ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG ĐIỆN-ĐIỆN TỬ

--○🕮○--

BÁO CÁO

MÔN: ĐỒ ÁN THIẾT KẾ I

Đề tài : Thiết kế hệ thống cảnh báo xâm nhập mặn

Thành viên nhóm:

Nguyễn Trung Mạnh	20224054
Nguyễn Công Tú	20224183
Tô Mạnh Quang	20224116
Nguyễn Văn Quang	20223732

GVHD: **Thầy Nguyễn Hữu Phát**

Trợ giảng: Trần Văn Tình

Hà Nội - 2025

Abstract — Một trong những vấn đề đáng quan tâm hiện nay ở các vùng ven biển và hạ lưu sông là hiện tượng xâm nhập mặn, đặc biệt vào mùa khô hoặc thời điểm nước biển dâng cao. Việc giám sát và cảnh báo độ mặn kịp thời giúp người dân và nhà quản lý đưa ra các biện pháp ứng phó phù hợp để bảo vệ mùa màng, nguồn nước và đời sống sinh hoạt. Từ vấn đề trên, trong bài báo cáo này em sẽ trình bày thiết kế mô hình cảnh báo xâm nhập mặn dựa trên cảm biến đo độ mặn kết hợp với hệ thống giám sát dữ liệu theo thời gian thực.

Keywords — Hệ thống cảnh báo xâm nhập mặn, Salinity monitoring, Internet of Things, real-time alert

Đặt vấn đề

Hiện nay với sự phát triển mạnh mẽ của công nghệ số và cuộc cách mạng công nghiệp 4.0, thuật ngữ **IoT (Internet of Things - Internet vạn vật)** ngày càng trở nên phổ biến trong việc tự động hóa và quản lý từ xa các thiết bị giám sát môi trường. Trong thực tế, tình trạng xâm nhập mặn ở các khu vực như Đồng bằng sông Cửu Long đang trở nên ngày càng nghiêm trọng do biến đổi khí hậu, giảm lưu lượng nước ngọt từ thượng nguồn, và nước biển dâng. Độ mặn cao ảnh hưởng lớn đến sản xuất nông nghiệp, nuôi trồng thủy sản, cũng như chất lượng nước sinh hoạt của người dân.

Tuy nhiên, việc đo độ mặn ở nhiều khu vực hiện nay vẫn còn thực hiện thủ công hoặc chưa được giám sát liên tục, dẫn đến thiếu thông tin kịp thời để đưa ra phương án ứng phó. Vì vậy, cần có một hệ thống cảnh báo sớm xâm nhập mặn có khả năng **giám sát từ xa**, **tự động**, và **cập nhật dữ liệu theo thời gian thực**. Hệ thống này sẽ giúp các đơn vị quản lý cũng như người dân nắm bắt kịp thời diễn biến độ mặn, từ đó chủ động trong việc ứng phó và bảo vệ tài nguyên nước.

Mục tiêu nghiên cứu

Đồ án tập chung nghiên cứu, tìm hiểu các vấn đề cơ bản về cảm biến độ dẫn điện EC sau đó chuyển đổi chuyển đổi thông số độ dẫn điện sang độ mặn sau đó thu thập dữ liệu từ cảm biến EC và truyền về hệ thống xử lý. Thiết kế hệ thống IoT để gửi dữ liệu lên máy chủ có thể truy cập từ xa qua web/app di động. Tích hợp cơ chế cảnh báo thông minh khi độ mặn vượt quá ngưỡng an toàn.

Phát triển các phiên bản cảm biến độ dẫn diện rồi so sánh độ chính xác với các cảm biến thương mại

Bảng phân công nhiệm vụ

Tên – Mssv	Nhiệm vụ
Nguyễn Trung Mạnh - 20224054	Thiết kế cảm biến EC
Nguyễn Công Tú - 20224183	Thiết kế mạch in cho cảm biến EC
Tô Mạnh Quang - 20224116	irebase &lưu trữ dữ liệu SDCard , thời gian thực
Nguyễn Văn Quang - 20223732	Thiết kế giao diện Web

Mục Lục

I. Thiết kế cảm biến EC và chuyển đổi EC sang độ mặn

1. Cơ sở lý thuyết

a) Định nghĩa độ dẫn điện EC (Electrical Conductivity)

Độ dẫn điện (EC) là đại lượng đo khả năng dẫn dòng điện của một dung dịch. Nó phản ánh mức độ các ion tích điện (như Na^+, Cl^-, K^+, NO_3^-…) có trong dung dịch – càng nhiều ion, độ dẫn điện càng cao.

b) Đơn vị của độ dẫn điện

Siemens trên mét (S/m) – trong hệ SI

Thực tế thường dùng: mS/cm (milliSiemens/cm), µS/cm (microSiemens/cm)

c) Công thức tính độ dẫn điện EC

Độ dẫn điện EC được tính bằng công thức: $EC=\frac{K}{R}$

K: là hằng số cell (cell constant), phụ thuộc cấu trúc đầu đo (đơn vị: cm^{-1})

R: là điện trở giữa hai điện cực (Ω)

2. Xây dựng nguyên lý hoạt động của cảm biến độ dẫn điện EC

Cảm biến độ dẫn điện hoạt động dựa trên nguyên lý đo điện trở (hoặc độ dẫn) của dung dịch giữa hai hoặc nhiều điện cực được nhúng trong chất lỏng

Đầu cảm biến bao gồm 2 điện cực được làm bằng chất liệu có thể chống oxy hóa cũng nhưng chống ăn mòn trong thời gian dài.

Để đo được độ dẫn điện của dung dịch thì chúng ta phải cho dòng điện xoay chiều AC đi qua 2 điện cực khi cắm vào dung dịch. Bản thân dung dịch có trở kháng là Z_{EC} nên khi cho tín hiệu xoay chiều đi qua sẽ làm suy hao biên độ của điện áp AC từ đó dựa vào độ suy hoa của biên độ điện áp ta có thể tính được trở kháng của dung dịch rồi áp dụng công thức $EC=\frac{K}{R}$ để suy ra độ dẫn điện EC ở đây Z_{EC} có thể được coi là R ở công thức tính EC.

Chúng ta có rất nhiều loại tín hiệu xoay chiều nhưng để lý tưởng nhất thì ta có 2 loại tín hiệu xoay chiều là xung vuông và xung hình sine cho việc thiết kế cảm biến đo độ dẫn điện EC.

Câu hỏi đặt ra là tại sao không dùng tín hiệu 1 chiều DC thay cho tín hiệu xoay chiều AC để dễ đo đạc? Là vì các nguyên nhân chính thứ nhất là tín hiệu DC gây phân cực điện cực dẫn đến kết quả đo không chính xác và bị thay đổi theo thời gian. Nguyên nhân thứ 2 là gây phải ứng điện phân làm hỏng điện cực và tạo bọt khí gây nên sai số của kết quả đo.

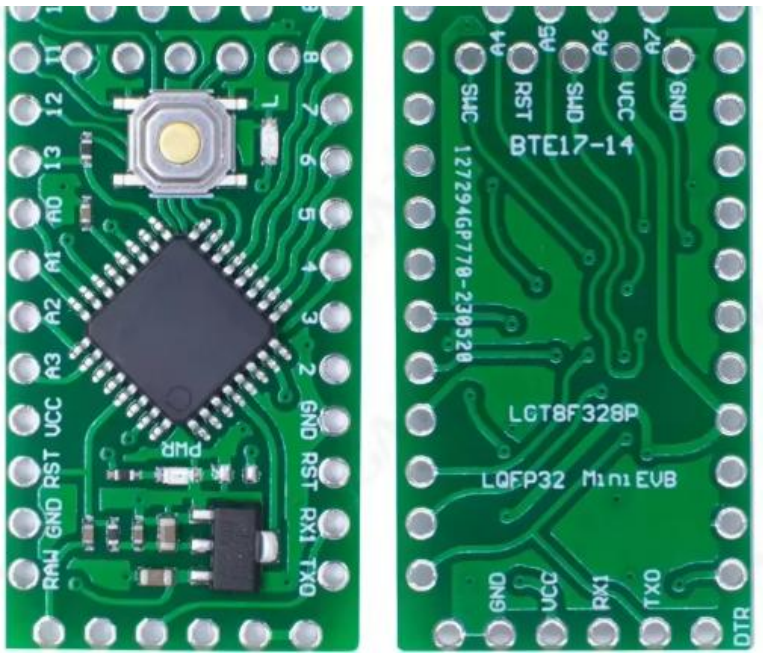
Dựa vào các yếu tố trên thì chúng ta nên chọn loại tín hiệu đưa qua dung dịch có dạng hình sine vì đối với xung vuông thì trong 1 chu kỳ vẫn có 1 khoảng thời gian ngắn là tín hiệu DC dẫn đến kết quả đo không còn được chính xác.

3. Xây dựng sơ đồ khối của cảm biến EC

a) Giới thiệu linh kiện

➢ Vi điều khiển

- Dựa vào yêu cầu tạo ra sóng sine có biên độ vài volt và tần số cao để cho qua dung dịch thì ta phải chọn vi điều khiển có đủ chắc năng chuyển đổi từ số sang tương tự(DAC- Digital-to-Analog Converter) và đọc tín hiệu tương tự sau khi tín hiệu qua dung dịch chuyển đổi sang tín hiệu số(ADC- Analog-to-Digital Converter).
- Tiêu chí lựa chọn module vi điều khiển là có kích thước nhỏ và giá thành rẻ, có đầy đủ các giao thức cần thiết(I2C, UART, SPI).
- Từ các yêu cầu trên ta lựa chọn module vi điều khiển LGT8F328P có đầy đủ các chức năng cần thiết



-Ưu điểm: dễ lập trình, có các thư viện hỗ trợ

-Nhược điểm: chỉ phù hợp với dự án nhỏ

-Các thông số cơ bản liên quan đến dự án

CPU core AVR core (tương thích ATmega328P)

Tốc độ xung nhịp (Clock) Lên đến 32 MHz (so với 16 MHz của ATmega328P)

Điện áp hoạt động 1.8V – 5.5V (hoạt động tốt cả ở 3.3V và 5V)

Flash 32 KB

SRAM 2 KB

EEPROM 1 KB

GPIO (Digital I/O) 23 chân

Chân PWM 6 kênh

Chân analog (ADC) 8 kênh 10-bit (A0–A7)

DAC Có 1 kênh DAC (8-bit), tích hợp phần cứng (ưu điểm so với ATmega328P)

ADC 10-bit, tốc độ cao (hơn ATmega328P), có thể chọn nguồn tham chiếu nội/ngoại

Giao tiếp UART, I2C, SPI (tương thích với Arduino)

- Cảm biến nhiệt độ

Do điện trở của dung dịch có thể thay đổi theo nhiệt độ dẫn đến việc đo đạc các thông số không ổn định nên ta cần phải đưa chỉ số EC về điều kiện nhiệt độ 25 độ C. Từ yêu cầu trên ta phải lựa chọn cảm biến nhiệt độ có khả năng chống nước và độ chính xác cao nên để phù hợp thì ta chọn cảm biến DS18B20 có giá thành rẻ và dễ sử dụng qua giao tiếp 1-wire.

- Màn hình hiển thị

-Tiêu chí lựa chọn màn hình có kích thước nhỏ gọn, dễ giao tiếp có thể hiển thị đầy đủ thông số độ mặn, độ dẫn điện và nhiệt độ rõ ràng. Nên ta chọn màn hình OLED 0.96 inch I2C Display (128x64, SSD1306)

-Thông số kỹ thuật

Loại màn hình OLED đơn sắc (monochrome)

Kích thước 0.96 inch (đường chéo)

Độ phân giải 128 x 64 pixel

Driver IC SSD1306

Giao tiếp I2C (có 4 chân: GND, VCC, SCL, SDA)

Màu hiển thị Thường là trắng, xanh dương, hoặc vàng-xanh (2 màu)

Điện áp hoạt động 3.3-5V

- Các linh kiện thụ động và chủ động khác

Tên linh kiện:

-Điện trở: 10k, 2.2M, 820, 240 ohm.

-Tụ điện: tụ gốm, tụ hóa.

-IC khuếch đại thuật toán: LM358.

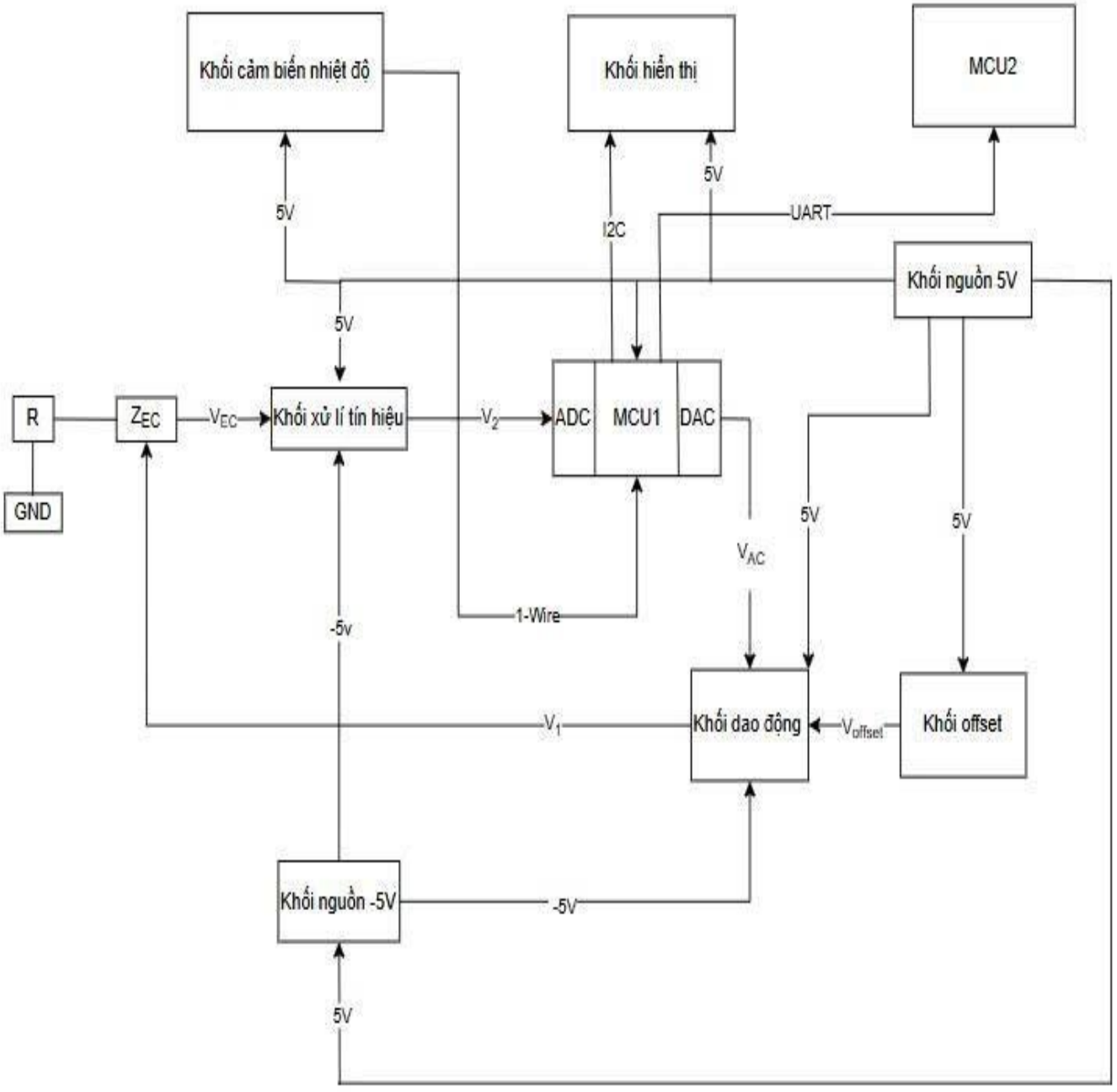
-Cổng terminal.

-IC nguồn: ICL7660, LM317.

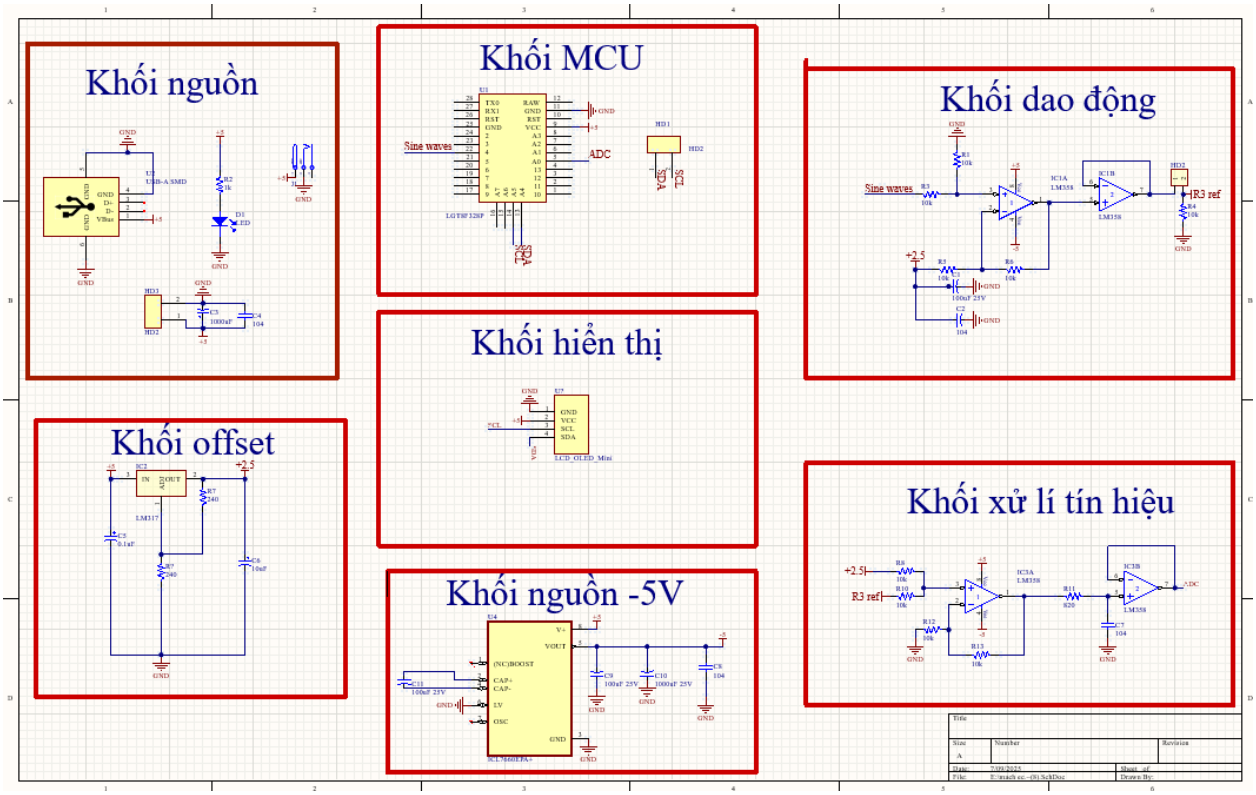
- Nguồn điện

Sử dụng nguồn adapter có độ chính xác cao thông số kĩ thuật 5V DC và dòng điện tối đa 1A

b) Sơ đồ khối của cảm biến EC



Dựa vào sơ đồ khối ta xây dựng được sơ đồ nguyên lý của cảm biến EC



c) Nguyên lí hoạt động

-Khối MCU sẽ phát sóng sine từ chân DAC của LGT8F328P (chân D4). Tín hiệu sóng sine phát ra có dạng $V_{AC}=Asin(wt)+V_{offset}$ do dải điện áp hoạt động của LGT8F328P từ 0-5V nên không thể xuất hiện điện áp âm do đó trong tín hiệu AC phát ra có thành phần V_{offset} là điện áp DC thành phần điện áp trung tâm để sóng sine có thể dao động xung quanh.

-Đối với cảm biến EC thì ta sẽ tạo ra tín hiệu AC không quá cao cũng không quá bé nên ta chọn V_{AC} có điện áp đỉnh đỉnh(peak to peak) là khoảng 1200mV để cho qua 2 đầu cảm biến.

-Code DAC phát sóng sine bằng vi điều khiển LGT8F328P

```
#include <Arduino.h>


#define DAC_PIN 4            // D4 = PD4 = DAC1 trên LGT8F328P

#define SAMPLES 256


uint8_t sineTable[SAMPLES];

volatile uint16_t idx = 0;
```

```
void setupSineTable(uint8_t offset, uint8_t amplitude) {
  for (uint16_t i = 0; i < SAMPLES; i++) {
    float angle = TWO_PI * i / SAMPLES;
    sineTable[i] = offset + amplitude * sin(angle);
  }
}

void setup() {
  // Tạo bảng sóng sine với offset 128 (~2.5V), biên độ ±31 (~0.6V)
  setupSineTable(128, 31);  // 1.2Vpp nếu Vref = 5V

  pinMode(DAC_PIN, ANALOG);          // Bật DAC1 tại D4
  analogWrite(DAC_PIN, sineTable[0]); // Mẫu đầu tiên

  // Cấu hình Timer1 để phát DAC ở tốc độ 256kHz (→ 1kHz sine)
  noInterrupts();
  TCCR1A = 0;
  TCCR1B = (1 << WGM12) | (1 << CS10);  // CTC mode, prescaler = 1
  OCR1A = 124;                      // (32MHz / 256kHz) - 1 = 124
  TIMSK1 = (1 << OCIE1A);              // Cho phép ngắt Timer1
  interrupts();
}

ISR(TIMER1_COMPA_vect) {
  analogWrite(DAC_PIN, sineTable[idx]);
  idx++;
  if (idx >= SAMPLES) idx = 0;
}
```

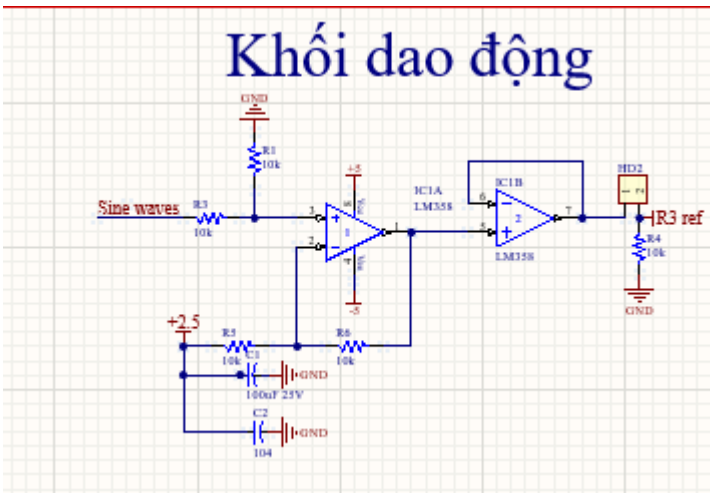
```
void loop() {
  // Không cần gì thêm
}
```

Hình ảnh kết quả đo VAC tại chân D4

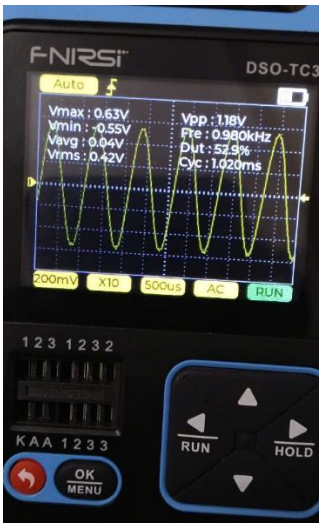
-Nhưng V_{offset} lại là thành phần không mong muốn khi cho qua dung dịch vì bản chất nó là thành phần điện áp DC nên ta phải loại bỏ thành phần này.

-Để có thể loại bỏ đi thành phần Voffset thì ta phải thiết kế 1 mạch tạo dao động có đầu ra $V_1=V_{AC}-V_{offset}$ =Asin(wt) đó chính là khối dao động có output là V_1

Sơ đồ nguyên lí của khối dao động



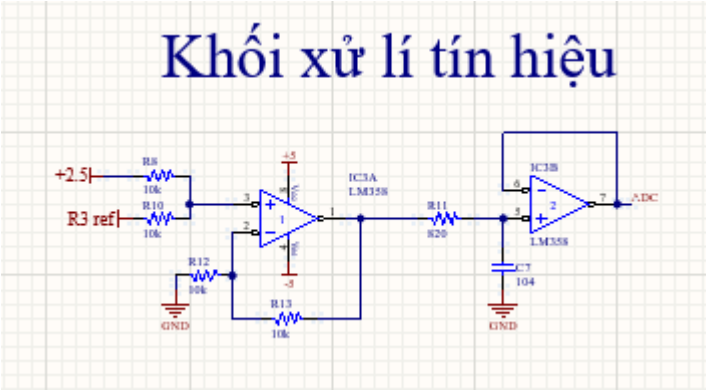
Hình ảnh kết quả đo V1



-Sau khi có tín hiệu đầu ra V_1 thì ta phải cho qua bộ đệm mục đích là để tách trở kháng giữa nguồn và tải vì đặc điểm của bộ đệm là có trở kháng vào cao và trở kháng ra thấp kết quả đầu ra sau khi đi qua dung dịch tín hiệu hình sine vẫn được giữ nguyên hình dạng mà không bị méo.

-Tín hiệu sau khi qua dung dịch là V_{EC} do đã loại bỏ thành phần DC nên lúc này tín hiệu hình sine sẽ dao động quanh mốc 0V cho nên đỉnh dưới sẽ có biên độ âm mà dải hoạt động của vi điều khiển là từ 0-5V nên khi đưa ngay tín hiệu V_{EC} vào chân ADC của vi điều khiển thì sẽ bị sai lệch thậm chí là hỏng vi điều khiển. Cho nên ta lại phải thiết kế 1 mạch xử lí tín hiệu V_{EC} sao cho giữ nguyên đặc tính biên độ đỉnh đỉnh của V_{EC} và dịch điện áp trung tâm lên 2.5V từ đó có thể loại bỏ được điện áp âm

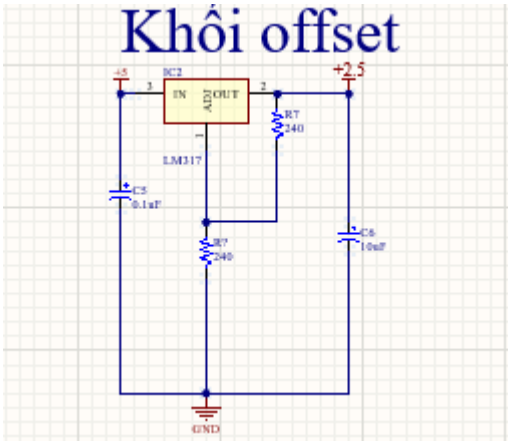
Sơ đồ nguyên lí khối xử lí tín hiệu



Hình ảnh đầu ra V2 của khối xử lí tín hiệu

-Chức năng của khối offset là để tạo ra điện áp Voffset có điện áp 2.5V ổn định để cấp cho khối dao động và khối xử lí tín hiệu

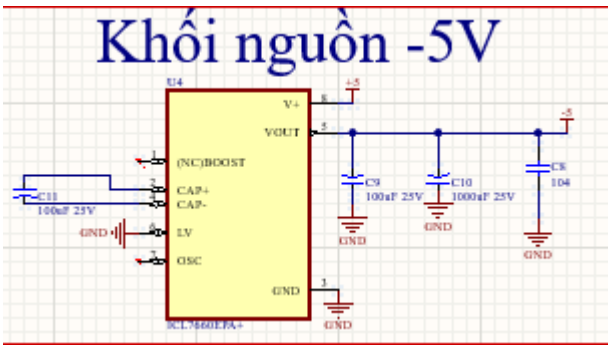
Sơ đồ nguyên lí của khối offset



Hình ảnh đầu ra Voffset

-Do trong mạch cần phải xử lí các tín hiệu có điện áp âm nên ta phải tạo ra điện áp âm để cấp cho các ic khuếch đại thuật toán LM358 làm điện áp tham chiếu cho các op-amp có điện áp là -5V.

Sơ đồ nguyên lí của khối nguồn -5V



Hình ảnh đầu ra của khối nguồn -5V

-Sau khi tín hiệu điện áp V_{EC} được xử lí thành điện áp V_2 nhưng bản chất biên độ đỉnh đỉnh vẫn được giữ nguyên không thay đổi, V_2 được đưa qua bộ đệm rồi đưa vào ADC của vi điều khiển để đọc giá trị điện áp V_2 giá trị này phản ánh sự suy giảm điện áp so với V_1, để có thể thấy được sự thay đổi của V_2 ta mắc 2 điện cực phân áp với điện trở R từ đó ta tính được trở kháng Z_{EC} của dung dịch qua công thức phân áp:

$$V_2=\frac{V_1 \cdot R}{z_{EC}+R}$$

-Code ADC đọc biên độ đỉnh đỉnh

```
    #define ADC_PIN A0
#define ADC_SAMPLES 256
#define VPP_BUFFER_SIZE 200

// === Bộ đệm mẫu ADC và trung bình Vpp ===
uint16_t adcSamples[ADC_SAMPLES];
float vppBuffer[VPP_BUFFER_SIZE];
uint16_t vppIndex = 0;
bool vppBufferFull = false;
float lastVpp = 0;

// === Biến cờ xử lý định kỳ (ví dụ: mỗi chu kỳ sine) ===
volatile bool readyToProcess = false;

// === Tùy chỉnh Vref nếu cần ===
const float VREF_MV = 5000.0;  // mV

void setup() {
  Serial.begin(115200);
  analogReference(DEFAULT);      // Vref = 5V
  analogReadResolution(12);      // 12-bit (0–4095)

  // === Mô phỏng: cứ mỗi 100ms thì xử lý 1 lần ===
  noInterrupts();
  TCCR2A = 0;
  TCCR2B = (1 << WGM21) | (1 << CS22); // CTC, prescaler = 64
  OCR2A = 124;
  TIMSK2 = (1 << OCIE2A);
  interrupts();
```

```
}

// === Ngắt Timer2: mỗi ~1ms, đếm đến 100ms thì đặt cờ xử lý ===
volatile uint16_t msCounter = 0;
ISR(TIMER2_COMPA_vect) {
  msCounter++;
  if (msCounter >= 100) {
    msCounter = 0;
    readyToProcess = true;
  }
}

void sampleADC() {
  uint16_t Vmin = 4095;
  uint16_t Vmax = 0;

  for (uint16_t i = 0; i < ADC_SAMPLES; i++) {
    analogRead(ADC_PIN);  // bỏ mẫu đầu
    uint16_t val = analogRead(ADC_PIN);
    adcSamples[i] = val;
    if (val < Vmin) Vmin = val;
    if (val > Vmax) Vmax = val;
  }

  // === Tính biên độ (Vpp) tính theo mV ===
  lastVpp = (Vmax - Vmin) * (VREF_MV / 4095.0);

  // === Lưu vào vòng đệm ===
  vppBuffer[vppIndex++] = lastVpp;
```

```
  if (vppIndex >= VPP_BUFFER_SIZE) {
    vppIndex = 0;
    vppBufferFull = true;
  }
}

float getAverageVpp() {
  float sum = 0;
  uint16_t count = vppBufferFull ? VPP_BUFFER_SIZE : vppIndex;
  for (uint16_t i = 0; i < count; i++) sum += vppBuffer[i];
  return (count > 0) ? (sum / count) : 0;
}

void loop() {
  if (readyToProcess) {
    readyToProcess = false;

    sampleADC();

    Serial.print("Current Vpp: ");
    Serial.print(lastVpp, 2);
    Serial.print(" mV | Avg (");
    Serial.print(vppBufferFull ? VPP_BUFFER_SIZE : vppIndex);
    Serial.print("): ");
    Serial.print(getAverageVpp(), 2);
    Serial.println(" mV");
  }
}
```

-Các giá trị V_1, V_2, R đã biết trước từ đó tìm ra nghiệm Z_{EC} sau đó ta hiệu chuẩn với dung dịch đã biết trước EC để tìm ra hệ số K của cảm biến sau đó giá trị EC sẽ được tính bởi công thức:

$$EC = \frac{K}{z_{EC}}$$

-Khi đo được giá trị EC của dung dịch thì ta có thể quy đổi sang độ mặn theo công thức

$$\text{Salinity (ppt)} \approx EC \times 0.64$$

Trong đó:

EC: độ dẫn điện, đơn vị là mS/cm

Salinity: độ mặn, đơn vị là ppt (parts per thousand – phần nghìn)

Trước khi chuyển đổi sang độ mặn thì ta phải áp dụng thuật toán bù nhiệt để đưa giá trị EC về chuẩn giá trị ở 25 độ C bằng công thức:

$$EC_{25} = \frac{EC_T}{1 + \alpha(T - 25)}$$

EC_{25} : EC đã bù nhiệt, tại 25°C (mS/cm hoặc µS/cm)

EC_T : EC đo được tại nhiệt độ thực tế T

T: Nhiệt độ thực tế

α: Hệ số bù nhiệt (Temperature Coefficient), thường từ 0.018 – 0.02 (tức 1.8–2%/°C)

-Nhiệt độ T thực tế sẽ được đo bằng cảm biến nhiệt độ DS18B20 sử dụng giao tiếp 1-wire

-Code đọc nhiệt độ từ cảm biến DS18B20:

```
// Chân dữ liệu của DS18B20 kết nối với D2
#define ONE_WIRE_BUS 2


// Khởi tạo giao tiếp 1-Wire và đối tượng cảm biến
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);


void setup() {
  Serial.begin(9600);
  sensors.begin();  // Khởi động cảm biến DS18B20
```

```
}

void loop() {
  sensors.requestTemperatures(); // Yêu cầu đọc nhiệt độ
  float tempC = sensors.getTempCByIndex(0); // Lấy nhiệt độ từ cảm biến số 0

  Serial.print("Nhiet do: ");
  Serial.print(tempC);
  Serial.println(" °C");

  delay(1000); // Đọc mỗi 1 giây
}
```

-Sau khi có được độ mặn của dung dịch thì vi điều khiển LGT8F328P ở khối MCU sẽ sử dụng giao thức I2C để hiển thị dữ liệu ra màn hình LCD

-Code giao tiếp I2C với LGT8F328P cho khối hiển thị:

```
// Định nghĩa kích thước màn hình OLED
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

// Khởi tạo đối tượng màn hình OLED dùng I2C
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

// Thông số hiệu chỉnh EC
float alpha = 0.019;  // Hệ số bù nhiệt
float temperature = 25.0; // Nhiệt độ mặc định (có thể thay bằng cảm biến thật)

void setup() {
  Serial.begin(9600);

  // Khởi động OLED
```

```
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("Không tìm thấy màn hình OLED"));
    for (;;);
  }

  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 0);
  display.println("Khoi dong OLED...");
  display.display();
  delay(1000);
}

void loop() {
  // Giả lập điện áp đo được từ cảm biến EC (hoặc thay bằng analogRead)
  float voltage = analogRead(A0) * (5.0 / 1023.0);  // đọc từ chân A0
  float EC = voltage * 2.0; // Tạm giả lập: mỗi 1V tương ứng 2 mS/cm
  float EC25 = EC / (1 + alpha * (temperature - 25)); // Bù nhiệt độ

  // Đổi sang độ mặn:
  float salinity = EC25 * 0.64;  // ppt

  // Hiển thị lên OLED
  display.clearDisplay();
  display.setCursor(0, 0);
  display.println("EC Sensor Display");
  display.setCursor(0, 16);
  display.print("EC25: ");
```

```
  display.print(EC25, 2);
  display.println(" mS/cm");

  display.setCursor(0, 32);
  display.print("Salinity: ");
  display.print(salinity, 2);
  display.println(" ppt");

  display.display();

  delay(1000); // Cập nhật mỗi giây
}
```

-Để có thể đóng gói dữ liệu EC và độ mặn để truyền đi thì phải chọn giao thức truyền thông giao tiếp giữa 2 vi điều khiển bằng giao thức UART

-UART (Universal Asynchronous Receiver/Transmitter) là giao thức truyền dữ liệu nối tiếp (serial communication) không đồng bộ, dùng để truyền và nhận dữ liệu giữa 2 thiết bị

-Code đóng gói dữ liệu và truyền tới vi điều khiển khác bằng UART

```
const int EC_PIN = A0;

float alpha = 0.019;          // Hệ số bù nhiệt độ
float temperature = 25.0;     // Tạm thời giả lập nhiệt độ (°C)

// Tạm quy 1V tương ứng 2.0 mS/cm (tùy thuộc mạch của bạn)
float voltageToEC(float voltage) {
  return voltage * 2.0;
}

void setup() {
  Serial.begin(9600); // Gửi dữ liệu qua UART TX
```

```
  analogReference(DEFAULT); // hoặc INTERNAL nếu bạn dùng Vref khác
}

void loop() {
  // Đọc điện áp từ cảm biến EC (tín hiệu DAC qua buffer → A0)
  int raw = analogRead(EC_PIN);
  float voltage = raw * (5.0 / 1023.0); // nếu dùng Vref = 5V

  // Chuyển sang EC thô
  float ecRaw = voltageToEC(voltage);

  // Bù nhiệt độ
  float ec25 = ecRaw / (1 + alpha * (temperature - 25));

  // Tính độ mặn
  float salinity = ec25 * 0.64;

  // Đóng gói chuỗi JSON đơn giản
  Serial.print("{\"ec\":");
  Serial.print(ec25, 2);
  Serial.print(",\"sal\":");
  Serial.print(salinity, 2);
  Serial.println("}");

  delay(1000); // Gửi mỗi 1 giây
}
```

Hình ảnh đầu ra khối hiển thị

II. Hướng dẫn sử dụng phần mềm Altium 21

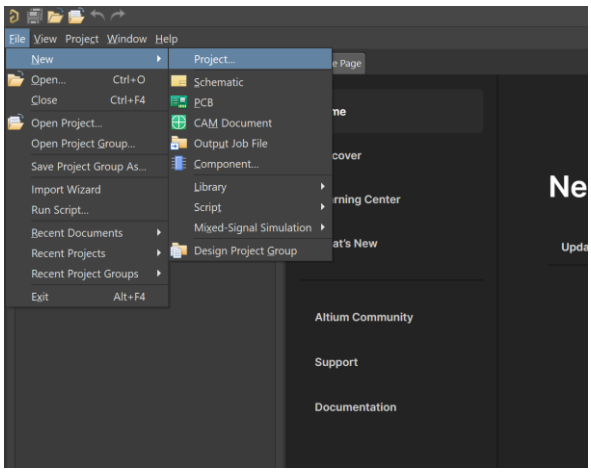
1. Tạo Project

Việc tạo Project là bước đầu tiên và rất quan trọng khi bắt đầu thiết kế một mạch điện tử trong Altium Designer. Project giúp ta quản lý tập trung toàn bộ các file như sơ đồ nguyên lý (*Schematic*), mạch in (*PCB Layout*), thư viện linh kiện, và các tập tin báo cáo (DRC, BOM...) trong cùng một cấu trúc rõ ràng.

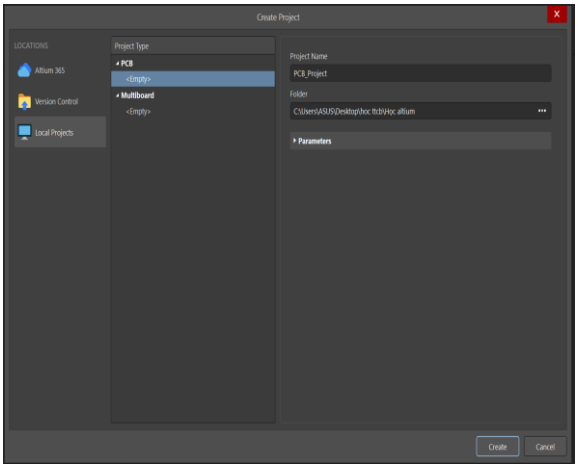
Quản lý theo project không chỉ giúp quá trình thiết kế trở nên gọn gàng, dễ kiểm soát, mà còn tránh được các lỗi sai sót như thiếu file, mất liên kết giữa schematic và PCB. Ngoài ra, nó cũng giúp dễ dàng chia sẻ hoặc lưu trữ toàn bộ thiết kế dưới dạng một thư mục duy nhất.

Các bước tạo 1 Project mới hoàn chỉnh

- Bước 1 : Trên giao diện chính, chọn mục File → New → Project. Sau đó đặt tên cho Project và chọn nơi lưu Project như ở 2 hình 1.1 và 1.2. chọn Create để hoàn tất tạo Project quản lý sơ bộ .

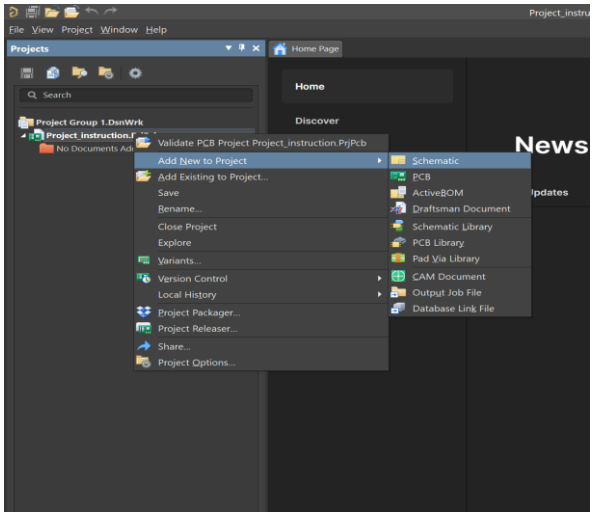


Hình 1.1

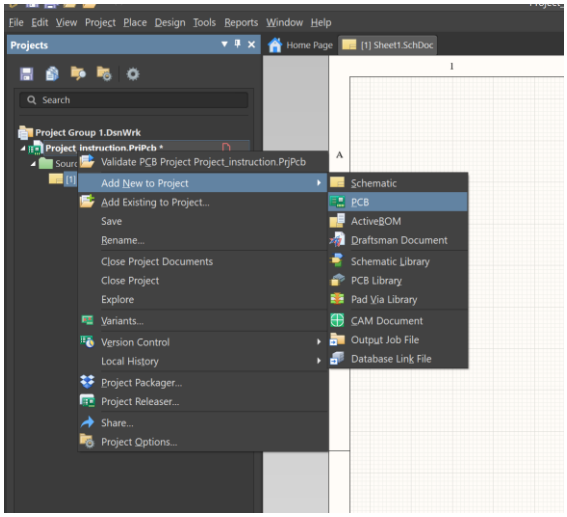


Hình 1.2

- Bước 2: Thêm vào Project vừa tạo 2 file schematic và PCB. Sau khi tạo xong lưu lại (Ctrl+S) trước khi tiếp tục các bước tiếp theo. Bước này được minh họa ở hình 1.3 và 1.4



Hình 1.3



Hình 1.4

2. Một số phím tắt cơ bản khi sử dụng Altium

a) Phím tắt trong môi trường Schematic (Sch)

- Các phím dùng để di chuyển linh kiện:
  - M (Move): Di chuyển linh kiện
  - Space: Xoay linh kiện theo chiều kim đồng hồ
- Các phím dùng để sao chép:
  - C (Copy): Sao chép linh kiện
  - Ctrl + C / Ctrl + V: Sao chép và dán
- Phím vẽ dây kết nối:

- W (Wire): Vẽ dây nối giữa các chân linh kiện

- Phím điều chỉnh ký hiệu và tên linh kiện:

- T + A + A: Đánh số cho toàn bộ linh kiện
- Đánh nhãn cho linh kiện:
- P+N : Các linh kiện có nhãn giống nhau sẽ kết nối với nhau
- Update các linh kiện từ file Schematic sang PCB :
- D+U

b) Phím tắt trong môi trường PCB Layout (Pcb)

- Phím dùng để di chuyển, xoay linh kiện:

- M (Move): Di chuyển linh kiện

- R (Rotate): Xoay linh kiện

- Space: Xoay linh kiện khi đang kéo

- Phím dùng để vẽ đường dẫn (routing):

- P → T (Place → Track): Vẽ đường mạch

- Shift + Spacebar: Thay đổi kiểu vẽ dây (45°, 90°, đường cong...)

- Phím để thêm Via, Pad, Hole:

- P → V: Đặt Via

- P → P: Đặt Pad

- P → H: Đặt Hole

- Phím kiểm tra khoảng cách, đo lường:

- Ctrl + M: Đo khoảng cách giữa hai điểm bất kỳ

- Tab (khi đang vẽ): Mở thuộc tính đối tượng

- Phím để bật tắt lớp, xem nhanh:

- L: Hiển thị/ẩn layer

- Shift + S: Chuyển đổi chế độ single-layer mode

- Chuyển đổi mạch 2D hoặc 3D : Nhấn 2 hoặc 3
- Đổ đồng cho mạch : P+G

3. Thêm thư viện và tạo thư viện trong Altium 21

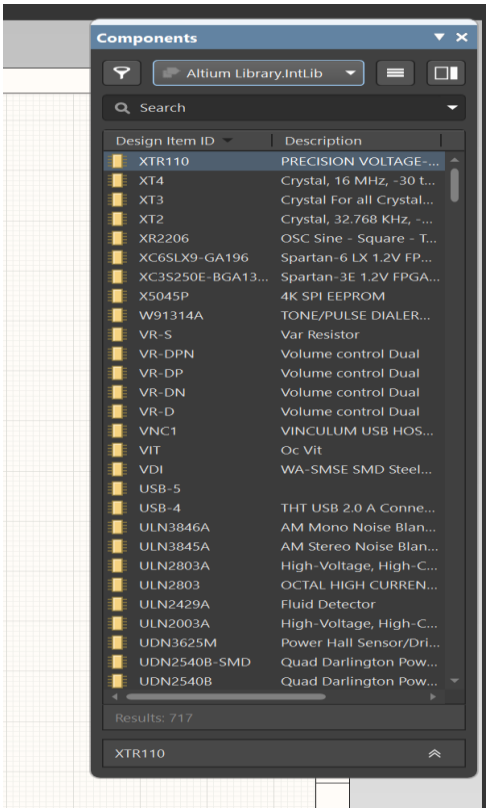
a) Thêm thư viện có sẵn

- Bước 1: Chuẩn bị thư viện
- Tải thư viện định dạng .SchLib (thư viện sơ đồ nguyên lý) hoặc .PcbLib (thư viện footprint PCB).

- Lưu các file thư viện vào một thư mục dễ nhớ trên máy tính (ví dụ: D:\ThuVien_LinhKien\).

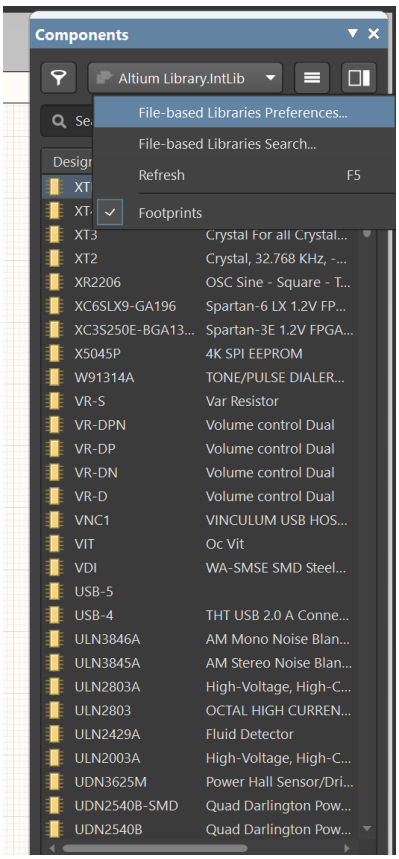
- Bước 2 : Thêm thư viện vào Altium

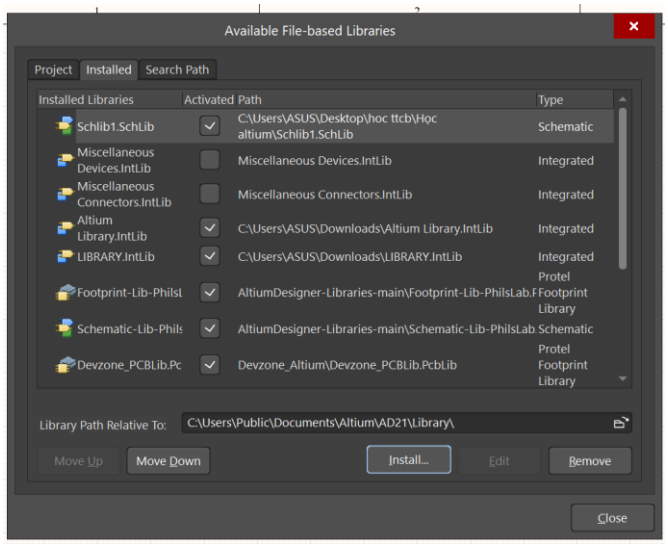
- Panels => Components : Mở bảng thư viện linh kiện sẽ ra được bảng như hình 3.1



Hình 3.1

- Thêm file thư viện đã tải vào phần mềm bằng các thao tác nhấn vào biểu tượng  chọn File-based Libraries Prefernce như hình 3.2 , 3.3





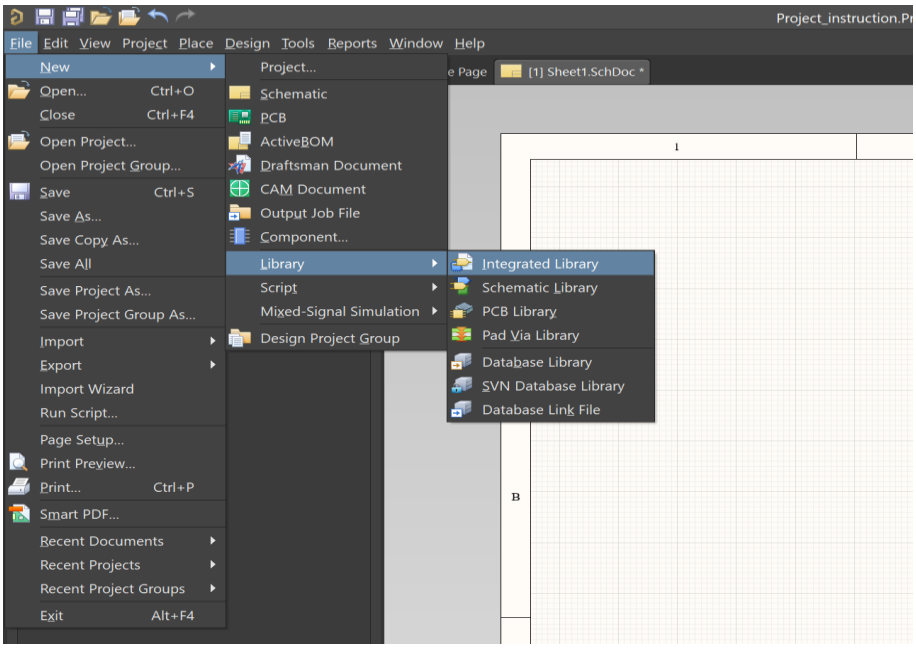
Hình 3.2 Hình 3.3

- Chọn Install.. rồi tìm kiếm file thư viện đã tải về trong máy tính, nhấn Open.

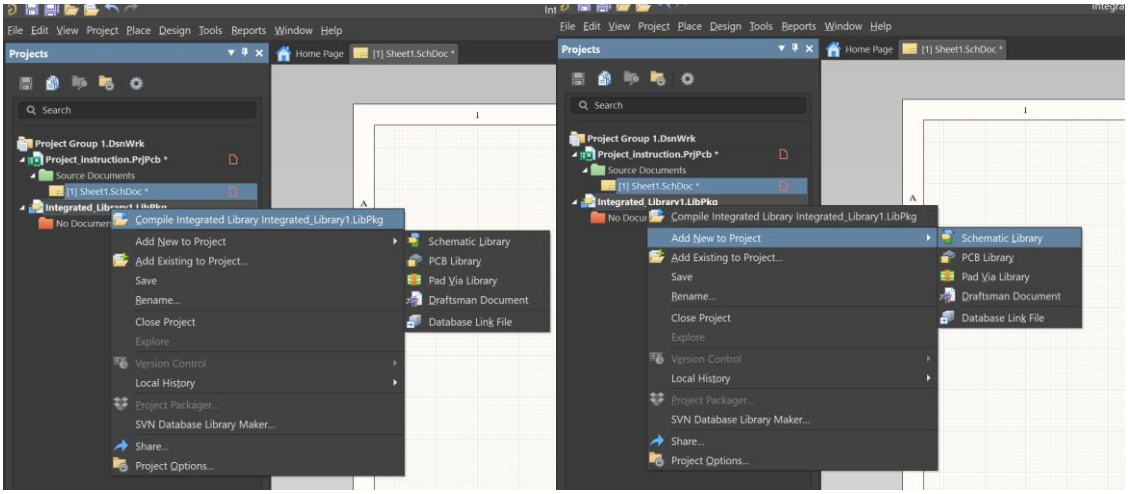
b) Tạo thư viện linh kiện

- Bước 1: Tạo library chứa các linh kiện mình muốn thêm vào

Tạo Library tương tự như tạo Project : File => New => Library => Integrated Library . Sau đó thêm Schematic library và PCB library vào file thư viện vừa tạo như hình 3.4 , 3.5 .



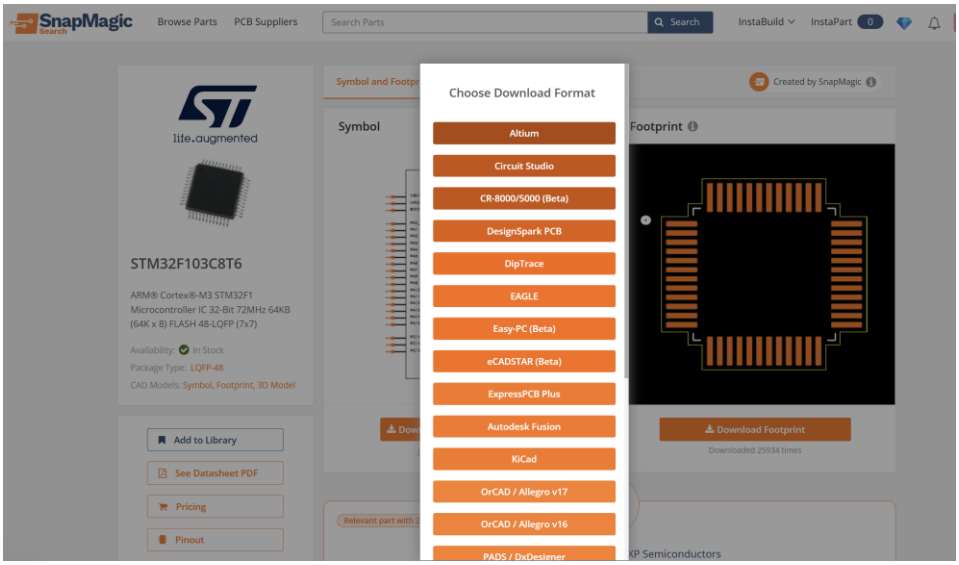
Hình 3.4



Hình 3.5

- Bước 2 : Truy cập trang Web **snapeda.com** để tải những linh kiện cần thêm vào thư viện mới.

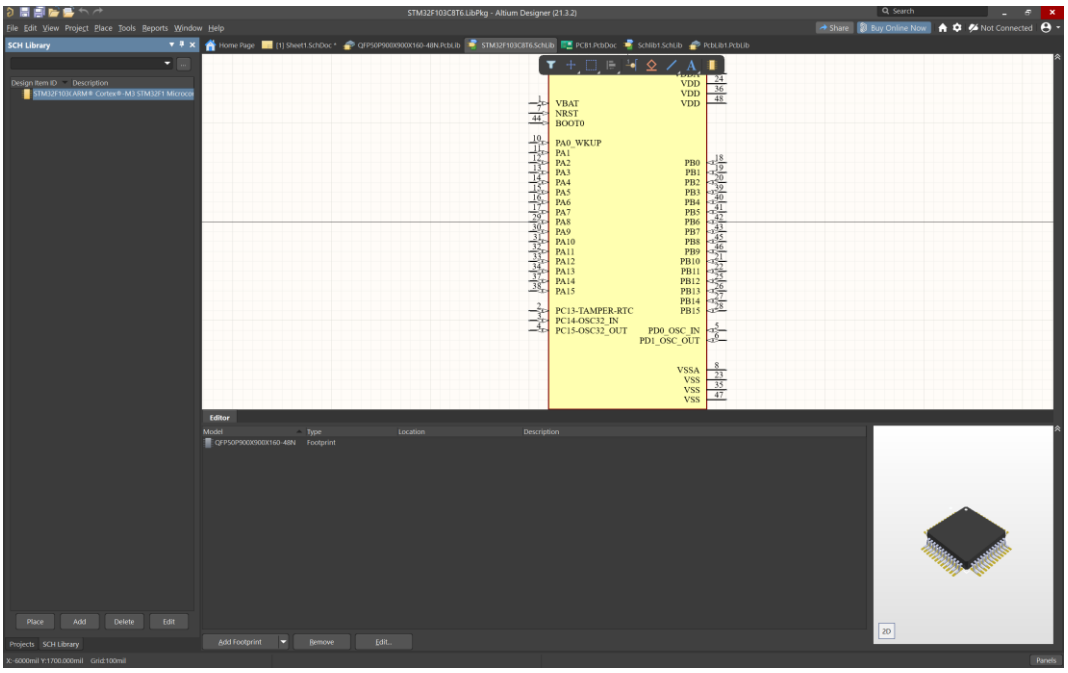
- Tìm kiếm linh kiện cần tải về, nhấn **Download footprint => Altium**



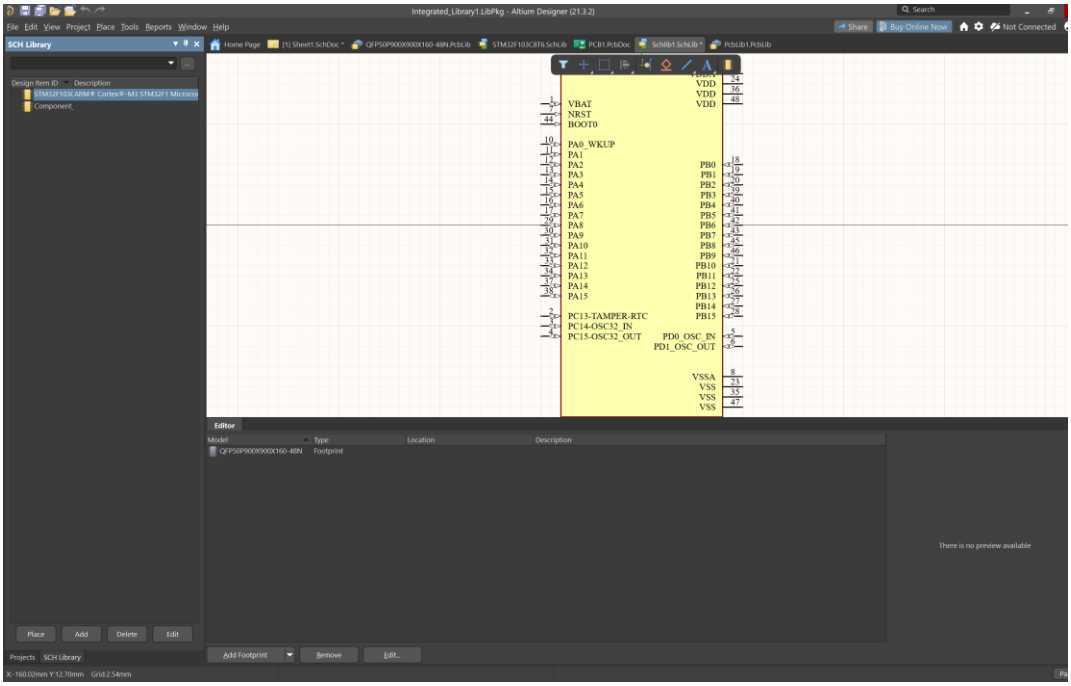
Hình 3.6

- Bước 3 : Mở 2 file thư viện Sch và PCB vừa tải về trên Altium .
- Bước 4 : Sao chép 2 file thư viện Sch và PCB vừa tải trên Snapeda vào 2 file thư viện Sch và PCB mà ta tạo ở bước 1.

- Chọn file cần thêm (file Sch) : Nhấn vào SCH library(của file cần copy) => Chọn STM32F103..=> Ctrl+C (được Hình 3.7)
- Chọn file SCH library (đã tạo ở bước 1) => Nhấn vào SCH library => Ctrl+V.(được Hình 3.8)



Hình 3.7



Hình 3.8

- Tương tự với file PCB library

Bước 5 : Thêm thư viện tạo ở bước 1 vào altium như phần 3a.

4. Vẽ mạch Schematic và PCB

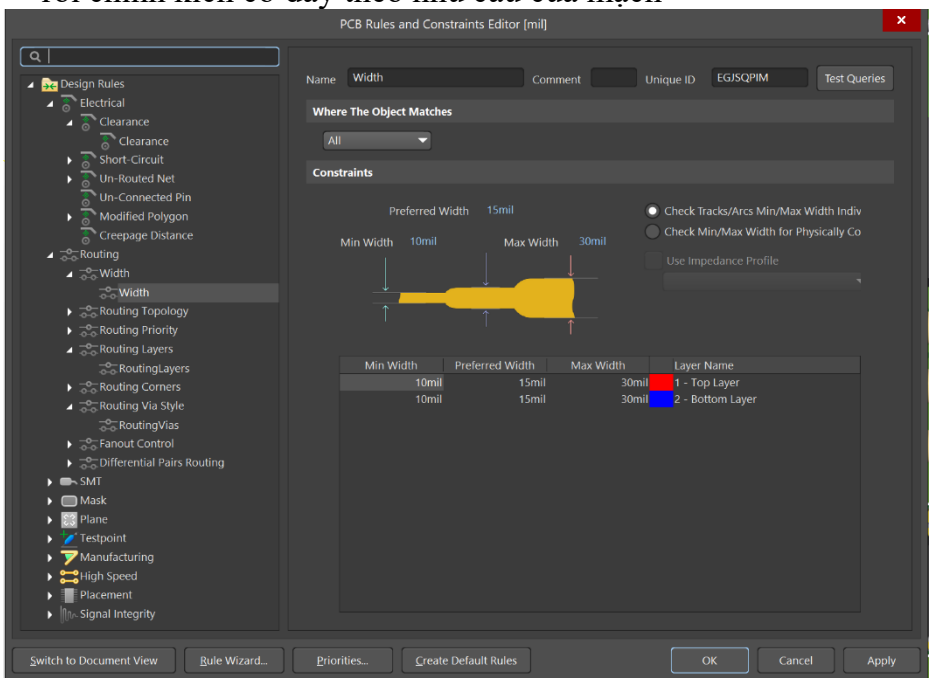
a) Vẽ mạch Schematic

- Bước 1 : Lấy linh kiện từ thư viện **Panels => Componants => Chọn thư viện có linh kiện cần lấy ra (** nếu thư viện không có thì làm theo phần 3b)
- Bước 2 : Sắp xếp linh kiện rồi chỉnh lại các thông số theo sơ đồ nguyên lý và đi dây
- Bước 3 : Đánh số lại toàn bộ linh kiện (T+A+A)

b) Vẽ PCB

- Bước 1 : Update sơ đồ mạch nguyên lý (Sch) sang mạch PCB (Sử dụng phím tắt D+U)
- Bước 2 : Sắp xếp các linh kiện vào bo mạch sao cho phù hợp với việc đi dây
- Bước 3 : Đi dây cho mạch PCB

- Chỉnh kích cỡ của dây trước khi đi : Design => Rules.. => Width được như hình 4.1 , rồi chỉnh kích cỡ dây theo nhu cầu của mạch

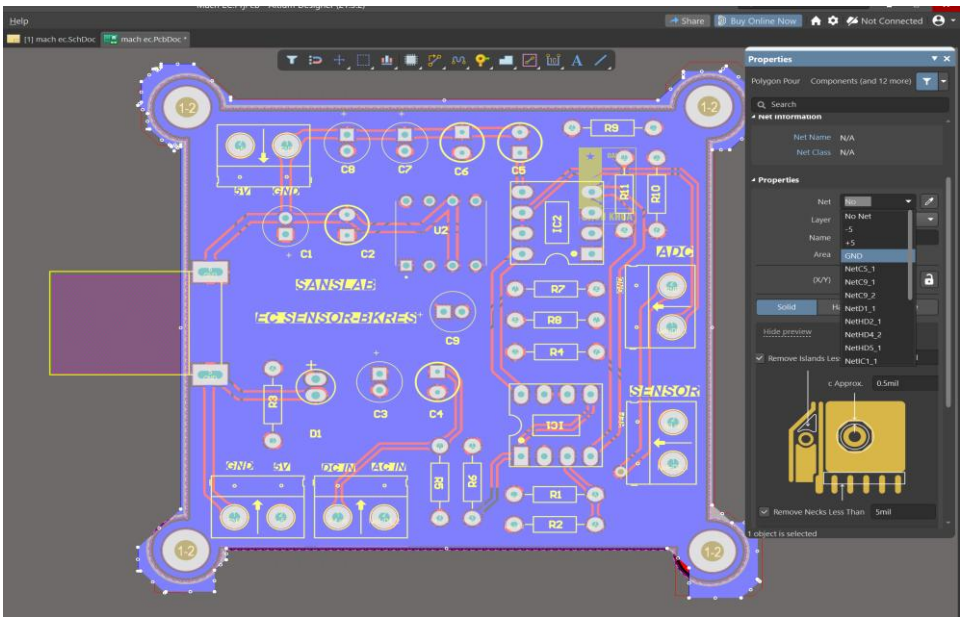


Hình 4.1

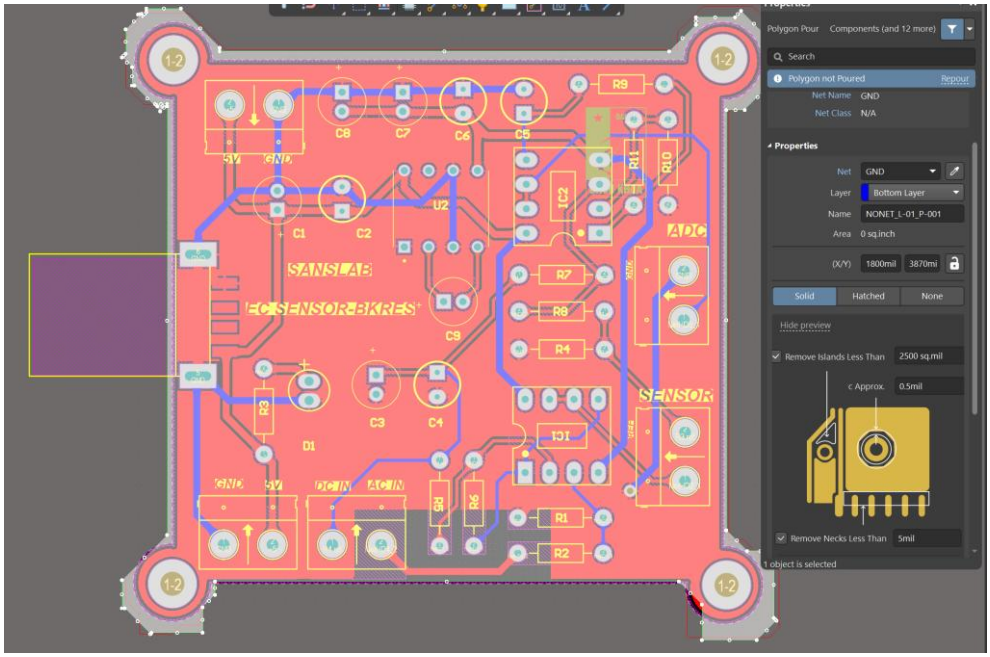
- Đi dây cho toàn mạch

- Bước 4 : đổ đồng cho mạch PCB (P+G)

- Nhấn P+G rồi đi đường bao quanh bo mạch PCB . Nhấn chuột phải ra ngoài.
- Panels => Properties => Net => GND , như hình 4.2 .
- Chọn repour để hoàn thành đổ đồng , như hình 4.3 .



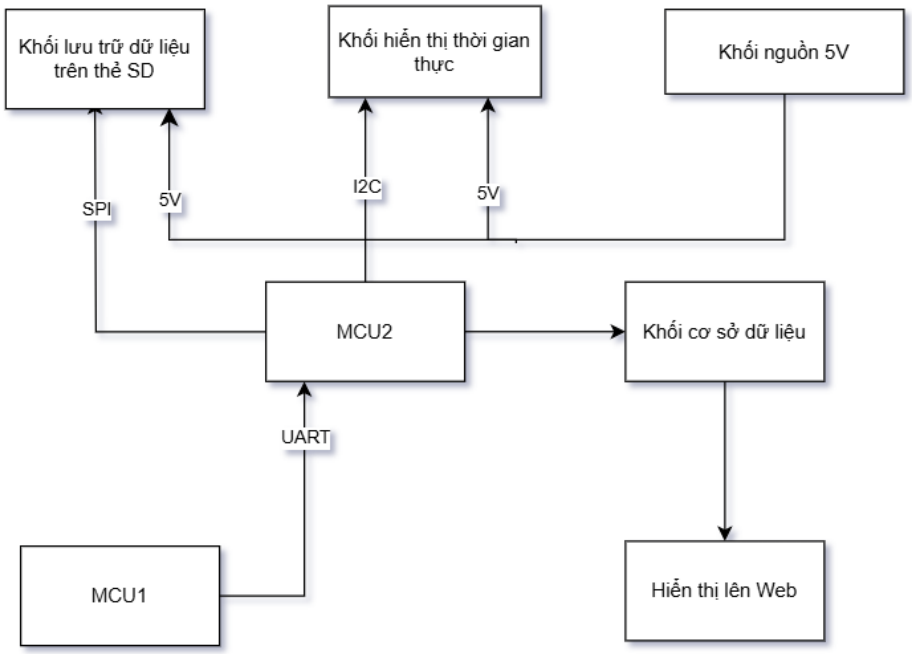
Hình 4.2



Hình 4.3

III . Truyền thông

1. Sơ đồ khối truyền thông



Hình 2.1: Sơ đồ khối truyền thông

Nguyên lý hoạt động lưu trữ vào SDCard:
ESP32 đọc tín hiệu analog từ cảm biến độ mặn, sau đó chuyển đổi thành giá trị số thông qua bộ ADC. Giá trị này được xử lý và quy đổi ra độ mặn. Tiếp theo, ESP32 lấy thời gian thực từ module RTC hoặc NTP để gắn kèm thời gian cho dữ liệu. Cuối cùng, ESP32 sử dụng giao tiếp SPI để truyền dữ liệu và ghi vào thẻ nhớ SD dưới dạng file văn bản.

Nguyên lý hoạt động của realtime clock:

ESP32 giao tiếp với module DS3231 qua giao tiếp I2C để lấy thời gian thực (giờ, phút, giây, ngày, tháng, năm). Sau đó, ESP32 xử lý dữ liệu thời gian và hiển thị lên màn hình LCD (cũng giao tiếp qua I2C). Quá trình này lặp lại liên tục theo chu kỳ để cập nhật thời gian thực trên LCD.

2. Giới thiệu linh kiện và hoàn thiện hệ thống

a) Khối vi điều khiển

Mục đích:

Là khối trung tâm của hệ thống, xử lý, tính toán các dữ liệu từ cảm biến, đưa ra các kết quả phù hợp và điều khiển các khối khác.

b) Linh kiện sử dụng

Hình 2.2 : ESP 32

- Ưu điểm:
- Tích hợp WiFi và Bluetooth, hỗ trợ IoT
- Tốc độ xử lý cao hơn nhiều so với Arduino Uno
- Nhiều chân GPIO, hỗ trợ ADC/DAC, PWM, SPI, I2C, UART
- Hỗ trợ nhiều thư viện và cộng đồng phát triển mạnh
- Nhược điểm:

- Lập trình phức tạp hơn Arduino Uno
- Một số module cần điện áp 3.3V, không tương thích trực tiếp với 5V
- Các thông số cơ bản:

- Vi điều khiển chính: Tensilica Xtensa LX6 Dual-core (hoặc Single-core tùy phiên bản)
- Nguồn vào: 3.3V (từ USB 5V qua bộ ổn áp)
- Flash Memory: 4MB (tùy board)
- SRAM: ~520KB
- WiFi: 802.11 b/g/n
- Bluetooth: v4.2 BR/EDR và BLE
- GPIO: 30 chân (nhiều chân đa chức năng)
- Tốc độ xung nhịp: lên đến 240MHz

c) Khối đo thời gian thực

- Giới thiệu

Khối thời gian thực có chức năng lưu trữ thông tin ngày tháng năm cũng như giờ phút giây, nó sẽ hoạt động như một chiếc đồng hồ và có thể cung cấp cho vi điều khiển thời gian thực.

Hình 2.3 : Modun thời gian thực.

- Module thời gian thực RTC DS3231

- **Thông số chính của DS3231:**

Điện áp hoạt động: 2.3V – 5.5V (tương thích với cả 3.3V và 5V)

Sai số thời gian: ±2 ppm (tương đương ~1 phút/năm)

Tích hợp bộ bù nhiệt độ (TCXO): giúp duy trì độ chính xác cao ngay cả khi nhiệt độ thay đổi.

Lưu trữ thời gian: Giờ, phút, giây, ngày, thứ, tháng, năm (bao gồm cả năm nhuận)

Giao tiếp: I2C (địa chỉ mặc định 0x68)

Tích hợp pin dự phòng (thường là pin CR2032): giúp đồng hồ vẫn chạy khi mất điện.

Có hai ngõ ra báo thức (Alarm 1 & 2) và tín hiệu Square Wave có thể lập trình.

Ưu điểm DS3231:

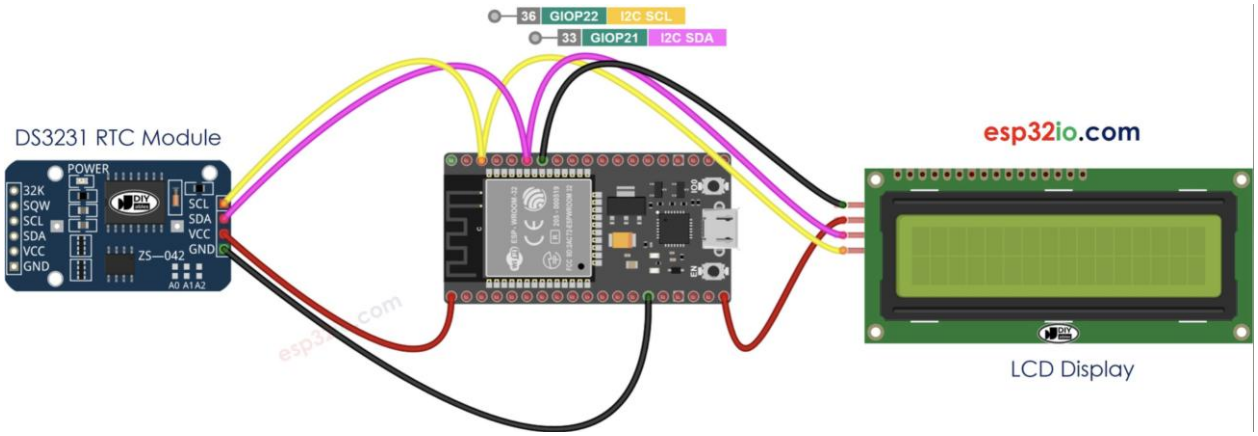
- Rất chính xác (±1 phút/năm)
- Có bù nhiệt độ tự động
- Tích hợp thạch anh và pin dự phòng
- Hỗ trợ báo thức và xuất xung vuông
- Giao tiếp I2C, dễ dùng với Arduino/ESP32

Nhược điểm:

- Giá cao hơn DS1307
- Không có EEPROM tích hợp
- Chỉ hỗ trợ I2C, không có SPI

- Sơ đồ kết nối chân

ESP32	RTC DS3231
GND	GND
5V	VCC
G21	SDA
G22	SCL

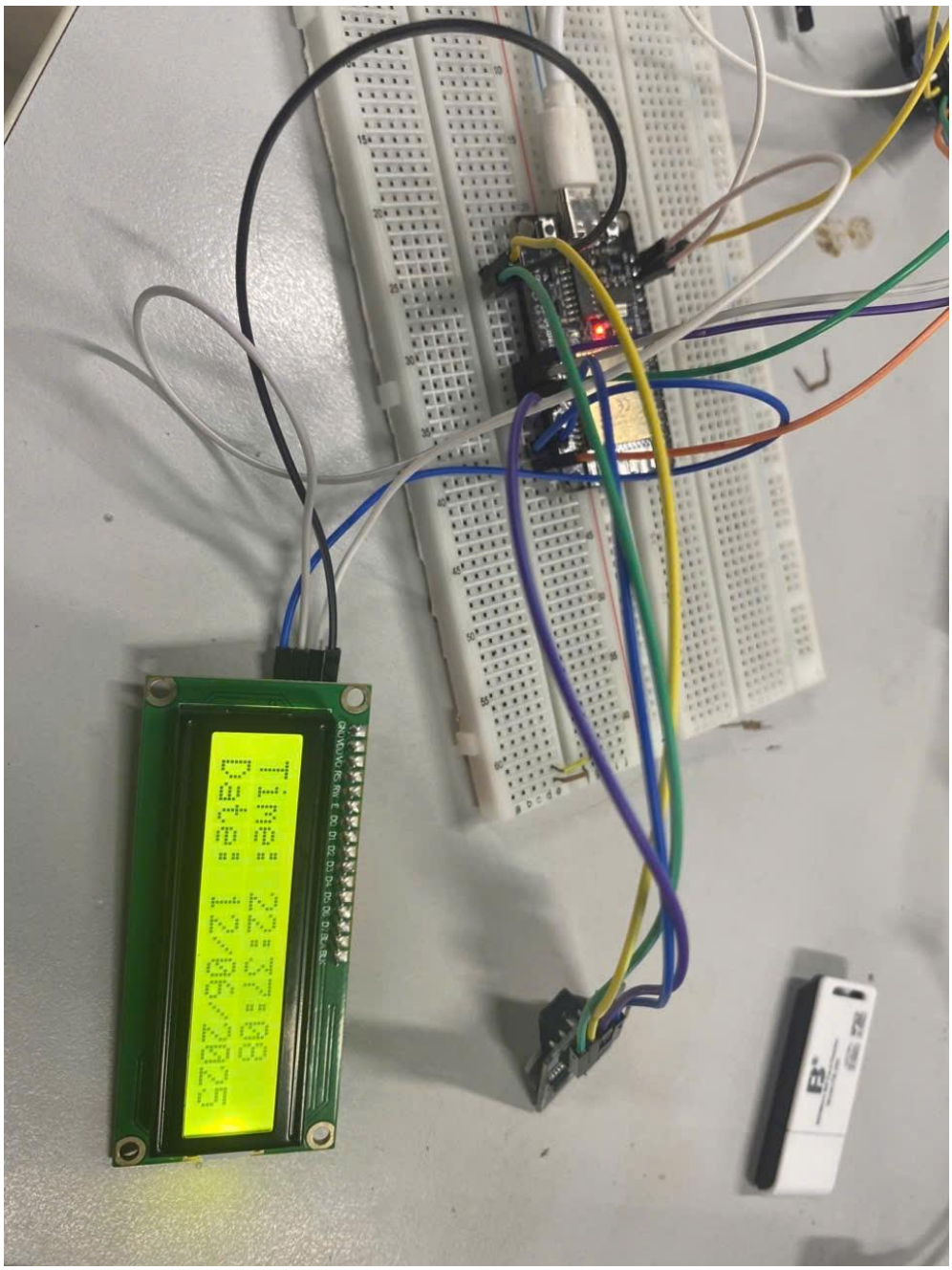


Hình 2.4 : Sơ đồ kết nối thời gian thực.

real_time_clock.ino

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <RTClib.h>

// LCD I2C 16x2
LiquidCrystal_I2C lcd(0x27, 16, 2);

// RTC
RTC_DS3231 rtc;

unsigned long lastUpdate = 0;

void setup() {
  Serial.begin(115200);

  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Khoi dong...");

  delay(1000);
  lcd.clear();

  if (!rtc.begin()) {
    Serial.println("RTC khong hoat dong");
    while (1);
  }
```

Hình 2.5 : Code realtime clock



Hình 2.6 : Kết quả quản lý thời gian thực

d) Khối lưu dữ liệu

➢ Mục đích

Khối lưu dữ liệu nhằm mục đích lưu trữ dữ liệu trong quá trình đo và thu thập dữ liệu cho các mục đích sau này. Khối lưu dữ liệu không bị ảnh hưởng bởi các kết nối không gian mà lưu trực tiếp sau mỗi lần đo.

➢ Module Sdcard

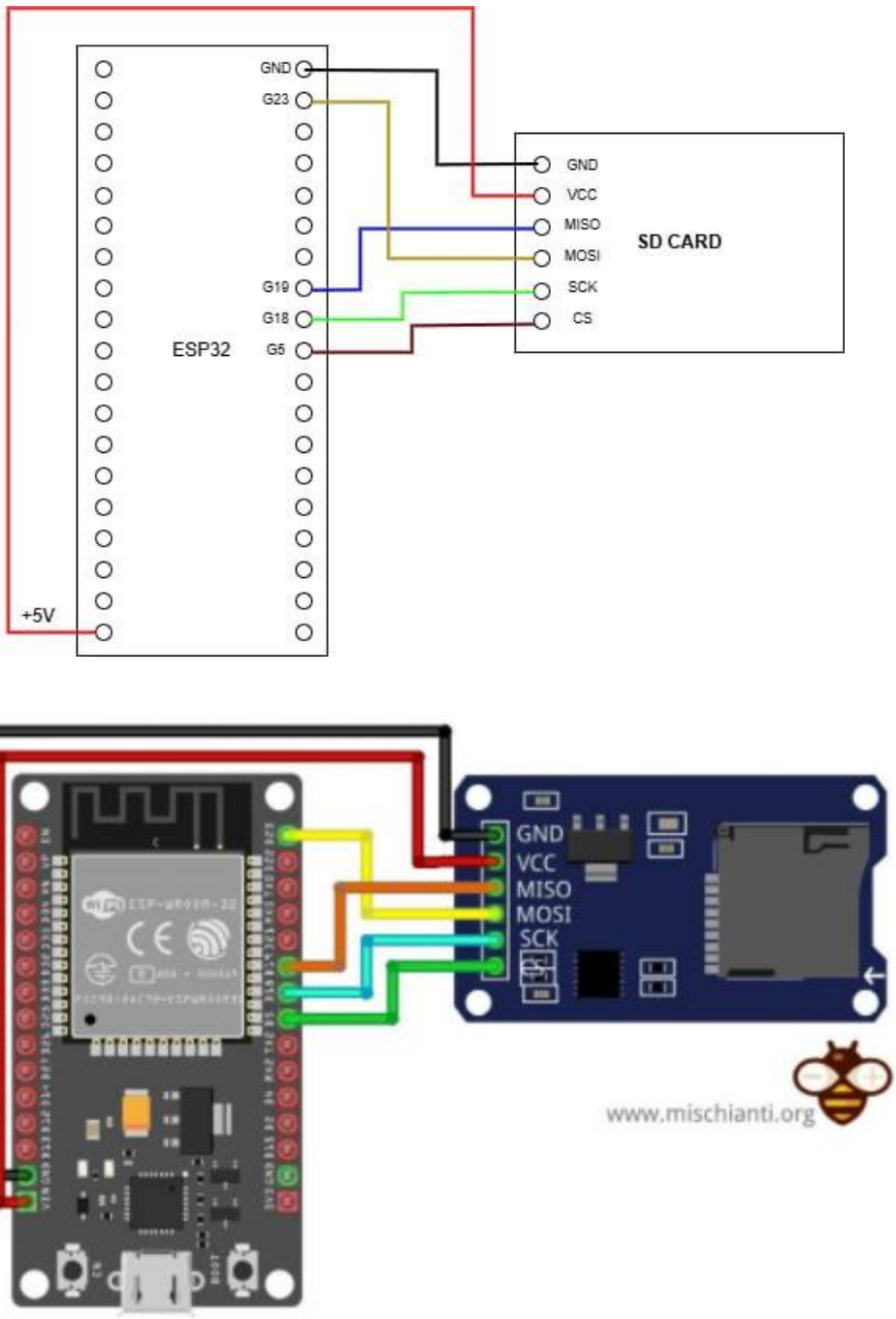
- Giới thiệu: Module Sdcard là module hỗ trợ cho việc giao tiếp giữa bộ nhớ của thẻ SD với vi điều khiển nhằm mục đích truy xuất dữ liệu trong thẻ SD.

- Thông số:
. Điện áp hoạt động: 3.3 – 5VDC
. Chuẩn giao tiếp: SPI
. Hỗ trợ định dạng FAT16 và FAT32

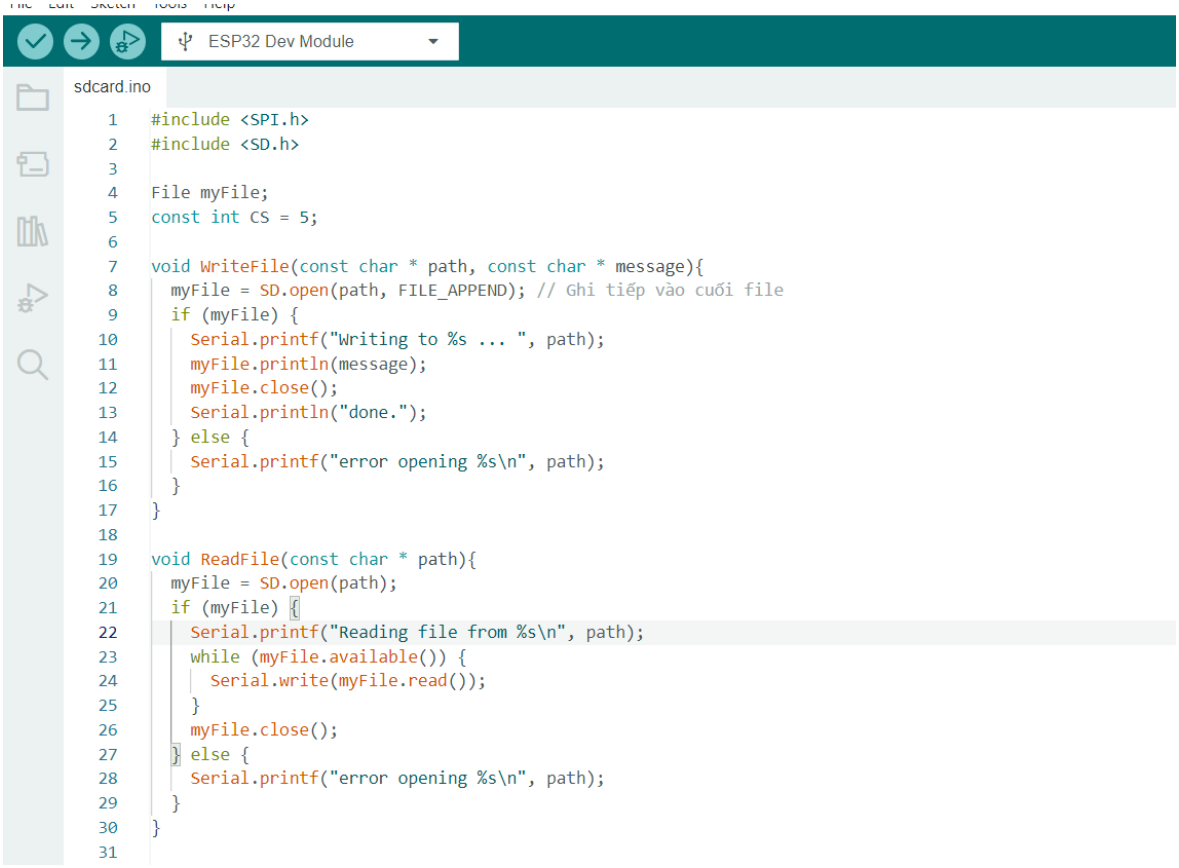
Hình 2.7 : Thẻ nhớ SDcard

- Sơ đồ kết nối với ESP32

ESP32	SDCard
GND	GND
Vin	VCC
G23	MOSI
G19	MISO
G18	SCK
G5	CS



Hình 2.8 : Sơ đồ kết nối SDcard



Hình 2.9 : Code lưu trữ thẻ SDcard

```
RTC_Time, Salinity(ppt), Temperature(C)
2025-06-12 22:35:28, 33.50, 30.5
2025-06-12 22:36:28, 26.40, 26.7
2025-06-12 22:37:28, 21.80, 34.4
2025-06-12 22:38:28, 19.90, 31.0
2025-06-12 22:39:28, 26.40, 30.7
2025-06-12 22:40:28, 31.20, 34.9
RTC_Time, Salinity(ppt), Temperature(C)
2025-07-10 22:36:50, 24.40, 34.9
```

Hình 2.10 : Kết quả lưu dữ liệu vào thẻ SD

e) Khối hiển thị

➢ Mục đích

Hỗ trợ hiển thị dữ liệu tức thời ngay tại vị trí đo

➢ Màn hình LCD 1602 - Thông số kỹ thuật .

- Nguồn cung cấp: 5VDC .
- Sơ đồ chân:

No	Symbol	Function
1	VSS	Ground
2	VDD	5V +
3	V0	Contrast
4	RS	Register
5	RW	Read/Write
6	E	Enable
7	D0	Data bus
8	D1	Data bus
9	D2	Data bus
10	D3	Data bus
11	D4	Data bus
12	D5	Data bus
13	D6	Data bus
14	D7	Data bus
15	A	Anode (5V+)
16	K	Cathode (GND)

Hình 2.11 : Sơ đồ chân của màn hình LCD 1602

f) Module phát wifi

Thông số kỹ thuật module WiFi 4G LTE

- Tên thiết bị: Telkomsel Mobile Broadband
- Chuẩn kết nối: 4G LTE, tốc độ tối đa 300 Mbps
- SSID mặc định: 4GUFI-XX
- Mật khẩu WiFi mặc định: 1234567890
- Địa chỉ IP quản trị: 192.168.100.1
- Tài khoản đăng nhập: admin / admin
- Xuất xứ: Trung Quốc (Made in China)

3. Firebase

a) Giới thiệu Firebase Realtime Database

Firebase Realtime Database là một dịch vụ cơ sở dữ liệu đám mây được cung cấp bởi Google, cho phép lưu trữ và đồng bộ dữ liệu theo thời gian thực giữa client (ứng dụng) và server.

Vai trò

- Là nơi lưu trữ dữ liệu cảm biến, cho phép giao diện web đọc và truy vấn lịch sử dữ liệu theo thời gian thực.
- Khi dữ liệu thay đổi, tất cả các client đang kết nối đều được cập nhật ngay lập tức.
- Lưu trữ dạng JSON: Dữ liệu được lưu dưới dạng cây JSON, dễ thao tác và truy xuất.
- Đồng bộ đa nền tảng: Hỗ trợ Android, iOS, Web.
- Làm việc ngoại tuyến: Ứng dụng vẫn hoạt động khi mất kết nối và sẽ đồng bộ lại khi có mạng.
- Bảo mật: Sử dụng quy tắc bảo mật dựa trên người dùng (Firebase Authentication).

b) Cấu trúc dữ liệu trên Firebase

Cấu trúc dữ liệu trên Fỉebase:

Hình 3.1: Cấu trúc dữ liệu trên Firebase

Trong đó, mỗi node history chứa các bản ghi đo có key là timestamp và giá trị gồm:

Hình 3.2: Thông tin trong mỗi node

Bao gồm:

- formatted_time: Định dạng thời gian dưới dạng chuỗi
- name: Tên vị trí cảm biến
- salinity: Độ măn (đơn vị PPT)
- temperature: Nhiệt độ (đơn vị độ C)
- timestamp: Định dạng thời gian dưới dạng số

- Quản lí dữ liệu từ ESP32 gửi lên

- Mô tả ESP32 định dạng dữ liệu và đẩy lên các đường dẫn cụ thể trong Realtime Database.

```
String path = "/He_thong_canh_bao/sensors/sensor1/history/" + timestampStr;
```

Các hàm

- Firebase API được sử dụng (set, push, update)

```
Firebase.setFloat(fbdo, path + "/temperature", temp1);
Firebase.setFloat(fbdo, path + "/salinity", sal1);
Firebase.setString(fbdo, path + "/formatted_time", rtcFormatted);
```

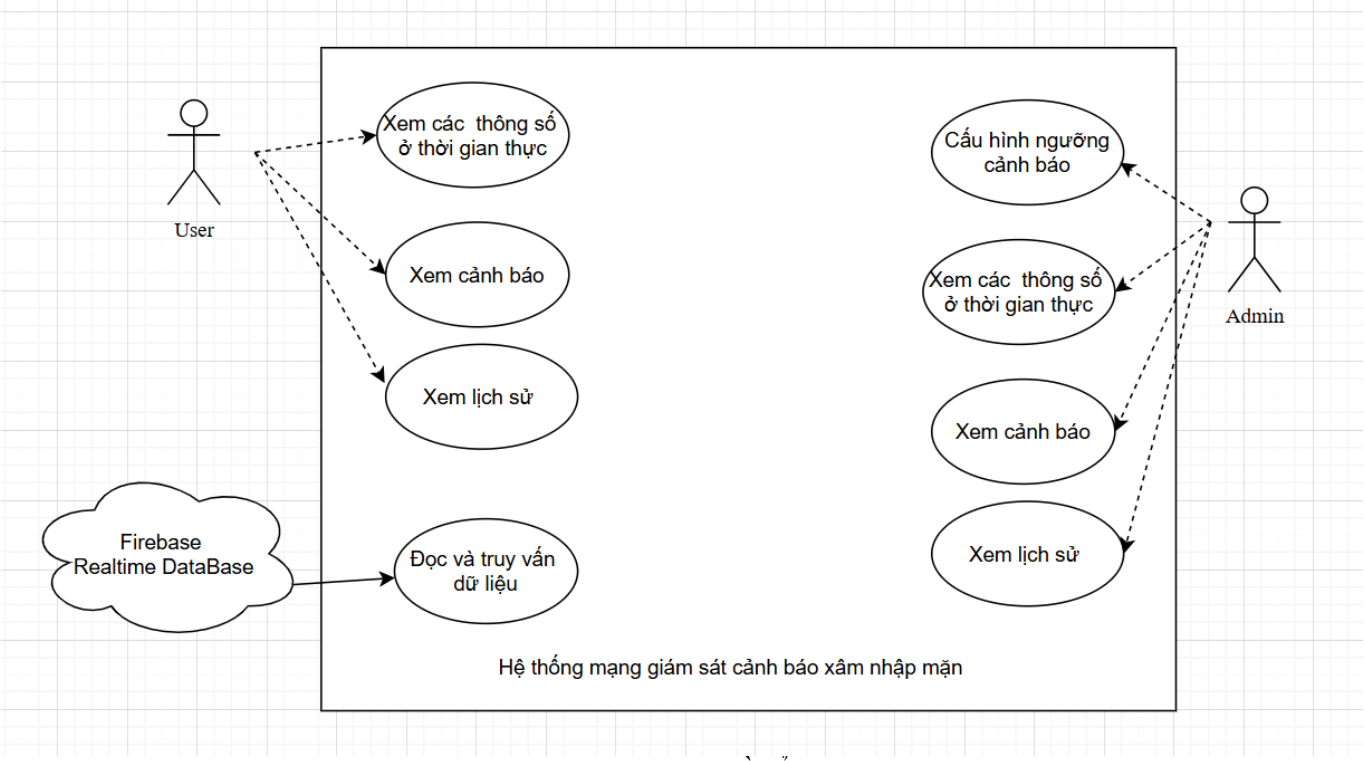
IV. Giao diện Web

1. Mục tiêu

- Thiết kế và triển khai giao diện web giúp hiển thị dữ liệu cảm biến về độ mặn và nhiệt độ theo thời gian thực từ Firebase
- Cung cấp công cụ theo dõi lịch sử và cảnh báo người dùng khi có nguy cơ xâm nhập mặn.
- Giao diện rõ ràng , thận diện , phù hợp với người dùng

Tổng quan Giao diện Web:

Sơ đồ tổng quan



Hình 3.3: Sơ đồ tổng quan Web

2. Các chức năng chính

a) Hiển thị cảm biến trên bản đồ

Giao diện hiển thị các cảm biến (ví dụ: "Cửa Sông", "Ao nuôi", v.v.) được đặt trên tọa độ cụ thể.

Khi click vào cảm biến, popup nhỏ hiện lên hiển thị dữ liệu mới nhất gồm:

- Độ mặn (PPT)
- Nhiệt độ (°C)
- Cảnh báo xâm nhập mặn (mức độ)
- Đường dẫn xem chi tiết

b) Popup chi tiết

Khi chọn "Xem chi tiết", người dùng được chuyển sang popup lớn chứa:

- Dữ liệu của cảm biến
- Biểu đồ biến đổi độ mặn trong thời gian gần nhất (20 lần đo)
- Bảng lịch sử dữ liệu (tìm kiếm theo ngày)

c) Tìm kiếm lịch sử dữ liệu

- Người dùng có thể chọn khoảng ngày để tra cứu lịch sử độ mặn và nhiệt độ.
- Dữ liệu được hiển thị dạng bảng.

3. Công nghệ sử dụng

- **Ngôn ngữ sử dụng :**
  - HTML : Vai trò tạo cấu trúc và nội dung cho trang web

- CSS : Vai trò tạo kiểu dáng, bố cục
- JavaScript: Xử lý toàn bộ logic động, tương tác người dùng, truy vấn dữ liệu từ Firebase, và cập nhật nội dung động

- **Realtime Database:** Firebase Realtime Database
- **Thư viện vẽ biểu đồ:** Chart.js được sử dụng để trực quan hóa dữ liệu độ mặn dưới dạng biểu đồ đường.
- **Nền tảng lưu trữ:** Firebase Hosting

4. Chi tiết triển khai

a) Kết nối và khởi tạo Firebase

Đoạn mã khởi tạo kết nối với Firebase, cho phép web truy cập các dịch vụ Firebase:

```
const firebaseConfig = {
    apiKey: "AIzaSyDnf-WMvqFhibhCYCIAqFRFtb_a5LqxLBg",
    authDomain: "fir-demo-esp32-ec28b.firebaseapp.com",
    databaseURL: "https://fir-demo-esp32-ec28b-default-rtdb.firebaseio.com",
    projectId: "fir-demo-esp32-ec28b",
    storageBucket: "fir-demo-esp32-ec28b.firebasestorage.app",
    messagingSenderId: "469279759359",
    appId: "1:469279759359:web:46ad51afdb3c0682a1e5ca",
    measurementId: "G-D464VST3VS"
};
firebase.initializeApp(firebaseConfig);
const db = firebase.database();
```

- **firebaseConfig**: đối tượng chứa các API và thông tin cần thiết để web kết nối chính xác với Firebase .
- **firebase.initializeApp(firebaseConfig):** Khởi tạo Firebase với cấu hình đã cho
- **const db = firebase.database();** : Lấy tham chiếu đến dịch vụ Realtime Database. Biến 'db' sẽ được sử dụng cho mọi tương tác dữ liệu

b) Hiển thị Thông tin Tóm tắt (Pop-up nhỏ)

Khi người dùng nhấp vào một biểu tượng cảm biến trên bản đồ, một pop-up nhỏ sẽ xuất hiện, hiển thị thông tin cập nhật mới nhất của cảm biến đó

❖ Chức năng handleSensorClick(sensorId, e):

- Kích hoạt khi người dùng click vào biểu tượng cảm biến trên bản đồ
- Tính toán vị trí hiển thị của pop-up nhỏ sao cho nó xuất hiện gần con trỏ chuột và không bị tràn ra khỏi màn hình.
- Sử dụng Firebase **.on('value')** để cập nhật **realtime** dữ liệu mới nhất của cảm

```
function handleSensorClick(sensorId, e) {
    const popup = document.getElementById('popup');
    const popupWidth = 300;
    const popupHeight = 170;

    let x = e.clientX + 10;
    let y = e.clientY + 10;

    if (x + popupWidth > window.innerWidth) x = window.innerWidth - popupWidth - 20;
    if (y + popupHeight > window.innerHeight) y = window.innerHeight - popupHeight - 20;

    popup.style.left = x + 'px';
    popup.style.top = y + 'px';
    popup.style.display = 'block';

    document.getElementById('popup-content').innerHTML = `Đang tải dữ liệu cho <strong>${sensorNameMap[sensorId] || sensorId}</strong>...`;
```

biến.

❖ Định dạng thông tin hiển thị:

- Độ mặn: ${latest.salinity} PPT
- Nhiệt độ: ${latest.temperature} °C
- Cảnh báo: xác định bằng hàm getAlertMessage(salinity)
- Thời gian: định dạng bằng formatTimestampReadable(timestamp)

```
const latestTimestamp = Object.keys(data)[0];
const latest = data[latestTimestamp];
latest.timestamp = latestTimestamp;

// Cập nhật lastSensorData với dữ liệu mới nhất cho popup nhỏ
lastSensorData = {
    ...latest,
    id: sensorId,
    name: sensorNameMap[sensorId] || latest.name || sensorId,
};
const alertMsg = getAlertMessage(latest.salinity);
document.getElementById('popup-content').innerHTML = `
    <strong>${lastSensorData.name}</strong><br>
    Độ mặn: ${latest.salinity} PPT<br>
    Nhiệt độ: ${latest.temperature} °C<br>
    Cảnh báo: ${alertMsg}<br>
    Thời gian: ${formatTimestampReadable(latest.timestamp)}
`;
```

➢ **Hiển thị Thông tin chi tiết (Pop-up lớn)**

Khi người dùng nhấn nút "Xem chi tiết" từ pop-up nhỏ, một pop-up lớn sẽ hiện ra, cung cấp cái nhìn toàn diện về dữ liệu cảm biến, biểu đồ xu hướng độ mặn và bảng lịch sử dữ liệu

❖ **Chức năng openDetail()**

Hàm được kích hoạt khi người dùng muốn xem thông tin chi tiết của một cảm biến

- Mở pop-up lớn chứa thông tin chi tiết, biểu đồ và bảng lịch sử
- Hiển thị biểu đồ độ mặn bằng Chart.js. với 20 bản ghi mới nhất
- Hiển thị bảng lịch sử với chức năng tìm kiếm

Sử dụng .on() trong biểu đồ realtime:

- Dữ liệu luôn được tự động cập nhật ngay lập tức mà không cần người dùng thực hiện bất kỳ thao tác nào
- Quản lý và hủy listener giải phóng tài nguyên mạng và bộ nhớ mà nó đang chiếm dụng, giúp tối ưu hóa hiệu suất và tránh rò rỉ listener

Chuẩn bị dữ liệu cho biểu đồ:

- labels: entries.map(e => formatChartLabel(e[0])) : Chứa các nhãn thời gian cho trục X của biểu đồ, được định dạng bằng formatChartLabel()

- data: entries.map(e => e[1].salinity) : Chứa các giá trị độ mặn tương ứng cho trục Y của biểu đồ

Cấu hình biểu đồ Chart.js:

```
chart = new Chart(ctx, {
    type: 'line',
    data: {
        labels: entries.map(e => formatChartLabel(e[0])), // Lấy labels từ entries mới
        datasets: [{
            label: 'Độ mặn (PPT)',
            data: entries.map(e => e[1].salinity), // Lấy data từ entries mới
            borderColor: 'red',
            fill: false,
            tension: 0.4,
            pointRadius: 2,
            pointHoverRadius: 5
        }]
    },
    options: {
        responsive: false,
        maintainAspectRatio: false,
        plugins: {
            legend: {
                display: true
            },
            tooltip: {
                callbacks: {
                    title: function(context) {
                        return `Thời gian: ${context[0].label.replace('\n', ' ')}`;
                    },
                    label: function(context) {
                        return `Độ mặn: ${context.raw} PPT`;
                    }
                }
            }
```

- Biểu đồ được cấu hình với loại type: 'line' để hiển thị xu hướng. Các tùy chọn (options) được đặt để kiểm soát giao diện và tương tác
  - responsive: false, maintainAspectRatio: false: Tắt tính năng responsive tự động của Chart.js để kiểm soát kích thước bằng CSS/JS
  - plugins.legend: Hiển thị chú giải.
  - plugins.tooltip: Cấu hình tooltip để hiển thị thông tin chi tiết khi di chuột qua các điểm dữ liệu, bao gồm thời gian đầy đủ và giá trị độ mặn

➢ **Chức năng tra cứu lịch sử dữ liệu**

Phần này giúp người dùng xem toàn bộ các bản ghi dữ liệu lịch sử của một cảm biến dưới dạng bảng theo một khoảng thời gian cụ thể

❖ Mô tả chức năng loadAndDisplayHistory(sensorId, startTime, endTime) :

Hàm này có nhiệm vụ tải và hiển thị dữ liệu lịch sử của một cảm biến trong một khoảng thời gian nhất định.

- Nó nhận vào sensorId để xác định cảm biến, startTime và endTime (đối tượng date) để xác định khoảng thời gian.
- Hàm chuyển đổi các đối tượng Date thành timestamp Unix để phù hợp với cấu trúc dữ liệu trong Firebase
- Quản lý Listener: Hủy bỏ currentHistoryTableListener cũ trước khi thiết lập listener mới.
- Sử dụng truy vấn Firebase .on('value', ...) với orderByKey().startAt().endAt() để lấy dữ liệu lịch sử. Nhờ .on(), bảng lịch sử cũng sẽ tự động cập nhật nếu có dữ liệu mới được thêm vào Firebase trong khoảng thời gian đang hiển thị
- Sau khi nhận được dữ liệu, nó sẽ tạo các hàng HTML cho bảng và cập nhật

```
function loadAndDisplayHistory(sensorId, startTime, endTime) {
    const historyDataContainer = document.getElementById('salinity-history-data');
    historyDataContainer.innerHTML = "<p>Đang tải dữ liệu...</p>";
    const startTimestamp = Math.floor(startTime.getTime() / 1000);
    let tempEndDate = new Date(endTime.getTime());
    tempEndDate.setHours(23, 59, 59, 999);
    const endTimestamp = Math.floor(tempEndDate.getTime() / 1000);
    const sensorRef = db.ref('He_thong_canh_bao/sensors/' + sensorId + '/history');
    currentHistoryTableListener = sensorRef.orderByKey().startAt(startTimestamp.toString()).endAt(endTimestamp.toString()).on('value', snapshot =>
        const data = snapshot.val();
        console.log(`Firebase history data for ${sensorId} (range realtime):`, data);
        if (!data || Object.keys(data).length === 0) {
            historyDataContainer.innerHTML = "<p>Không có dữ liệu lịch sử trong khoảng thời gian này.</p>";
            return;
        }
        const entries = Object.entries(data).sort((a, b) => parseInt(a[0]) - parseInt(b[0]));
        let historyHtml = '<table><thead><tr><th>Thời gian</th><th>Độ mặn (PPT)</th><th>Nhiệt độ (°C)</th></tr></thead><tbody>';
        // Hiển thị dữ liệu từ mới nhất đến cũ nhất
        for (let i = entries.length - 1; i >= 0; i--) {
            const entryData = entries[i][1];
            const timestampKey = entries[i][0];
            historyHtml += `
                <tr>
                    <td>${formatTimestampReadable(timestampKey)}</td>
                    <td>${entryData.salinity}</td>
                    <td>${entryData.temperature}</td>
                </tr>
            `;
        }
        historyHtml += '</tbody></table>';
        historyDataContainer.innerHTML = historyHtml;
    }, error => {
        console.error("Lỗi khi tải dữ liệu lịch sử:", error);
        historyDataContainer.innerHTML = "<p>Đã xảy ra lỗi khi tải dữ liệu lịch sử.</p>";
    });
}
```

❖ Chức năng searchHistoryByDate()

- Cho phép người dùng tìm kiếm dữ liệu theo ngày
- Hàm được gọi khi người dùng nhấn nút "Tìm kiếm". Nó đọc giá trị ngày bắt đầu và ngày kết thúc, chuyển đổi sang đối tượng Date, kiểm tra tính hợp lệ của ngày, sau đó gọi loadAndDisplayHistory() với các ngày đã chọn để cập nhật bảng..

```
function searchHistoryByDate() {
    if (!currentDetailedSensorId) {
        alert("Vui lòng chọn một cảm biến trước.");
        return;
    }
    const startDateInput = document.getElementById('startDate');
    const endDateInput = document.getElementById('endDate');
    let startDate = new Date(startDateInput.value);
    let endDate = new Date(endDateInput.value);
    endDate.setHours(23, 59, 59, 999);
    if (isNaN(startDate.getTime()) || isNaN(endDate.getTime())) {
        alert("Vui lòng nhập ngày bắt đầu và ngày kết thúc hợp lệ.");
        return;
    }
    if (startDate.getTime() > endDate.getTime()) {
        alert("Ngày bắt đầu không thể lớn hơn ngày kết thúc.");
        return;
    }
    loadAndDisplayHistory(currentDetailedSensorId, startDate, endDate);
}
```

➢ **Các hàm khác**

- sensorNameMap: Một đối tượng JavaScript giúp ánh xạ ID cảm biến kỹ thuật sang tên gọi thân thiện với người dung
- **getAlertMessage(salinity)**: Chuyển đổi giá trị độ mặn thành thông báo cảnh báo (Bình thường, Cảnh báo trung bình, Cảnh báo cao) dựa trên các ngưỡng định trước
- **formatTimestampReadable(ts)**: Chuyển đổi timestamp Unix từ Firebase thành định dạng ngày giờ dễ đọc cho người dung
- **formatChartLabel(ts)**: Định dạng timestamp thành nhãn phù hợp cho trục X của biểu đồ
- **clearDetailView()**: Hàm dùng để làm sạch nội dung của pop-up chi tiết, bao gồm hủy đối tượng chart của Chart.js để chuẩn bị cho việc vẽ biểu đồ mới

d) Kết quả giao diện

Tổng quan :

Hình 3.4: Bản đồ cảm biến

Khi click vào 1 cảm biến

Hình 3.5: Thông tin trong Popup nhỏ

Khi click vào xem chi tiết

Hình 3.6: Thông tin chi tiết

e) Tiến hành deloy lên Firebase Hosting

Để người dùng có thể theo dõi hệ thống bất cứ nơi đâu, nhóm tiến hành Deloy lên Firebase Hosting để đưa ứng dụng web từ máy tính cục bộ lên môi trường trực tuyến giúp mọi người dễ dàng truy cập

Firebase Hosting là một dịch vụ lưu trữ nội dung web tĩnh và động nhanh chóng, an toàn và đáng tin cậy, được tối ưu hóa cho các ứng dụng web và giao diện người dùng. Việc triển khai giao diện web lên Firebase Hosting chủ yếu thông qua **Firebase Command Line Interface (CLI)**.

➢ **Các bước chi tiết để triển khai**

- Cài đặt Node.js và npm (nếu chưa có):
- Cài đặt Firebase CLI:

Mở terminal hoặc Command Prompt và chạy lệnh sau để cài đặt Firebase CLI một cách toàn cục (global) trên hệ thống npm install -g firebase-tools

Lệnh này sẽ tải xuống và cài đặt gói firebase-tools, cung cấp các lệnh cần thiết để tương tác với Firebase từ dòng lệnh

- Đăng nhập vào Firebase:

Sau khi cài đặt CLI, đăng nhập vào tài khoản Google/Firebase. Chạy lệnh firebase login

Lệnh này sẽ mở một cửa sổ trình duyệt, yêu cầu xác thực tài khoản Google của mình và cấp quyền cho Firebase CLI. Sau khi đăng nhập thành công, bạn sẽ thấy thông báo xác nhận trong terminal.

- Khởi tạo dự án Firebase trong thư mục web:

Điều hướng đến thư mục gốc của dự án giao diện web của bạn (nơi chứa các tệp index.html, script.js, style.css và bất kỳ tài nguyên web nào khác). Sau đó, chạy lệnh: firebase init

- Chọn dịch vụ Hosting.
- Chọn project Firebase đã tạo (ví dụ: fir-demo-esp32-ec28b).
- Chọn thư mục public là public/ (hoặc nơi chứa file HTML).
- Chọn **Không ghi đè index.html nếu đã có**

- Triển khai lên Firebase Hosting

Sau khi cấu hình xong, triển khai với lệnh:

firebase deploy

Firebase tạo liên kết dạng:

```
C:\Users\QUANG\Desktop\DA1>firebase deploy

=== Deploying to 'web-doman'...

i  deploying hosting
i  hosting[web-doman]: beginning deploy...
i  hosting[web-doman]: found 5 files in public
+  hosting[web-doman]: file upload complete
i  hosting[web-doman]: finalizing version...
+  hosting[web-doman]: version finalized
i  hosting[web-doman]: releasing new version...
+  hosting[web-doman]: release complete

+  Deploy complete!

Project Console: https://console.firebase.google.com/project/web-doman/overview
Hosting URL: https://web-doman.web.app
```

Link web : *https://web-doman.web.app*

Truy cập vào đường link ta sẽ thấy được giao diện web

Code Nguồn :

https://drive.google.com/drive/folders/1xbdw3KpHf1wsjh43NrgirNTZ50NPVTpK?usp=drive_link

V. Kết Luận

Hệ thống cảm biến EC đã phản hồi được dữ liệu nhanh chóng , các khối đã đo được output đúng lý thuyết . Tuy nhiên ADC của vi điều khiển vẫn chưa đọc được dữ liệu ổn định cho nên ta cần làm ổn định tín hiệu .

Giao diện web đã hiển thị dữ liệu cảm biến ngay lập tức theo thời gian thực , giúp người dùng có thể giám sát liên tục. Tuy nhiên giao diện còn thiếu phần xác thực, phân quyền , không có chức năng đăng nhập , cho phép bất cứ ai có đường link đều có thể xem dữ liệu, tiềm ẩn rủi ro bảo mật. Do vậy cần cải tiến thêm chức năng đăng nhập người dùng

Từ quá trình triển khai và sử dụng Firebase trong hệ thống, có thể nhận thấy nền tảng này phù hợp với các ứng dụng IoT nhờ khả năng truyền dữ liệu thời gian thực, dễ tích hợp với ESP32 và hỗ trợ lưu trữ đám mây tiện lợi. Gói miễn phí cũng giúp tiết kiệm chi phí trong giai đoạn thử nghiệm. Tuy nhiên, Firebase vẫn tồn tại một số hạn chế như yêu cầu kết nối Internet liên tục, giới hạn trong gói miễn phí, không phù hợp với dữ liệu quan hệ và tiềm ẩn rủi ro bảo mật nếu không cấu hình đúng cách. Do đó, việc lựa chọn Firebase cần được cân nhắc kỹ tùy theo quy mô và yêu cầu bảo mật của hệ thống.

Tài liệu tham khảo :

1. ESP32: Guide for DS3231 Real Time Clock Module (RTC)

https://randomnerdtutorials.com/esp32-ds3231-real-time-clock-arduino/

2. MicroSD Card Interfacing with ESP32

https://www.electronicwings.com/esp32/microsd-card-interfacing-with-esp32

3. Firebase Realtime Database

https://firebase.google.com/docs/database

4. Read and Write Data on the Web

https://firebase.google.com/docs/database/web/read-and-write

5. Cảm biến EC

https://thecavepearlproject.org/2017/08/12/measuring-electrical-conductivity-with-an-arduino-part1-overview/

lgt8fx/docs/LGT8FX8P_databook_v1.0.4.en.pdf at master · dbuezas/lgt8fx · GitHub

Operational Amplifier Basics - Op-amp tutorial

6. Giao tiếp web với Firebase

https://www.youtube.com/watch?v=r1KlhCFfatM

7. Deloy web lên Firebase Hosting

https://blog.pirago.vn/deploy-web-voi-firebase/