TRABAJO DE SISTEMAS ELECTRÓNICOS INDUSTRIALES

CURSO 2021/22

Parte MICROCONTOLADORES:

PLATAFORMA AUTOESTABILIZABLE

Integrantes:

Gómez-Pamo González-Cela, Jorge	54637
González Alday, Javier Pío	54639
González Denia, Adrián	54647

Grupo de clase: A404

ÍNDICE

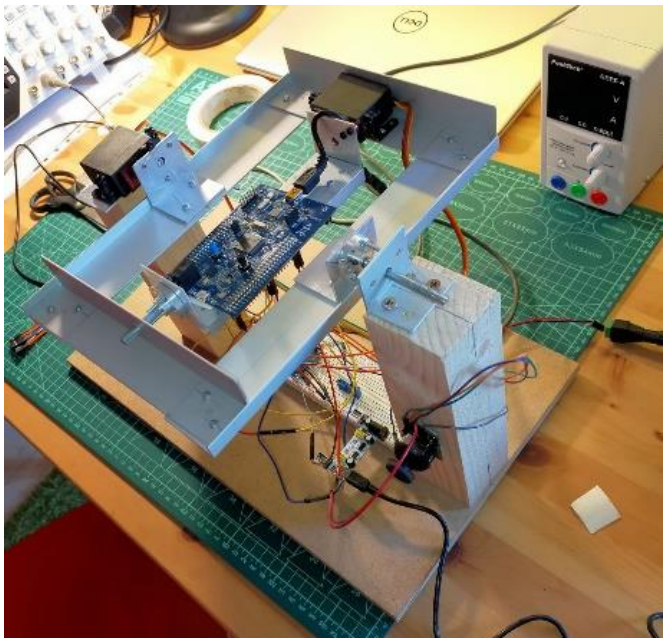
Introducción

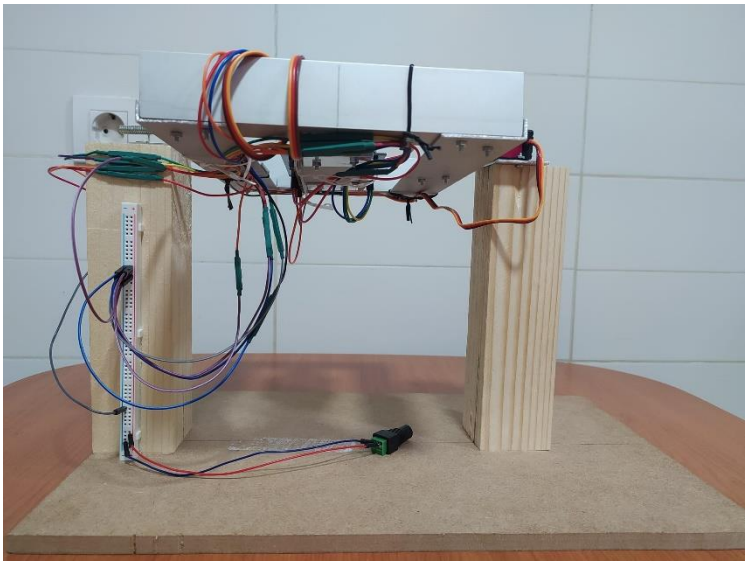
El proyecto consiste en un estabilizador de dos ejes. Se dispone de una maqueta que contendrá una base en la que se sostendrá la placa STM32F407, sujeta a dos ejes que incorporarán servomotores para compensar el giro e inclinación de la base.

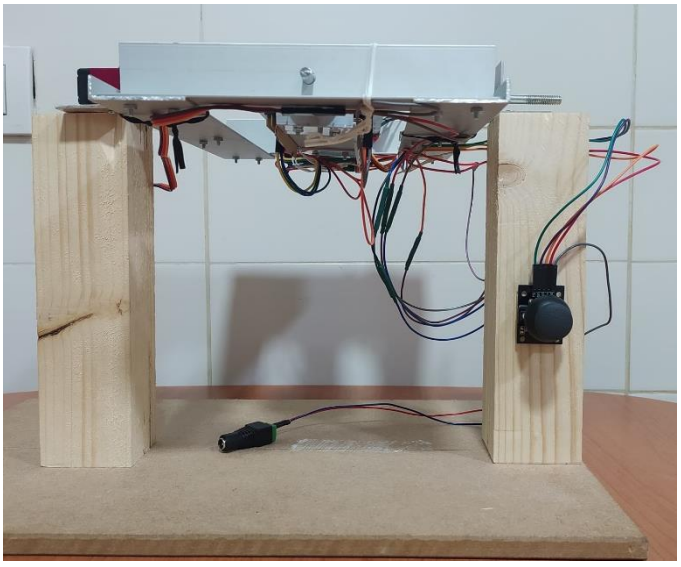
Para ello se usará el acelerómetro integrado en la placa STM para detectar la inclinación de la misma y mediante unos servomotores se estabilizará la plataforma que sostiene la placa. Por otra parte, se propondrá también el control de los servomotores con un potenciómetro, pudiendo intercambiar entre los modos de funcionamiento con la pulsación de un botón.

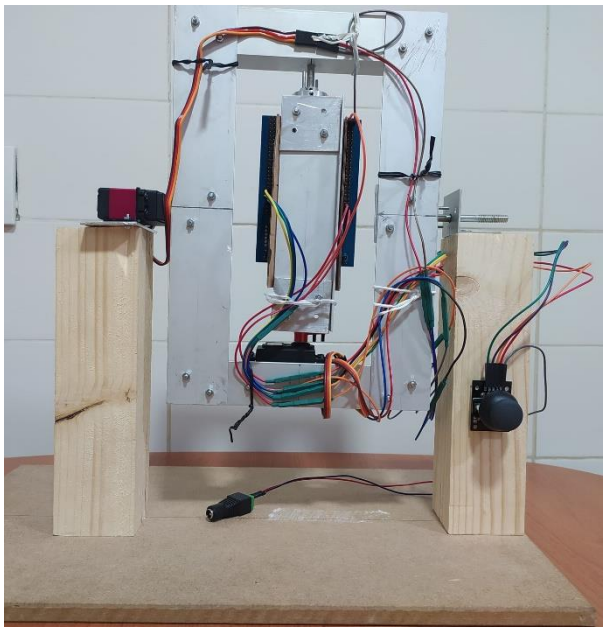
Fotos del sistema

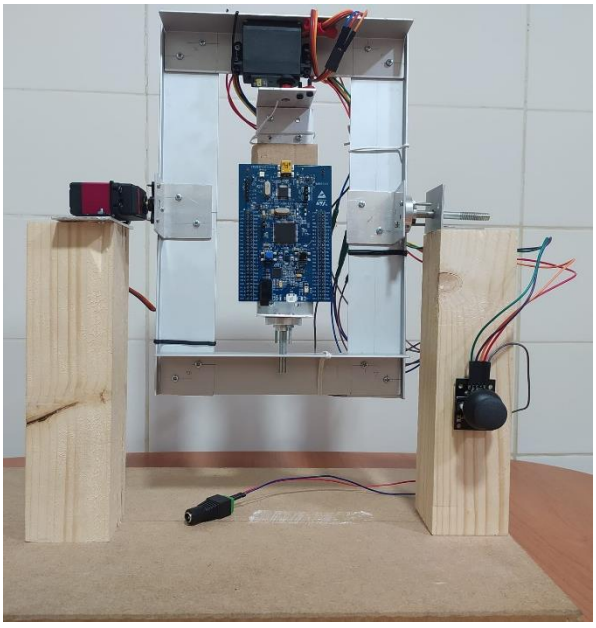
A continuación se muestran varias fotos e imágenes de los bocetos y maqueta del proyecto.

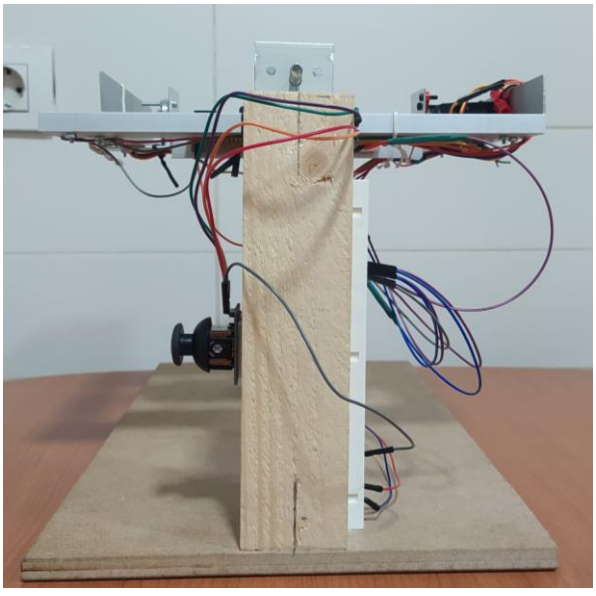












Código del sistema

Como se ha mencionado anteriormente, el control de servomotores se realizará mediante dos modos diferentes:

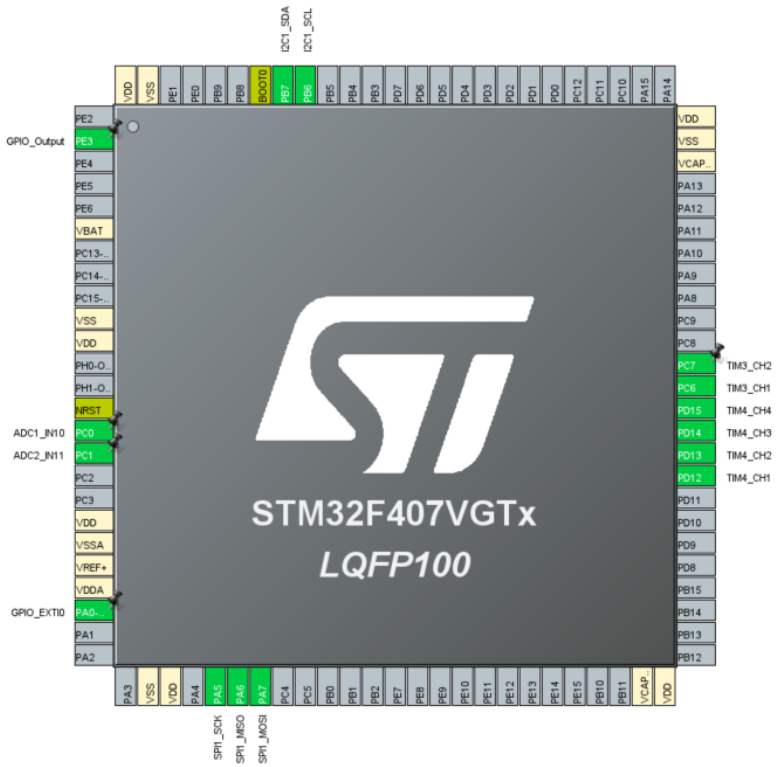
Como modo por defecto se utilizará el modo de estabilización. Para ello se configura un temporizador en modo PWM, habilitando un canal para cada servomotor. Se utiliza como referencia la inclinación del acelerómetro, siendo la posición de reposo del servomotor la inclinación nula. En función del giro del mismo, se configurará la señal PWM que modificará la posición del servomotor.

Como segundo modo de funcionamiento, se realiza una lectura de un potenciómetro para el movimiento de cada servomotor, utilizando un conversor ADC y la misma señal PWM que en función del giro del potenciómetro llevará al servomotor a una posición concreta.

Para el cambio de modo de funcionamiento se usará el botón de usuario de la placa. Puesto que se requiere rapidez en el cambio de lectura, se configurará el botón como una interrupción.

Configuración del STM32CubeIDE

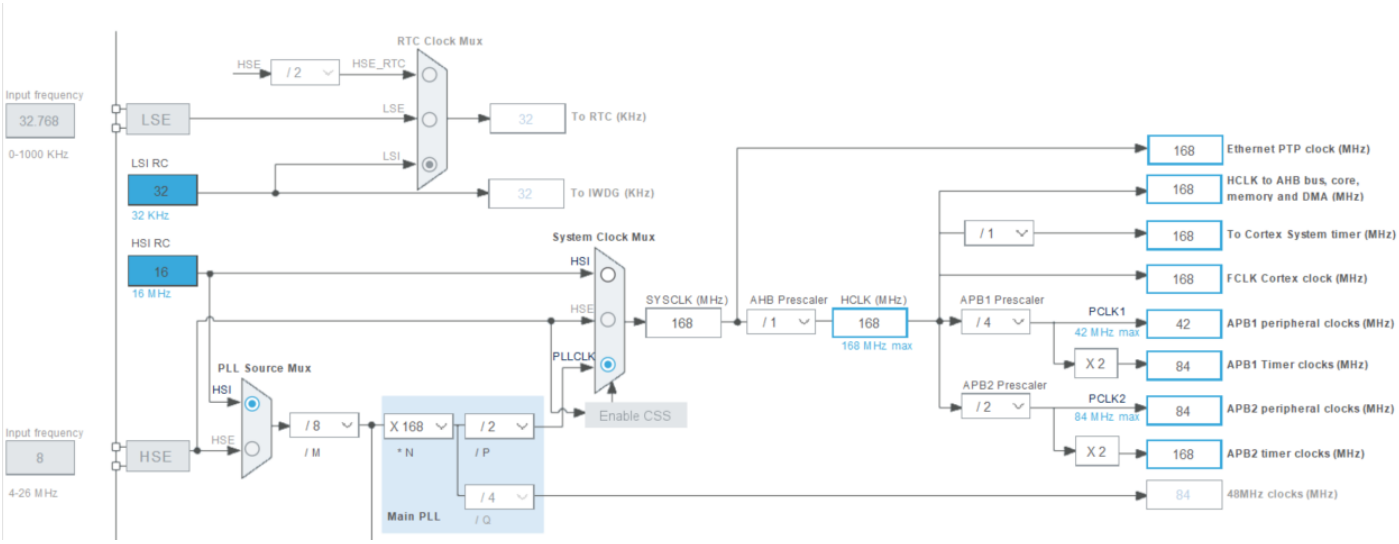
Para la configuración de todos los componentes involucrados en el sistema se ha configurado de la siguiente forma el proyecto de STM322CubeIDE:



Por partes, se han utilizado los siguientes pines:

- **Comunicación SPI** (PE3, PA5, PA6, PA7): En configuración Full-Duplex, con transmisión de 8 bits.
- **Comunicación I2C** (PB6, PB7): En Fast Mode y 400kHz de frecuencia del canal I2C
- **Temporizadores para servomotores y LEDS** (PC6, PC7, PD12, PD13, PD14, PD15): TIM3 con canales 1 y 2, y el TIM4 con canales 1,2, 3 y 4. Ambos temporizadores en modo PWM, Prescaler: 83, Period: 20000, Counter Mode UP.
- **Conversión A/D** (PC0, PC1): Se utilizan 2 conversores diferentes para liberar de trabajo 1 solo.
- **Botón de cambio de modo** (PA0): En modo interrupción.

La configuración de relojes internos queda de la siguiente forma:



Explicación por partes del código utilizado

- **Variables globales utilizadas**
  - **Comunicación SPI**
    - uint8_t spiTxBuf[2]
    - uint8_t spiRxBuf[2]
  - **Actualización del acelerómietro:** Obtención de los valores de aceleración en los tres ejes.
    - volatile int16_t accel_x
    - volatile int16_t accel_y
    - volatile int16_t accel_z
  - **Lectura del potenciómetro:** Obtención del valor entero según la posición del joystick, en el eje x, será necesario añadir un offset debido a la calibración.
    - int16_t adcval_x
    - int16_t adcval_y
  - **Modificación del ancho de pulso LEDS PWM:** Regulación del encendido de leds por PWM en función de la inclinación de la placa.
    - int16_t lights_x
    - int16_t lights_x
  - **Modificación del ancho de pulso SERVOMOTORES PWM:** Regulación de la posición de los servomotores por PWM en función de la inclinación de la placa.
    - int16_t angle_x
    - int16_t angle_x

- **Funciones declaradas del código**

Para la realizar las conversiones necesarias en cuanto a la recepción de lecturas tanto del acelerómetro, como del potenciómetro al ancho de pulso del PWM con un periodo predefinido, se han utilizado dos funciones de mapeo.

<u>Mapeo del acelerómetro</u>:

Esta función se encarga de realizar la conversión de los datos obtenidos por el acelerómetro al ancho del pulso del PWM en el modo de funcionamiento de estabilización. Retorna un valor concreto dentro del periodo definido que establece la posición del servomotor.

Se establece una histéresis en el origen, donde se considera que cualquier valor dentro de dicho rango es estable, por lo que no es necesario mover los motores.

Mapeo del ADC:

Análogo al mapeo del acelerómetro, se encarga de realizar la conversión de los datos obtenidos por el convertidor al ancho del pulso del PWM en el modo de funcionamiento de estabilización. Retorna un valor concreto dentro del periodo definido que establece la posición del servomotor.

- **Lectura del acelerómetro**

Para el uso del acelerómetro (LIS3DSH) se utilizará comunicación serie SPI, cuyos registros de lectura y escritura y los pines de conexión se pueden obtener de la hoja de características. La lectura de estos registros se ha realizado por polling, ya que se ha considerado que su lectura es más rápida que la dinámica del sistema. De la lectura del acelerómetro se obtiene la inclinación de la placa, que se usará como referencia para la estabilización de la plataforma.

Para la inicialización del acelerómetro hay que acceder al registro 0x20 del mismo (Control Regist 4) y escribir el valor 0x67 para activar los outputs de todos los ejes con una tasa de refresco de 100Hz. El código es el siguiente:

```
//Inicialización de los registros del acelerómetro
  HAL_GPIO_WritePin(GPIOE,GPIO_PIN_3,GPIO_PIN_RESET);
  spiTxBuf[0]=0x20;
  spiTxBuf[1]=0x67;
  HAL_SPI_Transmit(&hspi1,spiTxBuf,2,50);
  HAL_GPIO_WritePin(GPIOE,GPIO_PIN_3,GPIO_PIN_SET);
```

La lectura de los registros se realiza de la siguiente forma:

```
  HAL_GPIO_WritePin(GPIOE,GPIO_PIN_3,GPIO_PIN_RESET);
  spiTxBuf[0]=0x28|0x80; //OUT_X_L
  HAL_SPI_Transmit(&hspi1,spiTxBuf,1,50);
  HAL_SPI_Receive(&hspi1,&spiRxBuf[1],1,50);
  HAL_GPIO_WritePin(GPIOE,GPIO_PIN_3,GPIO_PIN_SET);

  HAL_GPIO_WritePin(GPIOE,GPIO_PIN_3,GPIO_PIN_RESET);
  spiTxBuf[0]=0x29|0x80; //OUT_X_H
  HAL_SPI_Transmit(&hspi1,spiTxBuf,1,50);
  HAL_SPI_Receive(&hspi1,&spiRxBuf[2],1,50);
  HAL_GPIO_WritePin(GPIOE,GPIO_PIN_3,GPIO_PIN_SET);

  HAL_GPIO_WritePin(GPIOE,GPIO_PIN_3,GPIO_PIN_RESET);
  spiTxBuf[0]=0x2a|0x80; //OUT_Y_L
  HAL_SPI_Transmit(&hspi1,spiTxBuf,1,50);
  HAL_SPI_Receive(&hspi1,&spiRxBuf[3],1,50);
  HAL_GPIO_WritePin(GPIOE,GPIO_PIN_3,GPIO_PIN_SET);

  HAL_GPIO_WritePin(GPIOE,GPIO_PIN_3,GPIO_PIN_RESET);
  spiTxBuf[0]=0x2b|0x80; //OUT_Y_H
  HAL_SPI_Transmit(&hspi1,spiTxBuf,1,50);
  HAL_SPI_Receive(&hspi1,&spiRxBuf[4],1,50);
  HAL_GPIO_WritePin(GPIOE,GPIO_PIN_3,GPIO_PIN_SET);

  HAL_GPIO_WritePin(GPIOE,GPIO_PIN_3,GPIO_PIN_RESET);
  spiTxBuf[0]=0x2c|0x80; //OUT_Z_L
  HAL_SPI_Transmit(&hspi1,spiTxBuf,1,50);
  HAL_SPI_Receive(&hspi1,&spiRxBuf[5],1,50);
  HAL_GPIO_WritePin(GPIOE,GPIO_PIN_3,GPIO_PIN_SET);

  HAL_GPIO_WritePin(GPIOE,GPIO_PIN_3,GPIO_PIN_RESET);
  spiTxBuf[0]=0x2d|0x80; //OUT_Z_H
  HAL_SPI_Transmit(&hspi1,spiTxBuf,1,50);
  HAL_SPI_Receive(&hspi1,&spiRxBuf[6],1,50);
  HAL_GPIO_WritePin(GPIOE,GPIO_PIN_3,GPIO_PIN_SET);

     // Carga de los datos del buffer a las variables
     accel_x = (spiRxBuf[2]<<8)|spiRxBuf[1];
     accel_y = (spiRxBuf[4]<<8)|spiRxBuf[3];
     accel_z = (spiRxBuf[6]<<8)|spiRxBuf[5];

     lights_x = map_acc(accel_x);
     lights_y = map_acc(accel_y) ;
```

Finalmente, los valores obtenidos son convertidos a valores de lights, que corresponden con el valor de PWM de los leds, mediante interpolación. Los valores límites de accel son 16384 que corresponde con 90º y -16384 que corresponde con -90º, los valores límites de lights son 20000 y -20000. Se ha añadido una protección por si el valor de accel excede sus límites y un margen de reposo de ±100 unidades donde se mantiene el balancín horizontal.

```
int16_t map_acc(int16_t accel){
    if (accel > 16384) return  T_lights;
    else if (accel < -16384) return -T_lights;
    else if(accel < 100 && accel > -100) return 0;
    return (accel - (-16384)) * (T_lights - (-T_lights)) / (16384 - (-16384)) - T_lights;
}
```

- **Control de los servomotores**

Con el uso de la función map_angle se convierte el valor de lights a su correspondiente valor en PWM mediante una interpolación, loas valores límites de light son -20000 y 20000 y los del PWM son 2500 y 500. Como los motores no giran en el mismo sentido, el valor de entrada de las funciones de map_angle son de signo contrario para el eje x e y.

```
int16_t map_angle(int16_t light){
    return (light - (-T_lights)) * (2500 - 500) / (T_lights - (-T_lights)) + 500;
}
```

El control de los servomotores se realizará mediante distintos canales de PWM. Se hace uso de una variable que va aumentando o disminuyendo para igualarse al valor del PWM. De esta forma los cambios de posición del balancín son progresivos y suaves. Cada dos milisegundos aumento o disminuye el PWM una unidad.

```
angle_x = map_angle(lights_x);
angle_y = map_angle(-lights_y);

if(HAL_GetTick() - start_x > 2){
    if(prev_x < angle_x)
        prev_x += 1;
    else
        prev_x -= 1;
    start_x = HAL_GetTick();}

if(HAL_GetTick() - start_y > 2){
    if(prev_y < angle_y)
        prev_y += 1;
    else
        prev_y -= 1;
    start_y = HAL_GetTick();}

__HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_1, prev_x);
__HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_2, prev_y);
```

- **Lectura de los potenciómetros**

Se emplea un conversor ADC y la misma señal PWM que en función del giro del potenciómetro llevará al servomotor a una posición concreta.

La lectura del conversor será por “polling”, “single channel”, “continuos conversion mode”, y se les asignará a las variables “adcval” el valor obtenido de la conversión.

A continuación, el valor de las variables “adcval” son convertidas en valores de lights, que se usarán como valor de “compare” de los PWM de los leds mediante interpolación. Los valores límites de adcval son 4095 y 0 y los de lights son 20000 y -20000.

Se han añadido protecciones por si el valor de adcval excede los limites y se ha añadido un margen de reposo de ±100 unidades donde se mantiene el balancín horizontal.

El valor final se divide entre dos para reducir el rango de movilidad de ±90 grados a ±45 y aumentar la precisión de la posición.

```
int16_t map_adc(int16_t adc){
    if (adc > 4095) return T_lights/2;
    else if (adc < 0) return -T_lights/2;
    else if(adc < 2148 && adc > 1947) return 0;
    return ((adc - 0) * (T_lights - (-T_lights)) / (4096 - 0) - T_lights)/2;
}
```

Posteriormente se mapearán estos valores para su conversión al ancho de pulso PWM.

```
HAL_ADC_Start(&hadc1);
if(HAL_ADC_PollForConversion(&hadc1, 100)==HAL_OK){
    adcval_x = HAL_ADC_GetValue(&hadc1) - 450;
    if(adcval_x < 0) adcval_x = 0;
}
HAL_ADC_Stop(&hadc1);

HAL_ADC_Start(&hadc2);
if(HAL_ADC_PollForConversion(&hadc2, 100)==HAL_OK){
    adcval_y = HAL_ADC_GetValue(&hadc2);
}
HAL_ADC_Stop(&hadc2);

lights_x = map_adc(adcval_x);
lights_y = map_adc(adcval_y);
```

- **Cambio de modo de funcionamiento**

Para el cambio de modo de funcionamiento se usará el botón de usuario de la placa. Puesto que se requiere rapidez en el cambio de lectura, se configurará el botón como una interrupción.

Para ello se emplea el siguiente callback:

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin){
    if (GPIO_Pin==GPIO_PIN_0){
            mode = !mode;
        }
}
```

- **Encendido de leds**

En el encendido de leds se ha utilizado del mismo modo un temporizador en modo PWM y las mismas variables de “lights”, de tal forma que permite un alumbrado incremental en función de la inclinación de la placa.

```
//Encendido de las leds por PWM
if(lights_x >= 0){
  __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_1, 0);
  __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_3, lights_x);
}
else{
  __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_1, -lights_x);
  __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_3, 0);
}

if(lights_y >= 0){
  __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_2, lights_y);
  __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_4, 0);
}
else{
  __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_2, 0);
  __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_4, -lights_y);
}
```

- **Control del display OLED**

Por último, se ha incluido al sistema una pantalla OLED que muestra en tiempo real la inclinación de la plataforma. Con ella se pretende mostrar los grados en los que se encuentra la placa estabilizada.

Para la obtención en grados de la inclinación de la placa se han utilizado las siguientes transformaciones de las variables obtenidas del acelerómetro:

```
grad_accel_x = atan(-accel_x/sqrt(pow(accel_y,2) + pow(accel_z,2)))*(180.0/3.1416);
grad_accel_y = atan(accel_y/sqrt(pow(accel_x,2) + pow(accel_z,2)))*(180.0/3.1416);
```

Para facilitar el control del display para el dibujado y escritura de texto en el mismo se ha hecho uso de 2 librerías externas:

- **SSD1306**: Librería que incluye funciones relacionadas para la creación de mensajes en la pantalla OLED de 124x68 de resolución. Su funcionamiento consiste en un conjunto de funciones que van creando un mensaje concreto para ser enviado de forma serial por un puesto I2C del dispositivo. Esta comunicación se realiza usando la función HAL_Trasnmit_Master.
- **FONTS**: En esta se incluyen los mapas de bits de 3 tipografías de distintos tamaños, será utilizada por la librería anterior con el fin de crear el mensaje serie concreto para generar mensajes en la pantalla.

Como lo que se necesita para crear el mensaje en la pantalla son strings, se pansan las variables de los grados obtenidas anteriormente de int a string, y se muestran por la pantalla. El código es el siguiente:

```
//Muestra por pantalla OLED
sprintf(buf_x, "%d", grad_accel_x);
sprintf(buf2_x, "%-10s", buf_x); //Relleno de espacios el resto del string
sprintf(buf_y, "%d", grad_accel_y);
sprintf(buf2_y, "%-10s", buf_y); //Relleno de espacios el resto del string

SSD1306_GotoXY (5, 10); // goto 10, 10
SSD1306_Puts ("X: ", &Font_7x10, 1);
SSD1306_GotoXY (30, 10);
SSD1306_Puts (buf2_x, &Font_7x10, 1);

SSD1306_GotoXY (5, 30);
SSD1306_Puts ("Y: ", &Font_7x10, 1);
SSD1306_GotoXY (30, 30);
SSD1306_Puts (buf2_y, &Font_7x10, 1);

SSD1306_UpdateScreen(); // update screen
```

Conclusiones

El proceso de diseño y programación del sistema ha seguido un desarrollo lineal. Se ha partido de una base sencilla que consistía en la estabilización de la base de la maqueta en un solo eje, utilizando comunicaciones serie SPI para el acelerómetro y el servomotor en el eje, dejando el segundo eje libre para posibles mejoras. Una vez obtenido el prototipo funcional, se procede a la ampliación del proyecto.

Primero, se añade el segundo modo de funcionamiento, pudiendo girar la base de forma manual con la lectura de un potenciómetro.

Para poder introducir el modo de funcionamiento, sin que choque con modo de estabilización, se ha establecido que se cambie de modo con la pulsación del botón de usuario de la placa. A esto se le añade también el encendido de los leds de la placa con un PWM, tal que cuando más inclinación tenga la base, más se iluminarán.

Para finalizar, con el último prototipo funcional con ambos modos de funcionamiento, se propone la adición del segundo eje con las mismas funcionalidades que el primero.

Enlace al repositorio del proyecto:

https://github.com/JPioGA/TRABAJOS_SED