Escuela Universitaria
Politécnica - La Almunia
Centro adscrito
Universidad Zaragoza

ESCUELA UNIVERSITARIA POLITÉCNICA
DE LA ALMUNIA DE DOÑA GODINA (ZARAGOZA)

ANEXOS

Dispositivo IoT para la monitorización de parámetros fisiológicos

IoT device for monitoring physiological parameters

242.21.29

Autor: Elioenay Pérez López

Director: Dr. David Asiain Ansorena

Fecha: 06 2022

Página intencionadamente en blanco.

INDICE DE CONTENIDO

1. CÓDIGO FUENTE

1.1. LIBRERÍA DEL SENSOR MAX30205 EN C.

Archivo MAX30205.h de la librería creada para el sensor MAX30205 en C:

```
#ifndef __MAX30205_H_
#define __MAX30205_H_

#define MAX30205_ADDRESS        0x90  //Dirreccion de 8 bit

#define MAX30205_TEMPERATURE    0x00
#define MAX30205_CONFIGURATION  0x01
#define MAX30205_THYST          0x02
#define MAX30205_TOS            0x03


  union MAX_Configuration{
        uint8_t all;
        struct BitField_s{
              uint8_t shutdown    : 1;
              uint8_t comp_int    : 1;
              uint8_t os_polarity : 1;
              uint8_t fault_queue : 2;
              uint8_t data_format : 1;
              uint8_t timeout     : 1;
              uint8_t one_shot    : 1;
         }bits;
  } max_Configuration_u;


  float MAX_temperature;
  uint8_t MAX_sensorAddress;
  void  MAX_shutdown(void);
  void  MAX_begin(void);
  float MAX_getTemperature(void);
  uint16_t MAX_readConfiguration(void);

  void MAX_writeRegister(uint8_t subAddress, uint16_t value);
  void MAX_readRegister(uint8_t subAddress, uint16_t *value);

#endif
```

Archivo MAX30205.c de la librería creada para el sensor MAX30205 en C:

```
#include "main.h"
#include "MAX30205.h"

uint8_t MAX_sensorAddress = MAX30205_ADDRESS;


float MAX_getTemperature(void){
      max_Configuration_u.bits.shutdown = 1;
      max_Configuration_u.bits.one_shot = 1;

      uint16_t local_config = (0x00FF & max_Configuration_u.all);
      MAX_writeRegister(MAX30205_CONFIGURATION, local_config);

      HAL_Delay(60);      //50 ms de conversión según datasheet

      uint16_t readRaw;
      MAX_readRegister(MAX30205_TEMPERATURE, &readRaw);
     MAX_temperature = readRaw * 0.00390625;

      return  MAX_temperature;
}



uint16_t MAX_readConfiguration(void){
    uint16_t data;
    MAX_readRegister(MAX30205_CONFIGURATION , &data);

    return data;
}



void MAX_shutdown(void){
  uint16_t reg;
  MAX_readRegister(MAX30205_CONFIGURATION, &reg);
  MAX_writeRegister(MAX30205_CONFIGURATION, reg | 0x80);
}



void MAX_begin(void){
      max_Configuration_u.bits.shutdown = 1;
      max_Configuration_u.bits.comp_int = 0;
```

```
      max_Configuration_u.bits.os_polarity = 0;
      max_Configuration_u.bits.fault_queue = 0;
      max_Configuration_u.bits.data_format = 0;
      max_Configuration_u.bits.timeout = 1;
      max_Configuration_u.bits.one_shot = 1;

      uint16_t local_config = (0x00FF & max_Configuration_u.all);
      MAX_writeRegister(MAX30205_CONFIGURATION, local_config);
}


void MAX_writeRegister(uint8_t subAddress, uint16_t value) {
  uint8_t hi = ((value >> 8) & 0xFF);
  uint8_t lo = (value & 0xFF);
  uint8_t cmdData[3] = {subAddress, hi, lo};

  HAL_I2C_Master_Transmit(&hi2c1, MAX_sensorAddress, cmdData, 3,
100);
}


void MAX_readRegister(uint8_t subAddress, uint16_t *value) {
      uint8_t data[2];
      uint8_t cmdData[1] = {subAddress};

      HAL_I2C_Master_Transmit(&hi2c1, MAX_sensorAddress, cmdData,
1, 100);
      HAL_I2C_Master_Receive(&hi2c1, MAX_sensorAddress, data, 2,
500);

      *value = ((uint16_t)data[0] << 8) | data[1];

}
```

1.2. CÓDIGO DE APLICACIÓN LORAWAN

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************
  * @file    lora_app.c
  * @author  MCD Application Team
  * @brief   Application of the LRWAN Middleware
  ******************************************************************
  * @attention
```

```
  *
  * Copyright (c) 2022 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
  ******************************************************************
  */
/* USER CODE END Header */

/* Includes ------------------------------------------------------------
--------*/
#include "platform.h"
#include "Region.h" /* Needed for LORAWAN_DEFAULT_DATA_RATE */
#include "sys_app.h"
#include "lora_app.h"
#include "stm32_seq.h"
#include "stm32_timer.h"
#include "utilities_def.h"
#include "lora_app_version.h"
#include "lorawan_version.h"
#include "subghz_phy_version.h"
#include "lora_info.h"
#include "LmHandler.h"
#include "stm32_lpm.h"
#include "adc_if.h"
#include "sys_conf.h"
#include "CayenneLpp.h"
#include "sys_sensors.h"

/* USER CODE BEGIN Includes */

#include <stdio.h>
#include "MAX30205.h"

/* USER CODE END Includes */

/* External variables ---------------------------------------------------
---------*/
/* USER CODE BEGIN EV */


int8_t snr_recibido = 0;            //Para enviar los parámetros
int8_t rssi_recibido = 0;


/* USER CODE END EV */
```

```
/* Private typedef -----------------------------------------------------
--------*/
/**
  * @brief LoRa State Machine states
  */
typedef enum TxEventType_e
{
  /**
    * @brief Appdata Transmission issue based on timer every
TxDutyCycleTime
    */
  TX_ON_TIMER,
  /**
    * @brief Appdata Transmission external event plugged on
OnSendEvent( )
    */
  TX_ON_EVENT
  /* USER CODE BEGIN TxEventType_t */

  /* USER CODE END TxEventType_t */
} TxEventType_t;

/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

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

/* USER CODE END PD */

/* Private macro -------------------------------------------------------
--------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private function prototypes -----------------------------------------
--------*/
/**
  * @brief  LoRa End Node send request
  */
static void SendTxData(void);

/**
  * @brief  TX timer callback function
  * @param  context ptr of timer context
  */
static void OnTxTimerEvent(void *context);

/**
```

```
  * @brief  join event callback function
  * @param  joinParams status of join
  */
static void OnJoinRequest(LmHandlerJoinParams_t *joinParams);

/**
  * @brief  tx event callback function
  * @param  params status of last Tx
  */
static void OnTxData(LmHandlerTxParams_t *params);

/**
  * @brief callback when LoRa application has received a frame
  * @param appData data received in the last Rx
  * @param params status of last Rx
  */
static void OnRxData(LmHandlerAppData_t *appData, LmHandlerRxParams_t
*params);

/*!
 * Will be called each time a Radio IRQ is handled by the MAC layer
 *
 */
static void OnMacProcessNotify(void);

/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private variables ---------------------------------------------------
--------*/
static ActivationType_t ActivationType =
LORAWAN_DEFAULT_ACTIVATION_TYPE;

/**
  * @brief LoRaWAN handler Callbacks
  */
static LmHandlerCallbacks_t LmHandlerCallbacks =
{
  .GetBatteryLevel =           GetBatteryLevel,
  .GetTemperature =            GetTemperatureLevel,
  .GetUniqueId =               GetUniqueId,
  .GetDevAddr =                GetDevAddr,
  .OnMacProcess =              OnMacProcessNotify,
  .OnJoinRequest =             OnJoinRequest,
  .OnTxData =                  OnTxData,
  .OnRxData =                  OnRxData
};

/**
  * @brief LoRaWAN handler parameters
  */
```

```
static LmHandlerParams_t LmHandlerParams =
{
  .ActiveRegion =             ACTIVE_REGION,
  .DefaultClass =             LORAWAN_DEFAULT_CLASS,
  .AdrEnable =                LORAWAN_ADR_STATE,
  .TxDatarate =               LORAWAN_DEFAULT_DATA_RATE,
  .PingPeriodicity =          LORAWAN_DEFAULT_PING_SLOT_PERIODICITY
};

/**
  * @brief Type of Event to generate application Tx
  */
static TxEventType_t EventType = TX_ON_TIMER;

/**
  * @brief Timer to handle the application Tx
  */
static UTIL_TIMER_Object_t TxTimer;

/* USER CODE BEGIN PV */

static uint8_t AppDataBuffer[LORAWAN_APP_DATA_BUFFER_MAX_SIZE];
static LmHandlerAppData_t AppData = { 0, 0, AppDataBuffer };

/* USER CODE END PV */

/* Exported functions ---------------------------------------------
---------*/
/* USER CODE BEGIN EF */

/* USER CODE END EF */

void LoRaWAN_Init(void)
{
  /* USER CODE BEGIN LoRaWAN_Init_1 */

  /* USER CODE END LoRaWAN_Init_1 */

  UTIL_SEQ_RegTask((1 << CFG_SEQ_Task_LmHandlerProcess), UTIL_SEQ_RFU,
LmHandlerProcess);
  UTIL_SEQ_RegTask((1 << CFG_SEQ_Task_LoRaSendOnTxTimerOrButtonEvent),
UTIL_SEQ_RFU, SendTxData);
  /* Init Info table used by LmHandler*/
  LoraInfo_Init();

  /* Init the Lora Stack*/
  LmHandlerInit(&LmHandlerCallbacks);

  LmHandlerConfigure(&LmHandlerParams);

  /* USER CODE BEGIN LoRaWAN_Init_2 */
  /* USER CODE END LoRaWAN_Init_2 */
```

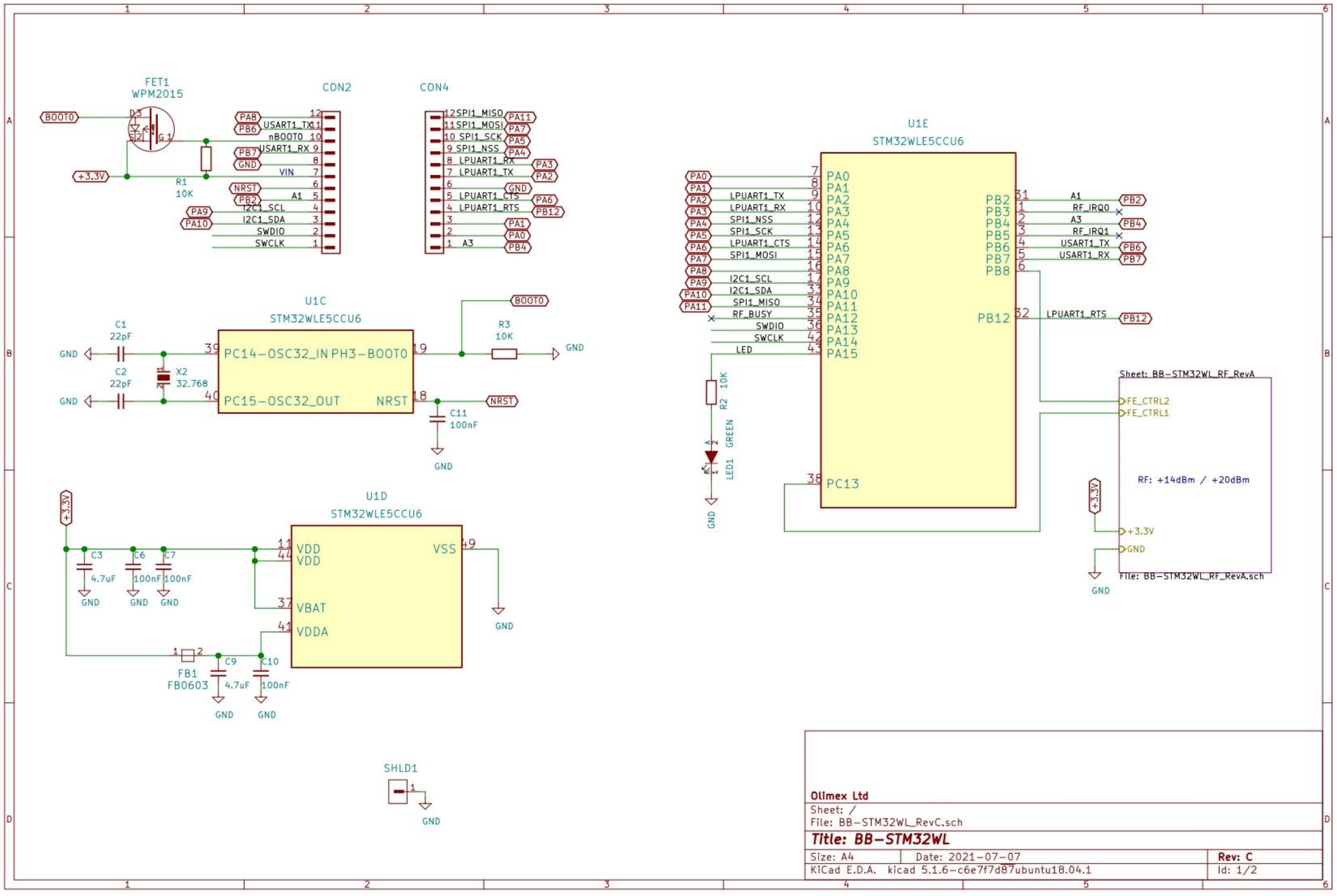
```
  LmHandlerJoin(ActivationType);

  if (EventType == TX_ON_TIMER)
  {
    /* send every time timer elapses */
    UTIL_TIMER_Create(&TxTimer,  0xFFFFFFFFU, UTIL_TIMER_ONESHOT,
OnTxTimerEvent, NULL);
    UTIL_TIMER_SetPeriod(&TxTimer,  APP_TX_DUTYCYCLE);
    UTIL_TIMER_Start(&TxTimer);
  }
  else
  {
    /* USER CODE BEGIN LoRaWAN_Init_3 */
    /* USER CODE END LoRaWAN_Init_3 */
  }

  /* USER CODE BEGIN LoRaWAN_Init_Last */

  /* USER CODE END LoRaWAN_Init_Last */
}

/* USER CODE BEGIN PB_Callbacks */

/* USER CODE END PB_Callbacks */

/* Private functions ---------------------------------------------
--------*/
/* USER CODE BEGIN PrFD */

/* USER CODE END PrFD */

static void OnRxData(LmHandlerAppData_t *appData, LmHandlerRxParams_t
*params)
{
  /* USER CODE BEGIN OnRxData_1 */

      if(appData->Port == LORAWAN_USER_APP_PORT){
            if(appData->Buffer[0] == 'X'){
                  snr_recibido = params->Snr;
                  rssi_recibido = params->Rssi;
            }
      }

  /* USER CODE END OnRxData_1 */
}

static void SendTxData(void)
{
  /* USER CODE BEGIN SendTxData_1 */

      float temperatura = 0;
```

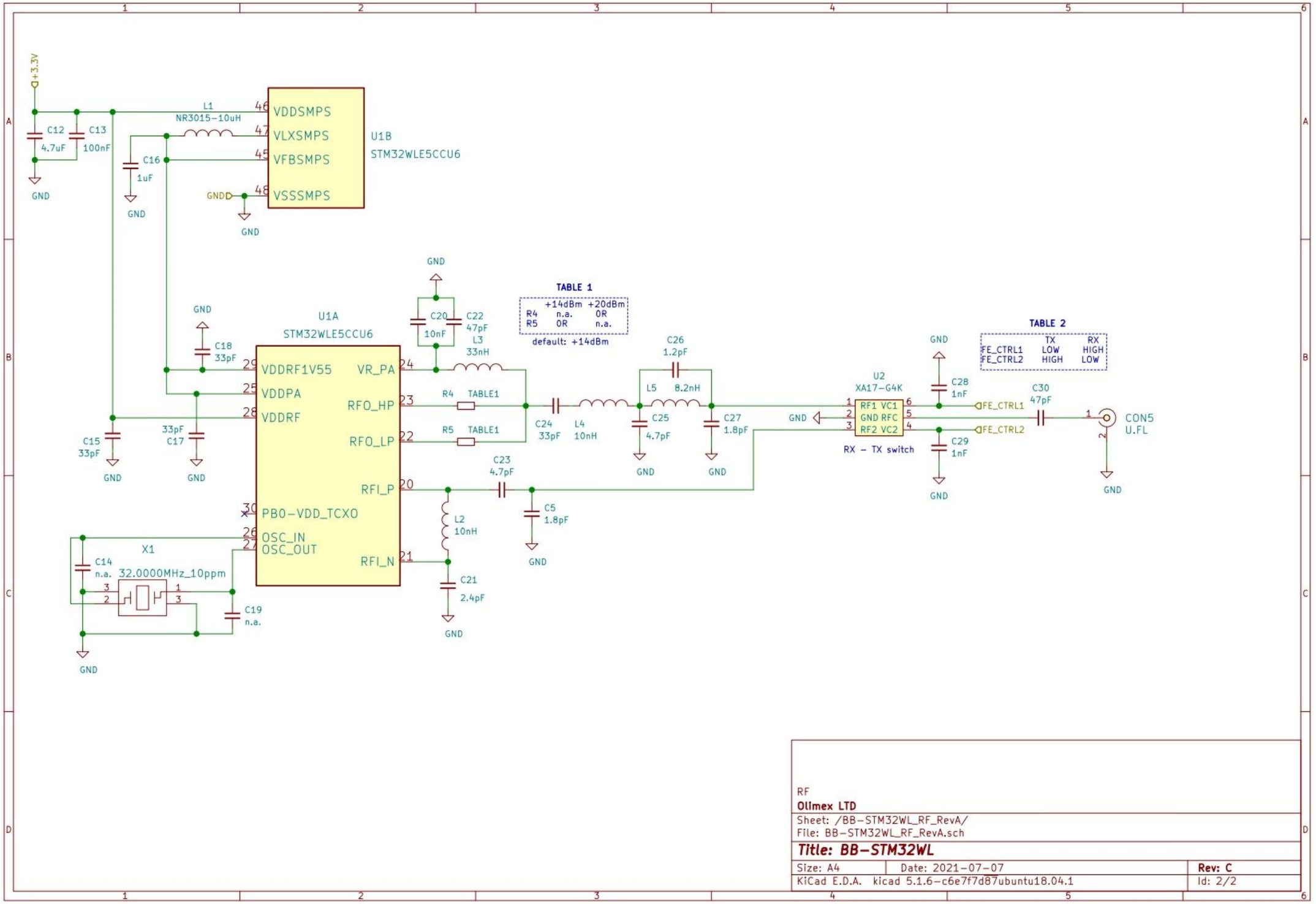
```
      UTIL_TIMER_Time_t nextTxIn = 0;

      temperatura = MAX_getTemperature();

      AppData.Port = LORAWAN_USER_APP_PORT;
      AppData.BufferSize = snprintf((char *) AppData.Buffer,
LORAWAN_APP_DATA_BUFFER_MAX_SIZE, "%.2f;%d:%d", temperatura,
snr_recibido, rssi_recibido);

      LmHandlerSend(&AppData, LORAWAN_DEFAULT_CONFIRMED_MSG_STATE,
nextTxIn, false);

  /* USER CODE END SendTxData_1 */
}

static void OnTxTimerEvent(void *context)
{
  /* USER CODE BEGIN OnTxTimerEvent_1 */

  /* USER CODE END OnTxTimerEvent_1 */
  UTIL_SEQ_SetTask((1 << CFG_SEQ_Task_LoRaSendOnTxTimerOrButtonEvent),
CFG_SEQ_Prio_0);

  /*Wait for next tx slot*/
  UTIL_TIMER_Start(&TxTimer);
  /* USER CODE BEGIN OnTxTimerEvent_2 */

  /* USER CODE END OnTxTimerEvent_2 */
}

/* USER CODE BEGIN PrFD_LedEvents */

/* USER CODE END PrFD_LedEvents */

static void OnTxData(LmHandlerTxParams_t *params)
{
  /* USER CODE BEGIN OnTxData_1 */
  /* USER CODE END OnTxData_1 */
}

static void OnJoinRequest(LmHandlerJoinParams_t *joinParams)
{
  /* USER CODE BEGIN OnJoinRequest_1 */
  /* USER CODE END OnJoinRequest_1 */
}

static void OnMacProcessNotify(void)
{
  /* USER CODE BEGIN OnMacProcessNotify_1 */

  /* USER CODE END OnMacProcessNotify_1 */
```

```
  UTIL_SEQ_SetTask((1 << CFG_SEQ_Task_LmHandlerProcess),
CFG_SEQ_Prio_0);

  /* USER CODE BEGIN OnMacProcessNotify_2 */

  /* USER CODE END OnMacProcessNotify_2 */
}
```

2. ESQUEMA ELECTRÓNICO

Esquema electrónico de la placa Olimex BB-STM32WL:




+3.3V
C12 4.7uF
C13 100nF
GND
L1 NR3015-10uH
C16 1uF
U1B STM32WLE5CCU6
VDDSMPS
VLXSMPS
VFBSMPS
VSSSMPS
GNDD
U1A STM32WLE5CCU6
VDDRF1V55
VDDPA
VDDRF
VR_PA
RFO_HP
RFO_LP
RFI_P
RFI_N
PB0-VDD_TCXO
OSC_IN
OSC_OUT
C18 33pF
C15 33pF
C17 33pF
C20 10nF
C22 47pF
L3 33nH
TABLE 1
+14dBm +20dBm
R4 n.a. 0R
R5 0R n.a.
default: +14dBm
R4 TABLE1
R5 TABLE1
C24 33pF
L4 10nH
C26 1.2pF
L5 8.2nH
C25 4.7pF
C27 1.8pF
C23 4.7pF
L2 10nH
C5 1.8pF
C21 2.4pF
X1
C14 n.a.
32.0000MHz_10ppm
C19 n.a.
U2 XA17-G4K
RF1 VC1
GND RFC
RF2 VC2
RX - TX switch
TABLE 2
TX RX
FE_CTRL1 LOW HIGH
FE_CTRL2 HIGH LOW
C28 1nF
C29 1nF
C30 47pF
FE_CTRL1
FE_CTRL2
CON5 U.FL
RF
Olimex LTD
Sheet: /BB-STM32WL_RF_RevA/
File: BB-STM32WL_RF_RevA.sch
Title: BB-STM32WL
Size: A4
Date: 2021-07-07
Rev: C
KiCad E.D.A. kicad 5.1.6-c6e7f7d87ubuntu18.04.1
Id: 2/2

Relación de documentos

(_) Memoria	109	páginas
(X) Anexos	17	páginas

La Almunia, a 20 de 06 de 2022

Firmado: Elioenay Pérez López