- Algoritmos para la implantación de funciones y programas.

-Desarrollo 1:

*-Función de inicialización **prog_gpio**:*

Salvamos en registros el numero del terminal GPIO y la funcionalidad (entrada/salida)
Calculamos la dirección base lógica GPIO a raíz de la dirección física conocida
Guardamos dicha dirección en variable global
Obtenemos offset asociado al registro que contiene al terminal (GPSELn)
Sumamos el offset a la dirección de memoria

Accedemos swi supervisor
    Accedemos al contenido del registro
    Implantamos la funcionalidad seleccionada
    Escribimos el contenido en el registro
Salimos swi supervisor

*-Función de uso **valor_gpio:***

Salvamos en registros el numero del terminal GPIO y el valor de salida
Guardamos dirección base lógica
Seleccionamos configuración del terminal (SET/CLEAR)
Sumamos offset a la dirección de memoria

Accedemos swi supervisor
    Escribimos máscara en el registro
Salimos swi supervisor

*-Función de uso **leeValor_gpio:***

Salvamos numero de terminal GPIO que se desea leer
guardamos dirección base lógica
Averiguamos offset correspondiente al terminal (GPLEV0/1)

Accedemos swi supervisor
    Aplicamos máscara sobre contenido del registro
    guardamos resultado
Salimos swi supervisor

devolvemos resultado

*-Función main **EC21:***

Configuramos leds como salida
Configuramos botones como entrada
Encendemos led verde extremo

BUCLE hasta 20 iteraciones hacer:

leer botón izquierdo
    si pulsado

        si led encendido está en el extremo
        volver BUCLE
        apagar led actual
        encender led izquierdo

volver BUCLE
leer botón derecho
si pulsado

si led encendido esta en el extremo
volver BUCLE
apagar led actual
encender led derecho
volver BUCLE

si ninguno esta pulsado volver al bucle.

-Desarrollo 2:

*-Función de Configuración **progEv_gpio:***

Salvamos terminal GPIO y tipo de evento que debe generar (Flanco rising/falling)
Accedemos a dirección base lógica
Estudiamos la configuración (GPREN/GPFEN)
Calculamos el registro encargado del terminal
Sumamos offset a la dirección de memoria

Accedemos swi supervisor
Accedemos al registro
Implantamos evento al terminal
Salimos swi supervisor

*-Función de uso **leeEv_gpio:***

Salvamos el terminal GPIO del que se desea leer evento
Accedemos a la dirección base lógica
Averiguamos registro que se encarga del terminal (GPEDS0/1)
Sumamos offset a dirección de memoria

Accedemos swi supervisor
Leemos si estado de evento
Guardamos valor
Salimos swi supervisor

devolvemos estado de evento

*-Función de uso **clearEv_gpio:***

Salvamos el terminal GPIO del que se desea eliminar el evento
Accedemos a la dirección base lógica
Averiguamos registro que se encarga del terminal
Sumamos offset a dirección de memoria

Accedemos swi supervisor
Eliminamos aviso de evento sobre el terminal
Salimos swi supervisor

*-Función main **EC22:***

Configuramos leds como salida
Configuramos botones como entrada
Configuramos botones para que generen eventos de flanco rising
eliminamos posibles eventos anteriores

Encendemos led verde extremo

BUCLE hasta 20 iteraciones hacer:

leer botón izquierdo
si pulsado

si led encendido está en el extremo
apagar led actual y encender led opuesto
volver BUCLE

apagar led actual
encender led izquierdo
volver BUCLE

leer botón derecho
si pulsado

si led encendido esta en el extremo
apagar led actual y encender led opuesto
volver BUCLE

apagar led actual
encender led derecho
volver BUCLE

si ninguno se ha detectado ningún evento volver a BUCLE.

-Desarrollo 3:

*-Función de uso **gen_tono:***

Salvamos en registros periodo de nota y frecuencia
Calculamos la longitud delay

WHILE hasta el valor de frecuencia:
encender zumbador
delay

apagar zumbador
delay

volver a WHILE

*-Función de retardo **delay**:*

REPITE: decrementar longitud delay hasta que se anula
si no es nulo
volver a WHILE

***-**Función main **EC23**:*

Configurar zumbador como entrada
Punteros a los arrays de frecuencias y periodos

WHILE hasta numero de notas hacer:

leer contenido posición Frecuencia
leer contenido posición Periodos
producir tono
volver WHILE

termina programa

- Descripción desarrollo de la practica.

En esta práctica hemos programado una serie de Drivers con el objetivo de ejercer control sobre los pines GPIO de la Raspberry Pi.

Inicialmente, tuvimos que desarrollar una función de inicialización la cual proporciona la dirección base lógica GPIO y configura la funcionalidad del terminal deseado (entrada o salida).
Más tarde, dos funciones de uso que determinaban el valor de salida del terminal (leds) y la lectura del valor que devuelve el mismo (botones).

Una vez completado el primer Driver, elaboramos un main que permitía conectar la 'Berry Clip' y usar sus leds para rotar la luz de derecha a izquierda y viceversa según el botón que se apretaba.

Más adelante en el siguiente desarrollo de la práctica, nuestra misión era mover la luz por los leds de igual forma pero uno a uno. Para ello, fue preciso ampliar el Driver anterior con nuevas funciones que permitan a los botones generar eventos de flanco ascendentes o descendentes y leer en los registros de control para saber si se pulsaban o no.

En el tercer y último desarrollo, el Driver a programar no precisaba de función de inicialización porque coincidía con la elaborada en el primero. De tal forma que solo construimos una función de uso la cual tomaba dos parámetros y conseguía reproducir un sonido gracias a la función delay. Por último, implementamos un main para comprobar el funcionamiento de la función emitiendo una serie de notas seguidas consiguiendo una melodía.

- Descripción de las Pruebas realizadas.

Cabe destacar que previamente a la elaboración de los programas main, antes explicados, para comprobar el correcto funcionamiento de las funciones implementadas, realizamos programas de prueba como el encendido de un solo led, para saber si los algoritmos funcionaban bien y entonces emplearlos en el main, y la reproducción de solo una nota por parte del zumbador.

Resulta imprescindible la creación de estos programas de prueba si queremos asegurarnos de que el código de nuestras funciones es correcto y podemos usarlos en los main con seguridad.

- Respuestas razonadas.

-Describe los pasos necesarios para programar como salida un pin genérico del dispositivo GPIO. Usa como ejemplo la programación del pin GPIO25.

Para programar como salida un pin genérico del dispositivo GPIO necesitamos acceder al contenido del registro GPSELn que almacena los 3 bits asociados a dicho pin y configurar su funcionalidad a 001.

Explicaremos un ejemplo con el pin GPIO25. Inicialmente, guardamos la dirección base lógica GPIO. Seguidamente, determinamos qué registro se encarga del pin. Para ello, hemos de dividir 25 entre 10, obteniendo que 2 será el nivel del GPSEL y el resto 5 se multiplicará por el numero de bits que se le asocie a cada pin, es decir 15.
Por consiguiente, sabemos que debemos de sumar a la dirección de memoria el offset correspondiente (GPSEL2) y que tanto la máscara (7), para anular los 3 bits, como el bit para configurar salida han de desplazarse 15 veces.
Finalmente, accedemos al modo supervisor con tal de manipular los registros y modificamos los 3 bits deseados para implantar la funcionalidad deseada sobre el pin requerido
Una vez terminado el proceso, escribimos el nuevo contenido en el registro previo y salimos de modo supervisor.

-¿Se puede producir un evento en un terminal inicializado como salida?

No se puede producir un evento en un terminal GPIO inicializado como salida porque no se puede recibir valores a raíz del mismo, solo se determina su valor de salida que puede ser 0 y 1(3,3V).

-Indica cómo se ha conseguido pasar de la representación de una nota mediante una letra a los parámetros necesarios para que suene esa nota durante un segundo.

Para asociar las letras a sus respectivos parámetros podemos implementar variables que contengan dichos valores y después acceder a ellos.
Por ejemplo:

```
A: 2273, 440
LDR R4, =A
LDR R0, [R4] @Periodo
LDR R1, [R4, #TAM_ELEM] @Frecuencia
BL gen_tono
```