

# Conversie VFD → Afișaj LED cu funcție duală: ceas & frecvențmetru

## VFD LED display conversion: clock & frequency meter for KENWOOD R\_1000

Autor: Traian Angheluță YO3HFP

e-mail: yo3hfp@yahoo.com

Data: 09.05.2025

---

### Descriere generală

Acest proiect înlocuiește complet modulul original de afișaj VFD al receptorului de unde scurte Kenwood R-1000, păstrând însă funcționalitatea esențială și autenticul aspect estetic al panoului frontal. Inițial, receptorul era echipat cu un afișaj VFD controlat de circuitul integrat OKI MSM5524, greu sau de negăsit și imposibil de reparat în cazul defectării.

### Proiectul propus oferă:

Afișaj LED pe 6 cifre (7 segmente), compatibil cu spațiul frontal existent

Comutare între ceas și frecvențmetru

Afișare frecvență în format X.XXXX MHz

Afișare oră și minut cu separator clipitor

Butoane originale păstrate:

FREQ/CLOCK (comutare între moduri)

HOURL și MINUTE (setare manuală ceas)

Comandă pentru reglaj luminozitate: apăsare simultană HOURL + MINUTE

---

### Componente necesare

Microcontroler ATmega328P în varianta DIP-28

Afișaj LED cu 6 cifre / 7 segmente (cu anod comun)

RTC DS3231 (modul I2C)

Modul contor frecvență (bază FreqCount)

Butoane push 3x (montate pe panoul frontal)

Rezistențe și fire de conexiune

---

### Funcționalitate

Funcție	Activare
Afișaj ceas	Pin A1 = LOW (activat de butonul original)
Afișaj frecvență	Pin A1 = HIGH (default)
Setare oră	Buton D2 (HOURL) apăsăat în modul ceas
Setare minut	Buton D8 (MINUTE) apăsăat în modul ceas
Reglare luminozitate	Apăsare simultană D2 + D8

---

## Software

Sketch-ul a fost scris în Arduino IDE și folosește următoarele librării:

SevSeg – pentru controlul afișajului 7 segmente

RTClib – pentru accesarea și ajustarea ceasului DS3231

FreqCount – pentru măsurarea frecvenței RF

Luminozitatea afișajului se comută între două nivele (ex. 100% și 20%) la apăsarea simultană a butoanelor HOUR și MINUTE, și este confirmată prin afișarea textului brt timp de 1 secundă.

---

## Alocare pini

Funcție	Pin ATmega328P
Afișaj segmente	D6–D13, A0
Cifre afișaj	A2, A3, D0–D4
Frecvență IN	D5
Buton ceas/frecvență	A1
Buton setare oră	D2
Buton setare minut	D8
RTC SDA / SCL	A4 / A5

---

### Codul:

```
/*Conversie VFD → Afișaj LED cu funcție duală: ceas & frecvențmetru
Autor: Traian YO3HFP
e-mail: yo3hfp@yahoo.com
Data: 09.05.2025
```

```
Acest proiect înlocuiește complet sistemul original cu afișaj VFD
al receptorului de unde scurte Kenwood R-1000,
păstrând însă funcționalitatea esențială și autenticul aspect estetic al panoului frontal.
Inițial, receptorul era echipat cu un afișaj VFD controlat de circuitul integrat OKI MSM5524,
greu (sau de negăsit) și imposibil de reparat în cazul defectării.
Configurație pentru Arduino NANO
Butonul DIMMER de pe frontala receptorului va trebui sa fie în cazul acestui proiect, fără
reținere
*/
```

```
#include <SevSeg.h>
#include <Wire.h>
#include "RTClib.h"
#include <FreqCount.h>
```

```
SevSeg sevseg;
RTC_DS3231 rtc;
```

```
const byte switchPin = A1;          // Comutator mod: ceas/frecventa
const byte setHourPin = 2;          // Buton setare ora
const byte setMinutePin = 8;        // Buton setare minut
```

```
unsigned long lastMillis = 0;
bool showTime = false;
bool isDim = false;
unsigned long showBrightnessUntil = 0;
```

```
void setup() {
```

Proiect de înlocuire a afișajului pentru Kenwood R-1000

```

// Setări afișaj
byte numDigits = 6;
byte digitPins[] = {A2, 3, 4, A3, 0, 1};
byte segmentPins[] = {6, 7, A0, 9, 10, 11, 12, 13};
bool resistorsOnSegments = true;
byte hardwareConfig = COMMON_ANODE;

sevseg.begin(hardwareConfig, numDigits, digitPins, segmentPins, resistorsOnSegments);
sevseg.setBrightness(90); // valoare inițială: luminozitate normală

// RTC
Wire.begin();
rtc.begin();

// Pini butoane
pinMode(switchPin, INPUT_PULLUP);
pinMode(setHourPin, INPUT_PULLUP);
pinMode(setMinutePin, INPUT_PULLUP);

// FreqCount
FreqCount.begin(500); // 0.5 secunde
}

void loop() {
    showTime = (digitalRead(switchPin) == LOW);

    if (showTime) {
        handleTimeSetting(); // verifică butoanele H și M pentru setare oră
        // Afișează ora doar dacă nu e activ mesajul de luminozitate
        extern unsigned long showBrightnessUntil; // declarare externă
        if (millis() >= showBrightnessUntil) {
            displayTime();
        }
    } else {
        if (FreqCount.available()) {
            unsigned long count = FreqCount.read();
            count = count * 2.0; // compensare pentru 0.5s

            long freq = (long)count * 10 - 455000;
            if (freq < 0) freq = 0;

            freq = freq / 100; // sute de Hz
            sevseg.setNumber(freq, 4); // format X.XXX.X
        }
    }

    handleBrightnessToggle(); // verifică apăsarea simultană D2 + D8
    sevseg.refreshDisplay(); // actualizează display-ul
}

// Afișare HH-MM cu separator clipitor
void displayTime() {
    static unsigned long lastToggle = 0;
    static bool dashVisible = true;

    unsigned long currentMillis = millis();
    if (currentMillis - lastToggle >= 500) {
        dashVisible = !dashVisible;
        lastToggle = currentMillis;
    }

    DateTime now = rtc.now();
    int hour = now.hour();
    int minute = now.minute();

    char timeStr[7];
    if (dashVisible) {
        sprintf(timeStr, sizeof(timeStr), "%02d-%02d", hour, minute);
    } else {

```

```

    snprintf(timeStr, sizeof(timeStr), "%02d %02d", hour, minute);
}

sevseg.setChars(timeStr);
}

// Setare ora și minut cu debounce
void handleTimeSetting() {
    static unsigned long lastHourPress = 0;
    static unsigned long lastMinutePress = 0;
    static unsigned long lastBrightnessToggle = 0;

    bool hourPressed = digitalRead(setHourPin) == LOW;
    bool minutePressed = digitalRead(setMinutePin) == LOW;

    // Dacă ambele butoane sunt apăstate simultan
    if (hourPressed && minutePressed) {
        if (millis() - lastBrightnessToggle > 500) {
            isDim = !isDim;
            sevseg.setBrightness(isDim ? -10 : 200); // Valori recomandate in cazul afisajului OPD-
S3010LR-BW (la nevoie, modificați)
            sevseg.setChars("brt"); // mesaj vizual
            showBrightnessUntil = millis() + 1000; // Afișează "brt" 1 sec.
            lastBrightnessToggle = millis();
        }
        return;
    }

    // Doar unul dintre butoane e apăsat:
    if (hourPressed && millis() - lastHourPress > 300) {
        DateTime now = rtc.now();
        rtc.adjust(DateTime(now.year(), now.month(), now.day(), (now.hour() + 1) % 24,
now.minute(), 0));
        lastHourPress = millis();
    }

    if (minutePressed && millis() - lastMinutePress > 300) {
        DateTime now = rtc.now();
        rtc.adjust(DateTime(now.year(), now.month(), now.day(), now.hour(), (now.minute() + 1) %
60, 0));
        lastMinutePress = millis();
    }

    // Dacă încă trebuie să afișeze "brt"
    if (millis() < showBrightnessUntil) {
        return; // evităm să suprascriem cu ora/frecvența
    }
}

// Schimbă luminozitatea când D2 + D8 sunt apăstate simultan
void handleBrightnessToggle() {
    static bool lastBothPressed = false;
    static bool isDim = false;

    bool hourPressed = digitalRead(setHourPin) == LOW;
    bool minutePressed = digitalRead(setMinutePin) == LOW;
    bool bothPressed = hourPressed && minutePressed;

    // Detectare tranziție LOW -> HIGH (apasare simultana)
    if (bothPressed && !lastBothPressed) {
        isDim = !isDim;
        sevseg.setBrightness(isDim ? 20 : 90);
    }

    lastBothPressed = bothPressed;
}

```

