50 en 50, entre los límites 50 y 1000, permitiendo controlar así la rapidez del parpadeo.

7. Colaboradores

Programadores

1. Gonzalo Aguilera Salas 2 BAC C
2. Eduardo Barba Aguilar 2 BAC C
3. Alejandro Copero Román 2 BAC C
4. David Heredia Soto 2 BAC C
5. Fernando Letrán García 2 BAC D



6. Borja García Barea 2 BAC E
7. Arturo Paz Payá 2 BAC E

Diseño, construcción y acabados

1. Juan Manuel Fernandez Andrades 2 BAC A
2. Luis Moreno Martin 2 BAC C
3. Christian Bermúdez 2 ESO E

Referencias

- [1] García DG. Apuntes de ARDUINO Nivel Pardillo. Daniel Gallardo García;.
- [2] García DG. Apuntes de ARDUINO Nivel Enteraillo. García, Daniel Gallardo;.
- [3] Herrador RE. Guía de usuario de ARDUINO. Rafael Enríquez Herrador; 2009.
- [4] Banzi M. Getting started with Arduino. O'Rei; 2008.



APÉNDICES

A. Listado completo del programa

```

// declaracion de variables
// Freno
const int luzFreno = 3;
const int pulsadorFreno = 10;
// Intermitentes
const int intermitenteIzquierdo= 5;
const int intermitenteDerecho= 4;
const int pulsadorIzquierdo= 12;
const int pulsadorDerecho= 11;
// Luces cortas y largas
const int Largas = 6;
const int Cortas = 7;
const int pulsadorLuces = 13;

// variables de estado
int estadoFreno = LOW;
int estadoLuces = 0;
int estadoIIz = 0;
int estadoIDr = 0;
int estadoAlarma = 0; // controla si la bici est en posicin o no de emergencia.
int estadoEmergencia = 0; // controla el parpadeo de emergencia

// variables temporales
unsigned long t = 0;
unsigned long retardo = 10; // segundos que dura el parpadeo de los intermitentes
unsigned long cadencia =200; // cadencia o frecuencia de parpadeo
unsigned long cadenciaEmergencia = 500; // frecuencia de parpadeo en el estado de
emergencia
// tiempo intermitente izquierdo
unsigned long tIIz = 0;
unsigned long difIIz = 0;
// tiempo intermitente derecho
unsigned long tIDr = 0;
unsigned long difIDr = 0;
// parpadeo luces largas cortas
unsigned long cadenciaPotencia = 2;

void setup () {
    // salidas
    pinMode(luzFreno ,OUTPUT);
    pinMode(intermítenteIzquierdo ,OUTPUT);
    pinMode(intermítenteDerecho ,OUTPUT);
    pinMode(Largas ,OUTPUT);
    pinMode(Cortas ,OUTPUT);
    // pulsadores
    pinMode(pulsadorFreno ,INPUT);
    pinMode(pulsadorIzquierdo ,INPUT);
    pinMode(pulsadorDerecho ,INPUT);
    pinMode(pulsadorLuces ,INPUT);
}

void loop () {
    // Prueba de encendido
    /* digitalWrite(luzFreno,HIGH);

```



```
digitalWrite(intermitenteIzquierdo,HIGH);
digitalWrite(intermitenteDerecho,HIGH);
digitalWrite(Largas,HIGH);
digitalWrite(Cortas,HIGH); */

// *****
// Conteo tiempo
// *****
t = millis(); // microsegundos despues de encendido
// if (((t/cantidadMicrosegundos) %2)==0) {estadoGeneral=LOW;} else {estadoGeneral=
HIGH;}

// *****
// Pulso luz de freno
// *****
if (digitalRead(pulsadorFreno) == HIGH) { // 1a condicional
delay(50);
    if (digitalRead(pulsadorFreno) == HIGH) { // 2a condicional
        estadoFreno=HIGH;
        estadoAlarma = LOW; // desconecta el estado de alarma
    } // 2a condicional FIN
} else {
    estadoFreno=LOW;
}// 1a condicional FIN

// Enciendo luz freno
digitalWrite(luzFreno ,estadoFreno);

// *****
// Pulso luz LARGAS / CORTAS
// *****
if (digitalRead(pulsadorLuces) == HIGH && estadoAlarma==LOW) { // 1a condicional
delay(200);
    if (digitalRead(pulsadorLuces) == HIGH && estadoAlarma==LOW) { // 2a
        condicional
        estadoLuces = estadoLuces + 1;
        // estadoAlarma = LOW; // desconecta el estado de alarma; mejor se quita el
        // estado de alarma pulsando el freno.
    } // 2a condicional FIN
}// 1a condicional FIN

// desbordamiento
if (estadoLuces>=3) { estadoLuces =0; } // condicional de desbordamiento

if (estadoLuces==1 && (t/cadenciaPotencia)%2==HIGH ) {
    digitalWrite(Largas ,LOW);
    digitalWrite(Cortas ,HIGH);
} else if (estadoLuces==2 && (t/cadenciaPotencia)%2==HIGH) {
    digitalWrite(Largas ,HIGH);
    digitalWrite(Cortas ,HIGH);
} else {
    digitalWrite(Largas ,LOW);
    digitalWrite(Cortas ,LOW);
}

// Al encender las luces, pulsos de la luz de freno
// Reconocimiento de pulsos
if (estadoLuces>=1 && t%1500<=200) {
    digitalWrite(luzFreno ,HIGH);
} else {
    digitalWrite(luzFreno ,estadoFreno); // como este el freno
```



```
}

// *****
// Intermitentes IZQUIERDOS
// *****
if (digitalRead(pulsadorIzquierdo) == HIGH && estadoAlarma==LOW) { // 1ª
    condicional
    delay(100);
    if (digitalRead(pulsadorIzquierdo) == HIGH && estadoAlarma==LOW) { // 2ª
        condicional
        difIIz = retardo * 1000; // diferencia entre los milisegundos actuales mas el
        retardo
        tIIz = t + difIIz; // tiempo en el que acaba la secuencia
        difIDr = 0; // anula el intermitente izquierdo
    } // 2ª condicional FIN
} // 1ª condicional FIN
// } else {
//     estadoIIz = LOW;
// } // 1ª condicional FIN

if (difIIz >0) {
    estadoIIz = (difIIz/cadencia)%2;
    digitalWrite (intermitenteIzquierdo ,estadoIIz);
    difIIz = tIIz - t; // voy calculando la diferencia
} else if (difIIz <=0) {
    difIIz = 0; // fuerza a cero
    digitalWrite (intermitenteIzquierdo ,LOW); // apaga intermitente
}

// *****
// Intermitentes DERECHOS
// *****
if (digitalRead(pulsadorDerecho) == HIGH && estadoAlarma==LOW) { // 1ª condicional
    delay(100);
    if (digitalRead(pulsadorDerecho) == HIGH && estadoAlarma==LOW) { // 2ª
        condicional
        difIDr = retardo * 1000; // diferencia entre los milisegundos actuales mas el
        retardo
        tIDr = t + difIDr; // tiempo en el que acaba la secuencia
        difIIz = 0; // anula el intermitente izquierdo
    } // 2ª condicional FIN
} // 1ª condicional FIN
// } else {
//     estadoIIz = LOW;
// } // 1ª condicional FIN

if (difIDr >0) {
    estadoIDr = (difIDr/cadencia)%2;
    digitalWrite (intermitenteDerecho ,estadoIDr);
    difIDr = tIDr - t; // voy calculando la diferencia
} else if (difIDr <=0) {
    difIDr = 0; // fuerza a cero
    digitalWrite (intermitenteDerecho ,LOW); // apaga intermitente
}

// ***** FIN PROGRAMA PRINCIPAL
*****



// *****
// Luces de Emergencia
// *****
```



```
// Se encienden todas las luces (parpadeando) al pulsar los dos intermitentes a la vez
if (digitalRead(pulsadorIzquierdo) == HIGH && digitalRead(pulsadorDerecho) == HIGH)
    { // 1a condicional
    delay(300);
        if (digitalRead(pulsadorIzquierdo) == HIGH && digitalRead(pulsadorDerecho) == HIGH) { // 2a condicional
            estadoAlarma = HIGH;
            difIIz=0; // anula intermitente izquierdo
            difIDr=0; // anula intermitente derecho
            estadoLuces = LOW; // apaga las luces
            estadoFreno = LOW; // anula el freno
        } // Fin del 2o condicional
    } // Fin del 1o condicional

if (estadoAlarma) {
    estadoEmergencia = (t/cadenciaEmergencia)%2;
    digitalWrite(Largas ,estadoEmergencia^HIGH);
    digitalWrite(Cortas ,estadoEmergencia);
    digitalWrite(intermiteDerecho ,estadoEmergencia^HIGH);
    digitalWrite(intermiteIzquierdo ,estadoEmergencia^HIGH);
    digitalWrite(luzFreno ,estadoEmergencia);
} // condicional de Alarma

// ****
// En estado de Alarma, pulso Intermitente derecho para aumentar la cadencia de emergencia
// ****

if (digitalRead(pulsadorDerecho) == HIGH && estadoAlarma==HIGH) { // 1a condicional
delay(50);
    if (digitalRead(pulsadorDerecho) == HIGH && estadoAlarma==HIGH) { // 2a condicional
        cadenciaEmergencia = (cadenciaEmergencia+50)*(cadenciaEmergencia <=1000)
            +1000*(cadenciaEmergencia>1000);
    } // 2a condicional FIN
} // 1a condicional FIN

// ****
// En estado de Alarma, pulso Intermitente izquierdo para disminuir la cadencia de emergencia
// ****

if (digitalRead(pulsadorIzquierdo) == HIGH && estadoAlarma==HIGH) { // 1a condicional
delay(50);
    if (digitalRead(pulsadorIzquierdo) == HIGH && estadoAlarma==HIGH) { // 2a condicional
        cadenciaEmergencia = (cadenciaEmergencia -50)*(cadenciaEmergencia >50)+50*(
            cadenciaEmergencia <=50); // cadencia minima 50
    } // 2a condicional FIN
} // 1a condicional FIN

} // FIN DEL PROGRAMA
```