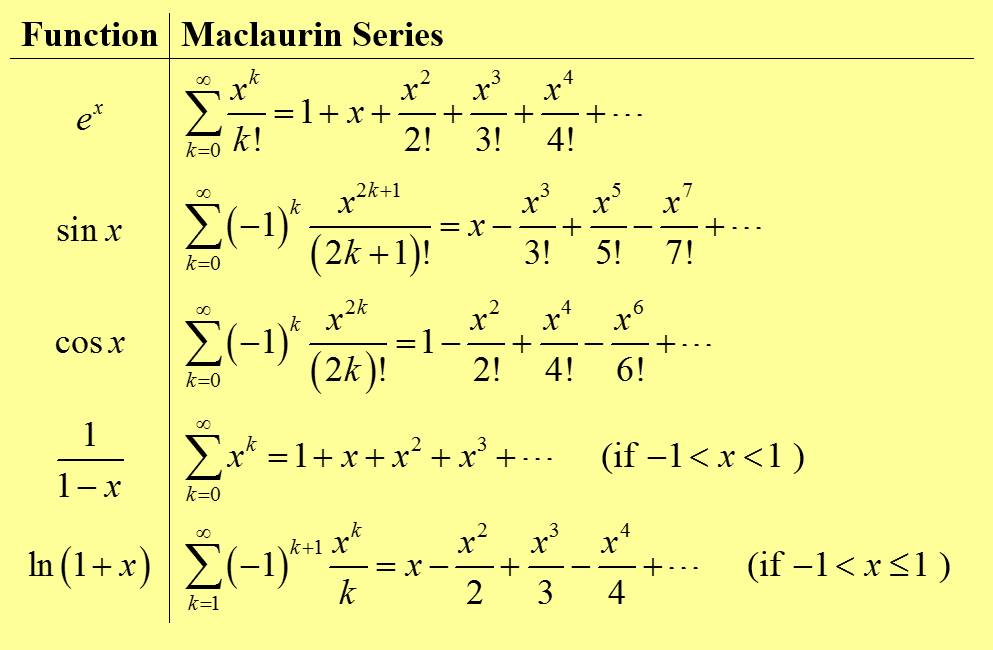
Bản chất của giải thuật là các vòng lặp. Bản chất của các hàm toán là các khai triển đại số. Một khai triển Maclaurin hầu như giải quyết hết các bài toán về hàm số bao gồm cả hàm vô tỷ và hàm tuần hoàn. Tuy nhiên, việc xây dựng các hàm truyền trên các IDE, bộ giải mã thường chỉ sử dụng tới vô cùng bé bậc 5 hoặc bậc 7, điều này xuất phát từ việc xử lý tính toán của vi xử lý nhỏ thường không cần độ chính xác cao, thay vào đó là độ ổn định và tính toán nhanh, và điều này thường khiến hệ thống mất tính chính xác khi xử lý những giải tích lượng giác.

VD: một esp8266 32 bit có thể giải quyết một tính toán 2^32 bit, thế nhưng trong các vi xử lý, họ chỉ xây dựng đến vô cùng bé bậc 5, khoảng 8 ký tự, 16 bit rất nhỏ so với 2^32. Đó là nguyên nhân gây ảnh hưởng đến các tính toán tức thời, bao gồm tính số đo góc cánh tay robot, drone và các ứng dụng khác.

Update: Việc đại số hóa các hàm lượng giác cho ra kết quả chính xác hơn so với sử dụng hàm lượng giác mà IDE cung cấp. Có thể hiểu như thế này, khi bạn vẽ một đường cong trên cánh tay robot, bạn phải tính n đường cong tương đương với n bậc của cánh tay. Nhưng trong trường hợp đại số, ta dùng vô cùng bé bậc thứ n tương đương với bậc của cánh tay để xác định vị trí chính xác trên tọa độ, thay vì phụ thuộc vào sai số của hàm do liên quan đến số pi. Một cách dễ hình dung, ta vẽ các đường thẳng vuông góc nhau cho đến khi cánh tay chạm đến điểm chỉ định. Và đường thẳng dễ xác định hơn đường cong. Đó là cách maclaurin hoạt động.



[**Đặng Lâm Tùng**](https://www.facebook.com/groups/1490715437881798/user/100008117010365/?__cft__%5b0%5d=AZUyRaI76yrHJQKcniXgNy9aZO1XnfTnlXJ2EFbvrsjg0XT91kq4LNFkQPvePOuNEi9f5IvKk-fPEeJalXcmSh8eMrsxjFu6z5zhdmpDj2UZ348FW9r9VQF5ztuI8NtiOSD9Li41Sx-RE4XslsaR8fhx&__tn__=R%5d-R) câu này em thấy nhiều người hay nói, nhưng việc tính toán đường bay đến mặt trăng được tính toán từ trước ở dưới mặt đất, AGC chủ yếu dùng để điều khiển và xử lý dữ liệu từ hệ thống trên tàu là chính. Nên yêu cầu tính toán của AGC không cần cao là vì vậy.

Mà AGC là trung tâm nhưng hệ thống điện và dây dẫn cũng rất quan trọng nữa.

Tùy theo bài toán yêu cầu độ chính xác bao nhiêu và thời gian tính toán. Nếu mcu đủ tài nguyên cpu, tại sao mình lại không tự implement function đó?

Không bạn ạ, thực tế vẫn có số double (64 bit) trên mấy dòng esp, và các phép tính sin, cos, tan,.. Vẫn có double

Tuy nhiên theo như mình biết thì các dòng esp không có FPU ( floating point unit) nên đôi khi tính toán lũy thừa số thập phân là cực kì lâu. Chưa tính tới vấn đề khác.

Trên arm thì họ đã lược bớt và dùng arm math, quy định số float từ range -1 -> 1 tương đương, để tính các phép toán như sin cos, sqrt, .. Tuy nhiên vẫn khá chậm trên các dòng không có fpu

Mình k biết về fpu. Có lẽ là giống xử lý phân luồng. Nhưng việc đại số hóa các hàm lượng giác cho ra kết quả chính xác hơn so với sử dụng hàm lượng giác mà IDE cung cấp. Có thể hiểu như thế này, khi bạn vẽ một đường cong trên cánh tay robot, bạn phải tính n đường cong tương đương với n bậc của cánh tay. Nhưng trong trường hợp đại số, ta dùng vô cùng bé bậc thứ n tương đương với bậc của cánh tay để xác định vị trí chính xác trên tọa độ, thay vì phụ thuộc vào sai số của hàm do liên quan đến số pi. Một cách dễ hình dung, ta vẽ các đường thẳng vuông góc nhau cho đến khi cánh tay chạm đến điểm chỉ định. Và đường thẳng dễ xác định hơn đường cong. Đó là cách maclaurin hoạt động.

Vậy thì bạn nên tìm hiểu về FPU, nó không liên quan đến luồng, nó là tập lệnh hỗ trợ cho số chấm động (float/doube) cái mà bạn cần hướng đến

Các hàm lượng giác mà compiler cung cấp người ta thường dùng thư viện chuẩn (standard lib), nếu bạn cần độ chính xác cao hơn thì có thể tự build và đo tốc độ, và có thể giải thích vì sao họ giới hạn ở đó. Đối vs esp32 để xử lý float đến bậc 7 thì cũng khá lâu đó

Thú thật đọc cũng ko hiểu gì vì quá tầm nhưng mình thấy giả sử dùng floating point sau thập phân 5 tới 6 số là đủ cho tới phần triệu rồi. Hệ robot thông thường sai số lặp lại tới phần ngàn là chúa tể robot rồi, trong khi nếu để tới cái vòng abcxyz kia nó ngốn hết Ram của chip khi phải lưu và xử lý đám số thừa kia .Điều bạn đề cập chắc giành cho những dòng chip cao hơn, mắc tiền và chuyên dụng cho những ứng dụng cao cấp. Ví dụ như cái máy khắc nano in thạch bản dùng để sản xuất chip tới công nghệ xx nano met vừa rồi Mỹ cấm bán cho Trung quốc đấy. ESP là của china mà nó không có máy nó chạy tới hàng triệu thôi.

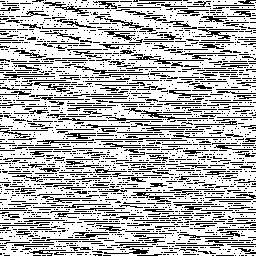
Các cụ tính mô phỏng matlab đòi sai số nhỏ như con vi trùng nhưng ra hardware hay cấu truc cơ khí có cần mịn thế đâu, nhiều khi muốn cũng chẳng được

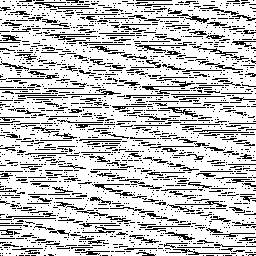
Dành cho các con dân chế cháo mà dính tới mật mã, nhất là giao thức GET.

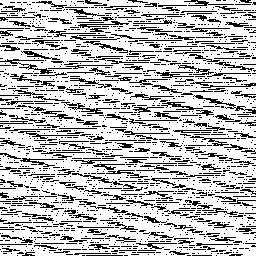
4 hình trên là mẫu random bit từ cổng A0 với điện áp tham chiếu 5v.

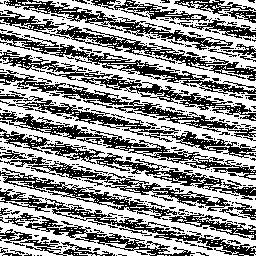
2 hình dưới là bộ mẫu lấy từ trích xuất Von Neumann.

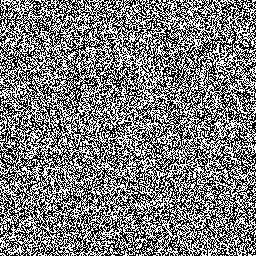
Về cơ bản thì arduino rất khó đưa về trạng thái chồng chập để tạo lát cắt bit, mà cắt nhiều bit thì lại chậm, nguyên nhân xuất phát từ khoảng điện áp tham chiếu trải rộng từ 0 đến 5v mà không có một nguồn phát ngẫu nhiên nào đáp ứng được cái dải áp này cả. Nhưng với tốc độ lấy mẫu cao thì việc tràn entropy trong vi xử lý là điều dễ hiểu, kết quả là tất cả các bộ rand() xảy ra hai trường hợp, hoặc là bộ số không bao giờ thay đổi, hoặc vi xử lý luôn trong tình trạng tràn entropy dẫn đến vi xử lý siêu nóng làm giảm hiệu suất vxl. Giải pháp đưa ra là dùng bộ nhiễu trên con gia tốc MCU6050 làm nguồn trích xuất mẫu Von Neumann. Phân bố vừa đều vừa đẹp. Toẹt dzời.

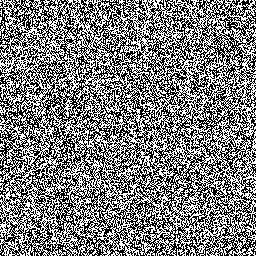












[**Lê Công Văn**](https://www.facebook.com/groups/1490715437881798/user/100008692519387/?__cft__%5b0%5d=AZVzb4hmaEzHSmWL10O80jrKLpuMQ7MDOJvPJNzd81FXSzOALiwHVq6Yc5AJZiVEvd4mTiBpecFi2NPj8L8mHLwtJMG59voj2xwy1k-hzVjfDnVFFHmvA-CSvNmIRUiwAxd6qcj42cxn-vgPsSd2ZjME&__tn__=R%5d-R) mỗi chấm đen là bit 1, chấm trắng là bit 0. Món này không khoai và cần ạ.

À, trên thuyết về quantum có một trò chơi điền vào ô trống. Em đang liên tưởng tới chúng.

Bữa mình có nghiên cứu dùng cách mạch tạo xung nhọn sau đó đọc cái xung đó bằng chân A0 + với random() của Arduino nó ra kết quả cũng gần ngẫu nhiên nhưng mình vẫn có thể dự đoán được nó trả về giá trị cao hay thấp theo xung nhận được

[**Bão Vjper**](https://www.facebook.com/groups/1490715437881798/user/100004374550540/?__cft__%5b0%5d=AZVzb4hmaEzHSmWL10O80jrKLpuMQ7MDOJvPJNzd81FXSzOALiwHVq6Yc5AJZiVEvd4mTiBpecFi2NPj8L8mHLwtJMG59voj2xwy1k-hzVjfDnVFFHmvA-CSvNmIRUiwAxd6qcj42cxn-vgPsSd2ZjME&__tn__=R%5d-R) nếu có một nguồn điện từ được mã hóa phù hợp với nguyên tắc 0, 5 v gần vi điều khiển, có thể kiểm soát được bộ số ngẫu nhiên đọc trên chân A0

Chào các bạn, hiện nay các vấn đề về tỏa nhiệt trong mạch điện tử CMOS là một vấn đề nan giải với rất nhiều bạn. Một trong những vấn đề này đến từ việc chồng chéo kiến thức giữa FET, trường cổng nổi, áp, và logic. Chuyện tỏa nhiệt không phải là hiếm, nhưng chuyện để tỏa nhiệt như một "tính năng", điển hình là trong con V Active 3 thò thụt nào ấy là một ví dụ, hay trong các mạch điều khiển ô tô không chỉ gây bất tiện cho người dùng mà còn làm cho nguy cơ cháy nổ tăng lên. Nên mình viết bài này để giải trí. Mong giúp ích cho mọi người.

Về cơ bản: nhiệt trong CMOS được chia làm nhiệt tĩnh. nhiệt sạc xả, và nhiệt ngắn mạch.

Nhiệt tĩnh liên quan tới một giá trị là điện áp nền. Điện áp này có nhiệm vụ kéo electron lên một giá trị đủ để tạo một mật độ đủ để chúng có thể tạo dòng. Hình dung giống như hai lớp của một mặt phẳng và tùy vào áp được cung cấp ở 5 hoặc 12V, chúng đều tạo ra một vùng hoạt động và vùng khan hiếm, thường thì vùng khan hiếm sẽ nằm dưới cùng, tiếp đều là vùng hoạt động, tiếp đến là một lớp oxit ngăn cách giữa áp nền và vùng hoạt động. Nếu lớp oxit đủ dày, chúng sẽ không có dòng rò, nhưng ở dưới 130nm tương đương với 20A, dòng rò đủ để giảm hiệu suất của CMOS nói chung gấp nhiều lần.

Nhiệt sạc xả, liên quan trực tiếp đến tần số hoạt động của CMOS: P =\alpha CV^{2}f.

Crowbar: Giống như một kiến trúc phân tầng, một đầu ra logic sẽ không cho phép một dòng hoặc một áp quá nhiệt chạy qua. Các kiến trúc này đảm bảo sự cách ly đầy đủ cho lõi, đóng vai trò xử lý và cổng logic, đóng vai trò kết nối với các module khác. Thế nên chúng luôn sinh ra một lượng nhiệt do "sự cách li" giữa bộ xử lý, vốn nhạy cảm với dòng và cổng logic, vốn nhạy cảm với áp.

Cái mình hay gặp nhất đó là khi thiết kế hoặc thi công mạch, cái mà cách bạn đưa vào không chỉ là áp mà còn là dòng. Thậm chí có trường hợp, tín hiệu xử lý liên module, hay giữa các module khác nhau, không khác gì bật tắt một thiết bị cả. Đối với tín hiệu, chúng ta cần là áp nhiều hơn là dòng, và cách để duy trì áp không dòng đó là tạo một cầu diot ngược- biện pháp đơn giản nhất. Chúng thật sự chỉ duy trì điện áp mà không cho dòng chảy qua. Đáp ứng được các yêu cầu về tín hiệu cho thiết bị.

Chứng tỏ b cũng nghiên cứu khá kĩ vấn đề này, những ai làm thiết kế đều lo về chuyện này, nhưng ngày nay có vẻ nó không thật sự được chú tâm tới.

Bởi thiết bị gần như được thiết kế và sx đại trà, cái mà trên 70% dân kĩ thuật dùng là cả một nền tảng TQ đã áp dụng lên.

Còn số ít nghiên cứu mò mẫm lối đi riêng nhưng cũng rất dễ bị mất sức do lội ngược dòng.

Tất nhiên tinh thần ấy vẫn rất hoan nghênh.