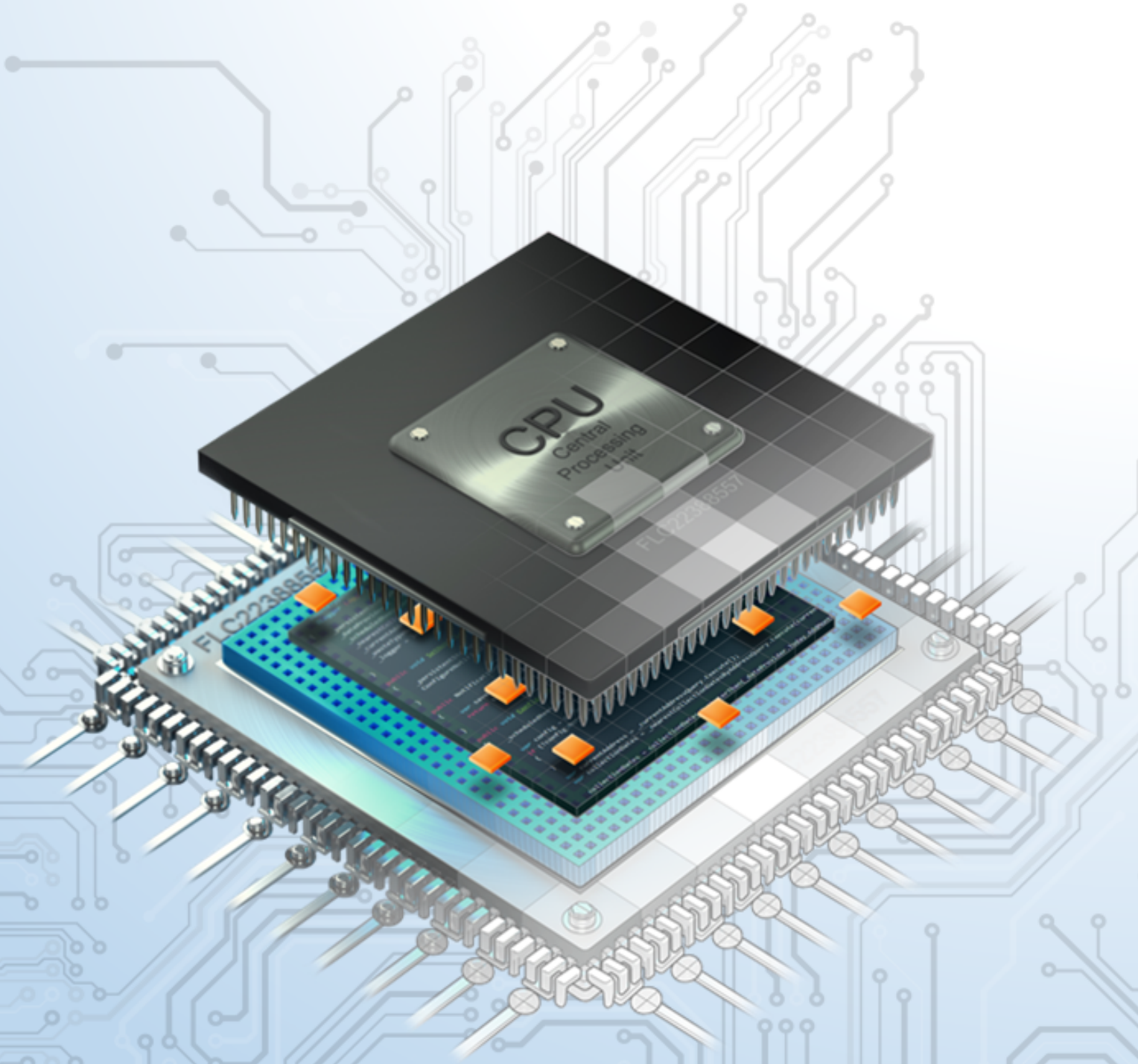
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
COMPUTER ENGINEERING

Microcontroller

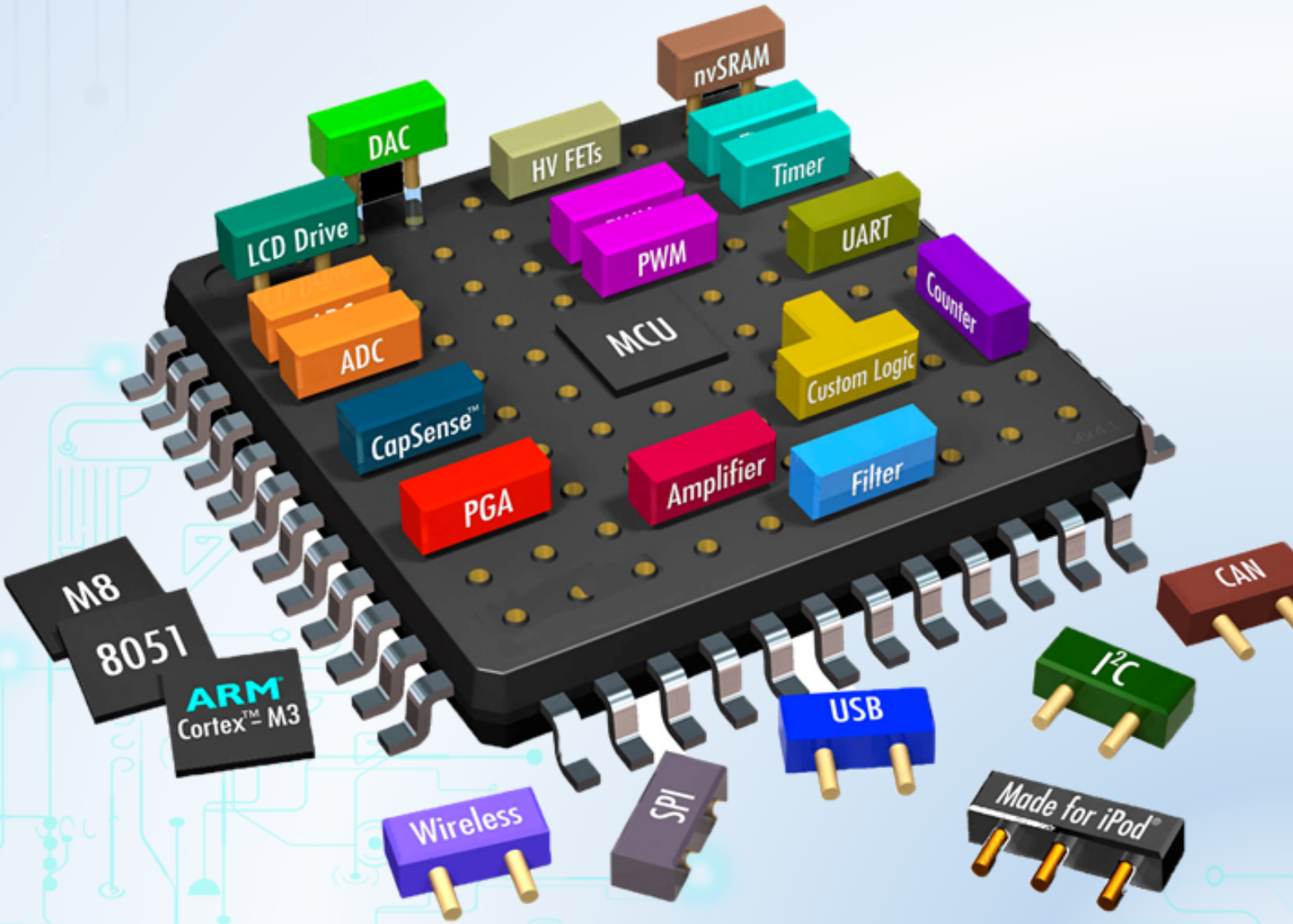


Dr. Le Trong Nhan

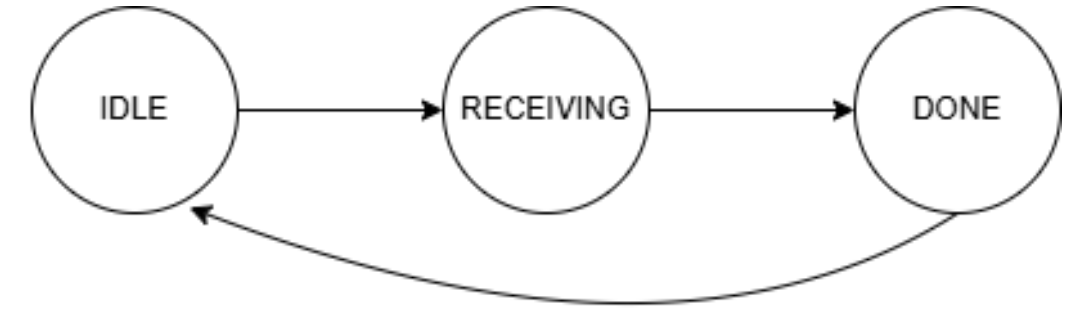
Mục lục

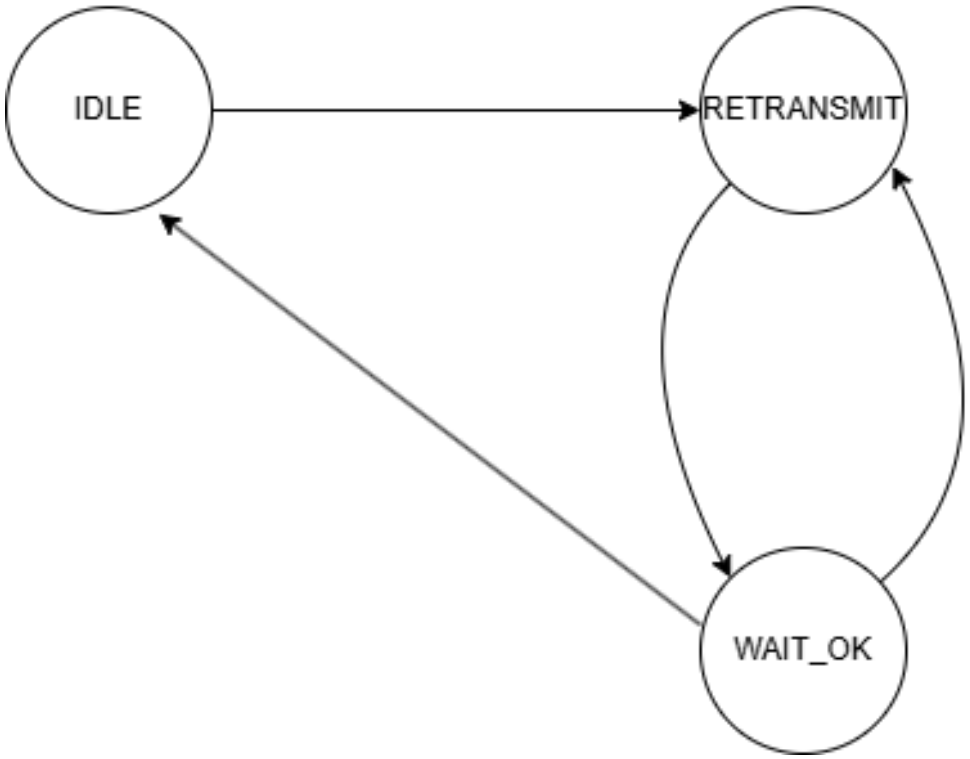
CHƯƠNG 1

Flow and Error Control in Communication



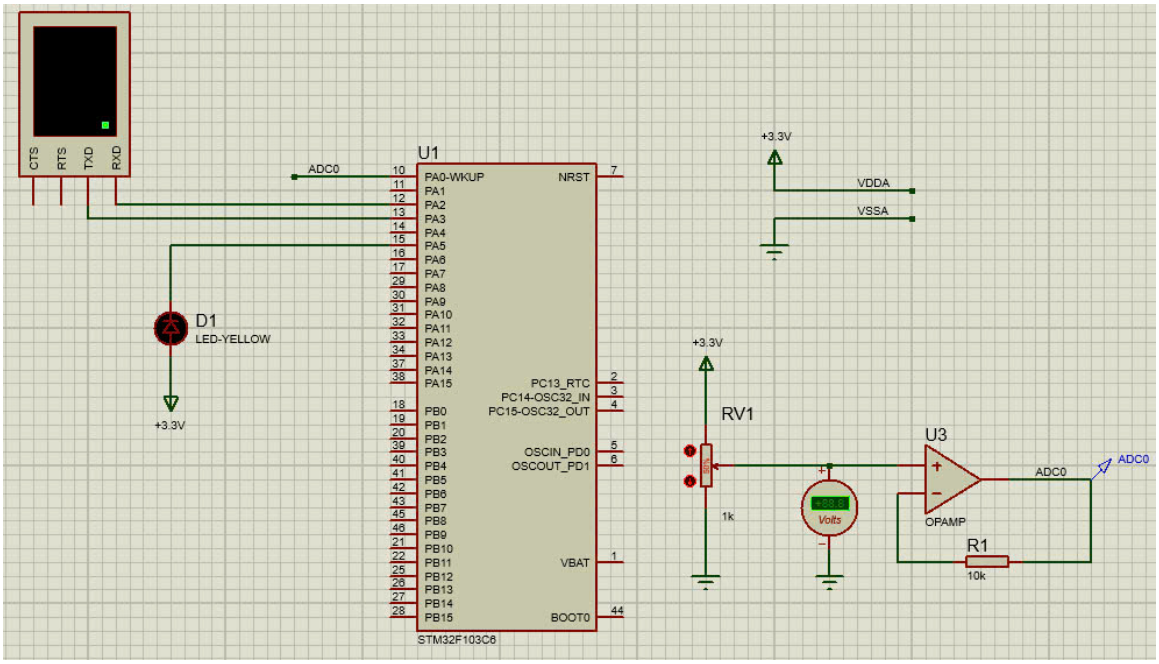
1 FSM





Hình 1.1: Các máy trạng thái khi giao tiếp UART

2 Proteus



Hình 1.2: Schematic in Proteus

3 Code

```c
#include "main.h"

/* Private includes
   ----------------------------------------------------------
   */
/* USER CODE BEGIN Includes */
#include "string.h"
#include <stdio.h>
#include "software_timer.h"
/* USER CODE END Includes */

/* Private typedef
   -----------------------------------------------------------
   */
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define
   ----------------------------------------------------------
   */
/* USER CODE BEGIN PD */
```

```
17 #define MAX_BUFFER_SIZE 200
18 #define RTS 1
19 #define OK 2
20 #define THREE_SECOND 3
21 /* USER CODE END PD */
22
23 /* Private macro
   -------------------------------------------------------------
   */
24 /* USER CODE BEGIN PM */
25
26 /* USER CODE END PM */
27
28 /* Private variables
   ---------------------------------------------------------
   */
29 ADC_HandleTypeDef hadc1;
30
31 TIM_HandleTypeDef htim2;
32
33 UART_HandleTypeDef huart2;
34
35 /* USER CODE BEGIN PV */
36
37 /* USER CODE END PV */
38
39 /* Private function prototypes
   -----------------------------------------------*/
40 void SystemClock_Config(void);
41 static void MX_GPIO_Init(void);
42 static void MX_ADC1_Init(void);
43 static void MX_USART2_UART_Init(void);
44 static void MX_TIM2_Init(void);
45 /* USER CODE BEGIN PFP */
46
47 /* USER CODE END PFP */
48
49 /* Private user code
   ---------------------------------------------------------
   */
50 /* USER CODE BEGIN 0 */
51 uint8_t temp = 0;  // B i ỉn        l u  d   l i u  n h n
    t  UART
52 uint8_t buffer[MAX_BUFFER_SIZE];  //  B     m         l u
    d   l i u  n h n  t   UART
53 uint8_t index_buffer = 0;  //  C h   s   h i n  t i  c a
    b     m
54 uint8_t buffer_flag = 0;  // C ỉí ?        nh   d u  d
   l i u          c    n h n  ho n  t t
```

```c
int command_state = THREE_SECOND;
uint8_t cmd_data[MAX_BUFFER_SIZE];
static uint32_t last_adc_value = 0;
static uint32_t adc_value = 1234;
char adc_response[20];
int transmit_counter=0;
static enum {
    PARSER_IDLE,
    PARSER_RECEIVING,
    PARSER_DONE
  } parser_state = PARSER_IDLE;
   static enum {
       UART_IDLE,
       UART_SEND_ADC,
       UART_WAIT_OK,
       UART_RETRANSMIT
     } uart_state = UART_IDLE;

void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
    if (huart->Instance == USART2)  // Kiểm tra nếu là UART2
    {
    //  HAL_UART_Transmit(&huart2, &temp, 1, 50);
        buffer[index_buffer++] = temp; // Lưu dữ liệu nhận được vào buffer tại vị trí index_buffer
        // Nếu bộ đệm đầy, reset chỉ số về 0
        if (index_buffer == MAX_BUFFER_SIZE)
            index_buffer = 0;
    }

    // Đặt cờ buffer_flag thành 1 để báo có dữ liệu mới
    buffer_flag = 1;

    // Tiếp tục nhận dữ liệu từ UART (bật chế độ ngắt nhận UART)
    HAL_UART_Receive_IT(&huart2, &temp, 1);
}

void command_parser_fsm() {
    static uint8_t cmd_idx = 0;
    char characters = buffer[index_buffer - 1];  // Lấy ký tự mới nhất từ buffer
    switch (parser_state) {
        case PARSER_IDLE:
            if (characters == '!') {  // Bắt đầu chuỗi lệnh
```

```
97                  cmd_idx = 0;
98                  cmd_data[cmd_idx++] = characters;
99                  parser_state = PARSER_RECEIVING;
100             }
101             break;
102 
103         case PARSER_RECEIVING:
104             cmd_data[cmd_idx++] = characters;  // L u  k í
    t
105             if (characters == '#') {  //  K ỉt  th c
    c h u i   l nh
106                 parser_state = PARSER_DONE;
107                 cmd_data[cmd_idx] = '\0';  //  K ỉt  th  c
    c h u i
108             }
109             break;
110 
111         case PARSER_DONE:
112             //  K i m  tra  c h u i   l nh
113             if (strcmp((char *)cmd_data, "!RTS#") == 0) {
114                 command_state = RTS;  //  L nh  y  u  c u
    g i  gi   t r  ADC
115             } else if (strcmp((char *)cmd_data, "!OK#") ==
   0) {
116                 command_state = OK;  //  í ? nh  d u
    k ỉt  th  c  giao  t i ỉp
117             }
118             parser_state = PARSER_IDLE;  // Quay  l i
    t r ng  th  i  ban   u
119             break;
120 
121         default:
122             parser_state = PARSER_IDLE;
123             break;
124     }
125 
126     // Reset index_buffer khi   t   g i i  h n
127     if (index_buffer >= MAX_BUFFER_SIZE) {
128         index_buffer = 0;
129     }
130 }
131 
132 
133 void uart_communication_fsm() {
134 
135     switch (uart_state) {
136         case UART_IDLE:
137         //  transmit_counter=0;
138             if (command_state == RTS) {
```

```
139                 HAL_GPIO_WritePin(LED_YELLOW_GPIO_Port,
    LED_YELLOW_Pin, RESET);
140                 //  adc_value = HAL_ADC_GetValue(&hadc1);  //
      í ? í ?c  gi    t r   ADC
141                   snprintf(adc_response, sizeof(adc_response)
    , "!ADC=%lu#\r\n", (uint32_t)adc_value);  //  í ?  nh
     d ng   c h u i   l nh
142                  //     HAL_UART_Transmit(&huart2, (uint8_t *)
    adc_response, strlen(adc_response), 100);  //  G i  qua
    UART
143                   last_adc_value = adc_value;  // L u  gi
     t r   ADC
144                   uart_state = UART_RETRANSMIT;  // C h u y n
    sang  t r ng  th i  ch ỉí ?  p h n  h i
145                 //  setTimer(0, 3000);  //  B t    u
      m    th ỉí ?i  gian  3  gi  y
146                 }
147                 break;
148
149             case UART_WAIT_OK:
150                 if (isTimerExpired(0)) {  //  H ỉt  th ỉí ?i  gian
     ch ỉí ?
151                   uart_state = UART_RETRANSMIT;  // C h u y n
    sang  t r ng  th i  g i  l i
152                 }
153                 if(transmit_counter==5&&(command_state != OK)){
154                   uart_state = UART_RETRANSMIT;
155                 }
156                 if (command_state == OK) {  //  N h n  p h n
     h i  !OK#
157                 HAL_GPIO_WritePin(LED_YELLOW_GPIO_Port,
    LED_YELLOW_Pin, SET);
158                   transmit_counter=0;
159                   command_state=THREE_SECOND;
160                   uart_state = UART_IDLE;
161                 }
162                 break;
163
164             case UART_RETRANSMIT:
165               if(transmit_counter<5){
166             snprintf(adc_response, sizeof(adc_response), "!ADC
    =%lu#\r\n", (uint32_t)last_adc_value);  //  í ?  nh
     d ng   c h u i  ADC
167             HAL_UART_Transmit(&huart2, (uint8_t *)adc_response,
     strlen(adc_response), 100);  //  G i   l i  qua UART
168             if(transmit_counter<4) setTimer(0, 3000);
169               // Reset  l i  th ỉí ?i  gian  ch ỉí ?
170 //            if(transmit_counter==4) clear_timer_flag(0);
```

```
171         transmit_counter++;
172         uart_state = UART_WAIT_OK;  // Chuyn  vỉí?
    trng  th i  chỉí?  phn  h i
173           }
174           else {
175           //  uart_state = UART_IDLE;
176           //  command_state=THREE_SECOND;
177           //  transmit_counter=0;
178             uart_state = UART_WAIT_OK;
179           }
180             break;
181
182         default:
183             uart_state = UART_IDLE;
184             break;
185     }
186 }
187
188 /* USER CODE END 0 */
189
190 /**
191   * @brief  The application entry point.
192   * @retval int
193   */
194 int main(void)
195 {
196   /* USER CODE BEGIN 1 */
197
198   /* USER CODE END 1 */
199
200   /* MCU Configuration
    --------------------------------------------------------
    */
201
202   /* Reset of all peripherals, Initializes the Flash
    interface and the Systick. */
203   HAL_Init();
204
205   /* USER CODE BEGIN Init */
206
207   /* USER CODE END Init */
208
209   /* Configure the system clock */
210   SystemClock_Config();
211
212   /* USER CODE BEGIN SysInit */
213
214   /* USER CODE END SysInit */
215
```

```
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_ADC1_Init();
  MX_USART2_UART_Init();
  MX_TIM2_Init();
  /* USER CODE BEGIN 2 */
  HAL_UART_Receive_IT(&huart2, &temp, 1);
  HAL_TIM_Base_Start_IT(&htim2);
//  HAL_UART_RxCpltCallback(&huart2);
 // HAL_ADC_Start(&hadc1);
     //HAL_ADC_GetValue(&hadc1);
     command_state = THREE_SECOND;
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {

    if (buffer_flag == 1) {
                  command_parser_fsm();  // Ph n  t  ch
    l nh
                  buffer_flag = 0;       // Reset  c ỉí ?
    n h n  buffer
              }
              uart_communication_fsm();
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

Program 1.1: main.c

4 Link Github

https://github.com/quangtrungcode/STM32LAB5.git