# **GIỚI THIỆU YÊU CẦU – GIỚI HẠN**

* 1. **GIỚI THIỆU**

Cân là một dụng cụ dùng để đo trọng lượng hoặc tính toán khối lượng, được ứng dụng rất nhiều trong cuộc sống của con người như: trong kinh doanh, trong chăm sóc sức khỏe… Có rất nhiều loại cân như: Cân treo, cân lò xo, cân đòn, cân điện tử…

Cân điện tử là cân sử dụng mạch điện tử và cảm biến lực để biến các tín hiệu điện thành con số thể hiện trọng lượng của vật mẫu. Công nghệ ngày càng phát triển, cân điện tử dần trở nên quen thuộc, thay thế cho những chiếc cân cơ truyền thống thông thường. Với ưu thế ở tính chính xác, kết quả cân hiển thị dưới dạng số nên rất dễ quan sát, dễ đọc kết quả với mọi vị trí nhìn, minh bạch, khả năng cân đa dạng (từ vật rất nhẹ đến vật rất nặng), cân được nhiều vật có kích thước, hình dạng khác nhau.

Bên cạnh sự phát triển của các bộ vi xử lí, vi điều khiển, IC số ngày càng thông dụng và hoàn thiện hơn. Cân điện tử cũng ngày càng đa dạng hơn. Nhưng có thể nói sự xuất hiện của Arduino vào năm 2005 tại Italia đã mở ra một hướng đi mới. Sự xuất hiện của Arduino đã hỗ trợ cho con người rất nhiều trong lập trình và thiết kế. Phần cứng của thiết bị đã được tích hợp nhiều chức năng cơ bản và mã nguồn mở. Thư viện rất phong phú và được chia sẻ miễn phí. Dựa vào những yếu tố đó chúng em chọn đề tài **“THIẾT KẾ CÂN ĐIỆN TỬ DÙNG ARDUINO”** để nghiên cứu và chọn làm đề tài Đồ án môn học này.

* 1. **GIỚI HẠN**
* Sử dụng arduino nano làm khối xử lý trung tâm.
* Cảm biến loadcell có hạn mức cân tối đa là 5Kg.
* Sử dụng module Hx711 để khếch đại tín hiệu từ cảm biến loadcell và chuyển đổi ADC.
* Sử dụng phím ma trận 4x4 để điều khiển.
* Dùng buzzer 5V làm còi cảnh báo.
* Kết quả cân được sẽ hiển thị lên LCD 20x4.
* Các giới hạn trong quá trình sử dụng cân: tổng thành tiền tối đa là 1 tỷ đồng, giá thành một sản phẩm không được vượt quá 1 triệu đồng, khối lượng một lần cân không vượt quá 5Kg, ở chế độ đếm thì số sản phẩm thì đếm tối đa là 1000 sản phẩm.

# **THIẾT KẾ SƠ ĐỒ KHỐI**

* 1. **GIỚI THIỆU**

Đề tài **“cân điện tử”** sử dụng cảm biến Loadcell 5kg. Người dùng có thể cân được vật mẫu có khối lượng nhỏ hơn hoặc bằng 5kg, nhận được cảnh báo khi cân vật mẫu quá khối lượng cho phép. Ngoài ra, có thể tính giá tiền của sản phẩm bằng cách nhập giá tiền từ keypad 4x4. Màn hình có thể hiển thị đầy đủ các thông tin cần thiết cho người dùng như: khối lượng, giá tiền.

* 1. **THIẾT KẾ SƠ ĐỒ KHỐI**

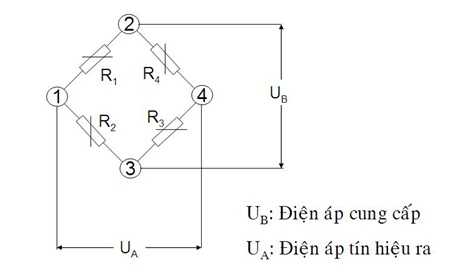
Từ các yêu cầu của đề tài thì sơ đồ khối của hệ thống như hình sau:



**Hình 2.1** Sơ đồ khối tổng quát cân điện tử

Chức năng từng khối:

* Khối nguồn: Có chức năng cấp nguồn cho toàn bộ mạch để hoạt động.
* Khối cảm biến: Chuyển đổi lực hoặc trọng lượng thành tín hiệu điện.
* Khối ADC: Chuyển đổi tín hiệu điện áp thu được từ cảm biến thành tín hiệu Digital rồi gửi đến vi điều khiển.
* Khối xử lý trung tâm: Sử dụng bộ điều khiển trung tâm là Board Arduino Nano, tạo ra các lệnh điều khiển, điều khiển các hoạt động của hệ thống. Nhận tín hiệu từ Keypad, từ cảm biến sau khi chuyển đổi ADC và gửi tín hiệu điều khiển sang các khối khác.
* Khối nhập giá tiền và điều khiển bằng keypad 4x4: Điều khiển, nhập giá tiền của sản phẩm sau đó gửi về Arduino để xử lý.
* Khối hiển thị: Hiển thị thông tin, giao tiếp với người sử dụng.
* Khối ngõ ra: Sử dụng Buzzer để cảnh báo khi vật mẫu đặt lên cân quá khối lượng cho phép.
  1. **THIẾT KẾ SƠ ĐỒ NGUYÊN LÝ**
     1. **KHỐI CẢM BIẾN**
* Chức năng: Loadcell là thiết bị cảm biến chuyển đổi lực thành tín hiệu điện áp.
* Trên thị trường hiện tại có hai loại loadcell là loadcell tương tự và loadcell số, cùng rất nhiều hạng cân khác nhau. Đối với loadcell số do được tích hợp bộ chuyển ADC bên trong nên sẽ ít bị nhiễu hơn, cũng vì vậy mà cấu tạo bên trong cũng phức tạp hơn nên việc chế tạo ra các loadcell có hình dạng theo từng nhu cầu là khá khó, và tất nhiên là giá thành cũng cao hơn loadcell tương tự. Còn đối với loadcell tương tự thì ngược lại, nó rất dễ chế tạo theo nhiều hình dạng khác nhau để người dùng chọn lừa và giá thành cùng thấp hơn loadcell số, có một nhượt điểm là ngõ ra của loadcell tương tự là tín hiệu điện áp rất nhỏ và dễ bị nhiễu.
* Trong đồ án môn học này với mục đích nghiên cứu nên loadcell cần nhỏ gọn để thuận tiện cho việc lắp đặt, đặc biệt là giá thành phù hợp nên nhóm em chọn loadcell tương tự mức cân tối đa là 5kg.
* Cấu tạo: Loadcell được cấu tạo bởi hai thành phần, thành phần thứ nhất là “Straingage” và thành phần còn lại là “Load”. “Strain gage” là một điện trở đặc biệt chỉ nhỏ bằng móng tay, có điện trở thay đổi khi bị nén hay kéo dãn và được nuôi bằng một nguồn điện ổn định, được dán chết lên “Load” - một thanh kim loại chịu tải có tính đàn hồi.
* Nguyên lý hoạt động:
* Hoạt động dựa trên nguyên lý cầu điện trở cân bằng “Wheatstone”**.** Giá trị lực tác dụng tỉ lệ với sự thay đổi điện trở cảm ứng trong cầu điện trở, và trả về tín hiệu điện áp tỉ lệ.
* Cấu tạo chính của loadcell gồm các điện trở strain gauges R1, R2, R3, R4 kết nối thành 1 cầu điện trở Wheatstone như hình dưới và được dán vào bề mặt của thân loadcell.

[](http://codientrangan.com/)

**Hình 2.2** Sơ đồ kết nối các điện trở Strain gauges R1, R2, R3, R4

* Tại trạng thái cân bằng (trạng thái không tải), điện áp tín hiệu ra là số không hoặc gần bằng không khi bốn điện trở được gắn phù hợp về giá trị.
* Khi có tải trọng hoặc lực tác động lên thân loadcell làm cho thân loadcell bị biến dạng (giãn hoặc nén), điều đó dẫn tới sự thay đổi chiều dài và tiết diện của các sợi kim loại của điện trở strain gauges dán trên thân loadcell dẫn đến một sự thay đổi giá trị của các điện trở strain gauges. Sự thay đổi này dẫn tới sự thay đổi trong điện áp đầu ra. Đó là lý do tại sao cầu điện trở Wheatstone còn được gọi là một mạch cầu cân bằng.
* Sự thay đổi điện áp này là rất nhỏ, do đó nó chỉ có thể được đo và chuyển thành số sau khi đi qua bộ khuếch đại.

**Bảng 2.1** Thông số kỹ thuật cảm biến Loadcell 5kg

|  |  |
| --- | --- |
| Tải trọng | 5Kg |
| Rated Output (mV/V) | 1.0 0.15 |
| Độ lệch tuyến tính (%) | 0.05 |
| Ảnh hưởng nhiệt độ tới độ nhạy %RO/ độ C | 0.003 |
| Ảnh hưởng nhiệt độ tới điểm không %RO/ độ C | 0.02 |
| Độ cân bằng điểm không %RO | 1 |
| Trở kháng đầu vào (Ω ) | 1066 20 |
| Trở kháng ngõ ra (Ω ) | 1000 20 |
| Trở kháng cách li (MΩ) 50V | 2000 |
| Điện áp hoạt động | 5V |
| Nhiệt độ hoạt động | -20 ~ 65 độ C |
| Safe Overload %RO | 120 |
| Ultimate overload %RO | 150 |
| Chất liệu cảm biến | Nhôm |
| Độ dài dây | 180mm |
| Dây đỏ | Ngõ vào (+) |
| Dây đen | Ngõ vào (-) |
| Dây xanh Lá | Ngõ ra (+) |
| Dây trắng | Ngõ ra (-) |

A close up of a device

Description automatically generated

**Hình 2.3** Cảm biến Loadcell (5kg)

* + 1. **KHỐI ADC**
* Chức năng: Chuyển đổi tín hiệu điện áp thu được từ cảm biến thành tín hiệu Digital rồi truyền đến vi điều khiển.
* Sử dụng module ADC 24bit Hx711.

Có nhiều loại module ADC nhưng phải dùng module Hx711 do Output của Loadcell có điện áp rất nhỏ, cỡ khoảng 1-3mV. Vì vậy cần những bộ ADC có độ phân giải cao để có thể đọc được mức điện áp trên.

Giả sử: sử dụng các bộ ADC 8bit => Độ phân giải Analog = 5V/256 = 19.53mV, giá trị này lớn hơn gấp nhiều lần so với output của loadcell nên sẽ không đọc được.

Giao tiếp 2 dây với vi điều khiển: 2 chân SCK (clock) và DT (Data)

**Bảng 2.2** Thông số kỹ thuật module ADC Hx711

|  |  |
| --- | --- |
| Điện áp hoạt động | 2.7 – 5V |
| Dòng tiêu thụ | <1.5 mA |
| Tốc độ lấy mẫu | 10 – 80 SPS (tùy chỉnh) |
| Độ phân giải | 24bit ADC |
| Độ phân giải điện áp | 40 mV |
| Kích thước | 38 x 21 x 10 mm |

A close up of text on a white background

Description automatically generated

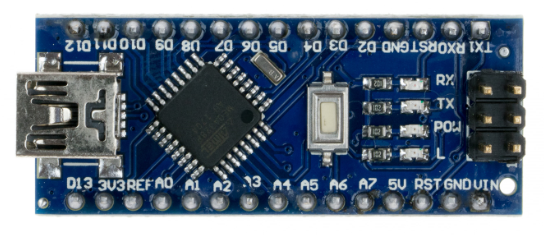
**Hình 2.4** Sơ đồ nguyên lý của module Hx711

A circuit board

Description automatically generated

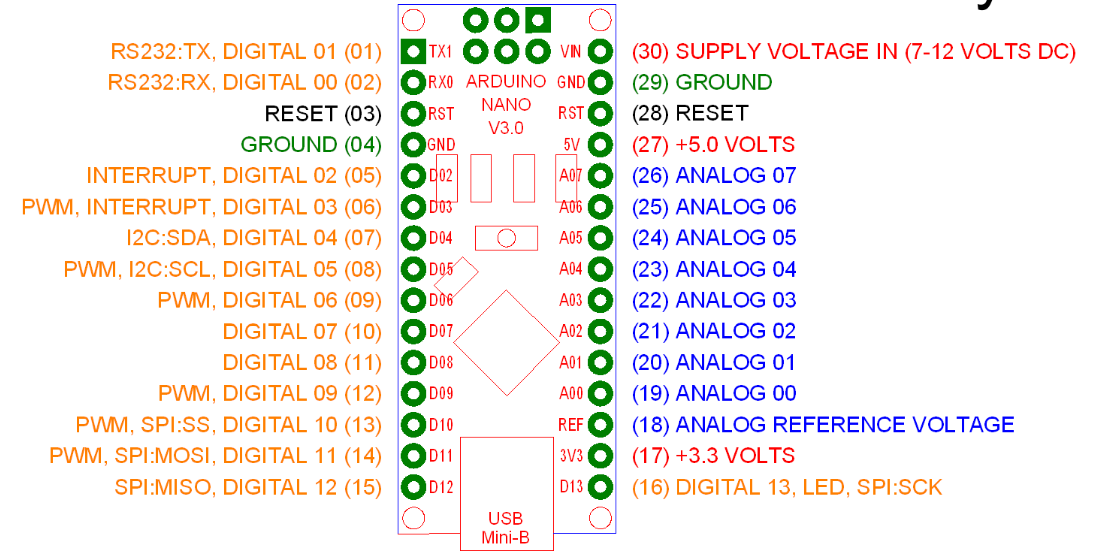
**Hình 2.5** Module ADC 24bit Hx711

* + 1. **KHỐI XỬ LÝ TRUNG TÂM**
* Chức năng: Là trung tâm xử lý, điều khiển các hoạt động của hệ thống. Nhận tín hiệu từ Keypad, từ cảm biến sau khi chuyển đổi ADC và gửi tín hiệu điều khiển sang các khối khác.
* Lựa chọn linh kiện:
* Trong những năm gần đây, sự phát triển mạnh mẽ của Arduino đã cho thấy tầm ảnh hưởng rất nhiều tới thế giới hiện tại, với cộng đồng mạng hỗ trợ, thư viện hỗ trợ hầu hết các module, các sản phẩm thiết bị ngoại vi khác, thấy được sự quan trọng và phù hợp với yêu cầu trong đề tài này, nên nhóm chúng em chọn Arduino. Các loại bộ điều khiển trung tâm khác cũng có thể sử dụng nhưng do thiết kế và lập trình phức tạp hơn, nên lựa chọn Arduino để có thể đơn giản hóa các khâu như thiết kế, lập trình, …
* Bộ điều khiển trung tâm chúng em chọn sử dụng là Arduino Nano V3.0 vì có thể lập trình cũng như kết nối dễ dàng hơn nhờ cộng đồng người dùng đông đảo. Arduino Nano V3.0 sử dụng vi điều khiển ATmega328 họ 8 bit, với 14 chân digital dùng để đọc hoặc xuất tín hiệu. Arduino Nano có thể được cấp nguồn 5V thông qua cổng mini-USB hoặc cấp nguồn ngoài với điện áp khuyên dùng là 7-12V DC và giới hạn là 6-20V DC. Thường thì cấp nguồn bằng pin vuông 9V là hợp lí nhất nếu không có sẵn nguồn từ cổng USB. Nếu cấp nguồn vượt quá ngưỡng giới hạn trên, sẽ làm hỏng Arduino Nano.



**Hình 2.6** Arduino Nano V3.0

* Các chân năng lượng:
  + - * GND (Ground): cực âm của nguồn điện cấp cho Arduino Nano. Khi bạn dùng các thiết bị sử dụng những nguồn điện riêng biệt thì những chân này phải được nối với nhau.
      * 5V: cấp điện áp 5V đầu ra. Dòng tối đa cho phép ở chân này là 500mA.
      * 3V3: cấp điện áp 3.3V đầu ra. Dòng tối đa cho phép ở chân này là 50mA.
      * Vin (Voltage Input): để cấp nguồn ngoài cho Arduino Nano, bạn nối cực dương của nguồn với chân này và cực âm của nguồn với chân GND.
      * RESET: việc nhấn nút Reset trên board để reset vi điều khiển tương đương với việc chân RESET được nối với GND qua 1 điện trở 10KΩ.
* Các cổng vào ra: Arduino Nano có 14 chân digital dùng để đọc hoặc xuất tín hiệu. Chúng chỉ có 2 mức điện áp là 0V và 5V với dòng vào/ra tối đa trên mỗi chân là 40mA. Ở mỗi chân đều có các điện trở pull-up từ được cài đặt ngay trong vi điều khiển ATmega328 (mặc định thì các điện trở này không được kết nối).
  + - * Chân Serial: RX và TX: dùng để gửi (transmit – TX) và nhận (receive – RX) dữ liệu TTL Serial. Arduino Nano có thể giao tiếp với thiết bị khác thông qua 2 chân này. Kết nối bluetooth thường thấy nói nôm na chính là kết nối Serial không dây. Nếu không cần giao tiếp Serial, bạn không nên sử dụng 2 chân này nếu không cần thiết
      * Chân PWM: 6, 8, 9, 12, 13 và 14: cho phép bạn xuất ra xung PWM với độ phân giải 8bit (giá trị từ 0 → -1 tương ứng với 0V → 5V) bằng hàm analogWrite(). Nói một cách đơn giản, bạn có thể điều chỉnh được điện áp ra ở chân này từ mức 0V đến 5V thay vì chỉ cố định ở mức 0V và 5V như những chân khác.
      * Chân giao tiếp SPI: 13 (SS), 14 (MOSI), 15 (MISO), 16 (SCK). Ngoài các chức năng thông thường, 4 chân này còn dùng để truyền phát dữ liệu bằng giao thức SPI với các thiết bị khác.
      * Arduino Nano có 8 chân analog (A0 → A7) cung cấp độ phân giải tín hiệu 10bit (0 → -1) để đọc giá trị điện áp trong khoảng 0V → 5V. Với chân AREF trên board, bạn có thể để đưa vào điện áp tham chiếu khi sử dụng các chân analog. Tức là nếu bạn cấp điện áp 2.5V vào chân này thì bạn có thể dùng các chân analog để đo điện áp trong khoảng từ 0V → 2.5V với độ phân giải vẫn là 10bit.
      * Đặc biệt, Arduino Nano có 2 chân D4 (SDA) và D5 (SCL) hỗ trợ giao tiếp I2C/TWI với các thiết bị khác.



**Hình 2.7** Sơ đồ chân của Arduino Nano

**Bảng 2.3** Thông số kỹ thuật Board Arduino Nano

|  |  |
| --- | --- |
| Điện áp hoạt động | 5V – DC |
| Tần số hoạt động | 16 MHz |
| Điện áp vào (khuyên dùng) | 7-12 V - DC |
| Điện áp vào (giới hạn) | 6-20 V - DC |
| Số chân Digital I/O | 14 (6 chân PWM) |
| Số chân Analog | 8 (độ phân giải 10bit) |
| Dòng tối đa trên mỗi chân I/O | 40 mA |
| Dòng ra tối đa (5V) | 500 mA |
| Dòng ra tối đa (3.3V) | 50 mA |
| Bộ nhớ Flash | 32 KB trong đó 2 KB được sử dụng bởi Bootloader |
| SRAM | 2 KB |
| EEPROM | 1 KB |

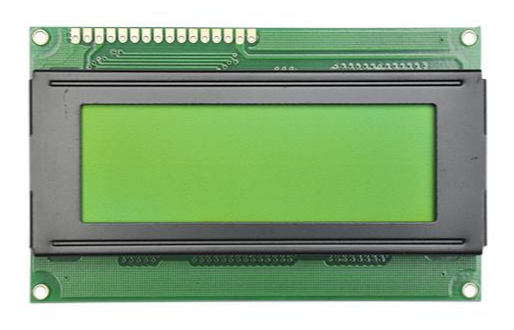
A close up of text on a white background

Description automatically generated

**Hình 2.8** Sơ đồ nguyên lý kết nối

* + - * Chân D13: Nối với khối âm thanh.
      * Chân D4 - D10: Nối với khối hiển thị.
      * Chân A0 – A7: Nối với keypad 4x4.
      * Chân D11 - D12: Giao tiếp 2 dây với Module ADC Hx711.
    1. **KHỐI HIỂN THỊ**
* Chức năng: Môi trường giao tiếp giữa người dùng và thiết bị, hiển thị thông tin như khối lượng, số lượng, giá tiền …
* Lựa chọn linh kiện:

Hiện nay, màn hình hiển thị có rất nhiều loại được sử dụng vào từng mục đích khác nhau như: LCD 10.1 inch cảm ứng điện dung Waveshare, màn hình cảm ứng Arduino TFT Shield 2,4inch, màn hình LCD Graphic, LCD text, LCD HMI. Tùy vào mục đích và cách sử dụng cũng như ứng dụng của dự án mà chúng ta sử dụng màn hình loại nào cho tiện lợi. Màn hình text LCD 20x4 sử dụng driver HD44780, có khả năng hiển thị 4 dòng với mỗi dòng 20 kí tự, màn hình có độ bền cao, rất phổ biến, nhiều code mẫu và dễ sử dụng thích hợp cho những người mới học và làm dự án. Nhận thấy điều ấy, nhóm chúng em quyết định chọn màn hình LCD 20x4.



**Hình 2.9** Màn hình LCD 20x4

LCD có nhiều loại và số chân của chúng cũng khác nhau nhưng có 2 loại phổ biến là loại 14 chân và loại 16 chân, sự khác nhau là các chân nguồn cung cấp, còn các chân điều khiển thì không thay đổi.

**Bảng 2.4** Các chân của LCD 20x4

|  |  |  |  |
| --- | --- | --- | --- |
| Thứ tự | Tên tín hiệu | I/O | Mô tả |
| 1 | VSS | Nguồn | GND |
| 2 | VDD | Nguồn | +5V |
| 3 | VO | Điện áp | Điều khiển ánh sáng nền |
| 4 | RS | INPUT | Register Select |
| 5 | R/W | INPUT | Read/Write |
| 6 | E | INPUT | Enable (strobe) |
| 7 | D0 | I/O | DATA LSB |
| 8 | D1 | I/O | DATA |
| 9 | D2 | I/O | DATA |
| 10 | D3 | I/O | DATA |
| 11 | D4 | I/O | DATA |
| 12 | D5 | I/O | DATA |
| 13 | D6 | I/O | DATA |
| 14 | D7 | I/O | DATA MSB |
| 15 | A | I | Nguồn dương +5V |
| 16 | K | I | GND |

Trong 16 chân của LCD được chia ra làm 3 dạng tín hiệu như sau:

* + - * Các chân cấp nguồn: Chân số 1 là chân nối mass (0V), chân thứ 2 là VDD nối với nguồn +5V. Chân thứ 3 dùng để chỉnh contrast thường nối với biến trở.
      * Các chân điều khiển: Chân số 4 là chân RS dùng để điều khiển lựa chọn thanh ghi. Chân R/W dùng để điều khiển quá trình đọc và ghi. Chân E là chân cho phép dạng xung chốt.
      * Các chân dữ liệu D7÷D0: Chân số 7 đến chân số 14 là 8 chân dùng để trao đổi dữ liệu giữa thiết bị điều khiển và LCD.
* Thông số kỹ thuật:
  + - * Điện áp hoạt động: 5VDC
      * Dòng điện tiêu thụ: 350uA - 600uA.
      * Nhiệt độ hoạt động: -30°C đến 75°C.
      * Kích thước 96 x 60 mm, chữ đen, nền xanh lá.
      * Đèn Led nền có thể điều khiển bằng biến trở hoặc PWM.

A close up of text on a white background

Description automatically generated

**Hình 2.10** Sơ đồ nối chân của LCD 20x4

* + 1. **KHỐI NHẬP GIÁ TIỀN VÀ ĐIỀU KHIỂN BẰNG KEYPAD 4x4**
* Lựa chọn linh kiện: Hệ thống với giá tiền nhập vào là các con số và một vài nút nhấn điều khiển đơn giản, không yêu cầu các ký tự đặc biệt nên chúng em sử dụng keypad4x4. Bàn phím này có 16 nút nhấn, xếp thành hình vuông 4 ngang 4 dọc nhưng chỉ cần 8 chân kết nối để điều khiển.



**Hình 2.11** Keypad 4x4

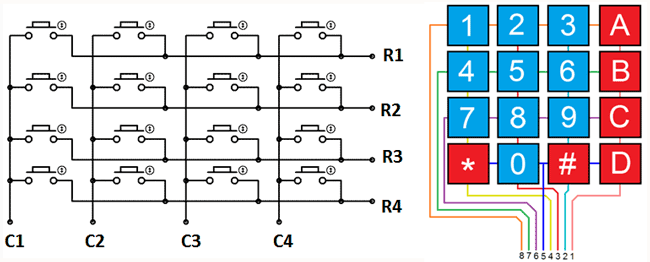
Với cách kết nối dạng ma trận thì 16 phím chỉ dùng 8 tín hiệu: 4 cho hàng và 4 cho cột – gọi là ma trận 4×4 sẽ được 16 phím.

Trong ma trận 4×4 thì có 4 hàng và 4 cột, hàng được chọn là tín hiệu vào – cột được chọn là tín hiệu ra, hàng thì treo lên nguồn Vcc qua điện trở nên mức logic của hàng luôn là mức 1.

Các phím nhấn thường hở nên 4 hàng luôn ở mức 1 hay R4R3R2R1 =1111.

Cột là tín hiệu ra nên chúng ta điều khiển xuất dữ liệu ra cột tùy ý. Để phân biệt các phím thì mỗi phím có 1 tên được đánh theo số thập lục phân từ ‘0’ đến ‘F’.

Để xem có phím nào nhấn hay không ta tiến hành quét từng cột bằng cách cho 1 cột ở mức 0, 3 cột còn lại ở mức 1 và kiểm tra tất cả các hàng, nếu tất cả các hàng vẫn ở mức logic 1 tức là không có nhấn phím, nếu có 1 hàng xuống mức 0 thì đã có nhấn phím.



**Hình 2.12** Sơ đồ nguyên lý của module bàn phím 4x4

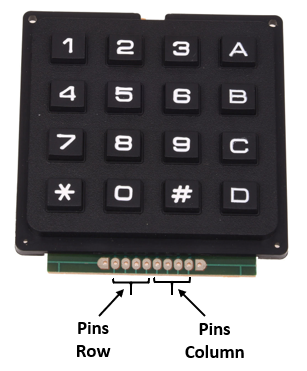
Mã quét cột là 4 bit thấp bao gồm các trạng thái: “1110B”, “1101B”, “1011B”, “0111B” kết hợp với 4 bit cao của hàng luôn là “1111B” nên dữ liệu kết hợp theo byte viết dạng số hex là “FEH”, “FDH”, “FBH”, “F7H”.

Gán cột bắt đầu từ 0, hàng gán bằng FFH, mã phím ban đầu gán bằng FFH.

Xuất lần lượt từng mã quét cột ra port quét phím, tiến hành kiểm tra từng hàng xem hàng nào bằng 0 thì phím tương ứng được nhấn. Khi đó ta gán giá trị tương ứng cho từng hàng.

Cuối cùng kiểm tra hàng xem có bằng FFH hay không, nếu bằng thì không phím nào được nhấn trong lúc thực hiện chương trình quét, thoát với mã phím bằng FFH, nếu khác FFH thì đã có nhấn phím, tiến hành tính toán mã phím và thoát với mã phím là của phím đã nhấn.

* + - * Nếu quét cột thứ 0 cột bằng 0, theo công thức mã phím sẽ có giá trị từ 0H đến 3H.
      * Nếu quét cột thứ 1 cột bằng 1, theo công thức mã phím sẽ có giá trị từ 4H đến 7H.
      * Nếu quét cột thứ 2 cột bằng 2, theo công thức mã phím sẽ có giá trị từ 8H đến BH.
      * Nếu quét cột thứ 3 cột bằng 3, theo công thức mã phím sẽ có giá trị từ CH đến FH



**Hình 2.13** Các chân của Keypad 4x4

**Bảng 2.5** Kết nối Keypad với Arduino

|  |  |
| --- | --- |
| Bàn phím | Arduino Nano |
| R1 | 4 |
| R2 | 5 |
| R3 | 6 |
| R4 | 7 |
| C1 | 8 |
| C2 | 9 |
| C3 | 10 |
| C4 | 11 |

* + 1. **KHỐI NGÕ RA**
* Chức năng: Cảnh báo khi vật mẫu đặt lên cân quá khối lượng cho phép.
* Lựa chọn linh kiện: Còi Buzzer

Sử dụng còi Buzzer để làm còi báo động. Vì còi Buzzer có tuổi thọ cao, hiệu suất ổn định, nhỏ gọn phù hợp với thiết kế.

Buzzer còn gọi là loa mini hay còi báo. Nó dùng để phát ra các âm thanh. Gồm 2 chân: Chân GND nối mass và chân tín hiệu nối pin Arduino.



**Hình 2.14** Buzzer

**Bảng 2.6** Thông số kỹ thuật Buzzer

|  |  |
| --- | --- |
| Điện áp hoạt động | 3.5 – 5.5 VDC |
| Kích thước | Đường kính 12mm, cao 9.7mm |
| Biên độ âm | >80 dB |
| Tần số cộng hưởng | 2300Hz ± 500Hz |
| Dòng điện tiêu thụ | <25mA |

A close up of text on a white background

Description automatically generated

**Hình 2.15** Sơ đồ kết nối kết nối Buzzer

* + 1. **KHỐI NGUỒN**
* Chức năng: Cấp nguồn cho toàn bộ mạch để hoạt động.

**Bảng 2.7** Liệt kê dòng điện sử dụng

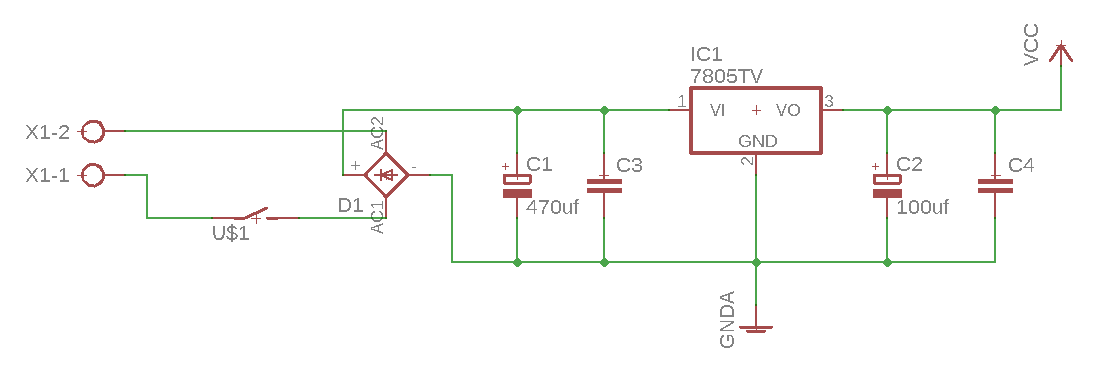
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên linh kiện** | **Dòng điện cực đại (mA)** | **Số lượng** | **Giá trị (A)** |
| 1 | Arduino Nano | 500 | 1 | 0.5 |
| 2 | LCD 20x04 | 40 | 1 | 0.04 |
| 3 | Buzzer | 25 | 1 | 0.025 |
| 4 | Module Hx711 | 1.5 | 1 | 0.0015 |
| **Tổng cộng:** | | | | **0.5665** |

Chọn dòng điện cấp cho board điều khiển: I = 1 (A)

**Bảng 2.8** Liệt kê điện áp hoạt động của linh kiện

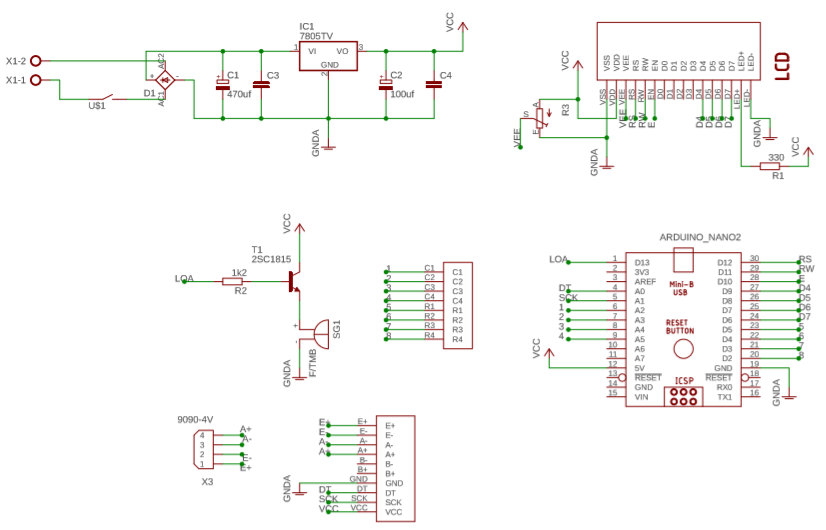
|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Linh kiện** | **Số lượng** | **Điện áp hoạt động** |
| 1 | Arduino nano | 1 | 5V |
| 2 | LCD 20x4 | 1 | 5V |
| 3 | Buzzer | 1 | 5V |
| 4 | HX711 | 1 | 2.7 – 5V |
| 5 | Cảm biến Loadcell | 1 | 5V |
|  |  |  |  |

Từ bảng liệt kê trên, chọn mức điện áp sử dụng là 5VDC. Ta sẽ dùng Pin 9V cho qua IC 7805 để đưa mức điện áp về 5V cấp nguồn cho mạch.



**Hình 2.16** Sơ đồ nguyên lý khối nguồn

* + 1. **SƠ ĐỒ NGUYÊN LÝ TOÀN MẠCH**

****

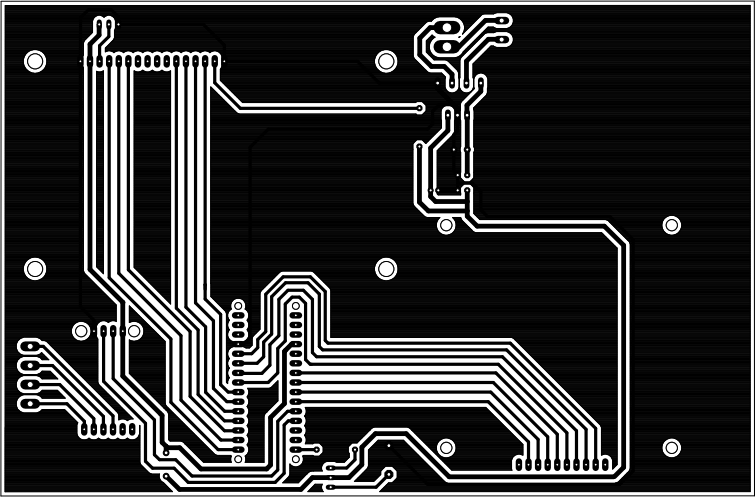
**Hình 2.17** Sơ đồ nguyên lý toàn mạch

# **THI CÔNG MẠCH – VIẾT CHƯƠNG TRÌNH**

* 1. **THI CÔNG PCB**

**Bảng 3.1** Các linh kiện sử dụng

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TT | Tên linh kiện | Thông số kỹ thuật | | Kích thước |
| 1 | Arduino Nano | Vi điều khiển | ATmega328 | 45 x 18mm |
| Điện áp hoạt động | 5V-DC |
| Dòng tiêu thụ | 0.5A |
| Điện áp vào giới hạn | 6-20V-DC |
| Số chân Digital | 14 |
| Số chân Analog | 8 |
| Dòng tối đa trên mỗi chân I/O | 40 mA |
| 2 | Keypad 4x4 | Nhiệt độ hoạt động | 0 ~ 70 0C | 65 x 64mm |
| Đầu nối ra | 8 chân |
| 3 | LCD 20x4 | Điện áp hoạt động | 5V | 98 x 60 x 13.5 mm |
| Màu chữ | Trắng |
| Màu nền | Xanh |
| Khoảng cách giữa 2 chân kết nối | 0.1 inch |
| 4 | Buzzer | Điện áp hoạt động | 3.5 – 5.5V | Đường kính 12mm, cao 9.7mm |
| Biên độ âm | >80dB |
| Tần số cộng hưởng | 2300Hz ± 500Hz |
| Dòng điện tiêu thụ | < 25mA |
| 5 | Module ADC HX711 | Điện áp hoạt động | 2.7 – 5V | 38 x 21 x 10 mm |
| Dòng tiêu thụ | <1.5 mA |
| Tốc độ lấy mẫu | 10 – 80 SPS |
| Độ phân giải | 24bit |
| Độ phân giải điện áp | 40 mV |
| 6 | Transistor | Model | NPN – TO92 |  |
| Nhiệt độ hoạt động | -55 – 125 0C |
| 7 | Điện trở R1 | 1.2 kΩ | |  |
| 8 | Điện trở R2 | 330 Ω | |  |
| 9 | Biến trở R3 | Biến trở tinh chỉnh 10K (3296W-103) | | 9.5 x 4.8 x 10 mm |
| 10 | Tụ hóa | Điện dung | 2200 uF | 10 x 20 mm |
| Điện áp | 16V |
| Nhiệt độ hoạt động | -55 – 125 0C |
| 11 | IC 7805 | Điện áp đầu ra | 5V |  |
| Dòng điện ra | 1.5A |
| 12 | Diode cầu | Điện áp tối đa | 700V | 14.2 x 3.4 x 10.6 mm |
| Dòng điện định mức | 3A |
| Nhiệt độ hoạt động | -55 – 150 0C |



**Hình 3.1** Sơ đồ mạch in

* 1. **HÀN LINH KIỆN**
  2. **VIẾT CHƯƠNG TRÌNH ĐIỀU KHIỂN**

### YÊU CẦU GIẢI THUẬT

Khi khởi động chương trình sẽ tự động đọc các thông số về giá sản phẩm, khối lượng đĩa cân, khối lượng mẫu và các thông số trạng thái được lưu trong vùng nhớ EEPROM phục hồi về trạng thái trước khi tắt cân.

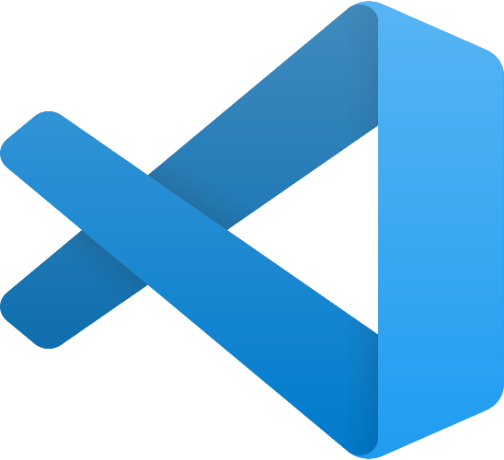
* Ở chế độ làm việc (mode = 0):
* Ấn 0 đến 9: Chọn 1 trong 10 loại sản phẩm được thiết lập sẵn. Tùy vào từng loại sản phẩm sẽ chọn chế độ cân hay đếm.
* Ấn “#”: Chọn chế độ cân sản phẩm bất kì – “mặc định”, nếu đang ở chế độ cân mặc định mà bấm “\*” thì sẽ chuyển đổi qua lại giữa hai chế độ “cân mặc định” và “đếm mặc định”.
* Ấn “\*”:
* Nếu ở chế độ cân khối lượng: thì chỉnh cân về mức 0.
* Nếu ở chế độ đếm: sẽ tiến hành lấy chỉ số cân hiện tại làm mẫu để đếm.
* Ấn “A”: Chuyển sang chế độ cài đặt (mode = 1) cho sản phẩm hiện tại.
* Ở chế độ cài đặt (mode = 1):
* Ấn 0 đến 9: Nhập giá sản phẩm.
* Ấn “C”: Xóa toàn bộ giá vừa nhâp.
* Ấn “D”: Xóa số vừa nhập sau cùng.
* Ấn “#”: Thiết lập chế độ cho sản phảm là cân hay đếm.
* Ấn “A”: Thoát chế độ cài đặt mà không lưu.
* Ấn “B”: Lưu lại cài đặt và chuyển cân về chế độ làm việc.
* Khi cân vượt mức 5Kg thì sẽ có chuông cảnh báo.
* Tùy vào các thông số được cấu hình như trên mà LCD sẽ hiển thị phù hợp.

### LƯU ĐỒ GIẢI THUẬT



**Hình 3.2** Lưu đồ giải thuật

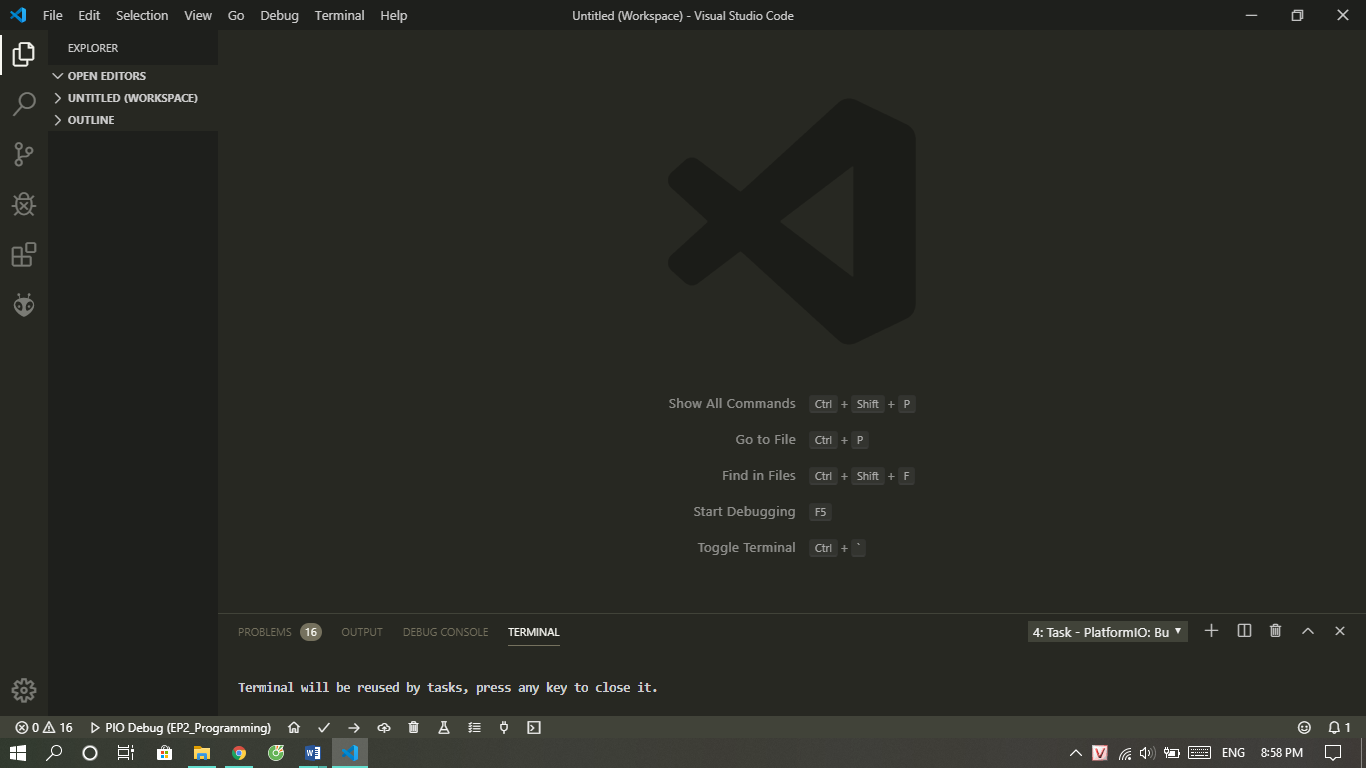
### PHẦN MỀM LẬP TRÌNH – VISUAL STUDIO CODE

Microsoft giới thiệu Visual Studio Code vào tháng 4 năm 2015 ở hội nghị Build. Đặc điểm nổi bật là đơn giản, gọn nhẹ, dễ dàng cài đặt. Visual Studio Code có thể cài đặt được trên cả Windows, Linux và Mac OS và hỗ trợ nhiều ngôn ngữ.

**Hình 3.3** Logo của phầm mềm Visual studio code

Lý do chọn Visual Studio Code thay vì IDE Arduino mặc định:

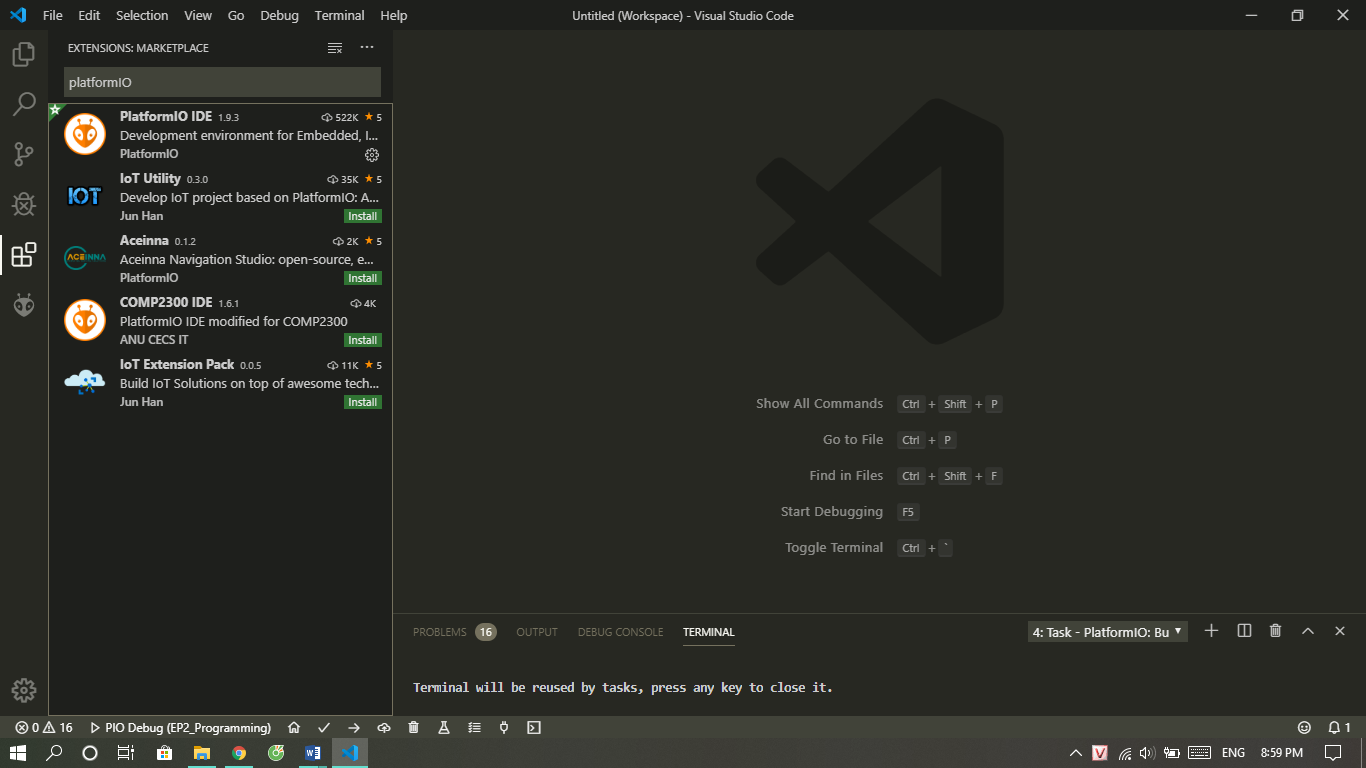
* Giao diện đơn giản, đẹp mắt.
* Gọn nhẹ, hoạt động mượt mà và ổn định.
* Trình quản lý package rất trực quan và phong phú.
* Gợi ý code thông minh, nâng cao hiệu suất lập trình rất nhiều, nhất là các chương trình quá nhiều dòng lệnh.
* Tự động tìm, sửa lỗi.
* Hỗ trợ lập trình rất nhiều loại vi điều khiển.



**Hình 3.4** Giao điện visual studio code khi mới khởi động.

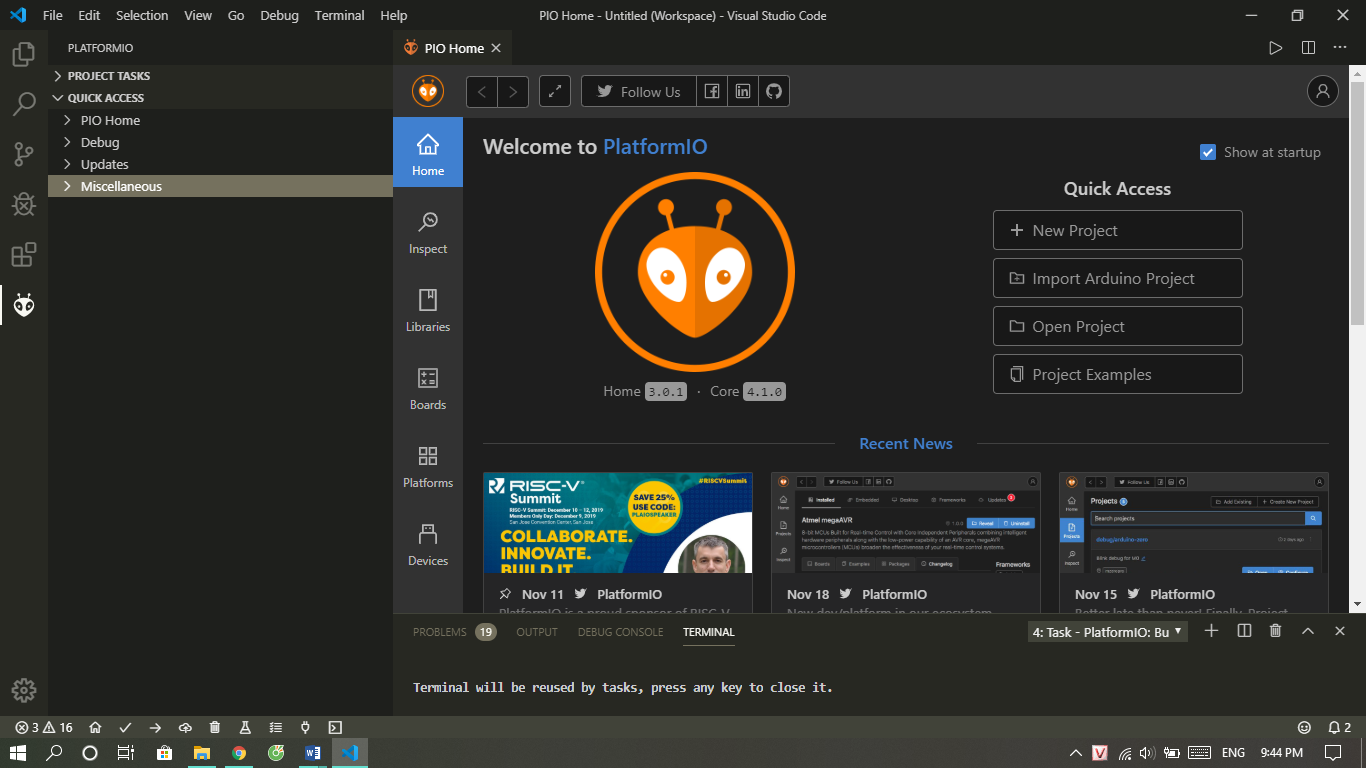
Tuy nhiên đề lập trình được cho arduino chúng ta cần phải cài thêm PlatformIO là một package mà bạn có thể cài đặt trên Visual Studio Code, sẽ thấy dễ dàng và thoải mái hơn, khi sử dụng PlatformIO trên Visual Studio Code để lập trình.

Để cài đặt FlatformIO chúng ta làm như sau: Click vào **Extensions** (biểu tượng 4 ô vuông trên thanh công cụ bên trái) 🡪 gõ từ khóa **FlatformIO** 🡪 Click **install**.



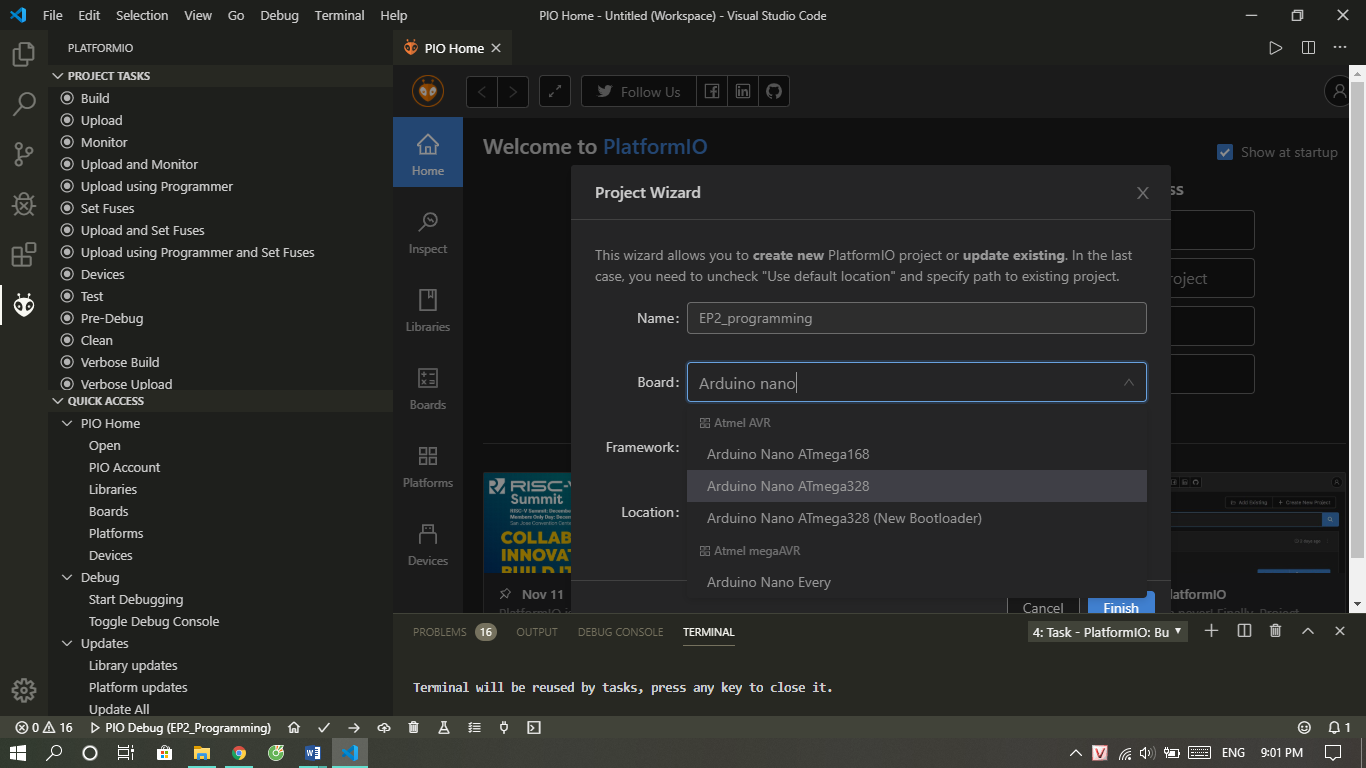
**Hình 3.5** Cài đặt FlatformIO

Sau khi cài đặt FlatformIO, để tạo project mới ta làm như sau: Click vào biểu tượng **FlatformIO** (hình con bọ) 🡪 Click nút **Home** trên thanh công cụ nằm ngang bên dưới, giao diện hiện lên như hình.



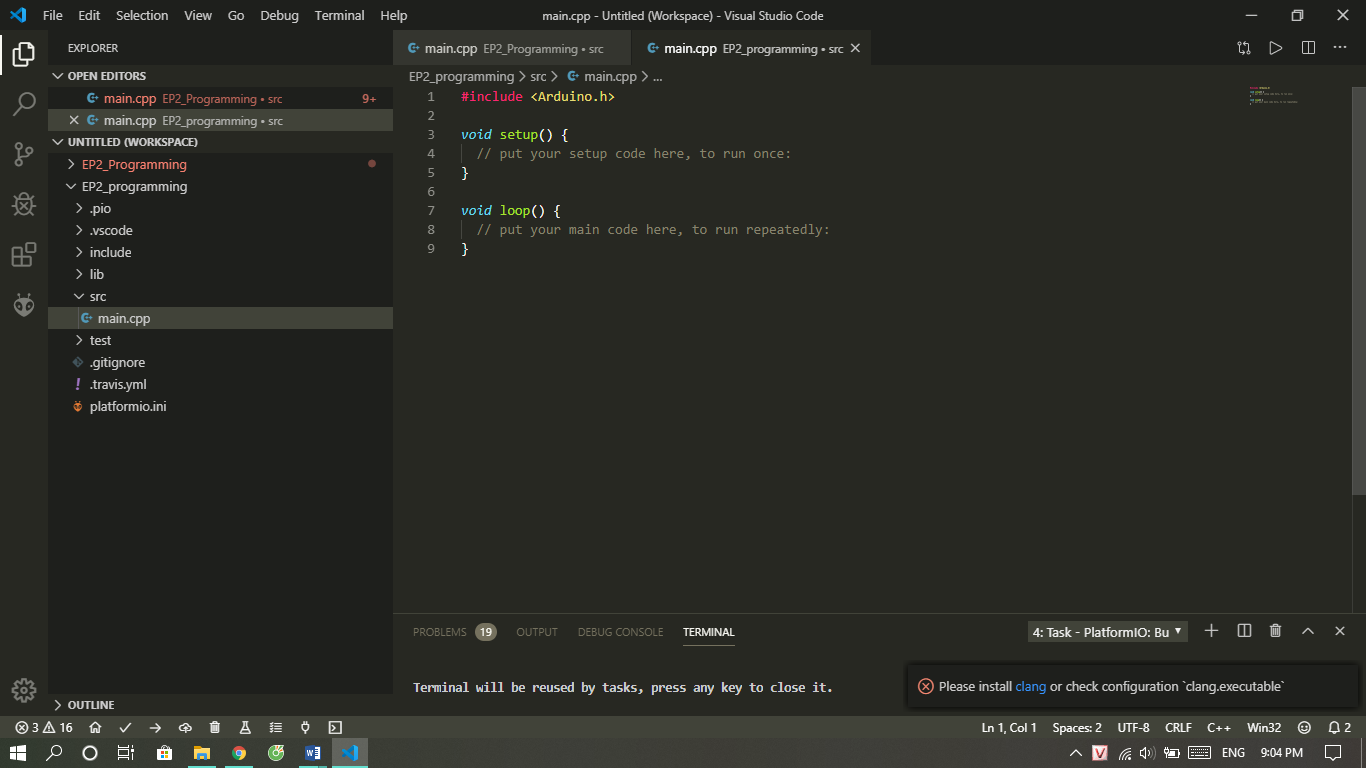
**Hình 3.6** Giao điện FlatformIO

Click **New Project** 🡪 Điền các thông tin về project 🡪 chọn **Finish.**



**Hình 3.7** Tạo dự án mới bằng FlatformIO

Sau khi tạo xong project giao diện lập trình sẽ hiển thị như hình sau:



**Hình 3.8** Giao diện khi tạo project thành công

**Lưu ý:** Nếu khi nạp chương trình mà gặp sự cố thì có thể chép toàn bộ chương trình sang IDE măc định để nạp, việc này không mất quá nhiều thời gian. Hơn nữa vì ngôn ngữ lập trình cho Arduino dựa trên nền tảng C/C++ nên bạn có thể đổi tên miền “.cpp” thành tên miền “.ino” để chuyển thành file arduino nhanh chóng.

### CHƯƠNG TRÌNH ĐIỀU KHIỂN

***Khai báo thư viện, định nghĩa các chân và từ khóa và cấu hình cho LCD, phím ma trận, Hx711.***

/\*

--- HCMUTE - FACULTY OF ELECTRICAL AND ELECTRONIC ENGINEERING ---

@Date: 11/2019

@Project: Electronic Scale

@Author: Nguyen Tan Hung

Nguyen Minh Tuan

@Consultor: Ths. Nguyen Dinh Phu

\*/

#include "Arduino.h"

#include "HX711.h"

#include <LiquidCrystal.h>

#include <EEPROM.h>

#include <Keypad.h>

#include <string.h>

#define ALERT 13

//DEFINE HX711 PIN

#define HX711\_DATA A0

#define HX711\_SCK A1

//DEFINE LCD PIN

#define LCD\_D4 9

#define LCD\_D5 8

#define LCD\_D6 7

#define LCD\_D7 6

#define E 10

#define RW 11

#define RS 12

//DEFINE MATRIX KEY PIN

#define ROW0 5

#define ROW1 4

#define ROW2 3

#define ROW3 2

#define COL0 A2

#define COL1 A3

#define COL2 A4

#define COL3 A5

//DEFINE ADDRESS ON EEPROM

#define ADDR\_PD\_PRICE 0 // FROM 0 TO 29

#define ADDR\_PD\_KIND 30 // FROM 30 TO 39

#define ADDR\_DF\_PRICE 40 // FROM 40 TO 42

#define ADDR\_DF\_KIND 43

#define ADDR\_MODE 44

#define ADDR\_DF 45

#define ADDR\_ID 46

#define ADDR\_ZERO 50 // FROM 50 TO 52

#define ADDR\_SGN\_ZERO 53

#define ADDR\_SAMPLE 54 // FROM 54 TO 56

//DEFINE KEY VALUE

#define CLEAR 12

#define UNDO 13

#define ENTER 11

#define CANCEL 10

#define KIND 15

#define DEFAULT 15

#define CORRECT 14

#define EDIT 10

// CONFIGURE LCD

LiquidCrystal lcd**(**RS**,** RW**,** E**,** LCD\_D4**,**LCD\_D5**,** LCD\_D6**,** LCD\_D7**);**

// CONFIGURE AND DECLARE GLOBAL VARIABLE FOR HX711

HX711 scale**(**HX711\_DATA**,** HX711\_SCK**);**

float zero**,** weight**,** amount**;**

float sample**;**

bool sgn\_zero**;**

// CONFIGURE AND DECLARE GLOBEL VARIABLE FOR MATRIX KEY

unsigned char key **=** 16**,** key\_value **;**

const byte col\_array**[**4**]={**COL0**,**COL1**,**COL2**,**COL3**};**

const byte row\_array**[**4**]={**ROW0**,**ROW1**,**ROW2**,**ROW3**};**

const unsigned char key\_array**[]={**16**,** 1**,** 2**,** 3**,** 10**,** 4**,** 5**,** 6**,** 11**,** 7**,** 8**,** 9**,** 12**,** 14**,** 0**,** 15**,** 13 **};**

char key\_map**[**4**][**4**]** **={** **{**1**,** 2**,** 3**,** 4**},**

**{**5**,** 6**,** 7**,** 8**},**

**{**9**,** 10**,** 11**,** 12**},**

**{**13**,** 14**,** 15**,** 16**}};**

Keypad keypad **=** Keypad**(**makeKeymap**(**key\_map**),** row\_array**,** col\_array**,** 4**,** 4**);**

// PRODUCT'S DATA AND DECLARE GLOBAL VARIABLE

const char**\*** product\_array**[**10**]={**"Beef"**,**"Pork"**,**"Fish"**,**"Chicken"**,**"Banana"**,**"Tomato"**,**"Egg"**,**"Milk"**,**"Pen"**,**"Book"**};**

unsigned long int price\_array**[**10**]={};**

bool kind\_array**[**10**]={};**

int id**;**

char**\*** product**;**

unsigned long int pd\_price**,** tmp\_price**,** total**,**df\_price**,** pass**;**

bool df\_kind**,** mode**,** pd\_kind**,** default\_**,** kind\_tmp**;**

***Chương trình đọc khối lượng đã chuyển đổi từ Hx711.***

// ######################## SUB-FUNCTION ######################

float transfer\_weight**(**int n**)**

**{**

float weight**;**

weight **=** scale**.**get\_units**(**n**)** **-** zero**;**

**if(** weight **<** 0**)** weight **=**0**;**

**return** weight**;**

**}**

***Chương trình chuông cảnh báo.***

void alert**(**int time**)**

**{**

digitalWrite**(**ALERT**,** HIGH**);**

delay**(**time**);**

digitalWrite**(**ALERT**,** LOW**);**

**}**

***Chương trình ghi 3 byte lên EEPROM***

void eeprom\_write\_3byte**(**int min\_addr**,** unsigned long int value**)**

**{**

EEPROM**.**write**(**min\_addr**,** value**/**65536**);**

delay**(**5**);**

EEPROM**.**write**(**min\_addr **+** 1**,** **(**value**%**65536**)/**256**);**

delay**(**5**);**

EEPROM**.**write**(**min\_addr **+** 2**,** value**%**256**);**

delay**(**5**);**

**}**

***Chương trình đọc 3 byte trên EEPROM***

unsigned long int eeprom\_read\_3byte**(**int min\_addr**)**

**{**

unsigned long int value **=** 0**;**

value **+=** **static\_cast<**unsigned long int**>** **(**EEPROM**.**read**(**min\_addr**))\***65536**;**

delay**(**5**);**

value **+=** **static\_cast<**unsigned long int**>** **(**EEPROM**.**read**(**min\_addr **+** 1**))\***256**;**

delay**(**5**);**

value **+=** **static\_cast<**unsigned long int**>** **(**EEPROM**.**read**(**min\_addr **+** 2**));**

delay**(**5**);**

**return** **static\_cast<**unsigned long int**>** **(**value**);**

**}**

***Chương trình hiển thị số dưới dạng tiền và căn lề phải***

void lcd\_money\_right\_alignment**(**int lcd\_col**,** int lcd\_row**,** long int number**)**

**{**

**if(**number **<** 1000000000**)**

**{** int size **=** 1**;**

long int diff **=** 1**,** number\_copy **=** number**;**

**for(**number\_copy**;** number\_copy **>** 9**;** number\_copy **/=** 10**)**

**{**

size**++;**

diff **\*=** 10**;**

**}**

**if(**size **<** 4**)** size **=** size**;**

**else** **if(**size **<** 7**)** size**++;**

**else** size **+=** 2**;**

lcd**.**setCursor**(**lcd\_col**,** lcd\_row**);**

**for(**int i **=** 0**;** i **<** 19 **-** lcd\_col **-** size**;** i**++)** lcd**.**print**(**" "**);**

lcd**.**setCursor**(**19**,** lcd\_row**);**

lcd**.**print**(**'d'**);**

lcd**.**setCursor**(**19 **-** size**,** lcd\_row**);**

number\_copy **=** number**;**

**while(**size **>** 0**)**

**{**

**if(**size**%**4 **==** 0**)** lcd**.**print**(**','**);**

**else**

**{**

lcd**.**print**(**number**/**diff**);**

number **%=** diff**;**

diff **/=** 10**;**

**}**

size**--;**

**}**

**}**

**}**

***Chương trình LCD hiển thị chữ căn lề phải***

void lcd\_text\_right\_alignment**(**int lcd\_col**,** int lcd\_row**,** char**\*** string**)**

**{**

int size**;**

size**=** strlen**(**string**);**

lcd**.**setCursor**(**lcd\_col**,**lcd\_row**);**

**for(**int I **=** 0**;**I **<** 20 **–** size **-** lcd\_col**;** i**++)** lcd**.**print**(**" "**);**

lcd**.**print**(**string**);**

**}**

***Chương trình LCD ở chế độ đếm số lượng***

void lcd\_counter**(**void**)**

**{**

lcd**.**setCursor**(**3**,** 0**);** lcd**.**print**(**"Counter:"**);**

lcd**.**setCursor**(**0**,** 1**);** lcd**.**print**(**"Price:"**);**

lcd**.**setCursor**(**0**,** 2**);** lcd**.**print**(**"Amount:"**);**

lcd**.**setCursor**(**0**,** 3**);** lcd**.**print**(**"Total:"**);**

**if(**default\_ **==** 0**)**

**{**

lcd**.**setCursor**(**12**,** 0**);** lcd**.**print**(**product**);**

lcd\_money\_right\_alignment**(**7**,** 1**,** pd\_price**);**

**}else**

**{**

lcd**.**setCursor**(**12**,** 0**);** lcd**.**print**(**"Default"**);**

lcd\_money\_right\_alignment**(**7**,** 1**,** df\_price**);**

**}**

**}**

***Chương trình LCD ở chế độ cân khối lượng***

void lcd\_weight**(**void**)**

**{**

lcd**.**setCursor**(**4**,** 0**);** lcd**.**print**(**"Weigh:"**);**

lcd**.**setCursor**(**0**,** 1**);** lcd**.**print**(**"Price:"**);**

lcd**.**setCursor**(**0**,** 2**);** lcd**.**print**(**"Weight:"**);**

lcd**.**setCursor**(**0**,** 3**);** lcd**.**print**(**"Total:"**);**

**if(**default\_ **==** 0**)**

**{**

lcd**.**setCursor**(**11**,** 0**);** lcd**.**print**(**product**);**

lcd\_money\_right\_alignment**(**7**,** 1**,** pd\_price**);**

**}else**

**{**

lcd**.**setCursor**(**11**,** 0**);** lcd**.**print**(**"Default"**);**

lcd\_money\_right\_alignment**(**7**,** 1**,** df\_price**);**

**}**

**}**

***Chương trình xử lý khối lượng đọc được từ Hx711 dựa vào cấu hình cân hiện tại và hiển thị tương ứng.***

void lcd\_calculation**(** float weight**)**

**{**

**if** **(**default\_ **==** 1**)**

**{**

**if(**df\_kind **==** 0**)**

**{**

total **=** round**(**weight**\***df\_price**/**1000**);**

lcd\_money\_right\_alignment**(**7**,** 3**,** total**);**

**if(**weight **<** 1000**)**

**{**

lcd\_text\_right\_alignment**(**10**,** 2**,** "Gam"**);**

**if(**weight **>=** 100**)** lcd**.**setCursor**(**10**,** 2**);**

**else** **if** **(**weight **>=** 10**)** lcd**.**setCursor**(**11**,** 2**);**

**else** lcd**.**setCursor**(**12**,** 2**);**

lcd**.**print**(**weight**);**

**}else**

**{**

weight **=** weight**/**1000**;**

lcd\_text\_right\_alignment**(**10**,** 2**,** "Kg"**);**

lcd**.**setCursor**(**13**,** 2**);** lcd**.**print**(**weight**);**

**}**

**}else**

**{**

amount **=** round**(**weight**/(**sample **+** 0.0001**));**

**if(** amount **<** 1000**)**

**{**

total **=** df\_price**\***amount**;**

lcd**.**setCursor**(**15**,** 2**);** lcd**.**print**(**" "**);**

**if(**amount **>=** 100**)** lcd**.**setCursor**(**17**,** 2**);**

**else** **if** **(**amount **>** 9**)** lcd**.**setCursor**(**18**,** 2**);**

**else** lcd**.**setCursor**(**19**,** 2**);**

lcd**.**print**(static\_cast<**int**>(**amount**));**

lcd\_money\_right\_alignment**(**7**,** 3**,** total**);**

**}else**

**{**

lcd\_text\_right\_alignment**(**15**,** 2**,** "---"**);**

lcd\_text\_right\_alignment**(**7**,** 3**,** "---"**);**

**}**

**}**

**}else**

**{**

**if(**pd\_kind **==** 0**)**

**{**

total **=** round**(**weight**\***pd\_price**/**1000**);**

lcd\_money\_right\_alignment**(**7**,** 3**,** total**);**

**if(**weight**<**1000**)**

**{**

lcd\_text\_right\_alignment**(**10**,** 2**,** "Gam"**);**

**if(**weight **>=** 100**)** lcd**.**setCursor**(**10**,** 2**);**

**else** **if** **(**weight **>=** 10**)** lcd**.**setCursor**(**11**,** 2**);**

**else** lcd**.**setCursor**(**12**,** 2**);**

lcd**.**print**(**weight**);**

**}else**

**{**

weight **=** weight**/**1000**;**

lcd\_text\_right\_alignment**(**10**,** 2**,** "Kg"**);**

lcd**.**setCursor**(**13**,** 2**);** lcd**.**print**(**weight**);**

**}**

**}else**

**{**

amount **=** round**(**weight**/(**sample **+** 0.0001**));**

**if** **(**amount **<** 1000**)**

**{**

total **=** pd\_price**\***amount**;**

lcd**.**setCursor**(**15**,** 2**);** lcd**.**print**(**" "**);**

**if(**amount **>=** 100**)** lcd**.**setCursor**(**17**,** 2**);**

**else** **if** **(**amount **>=** 10**)** lcd**.**setCursor**(**18**,** 2**);**

**else** lcd**.**setCursor**(**19**,** 2**);**

lcd**.**print**(static\_cast<**long int**>(**amount**));**

lcd\_money\_right\_alignment**(**7**,** 3**,** total**);**

**}else**

**{**

lcd\_text\_right\_alignment**(**15**,** 2**,** "---"**);**

lcd\_text\_right\_alignment**(**7**,** 3**,** "---"**);**

**}**

**}**

**}**

**}**

***Chương trình LCD ở chế độ cài đặt***

void lcd\_setting**(**void**)**

**{**

lcd**.**setCursor**(**5**,** 0**);** lcd**.**print**(**"Setting"**);**

lcd**.**setCursor**(**0**,** 1**);** lcd**.**print**(**"Product:"**);**

lcd**.**setCursor**(**0**,** 2**);** lcd**.**print**(**"Mode:"**);**

lcd**.**setCursor**(**0**,** 3**);** lcd**.**print**(**"Price:"**);**

**if(**default\_ **==** 0**)** lcd\_text\_right\_alignment**(**10**,** 1**,** product**);**

**else** lcd\_text\_right\_alignment**(**10**,** 1**,** "default"**);**

**if(**kind\_tmp **==** 0**)** lcd\_text\_right\_alignment**(**10**,** 2**,** "weigh"**);**

**else** lcd\_text\_right\_alignment**(**10**,** 2**,** "counter"**);**

lcd\_money\_right\_alignment**(**8**,** 3**,** tmp\_price**);**

**}**

***Chương trình hiệu ứng LCD khi khởi động***

void lcd\_welcome**(**void**)**

**{**

char project**[]** **=** **{**'D'**,**'O'**,**' '**,**'A'**,**'N'**,**' '**,**'2'**,**':'**,**' '**,**'C'**,**'A'**,**'N'**,**' '**,**'D'**,**'I'**,**'E'**,**'N'**,**' '**,**'T'**,**'U'**};**

lcd**.**setCursor**(**4**,** 1**);** lcd**.**print**(**"HCMUTE-FEEE"**);**

lcd**.**setCursor**(**0**,** 2**);**

**for(**int i **=** 0**;** i **<** **sizeof(**project**);** i**++)**

**{**

lcd**.**print**(**project**[**i**]);** delay**(**90**);**

**}**

delay**(**1000**);**

lcd**.**clear**();**

lcd**.**setCursor**(**0**,** 0**);** lcd**.**print**(**"Author:"**);**

**for(**int i **=** 3**;** i **>** 0**;** i**--)**

**{**

lcd**.**setCursor**(**1**,** i**);** lcd**.**print**(**"1. NGUYEN TAN HUNG"**);**

delay**(**500**);**

**if(**i **>** 1**)**

**{**

lcd\_text\_right\_alignment**(**1**,** i**,** " "**);**

**}**

**}**

**for(**int i **=** 3**;** i **>** 1**;** i**--)**

**{**

lcd**.**setCursor**(**1**,** i**);** lcd**.**print**(**"2. NGUYEN MINH TUAN"**);**

delay**(**500**);**

**if(**i **>** 2**)**

**{**

lcd\_text\_right\_alignment**(**1**,** i**,** " "**);**

**}**

**}**

delay**(**1000**);**

lcd**.**clear**();**

lcd**.**setCursor**(**0**,** 1**);** lcd**.**print**(**"Consultor:"**);**

**for(**int i **=** 0**;** i **<** 5**;** i**++)**

**{**

**if(**i**%**2 **==** 0**)**

**{**

lcd**.**setCursor**(**0**,** 2**);** lcd**.**print**(**"Ths. NGUYEN DINH PHU"**);**

**}else**

**{**

lcd\_text\_right\_alignment**(**0**,** 2**,** " "**);**

**}**

delay**(**500**);**

**}**

delay**(**1000**);**

**}**

***Chương trình LCD tự động thay đổi hiển thị khi cấu hình thay đổi.***

void lcd\_refresh**()**

**{**

lcd**.**clear**();**

**if** **(**mode **==** 0**)**

**{**

**if** **(**default\_ **==** 1**)**

**{**

**if** **(**df\_kind **==** 0**)** lcd\_weight**();**

**else** lcd\_counter**();**

**}else**

**{**

**if** **(**pd\_kind **==** 0**)** lcd\_weight**();**

**else** lcd\_counter**();**

**}**

**}else** lcd\_setting**();**

**}**

***Chương trình LCD lúc hiệu chỉnh cân.***

void lcd\_correct**(**bool type**)**

**{**

**if** **(**type **==** 0**)**

**{**

lcd**.**clear**();**

char**\*** wait**[**3**]** **=** **{**". "**,**".. "**,**"..."**};**

lcd**.**setCursor**(**3**,** 1**);**

lcd**.**print**(**"Correcting"**);**

**for(**int i **=** 0**;** i **<** 3**;** i**++)**

**{**

lcd**.**setCursor**(**13**,** 1**);**

lcd**.**print**(**wait**[**i**]);**

delay**(**500**);**

**}**

zero **=** scale**.**get\_units**(**5**);**

**if(**zero **<** 0 **)** sgn\_zero **=** 1**;**

**else** sgn\_zero **=** 0**;**

eeprom\_write\_3byte**(**ADDR\_ZERO**,** abs**(**zero**));**

EEPROM**.**write**(**ADDR\_SGN\_ZERO**,** sgn\_zero**);**

lcd**.**setCursor**(**5**,** 2**);**

lcd**.**print**(**"Completed!"**);**

delay**(**400**);**

**}**

**else**

**{**

lcd**.**clear**();**

char**\*** wait**[**3**]** **=** **{**". "**,**".. "**,**"..."**};**

lcd**.**setCursor**(**2**,**1**);**

lcd**.**print**(**"Taking sample"**);**

**for(**int i **=** 0**;** i **<** 3**;** i**++)**

**{**

lcd**.**setCursor**(**15**,** 1**);**

lcd**.**print**(**wait**[**i**]);**

delay**(**500**);**

**}**

sample **=** transfer\_weight**(**5**);**

eeprom\_write\_3byte**(**ADDR\_SAMPLE**,** sample**);**

lcd**.**setCursor**(**5**,** 2**);**

lcd**.**print**(**"Completed!"**);**

delay**(**400**);**

**}**

**}**

***Chương trình đọc giá trị từ phím ma trận.***

unsigned char get\_key**()**

**{**

unsigned char key **=** 0**;**

unsigned char temp **=** keypad**.**getKey**();**

**if((**int**)**keypad**.**getState**()** **==** PRESSED**)**

**{**

**if(**temp **!=** 0**)** key **=** temp**;**

**}**

**return** key**;**

**}**

***Chương trình thực thi phím nhấn khi ở chế độ làm việc***

void exe\_key\_when\_scale**(**unsigned char key\_value**)**

**{**

**if** **(**key\_value **==** DEFAULT **)**

**{**

alert**(**20**);**

**if(**default\_ **==** 0**)**

**{**

default\_ **=** 1**;**

EEPROM**.**write**(**ADDR\_DF**,** default\_**);**

**}else**

**{**

df\_kind **=** **!**df\_kind**;**

EEPROM**.**write**(**ADDR\_DF\_KIND**,** df\_kind**);**

**}**

lcd\_refresh**();**

**}**

**if** **(**key\_value **==** CORRECT**)**

**{**

alert**(**20**);**

**if** **(**default\_ **==** 1**)**

**{**

lcd\_correct**(**df\_kind**);**

**}else**

**{**

lcd\_correct**(**pd\_kind**);**

**}**

lcd\_refresh**();**

**}**

**if(**key\_value**==**EDIT**)**

**{**

alert**(**20**);**

mode **=** 1**;**

EEPROM**.**write**(**ADDR\_MODE**,** mode**);**

**if(**default\_ **==** 1**)**

**{**

tmp\_price **=** df\_price**;**

kind\_tmp **=** df\_kind**;**

**}**

**else**

**{**

tmp\_price **=** pd\_price**;**

kind\_tmp **=** pd\_kind**;**

**}**

lcd\_refresh**();**

**}**

**if(**key\_value**<**10**)**

**{**

alert**(**20**);**

id**=**key\_value**;**

EEPROM**.**write**(**ADDR\_ID**,** id**);**

pd\_price **=** price\_array**[**id**];**

pd\_kind**=** kind\_array**[**id**];**

product **=** product\_array**[**id**];**

default\_ **=** 0**;**

EEPROM**.**write**(**ADDR\_DF**,** default\_**);**

lcd\_refresh**();**

**}**

**}**

***Chương trình thực thi phím nhấn khi ở chế độ cài đặt***

void exe\_key\_when\_setting**(**unsigned char key\_value**)**

**{**

char**\*** wait**[**3**]={**". "**,**".. "**,**"..."**};**

**for(**int i **=** 0**;** i **<** 3**;** i**++)**

**{**

lcd**.**setCursor**(**12**,** 0**);**

lcd**.**print**(**wait**[**i**]);**

delay**(**20**);**

**}**

**if(**key\_value **==** CLEAR**)**

**{**

tmp\_price **=** 0**;**

**if(**default\_ **==** 1**)** kind\_tmp **=** df\_kind**;**

**else** kind\_tmp **=** pd\_kind**;**

lcd\_money\_right\_alignment**(**8**,**3**,**tmp\_price**);**

**if(** kind\_tmp **==** 0**)** lcd\_text\_right\_alignment**(**10**,** 2**,** "weigh"**);**

**else** lcd\_text\_right\_alignment**(**10**,** 2**,** "counter"**);**

**}**

**if(**key\_value **==** ENTER**)**

**{**

alert**(**20**);**

**if(**default\_ **==** 1**)**

**{**

df\_price **=** tmp\_price**;**

eeprom\_write\_3byte**(**ADDR\_DF\_PRICE**,** df\_price**);**

df\_kind **=** kind\_tmp**;**

EEPROM**.**write**(**ADDR\_DF\_KIND**,** df\_kind**);**

**}else**

**{**

pd\_price **=** tmp\_price**;**

eeprom\_write\_3byte**(**id**\***3**,** pd\_price**);**

pd\_kind **=** kind\_tmp**;**

EEPROM**.**write**(**ADDR\_PD\_KIND **+** id**,** pd\_kind**);**

**}**

mode **=**0**;**

EEPROM**.**write**(**ADDR\_MODE**,** mode**);**

lcd\_refresh**();**

**}**

**if(**key\_value**<**10**)**

**{**

**if(**tmp\_price**<=**100000**)** tmp\_price **=** tmp\_price**\***10 **+** key\_value**;**

lcd\_money\_right\_alignment**(**8**,** 3**,** tmp\_price**);**

**}**

**if(**key\_value **==** CANCEL**)**

**{**

alert**(**20**);**

mode **=** 0**;**

EEPROM**.**write**(**ADDR\_MODE**,** mode**);**

lcd\_refresh**();**

**}**

**if(**key\_value **==** UNDO**)**

**{**

tmp\_price **/=** 10**;**

lcd\_money\_right\_alignment**(**8**,** 3**,** tmp\_price**);**

**}**

**if(**key\_value **==** KIND**)**

**{**

kind\_tmp **=** **!**kind\_tmp**;**

**if(** kind\_tmp **==** 0**)** lcd\_text\_right\_alignment**(**10**,** 2**,** "weigh"**);**

**else** lcd\_text\_right\_alignment**(**10**,** 2**,** "counter"**);**

**}**

**}**

***Chương trình thiết lập và khởi tạo các giá trị ban đầu***

// ######################### MAIN FUNCTION ###################

void setup**()**

**{**

pinMode**(**ALERT**,** OUTPUT**);**

pinMode**(**RS**,** OUTPUT**);**

pinMode**(**RW**,** OUTPUT**);**

pinMode**(**LCD\_D4**,** OUTPUT**);**

pinMode**(**LCD\_D5**,** OUTPUT**);**

pinMode**(**LCD\_D6**,** OUTPUT**);**

pinMode**(**LCD\_D7**,** OUTPUT**);**

pinMode**(**ROW0**,** OUTPUT**);**

pinMode**(**ROW1**,** OUTPUT**);**

pinMode**(**ROW2**,** OUTPUT**);**

pinMode**(**ROW3**,** OUTPUT**);**

pinMode**(**COL0**,** INPUT**);**

pinMode**(**COL1**,** INPUT**);**

pinMode**(**COL2**,** INPUT**);**

pinMode**(**COL3**,** INPUT**);**

digitalWrite**(**ALERT**,** LOW**);**

//SETUP SCALE

scale**.**set\_scale**(**385.f**);**

scale**.**tare**();**

sample **=** eeprom\_read\_3byte**(**ADDR\_SAMPLE**);**

zero **=** eeprom\_read\_3byte**(**ADDR\_ZERO**);**

sgn\_zero **=** EEPROM**.**read**(**ADDR\_SGN\_ZERO**);**

**if** **(**sgn\_zero **==** 1**)** zero **=** **-**zero**;**

//SETUP DATABASE OF PRODUCT

default\_ **=** EEPROM**.**read**(**ADDR\_DF**);**

mode **=** EEPROM**.**read**(**ADDR\_MODE**);**

df\_kind **=** EEPROM**.**read**(**ADDR\_DF\_KIND**);**

df\_price **=** eeprom\_read\_3byte**(**ADDR\_DF\_PRICE**);**

**for(**int i **=** 0**;** i **<** 10**;** i**++)**

**{**

price\_array**[**i**]** **=** eeprom\_read\_3byte**(**i**\***3**);**

kind\_array**[**i**]** **=** EEPROM**.**read**(**ADDR\_PD\_KIND **+** i**);**

delay**(**5**);**

**}**

id **=** EEPROM**.**read**(**ADDR\_ID**);**

pd\_kind **=** kind\_array**[**id**];**

pd\_price **=** price\_array**[**id**];**

product **=** product\_array**[**id**];**

pass **=** 0**;**

total **=** 0**;**

//SETUP LCD

lcd**.**begin**(**20**,** 4**);**

lcd\_welcome**();**

lcd\_refresh**();**

**}**

***Chương trình chính***

void loop**()**

**{**

key **=** get\_key**();**

key\_value **=** key\_array**[**key**];**

**if(**mode **==** 0**)**

**{**

**if(**pass **<** 100000**)** pass**++;**

**else**

**{**

weight **=** transfer\_weight**(**5**);**

**if(**weight **>** 5000**)** alert**(**200**);**

lcd\_calculation**(**weight**);**

pass **=** 0**;**

**}**

exe\_key\_when\_scale**(**key\_value**);**

**}**

**else**

**{**

exe\_key\_when\_setting**(**key\_value**);**

**}**

**}**

# **KẾT QUẢ THỰC HIỆN – HƯỚNG PHÁT TRIỂN**

## **KẾT QUẢ THỰC HIỆN**

Qua quá trình nghiên cứu và thực hiện đề tài, nhóm thực hiện đề tài nhận thấy đề tài này có khả năng ứng dụng vào thực tế cao, đáp ứng được nhu cầu ngày càng cao của con người và đi đúng xu hướng của thời hiện đại. Đồng thời đề tài này cũng là nguồn tài liệu có giá trị cho những bạn sinh viên khóa tiếp theo có thể tham khảo khi nghiên cứu những đề tài liên quan. Bên cạnh đó chúng em cũng bổ sung cho mình những kiến thức hay và bổ ích.

Biết cách sử dụng Arduino Nano: Arduino Nano là một trong những board mạch thông minh được sử dụng rộng rãi. Tuy nhiên, để sử dụng thành thạo và đầy đủ các chức năng của nó thì không dễ. Trong đề tài này nhóm thực hiện khả năng hiển thị thông tin theo yêu cầu trên LCD 20x4, nhận thông tin và giải mã để sử dụng từ Keypad 4x4.

Biết cách sử dụng LCD 20x4 thông qua Arduino: LCD là một màn hình hiển thị đơn giản, giá thành rẻ và dễ sử dụng. Qua quá trình nghiên cứu, nhóm đã biết cách sử dụng LCD để hiển thị những thông tin mong muốn.

Biết cách sử dụng Keypad 4x4: Keypad 4x4 thực chất là một ma trận phím nhấn với 4 hàng, 4 cột, 16 phím nhấn. Qua quá trình nghiên, nhóm đã biết cách dùng Arduino để nhận dữ liệu phím nhấn từ Keypad sau đó xử lý mã phím để dùng cho mục đích mong muốn.

Biết cách chuyển đổi ADC thông qua module ADC 24bit HX711.

Nhận xét: Sau thời gian 15 tuần nghiên cứu và thực hiện đề tài, mạch cơ bản đã đáp ứng được yêu cầu thiết kế ban đầu: Giao diện thiết kế dễ sử dụng, tháo tác điều khiển đơn giản, mạch chạy đúng với yêu cầu.

## **HƯỚNG PHÁT TRIỂN**

* Thứ nhất: Mô hình cân điện tử này có thể thiết kế thêm phần giao tiếp UART với máy tính để kết nối với phần mềm quản lý bán hàng.
* Thứ hai: Sử dùng board arduino mega 2560, thay đổi màn hình hiển thị thành TFT để việc thao tác với cân thoải mái và hiển thị được nhiều thông tin hơn, cũng có thể thêm chức năng kết nối internet và truyền dữ liệu lên website dùng cho việc quản lý bán hàng từ xa.