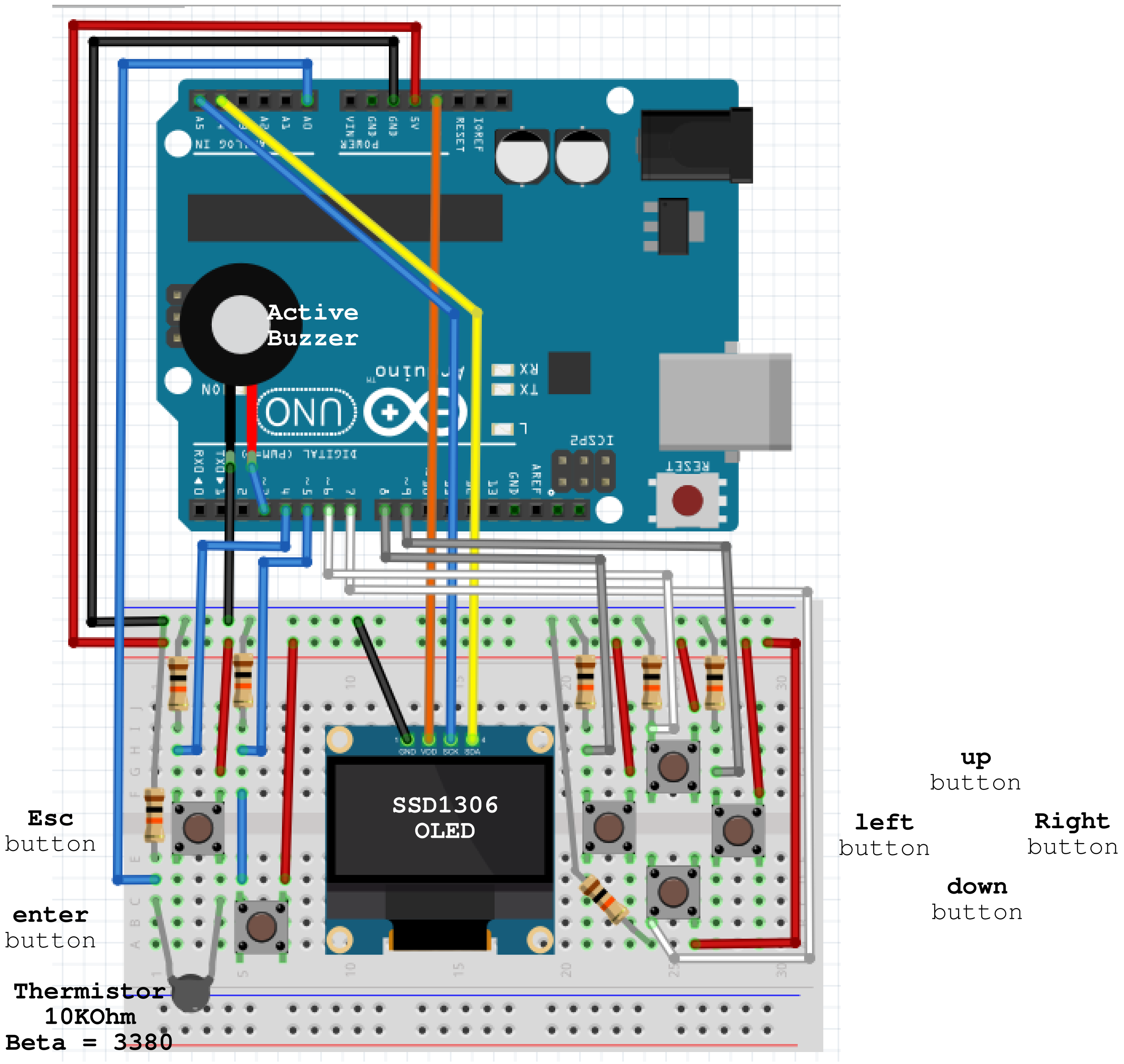
ATMEGA328/ARDUINO - PROJECT - DIGITAL ALARM CLOCK - OLED
https://github.com/teaksoon/p_daco



HARDWARE
1x Computer with Arduino IDE Software
1x USB 2.0 Type A/B Data Cable
1x Arduino Uno Board
1x Solderless Breadboard
Nx Jumper wires

1x Active Buzzer
6x Tactile Switch with 6x 10KOhm Resistor
1x SSD1306 OLED Module i2c 64x128 pixel

- Modular Design Extension -

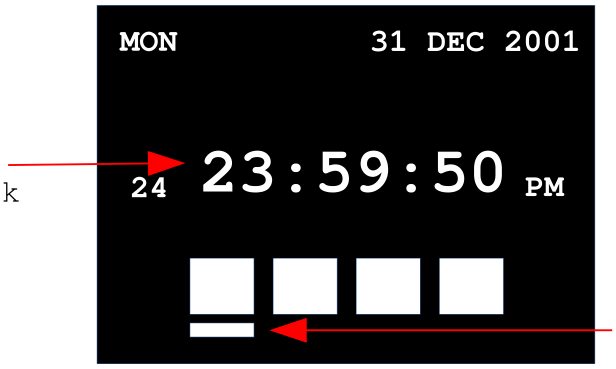
1x 10Kohm Thermistor(Beta=3380) with 1x 10KOhm Resistor

ATMEGA328/ARDUINO - PROJECT - DIGITAL ALARM CLOCK - OLED
https://github.com/teaksoon/p_daco

Source code: **p_daco_clock**
Download from:
https://github.com/teaksoon/p_daco/blob/main/2022_01_08_p_daco_source.zip
Upload PROGRAM, watch the OLED Screen

While in the MENU MODE and the Navigation Bar is at the CLOCK FUNCTION, the OLED screen will display CLOCK FUNCTION “Live” Clock (updated every 1 second)

Clock Ticker
– “Live” Clock



ESC Button – Move Navigation Bar to first MENU OPTION

LEFT/RIGHT Button – Move MENU Navigation Bar. Show different FUNCTION Live Data

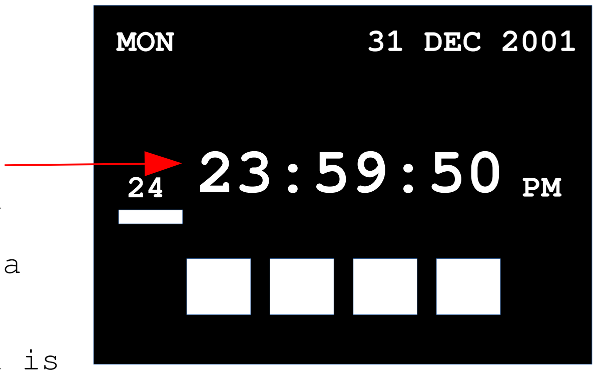
ENTER Button – Move into FUNCTION EDIT MODE

Presssing ENTER Button while in MENU MODE, will enter the EDIT MODE (CLOCK FUNCTION)

While in CLOCK FUNCTION EDIT MODE, “Live” Clock runs in the background, while the Screen allow Data Editing at Navigation Bar position

Clock Ticker
– “Live” Clock stopped on the Screen for Data editing

– “Live” Clock is still running at the background, will resume when return from ESC



ESC Button – Cancel Edited Data and return to MENU MODE, continue with the previous Data

LEFT/RIGHT Button – Move Navigation Bar

UP/DOWN Button – Change data at the Navigation Bar position

ENTER Button – Save Edited Data and return to MENU MODE, continue from Saved Data

At this stage we have the “Live” Time and Date (can be viewed in both 12Hours and 24Hours Mode) updated every 1 second. Seconds, Minutes, Hour, AM/PM state, Day, Month, Year and Day of Week will be adjusted accordingly after every seconds.

We are also able to change Time and Date in both 12 hours and 24 hours mode with our tactile buttons

Meaning, we have a fully functional Digital Clock now

Next, we will be adding the Alarm Time Setting and Alarm Trigger

When changing Data (Month/Year) using the buttons (increase/decrease) may have effects the other Data

Year Change: may effect the range of Days for FEB month and the Day Data
For example: Original Data is 29 FEB 2000
Changing Year from 2000 (Leap Year) to 2001 (non-Leap Year) will make 29 FEB 2001 invalid. Day needs to be adjusted to 28 FEB 2001 because there is no 29 FEB 2001. At the same time, FEB 2001 will have a range of Days (1 to 28) where previously FEB 2000 have range of Days (1 to 29)

Month Change: may effect the Day as each Month has different number of Days
For example: Original Data is 31 DEC 2001
Changing month from DEC to NOV will make 31 NOV 2001 invalid. Day needs to be adjusted to 30 NOV 2001 because there is no 31 NOV 2001

Day Change: The range of Days is from 1 to last day of Month. Last day of Month for FEB will depend on Leap Year or Non-Leap Year
For example:
for FEB 2000, range of Days will be 1 to 29
for FEB 2001, range of Days will be 1 to 28

Day of Week Change: just cycle within 1 to 7 (MON to SUNDAY)
Seconds Change and Minutes Change: just cycle within 0 to 59

CLOCK HOUR MODE CHANGE:
Changing between 12Hours Mode and 24Hours Mode will cause some changes in the Hour value and AM/PM State

24Hours Mode to 12Hours Mode:
Anything more than 11Hours is a PM otherwise AM
Anything more than 12Hours needs to be adjusted, minus 12
0 Hours needs to be adjusted to 12

12Hours Mode to 24Hours Mode:
When in PM mode, anything less than 12 need to be adjusted, plus 12
When in AM mode, 12AM needs to be adjusted to 0

```
void clock_edit_hhmode() {
  if (clo_hhmode_t == HHMODE_24) {
    // Currently 24 Hours Mode
    if (clo_hh_t > 11) {
      clo_ampm_t = PM;  // hh more than 11, PM
      if (clo_hh_t > 12) clo_hh_t = clo_hh_t-12; // PM, -12 when more 12
    } else {
      clo_ampm_t = AM;  // hh = 11 or less, AM
      if (clo_hh_t == 0) clo_hh_t=12; // 0 in 24hours = 12 in 12hours Mode
    }
    clo_hhmode_t = HHMODE_12; // Switch to 12hours Mode
  } else {
    // Currently 12 Hours Mode
    if ( clo_ampm_t == PM) {
      if (clo_hh_t < 12) clo_hh_t = clo_hh_t+12; // PM, +12 when less 12
    } else {
      if (clo_hh_t == 12) clo_hh_t = 0; // 12 in 12hours = 0 in 24hours Mode
    }
    clo_hhmode_t = HHMODE_24; // Switch to 24hours Mode
  }
  sf_show_hhmode(clo_hhmode_t);  // hhmode
  sf_show_ampm(clo_ampm_t); // ampm
  g_int_padl(clo_hh_t,tmp_s,2,'0');p_ssd1306_string_L(3,24,tmp_s); // hh
}
```

HOUR CHANGE:
We will need to do some adjustments to the Hour Value and the AM/PM state whhen adding or reducing on existing Hour Data

In 24Hours Mode:
Adding 1 Hour to 23 Hours will turn 0 Hours (PM to AM)
Reduce 1 Hour from 0 Hours will turn to 23 Hours (AM to PM)
Anything more than 11 Hour is PM otherwise AM

In 12Hours Mode:
Adding 1 Hour to 11PM will turn 12AM (PM to AM)
Adding 1 Hour to 12PM will turn 1PM (there is no 0 PM, 12 becomes 1)
Adding 1 Hour to 11AM will turn to 12PM (AM to PM)
Reduce 1 Hour from 12AM will turn 11PM (AM to PM)
Reduce 1 Hour from 1PM will turn 12PM (there is no 0 PM, 1 becomes 12)
Reduce 1 Hour from 12PM will turn 11AM (PM t AM)

```
void clock_edit_hh(int8_t editDir) {
  if (clo_hhmode_t == HHMODE_24)  {
    // 24 hours mode, cycle between 0 to 23
    clo_hh_t = g_next_nn(editDir,clo_hh_t,0,23);
    clo_ampm_t = (clo_hh_t > 11) ? PM:AM;
  } else {
    // 12 hours mode
    if (editDir == 1) {
        // Increase One Hour
        if (clo_ampm_t == PM) {
          // This is PM
          if (clo_hh_t == 11) clo_ampm_t = AM; // 11PM to 12AM
          clo_hh_t++;
          if (clo_hh_t > 12) clo_hh_t=1; // next hour affer 12 is 1
        } else {
          // This is AM
          if (clo_hh_t == 11) clo_ampm_t=PM; // 11AM to 12PM
          clo_hh_t++;
          if (clo_hh_t > 12) clo_hh_t= 1; // next hour after 12 is 1
        }
    } else {
      if (editDir == -1) {
        // Reduce One Hour
        if (clo_ampm_t == PM) {
          // This is PM
          if (clo_hh_t == 12) clo_ampm_t = AM; // 12PM to 11AM
          clo_hh_t--;
          if (clo_hh_t < 1) clo_hh_t=12; // previous hour before 1 is 12
        } else {
          // This is AM
          if (clo_hh_t == 12) clo_ampm_t = PM; // 12AM to 11PM
          clo_hh_t--;
          if (clo_hh_t < 1) clo_hh_t=12; // previous hour before 1 is 12
        }
      }
    }
  }
  sf_show_ampm(clo_ampm_t); // ampm
  g_int_padl(clo_hh_t,tmp_s,2,'0');p_ssd1306_string_L(3,24,tmp_s); // hh
}
```