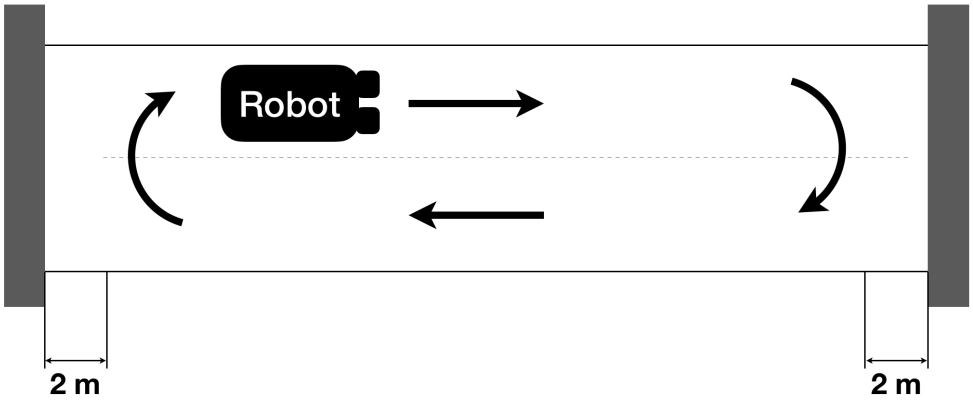
PROPUESTA DE EJERCICIO DE PROGRAMACIÓN

Dispone de un robot programable que debe realizar labores de vigilancia con las siguientes condiciones:

- Se moverá en línea recta entre 2 muros (ver figura), a 60 rpm y manteniendo una distancia de 2 m.
- Funcionará únicamente por la noche.
- Durante la vigilancia, se encenderá una cámara para grabar imágenes.
- Durante la vigilancia se hará parpadear una señal luminosa con un periodo de 1 s.



El robot está equipado con sensores propios (ultrasonidos en el puerto 1 y luminosidad en el puerto 2), unas luces de a bordo y una videocámara.

Se pide:

- Diagramas de flujo de un procedimiento que haga parpadear las luces una vez y del programa principal.
- El código del programa principal del robot (utilice el juego de instrucciones y la función de parpadeo).
- Durante las pruebas iniciales, se observa que en ocasiones el robot colisiona con los muros. Al parecer, durante el parpadeo el microprocesador no puede procesar las lecturas de los sensores. Rediseñe el código del programa principal prescindiendo de la función parpadeo. En su lugar, utilice las siguientes funciones: `millis()`, que devuelve el número de milisegundos transcurridos desde que se inició el programa y `switchOnBoardLights()` que cambia el estado de las luces de a bordo.

*Al final del enunciado podrá consultar el juego de instrucciones del lenguaje de programación del robot.

ANEXO: juego de instrucciones del lenguaje de programación del robot. Todas las reglas de sintaxis (declaración de variables, estructuras de control y de decisión, etc.), son similares a las de lenguajes de programación como el utilizado en Arduino, C++, Java, etc.

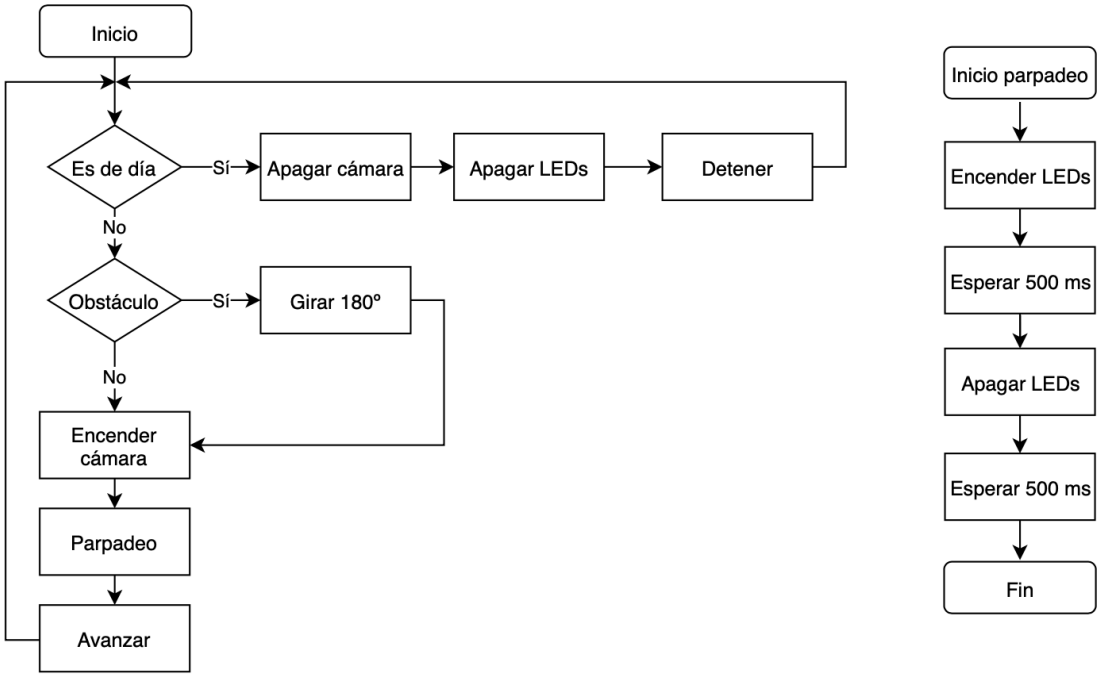
<table>
<tr><th>Instrucción/Función</th><th>Explicación</th></tr>
<tr><td><code>move(d,valor)</code></td><td>Instrucción que determina el movimiento del robot. El parámetro “d” indica el tipo de movimiento y “valor" la potencia (en movimientos lineales) o el ángulo (en caso de un giro, donde la potencia no es necesario especificarla, se calcula automáticamente):
<table>
<tr><td>d=0: movimiento hacia adelante.</td><td rowspan="2">valor: potencia, entre 0 (parado) y 255 (máx. velocidad que puede alcanzar).</td></tr>
<tr><td>d=1: movimiento hacia atrás.</td></tr>
<tr><td>d=2: giro.</td><td>valor: ángulo de giro, entre 0 y 180º</td></tr>
</table>
Al programar un giro, el robot describe ya la trayectoria (en arco) necesaria.</td></tr>
<tr><td><code>calcPower(n)</code></td><td>Función que traduce la velocidad deseada en rpm a un valor de potencia (0 - 255)</td></tr>
<tr><td><code>readUltrasonic(p)</code></td><td>Función específica para utilizar un sensor de ultrasonidos. Devuelve un número entero con la distancia en cm desde el sensor a un posible objeto. Como parámetro se debe indicar el puerto “p” al que está conectado el sensor.</td></tr>
<tr><td><code>readPort(p)</code></td><td>Función genérica utilizada para tomar lecturas en un puerto analógico “p”. Devuelve un valor entero.</td></tr>
<tr><td><code>calcIllumination(x)</code></td><td>Función que devuelve un valor de luminosidad a partir de una medida analógica.</td></tr>
<tr><td><code>setOnBoardLights(n)</code></td><td>Instrucción que controla el encendido (n=1) o apagado (n=0) de las luces LED que incorpora el robot.</td></tr>
<tr><td><code>enableCamera(n)</code></td><td>Instrucción para encender (n=1) o apagar (n=0) la cámara.</td></tr>
</table>

SOLUCIÓN AL EJERCICIO PROPUESTO DE PROGRAMACIÓN

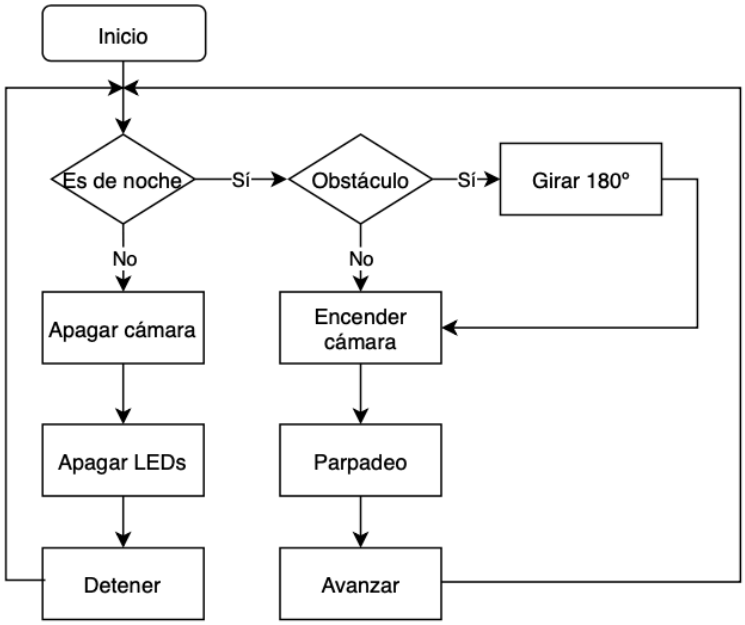
DIAGRAMAS DE FLUJO DE LA FUNCIÓN PARPADEO Y DEL PROGRAMA PRINCIPAL

SOLUCIÓN A: en el programa principal (fig. izquierda) se deben hacer 2 comprobaciones: si es de día, directamente se apagarán la cámara y las luces de a bordo y se detendrá el robot. Si es de noche se deberá poner en marcha el robot y se verificará si hay obstáculos a 2 metros. Si los hay, se programará un giro. Si no, simplemente se avanzará en línea recta, encendiendo la cámara y produciendo un parpadeo.

Este parpadeo será una función aparte. Es necesario incluir dos bloques de "espera" para que el ojo perciba que las luces se apagan y para que el parpadeo tenga un periodo de 1 s (0,5 s encendidas y 0,5 s apagadas).



SOLUCIÓN B: como alternativa para el algoritmo principal, se puede comprobar en primer lugar si es de noche en lugar de si es día. En ese caso la disposición espacial del diagrama cambia ligeramente:



El orden de los bloques de proceso cuando están en serie (p.ej. Apagar cámara+Apagar LEDs+Detener) es indiferente.

CODIFICACIÓN DEL PROGRAMA PRINCIPAL

El programa admite dos soluciones a la hora de evaluar el nivel de iluminación “L”: si se especifica primero qué hacer si es de día o si se especifica primero qué hacer si es de noche.

SOLUCIÓN A: se evalúa si es de día, si el nivel de iluminación es >10.

```
int v=calcPower(60);

while(true){

  if(calcIllumination(readPort(2))>10.0){
    move(0,0);
    setOnBoardLights(0);
    enableCamera(0);
  }

  else{
    if(readUltrasonic(1)<=200){
      move(2,180);
    }
    else{
      move(0,v);
    }
    enableCamera(1);
    parpadeo();
  }
}
```

Estas 3 líneas pueden ir en cualquier orden

Estos 3 bloques pueden ir en cualquier orden

SOLUCIÓN B: se evalúa si es de noche, si el nivel de iluminación es <10.

```
int v=calcPower(60);

while(true){

  if(calcIllumination(readPort(2))<10.0){
    if(readUltrasonic(1)<=200){
      move(2,180);
    }
    else{
      move(0,v);
    }
    enableCamera(1);
    parpadeo();
  }

  else{
    setOnBoardLights(0);
    enableCamera(0);
    move(0,0);
  }
}
```

Estos 3 bloques pueden ir en cualquier orden

Estas 3 líneas pueden ir en cualquier orden

CODIFICACIÓN DEL PROGRAMA PRINCIPAL CON MEJORA (SIN LA FUNCIÓN PARPADEO)

Se utiliza el reloj interno del procesador a través de la función `millis()` que se facilita en el enunciado. La estrategia consiste en cronometrar el tiempo transcurrido entre dos instantes midiendo la diferencia entre dos llamadas a esta función:

1) Tomar una muestra de tiempo (`t_inicial`). Esta variable almacenará un valor muy grande, por lo que el tipo de dato más adecuado es `unsigned long`.
2) Periódicamente tomar nuevas muestras de tiempo (`t_actual`).
3) Hacer la diferencia entre ambas muestras. Si es igual o superior a un valor determinado (500 ms, el tiempo a contar), se ejecutarán las instrucciones deseadas. Se actualizará el valor de `t_inicial` a `t_actual` para que sirva de referencia de tiempos para la siguiente iteración. De esta manera se puede continuar con la ejecución del programa sin detenerla.

SOLUCIÓN A: se evalúa si es de día, si el nivel de iluminación es >10.

```
int v=calcPower(60);
unsigned long t_inicial=millis();

while(true){
  if(calcula_iluminacion(readPort(2)>10.0){
    move(0,0);
    setOnBoardLights(0);
    enableCamera(0);
  }

  else{
    enableCamera(1);
    if(readUltrasonic(1)<=200){
      move(2,180);
    }
    else{
      move(0,v);
    }

    unsigned long t_actual=millis();
    if(t_actual-t_inicial>=500){
      switchOnBoardLights();
      t_inicial=t_actual;
    }

  }
}
```

SOLUCIÓN B: se evalúa si es de noche, si el nivel de iluminación es <10.

```
int v=calcPower(60);

unsigned long t_inicial=millis();

while(true){
  if(calcula_iluminacion(readPort(2))<10.0){
    if(readUltrasonic(1)<=200){
      move(2,180);
    }
    else{
      move(0,v);
    }
    enableCamera(1);

    unsigned long t_actual=millis();
    if(t_actual-t_inicial>=500){
      switchOnBoardLights();
      t_inicial=t_actual;
    }
  }

  else{
    setOnBoardLights(0);
    enableCamera(0);
    move(0,0);
  }
}
```

CRITERIOS DE EVALUACIÓN EJERCICIO PROPUESTO DE PROGRAMACIÓN

CONTEXTUALIZACIÓN

- **Temario oposiciones:** temas 9 (sistemas informáticos) y 70 (control programado).
- **Currículo Tecnología** (según convocatoria oposiciones, pto 7.2.1.1. Primera prueba fase de oposición/ parte A "*en todo caso, supondrán la aplicación práctica de conceptos relacionados con el temario de la especialidad y/o los currículos vigentes*"). Materias implicadas: tecnología (4º ESO), robótica (4º ESO), TIC-I (1º de bachillerato), TIC-II (2º de bachillerato).

CRITERIOS DE EVALUACIÓN

La calificación de cada uno de los ítems será a "todo o nada".

PLANTEAMIENTO (5 puntos)

Identifica el problema o supuesto práctico. - *Propone como solución al programa principal y a la función parpadeo sendos diagramas de flujo.*	1
Identifica el problema o supuesto práctico. - *Propone como solución a la codificación del programa principal un código que utiliza el juego de instrucciones del enunciado.*	1
Plantea adecuadamente la resolución del problema o supuesto práctico. - *Incluye en el diagrama de flujo bloques de decisión para 2 condiciones (iluminación y presencia de obstáculos) y 7 acciones (encender cámara, apagar cámara, parpadeo LEDs, apagar LEDs, avanzar, detener y girar). Los rectángulos colocados en serie se pueden colocar en cualquier orden.*	1
Plantea adecuadamente la resolución del problema o supuesto práctico. *- Codifica el programa principal en una estructura if/else e incluye las instrucciones de acuerdo al enunciado.*	2

DESARROLLO Y RESULTADO FINAL (90 puntos)

Explica de forma acertada y precisa cada una de las etapas en el desarrollo del problema o supuesto práctico. - *En particular, es deseable que se explique el planteamiento del algoritmo que resuelve el programa principal, así como el procedimiento sobre cómo utilizar el reloj interno del procesador para verificar el paso del tiempo (último apartado de mejora).*	5
Resuelve el problema o supuesto práctico de forma correcta, obteniendo un resultado coherente. - *El diagrama de flujo del programa principal incluye las dos verificaciones: el nivel de iluminación y la presencia de obstáculos. Se puede verificar si es de día o si es de noche.*	5
Resuelve el problema o supuesto práctico de forma correcta, obteniendo un resultado coherente. - *El diagrama de flujo del programa principal sitúa adecuadamente las acciones (rectángulos) a llevar a cabo en función de las condiciones evaluadas.*	5
Resuelve el problema o supuesto práctico de forma correcta, obteniendo un resultado coherente. - *El diagrama de flujo del programa principal contiene lazos de realimentación para comprobar el estado de los sensores de forma ininterrumpida.*	5
Resuelve el problema o supuesto práctico de forma correcta, obteniendo un resultado coherente. - *El diagrama de flujo de la función "parpadeo" presenta una estructura lineal formada por 4 acciones: encender LEDs, esperar 500 ms, apagar LEDs y esperar 500 ms. Es imprescindible que se incluyan los dos bloques de espera y que se especifique el tiempo (se admite en ms o en s).*	10

Resuelve el problema o supuesto práctico de forma correcta, obteniendo un resultado coherente. - *La estructura del programa consiste en una estructura de decisión de tipo if/else. La sintaxis obedece a la forma:* *if(condición) { instrucciones } else {instrucciones en caso contrario}* *Donde "condición" es la verificación de si es de día o de noche (soluciones A o B descritas anteriormente:* *Para la solución A es: calcIllumination(readport(2))>10.0)* *Para la solución B es: calcIllumination(readport(2))<10.0)* *Entre paréntesis, debe incluir el parámetro de entrada para la llamada a la función, que debe ser obligatoriamente la del juego de instrucciones que permite la lectura del puerto 2 donde está conectado el sensor, es decir readport(2).* *Se admite como válido resolverlo en 2 instrucciones, declarando primero una variable y después utilizándola dentro de la condición. Por ejemplo:* *float lecturaPuerto = readPort(2));* *if(calcIllumination(lecturaPuerto)<3.0) { instrucciones }*	15
Resuelve el problema o supuesto práctico de forma correcta, obteniendo un resultado coherente. - *Dentro de "else" se incluyen las instrucciones necesarias para cumplir las especificaciones del enunciado, de acuerdo a la solución (dependiendo de que la verificación de la iluminación sea <10.0 o >10.0), que serán las mostradas en la solución (alternativas A y B).*	10
Resuelve el problema o supuesto práctico de forma correcta, obteniendo un resultado coherente. *Se incluye en el lugar correcto la instrucción para mover el robot hacia adelante a 60 rpm. Para ello se admiten varias soluciones:* - *Declarar una variable de tipo entero con nombre genérico como resultado de la llamada a la función calcPower del juego de instrucciones, con la sintaxis: int nombre = calcPower(60). En este caso la llamada a la función move será move(0,nombre);* - *Utilizar dentro de la instrucción move del juego de instrucciones la llamada a la función como parámetro, con la sintaxis: move(0,calcPower(60))*	10
Resuelve el problema o supuesto práctico de forma correcta, obteniendo un resultado coherente. *Anidado dentro de la estructura if/else general se encuentra otra sentencia "if" con la segunda verificación (presencia de obstáculos). Dicha verificación responderá obligatoriamente a la sentencia:* *if(readUltrasonic(1)<=200)* *En caso de que la verificación inicial sea nivel de iluminación<10.0 esta verificación aparecerá dentro del "else". En caso de que sea >10.0 aparecerá dentro del "if" (ver código de la solución).* *Se admite como válido declarar una variable previa a la sentencia if y utilizarla después dentro de la condición, por ejemplo:* *int lecturaPuerto1 = readUltrasonic(1);* *if(lecturaPuerto1 <=200){* *A continuación se incluirá la instrucción, que de acuerdo al juego de instrucciones será un giro de 180º, como:* *move(2,180);* *Las llaves { y } antes y después de esta instrucción (si sólo ha escrito una única instrucción) son opcionales.*	15

Resuelve el problema o supuesto práctico de forma correcta, obteniendo un resultado coherente.
- *Declara al inicio del programa una variable t_inicial con la sintaxis:*
 	
`unsigned long t_inicial=millis();`
 	
- *Sustituye la instrucción con la función de parpadeo por una porción de código como la siguiente, colocado en el lugar apropiado según la solución sea A o B.*
 	
`unsigned long t_actual=millis();`
`if(t_actual-t_inicial>=500)`
`  switchOnBoardLights();`
`  t_inicial=t_actual;`
`}`
 	
Donde:
- *t_actual es un nombre genérico para una variable que guarda el resultado de la función millis(), incluyendo el valor en tiempo real del reloj interno.*
- *t_inicial es un nombre genérico para una variable que guarda el resultado de la última referencia de tiempos que se tomó y contra la que se mide la diferencia con el valor actual para saber el tiempo transcurrido.*
 	
Estas 2 variables deben ser declaradas como tipo "unsigned long". Se admite el tipo "int" o "long" (acertar el tipo unsigned long se valorará en el apartado de "expresión").
 	
Se considera válido declarar una variable para almacenar el tiempo transcurrido, como la diferencia entre el tiempo inicial y el tiempo actual, utilizándola así:
 	
`unsigned long t_actual=millis();`
`unsigned long t_transcurrido=t_actual-t_inicial;`
`if(t_transcurrido>=500)`
`  switchOnBoardLights();`
`  t_inicial=t_actual;`
`}`
	10

EXPRESIÓN (5 puntos)

Utiliza adecuadamente los conceptos y la terminología técnica y se expresa con corrección gramatical y ortográfica. - *Emplea simbología normalizada en la representación de los diagramas de flujo (rectángulos, rombos, flechas unidireccionales) y sigue las reglas de su diseño (disposición de arriba a abajo y de izquierda a derecha, rectángulos con sólo una entrada, etc.).*	2
Utiliza adecuadamente los conceptos y la terminología técnica y se expresa con corrección gramatical y ortográfica. - *Utiliza márgenes más amplios (sangrado) para los bloques de instrucciones significativos, como las que se definen dentro de una estructura if/else, emplea el punto y coma ; al final de cada instrucción, el asterisco * como signo de multiplicación y el punto . en los números decimales, tal y como se muestra en la solución.*	2
Utiliza adecuadamente los conceptos y la terminología técnica y se expresa con corrección gramatical y ortográfica. - *Declara las variables utilizando tipos de datos adecuados. Por ejemplo: Unsigned long para las medidas de tiempos (variables tipo t_actual, t_transcurrido y t_inicial) cuando se utiliza la función millis()*	1