E.T.S. de Ingeniería Industrial, Informática
y de Telecomunicación

Visor de infrarrojos de 360° para robot basado en Arduino

Grado en Ingeniería Eléctrica y Electrónica

Trabajo Fin de Grado

Autor: Javier Gago Balda

Tutor: Vicente Senosiain

Fecha: Pamplona, 25 de junio de 2014

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

RESUMEN

El Trabajo Fin de Grado trata sobre una plataforma robótica ya construida extraída de un coche teledirigido de juguete a la que se adapta un sensor infrarrojo con un mecanismo que permite un ángulo de visión de 360°. Esto se consigue mediante un pequeño motor que hace girar una plataforma constantemente, y sobre esta plataforma se coloca un espejo con el ángulo necesario para que la luz infrarroja rebote en él y sea detectada por el sensor.

Además, se construye una fuente de radiación infrarroja independiente. Esta fuente debe emitir la luz con una determinada frecuencia y ciclo de trabajo para evitar que el sensor reciba emisiones infrarrojas producidas por otros dispositivos. Mediante la programación de un microcontrolador (Arduino) se controla la tracción y la dirección del robot con el fin de conseguir que el robot se dirija hacia la fuente de radiación infrarroja, siendo esta fuente móvil.

LISTA DE PALABRAS CLAVE

Robot

Microcontrolador

Arduino

Driver

Sensor

Infrarrojo

Tracción

Interruptor

Motor

Dirección

Encoder

Emisor

Receptor

Índice

MEMORIA

1. INTRODUCCIÓN

El Trabajo Fin de Grado consiste en la construcción de un robot que sea capaz de dirigirse hacia la luz infrarroja de un mando. Este mando emitirá una luz de una frecuencia en concreto para evitar la detección de luces infrarrojas emitidas por otros sistemas que pueden provocar errores en la detección del receptor. Su alimentación se realizará mediante una pila de 9V, que estará conectada a la placa protoboard mediante un interruptor para facilitar su conexión y desconexión.

Para la construcción del robot se ha adquirido un coche teledirigido, del que se ha aprovechado su carrocería básica, así como los motores y el interruptor. Para implementar la lógica se ha utilizado el Arduino UNO por razones que se explicarán más adelante. También es necesario el uso de un driver de potencia debido al consumo de corriente de los motores, el cual el Arduino por sí sólo no puede proporcionar. Se trata de la placa Arduino Motor Shield, y se conecta encima de la placa Arduino.

Sobre la base del coche teledirigido se va a montar, a parte del Arduino y de la placa Arduino Motor Shield, una estructura giratoria controlada por un interruptor óptico ranurado en cuya base se colocará un espejo, el circuito receptor de la luz infrarroja y las pilas que se van a utilizar para la alimentación tanto del Arduino como del resto de circuitos ya mencionados. Así mismo, el montaje resultante es de la siguiente manera:

.

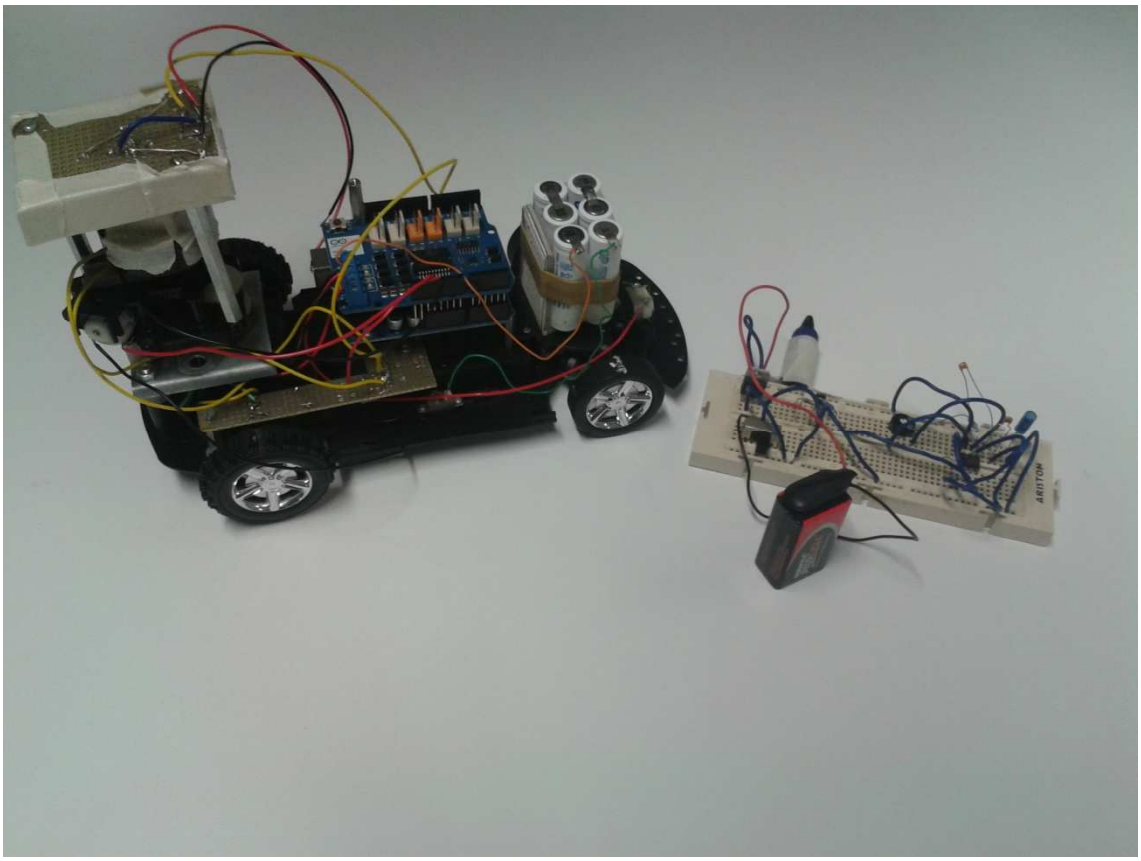


Figura 1: robot completo y mando infrarrojo (elaboración propia)

2. DESARROLLO DEL TRABAJO

El trabajo se va a desarrollar paso a paso, empezando por la construcción del mando emisor de la luz infrarroja y siguiendo por la construcción de los circuitos necesarios para el correcto funcionamiento del robot. A continuación se procederá a la lectura de dichos circuitos y al control del robot con la utilización del Arduino. Una vez comprobado el correcto funcionamiento de las diferentes partes del proyecto, se montará todo sobre la plataforma robótica, se procederá a realizar el programa para que el robot se dirija hacia la luz infrarroja y se comprobará su correcto funcionamiento.

2.1 Estructura del robot

El robot consta de un coche teledirigido de juguete que está compuesto por dos motores. En primer lugar está el motor delantero, el cual se encarga de la dirección. Tiene tres posiciones: recto, hacia la derecha o hacia la izquierda. Tiene dos cables, uno rojo y otro negro. Tomando el cable rojo como positivo y el negro como negativo, su funcionamiento es el siguiente:

- Si recibe 5V positivos gira hacia la izquierda
- Si ve 5V inversamente polarizados gira hacia la derecha
- Si no recibe ninguna tensión el coche se moverá recto

En segundo lugar está el motor trasero, el cual se encarga de la tracción. Éste también tiene tres posiciones: hacia delante, hacia atrás o parado. Consta de dos cables (rojo y negro), y su funcionamiento es parecido al de la dirección:

- Con 5V positivos se moverá hacia delante
- Si ve 5V negativos se moverá hacia atrás
- Si no ve ninguna tensión se parará

Además de los motores, el robot también tiene un interruptor ON/OFF, el cual se va a utilizar para una mayor comodidad en la conexión y desconexión de la alimentación tanto del Arduino como de los motores y circuitos de los sensores.

Las dimensiones del robot son de 12cm de ancho y 23,5cm de largo, y tiene una altura máxima de 5cm, que es el diámetro de las ruedas traseras. Se ha de tener en cuenta que esta altura va a aumentar considerablemente, ya que tanto el Arduino, como las pilas y los circuitos de los sensores van a ir montados encima de ello.

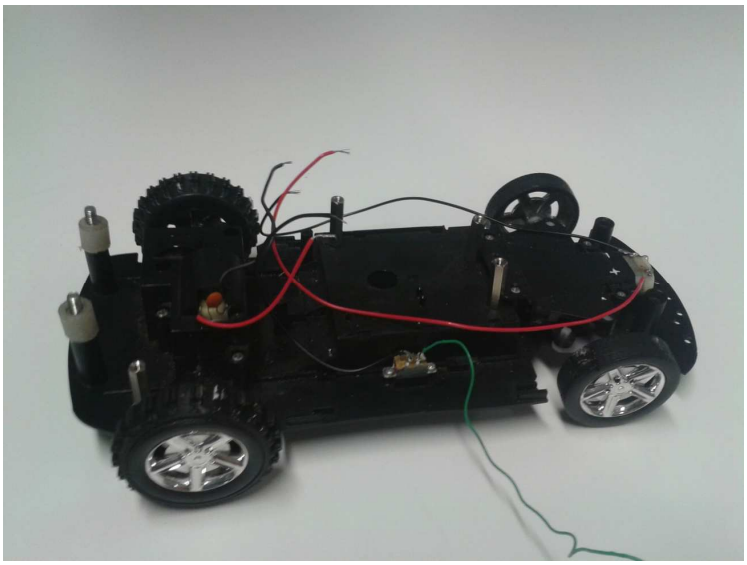


Figura 2: plataforma robótica (elaboración propia)

2.2 Fuente de radiación infrarroja

La fuente infrarroja es el dispositivo con el que se indica al robot a dónde debe moverse. Esta fuente va a emitir una señal escalonada de 38 kHz (tiene un margen desde 36 a 40 kHz, teniendo la mayor sensibilidad en los 38kHz), con un ciclo de trabajo del 50% mediante un LED infrarrojo OP165A.

Figura 3: LED infrarrojo OP165A (tienda opiron)

El motivo del uso de la señal a una determinada frecuencia se debe a la necesidad de evitar la detección de señales infrarrojas que pueden ser emitidas por otros dispositivos y pueden ser recibidas por el sensor, lo cual provocaría errores en su funcionamiento. Estas señales escalonadas a una determinada frecuencia son muy utilizadas, por ejemplo, en mandos de la televisión.

La luz infrarroja de 38kHz se consigue utilizando un temporizador 555, un transistor y los correspondientes condensadores y resistencias calculadas para conseguir la señal deseada.

Figura 4: temporizado NE555N (oddwires)

No se debe olvidar la utilización de resistencias en serie con el LED y en la base del transistor para limitar la corriente que circula por ellos y evitar su destrucción. El transistor utilizado es el BC548. En primer lugar, se montó el siguiente circuito para conseguir la señal deseada (cambiando el transistor 2N2222 NPN por el transistor BC548 disponible en el laboratorio):

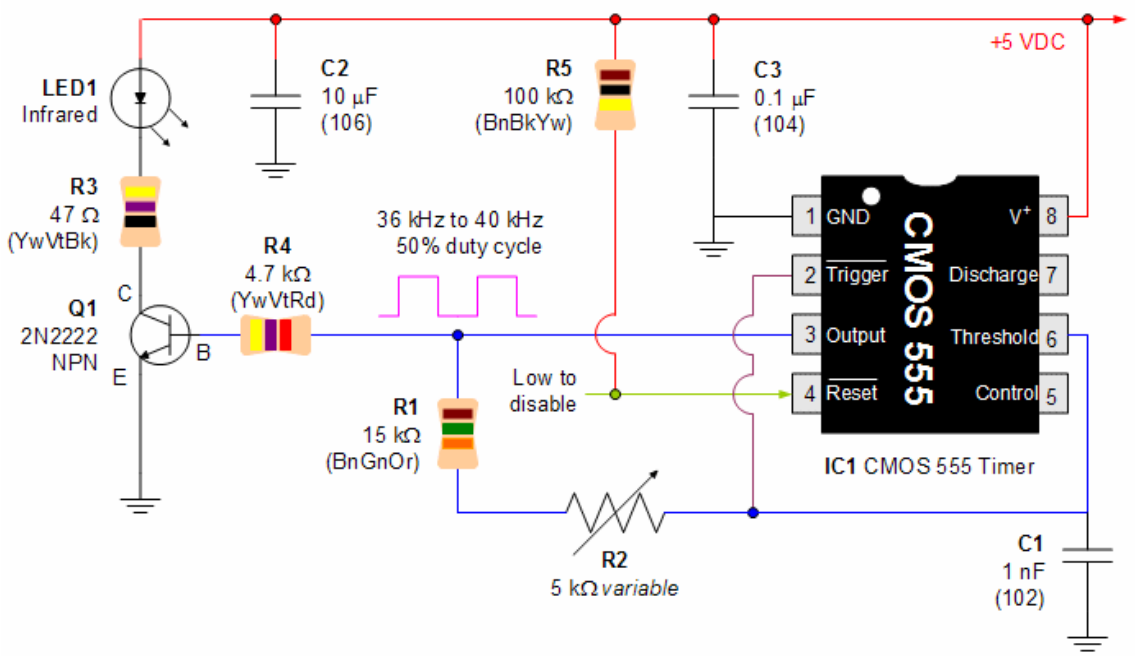


Figura 5. Circuito fuente de radiación infrarroja (ucontrol)

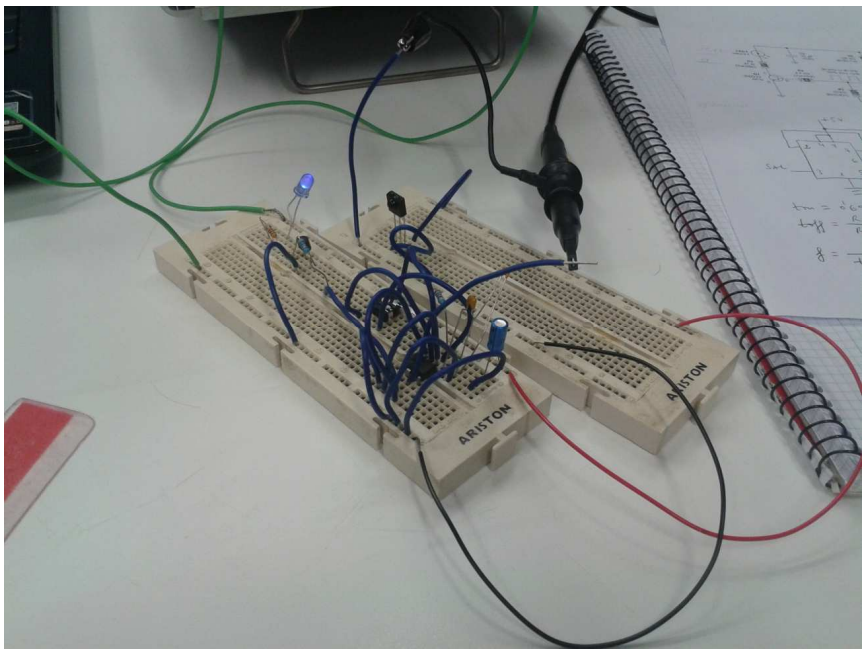


Figura 6: montaje del circuito para obtener la señal de 38kHz y ciclo de trabajo del 50% (elaboración propia)

En la práctica, con este circuito no se conseguía una señal cuadrada de 38kHz, por lo que se modifico el montaje del temporizador. El circuito resultante es el siguiente:

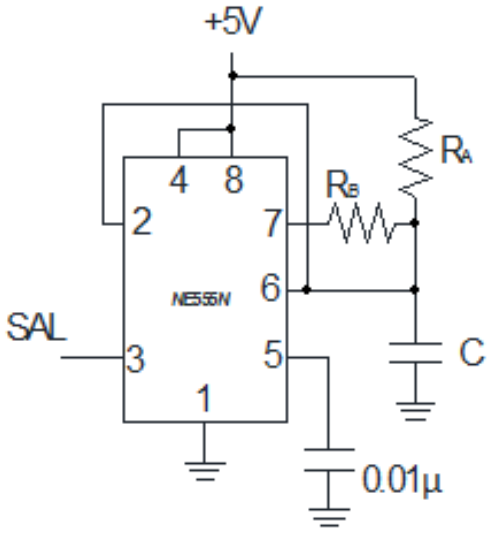


Figura 7: montaje del temporizador para conseguir la señal deseada (elaboración propia)

Se deben calcular los valores de las resistencias y el condensador para obtener la señal que estamos buscando (38 kHz y un ciclo de trabajo del 50%). Para ello se hace uso de las siguientes fórmulas:

$$t_{on} = 0{,}693 \cdot R_A \cdot C$$

$$t_{off} = \frac{R_A \cdot R_B}{R_A + R_B} \cdot C \cdot \ln \frac{R_B - 2R_A}{2R_B - R_A}$$

$$f = \frac{1}{t_{on} + t_{off}}$$

Como el ciclo de trabajo es del 50%, el valor de t_{on} y t_{off} tiene que ser el mismo ($t_{on}=t_{off}$). Sabiendo esto y el valor de la frecuencia, se obtienen los valores de t_{on} y t_{off}.

$$f = \frac{1}{t_{on} + t_{off}} = \frac{1}{2t_{on}} = 38000 \rightarrow t_{on} = \frac{1}{2 \cdot 38000} = 13{,}1579\mu s = t_{off}$$

Una vez obtenidos estos valores, se deben sacar los valores de las resistencias y el condensador.

$$t_{on} = 0{,}693 \cdot R_A \cdot C = 13{,}1579\mu s$$

$$R_A \cdot C = \frac{13{,}1579\mu s}{0{,}693} = 18{,}9869\mu s$$

Se toman los siguientes valores normalizados y aproximados:

- $R_B = 2k\Omega$
- $C = 10nF$

Una vez decididos estos valores, se debe obtener el valor de R_B. Para ello se hace uso de la siguiente ecuación:

$$t_{off} = \frac{R_A \cdot R_B}{R_A + R_B} \cdot C \cdot \ln \frac{R_B - 2R_A}{2R_B - R_A} = 13{,}1579\mu s$$

El valor normalizado que más se aproxima al cumplimiento de esta ecuación es el de la resistencia de 820 Ω, pero realizando el montaje se observa que se consigue una señal de 34.45 kHz. Por lo tanto, en R_B se utiliza un potenciómetro de 1 kΩ y se ajusta manualmente hasta conseguir la frecuencia deseada de 38 kHz. Esta frecuencia se obtiene con una resistencia de 735 Ω.

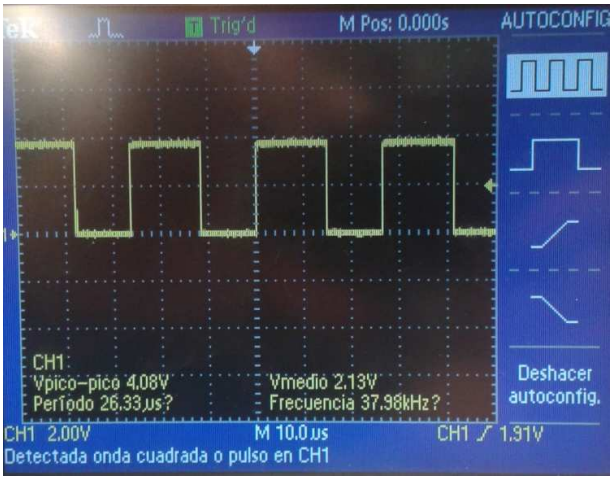


Figura 8: Señal obtenida con los valores de resistencia y capacidad calculados (elaboración propia)

Una vez realizado correctamente el montaje del temporizador, se añade el LED infrarrojo un transistor BC548 de manera que se consiga emitir una luz con la misma frecuencia y el mismo ciclo de trabajo de la señal obtenida en la salida de temporizador.

Figura 9: transistor BC548 (futurlec)

No se deben olvidar la colocación de las resistencias en serie tanto con el LED como en la base del transistor para limitar la corriente por debajo de los límites máximos que vienen recogidos en las hojas de características de dichos elementos.

Por lo que respecta a la alimentación de la fuente, se va a utilizar una pila de 9V, y un regulador de tensión de 5V, el L7805CV. Además, se va a utilizar un interruptor para la una conexión y desconexión más práctica de la alimentación.

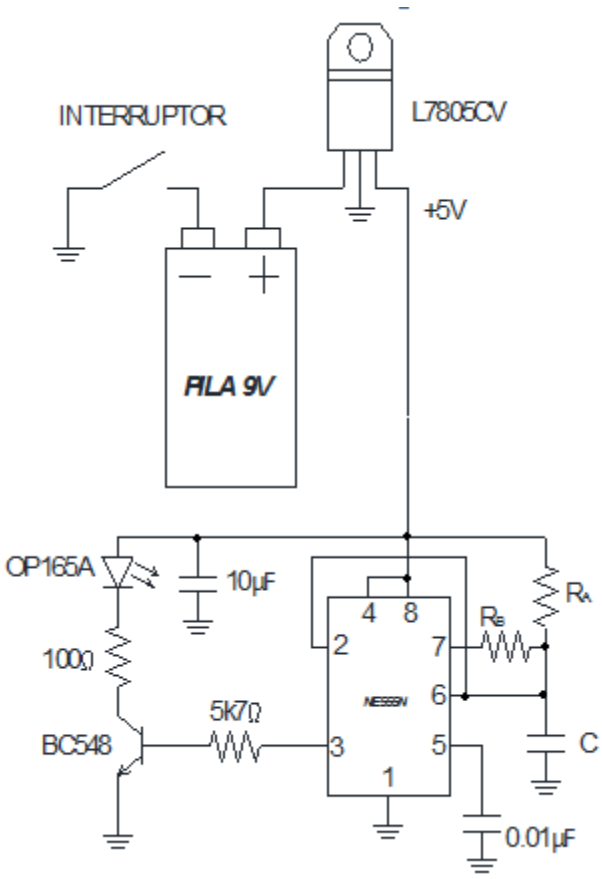


Figura 10: circuito emisor de radiación infrarroja (elaboración propia)

El LED infrarrojo emite la luz de manera muy esparcida. Sin embargo, en este caso se requiere que la luz vaya lo mas recta posible, es decir, como si se tratara de un hierro rígido, para que la detección de la posición de la fuente de radiación respecto del robot sea lo más precisa posible. Para ello se realiza un apantallamiento al LED utilizando un cilindro de goma y una punta de un bolígrafo, cuyo agujero es más pequeño que la del cilindro.

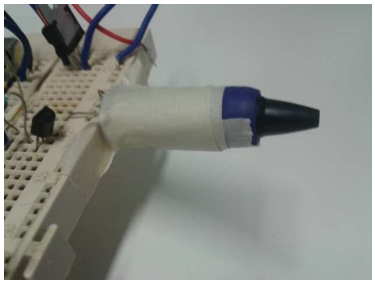


Figura 11: apantallamiento del LED infrarrojo (elaboración propia)

Una vez realizado todo esto, ya se ha finalizado con la construcción de la fuente de radiación infrarroja. El montaje de ésta se realiza en una protoboard, como se puede observar en la siguiente figura:

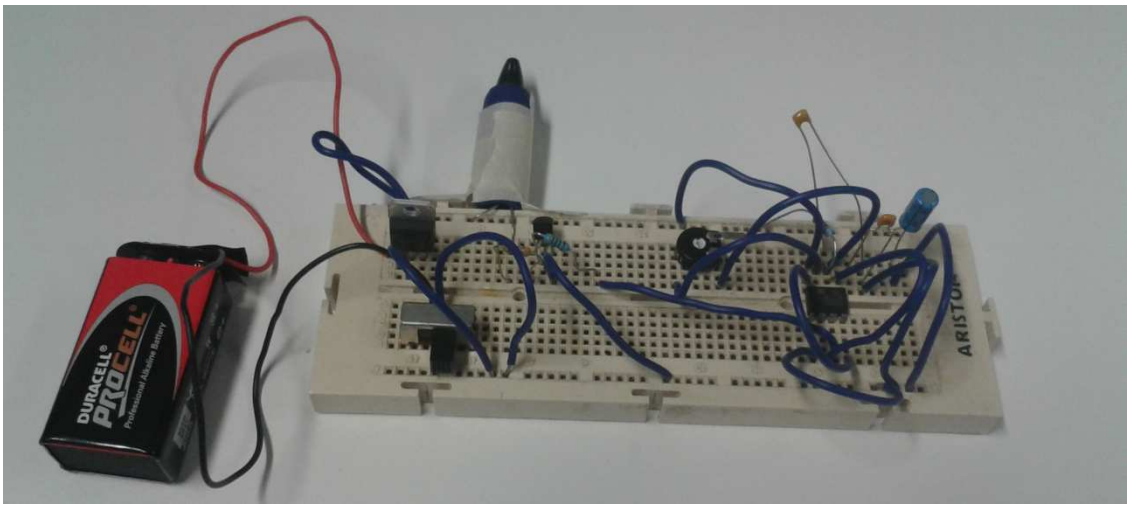


Figura 12: montaje fuente de radiación infrarroja (elaboración propia)

2.3 Sensor infrarrojo

Una vez conseguida la señal de luz infrarroja cuadrada a 38kHz en el emisor infrarrojo, se procede a montar el circuito receptor de dicha luz. Para ello, lo primero que se debe hacer es decidir qué sensor se va a utilizar. En verdad, esta decisión se debe tomar antes de realizar el mando infrarrojo, ya que se necesita saber la frecuencia de la señal escalonada y su ciclo de trabajo. Como en este caso es de 38kHz y el ciclo de trabajo del 50%, el sensor utilizado es el PNA4602M, ya que cumple las características necesarias para la correcta captación de la luz emitida.

Figura 13: sensor infrarrojo PNA4602M (vitronia)

En primer lugar, se debe comprobar que el sensor infrarrojo funciona correctamente. Para ello se realiza un test. Como tiene un semiconductor/chip en su interior, debe estar alimentado a 5V para su funcionamiento. Su conexión es de la siguiente manera:

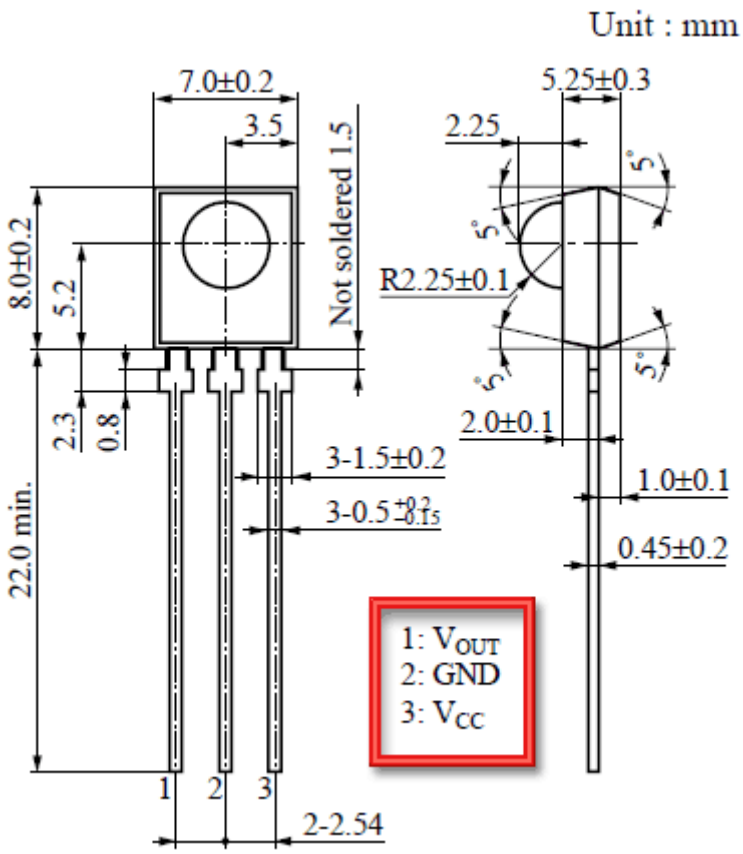


Figura 14: dimensiones y conexión del sensor infrarrojo PNA4602M (vitronia)

- El pin 1 es la salida, de manera que conectamos esta patilla a un LED visible y a una resistencia
- El pin 2 es tierra
- El pin 3 es Vcc, que lo conectamos a 5V

Para realizar el test y comprobar el correcto funcionamiento del sensor se realiza el montaje que se observa a continuación, con la diferencia que se utiliza la fuente de alimentación disponible en el laboratorio en lugar de las pilas:

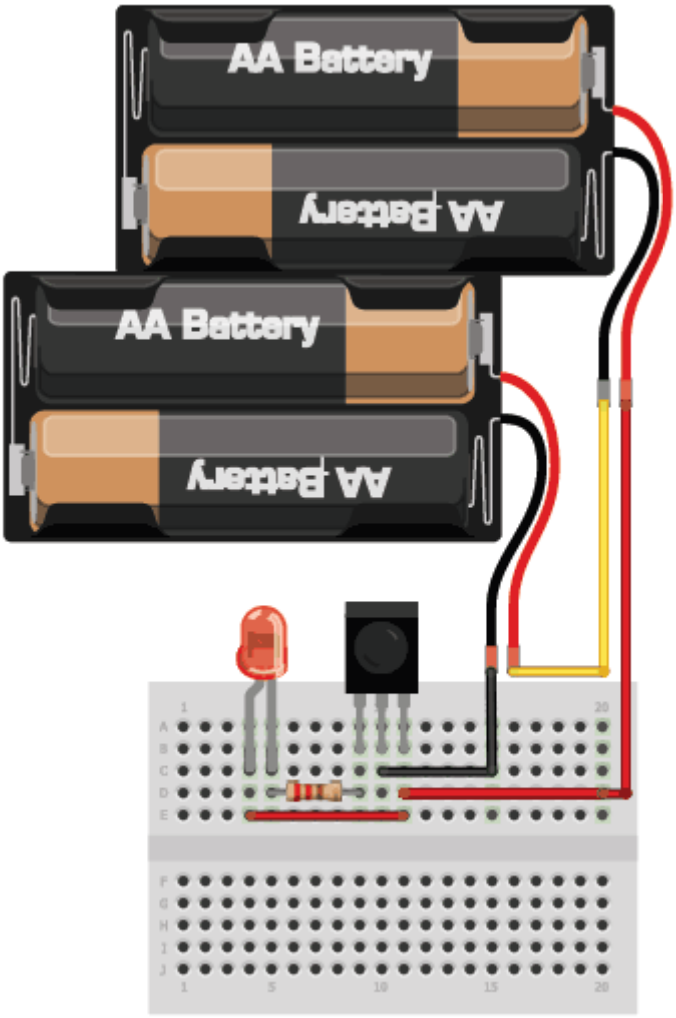


Figura 15: circuito para comprobar el correcto funcionamiento del sensor (adrafuit)

Cuando el detector ve una señal infrarroja de 38 kHz, su salida es baja y asi se enciende el LED. Al contrario, cuando no recibe dicha señal su salida es alta, por lo que el LED ve la misma tensión en las dos patillas y no emite luz. De esta manera se comprueba que el detector no tiene ningún defecto y que su funcionamiento es el correcto.

Una vez comprobado esto, se procede a observar la señal de salida del sensor utilizando un osciloscopio.

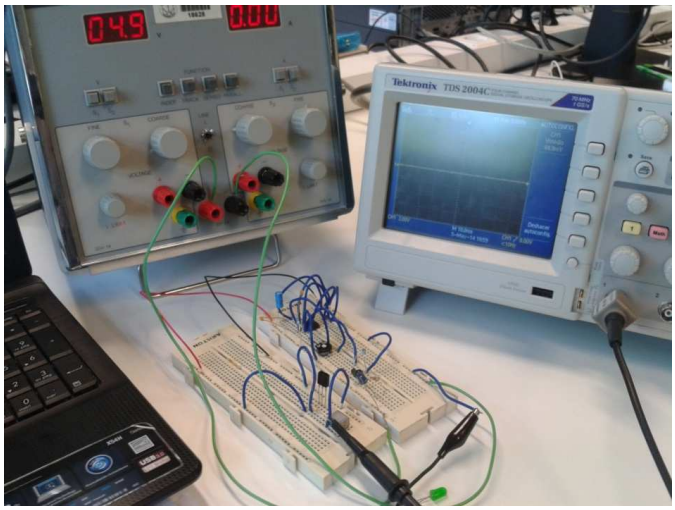


Figura 16: montaje del circuito para estudiar la salida del sensor (elaboración propia)

Se puede observar que cuando el sensor recibe la señal infrarroja su salida es 0, mientras que cuando existe un obstáculo que interfiere esta señal y no llega al sensor la salida es de 5V. Sin embargo, esta salida de 5V es "inestable", es decir, no se mantiene en los 5V sino que varía un poco debido a la contaminación infrarroja que existe en el ambiente.

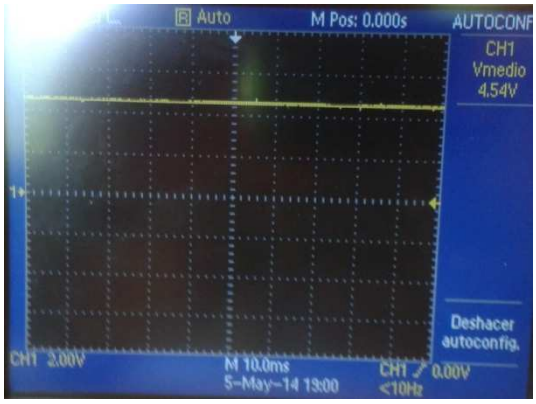


Figura 17: salida sensor con obstáculo

(elaboración propia)

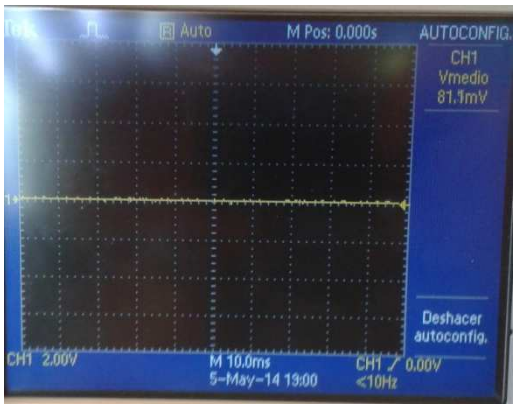


Figura 18: salida sensor sin obstáculo

(elaboración propia)

Para arreglar este problema se utiliza un disparador, concretamente el HEF40106. Cabe decir que dicho disparador invierte su salida, es decir, que cuando le llega una tensión cercana a 5V, su salida es de 0V; al contrario, si le llega una tensión de 0V, su salida será de 5V.

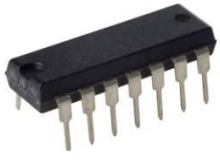


Figura 19: disparador HEF40106 (retroamplis)

Con esto se consigue que, si la variación de la tensión es muy grande, es decir, baje mucho, el Arduino no lea un 0. De esta manera evitamos errores en la lectura de la señal.

Además del disparador, también se ha colocado un filtro pasabajo en la salida del sensor con el fin de eliminar las señales de alta frecuencia. Dicho filtro pasabajo está compuesto por una resistencia de 100kΩ y un condensador de 0,1µF.

También se han puesto condensadores tanto en la alimentación del sensor como en la del disparador, con el fin de estabilizar la tensión. Finalmente, el circuito resultante para la lectura del sensor es el siguiente:

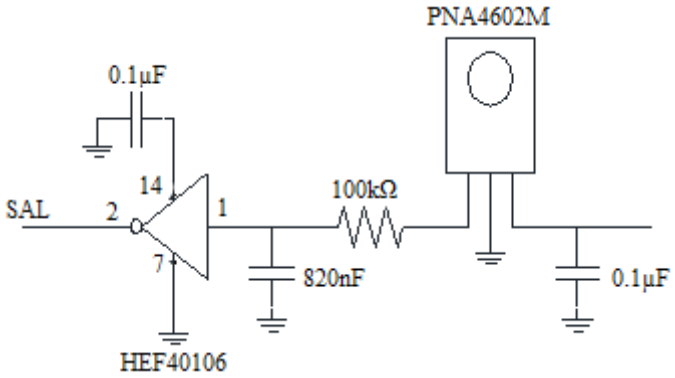


Figura 20: circuito receptor de luz infrarroja (elaboración propia)

Este circuito, una vez que comprobado su correcto funcionamiento, se monta en una placa, e irá montada en la parte trasera de la plataforma robótica, justo encima de la estructura giratoria, como ya se comentará más adelante.

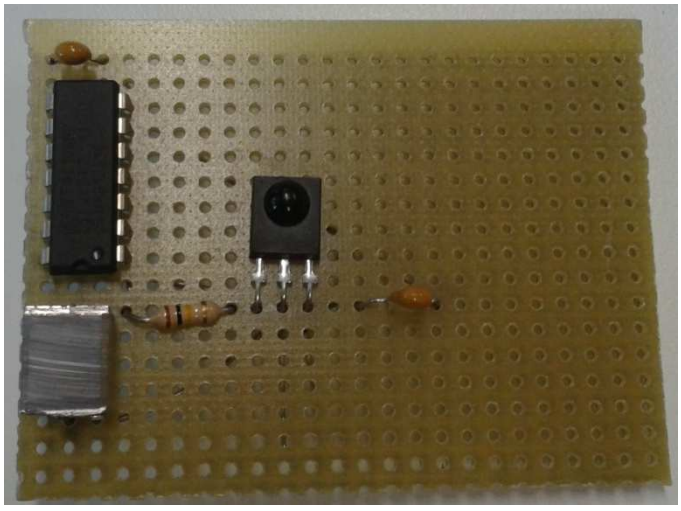


Figura 21: montaje circuito receptor infrarrojo (elaboración propia)

2.4 Montaje estructura giratoria

Como el robot tiene que tener una visión de 360° y el sensor infrarrojo no puede girar debido a que los cables se enredarían y se acabarían rompiendo, se ha optado por construir una base giratoria con un espejo encima. Este giro se realizará mediante un motor que será alimentado a 3,3V con el Arduino, como ya se verá más adelante. El motivo de alimentarlo con esta tensión es que el Arduino dispone de un Pin con dicho valor de tensión y, además, si alimentamos el motor a esta tensión gira a una velocidad angular adecuada para este proyecto. En la base giratoria irá un espejo, el cual tendrá el ángulo idóneo para que la luz infrarroja choque en él y sea dirigida hacia el sensor. El sensor deberá ir colocado justo encima del espejo, en dirección perpendicular a la base donde éste está colocado.

Si se realiza este montaje, surge un problema: la luz emitida por la fuente infrarroja no sólo llega al sensor rebotando en el espejo, sino que puede llegar de forma directa desde cualquier dirección sin pasar por él. Esto conlleva que el receptor infrarrojo puede estar detectando la luz emitida aunque ésta no se encuentre en frente del espejo. De esta manera, resulta imposible realizar el control de la posición del mando infrarrojo, ya que aunque el sensor detecte la luz, no se sabe si esta luz ha sido emitida desde la dirección a la que está dirigido el espejo o proviene de otra cualquier dirección.

Para solucionar este problema se debe apantallar el espejo para asegurarnos que el único camino posible que puede realizar la luz para llegar al receptor sea chocando en él. Se forran todos los lados del espejo, excluyendo el lado por el que debe pasar la luz, y se forra también por los 4 lados a la altura a la que está el sensor. Así se evitan falsas detecciones de la ubicación del mando. El montaje resultante se puede observar a continuación:

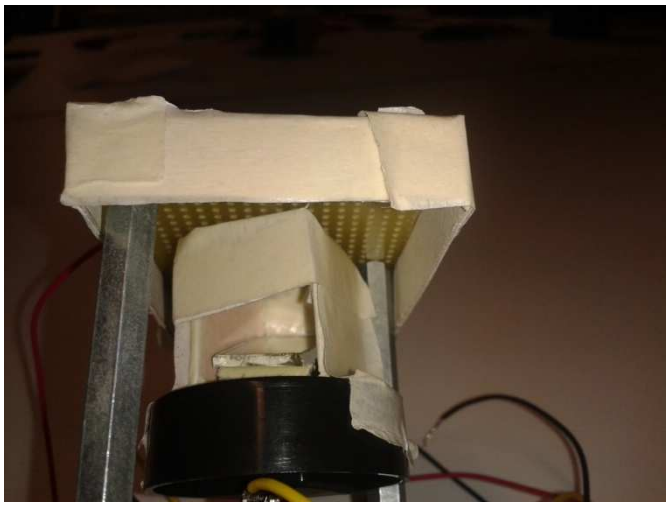


Figura 22: montaje estructura giratoria (elaboración propia)

2.4.1 Interruptor óptico ranurado

Para el futuro control del robot es necesario saber la posición a la que está situado el espejo, es decir, hacia dónde está mirando, para saber de dónde viene la luz y a donde tiene que dirigirse el robot. Para ello, se utiliza un disco ranurado y un interruptor óptico ranurado, exactamente, el OPB620.

Figura 23: interruptor óptico ranurado OPB620 (ptrobotics)

Así se logra transformar el movimiento angular en una serie de impulsos digitales. Estos impulsos generados serán utilizados para controlar el desplazamiento angular de la base giratoria, y en consecuencia, la posición del espejo. El disco está dividido en zonas opacas y transparentes. Estas zonas actúan como un obturador de cámara, las cuales harán la función de permitir o denegar el paso de la luz en forma alternada. Estos cambios serán detectados por el interruptor óptico ranurado, y podrán ser leídos por el Arduino.

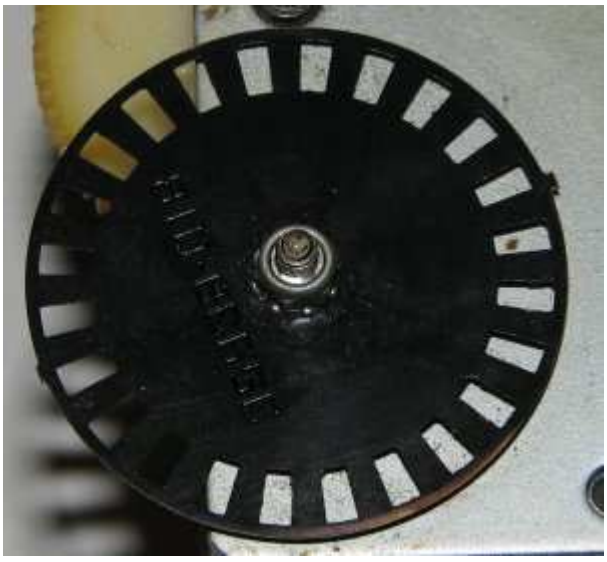


Figura 24: disco ranurado (zumbrovalley)

El OPB620 está compuesto por un diodo emisor de infrarrojo y un fototransistor de silicio NPN, colocado cada uno en cada extremo del elemento con sus respectivos pares de pines.

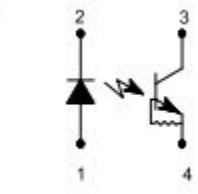


Figura 25: circuito interno del OPB620 (Optekinc)

Este sensor se utiliza para saber en qué posición se encuentra mirando el espejo en cada momento. Lo primero que se va a hacer es comprobar su funcionamiento. Para ello se realiza el siguiente circuito:

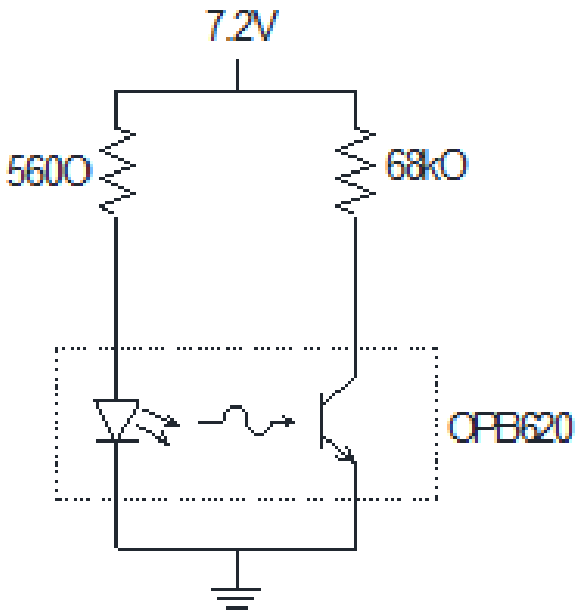


Figura 26: circuito para el control del giro de la estructura giratoria (elaboración propia)

Lo primero que se debe hacer es calcular las resistencias que se han de poner para no cargarse el interruptor óptico. En primer lugar, se calcula la resistencia que se ha de poner en serie con el diodo emisor de infrarrojos. Si se mira la hoja de características, se observa que la corriente máxima que debe circular por él es de 10mA; por lo tanto, teniendo en cuenta que está alimentado con las pilas (6 pilas x 1,2V/pila = 7,2V) y la tensión del diodo, la resistencia que se debe poner en serie es de:

$$R_{diodo} = \frac{V}{I} = \frac{7{,}2 - 1{,}6}{0{,}01} = 560\Omega$$

La resistencia mínima que se debe utilizar es la de 560Ω. Lo primero que hay que hacer es ver si ese valor está normalizado, es decir, si existen resistencias con ese valor. En caso de que no sea así, se utiliza la resistencia normalizada superior más cercana. En este caso el valor de la resistencia sí que está normalizado, por lo que utilizaremos una de 560Ω.

A continuación se calculará la resistencia que ha de ponerse en serie con el fototransistor. Volviendo a la hoja de características, observamos que la corriente máxima que debe circular por el colector es de 100µA, por lo que la resistencia que se debe emplear es:

$$R_{fototransistor} = \frac{V}{I} = \frac{7{,}2 - 0{,}4}{100\mu} = 68k\Omega$$

En este caso, al igual que en el anterior, el valor de la resistencia obtenido también está normalizado, por lo que utilizaremos una resistencia de 68kΩ.

Una vez realizado esto, se monta el circuito. Para ello, en primer lugar se sueldan las resistencias calculadas en un trozo de placa, como se puede observar en la figura de abajo:

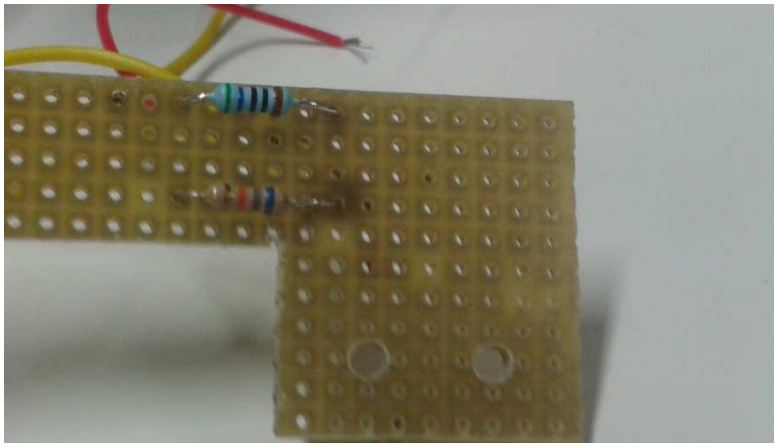


Figura 27: montaje de las resistencias del circuito del OPB620 (elaboración propia)

En segundo lugar se debe colocar el OPB620. Se utiliza un "carril" donde irá colocado el interruptor óptico ranurado, para que éste no se quede fijo y pueda ser ajustado de manera que detecte todos los agujeros del disco ranurado. Este disco debe estar colocado centrado en el eje de la estructura giratoria.

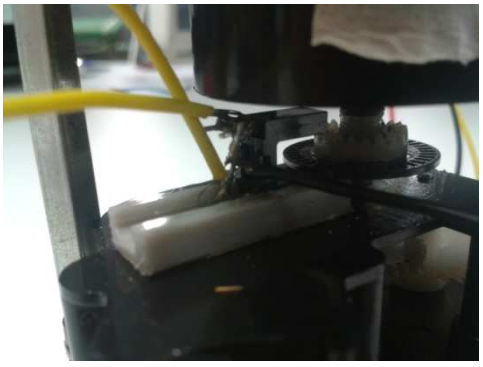


Figura 28: montaje interruptor óptico ranurado OPB620 (elaboración propia)

Si se estudia la tensión en la salida 3, que es la que va a ser leída por el Arduino, se observan dos casos diferentes:

- Cuando no existe ningún obstáculo, es decir, cuando al fototransistor llega la luz del diodo, hay 0,08V.
- Cuando existe algún obstáculo, es decir, cuando al fototransistor no le llega la luz del diodo, hay 5,4V.

Estos valores son los que van a ser leídos por el Arduino y se van a utilizar para saber en todo momento la orientación del espejo giratorio.

El resultado del montaje completo de la estructura giratoria y su control mediante el OPB620 es el siguiente:

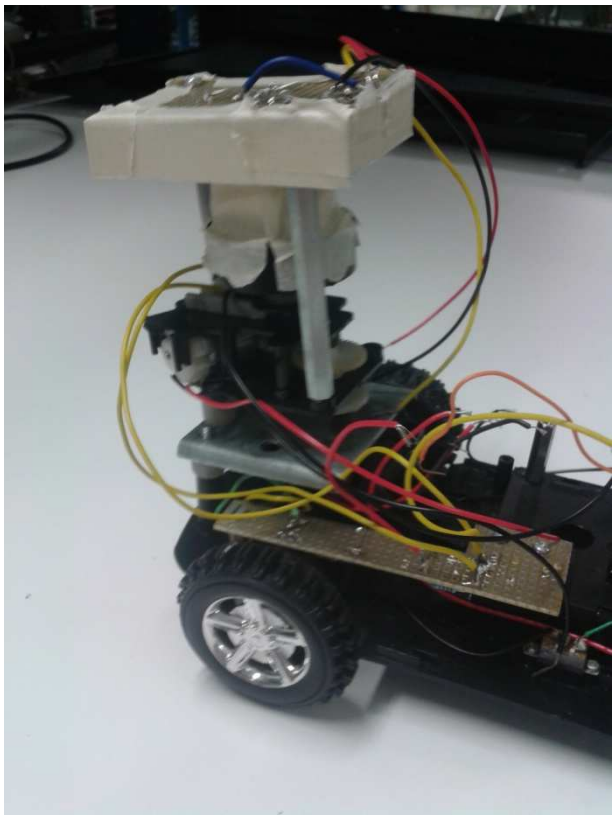


Figura 29: estructura giratoria completa (elaboración propia)

2.5 Microcontrolador

Un microcontrolador es un circuito integrado que está formado por las tres unidades funcionales de un ordenador: microprocesador, memoria y periféricos de entrada y salida.

La forma en la que funciona un microcontrolador se determina por el programa almacenado en su memoria. Este programa se puede diseñar y escribir en diferentes lenguajes de programación y tras una compilación, se descarga en la memoria interna del microcontrolador en lenguaje ejecutable. Esto, unido a su alta flexibilidad, hacen que los microcontroladores se empleen en multitud de aplicaciones: automatización, robótica, domótica, medicina, aeronáutica, automoción, telecomunicaciones, etc.

Las principales características de los microcontroladores son:

- *Microprocesador*: típicamente de 8 bits, pero existen versiones de 4, 32 y hasta 64 bits con arquitectura Harvard, con memoria/bus de datos separada de la memoria/bus de instrucciones de programa, o arquitectura de von Neumann con memoria/bus de datos y memoria/bus de programa compartida.

- *Memoria de Programa*: puede ser una memoria ROM (Ready Only Memory), EPROM (Electrically Programmable ROM), EEPROM (Electrically Erasable/Programmable ROM) o Flash. Es la encargada de almacenar el código del programa que ejecutará el microprocesador.

- *Memoria de Datos*: es una memoria RAM (Random Acces Memory) que típicamente puede ser de 1, 2, 4, 8, 16 o 32 kilobytes.

- *Generador de Reloj*: cristal de cuarzo que produce unos impulsos con una determinada frecuencia y genera una señal oscilante. Esta frecuencia suele ir desde 1 a 40 MHz.

- *Interfaz de Entrada/Salida*: puertos paralelos, seriales (UARTs, Universal Asynchronous Receiver/Transmitter(, I2C (Inter-Integrated Circuit), Interfaces de periféricos seriales (SPIs, Serial Peripheral Interfaces), Red de Área de Controladores (CAN, Controller Area Network), USB (Universal Serial Bus), etc.

- Otras opciones:

    - Conversores Analógicos-Digitales (A/D, analog-to-digital) para convertir un nivel de voltaje en un cierto pin a un valor digital manipulable por el programa del microcontrolador. Estos Conversores A/D suelen tener una resolución típica de 10 bits, aunque existen versiones de 12, 16 o 32 bits.
    - Moduladores por Ancho de Pulso (PWM, Pulse Width Modulation) para generar ondas cuadradas de frecuencia fija pero con ancho de pulso variable. Aunque cualquier salida digital del microcontrolador puede ser programada para hacer esta función mediante el uso de interrupciones y temporizadores, muchos microcontroladores incluyen algunas salidas especialmente dedicadas a este efecto, lo cual simplifica su uso.

Para este trabajo desde un primer momento se decidió que el microcontrolador a usar fuera un Arduino ya que se amoldaba perfectamente a las características del trabajo. En los siguientes apartados se realizará una explicación más exhaustiva de las razones por las que se eligió Arduino.

2.5.1 Arduino

El Arduino es una plataforma de hardware y software *open source* que está basado en una placa que permite conectar sensores y actuadores mediante entradas y salidas analógicas y digitales y en un entorno de desarrollo basado en el lenguaje de programación Processing.

Figura 30: logo Arduino (Arduino)

Al ser open source tanto su diseño como su distribución es libre, es decir, puede utilizarse libremente para desarrollar cualquier tipo de proyecto sin tener que adquirir ningún tipo de licencia. El texto de la referencia de Arduino está publicado bajo la licencia *Creative Commons Reconocimiento-Compartir bajo la misma licencia 3.0*. los ejemplos de código de la referencia están liberados al dominio público.

2.5.1.1 Ventajas

Las razones por las que se eligió Arduino como plataforma sobre la que desarrollar el proyecto fueron cuatro:

- **Barato**: por apenas 20€ se puede conseguir una placa Arduino completamente funcional incluyendo un módulo de interfaz Serie-USB.
- **Popular**: la plataforma Arduino es ampliamente conocida por miles de desarrolladores en todo el mundo, desde profesionales hasta personas que desarrollan aplicaciones por *hobby*. Además existen tutoriales y foros que facilitan notablemente diferentes apartados de los proyectos.
- **Versátil**: una misma placa de Arduino puede servir para proyectos de domótica, robótica, control de sistemas o como programador de otros microcontroladores.
- **Open source**: se encuentran en la misma página web de Arduino los planos y esquemas de las diferentes placas, por lo que se puede modificar cualquier componente que se prefiera.

2.5.1.2 Partes de una placa Arduino

Para explicar las principales partes que componen una placa Arduino se va a utilizar como modelo la Arduino UNO. Las principales partes son:

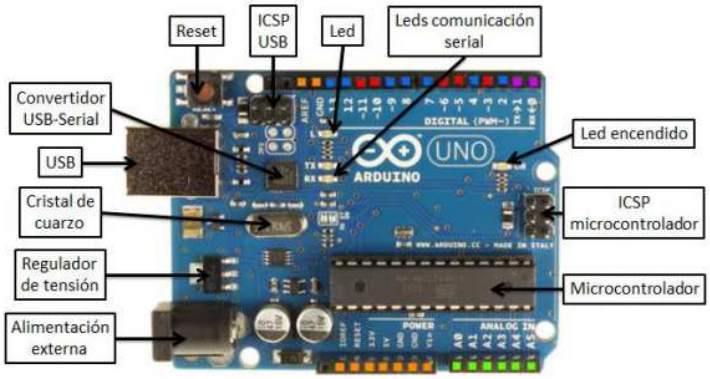


Figura 31: partes de una placa Arduino UNO (Arduino)

- **Pines**:
    - **Terminales de alimentación y de referencia** (naranja): a través de estos pines se puede alimentar a la placa al mismo tiempo que sirven como referencia de tensión para los circuitos.
    - **Terminales digitales** (azul, rojo, morado y verdes): estos pines tomarán valores de 0s y 1s. Se pueden configurar como punes de entrada o de salida. Las entradas analógicas también se pueden configurar como digitales.
    - **Terminales PWM** (azul): mediante estos pines se pueden generar señales PWM. El ciclo de trabajo de la señal se puede ajustar con una resolución de 1 byte (desde 0 hasta 255).
    - **Terminales puerto serie** (morado): estos pines permiten enviar y recibir datos de otros dispositivos mediante puerto serie.
    - **Terminales analógicos** (verde): estos terminales cuentan con conversadores A/D de 10 bits (desde 0 hasta 1023).

- **Microcontrolador**: las placas Arduino emplean generalmente los microcontroladores ATmega328 y ATmega2560. Son chips sencillos y de bajo coste que permiten el desarrollo de múltiples diseños.

- **Terminal ICSP microcontrolador**: permite programar el bootloader del microcontrolador ATmega y poder cargar los programas directamente en el microcontrolador sin tener que necesitar programadores externos.

Figura 32: programación de un Arduino UNO a través de otro dispositivo (arduino)

El *bootloader* es un conjunto mínimo de instrucciones que permanece almacenado en la memoria Flash del microcontrolador. Le permite interactuar con la *interface* de Arduino, interpretar los programas que se le cargan, recibir y enviar datos por los diferentes puertos o generar señales de control y permite comunicación USB.

- **LED encendido**: LED que indica si la placa tiene alimentación suficiente como para funcionar.

- **LEDs comunicación serial**: estos LEDs se encienden cuando hay una comunicación por el puerto serie de la placa. Si recibe un dato se encenderá el LED RX (*receive*) y si transmite un dato se encenderá el LED TX (*transmit*).

- **LED**: este LED está unido mediante una resistencia interna (resistencia *pull-up*) al terminal 13. Permite comprobar el correcto funcionamiento de la salida digital 13 sin necesidad de conectar ningún elemento externo a esta para limitar la corriente proporcionada por esta salida.

- **ICSP USB**: permiten emplear la placa de Arduino como un programador de otros microcontroladores.

- **Reset**: sirve para resetear el microcontrolador.

- **Convertidor Serie – USB**: este dispositivo permite la conversión de los datos que llegan por el USB a datos entendibles por el microcontrolador, es decir, transforma los datos a serie. Permite la programación directa del Arduino desde el ordenador.

- **Terminal USB**: permite tanto alimentar la placa como programarla

- **Cristal de cuarzo**: dispositivo que permite que lo microcontroladores operen a una cierta frecuencia. En Arduino esta frecuencia es de 8 a 16 MHz.

- **Regulador de tensión**: sirve para, independientemente de la tensión de alimentación de la placa, cada elemento interno de la placa obtenga o bien 3,3V ó 5V.

- **Alimentación externa**: permite alimentar la placa desde un dispositivo externo sin emplear un cable USB.

2.5.1.3 Lenguaje de programación

Las placas Arduino se programan mediante un lenguaje propio basado en el lenguaje de alto nivel **Processing**, aunque también es posible emplear otros lenguajes de programación y aplicaciones como C++, Java, Matlab o Python, y luego programarse mediante un compilador **AVR** (**A**lf (Egil Bogen) and **V**egard (Wollans)´s **R**ISC processor), el cual corresponde con la familia de microcontroladores de Atmel que incluyen las placas de Arduino.

Sin embargo, uno de los principales atractivos de Arduino es su interfaz de programación. El equipo de Arduino ha desarrollado una aplicación en el lenguaje Processing (que a su vez está basado en Java), que permite una programación muy sencilla a través de un lenguaje en pseudocódigo. Esto hace que el diseñador pueda desentenderse de aspectos engorrosos de la programación del microcontrolador y concentrarse en otros aspectos del proyecto.

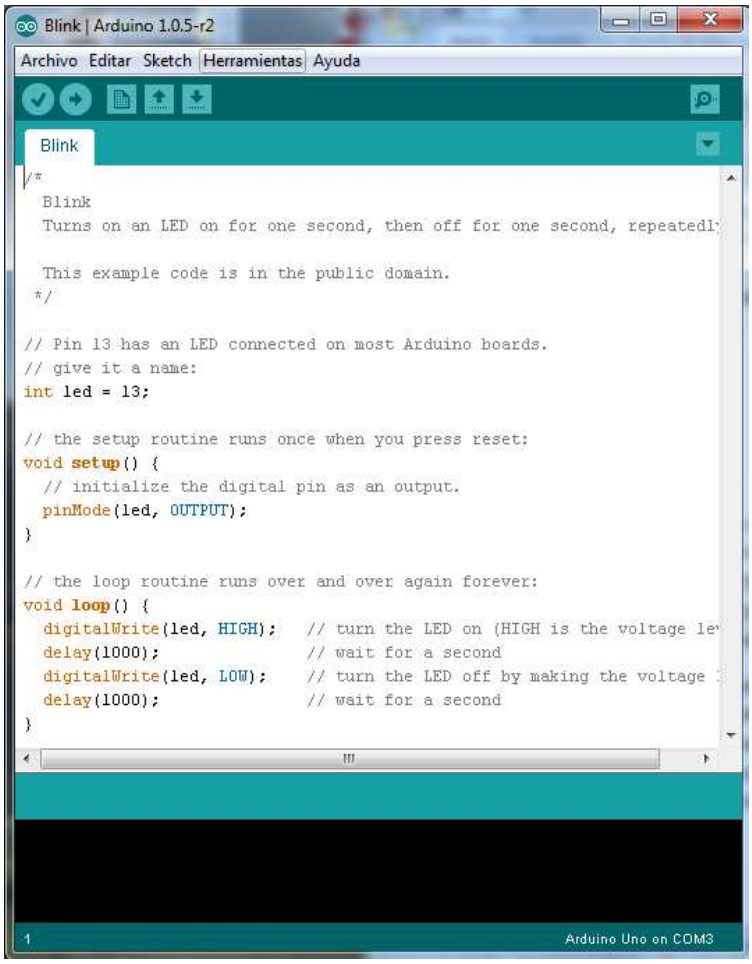


Figura 33: interfaz del IDE de Arduino (arduino)

La imagen anterior es la pantalla del entorno de desarrollo de Arduino. En la parte superior se encuentran las típicas opciones de todos los programas. En la siguiente fila se tienen los elementos propios de Arduino como verificar, cargar, nuevo, abrir, guardar y monitor serial. Este último sirve para ver qué datos se escriben en el puerto serie.

Después se encuentra el nombre del *sketch* (en este caso Blink) y el *sketch* propiamente dicho. Un *sketch* es como se denomina en Arduino a un programa. En la parte inferior hay una consola que informa de los errores producidos a la hora de compilar el código. Por último, se muestra el modelo de placa sobre el que se va a volcar el *sketch* y el puerto en el que está conectada al ordenador.

Todo *sketch* está formado por tres partes. Para explicar estas partes se va a utilizar como ejemplo un sencillo *sketch* que permite encender y apagar el LED que incorpora la placa.

- **Declaración de variables**

Lo primero que se debe hacer al empezar a escribir un *sketch* es definir las variables y constantes que formarán el *sketch*.

```
int led = 13;
```

En este caso se ha definido la variable LED como un entero y de valor 13.

- **Configuración de la placa**

Tras declarar las variables y constantes que va a emplear el *sketch* se procede a configurar la placa. Para ello se emplea la sintaxis *void setup()*. Las funciones más empleadas aquí son dos: por un lado está la función *pinMode* que permite definir los terminales como entradas o como salidas, y por otro lado está la función *Serial.begin* que establece la velocidad de datos en bits por segundo (baudioes) para la transmisión de datos en serie.

```
void setup() {
  pinMode(led, OUTPUT);
```

En este caso se ha definido el pin 13 (ya que se había declarado la variable LED con el valor de 13) como un pin de salida.

- **Bucle del programa principal**

Una vez configurada la placa se llega al bucle del *sketch*, es decir, lo que va a estar continuamente ejecutándose. Se emplea la sintaxis *void loop()*. Aquí las funciones más utilizadas son las funciones de lectura y de escritura de pines: *digitalWrite, digitalRead, analogWrite, analogRead*.

```
void loop() {
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
  delay(1000);
```

En este caso en primer lugar se escribe un 1 lógico en el pin 13, después se espera 1 segundo, después se escribe un 0 lógico en el pin 13 y se vuelve a esperar 1 segundo. Este bucle se estaría repitiendo indefinidamente mientras no reseteemos la placa o le quitemos la alimentación.

De esta manera en apenas 10 líneas ha sido posible controlar el LED interno de la placa Arduino. Una vez que se ha comprendido cómo se enciende y cómo se apaga un LED es posible poder manipular otros dispositivos como por ejemplo sensores o motores.

2.5.1.4 Modelos de Arduino

En la actualidad existen infinidad de modelos Arduino: UNO, Leonardo, Mega, LilyPad, Micro, Pro Mini…

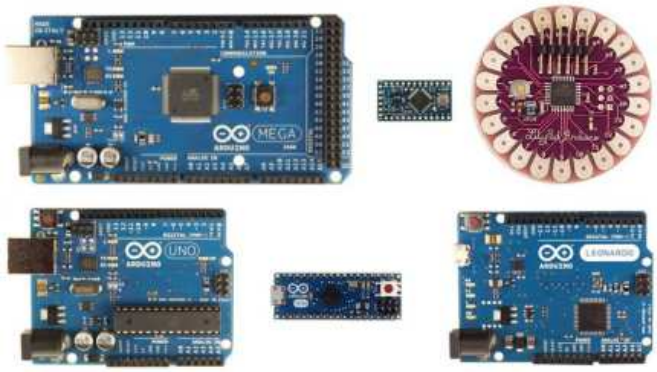


Figura 34: diferentes modelos de Arduino a escala. Fila superior de izquierda a derecha: Mega, Pro Mini y LilyPad. Fila inferior de izquierda a derecha: UNO, Micro y Leonardo (arduino)

Tras una primera criba se decidió que se iba a estudiar a fondo tres modelos de ellos: Mega, UNO y Pro Mini.

- **Arduino Mega**

El Arduino Mega es un microcontrolador basado en el ATmega2560. Dispone de 54 terminales digitales (14 de ellos se pueden emplear como salidas PWM), 16 terminales analógicos y 4 puertos serie.

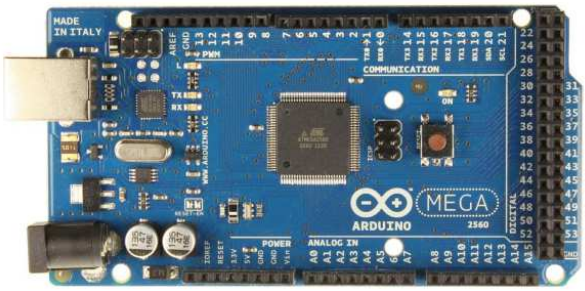


Figura 35: Arduino Mega (arduino)

Posee una memoria interna de tipo flash de 256kB que permite guardar códigos realmente extensos (8kB son empleados por el bootloader). Además cuenta con 8kB de SRAM y 4kB de EEPROM, al cual se puede acceder desde la librería EEPROM.

Todo esto hace que este modelo sea el más apropiado para los proyectos más complejos en los que se necesiten multitud de entradas y salidas o más memoria. Por otro lado se trata del Arduino de mayor tamaño.

- **Arduino UNO**

El Arduino UNO es un mircocontrolador basado en el ATmega328. Dispone de 14 terminales digitales (6 de ellos se pueden emplear como salidas PWM), 6 terminales analógicos y un puerto serie.

Figura 36: Arduino Pro Mini (arduino)

Posee una memoria interna de 32kB (0,5kB son empleados por el bootloader). Además cuenta con 2kB de SRAM y 1kB de EEPROM, al que también se puede acceder desde la librería EEPROM.

Este modelo es el más apropiado para proyectos en los que no se necesiten muchas entradas y salidas y el tamaño no sea un problema, ya que es el modelo más económico.

- **Arduino Pro Mini**

El Arduino Pro Mini es un microcontrolador que cuenta con las mismas características que el Arduino UNO: mismo controlador, mismo número de entradas y salidas digitales y analógicas y misma memoria.

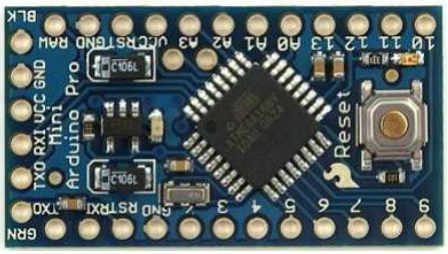


Figura 37: Arduino Pro Mini (arduino)

La gran diferencia del Arduino Pro Mini con respecto al Arduino UNO es su tamaño, ya que en muy poco espacio se dispone de prácticamente todas las opciones que ofrece el Arduino UNO. Esto hace que sea el modelo más apropiado a la hora de realizar proyectos en los que el tamaño sea un factor clave. Es necesario destacar que por razones de coste esta placa no incorpora el circuito integrado de interfaz Serie-USB, con lo cual será necesario disponer de hardware adicional para programarlo.

2.5.1.5 Elección del microcontrolador

Las características de los tres modelos estudiados se resumen en la siguiente tabla:

Modelo	Arduino Mega	Arduino UNO	Arduino Pro Mini
Micro	ATmega2560	ATmega328	ATmega328
Tensión de funcionamiento	5V	5V	5V
Tensión recomendada	7-12V	7-12V	7-12V
Tensión límite	6-20V	6-20V	6-20V
Corrientes pines entrada/salida	40mA	40mA	40mA
Pines digitales	54 (14 PWM)	16 (6 PWM)	16 (6 PWM)
Pines analógicos	16	6	6

Puertos serie	4	1	1
Memoria interna	256kB (8kB bootloader)	32kB (0,5kB bootlader)	32kB (0,5kB bootlader)
SRAM	8kB	2kB	2kB
EEPROM	4kB	1kB	1kB
Frecuencia	16MHz	16MHz	16MHz
Ventajas	Gran cantidad de entradas y salidas y mucha memoria	Económico	Reducida tamaño
Desventajas	Gran tamaño	-	Hay que soldar los pines y no dispone de conversor USB-SERIE

Tabla 1: comparativa entre el Arduino Mega, Arduino UNO y Arduino Pro Mini

Tras este análisis se desechó el Arduino Mega. Para este proyecto no se van a emplear multitud de sensores, por lo que se desaprovecharían gran cantidad de entradas y salidas. Por l contrario, por dimensiones del robot y necesidad de colocar las pilas y la base robótica en él, además del Arduino, hacen que su tamaño sea excesivamente grande.

Figura 38: Comparativa a escala tamaño (arduino)

Finalmente se decidió hacer el proyecto empleando el Arduino UNO debido a que su tamaño se ajusta adecuadamente a las dimensiones del robot, y es la opción más barata de las tres opciones. Además, utilizando esta placa no es necesario soldar los pines a la placa, como en el caso del Arduino Pro Mini, y no es necesario comprar un dispositivo externo para poder programarlo.

2.5.2 Arduino Motor Shield

El control de la tracción y la dirección del robot se va a realizar utilizando el Arduino. Debido a que el motor necesita mucha corriente y el Arduino no es capaz de proporcionárselo, se va a utilizar un driver de potencia para el Arduino, concretamente el ARDUINO MOTOR SHIELD.

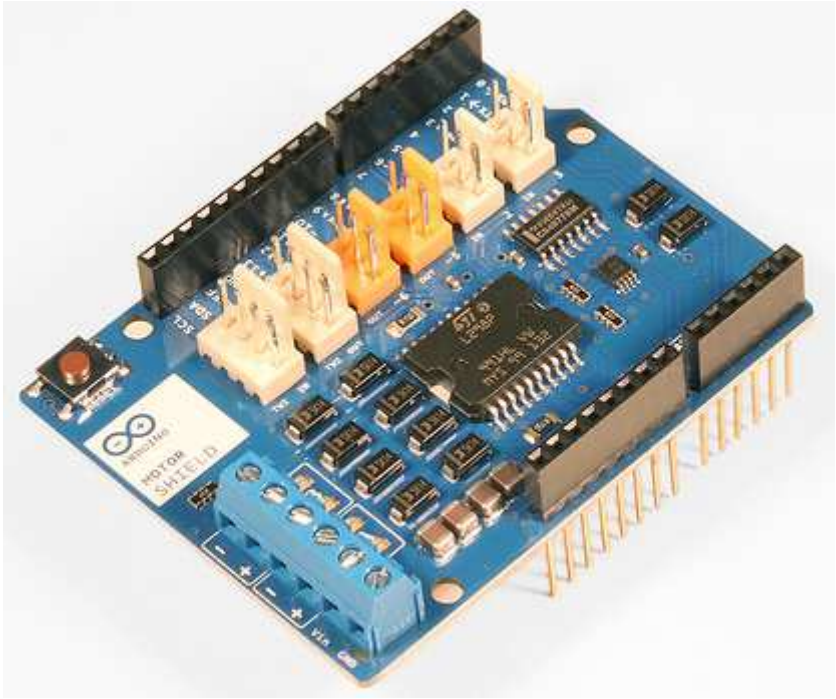


Figura 39: Arduino Motor Shield (arduino)

Este dispositivo se basa en el L298, que se trata de un controlador dual de puente completo diseñado para manejar cargas inductivas como relés, solenoides, motores de corriente continua y motores paso a paso. Permite controlar dos motores de corriente continua con la placa Arduino, controlando la velocidad y la dirección de cada uno independientemente. También permite medir la corriente absorbida por cada motor, entre otras características. Es compatible con TinkerKit, lo que significa que se puede crear rápidamente proyectos conectando módulos TinkerKit a la placa.

2.5.2.1 Resumen Arduino Motor Shield

Tensión de funcionamiento	De 5V a 12V
Controlador del motor	L298P, controla 2 motores de corriente continua o 1 motor de paso a paso
Corriente máxima	2A por canal o 4A máximo (con una fuente de alimentación externa)
Detección de corriente	1,65V/A

2.5.2.2 Alimentación

El Arduino Motor Shield debe ser alimentado sólo por una fuente de alimentación externa. Debido a que el circuito integrado L298 está montado en la placa, ésta tiene dos conexiones de alimentación separadas, uno para la lógica y otro para el motor. La corriente requerida por el motor muchas veces excede el máximo que puede ofrecer el USB. La fuente externa (no USB) puede venir también de un adaptador AC/DC o una batería. El adaptador se puede conectar enchufando un conector de 2,1mm en el conector de alimentación de la placa Arduino en el que está montado el protector del motor o mediante la conexión de los cables que conducen la alimentación a los terminales de tornillo Vin y GND, teniendo cuidado en respetar las polaridades.

Para evitar posibles daños a la placa Arduino donde está montado el escudo protector, se recomienda utilizar una fuente de alimentación externa que proporcione un voltaje entre 7 y 12V. Si el motor requiere más de 9V, se recomienda separar la alimentación del escudo y de la placa Arduino donde está montado el escudo. Esto es posible cortando el puente "Vin conexión" el cual está conectado en el lado posterior del escudo protector. El límite absoluto para el Vin que los soportan los terminales de tornillo es de 18V.

Los pines de alimentación son los siguientes:

- **Vin** en el bloque de terminales de tornillo, es la tensión de entrada al motor conectado al escudo. Una fuente de alimentación externa conectada a este pin también proporciona alimentación a la placa Arduino en donde está montado. Cortando el puente "Vin Connect" se consigue que esta alimentación solo llegue al motor.
- **GND**: tierra en el boque de terminales de tornillo.

El escudo puede proporcionar 2 amperios por canal, con un total de 4 amperios máximo.

2.5.2.3 Entradas y salidas

El escudo tiene dos canales separados, llamados A y B, de los cuales cada uno utiliza 4 pines del Arduino para controlar el motor. En total hay 8 pines en uso en este escudo. Se puede utilizar cada can por separado para controlar dos motores de corriente continua o combinarlos para controlar un motor paso a paso bipolar.

En la siguiente tabla se muestran los pines del escudo, separados por cada canal:

Función	Pines del canal A	Pines del canal B
Dirección	D12	D13
PWM	D3	D11
Freno	D9	D8
Detección de Corriente	A0	A1

Tabla 3: configuración de los pones de cada canal

Si necesitas más pines y los pines de Freno y Detección de Corriente no son necesarios se pueden deshabilitar estos pines cortando los respectivos puentes en la parte de atrás del escudo, y utilizarlos como pines digitales normales.

Las tomas adicionales del escudo se describen de la siguiente manera:

- **Terminales de tornillo** para conectar los motores y su fuente de alimentación.
- **2 conectores TinkerKit** para dos Entradas Analógicas (en blanco), conectados a A2 y A3.
- **2 conectores TinkerKit** para dos Salidas Analógicas (en naranja en la mitad), conectados a salidas PWM en los pines D5 y D6.
- **2 conectores TinkerKit** para el interfaz TWI (en blanco con 4 pines), uno para entrada y el otro para salida.

2.5.2.4 Conexión de motores

Motor de corriente continua. Se pueden controlar dos motores de corriente continua conectando los dos cables de cada uno en los terminales (+) y (-) de cada canal A y B. de esta manera se puede controlar la dirección estableciendo *HIGH* o *LOW* a los pines **DIR A** y **DIR B**, al igual que se puede controlar la su velocidad variando los valores del ciclo de trabajo de **PWM A** y **PWM B**. Los pines **Break A** y **Break B**, si tienen el valor *HIGH*, frenaran efectivamente los motores de corriente continua en lugar de dejar que vayan parándose cortando la alimentación. Se puede medir la corriente que pasa a través del motor de corriente continua leyendo los pines **SNS0** y **SNS1**. En cada canal habrá una tensión proporcional a la corriente medida, la cual puede ser leída como una entrada analógica normal, mediante la función *analogRead()* en las entradas analógicas A0 y A1. Por conveniencia, está calibrado para que cuando el canal está entregando la corriente máxima posible (que es de 2ª) la tensión sea de 3,3V.

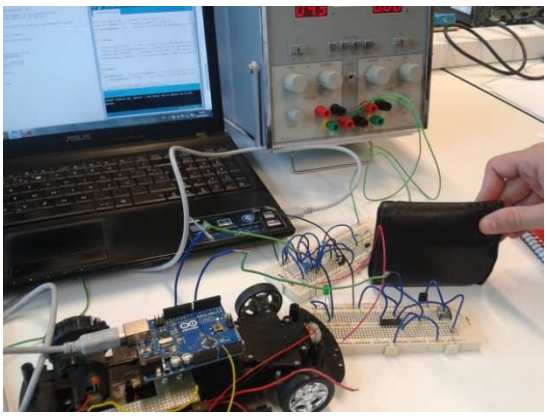
2.6 Lectura sensor infrarrojo mediante Arduino

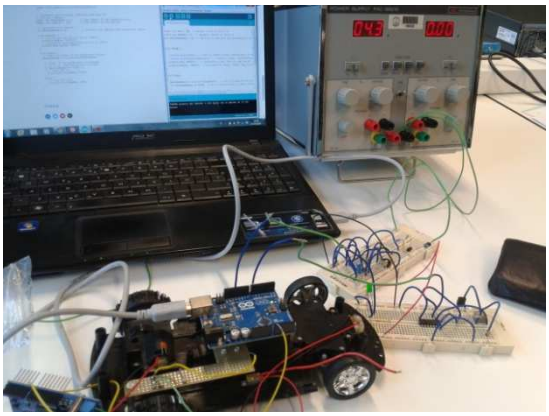
Una vez entendido el funcionamiento del Arduino y de la placa Arduino Motor Shield, se va a proceder a leer la información del sensor mediante el Arduino UNO. Para ello, se va a utilizar el programa llamado “lecturaSensorLed”.


```
const int led = 12;  // asigna a led el valor 13
const int sensor = 7;  // asigna a sensor el valor 8
int entradaSensor = 0;  // declara la variable que determina si se recibe la señal infrarroja o no


void setup()
{
   pinMode(led, OUTPUT);  // configura el led (PIN 13) como sailda
  pinMode(sensor, INPUT);  // configura el sensor (PIN8) como entrada
}

void loop()
{
  entradaSensor = digitalRead(sensor);  // lee el estado de la entrada del sensor
  if (entradaSensor == HIGH)  // si el sensor recibe la señal infrarroja
  {
    digitalWrite(led, HIGH);  // enciende el LED
    delay(200);  // espera 200 milisegundos
  }
  else
  {
    digitalWrite(led, LOW);  // apaga el LED
    delay(200);   // espera 200 milisegundos
  }
}
```

Figura 40: programa utilizado para leer la salida del sensor infrarrojo (elaboración propia)

Con este programa se lee el valor del sensor infrarrojo por el pin 7, que inicialmente se ha definido como una entrada digital. Se monta un circuito con un LED verde y una resistencia para limitar la corriente en el pin 12, que ha sido definido como salida digital. Cuando el Arduino lee 5V en el pin 7, la salida del pin 12 se pone a 5V, por lo que el LED se enciende; al contrario, cuando el pin 7 ve 0V, el pin 12 se pone al mismo voltaje y el LED está apagado. La sensibilidad de este programa es de 200 milisegundos, es decir, actualiza el valor del sensor cada 200 milisegundos debido a los delays que se han puesto. Esto se debe a que para este programa no es necesaria una mayor rapidez, y así se hace trabajar menos al Arduino, aunque no habría ningún problema en realizar el mismo programa sin el delay.





Figuras 41 y 42: lectura de la salida del sensor infrarrojo con y sin obstáculo (elaboración propia)

2.7 Lectura posición espejo mediante Arduino

Para el correcto funcionamiento del robot es necesario tener un control de la posición del espejo, como ya se ha comentado anteriormente. Para tener dicho control se utiliza el interruptor óptico ranurado OPB620, y el circuito con las dos resistencias calculadas. La salida del pin 3 ha de ser leída mediante el Arduino. Se utiliza el siguiente programa para comprobar el correcto funcionamiento en la lectura:

OPBsimple | Abrir

```
// OPB simple

int obstaculo;  // declara la variable obstaculo
int Pin=12;  // asigna a Pin el valor 12

void setup()
{
  Serial.begin(9600);  // abre el Puerto serie configurando la velocidad en 9600bps
  pinMode(Pin, INPUT);  // configura el PIN 12 como entrada
}

void loop()
{
  obstaculo=digitalRead(Pin);  // lee el estado de la entrada del sensor
  Serial.print(obstaculo);  // envía el valor 'obstaculo' al puerto
  Serial.print(",");  // envía una coma al puerto
  delay(500);  // espera medio segundo
```

Figura 43: programa utilizado para leer la salida del interruptor óptico ranurado (elaboración propia)

Con este programa se lee mediante el Arduino el estado del sensor, y se muestra en el PC su valor: será 1 cuando no hay obstáculo, y 0 cuando sí lo hay. Se ha de recordar que el disparador invierte el valor del sensor. Se utiliza un retardo de medio segundo para que no se tomen demasiadas medidas por segundo y se pueda observar mejor el cambio cuando existe un obstáculo y cuando éste desaparece:

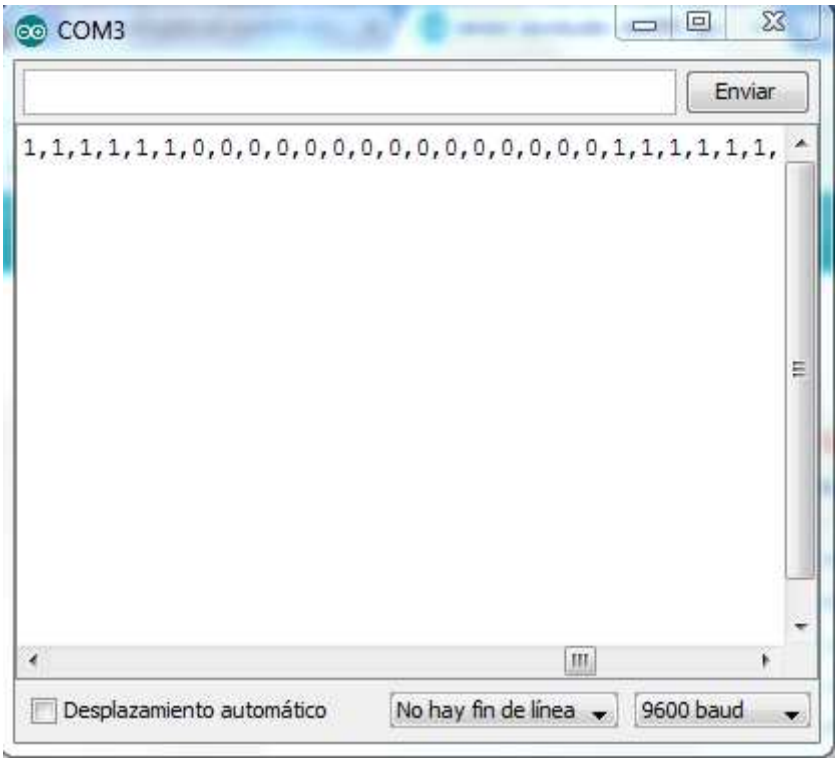


Figura 44: valores leídos por el Arduino y mostrados por el puerto serie en el PC (elaboración propia)

Una vez comprobado el correcto funcionamiento del interruptor óptico y su lectura, se procede a montarlo y comprobar que controla correctamente el número de vueltas que da la base giratoria y en qué posición se encuentra en cada momento. Para ello se utiliza el siguiente programa:

Figura 45: programa utilizado para verificar el correcto funcionamiento del control de la estructura giratoria (elaboración propia)

Mediante este programa se comprueba el correcto funcionamiento del circuito diseñado para el control de la estructura giratoria. Se observa, mandando la variable “vuelta” por el puerto serie y visualizándolo en el PC, que se incrementa cada vez que el espejo completa una vuelta.

2.8 Prueba dirección Arduino

Una vez dominado la parte de los sensores del robot, es decir, la lectura de la luz infrarroja y la posición en la que se encuentra en cada momento, se procede a comprobar el funcionamiento de los motores los cuales controlan la dirección y la tracción.

En primer lugar se va a estudiar cómo se controla la dirección. Para ello, se deben conectar los dos cables del motor delantero a la salida B del Arduino Motor Shield, el rojo al positivo y el negro al negativo. Como ya se ha explicado antes, existen tres situaciones diferentes:

- Para ir hacia la derecha
    - DIR B = LOW
    - Break B = LOW
    - PWM B = 255

- Para ir hacia la izquierda
    - DIR B = HIGH
    - Break B = LOW
    - PWM B = 255

- Para ir hacia recto
    - Break B = HIGH

Se realiza un programa para comprobar el correcto funcionamiento de la dirección. Dicho programa sigue la siguiente secuencia repetidamente:

1. Gira hacia la izquierda durante un segundo
2. Coloca las ruedas en dirección recta durante un segundo
3. Gira hacia la derecha durante un segundo
4. Coloca las ruedas en dirección recta durante un segundo

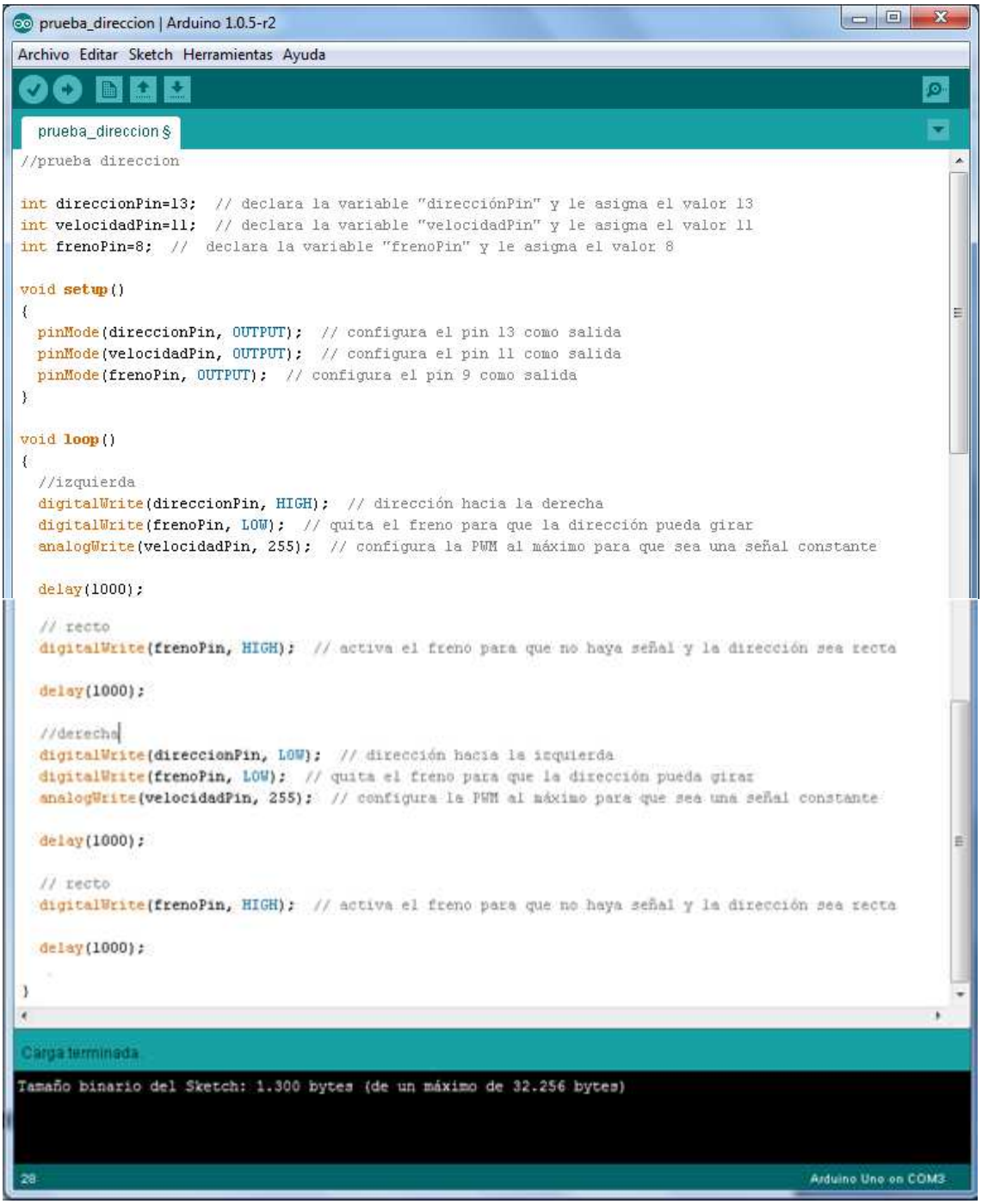


Figura 46: programa utilizado para comprobar el correcto funcionamiento de la dirección (elaboración propia)

2.9 Prueba tracción motor

Una vez estudiada la dirección del robot se va a probar el funcionamiento de su tracción. Para ellos se conectan los cables del motor a canal A respetando las polaridades y se alimenta la placa mediante las pilas. Al igual que en el caso de la dirección, con la tracción también tenemos tres estados diferentes:

- Hacia delante
  - DIR A = HIGH
  - Break A = LOW
  - PWM A = 255

- Hacia atrás
  - DIR A = LOW
  - Break A = LOW
  - PWM A = 255

- Parada
  - Break A = HIGH

Una vez sabido esto, se carga el siguiente programa en el Arduino (llamado “motores_prueba”):

motores_prueba

```
//motores

int direccionPin=12;  // declara la variable direccionPin y le asigna el valor 12
int velocidadPin=3;  // declara la variable velocidadPin y le asigna el valor 3
int frenoPin=9;  // declara la variable frenoPin y le asigna el valor 9

void setup()
{
  pinMode(direccionPin, OUTPUT);  //configura el pin 12 (dirección) como salida
  pinMode(velocidadPin, OUTPUT);  //configura el pin 3 (velocidad) como salida
  pinMode(frenoPin, OUTPUT);  //configura el pin 9 (freno) como salida
}

void loop()
{
  //hacia delante
  digitalWrite(direccionPin, HIGH);  // dirección hacia delante
  digitalWrite(frenoPin, LOW);  // quita el freno
  analogWrite(velocidadPin, 50);  // configura la velocidad

  delay(3000);  // espera 3 segundos

  digitalWrite(frenoPin, HIGH);  // activa el freno

  delay(1000);  //espera 1 segundo

  //hacia atrás
  digitalWrite(direccionPin, LOW);  // dirección hacia atrás
  digitalWrite(frenoPin, LOW);  // quita el freno
  analogWrite(velocidadPin, 50);  // configura la velocidad

  delay(3000);  // espera 3 segundos

  digitalWrite(frenoPin, HIGH);  // activa el freno

  delay(1000);  // espera 1 segundo
}
```

Figura 47: programa utilizado para comprobar el correcto funcionamiento de la dirección (elaboración propia)

Este programa consiste en hacer andar al robot tres segundos hacia delante y luego pararlo durante 1 segundo. Después de este segundo parado, se hace andar al robot hacia atrás durante otros 3 segundos a la misma velocidad a la que lo ha hecho hacia delante y se para durante otro segundo, y así repetidamente.

2.10 Montaje y programa completo

Después de comprobar el correcto funcionamiento de todas las partes del robot por separado, se procede a montarlos sobre la plataforma robótica.

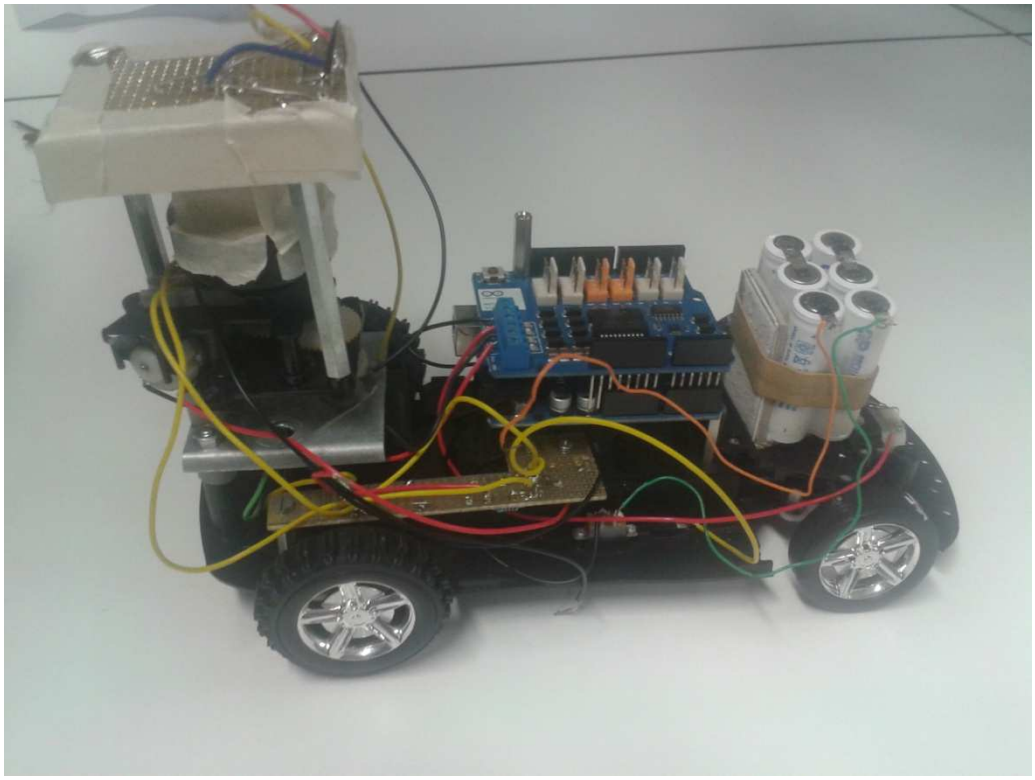


Figura 48: montaje robot (elaboración propia)

Lo primero que se hace es alimentar el Arduino utilizando las 6 pilas conectadas en serie a través de un interruptor, el cual nos dará una mayor comodidad en la conexión y desconexión de la alimentación. Las conexiones de cada uno de los circuitos con el Arduino a través de la placa Arduino Motor Shield se debe realizar de la siguiente manera:

- El circuito receptor de la luz infrarroja será alimentado a 5V mediante el pin de 5V y el de GND que tiene el Arduino. La salida del circuito receptor irá a una entrada digital, en concreto el pin 7.

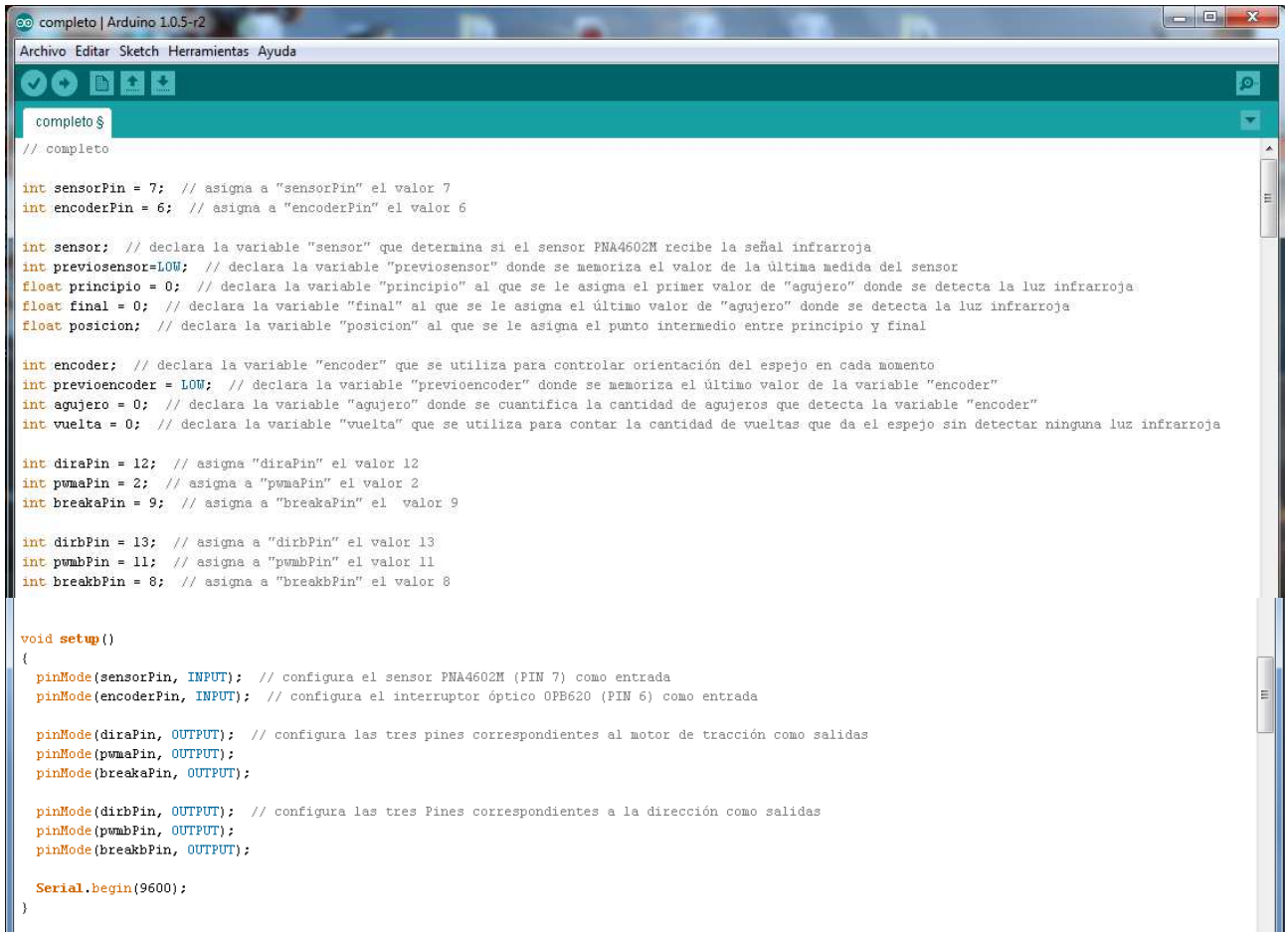
- El circuito que controla el giro del espejo será alimentado en paralelo con la alimentación del Arduino, es decir, con las 6 pilas conectadas en serie, a una tensión de 7,2V. El cable positivo (rojo) irá al terminal Vin, mientras que el cable negro irá al terminal GND.

- El robot encargado de hacer girar a la estructura giratoria será alimentado con el Arduino a una tensión de 3,3V, ya que a esa tensión su velocidad angular es adecuada. el cable rojo irá conectado al pin de 3,3V que tiene el Arduino, y el negro se conectará al pin GND.

- El motor de tracción del robot va a ser alimentado con los terminales del canal A de la placa Arduino Motor Shield. El cable rojo irá conectado con el terminal (+), mientras que el negro se conectará al terminal (-).

- La dirección del robot se va a alimentar con los terminales del canal B de la placa Arduino Motor Shield. Los cables rojo y negro irán conectados a los terminales (+) y (–) del canal respectivamente.

A continuación se debe realizar el programa completo, de tal manera que se logre que el robot se dirija hacia la luz infrarroja. El programa utilizado es el siguiente:




```
// completo

int sensorPin = 7;  // asigna a "sensorPin" el valor 7
int encoderPin = 6;  // asigna a "encoderPin" el valor 6

int sensor;  // declara la variable "sensor" que determina si el sensor PNA4602M recibe la señal infrarroja
int previosensor=LOW;  // declara la variable "previosensor" donde se memoriza el valor de la última medida del sensor
float principio = 0;  // declara la variable "principio" al que se le asigna el primer valor de "agujero" donde se detecta la luz infrarroja
float final = 0;  // declara la variable "final" al que se le asigna el último valor de "agujero" donde se detecta la luz infrarroja
float posicion;  // declara la variable "posicion" al que se le asigna el punto intermedio entre principio y final

int encoder;  // declara la variable "encoder" que se utiliza para controlar orientación del espejo en cada momento
int previoencoder = LOW;  // declara la variable "previoencoder" donde se memoriza el último valor de la variable "encoder"
int agujero = 0;  // declara la variable "agujero" donde se cuantifica la cantidad de agujeros que detecta la variable "encoder"
int vuelta = 0;  // declara la variable "vuelta" que se utiliza para contar la cantidad de vueltas que da el espejo sin detectar ninguna luz infrarroja

int diraPin = 12;  // asigna "diraPin" el valor 12
int pwmaPin = 2;  // asigna a "pwmaPin" el valor 2
int breakaPin = 9;  // asigna a "breakaPin" el  valor 9

int dirbPin = 13;  // asigna a "dirbPin" el valor 13
int pwmbPin = 11;  // asigna a "pwmbPin" el valor 11
int breakbPin = 8;  // asigna a "breakbPin" el valor 8


void setup()
{
  pinMode(sensorPin, INPUT);  // configura el sensor PNA4602M (PIN 7) como entrada
  pinMode(encoderPin, INPUT);  // configura el interruptor óptico OPB620 (PIN 6) como entrada

  pinMode(diraPin, OUTPUT);  // configura las tres pines correspondientes al motor de tracción como salidas
  pinMode(pwmaPin, OUTPUT);
  pinMode(breakaPin, OUTPUT);

  pinMode(dirbPin, OUTPUT);  // configura las tres Pines correspondientes a la dirección como salidas
  pinMode(pwmbPin, OUTPUT);
  pinMode(breakbPin, OUTPUT);

  Serial.begin(9600);
}
```

```
void loop()
{
  //ENCODER
  encoder = digitalRead(encoderPin);  // lee el estado del encoder mediante el interruptor óptico OPB620
  if (encoder == HIGH)  // si el OPB620 detecta un agujero
  {
    if (previoencoder == LOW)  // si en la anterior medida no detectó un agujero
    {
      agujero++;  // incrementa agujero
      previoencoder = HIGH;  // asigna el valor "HIGH" a "previoencoder" para que sólo incremente "agujero" una vez por cada agujero
    }
  }
  if (encoder == LOW) // si el OPB620 no detecta un agujero
  {
    previoencoder = LOW;  // se memoriza el valor de "encoder"
  }
  if (agujero >= 38)  // si "agujero" llega a 38
  {
    agujero = 0;  // inicializa "agujero"
    vuelta++;  // incrementa "vuelta"
  }


  // SENSOR INFRARROJO
  sensor = digitalRead(sensorPin);  // lee el estado de la salida del sensor PNA4602M
  if ((sensor == HIGH) && (previosensor == LOW))  // si el sensor detecta la luz infrarroja y en la anterior medida no la detectaba
  {
    principio = agujero; // asigna el valor "agujero" a la variable "principio", indicando la orientación donde empieza a detectar la luz infrarroja
    vuelta = 0;  // inicializa la variable "vuelta"
  }
  if ((sensor == LOW) && (previosensor == HIGH))  // si el sensor no detecta la luz infrarroja y en la anterior medida si la detectaba
  {
    previosensor = LOW;  // asigna el valor LOW a "previosensor"
    final = agujero;  //  asigna el valor "agujero" a la variable final, indicando la orientación a la que deja de detectar la luz infrarroja

    if (final > principio)  // si el valor de "final" es mayor que "principio"
    {
      posicion = (principio+final)/2;  // calcula la orientación promedio de donde se recibe la luz infrarroja
    }
    else
    {
      final = final + 38;  // calcula la orientación promedio de donde se recibe la luz infrarroja
      posicion = ((principio+final)/2)-38;
      if (posicion < 0)
      {
        posicion=posicion+38;
      }
    }
    principio = 0;  // inicializa la variable "principio"
    final = 0;  // inicializa la variable "final"
  }
  previosensor = sensor;  // memoriza la última medición del sensor


  // tracción y dirección
  if (vuelta > 5)  // si el espejo ha dado 5 vueltas completas sin detectar ninguna luz
  {
    digitalWrite(breakaPin, HIGH);  // para motor
  }

 if (posicion>=0 && posicion<l2)  // si el espejo está orientado entre el agujero 0 y el l2
  {
    digitalWrite(diraPin, LOW);  // configura la tracción del motor hacia atras
    digitalWrite(breakaPin, LOW);  // desactiva el freno
    analogWrite(pwmaPin, 255);  // configura la velocidad del motor mediante una PWM

    digitalWrite(dirbPin, LOW);  // configura la dirección hacia la izquierda
    digitalWrite(breakbPin, LOW);  // desactiva el freno
    analogWrite(pwmbPin, 255);  // configura la PWM al máximo para que sea una señal constante, no escalonada
  }


  if (posicion>=l2 && posicion <l7)  // si el espejo está orientado entre el agujero l2 y el l7
  {
    digitalWrite(diraPin, HIGH);  // configura la tracción del motor hacia adelante
    digitalWrite(breakaPin, LOW);  // desactiva el freno
    analogWrite(pwmaPin, 255);  // configura la velocidad del motor mediante una PWM

    digitalWrite(dirbPin, HIGH);  // configura la dirección hacia la derecha
    digitalWrite(breakbPin, LOW);  // desactiva el freno
    analogWrite(pwmbPin, 255);  // configura la PWM al máximo para que sea una señal constante
  }
```

```
  if (posicion>=17 && posicion <=21)  // si el espejo está orientado entre el agujero 17 y el 21
  {
    digitalWrite(diraPin, HIGH);  // configura la tracción del motor hacia adelante
    digitalWrite(breakaPin, LOW);  // desactiva el freno
    analogWrite(pwmaPin, 255);  // configura la velocidad del motor mediante una PWM

    digitalWrite(breakbPin, HIGH);  // activa el freno, para que la dirección sea recta
  }


    if (posicion>21 && posicion <=26)  // si el espejo está orientado entre el agujero 21 y el 26
  {
    digitalWrite(diraPin, HIGH);  // configura la tracción del motor hacia adelante
    digitalWrite(breakaPin, LOW);  // desactiva el freno
    analogWrite(pwmaPin, 255);  // configura la velocidad del motor mediante una PWM

    digitalWrite(dirbPin, LOW);  // configura la dirección hacia la izquierda
    digitalWrite(breakbPin, LOW);  // desactiva el freno
    analogWrite(pwmbPin, 255);  // configura la PWM al máximo para que sea una señal constante
  }


  if (posicion>26 && posicion <38)  // si el espejo está orientado entre el agujero 26 y el 38
  {
    digitalWrite(diraPin, LOW);  // configura la tracción del motor hacia atrás
    digitalWrite(breakaPin, LOW);  // desactiva el freno
    analogWrite(pwmaPin, 255);  // configura la velocidad del motor mediante una PWM

    digitalWrite(dirbPin, HIGH);  // configura la dirección hacia la derecha
    digitalWrite(breakbPin, LOW);  // desactiva el freno
    analogWrite(pwmbPin, 255);  // configura la PWM al máximo para que sea una señal constante
  }


}
```

106 Arduino Uno on COM3

Figura 49: programa comleto Arduino (elaboración propia)

Con este programa lo primero se realiza es detectar en qué dirección de los 360° se recibe la luz infrarroja. Como el disco tiene 38 ranuras, los 360° se dividen entre 38, lo cual se tienen 38 zonas de aproximadamente 9,5° cada una. Como el detector no detecta la luz en una única zona, sino que lo hace en varias zonas consecutivas, se realiza la media de todas las zonas donde se ve la luz. Mediante el control de la posición del espejo se detecta la zona en la que se empieza a detectar la luz infrarroja y la zona en la que se deja de verla, y mediante una sencilla operación matemática se obtiene la posición media.

Una vez obtenida posición de la luz, se indican diferentes direcciones y tracción, con el fin de dirigirnos a ella. Existen diferentes casos:

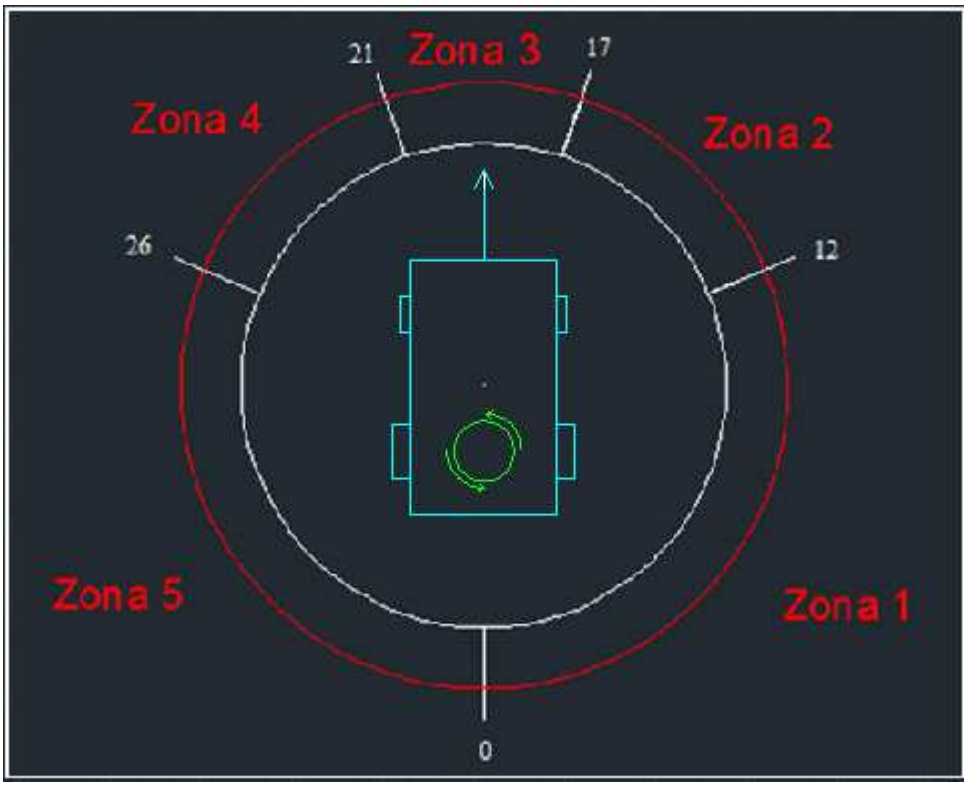


Figura 50: diferentes zonas de detección de luz infrarroja (elaboración propia)

- **Zona 1**: es cuando el espejo detecta la luz en la zona trasera derecha del robot. En este caso el coche dará marcha atrás con la dirección hacia la izquierda hasta que se detecte la luz en la zona 2.

- **Zona 2**: se trata de cuando la luz proviene de la parte delantera derecha del robot. La tracción será hacia delante, y la dirección hacia la derecha.

- **Zona 3**: en esta zona la luz viene de frente, por lo que la dirección será hacia delante, al igual la tracción.

- **Zona 4**: se trata de cuando la luz proviene de la parte delantera izquierda del robot. La tracción será hacia delante, mientras que la dirección será hacia la izquierda.

- **Zona 5**: es cuando la luz es detectada en la zona trasera izquierda del robot. El coche dará marcha atrás con la dirección hacia la derecha hasta que la luz sea detectada en la zona 4.

Se debe tener en cuenta que, al no tener una referencia de inicio de vuelta, es necesario marcar una posición inicial en concreto para el correcto funcionamiento del robot. Por eso, el robot debe siempre empezar su funcionamiento con el espejo en la posición 0, es decir, mirando hacia atrás. Es muy importante realizar este ajuste manualmente antes de cada uso del robot.

Además, se ha puesto un control para que el robot no quede todo el rato en funcionamiento: si el espejo realiza 5 vueltas sin detectar ninguna luz, el robot se parará automáticamente hasta que vuelva a detectar la luz infrarroja y se dirija hacia ella.

3. CONCLUSIONES

Aunque se haya conseguido un correcto funcionamiento del robot, existen muchos aspectos en los que se puede mejorar su funcionamiento, sobretodo en el aspecto de la precisión y evitar errores.

En primer lugar, el control de la posición no tiene ningún punto de referencia, por lo que si algún agujero del disco ranurado no es detectado por el OPB620, este error será arrastrado constantemente, y si ocurre el mismo error unas cuantas veces el resultado de dicho control será desastroso, ya que aunque el receptor detecte la luz, el Arduino lo leerá en una posición totalmente diferente a la que en realidad se encuentra el emisor de luz, por lo que el robot se moverá a otro sitio totalmente diferente al que debería hacerlo.

Además, es necesario colocar manualmente el espejo en una concreta posición (llamada posición cero en este trabajo) para que concuerden las posiciones detectadas con las que lee el Arduino.

Por último, la emisión de luz infrarroja es muy esparcida, no va en una única dirección. Por lo tanto, sería conveniente estudiar mejor la dirección de dicha luz y diseñar un apantallamiento adecuado, con el fin de evitar errores de detección de luz cuando no es debido.

3.1 Posibles mejoras

Una posible mejora que se puede realizar en este proyecto es la de realizar PCBs para el montaje de los circuitos. De esta manera, los circuitos montados resultan de menor tamaño y sus conexiones son más limpias y seguras. Además, realizando un PCB para la fuente de radiación infrarroja, es posible encapsularlo, como si fuera un mando de la televisión o un mando abridor de garajes. De esta manera, se podría evitar posibles desconexiones que se pueden realizar en los elementos, debido que éstas están todas a la vista en la protoboard. Además, sería aconsejable diseñar un chasis para el robot, dejando siempre el espejo a la vista para poder recibir la luz infrarroja.

Otro aspecto en el que se puede mejorar es el control del giro de la estructura giratoria. El problema que tiene el sistema empleado es que cuenta las vueltas según el número de agujeros que detecta, y si falla en la detección de un agujero arrastra el error sin corregirlo. Esto se podría arreglar utilizando otro OPB620 y un único agujero a otra profundidad diferente a la de los 38 agujeros utilizados en este proyecto. Así, cada vez que este segundo interruptor óptico detectase el agujero, se sabría que la vuelta ha concluido, independientemente de que no se hayan contabilizado los 38 agujeros, y así no se arrastra el error constantemente.

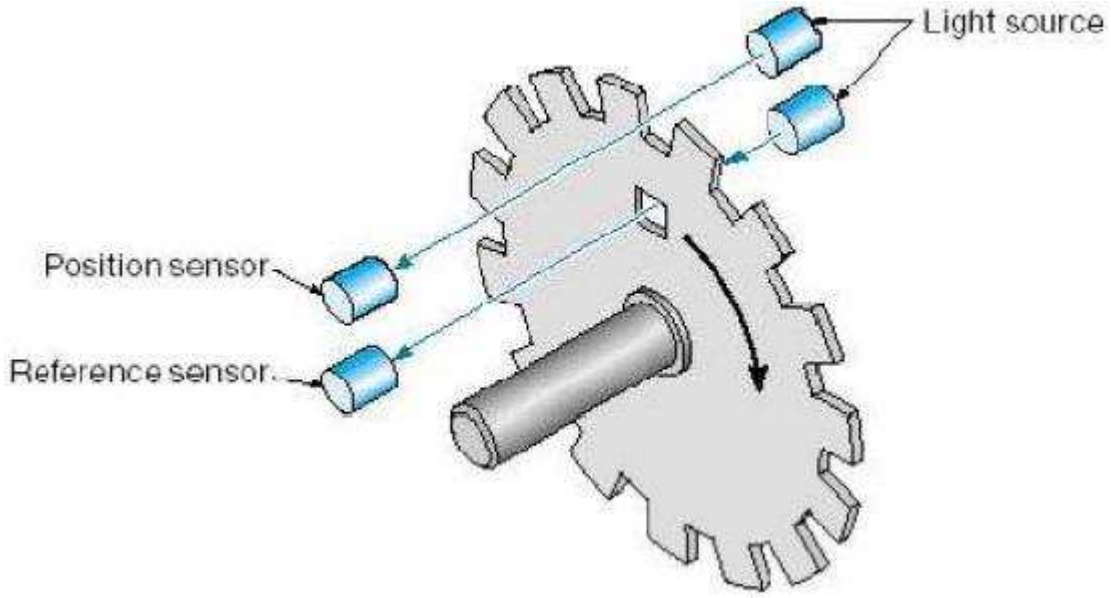


Figura 51: Mejora del control de giro (robopet)

Por lo que respecta a la recepción infrarroja, también se podrían mejorar el apantallamiento de la luz y el posicionamiento del espejo para una mayor eficacia. Para ello, sería recomendable diseñar un apantallamiento firme utilizando un material el cual la luz infrarroja no pueda atravesar.

4. BIBLIOGRAFÍA

Libros:

Arduino Cookbook. Michael Margolis

Getting started with Arduino. Massimo Banzi

Páginas webs:

www.arduino.cc

www.geekfactory.mx

www.adafruit.com

www.bricogeek.com

es.rs-online.com/web/

www.superrobotica.com

www.ideas-tecnologicas.blogspot.com.es

www.alldatasheets.com

www.unavarra.es

www.robotc.cl

http://lra.unileon.es/es

www.bitsdelocos.es

www.righto.com

PRESUPUESTO

En este apartado se va a calcular el presupuesto del robot, teniendo en cuenta todos los elementos utilizados. Hay que tener en cuenta que tanto la base del robot proveniente de un juguete (exactamente un coche teledirigido), así como parte de la estructura giratoria no son elementos que se hayan comprado, sino que han sido extraídos de otros juguetes. Es por esto que se desconoce su precio real, por lo que se hace una estimación aproximada, para tener un presupuesto aproximado del robot completo.

Componente	Fabricante	Unidades	Precio/unidad (€)	Precio (€)
FUENTE INFRARROJA				
LED IR OP165	OPTEK	1	0,74	0,74
Interruptor		1	1,45	1,45
Regulador L7805CV	StMicroelectronics	1	0,48	0,48
Temporizador NE555N	STMicroelectronics	1	0,274	0,274
Transistor BC548B	Fairchild Semiconductor	1	0,042	0,042
Pila 9V	Duracell	1	1,6	1,6
Potenciómetro	Piher	1	0,39	0,39
Resistencia 100Ω	RS	1	0,019	0,019
Resistencia 2kΩ	RS	1	0,02	0,02
Resistencia 4k7Ω	RS	1	0,019	0,019
Condensador 10nF	Murata	2	0,318	0,636
Condensador 10µF	Panasonic	1	0,078	0,078
Encapsulado LED		1	0,2	0,2
Placa de prueba	Elditest	1	16,49	16,49
TOTAL FUENTE INFRARROJA				**22,438**

CIRCUITO SENSOR INFRARROJO				
Sensor IR PNA4602M	Panasonic	1	1,48	1,48
Disparador HEF40106	NXP Semiconductors	1	0,64	0,64
Resistencia 100kΩ	RS	1	0,02	0,02
Condensador 0,1µF	RS	2	0,497	0,994
Condensador 820nF	AVX	1	0,634	0,634
TOTAL FUENTE INFRARROJA				**3,768**

ESTRUCTURA GIRATORIA				
Base giratoria		1	5	5
Espejo		1	0,2	0,2
Interruptor óptico OPB620	Optek	1	2,93	2,93
Disco rasurado		1	1,87	1,87
Resistencia 560Ω	RS	1	0,015	0,015
Resistencia 68kΩ	RS	1	0,019	0,019
TOTAL ESTRUCTURA GIRATORIA				**10,034**

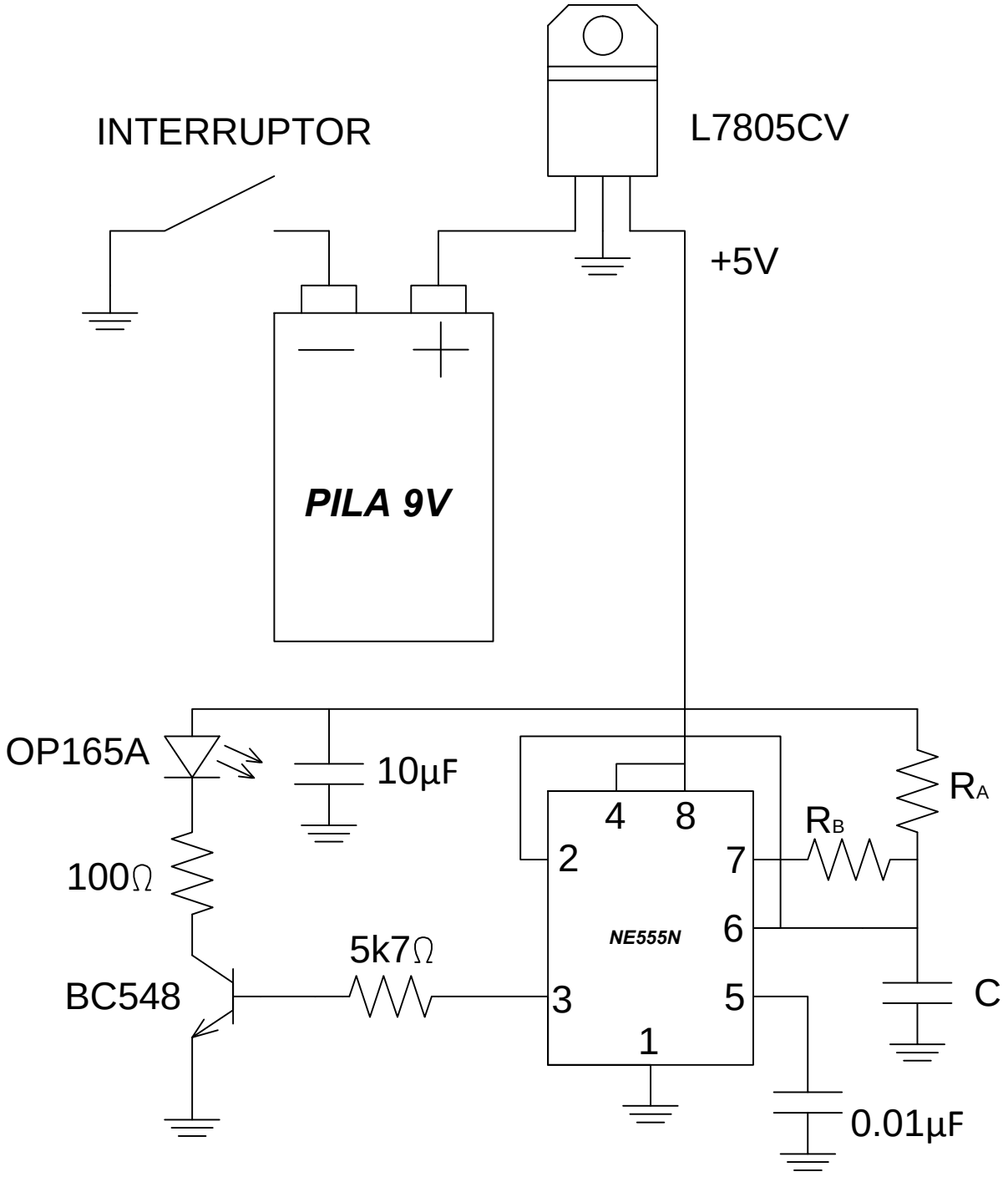
MICROCONTROLADOR				
Arduino UNO	Arduino	1	22	22
Arduino Motor Shield	Arduino	1	33,4	33,4
TOTAL MICROCONTROLADOR				**55,4**

OTROS ELEMENTOS				
Tornillería , barras y base metálica		1	3	3
Placas	Roth Elektronik	1	6,13	6,13
Pilas recargables AA	GP	6	3,1	18,6
Cables		3m	6,91€/100m	0,2073
Base coche teledirigido		1	20	20
TOTAL OTROS ELEMENTOS				**47,9373**

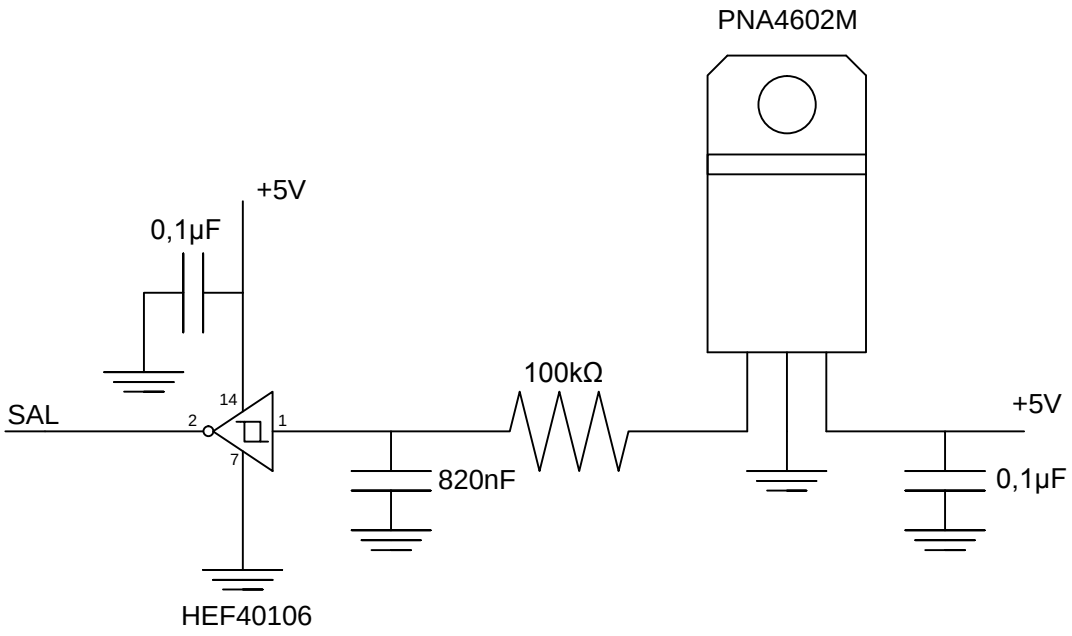
TOTAL ROBOT	**139,58**

Tabla 4: presupuesto

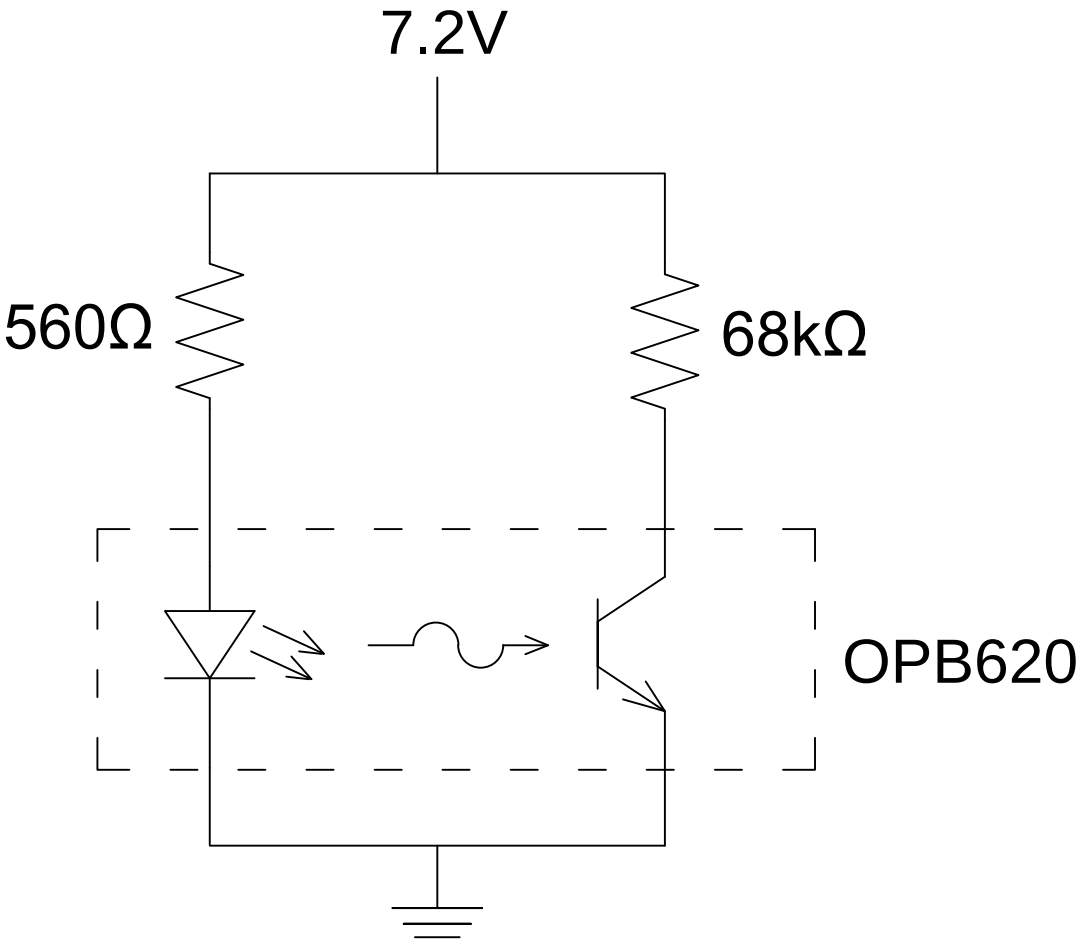
PLANOS



Nafarroako Unibertsitate Publikoa / Universidad Pública de Navarra	**E.T.S.I.I.T.** **INGENIERO ELÉCTRICO Y ELECTRÓNICO**	DEPARTAMENTO: **DEPARTAMENTO DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA**
PROYECTO: **VISOR INFRARROJO DE 360º PARA ROBOT BASADO EN ARDUINO**		REALIZADO: **GAGO BALDA, JAVIER** FIRMA:
PLANO: **ESQUEMA ELÉCTRICO DE LA FUENTE DE RADIACÍON INFRARROJA**		FECHA: **27/06/2014** Nº PLANO: **1º**



Nafarroako Unibertsitate Publikoa / Universidad Pública de Navarra	**E.T.S.I.I.T.** **INGENIERO ELÉCTRICO Y ELECTRÓNICO**	DEPARTAMENTO: **DEPARTAMENTO DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA**
PROYECTO: **VISOR INFRARROJO DE 360º PARA ROBOT BASADO EN ARDUINO**		REALIZADO: **GAGO BALDA, JAVIER** FIRMA:
PLANO: **ESQUEMA ELÉCTRICO DEL RECEPTOR INFRARROJO**		FECHA: **27/06/2014** Nº PLANO: **2º**



Nafarroako Unibertsitate Publikoa / Universidad Pública de Navarra	**E.T.S.I.I.T.** **INGENIERO ELÉCTRICO Y ELECTRÓNICO**	DEPARTAMENTO: **DEPARTAMENTO DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA**
PROYECTO: **VISOR INFRARROJO DE 360º PARA ROBOT BASADO EN ARDUINO**		REALIZADO: **GAGO BALDA, JAVIER** FIRMA:
PLANO: **ESQUEMA ELÉCTRICO DEL ENCODER**		FECHA: **27/06/2014** Nº PLANO: **3º**