_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE UPV INGENIEROS DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

FOCUS ASSISTANT: SISTEMA REMOTO DE CONTROL DE FOCO

UNIDAD ACTUADORA

Pedro Jaén del Hierro

Tutor: José Manuel Mossi García

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2015-16

Valencia, 13 de septiembre de 2016

Resumen

El objetivo principal de este trabajo consiste en la realización de la unidad actuadora de un follow focus o asistente de enfoque con el fin de controlar remotamente el foco en cámaras situadas sobre plataformas de estabilización, siendo así inaccesibles para el operario.

Esta primera parte del proyecto se basa en la parte electrónica y programable junto con sus componentes. Dejando así la parte de la unión con la cámara para el compañero cuyo proyecto es la segunda parte del mismo.

Para alcanzar el objetivo marcado, se ha procedido a estudiar en primera instancia los conceptos fundamentales de los usos y características de un follow focus. Además se hace referencia los tipos de follow focus existentes.

A continuación, se expondrán las partes que lo compondrán, y el código que programará el sistema para llevar a cabo la tarea correspondiente.

Por último, se examinara el resultado, analizando qué posibles mejoras podrían incorporarse para líneas futuras.

Resum

L'objectiu principal d'aquest treball consisteix en la realització de la unitat actora d'un "follow focus" o assistent d'enfocament amb la finalitat de controlar remotament el focus amb càmeres situades sobre plataformes d'estabilització, sent així inaccessibles per a l'operari.

Aquesta primera part del projecte està basada en la part electrònica i programable junt als seus components. Deixant així la part d'unió amb la càmera per al company que realitzarà la segona part del projecte.

Per aconseguir l'objectiu marcat, s'ha procedit a estudiar, en primera instància, els conceptes fonamentals dels usos i característiques d'un "follow focus". A més, es fa referència als tipus de "follow focus" existents.

A continuació, s'exposaran les parts que el composen i el codi que programarà el sistema per a dur a terme la tasca corresponent.

Per últim, s'examinarà el resultat analitzant quines són les possibles millores que podrien incorporar-se en línies futures.

Abstract

The main objective of this work is the realization of the actuating unit of a follow focus or focus assistant. In order to remotely control the focus of the cameras in the stabilization platforms, thus being inaccessible to the operator.

This first part of the project is based on the electronic and programmable part with their components. Leaving the part of the union with the camera for the partner whose project is the second part of the same.

To reach the objective set, we proceeded to study at first instance the fundamental concepts of the uses and characteristics of a follow focus. Also a reference is going to be done to the existing follow focus.

It will be continued with the parts that make up the system and the code schedule to carry out the corresponding task.

Finally the result is examined, analyzing possible improvements that could be incorporated for future lines.

AGRADECIMIENTOS

Quisiera agradecer a una serie de personas el apoyo y ayuda aportados en estos años de enseñanza y también durante la realización de este trabajo.

Primero a mis padres y hermana, Pedro, Carmen y Laura, por la educación y formación que he recibido sobre todo como persona, porque sin ellos dudosamente estaría en la situación en la que me encuentro.

Agradecer a mi amigo y compañero de clase Alberto, por todo el apoyo y ayuda recibida durante la carrera y más en especial estos últimos meses. Muchas gracias Albert.

También he querido acordarme de varios compañeros de clase, que tantas horas hemos compartido.

Por ultimo agradecer a mis amigos y a toda la gente que durante los malos momentos ha sido un apoyo. A todos vosotros, muchas gracias. LMSTR.

Índice

Capítulo 1. INTRODUCCIÓN

1.1 Motivación

Actualmente, podemos observar como la evolución de equipos de audiovisual, va en constante aumento y cada vez se alcanza un nivel superior, tanto en el mundo amateur como el profesional. Este hecho es debido a los grandes avances en los sensores de estas, que acercan un nivel profesional a una gama de productos asequibles para el mundo amateur.

Han aparecido productos que abren la puerta a nuevos límites en la fotografía y grabación de video, a través de mejoras en componentes de la propia cámara, o de sistemas que se acoplan a esta.

En el caso de nuevos sistemas que son capaces de integrarse en la cámara, han aparecido muchos productos de distintos fabricantes, los cuales abren el camino a nuevas funciones y aportan mucha versatilidad en este contexto audiovisual, como lo son los gimbal, drones, sliders, etc. que producen capturas de video mucho más novedosas y accesibles al usuario aficionado a la fotografía.

Además, es importen observar que nuevos productos han ido apareciendo según han ido observando problemas con estos nuevos productos. En el caso que se va a estudiar se ha decidido optar por la vía de los gimbal, observando uno de los principales problemas: el enfoque de estos. Esto es debido a que la anilla del foco de los objetivos que utilizan las cámaras DSLR son manuales, es necesario hacer rotar la anilla de enfoque con la mano, creando así una inestabilidad en el gimbal.

Es por eso que se ha propuesto crear para este proyecto un follow focus de bajo coste que sea adecuado para todos los objetivos, a través de un motor y un sistema de control, con el cual podamos enfocar diferentes puntos de enfoque, mientras se está utilizando un gimbal, de manera cómoda e intuitiva.

De esta manera y a lo largo de este proyecto se mostrara el montaje e implementación de un follow focus motorizado, así como los resultados que se han logrado con él.

1.2 Objetivo

Como objetivo de este proyecto, nos centraremos en la realización de un follow focus realizado con un microcontrolador y un motor. Concretamente, la parte correspondiente a la programación e implementación del sistema, ya que la parte mecánica del follow focus corresponderá a la segunda parte de este mismo trabajo realizada por Alberto Muñoz.

1.3 Metodología

Se explica a continuación el plan de desarrollo y las líneas marcadas para la correcta estructuración y desarrollo del trabajo de fin de grado.

Estudio de los diferentes sistemas de follow focus: como cualquier inicio de proyecto, se realiza una etapa de formación para el posterior desarrollo. Para este proyecto, se han estudiado los follow focus, las funciones y características que tienen.

Estudio de las partes de las que se va a componer: se tendrá que estudiar los diferentes componentes y piezas para lograr el resultado esperado, haciendo hincapié en el microcontrolador y motor, pues serán dos piezas fundamentales en en el proyecto.

Estudio de la programación del microcontrolador: para realizar las funciones que deseamos, vamos a tener que programar al microcontrolador.

Estudio de los resultados y líneas futuras: para finalizar, se realiza un análisis del resultado del trabajo y se evaluaran las posibles mejoras futuras que incorporar al sistema.

Los cuatro estudios anteriormente mencionados han sido utilizados para la elaboración de este trabajo de fin de grado.

1.4 Estructura

El trabajo de fin grado está estructurado en los siguientes puntos:

Capítulo1. Introducción: se presentan la motivación, objetivos y estructura del trabajo realizado.

Capítulo 2. Follow focus: se explica que es un follow focus, detallando sus funciones, para que se usa, y que situaciones ha provocado el uso de estos dispositivos. También se hace diferencia entre los dos grandes tipos de follow focus que hay, manual y con motor.

Capítulo 3. Componentes del Focus Assistance: en este capítulo se presentan los diferentes componentes que va a formar nuestro follow focus. Haciendo especial hincapié en el motor y la CPU, pues serán las dos partes más importantes de este.

Capítulo 4. Desarrollo: se va a explicar el desarrollo del código programado de Arduino, explicando las diferentes fases que se ha ido desarrollando con detalles acerca del código y de el montaje de este.

Capítulo 5. Resultados y líneas futuras: se presenta el resultado final de este trabajo de fin de grado, analizando las funciones que tiene nuestro asistente de foco viendo de qué manera podría seguir mejorándose.

Capítulo 2. FOLLOW FOCUS

Ante el problema de enfoque surgido con el uso de gimbals, trípodes u otros dispositivos con los que es imposible enfocar, debido a que en el caso de enfoque manual se desestabilizaría la imagen provocando un efecto indeseado. Se ha desarrollado un nuevo producto dirigido a usuarios que sufren este problema.

El follow focus o seguimiento de enfoque, es un mecanismo de control del enfoque de la cámara, esta herramienta no tiene por qué ser obligatorio en el uso de cámaras, ya que el operador de cámara es capaz de girar el anillo del enfoque de forma manual, pero existen situaciones en las que resulta incómodo, imposible o perjudicas a la imagen grabada causando inestabilidad ya sea porque se está usando como complemento a un gimbal o un trípode.

Figura 1. Gimbal del fabricante DJI

Este dispositivo es un mecanismo de control del enfoque de la cámara, que funciona a través de un conjunto de engranajes que están unidos al anillo del enfoque en el objetivo. Este engranaje se manipula con un pomo , el cual al girar provocará que gire también el anillo del enfoque del objetivo.

Existen dos clases diferenciables de follow focus, el primero de ellos se trata de un sistema de engranajes que giran gracias a la fuerza que tu ejerces con la mano, el segundo tipo varía en que en lugar de ejercer la fuerza para hacer girar el anillo con la mano, lleva integrado un motor controlado por un mando.

2.1 Follow Focus Manual

Este follow focus carece de motor, por lo tanto los engranajes de los que dispone deben de ser girados con la fuerza de nuestra mano, está compuesto por:

- El engranaje de la lente: que permite girar el anillo de enfoque, se trata de un anillo dentado que se adapta al tamaño del objetivo.
- Engranaje: probablemente esta es la parte más importante de todo el sistema ya que el transmite el giro desde el pomo hasta el anillo de enfoque.
- Pomo: es la pieza que el usuario gira para poder mover el foco del objetivo.
- Abrazadera: esta es la parte que se conecta a las dos barras que forman la base de la mayoría de las plataformas de apoyo de la cámara, según el fabricante tienen medidas diferentes, de diámetro y separación, a las cuales se tendrá que adaptar follow focus.

Figura 2. Follow Focus Manual

2.2 Follow Focus con motor

Al igual que la versión sin motorizar, este dispositivo es capaz de mover la anilla del enfoque del objetivo, solo que esta vez se ha integrado un motor en el interior que nos evita tener que

mover el pomo, en cambio existen nuevas partes que sustituirán a algunas piezas del Follow Focus manual.

Estas piezas son:

- Motor: el motor ira ensamblado junto al engranaje principal, siendo este el que siga haciendo de transmisor, cambiando únicamente la procedencia de la fuerza.

- Controlador del motor: esta parte será dirigida por un mando que controlara el motor del engranaje y a su vez el foco. Puede tener características que con el manual no tiene, como guardado de posición y transición entre dos enfoques.

Estos últimos follow focus, reúnen características extras además del movimiento del anillo de enfoque. Gracias a la electrónica que llevan pueden incorporar características que añadan facilidades a la hora de trabajar con el follow focus. Estas características pueden ser como un calibrado del sistema, en el cual el sistema detecta los límites del objetivo, evitando así forzar el objetivo pudiendo llegar a dañarlo.

Otra función añadida es la opción de guardar posiciones del enfoque, pudiendo realizar así transiciones de A a B a la velocidad que se elija.

Figura 3. Follow Focus a motor

Capítulo 3. Componentes del Focus Assistant

3.1 Microcontrolador

Cuando se habla de microcontrolador, se entiende por circuito integrado programable, capaz de ejecutar ordenes grabadas en su memoria. Este está compuesto por varios bloques funcionales, los cuales cumplen una tarea concreta. Las principales partes que lo componen son: unidad central de procesamiento, memoria y entradas y salidas.

Han sido dos familias de CPU que han encajado en nuestros requisitos las que hemos estudiado con el objetivo de escoger una de ellas para este proyecto.

3.1.1 Raspberry

La Raspberry es realmente un ordenador de placa reducida de bajo coste, que utiliza como software Linux ARM.

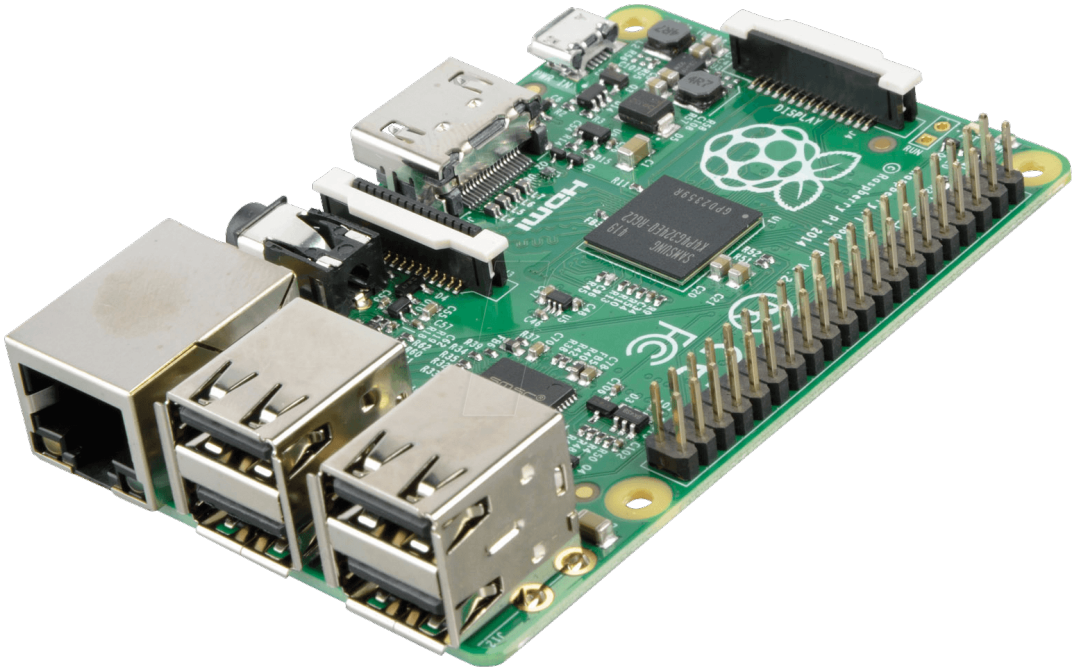


Figura 4. Raspberry Pi

El microprocesador que utiliza esta placa es ARM1176JZF-S a 700MHz, además incluye una GPU, y una memoria RAM de 512 MB. Las entradas y salidas de la Raspberry se denominan GPIO (General Purpose Input/Output) con diferentes funciones.

Existen 3 modelos de Raspberry: Raspberry 1, Raspberry 2 y Raspberry 3.

Al ser un ordenador, podemos programar en diferentes lenguajes, adaptándose a nuestras preferencias y al diseño que se requiere.[1]

3.1.2 Arduino

Arduino es una compañía de hardware libre, que diseña y manufactura placas de desarrollo de hardware y software. Esta es compuesta por circuitos impresos que integran un microcontrolador junto a un entorno de desarrollo.

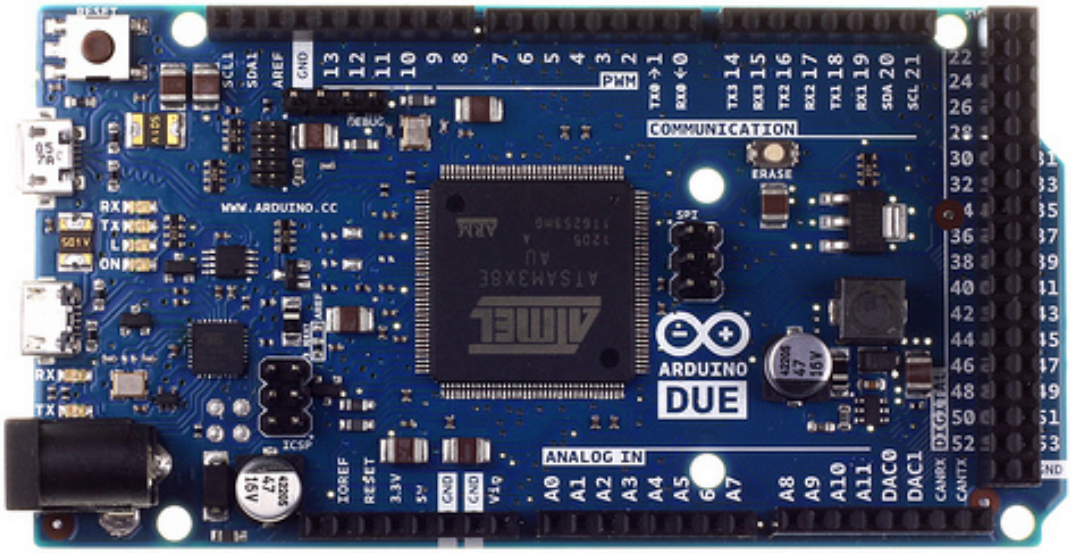


Figura 5. Arduino DUE

Este Hardware consiste en un microcontrolador, ATMEL AVR, junto a puertos digitales y analógicos de entrada y salida, los cuales son capaces de conectarse a placas de expansión o shields, que amplían las características básicas de la placa. Utiliza memoria SRAM y memoria Flash en la que guardar el programa.

El lenguaje de programación empleado, consiste en el entorno de Processing y lenguaje de programación basado en Wiring.

Una vez estudiado las dos posibles opciones de CPU para nuestro follow focus, ha sido seleccionado el modelo Arduino para nuestro proyecto por varias razones:

- Pese a ser menos potente que la placa Raspberry, ha pesado más su simplicidad frente a proyectos de hardware, ya que tiene la capacidad analógica y en tiempo real que permite la flexibilidad de trabajar con casi cualquier tipo de sensor o chip .

- El entorno de Arduino es mucho más fácil de usar que Linux, ya que este proyecto se va a basar en programación de hardware sin una interfaz gráfica que necesite de un sistema operativo.
- La simplicidad hace que Arduino sea más robusto a la hora de un apagado accidental del sistema.
- Facilidad a la hora de conectar las salidas y entradas, tanto analógicas como digitales

Una vez decidido vamos a escoger el Arduino UNO, debido a la experiencia que tenemos ya en este modelo en concreto.

3.1.2.1 Arduino UNO

Arduino UNO es una placa compuesta por el ATmega328P, microcontrolador con 32 KB de memoria Flash y 2KB de memoria RAM, funciona a 16 MHz y 5V de alimentación.

El número de entradas y salidas que tiene en total son 14 las cuales son suficientes para conectar todos los componentes que se va a necesitar.

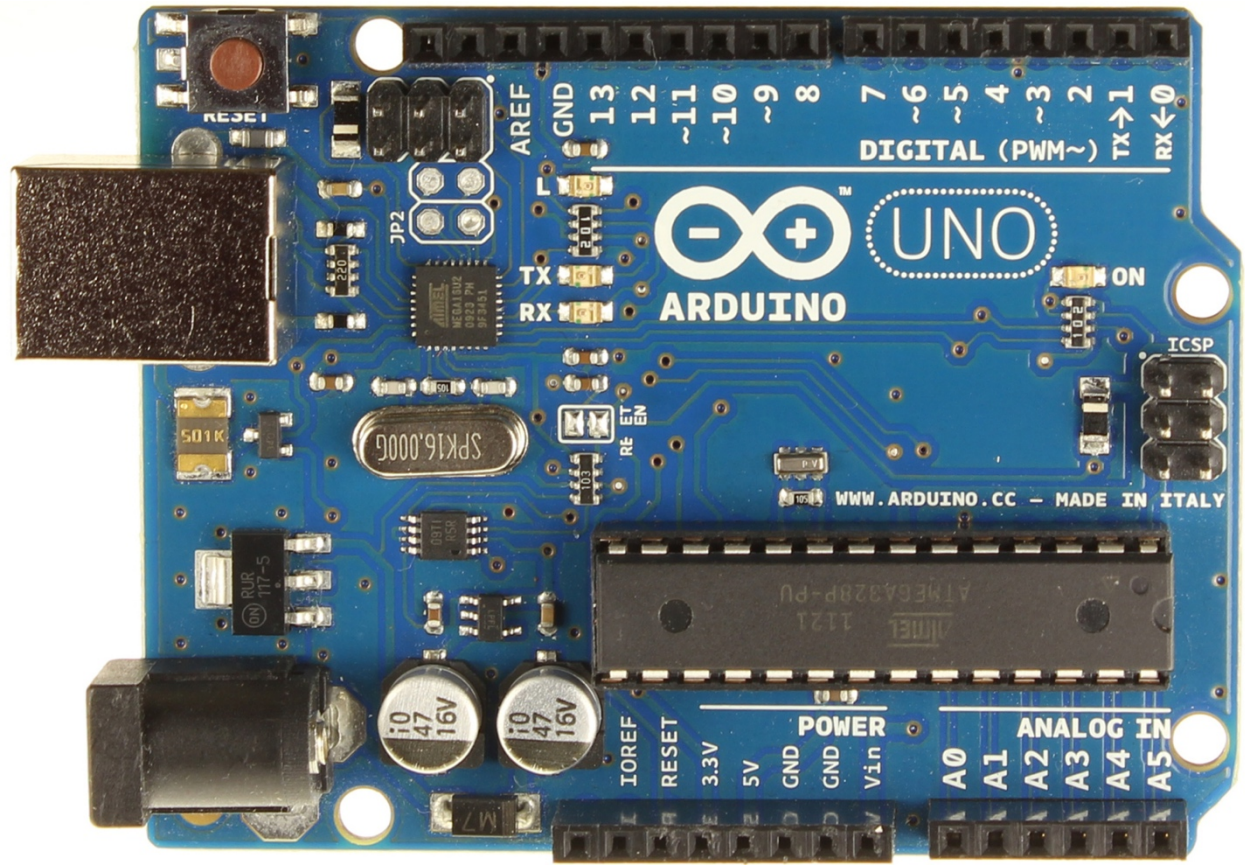


Figura 6. Arduino UNO

3.2 Motor

El motor es pieza fundamental en este proyecto, ya que será el componente encargado de transmitir las ordenes que recibirá de los controles.

Las características del motor van a ser clave, ya que necesitamos un sistema que al realizar los cambios del enfoque no desestabilice la imagen, por lo tanto se ha pensado en utilizar motores paso a paso.

Estos motores presenta ventajas de tener precisión a través de los pulsos eléctricos que generan un desplazamiento angular que puede ser programado. [3]

El motor que se ha elegido es de la marca Nema, modelo 17

3.2.1 Nema 17

El Nema 17 es un motor paso a paso bipolar, se ha elegido por diferentes motivos:

- Hace 200 pasos por vuelta, es decir, cada paso es de 1.8º, esta característica es muy deseable a la hora de buscar precisión a la hora de realizar el enfoque.
- La fuerza que genera es de 3,2Kg/cm, que será suficiente para poder mover el anillo de enfoque
- Las dimensiones(42.3×48mm) ajustadas permitirá que llegado el momento de instalarlo en el objetivo sea más fácil. [4]

Figura 7. Motor paso a paso Nema 17

Una ventaja que se tiene con este tipo de motores, es que existe un controlador de motores paso a paso, llamado EasyDriver.

3.2.1.1 EasyDriver

EasyDriver es un controlador de motores paso a paso compatible con cualquier dispositivo que pueda proporcionar pulsos de 5V.

Gracias a este placa, el manejo del motor se vuelve extremadamente sencillo, ya que con solo dos pines podremos controlar la dirección y el paso del motor.

Además se puede configurar en 4 posiciones con los pines MS1 y MS2, con los que se puede ajustar los pasos, modificando que con cada paso el motor haga o un paso, medio paso, un cuarto o un octavo de paso, afinando más la precisión del sistema. [5]

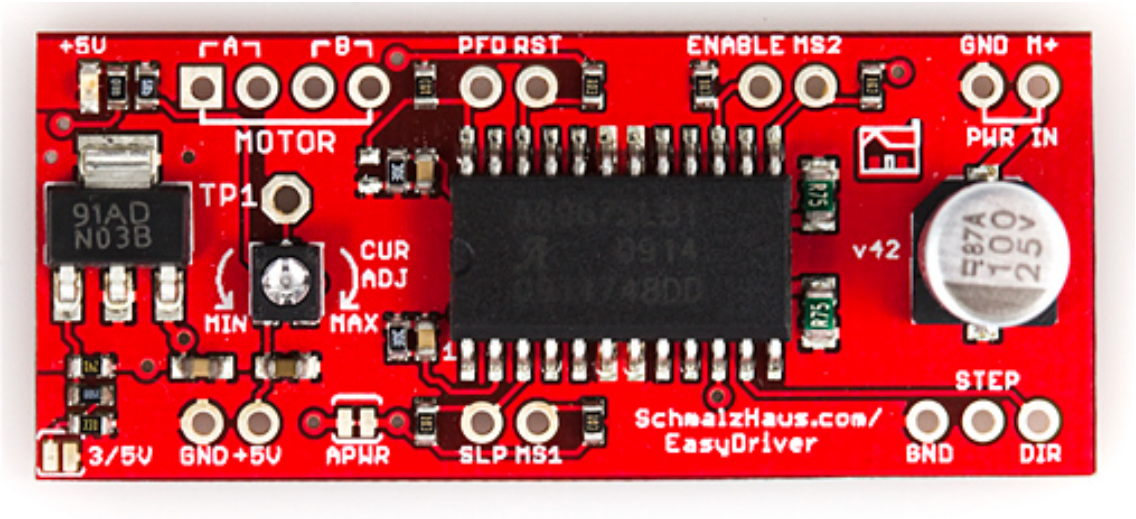


Figura 8. Controlador motor paso a paso EasyDriver

3.3 Controles del sistema

Para controlar el motor, hubieron varias propuestas.

La primera de ellas fue utilizar un potenciómetro, con el cual pudiéramos leer su valor analógico desde Arduino, y según estuviese, manipular el motor paso a paso.

Pero pronto nos dimos cuenta que esta opción, no era ergonómica para utilizar con un gimbal, debido a que sería obligatorio utilizar a un operario que manipule el potenciómetro, ya que a la persona que controlase el gimbal, le resultaría imposible sujetar las dos asas y además mover el pomo del potenciómetro.

3.3.1 Joystick

Seguidamente, pensamos en la utilización de un Joystick en el cual controlemos al motor, moviéndolo en uno de los ejes. Un Joystick no deja de ser dos potenciómetros a 90º que transforman el movimiento en X e Y del mando en una señal eléctrica proporcional a su posición.

Esta opción permite añadirlo a una de las asas del gimbal y la misma persona puede encargarse de sujetarlo mientras utiliza el follow focus.

Figura 9. Joystick Arduino

3.3.2 Pulsadores

Para las demás funciones del gimbal, como guardar posiciones y cambiar la velocidad, se pensó en utilizar pulsadores, que se pueden incorporar en las asas del gimbal al igual que el Joystick permitiendo su uso mientras se sujeta.

Figura 10. Switch

3.3.3 LED

Para señalizar la velocidad a la que funciona el motor en ese instante, se va a colocar una tira de LEDS RGB de Adafruit que programaremos para que señalice la velocidad a la que se está enfocando.

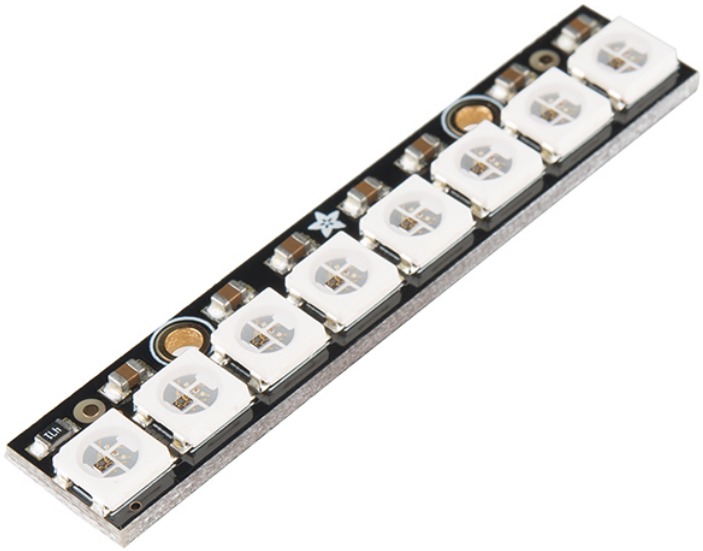


Figura 11. Tira de LED Adafruit

Capítulo 4. DESARROLLO

A continuación vamos a analizar y desglosar el código empleado para el desarrollo de las diferentes funciones que compone y puede ejecutar nuestro producto.

Para ello, me gustaría analizarlo de una forma escalable, es decir, iré analizándolo el código acorde a la evolución de las funciones que hemos ido programando hasta llegar al código final, por lo que lo desglosaremos por fases.

4.1 FASE 1 Joystick

Ahora pues, el primer objetivo del proyecto era poder manejar el motor paso a paso (Stepper Motor) con un external pad y de forma muy precisa.

La primera idea era manejarlo mediante un rotary encoder, el cual nos permitía manejar el motor en función a los golpes hacia derecha o izquierda que nosotros le efectuábamos. Pero tras un cumulo de problemas con latencias y propios problemas que había de transmisión de las órdenes por parte del potenciómetro, al final acabamos desestimando esta opción.

A parte, que a pesar de funcionar a tiempo real, los pasos del motor se manifestaban en forma de golpes y por tanto no obteníamos un sistema del todo preciso, ya que la sensibilidad a la hora de desempeñar el giro era bajísima.

Entonces, al final decidimos probar con un modulo Joystick. Para nuestra sorpresa, nos dimos cuenta que a la hora de realizar un manejo externo del motor, la forma más fácil e intuitiva de aplicación era con el Joystick.

Por consiguiente, todo lo que necesitábamos para esta primera fase era:

- Placa Arduino UNO
- Motor paso a paso (StepperMotor)
- EasyDriver (Controlador del motor)
- Joystick
- Alimentación 12V – 1ª

Aquí adjunto un esquemático de esta primera fase:

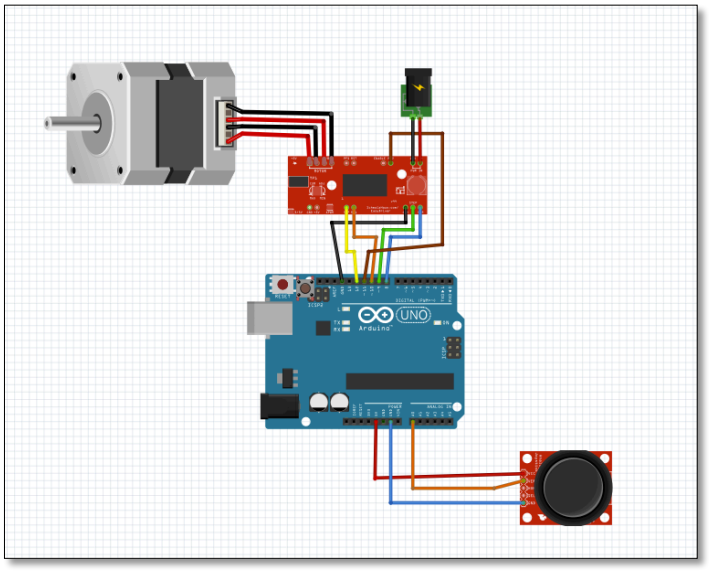


Figura 12. Fase 1- Joystick

Adjuntamos la primera parte del código de la Fase 1:

FASE_1

```
// Asigamos los Pines Digitales

#define DIR 8      // pin DIR del driver -> pin 8 Placa Arduino
#define STEP 9     // pin STEP del driver -> pin 9 Placa Arduino
#define MS1 10     // pin MS1 del driver -> pin 10 Placa Arduino
#define MS2 11     // pin MS2 del driver -> pin 11 Placa Arduino
#define SLEEP 12   // pin SLP del driver -> pin 12 Placa Arduino



// Asigamos los Pines Analogicos

#define Y_pin A0          // Eje Y del joystick -> pin A0 Placa Arduino

int velocidad = 15;  // Marca el delay entre los pasos del motor

void setup() {

   pinMode(MS1, OUTPUT);
   pinMode(MS2, OUTPUT);
   pinMode(DIR, OUTPUT);
   pinMode(STEP, OUTPUT);
   pinMode(SLEEP, OUTPUT);

   digitalWrite(SLEEP, HIGH);
   delay(5);  // Esperar al que el Driver despierte


   digitalWrite(MS1, LOW);      // Metodo de pasos
   digitalWrite(MS2, HIGH);     // Metodo de pasos
}
```

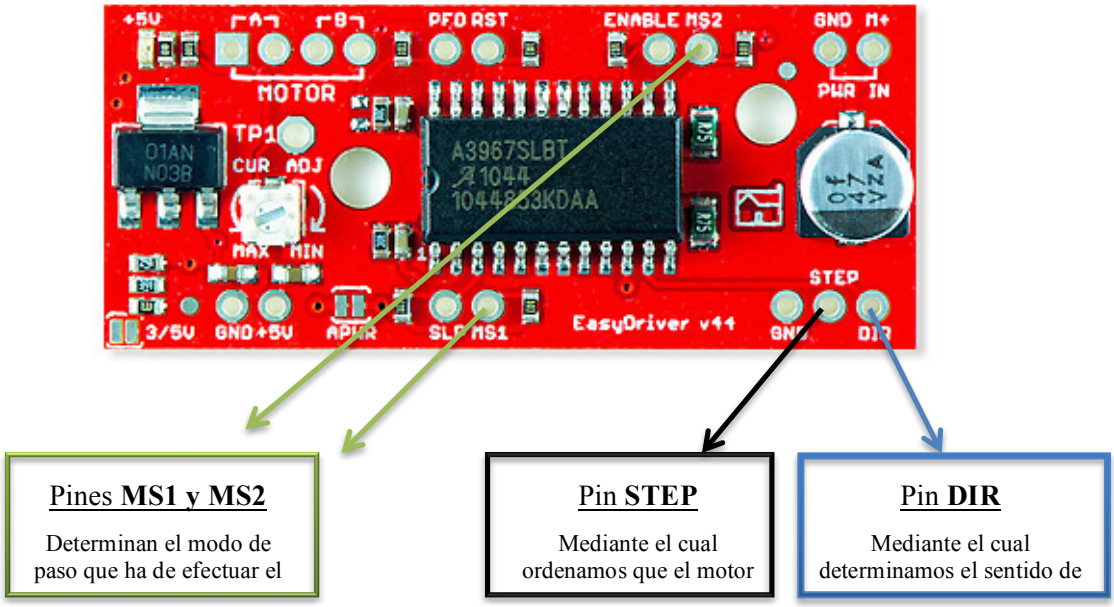
Compilado

```
Sketch uses 1.296 bytes (4%) of program storage space. Maximum is 32.256 bytes.
Global variables use 13 bytes (0%) of dynamic memory, leaving 2.035 bytes for local
```

Figura 13. Fase 1- Joystick(Código 1º parte)

4.1.1 Pines digitales

Primero declaramos los pines del Easydriver en la placa Arduino.




```
void setup() {

   pinMode(MS1, OUTPUT);
   pinMode(MS2, OUTPUT);
   pinMode(DIR, OUTPUT);
   pinMode(STEP, OUTPUT);
   pinMode(SLEEP, OUTPUT);

   digitalWrite(SLEEP, HIGH);
   delay(5);  // Esperar al que el Driver despierte


   digitalWrite(MS1, LOW);      // Metodo de pasos
   digitalWrite(MS2, HIGH);     // Metodo de pasos
}
```

Compilado

Figura 14. Configuración EasyDriver

Declaramos los pines del easydriver, que hemos asignado en la placa, como OUTPUTS ya que la placa ha de ir enviando órdenes al Driver.

- Interpretación de las órdenes:
  - **DIR**
    - LOW == sentido horario
    - HIGH == sentido antihorario

- **STEP**
  - HIGH == el motor da un paso

- **MS1**
  - HIGH / LOW
- **MS2**
  - HIGH / LOW

MS1	MS2	Tipo de paso
LOW	LOW	Paso completo
HIGH	LOW	Medio Paso
LOW	HIGH	Cuarto de Paso
HIGH	HIGH	Octavo de Paso

Como observamos, con estos pines de estado variamos el grado de los pasos que queremos ejecutar. En nuestro caso, como queríamos precisión hemos optado por cada vez que el motor de un paso, avance un cuarto de paso. Por tanto si un paso completo para este motor equivale a 1,8º, nosotros cada vez que demos un paso avanzaremos 0,45º.

4.1.2 Pines Analógicos

Como vemos en el código, asignamos la componente del eje Y del Joystick para que la placa Arduino lea la componente analógica directamente del Joystick en tiempo real y de esta manera podamos desarrollar una función capaz de interpretar la dirección en la que movemos el Joystick para posteriormente ordenar al motor con las correspondientes órdenes.

Por último, antes de comentar la función implementada en la Fase1, nombrar que hemos declarado una variable que nos marcará la velocidad del motor paso a paso.

Aquí adjunto la segunda parte del código de la Fase 1, donde se ejecuta la función del Joystick:

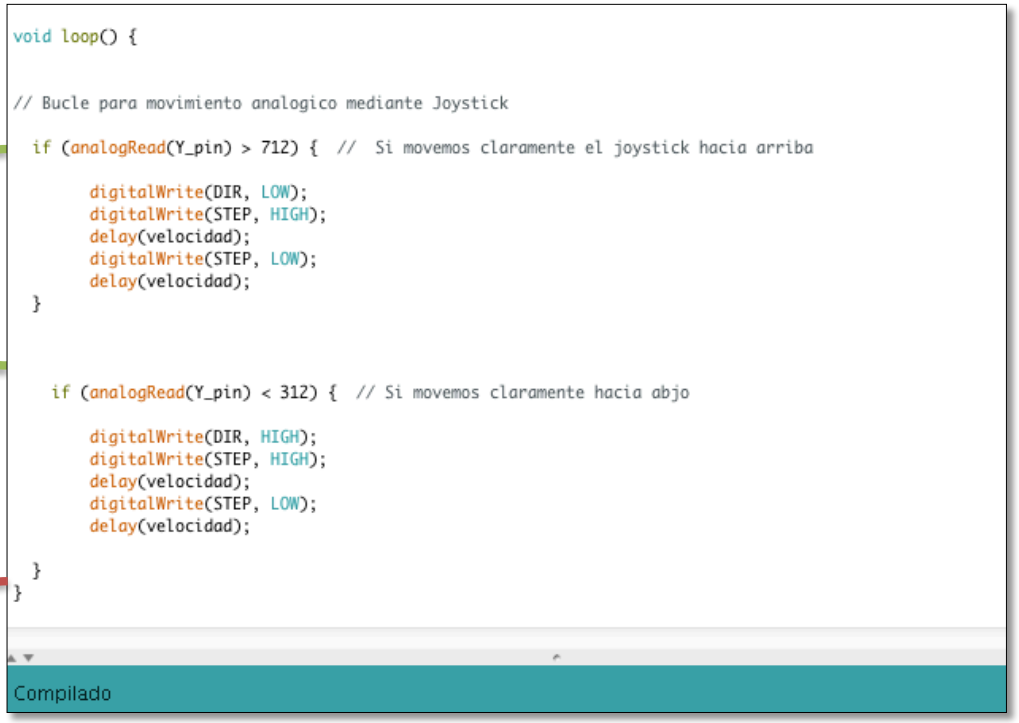


Figura 15. Funcion Joystick

Lo que hace el código es:

1. Lee la posición del eje_Y del Joystick que varía digitalmente entre (**0** y **1023)**
2. Si la posición supera **712**
3. Indicamos que la dirección del motor ha de ser DIR = LOW (sentido horario)
4. Activamos (HIGH) el pin STEP para que el motor avance un paso.
5. Esperamos un delay, que lo marca la variable velocidad
6. Desactivamos (LOW) el pin STEP para que el motor no avance.
7. Esperamos un delay, que lo marca la variable velocidad
8. Y finalmente volvemos a leer la posición del eje_Y del Joystick para continuar avanzando o no.

Lo que hace el código es:

1. Lee la posición del eje_Y del Joystick que varía digitalmente entre (**0** y **1023)**
2. Si la posición es inferior a **312**

3. Indicamos que la dirección del motor ha de ser DIR = HIGH (sentido antihorario)
4. Activamos (HIGH) el pin STEP para que el motor avance un paso.
5. Esperamos un delay, que lo marca la variable velocidad
6. Desactivamos (LOW) el pin STEP para que el motor no avance.
7. Esperamos un delay, que lo marca la variable velocidad
8. Y finalmente volvemos a leer la posición del eje_Y del Joystick para continuar avanzando o no.

Finalmente aclarar, que hemos elegido como tope las posiciones 312 y 712, ya que de esta forma nos aseguramos que el Joystick no se ha tocado por error y por consiguiente queremos efectuar un movimiento horario o antihorario claro.

4.2 Fase 2 Joystick+Pulsador+LEDS

Una vez el Joystick funcionaba correctamente, nuestro siguiente objetivo era poder cambiar la velocidad del motor mediante algún pulsador.

A ello se sumo que estaría interesante que tuviésemos un chivato visual mediante el cual nosotros podríamos visualizar la velocidad en la que nos encontrábamos en todo momento.

Por ello, decidimos sumar al proyecto un módulo de 8 leds (RGB).

Por consiguiente, todo lo que necesitábamos añadimos fue:

- Pulsador
- Módulo Leds RGB

Aquí adjunto un esquemático de esta segunda fase:

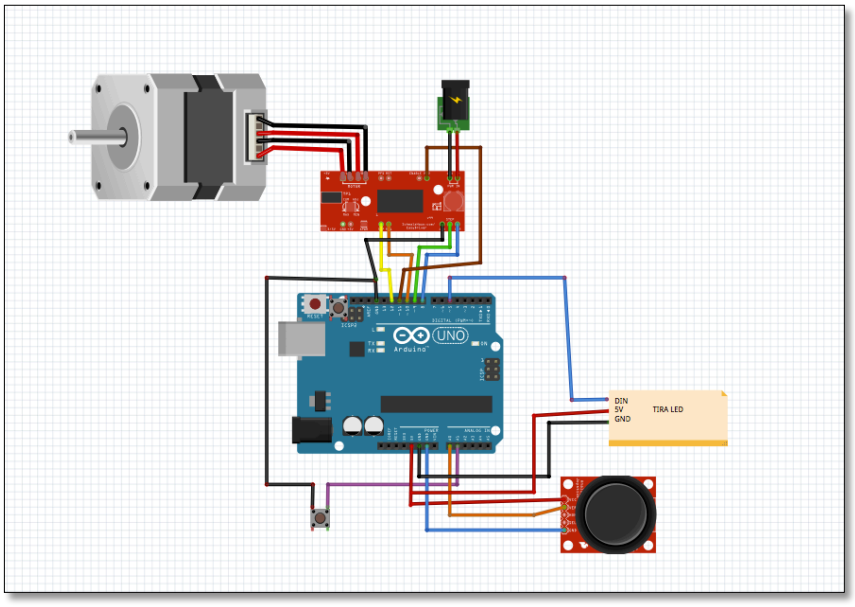


Figura 16. Fase 2

Aquí adjunto la primera parte del código de la Fase 2.

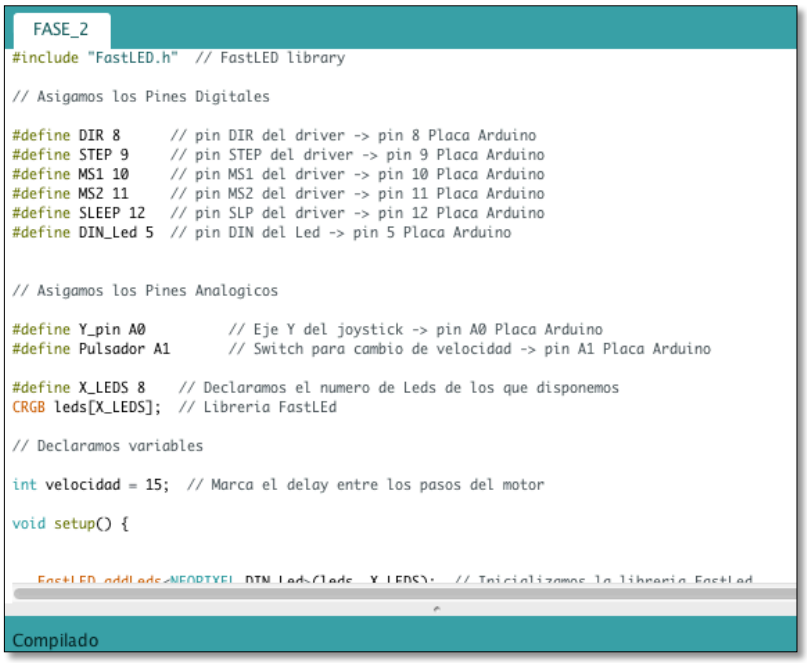


Figura 17. Fase 2, Código

Primeramente, destacar que tuvimos que incluir la librería FastLed para poder trabajar con nuestro módulo Led, mediante la cual podemos asignar los colores que deseamos, brillo del led, etc.

Para poder incluir la librería en arduino tuvimos que seguir los siguientes pasos:

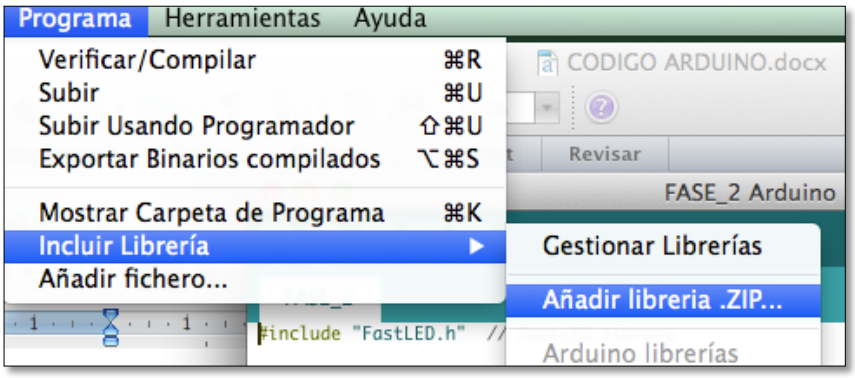


Figura 18. Añadir librerías a Arduino

4.2.1 Pines digitales

Para poder manejar el RGB declaramos el pin **DIN** del módulo led, mediante el cual, a través de arduino le mandaremos la órdenes a ejecutar.

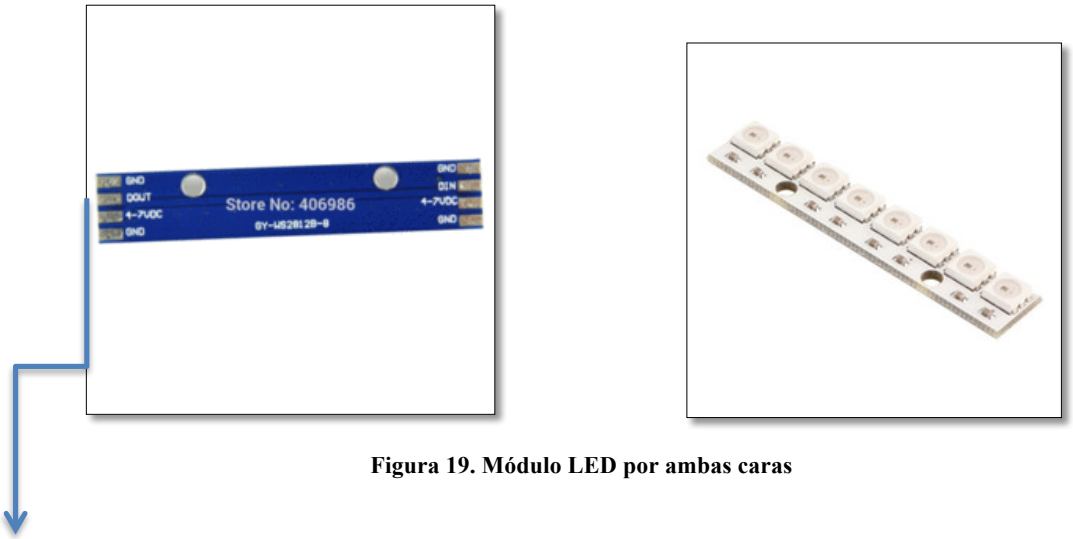


Figura 19. Módulo LED por ambas caras

Como podemos ver se compone de DIN, DOUT, VCC, GND.

Nosotros le hemos asignado DIN ya que le enviaremos órdenes y el módulo tendrá que interpretarlas.

4.2.2 Pines Analógicos

También asignamos un pin al pulsador para la comunicación analógica que nos permita interpretar si hemos presionado el pulsador o no.

Aquí adjunto la segunda parte del código de la Fase 2:

```
void setup() {


  FastLED.addLeds<NEOPIXEL,DIN_Led>(leds, X_LEDS);  // Inicializamos la libreria FastLed
  FastLED.clear();

  // Inicializamos los 2 primeros leds en Azul marcando la primera velocidad

  leds[0] = CRGB::Blue;
  leds[1] = CRGB::Blue;
  FastLED.setBrightness(50);  // Brillo de los leds que queremos mostrar
  FastLED.show();             // Encendemos leds

  pinMode(MS1, OUTPUT);
  pinMode(MS2, OUTPUT);
  pinMode(DIR, OUTPUT);
  pinMode(STEP, OUTPUT);
  pinMode(SLEEP, OUTPUT);


  pinMode(Pulsador, INPUT);
  digitalWrite(Pulsador,HIGH);
```

Figura 20. Código Fase 2.2

Como apreciamos en el código, inicializamos la librería FastLed y seguidamente inicializamos los 2 primeros led de color azul (Blue) dándole una determinada intensidad con la función setBrightness y seguidamente damos la orden de encenderlos mediante show.

Finalmente asignamos el pin de pulsador como entrada y lo dejamos en el estado HIGH, de forma que cuando lo pulsemos, el pulsador se pondrá LOW y mediante ello determinaremos que está pulsado.

Aquí adjunto la tercera parte del código de la Fase 2.

```
if (!digitalRead(Pulsador)) {  //  Si pulsamos el boton

  delay(200);  // delay for deboucing
  switch (velocidad) {  // Analizamos la variable velocid
    case 1:
      velocidad=15;
      FastLED.clear(); // dejamos de mostrar
      leds[0] = CRGB::Blue;
      leds[1] = CRGB::Blue;
      FastLED.setBrightness(50);
      FastLED.show();
      break;
    case 5:
      velocidad=1;
      leds[0] = CRGB::Blue;
      leds[1] = CRGB::Blue;
      leds[2] = CRGB::Green;
      leds[3] = CRGB::Green;
      leds[4] = CRGB::Orange;
      leds[5] = CRGB::Orange;
      leds[6] = CRGB::Red;
      leds[7] = CRGB::Red;
      FastLED.setBrightness(50);
      FastLED.show();
      break;
    case 10:
      velocidad=5;
      leds[0] = CRGB::Blue;
      leds[1] = CRGB::Blue;
      leds[2] = CRGB::Green;
      leds[3] = CRGB::Green;
      leds[4] = CRGB::Orange;
      leds[5] = CRGB::Orange;
      FastLED.setBrightness(50);
      FastLED.show();
      break;
     case 15:
      velocidad=10;
      leds[0] = CRGB::Blue;
      leds[1] = CRGB::Blue;
      leds[2] = CRGB::Green;
      leds[3] = CRGB::Green;
      FastLED.setBrightness(50);
      FastLED.show();
      break;
  }
}
```

Figura 21. Código Fase 2.3

En esta parte implementamos la función mediante la cual podemos cambiar de velocidad y a su vez mostramos la velocidad en la que nos encontramos a través del módulo led.

Para ello nos ayudamos de la función switch.

Lo que hace el código es:

1. Lee el estado del pulsador y si está LOW (pulsado) entra en la función switch.
2. Ahora lo que hace la función es leer la variable velocidad.
3. Comprueba que velocidad=15 (como la hemos inicializado)
4. Entonces pasa al caso 15 y lo ejecuta.
5. Al ejecutarlo cambia la variable velocidad a velocidad=10.
6. Y ahora activa los 2 siguientes led en verde para indicar que nos encontramos en la segunda velocidad.

7. Ahora, en caso de volver a pulsar el pulsador volvería ha efectuar la misma dinámica

    a. Cambiaría al caso 10
    b. Cambiaría velocidad = 5
    c. Enciende los siguientes 2 leds en rojo.
8. Este proceso se repite cada vez que pulsamos el botón.

Finalmente, el resto de código ya venía implementado en la Fase 1.

4.3 Fase 3 Joystick+Pulsador+LEDS+Posiciones

Finalmente, nuestro objetivo era poder memorizar posiciones concretas del motor ya que a la hora de realizar enfoques con cámaras puede interesarnos mucho automatizar los enfoques para determinadas escenas.

A esto le añadimos como guinda, poder incluir un chivato acústico que nos indicase cuando habíamos guardado una cierta posición, así que le incluimos un zumbador.

Por consiguiente, todo lo que necesitábamos añadimos fue:

- 2 Pulsadores
- zumbador

Aquí adjunto un esquemático de esta tercera fase:

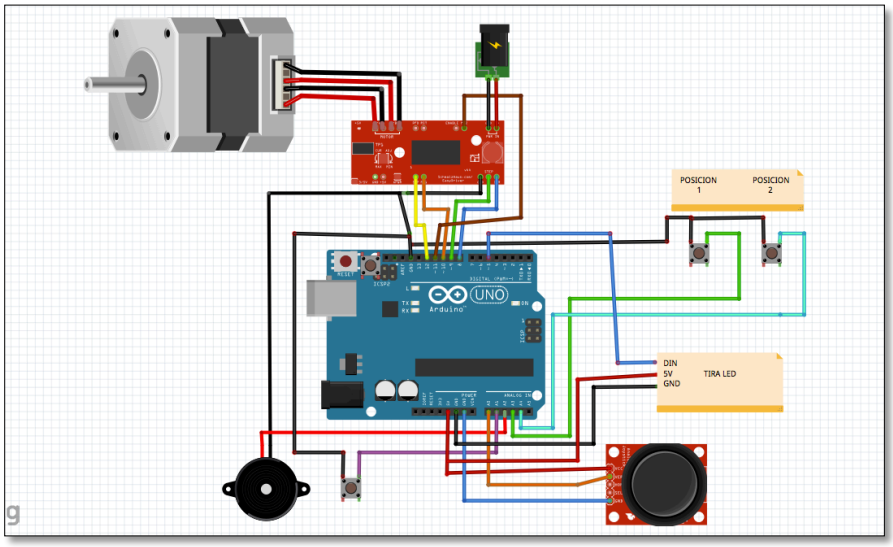


Figura 22. Fase 3

Aquí adjunto la primera parte del código de la Fase 3:

FASE_3

```
#include "FastLED.h"  // FastLED library

// Asigamos los Pines Digitales

#define DIR 8      // pin DIR del driver -> pin 8 Placa Arduino
#define STEP 9     // pin STEP del driver -> pin 9 Placa Arduino
#define MS1 10     // pin MS1 del driver -> pin 10 Placa Arduino
#define MS2 11     // pin MS2 del driver -> pin 11 Placa Arduino
#define SLEEP 12   // pin SLP del driver -> pin 12 Placa Arduino
#define DIN_Led 5  // pin DIN del Led -> pin 5 Placa Arduino


// Asigamos los Pines Analogicos

#define Y_pin A0          // Eje Y del joystick -> pin A0 Placa Arduino
#define Pulsador A1       // Switch para cambio de velocidad -> pin A1 Placa Arduino
#define zumbador A2       // Zumbador -> pin A2 Placa Arduino
#define Button_1 A3       // Boton para la Posicion 1 -> pin A3 Placa Arduino
#define Button_2 A4       // Boton para la Posicion 2 -> pin A4 Placa Arduino


#define X_LEDS 8    // Declaramos el numero de Leds de los que disponemos
CRGB leds[X_LEDS];  // Libreria FastLEd

// Declaramos variables

int StepperPosition=0;    // variable para registrar la posicion del motor
int Posicion_1=0;         // Registrar Posicion 1
int Posicion_2=0;         // Registrar Posicion 2
int result;               // Diferencias de pasos entre posiciones

int time;  //  Para detectar el tiempo durante el cual los pulsadores estan accionados
int velocidad = 15;  // Marca el delay entre los pasos del motor
```

Figura 23. Código Fase 3.1

4.3.1 Pines Analógicos

Asignamos los nuevos pines analógicos para los 2 nuevos pulsadores.

También declaramos 5 nuevas variables para el control a tiempo real de las posiciones del motor y para determinar la diferencia entre el guardado de la posición y el propio camino hasta esa cierta posición. Por ello declaramos una variable time mediante la cual podremos determinar el tiempo que hemos pulsado los botones.

Es decir, la idea es que cuando mantengamos pulsado 1 botón durante un cierto tiempo, el programa interpretará que deseamos guardar esa posición, pero si el botón es pulsado en un mínimo click el programa interpretará que queremos avanzar a la posición que hemos guardado con antelación.

También asignamos un pin de salida para el zumbador, mediante el cual le enviaremos la orden de que se active o no en función de lo que deseemos.

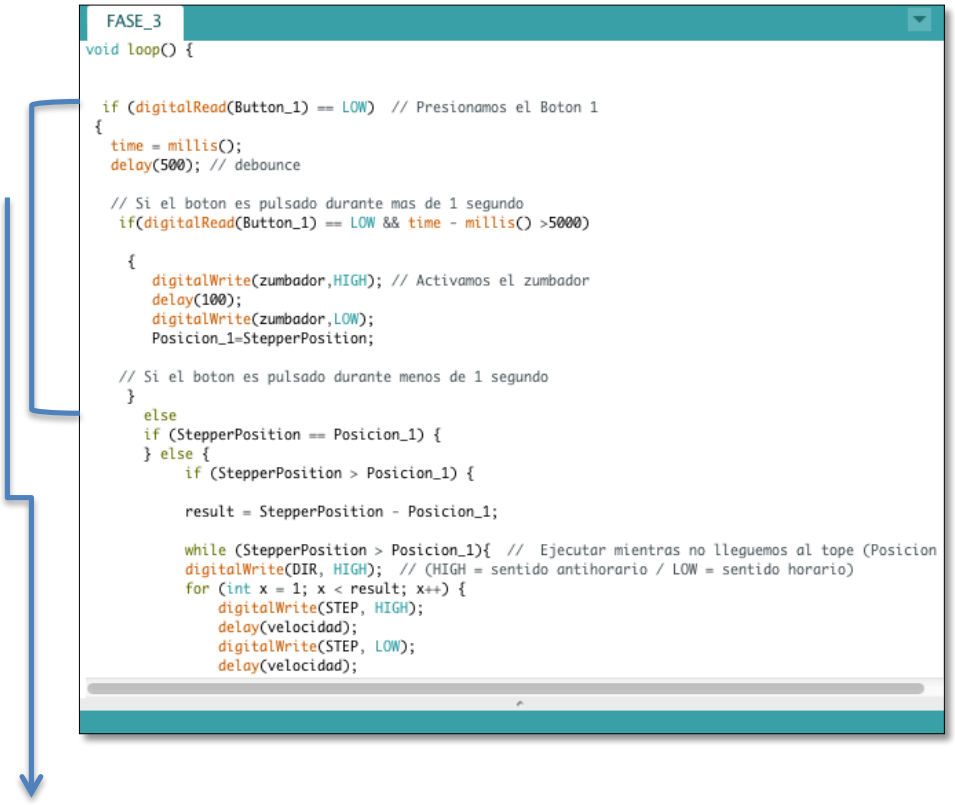
FASE_3

```
void setup() {

   pinMode(zumbador, OUTPUT);
   digitalWrite(zumbador,LOW);

   pinMode(Button_1, INPUT);
   pinMode(Button_2, INPUT);
   digitalWrite(Button_1,HIGH);
   digitalWrite(Button_2,HIGH);

   FastLED.addLeds<NEOPIXEL,DIN_Led>(leds, X_LEDS);  // Inicializamos la libreria FastLed
   FastLED.clear();

   // Inicializamos los 2 primeros leds en Azul marcando la primera velocidad

   leds[0] = CRGB::Blue;
   leds[1] = CRGB::Blue;
   FastLED.setBrightness(50);  // Brillo de los leds que queremos mostrar
   FastLED.show();             // Encendemos leds

   pinMode(MS1, OUTPUT);
   pinMode(MS2, OUTPUT);
   pinMode(DIR, OUTPUT);
   pinMode(STEP, OUTPUT);
   pinMode(SLEEP, OUTPUT);


   pinMode(Pulsador, INPUT);
   digitalWrite(Pulsador,HIGH);

   digitalWrite(SLEEP, HIGH);
   delay(5);  // Esperar al que el Driver despierte
```

Figura 24. Fase 3.2

Aquí especificamos como siempre todas las entradas y salidas.

Ahora vamos de lleno con la nueva función implementada.



```
FASE_3
void loop() {


  if (digitalRead(Button_1) == LOW)  // Presionamos el Boton 1
 {
   time = millis();
   delay(500); // debounce

   // Si el boton es pulsado durante mas de 1 segundo
    if(digitalRead(Button_1) == LOW && time - millis() >5000)

    {
       digitalWrite(zumbador,HIGH); // Activamos el zumbador
       delay(100);
       digitalWrite(zumbador,LOW);
       Posicion_1=StepperPosition;

    // Si el boton es pulsado durante menos de 1 segundo
     }
       else
       if (StepperPosition == Posicion_1) {
       } else {
            if (StepperPosition > Posicion_1) {

            result = StepperPosition - Posicion_1;

            while (StepperPosition > Posicion_1){  //  Ejecutar mientras no lleguemos al tope (Posicion
            digitalWrite(DIR, HIGH);  // (HIGH = sentido antihorario / LOW = sentido horario)
            for (int x = 1; x < result; x++) {
                digitalWrite(STEP, HIGH);
                delay(velocidad);
                digitalWrite(STEP, LOW);
                delay(velocidad);
```

Lo que hace el código es:

1. Lee el estado del pulsador y si está LOW (pulsado) entra en la función switch.
2. Ahora mide el tiempo de pulsación
    a. Si la pulsación supera el segundo (guardar posición), entonces entra en el primer bucle
        i. Entonces activa el zumbador para indicarnos que se ha guardado la posición.
        ii. Guarda la posición del motor en la posición 1.

    b. Si la pulsación no supera el segundo (ir hacia la posición) entonces entra en el segundo bucle

- Si la posición 1 es menor que la posición del motor entonces hace los cálculos necesarios para sacar la diferencia y llegar a la posición 1, dando tantos pasos como necesitemos

```
        if (StepperPosition > Posicion_1) {

        result = StepperPosition - Posicion_1;

        while (StepperPosition > Posicion_1){  //  Ejecutar mier
        digitalWrite(DIR, HIGH);  // (HIGH = sentido antihorario
        for (int x = 1; x < result; x++) {
            digitalWrite(STEP, HIGH);
            delay(velocidad);
            digitalWrite(STEP, LOW);
            delay(velocidad);
       }
        StepperPosition=Posicion_1;
   }
```

- Si la posición 1 es mayor que la posición del motor entonces hace los cálculos necesarios para sacar la diferencia y llegar a la posición 1, dando tantos pasos como necesitemos.

Esta operación se repite para el pulsador 2.

Finalmente comentar que los pasos se van almacenando mediante la función del Joystick.

```
  if (analogRead(Y_pin) > 712) {  //  Si movemos claramente el joystick hacia arriba

        digitalWrite(DIR, LOW);
        digitalWrite(STEP, HIGH);
        delay(velocidad);
        digitalWrite(STEP, LOW);
        delay(velocidad);
        StepperPosition = StepperPosition + 1; // Almacenamos lo pasos
  }


    if (analogRead(Y_pin) < 312) {  // Si movemos claramente hacia abjo

        digitalWrite(DIR, HIGH);  // (HIGH = anti-clockwise / LOW = clockwise)
        digitalWrite(STEP, HIGH);
        delay(velocidad);
        digitalWrite(STEP, LOW);
        delay(velocidad);
        StepperPosition = StepperPosition - 1; // Almacenamos los pasos

  }
}
```

Figura 25. Pasos almacenados

Capítulo 5. CONCLUSIÓNES Y LÍNEAS FUTURAS

El resultado obtenido de todo este proyecto es un asistente de foco o follow focus, hecho con un Arduino UNO y un motor paso a paso, modelo NEMA 17, que se le ha integrado diferentes módulos:

- Controlador EasyDriver
- Tira de 8 LEDS marca Adafruit
- 3 Pulsadores
- Un zumbador

Todo ello, ha compuesto un sistema encabezado por el Arduino que unido e instalado al gimbal, va a llevar a cabo las siguientes funciones:

- El movimiento del motor se ha afinado y configurado a un cuarto de paso para que los pasos sean suaves evitando la brusquedad.
- La dirección del motor se realiza gracias a un Joystick instalado en una de las asas del gimbal.
- Se ha configurado 4 velocidades del stepper, pudiendo visualizar a través de los LED la velocidad actual. Para variar las velocidades, se ha instalado un pulsador que actúa como un switch.
- Se ha programado el Arduino para que realice la función de guardado de posiciones, pudiendo realizar transiciones entre diferentes enfoques. La velocidad de estas transiciones son también configurables con el mismo switch que la velocidad general del motor.

Las próximas mejoras que podrían aplicarse a este Focus Assistant seria :

- Comunicación inalámbrica entre sistema de control y motor. Instalando un módulo Bluetooth, podría servir de comunicación entre el Arduino y los controles, que son el joystick y los pulsadores, con el fin de evitar cables que puedan quedar obstruyendo algún movimiento en caso de no estar bien colocados.
- Calibrado del sistema: esta función sirve para conocer los límites del objetivo, conociendo los grados que puede girar el anillo de enfoque, evitando así forzar este.

Capítulo 6. BIBLIOGRAFÍA

[1].- Raspberry Pi 3 Datasheet. http://uk.rs-online.com/web/p/processor-microcontroller-development-kits/8968660/

[2].- Arduino UNO Datasheet. https://www.arduino.cc/en/Main/ArduinoBoardUno

[3].- Quintero E. A. "Control de Posición para un telescopio con motores paso a paso", 2008

[4].- Stepper Motor NEMA 17. http://www.pbclinear.com/Download/DataSheet/Stepper-Motor-Support-Document.pdf

[5].- EasyDriver 3967. https://www.sparkfun.com/datasheets/Robotics/A3967.pdf

[6].- Stepper Library. https://www.arduino.cc/en/Reference/Stepper

Libros y otra documentación consultada:

- Ribas Lequerica, J. "Manual Imprescindible de Arduino Práctico" 2013