

# SAPLM

## Sistema Auxiliar para Personas con Limitaciones Motrices

**Escuela de Educación Técnica N°9  
Curso 7º1ª 2019**

**Desarrolladores**  
Da Cruz, Agustín  
Villegas, Rocío  
Zatloukal Maule, Julián

**Profesores implicados**  
Otero, Diego  
Castro, Pujol  
Della Paolera, Sergio

# Índice

<b>1. Abstract</b>	<b>3</b>
<b>2. Diagrama en bloques</b>	<b>4</b>
<b>3. Diagrama de flujo para encender estufa</b>	<b>5</b>
<b>4. Comandos de voz disponibles</b>	<b>6</b>
<b>5. Diagrama de flujo</b>	<b>7</b>
<b>6. Imagenes de la aplicación</b>	<b>8</b>
<b>7. Carpeta de campo</b>	<b>11</b>
<b>8. ComposeCommand diagrama de flujo</b>	<b>14</b>
<b>9. DecomposeCommand diagrama de flujo</b>	<b>17</b>
<b>10.Circuito impreso, módulo principal</b>	<b>20</b>
<b>11.Circuito impreso, módulo de potencia</b>	<b>26</b>
<b>12.Circuito impreso, módulo de mandos motrices</b>	<b>32</b>
<b>13.Código de fuente, módulo principal</b>	<b>38</b>
<b>14.Código de fuente, módulo de potencia</b>	<b>72</b>
<b>15.Código de fuente, módulo de mandos motrices</b>	<b>102</b>
<b>16.Código de fuente, aplicación SAPLM</b>	<b>134</b>
<b>17.Código de fuente, Diseño de activities</b>	<b>222</b>

# TECNICA-MENTE 2019 – CICLO SUPERIOR

ESCUELA DE EDUCACIÓN SECUNDARIA TECNICA N°: 9 Antonio José Rodríguez.

DISTRITO: Lanús.

REGIÓN: 2.

TÍTULO DEL PROYECTO: Sistema Auxiliar para Personas con Limitaciones Motrices.

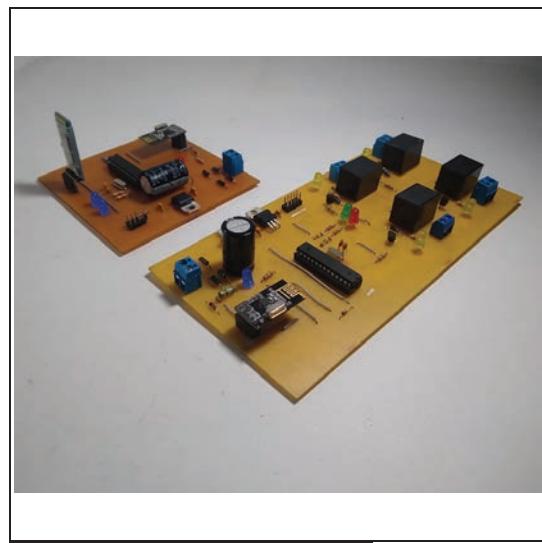
Equipo responsable: Zatloukal Maule, Julián; Da Cruz, Agustín Tomás; Villegas Cabral, Rocío.

Docente Tutor: Otero, Diego; Castro, Pujol; Della Paolera, Sergio.

## 1. Objetivo del Proyecto

Las relaciones cercanas con personas de capacidades limitadas nos llevaron a plantearnos la idea de implementar una ayuda para mejorar su calidad de vida a partir de nuestros conocimientos como técnicos electrónicos. A principio de año, junto a nuestros profesores, ideamos un servicio sistema para beneficiarlos; al cual luego lo bautizamos **Sistema Auxiliar para Personas con Limitaciones Motrices (SAPLM)**.

El objetivo que quisiéramos alcanzar consta en facilitar su vida cotidiana a través de la simplificación de los mandos y controles de sistemas tanto eléctricos como electromecánicos hogareños sin la necesidad de un tercero que lo asista en estas tareas, como podría ser desde un simple encendido de luces hasta regular la calefacción mediante un simple comando por voz procesado por teléfono celular.



## 2. Descripción del Proyecto

Este sistema se compone en primera instancia de un teléfono móvil Android, el cual el paciente dispone en todo momento, en este se reciben y procesan los comandos de voz mediante la aplicación diseñada, programada y desarrollada por nosotros. Esta aplicación fue desarrollada en Android Studio, el programa oficial de Android para desarrollo de aplicaciones móviles, y programada en lenguaje Java. Ésta integra dos librerías las cuales resuelven el reconocimiento de voz, pocket-sphinx y la API oficial de Google destinada a este propósito.

Estos comandos luego de ser procesados en la aplicación se envían mediante Bluetooth a lo que nosotros denominamos la placa principal o maestra. La cual, inmediatamente a través de un sistema de radiofrecuencia, retransmite el comando recibido a la placa de mando correspondiente. Estas llamadas placas de mando son aquellas que interactuar directamente con el dispositivo o maquinaria a controlar, reciben un simple comando mediante el sistema de radiofrecuencia enviado desde la placa central y cumplen la orden.

Concretamente en nuestro proyecto se exponen dos placas de mando, una al cual la llamamos "Placa de mandos de potencia" y a la otra "Placa de mandos motrices". Ambas se componen esencialmente del modulo de radiofrecuencia junto a un microcontrolador de 8 bits de la familia AVR y de todos los elementos para la regulación del voltaje. La "Placa de mandos de potencia" dispone de 4 contactos electromecánicos los cuales controlaran elementos simples, como una estufa, ventilador, luz, velador u otros. En cuanto a la "Placa de mandos motrices" esta dispone de dos drivers para motores paso a paso los cuales manejaran una camilla y una cortina.

El proyecto está planteado de manera extensible de manera que soporte la adición de nuevas placas de mando al sistema ya planteado; luego de una correspondiente actualización del firmware de la placa principal.

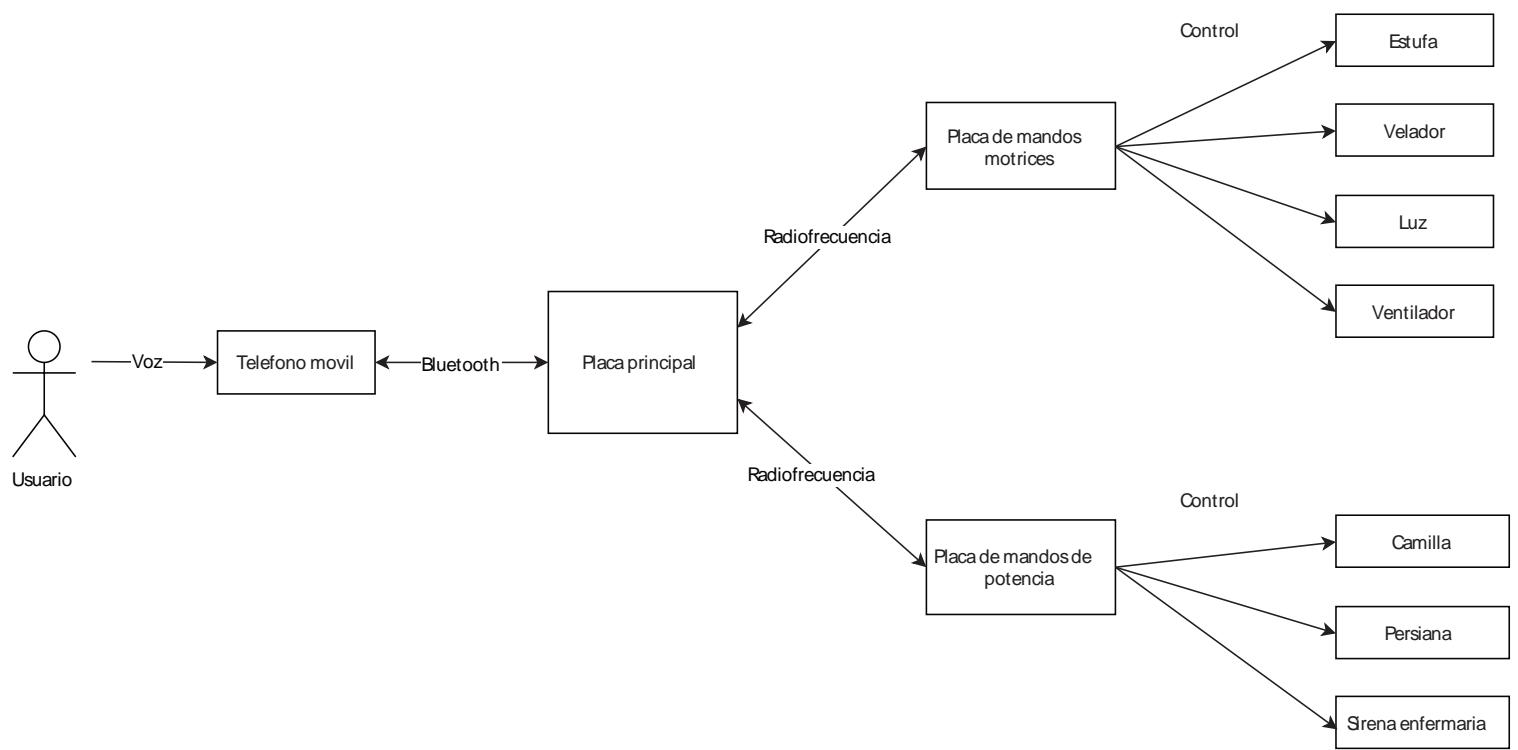
## 3. ESTADO DE DESARROLLO DEL PROYECTO:

Luego de cuatro meses de desarrollo nos encontramos con una firme base del proyecto ya armada. Largas tardes y noches resultaron en una aplicación capaz de recibir comandos de voz con una exactitud competente y de transmitirlos mediante el protocolo Bluetooth. Los firmware de las placas ya están asentados.

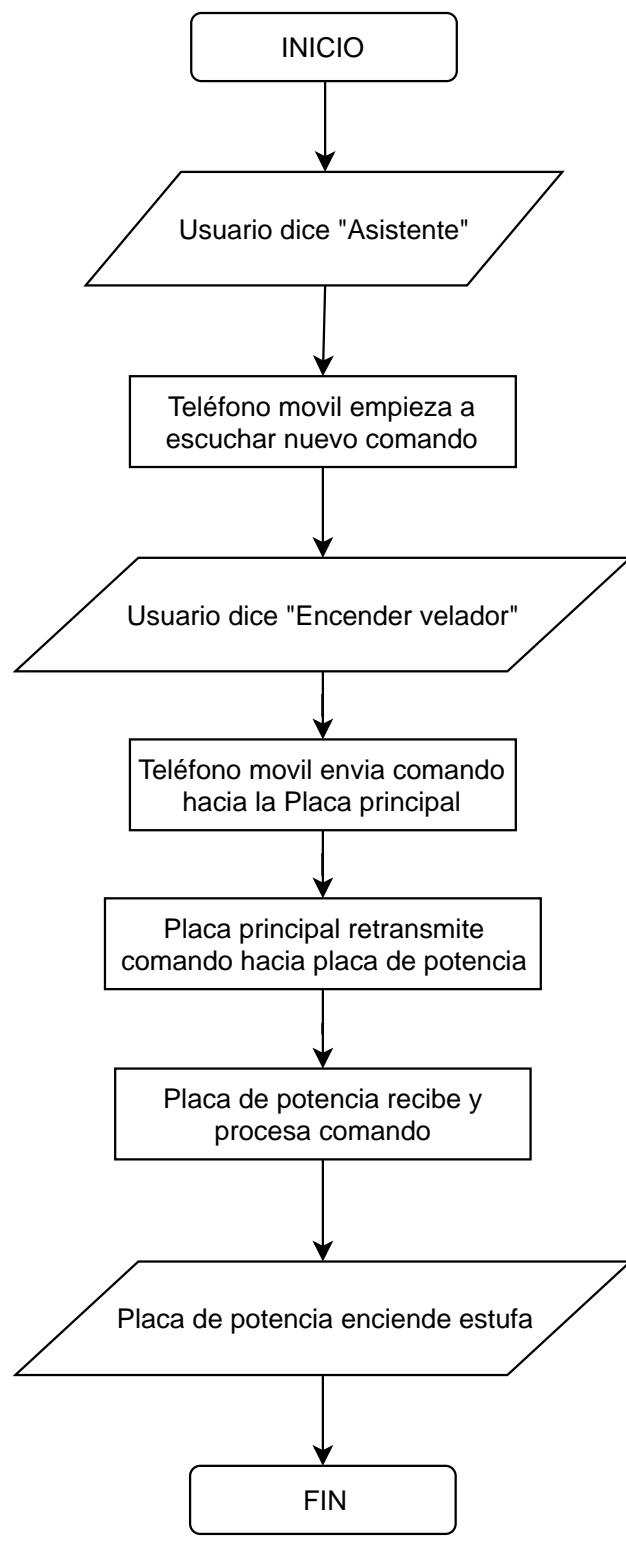
Todas las placas se encuentran diseñadas pero por el momento solo construimos dos de ellas, las cuales funcionan a la perfección. La placa principal sufrió un rediseño luego de que en una jornada de trabajo caigamos en la cuenta de que esta tenía una configuración de conexiones errónea. Se decidió construir primero la "Placa de mandos de potencia" debido a su simplicidad, esta no trajo dificultades al armado y sus modificaciones fueron mínimas luego de ser completada.

Con ambas placas en marcha y una aplicación funcional el sistema esta listo para ser utilizado en situaciones sencillas como prender una estufa o luz; sin embargo no lo son para alguien con capacidades diferentes.

# Diagrama de bloques



# Diagrama de flujo para encender una estufa



# Comandos por voz

- Palabra clave : Asistente

## Comandos disponibles

Encender estufa	Apagar estufa
Encender velador	Apagar velador
Encender ventilador	Apagar ventilador
Llamar enfermería	

## Estados de escucha



Escucha  
deshabilitada

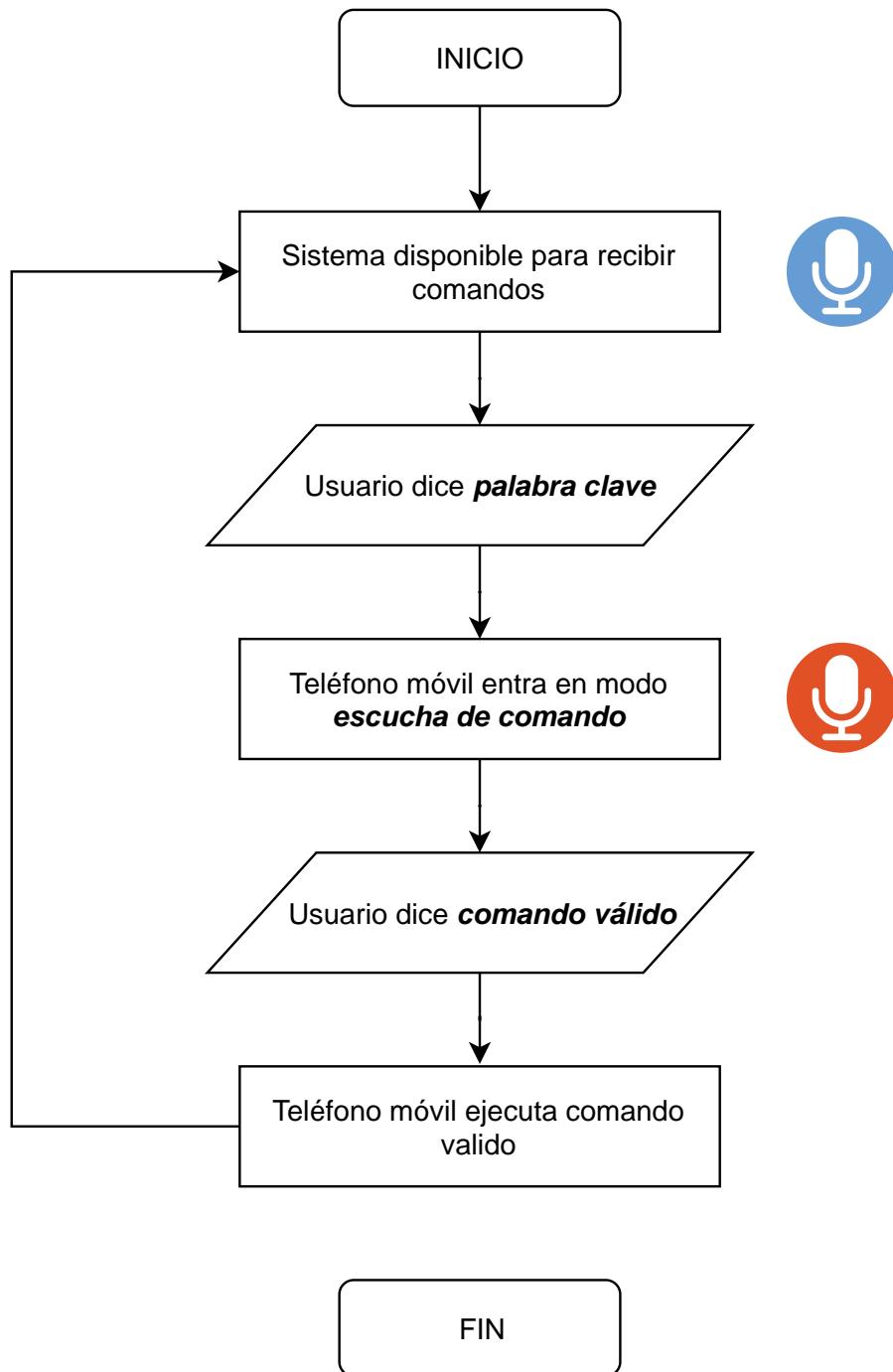


Escuchando  
palabra clave



Escuchando  
comando

# Diagrama de flujo generico para accionar comando



# Interfaz gráfica de la aplicación

- Pantalla principal



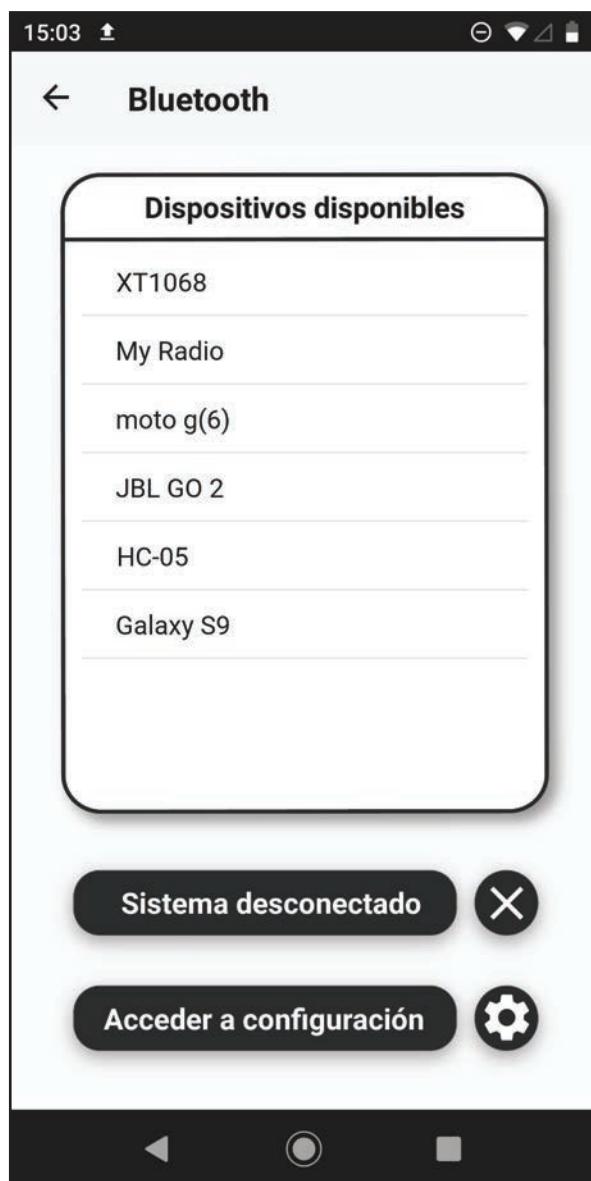
# Interfaz gráfica de la aplicación

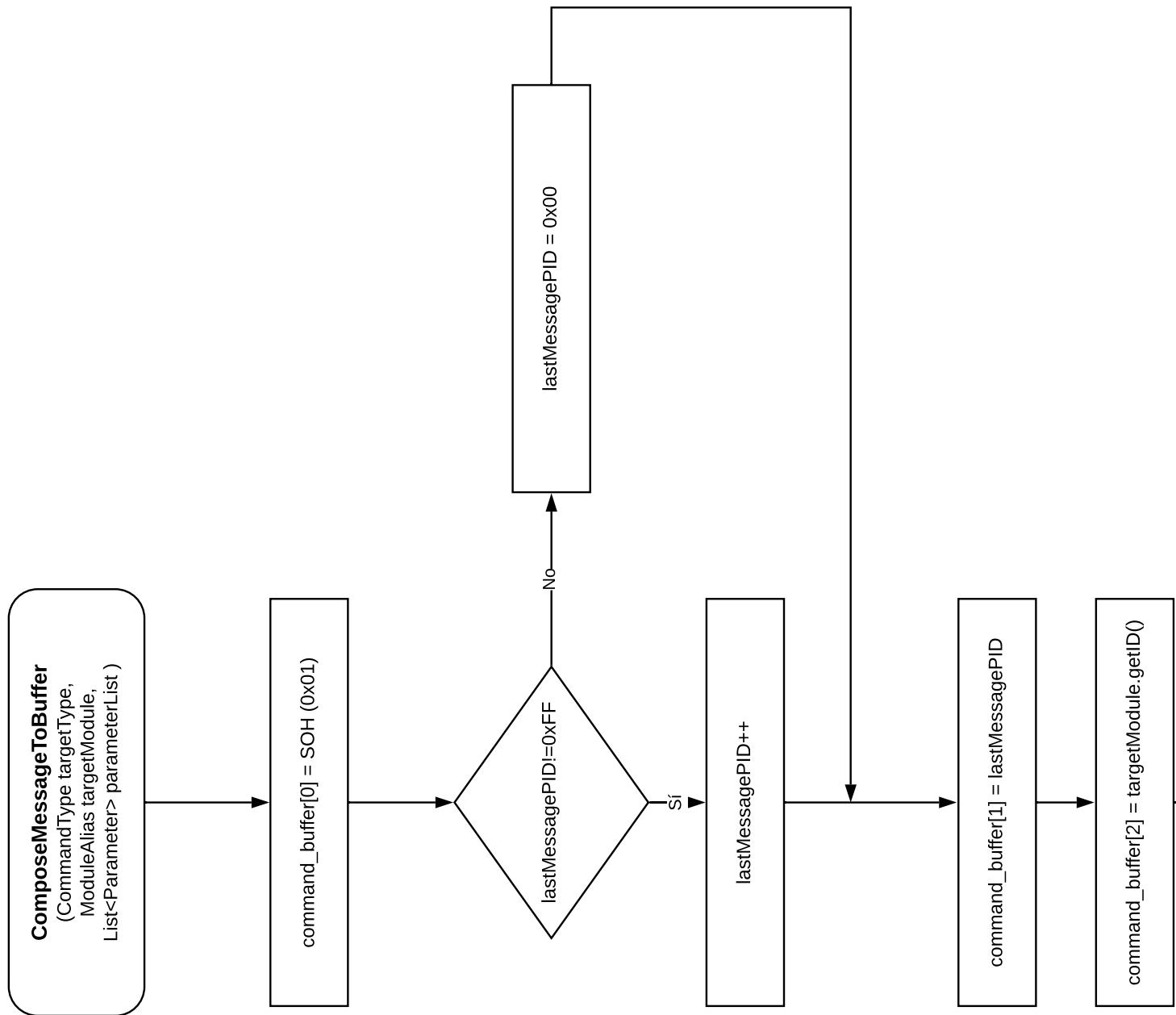
- **Menú de configuraciones**

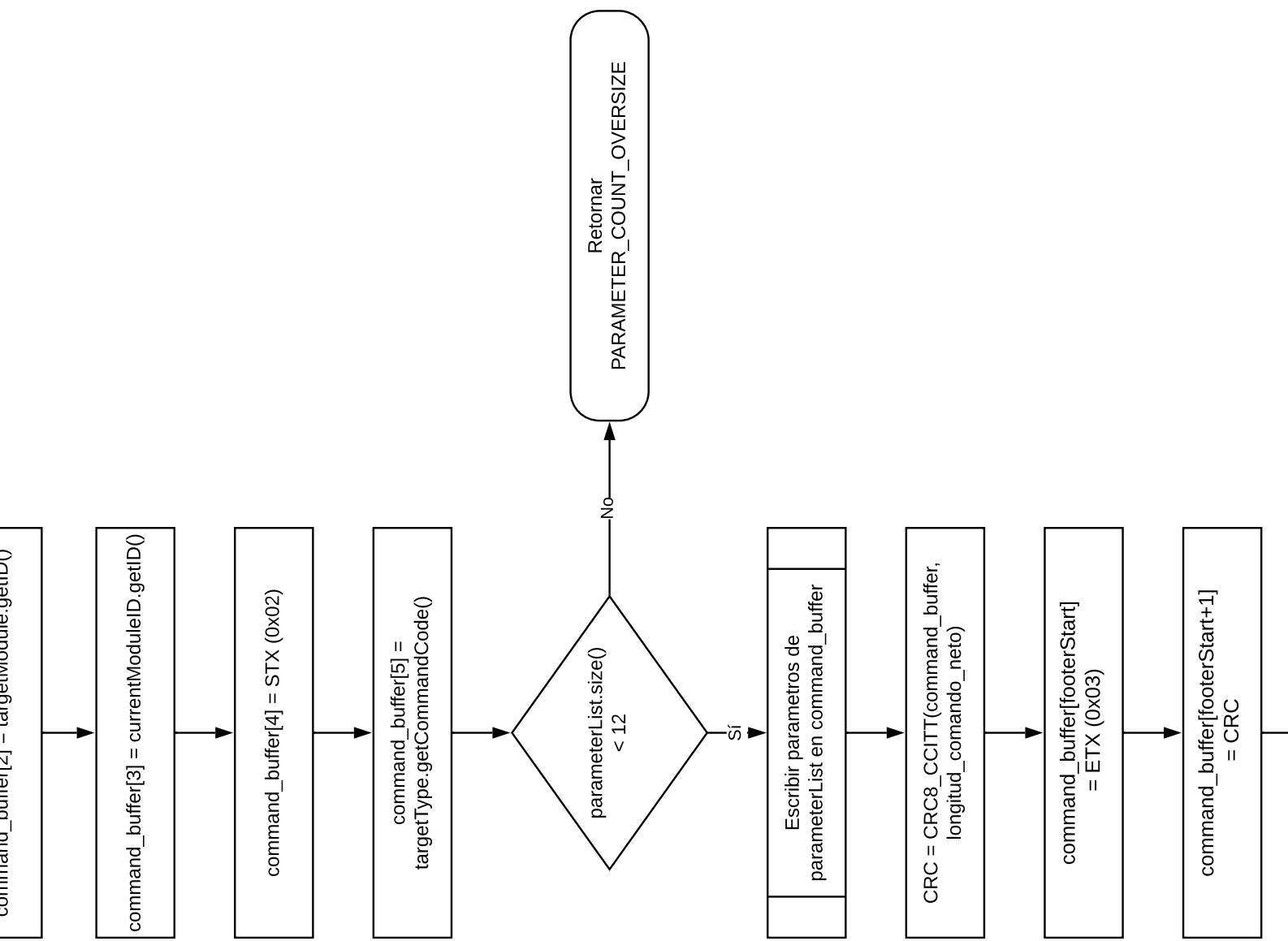


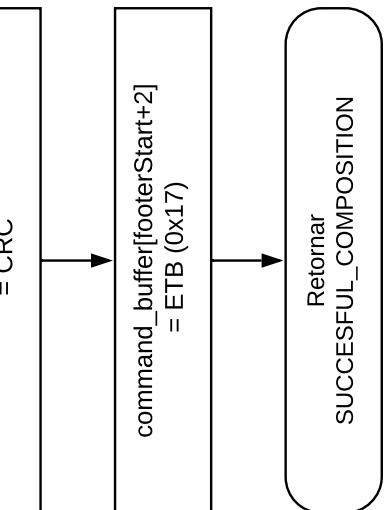
# Interfaz gráfica de la aplicación

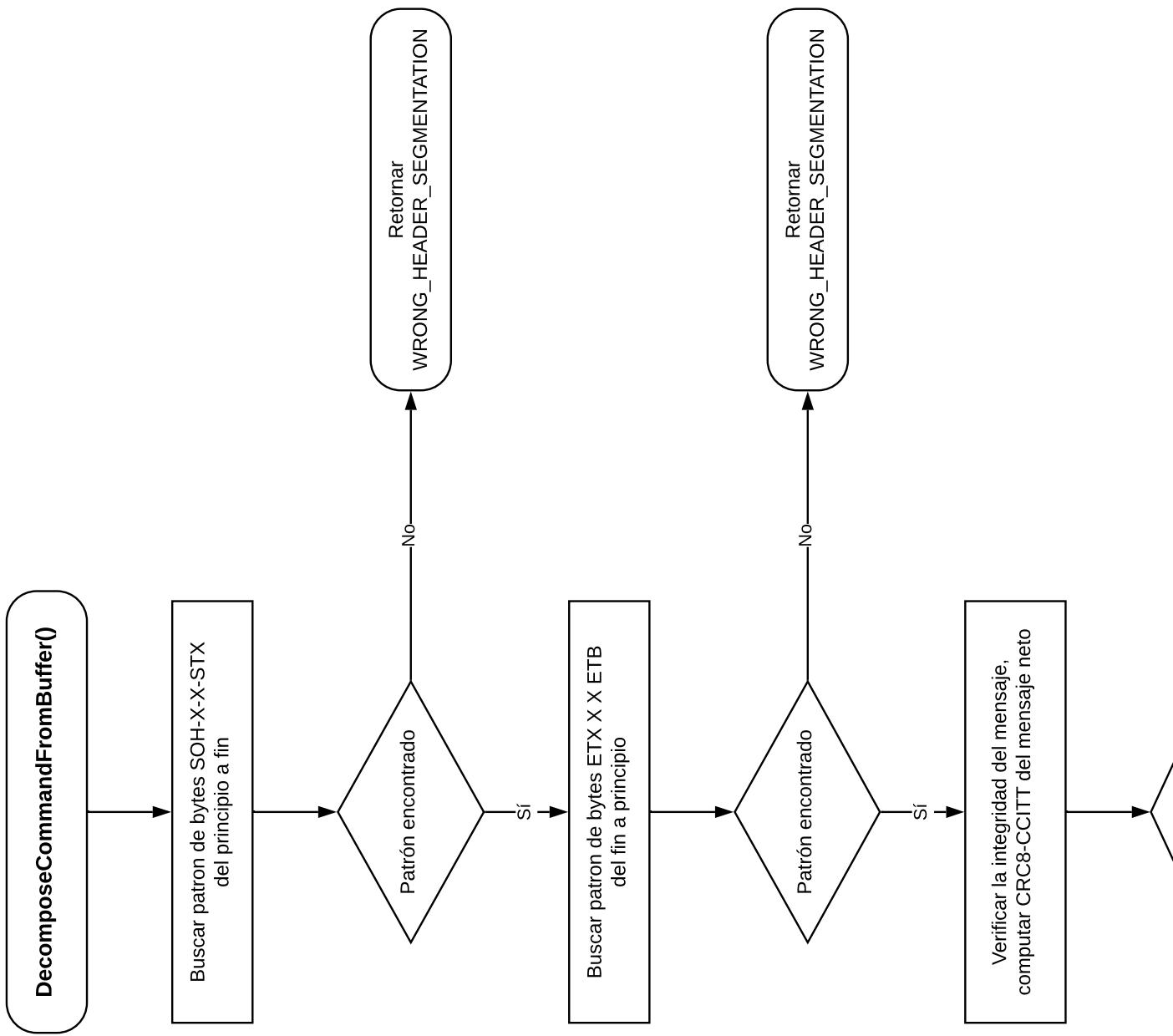
- Menu de Bluetooth

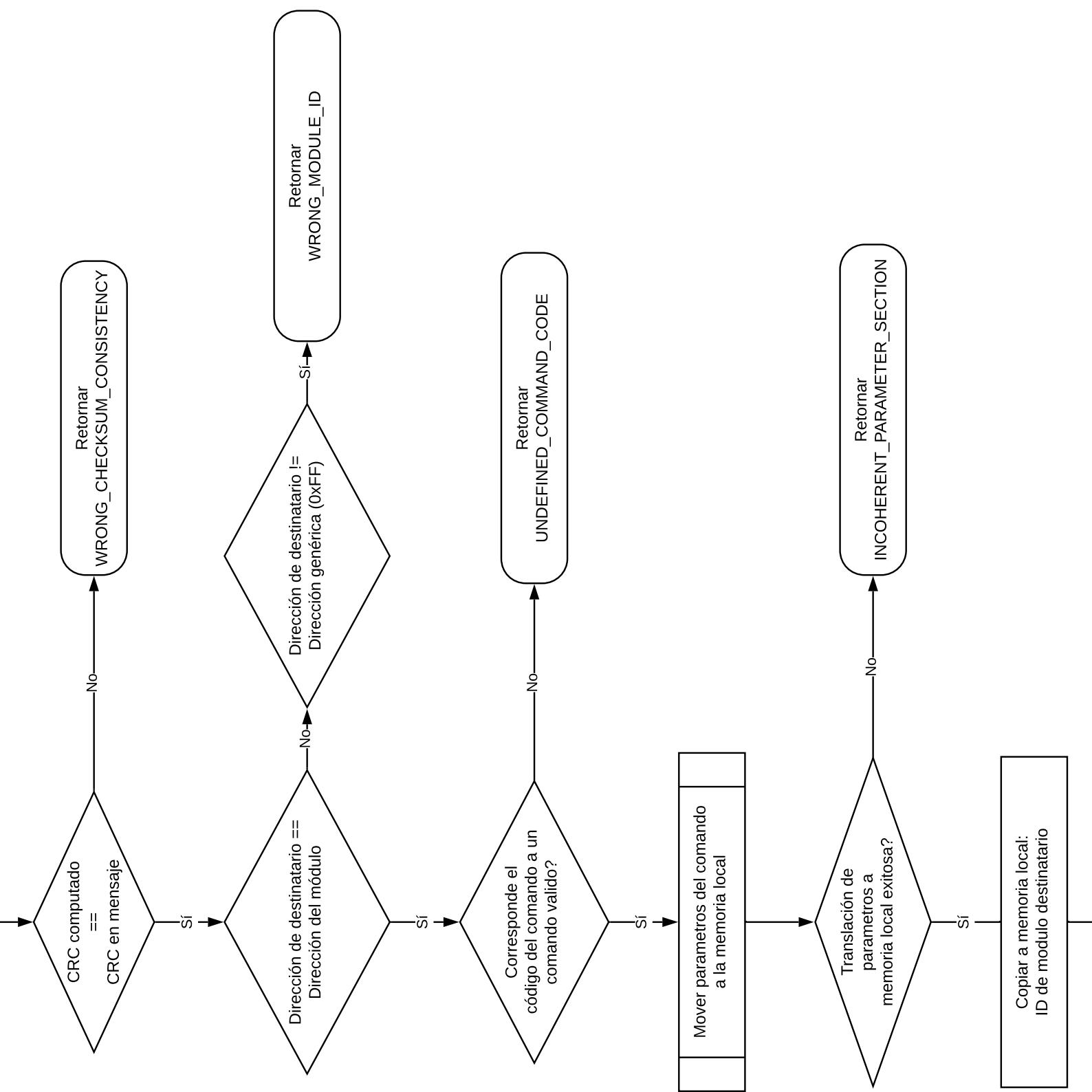




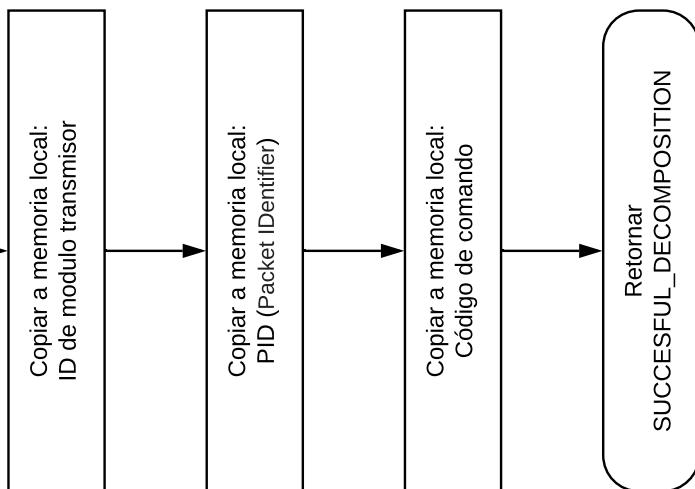








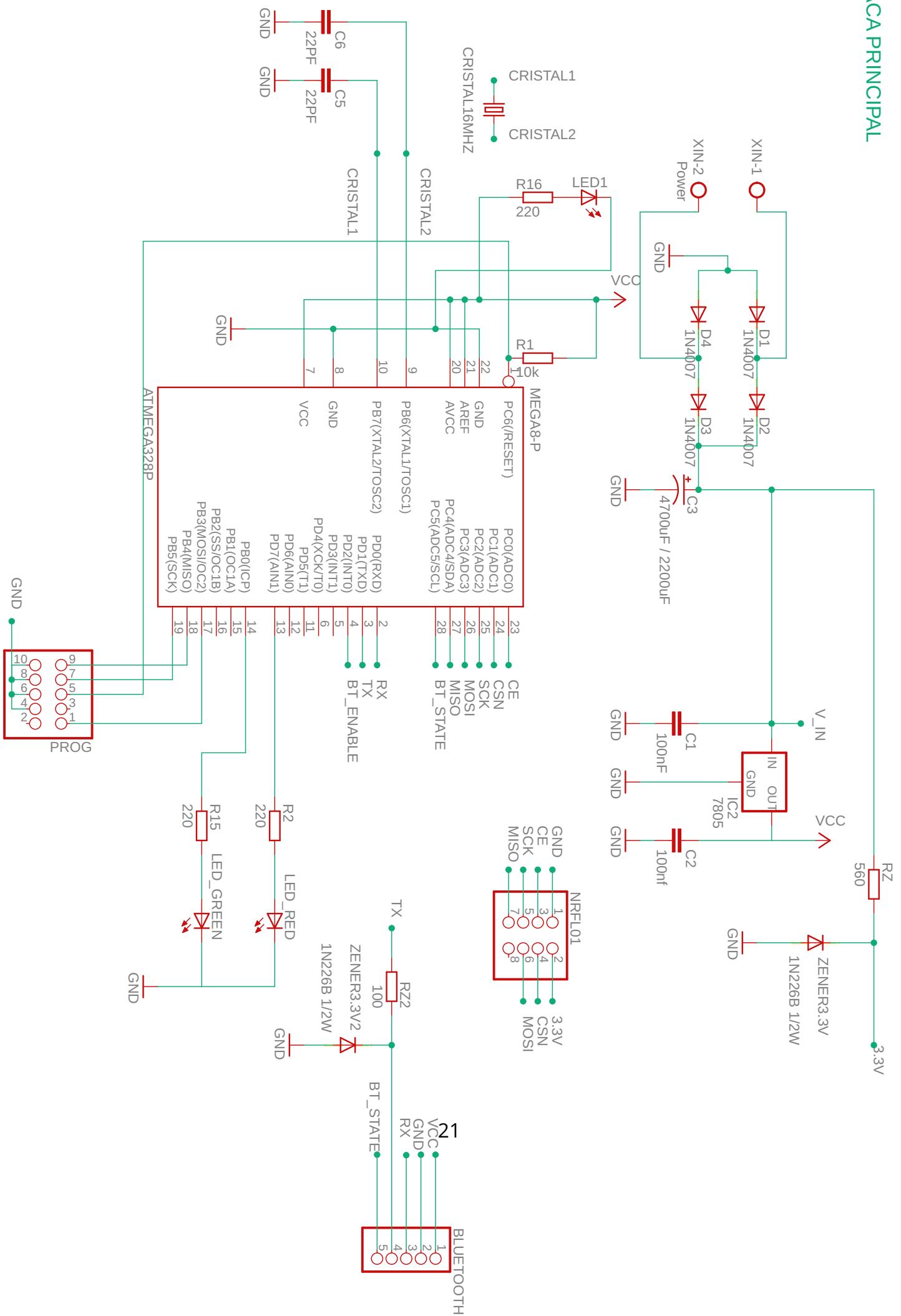
ID de módulo desuniditario

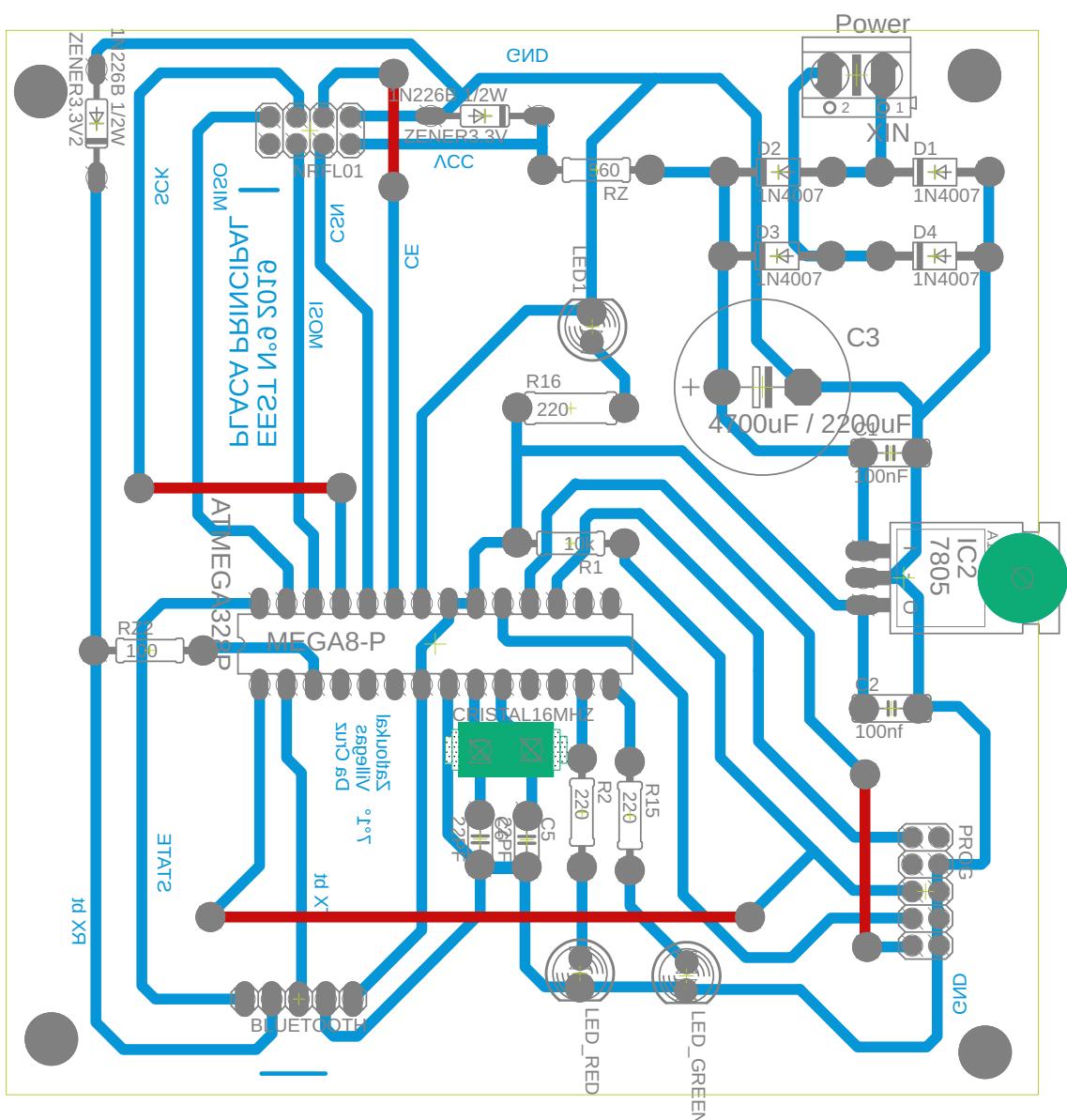


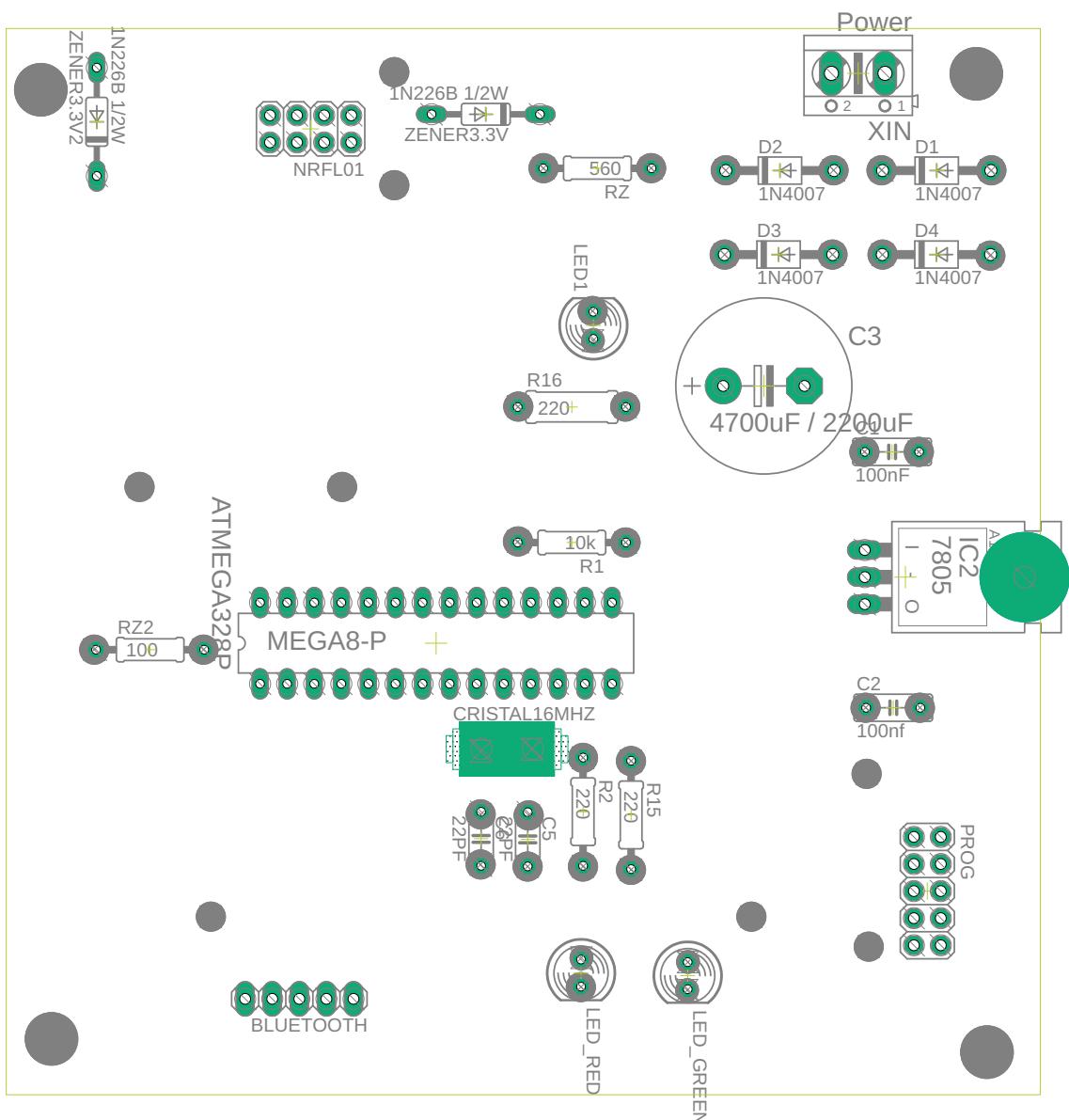
# • Circuito impreso: Módulo principal

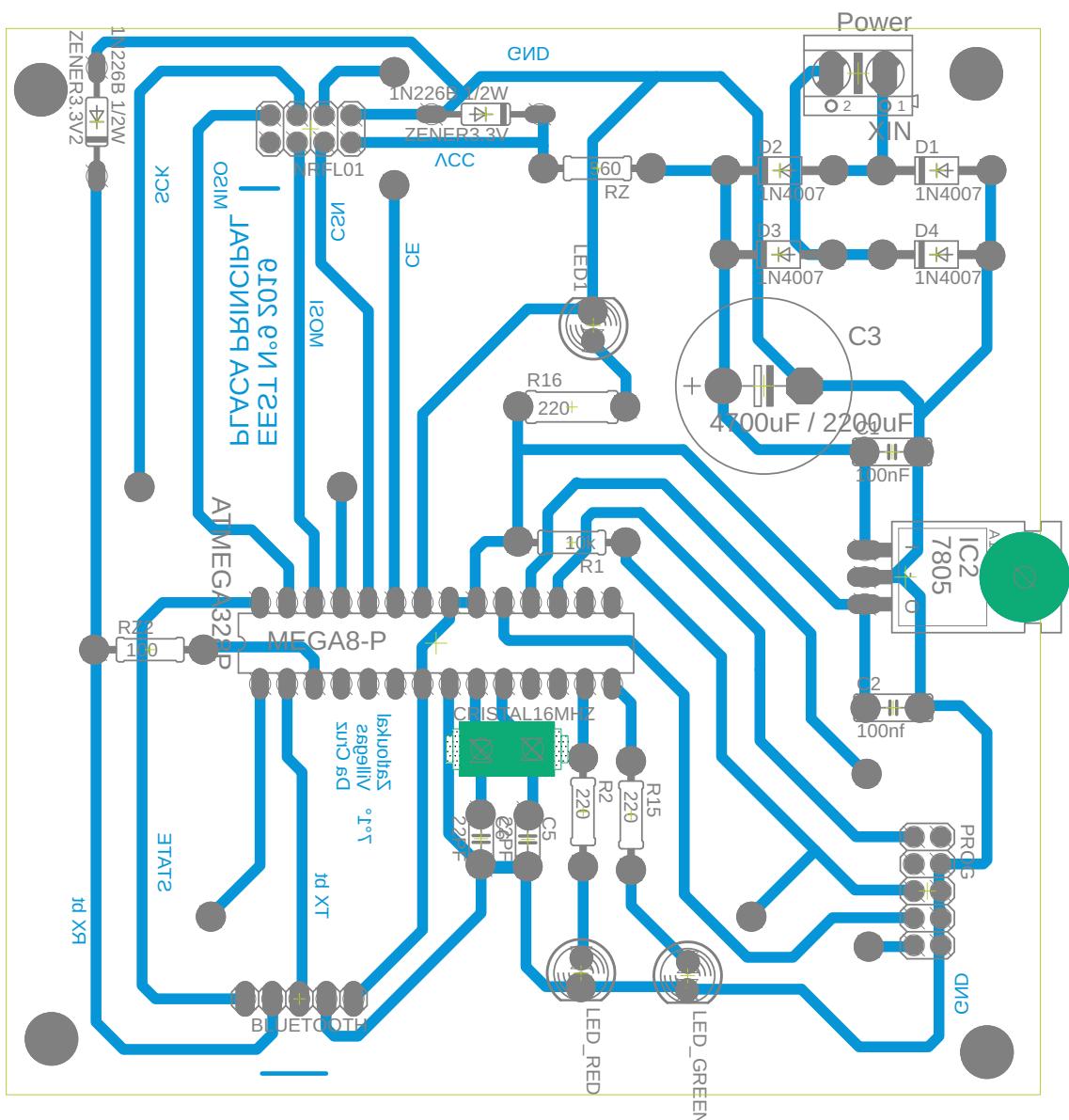
Lista de componentes	Precio x unidad	Cantidad
Modulo de radiofrecuencia emisor/receptor	\$ 100	1
Zocalo de 14x2	\$ 60	1
Capacitores Ceramicos	\$ 2.50	4
Leds	\$ 4.50	3
Diodos	\$ 5	4
Resistencias de 1/4 w	\$ 2.50	6
Capacitor Electrolitico	\$ 10	1
Cristal de 16Mhz	\$ 30	1
Diodos zener de 1/2 w	\$ 6	2
Regulador de tension	\$ 25	1
Tira de 25 pinhead	\$ 15	1
Borneras	\$ 15	1
Plaqueta virgen 10x10	\$ 90	1
Módulo Bluetooth HC-05	\$390	1
Total	\$ 803.50	

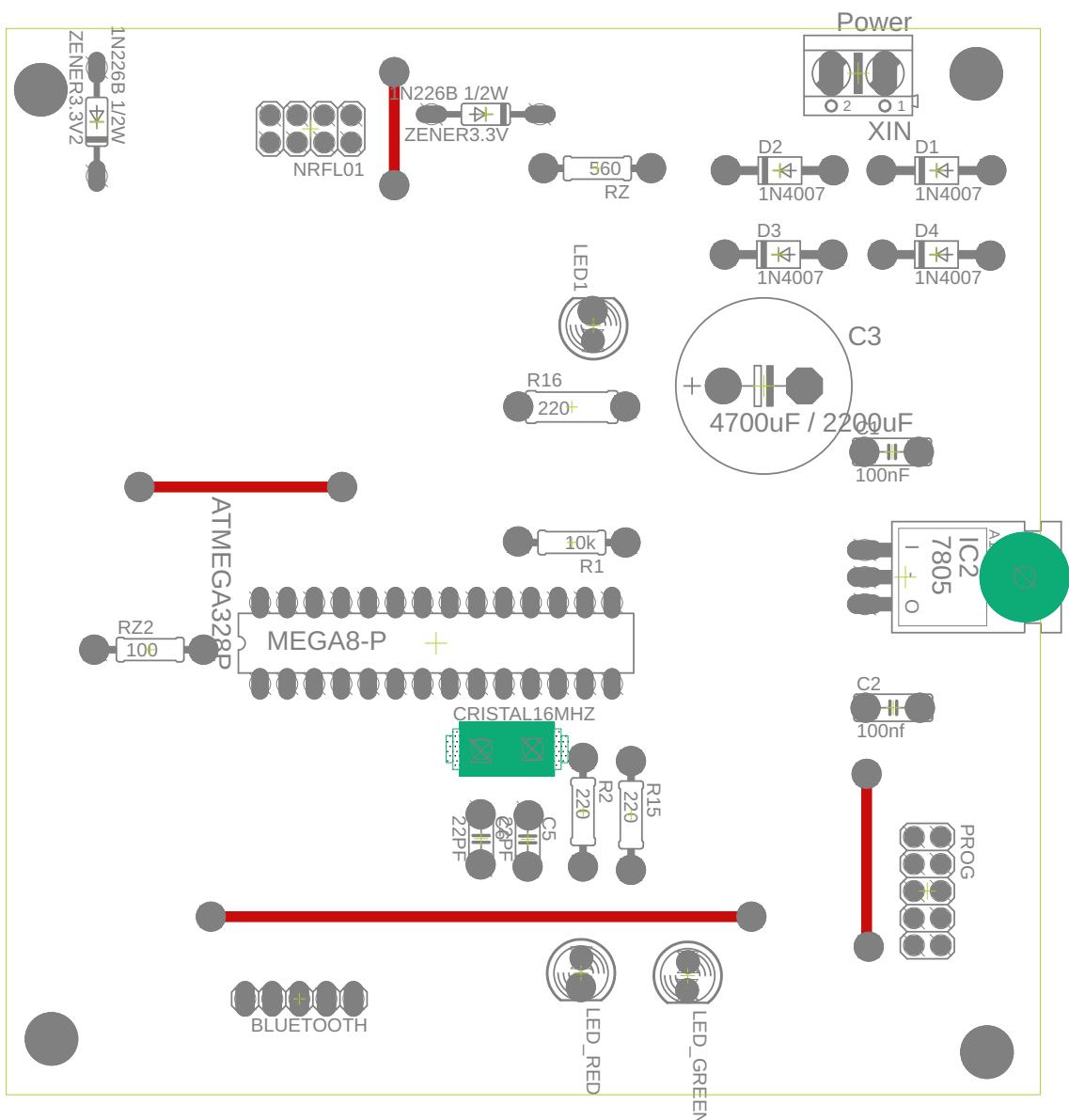
# PLACA PRINCIPAL





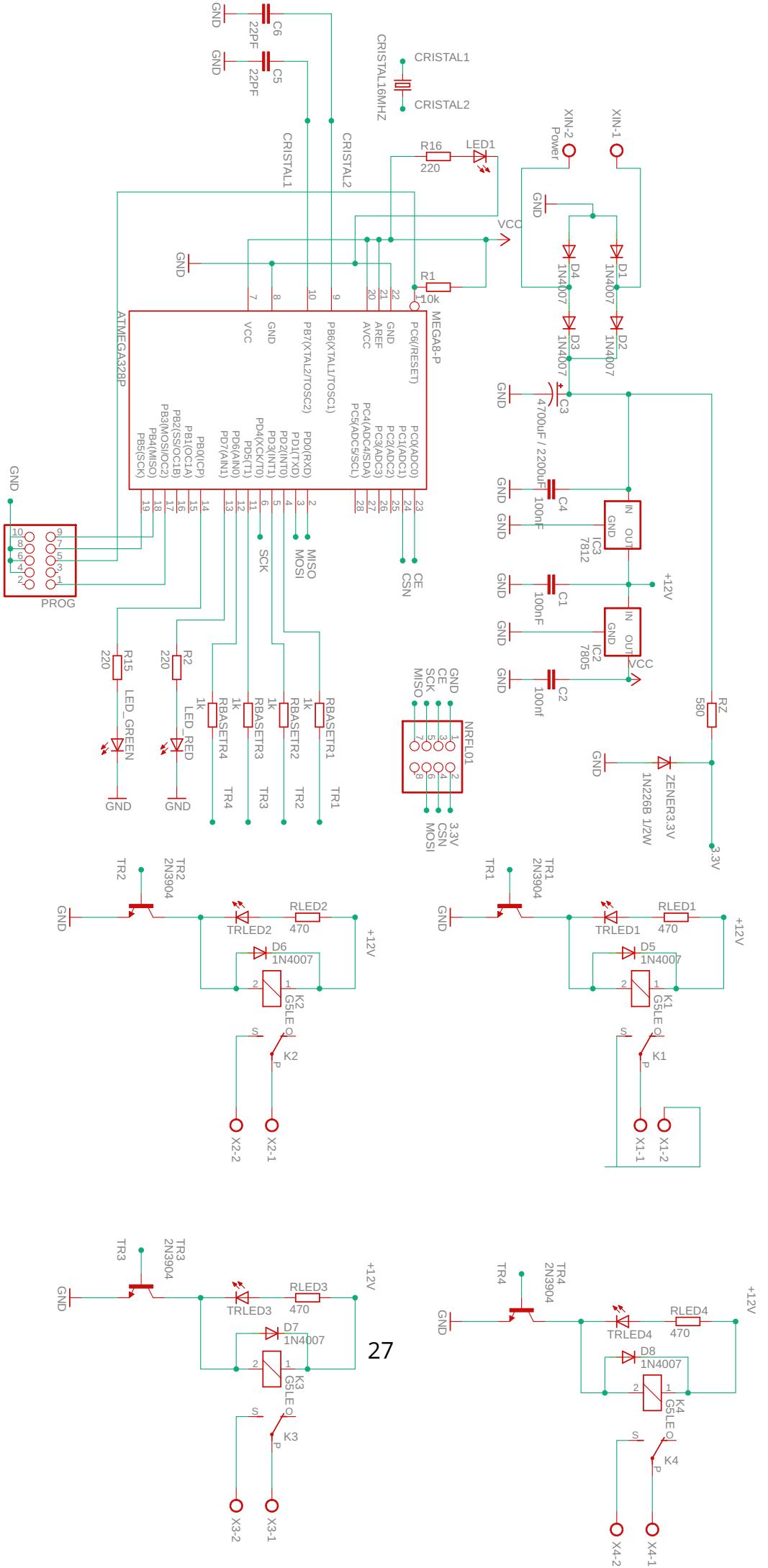


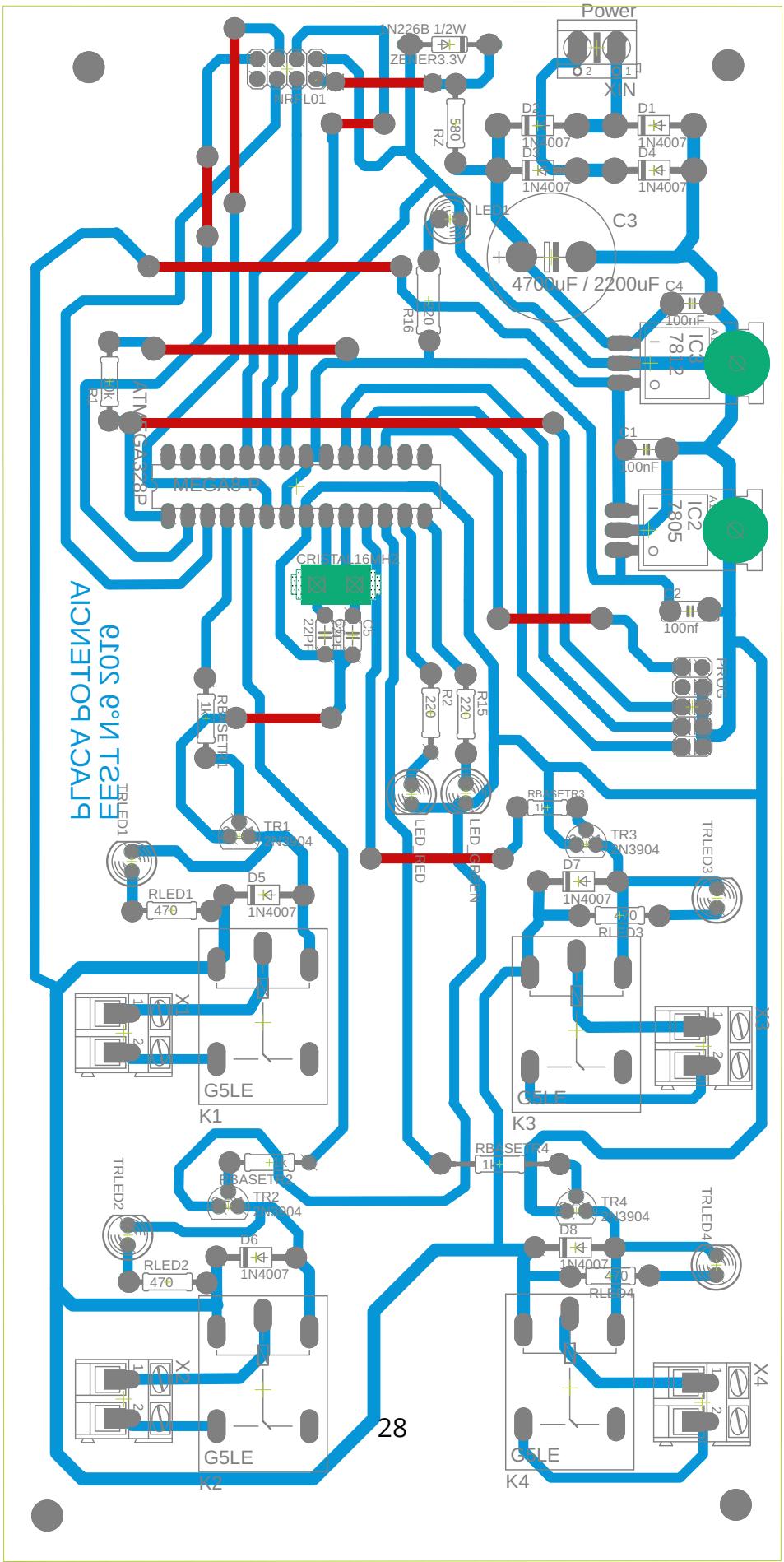


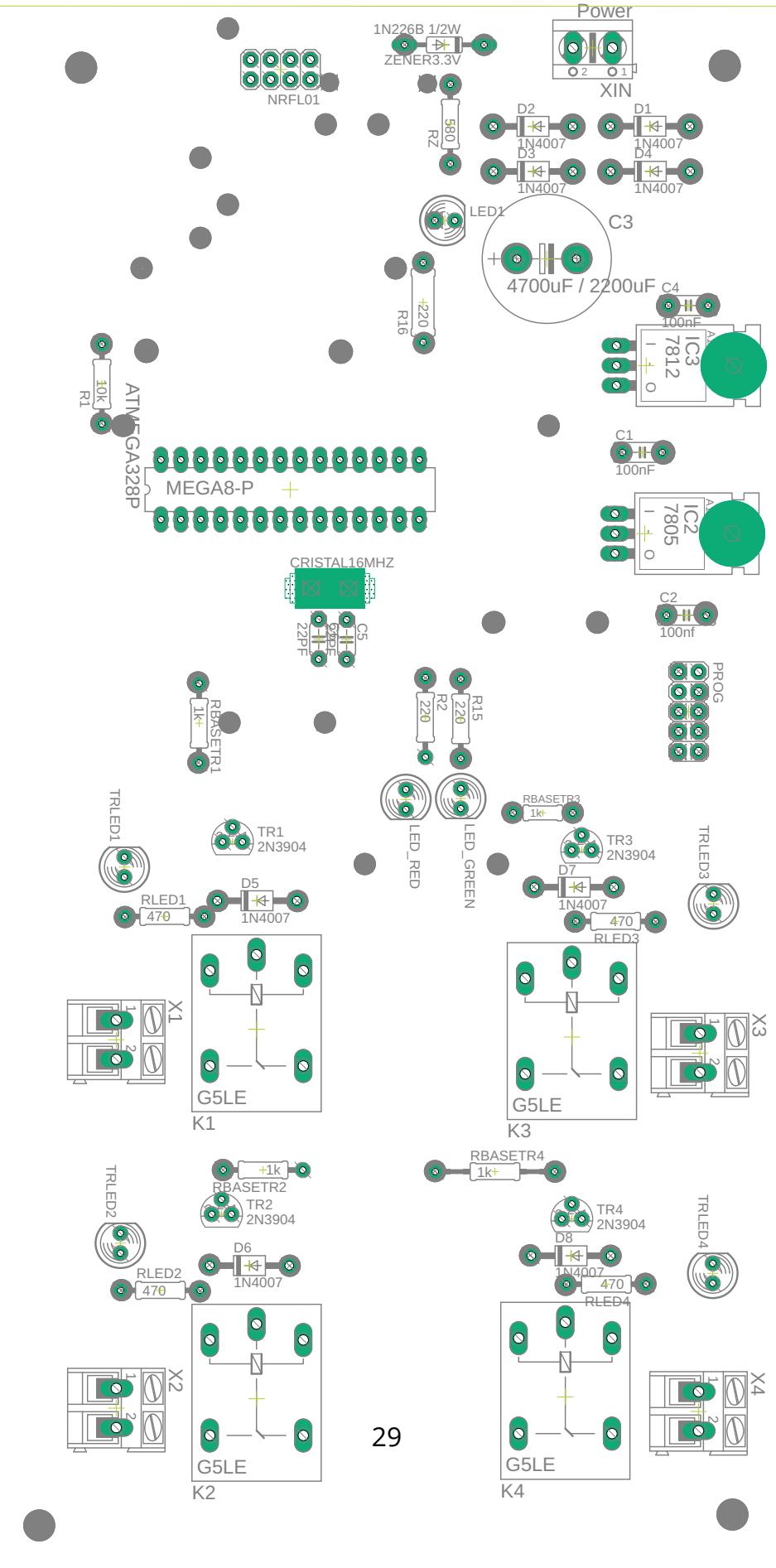


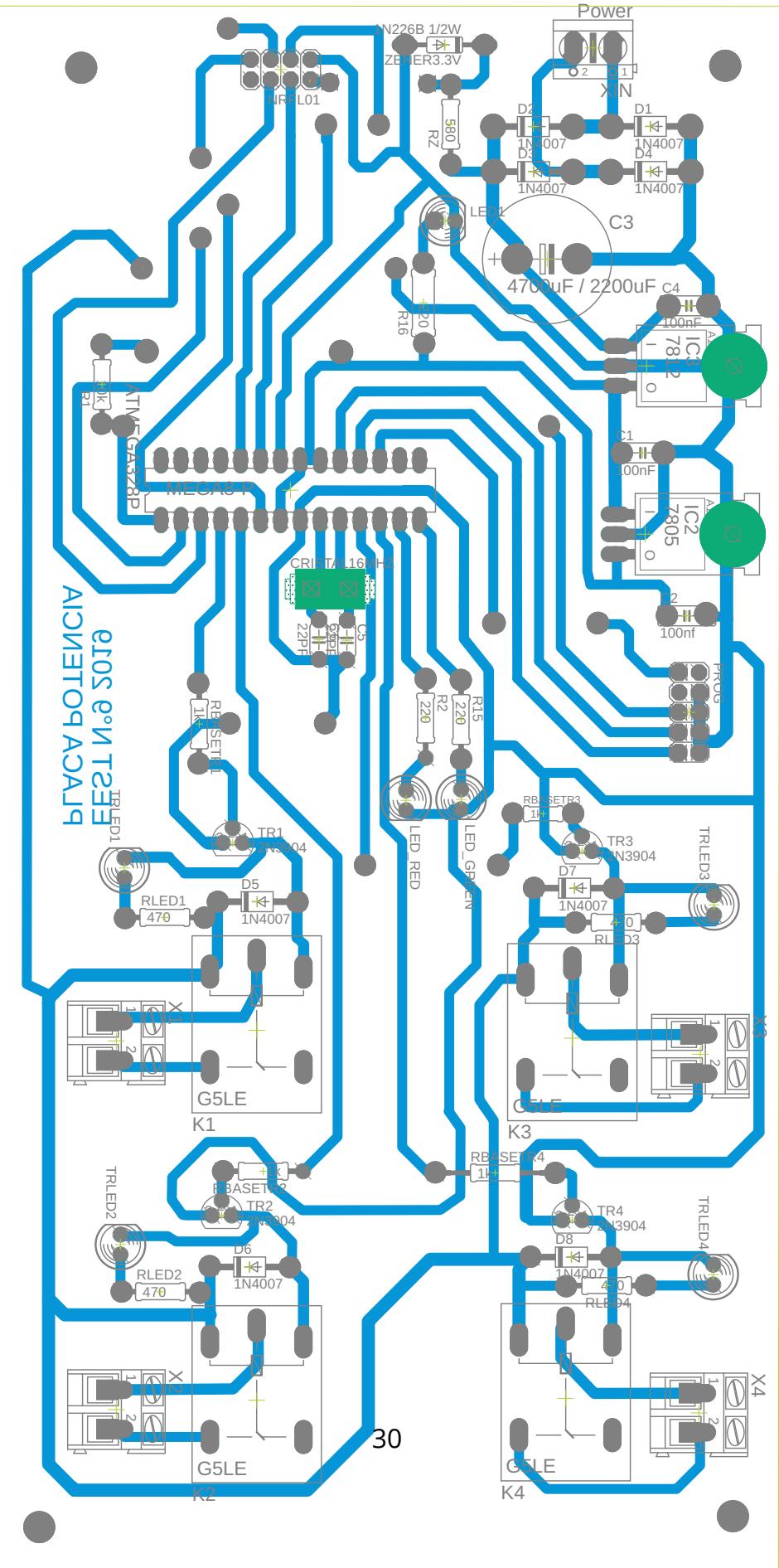
# • Circuito impreso: Módulo de mandos de potencia

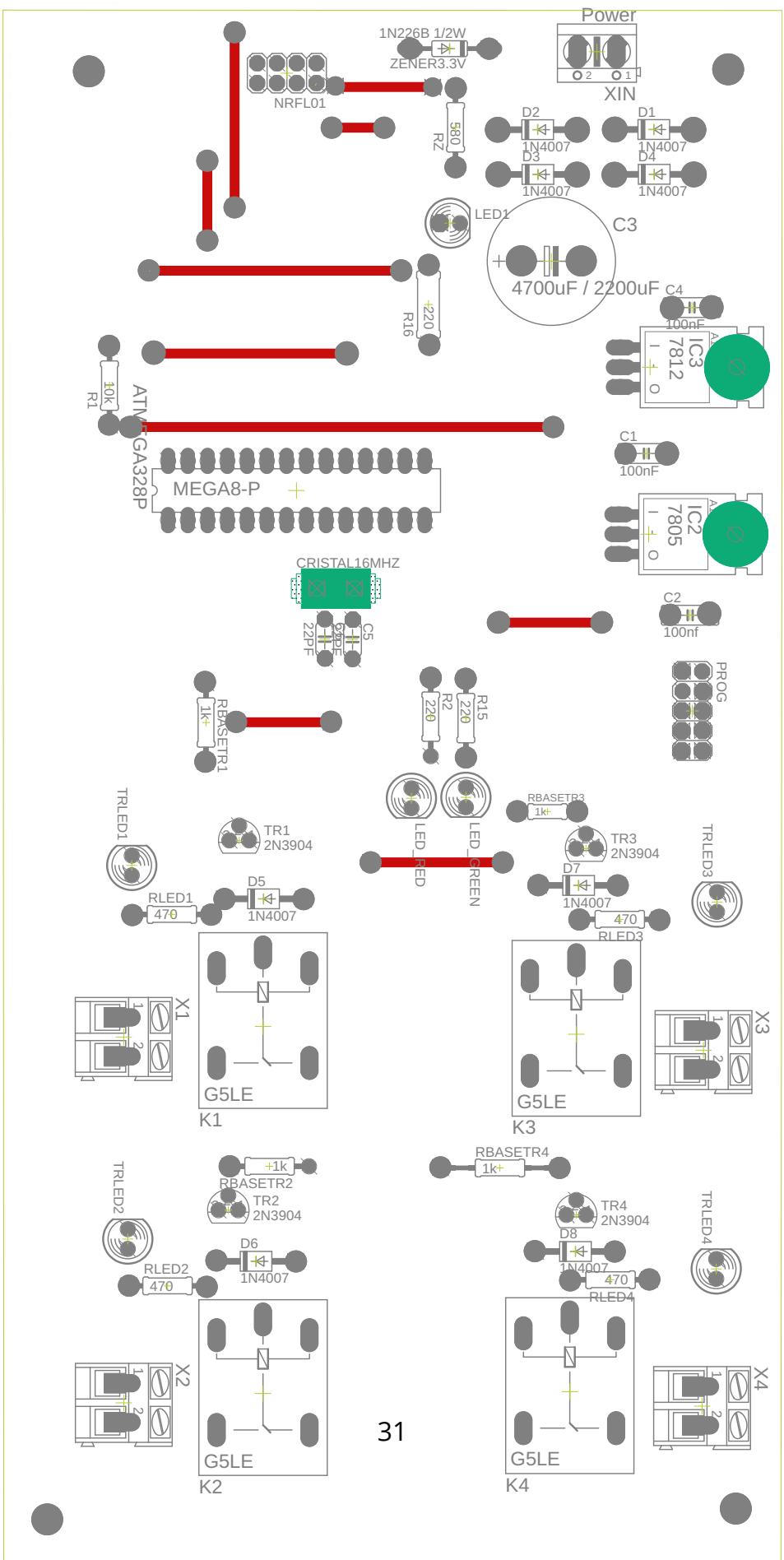
Lista de componentes	Cantidad	Precio x unidad
Capacitores ceramicos	5	\$ 2.50
Capacitor electrolico	1	\$ 10
Resistencias de 1/4 w	13	\$ 1.50
Cristal de 16Mhz	1	\$ 30
Leds de 5mm	7	\$ 4.50
Diodos	8	\$ 2.50
Diodo zener de 1/2 w	1	\$ 6
Borneras	4	\$ 15
Reguladores de tensión	5	\$ 25
Modulo de radiofrecuencia emisor/receptor	1	\$ 100
Microcontrolador atmega 328p	1	\$ 250
Relays	1	\$ 40
Tira de 20 pinhead	1	\$ 15
Zocalo 14x2	1	\$ 60
Plaqueta virgen de 20x10	1	\$ 150
Transistores	4	\$ 40
	Total	\$ 1,089.50







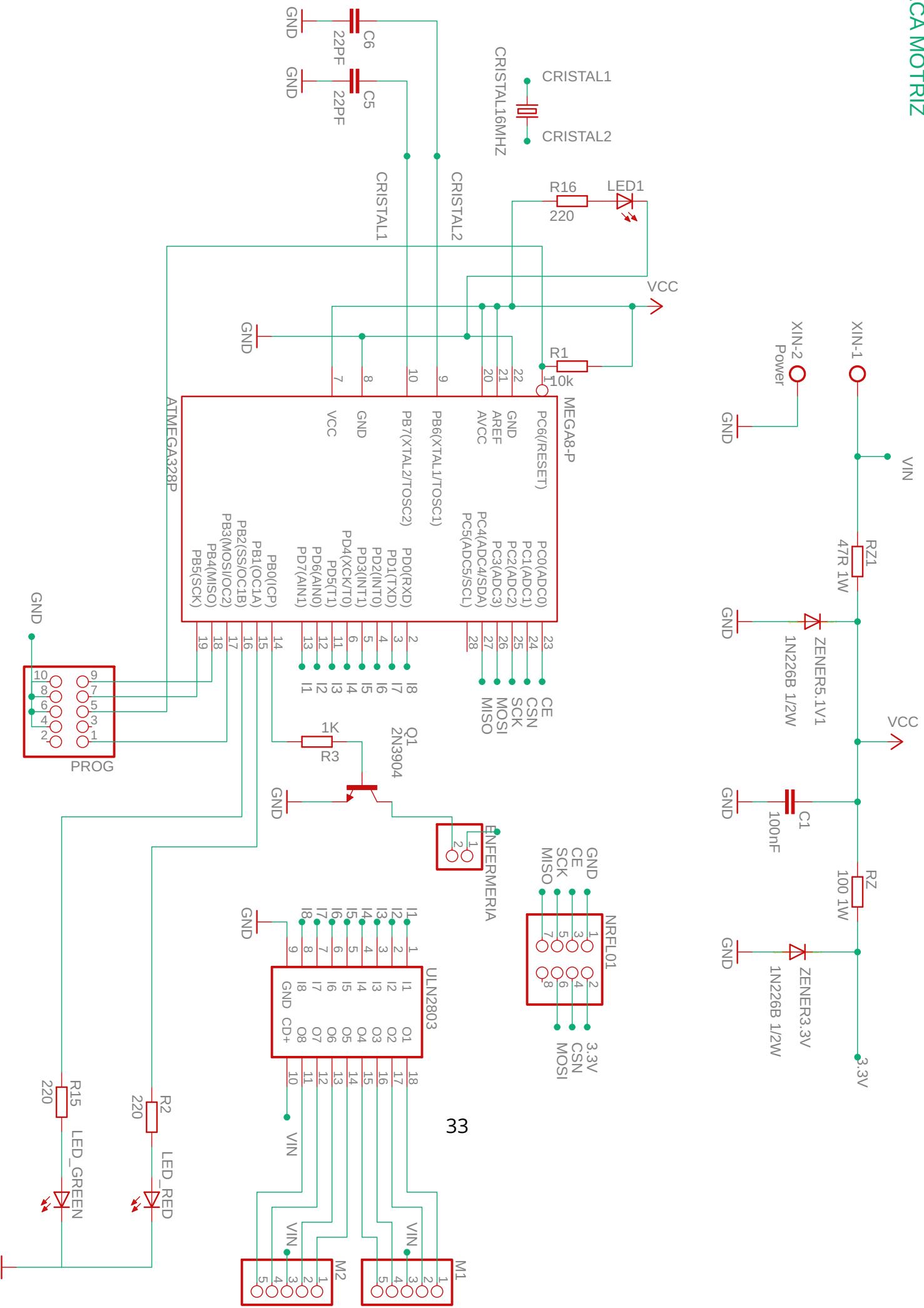


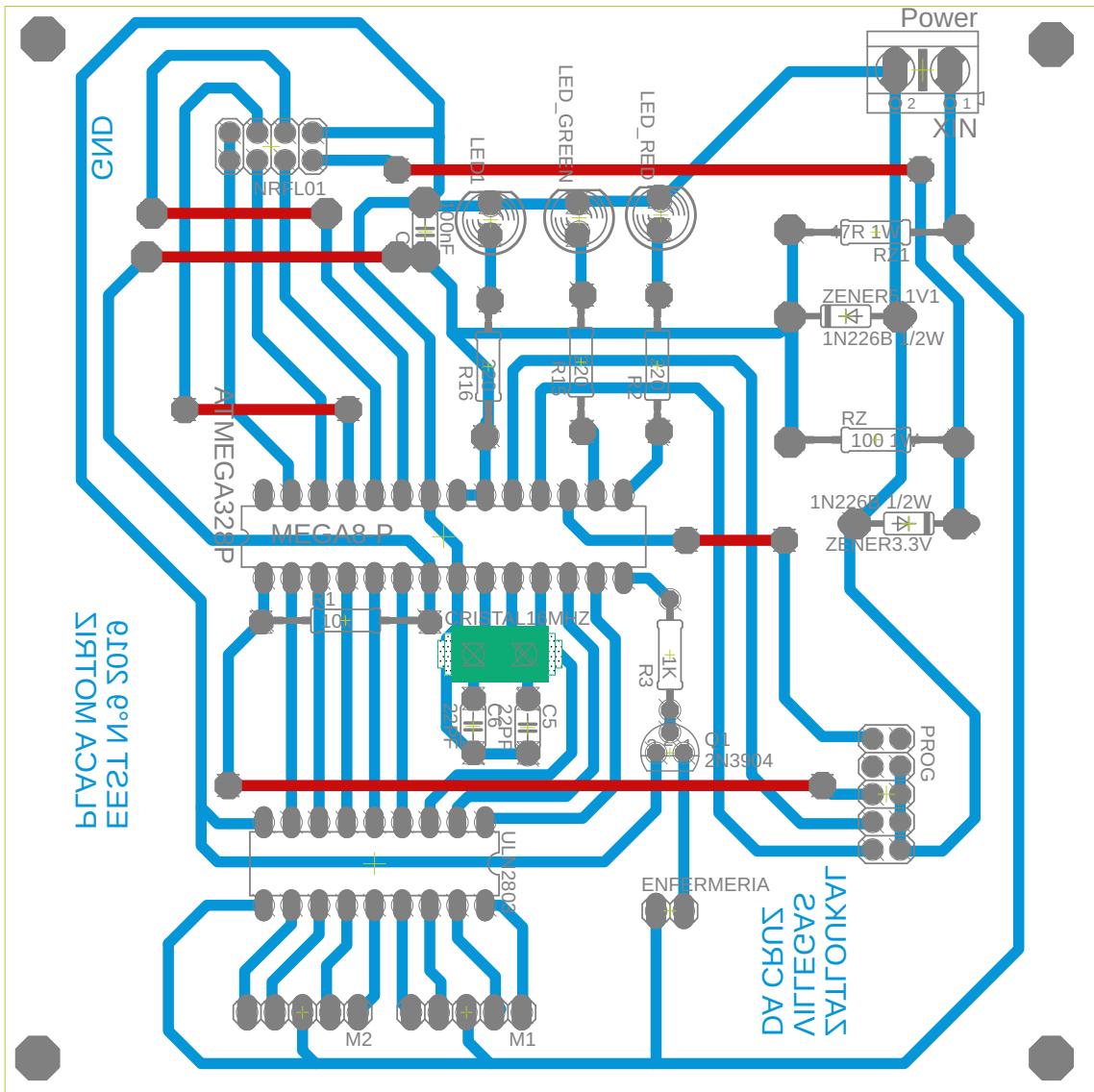


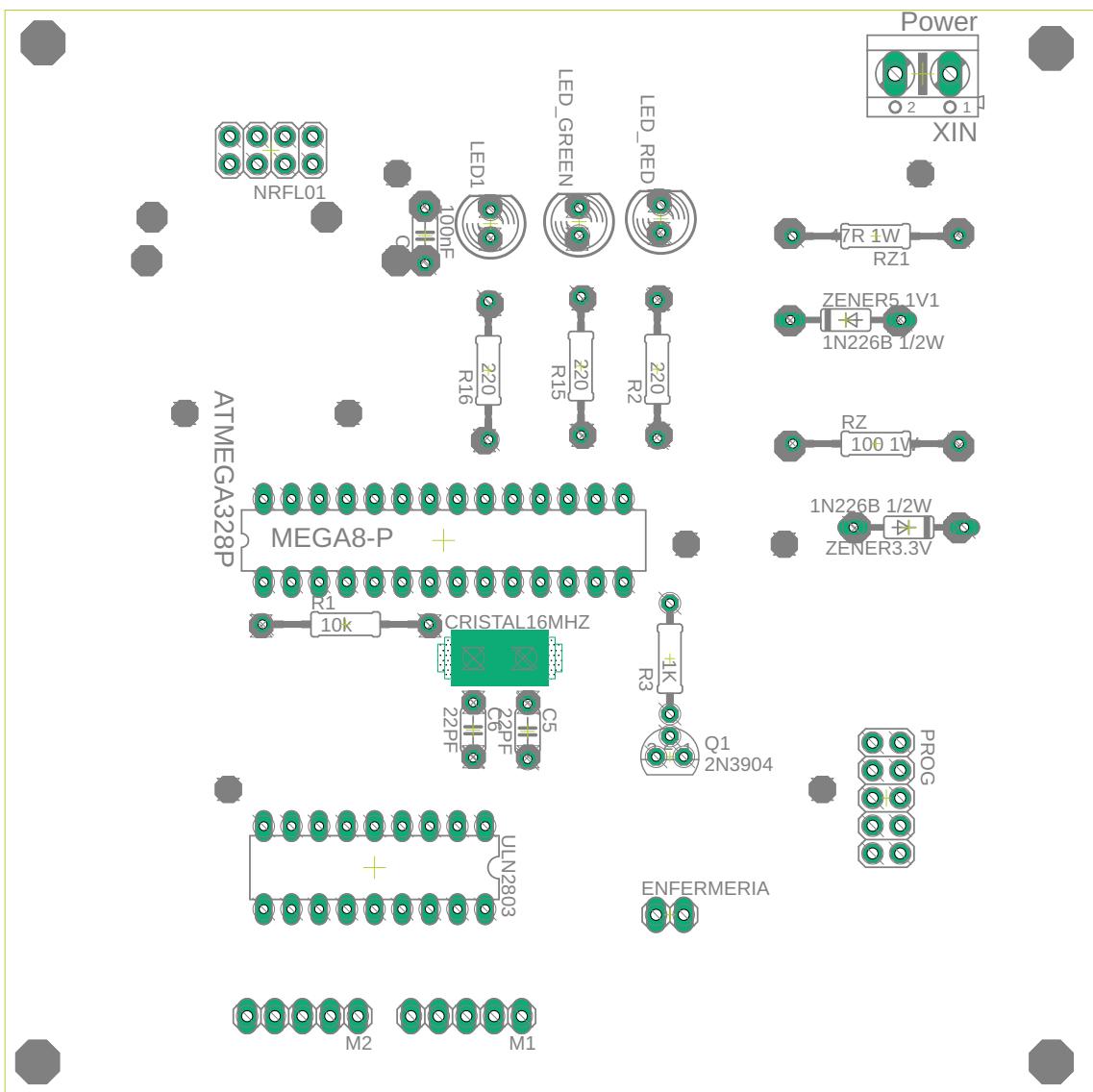
# • Circuito impreso: Módulo de mandos motrices

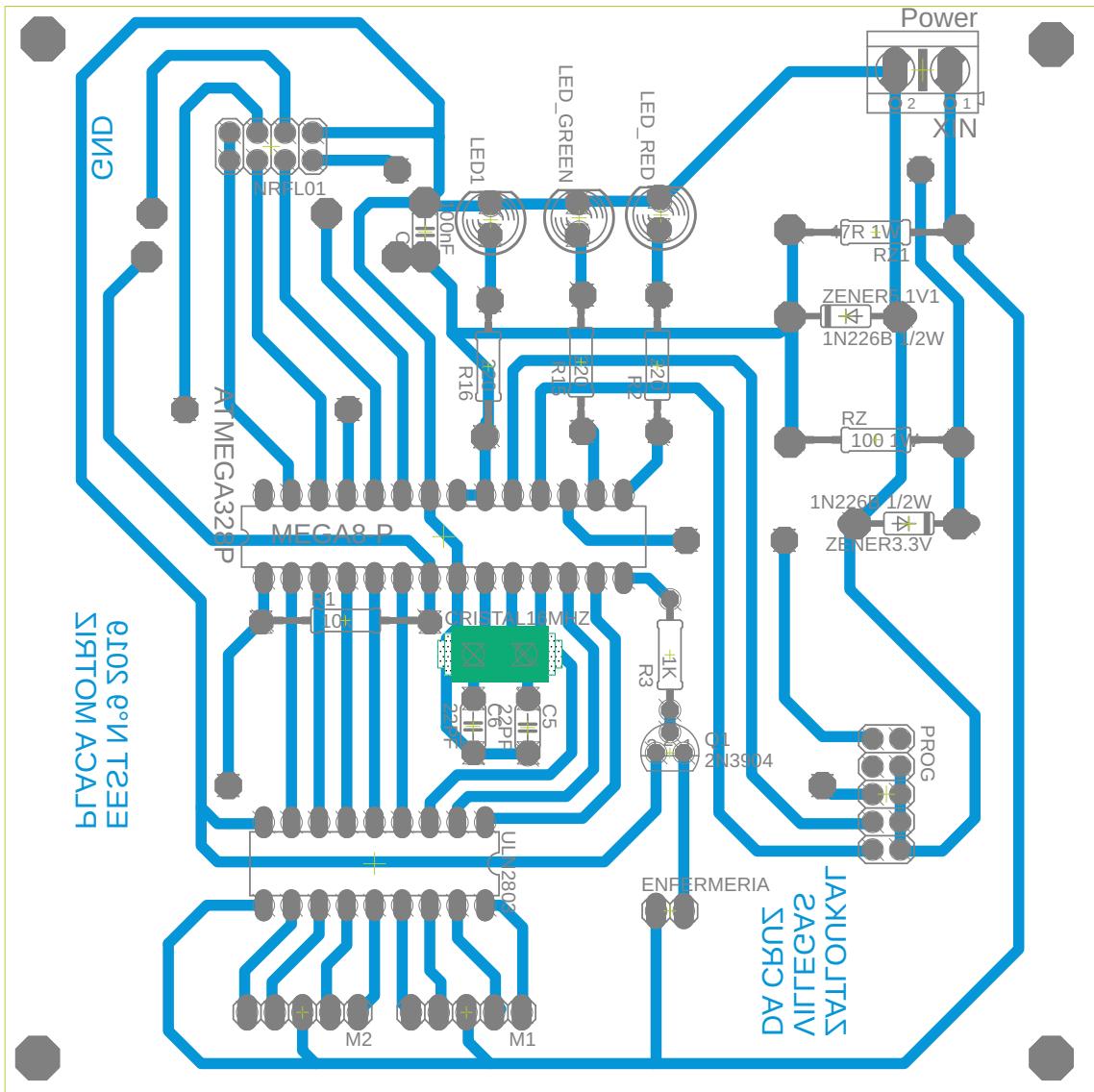
Lista de componentes	Cantidad	Precio x unidad
Capacitores ceramicos	3	\$ 2.50
Resistencias de 1/4 w	7	\$ 1.50
Leds de 5mm	3	\$ 4.50
Diodos zener de 1/2 w	2	\$ 6
Cristal de 16Mhz	1	\$ 30
Microcontrolador Atmega 328p	1	\$ 250
Modulo de radiofrecuencia emisor/receptor	1	\$ 100
Zocalo 14x2	1	\$ 60
ULN2803	1	\$ 60
Zocalo 18x2	1	\$ 60
Tira de 32 pinhead	1	\$ 15
Borneras	1	\$ 15
Transistor	1	\$ 40
Plaqueta virgen de 10x10	1	\$ 90.00
Total	\$ 763.50	

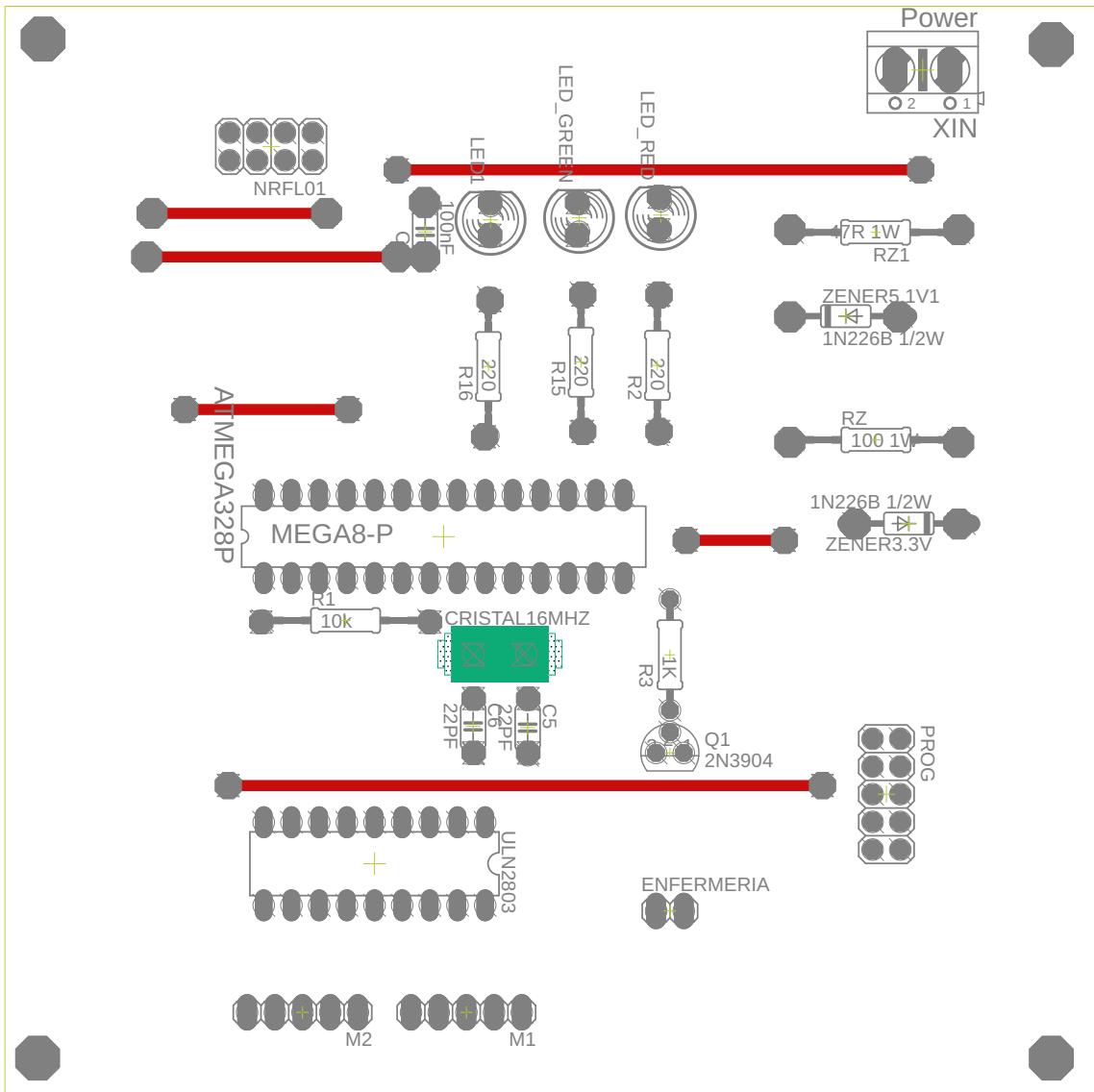
# PLACA MOTRIZ













## 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

```
1 #define F_CPU 16000000UL
2
3 #include <avr/io.h>
4 #include <util/delay.h>
5 #include <avr/interrupt.h>
6 #include <stdlib.h>
7 #include <string.h>
8 #include <stdbool.h>
9 #include <stdint.h>
10
11 #include "UART_Bluetooth.h"
12 #include "nrf24.h"
13
14 void initIO();
15 char messageTest[] = "UART TESTING COMMANDS! \n";
16
17 int main(void)
18 {
19     cli(); // Interrupts off
20     initIO();
21     initBluetoothUart();
22     setupReceiveMode();
23     nrf24_initRF_SAFE(POWER_BOARD_RF, RECEIVE); // CONNECTION TO POWER BOARD AND ↵
24     MOTORIZED BOARD : GENERAL RF CHANNEL 11
25     sei(); // Interrupts on
26     while (1)
27     {
28         if (commandAvailable) {
29             cli();
30             processReceivedLine();
31             setupReceiveMode();
32         }
33         // Disable UART
34
35         if(nrf24_dataReady())
36         {
37             cli();
38             nrf24_getData(command_buffer);
39             CommandStatus status = DecomposeMessageFromBuffer();
40             if (status==SUCCESSFUL_DECOMPOSITION) { RetransmissionToPhone(); }
41             sei();
42         }
43
44         if (nrf24_checkAvailability()==false) { nrf24_initRF_SAFE(POWER_BOARD_RF, ↵
45             RECEIVE); }
46
47     }
48 }
```

```
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

```
1
2 #include "Command_Handler.h"
3 #include "UART_Bluetooth.h"
4 #include "nrf24.h"
5 #include "crc.h"
6
7
8
9 const CommandType commandList[] = {
10     { .handlerFunction = &UPDATE_ALL_DEVICES_VALUE_H},
11     { .handlerFunction = &UPDATE_DEVICE_VALUE_H},
12     { .handlerFunction = &GET_ALL_DEVICES_VALUE_H},
13     { .handlerFunction = &GET_DEVICE_VALUE_H},
14     { .handlerFunction = &MESSAGE_STATUS_H}
15 };
16 #define commandListLength (uint8_t)(sizeof commandList/sizeof commandList[0])
17
18 bool initliazeMemory(){
19     if(memoryInitialized) return false;
20     parameter[0].startingPointer = (void*)calloc(23,1);
21     parameter[1].startingPointer = (void*)calloc(2,1);
22     parameter[2].startingPointer = (void*)calloc(2,1);
23     for (uint8_t x = 3; x<12; x++) parameter[x].startingPointer = (void*)calloc(1,1);
24     command_buffer = (uint8_t*)calloc(32,1);
25     if(command_buffer==NULL) return false;
26     for (uint8_t x = 0; x<12; x++) { if(parameter[x].startingPointer==NULL)
27         return false; }
28     memoryInitialized = true;
29     return true;
30 }
31 CommandStatus DecomposeMessageFromBuffer(){
32     // Search for header
33     uint8_t* headerStart = command_buffer;
34     uint8_t* footerEnd = command_buffer+31;
35
36     for(;headerStart!=(command_buffer+22);headerStart++){
37         if (*headerStart==SOH&&(*(headerStart+4)==STX)){
38             for(;footerEnd!=(command_buffer+6);footerEnd--){
39                 if (*footerEnd==ETB&&(*footerEnd-2)==ETX)){
40                     uint8_t netMessageLength = ((footerEnd-2)-headerStart);
41                     crc_t crc;
42                     crc = crc_init();
43                     crc = crc_update(crc, headerStart, netMessageLength);
44                     crc = crc_finalize(crc);
45                     if (*(footerEnd-1)!=crc) return WRONG_CHECKSUM_CONSISTENCY;
46                     if (*(headerStart+2)!=currentModuleID&&*(headerStart+2) !=0xFF&&currentModuleID!=0x01) return WRONG_MODULE_ID;
47                     lastTargetModuleID = *(headerStart+2);
48                     lastTransmitterModuleID = *(headerStart+3);
49                     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,      ↴
59              *parameterStart);
60        parameterStart+=((*parameterStart)+1);
61        if (parameterStart>=(footerEnd-2)) break;
62    }
63
64    return SUCCESFUL_DECOMPOSITION;
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_OVERSIZED;
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);  
42
```

```
98     *parameterStart = ETX;
99     *(parameterStart+1) = crc;
100    *(parameterStart+2) = ETB;
101
102    return SUCCESFUL_COMPOSITION;
103 }
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_TRANSMISSION_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: ↵
153     GENERAL RF CHANNEL 112
154     nrf24_send(command_buffer);
155     while(nrf24_isSending());
156
157     uint8_t messageStatusSecond = nrf24_lastMessageStatus();
158     if(messageStatusSecond == NRF24_TRANSMISSION_OK) { boardState[1] = 0xFF; }
159     else if(messageStatusSecond == NRF24_MESSAGE_LOST) { boardState[1]= 0x00; }
160
161     writeParameterValue(0, &boardState[0], 1);
162     writeParameterValue(1, &boardState[1], 1);
163     ComposeMessageToBuffer(UPDATE_ALL_DEVICES_VALUE_ID, 2, PHONE_MODULE); // ↵
164     PHONE_MODULE should be lastTransmitterModuleID
165     transmitMessageSync(command_buffer, 32);
166 }
167
168 void GET_DEVICE_VALUE_H() {
169     _delay_ms(100);
170     uint8_t deviceIndex = *((uint8_t*)parameter[0].startingPointer);
171     uint8_t deviceValue;
172
173     switch(deviceIndex){
174     case 0:
175         ComposeMessageToBuffer(MESSAGE_STATUS_ID, 0, POWER_MODULE);
176         nrf24_initRF_SAFE(POWER_BOARD_RF, TRANSMIT);      // CONNECTION TO ↵
177         MODULE: GENERAL RF CHANNEL 112
178         nrf24_send(command_buffer);
179         while(nrf24_isSending());
180
181         uint8_t messageStatus = nrf24_lastMessageStatus();
182         if(messageStatus == NRF24_TRANSMISSION_OK) { deviceValue = 0xFF; }
183         else if(messageStatus == NRF24_MESSAGE_LOST) { deviceValue= 0x00; }
184         break;
185     case 1:
186         ComposeMessageToBuffer(MESSAGE_STATUS_ID, 0, MOTOR_MODULE);
187         nrf24_initRF_SAFE(MOTORIZED_BOARD_RF, TRANSMIT);      // CONNECTION TO ↵
188         GENERAL RF CHANNEL 112
189         nrf24_send(command_buffer);
190         while(nrf24_isSending());
191
192         uint8_t messageStatusSecond 44 nrf24_lastMessageStatus();
193         if(messageStatusSecond == NRF24_TRANSMISSION_OK) { deviceValue = ↵
```

```
191         0xFF; }
192     else if(messageStatusSecond == NRF24_MESSAGE_LOST) { deviceValue=
193         0x00; }
194     break;
195 }
196 writeParameterValue(0, &deviceIndex, 1);
197 writeParameterValue(1, &deviceValue, 2);
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_SUCCESSFUL_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(),
102     GET_ALL_DEVICES_VALUE_H(), GET_DEVICE_VALUE_H(), MESSAGE_STATUS_H();  
47
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: ↵
81     // GENERAL RF CHANNEL 112 (lastTargetModuleID-1) offset 1
82     uint8_t targetModuleID = lastTargetModuleID;
83     uint8_t RF_TIME_OUT;
84     while(RF_TIME_OUT!=0xFF)
85     {
86         if(nrf24_dataReady()){
87             nrf24_getData(command_buffer);
88             CommandStatus status = DecomposeMessageFromBuffer();
89             if
90                 (status==SUCCEFUL_DECOMPOSITION&&lastTargetModuleID==targetModuleI ↵
91 D) {
92                 transmitMessageSync(command_buffer, 32);
93                 return true;
94             }
95         }
96     }
97
98 void processReceivedLine(){
99     commandAvailable = false;
100
101    CommandStatus status = DecomposeMessageFromBuffer();
```

```
102     if(status==SUCCEFUL_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 didModuleReply = 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_SUCCEFUL_TRANSMISSION:
124                     writeParameterValue(0, &(uint8_t){SUCCEFUL_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<<RCIE0);
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()
```

```
154         ==SUCCEFUL_DECOMPOSITION)) {
155             disableUART(); commandAvailable = true;
156         }
157         else if(*uartBufferPos==uartCarriageReturnChar) {
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 =
163                     command_buffer+(uartBufferSize-1);
164                 if (*(uartBufferPos-x)!=uartCarriageReturnChar)
165                     { hasToReturnCarriage = false; break; }
166             }
167             if (hasToReturnCarriage) {
168                 uartBufferPos = command_buffer;
169             }
170             else {
171                 uartBufferPos = savedUartBufferPos;
172             }
173         }
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();
```

53  
54 #endif /\* UART\_BLUETOOTH\_H \*/

```
1  /**
2   * \file
3   * Functions and types for CRC checks.
4   *
5   * Generated on Wed Sep 11 13:55:53 2019
6   * by pycrc v0.9.2, https://pycrc.org
7   * using the configuration:
8   * - Width          = 8
9   * - Poly           = 0x07
10  * - XorIn          = 0x00
11  * - ReflectIn     = False
12  * - XorOut         = 0x00
13  * - ReflectOut    = False
14  * - Algorithm      = bit-by-bit-fast
15  */
16 #include "crc.h"      /* include the header file generated with pycrc */
17 #include <stdlib.h>
18 #include <stdint.h>
19 #include <stdbool.h>
20
21
22
23 crc_t crc_update(crc_t crc, const void *data, size_t data_len)
24 {
25     const unsigned char *d = (const unsigned char *)data;
26     unsigned int i;
27     bool bit;
28     unsigned char c;
29
30     while (data_len--) {
31         c = *d++;
32         for (i = 0x80; i > 0; i >>= 1) {
33             bit = crc & 0x80;
34             if (c & i) {
35                 bit = !bit;
36             }
37             crc <<= 1;
38             if (bit) {
39                 crc ^= 0x07;
40             }
41         }
42         crc &= 0xff;
43     }
44     return crc & 0xff;
45 }
```

```
1  /**
2   * \file
3   * Functions and types for CRC checks.
4   *
5   * Generated on Wed Sep 11 13:56:48 2019
6   * by pycrc v0.9.2, https://pycrc.org
7   * using the configuration:
8   * - Width          = 8
9   * - Poly           = 0x07
10  * - XorIn          = 0x00
11  * - ReflectIn     = False
12  * - XorOut         = 0x00
13  * - ReflectOut    = False
14  * - Algorithm      = bit-by-bit-fast
15  *
16  * This file defines the functions crc_init(), crc_update() and crc_finalize().
17  *
18  * The crc_init() function returns the initial \c crc value and must be called
19  * before the first call to crc_update().
20  * Similarly, the crc_finalize() function must be called after the last call
21  * to crc_update(), before the \c crc is being used.
22  * is being used.
23  *
24  * The crc_update() function can be called any number of times (including zero
25  * times) in between the crc_init() and crc_finalize() calls.
26  *
27  * This pseudo-code shows an example usage of the API:
28  * \code{.c}
29  * crc_t crc;
30  * unsigned char data[MAX_DATA_LEN];
31  * size_t data_len;
32  *
33  * crc = crc_init();
34  * while ((data_len = read_data(data, MAX_DATA_LEN)) > 0) {
35  *     crc = crc_update(crc, data, data_len);
36  * }
37  * crc = crc_finalize(crc);
38  * \endcode
39  */
40 #ifndef CRC_H
41 #define CRC_H
42
43 #include <stdlib.h>
44 #include <stdint.h>
45
46 #ifdef __cplusplus
47 extern "C" {
48 #endif
49
50
51 /**
52  * 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 #define UCPHA0 1
2
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);
```

```
52     // Auto Acknowledgment
53     nrf24_configRegister(EN_AA,(1<<ENAA_P0)|(1<<ENAA_P1)|(0<<ENAA_P2)|      ↵
54     (0<<ENAA_P3)|(0<<ENAA_P4)|(0<<ENAA_P5));
55
56     // Enable RX addresses
57     nrf24_configRegister(EN_RXADDR,(1<<ERX_P0)|(1<<ERX_P1)|(0<<ERX_P2)|      ↵
58     (0<<ERX_P3)|(0<<ERX_P4)|(0<<ERX_P5));
59
60     // Auto retransmit delay: 1000 us and Up to 15 retransmit trials
61     nrf24_configRegister(SETUP_RETR,(0x04<<ARD)|(0x0F<<ARC));
62
63     // Dynamic length configurations: No dynamic length
64     nrf24_configRegister(DYNPD,(0<<DPL_P0)|(0<<DPL_P1)|(0<<DPL_P2)|(0<<DPL_P3)|      ↵
65     (0<<DPL_P4)|(0<<DPL_P5));
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)      ↵
72         return false;
73     if (nrf24_checkRegister(CONFIG,nrf24_CONFIG,1)==false) return false;
74     if (nrf24_checkRegister(SETUP_RETR,(0x04<<ARD)|(0x0F<<ARC),1)==false) return      ↵
75         false;
76     if (nrf24_checkRegister(DYNPD,(0<<DPL_P0)|(0<<DPL_P1)|(0<<DPL_P2)|      ↵
77     (0<<DPL_P3)|(0<<DPL_P4)|(0<<DPL_P5),1)==false) return false;
78
79     return true;
80 }
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 }
```

```
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_rx_fifoEmpty();;
144 }
145
146 /* Checks if receive FIFO is empty or not */
147 uint8_t nrf24_rx_fifoEmpty()          61
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 */          62
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_TRANSMISSION_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()
```

```
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 */ 65
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_transmitSync(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
485             { faultyRF_Alarm(); }
486     }
487
488     if (initMode==RECEIVE){
489         nrf24_tx_address(CURRENT_BOARD_ADDRESS);
490         nrf24_rx_address(BOARD_ADDRESS[boardIndex]);
491     }else{
492         nrf24_tx_address(BOARD_ADDRESS[boardIndex]);
493         nrf24_rx_address(CURRENT_BOARD_ADDRESS);
494     }
495
496
497     nrf24_powerUpRx();
498 }
499 }
```

```
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_TRANSMISSION_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_RF = 0,
47     POWER_BOARD_RF = 1,
48     MOTORIZED_BOARD_RF = 2
49 };
50 typedef enum CommandsBoard CommandsBoard;
51
52 extern void nrf24_initRF_SAFE(uint8_t boardIndex,TransmissionMode initMode);69
```

```
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();
```

105 #endif

106



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

### Contenidos:

main.c  
Command\_Handler.c  
Command\_Handler.h  
nrf24.c  
nrf24.h  
nRF24L01\_Definitions.h  
crc.c  
crc.h

```

1 #ifndef F_CPU
2 #define F_CPU 16000000UL
3 #endif
4 #include <avr/io.h>
5 #include <util/delay.h>
6 #include <avr/interrupt.h>
7 #include <stdlib.h>
8 #include <string.h>
9 #include <stdbool.h>
10 #include <stdint.h>
11
12 #include "nrf24.h"
13
14 void initIO();
15
16 int main(void)
17 {
18     //sei();      // Interrupts on
19     initIO();
20     nrf24_initRF_SAFE(MAIN_BOARD, RECEIVE); // CONNECTION TO MAIN BOARD : GENERAL ↵
21     RF CHANNEL 112
22
23     while (1)
24     {
25         if(nrf24_dataReady())
26         {
27             nrf24_getData(command_buffer);
28             CommandStatus status = DecomposeMessageFromBuffer();
29             if (status==SUCCEFUL_DECOMPOSITION) { HandleAvailableCommand(); }
30             else
31             {
32                 bit_flip(PORTD, BIT(7)); _delay_ms(250); bit_flip(PORTD, BIT(7));
33             }
34         }
35     }
36 }
37
38
39 void initIO(){
40     /*
41         Input/Output pin initialization
42         1 : OUTPUT | 0 : INPUT | 0b76543210 Bit order
43         ATTACHMENTS
44             RELAY 0    : PD3          |  OUTPUT
45             RELAY 1    : PD2          |  OUTPUT
46             RELAY 2    : PD6          |  OUTPUT
47             RELAY 3    : PD5          |  OUTPUT
48             RED LED   : PD7          |  OUTPUT
49             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
```

```
1
2 #include "Command_Handler.h"
3 #include "nrf24.h"
4 #include "crc.h"
5
6
7
8 const CommandType commandList[] = {
9     { .handlerFunction = &UPDATE_ALL_DEVICES_VALUE_H},
10    { .handlerFunction = &UPDATE_DEVICE_VALUE_H},
11    { .handlerFunction = &GET_ALL_DEVICES_VALUE_H},
12    { .handlerFunction = &GET_DEVICE_VALUE_H},
13    { .handlerFunction = &MESSAGE_STATUS_H}
14 };
15 #define commandListLength (uint8_t)(sizeof commandList/sizeof commandList[0])
16
17 bool initliazeMemory(){
18     if(memoryInitialized) return false;
19     parameter[0].startingPointer = (void*)calloc(23,1);
20     parameter[1].startingPointer = (void*)calloc(2,1);
21     parameter[2].startingPointer = (void*)calloc(2,1);
22     for (uint8_t x = 3; x<12; x++) parameter[x].startingPointer = (void*)calloc    ↴
23         (1,1);
24     command_buffer = (uint8_t*)calloc(32,1);
25     if(command_buffer==NULL) return false;
26     for (uint8_t x = 0; x<12; x++) { if(parameter[x].startingPointer==NULL)      ↴
27         return false; }
28     memoryInitialized = true;
29     return true;
30 }
31
32 CommandStatus DecomposeMessageFromBuffer(){
33     // Search for header
34     uint8_t* headerStart = command_buffer;
35     uint8_t* footerEnd = command_buffer+31;
36
37     for(;headerStart!=(command_buffer+22);headerStart++){
38         if (*headerStart==SOH&&(*(headerStart+4)==STX)){
39             for(;footerEnd!=(command_buffer+6);footerEnd--){
40                 if (*footerEnd==ETB&&(*footerEnd-2)==ETX)){
41                     uint8_t netMessageLength = ((footerEnd-2)-headerStart);
42                     crc_t crc;
43                     crc = crc_init();
44                     crc = crc_update(crc, headerStart, netMessageLength);
45                     crc = crc_finalize(crc);
46                     if (*(footerEnd-1)!=crc) return WRONG_CHECKSUM_CONSISTENCY;
47                     if (*(headerStart+2)!=currentModuleID&&*(headerStart+2)!    ↴
48                         =0xFF&&currentModuleID!=0x01) return WRONG_MODULE_ID;
49                     lastTargetModuleID = *(headerStart+2);
50                     lastTransmitterModuleID = *(headerStart+3);
51                     if (*(headerStart+5)>commandListLength-1) return      ↴
52                         UNDEFINED_COMMAND_CODE;
53                 }
54             }
55         }
56     }
57 }
```

```

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,
58             *parameterStart);
59         parameterStart+=((*parameterStart)+1);
60         if (parameterStart>=(footerEnd-2)) break;
61     }
62
63     return SUCCESSFUL_DECOMPOSITION;
64 }
65 }
66 }
67 return WRONG_HEADER_SEGMENTATION;
68 }
69
70 CommandStatus ComposeMessageToBuffer(CommandTypeID targetTypeID, uint8_t
71 parameterCount, uint8_t targetBoardID){
72     memset(command_buffer, 0, 32);
73     command_buffer[0] = SOH;
74     if (lastMessagePID==0xFF) { lastMessagePID++; } else { lastMessagePID = 0; }
75     command_buffer[1] = lastMessagePID;
76     command_buffer[2] = targetBoardID;
77     command_buffer[3] = currentModuleID;
78     command_buffer[4] = STX;
79     command_buffer[5] = targetTypeID;
80
81     if (parameterCount>12) return PARAMETER_COUNT_OVERSIZE;
82
83     uint8_t* parameterStart = &command_buffer[6];
84
85     for (uint8_t x = 0; x < parameterCount; x++){
86         *parameterStart = parameter[x].byteLength;
87         memcpy(parameterStart+1, parameter[x].startingPointer, parameter
88             [x].byteLength);
89         parameterStart+=(parameter[x].byteLength)+1;
90     }
91
92     crc_t crc;
93     crc = crc_init();
94     uint8_t crc_length = ((parameterStart)-(&command_buffer[0]));
95     crc = crc_update(crc, &command_buffer[0], crc_length);
96     crc = crc_finalize(crc);
97
98     *parameterStart = ETX;
99     *(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: ↵
109     GENERAL RF CHANNEL 112
110     nrf24_send(command_buffer);
111     while(nrf24_isSending());
112
113     uint8_t messageStatus = nrf24_lastMessageStatus();
114     if(messageStatus == NRF24_TRANSMISSION_OK) { return RF_SUCCESFUL_TRANSMISSION; } ↵
115     else if(messageStatus == NRF24_MESSAGE_LOST) { return RF_UNREACHEABLE_MODULE; } ↵
116     return RF_UNREACHEABLE_MODULE;
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,
179                           PHONE_MODULE); // PHONE_MODULE deberia ser lastTransmitterModuleID
180
181     nrf24_initRF_SAFE(MAIN_BOARD, TRANSMIT);
182     nrf24_send(command_buffer);
183     while(nrf24_isSending());
184     uint8_t messageStatus = nrf24_lastMessageStatus();
185 }
186 void GET_DEVICE_VALUE_H() {
187     _delay_ms(50);
188     uint8_t deviceIndex = *((uint8_t*)parameter[0].startingPointer);
189     writeParameterValue(0, &deviceIndex, 1);
190     writeParameterValue(1, &deviceStoredValue[deviceIndex], 2);
191     ComposeMessageToBuffer(UPDATE_DEVICE_VALUE_ID, 2, PHONE_MODULE); // PHONE_MODULE deberia ser lastTransmitterModuleID
192
193     nrf24_initRF_SAFE(MAIN_BOARD, TRANSMIT);
194     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_SUCCESSFUL_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(),
103     GET_ALL_DEVICES_VALUE_H(), GET_DEVICE_VALUE_H(), MESSAGE_STATUS_H();  
81
103 extern CommandStatus DecomposeMessageFromBuffer();
```

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

```
1 #define UCPHA0 1
2 #define BAUD_RATE 38400UL
3 #define UBRR_VALUE ((F_CPU)/(2UL*BAUD_RATE))-1
4
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],
16   &MOTORIZED_BOARD_ADDR[0];
17 uint8_t* CURRENT_BOARD_ADDRESS = &POWER_BOARD_ADDR[0];
18
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)|
49       (0<<ENAA_P3)|(0<<ENAA_P4)|(0<<ENAA_P5));          ↵
50     // Enable RX addresses
```

```

51     (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)|    ↵
56         (0<<DPL_P4)|(0<<DPL_P5));
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, ((0x03)<<RF_PWR),1)==false)    ↵
68             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)|    ↵
71             (0<<ENAA_P3)|(0<<ENAA_P4)|(0<<ENAA_P5),1)==false) return false;
72         if (nrf24_checkRegister(SETUP_RETR,(0x04<<ARD)|(0x0F<<ARC),1)==false) return ↵
73             false;
74         if (nrf24_checkRegister(DYNPD,(0<<DPL_P0)|(0<<DPL_P1)|(0<<DPL_P2)|    ↵
75             (0<<DPL_P3)|(0<<DPL_P4)|(0<<DPL_P5),1)==false) return false;
76
77         return true;
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 occurred 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);86
```

```
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();
```

```
252     /* Transmission went OK */
253     if((rv & ((1 << TX_DS))))
254     {
255         return NRF24_TRANSMISSION_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_transmitSync(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))
```

```
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
485             { faultyRF_Alarm(); }
486     }
487
488     if (initMode==TRANSMIT){
489         nrf24_tx_address(CURRENT_BOARD_ADDRESS);
490         nrf24_rx_address(BOARD_ADDRESS[boardIndex]);
491     }else{
492         nrf24_tx_address(BOARD_ADDRESS[boardIndex]);
493         nrf24_rx_address(CURRENT_BOARD_ADDRESS);
494     }
495     nrf24_powerUpRx();
496 }
```

```
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_TRANSMISSION_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); 93
```

```
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  /* Memory Map */
2  #define CONFIG      0x00
3  #define EN_AA       0x01
4  #define EN_RXADDR   0x02
5  #define SETUP_AW    0x03
6  #define SETUP_RETR  0x04
7  #define RF_CH       0x05
8  #define RF_SETUP    0x06
9  #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
```

```
1  /**
2   * \file
3   * Functions and types for CRC checks.
4   *
5   * Generated on Wed Sep 11 13:55:53 2019
6   * by pycrc v0.9.2, https://pycrc.org
7   * using the configuration:
8   * - Width          = 8
9   * - Poly           = 0x07
10  * - XorIn          = 0x00
11  * - ReflectIn     = False
12  * - XorOut         = 0x00
13  * - ReflectOut    = False
14  * - Algorithm      = bit-by-bit-fast
15  */
16 #include "crc.h"      /* include the header file generated with pycrc */
17 #include <stdlib.h>
18 #include <stdint.h>
19 #include <stdbool.h>
20
21
22
23 crc_t crc_update(crc_t crc, const void *data, size_t data_len)
24 {
25     const unsigned char *d = (const unsigned char *)data;
26     unsigned int i;
27     bool bit;
28     unsigned char c;
29
30     while (data_len--) {
31         c = *d++;
32         for (i = 0x80; i > 0; i >>= 1) {
33             bit = crc & 0x80;
34             if (c & i) {
35                 bit = !bit;
36             }
37             crc <<= 1;
38             if (bit) {
39                 crc ^= 0x07;
40             }
41         }
42         crc &= 0xff;
43     }
44     return crc & 0xff;
45 }
```

```
1  /**
2   * \file
3   * Functions and types for CRC checks.
4   *
5   * Generated on Wed Sep 11 13:56:48 2019
6   * by pycrc v0.9.2, https://pycrc.org
7   * using the configuration:
8   * - Width          = 8
9   * - Poly           = 0x07
10  * - XorIn          = 0x00
11  * - ReflectIn     = False
12  * - XorOut         = 0x00
13  * - ReflectOut    = False
14  * - Algorithm      = bit-by-bit-fast
15  *
16  * This file defines the functions crc_init(), crc_update() and crc_finalize().
17  *
18  * The crc_init() function returns the initial \c crc value and must be called
19  * before the first call to crc_update().
20  * Similarly, the crc_finalize() function must be called after the last call
21  * to crc_update(), before the \c crc is being used.
22  * is being used.
23  *
24  * The crc_update() function can be called any number of times (including zero
25  * times) in between the crc_init() and crc_finalize() calls.
26  *
27  * This pseudo-code shows an example usage of the API:
28  * \code{.c}
29  * crc_t crc;
30  * unsigned char data[MAX_DATA_LEN];
31  * size_t data_len;
32  *
33  * crc = crc_init();
34  * while ((data_len = read_data(data, MAX_DATA_LEN) > 0) {
35  *     crc = crc_update(crc, data, data_len);
36  * }
37  * crc = crc_finalize(crc);
38  * \endcode
39  */
40 #ifndef CRC_H
41 #define CRC_H
42
43 #include <stdlib.h>
44 #include <stdint.h>
45
46 #ifdef __cplusplus
47 extern "C" {
48 #endif
49
50
51 /**
52  * The definition of the used algorithm. 99
```

```
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
```



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

### Contenidos:

main.c  
Command\_Handler.c  
Command\_Handler.h  
nrf24.c  
nrf24.h  
nRF24L01\_Definitions.h  
crc.c  
crc.h

```
1 #ifndef F_CPU
2 #define F_CPU 16000000UL
3 #endif
4 #include <avr/io.h>
5 #include <util/delay.h>
6 #include <avr/interrupt.h>
7 #include <stdlib.h>
8 #include <string.h>
9 #include <stdbool.h>
10 #include <stdint.h>
11
12 #include "nrf24.h"
13
14 void initIO();
15
16 int main(void)
17 {
18     initIO();
19     nrf24_initRF_SAFE(MAIN_BOARD, RECEIVE); // CONNECTION TO MAIN BOARD : GENERAL ↵
12          RF CHANNEL 112
20
21     while (1)
22     {
23         if(nrf24_dataReady())
24         {
25
26             nrf24_getData(command_buffer);
27             CommandStatus status = DecomposeMessageFromBuffer();
28             if (status==SUCCESSFUL_DECOMPOSITION) { HandleAvailableCommand(); }
29         }
30
31         if (nrf24_checkAvailability()==false) { nrf24_initRF_SAFE(MAIN_BOARD, ↵
RECEIVE); }
32     }
33 }
34
35
36 void initIO(){
37     /*
38         Input/Output pin initialization
39         1 : OUTPUT | 0 : INPUT | 0b76543210 Bit order
40         ATTACHMENTS
41             NURSE SIGN    : PB0           |   OUTPUT
42             GREEN LED    : PB1           |   OUTPUT      (SWAPPED IN PCB)
43             RED LED      : PB2           |   OUTPUT
44             STEP MOTOR A (CURTAIN)
45                 TERMINAL NO.1 : PD0          |   OUTPUT
46                 TERMINAL NO.2 : PD1          |   OUTPUT
47                 TERMINAL NO.3 : PD2          |   OUTPUT
48                 TERMINAL NO.4 : PD3          |   OUTPUT
49             STEP MOTOR B (STRETCHER)      103 |   OUTPUT
50                 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
```

```
1
2 #include "Command_Handler.h"
3 #include "nrf24.h"
4 #include "crc.h"
5
6
7
8 const CommandType commandList[] = {
9     { .handlerFunction = &UPDATE_ALL_DEVICES_VALUE_H},
10    { .handlerFunction = &UPDATE_DEVICE_VALUE_H},
11    { .handlerFunction = &GET_ALL_DEVICES_VALUE_H},
12    { .handlerFunction = &GET_DEVICE_VALUE_H},
13    { .handlerFunction = &MESSAGE_STATUS_H}
14 };
15 #define commandListLength (uint8_t)(sizeof commandList/sizeof commandList[0])
16
17 bool initliazeMemory(){
18     if(memoryInitialized) return false;
19     parameter[0].startingPointer = (void*)calloc(23,1);
20     parameter[1].startingPointer = (void*)calloc(2,1);
21     parameter[2].startingPointer = (void*)calloc(2,1);
22     for (uint8_t x = 3; x<12; x++) parameter[x].startingPointer = (void*)calloc    ↵
23         (1,1);
24     command_buffer = (uint8_t*)calloc(32,1);
25     if(command_buffer==NULL) return false;
26     for (uint8_t x = 0; x<12; x++) { if(parameter[x].startingPointer==NULL)      ↵
27         return false; }
28     memoryInitialized = true;
29     return true;
30 }
31
32 CommandStatus DecomposeMessageFromBuffer(){
33     // Search for header
34     uint8_t* headerStart = command_buffer;
35     uint8_t* footerEnd = command_buffer+31;
36
37     for(;headerStart!=(command_buffer+22);headerStart++){
38         if (*headerStart==SOH&&(*(headerStart+4)==STX)){
39             for(;footerEnd!=(command_buffer+6);footerEnd--){
40                 if (*footerEnd==ETB&&(*footerEnd-2)==ETX)){
41                     uint8_t netMessageLength = ((footerEnd-2)-headerStart);
42                     crc_t crc;
43                     crc = crc_init();
44                     crc = crc_update(crc, headerStart, netMessageLength);
45                     crc = crc_finalize(crc);
46                     if (*(footerEnd-1)!=crc) return WRONG_CHECKSUM_CONSISTENCY;
47                     if (*(headerStart+2)!=currentModuleID&&*(headerStart+2)!    ↵
48                         =0xFF&&currentModuleID!=0x01) return WRONG_MODULE_ID;
49                     lastTargetModuleID = *(headerStart+2);
50                     lastTransmitterModuleID = *(headerStart+3);
51                     if (*(headerStart+5)>commandListLength-1) return      ↵
52                         UNDEFINED_COMMAND_CODE;
53                 }
54             }
55         }
56     }
57 }
```

```
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,     ↴
58                         *parameterStart);
59                 parameterStart+=((*parameterStart)+1);
60                 if (parameterStart>=(footerEnd-2)) break;
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      ↴
91               [x].byteLength);
92         parameterStart+=(parameter[x].byteLength)+1;
93     }
94
95     crc_t crc;
96     crc = crc_init();
97     uint8_t crc_length = ((parameterStart)-(&command_buffer[0]));  
106
98     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 400
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){108
```

```
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 ↴
91                                     guardan en grados
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(),
98             GET_ALL_DEVICES_VALUE_H(), GET_DEVICE_VALUE_H(), MESSAGE_STATUS_H();    ↴
99 extern CommandStatus ComposeMessageToBuffer(CommandTypeID targetTypeID, uint8_t    ↴
100          parameterCount, uint8_t targetBoardID);
101 extern CommandStatus DecomposeMessageFromBuffer();
102 extern void writeParameterValue(uint8_t parameterIndex, void* parameterData,      ↴
103          uint8_t parameterByteLength);
```

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

```
1 #define UCPHA0 1
2 #define BAUD_RATE 9600UL
3 #define UBRR_VALUE ((F_CPU)/(2UL*BAUD_RATE))-1
4
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],
17     &MOTORIZED_BOARD_ADDR[0]};
18 uint8_t* CURRENT_BOARD_ADDRESS = &MOTORIZED_BOARD_ADDR[0];
19
20 uint8_t GENERAL_RF_CHANNEL = 112;
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)|
50         (0<<ENAA_P3)|(0<<ENAA_P4)|(0<<ENAA_P5));
51     // Enable RX addresses
```

---

...Proyecto de placa motriz\Proyecto de placa motriz\nrf24.c

---

```

51     nrf24_configRegister(EN_RXADDR,(1<<ERX_P0)|(1<<ERX_P1)|(0<<ERX_P2)|    ↵
52         (0<<ERX_P3)|(0<<ERX_P4)|(0<<ERX_P5));
53     // Auto retransmit delay: 1000 us and Up to 15 retransmit trials
54     nrf24_configRegister(SETUP_RETR,(0x04<<ARD)|(0x0F<<ARC));
55     // Dynamic length configurations: No dynamic length
56     nrf24_configRegister(DYNPD,(0<<DPL_P0)|(0<<DPL_P1)|(0<<DPL_P2)|(0<<DPL_P3)|    ↵
57         (0<<DPL_P4)|(0<<DPL_P5));
58
59 }                                     ↵
60
61 bool nrf24_checkConfig(){           ↵
62     // Check all registers           ↵
63     if (nrf24_checkRegister(RF_CH, selectedChannel,1)==false) return false;
64     if (nrf24_checkRegister(RX_PW_P0, 0x00,1)==false) return false;
65     if (nrf24_checkRegister(RX_PW_P1, payload_len,1)==false) return false;
66     if (nrf24_checkRegister(RX_PW_P2, 0x00,1)==false) return false;
67     if (nrf24_checkRegister(RX_PW_P3, 0x00,1)==false) return false;
68     if (nrf24_checkRegister(RX_PW_P4, 0x00,1)==false) return false;
69     if (nrf24_checkRegister(RX_PW_P5, 0x00,1)==false) return false;
70     if (nrf24_checkRegister(RF_SETUP, (0<<RF_DR)|(0x03)<<RF_PWR),1)==false)    ↵
71         return false;
72     if (nrf24_checkRegister(CONFIG,nrf24_CONFIG,1)==false) return false;
73     if (nrf24_checkRegister(EN_AA,(1<<ENAA_P0)|(1<<ENAA_P1)|(0<<ENAA_P2)|    ↵
74         (0<<ENAA_P3)|(0<<ENAA_P4)|(0<<ENAA_P5),1)==false) return false;
75     if (nrf24_checkRegister(SETUP_RETR,(0x04<<ARD)|(0x0F<<ARC),1)==false) return ↵
76         false;
77     if (nrf24_checkRegister(DYNPD,(0<<DPL_P0)|(0<<DPL_P1)|(0<<DPL_P2)|    ↵
78         (0<<DPL_P3)|(0<<DPL_P4)|(0<<DPL_P5),1)==false) 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 }
```

---

```
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_rx_fifoEmpty();;
133 }
134
135 /* Checks if receive FIFO is empty or not */
136 uint8_t nrf24_rx_fifoEmpty()
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()          116
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         /* 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 exceeded (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;
```

```
252     rv = nrf24_getStatus();
253
254     /* Transmission went OK */
255     if((rv & ((1 << TX_DS))))
256     {
257         return NRF24_TRANSMISSION_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 }
```

```
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_transmitSync(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()      122
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
474             { faultyRF_Alarm(); }
475     }
476
477     if (initMode==TRANSMIT){
478         nrf24_tx_address(CURRENT_BOARD_ADDRESS);
479         nrf24_rx_address(BOARD_ADDRESS[boardIndex]);
480     }else{
481         nrf24_tx_address(BOARD_ADDRESS[boardIndex]);
482         nrf24_rx_address(CURRENT_BOARD_ADDRESS);
483     }
484     nrf24_powerUpRx();
485 }
```

```
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_TRANSISSTION_OK 0
33 #define NRF24_MESSAGE_LOST 1
34
35 #define AVAILABLE_COMMAND_BOARDS 2
36 #define CLEAR_FAULTY_RF_LED bit_clear(PORTB, BIT(1))
37 #define FLIP_FAULTY_RF_LED bit_flip(PORTB, BIT(1))
38
39
40 enum TransmissionMode {
41     RECEIVE,
42     TRANSMIT
43 };
44 typedef enum TransmissionMode TransmissionMode;
45
46 enum CommandsBoard {
47     MAIN_BOARD = 0,
48     POWER_BOARD = 1,
49     MOTORIZED_BOARD = 2
50 };
51 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();25
104
```

---

105 #endif

106

```
1  /* Memory Map */
2  #define CONFIG      0x00
3  #define EN_AA       0x01
4  #define EN_RXADDR   0x02
5  #define SETUP_AW    0x03
6  #define SETUP_RETR  0x04
7  #define RF_CH       0x05
8  #define RF_SETUP    0x06
9  #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
```

```
1  /**
2   * \file
3   * Functions and types for CRC checks.
4   *
5   * Generated on Wed Sep 11 13:55:53 2019
6   * by pycrc v0.9.2, https://pycrc.org
7   * using the configuration:
8   * - Width          = 8
9   * - Poly           = 0x07
10  * - XorIn          = 0x00
11  * - ReflectIn     = False
12  * - XorOut         = 0x00
13  * - ReflectOut    = False
14  * - Algorithm      = bit-by-bit-fast
15  */
16 #include "crc.h"      /* include the header file generated with pycrc */
17 #include <stdlib.h>
18 #include <stdint.h>
19 #include <stdbool.h>
20
21
22
23 crc_t crc_update(crc_t crc, const void *data, size_t data_len)
24 {
25     const unsigned char *d = (const unsigned char *)data;
26     unsigned int i;
27     bool bit;
28     unsigned char c;
29
30     while (data_len--) {
31         c = *d++;
32         for (i = 0x80; i > 0; i >>= 1) {
33             bit = crc & 0x80;
34             if (c & i) {
35                 bit = !bit;
36             }
37             crc <<= 1;
38             if (bit) {
39                 crc ^= 0x07;
40             }
41         }
42         crc &= 0xff;
43     }
44     return crc & 0xff;
45 }
```

```
1  /**
2   * \file
3   * Functions and types for CRC checks.
4   *
5   * Generated on Wed Sep 11 13:56:48 2019
6   * by pycrc v0.9.2, https://pycrc.org
7   * using the configuration:
8   * - Width          = 8
9   * - Poly           = 0x07
10  * - XorIn          = 0x00
11  * - ReflectIn     = False
12  * - XorOut         = 0x00
13  * - ReflectOut    = False
14  * - Algorithm      = bit-by-bit-fast
15  *
16  * This file defines the functions crc_init(), crc_update() and crc_finalize().
17  *
18  * The crc_init() function returns the initial \c crc value and must be called
19  * before the first call to crc_update().
20  * Similarly, the crc_finalize() function must be called after the last call
21  * to crc_update(), before the \c crc is being used.
22  * is being used.
23  *
24  * The crc_update() function can be called any number of times (including zero
25  * times) in between the crc_init() and crc_finalize() calls.
26  *
27  * This pseudo-code shows an example usage of the API:
28  * \code{.c}
29  * crc_t crc;
30  * unsigned char data[MAX_DATA_LEN];
31  * size_t data_len;
32  *
33  * crc = crc_init();
34  * while ((data_len = read_data(data, MAX_DATA_LEN)) > 0) {
35  *     crc = crc_update(crc, data, data_len);
36  * }
37  * crc = crc_finalize(crc);
38  * \endcode
39  */
40 #ifndef CRC_H
41 #define CRC_H
42
43 #include <stdlib.h>
44 #include <stdint.h>
45
46 #ifdef __cplusplus
47 extern "C" {
48 #endif
49
50
51 /**
52  * 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
```



# Código de fuente: Aplicación SAPLM

## Contenidos:

SAPLM.bluetoothActivities

    AvailableDevice

    BluetoothActivity

    BluetoothConnection

    CRC8\_CCITT

    DeviceListAdapter

    UnifiedTransmissionProtocol

SAPLM.commandActivity

    CommandItem

    CommandsActivity

    CommandsListAdapter

SAPLM.settingsActivities

    AppearanceMiscellaneousActivity

    AvailableSetting

    BluetoothSettingsActivity

    CommandsSettingsActivity

    ContactsCommunicationActivity

    DeveloperConsoleActivity

    InformationHelpActivity

    InternetConnectivityActivity

    MyDividerItemDecoration

    RecyclerTouchListener

    settings\_layout

    SettingsLayoutAdapter

    TerminalManager

    VoiceRecognitionSettingsActivity

SAPLM.Application

SAPLM.ContactsActivity

SAPLM.GoogleSpeechRecognizer

SAPLM.MainActivity

SAPLM.MainMenuHandler

SAPLM.MicrophoneHandler

SAPLM.PocketsphinxListener

SAPLM.TextToSpeechHandler

## Archivo - Application.java

```
1 package com.example.SAPLM;
2
3 import android.app.Activity;
4 import androidx.lifecycle.LifecycleObserver;
5 import android.bluetooth.BluetoothAdapter;
6 import android.bluetooth.BluetoothDevice;
7 import android.content.Context;
8 import android.content.IntentFilter;
9 import android.os.Bundle;
10 import android.widget.Toast;
11
12 import com.example.SAPLM.bluetoothActivities.BluetoothConnection;
13 import com.example.SAPLM.settingsActivities.DeveloperConsoleActivity;
14
15 import java.io.IOException;
16
17 public class Application extends android.app.Application implements LifecycleObserver {
18
19     @Override
20     public void onCreate() {
21         super.onCreate();
22
23         // ProcessLifecycleOwner.get().getLifecycle().addObserver(this);
24
25         IntentFilter bluetoothAdapterActionChangedFilter = new IntentFilter(BluetoothAdapter
26 .ACTION_STATE_CHANGED);
26         registerReceiver(BluetoothConnection.bluetoothAdapterStatusBR,
27         bluetoothAdapterActionChangedFilter);
28
29         IntentFilter bluetoothScanModeStatusChangedFilter = new IntentFilter();
30         bluetoothScanModeStatusChangedFilter.addAction(BluetoothAdapter.
31 ACTION_DISCOVERY_STARTED);
30         bluetoothScanModeStatusChangedFilter.addAction(BluetoothAdapter.
32 ACTION_DISCOVERY_FINISHED);
31         bluetoothScanModeStatusChangedFilter.addAction(BluetoothAdapter.
32 ACTION_SCAN_MODE_CHANGED);
32         registerReceiver(BluetoothConnection.bluetoothScanModeStatusBR,
33         bluetoothScanModeStatusChangedFilter);
34
35         IntentFilter bluetoothDeviceStatusChangedFilter = new IntentFilter();
35         bluetoothDeviceStatusChangedFilter.addAction(BluetoothDevice.ACTION_ACL_CONNECTED);
36         bluetoothDeviceStatusChangedFilter.addAction(BluetoothDevice.ACTION_ACL_DISCONNECTED
37 );
37         bluetoothDeviceStatusChangedFilter.addAction(BluetoothDevice.
38 ACTION_ACL_DISCONNECT_REQUESTED);
38         bluetoothDeviceStatusChangedFilter.addAction(BluetoothDevice.
39 ACTION_BOND_STATE_CHANGED);
39         registerReceiver(BluetoothConnection.bluetoothDeviceStatusBR,
40         bluetoothDeviceStatusChangedFilter);
41
41         Thread internetDetector = new Thread() {
42             @Override
43             public void run() {
44                 super.run();
45                 while(true) {
46                     try {
47                         internetConnectionStatus = isConnected();
48                         Thread.sleep(10000);
49                         if (internetConnectionStatus) {Application.toastMessage("Internet
50 connection detected");}
50                         else {Application.toastMessage("Proper internet connection has not
51 been detected");}
51                     } catch (Throwable th) { th.printStackTrace(); Application.toastMessage(
52 "An error occurred while checking the internet connection.");}
52                 }
53             }
54         };
55     }
56 }
```

Archivo - Application.java

```
51 th.toString());}
52 }
53 }
54 };
55
56 internetDetector.start();
57
58 Application.persistentDataSave.putString("MAIN_TEXT_KEY", new String());
59 Application.persistentDataSave.putString("ONGOING_TEXT_KEY", new String());
60
61 }
62
63 @Override
64 public void onTerminate() {
65     super.onTerminate();
66 }
67
68 private static Activity mActivity;
69 private static Context mContext;
70 public static Bundle persistentDataSave = new Bundle();
71
72
73 public static Activity getActivity() { return mActivity; }
74 public static void setActivity(Activity mActivitySet) { mActivity = mActivitySet; }
75
76 public static Context getContext() { return mContext; }
77 public static void setContext(Context mContextSet) { mContext = mContextSet; }
78
79 public static void toastMessage(final String message){
80     mActivity.runOnUiThread(new Runnable() {
81         @Override
82         public void run() {
83             if (mActivity != null) {
84                 //Toast.makeText(mActivity, message, Toast.LENGTH_SHORT).show();
85             }
86         }
87     });
88 }
89
90 private static boolean internetConnectionStatus;
91
92 public static boolean isSystemOnline(){
93     return internetConnectionStatus;
94 }
95
96 public boolean isConnected() throws InterruptedException, IOException {
97     final String command = "ping -c 1 google.com";
98     return Runtime.getRuntime().exec(command).waitFor() == 0;
99 }
100
101
102
103
104
105
106 }
107
```

## Archivo - MainActivity.java

Archivo - MainActivity.java

```
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
98 public void onClickListener_settings(View view){ mainMenuHandler.settingsButtonOnClick();
99 }
100
101 public void onClickListener_contacts(View view){ mainMenuHandler.contactsButtonOnClick();
102 }
103
104 public void onClickListener_bluetooth(View view){ mainMenuHandler.
105     bluetoothButtonOnClick(); }
106
107
108     PocketsphinxListener.shutdownRecognizer();
109     GoogleSpeechRecognizer.shutdownRecognizer();
110     BluetoothConnection.closeAllConnections();
111     TextToSpeechHandler.shutdownTextSpeech();
112
113     super.onDestroy();
```

## Archivo - MainMenuHandler.java

## Archivo - MainMenuHandler.java

```
61         animatorContacts.addUpdateListener(new ValueAnimator.AnimatorUpdateListener
62     () {
63             @Override
64             public void onAnimationUpdate(ValueAnimator animation) {
65                 float value = ((Float) (animation.getAnimatedValue()))
66                     .floatValue();
67
68                 float translationX = (float) (Math.cos(value * (Math.PI / 180.0f))
69                     * (radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
70                 float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.
71                     0d) - radialMenuRadius * translationX));
72
73                 if (value > 180.0f) {
74                     translationY = -translationY;
75                 }
76
77             imgBtn_contacts.setTranslationX(convertDpToPixel(translationX));
78             imgBtn_contacts.setTranslationY(convertDpToPixel(translationY));
79
80         }
81
82         imgBtn_contacts.setVisibility(View.VISIBLE);
83         animatorContacts.start();
84
85         ValueAnimator animatorSettings = ValueAnimator.ofFloat(360, 216);
86         animatorSettings.setDuration(animationDuration);
87         animatorSettings.setInterpolator(new AccelerateDecelerateInterpolator());
88         animatorSettings.addUpdateListener(new ValueAnimator.AnimatorUpdateListener
89     () {
90             @Override
91             public void onAnimationUpdate(ValueAnimator animation) {
92                 float value = ((Float) (animation.getAnimatedValue()))
93                     .floatValue();
94
95                 float translationX = (float) (Math.cos(value * (Math.PI / 180.0f))
96                     * (radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
97                 float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.
98                     0d) - radialMenuRadius * translationX));
99
100                if (value > 180.0f) {
101                    translationY = -translationY;
102                }
103
104                imgBtn_settings.setTranslationX(convertDpToPixel(translationX));
105                imgBtn_settings.setTranslationY(convertDpToPixel(translationY));
106
107            }
108
109            imgBtn_settings.setVisibility(View.VISIBLE);
110            animatorSettings.start();
111
112            ValueAnimator animatorBluetooth = ValueAnimator.ofFloat(360, 288);
113            animatorBluetooth.setDuration(animationDuration);
114            animatorBluetooth.setInterpolator(new AccelerateDecelerateInterpolator());
115            animatorBluetooth.addUpdateListener(new ValueAnimator.
116
117                AnimatorUpdateListener() {
118                    @Override
119                    public void onAnimationUpdate(ValueAnimator animation) {
120                        float value = ((Float) (animation.getAnimatedValue()))
121                            .floatValue();
122
123                        float translationX = (float) (Math.cos(value * (Math.PI / 180.0f))
124                            * (radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
125                        float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.
126                            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).
128             setInterpolator(new AccelerateDecelerateInterpolator());
129             imgBtn_contacts.animate().alpha(1.0f).setDuration(animationDuration).
130             setInterpolator(new AccelerateDecelerateInterpolator());
131             imgBtn_settings.animate().alpha(1.0f).setDuration(animationDuration).
132             setInterpolator(new AccelerateDecelerateInterpolator());
133             imgBtn_bluetooth.animate().alpha(1.0f).setDuration(animationDuration).
134             setInterpolator(new AccelerateDecelerateInterpolator()).withEndAction(new Runnable() {
135
136                 @Override
137                 public void run() {
138                     isAnimationRunning = false;
139                 }
140             });
141
142             menuUncoiled = true;
143
144         } else {
145
146             ValueAnimator animatorCommands = ValueAnimator.ofFloat(72, 360);
147             animatorCommands.setDuration(animationDuration);
148             animatorCommands.setInterpolator(new AccelerateDecelerateInterpolator());
149             animatorCommands.addUpdateListener(new ValueAnimator.AnimatorUpdateListener
() {
150
151                 @Override
152                 public void onAnimationUpdate(ValueAnimator animation) {
153                     float value = ((Float) (animation.getAnimatedValue()))
154                         .floatValue();
155
156                     float translationX = (float) (Math.cos(value * (Math.PI / 180.0f))
157 * (radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
158                     float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.
159 0d) - radialMenuRadius * translationX));
160
161                     if (value > 180.0f) {
162                         translationY = -translationY;
163                     }
164
165                     imgBtn_commands.setTranslationX(convertDpToPixel(translationX));
166                     imgBtn_commands.setTranslationY(convertDpToPixel(translationY));
167
168                 }
169             });
170             imgBtn_commands.setVisibility(View.VISIBLE);
171             animatorCommands.start(); 141
172
173             ValueAnimator animatorContacts = ValueAnimator.ofFloat(144, 360);
174             animatorContacts.setDuration(animationDuration);
175             animatorContacts.setInterpolator(new AccelerateDecelerateInterpolator());
176             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))
176 * (radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
177                 float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.
178 0d) - radialMenuRadius * translationX));
179                 if (value > 180.0f) {
180                     translationY = -translationY;
181                 }
182
183                 imgBtn_contacts.setTranslationX(convertDpToPixel(translationX));
184                 imgBtn_contacts.setTranslationY(convertDpToPixel(translationY));
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))
199 * (radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
200                     float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.
201 0d) - radialMenuRadius * translationX));
202                     if (value > 180.0f) {
203                         translationY = -translationY;
204                     }
205
206                     imgBtn_settings.setTranslationX(convertDpToPixel(translationX));
207                     imgBtn_settings.setTranslationY(convertDpToPixel(translationY));
208                 }
209             );
210             imgBtn_settings.setVisibility(View.VISIBLE);
211             animatorSettings.start();
212
213             ValueAnimator animatorBluetooth = ValueAnimator.ofFloat(288, 360);
214             animatorBluetooth.setDuration(animationDuration);
215             animatorBluetooth.setInterpolator(new AccelerateDecelerateInterpolator());
216             animatorBluetooth.addUpdateListener(new ValueAnimator.
AnimatorUpdateListener() {
217                 @Override
218                 public void onAnimationUpdate(ValueAnimator animation) {
219                     float value = ((Float) (animation.getAnimatedValue()))
220                         .floatValue();
221                     float translationX = (float) (Math.cos(value * (Math.PI / 180.0f))
222 * (radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
223                     float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.
224 0d) - radialMenuRadius * translationX));
225                     if (value > 180.0f) {
226                         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).
236     setInterpolator(new AccelerateDecelerateInterpolator());
237     imgBtn_contacts.animate().alpha(0.0f).setDuration(animationDuration).
238     setInterpolator(new AccelerateDecelerateInterpolator());
239     imgBtn_settings.animate().alpha(0.0f).setDuration(animationDuration).
240     setInterpolator(new AccelerateDecelerateInterpolator());
241     imgBtn_bluetooth.animate().alpha(0.0f).setDuration(animationDuration).
242     setInterpolator(new AccelerateDecelerateInterpolator()).withEndAction(new Runnable() {
243         @Override
244         public void run() {
245             isAnimationRunning = 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

```

Archivo - MainMenuHandler.java

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

Archivo - ContactsActivity.java

```
1 package com.example.SAPLM;
2
3 import android.graphics.PorterDuff;
4 import android.graphics.drawable.Drawable;
5 import android.os.Bundle;
6 import androidx.core.content.ContextCompat;
7 import androidx.appcompat.app.AppCompatActivity;
8 import androidx.appcompat.widget.Toolbar;
9 import android.view.MenuItem;
10
11 public class ContactsActivity extends AppCompatActivity {
12     private static Toolbar toolbar;
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.contacts_layout);
18         Application.setActivity(this);
19         Application.setContext(this);
20
21         toolbar = this.findViewById(R.id.toolbar);
22
23         setSupportActionBar(toolbar);
24
25         // add back arrow to toolbar
26         if (getSupportActionBar() != null){
27             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
28             getSupportActionBar().setDisplayShowHomeEnabled(true);
29
30             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
abc_ic_ab_back_material);
31             upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
32             getSupportActionBar().setHomeAsUpIndicator(upArrow);
33         }
34
35         TextToSpeechHandler.initializeTextToSpeech();
36         try {
37             Thread.sleep(250);
38         }catch (Throwable th) { Application.toastMessage(th.toString());}
39
40         TextToSpeechHandler.addToSpeakingQueue("holá");
41     }
42
43     @Override
44     public boolean onOptionsItemSelected(MenuItem item) {
45         if (item.getItemId() == android.R.id.home) // Press Back Icon
46         {
47             finish();
48         }
49
50         return super.onOptionsItemSelected(item);
51     }
52 }
53 }
```

```

1 package com.example.SAPLM;
2
3 import android.graphics.PorterDuff;
4 import android.view.animation.AccelerateDecelerateInterpolator;
5
6 import java.util.concurrent.Callable;
7 import java.util.concurrent.ExecutorService;
8 import java.util.concurrent.Executors;
9 import java.util.concurrent.Future;
10
11 public class MicrophoneHandler extends MainActivity {
12
13     public enum MicrophoneState { INHIBITED, LISTENING_KEYOWRD, LISTENING_COMMAND}
14
15     static private MicrophoneState currentState = MicrophoneState.LISTENING_KEYOWRD;
16
17     public static void microphoneOnClick(){
18         switch (currentState){
19             case LISTENING_KEYOWRD:
20                 PocketsphinxListener.stopRecognizer();
21                 setState(MicrophoneState.INHIBITED);
22                 break;
23             case LISTENING_COMMAND:
24                 GoogleSpeechRecognizer.forceStopRecognizer();
25                 //PocketsphinxListener.startListening();
26                 setState(MicrophoneState.LISTENING_KEYOWRD);
27                 break;
28             case INHIBITED:
29                 setState(MicrophoneState.LISTENING_KEYOWRD);
30                 PocketsphinxListener.startListening();
31                 break;
32         }
33     }
34 }
35
36     public static MicrophoneState getState(){
37         return currentState;
38     }
39
40     public static void setState(MicrophoneState desiredState){
41         switch (desiredState){
42             case INHIBITED:
43                 imgBtn_microphoneBackground.clearColorFilter();
44                 imgBtn_microphoneBackground.setColorFilter(Application.getContext().getColor
(R.color.ColorBackMicInhibited), PorterDuff.Mode.SRC_ATOP);
45                 imgBtn_microphone.setImageResource(R.drawable.ic_main_mic_off);
46                 endAnimation();
47                 currentState = MicrophoneState.INHIBITED;
48                 break;
49             case LISTENING_KEYOWRD:
50                 imgBtn_microphoneBackground.clearColorFilter();
51                 imgBtn_microphoneBackground.setColorFilter(Application.getContext().getColor
(R.color.ColorBackMicKeyword), PorterDuff.Mode.SRC_ATOP);
52                 imgBtn_microphone.setImageResource(R.drawable.ic_main_mic_listening);
53                 endAnimation();
54                 currentState = MicrophoneState.LISTENING_KEYOWRD;
55                 break;          146
56             case LISTENING_COMMAND:
57                 imgBtn_microphoneBackground.clearColorFilter();
58                 imgBtn_microphoneBackground.setColorFilter(Application.getContext().getColor
(R.color.ColorBackMicCommand), PorterDuff.Mode.SRC_ATOP);
59                 imgBtn_microphone.setImageResource(R.drawable.ic_main_mic_listening);
60                 startAnimation();

```

```

61             currentState = MicrophoneState.LISTENING_COMMAND;
62         }
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)
77                         .setInterpolator(new AccelerateDecelerateInterpolator());
78                     imgBtn_microphone.animate().scaleX(0.90f).setDuration(350).
79                         setInterpolator(new AccelerateDecelerateInterpolator());
80                     imgBtn_microphoneBackground.animate().scaleY(0.90f).setDuration(350)
81                         .setInterpolator(new AccelerateDecelerateInterpolator());
82                     imgBtn_microphone.animate().scaleY(0.90f).setDuration(350).
83                         setInterpolator(new AccelerateDecelerateInterpolator()).withEndAction(new Runnable() {
84                             @Override
85                             public void run() {
86                                 animationHasFinished = true;
87                             }
88                         });
89                     while(animationHasFinished==false);
90                     animationHasFinished= false;
91                     imgBtn_microphoneBackground.animate().scaleX(1.00f).setDuration(350)
92                         .setInterpolator(new AccelerateDecelerateInterpolator());
93                     imgBtn_microphone.animate().scaleX(1.00f).setDuration(350).
94                         setInterpolator(new AccelerateDecelerateInterpolator());
95                     imgBtn_microphoneBackground.animate().scaleY(1.00f).setDuration(350)
96                         .setInterpolator(new AccelerateDecelerateInterpolator());
97                     imgBtn_microphone.animate().scaleY(1.00f).setDuration(350).
98                         setInterpolator(new AccelerateDecelerateInterpolator()).withEndAction(new Runnable() {
99                             @Override
100                            public void run() {
101                                animationHasFinished = true;
102                            }
103                        });
104                    });
105                });
106            }
107
108        public static void endAnimation(){
109            terminateAnimation = true;          147
110        }
111
112    }
113

```

Archivo - TextToSpeechHandler.java

```
1 package com.example.SAPLM;
2
3 import android.speech.tts.TextToSpeech;
4 import android.speech.tts.Voice;
5
6 import java.util.Locale;
7 import java.util.Set;
8
9 public class TextToSpeechHandler {
10
11     private static TextToSpeech textToSpeech;
12     public static boolean isInitialized = false;
13
14     public static void initializeTextToSpeech() {
15         if (isInitialized == false) {
16             textToSpeech = new TextToSpeech(Application.getContext(), new TextToSpeech.
17                 OnInitListener() {
18                     @Override
19                     public void onInit(int status) {
20                         try {
21                             if (status != TextToSpeech.ERROR) {
22                                 textToSpeech.setLanguage(new Locale("spa"));
23                                 //textToSpeech.setLanguage(Locale.UK);
24                                 //textToSpeech.setOnUtteranceProgressListener(utteranceListener
25                             );
26                             textToSpeech.setPitch(1.0f);
27                             textToSpeech.setSpeechRate(1.0f);
28                             textToSpeech.getVoices();
29                             isInitialized = true;
30                         }
31                     } catch (Throwable th) {
32                         Application.toastMessage(th.toString());
33                     }
34                 });
35         }
36
37     public static void addToSpeakingQueue(String text) {
38         try {
39             if (isInitialized) {
40                 textToSpeech.speak(text, TextToSpeech.QUEUE_ADD, null, "232");
41             }
42         } catch (Throwable th) {
43             th.printStackTrace();
44             Application.toastMessage(th.toString());
45         }
46     }
47
48     public static void shutdownTextSpeech() {
49         if (textToSpeech != null) {
50
51             textToSpeech.stop();
52             textToSpeech.shutdown();
53         }
54     }
55
56     public static Set<Voice> getVoicesList() {
57         if (isInitialized) {
58             return textToSpeech.getVoices();
59         } else {
60             return null;
61         }
62     }
63 }
```

```
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
```

```

1 package com.example.SAPLM;
2
3 import java.io.File;
4
5 import edu.cmu.pocketsphinx.Assets;
6 import edu.cmu.pocketsphinx.Hypothesis;
7 import edu.cmu.pocketsphinx.RecognitionListener;
8 import edu.cmu.pocketsphinx.SpeechRecognizer;
9 import edu.cmu.pocketsphinx.SpeechRecognizerSetup;
10
11 public final class PocketsphinxListener implements RecognitionListener {
12     private static SpeechRecognizer recognizer;
13     private static String KEYPHRASE = "asistente";
14     private static final String KWS_SEARCH = "wakeup";
15
16
17     @Override
18     public void onPartialResult(Hypothesis hypothesis) {
19         if (hypothesis == null)
20             return;
21
22         String text = hypothesis.getHypstr();
23
24         if (text.contains(KEYPHRASE)) {
25
26             Application.toastMessage(KEYPHRASE);
27             recognizer.stop();
28             //recognizer.cancel();
29             //recognizer.shutdown();
30             GoogleSpeechRecognizer.startSpeechListening();
31             //recognizer.startListening(KWS_SEARCH);
32         }
33
34     }
35
36
37     @Override
38     public void onResult(Hypothesis hypothesis) {
39         if (hypothesis != null) {
40             String text = hypothesis.getHypstr();
41
42         }
43     }
44
45     @Override
46     public void onBeginningOfSpeech() {
47     }
48
49     @Override
50     public void onEndOfSpeech() {
51
52     }
53
54     public static void setupRecognizer(){
55         try {
56             Assets assets = new Assets(Application.getContext());
57             File assetDir = assets.syncAssets();
58             recognizer = SpeechRecognizerSetup.defaultSetup()
59                 .setAcousticModel(new File(assetDir, "es-es-ptm"))
60                 .setDictionary(new File(assetDir, "cmudict-es-es.dict"))
61                 .setKeywordThreshold(Float.MIN_VALUE)
62                 )
63                 .setRawLogDir(assetDir) // To disable logging of raw audio comment out

```

Archivo - PocketsphinxListener.java

```
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

```
1 package com.example.SAPLM;
2
3 import android.content.Intent;
4 import android.os.Bundle;
5 import android.speech.RecognitionListener;
6 import android.speech.RecognizerIntent;
7 import android.speech.SpeechRecognizer;
8 import android.view.View;
9
10 import com.example.SAPLM.bluetoothActivities.BluetoothConnection;
11 import com.example.SAPLM.bluetoothActivities.UnifiedTransmissionProtocol;
12
13 import org.apache.commons.lang3.StringUtils;
14
15 import java.text.Normalizer;
16 import java.text.SimpleDateFormat;
17 import java.util.ArrayList;
18 import java.util.Arrays;
19 import java.util.Calendar;
20 import java.util.Date;
21 import java.util.Locale;
22
23 public class GoogleSpeechRecognizer extends MainActivity implements RecognitionListener {
24
25     private static SpeechRecognizer speechRecognizer = null;
26
27     private boolean forceCancel = false;
28
29     public static void startSpeechListening() {
30
31
32         String language = "es-ES";
33         Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
34         intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,language);
35         intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, language);
36         intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_PREFERENCE, language);
37         intent.putExtra(RecognizerIntent.EXTRA_ONLY_RETURN_LANGUAGE_PREFERENCE, language);
38         intent.putExtra(RecognizerIntent.EXTRA_CALLING_PACKAGE, Application.getActivity().
39             getPackageName());
40         intent.putExtra(RecognizerIntent.EXTRA_PARTIAL_RESULTS,true);
41         if (Application.isSystemOnline()==false) { intent.putExtra(RecognizerIntent.
42             EXTRA_PREFER_OFFLINE,true); }
43         speechRecognizer.startListening(intent);
44
45         if (Application.isSystemOnline()) {
46             Thread autoStop = new Thread() {
47                 @Override
48                 public void run() {
49                     super.run();
50                     try {
51                         Thread.sleep(4000);
52                         forceStopRecognizer();
53                     } catch (Throwable th) {
54                         Application.toastMessage(th.toString());
55                     }
56                 };
57             autoStop.start();
58         }
59     }
60 }
```

```

62     public static void setupRecognizer(){
63         speechRecognizer = SpeechRecognizer.createSpeechRecognizer(Application.getContext()
64     );
65     speechRecognizer.setRecognitionListener(new GoogleSpeechRecognizer());
66 }
67
68     public static void shutdownRecognizer(){
69         speechRecognizer.stopListening();
70         speechRecognizer.cancel();
71         speechRecognizer.destroy();
72         speechRecognizer = null;
73         System.gc();
74 }
75
76     public static void forceStopRecognizer(){
77         Application.getActivity().runOnUiThread(
78             new Runnable() {
79                 @Override
80                 public void run() {
81                     speechRecognizer.stopListening();
82                     MicrophoneHandler.setState(MicrophoneHandler.MicrophoneState.
83 LISTENING_KEYWORD);
84                 }
85             }
86         );
87 }
88
89     public static void analyseListenerResults(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); 153
109                 UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
110                 break outerloop;
111             }
112
113             if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(
hypothesisSplit).contains("luz")) {

```

Archivo - GoogleSpeechRecognizer.java

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

154

```

162             break outerloop;
163         }
164
165         if (Arrays.asList(hypothesisSplit).contains("llamar") && Arrays.asList(
166             hypothesisSplit).contains("enfermeria")) {
167             UnifiedTransmissionProtocol.modulesList.get(3).getDevicesList().get(2).
168             setData(Boolean.TRUE);
169             UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
170             break outerloop;
171         }
172
173         if (Arrays.asList(hypothesisSplit).contains("subir") && Arrays.asList(
174             hypothesisSplit).contains("toda") && Arrays.asList(hypothesisSplit).contains("cortina")) {
175             UnifiedTransmissionProtocol.modulesList.get(3).getDevicesList().get(1).
176             setData((byte)7);
177             UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
178             break outerloop;
179         }
180
181         if (Arrays.asList(hypothesisSplit).contains("bajar") && Arrays.asList(
182             hypothesisSplit).contains("toda") && Arrays.asList(hypothesisSplit).contains("cortina")) {
183             UnifiedTransmissionProtocol.modulesList.get(3).getDevicesList().get(1).
184             setData((byte)0);
185             UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
186             break outerloop;
187         }
188
189         if (Arrays.asList(hypothesisSplit).contains("bajar") && Arrays.asList(
190             hypothesisSplit).contains("toda") && Arrays.asList(hypothesisSplit).contains("camilla")) {
191             UnifiedTransmissionProtocol.modulesList.get(3).getDevicesList().get(0).
192             setData((byte)0);
193             UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
194             break outerloop;
195         }
196
197         if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(
198             hypothesisSplit).contains("rele")) {
199             for (String word : hypothesisSplit) {
200                 if (StringUtils.isNumeric(word) == true) {
201                     int number = Integer.parseInt(word);
202                     if (number<4) {
203                         UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList().
204                         get(number).setData(Boolean.TRUE);
205                         UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem(
206                         );
207                     }
208                     break outerloop;
209                 }
210             }
211         }
212
213         if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(
214             hypothesisSplit).contains("rele")) {
215             for (String word : hypothesisSplit) {
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2197
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2279
2280
2281
2282
228
```

Archivo - GoogleSpeechRecognizer.java

```
211             if (StringUtils.isNumeric(word) == true) {
212                 int number = Integer.parseInt(word);
213                 if (number<4) {
214                     UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList()
215                         .get(number).setData(Boolean.FALSE);
216                     UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem(
217                         );
218                 }
219             }
220         }
221
222         if (Arrays.asList(hypothesisSplit).contains("decir") && Arrays.asList(
223             hypothesisSplit).contains("hora")){
224             Date currentTime = Calendar.getInstance().getTime();
225             SimpleDateFormat df = new SimpleDateFormat("d-MMM-yyyy", new Locale("spa"))
226             ;
227             String todayAsString = df.format(currentTime);
228             SimpleDateFormat onlyTimeFormar = new SimpleDateFormat("HH:mm:ss", new
229             Locale("spa")));
230             String onlyTimeString = onlyTimeFormar.format(currentTime);
231
232             //TextToSpeechHandler.initializeTextToSpeech();
233             TextToSpeechHandler.addToSpeakingQueue("Hoy es " + todayAsString + " y son
234             las" + onlyTimeString);
235             Application.toastMessage("Decir hora");
236         }
237     }
238
239     @Override
240     public void onReadyForSpeech(Bundle params) {
241         MicrophoneHandler.setState(MicrophoneHandler.MicrophoneState.LISTENING_COMMAND);
242     }
243
244     @Override
245     public void onBeginningOfSpeech() {
246
247     }
248
249     @Override
250     public void onRmsChanged(float rmsdB) {
251
252     }
253
254     @Override
255     public void onBufferReceived(byte[] buffer) {
256
257     }
258
259     @Override
260     public void onEndOfSpeech() {
261
262         //Application.toastMessage("End of speech");
263     }
264
265     @Override
266     public void onError(int error) {
267
268         //Application.toastMessage("On error" + error);
269
270         if (forceCancel==false) {
```

Archivo - GoogleSpeechRecognizer.java

```
268         PocketsphinxListener.startListening();
269         MicrophoneHandler.setState(MicrophoneHandler.MicrophoneState.LISTENING_KEYWORD)
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     ;
279     analyseListenerResults(resultsArrayList, confidenceScore);
280
281     StringBuilder builder = new StringBuilder();
282     for (String line : resultsArrayList) {
283         builder.append(line + "\n");
284     }
285
286     MainActivity.textView_speechRecognitionResults.setText(builder.toString());
287     MainActivity.textView_speechRecognitionResults.setVisibility(View.VISIBLE);
288
289
290     MicrophoneHandler.setState(MicrophoneHandler.MicrophoneState.LISTENING_KEYWORD);
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

```
1 package com.example.SAPLM.commandActivity;
2
3 import com.example.SAPLM.R;
4 import com.example.SAPLM.bluetoothActivities.UnifiedTransmissionProtocol;
5
6 import java.util.List;
7
8 public class CommandItem {
9     public enum ItemType { CHILD, PARENT}
10    public enum BoardGroupState {FOLDED, UNFOLDED}
11
12    private String itemName;
13    public ItemType itemType = ItemType.CHILD;
14    private int deviceStateIconID;
15    private int deviceStateInteger = 0;
16    private int moduleIndex = 0;
17    private int deviceIndex = 0;
18    private List<String> childInternalID;
19    private BoardGroupState groupState = BoardGroupState.UNFOLDED;
20
21
22
23    public CommandItem( int moduleIndex, int deviceIndex ){
24        this.itemName = UnifiedTransmissionProtocol.modulesList.get(moduleIndex) .
25            getDevicesList().get(deviceIndex).getName();
26        this.itemType = ItemType.CHILD;
27        this.moduleIndex = moduleIndex;
28        this.deviceIndex = deviceIndex;
29        this.deviceStateIconID = UnifiedTransmissionProtocol.modulesList.get(moduleIndex) .
30            getDevicesList().get(deviceIndex).getIconResourceId();
31    }
32
33    public CommandItem(int boardInternalId){
34        this.itemName = UnifiedTransmissionProtocol.modulesList.get(boardInternalId) .
35            getModuleName();
36        this.itemType = ItemType.PARENT;
37        this.deviceStateIconID = UnifiedTransmissionProtocol.modulesList.get(boardInternalId)
38            .getIconResourceId();
39    }
40
41
42    //Getters
43
44    public BoardGroupState getGroupState (){
45        return groupState;
46    }
47
48    public int getDeviceStateIconID() {
49        return deviceStateIconID;
50    }
51
52    public int getDeviceStateInteger() { 158
53        return deviceStateInteger;
54    }
55
56    public int getDeviceIndex() {
57        return deviceIndex;
58    }
```

```
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().
79             get(deviceIndex).getDataAsString();
80     }
81
82     public int getProgressBarValue() {
83         return UnifiedTransmissionProtocol.modulesList.get(moduleIndex).getDevicesList().
84             get(deviceIndex).getProgressBarValue();
85     }
86
87     // Setters
88     public void setChildInternalID(List<String> childInternalID) {
89         this.childInternalID = childInternalID;
90     }
91
92     public void setDeviceStateIconID(int deviceStateIconID) {
93         this.deviceStateIconID = deviceStateIconID;
94     }
95
96     public void setDeviceStateInteger(int deviceStateInteger) {
97         this.deviceStateInteger = deviceStateInteger;
98     }
99
100    public void setModuleIndex(int moduleIndex) {
101        this.moduleIndex = moduleIndex;
102    }
103
104    public void setItemName(String itemName) {
105        this.itemName = itemName;
106    }
107
108    public void setItemType(ItemType itemType) {
109        this.itemType = itemType;
110    }
111
112    public void setDeviceIndex(int deviceIndex) {
113        this.deviceIndex = deviceIndex;
114    }
115
116    public void setGroupState (BoardGroupState groupState) {
117        this.groupState = groupState;
118    }
119 }
```

Archivo - CommandsActivity.java

```
1 package com.example.SAPLM.commandActivity;
2
3 import android.app.Activity;
4 import android.content.Context;
5 import android.content.Intent;
6 import android.graphics.PorterDuff;
7 import android.graphics.drawable.Drawable;
8 import android.os.Bundle;
9 import androidx.core.content.ContextCompat;
10 import androidx.appcompat.app.AppCompatActivity;
11 import androidx.appcompat.widget.Toolbar;
12 import androidx.recyclerview.widget.DefaultItemAnimator;
13 import androidx.recyclerview.widget.DividerItemDecoration;
14 import androidx.recyclerview.widget.LinearLayoutManager;
15 import androidx.recyclerview.widget.RecyclerView;
16
17 import android.os.Looper;
18 import android.view.MenuItem;
19 import android.view.View;
20
21 import com.example.SAPLM.Application;
22 import com.example.SAPLM.R;
23 import com.example.SAPLM.bluetoothActivities.BluetoothConnection;
24 import com.example.SAPLM.settingsActivities.AvailableSetting;
25 import com.example.SAPLM.settingsActivities.RecyclerTouchListener;
26 import com.example.SAPLM.settingsActivities.SettingsLayoutAdapter;
27
28 import java.util.ArrayList;
29 import java.util.Arrays;
30 import java.util.List;
31
32 public class CommandsActivity extends AppCompatActivity {
33     private static Toolbar toolbar;
34
35     public static List<CommandItem> commandList = new ArrayList<>();
36     private RecyclerView recyclerView;
37     public static CommandsListAdapter mAdapter;
38
39
40     @Override
41     protected void onCreate(Bundle savedInstanceState) {
42         super.onCreate(savedInstanceState);
43         setContentView(R.layout.commands_layout);
44         Application.setActivity(this);
45         Application.setContext(this);
46
47         toolbar = this.findViewById(R.id.toolbar);
48
49         setSupportActionBar(toolbar);
50
51         // add back arrow to toolbar
52         if (getSupportActionBar() != null){
53             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
54             getSupportActionBar().setDisplayShowHomeEnabled(true);
55
56             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
      abc_ic_ab_back_material); 160
      upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
      ;
57             getSupportActionBar().setHomeAsUpIndicator(upArrow);
58         }
59
60         recyclerView = (RecyclerView) findViewById(R.id.commandsRecyclerView);
61 }
```

```

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(
70             getApplicationContext());
71
72         // horizontal RecyclerView
73         // keep movie_list_row.xml width to `wrap_content`
74         // RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(
75             getApplicationContext(), LinearLayoutManager.HORIZONTAL, false);
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,
82             LinearLayoutManager.HORIZONTAL, 16));
82         recyclerView.setItemAnimator(new DefaultItemAnimator());
83
84         recyclerView.setAdapter(mAdapter);
85
86         // row click listener
87         recyclerView.addOnItemTouchListener(new RecyclerTouchListener(getApplicationContext(),
88             recyclerView, new RecyclerTouchListener.ClickListener() {
89
90             @Override
91             public void onClick(View view, int position) {
92                 CommandItem commandItem = commandList.get(position);
93                 if (commandItem.getItemType() == CommandItem.ItemType.PARENT) {
94
95                     } else{
96                         commandItem.onClickHandler();
97                     }
98
99                     mAdapter.notifyDataSetChanged();
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() {
161

```

Archivo - CommandsActivity.java

```
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(
142     Arrays.asList("LED")));
143
144     commandList.add(new CommandItem(1));
145     commandList.add(new CommandItem(1,0));
146     commandList.add(new CommandItem(1, 1));
147
148     commandList.add(new CommandItem(2));
149     commandList.add(new CommandItem(2,0));
150     commandList.add(new CommandItem(2, 1));
151     commandList.add(new CommandItem(2, 2));
152     commandList.add(new CommandItem(2, 3));
153
154     commandList.add(new CommandItem(3));
155     commandList.add(new CommandItem(3,0));
156     commandList.add(new CommandItem(3, 1));
157     commandList.add(new CommandItem(3, 2));
158 }
159 }
```

```

1 package com.example.SAPLM.commandActivity;
2
3 import android.graphics.Color;
4 import android.graphics.PorterDuff;
5 import android.graphics.Typeface;
6 import android.graphics.drawable.ColorDrawable;
7 import android.graphics.drawable.Drawable;
8
9 import androidx.constraintlayout.widget.ConstraintLayout;
10 import androidx.constraintlayout.widget.Guideline;
11 import androidx.recyclerview.widget.RecyclerView;
12 import android.view.LayoutInflater;
13 import android.view.View;
14 import android.view.ViewGroup;
15 import android.widget.ImageView;
16 import android.widget.ProgressBar;
17 import android.widget.TextView;
18
19 import com.example.SAPLM.Application;
20 import com.example.SAPLM.R;
21 import com.example.SAPLM.settingsActivities.AvailableSetting;
22 import com.example.SAPLM.settingsActivities.SettingsLayoutAdapter;
23
24 import java.util.List;
25
26 public class CommandsListAdapter extends RecyclerView.Adapter<CommandsListAdapter.ViewHolder> {
27
28     private List<CommandItem> commandsList;
29
30     public class ViewHolder extends RecyclerView.ViewHolder {
31         public TextView command_title, command_state_text;
32         public ImageView command_icon, arrow_list;
33         public ProgressBar status_progress_bar;
34         public Guideline main_guide;
35
36         public viewHolder(View view) {
37             super(view);
38             command_title = (TextView) view.findViewById(R.id.command_title_textView);
39             status_progress_bar = (ProgressBar) view.findViewById(R.id.
40             command_state_progressBar);
41             command_state_text = (TextView) view.findViewById(R.id.command_state_textView);
42             command_icon = (ImageView) view.findViewById(R.id.command_icon);
43             arrow_list = (ImageView) view.findViewById(R.id.arrow_list_imageButton);
44             main_guide = (Guideline) view.findViewById(R.id.guideline_main);
45         }
46     }
47
48     public CommandsListAdapter(List<CommandItem> cmdList) {
49         this.commandsList = cmdList;
50     }
51
52     @Override
53     public viewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
54         View itemView = LayoutInflater.from(parent.getContext())
55             .inflate(R.layout.commands_layout_child_item_style, parent, false);
56
57         return new viewHolder(itemView);
58     }
59
60     @Override
61     public void onBindViewHolder(viewHolder holder, int position) {

```

## Archivo - CommandsListAdapter.java

```

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
100    cmdIcon.setColorFilter(Application.getContext().getColor(R.color.colorPrimary),
    PorterDuff.Mode.SRC_ATOP);
101    holder.command_icon.setImageDrawable(cmdIcon);
102
103    @Override
104    public int getItemCount() {
105        return commandsList.size();
106    }
107 }

```

Archivo - settings\_layout.java

```
1 package com.example.SAPLM.settingsActivities;
2
3 import android.app.Activity;
4 import android.content.Context;
5 import android.content.Intent;
6 import android.graphics.PorterDuff;
7 import android.graphics.drawable.Drawable;
8 import android.os.Bundle;
9 import androidx.core.content.ContextCompat;
10 import androidx.appcompat.app.AppCompatActivity;
11 import androidx.recyclerview.widget.DefaultItemAnimator;
12 import androidx.recyclerview.widget.DividerItemDecoration;
13 import androidx.recyclerview.widget.LinearLayoutManager;
14 import androidx.recyclerview.widget.RecyclerView;
15 import androidx.appcompat.widget.Toolbar;
16 import android.view.MenuItem;
17 import android.view.View;
18
19 import com.example.SAPLM.Application;
20 import com.example.SAPLM.R;
21
22 import java.util.ArrayList;
23 import java.util.List;
24
25 public class settings_layout extends AppCompatActivity {
26
27     private static Toolbar toolbar;
28
29     public static List<AvailableSetting> settingsList = new ArrayList<>();
30     private RecyclerView recyclerView;
31     public static SettingsLayoutAdapter mAdapter;
32
33
34     @Override
35     protected void onCreate(Bundle savedInstanceState) {
36         super.onCreate(savedInstanceState);
37         setContentView(R.layout.settings_layout);
38         Application.setActivity(this);
39         Application.setContext(this);
40
41         toolbar = findViewById(R.id.toolbar);
42
43         setSupportActionBar(toolbar);
44
45         // add back arrow to toolbar
46         if (getSupportActionBar() != null){
47             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
48             getSupportActionBar().setDisplayShowHomeEnabled(true);
49
50
51             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
52             abc_ic_ab_back_material);
53             upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
54             ;
55             getSupportActionBar().setHomeAsUpIndicator(upArrow);
56             165
57             recyclerView = (RecyclerView) findViewById(R.id.settingsRecyclerView);
58             mAdapter = new SettingsLayoutAdapter(settingsList);
59             recyclerView.setHasFixedSize(true);
60
61 }
```

Archivo - settings\_layout.java

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

Archivo - settings\_layout.java

```
119         R.drawable.ic_setting_internet_connectivity,
120         InternetConnectivityActivity.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.
132             voice_recognition_settings_title),
133         Application.getContext().getString(R.string.
134             voice_recognition_settings_subtitle),
135         R.drawable.ic_setting_voice_recognition,
136         VoiceRecognitionSettingsActivity.class
137     ));
138
139     settingsList.add(new AvailableSetting(
140         Application.getContext().getString(R.string.commands_settings_title),
141         Application.getContext().getString(R.string.commands_settings_subtitle),
142         R.drawable.ic_setting_commands_3,
143         CommandsSettingsActivity.class
144     ));
145
146     settingsList.add(new AvailableSetting(
147         Application.getContext().getString(R.string.contacts_communication_title),
148         Application.getContext().getString(R.string.contacts_communication_subtitle
149     ),
150         R.drawable.ic_setting_contacts_and_communication,
151         ContactsCommunicationActivity.class
152     ));
153
154     settingsList.add(new AvailableSetting(
155         Application.getContext().getString(R.string.appearance_miscellaneous_title)
156
157         Application.getContext().getString(R.string.
158             appearance_miscellaneous_subtitle),
159         R.drawable.ic_setting_appearance_and_miscellaneous,
160         AppearanceMiscellaneousActivity.class
161     ));
162
163     settingsList.add(new AvailableSetting(
164         Application.getContext().getString(R.string.
165             developer_console_activity_title),
166         Application.getContext().getString(R.string.developer_console_subtitle),
167         R.drawable.ic_setting_developer_console,
168         DeveloperConsoleActivity.class
169     ));
170
171     settingsList.add(new AvailableSetting(
172         Application.getContext().getString(R.string.information_help_title),
173         Application.getContext().getString(R.string.information_help_subtitle),
174         R.drawable.ic_setting_information_and_help,
175         InformationHelpActivity.class
176     ));
177
178     // notify adapter about data set changes
179     // so that it will render the list with new data
180     // mAdapter.notifyDataSetChanged();
181 }
```

```
176  
177  
178 }  
179
```

## Archivo - TerminalManager.java

```
1 package com.example.SAPLM.settingsActivities;
2
3 import android.graphics.Typeface;
4 import android.text.SpannableStringBuilder;
5 import android.text.Spanned;
6 import android.text.style.StyleSpan;
7
8 import com.example.SAPLM.bluetoothActivities.UnifiedTransmissionProtocol;
9
10 import java.util.ArrayList;
11 import java.util.Arrays;
12 import java.util.List;
13 import java.util.regex.Matcher;
14 import java.util.regex.Pattern;
15
16
17 public class TerminalManager {
18
19     public static void handleTerminalInput(String newTerminalInput) {
20
21         }
22
23     public static List<String> currentParameters = new ArrayList<>();
24
25     public static List<TerminalCommand> commandsList = new ArrayList<>(Arrays.asList(
26             new TerminalCommand("COMPOSE", new Runnable() {
27                 @Override
28                 public void run() {
29                     UnifiedTransmissionProtocol.CommandType command =
30                         UnifiedTransmissionProtocol.CommandType.definedCommandList.stream().filter(p-> p.
31                         getCommandTag().equals((currentParameters.get(0)))).findFirst().orElse(null);
32                     UnifiedTransmissionProtocol.Module target = UnifiedTransmissionProtocol.
33                         modulesList.stream().filter(p-> p.getModuleAlias().equals((currentParameters.get(1)))).
34                         findFirst().orElse(null);
35
36                     if (command==null) {
37                         SpannableStringBuilder builder= new SpannableStringBuilder();
38                         builder.append("Tipo de comando \\"");
39                         .append(currentParameters.get(0), new StyleSpan(Typeface.
40                             BOLD), Spanned.SPAN_EXCLUSIVE_EXCLUSIVE)
41                         .append("\" no reconocido");
42                         DeveloperConsoleActivity.sendTextToTerminal(builder);
43                         return;
44                     }
45
46                     if (target==null) {
47                         SpannableStringBuilder builder= new SpannableStringBuilder();
48                         builder.append("Alias de modulo \\"");
49                         .append(currentParameters.get(1), new StyleSpan(Typeface.
50                             BOLD), Spanned.SPAN_EXCLUSIVE_EXCLUSIVE)
51                         .append("\" no encontrado");
52                         DeveloperConsoleActivity.sendTextToTerminal(builder);
53                         return;
54                     }
55
56                     System.out.println( insertSpaces(UnifiedTransmissionProtocol.bytesToHex(
57                         UnifiedTransmissionProtocol.getCommand_buffer() ), " ", 2 ) );
58                 }
59             })
60         ));
61
62     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

```
1 package com.example.SAPLM.settingsActivities;
2
3 public class AvailableSetting {
4     private String title, subtitle;
5     private int iconResourceID;
6     private Class targetActivity;
7
8     public AvailableSetting() {
9     }
10
11    public AvailableSetting(String title, String subtitle, int iconResourceID, Class
targetActivity) {
12        this.title = title;
13        this.subtitle = subtitle;
14        this.iconResourceID = iconResourceID;
15        this.targetActivity = targetActivity;
16    }
17
18    public String getTitle() {
19        return title;
20    }
21
22    public void setTitle(String name) {
23        this.title = name;
24    }
25
26    public String getSubtitle() {
27        return subtitle;
28    }
29
30    public void setSubtitle(String subtitle) {
31        this.subtitle = subtitle;
32    }
33
34    public int getIconResourceID() {
35        return iconResourceID;
36    }
37
38    public void setIconResourceID(int iconResourceID) {
39        this.iconResourceID = iconResourceID;
40    }
41
42    public Class getTargetActivity() { return targetActivity; }
43
44    public void setTargetActivity(Class targetActivity) { this.targetActivity =
targetActivity; }
45 }
46
```

Archivo - RecyclerTouchListener.java

```
1 package com.example.SAPLM.settingsActivities;
2
3 import android.content.Context;
4 import androidx.recyclerview.widget.RecyclerView;
5 import android.view.GestureDetector;
6 import android.view.MotionEvent;
7 import android.view.View;
8
9 /**
10 * Created by Ravi Tamada on 03/09/16.
11 * updated by Ravi on 14/11/17
12 * www.androidhive.info
13 */
14 public class RecyclerTouchListener implements RecyclerView.OnItemTouchListener {
15
16     private GestureDetector gestureDetector;
17     private ClickListener clickListener;
18
19     public RecyclerTouchListener(Context context, final RecyclerView recyclerView, final
20         ClickListener clickListener) {
21         this.clickListener = clickListener;
22         gestureDetector = new GestureDetector(context, new GestureDetector.
23             SimpleOnGestureListener() {
24                 @Override
25                 public boolean onSingleTapUp(MotionEvent e) {
26                     return true;
27                 }
28
29                 @Override
30                 public void onLongPress(MotionEvent e) {
31                     View child = recyclerView.findChildViewUnder(e.getX(), e.getY());
32                     if (child != null && clickListener != null) {
33                         clickListener.onLongClick(child, recyclerView.getChildAdapterPosition(
34                             child));
35                     }
36                 }
37             });
38     }
39
40     @Override
41     public boolean onInterceptTouchEvent(RecyclerView rv, MotionEvent e) {
42
43         View child = rv.findChildViewUnder(e.getX(), e.getY());
44         if (child != null && clickListener != null && gestureDetector.onTouchEvent(e)) {
45             clickListener.onClick(child, rv.getChildAdapterPosition(child));
46         }
47         return false;
48     }
49
50     @Override
51     public void onTouchEvent(RecyclerView rv, MotionEvent e) {
52
53     }
54
55     @Override
56     public interface ClickListener {
57         void onClick(View view, int position);
58
59         void onLongClick(View view, int position);
60     }
```

```
61 }
```

Archivo - SettingsLayoutAdapter.java

```
1 package com.example.SAPLM.settingsActivities;
2
3 import android.graphics.PorterDuff;
4 import android.graphics.drawable.Drawable;
5 import androidx.recyclerview.widget.RecyclerView;
6 import android.view.LayoutInflater;
7 import android.view.View;
8 import android.view.ViewGroup;
9 import android.widget.ImageView;
10 import android.widget.TextView;
11
12 import com.example.SAPLM.Application;
13 import com.example.SAPLM.R;
14
15 import java.util.List;
16
17 public class SettingsLayoutAdapter extends RecyclerView.Adapter<SettingsLayoutAdapter.
viewHolder> {
18
19     private List<AvailableSetting> settingsList;
20
21     public class viewHolder extends RecyclerView.ViewHolder {
22         public TextView setting_title, setting_subtitle;
23         public ImageView setting_icon;
24
25         public viewHolder(View view) {
26             super(view);
27             setting_title = (TextView) view.findViewById(R.id.setting_title);
28             setting_subtitle = (TextView) view.findViewById(R.id.setting_subtitle);
29             setting_icon = (ImageView) view.findViewById(R.id.setting_icon);
30         }
31     }
32
33
34     public SettingsLayoutAdapter(List<AvailableSetting> settingsList) {
35         this.settingsList = settingsList;
36     }
37
38     @Override
39     public viewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
40         View itemView = LayoutInflater.from(parent.getContext())
41             .inflate(R.layout.settings_layout_style, parent, false);
42
43         return new viewHolder(itemView);
44     }
45
46     @Override
47     public void onBindViewHolder(viewHolder holder, int position) {
48         AvailableSetting setting = settingsList.get(position);
49         holder.setting_title.setText(setting.getTitle());
50         holder.setting_subtitle.setText(setting.getSubtitle());
51
52         Drawable icon = Application.getContext().getDrawable(setting.getIconResourceID());
53         icon.setColorFilter(Application.getContext().getColor(R.color.colorPrimary),
PorterDuff.Mode.SRC_ATOP);
54         holder.setting_icon.setImageDrawable(icon);
55     }
56
57     @Override
58     public int getItemCount() {
59         return settingsList.size();
60     }
61 }
```

Archivo - InformationHelpActivity.java

```
1 package com.example.SAPLM.settingsActivities;
2
3 import android.graphics.PorterDuff;
4 import android.graphics.drawable.Drawable;
5 import android.os.Bundle;
6 import androidx.core.content.ContextCompat;
7 import androidx.appcompat.app.AppCompatActivity;
8 import androidx.appcompat.widget.Toolbar;
9 import android.view.MenuItem;
10
11 import com.example.SAPLM.Application;
12 import com.example.SAPLM.R;
13
14 public class InformationHelpActivity extends AppCompatActivity {
15     private static Toolbar toolbar;
16
17     @Override
18     protected void onCreate(Bundle savedInstanceState) {
19         super.onCreate(savedInstanceState);
20         setContentView(R.layout.information_help_layout);
21         Application.setActivity(this);
22         Application.setContext(this);
23
24         toolbar = this.findViewById(R.id.toolbar);
25
26         setSupportActionBar(toolbar);
27
28         // add back arrow to toolbar
29         if (getSupportActionBar() != null){
30             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
31             getSupportActionBar().setDisplayShowHomeEnabled(true);
32
33             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
34             abc_ic_ab_back_material);
35             upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
36             ;
37             getSupportActionBar().setHomeAsUpIndicator(upArrow);
38         }
39     }
40
41     @Override
42     public boolean onOptionsItemSelected(MenuItem item) {
43         if (item.getItemId() == android.R.id.home) // Press Back Icon
44         {
45             finish();
46         }
47     }
48 }
49
```

```

1 package com.example.SAPLM.settingsActivities;
2
3 import android.content.Context;
4 import android.content.res.Resources;
5 import android.content.res.TypedArray;
6 import android.graphics.Canvas;
7 import android.graphics.Rect;
8 import android.graphics.drawable.Drawable;
9 import androidx.recyclerview.widget.LinearLayoutManager;
10 import androidx.recyclerview.widget.RecyclerView;
11 import android.util.TypedValue;
12 import android.view.View;
13
14 /**
15  * Created by Ravi on 30/10/15.
16  * updated by Ravi on 14/11/17
17  */
18 public class MyDividerItemDecoration extends RecyclerView.ItemDecoration {
19
20     private static final int[] ATTRS = new int[]{
21         android.R.attr.listDivider
22     };
23
24     public static final int HORIZONTAL_LIST = LinearLayoutManager.HORIZONTAL;
25
26     public static final int VERTICAL_LIST = LinearLayoutManager.VERTICAL;
27
28     private Drawable mDivider;
29     private int mOrientation;
30     private Context context;
31     private int margin;
32
33     public MyDividerItemDecoration(Context context, int orientation, int margin) {
34         this.context = context;
35         this.margin = margin;
36         final TypedArray a = context.obtainStyledAttributes(ATTRS);
37         mDivider = a.getDrawable(0);
38         a.recycle();
39         setOrientation(orientation);
40     }
41
42     public void setOrientation(int orientation) {
43         if (orientation != HORIZONTAL_LIST && orientation != VERTICAL_LIST) {
44             throw new IllegalArgumentException("invalid orientation");
45         }
46         mOrientation = orientation;
47     }
48
49     @Override
50     public void onDrawOver(Canvas c, RecyclerView parent, RecyclerView.State state) {
51         if (mOrientation == VERTICAL_LIST) {
52             drawVertical(c, parent);
53         } else {
54             drawHorizontal(c, parent);
55         }
56     }
57
58     public void drawVertical(Canvas c, RecyclerView parent) {
59         final int left = parent.getPaddingLeft();
60         final int right = parent.getWidth() - parent.getPaddingRight();
61
62         final int childCount = parent.getChildCount();
63         for (int i = 0; i < childCount; i++) {
176

```

```

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

```
1 package com.example.SAPLM.settingsActivities;
2
3 import android.graphics.PorterDuff;
4 import android.graphics.drawable.Drawable;
5 import android.os.Bundle;
6 import androidx.core.content.ContextCompat;
7 import androidx.appcompat.app.AppCompatActivity;
8 import androidx.appcompat.widget.Toolbar;
9 import android.view.MenuItem;
10
11 import com.example.SAPLM.Application;
12 import com.example.SAPLM.R;
13
14 public class CommandsSettingsActivity extends AppCompatActivity {
15     private static Toolbar toolbar;
16
17     @Override
18     protected void onCreate(Bundle savedInstanceState) {
19         super.onCreate(savedInstanceState);
20         setContentView(R.layout.commands_settings_layout);
21         Application.setActivity(this);
22         Application.setContext(this);
23
24         toolbar = this.findViewById(R.id.toolbar);
25
26         setSupportActionBar(toolbar);
27
28         // add back arrow to toolbar
29         if (getSupportActionBar() != null){
30             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
31             getSupportActionBar().setDisplayShowHomeEnabled(true);
32
33             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
34             abc_ic_ab_back_material);
35             upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
36             ;
37             getSupportActionBar().setHomeAsUpIndicator(upArrow);
38         }
39     }
40
41     @Override
42     public boolean onOptionsItemSelected(MenuItem item) {
43         if (item.getItemId() == android.R.id.home) // Press Back Icon
44         {
45             finish();
46         }
47     }
48 }
49
```

Archivo - DeveloperConsoleActivity.java

```
1 package com.example.SAPLM.settingsActivities;
2
3 import android.graphics.Color;
4 import android.graphics.PorterDuff;
5 import android.graphics.Typeface;
6 import android.graphics.drawable.Drawable;
7 import android.os.Bundle;
8 import androidx.core.content.ContextCompat;
9 import androidx.appcompat.app.AppCompatActivity;
10 import androidx.appcompat.widget.Toolbar;
11
12 import android.os.Looper;
13 import android.text.Html;
14 import android.text.Spanned;
15 import android.text.SpannedString;
16 import android.util.TypedValue;
17 import android.view.MenuItem;
18 import android.view.View;
19 import android.widget.EditText;
20 import android.widget.ImageButton;
21 import android.widget.LinearLayout;
22 import android.widget ScrollView;
23 import android.widget.TextView;
24
25 import com.example.SAPLM.Application;
26 import com.example.SAPLM.R;
27
28 public class DeveloperConsoleActivity extends AppCompatActivity {
29     private static Toolbar toolbar;
30     private static ImageButton enterButton;
31     private static ScrollView scrollViewTerminal;
32     private static EditText editTextTerminal;
33     private static LinearLayout terminalLinearLayout;
34     private static TextView mainTextView;
35
36     private static final String mainText_KEY = "MAIN_TEXT_KEY";
37     private static final String ongoingText_KEY = "ONGOING_TEXT_KEY";
38
39     @Override
40     protected void onCreate(Bundle savedInstanceState) {
41         super.onCreate(savedInstanceState);
42         setContentView(R.layout.developer_console_layout);
43         Application.setActivity(this);
44         Application.setContext(this);
45
46         toolbar = this.findViewById(R.id.toolbar);
47         enterButton = this.findViewById(R.id.enterButton);
48         scrollViewTerminal = this.findViewById(R.id.scrollViewTerminal);
49         editTextTerminal = this.findViewById(R.id.editTextTerminal);
50         terminalLinearLayout = this.findViewById(R.id.terminalLinearLayout);
51
52
53
54         setSupportActionBar(toolbar);
55
56
57         // add back arrow to toolbar          179
58         if (getSupportActionBar() != null){
59             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
60             getSupportActionBar().setDisplayShowHomeEnabled(true);
61
62             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
abc_ic_ab_back_material);
```

Archivo - DeveloperConsoleActivity.java

```
63         upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP
64     );
65     getSupportActionBar().setHomeAsUpIndicator(upArrow);
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);
```

180

Archivo - DeveloperConsoleActivity.java

```
119
120      if (Looper.myLooper() == Looper.getMainLooper()){
121          mainTextView.setText(Html.fromHtml(newString, Html.FROM_HTML_MODE_LEGACY),
122          TextView.BufferType.SPANNABLE);
123      } else {
124          Application.getActivity().runOnUiThread(new Runnable() {
125              @Override
126              public void run() {
127                  mainTextView.setText(Html.fromHtml(Application.persistentDataSave.
128                  getString(mainText_KEY), Html.FROM_HTML_MODE_LEGACY), TextView.BufferType.SPANNABLE);
129                  scrollViewTerminal.fullScroll(View.FOCUS_DOWN);
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.
139      toHtml(spannedConverted, Html.FROM_HTML_MODE_LEGACY);
140      Application.persistentDataSave.putString(mainText_KEY,newString);
141
142      if (mainTextView!=null) {
143          if (Looper.myLooper() == Looper.getMainLooper()) {
144              mainTextView.setText(Html.fromHtml(newString, Html.FROM_HTML_MODE_LEGACY),
145              TextView.BufferType.SPANNABLE);
146              scrollViewTerminal.fullScroll(View.FOCUS_DOWN);
147          } else {
148              Application.getActivity().runOnUiThread(new Runnable() {
149                  @Override
150                  public void run() {
151                      mainTextView.setText(Html.fromHtml(Application.persistentDataSave.
152                      getString(mainText_KEY), Html.FROM_HTML_MODE_LEGACY), TextView.BufferType.SPANNABLE);
153                      scrollViewTerminal.fullScroll(View.FOCUS_DOWN);
154                  }
155              });
156
157
158  public static void clearTerminalText(){
159      Application.persistentDataSave.putString(mainText_KEY,"");
160
161      if (Looper.myLooper() == Looper.getMainLooper()){
162          mainTextView.setText("");
163          scrollViewTerminal.fullScroll(View.FOCUS_DOWN);
164      } else {
165          Application.getActivity().runOnUiThread(new Runnable() {
166              @Override
167              public void run() {
168                  mainTextView.setText("");
169                  scrollViewTerminal.fullScrol181l(View.FOCUS_DOWN);
170              }
171          });
172      }
173  }
174
175
176 }
```

Archivo - DeveloperConsoleActivity.java

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

182

Archivo - BluetoothSettingsActivity.java

```
1 package com.example.SAPLM.settingsActivities;
2
3 import android.graphics.PorterDuff;
4 import android.graphics.drawable.Drawable;
5 import android.os.Bundle;
6 import androidx.core.content.ContextCompat;
7 import androidx.appcompat.app.AppCompatActivity;
8 import androidx.appcompat.widget.Toolbar;
9 import android.view.MenuItem;
10
11 import com.example.SAPLM.Application;
12 import com.example.SAPLM.R;
13
14 public class BluetoothSettingsActivity extends AppCompatActivity {
15     private static Toolbar toolbar;
16
17     @Override
18     protected void onCreate(Bundle savedInstanceState) {
19         super.onCreate(savedInstanceState);
20         setContentView(R.layout.bluetooth_settings_layout);
21         Application.setActivity(this);
22         Application.setContext(this);
23
24         toolbar = this.findViewById(R.id.toolbar);
25
26         setSupportActionBar(toolbar);
27
28         // add back arrow to toolbar
29         if (getSupportActionBar() != null){
30             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
31             getSupportActionBar().setDisplayShowHomeEnabled(true);
32
33             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
34             abc_ic_ab_back_material);
35             upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
36             ;
37             getSupportActionBar().setHomeAsUpIndicator(upArrow);
38         }
39     }
40
41     @Override
42     public boolean onOptionsItemSelected(MenuItem item) {
43         if (item.getItemId() == android.R.id.home) // Press Back Icon
44         {
45             finish();
46         }
47     }
48 }
49
```

Archivo - InternetConectivityActivity.java

```
1 package com.example.SAPLM.settingsActivities;
2
3 import android.graphics.PorterDuff;
4 import android.graphics.drawable.Drawable;
5 import android.os.Bundle;
6 import androidx.core.content.ContextCompat;
7 import androidx.appcompat.app.AppCompatActivity;
8 import androidx.appcompat.widget.Toolbar;
9 import android.view.MenuItem;
10
11 import com.example.SAPLM.Application;
12 import com.example.SAPLM.R;
13
14 public class InternetConectivityActivity extends AppCompatActivity {
15     private static Toolbar toolbar;
16
17     @Override
18     protected void onCreate(Bundle savedInstanceState) {
19         super.onCreate(savedInstanceState);
20         setContentView(R.layout.internet_conectivity_layout);
21         Application.setActivity(this);
22         Application.setContext(this);
23
24         toolbar = this.findViewById(R.id.toolbar);
25
26         setSupportActionBar(toolbar);
27
28         // add back arrow to toolbar
29         if (getSupportActionBar() != null){
30             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
31             getSupportActionBar().setDisplayShowHomeEnabled(true);
32
33             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
34             abc_ic_ab_back_material);
35             upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
36             ;
37             getSupportActionBar().setHomeAsUpIndicator(upArrow);
38         }
39     }
40
41     @Override
42     public boolean onOptionsItemSelected(MenuItem item) {
43         if (item.getItemId() == android.R.id.home) // Press Back Icon
44         {
45             finish();
46         }
47     }
48 }
49
```

Archivo - ContactsCommunicationActivity.java

```
1 package com.example.SAPLM.settingsActivities;
2
3 import android.graphics.PorterDuff;
4 import android.graphics.drawable.Drawable;
5 import android.os.Bundle;
6 import androidx.core.content.ContextCompat;
7 import androidx.appcompat.app.AppCompatActivity;
8 import androidx.appcompat.widget.Toolbar;
9 import android.view.MenuItem;
10
11 import com.example.SAPLM.Application;
12 import com.example.SAPLM.R;
13
14 public class ContactsCommunicationActivity extends AppCompatActivity {
15     private static Toolbar toolbar;
16
17     @Override
18     protected void onCreate(Bundle savedInstanceState) {
19         super.onCreate(savedInstanceState);
20         setContentView(R.layout.contacts_communication_layout);
21         Application.setActivity(this);
22         Application.setContext(this);
23
24         toolbar = this.findViewById(R.id.toolbar);
25
26         setSupportActionBar(toolbar);
27
28         // add back arrow to toolbar
29         if (getSupportActionBar() != null){
30             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
31             getSupportActionBar().setDisplayShowHomeEnabled(true);
32
33             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
34             abc_ic_ab_back_material);
35             upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
36             ;
37             getSupportActionBar().setHomeAsUpIndicator(upArrow);
38         }
39     }
40
41     @Override
42     public boolean onOptionsItemSelected(MenuItem item) {
43         if (item.getItemId() == android.R.id.home) // Press Back Icon
44         {
45             finish();
46         }
47     }
48 }
49
```

```

1 package com.example.SAPLM.settingsActivities;
2
3 import android.graphics.PorterDuff;
4 import android.graphics.drawable.Drawable;
5 import android.os.Bundle;
6 import androidx.core.content.ContextCompat;
7 import androidx.appcompat.app.AppCompatActivity;
8 import androidx.appcompat.widget.Toolbar;
9 import android.view.MenuItem;
10
11 import com.example.SAPLM.Application;
12 import com.example.SAPLM.R;
13
14 public class AppearanceMiscellaneousActivity extends AppCompatActivity {
15
16     private static Toolbar toolbar;
17
18     @Override
19     protected void onCreate(Bundle savedInstanceState) {
20         super.onCreate(savedInstanceState);
21         setContentView(R.layout.appearance_miscellaneous_layout);
22         Application.setActivity(this);
23         Application.setContext(this);
24
25         toolbar = this.findViewById(R.id.toolbar);
26
27
28         setSupportActionBar(toolbar);
29
30         // add back arrow to toolbar
31         if (getSupportActionBar() != null){
32             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
33             getSupportActionBar().setDisplayShowHomeEnabled(true);
34
35             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
36 abc_ic_ab_back_material);
37             upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
38 ;
39             getSupportActionBar().setHomeAsUpIndicator(upArrow);
40         }
41     }
42
43     @Override
44     public boolean onOptionsItemSelected(MenuItem item) {
45         if (item.getItemId() == android.R.id.home) // Press Back Icon
46         {
47             finish();
48         }
49
50         return super.onOptionsItemSelected(item);
51     }
52 }
```

Archivo - VoiceRecognitionSettingsActivity.java

```
1 package com.example.SAPLM.settingsActivities;
2
3 import android.content.DialogInterface;
4 import android.graphics.PorterDuff;
5 import android.graphics.drawable.Drawable;
6 import android.os.Bundle;
7 import android.speech.tts.Voice;
8 import androidx.core.content.ContextCompat;
9 import androidx.appcompat.app.AlertDialog;
10 import androidx.appcompat.app.AppCompatActivity;
11 import androidx.appcompat.widget.Toolbar;
12 import android.view.MenuItem;
13 import android.view.View;
14 import android.widget.Button;
15 import android.widget.EditText;
16 import android.widget.ProgressBar;
17
18 import com.example.SAPLM.Application;
19 import com.example.SAPLM.R;
20 import com.example.SAPLM.TextToSpeechHandler;
21 import com.github.ybq.android.spinkit.sprite.Sprite;
22 import com.github.ybq.android.spinkit.style.FadingCircle;
23
24 import java.text.SimpleDateFormat;
25 import java.util.ArrayList;
26 import java.util.Calendar;
27 import java.util.Date;
28 import java.util.Locale;
29
30 public class VoiceRecognitionSettingsActivity extends AppCompatActivity {
31     private static Toolbar toolbar;
32
33     private Button speech_test_btn;
34     private Button set_voice_btn;
35     private EditText phrase_editText;
36     private static ProgressBar progressBar;
37
38     @Override
39     protected void onCreate(Bundle savedInstanceState) {
40         super.onCreate(savedInstanceState);
41         setContentView(R.layout.voice_recognition_settings_layout);
42         Application.setActivity(this);
43         Application.setContext(this);
44
45         toolbar = this.findViewById(R.id.toolbar);
46         speech_test_btn = this.findViewById(R.id.speech_test_btn);
47         set_voice_btn = this.findViewById(R.id.set_voice_btn);
48         phrase_editText = this.findViewById(R.id.phrase_editText);
49
50         setSupportActionBar(toolbar);
51
52         // add back arrow to toolbar
53         if (getSupportActionBar() != null){
54             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
55             getSupportActionBar().setDisplayShowHomeEnabled(true);
56
57             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
58             abc_ic_ab_back_material);
59             upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
59             ;
59             getSupportActionBar().setHomeAsUpIndicator(upArrow);
60         }
61 }
```

## Archivo - VoiceRecognitionSettingsActivity.java

```
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-MMM-yy", 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"
94         + onlyTimeString);
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
114     DialogInterface.OnClickListener() {
115         @Override
116         public void onClick(DialogInterface dialog, int which) {
117             TextToSpeechHandler.setVoice(voiceArray[which]);
118         }
119     });
120     builder.show();
121 }
```

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

```

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, (
7             byte) 0x12, (byte) 0x15, (byte) 0x38, (byte) 0x3f, (byte) 0x36, (byte) 0x31, (byte) 0x24, (
8                 byte) 0x23, (byte) 0x2a, (byte) 0x2d,
9                 (byte) 0x70, (byte) 0x77, (byte) 0x7e, (byte) 0x79, (byte) 0x6c, (byte) 0x6b, (
10                byte) 0x62, (byte) 0x65, (byte) 0x48, (byte) 0x4f, (byte) 0x46, (byte) 0x41, (byte) 0x54, (
11                    byte) 0x53, (byte) 0x5a, (byte) 0x5d,
12                    (byte) 0xe0, (byte) 0xe7, (byte) 0xee, (byte) 0xe9, (byte) 0xfc, (byte) 0xfb, (
13                        byte) 0xf2, (byte) 0xf5, (byte) 0xd8, (byte) 0xdf, (byte) 0xd6, (byte) 0xd1, (byte) 0xc4, (
14                            byte) 0xc3, (byte) 0xca, (byte) 0xcd,
15                            (byte) 0x90, (byte) 0x97, (byte) 0x9e, (byte) 0x99, (byte) 0x8c, (byte) 0x8b, (
16                                byte) 0x82, (byte) 0x85, (byte) 0xa8, (byte) 0xaf, (byte) 0xa6, (byte) 0xa1, (byte) 0xb4, (
17                                    byte) 0xb3, (byte) 0xba, (byte) 0xbd,
18                                    (byte) 0xc7, (byte) 0xc0, (byte) 0xc9, (byte) 0xce, (byte) 0xdb, (byte) 0xdc, (
19                                        byte) 0xd5, (byte) 0xd2, (byte) 0xff, (byte) 0xf8, (byte) 0xf1, (byte) 0xf6, (byte) 0xe3, (
20                                            byte) 0xe4, (byte) 0xed, (byte) 0xea,
21                                            (byte) 0xb7, (byte) 0xb0, (byte) 0xb9, (byte) 0xbe, (byte) 0xab, (byte) 0xac, (
22                                                byte) 0xa5, (byte) 0xa2, (byte) 0x8f, (byte) 0x88, (byte) 0x81, (byte) 0x86, (byte) 0x93, (
23                                                    byte) 0x94, (byte) 0x9d, (byte) 0x9a,
24                                                    (byte) 0x27, (byte) 0x20, (byte) 0x29, (byte) 0x2e, (byte) 0x3b, (byte) 0x3c, (
25                                                        byte) 0x35, (byte) 0x32, (byte) 0x1f, (byte) 0x18, (byte) 0x11, (byte) 0x16, (byte) 0x03, (
26                                                            byte) 0x04, (byte) 0xd, (byte) 0xa,
27                                                            (byte) 0x57, (byte) 0x50, (byte) 0x59, (byte) 0x5e, (byte) 0x4b, (byte) 0x4c, (
28                                                                byte) 0x45, (byte) 0x42, (byte) 0x6f, (byte) 0x68, (byte) 0x61, (byte) 0x66, (byte) 0x73, (
29                                                                    byte) 0x74, (byte) 0x7d, (byte) 0x7a,
30                                                                    (byte) 0x89, (byte) 0x8e, (byte) 0x87, (byte) 0x80, (byte) 0x95, (byte) 0x92, (
31                                                                        byte) 0x9b, (byte) 0x9c, (byte) 0xb1, (byte) 0xb6, (byte) 0xbf, (byte) 0xb8, (byte) 0xad, (
32                                                                            byte) 0xaa, (byte) 0xa3, (byte) 0xa4,
33                                                                            (byte) 0xf9, (byte) 0xfe, (byte) 0xf7, (byte) 0xf0, (byte) 0xe5, (byte) 0xe2, (
34                                                                                byte) 0xeb, (byte) 0xec, (byte) 0xc1, (byte) 0xc6, (byte) 0xcf, (byte) 0xc8, (byte) 0xdd, (
35                                                                                    byte) 0xda, (byte) 0xd3, (byte) 0xd4,
36                                                                                    (byte) 0x69, (byte) 0x6e, (byte) 0x67, (byte) 0x60, (byte) 0x75, (byte) 0x72, (
37                                                                                        byte) 0x7b, (byte) 0x7c, (byte) 0x51, (byte) 0x56, (byte) 0x5f, (byte) 0x58, (byte) 0x4d, (
38                                                                                            byte) 0x4a, (byte) 0x43, (byte) 0x44,
39                                                                                            (byte) 0x19, (byte) 0x1e, (byte) 0x17, (byte) 0x10, (byte) 0x05, (byte) 0x02, (
40                                                                                                byte) 0x0b, (byte) 0x0c, (byte) 0x21, (byte) 0x26, (byte) 0x2f, (byte) 0x28, (byte) 0x3d, (
41                                                                                                    byte) 0x3a, (byte) 0x33, (byte) 0x34,
42                                    (byte) 0x4e, (byte) 0x49, (byte) 0x40, (byte) 0x47, (byte) 0x52, (byte) 0x55, (
43                                         byte) 0x5c, (byte) 0x5b, (byte) 0x76, (byte) 0x71, (byte) 0x78, (byte) 0x7f, (byte) 0x6a, (
44                                         byte) 0x6d, (byte) 0x64, (byte) 0x63,
45                                         (byte) 0x3e, (byte) 0x39, (byte) 0x30, (byte) 0x37, (byte) 0x22, (byte) 0x25, (
46                                         byte) 0x2c, (byte) 0x2b, (byte) 0x06, (byte) 0x01, (byte) 0x08, (byte) 0x0f, (byte) 0x1a, (
47                                         byte) 0x1d, (byte) 0x14, (byte) 0x13,
48                                         (byte) 0xae, (byte) 0xa9, (byte) 0xa0, (byte) 0xa7, (byte) 0xb2, (byte) 0xb5, (
49                                         byte) 0xbc, (byte) 0xbb, (byte) 0x96, (byte) 0x91, (byte) 0x98, (byte) 0x9f, (byte) 0x8a, (
50                                         byte) 0x8d, (byte) 0x84, (byte) 0x83,
51                                         (byte) 0xde, (byte) 0xd9, (byte) 0xd0, (byte) 0xd7, (byte) 0xc2, (byte) 0xc5, (
52                                         byte) 0xcc, (byte) 0xcb, (byte) 0xe6, (byte) 0xe1, (byte) 0xe8, (byte) 0xef, (byte) 0xfa, (
53                                         byte) 0xfd, (byte) 0xf4, (byte) 0xf3
54     };
55
56     public static byte crc_CCITT(byte[] data, int data_len)
57     {
58         byte crc = 0;
59         byte[] d = data;
60         int dataIndex = 0;
61         byte tbl_idx = 0;
62
63         while (data_len!=0) {
64
65             byte val = d[dataIndex];
66             dataIndex++;
67
68             byte temp = crc ^ val;
69             crc = crc_table[tbl_idx];
70             tbl_idx = (tbl_idx + 1) & 0x0f;
71
72             if (temp >= 0x80) {
73                 crc = crc ^ 0x01;
74             }
75
76         }
77
78         return crc;
79     }
80
81 }
```

Archivo - CRC8\_CCITT.java

```
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

```
1 package com.example.SAPLM.bluetoothActivities;
2
3 import com.example.SAPLM.R;
4
5 public class AvailableDevice {
6
7     public enum DeviceState { DEFAULT, DISCONNECTED, CONNECTED, LOADING}
8     private String deviceName;
9     private DeviceState currentState;
10    private int deviceStateIconID;
11
12    public AvailableDevice() {
13    }
14
15    public AvailableDevice(String deviceName, DeviceState defaultState) {
16        this.deviceName = deviceName;
17        this.currentState = defaultState;
18    }
19
20    public void setCurrentState(DeviceState currentState) {
21        this.currentState = currentState;
22    }
23
24    public DeviceState getCurrentState() {
25        return currentState;
26    }
27
28    public void setDeviceName(String deviceName) {
29        this.deviceName = deviceName;
30    }
31
32    public String getDeviceName() {
33        return deviceName;
34    }
35
36    public int getDeviceStateIconID() {
37        switch (currentState){
38            case DEFAULT:
39                return 0;
40            case CONNECTED:
41                return R.drawable.ic_done_icon;
42            case LOADING:
43                return R.drawable.loading_animation;
44            case DISCONNECTED:
45                return R.drawable.ic_close_icon;
46        }
47        return 0;
48    }
49
50    public int getTextAppearance(){
51        switch (currentState){
52            case DEFAULT:
53                return R.style.btDeviceListAppearance;
54            case CONNECTED:
55                return R.style.btDeviceListSelectedAppearance;
56            case LOADING:
57                return R.style.btDeviceListAppearance; 192
58            case DISCONNECTED:
59                return R.style.btDeviceListAppearance;
60        }
61        return R.style.btDeviceListAppearance;
62    }
63 }
```

Archivo - BluetoothActivity.java

```
1 package com.example.SAPLM.bluetoothActivities;
2
3 import android.content.Context;
4 import android.content.Intent;
5 import android.graphics.Color;
6 import android.graphics.PorterDuff;
7 import android.graphics.drawable.Drawable;
8 import android.os.Bundle;
9 import android.os.Looper;
10 import androidx.core.content.ContextCompat;
11 import androidx.appcompat.app.AppCompatActivity;
12 import androidx.recyclerview.widget.DefaultItemAnimator;
13 import androidx.recyclerview.widget.DividerItemDecoration;
14 import androidx.recyclerview.widget.LinearLayoutManager;
15 import androidx.recyclerview.widget.RecyclerView;
16 import androidx.appcompat.widget.Toolbar;
17 import android.view.MenuItem;
18 import android.view.View;
19 import android.widget.ImageView;
20 import android.widget.ProgressBar;
21 import android.widget.TextView;
22
23 import com.example.SAPLM.Application;
24 import com.example.SAPLM.R;
25 import com.example.SAPLM.settingsActivities.BluetoothSettingsActivity;
26 import com.example.SAPLM.settingsActivities.RecyclerTouchListener;
27 import com.github.ybq.android.spinkit.sprite.Sprite;
28 import com.github.ybq.android.spinkit.style.Circle;
29
30 import java.util.ArrayList;
31 import java.util.List;
32
33 public class BluetoothActivity extends AppCompatActivity {
34     private static Toolbar toolbar;
35
36
37     public static List<AvailableDevice> deviceList = new ArrayList<>();
38     private RecyclerView recyclerView;
39     public static DeviceListAdapter mAdapter;
40
41     public static ImageView imgView_bluetoothStatusIcon;
42     public static TextView textView_bluetoothState;
43     public static ProgressBar loading_progressBar;
44
45     public static Boolean guiInitialized = false;
46
47
48     @Override
49     protected void onCreate(Bundle savedInstanceState) {
50         super.onCreate(savedInstanceState);
51         setContentView(R.layout.bluetooth_layout);
52         Application.setActivity(this);
53         Application.setContext(this);
54
55         toolbar = this.findViewById(R.id.toolbar);
56         textView_bluetoothState = this.findViewById(R.id.textView_btStatus);
57         imgView_bluetoothStatusIcon = this.findViewById(R.id.imgView_btStatusIcon);
58         loading_progressBar = this.findViewById(R.id.loading_progressBar);
59
60         Sprite fadingCircle = new Circle();
61         fadingCircle.setColor(Color.WHITE);
62         loading_progressBar.setIndeterminateDrawable(fadingCircle);
63 }
```

Archivo - BluetoothActivity.java

```
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
```

## Archivo - BluetoothActivity.java

```
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,
151     BLUETOOTH_DISABLED, WAITING, BT_TURNING_OFF, BT_TURNING_ON, UNDEFINED}
152     public static bluetoothConnectionState currentBluetoothState = bluetoothConnectionState
153     .UNDEFINED;
154
155     public static void setBluetoothStateGui(bluetoothConnectionState state){
156         if (guiInitialized) {
157             currentBluetoothState = state;
158
159             if (Looper.myLooper() == Looper.getMainLooper()) {
160                 switch (currentBluetoothState) {
161                     case SYSTEM_CONNECTED:
162                         textView_bluetoothState.setText(R.string.system_connected_message);
163                         imgView_bluetoothStatusIcon.setImageResource(R.drawable.
164                         ic_done_icon);
165                         imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
166                         loading_progressBar.setVisibility(View.GONE);
167                         break;
168                     case BLUETOOTH_DISABLED:
169                         textView_bluetoothState.setText(R.string.bluetooth_disabled_message
170                     );
171                         imgView_bluetoothStatusIcon.setImageResource(R.drawable.
172                         ic_bluetooth_disabled);
173                         imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
174                         loading_progressBar.setVisibility(View.GONE);
175                         break;
176                     case BT_TURNING_ON:    195
177                         textView_bluetoothState.setText(R.string.
178                         bluetooth_initializing_adapater);
179                         imgView_bluetoothStatusIcon.setVisibility(View.GONE);
180                         loading_progressBar.setVisibility(View.VISIBLE);
181                         break;
182                     case BT_TURNING_OFF:
```

```

177                     textView_bluetoothState.setText(R.string.bluetooth_closing_adapter)
178                 ;
179                 imgView_bluetoothStatusIcon.setVisibility(View.GONE);
180                 loading_progressBar.setVisibility(View.VISIBLE);
181                 break;
182             case SYSTEM_DISCONNECTED:
183                 textView_bluetoothState.setText(R.string.
184                     system_disconnected_message);
185                 imgView_bluetoothStatusIcon.setImageResource(R.drawable.
186                     ic_close_icon);
187                 imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
188                 loading_progressBar.setVisibility(View.GONE);
189                 break;
190             case WAITING:
191                 textView_bluetoothState.setText(R.string.
192                     bluetooth_attempting_connection_message);
193                 imgView_bluetoothStatusIcon.setVisibility(View.GONE);
194                 loading_progressBar.setVisibility(View.VISIBLE);
195                 break;
196             } else {
197                 Application.getActivity().runOnUiThread(new Runnable() {
198                     @Override
199                     public void run() {
200                         switch (currentBluetoothState) {
201                             case SYSTEM_CONNECTED:
202                                 textView_bluetoothState.setText(R.string.
203                                     system_connected_message);
204                                 imgView_bluetoothStatusIcon.setImageResource(R.drawable.
205                                     ic_done_icon);
206                                 imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
207                                 loading_progressBar.setVisibility(View.GONE);
208                                 break;
209                             case BLUETOOTH_DISABLED:
210                                 textView_bluetoothState.setText(R.string.
211                                     bluetooth_disabled_message);
212                                 imgView_bluetoothStatusIcon.setImageResource(R.drawable.
213                                     ic_bluetooth_disabled);
214                                 imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
215                                 loading_progressBar.setVisibility(View.GONE);
216                                 break;
217                             case BT_TURNING_ON:
218                                 textView_bluetoothState.setText(R.string.
219                                     bluetooth_initializing_adapater);
220                                 imgView_bluetoothStatusIcon.setVisibility(View.GONE);
221                                 loading_progressBar.setVisibility(View.VISIBLE);
222                                 break;
223                             case BT_TURNING_OFF:
224                                 textView_bluetoothState.setText(R.string.
225                                     bluetooth_closing_adapter);
226                                 imgView_bluetoothStatusIcon.setVisibility(View.GONE);
227                                 loading_progressBar.setVisibility(View.VISIBLE);
228                                 break;
229                             case SYSTEM_DISCONNECTED:
230                                 textView_bluetoothState.setText(R.string.
231                                     system_disconnected_message);
232                                 imgView_bluetoothStatusIcon.setImageResource(R.drawable.
233                                     ic_close_icon);
234                                 imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
235                                 loading_progressBar.setVisibility(View.GONE);
236                                 break;
237                             case WAITING:
238                                 textView_bluetoothState.setText(R.string.
239                                     bluetooth_attempting_connection_message);
240                                 imgView_bluetoothStatusIcon.setVisibility(View.GONE);
241                                 loading_progressBar.setVisibility(View.VISIBLE);
242                                 break;
243                         }
244                     }
245                 });
246             }
247         }
248     }
249 }

```

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

```
1 package com.example.SAPLM.bluetoothActivities;
2
3 import android.graphics.Color;
4 import android.graphics.PorterDuff;
5 import android.graphics.drawable.ColorDrawable;
6 import android.graphics.drawable.Drawable;
7 import androidx.recyclerview.widget.RecyclerView;
8 import android.view.LayoutInflater;
9 import android.view.View;
10 import android.view.ViewGroup;
11 import android.widget.ImageView;
12 import android.widget.TextView;
13
14 import com.example.SAPLM.Application;
15 import com.example.SAPLM.R;
16
17 import java.util.List;
18
19 public class DeviceListAdapter extends RecyclerView.Adapter<DeviceListAdapter.ViewHolder> {
20
21     private List<AvailableDevice> deviceList;
22
23     public class ViewHolder extends RecyclerView.ViewHolder {
24         public TextView textView_device_name;
25         public ImageView imgView_device_state_icon;
26
27         public viewHolder(View view) {
28             super(view);
29             textView_device_name = (TextView) view.findViewById(R.id.
30             textView_bluetooth_device_name);
31             imgView_device_state_icon = (ImageView) view.findViewById(R.id.
32             imageView_bluetooth_device_state_icon);
33         }
34     }
35
36     public DeviceListAdapter(List<AvailableDevice> deviceList) {
37         this.deviceList = deviceList;
38     }
39
40     @Override
41     public viewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
42         View itemView = LayoutInflater.from(parent.getContext())
43             .inflate(R.layout.bluetooth_list_layout_style, parent, false);
44
45         return new viewHolder(itemView);
46     }
47
48     @Override
49     public void onBindViewHolder(viewHolder holder, int position) {
50         AvailableDevice device = deviceList.get(position);
51         holder.textView_device_name.setText(device.getDeviceName());
52         holder.textView_device_name.setTextAppearance(device.getTextAppearance());
53
54         int iconID = device.getDeviceStateIconID();
55
56         if (iconID!=0) { 198
57             Drawable icon = Application.getContext().getDrawable(iconID);
58             icon.setColorFilter(Application.getContext().getColor(R.color.colorPrimary),
59             PorterDuff.Mode.SRC_ATOP);
59             holder.imgView_device_state_icon.setImageDrawable(icon);
60         } else {
60             holder.imgView_device_state_icon.clearColorFilter();
61         }
62     }
63 }
```

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
```

```

1 package com.example.SAPLM.bluetoothActivities;
2
3 import android.bluetooth.BluetoothAdapter;
4 import android.bluetooth.BluetoothDevice;
5 import android.bluetooth.BluetoothServerSocket;
6 import android.bluetooth.BluetoothSocket;
7 import android.content.BroadcastReceiver;
8 import android.content.Context;
9 import android.content.Intent;
10 import android.widget.TabHost;
11
12 import com.example.SAPLM.Application;
13 import com.example.SAPLM.settingsActivities.DeveloperConsoleActivity;
14 import com.example.SAPLM.settingsActivities.TerminalManager;
15
16
17 import java.io.InputStream;
18 import java.io.OutputStream;
19 import java.nio.charset.StandardCharsets;
20 import java.util.ArrayList;
21 import java.util.List;
22 import java.util.Set;
23 import java.util.UUID;
24 import java.util.concurrent.Executor;
25 import java.util.concurrent.Executors;
26 import java.util.concurrent.ScheduledExecutorService;
27 import java.util.concurrent.ScheduledFuture;
28
29
30 public class BluetoothConnection extends BluetoothActivity {
31
32     static ScheduledExecutorService bluetoothThreadPool= Executors.newScheduledThreadPool(1)
33     ;
34     static ScheduledFuture<?> bluetoothComThread;
35     public static OutputStream outputStream;
36     public static InputStream inputStream;
37     public static BluetoothAdapter btAdapter = BluetoothAdapter.getDefaultAdapter();
38     public static BluetoothDevice targetDevice;
39     public static BluetoothDevice[] bluetoothDevicesArray;
40     public static List<BluetoothDevice> bluetoothDevicesList;
41     public static BluetoothSocket btSocket;
42     public static BluetoothServerSocket btServerSocket;
43     public static UUID btUUID = UUID.fromString("00001101-0000-1000-8000-00805f9b34fb");
44     public static String btString = "HC-05";
45     public static Boolean bluetoothConnected = false;
46
47     private static Executor bluetoothConnectorExecutor = Executors.newSingleThreadExecutor()
48     ;
49     private static Boolean isListAvailableBluetoothDevicesRunning = false;
50
51     public static void listAvailableBluetoothDevices() {
52         if ((!isListAvailableBluetoothDevicesRunning)) {
53             try {
54                 BluetoothActivity.setBluetoothStateGui(blueoothConnectionState.WAITING);
55                 isListAvailableBluetoothDevicesRunning = true;
56                 bluetoothConnectorExecutor.execute(new Runnable() {
57                     @Override
58                     public void run() {
59                         if (btAdapter != null) {
60                             if (btAdapter.isEnabled()) {
61                                 Set<BluetoothDevice> bondedDevices = btAdapter.
62                                     getBondedDevices();
63                             }
64                         }
65                     }
66                 });
67             } catch (Exception e) {
68                 e.printStackTrace();
69             }
70         }
71     }
72
73     public static void connectToTargetDevice() {
74         if (targetDevice != null) {
75             try {
76                 btSocket = targetDevice.createRfcommSocketToServiceRecord(btUUID);
77                 btAdapter.connect(btSocket, targetDevice);
78             } catch (IOException e) {
79                 e.printStackTrace();
80             }
81         }
82     }
83
84     public static void disconnectFromTargetDevice() {
85         if (btSocket != null) {
86             try {
87                 btSocket.close();
88             } catch (IOException e) {
89                 e.printStackTrace();
90             }
91         }
92     }
93
94     public static void sendCommand(String command) {
95         if (outputStream != null) {
96             try {
97                 outputStream.write(command.getBytes(StandardCharsets.UTF_8));
98             } catch (IOException e) {
99                 e.printStackTrace();
100            }
101        }
102    }
103
104    public static void receiveData() {
105        if (inputStream != null) {
106            try {
107                byte[] buffer = new byte[1024];
108                int bytes = inputStream.read(buffer);
109                String data = new String(buffer, 0, bytes);
110                Log.d("BT", "Received data: " + data);
111            } catch (IOException e) {
112                e.printStackTrace();
113            }
114        }
115    }
116
117    public static void startComThread() {
118        bluetoothComThread = bluetoothThreadPool.scheduleAtFixedRate(new Runnable() {
119            @Override
120            public void run() {
121                receiveData();
122            }
123        }, 0, 1, TimeUnit.SECONDS);
124    }
125
126    public static void stopComThread() {
127        bluetoothComThread.cancel(true);
128    }
129
130    public static void setBluetoothConnected(Boolean connected) {
131        bluetoothConnected = connected;
132    }
133
134    public static Boolean getBluetoothConnected() {
135        return bluetoothConnected;
136    }
137
138    public static void setBluetoothStateGui(BluetoothConnectionState state) {
139        BluetoothActivity.setBluetoothStateGui(state);
140    }
141
142    public static BluetoothConnectionState getBluetoothConnectionState() {
143        return blueoothConnectionState;
144    }
145
146    public static void setBlueoothConnectionState(BluetoothConnectionState state) {
147        blueoothConnectionState = state;
148    }
149
150    public static void setBlueoothConnectionState(BluetoothConnectionState state, String message) {
151        blueoothConnectionState = state;
152        Log.d("BT", message);
153    }
154
155    public static void setBlueoothConnectionState(BluetoothConnectionState state, String message, Boolean error) {
156        blueoothConnectionState = state;
157        Log.d("BT", message);
158        if (error) {
159            Log.e("BT", "Error: " + message);
160        }
161    }
162
163    public static void setBlueoothConnectionState(BluetoothConnectionState state, String message, Boolean error, Boolean warning) {
164        blueoothConnectionState = state;
165        Log.d("BT", message);
166        if (error) {
167            Log.e("BT", "Error: " + message);
168        }
169        if (warning) {
170            Log.w("BT", "Warning: " + message);
171        }
172    }
173
174    public static void setBlueoothConnectionState(BluetoothConnectionState state, String message, Boolean error, Boolean warning, Boolean info) {
175        blueoothConnectionState = state;
176        Log.d("BT", message);
177        if (error) {
178            Log.e("BT", "Error: " + message);
179        }
180        if (warning) {
181            Log.w("BT", "Warning: " + message);
182        }
183        if (info) {
184            Log.i("BT", "Info: " + message);
185        }
186    }
187
188    public static void setBlueoothConnectionState(BluetoothConnectionState state, String message, Boolean error, Boolean warning, Boolean info, Boolean debug) {
189        blueoothConnectionState = state;
190        Log.d("BT", message);
191        if (error) {
192            Log.e("BT", "Error: " + message);
193        }
194        if (warning) {
195            Log.w("BT", "Warning: " + message);
196        }
197        if (info) {
198            Log.i("BT", "Info: " + message);
199        }
200        if (debug) {
201            Log.d("BT", "Debug: " + message);
202        }
203    }
204
205    public static void setBlueoothConnectionState(BluetoothConnectionState state, String message, Boolean error, Boolean warning, Boolean info, Boolean debug, Boolean verbose) {
206        blueoothConnectionState = state;
207        Log.d("BT", message);
208        if (error) {
209            Log.e("BT", "Error: " + message);
210        }
211        if (warning) {
212            Log.w("BT", "Warning: " + message);
213        }
214        if (info) {
215            Log.i("BT", "Info: " + message);
216        }
217        if (debug) {
218            Log.d("BT", "Debug: " + message);
219        }
220        if (verbose) {
221            Log.v("BT", "Verbose: " + message);
222        }
223    }
224
225    public static void setBlueoothConnectionState(BluetoothConnectionState state, String message, Boolean error, Boolean warning, Boolean info, Boolean debug, Boolean verbose, Boolean trace) {
226        blueoothConnectionState = state;
227        Log.d("BT", message);
228        if (error) {
229            Log.e("BT", "Error: " + message);
230        }
231        if (warning) {
232            Log.w("BT", "Warning: " + message);
233        }
234        if (info) {
235            Log.i("BT", "Info: " + message);
236        }
237        if (debug) {
238            Log.d("BT", "Debug: " + message);
239        }
240        if (verbose) {
241            Log.v("BT", "Verbose: " + message);
242        }
243        if (trace) {
244            Log.t("BT", "Trace: " + message);
245        }
246    }
247
248    public static void setBlueoothConnectionState(BluetoothConnectionState state, String message, Boolean error, Boolean warning, Boolean info, Boolean debug, Boolean verbose, Boolean trace, Boolean log) {
249        blueoothConnectionState = state;
250        Log.d("BT", message);
251        if (error) {
252            Log.e("BT", "Error: " + message);
253        }
254        if (warning) {
255            Log.w("BT", "Warning: " + message);
256        }
257        if (info) {
258            Log.i("BT", "Info: " + message);
259        }
260        if (debug) {
261            Log.d("BT", "Debug: " + message);
262        }
263        if (verbose) {
264            Log.v("BT", "Verbose: " + message);
265        }
266        if (trace) {
267            Log.t("BT", "Trace: " + message);
268        }
269        if (log) {
270            Log.l("BT", "Log: " + message);
271        }
272    }
273
274    public static void setBlueoothConnectionState(BluetoothConnectionState state, String message, Boolean error, Boolean warning, Boolean info, Boolean debug, Boolean verbose, Boolean trace, Boolean log, Boolean logcat) {
275        blueoothConnectionState = state;
276        Log.d("BT", message);
277        if (error) {
278            Log.e("BT", "Error: " + message);
279        }
280        if (warning) {
281            Log.w("BT", "Warning: " + message);
282        }
283        if (info) {
284            Log.i("BT", "Info: " + message);
285        }
286        if (debug) {
287            Log.d("BT", "Debug: " + message);
288        }
289        if (verbose) {
290            Log.v("BT", "Verbose: " + message);
291        }
292        if (trace) {
293            Log.t("BT", "Trace: " + message);
294        }
295        if (log) {
296            Log.l("BT", "Log: " + message);
297        }
298        if (logcat) {
299            Log.c("BT", "Logcat: " + message);
300        }
301    }
302
303    public static void setBlueoothConnectionState(BluetoothConnectionState state, String message, Boolean error, Boolean warning, Boolean info, Boolean debug, Boolean verbose, Boolean trace, Boolean log, Boolean logcat, Boolean logcatRaw) {
304        blueoothConnectionState = state;
305        Log.d("BT", message);
306        if (error) {
307            Log.e("BT", "Error: " + message);
308        }
309        if (warning) {
310            Log.w("BT", "Warning: " + message);
311        }
312        if (info) {
313            Log.i("BT", "Info: " + message);
314        }
315        if (debug) {
316            Log.d("BT", "Debug: " + message);
317        }
318        if (verbose) {
319            Log.v("BT", "Verbose: " + message);
320        }
321        if (trace) {
322            Log.t("BT", "Trace: " + message);
323        }
324        if (log) {
325            Log.l("BT", "Log: " + message);
326        }
327        if (logcat) {
328            Log.c("BT", "Logcat: " + message);
329        }
330        if (logcatRaw) {
331            Log.r("BT", "LogcatRaw: " + message);
332        }
333    }
334
335    public static void setBlueoothConnectionState(BluetoothConnectionState state, String message, Boolean error, Boolean warning, Boolean info, Boolean debug, Boolean verbose, Boolean trace, Boolean log, Boolean logcat, Boolean logcatRaw, Boolean logcatRawRaw) {
336        blueoothConnectionState = state;
337        Log.d("BT", message);
338        if (error) {
339            Log.e("BT", "Error: " + message);
340        }
341        if (warning) {
342            Log.w("BT", "Warning: " + message);
343        }
344        if (info) {
345            Log.i("BT", "Info: " + message);
346        }
347        if (debug) {
348            Log.d("BT", "Debug: " + message);
349        }
350        if (verbose) {
351            Log.v("BT", "Verbose: " + message);
352        }
353        if (trace) {
354            Log.t("BT", "Trace: " + message);
355        }
356        if (log) {
357            Log.l("BT", "Log: " + message);
358        }
359        if (logcat) {
360            Log.c("BT", "Logcat: " + message);
361        }
362        if (logcatRaw) {
363            Log.r("BT", "LogcatRaw: " + message);
364        }
365        if (logcatRawRaw) {
366            Log.r("BT", "LogcatRawRaw: " + message);
367        }
368    }
369
370    public static void setBlueoothConnectionState(BluetoothConnectionState state, String message, Boolean error, Boolean warning, Boolean info, Boolean debug, Boolean verbose, Boolean trace, Boolean log, Boolean logcat, Boolean logcatRaw, Boolean logcatRawRaw, Boolean logcatRawRawRaw) {
371        blueoothConnectionState = state;
372        Log.d("BT", message);
373        if (error) {
374            Log.e("BT", "Error: " + message);
375        }
376        if (warning) {
377            Log.w("BT", "Warning: " + message);
378        }
379        if (info) {
380            Log.i("BT", "Info: " + message);
381        }
382        if (debug) {
383            Log.d("BT", "Debug: " + message);
384        }
385        if (verbose) {
386            Log.v("BT", "Verbose: " + message);
387        }
388        if (trace) {
389            Log.t("BT", "Trace: " + message);
390        }
391        if (log) {
392            Log.l("BT", "Log: " + message);
393        }
394        if (logcat) {
395            Log.c("BT", "Logcat: " + message);
396        }
397        if (logcatRaw) {
398            Log.r("BT", "LogcatRaw: " + message);
399        }
400        if (logcatRawRaw) {
401            Log.r("BT", "LogcatRawRaw: " + message);
402        }
403        if (logcatRawRawRaw) {
404            Log.r("BT", "LogcatRawRawRaw: " + message);
405        }
406    }
407
408    public static void setBlueoothConnectionState(BluetoothConnectionState state, String message, Boolean error, Boolean warning, Boolean info, Boolean debug, Boolean verbose, Boolean trace, Boolean log, Boolean logcat, Boolean logcatRaw, Boolean logcatRawRaw, Boolean logcatRawRawRaw, Boolean logcatRawRawRawRaw) {
409        blueoothConnectionState = state;
410        Log.d("BT", message);
411        if (error) {
412            Log.e("BT", "Error: " + message);
413        }
414        if (warning) {
415            Log.w("BT", "Warning: " + message);
416        }
417        if (info) {
418            Log.i("BT", "Info: " + message);
419        }
420        if (debug) {
421            Log.d("BT", "Debug: " + message);
422        }
423        if (verbose) {
424            Log.v("BT", "Verbose: " + message);
425        }
426        if (trace) {
427            Log.t("BT", "Trace: " + message);
428        }
429        if (log) {
430            Log.l("BT", "Log: " + message);
431        }
432        if (logcat) {
433            Log.c("BT", "Logcat: " + message);
434        }
435        if (logcatRaw) {
436            Log.r("BT", "LogcatRaw: " + message);
437        }
438        if (logcatRawRaw) {
439            Log.r("BT", "LogcatRawRaw: " + message);
440        }
441        if (logcatRawRawRaw) {
442            Log.r("BT", "LogcatRawRawRaw: " + message);
443        }
444        if (logcatRawRawRawRaw) {
445            Log.r("BT", "LogcatRawRawRawRaw: " + message);
446        }
447    }
448
449    public static void setBlueoothConnectionState(BluetoothConnectionState state, String message, Boolean error, Boolean warning, Boolean info, Boolean debug, Boolean verbose, Boolean trace, Boolean log, Boolean logcat, Boolean logcatRaw, Boolean logcatRawRaw, Boolean logcatRawRawRaw, Boolean logcatRawRawRawRaw, Boolean logcatRawRawRawRawRaw) {
450        blueoothConnectionState = state;
451        Log.d("BT", message);
452        if (error) {
453            Log.e("BT", "Error: " + message);
454        }
455        if (warning) {
456            Log.w("BT", "Warning: " + message);
457        }
458        if (info) {
459            Log.i("BT", "Info: " + message);
460        }
461        if (debug) {
462            Log.d("BT", "Debug: " + message);
463        }
464        if (verbose) {
465            Log.v("BT", "Verbose: " + message);
466        }
467        if (trace) {
468            Log.t("BT", "Trace: " + message);
469        }
470        if (log) {
471            Log.l("BT", "Log: " + message);
472        }
473        if (logcat) {
474            Log.c("BT", "Logcat: " + message);
475        }
476        if (logcatRaw) {
477            Log.r("BT", "LogcatRaw: " + message);
478        }
479        if (logcatRawRaw) {
480            Log.r("BT", "LogcatRawRaw: " + message);
481        }
482        if (logcatRawRawRaw) {
483            Log.r("BT", "LogcatRawRawRaw: " + message);
484        }
485        if (logcatRawRawRawRaw) {
486            Log.r("BT", "LogcatRawRawRawRaw: " + message);
487        }
488        if (logcatRawRawRawRawRaw) {
489            Log.r("BT", "LogcatRawRawRawRawRaw: " + message);
490        }
491    }
492
493    public static void setBlueoothConnectionState(BluetoothConnectionState state, String message, Boolean error, Boolean warning, Boolean info, Boolean debug, Boolean verbose, Boolean trace, Boolean log, Boolean logcat, Boolean logcatRaw, Boolean logcatRawRaw, Boolean logcatRawRawRaw, Boolean logcatRawRawRawRaw, Boolean logcatRawRawRawRawRaw, Boolean logcatRawRawRawRawRawRaw) {
494        blueoothConnectionState = state;
495        Log.d("BT", message);
496        if (error) {
497            Log.e("BT", "Error: " + message);
498        }
499        if (warning) {
500            Log.w("BT", "Warning: " + message);
501        }
502        if (info) {
503            Log.i("BT", "Info: " + message);
504        }
505        if (debug) {
506            Log.d("BT", "Debug: " + message);
507        }
508        if (verbose) {
509            Log.v("BT", "Verbose: " + message);
510        }
511        if (trace) {
512            Log.t("BT", "Trace: " + message);
513        }
514        if (log) {
515            Log.l("BT", "Log: " + message);
516        }
517        if (logcat) {
518            Log.c("BT", "Logcat: " + message);
519        }
520        if (logcatRaw) {
521            Log.r("BT", "LogcatRaw: " + message);
522        }
523        if (logcatRawRaw) {
524            Log.r("BT", "LogcatRawRaw: " + message);
525        }
526        if (logcatRawRawRaw) {
527            Log.r("BT", "LogcatRawRawRaw: " + message);
528        }
529        if (logcatRawRawRawRaw) {
530            Log.r("BT", "LogcatRawRawRawRaw: " + message);
531        }
532        if (logcatRawRawRawRawRaw) {
533            Log.r("BT", "LogcatRawRawRawRawRaw: " + message);
534        }
535        if (logcatRawRawRawRawRawRaw) {
536            Log.r("BT", "LogcatRawRawRawRawRawRaw: " + message);
537        }
538    }
539
540    public static void setBlueoothConnectionState(BluetoothConnectionState state, String message, Boolean error, Boolean warning, Boolean info, Boolean debug, Boolean verbose, Boolean trace, Boolean log, Boolean logcat, Boolean logcatRaw, Boolean logcatRawRaw, Boolean logcatRawRawRaw, Boolean logcatRawRawRawRaw, Boolean logcatRawRawRawRawRaw, Boolean logcatRawRawRawRawRawRaw) {
541        blueoothConnectionState = state;
542        Log.d("BT", message);
543        if (error) {
544            Log.e("BT", "Error: " + message);
545        }
546        if (warning) {
547            Log.w("BT", "Warning: " + message);
548        }
549        if (info) {
550            Log.i("BT", "Info: " + message);
551        }
552        if (debug) {
553            Log.d("BT", "Debug: " + message);
554        }
555        if (verbose) {
556            Log.v("BT", "Verbose: " + message);
557        }
558        if (trace) {
559            Log.t("BT", "Trace: " + message);
560        }
561        if (log) {
562            Log.l("BT", "Log: " + message);
563        }
564        if (logcat) {
565            Log.c("BT", "Logcat: " + message);
566        }
567        if (logcatRaw) {
568            Log.r("BT", "LogcatRaw: " + message);
569        }
570        if (logcatRawRaw) {
571            Log.r("BT", "LogcatRawRaw: " + message);
572        }
573        if (logcatRawRawRaw) {
574            Log.r("BT", "LogcatRawRawRaw: " + message);
575        }
576        if (logcatRawRawRawRaw) {
577            Log.r("BT", "LogcatRawRawRawRaw: " + message);
578        }
579        if (logcatRawRawRawRawRaw) {
580            Log.r("BT", "LogcatRawRawRawRawRaw: " + message);
581        }
582        if (logcatRawRawRawRawRawRaw) {
583            Log.r("BT", "LogcatRawRawRawRawRawRaw: " + message);
584        }
585    }
586
587    public static void setBlueoothConnectionState(BluetoothConnectionState state, String message, Boolean error, Boolean warning, Boolean info, Boolean debug, Boolean verbose, Boolean trace, Boolean log, Boolean logcat, Boolean logcatRaw, Boolean logcatRawRaw, Boolean logcatRawRawRaw, Boolean logcatRawRawRawRaw, Boolean logcatRawRawRawRawRaw, Boolean logcatRawRawRawRawRawRaw) {
588        blueoothConnectionState = state;
589        Log.d("BT", message);
590        if (error) {
591            Log.e("BT", "Error: " + message);
592        }
593        if (warning) {
594            Log.w("BT", "Warning: " + message);
595        }
596        if (info) {
597            Log.i("BT", "Info: " + message);
598        }
599        if (debug) {
600            Log.d("BT", "Debug: " + message);
601        }
602        if (verbose) {
603            Log.v("BT", "Verbose: " + message);
604        }
605        if (trace) {
606            Log.t("BT", "Trace: " + message);
607        }
608        if (log) {
609            Log.l("BT", "Log: " + message);
610        }
611        if (logcat) {
612            Log.c("BT", "Logcat: " + message);
613        }
614        if (logcatRaw) {
615            Log.r("BT", "LogcatRaw: " + message);
616        }
617        if (logcatRawRaw) {
618            Log.r("BT", "LogcatRawRaw: " + message);
619        }
620        if (logcatRawRawRaw) {
621            Log.r("BT", "LogcatRawRawRaw: " + message);
622        }
623        if (logcatRawRawRawRaw) {
624            Log.r("BT", "LogcatRawRawRawRaw: " + message);
625        }
626        if (logcatRawRawRawRawRaw) {
627            Log.r("BT", "LogcatRawRawRawRawRaw: " + message);
628        }
629        if (logcatRawRawRawRawRawRaw) {
630            Log.r("BT", "LogcatRawRawRawRawRawRaw: " + message);
631        }
632    }
633
634    public static void setBlueoothConnectionState(BluetoothConnectionState state, String message, Boolean error, Boolean warning, Boolean info, Boolean debug, Boolean verbose, Boolean trace, Boolean log, Boolean logcat, Boolean logcatRaw, Boolean logcatRawRaw, Boolean logcatRawRawRaw, Boolean logcatRawRawRawRaw, Boolean logcatRawRawRawRawRaw, Boolean logcatRawRawRawRawRawRaw) {
635        blueoothConnectionState = state;
636        Log.d("BT", message);
637        if (error) {
638            Log.e("BT", "Error: " + message);
639        }
640        if (warning) {
641            Log.w("BT", "Warning: " + message);
642        }
643        if (info) {
644            Log.i("BT", "Info: " + message);
645        }
646        if (debug) {
647            Log.d("BT", "Debug: " + message);
648        }
649        if (verbose) {
650            Log.v("BT", "Verbose: " + message);
651        }
652        if (trace) {
653            Log.t("BT", "Trace: " + message);
654        }
655        if (log) {
656            Log.l("BT", "Log: " + message);
657        }
658        if (logcat) {
659            Log.c("BT", "Logcat: " + message);
660        }
661        if (logcatRaw) {
662            Log.r("BT", "LogcatRaw: " + message);
663        }
664        if (logcatRawRaw) {
665            Log.r("BT", "LogcatRawRaw: " + message);
666        }
667        if (logcatRawRawRaw) {
668            Log.r("BT", "LogcatRawRawRaw: " + message);
669        }
670        if (logcatRawRawRawRaw) {
671            Log.r("BT", "LogcatRawRawRawRaw: " + message);
672        }
673        if (logcatRawRawRawRawRaw) {
674            Log.r("BT", "LogcatRawRawRawRawRaw: " + message);
675        }
676        if (logcatRawRawRawRawRawRaw) {
677            Log.r("BT", "LogcatRawRawRawRawRawRaw: " + message);
678        }
679    }
680
681    public static void setBlueoothConnectionState(BluetoothConnectionState state, String message, Boolean error, Boolean warning, Boolean info, Boolean debug, Boolean verbose, Boolean trace, Boolean log, Boolean logcat, Boolean logcatRaw, Boolean logcatRawRaw, Boolean logcatRawRawRaw, Boolean logcatRawRawRawRaw, Boolean logcatRawRawRawRawRaw, Boolean logcatRawRawRawRawRawRaw) {
682        blueoothConnectionState = state;
683        Log.d("BT", message);
684        if (error) {
685            Log.e("BT", "Error: " + message);
686        }
687        if (warning) {
688            Log.w("BT", "Warning: " + message);
689        }
690        if (info) {
691            Log.i("BT", "Info: " + message);
692        }
693        if (debug) {
694            Log.d("BT", "Debug: " + message);
695        }
696        if (verbose) {
697            Log.v("BT", "Verbose: " + message);
698        }
699        if (trace) {
700            Log.t("BT", "Trace: " + message);
701        }
702        if (log) {
703            Log.l("BT", "Log: " + message);
704        }
705        if (logcat) {
706            Log.c("BT", "Logcat: " + message);
707        }
708        if (logcatRaw) {
709            Log.r("BT", "LogcatRaw: " + message);
710        }
711        if (logcatRawRaw) {
712            Log.r("BT", "LogcatRawRaw: " + message);
713        }
714        if (logcatRawRawRaw) {
715            Log.r("BT", "LogcatRawRawRaw: " + message);
716        }
717        if (logcatRawRawRawRaw) {
718            Log.r("BT", "LogcatRawRawRawRaw: " + message);
719        }
720        if (logcatRawRawRawRawRaw) {
721            Log.r("BT", "LogcatRawRawRawRawRaw: " + message);
722        }
723        if (logcatRawRawRawRawRawRaw) {
724            Log.r("BT", "LogcatRawRawRawRawRawRaw: " + message);
725        }
726    }
727
728    public static void setBlueoothConnectionState(BluetoothConnectionState state, String message, Boolean error, Boolean warning, Boolean info, Boolean debug, Boolean verbose, Boolean trace, Boolean log, Boolean logcat, Boolean logcatRaw, Boolean logcatRawRaw, Boolean logcatRawRawRaw, Boolean logcatRawRawRawRaw, Boolean logcatRawRawRawRawRaw, Boolean logcatRawRawRawRawRawRaw) {
729        blueoothConnectionState = state;
730        Log.d("BT", message);
731        if (error) {
732            Log.e("BT", "Error: " + message);
733        }
734        if (warning) {
735            Log.w("BT", "Warning: " + message);
736        }
737        if (info) {
738            Log.i("BT", "Info: " + message);
739        }
740        if (debug) {
741            Log.d("BT", "Debug: " + message);
742        }
743        if (verbose) {
744            Log.v("BT", "Verbose: " + message);
745        }
746        if (trace) {
747            Log.t("BT", "Trace: " + message);
748        }
749        if (log) {
750            Log.l("BT", "Log: " + message);
751        }
752        if (logcat) {
753            Log.c("BT", "Logcat: " + message);
754        }
755        if (logcatRaw) {
756            Log.r("BT", "LogcatRaw: " + message);
757        }
758        if (logcatRawRaw) {
759            Log.r("BT", "LogcatRawRaw: " + message);
760        }
761        if (logcatRawRawRaw) {
762            Log.r("BT", "LogcatRawRawRaw: " + message);
763        }
764        if (logcatRawRawRawRaw) {
765            Log.r("BT", "LogcatRawRawRawRaw: " + message);
766        }
767        if (logcatRawRawRawRawRaw) {
768            Log.r("BT", "LogcatRawRawRawRawRaw: " + message);
769        }
770        if (logcatRawRawRawRawRawRaw) {
771            Log.r("BT", "LogcatRawRawRawRawRawRaw: " + message);
772        }
773    }
774
775    public static void setBlueoothConnectionState(BluetoothConnectionState state, String message, Boolean error, Boolean warning, Boolean info, Boolean debug, Boolean verbose, Boolean trace, Boolean log, Boolean logcat, Boolean logcatRaw, Boolean logcatRawRaw, Boolean logcatRawRawRaw, Boolean logcatRawRawRawRaw, Boolean logcatRawRawRawRawRaw, Boolean logcatRawRawRawRawRawRaw) {
776        blueoothConnectionState = state;
777        Log.d("BT", message);
778        if (error) {
779            Log.e("BT", "Error: " + message);
780        }
781        if (warning) {
782            Log.w("BT", "Warning: " + message);
783        }
784        if (info) {
785            Log.i("BT", "Info: " + message);
786        }
787        if (debug) {
788            Log.d("BT", "Debug: " + message);
789        }
790        if (verbose) {
791            Log.v("BT", "Verbose: " + message);
792        }
793        if (trace) {
794            Log.t("BT", "Trace: " + message);
795        }
796        if (log) {
797            Log.l("BT", "Log: " + message);
798        }
799        if (logcat) {
800            Log.c("BT", "Logcat: " + message);
801        }
802        if (logcatRaw) {
803            Log.r("BT", "LogcatRaw: " + message);
804        }
805        if (logcatRawRaw) {
806            Log.r("BT", "LogcatRawRaw: " + message);
807        }
808        if (logcatRawRawRaw) {
809            Log.r("BT", "LogcatRawRawRaw: " + message);
810        }
811        if (logcatRawRawRawRaw) {
812            Log.r("BT", "LogcatRawRawRawRaw: " + message);
813        }
814        if (logcatRawRawRawRawRaw) {
815            Log.r("BT", "LogcatRawRawRawRawRaw: " + message);
816        }
817        if (logcatRawRawRawRawRawRaw) {
818            Log.r("BT", "LogcatRawRawRawRawRawRaw: " + message);
819        }
820    }
821
822    public static void setBlueoothConnectionState(BluetoothConnectionState state, String message, Boolean error, Boolean warning, Boolean info, Boolean debug, Boolean verbose, Boolean trace, Boolean log, Boolean logcat, Boolean logcatRaw, Boolean logcatRawRaw, Boolean logcatRawRawRaw, Boolean logcatRawRawRawRaw, Boolean logcatRawRawRawRawRaw, Boolean logcatRawRawRawRawRawRaw) {
823        blueoothConnectionState = state;
824        Log.d("BT", message);
825        if (error) {
826            Log.e("BT", "Error: " + message);
827        }
828        if (warning) {
829            Log.w("BT", "Warning: " + message);
830        }
831        if (info) {
832            Log.i("BT", "Info: " + message);
833        }
834        if (debug) {
835            Log.d("BT", "Debug: " + message);
836        }
837        if (verbose) {
838            Log.v("BT", "Verbose: " + message);
839        }
840        if (trace) {
841            Log.t("BT", "Trace: " + message);
842        }
843        if (log) {
844            Log.l("BT", "Log: " + message);
845        }
846        if (logcat) {
847            Log.c("BT", "Logcat: " + message);
848        }
849        if (logcatRaw) {
850            Log.r("BT", "LogcatRaw: " + message);
851        }
852        if (logcatRawRaw) {
853            Log.r("BT", "LogcatRawRaw: " + message);
854        }
855        if (logcatRawRawRaw) {
856            Log.r("BT", "LogcatRawRawRaw: " + message);
857        }
858        if (logcatRawRawRawRaw) {
859            Log.r("BT", "LogcatRawRawRawRaw: " + message);
860        }
861        if (logcatRawRawRawRawRaw) {
862            Log.r("BT", "LogcatRawRawRawRawRaw: " + message);
863        }
864        if (logcatRawRawRawRawRawRaw) {
865            Log.r("BT", "LogcatRawRawRawRawRawRaw: " + message);
866        }
867    }
868
869    public static void setBlueoothConnectionState(BluetoothConnectionState state, String message, Boolean error, Boolean warning, Boolean info, Boolean debug, Boolean verbose, Boolean trace, Boolean log, Boolean logcat, Boolean logcatRaw, Boolean logcatRawRaw, Boolean logcatRawRawRaw, Boolean logcatRawRawRawRaw, Boolean logcatRawRawRawRawRaw, Boolean logcatRawRawRawRawRawRaw) {
870        blueoothConnectionState = state;
871        Log.d("BT", message);
872        if (error) {
873            Log.e("BT", "Error: " + message);
874        }
875        if (warning) {
876            Log.w("BT", "Warning: " + message);
877        }
878        if (info) {
879            Log.i("BT", "Info: " + message);
880        }
881        if (debug) {
882            Log.d("BT", "Debug: " + message);
883        }
884        if (verbose) {
885            Log.v("BT", "Verbose: " + message);
886        }
887        if (trace) {
888            Log.t("BT", "Trace: " + message);
889        }
890        if (log) {
891            Log.l("BT", "Log: " + message);
892        }
893        if (logcat) {
894            Log.c("BT", "Logcat: " + message);
895        }
896        if (logcatRaw) {
897            Log.r("BT", "LogcatRaw: " + message);
898        }
899        if (logcatRawRaw) {
900            Log.r("BT", "LogcatRawRaw: " + message);
901        }
902        if (logcatRawRawRaw) {
903            Log.r("BT", "LogcatRawRawRaw: " + message);
904        }
905        if (logcatRawRawRawRaw) {
906            Log.r("BT", "LogcatRawRawRawRaw: " + message);
907        }
908        if (logcatRawRawRawRawRaw) {
909            Log.r("BT", "LogcatRawRawRawRawRaw: " + message);
910        }
911        if (logcatRawRawRawRawRawRaw) {
912            Log.r("BT", "LogcatRawRawRawRawRawRaw: " + message);
913        }
914    }
915
916    public static void setBlueoothConnectionState(BluetoothConnectionState state, String message, Boolean error, Boolean warning, Boolean info, Boolean debug, Boolean verbose, Boolean trace, Boolean log, Boolean logcat, Boolean logcatRaw, Boolean logcatRawRaw, Boolean logcatRawRawRaw, Boolean logcatRawRawRawRaw, Boolean logcatRawRawRawRawRaw, Boolean logcatRawRawRawRawRawRaw) {
917        blueoothConnectionState = state;
918        Log.d("BT", message);
919        if (error) {
920            Log.e("BT", "Error: " + message);
921        }
922        if (warning) {
923            Log.w("BT", "Warning: " + message);
924        }
925        if (info) {
926            Log.i("BT", "Info: " + message);
927        }
928        if (debug) {
929            Log.d("BT", "Debug: " + message);
930        }
931        if (verbose) {
932            Log.v("BT", "Verbose: " + message);
933        }
934        if (trace) {
935            Log.t("BT", "Trace: " + message);
936        }
937        if (log) {
938            Log.l("BT", "Log: " + message);
939        }
940        if (logcat) {
941            Log.c("BT", "Logcat: " + message);
942        }
943        if (logcatRaw) {
944            Log.r("BT", "LogcatRaw: " + message);
945        }
946        if (logcatRawRaw) {
947            Log.r("BT", "LogcatRawRaw: " + message);
948        }
949        if (logcatRawRawRaw) {
950            Log.r("BT", "LogcatRawRawRaw: " + message);
951        }
952        if (logcatRawRawRawRaw) {
953            Log.r("BT", "LogcatRawRawRawRaw: " + message);
954        }
955        if (logcatRawRawRawRawRaw) {
956            Log.r("BT
```

```

61                         bluetoothDevicesList = new ArrayList<>();
62                         bluetoothDevicesList.addAll(bondedDevices);
63
64                         if (bondedDevices.size() > 0) {
65                             bluetoothDevicesArray = bondedDevices.toArray(new
66                             BluetoothDevice[bondedDevices.size()]);
67                             }
68                         updateBluetoothListGUI();
69                         } else {
70                             BluetoothActivity.setBluetoothStateGui(
71                             bluetoothConnectionState.BLUETOOTH_DISABLED);
72                         }
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[
91                 x].getName(), AvailableDevice.DeviceState.DEFAULT));
92             }
93
94             if (bluetoothConnected) {
95                 BluetoothActivity.setBluetoothStateGui(blueoothConnectionState.
96                 SYSTEM_CONNECTED);
97                 deviceList.get(blueoothDevicesList.indexOf(targetDevice)).setCurrentState(
98                 AvailableDevice.DeviceState.CONNECTED);
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)&&
110         isBluetoothAdapterReady() {
111             try {
112                 attemptingBluetoothConnection = true;
113                 if (bluetoothConnected) closeAllConnections();
114                 BluetoothActivity.setBluetoothStateGui(blueoothConnectionState.WAITING);
115                 targetDevice = bluetoothDevicesArray[deviceIndex];
116                 btSocket = targetDevice.createRfcommSocketToServiceRecord(btUUID);
117                 btSocket.connect();
118             }
119         }
120     }

```

```

117             if (btSocket.isConnected()) {
118                 inputStream = btSocket.getInputStream();
119                 outputStream = btSocket.getOutputStream();
120                 bluetoothConnected = true;
121                 BluetoothActivity.setBluetoothStateGui(blueoothConnectionState.
122                     SYSTEM_CONNECTED);
122                 BluetoothActivity.updateDevicesList();
123             } else {
124                 bluetoothConnected = false;
125                 BluetoothActivity.setBluetoothStateGui(blueoothConnectionState.
126                     SYSTEM_DISCONNECTED);
126             }
127             attemptingBluetoothConnection = false;
128         } catch (Throwable th) {
129             bluetoothConnected = false;
130             Application.toastMessage(th.toString());
131             BluetoothActivity.setBluetoothStateGui(blueoothConnectionState.
132                 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) &&
142     isBluetoothAdapterReady() {
143         bluetoothConnectorExecutor.execute(new Runnable() {
144             @Override
145             public void run() {
146                 connectBluetoothDeviceSync(attemptingConnectionDeviceIndex);
147             }
148         });
149     }
150 }
151
152 public static void sendCommandBuffer(byte[] data) {
153     try {
154         outputStream.write(new byte[] {(byte) 0x7F, (byte) 0x7F, (byte) 0x7F, (byte) 0x7F});
155         outputStream.write(data, 0, 20);
156     } catch (Throwable th) {
157         Application.toastMessage(th.toString());
158     }
159 }
160
161 public static byte[] receiveCommandBuffer(int timeout) {
162     long startTimeMillis = System.currentTimeMillis();
163     while((System.currentTimeMillis()-startTimeMillis)<500L) {
164         try {
165             if (inputStream.available() >= 32) {
166                 byte[] inputBuffer = new byte[32];
167                 inputStream.read(inputBuffer, 0, 32);
168                 return inputBuffer;
169             }
170         } catch (Throwable th){  
202  
            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 =
197                                         inputBuffer;
198                                     DeveloperConsoleActivity.sendTextToTerminal("Received
199                                         from: " + UnifiedTransmissionProtocol.lastMessageTransmitterModule.getModuleName() + " : "
200                                         + TerminalManager.insertSpaces(UnifiedTransmissionProtocol.bytesToHex(inputBuffer), " ", 2)
201                                         );
202                                     if (UnifiedTransmissionProtocol.
203                                         DecomposeMessageFromBuffer() == UnifiedTransmissionProtocol.CommandStatus.
204                                         SUCCESSFUL_DECOMPOSITION) {
205                                             UnifiedTransmissionProtocol.handleReceivedCommand()
206                                         ;
207                                         } else {
208                                             Application.toastMessage("UNSUCCESSFUL
209                                         DECOMPOSITION");
210                                         }
211                                         }
212                                     } catch (Throwable th) {
213                                         Application.toastMessage(th.toString());
214                                         }
215                                     isAutoConnectRunning = false;
216                                 }
217                             });
218                         }
219
220
221
222
223                         203
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                 }
238                 while (!terminateAutoConnect && isBluetoothAdapterReady()) {
239                     listAvailableBluetoothDevices();
240                     while (isListAvailableBluetoothDevicesRunning) {}
241                     for (int x = 0; x < bluetoothDevicesArray.length; x++) {
242                         // MAC ADDRESS HC-05 : 98:D3:91:FD:38:05
243                         // MAC ADDRESS XT1068 : 5C:51:88:A2:38:97
244                         if (bluetoothDevicesArray[x].getAddress().compareToIgnoreCase("98:D3:91:FD:38:05") == 0) {
245                             if (!terminateAutoConnect&&!bluetoothConnected)
246                                 connectBluetoothDeviceSync(x);
247                             }
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){ 204
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,
297                     BluetoothAdapter.ERROR);
298
299                 switch (stateCode){
300                     case BluetoothAdapter.STATE_OFF:
301                         //Bluetooth adapter turned off handler
302                         BluetoothActivity.setBluetoothStateGui(blueoothConnectionState.
303                             BLUETOOTH_DISABLED);
304                         break;
305                     case BluetoothAdapter.STATE_TURNING_OFF:
306                         //Bluetooth adapter turning off handler
307                         terminateBTAutoConnect();
308                         BluetoothActivity.setBluetoothStateGui(blueoothConnectionState.
309                             BT_TURNING_OFF);
310                         break;
311                     case BluetoothAdapter.STATE_ON:
312                         BluetoothActivity.setBluetoothStateGui(blueoothConnectionState.
313                             SYSTEM_DISCONNECTED);
314                         updateDevicesList();
315                         initBTAutoConnectService();
316
317                         break;
318                     case BluetoothAdapter.STATE_TURNING_ON:
319                         //Bluetooth adapter turning on handler
320                         BluetoothActivity.setBluetoothStateGui(blueoothConnectionState.
321                             BT_TURNING_ON);
322                         break;
323
324         public static BroadcastReceiver bluetoothScanModeStatusBR = new BroadcastReceiver() {
325
326             @Override
327             public void onReceive(Context context, Intent intent) {
328                 final String action = intent.getAction();
329
330                 if (action.equals(BluetoothAdapter.ACTION_SCAN_MODE_CHANGED)) {
331
332                     int mode = intent.getIntExtra(BluetoothAdapter.EXTRA_SCAN_MODE,
333                         BluetoothAdapter.ERROR);
334
335                     switch (mode) {
336                         case BluetoothAdapter.SCAN_MODE_CONNECTABLE_DISCOVERABLE:
337
338                             break;
339                         case BluetoothAdapter.SCAN_MODE_CONNECTABLE:
340
341                             break;
342                         case BluetoothAdapter.SCAN_MODE_NONE:
343
344                             break;
345
346             }
347         }
348     }
349 }
```

```
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(blueoothConnectionState.
363 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
```

```

1 package com.example.SAPLM.bluetoothActivities;
2
3 import com.example.SAPLM.R;
4 import com.example.SAPLM.commandActivity.CommandsActivity;
5 import com.example.SAPLM.settingsActivities.DeveloperConsoleActivity;
6 import com.example.SAPLM.settingsActivities.TerminalManager;
7
8
9 import java.io.UnsupportedEncodingException;
10 import java.nio.ByteBuffer;
11 import java.nio.ByteOrder;
12 import java.nio.charset.StandardCharsets;
13 import java.util.ArrayList;
14 import java.util.Arrays;
15 import java.util.List;
16 import java.util.concurrent.Executor;
17 import java.util.concurrent.Executors;
18 import java.util.regex.Matcher;
19 import java.util.regex.Pattern;
20
21
22
23 public class UnifiedTransmissionProtocol {
24
25     public static List<Module> modulesList = new ArrayList<UnifiedTransmissionProtocol.
26     Module>(Arrays.asList(
27         new Module("Dispositivo Móvil", "PHONE", (byte) 0x00, R.drawable.
28             ic_icon_phone_module, new ArrayList<Module.Device>(Arrays.asList(
29                 new Module.Device("Módulo principal", new Boolean(false), R.drawable.
30                     ic_icon_main_board, (byte) 0x00, new Boolean(true))
31                 ))),
32         new Module("Módulo principal", "MAIN", (byte) 0x01, R.drawable.ic_icon_main_board,
33         new ArrayList<Module.Device>(Arrays.asList(
34             new Module.Device("Módulo de potencia", new Boolean(false), R.drawable.
35                 icons_board, (byte) 0x00, new Boolean(true)),
36             new Module.Device("Módulo motriz", new Boolean(false), R.drawable.
37                 icons_board, (byte) 0x01, new Boolean(true))
38             ))),
39         new Module("Módulo de potencia", "POWER", (byte) 0x02, R.drawable.icons_board, new
40         ArrayList<Module.Device>(Arrays.asList(
41             new Module.Device("Velador", new Boolean(false), R.drawable.
42                 icons_desk_lamp, (byte) 0x00, new Boolean(false)),
43             new Module.Device("Luz", new Boolean(false), R.drawable.icons_bulb, (
44                 byte) 0x01, new Boolean(false)),
45             new Module.Device("Ventilador", new Boolean(false), R.drawable.icons_fan
46             , (byte) 0x02, new Boolean(false)),
47             new Module.Device("Estufa", new Boolean(false), R.drawable.icons_heater,
48             (byte) 0x03, new Boolean(false))
49             ))),
50         new Module("Módulo motriz", "MOTOR", (byte) 0x03, R.drawable.icons_board, new
51         ArrayList<Module.Device>(Arrays.asList(
52             new Module.Device("Camilla", new Byte((byte) 0), R.drawable.
53                 icons_stretcher, (byte) 0x00, new Boolean(false)),
54             new Module.Device("Cortina", new Byte((byte) 0), R.drawable.icons_curtain
55             , (byte) 0x01, new Boolean(false)),
56             new Module.Device("Llamar enfermera", new Boolean(false), R.drawable.
57                 icons_nurse, (byte) 0x02, new Boolean(false))))))
58     );
59
60     protected static byte[] command_buffer = new byte[32];
61
62     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    }

```

Archivo - UnifiedTransmissionProtocol.java

```

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         );
117             Pattern footerPattern = Pattern.compile("\u0003.{1}\u0017", Pattern.DOTALL
118         );
119
120             Matcher headerMatcher = headerPattern.matcher(commandString);
121             Matcher footerMatcher = footerPattern.matcher(commandString);
122
123             if (headerMatcher.find()==false) return CommandStatus.WRONG_HEADER_SEGMENTATION
124 ;
125             if (footerMatcher.find()==false) return CommandStatus.WRONG_FOOTER_SEGMENTATION
126 ;
127
128             int headerstartIndex = headerMatcher.start();
129             int parameterstartIndex = headerMatcher.end()+1;
130             int footerstartIndex = footerMatcher.start();
131
132             lastMessagePID = command_buffer[headerstartIndex+1]; //verificar si hay que
133             convertir a desde unsigned
134             if (command_buffer[headerstartIndex+2]!=currentModuleID.getInternalCode())
135                 return CommandStatus.WRONG_MODULE_ID;
136
137             lastMessageTransmitterModule = Module.getModule(command_buffer[headerstartIndex
138             +3]);
139
140             lastCommandType = CommandType.definedCommandList.stream().filter(p-> p.
141             getCommandCode()==command_buffer[headerMatcher.end()]).findFirst().orElse(null);
142
143             lastParameterList = new ArrayList<Parameter>();
144
145             while(parameterstartIndex!=footerstartIndex) {
146                 int parameterLength = command_buffer[parameterstartIndex];
147                 byte[] parameterData = new byte[parameterLength];
148                 System.arraycopy(command_buffer, parameterstartIndex+1, parameterData, 0,
149                 parameterLength);
150                 lastParameterList.add(new Parameter(parameterData));
151                 parameterstartIndex+=(1+parameterLength);
152             }
153
154             if (command_buffer[footerstartIndex+1]!=CRC8_CCITT.crc_CCITT(command_buffer,
155             footerstartIndex)) return CommandStatus.WRONG_CHECKSUM_CONSISTENCY;
156
157         } catch (UnsupportedEncodingException e) {
158             e.printStackTrace();
159             return CommandStatus.FAILED_DECOMPOSITION;
160         }
161
162         isAvailableCommand = true;
163         return CommandStatus.SUCCESSFUL_DECOMPOSITION;
164     }
165
166     public static void handleReceivedCommand() {
167         if (isAvailableCommand) {
168             if (lastCommandType!=null) lastCommandType.handleCommand();
169             isAvailableCommand = false;
170         }
171     }

```

```

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,
195     List<Device> devicesList) {
196         this.moduleName = moduleName;
197         this.internalCode = internalCode;
198         this.iconResourceId = iconResourceId;
199         this.devicesList = devicesList;
200     }
201
202     Module() {
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() { 210
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.
231             Parameter>();
232
233         for (int x = 0; x < devicesList.size(); x++) {
234             for (Module.Device device : devicesList) {
235                 if (x==byteToUnsignedInt(device.getInternalIndex())) {
236                     Parameter newParameter = null;
237
238                     if (device.getData().getClass()==Byte.class) {
239                         ByteBuffer byteBuffer = ByteBuffer.allocate(Byte.BYTES);
240                         byteBuffer.order(ByteOrder.LITTLE_ENDIAN);
241                         byteBuffer.put((byte) device.getData());
242                         newParameter = new Parameter(byteBuffer.array());
243                     } else if (device.getData().getClass()==Short.class) {
244                         ByteBuffer shortBuffer = ByteBuffer.allocate(Short.BYTES);
245                         shortBuffer.order(ByteOrder.LITTLE_ENDIAN);
246                         shortBuffer.putShort((short) device.getData());
247                         newParameter = new Parameter(shortBuffer.array());
248                     } else if (device.getData().getClass()==Integer.class) {
249                         ByteBuffer intBuffer = ByteBuffer.allocate(Integer.BYTES);
250                         intBuffer.order(ByteOrder.LITTLE_ENDIAN);
251                         intBuffer.putInt((int) device.getData());
252                         newParameter = new Parameter(intBuffer.array());
253                     } else if (device.getData().getClass()==Float.class) {
254                         ByteBuffer floatBuffer = ByteBuffer.allocate(Float.BYTES);
255                         floatBuffer.order(ByteOrder.LITTLE_ENDIAN);
256                         floatBuffer.putFloat((float) (double)device.getData());
257                         newParameter = new Parameter(floatBuffer.array());
258                     } else if (device.getData().getClass()==Double.class) {
259                         ByteBuffer doubleBuffer = ByteBuffer.allocate(Float.BYTES);
260                         doubleBuffer.order(ByteOrder.LITTLE_ENDIAN);
261                         doubleBuffer.putFloat((float) (double)device.getData());
262                         newParameter = new Parameter(doubleBuffer.array());
263                     } else if (device.getData().getClass()==String.class) {
264                         String data = (String) device.getData();
265                         newParameter = new Parameter(data.getBytes(StandardCharsets.
266                             US_ASCII));
267                     } else if (device.getData().getClass()==Boolean.class) {
268                         byte data = (byte) 0x00;
269                         if ((Boolean)device.getData()==true) data = (byte) 0xFF;
270                         ByteBuffer booleanBuffer = ByteBuffer.allocate(Byte.BYTES);
271                         booleanBuffer.order(ByteOrder.LITTLE_ENDIAN);
272                         booleanBuffer.put(data);
273                         newParameter = new Parameter(booleanBuffer.array());
274                     }
275
276                     if (newParameter!=null) exportList.add(newParameter);
277                     break;
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     );
291         buffer.order(ByteOrder.LITTLE_ENDIAN);
292
293         if (deviceToUpdate.getData().getClass() == Byte.class) {
294             deviceToUpdate.setData(new Byte(buffer.get()));
295         } else if (deviceToUpdate.getData().getClass() == Short.class) {
296             deviceToUpdate.setData(new Short(buffer.getShort()));
297         } else if (deviceToUpdate.getData().getClass() == Integer.class) {
298             deviceToUpdate.setData(new Integer(buffer.getInt()));
299         } else if (deviceToUpdate.getData().getClass() == Float.class) {
300             deviceToUpdate.setData(new Float(buffer.getFloat()));
301         } else if (deviceToUpdate.getData().getClass() == Double.class) {
302             deviceToUpdate.setData(new Double(buffer.getFloat()));
303         } else if (deviceToUpdate.getData().getClass() == String.class) {
304             deviceToUpdate.setData(new String(parameterList.get(x).getParameterData()
305             (), StandardCharsets.US_ASCII));
306         } else if (deviceToUpdate.getData().getClass() == Boolean.class) {
307             if (buffer.get() == (byte) 0xFF) {
308                 deviceToUpdate.setData(new Boolean(true));
309             } else {
310                 deviceToUpdate.setData(new Boolean(false));
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(Double.BYTES);
374                     doubleBuffer.order(ByteOrder.LITTLE_ENDIAN);
375                     doubleBuffer.putFloat((float) ((double) devicesList.get(x).getData()))
376
377                     return doubleBuffer.array();
378                 } else if (devicesList.get(x).getData().getClass() == String.class) {
379                     String data = (String) devicesList.get(x).getData();
380                     return data.getBytes(StandardCharsets.US_ASCII);
381                 } else if (devicesList.get(x).getData().getClass() == Boolean.class) {
382                     byte data = (byte) 0x00;
383                     if ((Boolean) devicesList.get(x).getData() == true) data = (byte) 0xFF;
384                     ByteBuffer booleanBuffer = ByteBuffer.allocate(Byte.BYTES);
385                     booleanBuffer.order(ByteOrder.LITTLE_ENDIAN);
386                     booleanBuffer.put(data);
387                     return booleanBuffer.array();
388                 }
389             }
390
391             return null;
392         }
393
394
395         private static Executor synchronizationExecutor = Executors.newSingleThreadExecutor
396     ();
397
398         private static boolean synchronizationRunning = false;
399
400         public static boolean awaitingModuleResponse = false;
401         public static byte awaitingModuleResponse_ModuleInternalCode = (byte) 0x00;
402         public static boolean awaitingSynchronization = false;
403
404
405         213
406         public static void syncAllDevicesValueToSystem() {
407             // Send data to system from internal data
408
409             if (synchronizationRunning == false) {
410                 synchronizationExecutor.execute(new Runnable() {

```

## Archivo - UnifiedTransmissionProtocol.java

```

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 &&
416                                 dev.isSynchronizedData() == false) {
417                                 List<Parameter> export = mod.
418                                     exportDataToParameterList();
419                                 ComposeMessageToBuffer(CommandType.
420                                     UPDATE_ALL_DEVICES_VALUE, mod, export);
421                                 DeveloperConsoleActivity.sendTextToTerminal("Sent
422                                     to: " + mod.getModuleName() + " : " + TerminalManager.insertSpaces(bytesToHex(
423                                     command_buffer), " ", 2));
424                                 awaitingModuleResponse = true;
425                                 awaitingModuleResponse_ModuleInternalCode = mod.
426                                     internalCode;
427                                 BluetoothConnection.sendCommandBuffer(
428                                     command_buffer);
429                                 long startTimeMillis = System.currentTimeMillis();
430                                 while ((System.currentTimeMillis() -
431                                     startTimeMillis) < 1000L) {
432                                     if (awaitingModuleResponse == false) {
433                                         for (Module.Device devSync : mod.
434                                             getDevicesList()) {
435                                             devSync.setSynchronizedState(true);
436                                         }
437                                     }
438                                     awaitingModuleResponse = false;
439                                     break; //Only send one message per module
440                                 }
441                             }
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                 synchronizationRunning = true;
452
453                 synchronizationRunning = false;
454             }
455         }
456
457
458
459
460             public static Module getModule(byte internalCode) {
461                 for (Module mod : modulesList) {
214

```

```

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
483             .lineSeparator();
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() {      215
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 {
554                     setData(Boolean.FALSE); }
555                 break;
556             case "Luz":
557                 if ((boolean) data == false) { setData(Boolean.TRUE); } else {
558                     setData(Boolean.FALSE); }
559                 break;
560             case "Ventilador":
561                 if ((boolean) data == false) { setData(Boolean.TRUE); } else {
562                     setData(Boolean.FALSE); }
563                 break;
564             case "Estufa":
565                 if ((boolean) data == false) { setData(Boolean.TRUE); } else {
566                     setData(Boolean.FALSE); }
567                 break;
568             case "Camilla":
569                 if (byteToUnsignedInt((byte) data) < 3) { setData((byte) (((byte) data) +
570 1)); } else { setData((byte) 0x00); }
571                 break;
572             case "Cortina":
573                 if (byteToUnsignedInt((byte) data) < 7) { setData((byte) (((byte) data) +
574 1)); } else { setData((byte) 0x00); }
575                 break;
576             case "Llamar enfermera":
577                 if ((boolean) data == false) {
578                     setData(Boolean.TRUE);
579                     autoToggleNurseStateThread = new Thread(
580                         autoToggleNurseStateRunnable);
581                     autoToggleNurseStateThread.start();
582                 } else { setData(Boolean.FALSE); }
583                 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
587 = 100; }
588                 break;
589             case "Módulo de potencia":
590                 if ((boolean) data == false) { returnValue = 0; } else { returnValue
591 = 100; }
592                 break;
593             case "Módulo motriz":
594                 if ((boolean) data == false) { returnValue = 0; } else { returnValue
595 = 100; }
596                 break;
597             case "Velador":
598                 if ((boolean) data == false) { returnValue = 0; } else { returnValue
599 = 100; }
600                 break;
601             case "Luz":
602                 if ((boolean) data == false) { returnValue = 0; } else { returnValue
603 = 100; }
604                 break;
605             case "Ventilador":
606                 if ((boolean) data == false) { returnValue = 0; } else { returnValue
607 = 100; }
608                 break;
609             case "Estufa":
610                 if ((boolean) data == false) { returnValue = 0; } else { returnValue
611 = 100; }
612                 break;
613             case "Camilla":
614                 returnValue = Math.round((float)((byte) data)*33.333f));
615                 break;
616             case "Cortina":
617                 returnValue = Math.round((float)((byte) data)*14.286f));
618                 break;
619             case "Llamar enfermera":
620                 if ((boolean) data == false) { returnValue = 0; } else { returnValue
621 = 100; }
622                 break;
623         }
624         return returnValue;
625     }
626
627     public String getDataAsString(){
628         String returnString = "";
629         switch (name){
630             case "Módulo principal":
631                 if ((boolean) data == false) { returnString = "Desconectado"; } else
632 { returnString = "En linea"; } 217
633                 break;
634             case "Módulo de potencia":
635                 if ((boolean) data == false) { returnString = "Desconectado"; } else
636 { returnString = "En linea"; }
637                 break;
638             case "Módulo motriz":
639

```

```

631                     if ((boolean) data == false) { returnString = "Desconectado"; } else
632             { returnString = "En linea"; }
633             break;
634         case "Velador":
635             if ((boolean) data == false) { returnString = "Apagado"; } else {
636                 returnString = "Encendido"; }
637                 break;
638             case "Luz":
639                 if ((boolean) data == false) { returnString = "Apagado"; } else {
640                     returnString = "Encendido"; }
641                     break;
642                 case "Ventilador":
643                     if ((boolean) data == false) { returnString = "Apagado"; } else {
644                         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 {
655                     returnString = "Llamando"; }
656                     break;
657             }
658         return returnString;
659     }
660
661     public static Thread autoToggleNurseStateThread;
662     Runnable autoToggleNurseStateRunnable = new Runnable() {
663         @Override
664         public void run() {
665             try {
666                 Thread.sleep(3000);
667                 modulesList.get(3).getDevicesList().get(2).setData(new Boolean(
668                     false));
669                 modulesList.get(3).getDevicesList().get(2).setSynchronizedState(
670                     true);
671                 CommandsActivity.updateGUI();
672             } catch (InterruptedException e) {
673                 e.printStackTrace();
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;

```

Archivo - UnifiedTransmissionProtocol.java

```
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                 ByteBuffer buffer = ByteBuffer.wrap(lastParameterList.get(0).
    getParameterData());
742
743                 modulesList.get(byteToUnsignedInt(lastMessageTransmitterModule.
    getInternalCode())).updateDeviceValue(buffer.get(), lastParameterList.get(1));
744
745             } })
746     );
747 }
```

```

744     getParameterData();
745
746             System.out.println("Running UPDATE_DEVICE_VALUE!");
747
748
749         }}}},
750     new CommandType((byte)0x02, "GET_ALL_DEVICES_VALUE", new Runnable() {
751         @Override
752         public void run() {
753             ComposeMessageToBuffer(defineCommandList.stream().filter(p-> p.
754             getCommandTag().equals(("UPDATE_ALL_DEVICES_VALUE"))).findFirst().orElse(null),
755                     lastMessageTransmitterModule,
756                     modulesList.get(byteToUnsignedInt(
757                         lastMessageTransmitterModule.getInternalCode()))).exportDataToParameterList());
758
759             // SEND COMMAND_BUFFER VIA BLUETOOTH
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).
768             getParameterData());
769             byte[] deviceData = modulesList.get(byteToUnsignedInt(
770                 lastMessageTransmitterModule.getInternalCode())).exportDeviceValue(buffer.get());
771
772             ComposeMessageToBuffer(defineCommandList.stream().filter(p-> p.
773             getCommandTag().equals(("UPDATE_DEVICE_VALUE"))).findFirst().orElse(null),
774                     lastMessageTransmitterModule,
775                     new ArrayList<Parameter>(Arrays.asList(new Parameter(new
776                     byte[] {buffer.get()}), new Parameter(deviceData))));
777
778             // SEND COMMAND_BUFFER VIA BLUETOOTH
779
780             System.out.println("Running GET_DEVICE_VALUE!");
781         }}),
782     new CommandType((byte)0x04, "MESSAGE_STATUS", new Runnable() {
783         @Override
784         public void run() {
785
786             ByteBuffer buffer = ByteBuffer.wrap(lastParameterList.get(0).
787             getParameterData());
788             if (buffer.get()==(byte)0x09){
789                 Module.awaitingModuleResponse = false;
790             }
791
792
793         });
794
795         public static CommandType UPDATE_ALL_DEVICES_VALUE = defineCommandList.get(0);
796         public static CommandType UPDATE_DEVICE_VALUE = defineCommandList.get(1);
797         public static CommandType GET_ALL_DEVICES_VALUE = defineCommandList.get(2);
798         public static CommandType GET_DEVICE_VALUE = defineCommandList.get(3);
799         public static CommandType MESSAGE_STATUS = defineCommandList.get(4);
800     }

```

```
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
```



## Código de fuente: Diseño de activities, (Lenguaje: XML)

### Contenidos:

activity\_main.xml  
appearance\_miscellaneous\_layout.xml  
bluetooth\_layout.xml  
bluetooth\_list\_layout\_style.xml  
bluetooth\_settings\_layout.xml  
commands\_layout.xml  
commands\_layout\_child\_item\_style.xml  
commands\_layout\_parent\_item\_style.xml  
commands\_settings\_layout.xml  
contacts\_communication\_layout.xml  
contacts\_layout.xml  
developer\_console\_layout.xml  
information\_help\_layout.xml  
internet\_connectivity\_layout.xml  
settings\_layout.xml  
settings\_layout\_style.xml  
voice\_recognition\_settings\_layout.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com
 /apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     android:background="@android:color/background_light"
8     tools:context=".MainActivity">
9
10    <ImageButton
11        android:id="@+id/imgBtn_microphone_background"
12        android:layout_width="160dp"
13        android:layout_height="160dp"
14        android:layout_marginStart="8dp"
15        android:layout_marginEnd="8dp"
16        android:background="?attr/selectableItemBackgroundBorderless"
17        android:onClick="onClickListener_microphone"
18        android:scaleType="fitXY"
19        app:layout_constraintBottom_toBottomOf="parent"
20        app:layout_constraintEnd_toEndOf="parent"
21        app:layout_constraintStart_toStartOf="parent"
22        app:layout_constraintTop_toTopOf="parent"
23        android:tint="@color/ColorBackMicKeyword"
24        app:srcCompat="@drawable/ic_main_mic_circle"
25        android:stateListAnimator="@null"/>
26
27    <ImageButton
28        android:id="@+id/imgBtn_microphone"
29        android:layout_width="115dp"
30        android:layout_height="115dp"
31        android:layout_marginStart="8dp"
32        android:layout_marginEnd="8dp"
33        android:background="?attr/selectableItemBackgroundBorderless"
34        android:onClick="onClickListener_microphone"
35        android:scaleType="fitXY"
36        android:stateListAnimator="@null"
37        android:tint="@color/ColorMic"
38        app:layout_constraintBottom_toBottomOf="parent"
39        app:layout_constraintEnd_toEndOf="parent"
40        app:layout_constraintStart_toStartOf="parent"
41        app:layout_constraintTop_toTopOf="parent"
42        app:layout_constraintVertical_bias="0.501"
43        app:srcCompat="@drawable/ic_main_mic_listening" />
44
45    <ImageButton
46        android:id="@+id/imgBtn_commands"
47        android:layout_width="55dp"
48        android:layout_height="55dp"
49        android:layout_marginStart="284dp"
50        android:alpha="0.0"
51        android:background="?attr/selectableItemBackgroundBorderless"
52        android:onClick="onClickListener_commands"
53        android:scaleType="fitXY"
54        android:visibility="gone"
55        app:layout_constraintBottom_toBottomOf="parent"
56        app:layout_constraintStart_toStartOf="parent"
57        app:layout_constraintTop_toTopOf="parent"
58        app:srcCompat="@drawable/ic_commands_menu" />
59
60    <ImageButton
61        android:id="@+id/imgBtn_contacts"
62        android:layout_width="55dp"
```

```
63      android:layout_height="55dp"
64      android:layout_marginStart="283dp"
65      android:alpha="0.0"
66      android:background="?attr/selectableItemBackgroundBorderless"
67      android:onClick="onClickListener_contacts"
68      android:scaleType="fitXY"
69      android:visibility="gone"
70      app:layout_constraintBottom_toBottomOf="parent"
71      app:layout_constraintStart_toStartOf="parent"
72      app:layout_constraintTop_toTopOf="parent"
73      app:srcCompat="@drawable/ic_contacts_menu" />
74
75  <ImageButton
76      android:id="@+id/imgBtn_bluetooth"
77      android:layout_width="55dp"
78      android:layout_height="55dp"
79      android:layout_marginStart="283dp"
80      android:alpha="0.0"
81      android:background="?attr/selectableItemBackgroundBorderless"
82      android:onClick="onClickListener_bluetooth"
83      android:scaleType="fitXY"
84      android:visibility="gone"
85      app:layout_constraintBottom_toBottomOf="parent"
86      app:layout_constraintStart_toStartOf="parent"
87      app:layout_constraintTop_toTopOf="parent"
88      app:srcCompat="@drawable/ic_icono_bluetooth_remake" />
89
90  <ImageButton
91      android:id="@+id/imgBtn_settings"
92      android:layout_width="55dp"
93      android:layout_height="55dp"
94      android:layout_marginStart="283dp"
95      android:alpha="0.0"
96      android:background="?attr/selectableItemBackgroundBorderless"
97      android:onClick="onClickListener_settings"
98      android:scaleType="fitXY"
99      android:visibility="gone"
100     app:layout_constraintBottom_toBottomOf="parent"
101     app:layout_constraintStart_toStartOf="parent"
102     app:layout_constraintTop_toTopOf="parent"
103     app:srcCompat="@drawable/ic_settings_menu" />
104
105 <ImageButton
106     android:id="@+id/imgBtn_menu"
107     android:layout_width="55dp"
108     android:layout_height="55dp"
109     android:layout_marginStart="8dp"
110     android:layout_marginEnd="8dp"
111     android:background="?attr/selectableItemBackgroundBorderless"
112     android:onClick="onClickListener_menu"
113     android:scaleType="fitXY"
114     app:layout_constraintBottom_toBottomOf="parent"
115     app:layout_constraintEnd_toEndOf="parent"
116     app:layout_constraintStart_toEndOf="@+id/imgBtn_microphone_background"
117     app:layout_constraintTop_toTopOf="parent"
118     app:srcCompat="@drawable/ic_bluetooth_menu" />
119
120 <TextView
121     android:id="@+id/textView_speechRecognitionResults"
122     style="@style/hypothesisTextAppearance"
123     android:layout_width="311dp"
124     android:layout_height="101dp"
125     android:layout_marginStart="8dp"
```

224

Archivo - activity\_main.xml

```
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\nholia comer te va picante\nholia 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>
```

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com
 /apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent">
7
8     <androidx.recyclerview.widget.RecyclerView
9         android:id="@+id/commandsRecyclerView"
10        android:layout_width="match_parent"
11        android:layout_height="match_parent"
12        app:layout_behavior="@string/appbar_scrolling_view_behavior"
13        android:clipToPadding="false"
14        android:paddingBottom="8dp"
15        android:scrollbarStyle="outsideOverlay"
16        app:layout_constraintBottom_toBottomOf="parent"
17        app:layout_constraintStart_toStartOf="parent"
18        app:layout_constraintTop_toBottomOf="@+id/toolbar"
19        app:layout_constraintVertical_bias="0.0"
20        android:paddingTop="?attr actionBarSize" />
21
22
23     <androidx.appcompat.widget.Toolbar
24         android:id="@+id/toolbar"
25         android:layout_width="0dp"
26         android:layout_height="57dp"
27         android:background="@color/toolBarColor"
28         app:titleTextColor="@color/colorPrimary"
29         android:minHeight="?attr actionBarSize"
30         android:outlineAmbientShadowColor="@android:color/background_light"
31         android:outlineSpotShadowColor="@android:color/background_light"
32         android:theme="?attr actionBarTheme"
33         app:title="@string/commands_activity_title"
34         app:layout_constraintEnd_toEndOf="parent"
35         app:layout_constraintHorizontal_bias="0.0"
36         app:layout_constraintStart_toStartOf="parent"
37         app:layout_constraintTop_toTopOf="parent"
38         app:subtitleTextColor="@android:color/background_light"
39         app:titleTextAppearance="@style/toolBarTextAppearance"/>
40
41
42
43 </androidx.constraintlayout.widget.ConstraintLayout>
```

Archivo - contacts\_layout.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com
 /apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent">
7
8     <androidx.appcompat.widget.Toolbar
9         android:id="@+id/toolbar"
10        android:layout_width="0dp"
11        android:layout_height="57dp"
12        android:background="@color/toolBarColor"
13        app:titleTextColor="@color/colorPrimary"
14        android:minHeight="?attr/actionBarSize"
15        android:outlineAmbientShadowColor="@android:color/background_light"
16        android:outlineSpotShadowColor="@android:color/background_light"
17        android:theme="?attr actionBarTheme"
18        app:title="@string/contacts_activity_title"
19        app:layout_constraintEnd_toEndOf="parent"
20        app:layout_constraintHorizontal_bias="0.0"
21        app:layout_constraintStart_toStartOf="parent"
22        app:layout_constraintTop_toTopOf="parent"
23        app:subtitleTextColor="@android:color/background_light"
24        app:titleTextAppearance="@style/toolBarTextAppearance"/>
25
26
27
28 </androidx.constraintlayout.widget.ConstraintLayout>
```

Archivo - settings\_layout.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com
 /apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent">
7
8     <androidx.appcompat.widget.Toolbar
9         android:id="@+id/toolbar"
10        android:layout_width="0dp"
11        android:layout_height="57dp"
12        android:background="@color/toolBarColor"
13        android:minHeight="?attr/actionBarSize"
14        android:outlineAmbientShadowColor="@android:color/background_light"
15        android:outlineSpotShadowColor="@android:color/background_light"
16        android:theme="?attr actionBarTheme"
17        app:layout_constraintEnd_toEndOf="parent"
18        app:layout_constraintHorizontal_bias="0.0"
19        app:layout_constraintStart_toStartOf="parent"
20        app:layout_constraintTop_toTopOf="parent"
21        app:subtitleTextColor="@android:color/background_light"
22        app:title="@string/settings_activity_title"
23        app:titleTextColor="@color/colorPrimary"
24        app:titleTextAppearance="@style/toolBarTextAppearance"/>
25
26     <androidx.recyclerview.widget.RecyclerView
27         android:id="@+id/settingsRecyclerView"
28         android:layout_width="match_parent"
29         android:layout_height="600dp"
30         android:clipToPadding="false"
31         android:scrollbarStyle="outsideOverlay"
32         android:paddingBottom="8dp"
33         app:layout_constraintStart_toStartOf="parent"
34         app:layout_constraintTop_toBottomOf="@+id/toolbar" />
35
36 </androidx.constraintlayout.widget.ConstraintLayout>
37
```

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com
 /apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     android:clipToPadding="false">
8
9     <androidx.appcompat.widget.Toolbar
10        android:id="@+id/toolbar"
11        android:layout_width="0dp"
12        android:layout_height="57dp"
13        android:background="@color/toolBarColor"
14        android:minHeight="?attr/actionBarSize"
15        android:outlineAmbientShadowColor="@android:color/background_light"
16        android:outlineSpotShadowColor="@android:color/background_light"
17        android:theme="?attr actionBarTheme"
18        app:layout_constraintEnd_toEndOf="parent"
19        app:layout_constraintHorizontal_bias="0.0"
20        app:layout_constraintStart_toStartOf="parent"
21        app:layout_constraintTop_toTopOf="parent"
22        app:subtitleTextColor="@android:color/background_light"
23        app:title="@string/bluetooth_activity_title"
24        app:titleTextAppearance="@style/toolBarTextAppearance"
25        app:titleTextColor="@color/colorPrimary" />
26
27     <ImageView
28        android:id="@+id/imageView"
29        android:layout_width="0dp"
30        android:layout_height="418dp"
31        android:layout_marginBottom="8dp"
32        android:scaleType="fitXY"
33        app:layout_constraintBottom_toBottomOf="@+id/guideline4"
34        app:layout_constraintEnd_toEndOf="@+id/guideline2"
35        app:layout_constraintHorizontal_bias="0.0"
36        app:layout_constraintStart_toStartOf="@+id/guideline"
37        app:layout_constraintTop_toBottomOf="@+id/guideline3"
38        app:layout_constraintVertical_bias="0.012"
39        app:srcCompat="@drawable/bt_dialog" />
40
41
42     <ImageView
43        android:id="@+id/imgView_bluetoothConfig"
44        android:layout_width="320dp"
45        android:layout_height="78dp"
46        android:layout_marginBottom="24dp"
47        android:scaleType="fitXY"
48        app:layout_constraintBottom_toBottomOf="parent"
49        app:layout_constraintEnd_toEndOf="parent"
50        app:layout_constraintStart_toStartOf="parent"
51        app:layout_constraintTop_toBottomOf="@+id/imgView_bluetoothStatus"
52        app:layout_constraintVertical_bias="0.88"
53        app:srcCompat="@drawable/bt_button" />
54
55     <ImageView
56        android:id="@+id/imgView_bluetoothStatus"
57        android:layout_width="320dp"
58        android:layout_height="78dp"
59        android:scaleType="fitXY"
60        app:layout_constraintBottom_toBottomOf="parent"
61        app:layout_constraintEnd_toEndOf="parent"
62        app:layout_constraintHorizontal_bias="0.493"
```

```
63      app:layout_constraintStart_toStartOf="parent"
64      app:layout_constraintTop_toBottomOf="@+id/imageView"
65      app:layout_constraintVertical_bias="0.01"
66      app:srcCompat="@drawable/bt_button" />
67
68 <TextView
69     android:id="@+id/textView_btSearch"
70     android:layout_width="215dp"
71     android:layout_height="wrap_content"
72     android:text="@string/bluetooth_search_message"
73     android:textAlignment="center"
74     android:maxLines="1"
75     android:textAppearance="@style/btButtonTextAppearance"
76     app:layout_constraintBottom_toBottomOf="@+id/imgView_bluetoothConfig"
77     app:layout_constraintEnd_toEndOf="@+id/imgView_bluetoothConfig"
78     app:layout_constraintHorizontal_bias="0.27"
79     app:layout_constraintStart_toStartOf="@+id/imgView_bluetoothConfig"
80     app:layout_constraintTop_toTopOf="@+id/imgView_bluetoothConfig"
81     app:layout_constraintVertical_bias="0.4" />
82
83 <TextView
84     android:id="@+id/textView_btStatus"
85     android:layout_width="215dp"
86     android:layout_height="wrap_content"
87     android:maxLines="1"
88     android:textAlignment="center"
89     android:text="@string/system_disconnected_message"
90     android:textAppearance="@style/btButtonTextAppearance"
91     android:textColor="@android:color/background_light"
92     app:layout_constraintBottom_toBottomOf="@+id/imgView_bluetoothStatus"
93     app:layout_constraintEnd_toEndOf="@+id/imgView_bluetoothStatus"
94     app:layout_constraintHorizontal_bias="0.26"
95     app:layout_constraintStart_toStartOf="@+id/imgView_bluetoothStatus"
96     app:layout_constraintTop_toTopOf="@+id/imgView_bluetoothStatus"
97     app:layout_constraintVertical_bias="0.422" />
98
99 <TextView
100    android:id="@+id/textView_btDialog"
101    android:layout_width="wrap_content"
102    android:layout_height="wrap_content"
103    android:text="Dispositivos disponibles"
104    android:textAppearance="@style/btDialogTextAppearance"
105    app:layout_constraintBottom_toBottomOf="@+id/imageView"
106    app:layout_constraintEnd_toEndOf="@+id/imageView"
107    app:layout_constraintHorizontal_bias="0.5"
108    app:layout_constraintStart_toStartOf="@+id/imageView"
109    app:layout_constraintTop_toTopOf="@+id/imageView"
110    app:layout_constraintVertical_bias="0.040" />
111
112 <ImageButton
113     android:id="@+id/imgBtn_btConfig"
114     android:layout_width="30dp"
115     android:layout_height="30dp"
116     android:background="?attr/selectableItemBackground"
117     android:onClick="configBtn_onClick"
118     android:scaleType="fitXY"
119     android:tint="@color/colorPrimary" 230
120     app:layout_constraintBottom_toTopOf="@+id/imageView"
121     app:layout_constraintEnd_toEndOf="@+id/toolbar"
122     app:layout_constraintHorizontal_bias="0.94"
123     app:layout_constraintStart_toStartOf="@+id/toolbar"
124     app:layout_constraintTop_toTopOf="@+id/toolbar"
125     app:layout_constraintVertical_bias="0.5"
```

```
126        app:srcCompat="@drawable/ic_config_icon" />
127
128    <ImageView
129        android:id="@+id/imgView_btStatusIcon"
130        android:layout_width="35dp"
131        android:layout_height="35dp"
132        android:layout_marginBottom="8dp"
133        android:scaleType="fitXY"
134        android:tint="@android:color/background_light"
135        app:layout_constraintBottom_toBottomOf="@+id/imgView_bluetoothStatus"
136        app:layout_constraintEnd_toEndOf="@+id/imgView_bluetoothStatus"
137        app:layout_constraintHorizontal_bias="0.934"
138        app:layout_constraintStart_toStartOf="@+id/imgView_bluetoothStatus"
139        app:layout_constraintTop_toTopOf="@+id/imgView_bluetoothStatus"
140        app:srcCompat="@drawable/ic_close_icon"
141
142    />
143
144    <androidx.recyclerview.widget.RecyclerView
145        android:id="@+id/deviceListRecyclerView"
146        android:layout_width="0dp"
147        android:layout_height="0dp"
148        android:clipToPadding="false"
149        android:scrollbarStyle="outsideOverlay"
150        app:layout_constraintBottom_toBottomOf="@+id/bluetoothList_BottomGuideLine"
151        app:layout_constraintEnd_toEndOf="@+id/bluetoothList_EndGuideLine"
152        app:layout_constraintStart_toStartOf="@+id/bluetoothList_StartGuideLine"
153        app:layout_constraintTop_toTopOf="@+id/bluetoothList_TopGuideLine"
154    />
155
156    <ImageView
157        android:id="@+id/imgView_btDiscovery"
158        android:layout_width="30dp"
159        android:layout_height="30dp"
160        android:scaleType="fitXY"
161        android:tint="@android:color/background_light"
162        app:layout_constraintBottom_toBottomOf="@+id/imgView_bluetoothConfig"
163        app:layout_constraintEnd_toEndOf="@+id/imgView_bluetoothConfig"
164        app:layout_constraintHorizontal_bias="0.925"
165        app:layout_constraintStart_toStartOf="@+id/imgView_bluetoothConfig"
166        app:layout_constraintTop_toTopOf="@+id/imgView_bluetoothConfig"
167        app:layout_constraintVertical_bias="0.42"
168        app:srcCompat="@drawable/ic_bluetooth_searching" />
169
170    <ProgressBar
171        android:id="@+id/loading_progressBar"
172        style="?android:attr/progressBarStyle"
173        android:layout_width="26dp"
174        android:layout_height="26dp"
175        android:scaleType="fitXY"
176
177        android:visibility="gone"
178        app:layout_constraintBottom_toBottomOf="@+id/imgView_bluetoothStatus"
179
180        app:layout_constraintEnd_toEndOf="@+id/imgView_bluetoothStatus"
181        app:layout_constraintHorizontal_bias="0.919"
182        app:layout_constraintStart_toStartOf="@+id/imgView_bluetoothStatus"
183        app:layout_constraintTop_toTopOf="@+id/imgView_bluetoothStatus"
184        app:layout_constraintVertical_bias="0.43" />
185
186    <androidx.constraintlayout.widget.Guideline
187        android:id="@+id/guideline"
188        android:layout_width="wrap_content"
```

```
189     android:layout_height="wrap_content"
190     android:orientation="vertical"
191     app:layout_constraintGuide_percent="0.02" />
192
193     <androidx.constraintlayout.widget.Guideline
194         android:id="@+id/bluetoothList_StartGuideLine"
195         android:layout_width="wrap_content"
196         android:layout_height="wrap_content"
197         android:orientation="vertical"
198         app:layout_constraintGuide_percent="0.12" />
199
200     <androidx.constraintlayout.widget.Guideline
201         android:id="@+id/bluetoothList_EndGuideLine"
202         android:layout_width="wrap_content"
203         android:layout_height="wrap_content"
204         android:orientation="vertical"
205         app:layout_constraintGuide_percent="0.88" />
206
207     <androidx.constraintlayout.widget.Guideline
208         android:id="@+id/guideline2"
209         android:layout_width="wrap_content"
210         android:layout_height="wrap_content"
211         android:orientation="vertical"
212         app:layout_constraintGuide_percent="0.98055553" />
213
214     <androidx.constraintlayout.widget.Guideline
215         android:id="@+id/guideline3"
216         android:layout_width="wrap_content"
217         android:layout_height="wrap_content"
218         android:orientation="horizontal"
219         app:layout_constraintGuide_percent="0.1" />
220
221     <androidx.constraintlayout.widget.Guideline
222         android:id="@+id/bluetoothList_TopGuideLine"
223         android:layout_width="wrap_content"
224         android:layout_height="wrap_content"
225         android:orientation="horizontal"
226         app:layout_constraintGuide_percent="0.17" />
227
228     <androidx.constraintlayout.widget.Guideline
229         android:id="@+id/bluetoothList_BottomGuideLine"
230         android:layout_width="wrap_content"
231         android:layout_height="wrap_content"
232         android:orientation="horizontal"
233         app:layout_constraintGuide_percent="0.688" />
234
235     <androidx.constraintlayout.widget.Guideline
236         android:id="@+id/guideline4"
237         android:layout_width="wrap_content"
238         android:layout_height="wrap_content"
239         android:orientation="horizontal"
240         app:layout_constraintGuide_percent="0.74" />
241
242
243 </androidx.constraintlayout.widget.ConstraintLayout>
```

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com
 /apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:id="@+id/relativeLayout"
6     android:layout_width="match_parent"
7     android:layout_height="90dp"
8     android:background="?android:attr/selectableItemBackground"
9     android:clickable="true"
10    android:focusable="true"
11    android:orientation="vertical"
12    android:paddingLeft="8dp"
13    android:paddingTop="8dp"
14    android:paddingRight="8dp"
15    android:paddingBottom="8dp">
16
17    <TextView
18        android:id="@+id/setting_title"
19        android:layout_width="250dp"
20        android:layout_height="wrap_content"
21        android:layout_marginTop="8dp"
22        android:layout_marginEnd="8dp"
23        android:fontFamily="@font/droid_sans_bold"
24        android:text="Lorem ipsum dolor"
25        android:textColor="@color/colorPrimary"
26        android:textSize="18sp"
27        android:textStyle="bold"
28        app:layout_constraintEnd_toEndOf="parent"
29        app:layout_constraintTop_toTopOf="parent"
30        tools:layout_conversion_wrapHeight="56"
31        tools:layout_conversion_wrapWidth="788" />
32
33    <TextView
34        android:id="@+id/setting_subtitle"
35        android:layout_width="250dp"
36        android:layout_height="wrap_content"
37        android:layout_marginEnd="8dp"
38        android:fontFamily="@font/droid_sans"
39        android:lines="2"
40        android:maxLines="2"
41        android:text="Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
 eiusmod tempor incididunt ut labore et dolore magna aliqua. "
42        app:layout_constraintEnd_toEndOf="parent"
43        app:layout_constraintTop_toBottomOf="@+id/setting_title"
44        tools:layout_conversion_wrapHeight="94"
45        tools:layout_conversion_wrapWidth="788" />
46
47    <ImageView
48        android:id="@+id/setting_icon"
49        android:layout_width="50dp"
50        android:layout_height="50dp"
51        android:scaleType="fitXY"
52        app:layout_constraintBottom_toBottomOf="parent"
53        app:layout_constraintEnd_toStartOf="@+id/setting_title"
54        app:layout_constraintHorizontal_bias="0.5"
55        app:layout_constraintStart_toStartOf="parent"
56        app:layout_constraintTop_toTopOf="parent"
57        app:srcCompat="@drawable/ic_setting_contacts_and_communication"
58        tools:layout_conversion_wrapHeight="158"
59        tools:layout_conversion_wrapWidth="158" />
60
61 </androidx.constraintlayout.widget.ConstraintLayout>
```

Archivo - information\_help\_layout.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com
 /apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent">
7
8     <androidx.appcompat.widget.Toolbar
9         android:id="@+id/toolbar"
10        android:layout_width="0dp"
11        android:layout_height="57dp"
12        android:background="@color/toolBarColor"
13        android:minHeight="?attr/actionBarSize"
14        android:outlineAmbientShadowColor="@android:color/background_light"
15        android:outlineSpotShadowColor="@android:color/background_light"
16        android:theme="?attr actionBarTheme"
17        app:layout_constraintEnd_toEndOf="parent"
18        app:layout_constraintHorizontal_bias="0.0"
19        app:layout_constraintStart_toStartOf="parent"
20        app:layout_constraintTop_toTopOf="parent"
21        app:subtitleTextColor="@android:color/background_light"
22        app:title="@string/information_help_title"
23        app:titleTextAppearance="@style/toolBarTextAppearance"
24        app:titleTextColor="@color/colorPrimary" />
25
26
27
28 </androidx.constraintlayout.widget.ConstraintLayout>
```

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com
 /apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent">
7
8
9
10    <androidx.appcompat.widget.Toolbar
11        android:id="@+id/toolbar"
12        android:layout_width="0dp"
13        android:layout_height="57dp"
14        android:background="@color/toolBarColor"
15        android:minHeight="?attr/actionBarSize"
16        android:outlineAmbientShadowColor="@android:color/background_light"
17        android:outlineSpotShadowColor="@android:color/background_light"
18        android:theme="?attr actionBarTheme"
19        app:title="@string/commands_settings_title"
20        app:layout_constraintEnd_toEndOf="parent"
21        app:layout_constraintHorizontal_bias="0.0"
22        app:layout_constraintStart_toStartOf="parent"
23        app:layout_constraintTop_toTopOf="parent"
24        app:subtitleTextColor="@android:color/background_light"
25        app:titleTextColor="@color/colorPrimary"
26        app:titleTextAppearance="@style/toolBarTextAppearance"/>
27
28
29
30
31 </androidx.constraintlayout.widget.ConstraintLayout>
```

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com
 /apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     android:orientation="vertical">
8
9
10    <androidx.appcompat.widget.Toolbar
11        android:id="@+id/toolbar"
12        android:layout_width="match_parent"
13        android:layout_height="wrap_content"
14        android:background="@color/toolBarColor"
15        android:minHeight="?attr/actionBarSize"
16        android:outlineAmbientShadowColor="@android:color/background_light"
17        android:outlineSpotShadowColor="@android:color/background_light"
18        android:theme="?attr actionBarTheme"
19        app:layout_constraintEnd_toEndOf="parent"
20        app:layout_constraintHorizontal_bias="0.0"
21        app:layout_constraintStart_toStartOf="parent"
22        app:layout_constraintTop_toTopOf="parent"
23        app:subtitleTextColor="@android:color/background_light"
24        app:title="@string/developer_console_activity_title"
25        app:titleTextAppearance="@style/toolBarTextAppearance"
26        app:titleTextColor="@color/colorPrimary">
27
28
29
30    </androidx.appcompat.widget.Toolbar>
31
32    <ImageButton
33        android:id="@+id/clearTerminalButton"
34        android:layout_width="45dp"
35        android:layout_height="45dp"
36        android:background="?attr/selectableItemBackground"
37        android:onClick="onClickClearButton"
38
39        android:scaleType="centerInside"
40        app:layout_constraintHorizontal_bias="0.96"
41        app:layout_constraintVertical_bias="0.01"
42
43        app:layout_constraintBottom_toBottomOf="parent"
44        app:layout_constraintEnd_toEndOf="parent"
45        app:layout_constraintStart_toStartOf="parent"
46        app:layout_constraintTop_toTopOf="parent"
47
48        app:srcCompat="@drawable/ic_delete_sweep_24px" />
49
50    <androidx.constraintlayout.widget.ConstraintLayout
51        android:id="@+id/main_area"
52        android:layout_width="0dp"
53        android:layout_height="0dp"
54        android:layout_marginLeft="8dp"
55        android:layout_marginRight="8dp"
56        app:layout_constraintBottom_toBottomOf="parent"
57        app:layout_constraintLeft_toLeftOf="parent"
58        app:layout_constraintRight_toRightOf="parent"
59        app:layout_constraintTop_toBottomOf="@+id/toolbar">
60
61        <ScrollView
62            android:id="@+id/scrollViewTerminal"
```

```
63         android:layout_width="match_parent"
64         android:layout_height="0dp"
65         app:layout_constraintBottom_toTopOf="@+id/editTextTerminal"
66         app:layout_constraintLeft_toLeftOf="parent"
67         app:layout_constraintRight_toRightOf="parent"
68         app:layout_constraintTop_toTopOf="parent"
69         android:fastScrollEnabled="true"
70         android:fastScrollAlwaysVisible="true"
71         android:fadeScrollbars="false">
72
73     <LinearLayout
74         android:id="@+id/terminalLinearLayout"
75         android:layout_width="match_parent"
76         android:layout_height="wrap_content"
77         android:orientation="vertical" />
78 </ScrollView>
79
80 <EditText
81     android:id="@+id/editTextTerminal"
82     android:layout_width="0dp"
83     android:layout_height="50dp"
84     android:ems="10"
85
86     android:fontFamily="@font/roboto_mono"
87
88
89     android:hint="Comando"
90     android:inputType="textPersonName"
91     app:layout_constraintBottom_toBottomOf="parent"
92     app:layout_constraintHorizontal_bias="0.0"
93
94
95     app:layout_constraintHorizontal_weight="8"
96     app:layout_constraintLeft_toLeftOf="parent"
97     app:layout_constraintRight_toLeftOf="@+id/enterButton"
98     app:layout_constraintTop_toBottomOf="@+id/scrollViewTerminal" />
99
100 <ImageButton
101     android:id="@+id/enterButton"
102     android:layout_width="0dp"
103     android:layout_height="50dp"
104     android:background="?attr/selectableItemBackground"
105     android:onClick="onClickEnterButton"
106     android:scaleType="centerInside"
107
108     app:layout_constraintBottom_toBottomOf="parent"
109     app:layout_constraintHorizontal_weight="2"
110     app:layout_constraintLeft_toRightOf="@+id/editTextTerminal"
111     app:layout_constraintRight_toRightOf="parent"
112     app:layout_constraintTop_toBottomOf="@+id/scrollViewTerminal"
113     app:srcCompat="@drawable/ic_subdirectory_arrow_left_24px" />
114
115 </androidx.constraintlayout.widget.ConstraintLayout>
116
117
118
119
120
121 </androidx.constraintlayout.widget.ConstraintLayout>
```

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com
 /apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent">
7
8     <androidx.appcompat.widget.Toolbar
9         android:id="@+id/toolbar"
10        android:layout_width="0dp"
11        android:layout_height="57dp"
12        android:background="@color/toolBarColor"
13        android:minHeight="?attr/actionBarSize"
14        android:outlineAmbientShadowColor="@android:color/background_light"
15        android:outlineSpotShadowColor="@android:color/background_light"
16        android:theme="?attr actionBarTheme"
17        app:layout_constraintEnd_toEndOf="parent"
18        app:layout_constraintHorizontal_bias="0.0"
19        app:layout_constraintStart_toStartOf="parent"
20        app:layout_constraintTop_toTopOf="parent"
21        app:subtitleTextColor="@android:color/background_light"
22        app:title="@string/bluetooth_settings_title"
23        app:titleTextColor="@color/colorPrimary"
24        app:titleTextAppearance="@style/toolBarTextAppearance"/>
25
26
27
28 </androidx.constraintlayout.widget.ConstraintLayout>
```

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com
 /apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:id="@+id/relativeLayout"
6     android:layout_width="wrap_content"
7     android:layout_height="wrap_content"
8     android:background="?android:attr/selectableItemBackground"
9     android:clickable="true"
10    android:focusable="true"
11    android:orientation="vertical"
12    android:paddingLeft="8dp"
13    android:paddingTop="8dp"
14    android:paddingRight="8dp"
15    android:paddingBottom="8dp">
16
17    <TextView
18        android:id="@+id/textView_bluetooth_device_name"
19        android:layout_width="wrap_content"
20        android:layout_height="0dp"
21        android:layout_marginStart="8dp"
22        android:layout_marginTop="2dp"
23        android:layout_marginEnd="8dp"
24        android:text="Lorem ipsum dolor"
25        android:textAppearance="@style/btDeviceListAppearance"
26        android:textColor="@color/colorPrimary"
27        app:layout_constraintEnd_toEndOf="parent"
28        app:layout_constraintHorizontal_bias="0.03"
29        app:layout_constraintStart_toStartOf="parent"
30        app:layout_constraintTop_toTopOf="parent"
31        tools:layout_conversion_wrapHeight="56"
32        tools:layout_conversion_wrapWidth="788" />
33
34    <ImageView
35        android:id="@+id/imageView_bluetooth_device_state_icon"
36        android:layout_width="25dp"
37        android:layout_height="25dp"
38        android:layout_marginStart="215dp"
39        android:scaleType="fitXY"
40        android:tint="@color/colorPrimary"
41        app:layout_constraintStart_toStartOf="parent"
42        app:layout_constraintTop_toTopOf="parent"
43        app:srcCompat="@drawable/ic_close_icon" />
44
45 </androidx.constraintlayout.widget.ConstraintLayout>
```

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com
 /apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent">
7
8     <androidx.appcompat.widget.Toolbar
9         android:id="@+id/toolbar"
10        android:layout_width="0dp"
11        android:layout_height="57dp"
12        android:background="@color/toolBarColor"
13        android:minHeight="?attr/actionBarSize"
14        android:outlineAmbientShadowColor="@android:color/background_light"
15        android:outlineSpotShadowColor="@android:color/background_light"
16        android:theme="?attr actionBarTheme"
17        app:layout_constraintEnd_toEndOf="parent"
18        app:layout_constraintHorizontal_bias="0.0"
19        app:layout_constraintStart_toStartOf="parent"
20        app:layout_constraintTop_toTopOf="parent"
21        app:subtitleTextColor="@android:color/background_light"
22        app:title="@string/internet_connectivity_title"
23        app:titleTextAppearance="@style/toolBarTextAppearance"
24        app:titleTextColor="@color/colorPrimary" />
25
26
27
28 </androidx.constraintlayout.widget.ConstraintLayout>
```

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com
 /apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent">
7
8     <androidx.appcompat.widget.Toolbar
9         android:id="@+id/toolbar"
10        android:layout_width="0dp"
11        android:layout_height="57dp"
12        android:background="@color/toolBarColor"
13        android:minHeight="?attr/actionBarSize"
14        android:outlineAmbientShadowColor="@android:color/background_light"
15        android:outlineSpotShadowColor="@android:color/background_light"
16        android:theme="?attr actionBarTheme"
17        app:title="@string/contacts_communication_title"
18        app:layout_constraintEnd_toEndOf="parent"
19        app:layout_constraintHorizontal_bias="0.0"
20        app:layout_constraintStart_toStartOf="parent"
21        app:layout_constraintTop_toTopOf="parent"
22        app:subtitleTextColor="@android:color/background_light"
23        app:titleTextColor="@color/colorPrimary"
24        app:titleTextAppearance="@style/toolBarTextAppearance"/>
25
26
27
28 </androidx.constraintlayout.widget.ConstraintLayout>
```

Archivo - appearance\_miscellaneous\_layout.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com
 /apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent">
7
8     <androidx.appcompat.widget.Toolbar
9         android:id="@+id/toolbar"
10        android:layout_width="0dp"
11        android:layout_height="57dp"
12        android:background="@color/toolBarColor"
13        android:minHeight="?attr/actionBarSize"
14        android:outlineAmbientShadowColor="@android:color/background_light"
15        android:outlineSpotShadowColor="@android:color/background_light"
16        android:theme="?attr actionBarTheme"
17        app:layout_constraintEnd_toEndOf="parent"
18        app:layout_constraintHorizontal_bias="0.0"
19        app:layout_constraintStart_toStartOf="parent"
20        app:layout_constraintTop_toTopOf="parent"
21        app:subtitleTextColor="@android:color/background_light"
22        app:title="@string/appearance_miscellaneous_title"
23        app:titleTextColor="@color/colorPrimary"
24        app:titleTextAppearance="@style/toolBarTextAppearance"/>
25
26
27
28 </androidx.constraintlayout.widget.ConstraintLayout>
```

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com
 /apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:id="@+id/relativeLayout"
6     android:layout_width="match_parent"
7     android:layout_height="60dp"
8     android:background="?android:attr/selectableItemBackground"
9     android:clickable="true"
10    android:focusable="true"
11    android:orientation="vertical"
12    android:paddingLeft="4dp"
13    android:paddingTop="4dp"
14    android:paddingRight="4dp"
15    android:paddingBottom="4dp">
16
17    <TextView
18        android:id="@+id/command_state_textView"
19        android:layout_width="60dp"
20        android:layout_height="wrap_content"
21        android:fontFamily="@font/roboto"
22        android:text="Encendido"
23        android:textAlignment="center"
24        android:textColor="@color/colorPrimary"
25        android:textSize="9sp"
26        app:layout_constraintBottom_toBottomOf="parent"
27        app:layout_constraintStart_toEndOf="@+id/guideline10"
28        app:layout_constraintTop_toTopOf="parent"
29        app:layout_constraintVertical_bias="1.0" />
30
31    <TextView
32        android:id="@+id/command_title_textView"
33        android:layout_width="wrap_content"
34        android:layout_height="wrap_content"
35        android:layout_marginStart="8dp"
36        android:layout_marginEnd="8dp"
37        android:fontFamily="@font/roboto"
38        android:text="Velador numero 3"
39        android:textColor="@color/colorPrimary"
40        android:textSize="18sp"
41        app:layout_constraintBottom_toBottomOf="parent"
42        app:layout_constraintEnd_toStartOf="@+id/arrow_list_imageButton"
43        app:layout_constraintHorizontal_bias="0.0"
44        app:layout_constraintStart_toEndOf="@+id/command_icon"
45        app:layout_constraintTop_toTopOf="parent"
46        app:layout_constraintVertical_bias="0.514"
47        tools:layout_conversion_wrapHeight="56"
48        tools:layout_conversion_wrapWidth="788" />
49
50    <ImageView
51        android:id="@+id/command_icon"
52        android:layout_width="45dp"
53        android:layout_height="45dp"
54        android:scaleType="fitXY"
55        android:tint="@color/colorPrimary"
56        app:layout_constraintBottom_toBottomOf="parent"
57        app:layout_constraintStart_toEndOf="@+id/guideline_main"
58        app:layout_constraintTop_toTopOf="parent"
59        app:layout_constraintVertical_bias="0.5"
60        app:srcCompat="@drawable/icons_board" />
61
62    <ProgressBar
```

```
63        android:id="@+id/command_state_progressBar"
64        style="?android:progressBarStyleHorizontal"
65        android:layout_width="30dp"
66        android:layout_height="30dp"
67        android:indeterminate="false"
68        android:progress="0"
69        android:max="100"
70        android:secondaryProgress="0"
71        android:tint="@color/colorPrimary"
72        android:rotation="270"
73        app:layout_constraintBottom_toBottomOf="parent"
74        app:layout_constraintStart_toEndOf="@+id/guideline7"
75        app:layout_constraintTop_toTopOf="parent"
76        app:layout_constraintVertical_bias="0.5"
77        android:progressDrawable="@drawable/progressbar_onboarding_view"
78        android:layout_gravity="center"/>
79
80    <androidx.constraintlayout.widget.Guideline
81        android:id="@+id/guideline_main"
82        android:layout_width="wrap_content"
83        android:layout_height="wrap_content"
84        android:orientation="vertical"
85        app:layout_constraintGuide_percent="0.15" />
86
87    <androidx.constraintlayout.widget.Guideline
88        android:id="@+id/guideline7"
89        android:layout_width="wrap_content"
90        android:layout_height="wrap_content"
91        android:orientation="vertical"
92        app:layout_constraintGuide_percent="0.88" />
93
94    <androidx.constraintlayout.widget.Guideline
95        android:id="@+id/guideline10"
96        android:layout_width="wrap_content"
97        android:layout_height="wrap_content"
98        android:orientation="vertical"
99        app:layout_constraintGuide_percent="0.8375" />
100
101   <ImageButton
102       android:id="@+id/arrow_list_imageButton"
103       android:layout_width="35dp"
104       android:layout_height="35dp"
105       android:background="?attr/selectableItemBackground"
106       android:scaleType="fitXY"
107       android:tint="@color/colorPrimary"
108       android:visibility="gone"
109       app:layout_constraintBottom_toBottomOf="parent"
110       app:layout_constraintStart_toEndOf="@+id/guideline7"
111       app:layout_constraintTop_toTopOf="parent"
112       app:layout_constraintVertical_bias="0.5"
113       app:srcCompat="@drawable/icons_upward_arrow" />
114
115
116 </androidx.constraintlayout.widget.ConstraintLayout>
```

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com
 /apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:id="@+id/relativeLayout"
6     android:layout_width="match_parent"
7     android:layout_height="55dp"
8     android:background="?android:attr/selectableItemBackground"
9     android:clickable="true"
10    android:focusable="true"
11    android:orientation="vertical"
12    android:paddingLeft="8dp"
13    android:paddingTop="8dp"
14    android:paddingRight="8dp"
15    android:paddingBottom="8dp">
16
17    <TextView
18        android:id="@+id/parent_board_title_TextView"
19        android:layout_width="wrap_content"
20        android:layout_height="wrap_content"
21        android:fontFamily="@font/roboto"
22        android:text="Placa de mandos motrices"
23        android:textColor="@color/colorPrimary"
24        android:textSize="18sp"
25        app:layout_constraintBottom_toBottomOf="parent"
26        app:layout_constraintStart_toEndOf="@+id/guideline6"
27        app:layout_constraintTop_toTopOf="parent"
28        app:layout_constraintVertical_bias="0.5"
29        tools:layout_conversion_wrapHeight="56"
30        tools:layout_conversion_wrapWidth="788" />
31
32    <ImageView
33        android:id="@+id/parent_board_icon_ImageView"
34        android:layout_width="40dp"
35        android:layout_height="40dp"
36        android:scaleType="fitXY"
37        android:tint="@color/colorPrimary"
38        app:layout_constraintBottom_toBottomOf="parent"
39        app:layout_constraintStart_toEndOf="@+id/guideline_main"
40        app:layout_constraintTop_toTopOf="parent"
41        app:layout_constraintVertical_bias="0.5"
42        app:srcCompat="@drawable/icons_board" />
43
44    <androidx.constraintlayout.widget.Guideline
45        android:id="@+id/guideline_main"
46        android:layout_width="wrap_content"
47        android:layout_height="wrap_content"
48        android:orientation="vertical"
49        app:layout_constraintGuide_percent="0.04" />
50
51    <androidx.constraintlayout.widget.Guideline
52        android:id="@+id/guideline6"
53        android:layout_width="wrap_content"
54        android:layout_height="wrap_content"
55        android:orientation="vertical"
56        app:layout_constraintGuide_percent="0.17" />
57
58    <androidx.constraintlayout.widget.Guideline
59        android:id="@+id/guideline8"
60        android:layout_width="wrap_content"
61        android:layout_height="wrap_content"
62        android:orientation="vertical"
```

```
63     app:layout_constraintGuide_percent="0.90" />
64
65 <ImageButton
66     android:id="@+id/arrow_list_imageButton"
67     android:layout_width="35dp"
68     android:layout_height="35dp"
69     android:background="?attr/selectableItemBackground"
70     android:scaleType="fitXY"
71     android:tint="@color/colorPrimary"
72     app:layout_constraintBottom_toBottomOf="parent"
73     app:layout_constraintStart_toEndOf="@+id/guideline8"
74     app:layout_constraintTop_toTopOf="parent"
75     app:layout_constraintVertical_bias="0.5"
76     app:srcCompat="@drawable/icons_upward_arrow" />
77
78
79 </androidx.constraintlayout.widget.ConstraintLayout>
```

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com
 /apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent">
7
8     <androidx.appcompat.widget.Toolbar
9         android:id="@+id/toolbar"
10        android:layout_width="0dp"
11        android:layout_height="57dp"
12        android:background="@color/toolBarColor"
13        app:titleTextColor="@color/colorPrimary"
14        android:minHeight="?attr/actionBarSize"
15        android:outlineAmbientShadowColor="@android:color/background_light"
16        android:outlineSpotShadowColor="@android:color/background_light"
17        android:theme="?attr actionBarTheme"
18        app:title="@string/voice_recognition_settings_title"
19        app:layout_constraintEnd_toEndOf="parent"
20        app:layout_constraintHorizontal_bias="0.0"
21        app:layout_constraintStart_toStartOf="parent"
22        app:layout_constraintTop_toTopOf="parent"
23        app:subtitleTextColor="@android:color/background_light"
24        app:titleTextAppearance="@style/toolBarTextAppearance"/>
25
26     <Button
27         android:id="@+id/set_voice_btn"
28         android:layout_width="172dp"
29         android:layout_height="53dp"
30         android:layout_marginStart="8dp"
31         android:layout_marginTop="180dp"
32         android:layout_marginEnd="8dp"
33         android:layout_marginBottom="8dp"
34         android:onClick="setVoiceOnClick"
35         android:text="CAMBIAR VOZ"
36         app:layout_constraintBottom_toBottomOf="parent"
37         app:layout_constraintEnd_toEndOf="parent"
38         app:layout_constraintStart_toStartOf="parent"
39         app:layout_constraintTop_toBottomOf="@+id/toolbar"
40         app:layout_constraintVertical_bias="0.349" />
41
42     <Button
43         android:id="@+id/speech_test_btn"
44         android:layout_width="172dp"
45         android:layout_height="53dp"
46         android:layout_marginStart="8dp"
47         android:layout_marginTop="180dp"
48         android:layout_marginEnd="8dp"
49         android:layout_marginBottom="8dp"
50         android:onClick="speechTestOnClick"
51         android:text="PRUEBA"
52         app:layout_constraintBottom_toBottomOf="parent"
53         app:layout_constraintEnd_toEndOf="parent"
54         app:layout_constraintStart_toStartOf="parent"
55         app:layout_constraintTop_toBottomOf="@+id/toolbar"
56         app:layout_constraintVertical_bias="0.957" />
57
58     <EditText
59         android:id="@+id/phrase_editText"
60         android:layout_width="wrap_content"
61         android:layout_height="wrap_content"
62         android:layout_marginStart="8dp"
```

```
63      android:layout_marginTop="52dp"
64      android:layout_marginEnd="8dp"
65      android:layout_marginBottom="8dp"
66      android:ems="10"
67      android:inputType="textPersonName"
68      android:shadowColor="@color/colorPrimary"
69      android:text="Frase"
70      app:layout_constraintBottom_toBottomOf="parent"
71      app:layout_constraintEnd_toEndOf="parent"
72      app:layout_constraintHorizontal_bias="0.503"
73      app:layout_constraintStart_toStartOf="parent"
74      app:layout_constraintTop_toBottomOf="@+id/toolbar"
75      app:layout_constraintVertical_bias="0.08" />
76
77 <ProgressBar
78     android:id="@+id/progressBar"
79     style="?android:attr/progressBarStyle"
80     android:layout_width="wrap_content"
81     android:layout_height="wrap_content"
82     android:layout_marginStart="8dp"
83     android:layout_marginTop="444dp"
84     android:layout_marginEnd="8dp"
85     android:layout_marginBottom="8dp"
86     app:layout_constraintBottom_toBottomOf="parent"
87     app:layout_constraintEnd_toEndOf="parent"
88     app:layout_constraintStart_toStartOf="parent"
89     app:layout_constraintTop_toBottomOf="@+id/toolbar"
90     app:layout_constraintVertical_bias="0.0" />
91
92
93 </androidx.constraintlayout.widget.ConstraintLayout>
```

JAVA.pdf