ANEXO A

- **Tabla de direcciones X10**

Tabla de direcciones de Casa

Dirección de casa	Código de casa			
	H1	H2	H3	H4
A	0	1	1	0
B	1	1	1	0
C	0	0	1	0
D	1	0	1	0
E	0	0	0	1
F	1	0	0	1
G	0	1	0	1
H	1	1	0	1
I	0	1	1	1
J	1	1	1	1
K	0	0	1	1
L	1	0	1	1
M	0	0	0	0
N	1	0	0	0
O	0	1	0	0
P	1	1	0	0

Tabla de direcciones de Unidad y Función

Dirección de Unidad		Código de control				Sufijo
		D1	D2	D4	D8	D16
1		0	1	1	0	0
2		1	1	1	0	0
3		0	0	1	0	0
4		1	0	1	0	0
5		0	0	0	1	0
6		1	0	0	1	0
7		0	1	0	1	0
8		1	1	0	1	0
9		0	1	1	1	0
10		1	1	1	1	0
11		0	0	1	1	0
12		1	0	1	1	0
13		0	0	0	0	0
14		1	0	0	0	0
15		0	1	0	0	0
16		1	1	0	0	0
Código de Comandos	Apagar todas las Unidades	0	0	0	0	1
	Encender Todas las Luces	0	0	0	1	1
	Encender	0	0	1	0	1
	Apagar	0	0	1	1	1
	Atenuar Intensidad	0	1	0	0	1
	Aumentar Intensidad	0	1	0	1	1
	Apagar todas las Luces	0	1	1	0	1
	Código Extendido (4)	0	1	1	1	1
	Petición de Saludo (1)	1	0	0	0	1
	Aceptación de Saludo	1	0	0	1	1
	Atenuación Preestablecida (2)	1	0	1	X	1
	Datos Extendidos (Analógico) (3)	1	1	0	0	1
	Estado = On	1	1	0	1	1
	Estado = Off	1	1	1	0	1
	Petición de Estado	1	1	1	1	1

ANEXO B

- **Direcciones de los módulos de control**

MÓDULO CONTROL DE LUZ									Recibe
Cabecera	**1**		**1**		**1**		**0**		
Casa seleccionada	**0**	1	**1**	0	**1**	0	**0**	1	
Unidad	**0**	1	**1**	0	**1**	0	**0**	1	
ON	**0**	1	**0**	1	**1**	0	**0**	1	a
OFF	**0**	1	**0**	1	**1**	0	**1**	0	b
Atenuar	**0**	1	**1**	0	**0**	1	**0**	1	c
Aumentar	**0**	1	**1**	0	**0**	1	**1**	0	d

MÓDULO CONTROL DE PERSIANAS									Recibe
Cabecera	**1**		**1**		**1**		**0**		
Casa seleccionada	**0**	1	**1**	0	**1**	0	**0**	1	
Unidad	**1**	0	**1**	0	**1**	0	**0**	1	
ON	**0**	1	**0**	1	**1**	0	**0**	1	e
OFF	**0**	1	**0**	1	**1**	0	**1**	0	f

MÓDULO CONTROL DE TOMACORRIENTE									Recibe
Cabecera	**1**		**1**		**1**		**0**		
Casa seleccionada	**0**	1	**1**	0	**1**	0	**0**	1	
Unidad	**0**	1	**0**	1	**1**	0	**0**	1	
ON	**0**	1	**0**	1	**1**	0	**0**	1	g
OFF	**0**	1	**0**	1	**1**	0	**1**	0	h

ANEXO C

- **Programa del Módulo de Control**

  - **Diagrama de Flujo**

Diagrama de Flujo del Programa Principal del Módulo de Control

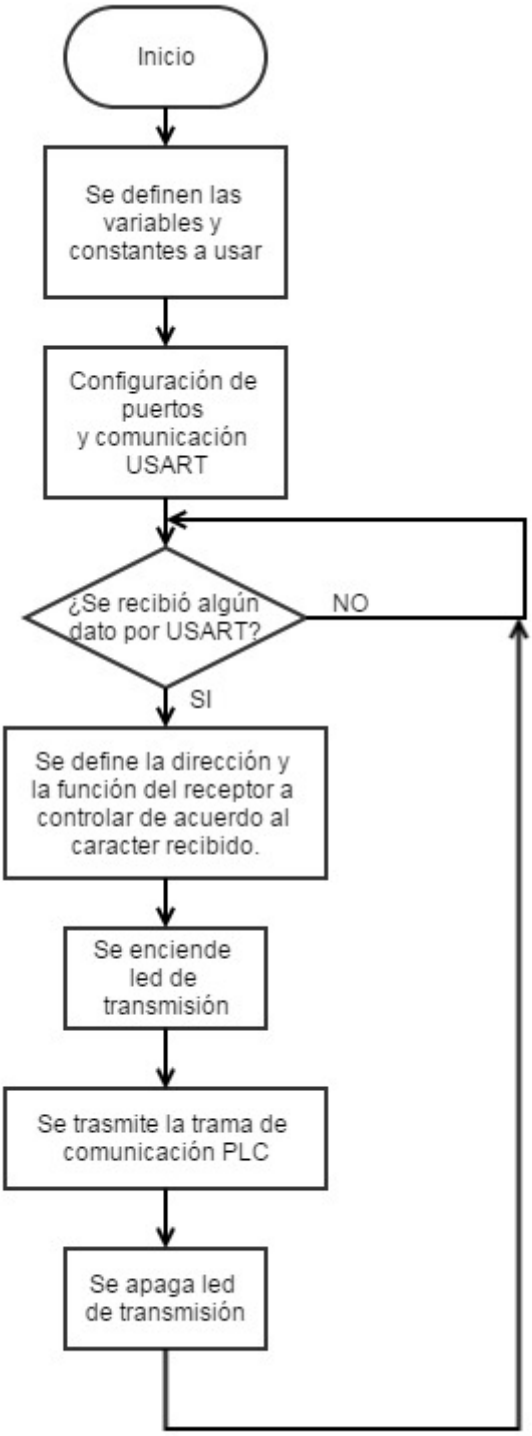


Diagrama de Flujo de Transmisión de Pulsos PLC

- **Código C++**

```
/*
 * Programa Módulo de Control
 *
 *  Author: Miguel Guzmán Guerra - Renzo Burga Velarde
 *
 */

//Se define uso de librerias
#define F_CPU 20000000L
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>
#define high(x)      ((x)>>8)
#define low(x)       ((x) & 0xFF)

//Declaración de funciones
void Config_Puertos(void);
void USART_Config(void);
char USART_Receive(void);
void tx_trama(unsigned char, unsigned char);
void tx_registro(unsigned char, int);

//declaración de constantes y variables
char data;
unsigned char cabecera=0b1110;
unsigned char casa=0b01101001;



int main(void)
{
	Config_Puertos(); // configuración de puertos
	USART_Config();     // configuración de comunicación USART
	unsigned char unidad;
	unsigned char funcion;

	while(1)
	{
	inicio:
		data = USART_Receive(); //recibe la instrucción de la tablet

		//Se asigna dirección y función de acuerdo a caracter recibido
		if (data =='a'){
			unidad=0b01101001; //luz
			funcion=0b01011001; //on
		}
		if (data =='b'){
			unidad=0b01101001; //luz
			funcion=0b01011010; //off
		}
		if (data =='c'){
			unidad=0b01101001; //luz
			funcion=0b01100101; //atenuar
		}
		if (data =='d'){
			unidad=0b01101001; //luz
			funcion=0b01100110; //aumentar
		}
		if (data =='e'){
			unidad=0b10101001; //tomacorrientes
```

```
                        funcion=0b01011001; //on
                }
                if (data =='f'){
                        unidad=0b10101001; //tomacorrientes
                        funcion=0b01011010; //off
                }
                if (data =='g'){
                        unidad=0b01011001; //cortina
                        funcion=0b01011001; //abierta
                }
                if (data =='h'){
                        unidad=0b01011001; //cortina
                        funcion=0b01011010; //cerrada
                }
                if ((data!='a') & (data!='b') & (data!='c') & (data!='d') &
(data!='e') & (data!='f') & (data!='g') & (data!='h'))
                {
                        goto inicio; //no se recibió un valor válido
                }

                PORTD |= (1<<PORTD5); // Led de transmisión encendido
                tx_trama(unidad, funcion); //transmite trama
                PORTD &=~ (1<<PORTD5); //Led de transmisión apagado

                goto inicio;
        }
}
//función de configuración de puertos
//no devuelve valor alguno
//no recive valor alguno
void Config_Puertos(void)
{
        DDRD = (1<<DDD5)|(0<<DDD3)|(1<<DDD2);
        PORTD = 0x00;
}

//función de configuración de USART
//no devuelve valor alguno
//no recive valor alguno
void USART_Config(void)
{
        UBRR0H = high(129); /*configurando el baudrate*/
        UBRR0L = low (129);
        UCSR0B = (1<<TXEN0); /*habilitar RX */
        UCSR0C = (1<<UCSZ01)|(1<<UCSZ00); /*1 bit de parada*/
        UCSR0B = (1<<RXEN0)|(1<<TXEN0); /*habilitar RX */
        UCSR0C = (1<<UCSZ01)|(1<<UCSZ00); /*1 bit de parada*/
}

//función de recpción de caracter por USART
//devuelve el caracter recibido
//no recive valor alguno
char USART_Receive( void )
{
        /* Espera dato recibido */
        while ( !(UCSR0A & (1<<RXC0)) )
        ;
        /* Recibe y retorna el dato */
        return UDR0;
}

//función de transmisión de tramas
```

```
//no devuelve valor alguno
//recibe la dirección de unidad y funcion
void tx_trama(unsigned char unidad, unsigned char funcion)
{
	unsigned char sufijo1=0b00000001;
	unsigned char sufijo2=0b00000010;
	int cont;
	//------------------------------------------------------------------------
------------
	inicio: /* Verifica que no esté en un cruce por cero para iniciar la
transmisión*/
	if (PIND & (1<<PIND3))
	{
		goto inicio; //estamos en un cruce por cero volver a inicio
	}
	//------------------------------------------------------------------------
------------
	cont=4;
	tx_registro(cabecera,cont);
	cont=8;
	tx_registro(casa,cont);
	cont=8;
	tx_registro(unidad,cont);
	cont=2;
	tx_registro(sufijo1,cont);
	cont=4;
	tx_registro(cabecera,cont);
	cont=8;
	tx_registro(casa,cont);
	cont=8;
	tx_registro(funcion,cont);
	cont=2;
	tx_registro(sufijo2,cont);
}

//función de transmisión de una trama
//no devuelve valor alguno
//recibe la trama a transmitir y su longitud en bits
void tx_registro(unsigned char registro, int contador)
{
	int	i;
	unsigned char aux;

	for (i=0;i<contador;i=i+1)
	{
		envia_registro:
		if (!(PIND & (1<<PIND3)))
		{
			goto envia_registro; //aun no hay cruce por cero
		}

		if (PIND & (1<<PIND3))
		{
			//	Hay cruce por cero se transmitira un bit
			aux= registro & (1<<(contador-1)); //	Me quedo con el MSB
de registro

			aux=(aux>>(contador-1));
			if (aux==1)	//	Si el bit recogido es igual a 1 se envia
el pulso
			{
				PORTD |= (1<<PORTD2);
				_delay_ms(2);
```

```
                        PORTD &=~ (1<<PORTD2);
                    }
                    if (aux==0)   //     Si el bit recogido es igual a 0 no se
envia el pulso
                    {
                        PORTD &=~ (1<<PORTD2);
                        _delay_ms(2);
                    }
                    registro=registro<<1; //se corre registro
                    //se retorna a enviar el siguiente bit
                }
            }

}
```

ANEXO D

- **Programa del Módulo de Controlable**

  - **Diagrama de Flujo**

Diagrama de Flujo del Programa Principal del Módulo Controlable

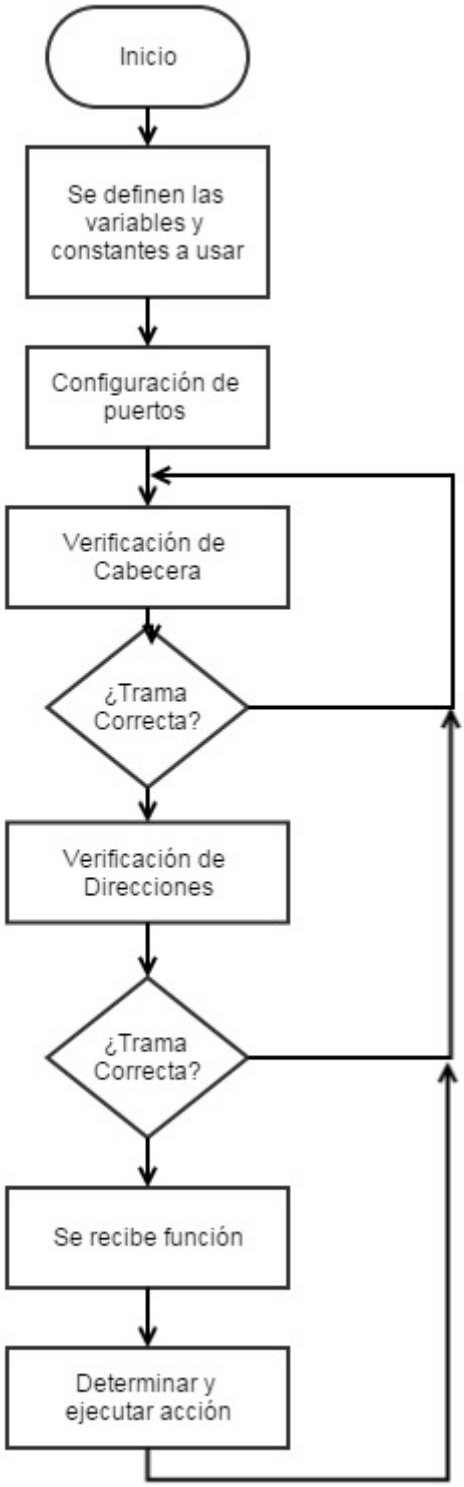
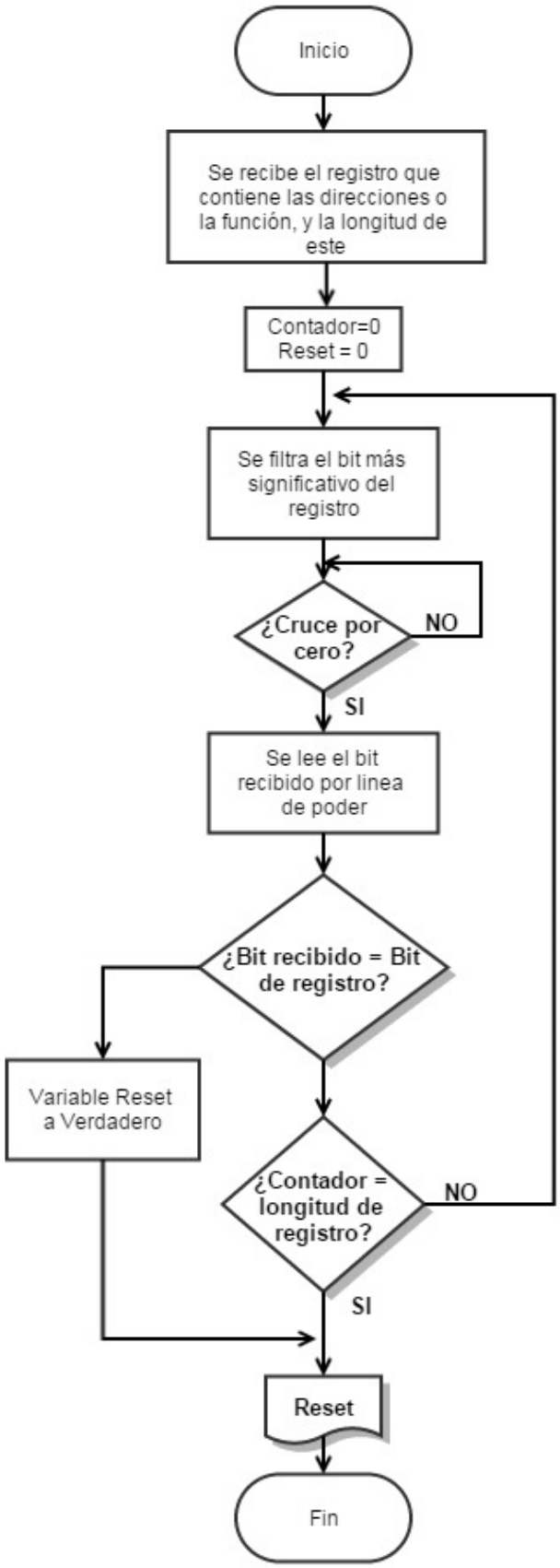


Diagrama de Flujo de Recepción de Pulsos PLC



- **Código C++ del Módulo Controlable de Control de Intensidad Luminosa**

```
/*
 * Programa Módulo de Controlable
 *
 *  Author: Miguel Guzmán Guerra - Renzo Burga Velarde
 *
 */

//Se define uso de librerias
#define F_CPU 20000000L
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>
#define high(x)     ((x)>>8)
#define low(x)      ((x) & 0xFF)

//Declaración de funciones
void Config_Puertos(void);
char rx_trama();
char rx_registro(unsigned char, int, int);
unsigned char rx_funcion(int);
int    verifica_funcion(unsigned char,int);
void Delay_us(int n);

//declaración de constantes y variables
char data;
unsigned char cabecera=0b1110;
unsigned char casa=0b01101001;
unsigned char unidad=0b01101001;
unsigned char funcion_on=0b01011001;
unsigned char funcion_off=0b01011010;
unsigned char funcion_atenuar=0b01100101;
unsigned char funcion_aumentar=0b01100110;
unsigned char sufijo1=0b00000001;
unsigned char sufijo2=0b00000010;


int main(void)
{
      Config_Puertos(); // configuración de puertos

      char reset;
      int cont;
      int estado_luz=5500; //variable usada para realizar el delay
      unsigned char funcion; // para control de iluminación

      while(1)
      {
            inicio:
            PORTD &=~ (1<<PORTD5); //Led de recepcion apagado

            // leemos la cabecera ----------------------------------------------
---
            cont=4;
            reset=rx_registro(cabecera,cont,estado_luz);
            if (reset==1)
            {
                  goto inicio;
            }

            PORTD |= (1<<PORTD5); // Led de recepcion encendido
```

```
                // leemos codigo de casa ----------------------------------------
----
                cont=8;
                reset=rx_registro(casa,cont,estado_luz);
                if (reset==1)
                {
                        goto inicio;
                }

                //leemos codigo de unidad ----------------------------------------
----
                cont=8;
                reset=rx_registro(unidad,cont,estado_luz);
                if (reset==1)
                {
                        goto inicio;
                }

                //leemos sufijo1 ----------------------------------------------
----
                cont=2;
                reset=rx_registro(sufijo1,cont,estado_luz);
                if (reset==1)
                {
                        goto inicio;
                }

                //leemos cabecera ----------------------------------------------
----
                cont=4;
                reset=rx_registro(cabecera,cont,estado_luz);
                if (reset==1)
                {
                        goto inicio;
                }

                //leemos codigo de casa ----------------------------------------
----
                cont=8;
                reset=rx_registro(casa,cont,estado_luz);
                if (reset==1)
                {
                        goto inicio;
                }

                //leemos codigo de funcion -------------------------------------
---
                funcion=rx_funcion(estado_luz);

                //leemos el sufijo2 --------------------------------------------
---
                cont=2;
                reset=rx_registro(sufijo2,cont,estado_luz);
                if (reset==1)
                {
                        goto inicio;
                }

                PORTB=funcion;
                estado_luz=verifica_funcion(funcion,estado_luz);
```

```
                //EVALUAR LA FUNCION RECIBIDA HACER LOS CAMBIOS NECESARIOS
                goto inicio;
        }
}

//función de configuración de puertos
//no devuelve valor alguno
//no recive valor alguno
void Config_Puertos(void)
{
        DDRD = (1<<DDD7)|(1<<DDD5)|(0<<DDD3)|(0<<DDD2);
        PORTD = 0x00;
        DDRB = 0xFF;
        PORTB= 0x00;
}

//función de recepción de tramas
//devuelve el estado de reset para verificar recepción correcta
//recibe el valor de las diferentes tramas, la longitud de estas y el estado
//del delay para el control de intensidad.
char rx_registro(unsigned char registro, int contador, int estado)
{
        int    i;
        unsigned char aux;
        char reset;

        for (i=0;i<contador;i=i+1)
        {
                nolee_registro:
                if (!(PIND & (1<<PIND3)))
                {
                        goto nolee_registro; //aun no hay cruce por cero
                }

                if (PIND & (1<<PIND3))
                {
                        PORTD &=~ (1<<PORTD7);
                        //     Hay cruce por cero se leera un bit
                        _delay_us(20);
                        aux= registro & (1<<(contador-1)); //    Me quedo con el MSB
de registro
                        aux=(aux>>(contador-1));
                        if (aux==0)   //     Si el dato del registro es igual a 0 se
analiza similitud
                        {
                                if (!(PIND & (1<<PIND2)))//es igual al dato leido
                                {
                                        if (estado==0)
                                        {
                                                PORTD |= (1<<PORTD7);
                                                _delay_us(800);
                                                goto sigue1;
                                        }
                                        _delay_us(800);
                                        if (estado<5499)
                                        {
                                                Delay_us(estado);
                                                PORTD |= (1<<PORTD7);
                                        }
                                        sigue1:
                                        ;
                                }
```

```
                        if (PIND & (1<<PIND2)) //es diferente sale de la
funcion
                        {
                                if (estado==0)
                                {
                                        PORTD |= (1<<PORTD7);
                                        _delay_us(800);
                                        goto sigue2 ;
                                }

                                _delay_us(800);

                                if (estado<5499)
                                {
                                        Delay_us(estado);
                                        PORTD |= (1<<PORTD7);
                                }
                                sigue2:
                                reset=1;
                                goto resetear;
                        }
                }
                if (aux==1)   //     Si el bit recogido es igual a 1 se
analiza similitud
                {
                        if (!(PIND & (1<<PIND2)))//es diferente al dato leido
                        {
                                if (estado==0)
                                {
                                        PORTD |= (1<<PORTD7);
                                        _delay_us(800);
                                        goto sigue3;
                                }

                                _delay_us(800);
                                if (estado<5499)
                                {
                                        Delay_us(estado);
                                        PORTD |= (1<<PORTD7);
                                }
                                sigue3:
                                reset=1;
                                goto resetear;
                        }
                        if (PIND & (1<<PIND2))
                        {
                                if (estado==0)
                                {
                                        PORTD |= (1<<PORTD7);
                                        _delay_us(800);
                                        goto sigue4;
                                }

                                _delay_us(800);
                                if (estado<5499)
                                {
                                        Delay_us(estado);
                                        PORTD |= (1<<PORTD7);
                                }
                                sigue4: ;
                        }
                }
```

```
                        registro=registro<<1; //se corre registro
                        //se retorna a leer el siguiente bit

                }
        }
        reset=0;
        resetear:
        return reset;

}

//función de recepción de funciones
//Devuelve el caracter recibido
//recibe la variable de estado de luz
unsigned char rx_funcion(int estado)
{
        int    i;
        unsigned char aux1;

        for (i=8;i>0;i=i-1)
        {
                nolee_registro:
                if (!(PIND & (1<<PIND3)))
                {
                        goto nolee_registro; //aun no hay cruce por cero
                }

                if (PIND & (1<<PIND3))
                {
                        PORTD &=~ (1<<PORTD7);
                        //     Hay cruce por cero se leera un bit
                        _delay_us(50);

                        if (PIND & (1<<PIND2))
                        {
                                aux1 |= (1<<(i-1));
                        }

                        if (!(PIND & (1<<PIND2)))
                        {
                                aux1 &=~ (1<<(i-1));
                        }
                }

                if (estado<5000)
                {
                        Delay_us(estado);
                        PORTD |= (1<<PORTD7);
                }
                _delay_ms(1);
        }

        return aux1;
}

//función de acción de acuerdo a la función recibida
//Devuelve el estado del delay
//recibe la funcion y el estado del delay
int    verifica_funcion(unsigned char funcion, int estado)
{
        if (funcion==funcion_on) //si se enciende el foco desde el cruce por cero
        {                                       // el delay debe ser 0
```

```
                estado=0;
        }

        if (funcion==funcion_off) //apaga el foco todo el tiempo
        {
                estado=5500;
        }

        if (funcion==funcion_atenuar) //aumenta el delay disminuye la intensidad
        {
                estado=estado+200;
                if (estado>5400)
                {
                        estado=5500;
                }
        }

        if (funcion==funcion_aumentar) //disminuy el delay aumenta la intensidad
        {
                estado=estado-200;
                if (estado<0)
                {
                        estado=0;
                }
        }

        return estado;
}

//función de retardo en base a una variable
//no devuelve nada
//recibe la variable con el valor en us del tiempo de delay
void Delay_us(int n) //sirve para hacer un delay basado en una variable
{
        while (n--)
        {
                _delay_us(1);
        }
}
```

- **Código C++ del Módulo Controlable de Control de Tomacorrientes**

```
/*
 * Programa Módulo de Controlable
 *
 *  Author: Miguel Guzmán Guerra - Renzo Burga Velarde
 *
 */

//Se define uso de librerias
#define F_CPU 20000000L
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>
#define high(x)     ((x)>>8)
#define low(x)      ((x) & 0xFF)

//Declaración de funciones
void Config_Puertos(void);
char rx_trama();
char rx_registro(unsigned char, int);
```

```
unsigned char rx_funcion();
void   verifica_funcion(unsigned char);
void Delay_us(int n);

//declaración de constantes y variables
char data;
unsigned char cabecera=0b1110;
unsigned char casa=0b01101001;
unsigned char unidad=0b01011001;
unsigned char funcion_on=0b01011001;
unsigned char funcion_off=0b01011010;
unsigned char sufijo1=0b00000001;
unsigned char sufijo2=0b00000010;


int main(void)
{
Config_Puertos();

char reset;
int cont;
unsigned char funcion;

while(1)
{
inicio:
PORTD &=~ (1<<PORTD5); //Led de recepcion apagado

// leemos la cabecera ------------------------------------------------
cont=4;
reset=rx_registro(cabecera,cont);
if (reset==1)
{
goto inicio;
}

PORTD |= (1<<PORTD5); // Led de recepcion encendido

// leemos codigo de casa ---------------------------------------------
cont=8;
reset=rx_registro(casa,cont);
if (reset==1)
{
goto inicio;
}

//leemos codigo de unidad --------------------------------------------
cont=8;
reset=rx_registro(unidad,cont);
if (reset==1)
{
goto inicio;
}

//leemos sufijo1 -----------------------------------------------------
cont=2;
reset=rx_registro(sufijo1,cont);
if (reset==1)
{
goto inicio;
}
```

```
//leemos cabecera ----------------------------------------------------
cont=4;
reset=rx_registro(cabecera,cont);
if (reset==1)
{
goto inicio;
}

//leemos codigo de casa ----------------------------------------------
cont=8;
reset=rx_registro(casa,cont);
if (reset==1)
{
goto inicio;
}

//leemos codigo de funcion -------------------------------------------
funcion=rx_funcion();

//leemos el sufijo2 --------------------------------------------------
cont=2;
reset=rx_registro(sufijo2,cont);
if (reset==1)
{
goto inicio;
}

PORTB=funcion;
verifica_funcion(funcion);

//EVALUAR LA FUNCION RECIBIDA HACER LOS CAMBIOS NECESARIOS
goto inicio;
}
}

//función de configuración de puertos
//no devuelve valor alguno
//no recive valor alguno
void Config_Puertos(void)
{
DDRD = (1<<DDD7)|(1<<DDD6)|(1<<DDD5)|(0<<DDD3)|(0<<DDD2);
PORTD = 0x00;
}

//función de recepción de tramas
//devuelve el estado de reset para verificar recepción correcta
//recibe el valor de las diferentes tramas, la longitud de estas y el estado
//del delay para el control de intensidad.
char rx_registro(unsigned char registro, int contador)
{
	int	i;
	unsigned char aux;
	char reset;

	for (i=0;i<contador;i=i+1)
	{
		nolee_registro:
		if (!(PIND & (1<<PIND3)))
		{
			goto nolee_registro; //aun no hay cruce por cero
		}
```

```
                if (PIND & (1<<PIND3)) //hay cruce por cero se leera un bit
                {
                        _delay_us(20);
                        aux= registro & (1<<(contador-1)); //    Me quedo con el MSB
de registro
                        aux=(aux>>(contador-1));
                        if (aux==0)   //     Si el dato del registro es igual a 0 se
analiza similitud
                        {
                                if (!(PIND & (1<<PIND2)))//es igual al dato leido
                                {
                                        _delay_ms(2);
                                }
                                if (PIND & (1<<PIND2)) //es diferente sale de la
funcion
                                {
                                        _delay_ms(2);
                                        reset=1;
                                        goto resetear;       //resetea el proceso de
recepción
                                }
                        }
                        if (aux==1)   //     Si el bit recogido es igual a 1 se
analiza similitud
                        {
                                if (!(PIND & (1<<PIND2)))//es diferente al dato leido
                                {
                                        _delay_ms(2);
                                        reset=1;
                                        goto resetear; //resetea el proceso de
recepción
                                }
                                if (PIND & (1<<PIND2))
                                {
                                        _delay_ms(2);
                                }
                        }
                        registro=registro<<1; //se corre registro
                        //se retorna a leer el siguiente bit

                }
        }
        reset=0;
        resetear:
        return reset;

}

//función de recepción de funciones
//Devuelve el caracter recibido
//recibe la variable de estado de luz
unsigned char rx_funcion()
{
int    i;
unsigned char aux1;

for (i=8;i>0;i=i-1)
{
nolee_registro:
if (!(PIND & (1<<PIND3)))
{
        goto nolee_registro; //aun no hay cruce por cero
```

```
}

if (PIND & (1<<PIND3))
{
       //     Hay cruce por cero se leera un bit
       _delay_us(50);

       if (PIND & (1<<PIND2))
       {
       aux1 |= (1<<(i-1));
       }

       if (!(PIND & (1<<PIND2)))
       {
       aux1 &=~ (1<<(i-1));
       }
}
_delay_ms(2);
}

return aux1;
}

//función de acción de acuerdo a la función recibida
//Devuelve el estado del delay
//recibe la funcion y el estado del delay
void verifica_funcion(unsigned char funcion)
{
if (funcion==funcion_on)
{
PORTD |= (1<<PORTD7);
}

if (funcion==funcion_off)
{
PORTD &=~ (1<<PORTD7);
}

}

//función de retardo en base a una variable
//no devuelve nada
//recibe la variable con el valor en us del tiempo de delay

void Delay_us(int n)
{
while (n--)
{
_delay_us(1);
}
}
```

- **Código C++ del Módulo Controlable de Control de Cortinas**

```
/*
 * Programa Módulo de Controlable
 *
 *  Author: Miguel Guzmán Guerra - Renzo Burga Velarde
 *
 */

//Se define uso de librerias
```

```
#define F_CPU 20000000L
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>
#define high(x)      ((x)>>8)
#define low(x)       ((x) & 0xFF)

//Declaración de funciones
void Config_Puertos(void);
char rx_trama();
char rx_registro(unsigned char, int);
unsigned char rx_funcion();
void   verifica_funcion(unsigned char);
void Delay_us(int n);

//declaración de constantes y variables
char data;
unsigned char cabecera=0b1110;
unsigned char casa=0b01101001;
unsigned char unidad=0b10101001;
unsigned char funcion_on=0b01011001;
unsigned char funcion_off=0b01011010;
unsigned char sufijo1=0b00000001;
unsigned char sufijo2=0b00000010;


int main(void)
{
	Config_Puertos();

	char reset;
	int cont;
	unsigned char funcion;

	while(1)
	{
		inicio:
		PORTD &=~ (1<<PORTD5); //Led de recepcion apagado

		// leemos la cabecera ---------------------------------------------
---
		cont=4;
		reset=rx_registro(cabecera,cont);
		if (reset==1)
		{
			goto inicio;
		}

		PORTD |= (1<<PORTD5); // Led de recepcion encendido

		// leemos codigo de casa ------------------------------------------
----
		cont=8;
		reset=rx_registro(casa,cont);
		if (reset==1)
		{
			goto inicio;
		}

		//leemos codigo de unidad ----------------------------------------
----
		cont=8;
```

```
                reset=rx_registro(unidad,cont);
                if (reset==1)
                {
                        goto inicio;
                }

                //leemos sufijo1 ------------------------------------------------
----
                cont=2;
                reset=rx_registro(sufijo1,cont);
                if (reset==1)
                {
                        goto inicio;
                }

                //leemos cabecera -----------------------------------------------
----
                cont=4;
                reset=rx_registro(cabecera,cont);
                if (reset==1)
                {
                        goto inicio;
                }

                //leemos codigo de casa -----------------------------------------
----
                cont=8;
                reset=rx_registro(casa,cont);
                if (reset==1)
                {
                        goto inicio;
                }

                //leemos codigo de funcion --------------------------------------
---
                funcion=rx_funcion();

                //leemos el sufijo2 ---------------------------------------------
---
                cont=2;
                reset=rx_registro(sufijo2,cont);
                if (reset==1)
                {
                        goto inicio;
                }

                PORTB=funcion;
                verifica_funcion(funcion);

                //EVALUAR LA FUNCION RECIBIDA HACER LOS CAMBIOS NECESARIOS
                goto inicio;
        }
}

//función de configuración de puertos
//no devuelve valor alguno
//no recive valor alguno
void Config_Puertos(void)
{
        DDRD = (1<<DDD7)|(1<<DDD6)|(1<<DDD5)|(0<<DDD3)|(0<<DDD2);
        PORTD = 0x00;
}
```

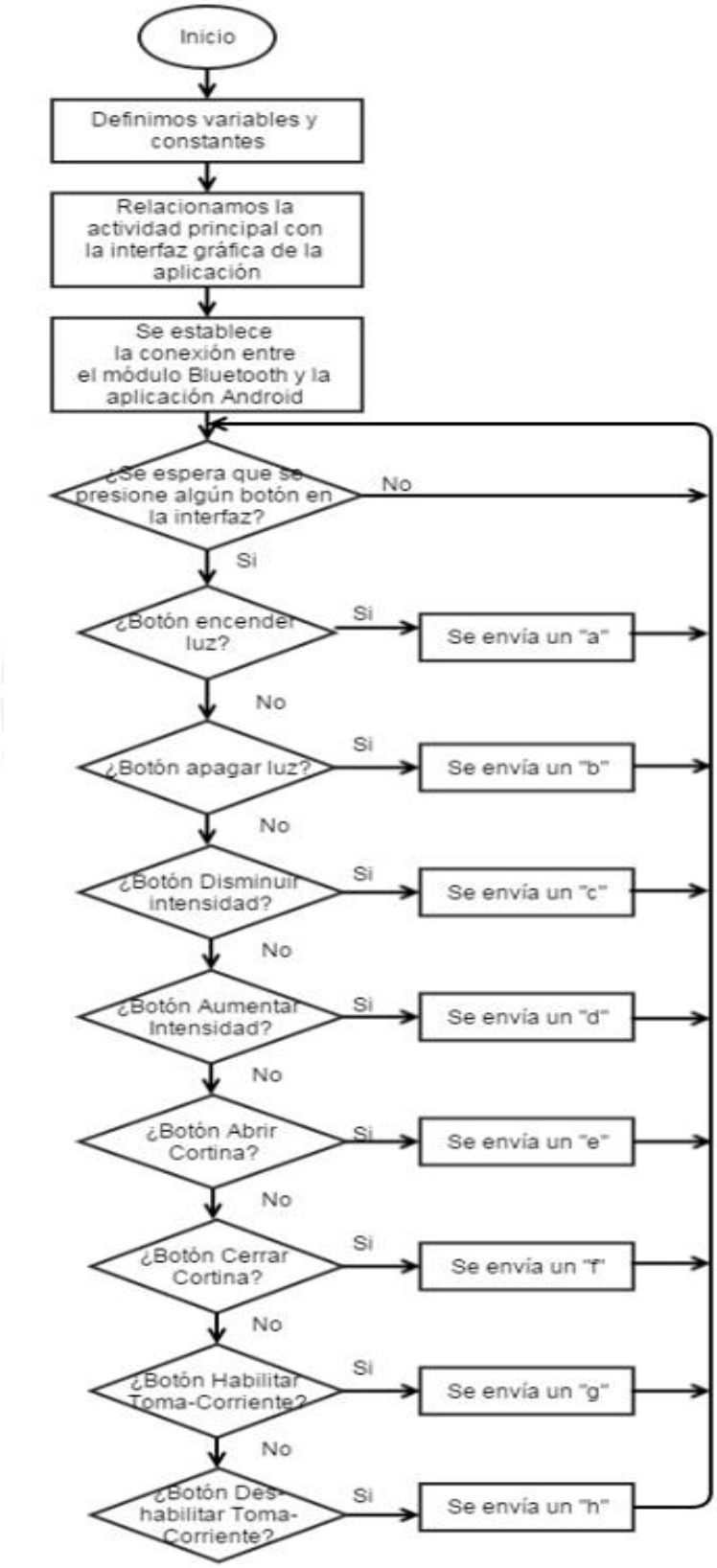
```
//función de recepción de tramas
//devuelve el estado de reset para verificar recepción correcta
//recibe el valor de las diferentes tramas y la longitud de estas
char rx_registro(unsigned char registro, int contador)
{
	int	i;
	unsigned char aux;
	char reset;

	for (i=0;i<contador;i=i+1)
	{
		nolee_registro:
		if (!(PIND & (1<<PIND3)))
		{
			goto nolee_registro; //aun no hay cruce por cero
		}

		if (PIND & (1<<PIND3)) //hay cruce por cero se leera un bit
		{
			_delay_us(20);
			aux= registro & (1<<(contador-1)); //	Me quedo con el MSB
de registro
			aux=(aux>>(contador-1));
			if (aux==0)	//	Si el dato del registro es igual a 0 se
analiza similitud
			{
				if (!(PIND & (1<<PIND2)))//es igual al dato leido
				{
					_delay_ms(2);
				}
				if (PIND & (1<<PIND2)) //es diferente sale de la
funcion
				{
					_delay_ms(2);
					reset=1;
					goto resetear;	//resetea el proceso de
recepción
				}
			}
			if (aux==1)	//	Si el bit recogido es igual a 1 se
analiza similitud
			{
				if (!(PIND & (1<<PIND2)))//es diferente al dato leido
				{
					_delay_ms(2);
					reset=1;
					goto resetear; //resetea el proceso de
recepción
				}
				if (PIND & (1<<PIND2))
				{
					_delay_ms(2);
				}
			}
			registro=registro<<1; //se corre registro
			//se retorna a leer el siguiente bit

		}
	}
	reset=0;
	resetear:
```

```
	return reset;

}

//función de recepción de funciones
//Devuelve el caracter recibido
//no recibe nada
unsigned char rx_funcion()
{
	int	i;
	unsigned char aux1;

	for (i=8;i>0;i=i-1)
	{
		nolee_registro:
		if (!(PIND & (1<<PIND3)))
		{
			goto nolee_registro; //aun no hay cruce por cero
		}

		if (PIND & (1<<PIND3))
		{
			//	Hay cruce por cero se leera un bit
			_delay_us(50);

			if (PIND & (1<<PIND2))
			{
				aux1 |= (1<<(i-1));
			}

			if (!(PIND & (1<<PIND2)))
			{
				aux1 &=~ (1<<(i-1));
			}
		}
		_delay_ms(2);
	}

	return aux1;
}

//función de acción de acuerdo a la función recibida
//no devuelve nada
//recibe la funcion
void verifica_funcion(unsigned char funcion)
{
	if (funcion==funcion_on)
	{
		PORTD &=~ (1<<PORTD6);
		PORTD &=~ (1<<PORTD7);
		PORTD |= (1<<PORTD6);
		PORTD &=~ (1<<PORTD7);
	}

	if (funcion==funcion_off)
	{
		PORTD &=~ (1<<PORTD6);
		PORTD &=~ (1<<PORTD7);
		PORTD &=~ (1<<PORTD6);
		PORTD |= (1<<PORTD7);
	}
```

```
}

//función de retardo en base a una variable
//no devuelve nada
//recibe la variable con el valor en us del tiempo de delay
void Delay_us(int n)
{
	while (n--)
	{
		_delay_us(1);
	}
}
```

ANEXO D

- **Programa de la Aplicación Móvil**
  - **Diagrama de Flujo**



- **Código Eclipse**

❖ El Activity Main en java:

```
//////////////////////////////////////////////////////////////////////////////////////
//Tesis: Sistema Domotico de control centralizado con comunicacion por linea de
poder
//Autores: Burga, Renzo; Guzman, Miguel
//////////
//Este código recibe un instruccion por la interfaz grafica y segun la instruccion
//se envia una señal al modulo central .
//////
//Esta parte de codigo se encarga de enlazar el interfaz grafico y darle funciones
//a cada elemento para que el usuario pueda intuir el control
//////////////////////////////////////////////////////////////////////////////////////


package pe.edu.pucp.domoblue;

//En esta seccion se encuentran las diferentes librerias para las funciones utilizadas en
el codigo
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.lang.reflect.Method;
import java.util.UUID;

import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.os.Build;
import android.os.Bundle;
import android.os.Handler;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ImageButton;
import android.widget.TextView;
import android.widget.Toast;
```

```
//inicio del codigo

public class MainActivity extends Activity {
        //Inicialización del las variables y constantes
        private static final String TAG = "bluetooth2";

         ImageButton
btnOn1,btnOff1,btnUp1,btnDown1,btnOp1,btnCl1,btnEn1,btnDis1 ;
         TextView txtArduino;
         Handler h;

         final int RECIEVE_MESSAGE = 1;       // Estado para el handler es la clase que
controlara los mensajes
         private BluetoothAdapter btAdapter = null;
         private BluetoothSocket btSocket = null;
         private StringBuilder sb = new StringBuilder();

         private ConnectedThread mConnectedThread;

         // Se define el UUID para la comunicación serial
         private static final UUID MY_UUID = UUID.fromString("00001101-0000-1000-
8000-00805F9B34FB");

         // Se coloca la MAC address del HC-05 que se utilizara para realizar la conexión
         private static String address = "98:D3:31:20:0B:2A";

   @Override
   protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_main);
     //En este segmento
     //Se relaciona el Main Activity con el Layout
     btnOn1= (ImageButton)findViewById(R.id.btnOn1);
        //Boton de prendido de lampara
     btnOff1 =(ImageButton)findViewById(R.id.btnOff1);           //Boton de apagado
de lampara
     btnUp1  =(ImageButton)findViewById(R.id.btnUp1);            //Boton de aumento
de intensidad
     btnDown1 = (ImageButton)findViewById(R.id.btnDown1);         //Boton de
atenuacion de la intensidad
     btnOp1 = (ImageButton)findViewById(R.id.btnOp1);            //Boton de Abrir
cortinas
     btnCl1 = (ImageButton)findViewById(R.id.btnCl1);           //Boton de Cerrar
cortinas
     btnEn1 = (ImageButton)findViewById(R.id.btnEn1);            //Boton de Habilitar
toma corriente
     btnDis1 = (ImageButton)findViewById(R.id.btnDis1);          //Boton de D
```

```
    txtArduino = (TextView)findViewById(R.id.txtArduino);     // Para que se obtenga
un mensaje del modulo

    h = new Handler() {
      public void handleMessage(android.os.Message msg) {
        switch (msg.what) {
        case RECIEVE_MESSAGE:                                    // Si se recibe u mensaje
          byte[] readBuf = (byte[]) msg.obj;
          String strIncom = new String(readBuf, 0, msg.arg1);            // Se crea una
cadena por un arreglo de bytes
          sb.append(strIncom);                                  // Se ingresa el contenido a
la cadena
          int endOfLineIndex = sb.indexOf("\r\n");                    // Se determina el
fin de linea
          if (endOfLineIndex > 0) {                            // Si el fin de linea,
            String sbprint = sb.substring(0, endOfLineIndex);           // extraer la
cadena
            sb.delete(0, sb.length());                          // y limpiar
            txtArduino.setText("Data from Arduino: " + sbprint);         // Actualizar el
text view
            btnOff1.setEnabled(true);
            btnOn1.setEnabled(true);
            btnUp1.setEnabled(true);
            btnDown1.setEnabled(true);
            btnOp1.setEnabled(true);
            btnCl1.setEnabled(true);
            btnEn1.setEnabled(true);
            btnDis1.setEnabled(true);
          }
          //Log.d(TAG, "...String:"+ sb.toString() +  "Byte:" + msg.arg1 + "...");
          break;
        }
      };
    };

    btAdapter = BluetoothAdapter.getDefaultAdapter();      // Obtener el adaptador
de bluetooth
    checkBTState();


    btnOn1.setOnClickListener(new OnClickListener() {
      public void onClick(View v) {
       btnOn1.setEnabled(true);
       mConnectedThread.write("a");    // Se envia "a" via Bluetooth
       //Toast.makeText(getBaseContext(), "Turn on LED",
Toast.LENGTH_SHORT).show();
      }
```

```
        });

      btnOff1.setOnClickListener(new OnClickListener() {
       public void onClick(View v) {
        btnOff1.setEnabled(true);
        mConnectedThread.write("b");    // Se envia "b" via Bluetooth
        //Toast.makeText(getBaseContext(), "Turn off LED",
Toast.LENGTH_SHORT).show();
        }
      });
      btnUp1.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
         btnUp1.setEnabled(true);
         mConnectedThread.write("d");    // Se envia "d" via Bluetooth
         //Toast.makeText(getBaseContext(), "Turn off LED",
Toast.LENGTH_SHORT).show();
        }
       });
      btnDown1.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
         btnDown1.setEnabled(true);
         mConnectedThread.write("c");    // Se envia "c" via Bluetooth
         //Toast.makeText(getBaseContext(), "Turn off LED",
Toast.LENGTH_SHORT).show();
        }
       });
      btnOp1.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
         btnOp1.setEnabled(true);
         mConnectedThread.write("e");    // Se envia "e" via Bluetooth
         //Toast.makeText(getBaseContext(), "Turn off LED",
Toast.LENGTH_SHORT).show();
        }
       });
      btnCl1.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
         btnCl1.setEnabled(true);
         mConnectedThread.write("f");    // Se envia "f" via Bluetooth
         //Toast.makeText(getBaseContext(), "Turn off LED",
Toast.LENGTH_SHORT).show();
        }
       });
      btnEn1.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
         btnEn1.setEnabled(true);
         mConnectedThread.write("g");    // Se envia "g" via Bluetooth
```

```
            //Toast.makeText(getBaseContext(), "Turn off LED",
Toast.LENGTH_SHORT).show();
          }
        });
      btnDis1.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
          btnDis1.setEnabled(true);
          mConnectedThread.write("h");    // Se envia "h" via Bluetooth
          //Toast.makeText(getBaseContext(), "Turn off LED",
Toast.LENGTH_SHORT).show();
          }
        });
  }
  private BluetoothSocket createBluetoothSocket(BluetoothDevice device) throws
IOException {
    if(Build.VERSION.SDK_INT >= 10){
      try {
        final Method  m =
device.getClass().getMethod("createInsecureRfcommSocketToServiceRecord", new
Class[] { UUID.class });
        return (BluetoothSocket) m.invoke(device, MY_UUID);
      } catch (Exception e) {
        Log.e(TAG, "Could not create Insecure RFComm Connection",e);
      }
    }
    return  device.createRfcommSocketToServiceRecord(MY_UUID);
  }


  @Override
  public void onResume() {
   super.onResume();

    Log.d(TAG, "...onResume - try connect...");

    // se establece un indicador a el nodo remoto usado su direccion MAC.
    BluetoothDevice device = btAdapter.getRemoteDevice(address);

    // requerimientos para hacer una conexion
    //   La direccion MAC que ya tenemos.
    //   Y un ID o un UUID.  En este caso se usara un UUID para SSP

    try {
      btSocket = createBluetoothSocket(device);
    } catch (IOException e) {
      errorExit("Fatal Error", "In onResume() and socket create failed: " +
e.getMessage() + ".");
```

```
    }

    // Este codigo hace que deje de buscar dispositivos
    // para que no haya interferencias en la transferencia de la aplicacion.
    btAdapter.cancelDiscovery();

    // Establece la coneccion.  Este esperara hasta que se logre conectar.
    Log.d(TAG, "...Connecting...");
    try {
      btSocket.connect();
      Log.d(TAG, "....Connection ok...");
    } catch (IOException e) {
      try {
        btSocket.close();
      } catch (IOException e2) {
        errorExit("Fatal Error", "In onResume() and unable to close socket during
connection failure" + e2.getMessage() + ".");
      }
    }

    // Se crea un data stream para que se comunique mediante un server
    Log.d(TAG, "...Create Socket...");

    mConnectedThread = new ConnectedThread(btSocket);
    mConnectedThread.start();
  }

  @Override
  public void onPause() {
    super.onPause();

    Log.d(TAG, "...In onPause()...");

    try     {
      btSocket.close();
    } catch (IOException e2) {
      errorExit("Fatal Error", "In onPause() and failed to close socket." +
e2.getMessage() + ".");
    }
  }

  private void checkBTState() {
    // Revisa el estado del bluetooth para que no este apagado
    //Los emuladores no soportan el bluetooth por ende saldra un null
    if(btAdapter==null) {
      errorExit("Fatal Error", "Bluetooth not support");
    } else {
```

```
      if (btAdapter.isEnabled()) {
        Log.d(TAG, "...Bluetooth ON...");
      } else {
        //Con este codigo se activa el bluetooh
        Intent enableBtIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(enableBtIntent, 1);
      }
     }
    }

    private void errorExit(String title, String message){
     Toast.makeText(getBaseContext(), title + " - " + message,
Toast.LENGTH_LONG).show();
     finish();
    }
    private class ConnectedThread extends Thread {
      private final InputStream mmInStream;
      private final OutputStream mmOutStream;

      public ConnectedThread(BluetoothSocket socket) {
        InputStream tmpIn = null;
        OutputStream tmpOut = null;

        // se obtienen las entradas y salidas,utilizando objetos temporales
        // member streams are final
        try {
          tmpIn = socket.getInputStream();
          tmpOut = socket.getOutputStream();
        } catch (IOException e) { }

        mmInStream = tmpIn;
        mmOutStream = tmpOut;
      }

      public void run() {
        byte[] buffer = new byte[256];  // Se crea un buffer para el stream
        int bytes; // Y los Bytes se regresan de la funcion read()

        // Se mantiente esperando una señal de entrada
        while (true) {
          try {
            // Se lee de la señal de entrada
            bytes = mmInStream.read(buffer);       // Se obtinen los bytes del stream
            h.obtainMessage(RECIEVE_MESSAGE, bytes, -1, buffer).sendToTarget();
//se envia un mensaje al handler
          } catch (IOException e) {
```

```
                break;
            }
        }
    }

    /* Se llama a esta funcion de la actividad principal para poder enviar datos */
    public void write(String message) {
        Log.d(TAG, "...Data to send: " + message + "...");
        byte[] msgBuffer = message.getBytes();
        try {
            mmOutStream.write(msgBuffer);
        } catch (IOException e) {
            Log.d(TAG, "...Error data send: " + e.getMessage() + "...");
        }
    }
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
}


@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();
    if (id == R.id.action_settings) {
        return true;
    }
    return super.onOptionsItemSelected(item);
}
}
```

❖ El Activity Main en XML

```
//Activity_main.xml

// Este codigo es el Layout Gráfico en modo código aca se podra observar
// la posición de los botones como los textos y ademas agregar diferentes tipos
// elementos con los que el usuario puede interactuar.
// Ademas en este codigo se relacionan los Strings y los ID de los
// Elementos gráficos
```

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:background="@drawable/fondo"
  android:paddingBottom="@dimen/activity_vertical_margin"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  android:visibility="visible"
  tools:context="pe.edu.pucp.domoblue.MainActivity" >


  <TextView
    android:id="@+id/txtArduino"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_marginBottom="28dp"
    android:layout_marginRight="30dp"
    android:text="@string/app_name"
    android:textAppearance="?android:attr/textAppearanceMedium" />

  <ImageButton
    android:id="@+id/btnOff1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_marginRight="85dp"
    android:layout_marginTop="25dp"
    android:layout_toLeftOf="@+id/btnOp1"
    android:adjustViewBounds="true"
    android:contentDescription="@string/Apagado_txt"
    android:maxHeight="120dp"
    android:maxWidth="120dp"
    android:scaleType="fitXY"
    android:src="@drawable/lampoff" />

  <ImageButton
    android:id="@+id/btnOn1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignTop="@+id/btnOff1"
    android:layout_toLeftOf="@+id/btnOff1"
    android:adjustViewBounds="true"
    android:contentDescription="@string/Prendido_txt"
```

```
    android:maxHeight="120dp"
    android:maxWidth="120dp"
    android:scaleType="fitXY"
    android:src="@drawable/lampon" />

<ImageButton
    android:id="@+id/btnDown1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/btnUp1"
    android:layout_below="@+id/btnUp1"
    android:adjustViewBounds="true"
    android:baselineAlignBottom="true"
    android:contentDescription="@string/Disminuir_txt"
    android:maxHeight="120dp"
    android:maxWidth="120dp"
    android:scaleType="fitXY"
    android:src="@drawable/dow" />

<ImageButton
    android:id="@+id/btnUp1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignTop="@+id/btnOn1"
    android:layout_toLeftOf="@+id/btnOn1"
    android:adjustViewBounds="true"
    android:contentDescription="@string/Aumentar_txt"
    android:maxHeight="120dp"
    android:maxWidth="120dp"
    android:scaleType="fitXY"
    android:src="@drawable/up" />

<ImageButton
    android:id="@+id/btnEn1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_above="@+id/txtArduino"
    android:layout_marginBottom="91dp"
    android:layout_toRightOf="@+id/btnDis1"
    android:adjustViewBounds="true"
    android:contentDescription="@string/Habilitar_txt"
    android:maxHeight="120dp"
    android:maxWidth="120dp"
    android:scaleType="fitXY"
    android:src="@drawable/plug" />

<ImageButton
```

```
    android:id="@+id/btnDis1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/txtArduino"
    android:layout_alignTop="@+id/btnEn1"
    android:adjustViewBounds="true"
    android:contentDescription="@string/Deshabilitar_txt"
    android:maxHeight="120dp"
    android:maxWidth="120dp"
    android:scaleType="fitXY"
    android:src="@drawable/plugoff" />

  <ImageButton
    android:id="@+id/btnOp1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
    android:layout_below="@+id/btnOff1"
    android:layout_marginRight="258dp"
    android:adjustViewBounds="true"
    android:contentDescription="@string/Abrir_txt"
    android:maxHeight="120dp"
    android:maxWidth="120dp"
    android:scaleType="fitXY"
    android:src="@drawable/curtain" />

  <ImageButton
    android:id="@+id/btnCl1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/btnOp1"
    android:layout_alignTop="@+id/btnOp1"
    android:layout_marginLeft="132dp"
    android:adjustViewBounds="true"
    android:contentDescription="@string/Cerrar_txt"
    android:maxHeight="120dp"
    android:maxWidth="120dp"
    android:scaleType="fitXY"
    android:src="@drawable/cutainclose" />

</RelativeLayout>
```

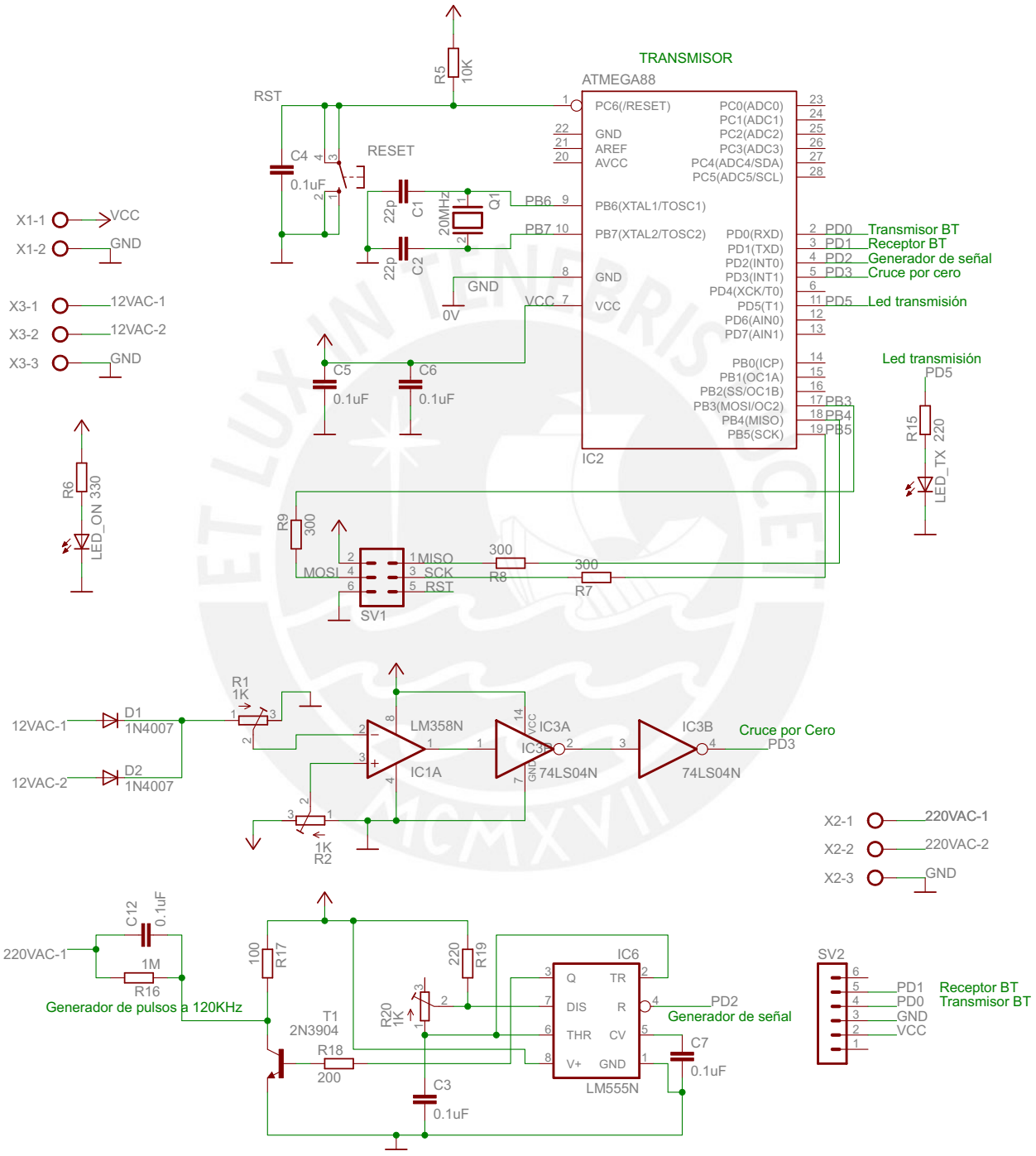
❖ El String en XML

```
//String.XML

<?xml version="1.0" encoding="utf-8"?>
<resources>
// En esta parte del codigo se almacenan las cadenas que seran usadas en el
//interfaz gráfico de la aplicacion
   <string name="app_name">DOMO PUCP</string>
   <string name="hello_world">Hello world!</string>
   <string name="action_settings">Settings</string>
   <string name="Bluetooth_txt">Bluetooth</string>
   <string name="disemp_txt">Dispositivos Emparejados</string>
   <string name="Prendido_txt">Prender</string>
   <string name="Apagado_txt">Apagar</string>
   <string name="Aumentar_txt">Aumentar</string>
   <string name="Disminuir_txt">Disminuir</string>
   <string name="Abrir_txt">Abrir</string>
   <string name="Cerrar_txt">Cerrar</string>
   <string name="Habilitar_txt">Habilitar</string>
   <string name="Deshabilitar_txt">Deshabilitar</string>

</resources>
```

❖ El Android Manifest en XML

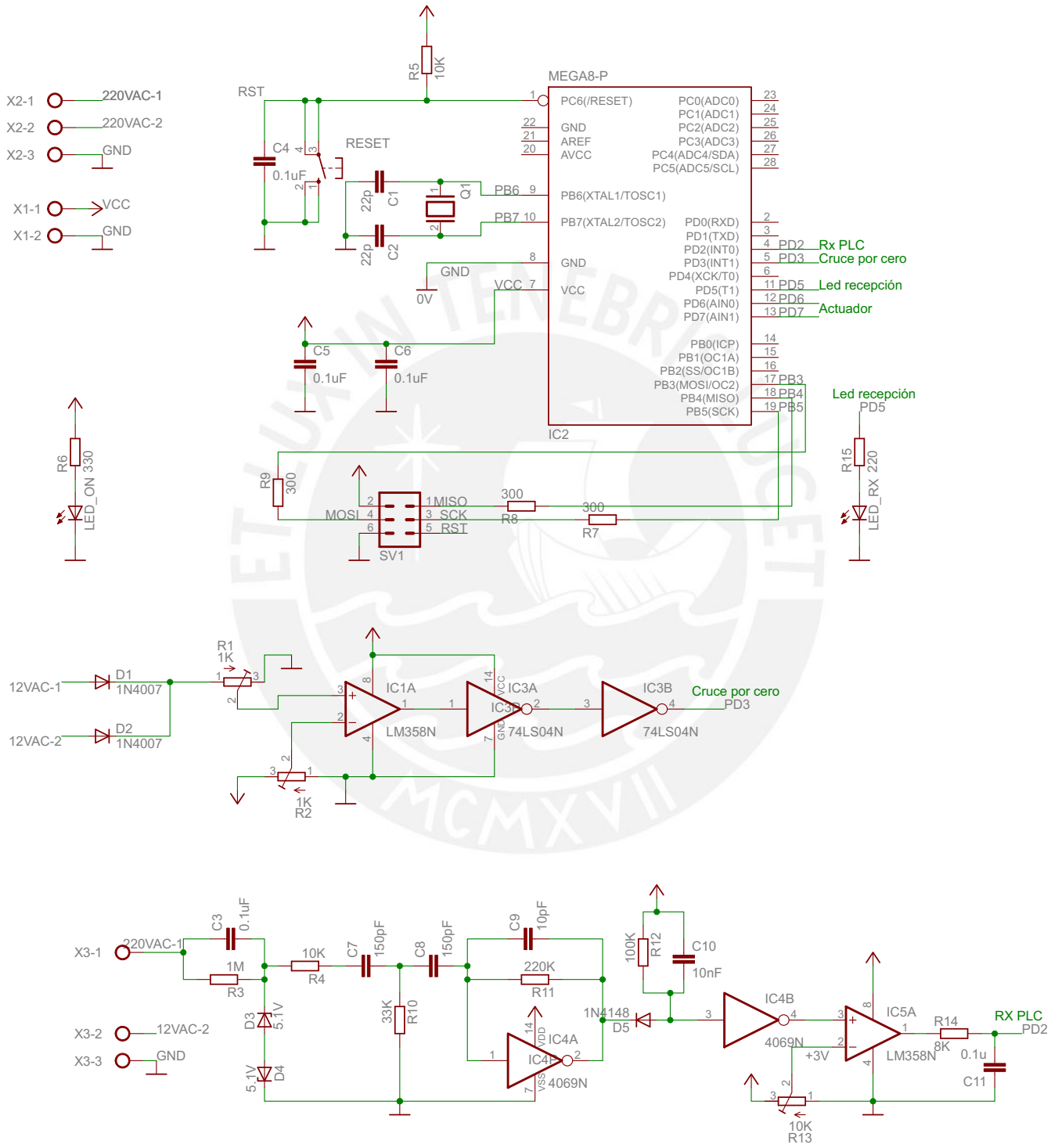
```
//AndroidManifest.Xml

// Es el documento del proyecto donde se pueden ver las principales
// caractersticas de la aplicación


<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="pe.edu.pucp.domoblue"
   android:versionCode="1"
   android:versionName="1.0" >

 //Permisos para poder usar la función Bluetooth en el Android
   <uses-permission android:name="android.permission.BLUETOOTH" />
   <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />

  // Describe la versión mínima y la preferencial
   <uses-sdk
      android:minSdkVersion="19"
      android:targetSdkVersion="19" />
```

```
        // Da las caracteristicas basicas de la aplicación
    <application
      android:allowBackup="true"
      android:icon="@drawable/pucp2"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >
      <activity
        android:name=".MainActivity"
        android:label="@string/app_name" >
        <intent-filter>
          <action android:name="android.intent.action.MAIN" />

          <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
      </activity>
    </application>

</manifest>
```



- **Lista de componentes**

Nro.	Referencia	Valor	Dispositivo
1	C1,C2	22p	Capacitor
2	C3,C4,C5,C6,C7,C12	0,1uF	Capacitor
3	D1,D2	1N4007	Diodo
4	IC1	LM358N	Integrado
5	IC2	ATMEGA88PA	Microprocesador
6	IC3	74LS04	Integrado
7	IC6	Lm555	Integrado
8	LED_ON, LED, RX		LED
9	Q1	20000MHz	Cristal
10	R1,R2,R20	1K	Resistencia
11	R5	10k	Resistencia
12	R6	330	Resistencia
13	R7, R8, R9	300	Resistencia
14	R15, R19	220	Resistencia
15	R16	1M	Resistencia
16	R17	100	Resistencia
17	R18	200	Resistencia
18	Reseteo		Botón

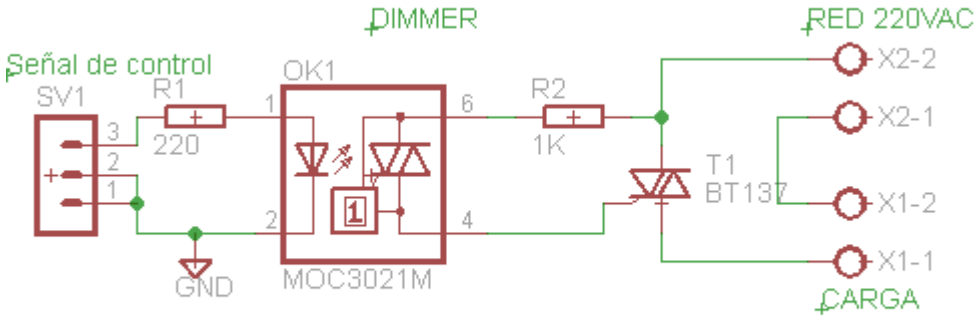


- **Lista de componentes**

Nro.	Referencia	Valor	Dispositivo
1	C1,C2	22p	Capacitor
2	C3,C4,C5,C6,C11	0,1uF	Capacitor
3	C7, C8	150 pF	Capacitor
4	C9, C10	10nF	Capacitor
5	D1,D2	1N4007	Diodo
6	D3,D4	5,1V	Diodo Zener
7	D5	1N4148	Diodo
8	IC1,IC5	LM358N	Integrado
9	IC2	ATMEGA88PA	Microprocesador
10	IC3	74LS04	Integrado
11	IC4	4069N	Integrado
12	LED_ON, LED, RX		LED
13	Q1	20000MHz	Cristal
14	R1,R2	1K	Potenciómetro
15	R3	1M	Resistencia
16	R4, R5	10k	Resistencia
17	R6	330	Resistencia
18	R7, R8, R9	300	Resistencia
19	R10	33K	Resistencia
20	R11	220K	Resistencia
21	R12	100K	Resistencia
22	R13	10K	Resistencia
23	R14	8K	Resistencia
24	R15	220	Resistencia
25	Reseteo		Botón

ANEXO G

- **Esquemático del Dimmer**

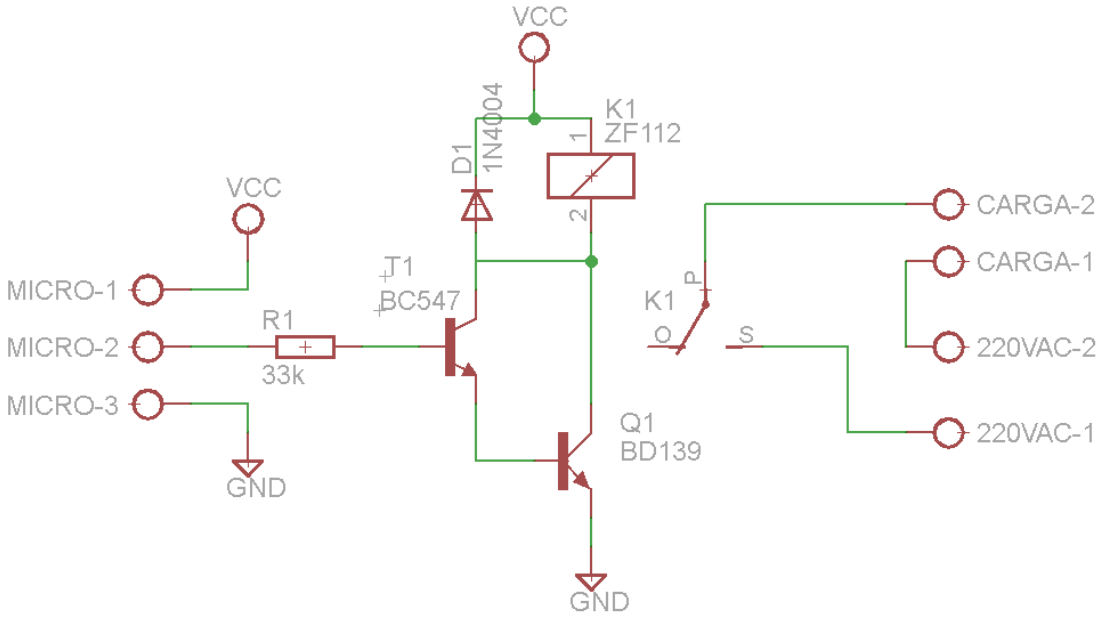


- **Lista de componentes**

Nro.	Referencia	Valor	Dispositivo
1	OK1	MOC3021	Optoaisador
2	T1	BT137	Triac
3	R1	220	Resistencia
4	R2	1k	Resistencia

ANEXO H

- **Esquemático del Relé**

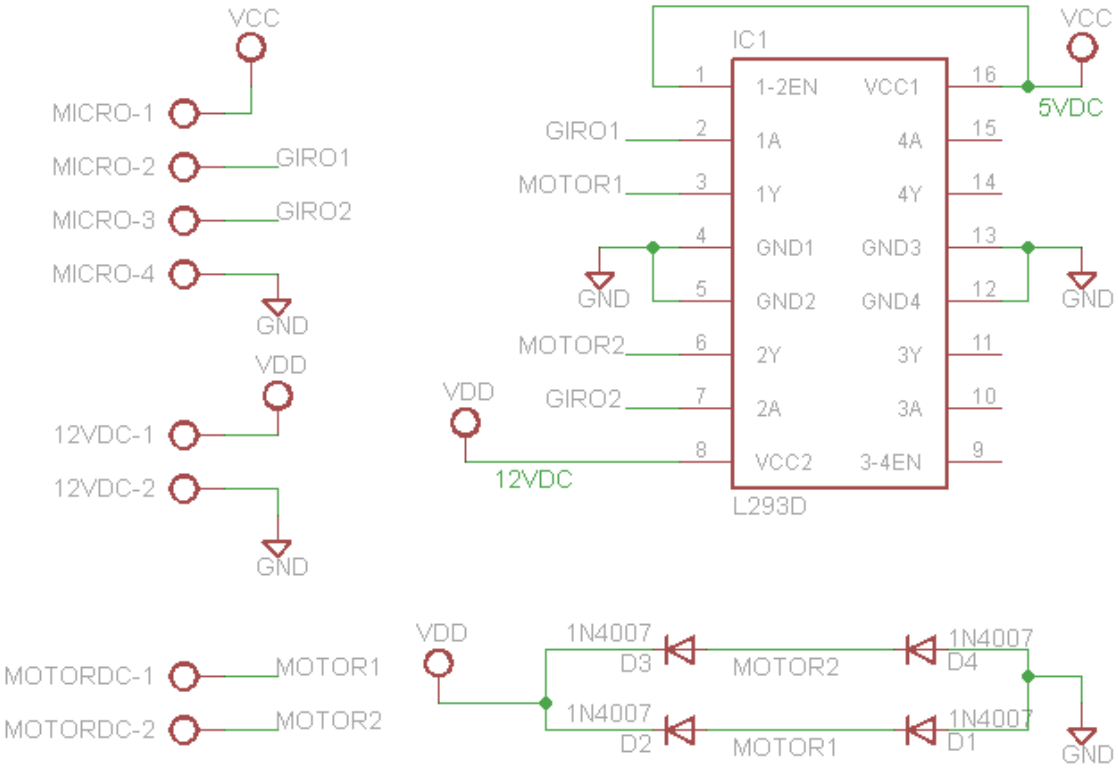


- **Lista de componentes**

Nro.	Referencia	Valor	Dispositivo
1	D1	1N4004	Diodo
2	K1	ZF112	Rele
3	Q1	BD139	Transistor
4	Q2	BC547	Transistor
5	R1	33k	Resistencia

ANEXO I

- **Esquemático del Puente H**



- **Lista de componentes**

Nro.	Referencia	Valor	Dispositivo
1	D1,D2,D3,D4	1N4007	Diodo
2	IC1	L293D	Integrado