SAPLM

Código de fuente: Módulo principal (Lenguaje: AVR-GCC)

Contenidos:

main.c
Command_Handler.c
Command_Handler.h
UART_Bluetooth.c
UART_Bluetooth.h
crc.c
crc.h
nrf24.c
nrf24.h

```c
#define F_CPU                   16000000UL

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <stdint.h>

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

void initIO();
char messageTest[] = "UART TESTING COMMANDS! \n";

int main(void)
{
    cli();  // Interrupts off
    initIO();
    initBluetoothUart();
    setupReceiveMode();
    nrf24_initRF_SAFE(POWER_BOARD_RF, RECEIVE); // CONNECTION TO POWER BOARD AND MOTORIZED BOARD : GENERAL RF CHANNEL 11
    sei();  // Interrupts on
    while (1)
    {
        if (commandAvailable) {
            cli();
            processReceivedLine();
            setupReceiveMode();

        }

         // Disable UART

        if(nrf24_dataReady())
        {
            cli();
            nrf24_getData(command_buffer);
            CommandStatus status = DecomposeMessageFromBuffer();
            if (status==SUCCESFUL_DECOMPOSITION) { RetransmissionToPhone();  }
            sei();
        }

        if (nrf24_checkAvailability()==false) { nrf24_initRF_SAFE(POWER_BOARD_RF, RECEIVE); }

    }
}


```

```
51  void initIO(){
52      /*
53          Input/Output pin initialization
54          1 : OUTPUT | 0 : INPUT | 0b76543210 Bit order
55          ATTACHMENTS
56              RED LED     : PD7               |   OUTPUT
57              GREEN LED   : PB0               |   OUTPUT
58          HC-05
59              TX          : PD0 (RX ATMEGA)   |   INPUT
60              RX          : PD1 (TX ATMEGA)   |   OUTPUT
61              KEY/ENABLE  : PD2               |   OUTPUT
62              STATE       : PC5               |   INPUT
63          nRF24L01
64              CE   : PC0                      |   OUTPUT
65              CSN  : PC1                      |   OUTPUT
66              MISO : PD0 (MSPIM MISO ATMEGA)  |   INPUT
67              MOSI : PD1 (MSPIM MOSI ATMEGA)  |   OUTPUT
68              SCK  : PD4 (MSPIM XCK)          |   OUTPUT
69      */
70      DDRD = 0b11111110;
71      DDRB = 0b00101001;
72      DDRC = 0b11011111;
73  }
74
75
76
77
78
79
```

```c

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



const CommandType commandList[] = {
    { .handlerFunction = &UPDATE_ALL_DEVICES_VALUE_H},
    { .handlerFunction = &UPDATE_DEVICE_VALUE_H},
    { .handlerFunction = &GET_ALL_DEVICES_VALUE_H},
    { .handlerFunction = &GET_DEVICE_VALUE_H},
    { .handlerFunction = &MESSAGE_STATUS_H}
};
#define commandListLength (uint8_t)(sizeof commandList/sizeof commandList[0])

bool initliazeMemory(){
    if(memoryInitialized) return false;
    parameter[0].startingPointer = (void*)calloc(23,1);
    parameter[1].startingPointer = (void*)calloc(2,1);
    parameter[2].startingPointer = (void*)calloc(2,1);
    for (uint8_t x = 3; x<12; x++) parameter[x].startingPointer = (void*)calloc
      (1,1);
    command_buffer = (uint8_t*)calloc(32,1);
    if(command_buffer==NULL) return false;
    for (uint8_t x = 0; x<12; x++) { if(parameter[x].startingPointer==NULL)
      return false; }
    memoryInitialized = true;
    return true;
}

CommandStatus DecomposeMessageFromBuffer(){
    // Search for header
    uint8_t* headerStart = command_buffer;
    uint8_t* footerEnd = command_buffer+31;

    for(;headerStart!=(command_buffer+22);headerStart++){
        if (*headerStart==SOH&&(*(headerStart+4)==STX)){
            for(;footerEnd!=(command_buffer+6);footerEnd--){
                if (*footerEnd==ETB&&(*(footerEnd-2)==ETX)){
                    uint8_t netMessageLength = ((footerEnd-2)-headerStart);
                    crc_t crc;
                    crc = crc_init();
                    crc = crc_update(crc, headerStart, netMessageLength);
                    crc = crc_finalize(crc);
                    if (*(footerEnd-1)!=crc) return WRONG_CHECKSUM_CONSISTENCY;
                    if (*(headerStart+2)!=currentModuleID&&*(headerStart+2)!
                     =0xFF&&currentModuleID!=0x01) return WRONG_MODULE_ID;
                    lastTargetModuleID = *(headerStart+2);
                    lastTransmitterModuleID = *(headerStart+3);
                    if (*(headerStart+5)>commandListLength-1) return
```

```
                    UNDEFINED_COMMAND_CODE;
50                  lastMessageCommandType = commandList[*(headerStart+5)];
51                  lastMessagePID = *(headerStart+1);
52
53                  uint8_t* parameterStart = headerStart+6;
54
55                  for (uint8_t x = 0; x < 12; x++) {
56                      realloc(parameter[x].startingPointer, *parameterStart);
57                      parameter[x].byteLength = *parameterStart;
58                      memcpy(parameter[x].startingPointer,parameterStart+1,
                     *parameterStart);
59                      parameterStart+=((*parameterStart)+1);
60                      if (parameterStart>=(footerEnd-2)) break;
61                  }
62
63                  return SUCCESFUL_DECOMPOSITION;
64              }
65          }
66      }
67  }
68  return WRONG_HEADER_SEGMENTATION;
69 }
70
71 CommandStatus ComposeMessageToBuffer(CommandTypeID targetTypeID, uint8_t
    parameterCount, uint8_t targetBoardID){
72
73      memset(command_buffer, 0, 32);
74
75      command_buffer[0] = SOH;
76      if (lastMessagePID==0xFF) { lastMessagePID++; } else { lastMessagePID = 0; }
77      command_buffer[1] = lastMessagePID;
78      command_buffer[2] = targetBoardID;
79      command_buffer[3] = currentModuleID;
80      command_buffer[4] = STX;
81      command_buffer[5] = targetTypeID;
82
83      if (parameterCount>12) return PARAMETER_COUNT_OVERSIZE;
84
85      uint8_t* parameterStart = &command_buffer[6];
86
87      for (uint8_t x = 0; x < parameterCount; x++){
88          *parameterStart = parameter[x].byteLength;
89          memcpy(parameterStart+1, parameter[x].startingPointer, parameter
              [x].byteLength);
90          parameterStart+=(parameter[x].byteLength)+1;
91      }
92
93      crc_t crc;
94      crc = crc_init();
95      uint8_t crc_length = ((parameterStart)-(&command_buffer[0]));
96      crc = crc_update(crc, &command_buffer[0], crc_length);
97      crc = crc_finalize(crc);
```

```
 98
 99      *parameterStart = ETX;
100      *(parameterStart+1) = crc;
101      *(parameterStart+2) = ETB;
102
103      return SUCCESFUL_COMPOSITION;
104  }
105
106  void HandleAvailableCommand(){
107      lastMessageCommandType.handlerFunction();
108  }
109
110  RF_TransmissionStatus RetransmissionToModule(){
111      nrf24_initRF_SAFE((lastTargetModuleID-1), TRANSMIT);    // CONNECTION TO
           MODULE: GENERAL RF CHANNEL 112, (lastTargetModuleID-1) offset 1
112      nrf24_send(command_buffer);
113      while(nrf24_isSending());
114
115      uint8_t messageStatus = nrf24_lastMessageStatus();
116      if(messageStatus == NRF24_TRANSMISSON_OK) { return
           RF_SUCCESFUL_TRANSMISSION; }
117      else if(messageStatus == NRF24_MESSAGE_LOST) { return
           RF_UNREACHEABLE_MODULE;}
118      return RF_UNREACHEABLE_MODULE;
119  }
120
121  void RetransmissionToPhone(){
122      transmitMessageSync(command_buffer, 32);
123  }
124
125
126
127  void writeParameterValue(uint8_t parameterIndex, uint8_t* parameterData, uint8_t
       parameterByteLength){
128      parameter[parameterIndex].startingPointer = (uint8_t*) realloc(parameter
           [parameterIndex].startingPointer, parameterByteLength);
129      memcpy(parameter[parameterIndex].startingPointer, parameterData,
           parameterByteLength);
130      parameter[parameterIndex].byteLength = parameterByteLength;
131  }
132
133  void UPDATE_ALL_DEVICES_VALUE_H() {}
134  void UPDATE_DEVICE_VALUE_H() {}
135  void GET_ALL_DEVICES_VALUE_H() {
136      _delay_ms(100);
137
138      uint8_t boardState[2];
139
140      ComposeMessageToBuffer(MESSAGE_STATUS_ID, 0, POWER_MODULE);
141      nrf24_initRF_SAFE(POWER_BOARD_RF, TRANSMIT);    // CONNECTION TO MODULE:
           GENERAL RF CHANNEL 112
142      nrf24_send(command_buffer);
```

```
143     while(nrf24_isSending());
144
145     uint8_t messageStatus = nrf24_lastMessageStatus();
146     if(messageStatus == NRF24_TRANSMISSON_OK) { boardState[0] = 0xFF; }
147     else if(messageStatus == NRF24_MESSAGE_LOST) { boardState[0]= 0x00; }
148
149     _delay_ms(50);
150
151     ComposeMessageToBuffer(MESSAGE_STATUS_ID, 0, MOTOR_MODULE);
152     nrf24_initRF_SAFE(MOTORIZED_BOARD_RF, TRANSMIT);    // CONNECTION TO MODULE:
          GENERAL RF CHANNEL 112
153     nrf24_send(command_buffer);
154     while(nrf24_isSending());
155
156     uint8_t messageStatusSecond = nrf24_lastMessageStatus();
157     if(messageStatusSecond == NRF24_TRANSMISSON_OK) { boardState[1] = 0xFF; }
158     else if(messageStatusSecond == NRF24_MESSAGE_LOST) { boardState[1]= 0x00; }
159
160
161     writeParameterValue(0, &boardState[0], 1);
162     writeParameterValue(1, &boardState[1], 1);
163     ComposeMessageToBuffer(UPDATE_ALL_DEVICES_VALUE_ID, 2, PHONE_MODULE);  //
          PHONE_MODULE should be lastTransmitterModuleID
164     transmitMessageSync(command_buffer, 32);
165 }
166
167 void GET_DEVICE_VALUE_H() {
168     _delay_ms(100);
169     uint8_t deviceIndex = *((uint8_t*)parameter[0].startingPointer);
170     uint8_t deviceValue;
171
172     switch(deviceIndex){
173         case 0:
174             ComposeMessageToBuffer(MESSAGE_STATUS_ID, 0, POWER_MODULE);
175             nrf24_initRF_SAFE(POWER_BOARD_RF, TRANSMIT);    // CONNECTION TO
                  MODULE: GENERAL RF CHANNEL 112
176             nrf24_send(command_buffer);
177             while(nrf24_isSending());
178
179             uint8_t messageStatus = nrf24_lastMessageStatus();
180             if(messageStatus == NRF24_TRANSMISSON_OK) { deviceValue = 0xFF; }
181             else if(messageStatus == NRF24_MESSAGE_LOST) { deviceValue= 0x00; }
182             break;
183         case 1:
184             ComposeMessageToBuffer(MESSAGE_STATUS_ID, 0, MOTOR_MODULE);
185             nrf24_initRF_SAFE(MOTORIZED_BOARD_RF, TRANSMIT);    // CONNECTION TO
                  MODULE: GENERAL RF CHANNEL 112
186             nrf24_send(command_buffer);
187             while(nrf24_isSending());
188
189             uint8_t messageStatusSecond = nrf24_lastMessageStatus();
190             if(messageStatusSecond == NRF24_TRANSMISSON_OK) { deviceValue =
```

```
                0xFF; }
191             else if(messageStatusSecond == NRF24_MESSAGE_LOST) { deviceValue=
                0x00; }
192             break;
193     }
194
195     writeParameterValue(0, &deviceIndex, 1);
196     writeParameterValue(1, &deviceValue, 2);
197
198     ComposeMessageToBuffer(UPDATE_DEVICE_VALUE_ID, 2, PHONE_MODULE);  //
        PHONE_MODULE should be lastTransmitterModuleID
199
200     transmitMessageSync(command_buffer, 32);
201 }
202 void MESSAGE_STATUS_H() {}
```

```
 1
 2
 3  #ifndef COMMAND_HANDLER_H_
 4  #define COMMAND_HANDLER_H_
 5
 6  #ifndef nullptr
 7  #define nullptr ((void *)0)
 8  #endif
 9
10  #ifndef F_CPU
11  #define F_CPU               16000000UL
12  #endif
13
14  #include <stdbool.h>
15  #include <stdint.h>
16  #include <stdio.h>
17  #include <string.h>
18  #include <stdlib.h>
19  #include <avr/io.h>
20  #include <util/delay.h>
21
22  #ifndef BIT_MANIPULATION_MACRO
23  #define BIT_MANIPULATION_MACRO 1
24  #define bit_get(p,m) ((p) & (m))
25  #define bit_set(p,m) ((p) |= (m))
26  #define bit_clear(p,m) ((p) &= ~(m))
27  #define bit_flip(p,m) ((p) ^= (m))
28  #define bit_write(c,p,m) (c ? bit_set(p,m) : bit_clear(p,m))
29  #define BIT(x) (0x01 << (x))
30  #define LONGBIT(x) ((unsigned long)0x00000001 << (x))
31  #endif
32
33  typedef struct CommandType {
34      void (*handlerFunction)();
35  } CommandType;
36
37  typedef enum   {
38      SUCCESFUL_DECOMPOSITION,
39      WRONG_HEADER_SEGMENTATION,
40      WRONG_FOOTER_SEGMENTATION,
41      WRONG_CHECKSUM_CONSISTENCY,
42      WRONG_MODULE_ID,
43      UNDEFINED_COMMAND_CODE,
44      PARAMETER_DATA_OVERFLOW,
45      PARAMETER_COUNT_OVERSIZE,
46      RETRANSMISSION_FAILED,
47      SUCCESFUL_RETRANSMISSION,
48      SUCCESFUL_COMPOSITION
49  } CommandStatus;
50
51
52  typedef enum {
```

```
53      RF_SUCCESFUL_TRANSMISSION,
54      RF_UNREACHEABLE_MODULE,
55      RF_ACKNOWLEDGE_FAILED
56  } RF_TransmissionStatus;
57
58  typedef enum   {
59      UPDATE_ALL_DEVICES_VALUE_ID,
60      UPDATE_DEVICE_VALUE_ID,
61      GET_ALL_DEVICES_VALUE_ID,
62      GET_DEVICE_VALUE_ID,
63      MESSAGE_STATUS_ID
64  } CommandTypeID;
65
66  typedef struct {
67      void *startingPointer;
68      uint8_t byteLength;
69  } Parameter;
70
71  typedef enum   {
72      PHONE_MODULE = 0x00,
73      MAIN_MODULE = 0x01,
74      POWER_MODULE = 0x02,
75      MOTOR_MODULE = 0x03,
76  } ModuleInternalCode;
77
78
79  #define currentModuleID MAIN_MODULE
80
81  #define SOH 0x01
82  #define STX 0x02
83  #define ETX 0x03
84  #define ETB 0x17
85  #define ON_STATE    0xFF
86  #define OFF_STATE   0x00
87
88  #define AVAILABLE_DEVICES 4
89  uint16_t device_value[AVAILABLE_DEVICES];
90
91  uint8_t *command_buffer;
92  Parameter parameter[12];
93  bool memoryInitialized;
94
95  uint8_t lastMessagePID;
96  uint8_t lastTargetModuleID;
97  uint8_t lastTransmitterModuleID;
98  CommandType lastMessageCommandType;
99
100 extern bool initliazeMemory();
101 extern void UPDATE_ALL_DEVICES_VALUE_H(), UPDATE_DEVICE_VALUE_H(),
      GET_ALL_DEVICES_VALUE_H(), GET_DEVICE_VALUE_H(), MESSAGE_STATUS_H();
102 extern CommandStatus DecomposeMessageFromBuffer();
103 extern void HandleAvailableCommand();
```

```
104  extern RF_TransmissionStatus RetransmissionToModule();
105  extern CommandStatus ComposeMessageToBuffer(CommandTypeID targetTypeID, uint8_t
       parameterCount, uint8_t targetBoardID);
106  void writeParameterValue(uint8_t parameterIndex, uint8_t* parameterData, uint8_t
       parameterByteLength);
107  void RetransmissionToPhone();
108  #endif /* COMMAND_HANDLER_H_ */
```

```
 1
 2
 3  #include "UART_Bluetooth.h"
 4  #include <avr/io.h>
 5  #include <avr/interrupt.h>
 6  #include "Command_Handler.h"
 7  #include "nrf24.h"
 8  #include <stdlib.h>
 9  #include <string.h>
10
11  uint8_t* uartBufferPos;
12  uint8_t* uartTxMessageEnd;
13  bool commandAvailable;
14
15  void initBluetoothUart(){
16      // UART Initialization : 8-bit : No parity bit : 1 stop bit
17      UBRR0H = (BRC >> 8); UBRR0L =  BRC;             // UART BAUDRATE
18      UCSR0A |= (1 << U2X0);                          // DOUBLE UART SPEED
19      UCSR0C |= (1 << UCSZ01) | (1 << UCSZ00);        // 8-BIT CHARACTER SIZE
20
21      // Setup UART buffer
22      initliazeMemory();
23      uartBufferPos = command_buffer;
24  }
25
26  void transmitMessage(uint8_t* message, uint8_t length){
27      while (!(UCSR0A & (1<<UDRE0)));
28      uartBufferPos = command_buffer;
29      uartTxMessageEnd = (command_buffer+length);
30      memcpy(command_buffer, message, length);
31      UCSR0A |= (1<<TXC0) | (1<<RXC0);
32      UCSR0B |= (1<<TXEN0) | (1<<TXCIE0);
33      UCSR0B &=~(1<<RXEN0) &~(1<<RXCIE0);
34
35      uartBufferPos++;
36      UDR0 = *(command_buffer);
37  }
38
39  void transmitMessageSync(uint8_t* message, uint8_t length){
40      while (!(UCSR0A & (1<<UDRE0)));
41      uartBufferPos = command_buffer;
42      uartTxMessageEnd = (command_buffer+length);
43      memcpy(command_buffer, message, length);
44      UCSR0A |= (1<<TXC0) | (1<<RXC0);
45      UCSR0B |= (1<<TXEN0) | (1<<TXCIE0);
46      UCSR0B &=~(1<<RXEN0) &~(1<<RXCIE0);
47      sei();
48
49      uartBufferPos++;
50      UDR0 = *(command_buffer);
51
52      while (transmissionState());
```

```
53
54  }
55
56  bool transmissionState(){
57      // True : Currently transmitting | False : Transmission finished
58      if (uartBufferPos!=uartTxMessageEnd)
59      {
60          return true;
61      }
62      else
63      {
64          return false;
65      }
66  }
67
68
69  void setupReceiveMode(){
70      while (!(UCSR0A & (1<<UDRE0)));
71      uartBufferPos = command_buffer;
72
73      UCSR0A |= (1<<RXC0) | (1<<TXC0);
74      UCSR0B &=~(1<<TXEN0) &~(1<<TXCIE0);
75      UCSR0B |= (1<<RXEN0) | (1<<RXCIE0);
76      sei();
77  }
78
79  bool catchModuleReply(){
80      nrf24_initRF_SAFE((lastTargetModuleID-1), RECEIVE); // CONNECTION TO MODULE:
          GENERAL RF CHANNEL 112 (lastTargetModuleID-1) offset 1
81      uint8_t targetModuleID = lastTargetModuleID;
82      uint8_t RF_TIME_OUT;
83      while(RF_TIME_OUT!=0xFF)
84      {
85          if(nrf24_dataReady()){
86              nrf24_getData(command_buffer);
87              CommandStatus status = DecomposeMessageFromBuffer();
88              if
                  (status==SUCCESFUL_DECOMPOSITION&&lastTargetModuleID==targetModuleI
                  D) {
89                  transmitMessageSync(command_buffer, 32);
90                  return true;
91              }
92          }
93          RF_TIME_OUT++; _delay_ms(2);
94      }
95      return false;
96  }
97
98  void processReceivedLine(){
99      commandAvailable = false;
100
101     CommandStatus status = DecomposeMessageFromBuffer();
```

```
102     if(status==SUCCESFUL_DECOMPOSITION) {
103         if (lastTargetModuleID==MAIN_MODULE){
104             //Executed by main module
105             HandleAvailableCommand();
106         } else {
107             //Retransmitted to other module
108 
109             RF_TransmissionStatus RF_Status = RetransmissionToModule();
110 
111             //Catch module reply
112 
113             //bool didModuleRelpy = catchModuleReply();
114 
115             // Send RF STATUS
116             switch (RF_Status) {
117                 case RF_UNREACHEABLE_MODULE:
118                 writeParameterValue(0, &(uint8_t){RETRANSMISSION_FAILED}, 1);
119                 break;
120                 case RF_ACKNOWLEDGE_FAILED:
121                 writeParameterValue(0, &(uint8_t){RETRANSMISSION_FAILED}, 1);
122                 break;
123                 case RF_SUCCESFUL_TRANSMISSION:
124                 writeParameterValue(0, &(uint8_t){SUCCESFUL_RETRANSMISSION}, 1);
125                 break;
126             }
127             ComposeMessageToBuffer(MESSAGE_STATUS_ID, 1, PHONE_MODULE);
128             transmitMessageSync(command_buffer, 32);
129 
130 
131         }
132     }else {
133 }
134 
135 
136 }
137 
138 void disableUART(){
139     UCSR0B &=~(1<<TXEN0) &~(1<<TXCIE0);
140     UCSR0B &=~(1<<RXEN0) &~(1<<RXCIE0);
141 }
142 
143 ISR(USART_TX_vect){
144     if (uartBufferPos!=uartTxMessageEnd){
145         UDR0 = *uartBufferPos;
146         uartBufferPos++;
147     }
148 }
149 
150 ISR(USART_RX_vect){
151     if(uartBufferPos!=(command_buffer+uartBufferSize)) {
152         *uartBufferPos=UDR0;
153         if ((*uartBufferPos==ETB)&&(DecomposeMessageFromBuffer()
```

```
            ==SUCCESFUL_DECOMPOSITION)) {
154             disableUART(); commandAvailable = true;
155         }
156         else if(*uartBufferPos==uartCarriageReturnChar) {
157
158             bool hasToReturnCarriage = true;
159             uint8_t* savedUartBufferPos = uartBufferPos+1;
160
161             for (uint8_t x = 1; x < 4; x++) {
162                 if ((uartBufferPos-x)<command_buffer) uartBufferPos =
                      command_buffer+(uartBufferSize-1);
163                 if (*(uartBufferPos-x)!=uartCarriageReturnChar)
                      { hasToReturnCarriage = false; break; }
164             }
165             if (hasToReturnCarriage) {
166                  uartBufferPos = command_buffer;
167
168             } else {
169                 uartBufferPos = savedUartBufferPos;
170             }
171
172         } else {
173             uartBufferPos++;
174         }
175
176     } else {
177         uartBufferPos = command_buffer;
178         *uartBufferPos=UDR0;
179     }
180 }
```

```
1
2
3  #ifndef UART_BLUETOOTH_H_
4  #define UART_BLUETOOTH_H_
5
6
7  #include <stdbool.h>
8  #include <stdint.h>
9
10 #ifndef F_CPU
11 #define F_CPU           16000000UL
12 #endif
13
14 #ifndef BAUD
15 #define BAUD            9600
16 #endif
17
18 #ifndef BRC
19 #define BRC             F_CPU/8/BAUD-1
20 #endif
21
22 #ifndef nullptr
23 #define nullptr         nullptr ((void*)0)
24 #endif
25
26 #define uartBufferSize          32
27 #define uartEndMsgChar          '$'
28 #define uartCarriageReturnChar  0x7F
29
30 #ifndef BIT_MANIPULATION_MACRO
31 #define BIT_MANIPULATION_MACRO 1
32 #define bit_get(p,m) ((p) & (m))
33 #define bit_set(p,m) ((p) |= (m))
34 #define bit_clear(p,m) ((p) &= ~(m))
35 #define bit_flip(p,m) ((p) ^= (m))
36 #define bit_write(c,p,m) (c ? bit_set(p,m) : bit_clear(p,m))
37 #define BIT(x) (0x01 << (x))
38 #define LONGBIT(x) ((unsigned long)0x00000001 << (x))
39 #endif
40
41
42 extern bool commandAvailable;
43
44 extern void initBluetoothUart();
45 extern void transmitMessage(uint8_t* message, uint8_t length);
46 extern void transmitMessageSync(uint8_t* message, uint8_t length);
47 extern bool transmissionState();
48 extern void setupReceiveMode();
49 extern void processReceivedLine();
50 extern void disableUART();
51
52
```

```
53
54  #endif /* UART_BLUETOOTH_H_ */
```

```c
/**
 * \file
 * Functions and types for CRC checks.
 *
 * Generated on Wed Sep 11 13:55:53 2019
 * by pycrc v0.9.2, https://pycrc.org
 * using the configuration:
 *  - Width         = 8
 *  - Poly          = 0x07
 *  - XorIn         = 0x00
 *  - ReflectIn     = False
 *  - XorOut        = 0x00
 *  - ReflectOut    = False
 *  - Algorithm     = bit-by-bit-fast
 */
#include "crc.h"     /* include the header file generated with pycrc */
#include <stdlib.h>
#include <stdint.h>
#include <stdbool.h>



crc_t crc_update(crc_t crc, const void *data, size_t data_len)
{
    const unsigned char *d = (const unsigned char *)data;
    unsigned int i;
    bool bit;
    unsigned char c;

    while (data_len--) {
        c = *d++;
        for (i = 0x80; i > 0; i >>= 1) {
            bit = crc & 0x80;
            if (c & i) {
                bit = !bit;
            }
            crc <<= 1;
            if (bit) {
                crc ^= 0x07;
            }
        }
        crc &= 0xff;
    }
    return crc & 0xff;
}

```

```c
/**
 * \file
 * Functions and types for CRC checks.
 *
 * Generated on Wed Sep 11 13:56:48 2019
 * by pycrc v0.9.2, https://pycrc.org
 * using the configuration:
 *  - Width         = 8
 *  - Poly          = 0x07
 *  - XorIn         = 0x00
 *  - ReflectIn     = False
 *  - XorOut        = 0x00
 *  - ReflectOut    = False
 *  - Algorithm     = bit-by-bit-fast
 *
 * This file defines the functions crc_init(), crc_update() and crc_finalize().
 *
 * The crc_init() function returns the inital \c crc value and must be called
 * before the first call to crc_update().
 * Similarly, the crc_finalize() function must be called after the last call
 * to crc_update(), before the \c crc is being used.
 * is being used.
 *
 * The crc_update() function can be called any number of times (including zero
 * times) in between the crc_init() and crc_finalize() calls.
 *
 * This pseudo-code shows an example usage of the API:
 * \code{.c}
 * crc_t crc;
 * unsigned char data[MAX_DATA_LEN];
 * size_t data_len;
 *
 * crc = crc_init();
 * while ((data_len = read_data(data, MAX_DATA_LEN)) > 0) {
 *     crc = crc_update(crc, data, data_len);
 * }
 * crc = crc_finalize(crc);
 * \endcode
 */
#ifndef CRC_H
#define CRC_H

#include <stdlib.h>
#include <stdint.h>

#ifdef __cplusplus
extern "C" {
#endif


/**
 * The definition of the used algorithm.
```

```
53   *
54   * This is not used anywhere in the generated code, but it may be used by the
55   * application code to call algorithm-specific code, if desired.
56   */
57  #define CRC_ALGO_BIT_BY_BIT_FAST 1
58
59
60  /**
61   * The type of the CRC values.
62   *
63   * This type must be big enough to contain at least 8 bits.
64   */
65  typedef uint_fast8_t crc_t;
66
67
68  /**
69   * Calculate the initial crc value.
70   *
71   * \return     The initial crc value.
72   */
73  static inline crc_t crc_init(void)
74  {
75      return 0x00;
76  }
77
78
79  /**
80   * Update the crc value with new data.
81   *
82   * \param[in] crc      The current crc value.
83   * \param[in] data     Pointer to a buffer of \a data_len bytes.
84   * \param[in] data_len Number of bytes in the \a data buffer.
85   * \return             The updated crc value.
86   */
87  crc_t crc_update(crc_t crc, const void *data, size_t data_len);
88
89
90  /**
91   * Calculate the final crc value.
92   *
93   * \param[in] crc  The current crc value.
94   * \return     The final crc value.
95   */
96  static inline crc_t crc_finalize(crc_t crc)
97  {
98      return crc;
99  }
100
101
102 #ifdef __cplusplus
103 }           /* closing brace for extern "C" */
104 #endif
```

```
105
106 #endif      /* CRC_H */
107
```

```
1
2  #define UCPHA0 1
3
4  #include "nrf24.h"
5  #include "UART_Bluetooth.h"
6
7  volatile uint8_t payload_len;
8  volatile uint8_t selectedChannel;
9
10 uint8_t MOTORIZED_BOARD_ADDR[5] =   {0xF0,0xF0,0xF0,0xF0,0xC9};
11 uint8_t MAIN_BOARD_ADDR[5] =           {0xA4,0xA4,0xA4,0xA4,0xA4};
12 uint8_t POWER_BOARD_ADDR[5] =       {0xF0,0xF0,0xF0,0xF0,0xF0};
13
14 uint8_t NULL_ADDR[5] =      {0x00,0x00,0x00,0x00,0x00};
15
16 uint8_t* BOARD_ADDRESS[3] = {&MAIN_BOARD_ADDR[0], &POWER_BOARD_ADDR[0],
     &MOTORIZED_BOARD_ADDR[0]};
17 uint8_t* CURRENT_BOARD_ADDRESS = &MAIN_BOARD_ADDR[0];
18
19 const uint8_t GENERAL_RF_CHANNEL = 112;
20
21
22 void nrf24_init()
23 {
24     nrf24_setupPins();
25     nrf24_ce_digitalWrite(LOW);
26     nrf24_csn_digitalWrite(HIGH);
27 }
28
29 void nrf24_config(uint8_t channel, uint8_t pay_length)
30 {
31     /* Use static payload length ... */
32     payload_len = pay_length;
33     selectedChannel = channel;
34
35     // Set RF channel
36     nrf24_configRegister(RF_CH,channel);
37
38     // Set length of incoming payload
39     nrf24_configRegister(RX_PW_P0, 0x00); // Auto-ACK pipe ...
40     nrf24_configRegister(RX_PW_P1, payload_len); // Data payload pipe
41     nrf24_configRegister(RX_PW_P2, 0x00); // Pipe not used
42     nrf24_configRegister(RX_PW_P3, 0x00); // Pipe not used
43     nrf24_configRegister(RX_PW_P4, 0x00); // Pipe not used
44     nrf24_configRegister(RX_PW_P5, 0x00); // Pipe not used
45
46     // 1 Mbps, TX gain: 0dbm
47     nrf24_configRegister(RF_SETUP, (0<<RF_DR)|((0x03)<<RF_PWR));
48
49     // CRC enable, 1 byte CRC length
50     nrf24_configRegister(CONFIG,nrf24_CONFIG);
51
```

```
52      // Auto Acknowledgment
53      nrf24_configRegister(EN_AA,(1<<ENAA_P0)|(1<<ENAA_P1)|(0<<ENAA_P2)|
          (0<<ENAA_P3)|(0<<ENAA_P4)|(0<<ENAA_P5));
54
55      // Enable RX addresses
56      nrf24_configRegister(EN_RXADDR,(1<<ERX_P0)|(1<<ERX_P1)|(0<<ERX_P2)|
          (0<<ERX_P3)|(0<<ERX_P4)|(0<<ERX_P5));
57
58      // Auto retransmit delay: 1000 us and Up to 15 retransmit trials
59      nrf24_configRegister(SETUP_RETR,(0x04<<ARD)|(0x0F<<ARC));
60
61      // Dynamic length configurations: No dynamic length
62      nrf24_configRegister(DYNPD,(0<<DPL_P0)|(0<<DPL_P1)|(0<<DPL_P2)|(0<<DPL_P3)|
          (0<<DPL_P4)|(0<<DPL_P5));
63
64  }
65
66
67
68  bool nrf24_checkConfig(){
69      // Check all registers
70      if (nrf24_checkRegister(RF_CH, selectedChannel,1)==false) return false;
71      if (nrf24_checkRegister(RF_SETUP, (0<<RF_DR)|((0x03)<<RF_PWR),1)==false)
          return false;
72      if (nrf24_checkRegister(CONFIG,nrf24_CONFIG,1)==false) return false;
73      if (nrf24_checkRegister(SETUP_RETR,(0x04<<ARD)|(0x0F<<ARC),1)==false) return
          false;
74      if (nrf24_checkRegister(DYNPD,(0<<DPL_P0)|(0<<DPL_P1)|(0<<DPL_P2)|
          (0<<DPL_P3)|(0<<DPL_P4)|(0<<DPL_P5),1)==false) return false;
75
76      return true;
77  }
78
79  bool nrf24_checkAvailability(){
80      if (nrf24_checkRegister(RF_CH, selectedChannel,1)==true) { return true; }
          else { return false;}
81  }
82
83
84
85
86  void faultyRF_Alarm(){
87      CLEAR_FAULTY_RF_LED;
88      for (uint8_t x = 0; x < 6; x++)
89      {
90          FLIP_FAULTY_RF_LED;
91          _delay_ms(125);
92      }
93      _delay_ms(250);
94  }
95
96
```

```
 97
 98 /* Set the RX address */
 99 void nrf24_rx_address(uint8_t * adr)
100 {
101     nrf24_ce_digitalWrite(LOW);
102     nrf24_writeRegister(RX_ADDR_P1,adr,nrf24_ADDR_LEN);
103     nrf24_ce_digitalWrite(HIGH);
104 }
105
106 /* Set the secondary RX address */
107 void nrf24_secondary_rx_address(uint8_t * adr)
108 {
109     nrf24_ce_digitalWrite(LOW);
110     nrf24_writeRegister(RX_ADDR_P2,adr,1);  // One byte long
111     nrf24_ce_digitalWrite(HIGH);
112 }
113
114
115 /* Returns the payload length */
116 uint8_t nrf24_payload_length()
117 {
118     return payload_len;
119 }
120
121 /* Set the TX address */
122 void nrf24_tx_address(uint8_t* adr)
123 {
124     /* RX_ADDR_P0 must be set to the sending addr for auto ack to work. */
125     nrf24_writeRegister(RX_ADDR_P0,adr,nrf24_ADDR_LEN);
126     nrf24_writeRegister(TX_ADDR,adr,nrf24_ADDR_LEN);
127 }
128
129 /* Checks if data is available for reading */
130 /* Returns 1 if data is ready ... */
131 uint8_t nrf24_dataReady()
132 {
133     // See note in getData() function - just checking RX_DR isn't good enough
134     uint8_t status = nrf24_getStatus();
135
136     // We can short circuit on RX_DR, but if it's not set, we still need
137     // to check the FIFO for any pending packets
138     if ( status & (1 << RX_DR) )
139     {
140         return 1;
141     }
142
143     return !nrf24_rxFifoEmpty();;
144 }
145
146 /* Checks if receive FIFO is empty or not */
147 uint8_t nrf24_rxFifoEmpty()
148 {
```

```
149     uint8_t fifoStatus;
150
151     nrf24_readRegister(FIFO_STATUS,&fifoStatus,1);
152
153     return (fifoStatus & (1 << RX_EMPTY));
154 }
155
156 /* Returns the length of data waiting in the RX fifo */
157 uint8_t nrf24_payloadLength()
158 {
159     uint8_t status;
160     nrf24_csn_digitalWrite(LOW);
161     spi_transfer(R_RX_PL_WID);
162     status = spi_transfer(0x00);
163     nrf24_csn_digitalWrite(HIGH);
164     return status;
165 }
166
167 /* Reads payload bytes into data array */
168 void nrf24_getData(uint8_t* data)
169 {
170     /* Pull down chip select */
171     nrf24_csn_digitalWrite(LOW);
172
173     /* Send cmd to read rx payload */
174     spi_transfer( R_RX_PAYLOAD );
175
176     /* Read payload */
177     nrf24_transferSync(data,data,payload_len);
178
179     /* Pull up chip select */
180     nrf24_csn_digitalWrite(HIGH);
181
182     /* Reset status register */
183     nrf24_configRegister(STATUS,(1<<RX_DR));
184 }
185
186 /* Returns the number of retransmissions occured for the last message */
187 uint8_t nrf24_retransmissionCount()
188 {
189     uint8_t rv;
190     nrf24_readRegister(OBSERVE_TX,&rv,1);
191     rv = rv & 0x0F;
192     return rv;
193 }
194
195 // Sends a data package to the default address. Be sure to send the correct
196 // amount of bytes as configured as payload on the receiver.
197 void nrf24_send(uint8_t* value)
198 {
199     /* Go to Standby-I first */
200     nrf24_ce_digitalWrite(LOW);
```

```
201
202     /* Set to transmitter mode , Power up if needed */
203     nrf24_powerUpTx();
204
205     /* Do we really need to flush TX fifo each time ? */
206     #if 1
207     /* Pull down chip select */
208     nrf24_csn_digitalWrite(LOW);
209
210     /* Write cmd to flush transmit FIFO */
211     spi_transfer(FLUSH_TX);
212
213     /* Pull up chip select */
214     nrf24_csn_digitalWrite(HIGH);
215     #endif
216
217     /* Pull down chip select */
218     nrf24_csn_digitalWrite(LOW);
219
220     /* Write cmd to write payload */
221     spi_transfer(W_TX_PAYLOAD);
222
223     /* Write payload */
224     nrf24_transmitSync(value,payload_len);
225
226     /* Pull up chip select */
227     nrf24_csn_digitalWrite(HIGH);
228
229     /* Start the transmission */
230     nrf24_ce_digitalWrite(HIGH);
231 }
232
233 uint8_t nrf24_isSending()
234 {
235     uint8_t status;
236
237     /* read the current status */
238     status = nrf24_getStatus();
239
240     /* if sending successful (TX_DS) or max retries exceded (MAX_RT). */
241     if((status & ((1 << TX_DS)  | (1 << MAX_RT))))
242     {
243         return 0; /* false */
244     }
245
246     return 1; /* true */
247
248 }
249
250 uint8_t nrf24_getStatus()
251 {
252     uint8_t rv;
```

```
253     nrf24_csn_digitalWrite(LOW);
254     rv = spi_transfer(NOP);
255     nrf24_csn_digitalWrite(HIGH);
256     return rv;
257 }
258 
259 uint8_t nrf24_lastMessageStatus()
260 {
261     uint8_t rv;
262 
263     rv = nrf24_getStatus();
264 
265     /* Transmission went OK */
266     if((rv & ((1 << TX_DS))))
267     {
268         return NRF24_TRANSMISSON_OK;
269     }
270     /* Maximum retransmission count is reached */
271     /* Last message probably went missing ... */
272     else if((rv & ((1 << MAX_RT))))
273     {
274         return NRF24_MESSAGE_LOST;
275     }
276     /* Probably still sending ... */
277     else
278     {
279         return 0xFF;
280     }
281 }
282 
283 void nrf24_powerUpRx()
284 {
285     nrf24_csn_digitalWrite(LOW);
286     spi_transfer(FLUSH_RX);
287     nrf24_csn_digitalWrite(HIGH);
288 
289     nrf24_configRegister(STATUS,(1<<RX_DR)|(1<<TX_DS)|(1<<MAX_RT));
290 
291     nrf24_ce_digitalWrite(LOW);
292     nrf24_configRegister(CONFIG,nrf24_CONFIG|((1<<PWR_UP)|(1<<PRIM_RX)));
293     nrf24_ce_digitalWrite(HIGH);
294 }
295 
296 void nrf24_powerUpTx()
297 {
298     nrf24_configRegister(STATUS,(1<<RX_DR)|(1<<TX_DS)|(1<<MAX_RT));
299 
300     nrf24_configRegister(CONFIG,nrf24_CONFIG|((1<<PWR_UP)|(0<<PRIM_RX)));
301 }
302 
303 void nrf24_powerDown()
304 {
```

```
305     nrf24_ce_digitalWrite(LOW);
306     nrf24_configRegister(CONFIG,nrf24_CONFIG);
307 }
308 
309 uint8_t spi_transfer(uint8_t tx)
310 {
311     uint8_t i = 0;
312     uint8_t rx = 0;
313 
314     nrf24_sck_digitalWrite(LOW);
315 
316     for(i=0;i<8;i++)
317     {
318 
319         if(tx & (1<<(7-i)))
320         {
321             nrf24_mosi_digitalWrite(HIGH);
322         }
323         else
324         {
325             nrf24_mosi_digitalWrite(LOW);
326         }
327 
328         nrf24_sck_digitalWrite(HIGH);
329 
330         rx = rx << 1;
331         if(nrf24_miso_digitalRead())
332         {
333             rx |= 0x01;
334         }
335 
336         nrf24_sck_digitalWrite(LOW);
337 
338     }
339 
340     return rx;
341 }
342 
343 /* send and receive multiple bytes over SPI */
344 void nrf24_transferSync(uint8_t* dataout,uint8_t* datain,uint8_t len)
345 {
346     uint8_t i;
347 
348     for(i=0;i<len;i++)
349     {
350         datain[i] = spi_transfer(dataout[i]);
351     }
352 
353 }
354 
355 /* send multiple bytes over SPI */
356 void nrf24_transmitSync(uint8_t* dataout,uint8_t len)
```

```
357 {
358     uint8_t i;
359 
360     for(i=0;i<len;i++)
361     {
362         spi_transfer(dataout[i]);
363     }
364 
365 }
366 
367 /* Clocks only one byte into the given nrf24 register */
368 void nrf24_configRegister(uint8_t reg, uint8_t value)
369 {
370     nrf24_csn_digitalWrite(LOW);
371     spi_transfer(W_REGISTER | (REGISTER_MASK & reg));
372     spi_transfer(value);
373     nrf24_csn_digitalWrite(HIGH);
374 }
375 
376 /* Read single register from nrf24 */
377 void nrf24_readRegister(uint8_t reg, uint8_t* value, uint8_t len)
378 {
379     nrf24_csn_digitalWrite(LOW);
380     spi_transfer(R_REGISTER | (REGISTER_MASK & reg));
381     nrf24_transferSync(value,value,len);
382     nrf24_csn_digitalWrite(HIGH);
383 }
384 
385 /* Write to a single register of nrf24 */
386 void nrf24_writeRegister(uint8_t reg, uint8_t* value, uint8_t len)
387 {
388     nrf24_csn_digitalWrite(LOW);
389     spi_transfer(W_REGISTER | (REGISTER_MASK & reg));
390     nrf24_transmitSync(value,len);
391     nrf24_csn_digitalWrite(HIGH);
392 }
393 
394 /* Check single register from nrf24 */
395 bool nrf24_checkRegister(uint8_t reg, uint8_t desiredValue, uint8_t len)
396 {
397     uint8_t registerValue;
398     nrf24_readRegister(reg,&registerValue,len);
399     if (registerValue==desiredValue) { return true; } else { return false; }
400 }
401 
402 #define RF_DDR  DDRC
403 #define RF_PORT PORTC
404 #define RF_PIN  PINC
405 
406 #define set_bit(reg,bit) reg |= (1<<bit)
407 #define clr_bit(reg,bit) reg &= ~(1<<bit)
408 #define check_bit(reg,bit) (reg&(1<<bit))
```

```
409
410 /* ---------------------------------------------------------------------- */
411
412 void nrf24_setupPins()
413 {
414     set_bit(RF_DDR,0); // CE output
415     set_bit(RF_DDR,1); // CSN output
416     set_bit(RF_DDR,2); // SCK output
417     set_bit(RF_DDR,3); // MOSI output
418     clr_bit(RF_DDR,4); // MISO input
419 }
420 /* ---------------------------------------------------------------------- */
421 void nrf24_ce_digitalWrite(uint8_t state)
422 {
423     if(state)
424     {
425         set_bit(RF_PORT,0);
426     }
427     else
428     {
429         clr_bit(RF_PORT,0);
430     }
431 }
432 /* ---------------------------------------------------------------------- */
433 void nrf24_csn_digitalWrite(uint8_t state)
434 {
435     if(state)
436     {
437         set_bit(RF_PORT,1);
438     }
439     else
440     {
441         clr_bit(RF_PORT,1);
442     }
443 }
444 /* ---------------------------------------------------------------------- */
445 void nrf24_sck_digitalWrite(uint8_t state)
446 {
447     if(state)
448     {
449         set_bit(RF_PORT,2);
450     }
451     else
452     {
453         clr_bit(RF_PORT,2);
454     }
455 }
456 /* ---------------------------------------------------------------------- */
457 void nrf24_mosi_digitalWrite(uint8_t state)
458 {
459     if(state)
460     {
```

```
461         set_bit(RF_PORT,3);
462     }
463     else
464     {
465         clr_bit(RF_PORT,3);
466     }
467 }
468 /* ------------------------------------------------------------------------ */
469 uint8_t nrf24_miso_digitalRead()
470 {
471     return check_bit(RF_PIN,4);
472 }
473 /* ------------------------------------------------------------------------ */
474
475 void nrf24_initRF_SAFE(uint8_t boardIndex,TransmissionMode initMode){
476
477     initliazeMemory();
478     bool successfulRfInit = false;
479
480     while(successfulRfInit==false){
481         nrf24_powerDown();
482         nrf24_init();
483         nrf24_config(GENERAL_RF_CHANNEL,32);
484         if (nrf24_checkConfig()) { successfulRfInit = true; } else
              { faultyRF_Alarm(); }
485     }
486
487
488
489     if (initMode==RECEIVE){
490         nrf24_tx_address(CURRENT_BOARD_ADDRESS);
491         nrf24_rx_address(BOARD_ADDRESS[boardIndex]);
492         }else{
493         nrf24_tx_address(BOARD_ADDRESS[boardIndex]);
494         nrf24_rx_address(CURRENT_BOARD_ADDRESS);
495     }
496
497
498     nrf24_powerUpRx();
499 }
```

```
#ifndef NRF24
#define NRF24

#ifndef F_CPU
#define F_CPU 16000000UL
#endif

#include "nRF24L01_Definitions.h"
#include "Command_Handler.h"
#include <stdint.h>
#include <stdbool.h>
#include <avr/io.h>
#include <avr/delay.h>



#ifndef BIT_MANIPULATION_MACRO
#define BIT_MANIPULATION_MACRO 1
#define bit_get(p,m) ((p) & (m))
#define bit_set(p,m) ((p) |= (m))
#define bit_clear(p,m) ((p) &= ~(m))
#define bit_flip(p,m) ((p) ^= (m))
#define bit_write(c,p,m) (c ? bit_set(p,m) : bit_clear(p,m))
#define BIT(x) (0x01 << (x))
#define LONGBIT(x) ((unsigned long)0x00000001 << (x))
#endif

#define LOW 0
#define HIGH 1
#define nrf24_ADDR_LEN 5
#define nrf24_CONFIG ((1<<EN_CRC)|(0<<CRCO))
#define NRF24_TRANSMISSON_OK 0
#define NRF24_MESSAGE_LOST   1

#define CLEAR_FAULTY_RF_LED         bit_clear(PORTD, BIT(7))
#define FLIP_FAULTY_RF_LED          bit_flip(PORTD, BIT(7))


enum TransmissionMode {
    RECEIVE,
    TRANSMIT
};
typedef enum TransmissionMode TransmissionMode;

enum CommandsBoard {
    MAIN_BOARD_RF = 0,
    POWER_BOARD_RF = 1,
    MOTORIZED_BOARD_RF = 2
};
typedef enum CommandsBoard CommandsBoard;

extern void nrf24_initRF_SAFE(uint8_t boardIndex,TransmissionMode initMode);
```

```
53
54  void    nrf24_init();
55  void    nrf24_rx_address(uint8_t* adr);
56  void    nrf24_tx_address(uint8_t* adr);
57  void    nrf24_config(uint8_t channel, uint8_t pay_length);
58  bool    nrf24_checkRegister(uint8_t reg, uint8_t desiredValue, uint8_t len);
59  bool    nrf24_checkConfig();
60  bool    nrf24_checkAvailability();
61
62  void faultyRF_Alarm();
63
64  uint8_t selectedTX_ADDRESS;
65  uint8_t selectedRX_ADDRESS;
66
67  uint8_t nrf24_dataReady();
68  uint8_t nrf24_isSending();
69  uint8_t nrf24_getStatus();
70  uint8_t nrf24_rxFifoEmpty();
71
72  void    nrf24_send(uint8_t* value);
73  void    nrf24_getData(uint8_t* data);
74
75  uint8_t nrf24_payloadLength();
76
77  uint8_t nrf24_lastMessageStatus();
78  uint8_t nrf24_retransmissionCount();
79
80  uint8_t nrf24_payload_length();
81
82  void    nrf24_powerUpRx();
83  void    nrf24_powerUpTx();
84  void    nrf24_powerDown();
85
86  uint8_t spi_transfer(uint8_t tx);
87  void    nrf24_transmitSync(uint8_t* dataout,uint8_t len);
88  void    nrf24_transferSync(uint8_t* dataout,uint8_t* datain,uint8_t len);
89  void    nrf24_configRegister(uint8_t reg, uint8_t value);
90  void    nrf24_readRegister(uint8_t reg, uint8_t* value, uint8_t len);
91  void    nrf24_writeRegister(uint8_t reg, uint8_t* value, uint8_t len);
92
93  extern void nrf24_setupPins();
94
95  extern void nrf24_ce_digitalWrite(uint8_t state);
96
97  extern void nrf24_csn_digitalWrite(uint8_t state);
98
99  extern void nrf24_sck_digitalWrite(uint8_t state);
100
101 extern void nrf24_mosi_digitalWrite(uint8_t state);
102
103 extern uint8_t nrf24_miso_digitalRead();
104
```

```
105  #endif
106
```

SAPLM

Código de fuente: Módulo de potencia (Lenguaje: AVR-GCC)

Contenidos:

```c
#ifndef F_CPU
#define F_CPU 16000000UL
#endif
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <stdint.h>

#include "nrf24.h"

void initIO();

int main(void)
{
    //sei();    // Interrupts on
    initIO();
    nrf24_initRF_SAFE(MAIN_BOARD, RECEIVE); // CONNECTION TO MAIN BOARD : GENERAL RF CHANNEL 112

    while (1)
    {
        if(nrf24_dataReady())
        {
            nrf24_getData(command_buffer);
            CommandStatus status = DecomposeMessageFromBuffer();
            if (status==SUCCESFUL_DECOMPOSITION) { HandleAvailableCommand(); } else
            {
                bit_flip(PORTD, BIT(7)); _delay_ms(250); bit_flip(PORTD, BIT(7));
            }
        }

        if (nrf24_checkAvailability()==false) { nrf24_initRF_SAFE(MAIN_BOARD, RECEIVE); }
    }
}


void initIO(){
    /*
        Input/Output pin initialization
        1 : OUTPUT | 0 : INPUT | 0b76543210 Bit order
        ATTACHMENTS
            RELAY 0     : PD3               |   OUTPUT
            RELAY 1     : PD2               |   OUTPUT
            RELAY 2     : PD6               |   OUTPUT
            RELAY 3     : PD5               |   OUTPUT
            RED LED     : PD7               |   OUTPUT
            GREEN LED   : PB0               |   OUTPUT
```

```
50          nRF24L01
51              CE   : PC0                       |   OUTPUT
52              CSN  : PC1                       |   OUTPUT
53              MISO : PD0 (MSPIM MISO ATMEGA)   |   INPUT
54              MOSI : PD1 (MSPIM MOSI ATMEGA)   |   OUTPUT
55              SCK  : PD4 (MSPIM XCK)           |   OUTPUT
56      */
57      DDRD = 0b11111110;
58      DDRB = 0b00101001;
59      DDRC = 0b11011111;
60  }
61
62
63
64
65
66
```

```c

#include "Command_Handler.h"
#include "nrf24.h"
#include "crc.h"



const CommandType commandList[] = {
    { .handlerFunction = &UPDATE_ALL_DEVICES_VALUE_H},
    { .handlerFunction = &UPDATE_DEVICE_VALUE_H},
    { .handlerFunction = &GET_ALL_DEVICES_VALUE_H},
    { .handlerFunction = &GET_DEVICE_VALUE_H},
    { .handlerFunction = &MESSAGE_STATUS_H}
};
#define commandListLength (uint8_t)(sizeof commandList/sizeof commandList[0])

bool initliazeMemory(){
    if(memoryInitialized) return false;
    parameter[0].startingPointer = (void*)calloc(23,1);
    parameter[1].startingPointer = (void*)calloc(2,1);
    parameter[2].startingPointer = (void*)calloc(2,1);
    for (uint8_t x = 3; x<12; x++) parameter[x].startingPointer = (void*)calloc(1,1);
    command_buffer = (uint8_t*)calloc(32,1);
    if(command_buffer==NULL) return false;
    for (uint8_t x = 0; x<12; x++) { if(parameter[x].startingPointer==NULL) return false; }
    memoryInitialized = true;
    return true;
}

CommandStatus DecomposeMessageFromBuffer(){
    // Search for header
    uint8_t* headerStart = command_buffer;
    uint8_t* footerEnd = command_buffer+31;

    for(;headerStart!=(command_buffer+22);headerStart++){
        if (*headerStart==SOH&&(*(headerStart+4)==STX)){
            for(;footerEnd!=(command_buffer+6);footerEnd--){
                if (*footerEnd==ETB&&(*(footerEnd-2)==ETX)){
                    uint8_t netMessageLength = ((footerEnd-2)-headerStart);
                    crc_t crc;
                    crc = crc_init();
                    crc = crc_update(crc, headerStart, netMessageLength);
                    crc = crc_finalize(crc);
                    if (*(footerEnd-1)!=crc) return WRONG_CHECKSUM_CONSISTENCY;
                    if (*(headerStart+2)!=currentModuleID&&*(headerStart+2)!=0xFF&&currentModuleID!=0x01) return WRONG_MODULE_ID;
                    lastTargetModuleID = *(headerStart+2);
                    lastTransmitterModuleID = *(headerStart+3);
                    if (*(headerStart+5)>commandListLength-1) return UNDEFINED_COMMAND_CODE;
```

```
49                      lastMessageCommandType = commandList[*(headerStart+5)];
50                      lastMessagePID = *(headerStart+1);
51
52                      uint8_t* parameterStart = headerStart+6;
53
54                      for (uint8_t x = 0; x < 12; x++) {
55                          realloc(parameter[x].startingPointer, *parameterStart);
56                          parameter[x].byteLength = *parameterStart;
57                          memcpy(parameter[x].startingPointer,parameterStart+1,
                        *parameterStart);
58                          parameterStart+=((*parameterStart)+1);
59                          if (parameterStart>=(footerEnd-2)) break;
60                      }
61
62                      return SUCCESFUL_DECOMPOSITION;
63                  }
64              }
65          }
66      }
67      return WRONG_HEADER_SEGMENTATION;
68  }
69
70  CommandStatus ComposeMessageToBuffer(CommandTypeID targetTypeID, uint8_t
      parameterCount, uint8_t targetBoardID){
71      memset(command_buffer, 0, 32);
72      command_buffer[0] = SOH;
73      if (lastMessagePID==0xFF) { lastMessagePID++; } else { lastMessagePID = 0; }
74      command_buffer[1] = lastMessagePID;
75      command_buffer[2] = targetBoardID;
76      command_buffer[3] = currentModuleID;
77      command_buffer[4] = STX;
78      command_buffer[5] = targetTypeID;
79
80      if (parameterCount>12) return PARAMETER_COUNT_OVERSIZE;
81
82      uint8_t* parameterStart = &command_buffer[6];
83
84      for (uint8_t x = 0; x < parameterCount; x++){
85          *parameterStart = parameter[x].byteLength;
86          memcpy(parameterStart+1, parameter[x].startingPointer, parameter
              [x].byteLength);
87          parameterStart+=(parameter[x].byteLength)+1;
88      }
89
90      crc_t crc;
91      crc = crc_init();
92      uint8_t crc_length = ((parameterStart)-(&command_buffer[0]));
93      crc = crc_update(crc, &command_buffer[0], crc_length);
94      crc = crc_finalize(crc);
95
96      *parameterStart = ETX;
97      *(parameterStart+1) = crc;
```

```
 98      *(parameterStart+2) = ETB;
 99
100      return SUCCESFUL_COMPOSITION;
101  }
102
103  void HandleAvailableCommand(){
104      lastMessageCommandType.handlerFunction();
105  }
106
107  RF_TransmissionStatus RetransmissionToModule(){
108      nrf24_initRF_SAFE(lastTargetModuleID, TRANSMIT);    // CONNECTION TO MODULE:
           GENERAL RF CHANNEL 112
109      nrf24_send(command_buffer);
110      while(nrf24_isSending());
111
112      uint8_t messageStatus = nrf24_lastMessageStatus();
113      if(messageStatus == NRF24_TRANSMISSON_OK) { return
           RF_SUCCESFUL_TRANSMISSION; }
114      else if(messageStatus == NRF24_MESSAGE_LOST) { return
           RF_UNREACHEABLE_MODULE;}
115      return RF_UNREACHEABLE_MODULE;
116  }
117
118  void writeParameterValue(uint8_t parameterIndex, void* parameterData, uint8_t
       parameterByteLength){
119      parameter[parameterIndex].startingPointer = (uint8_t*) realloc(parameter
           [parameterIndex].startingPointer, parameterByteLength);
120      memcpy(parameter[parameterIndex].startingPointer, parameterData,
           parameterByteLength);
121      parameter[parameterIndex].byteLength = parameterByteLength;
122  }
123
124  void UPDATE_ALL_DEVICES_VALUE_H() {
125      for (uint8_t x = 0; x < AVAILABLE_DEVICES;x++)
126      {
127          deviceStoredValue[x] = *((uint8_t*)parameter[x].startingPointer);
128          switch (x) {
129              case 0x00:
130                  bit_write(deviceStoredValue[x], PORTD, BIT(3));
131                  break;
132              case 0x01:
133                  bit_write(deviceStoredValue[x], PORTD, BIT(2));
134                  break;
135              case 0x02:
136                  bit_write(deviceStoredValue[x], PORTD, BIT(6));
137                  break;
138              case 0x03:
139                  bit_write(deviceStoredValue[x], PORTD, BIT(5));
140                  break;
141          }
142      }
143
```

```
144
145
146 }
147
148 void UPDATE_DEVICE_VALUE_H() {
149     uint8_t deviceIndex = *((uint8_t*)parameter[0].startingPointer);
150     uint8_t deviceValue = *((uint8_t*)parameter[1].startingPointer);
151
152     switch (deviceIndex) {
153         case 0:
154             bit_write(deviceValue, PORTD, BIT(3));
155             break;
156         case 1:
157             bit_write(deviceValue, PORTD, BIT(2));
158             break;
159         case 2:
160             bit_write(deviceValue, PORTD, BIT(6));
161             break;
162         case 3:
163             bit_write(deviceValue, PORTD, BIT(5));
164             break;
165     }
166
167     deviceStoredValue[deviceIndex] = deviceValue;
168
169 }
170 void GET_ALL_DEVICES_VALUE_H() {
171     _delay_ms(50);
172
173     for (uint8_t x = 0; x < AVAILABLE_DEVICES;x++)
174     {
175         writeParameterValue(x, &deviceStoredValue[x], 2);
176     }
177
178     ComposeMessageToBuffer(UPDATE_ALL_DEVICES_VALUE_ID, AVAILABLE_DEVICES,
        PHONE_MODULE);  // PHONE_MODULE deberia ser lastTransmitterModuleID
179
180     nrf24_initRF_SAFE(MAIN_BOARD, TRANSMIT);
181     nrf24_send(command_buffer);
182     while(nrf24_isSending());
183     uint8_t messageStatus = nrf24_lastMessageStatus();
184 }
185 void GET_DEVICE_VALUE_H() {
186     _delay_ms(50);
187     uint8_t deviceIndex = *((uint8_t*)parameter[0].startingPointer);
188     writeParameterValue(0, &deviceIndex, 1);
189     writeParameterValue(1, &deviceStoredValue[deviceIndex], 2);
190     ComposeMessageToBuffer(UPDATE_DEVICE_VALUE_ID, 2, PHONE_MODULE);  //
        PHONE_MODULE deberia ser lastTransmitterModuleID
191
192     nrf24_initRF_SAFE(MAIN_BOARD, TRANSMIT);
193     nrf24_send(command_buffer);
```

```
194     while(nrf24_isSending());
195     uint8_t messageStatus = nrf24_lastMessageStatus();
196 }
197
198
199 void MESSAGE_STATUS_H() {}
```

```
 1
 2
 3  #ifndef COMMAND_HANDLER_H_
 4  #define COMMAND_HANDLER_H_
 5
 6  #ifndef nullptr
 7  #define nullptr ((void *)0)
 8  #endif
 9
10  #ifndef F_CPU
11  #define F_CPU               16000000UL
12  #endif
13
14  #include <stdbool.h>
15  #include <stdint.h>
16  #include <stdio.h>
17  #include <string.h>
18  #include <stdlib.h>
19  #include <avr/io.h>
20  #include <util/delay.h>
21
22  #ifndef BIT_MANIPULATION_MACRO
23  #define BIT_MANIPULATION_MACRO 1
24  #define bit_get(p,m) ((p) & (m))
25  #define bit_set(p,m) ((p) |= (m))
26  #define bit_clear(p,m) ((p) &= ~(m))
27  #define bit_flip(p,m) ((p) ^= (m))
28  #define bit_write(c,p,m) (c ? bit_set(p,m) : bit_clear(p,m))
29  #define BIT(x) (0x01 << (x))
30  #define LONGBIT(x) ((unsigned long)0x00000001 << (x))
31  #endif
32
33  typedef struct CommandType {
34      void (*handlerFunction)();
35  } CommandType;
36
37  typedef enum   {
38      SUCCESFUL_DECOMPOSITION,
39      WRONG_HEADER_SEGMENTATION,
40      WRONG_FOOTER_SEGMENTATION,
41      WRONG_CHECKSUM_CONSISTENCY,
42      WRONG_MODULE_ID,
43      UNDEFINED_COMMAND_CODE,
44      PARAMETER_DATA_OVERFLOW,
45      PARAMETER_COUNT_OVERSIZE,
46      RETRANSMISSION_FAILED,
47      SUCCESFUL_RETRANSMISSION,
48      SUCCESFUL_COMPOSITION
49  } CommandStatus;
50
51
52  typedef enum {
```

```
53      RF_SUCCESFUL_TRANSMISSION,
54      RF_UNREACHEABLE_MODULE,
55      RF_ACKNOWLEDGE_FAILED
56  } RF_TransmissionStatus;
57
58  typedef enum   {
59      UPDATE_ALL_DEVICES_VALUE_ID,
60      UPDATE_DEVICE_VALUE_ID,
61      GET_ALL_DEVICES_VALUE_ID,
62      GET_DEVICE_VALUE_ID,
63      MESSAGE_STATUS_ID
64  } CommandTypeID;
65
66  typedef struct {
67      void *startingPointer;
68      uint8_t byteLength;
69  } Parameter;
70
71  typedef enum   {
72      PHONE_MODULE = 0x00,
73      MAIN_MODULE = 0x01,
74      POWER_MODULE = 0x02,
75      MOTOR_MODULE = 0x03,
76  } ModuleInternalCode;
77
78
79  #define currentModuleID POWER_MODULE
80
81
82  #define SOH 0x01
83  #define STX 0x02
84  #define ETX 0x03
85  #define ETB 0x17
86  #define ON_STATE    0xFF
87  #define OFF_STATE   0x00
88
89  #define AVAILABLE_DEVICES 4
90  uint8_t deviceStoredValue[AVAILABLE_DEVICES];
91
92  uint8_t *command_buffer;
93  Parameter parameter[12];
94  bool memoryInitialized;
95
96  uint8_t lastMessagePID;
97  uint8_t lastTargetModuleID;
98  uint8_t lastTransmitterModuleID;
99  CommandType lastMessageCommandType;
100
101 extern bool initliazeMemory();
102 extern void UPDATE_ALL_DEVICES_VALUE_H(), UPDATE_DEVICE_VALUE_H(),
      GET_ALL_DEVICES_VALUE_H(), GET_DEVICE_VALUE_H(), MESSAGE_STATUS_H();
103 extern CommandStatus DecomposeMessageFromBuffer();
```

```
104  extern void HandleAvailableCommand();
105  extern RF_TransmissionStatus RetransmissionToModule();
106  extern CommandStatus ComposeMessageToBuffer(CommandTypeID targetTypeID, uint8_t
       parameterCount, uint8_t targetBoardID);
107  void writeParameterValue(uint8_t parameterIndex, void* parameterData, uint8_t
       parameterByteLength);
108
109  #endif /* COMMAND_HANDLER_H_ */
```

```
1
2  #define UCPHA0 1
3  #define BAUD_RATE 38400UL
4  #define UBRR_VALUE ((F_CPU)/(2UL*BAUD_RATE))-1
5
6  #include "nrf24.h"
7
8  uint8_t payload_len;
9  uint8_t selectedChannel;
10
11 uint8_t MOTORIZED_BOARD_ADDR[5] =   {0xF0,0xF0,0xF0,0xF0,0xC9};
12 uint8_t MAIN_BOARD_ADDR[5] =           {0xA4,0xA4,0xA4,0xA4,0xA4};
13 uint8_t POWER_BOARD_ADDR[5] =       {0xF0,0xF0,0xF0,0xF0,0xF0};
14
15 uint8_t* BOARD_ADDRESS[3] = {&MAIN_BOARD_ADDR[0], &POWER_BOARD_ADDR[0],
     &MOTORIZED_BOARD_ADDR[0]};
16 uint8_t* CURRENT_BOARD_ADDRESS = &POWER_BOARD_ADDR[0];
17
18 uint8_t GENERAL_RF_CHANNEL = 112;
19
20
21
22 void nrf24_init()
23 {
24     nrf24_setupPins();
25     nrf24_ce_digitalWrite(LOW);
26     nrf24_csn_digitalWrite(HIGH);
27 }
28
29 void nrf24_config(uint8_t channel, uint8_t pay_length)
30 {
31     /* Use static payload length ... */
32     payload_len = pay_length;
33     selectedChannel = channel;
34     // Set RF channel
35     nrf24_configRegister(RF_CH,channel);
36     // Set length of incoming payload
37     nrf24_configRegister(RX_PW_P0, 0x00); // Auto-ACK pipe ...
38     nrf24_configRegister(RX_PW_P1, payload_len); // Data payload pipe
39     nrf24_configRegister(RX_PW_P2, 0x00); // Pipe not used
40     nrf24_configRegister(RX_PW_P3, 0x00); // Pipe not used
41     nrf24_configRegister(RX_PW_P4, 0x00); // Pipe not used
42     nrf24_configRegister(RX_PW_P5, 0x00); // Pipe not used
43     // 1 Mbps, TX gain: 0dbm
44     nrf24_configRegister(RF_SETUP, (0<<RF_DR)|((0x03)<<RF_PWR));
45     // CRC enable, 1 byte CRC length
46     nrf24_configRegister(CONFIG,nrf24_CONFIG);
47     // Auto Acknowledgment
48     nrf24_configRegister(EN_AA,(1<<ENAA_P0)|(1<<ENAA_P1)|(0<<ENAA_P2)|
         (0<<ENAA_P3)|(0<<ENAA_P4)|(0<<ENAA_P5));
49     // Enable RX addresses
50     nrf24_configRegister(EN_RXADDR,(1<<ERX_P0)|(1<<ERX_P1)|(0<<ERX_P2)|
```

```
        (0<<ERX_P3)|(0<<ERX_P4)|(0<<ERX_P5));
51      // Auto retransmit delay: 1000 us and Up to 15 retransmit trials
52      nrf24_configRegister(SETUP_RETR,(0x04<<ARD)|(0x0F<<ARC));
53      // Dynamic length configurations: No dynamic length
54      nrf24_configRegister(DYNPD,(0<<DPL_P0)|(0<<DPL_P1)|(0<<DPL_P2)|(0<<DPL_P3)|
          (0<<DPL_P4)|(0<<DPL_P5));
55
56  }
57
58  bool nrf24_checkConfig(){
59      // Check all registers
60      if (nrf24_checkRegister(RF_CH, selectedChannel,1)==false) return false;
61      if (nrf24_checkRegister(RX_PW_P0, 0x00,1)==false) return false;
62      if (nrf24_checkRegister(RX_PW_P1, payload_len,1)==false) return false;
63      if (nrf24_checkRegister(RX_PW_P2, 0x00,1)==false) return false;
64      if (nrf24_checkRegister(RX_PW_P3, 0x00,1)==false) return false;
65      if (nrf24_checkRegister(RX_PW_P4, 0x00,1)==false) return false;
66      if (nrf24_checkRegister(RX_PW_P5, 0x00,1)==false) return false;
67      if (nrf24_checkRegister(RF_SETUP, (0<<RF_DR)|((0x03)<<RF_PWR),1)==false)
          return false;
68      if (nrf24_checkRegister(CONFIG,nrf24_CONFIG,1)==false) return false;
69      if (nrf24_checkRegister(EN_AA,(1<<ENAA_P0)|(1<<ENAA_P1)|(0<<ENAA_P2)|
          (0<<ENAA_P3)|(0<<ENAA_P4)|(0<<ENAA_P5),1)==false) return false;
70      if (nrf24_checkRegister(SETUP_RETR,(0x04<<ARD)|(0x0F<<ARC),1)==false) return
          false;
71      if (nrf24_checkRegister(DYNPD,(0<<DPL_P0)|(0<<DPL_P1)|(0<<DPL_P2)|
          (0<<DPL_P3)|(0<<DPL_P4)|(0<<DPL_P5),1)==false) return false;
72
73      return true;
74  }
75
76  bool nrf24_checkAvailability(){
77      if (nrf24_checkRegister(RF_CH, selectedChannel,1)==true) { return true; }
          else { return false;}
78  }
79
80
81
82
83  void faultyRF_Alarm(){
84      CLEAR_FAULTY_RF_LED;
85      for (uint8_t x = 0; x < 6; x++)
86      {
87          FLIP_FAULTY_RF_LED;
88          _delay_ms(125);
89      }
90      _delay_ms(250);
91  }
92
93
94
95  /* Set the RX address */
```

```
 96  void nrf24_rx_address(uint8_t * adr)
 97  {
 98      nrf24_ce_digitalWrite(LOW);
 99      nrf24_writeRegister(RX_ADDR_P1,adr,nrf24_ADDR_LEN);
100      nrf24_ce_digitalWrite(HIGH);
101  }
102
103  /* Returns the payload length */
104  uint8_t nrf24_payload_length()
105  {
106      return payload_len;
107  }
108
109  /* Set the TX address */
110  void nrf24_tx_address(uint8_t* adr)
111  {
112      /* RX_ADDR_P0 must be set to the sending addr for auto ack to work. */
113      nrf24_writeRegister(RX_ADDR_P0,adr,nrf24_ADDR_LEN);
114      nrf24_writeRegister(TX_ADDR,adr,nrf24_ADDR_LEN);
115  }
116
117  /* Checks if data is available for reading */
118  /* Returns 1 if data is ready ... */
119  uint8_t nrf24_dataReady()
120  {
121      // See note in getData() function - just checking RX_DR isn't good enough
122      uint8_t status = nrf24_getStatus();
123
124      // We can short circuit on RX_DR, but if it's not set, we still need
125      // to check the FIFO for any pending packets
126      if ( status & (1 << RX_DR) )
127      {
128          return 1;
129      }
130
131      return !nrf24_rxFifoEmpty();
132  }
133
134  /* Checks if receive FIFO is empty or not */
135  uint8_t nrf24_rxFifoEmpty()
136  {
137      uint8_t fifoStatus;
138
139      nrf24_readRegister(FIFO_STATUS,&fifoStatus,1);
140
141      return (fifoStatus & (1 << RX_EMPTY));
142  }
143
144  /* Returns the length of data waiting in the RX fifo */
145  uint8_t nrf24_payloadLength()
146  {
147      uint8_t status;
```

```
148     nrf24_csn_digitalWrite(LOW);
149     spi_transfer(R_RX_PL_WID);
150     status = spi_transfer(0x00);
151     nrf24_csn_digitalWrite(HIGH);
152     return status;
153 }
154 
155 /* Reads payload bytes into data array */
156 void nrf24_getData(uint8_t* data)
157 {
158     /* Pull down chip select */
159     nrf24_csn_digitalWrite(LOW);
160 
161     /* Send cmd to read rx payload */
162     spi_transfer( R_RX_PAYLOAD );
163 
164     /* Read payload */
165     nrf24_transferSync(data,data,payload_len);
166 
167     /* Pull up chip select */
168     nrf24_csn_digitalWrite(HIGH);
169 
170     /* Reset status register */
171     nrf24_configRegister(STATUS,(1<<RX_DR));
172 }
173 
174 /* Returns the number of retransmissions occured for the last message */
175 uint8_t nrf24_retransmissionCount()
176 {
177     uint8_t rv;
178     nrf24_readRegister(OBSERVE_TX,&rv,1);
179     rv = rv & 0x0F;
180     return rv;
181 }
182 
183 // Sends a data package to the default address. Be sure to send the correct
184 // amount of bytes as configured as payload on the receiver.
185 void nrf24_send(uint8_t* value)
186 {
187     /* Go to Standby-I first */
188     nrf24_ce_digitalWrite(LOW);
189 
190     /* Set to transmitter mode , Power up if needed */
191     nrf24_powerUpTx();
192 
193     /* Do we really need to flush TX fifo each time ? */
194     #if 1
195         /* Pull down chip select */
196         nrf24_csn_digitalWrite(LOW);
197 
198         /* Write cmd to flush transmit FIFO */
199         spi_transfer(FLUSH_TX);
```

```
200
201         /* Pull up chip select */
202         nrf24_csn_digitalWrite(HIGH);
203     #endif
204
205     /* Pull down chip select */
206     nrf24_csn_digitalWrite(LOW);
207
208     /* Write cmd to write payload */
209     spi_transfer(W_TX_PAYLOAD);
210
211     /* Write payload */
212     nrf24_transmitSync(value,payload_len);
213
214     /* Pull up chip select */
215     nrf24_csn_digitalWrite(HIGH);
216
217     /* Start the transmission */
218     nrf24_ce_digitalWrite(HIGH);
219 }
220
221 uint8_t nrf24_isSending()
222 {
223     uint8_t status;
224
225     /* read the current status */
226     status = nrf24_getStatus();
227
228     /* if sending successful (TX_DS) or max retries exceded (MAX_RT). */
229     if((status & ((1 << TX_DS)  | (1 << MAX_RT))))
230     {
231         return 0; /* false */
232     }
233
234     return 1; /* true */
235 }
236
237 uint8_t nrf24_getStatus()
238 {
239     uint8_t rv;
240     nrf24_csn_digitalWrite(LOW);
241     rv = spi_transfer(NOP);
242     nrf24_csn_digitalWrite(HIGH);
243     return rv;
244 }
245
246 uint8_t nrf24_lastMessageStatus()
247 {
248     uint8_t rv;
249
250     rv = nrf24_getStatus();
251
```

```
252     /* Transmission went OK */
253     if((rv & ((1 << TX_DS))))
254     {
255         return NRF24_TRANSMISSON_OK;
256     }
257     /* Maximum retransmission count is reached */
258     /* Last message probably went missing ... */
259     else if((rv & ((1 << MAX_RT))))
260     {
261         return NRF24_MESSAGE_LOST;
262     }
263     /* Probably still sending ... */
264     else
265     {
266         return 0xFF;
267     }
268 }
269
270 void nrf24_powerUpRx()
271 {
272     nrf24_csn_digitalWrite(LOW);
273     spi_transfer(FLUSH_RX);
274     nrf24_csn_digitalWrite(HIGH);
275
276     nrf24_configRegister(STATUS,(1<<RX_DR)|(1<<TX_DS)|(1<<MAX_RT));
277
278     nrf24_ce_digitalWrite(LOW);
279     nrf24_configRegister(CONFIG,nrf24_CONFIG|((1<<PWR_UP)|(1<<PRIM_RX)));
280     nrf24_ce_digitalWrite(HIGH);
281 }
282
283 void nrf24_powerUpTx()
284 {
285     nrf24_configRegister(STATUS,(1<<RX_DR)|(1<<TX_DS)|(1<<MAX_RT));
286
287     nrf24_configRegister(CONFIG,nrf24_CONFIG|((1<<PWR_UP)|(0<<PRIM_RX)));
288 }
289
290 void nrf24_powerDown()
291 {
292     nrf24_ce_digitalWrite(LOW);
293     nrf24_configRegister(CONFIG,nrf24_CONFIG);
294 }
295
296 uint8_t spi_transfer(uint8_t tx)
297 {
298     uint8_t i = 0;
299     uint8_t rx = 0;
300
301     nrf24_sck_digitalWrite(LOW);
302
303     for(i=0;i<8;i++)
```

```
304     {
305 
306         if(tx & (1<<(7-i)))
307         {
308             nrf24_mosi_digitalWrite(HIGH);
309         }
310         else
311         {
312             nrf24_mosi_digitalWrite(LOW);
313         }
314 
315         nrf24_sck_digitalWrite(HIGH);
316 
317         rx = rx << 1;
318         if(nrf24_miso_digitalRead())
319         {
320             rx |= 0x01;
321         }
322 
323         nrf24_sck_digitalWrite(LOW);
324 
325     }
326 
327     return rx;
328 }
329 
330 /* send and receive multiple bytes over SPI */
331 void nrf24_transferSync(uint8_t* dataout,uint8_t* datain,uint8_t len)
332 {
333     uint8_t i;
334 
335     for(i=0;i<len;i++)
336     {
337         datain[i] = spi_transfer(dataout[i]);
338     }
339 
340 }
341 
342 /* send multiple bytes over SPI */
343 void nrf24_transmitSync(uint8_t* dataout,uint8_t len)
344 {
345     uint8_t i;
346 
347     for(i=0;i<len;i++)
348     {
349         spi_transfer(dataout[i]);
350     }
351 
352 }
353 
354 /* Clocks only one byte into the given nrf24 register */
355 void nrf24_configRegister(uint8_t reg, uint8_t value)
```

```
356 {
357     nrf24_csn_digitalWrite(LOW);
358     spi_transfer(W_REGISTER | (REGISTER_MASK & reg));
359     spi_transfer(value);
360     nrf24_csn_digitalWrite(HIGH);
361 }
362
363 /* Read single register from nrf24 */
364 void nrf24_readRegister(uint8_t reg, uint8_t* value, uint8_t len)
365 {
366     nrf24_csn_digitalWrite(LOW);
367     spi_transfer(R_REGISTER | (REGISTER_MASK & reg));
368     nrf24_transferSync(value,value,len);
369     nrf24_csn_digitalWrite(HIGH);
370 }
371
372 /* Write to a single register of nrf24 */
373 void nrf24_writeRegister(uint8_t reg, uint8_t* value, uint8_t len)
374 {
375     nrf24_csn_digitalWrite(LOW);
376     spi_transfer(W_REGISTER | (REGISTER_MASK & reg));
377     nrf24_transmitSync(value,len);
378     nrf24_csn_digitalWrite(HIGH);
379 }
380
381 /* Check single register from nrf24 */
382 bool nrf24_checkRegister(uint8_t reg, uint8_t desiredValue, uint8_t len)
383 {
384     uint8_t registerValue;
385     nrf24_readRegister(reg,&registerValue,len);
386     if (registerValue==desiredValue) { return true; } else { return false; }
387 }
388
389 #define RF_DDR  DDRD
390 #define RF_PORT PORTD
391 #define RF_PIN  PIND
392
393 #define CE_CSN_DDR  DDRC
394 #define CE_CSN_PORT PORTC
395 #define CE_CSN_PIN  PINC
396
397 #define MISO_BIT_POS    0
398 #define MOSI_BIT_POS    1
399 #define SCK_BIT_POS     4
400
401 #define CE_BIT_POS      0
402 #define CSN_BIT_POS     1
403
404 #define set_bit(reg,bit) reg |= (1<<bit)
405 #define clr_bit(reg,bit) reg &= ~(1<<bit)
406 #define check_bit(reg,bit) (reg&(1<<bit))
407
```

```
408 /* ------------------------------------------------------------------------ */
409
410 void nrf24_setupPins()
411 {
412     set_bit(CE_CSN_DDR, CE_BIT_POS); // CE output
413     set_bit(CE_CSN_DDR, CSN_BIT_POS); // CSN output
414
415     clr_bit(RF_DDR, MISO_BIT_POS); // MISO input
416     set_bit(RF_DDR, MOSI_BIT_POS); // MOSI output
417     set_bit(RF_DDR, SCK_BIT_POS); // SCK output
418 }
419 /* ------------------------------------------------------------------------ */
420 void nrf24_ce_digitalWrite(uint8_t state)
421 {
422     if(state)
423     {
424         set_bit(CE_CSN_PORT, CE_BIT_POS);
425     }
426     else
427     {
428         clr_bit(CE_CSN_PORT, CE_BIT_POS);
429     }
430 }
431 /* ------------------------------------------------------------------------ */
432 void nrf24_csn_digitalWrite(uint8_t state)
433 {
434     if(state)
435     {
436         set_bit(CE_CSN_PORT, CSN_BIT_POS);
437     }
438     else
439     {
440         clr_bit(CE_CSN_PORT, CSN_BIT_POS);
441     }
442 }
443 /* ------------------------------------------------------------------------ */
444 void nrf24_sck_digitalWrite(uint8_t state)
445 {
446     if(state)
447     {
448         set_bit(RF_PORT, SCK_BIT_POS);
449     }
450     else
451     {
452         clr_bit(RF_PORT, SCK_BIT_POS);
453     }
454 }
455 /* ------------------------------------------------------------------------ */
456 void nrf24_mosi_digitalWrite(uint8_t state)
457 {
458     if(state)
459     {
```

```
460         set_bit(RF_PORT, MOSI_BIT_POS);
461     }
462     else
463     {
464         clr_bit(RF_PORT, MOSI_BIT_POS);
465     }
466 }
467 /* ------------------------------------------------------------------------ */
468 uint8_t nrf24_miso_digitalRead()
469 {
470     return check_bit(RF_PIN, MISO_BIT_POS);
471 }
472 /* ------------------------------------------------------------------------ */
473
474
475 void nrf24_initRF_SAFE(uint8_t boardIndex,TransmissionMode initMode){
476
477     initliazeMemory();
478     bool successfulRfInit = false;
479
480     while(successfulRfInit==false){
481         nrf24_powerDown();
482         nrf24_init();
483         nrf24_config(GENERAL_RF_CHANNEL,32);
484         if (nrf24_checkConfig()) { successfulRfInit = true; } else
              { faultyRF_Alarm(); }
485     }
486
487     if (initMode==TRANSMIT){
488         nrf24_tx_address(CURRENT_BOARD_ADDRESS);
489         nrf24_rx_address(BOARD_ADDRESS[boardIndex]);
490         }else{
491         nrf24_tx_address(BOARD_ADDRESS[boardIndex]);
492         nrf24_rx_address(CURRENT_BOARD_ADDRESS);
493     }
494     nrf24_powerUpRx();
495 }
```

```
 1  #ifndef NRF24
 2  #define NRF24
 3
 4  #ifndef F_CPU
 5  #define F_CPU 16000000UL
 6  #endif
 7
 8  #include "nRF24L01_Definitions.h"
 9  #include "Command_Handler.h"
10  #include <stdint.h>
11  #include <stdbool.h>
12  #include <avr/io.h>
13  #include <avr/delay.h>
14
15
16
17  #ifndef BIT_MANIPULATION_MACRO
18  #define BIT_MANIPULATION_MACRO 1
19  #define bit_get(p,m) ((p) & (m))
20  #define bit_set(p,m) ((p) |= (m))
21  #define bit_clear(p,m) ((p) &= ~(m))
22  #define bit_flip(p,m) ((p) ^= (m))
23  #define bit_write(c,p,m) (c ? bit_set(p,m) : bit_clear(p,m))
24  #define BIT(x) (0x01 << (x))
25  #define LONGBIT(x) ((unsigned long)0x00000001 << (x))
26  #endif
27
28  #define LOW 0
29  #define HIGH 1
30  #define nrf24_ADDR_LEN 5
31  #define nrf24_CONFIG ((1<<EN_CRC)|(0<<CRCO))
32  #define NRF24_TRANSMISSON_OK 0
33  #define NRF24_MESSAGE_LOST   1
34
35  #define CLEAR_FAULTY_RF_LED         bit_clear(PORTD, BIT(7))
36  #define FLIP_FAULTY_RF_LED          bit_flip(PORTD, BIT(7))
37
38
39  enum TransmissionMode {
40      RECEIVE,
41      TRANSMIT
42  };
43  typedef enum TransmissionMode TransmissionMode;
44
45  enum CommandsBoard {
46      MAIN_BOARD = 0,
47      POWER_BOARD = 1,
48      MOTORIZED_BOARD = 2
49  };
50  typedef enum CommandsBoard CommandsBoard;
51
52  extern void nrf24_initRF_SAFE(uint8_t boardIndex,TransmissionMode initMode);
```

```
53
54  void    nrf24_init();
55  void    nrf24_rx_address(uint8_t* adr);
56  void    nrf24_tx_address(uint8_t* adr);
57  void    nrf24_config(uint8_t channel, uint8_t pay_length);
58  bool    nrf24_checkRegister(uint8_t reg, uint8_t desiredValue, uint8_t len);
59  bool    nrf24_checkConfig();
60  bool    nrf24_checkAvailability();
61
62  void faultyRF_Alarm();
63
64
65
66  uint8_t nrf24_dataReady();
67  uint8_t nrf24_isSending();
68  uint8_t nrf24_getStatus();
69  uint8_t nrf24_rxFifoEmpty();
70
71  void    nrf24_send(uint8_t* value);
72  void    nrf24_getData(uint8_t* data);
73
74  uint8_t nrf24_payloadLength();
75
76  uint8_t nrf24_lastMessageStatus();
77  uint8_t nrf24_retransmissionCount();
78
79  uint8_t nrf24_payload_length();
80
81  void    nrf24_powerUpRx();
82  void    nrf24_powerUpTx();
83  void    nrf24_powerDown();
84
85  uint8_t spi_transfer(uint8_t tx);
86  void    nrf24_transmitSync(uint8_t* dataout,uint8_t len);
87  void    nrf24_transferSync(uint8_t* dataout,uint8_t* datain,uint8_t len);
88  void    nrf24_configRegister(uint8_t reg, uint8_t value);
89  void    nrf24_readRegister(uint8_t reg, uint8_t* value, uint8_t len);
90  void    nrf24_writeRegister(uint8_t reg, uint8_t* value, uint8_t len);
91
92  extern void nrf24_setupPins();
93
94  extern void nrf24_ce_digitalWrite(uint8_t state);
95
96  extern void nrf24_csn_digitalWrite(uint8_t state);
97
98  extern void nrf24_sck_digitalWrite(uint8_t state);
99
100 extern void nrf24_mosi_digitalWrite(uint8_t state);
101
102 extern uint8_t nrf24_miso_digitalRead();
103
104 #endif
```

```
 1
 2  /* Memory Map */
 3  #define CONFIG      0x00
 4  #define EN_AA       0x01
 5  #define EN_RXADDR   0x02
 6  #define SETUP_AW    0x03
 7  #define SETUP_RETR  0x04
 8  #define RF_CH       0x05
 9  #define RF_SETUP    0x06
10  #define STATUS      0x07
11  #define OBSERVE_TX  0x08
12  #define CD          0x09
13  #define RX_ADDR_P0  0x0A
14  #define RX_ADDR_P1  0x0B
15  #define RX_ADDR_P2  0x0C
16  #define RX_ADDR_P3  0x0D
17  #define RX_ADDR_P4  0x0E
18  #define RX_ADDR_P5  0x0F
19  #define TX_ADDR     0x10
20  #define RX_PW_P0    0x11
21  #define RX_PW_P1    0x12
22  #define RX_PW_P2    0x13
23  #define RX_PW_P3    0x14
24  #define RX_PW_P4    0x15
25  #define RX_PW_P5    0x16
26  #define FIFO_STATUS 0x17
27  #define DYNPD       0x1C
28
29  /* Bit Mnemonics */
30
31  /* configuration register */
32  #define MASK_RX_DR  6
33  #define MASK_TX_DS  5
34  #define MASK_MAX_RT 4
35  #define EN_CRC      3
36  #define CRCO        2
37  #define PWR_UP      1
38  #define PRIM_RX     0
39
40  /* enable auto acknowledgment */
41  #define ENAA_P5     5
42  #define ENAA_P4     4
43  #define ENAA_P3     3
44  #define ENAA_P2     2
45  #define ENAA_P1     1
46  #define ENAA_P0     0
47
48  /* enable rx addresses */
49  #define ERX_P5      5
50  #define ERX_P4      4
51  #define ERX_P3      3
52  #define ERX_P2      2
```

```
53  #define ERX_P1      1
54  #define ERX_P0      0
55
56  /* setup of address width */
57  #define AW          0 /* 2 bits */
58
59  /* setup of auto re-transmission */
60  #define ARD         4 /* 4 bits */
61  #define ARC         0 /* 4 bits */
62
63  /* RF setup register */
64  #define PLL_LOCK    4
65  #define RF_DR       3
66  #define RF_PWR      1 /* 2 bits */
67
68  /* general status register */
69  #define RX_DR       6
70  #define TX_DS       5
71  #define MAX_RT      4
72  #define RX_P_NO     1 /* 3 bits */
73  #define TX_FULL     0
74
75  /* transmit observe register */
76  #define PLOS_CNT    4 /* 4 bits */
77  #define ARC_CNT     0 /* 4 bits */
78
79  /* fifo status */
80  #define TX_REUSE    6
81  #define FIFO_FULL   5
82  #define TX_EMPTY    4
83  #define RX_FULL     1
84  #define RX_EMPTY    0
85
86  /* dynamic length */
87  #define DPL_P0      0
88  #define DPL_P1      1
89  #define DPL_P2      2
90  #define DPL_P3      3
91  #define DPL_P4      4
92  #define DPL_P5      5
93
94  /* Instruction Mnemonics */
95  #define R_REGISTER    0x00 /* last 4 bits will indicate reg. address */
96  #define W_REGISTER    0x20 /* last 4 bits will indicate reg. address */
97  #define REGISTER_MASK 0x1F
98  #define R_RX_PAYLOAD  0x61
99  #define W_TX_PAYLOAD  0xA0
100 #define FLUSH_TX      0xE1
101 #define FLUSH_RX      0xE2
102 #define REUSE_TX_PL   0xE3
103 #define ACTIVATE      0x50
104 #define R_RX_PL_WID   0x60
```

```
105  #define NOP           0xFF
106
```

```c
/**
 * \file
 * Functions and types for CRC checks.
 *
 * Generated on Wed Sep 11 13:55:53 2019
 * by pycrc v0.9.2, https://pycrc.org
 * using the configuration:
 *  - Width         = 8
 *  - Poly          = 0x07
 *  - XorIn         = 0x00
 *  - ReflectIn     = False
 *  - XorOut        = 0x00
 *  - ReflectOut    = False
 *  - Algorithm     = bit-by-bit-fast
 */
#include "crc.h"     /* include the header file generated with pycrc */
#include <stdlib.h>
#include <stdint.h>
#include <stdbool.h>



crc_t crc_update(crc_t crc, const void *data, size_t data_len)
{
    const unsigned char *d = (const unsigned char *)data;
    unsigned int i;
    bool bit;
    unsigned char c;

    while (data_len--) {
        c = *d++;
        for (i = 0x80; i > 0; i >>= 1) {
            bit = crc & 0x80;
            if (c & i) {
                bit = !bit;
            }
            crc <<= 1;
            if (bit) {
                crc ^= 0x07;
            }
        }
        crc &= 0xff;
    }
    return crc & 0xff;
}

```

```c
/**
 * \file
 * Functions and types for CRC checks.
 *
 * Generated on Wed Sep 11 13:56:48 2019
 * by pycrc v0.9.2, https://pycrc.org
 * using the configuration:
 *  - Width         = 8
 *  - Poly          = 0x07
 *  - XorIn         = 0x00
 *  - ReflectIn     = False
 *  - XorOut        = 0x00
 *  - ReflectOut    = False
 *  - Algorithm     = bit-by-bit-fast
 *
 * This file defines the functions crc_init(), crc_update() and crc_finalize().
 *
 * The crc_init() function returns the inital \c crc value and must be called
 * before the first call to crc_update().
 * Similarly, the crc_finalize() function must be called after the last call
 * to crc_update(), before the \c crc is being used.
 * is being used.
 *
 * The crc_update() function can be called any number of times (including zero
 * times) in between the crc_init() and crc_finalize() calls.
 *
 * This pseudo-code shows an example usage of the API:
 * \code{.c}
 * crc_t crc;
 * unsigned char data[MAX_DATA_LEN];
 * size_t data_len;
 *
 * crc = crc_init();
 * while ((data_len = read_data(data, MAX_DATA_LEN)) > 0) {
 *     crc = crc_update(crc, data, data_len);
 * }
 * crc = crc_finalize(crc);
 * \endcode
 */
#ifndef CRC_H
#define CRC_H

#include <stdlib.h>
#include <stdint.h>

#ifdef __cplusplus
extern "C" {
#endif


/**
 * The definition of the used algorithm.
```

```
 53   *
 54   * This is not used anywhere in the generated code, but it may be used by the
 55   * application code to call algorithm-specific code, if desired.
 56   */
 57  #define CRC_ALGO_BIT_BY_BIT_FAST 1
 58
 59
 60  /**
 61   * The type of the CRC values.
 62   *
 63   * This type must be big enough to contain at least 8 bits.
 64   */
 65  typedef uint_fast8_t crc_t;
 66
 67
 68  /**
 69   * Calculate the initial crc value.
 70   *
 71   * \return     The initial crc value.
 72   */
 73  static inline crc_t crc_init(void)
 74  {
 75      return 0x00;
 76  }
 77
 78
 79  /**
 80   * Update the crc value with new data.
 81   *
 82   * \param[in] crc      The current crc value.
 83   * \param[in] data     Pointer to a buffer of \a data_len bytes.
 84   * \param[in] data_len Number of bytes in the \a data buffer.
 85   * \return             The updated crc value.
 86   */
 87  crc_t crc_update(crc_t crc, const void *data, size_t data_len);
 88
 89
 90  /**
 91   * Calculate the final crc value.
 92   *
 93   * \param[in] crc  The current crc value.
 94   * \return     The final crc value.
 95   */
 96  static inline crc_t crc_finalize(crc_t crc)
 97  {
 98      return crc;
 99  }
100
101
102  #ifdef __cplusplus
103  }           /* closing brace for extern "C" */
104  #endif
```

```
105
106 #endif      /* CRC_H */
107
```

SAPLM

Código de fuente: Módulo motriz (Lenguaje: AVR-GCC)

Contenidos:

```c
#ifndef F_CPU
#define F_CPU 16000000UL
#endif
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <stdint.h>

#include "nrf24.h"

void initIO();

int main(void)
{
    initIO();
    nrf24_initRF_SAFE(MAIN_BOARD, RECEIVE); // CONNECTION TO MAIN BOARD : GENERAL RF CHANNEL 112

    while (1)
    {
        if(nrf24_dataReady())
        {

            nrf24_getData(command_buffer);
            CommandStatus status = DecomposeMessageFromBuffer();
            if (status==SUCCESFUL_DECOMPOSITION) { HandleAvailableCommand(); }
        }

        if (nrf24_checkAvailability()==false) { nrf24_initRF_SAFE(MAIN_BOARD, RECEIVE); }
    }
}


void initIO(){
    /*
        Input/Output pin initialization
        1 : OUTPUT | 0 : INPUT | 0b76543210 Bit order
        ATTACHMENTS
            NURSE SIGN  : PB0               |   OUTPUT
            GREEN LED   : PB1               |   OUTPUT      (SWAPPED IN PCB)
            RED LED     : PB2               |   OUTPUT
        STEP MOTOR A (CURTAIN)
            TERMINAL NO.1 : PD0             |   OUTPUT
            TERMINAL NO.2 : PD1             |   OUTPUT
            TERMINAL NO.3 : PD2             |   OUTPUT
            TERMINAL NO.4 : PD3             |   OUTPUT
        STEP MOTOR B (STRETCHER)
            TERMINAL NO.1 : PD4             |   OUTPUT
```

```
51              TERMINAL NO.2 : PD5             |   OUTPUT
52              TERMINAL NO.3 : PD6             |   OUTPUT
53              TERMINAL NO.4 : PD7             |   OUTPUT
54          nRF24L01
55              CE   : PC0                      |   OUTPUT
56              CSN  : PC1                      |   OUTPUT
57              MISO : PD0 (MSPIM MISO ATMEGA)  |   INPUT
58              MOSI : PD1 (MSPIM MOSI ATMEGA)  |   OUTPUT
59              SCK  : PD4 (MSPIM XCK)          |   OUTPUT
60      */
61      DDRD = 0b11111111;
62      DDRB = 0b00101111;
63      DDRC = 0b11011111;
64  }
65
66
67
68
69
70
```

```c

#include "Command_Handler.h"
#include "nrf24.h"
#include "crc.h"



const CommandType commandList[] = {
    { .handlerFunction = &UPDATE_ALL_DEVICES_VALUE_H},
    { .handlerFunction = &UPDATE_DEVICE_VALUE_H},
    { .handlerFunction = &GET_ALL_DEVICES_VALUE_H},
    { .handlerFunction = &GET_DEVICE_VALUE_H},
    { .handlerFunction = &MESSAGE_STATUS_H}
};
#define commandListLength (uint8_t)(sizeof commandList/sizeof commandList[0])

bool initliazeMemory(){
    if(memoryInitialized) return false;
    parameter[0].startingPointer = (void*)calloc(23,1);
    parameter[1].startingPointer = (void*)calloc(2,1);
    parameter[2].startingPointer = (void*)calloc(2,1);
    for (uint8_t x = 3; x<12; x++) parameter[x].startingPointer = (void*)calloc(1,1);
    command_buffer = (uint8_t*)calloc(32,1);
    if(command_buffer==NULL) return false;
    for (uint8_t x = 0; x<12; x++) { if(parameter[x].startingPointer==NULL) return false; }
    memoryInitialized = true;
    return true;
}

CommandStatus DecomposeMessageFromBuffer(){
    // Search for header
    uint8_t* headerStart = command_buffer;
    uint8_t* footerEnd = command_buffer+31;

    for(;headerStart!=(command_buffer+22);headerStart++){
        if (*headerStart==SOH&&(*(headerStart+4)==STX)){
            for(;footerEnd!=(command_buffer+6);footerEnd--){
                if (*footerEnd==ETB&&(*(footerEnd-2)==ETX)){
                    uint8_t netMessageLength = ((footerEnd-2)-headerStart);
                    crc_t crc;
                    crc = crc_init();
                    crc = crc_update(crc, headerStart, netMessageLength);
                    crc = crc_finalize(crc);
                    if (*(footerEnd-1)!=crc) return WRONG_CHECKSUM_CONSISTENCY;
                    if (*(headerStart+2)!=currentModuleID&&*(headerStart+2)!=0xFF&&currentModuleID!=0x01) return WRONG_MODULE_ID;
                    lastTargetModuleID = *(headerStart+2);
                    lastTransmitterModuleID = *(headerStart+3);
                    if (*(headerStart+5)>commandListLength-1) return UNDEFINED_COMMAND_CODE;
```

```
49                      lastMessageCommandType = commandList[*(headerStart+5)];
50                      lastMessagePID = *(headerStart+1);
51
52                      uint8_t* parameterStart = headerStart+6;
53
54                      for (uint8_t x = 0; x < 12; x++) {
55                          realloc(parameter[x].startingPointer, *parameterStart);
56                          parameter[x].byteLength = *parameterStart;
57                          memcpy(parameter[x].startingPointer,parameterStart+1,
                        *parameterStart);
58                          parameterStart+=((*parameterStart)+1);
59                          if (parameterStart>=(footerEnd-2)) break;
60                      }
61
62                      return SUCCESFUL_DECOMPOSITION;
63                  }
64              }
65          }
66      }
67      return WRONG_HEADER_SEGMENTATION;
68  }
69
70  void HandleAvailableCommand(){
71      lastMessageCommandType.handlerFunction();
72  }
73
74  CommandStatus ComposeMessageToBuffer(CommandTypeID targetTypeID, uint8_t
      parameterCount, uint8_t targetBoardID){
75      memset(command_buffer, 0, 32);
76      command_buffer[0] = SOH;
77      if (lastMessagePID==0xFF) { lastMessagePID++; } else { lastMessagePID = 0; }
78      command_buffer[1] = lastMessagePID;
79      command_buffer[2] = targetBoardID;
80      command_buffer[3] = currentModuleID;
81      command_buffer[4] = STX;
82      command_buffer[5] = targetTypeID;
83
84      if (parameterCount>12) return PARAMETER_COUNT_OVERSIZE;
85
86      uint8_t* parameterStart = &command_buffer[6];
87
88      for (uint8_t x = 0; x < parameterCount; x++){
89          *parameterStart = parameter[x].byteLength;
90          memcpy(parameterStart+1, parameter[x].startingPointer, parameter
              [x].byteLength);
91          parameterStart+=(parameter[x].byteLength)+1;
92      }
93
94      crc_t crc;
95      crc = crc_init();
96      uint8_t crc_length = ((parameterStart)-(&command_buffer[0]));
97      crc = crc_update(crc, &command_buffer[0], crc_length);
```

```
 98      crc = crc_finalize(crc);
 99
100      *parameterStart = ETX;
101      *(parameterStart+1) = crc;
102      *(parameterStart+2) = ETB;
103
104      return SUCCESFUL_COMPOSITION;
105  }
106
107  void writeParameterValue(uint8_t parameterIndex, void* parameterData, uint8_t
       parameterByteLength){
108      parameter[parameterIndex].startingPointer = (uint8_t*) realloc(parameter
           [parameterIndex].startingPointer, parameterByteLength);
109      memcpy(parameter[parameterIndex].startingPointer, parameterData,
           parameterByteLength);
110      parameter[parameterIndex].byteLength = parameterByteLength;
111  }
112
113  void UPDATE_ALL_DEVICES_VALUE_H() {
114      for (uint8_t x = 0; x < AVAILABLE_DEVICES;x++)
115      {
116          deviceStoredValue[x] = *((uint8_t*)parameter[x].startingPointer);
117
118          switch (x) {
119              case 0:
120                  STRETCHER_POS_CHANGE_HANDLE(deviceStoredValue[x]);
121              break;
122              case 1:
123                  CURTAIN_POS_CHANGE_HANDLE(deviceStoredValue[x]);
124              break;
125              case 2:
126                  if (deviceStoredValue[x]==0xFF){
127                      for (uint8_t x = 0; x < 6; x++)
128                      {
129                          bit_flip(PORTB, BIT(0));
130                          bit_flip(PORTB, BIT(1));
131                          bit_flip(PORTB, BIT(2));
132                          _delay_ms(200);
133                      }
134                      bit_clear(PORTB, BIT(0));
135                      bit_clear(PORTB, BIT(1));
136                      bit_clear(PORTB, BIT(2));
137                  }
138              break;
139          }
140      }
141
142  }
143
144  #define MOTOR_DELAY_MS 1
145  #define CURTAIN_CALIBRATION_CONSTANT 200
146  #define STRETCHER_CALIBRATION_CONSTANT 50
```

```
147
148 void UPDATE_DEVICE_VALUE_H() {
149     const uint8_t deviceIndex = *((uint8_t*)parameter[0].startingPointer);
150     const uint8_t deviceValue = *((uint8_t*)parameter[1].startingPointer);
151
152     switch (deviceIndex) {
153         case 0:
154             STRETCHER_POS_CHANGE_HANDLE(deviceValue);
155         break;
156         case 1:
157             CURTAIN_POS_CHANGE_HANDLE(deviceValue);
158         break;
159         case 2:
160             for (uint8_t x = 0; x < 6; x++)
161             {
162                 bit_flip(PORTB, BIT(0));
163                 bit_flip(PORTB, BIT(1));
164                 bit_flip(PORTB, BIT(2));
165                 _delay_ms(200);
166             }
167             bit_clear(PORTB, BIT(0));
168             bit_clear(PORTB, BIT(1));
169             bit_clear(PORTB, BIT(2));
170         break;
171     }
172
173     deviceStoredValue[deviceIndex] = deviceValue;
174
175 }
176
177 void GET_ALL_DEVICES_VALUE_H() {}
178
179 void GET_DEVICE_VALUE_H() {
180     _delay_ms(100);
181     uint8_t deviceIndex = *((uint8_t*)parameter[0].startingPointer);
182     writeParameterValue(0, &deviceIndex, 1);
183     writeParameterValue(1, &deviceStoredValue[deviceIndex], 2);
184     ComposeMessageToBuffer(UPDATE_DEVICE_VALUE_ID, 2, 0x7C);
185
186     nrf24_initRF_SAFE(MAIN_BOARD, TRANSMIT);
187     nrf24_send(command_buffer);
188     while(nrf24_isSending());
189     uint8_t messageStatus = nrf24_lastMessageStatus();
190 }
191 void MESSAGE_STATUS_H() {}
192
193
194 uint8_t previousCurtainPosition = 0;
195 uint8_t previousStretcherPosition = 0;
196
197
198 void CURTAIN_POS_CHANGE_HANDLE(uint8_t positionToMove){
```

```
199     bit_set(PORTB, BIT(1));
200     bit_set(PORTB, BIT(2));
201
202
203     if (positionToMove<8) {
204         uint16_t degreesToMove = abs(positionToMove-previousCurtainPosition)
              *CURTAIN_CALIBRATION_CONSTANT;
205
206         if((positionToMove-previousCurtainPosition)>0){
207             for (uint16_t x = 0; x < degreesToMove;x++){
208                 PORTD = 0b00000011;
209                 _delay_ms(MOTOR_DELAY_MS);
210                 PORTD = 0b00000110;
211                 _delay_ms(MOTOR_DELAY_MS);
212                 PORTD = 0b00001100;
213                 _delay_ms(MOTOR_DELAY_MS);
214                 PORTD = 0b00001001;
215                 _delay_ms(MOTOR_DELAY_MS);
216             }
217             }else{
218             for (uint16_t x = 0; x < degreesToMove;x++){
219                 PORTD = 0b00001100;
220                 _delay_ms(MOTOR_DELAY_MS);
221                 PORTD = 0b00000110;
222                 _delay_ms(MOTOR_DELAY_MS);
223                 PORTD = 0b00000011;
224                 _delay_ms(MOTOR_DELAY_MS);
225                 PORTD = 0b00001001;
226                 _delay_ms(MOTOR_DELAY_MS);
227             }
228         }
229
230         PORTD = 0b00000000;
231         previousCurtainPosition = positionToMove;
232     }
233     bit_clear(PORTB, BIT(1));
234     bit_clear(PORTB, BIT(2));
235 }
236
237 void STRETCHER_POS_CHANGE_HANDLE(uint8_t positionToMove){
238     bit_set(PORTB, BIT(1));
239     bit_set(PORTB, BIT(2));
240
241     if (positionToMove<4) {
242         uint16_t degreesToMove = abs(positionToMove-previousStretcherPosition)
              *STRETCHER_CALIBRATION_CONSTANT;
243
244         if((positionToMove-previousCurtainPosition)>0){
245             for (uint16_t x = 0; x < degreesToMove;x++){
246                 PORTD = 0b00110000;
247                 _delay_ms(MOTOR_DELAY_MS);
248                 PORTD = 0b01100000;
```

```
249                 _delay_ms(MOTOR_DELAY_MS);
250                 PORTD = 0b11000000;
251                 _delay_ms(MOTOR_DELAY_MS);
252                 PORTD = 0b10010000;
253                 _delay_ms(MOTOR_DELAY_MS);
254             }
255             }else{
256             for (uint16_t x = 0; x < degreesToMove;x++){
257                 PORTD = 0b11000000;
258                 _delay_ms(MOTOR_DELAY_MS);
259                 PORTD = 0b01100000;
260                 _delay_ms(MOTOR_DELAY_MS);
261                 PORTD = 0b00110000;
262                 _delay_ms(MOTOR_DELAY_MS);
263                 PORTD = 0b10010000;
264                 _delay_ms(MOTOR_DELAY_MS);
265             }
266         }
267 
268         PORTD = 0b00000000;
269         previousStretcherPosition = positionToMove;
270     }
271     bit_clear(PORTB, BIT(1));
272     bit_clear(PORTB, BIT(2));
273 }
```

```
 1
 2
 3  #ifndef COMMAND_HANDLER_H_
 4  #define COMMAND_HANDLER_H_
 5
 6  #ifndef nullptr
 7  #define nullptr ((void *)0)
 8  #endif
 9
10  #ifndef F_CPU
11  #define F_CPU 16000000UL
12  #endif
13
14  #include <stdbool.h>
15  #include <stdint.h>
16  #include <stdio.h>
17  #include <string.h>
18  #include <stdlib.h>
19  #include <avr/io.h>
20  #include <util/delay.h>
21  #include "nrf24.h"
22
23  #ifndef BIT_MANIPULATION_MACRO
24  #define BIT_MANIPULATION_MACRO 1
25  #define bit_get(p,m) ((p) & (m))
26  #define bit_set(p,m) ((p) |= (m))
27  #define bit_clear(p,m) ((p) &= ~(m))
28  #define bit_flip(p,m) ((p) ^= (m))
29  #define bit_write(c,p,m) (c ? bit_set(p,m) : bit_clear(p,m))
30  #define BIT(x) (0x01 << (x))
31  #define LONGBIT(x) ((unsigned long)0x00000001 << (x))
32  #endif
33
34  #define currentModuleID 0x03
35  #define SOH 0x01
36  #define STX 0x02
37  #define ETX 0x03
38  #define ETB 0x17
39  #define ON_STATE    0xFF
40  #define OFF_STATE   0x00
41
42  typedef struct CommandType {
43      void (*handlerFunction)();
44  } CommandType;
45
46  typedef enum   {
47      SUCCESFUL_DECOMPOSITION,
48      WRONG_HEADER_SEGMENTATION,
49      WRONG_FOOTER_SEGMENTATION,
50      WRONG_CHECKSUM_CONSISTENCY,
51      WRONG_MODULE_ID,
52      UNDEFINED_COMMAND_CODE,
```

```
53      PARAMETER_DATA_OVERFLOW,
54      PARAMETER_COUNT_OVERSIZE,
55      RETRANSMISSION_FAILED,
56      SUCCESFUL_RETRANSMISSION,
57      SUCCESFUL_COMPOSITION
58  } CommandStatus;
59
60
61  typedef enum {
62      RF_SUCCESFUL_TRANSMISSION,
63      RF_UNREACHEABLE_MODULE,
64      RF_ACKNOWLEDGE_FAILED
65  } RF_TransmissionStatus;
66
67  typedef enum   {
68                  UPDATE_ALL_DEVICES_VALUE_ID,
69                  UPDATE_DEVICE_VALUE_ID,
70                  GET_ALL_DEVICES_VALUE_ID,
71                  GET_DEVICE_VALUE_ID,
72                  MESSAGE_STATUS_ID
73  } CommandTypeID;
74
75  typedef struct {
76      void *startingPointer;
77      uint8_t byteLength;
78  } Parameter;
79
80  Parameter parameter[12];
81  uint8_t *command_buffer;
82  bool memoryInitialized;
83  uint8_t lastMessagePID;
84  CommandType lastMessageCommandType;
85  uint8_t lastTargetModuleID;
86  uint8_t lastTransmitterModuleID;
87
88
89  #define AVAILABLE_DEVICES 3
90  uint8_t deviceStoredValue[AVAILABLE_DEVICES];       //Uint8, las posiciones no se
       guardan en grados
91
92
93
94  void STRETCHER_POS_CHANGE_HANDLE(uint8_t positionToMove);
95  void CURTAIN_POS_CHANGE_HANDLE(uint8_t positionToMove);
96
97  extern void UPDATE_ALL_DEVICES_VALUE_H(), UPDATE_DEVICE_VALUE_H(),
      GET_ALL_DEVICES_VALUE_H(), GET_DEVICE_VALUE_H(), MESSAGE_STATUS_H();
98  extern CommandStatus ComposeMessageToBuffer(CommandTypeID targetTypeID, uint8_t
      parameterCount, uint8_t targetBoardID);
99  extern CommandStatus DecomposeMessageFromBuffer();
100 extern void writeParameterValue(uint8_t parameterIndex, void* parameterData,
      uint8_t parameterByteLength);
```

```
101  extern void HandleAvailableCommand();
102  extern bool initliazeMemory();
103
104  #endif /* COMMAND_HANDLER_H_ */
```

```
 1
 2  #define UCPHA0 1
 3  #define BAUD_RATE 9600UL
 4  #define UBRR_VALUE ((F_CPU)/(2UL*BAUD_RATE))-1
 5
 6  #include "nrf24.h"
 7  #include "Command_Handler.h"
 8
 9  uint8_t payload_len;
10  uint8_t selectedChannel;
11
12  uint8_t MOTORIZED_BOARD_ADDR[5] =   {0xF0,0xF0,0xF0,0xF0,0xC9};
13  uint8_t MAIN_BOARD_ADDR[5] =            {0xA4,0xA4,0xA4,0xA4,0xA4};
14  uint8_t POWER_BOARD_ADDR[5] =       {0xF0,0xF0,0xF0,0xF0,0xF0};
15
16  uint8_t* BOARD_ADDRESS[3] = {&MAIN_BOARD_ADDR[0], &POWER_BOARD_ADDR[0],
      &MOTORIZED_BOARD_ADDR[0]};
17  uint8_t* CURRENT_BOARD_ADDRESS = &MOTORIZED_BOARD_ADDR[0];
18
19  uint8_t GENERAL_RF_CHANNEL = 112;
20
21
22
23  void nrf24_init()
24  {
25      nrf24_setupPins();
26      nrf24_ce_digitalWrite(LOW);
27      nrf24_csn_digitalWrite(HIGH);
28  }
29
30  void nrf24_config(uint8_t channel, uint8_t pay_length)
31  {
32      /* Use static payload length ... */
33      payload_len = pay_length;
34      selectedChannel = channel;
35      // Set RF channel
36      nrf24_configRegister(RF_CH,channel);
37      // Set length of incoming payload
38      nrf24_configRegister(RX_PW_P0, 0x00); // Auto-ACK pipe ...
39      nrf24_configRegister(RX_PW_P1, payload_len); // Data payload pipe
40      nrf24_configRegister(RX_PW_P2, 0x00); // Pipe not used
41      nrf24_configRegister(RX_PW_P3, 0x00); // Pipe not used
42      nrf24_configRegister(RX_PW_P4, 0x00); // Pipe not used
43      nrf24_configRegister(RX_PW_P5, 0x00); // Pipe not used
44      // 1 Mbps, TX gain: 0dbm
45      nrf24_configRegister(RF_SETUP, (0<<RF_DR)|((0x03)<<RF_PWR));
46      // CRC enable, 1 byte CRC length
47      nrf24_configRegister(CONFIG,nrf24_CONFIG);
48      // Auto Acknowledgment
49      nrf24_configRegister(EN_AA,(1<<ENAA_P0)|(1<<ENAA_P1)|(0<<ENAA_P2)|
          (0<<ENAA_P3)|(0<<ENAA_P4)|(0<<ENAA_P5));
50      // Enable RX addresses
```

```
51      nrf24_configRegister(EN_RXADDR,(1<<ERX_P0)|(1<<ERX_P1)|(0<<ERX_P2)|
          (0<<ERX_P3)|(0<<ERX_P4)|(0<<ERX_P5));
52      // Auto retransmit delay: 1000 us and Up to 15 retransmit trials
53      nrf24_configRegister(SETUP_RETR,(0x04<<ARD)|(0x0F<<ARC));
54      // Dynamic length configurations: No dynamic length
55      nrf24_configRegister(DYNPD,(0<<DPL_P0)|(0<<DPL_P1)|(0<<DPL_P2)|(0<<DPL_P3)|
          (0<<DPL_P4)|(0<<DPL_P5));
56
57  }
58
59  bool nrf24_checkConfig(){
60      // Check all registers
61      if (nrf24_checkRegister(RF_CH, selectedChannel,1)==false) return false;
62      if (nrf24_checkRegister(RX_PW_P0, 0x00,1)==false) return false;
63      if (nrf24_checkRegister(RX_PW_P1, payload_len,1)==false) return false;
64      if (nrf24_checkRegister(RX_PW_P2, 0x00,1)==false) return false;
65      if (nrf24_checkRegister(RX_PW_P3, 0x00,1)==false) return false;
66      if (nrf24_checkRegister(RX_PW_P4, 0x00,1)==false) return false;
67      if (nrf24_checkRegister(RX_PW_P5, 0x00,1)==false) return false;
68      if (nrf24_checkRegister(RF_SETUP, (0<<RF_DR)|((0x03)<<RF_PWR),1)==false)
          return false;
69      if (nrf24_checkRegister(CONFIG,nrf24_CONFIG,1)==false) return false;
70      if (nrf24_checkRegister(EN_AA,(1<<ENAA_P0)|(1<<ENAA_P1)|(0<<ENAA_P2)|
          (0<<ENAA_P3)|(0<<ENAA_P4)|(0<<ENAA_P5),1)==false) return false;
71      if (nrf24_checkRegister(SETUP_RETR,(0x04<<ARD)|(0x0F<<ARC),1)==false) return
          false;
72      if (nrf24_checkRegister(DYNPD,(0<<DPL_P0)|(0<<DPL_P1)|(0<<DPL_P2)|
          (0<<DPL_P3)|(0<<DPL_P4)|(0<<DPL_P5),1)==false) return false;
73
74      return true;
75  }
76
77  bool nrf24_checkAvailability(){
78      if (nrf24_checkRegister(RF_CH, selectedChannel,1)==true) { return true; }
          else { return false;}
79  }
80
81
82
83
84  void faultyRF_Alarm(){
85      CLEAR_FAULTY_RF_LED;
86      for (uint8_t x = 0; x < 6; x++)
87      {
88          FLIP_FAULTY_RF_LED;
89          _delay_ms(125);
90      }
91      _delay_ms(250);
92  }
93
94
95
```

```
 96  /* Set the RX address */
 97  void nrf24_rx_address(uint8_t * adr)
 98  {
 99      nrf24_ce_digitalWrite(LOW);
100      nrf24_writeRegister(RX_ADDR_P1,adr,nrf24_ADDR_LEN);
101      nrf24_ce_digitalWrite(HIGH);
102  }
103
104  /* Returns the payload length */
105  uint8_t nrf24_payload_length()
106  {
107      return payload_len;
108  }
109
110  /* Set the TX address */
111  void nrf24_tx_address(uint8_t* adr)
112  {
113      /* RX_ADDR_P0 must be set to the sending addr for auto ack to work. */
114      nrf24_writeRegister(RX_ADDR_P0,adr,nrf24_ADDR_LEN);
115      nrf24_writeRegister(TX_ADDR,adr,nrf24_ADDR_LEN);
116  }
117
118  /* Checks if data is available for reading */
119  /* Returns 1 if data is ready ... */
120  uint8_t nrf24_dataReady()
121  {
122      // See note in getData() function - just checking RX_DR isn't good enough
123      uint8_t status = nrf24_getStatus();
124
125      // We can short circuit on RX_DR, but if it's not set, we still need
126      // to check the FIFO for any pending packets
127      if ( status & (1 << RX_DR) )
128      {
129          return 1;
130      }
131
132      return !nrf24_rxFifoEmpty();;
133  }
134
135  /* Checks if receive FIFO is empty or not */
136  uint8_t nrf24_rxFifoEmpty()
137  {
138      uint8_t fifoStatus;
139
140      nrf24_readRegister(FIFO_STATUS,&fifoStatus,1);
141
142      return (fifoStatus & (1 << RX_EMPTY));
143  }
144
145  /* Returns the length of data waiting in the RX fifo */
146  uint8_t nrf24_payloadLength()
147  {
```

```
148     uint8_t status;
149     nrf24_csn_digitalWrite(LOW);
150     spi_transfer(R_RX_PL_WID);
151     status = spi_transfer(0x00);
152     nrf24_csn_digitalWrite(HIGH);
153     return status;
154 }
155 
156 /* Reads payload bytes into data array */
157 void nrf24_getData(uint8_t* data)
158 {
159     /* Pull down chip select */
160     nrf24_csn_digitalWrite(LOW);
161 
162     /* Send cmd to read rx payload */
163     spi_transfer( R_RX_PAYLOAD );
164 
165     /* Read payload */
166     nrf24_transferSync(data,data,payload_len);
167 
168     /* Pull up chip select */
169     nrf24_csn_digitalWrite(HIGH);
170 
171     /* Reset status register */
172     nrf24_configRegister(STATUS,(1<<RX_DR));
173 }
174 
175 /* Returns the number of retransmissions occured for the last message */
176 uint8_t nrf24_retransmissionCount()
177 {
178     uint8_t rv;
179     nrf24_readRegister(OBSERVE_TX,&rv,1);
180     rv = rv & 0x0F;
181     return rv;
182 }
183 
184 // Sends a data package to the default address. Be sure to send the correct
185 // amount of bytes as configured as payload on the receiver.
186 void nrf24_send(uint8_t* value)
187 {
188     /* Go to Standby-I first */
189     nrf24_ce_digitalWrite(LOW);
190 
191     /* Set to transmitter mode , Power up if needed */
192     nrf24_powerUpTx();
193 
194     /* Do we really need to flush TX fifo each time ? */
195     #if 1
196         /* Pull down chip select */
197         nrf24_csn_digitalWrite(LOW);
198 
199         /* Write cmd to flush transmit FIFO */
```

```
200         spi_transfer(FLUSH_TX);
201
202         /* Pull up chip select */
203         nrf24_csn_digitalWrite(HIGH);
204     #endif
205
206     /* Pull down chip select */
207     nrf24_csn_digitalWrite(LOW);
208
209     /* Write cmd to write payload */
210     spi_transfer(W_TX_PAYLOAD);
211
212     /* Write payload */
213     nrf24_transmitSync(value,payload_len);
214
215     /* Pull up chip select */
216     nrf24_csn_digitalWrite(HIGH);
217
218     /* Start the transmission */
219     nrf24_ce_digitalWrite(HIGH);
220 }
221
222 uint8_t nrf24_isSending()
223 {
224     uint8_t status;
225
226     /* read the current status */
227     status = nrf24_getStatus();
228
229     /* if sending successful (TX_DS) or max retries exceded (MAX_RT). */
230     if((status & ((1 << TX_DS)  | (1 << MAX_RT))))
231     {
232         return 0; /* false */
233     }
234
235     return 1; /* true */
236
237 }
238
239 uint8_t nrf24_getStatus()
240 {
241     uint8_t rv;
242     nrf24_csn_digitalWrite(LOW);
243     rv = spi_transfer(NOP);
244     nrf24_csn_digitalWrite(HIGH);
245     return rv;
246 }
247
248 uint8_t nrf24_lastMessageStatus()
249 {
250     uint8_t rv;
251
```

```
252     rv = nrf24_getStatus();
253
254     /* Transmission went OK */
255     if((rv & ((1 << TX_DS))))
256     {
257         return NRF24_TRANSMISSON_OK;
258     }
259     /* Maximum retransmission count is reached */
260     /* Last message probably went missing ... */
261     else if((rv & ((1 << MAX_RT))))
262     {
263         return NRF24_MESSAGE_LOST;
264     }
265     /* Probably still sending ... */
266     else
267     {
268         return 0xFF;
269     }
270 }
271
272 void nrf24_powerUpRx()
273 {
274     nrf24_csn_digitalWrite(LOW);
275     spi_transfer(FLUSH_RX);
276     nrf24_csn_digitalWrite(HIGH);
277
278     nrf24_configRegister(STATUS,(1<<RX_DR)|(1<<TX_DS)|(1<<MAX_RT));
279
280     nrf24_ce_digitalWrite(LOW);
281     nrf24_configRegister(CONFIG,nrf24_CONFIG|((1<<PWR_UP)|(1<<PRIM_RX)));
282     nrf24_ce_digitalWrite(HIGH);
283 }
284
285 void nrf24_powerUpTx()
286 {
287     nrf24_configRegister(STATUS,(1<<RX_DR)|(1<<TX_DS)|(1<<MAX_RT));
288
289     nrf24_configRegister(CONFIG,nrf24_CONFIG|((1<<PWR_UP)|(0<<PRIM_RX)));
290 }
291
292 void nrf24_powerDown()
293 {
294     nrf24_ce_digitalWrite(LOW);
295     nrf24_configRegister(CONFIG,nrf24_CONFIG);
296 }
297
298 uint8_t spi_transfer(uint8_t tx)
299 {
300     uint8_t i = 0;
301     uint8_t rx = 0;
302
303     nrf24_sck_digitalWrite(LOW);
```

```
304
305     for(i=0;i<8;i++)
306     {
307
308         if(tx & (1<<(7-i)))
309         {
310             nrf24_mosi_digitalWrite(HIGH);
311         }
312         else
313         {
314             nrf24_mosi_digitalWrite(LOW);
315         }
316
317         nrf24_sck_digitalWrite(HIGH);
318
319         rx = rx << 1;
320         if(nrf24_miso_digitalRead())
321         {
322             rx |= 0x01;
323         }
324
325         nrf24_sck_digitalWrite(LOW);
326
327     }
328
329     return rx;
330 }
331
332 /* send and receive multiple bytes over SPI */
333 void nrf24_transferSync(uint8_t* dataout,uint8_t* datain,uint8_t len)
334 {
335     uint8_t i;
336
337     for(i=0;i<len;i++)
338     {
339         datain[i] = spi_transfer(dataout[i]);
340     }
341
342 }
343
344 /* send multiple bytes over SPI */
345 void nrf24_transmitSync(uint8_t* dataout,uint8_t len)
346 {
347     uint8_t i;
348
349     for(i=0;i<len;i++)
350     {
351         spi_transfer(dataout[i]);
352     }
353
354 }
355
```

```
356 /* Clocks only one byte into the given nrf24 register */
357 void nrf24_configRegister(uint8_t reg, uint8_t value)
358 {
359     nrf24_csn_digitalWrite(LOW);
360     spi_transfer(W_REGISTER | (REGISTER_MASK & reg));
361     spi_transfer(value);
362     nrf24_csn_digitalWrite(HIGH);
363 }
364 
365 /* Read single register from nrf24 */
366 void nrf24_readRegister(uint8_t reg, uint8_t* value, uint8_t len)
367 {
368     nrf24_csn_digitalWrite(LOW);
369     spi_transfer(R_REGISTER | (REGISTER_MASK & reg));
370     nrf24_transferSync(value,value,len);
371     nrf24_csn_digitalWrite(HIGH);
372 }
373 
374 /* Write to a single register of nrf24 */
375 void nrf24_writeRegister(uint8_t reg, uint8_t* value, uint8_t len)
376 {
377     nrf24_csn_digitalWrite(LOW);
378     spi_transfer(W_REGISTER | (REGISTER_MASK & reg));
379     nrf24_transmitSync(value,len);
380     nrf24_csn_digitalWrite(HIGH);
381 }
382 
383 /* Check single register from nrf24 */
384 bool nrf24_checkRegister(uint8_t reg, uint8_t desiredValue, uint8_t len)
385 {
386     uint8_t registerValue;
387     nrf24_readRegister(reg,&registerValue,len);
388     if (registerValue==desiredValue) { return true; } else { return false; }
389 }
390 
391 #define RF_DDR  DDRC
392 #define RF_PORT PORTC
393 #define RF_PIN  PINC
394 
395 #define set_bit(reg,bit) reg |= (1<<bit)
396 #define clr_bit(reg,bit) reg &= ~(1<<bit)
397 #define check_bit(reg,bit) (reg&(1<<bit))
398 
399 /* ---------------------------------------------------------------------------- */
400 
401 void nrf24_setupPins()
402 {
403     set_bit(RF_DDR,0); // CE output
404     set_bit(RF_DDR,1); // CSN output
405     set_bit(RF_DDR,2); // SCK output
406     set_bit(RF_DDR,3); // MOSI output
407     clr_bit(RF_DDR,4); // MISO input
```

```
408 }
409 /* ------------------------------------------------------------------------ */
410 void nrf24_ce_digitalWrite(uint8_t state)
411 {
412     if(state)
413     {
414         set_bit(RF_PORT,0);
415     }
416     else
417     {
418         clr_bit(RF_PORT,0);
419     }
420 }
421 /* ------------------------------------------------------------------------ */
422 void nrf24_csn_digitalWrite(uint8_t state)
423 {
424     if(state)
425     {
426         set_bit(RF_PORT,1);
427     }
428     else
429     {
430         clr_bit(RF_PORT,1);
431     }
432 }
433 /* ------------------------------------------------------------------------ */
434 void nrf24_sck_digitalWrite(uint8_t state)
435 {
436     if(state)
437     {
438         set_bit(RF_PORT,2);
439     }
440     else
441     {
442         clr_bit(RF_PORT,2);
443     }
444 }
445 /* ------------------------------------------------------------------------ */
446 void nrf24_mosi_digitalWrite(uint8_t state)
447 {
448     if(state)
449     {
450         set_bit(RF_PORT,3);
451     }
452     else
453     {
454         clr_bit(RF_PORT,3);
455     }
456 }
457 /* ------------------------------------------------------------------------ */
458 uint8_t nrf24_miso_digitalRead()
459 {
```

```
460     return check_bit(RF_PIN,4);
461 }
462 /* ------------------------------------------------------------------------ */
463
464 void nrf24_initRF_SAFE(uint8_t boardIndex,TransmissionMode initMode){
465
466     initliazeMemory();
467     bool successfulRfInit = false;
468
469     while(successfulRfInit==false){
470         nrf24_powerDown();
471         nrf24_init();
472         nrf24_config(GENERAL_RF_CHANNEL,32);
473         if (nrf24_checkConfig()) { successfulRfInit = true; } else
            { faultyRF_Alarm(); }
474     }
475
476     if (initMode==TRANSMIT){
477         nrf24_tx_address(CURRENT_BOARD_ADDRESS);
478         nrf24_rx_address(BOARD_ADDRESS[boardIndex]);
479         }else{
480         nrf24_tx_address(BOARD_ADDRESS[boardIndex]);
481         nrf24_rx_address(CURRENT_BOARD_ADDRESS);
482     }
483     nrf24_powerUpRx();
484 }
```

```
#ifndef NRF24
#define NRF24

#ifndef F_CPU
#define F_CPU 16000000UL
#endif

#include "nRF24L01_Definitions.h"
#include "Command_Handler.h"
#include <stdint.h>
#include <stdbool.h>
#include <avr/io.h>
#include <avr/delay.h>



#ifndef BIT_MANIPULATION_MACRO
#define BIT_MANIPULATION_MACRO 1
#define bit_get(p,m) ((p) & (m))
#define bit_set(p,m) ((p) |= (m))
#define bit_clear(p,m) ((p) &= ~(m))
#define bit_flip(p,m) ((p) ^= (m))
#define bit_write(c,p,m) (c ? bit_set(p,m) : bit_clear(p,m))
#define BIT(x) (0x01 << (x))
#define LONGBIT(x) ((unsigned long)0x00000001 << (x))
#endif

#define LOW 0
#define HIGH 1
#define nrf24_ADDR_LEN 5
#define nrf24_CONFIG ((1<<EN_CRC)|(0<<CRCO))
#define NRF24_TRANSMISSON_OK 0
#define NRF24_MESSAGE_LOST   1

#define AVAILABLE_COMMAND_BOARDS    2
#define CLEAR_FAULTY_RF_LED         bit_clear(PORTB, BIT(1))
#define FLIP_FAULTY_RF_LED          bit_flip(PORTB, BIT(1))


enum TransmissionMode {
    RECEIVE,
    TRANSMIT
};
typedef enum TransmissionMode TransmissionMode;

enum CommandsBoard {
    MAIN_BOARD = 0,
    POWER_BOARD = 1,
    MOTORIZED_BOARD = 2
};
typedef enum CommandsBoard CommandsBoard;

```

```
53  extern void nrf24_initRF_SAFE(uint8_t boardIndex,TransmissionMode initMode);
54
55  void    nrf24_init();
56  void    nrf24_rx_address(uint8_t* adr);
57  void    nrf24_tx_address(uint8_t* adr);
58  void    nrf24_config(uint8_t channel, uint8_t pay_length);
59  bool    nrf24_checkRegister(uint8_t reg, uint8_t desiredValue, uint8_t len);
60  bool    nrf24_checkConfig();
61  bool    nrf24_checkAvailability();
62
63  void faultyRF_Alarm();
64
65
66
67  uint8_t nrf24_dataReady();
68  uint8_t nrf24_isSending();
69  uint8_t nrf24_getStatus();
70  uint8_t nrf24_rxFifoEmpty();
71
72  void    nrf24_send(uint8_t* value);
73  void    nrf24_getData(uint8_t* data);
74
75  uint8_t nrf24_payloadLength();
76
77  uint8_t nrf24_lastMessageStatus();
78  uint8_t nrf24_retransmissionCount();
79
80  uint8_t nrf24_payload_length();
81
82  void    nrf24_powerUpRx();
83  void    nrf24_powerUpTx();
84  void    nrf24_powerDown();
85
86  uint8_t spi_transfer(uint8_t tx);
87  void    nrf24_transmitSync(uint8_t* dataout,uint8_t len);
88  void    nrf24_transferSync(uint8_t* dataout,uint8_t* datain,uint8_t len);
89  void    nrf24_configRegister(uint8_t reg, uint8_t value);
90  void    nrf24_readRegister(uint8_t reg, uint8_t* value, uint8_t len);
91  void    nrf24_writeRegister(uint8_t reg, uint8_t* value, uint8_t len);
92
93  extern void nrf24_setupPins();
94
95  extern void nrf24_ce_digitalWrite(uint8_t state);
96
97  extern void nrf24_csn_digitalWrite(uint8_t state);
98
99  extern void nrf24_sck_digitalWrite(uint8_t state);
100
101 extern void nrf24_mosi_digitalWrite(uint8_t state);
102
103 extern uint8_t nrf24_miso_digitalRead();
104
```

```
105  #endif
106
```

```
 1
 2  /* Memory Map */
 3  #define CONFIG      0x00
 4  #define EN_AA       0x01
 5  #define EN_RXADDR   0x02
 6  #define SETUP_AW    0x03
 7  #define SETUP_RETR  0x04
 8  #define RF_CH       0x05
 9  #define RF_SETUP    0x06
10  #define STATUS      0x07
11  #define OBSERVE_TX  0x08
12  #define CD          0x09
13  #define RX_ADDR_P0  0x0A
14  #define RX_ADDR_P1  0x0B
15  #define RX_ADDR_P2  0x0C
16  #define RX_ADDR_P3  0x0D
17  #define RX_ADDR_P4  0x0E
18  #define RX_ADDR_P5  0x0F
19  #define TX_ADDR     0x10
20  #define RX_PW_P0    0x11
21  #define RX_PW_P1    0x12
22  #define RX_PW_P2    0x13
23  #define RX_PW_P3    0x14
24  #define RX_PW_P4    0x15
25  #define RX_PW_P5    0x16
26  #define FIFO_STATUS 0x17
27  #define DYNPD       0x1C
28
29  /* Bit Mnemonics */
30
31  /* configuration register */
32  #define MASK_RX_DR  6
33  #define MASK_TX_DS  5
34  #define MASK_MAX_RT 4
35  #define EN_CRC      3
36  #define CRCO        2
37  #define PWR_UP      1
38  #define PRIM_RX     0
39
40  /* enable auto acknowledgment */
41  #define ENAA_P5     5
42  #define ENAA_P4     4
43  #define ENAA_P3     3
44  #define ENAA_P2     2
45  #define ENAA_P1     1
46  #define ENAA_P0     0
47
48  /* enable rx addresses */
49  #define ERX_P5      5
50  #define ERX_P4      4
51  #define ERX_P3      3
52  #define ERX_P2      2
```

```
 53  #define ERX_P1      1
 54  #define ERX_P0      0
 55
 56  /* setup of address width */
 57  #define AW          0 /* 2 bits */
 58
 59  /* setup of auto re-transmission */
 60  #define ARD         4 /* 4 bits */
 61  #define ARC         0 /* 4 bits */
 62
 63  /* RF setup register */
 64  #define PLL_LOCK    4
 65  #define RF_DR       3
 66  #define RF_PWR      1 /* 2 bits */
 67
 68  /* general status register */
 69  #define RX_DR       6
 70  #define TX_DS       5
 71  #define MAX_RT      4
 72  #define RX_P_NO     1 /* 3 bits */
 73  #define TX_FULL     0
 74
 75  /* transmit observe register */
 76  #define PLOS_CNT    4 /* 4 bits */
 77  #define ARC_CNT     0 /* 4 bits */
 78
 79  /* fifo status */
 80  #define TX_REUSE    6
 81  #define FIFO_FULL   5
 82  #define TX_EMPTY    4
 83  #define RX_FULL     1
 84  #define RX_EMPTY    0
 85
 86  /* dynamic length */
 87  #define DPL_P0      0
 88  #define DPL_P1      1
 89  #define DPL_P2      2
 90  #define DPL_P3      3
 91  #define DPL_P4      4
 92  #define DPL_P5      5
 93
 94  /* Instruction Mnemonics */
 95  #define R_REGISTER    0x00 /* last 4 bits will indicate reg. address */
 96  #define W_REGISTER    0x20 /* last 4 bits will indicate reg. address */
 97  #define REGISTER_MASK 0x1F
 98  #define R_RX_PAYLOAD  0x61
 99  #define W_TX_PAYLOAD  0xA0
100  #define FLUSH_TX      0xE1
101  #define FLUSH_RX      0xE2
102  #define REUSE_TX_PL   0xE3
103  #define ACTIVATE      0x50
104  #define R_RX_PL_WID   0x60
```

```
105  #define NOP           0xFF
106
```

```c
/**
 * \file
 * Functions and types for CRC checks.
 *
 * Generated on Wed Sep 11 13:55:53 2019
 * by pycrc v0.9.2, https://pycrc.org
 * using the configuration:
 *  - Width         = 8
 *  - Poly          = 0x07
 *  - XorIn         = 0x00
 *  - ReflectIn     = False
 *  - XorOut        = 0x00
 *  - ReflectOut    = False
 *  - Algorithm     = bit-by-bit-fast
 */
#include "crc.h"     /* include the header file generated with pycrc */
#include <stdlib.h>
#include <stdint.h>
#include <stdbool.h>



crc_t crc_update(crc_t crc, const void *data, size_t data_len)
{
    const unsigned char *d = (const unsigned char *)data;
    unsigned int i;
    bool bit;
    unsigned char c;

    while (data_len--) {
        c = *d++;
        for (i = 0x80; i > 0; i >>= 1) {
            bit = crc & 0x80;
            if (c & i) {
                bit = !bit;
            }
            crc <<= 1;
            if (bit) {
                crc ^= 0x07;
            }
        }
        crc &= 0xff;
    }
    return crc & 0xff;
}

```

```c
/**
 * \file
 * Functions and types for CRC checks.
 *
 * Generated on Wed Sep 11 13:56:48 2019
 * by pycrc v0.9.2, https://pycrc.org
 * using the configuration:
 *  - Width         = 8
 *  - Poly          = 0x07
 *  - XorIn         = 0x00
 *  - ReflectIn     = False
 *  - XorOut        = 0x00
 *  - ReflectOut    = False
 *  - Algorithm     = bit-by-bit-fast
 *
 * This file defines the functions crc_init(), crc_update() and crc_finalize().
 *
 * The crc_init() function returns the inital \c crc value and must be called
 * before the first call to crc_update().
 * Similarly, the crc_finalize() function must be called after the last call
 * to crc_update(), before the \c crc is being used.
 * is being used.
 *
 * The crc_update() function can be called any number of times (including zero
 * times) in between the crc_init() and crc_finalize() calls.
 *
 * This pseudo-code shows an example usage of the API:
 * \code{.c}
 * crc_t crc;
 * unsigned char data[MAX_DATA_LEN];
 * size_t data_len;
 *
 * crc = crc_init();
 * while ((data_len = read_data(data, MAX_DATA_LEN)) > 0) {
 *     crc = crc_update(crc, data, data_len);
 * }
 * crc = crc_finalize(crc);
 * \endcode
 */
#ifndef CRC_H
#define CRC_H

#include <stdlib.h>
#include <stdint.h>

#ifdef __cplusplus
extern "C" {
#endif


/**
 * The definition of the used algorithm.
```

```
53   *
54   * This is not used anywhere in the generated code, but it may be used by the
55   * application code to call algorithm-specific code, if desired.
56   */
57  #define CRC_ALGO_BIT_BY_BIT_FAST 1
58
59
60  /**
61   * The type of the CRC values.
62   *
63   * This type must be big enough to contain at least 8 bits.
64   */
65  typedef uint_fast8_t crc_t;
66
67
68  /**
69   * Calculate the initial crc value.
70   *
71   * \return     The initial crc value.
72   */
73  static inline crc_t crc_init(void)
74  {
75      return 0x00;
76  }
77
78
79  /**
80   * Update the crc value with new data.
81   *
82   * \param[in] crc      The current crc value.
83   * \param[in] data     Pointer to a buffer of \a data_len bytes.
84   * \param[in] data_len Number of bytes in the \a data buffer.
85   * \return             The updated crc value.
86   */
87  crc_t crc_update(crc_t crc, const void *data, size_t data_len);
88
89
90  /**
91   * Calculate the final crc value.
92   *
93   * \param[in] crc  The current crc value.
94   * \return     The final crc value.
95   */
96  static inline crc_t crc_finalize(crc_t crc)
97  {
98      return crc;
99  }
100
101
102 #ifdef __cplusplus
103 }           /* closing brace for extern "C" */
104 #endif
```

```
105
106 #endif      /* CRC_H */
107
```

Archivo - Application.java

```java
package com.example.SAPLM;

import android.app.Activity;
import androidx.lifecycle.LifecycleObserver;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.Context;
import android.content.IntentFilter;
import android.os.Bundle;
import android.widget.Toast;

import com.example.SAPLM.bluetoothActivities.BluetoothConnection;
import com.example.SAPLM.settingsActivities.DeveloperConsoleActivity;

import java.io.IOException;

public class Application extends android.app.Application implements LifecycleObserver {

    @Override
    public void onCreate() {
        super.onCreate();

        // ProcessLifecycleOwner.get().getLifecycle().addObserver(this);

        IntentFilter bluetoothAdapterActionChangedFilter = new IntentFilter(BluetoothAdapter.ACTION_STATE_CHANGED);
        registerReceiver(BluetoothConnection.bluetoothAdapterStatusBR, bluetoothAdapterActionChangedFilter);

        IntentFilter bluetoothScanModeStatusChangedFilter = new IntentFilter();
        bluetoothScanModeStatusChangedFilter.addAction(BluetoothAdapter.ACTION_DISCOVERY_STARTED);
        bluetoothScanModeStatusChangedFilter.addAction(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
        bluetoothScanModeStatusChangedFilter.addAction(BluetoothAdapter.ACTION_SCAN_MODE_CHANGED);
        registerReceiver(BluetoothConnection.bluetoothScanModeStatusBR, bluetoothScanModeStatusChangedFilter);

        IntentFilter bluetoothDeviceStatusChangedFilter = new IntentFilter();
        bluetoothDeviceStatusChangedFilter.addAction(BluetoothDevice.ACTION_ACL_CONNECTED);
        bluetoothDeviceStatusChangedFilter.addAction(BluetoothDevice.ACTION_ACL_DISCONNECTED);
        bluetoothDeviceStatusChangedFilter.addAction(BluetoothDevice.ACTION_ACL_DISCONNECT_REQUESTED);
        bluetoothDeviceStatusChangedFilter.addAction(BluetoothDevice.ACTION_BOND_STATE_CHANGED);
        registerReceiver(BluetoothConnection.bluetoothDeviceStatusBR, bluetoothDeviceStatusChangedFilter);

        Thread internetDetector = new Thread(){
            @Override
            public void run() {
                super.run();
                while(true){
                    try {
                        internetConnectionStatus = isConnected();
                        Thread.sleep(10000);
                        if (internetConnectionStatus) {Application.toastMessage("Internet connection detected");}
                        else {Application.toastMessage("Proper internet connection has not been detected");}
                    } catch (Throwable th) { th.printStackTrace(); Application.toastMessage(
```

Archivo - Application.java

```
th.toString());}
                }
            }
        };

        internetDetector.start();

        Application.persistentDataSave.putString("MAIN_TEXT_KEY", new String());
        Application.persistentDataSave.putString("ONGOING_TEXT_KEY", new String());

    }

    @Override
    public void onTerminate() {
        super.onTerminate();
    }

    private static Activity mActivity;
    private static Context mContext;
    public static Bundle persistentDataSave = new Bundle();


    public static Activity getActivity() { return mActivity; }
    public static void setActivity(Activity mActivitySet) { mActivity = mActivitySet; }

    public static Context getContext() { return mContext; }
    public static void setContext(Context mContextoSet) { mContext = mContextoSet; }

    public static void toastMessage(final String message){
        mActivity.runOnUiThread(new Runnable() {
            @Override
            public void run() {
                if (mActivity != null) {
                    //Toast.makeText(mActivity, message, Toast.LENGTH_SHORT).show();
                }
            }
        });
    }

    private static boolean internetConnectionStatus;

    public static boolean isSystemOnline(){
        return internetConnectionStatus;
    }

    public boolean isConnected() throws InterruptedException, IOException {
        final String command = "ping -c 1 google.com";
        return Runtime.getRuntime().exec(command).waitFor() == 0;
    }



}
```

Archivo - MainActivity.java

```java
package com.example.SAPLM;

import android.Manifest;
import android.content.pm.PackageManager;
import android.os.Bundle;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.appcompat.app.AppCompatActivity;
import android.util.DisplayMetrics;
import android.view.View;
import android.widget.ImageButton;
import android.widget.TextView;

import com.example.SAPLM.bluetoothActivities.BluetoothConnection;
import com.example.SAPLM.settingsActivities.settings_layout;

public class MainActivity extends AppCompatActivity {

    static ImageButton imgBtn_microphone;
    static ImageButton imgBtn_microphoneBackground;
    static ImageButton imgBtn_commands;
    static ImageButton imgBtn_contacts;
    static ImageButton imgBtn_bluetooth;
    static ImageButton imgBtn_settings;
    static ImageButton imgBtn_menu;
    static MainMenuHandler mainMenuHandler;
    static TextView textView_speechRecognitionResults;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        if(savedInstanceState == null) {

            mainMenuHandler = new MainMenuHandler(this, this);
            Application.setActivity(this);
            Application.setContext(this);
            MainMenuHandler.dpiConstant = ((float) this.getResources().getDisplayMetrics().
densityDpi / DisplayMetrics.DENSITY_DEFAULT);

            imgBtn_microphone = findViewById(R.id.imgBtn_microphone);
            imgBtn_microphoneBackground = findViewById(R.id.imgBtn_microphone_background);
            imgBtn_commands = findViewById(R.id.imgBtn_commands);
            imgBtn_contacts = findViewById(R.id.imgBtn_contacts);
            imgBtn_bluetooth = findViewById(R.id.imgBtn_bluetooth);
            imgBtn_settings = findViewById(R.id.imgBtn_settings);
            imgBtn_menu = findViewById(R.id.imgBtn_menu);
            textView_speechRecognitionResults = findViewById(R.id.
textView_speechRecognitionResults);

            if (ContextCompat.checkSelfPermission(this,
                    Manifest.permission.RECORD_AUDIO)
                    != PackageManager.PERMISSION_GRANTED) {
                if (ActivityCompat.shouldShowRequestPermissionRationale(this,
                        Manifest.permission.RECORD_AUDIO)) {
                } else {
                    ActivityCompat.requestPermissions(this,
                            new String[]{Manifest.permission.RECORD_AUDIO},
                            527);
                }
            } else {
```

```
62                  GoogleSpeechRecognizer.setupRecognizer();
63                  PocketsphinxListener.setupRecognizer();
64                  PocketsphinxListener.startListening();
65
66              }
67
68              settings_layout.setupSettingsList();
69
70              BluetoothConnection.initBTAutoConnectService();
71
72          }
73      }
74
75      @Override
76      public void onRequestPermissionsResult(int requestCode,
77                                             String permissions[], int[] grantResults) {
78          switch (requestCode) {
79              case 527: {
80                  // If request is cancelled, the result arrays are empty.
81                  if (grantResults.length > 0
82                          && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
83
84                      GoogleSpeechRecognizer.setupRecognizer();
85                      PocketsphinxListener.setupRecognizer();
86                      PocketsphinxListener.startListening();
87
88                  } else { }
89              }
90          }
91      }
92
93      public void onClickListener_menu(View view){ mainMenuHandler.menuButtonOnClick(); }
94
95      public void onClickListener_commands(View view){ mainMenuHandler.commandsButtonOnClick(
    ); }
96
97      public void onClickListener_settings(View view){ mainMenuHandler.settingsButtonOnClick(
    ); }
98
99      public void onClickListener_contacts(View view){ mainMenuHandler.contactsButtonOnClick(
    ); }
100
101     public void onClickListener_bluetooth(View view){ mainMenuHandler.
    bluetoothButtonOnClick(); }
102
103     public void onClickListener_microphone(View view){ MicrophoneHandler.microphoneOnClick(
    ); }
104
105     @Override
106     public void onDestroy() {
107
108         PocketsphinxListener.shutdownRecognizer();
109         GoogleSpeechRecognizer.shutdownRecognizer();
110         BluetoothConnection.closeAllConnections();
111         TextToSpeechHandler.shutdownTextSpeech();
112
113         super.onDestroy();
114     }
115
116 }
117
```

Archivo - MainMenuHandler.java

```java
package com.example.SAPLM;

import android.animation.ValueAnimator;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.view.View;
import android.view.animation.AccelerateDecelerateInterpolator;

import com.example.SAPLM.bluetoothActivities.BluetoothActivity;
import com.example.SAPLM.commandActivity.CommandsActivity;
import com.example.SAPLM.settingsActivities.settings_layout;

public class MainMenuHandler extends MainActivity{

    public static float dpiConstant;
    public static float radialMenuRadius = 263.0f;
    public static boolean menuUncoiled = false;
    public static boolean isAnimationRunning = false;
    public static Context mainMenuContext;
    public static Activity mainMenuActivity;

    public MainMenuHandler(Context context, Activity activity){
        this.mainMenuActivity = activity;
        this.mainMenuContext = context;
    }

    public static void menuButtonOnClick(){
        long animationDuration = 850;

        if(isAnimationRunning==false) {
            isAnimationRunning = true;

            if (menuUncoiled == false) {
                ValueAnimator animatorCommands = ValueAnimator.ofFloat(360, 72);
                animatorCommands.setDuration(animationDuration);
                animatorCommands.setInterpolator(new AccelerateDecelerateInterpolator());
                animatorCommands.addUpdateListener(new ValueAnimator.AnimatorUpdateListener(
) {
                    @Override
                    public void onAnimationUpdate(ValueAnimator animation) {
                        float value = ((Float) (animation.getAnimatedValue()))
                                .floatValue();

                        float translationX = (float) (Math.cos(value * (Math.PI / 180.0f)) *
 (radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
                        float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.0d
) - radialMenuRadius * translationX));
                        if (value > 180.0f) {
                            translationY = -translationY;
                        }

                        imgBtn_commands.setTranslationX(convertDpToPixel(translationX));
                        imgBtn_commands.setTranslationY(convertDpToPixel(translationY));

                    }
                });
                imgBtn_commands.setVisibility(View.VISIBLE);
                animatorCommands.start();

                ValueAnimator animatorContacts = ValueAnimator.ofFloat(360, 144);
                animatorContacts.setDuration(animationDuration);
                animatorContacts.setInterpolator(new AccelerateDecelerateInterpolator());
```

Archivo - MainMenuHandler.java

```
61                  animatorContacts.addUpdateListener(new ValueAnimator.AnimatorUpdateListener
    () {
62                      @Override
63                      public void onAnimationUpdate(ValueAnimator animation) {
64                          float value = ((Float) (animation.getAnimatedValue()))
65                                  .floatValue();
66
67                          float translationX = (float) (Math.cos(value * (Math.PI / 180.0f))
    * (radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
68                          float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.
    0d) - radialMenuRadius * translationX));
69                          if (value > 180.0f) {
70                              translationY = -translationY;
71                          }
72
73                          imgBtn_contacts.setTranslationX(convertDpToPixel(translationX));
74                          imgBtn_contacts.setTranslationY(convertDpToPixel(translationY));
75
76                      }
77                  });
78                  imgBtn_contacts.setVisibility(View.VISIBLE);
79                  animatorContacts.start();
80
81                  ValueAnimator animatorSettings = ValueAnimator.ofFloat(360, 216);
82                  animatorSettings.setDuration(animationDuration);
83                  animatorSettings.setInterpolator(new AccelerateDecelerateInterpolator());
84                  animatorSettings.addUpdateListener(new ValueAnimator.AnimatorUpdateListener
    () {
85                      @Override
86                      public void onAnimationUpdate(ValueAnimator animation) {
87                          float value = ((Float) (animation.getAnimatedValue()))
88                                  .floatValue();
89
90                          float translationX = (float) (Math.cos(value * (Math.PI / 180.0f))
    * (radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
91                          float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.
    0d) - radialMenuRadius * translationX));
92                          if (value > 180.0f) {
93                              translationY = -translationY;
94                          }
95
96                          imgBtn_settings.setTranslationX(convertDpToPixel(translationX));
97                          imgBtn_settings.setTranslationY(convertDpToPixel(translationY));
98
99                      }
100                 });
101                 imgBtn_settings.setVisibility(View.VISIBLE);
102                 animatorSettings.start();
103
104                 ValueAnimator animatorBluetooth = ValueAnimator.ofFloat(360, 288);
105                 animatorBluetooth.setDuration(animationDuration);
106                 animatorBluetooth.setInterpolator(new AccelerateDecelerateInterpolator());
107                 animatorBluetooth.addUpdateListener(new ValueAnimator.
    AnimatorUpdateListener() {
108                     @Override
109                     public void onAnimationUpdate(ValueAnimator animation) {
110                         float value = ((Float) (animation.getAnimatedValue()))
111                                 .floatValue();
112
113                         float translationX = (float) (Math.cos(value * (Math.PI / 180.0f))
    * (radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
114                         float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.
    0d) - radialMenuRadius * translationX));
```

```
115                         if (value > 180.0f) {
116                             translationY = -translationY;
117                         }
118
119                         imgBtn_bluetooth.setTranslationX(convertDpToPixel(translationX));
120                         imgBtn_bluetooth.setTranslationY(convertDpToPixel(translationY));
121
122                     }
123                 });
124                 imgBtn_bluetooth.setVisibility(View.VISIBLE);
125                 animatorBluetooth.start();
126
127                 imgBtn_commands.animate().alpha(1.0f).setDuration(animationDuration).
    setInterpolator(new AccelerateDecelerateInterpolator());
128                 imgBtn_contacts.animate().alpha(1.0f).setDuration(animationDuration).
    setInterpolator(new AccelerateDecelerateInterpolator());
129                 imgBtn_settings.animate().alpha(1.0f).setDuration(animationDuration).
    setInterpolator(new AccelerateDecelerateInterpolator());
130                 imgBtn_bluetooth.animate().alpha(1.0f).setDuration(animationDuration).
    setInterpolator(new AccelerateDecelerateInterpolator()).withEndAction(new Runnable() {
131                     @Override
132                     public void run() {
133                         isAnimationRunning = false;
134                     }
135                 });
136
137                 menuUncoiled = true;
138
139
140             } else {
141
142
143                 ValueAnimator animatorCommands = ValueAnimator.ofFloat(72, 360);
144                 animatorCommands.setDuration(animationDuration);
145                 animatorCommands.setInterpolator(new AccelerateDecelerateInterpolator());
146                 animatorCommands.addUpdateListener(new ValueAnimator.AnimatorUpdateListener
    () {
147                     @Override
148                     public void onAnimationUpdate(ValueAnimator animation) {
149                         float value = ((Float) (animation.getAnimatedValue()))
150                                 .floatValue();
151
152                         float translationX = (float) (Math.cos(value * (Math.PI / 180.0f))
    * (radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
153                         float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.
    0d) - radialMenuRadius * translationX));
154                         if (value > 180.0f) {
155                             translationY = -translationY;
156                         }
157
158                         imgBtn_commands.setTranslationX(convertDpToPixel(translationX));
159                         imgBtn_commands.setTranslationY(convertDpToPixel(translationY));
160
161                     }
162                 });
163                 imgBtn_commands.setVisibility(View.VISIBLE);
164                 animatorCommands.start();
165
166                 ValueAnimator animatorContacts = ValueAnimator.ofFloat(144, 360);
167                 animatorContacts.setDuration(animationDuration);
168                 animatorContacts.setInterpolator(new AccelerateDecelerateInterpolator());
169                 animatorContacts.addUpdateListener(new ValueAnimator.AnimatorUpdateListener
    () {
```

```
170                     @Override
171                     public void onAnimationUpdate(ValueAnimator animation) {
172                         float value = ((Float) (animation.getAnimatedValue()))
173                                 .floatValue();
174 
175                         float translationX = (float) (Math.cos(value * (Math.PI / 180.0f))
    * (radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
176                         float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.
    0d) - radialMenuRadius * translationX));
177                         if (value > 180.0f) {
178                             translationY = -translationY;
179                         }
180 
181                         imgBtn_contacts.setTranslationX(convertDpToPixel(translationX));
182                         imgBtn_contacts.setTranslationY(convertDpToPixel(translationY));
183 
184                     }
185                 });
186                 imgBtn_contacts.setVisibility(View.VISIBLE);
187                 animatorContacts.start();
188 
189                 ValueAnimator animatorSettings = ValueAnimator.ofFloat(216, 360);
190                 animatorSettings.setDuration(animationDuration);
191                 animatorSettings.setInterpolator(new AccelerateDecelerateInterpolator());
192                 animatorSettings.addUpdateListener(new ValueAnimator.AnimatorUpdateListener
    () {
193                     @Override
194                     public void onAnimationUpdate(ValueAnimator animation) {
195                         float value = ((Float) (animation.getAnimatedValue()))
196                                 .floatValue();
197 
198                         float translationX = (float) (Math.cos(value * (Math.PI / 180.0f))
    * (radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
199                         float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.
    0d) - radialMenuRadius * translationX));
200                         if (value > 180.0f) {
201                             translationY = -translationY;
202                         }
203 
204                         imgBtn_settings.setTranslationX(convertDpToPixel(translationX));
205                         imgBtn_settings.setTranslationY(convertDpToPixel(translationY));
206 
207                     }
208                 });
209                 imgBtn_settings.setVisibility(View.VISIBLE);
210                 animatorSettings.start();
211 
212                 ValueAnimator animatorBluetooth = ValueAnimator.ofFloat(288, 360);
213                 animatorBluetooth.setDuration(animationDuration);
214                 animatorBluetooth.setInterpolator(new AccelerateDecelerateInterpolator());
215                 animatorBluetooth.addUpdateListener(new ValueAnimator.
    AnimatorUpdateListener() {
216                     @Override
217                     public void onAnimationUpdate(ValueAnimator animation) {
218                         float value = ((Float) (animation.getAnimatedValue()))
219                                 .floatValue();
220 
221                         float translationX = (float) (Math.cos(value * (Math.PI / 180.0f))
    * (radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
222                         float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.
    0d) - radialMenuRadius * translationX));
223                         if (value > 180.0f) {
224                             translationY = -translationY;
```

```
225                         }
226
227                         imgBtn_bluetooth.setTranslationX(convertDpToPixel(translationX));
228                         imgBtn_bluetooth.setTranslationY(convertDpToPixel(translationY));
229
230                     }
231                 });
232                 imgBtn_bluetooth.setVisibility(View.VISIBLE);
233                 animatorBluetooth.start();
234
235                 imgBtn_commands.animate().alpha(0.0f).setDuration(animationDuration).
    setInterpolator(new AccelerateDecelerateInterpolator());
236                 imgBtn_contacts.animate().alpha(0.0f).setDuration(animationDuration).
    setInterpolator(new AccelerateDecelerateInterpolator());
237                 imgBtn_settings.animate().alpha(0.0f).setDuration(animationDuration).
    setInterpolator(new AccelerateDecelerateInterpolator());
238                 imgBtn_bluetooth.animate().alpha(0.0f).setDuration(animationDuration).
    setInterpolator(new AccelerateDecelerateInterpolator()).withEndAction(new Runnable() {
239                     @Override
240                     public void run() {
241                         isAnimationRunning = false;
242                     }
243                 });
244
245                 menuUncoiled = false;
246             }
247         }
248     }
249
250     public static void commandsButtonOnClick(){
251         Context currentContext = Application.getContext();
252         Activity currentActivity = Application.getActivity();
253         Intent i = new Intent(currentContext, CommandsActivity.class);
254         currentContext.startActivity(i);
255     }
256
257     public static void contactsButtonOnClick(){
258         Context currentContext = Application.getContext();
259         Activity currentActivity = Application.getActivity();
260         Intent i = new Intent(currentContext, ContactsActivity.class);
261         currentContext.startActivity(i);
262     }
263
264     public static void bluetoothButtonOnClick(){
265         Context currentContext = Application.getContext();
266         Activity currentActivity = Application.getActivity();
267         Intent i = new Intent(currentContext, BluetoothActivity.class);
268         currentContext.startActivity(i);
269     }
270
271     public static void settingsButtonOnClick(){
272         Intent i = new Intent(Application.getContext(), settings_layout.class);
273         mainMenuContext.startActivity(i);
274         //mainMenuActivity.overridePendingTransition(R.anim.fade_in, R.anim.fade_out);
275     }
276
277
278
279
280
281
282
283
```

```
284     /**
285      * This method converts dp unit to equivalent pixels, depending on device density.
286      *
287      * @param dp A value in dp (density independent pixels) unit. Which we need to convert
    into pixels
288      * @return A float value to represent px equivalent to dp depending on device density
289      */
290     public static float convertDpToPixel(float dp){ return dp * dpiConstant; }
291
292     /**
293      * This method converts device specific pixels to density independent pixels.
294      *
295      * @param px A value in px (pixels) unit. Which we need to convert into db
296      * @return A float value to represent dp equivalent to px value
297      */
298     public static float convertPixelsToDp(float px){
299         return px / dpiConstant;
300     }
301 }
302
```

Archivo - ContactsActivity.java

```java
package com.example.SAPLM;

import android.graphics.PorterDuff;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import androidx.core.content.ContextCompat;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import android.view.MenuItem;

public class ContactsActivity extends AppCompatActivity {
    private static Toolbar toolbar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.contacts_layout);
        Application.setActivity(this);
        Application.setContext(this);

        toolbar = this.findViewById(R.id.toolbar);

        setSupportActionBar(toolbar);

        // add back arrow to toolbar
        if (getSupportActionBar() != null){
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
            getSupportActionBar().setDisplayShowHomeEnabled(true);

            final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.abc_ic_ab_back_material);
            upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP);
            getSupportActionBar().setHomeAsUpIndicator(upArrow);
        }

        TextToSpeechHandler.initializeTextToSpeech();
        try {
            Thread.sleep(250);
        }catch (Throwable th) { Application.toastMessage(th.toString());}

        TextToSpeechHandler.addToSpeakingQueue("hola");

    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == android.R.id.home) // Press Back Icon
        {
            finish();
        }

        return super.onOptionsItemSelected(item);
    }
}

```

Archivo - MicrophoneHandler.java

```java
package com.example.SAPLM;

import android.graphics.PorterDuff;
import android.view.animation.AccelerateDecelerateInterpolator;

import java.util.concurrent.Callable;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.Future;

public class MicrophoneHandler extends MainActivity {

    public enum MicrophoneState { INHIBITED, LISTENING_KEYOWRD, LISTENING_COMMAND}

    static private MicrophoneState currentState = MicrophoneState.LISTENING_KEYOWRD;

    public static void microphoneOnClick(){
        switch (currentState){
            case LISTENING_KEYOWRD:
                PocketsphinxListener.stopRecognizer();
                setState(MicrophoneState.INHIBITED);
                break;
            case LISTENING_COMMAND:
                GoogleSpeechRecognizer.forceStopRecognizer();
                //PocketsphinxListener.startListening();
                setState(MicrophoneState.LISTENING_KEYOWRD);
                break;
            case INHIBITED:
                setState(MicrophoneState.LISTENING_KEYOWRD);
                PocketsphinxListener.startListening();
                break;
        }

    }

    public static MicrophoneState getState(){
        return currentState;
    }

    public static void setState(MicrophoneState desiredState){
        switch (desiredState){
            case INHIBITED:
                imgBtn_microphoneBackground.clearColorFilter();
                imgBtn_microphoneBackground.setColorFilter(Application.getContext().getColor(R.color.ColorBackMicInhibited), PorterDuff.Mode.SRC_ATOP);
                imgBtn_microphone.setImageResource(R.drawable.ic_main_mic_off);
                endAnimation();
                currentState = MicrophoneState.INHIBITED;
                break;
            case LISTENING_KEYOWRD:
                imgBtn_microphoneBackground.clearColorFilter();
                imgBtn_microphoneBackground.setColorFilter(Application.getContext().getColor(R.color.ColorBackMicKeyword), PorterDuff.Mode.SRC_ATOP);
                imgBtn_microphone.setImageResource(R.drawable.ic_main_mic_listening);
                endAnimation();
                currentState = MicrophoneState.LISTENING_KEYOWRD;
                break;
            case LISTENING_COMMAND:
                imgBtn_microphoneBackground.clearColorFilter();
                imgBtn_microphoneBackground.setColorFilter(Application.getContext().getColor(R.color.ColorBackMicCommand), PorterDuff.Mode.SRC_ATOP);
                imgBtn_microphone.setImageResource(R.drawable.ic_main_mic_listening);
                startAnimation();
```

```
61                  currentState = MicrophoneState.LISTENING_COMMAND;
62                  break;
63          }
64      }
65
66      private static ExecutorService executor = Executors.newSingleThreadExecutor();
67      private static boolean terminateAnimation = false;
68      private static boolean animationHasFinished = false;
69
70      public static void startAnimation(){
71          terminateAnimation = false;
72          Future<Void> future = executor.submit(new Callable<Void>() {
73              public Void call() throws Exception {
74                  try {
75                      while (terminateAnimation == false) {
76                          imgBtn_microphoneBackground.animate().scaleX(0.90f).setDuration(350
    ).setInterpolator(new AccelerateDecelerateInterpolator());
77                          imgBtn_microphone.animate().scaleX(0.90f).setDuration(350).
    setInterpolator(new AccelerateDecelerateInterpolator());
78                          imgBtn_microphoneBackground.animate().scaleY(0.90f).setDuration(350
    ).setInterpolator(new AccelerateDecelerateInterpolator());
79                          imgBtn_microphone.animate().scaleY(0.90f).setDuration(350).
    setInterpolator(new AccelerateDecelerateInterpolator()).withEndAction(new Runnable() {
80                              @Override
81                              public void run() {
82                                  animationHasFinished = true;
83                              }
84                          });
85                          while(animationHasFinished==false);
86                          animationHasFinished= false;
87                          imgBtn_microphoneBackground.animate().scaleX(1.00f).setDuration(350
    ).setInterpolator(new AccelerateDecelerateInterpolator());
88                          imgBtn_microphone.animate().scaleX(1.00f).setDuration(350).
    setInterpolator(new AccelerateDecelerateInterpolator());
89                          imgBtn_microphoneBackground.animate().scaleY(1.00f).setDuration(350
    ).setInterpolator(new AccelerateDecelerateInterpolator());
90                          imgBtn_microphone.animate().scaleY(1.00f).setDuration(350).
    setInterpolator(new AccelerateDecelerateInterpolator()).withEndAction(new Runnable() {
91                              @Override
92                              public void run() {
93                                  animationHasFinished = true;
94                              }
95                          });;
96                          while(animationHasFinished==false);
97                          animationHasFinished = false;
98                      }
99                      terminateAnimation = false;
100                 }catch (Throwable th){ ;
101                     th.printStackTrace();
102                 }
103                 return null;
104             }
105         });
106     }
107
108     public static void endAnimation(){
109         terminateAnimation = true;
110     }
111
112 }
113
```

Archivo - TextToSpeechHandler.java

```java
package com.example.SAPLM;

import android.speech.tts.TextToSpeech;
import android.speech.tts.Voice;

import java.util.Locale;
import java.util.Set;

public class TextToSpeechHandler {

    private static TextToSpeech textToSpeech;
    public static boolean isInitialized = false;

    public static void initializeTextToSpeech(){
        if (isInitialized == false) {
            textToSpeech = new TextToSpeech(Application.getContext(), new TextToSpeech.OnInitListener() {
                @Override
                public void onInit(int status) {
                    try {
                        if (status != TextToSpeech.ERROR) {
                            textToSpeech.setLanguage(new Locale("spa"));
                            //textToSpeech.setLanguage(Locale.UK);
                            //textToSpeech.setOnUtteranceProgressListener(utteranceListener);
                            textToSpeech.setPitch(1.0f);
                            textToSpeech.setSpeechRate(1.0f);
                            textToSpeech.getVoices();
                            isInitialized = true;
                        }
                    } catch (Throwable th) {
                        Application.toastMessage(th.toString());
                    }
                }
            });
        }
    }

    public static void addToSpeakingQueue(String text){
        try {
            if (isInitialized) {
                textToSpeech.speak(text, TextToSpeech.QUEUE_ADD, null, "232");
            }
        } catch (Throwable th) {
            th.printStackTrace();
            Application.toastMessage(th.toString());
        }
    }

    public static void shutdownTextSpeech(){
        if(textToSpeech != null) {

            textToSpeech.stop();
            textToSpeech.shutdown();
        }
    }

    public static Set<Voice> getVoicesList(){
        if (isInitialized){
            return textToSpeech.getVoices();
        } else {
            return null;
        }
```

Archivo - TextToSpeechHandler.java

```
62      }
63
64      public static void setVoice(Voice targetVoice){
65          if(isInitialized) {
66              textToSpeech.setVoice(targetVoice);
67          }
68      }
69
70
71
72  }
73
74
75
```

Archivo - PocketsphinxListener.java

```java
package com.example.SAPLM;

import java.io.File;

import edu.cmu.pocketsphinx.Assets;
import edu.cmu.pocketsphinx.Hypothesis;
import edu.cmu.pocketsphinx.RecognitionListener;
import edu.cmu.pocketsphinx.SpeechRecognizer;
import edu.cmu.pocketsphinx.SpeechRecognizerSetup;

public final class PocketsphinxListener implements RecognitionListener {
    private static SpeechRecognizer recognizer;
    private static String KEYPHRASE = "asistente";
    private static final String KWS_SEARCH = "wakeup";


    @Override
    public void onPartialResult(Hypothesis hypothesis) {
        if (hypothesis == null)
            return;

        String text = hypothesis.getHypstr();

        if (text.contains(KEYPHRASE)) {

            Application.toastMessage(KEYPHRASE);
            recognizer.stop();
            //recognizer.cancel();
            //recognizer.shutdown();
            GoogleSpeechRecognizer.startSpeechListening();
            //recognizer.startListening(KWS_SEARCH);
        }

    }


    @Override
    public void onResult(Hypothesis hypothesis) {
        if (hypothesis != null) {
            String text = hypothesis.getHypstr();

        }
    }

    @Override
    public void onBeginningOfSpeech() {
    }

    @Override
    public void onEndOfSpeech() {

    }

    public static void setupRecognizer(){
        try {
            Assets assets = new Assets(Application.getContext());
            File assetDir = assets.syncAssets();
            recognizer = SpeechRecognizerSetup.defaultSetup()
                    .setAcousticModel(new File(assetDir, "es-es-ptm"))
                    .setDictionary(new File(assetDir, "cmudict-es-es.dict"))
                    .setKeywordThreshold(Float.MIN_VALUE
                    )
                    .setRawLogDir(assetDir) // To disable logging of raw audio comment out
```

```
 63 this call (takes a lot of space on the device)
 64
 65                     .getRecognizer();
 66             recognizer.addListener(new PocketsphinxListener());
 67             recognizer.addKeyphraseSearch(KWS_SEARCH, KEYPHRASE);
 68         } catch (Throwable th){
 69             Application.toastMessage(th.toString());
 70         }
 71     }
 72
 73     public static void startListening(){
 74         recognizer.startListening(KWS_SEARCH);
 75     }
 76
 77     public static void shutdownRecognizer(){
 78         if (recognizer != null) {
 79             recognizer.cancel();
 80             recognizer.shutdown();
 81         }
 82     }
 83
 84     public static void stopRecognizer(){
 85         recognizer.stop();
 86     }
 87
 88     @Override
 89     public void onError(Exception error) {
 90         Application.toastMessage("Error: " + error.toString());
 91     }
 92
 93     @Override
 94     public void onTimeout() {
 95         Application.toastMessage("Timeout!");
 96     }
 97
 98
 99
100
101 }
102
```

Archivo - GoogleSpeechRecognizer.java

```java
package com.example.SAPLM;

import android.content.Intent;
import android.os.Bundle;
import android.speech.RecognitionListener;
import android.speech.RecognizerIntent;
import android.speech.SpeechRecognizer;
import android.view.View;

import com.example.SAPLM.bluetoothActivities.BluetoothConnection;
import com.example.SAPLM.bluetoothActivities.UnifiedTransmissionProtocol;

import org.apache.commons.lang3.StringUtils;

import java.text.Normalizer;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Date;
import java.util.Locale;

public class GoogleSpeechRecognizer extends MainActivity implements RecognitionListener {

    private static SpeechRecognizer speechRecognizer = null;

    private boolean forceCancel = false;

    public static void startSpeechListening(){


        String language =  "es-ES";
        Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
        intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,language);
        intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, language);
        intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_PREFERENCE, language);
        intent.putExtra(RecognizerIntent.EXTRA_ONLY_RETURN_LANGUAGE_PREFERENCE, language);
        intent.putExtra(RecognizerIntent.EXTRA_CALLING_PACKAGE, Application.getActivity().
getPackageName());
        intent.putExtra(RecognizerIntent.EXTRA_PARTIAL_RESULTS,true);
        if (Application.isSystemOnline()==false) { intent.putExtra(RecognizerIntent.
EXTRA_PREFER_OFFLINE,true); }
        speechRecognizer.startListening(intent);


        if (Application.isSystemOnline()) {
            Thread autoStop = new Thread() {
                @Override
                public void run() {
                    super.run();
                    try {
                        Thread.sleep(4000);
                        forceStopRecognizer();
                    } catch (Throwable th) {
                        Application.toastMessage(th.toString());
                    }
                }
            };
            autoStop.start();
        }

    }

```

```
62      public static void setupRecognizer(){
63          speechRecognizer = SpeechRecognizer.createSpeechRecognizer(Application.getContext()
    );
64          speechRecognizer.setRecognitionListener(new GoogleSpeechRecognizer());
65      }
66
67      public static void shutdownRecognizer(){
68          speechRecognizer.stopListening();
69          speechRecognizer.cancel();
70          speechRecognizer.destroy();
71          speechRecognizer = null;
72          System.gc();
73      }
74
75      public static void forceStopRecognizer(){
76          Application.getActivity().runOnUiThread(
77                  new Runnable() {
78                      @Override
79                      public void run() {
80                          speechRecognizer.stopListening();
81                          MicrophoneHandler.setState(MicrophoneHandler.MicrophoneState.
    LISTENING_KEYOWRD);
82                      }
83                  }
84          );
85
86      }
87
88
89      public static void analiseListenerResults(ArrayList<String> data, float[]
    confidenceScore){
90          outerloop:
91          for (int i = 0; i < data.size(); i++) {
92              String hypothesis = Normalizer.normalize(((String) data.get(i)).toLowerCase(),
    Normalizer.Form.NFKD);
93              String[] hypothesisSplit = hypothesis.split(" ");
94
95              if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(
    hypothesisSplit).contains("velador")) {
96                  UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList().get(0).
    setData(Boolean.TRUE);
97                  UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
98                  break outerloop;
99              }
100
101             if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(
    hypothesisSplit).contains("estufa")) {
102                 UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList().get(3).
    setData(Boolean.TRUE);
103                 UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
104                 break outerloop;
105             }
106
107             if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(
    hypothesisSplit).contains("ventilador")) {
108                 UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList().get(2).
    setData(Boolean.TRUE);
109                 UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
110                 break outerloop;
111             }
112
113             if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(
    hypothesisSplit).contains("luz")) {
```

```
114                 UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList().get(1).
    setData(Boolean.TRUE);
115                 UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
116                 break outerloop;
117             }
118 
119             if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(
    hypothesisSplit).contains("velador")) {
120                 UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList().get(0).
    setData(Boolean.FALSE);
121                 UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
122                 break outerloop;
123             }
124 
125             if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(
    hypothesisSplit).contains("estufa")) {
126                 UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList().get(3).
    setData(Boolean.FALSE);
127                 UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
128                 break outerloop;
129             }
130 
131             if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(
    hypothesisSplit).contains("ventilador")) {
132                 UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList().get(2).
    setData(Boolean.FALSE);
133                 UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
134                 break outerloop;
135             }
136 
137             if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(
    hypothesisSplit).contains("luz")) {
138                 UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList().get(1).
    setData(Boolean.FALSE);
139                 UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
140                 break outerloop;
141             }
142 
143             if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(
    hypothesisSplit).contains("todo")) {
144                 for(UnifiedTransmissionProtocol.Module.Device dev :
    UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList()) {
145                     dev.setData(Boolean.TRUE);
146                 }
147                 UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
148                 break outerloop;
149             }
150 
151             if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(
    hypothesisSplit).contains("todo")) {
152                 for(UnifiedTransmissionProtocol.Module.Device dev :
    UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList()) {
153                     dev.setData(Boolean.FALSE);
154                 }
155                 UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
156                 break outerloop;
157             }
158 
159             if (Arrays.asList(hypothesisSplit).contains("llamar") && Arrays.asList(
    hypothesisSplit).contains("enfermera")) {
160                 UnifiedTransmissionProtocol.modulesList.get(3).getDevicesList().get(2).
    setData(Boolean.TRUE);
161                 UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
```

```
162                 break outerloop;
163             }
164 
165             if (Arrays.asList(hypothesisSplit).contains("llamar") && Arrays.asList(
    hypothesisSplit).contains("enfermeria")) {
166                 UnifiedTransmissionProtocol.modulesList.get(3).getDevicesList().get(2).
    setData(Boolean.TRUE);
167                 UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
168                 break outerloop;
169             }
170 
171             if (Arrays.asList(hypothesisSplit).contains("subir") && Arrays.asList(
    hypothesisSplit).contains("toda") && Arrays.asList(hypothesisSplit).contains("cortina")) {
172                 UnifiedTransmissionProtocol.modulesList.get(3).getDevicesList().get(1).
    setData((byte)7);
173                 UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
174                 break outerloop;
175             }
176 
177             if (Arrays.asList(hypothesisSplit).contains("bajar") && Arrays.asList(
    hypothesisSplit).contains("toda") && Arrays.asList(hypothesisSplit).contains("cortina")) {
178                 UnifiedTransmissionProtocol.modulesList.get(3).getDevicesList().get(1).
    setData((byte)0);
179                 UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
180                 break outerloop;
181             }
182 
183             if (Arrays.asList(hypothesisSplit).contains("subir") && Arrays.asList(
    hypothesisSplit).contains("toda") && Arrays.asList(hypothesisSplit).contains("camilla")) {
184                 UnifiedTransmissionProtocol.modulesList.get(3).getDevicesList().get(0).
    setData((byte)3);
185                 UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
186                 break outerloop;
187             }
188 
189             if (Arrays.asList(hypothesisSplit).contains("bajar") && Arrays.asList(
    hypothesisSplit).contains("toda") && Arrays.asList(hypothesisSplit).contains("camilla")) {
190                 UnifiedTransmissionProtocol.modulesList.get(3).getDevicesList().get(0).
    setData((byte)0);
191                 UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
192                 break outerloop;
193             }
194 
195 
196             if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(
    hypothesisSplit).contains("rele")) {
197                 for (String word : hypothesisSplit) {
198                     if (StringUtils.isNumeric(word) == true) {
199                         int number = Integer.parseInt(word);
200                         if (number<4) {
201                             UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList()
    .get(number).setData(Boolean.TRUE);
202                             UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem(
    );
203                         }
204                         break outerloop;
205                     }
206                 }
207             }
208 
209             if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(
    hypothesisSplit).contains("rele")) {
210                 for (String word : hypothesisSplit) {
```

```
211                     if (StringUtils.isNumeric(word) == true) {
212                         int number = Integer.parseInt(word);
213                         if (number<4) {
214                             UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList()
    .get(number).setData(Boolean.FALSE);
215                             UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem(
    );
216                         }
217                         break outerloop;
218                     }
219                 }
220             }
221 
222             if (Arrays.asList(hypothesisSplit).contains("decir") && Arrays.asList(
    hypothesisSplit).contains("hora")){
223                 Date currentTime = Calendar.getInstance().getTime();
224                 SimpleDateFormat df = new SimpleDateFormat("d-MMMM-yyyy", new Locale("spa")
    );
225                 String todayAsString = df.format(currentTime);
226                 SimpleDateFormat onlyTimeFormar = new SimpleDateFormat("HH:mm:ss", new
    Locale("spa"));
227                 String onlyTimeString = onlyTimeFormar.format(currentTime);
228 
229                 //TextToSpeechHandler.initializeTextToSpeech();
230                 TextToSpeechHandler.addToSpeakingQueue("Hoy es " + todayAsString + "y son
    las" + onlyTimeString);
231                 Application.toastMessage("Decir hora");
232             }
233 
234         }
235     }
236 
237     @Override
238     public void onReadyForSpeech(Bundle params) {
239         MicrophoneHandler.setState(MicrophoneHandler.MicrophoneState.LISTENING_COMMAND);
240     }
241 
242     @Override
243     public void onBeginningOfSpeech() {
244 
245     }
246 
247     @Override
248     public void onRmsChanged(float rmsdB) {
249 
250     }
251 
252     @Override
253     public void onBufferReceived(byte[] buffer) {
254 
255     }
256 
257     @Override
258     public void onEndOfSpeech() {
259 
260         //Application.toastMessage("End of speech");
261     }
262 
263     @Override
264     public void onError(int error) {
265         //Application.toastMessage("On error" + error);
266 
267         if (forceCancel==false) {
```

```
268             PocketsphinxListener.startListening();
269             MicrophoneHandler.setState(MicrophoneHandler.MicrophoneState.LISTENING_KEYOWRD)
    ;
270         }
271     }
272 
273     @Override
274     public void onResults(Bundle results) {
275         String str = new String();
276         ArrayList<String> resultsArrayList = results.getStringArrayList(SpeechRecognizer.
    RESULTS_RECOGNITION);
277         float[] confidenceScore = results.getFloatArray(SpeechRecognizer.CONFIDENCE_SCORES)
    ;
278         analiseListenerResults(resultsArrayList, confidenceScore);
279 
280         StringBuilder builder = new StringBuilder();
281         for (String line : resultsArrayList) {
282             builder.append(line + "\n");
283         }
284 
285         MainActivity.textView_speechRecognitionResults.setText(builder.toString());
286         MainActivity.textView_speechRecognitionResults.setVisibility(View.VISIBLE);
287 
288 
289 
290         MicrophoneHandler.setState(MicrophoneHandler.MicrophoneState.LISTENING_KEYOWRD);
291 
292         PocketsphinxListener.startListening();
293     }
294 
295     @Override
296     public void onPartialResults(Bundle partialResults) {
297         String str = new String();
298         ArrayList<String> resultsArrayList = partialResults.getStringArrayList(
    SpeechRecognizer.RESULTS_RECOGNITION);
299         float[] confidenceScore = partialResults.getFloatArray(SpeechRecognizer.
    CONFIDENCE_SCORES);
300 
301         StringBuilder builder = new StringBuilder();
302         for (String line : resultsArrayList) {
303             builder.append(line + "\n");
304         }
305 
306     }
307 
308     @Override
309     public void onEvent(int eventType, Bundle params) {
310 
311     }
312 }
313 
```

Archivo - CommandItem.java

```java
package com.example.SAPLM.commandActivity;

import com.example.SAPLM.R;
import com.example.SAPLM.bluetoothActivities.UnifiedTransmissionProtocol;

import java.util.List;

public class CommandItem {
    public enum ItemType { CHILD, PARENT}
    public enum BoardGroupState {FOLDED, UNFOLDED}

    private String itemName;
    public ItemType itemType = ItemType.CHILD;
    private int deviceStateIconID;
    private int deviceStateInteger = 0;
    private int moduleIndex = 0;
    private int deviceIndex = 0;
    private List<String> childInternalID;
    private BoardGroupState groupState = BoardGroupState.UNFOLDED;



    public CommandItem( int moduleIndex, int deviceIndex ){
        this.itemName = UnifiedTransmissionProtocol.modulesList.get(moduleIndex).
getDevicesList().get(deviceIndex).getName();
        this.itemType = ItemType.CHILD;
        this.moduleIndex = moduleIndex;
        this.deviceIndex = deviceIndex;
        this.deviceStateIconID = UnifiedTransmissionProtocol.modulesList.get(moduleIndex).
getDevicesList().get(deviceIndex).getIconResourceId();
    }

    public CommandItem(int boardInternalId){
        this.itemName = UnifiedTransmissionProtocol.modulesList.get(boardInternalId).
getModuleName();
        this.itemType = ItemType.PARENT;
        this.deviceStateIconID = UnifiedTransmissionProtocol.modulesList.get(boardInternalId
).getIconResourceId();
    }

    public void onClickHandler(){
        UnifiedTransmissionProtocol.modulesList.get(moduleIndex).getDevicesList().get(
deviceIndex).deviceOnClickHandler();
    }


    //Getters

    public BoardGroupState getGroupState (){
        return groupState;
    }

    public int getDeviceStateIconID() {
        return deviceStateIconID;
    }

    public int getDeviceStateInteger() {
        return deviceStateInteger;
    }

    public int getDeviceIndex() {
        return deviceIndex;
    }
```

```
 59
 60     public String getItemName() {
 61         return itemName;
 62     }
 63
 64     public int getModuleIndex(){
 65         return moduleIndex;
 66     }
 67
 68     public List<String> getChildInternalID() {
 69         return childInternalID;
 70     }
 71
 72     public ItemType getItemType() {
 73         return itemType;
 74     }
 75
 76
 77     public String getDeviceStateSubtitle(){
 78         return UnifiedTransmissionProtocol.modulesList.get(moduleIndex).getDevicesList().
    get(deviceIndex).getDataAsString();
 79     }
 80
 81     public int getProgressBarValue(){
 82         return UnifiedTransmissionProtocol.modulesList.get(moduleIndex).getDevicesList().
    get(deviceIndex).getProgressBarValue();
 83     }
 84
 85     // Setters
 86     public void setChildInternalID(List<String> childInternalID) {
 87         this.childInternalID = childInternalID;
 88     }
 89
 90
 91     public void setDeviceStateIconID(int deviceStateIconID) {
 92         this.deviceStateIconID = deviceStateIconID;
 93     }
 94
 95     public void setDeviceStateInteger(int deviceStateInteger) {
 96         this.deviceStateInteger = deviceStateInteger;
 97     }
 98
 99     public void setModuleIndex(int moduleIndex) {
100         this.moduleIndex = moduleIndex;
101     }
102
103     public void setItemName(String itemName) {
104         this.itemName = itemName;
105     }
106
107     public void setItemType(ItemType itemType) {
108         this.itemType = itemType;
109     }
110
111     public void setDeviceIndex(int deviceIndex) {
112         this.deviceIndex = deviceIndex;
113     }
114
115     public void setGroupState (BoardGroupState groupState){
116         this.groupState = groupState;
117     }
118
119 }
```

Archivo - CommandsActivity.java

```java
package com.example.SAPLM.commandActivity;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.graphics.PorterDuff;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import androidx.core.content.ContextCompat;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.recyclerview.widget.DefaultItemAnimator;
import androidx.recyclerview.widget.DividerItemDecoration;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.os.Looper;
import android.view.MenuItem;
import android.view.View;

import com.example.SAPLM.Application;
import com.example.SAPLM.R;
import com.example.SAPLM.bluetoothActivities.BluetoothConnection;
import com.example.SAPLM.settingsActivities.AvailableSetting;
import com.example.SAPLM.settingsActivities.RecyclerTouchListener;
import com.example.SAPLM.settingsActivities.SettingsLayoutAdapter;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class CommandsActivity extends AppCompatActivity {
    private static Toolbar toolbar;

    public static List<CommandItem> commandList = new ArrayList<>();
    private RecyclerView recyclerView;
    public static CommandsListAdapter mAdapter;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.commands_layout);
        Application.setActivity(this);
        Application.setContext(this);

        toolbar = this.findViewById(R.id.toolbar);

        setSupportActionBar(toolbar);

        // add back arrow to toolbar
        if (getSupportActionBar() != null){
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
            getSupportActionBar().setDisplayShowHomeEnabled(true);

            final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
abc_ic_ab_back_material);
            upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
;
            getSupportActionBar().setHomeAsUpIndicator(upArrow);
        }

        recyclerView = (RecyclerView) findViewById(R.id.commandsRecyclerView);
```

```
62
63          mAdapter = new CommandsListAdapter(commandList);
64
65          recyclerView.setHasFixedSize(true);
66
67          // vertical RecyclerView
68          // keep movie_list_row.xml width to `match_parent`
69          RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(
    getApplicationContext());
70
71          // horizontal RecyclerView
72          // keep movie_list_row.xml width to `wrap_content`
73          // RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(
    getApplicationContext(), LinearLayoutManager.HORIZONTAL, false);
74
75          recyclerView.setLayoutManager(mLayoutManager);
76
77          // adding inbuilt divider line
78          recyclerView.addItemDecoration(new DividerItemDecoration(this, LinearLayoutManager.
    VERTICAL));
79
80          // adding custom divider line with padding 16dp
81          //recyclerView.addItemDecoration(new MyDividerItemDecoration(this,
    LinearLayoutManager.HORIZONTAL, 16));
82          recyclerView.setItemAnimator(new DefaultItemAnimator());
83
84          recyclerView.setAdapter(mAdapter);
85
86          // row click listener
87          recyclerView.addOnItemTouchListener(new RecyclerTouchListener(getApplicationContext
    (), recyclerView, new RecyclerTouchListener.ClickListener() {
88              @Override
89              public void onClick(View view, int position) {
90                  CommandItem commandItem = commandList.get(position);
91                  if (commandItem.getItemType()== CommandItem.ItemType.PARENT){
92
93                  }else{
94                      commandItem.onClickHandler();
95                  }
96
97
98                  mAdapter.notifyDataSetChanged();
99
100             }
101
102             @Override
103             public void onLongClick(View view, int position) {
104
105             }
106         }));
107
108         if (commandList.isEmpty()) setupCommandsList();
109
110         mAdapter.notifyDataSetChanged();
111     }
112
113
114     public static void updateGUI(){
115         if (mAdapter!=null) {
116             if (Looper.myLooper() == Looper.getMainLooper()) {
117                 mAdapter.notifyDataSetChanged();
118             } else {
119                 Application.getActivity().runOnUiThread(new Runnable() {
```

```
120                     @Override
121                     public void run() {
122                         mAdapter.notifyDataSetChanged();
123                     }
124                 });
125             }
126         }
127
128     }
129
130     @Override
131     public boolean onOptionsItemSelected(MenuItem item) {
132         if (item.getItemId() == android.R.id.home) // Press Back Icon
133         {
134             finish();
135         }
136
137         return super.onOptionsItemSelected(item);
138     }
139
140     public static void setupCommandsList(){
141         //commandList.add(new CommandItem("MODULO PRINCIPAL","MAIN_BOARD", new ArrayList(
    Arrays.asList("LED"))));
142
143         commandList.add(new CommandItem(1));
144         commandList.add(new CommandItem(1,0));
145         commandList.add(new CommandItem(1, 1));
146
147         commandList.add(new CommandItem(2));
148         commandList.add(new CommandItem(2,0));
149         commandList.add(new CommandItem(2, 1));
150         commandList.add(new CommandItem(2, 2));
151         commandList.add(new CommandItem(2, 3));
152
153         commandList.add(new CommandItem(3));
154         commandList.add(new CommandItem(3,0));
155         commandList.add(new CommandItem(3, 1));
156         commandList.add(new CommandItem(3, 2));
157     }
158 }
159
```

Archivo - CommandsListAdapter.java

```java
package com.example.SAPLM.commandActivity;

import android.graphics.Color;
import android.graphics.PorterDuff;
import android.graphics.Typeface;
import android.graphics.drawable.ColorDrawable;
import android.graphics.drawable.Drawable;

import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.constraintlayout.widget.Guideline;
import androidx.recyclerview.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.ProgressBar;
import android.widget.TextView;

import com.example.SAPLM.Application;
import com.example.SAPLM.R;
import com.example.SAPLM.settingsActivities.AvailableSetting;
import com.example.SAPLM.settingsActivities.SettingsLayoutAdapter;

import java.util.List;

public class CommandsListAdapter extends RecyclerView.Adapter<CommandsListAdapter.viewHolder
> {

    private List<CommandItem> commandsList;

    public class viewHolder extends RecyclerView.ViewHolder {
        public TextView command_title, command_state_text;
        public ImageView command_icon, arrow_list;
        public ProgressBar status_progress_bar;
        public Guideline main_guide;

        public viewHolder(View view) {
            super(view);
            command_title = (TextView) view.findViewById(R.id.command_title_textView);
            status_progress_bar = (ProgressBar) view.findViewById(R.id.
command_state_progressBar);
            command_state_text = (TextView) view.findViewById(R.id.command_state_textView);
            command_icon = (ImageView) view.findViewById(R.id.command_icon);
            arrow_list = (ImageView) view.findViewById(R.id.arrow_list_imageButton);
            main_guide = (Guideline) view.findViewById(R.id.guideline_main);
        }
    }


    public CommandsListAdapter(List<CommandItem> cmdList) {
        this.commandsList = cmdList;
    }

    @Override
    public viewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View itemView = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.commands_layout_child_item_style, parent, false);

        return new viewHolder(itemView);
    }

    @Override
    public void onBindViewHolder(viewHolder holder, int position) {
```

```
62          CommandItem item = commandsList.get(position);
63          holder.command_title.setText(item.getItemName());
64
65          if (item.getItemType()== CommandItem.ItemType.PARENT){
66              holder.arrow_list.setVisibility(View.VISIBLE);
67              if (item.getGroupState()== CommandItem.BoardGroupState.UNFOLDED)
68              {
69                  Drawable icon = Application.getContext().getDrawable(R.drawable.
    icons_downward_arrow);
70                  icon.setColorFilter(Application.getContext().getColor(R.color.colorPrimary)
    , PorterDuff.Mode.SRC_ATOP);
71                  holder.arrow_list.setImageDrawable(icon);
72              }else {
73                  Drawable icon = Application.getContext().getDrawable(R.drawable.
    icons_upward_arrow);
74                  icon.setColorFilter(Application.getContext().getColor(R.color.colorPrimary)
    , PorterDuff.Mode.SRC_ATOP);
75                  holder.arrow_list.setImageDrawable(icon);
76              }
77              ConstraintLayout.LayoutParams params = (ConstraintLayout.LayoutParams) holder.
    main_guide.getLayoutParams();
78              params.guidePercent = 0.05f;
79              holder.main_guide.setLayoutParams(params);
80              holder.status_progress_bar.setVisibility(View.GONE);
81              holder.command_state_text.setVisibility(View.GONE);
82              holder.command_title.setTypeface(holder.command_title.getTypeface(), Typeface.
    BOLD);
83          }else {
84              ConstraintLayout.LayoutParams params = (ConstraintLayout.LayoutParams) holder.
    main_guide.getLayoutParams();
85              holder.command_state_text.setText(item.getDeviceStateSubtitle());
86              params.guidePercent = 0.15f;
87              holder.main_guide.setLayoutParams(params);
88              holder.arrow_list.setVisibility(View.GONE);
89              holder.command_state_text.setText(item.getDeviceStateSubtitle());
90              holder.command_state_text.setVisibility(View.VISIBLE);
91              holder.status_progress_bar.setIndeterminate(false);
92              holder.status_progress_bar.setProgress(item.getProgressBarValue());
93              holder.status_progress_bar.setVisibility(View.VISIBLE);
94
95          }
96
97
98          Drawable cmdIcon = Application.getContext().getDrawable(item.getDeviceStateIconID()
    );
99          cmdIcon.setColorFilter(Application.getContext().getColor(R.color.colorPrimary),
    PorterDuff.Mode.SRC_ATOP);
100         holder.command_icon.setImageDrawable(cmdIcon);
101     }
102
103     @Override
104     public int getItemCount() {
105         return commandsList.size();
106     }
107 }
```

Archivo - settings_layout.java

```java
package com.example.SAPLM.settingsActivities;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.graphics.PorterDuff;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import androidx.core.content.ContextCompat;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.DefaultItemAnimator;
import androidx.recyclerview.widget.DividerItemDecoration;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import androidx.appcompat.widget.Toolbar;
import android.view.MenuItem;
import android.view.View;

import com.example.SAPLM.Application;
import com.example.SAPLM.R;

import java.util.ArrayList;
import java.util.List;

public class settings_layout extends AppCompatActivity {

    private static Toolbar toolbar;

    public static List<AvailableSetting> settingsList = new ArrayList<>();
    private RecyclerView recyclerView;
    public static SettingsLayoutAdapter mAdapter;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.settings_layout);
        Application.setActivity(this);
        Application.setContext(this);

        toolbar = findViewById(R.id.toolbar);

        setSupportActionBar(toolbar);

        // add back arrow to toolbar
        if (getSupportActionBar() != null){
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
            getSupportActionBar().setDisplayShowHomeEnabled(true);


            final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
abc_ic_ab_back_material);
            upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
;
            getSupportActionBar().setHomeAsUpIndicator(upArrow);
        }

        recyclerView = (RecyclerView) findViewById(R.id.settingsRecyclerView);

        mAdapter = new SettingsLayoutAdapter(settingsList);

        recyclerView.setHasFixedSize(true);

```

```
62          // vertical RecyclerView
63          // keep movie_list_row.xml width to `match_parent`
64          RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(
    getApplicationContext());
65
66          // horizontal RecyclerView
67          // keep movie_list_row.xml width to `wrap_content`
68          // RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(
    getApplicationContext(), LinearLayoutManager.HORIZONTAL, false);
69
70          recyclerView.setLayoutManager(mLayoutManager);
71
72          // adding inbuilt divider line
73          recyclerView.addItemDecoration(new DividerItemDecoration(this, LinearLayoutManager.
    VERTICAL));
74
75          // adding custom divider line with padding 16dp
76          //recyclerView.addItemDecoration(new MyDividerItemDecoration(this,
    LinearLayoutManager.HORIZONTAL, 16));
77          recyclerView.setItemAnimator(new DefaultItemAnimator());
78
79          recyclerView.setAdapter(mAdapter);
80
81          // row click listener
82          recyclerView.addOnItemTouchListener(new RecyclerTouchListener(getApplicationContext
    (), recyclerView, new RecyclerTouchListener.ClickListener() {
83              @Override
84              public void onClick(View view, int position) {
85                  AvailableSetting setting = settingsList.get(position);
86
87                  Context currentContext = Application.getContext();
88                  Activity currentActivity = Application.getActivity();
89                  Intent i = new Intent(currentContext, setting.getTargetActivity());
90                  currentContext.startActivity(i);
91                  //currentActivity.overridePendingTransition(R.anim.fade_in, R.anim.fade_out
    );
92              }
93
94              @Override
95              public void onLongClick(View view, int position) {
96
97              }
98          }));
99
100         mAdapter.notifyDataSetChanged();
101
102     }
103
104     @Override
105     public boolean onOptionsItemSelected(MenuItem item) {
106         if (item.getItemId() == android.R.id.home) // Press Back Icon
107         {
108             finish();
109         }
110
111         return super.onOptionsItemSelected(item);
112     }
113
114     public static void setupSettingsList() {
115
116         settingsList.add(new AvailableSetting(
117             Application.getContext().getString(R.string.internet_conectivity_title),
118                 Application.getContext().getString(R.string.internet_conectivity_subtitle),
```

```
119             R.drawable.ic_setting_internet_connectivity,
120             InternetConectivityActivity.class
121         ));
122 
123         settingsList.add(new AvailableSetting(
124                 Application.getContext().getString(R.string.bluetooth_settings_title),
125                 Application.getContext().getString(R.string.bluetooth_settings_subtitle),
126                 R.drawable.ic_setting_bluetooth,
127                 BluetoothSettingsActivity.class
128         ));
129 
130         settingsList.add(new AvailableSetting(
131                 Application.getContext().getString(R.string.
    voice_recognition_settings_title),
132                 Application.getContext().getString(R.string.
    voice_recognition_settings_subtitle),
133                 R.drawable.ic_setting_voice_recognition,
134                 VoiceRecognitionSettingsActivity.class
135         ));
136 
137         settingsList.add(new AvailableSetting(
138                 Application.getContext().getString(R.string.commands_settings_title),
139                 Application.getContext().getString(R.string.commands_settings_subtitle),
140                 R.drawable.ic_setting_commands_3,
141                 CommandsSettingsActivity.class
142         ));
143 
144         settingsList.add(new AvailableSetting(
145                 Application.getContext().getString(R.string.contacts_communication_title),
146                 Application.getContext().getString(R.string.contacts_communication_subtitle
    ),
147                 R.drawable.ic_setting_contacts_and_communication,
148                 ContactsCommunicationActivity.class
149         ));
150 
151         settingsList.add(new AvailableSetting(
152                 Application.getContext().getString(R.string.appearance_miscellaneous_title)
    ,
153                 Application.getContext().getString(R.string.
    appearance_miscellaneous_subtitle),
154                 R.drawable.ic_setting_appearance_and_miscellaneous,
155                 AppearanceMiscellaneousActivity.class
156         ));
157 
158         settingsList.add(new AvailableSetting(
159                 Application.getContext().getString(R.string.
    developer_console_activity_title),
160                 Application.getContext().getString(R.string.developer_console_subtitle),
161                 R.drawable.ic_setting_developer_console,
162                 DeveloperConsoleActivity.class
163         ));
164 
165         settingsList.add(new AvailableSetting(
166                 Application.getContext().getString(R.string.information_help_title),
167                 Application.getContext().getString(R.string.information_help_subtitle),
168                 R.drawable.ic_setting_information_and_help,
169                 InformationHelpActivity.class
170         ));
171 
172         // notify adapter about data set changes
173         // so that it will render the list with new data
174         //mAdapter.notifyDataSetChanged();
175     }
```

```
176
177
178 }
179
```

Archivo - TerminalManager.java

```java
package com.example.SAPLM.settingsActivities;

import android.graphics.Typeface;
import android.text.SpannableStringBuilder;
import android.text.Spanned;
import android.text.style.StyleSpan;

import com.example.SAPLM.bluetoothActivities.UnifiedTransmissionProtocol;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.regex.Matcher;
import java.util.regex.Pattern;


public class TerminalManager {

    public static void handleTerminalInput(String newTerminalInput){

    }

    public static List<String> currentParameters = new ArrayList<>();

    public static List<TerminalCommand> commandsList = new ArrayList<>(Arrays.asList(
            new TerminalCommand("COMPOSE", new Runnable() {
                @Override
                public void run() {
                    UnifiedTransmissionProtocol.CommandType command = UnifiedTransmissionProtocol.CommandType.definedCommandList.stream().filter(p-> p.getCommandTag().equals((currentParameters.get(0)))).findFirst().orElse(null);
                    UnifiedTransmissionProtocol.Module target = UnifiedTransmissionProtocol.modulesList.stream().filter(p-> p.getModuleAlias().equals((currentParameters.get(1)))).findFirst().orElse(null);

                    if (command==null) {
                        SpannableStringBuilder builder= new SpannableStringBuilder();
                        builder.append("Tipo de comando \"")
                                .append(currentParameters.get(0), new StyleSpan(Typeface.BOLD), Spanned.SPAN_EXCLUSIVE_EXCLUSIVE)
                                .append("\" no reconocido");
                        DeveloperConsoleActivity.sendTextToTerminal(builder);
                        return;
                    }

                    if (target==null) {
                        SpannableStringBuilder builder= new SpannableStringBuilder();
                        builder.append("Alias de modulo \"")
                                .append(currentParameters.get(1), new StyleSpan(Typeface.BOLD), Spanned.SPAN_EXCLUSIVE_EXCLUSIVE)
                                .append("\" no encontrado");
                        DeveloperConsoleActivity.sendTextToTerminal(builder);
                        return;
                    }


                    System.out.println( insertSpaces(UnifiedTransmissionProtocol.bytesToHex( UnifiedTransmissionProtocol.getCommand_buffer() ), " ", 2) );
                }
            })
    ));

    public static class TerminalCommand{
```

```
57          private String commandBaseTag;
58          private Runnable commandHandler;
59
60          TerminalCommand(String commandBaseTag, Runnable commandHandler){
61              this.commandBaseTag = commandBaseTag;
62              this.commandHandler = commandHandler;
63          }
64
65          public void handleCommand(){
66
67          }
68      }
69
70      public static String insertSpaces(String text, String insert, int period) {
71          Pattern p = Pattern.compile("(.{" + period + "})", Pattern.DOTALL);
72          Matcher m = p.matcher(text);
73          return m.replaceAll("$1" + insert);
74      }
75  }
76
```

Archivo - AvailableSetting.java

```java
package com.example.SAPLM.settingsActivities;

public class AvailableSetting {
    private String title, subtitle;
    private int iconResourceID;
    private Class targetActivity;

    public AvailableSetting() {
    }

    public AvailableSetting(String title, String subtitle, int iconResourceID, Class targetActivity) {
        this.title = title;
        this.subtitle = subtitle;
        this.iconResourceID = iconResourceID;
        this.targetActivity = targetActivity;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String name) {
        this.title = name;
    }

    public String getSubtitle() {
        return subtitle;
    }

    public void setSubtitle(String subtitle) {
        this.subtitle = subtitle;
    }

    public int getIconResourceID() {
        return iconResourceID;
    }

    public void setIconResourceID(int iconResourceID) {
        this.iconResourceID = iconResourceID;
    }

    public Class getTargetActivity() { return targetActivity; }

    public void setTargetActivity(Class targetActivity) { this.targetActivity = targetActivity; }
}

```

Archivo - RecyclerTouchListener.java

```java
package com.example.SAPLM.settingsActivities;

import android.content.Context;
import androidx.recyclerview.widget.RecyclerView;
import android.view.GestureDetector;
import android.view.MotionEvent;
import android.view.View;

/**
 * Created by Ravi Tamada on 03/09/16.
 * updated by Ravi on 14/11/17
 * www.androidhive.info
 */
public class RecyclerTouchListener implements RecyclerView.OnItemTouchListener {

    private GestureDetector gestureDetector;
    private ClickListener clickListener;

    public RecyclerTouchListener(Context context, final RecyclerView recyclerView, final ClickListener clickListener) {
        this.clickListener = clickListener;
        gestureDetector = new GestureDetector(context, new GestureDetector.SimpleOnGestureListener() {
            @Override
            public boolean onSingleTapUp(MotionEvent e) {
                return true;
            }

            @Override
            public void onLongPress(MotionEvent e) {
                View child = recyclerView.findChildViewUnder(e.getX(), e.getY());
                if (child != null && clickListener != null) {
                    clickListener.onLongClick(child, recyclerView.getChildAdapterPosition(child));
                }
            }
        });
    }

    @Override
    public boolean onInterceptTouchEvent(RecyclerView rv, MotionEvent e) {

        View child = rv.findChildViewUnder(e.getX(), e.getY());
        if (child != null && clickListener != null && gestureDetector.onTouchEvent(e)) {
            clickListener.onClick(child, rv.getChildAdapterPosition(child));
        }
        return false;
    }

    @Override
    public void onTouchEvent(RecyclerView rv, MotionEvent e) {
    }

    @Override
    public void onRequestDisallowInterceptTouchEvent(boolean disallowIntercept) {

    }

    public interface ClickListener {
        void onClick(View view, int position);

        void onLongClick(View view, int position);
    }
```

```
61 }
```

Archivo - SettingsLayoutAdapter.java

```java
package com.example.SAPLM.settingsActivities;

import android.graphics.PorterDuff;
import android.graphics.drawable.Drawable;
import androidx.recyclerview.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import com.example.SAPLM.Application;
import com.example.SAPLM.R;

import java.util.List;

public class SettingsLayoutAdapter extends RecyclerView.Adapter<SettingsLayoutAdapter.viewHolder> {

    private List<AvailableSetting> settingsList;

    public class viewHolder extends RecyclerView.ViewHolder {
        public TextView setting_title, setting_subtitle;
        public ImageView setting_icon;

        public viewHolder(View view) {
            super(view);
            setting_title = (TextView) view.findViewById(R.id.setting_title);
            setting_subtitle = (TextView) view.findViewById(R.id.setting_subtitle);
            setting_icon = (ImageView) view.findViewById(R.id.setting_icon);
        }
    }


    public SettingsLayoutAdapter(List<AvailableSetting> settingsList) {
        this.settingsList = settingsList;
    }

    @Override
    public viewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View itemView = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.settings_layout_style, parent, false);

        return new viewHolder(itemView);
    }

    @Override
    public void onBindViewHolder(viewHolder holder, int position) {
        AvailableSetting setting = settingsList.get(position);
        holder.setting_title.setText(setting.getTitle());
        holder.setting_subtitle.setText(setting.getSubtitle());

        Drawable icon = Application.getContext().getDrawable(setting.getIconResourceID());
        icon.setColorFilter(Application.getContext().getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP);
        holder.setting_icon.setImageDrawable(icon);
    }

    @Override
    public int getItemCount() {
        return settingsList.size();
    }
}
```

Archivo - InformationHelpActivity.java

```java
package com.example.SAPLM.settingsActivities;

import android.graphics.PorterDuff;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import androidx.core.content.ContextCompat;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import android.view.MenuItem;

import com.example.SAPLM.Application;
import com.example.SAPLM.R;

public class InformationHelpActivity extends AppCompatActivity {
    private static Toolbar toolbar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.information_help_layout);
        Application.setActivity(this);
        Application.setContext(this);

        toolbar = this.findViewById(R.id.toolbar);

        setSupportActionBar(toolbar);

        // add back arrow to toolbar
        if (getSupportActionBar() != null){
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
            getSupportActionBar().setDisplayShowHomeEnabled(true);

            final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
abc_ic_ab_back_material);
            upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
;
            getSupportActionBar().setHomeAsUpIndicator(upArrow);
        }
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == android.R.id.home) // Press Back Icon
        {
            finish();
        }

        return super.onOptionsItemSelected(item);
    }
}

```

Archivo - MyDividerItemDecoration.java

```java
package com.example.SAPLM.settingsActivities;

import android.content.Context;
import android.content.res.Resources;
import android.content.res.TypedArray;
import android.graphics.Canvas;
import android.graphics.Rect;
import android.graphics.drawable.Drawable;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import android.util.TypedValue;
import android.view.View;

/**
 * Created by Ravi on 30/10/15.
 * updated by Ravi on 14/11/17
 */
public class MyDividerItemDecoration extends RecyclerView.ItemDecoration {

    private static final int[] ATTRS = new int[]{
            android.R.attr.listDivider
    };

    public static final int HORIZONTAL_LIST = LinearLayoutManager.HORIZONTAL;

    public static final int VERTICAL_LIST = LinearLayoutManager.VERTICAL;

    private Drawable mDivider;
    private int mOrientation;
    private Context context;
    private int margin;

    public MyDividerItemDecoration(Context context, int orientation, int margin) {
        this.context = context;
        this.margin = margin;
        final TypedArray a = context.obtainStyledAttributes(ATTRS);
        mDivider = a.getDrawable(0);
        a.recycle();
        setOrientation(orientation);
    }

    public void setOrientation(int orientation) {
        if (orientation != HORIZONTAL_LIST && orientation != VERTICAL_LIST) {
            throw new IllegalArgumentException("invalid orientation");
        }
        mOrientation = orientation;
    }

    @Override
    public void onDrawOver(Canvas c, RecyclerView parent, RecyclerView.State state) {
        if (mOrientation == VERTICAL_LIST) {
            drawVertical(c, parent);
        } else {
            drawHorizontal(c, parent);
        }
    }

    public void drawVertical(Canvas c, RecyclerView parent) {
        final int left = parent.getPaddingLeft();
        final int right = parent.getWidth() - parent.getPaddingRight();

        final int childCount = parent.getChildCount();
        for (int i = 0; i < childCount; i++) {
```

Archivo - MyDividerItemDecoration.java

```
64              final View child = parent.getChildAt(i);
65              final RecyclerView.LayoutParams params = (RecyclerView.LayoutParams) child
66                      .getLayoutParams();
67              final int top = child.getBottom() + params.bottomMargin;
68              final int bottom = top + mDivider.getIntrinsicHeight();
69              mDivider.setBounds(left + dpToPx(margin), top, right - dpToPx(margin), bottom);
70              mDivider.draw(c);
71          }
72      }
73
74      public void drawHorizontal(Canvas c, RecyclerView parent) {
75          final int top = parent.getPaddingTop();
76          final int bottom = parent.getHeight() - parent.getPaddingBottom();
77
78          final int childCount = parent.getChildCount();
79          for (int i = 0; i < childCount; i++) {
80              final View child = parent.getChildAt(i);
81              final RecyclerView.LayoutParams params = (RecyclerView.LayoutParams) child
82                      .getLayoutParams();
83              final int left = child.getRight() + params.rightMargin;
84              final int right = left + mDivider.getIntrinsicHeight();
85              mDivider.setBounds(left, top + dpToPx(margin), right, bottom - dpToPx(margin));
86              mDivider.draw(c);
87          }
88      }
89
90      @Override
91      public void getItemOffsets(Rect outRect, View view, RecyclerView parent, RecyclerView.
    State state) {
92          if (mOrientation == VERTICAL_LIST) {
93              outRect.set(0, 0, 0, mDivider.getIntrinsicHeight());
94          } else {
95              outRect.set(0, 0, mDivider.getIntrinsicWidth(), 0);
96          }
97      }
98
99      private int dpToPx(int dp) {
100         Resources r = context.getResources();
101         return Math.round(TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP, dp, r.
    getDisplayMetrics()));
102     }
103 }
```

Archivo - CommandsSettingsActivity.java

```java
package com.example.SAPLM.settingsActivities;

import android.graphics.PorterDuff;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import androidx.core.content.ContextCompat;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import android.view.MenuItem;

import com.example.SAPLM.Application;
import com.example.SAPLM.R;

public class CommandsSettingsActivity extends AppCompatActivity {
    private static Toolbar toolbar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.commands_settings_layout);
        Application.setActivity(this);
        Application.setContext(this);

        toolbar = this.findViewById(R.id.toolbar);

        setSupportActionBar(toolbar);

        // add back arrow to toolbar
        if (getSupportActionBar() != null){
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
            getSupportActionBar().setDisplayShowHomeEnabled(true);

            final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.abc_ic_ab_back_material);
            upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP);
            getSupportActionBar().setHomeAsUpIndicator(upArrow);
        }
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == android.R.id.home) // Press Back Icon
        {
            finish();
        }

        return super.onOptionsItemSelected(item);
    }
}

```

Archivo - DeveloperConsoleActivity.java

```java
package com.example.SAPLM.settingsActivities;

import android.graphics.Color;
import android.graphics.PorterDuff;
import android.graphics.Typeface;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import androidx.core.content.ContextCompat;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

import android.os.Looper;
import android.text.Html;
import android.text.Spanned;
import android.text.SpannedString;
import android.util.TypedValue;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.LinearLayout;
import android.widget.ScrollView;
import android.widget.TextView;

import com.example.SAPLM.Application;
import com.example.SAPLM.R;

public class DeveloperConsoleActivity extends AppCompatActivity {
    private static Toolbar toolbar;
    private static ImageButton enterButton;
    private static ScrollView scrollViewTerminal;
    private static EditText editTextTerminal;
    private static LinearLayout terminalLinearLayout;
    private static TextView mainTextView;

    private static final String mainText_KEY = "MAIN_TEXT_KEY";
    private static final String ongoingText_KEY = "ONGOING_TEXT_KEY";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.developer_console_layout);
        Application.setActivity(this);
        Application.setContext(this);

        toolbar = this.findViewById(R.id.toolbar);
        enterButton = this.findViewById(R.id.enterButton);
        scrollViewTerminal = this.findViewById(R.id.scrollViewTerminal);
        editTextTerminal = this.findViewById(R.id.editTextTerminal);
        terminalLinearLayout = this.findViewById(R.id.terminalLinearLayout);




        setSupportActionBar(toolbar);

        // add back arrow to toolbar
        if (getSupportActionBar() != null){
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
            getSupportActionBar().setDisplayShowHomeEnabled(true);

            final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
abc_ic_ab_back_material);
```

```
63              upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP
    );
64              getSupportActionBar().setHomeAsUpIndicator(upArrow);
65          }
66
67          mainTextView = new TextView(this);
68          mainTextView.setHint("SAPLM TERMINAL");
69          mainTextView.setTextColor(Color.BLACK);
70          mainTextView.setTextSize(TypedValue.COMPLEX_UNIT_SP, 54.0f /getResources().
    getDisplayMetrics().density);
71          mainTextView.setLayoutParams(new LinearLayout.LayoutParams(LinearLayout.
    LayoutParams.WRAP_CONTENT, LinearLayout.LayoutParams.WRAP_CONTENT));
72          mainTextView.setTypeface(Typeface.MONOSPACE);
73          terminalLinearLayout.addView(mainTextView);
74
75
76
77          scrollViewTerminal.fullScroll(View.FOCUS_DOWN);
78
79      }
80
81
82
83      @Override
84      protected void onPause() {
85          if(mainTextView!=null) {
86              try {
87                  Application.persistentDataSave.putString(ongoingText_KEY, editTextTerminal.
    getText().toString());
88              }catch (Throwable th){}
89          }
90          super.onPause();
91      }
92
93      @Override
94      protected void onResume() {
95          if (Application.persistentDataSave.containsKey(mainText_KEY)&&mainTextView!=null) {
96              try {
97                  mainTextView.setText(Html.fromHtml(Application.persistentDataSave.getString
    (mainText_KEY), Html.FROM_HTML_MODE_LEGACY), TextView.BufferType.SPANNABLE);
98                  editTextTerminal.setText(Application.persistentDataSave.getString(
    ongoingText_KEY));
99                  scrollViewTerminal.fullScroll(View.FOCUS_DOWN);
100             }catch (Throwable th){}
101         }
102         super.onResume();
103     }
104
105     @Override
106     public boolean onOptionsItemSelected(MenuItem item) {
107         if (item.getItemId() == android.R.id.home) // Press Back Icon
108         {
109             finish();
110         }
111
112         return super.onOptionsItemSelected(item);
113     }
114
115     public static void sendTextToTerminal(Spanned newText){
116
117         String newString = Application.persistentDataSave.getString(mainText_KEY) + Html.
    toHtml(newText, Html.FROM_HTML_MODE_LEGACY);
118         Application.persistentDataSave.putString(mainText_KEY,newString);
```

```
119
120         if (Looper.myLooper() == Looper.getMainLooper()){
121             mainTextView.setText(Html.fromHtml(newString, Html.FROM_HTML_MODE_LEGACY),
    TextView.BufferType.SPANNABLE);
122             scrollViewTerminal.fullScroll(View.FOCUS_DOWN);
123         } else {
124             Application.getActivity().runOnUiThread(new Runnable() {
125                 @Override
126                 public void run() {
127                     mainTextView.setText(Html.fromHtml(Application.persistentDataSave.
    getString(mainText_KEY), Html.FROM_HTML_MODE_LEGACY), TextView.BufferType.SPANNABLE);
128                     scrollViewTerminal.fullScroll(View.FOCUS_DOWN);
129                 }
130             });
131         }
132     }
133
134     public static void sendTextToTerminal(String newText){
135
136         Spanned spannedConverted = Html.fromHtml(newText,Html.FROM_HTML_MODE_LEGACY);
137
138         String newString = Application.persistentDataSave.getString(mainText_KEY) + Html.
    toHtml(spannedConverted, Html.FROM_HTML_MODE_LEGACY);
139         Application.persistentDataSave.putString(mainText_KEY,newString);
140
141         if (mainTextView!=null) {
142             if (Looper.myLooper() == Looper.getMainLooper()) {
143                 mainTextView.setText(Html.fromHtml(newString, Html.FROM_HTML_MODE_LEGACY),
    TextView.BufferType.SPANNABLE);
144                 scrollViewTerminal.fullScroll(View.FOCUS_DOWN);
145             } else {
146                 Application.getActivity().runOnUiThread(new Runnable() {
147                     @Override
148                     public void run() {
149                         mainTextView.setText(Html.fromHtml(Application.persistentDataSave.
    getString(mainText_KEY), Html.FROM_HTML_MODE_LEGACY), TextView.BufferType.SPANNABLE);
150                         scrollViewTerminal.fullScroll(View.FOCUS_DOWN);
151                     }
152                 });
153             }
154         }
155     }
156
157
158
159     public static void clearTerminalText(){
160         Application.persistentDataSave.putString(mainText_KEY,"");
161
162         if (Looper.myLooper() == Looper.getMainLooper()){
163             mainTextView.setText("");
164             scrollViewTerminal.fullScroll(View.FOCUS_DOWN);
165         } else {
166             Application.getActivity().runOnUiThread(new Runnable() {
167                 @Override
168                 public void run() {
169                     mainTextView.setText("");
170                     scrollViewTerminal.fullScroll(View.FOCUS_DOWN);
171                 }
172             });
173         }
174     }
175
176
```

Archivo - DeveloperConsoleActivity.java

```
177     public void onClickEnterButton(View view){
178         sendTextToTerminal(Html.fromHtml(editTextTerminal.getText().toString(),Html.
    FROM_HTML_MODE_LEGACY));
179         TerminalManager.handleTerminalInput(editTextTerminal.getText().toString());
180     }
181 
182     public void onClickClearButton(View view){
183         clearTerminalText();
184     }
185 
186 
187 }
188 
```

Archivo - BluetoothSettingsActivity.java

```java
package com.example.SAPLM.settingsActivities;

import android.graphics.PorterDuff;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import androidx.core.content.ContextCompat;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import android.view.MenuItem;

import com.example.SAPLM.Application;
import com.example.SAPLM.R;

public class BluetoothSettingsActivity extends AppCompatActivity {
    private static Toolbar toolbar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.bluetooth_settings_layout);
        Application.setActivity(this);
        Application.setContext(this);

        toolbar = this.findViewById(R.id.toolbar);

        setSupportActionBar(toolbar);

        // add back arrow to toolbar
        if (getSupportActionBar() != null){
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
            getSupportActionBar().setDisplayShowHomeEnabled(true);

            final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
abc_ic_ab_back_material);
            upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
;
            getSupportActionBar().setHomeAsUpIndicator(upArrow);
        }
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == android.R.id.home) // Press Back Icon
        {
            finish();
        }

        return super.onOptionsItemSelected(item);
    }
}

```

Archivo - InternetConectivityActivity.java

```java
package com.example.SAPLM.settingsActivities;

import android.graphics.PorterDuff;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import androidx.core.content.ContextCompat;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import android.view.MenuItem;

import com.example.SAPLM.Application;
import com.example.SAPLM.R;

public class InternetConectivityActivity extends AppCompatActivity {
    private static Toolbar toolbar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.internet_conectivity_layout);
        Application.setActivity(this);
        Application.setContext(this);

        toolbar = this.findViewById(R.id.toolbar);

        setSupportActionBar(toolbar);

        // add back arrow to toolbar
        if (getSupportActionBar() != null){
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
            getSupportActionBar().setDisplayShowHomeEnabled(true);

            final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
abc_ic_ab_back_material);
            upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
;
            getSupportActionBar().setHomeAsUpIndicator(upArrow);
        }
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == android.R.id.home) // Press Back Icon
        {
            finish();
        }

        return super.onOptionsItemSelected(item);
    }
}

```

Archivo - ContactsCommunicationActivity.java

```java
package com.example.SAPLM.settingsActivities;

import android.graphics.PorterDuff;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import androidx.core.content.ContextCompat;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import android.view.MenuItem;

import com.example.SAPLM.Application;
import com.example.SAPLM.R;

public class ContactsCommunicationActivity extends AppCompatActivity {
    private static Toolbar toolbar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.contacts_communication_layout);
        Application.setActivity(this);
        Application.setContext(this);

        toolbar = this.findViewById(R.id.toolbar);

        setSupportActionBar(toolbar);

        // add back arrow to toolbar
        if (getSupportActionBar() != null){
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
            getSupportActionBar().setDisplayShowHomeEnabled(true);

            final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
abc_ic_ab_back_material);
            upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
;
            getSupportActionBar().setHomeAsUpIndicator(upArrow);
        }
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == android.R.id.home) // Press Back Icon
        {
            finish();
        }

        return super.onOptionsItemSelected(item);
    }
}

```

Archivo - AppearanceMiscellaneousActivity.java

```java
package com.example.SAPLM.settingsActivities;

import android.graphics.PorterDuff;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import androidx.core.content.ContextCompat;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import android.view.MenuItem;

import com.example.SAPLM.Application;
import com.example.SAPLM.R;

public class AppearanceMiscellaneousActivity extends AppCompatActivity {

    private static Toolbar toolbar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.appearance_miscellaneous_layout);
        Application.setActivity(this);
        Application.setContext(this);

        toolbar = this.findViewById(R.id.toolbar);




        setSupportActionBar(toolbar);

        // add back arrow to toolbar
        if (getSupportActionBar() != null){
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
            getSupportActionBar().setDisplayShowHomeEnabled(true);

            final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
abc_ic_ab_back_material);
            upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
;
            getSupportActionBar().setHomeAsUpIndicator(upArrow);
        }
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == android.R.id.home) // Press Back Icon
        {
            finish();
        }

        return super.onOptionsItemSelected(item);
    }
}

```

Archivo - VoiceRecognitionSettingsActivity.java

```java
package com.example.SAPLM.settingsActivities;

import android.content.DialogInterface;
import android.graphics.PorterDuff;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.speech.tts.Voice;
import androidx.core.content.ContextCompat;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ProgressBar;

import com.example.SAPLM.Application;
import com.example.SAPLM.R;
import com.example.SAPLM.TextToSpeechHandler;
import com.github.ybq.android.spinkit.sprite.Sprite;
import com.github.ybq.android.spinkit.style.FadingCircle;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.Locale;

public class VoiceRecognitionSettingsActivity extends AppCompatActivity {
    private static Toolbar toolbar;

    private Button speech_test_btn;
    private Button set_voice_btn;
    private EditText phrase_editText;
    private static ProgressBar progressBar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.voice_recognition_settings_layout);
        Application.setActivity(this);
        Application.setContext(this);

        toolbar = this.findViewById(R.id.toolbar);
        speech_test_btn = this.findViewById(R.id.speech_test_btn);
        set_voice_btn = this.findViewById(R.id.set_voice_btn);
        phrase_editText = this.findViewById(R.id.phrase_editText);

        setSupportActionBar(toolbar);

        // add back arrow to toolbar
        if (getSupportActionBar() != null){
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
            getSupportActionBar().setDisplayShowHomeEnabled(true);

            final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
abc_ic_ab_back_material);
            upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
;
            getSupportActionBar().setHomeAsUpIndicator(upArrow);
        }

```

```
62          TextToSpeechHandler.initializeTextToSpeech();
63
64
65          progressBar = (ProgressBar)findViewById(R.id.progressBar);
66          Sprite fadingCircle = new FadingCircle();
67          fadingCircle.setColor(R.color.colorPrimary);
68          progressBar.setIndeterminateDrawable(fadingCircle);
69
70
71      }
72
73      @Override
74      public boolean onOptionsItemSelected(MenuItem item) {
75          if (item.getItemId() == android.R.id.home) // Press Back Icon
76          {
77              finish();
78          }
79
80          return super.onOptionsItemSelected(item);
81      }
82
83      public void speechTestOnClick(View view){
84          if (TextToSpeechHandler.isInitialized){
85              //TextToSpeechHandler.addToSpeakingQueue(phrase_editText.getText().toString());
86
87              Date currentTime = Calendar.getInstance().getTime();
88              SimpleDateFormat df = new SimpleDateFormat("d-MMMM-yyyy", new Locale("spa"));
89              String todayAsString = df.format(currentTime);
90              SimpleDateFormat onlyTimeFormar = new SimpleDateFormat("HH:mm:ss", new Locale("
    spa"));
91              String onlyTimeString = onlyTimeFormar.format(currentTime);
92
93              TextToSpeechHandler.addToSpeakingQueue("Hoy es " + todayAsString + "y son las"
    + onlyTimeString);
94
95          }else{
96              Application.toastMessage("Not initialized");
97          }
98      }
99
100     public static ArrayList<String> voiceName= new ArrayList<>();
101     public static Voice[] voiceArray;
102
103     public void setVoiceOnClick(View view){
104         AlertDialog.Builder builder = new AlertDialog.Builder(Application.getContext());
105
106         voiceArray = TextToSpeechHandler.getVoicesList().toArray(new Voice[
    TextToSpeechHandler.getVoicesList().size()]);
107
108         for(Voice voice : voiceArray){
109             voiceName.add(voice.getName());
110         }
111
112         builder.setTitle("Seleccionar voz");
113         builder.setItems(voiceName.toArray(new String[voiceName.size()]), new
    DialogInterface.OnClickListener() {
114             @Override
115             public void onClick(DialogInterface dialog, int which) {
116                 TextToSpeechHandler.setVoice(voiceArray[which]);
117             }
118         });
119         builder.show();
120     }
```

```
121
122     @Override
123     protected void onPause() {
124         if (TextToSpeechHandler.isInitialized){
125             //TextToSpeechHandler.shutdownTextSpeech();
126         }
127
128         super.onPause();
129     }
130 }
131
```

Archivo - CRC8_CCITT.java

```
 1 package com.example.SAPLM.bluetoothActivities;
 2
 3
 4 public class CRC8_CCITT {
 5     public static byte[] crc_table = {
 6             (byte) 0x00, (byte) 0x07, (byte) 0x0e, (byte) 0x09, (byte) 0x1c, (byte) 0x1b, (
   byte) 0x12, (byte) 0x15, (byte) 0x38, (byte) 0x3f, (byte) 0x36, (byte) 0x31, (byte) 0x24, (
   byte) 0x23, (byte) 0x2a, (byte) 0x2d,
 7             (byte) 0x70, (byte) 0x77, (byte) 0x7e, (byte) 0x79, (byte) 0x6c, (byte) 0x6b, (
   byte) 0x62, (byte) 0x65, (byte) 0x48, (byte) 0x4f, (byte) 0x46, (byte) 0x41, (byte) 0x54, (
   byte) 0x53, (byte) 0x5a, (byte) 0x5d,
 8             (byte) 0xe0, (byte) 0xe7, (byte) 0xee, (byte) 0xe9, (byte) 0xfc, (byte) 0xfb, (
   byte) 0xf2, (byte) 0xf5, (byte) 0xd8, (byte) 0xdf, (byte) 0xd6, (byte) 0xd1, (byte) 0xc4, (
   byte) 0xc3, (byte) 0xca, (byte) 0xcd,
 9             (byte) 0x90, (byte) 0x97, (byte) 0x9e, (byte) 0x99, (byte) 0x8c, (byte) 0x8b, (
   byte) 0x82, (byte) 0x85, (byte) 0xa8, (byte) 0xaf, (byte) 0xa6, (byte) 0xa1, (byte) 0xb4, (
   byte) 0xb3, (byte) 0xba, (byte) 0xbd,
10             (byte) 0xc7, (byte) 0xc0, (byte) 0xc9, (byte) 0xce, (byte) 0xdb, (byte) 0xdc, (
   byte) 0xd5, (byte) 0xd2, (byte) 0xff, (byte) 0xf8, (byte) 0xf1, (byte) 0xf6, (byte) 0xe3, (
   byte) 0xe4, (byte) 0xed, (byte) 0xea,
11             (byte) 0xb7, (byte) 0xb0, (byte) 0xb9, (byte) 0xbe, (byte) 0xab, (byte) 0xac, (
   byte) 0xa5, (byte) 0xa2, (byte) 0x8f, (byte) 0x88, (byte) 0x81, (byte) 0x86, (byte) 0x93, (
   byte) 0x94, (byte) 0x9d, (byte) 0x9a,
12             (byte) 0x27, (byte) 0x20, (byte) 0x29, (byte) 0x2e, (byte) 0x3b, (byte) 0x3c, (
   byte) 0x35, (byte) 0x32, (byte) 0x1f, (byte) 0x18, (byte) 0x11, (byte) 0x16, (byte) 0x03, (
   byte) 0x04, (byte) 0x0d, (byte) 0x0a,
13             (byte) 0x57, (byte) 0x50, (byte) 0x59, (byte) 0x5e, (byte) 0x4b, (byte) 0x4c, (
   byte) 0x45, (byte) 0x42, (byte) 0x6f, (byte) 0x68, (byte) 0x61, (byte) 0x66, (byte) 0x73, (
   byte) 0x74, (byte) 0x7d, (byte) 0x7a,
14             (byte) 0x89, (byte) 0x8e, (byte) 0x87, (byte) 0x80, (byte) 0x95, (byte) 0x92, (
   byte) 0x9b, (byte) 0x9c, (byte) 0xb1, (byte) 0xb6, (byte) 0xbf, (byte) 0xb8, (byte) 0xad, (
   byte) 0xaa, (byte) 0xa3, (byte) 0xa4,
15             (byte) 0xf9, (byte) 0xfe, (byte) 0xf7, (byte) 0xf0, (byte) 0xe5, (byte) 0xe2, (
   byte) 0xeb, (byte) 0xec, (byte) 0xc1, (byte) 0xc6, (byte) 0xcf, (byte) 0xc8, (byte) 0xdd, (
   byte) 0xda, (byte) 0xd3, (byte) 0xd4,
16             (byte) 0x69, (byte) 0x6e, (byte) 0x67, (byte) 0x60, (byte) 0x75, (byte) 0x72, (
   byte) 0x7b, (byte) 0x7c, (byte) 0x51, (byte) 0x56, (byte) 0x5f, (byte) 0x58, (byte) 0x4d, (
   byte) 0x4a, (byte) 0x43, (byte) 0x44,
17             (byte) 0x19, (byte) 0x1e, (byte) 0x17, (byte) 0x10, (byte) 0x05, (byte) 0x02, (
   byte) 0x0b, (byte) 0x0c, (byte) 0x21, (byte) 0x26, (byte) 0x2f, (byte) 0x28, (byte) 0x3d, (
   byte) 0x3a, (byte) 0x33, (byte) 0x34,
18             (byte) 0x4e, (byte) 0x49, (byte) 0x40, (byte) 0x47, (byte) 0x52, (byte) 0x55, (
   byte) 0x5c, (byte) 0x5b, (byte) 0x76, (byte) 0x71, (byte) 0x78, (byte) 0x7f, (byte) 0x6a, (
   byte) 0x6d, (byte) 0x64, (byte) 0x63,
19             (byte) 0x3e, (byte) 0x39, (byte) 0x30, (byte) 0x37, (byte) 0x22, (byte) 0x25, (
   byte) 0x2c, (byte) 0x2b, (byte) 0x06, (byte) 0x01, (byte) 0x08, (byte) 0x0f, (byte) 0x1a, (
   byte) 0x1d, (byte) 0x14, (byte) 0x13,
20             (byte) 0xae, (byte) 0xa9, (byte) 0xa0, (byte) 0xa7, (byte) 0xb2, (byte) 0xb5, (
   byte) 0xbc, (byte) 0xbb, (byte) 0x96, (byte) 0x91, (byte) 0x98, (byte) 0x9f, (byte) 0x8a, (
   byte) 0x8d, (byte) 0x84, (byte) 0x83,
21             (byte) 0xde, (byte) 0xd9, (byte) 0xd0, (byte) 0xd7, (byte) 0xc2, (byte) 0xc5, (
   byte) 0xcc, (byte) 0xcb, (byte) 0xe6, (byte) 0xe1, (byte) 0xe8, (byte) 0xef, (byte) 0xfa, (
   byte) 0xfd, (byte) 0xf4, (byte) 0xf3
22     };
23
24     public static byte crc_CCITT(byte[] data, int data_len)
25     {
26         byte crc = 0;
27         byte[] d = data;
28         int dataIndex = 0;
29         byte tbl_idx = 0;
30
31         while (data_len!=0) {
```

```
32              tbl_idx = (byte) (crc ^ d[dataIndex]);
33              crc = (byte) (crc_table[tbl_idx & 0xFF] & (byte)0xff);
34              dataIndex++;
35              data_len--;
36          }
37          return (byte) (crc & (byte)0xff);
38      }
39  }
40
```

Archivo - AvailableDevice.java

```java
package com.example.SAPLM.bluetoothActivities;

import com.example.SAPLM.R;

public class AvailableDevice {

    public enum DeviceState { DEFAULT, DISCONNECTED, CONNECTED, LOADING}
    private String deviceName;
    private DeviceState currentState;
    private int deviceStateIconID;

    public AvailableDevice() {
    }

    public AvailableDevice(String deviceName, DeviceState defaultState) {
        this.deviceName = deviceName;
        this.currentState = defaultState;
    }

    public void setCurrentState(DeviceState currentState) {
        this.currentState = currentState;
    }

    public DeviceState getCurrentState() {
        return currentState;
    }

    public void setDeviceName(String deviceName) {
        this.deviceName = deviceName;
    }

    public String getDeviceName() {
        return deviceName;
    }

    public int getDeviceStateIconID() {
        switch (currentState){
            case DEFAULT:
                return 0;
            case CONNECTED:
                return R.drawable.ic_done_icon;
            case LOADING:
                return R.drawable.loading_animation;
            case DISCONNECTED:
                return R.drawable.ic_close_icon;
        }
        return 0;
    }

    public int getTextAppearance(){
        switch (currentState){
            case DEFAULT:
                return R.style.btDeviceListAppearance;
            case CONNECTED:
                return R.style.btDeviceListSelectedAppearance;
            case LOADING:
                return R.style.btDeviceListAppearance;
            case DISCONNECTED:
                return R.style.btDeviceListAppearance;
        }
        return R.style.btDeviceListAppearance;
    }
}
```

Archivo - BluetoothActivity.java

```java
package com.example.SAPLM.bluetoothActivities;

import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.graphics.PorterDuff;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.os.Looper;
import androidx.core.content.ContextCompat;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.DefaultItemAnimator;
import androidx.recyclerview.widget.DividerItemDecoration;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import androidx.appcompat.widget.Toolbar;
import android.view.MenuItem;
import android.view.View;
import android.widget.ImageView;
import android.widget.ProgressBar;
import android.widget.TextView;

import com.example.SAPLM.Application;
import com.example.SAPLM.R;
import com.example.SAPLM.settingsActivities.BluetoothSettingsActivity;
import com.example.SAPLM.settingsActivities.RecyclerTouchListener;
import com.github.ybq.android.spinkit.sprite.Sprite;
import com.github.ybq.android.spinkit.style.Circle;

import java.util.ArrayList;
import java.util.List;

public class BluetoothActivity extends AppCompatActivity {
    private static Toolbar toolbar;


    public static List<AvailableDevice> deviceList = new ArrayList<>();
    private RecyclerView recyclerView;
    public static DeviceListAdapter mAdapter;

    public static ImageView imgView_bluetoothStatusIcon;
    public static TextView textView_bluetoothState;
    public static ProgressBar loading_progressBar;

    public static Boolean guiInitialized = false;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.bluetooth_layout);
        Application.setActivity(this);
        Application.setContext(this);

        toolbar = this.findViewById(R.id.toolbar);
        textView_bluetoothState = this.findViewById(R.id.textView_btStatus);
        imgView_bluetoothStatusIcon = this.findViewById(R.id.imgView_btStatusIcon);
        loading_progressBar = this.findViewById(R.id.loading_progressBar);

        Sprite fadingCircle = new Circle();
        fadingCircle.setColor(Color.WHITE);
        loading_progressBar.setIndeterminateDrawable(fadingCircle);

```

```
64          setSupportActionBar(toolbar);
65
66          // add back arrow to toolbar
67          if (getSupportActionBar() != null){
68              getSupportActionBar().setDisplayHomeAsUpEnabled(true);
69              getSupportActionBar().setDisplayShowHomeEnabled(true);
70
71              final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
    abc_ic_ab_back_material);
72              upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP
    );
73              getSupportActionBar().setHomeAsUpIndicator(upArrow);
74          }
75
76          recyclerView = (RecyclerView) findViewById(R.id.deviceListRecyclerView);
77
78          mAdapter = new DeviceListAdapter(deviceList);
79
80          recyclerView.setHasFixedSize(true);
81
82          // vertical RecyclerView
83          // keep movie_list_row.xml width to `match_parent`
84          RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(
    getApplicationContext());
85
86          // horizontal RecyclerView
87          // keep movie_list_row.xml width to `wrap_content`
88          // RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(
    getApplicationContext(), LinearLayoutManager.HORIZONTAL, false);
89
90          recyclerView.setLayoutManager(mLayoutManager);
91
92          // adding inbuilt divider line
93          recyclerView.addItemDecoration(new DividerItemDecoration(this, LinearLayoutManager.
    VERTICAL));
94
95          // adding custom divider line with padding 16dp
96          //recyclerView.addItemDecoration(new MyDividerItemDecoration(this,
    LinearLayoutManager.HORIZONTAL, 16));
97          recyclerView.setItemAnimator(new DefaultItemAnimator());
98
99          recyclerView.setAdapter(mAdapter);
100
101         // row click listener
102         recyclerView.addOnItemTouchListener(new RecyclerTouchListener(getApplicationContext
    (), recyclerView, new RecyclerTouchListener.ClickListener() {
103             @Override
104             public void onClick(View view, int position) {
105                 AvailableDevice device = deviceList.get(position);
106
107                 BluetoothConnection.connectBluetoothDeviceAsync(position);
108             }
109
110             @Override
111             public void onLongClick(View view, int position) {
112
113             }
114         }));
115
116         guiInitialized = true;
117
118         BluetoothConnection.listAvailableBluetoothDevices();
119
```

```
120     }
121
122     @Override
123     public boolean onOptionsItemSelected(MenuItem item) {
124         if (item.getItemId() == android.R.id.home) // Press Back Icon
125         {
126             finish();
127         }
128
129         return super.onOptionsItemSelected(item);
130     }
131
132     public static void updateDevicesList(){
133         if (guiInitialized) {
134             Application.getActivity().runOnUiThread(new Runnable() {
135                 @Override
136                 public void run() {
137                     BluetoothActivity.mAdapter.notifyDataSetChanged();
138                 }
139             });
140         }
141     }
142
143     public void configBtn_onClick(View view){
144         Context currentContext = Application.getContext();
145         Intent i = new Intent(currentContext, BluetoothSettingsActivity.class);
146         currentContext.startActivity(i);
147     }
148
149
150     public enum bluetoothConnectionState { SYSTEM_DISCONNECTED, SYSTEM_CONNECTED,
    BLUETOOTH_DISABLED, WAITING, BT_TURNING_OFF, BT_TURNING_ON, UNDEFINED}
151     public static bluetoothConnectionState currentBluetoothState = bluetoothConnectionState
    .UNDEFINED;
152
153     public static void setBluetoothStateGui(bluetoothConnectionState state){
154         if (guiInitialized) {
155             currentBluetoothState = state;
156
157             if (Looper.myLooper() == Looper.getMainLooper()) {
158                 switch (currentBluetoothState) {
159                     case SYSTEM_CONNECTED:
160                         textView_bluetoothState.setText(R.string.system_connected_message);
161                         imgView_bluetoothStatusIcon.setImageResource(R.drawable.
    ic_done_icon);
162                         imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
163                         loading_progressBar.setVisibility(View.GONE);
164                         break;
165                     case BLUETOOTH_DISABLED:
166                         textView_bluetoothState.setText(R.string.bluetooth_disabled_message
    );
167                         imgView_bluetoothStatusIcon.setImageResource(R.drawable.
    ic_bluetooth_disabled);
168                         imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
169                         loading_progressBar.setVisibility(View.GONE);
170                         break;
171                     case BT_TURNING_ON:
172                         textView_bluetoothState.setText(R.string.
    bluetooth_initializing_adapater);
173                         imgView_bluetoothStatusIcon.setVisibility(View.GONE);
174                         loading_progressBar.setVisibility(View.VISIBLE);
175                         break;
176                     case BT_TURNING_OFF:
```

```
177                         textView_bluetoothState.setText(R.string.bluetooth_closing_adapter)
    ;
178                         imgView_bluetoothStatusIcon.setVisibility(View.GONE);
179                         loading_progressBar.setVisibility(View.VISIBLE);
180                         break;
181                     case SYSTEM_DISCONNECTED:
182                         textView_bluetoothState.setText(R.string.
    system_disconnected_message);
183                         imgView_bluetoothStatusIcon.setImageResource(R.drawable.
    ic_close_icon);
184                         imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
185                         loading_progressBar.setVisibility(View.GONE);
186                         break;
187                     case WAITING:
188                         textView_bluetoothState.setText(R.string.
    bluetooth_attempting_connection_message);
189                         imgView_bluetoothStatusIcon.setVisibility(View.GONE);
190                         loading_progressBar.setVisibility(View.VISIBLE);
191                         break;
192                 }
193             } else {
194                 Application.getActivity().runOnUiThread(new Runnable() {
195                     @Override
196                     public void run() {
197                         switch (currentBluetoothState) {
198                             case SYSTEM_CONNECTED:
199                                 textView_bluetoothState.setText(R.string.
    system_connected_message);
200                                 imgView_bluetoothStatusIcon.setImageResource(R.drawable.
    ic_done_icon);
201                                 imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
202                                 loading_progressBar.setVisibility(View.GONE);
203                                 break;
204                             case BLUETOOTH_DISABLED:
205                                 textView_bluetoothState.setText(R.string.
    bluetooth_disabled_message);
206                                 imgView_bluetoothStatusIcon.setImageResource(R.drawable.
    ic_bluetooth_disabled);
207                                 imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
208                                 loading_progressBar.setVisibility(View.GONE);
209                                 break;
210                             case BT_TURNING_ON:
211                                 textView_bluetoothState.setText(R.string.
    bluetooth_initializing_adapater);
212                                 imgView_bluetoothStatusIcon.setVisibility(View.GONE);
213                                 loading_progressBar.setVisibility(View.VISIBLE);
214                                 break;
215                             case BT_TURNING_OFF:
216                                 textView_bluetoothState.setText(R.string.
    bluetooth_closing_adapter);
217                                 imgView_bluetoothStatusIcon.setVisibility(View.GONE);
218                                 loading_progressBar.setVisibility(View.GONE);
219                                 break;
220                             case SYSTEM_DISCONNECTED:
221                                 textView_bluetoothState.setText(R.string.
    system_disconnected_message);
222                                 imgView_bluetoothStatusIcon.setImageResource(R.drawable.
    ic_close_icon);
223                                 imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
224                                 loading_progressBar.setVisibility(View.GONE);
225                                 break;
226                             case WAITING:
227                                 textView_bluetoothState.setText(R.string.
```

Archivo - BluetoothActivity.java

```
227 bluetooth_attempting_connection_message);
228                                 imgView_bluetoothStatusIcon.setVisibility(View.GONE);
229                                 loading_progressBar.setVisibility(View.VISIBLE);
230                                 break;
231                         }
232                     }
233                 });
234             }
235         }
236     }
237
238
239 }
240
```

Archivo - DeviceListAdapter.java

```java
package com.example.SAPLM.bluetoothActivities;

import android.graphics.Color;
import android.graphics.PorterDuff;
import android.graphics.drawable.ColorDrawable;
import android.graphics.drawable.Drawable;
import androidx.recyclerview.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import com.example.SAPLM.Application;
import com.example.SAPLM.R;

import java.util.List;

public class DeviceListAdapter extends RecyclerView.Adapter<DeviceListAdapter.viewHolder> {

    private List<AvailableDevice> deviceList;

    public class viewHolder extends RecyclerView.ViewHolder {
        public TextView textView_device_name;
        public ImageView imgView_device_state_icon;

        public viewHolder(View view) {
            super(view);
            textView_device_name = (TextView) view.findViewById(R.id.
textView_bluetooth_device_name);
            imgView_device_state_icon = (ImageView) view.findViewById(R.id.
imageView_bluetooth_device_state_icon);
        }
    }


    public DeviceListAdapter(List<AvailableDevice> deviceList) {
        this.deviceList = deviceList;
    }

    @Override
    public viewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View itemView = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.bluetooth_list_layout_style, parent, false);

        return new viewHolder(itemView);
    }

    @Override
    public void onBindViewHolder(viewHolder holder, int position) {
        AvailableDevice device = deviceList.get(position);
        holder.textView_device_name.setText(device.getDeviceName());
        holder.textView_device_name.setTextAppearance(device.getTextAppearance());

        int iconID = device.getDeviceStateIconID();

        if (iconID!=0) {
            Drawable icon = Application.getContext().getDrawable(iconID);
            icon.setColorFilter(Application.getContext().getColor(R.color.colorPrimary),
PorterDuff.Mode.SRC_ATOP);
            holder.imgView_device_state_icon.setImageDrawable(icon);
        } else {
            holder.imgView_device_state_icon.clearColorFilter();
```

Archivo - DeviceListAdapter.java

```
61              Drawable transparentDrawable = new ColorDrawable(Color.TRANSPARENT);
62              holder.imgView_device_state_icon.setImageDrawable(transparentDrawable);
63          }
64      }
65
66      @Override
67      public int getItemCount() {
68          return deviceList.size();
69      }
70  }
71
```

Archivo - BluetoothConnection.java

```java
package com.example.SAPLM.bluetoothActivities;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothServerSocket;
import android.bluetooth.BluetoothSocket;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.widget.TabHost;

import com.example.SAPLM.Application;
import com.example.SAPLM.settingsActivities.DeveloperConsoleActivity;
import com.example.SAPLM.settingsActivities.TerminalManager;


import java.io.InputStream;
import java.io.OutputStream;
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.List;
import java.util.Set;
import java.util.UUID;
import java.util.concurrent.Executor;
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;
import java.util.concurrent.ScheduledFuture;


public class BluetoothConnection extends BluetoothActivity {

    static ScheduledExecutorService bluetoothThreadPool= Executors.newScheduledThreadPool(1);
    static ScheduledFuture<?> bluetoothComThread;
    public static OutputStream outputStream;
    public static InputStream inputStream;
    public static BluetoothAdapter btAdapter = BluetoothAdapter.getDefaultAdapter();
    public static BluetoothDevice targetDevice;
    public static BluetoothDevice[] bluetoothDevicesArray;
    public static List<BluetoothDevice> bluetoothDevicesList;
    public static BluetoothSocket btSocket;
    public static BluetoothServerSocket btServerSocket;
    public static UUID btUUID = UUID.fromString("00001101-0000-1000-8000-00805f9b34fb");
    public static String btString = "HC-05";
    public static Boolean bluetoothConnected = false;

    private static Executor bluetoothConnectorExecutor = Executors.newSingleThreadExecutor();
    private static Boolean isListAvailableBluetoothDevicesRunning = false;

    public static void listAvailableBluetoothDevices(){
        if ((!isListAvailableBluetoothDevicesRunning)) {
            try {
                BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.WAITING);
                isListAvailableBluetoothDevicesRunning = true;
                bluetoothConnectorExecutor.execute(new Runnable() {
                    @Override
                    public void run() {
                        if (btAdapter != null) {
                            if (btAdapter.isEnabled()) {
                                Set<BluetoothDevice> bondedDevices = btAdapter.getBondedDevices();

```

```
61                                  bluetoothDevicesList = new ArrayList<>();
62                                  bluetoothDevicesList.addAll(bondedDevices);
63
64                                  if (bondedDevices.size() > 0) {
65                                      bluetoothDevicesArray = bondedDevices.toArray(new
   BluetoothDevice[bondedDevices.size()]);
66                                  }
67                                  updateBluetoothListGUI();
68                              } else {
69                                  BluetoothActivity.setBluetoothStateGui(
   bluetoothConnectionState.BLUETOOTH_DISABLED);
70                              }
71                          }
72                          isListAvailableBluetoothDevicesRunning = false;
73                      }
74                  });
75              }catch (Throwable th)
76              {
77                  Application.toastMessage(th.toString()); th.printStackTrace();
78              }
79          }
80      }
81
82      public static Boolean isBluetoothAdapterReady(){
83          return ((btAdapter != null)&&(btAdapter.isEnabled()));
84      }
85
86      public static void updateBluetoothListGUI(){
87          if (BluetoothActivity.mAdapter!=null) {
88              BluetoothActivity.deviceList.clear();
89              for (int x = 0; x < bluetoothDevicesArray.length; x++) {
90                  BluetoothActivity.deviceList.add(new AvailableDevice(bluetoothDevicesArray[
   x].getName(), AvailableDevice.DeviceState.DEFAULT));
91              }
92
93              if (bluetoothConnected) {
94                  BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.
   SYSTEM_CONNECTED);
95                  deviceList.get(bluetoothDevicesList.indexOf(targetDevice)).setCurrentState(
   AvailableDevice.DeviceState.CONNECTED);
96              }
97              else {
98                  BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.
   SYSTEM_DISCONNECTED);
99              }
100
101             BluetoothActivity.updateDevicesList();
102         }
103     }
104
105     private static Boolean attemptingBluetoothConnection = false;
106
107
108     public static void connectBluetoothDeviceSync(int deviceIndex){
109         if ((!isListAvailableBluetoothDevicesRunning)&&(!attemptingBluetoothConnection)&&
   isBluetoothAdapterReady()) {
110             try {
111                 attemptingBluetoothConnection = true;
112                 if (bluetoothConnected) closeAllConnections();
113                 BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.WAITING);
114                 targetDevice = bluetoothDevicesArray[deviceIndex];
115                 btSocket = targetDevice.createRfcommSocketToServiceRecord(btUUID);
116                 btSocket.connect();
```

```
117                 if (btSocket.isConnected()) {
118                     inputStream = btSocket.getInputStream();
119                     outputStream = btSocket.getOutputStream();
120                     bluetoothConnected = true;
121                     BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.
    SYSTEM_CONNECTED);
122                     BluetoothActivity.updateDevicesList();
123                 } else {
124                     bluetoothConnected = false;
125                     BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.
    SYSTEM_DISCONNECTED);
126                 }
127                 attemptingBluetoothConnection = false;
128             } catch (Throwable th) {
129                 bluetoothConnected = false;
130                 Application.toastMessage(th.toString());
131                 BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.
    SYSTEM_DISCONNECTED);
132                 attemptingBluetoothConnection = false;
133             }
134         }
135     }
136 
137     private static int attemptingConnectionDeviceIndex;
138 
139     public static void connectBluetoothDeviceAsync(int deviceIndex){
140         attemptingConnectionDeviceIndex = deviceIndex;
141         if ((!isListAvailableBluetoothDevicesRunning)&&(!attemptingBluetoothConnection)&&
    isBluetoothAdapterReady()) {
142             bluetoothConnectorExecutor.execute(new Runnable() {
143                 @Override
144                 public void run() {
145                     connectBluetoothDeviceSync(attemptingConnectionDeviceIndex);
146                 }
147             });
148         }
149     }
150 
151     public static void sendCommandBuffer(byte[] data){
152         try {
153             outputStream.write(new byte[] {(byte)0x7F, (byte)0x7F, (byte)0x7F, (byte)0x7F})
    ;
154             outputStream.write(data,0, 20);
155         }catch (Throwable th) {
156             Application.toastMessage(th.toString());
157         }
158     }
159 
160     public static byte[] receiveCommandBuffer(int timeout){
161         long startTimeMillis = System.currentTimeMillis();
162         while((System.currentTimeMillis()-startTimeMillis)<500L){
163             try {
164                 if (inputStream.available() >= 32) {
165                     byte[] inputBuffer = new byte[32];
166                     inputStream.read(inputBuffer, 0, 32);
167                     return inputBuffer;
168                 }
169             }catch (Throwable th){
170                 Application.toastMessage(th.toString());
171             }
172         }
173 
174         return null;
```

```
175     }
176
177
178
179     public static class BluetoothReceiverService{
180
181         private static Executor receiverExecutor = Executors.newSingleThreadExecutor();
182         private static Boolean terminate = false;
183         private static Boolean isAutoConnectRunning = false;
184
185         public static void initialize(){
186             if (isAutoConnectRunning==false) {
187                 receiverExecutor.execute(new Runnable() {
188                     @Override
189                     public void run() {
190                         isAutoConnectRunning = true;
191                         while (terminate == false) {
192                             try {
193                                 if (inputStream.available()>= 32) {   //  >= 32
194                                     byte[] inputBuffer = new byte[32];
195                                     inputStream.read(inputBuffer, 0, 32);
196                                     UnifiedTransmissionProtocol.command_buffer =
    inputBuffer;
197                                     DeveloperConsoleActivity.sendTextToTerminal("Received
    from: " + UnifiedTransmissionProtocol.lastMessageTransmitterModule.getModuleName() + " : "
    + TerminalManager.insertSpaces(UnifiedTransmissionProtocol.bytesToHex(inputBuffer), " ", 2)
    );
198                                     if (UnifiedTransmissionProtocol.
    DecomposeMessageFromBuffer()== UnifiedTransmissionProtocol.CommandStatus.
    SUCCESSFUL_DECOMPOSITION) {
199                                         UnifiedTransmissionProtocol.handleReceivedCommand()
    ;
200                                     } else { Application.toastMessage("UNSUCCESSFUL
    DECOMPOSITION");}
201                                 }
202                             } catch (Throwable th) {
203                                 Application.toastMessage(th.toString());
204                             }
205                         }
206                         isAutoConnectRunning = false;
207                     }
208                 });
209             }
210         }
211
212         public static void terminateSync(){
213             terminate = true;
214             while(isAutoConnectRunning==true) { }
215             terminate = false;
216         }
217
218     }
219
220
221
222
223
224
225
226     private static Executor autoConnectExecutor = Executors.newSingleThreadExecutor();
227     private static Boolean terminateAutoConnect = false;
228     private static Boolean isAutoConnectRunning = false;
229
```

```
230     public static void initBTAutoConnectService(){
231         if (isAutoConnectRunning() == false) {
232             autoConnectExecutor.execute(new Runnable() {
233                 @Override
234                 public void run() {
235                     isAutoConnectRunning = true;
236                     try { Thread.sleep(5000);} catch (Throwable th) { th.printStackTrace();
    }
237                     while (!terminateAutoConnect && isBluetoothAdapterReady()) {
238                         listAvailableBluetoothDevices();
239                         while (isListAvailableBluetoothDevicesRunning) {}
240                         for (int x = 0; x < bluetoothDevicesArray.length; x++) {
241                             // MAC ADDRESS HC-05 : 98:D3:91:FD:38:05
242                             // MAC ADDRESS XT1068 : 5C:51:88:A2:38:97
243                             if (bluetoothDevicesArray[x].getAddress().compareToIgnoreCase("
    98:D3:91:FD:38:05") == 0) {
244                                 if (!terminateAutoConnect&&!bluetoothConnected)
    connectBluetoothDeviceSync(x);
245                             }
246                         }
247                         try { Thread.sleep(10000);} catch (Throwable th) { th.
    printStackTrace(); }
248                     }
249                     isAutoConnectRunning = false;
250                     terminateAutoConnect = false;
251                 }
252             });
253         }else{
254             terminateAutoConnect = false;
255         }
256     }
257
258     public static void terminateBTAutoConnect(){
259         terminateAutoConnect = true;
260     }
261
262     public static Boolean isAutoConnectRunning(){
263         return isAutoConnectRunning;
264     }
265
266
267
268
269     public static void connectionLost(){
270
271     }
272
273     public static void closeAllConnections(){
274         try {
275             bluetoothConnected = false;
276             if (bluetoothComThread != null) bluetoothComThread.cancel(true);
277             if (btSocket != null) btSocket.close();
278             if (outputStream != null) outputStream.close();
279             if (inputStream != null) inputStream.close();
280
281             btSocket = null;
282             outputStream = null;
283             inputStream = null;
284         } catch (Throwable th){
285             th.printStackTrace();
286         }
287     }
288
```

```
289
290
291     public static BroadcastReceiver bluetoothAdapterStatusBR = new BroadcastReceiver() {
292         @Override
293         public void onReceive(Context context, Intent intent) {
294             final String action = intent.getAction();
295             if (action.equals(BluetoothAdapter.ACTION_STATE_CHANGED)){
296                 final int stateCode = intent.getIntExtra(BluetoothAdapter.EXTRA_STATE,
    BluetoothAdapter.ERROR);
297
298                 switch (stateCode){
299                     case BluetoothAdapter.STATE_OFF:
300                         //Bluetooth adapter turned off handler
301                         BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.
    BLUETOOTH_DISABLED);
302                         break;
303                     case BluetoothAdapter.STATE_TURNING_OFF:
304                         //Bluetooth adapter turning off handler
305                         terminateBTAutoConnect();
306                         BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.
    BT_TURNING_OFF);
307                         break;
308                     case BluetoothAdapter.STATE_ON:
309                         BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.
    SYSTEM_DISCONNECTED);
310                         updateDevicesList();
311                         initBTAutoConnectService();
312
313                         break;
314                     case BluetoothAdapter.STATE_TURNING_ON:
315                         //Bluetooth adapter turning on handler
316                         BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.
    BT_TURNING_ON);
317                         break;
318                 }
319             }
320         }
321     };
322
323     public static BroadcastReceiver bluetoothScanModeStatusBR = new BroadcastReceiver() {
324
325         @Override
326         public void onReceive(Context context, Intent intent) {
327             final String action = intent.getAction();
328
329             if (action.equals(BluetoothAdapter.ACTION_SCAN_MODE_CHANGED)) {
330
331                 int mode = intent.getIntExtra(BluetoothAdapter.EXTRA_SCAN_MODE,
    BluetoothAdapter.ERROR);
332
333                 switch (mode) {
334                     case BluetoothAdapter.SCAN_MODE_CONNECTABLE_DISCOVERABLE:
335
336                         break;
337                     case BluetoothAdapter.SCAN_MODE_CONNECTABLE:
338
339                         break;
340                     case BluetoothAdapter.SCAN_MODE_NONE:
341
342                         break;
343                 }
344             }
345         }
```

```
346     };
347
348     public static BroadcastReceiver bluetoothDeviceStatusBR = new BroadcastReceiver() {
349         @Override
350         public void onReceive(Context context, Intent intent) {
351             String action = intent.getAction();
352
353             switch (action){
354                 case BluetoothDevice.ACTION_ACL_CONNECTED:
355                     terminateBTAutoConnect();
356                     BluetoothReceiverService.initialize();
357                     updateDevicesList();
358                     break;
359                 case BluetoothDevice.ACTION_ACL_DISCONNECTED:
360                     BluetoothReceiverService.terminateSync();
361                     closeAllConnections();
362                     BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.
    SYSTEM_DISCONNECTED);
363                     initBTAutoConnectService();
364                     updateBluetoothListGUI();
365                     //Application.toastMessage("Dispositivo desconectado!");
366                     break;
367                 case BluetoothDevice.ACTION_ACL_DISCONNECT_REQUESTED:
368
369                     break;
370                 case BluetoothDevice.ACTION_BOND_STATE_CHANGED:
371
372                     break;
373             }
374         }
375     };
376
377 }
378
```

Archivo - UnifiedTransmissionProtocol.java

```java
package com.example.SAPLM.bluetoothActivities;

import com.example.SAPLM.R;
import com.example.SAPLM.commandActivity.CommandsActivity;
import com.example.SAPLM.settingsActivities.DeveloperConsoleActivity;
import com.example.SAPLM.settingsActivities.TerminalManager;


import java.io.UnsupportedEncodingException;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.concurrent.Executor;
import java.util.concurrent.Executors;
import java.util.regex.Matcher;
import java.util.regex.Pattern;



public class UnifiedTransmissionProtocol {

    public static List<Module> modulesList = new ArrayList<UnifiedTransmissionProtocol.Module>(Arrays.asList(
            new Module("Dispositivo Movil", "PHONE", (byte)0x00, R.drawable.ic_icon_phone_module, new ArrayList<Module.Device>(Arrays.asList(
                    new Module.Device("Módulo principal", new Boolean(false), R.drawable.ic_icon_main_board, (byte)0x00, new Boolean(true))
            ))),
            new Module("Módulo principal","MAIN", (byte)0x01, R.drawable.ic_icon_main_board, new ArrayList<Module.Device>(Arrays.asList(
                    new Module.Device("Módulo de potencia", new Boolean(false), R.drawable.icons_board, (byte)0x00, new Boolean(true)),
                    new Module.Device("Módulo motriz", new Boolean(false), R.drawable.icons_board, (byte)0x01, new Boolean(true))
            ))),
            new Module("Módulo de potencia","POWER", (byte)0x02, R.drawable.icons_board, new ArrayList<Module.Device>(Arrays.asList(
                    new Module.Device("Velador", new Boolean(false), R.drawable.icons_desk_lamp, (byte)0x00, new Boolean(false)),
                    new Module.Device("Luz", new Boolean(false), R.drawable.icons_bulb, (byte)0x01, new Boolean(false)),
                    new Module.Device("Ventilador", new Boolean(false), R.drawable.icons_fan, (byte)0x02, new Boolean(false)),
                    new Module.Device("Estufa", new Boolean(false), R.drawable.icons_heater, (byte)0x03, new Boolean(false))
            ))),
            new Module("Módulo motriz","MOTOR", (byte)0x03, R.drawable.icons_board, new ArrayList<Module.Device>(Arrays.asList(
                    new Module.Device("Camilla", new Byte((byte)0), R.drawable.icons_stretcher, (byte)0x00, new Boolean(false)),
                    new Module.Device("Cortina", new Byte((byte)0), R.drawable.icons_curtain, (byte)0x01, new Boolean(false)),
                    new Module.Device("Llamar enfermera", new Boolean(false), R.drawable.icons_nurse, (byte)0x02, new Boolean(false))
            )))
    ));

    protected static byte[] command_buffer = new byte[32];

    private static Module currentModuleID = Module.getLocalModule();
```

```
 49     public static Module lastMessageTransmitterModule = new Module();
 50     public static byte lastMessagePID;
 51     private static CommandType lastCommandType;
 52     private static List<Parameter> lastParameterList;
 53     private static boolean isAvailableCommand;
 54
 55
 56     public static byte[] getCommand_buffer() {
 57         return command_buffer;
 58     }
 59
 60
 61
 62
 63
 64
 65     public static class Parameter {
 66         private byte[] parameterData;
 67
 68         Parameter(byte[] parameterData){
 69             this.parameterData = parameterData;
 70         }
 71
 72         public byte[] getParameterData() {
 73             return parameterData;
 74         }
 75
 76         public byte getParameterLength() {
 77             return (byte)parameterData.length;
 78         }
 79     }
 80
 81
 82
 83     public static CommandStatus ComposeMessageToBuffer(CommandType targetType, Module
    targetModule, List<Parameter> parameterList) {
 84         command_buffer = new byte[32];
 85         command_buffer[0] = SectionDividersChar.SOH.getCode();
 86         if (lastMessagePID!=0xFF) { lastMessagePID++; } else { lastMessagePID=(byte)0x00; }
 87         command_buffer[1] =  lastMessagePID;
 88         command_buffer[2] =  targetModule.getInternalCode(); // Final receiver
 89         command_buffer[3] =  currentModuleID.getInternalCode(); // Original transmitter
 90         command_buffer[4] = SectionDividersChar.STX.getCode();
 91         command_buffer[5] = targetType.getCommandCode();
 92
 93         if (parameterList.size()>12) return CommandStatus.PARAMETER_COUNT_OVERSIZE;
 94         int currentIndexWriting = 6;
 95         for (Parameter parameter : parameterList) {
 96             command_buffer[currentIndexWriting] = parameter.getParameterLength();
 97             currentIndexWriting++;
 98             System.arraycopy(parameter.getParameterData(), 0, command_buffer,
    currentIndexWriting, parameter.getParameterLength());
 99             currentIndexWriting+=parameter.getParameterLength();
100         }
101
102         byte CRC = CRC8_CCITT.crc_CCITT(command_buffer, currentIndexWriting);
103         int footerStartIndex = currentIndexWriting;
104
105         command_buffer[footerStartIndex] = SectionDividersChar.ETX.getCode();
106         command_buffer[footerStartIndex+1] = CRC;
107         command_buffer[footerStartIndex+2] = SectionDividersChar.ETB.getCode();
108         return CommandStatus.SUCCESSFUL_COMPOSITION;
109     }
```

```
110
111
112     public static CommandStatus DecomposeMessageFromBuffer() {
113         try {
114             String commandString = new String(command_buffer, "ISO-8859-1");
115             Pattern headerPattern = Pattern.compile("\\u0001(.{3}?)\\u0002", Pattern.DOTALL
    );
116             Pattern footerPattern = Pattern.compile("\\u0003(.{1}?)\\u0017", Pattern.DOTALL
    );
117             Matcher headerMatcher = headerPattern.matcher(commandString);
118             Matcher footerMatcher = footerPattern.matcher(commandString);
119
120             if (headerMatcher.find()==false) return CommandStatus.WRONG_HEADER_SEGMENTATION
    ;
121             if (footerMatcher.find()==false) return CommandStatus.WRONG_FOOTER_SEGMENTATION
    ;
122
123             int headerStartIndex = headerMatcher.start();
124             int parameterStartIndex = headerMatcher.end()+1;
125             int footerStartIndex = footerMatcher.start();
126
127             lastMessagePID = command_buffer[headerStartIndex+1];    //verificar si hay que
    convertir a desde unsigned
128             if (command_buffer[headerStartIndex+2]!=currentModuleID.getInternalCode())
    return CommandStatus.WRONG_MODULE_ID;
129
130             lastMessageTransmitterModule = Module.getModule(command_buffer[headerStartIndex
    +3]);
131
132             lastCommandType = CommandType.definedCommandList.stream().filter(p-> p.
    getCommandCode()==command_buffer[headerMatcher.end()]).findFirst().orElse(null);
133
134             lastParameterList = new ArrayList<Parameter>();
135
136             while(parameterStartIndex!=footerStartIndex) {
137                 int parameterLength = command_buffer[parameterStartIndex];
138                 byte[] parameterData = new byte[parameterLength];
139                 System.arraycopy(command_buffer, parameterStartIndex+1, parameterData, 0,
    parameterLength);
140                 lastParameterList.add(new Parameter(parameterData));
141                 parameterStartIndex+=(1+parameterLength);
142             }
143
144             if (command_buffer[footerStartIndex+1]!=CRC8_CCITT.crc_CCITT(command_buffer,
    footerStartIndex)) return CommandStatus.WRONG_CHECKSUM_CONSISTENCY;
145
146         } catch (UnsupportedEncodingException e) {
147             e.printStackTrace();
148             return CommandStatus.FAILED_DECOMPOSITION;
149         }
150
151
152         isAvailableCommand = true;
153         return CommandStatus.SUCCESSFUL_DECOMPOSITION;
154     }
155
156     public static void handleReceivedCommand() {
157         if (isAvailableCommand) {
158             if (lastCommandType!=null) lastCommandType.handleCommand();
159             isAvailableCommand = false;
160         }
161     }
162
```

```
163
164     enum CommandStatus {
165         SUCCESSFUL_DECOMPOSITION((byte)0x00),
166         SUCCESSFUL_COMPOSITION((byte)0x01),
167         WRONG_HEADER_SEGMENTATION((byte)0x02),
168         WRONG_FOOTER_SEGMENTATION((byte)0x03),
169         WRONG_CHECKSUM_CONSISTENCY((byte)0x04),
170         WRONG_MODULE_ID((byte)0x05),
171         UNDEFINED_COMMAND_CODE((byte)0x06),
172         PARAMETER_DATA_OVERFLOW((byte)0x07),
173         PARAMETER_COUNT_OVERSIZE((byte)0x08),
174         FAILED_DECOMPOSITION((byte)0x09),
175         FAILED_COMPOSITION((byte)0x0A);
176
177         private final byte code;
178         CommandStatus(byte code){
179             this.code = code;
180         }
181         public byte getCode() {
182             return code;
183         }
184
185     }
186
187     public static class Module extends UnifiedTransmissionProtocol{
188         private String moduleName;
189         private String moduleAlias;
190         private byte internalCode;
191         private int iconResourceId;
192         private List<Device> devicesList;
193
194         Module(String moduleName,String moduleAlias, byte internalCode, int iconResourceId,
    List<Device> devicesList){
195             this.moduleName = moduleName;
196             this.internalCode = internalCode;
197             this.iconResourceId = iconResourceId;
198             this.devicesList = devicesList;
199         }
200
201         Module(){
202
203         }
204
205         public List<Device> getDevicesList() {
206             return devicesList;
207         }
208
209         public int getIconResourceId() {
210             return iconResourceId;
211         }
212
213         public byte getInternalCode() {
214             return internalCode;
215         }
216
217         public String getModuleName() {
218             return moduleName;
219         }
220
221         public String getModuleAlias() {
222             return moduleAlias;
223         }
224
```

```
225         public List<Parameter> exportDataToParameterList(){
226             //Desarrollar esta funcion
227             // AVR-GCC USES LITTLE-ENDIAN
228             // AVR-GCC USES STRICTLY IEEE 754 32-BIT FLOAT POINT
229 
230             List<Parameter> exportList = new ArrayList<UnifiedTransmissionProtocol.
    Parameter>();
231 
232             for (int x = 0; x < devicesList.size(); x++) {
233                 for (Module.Device device : devicesList) {
234                     if (x==byteToUnsignedInt(device.getInternalIndex())) {
235                         Parameter newParameter = null;
236 
237                         if (device.getData().getClass()==Byte.class) {
238                             ByteBuffer byteBuffer = ByteBuffer.allocate(Byte.BYTES);
239                             byteBuffer.order(ByteOrder.LITTLE_ENDIAN);
240                             byteBuffer.put((byte) device.getData());
241                             newParameter = new Parameter(byteBuffer.array());
242                         } else if (device.getData().getClass()==Short.class) {
243                             ByteBuffer shortBuffer = ByteBuffer.allocate(Short.BYTES);
244                             shortBuffer.order(ByteOrder.LITTLE_ENDIAN);
245                             shortBuffer.putShort((short) device.getData());
246                             newParameter = new Parameter(shortBuffer.array());
247                         } else if (device.getData().getClass()==Integer.class) {
248                             ByteBuffer intBuffer = ByteBuffer.allocate(Integer.BYTES);
249                             intBuffer.order(ByteOrder.LITTLE_ENDIAN);
250                             intBuffer.putInt((int) device.getData());
251                             newParameter = new Parameter(intBuffer.array());
252                         } else if (device.getData().getClass()==Float.class) {
253                             ByteBuffer floatBuffer = ByteBuffer.allocate(Float.BYTES);
254                             floatBuffer.order(ByteOrder.LITTLE_ENDIAN);
255                             floatBuffer.putFloat((float) device.getData());
256                             newParameter = new Parameter(floatBuffer.array());
257                         } else if (device.getData().getClass()==Double.class) {
258                             ByteBuffer doubleBuffer = ByteBuffer.allocate(Float.BYTES);
259                             doubleBuffer.order(ByteOrder.LITTLE_ENDIAN);
260                             doubleBuffer.putFloat((float)((double)device.getData()));
261                             newParameter = new Parameter(doubleBuffer.array());
262                         } else if (device.getData().getClass()==String.class) {
263                             String data = (String) device.getData();
264                             newParameter = new Parameter(data.getBytes(StandardCharsets.
    US_ASCII));
265                         } else if (device.getData().getClass()==Boolean.class) {
266                             byte data = (byte) 0x00;
267                             if ((Boolean)device.getData()==true) data = (byte)0xFF;
268                             ByteBuffer booleanBuffer = ByteBuffer.allocate(Byte.BYTES);
269                             booleanBuffer.order(ByteOrder.LITTLE_ENDIAN);
270                             booleanBuffer.put(data);
271                             newParameter = new Parameter(booleanBuffer.array());
272                         }
273 
274                         if (newParameter!=null) exportList.add(newParameter);
275 
276                         break;
277                     }
278 
279                 }
280             }
281 
282             return exportList;
283         }
284 
285         public void updateDataFromParameterList(List<Parameter> parameterList){
```

```
286
287             for (int x = 0; x < parameterList.size(); x++) {
288                 Device deviceToUpdate = devicesList.get(x);
289                 ByteBuffer buffer = ByteBuffer.wrap(parameterList.get(x).getParameterData()
    );
290                 buffer.order(ByteOrder.LITTLE_ENDIAN);
291
292                 if (deviceToUpdate.getData().getClass()==Byte.class) {
293                     deviceToUpdate.setData(new Byte(buffer.get()));
294                 } else if (deviceToUpdate.getData().getClass()==Short.class) {
295                     deviceToUpdate.setData(new Short(buffer.getShort()));
296                 } else if (deviceToUpdate.getData().getClass()==Integer.class) {
297                     deviceToUpdate.setData(new Integer(buffer.getInt()));
298                 } else if (deviceToUpdate.getData().getClass()==Float.class) {
299                     deviceToUpdate.setData(new Float(buffer.getFloat()));
300                 } else if (deviceToUpdate.getData().getClass()==Double.class) {
301                     deviceToUpdate.setData(new Double(buffer.getFloat()));
302                 } else if (deviceToUpdate.getData().getClass()==String.class) {
303                     deviceToUpdate.setData(new String(parameterList.get(x).getParameterData
    (),StandardCharsets.US_ASCII));
304                 } else if (deviceToUpdate.getData().getClass()==Boolean.class) {
305                     if (buffer.get()==(byte)0xFF) {
306                         deviceToUpdate.setData(new Boolean(true));
307                     } else {
308                         deviceToUpdate.setData(new Boolean(false));
309                     }
310                 }
311
312
313                 devicesList.set( x, deviceToUpdate );
314             }
315         }
316
317         public void updateDeviceValue(byte internalIndex, byte[] data) {
318             for (int x = 0; x < devicesList.size(); x++) {
319                 if (devicesList.get(x).getInternalIndex()==internalIndex) {
320                     Device deviceToUpdate = devicesList.get(x);
321                     ByteBuffer buffer = ByteBuffer.wrap(data);
322                     buffer.order(ByteOrder.LITTLE_ENDIAN);
323
324                     if (deviceToUpdate.getData().getClass()==Byte.class) {
325                         deviceToUpdate.setData(new Byte(buffer.get()));
326                     } else if (deviceToUpdate.getData().getClass()==Short.class) {
327                         deviceToUpdate.setData(new Short(buffer.getShort()));
328                     } else if (deviceToUpdate.getData().getClass()==Integer.class) {
329                         deviceToUpdate.setData(new Integer(buffer.getInt()));
330                     } else if (deviceToUpdate.getData().getClass()==Float.class) {
331                         deviceToUpdate.setData(new Float(buffer.getFloat()));
332                     } else if (deviceToUpdate.getData().getClass()==Double.class) {
333                         deviceToUpdate.setData(new Double(buffer.getFloat()));
334                     } else if (deviceToUpdate.getData().getClass()==String.class) {
335                         deviceToUpdate.setData(new String(data,StandardCharsets.US_ASCII));
336                     } else if (deviceToUpdate.getData().getClass()==Boolean.class) {
337                         if (buffer.get()==(byte)0xFF) {
338                             deviceToUpdate.setData(new Boolean(true));
339                         } else {
340                             deviceToUpdate.setData(new Boolean(false));
341                         }
342                     }
343                 }
344             }
345         }
346
```

```
347
348         public byte[] exportDeviceValue(byte internalIndex) {
349             for (int x = 0; x < devicesList.size(); x++) {
350                 if (devicesList.get(x).getInternalIndex()==internalIndex) {
351
352                     if (devicesList.get(x).getData().getClass()==Byte.class) {
353                         ByteBuffer byteBuffer = ByteBuffer.allocate(Byte.BYTES);
354                         byteBuffer.order(ByteOrder.LITTLE_ENDIAN);
355                         byteBuffer.put((byte) devicesList.get(x).getData());
356                         return byteBuffer.array();
357                     } else if (devicesList.get(x).getData().getClass()==Short.class) {
358                         ByteBuffer shortBuffer = ByteBuffer.allocate(Short.BYTES);
359                         shortBuffer.order(ByteOrder.LITTLE_ENDIAN);
360                         shortBuffer.putShort((short) devicesList.get(x).getData());
361                         return shortBuffer.array();
362                     } else if (devicesList.get(x).getData().getClass()==Integer.class) {
363                         ByteBuffer intBuffer = ByteBuffer.allocate(Integer.BYTES);
364                         intBuffer.order(ByteOrder.LITTLE_ENDIAN);
365                         intBuffer.putInt((int) devicesList.get(x).getData());
366                         return intBuffer.array();
367                     } else if (devicesList.get(x).getData().getClass()==Float.class) {
368                         ByteBuffer floatBuffer = ByteBuffer.allocate(Float.BYTES);
369                         floatBuffer.order(ByteOrder.LITTLE_ENDIAN);
370                         floatBuffer.putFloat((float) devicesList.get(x).getData());
371                         return floatBuffer.array();
372                     } else if (devicesList.get(x).getData().getClass()==Double.class) {
373                         ByteBuffer doubleBuffer = ByteBuffer.allocate(Float.BYTES);
374                         doubleBuffer.order(ByteOrder.LITTLE_ENDIAN);
375                         doubleBuffer.putFloat((float)((double)devicesList.get(x).getData())
    );
376                         return doubleBuffer.array();
377                     } else if (devicesList.get(x).getData().getClass()==String.class) {
378                         String data = (String) devicesList.get(x).getData();
379                         return data.getBytes(StandardCharsets.US_ASCII);
380                     } else if (devicesList.get(x).getData().getClass()==Boolean.class) {
381                         byte data = (byte) 0x00;
382                         if ((Boolean)devicesList.get(x).getData()==true) data = (byte)0xFF;
383                         ByteBuffer booleanBuffer = ByteBuffer.allocate(Byte.BYTES);
384                         booleanBuffer.order(ByteOrder.LITTLE_ENDIAN);
385                         booleanBuffer.put(data);
386                         return booleanBuffer.array();
387                     }
388                 }
389             }
390
391             return null;
392         }
393
394
395         private static Executor synchronizationExecutor = Executors.newSingleThreadExecutor
    ();
396         private static boolean synchronizationRunning = false;
397
398         public static boolean awaitingModuleResponse = false;
399         public static byte awaitingModuleResponse_ModuleInternalCode = (byte)0x00;
400         public static boolean awaitingSynchronization = false;
401
402         public static void syncAllDevicesValueToSystem(){
403             // Send data to system from internal data
404
405
406             if (synchronizationRunning==false) {
407                 synchronizationExecutor.execute(new Runnable() {
```

```
408                     @Override
409                     public void run() {
410                         synchronizationRunning = true;
411                         do {
412                             awaitingSynchronization=false;
413                             for (Module mod : modulesList) {
414                                 for (Module.Device dev : mod.getDevicesList()) {
415                                     if (dev.isSynchronizableOnlyFromSystem() == false &&
    dev.isSynchronizedData() == false) {
416                                         List<Parameter> export = mod.
    exportDataToParameterList();
417 
418                                         ComposeMessageToBuffer(CommandType.
    UPDATE_ALL_DEVICES_VALUE, mod, export);
419                                         DeveloperConsoleActivity.sendTextToTerminal("Sent
    to: " + mod.getModuleName() + " : " + TerminalManager.insertSpaces(bytesToHex(
    command_buffer), " ", 2));
420                                         awaitingModuleResponse = true;
421                                         awaitingModuleResponse_ModuleInternalCode = mod.
    internalCode;
422                                         BluetoothConnection.sendCommandBuffer(
    command_buffer);
423                                         long startTimeMillis = System.currentTimeMillis();
424                                         while ((System.currentTimeMillis() -
    startTimeMillis) < 1000L) {
425                                             if (awaitingModuleResponse == false) {
426                                                 for (Module.Device devSync : mod.
    getDevicesList()) {
427                                                     devSync.setSynchronizedState(true);
428                                                 }
429                                                 break;
430                                             }
431                                         }
432                                         awaitingModuleResponse = false;
433                                         break; //Only send one message per module
434                                     }
435                                 }
436                             }
437                         } while (awaitingSynchronization==true);
438                         synchronizationRunning = false;
439                     }
440                 });
441             } else {
442                 awaitingSynchronization = true;
443             }
444 
445         }
446 
447         public static void syncAllDevicesValueFromSystem(){
448             // Retrieve data from system to internal data
449             // Metodo estatico
450 
451 
452             synchronizationRunning = true;
453 
454             synchronizationRunning = false;
455         }
456 
457 
458 
459 
460         public static Module getModule(byte internalCode) {
461             for (Module mod : modulesList) {
```

```
462                 if (mod.getInternalCode()==internalCode) return mod;
463             }
464             return null;
465         }
466 
467         public static Module getLocalModule() {
468             byte localModuleInternalCode = (byte) 0x00;
469             for (Module mod : modulesList) {
470                 if (mod.getInternalCode()==localModuleInternalCode) return mod;
471             }
472             return null;
473         }
474 
475         public String getModuleStatusTerminal() {
476 
477             String composedString = new String("");
478 
479 
480 
481             for (Device dev : devicesList) {
482                 composedString += dev.getName() + " : " + dev.getData().toString() + System
    .lineSeparator();
483             }
484 
485 
486 
487             return composedString;
488         }
489 
490 
491         public static class Device{
492             private String name;
493             public Object data;
494             private int iconResourceId;
495             private byte internalIndex;     // Index or internal code of the device
496             private boolean synchronizedData = false;
497             private boolean synchronizableOnlyFromSystem = false;   // Only synchronize
    from system
498 
499             Device(String name, Object data, int iconResourceId, byte internalIndex,
    boolean synchronizableOnlyFromSystem){
500                 this.name = name;
501                 this.data = data;
502                 this.iconResourceId = iconResourceId;
503                 this.internalIndex = internalIndex;
504                 this.synchronizableOnlyFromSystem = synchronizableOnlyFromSystem;
505             }
506 
507             public void setData(Object data) {
508                 if (synchronizableOnlyFromSystem == false) {
509                     CommandsActivity.updateGUI();
510                     this.data = data;
511                     synchronizedData = false;
512                 }
513             }
514 
515             public Object getData() {
516                 return data;
517             }
518 
519             public int getIconResourceId() {
520                 return iconResourceId;
521             }
```

```
522
523             public byte getInternalIndex() {
524                 return internalIndex;
525             }
526
527             public String getName() {
528                 return name;
529             }
530
531             public boolean isSynchronizableOnlyFromSystem() {
532                 return synchronizableOnlyFromSystem;
533             }
534
535             public boolean isSynchronizedData() {
536                 return synchronizedData;
537             }
538
539             public void setSynchronizedState(boolean synchronizedData) {
540                 this.synchronizedData = synchronizedData;
541             }
542
543             public void deviceOnClickHandler(){
544                 switch (name){
545                     case "Módulo principal":
546                         break;
547                     case "Módulo de potencia":
548                         break;
549                     case "Módulo motriz":
550                         break;
551
552                     case "Velador":
553                         if ((boolean)data == false) { setData(Boolean.TRUE); } else {
    setData(Boolean.FALSE); }
554                         break;
555                     case "Luz":
556                         if ((boolean)data == false) { setData(Boolean.TRUE); } else {
    setData(Boolean.FALSE); }
557                         break;
558                     case "Ventilador":
559                         if ((boolean)data == false) { setData(Boolean.TRUE); } else {
    setData(Boolean.FALSE); }
560                         break;
561                     case "Estufa":
562                         if ((boolean)data == false) { setData(Boolean.TRUE); } else {
    setData(Boolean.FALSE); }
563                         break;
564                     case "Camilla":
565                         if (byteToUnsignedInt((byte)data)<3) { setData((byte)(((byte)data)+
    1)); } else { setData((byte)0x00); }
566                         break;
567                     case "Cortina":
568                         if (byteToUnsignedInt((byte)data)<7) { setData((byte)(((byte)data)+
    1)); } else { setData((byte)0x00); }
569                         break;
570                     case "Llamar enfermera":
571                         if ((boolean)data == false) {
572                             setData(Boolean.TRUE);
573
574                             autoToggleNurseStateThread = new Thread(
    autoToggleNurseStateRunnable);
575                             autoToggleNurseStateThread.start();
576                         } else { setData(Boolean.FALSE); }
577                         break;
```

```
578                 }
579                 syncAllDevicesValueToSystem();
580             }
581 
582             public int getProgressBarValue(){
583                 int returnValue = 0;
584                 switch (name){
585                     case "Módulo principal":
586                         if ((boolean)data == false) { returnValue = 0; } else { returnValue
    = 100; }
587                         break;
588                     case "Módulo de potencia":
589                         if ((boolean)data == false) { returnValue = 0; } else { returnValue
    = 100; }
590                         break;
591                     case "Módulo motriz":
592                         if ((boolean)data == false) { returnValue = 0; } else { returnValue
    = 100; }
593                         break;
594 
595                     case "Velador":
596                         if ((boolean)data == false) { returnValue = 0; } else { returnValue
    = 100; }
597                         break;
598                     case "Luz":
599                         if ((boolean)data == false) { returnValue = 0; } else { returnValue
    = 100; }
600                         break;
601                     case "Ventilador":
602                         if ((boolean)data == false) { returnValue = 0; } else { returnValue
    = 100; }
603                         break;
604                     case "Estufa":
605                         if ((boolean)data == false) { returnValue = 0; } else { returnValue
    = 100; }
606                         break;
607 
608                     case "Camilla":
609                         returnValue = Math.round((float)(((byte)data)*33.333f));
610                         break;
611                     case "Cortina":
612                         returnValue = Math.round((float)(((byte)data)*14.286f));;
613                         break;
614                     case "Llamar enfermera":
615                         if ((boolean)data == false) { returnValue = 0; } else { returnValue
    = 100; }
616                         break;
617                 }
618                 return returnValue;
619             }
620 
621             public String getDataAsString(){
622                 String returnString = "";
623                 switch (name){
624                     case "Módulo principal":
625                         if ((boolean)data == false) { returnString = "Desconectado"; } else
    { returnString = "En linea"; }
626                         break;
627                     case "Módulo de potencia":
628                         if ((boolean)data == false) { returnString = "Desconectado"; } else
    { returnString = "En linea"; }
629                         break;
630                     case "Módulo motriz":
```

```
631                         if ((boolean)data == false) { returnString = "Desconectado"; } else
     { returnString = "En linea"; }
632                         break;
633
634                     case "Velador":
635                         if ((boolean)data == false) { returnString = "Apagado"; } else {
    returnString = "Encendido"; }
636                         break;
637                     case "Luz":
638                         if ((boolean)data == false) { returnString = "Apagado"; } else {
    returnString = "Encendido"; }
639                         break;
640                     case "Ventilador":
641                         if ((boolean)data == false) { returnString = "Apagado"; } else {
    returnString = "Encendido"; }
642                         break;
643                     case "Estufa":
644                         if ((boolean)data == false) { returnString = "Apagado"; } else {
    returnString = "Encendido"; }
645                         break;
646
647                     case "Camilla":
648                         returnString = "Posición " + byteToUnsignedInt((byte)data);
649                         break;
650                     case "Cortina":
651                         returnString = "Posición " + byteToUnsignedInt((byte)data);
652                         break;
653                     case "Llamar enfermera":
654                         if ((boolean)data == false) { returnString = "Disponible"; } else {
     returnString = "Llamando"; }
655                         break;
656                 }
657                 return returnString;
658             }
659
660             public static Thread autoToggleNurseStateThread;
661             Runnable autoToggleNurseStateRunnable = new Runnable() {
662                 @Override
663                 public void run() {
664                     try {
665                         Thread.sleep(3000);
666                         modulesList.get(3).getDevicesList().get(2).setData(new Boolean(
    false));
667                         modulesList.get(3).getDevicesList().get(2).setSynchronizedState(
    true);
668                         CommandsActivity.updateGUI();
669                     } catch (InterruptedException e) {
670                         e.printStackTrace();
671                     }
672                 }
673             };
674
675         }
676
677     }
678
679     enum SectionDividersChar {
680         SOH((byte)0x01),
681         STX((byte)0x02),
682         ETX((byte)0x03),
683         ETB((byte)0x17);
684
685         private final byte code;
```

```
686         SectionDividersChar(byte code){
687             this.code = code;
688         }
689         public byte getCode() {
690             return code;
691         }
692     }
693 
694     public static class CommandType{
695         private byte commandCode;
696         private String commandTag;
697         private Runnable handlerRunnable;
698 
699 
700         CommandType(byte cmdCode, String tag, Runnable handlerCommand){
701             this.commandCode = cmdCode;
702             this.commandTag = tag;
703             this.handlerRunnable = handlerCommand;
704         }
705 
706         public byte getCommandCode() {
707             return commandCode;
708         }
709 
710         public String getCommandTag() {
711             return commandTag;
712         }
713 
714         public void setCommandCode(byte commandCode) {
715             this.commandCode = commandCode;
716         }
717 
718         public void setCommandTag(String commandTag) {
719             this.commandTag = commandTag;
720         }
721 
722         public void handleCommand() {
723             handlerRunnable.run();
724         }
725 
726 
727         public static ArrayList<CommandType> definedCommandList = new ArrayList<CommandType
    >(Arrays.asList(
728                 new CommandType((byte)0x00, "UPDATE_ALL_DEVICES_VALUE", new Runnable() {
729                     @Override
730                     public void run() {
731 
732                         modulesList.get(byteToUnsignedInt(lastMessageTransmitterModule.
    getInternalCode())).updateDataFromParameterList(lastParameterList);
733 
734                         System.out.println("Running UPDATE_ALL_DEVICES_VALUE!");
735 
736                     }}),
737                 new CommandType((byte)0x01, "UPDATE_DEVICE_VALUE", new Runnable() {
738                     @Override
739                     public void run() {
740 
741 
742                         ByteBuffer buffer = ByteBuffer.wrap(lastParameterList.get(0).
    getParameterData());
743 
744                         modulesList.get(byteToUnsignedInt(lastMessageTransmitterModule.
    getInternalCode())).updateDeviceValue(buffer.get(),lastParameterList.get(1).
```

```
744 getParameterData());
745
746                         System.out.println("Running UPDATE_DEVICE_VALUE!");
747
748
749
750                     }}),
751                 new CommandType((byte)0x02, "GET_ALL_DEVICES_VALUE", new Runnable() {
752                     @Override
753                     public void run() {
754                         ComposeMessageToBuffer(definedCommandList.stream().filter(p-> p.
    getCommandTag().equals(("UPDATE_ALL_DEVICES_VALUE"))).findFirst().orElse(null),
755                                 lastMessageTransmitterModule,
756                                 modulesList.get(byteToUnsignedInt(
    lastMessageTransmitterModule.getInternalCode())).exportDataToParameterList());
757
758                         // SEND COMMAND_BUFFER VIA BLUETOOTH
759
760
761                         System.out.println("Running GET_ALL_DEVICES_VALUE!");
762                     }}),
763                 new CommandType((byte)0x03, "GET_DEVICE_VALUE", new Runnable() {
764                     @Override
765                     public void run() {
766
767                         ByteBuffer buffer = ByteBuffer.wrap(lastParameterList.get(0).
    getParameterData());
768                         byte[] deviceData = modulesList.get(byteToUnsignedInt(
    lastMessageTransmitterModule.getInternalCode())).exportDeviceValue(buffer.get());
769
770                         ComposeMessageToBuffer(definedCommandList.stream().filter(p-> p.
    getCommandTag().equals(("UPDATE_DEVICE_VALUE"))).findFirst().orElse(null),
771                                 lastMessageTransmitterModule,
772                                 new ArrayList<Parameter>(Arrays.asList(new Parameter(new
    byte[] {buffer.get()}), new Parameter(deviceData))));
773
774                         // SEND COMMAND_BUFFER VIA BLUETOOTH
775
776                         System.out.println("Running GET_DEVICE_VALUE!");
777                     }}),
778                 new CommandType((byte)0x04, "MESSAGE_STATUS", new Runnable() {
779                     @Override
780                     public void run() {
781
782                         ByteBuffer buffer = ByteBuffer.wrap(lastParameterList.get(0).
    getParameterData());
783                         if (buffer.get()==(byte)0x09){
784                             Module.awaitingModuleResponse = false;
785                         }
786
787
788
789
790                     }})
791         ));
792
793         public static CommandType UPDATE_ALL_DEVICES_VALUE = definedCommandList.get(0);
794         public static CommandType UPDATE_DEVICE_VALUE = definedCommandList.get(1);
795         public static CommandType GET_ALL_DEVICES_VALUE = definedCommandList.get(2);
796         public static CommandType GET_DEVICE_VALUE = definedCommandList.get(3);
797         public static CommandType MESSAGE_STATUS = definedCommandList.get(4);
798
799     }
```

```
800
801     public static int byteToUnsignedInt(byte x){
802         return ((int) x) & 0xff;
803     }
804
805
806     private static final char[] HEX_ARRAY = "0123456789ABCDEF".toCharArray();
807     public static String bytesToHex(byte[] bytes) {
808         char[] hexChars = new char[bytes.length * 2];
809         for (int j = 0; j < bytes.length; j++) {
810             int v = bytes[j] & 0xFF;
811             hexChars[j * 2] = HEX_ARRAY[v >>> 4];
812             hexChars[j * 2 + 1] = HEX_ARRAY[v & 0x0F];
813         }
814         return new String(hexChars);
815     }
816
817
818
819
820 }
821
```

Archivo - activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com
/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/background_light"
    tools:context=".MainActivity">

    <ImageButton
        android:id="@+id/imgBtn_microphone_background"
        android:layout_width="160dp"
        android:layout_height="160dp"
        android:layout_marginStart="8dp"
        android:layout_marginEnd="8dp"
        android:background="?attr/selectableItemBackgroundBorderless"
        android:onClick="onClickListener_microphone"
        android:scaleType="fitXY"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:tint="@color/ColorBackMicKeyword"
        app:srcCompat="@drawable/ic_main_mic_circle"
        android:stateListAnimator="@null"/>

    <ImageButton
        android:id="@+id/imgBtn_microphone"
        android:layout_width="115dp"
        android:layout_height="115dp"
        android:layout_marginStart="8dp"
        android:layout_marginEnd="8dp"
        android:background="?attr/selectableItemBackgroundBorderless"
        android:onClick="onClickListener_microphone"
        android:scaleType="fitXY"
        android:stateListAnimator="@null"
        android:tint="@color/ColorMic"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.501"
        app:srcCompat="@drawable/ic_main_mic_listening" />

    <ImageButton
        android:id="@+id/imgBtn_commands"
        android:layout_width="55dp"
        android:layout_height="55dp"
        android:layout_marginStart="284dp"
        android:alpha="0.0"
        android:background="?attr/selectableItemBackgroundBorderless"
        android:onClick="onClickListener_commands"
        android:scaleType="fitXY"
        android:visibility="gone"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/ic_commands_menu" />

    <ImageButton
        android:id="@+id/imgBtn_contacts"
        android:layout_width="55dp"
```

```
        android:layout_height="55dp"
        android:layout_marginStart="283dp"
        android:alpha="0.0"
        android:background="?attr/selectableItemBackgroundBorderless"
        android:onClick="onClickListener_contacts"
        android:scaleType="fitXY"
        android:visibility="gone"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/ic_contacts_menu" />

    <ImageButton
        android:id="@+id/imgBtn_bluetooth"
        android:layout_width="55dp"
        android:layout_height="55dp"
        android:layout_marginStart="283dp"
        android:alpha="0.0"
        android:background="?attr/selectableItemBackgroundBorderless"
        android:onClick="onClickListener_bluetooth"
        android:scaleType="fitXY"
        android:visibility="gone"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/ic_icono_bluetooth_remake" />

    <ImageButton
        android:id="@+id/imgBtn_settings"
        android:layout_width="55dp"
        android:layout_height="55dp"
        android:layout_marginStart="283dp"
        android:alpha="0.0"
        android:background="?attr/selectableItemBackgroundBorderless"
        android:onClick="onClickListener_settings"
        android:scaleType="fitXY"
        android:visibility="gone"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/ic_settings_menu" />

    <ImageButton
        android:id="@+id/imgBtn_menu"
        android:layout_width="55dp"
        android:layout_height="55dp"
        android:layout_marginStart="8dp"
        android:layout_marginEnd="8dp"
        android:background="?attr/selectableItemBackgroundBorderless"
        android:onClick="onClickListener_menu"
        android:scaleType="fitXY"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/imgBtn_microphone_background"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/ic_bluetooth_menu" />

    <TextView
        android:id="@+id/textView_speechRecognitionResults"
        style="@style/hypothesisTextAppearance"
        android:layout_width="311dp"
        android:layout_height="101dp"
        android:layout_marginStart="8dp"
```

```
126         android:layout_marginTop="8dp"
127         android:layout_marginEnd="8dp"
128         android:layout_marginBottom="8dp"
129         android:fontFamily="@font/roboto_bold"
130         android:text="Lorem ipsum dolor sit amet, consectetur\nNulla lobortis ac arcu vitae
     aliquam.\nCras feugiat, est quis volutpat interdum\nhola comer te va picante\nhola pepe
    aguante boquita"
131         android:textAlignment="center"
132         android:visibility="gone"
133         app:layout_constraintBottom_toBottomOf="parent"
134         app:layout_constraintEnd_toEndOf="parent"
135         app:layout_constraintStart_toStartOf="parent"
136         app:layout_constraintTop_toTopOf="parent"
137         app:layout_constraintVertical_bias="0.931" />
138
139 </androidx.constraintlayout.widget.ConstraintLayout>
```

Archivo - commands_layout.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/commandsRecyclerView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_behavior="@string/appbar_scrolling_view_behavior"
        android:clipToPadding="false"
        android:paddingBottom="8dp"
        android:scrollbarStyle="outsideOverlay"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/toolbar"
        app:layout_constraintVertical_bias="0.0"
        android:paddingTop="?attr/actionBarSize" />


    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="0dp"
        android:layout_height="57dp"
        android:background="@color/toolBarColor"
        app:titleTextColor="@color/colorPrimary"
        android:minHeight="?attr/actionBarSize"
        android:outlineAmbientShadowColor="@android:color/background_light"
        android:outlineSpotShadowColor="@android:color/background_light"
        android:theme="?attr/actionBarTheme"
        app:title="@string/commands_activity_title"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:subtitleTextColor="@android:color/background_light"
        app:titleTextAppearance="@style/toolBarTextAppearance"/>


</androidx.constraintlayout.widget.ConstraintLayout>
```

Archivo - contacts_layout.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="0dp"
        android:layout_height="57dp"
        android:background="@color/toolBarColor"
        app:titleTextColor="@color/colorPrimary"
        android:minHeight="?attr/actionBarSize"
        android:outlineAmbientShadowColor="@android:color/background_light"
        android:outlineSpotShadowColor="@android:color/background_light"
        android:theme="?attr/actionBarTheme"
        app:title="@string/contacts_activity_title"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:subtitleTextColor="@android:color/background_light"
        app:titleTextAppearance="@style/toolBarTextAppearance"/>


</androidx.constraintlayout.widget.ConstraintLayout>
```

Archivo - settings_layout.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="0dp"
        android:layout_height="57dp"
        android:background="@color/toolBarColor"
        android:minHeight="?attr/actionBarSize"
        android:outlineAmbientShadowColor="@android:color/background_light"
        android:outlineSpotShadowColor="@android:color/background_light"
        android:theme="?attr/actionBarTheme"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:subtitleTextColor="@android:color/background_light"
        app:title="@string/settings_activity_title"
        app:titleTextColor="@color/colorPrimary"
        app:titleTextAppearance="@style/toolBarTextAppearance"/>

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/settingsRecyclerView"
        android:layout_width="match_parent"
        android:layout_height="600dp"
        android:clipToPadding="false"
        android:scrollbarStyle="outsideOverlay"
        android:paddingBottom="8dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/toolbar" />

</androidx.constraintlayout.widget.ConstraintLayout>

```

Archivo - bluetooth_layout.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:clipToPadding="false">

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="0dp"
        android:layout_height="57dp"
        android:background="@color/toolBarColor"
        android:minHeight="?attr/actionBarSize"
        android:outlineAmbientShadowColor="@android:color/background_light"
        android:outlineSpotShadowColor="@android:color/background_light"
        android:theme="?attr/actionBarTheme"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:subtitleTextColor="@android:color/background_light"
        app:title="@string/bluetooth_activity_title"
        app:titleTextAppearance="@style/toolBarTextAppearance"
        app:titleTextColor="@color/colorPrimary" />

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="0dp"
        android:layout_height="418dp"
        android:layout_marginBottom="8dp"
        android:scaleType="fitXY"
        app:layout_constraintBottom_toBottomOf="@id/guideline4"
        app:layout_constraintEnd_toEndOf="@+id/guideline2"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="@+id/guideline"
        app:layout_constraintTop_toBottomOf="@+id/guideline3"
        app:layout_constraintVertical_bias="0.012"
        app:srcCompat="@drawable/bt_dialog" />


    <ImageView
        android:id="@+id/imgView_bluetoothConfig"
        android:layout_width="320dp"
        android:layout_height="78dp"
        android:layout_marginBottom="24dp"
        android:scaleType="fitXY"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/imgView_bluetoothStatus"
        app:layout_constraintVertical_bias="0.88"
        app:srcCompat="@drawable/bt_button" />

    <ImageView
        android:id="@+id/imgView_bluetoothStatus"
        android:layout_width="320dp"
        android:layout_height="78dp"
        android:scaleType="fitXY"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.493"
```

```
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/imageView"
        app:layout_constraintVertical_bias="0.01"
        app:srcCompat="@drawable/bt_button" />

    <TextView
        android:id="@+id/textView_btSearch"
        android:layout_width="215dp"
        android:layout_height="wrap_content"
        android:text="@string/bluetooth_search_message"
        android:textAlignment="center"
        android:maxLines="1"
        android:textAppearance="@style/btButtonTextAppearance"
        app:layout_constraintBottom_toBottomOf="@+id/imgView_bluetoothConfig"
        app:layout_constraintEnd_toEndOf="@+id/imgView_bluetoothConfig"
        app:layout_constraintHorizontal_bias="0.27"
        app:layout_constraintStart_toStartOf="@+id/imgView_bluetoothConfig"
        app:layout_constraintTop_toTopOf="@+id/imgView_bluetoothConfig"
        app:layout_constraintVertical_bias="0.4" />

    <TextView
        android:id="@+id/textView_btStatus"
        android:layout_width="215dp"
        android:layout_height="wrap_content"
        android:maxLines="1"
        android:textAlignment="center"
        android:text="@string/system_disconnected_message"
        android:textAppearance="@style/btButtonTextAppearance"
        android:textColor="@android:color/background_light"
        app:layout_constraintBottom_toBottomOf="@+id/imgView_bluetoothStatus"
        app:layout_constraintEnd_toEndOf="@+id/imgView_bluetoothStatus"
        app:layout_constraintHorizontal_bias="0.26"
        app:layout_constraintStart_toStartOf="@+id/imgView_bluetoothStatus"
        app:layout_constraintTop_toTopOf="@+id/imgView_bluetoothStatus"
        app:layout_constraintVertical_bias="0.422" />

    <TextView
        android:id="@+id/textView_btDialog"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Dispositivos disponibles"
        android:textAppearance="@style/btDialogTextAppearance"
        app:layout_constraintBottom_toBottomOf="@+id/imageView"
        app:layout_constraintEnd_toEndOf="@+id/imageView"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="@+id/imageView"
        app:layout_constraintTop_toTopOf="@+id/imageView"
        app:layout_constraintVertical_bias="0.040" />

    <ImageButton
        android:id="@+id/imgBtn_btConfig"
        android:layout_width="30dp"
        android:layout_height="30dp"
        android:background="?attr/selectableItemBackground"
        android:onClick="configBtn_onClick"
        android:scaleType="fitXY"
        android:tint="@color/colorPrimary"
        app:layout_constraintBottom_toTopOf="@+id/imageView"
        app:layout_constraintEnd_toEndOf="@+id/toolbar"
        app:layout_constraintHorizontal_bias="0.94"
        app:layout_constraintStart_toStartOf="@+id/toolbar"
        app:layout_constraintTop_toTopOf="@+id/toolbar"
        app:layout_constraintVertical_bias="0.5"
```

```
        app:srcCompat="@drawable/ic_config_icon" />

    <ImageView
        android:id="@+id/imgView_btStatusIcon"
        android:layout_width="35dp"
        android:layout_height="35dp"
        android:layout_marginBottom="8dp"
        android:scaleType="fitXY"
        android:tint="@android:color/background_light"
        app:layout_constraintBottom_toBottomOf="@+id/imgView_bluetoothStatus"
        app:layout_constraintEnd_toEndOf="@+id/imgView_bluetoothStatus"
        app:layout_constraintHorizontal_bias="0.934"
        app:layout_constraintStart_toStartOf="@+id/imgView_bluetoothStatus"
        app:layout_constraintTop_toTopOf="@+id/imgView_bluetoothStatus"
        app:srcCompat="@drawable/ic_close_icon"

        />

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/deviceListRecyclerView"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:clipToPadding="false"
        android:scrollbarStyle="outsideOverlay"
        app:layout_constraintBottom_toBottomOf="@+id/bluetoothList_BottomGuideLine"
        app:layout_constraintEnd_toEndOf="@+id/bluetoothList_EndGuideLine"
        app:layout_constraintStart_toStartOf="@+id/bluetoothList_StartGuideLine"
        app:layout_constraintTop_toTopOf="@+id/bluetoothList_TopGuideLine"
         />

    <ImageView
        android:id="@+id/imgView_btDiscovery"
        android:layout_width="30dp"
        android:layout_height="30dp"
        android:scaleType="fitXY"
        android:tint="@android:color/background_light"
        app:layout_constraintBottom_toBottomOf="@+id/imgView_bluetoothConfig"
        app:layout_constraintEnd_toEndOf="@+id/imgView_bluetoothConfig"
        app:layout_constraintHorizontal_bias="0.925"
        app:layout_constraintStart_toStartOf="@+id/imgView_bluetoothConfig"
        app:layout_constraintTop_toTopOf="@+id/imgView_bluetoothConfig"
        app:layout_constraintVertical_bias="0.42"
        app:srcCompat="@drawable/ic_bluetooth_searching" />

    <ProgressBar
        android:id="@+id/loading_progressBar"
        style="?android:attr/progressBarStyle"
        android:layout_width="26dp"
        android:layout_height="26dp"
        android:scaleType="fitXY"

        android:visibility="gone"
        app:layout_constraintBottom_toBottomOf="@+id/imgView_bluetoothStatus"

        app:layout_constraintEnd_toEndOf="@+id/imgView_bluetoothStatus"
        app:layout_constraintHorizontal_bias="0.919"
        app:layout_constraintStart_toStartOf="@+id/imgView_bluetoothStatus"
        app:layout_constraintTop_toTopOf="@+id/imgView_bluetoothStatus"
        app:layout_constraintVertical_bias="0.43" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline"
        android:layout_width="wrap_content"
```

Archivo - bluetooth_layout.xml

```
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.02" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/bluetoothList_StartGuideLine"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.12" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/bluetoothList_EndGuideLine"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.88" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.98055553" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.1" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/bluetoothList_TopGuideLine"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.17" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/bluetoothList_BottomGuideLine"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.688" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.74" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

Archivo - settings_layout_style.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/relativeLayout"
    android:layout_width="match_parent"
    android:layout_height="90dp"
    android:background="?android:attr/selectableItemBackground"
    android:clickable="true"
    android:focusable="true"
    android:orientation="vertical"
    android:paddingLeft="8dp"
    android:paddingTop="8dp"
    android:paddingRight="8dp"
    android:paddingBottom="8dp">

    <TextView
        android:id="@+id/setting_title"
        android:layout_width="250dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:fontFamily="@font/droid_sans_bold"
        android:text="Lorem ipsum dolor"
        android:textColor="@color/colorPrimary"
        android:textSize="18sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:layout_conversion_wrapHeight="56"
        tools:layout_conversion_wrapWidth="788" />

    <TextView
        android:id="@+id/setting_subtitle"
        android:layout_width="250dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:fontFamily="@font/droid_sans"
        android:lines="2"
        android:maxLines="2"
        android:text="Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. "
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toBottomOf="@id/setting_title"
        tools:layout_conversion_wrapHeight="94"
        tools:layout_conversion_wrapWidth="788" />

    <ImageView
        android:id="@+id/setting_icon"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:scaleType="fitXY"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/setting_title"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/ic_setting_contacts_and_communication"
        tools:layout_conversion_wrapHeight="158"
        tools:layout_conversion_wrapWidth="158" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Archivo - information_help_layout.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="0dp"
        android:layout_height="57dp"
        android:background="@color/toolBarColor"
        android:minHeight="?attr/actionBarSize"
        android:outlineAmbientShadowColor="@android:color/background_light"
        android:outlineSpotShadowColor="@android:color/background_light"
        android:theme="?attr/actionBarTheme"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:subtitleTextColor="@android:color/background_light"
        app:title="@string/information_help_title"
        app:titleTextAppearance="@style/toolBarTextAppearance"
        app:titleTextColor="@color/colorPrimary" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

Archivo - commands_settings_layout.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">



    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="0dp"
        android:layout_height="57dp"
        android:background="@color/toolBarColor"
        android:minHeight="?attr/actionBarSize"
        android:outlineAmbientShadowColor="@android:color/background_light"
        android:outlineSpotShadowColor="@android:color/background_light"
        android:theme="?attr/actionBarTheme"
        app:title="@string/commands_settings_title"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:subtitleTextColor="@android:color/background_light"
        app:titleTextColor="@color/colorPrimary"
        app:titleTextAppearance="@style/toolBarTextAppearance"/>



</androidx.constraintlayout.widget.ConstraintLayout>
```

Archivo - developer_console_layout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com
/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">


    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/toolBarColor"
        android:minHeight="?attr/actionBarSize"
        android:outlineAmbientShadowColor="@android:color/background_light"
        android:outlineSpotShadowColor="@android:color/background_light"
        android:theme="?attr/actionBarTheme"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:subtitleTextColor="@android:color/background_light"
        app:title="@string/developer_console_activity_title"
        app:titleTextAppearance="@style/toolBarTextAppearance"
        app:titleTextColor="@color/colorPrimary">



    </androidx.appcompat.widget.Toolbar>

    <ImageButton
        android:id="@+id/clearTerminalButton"
        android:layout_width="45dp"
        android:layout_height="45dp"
        android:background="?attr/selectableItemBackground"
        android:onClick="onClickClearButton"

        android:scaleType="centerInside"
        app:layout_constraintHorizontal_bias="0.96"
        app:layout_constraintVertical_bias="0.01"

        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"

        app:srcCompat="@drawable/ic_delete_sweep_24px" />

    <androidx.constraintlayout.widget.ConstraintLayout
        android:id="@+id/main_area"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/toolbar">

        <ScrollView
            android:id="@+id/scrollViewTerminal"
```

Archivo - developer_console_layout.xml

```
            android:layout_width="match_parent"
            android:layout_height="0dp"
            app:layout_constraintBottom_toTopOf="@id/editTextTerminal"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintRight_toRightOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            android:fastScrollEnabled="true"
            android:fastScrollAlwaysVisible="true"
            android:fadeScrollbars="false">

            <LinearLayout
                android:id="@+id/terminalLinearLayout"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:orientation="vertical" />
        </ScrollView>

        <EditText
            android:id="@+id/editTextTerminal"
            android:layout_width="0dp"
            android:layout_height="50dp"
            android:ems="10"

            android:fontFamily="@font/roboto_mono"


            android:hint="Comando"
            android:inputType="textPersonName"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintHorizontal_bias="0.0"


            app:layout_constraintHorizontal_weight="8"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintRight_toLeftOf="@+id/enterButton"
            app:layout_constraintTop_toBottomOf="@id/scrollViewTerminal" />

        <ImageButton
            android:id="@+id/enterButton"
            android:layout_width="0dp"
            android:layout_height="50dp"
            android:background="?attr/selectableItemBackground"
            android:onClick="onClickEnterButton"
            android:scaleType="centerInside"

            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintHorizontal_weight="2"
            app:layout_constraintLeft_toRightOf="@+id/editTextTerminal"
            app:layout_constraintRight_toRightOf="parent"
            app:layout_constraintTop_toBottomOf="@id/scrollViewTerminal"
            app:srcCompat="@drawable/ic_subdirectory_arrow_left_24px" />

    </androidx.constraintlayout.widget.ConstraintLayout>




</androidx.constraintlayout.widget.ConstraintLayout>
```

Archivo - bluetooth_settings_layout.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="0dp"
        android:layout_height="57dp"
        android:background="@color/toolBarColor"
        android:minHeight="?attr/actionBarSize"
        android:outlineAmbientShadowColor="@android:color/background_light"
        android:outlineSpotShadowColor="@android:color/background_light"
        android:theme="?attr/actionBarTheme"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:subtitleTextColor="@android:color/background_light"
        app:title="@string/bluetooth_settings_title"
        app:titleTextColor="@color/colorPrimary"
        app:titleTextAppearance="@style/toolBarTextAppearance"/>


</androidx.constraintlayout.widget.ConstraintLayout>
```

Archivo - bluetooth_list_layout_style.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/relativeLayout"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="?android:attr/selectableItemBackground"
    android:clickable="true"
    android:focusable="true"
    android:orientation="vertical"
    android:paddingLeft="8dp"
    android:paddingTop="8dp"
    android:paddingRight="8dp"
    android:paddingBottom="8dp">

    <TextView
        android:id="@+id/textView_bluetooth_device_name"
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="2dp"
        android:layout_marginEnd="8dp"
        android:text="Lorem ipsum dolor"
        android:textAppearance="@style/btDeviceListAppearance"
        android:textColor="@color/colorPrimary"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.03"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:layout_conversion_wrapHeight="56"
        tools:layout_conversion_wrapWidth="788" />

    <ImageView
        android:id="@+id/imageView_bluetooth_device_state_icon"
        android:layout_width="25dp"
        android:layout_height="25dp"
        android:layout_marginStart="215dp"
        android:scaleType="fitXY"
        android:tint="@color/colorPrimary"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/ic_close_icon" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Archivo - internet_conectivity_layout.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="0dp"
        android:layout_height="57dp"
        android:background="@color/toolBarColor"
        android:minHeight="?attr/actionBarSize"
        android:outlineAmbientShadowColor="@android:color/background_light"
        android:outlineSpotShadowColor="@android:color/background_light"
        android:theme="?attr/actionBarTheme"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:subtitleTextColor="@android:color/background_light"
        app:title="@string/internet_conectivity_title"
        app:titleTextAppearance="@style/toolBarTextAppearance"
        app:titleTextColor="@color/colorPrimary" />



</androidx.constraintlayout.widget.ConstraintLayout>
```

Archivo - contacts_communication_layout.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="0dp"
        android:layout_height="57dp"
        android:background="@color/toolBarColor"
        android:minHeight="?attr/actionBarSize"
        android:outlineAmbientShadowColor="@android:color/background_light"
        android:outlineSpotShadowColor="@android:color/background_light"
        android:theme="?attr/actionBarTheme"
        app:title="@string/contacts_communication_title"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:subtitleTextColor="@android:color/background_light"
        app:titleTextColor="@color/colorPrimary"
        app:titleTextAppearance="@style/toolBarTextAppearance"/>


</androidx.constraintlayout.widget.ConstraintLayout>
```

Archivo - appearance_miscellaneous_layout.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="0dp"
        android:layout_height="57dp"
        android:background="@color/toolBarColor"
        android:minHeight="?attr/actionBarSize"
        android:outlineAmbientShadowColor="@android:color/background_light"
        android:outlineSpotShadowColor="@android:color/background_light"
        android:theme="?attr/actionBarTheme"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:subtitleTextColor="@android:color/background_light"
        app:title="@string/appearance_miscellaneous_title"
        app:titleTextColor="@color/colorPrimary"
        app:titleTextAppearance="@style/toolBarTextAppearance"/>



</androidx.constraintlayout.widget.ConstraintLayout>
```

Archivo - commands_layout_child_item_style.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com
/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/relativeLayout"
    android:layout_width="match_parent"
    android:layout_height="60dp"
    android:background="?android:attr/selectableItemBackground"
    android:clickable="true"
    android:focusable="true"
    android:orientation="vertical"
    android:paddingLeft="4dp"
    android:paddingTop="4dp"
    android:paddingRight="4dp"
    android:paddingBottom="4dp">

    <TextView
        android:id="@+id/command_state_textView"
        android:layout_width="60dp"
        android:layout_height="wrap_content"
        android:fontFamily="@font/roboto"
        android:text="Encendido"
        android:textAlignment="center"
        android:textColor="@color/colorPrimary"
        android:textSize="9sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toEndOf="@id/guideline10"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="1.0" />

    <TextView
        android:id="@+id/command_title_textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginEnd="8dp"
        android:fontFamily="@font/roboto"
        android:text="Velador numero 3"
        android:textColor="@color/colorPrimary"
        android:textSize="18sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/arrow_list_imageButton"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toEndOf="@+id/command_icon"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.514"
        tools:layout_conversion_wrapHeight="56"
        tools:layout_conversion_wrapWidth="788" />

    <ImageView
        android:id="@+id/command_icon"
        android:layout_width="45dp"
        android:layout_height="45dp"
        android:scaleType="fitXY"
        android:tint="@color/colorPrimary"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toEndOf="@id/guideline_main"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.5"
        app:srcCompat="@drawable/icons_board" />

    <ProgressBar
```

Archivo - commands_layout_child_item_style.xml

```
        android:id="@+id/command_state_progressBar"
        style="?android:progressBarStyleHorizontal"
        android:layout_width="30dp"
        android:layout_height="30dp"
        android:indeterminate="false"
        android:progress="0"
        android:max="100"
        android:secondaryProgress="0"
        android:tint="@color/colorPrimary"
        android:rotation="270"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toEndOf="@id/guideline7"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.5"
        android:progressDrawable="@drawable/progressbar_onboarding_view"
        android:layout_gravity="center"/>

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline_main"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.15" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline7"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.88" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline10"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.8375" />

    <ImageButton
        android:id="@+id/arrow_list_imageButton"
        android:layout_width="35dp"
        android:layout_height="35dp"
        android:background="?attr/selectableItemBackground"
        android:scaleType="fitXY"
        android:tint="@color/colorPrimary"
        android:visibility="gone"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toEndOf="@id/guideline7"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.5"
        app:srcCompat="@drawable/icons_upward_arrow" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

Archivo - commands_layout_parent_item_style.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/relativeLayout"
    android:layout_width="match_parent"
    android:layout_height="55dp"
    android:background="?android:attr/selectableItemBackground"
    android:clickable="true"
    android:focusable="true"
    android:orientation="vertical"
    android:paddingLeft="8dp"
    android:paddingTop="8dp"
    android:paddingRight="8dp"
    android:paddingBottom="8dp">

    <TextView
        android:id="@+id/parent_board_title_TextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:fontFamily="@font/roboto"
        android:text="Placa de mandos motrices"
        android:textColor="@color/colorPrimary"
        android:textSize="18sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toEndOf="@id/guideline6"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.5"
        tools:layout_conversion_wrapHeight="56"
        tools:layout_conversion_wrapWidth="788" />

    <ImageView
        android:id="@+id/parent_board_icon_ImageView"
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:scaleType="fitXY"
        android:tint="@color/colorPrimary"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toEndOf="@id/guideline_main"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.5"
        app:srcCompat="@drawable/icons_board" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline_main"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.04" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline6"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.17" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline8"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
```

Archivo - commands_layout_parent_item_style.xml

```
        app:layout_constraintGuide_percent="0.90" />

    <ImageButton
        android:id="@+id/arrow_list_imageButton"
        android:layout_width="35dp"
        android:layout_height="35dp"
        android:background="?attr/selectableItemBackground"
        android:scaleType="fitXY"
        android:tint="@color/colorPrimary"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toEndOf="@id/guideline8"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.5"
        app:srcCompat="@drawable/icons_upward_arrow" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

Archivo - voice_recognition_settings_layout.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="0dp"
        android:layout_height="57dp"
        android:background="@color/toolBarColor"
        app:titleTextColor="@color/colorPrimary"
        android:minHeight="?attr/actionBarSize"
        android:outlineAmbientShadowColor="@android:color/background_light"
        android:outlineSpotShadowColor="@android:color/background_light"
        android:theme="?attr/actionBarTheme"
        app:title="@string/voice_recognition_settings_title"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:subtitleTextColor="@android:color/background_light"
        app:titleTextAppearance="@style/toolBarTextAppearance"/>

    <Button
        android:id="@+id/set_voice_btn"
        android:layout_width="172dp"
        android:layout_height="53dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="180dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:onClick="setVoiceOnClick"
        android:text="CAMBIAR VOZ"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/toolbar"
        app:layout_constraintVertical_bias="0.349" />

    <Button
        android:id="@+id/speech_test_btn"
        android:layout_width="172dp"
        android:layout_height="53dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="180dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:onClick="speechTestOnClick"
        android:text="PRUEBA"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/toolbar"
        app:layout_constraintVertical_bias="0.057" />

    <EditText
        android:id="@+id/phrase_editText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
```

```
        android:layout_marginTop="52dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:ems="10"
        android:inputType="textPersonName"
        android:shadowColor="@color/colorPrimary"
        android:text="Frase"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.503"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/toolbar"
        app:layout_constraintVertical_bias="0.08" />

    <ProgressBar
        android:id="@+id/progressBar"
        style="?android:attr/progressBarStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="444dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/toolbar"
        app:layout_constraintVertical_bias="0.0" />


</androidx.constraintlayout.widget.ConstraintLayout>
```