**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**BÁO CÁO BÀI TẬP LỚN**

**Xây dựng hệ thống**

**ghép nối cảm biến đo nhiệt độ LM35**

**kết nối đến server**

**NGUYỄN BÁCH THẮNG**

thang.nb153505@sis.hust.edu.vn

**Ngành Công nghệ thông tin**

**Chuyên ngành Kỹ thuật máy tính**

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn:** | PGS. TS. Nguyễn Đình Thuận  Chữ ký của GVHD |
| **Bộ môn:** | Kỹ thuật máy tính |
| **Viện:** | Công nghệ thông tin và truyền thông |
| **HÀ NỘI, 8/2021** | |

Mục lục

[1. Tổng quan về hệ thống 3](#_Toc79484845)

[2. Ghép nối cảm biến nhiệt độ LM35 với cổng COM 4](#_Toc79484846)

[2.1. Thành phần của mạch ghép nối 4](#_Toc79484847)

[2.2. Giao tiếp giữa các thành phần với nhau 4](#_Toc79484848)

[3. Nhận dữ liệu từ cổng COM và gửi lên server 5](#_Toc79484849)

[3.1. Ứng dụng đọc và gửi nhận giá trị nhiệt độ 6](#_Toc79484850)

[3.2. Các thư viện chính được sử dụng 6](#_Toc79484851)

[4. Truy xuất dữ liệu từ database và vẽ đồ thị theo thời gian trên web 6](#_Toc79484852)

[4.1. Ứng dụng Web 6](#_Toc79484853)

[4.2. Controller 7](#_Toc79484854)

[4.3. Model 8](#_Toc79484855)

# Tổng quan về hệ thống

Timeline

Description automatically generated

Hình : Sơ đồ hệ thống ghép nối cơ bản

Luồng vận hành hệ thống sẽ thực hiện lần lượt như sau:

1. Hệ thống có phần cứng gồm: cảm biến nhiệt độ LM35, ATMEGA8 được mô phỏng trên Proteus kết nối với cổng COM.
2. Thông qua cổng COM, phần mềm trên máy tính sẽ thực hiện đọc ra giá trị và gửi giá trị đó lên server.
3. Server được vận hành dựa trên web app có nhiệm vụ bắt các gói tin được gửi lên và lưu lại vào database, đồng thời vẽ đồ thị nhiệt độ theo thời gian.

# Ghép nối cảm biến nhiệt độ LM35 với cổng COM

Diagram, schematic

Description automatically generated

Hình : Mạch ghép nối cảm biến nhiệt độ LM35 với cổng COM

## Thành phần của mạch ghép nối

Mạch ghép nối bao gồm 3 thành phần:

* Vi điều khiển Atmega8
* Cảm biến nhiệt độ LM35
* Cổng Com để giao tiếp với máy tính thông qua giao thức UART

## Giao tiếp giữa các thành phần với nhau

* Để vi điều khiển có thể đọc thông tin từ LM35, ta cần khởi tạo và đọc giá trị từ chân ADC như sau:

Text

Description automatically generated

Trong đó, hàm ADC\_Init(): khởi tạo ban đầu cho cổng ADC

hàm ADC\_Read(char channel): đọc và trả về giá trị của cảm biến nhiệt độ LM35

* Khởi tạo giao tiếp thông qua giao thức UART cho Atmega8 với cổng COM:

Text

Description automatically generated

Giao thức UART là giao thức không đồng bộ, do đó, dữ liệu truyền bằng UART sẽ được tổ chức thành các gói. Mỗi gói chứa 1 bit bắt đầu, 5 đến 9 bit dữ liệu (tùy thuộc vào UART), một bit chẵn lẻ tùy chọn và 1 hoặc 2 bit dừng.

Diagram

Description automatically generated

# Nhận dữ liệu từ cổng COM và gửi lên server

Chart, line chart

Description automatically generated

Hình 3: Giao diện ứng dụng trên máy tính để đọc và gửi dữ liệu

## Ứng dụng đọc và gửi nhận giá trị nhiệt độ

* Ứng dụng đọc và gửi dữ liệu được viết bằng Winform.

Windowns Forms là thuật ngữ mô tả một ứng dụng được viết dùng .NET FrameWorrk và có giao diện người dùng Windows Forms (màn hình windows).

* Chức năng chính:
  + Đọc nhiệt độ từ cổng COM và gửi lên server thông qua API
  + Hiển thị nhiệt độ theo thời gian

## Các thư viện chính được sử dụng

* **System.IO.Ports**: cung cấp giao tiếp với cổng COM
* **ZedGraph**: vẽ đồ thị theo thời gian
* **System.Net.Http**: cung cấp các giao tiếp với API

# Truy xuất dữ liệu từ database và vẽ đồ thị theo thời gian trên web

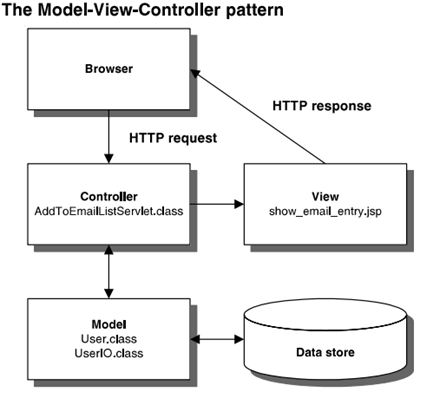
Chart, line chart

Description automatically generated

Hình 4 Giao diện trang web

## Ứng dụng Web

* Web app được viết theo cấu trúc MVC:



* Được kết nối với Database của SQL server sử dụng LINQ
* Các chức năng chính:
  + Nhận dữ liệu về nhiệt độ và lưu vào Database
  + Hiển thị giá trị trong Database lên trang Web

## Controller

* Đảm nhận nhiệm vụ lấy dữ liệu từ ứng dụng máy tính gửi lên

|  |
| --- |
| public ActionResult GetTemp()  {  List<Temp> lstTemp = null;  using (var dc = new DataClasses1DataContext())  {  lstTemp = dc.Temps.ToList();  }  return Json(lstTemp, JsonRequestBehavior.AllowGet);  } |

* Gửi dữ liệu vào database:

|  |
| --- |
| public ActionResult SetTemp(int Temperature)  {  var temp = new Temp();  using (var dc = new DataClasses1DataContext())  {  temp.Time = DateTime.Now;  if (Temperature < 100)  {  temp.Temperature = Temperature;  }  dc.Temps.InsertOnSubmit(temp);  dc.SubmitChanges();  }  return Json(temp, JsonRequestBehavior.AllowGet);  } |

* Đọc giá trị và vẽ đồ thị

|  |
| --- |
| public ContentResult JSON(int xStart = 0)  {  List<DataPoint> dataPoints = new List<DataPoint>();  var lstTemp = GetTempIndex();  foreach (var temp in lstTemp)  {  var y = Convert.ToDouble(temp.Temperature);  var x = Convert.ToDouble(temp.Id);  dataPoints.Add(new DataPoint(xStart + x, y));  }  return Content(JsonConvert.SerializeObject(dataPoints, \_jsonSetting), "application/json");  }  JsonSerializerSettings \_jsonSetting = new JsonSerializerSettings() { NullValueHandling = NullValueHandling.Ignore }; |

## Model

* Làm cầu nối giữa Controller và View trong việc vẽ đồ thị

|  |
| --- |
| public ContentResult JSON(int xStart = 0)  {  List<DataPoint> dataPoints = new List<DataPoint>();  var lstTemp = GetTempIndex();  foreach (var temp in lstTemp)  {  var y = Convert.ToDouble(temp.Temperature);  var x = Convert.ToDouble(temp.Id);  dataPoints.Add(new DataPoint(xStart + x, y));  }  return Content(JsonConvert.SerializeObject(dataPoints, \_jsonSetting), "application/json");  }  JsonSerializerSettings \_jsonSetting = new JsonSerializerSettings() { NullValueHandling = NullValueHandling.Ignore }; |