**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA ĐIỆN – ĐIỆN TỬ**

**BỘ MÔN VIỄN THÔNG**

----------------------o0o----------------------

****

**BÁO CÁO ĐỒ ÁN**

**ĐỀ TÀI: ĐỌC DỮ LIỆU CẢM BIẾN DHT11 VÀ**

**TRUYỀN TÍN HIỆU QUA MODULE SUB-GHZ**

**CỦA TEXAS INSTRUMENTS**

|  |  |  |
| --- | --- | --- |
| **GVHD:** | **Huỳnh Phú Minh Cường** | |
| **SVTH:** | **Lưu Công Dâng** | **1811671** |
|  | **Đặng Hồng Quân** | **1811173** |

*TP. Hồ Chí Minh, Tháng 8, Năm 2021*

**LỜI CẢM ƠN**

Trong thời gian thực hiện đề tài, những người thực hiện được sự giúp đỡ của gia đình, quý thầy cô và bạn bè nên đề tài đã được hoàn thành. Những người thực hiện xin chân thành gửi lời cảm ơn đến:

Thầy Huỳnh Phú Minh Cường, giảng viên trường Đại Học Bách Khoa TPHCM và các thành viên trong Lab RFICs đã trực tiếp hướng dẫn và nhiệt tình giúp đỡ tạo điều kiện để chúng em có thể hoàn thành tốt đề tài và đúng thời hạn.

Các em cũng xin chân thành cảm ơn đến các thầy cô trong khoa Điện - Điện tử của trường Đại Học Bách Khoa TPHCM đã tận tình dạy dỗ, chỉ bảo, cung cấp cho chúng em hiện những kiến thức nền, chuyên môn làm cơ sở để hoàn thành đề tài này.

Xin gửi lời cảm ơn đến những người bạn sinh viên khoa Điện-Điện tử đã giúp đỡ những người thực hiện đề tài để có thể hoàn thành tốt đề tài này.

Xin chân thành cảm ơn!

Người thực hiện đề tài:

Lưu Công Dâng

Đặng Hồng Quân

**MỤC LỤC**

[CHƯƠNG 1: TỔNG QUAN 1](#_Toc76318377)

[**1.1** **ĐẶT VẤN ĐỀ:** 1](#_Toc76318378)

[**1.2.** **MỤC TIÊU:** 1](#_Toc76318379)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 2](#_Toc76318380)

[**2.1.** **Công nghệ Sub-GHz:** 2](#_Toc76318381)

[**2.1.1.** **Khái niệm:** 2](#_Toc76318382)

[**2.1.2.** **Đặc điểm:** 3](#_Toc76318383)

[**2.1.3.** **Ứng dụng: Cảm biến không dây Sub-GHz:** 3](#_Toc76318384)

[**2.2.** **Tổng quan về Arduino:** 4](#_Toc76318385)

[**2.3.** **Giới thiệu phần cứng** 5](#_Toc76318386)

[**2.3.1.** **Arduino UNO R3:** 5](#_Toc76318387)

[**2.3.2.** **Cảm biến độ ẩm, nhiệt độ (DHT11):** 6](#_Toc76318388)

[**2.3.3.** **Module CC1101:** 8](#_Toc76318389)

[**2.4.** **Quá trình truyền dữ liệu giữa 2 module CC1101:** 9](#_Toc76318390)

[**2.4.1.** **Xử lý dữ liệu ở máy phát:** 9](#_Toc76318391)

[**2.4.2.** **Xử lý dữ liệu ở máy thu:** 10](#_Toc76318392)

[**2.4.3.** **Định dạng điều chế tín hiệu:** 10](#_Toc76318393)

[**2.4.4.** **Sửa lỗi:** 12](#_Toc76318394)

[**2.4.4.1.** **Sửa lỗi chuyển tiếp (FEC):** 12](#_Toc76318395)

[**2.4.4.2.** **Sửa lỗi xen kẽ:** 13](#_Toc76318396)

[**2.5.** **Giao tiếp với vi điều khiển:** 14](#_Toc76318397)

[**2.5.1.** **DHT11:** 14](#_Toc76318398)

[**2.5.1.1.** **Chuẩn giao tiếp 1-wire:** 14](#_Toc76318399)

[**2.5.1.2.** **DHT11 giao tiếp với Arduino:** 15](#_Toc76318400)

[**2.5.2.** **Module CC1101:** 18](#_Toc76318401)

[**2.5.2.1.** **Chuẩn giao tiếp SPI:** 18](#_Toc76318402)

[**2.5.2.2.** **CC1101 Giao tiếp với Arduino:** 18](#_Toc76318403)

[**2.6.** **Các phần mềm sử dụng:** 21](#_Toc76318404)

[**2.6.1.** **Arduino IDE:** 21](#_Toc76318405)

[**2.6.1.1.** **Giới thiệu:** 21](#_Toc76318406)

[**2.6.1.2.** **Giao diện chương trình:** 22](#_Toc76318407)

[**2.6.2.** **Microsoft Visual Studio 2019:** 22](#_Toc76318408)

[**2.6.2.1.** **Giới thiệu:** 22](#_Toc76318409)

[**2.6.2.2.** **Ứng dụng Windows Form:** 23](#_Toc76318410)

[CHƯƠNG 3: THIẾT KẾ VÀ XÂY DỰNG HỆ THỐNG 24](#_Toc76318411)

[**3.1.** **Sơ đồ phần cứng:** 24](#_Toc76318412)

[**3.1.1.** **Sơ đồ mạch phần cứng Arduino** ⭢ **CC1101:** 24](#_Toc76318413)

[**3.1.2.** **Sơ đồ mạch phần cứng Arduino** ⭢ **DHT11:** 24](#_Toc76318414)

[**3.2.** **Lưu đồ giải thuật:** 25](#_Toc76318415)

[**3.2.1.** **Lưu đồ giải thuật máy phát:** 25](#_Toc76318416)

[**3.2.2.** **Lưu đồ giải thuật máy thu:** 26](#_Toc76318417)

[**3.3.** **Lập trình hệ thống trên Arduino:** 27](#_Toc76318418)

[**3.3.1.** **Thu thập dữ liệu DHT11:** 27](#_Toc76318419)

[**3.3.2.** **Gửi dữ liệu ở máy phát:** 27](#_Toc76318420)

[**3.3.3.** **Nhận dữ liệu ở máy thu:** 28](#_Toc76318421)

[**3.4.** **Lập trình giao diện giám sát trên Visual Studio:** 30](#_Toc76318422)

[**3.4.1.** **Giao diện:** 30](#_Toc76318423)

[**3.4.2.** **Lập trình sự kiện:** 30](#_Toc76318424)

[CHƯƠNG 4: KẾT QUẢ VÀ NHẬN XÉT 32](#_Toc76318425)

[**4.1.** **Kết quả:** 32](#_Toc76318426)

[**4.2.** **Thử nghiệm khoảng tần số:** 33](#_Toc76318427)

[**4.3.** **Thử nghiệm khoảng cách truyền:** 34](#_Toc76318428)

[CHƯƠNG 5: TRUYỀN FILE TEXT. 35](#_Toc76318429)

[**5.1.** **Hướng giải quyết vấn đề:** 35](#_Toc76318430)

[**5.2.** **Lập trình hệ thống:** 35](#_Toc76318431)

[**5.2.1.** **Lập trình trên Arduino:** 39](#_Toc76318432)

[**5.2.2.** **Lập trình trên C# Winform:** 35](#_Toc76318433)

[**5.2.2.1.** **Giao diện:** 35](#_Toc76318434)

[**5.2.2.2.** **Gửi dữ liệu cho vi xử lý:** 36](#_Toc76318435)

[**5.2.2.3.** **Nhận dữ liệu từ vi xử lý:** 37](#_Toc76318436)

[**5.3.2.** **Nhận xét:** 39](#_Toc76318437)

[CHƯƠNG 6: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 41](#_Toc76318438)

[**6.1.** **Kết luận:** 41](#_Toc76318439)

[**6.1.1.** **Ưu điểm:** 41](#_Toc76318440)

[**6.1.2.** **Khuyết điểm:** 41](#_Toc76318441)

[**6.1.3.** **Kết quả đạt được:** 41](#_Toc76318442)

[**6.2.** **Hướng phát triển:** 42](#_Toc76318443)

[TÀI LIỆU THAM KHẢO 43](#_Toc76318444)

[PHỤ LỤC 44](#_Toc76318445)

MỤC LỤC HÌNH ẢNH

[*Hình 2.1 : Các loại Board Arduino.* 4](#_Toc76230794)

[*Hình 2.2: Board Arduino Uno R3* 5](#_Toc76230795)

[*Hình 2.3: Module cảm biến nhiệt độ và độ ẩm DHT11* 7](#_Toc76230796)

[*Hình 2.4: Module CC1101* 8](#_Toc76230797)

[*Hình 2.5: Điều chế FSK* 11](#_Toc76230798)

[*Hình 2.6 : Gaussian Filter* 11](#_Toc76230799)

[*Hình 2.7 : Nguyên tắc chung của xen kẽ ma trận* 13](#_Toc76230800)

[*Hình 2.8 : Sơ đồ nối chân của chuẩn giao tiếp 1-wire* 14](#_Toc76230801)

[*Hình 2.9 : Chuẩn giao tiếp 1-wire* 15](#_Toc76230802)

[*Hình 2.10 : Tín hiệu điều khiển khởi động DHT11* 16](#_Toc76230803)

[*Hình 2.11 : Tín hiệu Data từ DHT11 - Bit 0* 17](#_Toc76230804)

[*Hình 2.12 : Tín hiệu Data từ DHT11 - Bit 1* 17](#_Toc76230805)

[*Hình 2.13 : Chuẩn giao tiếp SPI* 21](#_Toc76230806)

[*Hình 2.14 : Giao diện làm việc của Arduino IDE* 22](#_Toc76230807)

[*Hình 2.15 : Giao diện Winform.* 23](#_Toc76230808)

[*Hình 3.1: Sơ đồ nối chân Arduino với CC1101* 24](#_Toc76230809)

[*Hình 3.2: Sơ đồ nối chân Arduino với DHT11* 24](#_Toc76230810)

[*Hình 3.3 : Lưu đồ giải thuật ở máy phát* 25](#_Toc76230811)

[*Hình 3.4 : Lưu đồ giải thuật ở máy thu* 26](#_Toc76230812)

[*Hình 3.5 : Giao diện giám sát.* 30](#_Toc76230813)

[*Hình 4.1 : Kết quả do nhiệt độ và độ ẩm.* 32](#_Toc76230814)

[*Hình 5.1 : Giao diện gửi file text.* 35](#_Toc76230815)

[*Hình 5.2 : Màn hình hiển thị file text bên máy phát.* 38](#_Toc76230816)

[*Hình 5.3: Màn hình hiển thị file text bên máy thu.* 39](#_Toc76230817)

MỤC LỤC TỪ VIẾT TẮT

|  |  |
| --- | --- |
| AC | Alternating Current |
| ADC | Analog  to Digital Converter |
| CRC | Cyclic Redundancy Check |
| DAC | Digital-to-Analog Converter |
| DC | Direct Current |
| FEC | Forward Error Correction |
| FIFO | First-In-First-Out |
| FSK | Frequency Shift Keying |
| GFSK | Gaussian Frequency-Shift Keying |
| HMI | Human-Machine-Interface, |
| IDE | Integrated Development Environment |
| IoT | Internet of Things |
| MISO | Master Input Slave Output |
| MOSI | Master Output Slave Input |
| MSK | Minimum-Shift Keying |
| NTC | Negative Temperature Coefficient |
| PLC | Programmable logic controller |
| PWM | Pulse-Width Modulation |
| RTC | Real-Time Control |
| RX | Receive, Receive Mode |
| SCADA | Supervisory Control And Data Acquisition |
| SCK | Serial Clock |
| SPI | Serial Peripheral Interface |
| SS | Slave Select |
| TX | Transmit, Transmit Mode |

# CHƯƠNG 1: TỔNG QUAN

1. **ĐẶT VẤN ĐỀ:**

Với việc cách cách mạng công nghiệp 4.0 phát triển, có nhiều tác động tích cực đến đời sống xã hội trên nhiều lĩnh vực, đặc biệt là công nghệ IoT, công nghệ truyền không dây được nghiên cứu để phát triển trong lĩnh vực nông nghiệp. Đồ án này trình bày về cách thức đọc nhiệt độ, độ ẩm không khí, truyền nhận dữ liệu không dây và đưa dữ liệu lên máy tính.

* 1. **MỤC TIÊU:**

Đề tài trình bày về phương thức đọc cảm biến độ ẩm, nhiệt độ của không khí và giao tiếp không dây, đưa dữ liệu thu được lên máy tính. Mục tiêu cần đạt được là:

* Đọc dữ liệu cảm biến.
* Truyền dữ liệu qua module CC1101.
* Đọc dữ liệu thông qua Serial.
* Khảo sát và điều chỉnh các khoảng cách và tần số.

# CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

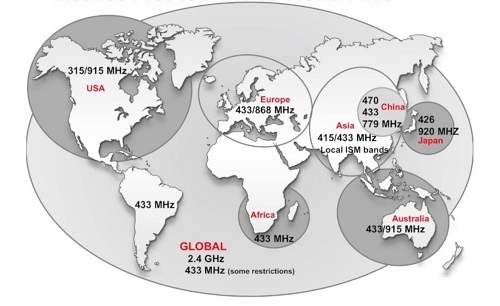
* 1. **Công nghệ Sub-GHz:**

Để kết nối và giao tiếp với các thiết bị có thể sử dụng giải pháp nối dây hoặc giải pháp wireless. Giao tiếp wireless là phương thức truyền dữ liệu trong không gian sử dụng sóng điện từ như sóng radio, sóng hồng ngoại,... Khi triển khai các ứng dụng IoT, công nghệ wireless 2.4 Ghz và wireless Sub - Ghz được ứng dụng chủ yếu.

* + 1. **Khái niệm:**

Công nghệ Sub - Ghz hoạt động ở dải tần nhỏ hơn 1 Ghz. Các quốc gia trên thế giới chủ yếu sử dụng dải tần miễn phí cấp phép 433 Mhz. Tuy nhiên có một số dải tần khác được sử dụng ở một số khu vực:

* Bắc Mỹ và Úc là 915 Mhz
* Châu Âu là 868 Mhz
* Trung Quốc là 470 Mhz và 779 Mhz
* Nhật Bản là 426 Mhz và 920 Mhz.



* + 1. **Đặc điểm:**

Sub - GHz có thể dễ dàng đạt được vài trăm mét trong nhà và vài km (dặm) ngoài trời, tùy thuộc vào điều kiện, có tần số thấp, bước sóng dài nên dễ dàng lan truyền qua các vật cản như tường, cây cối, tòa nhà trong các đô thị và trong nhà máy. Bước sóng dài của Sub – GHz làm cho ít suy hao đường truyền nên truyền đi được xa hơn.

Sub-GHz sử dụng băng tần hẹp và tốc độ truyền thấp nên tiêu thụ ít năng lượng. Sub-GHz cần tín hiệu năng lượng thấp từ bộ truyền cho cùng một tín hiệu ở bộ nhận nên Sub-GHz phù hợp cho các thiết bị cảm biến IoT dùng pin. Một viên pin có thể cấp năng lượng cho cảm biến Sub-GHz lên tới 10-20 năm. Sub-GHz hoạt động ở dải tần thấp hơn và có rất ít thiết bị wireless hoạt động ở dải tần này nên sẽ chống nhiễu tốt hơn.

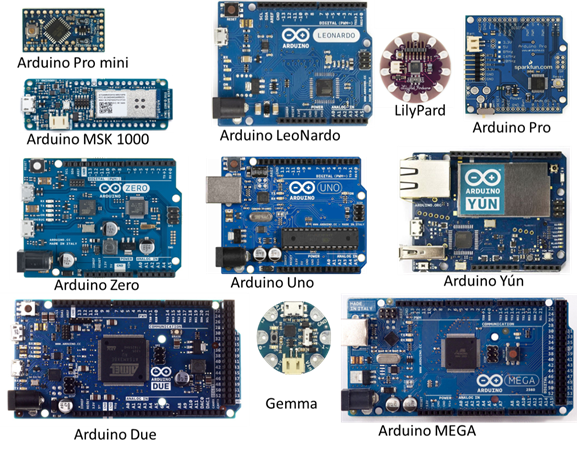
* + 1. **Ứng dụng-Cảm biến không dây Sub-GHz:**

Cảm biến wireless Sub-GHz bao gồm 2 phần chính: phần cảm biến và phần truyền wireless:

* Phần cảm biến có thể là module cảm biến hoặc các cảm biến hoàn chỉnh. Phần truyền wireless bao gồm mạch xử lý tín hiệu và anten truyền.
* Phần truyền wireless thường sử dụng pin để cấp năng lượng cho phần truyền và cho cả phần cảm biến.

Bộ thu wireless Sub - 1GHz gồm ăng ten thu, bộ xử lý tín hiệu và cổng xuất dữ liệu . Mỗi bộ thu wireless có thể kết nối dạng sao với nhiều bộ cảm biến wireless trong vùng phủ sóng của bộ thu. Vùng phủ sóng của bộ thu có thể lên tới vài km khi 2 ăng ten thu và phát nhìn thấy nhau. Khoảng cách truyền sẽ nhỏ hơn nếu có vật cản giữa cảm biến phát và bộ thu wireless. Dữ liệu từ bộ thu wireless có thể kết nối có dây đến PLC, HMI, SCADA, IoT gateway để phục vụ cho việc giám sát, phân tích dữ liệu và lập báo cáo.

* 1. **Tổng quan về Arduino:**

****

*Hình 2.1 : Các loại Board Arduino.*

Arduino thật ra là một bo mạch vi xử lý được dùng để lập trình tương tác với các thiết bị phần cứng như cảm biến, động cơ, đèn hoặc các thiết bị khác. Đặc điểm nổi bật của Arduino là môi trường phát triển ứng dụng cực kỳ dễ sử dụng, với một ngôn ngữ lập trình có thể học một cách nhanh chóng ngay cả với người ít am hiểu về điện tử và lập trình. Và điều làm nên hiện tượng Arduino chính là mức giá rất thấp và tính chất nguồn mở từ phần cứng tới phần mềm.

Các ứng dụng nổi bật của board mạch Arduino: robot đơn giản, điều khiển nhiệt độ, phát hiện chuyển động, game tương tác...

* 1. **Giới thiệu phần cứng**
* **Yêu cầu:**
* Thu thập dữ liệu về nhiệt độ, độ ẩm không khí.
* Gửi dữ liệu thông qua truyền thông với thiết bị do TI sản xuất..
* **Lựa chọn thiết bị:**
* Cảm biến độ ẩm, nhiệt độ DHT11.
* Module CC1101.
* Arduino Uno R3.
  + 1. **Arduino UNO R3:**



*Hình 2.2: Board Arduino Uno R3*

Arduino Uno là một bo mạch vi điều khiển dựa trên ATmega328. Arduino là một nền tảng mã nguồn mở, nguyên mẫu và tính đơn giản của nó làm cho nó trở thành lý tưởng cho những người có sở thích sử dụng cũng như các chuyên gia.

Arduino Uno có 14 chân đầu vào / đầu ra kỹ thuật số (trong đó 6 có thể được sử dụng làm đầu ra PWM), 6 đầu vào tương tự, bộ tạo dao động 16MHz, kết nối USB, giắc cắm nguồn, tiêu đề ICSP và nút đặt lại. Nó chứa mọi thứ cần thiết để hỗ trợ vi điều khiển; chỉ cần kết nối nó với một máy tính bằng cáp USB hoặc cấp điện cho nó bằng bộ chuyển đổi AC-to-DC hoặc pin để bắt đầu.

Arduino Uno khác với tất cả các bo mạch trước ở chỗ nó không sử dụng chip điều khiển FTDI USB-to-Serial. Thay vào đó, nó có chip vi điều khiển Atmega8U2 được lập trình như một bộ chuyển đổi từ USB sang nối tiếp.

* **Thông số kỹ thuật:**
* Chip điều khiển chính: ATmega328P họ 8 bit
* Chip nạp và giao tiếp UART: ATmega16U2
* Điện áp hoạt động: 5VDC
* Tần số hoạt động: 16MHz
* Điện áp vào khuyên dùng: 6-9VDC
* Số chân vào/ra (I/O số): 14 (6 chân có thể cho đầu ra PWM)
* Số chân vào tương tự: 6
* Dòng điện trong mỗi I/O: 20mA
* Dòng điện chân nguồn 3.3V: 50mA
  + 1. **Cảm biến độ ẩm, nhiệt độ (DHT11):**

DHT11 là cảm biến nhiệt độ, độ ẩm rất thông dụng hiện nay vì chi phí rẻ và rất dễ lấy dữ liệu thông qua giao tiếp 1-wire ( sử dụng 1 dây data để truyền dữ liệu). Cảm biến được tích hợp bộ tiền xử lý tín hiệu giúp dữ liệu nhận về được chính xác mà không cần phải qua bất kỳ tính toán nào.

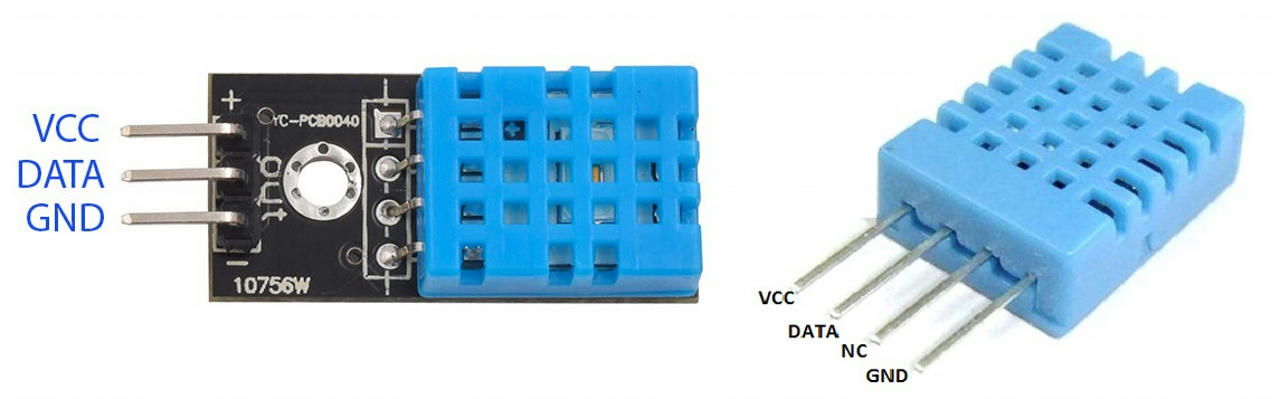
Cảm biến nhiệt độ và độ ẩm DHT11 bao gồm 3 thành phần chính. Một cảm biến độ ẩm loại điện trở, một điện trở nhiệt NTC (Negative Temperature Coefficient – hệ số nhiệt âm) để đo nhiệt độ và một vi điều khiển 8 bit có nhiệm vụ chuyển đổi tín hiệu tương tự từ cảm biến và gửi ra tín hiệu số duy nhất.

Tín hiệu số này có thể được đọc bởi bất kỳ vi điều khiển hoặc bộ vi xử lý nào để phân tích.

Cảm biến DHT11 có thể đo giá trị độ ẩm trong khoảng 20 – 90% Độ ẩm tương đối (RH) và nhiệt độ trong khoảng 0 – 50 °C. Khoảng thời gian lấy mẫu của cảm biến là 1 giây.

* **Sơ đồ chân:**

Cảm biến DHT11 gồm 2 chân cấp nguồn (VCC và GND), và 1 chân tín hiệu (DATA). Hiện nay, thông dụng ngoài thị trường có hai loại đóng gói cho DHT11: 3 chân và 4 chân như hình bên dưới



*Hình 2.3: Module cảm biến nhiệt độ và độ ẩm DHT11*

* **Thông số kỹ thuật:**
* Điện áp hoạt động: 3-5VDC.
* Dải nhiệt độ đo: 0-50°C. Sai số ±2°C.
* Dải độ ẩm đo: 20-80%. Sai số ±5%.
* Tần số lấy mẫu: 1Hz (mỗi giây một lần).
  + 1. **Module CC1101:**

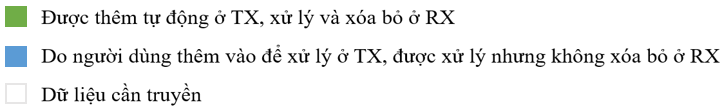
CC1101 là bộ thu phát Sub -1GHz giá rẻ dành cho các ứng dụng không dây yêu cầu rất ít năng lượng. Mạch được thiết kế để hoạt động trong các dải tần là 315, 433, 868 và 915 MHz và cho phép người dùng truyền nhận dữ liệu trong khoảng cách 500m, tốc độ truyền lên đến 600kbps.

**/**

*Hình 2.4: Module CC1101*

* **Thông số kỹ thuật:**
* Điện áp hoạt động: 1.8V ~ 3.6V
* Độ nhạy của máy thu: -110dBm với tốc độ truyền 1200
* Công suất phát tối đa: 10mW (+ 10dBm)
* Tốc độ truyền tối đa: 500kbps
* Băng tần 315Mhz, 433Mhz, 868Mhz, 915Mhz
* Tốc độ truyền 500kbps
* **Mục đích:**
* Truyền tín hiệu từ nơi phát đến nơi thu.
* Khoảng cách xa, chính xác, ít sai số.
  1. **Quá trình truyền dữ liệu giữa 2 module CC1101:**
     1. **Xử lý dữ liệu ở máy phát:**
* Một gói dữ liệu truyền đi có dạng:
* Phần mở đầu
* Đồng bộ ký tự
* Chiều dài gói data tùy chọn
* Địa chỉ gói data tùy chọn
* Data
* 2 byte CRC tùy chọn

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Phần mở đầu  1010…1010  (8 x n bit) | Đồng bộ ký tự  (16/32 bits) | Chiều dài  (8 bits) | Địa chỉ  (8 bits) | Data | CRC  (16 bits) |



Số byte mở đầu được lập trình với giá trị MDMCFG1.NUM\_PREAMBLE.

#define CC1101\_MDMCFG1 0x13 // Set 3 byte phần mở đầu

Đồng bộ ký tự là một giá trị hai byte được đặt trong thanh ghi SYNC1 và SYNC0.

#define CC1101\_SYNC1 0x04 // Sync word, high INT8U

#define CC1101\_SYNC0 0x05 // Sync word, low INT8U

Byte chiều dài gói tin phải được ghi vào TX FIFO. Giá trị byte độ dài bằng chiều dài của gói dữ liệu (bao gồm cả byte địa chỉ tùy chọn).

Khi bật TX, bộ điều chế sẽ bắt đầu truyền phần mở đầu. Khi phần mở đầu truyền xong, bộ điều chế sẽ truyền tiếp byte đồng bộ ký tự và sau đó là dữ liệu từ TX FIFO nếu có sẵn dữ liệu. Nếu TX FIFO trống, bộ điều chế sẽ tiếp tục gửi các byte mở đầu cho đến khi byte đầu tiên được ghi vào TX FIFO. Sau đó, bộ điều biến sẽ gửi từ đồng bộ và sau đó là các byte dữ liệu.

Dữ liệu phía sau byte đồng bộ ký tự được xáo trộn và mã hóa bởi Interleaver và FEC trước khi được điều chế.

* + 1. **Xử lý dữ liệu ở máy thu:**

Trong chế độ nhận, bộ giải điều chế và xử lý gói sẽ tìm kiếm các byte mở đầu và đồng bộ ký tự hợp lệ. Nếu tìm được sẽ bắt đầu nhận byte dữ liệu đầu tiên. Bộ xen kẽ sẽ khử xáo trộn các bit trước khi các quá trình xử lý khác được thực hiện.

Bộ giải mã FEC sẽ bắt đầu giải mã byte dữ liệu đầu tiên và bộ xử lý gói sẽ lưu trữ giá trị này dưới dạng độ dài gói. Tiếp theo là byte địa chỉ, bộ xử lý gói kiểm tra địa chỉ và chỉ tiếp tục nhận nếu địa chỉ trùng khớp.

Khi kết thúc tải trọng, trình xử lý gói sẽ tùy chọn ghi thêm hai byte trạng thái gói chứa trạng thái CRC, chỉ báo chất lượng liên kết và giá trị RSSI.

* + 1. **Định dạng điều chế tín hiệu:**

CC1101 hỗ trợ các định dạng điều chế 2-FSK, GFSK và MSK.

CC1101 cũng có khả năng sử dụng bộ lọc Gaussian để định hình 2-FSK (GFSK). Tính năng định hình phổ này cải thiện công suất kênh lân cận (ACP) và băng thông bị chiếm dụng. Trong các hệ thống 2-FSK phù hợp với sự thay đổi tần số đột ngột, phổ tần vốn rộng. Bằng cách làm cho sự dịch chuyển tần số trở nên 'mềm mại hơn', phổ có thể được thu hẹp đáng kể. Do đó, tốc độ dữ liệu cao hơn có thể được truyền trong cùng một băng thông bằng cách sử dụng GFSK.

Định dạng điều chế được đặt trong thanh ghi MDMCFG2.MOD\_FORMAT.

#define CC1101\_MDMCFG2 0x12 // Định dạng điều chế GFSK

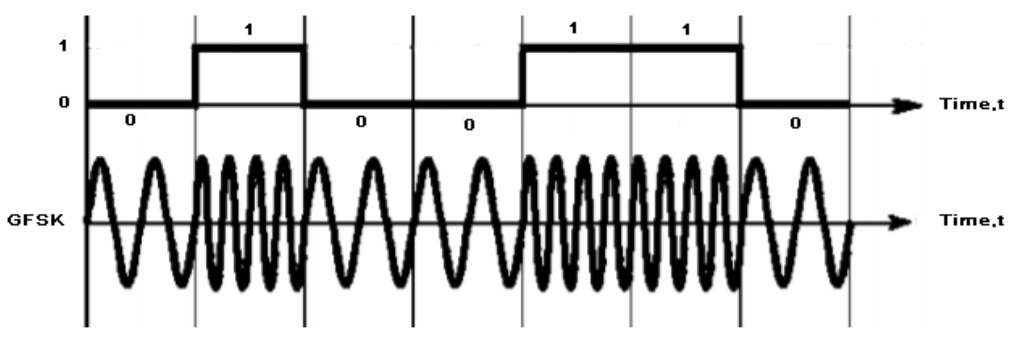
Độ lệch tần số được lập trình với các giá trị DEVIATION\_M và DEVIATION\_E trong thanh ghi DEVIATN.

#define CC1101\_DEVIATN 0x15 // DEVIATION\_M = 5

// DEVIATION\_E = 1

* **Kỹ thuật điều chế GFSK:**

Điều chế FSK ấn định các tần số khác nhau cho sóng mang tùy thuộc vào bit được truyền đi. Do đó, khi một bit 0 được truyền đi, sóng mang giả định một tần số tương ứng với một bit 0 trong thời gian của một bit. Khi bit 1 được truyền đi, tần số sóng mang được thay đổi thành giá trị tương ứng với bit 1 và tương tự, vẫn giữ nguyên tần số này trong suốt thời gian của một bit, như thể hiện trong hình sau:



*Hình 2.5: Điều chế FSK*

Trong điều chế Gaussian FSK (GFSK) dữ liệu được mã hóa dưới dạng các biến thể của tần số trong sóng mang theo cách tương tự như FSK. Do đó, bộ điều chế được sử dụng có thể giống bộ điều chế được sử dụng cho điều chế FSK. Tuy nhiên, các xung đi qua một bộ lọc Gaussian trước khi đi vào bộ điều biến xung để giảm độ rộng phổ của cùng một bộ lọc Gaussian.



*Hình 2.6 : Gaussian Filter*

Bộ lọc này có ưu điểm là giảm công suất dải biên, giảm nhiễu với các kênh lân cận.

Độ lệch tần số được tính theo công thức:

*= DEVIATION\_M)\**

Như vậy, giá trị của độ lệch là ± 5,157kHz, với tần số tinh thể là 26,0 MHz.

Bit 0: - deviation

Bit 1: + deviation

* + 1. **Sửa lỗi:**

Module CC1101 hỗ trợ tích hợp sửa lỗi chuyển tiếp và xen kẽ.

* + - 1. **Sửa lỗi chuyển tiếp (FEC):**

Để bật tùy chọn này, phải đặt thanh ghi MDMCFG1.FEC\_EN = 1.

Sửa lỗi chuyển tiếp (FEC) là một quá trình trong đó kết quả của các thuật toán được gửi dưới dạng thông tin bổ sung cùng với dữ liệu từ đầu truyền. Bằng cách lặp lại các thuật toán tương tự ở đầu xa, máy thu có khả năng phát hiện lỗi ở mức bit đơn và sửa chúng (lỗi có thể sửa được) mà không cần phải truyền lại dữ liệu.

Lược đồ FEC được áp dụng cho CC1101 là mã hóa tích chập. Bộ mã tích chập là mã tỷ lệ 1/2 với độ dài giới hạn là m = 4. Mã hóa một bit đầu vào và tạo ra hai bit đầu ra. Do đó, tốc độ dữ liệu hiệu quả giảm đi một nửa. Để truyền ở cùng một dải dữ liệu hiệu quả khi sử dụng FEC, cần phải sử dụng băng thông gấp đôi. Điều này sẽ yêu cầu băng thông máy thu cao hơn, và do đó làm giảm độ nhạy.

FEC được sử dụng trên trường dữ liệu và từ CRC để giảm tỷ lệ lỗi bit tổng khi hoạt động gần giới hạn độ nhạy. Dữ liệu bổ xung được thêm vào dữ liệu được truyền theo cách mà người nhận có thể khôi phục dữ liệu ban đầu khi có một số lỗi bit. Việc sử dụng FEC cho phép tiếp nhận chính xác hơn ở SNR thấp hơn, do đó mở rộng phạm vi truyền thông nếu băng thông máy thu không đổi.Ngoài ra, đối với SNR nhất định, việc sử dụng FEC làm giảm tỷ lệ lỗi gói (PER). Vì tỷ lệ lỗi gói (PER) có liên quan đến BER bởi:

PER = (1-(1-BER))\*packet\_length

Cuối cùng, trong môi trường vô tuyến ISM thực tế, các hiện tượng nhất thời và thay đổi theo thời gian sẽ tạo ra các lỗi không thường xuyên ngay cả trong các điều kiện thu sóng tốt. FEC sẽ che giấu các lỗi như vậy và kết hợp với việc xen kẽ các dữ liệu được mã hóa, thậm chí sửa các lỗi tiếp nhận bị lỗi trong thời gian tương đối dài (lỗi liên tục).

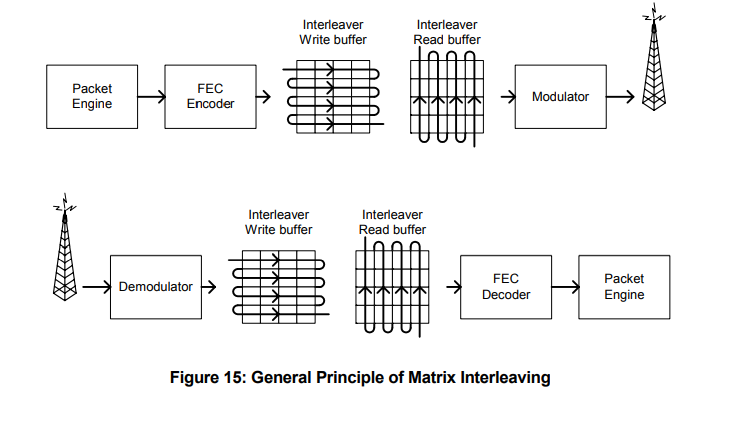
* + - 1. **Sửa lỗi xen kẽ:**

Dữ liệu nhận được qua các kênh vô tuyến sẽ thường gặp lỗi liên tục do nhiễu và cường độ tín hiệu thay đổi theo thời gian. Để giảm các lỗi kéo dài nhiều bit, phép xen kẽ được sử dụng khi FEC được kích hoạt. Sau khi loại bỏ xen kẽ, một khoảng lỗi liên tục trong luồng đã nhận sẽ trở thành các lỗi đơn lẻ được phân tán.

Trong bộ truyền, các bit dữ liệu từ bộ mã chập tốc độ ½ được ghi vào các hàng của ma trận, trong khi chuỗi bit cần truyền được đọc từ các cột của ma trận.Ngược lại, trong máy thu, các ký hiệu nhận được được ghi vào các cột của ma trận, trong khi dữ liệu được truyền vào bộ giải mã tích hợp được đọc từ các hàng của ma trận.

Bộ đệm xen kẽ và loại bỏ xen kẽ trên CC1101 là ma trận 4 x 4.

Sơ đồ :

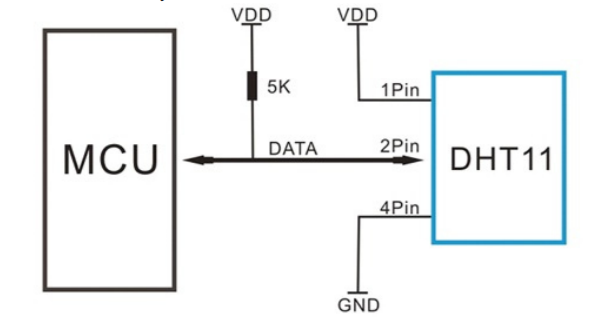


*Hình 2.7 : Nguyên tắc chung của xen kẽ ma trận*

Khi sửa lỗi chuyển tiếp và xen kẽ được sử dụng, cần ít nhất một byte bổ sung. Lượng dữ liệu được truyền phải là bội số của kích thước của bộ đệm xen kẽ (hai byte). Do đó, phần cứng điều khiển gói tự động chèn thêm một hoặc hai byte vào cuối gói, để tổng độ dài của dữ liệu được xen kẽ là một số chẵn. Những byte thừa này sẽ bị loại bỏ trước khi gói nhận được đi vào RX FIFO. Khi sử dụng FEC và xen kẽ, tải trọng dữ liệu tối thiểu là 2 byte.

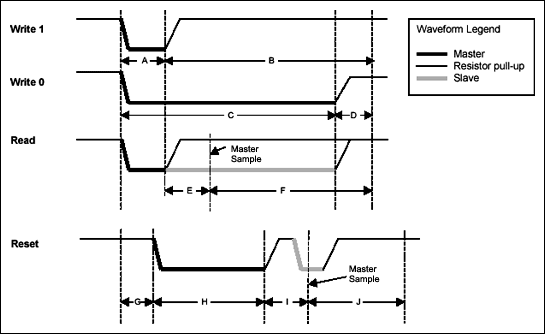
* 1. **Giao tiếp với vi điều khiển:**
     1. **DHT11:**
        1. **Chuẩn giao tiếp 1-wire:**

Chuẩn giao tiếp 1-Wire  dùng một dây để truyền nhận nên có tốc độ thấp . Chủ yếu sử dụng cho việc thu thập dữ liệu, truyền nhận dữ liệu thời tiết, nhiệt độ, công việc không yêu cầu tốc độ cao.



*Hình 2.8 : Sơ đồ nối chân của chuẩn giao tiếp 1-wire*

* **Cơ sở truyền nhận :**
* Write 1 : truyền đi bit 1 : Master kéo xuống 0 một khoảng thời gian A (us) rồi trả về 1 khoảng B (B>A).
* Write 0 : truyền đi bit 0 : Master kéo xuống 0 một khoảng thời gian C (us) rồi trả về 1 khoảng D (D<C).
* Read : Đọc một Bit : Master kéo xuống 0 khoảng A rồi trả về 1 . delay khoảng E rồi đọc giá trị slave gửi về . delay F.
* Restart : Chuẩn bị giao tiếp . Master ké0 xuống 0 một khoảng H rồi nhả lên mức 1 sau đó cấu hình Master là chân In delay I (us) rồi đọc giá trị slave trả về . Nếu =0 thì cho phép giao tiếp . =1 đường truyền lỗi hoặc slave đang bận.



*Hình 2.9 : Chuẩn giao tiếp 1-wire*

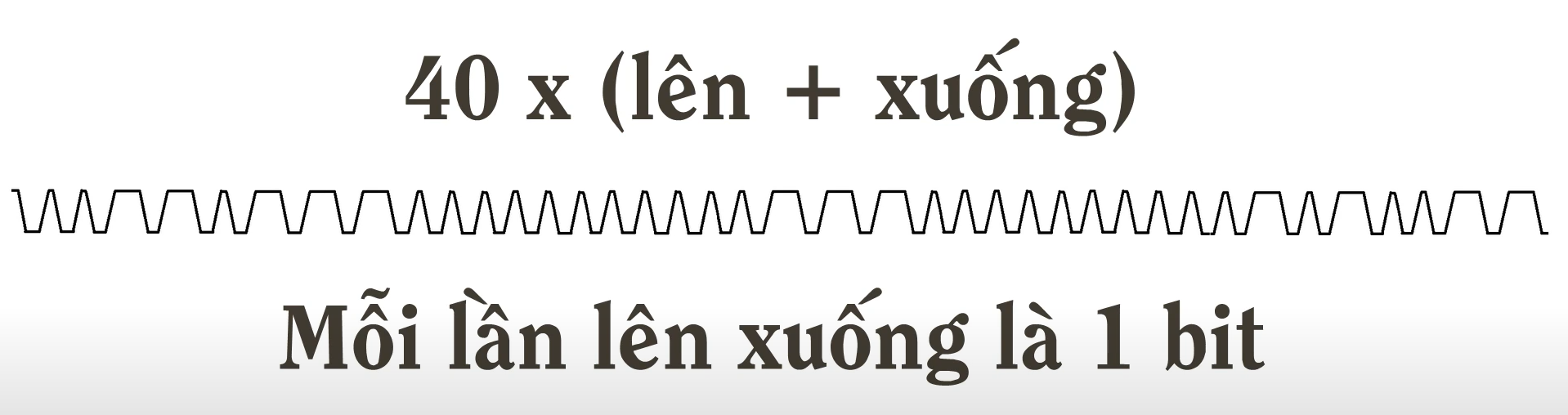
* + - 1. **DHT11 giao tiếp với Arduino:**

DHT11 giao tiếp với Arduino thông qua giao tiếp 1-wire.

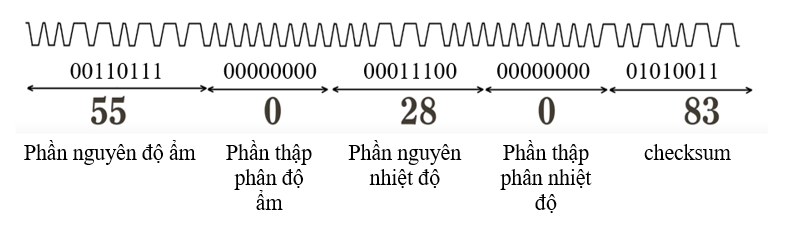
Dữ liệu truyền về từ cảm biến DHT11 bao gồm 40 bit và định dạng như sau:

8 bit biểu thị phần nguyên của độ ẩm + 8 bit biểu thị phần thập phân của độ ẩm + 8 bit biểu thị phần nguyên của nhiệt độ + 8 bit biểu thị phần thập phân của nhiệt độ + 8 bit checksum.

* Ví dụ: Xem xét dữ liệu nhận được từ cảm biến DHT11



Dữ liệu này có thể được phân tách dựa trên cấu trúc nêu trên như sau:



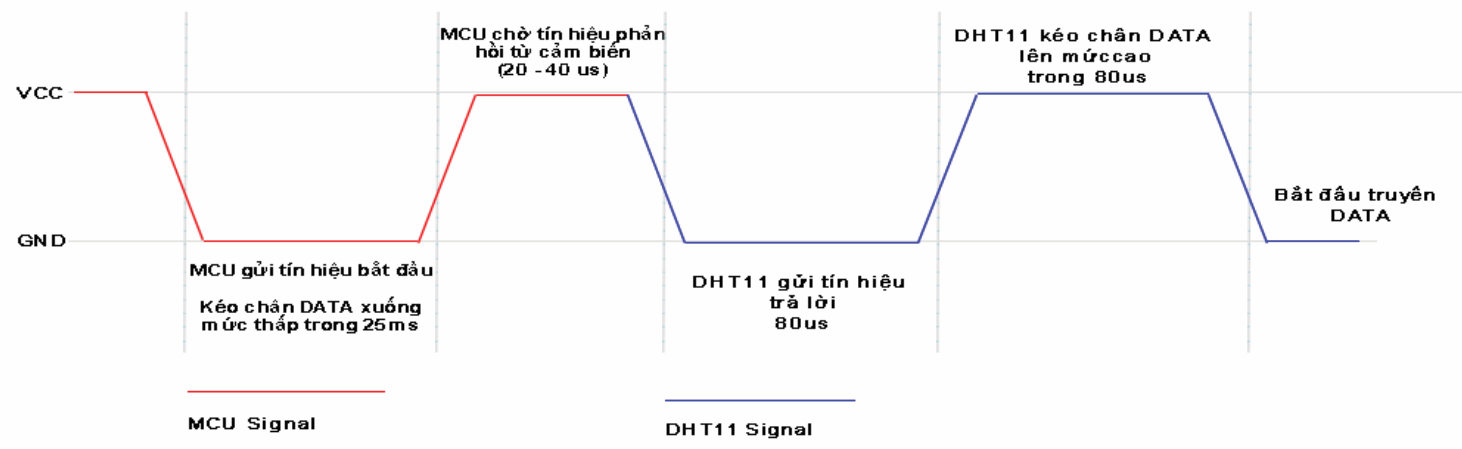
Để kiểm tra xem liệu dữ liệu nhận được có chính xác hay không, chúng ta cần thực hiện một phép tính nhỏ. Cộng tất cả các giá trị nguyên và thập phân của độ ẩm  và nhiệt độ và kiểm tra xem tổng có bằng giá trị checksum hay không, tức là dữ liệu 8 bit cuối cùng.

00110111 + 00000000 + 00011100 + 00000000 = 01010011

* **Nguyên lý hoạt động:**

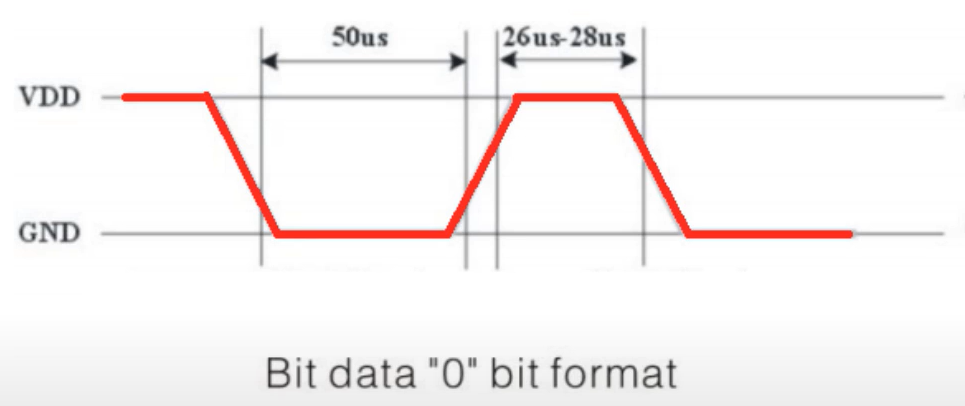
Để có thể giao tiếp với DHT11 theo chuẩn 1 chân vi xử lý thực hiện theo 2 bước:

* Gửi tin hiệu muốn đo (Start) tới DHT11, sau đó DHT11 xác nhận lại.
* Khi đã giao tiếp được với DHT11, Cảm biến sẽ gửi lại 5 byte dữ liệu và nhiệt độ đo được.
* Bước 1: gửi tín hiệu Start
* MCU thiết lập chân DATA là Output, kéo chân DATA xuống 0 trong khoảng thời gian >18ms. Khi đó DHT11 sẽ hiểu MCU muốn đo giá trị nhiệt độ và độ ẩm.
* MCU đưa chân DATA lên 1, sau đó thiết lập lại là chân đầu vào. Sau khoảng 20 - 40 us, DHT11 sẽ kéo chân DATA xuống thấp. Nếu >40us mà chân DATA ko được kéo xuống thấp nghĩa là ko giao tiếp được với DHT11.
* Chân DATA sẽ ở mức thấp 80us sau đó nó được DHT11 kéo nên cao trong 80us. Bằng việc giám sát chân DATA, MCU có thể biết được có giao tiếp được với DHT11 ko. Nếu tín hiệu đo được DHT11 lên cao, khi đó hoàn thiện quá trình giao tiếp của MCU với DHT.



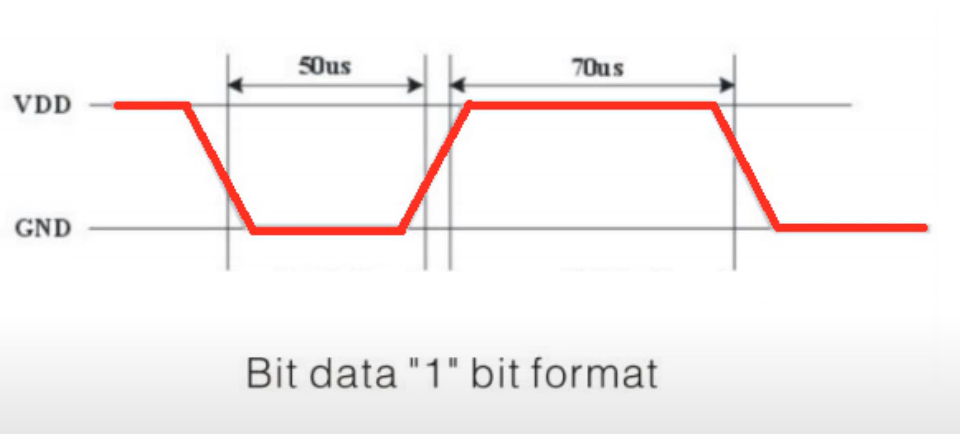
*Hình 2.10 : Tín hiệu điều khiển khởi động DHT11*

* Bước 2: đọc giá trị trên DHT11
* DHT11 sẽ trả giá trị nhiệt độ và độ ẩm về dưới dạng 5 byte.
* Trong đó:
* Byte 1: giá trị phần nguyên của độ ẩm (RH%)
* Byte 2: giá trị phần thập phân của độ ẩm (RH%)
* Byte 3: giá trị phần nguyên của nhiệt độ (TC)
* Byte 4 : giá trị phần thập phân của nhiệt độ (TC)
* Byte 5 : kiểm tra tổng.
* Nếu Byte 5 = (8 bit) (Byte1 +Byte2 +Byte3 + Byte4) thì giá trị độ ẩm và nhiệt độ là chính xác, nếu sai thì kết quả đo không có nghĩa.
* Đọc dữ liệu: Sau khi giao tiếp được với DHT11, DHT11 sẽ gửi liên tiếp 40 bit 0 hoặc 1 về MCU, tương ứng chia thành 5 byte kết quả của Nhiệt độ và độ ẩm.
* Sau khi tín hiệu được đưa về 0, ta đợi chân DATA của MCU được DHT11 kéo lên 1.
* Nếu chân DATA là 1 trong khoảng 26 - 28 us thì là bit 0.



*Hình 2.11 : Tín hiệu Data từ DHT11 - Bit 0*

* Nếu chân DATA là 1 trong khoảng 70 us là bit 1.



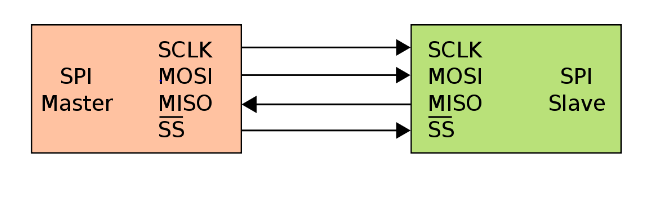
*Hình 2.12 : Tín hiệu Data từ DHT11 - Bit 1*

Do đó trong lập trình ta bắt sườn lên của chân DATA, sau đó delay 50us. Nếu giá trị đo được là 0 thì ta đọc được bit 0, nếu giá trị đo được là 1 thì giá trị đo được là 1. Cứ như thế ta đọc các bit tiếp theo.

* + 1. **Module CC1101:**
       1. **Chuẩn giao tiếp SPI:**

SPI (Serial Peripheral Interface) hoặc giao tiếp ngoại vi nối tiếp là một chuẩn đồng bộ nối tiếp để truyền dữ liệu ở chế độ song công toàn phần (full – duplex) tức trong cùng một thời điểm có thể xảy ra đồng thời quá trình truyền và nhận. SPI là một loại giao thức kiểu Master – Slave cung cấp một giao diện chi phí đơn giản và chi phí thấp giữa vi điều khiển và các thiết bị ngoại vi của nó.

SPI đôi khi được gọi là chuẩn truyền thông “4 dây” vì có 4 đường giao tiếp trong chuẩn này đó là SCK (Serial Clock), MISO (Master Input Slave Output), MOSI (Master Ouput Slave Input) và SS (Slave Select).



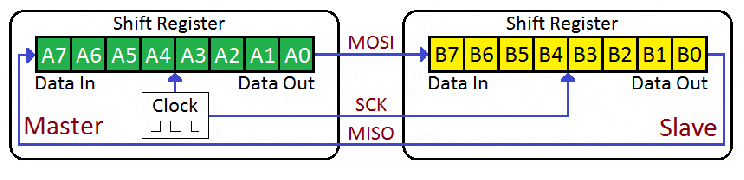
Giao tiếp SPI thường được sử dụng để giao tiếp vi xử lý hoặc vi điều khiển với bộ nhớ như EEPROM, RTC (Đồng hồ thời gian thực), ADC (Analog  to Digital Converter – Bộ chuyển đổi tương tự sang số), DAC (Digital-to-Analog Converter – Bộ chuyển đổi số sang tương tự), thiết bị hiển thị như màn hình LCD, IC âm thanh, các loại [cảm biến](https://dientuadenz.com/cac-loai-cam-bien/) như nhiệt độ và áp suất, thẻ nhớ như MMC hoặc thẻ SD hoặc thậm chí các bộ vi điều khiển khác.

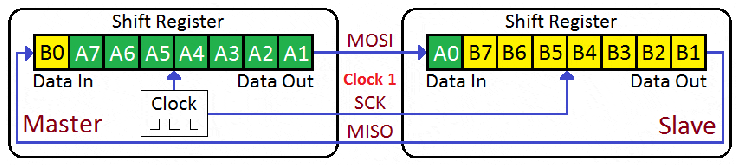
1. **CƠ CHẾ HOẠT ĐỘNG:**

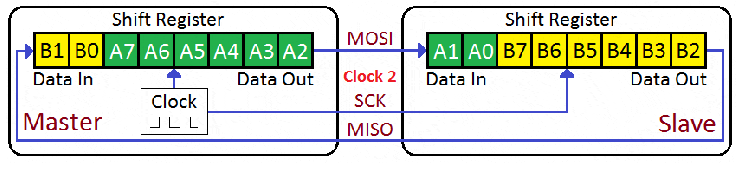
Mỗi thiết bị Master và Slave đều có một thanh ghi dịch 8 bits (Shift Register), một bộ tạo xung nhịp (Clock Generator).

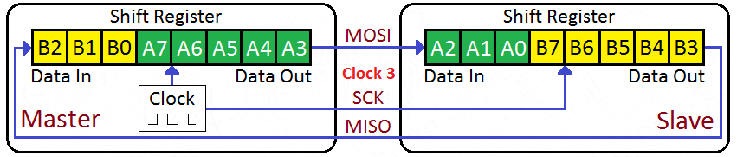
Khi Master truyền dữ liệu, Master truyền đi 8 bits dữ liệu vào thanh ghi dịch của nó, sau đó 8 bits dữ liệu được truyền theo đường tín hiệu MOSI sang thiết bị Slave. Tương tự khi Slave truyền dữ liệu, các bits trên thanh ghi dịch của Slave truyền đến Master thông qua đường tín hiệu MISO.

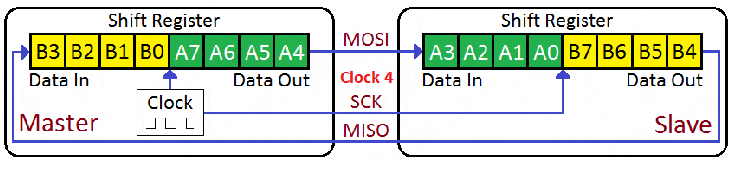
Bằng cách này, dữ liệu của hai thanh ghi sẽ được trao đổi với nhau. Việc đọc và ghi dữ liệu vào Slave diễn ra cùng một lúc nên tốc độ trao đổi dữ liệu diễn ra rất nhanh. Do đó, giao thức SPI là một giao thức rất có hiệu quả.

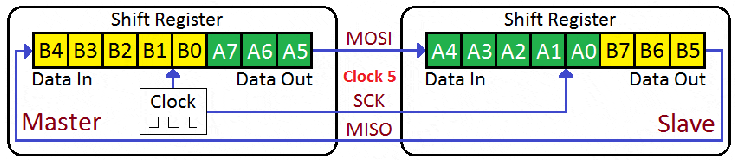


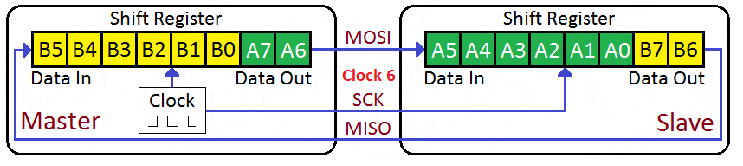


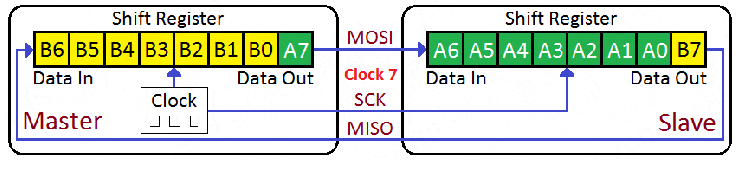


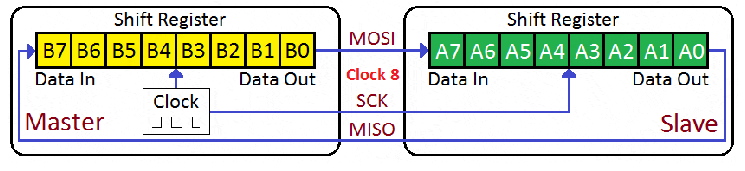






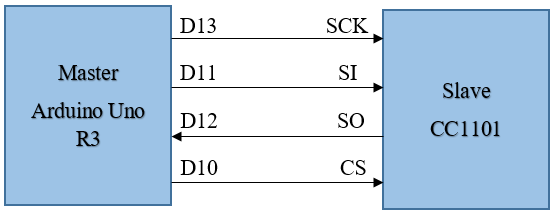






* + - 1. **CC1101 Giao tiếp với Arduino:**

CC1101 được cấu hình thông qua giao diện tương thích SPI 4 dây đơn giản (SI, SO, SCLK và CSn) trong đó CC1101 là Slave.



*Hình 2.13 : Chuẩn giao tiếp SPI*

Dữ liệu được truyền qua lại giữa 2 đường SO và SI. Điều này chỉ thực hiện được khi tín hiệu chân CS được thiết lập ở mức thấp LOW. 

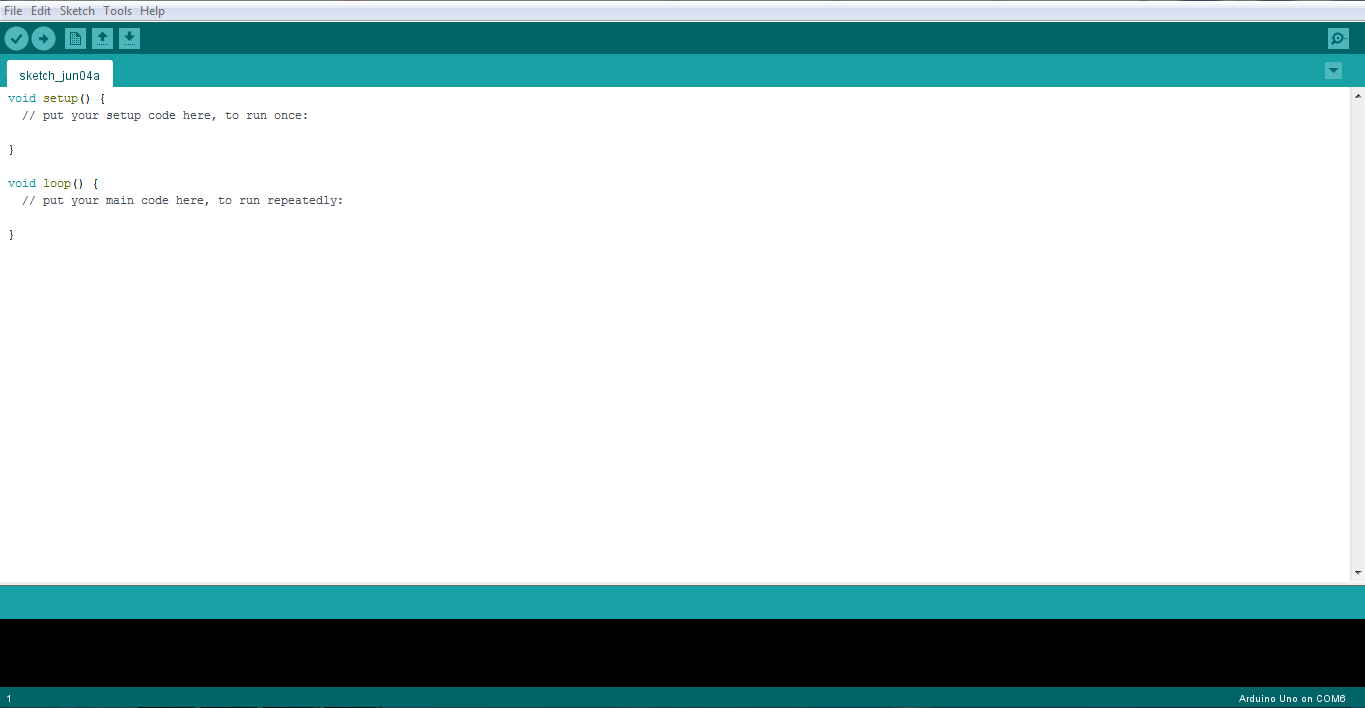
* 1. **Các phần mềm sử dụng:**
     1. **Arduino IDE:**
        1. **Giới thiệu:**

Arduino IDE (Integrated Development Environment) là một phần mềm mã nguồn mở chủ yếu được sử dụng để viết và biên dịch mã vào module Arduino. Đây là một phần mềm Arduino chính thức, nó giúp cho việc lập trình và biên dịch mã trở nên dễ dàng. Arduino IDE có các phiên bản cho các hệ điều hành như MAC, Windows, Linux và chạy trên nền tảng Java đi kèm với các chức năng và lệnh có sẵn đóng vai trò quan trọng để gỡ lỗi, chỉnh sửa và biên dịch mã trong môi trường.

Môi trường IDE chủ yếu chứa hai phần cơ bản: Trình chỉnh sửa và Trình biên dịch, phần đầu sử dụng để viết mã được yêu cầu và phần sau được sử dụng để biên dịch và tải mã lên module Arduino.

Arduino IDE hỗ trợ cả ngôn ngữ C và C ++.

* + - 1. **Giao diện chương trình:**



*Hình 2.14 : Giao diện làm việc của Arduino IDE*

* + 1. **Microsoft Visual Studio 2019:**
       1. **Giới thiệu:**

Microsoft Visual Studio hay gọi tắt là Visual Studio là một phần mềm lập trình hệ thống được sản xuất trực tiếp từ [Microsoft](https://www.microsoft.com/vi-vn/). Từ khi ra đời đến nay, Visual Studio đã có rất nhiều các phiên bản sử dụng khác nhau. Điều đó, giúp cho người dùng có thể lựa chọn được phiên bản tương thích với dòng máy của mình cũng như cấu hình sử dụng phù hợp nhất.

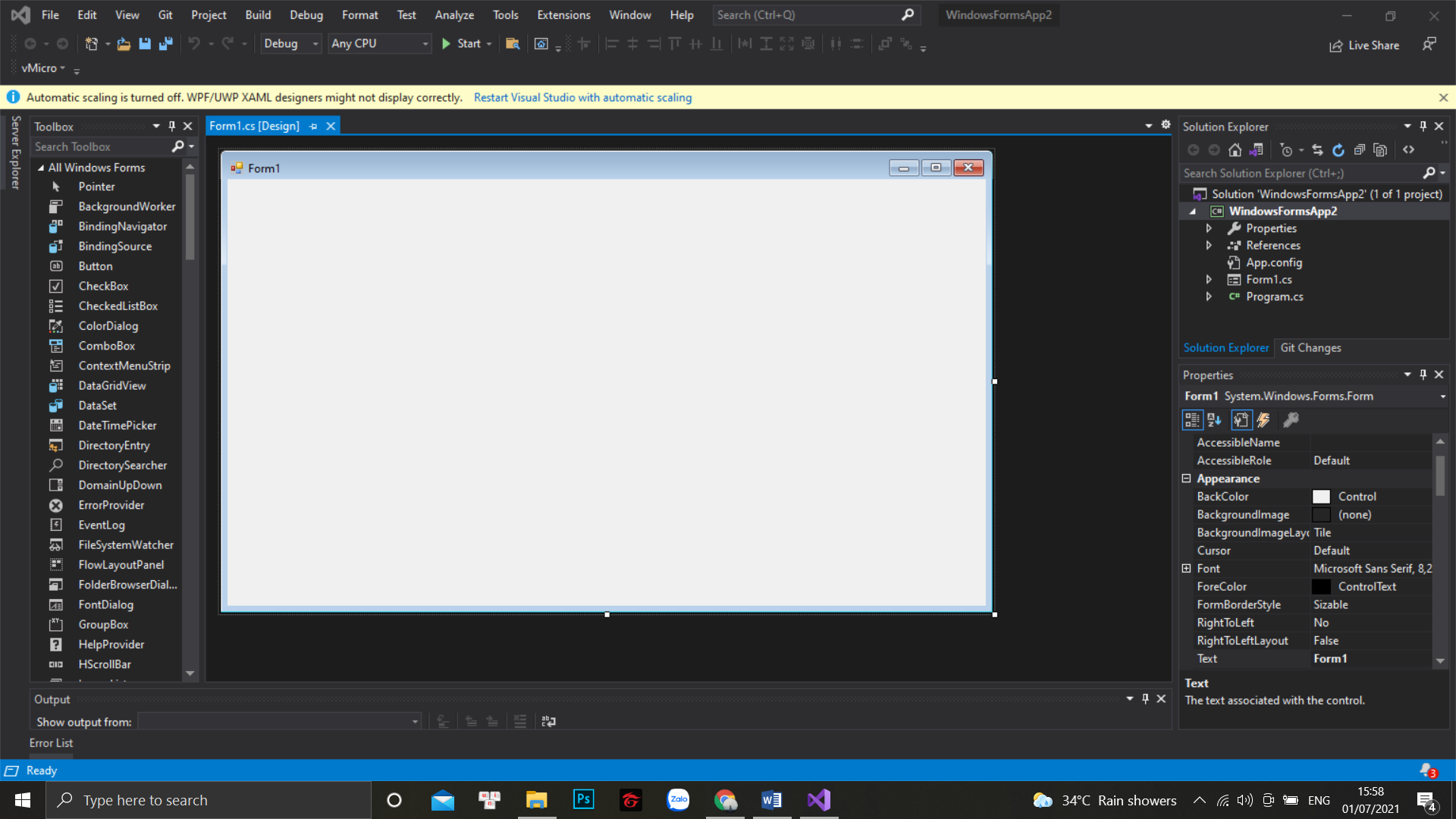
Microsoft Visual Studio hay gọi tắt là Visual Studio đây là tập công cụ hoàn chỉnh dùng để xây dựng ứng dụng Web (ASP.NET Web Applications), dịch vụ XML, ứng dụng để bàn (Desktop application), ứng dụng màn hình với bàn phím (Console Applications) và ứng dụng trên điện thoại di động (Mobile Applications).

Các ngôn ngữ lập trình dùng Visual Studio để phát triển ứng dụng là Visual basic, Visual C++, Visual C# và Visual J#. Cả 4 ngôn ngữ lập trình chính trên đều sử dụng chung một IDE (Integrated Development Environment), nơi cho phép chúng ta chia sẻ các tiện ích và công cụ nhằm tạo nên giải pháp tích hợp.

* + - 1. **Ứng dụng Windows Form:**

Windows Forms hay còn gọi tắt là Winform là thuật ngữ mô tả một ứng dụng được viết dùng .NET FrameWork và có giao diện người dùng Windows Forms (màn hình windows). Mỗi màn hình windows cung cấp một giao diện giúp người dùng giao tiếp với ứng dụng.

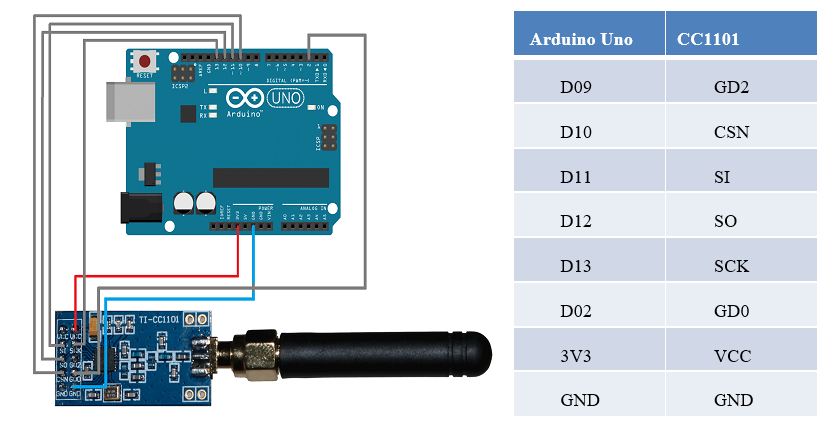
Windows Form bao gồm các phần cơ bản là Form, Button, Panel, Textbox, ComboBox, RadioButton,… Người dùng tương tác trực tiếp với hệ thống thông qua giao diện Winform bằng bàn phím và chuột.



*Hình 2.15 : Giao diện Winform.*

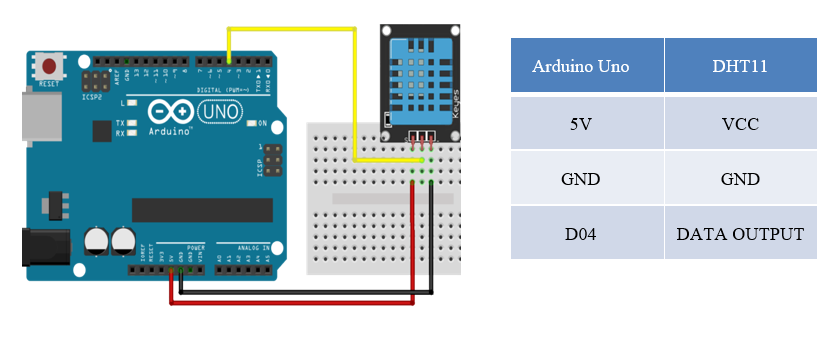
# CHƯƠNG 3: THIẾT KẾ VÀ XÂY DỰNG HỆ THỐNG

* 1. **Sơ đồ phần cứng:**
     1. **Sơ đồ mạch phần cứng Arduino** ⭢ **CC1101:**

****

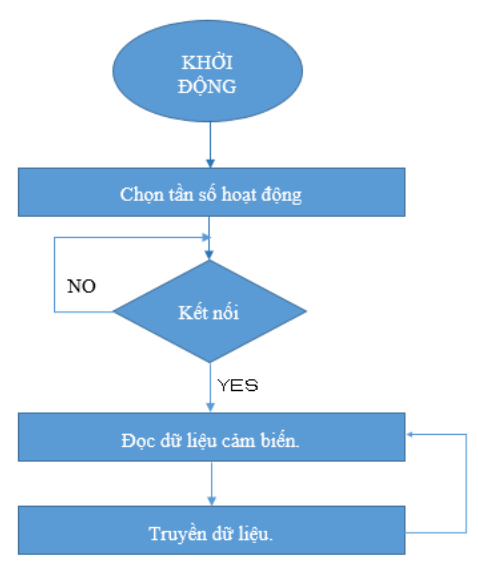
*Hình 3.1: Sơ đồ nối chân Arduino với CC1101*

* + 1. **Sơ đồ mạch phần cứng Arduino** ⭢ **DHT11:**

****

*Hình 3.2: Sơ đồ nối chân Arduino với DHT11*

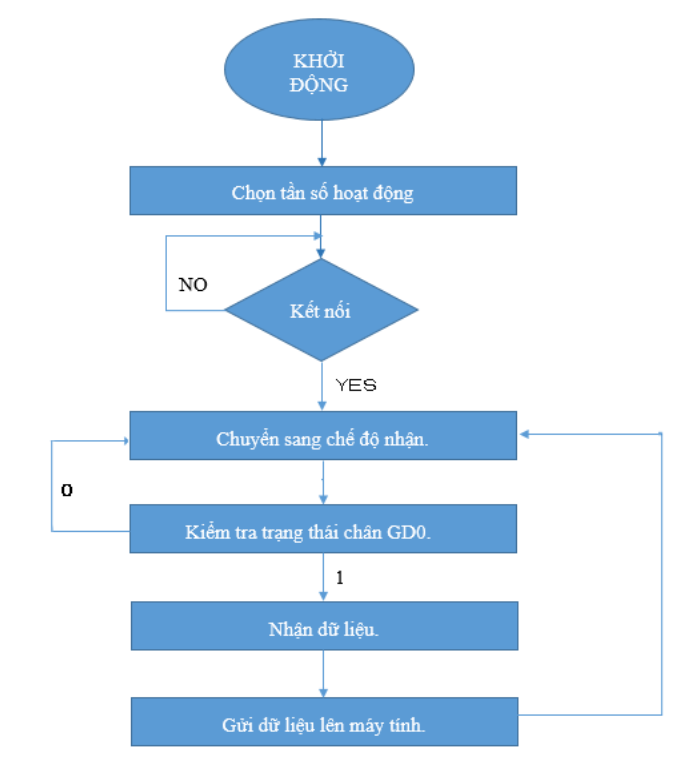
* 1. **Lưu đồ giải thuật:**
     1. **Lưu đồ giải thuật máy phát:**



*Hình 3.3 : Lưu đồ giải thuật ở máy phát*

Giải thích: Khi khởi động, hệ thống sẽ thiết lập các thông số cần thiết (tần số, kênh,…) và khởi động Module. Sau khi kiểm tra kết nối, chương trình sẽ bắt đầu đọc dữ liệu từ các cảm biến và lưu giá trị, đóng gói thành 1 gói tin và được truyền đi. Khi truyền thành công 1 gói tin, hệ thống tiếp tục đọc giá trị nhiệt độ và độ ẩm mới từ cảm biến.

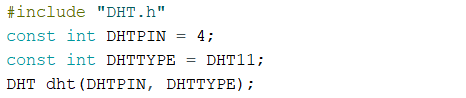
* + 1. **Lưu đồ giải thuật máy thu:**



*Hình 3.4 : Lưu đồ giải thuật ở máy thu*

Giải thích: Khi khởi động, hệ thống sẽ thiết lập các thông số cần thiết (tần số, kênh,…). Sau khi kiểm tra kết nối, hệ thống cài đặt Module CC1101 ở chế độ thu tín hiệu. Tiếp theo, chương trình kiểm tra việc nhận các gói tin và bắt đầu nhận. Cuối cùng các gói tin sẽ được nhận, gửi dữ liệu lên máy tính để hiển thị và bắt đầu nhận gói tin mới.

* 1. **Lập trình hệ thống trên Arduino:**
     1. **Thu thập dữ liệu DHT11:**
* Khai báo thư viện và các biến cho cảm biến DHT11:

****

* Thư viện *DHT.h* hỗ trợ đọc dữ liệu trên cả DHT11 và DHT22. Ở đây ta sử dụng DHT11 nên khai báo biến DHTTYPE = DHT11.
* Chân nhận dữ liệu cảm biến là D04.
* Đọc dữ liệu nhiệt độ và độ ẩm:



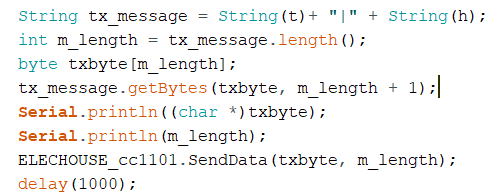
* Kiểu dữ liệu của độ ẩm và nhiệt độ là float.
* Giá trị trả về nằm trong 2 hàm *dht.readHumidity()* và *dth.readTemperature()* tương ứng với đó là giá trị độ ẩm và nhiệt độ.
  + 1. **Gửi dữ liệu ở máy phát:**
* Khai báo:
* Thư viện *ELECHOUSE\_CC1101.h*

****

* Trong phần setup khai báo tần số sóng mang 433MHz. Có thể thay đổi tần số thành 866MHz (F\_866) hoặc 915MHz (F\_915).

****

* Xử lý dữ liệu và truyền đi:



* Đầu tiên phải chuyển dữ liệu thu được thành chuỗi ký tự. Chuỗi truyền đi có dạng <nhiệt độ>|<độ ẩm>.
* Lấy chiều dài của chuỗi để thêm vào hàm truyền.
* Chuyển chuỗi thành mảng kiểu Byte để lấy mã ASCII của các ký tự trong chuỗi.
* Cuối cùng là truyền dữ liệu với hàm:

ELECHOUSE\_cc1101.SendData(txbyte, m\_length);

* + 1. **Nhận dữ liệu ở máy thu:**
* Khai báo:
* Khai báo thư viện *ELECHOUSE\_CC1101.h .*

****

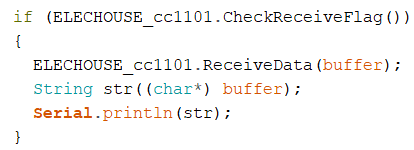
* Trong phần setup:
* Khai báo tốc độ Baud là 9600.
* Khai báo tần số sóng mang 433MHz.



* Nhận dữ liệu:
* Để nhận dữ liệu ta phải đặt CC1101 vào chế độ nhận bằng hàm:



* Quá trình nhận dữ liệu:



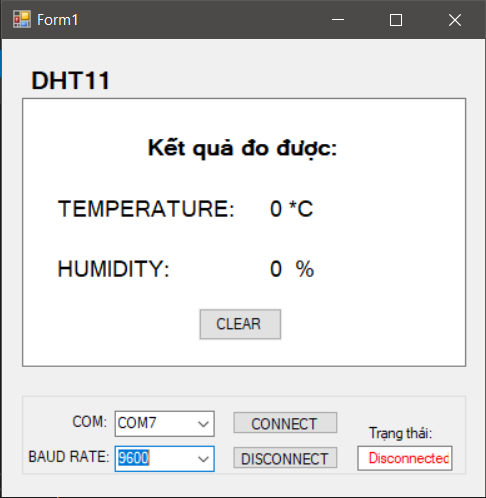
* CC1101 sẽ liên tục kiểm tra trạng thái chân GD0 bằng hàm:

ELECHOUSE\_cc1101.CheckReceiveFlag()

Nếu có dữ liệu ở RX FIFO thì chân GD0 sẽ được kéo lên mức cao.

* Dữ liệu nhận được sẽ được đưa vào mảng buffer.
* Cuối cùng dữ liệu sẽ được chuyển thành chuỗi ký tự rồi gửi lên Serial Port.
* Lặp lại quá trình trên cho đến khi hết dữ liệu được gửi.

* 1. **Lập trình giao diện giám sát trên Visual Studio:**
     1. **Giao diện:**

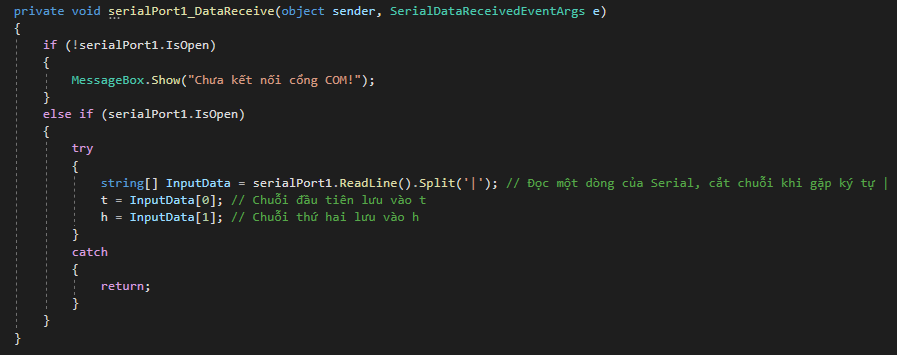
****

*Hình 3.5 : Giao diện giám sát.*

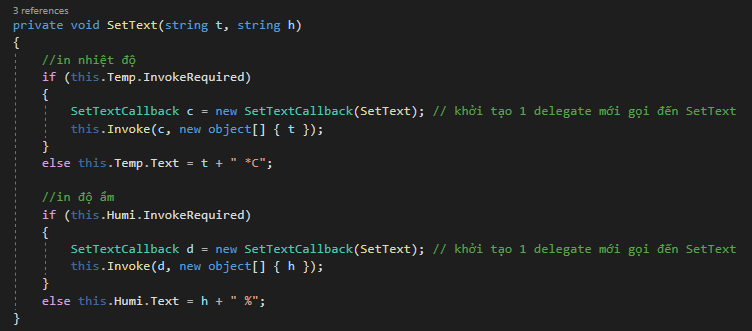
* Giao diện giám sát trên máy tính thu thập dữ liệu qua serial port, giao diện hiển thị giá trị nhiệt độ và độ ẩm tức thời.
* Giao diện cho phép kết nối và ngắt kết nối với vi xử lý thông qua các nút bấm.
* Trước khi kết nối cần chọn cổng COM và tốc độ Baud phù hợp với vi xử lý.
  + 1. **Lập trình sự kiện:**

Vi điều khiển liên tục gửi dữ liệu lên máy tính, đọc chuỗi dữ liệu phân tách lưu vào các biến đồng thời hiển thị ra các textbox.

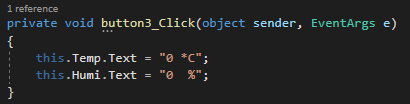
* Dùng hàm *Split()* để phát hiện ký tự ‘|’ và tiến hành tách chuỗi. Chuỗi nhận từ vi xử lý sẽ được tách thành 2 chuỗi con và được lưu vào 2 biến t và h tương ứng với nhiệt độ và độ ẩm.

**

* Dữ liệu trong hai biến t và h được hiển thị lên các textbox tương ứng.



* Sự kiện nhấn nút *Clear :*khi nhấn dữ liệu nhiệt độ và độ ẩm hiển thị trên textbox được trả về 0.

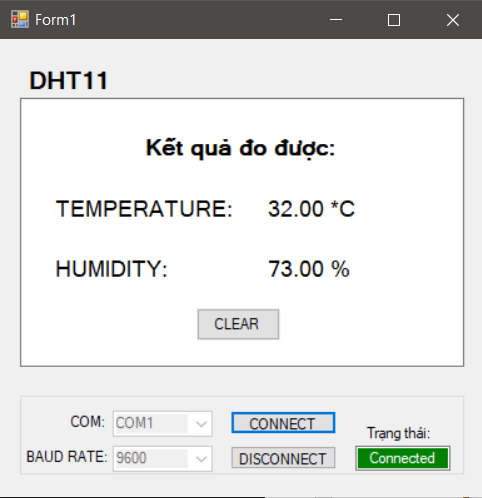


# CHƯƠNG 4: KẾT QUẢ VÀ NHẬN XÉT

* 1. **Kết quả:**

Thực hiện truyền nhận dữ liệu nhiệt độ và độ ẩm ở tần số 433MHz.

Kết quả thu được ở máy thu.



*Hình 4.1 : Kết quả do nhiệt độ và độ ẩm.*

Nhận xét:

* Tín hiệu thu được ổn định ở tần số 433 MHz.
* Kết quả đọc được từ module DHT11 tương đối chính xác.
  1. **Thử nghiệm khoảng tần số:**
* **Cách thức đo đạc, thử nghiệm:**

B1: Chuẩn bị cài đặt máy phát và máy thu.

B2: Đặt máy phát và máy thu ở vị cố định cách nhau 1m.

B3: Chỉnh tần số của máy phát và máy thu và ghi kết quả vào bảng.

|  |  |  |
| --- | --- | --- |
| Tần số (MHz) | Hoạt động | Không hoạt động |
| 315 |  | X |
| 433 | X |  |
| 868 | X (không ổn định) |  |
| 915 | X (không ổn định) |  |

* **Đánh giá:** Với số liệu của nhà sản xuất, module CC1101 hoạt động ở tần số 433Mhz tốt, các tần số khác không ổn định.

* 1. **Thử nghiệm khoảng cách truyền:**
* **Cách thức đo đạc, thử nghiệm:**

B1: Chuẩn bị cài đặt máy phát và máy thu.

B2: Cố định máy phát và di chuyển máy thu ở nhiều vị trí khoảng cách khác nhau, yêu cầu không có vật cản chắn sóng.

B3: Ghi kết quả vào bảng:

|  |  |
| --- | --- |
| Khoảng cách (m) | Kết quả |
| 10 | Tín hiệu ổn định. |
| 20 | Tín hiệu ổn định. |
| 50 | Tín hiệu ổn định. |
| 70 | Tín hiệu ổn định. |
| 80 | Bắt đầu xuất hiện lỗi. |

* **Đánh giá:** Module CC1101 cho phép tầm hoạt động với khoảng cách truyền lớn. Trong quá trình thử nghiệm, tính ổn định của module càng giảm đối với khoảng cách càng lớn do những yếu tố như dây nối giữa module và vi xử lý, địa điểm đo,…

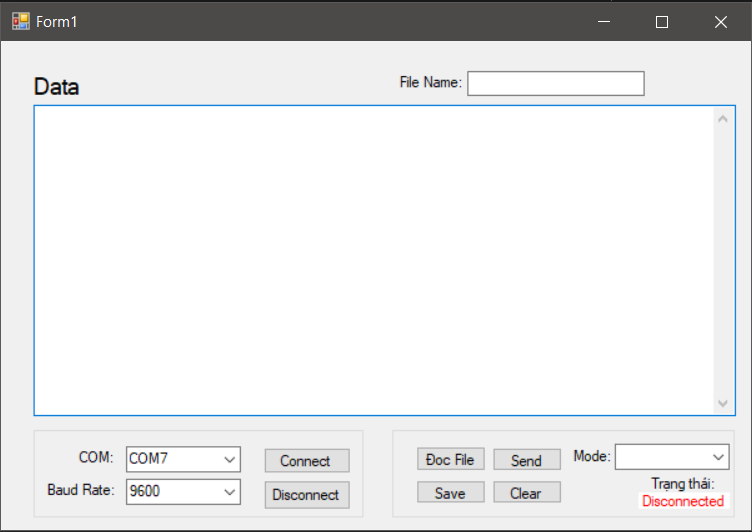
# CHƯƠNG 5: TRUYỀN FILE TEXT.

* 1. **Hướng giải quyết vấn đề:**

Như đã tìm hiểu từ trước, một gói tin tối đa có thể truyền được là 61 ký tự. Như vậy muốn truyền đi một văn bản có nhiều hơn 61 ký tự ta có thể chia nhỏ nó ra thành nhiều chuỗi.

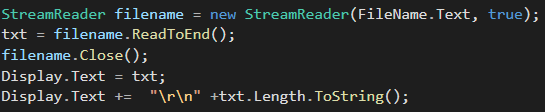
Để thực hiện được việc này, nhóm em đã sử dụng C# Winform để tạo một giao diện có thể đọc file text và chia văn bản thành các chuỗi nhỏ, sau đó gửi từng chuỗi đó cho vi xử lý.

* 1. **Lập trình hệ thống:**
     1. **Lập trình trên C# Winform:**
        1. **Giao diện:**

****

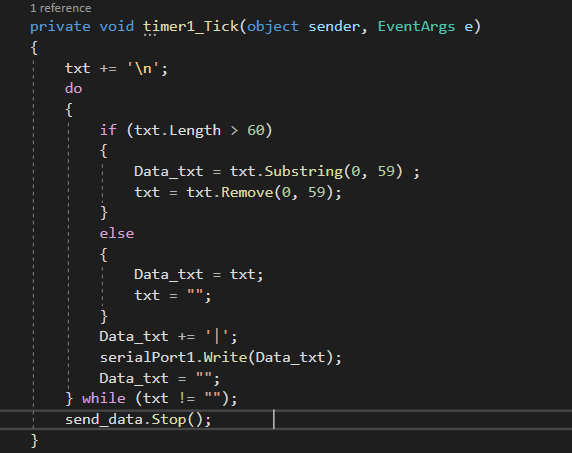
*Hình 5.1 : Giao diện gửi file text.*

* + - 1. **Gửi dữ liệu cho vi xử lý:**
* Đọc file:



File text lưu trong thư mục Debug và được đọc vào biến *txt.* Ở chế độ gửi, màn hình được sử dụng để hiển thị nội dung trong file text.

* Xử lý chuỗi và gửi cho vi xử lý:

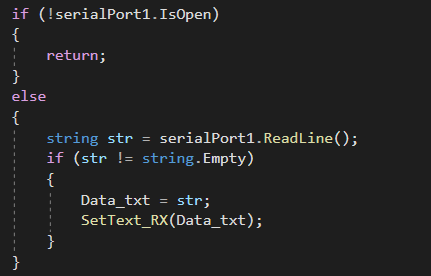


Trong sự kiện timer1\_Tick, sử dụng vòng lặp *do-while* để kiểm tra biến *txt* có dữ liệu hay không. Nếu có, sử dụng lệnh *if*  để kiểm tra chiều dài của biến txt:

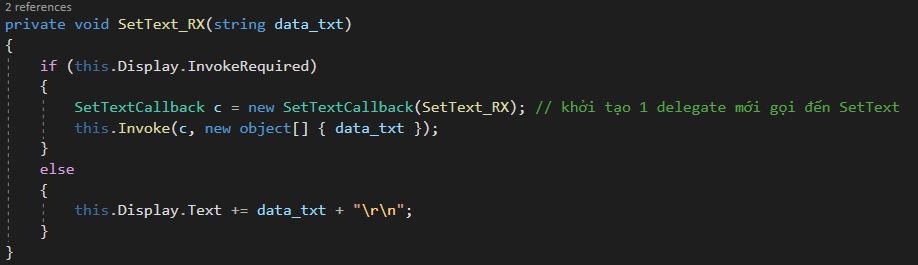
* Nếu chiều dài biến txt lớn hơn 60, dùng hàm *Substring()* lấy 60 ký tự đầu tiên và gán vào biến *Data\_txt*, sau đó xóa 60 ký tự đó khỏi biến *txt.* Lặp lại quá trình cho tới khi chiều dài biến *txt* nhỏ hơn hoặc bằng 60.
* Nếu chiều dài biến *txt* nhỏ hơn hoặc bằng 60, cho tất cả các ký tự trong *txt* vào *Data\_txt* và xóa *txt*.

Sau khi lấy được 60 ký tự vào biến *Data\_txt*, ta gửi nó cho vi xử lý qua Serial Port bằng lệnh *serialPort1.Write(Data\_txt).*

* + - 1. **Nhận dữ liệu từ vi xử lý:**
* **Nhận và hiển thị dữ liệu:**

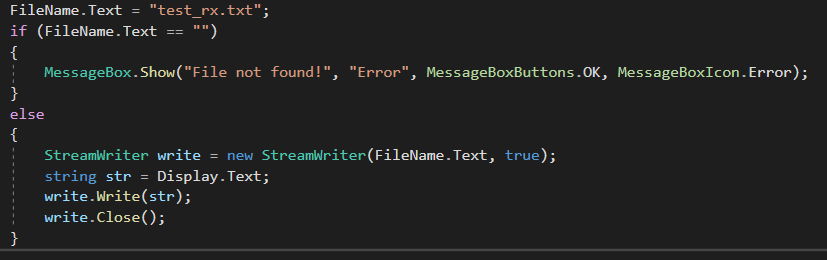
****

Ở chế độ nhận, chương trình liên tục đọc dữ liệu trên Serial Port. Khi phát hiện dữ liệu, chương trình sẽ hiển thị dữ liệu đó lên màn hình bằng hàm *SetText\_RX().*



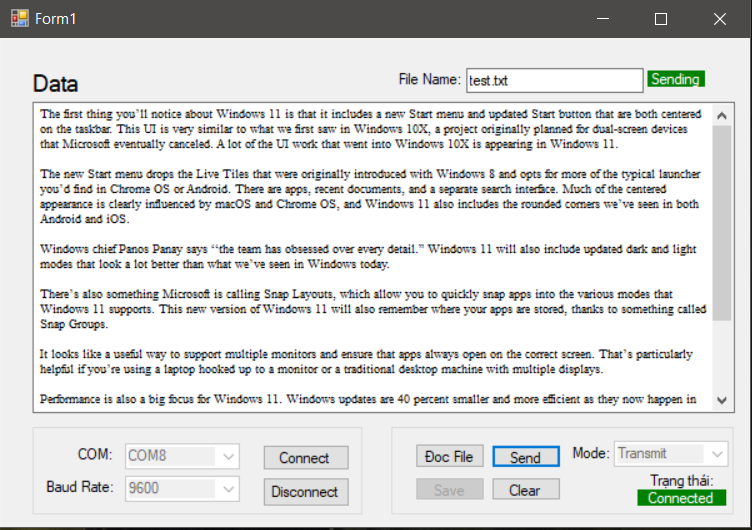
* **Lưu dữ liệu nhận được:**

Nhấn nút save để lưu dữ liệu.



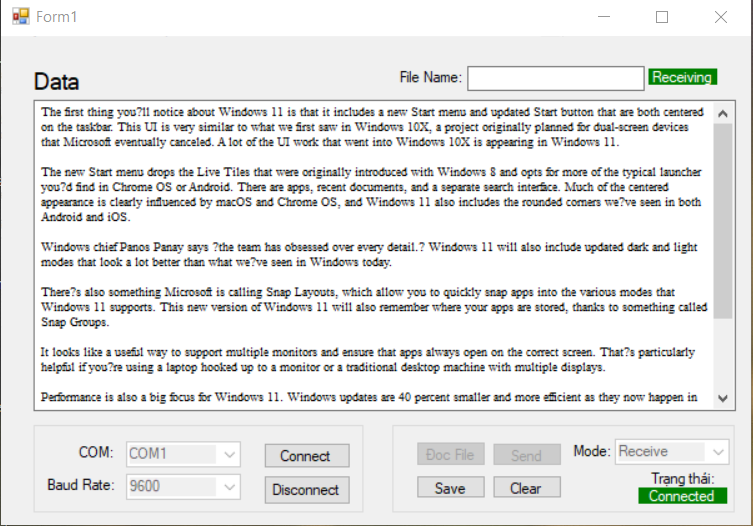
Dữ liệu được lưu vào file *test\_rx.txt* trong thư mục Debug.

* 1. **Kết quả và nhận xét:**
     1. **Kết quả:**
* **Máy phát:**



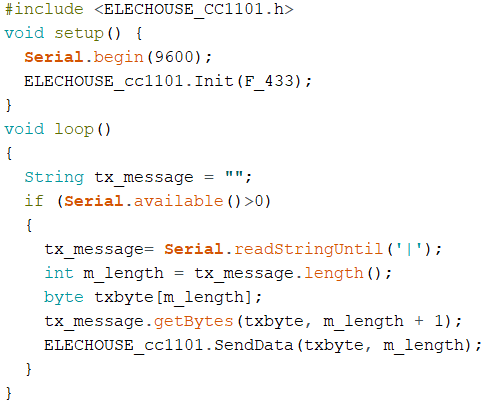
*Hình 5.2 : Màn hình hiển thị file text bên máy phát.*

* **Máy thu:**



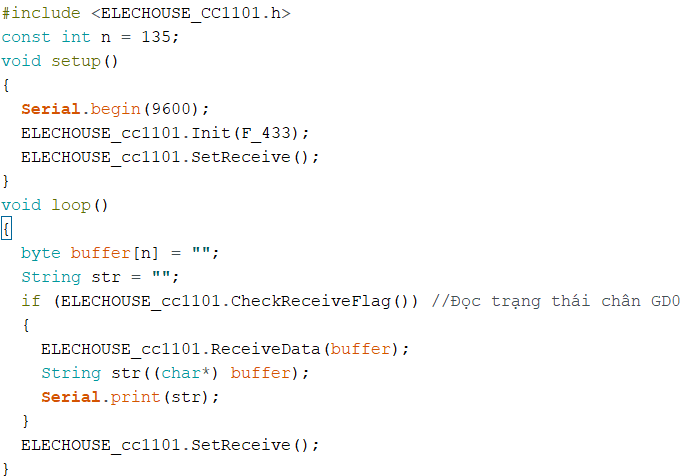
*Hình 5.3: Màn hình hiển thị file text bên máy thu.*

* + 1. **Lập trình trên Arduino:**
* **Máy phát:**



Khi vi xử lý phát hiện có dữ liệu ở Serial Port, dữ liệu sẽ được đọc liên tục cho tới khi gặp ký tự gạch đứng “|”. Tiếp theo, vi xử lý sẽ gửi dữ liệu cho máy thu như bình thường.

* **Máy thu:**



Dữ liệu nhận được ở máy thu sẽ gửi cho máy tính thông qua Serial Port.

* + 1. **Nhận xét:**
* Đã thực hiện được yêu cầu đặt ra là truyền file text có nhiều hơn 61 ký tự.
* Còn một số vấn đề chưa giải quyết được:
* Thời gian truyền còn khá lâu ( mất khoảng 5s để truyền 5000 ký tự).
* Chưa hiển thị được tiếng Việt cũng như một số ký tự đặc biệt.

# CHƯƠNG 6: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

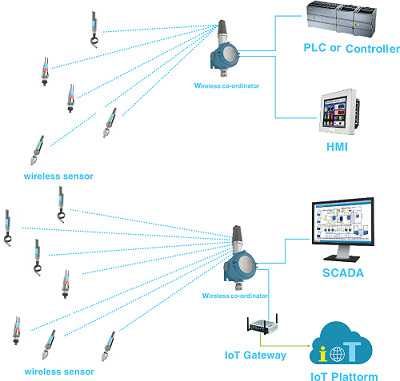
* 1. **Kết luận:**

Qua đồ án lần này, em đã có được kiến thức về công nghệ truyền không dây hoàn toàn mới. Trong quá trình nghiên cứu và thực hiện chúng em nhận thấy các ưu nhược điểm sau:

* + 1. **Ưu điểm:**
* Học được cách làm việc chuyên nghiệp với một đề tài hoàn chỉnh, khả năng làm việc theo nhóm.
* Cách thức nghiên cứu, nhìn nhận và xử lý 1 vấn đề mới.
* Khả năng xử lý các vấn đề trong quá trình thực hiện đề tài (thay đổi thiết bị, sửa lỗi phần cứng, phần mềm,….).
* Tăng khả năng đọc hiểu datasheet của các linh kiện và tìm hiểu, sửa các lỗi sai có trong thư viện có sẵn.
  + 1. **Khuyết điểm:**
* Chưa hiểu được tất cả vấn đề trong data sheet của các linh kiện có sẵn.
* Sắp xếp thời gian làm việc chưa hợp lý.
* Chưa đạt được mọi mục tiêu theo yêu cầu của thầy hướng dẫn: Thực hiện hệ thống truyền nhận thành một mạng lưới, truyền nhận 1 file ảnh,...
  + 1. **Kết quả đạt được:**
* Nội dung 1: Tìm hiểu về cảm biến, những loại module Sub – 1GHz, vi xử lý:
* Tìm hiểu các loại cảm biến, module Sub – 1GHz có mặt trên thị trường.
* Chọn loại cảm biến, module phù hợp.
* Tận dụng những thiết bị có sẵn.
* Nội dung 2: Tìm hiểu về nguyên lý của Sub – 1GHz và cách sử dụng các loại cảm biến đã chọn với vi xử lý trung tâm:
* Có kiến thức mới về Sub- 1GHz.
* Hiểu rõ được nguyên lý hoạt động của loại cảm biến đã chọn.
* Đọc datasheet, nghiên cứu nguyên lý đóng gói, truyền và nhận của module CC1101.
* Nội dung 3: Nạp và đọc kết quả thu được trên Arduino IDE và Serial.
* Nội dung 4: Điều chỉnh và thử nghiệm các khoảng cách và tần số:
* Xây dựng một hệ thống hoàn chỉnh.
* Thử nghiệm các khoảng cách và tần số và so sánh với tính toán lý thuyết.
  1. **Hướng phát triển:**

Do đề tài mới chỉ dừng ở mức truyền nhận tín hiệu đơn giản, chúng ta có thể phát triển nó theo nhiều hướng như:

* Dùng mạng wireless để kết nối các cảm biến wireless xung quanh đưa dữ liệu về bộ nhận wireless trung tâm.
* Tìm hiểu các mode khác nhau như MSK trên module.
* ….



# TÀI LIỆU THAM KHẢO

CC1101 DataSheet, <https://www.ti.com/lit/ds/symlink/cc1101.pdf?ts=1622802020732>

CC1101 library for Arduino, <https://github.com/simonmonk/CC1101_arduino>

Các bài viết về lập trình Arduino, <http://arduino.vn/>

DHT11 library for Arduino, <https://www.arduino.cc/reference/en/libraries/dht-sensor-library/>

DHT11 DataSheet, <https://datasheetspdf.com/pdf/785590/D-Robotics/DHT11/1>

# PHỤ LỤC

**Mã nguồn chương trình:**

**Chương trình máy phát:**

#include <ELECHOUSE\_CC1101.h>

#include "DHT.h"

const int DHTPIN = 4;

const int DHTTYPE = DHT11;

DHT dht(DHTPIN, DHTTYPE);

void setup() {

Serial.begin(9600);

dht.begin();

ELECHOUSE\_cc1101.Init(F\_433);

}

void loop() {

float h = dht.readHumidity();

float t = dht.readTemperature();

String tx\_message = String(t)+ "|" + String(h);

int m\_length = tx\_message.length();

byte txbyte[m\_length];

tx\_message.getBytes(txbyte, m\_length + 1); //m\_length + 1: cộng 1 byte địa chỉ

ELECHOUSE\_cc1101.SendData(txbyte, m\_length);

delay(1000);

}

**Chương trình máy thu:**

#include <ELECHOUSE\_CC1101.h>

const int n = 135;

void setup(){

Serial.begin(9600);

ELECHOUSE\_cc1101.Init(F\_433);

}

void loop(){

ELECHOUSE\_cc1101.SetReceive();

byte buffer[n] = "";

String str = "";

if (ELECHOUSE\_cc1101.CheckReceiveFlag()) //Đọc trạng thái chân GD0

{

ELECHOUSE\_cc1101.ReceiveData(buffer);

String str((char\*) buffer);

Serial.println(str);

}

}