TRABAJO SED: MICROS

Raquel García Franco
Irene Álvarez Pérez
Lis Fortea Muñoz

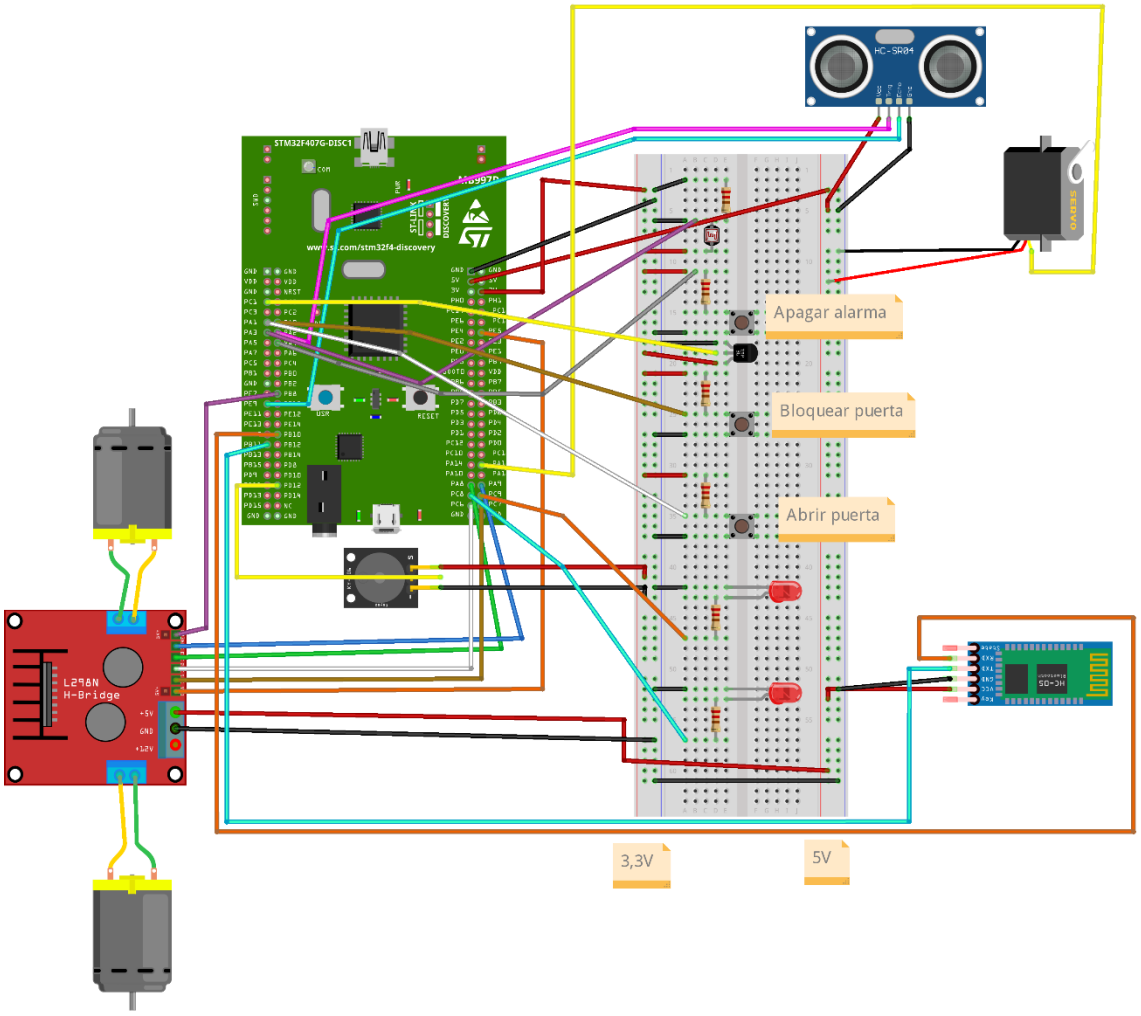
ÍNDICE

1. INTRODUCCIÓN

El presente trabajo trata del control de algunos aspectos propios de una casa, mediante el microcontrolador STM32F407. Para ello se ha realizado una maqueta a escala, y se ha implementado el control de varios aspectos de esta, como son el control de una puerta o de una alarma. Más adelante se detallarán todos los controles implementados. También se ha realizado una aplicación para móvil, desde la cual se pueden controlar todos los aspectos de la casa de una manera cómoda. En el Anexo 1 se puede observar el desarrollo de esta.

2. DIAGRAMAS

A continuación, se muestra un esquema de las conexiones a la placa de todos los sensores y actuadores utilizados, realizada con Fritzing:



3. DESCRIPCIÓN DE COMPONENTES

- STM32F407G-DISC1

El STM32F407G-DISC1 es el microcontrolador con el que se ha realizado el trabajo. Es un microcontrolador de 32 bits con núcleo ARM Cortex-M4. Algunas de sus características principales son las siguientes:

- 32 bits
- Núcleo ARM Cortex-M4
- 1 MB de memoria FLASH
- 192 KB de memoria RAM
- USB OTG FS
- Acelerómetro LI3DSH de 3 ejes
- Micrófono MEMS digital omnidireccional
- DAC
- Botones de usuario y reset
- 8 LEDs
- 3 conectores
- Depurador ST-LINK / V2-A integrado

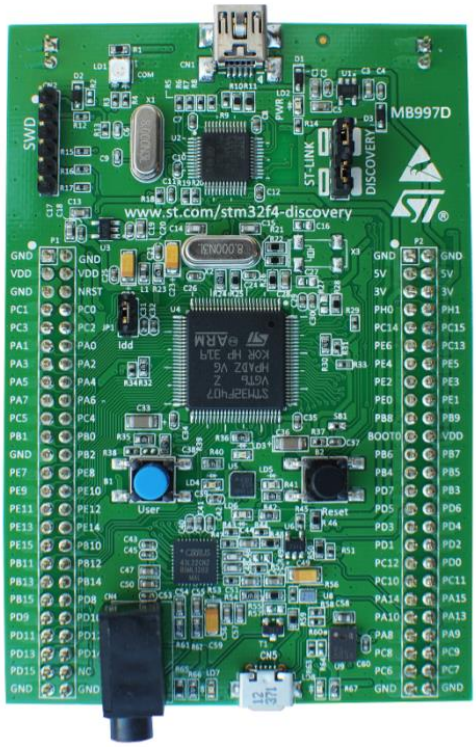


FIGURA 1 – STM32F407VG-DISC1

• Servomotor Datan S1213

Utilizamos este servomotor para controlar la apertura y cierre de la puerta de la casa, ya que cuenta con potencia suficiente para abrir la puerta.

FIGURA 2 - Servomotor

Especificaciones técnicas:

Voltaje (V)	4,5
Velocidad	0,21 s/60º
Par (kg · cm)	6,5
Ángulo de rotación	180º
Peso (g)	46

- ## Motor DC con reductor

Para el control de las persianas, hemos decidido utilizar un motor de DC con reductora. Hemos elegido este modelo porque presenta par suficiente para la tarea, a la vez que presenta un tamaño muy reducido.

FIGURA 3 – Motor DC con reductor

Especificaciones técnicas:

Voltaje (V)	6
Velocidad	10 RPM
Par (kg · cm)	5
Relación máxima de engranajes	1:1000
Corriente sin carga (mA)	15

- ## LDR

Fotorresistor sensible a la luz que emite una salida analógica proporcional al nivel de luminosidad ambiente.

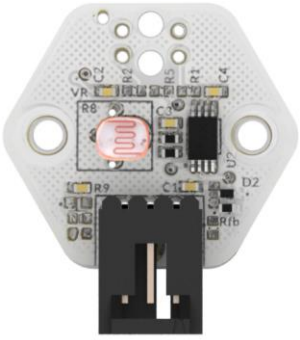


FIGURA 4 – LDR

- ## Zumbador Pasivo

Zumbador pasivo que emite un sonido u otro según la frecuencia que se le ordene reproducir. Convierte la señal eléctrica en una onda sonora.

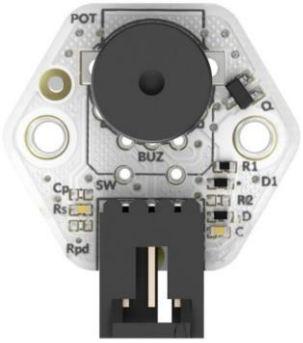


FIGURA 5 - Zumbador

- ## Ultrasonidos HC-SR04

Sensor de ultrasonidos, cuyo principio se basa en la emisión de una onda y el cálculo del tiempo que tarda en volver al sensor tras reflejarse en un objeto.

FIGURA 6 – HC-SR04

Especificaciones técnicas:

Voltaje (V)	5
Corriente (mA)	15
Frecuencia (Hz)	40
Rango máximo/mínimo (cm)	400/2
Ángulo de medida (º)	15

- ## LED

Led integrado en una placa que incluye ya una resistencia para su protección.

FIGURA 7 – LED

- ## HC 05 ZS – 040

Hemos decidido implementar una aplicación bluetooth para poder controlar todo el funcionamiento de la casa desde el móvil.

FIGURA 8 – Módulo Bluetooth HC-06

Especificaciones técnicas:

Frecuencia	Banda ISM de 2,4 GHz
Modulación	GFSK
Protocolo USB	USB v1.1/2.0
Potencia de transmisión	< 4dBm, Clase 2

- ## L298N

Para controlar los motores de continua, se ha decidido utilizar el controlador L298N, ya que ya disponíamos de uno, y es adecuando para el control del motor que utilizamos.

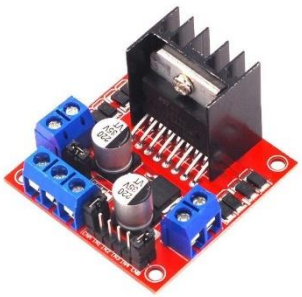


FIGURA 9 – Puente H L298

Especificaciones técnicas:

Voltaje operativo mínimo (V)	5
Voltaje operativo máximo (V)	35
Voltaje lógico (V)	5
Máxima corriente (A)	2

4. EXPLICACIÓN DEL FUNCIONAMIENTO

Nuestra casa domótica cuenta con distintas funcionalidades:

- Sistema de control de la iluminación, que cuenta con un LDR que mide la luminosidad del ambiente. Si es de noche o si el usuario lo solicita a través de la aplicación, las persianas se bajan. La subida de dichas persianas se realiza cuando el usuario lo solicita.
- Sistema de control de la temperatura. Cuando la temperatura de la estancia baja de 20ºC, el ventilador se acciona en modo calor. En cambio, se la temperatura es superior a 25ºC, se acciona en modo frío. El usuario puede solicitar ambas acciones en la aplicación y, adicionalmente, puede parar el ventilador.
- Control de la puerta de la vivienda. Cuenta con dos botones, uno para bloquear el sistema y otro para abrir y cerrar la puerta si el sistema no está bloqueado. Además, si han pasado 10 segundos desde que está abierta la puerta, dicha puerta se cierra y bloquea automáticamente.
- Sistema de alarma. La alarma está conectada al ultrasonidos, de forma que si se detecta presencia a menos de 10 cm, suena la alarma. Esta alarma puede desconectarse con un botón habilitado para tal efecto o desde la aplicación.

5. EXPLICACIÓN DE ALGUNAS FUNCIONES IMPORTANTES

6.1 Alarma

Para el control de la alarma se ha usado un ultrasonidos y un zumbador, aparte del bluetooth usado para la comunicación con la app.

```
void alarma(void){ //Función completa de la alarma
      HCSR04_Read(); //Leemos el valor del ultrasonidos
      HAL_Delay(100);
      if(Distance<10){ //Si la distancia es menor de 10 cm
```

```
                tiempo_alarma=HAL_GetTick(); //tomamos el tiempo en ese
instante
                htim4.Instance->CCR1=zumb; //encendemos el zumbador
                sonando=1;
        }
        if(sonando==1){ //si está sonando
                if(HAL_GetTick()-
tiempo_alarma>5000||(debouncer2(&boton2,GPIOA,GPIO_PIN_4))==1||readBuf
[0]==49){ //si pasan 5 s, pulso el botón, o lo pido desde la app la
desactivo
                        htim4.Instance->CCR1=0; //paro el zumbador
                        tiempo_alarma=0; //reseteo tiempos y flags
                        sonando=0;
                        readBuf[0]=0;
                }
        }
}
```

Como vemos, en esta función llamamos a la función HCSR04_Read(), que es la encargada de generar el pulso en el trigger del ultrasonidos y habilitar las interrupciones del echo, para esperar a la recepción de la señal por parte de este, y así calcular la distancia.

```
void HCSR04_Read (void) //Función de lectura del ultrasonidos
{
        //enviamos un pulso en el pin TRIG
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, GPIO_PIN_SET); // ponemos
el pin TRIG on
        delay(10);  // esperamos 10 us
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, GPIO_PIN_RESET);// ponemos
el pin TRIG off

        __HAL_TIM_ENABLE_IT(&htim1, TIM_IT_CC1); //Habilitamos las
interrupciones para esperar a la recepción
}
```

Este cálculo se realiza en el callback correspondiente al temporizador `HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)`. En este, se comprueba que la recepción de la interrupción se produzca en el canal 1, y se lee el valor recibido por ella. Entonces, se pone a 1 un flag que indica que ya se ha leído el primer valor, y se pone el modo de la detección de interrupción con polaridad negativa mediante `__HAL_TIM_SET_CAPTUREPOLARITY(htim, TIM_CHANNEL_1, TIM_INPUTCHANNELPOLARITY_FALLING);`.
Tras esto, comprobamos que ya se ha tomado la primera medida y leemos la segunda. Entonces calculamos la diferencia entre ambas. Al tener ya el tiempo medido entre el momento de envío de la señal y el de recepción, ya podemos calcular la distancia al objeto.
Para calcular esta distancia, simplemente multiplicamos la mitad de este tiempo por la velocidad del sonido, 343 m/s, y ya tenemos la distancia al objeto. Una vez tenemos esto calculado, reseteamos todas las flag y contadores y volvemos a poner el detector con polaridad positiva, y desactivamos las interrupciones.

Una vez que tenemos la distancia calculada, simplemente comprobamos si es menor de 10 cm, y entonces guardamos el tiempo actual y encendemos el zumbador.

En caso de que la alarma esté sonando, esta se puede desactivar mediante un pulsador, con su debouncer correspondiente, mediante la app o al pasar 5 segundos, que se desactiva automáticamente, y se reinician todas las variables y flags.

6.2 Puerta

La función de control de la puerta puede verse a continuación:

```
void puerta(void)  //PUERTA
{

      if((debouncer2(&boton3,GPIOA,GPIO_PIN_0))==1||readBuf[0]==50)
//si pulsamos el botón de desbloqueo o mandamos la orden desde la
aplicación
      {
               if(bloqueo==1 && abierto==0) //en caso de que la puerta
este bloqueada y cerrada
               {
                     bloqueo=0; //se desbloquea
                }
               else if (bloqueo==0 && abierto==0) //si esta cerrada y
desbloqueada
               {
                     bloqueo=1; //se bloquea
               }
                readBuf[0]=0; //pongo a cero la variable que recibe el
valor del bluetooth
      }


      if ((debouncer2(&boton4,GPIOA,GPIO_PIN_1))==1||readBuf[0]==51)
//si pulso el botón de apertura o mando la orden desde la app
      {
               if(abierto==1) //si está abierta
               {
                     //abierto=0;
                     espera_puerta=0;//pongo el tiempo de espera a 0
                     cerrando=1; //activo el flag que indica que voy a
cerrar la puerta
               }
               else
              {
                    //abierto=1;
                    abriendo=1; //activo el flag que indica que voy a
abrir la puerta

              }
                    readBuf[0]=0; //pongo a cero la variable que recibe
el valor del bluetooth
      }

      if(abierto==0 && bloqueo==0 && abriendo==1) //Si está cerrada,
no bloqueada y el flag de apertura activado
      {
               servo(&htim2, 0); //pongo el servo a cero
grados(posición de la puerta abierta)
               abierto=1; //indico que ya está abierta la puerta
               espera_puerta = HAL_GetTick(); //tomo el tiempo actual
```

```
                abriendo=0; //pongo el flag de apertura a 0
        }
        if(HAL_GetTick()-espera_puerta > 10000  &&  abierto==1 &&
cerrando==0) //si han pasado 10s y está abierta, la cierro y la
bloqueo
         {
//            bloqueo=1;
              espera_puerta=0; //reseteo el tiempo de espera
              cerrando = 1; //indico que quiero cerrar la puerta
         }

        if(abierto==1 && bloqueo==0 && cerrando==1) //Si está abierta,
no bloqueada y quiero cerrarla
        {
              servo(&htim2, 90);//ordeno al servo la posición de la
        puerta cerrada
              abierto=0; //indico que está cerrada
              espera_puerta = 0; //reseteo el tiempo
              cerrando=0; //pongo el flag de ciere a 0
              bloqueo=1; //bloqueo la puerta
        }
              HAL_GPIO_WritePin(GPIOC, GPIO_PIN_9, bloqueo); //control
de la luz. ENCENDIDA->Bloqueada
}
```

Como vemos, en primer lugar, se evalúa el aso de que se pulse el botón de bloqueo/desbloqueo o se mande dicha orden desde la app. En ese caso, si está bloqueada, se desbloquea, y si esta desbloqueada, se bloquea.
A continuación, se comprueba la pulsación del botón de apertura/cierre de la puerta, o la orden homóloga desde la app. En caso de que la puerta esté abierta, se resetea el tiempo de espera de la puerta, y se activa el flag que indica que se está cerrando. En caso contrario, que esté cerrada, se activa el flag que indica que está abriéndose.
Ahora se analizan las diferentes casuísticas. En primer lugar, si la puerta está cerrada, desbloqueada y se pide la apertura, se mueve el servo a la posición de la puerta abierta y se almacena el tiempo actual en una variable. También se activa un flag que indica que la puerta está abierta.
Después, se evalúa el caso en el que hayan pasado más de 10 segundos y la puerta esté abierta sin que nadie haya pedido el cierre de esta. Entonces se activa el cierre automático.
El último caso evaluado, consiste en que la puerta esté abierta, desbloqueada y se pida el cierre. Entonces se manda la posición de cierre al servo, se resetean todos los flags y se bloquea la puerta.
Lo último que tiene esta función es el control del led que marca si la puerta está o no bloqueada.
Como vemos, la puerta puede controlarse desde la app o desde el pulsador habilitado a tal efecto. En primer lugar, tenemos un botón que bloquea y desbloquea la puerta, ya que si esta no está desbloqueada no puede moverse. En la app también existe un botón dedicado a esto. Luego tenemos otro que la abre o cierra, en función de la posición en la que esté actualmente. Si abrimos la puerta, y no se pulsa el botón para cerrarla, esta se cierra a los 10 s.

6.3 LDR

La función de control del LDR se puede ver a continuación:

```
void LDR(void) //función de lectura del LDR
{
	HAL_ADC_Start_IT(&hadc1);
	if(LDR_val<60) //en caso de que el valor sea menor a 60 (luz
	ambiente)
		HAL_GPIO_WritePin(GPIOC, GPIO_PIN_8,1); //Encendemos la
		luz
	else
		HAL_GPIO_WritePin(GPIOC, GPIO_PIN_8,0); //Apagamos la luz
}
```

Como vemos, en primer lugar, se inicia el convertidor ADC en modo interrupciones. Luego, simplemente comprobamos si el valor medido es menor de 60 para encender las luces. El callback correspondiente, que asigna los valores a la variable LDR_val es el siguiente:

```
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc){
	if (hadc->Instance==ADC1)
		LDR_val=HAL_ADC_GetValue(&hadc1); //guardamos el valor
medido en LDR_val
	if (hadc->Instance==ADC2)
		sensorTemp_val=HAL_ADC_GetValue(&hadc2);
}
```

6.4 Persianas

Para el control de las persianas, se han implementado tres funciones de control del motor de continua, una para subir la persiana, otra para bajarla y otra para parar el motor.

```
void subePersiana(int s) //Función de subida de la persiana
{
	//TIM9->CCR1=s;
	__HAL_TIM_SET_COMPARE(&htim9, TIM_CHANNEL_1, s);
	HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6,GPIO_PIN_SET); //Giro
horario
	HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7,GPIO_PIN_RESET);
}

void bajaPersiana(int s) //Función para bajar la persiana
{
	//TIM9->CCR1=s;
	__HAL_TIM_SET_COMPARE(&htim9, TIM_CHANNEL_1, s);
	HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6,GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7,GPIO_PIN_SET); //Giro
antihorario
}

void pareMotor() //Función que para el motor
{
	//TIM9->CCR1=s;
	__HAL_TIM_SET_COMPARE(&htim9, TIM_CHANNEL_1, 0);
	HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6,GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7,GPIO_PIN_RESET);
}
```

El código que controla las persianas es el siguiente:

```
void persianas(){ //Función del control completo de la persiana

if(bajada==0){ //Si está subida
	if(bajando==0){ //Y no se está subiendo
		if(readBuf[0]==52||LDR_val<60){ //Si detecta que pido
	desde la aplicación que baje
			bajaPersiana(5000); //bajo la persiana
			tiempo_motor=HAL_GetTick(); //cojo el tiempo
			bajando=1; //pongo el flag de subiendo a 1
		}
	}
	else if(bajando==1){
		if(HAL_GetTick()-tiempo_motor>tiempo_persiana){ //si ya
	ha llegado abajo la persiana
		pareMotor(); //paro el motor
		bajando=0; //pongo el flag de bajando a 0
		bajada=1;   //declaro que ya esta bajada
		readBuf[0]=0; //reseteo la variable de recepción del
	bluetooth
		}
	}
}
else if(bajada==1){ //si esta bajada
	if(subiendo==0){ //no se está subiendo aun
		if(readBuf[0]==52){  //las persianas se suben al pedirlo
	desde el movil o al bajar la
			subePersiana(5000);              // luminosidad
	(hacerse de noche)
			tiempo_motor=HAL_GetTick();
			subiendo=1; //activo el flag de subiendo
		}
	}
	else if(subiendo==1){ //si está subiendo
		if(HAL_GetTick()-tiempo_motor>tiempo_persiana){ //y ha
		acabado de subir
		pareMotor(); //paro el motor
		subiendo=0; //reseteo flags
		bajada=0;
		readBuf[0]=0;
		}
	}
}
}
```

5.5 Ventilador

La función que controla el ventilador es la siguiente:

```
void ventilador(){

	//if(readBuf[0]==53||sensorTemp_val<30){//el ventilador da calor
si se pide desde el movil o al subir la temperatura
	if (readBuf[0]==55){ //parar motor desde aplicación
		pararMovimiento();
	}
	else if(readBuf[0]==53 || sensorTemp_val<30){
			movimientoCalor(5000);   // por debajo de 20 ºC
				//tiempo_motor_ventilador=HAL_GetTick();
```

```
                        dando_calor=1;

                }

        else if(readBuf[0]==54 || sensorTemp_val>190){  //el ventilador
da frio si se pide desde el movil o al subir la temperatura
                        movimientoFrio(5000);                   //
por encima de 25 ºC
                        //tiempo_motor_ventilador=HAL_GetTick();
                        dando_frio=1;

                }
}
```

Las funciones movimientoFrio, movimientoCalor y pararMovimiento son análogas a las de la persiana.
También se utiliza para el control del ventilador una función que lee el valor del sensor de temperatura:

```
void temperatura(void) //Función para leer la temperatura
{
        HAL_ADC_Start_IT(&hadc2);
        if(sensorTemp_val>10) //si la temperatura es mayor de 10 grados
                HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14,1);
        else
                HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14,0);
        }
```

5.6 Debouncer

Implementamos también una función para evitar los rebotes en los pulsadores, y así evitar falsas pulsaciones. La función es la siguiente:

```
int debouncer2(volatile int* button_int, GPIO_TypeDef* GPIO_port,
uint16_t GPIO_number){
        static uint8_t cuenta_boton=0;
        static int cuenta=0;

        if (*button_int==1){
                if (cuenta_boton==0) {
                        cuenta=HAL_GetTick();
                        cuenta_boton++;
                }
                if (HAL_GetTick()-cuenta>=20){
                        cuenta=HAL_GetTick();
                        if (HAL_GPIO_ReadPin(GPIO_port, GPIO_number)!=1){
                                cuenta_boton=1;
                        }
                        else{
                                cuenta_boton++;
                        }
                        if (cuenta_boton==3){ //Periodo antirebotes
                                cuenta_boton=0;
                                *button_int=0;
                                return 1;
                        }
                }
```

```
	}
	return 0;
	}
```

En ella, si se detecta la pulsación del botón, guardamos el tiempo actual en la variable cuenta, y aumentamos el valor de la variable cuenta_boton. Si han pasado más de 20 ms desde que se detectó la pulsación, se vuelve a actualizar el tiempo actual en la variable cuenta. Si la variable cuenta_boton vale más de 1, es decir se ha detectado más de una pulsación en los 20 ms, cuenta_boton vuelve a valer 1. En caso contrario se aumenta el valor de cuenta_boton. Cuando esta variable valga 3, que es el periodo considerado, se reinician todas las variables y la función devuelve un 1.
Para complementar el control de los botones, estos se controlan por interrupciones, por lo que tenemos el correspondiente callback, que activa el flag del botón pulsado.

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) //Callback de los
botone
{
	if(GPIO_Pin == GPIO_PIN_0)//Botón de bloqueo
	{
		boton3=1;
	}
	if(GPIO_Pin == GPIO_PIN_1)//Botón de apertura de la puerta
	{
		boton4=1;
	}
	if(GPIO_Pin == GPIO_PIN_4)//Bloqueo de desactivación de la
alarma
	{
		boton2=1;
	}
}
```

5.7 Función servo

Hemos implementado una función para el control de los servos, cuyo código se puede ver a continuación:

```
void servo(TIM_HandleTypeDef* htim, int grados){
 const int MAX=20;
 float ms= grados/90.0f +0.5f;
 float ciclo = ms/(float)MAX;
 mov =htim->Instance->ARR*ciclo;
 htim->Instance->CCR1 = mov;
}
```

A esta función, le entran por parámetros en temporizador asociado al servo y los grados que se quiere mover.
Una vez introducidos estos datos, la función calcula la ecuación de la recta para asociar los grados que quiero moverlo a los ms que necesito que se esté moviendo. Después calcula el porcentaje del ciclo que está encendido, y lo multiplica por ARR, que es propio del tim. Ese valor es el que finalmente asociamos al CCR1 del tim.

5.8 Bluetooth

Hemos implementado una comunicación por bluetooth para poder controlar la casa desde una aplicación móvil diseñada a tal efecto. El control del bluetooth

desde el código es muy sencillo. Simplemente definimos el callback correspondiente:

```
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart) { //callback
para el bluetooth
 /* Se recibe el caracter y se pide el siguiente*/
if(huart->Instance==huart3.Instance)
 HAL_UART_Receive_IT(&huart3, (uint8_t*)readBuf, 1);
}
```

En este, en caso de detectar algo por el canal correspondiente al bluetooth, activamos el UART para que reciba y almacenamos el carácter en la variable readBuf. En el código principal simplemente tenemos que iniciar el USART y ponerlo en modo recepción:

```
MX_USART3_UART_Init();
HAL_UART_Receive_IT(&huart3, (uint8_t*)readBuf, 1);
```

Con esto, ya podemos utilizar el valor recibido y almacenado en readBuf para controlar aspectos de la casa.

6. BIBLIOGRAFÍA

Bitbucket Alberto Brunete:

https://bitbucket.org/abrunete/sistemas-electronicos-digitales/src/master/

Motor corriente continua:

https://programmerclick.com/article/578631072

Anexo 1: App móvil

Para mejorar el control de la casa, hemos diseñado una aplicación móvil muy sencilla, mediante MIT App Inventor, que nos permite controlar los aspectos básicos de la casa. La interfaz de la aplicación se puede ver en la figura siguiente:

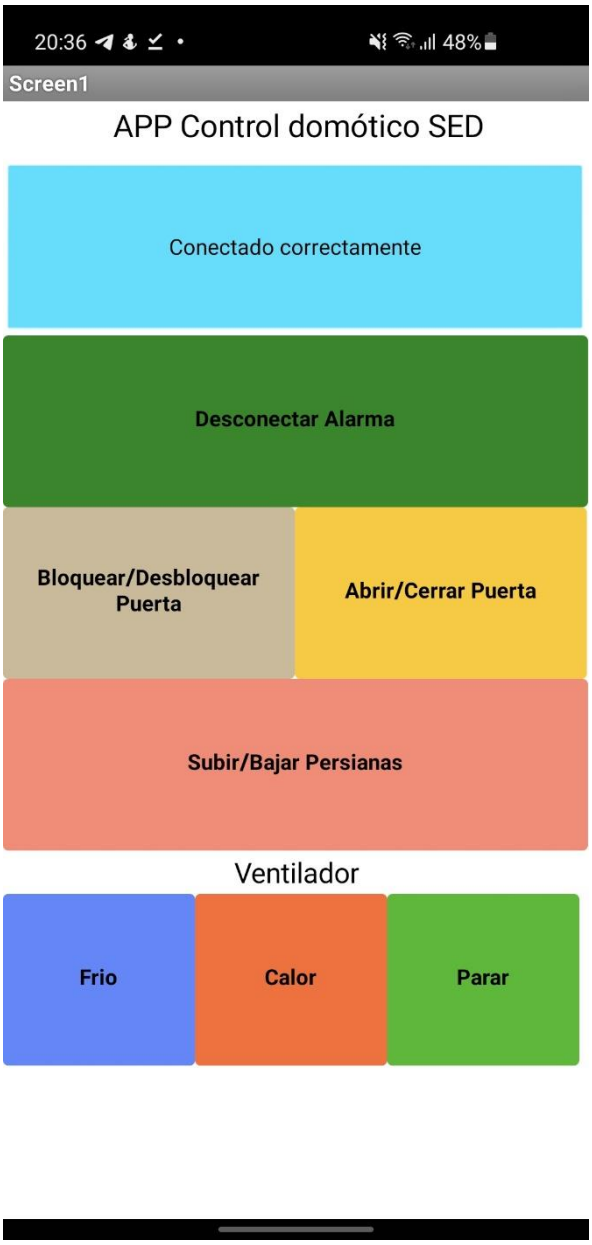


FIGURA 10 – Interfaz App

El código programado para controlar el bluetooth se muestra a continuación:

FIGURA 11 – Código App

Como podemos ver, MIT App Inventor se programa en Scratch, lo que facilita enormemente la programación. Podemos ver en la figura superior, como hay un primer bloque de conexión con el bluetooth, el cual hace que se nos muestren los bluetooth disponibles. A continuación, ese dispositivo se conecta, y si todo va bien se muestra el mensaje “Conectado correctamente”.
El resto de los bloques simplemente detectan la pulsación de alguno de los botones y envían al bluetooth el número correspondiente, que luego se interpretará de una forma u otra en el código principal.