```
//
// Capacimetro para aceitunas
// V1.0
//
#include <Capacitor.h>
//
#include <RTCZero.h>

/* Create an rtc object */
RTCZero rtc;

/* Change these values to set the current initial time */
const byte seconds = 0;
const byte minutes = 0;
const byte hours = 0;

/* Change these values to set the current initial date */
const byte day = 1;
const byte month = 1;
const byte year = 23;

// OLED
#include <SPI.h>
#include <Wire.h>
//#include <Adafruit_GFX.h> // No usamos librería gráfica
#include <Adafruit_SSD1306.h>
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels
// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
// The pins for I2C are defined by the Wire-library.
// On an arduino UNO:       A4(SDA), A5(SCL) (Arduino FIO)
// On an arduino MEGA 2560: 20(SDA), 21(SCL)
// On an arduino LEONARDO:   2(SDA),  3(SCL), ...
// MKRZERO: 11 SDA, 12 SCL
#define OLED_RESET     -1 // Reset pin # (or -1 if sharing Arduino reset pin)
#define SCREEN_ADDRESS 0x3C ///< See datasheet for Address; 0x3D for 128x64, 0x3C for
128x32
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
//
//
#include <SD.h>
#include "ArduinoLowPower.h"

int ISRCounter = 0; //Cuenta numero de datos guardados en un fichero de la SD
// Declarada volatil porque se actualiza dentro de una interrupción
volatile int numeroFichero = 1; // Indice del primer fichero que se crea en la SD
String nombreFicheroActual = "oilD_1.csv"; // nombre del primer fichero que crea;
String Version = "(v1.0)";
//const int chipSelect = 4; // CS de la SD
const int chipSelect = SDCARD_SS_PIN; // Para MKR ZERO
const int intPin0 = 0;
const int intPin1 = 1;
const int wakeUpPin = 7;
const int timeThreshold = 200; //tiempo retardo antirrebotes por software de los
pulsadores
long startTime = 0; // para antirrebote pulsadores

#define CapOUT        A2     // Punto común de medida
#define CapIN_L       A3     // Medida capacidad (<1uF)
// Definición de los dos pines de conexión (A3, A2), para capacidades BAJAS
Capacitor pFcap(A3,A2);
// Medidas Offset para el calibrado
float Off_pF_Hr = 0;
float Off_pF_H = 0;
float Off_pF_Low = 0;
float Off_GND = 0;

float medida;
float valor;
String unidadFinal;
float valorFinal;
String unidad;
String tipo;
```

```
72  // Capacidades BAJAS < 1uF
73  int Repe = 30;
74
75  ////////////////////////////////////////////////////////////////////////////////
76  void setup() {
77
78    pinMode(intPin0, INPUT_PULLUP);
79    pinMode(intPin1, INPUT_PULLUP);
80    // pinMode(wakeUpPin, INPUT_PULLUP);
81    // Configuramos los pines de interrupciones (los dos pulsadores) para que
82    // detecten un cambio de bajo a alto
83    attachInterrupt(digitalPinToInterrupt(intPin0), guardaMedida, RISING);
84    attachInterrupt(digitalPinToInterrupt(intPin1), creaFicheroNuevo, RISING);
85
86    LowPower.attachInterruptWakeup(wakeUpPin, wakeUp, CHANGE);
87
88    // Estructura de las variables de calibrado y sus valores por defecto
89    // ... para ATMEGA328P con la librería: Capacitor.h
90    //void Capacitor::Calibrate(float strayCap, float pullupRes);
91    //#define STRAY_CAP (26.30);
92    //#define R_PULLUP (34.80);
93    // ARDUINO UNO pFcap.Calibrate(41.95,37.50);  // Se puede comentar esta línea, si
      los valores C,R son 26.30 y 34.80
94    // pFcap.Calibrate(41.95,34.4); // Arduino FIO (obtenido experimentalmente)
95    pFcap.Calibrate(2.6,38); // Arduino MKRZERO (obtenido experimentalmente)
96
97    pinMode(CapOUT, OUTPUT);
98    pinMode(CapIN_L, OUTPUT);
99
100   // RTC
101   //
102   rtc.begin(); // inicializa el RTC
103   // Set the time
104   rtc.setHours(hours);
105   rtc.setMinutes(minutes);
106   rtc.setSeconds(seconds);
107
108   // Set the date
109   rtc.setDay(day);
110   rtc.setMonth(month);
111   rtc.setYear(year);
112
113
114   Serial.begin(9600);
115 //  while (!Serial) {
116 //    ; // Espera conexión con puerto serie. Necesario para USB nativo
117 //  }
118   Serial.print(F("### Cap. aceituna "));
119   Serial.print(Version);
120   Serial.println(F(" ###"));
121
122   // Inicializa el OLED y pone el mensaje de inicio
123   // SSD1306_SWITCHCAPVCC = generate display voltage from 3.3V internally
124   if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
125     Serial.println(F("Pantalla OLED no detectada"));
126     for(;;); // lazo infinito
127   }
128   cabecera_display();
129   // Comprueba que la SD está conectada:
130   Serial.println("Inicializando tarjeta SD...");
131   display.setCursor(5,18);
132   display.print("Comp.tarjeta SD...");
133   display.display();
134   delay(2000);
135   if (!SD.begin(chipSelect)) {
136     Serial.println("Fallo en SD o tarjeta no presente");
137     cabecera_display();
138     display.setCursor(0,24);
139     display.print("Fallo en SD");
140     display.display();
141     // don't do anything more:
142     while (1);
143   }
```

```
    cabecera_display();
    display.setTextSize(1);
    display.setTextColor(SSD1306_WHITE);
    delay(2000);
    display.setCursor(5,20);
    display.print("....CALIBRANDO");
    display.display();
    calibrado();
    delay(3000);
    cabecera_display();
    sdCalOk();

  }

  ////////////////////////////// CAPACIDAD ////////////////////////////////////
  void loop() {
        tipo = "  <1uF";
        midePF();
        valor = valor - Off_pF_Low;
        // No muestra valores inferiores a 1 pF
        if (valor < 1) {
          valor = 0;
          unidad = " pF";
        }else if (valor > 1000000) {
          valor = -999;
          unidad = "> 1uF";
        }else if (valor > 1000) {
          valor = valor / 1000;
          unidad = " nF";
        }else{
          // Ajuste fino (pF)
          if (valor < 35){
            valor = valor - 9;
          }else if (valor < 50){
            valor = valor - 13;
          }else if (valor < 65){
            valor = valor - 17;
          }else {
            valor = valor - 22;
          }
          unidad = " pF";
        }

      // Test OK >>> se muestra la medida en el display
      if (valor < 0 && valor != -999) {valor = 0;}


      // Presenta los resultados de la medida y muestra la actividad en el display
      Serial.print(F("Capacidad"));
      Serial.print(tipo);
      Serial.print(F(" = "));

      if (valor != -999) {
        if (valor < 1000 &&  unidad != " pF" ) {
          cabecera_display();
          sdCalOk();
          valorFinal = valor;
          unidadFinal = unidad;
          display.setTextSize(2);             // Draw 2X-scale text
          display.setCursor(5,23);
          display.print(valorFinal);
          display.display();
          Serial.print(valorFinal,2);
        }else {
          cabecera_display();
          sdCalOk();
          valorFinal = valor;
          unidadFinal = unidad;
          display.setTextSize(2);
          display.setCursor(5,23);
          display.print(valorFinal);
          display.display();
          Serial.print(valorFinal,0);
```

```
      }
    }
    display.print(unidadFinal);
    display.display();
    Serial.println(unidadFinal);
    // Retardo entre cada actualización de medida
    delay(2500);
    Serial.println(rtc.getMinutes());
    Serial.println(rtc.getSeconds());
    if (rtc.getMinutes()>= 1) {
      display.clearDisplay();
      display.display();
      LowPower.sleep();
    }

}
////////////////////////////////////////////////////////////////////////////////////////////
void calibrado() {
  midePF();
  Off_pF_Low = valor;
  Serial.print(F("Offset <1uF (pF): "));
  Serial.println(Off_pF_Low);
}

//////////////////////////////////////////////////////
void midePF(){
  //descargaCap();
  float valorMedio = 0;                    // Reinicia valor medio
  for(int i = 0; i < Repe; i++){
    valor = pFcap.Measure();
    valorMedio = valorMedio + valor;
    //descargaCap();
  }
  valor = valorMedio / Repe;               // Guarda el valor promedio de las muestras
}
////////////////////////////////////////////////////////////////////////
// Muestra cabecera y versión
void cabecera_display(void) {
  // display oled
  display.clearDisplay();
  display.setTextSize(1);  // Normal 1:1 escala de pixel
  display.setCursor(0,0);
  display.setTextColor(SSD1306_BLACK, SSD1306_WHITE); // Texto invertido
  display.println(F(" Medida CAP ACEITUNA "));
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(1,10);
  display.print(Version);
  display.display(); // muestra buffer del display
}
////////////////////////////////////////////////////////////////////////

// Muestra datos actualizados en display
void sdCalOk (void) {
  display.setCursor(45,10);
  display.print("SD:Ok  Cal:Ok");
  display.setCursor(5,52);
  display.print("                    ");
  display.setCursor(5,52);
  display.print(nombreFicheroActual);
  display.setCursor(88,52);
  display.print(ISRCounter);
  display.display();
}

////////////////////////////////////////////////////////////////////////
// ISR pin 0, guarda medida en fichero
void guardaMedida()
{
  if (millis() - startTime > timeThreshold)
  {
    rtc.setHours(hours);
    rtc.setMinutes(minutes);
```

```
      rtc.setSeconds(seconds);
      ISRCounter++;
      startTime = millis();
      Serial.print("entrando interrupt...");
      Serial.println(ISRCounter);
      Serial.println(nombreFicheroActual);
      display.setTextSize(1);
      display.setCursor(119,58);
      display.print("*");
      display.display();
      File dataFile = SD.open(nombreFicheroActual, FILE_WRITE);
      // Si el fichero está disponible, se guarda el dato
      if (dataFile) {
        dataFile.print(valorFinal);
        dataFile.print(",");
        dataFile.println(unidadFinal);
        dataFile.close();
        // Vuelca también al puerto serie (para depuración)
        Serial.print("dato grabado en SD: ");
        Serial.print(valorFinal);
        Serial.println(unidadFinal);
      } else {  // Si no se abre el fichero, muestra error
        Serial.println("error abriendo archivo");
      }
    }
  }

  ////////////////////////////////////////////////////////////////////////
  // ISR pin 1, crea nombre de fichero nuevo
  void creaFicheroNuevo()
  {
    if (millis() - startTime > timeThreshold) {
      startTime = millis();
      numeroFichero++;
      ISRCounter = 0;
      nombreFicheroActual = "oilD_"+ (String)numeroFichero+".csv";
      Serial.print("Nuevo fichero: ");
      Serial.println(nombreFicheroActual);
    }
  }

  ////////////////////////////////////////////////////////////////////////
  void wakeUp() {
    // This function will be called once on device wakeup
    // You can do some little operations here (like changing variables which will be
    used in the loop)
    // Remember to avoid calling delay() and long running functions since this
    functions executes in interrupt context
  //  rtc.setHours(hours);
  //  rtc.setMinutes(minutes);
  //  rtc.setSeconds(seconds);
  }

  // FIN
```