[CHƯƠNG 4. LẬP TRÌNH HỆ THỐNG](#_Toc516675281)

[4.1. Sơ đồ giải thuật](#_Toc516675282)

[4.2. Chương trình đọc giá trị quang trở](#_Toc516675283)

Để MCU có thể điều khiển động cơ hướng theo ánh sáng mặt trời thì cần có giá trị quang trở để xác định được cường độ ánh sáng đang chiếu vào tấm pin.

[4.2.1. Sơ đồ giải thuật](#_Toc516675284)

[4.2.2. Lập trình](#_Toc516675285)

Trong sơ đồ giải thuật, để đọc giá trị quang trở thì phải đọc giá trị điện áp đầu quang trở thông qua cổng ADC2.

Giá trị ADC đọc về là 12bit tương đương với scale: 0-VDD(VDD=3V) về 0-4096. Chương trình sẽ không quy đổi giá trị raw sang điện áp mà sẽ sử dụng trực tiếp để điều khiển động cơ sao cho 4 giá trị ADC đọc về là cực đại.

Sau khi khởi tạo các kênh ADC, giá trị ADC được đọc về và đẩy vào module motor task liên tục trong vòng lặp while.

[4.3. Chương trình điều khiển động cơ bằng xung PWM](#_Toc516675283)

[4.3.1. Sơ đồ giải thuật](#_Toc516675284)

[4.3.2. Lập trình](#_Toc516675285)

Chương trình điều khiển động cơ bằng cách băm xung PWM vào module L298 để quay động cơ.

Điện áp cấp vào L298 là 5V, trong giải thuật chọn 55% là phần trăm xung PWM cho motor 1 để quay tấm pin trên, 65% xung cho motor 2 để quay toàn bộ tấm pin và đế trên.

Để quay 1 motor thì chương trình phải set 1 chân khiển của nó về GND và chân còn lại băm xung PWM, nếu muốn quay ngược lại thì cấp ngược chân motor.

Chương trình đọc giá trị quang trở về liên tục để so sánh và điều khiển ra xung PWM sao cho đến khi cả 4 giá trị quang trở là tương đương nhau.

[4.4. Chương trình giao tiếp MPU6050 qua I2C](#_Toc516675286)

[4.4.1. Sơ đồ giải thuật](#_Toc516675287)

[4.4.2. Lập trình](#_Toc516675288)

Trong đề tài này, chúng ta sử dụng MPU6050 để đo nhiệt độ xe và xác định hướng, vận tốc cho xe để gửi về server.

MCU trước tiên init I2C với MCU là master và MPU6050 là slave, xung clock 400000KHz.

Trước khi hoạt động, chương trình kiểm tra module MPU6050 đã được cắm vào bằng cách ghi địa chỉ của MPU6050 là 0xD0 và chờ kết quả trả về xem đã có thiết bị chưa. Sau đó, config MPU6050 gyro accel và đọc ở tần số lấy mẫu 100Hz, độ nhạy 2g-250s.

Trong vòng lặp while chính, MCU đọc ở thanh ghi ACCEL\_XOUT (0x3B) 6 giá trị 14 giá trị liên tiếp như bảng sau:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0x3B | 0x3C | 0x3D | 0x3E | 0x3F | 0x40 | 0x41 | 0x42 | 0x43 | 0x44 | 0x45 | 0x46 | 0x47 | 0x48 |
| accel x | | accel y | | accel z | | temperature | | gyro x | | gyro y | | gyro z | |

Các giá trị đọc được là dữ liệu raw 16bit. Vì vậy, sau khi đọc chúng ta phải scale lại theo đơn vị m/s2 và g bằng cách chia cho mức độ nhạy.

Độ nhạy gyro:

MPU6050\_GYRO\_SENS\_250       ((float) 131)

MPU6050\_GYRO\_SENS\_500       ((float) 65.5)

MPU6050\_GYRO\_SENS\_1000      ((float) 32.8)

MPU6050\_GYRO\_SENS\_2000      ((float) 16.4)

Độ nhạy accel:

MPU6050\_ACCE\_SENS\_2         ((float) 16384)

MPU6050\_ACCE\_SENS\_4         ((float) 8192)

MPU6050\_ACCE\_SENS\_8         ((float) 4096)

MPU6050\_ACCE\_SENS\_16        ((float) 2048)

[4.5. Chương trình giao tiếp với A9G qua USART](#_Toc516675289)

[4.5.1. Giới thiệu về AT command và lệnh AT dùng cho A9G](#_Toc516675290)

4.5.1.1 Giới thiệu

Hiện nay, có nhiều module mà khi xử lý bằng các thao tác lệnh thông thường như dịch thanh ghi địa chỉ thì sẽ rất phức tạp nên các module này sẽ được thiết kế phần cứng và firmware để xử lý các tác vụ bên trong từ các yêu cầu lệnh bên ngoài.

1 trong số các tập lệnh để request module chính là tập lệnh AT.

Tập lệnh AT(hoặc at) viết tắt là ATtention. Tập lệnh thường được sử dụng để communicate với các module SIM/GPS/GPRS. Tiền tố “AT” luôn là từ bắt đầu của mọi dòng lệnh trong tập lệnh.

Trong đề tài này, module A9G giao tiếp với MCU qua chuẩn truyền UART.

Module A9G sử dụng baudrate mặc định là 115200 nhưng nó có thể detect bauderate mới 1 cách tự động không cần setup.

Module A9G sẽ trả về chuỗi giống như chuỗi đã nhận + chuỗi trả lời từ chân RD. Lưu ý: tất cả các lệnh phải có dấu xuống dòng ở cuối câu.

Dưới đây là một số tác vụ có thể thực hiện bằng lệnh AT với modem GSM/GPRS hoặc điện thoại di động:

* Nhận thông tin cơ bản về điện thoại di động hoặc modem GSM / GPRS.
* Nhận thông tin cơ bản về người đăng ký.
* Nhận trạng thái hiện tại của điện thoại di động hoặc modem GSM / GPRS.
* Thiết lập kết nối dữ liệu hoặc kết nối thoại với modem từ xa.
* Gửi và nhận fax.
* Gửi, đọc, viết, hoặc xóa tin nhắn SMS và nhận thông báo về tin nhắn SMS mới nhận được.
* Đọc mục nhập, viết hoặc tìm kiếm danh bạ.
* Thực hiện các nhiệm vụ liên quan đến bảo mật, chẳng hạn như mở hoặc đóng khóa cơ sở, kiểm tra xem cơ sở có bị khóa và thay đổi mật khẩu hay không.
* Kiểm soát việc trình bày các mã kết quả / thông báo lỗi của các lệnh AT.
* Nhận hoặc thay đổi cấu hình của điện thoại di động hoặc modem GSM/GPRS.
* Lưu và khôi phục cấu hình của điện thoại di động hoặc modem GSM/GPRS.

Lưu ý là các nhà sản xuất điện thoại di động thường không sử dụng tập lệnh AT trong điện thoại di động của họ. Ngoài ra, tác vụ của các lệnh AT được triển khai có thể khác với lệnh được xác định trong tiêu chuẩn.

Các modem GSM / GPRS được thiết kế cho các ứng dụng không dây có hỗ trợ tốt hơn các lệnh AT so với các điện thoại di động thông thường.

4.5.1.2 Một số lệnh thông dụng

|  |  |  |
| --- | --- | --- |
| Lệnh | Trả lời | Ý nghĩa |
| “AT” | “OK” | Kiểm tra A9G đã hoạt động |
| “AT+GPS=1” | “OK” | Init GPS |
| “AT+GPSRD=1” | “OK”......GPS data | Đọc GPS liên tục |
| “AT+LOCATION=2” | GPS data “OK” | Đọc GPS 1 lần |
| “AT+CIPSTATUS=1” | “+CIPSTATUS:0,IP INITIAL” chưa kết nối.  “+CIPSTATUS:0,CONNECT OK” đã kết nối. | Kiểm tra trạng thái kết nối internet |
| “AT+RST=1” | “init...” | Reset A9G |
| “AT+CIPSTART=”TCP”,”123.20.86.36”,3000” | “OK” thành công  “CME:ERROR” lỗi | Request connect tới server IP 123.20.86.36 port 3000. |
| “AT+CIPSEND” | “>” | Bắt đầu gửi chuỗi về server |
| “AT+CIPCLOSE” | “OK” | Ngắt kết nối với server |

[4.5.2. Giải thuật](#_Toc516675291)

Khởi tạo DMA UART1 ở bauderate 115200. Sau đó gửi lệnh reset A9G và chờ đến khi đã reset xong. Nếu không có module A9G thì sau timeout sẽ báo về 1 cổng UART khác để thông báo init fail.

MCU gửi command qua DMA UART và nhận data về qua UART thường.

Các data nhận về sẽ đẩy vào 1 chuỗi buffer và trả về chuỗi khi nhận được kí tự xuống dòng ở cuối câu.

Sau khi module A9G đã ở trạng thái sẵn sàng, MCU sẽ yêu cầu A9G thông tin về GPS hiện tại và truyền các tham số ID(mã số tài xế), nhiệt độ xe(‘C), dung lượng pin acquy(%), sau đó request connect tới server. Sau khi connect thành công, MCU sẽ gửi các thông tin vừa lấy được tới server và chờ chuỗi “Send successfully.” từ phía server gửi về để tiếp tục gửi 1 gói tin mới.

Tuy nhiên, module A9G thực sự chưa ổn định nên sau khi kết nối nó có thể disconnect bất cứ lúc nào sau timeout nên kể cả khi kiểm tra trạng thái kết nối server trước khi gửi tin cũng có thể disconnect ngay sau khi kiểm tra, mà như vậy thì gói tiếp theo sẽ không tới server nên sau khi đã nhận chuỗi “Send successfully.” từ server MCU sẽ request disconnect với server và khi gửi gói mới sẽ request connect.

[4.6. Chương trình giao tiếp pcf8574 ghi LCD20 qua I2C](#_Toc516675292)

[4.6.1. Sơ đồ giải thuật](#_Toc516675293)

[4.6.2. Lập trình](#_Toc516675294)

Khi lập trình I2C để giao tiếp STM32F407 và module PCF8574, thì byte đầu tiên sẽ bao gồm 7bit địa chỉ và 1bit W vào vị trí có trọng số thấp nhất(ghi dữ liệu), lúc này Byte đầu tiên của giao tiếp I2C sẽ có giá trị: 01001110 (0x4E)

Sau khi init I2C với MCU là master và PCF8574 là slave xung clock 100000KHz, MCU sẽ kiểm tra module PCF8574 đã kết nối chưa bằng cách write I2C địa chỉ của PCF8574(0x4E). Nếu không tìm thấy thiết bị, MCU sẽ thông báo cho dev biết qua cổng UART2.

Sau khi init I2C, MCU sẽ clear screen bằng cách gửi command 0x02, sau đó reset con trỏ vể đầu dòng bằng command 0x80(về ngoài cùng hàng) và 0x0C(về dòng trên cùng cột).

Trong chương trình chính, lcd\_task sẽ lấy data battery level(sau 1h lấy 1 lần) và thời gian thực từ A9G và ghi ra LCD từ địa chỉ 0x4E.

[4.7. Chương trình gửi dữ liệu từ vi điều khiển](#_Toc516675295)

[4.7.1. Sơ đồ giải thuật](#_Toc516675296)

[4.7.2. Lập trình](#_Toc516675297)

[4.8. Chương trình đo dung lượng pin](#_Toc516675298)

[4.8.1. Sơ đồ giải thuật](#_Toc516675299)

[4.8.2. Lập trình](#_Toc516675300)

giá trị dung lượng pin được đọc bằng cách đọc giá trị điện áp đầu điện trở trong mạch phân áp thông qua cổng ADC2.

Giá trị ADC đọc về là 12bit tương đương với scale: 0-VDD(VDD=3V) về 0-4096. Nguồn acquy có giá trị khoảng 12V, trước khi chuyển đổi từ điện áp đọc được ở MCU thì phải có bước calib để hiệu chỉnh công thức tính toán ở đầu MCU sao cho đúng với điện áp trên acquy.

Khi đã đo được chính xác điện áp 2 đầu acquy thì dựa vào công thức tính dung lượng cho bình acquy trong bảng sau để đưa ra dung lượng pin.

Sau khi khởi tạo các kênh ADC, giá trị ADC được đọc về và tính toán theo công thức tìm được bởi calib và thực hiện tác vụ liên tục trong vòng lặp while.

[4.9. Chương trình kiểm tra lỗi](#_Toc516675301)

[4.9.1. Sơ đồ giải thuật](#_Toc516675302)

[4.9.2. Lập trình](#_Toc516675303)

[4.10. Watchdog timer](#_Toc516675304)

[4.10.1. Khái quát về Watchdog timer](#_Toc516675305)

Watchdog timer là bộ đếm thời gian hoạt động liên tục nhằm tự động thực hiện một nhiệm vụ nào đó. Sau một khoảng thời gian được định trước nếu bộ đếm không được dừng hoặc refresh nó sẽ kích hoạt tác vụ đã được định sẵn ví dụ như reset thiết bị, tắt màn hình, phát tín hiệu cảnh báo, ... Trong quá trình đếm nếu bộ đếm được refresh nó cứ thế tự động đếm tiếp nhưng bắt đầu lại với thời gian là 0.

Bộ đếm này có thể hoạt động độc lập nên khi chương trình lỗi, bị đứng ở 1 tác vụ nào đó thì MCU sẽ thực hiện tác vụ định sẵn thường là reset MCU.

Thông thường Watchdog timer được nhắc đến như một sự hỗ trợ mang tính phần cứng, được sử dụng bởi phần mềm qua các trình điều khiển.

Tính năng này thường được sử dụng cho nhiều thiết bị nhúng đòi hỏi phải hoạt động 24/7 ví dụ như hệ thống của đề tài này.

[4.10.2. Watchdog timer trong STM32F407VGT](#_Toc516675306)

Có 2 loại Watchdog timer trong vi điều khiển STM32F407VGT6

#### 4.10.2.1. Independent Watchdog (IWDG)

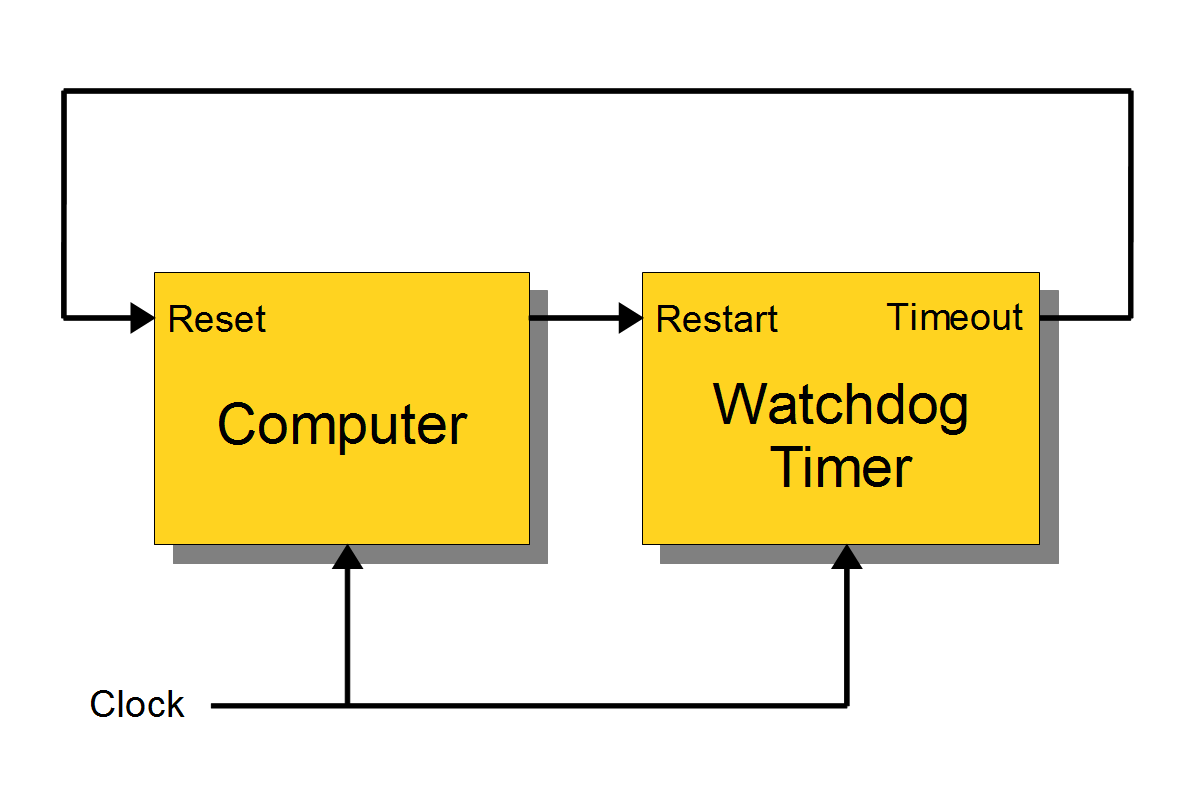
Watchdog độc lập dựa trên bộ đếm ngược 12 bit và bộ chia 8 bit. Nó được điều khiển từ một RC 32 kHz độc lập và khi nó hoạt động độc lập với đồng hồ chính, nó có thể hoạt động ở chế độ Stop và Standby. Nó có thể được sử dụng như một cơ quan giám sát để thiết lập lại thiết bị khi xảy ra sự cố hoặc làm bộ đếm thời gian chạy miễn phí để quản lý thời gian chờ của ứng dụng. Nó có thể cấu hình bằng phần cứng hoặc phần mềm thông qua các byte tùy chọn. Bộ đếm có thể bị đóng băng trong chế độ gỡ lỗi.

Đây là kiểu Watchdog được sử dụng trong mạch điều khiển của đề tài.

#### 4.10.2.2. System window Watchdog (WWDG)

Cơ chế này dựa trên bộ đếm ngược 7 bit có thể được đặt chạy tự do. Nó có thể được sử dụng như một cơ quan giám sát để thiết lập lại thiết bị khi xảy ra sự cố. Nó được định giờ từ đồng hồ APB1 (PCLK1) bắt nguồn từ đồng hồ chính. Nó có khả năng ngắt cảnh báo sớm và bộ đếm có thể bị đóng băng ở chế độ gỡ lỗi.

[4.10.3. Sơ đồ giải thuật](#_Toc516675307)



[4.10.4. Lập trình](#_Toc516675308)

Watchdog timer sử dụng bộ chia 128 cho tần số 32kHz và bộ đếm 12 bit tương ứng với thời gian là 32 giây. Ta phải canh thời gian để xóa bộ đếm cho Watchdog khỏi bị tràn.

Khi vi điều khiển bị treo, Timer không hoạt động, khi đó, Watchdog sẽ tự reset lại hệ thống.