



UNCUYO
UNIVERSIDAD
NACIONAL DE CUYO



FACULTAD DE
INGENIERÍA

Microcontroladores y electrónica de potencia

Proyecto global integrador
Desvío de cajas mediante barrera automática

Ferran Martinez
Legajo N° 12601
Ingeniería en mecatrónica

Índice

Introducción.....	3
Esquema tecnológico	4
Detalle de módulos.....	6
Microcontrolador	6
Motor 1	6
Motor 2	7
Sensor 1 y Sensor 2	8
LCD	9
CP2104 TTL to USB.....	9
Leds	10
Funcionamiento general	11
1 Inicio del sistema	11
2 Detección de una caja y desviación.....	12
3 Caja procesada correctamente.....	13
4 Detección de otra caja	14
5 No detección de una caja que fue desviada	14
Programación	15
Configuración del clock.....	15
Configuración de los pines	16
Configuración de los timers	18
TIM1	18
TIM2	18
TIM3	18
Interrupciones y prioridad de interrupción.....	19
Programa	20
Variables globales.....	20
Funcion menú para la selección de opciones.....	22
Inicializacion del sistema	24
Captura y proceso de las entradas del usuario	24
Interrupción de los sensores	27
Movimiento del motor 1	29
Mensajes por LCD.....	34

Etapas de montaje	35
Conclusiones	38
Referencias	39

Introducción

En el presente proyecto se realiza el desarrollo del control de cajas que circulan en una cinta transportadora y el desvío a otro canal u otra cinta en caso de ser necesario.

La idea para el desarrollo del proyecto reside en imitar los sistemas de clasificación de maletas o valijas que hay en el interior del aeropuerto (véase la figura 1), donde se identifica la maleta y se deriva al avión que le corresponda.

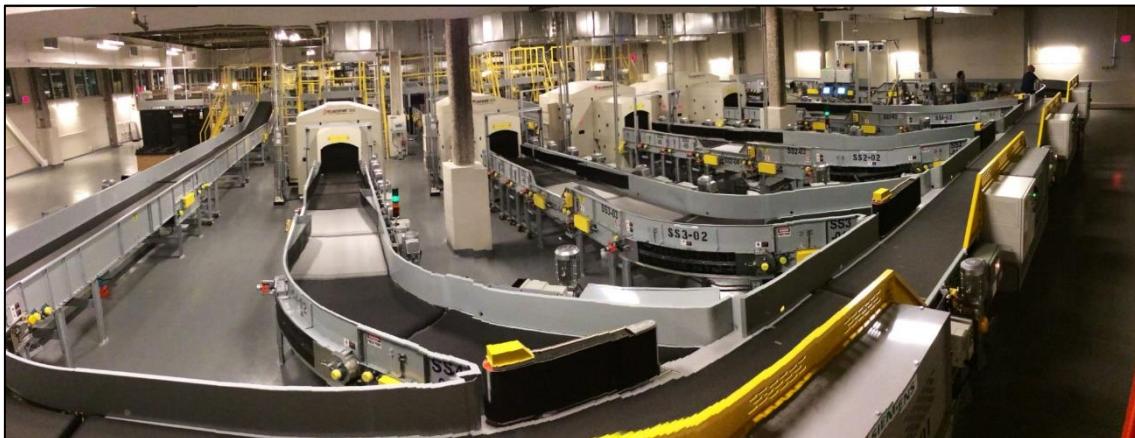


Figura N° 1: Sistema de direccionamiento de equipajes de un aeropuerto.

El objetivo del proyecto es identificar correctamente dos tipos de cajas: las negras y las blancas. Para lograr esto, se utilizará un sensor que será capaz de diferenciar entre ambos tipos de cajas.

Cuando el sensor detecte una caja blanca, la compuerta o barrera se activará para desviar esta caja del camino original. Mientras tanto, las cajas negras continuarán su curso normal.

Esquema tecnológico

Para el correcto funcionamiento del sistema, se plantea la siguiente trama lógica que debe realizar el microcontrolador emplazado.

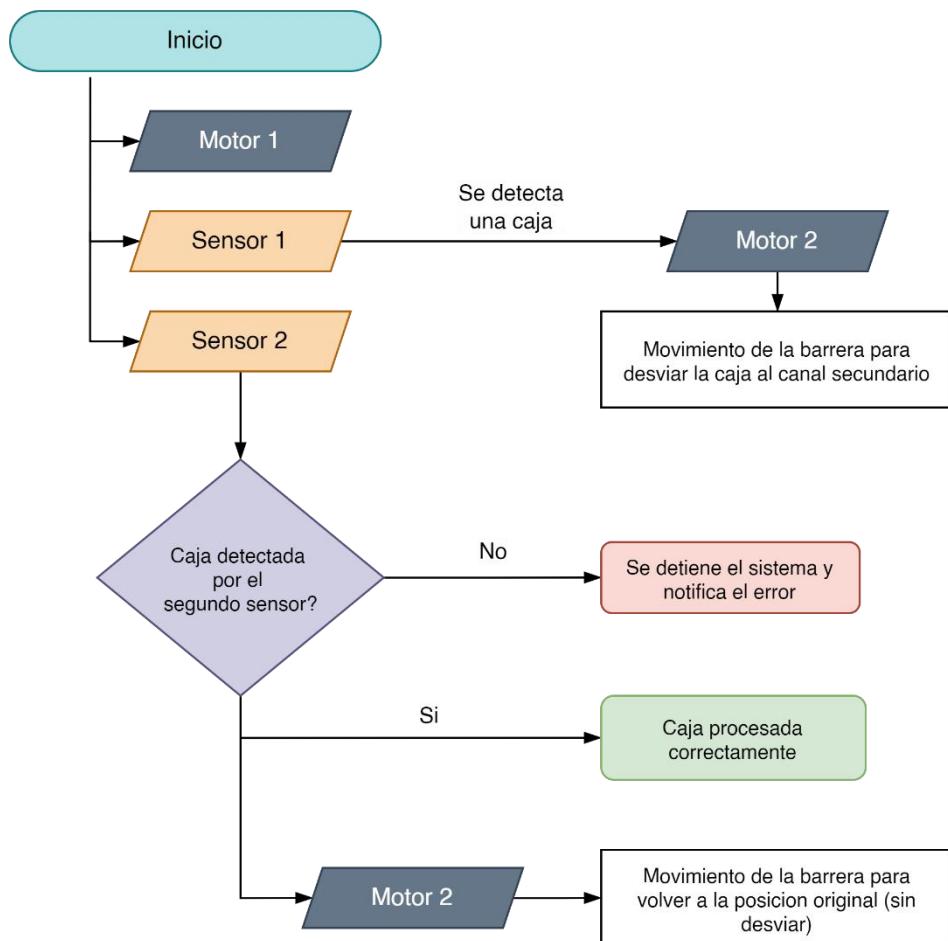


Diagrama N° 1

El funcionamiento propuesto con el diagrama N° 1 se desarrolla de la siguiente forma

1. Al iniciar se pone al sistema en marcha, lo que activa al motor 1 poniendo en movimiento a las cajas en el canal principal. Al iniciar, el sistema tendrá disposición de las señales de los dos sensores colocados a lo largo del canal .
2. Una vez se ha iniciado el sistema, las cajas empiezan a circular mediante la cinta transportadora, y por lo tanto en algún momento se detectará a través del primer sensor una caja blanca.

3. Al detectarse una caja con el primer sensor, se activa el segundo motor, que será el encargado de posicionar la barrera o compuerta de tal forma que la caja blanca tope y sea desviada a un segundo canal.
4. Finalmente, para saber si el proceso del desvío se realizó de forma correcta, un segundo sensor debe detectar que la caja blanca fue desviada correctamente.
5. En caso de que el segundo sensor no detecte una caja, el sistema se detiene, frenando al motor encargado de la cinta transportadora y deshabilitando las señales de los sensores.

Para lograr el cometido propuesto en el diagrama N° 1, se opta por escoger el siguiente esquema de componentes

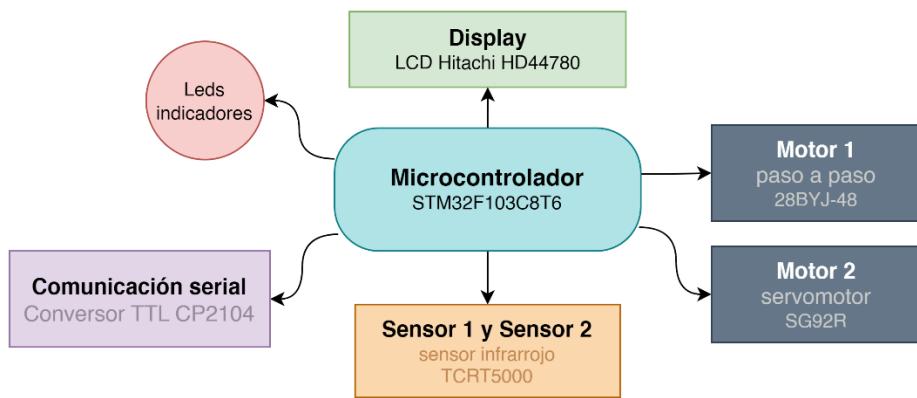


Diagrama N° 2

Detalle de módulos

A continuación, se detallará la elección de los componentes (véase el diagrama Nº 2) utilizados para el desarrollo del proyecto.

Microcontrolador

Se selecciono el microcontrolador STM32F103C8T6 para la gestión del sistema. Está basado en una arquitectura ARM córtex M3 de 32 bits operando en 72MHz de frecuencia.

Dicho microcontrolador viene acoplado en la placa de desarrollo conocida como bluepill, facilitando el acceso a los pines y alimentación del microcontrolador.

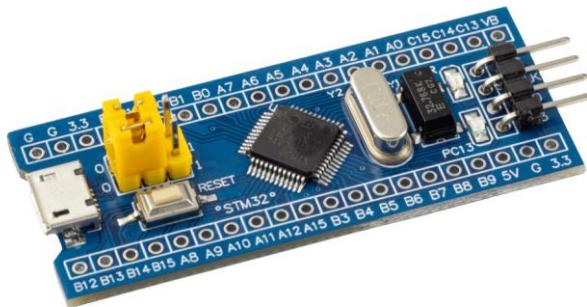


Figura Nº 2: Placa de desarrollo bluepill con el microcontrolador STM32F103C8T6 incorporado

Otras características importantes del microcontrolador

- › Memoria flash de 64kB y 20kB de memoria RAM
- › Interfaz de comunicación USART, SPI e I2C
- › 3 timers de 16 bits y un timer de 32 bits (advanced timer)
- › Opera con tensión de 2V a 3.6V
- › 26 pines de entrada y salida

Con estas características el microcontrolador es suficientemente apto para controlar el funcionamiento del sistema de clasificación de cajas.

Motor 1

Para el motor encargado de accionar la cinta transportadora se ha seleccionado un motor paso a paso unipolar 28BYJ-48 y para manejar el motor se usa el integrado TBD62003APG, que nos permite alimentar cada una de las bobinas del motor según la lógica deseada.

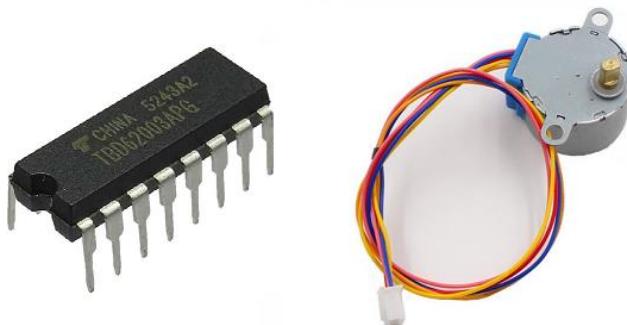


Figura N° 3

El motor ofrece las siguientes características

Tensión de alimentación	5 V
Cantidad de fases	4
Resistencia por bobina	50 Ω
Reducción	1/64 (5.265°/64)
Torque	34.3 mN.m

Tabla N° 1

La selección de un motor paso a paso junto con al integrado es suficiente para el manejo de la cinta transportadora, ya que podemos controlar la velocidad del motor variando la frecuencia con la que se energizan las bobinas y además contar la cantidad de pulsos que requiere que una caja se traslade desde el primer sensor hasta el segundo sensor.

Motor 2

Para el movimiento de la compuerta o barrera se consideraron dos opciones, incluir un servomotor o un motor paso a paso. Se decidió seleccionar el servomotor ya que utiliza menos pines que el paso a paso (considerando que se contaba con un motor paso a paso unipolar) y es más sencillo de programar su accionamiento.



Figura N° 4: Servomotor Tower Pro SG92R con un rango de 180°

Para el proyecto se usa el servomotor ilustrado en la figura N° 4, cuyas especificaciones se detallan en la siguiente tabla:

Tensión de alimentación	4.8 V
Periodo	20 ms
Ancho de pulso	1 ~ 2.5 ms
Rango de rotación	180 °
Velocidad de giro	0.08 s/60°

Tabla N° 2

La barrera que desviará las cajas se encontrara en todo momento en posición de no desviar, es decir que se encontrará en paralelo con la dirección de las cajas. Cuando se detecta una caja esta barrera se colocará gracias al servomotor en una posición de 45°. Por lo tanto, no existe ningún problema al usar el servomotor SG92R para el desarrollo del proyecto.

Sensor 1 y Sensor 2

Para detectar las cajas se usa un sensor infrarrojo tanto para detectar la caja a desviar como para confirmar que la caja fue desviada correctamente. En este caso se usa el sensor TCRT5000, el cual es un módulo seguidor de línea, donde da un nivel de alto lógico hasta que detecta un obstáculo, a excepción de que dicho obstáculo sea de color negro, en tal caso no lo detecta y la salida del sensor seguirá siendo de un alto lógico.



Figura N° 5 : Módulo TCRT500 seguidor de línea

Además, el sensor cuenta con un potenciómetro que le permite ajustar el rango de detección, permitiendo que se coloque a un costado de la cinta transportadora y se pueda ajustar a las características de dicha cinta.

LCD

Para la visualización de datos de interés como puede ser la cantidad de cajas procesadas o en caso de haber un error, se selecciona un LCD simple como es el Hitachi HDD44780 de 16x2 (16 columnas y dos filas). Se selecciono este elemento porque es sencillo de programar y para el caso propuesto cumple los requisitos mínimos para mostrar la información del sistema.



Figura N° 6 : LCD Hitachi HDD44780 16x2 de pantalla azul

CP2104 TTL to USB

El conversor serie va a permitir que nos comuniquemos con el microcontrolador desde nuestro computador o PC a través de un canal serie. Este módulo cuenta con un USB y 6 pines, los cuales se comunicarán con la UART del microcontrolador.



Figura N° 7: Convertidor serie con un módulo USB 2.0

Leds

Considerando que puede surgir un error al desviar las cajas, es necesario no solamente contar con el LCD para dar el aviso, si no que debería haber otros indicadores visuales más sencillos de ver, en este caso, el indicador visual será un led para señalar que se está desviando una caja y otro led intermitente para notificar que una caja no fue desviada.



Figura N° 8 : Leds de colores

Funcionamiento general

Para abordar el desarrollo del proyecto que hace que el funcionamiento del sistema sea lo más completo posible, se ha dividido el mismo en las siguientes partes

1. Inicio y configuración del sistema por comandos en consola.
2. Detección de una caja y su posterior desviación mediante la barrera.
3. Detección de que la caja fue desviada correctamente.
4. Detección de otra caja.
5. No detección de que la caja fue desviada.

De esta forma se espera un mejor entendimiento del programa desarrollado en c que se explicará en la sección posterior.

1 Inicio del sistema

El sistema se encuentra inicialmente detenido, es decir que el motor 1 encargado de la cinta transportadora se encuentra inmovilizado y los sensores estarán inhabilitados. Inicialmente contamos con un menú para configurar el funcionamiento.

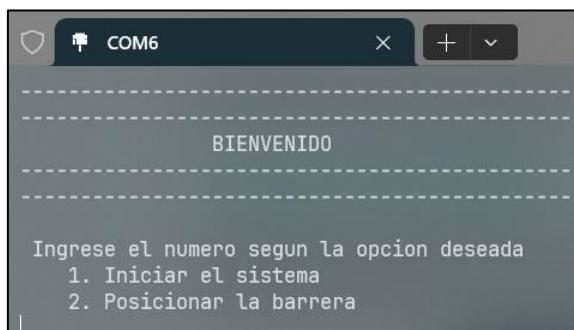


Figura N° 9: Menú principal cuando el sistema está detenido.

En dicho menú podemos iniciar el sistema o, en caso de que haya elementos sobre la cinta, dar una posición inicial como puede ser la posición para desviar cajas.

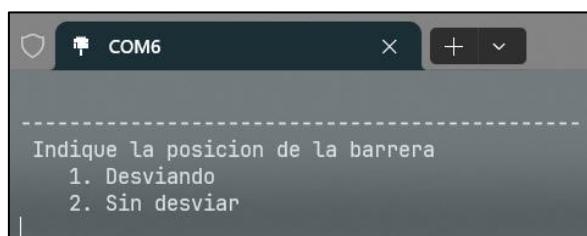


Figura N° 10: Opciones para el cambio de posición de la barrera desde la consola de comandos del PC.

Al iniciar el sistema se abre otro menú en el cual indicaremos la velocidad deseada de la cinta.

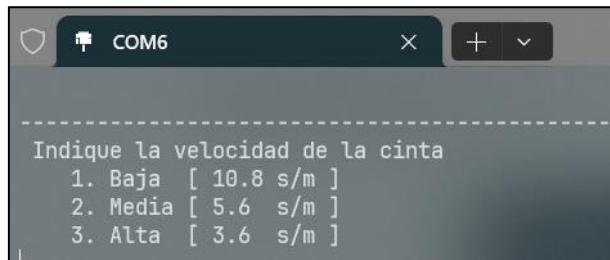


Figura N° 11: Menú para la selección inicial de la velocidad.

En el menú se observa una unidad que no es de velocidad, si no su inversa, y esto es así para ayudar a entender al usuario cuanto tiempo tardará una caja en recorrer un metro, por lo que según sea la distancia entre sensores, será más fácil conocer el tiempo que tardará en completarse el proceso.

Una vez se ha seleccionado una velocidad, se enciende el motor 1 y la cinta empezará a trasladar las respectivas cajas y los sensores se encontrarán habilitados para la detección de cajas.

Cuando el sistema se encuentra en marcha, por consola contamos con el siguiente menú.

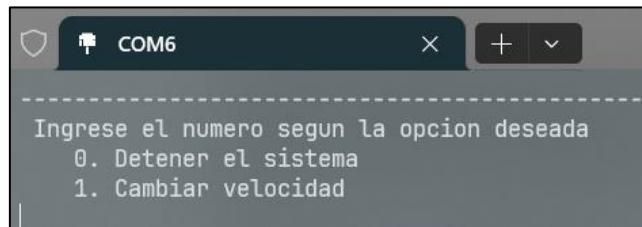


Figura N° 12: Menú del sistema en marcha.

En dicho menú podemos modificar la velocidad de la cinta o en caso de ser necesario, detener completamente el sistema, lo que hará que el motor 1 deje de funcionar, se deshabiliten los sensores y nuevamente el usuario se encuentre en el estado inicial.

2 Detección de una caja y desviación

Una vez se ha iniciado el sistema, los sensores ya se encuentran habilitados para la detección de cajas. Como se comentó en la sección de detalle de módulos, los sensores utilizados no detectan cajas de color negro y a nuestro favor estas cajas no detectadas no serán desviadas y no se iniciará ningún proceso para desviarlas.

Al detectarse otra caja que no sea de color negro, se inicia el proceso para desviar la misma a un canal secundario. La barrera se encuentra en una posición intermedia entre los dos sensores (véase la figura N°13), por lo que al detectar una caja la barrera no cambia de posición automáticamente, si no que espera una cierta cantidad de tiempo antes de hacerlo.

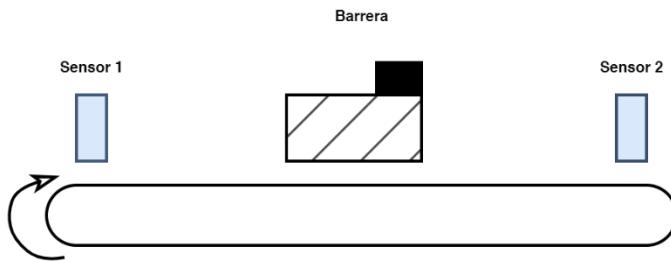


Figura N° 13: Disposición de los sensores y la barrera sobre la cinta transportadora.

Este retraso entre la detección de la caja y el desvío de esta permite que si hay alguna caja negra en camino ésta no sea desviada por equivocación. Por lo tanto, el retraso permitirá a dicha caja negra pasar por la barrera sin ser desviada y luego la barrera se activará para desviar a la caja detectada por el primer sensor.

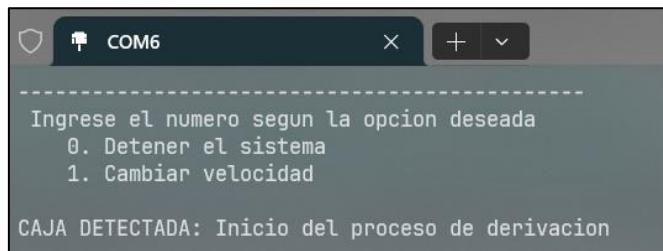


Figura N° 14: Mensaje por puerto serie de la detección de la caja.

3 Caja procesada correctamente

En este punto, el sistema se encuentra en proceso y el motor 2 se encuentra en posición de desviar, entonces solo queda que el segundo sensor detecte la caja que ha sido desviada.

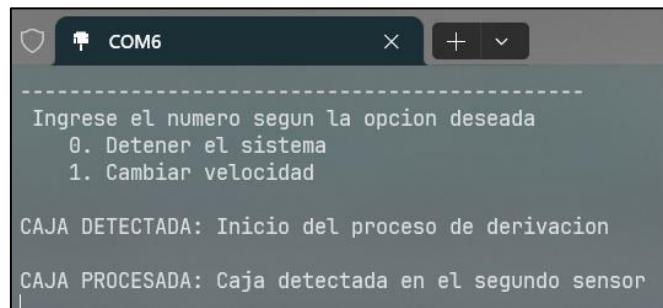


Figura N° 15: Mensaje por puerto serie de la detección de la caja en el segundo sensor.

Durante el tiempo que la caja se encuentre entre el primer y el segundo sensor se activará un led para indicar que el proceso de desviación se está llevando a cabo. Durante este tiempo el sensor debe detectar a la caja y en dicho caso se aumentará en uno el contador de cajas procesadas y se informará a través del LCD.

Finalmente, una vez se procesó la caja se procede a modificar la posición de la barrera a su posición inicial, es decir, sin desviar.

4 Detección de otra caja

Puede ser que en el lapso de 5 segundos que dura el proceso, el primer sensor detecte otra caja a desviar. En este caso se reinicia el proceso, pero manteniendo la posición de la barrera para desviar. De esta forma evitamos accionar múltiples veces el motor 2 cuando se detecten dos cajas consecutivas.

5 No detección de una caja que fue desviada

Puede suceder que se haya detectado una caja y que la misma deba ser desviada, pero resulte que el segundo sensor no la detecte. Por lo tanto, podría considerarse una falla de alguno de los sensores o del motor 2 que mueve la barrera para el desvío de las cajas.

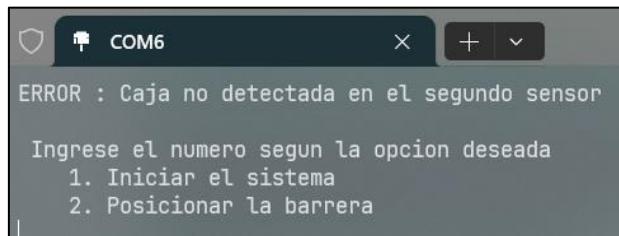


Figura N° 16: Mensaje de error acerca del desvío fallido de la caja.

En este caso se detiene el sistema, esto considera el frenado del motor 1, un aviso de error al usuario y el encendido intermitente de un led de alerta.

Para volver a iniciar el sistema una vez se haya detectado la falla o problema que inicio el error, se vuelve a ingresar al menú de inicio. Al iniciar nuevamente se apagará el led de alerta y se iniciará el sistema como se consideró en el primer punto.

Programación

Al realizar el desarrollo con un microcontrolador del modelo STM32, el proveedor del modelo ofrece un software para su desarrollo, dicho software es el STM32CubelIDE, un entorno de desarrollo para la configuración, programación y debug de proyectos en microcontroladores STM32.



Figura N° 17: Logo del STM32CubelIDE

Este entorno de desarrollo nos facilita el desarrollo proyecto ofreciendo

- Selección del microcontrolador con el que se trabajará, lo que prepara las librerías necesarias y genera los archivos necesarios para su funcionamiento.
- Fácil asignación de los pines del microcontrolador a través de una imagen del microcontrolador mismo.
- Rápida configuración de timers y otras funciones para los pines como lectura, conectividad y analógicos.

Configuración del clock

La configuración más importante para el microcontrolador es la frecuencia de clock, esta frecuencia determina la rapidez con la que el microcontrolador trabajará y será la base para configurar los timers.

Al trabajar con la placa de desarrollo Bluepill el clock viene de un cristal externo, por lo tanto, esto debe indicarse desde el entorno de desarrollo.

RCC Mode and Configuration

Mode	
High Speed Clock (HSE)	Crystal/Ceramic Resonator
Low Speed Clock (LSE)	Disable
<input checked="" type="checkbox"/> Master Clock Output	BYPASS Clock Source
	Crystal/Ceramic Resonator

Figura N° 18: configuración del clock externo.

Una vez configurado el clock externo, ahora se puede modificar la frecuencia a la que trabajará el microcontrolador. Recordemos que la frecuencia máxima es de 72MHz y para este proyecto se ha decidido trabajar con esta misma frecuencia.

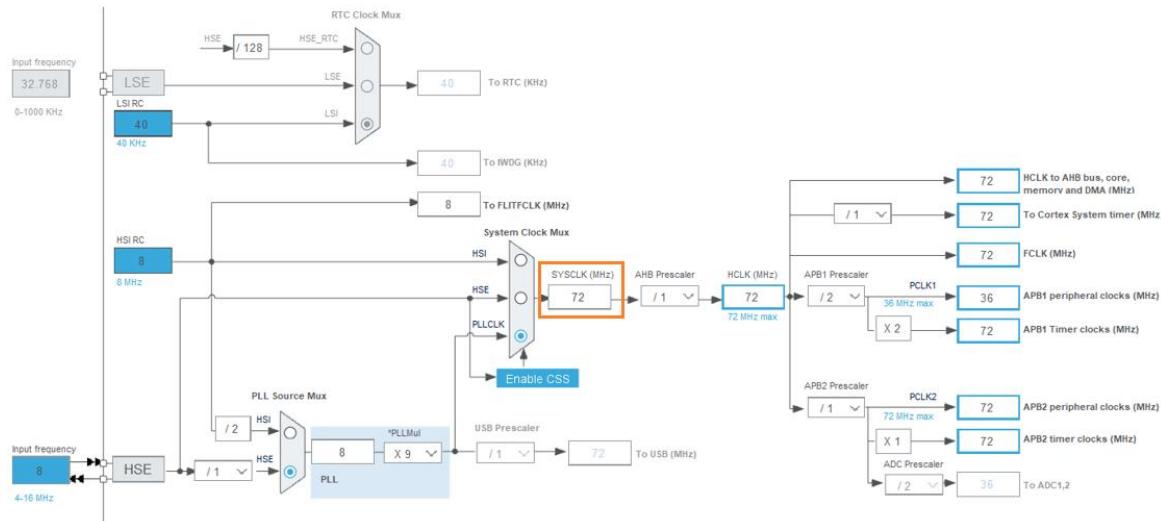


Figura N° 19: Vista del clock configuration del entorno. Se remarcó la selección de 72 MHz para el SYSCLK.

Esta configuración va a ocupar dos pines en nuestro esquema de pines de la siguiente sección. Estos pines serán **RCC_OSC_IN** y **RCC_OSC_OUT**.

Configuración de los pines

Para este proyecto, la disposición de los pines será la siguiente

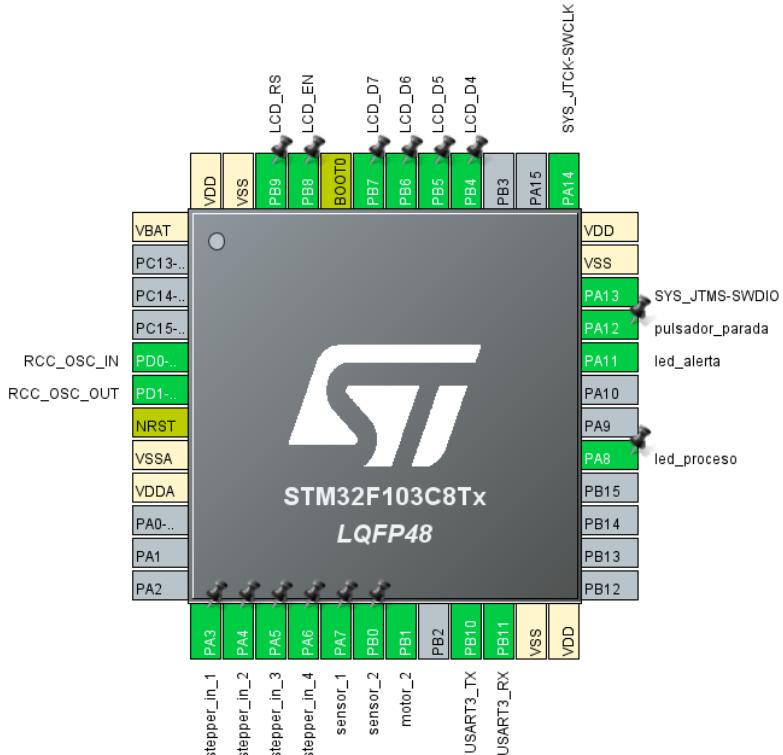


Figura N° 20: Distribución de pines para el microcontrolador seleccionado

En total se ocupan 18 pines y el detalle de cada uno se explica en la siguiente tabla

PIN	Detalle	Configurado como
A3	Salida 1 del motor 1	GPIO
A4	Salida 2 del motor 1	GPIO
A5	Salida 3 del motor 1	GPIO
A6	Salida 4 del motor 1	GPIO
PB1	Salida al PWM del motor 2	PWM
PB0	Entrada de la señal del sensor 2	GP EXTI input
A7	Entrada de la señal del sensor 1	GP EXTI input
A12	Entrada de la señal del pulsador de parada	GP EXTI input
A11	Led de alerta	PWM
A8	Led de proceso	GPIO
B4	Conexión a la terminal D4 del LCD	GP Output
B5	Conexión a la terminal D5 del LCD	GP Output
B6	Conexión a la terminal D6 del LCD	GP Output
B7	Conexión a la terminal D7 del LCD	GP Output
B8	Conexión a la terminal EN del LCD	GP Output
B9	Conexión a la terminal RS del LCD	GP Output
B10	Pin TX para la UART	USART
B11	Pin RX para la UART	USART

Tabla N° 3

Donde:

- GP EXTI input** General purpose input with external interruption
- GP Output** General purpose output
- USART** Universal Sync/Async Receiver/Transmitter

Configuración de los timers

Se configuraron tres de los cuatro timers disponibles en el controlador, donde se ocupan para las siguientes funcionalidades

TIM1: Temporización de 1s para led de alerta

TIM2: Control del motor 1

TIM3: Control del motor 2

Cada timer cuenta a su vez con cuatro canales (channels), permitiendo que un pin determinado tenga una configuración de salida basada en la temporización del propio timer.

TIM1

Usado para encender y apagar el led de alerta cuando una caja no es detectada.
Dicho led se enciende y apaga en un segundo.

Clock source	Internal clock	CHANNEL 4
PSC	36000	
ARR	2000	
Frecuencia	1 Hz	

PWM generation mode 1
Pulse: 1000

TIM2

Usado para controlar la frecuencia a la que se emiten los pulsos al motor paso a paso (motor 1). Recordemos que cada bobina debe ser energizada una cierta cantidad de tiempo para luego energizar la siguiente y seguir así según el patrón de paso designado.

Clock source	Internal clock
PSC	3000
ARR	240
Frecuencia	100 Hz

Para este timer no se usan canales de salida.

TIM3

Usado para controlar la posición del servomotor. Se deben tener en cuenta las características del motor provistas en la tabla N°1.

Clock source	Internal clock	CHANNEL 4
PSC	22	
ARR	65454	
Frecuencia	50 Hz	

PWM generation mode 1
Pulse: 0

Estos valores de PSC y ARR se seleccionaron al momento de comprobar las posiciones del servomotor.

Interrupciones y prioridad de interrupción

Los timers mencionados anteriormente se usarán para diferentes procesos que se comentarán en la sección de programa. Dichos procesos deben ejecutarse cuando el timer se desborda, es decir que el contador del timer alcanza el valor de ARR. Para tomar acción una vez esto ocurre se deben habilitar las interrupciones por desborde de timer.

Otra interrupción que debe habilitarse son las generadas por los sensores, cuyos pines fueron configurados como EXTI, pero igualmente deben habilitarse las interrupciones cuando se lea un nivel de bajo.

NVIC Interrupt Table	Enabled	Preemption Priority
Non maskable interrupt	<input checked="" type="checkbox"/>	0
Hard fault interrupt	<input checked="" type="checkbox"/>	0
Memory management fault	<input checked="" type="checkbox"/>	0
Prefetch fault, memory access fault	<input checked="" type="checkbox"/>	0
Undefined instruction or illegal state	<input checked="" type="checkbox"/>	0
System service call via SWI instruction	<input checked="" type="checkbox"/>	0
Debug monitor	<input checked="" type="checkbox"/>	0
Pendable request for system service	<input checked="" type="checkbox"/>	0
EXTI line0 interrupt	<input checked="" type="checkbox"/>	0
EXTI line[15:10] interrupts	<input checked="" type="checkbox"/>	0
TIM2 global interrupt	<input checked="" type="checkbox"/>	2
USART3 global interrupt	<input checked="" type="checkbox"/>	3
Time base: System tick timer	<input checked="" type="checkbox"/>	15

Tabla N° 4: Tabla de interrupciones ordenadas por prioridad

En la tabla N°4 se encuentran las interrupciones habilitadas en el microcontrolador. En la tercera columna se encuentra el nivel de prioridad de cada una de ellas, siendo 0 una prioridad alta y 15 la prioridad más baja.

Las interrupciones se identifican para cada elemento como

EXTI line0	Sensor 2
EXTI line[10:15]	Sensor 1
	Pulsador de parada
TIM2	Motor 1
USART3	Mensajes por puerto serie

Es importante que el nivel de prioridad de las interrupciones de USART sea lo más bajo posible, ya que se envían y procesan mensajes constantemente y si no se establece dicho nivel de prioridad, estos mensajes pueden interrumpir el funcionamiento del motor 1.

Programa

Para abordar el programa desarrollado en lenguaje c, se dividirá la explicación de este en las siguientes partes

1. Variables globales y estados de los elementos
2. Inicialización del sistema
3. Funcion menú para la selección de opciones
4. Captura y proceso de las entradas del usuario
5. Interrupción de los sensores
6. Movimiento del motor1
7. Mensajes por LCD

De esta forma se espera un mejor entendimiento del programa desarrollado en c que se

Variables globales

```
// =====
// Variables para UART
uint8_t RX_char = 0x00;
//\033[%dm %3d\033[m
uint8_t MSG_cajaD[] = "\n\033[0;31mCAJA DETECTADA:\033[0m Inicio del proceso de derivacion\r\n";
uint8_t MSG_cajaP[] = "\nCAJA PROCESADA: Caja detectada en el segundo sensor\r\n";
uint8_t MSG_cajaEC[] = "\nCAJA EN COLA: Nueva caja detectada durante el proceso\r\n";
uint8_t MSG_pulsadorP[] = "\nSISTEMA DETENIDO: Se pulso el interruptor de parada\r\n";
uint8_t MSG_sisMarcha[] = "\nEl sistema se encuentra procesando una caja, aguarde a que termine\r\n";
uint8_t MSG_opcIncorrecta[] = "\nOpcion incorrecta \r\n";

// =====
// Variables para el sistema
uint16_t contador_cajas = 0;
int pulso_st stepper = 0;

uint16_t pulsos_proceso = 0;
uint16_t pulsos_caja_colas = 0;

const uint16_t PULSOS_XM = 1086;
const uint8_t D_SENSORES = 2;
```

Aquí las variables importantes son las que se usarán para llevar la cuenta de los pulsos del motor y la configuración de la distancia entre sensores. En la parte del movimiento del motor se explicarán los valores de dichas variables.

```
// ESTADOS
// =====
enum sistema {
    detenido, pendiente, en_proceso, caja_procesada
} estado_sistema;

// =====
// Determinamos las frecuencias de los pulsos segun la configuracion del timer
// PSC = 3000
// ARR de 240 → f = 100Hz
// ARR de 120 → f = 200Hz
// ARR de 80 → f = 300Hz
enum estado_motor_1 {
    parado = 0, v_baja = 240, v_media = 120, v_alta = 80
} motor_1;

// =====
// Estado del motor_2
enum estado_motor_2 {
    desviando = 4350, sin_desviar = 2625
} motor_2;

// =====
// Estado de las cajas (cola de cajas)
enum estado_cajas {
    cola_vacia, caja_en_colas
} cola_de_cajas;

// =====
// Para determinar la seleccion del usuario
enum opciones {
    none, por_iniciar, mover_barrera, baja = 100, media = 200, alta = 300
} seleccion_usuario;
```

En la sección estados se identifican los diferentes estados para cada elemento del sistema. En el código expuesto se observan los siguientes elementos

estado_sistema	Determina el estado del proceso de derivar cajas
motor_1	Determina la velocidad del motor 1
motor_2	Determina la posición del motor 2
cola_de_cajas	Para conocer si existe alguna caja en cola
opciones	Para manejar las opciones del usuario en la recepción de datos

Funcion menú para la selección de opciones

```
#include "menu.h"

uint8_t lineas[50] = "-----\r\n";
uint8_t saludo[50] = " BIENVENIDO \r\n";

uint8_t blank[50] = " \r\n"; \r\n";

uint8_t inicio[50] = " Ingrese el numero segun la opcion deseada \r\n"; \r\n";

// Opciones generales
uint8_t opcn_0[50] = " 0. Detener el sistema \r\n"; \r\n";
uint8_t opcn_1[50] = " 1. Iniciar el sistema \r\n"; \r\n";
uint8_t opcn_2[50] = " 2. Posicionar la barrera \r\n"; \r\n";
uint8_t opcn_3[50] = " 1. Cambiar velocidad \r\n"; \r\n";

// Opcion 1 (Seleccion de velocidad)
uint8_t opc1_1[50] = " Indique la velocidad de la cinta \r\n"; \r\n";
uint8_t opc1_2[50] = " Indique el cambio a otra velocidad \r\n"; \r\n";
uint8_t opc1_3[50] = " 1. Baja [ 10.8 s/m ] \r\n"; \r\n";
uint8_t opc1_4[50] = " 2. Media [ 5.6 s/m ] \r\n"; \r\n";
uint8_t opc1_5[50] = " 3. Alta [ 3.6 s/m ] \r\n"; \r\n";

// Opcion 2 (Movimiento de la barrera)
uint8_t opc2_1[50] = " Indique la posicion de la barrera \r\n"; \r\n";
uint8_t opc2_2[50] = " 1. Desviando \r\n"; \r\n";
uint8_t opc2_3[50] = " 2. Sin desviar \r\n"; \r\n";

// Opcion 4 (Error en la deteccion de la caja)
uint8_t MSG_cajaND[50] = "ERROR : Caja no detectada en el segundo sensor\r\n"; \r\n";

// Especiales
uint8_t clear[] = "\e[2J";
```

Estos serán los diferentes mensajes que se mostrarán por consola. Se declaran con anterioridad para usarlos directamente con la función `HAL_UART_Transmit(huart, pData, Size, Timeout)`.

Luego se define la función `menu_seleccion()`, que manejará las vistas por consola para cada momento del proceso.

```
void menu_seleccion(int opc, int vel) {  
  
    HAL_UART_Transmit(&huart3, clear, sizeof(clear), 100);  
  
    // ======  
    // Menu a mostrar cuando se inicia el sistema / Sistema detenido  
    if (opc == 0 || opc == 4) {  
        if (opc == 0) {  
            HAL_UART_Transmit(&huart3, lineas, sizeof(lineas), 100);  
            HAL_UART_Transmit(&huart3, lineas, sizeof(lineas), 100);  
            HAL_UART_Transmit(&huart3, saludo, sizeof(lineas), 100);  
            HAL_UART_Transmit(&huart3, lineas, sizeof(lineas), 100);  
  
        } else {  
            HAL_UART_Transmit(&huart3, MSG_cajaND, sizeof(MSG_cajaND), 100);  
        }  
  
        HAL_UART_Transmit(&huart3, blank, sizeof(lineas), 100);  
        HAL_UART_Transmit(&huart3, inicio, sizeof(lineas), 100);  
        HAL_UART_Transmit(&huart3, opcn_1, sizeof(lineas), 100);  
        HAL_UART_Transmit(&huart3, opcn_2, sizeof(lineas), 100);  
    }  
  
    // ======  
    // Seleccion de velocidades  
    if (opc == 1) {  
        HAL_UART_Transmit(&huart3, blank, sizeof(lineas), 100);  
  
        // Condicionamos si se inicia de cero o es un cambio de velocidad  
        if (vel == 0) {  
            HAL_UART_Transmit(&huart3, opc1_1, sizeof(lineas), 100);  
        } else {  
            HAL_UART_Transmit(&huart3, opc1_2, sizeof(lineas), 100);  
        }  
  
        if (vel != 100) {  
            HAL_UART_Transmit(&huart3, opc1_3, sizeof(lineas), 100);  
        }  
        if (vel != 200) {  
            HAL_UART_Transmit(&huart3, opc1_4, sizeof(lineas), 100);  
        }  
        if (vel != 300) {  
            HAL_UART_Transmit(&huart3, opc1_5, sizeof(lineas), 100);  
        }  
    }  
  
    // ======  
    // Cambio de la posicion de la barrera  
    if (opc == 2) {  
        HAL_UART_Transmit(&huart3, blank, sizeof(lineas), 100);  
        HAL_UART_Transmit(&huart3, opc2_1, sizeof(lineas), 100);  
        HAL_UART_Transmit(&huart3, opc2_2, sizeof(lineas), 100);  
        HAL_UART_Transmit(&huart3, opc2_3, sizeof(lineas), 100);  
    }  
  
    // ======  
    // Sistema en marcha  
    if (opc == 3) {  
        HAL_UART_Transmit(&huart3, lineas, sizeof(lineas), 100);  
        HAL_UART_Transmit(&huart3, inicio, sizeof(lineas), 100);  
        HAL_UART_Transmit(&huart3, opcn_0, sizeof(lineas), 100);  
        HAL_UART_Transmit(&huart3, opcn_3, sizeof(lineas), 100);  
    }  
  
    return;  
}
```

Inicialización del sistema

```
int main(void) {
    // Inicio del LCD
    LCD_Init();
    LCD_Clear();

    // =====
    // Para evitar el salto a PeriodElapsedCallback al iniciar los temporizadores con IT
    // TIM1→SR &= ~TIM_SR UIF_Msk;

    // =====
    // Variables de estado iniciales
    estado_sistema = detenido;
    cola_de_cajas = cola_vacia;
    motor_1 = parado;
    motor_2 = sin_desviar;
    seleccion_usuario = none;

    // Estado de la compuerta (motor_2 inicialmente sin desviar)
    TIM3→CCR1 = motor_2;
    // Podria venir por flash

    // =====
    // Lanzamos el menu por UART
    menu_seleccion(0, 0);
    // Activamos las interrupciones para recibir por UART
    HAL_UART_Receive_IT(&huart3, &RX_char, 1);

    while (1) {
    }
}
```

La función principal inicializa los estados del sistema, inicia la comunicación con el LCD y genera el menú de opciones para el usuario.

Captura y proceso de las entradas del usuario

La captura de las entradas del usuario se realiza mediante la interrupción de la USART, donde la función `HAL_UART_Receive_IT(huart, pData, Size)` habilita la interrupción cada vez que se recibe un carácter desde el pin Rx.

Esta interrupción se maneja con la función de callback
`HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)`

```
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart) {
    UNUSED(huart);

    // . .
    // . .

}
```

Y dentro de dicha función encontramos los diferentes bloques para procesar las opciones del usuario, siendo estos

```
// =====
// El sistema esta detenido, las opciones que se manejan son
// 1. Iniciar
// 2. Cambiar posicion de barrera
if (seleccion_usuario == none && estado_sistema == detenido) {

    // Si esta encendido el led de alerta lo apagamos
    if (htim1.ChannelState[3] == HAL_TIM_CHANNEL_STATE_BUSY) {
        HAL_TIM_PWM_Stop(&htim1, TIM_CHANNEL_4);
    }

    if (RX_char == '1') {
        seleccion_usuario = por_iniciar;
        menu_seleccion(1, 0);
    } else if (RX_char == '2') {
        seleccion_usuario = mover_barrera;
        menu_seleccion(2, 0);
    } else {
        HAL_UART_Transmit(&huart3, MSG_opcIncorrecta,
                          sizeof(MSG_opcIncorrecta), 100);
    }

    RX_char = 0x00;
    HAL_UART_Receive_IT(&huart3, &RX_char, 1);
    return;
}

// =====
// El usuario va a seleccionar una velocidad de inicio / Se inicia el sistema
if (seleccion_usuario == por_iniciar) {
    if (RX_char == '1') {
        seleccion_usuario = baja;
        motor_1 = v_baja;
    } else if (RX_char == '2') {
        seleccion_usuario = media;
        motor_1 = v_media;
    } else if (RX_char == '3') {
        seleccion_usuario = alta;
        motor_1 = v_alta;
    } else {
        HAL_UART_Transmit(&huart3, MSG_opcIncorrecta,
                          sizeof(MSG_opcIncorrecta), 100);

        RX_char = 0x00;
        HAL_UART_Receive_IT(&huart3, &RX_char, 1);
        return;
    }

    // Menu → sistema en marcha
    menu_seleccion(3, 0);
    // Set de estados
    estado_sistema = pendiente;

    // Iniciamos motor_1 y motor_2
    TIM2→ARR = motor_1;
    if (htim2.State != HAL_TIM_STATE_BUSY)
        HAL_TIM_Base_Start_IT(&htim2);
    HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_4);

    RX_char = 0x00;
    HAL_UART_Receive_IT(&huart3, &RX_char, 1);
    return;
}
```

```
// =====
// Con el sistema en marcha solo se pueden hacer dos cosas, siempre
// y cuando el sistema se encuentre en estado de pendiente
// 0. Detener el sistema
// 1. Cambiar velocidad
if (estado_sistema == pendiente) {

    if (RX_char == '0') {
        estado_sistema = detenido;
        motor_1 = parado;
        seleccion_usuario = none;

        // Paramos a motor_1 y motor_2
        HAL_TIM_Base_Stop_IT(&htim2);
        stop_stepper();
        HAL_TIM_PWM_Stop(&htim3, TIM_CHANNEL_4);

        // Mostramos menu de inicio
        menu_seleccion(0, 0);

    } else if (RX_char == '1') {
        seleccion_usuario = por_iniciar;
        menu_seleccion(1, (int) seleccion_usuario);

    } else {
        HAL_UART_Transmit(&huart3, MSG_opcIncorrecta,
                          sizeof(MSG_opcIncorrecta), 100);
    }

    RX_char = 0x00;
    HAL_UART_Receive_IT(&huart3, &RX_char, 1);
    return;

} else if (estado_sistema != pendiente && estado_sistema != detenido) {
    // El sistema esta en marcha pero procesando una caja, entonces
    // hay que esperar a que el proceso termine
    HAL_UART_Transmit(&huart3, MSG_sisMarcha, sizeof(MSG_sisMarcha), 100);

    RX_char = 0x00;
    HAL_UART_Receive_IT(&huart3, &RX_char, 1);
    return;
}
```

```
// =====
// Con el sistema detenido, el usuario puede decidir la posicion inicial
// de la barrera
// 1. Desviando
// 2. Sin desviar
if (seleccion_usuario == mover_barrera && estado_sistema == detenido) {

    if (RX_char == '1') {
        motor_2 = desviando;
    } else if (RX_char == '2') {
        motor_2 = sin_desviar;
    } else {
        HAL_UART_Transmit(&huart3, MSG_opcIncorrecta,
                           sizeof(MSG_opcIncorrecta), 100);

        RX_char = 0x00;
        HAL_UART_Receive_IT(&huart3, &RX_char, 1);
        return;
    }

    // Posicionamos la barrera
    TIM3→CCR4 = motor_2;
    HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_4);

    // Seteamos menu y estado de seleccion
    seleccion_usuario = none;
    menu_seleccion(0, 0);

    RX_char = 0x00;
    HAL_UART_Receive_IT(&huart3, &RX_char, 1);
    return;
}
```

Interrupción de los sensores

Las interrupciones externas se manejan con la función `HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)`, que se llamará cada vez que se detecte una interrupción por parte de alguno de los sensores o del pulsador de parada.

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) {
    UNUSED(GPIO_Pin);
    // ...
    // ...
}
```

```
// =====
// SENSOR 1
// Se detecta una caja, pero se tiene en cuenta que quizas se este procesando
// una caja en ese momento
if (GPIO_Pin == sensor_1_Pin) {

    // Se detecto una caja, entonces se inicia el proceso de desviacion
    if (estado_sistema == pendiente) {
        estado_sistema = en_proceso;

        // Se inicia el contador de pulsos
        pulsos_proceso = 0;

        // Se indica con el led de proceso
        HAL_GPIO_WritePin(led_proceso_GPIO_Port, led_proceso_Pin, 1);

        // Se transmite por UART
        HAL_UART_Transmit(&huart3, MSG_cajaD, sizeof(MSG_cajaD), 100);

    }

    // En el caso de que una caja se este procesando, entonces se coloca
    // una caja en cola
    if (estado_sistema == en_proceso && motor_2 == desviando) {
        cola_de_cajas = caja_en_colas;
    }

}

// =====
// SENSOR 2
// La caja fue derivada/procesada correctamente
if (GPIO_Pin == sensor_2_Pin && estado_sistema == en_proceso
    && motor_2 == desviando) {
    estado_sistema = caja_procesada;
    contador_cajas++;

    // Mensajes para el LCD
    LCD_Clear();
    LCD_msg("Caja procesada", 1, 1);
    char aux[12];
    char texto[] = "Cantidad : ";
    int_to_str((int) contador_cajas, aux);
    strcat(texto, aux);
    LCD_msg(texto, 2, 1);

    // Se comunica a traves de uart
    HAL_UART_Transmit(&huart3, MSG_cajaP, sizeof(MSG_cajaP), 100);

}
```

```
// =====
// PULSADOR DE PARADA
// Se usa para interrumpir y detener al sistema
if (GPIO_Pin == pulsador_parada_Pin && estado_sistema != detenido) {
    estado_sistema = detenido;
    seleccion_usuario = none;

    // Detenemos motor_1 y motor_2
    HAL_TIM_Base_Stop_IT(&htim2);
    stop_stepper();
    HAL_TIM_PWM_Stop(&htim3, TIM_CHANNEL_4);

    // Mostramos menu de inicio
    HAL_UART_Transmit(&huart3, MSG_pulsadorP, sizeof(MSG_pulsadorP), 100);
    menu_seleccion(0, 0);
}
```

Movimiento del motor 1

Como se habrá visto en la sección de variables globales y estados, se declaran algunas relacionadas con los pulsos del motor, pulsos del proceso y para las velocidades del motor se usan valores de 240, 120 y 80.

Para llegar a estos valores primero se empieza conociendo que el motor paso a paso 28BYJ-48 necesita de 2048 pulsos de paso completo para que el eje de este realice una revolución.

Cuando trabaja con una frecuencia de 100Hz, a la salida del eje obtenemos una velocidad de

$$\frac{100 \text{ Hz}}{2048} 2\pi = 0.3 \frac{\text{rad}}{\text{s}}$$

Esta velocidad, en una rueda de 10cm de diámetro que mueve a la cinta transportadora, se traduce en una velocidad lineal de 0,015 m/s. Si mantenemos una distancia de dos metros entre los dos sensores, una caja tardaría 133 segundos o unos 2 minutos y medio.

Por lo tanto, para agilizar el proceso y ofrecer al usuario una gama de velocidades variada, se opta por incluir una transmisión que aumente la velocidad a la salida.

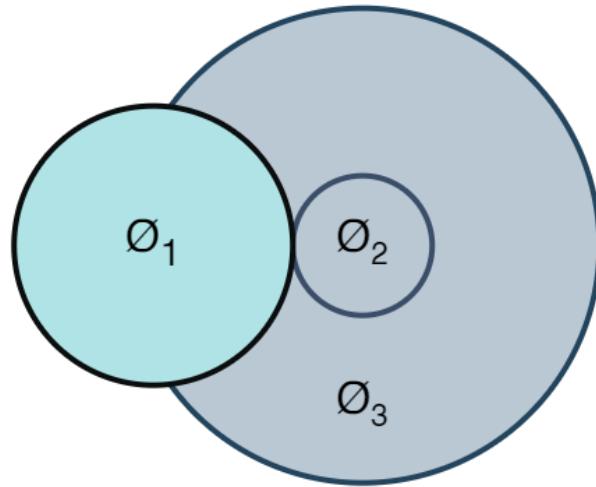


Figura N° 21: Relación de transmisión de la cinta transportadora.

Si tomamos los siguientes valores para los diámetros del juego de engranajes de la figura N°21

$$\varnothing_1 \quad 6 \text{ cm}$$

$$\varnothing_2 \quad 1 \text{ cm}$$

$$\varnothing_3 \quad 10 \text{ cm}$$

Y planteando diferentes frecuencias de trabajo, se obtienen las siguientes velocidades en la cinta transportadora

Frecuencia de trabajo	Valor de ARR en TIM2	Velocidad	Tiempo por metro recorrido
100 Hz	240	0,09 m/s	10,86 s/m
200 Hz	120	0,18 m/s	5,43 s/m
300 Hz	80	0,27 m/s	3,62 s/m

Y finalmente, la relación más importante para el control de las cajas es la cantidad de pulsos que debe realizar el motor para que una caja recorra un metro, siendo este valor de 1086 pulsos por metro recorrido.

Esto explica la constante [PULSOS_XM](#), que determina cuantos pulsos se necesitan para que la caja recorra un metro y así determinar cuántos pulsos se necesitan para recorrer la distancia entre los sensores, definida como [D_SENSORES](#).

Para movilizar al motor paso a paso se usan las siguientes funciones

```
// Da un pulso al stepper en un sentido u otro
void init stepper(int pulse, int dir) {
    if (dir == 1) {
        HAL_GPIO_WritePin(STEPPER gpio_port[0], STEPPER.in_pin[0],
                          full_step[pulse][0]);
        HAL_GPIO_WritePin(STEPPER gpio_port[1], STEPPER.in_pin[1],
                          full_step[pulse][1]);
        HAL_GPIO_WritePin(STEPPER gpio_port[2], STEPPER.in_pin[2],
                          full_step[pulse][2]);
        HAL_GPIO_WritePin(STEPPER gpio_port[3], STEPPER.in_pin[3],
                          full_step[pulse][3]);
    } else if (2) {
        HAL_GPIO_WritePin(STEPPER gpio_port[0], STEPPER.in_pin[0],
                          full_step_inv[pulse][0]);
        HAL_GPIO_WritePin(STEPPER gpio_port[1], STEPPER.in_pin[1],
                          full_step_inv[pulse][1]);
        HAL_GPIO_WritePin(STEPPER gpio_port[2], STEPPER.in_pin[2],
                          full_step_inv[pulse][2]);
        HAL_GPIO_WritePin(STEPPER gpio_port[3], STEPPER.in_pin[3],
                          full_step_inv[pulse][3]);
    }
}

void stop stepper() {
    HAL_GPIO_WritePin(STEPPER gpio_port[0], STEPPER.in_pin[0], 0);
    HAL_GPIO_WritePin(STEPPER gpio_port[1], STEPPER.in_pin[1], 0);
    HAL_GPIO_WritePin(STEPPER gpio_port[2], STEPPER.in_pin[2], 0);
    HAL_GPIO_WritePin(STEPPER gpio_port[3], STEPPER.in_pin[3], 0);
}
```

Donde

```
uint8_t full_step[4][4] = {
    {1, 1, 0, 0},
    {0, 1, 1, 0},
    {0, 0, 1, 1},
    {1, 0, 0, 1}
};

uint8_t full_step_inv[4][4] = {
    {0, 0, 1, 1},
    {0, 1, 1, 0},
    {1, 1, 0, 0},
    {1, 0, 0, 1}
};

// Configuración de los pines para el stepper
const STEPPER_PINS STEPPER = {
    {GPIOA, GPIOA, GPIOA, GPIOA},
    {GPIO_PIN_3, GPIO_PIN_4, GPIO_PIN_5, GPIO_PIN_6}
};
```

Ahora que sabemos cómo se moviliza al motor1 y cuáles son las constantes que ayudan a llevar la cuenta de los pulsos, se presenta el código que evalúa cada uno de los pulsos determinado por la velocidad seleccionada. Este código se ejecuta dentro de la función de callback `HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)`.

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
    UNUSED(htim);
    // . . .
    // . . .
}
```

Que contiene la siguiente parte

```
init_stepper(pulso_st stepper, (int) 1);

// El pulso del stepper toma valores entre 0 y 3
pulso_st stepper++;
if (pulso_st stepper >= 4) {
    pulso_st stepper = 0;
}

if (estado_sistema == en_proceso || estado_sistema == caja_procesada
    || cola_de_cajas == caja_en_colas) {

    pulsos_proceso++;
    // Segundo contador de pulsos para la caja en cola
    if (cola_de_cajas == caja_en_colas) {
        pulsos_caja_colas++;
    }

    // Movimiento de motor_2 a la mitad del trayecto
    if ((pulsos_proceso >= (uint16_t)((PULSOS_XM * D_SENSORES) / 2))
        && estado_sistema == en_proceso) {

        motor_2 = desviando;
        TIM3→CCR4 = motor_2;
    }

    // . . .
    // . . .
}
```

Los pulsos de proceso llevarán la cuenta de la cantidad de pulsos que necesita el motor 1 para trasladar la caja desde el primer sensor hasta el segundo sensor. Es por ello por lo que esta variable solo aumenta cuando el sistema se encuentre en proceso, haya una caja en cola o la caja haya sido procesada lo que habilitará pasar los pulsos de la cola en la caja a los pulsos del proceso.

Finalmente, cuando los pulsos alcanzan los valores calculados anteriormente, se procede a realizar lo indicado en la siguiente porción de código.

```
// ~~~~~
// La caja deberia haber alcanzado el segundo sensor
if (pulsos_proceso >= (PULSOS_XM * D_SENSORES)) {

    // En caso de no haber detectado la segunda caja
    // → Se enciende el led de alerta
    // → Se para motor_1
    // → Se avisa por sistema
    if (estado_sistema != caja_procesada) {
        estado_sistema = detenido;
        seleccion_usuario = none;

        // Se detiene motor_1 y motor_2
        HAL_TIM_Base_Stop_IT(&htim2);
        stop_steer();
        HAL_TIM_PWM_Stop(&htim3, TIM_CHANNEL_4);

        // Se enciende el led de alerta
        HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_4);

        // Se reinicia el menu por UART
        menu_seleccion(4, 0);
        return;
    }

    // Si habia una caja en cola, entonces movemos la cantidad de pulsos
    // contados de la caja en cola al proceso general
    if (cola_de_cajas == caja_en_cola) {
        cola_de_cajas = cola_vacia;
        estado_sistema = en_proceso;

        pulsos_proceso = pulsos_caja_cola;
        pulsos_caja_cola = 0;
        return;
    }

    // Finalmente si era una sola caja y fue procesada,
    // se coloca la barrera nuevamente a sin_desviar
    motor_2 = sin_desviar;
    TIM3→CCR4 = motor_2;

    // Se apaga el led de proceso
    HAL_GPIO_WritePin(led_proceso_GPIO_Port, led_proceso_Pin, 0);

    // Y se cambia al estado pendiente
    estado_sistema = pendiente;
}
```

Mensajes por LCD

Se usa una librería (ECUAL) para el manejo del LCD, facilitando así el desarrollo del programa y evitando posibles errores que puedan surgir al momento de realizar la comunicación con el mismo.

La librería cuenta con las siguientes funciones

```
void LCD_Init();           // Initialize The LCD For 4-Bit Interface
void LCD_Clear();          // Clear The LCD Display
void LCD_SL();             // Shift The Entire Display To The Left
void LCD_SR();             // Shift The Entire Display To The Right
void LCD_CMD(unsigned char); // Send Command To LCD
void LCD_DATA(unsigned char); // Send 4-Bit Data To LCD
void LCD_Set_Cursor(unsigned char, unsigned char); // Set Cursor Position
void LCD_Write_Char(char);   // Write Character To LCD At Current Position
void LCD_Write_String(char*); // Write A String To LCD

void LCD_msg(char*, unsigned char, unsigned char);
```

Para un mejor manejo y a fin de evitar código repetitivo, se define la función `LCD_msg(char*, unsigned char, unsigned char)` para resumir las funciones de `LCD_Set_Cursor(unsigned char, unsigned char)` y `LCD_Write_String(char*)`.

Este nueva función pide como argumento la cadena de caracteres a mostrar y la posición de tal cadena de caracteres en el LCD.

Etapas de montaje

El proyecto puede montarse fácilmente en una placa experimental o protoboard, y como se muestra en la figura N° 22 las conexiones que hay en dicha placa son para alimentar a los diferentes elementos del sistema como los sensores y los motores.

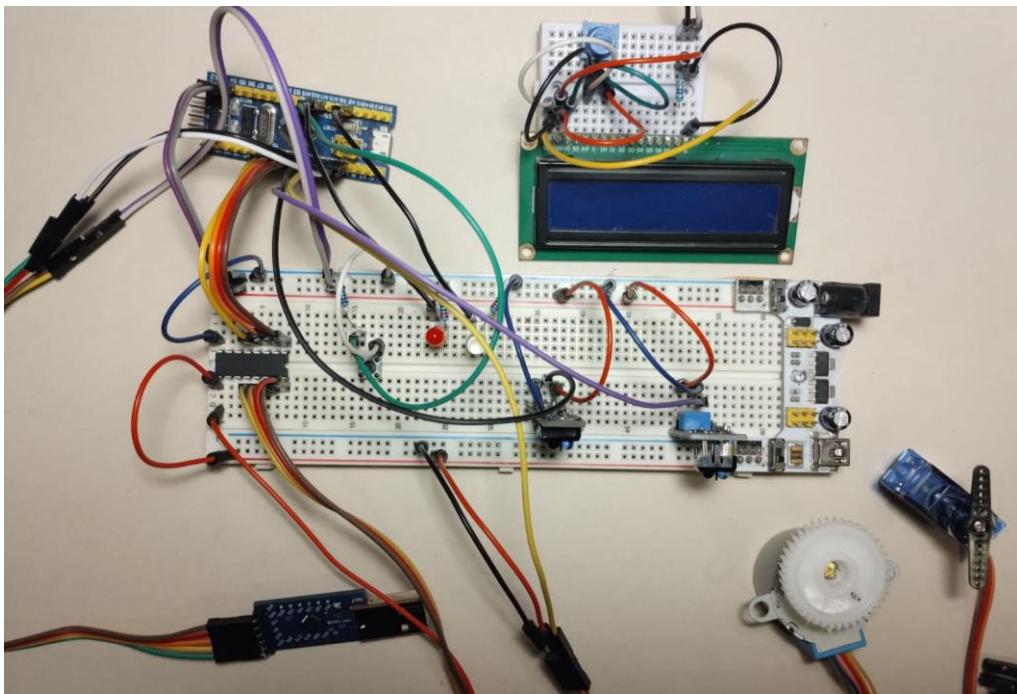


Figura N° 22

El circuito se alimenta con un transformador que cuenta con 5V y 1A de salida, y con ello es suficiente para que todo el sistema funcione correctamente.

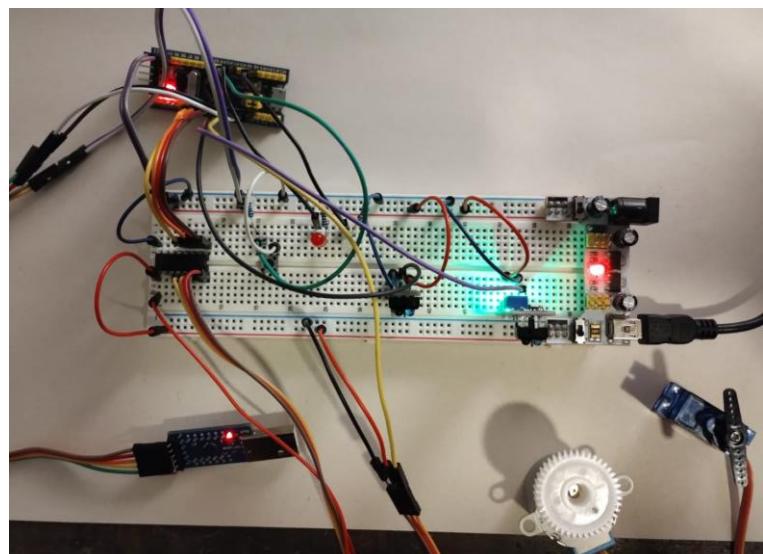


Figura N° 23: En la figura no se ha incluido el LCD, aun así, todo el sistema funciona con el transformador de 5V y 1A .

En la figura Nº 24 se verifica que el primer sensor detecta colores blancos, pero no detecta un objeto con color negro.

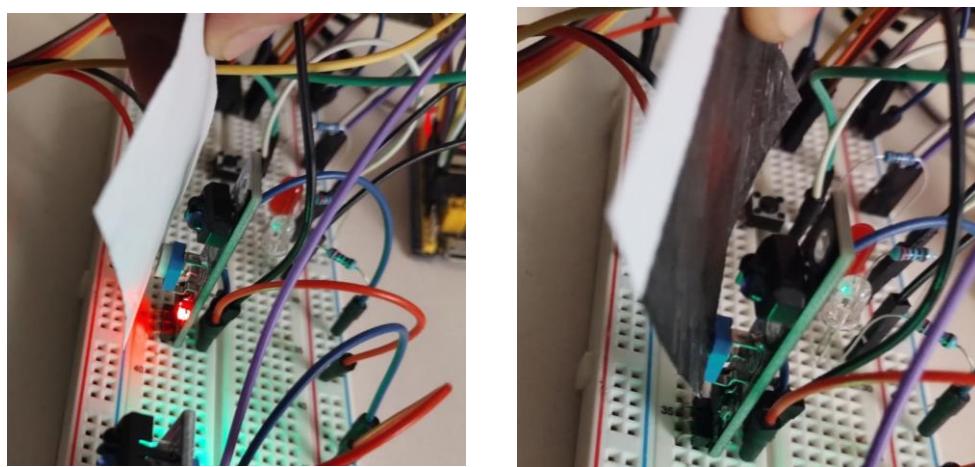
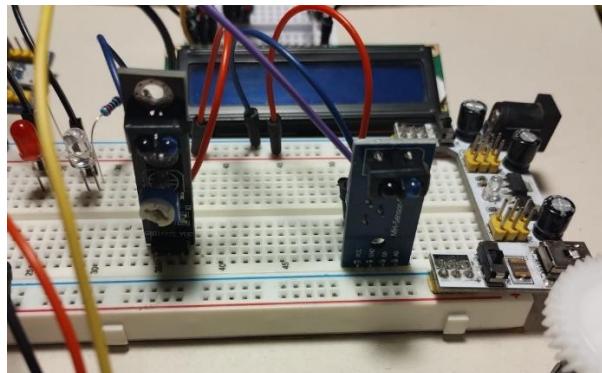
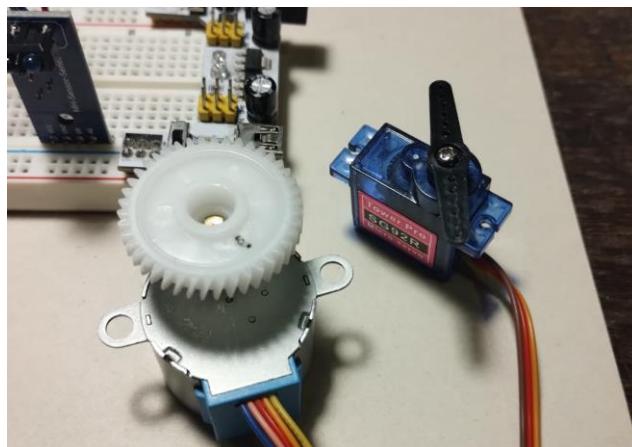


Figura Nº 24: Sensores colocados en la protoboard, siendo el primer sensor el que está ubicado a la izquierda y el segundo sensor el ubicado a la derecha. Se comprueba en las imágenes inferiores que el sensor detecta (con un led rojo) el papel blanco, pero no detecta al papel pintado de negro.

En el montaje realizado a la fecha no se usa la transmisión comentada para lograr una mayor velocidad, sin embargo, se usa un engranaje colocado en el eje para visualizar el inicio del motor, el cambio de velocidad y el frenado de este.



De forma similar, el servomotor no cuenta con una barrea colocada, si no que trae acoplado un plástico alargado que nos facilita conocer la posición que tendría la barrera en su lugar.

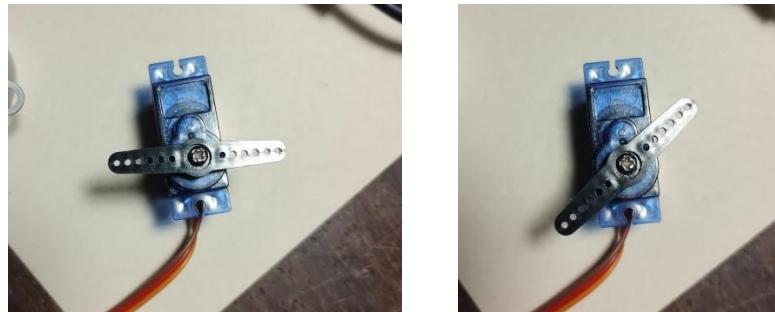
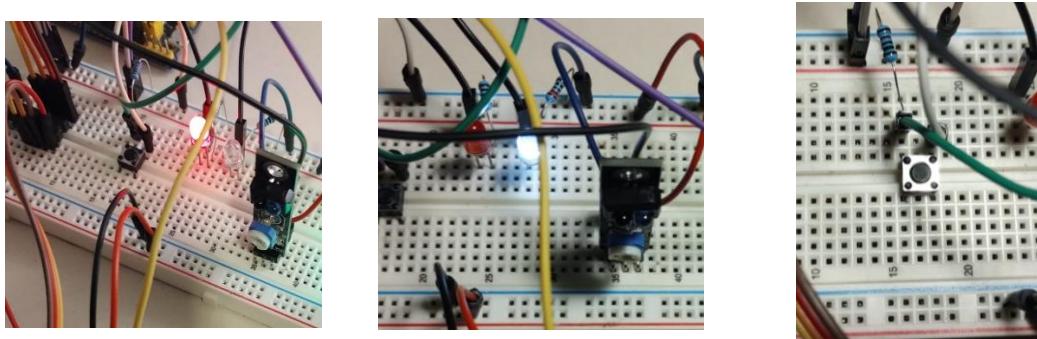


Figura N° 25: Posiciones que adoptara el servomotor. A la izquierda cuando se encuentra la barrera sin desviar, y a la derecha cuando se encuentra en posición de desviar.

En la protoboard se pueden encontrar implantados los leds para el proceso y para la alerta y el pulsador de parada.



Finalmente, un elemento importante en la desviación de las cajas es el conteo de estas, y para ello se implementa el display LCD que nos dará dicha información sin tener que conectarnos por puerto serie al microcontrolador.

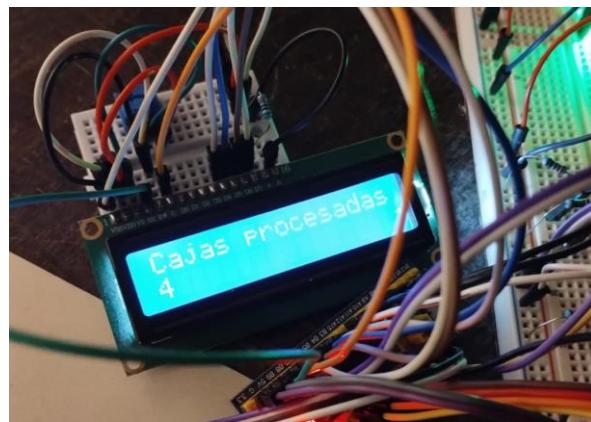


Figura N° 26: El LCD se ha instalado en otra protoboard más pequeña con el fin de aliviar las conexiones en la protoboard principal.

Ensayo de ingeniería

Si se requiere llevar el proyecto a una solución industrial o real, es necesario realizar algunos cambios importantes respecto del prototipo desarrollado en este proyecto.

Los elementos para revisar serán

Motores

El motor paso a paso implementado ha sido una buena opción para el desarrollo y ha ayudado a resolver el problema del tiempo que pasa la caja entre sensor y sensor y en qué momento debe ser desviado. Sin embargo, no se han contemplado el torque que debe ejercer para mover cargas reales. En una aplicación real se recomienda el uso de un motor de CC con encoders integrados y ajustados a las posibles variaciones de carga (torque variable) que haya en la cinta.

Para el segundo motor, el motor de la barrera, se ofrecen dos soluciones, una primer solución consta de un motor de CC servocontrolado y para la segunda solución se propone un accionamiento neumático.

La primera opción aumenta el coste del sistema pero es efectivo y en caso de mejorar al sistema, esta opción es fácilmente configurable.

El accionamiento neumático es una opción más barata, pero requiere de la instalación del sistema neumático para el propio accionamiento. Además, el cierre y apertura de la barrera mediante este accionamiento es menos configurable.

Sensores

Para un caso real, donde las cajas suponen equipaje, contamos con una gran variedad de colores y formas, sumado a que es la etiqueta del equipaje la que nos indica el avión que deben abordar. Entonces para detectar estos detalles se recomienda el uso de una cámara que mediante el uso de visión artificial podemos usar para detectar y clasificar los distintos tipos de equipaje.

Display

El uso del display LCD en este proyecto es meramente informativo, en un caso real no es necesario mostrar en un LCD el conteo de las cajas. Sin embargo, con lo que si debería contar el sistema es con un SCADA, el cual mediante módulos de comunicación Modbus o CAN (sustituyendo a la comunicación serie mediante TTL a USB) nos informa en todo momento del estado del sistema.

Se suma a esto que el sistema es un solo derivador, pero en un caso real se cuenta con una mayor cantidad de derivaciones y por lo tanto diferentes cintas y barreras. Es por ello que con el SCADA tendríamos acceso al estado y la manipulación de todos los dispositivos.

Conclusiones

Inicialmente el proyecto contaba con un motor de corriente continua con escobillas (BDC) y a través de un encoder óptico se pensaba medir su velocidad. Si alguna perturbación ralentizaba la cinta, se media la velocidad y se podía calcular en qué momento debería posicionarse la barrera.

Este caso inicial presentó varias complicaciones, sobre todo al momento del montaje, ya que, al no contar con un torque preciso sobre el eje del motor, la velocidad se media con el motor en vacío (sin carga que mover) y tal velocidad era muy alta, lo que causaba mediciones poco precisas.

Para resolver este problema se cambió el motor de corriente continua por un motor paso a paso, facilitando gran parte del trabajo ya que ahora el control de la velocidad de la cinta puede realizarse mediante los pulsos que alimentan las bobinas del motor paso a paso.

Otro elemento importante que no se consideró al iniciar el proyecto fue la implementación de la comunicación serie con el microcontrolador. Gracias a la idea de poder comunicar al usuario con el sistema se abrió un abanico de posibles interacciones o necesidades del usuario como el cambio de velocidad o el cambio en la posición inicial de la barrera.

Aunque el proyecto alcanzó los objetivos y expectativas propuestos, se plantean algunas mejoras para aplicar a futuro

- › Montaje de una cinta para comprobar que los cálculos realizados para la relación pulsos/metro son correctos.
- › Mejorar el programa aplicando funciones que resuman ciertas porciones de código como la selección del usuario y las decisiones a tomar cada vez que un sensor detecta una caja.
- › Mejorar el menú de selección para que el usuario pueda inicialmente configurar la distancia entre sensores o una velocidad de cinta diferente a las ofrecidas.
- › Usar un sensor RGB que detecte diferentes tipos de color en las cajas, de esta forma el usuario podría elegir qué tipo de cajas quiere derivar.
- › Acceder a la memoria flash para guardar el conteo de cajas y permitir al usuario reiniciar esta cantidad por consola.

Referencias

<https://lastminuteengineers.com/28byj48-stepper-motor-arduino-tutorial/>

<https://www.st.com/en/microcontrollers-microprocessors/stm32f1-series.html>

<https://deepbluembedded.com/>

https://github.com/Khaled-Magdy-DeepBlue/STM32_Course_DeepBlue