2017

8X8 DOT MATRIX WORD CLOCK

This is an AVR based mini word clock featuring:

- Real Time Clock (DS3231)
- LiPo battery with integrated USB charging circuit (MCP73831)
- low battery sensing

The unit is composed of two stacked PCB and one 8x8 dot-matrix display module.

Heol Fief

github.com/heolfief

Table of Contents

1. 8x8 Dot matrix letters and MCU PORTs configuration

a. Letters display configuration

FIGURE 1: LETTERS CONFIGURATION

Figure 1 is the matrix configuration for displaying time in the following format:

Minutes	Link	Hours
	PAST	ONE
FIVE	TO	TWO
TEN		THREE
FIFTEEN		FOUR
TWENTY		FIVE
TWENTY FIVE		SIX
HALF		SEVEN
		EIGHT
		NINE
		TEN
		ELEVEN
		TWELVE

e.g.: HALF PAST TEN (10:30) or TEN TO THREE (02:50).

Note that a.m. and p.m. are considered the same.

b. LED matrix pinout

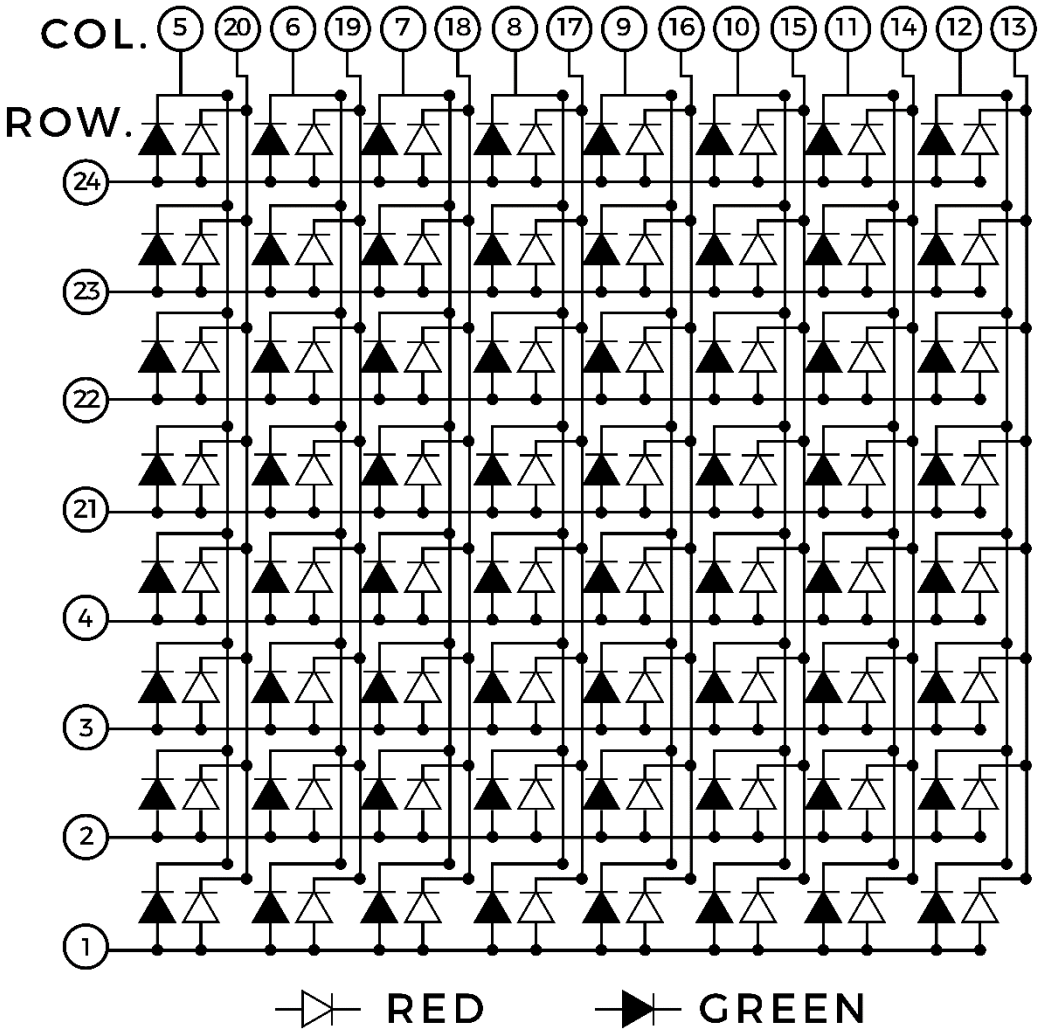

FIGURE 2: LED MATRIX PINOUT

This is the pinout diagram of the “1588ABEG-5” 8x8 dot matrix display.
In this design, only the green LED are used. Therefore, red LED cathodes (pins 20, 19, 18, 17, 16, 15, 14 and 13) are connected to VCC (LEDs off).

c. MCU ports configuration

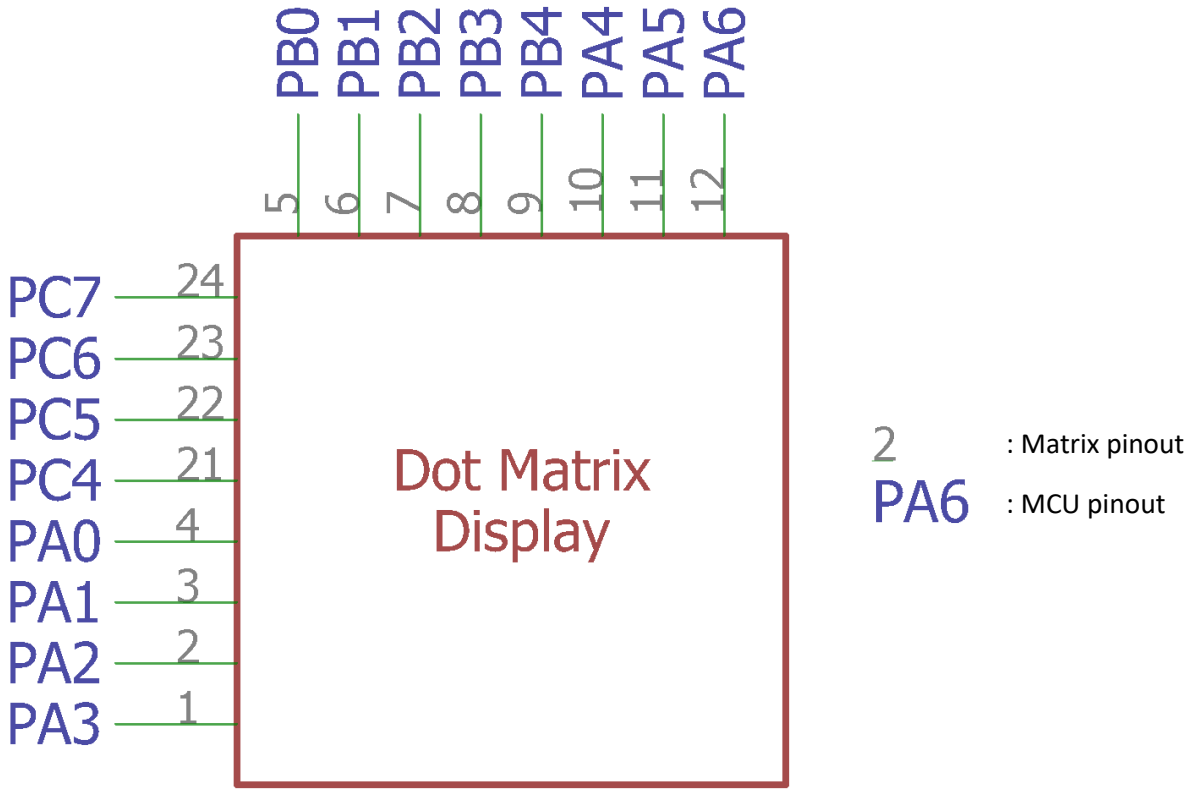

FIGURE 3: MCU TO MATRIX CONNECTIONS

Figure 3 is the connection diagram between the AVR pins and the Matrix pins.
'PXy' stands for PORTX pin y.

2. Battery voltage sensing calculations

a. Voltage divider:

LiPo battery voltage needs to be stepped down from 4.3v (maximum value) to 3.3v to be read properly by the MCU running at 3.3v.

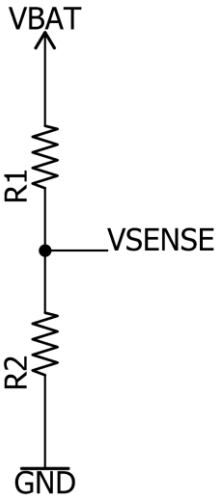

FIGURE 4: BATTERY SENSING VOLTAGE DIVIDER

$$V_{sense} = \frac{R_2}{R_1+R_2}V_{bat}$$
$$\Leftrightarrow R_2 = -\frac{V_{sense}}{V_{sense}+V_{bat}}R_1$$
$$\Leftrightarrow R_2 = -\frac{3.3}{3.3+4.3}R_1$$
$$\Leftrightarrow R_2 = 3.3R_1$$

For **R_1 = 100kΩ**, **R_2 =330kΩ**.

By choosing such high resistor values, we limit the current going from V_{bat} to GND, limiting battery discharge.

Here, $I_{wasted} = \frac{V_{bat}}{R_1+R_2} = \frac{3.7}{430}k = 8.6\mu A$ which won't discharge the battery too quickly.

By taking the formula in reverse, we get: $V_{sense} = 330\frac{k}{330k+100k}V_{bat} = \frac{33}{43}V_{bat}$ which means for example if the MCU mesures 2.5v, the battery voltage V_{bat} equals $2.5/\frac{33}{43} = 2.5\frac{43}{33} = 3.26$v.

a. Sensing voltage:

The ATmega32A has a 10 bit DAC which means it can measure voltages from GND (0v) to AREF (3,3v in this case), with a 10 bits resolution (0 to 1023).

To get the battery voltage from the 10 bit number returned by the ADC conversion, we need to apply this formula: $V_{bat} = \frac{3.3ADC_{value}}{1023} \times \frac{43}{33}$.

In C, it is done by executing this line of code: `float voltage = adcvalue*3.3*43/1023/33;`