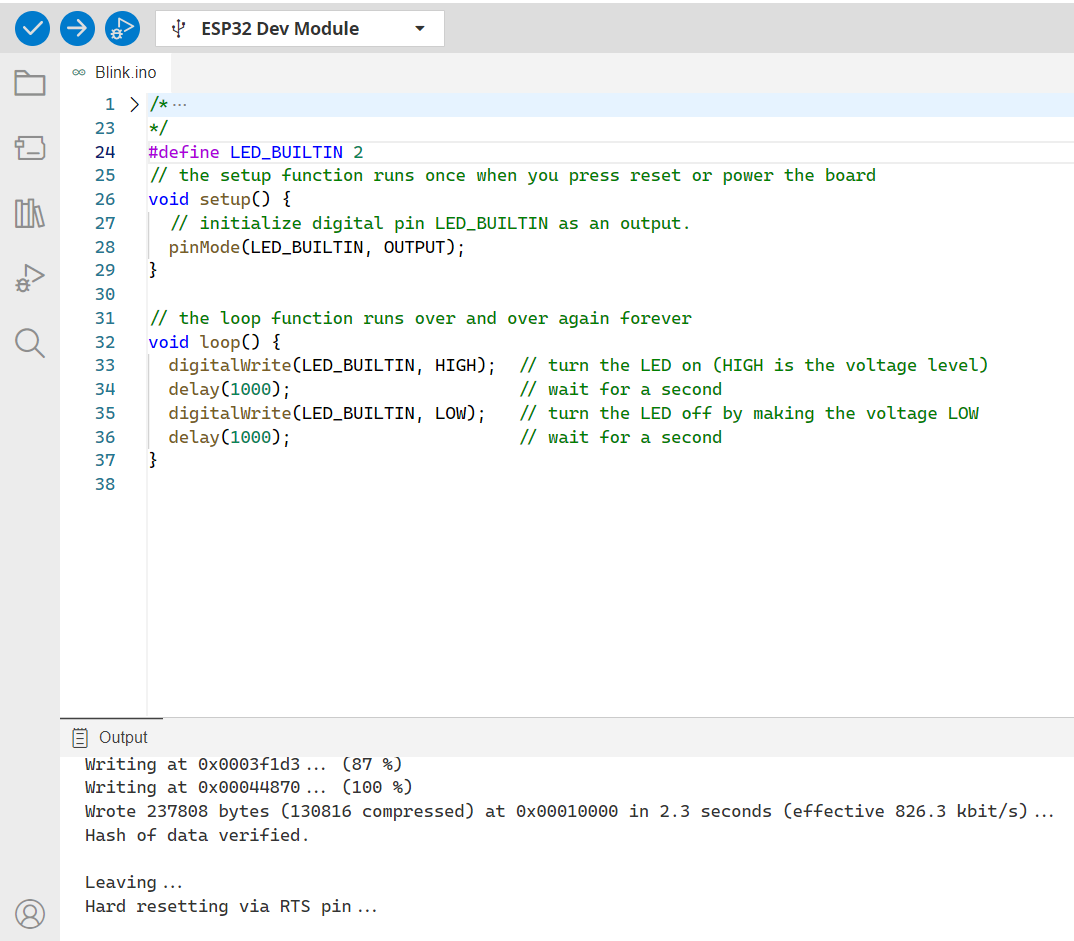
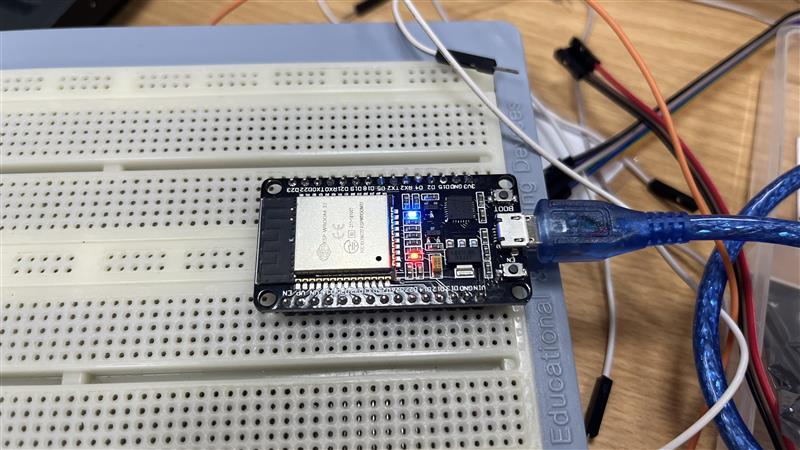
Lab 2 Report

Nguyễn Tiểu Phương 20210692

# 3.0. Set up tool suite and test ESP32 module



The sample code, uploaded to the ESP.



The blue led is blinking as expected.

# 3.1. The Tiny RTC module DS1307 + AT24C32 + DS18B20

### Can the three 3 ICs functions simultaneously? Explain.

Yes, the three ICs, DS1307, AT24C32, and DS18B20, can function simultaneously on the Tiny RTC module.

* DS1307 and AT24C32 both utilize the I2C communication protocol. This means they share the same two wires (SDA and SCL) for communication with the microcontroller.
* DS18B20 uses a 1-Wire protocol, which is independent of the I2C bus. It requires only one data line for communication.

The Tiny RTC module is specifically designed to support these three ICs. It has dedicated pins for connecting each component:

* SDA and SCL pins for I2C communication with DS1307 and AT24C32.
* A separate DS pin for the 1-Wire communication with the DS18B20 temperature sensor.

Therefore, the module can handle communication with all three ICs independently without interference, allowing them to function simultaneously.

### What are the required connections to work with the Tiny RTC module?

* SCL of RTC 🡪 SCL of ESP32 i.e. Pin D22.
* SDA of RTC 🡪 SDA of ESP32 i.e. Pin D21.
* GND of RTC 🡪 GND of ESP32.
* Vcc of RTC 🡪 Vcc of ESP32.
* DS pin of RTC 🡪 data pin of the DS18B20 sensor (D4 on ESP32)

**Power supply:**

* Common ground GND.
* VCC for Tiny RTC 🡪 Vin from ESP32 (5V power)
* Some capacitors for extra stabilization

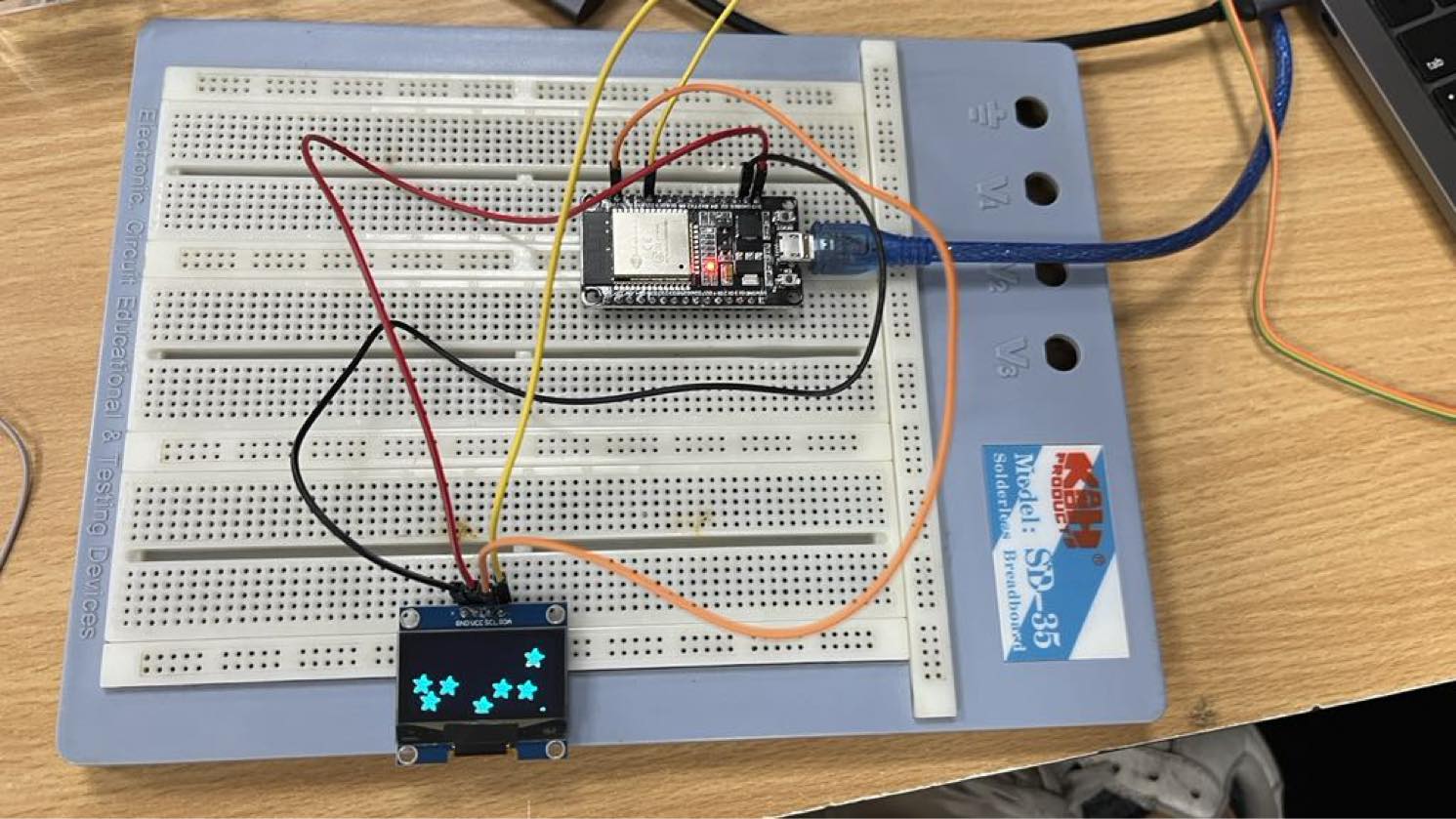
**Digital signal:**

* 3.3V for SH1106 Vcc from ESP32.
* VBAT for Tiny RTC 🡪 3V3 (replacing battery power).

### What are the addresses of DS1307 and AT24C32?

* AT24C32: A0 - A2, the device address inputs. These pins can be configured high or low to change the device’s address.
* DS1307: 0x68 (I2C connection)

# 3.2. The OLED SH1106 module



The OLED is up and running.

# 3.3. Data log system hardware

A circuit board with wires and a computer

Description automatically generated

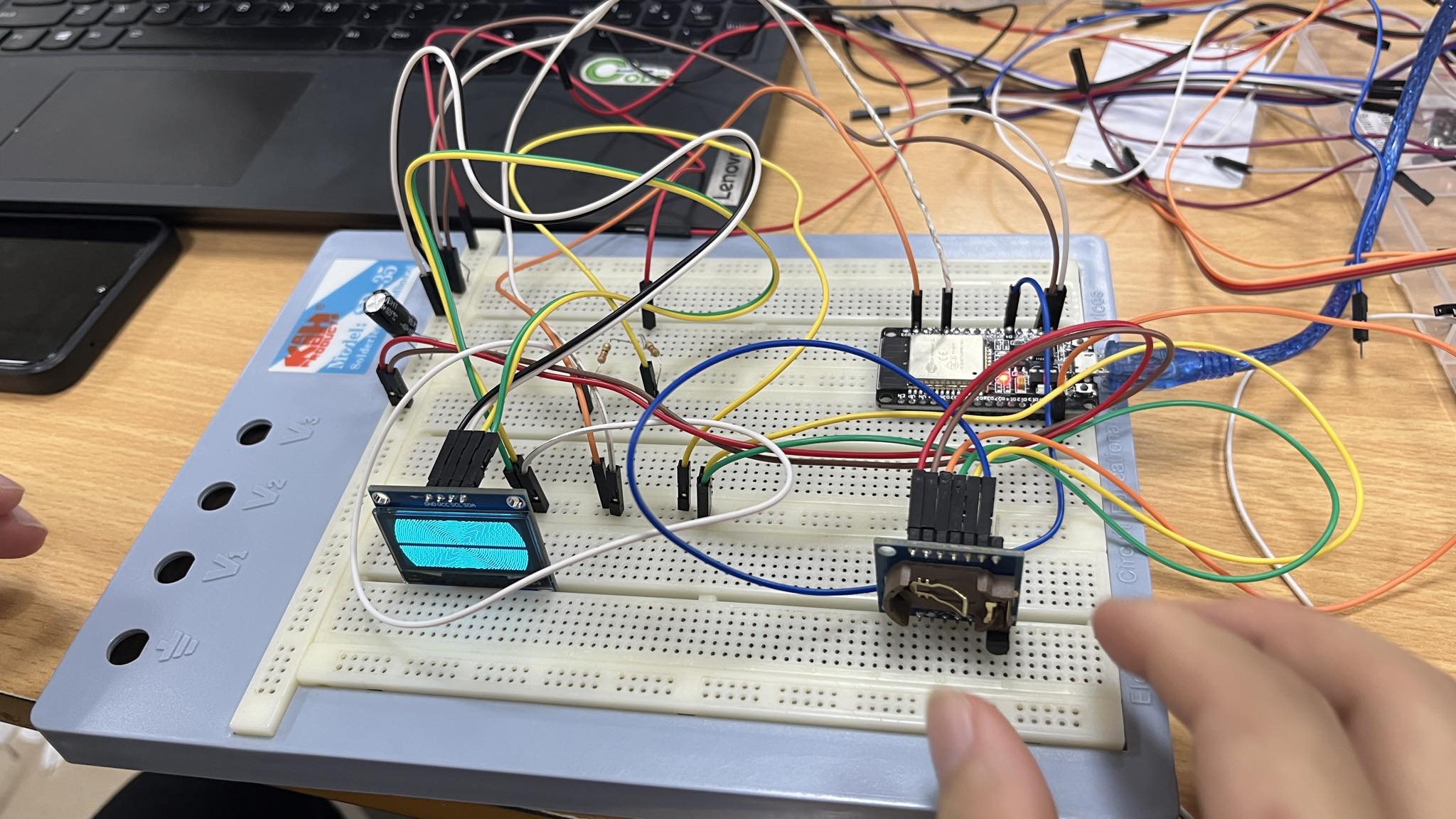
The fully assembled circuit according to the reference schema.

# 3.4. Working with SH1106

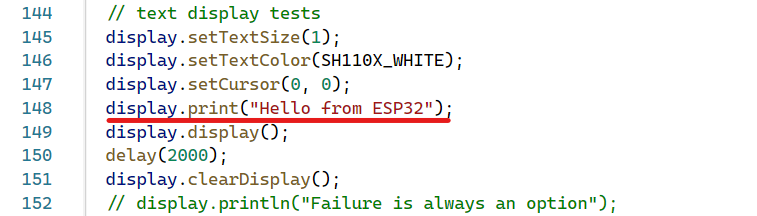
A screenshot of a computer

Description automatically generated

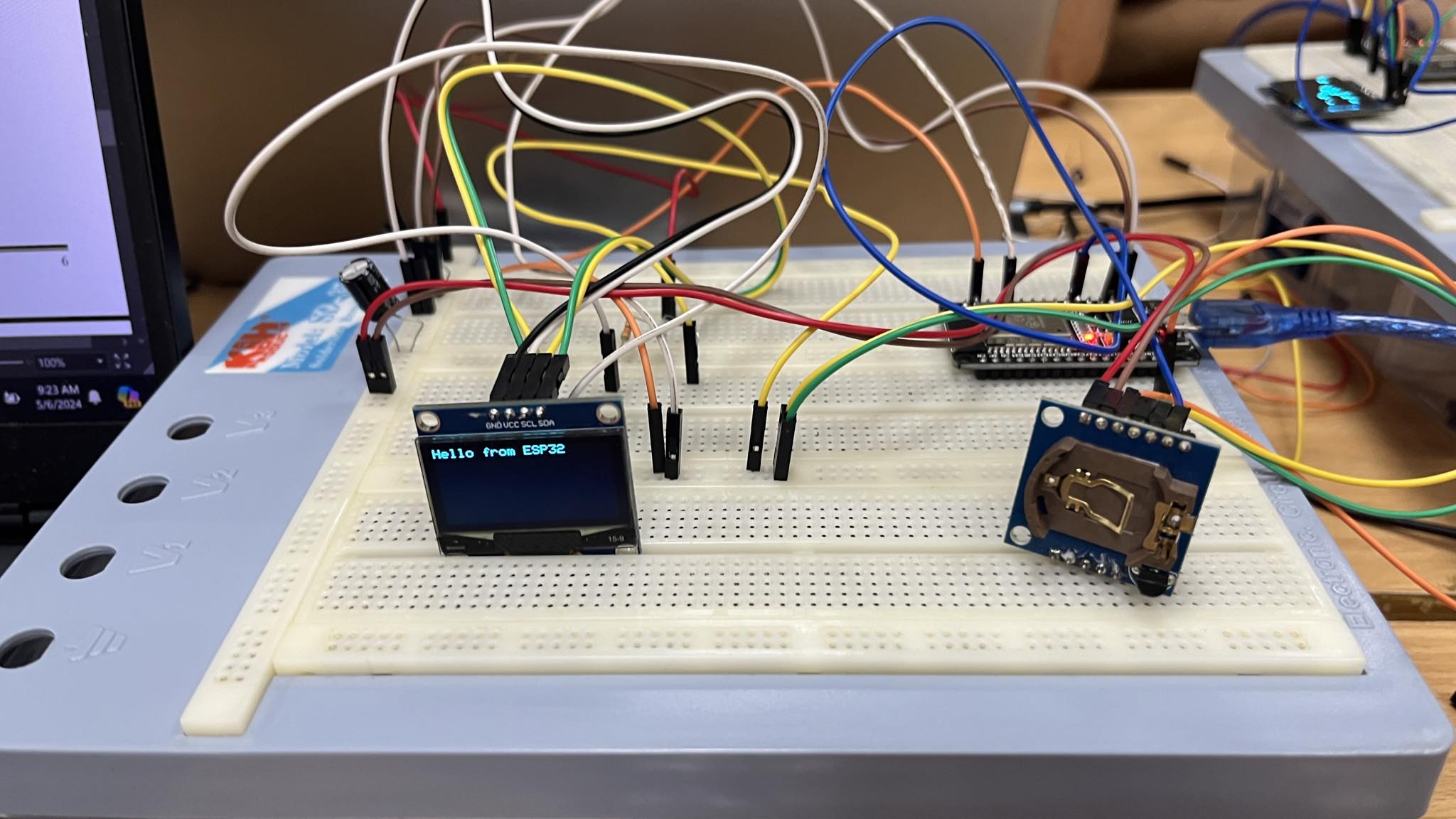
The example code downloaded and flashed. Module SH1106 is at the address 0x3C.



The default Adafruit welcome animation on the OLED displays as written in the code. **For the video, please go** [**here**](https://husteduvn-my.sharepoint.com/:f:/g/personal/phuong_nt210692_sis_hust_edu_vn/EuZD-Q4_L85DhCUkboyhmX8BXkW52AA-h_Bf1PvbX-uJVw?e=FzihbQ)**.**



Add the above line, and comment the rest of the code to keep the string on the screen.



The string “Hello from ESP32” is displayed on the OLED.

Full source code

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

  This is an example for our Monochrome OLEDs based on SH110X drivers

  This example is for a 128x64 size display using I2C to communicate

  3 pins are required to interface (2 I2C and one reset)

  Adafruit invests time and resources providing this open source code,

  please support Adafruit and open-source hardware by purchasing

  products from Adafruit!

  Written by Limor Fried/Ladyada  for Adafruit Industries.

  BSD license, check license.txt for more information

  All text above, and the splash screen must be included in any redistribution

  i2c SH1106 modified by Rupert Hirst  12/09/21

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <SPI.h>

#include <Wire.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SH110X.h>

/\* Uncomment the initialize the I2C address , uncomment only one, If you get a totally blank screen try the other\*/

#define i2c\_Address 0x3c //initialize with the I2C addr 0x3C Typically eBay OLED's

//#define i2c\_Address 0x3d //initialize with the I2C addr 0x3D Typically Adafruit OLED's

#define SCREEN\_WIDTH 128 // OLED display width, in pixels

#define SCREEN\_HEIGHT 64 // OLED display height, in pixels

#define OLED\_RESET -1   //   QT-PY / XIAO

Adafruit\_SH1106G display = Adafruit\_SH1106G(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, OLED\_RESET);

#define NUMFLAKES 10

#define XPOS 0

#define YPOS 1

#define DELTAY 2

#define LOGO16\_GLCD\_HEIGHT 16

#define LOGO16\_GLCD\_WIDTH  16

static const unsigned char PROGMEM logo16\_glcd\_bmp[] =

{ B00000000, B11000000,

  B00000001, B11000000,

  B00000001, B11000000,

  B00000011, B11100000,

  B11110011, B11100000,

  B11111110, B11111000,

  B01111110, B11111111,

  B00110011, B10011111,

  B00011111, B11111100,

  B00001101, B01110000,

  B00011011, B10100000,

  B00111111, B11100000,

  B00111111, B11110000,

  B01111100, B11110000,

  B01110000, B01110000,

  B00000000, B00110000

};

void setup()   {

  Serial.begin(9600);

  // Show image buffer on the display hardware.

  // Since the buffer is intialized with an Adafruit splashscreen

  // internally, this will display the splashscreen.

  delay(250); // wait for the OLED to power up

  display.begin(i2c\_Address, true); // Address 0x3C default

 //display.setContrast (0); // dim display

  display.display();

  delay(2000);

  // Clear the buffer.

  display.clearDisplay();

  // draw a single pixel

  display.drawPixel(10, 10, SH110X\_WHITE);

  // Show the display buffer on the hardware.

  // NOTE: You \_must\_ call display after making any drawing commands

  // to make them visible on the display hardware!

  display.display();

  delay(2000);

  display.clearDisplay();

  // draw many lines

  testdrawline();

  display.display();

  delay(2000);

  display.clearDisplay();

  // draw rectangles

  testdrawrect();

  display.display();

  delay(2000);

  display.clearDisplay();

  // draw multiple rectangles

  testfillrect();

  display.display();

  delay(2000);

  display.clearDisplay();

  // draw mulitple circles

  testdrawcircle();

  display.display();

  delay(2000);

  display.clearDisplay();

  // draw a SH110X\_WHITE circle, 10 pixel radius

  display.fillCircle(display.width() / 2, display.height() / 2, 10, SH110X\_WHITE);

  display.display();

  delay(2000);

  display.clearDisplay();

  testdrawroundrect();

  delay(2000);

  display.clearDisplay();

  testfillroundrect();

  delay(2000);

  display.clearDisplay();

  testdrawtriangle();

  delay(2000);

  display.clearDisplay();

  testfilltriangle();

  delay(2000);

  display.clearDisplay();

  // draw the first ~12 characters in the font

  testdrawchar();

  display.display();

  delay(2000);

  display.clearDisplay();

  // text display tests

  display.setTextSize(1);

  display.setTextColor(SH110X\_WHITE);

  display.setCursor(0, 0);

  display.print("Hello from ESP32");

  display.display();

  delay(2000);

  display.clearDisplay();

}

void loop() {

}

void testdrawbitmap(const uint8\_t \**bitmap*, uint8\_t *w*, uint8\_t *h*) {

  uint8\_t icons[NUMFLAKES][3];

  // initialize

  for (uint8\_t f = 0; f < NUMFLAKES; f++) {

    icons[f][XPOS] = random(display.width());

    icons[f][YPOS] = 0;

    icons[f][DELTAY] = random(5) + 1;

    Serial.print("x: ");

    Serial.print(icons[f][XPOS], DEC);

    Serial.print(" y: ");

    Serial.print(icons[f][YPOS], DEC);

    Serial.print(" dy: ");

    Serial.println(icons[f][DELTAY], DEC);

  }

  while (1) {

    // draw each icon

    for (uint8\_t f = 0; f < NUMFLAKES; f++) {

      display.drawBitmap(icons[f][XPOS], icons[f][YPOS], bitmap, w, h, SH110X\_WHITE);

    }

    display.display();

    delay(200);

    // then erase it + move it

    for (uint8\_t f = 0; f < NUMFLAKES; f++) {

      display.drawBitmap(icons[f][XPOS], icons[f][YPOS], bitmap, w, h, SH110X\_BLACK);

      // move it

      icons[f][YPOS] += icons[f][DELTAY];

      // if its gone, reinit

      if (icons[f][YPOS] > display.height()) {

        icons[f][XPOS] = random(display.width());

        icons[f][YPOS] = 0;

        icons[f][DELTAY] = random(5) + 1;

      }

    }

  }

}

void testdrawchar(void) {

  display.setTextSize(1);

  display.setTextColor(SH110X\_WHITE);

  display.setCursor(0, 0);

  for (uint8\_t i = 0; i < 168; i++) {

    if (i == '\n') continue;

    display.write(i);

    if ((i > 0) && (i % 21 == 0))

      display.println();

  }

  display.display();

  delay(1);

}

void testdrawcircle(void) {

  for (int16\_t i = 0; i < display.height(); i += 2) {

    display.drawCircle(display.width() / 2, display.height() / 2, i, SH110X\_WHITE);

    display.display();

    delay(1);

  }

}

void testfillrect(void) {

  uint8\_t color = 1;

  for (int16\_t i = 0; i < display.height() / 2; i += 3) {

    // alternate colors

    display.fillRect(i, i, display.width() - i \* 2, display.height() - i \* 2, color % 2);

    display.display();

    delay(1);

    color++;

  }

}

void testdrawtriangle(void) {

  for (int16\_t i = 0; i < min(display.width(), display.height()) / 2; i += 5) {

    display.drawTriangle(display.width() / 2, display.height() / 2 - i,

                         display.width() / 2 - i, display.height() / 2 + i,

                         display.width() / 2 + i, display.height() / 2 + i, SH110X\_WHITE);

    display.display();

    delay(1);

  }

}

void testfilltriangle(void) {

  uint8\_t color = SH110X\_WHITE;

  for (int16\_t i = min(display.width(), display.height()) / 2; i > 0; i -= 5) {

    display.fillTriangle(display.width() / 2, display.height() / 2 - i,

                         display.width() / 2 - i, display.height() / 2 + i,

                         display.width() / 2 + i, display.height() / 2 + i, SH110X\_WHITE);

    if (color == SH110X\_WHITE) color = SH110X\_BLACK;

    else color = SH110X\_WHITE;

    display.display();

    delay(1);

  }

}

void testdrawroundrect(void) {

  for (int16\_t i = 0; i < display.height() / 2 - 2; i += 2) {

    display.drawRoundRect(i, i, display.width() - 2 \* i, display.height() - 2 \* i, display.height() / 4, SH110X\_WHITE);

    display.display();

    delay(1);

  }

}

void testfillroundrect(void) {

  uint8\_t color = SH110X\_WHITE;

  for (int16\_t i = 0; i < display.height() / 2 - 2; i += 2) {

    display.fillRoundRect(i, i, display.width() - 2 \* i, display.height() - 2 \* i, display.height() / 4, color);

    if (color == SH110X\_WHITE) color = SH110X\_BLACK;

    else color = SH110X\_WHITE;

    display.display();

    delay(1);

  }

}

void testdrawrect(void) {

  for (int16\_t i = 0; i < display.height() / 2; i += 2) {

    display.drawRect(i, i, display.width() - 2 \* i, display.height() - 2 \* i, SH110X\_WHITE);

    display.display();

    delay(1);

  }

}

void testdrawline() {

  for (int16\_t i = 0; i < display.width(); i += 4) {

    display.drawLine(0, 0, i, display.height() - 1, SH110X\_WHITE);

    display.display();

    delay(1);

  }

  for (int16\_t i = 0; i < display.height(); i += 4) {

    display.drawLine(0, 0, display.width() - 1, i, SH110X\_WHITE);

    display.display();

    delay(1);

  }

  delay(250);

  display.clearDisplay();

  for (int16\_t i = 0; i < display.width(); i += 4) {

    display.drawLine(0, display.height() - 1, i, 0, SH110X\_WHITE);

    display.display();

    delay(1);

  }

  for (int16\_t i = display.height() - 1; i >= 0; i -= 4) {

    display.drawLine(0, display.height() - 1, display.width() - 1, i, SH110X\_WHITE);

    display.display();

    delay(1);

  }

  delay(250);

  display.clearDisplay();

  for (int16\_t i = display.width() - 1; i >= 0; i -= 4) {

    display.drawLine(display.width() - 1, display.height() - 1, i, 0, SH110X\_WHITE);

    display.display();

    delay(1);

  }

  for (int16\_t i = display.height() - 1; i >= 0; i -= 4) {

    display.drawLine(display.width() - 1, display.height() - 1, 0, i, SH110X\_WHITE);

    display.display();

    delay(1);

  }

  delay(250);

  display.clearDisplay();

  for (int16\_t i = 0; i < display.height(); i += 4) {

    display.drawLine(display.width() - 1, 0, 0, i, SH110X\_WHITE);

    display.display();

    delay(1);

  }

  for (int16\_t i = 0; i < display.width(); i += 4) {

    display.drawLine(display.width() - 1, 0, i, display.height() - 1, SH110X\_WHITE);

    display.display();

    delay(1);

  }

  delay(250);

}

# 3.5. Working with DS1307

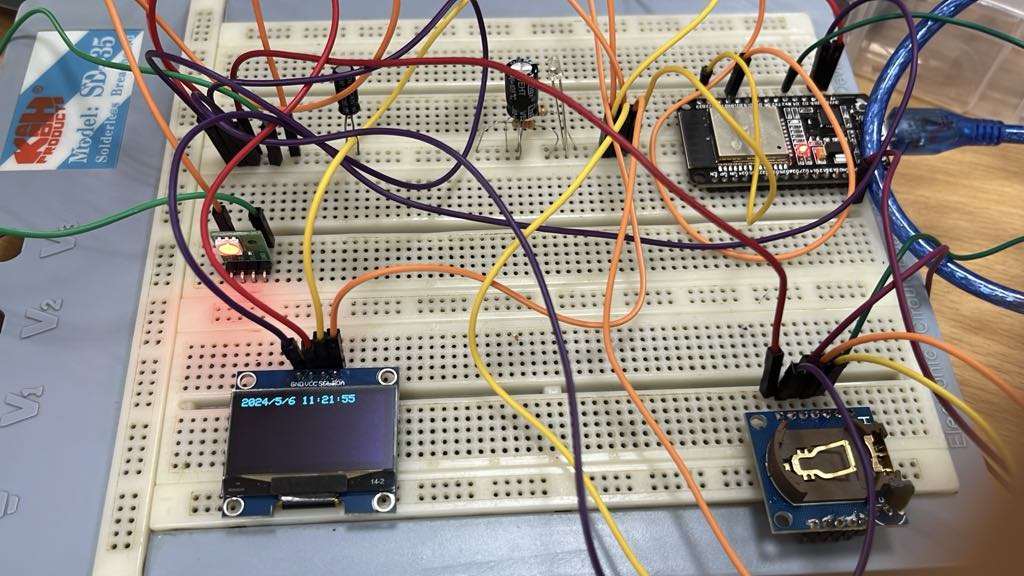
A screenshot of a computer

Description automatically generated

The result, as shown in the Serial Monitor. The program output the date and time, as well as the time elapsed since (the UNIX year!) January 1st 1970.

To print the current date time to the SH1106 LCD with the format “yyyy/mm/dd hh:mm:ss”, we just need to uncomment the line that adjust DateTime for the rtc:

rtc.adjust(DateTime(F(\_\_DATE\_\_), F(\_\_TIME\_\_)));



The current date time printed on the OLED.

Full source code

#include <Wire.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SH110X.h>

#include "RTClib.h"

#include <OneWire.h>

#include <DallasTemperature.h>

RTC\_DS1307 rtc;

#define i2c\_Address 0x3c //initialize with the I2C addr 0x3C Typically eBay OLED's

#define SCREEN\_WIDTH 128 // OLED display width, in pixels

#define SCREEN\_HEIGHT 64 // OLED display height, in pixels

#define OLED\_RESET -1   //   QT-PY / XIAO

Adafruit\_SH1106G display = Adafruit\_SH1106G(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, OLED\_RESET);

char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};

#define ONE\_WIRE\_BUS 4

*OneWire* oneWire(*ONE\_WIRE\_BUS*);

*DallasTemperature* sensors(&*oneWire*);

void setup() {

  Serial.begin(9600);

  Wire.begin();

  rtc.begin();

#ifndef ESP8266

  while (!Serial); // wait for serial port to connect. Needed for native USB

#endif

  if (!rtc.isrunning()) {

    Serial.println("RTC is NOT running, let's set the time!");

    rtc.adjust(DateTime(F(\_\_DATE\_\_), F(\_\_TIME\_\_)));

  }

  display.begin(i2c\_Address, true);

  display.display();

  delay(2000);

  sensors.begin();

}

void loop() {

  sensors.requestTemperatures();

  DateTime now = rtc.now();

  display.clearDisplay();

  display.setTextSize(1);

  display.setTextColor(SH110X\_WHITE);

  display.setCursor(0, 0);

  display.println("Current Temperature:");

  display.printf("\n");

  display.setTextSize(2);

  display.print(sensors.getTempCByIndex(0));

  display.printf(" %cC\n", 248);

  display.printf("\n");

  display.setTextSize(1);

  display.print(now.year(), DEC);

  display.print('/');

  display.print(now.month(), DEC);

  display.print('/');

  display.print(now.day(), DEC);

  display.print(" ");

  display.print(now.hour(), DEC);

  display.print(':');

  display.print(now.minute(), DEC);

  display.print(':');

  display.print(now.second(), DEC);

  display.display();

  delay(1000);

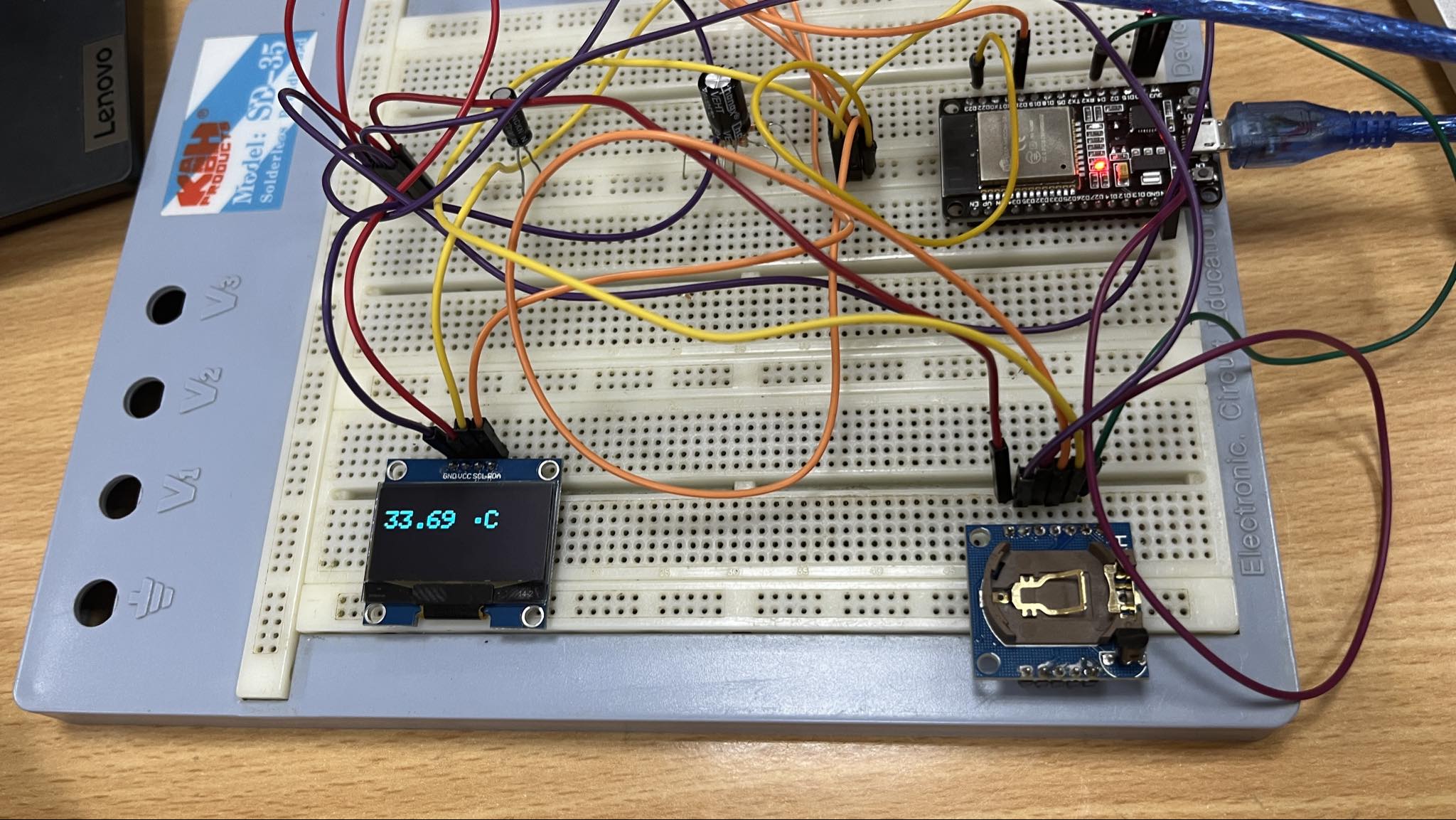
}

# 3.6. Working with DS18B20

A screenshot of a computer

Description automatically generated

The result when we run the sketch, as shown in Serial Monitor. Here, it output the temperature in real time with baud rate of 9600.



The temperature as shown in the OLED screen.

Full source code

#include <Wire.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SH110X.h>

#include "RTClib.h"

#include <OneWire.h>

#include <DallasTemperature.h>

RTC\_DS1307 rtc;

#define i2c\_Address 0x3c //initialize with the I2C addr 0x3C Typically eBay OLED's

#define SCREEN\_WIDTH 128 // OLED display width, in pixels

#define SCREEN\_HEIGHT 64 // OLED display height, in pixels

#define OLED\_RESET -1   //   QT-PY / XIAO

Adafruit\_SH1106G display = Adafruit\_SH1106G(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, OLED\_RESET);

char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};

#define ONE\_WIRE\_BUS 4

OneWire oneWire(ONE\_WIRE\_BUS);

DallasTemperature sensors(&*oneWire*);

void setup() {

  Serial.begin(9600);

  Wire.begin();

  rtc.begin();

#ifndef ESP8266

  while (!Serial); // wait for serial port to connect. Needed for native USB

#endif

  if (!rtc.isrunning()) {

    Serial.println("RTC is NOT running, let's set the time!");

    rtc.adjust(DateTime(F(\_\_DATE\_\_), F(\_\_TIME\_\_)));

  }

  display.begin(i2c\_Address, true);

  display.display();

  delay(2000);

  sensors.begin();

}

void loop() {

  sensors.requestTemperatures();

  DateTime now = rtc.now();

  display.clearDisplay();

  display.setTextSize(1);

  display.setTextColor(SH110X\_WHITE);

  display.setCursor(0, 0);

  display.setTextSize(2);

  display.print(sensors.getTempCByIndex(0));

  display.printf(" %cC\n", 248);

  display.printf("\n");

  display.display();

  delay(1000);

}

# 3.7. Lab 2 project

### Explanation

#### Explain the data structure and data storage mechanism (+ the log rotation).

For the data structure: Temperature is stored as float, while the time is saved in the built-in DateTime type. A better way to implement this is to store these 2 together, but in class time, I haven’t optimize this yet.

The ESP32's EEPROM library allows byte-wise storage. We need to convert the data structure into a byte array before storing it in EEPROM. For each piece of new data recorded, we need to propagate the current address for writing.

Since EEPROM has limited space, a log rotation mechanism ensures we don't overwrite existing data. Here's how I implement it:

* Maintain a variable (e.g., currentAddress) to track the index of the next record to be written.
* Check if currentAddress reaches the maximum number of records that can fit in the allocated EEPROM space (calculate based on data record size and total space).
* If the limit is reached, reset currentAddress to 0 (overwrite the oldest record).
* Write the new data record at the address corresponding to the currentAddress.
* Update currentAddress for the next record.

This approach ensures continuous logging by overwriting the oldest data when the limit is reached.

#### Explain the sequence diagram of the firmware on ESP32. How does it receive and handle the commands from PC?

The sequence diagram can be visualized with the following steps:

**1. Startup:**

* ESP32 initializes sensors (DS18B20, DS1307), display (SH1106), and serial communication.
* ESP32 enters a loop waiting for commands from the PC.

**2. Receiving and Handling Commands:**

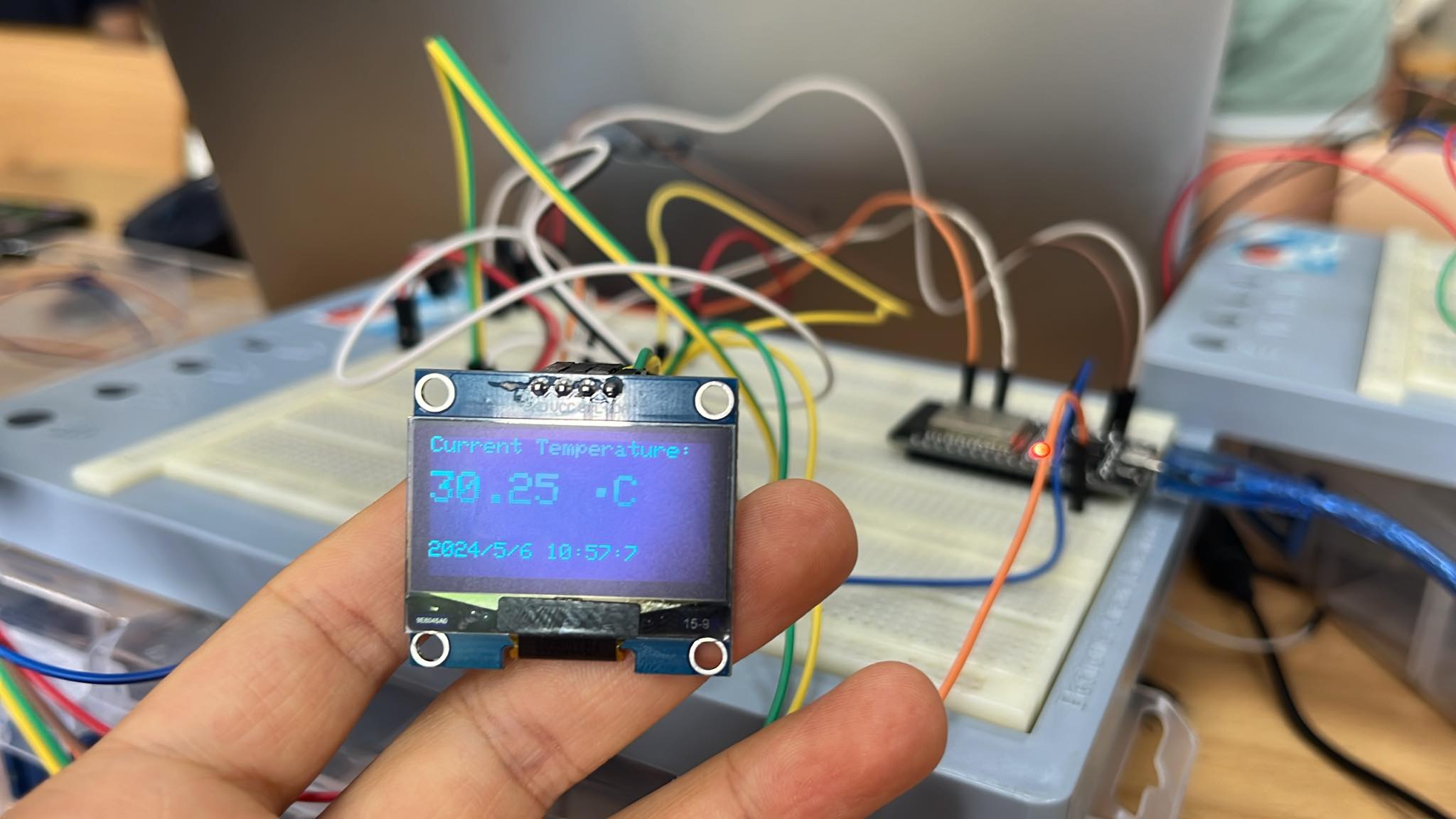
* ESP32 continuously checks for incoming data on the serial port.
* When data arrives, it reads the entire command string.
* Based on the received command:
  + **"START"**:
    - Read temperature from DS18B20.
    - Read timestamp from DS1307.
    - Format the data for display ("Current Temperature:", "tt °C", "YYYY/MM/DD hh:mm:ss").
    - Display the formatted data on the SH1106.
    - Pack the temperature and timestamp into a data structure.
    - Apply log rotation logic (explained in section 1).
    - Store the data in EEPROM using the calculated address.
    - Send the formatted data (Temp, Date, Time) to the PC via serial communication.
    - Wait for 5 seconds and repeat the process (steps a-f).
  + **"STOP"**:
    - Exit the loop that continuously reads temperature and performs actions (by setting isLogging to False).
    - ESP32 waits for further commands.
  + **"GETMIN"**:
    - Search through all stored records in EEPROM (iterate through addresses based on data structure size and allocated space).
    - Find the record with the lowest temperature.
    - Format and send the data record (Temp, Date, Time) with the minimum temperature to the PC via serial communication.
  + **"GETMAX"**: Similar to "GETMIN" but search for the record with the highest temperature.
* If an unrecognized command is received, send an error message back to the PC.

**3. Loop:**

* ESP32 continues to wait for new commands from the PC and repeats steps 2 and 3.

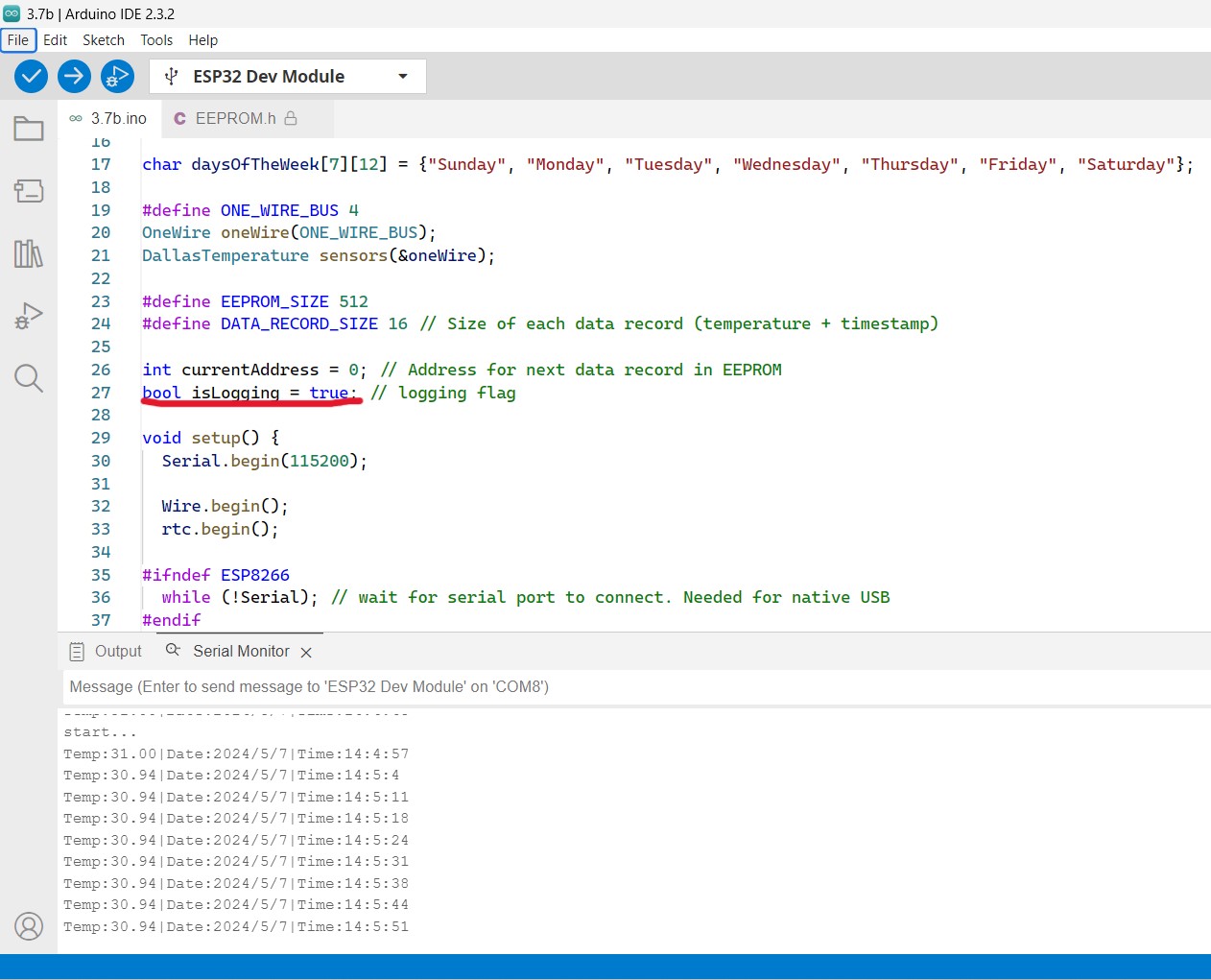
### Result

**For the video demonstration, please go** [**here**](https://husteduvn-my.sharepoint.com/:f:/g/personal/phuong_nt210692_sis_hust_edu_vn/EuZD-Q4_L85DhCUkboyhmX8BXkW52AA-h_Bf1PvbX-uJVw?e=FzihbQ)**.**



The data displayed similar as in the guide, with the correct date time and temperature.

### START



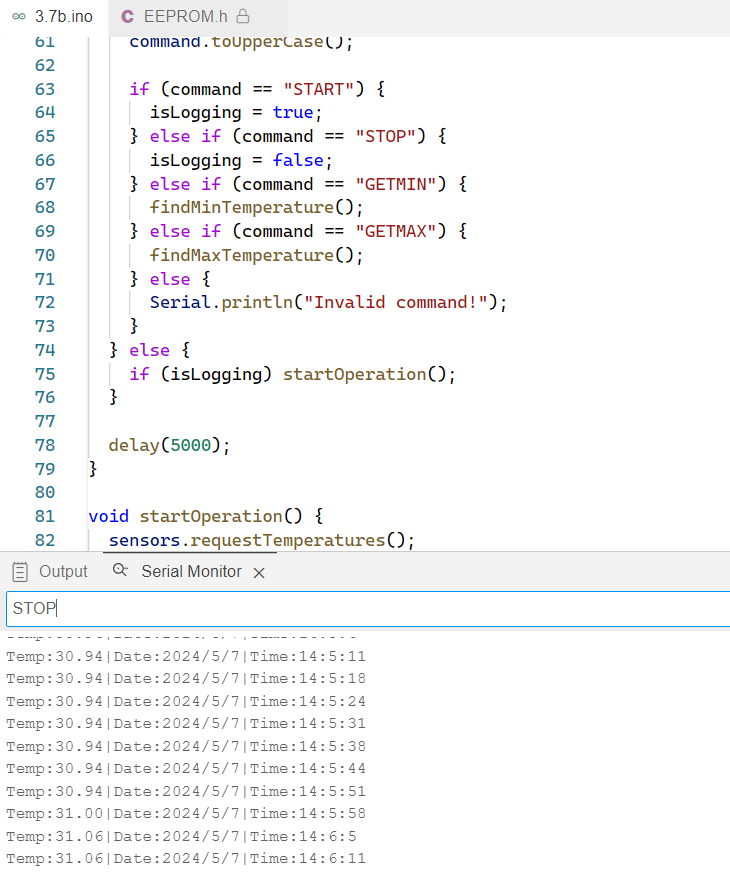
Upon successful flashing, the program starts running. It logs and updates the data by default, since I set isLogging to True.

A computer with wires connected to a circuit board

Description automatically generated

The OLED display also updates the data with the same frequency as in the Serial Monitor.

### STOP



We try stopping the program.

A screenshot of a computer

Description automatically generated

From 14:06 PM, we sent the command STOP, the program stopped. We turned it back on by sending START again, and it started updating at that time, which is 14:15 PM.

### GETMIN - GETMAX

As the name implies, these functions return the minimum and maximum temperature recorded. Note that the data storage of the ESP is quite limited, hence, the min and max values returned are in respect to the records which are still presented in the memory at the time of inquiry.

(For demonstration, please see the video mentioned above.)

Full source code

#include <Wire.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SH110X.h>

#include "RTClib.h"

#include <OneWire.h>

#include <DallasTemperature.h>

#include <EEPROM.h>

RTC\_DS1307 rtc;

#define i2c\_Address 0x3c

#define SCREEN\_WIDTH 128

#define SCREEN\_HEIGHT 64

#define OLED\_RESET -1

Adafruit\_SH1106G display = Adafruit\_SH1106G(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, OLED\_RESET);

char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};

#define ONE\_WIRE\_BUS 4

*OneWire* oneWire(*ONE\_WIRE\_BUS*);

*DallasTemperature* sensors(&*oneWire*);

#define EEPROM\_SIZE 512

#define DATA\_RECORD\_SIZE 16 // Size of each data record (temperature + timestamp)

int currentAddress = 0; // Address for next data record in EEPROM

bool isLogging = true; // logging flag

void setup() {

  Serial.begin(115200);

  Wire.begin();

  rtc.begin();

#ifndef ESP8266

  while (!Serial); // wait for serial port to connect. Needed for native USB

#endif

  if (!rtc.isrunning()) {

    Serial.println("RTC is NOT running, let's set the time!");

    rtc.adjust(DateTime(F(\_\_DATE\_\_), F(\_\_TIME\_\_)));

  }

  display.begin(i2c\_Address, true);

  display.display();

  delay(2000);

  sensors.begin();

  Serial.println("start...");

  if (!EEPROM.begin(EEPROM\_SIZE))

  {

    Serial.println("failed to initialise EEPROM"); delay(1000000);

  }

}

void loop() {

  if (Serial.available()) {

    String command = Serial.readStringUntil('\n');

    command.toUpperCase();

    if (command == "START") {

      isLogging = true;

    } else if (command == "STOP") {

      isLogging = false;

    } else if (command == "GETMIN") {

      findMinTemperature();

    } else if (command == "GETMAX") {

      findMaxTemperature();

    } else {

      Serial.println("Invalid command!");

    }

  } else {

    if (isLogging) startOperation();

  }

  delay(5000);

}

void startOperation() {

  sensors.requestTemperatures();

  DateTime now = rtc.now();

  saveRecord(sensors.getTempCByIndex(0), now);

  display.clearDisplay();

  sendDataToPC(sensors.getTempCByIndex(0), now);

  display.setTextSize(1);

  display.setTextColor(SH110X\_WHITE);

  display.setCursor(0, 0);

  display.println("Current Temperature:");

  display.printf("\n");

  display.setTextSize(2);

  display.print(sensors.getTempCByIndex(0));

  display.printf(" %cC\n", 248);

  display.printf("\n");

  display.setTextSize(1);

  display.print(now.year(), DEC);

  display.print('/');

  display.print(now.month(), DEC);

  display.print('/');

  display.print(now.day(), DEC);

  display.print(" ");

  display.print(now.hour(), DEC);

  display.print(':');

  display.print(now.minute(), DEC);

  display.print(':');

  display.print(now.second(), DEC);

  display.display();

  delay(1000);

}

void saveRecord(float *temperature*, *DateTime* *now*) {

  if (currentAddress + DATA\_RECORD\_SIZE >= EEPROM\_SIZE) {

    currentAddress = 0; // Reset address to the beginning

  }

  EEPROM.put(currentAddress, temperature);

  currentAddress += sizeof(float);

  EEPROM.put(currentAddress, now.year());

  currentAddress += sizeof(int);

  EEPROM.put(currentAddress, now.month());

  currentAddress += sizeof(int);

  EEPROM.put(currentAddress, now.day());

  currentAddress += sizeof(int);

  EEPROM.put(currentAddress, now.hour());

  currentAddress += sizeof(int);

  EEPROM.put(currentAddress, now.minute());

  currentAddress += sizeof(int);

  EEPROM.put(currentAddress, now.second());

  currentAddress += sizeof(int);

  EEPROM.commit();

}

void sendDataToPC(float *temperature*, *DateTime* *now*) {

  Serial.print("Temp:");

  Serial.print(temperature);

  Serial.print("|");

  Serial.print("Date:");

  Serial.print(now.year(), DEC);

  Serial.print("/");

  Serial.print(now.month(), DEC);

  Serial.print("/");

  Serial.print(now.day(), DEC);

  Serial.print("|");

  Serial.print("Time:");

  Serial.print(now.hour(), DEC);

  Serial.print(":");

  Serial.print(now.minute(), DEC);

  Serial.print(":");

  Serial.println(now.second(), DEC);

}

void findMinTemperature() {

  float minTemp = 0;

  int minAddress = 0;

  for (int i = 0; i < EEPROM\_SIZE; i += DATA\_RECORD\_SIZE) {

    float temp = EEPROM.readFloat(i);

    if (temp < minTemp || minTemp == 0) {

      minTemp = temp;

      minAddress = i;

    }

  }

  if (minTemp != 0) {

    DateTime timestamp;

    timestamp.year();

    timestamp.month();

    timestamp.day();

    timestamp.hour();

    timestamp.minute();

    timestamp.second();

    sendDataToPC(minTemp, timestamp);

  } else {

    Serial.println("No data records found!");

  }

}

void findMaxTemperature() {

   float maxTemp = -INFINITY; // Initialize maxTemp to negative infinity

  int maxAddress = 0;

  for (int i = 0; i < EEPROM\_SIZE; i += DATA\_RECORD\_SIZE) {

    float temp = EEPROM.readFloat(i);

    if (temp > maxTemp) {

      maxTemp = temp;

      maxAddress = i;

    }

  }

  if (maxTemp != -INFINITY) {

    DateTime timestamp;

    timestamp.year();

    timestamp.month();

    timestamp.day();

    timestamp.hour();

    timestamp.minute();

    timestamp.second();

    sendDataToPC(maxTemp, timestamp);

  } else {

    Serial.println("No data records found!");

  }

}

THE END