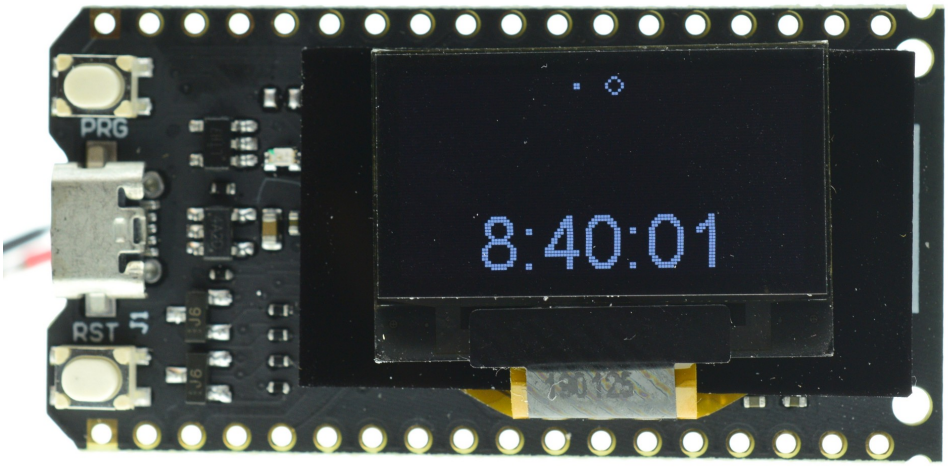
ESP32 TTGO V2.0 OLED Clock Demo



https://github.com/ThingPulse/esp8266-oled-ssd1306/tree/master/examples/SSD1306ClockDemo

Screen definition

```
SSD1306Wire  display(0x3c, 4, 15);
```

Screen reset

```
  pinMode(16, OUTPUT);
  digitalWrite(16, 1);
```

Demo

```
/**
* The MIT License (MIT)
*
* Copyright (c) 2018 by ThingPulse, Daniel Eichhorn
*
* Permission is hereby granted, free of charge, to any person obtaining a copy
* of this software and associated documentation files (the "Software"), to deal
* in the Software without restriction, including without limitation the rights
* to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
* copies of the Software, and to permit persons to whom the Software is
* furnished to do so, subject to the following conditions:
*
* The above copyright notice and this permission notice shall be included in all
* copies or substantial portions of the Software.
*
* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
* SOFTWARE.
*
* ThingPulse invests considerable time and money to develop these open source libraries.
* Please support us by buying our products (and not the clones) from
```

```
* https://thingpulse.com
*
*/

#include <TimeLib.h>
#include <Wire.h> // Only needed for Arduino 1.6.5 and earlier
#include "SSD1306Wire.h" // legacy include: `#include "SSD1306.h"`


// Include the UI lib
#include "OLEDDisplayUi.h"

// Include custom images

SSD1306Wire display(0x3c, 4, 15);

OLEDDisplayUi ui ( &display );

int screenW = 128;
int screenH = 64;
int clockCenterX = screenW/2;
int clockCenterY = ((screenH-16)/2)+16; // top yellow part is 16 px height
int clockRadius = 23;
const unsigned char activeSymbol[] PROGMEM = {
B00000000,
B00000000,
B00011000,
B00100100,
B01000010,
B01000010,
B00100100,
B00011000
};

const unsigned char inactiveSymbol[] PROGMEM = {
B00000000,
B00000000,
B00000000,
B00000000,
B00011000,
B00011000,
B00000000,
B00000000
};
// utility function for digital clock display: prints leading 0
String twoDigits(int digits){
if(digits < 10) {
String i = '0'+String(digits);
return i;
}
else {
return String(digits);
}
}

void clockOverlay(OLEDDisplay *display, OLEDDisplayUiState* state) {

}

void analogClockFrame(OLEDDisplay *display, OLEDDisplayUiState* state, int16_t x, int16_t y) {
// ui.disableIndicator();

// Draw the clock face
// display->drawCircle(clockCenterX + x, clockCenterY + y, clockRadius);
display->drawCircle(clockCenterX + x, clockCenterY + y, 2);
//
//hour ticks
for( int z=0; z < 360;z= z + 30 ){
//Begin at 0° and stop at 360°
float angle = z ;
angle = ( angle / 57.29577951 ) ; //Convert degrees to radians
```

```
int x2 = ( clockCenterX + ( sin(angle) * clockRadius ) );
int y2 = ( clockCenterY - ( cos(angle) * clockRadius ) );
int x3 = ( clockCenterX + ( sin(angle) * ( clockRadius - ( clockRadius / 8 ) ) ) );
int y3 = ( clockCenterY - ( cos(angle) * ( clockRadius - ( clockRadius / 8 ) ) ) );
display->drawLine( x2 + x , y2 + y , x3 + x , y3 + y);
}

// display second hand
float angle = second() * 6 ;
angle = ( angle / 57.29577951 ) ; //Convert degrees to radians
int x3 = ( clockCenterX + ( sin(angle) * ( clockRadius - ( clockRadius / 5 ) ) ) );
int y3 = ( clockCenterY - ( cos(angle) * ( clockRadius - ( clockRadius / 5 ) ) ) );
display->drawLine( clockCenterX + x , clockCenterY + y , x3 + x , y3 + y);
//
// display minute hand
angle = minute() * 6 ;
angle = ( angle / 57.29577951 ) ; //Convert degrees to radians
x3 = ( clockCenterX + ( sin(angle) * ( clockRadius - ( clockRadius / 4 ) ) ) );
y3 = ( clockCenterY - ( cos(angle) * ( clockRadius - ( clockRadius / 4 ) ) ) );
display->drawLine( clockCenterX + x , clockCenterY + y , x3 + x , y3 + y);
//
// display hour hand
angle = hour() * 30 + int( ( minute() / 12 ) * 6 ) ;
angle = ( angle / 57.29577951 ) ; //Convert degrees to radians
x3 = ( clockCenterX + ( sin(angle) * ( clockRadius - ( clockRadius / 2 ) ) ) );
y3 = ( clockCenterY - ( cos(angle) * ( clockRadius - ( clockRadius / 2 ) ) ) );
display->drawLine( clockCenterX + x , clockCenterY + y , x3 + x , y3 + y);
}

void digitalClockFrame(OLEDDisplay *display, OLEDDisplayUiState* state, int16_t x, int16_t y) {
String timenow = String(hour())+":"+twoDigits(minute())+":"+twoDigits(second());
display->setTextAlignment(TEXT_ALIGN_CENTER);
display->setFont(ArialMT_Plain_24);
display->drawString(clockCenterX + x , clockCenterY + y, timenow );
}

// This array keeps function pointers to all frames
// frames are the single views that slide in
FrameCallback frames[] = { analogClockFrame, digitalClockFrame };

// how many frames are there?
int frameCount = 2;

// Overlays are statically drawn on top of a frame eg. a clock
OverlayCallback overlays[] = { clockOverlay };
int overlaysCount = 1;

void setup() {
Serial.begin(9600);
Serial.println();
// "Manual" Screen reset
pinMode(16, OUTPUT);
digitalWrite(16, 1);

// The ESP is capable of rendering 60fps in 80Mhz mode
// but that won't give you much time for anything else
// run it in 160Mhz mode or just set it to 30 fps
ui.setTargetFPS(60);

// Customize the active and inactive symbol
ui.setActiveSymbol(activeSymbol);
ui.setInactiveSymbol(inactiveSymbol);

// You can change this to
// TOP, LEFT, BOTTOM, RIGHT
ui.setIndicatorPosition(TOP);

// Defines where the first frame is located in the bar.
ui.setIndicatorDirection(LEFT_RIGHT);

// You can change the transition that is used
```

```
// SLIDE_LEFT, SLIDE_RIGHT, SLIDE_UP, SLIDE_DOWN
ui.setFrameAnimation(SLIDE_LEFT);

// Add frames
ui.setFrames(frames, frameCount);

// Add overlays
ui.setOverlays(overlays, overlaysCount);

// Initialising the UI will init the display too.
ui.init();

display.flipScreenVertically();

unsigned long secsSinceStart = millis();
// Unix time starts on Jan 1 1970. In seconds, that's 2208988800:
const unsigned long seventyYears = 2208988800UL;
// subtract seventy years:
unsigned long epoch = secsSinceStart - seventyYears * SECS_PER_HOUR;
setTime(epoch);

}


void loop() {
int remainingTimeBudget = ui.update();

if (remainingTimeBudget > 0) {
// You can do some work here
// Don't do stuff if you are below your
// time budget.
delay(remainingTimeBudget);

}

}
```