Arduino OLED Clock

1. A Menu Based System
2. Time/Date Setting in both 12hours and 24hours mode (with automatic hour conversion)
3. A Running Time/Date in both 12hours and 24hours mode (with automatic hour conversion)
4. Alarm time setting and Alarm Monitoring/Trigger
5. A Countdown Timer (seconds) with initial Time Setting
6. Alarm Sound Duration Setting and Testing

Hardware:
1x Arduino Uno
1x Solderless Breadboard

    Jumper wires

1x Active Buzzer

6x Tactile Button
6x Resistor 10Kohm

1x SSD1306 OLED i2c 64x128pixel

Optional:

1x Thermistor 10Kohm Beta=3380
1x Resistor 10KOhm

Project: **PACKAGE_A Arduino OLED Clock**
          :
          :
          : by TeakSoon Ding for STEMKRAF (OCT-2021)

Arduino OLED Clock

This project will be built in many stages. We will start building up various parts, accumulating them until the entire project is completed

```
01. pa0A_buzzer_test                  - active buzzer test
02. pa0B_button_digital_raw      - raw digital button reading
*03. pa0C_button_analog_raw      - raw analog button reading
04. pa0D_button_process          - process raw button readings
05. pa0E_oled_libsk.ino          - OLED with LIBSK_OLED Library
05. pa0F_libsk_oled_example      - LIBSK_OLED Library example
06. pa0G_clock_button_oled       - Show processed button on OLED
07. pa0H_clock_menu              - Clock Menu System
08. pa0I_clock_function_ext      - Clock "ext" function
09. pa0J_clock_function_timer    - Clock Countdown Timer function
```

10. pa0Z_arduino_oled_clock
- Fully functional Arduino OLED Clock with Alarm function together with the earlier "ext" and Countdown Timer functions, extended from Pa0G, P0H, P0I, P0J

11. pa0Z_arduino_oled_clock_digital_pin
- Same as 10. except that, alternative way to make the Navigation Buttons (Left/Bottom/Right/Up) by using the Digital Pins instead of Analog Pin
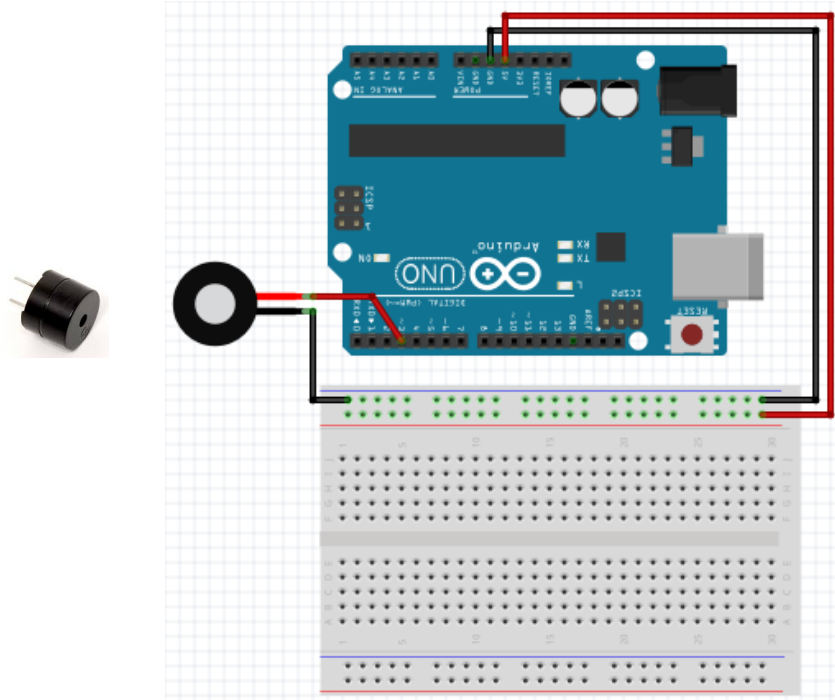
Bonus...

12. pa0Z_arduino_oled_clock_therm_analog
- the pa0Z_arduino_oled_clock is now being extended to have additional Thermometer function with Analog Clock Style display (Pixel Graphics Mode)

Program: pa0A_buzzer_test
   (1/2): active buzzer test
          :
          : by TeakSoon Ding for STEMKRAF (OCT-2021)

Active Buzzer:
The Active Buzzer will automatically make sound when it receives electricity.
Active buzzer have polarity, longer shorter leg connect to Ground (-ve)

Passive Buzzer/Speaker: ?
The Passive Buzzer will only make sound when we make vibrates the speaker
diaphragm with series of ON/OFF voltage (vibrations/seconds or frequencies
in hertz) .

Piezo and regular speakers are Passive.

We can use the tone() function from the Arduino library to make sound with
the Passive Buzzer and Active Buzzer (if you do not like the tone()
function, Arduino allows us to make our own).

Warning: the tone() function from the Arduino Library may interfere with the
PWM for Arduino Pin 3 and Pin 11.

That is the reason why it is better for us to place the Active Buzzer on
Arduino Pin 3 or 11 since these two pins are not suitable to do other PWM
work if tone() function is used.

Program: pa0A_buzzer_test
  (2/2): active buzzer test
         :
   : by TeakSoon Ding for STEMKRAF (OCT-2021)

– Upload this program with the Arduino IDE Software
– and listen

This program will make a 1000 milliseconds "sound" with the Active Buzzer
(or "Hello World") . After that, it does nothing.

If you missed it, press the "Reset Button" on the Arduino Uno to run the
Program again without having to make new Uploads.

Note: There is a limit of how many times you can upload into the Atmega328
micro-controller. If you can avoid upload, by all means avoid it, there is no
need to upload again if we just want to re-start the Program.

Reset Button
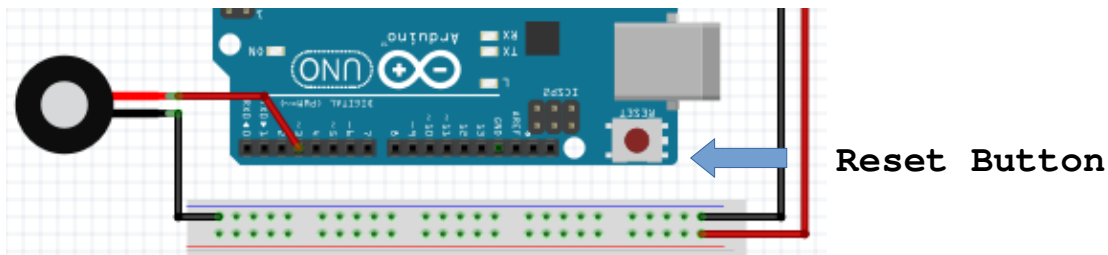
Lets look at some of the program codes.

digitalWrite(BUZZER_PIN, HIGH);
– this will make BUZZER_PIN receive 5V, active buzzer gets 5V and it makes
sound.

digitalWrite(BUZZER_PIN, LOW);
–this will make the BUZZER_PIN receive 0V, active buzzer gets 0V and will
stop making sound.

If you were to place the active buzzer with a (LED+220Ohm resistor), it
will make a light up for 1000 milliseconds and goes off.

Note:
We use the millis() function to pause the between the ON and OFF, we want to
avoid using the delay() function. We will need to use more millis() later in
our project, the delay() function is totally avoidable. The less functions we
use, the smaller and more efficient our program will run.

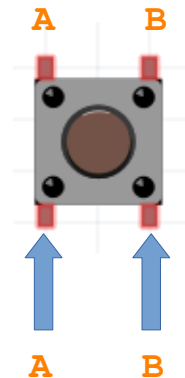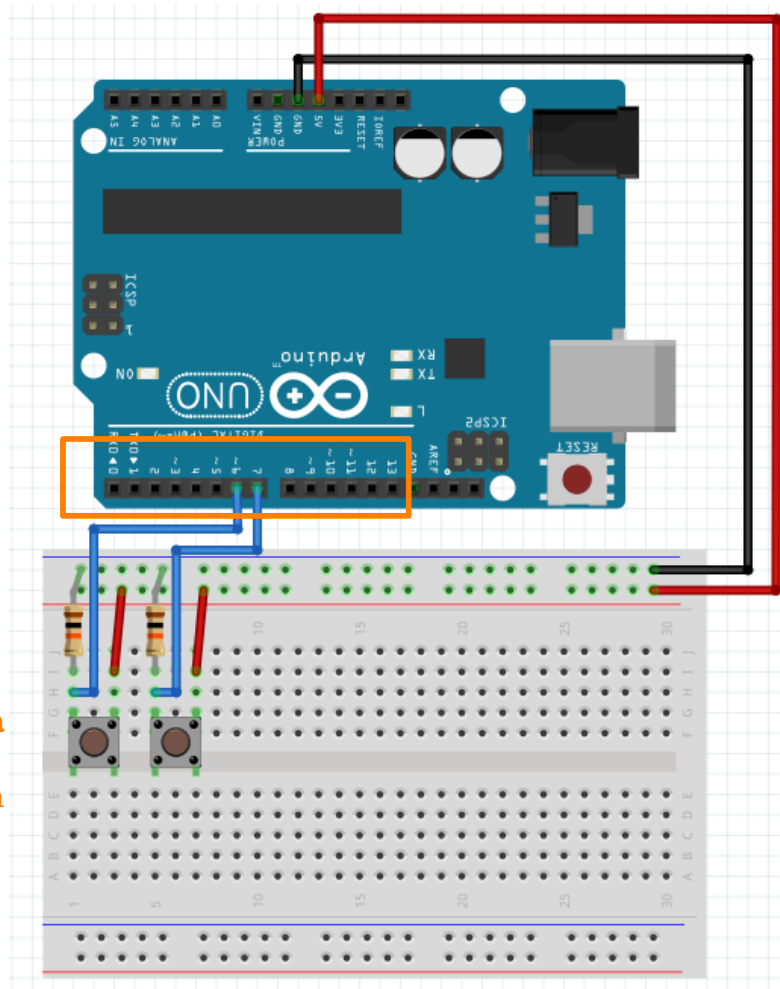Program: pa0B_button_digital_raw
   (1/2): raw digital button reading
            :
            : by TeakSoon Ding for STEMKRAF (OCT-2021)

Analog Pins
Pin A0 to A5

Digital Pins
Pin 0 to 13

Tactile Button
with **10Kohm**
resistor, each

Tactile Button:
The Tactile Buttons are mechanical switch with two disconnected terminals
(A&B). When its button is pressed down the terminals (A&B) are connected,
when button is released, the terminals (A&B) are automatically disconnected.
They can come in many shapes and sizes.

When connected to the physical Arduino Pins, pressing or releasing the
tactile button will connect or disconnect electric flow to the connected pin.
In this project, we have 2 Tactile Buttons connected to 2 Digital Pins

Digital Pins:
These are physical connectors on the Arduino Uno and is physically linked to
the atmega328 micro-controller pins,

Each Digital Pin can have one of these two states:
1. **HIGH** Voltage (5v or very near to 5v)
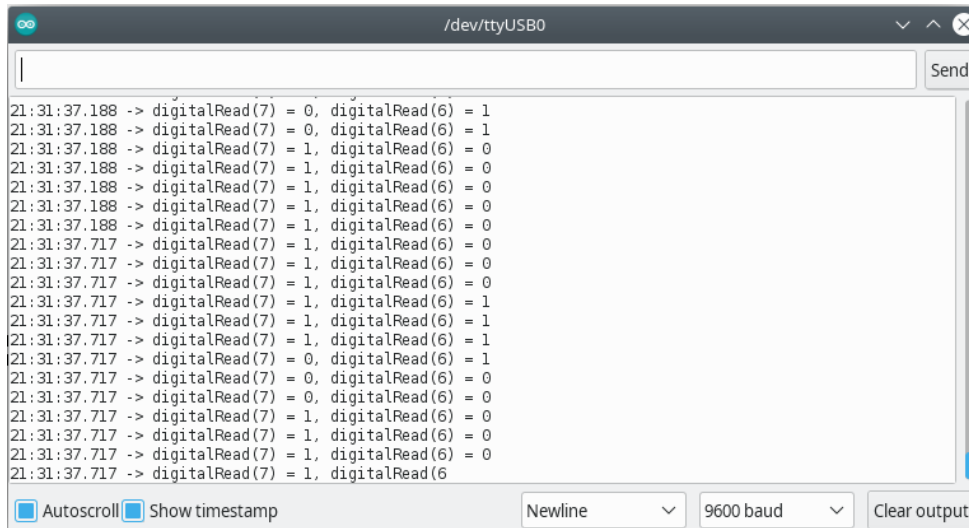2. **LOW** Voltage (0v or very near to 0v)

Note: Do not use Pin 0 and 1 as they are used for Serial Communication. We
can use the Analog Pins as Digital Pins but we cannot use Digital Pins as
Analog Pins

Program: pa0B_button_digital_raw
  (2/2): raw digital button reading
          :
          : by TeakSoon Ding for STEMKRAF (OCT-2021)

– Upload this program with the Arduino IDE Software
– Open the Serial Monitor from the Arduino IDE Software
– Watch the Serial Monitor Screen
– Press/Hold and Release each of the buttons

This program reads from the Arduino Digital Pin 6 and Pin 7 (where the Tactile buttons are connected), then display the raw data two Pins on the Serial Monitor Screen.

"Tactile Button connected to Pin 6"
– When this button is **pressed**, 5volt will flow into Pin 6 .
The code digitalRead(6) will give us **1**

– When this button is **released,** 0volt will flow into Pin 6 .
The code digitalRead(6) will give us **0**

"Tactile Button connected to Pin 7"
– When this button is **pressed**, 5volt will flow into Pin 7 .
The code digitalRead(7) will give us **1**

– When this button is **released,** 0volt will flow into Pin 7 .
The code digitalRead(7) will give us **0**

The value from digitalRead() will let us know whether a button has been pressed or released. Our program will then respond accordingly.

Note:
if you noticed from the Serial Monitor, even if we do the fastest press/release, we will still get a few readings from our single action . What we need is, one reading from one press down action, will do some processing work in the later stage
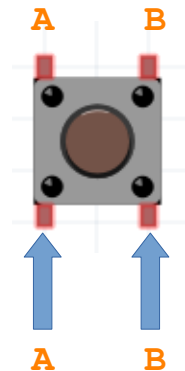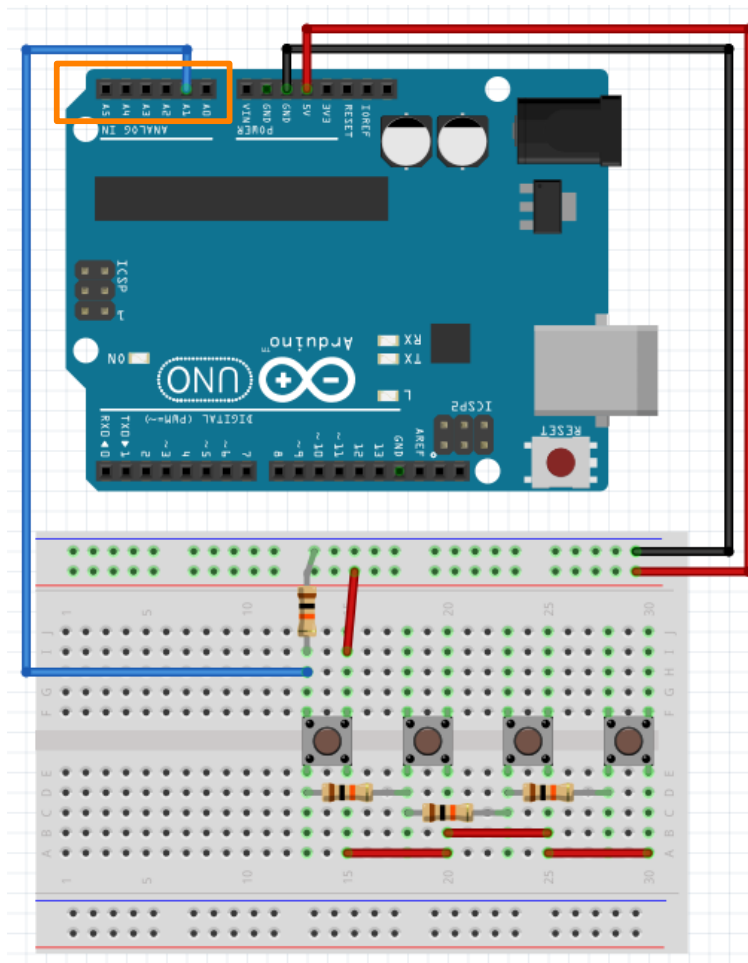
Program: pa0C_button_analog_raw
    (1/2): raw analog button reading
            :
            : by TeakSoon Ding for STEMKRAF (OCT-2021)

Analog Pins
Pin A0 to A5

Tactile Button
with **10Kohm resistor**, each

Tactile Button:
The Tactile Buttons are mechanical switch with two disconnected terminals (A&B). When its button is pressed down the terminals (A&B) are connected, when button is released, the terminals (A&B) are automatically disconnected. They can come in many shapes and sizes.

When connected to the physical Arduino Pins, pressing or releasing the tactile button will connect or disconnect electric flow to the connected pin. **In this project, we have 4 Tactile Buttons, all connected to a single Analog Pin .**

Analog Pins (also known as ADC Pins):
The Analog Value at the Analog Pin are the actual voltage, represented by number 0 to 1023. The reason for 0 to 1023 is because the micro-controller uses 10-bits Memory
- 0volt in Analog Pin (binary 0000000000 = 0)
- 5volt in Analog Pin (binary 1111111111 = 1023)

Since 1023 is 5V, we can actually calculate the Analog Value for the different voltage at the Analog Pin.

Program: pa0C_button_analog_raw
   (2/2): raw analog button reading
           :
           : by TeakSoon Ding for STEMKRAF (OCT-2021)

- Upload this program with the Arduino IDE Software
- Open the Serial Monitor from the Arduino IDE Software
- Watch the Serial Monitor Screen
- Press/Hold and Release each of the buttons
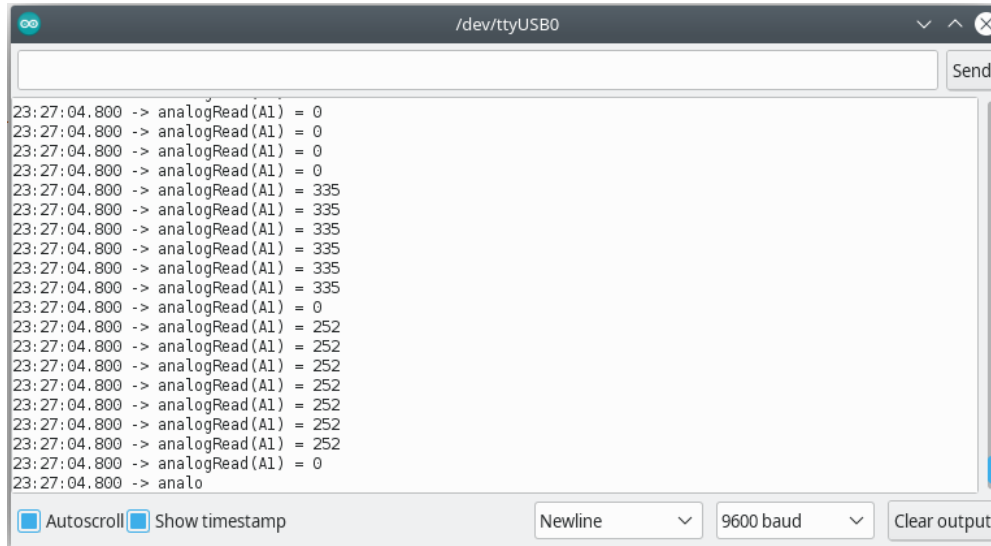
This program reads from Arduino Analog Pin A1 and display the Analog Value to the Serial Monitor Screen.

In this program, all 4 tactile buttons is connected to the same Analog Pin. Each button when pressed, will cause different amount of voltage at the Analog Pin. This is because we have different combined resistors values at each Tactile button. Different amount of Voltage, will give different Analog Value when we retrieve the value by using the **analogRead()** code.

* WHEN WE USE MULTIPLE TACTILE BUTTON ON THE SAME ANALOG PIN. WE NEED TO IDENTIFY EACH OF BUTTONS. THE FOLLOWING, IS WHAT WE MUST DO, OTHERWISE WE DO NOT KNOW WHICH BUTTON IS PRESSED *

Hold-down one of the button (without releasing), we will see a lot of readings scrolling on the Serial Monitor, sometimes with some small variations. **RECORD** the **number that appears the most** on the Scrolling Serial Monitor Screen while holding down one particular button. **This number will represent that button**.

Do this, for all the 4 buttons and record the numbers associated with each of the button. This RECORDED number can vary from one setup to another, we will always need to do this process when we use multiple button on a single Analog Pin. (The values of cos can be calculated, we just took the easy way out)

When the **all tactile buttons are released**, the Analog Pin reading **analogRead() will become 0, since it is 0 Volt** .

Program: pa0D_button_process
   (1/3): process raw button readings
            :
            : by TeakSoon Ding for STEMKRAF (OCT-2021)

Issues with tactile buttons:
Tactile buttons are not like push buttons in our Computer Software
Screen. These button are mechanical device with physical parts.
There will be a lot of things we cannot see with our eyes or can
be accurately controlled by physical abilities.

Here are some of the issues we will face when using tactile
buttons unprocessed.

Common Issues(1): Press once but runs Multiple Times
– This is caused my the mechanical nature of the button, codes can
be triggered multiple times even though we press just once
(mechanically, it can send multiple signals to the micro-
controller) . This issue can happen in both Digital and Analog
Pins.

Suggested Fix:
– Use an Indicator. Once button press is detected, immediately
sets an indicator sot that program will not run the code anymore.
When button is released, indicator is switched OFF for fresh
clicks.

– A small delay (few milliseconds) is sometimes inserted after
reading, to minimize potential of mechanical triggers before the
indicator can be set ON.

Common Issues(2): Analog Pin identifies wrong Button
– Analog Pin use a range of values derived from the Voltage on the
Pin to identify a button pressed. Analog Pins sometimes suffer
"voltage jitters", causing random stray voltage on the Analog pin.
The random stray can go into a legal number range of another
button, causing wrong button being triggered (which is
unacceptable)

Suggested Fix:
We perform multiple readings from Analog Pin into an array and
then, is SORTED. The stray values will go to either top or bottom
of the sorted Array. The middle array element should be the most
reliable reading. For example, a "5-readings" sorted array can
filter out at least 2 strays.
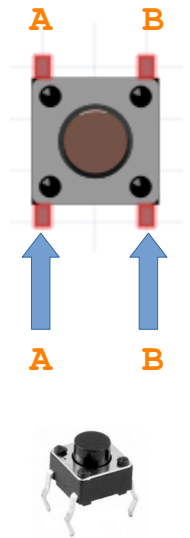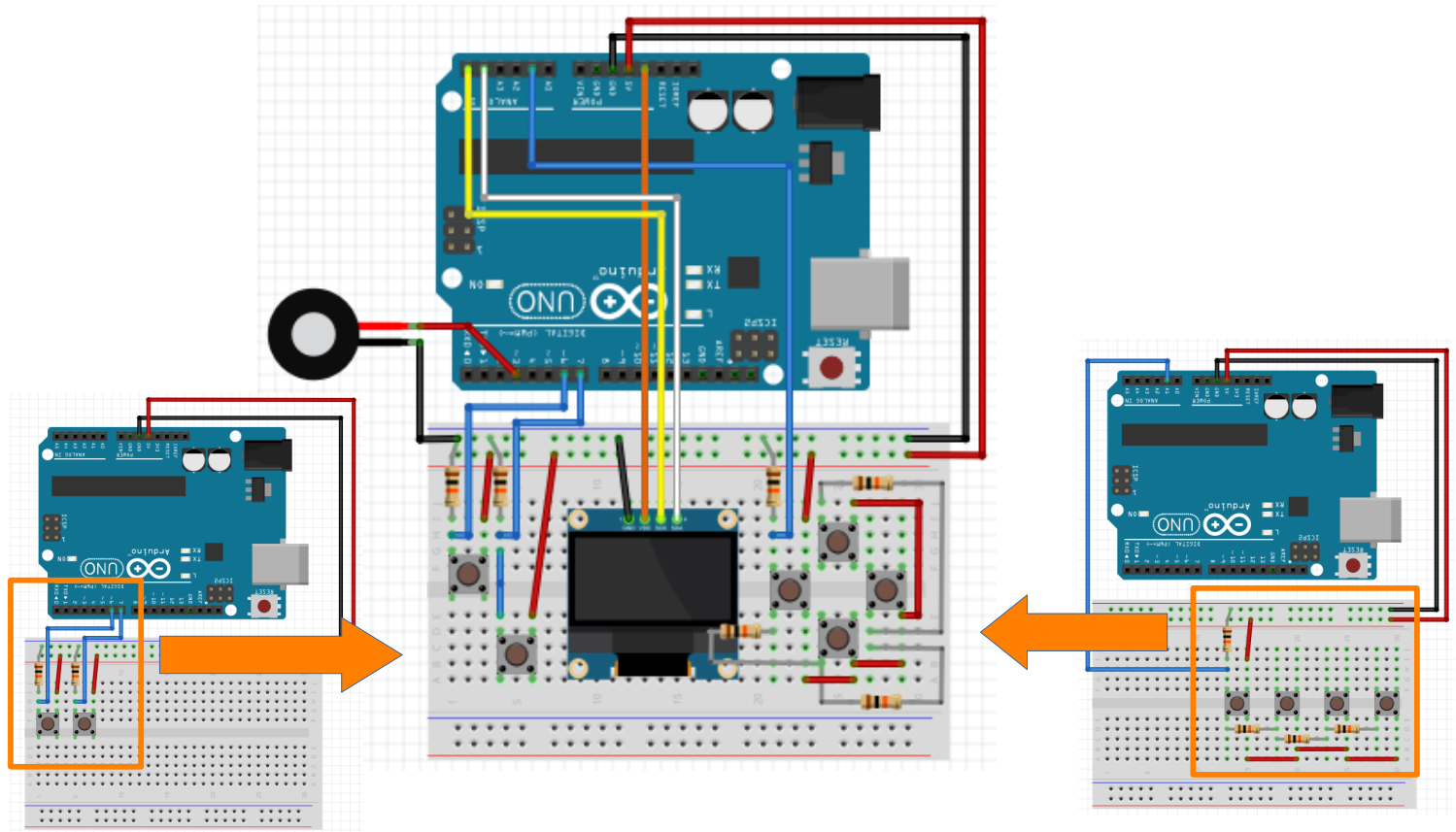
Program: pa0D_button_process
   (2/3): process raw button readings
         :
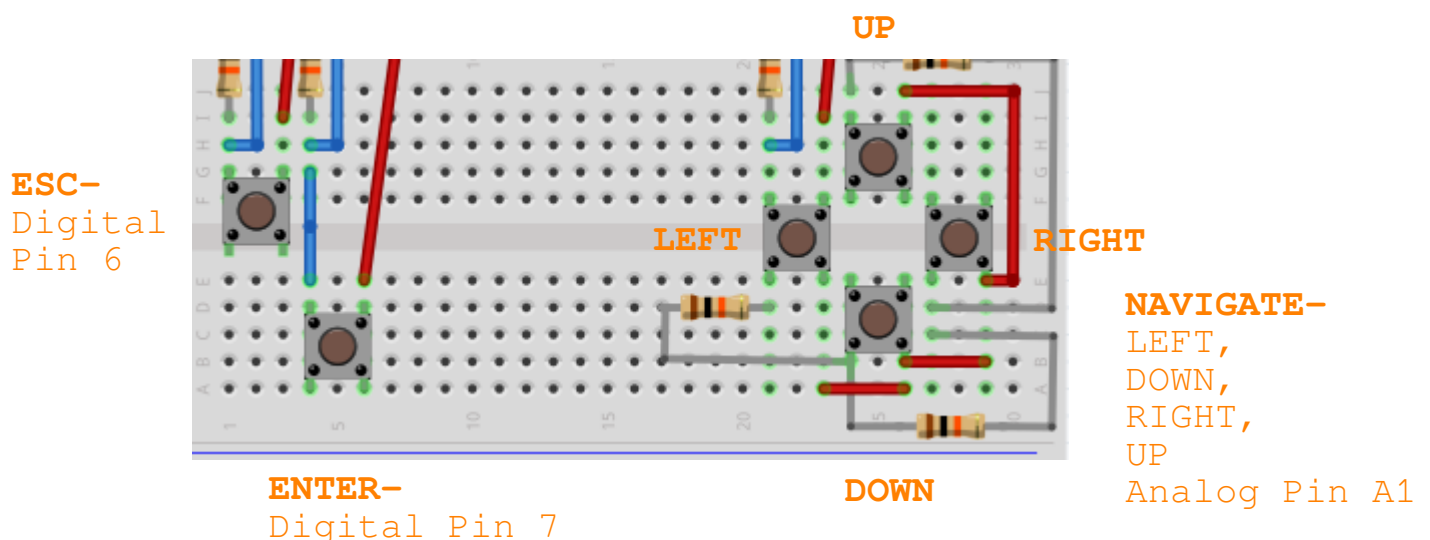   : by TeakSoon Ding for STEMKRAF (OCT-2021)

Since we are making a Clock with user interactions, lets arrange our buttons a little bit, making it is easier for user to operate

Tactile Button Processing:
A typical electronic device will have some kind of Navigation buttons (1.Left, 2.Down, 3.Right, 4.Up), 5.Enter Button and 6. Escape Button

In this program, ENTER and ESC will be using a Digital Pin each, while the 4 Navigation buttons, will be using a single Analog Pin.


UP

ESC–
Digital
Pin 6

LEFT

RIGHT

NAVIGATE–
LEFT,
DOWN,
RIGHT,
UP
Analog Pin A1

ENTER–
Digital Pin 7

DOWN

Program: pa0D_button_process
  (3/3): process raw button readings
          :
          : by TeakSoon Ding for STEMKRAF (OCT-2021)

- Upload this program with the Arduino IDE Software
- Open the Serial Monitor from the Arduino IDE Software
- Watch the Serial Monitor Screen
- Press/Hold and Release each of the buttons
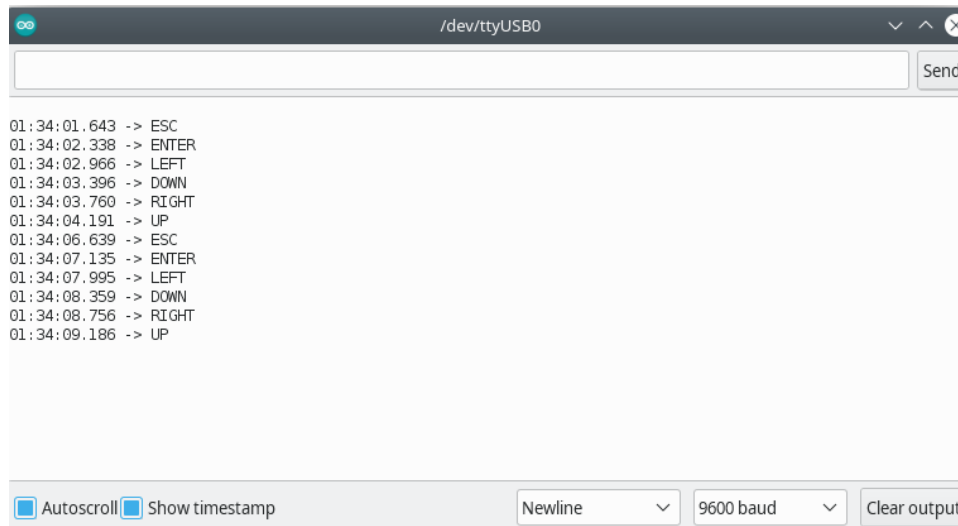
This program reads from Analog Pin A1 (Left/Down/Right/Up) Navigation Button, Digital Pin 7 (Enter) and Digital Pin 6 (Esc)

When any of the connected button is pressed, it is identified and displayed on the Serial Monitor Screen.

If you have noticed, the Program no longer output non-stop values to the Serial Monitor Screen. It will only display on the screen once when pressed or released. Stray data are also being filtered out and you do not see multiple-trigger from a single press anymore.

NOTE:
The Analog Pin is not as stable as the Digital Pin as it use a range of voltage, voltage can spikes or drops or sometimes influenced by external interference like magnets or other electrical devices (that is why in this project the active buzzer is placed far away from them). Then why still use it ? Because we can have multiple buttons to a single Pin, imagine if we have 20 buttons, we will not have enough digital pins for that on Arduino Uno.

The Digital Pins are more stable as it does not care much about spikes or drops as it will just take 5V and 0V as HIGH and LOW reading. If you look at the source code, even the processing of Digital Pins are much easier than Analog Pins.

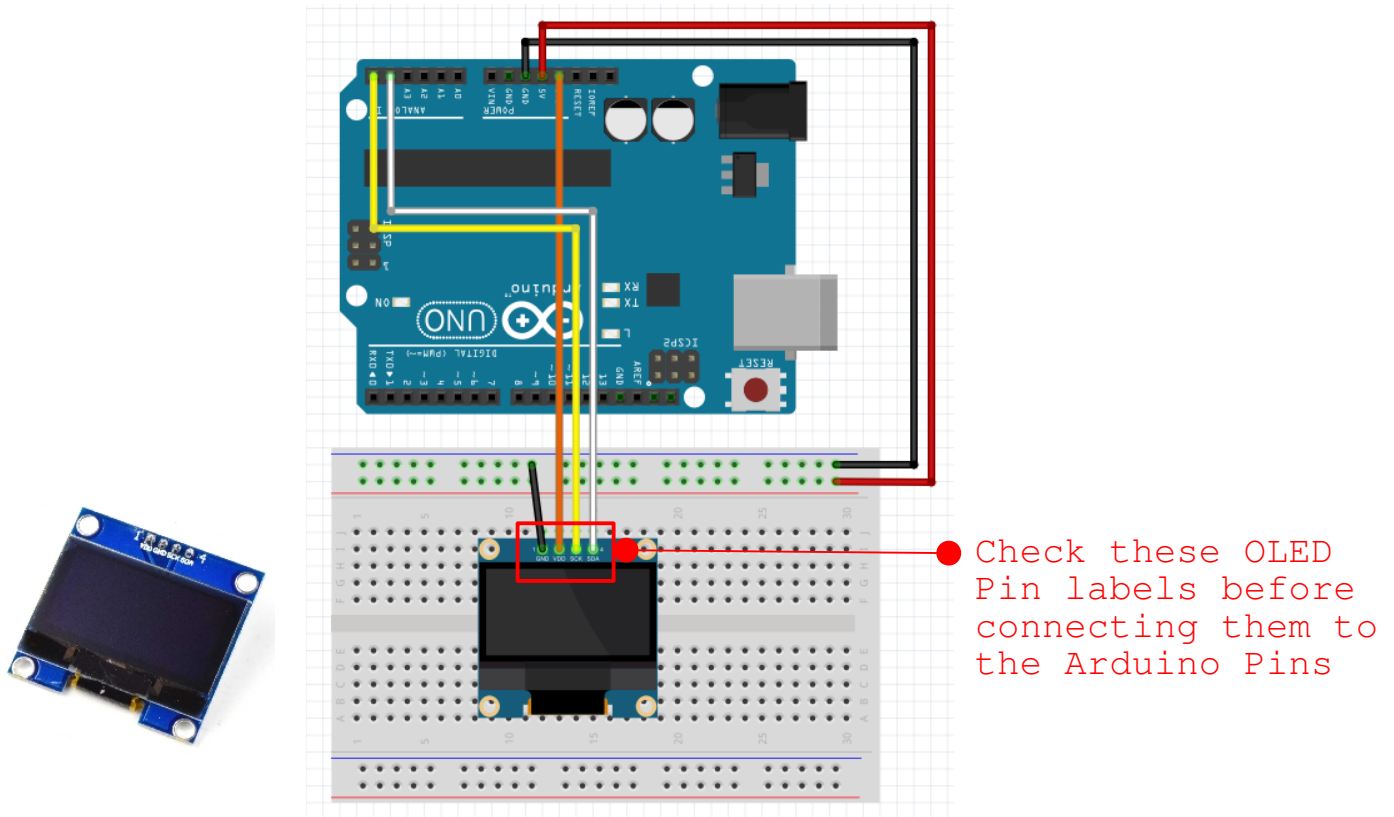Check these OLED Pin labels before connecting them to the Arduino Pins

SSD1306 OLED i2c 64x128pixel:
This OLED Display Module comes with the i2c interface. The i2c device requires 4 wires and the micro-controller must have support this interface.

1. Ground (-ve)
2. VCC (+ve) usually 3.3v, sometimes 5v to be safe, we use 3.3v
3. SCL   - System Clock
4. SDA   - Data

The SCL pin is connected to Arduino Pin A4 .
The SDA Pin is connected to Arduino Pin A5 .

In some Arduino Uno board have special pin for SCL and SDA, next to Pin 13.

WARNING:
There are many manufacturer for this device. Most of these i2c OLED modules looks the same but **the position of the Pins may not be the same.** Always read the Pin labels before connecting it to your Arduino Uno

There are also the SPI version with 6 Pins, in our example we use the i2c version that uses 4 Pins.

Program: pa0E_oled_libsk
   (2/2): OLED with LIBSK_OLED Library
          :
          : by TeakSoon Ding for STEMKRAF (OCT-2021)

The Standard Arduino Library does not have any functions to send output the the OLED Screen. STEMKRAF has programmed a Library (LIBSK OLED) that has functions that can send output to this OLED Screen.

This program uses functions from the LIBSK OLED Library, **we need to install the LIBSK OLED Library** into our Arduino IDE Software before we can upload this Program.

1. Find [Program Folder] from the Arduino IDE Software,
- "File|Preferences|Sketchbook Location: [Program Folder]"
2. Use the Computer File Browser,
- Look for existing folder "libraries" inside the [Program Folder]
3.Inside the "libraries" folder,
- Create a new folder named "libsk_oled"
4.Copy "libsk_c_oled.h" file into the newly created "libsk_oled" folder.
DONE!!!

- You can now Upload this program with the Arduino IDE Software
- Watch the OLED Screen (no more Serial Monitor)

We only need 3 lines of code to get the OLED to display something on its screen with our "LIBSK OLED" Library.

The following code, at the top of our Program will allow us to use the functions from the LIBSK OLED Library in our Program.

#include <libsk_c_oled.h>

Next, we only need to run the following code once before other LIBSK functions. This is to connect and initialize the OLED device

sk_oled_begin();

Then, we run the following code to display a message "SSD1306 OLED" on the OLED screen.

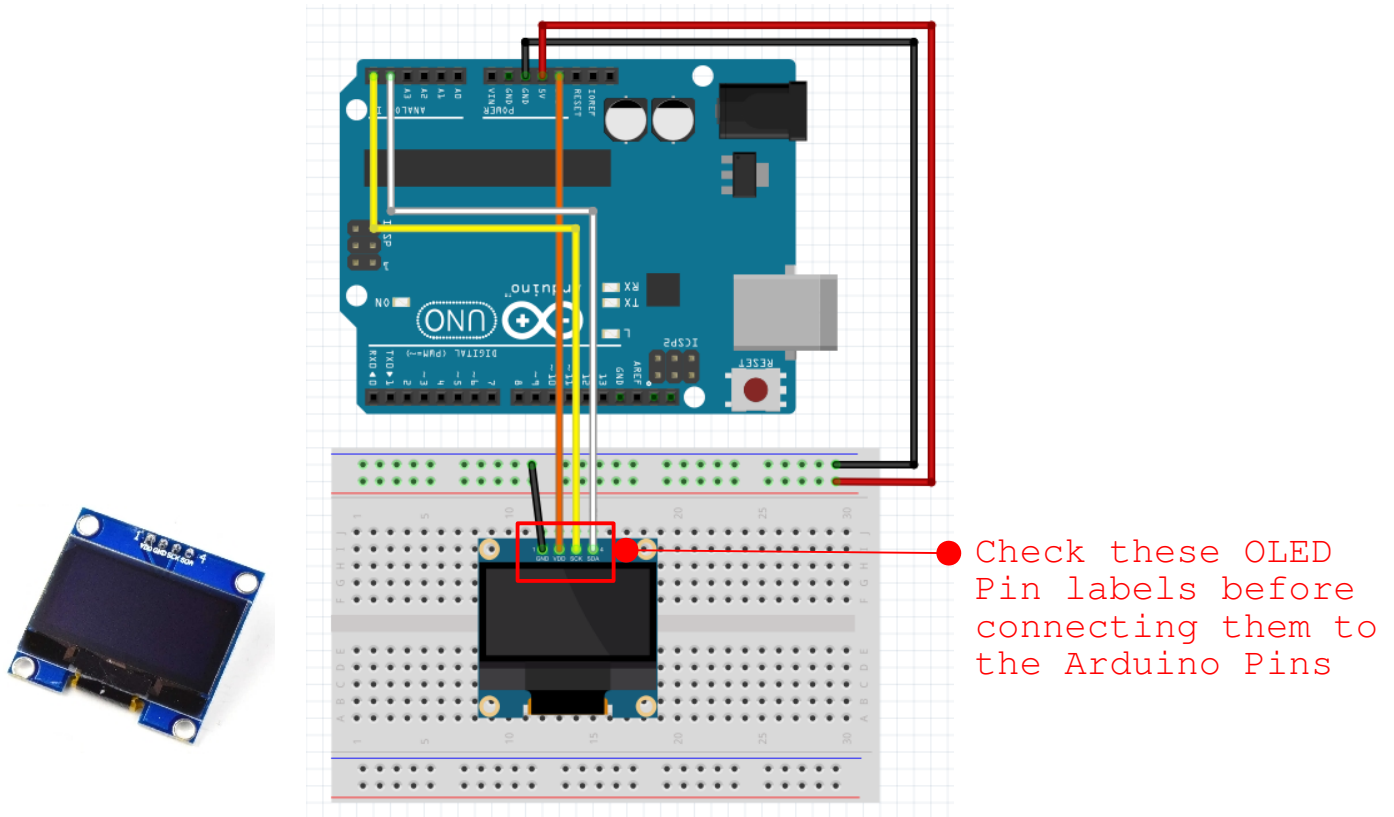**sk_oled_showString(1,5,"SSD1306 OLED");**

Program: pa0F_libsk_oled_example
   (1/2): LIBSK OLED example (all functions demo)
          :
          : by TeakSoon Ding for STEMKRAF (OCT-2021)

Check these OLED
Pin labels before
connecting them to
the Arduino Pins

The "LIBSK OLED" OLED Library ("C" Version)
This library is kept as small as possible. In order to keep it as small and
as simple as possible, it can only display ASCII char (32 to 126) from a
single built-in font array and display flexible 8-bit bitmap on this specific
OLED Display. In total, this Library only has 7 functions.

1.**sk_oled_begin()**
- Run this just once, before other library display functions

2.**sk_oled_clearScreen()**
- Clears the entire OLED screen

3.**sk_oled_showBitmap(_row,_col,_a_byt,_len,_len_idx)**
- Display 8-bit bitmap (8pixel veritcal, 1-128 pixel horizontal)

4.**sk_oled_showChar(_row,_col,_cha)**
- Show a ASCII Char (32 to 126)

5.**sk_oled_showString(_row,_col,_str)**
- Show a String (array of Char terminated by NULL)

6.**sk_oled_showChar_L(_row,_col,_cha)**
- Same as "_showChar()" but doubled in size

7.**sk_oled_showString_L(_row,_col,_str)**
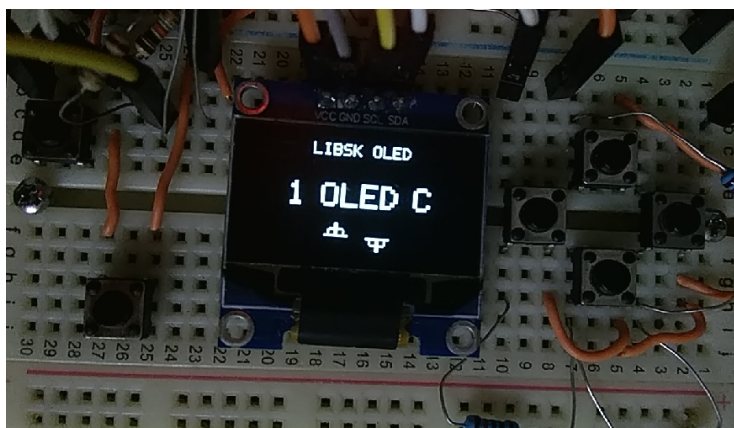- Same as "_showString()" but doubled in size

Program: pa0F_libsk_oled_example
   (2/2): LIBSK OLED example (all functions demo)
          :
          : by TeakSoon Ding for STEMKRAF (OCT-2021)

- Upload this program with the Arduino IDE Software
- Watch the OLED Screen

This Program demonstrates how to use all the functions from the LIBSK OLED
"C" Version Library.

**If you have noticed, this OLED Library is very small when compared to many
other popular Arduino 3rd Party OLED Libraries. In order to keep this library
as small as possible, many error corrections are not coded intentionally in
this Library. Use within the following limits**

_row
(0 to 8 - top to bottom),Char and Bitmap
(0 to 7 - top to bottom),"L" Char (uses 2 rows each)

_col
(0 to 127 - left to right),Bitmap (each Bitmap Byte is 1pixel)
(0 to 123 -left to right), Char (each Char is 5pixel)
(0 to 118 -left to right), "L" Char (each "L" Char is 10pixel)

_cha
ASCII Char (32 to 126)

_str
Array of Chars, terminated by NULL.
1-pixel gap between the Chars, 2-pixel gap between "L" Chars

sk_oled_showBitmap(_row,_col,_a_byt,_len,_len_idx)
_a_byt
Array of Byte (8-bits each) for
_len
Total elements in _a_byt to be used for _showBitmap()
_len_idx
Starting index from _a_byt (0 to _len-1) - for _showBitmap()
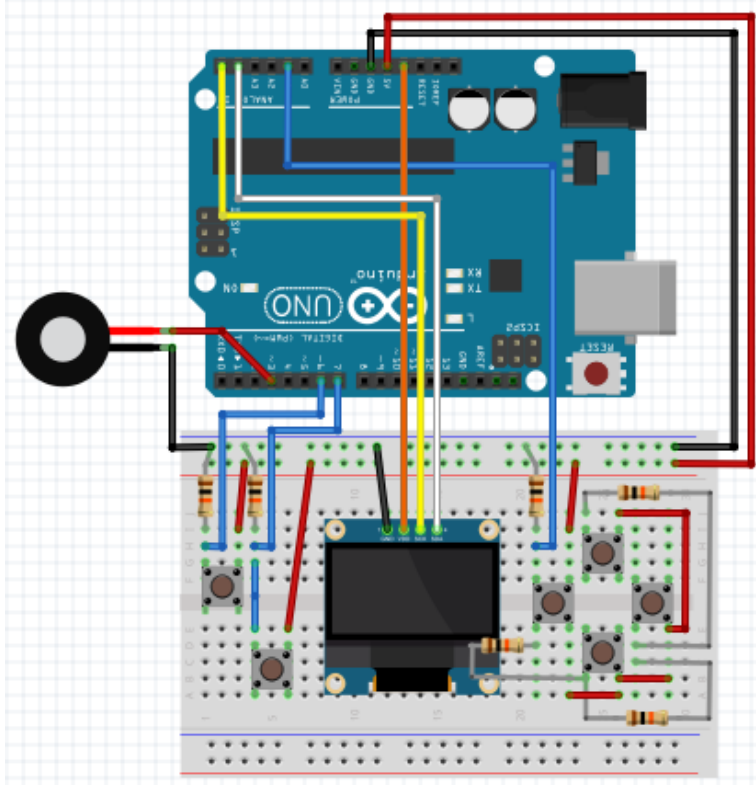
Program: pa0G_clock_button_oled
  (1/2): Clock buttons and OLED
        :
        : by TeakSoon Ding for STEMKRAF (OCT-2021)

Our atmega328 micro controller will eventually be be detached from the Development Desktop Computer for deployment. When deployed we wont have the Serial Monitor for display anymore.

Instead of using Serial Monitor, we use the OLED Screen for display. Program size will also be significantly reduced when Serial Class from the Arduino Library is not used.

Program: pa0G_clock_button_oled
   (1/2): Clock buttons and OLED
          :
          : by TeakSoon Ding for STEMKRAF (OCT-2021)

– Upload this program with the Arduino IDE Software
– Watch the OLED Screen
– Press and Release each of the buttons
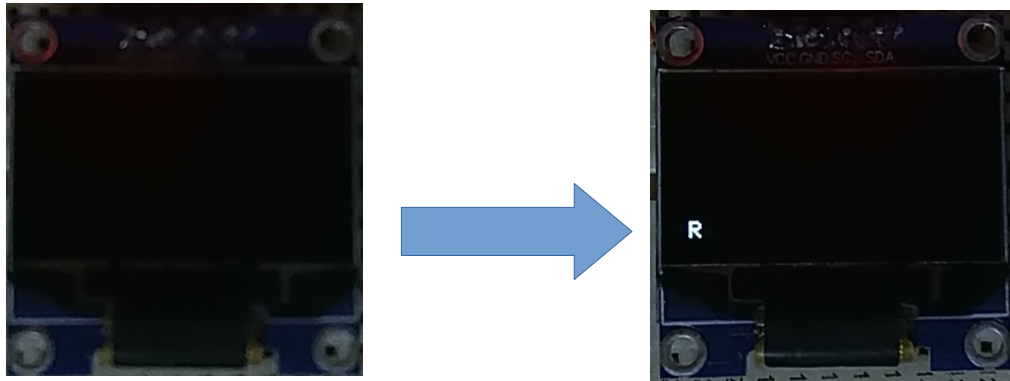
Here we started off with blank OLED screen. When any of the button is pressed, we do some processing to identify the button. Once identified, we send the button identity to the OLED Screen.

sk_oled_showChar(0,7,'?');

? is the identity of the button pressed.
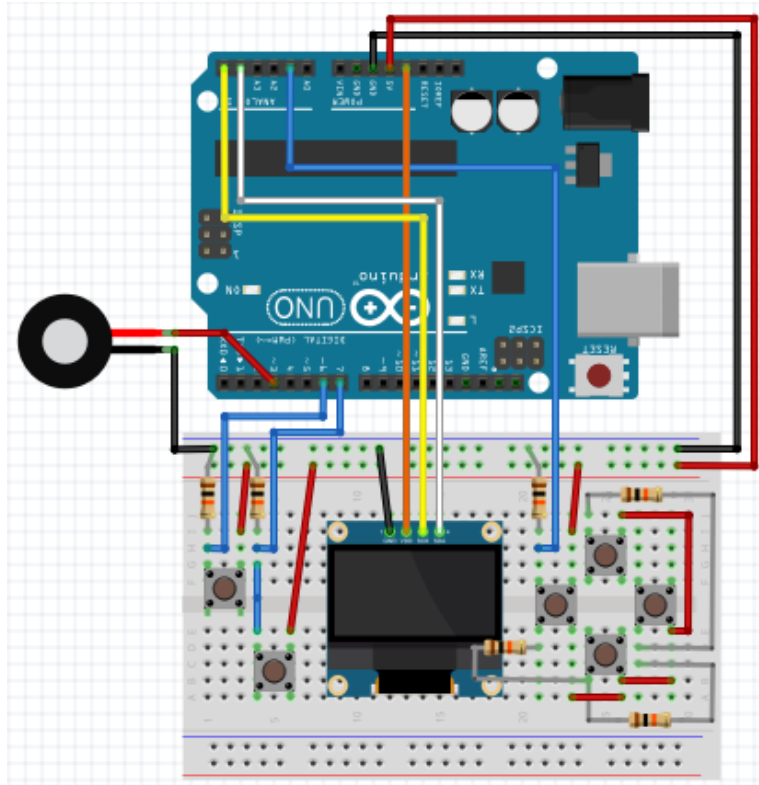
Program: pa0H_clock_menu
   (1/2): Clock Menu
          :
          : by TeakSoon Ding for STEMKRAF (OCT-2021)

Most device will have menu for user to select. This Program will display a
Menu on the OLED Screen. The buttons will be used to navigate between the
Menu Options.

This will be the foundation for all micro-controller device that requires
Menu Selections.
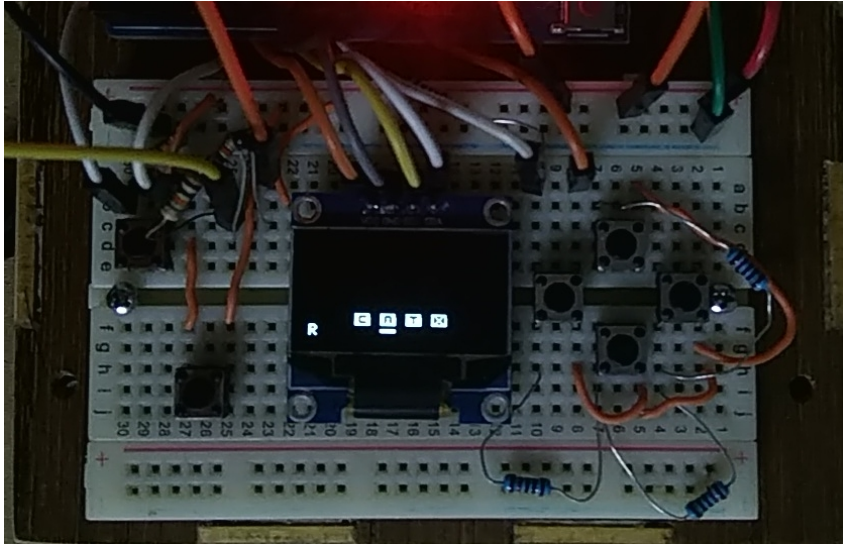
Program: pa0H_clock menu
    (2/2): Clock Menu
            :
            : by TeakSoon Ding for STEMKRAF (OCT-2021)

- Upload this program with the Arduino IDE Software
- Watch the OLED Screen
- Press and Release each of the buttons

MENU MODE:

ESC is pressed – first Menu Option will be selected

LEFT or **RIGHT** Button is pressed – Menu Selector Bar will move between the Menu Options

UP and **DOWN** Button is pressed – Not used in Menu Mode

ENTER is pressed – Menu will be disabled and the Function Mode will be activated.

FUNCTION MODE:

ESC is pressed – return to Menu Mode

ENTER is pressed – in this program, it will also return to Menu Mode as we have not coded anything yet. Normally Enter will be used as "Save and return" or to perform some other task

LEFT or **RIGHT** is pressed – in this program, they do nothing. Normally Left and Right button will be used to move between different items in the Function Option or to perform some other task

UP and **DOWN** is pressed – in this program, they do nothing. Normally Up and Down button will be used to increase or decrease values (editing) in the Function Option or to perform some other task)

NOTE:
There is NO FIXED RULES on how to make a Menu or how buttons should be operated or behave. This Program is just showing an example on how buttons can be used together with the OLED to make a Menu driven Electronic Device.
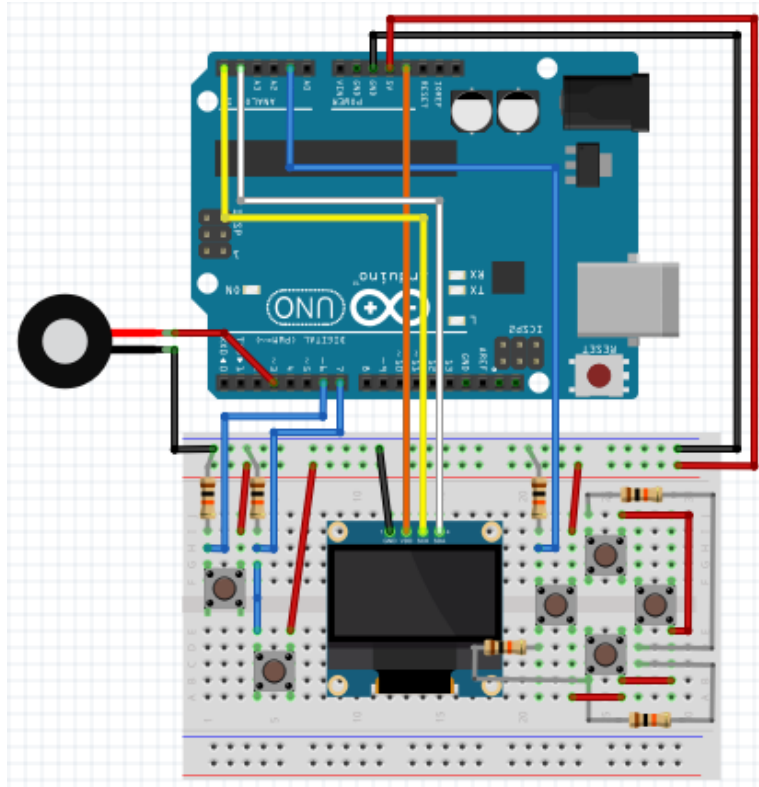
Program: pa0I_clock_function_ext
   (1/2): Clock Function "ext"
         :
         : by TeakSoon Ding for STEMKRAF (OCT-2021)

This is one simple function where after we press the ENTER button from the Menu Mode, it goes into this "EXT" function.

In this function we can set value for Buzzer Alarm timeout duration. We can also test out the duration in this Function

Setting value is very common in many devices with user interactions. The purpose of this is to demonstrate coding of a very simple data editing with buttons and OLED
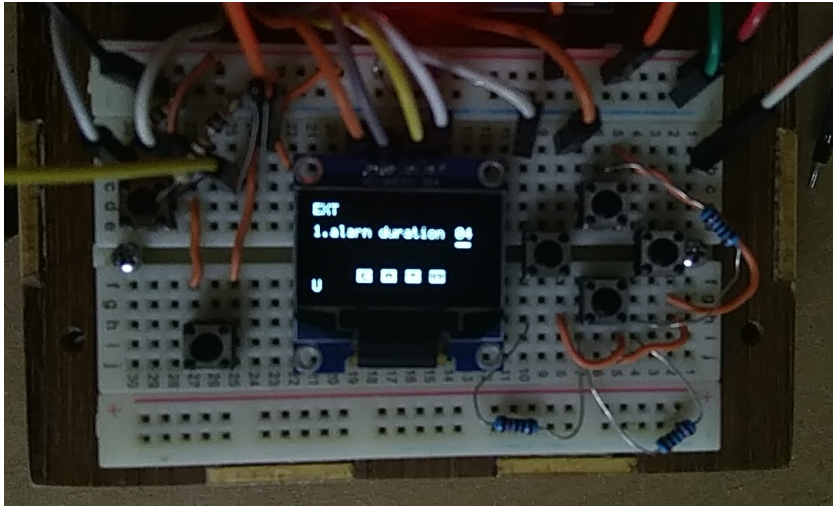
Program: pa0I_clock_function_ext
  (2/2): Clock Function "ext"
         :
         : by TeakSoon Ding for STEMKRAF (OCT-2021)

– Upload this program with the Arduino IDE Software
– Use the tactile buttons

This is the EXT function after pressing ENTER key from Menu Mode

As you can see, the cursor selection is now on the "04", pressing UP Button
will increase the value and DOWN Button will reduce the number.

When you press the LEFT or RIGHT Button Cursor selection move to a different
location. This is controlled by our program.

This position shown on this picture is for us to test the Alarm duration. If
we press ENTER button here, the BUZZER will make sound for 4 seconds. (if
you do not wish to listen for 4 seconds, you can hit any button to
immediately stop the sound, this ability has already been programmed, you can
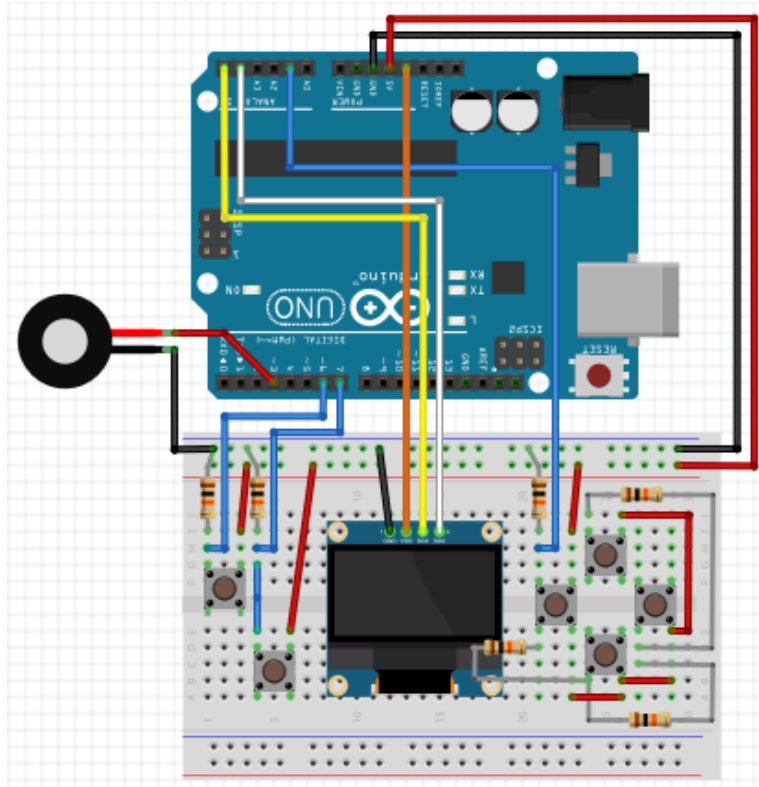check the source codes to find out how it is done)

Program: pa0J_clock_function_timer
   (1/2): Clock Function Countdown Timer
          :
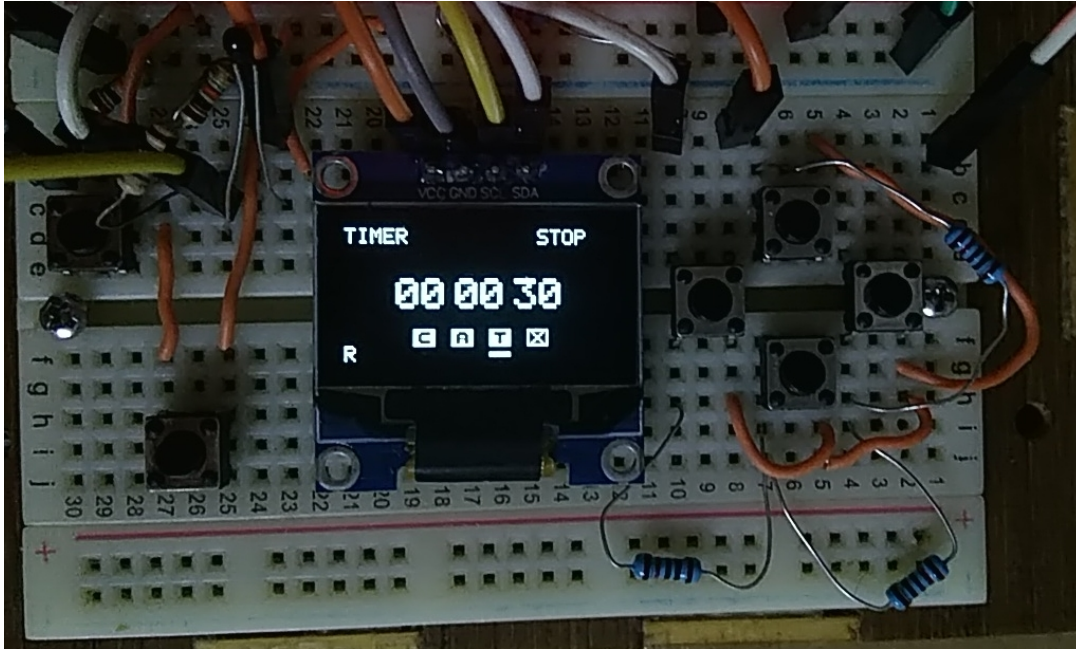   : by TeakSoon Ding for STEMKRAF (OCT-2021)

In this Count Down Timer function, we start make use of the processing loop, updating our live countdown timer with the non-blocking millis() function.

Program: pa0J_clock_function_timer
   (2/2): Clock Function Countdown Timer
          :
          : by TeakSoon Ding for STEMKRAF (OCT-2021)

– Upload this program with the Arduino IDE Software
– Use Buttons

This is the COUNTDOWN TIMER function after pressing ENTER key from Menu Mode

Once inside this function, you can use LEFT/RIGHT button to move cursor to
change the TIMER duration, HH:MM:SS

UP/DOWN Button will change the initial countdown numbers. To start the
Countdown Timer, just hit ENTER button, Counter will reduce by 1 seconds
until the whole time becomes zero and Alarm will start.

When Countdown is active, you cannot move the editing cursor. You can pause
the Countdown by pressing ENTER. Once paused, you can change the values.

To exit the Countdown Timer, press the ESC BUTTON
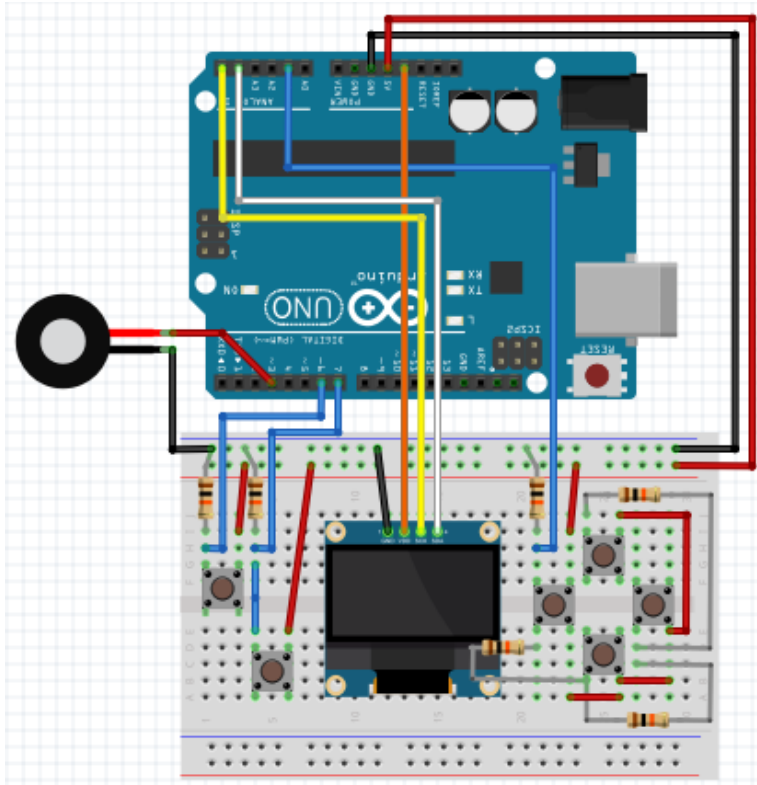
Program: pa0Z_arduino_oled_clock
  (1/2): Arduino OLED Clock
        :
        : by TeakSoon Ding for STEMKRAF (OCT-2021)

And finally a fully functional Arduino OLED Clock,
1. A Menu Based System
2. Time/Date Setting in both 12hours and 24hours mode (with automatic hour conversion)
3. A Running Time/Date in both 12hours and 24hours mode (with automatic hour conversion)
4. Alarm time setting and Alarm Monitoring/Trigger
5. A Countdown Timer (seconds) with initial Time Setting
6. Alarm Sound Duration Setting and Testing

The entire Program size is as below, (very comfortable below 8K size)

Sketch uses 7542 bytes (23%) of program storage space. Maximum is 32256 bytes.
Global variables use 391 bytes (19%) of dynamic memory, leaving 1657 bytes for local variables. Maximum is 2048 bytes.
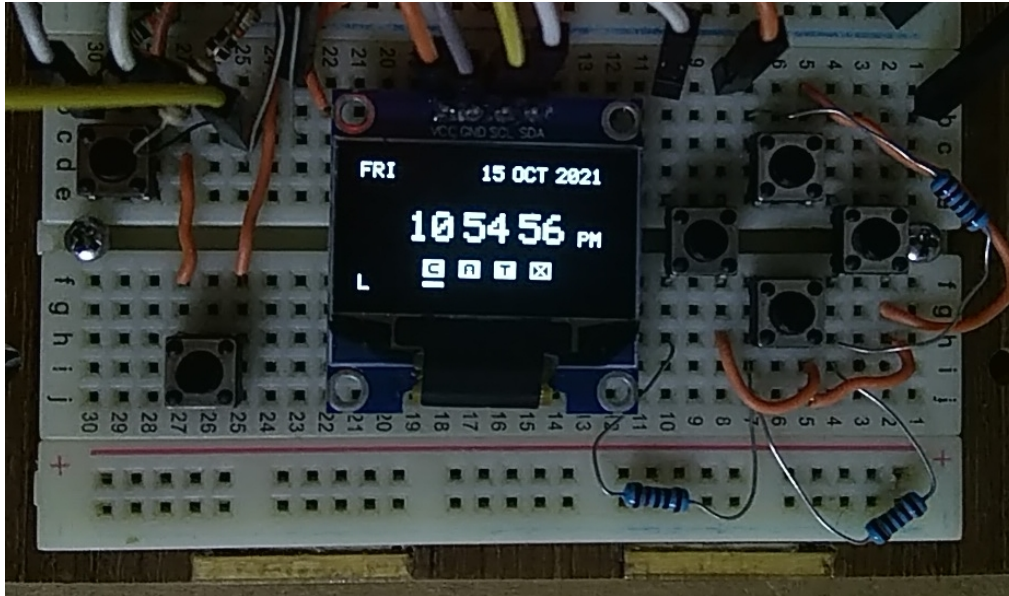
Program: pa0Z_arduino_oled_clock
  (2/2): Arduino OLED Clock
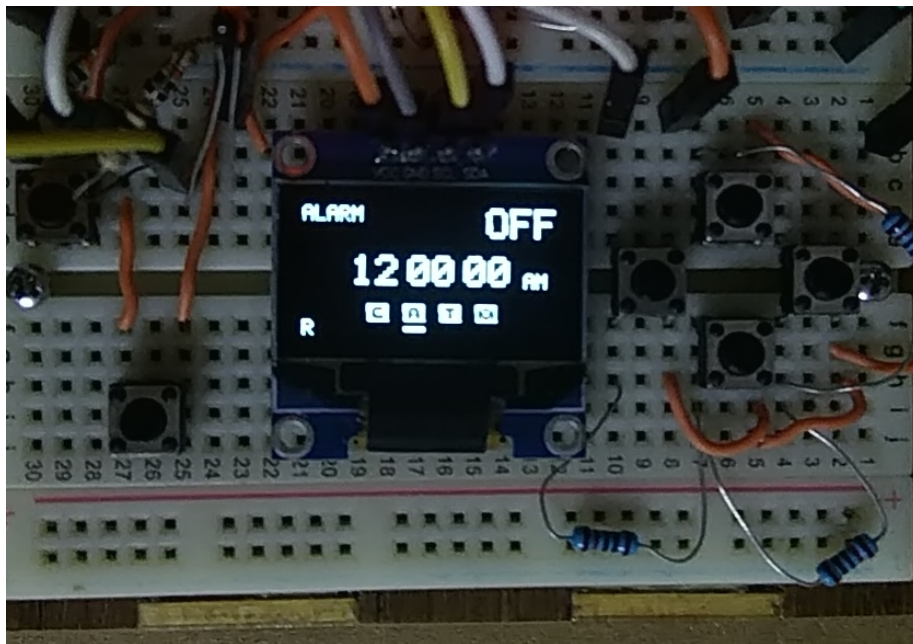   :
         : by TeakSoon Ding for STEMKRAF (OCT-2021)

– Upload this program with the Arduino IDE Software
– Use Buttons

This is a live running clock, updated every 1 seconds. You can change the
Time/Date by pressing Enter button while in Menu Mode. Conversion between 12
Hours and 24 hours mode are automatic since it has been programmed.

Editing the Alarm Time is also the same like the rest of the functions. Press
ENTER from Menu Mode, use LEFT/RIGHT button to move selection cursor between
the items, Use UP/DOWN button to change the values and Hit ENTER again to
save and Exit. If you wish to abandon editing and return to previous value,
simply press the ESC button while editing.
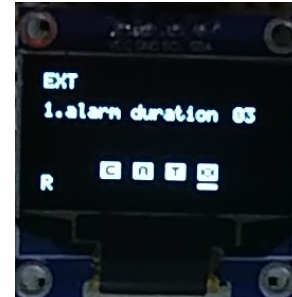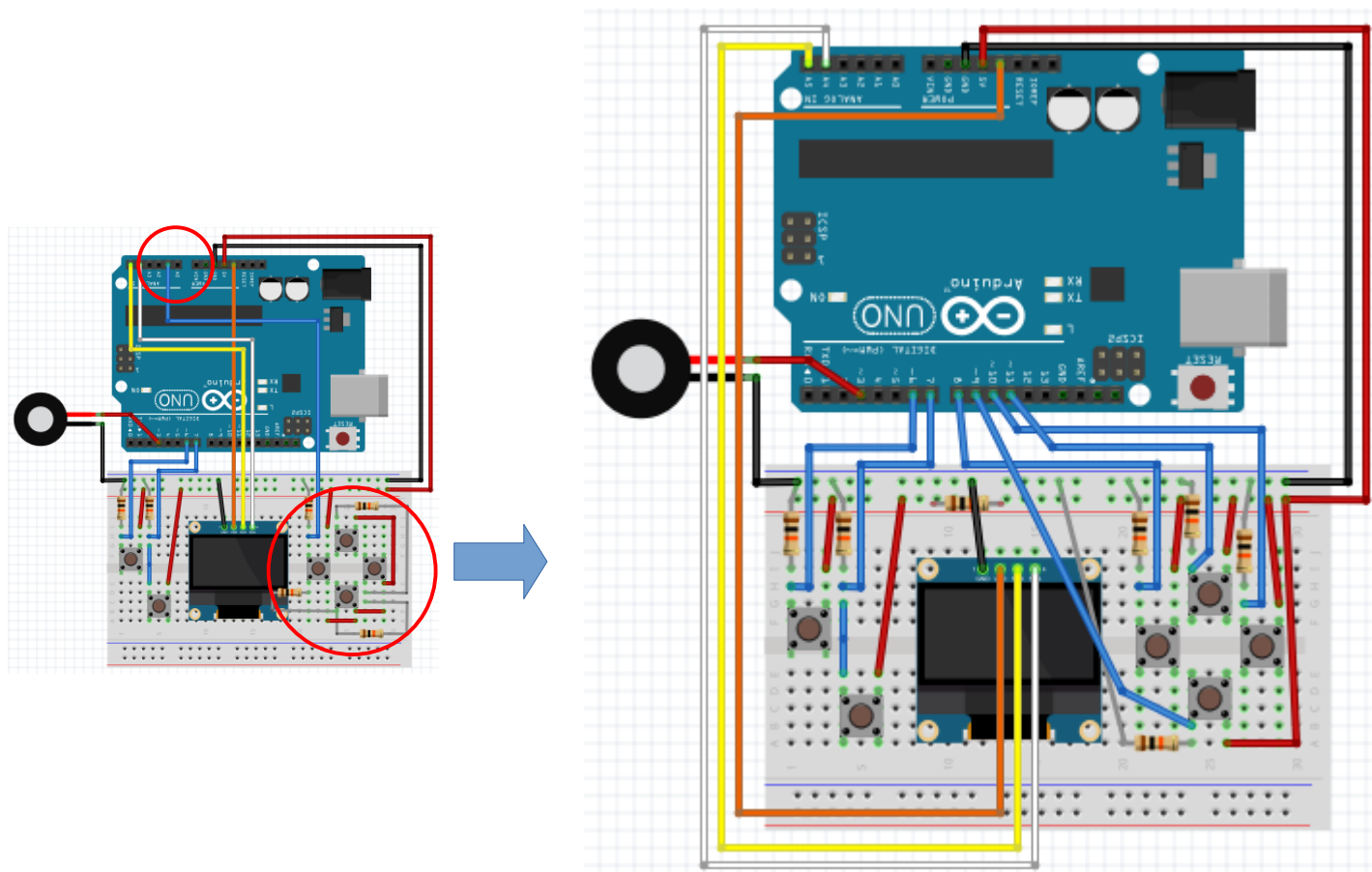
Program: pa0Z_arduino_oled_clock_digital_pin
 (1/1): Arduino OLED Clock using digital pins
           :
           : by TeakSoon Ding for STEMKRAF (OCT-2021)

This version will produce the same Arduino OLED Clock, difference is that Analog Pin has been replaced with Digital Pins. Sometimes Analog Pin can be unstable, so this is the alternative

LEFT BUTTON to PIN 8
DOWN BUTTON to PIN 9
RIGHT BUTTON to PIN 10
UP BUTTON   to PIN 11

Previously all LEFT/DOWN/RIGHT and UP button goes to PIN A1.

You can compare the codes, they are almost identical difference is only at the button processing portion of the code where Analog Button Processing is no longer needed.

STEMKRAF — ARDUINO
https://github.com/teaksoon/stemkraf
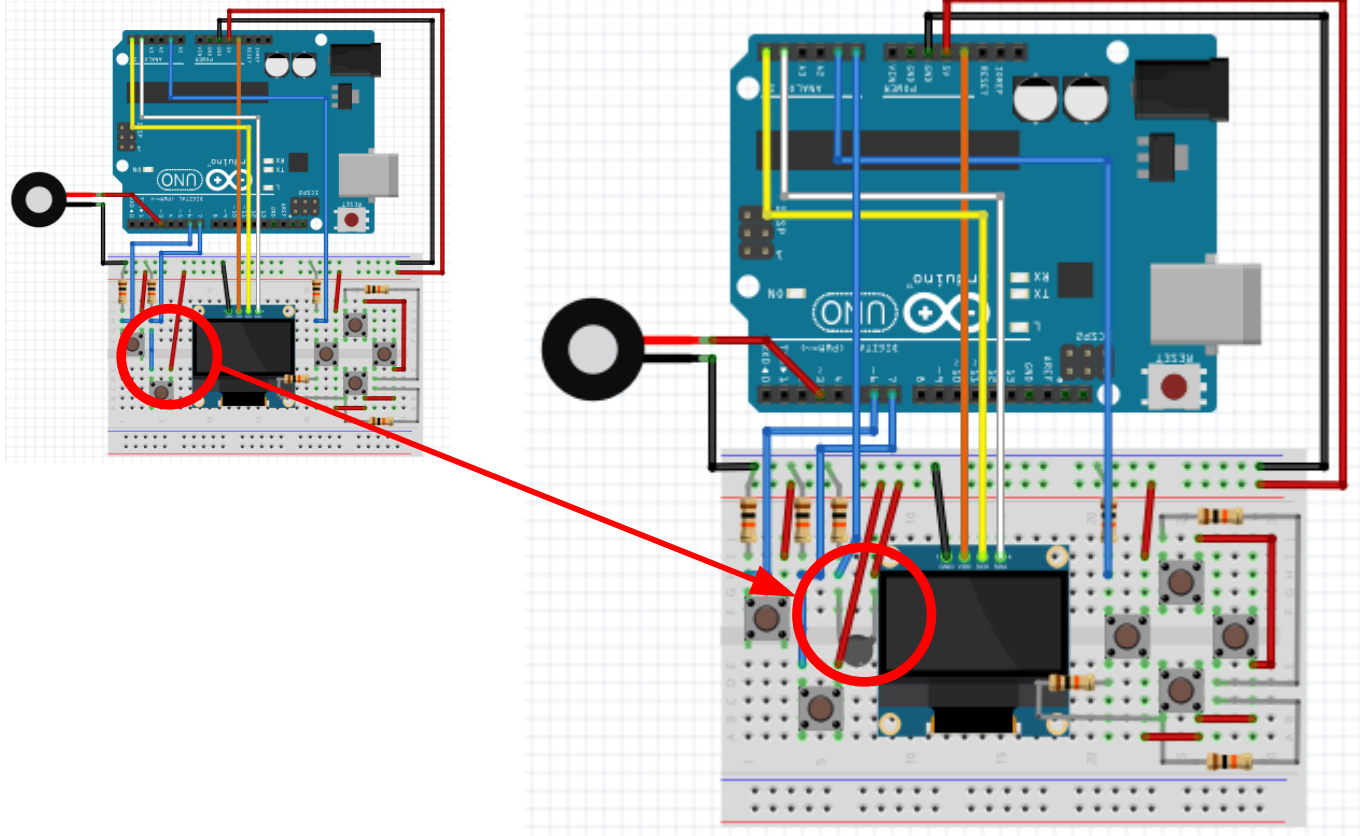
Program: pa0Z_arduino_oled_clock_therm_analog
   (1/2): Arduino OLED Clock with Thermometer and Analog Display
          :
          : by TeakSoon Ding for STEMKRAF (OCT-2021)

And as a bonus, Arduino OLED Clock Program is now being extended with an additional function screen, to show room temperature using a thermistor and also to show time in Analog Clock format (Graphics).

Here we have added a 10Kohm Thermistor into our circuit, now we can read the room temperature and put that information into our OLED Clock

However, there is a price to pay for using pixel Graphics and using the Thermistor, program size and Memory usage has increased significantly.

Now with pixel Graphics and Thermometer,

Sketch uses **10826 bytes (33%) of program storage** space. Maximum is 32256 bytes.
Global variables use **1443 bytes (70%) of dynamic memory,** leaving 605 bytes for local variables. Maximum is 2048 bytes.

vs Previously without pixel Graphics and Thermometer,

Sketch uses **7542 bytes (23%) of program storage** space. Maximum is 32256 bytes.
Global variables use **391 bytes (19%) of dynamic memory,** leaving 1657 bytes for local variables. Maximum is 2048 bytes.

Program: pa0Z_arduino_oled_clock_therm_analog
   (2/2): Arduino OLED Clock with Thermometer and Analog Display
         :
         : by TeakSoon Ding for STEMKRAF (OCT-2021)

- Upload this program with the Arduino IDE Software
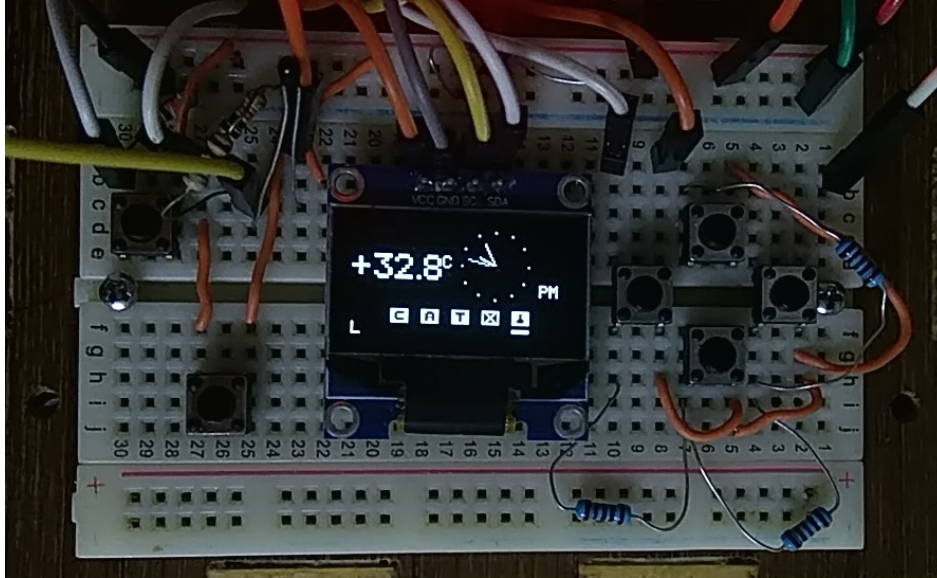
This program uses the "G" version of the LIBSK OLED Library.

The "G" Version allows us to display graphics, as you can see in this OLED screen, it s showing a circular clock face, with clock arms, rotating like our regular analog clock. To compile/upload the new OLED clock program, you need to install the following OLED library first.

Installation is very similar to the "C" Version difference is just one file. We use "libsk_g_oled.h" instead of "libsk_c_oled.h"

1. Find [Program Folder] from the Arduino IDE Software,
- "File|Preferences|Sketchbook Location: [Program Folder]"
2. Use the Computer File Browser,
- Look for existing folder "libraries" inside the [Program Folder]
3.Inside the "libraries" folder,
- Create a new folder named "libsk_oled"
4.Copy "libsk_g_oled.h" file into the newly created "libsk_oled" folder.

DONE!!!