

ĐẠI HỌC QUỐC GIA TP HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA



**BÀI TẬP LỚN
MÔN ĐO LUỒNG CÔNG NGHIỆP
ĐỀ TÀI**

**THIẾT KẾ MẠCH ĐO VÀ HIỂN THỊ KHOẢNG CÁCH SỬ
DỤNG CẨM BIÊN SIÊU ÂM**

Lớp L01 --- HK 232

Giảng viên hướng dẫn: ThS.Nguyễn Đức Hoàng

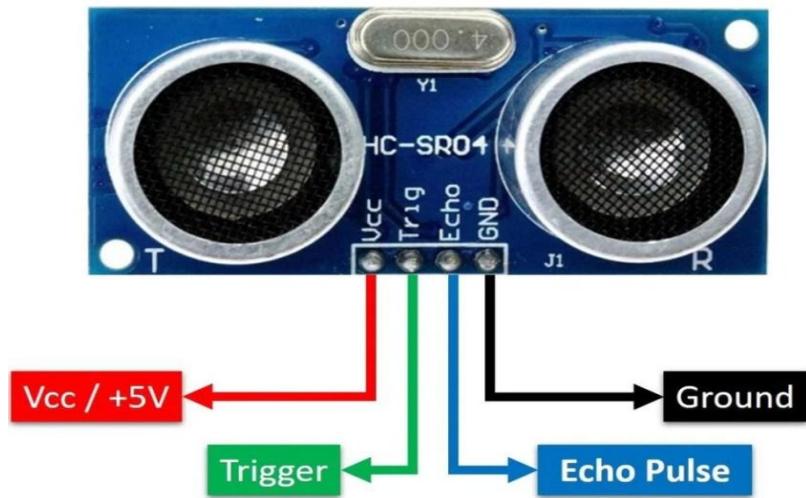
STT	Sinh viên thực hiện	MSSV
1	Lương Hồ Khanh Duy	2113013
2	Trần Kim Khanh	2113717

MỤC LỤC

I CƠ SỞ LÝ THUYẾT	1
1.1. Cảm biến siêu âm HC – SR04	1
1.1.1. Thông số kỹ thuật	1
1.1.2. Sơ đồ nguyên lý và cấu tạo	1
1.1.3. Nguyên lý hoạt động	3
1.1.4 Hiệu chỉnh (calib) giá trị đo :	5
1.2 Vị điều khiển STM32F103C8T6	5
II. Ý tưởng thực hiện	8
2.1. Hiển thị giá trị chính xác	8
2.2. Calib một khoảng đo bất kỳ	8
III. Thiết kế mạch	9
3.2. Phần mềm	10
3.2.1. Cấu hình trên phần mềm STM32CubeIDE	10
3.2.1. Viết chương trình trên Stm32cubeIDE	12
IV. Thực nghiệm thông qua giao diện hiển thị trên máy tính	14
4.1. Chế độ hiển thị giá trị chính xác	14
4.2. Chế độ hiệu chỉnh (Calib)	15

I. CƠ SỞ LÝ THUYẾT

1.1. Cảm biến siêu âm HC-SR04



1.1.1. Thông số kỹ thuật

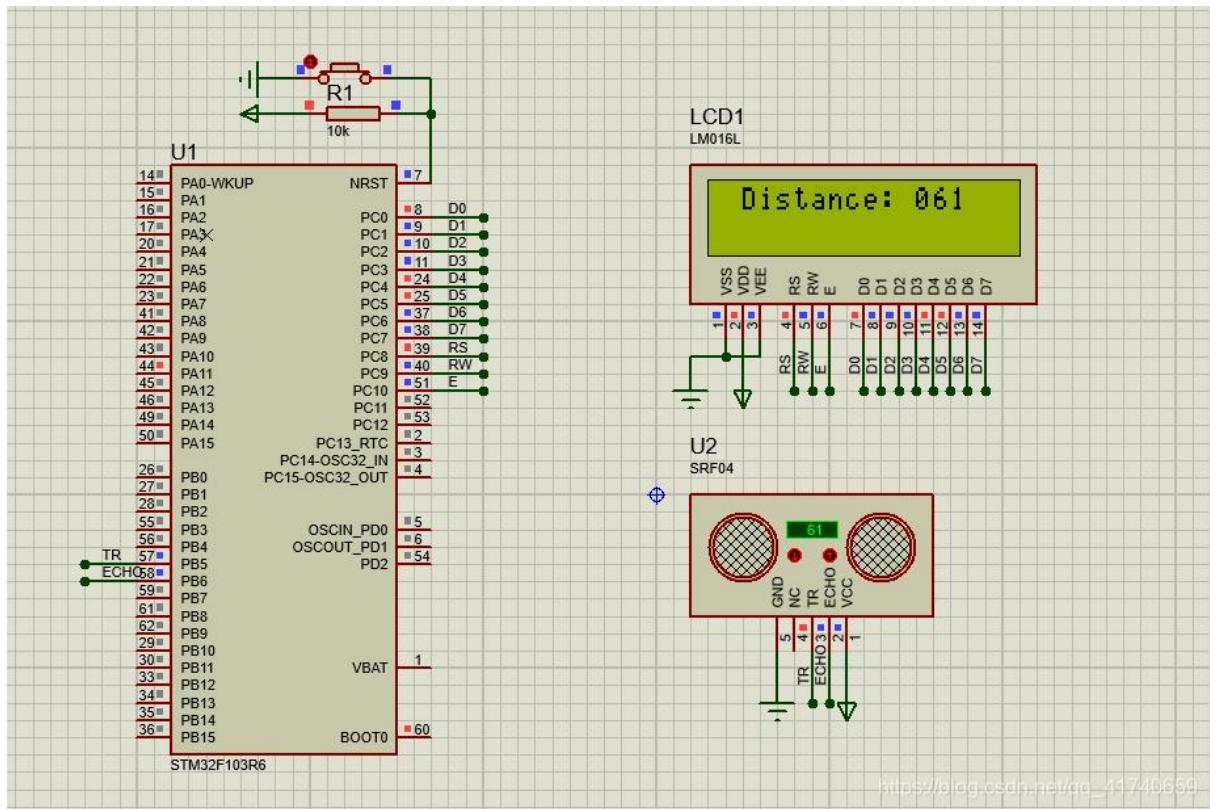
- Điện áp hoạt động: 5VDC
- Dòng tiêu thụ: 10~40mA
- Tín hiệu giao tiếp: TTL
- Chân tín hiệu: Echo, Trigger.
- Góc quét: < 15 độ
- Tần số phát sóng: 40Khz
- Khoảng cách đo được: 2~450cm (khoảng cách xa nhất đạt được ở điều khiển lý tưởng với không gian trống và bề mặt vật thể bằng phẳng, trong điều kiện bình thường cảm biến cho kết quả chính xác nhất ở khoảng cách < 100cm)
- Sai số: 0.3cm (khoảng cách càng gần, bề mặt vật thể càng phẳng sai số càng nhỏ).
- Kích thước: 43mm x 20mm x 17mm

1.1.2. Sơ đồ nguyên lý và cấu tạo

Cấu tạo gồm : máy phát sóng siêu âm (T) , máy thu (R) và mạch điều khiển gồm 4 chân:

- Vcc : Chân cấp nguồn cho cảm biến (nguồn 5V, nguồn càng chuẩn cảm biến càng chính xác)

- GND : Chân nối đất của cảm biến
- Trigger : Chân Trigger là chân đầu vào. Chân này phải được giữ ở mức cao trong khoảng 10us để khởi tạo phép đo bằng cách gửi sóng siêu âm.
- Echo Pulse : Chân Echo là một pin đầu ra. Chân này sẽ trả về tín hiệu xung khi sóng siêu âm phản xạ lại. Đo độ rộng xung này ta sẽ tính ra khoảng cách từ cảm biến đến vật cản.



Cấu tạo chi tiết của nó gồm 3 phần:

- Phần phát tín hiệu

Các đầu phát và đầu thu siêu âm là các loa gồm được chế tạo đặc biệt, hoạt động phát siêu âm có cường độ cao nhất ở một tần số nào đó (thường là 40kHz cho các ứng dụng đo khoảng cách). Các loa này cần có nguồn tín hiệu điều khiển có điện áp cao mới phát tốt được (theo datasheet thì là ~30V). Chính vì vậy trong phần phát, phần đệm công suất sử dụng một con MAX232 làm nhiệm vụ đệm. Nó sẽ lấy tín hiệu từ bộ điều khiển, khuỷch đại biên độ lên +/-30V cung cấp cho loa gồm.

Để tiết kiệm nguồn cho module cảm biến, phần cấp điện cho MAX232 được điều khiển thông qua một transistor PNP, khi không hoạt động, bộ điều khiển sẽ làm cho transistor này ngưng dẫn, hạn chế tiêu thụ dòng.

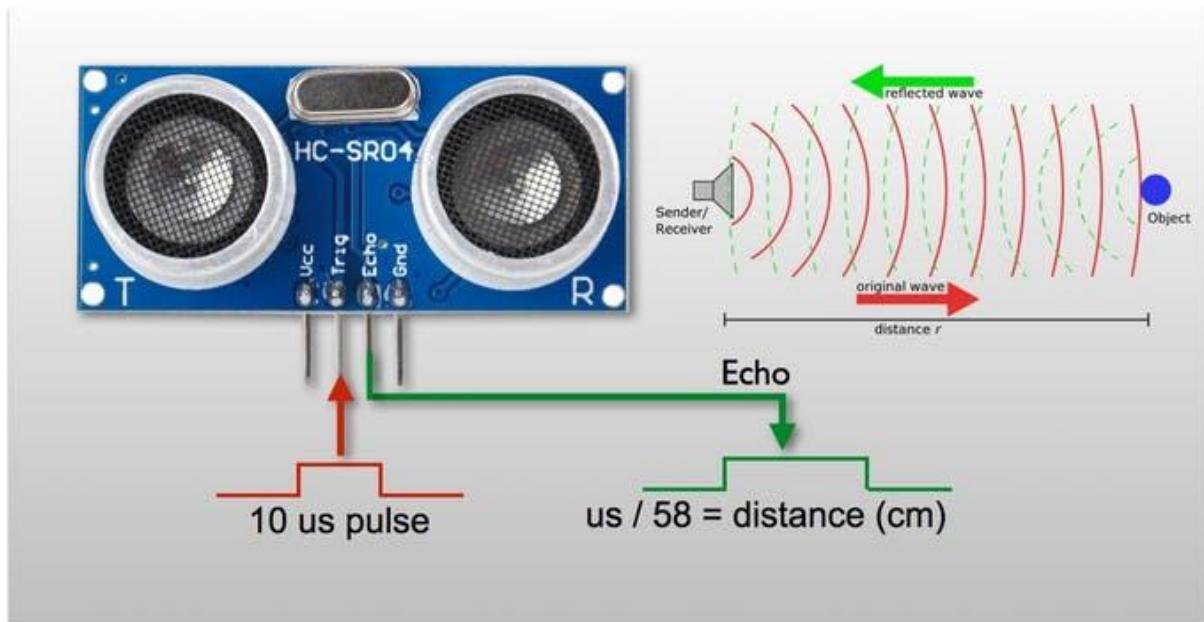
- Phần thu tín hiệu

Khi loa gồm làm đầu thu (loa này được chế tạo chỉ nhạy với một tần số nào đó - 40KHz) thu được sóng siêu âm, nó sẽ phát ra một điện thế giữa hai cực. Điện thế này là rất nhỏ, vì vậy nó được đưa qua một OPAM, ở đây là TL072 (Một số module sử dụng LM324,...). Tín hiệu này liên tục được khuỷch đại biên độ và cuối cùng là đưa qua một bộ so sánh, kết hợp với tín hiệu từ bộ điều khiển để đưa về bộ điều khiển thông qua một trans NPN

- Phần xử lý, điều khiển

Phần xử lý, điều khiển thường sử dụng một vi điều khiển (PIC16F688, STC11,...) làm nhiệm vụ phát xung, xử lý tính toán thời gian từ khi phát đến khi thu được sóng siêu âm do nó phát ra nếu nhận được tín hiệu TRIG.

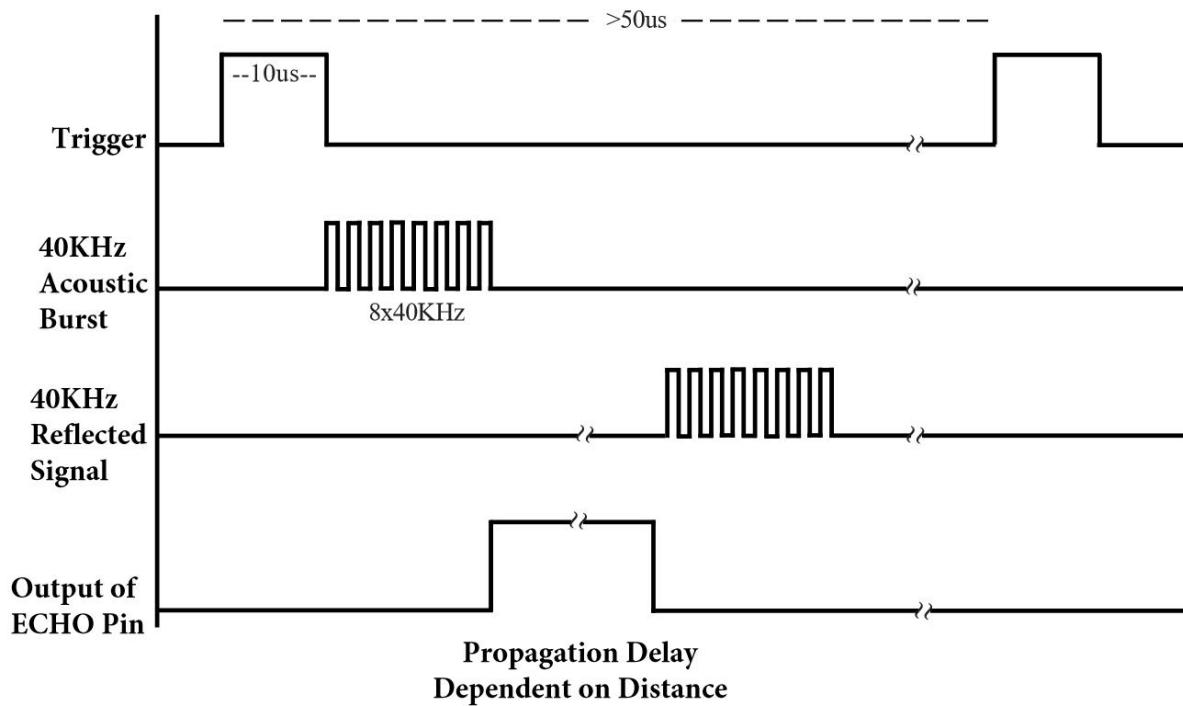
1.1.3. Nguyên lý hoạt động



Để cảm biến hoạt động, cần tạo 1 xung có độ rộng 10 μ s gửi đến ngoại vi qua chân TRIG. Khi chân Trigger được kích xung, máy phát sóng siêu âm phát ra 1 sóng có tần số 40KHz, khi chạm vật cản, sóng này phản xạ lại về máy thu. Khoảng thời gian từ khi

phát sóng đến khi thu sóng tương đương với thời gian chân Echo ở mức 1, dựa vào thời gian đó và vận tốc âm thanh (340m/s) ta có thể tính được khoảng cách từ cảm biến đến vật cản. Dùng “equation 1” ta sẽ tính được khoảng cách bằng công thức vật lý $S = v*t$

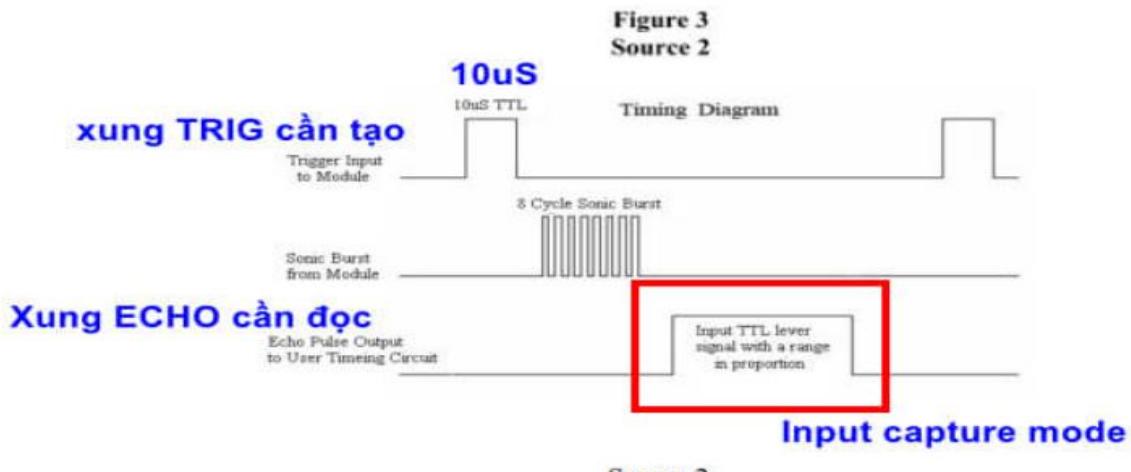
HC-SR04 ULTRASONIC MODULE



Taking Distance Measurements

The HC-SR04 can be triggered to send out an ultrasonic burst by setting the TRIG pin to HIGH. Once the burst is sent the ECHO pin will automatically go HIGH. This pin will remain HIGH until the the burst hits the sensor again. You can calculate the distance to the object by keeping track of how long the ECHO pin stays HIGH. The time ECHO stays HIGH is the time the burst spent traveling. Using this measurement in equation 1 along with the speed of sound will yield the distance travelled. A summary of this is listed below, along with a visual representation in Figure 2.

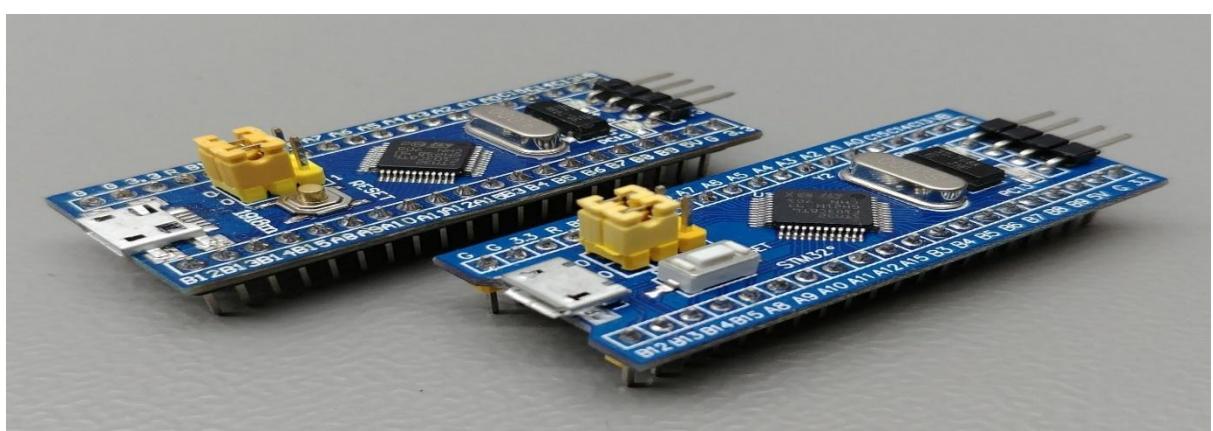
1. Set TRIG to HIGH
2. Set a timer when ECHO goes to HIGH
3. Keep the timer running until ECHO goes to LOW
4. Save that time
5. Use equation 1 to determine the distance travelled



1.1.4 Hiệu chỉnh (calib) giá trị đo :

Là hiệu chỉnh giá trị cảm biến đo được về một khoảng đo mong muốn. Dựa vào tính chất tuyến tính của độ nhạy trong cảm biến HC – SR04, ta có thể hiệu chỉnh một khoảng đo đọc từ cảm biến bằng cách gán giá trị zero và span cho điểm đầu và điểm cuối của tầm đo và cho phép bộ điều khiển tính toán dựa trên phương trình tuyến tính đó.

1.2 Vị điều khiển STM32F103C8T6



Giới thiệu tổng quan

- STM32F103C8T6 là vi điều khiển 32bit, thuộc họ F1 của dòng chip STM32 hãng ST. Phần mềm lập trình : có khá nhiều trình biên dịch cho STM32 như : IAR Embedded Workbench, Keil C...
- Lõi ARM Cortex M3.
- Tốc độ tối đa 72Mhz.
- Bộ nhớ : 64 kbytes bộ nhớ Flash 20 kbytes SRAM
- Clock, reset và quản lý nguồn

Điện áp hoạt động từ 2.0 → 3.6V.

Sử dụng thạch anh ngoài từ 4Mhz → 20Mhz.

Thạch anh nội dùng dao động RC ở mode 8Mhz hoặc 40Khz.

- Chế độ điện áp thấp:

Có các mode: ngủ, ngừng hoạt động hoặc hoạt động ở chế độ chờ.

Cáp nguồn ở chân Vbat bằng pin ngoài để dùng bộ RTC và sử dụng dữ liệu được lưu trữ khi mất nguồn cấp chính.

- 2 bộ ADC 12 bit với 9 kênh cho mỗi bộ

Khoảng giá trị chuyển đổi từ 0 – 3.6 V

Có chế độ lấy mẫu 1 kênh hoặc nhiều kênh.

- DMA:

7 kênh DMA

Có hỗ trợ DMA cho ADC, UART, I2C, SPI.

- 7 bộ Timer:

Timer 16 bit hỗ trợ các mode Input Capture/ Output Compare/ PWM.

1 Timer 16 bit hỗ trợ để điều khiển động cơ với các mode bảo vệ ngắt Input, dead-time.

Watchdog Timer để bảo vệ và kiểm tra lỗi. 1 Systick Timer 24 bit để xuông cho hàm Delay,....

- Có hỗ trợ 9 kênh giao tiếp:

bộ I2C.

bộ USART

SPI

1 CAN

USB 2.0 full-speed interface

- Kiểm tra lỗi CRC và 96-bit ID.

1.2.2. Thông số kỹ thuật

- Vi điều khiển: STM32F103C8T6.
- Điện áp cấp 5VDC qua cổng Micro USB sẽ được chuyển đổi thành 3.3VDC qua IC nguồn và cấp cho Vi điều khiển chính.
- Tích hợp sẵn thạch anh 8Mhz.
- Tích hợp sẵn thạch anh 32Khz cho các ứng dụng RTC.
- Ra chân đầy đủ tất cả các GPIO và giao tiếp: CAN, I2C, SPI, UART, USB,...
- Tích hợp Led trạng thái nguồn, Led PC13, Nút Reset.
- Kích thước: 53.34 x 15.24mm.

II. Ý tưởng thực hiện

2.1. Hiển thị giá trị chính xác

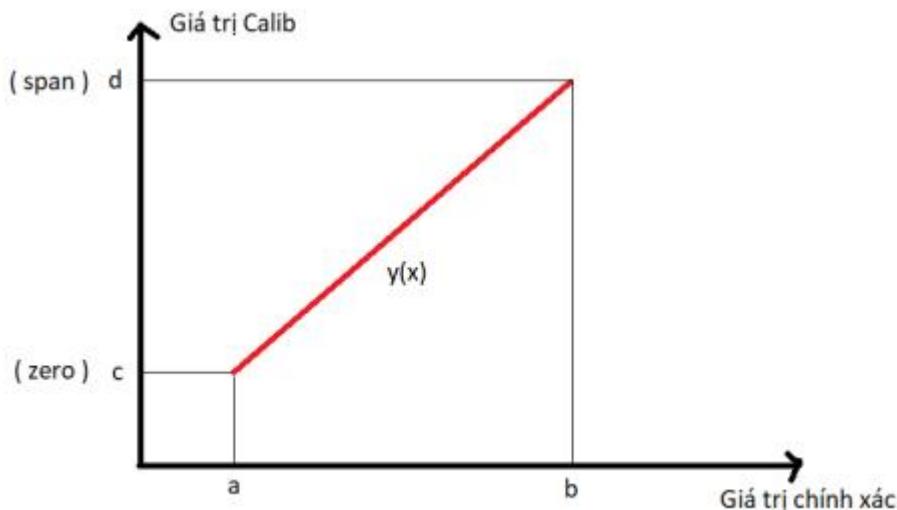
Ta tính toán độ chênh lệch về thời gian giữa 2 lần thay đổi mức của chân Echo và đặt là biến Difference. Dựa vào công thức : $Khoảng cách = Thời gian * Vận tốc$

Ta tính được giá trị khoảng cách Distance là :

$$distance = \frac{difference * 340 * 100 * 10^{-6}}{2} = difference * 0.017 \text{ (cm / s)}$$

Trong đó : difference đọc từ cảm biến là đơn vị us nên phải đổi về đơn vị s và phải chia 2 giá trị tính được vì thời gian đo được tính cho cả sóng truyền đi và truyền về.

2.2. Calib một khoảng đo bất kỳ.



Để calib giá trị nhiệt độ ta muốn đo ($a \rightarrow b$) về một khoảng giá trị nhiệt độ đo mong muốn ($c \rightarrow d$), ta thực hiện như sau:

Vì độ nhạy của cảm biến là tuyến tính nên ta thiết lập một phương trình bậc nhất $y(x)$ với điểm đầu là (a, c) và điểm cuối là (b, d):

Đặt $y(x) = K_1x + K_2$ ta có 2 điều kiện là $y(a) = c$ và $y(b) = d$, thay vào ta được

$$\begin{cases} c = K_1a + K_2 & (1) \\ d = K_1b + K_2 & (2) \end{cases}$$

Tính K₁:

Lấy (1) - (2) ta được:

$$c - d = K_1(a - b) \Rightarrow K_1 = \frac{c - d}{a - b}$$

Tính K₂:

Lấy (1)*b-(2)*a ta được:

$$cb - da = K_2(b - a) \Rightarrow K_2 = \frac{ad - bc}{a - b}$$

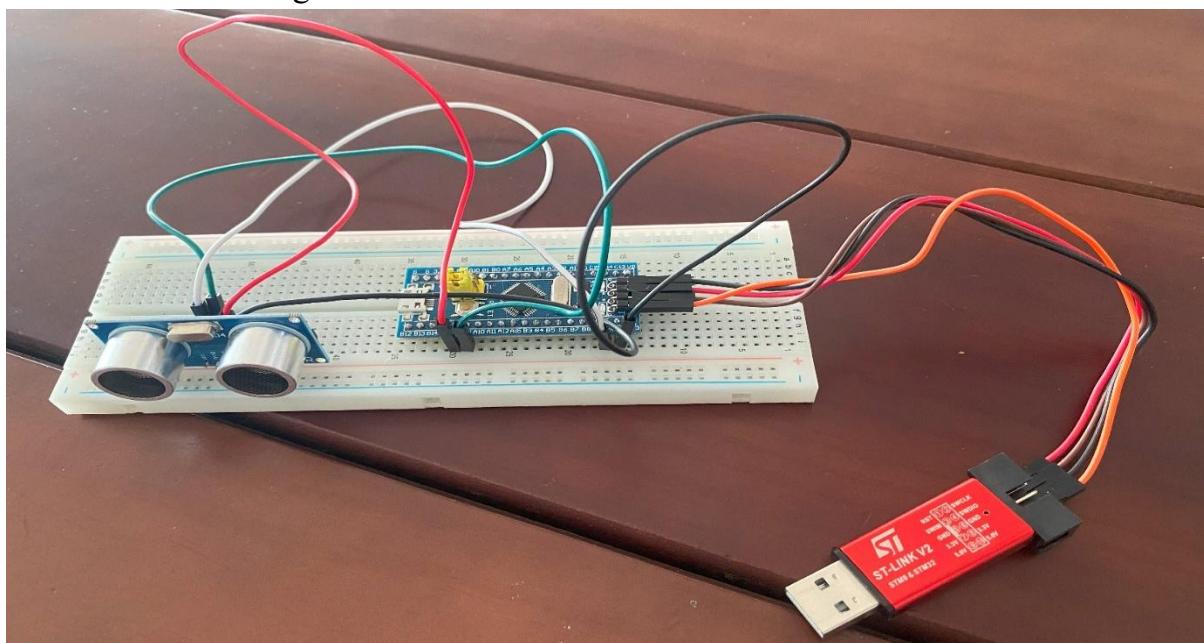
Vậy phương trình cần tìm là:

$$y = \frac{(c - d) * x + a * d - b * c}{a - b}$$

- + Dựa vào phương trình trên ta dùng giá trị đo chính xác là khoảng cách để tính toán ra giá trị hiển thị tương ứng sau khi calib.

III. Thiết kế mạch

3.1. Phân cứng



3.2. Phần mềm

3.2.1. Cấu hình trên phần mềm STM32CubeIDE



Cấu hình GPIO cho chân TRIG và chân ECHO của HC-SR04:

- Chân TRIG được cấu hình là chân đầu ra (output), để điều khiển tín hiệu siêu âm được phát ra.
- Chân ECHO được cấu hình là chân đầu vào (input), để đọc tín hiệu siêu âm phản xạ lại.

Trong đề tài này, Chân TRIG của HC-SR04 được kết nối với chân PA9 của STM32F103C8T6, chân ECHO được kết nối với chân PA8.

TIM1 Mode and Configuration

Mode

Slave Mode	Disable
Trigger Source	Disable
Clock Source	Internal Clock
Channel1	Input Capture direct mode
Channel2	Disable
Channel3	Disable

Configuration

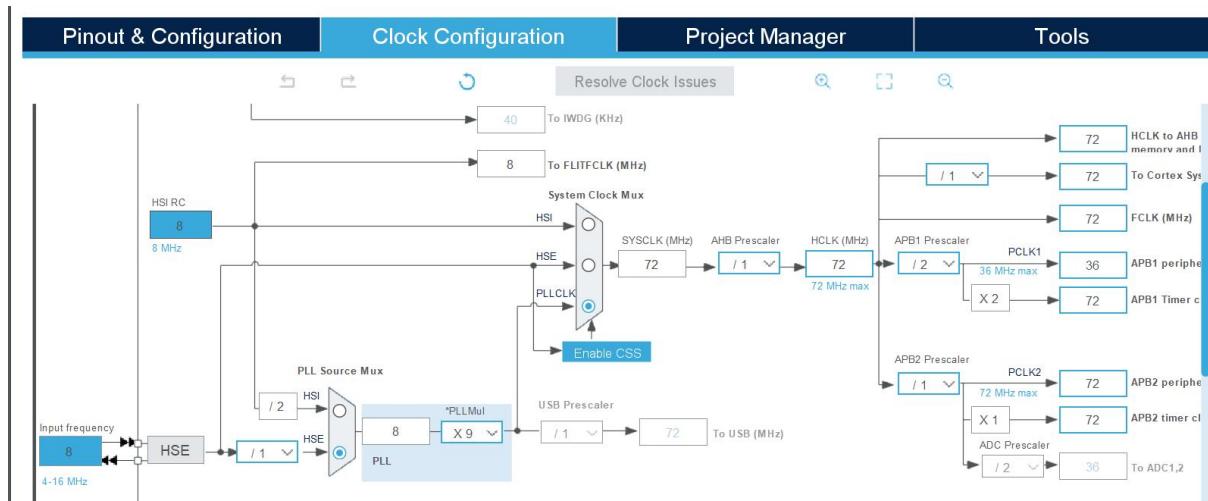
Reset Configuration

DMA Settings		GPIO Settings	
Parameter Settings		User Constants	NVIC Settings
TIM1 update interrupt	<input type="checkbox"/>	0	0
TIM1 trigger and commutation inte...	<input type="checkbox"/>	0	0
TIM1 capture compare interrupt	<input checked="" type="checkbox"/>	0	0

Cấu hình Clock và NVIC Setting

DMA Settings		GPIO Settings	
Parameter Settings		User Constants	NVIC Settings
Configure the below parameters :			
<input type="checkbox"/> Search (Ctrl+F)			
Counter Settings <ul style="list-style-type: none"> Prescaler (PSC - 16 b... 71) Counter Mode Up Counter Period (Auto... 0xffff-1) 			

Cấu hình Parameter Settings



Cấu hình Clock

Sử dụng timer để đo thời gian phản hồi của tín hiệu siêu âm.

- + Để đo khoảng cách, ta cần tính thời gian mà tín hiệu siêu âm phát ra và phản hồi lại. Vì điều khiển STM32F103C8T6 có sẵn các timer để thực hiện nhiệm vụ này. Ở đây ta sử dụng TIM1.
- + Tiến hành cấu hình và thiết lập ngắt cho TIM1.

3.2.1. Viết chương trình trên Stm32cubeIDE

```

4 * Created on: Mar 23, 2024
5 * Author: Ms KimKhanh
6 */
7
8 #include "stm32f1xx_hal.h"
9 #include "HC_SR04.h"
10
11
12 extern TIM_HandleTypeDef htim1;
13
14 uint32_t IC_Val1 = 0;
15 uint32_t IC_Val2 = 0;
16 uint32_t Difference = 0;
17 uint8_t Is_First_Captured = 0; // is the first value captured ?
18 uint8_t Distance = 0;
19
20 float mode = 0;
21 float c = 0;
22 float d = 0;
23 float a = 0;
24 float b = 0;
25 float y = 0;
26 float x = 0;
27 float zero;
28 float span;
29 float initial = 0;
30 float display = 0;
31
32 #define TRIG_PIN GPIO_PIN_9
33 #define TRIG_PORT GPIOA
34 #define ECHO_PIN GPIO_PIN_8
35 #define ECHO_PORT GPIOA

```

Khai báo biến sử dụng trong chương trình

```
38 void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
39 {
40     if (htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1) // if the interrupt source is channel
41     {
42         if (Is_First_Captured==0) // if the first value is not captured
43         {
44             IC_Val1 = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_1); // read the first value
45             Is_First_Captured = 1; // set the first captured as true
46             // Now change the polarity to falling edge
47             __HAL_TIM_SET_CAPTUREPOLARITY(htim, TIM_CHANNEL_1, TIM_INPUTCHANNELPOLARITY_FALLING);
48         }
49     else if (Is_First_Captured==1) // if the first is already captured
50     {
51         IC_Val2 = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_1); // read second value
52         __HAL_TIM_SET_COUNTER(htim, 0); // reset the counter
53
54         if (IC_Val2 > IC_Val1)
55         {
56             Difference = IC_Val2-IC_Val1;
57         }
58     else if (IC_Val1 > IC_Val2)
59     {
60         Difference = (0xffff - IC_Val1) + IC_Val2;
61     }
62     Distance = Difference * .034/2;
63     if (initial ==0)
64     {
65         display = Distance;
66     }
67     if (mode ==1)
68     {
69         initial++;
70         if (zero == 1 && span == 0)
71         {
72             a = Distance;
73         }
74         if (zero == 0 && span == 1 )
75         {
76             b = Distance;
77         }
78     }
79     else if (mode == 0 && initial != 0 && zero == 0 && span == 0)
80     {
81         x = Distance;
82         y = (x*(c-d)+a*d-b*c)/(a-b);
83         display = y;
84     }
85     Is_First_Captured = 0; // set it back to false
86     // set polarity to rising edge
87     __HAL_TIM_SET_CAPTUREPOLARITY(htim, TIM_CHANNEL_1, TIM_INPUTCHANNELPOLARITY_RISING);
88     __HAL_TIM_DISABLE_IT(&htim1, TIM_IT_CC1);
89 }
90 }
91 }
```

Hàm xử lý ngắt (Interrupt)

```
93 void delay(uint16_t time){
94     __HAL_TIM_SET_COUNTER(&htim1, 0);
95     while(__HAL_TIM_GET_COUNTER(&htim1) < time);
96 }
97
98 uint8_t HCSR04_GetDis (void)
99 {
100     HAL_GPIO_WritePin(TRIG_PORT, TRIG_PIN, GPIO_PIN_SET); // pull the TRIG pin HIGH
101     delay(10); // wait for 10 us
102     HAL_GPIO_WritePin(TRIG_PORT, TRIG_PIN, GPIO_PIN_RESET); // pull the TRIG pin low
103
104     __HAL_TIM_ENABLE_IT(&htim1, TIM_IT_CC1);
105
106 }
```

Tạo xung 10us bằng hàm delay và Chương trình đọc cảm biến siêu âm HC-SR04

Hàm `delay(μs)` hoạt động bằng cách là đếm từ 0 → “time” mong muốn với tần số 1MHz, khi hoàn thành sẽ reset lại bộ đếm.

```
19  /* Includes */
20  #include "main.h"
21  #include "HC_SR04.h"
22
23  TIM_HandleTypeDef htim1;
24
25  void SystemClock_Config(void);
26  static void MX_GPIO_Init(void);
27  static void MX_TIM1_Init(void);
28
29  uint8_t distance = 0;
30
31 int main(void)
32 {
33
34     HAL_Init();
35
36
37     SystemClock_Config();
38
39
40     MX_GPIO_Init();
41     MX_TIM1_Init();
42     HAL_TIM_IC_Start_IT(&htim1, TIM_CHANNEL_1);
43
44     while (1)
45     {
46         distance = HCSR04_GetDis();
47         HAL_Delay(100);
48
49     }
50
51 }
52 }
```

Hàm `while(1)` là gọi và nhận giá trị khoảng cách.

IV. Thực nghiệm thông qua giao diện hiển thị trên máy tính

4.1. Chế độ hiển thị giá trị chính xác

Bước 1: Chính biến mode = 0

Khi đó giá trị hiển thị trên biến mode sẽ là giá trị thực tế (display = Distance)

Expression	Type	Value	Address
display	float	5	0x20000060
+ Add new expression			
Expression	Type	Value	Address
Distance	uint8_t	5 '\005'	0x20000035
a	float	0	0x20000044
b	float	0	0x20000048
c	float	0	0x2000003c
d	float	0	0x20000040
x	float	0	0x20000050
y	float	0	0x2000004c
mode	float	0	0x20000038
initial	float	0	0x2000005c
display	float	5	0x20000060
zero	float	0	0x20000054
span	float	0	0x20000058
+ Add new expression			

4.2. Chế độ hiệu chỉnh (Calib)

Bước 1: Chính biến mode = 1

Bước 2: Đặt giá trị zero bằng cách dịch chuyển vật đến 5cm và thay đổi biến zero = 1, khi đó biến a = 5cm, nhập từ giao diện người dùng giá trị zero = 10cm ở biến c. Sau khi nhập c, thay đổi biến zero = 0.

Bước 3: Đặt giá trị span bằng cách dịch chuyển vật đến 10cm và thay đổi biến span = 1, khi đó biến b = 10cm, nhập từ giao diện người dùng giá trị span = 20cm ở biến d. Sau khi nhập d, thay đổi biến span = 0.

Bước 4: Chính biến mode = 0 để đo ở chế độ hiệu chỉnh (Calib).

Ví dụ 1: Giá trị zero tại 5cm là 10, và giá trị span tại 10cm là 20. Và đây là kết quả hiển thị ở vị trí thực tế 7cm sau khi đã hiệu chỉnh.

Expression	Type	Value
mode	float	0
display	float	14
a	float	5
b	float	10
c	float	10
d	float	20
x	float	7
y	float	14
initial	float	3530
zero	float	0
span	float	0
distance	uint8_t	7 '\a'
Distance	uint8_t	7 '\a'
+ Add new expression		

Ví dụ 2: Giá trị zero tại 5cm là 15, và giá trị span tại 10cm là 30. Và đây là kết quả hiển thị ở vị trí thực tế 6cm sau khi đã hiệu chỉnh.

Expression	Type	Value	Address
mode	float	0	0x200
display	float	18	0x200
a	float	5	0x200
b	float	10	0x200
c	float	15	0x200
d	float	30	0x200
x	float	6	0x200
y	float	18	0x200
initial	float	1040	0x200
zero	float	0	0x200
span	float	0	0x200
Distance	uint8_t	6 '\006'	0x200
+ Add new expression			