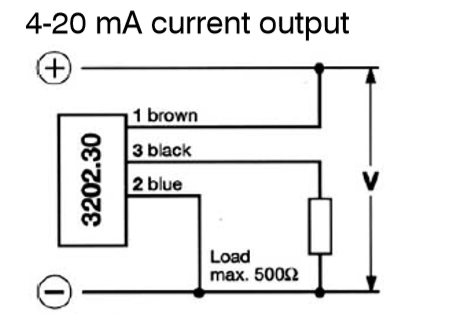
Uno de los mayores inconvenientes a la hora de realizar la prueba piloto fue la manera en la que se obtenían los datos de los experimentos. En un primer momento se intento obtener los datos grabando directamente del display del banco de ensayos. La lentitud a la hora de transcribir esos datos obligo a desechar ese método. Es entonces cuando se comienza a trabajar en crear un adquisidor de datos automático que permita la rápida y cómoda obtención de los mismos. Se opta por usar Arduino como módulo adquisidor de datos, debido a su bajo precio, la comodidad que supone trabajar con openhardware y la gran comunidad que engloba a arduino, con todas las ventajas que esto supone. En un principio se opta por monitorizar 2 variables, la velocidad del viento y la velocidad de giro del aerogenerador. Para ello se emplearan sensores disponibles en el laboratorio de fluidos de la universidad.

ANEMÓMETRO CALORIMÉTRICO

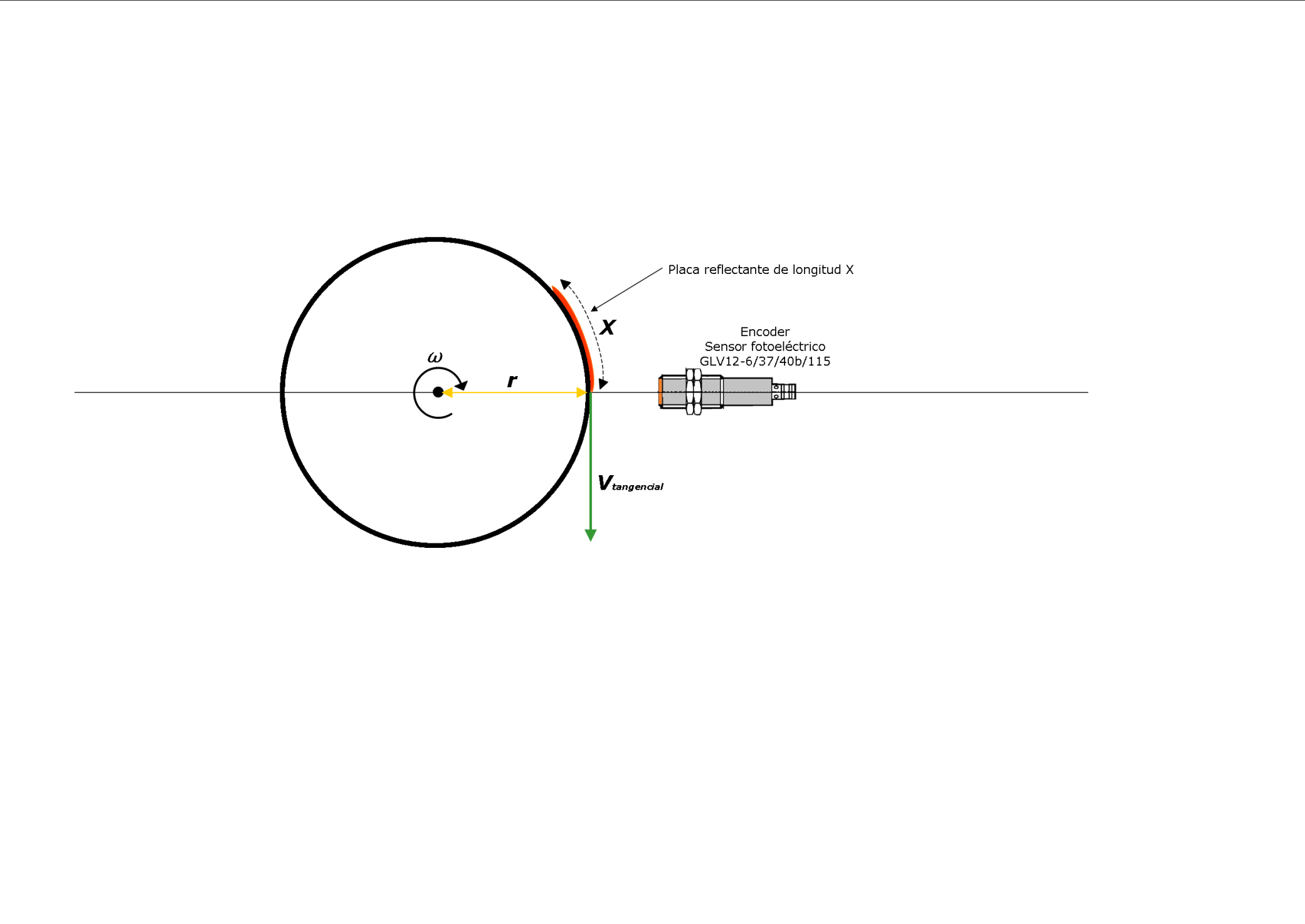
El anemómetro utilizado es el vent-captor 3202.30 que es un anemómetro de hilo caliente o calorimétrico. En función de la velocidad del viento aporta una señal en forma corriente entre 4 y 20 mA. Este tipo de sensores es muy común en la industria, para medir velocidades de flujo dentro de conductos. La ventaja de este tipo de anemómetros es la alta frecuencia de adquisición de datos ya que la medición la realiza por un principio de enfriamiento y sin partes móviles. Además puede medir velocidades del viento entre 0 y 50 m/s, teniendo además la capacidad de ajustarse a cualquier rango de medidas, aumentado la resolución para rangos de medida pequeños. Esto se debe a que el anemómetro permite ajustar tanto el mínimo como el máximo. Este sensor cuenta con 3 cables de diferentes colores: marrón (tensión positiva), azul (tensión negativa) y negro (señal).



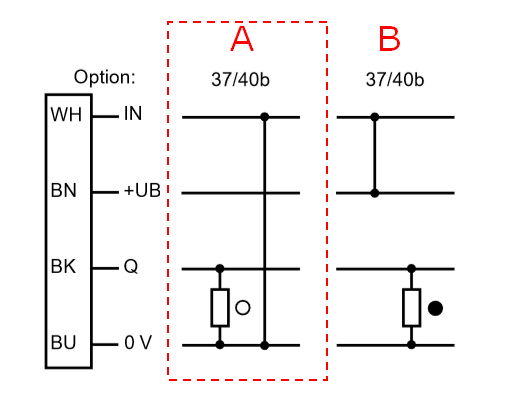
ENCODER

Para contar la velocidad de giro del aerogenerador se realizará un encoder capaz de contar la velocidad instantánea del aerogenerador. Para llevar a cabo este cometido se utilizará un sensor fotoeléctrico (PNP) disponible en el laboratorio. Este sensor fotoeléctrico se comporta como un sensor discreto, es decir, la información que proporciona es: encendido o apagado. Estará “encendido” cuando el láser que emite sea devuelto por una superficie reflectante y estará “apagado” cuando el láser que emite no sea reflejado. Para saber a qué velocidad está girando una rueda lo que se hace es colocar una superficie reflectante en el perímetro de la rueda giratoria, con una longitud conocida y calcular cuánto tiempo ha estado reflectando. De esta manera se puede saber que velocidad tiene la rueda, sabiendo la velocidad lineal de la superficie reflectante en el perímetro.

Este sensor fotoeléctrico únicamente nos está dando la información de cuánto tiempo tarda el segmento reflectante en pasar por delante del sensor. Con la información del tiempo y el tamaño de la placa podemos saber la velocidad tangencial en el perímetro del aerogenerador. Y sabiendo el radio del aerogenerador podemos calcular la velocidad angular del mismo aplicando las siguientes fórmulas.



Las conexiones del sensor disponible en el laboratorio se realizan de la siguiente manera.



En principio a diferencia del anemómetro calorimétrico este sensor cuenta con 4 cables: marrón (tensión positiva), azul (tensión negativa), negro (señal) y blanco (elegir funcionamiento). El cable blanco es el único que es diferente con respecto al anemómetro. Este cable sirve para elegir el tipo de funcionamiento.

Funcionamiento tipo A: Cuando se refleja el láser se induce tensión entre el cable negro y azul, es decir, se enciende. Mientras no se refleje se encuentra apagado.

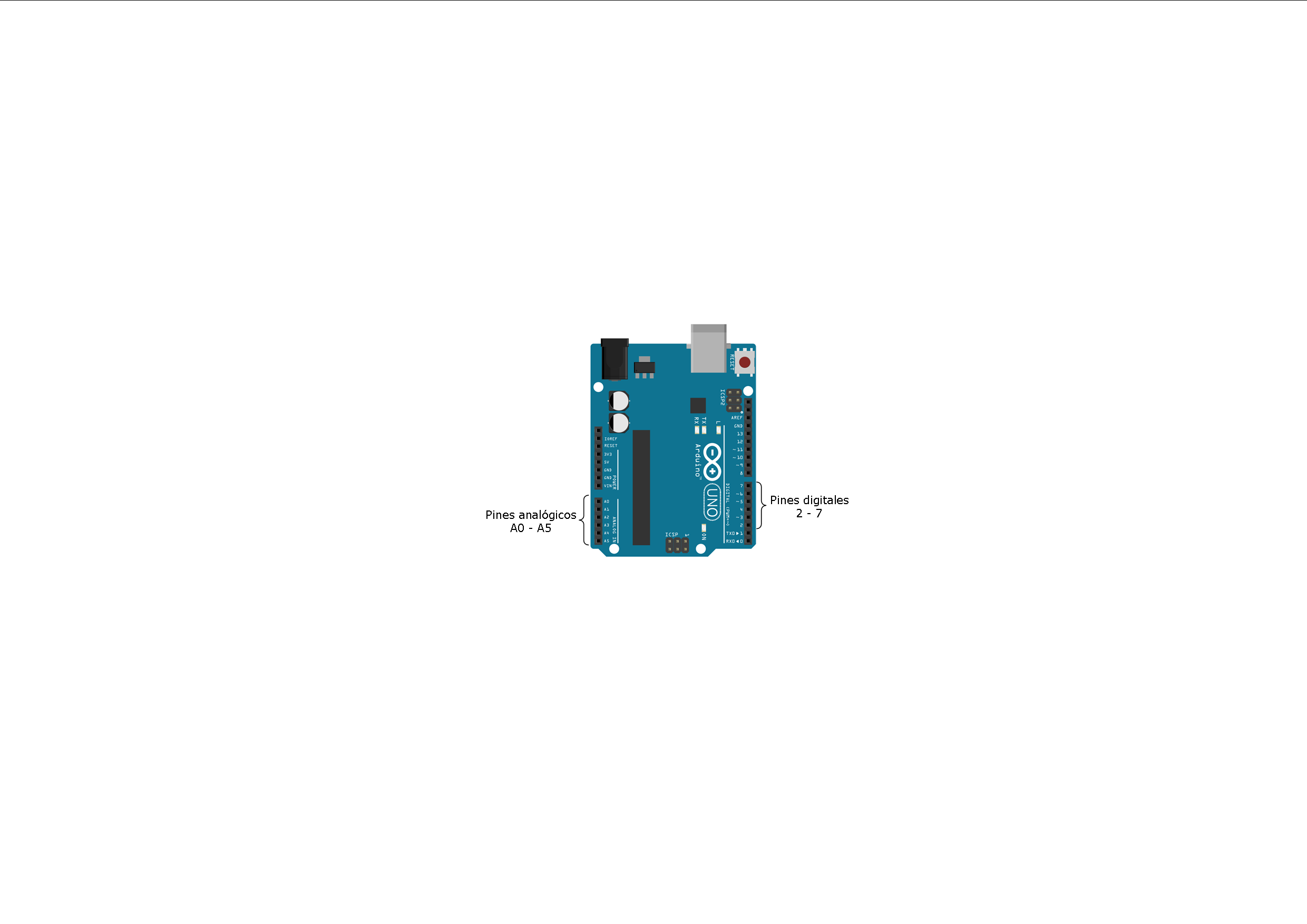
Funcionamiento tipo B: Cuando no se refleja el láser se induce tensión entre el cable negro y azul. Mientras el láser no se refleje esta encendido, en cuanto el láser se refleja se apaga.

Nosotros emplearemos la configuración tipo A, ya que se asegurarán menos tiempo de funcionamiento del sensor entre experimentos y en tiempos muertos, produciendo un ahorro de energía (aunque sea mínimo) y disminuyendo el calentamiento de los componentes electrónicos.

Otro aspecto clave es que cuando hablamos de “encendido”, lo que hace este sensor es dar un salto de tensión igual a la tensión de entrada. Dado que por sencillez se alimentarán ambos sensores a 24 V, el salto de tensión que proporcionará es de 24 V.

IMPLEMENTACIÓN DE SENSORES EN ARDUINO.

Como ya hemos visto en los apartados anteriores, ambos sensores funcionan de manera diferente y por ello a la hora de implementarlos en la placa de arduino cada uno tendrá un método de implementación. La diferencia fundamental entre ambos sensores es la manera de aportar información. El anemómetro la aporta de manera analógica, es decir, aporta en valor de corriente dentro de un rango. En cambio el sensor fotoeléctrico aporta información de manera discreta, es decir, aporta información en forma de 0-1, encendido-apagado. A la hora de implementar sensores en arduino hay que tener en cuenta esta diferencia porque hay que conectarlos a pines diferentes. Arduino cuenta con pines analógicos y digitales. Por lo tanto el anemómetro habrá que conectarlo a un pin analógico y el encoder a un pin digital.



IMPLEMENTACIÓN ANEMÓMETRO CALORIMÉTRICO

Este anemómetro requiere una fuente de tensión de 24 ± 30% V y la señal la aporta en forma de corriente entre 4 y 20 mA. Como el arduino únicamente es capaz de leer señales analógicas de tensión de entre 0 y 5 V, esta señal de corriente la convertiremos a señal de tensión utilizando una resistencia. La resistencia máxima que acepta el sensor es de 500 Ω, nosotros utilizaremos una resistencia de 220 Ω. Es más correcto trabajar con una resistencia de 250 Ω para aumentar el rango de medición del sensor, pero para evitar que la placa trabaje con tensiones cercanas a la máxima se utilizará una resistencia menor. La placa arduino asigna un valor entre 0 y 1023 a las tensiones comprendidas entre 0 y 5 voltios.

*5*

*1023*

*X*

*3,52*

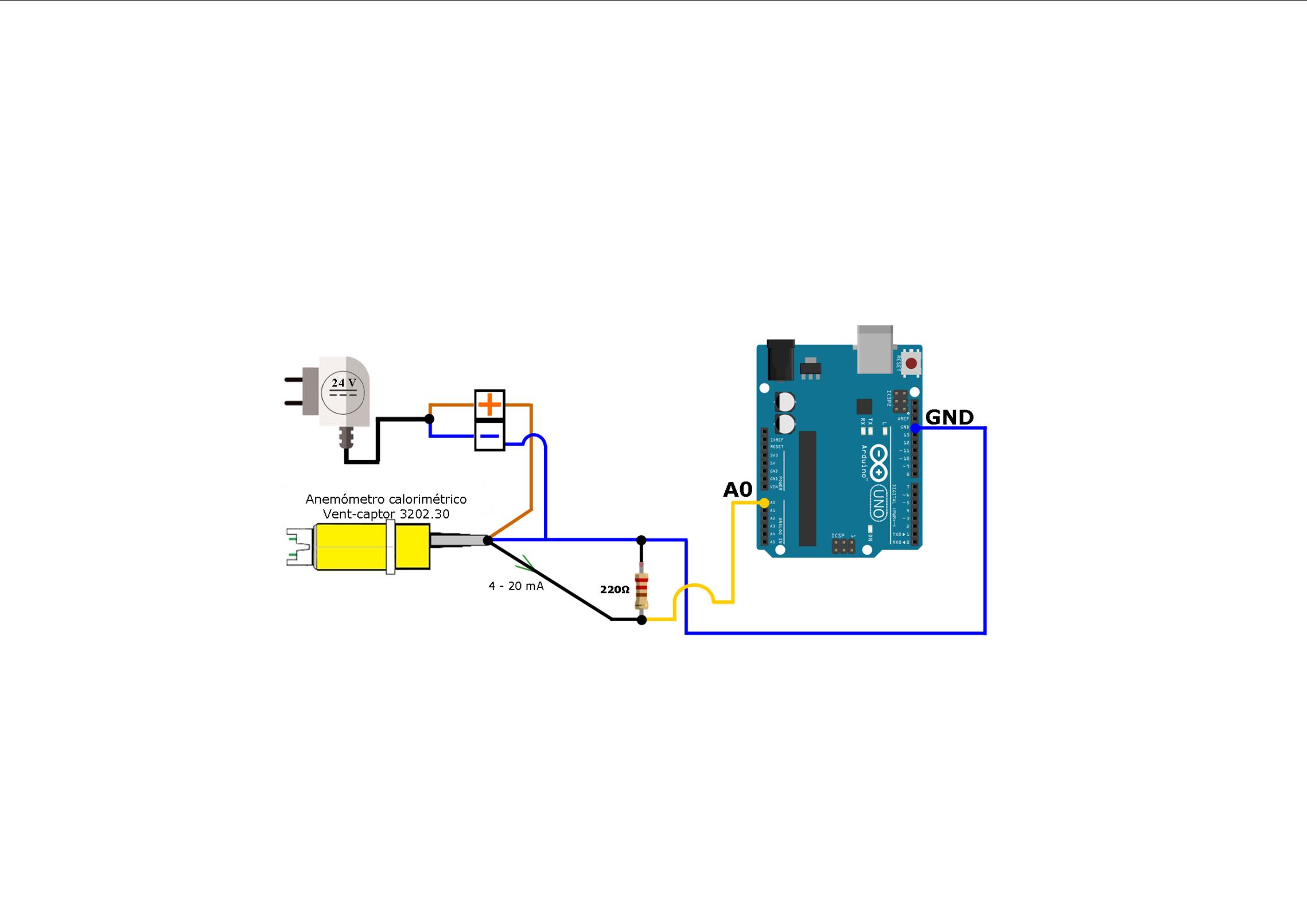
*Rango (V)*

*Nº de mediciones*

La resolución de la medida irá condicionada por la cantidad de mediciones y del rango en el que ajustemos el anemómetro. Teniendo en cuenta que la velocidad máxima del viento que genera el túnel de viento del laboratorio es de entorno a 12 m/s, ajustaremos el anemómetro para trabajar entre 5 y 15 m/s, es decir, trabajará en el rango de velocidades del viento de 10 m/s. Sabiendo esto se puede saber la resolución que tendrá el anemómetro. Nótese que cuento más pequeño sea el rango que se quiere medir mayor será la resolución del anemómetro.

Como ya se ha comentado antes esta resolución se puede mejorar disminuyendo el rango de medida o incluso sustituyendo la resistencia de 220 Ω por una de 250 Ω. Pero consideramos que esta resolución es suficiente para llevar a cabo los experimentos y además tiene la ventaja de que el arduino nunca va a registrar tensiones cercanas al máximo, eliminando el sobrecalentamiento de la placa y los componentes.

El esquema de conexión es el siguiente:

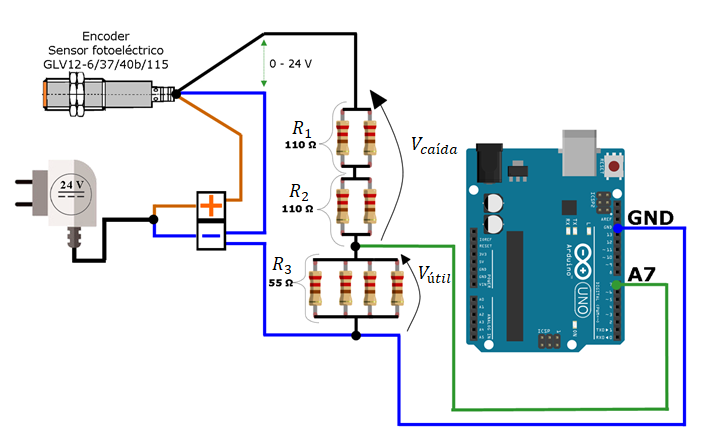


Realizando la conexión vista en el esquema anterior se consigue generar una tensión en el pin A0 de entre 0 y 4,40 V. Con esta información ya podremos realizar posteriormente la calibración necesaria para obtener datos de velocidad del viento.

IMPLEMENTACIÓN DEL ENCODER.

Ya hemos visto como funciona el sensor fotoeléctrico, ahora vamos a implantar este sensor de manera que mejoremos la resolución y frecuencia de adquisición de datos. Para aumentar la frecuencia de mediciones de la velocidad de giro se pondrán 12 segmentos de material reflectante de 3 centímetros. Por lo tanto sabremos a qué velocidad está girando aerogenerador 12 veces por vuelta. Como ya habíamos visto anteriormente, cuando el sensor se encuentre “encendido” proporcionará un salta de tensión de 24 V. Arduino admite trabajar con tensiones de hasta 5 V, por lo tanto debemos transformar este salto de tensión de 24 V a una tensión menor. Por ello añadiremos una serie de resistencia para dividir la tensión. Todas las resistencias con las que se cuentan son de 220 Ω. A la hora de añadir resistencias a la salida del sensor hay que tener en cuenta que la corriente máxima permitida a la salida es de 300 mA. Teniendo en cuenta que el sensor se alimenta a 24 V podemos saber cuál es la resistencia mínima.

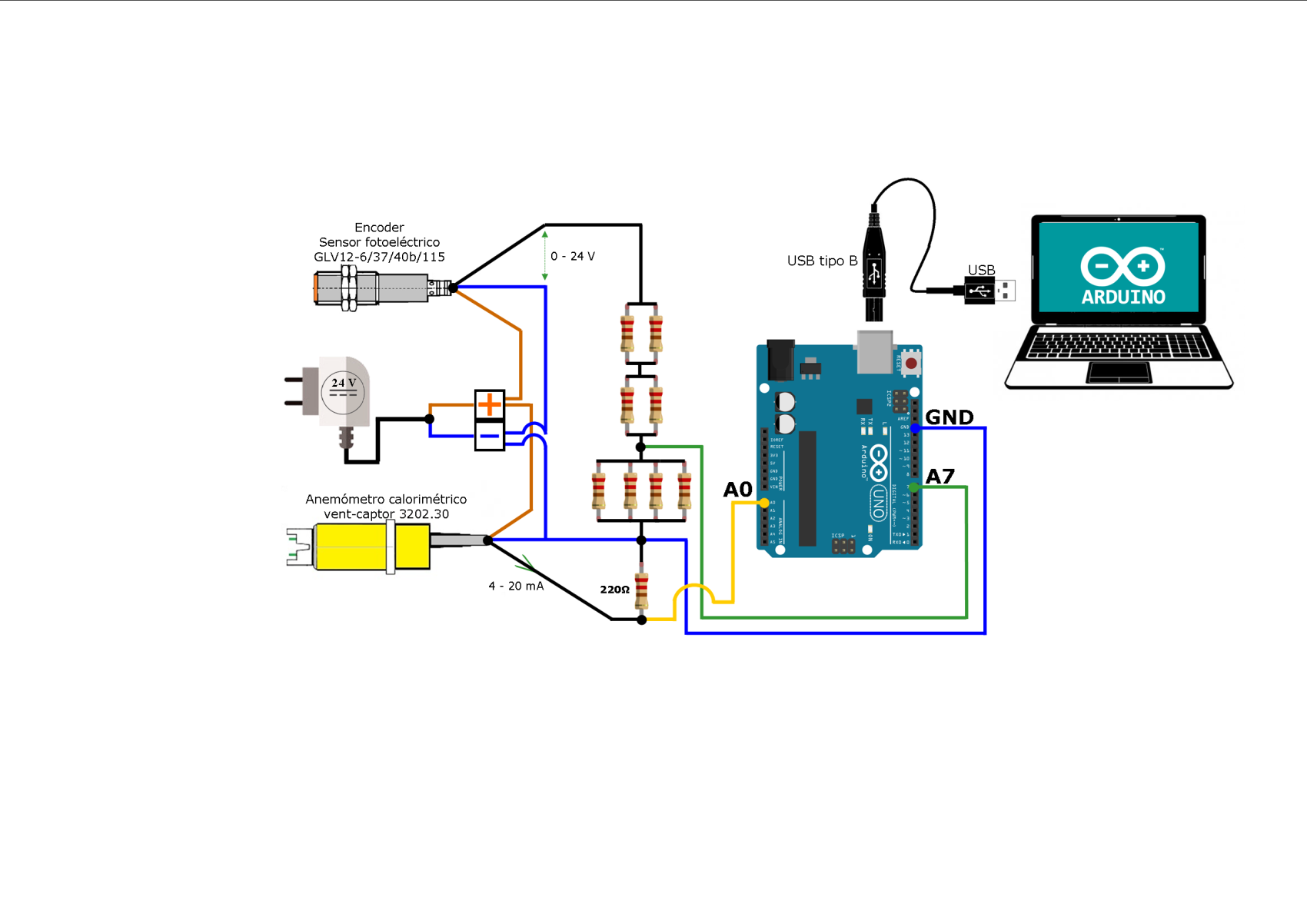
Tras probar diferentes configuraciones en las que se observa que las resistencias no son capaces de soportar potencias altas se llega al diseño final en el que se utilizan varias resistencias en paralelo para disminuir la potencia que cada una soporta por individual.



Por lo tanto el arduino tendrá una entrada de 4,8 V en el pin digital 7. Este será el valor con el que trabajaremos para leer la velocidad de giro del aerogenerador. Es interesante destacar que cada vez que el sensor realiza una lectura está disipando 2,1 W. Esta potencia podría disminuirse bajando la tensión a la que se alimenta el sensor, pero eso implicaría contar con dos fuentes de alimentación distintas para cada sensor, por sencillez únicamente se empleará una fuente de alimentación de 24 Voltios.

MONTAJE FINAL

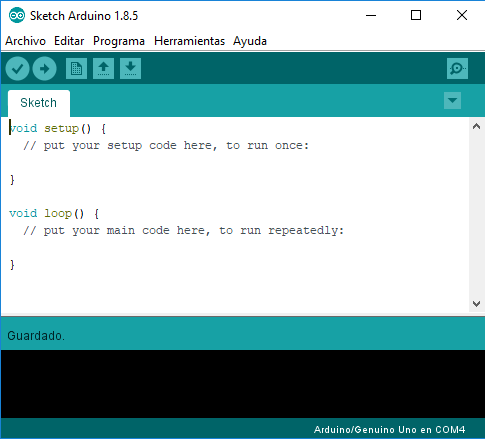
Una vez que ya hemos realizado el esquema individual de cada sensor se realiza el montaje final incluyendo ambos sensores. Es entonces cuando pasamos a trabajar con el software de Arduino para monitorizar ambas variables. Para ello conectamos en serie la placa arduino con el ordenador usando el puerto USB tipo B de la placa arduino y un puerto USB del ordenador.



COMUNICACIÓN ARDUINO-PC

Arduino es un lenguaje de programación sencillo y a la vez muy completo, respaldado por una gran comunidad de seguidores. Dado que se trata de un hardware/software libre dicha comunidad trabaja continuamente en su mejora. La realización del programa que se encarga de monitorizar estos sensores ha sido fácil gracias a la gran cantidad de contenido disponible en la web.

Mediante el cable USB se comunican en serie la placa de Arduino y el ordenador, empleando el software de Arduino. Al ejecutar se abrirá la siguiente ventana.



En ella se puede apreciar como directamente se abre un “sketch” básico de arduino. En él se pueden apreciar dos partes principales, aunque en la práctica se pueden definir en 3 partes:

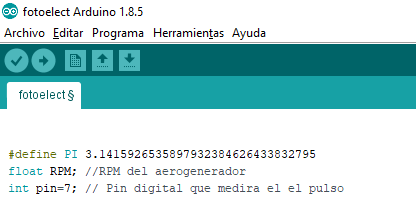
* Designación de variables: En la primera parte del sketch, antes del void setup se definen todas las variables a utilizar durante la ejecución.
* Void setup: es la parte del programa que se ejecuta únicamente al inicio. Aquí se recojen todos los comandos necesarios para inicializar el programa.
* Void loop: es la parte del programa que se ejecutará en bucle en todo momento después de inicializar el programa.

DEFINICIÓN DE VARIABLES

Las variables se pueden definir de muchas maneras diferentes, por ello, solo se nombrarán las maneras de definir variables que nos serán útiles para el programa. Las variables las designaremos principalmente de 3 maneras:

* int: para designar variables como un número entero
* float: para designar variables con decimales, hasta 7 decimales de precisión.
* #define: sirve para definir constantes.

Es importante saber que los cálculos matemáticos con números enteros son más rápidos que con números designados por la variable “float”, por ello, habrá que designar como “int” o entero, todas aquellas variables posibles. Además, hay que tener en cuenta que las variables se pueden definir acompañadas de un valor o sin él, porque es posible que su valor dependa de otras variables o que valla cambiando durante la ejecución del programa.



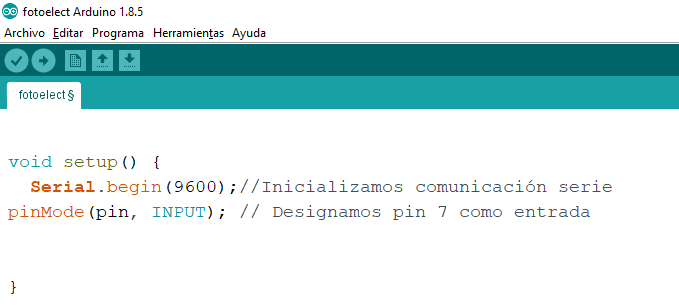
Se ve como se definen las constantes con “#define”, las variables con decimales con “float” y los valores enteros con “int”. Nótese que la variable RPM se define sin ningún valor porque es una variable que cambia de valor constantemente en función de la velocidad de aerogenerador.

VOID SETUP

En esta parte del programa se ejecutan los comandos de inicialización. Las acciones más habituales son:

Inicializar la comunicación serie arduino-PC: se inicializa la comunicación indicando la velocidad de transmisión en baudios.  Un baudio es una unidad de medida utilizada en telecomunicaciones, que representa el número de símbolos por segundo en un medio de transmisión digital. Es importante resaltar que no se debe confundir la [velocidad en baudios](https://es.wikipedia.org/wiki/Velocidad_en_baudios) (baud rate) con la [tasa de bits](https://es.wikipedia.org/wiki/Velocidad_en_bits_por_segundo) (bit rate), ya que cada evento de señalización (símbolo) transmitido puede transportar uno o más [bits](https://es.wikipedia.org/wiki/Bit). El valor típico utilizado es de 9600 baudios en la comunicación con arduino, aunque velocidades mayores también son soportadas.

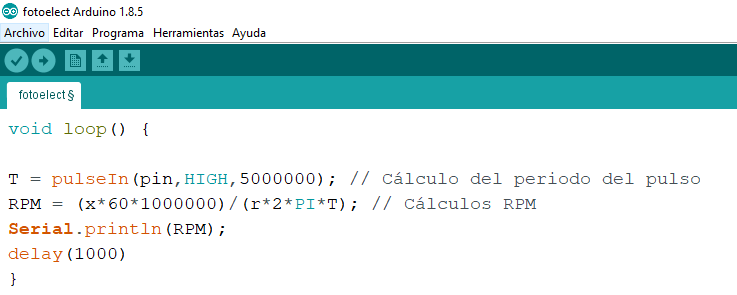
Designar pines: Dado que en arduino todos los pines pueden actuar como entrada o salida es importante designar en el void setup que cometido realizará cada pin.



Ejemplo de inicialización de comunicación serie y designación de pin.

VOID LOOP.

En esta parte del programa se definen los comandos que queremos que se ejecuten en bucle. De las tres partes de las que se compone un programa de arduino esta es la que más trabajo de programación requiere. Este apartado será el cerebro del programa.



En el ejemplo anterior se pueden observar diferentes funciones utilizadas recurrentemente en arduino:

* pulseIn: para designar la duración de un pulso eléctrico.
* Serial.println (valor): mostrar por comunicación serie algún valor.
* Delay(milisegundos): función para forzar una pausa dentro del programa.