```c

#include "Command_Handler.h"
#include "UART_Bluetooth.h"
#include "nrf24.h"


const commandType availableCommand[AVAILABLE_COMMANDS] = {
    { .commandBase = "ROTATE_FORWARDS", .nParameters = 3, .handlerFunction = &ROTATE_FORWARDS_HANDLE},
    { .commandBase = "ROTATE_BACKWARDS", .nParameters = 1, .handlerFunction = &ROTATE_BACKWARDS_HANDLE},
    { .commandBase = "TURN_LED_ON", .nParameters = 1, .handlerFunction = &TURN_LED_ON_HANDLE},
    { .commandBase = "TURN_LED_OFF", .nParameters = 1, .handlerFunction = &TURN_LED_OFF_HANDLE},
    { .commandBase = "TURN_RELAY_ON", .nParameters = 1, .handlerFunction = &TURN_RELAY_ON_HANDLE},
    { .commandBase = "TURN_RELAY_OFF", .nParameters = 1, .handlerFunction = &TURN_RELAY_OFF_HANDLE},
    { .commandBase = "UART_TEST", .nParameters = 0, .handlerFunction = &UART_TEST_HANDLER},
    { .commandBase = "BUILT_IN_LED_TEST", .nParameters = 0, .handlerFunction = &BUILT_IN_LED_TEST_HANDLER},
    { .commandBase = "TURN_EVERYTHING_ON", .nParameters = 0, .handlerFunction = &TURN_EVERYTHING_ON_HANDLE},
    { .commandBase = "TURN_EVERYTHING_OFF", .nParameters = 0, .handlerFunction = &TURN_EVERYTHING_OFF_HANDLE},
    { .commandBase = "CALL_NURSE", .nParameters = 0, .handlerFunction = &CALL_NURSE_HANDLE}
};

bool initliazeMemory(){
    if(memoryInitialized) return false;
    parameter[0] = (void*)calloc(28,1);
    parameter[1] = (void*)calloc(28,1);
    parameter[2] = (void*)calloc(28,1);
    command_buffer = (uint8_t*)calloc(32,1);
    if(parameter[0]==nullptr||parameter[1]==nullptr||parameter[2]==nullptr||command_buffer==nullptr) return false;
    memoryInitialized = true;
    return true;
}


void composeCommand(void* output_buffer, commandType* commandT, void** inputParameter){
    strcpy(output_buffer, commandT->commandBase);
    char* startParamPTR = (char*)(output_buffer+strlen(commandT->commandBase));
    char* endParamPTR = (char*)(startParamPTR+1+strlen(*inputParameter));

    for (uint8_t index = 0; index < commandT->nParameters; index++){
        *startParamPTR='[';
```

```
40          strcpy(startParamPTR+1, *inputParameter);
41          *endParamPTR=']';
42          startParamPTR=(endParamPTR+1);
43          if (index!=(commandT->nParameters-1)){
44              inputParameter++;
45              uint8_t len = strlen(*inputParameter);
46              endParamPTR = (char*)(startParamPTR+len+1);
47          }
48      }
49      *startParamPTR='\0';
50  }
51
52  bool decomposeCommand(void* input_buffer, commandType* commandT, void**
      outputParameter){
53
54      for (uint8_t index = 0; index < AVAILABLE_COMMANDS; index++){
55          if (memmem(input_buffer, COMMAND_BUFFER_SIZE, availableCommand
              [index].commandBase, strlen(availableCommand[index].commandBase))!
              =nullptr)
56          {
57              *commandT = availableCommand[index]; break;
58          }
59          else if (index==(AVAILABLE_COMMANDS-1)) { return false;}
60      }
61
62      for (uint8_t x = 0; x < commandT->nParameters; x++){
63          uint8_t* startNumPTR = memchr(input_buffer, '[', COMMAND_BUFFER_SIZE);
64          uint8_t* endNumPTR = memchr(input_buffer, ']', COMMAND_BUFFER_SIZE);
65          if (startNumPTR==nullptr||endNumPTR==nullptr) { if(x==0) return false;
              break; }
66          (*startNumPTR) = 0x20;
67          (*endNumPTR) = 0x20;
68          startNumPTR++;
69          uint32_t bytes = ((endNumPTR)) - ((startNumPTR));
70          if (bytes>PARAMETER_BUFFER_SIZE) return false;
71          memcpy(outputParameter[x], startNumPTR, bytes);
72      }
73
74      return true;
75  }
76
77  void ROTATE_FORWARDS_HANDLE() {}
78
79  void ROTATE_BACKWARDS_HANDLE() {}
80
81  void TURN_LED_ON_HANDLE() {}
82
83  void TURN_LED_OFF_HANDLE() {}
84
85  void TURN_RELAY_ON_HANDLE() {
86      composeCommand(command_buffer, &availableCommand[4], parameter);
87
```

```
 88      nrf24_send(command_buffer);
 89      while(nrf24_isSending());
 90
 91      uint8_t messageStatus = nrf24_lastMessageStatus();
 92      if(messageStatus == NRF24_TRANSMISSON_OK) { transmitMessageSync("Successful
           RF transmission! \n", 29); }
 93      else if(messageStatus == NRF24_MESSAGE_LOST) { transmitMessageSync("Failure
           on RF transmission! \n", 29); }
 94
 95      uint8_t retransmissionCount = nrf24_retransmissionCount();
 96      char* retransmissionString = malloc(32);
 97      sprintf(retransmissionString, "Retransmission count: %d \n",
           retransmissionCount);
 98      transmitMessageSync(retransmissionString, strlen(retransmissionString));
 99      free(retransmissionString);
100  }
101
102  void TURN_RELAY_OFF_HANDLE() {
103      composeCommand(command_buffer, &availableCommand[5], parameter);
104
105      nrf24_send(command_buffer);
106      while(nrf24_isSending());
107
108      uint8_t messageStatus = nrf24_lastMessageStatus();
109      if(messageStatus == NRF24_TRANSMISSON_OK) { transmitMessageSync("Successful
           RF transmission! \n", 29); }
110      else if(messageStatus == NRF24_MESSAGE_LOST) { transmitMessageSync("Failure
           on RF transmission! \n", 29); }
111
112      uint8_t retransmissionCount = nrf24_retransmissionCount();
113      char* retransmissionString = malloc(32);
114      sprintf(retransmissionString, "Retransmission count: %d \n",
           retransmissionCount);
115      transmitMessageSync(retransmissionString, strlen(retransmissionString));
116      free(retransmissionString);
117  }
118
119  void UART_TEST_HANDLER() {
120      transmitMessageSync("Successful UART transmission!\n", 30);
121  }
122
123  void BUILT_IN_LED_TEST_HANDLER(){
124      for (uint8_t x = 0; x < 8; x++) {
125          bit_flip(PORTD, BIT(7));
126          bit_flip(PORTB, BIT(0));
127          _delay_ms(250);
128      }
129      bit_clear(PORTD, BIT(7));
130      bit_clear(PORTB, BIT(0));
131  }
132
133  void TURN_EVERYTHING_ON_HANDLE(){
```

```
134     composeCommand(command_buffer, &availableCommand[8], parameter);
135
136     nrf24_send(command_buffer);
137     while(nrf24_isSending());
138
139     uint8_t messageStatus = nrf24_lastMessageStatus();
140     if(messageStatus == NRF24_TRANSMISSON_OK) { transmitMessageSync("Successful
          RF transmission! \n", 29); }
141     else if(messageStatus == NRF24_MESSAGE_LOST) { transmitMessageSync("Failure
          on RF transmission! \n", 29); }
142
143     uint8_t retransmissionCount = nrf24_retransmissionCount();
144     char* retransmissionString = malloc(32);
145     sprintf(retransmissionString, "Retransmission count: %d \n",
          retransmissionCount);
146     transmitMessageSync(retransmissionString, strlen(retransmissionString));
147     free(retransmissionString);
148 }
149
150 void TURN_EVERYTHING_OFF_HANDLE(){
151     composeCommand(command_buffer, &availableCommand[9], parameter);
152
153     nrf24_send(command_buffer);
154     while(nrf24_isSending());
155
156     uint8_t messageStatus = nrf24_lastMessageStatus();
157     if(messageStatus == NRF24_TRANSMISSON_OK) { transmitMessageSync("Successful
          RF transmission! \n", 29); }
158     else if(messageStatus == NRF24_MESSAGE_LOST) { transmitMessageSync("Failure
          on RF transmission! \n", 29); }
159
160     uint8_t retransmissionCount = nrf24_retransmissionCount();
161     char* retransmissionString = malloc(32);
162     sprintf(retransmissionString, "Retransmission count: %d \n",
          retransmissionCount);
163     transmitMessageSync(retransmissionString, strlen(retransmissionString));
164     free(retransmissionString);
165 }
166
167 void CALL_NURSE_HANDLE(){
168     composeCommand(command_buffer, &availableCommand[10], parameter);
169
170     nrf24_send(command_buffer);
171     while(nrf24_isSending());
172
173     uint8_t messageStatus = nrf24_lastMessageStatus();
174     if(messageStatus == NRF24_TRANSMISSON_OK) { transmitMessageSync("Successful
          RF transmission! \n", 29); }
175     else if(messageStatus == NRF24_MESSAGE_LOST) { transmitMessageSync("Failure
          on RF transmission! \n", 29); }
176
177     uint8_t retransmissionCount = nrf24_retransmissionCount();
```

```
178     char* retransmissionString = malloc(32);
179     sprintf(retransmissionString, "Retransmission count: %d \n",
          retransmissionCount);
180     transmitMessageSync(retransmissionString, strlen(retransmissionString));
181     free(retransmissionString);
182 }
183
184
```