|  |  |  |
| --- | --- | --- |
|  | VIỆN ĐÀO TẠO QUỐC TẾ FPT  THIẾT KẾ VI MẠCH |  |

**Tài liệu Thiết kế - Học kỳ II**

**COLD STORAGE MANAGEMENT**

|  |  |  |  |
| --- | --- | --- | --- |
| Giáo viên hướng dẫn: | **Trương Huy Hoàng** | | |
| Lớp: | **C1.2405.E0** | | |
| Thành viên trong nhóm: | Tên đầy đủ | | Mã số Sinh viên |
|  | 1 | **Nguyễn Hoàng Tâm** | JK-ENR-HA-11592 |
|  | 2 | **Hoàng Gia Bảo** | JK-ENR-HA-115 |
|  | 3 | **Nguyễn Đắc Hào** | JK-ENR-HA-115 |
|  | 4 | **Nguyễn Huy** | JK-ENR-HA-11555 |
|  | 5 | **Nguyễn Anh Quân** | JK-ENR-HA-11553 |

Tp.HCM, ngày 23 tháng 03 năm 2025

**LỜI CẢM ƠN**

Trong suốt học kì 2 tại trường FPT-JETKING. Nhóm đồ án được tiếp thu một cách có hệ thống các kiến thức khoa học tiên tiến hiện đại của ngành Chip-Design. Kết thúc học kỳ nhóm đồ án thực hiện đề tài “ **COLD STORAGE MANAGEMENT**"

Với lòng biết ơn sâu sắc nhất, nhóm đồ án xin gửi đến quý thầy đã truyền đạt vốn kiến thức quý báu cho nhóm đồ án trong suốt thời gian học tập tại trường. Nhờ có những lời hướng dẫn, dạy bảo của các thầy nên đề tài nghiên cứu của nhóm mới có thể hoàn thiện tốt đẹp.

Trong quá trình thực hiện nhóm luôn cố gắng để hoàn thiện dự án 1 cách tốt nhất nhưng có thể không tránh khỏi những sai sót. Nhóm đồ án rất mong nhận được những ý kiến đóng góp quý báu của quý thầy cô để kiến thức của nhóm trong lĩnh vực này được hoàn thiện hơn đồng thời có điều kiện bổ sung, nâng cao ý thức của nhóm đồ án .

**I. TỔNG QUAN DỰ ÁN: HỆ THỐNG QUẢN LÝ KHO LẠNH (COLD STORAGE MANAGEMENT**

## **1.1 Giới thiệu**

Trong nhiều lĩnh vực như thực phẩm, dược phẩm, y tế, và hóa chất, nhiệt độ và độ ẩm là những yếu tố quan trọng quyết định đến chất lượng và độ an toàn của sản phẩm. Kho lạnh đóng vai trò quan trọng trong việc bảo quản hàng hóa, nhưng nếu không được giám sát chặt chẽ, nhiệt độ không ổn định có thể gây hư hỏng sản phẩm, tổn thất kinh tế lớn và thậm chí ảnh hưởng đến sức khỏe con người.

Hiện nay, nhiều doanh nghiệp vẫn sử dụng phương pháp giám sát thủ công hoặc các hệ thống giám sát truyền thống thiếu tính linh hoạt và khả năng kết nối từ xa. Điều này gây ra nhiều bất cập như:

* Không phát hiện kịp thời khi nhiệt độ vượt ngưỡng cho phép.
* Khó lưu trữ và phân tích dữ liệu dài hạn để tối ưu hóa bảo quản.
* Tốn nhân lực để kiểm tra định kỳ thay vì tự động hóa.

## **1.2 Phạm vi và giới hạn của chủ đề.**

### **a, Phạm vi.**

Hệ thống Cold Storage Management tập trung vào giám sát và quản lý môi trường bảo quản trong kho lạnh. Phạm vi ứng dụng bao gồm:

* Giám sát nhiệt độ và độ ẩm trong các kho lạnh, xe đông lạnh hoặc khu vực bảo quản sản phẩm nhạy cảm với nhiệt độ.
* Cảnh báo và thông báo khi các thông số môi trường vượt ngưỡng quy định.
* Lưu trữ và phân tích dữ liệu để tối ưu hóa quy trình bảo quản.
* Giao tiếp IoT qua Wi-Fi để truyền dữ liệu lên nền tảng đám mây (ThingsBoard Cloud).

Hệ thống này không bao gồm:  
 ❌ Điều khiển thiết bị làm lạnh (ví dụ: bật/tắt máy nén lạnh).  
 ❌ Giám sát các thông số khác ngoài nhiệt độ và độ ẩm (như mức CO₂, oxy, áp suất).  
 ❌ Tích hợp AI hoặc học máy để dự đoán lỗi thiết bị làm lạnh.

### **b, giới hạn chức năng hoạt động**

✔ Thu thập dữ liệu nhiệt độ, độ ẩm từ cảm biến DHT11.  
 ✔ Truyền dữ liệu từ ESP8266 lên ThingsBoard Cloud.  
 ✔ Hiển thị dữ liệu trên giao diện ThingsBoard.  
 ✔ Hiển thị thông tin nhiệt độ, độ ẩm lên màn hình LCD 16x2.  
 ✔ Gửi cảnh báo qua email/SMS khi nhiệt độ vượt ngưỡng cài đặt.  
 ✔ Lưu trữ dữ liệu để phân tích xu hướng nhiệt độ theo thời gian.

Hệ thống có những giới hạn:  
 ⚠ Độ chính xác của cảm biến DHT11: ±2°C cho nhiệt độ và ±5% cho độ ẩm, không phù hợp cho yêu cầu giám sát có độ chính xác cao.  
 ⚠ Phụ thuộc vào kết nối Wi-Fi: Nếu mạng không ổn định, dữ liệu có thể bị gián đoạn.  
 ⚠ Không hỗ trợ kiểm soát chủ động: Hệ thống chỉ giám sát và cảnh báo, không điều khiển hệ thống làm lạnh.  
 ⚠ Dung lượng lưu trữ hạn chế: Dữ liệu phụ thuộc vào khả năng lưu trữ của ThingsBoard Cloud và phải có chiến lược xoá/lưu trữ dài hạn.

## **1.3 CHỨC NĂNG VÀ HOẠT ĐỘNG CỦA HỆ THỐNG COLD STORAGE MANAGEMENT**

**Chức năng chính của hệ thống**

✅ Giám sát thời gian thực: Đo nhiệt độ và độ ẩm từ DHT11.  
 ✅ Hiển thị dữ liệu: Hiển thị nhiệt độ và độ ẩm lên LCD 16x2 và giao diện ThingsBoard.  
 ✅ Cảnh báo vượt ngưỡng: Gửi thông báo khi điều kiện môi trường không đảm bảo.  
 ✅ Truy cập từ xa: Xem dữ liệu trên giao diện ThingsBoard Cloud qua Wi-Fi.  
 ✅ Lưu trữ và phân tích dữ liệu: Giúp theo dõi xu hướng nhiệt độ theo thời gian.

**Cơ chế hoạt động**

🔹 Bước 1: Cảm biến DHT11 đo nhiệt độ và độ ẩm.  
 🔹 Bước 2: ZUBoard 1CG xử lý dữ liệu và kiểm tra giá trị nhiệt độ, độ ẩm.  
 🔹 Bước 3: Hiển thị nhiệt độ và độ ẩm lên LCD 16x2 để quan sát tại chỗ.  
 🔹 Bước 4: ESP8266 truyền dữ liệu lên ThingsBoard Cloud.  
 🔹 Bước 5: Người dùng theo dõi dữ liệu trên giao diện trực tuyến.  
 🔹 Bước 6: Nếu nhiệt độ vượt ngưỡng, hệ thống gửi cảnh báo ngay lập tức.

# **II, Giới thiệu thiết bị và phần mềm**

## **2.1 Thành phần thiết bị phần cứng**

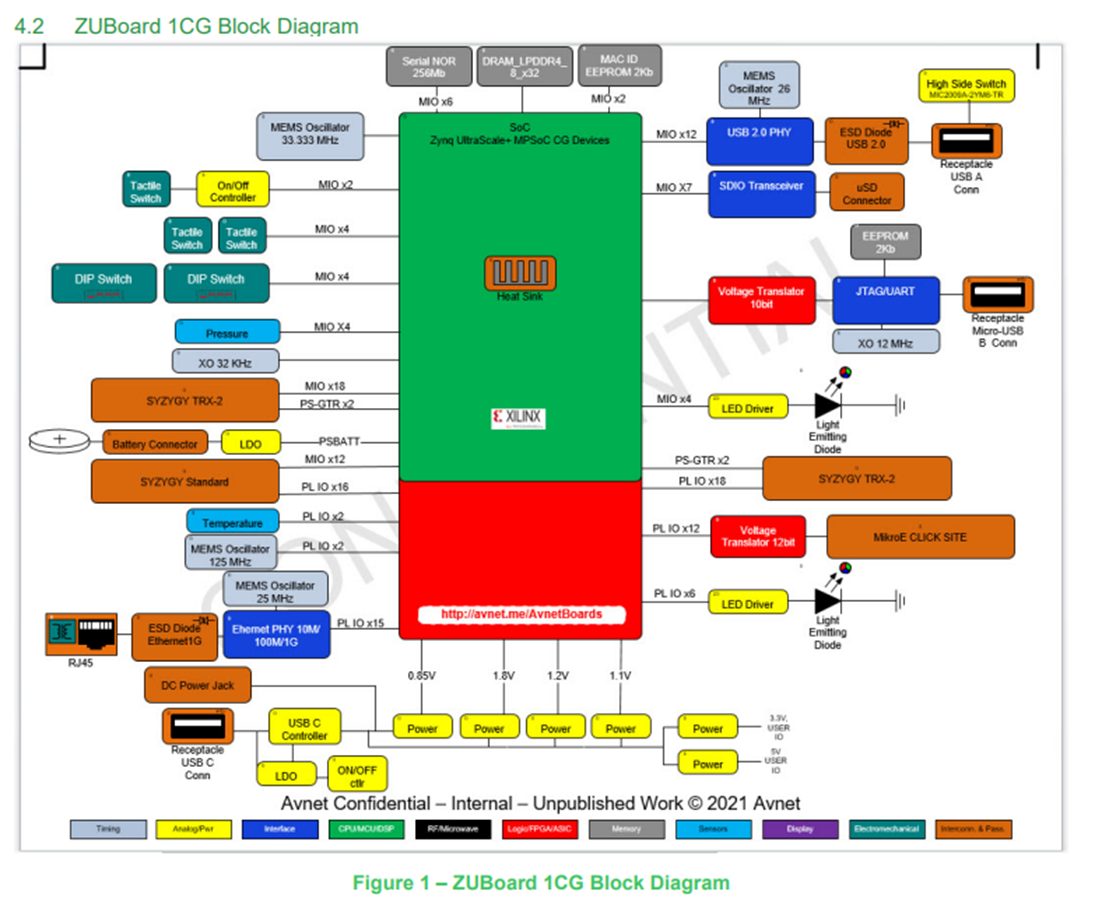
**Phần cứng:**

* ZUBoard 1CG: Xử lý và thu thập dữ liệu từ cảm biến.
* ESP8266/ESP32: Kết nối Wi-Fi và gửi dữ liệu lên đám mây.
* Cảm biến DHT11: Đo nhiệt độ đọ ẩm
* Màn hình LCD16x2: Hiển thị thông tin trực tiếp.
* Pin dự phòng: Duy trì năng lượng khi mất điện.

**Phần mềm:**

* Hệ điều hành: Vivado trên ZUBoard 1CG.
* Lập trình FPGA: Vivado.
* Giao thức truyền dữ liệu: I2C và Uart
* Cloud: ThingsBoard Cloud để quản lý dữ liệu và hiển thị dashboard.

### 2.1.1 Zuboard 1 CG

- ZUBoard 1CG là một bo mạch phát triển được thiết kế để cung cấp một nền tảng mạnh mẽ nhưng có chi phí hợp lý cho các kỹ sư nhúng, nhà phát triển FPGA, sinh viên và người đam mê công nghệ. Bo mạch này sử dụng AMD (Xilinx) Zynq UltraScale+ MPSoC XCZU1CG, kết hợp giữa vi xử lý ARM và FPGA trên một chip duy nhất. Điều này giúp ZUBoard 1CG có thể xử lý các ứng dụng nhúng phức tạp, từ AI, IoT, xử lý hình ảnh, video, đến mạng và bảo mật.

**Cấu tạo chi tiết của ZUBoard 1CG**

ZUBoard 1CG là một bo mạch phát triển nhúng của Avnet, được thiết kế với AMD Xilinx Zynq UltraScale+ MPSoC ZU1CG. Đây là một nền tảng mạnh mẽ dành cho các ứng dụng IoT, AI tại biên (Edge AI), xử lý tín hiệu và điều khiển thời gian thực.

**Dưới đây là phân tích chi tiết từng thành phần của ZUBoard 1CG:**

**1. Bộ xử lý trung tâm (SoC – System on Chip)**

Xilinx Zynq UltraScale+ ZU1CG MPSoC

* Hệ thống đa xử lý kết hợp:
  + 2 nhân ARM Cortex-A53 (64-bit, lên đến 1.3GHz)  
     ➝ Dùng để chạy hệ điều hành như PetaLinux, Ubuntu và xử lý các tác vụ phức tạp.
  + 1 nhân ARM Cortex-R5F (32-bit, chạy ở 500MHz)  
     ➝ Chuyên xử lý real-time, phù hợp cho các tác vụ điều khiển nhúng.
  + FPGA Programmable Logic (PL)  
     ➝ Có thể lập trình để tăng tốc xử lý tín hiệu, AI, điều khiển.
  + Bộ xử lý đồ họa Mali-400MP2  
     ➝ Hỗ trợ hiển thị đồ họa đơn giản trên màn hình.

Điểm mạnh của SoC ZU1CG  
 Sự kết hợp giữa CPU, MCU và FPGA trên cùng một chip giúp tối ưu hiệu suất.  
 Hỗ trợ Linux, FreeRTOS và Bare-metal cho ứng dụng nhúng và IoT.

**2. Bộ nhớ và lưu trữ**

RAM: 1GB DDR4 2400 MT/s (hỗ trợ truyền dữ liệu nhanh).  
 Bộ nhớ Flash: Quad-SPI NOR Flash 128MB để lưu trữ bootloader và firmware.  
 Khe cắm thẻ nhớ microSD:

* Hỗ trợ thẻ SD tối đa 32GB, dùng để cài hệ điều hành như PetaLinux, Ubuntu.

**3. Cổng kết nối và giao tiếp ngoại vi**

**A. Giao tiếp có dây**

USB:

* 1 cổng USB 3.0 Type-C (hỗ trợ cấp nguồn, dữ liệu, UART console).
* 2 cổng USB 2.0 Type-A để kết nối chuột, bàn phím, ổ đĩa USB.

Ethernet:

* 1 cổng Gigabit Ethernet (10/100/1000 Mbps) hỗ trợ giao tiếp mạng có dây tốc độ cao.

HDMI:

* 1 cổng HDMI TX (hỗ trợ độ phân giải Full HD).

GPIO (40-pin, tương thích Raspberry Pi):

* Có thể kết nối cảm biến, relay, mô-đun IoT như ESP8266, ESP32.

Camera Interface (MIPI CSI-2):

* 2x MIPI CSI-2 (hỗ trợ kết nối camera cho AI, xử lý ảnh).

**B. Giao tiếp không dây (cần mô-đun mở rộng)**

Wi-Fi & Bluetooth:

* Không tích hợp sẵn nhưng có thể mở rộng qua mô-đun ESP8266, ESP32 hoặc USB Wi-Fi.

**C. Các giao tiếp khác**

* I2C (Inter-Integrated Circuit): Dùng để kết nối cảm biến như MAX30102 (đo nhịp tim, SpO2).
* SPI (Serial Peripheral Interface): Kết nối với màn hình OLED, LCD.
* UART (Universal Asynchronous Receiver-Transmitter): Giao tiếp với vi điều khiển hoặc máy tính.
* CAN Bus: Hỗ trợ truyền dữ liệu trong hệ thống nhúng và ô tô.

**4. Hệ thống cấp nguồn**

Cổng cấp nguồn chính:

* USB Type-C (5V, 3A) – cấp nguồn chính cho mạch.
* Cổng Barrel Jack 12V (tùy chọn) – dùng khi cần công suất cao hơn.

IC quản lý nguồn (PMIC) của Renesas giúp điều phối điện năng hiệu quả.

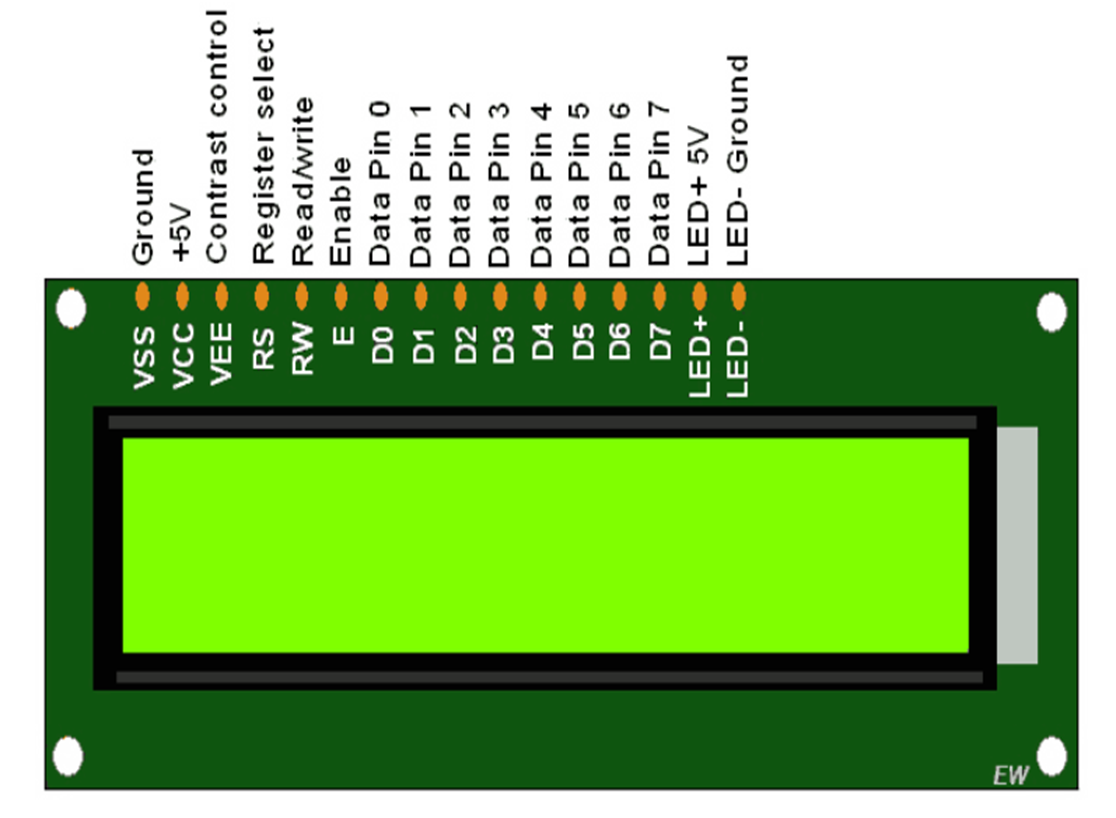
5. Hệ thống phần mềm

* Hệ điều hành hỗ trợ:  
   PetaLinux (dựa trên Yocto)  
   Ubuntu 20.04 LTS (chạy trên ARM Cortex-A53)  
   FreeRTOS (chạy trên Cortex-R5F)  
   Bare-metal (không hệ điều hành, tối ưu cho real-time)
* Công cụ lập trình:
  + Xilinx Vitis/Vivado: Lập trình FPGA và phát triển phần mềm.
  + PetaLinux SDK: Xây dựng hệ điều hành tùy chỉnh.
  + Python, C, C++: Viết phần mềm nhúng.

6. Ứng dụng thực tế của ZUBoard 1CG

IoT và hệ thống nhúng  
 Xử lý tín hiệu thời gian thực (như COLD STORAGE MANAGEMET)  
 AI tại biên (Edge AI)  
 Điều khiển robot và tự động hóa công nghiệp  
 Ứng dụng y tế thông minh (Smart Healthcare)

### **2.1.2 LCD16x2**



**Giới thiệu:**

Mô-đun LCD 16 × 2 là loại mô-đun LCD rất phổ biến được sử dụng trong các dự án nhúng dựa trên 8051. Nó bao gồm 16 hàng và 2 cột 5 × 7 hoặc 5 × 8 ma trận điểm LCD. Các mô-đun đang nói về ở đây là loại JHD162A, một loại rất phổ biến. Nó có sẵn trong một gói 16 chân với ánh sáng nền , chức năng điều chỉnh độ tương phản và mỗi ma trận điểm có độ phân giải 5 × 8 chấm. Số chân, tên của chúng và các chức năng tương ứng được hiển thị trong bảng bên dưới.

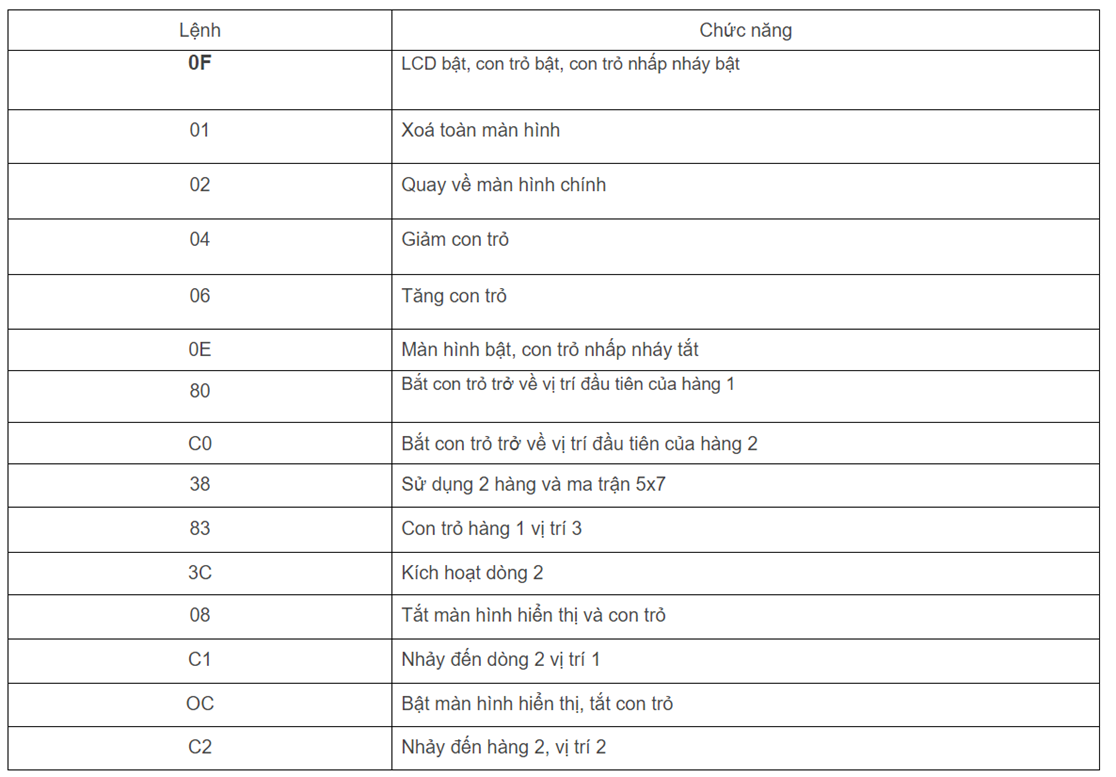


#### Hình 2.7: Tên và chức năng từng chân LCD

- Chân Vee: dùng để điều chỉnh độ tương phản của màn hình LCD và độ tương phản có thể được điều chỉnh sbằng cách thay đổi điện áp ở chân này.

Các lệnh cho mô-đun LCD 16x2:

Mô-đun LCD 16 × 2 có một bộ hướng dẫn lệnh đặt trước. Mỗi lệnh sẽ làm cho mô-đun thực hiện một nhiệm vụ cụ thể. Các lệnh thường được sử dụng và chức năng của chúng được đưa ra trong bảng dưới đây



#### Hình 2.9: Các lệnh và chức năng thường được sử dụng

Khởi tạo LCD:

Các bước phải được thực hiện để khởi tạo màn hình LCD được đưa ra dưới đây và các bước này là phổ biến cho hầu hết các ứng dụng.

B1: Gửi 38H đến dòng dữ liệu 8 bit để khởi tạo

B2: Gửi 0FH để bật LCD, con trỏ BẬT và con trỏ nhấp nháy ON.

B3:Gửi 06H để tăng vị trí con trỏ.

B4: Gửi 01H để xóa màn hình và trả về con trỏ.

Đưa dữ liệu vào LCD:

Các bước để gửi dữ liệu đến mô-đun LCD được đưa ra dưới đây. Mô-đun LCD có các chân RS, R / W và E. Chính trạng thái logic của các chân này làm cho mô-đun xác định xem đầu vào dữ liệu đã cho là lệnh hay dữ liệu được hiển thị.

Đặt R / W mức thấp.

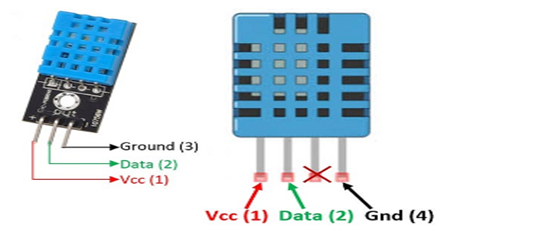
Đặt RS = 0 nếu byte dữ liệu là lệnh và tạo RS = 1 nếu byte dữ liệu là dữ liệu sẽ được hiển thị.

Đặt byte dữ liệu trên thanh ghi dữ liệu.

Xung E từ cao xuống thấp.

Lặp lại các bước trên để gửi dữ liệu khác.

### 2.1.3 DHT11



- DHT11 là một cảm biến nhiệt độ và độ ẩm thường được sử dụng đi kèm với một NTC chuyên dụng để đo nhiệt độ và một bộ vi điều khiển 8 bit để xuất ra các giá trị nhiệt độ và độ ẩm dưới dạng dữ liệu nối tiếp.

- Cảm biến gửi dữ liệu đo được qua giao thức 1-wire.

- Ghi lại giá trị nhiệt độ (°C) và độ ẩm (%).

Hình 2.4 Cảm biến DHT11

Thông số kỹ thuật:

- Dải đo độ ẩm: 20% - 80% RH (±5%)

- Dải đo nhiệt độ: 0°C - 50°C (±2°C)

- Giao tiếp: 1-wire

- Nguồn cung cấp: 3.3V - 5V

|  |  |  |
| --- | --- | --- |
| Số chân | Tên chân | Mô tả |
| 1 | Vcc | Nguồn 3.5V đến 5.5V |
| 2 | Data | Đầu ra cả nhiệt độ và độ ẩm thông qua dữ liệu nối tiếp |
| 3 | NC | Không có kết nối và do đó không sử dụng |
| 4 | Ground | Nối đất |

## 2.2 Phần mềm và công cụ

### 2.2.1. Vivado

Giới thiệu

Vivado Design Suite là bộ công cụ thiết kế FPGA và SoC (System on Chip) do Xilinx (nay thuộc AMD) phát triển. Đây là phần mềm thay thế công cụ ISE cũ, tối ưu hóa cho các dòng thiết bị UltraScale, Zynq, Kintex, Virtex, Artix, và Versal.

1. Điểm nổi bật của Vivado

* Hiệu suất cao: Tích hợp công cụ tổng hợp, ánh xạ và định tuyến tối ưu hơn ISE.
* Hỗ trợ SoC & HLS: Cung cấp IP Integrator để thiết kế sơ đồ khối và hỗ trợ lập trình phần cứng bằng HLS (High-Level Synthesis) từ C/C++.
* Mô phỏng và gỡ lỗi mạnh mẽ: Hỗ trợ Vivado Simulator, tích hợp công cụ mô phỏng bên ngoài như ModelSim.

2. Các thành phần chính của Vivado

A. Vivado IDE

Vivado IDE cung cấp một môi trường trực quan để thiết kế FPGA bằng Verilog, VHDL hoặc sơ đồ khối (Block Design).

* Schematic Editor: Xem và chỉnh sửa mạch điện tử.
* Constraint Editor: Cấu hình ràng buộc về thời gian và vị trí linh kiện.
* IP Catalog: Kho thư viện IP Core như UART, SPI, I2C, DDR, AXI...
* Simulation & Debugging: Hỗ trợ mô phỏng bằng Vivado Simulator hoặc tích hợp ModelSim.

B. Công cụ tổng hợp (Synthesis & Implementation)

🔹 Synthesis: Chuyển mã HDL thành cổng logic.  
 🔹 Implementation: Định vị (placement) và kết nối (routing) tài nguyên trên FPGA.  
 🔹 Timing Analysis: Kiểm tra thời gian trễ, tối ưu đường dữ liệu.  
 🔹 Bitstream Generation: Tạo file .bit để nạp lên FPGA.

C. IP Integrator (Thiết kế sơ đồ khối)

* Kéo-thả để kết nối các mô-đun phần cứng.
* Hỗ trợ Zynq Processing System (PS) + PL (FPGA Logic).
* Tích hợp nhanh các giao diện AXI, DDR, UART, Ethernet...

D. Vivado HLS (High-Level Synthesis)

* Chuyển đổi code C/C++ thành RTL để tổng hợp phần cứng.
* Tối ưu hóa thuật toán DSP, AI, và xử lý tín hiệu số.
* Hỗ trợ OpenCL để tăng tốc trên FPGA.

E. Hardware Manager

* Dùng để nạp bitstream và debug phần cứng.
* Hỗ trợ Integrated Logic Analyzer (ILA) để kiểm tra tín hiệu thời gian thực.
* Tương thích với JTAG, Serial Console để kiểm tra hoạt động hệ thống.

3. Ứng dụng của Vivado trong Dự Án Cold Storage Management

Dự án Cold Storage Management sử dụng Vivado để lập trình và cấu hình ZUBoard 1CG, giúp xử lý dữ liệu từ cảm biến và truyền lên ThingsBoard Cloud.

Quy trình thiết kế với Vivado:

1. Viết mã RTL (Verilog/VHDL) hoặc thiết kế Block Design để đọc dữ liệu từ DHT11.

2. Tích hợp giao tiếp với ESP8266 qua UART để truyền dữ liệu lên ThingsBoard Cloud.

3. Hiển thị dữ liệu nhiệt độ và độ ẩm trên LCD 16x2.

4. Cấu hình hệ thống cảnh báo khi nhiệt độ vượt ngưỡng, gửi thông báo tới người dùng.

### 2.2.2. Ardunio

Giới thiệu:

Phần mềm Arduino IDE (Integrated Development Environment) là một công cụ phát triển chính thức của nền tảng Arduino, cho phép người dùng viết, chỉnh sửa và tải mã nguồn lên các bảng mạch Arduino. Đây là một môi trường lập trình đơn giản và dễ sử dụng, phù hợp với những người mới làm quen với lập trình nhúng.

Điểm nổi bật của Ardunio:

- Giao diện đơn giản: Arduino IDE có giao diện trực quan và thân thiện với người dùng, với các công cụ cơ bản để viết mã, biên dịch và tải lên bảng Arduino.

- Ngôn ngữ lập trình: Arduino IDE sử dụng một ngôn ngữ lập trình dựa trên C/C++, được đơn giản hóa để dễ sử dụng hơn. Người dùng có thể lập trình để điều khiển các đầu vào/ra của bảng mạch Arduino như đèn LED, cảm biến, động cơ, v.v.

- Thư viện phong phú: Arduino IDE đi kèm với rất nhiều thư viện hỗ trợ, giúp bạn dễ dàng giao tiếp với các cảm biến, module, và thiết bị ngoại vi khác. Thư viện giúp tiết kiệm thời gian cho người dùng khi không phải tự viết mã từ đầu cho từng phần cứng.

- Cross-platform: Arduino IDE có thể hoạt động trên nhiều hệ điều hành khác nhau như Windows, macOS, và Linux.

- Tích hợp sẵn công cụ biên dịch và tải mã: Sau khi mã nguồn được viết xong, IDE sẽ biên dịch (compile) mã thành ngôn ngữ máy và tải lên bảng Arduino thông qua cổng USB.

- Mã nguồn mở và cộng đồng lớn: Arduino là mã nguồn mở, nghĩa là mọi người có thể đóng góp và cải tiến phần mềm. Cộng đồng Arduino rất lớn mạnh, luôn chia sẻ các dự án, ví dụ và hỗ trợ lẫn nhau.

Ứng dụng:

- Robot học: Sử dụng Arduino để điều khiển động cơ và cảm biến, xây dựng robot tự hành hoặc robot điều khiển từ xa.

- Nhà thông minh (Home Automation): Điều khiển đèn, thiết bị điện tử từ xa qua internet.

- IoT (Internet of Things): Kết nối Arduino với các hệ thống IoT để thu thập và chia sẻ dữ liệu trực tuyến.

- Giáo dục: Arduino là công cụ hữu ích trong giảng dạy về lập trình và điện tử, giúp học sinh và sinh viên dễ dàng hiểu được nguyên lý của vi điều khiển.

### 2.2.3 Thingboard

Giới thiệu:

ThingBoard là một nền tảng mã nguồn mở mạnh mẽ, được sử dụng chủ yếu trong lĩnh vực IoT (Internet of Things), cho phép thu thập, xử lý, và hiển thị dữ liệu từ các thiết bị kết nối. ThingBoard giúp các doanh nghiệp và nhà phát triển dễ dàng triển khai hệ thống IoT và giám sát dữ liệu theo thời gian thực. Với khả năng tùy biến cao và hỗ trợ nhiều giao thức, ThingBoard trở thành công cụ đắc lực trong việc quản lý thiết bị và phân tích dữ liệu IoT.

Đặc điểm nổi bật:

Mô hình dữ liệu thời gian thực: ThingBoard cho phép hiển thị và phân tích dữ liệu từ các thiết bị IoT trong thời gian thực. Người dùng có thể giám sát các thông số như nhiệt độ, độ ẩm, mức tiêu thụ năng lượng,... từ xa thông qua bảng điều khiển (dashboard) trực quan.

- Hỗ trợ giao thức đa dạng:

ThingBoard hỗ trợ nhiều giao thức kết nối như MQTT, CoAP, HTTP, cho phép tương tác và trao đổi dữ liệu giữa các thiết bị IoT một cách linh hoạt.

- Quản lý thiết bị:

ThingBoard cung cấp các tính năng mạnh mẽ để quản lý các thiết bị IoT. Người dùng có thể thêm, xóa, và giám sát trạng thái của hàng trăm, thậm chí hàng nghìn thiết bị.

- Thư viện widget phong phú:

ThingBoard cung cấp sẵn nhiều loại widget (biểu đồ, đồng hồ, đồ thị, bảng...) để người dùng dễ dàng tạo bảng điều khiển tùy chỉnh, giúp hiển thị dữ liệu trực quan và sinh động.

Khả năng mở rộng và bảo mật: ThingBoard hỗ trợ triển khai theo mô hình cloud và on-premises, dễ dàng mở rộng quy mô và bảo đảm an ninh dữ liệu thông qua các cơ chế xác thực, mã hóa.

Ứng dụng:

- Giám sát và điều khiển thiết bị IoT: ThingBoard thường được sử dụng trong các dự án IoT để giám sát trạng thái của thiết bị từ xa và đưa ra cảnh báo khi có sự cố.

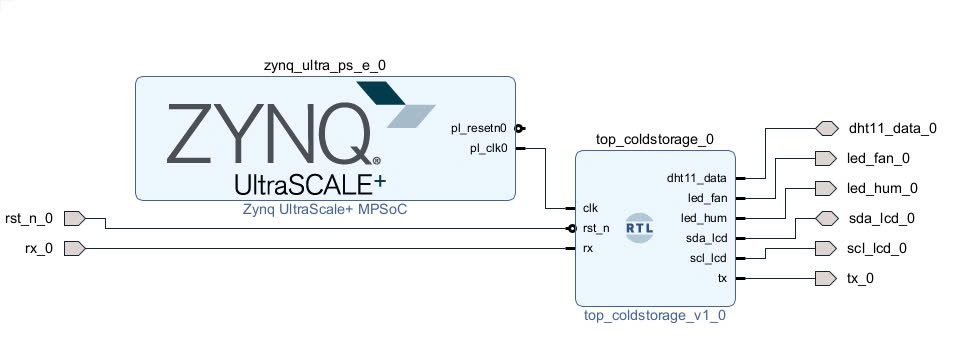
- Phân tích và báo cáo: Với khả năng thu thập và phân tích dữ liệu, ThingBoard giúp người dùng đưa ra quyết định nhanh chóng và chính xác dựa trên các thông tin thu thập được từ các cảm biến IoT.

- Nghiên cứu và phát triển (R&D): ThingBoard là công cụ hỗ trợ đắc lực cho các dự án nghiên cứu trong lĩnh vực IoT, giúp các nhà phát triển và kỹ sư thử nghiệm các hệ thống trước khi triển khai thực tế.

- Ứng dụng trong các ngành công nghiệp: Từ nông nghiệp thông minh, quản lý năng lượng, đến giám sát môi trường, ThingBoard đang được ứng dụng rộng rãi trong nhiều ngành công nghiệp khác nhau để tối ưu hóa quy trình và cải thiện hiệu suất.

# III, QUÁ TRÌNH VÀ THIẾT KẾ

## **3.1 SƠ ĐỒ KHỐI HỆ THỐNG**



Sơ đồ khối này mô tả kiến trúc phần cứng của hệ thống nhúng dựa trên **Zynq UltraScale+ MPSoC** và một mô-đun phần cứng có tên **top\_coldstorage\_v1\_0**. Dưới đây là giải thích về các thành phần chính:

**1. Zynq UltraScale+ MPSoC**

* Đây là bộ xử lý chính của hệ thống, bao gồm CPU (Processing System - PS) và khối lập trình FPGA (Programmable Logic - PL).
* Các tín hiệu đầu vào của Zynq:
  + **pl\_resetn0**: Reset cho khối PL.
  + **pl\_clk0**: Xung nhịp cấp cho khối PL.

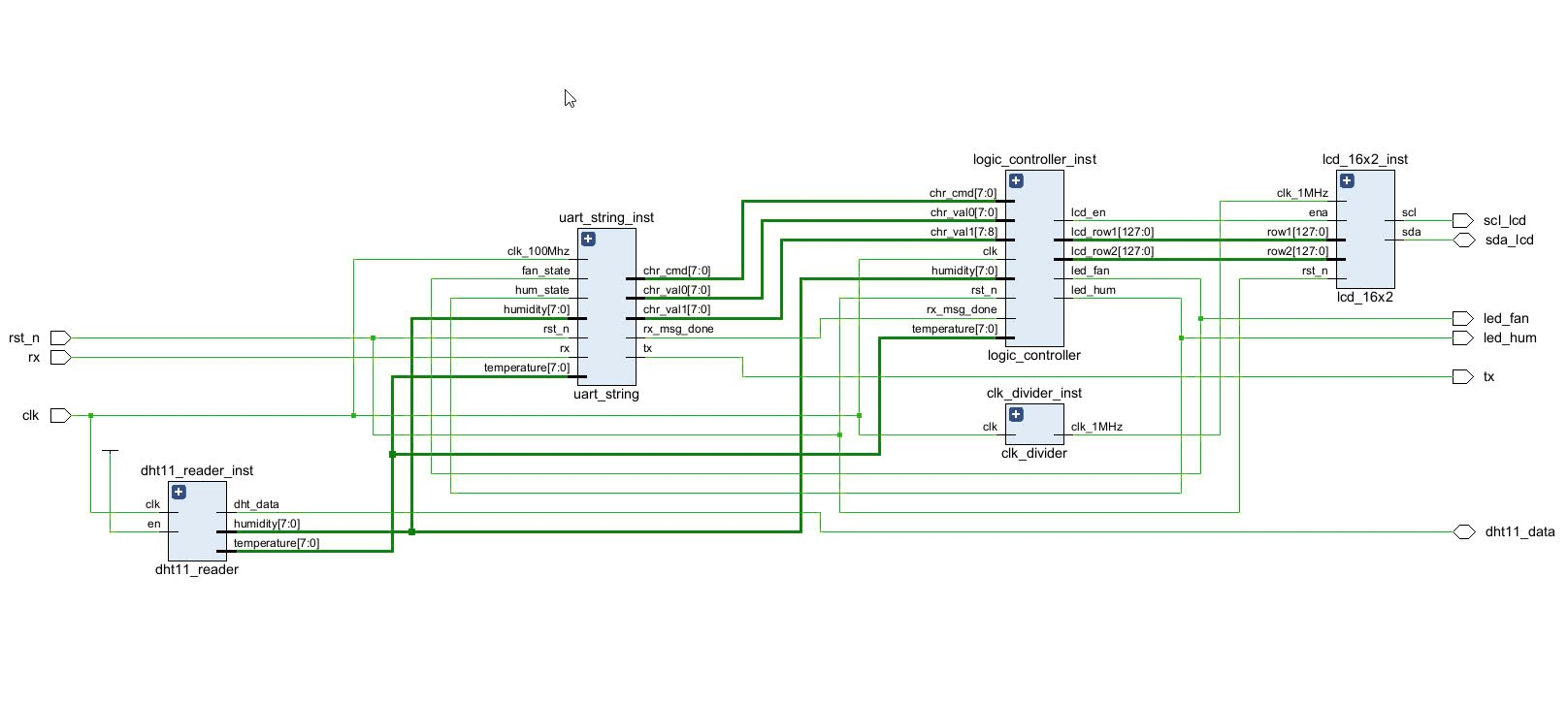
**2. Mô-đun top\_coldstorage\_v1\_0**

* Đây là một khối RTL (Register Transfer Level) trong PL, có nhiệm vụ xử lý dữ liệu từ các cảm biến và điều khiển thiết bị ngoại vi.
* **Các tín hiệu kết nối với Zynq:**
  + **clk**: Xung nhịp điều khiển từ Zynq.
  + **rst\_n**: Tín hiệu reset.
  + **rx**: Tín hiệu nhận dữ liệu.
* **Các tín hiệu đầu ra và đầu vào ngoại vi:**
  + **dht11\_data\_0**: Kết nối cảm biến nhiệt độ & độ ẩm DHT11.
  + **led\_fan\_0**: Điều khiển đèn LED cho quạt.
  + **led\_hum\_0**: Điều khiển đèn LED cho độ ẩm.
  + **sda\_lcd\_0, scl\_lcd\_0**: Giao tiếp I2C với màn hình LCD.
  + **tx\_0**: Tín hiệu truyền dữ liệu.

**Chức năng của hệ thống**

Hệ thống này có thể được sử dụng để quản lý nhiệt độ, độ ẩm trong kho lạnh:

* Đọc dữ liệu cảm biến DHT11.
* Hiển thị thông tin lên màn hình LCD thông qua giao tiếp I2C.
* Điều khiển quạt và đèn LED theo dữ liệu cảm biến.
* Truyền và nhận dữ liệu qua giao tiếp nối tiếp (rx, tx).



**1. DHT11 Reader (dht11\_reader\_inst)**

* Thành phần này chịu trách nhiệm đọc dữ liệu từ cảm biến **DHT11**.
* Các tín hiệu chính:
  + **clk**: Xung nhịp đầu vào.
  + **en**: Tín hiệu cho phép đọc dữ liệu.
  + **dht\_data**: Tín hiệu đầu vào từ cảm biến.
  + **humidity[7:0]**: Giá trị độ ẩm đọc được.
  + **temperature[7:0]**: Giá trị nhiệt độ đọc được.

**2. UART String (uart\_string\_inst)**

* Thành phần này xử lý dữ liệu để truyền qua giao tiếp **UART**.
* Các tín hiệu chính:
  + **clk\_100MHz**: Xung nhịp hệ thống.
  + **rx, tx**: Giao tiếp UART.
  + **chr\_cmd[7:0], chr\_val0[7:0], chr\_val1[7:0]**: Chuỗi dữ liệu gửi qua UART.
  + **rx\_msg\_done**: Báo hiệu khi có dữ liệu nhận.

**3. Logic Controller (logic\_controller\_inst)**

* Thành phần điều khiển logic hệ thống, xử lý dữ liệu từ cảm biến và điều khiển các thiết bị ngoại vi như quạt và LED.
* Các tín hiệu chính:
  + **clk**: Xung nhịp hệ thống.
  + **humidity[7:0], temperature[7:0]**: Dữ liệu từ cảm biến.
  + **led\_fan, led\_hum**: Điều khiển LED báo quạt và độ ẩm.
  + **rx\_msg\_done**: Xác nhận dữ liệu UART đã hoàn tất.

**4. Clock Divider (clk\_divider\_inst)**

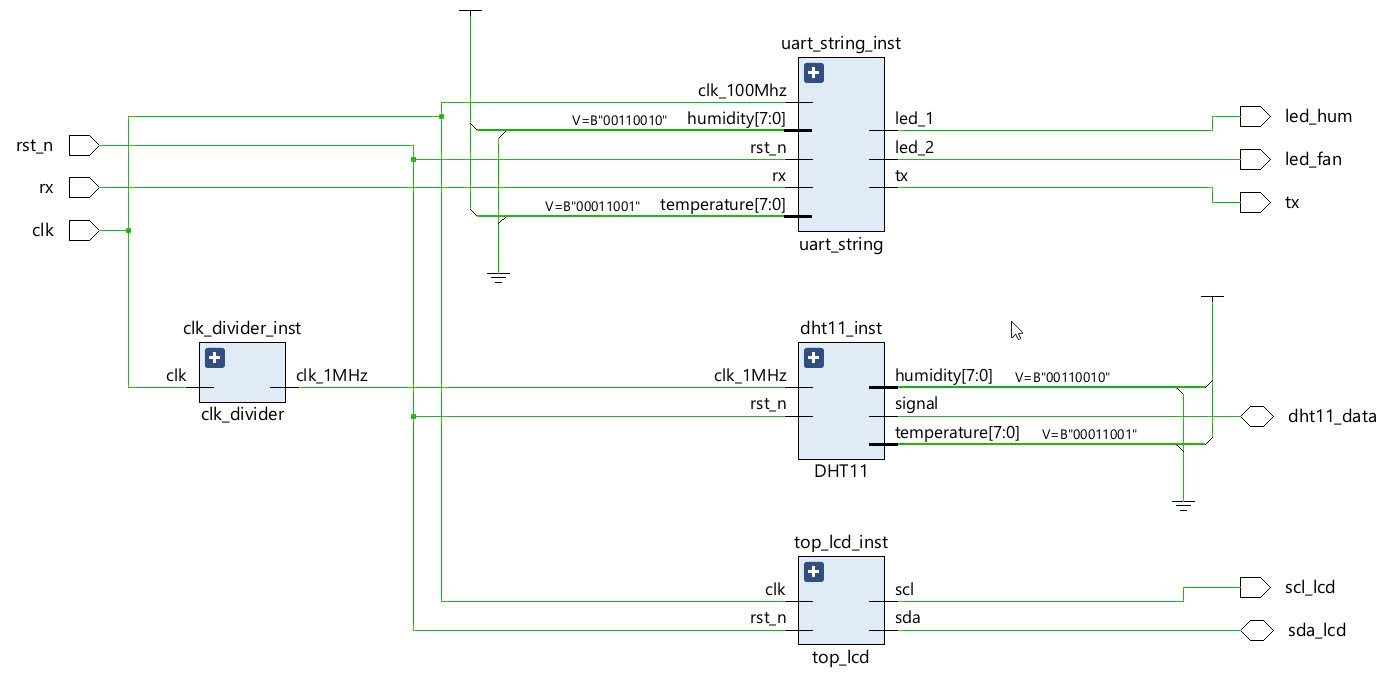
* Bộ chia tần số xung nhịp để cung cấp xung nhịp phù hợp cho các module khác.
* Tạo xung nhịp **1MHz** cho LCD và các thành phần khác.

**5. LCD Controller (lcd\_16x2\_inst)**

* Điều khiển màn hình LCD **16x2** để hiển thị nhiệt độ và độ ẩm.
* Các tín hiệu chính:
  + **clk\_1MHz**: Xung nhịp cung cấp cho LCD.
  + **lcd\_en, lcd\_row[127:0]**: Điều khiển hiển thị trên màn hình.
  + **scl\_lcd, sda\_lcd**: Tín hiệu giao tiếp **I2C** với LCD.

**Chức năng tổng thể của hệ thống**

1. **Đọc dữ liệu cảm biến DHT11** và gửi đến bộ điều khiển.
2. **Gửi dữ liệu qua UART** để truyền đến máy tính hoặc hệ thống giám sát.
3. **Điều khiển quạt và LED** dựa trên giá trị nhiệt độ, độ ẩm.
4. **Hiển thị dữ liệu trên LCD** bằng giao tiếp I2C.
5. **Đồng bộ xung nhịp** giữa các thành phần thông qua bộ chia tần số.



**1. Các tín hiệu đầu vào chính**

* **clk**: Tín hiệu xung nhịp hệ thống.
* **rst\_n**: Tín hiệu reset, active-low (reset khi ở mức 0).
* **rx**: Tín hiệu nhận dữ liệu từ UART (từ máy tính hoặc thiết bị khác).

**2. Bộ chia xung clk\_divider\_inst**

* Nhận **clk** đầu vào và chia tần số xuống **1MHz** cho các khối cần tốc độ chậm hơn.
* **clk\_1MHz** được đưa đến:
  + DHT11 (vì cảm biến này yêu cầu xung nhịp thấp).
  + LCD (nhiều module LCD I2C cần xung nhịp thấp để hoạt động ổn định).

**3. Khối đọc cảm biến dht11\_inst**

* Giao tiếp với cảm biến **DHT11** thông qua tín hiệu **dht11\_data**.
* Sử dụng **clk\_1MHz** và **rst\_n**.
* Xuất ra:
  + **humidity[7:0]**: Giá trị độ ẩm.
  + **temperature[7:0]**: Giá trị nhiệt độ.
  + **signal**: Tín hiệu điều khiển hoặc trạng thái.

**4. Giao tiếp UART uart\_string\_inst**

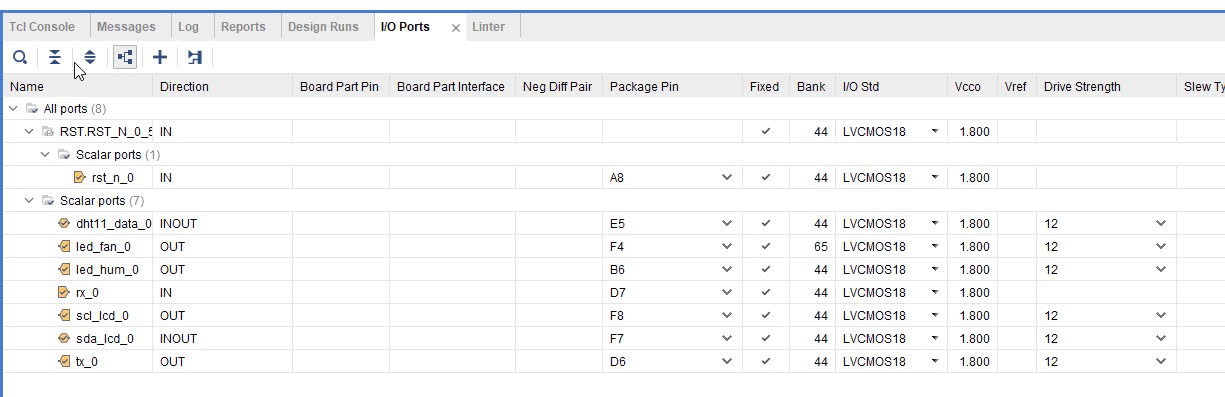
* Truyền dữ liệu nhiệt độ, độ ẩm qua cổng UART.
* Nhận **humidity**, **temperature**, và điều khiển:
  + **led\_1** → **led\_hum**: Điều khiển LED theo độ ẩm.
  + **led\_2** → **led\_fan**: Điều khiển LED theo quạt/nhiệt độ.
  + **tx**: Tín hiệu truyền dữ liệu UART.

**Hiển thị LCD – top\_lcd\_inst**

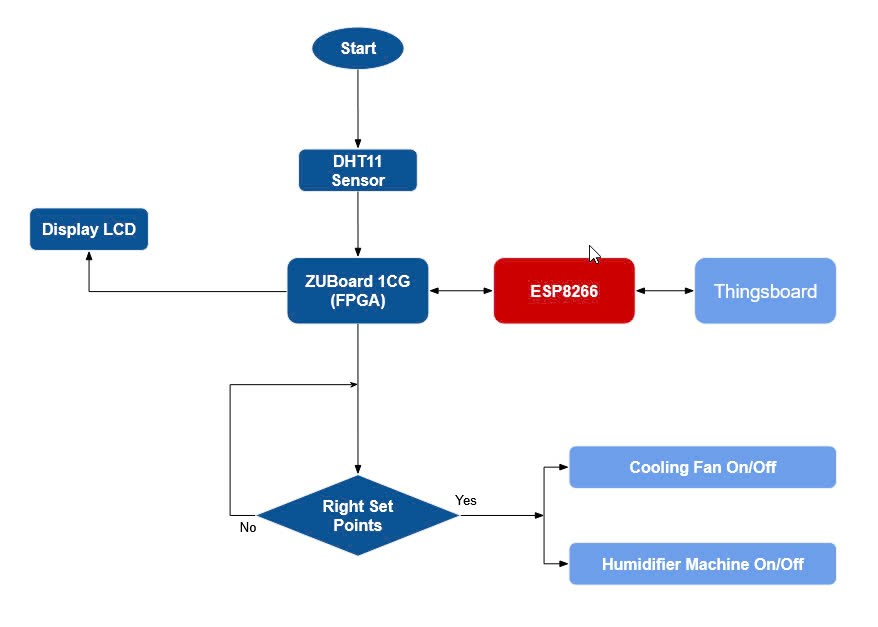
* Giao tiếp LCD I2C để hiển thị nhiệt độ và độ ẩm.
* Dùng tín hiệu:
  + **clk** và **rst\_n**.
  + Xuất ra: **scl\_lcd**, **sda\_lcd**: Hai đường truyền I2C.

**Luồng dữ liệu hệ thống**

1. **DHT11** đo nhiệt độ & độ ẩm → gửi về **dht11\_inst**.
2. Dữ liệu gửi đến:
   * **UART** để hiển thị lên terminal hoặc truyền cho thiết bị khác.
   * **LCD** để hiển thị trực tiếp trên màn hình.
   * **LED** để cảnh báo trực quan.



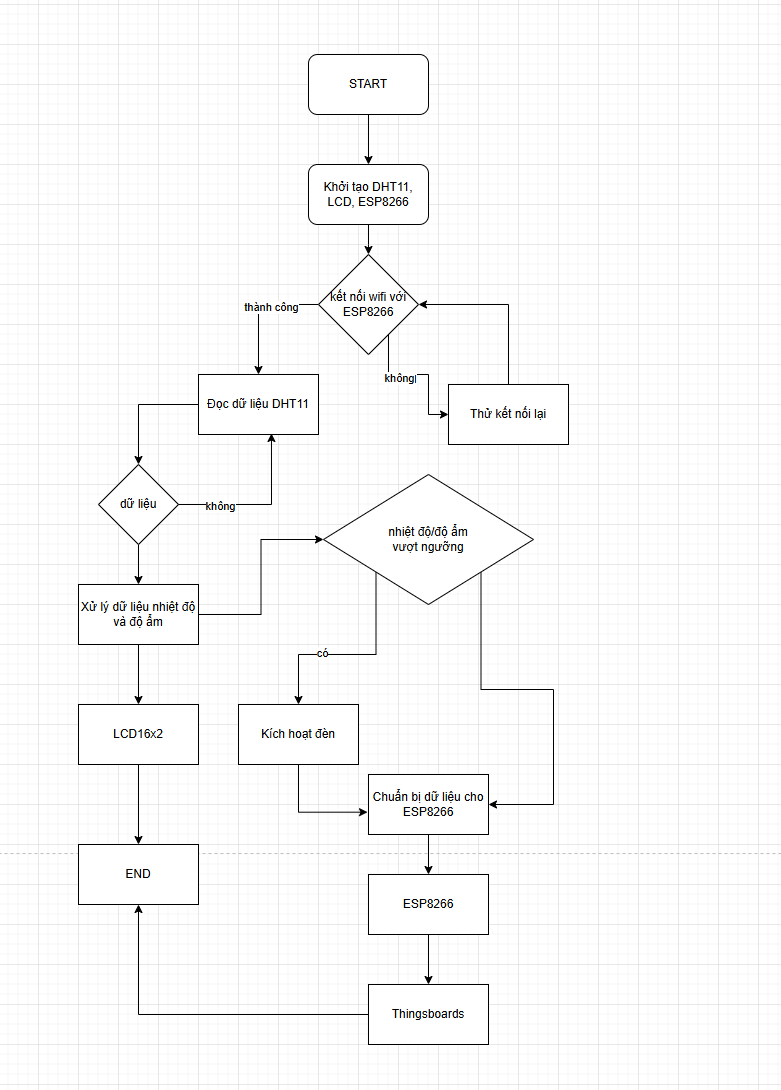
## **3.2 WORKFLOW**



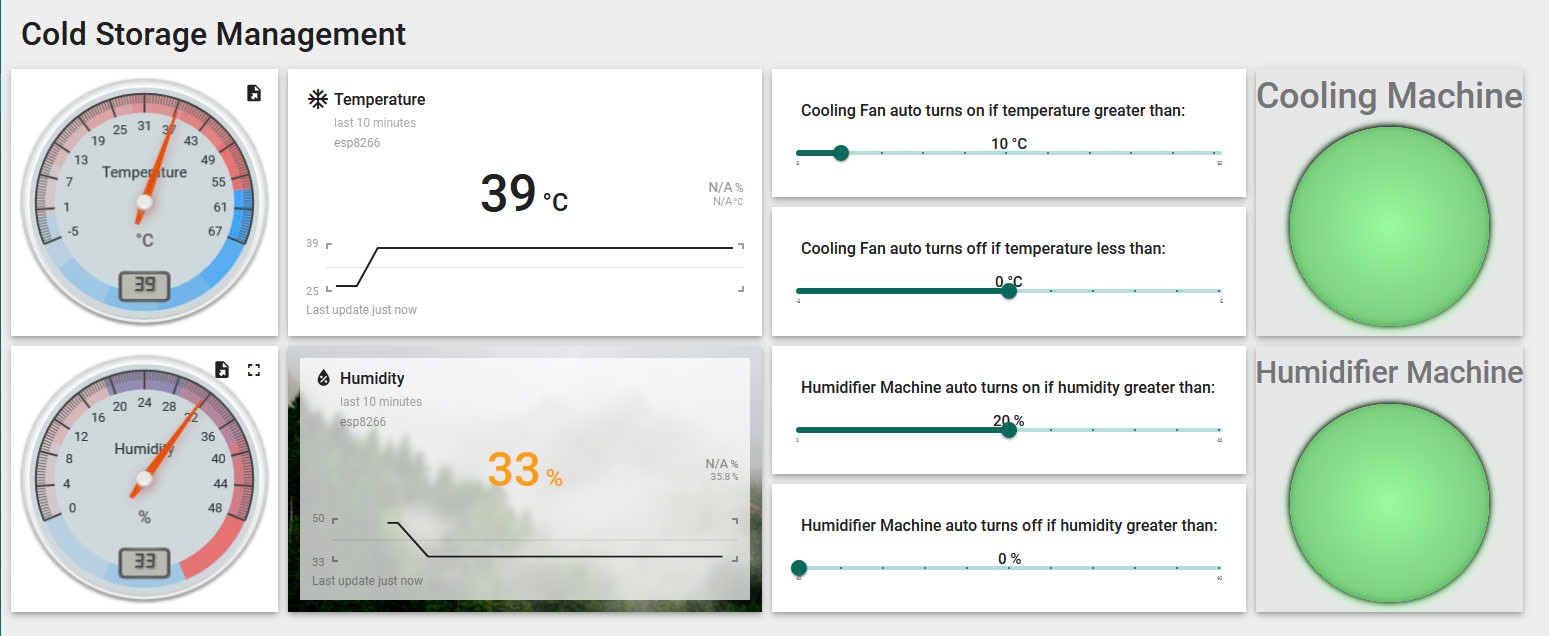
**Mô tả từng bước trong workflow**

1. **Bắt đầu (Start)**
   * Hệ thống khởi động và sẵn sàng thu thập dữ liệu.
2. **Đọc dữ liệu từ cảm biến DHT11**
   * Cảm biến **DHT11** thu thập dữ liệu về **nhiệt độ** và **độ ẩm**.
3. **Truyền dữ liệu đến ZUBoard 1CG (FPGA)**
   * **ZUBoard 1CG** (một loại FPGA) nhận dữ liệu từ DHT11 để xử lý.
4. **Hiển thị dữ liệu trên LCD**
   * FPGA có thể hiển thị dữ liệu **nhiệt độ/độ ẩm** lên **màn hình LCD**.
5. **Gửi dữ liệu lên ThingsBoard thông qua ESP8266**
   * **ESP8266** đóng vai trò là module WiFi, giúp FPGA truyền dữ liệu lên **ThingsBoard**, một nền tảng IoT để giám sát từ xa.
6. **Kiểm tra điều kiện nhiệt độ - độ ẩm ("Right Set Points")**
   * FPGA so sánh dữ liệu thực tế với các ngưỡng đặt trước (**Set Points**).
   * Nếu **chưa đạt** giá trị mong muốn → quay lại vòng lặp và tiếp tục kiểm tra.
   * Nếu **vượt ngưỡng**, thực hiện điều khiển.
7. **Điều khiển thiết bị làm mát và tạo độ ẩm**
   * Nếu nhiệt độ quá cao → **Bật/tắt quạt làm mát (Cooling Fan)**.
   * Nếu độ ẩm quá thấp → **Bật/tắt máy tạo ẩm (Humidifier Machine)**.

## **3.3 Flowchart**

****

**3.3 Kiểm thử và hoàn thiện  
Giao diện thingsboard**

****

# **IV, Kết luận và hướng phát triển**

## **4.1 Kết luận**

Hệ thống **Cold Storage Management** đã được thiết kế và triển khai thành công, đáp ứng các yêu cầu giám sát nhiệt độ và độ ẩm trong kho lạnh. Hệ thống có khả năng:  
✅ Đo lường chính xác nhiệt độ, độ ẩm bằng cảm biến DHT11.  
✅ Truyền dữ liệu lên ThingsBoard Cloud qua ESP8266 để giám sát từ xa.  
✅ Hiển thị thông tin trên màn hình LCD16x2 giúp quan sát trực tiếp.  
✅ Cảnh báo ngay lập tức khi nhiệt độ vượt ngưỡng cài đặt.  
✅ Lưu trữ dữ liệu để phân tích xu hướng và tối ưu quy trình bảo quản.

Mặc dù hệ thống đã hoạt động ổn định và đạt được mục tiêu ban đầu, nhưng vẫn còn một số hạn chế như:  
⚠ Độ chính xác của cảm biến DHT11 chưa cao.  
⚠ Phụ thuộc vào kết nối Wi-Fi, dễ bị gián đoạn dữ liệu.  
⚠ Chưa có khả năng điều khiển thiết bị làm lạnh tự động.

## **4.2 Ưu điểm**

✅ **Giám sát thời gian thực**: Hệ thống liên tục theo dõi nhiệt độ, độ ẩm và cập nhật dữ liệu tức thời lên nền tảng ThingsBoard Cloud.

✅ **Truy cập từ xa**: Người dùng có thể theo dõi dữ liệu từ bất kỳ đâu thông qua giao diện ThingsBoard trên trình duyệt hoặc điện thoại.

✅ **Cảnh báo kịp thời**: Khi nhiệt độ hoặc độ ẩm vượt ngưỡng cho phép, hệ thống sẽ gửi cảnh báo ngay qua email hoặc SMS để người quản lý có hành động xử lý kịp thời.

✅ **Tự động ghi nhận dữ liệu**: Lưu trữ dữ liệu trên đám mây giúp phân tích xu hướng nhiệt độ theo thời gian, hỗ trợ tối ưu hóa bảo quản hàng hóa.

✅ **Chi phí thấp, dễ triển khai**: Hệ thống sử dụng các linh kiện phổ biến như DHT11, ESP8266, ZUBoard 1CG giúp giảm chi phí và dễ dàng nhân rộng.

✅ **Tiết kiệm nhân lực**: Giúp giảm tải công việc giám sát thủ công, tiết kiệm thời gian và công sức của nhân viên.

✅ **Dễ dàng mở rộng**: Có thể nâng cấp với các cảm biến mới hoặc kết nối thêm nhiều thiết bị để mở rộng phạm vi giám sát.

## **4.3 Nhược điểm**

⚠ **Độ chính xác cảm biến hạn chế**: DHT11 có sai số ±2°C và ±5% độ ẩm, chưa phù hợp với những ứng dụng yêu cầu độ chính xác cao.

⚠ **Phụ thuộc vào kết nối Wi-Fi**: Nếu mạng yếu hoặc mất kết nối, dữ liệu có thể không được cập nhật kịp thời, ảnh hưởng đến việc giám sát.

⚠ **Không kiểm soát chủ động**: Hệ thống chỉ giám sát và cảnh báo, chưa thể tự động điều chỉnh nhiệt độ kho lạnh (ví dụ: bật/tắt máy nén lạnh khi cần thiết).

⚠ **Khả năng lưu trữ dữ liệu hạn chế**: Việc phụ thuộc vào ThingsBoard Cloud khiến hệ thống cần có chiến lược lưu trữ hoặc xoá bớt dữ liệu cũ để tránh quá tải.

⚠ **Không hỗ trợ nhiều thông số khác**: Hệ thống chỉ giám sát nhiệt độ và độ ẩm, chưa tích hợp các cảm biến khác như CO₂, áp suất không khí, hoặc mức oxy để đánh giá điều kiện kho lạnh toàn diện hơn.

## **4.4 Hướng phát triển**

Trong tương lai, hệ thống có thể được nâng cấp để cải thiện hiệu suất và mở rộng chức năng:  
🔹 **Nâng cấp cảm biến**: Sử dụng cảm biến có độ chính xác cao hơn như DHT22 hoặc SHT31.  
🔹 **Tích hợp AI**: Dùng mô hình học máy để dự đoán sự cố hệ thống làm lạnh và tối ưu bảo quản.  
🔹 **Tăng cường lưu trữ dữ liệu**: Kết hợp cơ sở dữ liệu SQL hoặc NoSQL để phân tích dữ liệu dài hạn.  
🔹 **Mở rộng giao tiếp**: Hỗ trợ thêm giao thức như LoRa, Zigbee để tăng tính linh hoạt trong truyền dữ liệu.  
🔹 **Tích hợp hệ thống điều khiển**: Tự động bật/tắt hệ thống làm lạnh khi nhiệt độ vượt ngưỡng.

Bằng cách nâng cấp và tối ưu hệ thống, **Cold Storage Management** có thể trở thành một giải pháp toàn diện, giúp doanh nghiệp quản lý kho lạnh hiệu quả hơn, giảm thiểu tổn thất và tối ưu hóa chi phí vận hành.