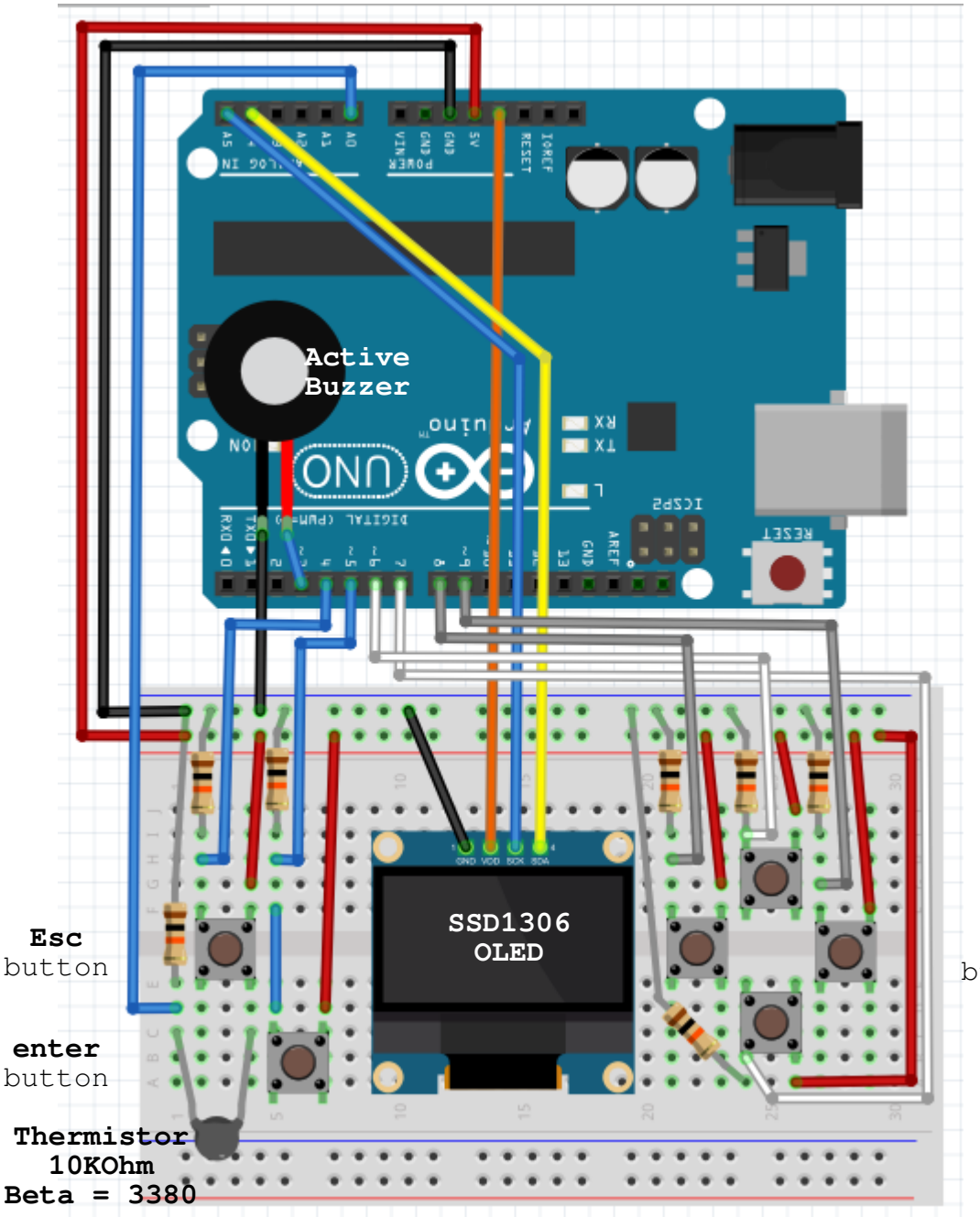
ATMEGA328/ARDUINO - PROJECT - DIGITAL ALARM CLOCK - OLED
https://github.com/teaksoon/p_daco



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 -

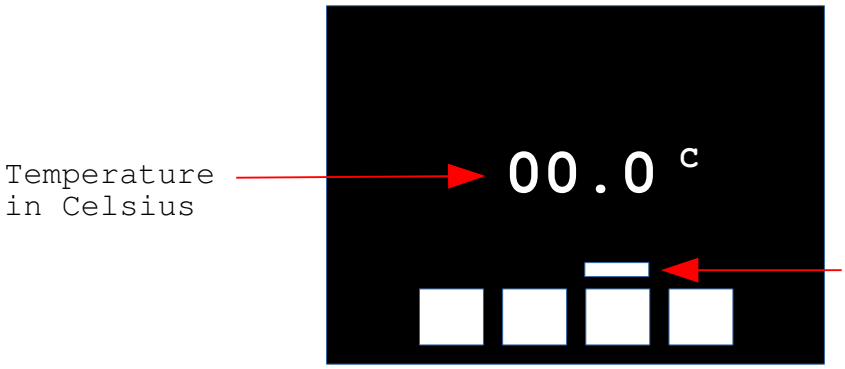
1x 10Kohm Thermistor(Beta=3380) with 1x 10KOhm Resistor

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

Source code: **p_daco_clock_therm**
Download from:
https://github.com/teaksoon/p_daco/blob/main/2022_01_10_p_daco_source.zip
Upload PROGRAM, watch the OLED Screen

While in the MENU MODE and the Navigation Bar is at the THERMO FUNCTION, the OLED screen will display "Live" Temperature readings (updated every 1 second). Our Thermometer has 2 display modes

1. Basic Live Temperature Display (default)
2. Live Temperature Display with Live Analog/Digital Clock Time/Date

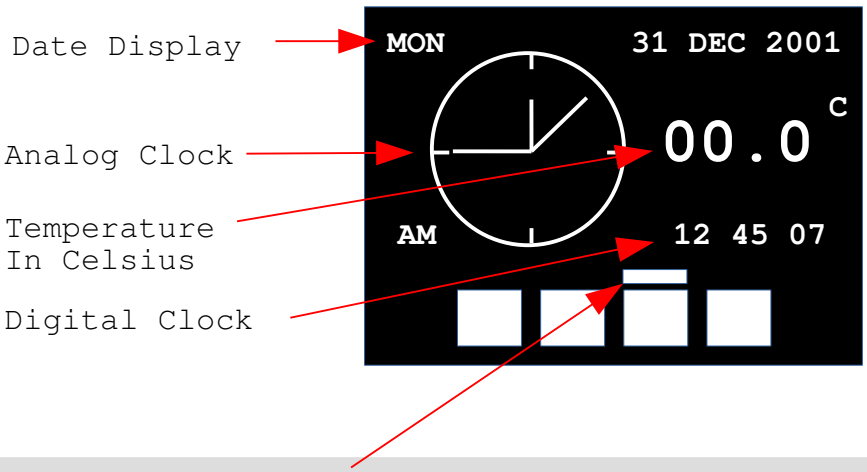


ESC Button - Move Navigation Bar to first MENU OPTION

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

ENTER Button - For this FUNCTION, we simply switch thermometer display modes

Presssing ENTER Button while in MENU MODE at this MENU ITEM, will switch between two display modes



ESC Button - Move Navigation Bar to first MENU OPTION

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

ENTER Button - For this FUNCTION, we simply switch thermometer display modes

Menu Navigation Bar has been changed to be on top of Menu Icons. This is to cater for the second thermometer display mode. The original Menu was too cramped

Temperature Reading from thermistor

We simply do 7 readings from the Thermistor and take the middle reading. This is to filter stray readings

```
float therm_b3380_celsius(uint8_t analogPin) {
const float tBeta     = 1.0/3380.0;   // Set by factory Beta=3380, 10Kohm
Thermistor use with 10Kohm Resistor
const float tControl  = 1.0/298.15;   // Factory Control Temperature=25
Celsius. (25+273.15) = 298.15 in Kelvin
float t_kelvin, t_cel;                // float Working Memory
int read_array[7];

  // Make 7 readings to filter strays
  for (int8_t i=0; i<7; i++) {
    read_array[i] = analogRead(analogPin);
  }
  t_kelvin  =
1.0/(tControl+(tBeta*(log((1023/(float)g_median_int(read_array,7))-1.0))));
  t_cel     = t_kelvin-273.15;  // Kelvin to Celsius
  return t_cel;
}

int g_median_int(int data_array[], int data_count) {
int temp;
  for(int8_t x=0; x < data_count; x++) {
    for(int8_t y=x+1; y < data_count; y++) {
      if(data_array[x] > data_array[y]) {
        temp = data_array[x];
        data_array[x] = data_array[y];
        data_array[y] = temp;
      }
    }
  }
  return data_array[(int)data_count/2];
}
```

1. Basic Live Temperature Display (default)

This will be updated every 1 seconds from our PROGRAM Super Loop. We will assume that the temperature will be within 2 digits. We also take in 1 decimal points from the temperature reading for display

```
void therm_show_basic() {
float t_celsius;
int temp_int;

  t_celsius = therm_b3380_celsius(pin_thermo);

  temp_int = (int) t_celsius; // whole number portion
  g_int_padl(temp_int,tmp_s,2,'0'); p_ssd1306_string_L(2,44,tmp_s);
  p_ssd1306_bitmap(3,70,bmp_dot,2,0);
  temp_int = (int) ((t_celsius-temp_int)*10); // fraction portion
  g_int_padl(temp_int,tmp_s,1,'0'); p_ssd1306_string_L(2,74,tmp_s);
  p_ssd1306_char(2,89,'C');
}
```

2. Live Temperature Display with Live Analog/Digital Clock Time/Date

This show everything from the **1. Basic Live Temperature Display Mode**. This mode will have additional information, Live Time/Date in both Analog Graphical and Digital Time/Date

```
void therm_show_clock() {
float t_celsius;
int temp_int;
int8_t hh12;
int8_t tmp;

  // Reset the entire screen for this
  p_ssd1306_clear();
  menu_show();
  p_ssd1306_bitmap(menu_rc[menu_active][0],menu_rc[menu_active]
[1],m_nav_bmp,10,0);

  // Temperature
  t_celsius = therm_b3380_celsius(pin_thermo);
  temp_int = (int) t_celsius; // whole number portion
  g_int_padl(temp_int,tmp_s,2,'0'); p_ssd1306_string_L(2,77,tmp_s);
  p_ssd1306_bitmap(3,103,bmp_dot,2,0);
  temp_int = (int) ((t_celsius-temp_int)*10); // fraction portion
  g_int_padl(temp_int,tmp_s,1,'0'); p_ssd1306_string_L(2,107,tmp_s);
  p_ssd1306_char(2,122,'C');

  // Digital Date/Time
  p_ssd1306_string(0,0,dow_str[clo_dow-1]);                          // dow
  g_int_padl(clo_dd,tmp_s,2,'0');p_ssd1306_string(0,67,tmp_s);    // dd
  p_ssd1306_string(0,82,mo_str[clo_mo-1]);                           // mo
  g_int_padl(clo_yyyy,tmp_s,4,'0');p_ssd1306_string(0,103,tmp_s); // yyyy
  tmp_s[2]='\0';
  tmp_s[0]=(clo_ampm==AM) ? 'A':'P';tmp_s[1]='M';
  p_ssd1306_string(5,6,tmp_s); // ampm

  if (clo_hhmode == HHMODE_12) {
    hh12 = clo_hh;
  } else {
    // 24 hours mode
    if (clo_hh > 12) {
      hh12 = clo_hh-12; // when more than 12, -12
    } else {
      if (clo_hh == 0) hh12 = 12; // 0 in 24hrs = 12
    }
  }
  g_int_padl(hh12,  tmp_s,2,'0');p_ssd1306_string(5, 67,tmp_s); // hh
  g_int_padl(clo_mm,tmp_s,2,'0');p_ssd1306_string(5, 82,tmp_s); // mm
  g_int_padl(clo_ss,tmp_s,2,'0');p_ssd1306_string(5, 97,tmp_s); // ss

  // Analog Clock Display
  p_ssd1306_circle(23,43,22,1);
  p_ssd1306_circle(23,43,1,1);
  for (int i=0;i<12;i++) {
    p_ssd1306_radial(23,43,21,19,12,i,1);
  }
  tmp = (hh12 == 12) ? 0:hh12;
  p_ssd1306_radial(23,43,11,3,12,tmp,1);     // hh
  p_ssd1306_radial(23,43,15,3,60,clo_mm,1);  // mm
  p_ssd1306_radial(23,43,17,3,60,clo_ss,1);  // ss
}
```

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

At this stage,our OLED Digital Clock is fully funtional with Alarm feature

Clock - Live Running Clock showing Time in 12/24Hours Mode, Date
(Automatically updated every seconds with auto AM/PM adjustments)

Clock - Time/Date Editing with 12/24Hours Mode Change with auto Hour, AM/PM adjustments

Alarm Clock - ON/OFF Switch and Alarm Time Display/Editing

Alarm Monitoring - Trigger Alarm when Alarm Time and Clock Time matches

Thermometer Display - Will display temperature in 2 modes, basic temperature or temperature with Time/Date in graphical Analog and Text

Utility - To set and test Alarm Buzzer duration time in seconds

Below is the amount of PROGRAM MEMORY and SRAM MEMORY from the ATMEGA328 micro-controller that has been used by our PROGRAM (PROGRAM codes that are not optimized yet, means we can still reduce)

```
Done uploading.

Sketch uses 10818 bytes (33%) of program storage space. Maximum is 32256 bytes.
Global variables use 1424 bytes (69%) of dynamic memory, leaving 624 bytes for local variables. Maximum is 2048 bytes.
```

We still have plenty of PROGRAM STORAGE MEMORY space left, since we only used 33%. Means we can still add things to it. Maybe we can have an option to update our Clock with Satellite Time (I do not have a Sattelite receiver, so I cant do it for now)

Anyway, this is the original target. We have completed our Digital Alarm Clock with Thermometer function

Next we need to clean up the codes and optimize our PROGRAM codes whenever possible.

We can upload this PROGRAM into a standalone ATMEGA328 chip and run the chip without the Arduino Board. We can make our own circuit board and case to hold the chip and the OLED. We can have different type of tactile buttons and resistors on our own circuit board.

We can also use small coin/lipo boosted to 5V with capacitor instead of the USB power supply and we can have a standalone real-life commercial product from this PROGRAM