# MEMORIA INSTALACIÓN PROTOTIPO PARA ADQUISICIÓN DE DATOS DE SENSORES CON ARDUINO 33IOT

Realizado por: José Juan Quintana Hernández
Área de Ingeniería de Sistemas y Automática
Universidad de Las Palmas de Gran Canaria (ULPGC)

Diciembre de 2023

CONTENIDO

**INTRODUCCIÓN** ..... 3
1. INTALACIÓN DEL DISPLAY ..... 4
2. SENSOR DIGITAL DE TEMPERATURA DEL AGUA DS18B20 ..... 7
3. SENSOR DE TEMPERATURA Y HUMEDAD DEL AIRE ..... 9
4. SENSOR DE LUMINOSIDAD ..... 9
5. DATALOGGER Y RELOJ ..... 12
6. PUESTA EN HORA DEL RELOJ ..... 18

# INTRODUCCIÓN

Este documento ha sido elaborado en el marco del proyecto "R+D+i TOWARDS AQUAPONIC DEVELOPMENT IN THE UP ISLANDS AND THE CIRCULAR ECONOMY. INTERREGIONAL FORWARD CHALLENGES", con acrónimo ISLANDAP ADVANCED (MAC2/1.1a/299), financiado por el Programa Operativo de Cooperación Territorial Madeira-Açores-Canarias (POMAC) 2014- 2020 de la Comisión Europea. Como complemento tangible a la actividad de ejecución 2.2.2 "Desarrollos experimentales con tecnologías blandas".

Una de las tareas del proyecto ISLANDAP ADVANDED, es la de divulgación y comunicación de las actividades de ejecución, para ello se ha optado por instalar en tres colegios de primaria prototipos acuapónicos para enseñar su funcionamiento y mantenimiento en el tiempo con el objetivo de cuidar y observar el crecimiento de las plantas y peces.

Para aumentar la implicación de los estudiantes, se ha optado por monitorizar algunas variables, y que estos las registren en un cuaderno a determinadas horas de día. Para ello se pensó en diseñar un sencillo dispositivo para monitorizar algunas de estas variables con las siguientes características.

En primer lugar, el dispositivo debería ser robusto y soportar en la medida de lo posible los accidentes que se pudieran producir, tanto por caídas como por agua.

En segundo lugar, no debería ser un equipo relativamente caro, los sensores deberían de tener un bajo mantenimiento, debería de poder accederse fácilmente a las medidas de éstos y finalmente la tensión de trabajo debería ser segura.

Con estos condicionantes, se optó por colocar un sensor de temperatura de agua de la pecera que pueda ser introducido en ella, un sensor de temperatura y humedad del aire y finalmente un sensor de luminosidad. Para el control de todo esto se optó por utilizar un controlador de bajo costo, pero que permitiera conexiones con wifi y bluetooth para futuras conexiones de los estudiantes, y además fácilmente programable, por ello nos decantamos por usar un Arduino Nano 33IOT, que está pensado para poder integrase fácilmente a la internet de las cosas IOT. Para mostrar los datos se optó por un display de 2 líneas de texto y de 16 caracteres por línea.

Además, con motivos de investigación también se consideró interesante poder monitorizar el estado del sistema durante un periodo de tiempo largo, por lo que se necesitó poner un lector de tarjetas microSD para almacenar los datos junto con la hora y fecha en un soporte no volátil.

Teniendo en cuenta que para la toma de datos es necesario disponer de la hora y fecha correctas, y que el Arduino cada vez que se queda sin alimentación pierde su hora y se pone a su hora de fábrica, se hace necesario disponer de un reloj alimentado por una pila para que mantenga la hora del sistema ante caídas de la tensión de alimentación.

Por tanto, resumiendo, los elementos que va a tener nuestro sistema van a ser los siguientes, incluidos los modelos elegidos:

1. Arduino Nano 33IOT link
2. Display de 2 líneas y 16 caracteres por línea, (Gravity: I2C 16x2 Arduino LCD with RGB Backlight Display V2.0, link)
3. Sensor de temperatura del agua (Dallas semiconductor, DS18B20 1-wire digital thermometer, link)
4. Sensor de temperatura y humedad ambiente (Sensor de humedad y temperatura AM2301B link).
5. Sensor de luminosidad (M5StickC Sensor de Luz Ambiente link)
6. Reloj y lector de tarjetas microSD (Adalogger FeatherWing - RTC + SD Add-on for all feather boards link)

Una vez decididos y comprados los elementos del sistema, el siguiente paso es realizar un prototipo en una placa protoboard para comprobar el correcto funcionamiento del sistema. En el resto del documento se describirá con esquemas realizados con **Fritzing** y modificados con el software libre **Inkscape** como conectar los distintos elementos.

El procedimiento utilizado ha sido el siguiente: primero se ha instalado el display y se ha probado su correcto funcionamiento con el programa **Arduino IDE** que permite programar el Arduino y transferirle el programa. Sobre esta primera instalación se ha agregado el sensor de temperatura del agua y se ha probado, a continuación, se han añadido los sensores de temperatura y humedad del aire y el de luminosidad y se han probado y finalmente se ha agregado la tarjeta de reloj y lectora de tarjetas de microSD y se ha probado el sistema completo.

La ventaja de realizar el montaje de esta manera es que se va probando el montaje por fases y también que no hay un esquema con todos los elementos conectados a la vez que hacen el sistema muy difícil de montar y de depurar.

Una vez dicho lo anterior, manos a la obra.

## 1. Instalación del display

El Arduino dispone de varios buses para conectar distintos elementos de forma digital. La idea de un bus es similar a la de un cable de red en un ordenador, en la que un microcontrolador y sus sensores comparten varios cables por los que se realiza la comunicación entre estos. Para la conexión del display, el sensor de temperatura y humedad ambiental, el de luminosidad y el reloj al microcontrolador se ha utilizado el bus **I2C** en la que las bornas **SDA** y **SCL** de todos estos dispositivos se conectarán entre sí. Este bus es capaz de manejar 112 dispositivos en un mismo bus información I2C.

Una vez conectados todos los dispositivos al bus, cada sensor tendrá una dirección única y el microcontrolador a voluntad enviará una petición de información a una dirección determinada y el dispositivo en cuestión le enviará la información requerida. En esta topología el microcontrolador es el que siempre toma la iniciativa y los sensores siempre están a la escucha. Como dato comentar que la velocidad de transmisión es de 100 kHz en modo estándar y de 400 kHz en modo de alta velocidad, lo cual es más que suficiente para la implementación que se va a llevar a cabo.

En la figura 1, se muestra el conexionado del Arduino con el display. El Arduino genera los 3.3 V para la alimentación de los sensores y están accesibles a todos los dispositivos por las dos líneas inferiores de la placa protoboard. En la fase de diseño, se tuvo que escoger un display alimentado a 3.3V, ya que en un principio la mayor parte de estos displays de 2 líneas y 16 caracteres funcionan a 5V.

Tal como se comentó al principio de este apartado, los terminales **SDA** y **SCL** del Arduino y del display hay que conectarlos entre sí. Pero debido a que a estos terminales están también conectados los otros dispositivos **I2C**, se ha optado por conectarlos todos a los **5 pines** marcados con **SCL** y **SDA** en la figura.

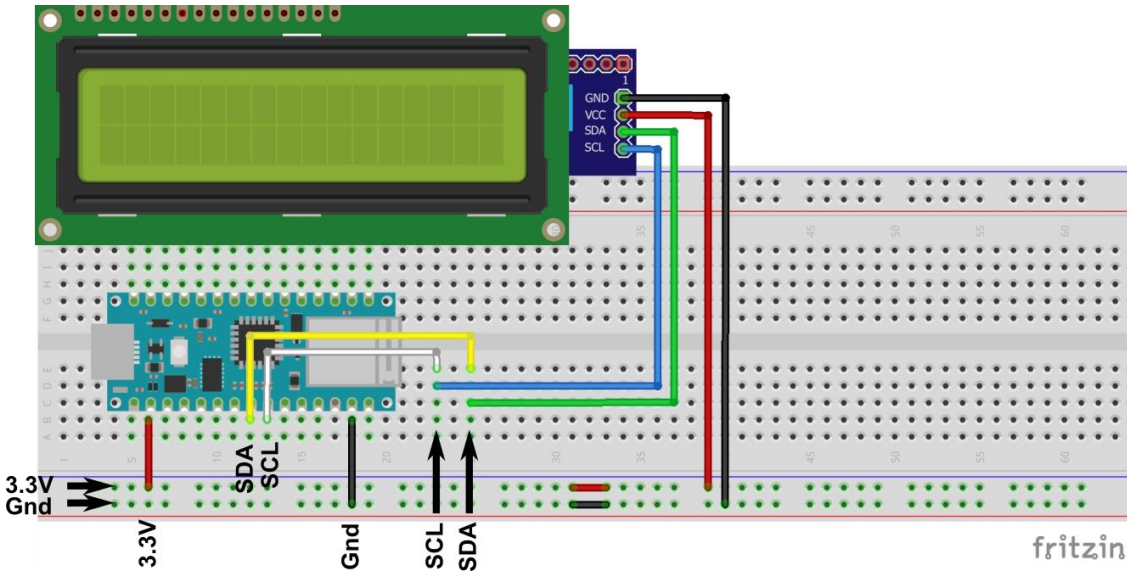


*Figura 1.- Conexión del display mediante bus I2C*

Una vez conectado el display al Arduino, el siguiente paso es comprobar si el cableado se ha realizado correctamente.

Para ello se debe descargar, mediante el programa para la programación de arduinos **Arduino IDE**, el **programa 1** en el Arduino indicándole el tipo de microcontrolador Arduino 33 IOT y el puerto de comunicaciones.

Cuando le demos al botón **upload**, el programa se compilará y detectará si tiene todas las librerías. Probablemente aparezca el mensaje de que el fichero **DFRobot_RGBLCD1602.h** no se encuentra, para añadirlo se debe ir a la pestaña **Tool** y seleccionar **manage libraries** y poner **DFRobot_RGBLCD1602**, aparecerá la librería y se debe **instalar**. En los siguientes apartados se deberá de proceder igual con todas las librerías no encontradas.

Esto permitirá compilar el programa correctamente y volcarlo al Arduino.

Si todo va bien, el **display** se encenderá y aparecerá el mensaje **Hello World** y el número de segundos.

Si esto es así podemos pasar al siguiente paso, en caso contrario, lo más probable es que haya un problema de cableado.

Programa 1.- Programa de Arduino para la prueba del montaje de la figura 1.

```
// Librería para el manejo del display
#include "DFRobot_RGBLCD1602.h"

// Se puede seleccionar el color del texto en el display
const int colorR = 128;
const int colorG = 128;
const int colorB = 128;

/*
Change the RGBaddr value based on the hardware version
-----------------------------------------
    Moudule      | Version| RGBAddr|
-----------------------------------------
 LCD1602 Module    | V1.0  | 0x60   |
-----------------------------------------
 LCD1602 Module    | V1.1  | 0x6B   |
-----------------------------------------
 LCD1602 RGB Module | V1.0  | 0x60   |
-----------------------------------------
 LCD1602 RGB Module | V2.0  | 0x2D   |
-----------------------------------------
*/

DFRobot_RGBLCD1602 lcd(/*RGBAddr*/0x2D ,/*lcdCols*/16,/*lcdRows*/2);  //16 characters and 2 lines of show

void setup() {

  // iniciamos el LCD
  lcd.init();
  // Definimos los colores
  lcd.setRGB(colorR, colorG, colorB);

  // Print a message to the LCD para pruebas
  lcd.print("hello, world!");
}
void loop() {

  lcd.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd.print(millis()/1000);

 delay(100);
}
```

Fin del código

## 2. Sensor digital de temperatura del agua DS18B20

Una vez funcione correctamente el display, lo siguiente es **añadir al circuito anterior** el sensor de temperatura del agua según la figura 2, y comprobar que la lectura es correcta en el display.

Este sensor utiliza para enviar los datos de la temperatura el bus de comunicaciones 1-wire que permite conectar varios sensores a una misma entrada (en este caso la entrada D2). Este bus necesita que la salida del sensor se conecta en pull-up mediante una resistencia de $4{,}6k\Omega$ a la tensión de alimentación.

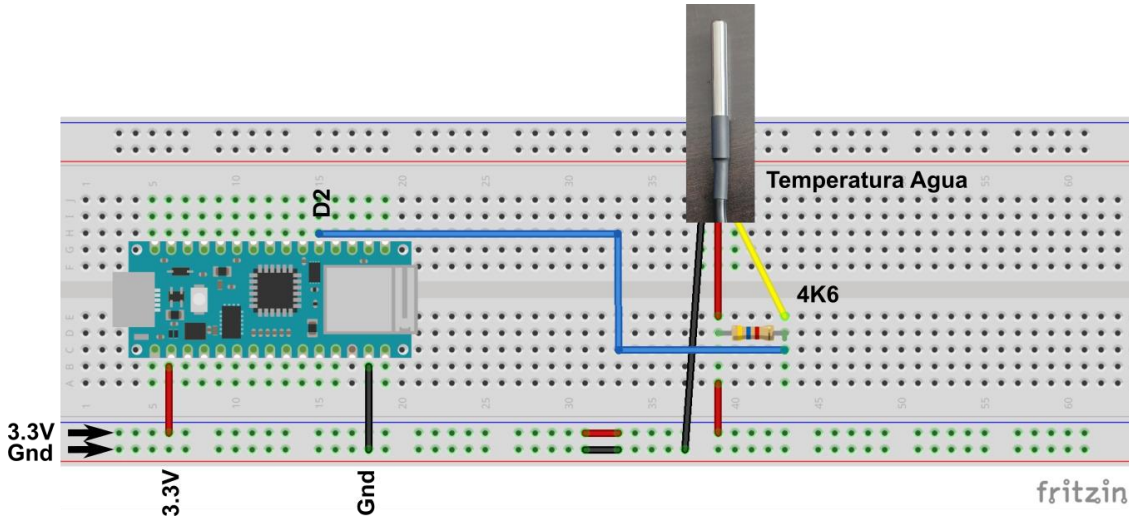


*Figura 2.- Conexión del sensor de temperatura del agua*

Para la prueba del correcto funcionamiento del sensor de temperatura, se debe copiar el código del programa 2, instalar las librerías necesarias y subir el programa al Arduino, si todo va correctamente, se mostrará en el display **H2O y la temperatura**.

Programa 2.- Programa de Arduino para la prueba del montaje de la figura 1.

```
// Librería para el manejo del display
#include "DFRobot_RGBLCD1602.h"

// Se puede seleccionar el color del texto en el display
const int colorR = 128;
const int colorG = 128;
const int colorB = 128;

/*
Change the RGBaddr value based on the hardware version
-----------------------------------------
    Moudule      | Version| RGBAddr|
-----------------------------------------
 LCD1602 Module    | V1.0  | 0x60   |
-----------------------------------------
 LCD1602 Module    | V1.1  | 0x6B   |
-----------------------------------------
```

```
 LCD1602 RGB Module  |  V1.0  | 0x60   |
-----------------------------------------
 LCD1602 RGB Module  |  V2.0  | 0x2D   |
-----------------------------------------
*/

DFRobot_RGBLCD1602 lcd(/*RGBAddr*/0x2D ,/*lcdCols*/16,/*lcdRows*/2);  //16 characters and 2 lines of show

// Para la instalación del sensor de temperatura del agua protocolo 1-wire
#include <OneWire.h>
#include <DallasTemperature.h>
OneWire ourWire1(2);              //Se establece el pin 2  como bus OneWire
DallasTemperature sensors1(&ourWire1); //Se declara una variable u objeto para nuestro sensor1
void setup() {
  // Se inicia el sensor de temperatura del agua
  sensors1.begin();   //Se inicia el sensor 1

  Serial.begin(115200);
  Serial.println("Adafruit AHT10/AHT20 demo!");


  // iniciamos el LCD
  lcd.init();
  // Definimos los colores
  lcd.setRGB(colorR, colorG, colorB);
}
void loop() {

  // Se lee la temperatura del agua y se guarda en un float
  sensors1.requestTemperatures();   //Se envía el comando para leer la temperatura
  float temp1= sensors1.getTempCByIndex(0); //Se obtiene la temperatura en ºC del sensor 1

  // Segunda fila del display
  // Ubicamos el cursor en la primera posición(columna:0) de la segunda línea(fila:1)
  lcd.setCursor(0, 1);
  lcd.print("H2O ");
  lcd.print(int(temp1));

// Envío de datos por el puerto serie para comprobaciones
  Serial.print(temp1);
  Serial.println("ºC, ");

  delay(100);
}
```

Fin del código

## 3. Sensor de temperatura y humedad del aire

La conexión del sensor se muestra en la figura 3 y debe ser añadida a la realizada en el apartado anterior. Para conectar este sensor sólo hace falta conectar la **alimentación** y las bornas **SDA** y **SCL** en las columnas **SDA** y **SCL**.

Debido a la simplicidad de esta conexión, se probará el código junto con el del sensor de luminosidad mostrado en la siguiente sección.

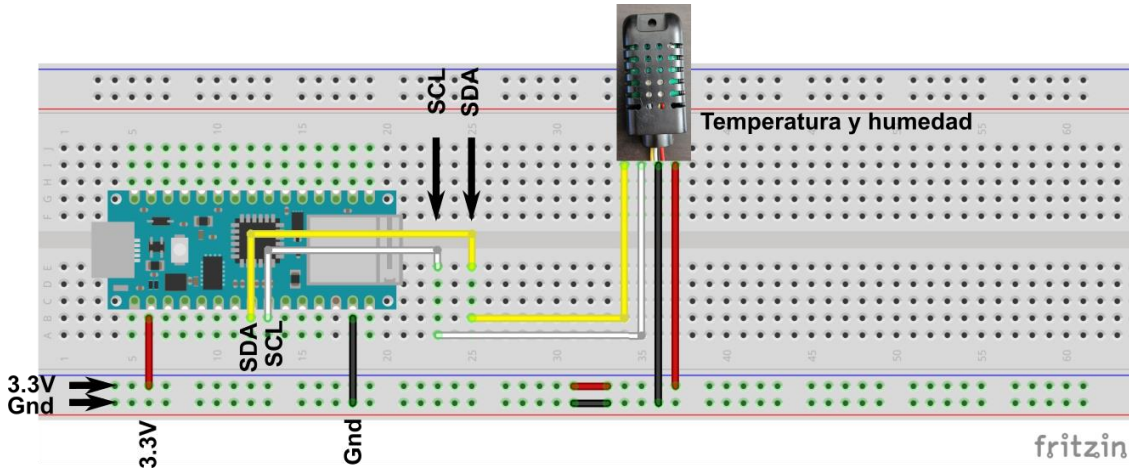


*Figura 3.- Conexión del sensor de temperatura y humedad*

## 4. Sensor de luminosidad

Este sensor también se conecta a través del bus I2c y su conexionado se puede ver en la figura 4.

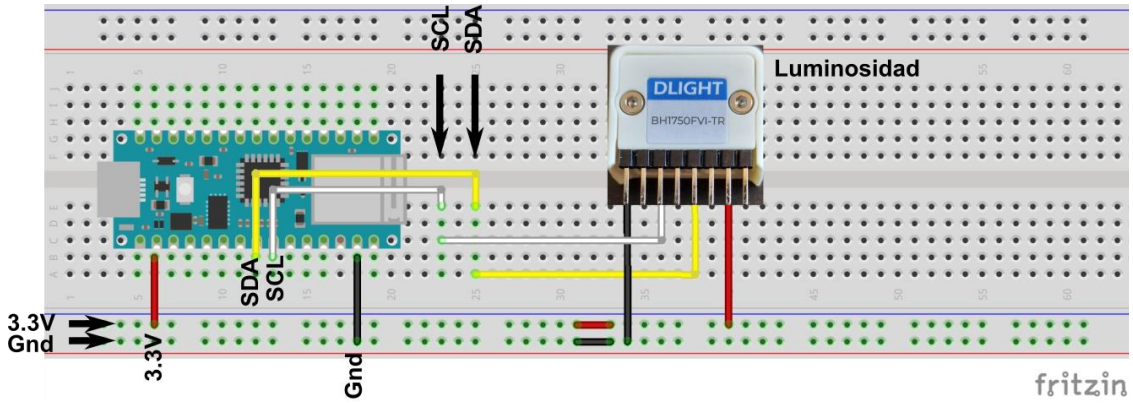


*Figura 4.- Conexión del sensor de luminosidad*

Para la prueba del correcto funcionamiento del sensor de temperatura y humedad y de luminosidad, se debe copiar el código del **programa 3**, instalar las librerías necesarias y subir el programa al Arduino, si todo va correctamente, se mostrará en el display las dos temperaturas, la humedad y la luminosidad.

Programa 3.- Programa de Arduino para la prueba del montaje de las figuras 3 y 4.

```
// Librería para el manejo del display
#include "DFRobot_RGBLCD1602.h"

// Se puede seleccionar el color del texto en el display
const int colorR = 128;
const int colorG = 128;
const int colorB = 128;

/*
Change the RGBaddr value based on the hardware version
-----------------------------------------
    Moudule      | Version| RGBAddr|
-----------------------------------------
 LCD1602 Module    |  V1.0  | 0x60   |
-----------------------------------------
 LCD1602 Module    |  V1.1  | 0x6B   |
-----------------------------------------
 LCD1602 RGB Module  |  V1.0  | 0x60   |
-----------------------------------------
 LCD1602 RGB Module  |  V2.0  | 0x2D   |
-----------------------------------------
*/

DFRobot_RGBLCD1602 lcd(/*RGBAddr*/0x2D ,/*lcdCols*/16,/*lcdRows*/2);  //16 characters and 2 lines of show

// Para la instalación del sensor de temperatura del agua protocolo 1-wire
#include <OneWire.h>
#include <DallasTemperature.h>
OneWire ourWire1(2);              //Se establece el pin 2  como bus OneWire
DallasTemperature sensors1(&ourWire1); //Se declara una variable u objeto para nuestro sensor1


// Sensor de humedad y temperatura
#include <Adafruit_AHTX0.h>
Adafruit_AHTX0 aht;

// Sensor de luminosidad
#include <BH1750.h>
BH1750 sensor_luz;
float lux;


void setup() {
  // Se inicia el sensor de temperatura del agua
```

```
 sensors1.begin();   //Se inicia el sensor 1

 Serial.begin(115200);
 Serial.println("Adafruit AHT10/AHT20 demo!");

 if (! aht.begin()) {
 Serial.println("Could not find AHT? Check wiring");
 while (1) delay(10);
 }
  // iniciamos el LCD
  lcd.init();
  // Definimos los colores
  lcd.setRGB(colorR, colorG, colorB);

  Serial.println("Sensor luminosidadbegin.....");
  sensor_luz.begin();

}
void loop() {
  // Lectura del sensor de luminosidad
  lux = sensor_luz.readLightLevel();

  // Lectura de datos del sensor de temperatura y humedad
  sensors_event_t humidity, temp;
  aht.getEvent(&humidity, &temp);// populate temp and humidity objects with fresh data

  // Se lee la temperatura del agua y se guarda en un float
  sensors1.requestTemperatures();   //Se envía el comando para leer la temperatura
  float temp1= sensors1.getTempCByIndex(0); //Se obtiene la temperatura en ºC del sensor 1


  // Primera fila del display
  // Ubicamos el cursor en la primera posición(columna:0) de la primera línea(fila:0)
  lcd.setCursor(0, 0);
  lcd.print("Air ");
  lcd.print(int(temp.temperature)); // Ponemos int para quitar los decimales del display
  lcd.print("C/Hum  ");
  lcd.print(int(humidity.relative_humidity));
  lcd.print("%");

  // Segunda fila del display
  // Ubicamos el cursor en la primera posición(columna:0) de la segunda línea(fila:1)
  lcd.setCursor(0, 1);
  lcd.print("H2O ");
  lcd.print(int(temp1));
  lcd.print("C/Lux ");
  lcd.print(lux);
```

```
// Envío de datos por el puerto serie para comprobaciones
  Serial.print(temp1);
  Serial.print("ºC, ");
  Serial.print(temp.temperature);
  Serial.print("ºC, ");
  Serial.print(humidity.relative_humidity);
  Serial.print("%, ");
  Serial.print(lux);
  Serial.println("lux. ");
  delay(100);
}
```

Fin del código

## 5. Datalogger y reloj

Como se comentó anteriormente, cada vez que el Arduino se queda sin tensión pierde su hora y se pone a la hora de fábrica, lo que hace que al guardar los datos estos no tengan sentido. Para corregir esto se debe poner un reloj con una pila de respaldo tipo CR1220.

Por otra parte, el Arduino no tiene una memoria RAM muy elevada y si se desea almacenar datos, hay que buscar algún medio para ello. Un medio muy eficaz es utilizar un lector de tarjetas microSD y utilizar una de estas tarjetas. En este proyecto se ha utilizado una tarjeta microSD de 32Gb que permite una captura de datos de más de 10 años tomando los datos cada 10 segundos.

Estas dos tareas se llevan a cabo mediante la placa mostrada en la figura 5 que dispone de dos elementos independientes por una parte un **reloj** conectado al Arduino con el bus **I2C** y por otra el **lector de tarjetas** microSD a través del del bus **SPI**.

La forma de conexión de esta placa al Arduino se muestra en la figura 5.

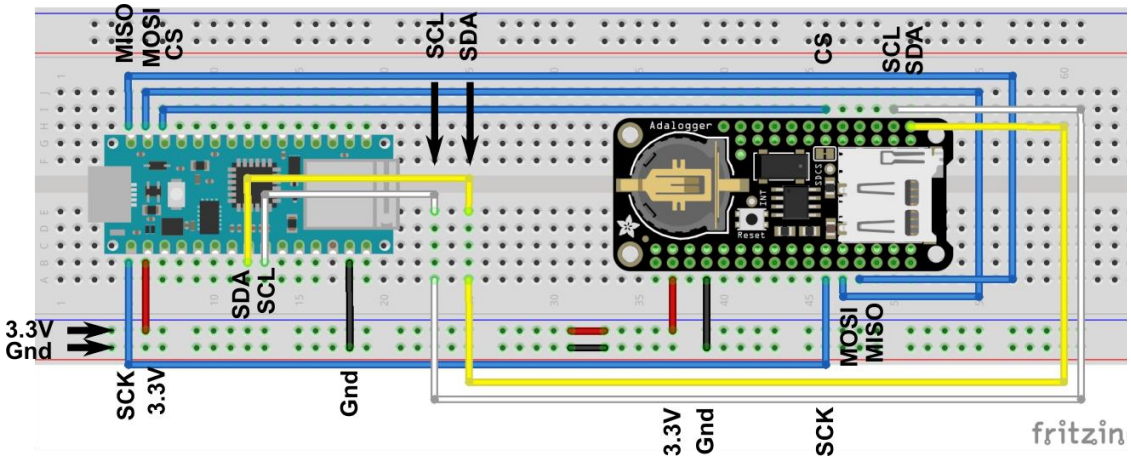


*Figura 5.- Conexión del reloj y lector de tarjetas*

Para la prueba del correcto funcionamiento es necesario que la tarjeta de memoria esté insertada.

Se debe copiar el código del programa 4, instalar las librerías necesarias y subir el programa al Arduino, si todo va correctamente, se mostrará en el display las dos temperaturas, la humedad y la luminosidad. Para ver la hora se debe activar en el **Arduino IDE** en la pestaña **Tool** – **Serial monitor**, se deberá ver la hora y los valores de las temperaturas, la humedad y la luminosidad y cada 10 segundos la información enviada al fichero separada por puntos y comas con el formato:

Año; mes; día; Hora; Minuto; segundo; Tra. Agua, Tra Aire; Humedad; Luminosidad.

Programa 4.- Programa de Arduino para la prueba del montaje final.

```
// Librería para el manejo del display bus I2C
#include "DFRobot_RGBLCD1602.h"

// Se puede seleccionar el color del texto en el display
const int colorR = 128;
const int colorG = 128;
const int colorB = 128;

/*
Change the RGBaddr value based on the hardware version
-----------------------------------------
    Moudule       | Version| RGBAddr|
-----------------------------------------
 LCD1602 Module     | V1.0  | 0x60   |
-----------------------------------------
 LCD1602 Module     | V1.1  | 0x6B   |
-----------------------------------------
 LCD1602 RGB Module  | V1.0  | 0x60   |
-----------------------------------------
 LCD1602 RGB Module  | V2.0  | 0x2D   |
-----------------------------------------
*/

DFRobot_RGBLCD1602 lcd(/*RGBAddr*/0x2D ,/*lcdCols*/16,/*lcdRows*/2);  //16 characters and 2 lines of show

// Sensor de temperatura del agua protocolo 1-wire
#include <OneWire.h>
#include <DallasTemperature.h>
OneWire ourWire1(2);              //Se establece el pin 2  como bus OneWire
DallasTemperature sensors1(&ourWire1); //Se declara una variable u objeto para nuestro sensor1

// Sensor de humedad y temperatura bus I2C
#include <Adafruit_AHTX0.h>
Adafruit_AHTX0 aht;

// Sensor de luminosidad Bus I2C
#include <BH1750.h>
```

```
BH1750 sensor_luz;
float lux;

// Parte dedicada al reloj bus I2C
// Real time clock
// Date and time functions using a PCF8523 RTC connected via I2C and Wire lib
#include "RTClib.h"

RTC_PCF8523 rtc;
char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
"Saturday"};

// Lector de tarjeras microSD a través del bus SPI
/*
 SD card datalogger

 SD card attached to SPI bus as follows:
 ** MOSI - pin 11
 ** MISO - pin 12
 ** CLK - pin 13
 ** CS - pin 10 (for Arduino nano 33iot)

*/
#include <SPI.h>
#include <SD.h>
const int chipSelect = 10;

// Zona para definir cada cuanto se guardan los datos
unsigned long previousMillis = 0;  // Se almacena el último insante en que se guardó el dato
const long interval = 10000;  // Intervalo para guardar los datos en la tarjeta microSD en ms



void setup() {
 // Se inicia el sensor de temperatura del agua
 sensors1.begin();   //Se inicia el sensor 1

 Serial.begin(115200);
 Serial.println("Adafruit AHT10/AHT20 demo!");

 if (! aht.begin()) {
 Serial.println("Could not find AHT? Check wiring");
 while (1) delay(10);
 }

 // iniciamos el LCD
 lcd.init();
```

```
// Definimos los colores
lcd.setRGB(colorR, colorG, colorB);

// Iniciamos el sensor de luminosidad
Serial.println("Sensor luminosidadbegin.....");
sensor_luz.begin();

// inicializar el reloj de respaldo
if (! rtc.begin()) {
  Serial.println("Couldn't find RTC");
  Serial.flush();
  while (1) delay(10);
}

if (! rtc.initialized() || rtc.lostPower()) {
  Serial.println("RTC is NOT initialized, let's set the time!");
  // When time needs to be set on a new device, or after a power loss, the
  // following line sets the RTC to the date & time this sketch was compiled
  rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  // This line sets the RTC with an explicit date & time, for example to set
  // January 21, 2014 at 3am you would call:
  // rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));
  //
  // Note: allow 2 seconds after inserting battery or applying external power
  // without battery before calling adjust(). This gives the PCF8523's
  // crystal oscillator time to stabilize. If you call adjust() very quickly
  // after the RTC is powered, lostPower() may still return true.
}

// When time needs to be re-set on a previously configured device, the
// following line sets the RTC to the date & time this sketch was compiled
// rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
// This line sets the RTC with an explicit date & time, for example to set
// January 21, 2014 at 3am you would call:
// rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));

// When the RTC was stopped and stays connected to the battery, it has
// to be restarted by clearing the STOP bit. Let's do this to ensure
// the RTC is running.
rtc.start();

// inicialización de la tarjeta de memoria
Serial.print("Initializing SD card...");

// see if the card is present and can be initialized:
if (!SD.begin(chipSelect)) {
  Serial.println("Card failed, or not present");
```

```
  // don't do anything more:
  while (1);
 }
 Serial.println("card initialized.");

}

void loop() {
 String registro; // Variable que genera el dato a enviar al fichero

 // Lectura del sensor de luminosidad
 lux = sensor_luz.readLightLevel();

 // Lectura de datos del sensor de temperatura y humedad
 sensors_event_t humidity, temp;
 aht.getEvent(&humidity, &temp);// Lee los datos de temperatura y humedad

 // Se lee la temperatura del agua y se guarda en un float
 sensors1.requestTemperatures();   //Se envía el comando para leer la temperatura
 float temp1= sensors1.getTempCByIndex(0); //Se obtiene la temperatura en ºC del sensor 1

 // Primera fila del display
 // Ubicamos el cursor en la primera posición(columna:0) de la primera línea(fila:0)
 lcd.setCursor(0, 0);
 lcd.print("Air ");
 lcd.print(int(temp.temperature)); // Ponemos int para quitar los decimales del display
 lcd.print("C/Hum  ");
 lcd.print(int(humidity.relative_humidity));
 lcd.print("%");

 // Segunda fila del display
 // Ubicamos el cursor en la primera posición(columna:0) de la segunda línea(fila:1)
 lcd.setCursor(0, 1);
 lcd.print("H2O ");
 lcd.print(int(temp1));
 lcd.print("C/Lux ");
 lcd.print(lux);

// Envío de datos por el puerto serie para comprobaciones
 Serial.print(temp1);
 Serial.print("ºC, ");
 Serial.print(temp.temperature);
 Serial.print("ºC, ");
 Serial.print(humidity.relative_humidity);
 Serial.print("%, ");
 Serial.print(lux);
 Serial.println("lux. ");
```

```
// Para la prueba de la hora del reloj
 DateTime now = rtc.now();

  Serial.print(now.year());
  Serial.print('/');
  Serial.print(now.month(), DEC);
  Serial.print('/');
  Serial.print(now.day(), DEC);
  Serial.print(" (");
  Serial.print(daysOfTheWeek[now.dayOfTheWeek()]);
  Serial.print(") ");
  Serial.print(now.hour(), DEC);
  Serial.print(':');
  Serial.print(now.minute(), DEC);
  Serial.print(':');
  Serial.print(now.second(), DEC);
  Serial.println();

// zona para guardar los datos cada intervalo de tiempo programado

 unsigned long currentMillis = millis();

 if (currentMillis - previousMillis >= interval) {
  // save the last time you blinked the LED
  previousMillis = currentMillis;

  // open the file. note that only one file can be open at a time,
  // so you have to close this one before opening another.
  File dataFile = SD.open("datos.txt", FILE_WRITE);

  // if the file is available, write to it:
  if (dataFile) {
   DateTime now = rtc.now();
   registro="";
   registro+=now.year();
   registro+="; ";
   registro+=now.month();
   registro+="; ";
   registro+=now.day();
   registro+="; ";
   registro+=now.hour();
   registro+="; ";
   registro+=now.minute();
   registro+="; ";
   registro+=now.second();
   registro+="; ";
```

```
    registro+=String(temp1);
    registro+="; ";
    registro+=String(temp.temperature);
    registro+="; ";
    registro+=String(humidity.relative_humidity);
    registro+="; ";
    registro+=String(lux);

    dataFile.println(registro);

    //dataFile.println(dataString);
    dataFile.close();

    // print to the serial port too:
    Serial.println(registro);
   }
   // if the file isn't open, pop up an error:
   else {
    Serial.println("error opening datalog.txt");
   }
  }

  //delay(100);
}
```

Fin del código

### 6. Puesta en hora del reloj

En el caso de que la hora que se muestra en el serial monitor difiera de la hora del PC, se deberá ejecutar este código para que la hora de la tarjera se sincronice con la del PC. La hora del reloj se podrá ver en el monitor serie.

Programa 5.- Programa de Arduino para ajustar la hora del reloj a la del PC

```
// Parte dedicada al reloj bus I2C
// Real time clock
// Date and time functions using a PCF8523 RTC connected via I2C and Wire lib
#include "RTClib.h"

RTC_PCF8523 rtc;
char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
"Saturday"};

// Lector de tarjeras microSD a través del bus SPI
/*
 SD card datalogger
```

```
 SD card attached to SPI bus as follows:
 ** MOSI - pin 11
 ** MISO - pin 12
 ** CLK - pin 13
 ** CS - pin 10 (for Arduino nano 33iot)

*/

void setup() {

if (! rtc.begin()) {
   Serial.println("Couldn't find RTC");
   Serial.flush();
   while (1) delay(10);
  }

  if (! rtc.initialized() || rtc.lostPower()) {
   Serial.println("RTC is NOT initialized, let's set the time!");
   // When time needs to be set on a new device, or after a power loss, the
   // following line sets the RTC to the date & time this sketch was compiled
   rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
   // This line sets the RTC with an explicit date & time, for example to set
   // January 21, 2014 at 3am you would call:
   // rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));
   //
   // Note: allow 2 seconds after inserting battery or applying external power
   // without battery before calling adjust(). This gives the PCF8523's
   // crystal oscillator time to stabilize. If you call adjust() very quickly
   // after the RTC is powered, lostPower() may still return true.
  }

  // Para ajustar la hora utilizar este comando con la hora deseada.
  // El formato es DateTime(yy,mm,dd,hh,mm,ss)
  // rtc.adjust(DateTime(2023,12,27,21,25,00));

  // Si se desea poner la hora en la que se compiló el programa utilizar el comando
  rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));

  // When time needs to be re-set on a previously configured device, the
  // following line sets the RTC to the date & time this sketch was compiled
  // rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  // This line sets the RTC with an explicit date & time, for example to set
  // January 21, 2014 at 3am you would call:
  // rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));

  // When the RTC was stopped and stays connected to the battery, it has
```

```
  // to be restarted by clearing the STOP bit. Let's do this to ensure
  // the RTC is running.
  rtc.start();

}

void loop() {

// Para la prueba de la hora del reloj
  DateTime now = rtc.now();

    Serial.print(now.year());
    Serial.print('/');
    Serial.print(now.month(), DEC);
    Serial.print('/');
    Serial.print(now.day(), DEC);
    Serial.print(" (");
    Serial.print(daysOfTheWeek[now.dayOfTheWeek()]);
    Serial.print(") ");
    Serial.print(now.hour(), DEC);
    Serial.print(':');
    Serial.print(now.minute(), DEC);
    Serial.print(':');
    Serial.print(now.second(), DEC);
    Serial.println();

  delay(100);
}
```

Fin del código