Universidad de Castilla - La Mancha

Diseño de Sistemas Basados en Microprocesador

Escuela Superior de Informática

Project 1
Environment/Weather Station

Autores:
Álvaro Cerdá Pulla
Juan Manuel Palacios Navas

Profesor:
Julián Caba Jiménez

7 de Junio de 2020

Índice

1. Introducción

Para este proyecto hemos configurado **5 sensores medioambientales** mediante interrupciones, timers, ADC con sus respectivos canales y el resto de recursos vistos durante el curso y anteriores prácticas.

Necesitamos configurar un pequeño panel de alarmas que consta de 3 LEDs verdes y 3 LEDs rojos que simularán el estado de 3 de los sensores que vamos a realizar.

También usaremos el **TIM 11** con un propósito general, que debido a que no es un sistema crítico, actualizaremos los datos cada 5 segundos, y éste nos ayudará a hacerlo. Su configuración será de:

- **Prescaler:** 42000 - 1
- **Period:** 10000 - 1

Dentro de su Callback, es donde haremos que muestre los datos por el LCD y cambie el estado de los LEDs si alguno de los sensores ha superado el umbral propio.

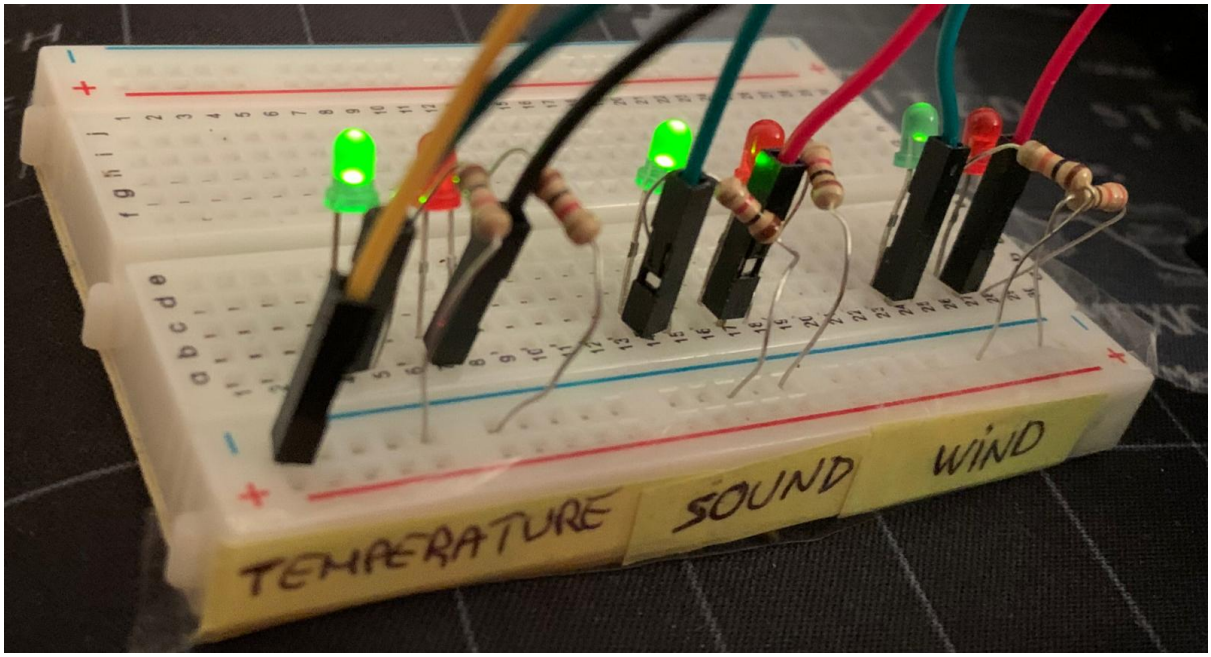

Figura 1: Protoboard con los LEDs de alarma

2. Sensor de Temperatura

Usaremos el sensor de temperatura, que estará conectado al **A2**, mediante ADC_CH2. El sensor de temperatura está compuesto por una **resistencia R1**, definido como R0; y un **termistor NTC**, definido como B. El valor de B puede variar entre 4250 4299K y la **resistencia** son 100k ohmios.

$$NTC = (\frac{Vcc}{V} - 1) * R0 \tag{1}$$

Para obtener el valor del NTC en ohmios usaremos la anterior fórmula, donde **V** es el voltaje recibido por el ADC. **Vcc** es el voltaje de la placa (5 voltios, que corresponde a $2^{12} - 1 = 4095$.)

Esto se debe a que la resolución máxima para 5V es de 12 bits, y **R0** es la resistencia de 100K ohmios, se ha elegido este valor porque es el valor de la resistencia cuando hay 25ºC.

Para pasar medir los grados Kelvin y pasarlos a Celsius, haremos uso de la siguiente fórmula:

$$T = \frac{1}{\frac{1}{To} + \frac{1}{B} * ln(\frac{NTC}{Ro})} \tag{2}$$

```
#define THRESHOLD_TEMPERATURE 20
#define R0 100000 //100 KOhmios
#define B 4275 //Punto medio entre 4250 y 4299

int  main(void)
{
while(1){
//Cambiamos el canal donde se eligiran los  valores  del  ADC
    sConfig.Channel = ADC_CHANNEL_4;
    sConfig.Rank = 1;
    sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
    HAL_ADC_ConfigChannel(&hadc1, &sConfig);


    HAL_ADC_Start_IT(&hadc1);
    HAL_Delay(1000);
    HAL_ADC_Stop_IT(&hadc1);
    }
}

void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
{
  if(hadc->Instance == ADC1)
  {
    case 4:
        //Obtenemos el valor del ADC y convertimos ese valor a grados centgrados.
        temperatureValue = HAL_ADC_GetValue(&hadc1);
        float  R = (4095.0/temperatureValue)-1.0;
        R = R0*R;
        temp = 1.0/(log(R/R0)/B+1/298.15) - 273.15;
        //Almacenamos en un vector de char para mandar al LCD.
        sprintf(temperature," %d",(int) temp);

    break;
  }
```

```
}

void HAL_TIM_OC_DelayElapsedCallback(TIM_HandleTypeDef *htim){
  if(htim11.Instance == htim->Instance){
    //Colocamos el cursor y enviamos la temperatura actual junto al icono creado.
    setCursor(12,0);
    enviarWrite(temperature,sizeof(temperature));
    write(5);

    //Si la temperatura supera el umbral, entonces se encendera la luz de aviso.
    if(temp > THRESHOLD_TEMPERATURE){
    HAL_GPIO_WritePin(GPIOA, greenTemperature_Pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOC, redTemperature_Pin, GPIO_PIN_SET);
    }
  }
}
```

Si la temperatura es mayor de 30ºC, entonces se activará el LED rojo de alarma de temperatrua, que no se desactivará aunque vuelva a bajar de 30ºC. Solo se volverá a poner en verde cuando pulsemos el botón de reset que hemos configurado para ello.

3. Sensor de Luminosidad

Para el sensor de luminosidad, haremos uso del ADC_CH4 conectado a **A4**. Para convertir el valor del ADC a Lux, utilizamos la fórmula siguiente:

$$Lux = e^{valorADC/265{,}8} \tag{3}$$

```
int main(void)
{
while(1){
//Cambiamos el canal donde se eligiran los valores del ADC
    sConfig.Channel = ADC_CHANNEL_8;
    sConfig.Rank = 1;
    sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
    HAL_ADC_ConfigChannel(&hadc1, &sConfig);

    HAL_ADC_Start_IT(&hadc1);
    HAL_Delay(1000);
    HAL_ADC_Stop_IT(&hadc1);
    }
}

void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
{
  if(hadc->Instance == ADC1)
  {
    case 8:
        //Obtenemos el valor del ADC y convertimos a Lux
        brValue = HAL_ADC_GetValue(&hadc1);
        lux = exp(brValue/265.8); //355.0 //resistencia

        //Ajustamos los valores para el LCD
        if(lux < 10000.0){
          int i = 2;
          while(i<7){
            luz[i] = ' ';
            i++;
          }
        }
        //Almacenamos en un vector de char para mandar al LCD
        sprintf(luz," %d",(int) lux);
      break;
  }
}

void HAL_TIM_OC_DelayElapsedCallback(TIM_HandleTypeDef *htim){
  if(htim11.Instance == htim->Instance){
    //Colocamos el cursor y enviamos la luminosidad actual junto al icono creado.
    setCursor(7,1);
    write(1); //Icono luz guardado en posicion 1
    enviarWrite(luz, sizeof(luz));
    enviarWrite(" L", sizeof(" L"));
  }
}
```

4. Sensor de Sonido/Ruido

En el sensor de sonido, usaremos el ADC_CH1, ubicado en **A1**, como entrada de éste. Su medida será en decibelios (dB).

```
#define THRESHOLD_NOISE 60
int main(void)
{
while(1){
//Cambiamos el canal donde se eligiran los valores del ADC
    sConfig.Channel = ADC_CHANNEL_1;
    sConfig.Rank = 1;
    sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
    HAL_ADC_ConfigChannel(&hadc1, &sConfig);

    HAL_ADC_Start_IT(&hadc1);
    HAL_Delay(1000);
    HAL_ADC_Stop_IT(&hadc1);
}

void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
{
  if(hadc->Instance == ADC1)
  {
    case 1:
    //Recogemos valor del ADC y convertimos a dB
        soundValue = HAL_ADC_GetValue(&hadc1);
        if(soundValue>0){
          db=10.0*log(soundValue/300.0)+20.0*log(soundValue/300.0);

          //Almacenamos en un vector de char para mandar al LCD
          sprintf(sonido, "%d", (int)db);
        }
    break;
  }
}

void HAL_TIM_OC_DelayElapsedCallback(TIM_HandleTypeDef *htim){
  if(htim11.Instance == htim->Instance){
  //Colocamos el cursor y enviamos la velocidad actual junto al icono creado
    setCursor(0,1);
    write(4); //Icono de sonido guardado en la posicion 4
    enviarWrite(sonido,sizeof(sonido));
    enviarWrite(" dB", sizeof(" dB"));

    //Si la potencia de sonido supera el umbral, entonces se enciende la luz de aviso
     if(db > THRESHOLD_NOISE){
      HAL_GPIO_WritePin(GPIOB, greenSound_Pin, GPIO_PIN_RESET);
      HAL_GPIO_WritePin(GPIOA, redSound_Pin, GPIO_PIN_SET);
    }
}
```

Si la potenica de sonido es mayor de 60 dB, entonces se activará el LED rojo de alarma de sonido, que no se desactivará aunque vuelva a bajar de 60 dB. Solo se volverá a poner en verde cuando pulsemos el botón de reset que hemos configurado para ello.

5. Sensor de Precipitación

Para el sensor de lluvia, usamos el **Touch** para simular la caída de una gota de lluvia. En este caso debemos configurar previamente el **PB5** con interrupciones, de manera que cuando lo pulsemos, sea como si hubiese detectado la gota.

Para ello debemos usaremos el mismo Callback que utilizaremos en el botón de resetear estados que haremos próximamente. Como estamos hablando del sensor de lluvia omitiré el código del botón anterior.

```
__disable_irq();
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin){
{
  if(GPIO_Pin == GPIO_PIN_5){
    lcd_azul();
  }
  __enable_irq();
}
```

Cuando se detecte una "gota", entonces el LED cambiará, durante un momento, a color azul.

Figura 2: LCD sin detectar gota de lluvia

Figura 3: LCD que detecta gota de lluvia

6. Sensor de Viento

Para el sensor de viento (anemómetro) teníamos la configuración que se entrega previamente junto al enunciado. Es una simulación que se realiza con el **potenciómetro**.

Está configurado mediante el **ADC1 IN0**, el **Timer 2** y el **PA5** como salida, además del **Timer 3** y el **PA6** como entrada.

El Timer 2 y el Timer 3 están configurados de la siguiente manera:

- **Prescaler**: 1120 - 1
- **Period**: 3000 - 1

Mediante el ADC1_IN0 recogemos el valor del potenciómetro que irá aumentando en un rango entre 75 y 1500, que guardamos en el CCR1 del Timer 2.

En el Timer 3, que tenemos como entrada, calcularemos la diferencia de tiempo entre los flancos de subida y de bajada para saber el tiempo en el que está activa la onda (valor 1), y de esta manera, obtener la velocidad.

Deberemos guardar el CCR1 antes y después de los flancos, por lo que tendremos el valor en el que se encuentra dentro del rango [75-1499] del que hemos hablado anteriormente.

Una vez lo tenemos, podremos obtener la velocidad a la que va la onda.

```
#define THRESHOLD_WIND 90
#define MAX_ANGLE 4096 //Angulo maximo potenciometro
int main(void)
{
while(1){
//Cambiamos el canal donde se eligiran los valores del ADC
    sConfig.Channel = ADC_CHANNEL_0;
    sConfig.Rank = 1;
    sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
    HAL_ADC_ConfigChannel(&hadc1, &sConfig);

    HAL_ADC_Start_IT(&hadc1);
    HAL_Delay(1000);
    HAL_ADC_Stop_IT(&hadc1);
}

void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
{
  if(hadc->Instance == ADC1)
  {
    case 0:
        HAL_TIM_PWM_Start_IT(&htim2, TIM_CHANNEL_1);
        HAL_TIM_IC_Start_IT(&htim3, TIM_CHANNEL_1);

        //Recogemos valor del ADC e introducimos en CCR1 despues de realizar la formula
        adcValue=HAL_ADC_GetValue(&hadc1);
        htim2.Instance->CCR1 = 75 + (adcValue*1425/MAX_ANGLE);
      break;
  }
}

void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim){
  int final, range;
```

```
34
35    int start = htim2.Instance->CCR1;
36    range = start - final;
37    speed = (range / 7.125) - 10;
38
39    //Ajustamos los valores para el LCD
40    if(speed < 10){
41      aux[2] = ' ';
42    }
43    //Almacenamos en un vector de char para mandar al LCD
44    sprintf(aux, "%d", speed);
45
46    if (htim3.Instance->CCR1==599){
47      htim3.Instance->CCR1=2999;
48    }else{
49      htim3.Instance->CCR1=599;
50    }
51    final = htim2.Instance->CCR1;
52    HAL_TIM_PWM_Stop_IT(&htim2, TIM_CHANNEL_1);
53    HAL_TIM_IC_Stop_IT(&htim3, TIM_CHANNEL_1);
54  }
55
56  void HAL_TIM_OC_DelayElapsedCallback(TIM_HandleTypeDef *htim){
57    if(htim11.Instance == htim->Instance){
58    //Colocamos el cursor y enviamos la velocidad actual junto al icono creado
59      setCursor(0,0);
60      write(3); //Icono de viento guardado en la posicion 3
61      enviarWrite(aux,sizeof(aux));
62      enviarWrite(" km/h", sizeof(" km/h"));
63
64      //Si la velocidad del viento supera el umbral, entonces se enciende la luz de aviso
65      if(speed > THRESHOLD_WIND){
66      HAL_GPIO_WritePin(GPIOA, greenWind_Pin, GPIO_PIN_RESET);
67      HAL_GPIO_WritePin(GPIOB, redWind_Pin, GPIO_PIN_SET);
68      }
69  }
```

Si la velocidad es mayor de 90 km/h, entonces se activará el LED rojo de alarma de viento, que no se desactivará aunque vuelva a bajar de 90 km/h. Solo se volverá a poner en verde cuando pulsemos el botón de reset que hemos configurado para ello.

7. Alarmas

Los sensores de temperatura, viento y ruido harán saltar una alarma si sobrepasan cierto umbral.

- **Temperatura**: 30 ºC
- **Viento**: 90 km/h
- **Ruido**: 60 dB

Cada uno de ellos tendrá dos LEDs, uno de color verde y el otro de color rojo. Mientras que estos umbrales no se sobrepasen entonces el LED verde permanecerá encendido mientras que el LED rojo permanecerá apagado.

Cuando uno de ellos sobrepase el umbral, entonces el LED verde se apagará y se encenderá el LED rojo sin posibilidad de retorno al estado anterior, a no ser que se pulse el botón de reseteo de estados que nosotros controlaremos.

Para configurar el botón de resetear los estados lo haremos de la siguiente manera, si detecta que se está pulsando el botón **(PA10)** que hemos configurado previamente, entonces cambiamos los tres estados de los LEDs.

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin){
    __disable_irq();
if(GPIO_Pin == GPIO_PIN_10){
    //Restart wind alarms
    HAL_GPIO_WritePin(GPIOA, greenWind_Pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOB, redWind_Pin, GPIO_PIN_RESET);

    //Restart temperature alarms
    HAL_GPIO_WritePin(GPIOA, greenTemperature_Pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOC, redTemperature_Pin, GPIO_PIN_RESET);

    //Restart sound alarms
    HAL_GPIO_WritePin(GPIOB, greenSound_Pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOA, redSound_Pin, GPIO_PIN_RESET);

    }
    __enable_irq();
}
```

8. LCD

Para el LCD haremos uso de los recursos que aplicamos en el Laboratorio 7. Definiremos los siguientes valores, para el posterior uso de ellos.

```
// Device I2C Arress

#define LCD_ADDRESS (0x3e<<1)
#define RGB_ADDRESS (0x62<<1)
#define LCD_5x8DOTS 0x00

// color  define
#define WHITE         0
#define RED           1
#define GREEN         2
#define BLUE          3

#define REG_RED       0x04        // pwm2
#define REG_GREEN     0x03        // pwm1
#define REG_BLUE      0x02        // pwm0

#define REG_MODE1     0x00
#define REG_MODE2     0x01
#define REG_OUTPUT 0x08

// commands
#define LCD_CLEARDISPLAY 0x01
#define LCD_RETURNHOME 0x02
#define LCD_ENTRYMODESET 0x04
#define LCD_DISPLAYCONTROL 0x08
#define LCD_CURSORSHIFT 0x10
#define LCD_FUNCTIONSET 0x20
#define LCD_SETCGRAMADDR 0x40
#define LCD_SETDDRAMADDR 0x80

// flags  for  display  entry  mode
#define LCD_ENTRYRIGHT 0x00
#define LCD_ENTRYLEFT 0x02
#define LCD_ENTRYSHIFTINCREMENT 0x01
#define LCD_ENTRYSHIFTDECREMENT 0x00

// flags  for  display  on/off  control
#define LCD_DISPLAYON 0x04
#define LCD_DISPLAYOFF 0x00
#define LCD_CURSORON 0x02
#define LCD_CURSOROFF 0x00
#define LCD_BLINKON 0x01
#define LCD_BLINKOFF 0x00

// flags  for  display/cursor  shift
#define LCD_DISPLAYMOVE 0x08
#define LCD_CURSORMOVE 0x00
#define LCD_MOVERIGHT 0x04
#define LCD_MOVELEFT 0x00

// flags  for  function  set
#define LCD_8BITMODE 0x10
#define LCD_4BITMODE 0x00
```

```
#define LCD_2LINE 0x08
#define LCD_1LINE 0x00
#define LCD_5x10DOTS 0x04
```

El método con el que inicializamos será el begin, mientras que mediante enviarWrite(texto, tamaño) podemos mostrar texto por el LCD, y para cambiar el color del fondo mediante setRGB(rojo, verde, azul).

```

void rgb_lcd_begin(uint8_t cols, uint8_t lines, uint8_t dotsize) {
    if (lines > 1) {
        _displayfunction |= LCD_2LINE;
    }
    _numlines = lines;
    _currline = 0;
    // for some 1 line displays you can select a 10 pixel high font
    if ((dotsize != 0) && (lines == 1)) {
        _displayfunction |= LCD_5x10DOTS;
    }
    HAL_Delay(50);
    // Send function set command sequence
    command(LCD_FUNCTIONSET | _displayfunction);
    HAL_Delay(5); // wait more than 4.1ms
    // second try
    command(LCD_FUNCTIONSET | _displayfunction);
    HAL_Delay(1);
    // third go
    command(LCD_FUNCTIONSET | _displayfunction);
    // finally, set # lines, font size, etc.
    command(LCD_FUNCTIONSET | _displayfunction);
    // turn the display on with no cursor or blinking default
    _displaycontrol = LCD_DISPLAYON | LCD_CURSOROFF | LCD_BLINKOFF;
    display();
    // clear it off
    clear();
    // Initialize to default text direction (for romance languages)
    _displaymode = LCD_ENTRYLEFT | LCD_ENTRYSHIFTDECREMENT;
    // set the entry mode
    command(LCD_ENTRYMODESET | _displaymode);

    // backlight init
    setReg(REG_MODE1, 0);
    // set LEDs controllable by both PWM and GRPPWM registers
    setReg(REG_OUTPUT, 0xFF);
    // set MODE2 values
    // 0010 0000 -> 0x20 (DMBLNK to 1, ie blinky mode)
    setReg(REG_MODE2, 0x20);
}
void command(uint8_t value) {
    unsigned char dta[2] = {0x80, value};
    HAL_I2C_Master_Transmit(&hi2c1, LCD_ADDRESS, dta, sizeof(dta), 100);
}
void clear() {
    command(LCD_CLEARDISPLAY); // clear display, set cursor position to zero
    HAL_Delay(2);
}
void display() {
```

```
50      _displaycontrol |= LCD_DISPLAYON;
51      command(LCD_DISPLAYCONTROL | _displaycontrol);
52  }
53  void setReg(unsigned char addr, unsigned char dta) {
54    unsigned char data[2] = {addr, dta};
55    HAL_I2C_Master_Transmit(&hi2c1, RGB_ADDRESS, data, 2, 100);
56  }
57  void setRGB(unsigned char r, unsigned char g, unsigned char b) {
58      setReg(REG_RED, r);
59      setReg(REG_GREEN, g);
60      setReg(REG_BLUE, b);
61  }
62  void write(uint8_t value) {
63
64      unsigned char dta[2] = {0x40, value};
65      HAL_I2C_Master_Transmit(&hi2c1, LCD_ADDRESS, dta, 2, 100);
66  }
67  void enviarWrite(unsigned char* dta, unsigned char len) {
68      for (int i = 0; i < len - 1; i++) {
69        write(dta[i]);
70      }
71  void setCursor(uint8_t col, uint8_t row) {
72      col = (row == 0 ? col | 0x80 : col | 0xc0);
73      unsigned char dta[2] = {0x80, col};
74      HAL_I2C_Master_Transmit(&hi2c1, LCD_ADDRESS, dta, sizeof(dta), 100);
75  }
76  void createChar(uint8_t location, uint8_t charmap[]) {
77      location &= 0x7; // we only have 8 locations 0-7
78      command(LCD_SETCGRAMADDR | (location << 3));
79      unsigned char dta[9];
80      dta[0] = 0x40;
81      for (int i = 0; i < 8; i++) {
82          dta[i + 1] = charmap[i];
83      }
84    enviarWrite(dta, sizeof(dta));
85  }
86  void lcd_rojo(){
87      setRGB(255, 0, 0);
88  }
89  void lcd_azul(){
90      setRGB(0, 0, 255);
91  }
92  void lcd_verde(){
93      setRGB(0, 255, 0);
94  }
```

Hemos creado una estrutura de unsigned char llamada BYTE, para realizar los iconos del LCD.

9. Fritzting

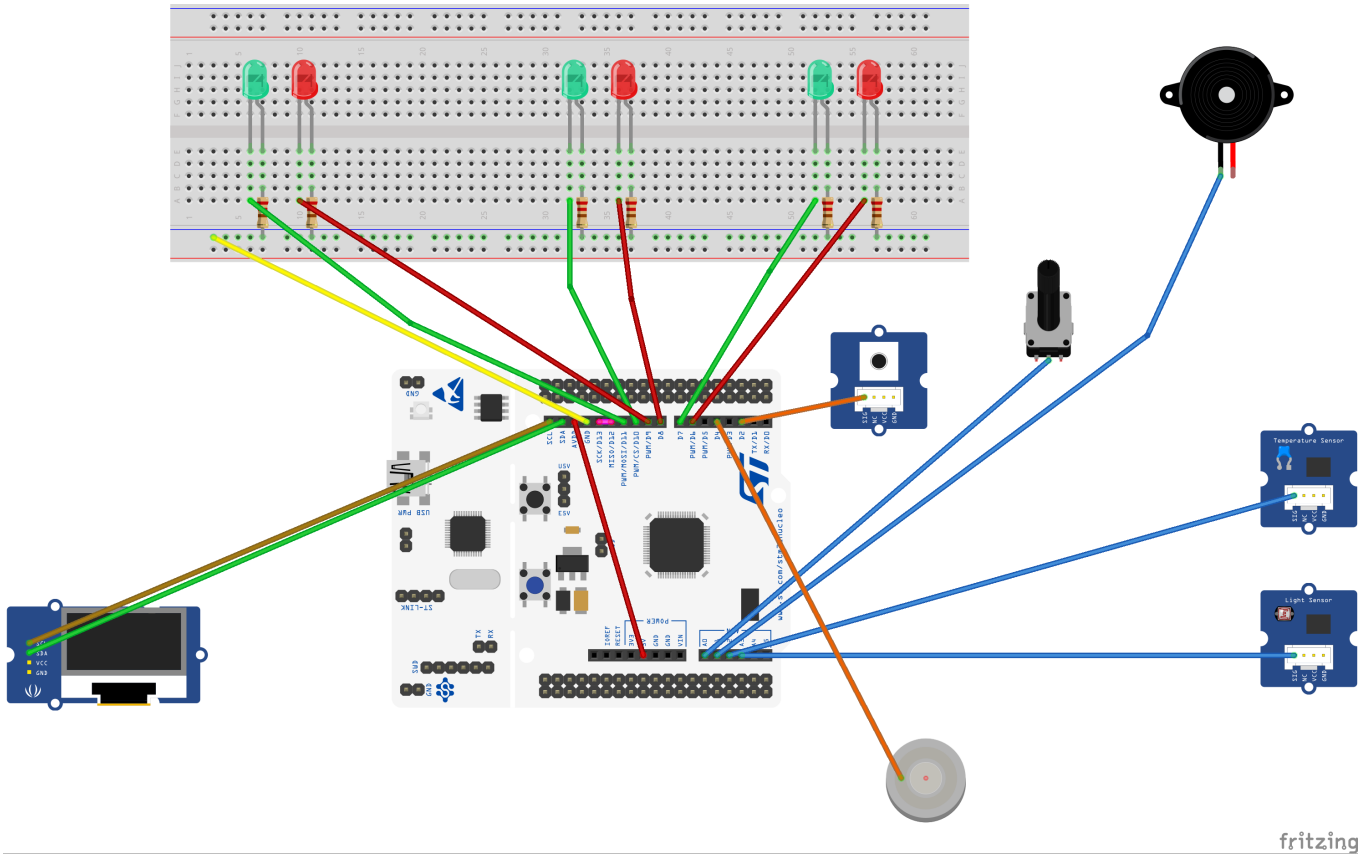

Figura 4: Esquema de conexiones

Usamos una Groove, que no está en Frizting, por lo que hacemos conexiones manuales. En los botones, obviamos el GND, VCC en el dibujo.