Universidad del País Vasco Euskal Herriko Unibertsitatea

GIPUZKOAKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE GIPUZKOA

GIPUZKOAKO INGENIARITZA ESKOLA

ESCUELA DE INGENIERÍA DE GIPUZKOA

GRADU AMAIERAKO LANA/ TRABAJO FIN DE GRADO

Grado en	Ingeniería Electrónica Industrial y Automática / Elektronika Industrial eta Automatikako	Gradua
Fecha	Julio de 2019	Data

Título del TFG / GRALaren titulua:

DESARROLLO DE MARCADORES ACTIVOS PARA APLICACIONES DE CAPTURA DE MOVIMIENTO 3D

Título del anexo / Anexoaren titulua:

PROGRAMAS DEL PROYECTO PARA LA COMUNICACIÓN CABLEADA E INALÁMBRICA MEDIANTE CODE COMPOSER STUDIO

Ikaslearen izen eta abizenak/ Nombre y apellidos del alumno/a:

Gorka Corral Malla

Zuzendariaren izen eta abizenak/ Nombre y apellidos del director/a:

Mikel Alberro Astarbe

Zuzendarikidearen izen eta abizenak/ Nombre y apellidos del codirector/a:

Guillermo Conde Salazar

ÍNDICE

1. Wired

```
/*********************************************************************************/
/*    Board: MSP430-CCRF
/*    Manufacture: OLIMEX
/*    COPYRIGHT (C) 2018
/*    Original Program Designed by: Penko Todorov Bozhkov
/*    Current Program Designed by:  Gorka Corral Malla
/*    Module Name:  Wired
/*    File   Name:  Wired.c
/*    Revision:     initial
/*    Date:         07.10.2018
/*    Built with Code Composer Studio C/C++ Compiler for MSP430
/*********************************************************************************/

// Target : CC430F5137
#include <cc430x513x.h>
#include  <intrinsics.h>

//Definitions
#define LED_On          P1OUT |= 0x01;
#define LED_Off         P1OUT &= (~0x01);
#define LED_Togg         P1OUT ^= 0x01;
#define LED_Chk         (P1IN & 0x01)

#define Pulse_On        P3OUT |= BIT1;
#define Pulse_Off       P3OUT &= (~BIT1);
#define Pulse_Togg       P3OUT ^= BIT1;
#define Pulse_Chk       (P3IN & BIT1)

#define Switch_On       P3OUT |= BIT6;
#define Start_Off       P3OUT &= (~BIT6);
#define Start_Togg       P3OUT ^= BIT6;
#define Start_Chk       (P3OUT & BIT6)

unsigned int j = 1;
unsigned int flag = 0;

/*********************************************************************************/
/*  Function name: delay
/*  Parameters
/*  Input   :  p
/*  Output  :  No
/*  Action: Simple delay
/*********************************************************************************/

void delay(volatile unsigned long p){
   while(p){p--;}
}
```

```
/*************************************************************************/
/*  Function name: ports_init
/*  Ports : P1, P2 and P3
/*  Input   :  P3.0 and P3.4
/*  Output  :  P3.1 and P3.6
/*  Action: Initialize all Port's directions and states
/*************************************************************************/

void ports_init(void){

  P1OUT = 0x00;
  P1DIR = 0x01;     // Led out
  P3IN = 0x00;
  P3OUT &= ~BIT0 & ~BIT1 & ~BIT4 & ~BIT6;   //Set directions to 0.
  P3DIR &= ~BIT0 & ~BIT4 | BIT1 | BIT6;           //Direct inputs and outputs
  P3REN = BIT0 | BIT1 | BIT4 | BIT6;
  P3SEL &= ~BIT0 & ~BIT1 & ~BIT4 & ~BIT6;
  P3DS &= ~BIT0 & ~BIT1 & ~BIT4 & ~BIT6;

  //P2DIR &= ~BIT3;
  //P2REN |= BIT3;
  //P2IES &= BIT3;
  //P2IFG = 0;
  //P2OUT |= BIT3;
  //P2IE  |= BIT3;
}

/*************************************************************************/
/*  Function name: timer1_A3_init
/*  Parameters
/*  Input   :  No
/*  Output  :  No
/*  Action: Initialize Timer1_A3 operation
/*************************************************************************/

//TIMER1 initialize - prescale:8 =>
// desired value:   2Hz, i.e. 1000000/8/62500(0xF424)

void timer1_A3_init(void){

  TA1CTL = 0x0004;   // Timer1_A3 clear.
  TA1CCTL0 = 0x0010; // Timer1_A3 Capture/compare 0 interrupt enable.
  TA1CCR0 = 0x0014;  // Set TACCR0 value to 5ms
  TA1CTL = 0x02D0;   // Selected: SMCLK, divider:8, Up mode.
  UCSCTL4 = 0000;    // XT1CLK clock selected
}
```

```
/**************************************************************************/
/*  Function name: init_devices
/*  Parameters
/*  Action: Initialize all peripherals
/**************************************************************************/

void init_devices(void){

 //1. Stop errant interrupts until set up
 _BIC_SR(GIE); // Disable interrupts during initialization process
 //2. Init System Clock
 // By default the FLL stabilizes MCLK and SMCLK to 1.048576 MHz and fDCO = 2.097152 MHz.

 //3. Init Peripherals
 ports_init();
 timer1_A3_init();

 _BIS_SR(GIE); // Global Interrupt enabled. Do this at the END of the initialization!!!!!!!!
 //all peripherals are now initialized
}

int main(void){
WDTCTL = WDTPW + WDTHOLD;                 // Stop WDT
  init_devices();

  while(1){

    TA1CTL = 0x0004;   // Timer1_A3 clear.
    P3OUT &= ~BIT1; //Pulse Init Off
    Start_Off; //Pulse Start Off
    TA1CCTL0 = 0x0000; // Timer1_A3 Capture/compare 0 interrupt disable.
    LED_Off;

    if(P3IN&BIT4){
       TA1CCTL0 = 0x0010; // Timer1_A3 Capture/compare 0 interrupt enable.
       TA1CCR0 = 0x0014;  // Set TACCR0 value to 5ms
       TA1CTL = 0x02D0;   // Selected: SMCLK, divider:8, Up mode.
       UCSCTL4 = 0000;    // XT1CLK clock selected
       Switch_On;
       LED_On;
       while(P3IN&BIT4){
          LED_Off;
          P3OUT |= BIT6;
       }
       while(!(P3IN&BIT4)){

          if(P3IN&BIT0 && !flag){
             TA1R = 0x0006; //reset sincro
             flag = 1;
             j=0;
```

```
        }

        if(!(P3IN&BIT0)){
          flag = 0;
        }
      }
      Start_Togg;
      LED_Off;
      while(P3IN&BIT4){;}
   }
  }
}

/*****************************************************************************/
/*  Timer1_A3 CC0 interrupt service routine.
/*****************************************************************************/

#pragma vector=TIMER1_A0_VECTOR
__interrupt void TIMER1_A0_Capture_Compare_ISR(void){

  if(j){
     LED_Togg;
     Pulse_Togg;
     j=0;
     TA1CCR0 = 0x000D;  // Set TACCR0 value to 3,2 ms
   }
  else{
       LED_Togg;
       Pulse_Togg;
       j=1;
       TA1CCR0 = 0x0014;  // Set TACCR0 value to 5,2 ms
     }
}
/*****************************************************************************/
```

2.1. WirelessMain

```
/*****************************************************************************/
/*    Board: MSP430-CCRF
/*    Manufacture: OLIMEX
/*    COPYRIGHT (C) 2011
/*    Original Program Designed by: Penko Todorov Bozhkov
/*    Current Program Designed by:  Gorka Corral Malla
/*    Module Name:  WirelessMain
/*    File   Name:  WirelessMain.c
/*    Revision:     initial
/*    Date:         07.10.2018
/*    Built with Code Composer Studio  C/C++ Compiler for MSP430
/*****************************************************************************/

// Target : CC430F5137
#include <cc430x513x.h>
#include  <intrinsics.h>
#include "RF_Connection.h"

//Definitions
#define LED_ON                  P1OUT |= 0x01;        P1DIR |= 0x01;
#define LED_OFF                 P1OUT &= (~0x01);   P1DIR |= 0x01;
#define LED_Togg                P1OUT ^= 0x01;       P1DIR |= 0x01;
#define LED_Chk                 (P1IN & 0x01)
#define timer1_A3_Stop_Mode     TA1CTL &= (~0x0030)
#define timer1_A3_Up_Mode       TA1CTL |= (0x0010)

/*****************************************************************************/
/*  Function name: delay
/*  Parameters
/*  Input   :  p
/*  Output  :  No
/*  Action: Simple delay
/*****************************************************************************/

void delay(volatile unsigned long p){
   while(p){p--;}
}

/*****************************************************************************/
/*  Function name: ports_init (real init in file: RF_Connection.c)
/*  Parameters
/*  Action: Initialize all Port's directions and states
/*****************************************************************************/

void ports_init(void){

  P1OUT = 0x00;
  P1DIR = 0x01;   // Led out
```

```
  P1REN = 0x00;
  P2OUT = 0x00;
  P2DIR = 0x00;
  P2REN = 0x00;
  P3OUT = 0x00;
  P3DIR = 0x00;
  P3REN = 0x00;
}

/*************************************************************************************/
/*  Function name: RTC_Clock_init
/*  Parameters
/*  Input   :  No
/*  Output  :  No
/*  Action: Set up RTC clock operation
/*************************************************************************************/

void RTC_Clock_init(void){

  P5OUT = 0x00;
  //P5DIR = 0x02;  // If this is uncommented RTC doesn't work!!!!!!!!!!!!!!!!! P5DIR register must
be set to 0x00!!!!!!

  P5SEL = 0x03; // Enabled alternative functions

  _BIC_SR(OSCOFF);          // Enable the LFXT1 crystal oscillator
  UCSCTL6 &= (~XTS);        // Select Low-frequency mode. XCAP bits define the capacitance at the
XIN and XOUT pins.

  UCSCTL6 |= (XCAP0);       // Set XCAP0
  UCSCTL6 |= (XCAP1);       // Set XCAP1
  // Then Qrystal Load Capacitance is:
  //  (XCAP1=0),(XCAP0=0) -> 2pF     (XCAP1=0),(XCAP0=1) -> 5.5pF
  //  (XCAP1=1),(XCAP0=0) -> 8.5pF   (XCAP1=1),(XCAP0=1) -> 12pF
  UCSCTL6 &= (~XT1OFF);     // Turns on the XT1.
  // Loop until XT1,XT2 & DCO stabilizes

  do{

    UCSCTL7 &= ~(XT2OFFG + XT1LFOFFG + XT1HFOFFG + DCOFFG); // Clear
XT2,XT1,DCO fault flags
    SFRIFG1 &= ~OFIFG;                    // Clear fault flags
  }while (SFRIFG1&OFIFG);                   // Test oscillator fault flag

  UCSCTL4 = ((SELA__XT1CLK | SELS__DCOCLKDIV) | SELM__DCOCLKDIV);   // ACLK
source: XT1CLK, SMCLK: DCOCLKDIV, MCLK: DCOCLKDIV
  SFRIFG1 &= (~OFIFG);      // Oscillator fault interrupt flag
  SFRIE1 |= (OFIE);         // Oscillator fault interrupt enable
}
```

```
/*************************************************************************************/
/*  Function name: timer1_A3_init
/* Parameters
/*  Input   :  No
/*  Output  :  No
/*  Action: Initialize Timer1_A3 operation
/*************************************************************************************/

//TIMER1 initialize - prescale:0
// desired value:   2Hz, i.e. 32768/2 = 16384(0x4000)
void timer1_A3_init(void){

   TA1CTL = 0x0004;   // Timer1_A3 clear.
   TA1CCTL0 = 0x0010; // Timer1_A3 Capture/compare 0 interrupt enable.
   TA1CCR0 = 0x0013;  // Set TACCR0 value to 5ms
   TA1CTL = 0x0210;   // Selected: SMCLK, divider:1, Up mode.
   UCSCTL4 = 0000;    // XT1CLK clock selected
}

/*************************************************************************************/
/*  Function name: init_devices
/*  Parameters
/*  Action: Initialize all peripherals
/*************************************************************************************/

void init_devices(void){

 //1. Stop errant interrupts until set up
 _BIC_SR(GIE); // Disable interrupts during initialization process
 //2. Init System Clock
 // By default the FLL stabilizes MCLK and SMCLK to 1.048576 MHz and fDCO = 2.097152 MHz.

 //3. Init Peripherals
 //RTC_Clock_init();
 ports_init();
 timer1_A3_init();
 //timer1_A3_Stop_Mode;

 //P3DIR |= BIT7;                // Show SMCLK to P3.7
 //P3SEL |= BIT7;

 _BIS_SR(GIE); // Global Interrupt enabled. Do this at the END of the initialization!!!!!!!!
 //all peripherals are now initialized
}
int main(void){

  WDTCTL = WDTPW + WDTHOLD;              // Stop WDT
  init_devices();
  while(1){
```

```
    RF_Connection_Test();
  }
}
```

2.2. RF_Connection(1)

```
/*******************************************************************************
/* File Name:  RF_Connection.c
/* Simple RF Link to Toggle Receiver's LED by pressing Transmitter's Button
/* Warning: This RF code example is setup to operate at 868Mhz frequency,
/* which might be out of allowable range of operation in certain countries. Please
/* refer to the appropriate legal sources before performing tests with this code
/* example.
/*
/* This code example can be loaded to 2 MSP430-CCRF devices. Each device will transmit
/* a small packet upon a button pressed. Each device will also toggle its LED upon
/* receiving said packet.
/*
/* The RF packet engine settings specify variable-length-mode with CRC check enabled
/* The RX packet also appends 2 status bytes regarding CRC check, RSSI and LQI info.
/* For specific register settings please refer to the comments for each register in
/* RF1A_REGISTER_CONFIG[] or the CC430x613x User's Guide and the SmartRF Studio
/*
/* All required changes, which enable this code to be portable for MSP430-CCRF,
/* were made by Penko T. Bozhkov -> Olimex LTD
/* All required changes and algorithms for active markers project were made by Gorka
/* Corral Malla
/*******************************************************************************/

#include "cc430x513x.h"
#include "HAL/RF1A.h"
#include "HAL/cc430x613x_PMM.h"
#include "RF_Connection.h"

//#define EMISOR
#define RECEPTOR

#ifdef MHZ_915
#include "HAL/RF_config_Olimex/smartrf_CC430F5137_915MHz_38k4Baud.h"
#elif defined MHZ_868
#include "HAL/RF_config_Olimex/smartrf_CC430F5137_868MHz_38k4Baud.h"
#endif


#define LED_TOGG    P1OUT ^= BIT0;  P1DIR |= BIT0;
#define PULSE_TOGG  P3OUT ^= BIT1;  P3DIR |= BIT1;
#define PULSE_UP    P3OUT |= BIT1;        P3DIR |= BIT1;
#define PULSE_DOWN  P3OUT &= ~BIT1;         P3DIR |= BIT1;
```

```
const unsigned char RF1A_REGISTER_CONFIG[CONF_REG_SIZE]=
{

  SMARTRF_SETTING_IOCFG2  ,  // IOCFG2: GDO2 signals on RF_RDYn
  SMARTRF_SETTING_IOCFG1  ,  // IOCFG1: GDO1 signals on RSSI_VALID
  SMARTRF_SETTING_IOCFG0  ,  // IOCFG0: GDO0 signals on PA power down signal to control RX/TX switch
  SMARTRF_SETTING_FIFOTHR , // FIFOTHR: RX/TX FIFO Threshold: 33 bytes in TX, 32 bytes in RX
  SMARTRF_SETTING_SYNC1   , // SYNC1: high byte of Sync Word
  SMARTRF_SETTING_SYNC0   , // SYNC0: low byte of Sync Word
  SMARTRF_SETTING_PKTLEN  , // PKTLEN: Packet Length in fixed mode, Maximum Length in variable-length mode
  SMARTRF_SETTING_PKTCTRL1, // PKTCTRL1: No status bytes appended to the packet
  SMARTRF_SETTING_PKTCTRL0, // PKTCTRL0: Fixed-Length Mode, No CRC
  SMARTRF_SETTING_ADDR    , // ADDR: Address for packet filtration
  SMARTRF_SETTING_CHANNR  , // CHANNR: 8-bit channel number, freq = base freq + CHANNR * channel spacing
  SMARTRF_SETTING_FSCTRL1 , // FSCTRL1: Frequency Synthesizer Control (refer to User's Guide/SmartRF Studio)
  SMARTRF_SETTING_FSCTRL0 , // FSCTRL0: Frequency Synthesizer Control (refer to User's Guide/SmartRF Studio)
  SMARTRF_SETTING_FREQ2   , // FREQ2: base frequency, high byte
  SMARTRF_SETTING_FREQ1   , // FREQ1: base frequency, middle byte
  SMARTRF_SETTING_FREQ0   , // FREQ0: base frequency, low byte
  SMARTRF_SETTING_MDMCFG4 , // MDMCFG4: modem configuration (refer to User's Guide/SmartRF Studio)
  SMARTRF_SETTING_MDMCFG3 , // MDMCFG3:          "          "
  SMARTRF_SETTING_MDMCFG2 , // MDMCFG2:          "          "
  SMARTRF_SETTING_MDMCFG1 , // MDMCFG1:          "          "
  SMARTRF_SETTING_MDMCFG0 , // MDMCFG0:          "          "
  SMARTRF_SETTING_DEVIATN , // DEVIATN: modem deviation setting (refer to User's Guide/SmartRF Studio)
  SMARTRF_SETTING_MCSM2   , // MCSM2: Main Radio Control State Machine Conf. : timeout for sync word search disabled
  SMARTRF_SETTING_MCSM1   , // MCSM1: CCA signals when RSSI below threshold, stay in RX after packet has been received
  SMARTRF_SETTING_MCSM0   , // MCSM0: Auto-calibrate when going from IDLE to RX or TX (or FSTXON )
  SMARTRF_SETTING_FOCCFG  , // FOCCFG: Frequency Offset Compensation Conf.
  SMARTRF_SETTING_BSCFG   , // BSCFG: Bit Synchronization Conf.
  SMARTRF_SETTING_AGCCTRL2, // AGCCTRL2: AGC Control
  SMARTRF_SETTING_AGCCTRL1, // AGCCTRL1:     "
  SMARTRF_SETTING_AGCCTRL0, // AGCCTRL0:     "
  SMARTRF_SETTING_WOREVT1 , // WOREVT1: High Byte Event0 Timeout
  SMARTRF_SETTING_WOREVT0 , // WOREVT0: High Byte Event0 Timeout
  SMARTRF_SETTING_WORCTRL , // WORCTL: Wave On Radio Control ****Feature unavailable in PG0.6****
  SMARTRF_SETTING_FREND1  , // FREND1: Front End RX Conf.
```

```
  SMARTRF_SETTING_FREND0  , // FREND0: Front End TX Conf.
  SMARTRF_SETTING_FSCAL3  , // FSCAL3: Frequency Synthesizer Calibration (refer to User's
Guide/SmartRF Studio)
  SMARTRF_SETTING_FSCAL2  , // FSCAL2:            "
  SMARTRF_SETTING_FSCAL1  , // FSCAL1:            "
  SMARTRF_SETTING_FSCAL0  , // FSCAL0:            "
  0x00                , // Reserved *read as 0*
  0x00                , // Reserved *read as 0*
  SMARTRF_SETTING_FSTEST  , // FSTEST: For test only, irrelevant for normal use case
  SMARTRF_SETTING_PTEST   , // PTEST: For test only, irrelevant for normal use case
  SMARTRF_SETTING_AGCTEST , // AGCTEST: For test only, irrelevant for normal use case
  SMARTRF_SETTING_TEST2   , // TEST2  : For test only, irrelevant for normal use case
  SMARTRF_SETTING_TEST1   , // TEST1  : For test only, irrelevant for normal use case
  SMARTRF_SETTING_TEST0     // TEST0  : For test only, irrelevant for normal use case
};


extern unsigned char packetReceived;
extern unsigned char packetTransmit;

unsigned char RxBuffer[255], RxBufferLength = 0;
const unsigned char TxBuffer[6]= {0x06, 0xAA, 0xBB, 0xCC, 0xDD, 0xEE};
unsigned char signalReceived = 0;
unsigned int i = 0;
unsigned int j = 1;
unsigned int count1 = 0;
unsigned int count2 = 0;
unsigned int fl = 0;
unsigned int time = 0;

/*****************************************************************************
/*  Function name: InitWireless_ports
/*  Parameters
/*  Input   :  No
/*  Output  :  No
/*  Action: Initialize Led and Button directions.
/*****************************************************************************

void InitWireless_ports(void){

  P2DIR &= ~BIT3;
  P2REN |= BIT3;
  P2IES &= BIT3;
  P2IFG = 0;
  P2OUT |= BIT3;
  P2IE  |= BIT3;

  // Set up LED
  P1OUT &= ~BIT0;
  P1DIR |= BIT0;
```

```
  P3OUT &= ~BIT1;
  P3DIR |= BIT1;
}

/*****************************************************************************
/*  Function name: Init_RF
/*  Parameters
/*  Input   :  No
/*  Output  :  No
/*  Action: Initialize RF radio core.
/*****************************************************************************

void Init_RF(void){

  // Increase PMMCOREV level to 2 in order to avoid low voltage error
  // when the RF core is enabled
  SetVCore(2);
  ResetRadioCore();
  WriteBurstReg(IOCFG2, (unsigned char*)RF1A_REGISTER_CONFIG, CONF_REG_SIZE);
  WritePATable();
  InitWireless_ports();
  ReceiveOn();
  //Wait for RX status to be reached
  while((Strobe(RF_SNOP) & 0x70) != 0x10);

}

/*****************************************************************************
/*  Function name: RF_Connection_Test
/*  Parameters
/*  Action: Sync up input camera signal with LEDs activation signal.
/*****************************************************************************

void RF_Connection_Test(void){

  Init_RF();

  Strobe( RF_SFRX);      /* Flush the receive FIFO of any residual data */

  while(1){

    if (signalReceived) {                 // Process a signal received --> transmit

      //P3OUT |= BIT6;
      signalReceived = 0;
#ifdef EMISOR
      P2IFG = 0;
      LED_TOGG;
      ReceiveOff();
```

```
      if(count2 == 80)
        {
         Transmit( (unsigned char*)TxBuffer, sizeof TxBuffer);
         count2 = 0;
        }
      count2 ++;

      //Wait for TX status to be reached before going back to low power mode
      //while((Strobe(RF_SNOP) & 0x70) != 0x20);

      P2IE |= BIT3;                     // Re-enable signal reception
#endif
    }
#ifdef EMISOR
    else if (packetTransmit)
    {
      count2 = 0;
      RF1AIE &= ~BIT9;                  // Disable RFIFG9 TX end-of-packet interrupts
      //P3OUT &= ~BIT6;

      ReceiveOn();
      // LED_TOGG;
      //Wait for RX status to be reached
      //while((Strobe(RF_SNOP) & 0x70) != 0x10);

      packetTransmit = 0;
      count2 = 0;
    }
#endif
#ifdef RECEPTOR
    if(packetReceived)              // Process a received packet
    {
      //char toggle = 1;
      LED_TOGG;
      // Read the length byte from the FIFO
      RxBufferLength = ReadSingleReg( RXBYTES );
      if (RxBufferLength != 0){
         ReadBurstReg(RF_RXFIFORD, RxBuffer, RxBufferLength);

         // Check the packet contents and don't toggle LED if they are different
         for(i = 0; i < RxBuffer[0]; i++){

            //if(RxBuffer[i] != TxBuffer[i]) toggle = 0;
         }
      }

      if(1)
      {
         //delay
            //TA1CTL = 0x0004;   // Timer1_A3 clear.
```

```
            //TA1CCTL0 = 0x0010; // Timer1_A3 Capture/compare 0 interrupt enable.
            j=1;
            TA1CCR0 = 0x0000;  // Set TACCR0 value to 0ms
            //TA1CTL = 0x0210;   // Selected: SMCLK, divider:1, Up mode.
            //UCSCTL4 = 0000;    // XT1CLK clock selected
          //count1 = 0;
      }
      packetReceived = 0;
      //count1 ++;
      Strobe( RF_SFRX);      /* Flush the receive FIFO of any residual data */
      ReceiveOn();
    }
#endif
  }
}

/**************************************************************************************
/*  PORT2, interrupt service routine.
/**************************************************************************************

#pragma vector=PORT2_VECTOR
__interrupt void PORT2_ISR(void)
{
  switch(__even_in_range(P2IV, 8))
  {
    case  0: break;
    case  2: break;                    // P2.0 IFG
    case  4: break;                    // P2.1 IFG
    case  6: break;                    // P2.2 IFG
    case  8:
       P2IE = 0;                      // Debounce by disabling signal reception
       if(P2IN & BIT3){
          signalReceived = 1;
       }else{
          P2IFG = 0;
          P2IE |= BIT3;
       }
       break;                        // P2.3 IFG
    case 10: break;                   // P2.4 IFG
    case 12: break;                   // P2.5 IFG
    case 14: break;                   // P2.6 IFG
    case 16: break;                   // P2.7 IFG

  }
}


/**************************************************************************************/
/*  Timer1_A3 CC0 interrupt service routine.
/**************************************************************************************/
```

```
#pragma vector=TIMER1_A0_VECTOR
__interrupt void TIMER1_A0_Capture_Compare_ISR(void)
{
          __delay_cycles(12);
   if(j){
     PULSE_UP;
     j=0;
     TA1CCR0 = 0x006F;  // Set TACCR0 value to 3,484 ms aprox
   }
  else{
       PULSE_DOWN;
       j=1;
       TA1CCR0 = 0x00A2;  // Set TACCR0 value to 4,920 ms aprox
      }
}
```

2.3. RF_Connection(2)

```
/*****************************************************************************/
/*    Manufacture: OLIMEX
/*    COPYRIGHT (C) 2011
/*   Original Program Designed by: Penko Todorov Bozhkov
/*   Current Program Designed by:  Gorka Corral Malla
/*    Module Name:  RF_Connection
/*    File   Name:  RF_Connection.h
/*    Revision Identifier:  initial
/*    Date:                 07.10.2018
/*****************************************************************************/

#ifndef __RF_Connection_H
#define __RF_Connection_H

/***** 1.All functions prototypes *****/
void InitWireless_ports(void);
void Init_RF(void);
void RF_Connection_Test(void);

#endif //_RF_Connection_H
```

2.4. RF1A(1)

```
/* File Name:  RF1A.c

#include "RF1A.h"
#include "cc430x513x.h"

unsigned char packetReceived = 0;
unsigned char packetTransmit = 0;
```

```
// ###########################################################
// This definitions and delay_RF were added by Penko Bozhkov
// ###########################################################
#define TXtoIDLE_Time     10    // TX to IDLE, no calibration: ~1us   => 0.3us *10 = 3us
#define RXtoIDLE_Time     2     // RX to IDLE, no calibration: ~0.1us => 0.3*2 = 0.6us
#define IDLEtoRX_Time     300   // IDLE to RX, no calibration: 75.1us => 0.3*300 = 90us

/*****************************************************************************/
/*  Function name: delay_RF
/*  Parameters
/*  Input   :  p
/*  Output  :  No
/*  Action: Simple delay
/*****************************************************************************/

void delay_RF(volatile unsigned long p){
        //while(p){p--;}    // delay_RF will take at least 6 MCU cycles(20Mhz/6=3.33MHz)
1/3.33Mhz = 0.3us
}

unsigned char Strobe(unsigned char strobe)
{
 unsigned char statusByte;

 // Check for valid strobe command
 if((strobe == 0xBD) || (strobe >= 0x30) && (strobe <= 0x3D))
 {
  RF1AIFCTL1 &= ~(RFSTATIFG);            // Clear the status read flag

  while( !(RF1AIFCTL1 & RFINSTRIFG)  );   // Wait for INSTRIFG
  RF1AINSTRB = strobe;                   // Write the strobe command

  if(strobe != 0x30) while( !(RF1AIFCTL1 & RFSTATIFG) );
  statusByte = RF1ASTATB;
 }
 else
  return 0;                          // Invalid strobe was sent

 return statusByte;
}

unsigned char ReadSingleReg(unsigned char addr)
{
 unsigned char data_out;

 // Check for valid configuration register address, 0x3E refers to PATABLE
 if ((addr <= 0x2E) || (addr == 0x3E))
  // Send address + Instruction + 1 dummy byte (auto-read)
  RF1AINSTR1B = (addr | RF_SNGLREGRD);
 else
```

```
    // Send address + Instruction + 1 dummy byte (auto-read)
    RF1AINSTR1B = (addr | RF_STATREGRD);

  while (!(RF1AIFCTL1 & RFDOUTIFG) );
  data_out = RF1ADOUT1B;                  // Read data and clears the RFDOUTIFG

  return data_out;
}

void WriteSingleReg(unsigned char addr, unsigned char value)
{
  while (!(RF1AIFCTL1 & RFINSTRIFG));       // Wait for the Radio to be ready for next instruction
  RF1AINSTRB = (addr | RF_REGWR);                       // Send address + Instruction

  RF1ADINB = value;                                                 // Write data in

  __no_operation();
}


void ReadBurstReg(unsigned char addr, unsigned char *buffer, unsigned char count)
{
  unsigned int i;

  while (!(RF1AIFCTL1 & RFINSTRIFG));       // Wait for INSTRIFG
  RF1AINSTR1B = (addr | RF_REGRD);          // Send addr of first conf. reg. to be read
                                // ... and the burst-register read instruction
  for (i = 0; i < (count-1); i++)
  {
    while (!(RFDOUTIFG&RF1AIFCTL1));        // Wait for the Radio Core to update the
RF1ADOUTB reg
    buffer[i] = RF1ADOUT1B;                 // Read DOUT from Radio Core + clears RFDOUTIFG
                                // Also initiates auo-read for next DOUT byte
  }
  buffer[count-1] = RF1ADOUT0B;             // Store the last DOUT from Radio Core
}

void WriteBurstReg(unsigned char addr, unsigned char *buffer, unsigned char count)
{
  // Write Burst works wordwise not bytewise - known errata
  unsigned char i;

  while (!(RF1AIFCTL1 & RFINSTRIFG));       // Wait for the Radio to be ready for next instruction
  RF1AINSTRW = ((addr | RF_REGWR)<<8 ) + buffer[0]; // Send address + Instruction

  for (i = 1; i < count; i++)
  {
    RF1ADINB = buffer[i];                   // Send data
    while (!(RFDINIFG & RF1AIFCTL1));       // Wait for TX to finish
  }
```

```
  i = RF1ADOUTB;                    // Reset RFDOUTIFG flag which contains status byte
}

void WriteSmartRFReg(const unsigned char SmartRFSetting[][2], unsigned char size)
{
  unsigned char i;
  for (i=0; i < (size); i++)
    WriteSingleReg(SmartRFSetting[i][0], SmartRFSetting[i][1]);
}

void ResetRadioCore (void)
{
  Strobe(RF_SRES);                  // Reset the Radio Core
  Strobe(RF_SNOP);                  // Reset Radio Pointer
}

// With aim to change RF output power, make your choice here!!!
void WritePATable(void)
{
  unsigned char valueRead = 0;

  while(valueRead != 0x03)    // Output Power: -30  [dBm]
  //while(valueRead != 0x25)    // Output Power: -12  [dBm]
  //while(valueRead != 0x2D)    // Output Power: -6   [dBm]
  //while(valueRead != 0x8D)    // Output Power:  0   [dBm]
  //while(valueRead != 0xC3)    // Output Power:  10  [dBm]
  //while(valueRead != 0xC0)    // Output Power:  Maximum  [dBm]
  //while(valueRead != 0xC6)    // Output Power(default):  8.8  [dBm]
  {
    /* Write the power output to the PA_TABLE and verify the write operation.  */
    unsigned char i = 0;

    /* wait for radio to be ready for next instruction */
    while( !(RF1AIFCTL1 & RFINSTRIFG));
    RF1AINSTRW = 0x7E03; // PA Table write (burst), Output Power: -30  [dBm]
    //RF1AINSTRW = 0x7E25; // PA Table write (burst), Output Power: -12  [dBm]
    //RF1AINSTRW = 0x7E2D; // PA Table write (burst), Output Power: -6   [dBm]
    //RF1AINSTRW = 0x7E8D; // PA Table write (burst), Output Power: 0    [dBm]
    //RF1AINSTRW = 0x7EC3; // PA Table write (burst), Output Power: 10   [dBm]
    //RF1AINSTRW = 0x7EC0; // PA Table write (burst), Output Power: Maximum  [dBm]
    //RF1AINSTRW = 0x7EC6; // PA Table write (burst), Output Power(default):  8.8  [dBm]

    /* wait for radio to be ready for next instruction */
    while( !(RF1AIFCTL1 & RFINSTRIFG));
      RF1AINSTR1B = RF_PATABRD;

    // Traverse PATABLE pointers to read
    for (i = 0; i < 7; i++)
    {
      while( !(RF1AIFCTL1 & RFDOUTIFG));
```

```
      valueRead  = RF1ADOUT1B;
    }
    while( !(RF1AIFCTL1 & RFDOUTIFG));
    valueRead  = RF1ADOUTB;
  }
}

void Transmit(unsigned char *buffer, unsigned char length)
{
  RF1AIES |= BIT9;
  RF1AIFG &= ~BIT9;                     // Clear pending interrupts
  RF1AIE |= BIT9;                       // Enable RFIFG9 TX end-of-packet interrupts

  /* RF1AIN_9 => Rising edge indicates SYNC sent/received and
   *           Falling edge indicates end of packet.
   *           Configure it to interrupt on falling edge.
   */
  WriteBurstReg(RF_TXFIFOWR, buffer, length);

  Strobe( RF_STX );                     // Strobe STX
}

void ReceiveOn(void)
{
  RF1AIFG &= ~BIT4;                     // Clear a pending interrupt
  RF1AIE  |= BIT4;                      // Enable the interrupt

  // Previous state has been Tx
  Strobe( RF_SIDLE );
  delay_RF(TXtoIDLE_Time);
  Strobe( RF_SRX );
  delay_RF(IDLEtoRX_Time);
}

void ReceiveOff(void)
{
  RF1AIE &= ~BIT4;                      // Disable RX interrupts
  RF1AIFG &= ~BIT4;                     // Clear pending IFG

  // Previous state has been Rx
  Strobe( RF_SIDLE );
  delay_RF(RXtoIDLE_Time);
  Strobe( RF_SFRX);      /* Flush the receive FIFO of any residual data */

}

// Called if an interface error has occurred. No interface errors should
// exist in application code, so this is intended to be used for debugging
// or to catch errant operating conditions.
static void RF1A_interface_error_handler(void)
```

```
{
  switch(__even_in_range(RF1AIFERRV,8))
  {
    case 0: break;                    // No error
    case 2:                           // Low core voltage error
      P1OUT &= ~BIT0;                                          // 00 = on LED's [D2,D1]
      P3OUT &= ~BIT6;
      __no_operation();
      break;
    case 4:                           // Operand Error
      P1OUT |= BIT0;                                           // 01 = on LED's [D2,D1]
      P3OUT &= ~BIT6;
      __no_operation();
      break;
    case 6:                           // Output data not available error
      P1OUT &= ~BIT0;                                          // 10 = on LED's [D2,D1]
      P3OUT |= BIT6;
      __no_operation();
      break;
    case 8:                           // Operand overwrite error
      P1OUT |= BIT0;                                           // 11 = on LED's [D2,D1]
      P3OUT |= BIT6;
      __no_operation();
      break;
  }
}

// If RF1A_interrupt_handler is called, an interface interrupt has occured.
static void RF1A_interrupt_handler(void)
{
  // RF1A interrupt is pending
  switch(__even_in_range(RF1AIFIV,14))
  {
    case  0: break;                   // No interrupt pending
    case  2:                          // RFERRIFG
      RF1A_interface_error_handler();
    case  4: break;                   // RFDOUTIFG
    case  6: break;                   // RFSTATIFG
    case  8: break;                   // RFDINIFG
    case 10: break;                    // RFINSTRIFG
    case 12: break;                    // RFRXIFG
    case 14: break;                    // RFTXIFG
  }
}

#pragma vector=CC1101_VECTOR
__interrupt void CC1101_ISR(void)
{
  switch(__even_in_range(RF1AIV,32))        // Prioritizing Radio Core Interrupts
  {
```

```
    case  0:                        // No RF core interrupt pending
      RF1A_interrupt_handler();          // means RF1A interface interrupt pending
      break;
    case  2: break;                   // RFIFG0
    case  4: break;                   // RFIFG1
    case  6: break;                   // RFIFG2
    case  8: break;                   // RFIFG3
    case 10:                         // RFIFG4 - RX end-of-packet
      packetReceived = 1;
      break;
    case 12: break;                   // RFIFG5
    case 14: break;                   // RFIFG6
    case 16: break;                   // RFIFG7
    case 18: break;                   // RFIFG8
    case 20:                         // RFIFG9 - TX end-of-packet
      packetTransmit = 1;
      break;
    case 22: break;                   // RFIFG10
    case 24: break;                   // RFIFG11
    case 26: break;                   // RFIFG12
    case 28: break;                   // RFIFG13
    case 30: break;                   // RFIFG14
    case 32: break;                   // RFIFG15
  }
  __bic_SR_register_on_exit(LPM3_bits);
}
```

2.5. RF1A(2)

```
/* -------------------------------------------------------------------------------------------
 *                                    Defines
 * -------------------------------------------------------------------------------------------
 */

/* Define whether or not to check for RF1AERROR flags during debug */
//#define DEBUG 1              // For debug of error flags ONLY

/* Define frequency of operation by commenting the symbol, SELECT_MHZ_FOR_OPERATION,
 * then uncommenting the appropriate frequency symbol (868 or 915)*/
```

```
//#define SELECT_MHZ_FOR_OPERATION  1
#define MHZ_868   1          // Select either 868 or 915MHz
//#define MHZ_915   1

#ifdef SELECT_MHZ_FOR_OPERATION
  #error: "Please define frequency of operation in RF1A.h"
#endif

#define CONF_REG_SIZE    47         /* There are 47 8-bit configuration registers */

unsigned char Strobe(unsigned char strobe);
unsigned char ReadSingleReg(unsigned char addr);
void WriteSingleReg(unsigned char addr, unsigned char value);
void ReadBurstReg(unsigned char addr, unsigned char *buffer, unsigned char count);
void WriteBurstReg(unsigned char addr, unsigned char *buffer, unsigned char count);
void ResetRadioCore (void);
void WritePATable(void);
void Transmit(unsigned char *buffer, unsigned char length);
void ReceiveOn(void);
void ReceiveOff(void);
void WriteSmartRFReg(const unsigned char SmartRFSetting[][2], unsigned char size);
```