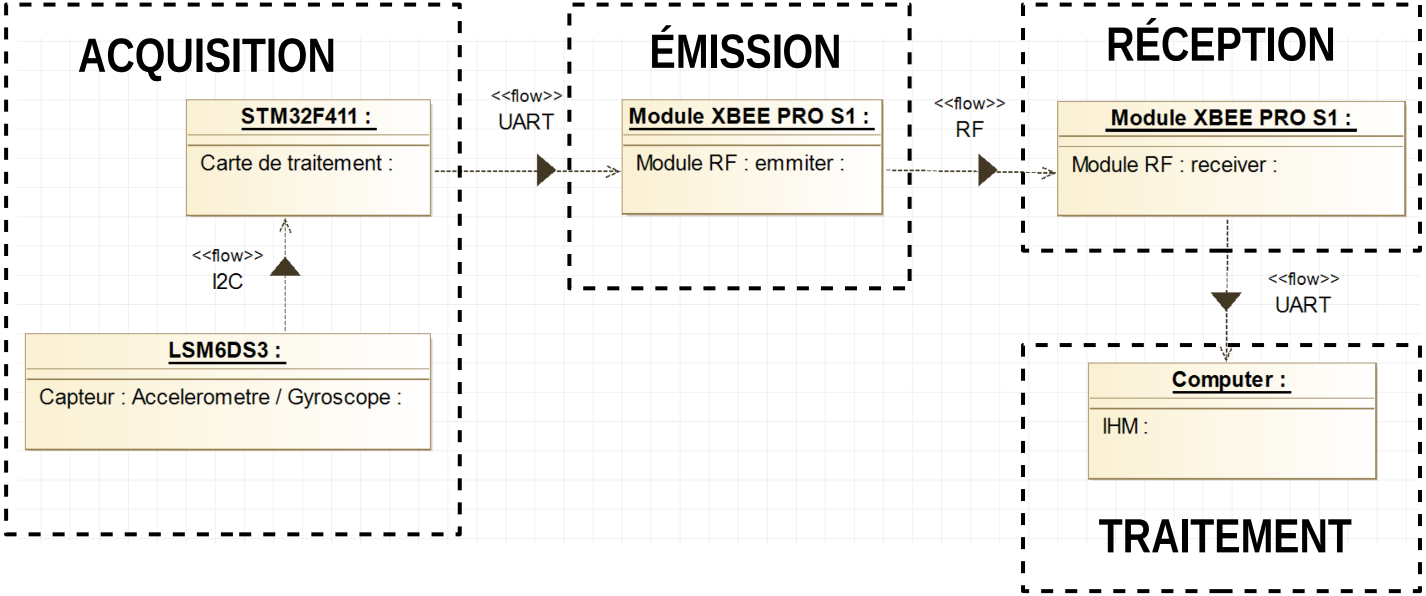
Projet EMB

Mise en service du
LSM6DS3

ARCHITECTURE DU SYSTÈME



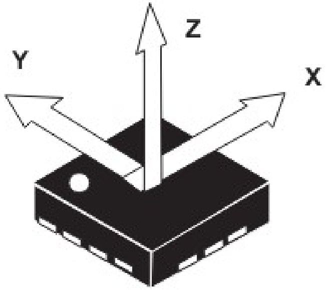
ACQUISITION

Le LSM6DS3 : *(STMicroelectronics)*

Accéléromètre 3D

Type : MEMS

Range :±2/±4/±8/±16 g

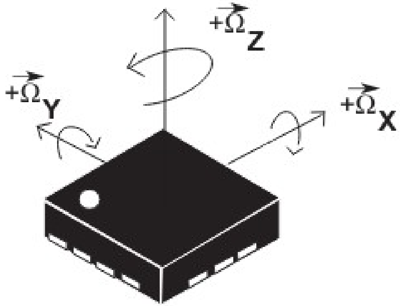


(TOP VIEW)

DIRECTION OF THE DETECTABLE ACCELERATIONS

Gyroscope 3D

±125/±245/±500/±1000/±2000 dps.



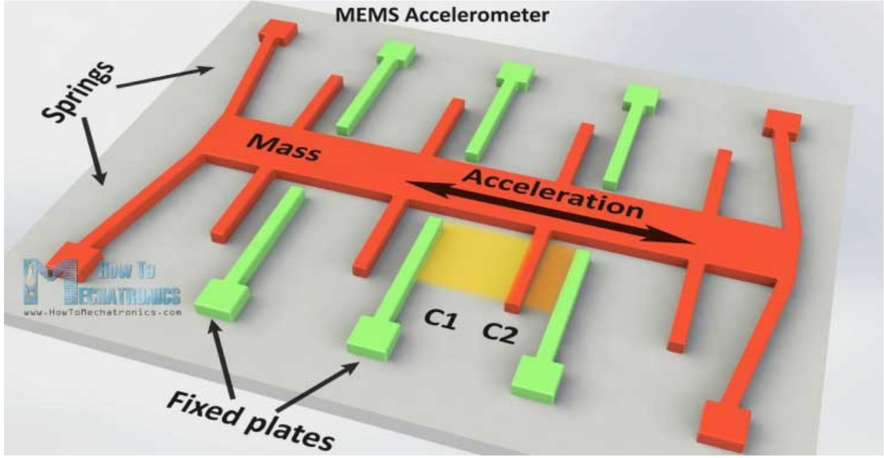
(TOP VIEW)

DIRECTIONS OF THE DETECTABLE ANGULAR RATES

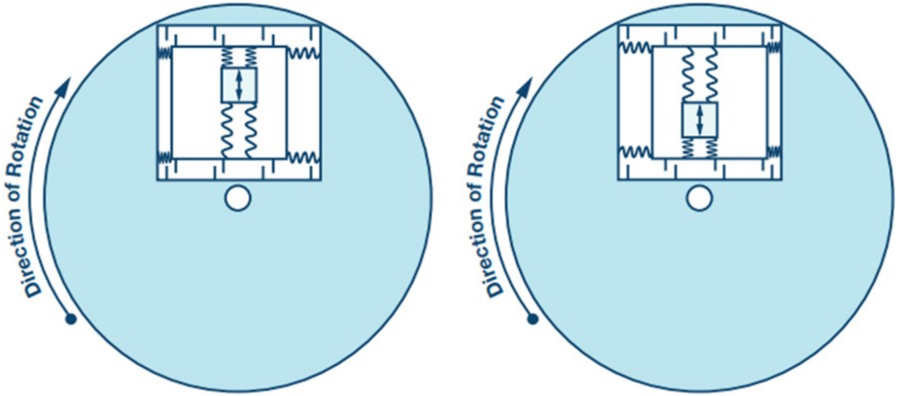
Module Grove - 3-Axis Digital Accelerometer v1.0

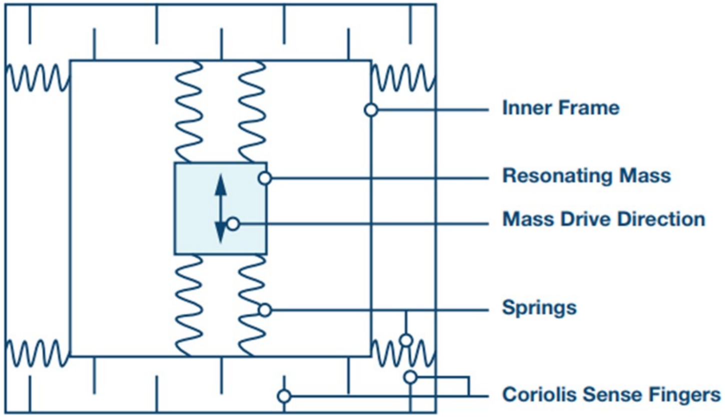
Principe Physique

Accéléromètre 3D



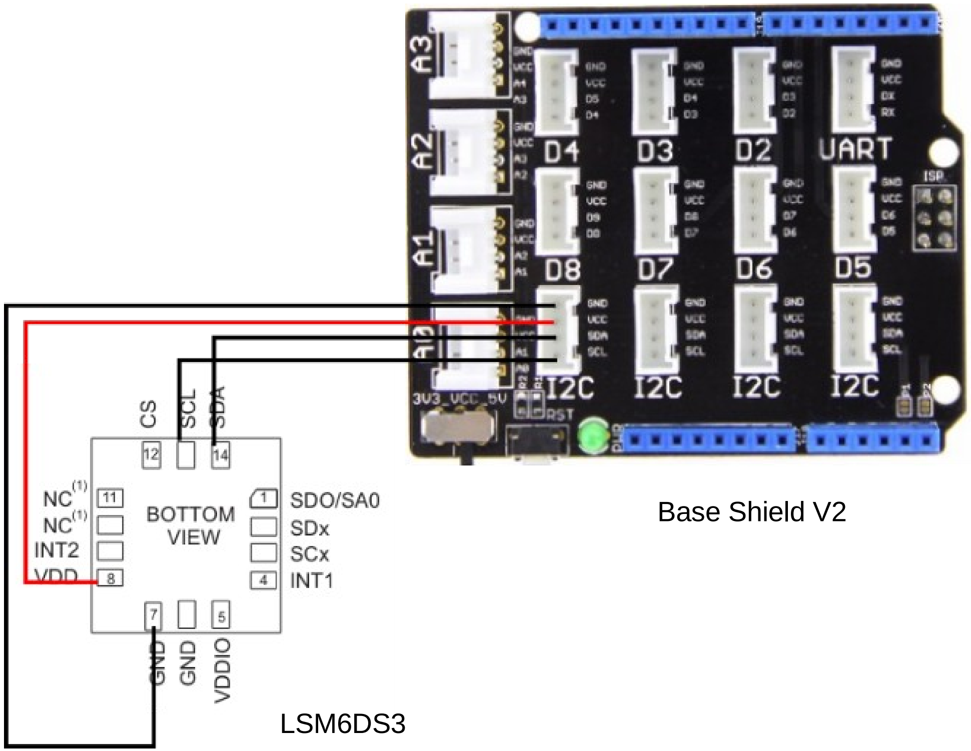
Gyroscope 3D





COMMUNICATION I2C

Hardware



Base Shield V2

LSM6DS3

Déterminer le SAD

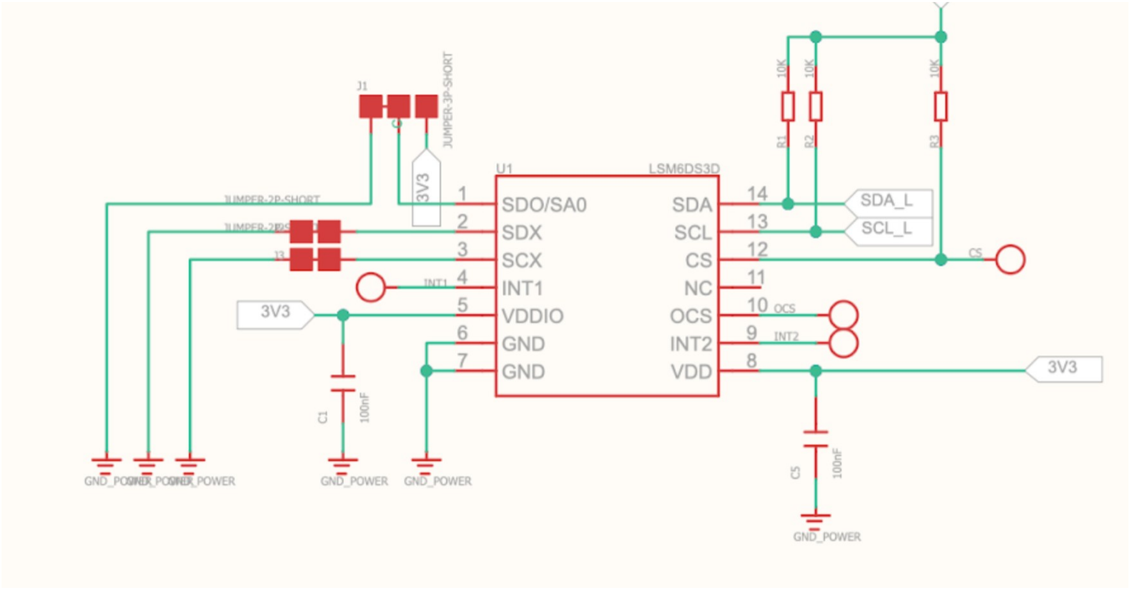
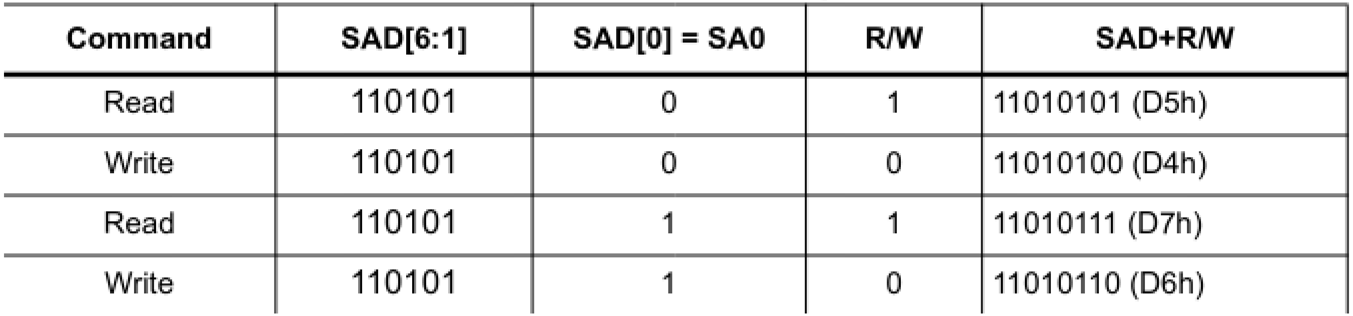
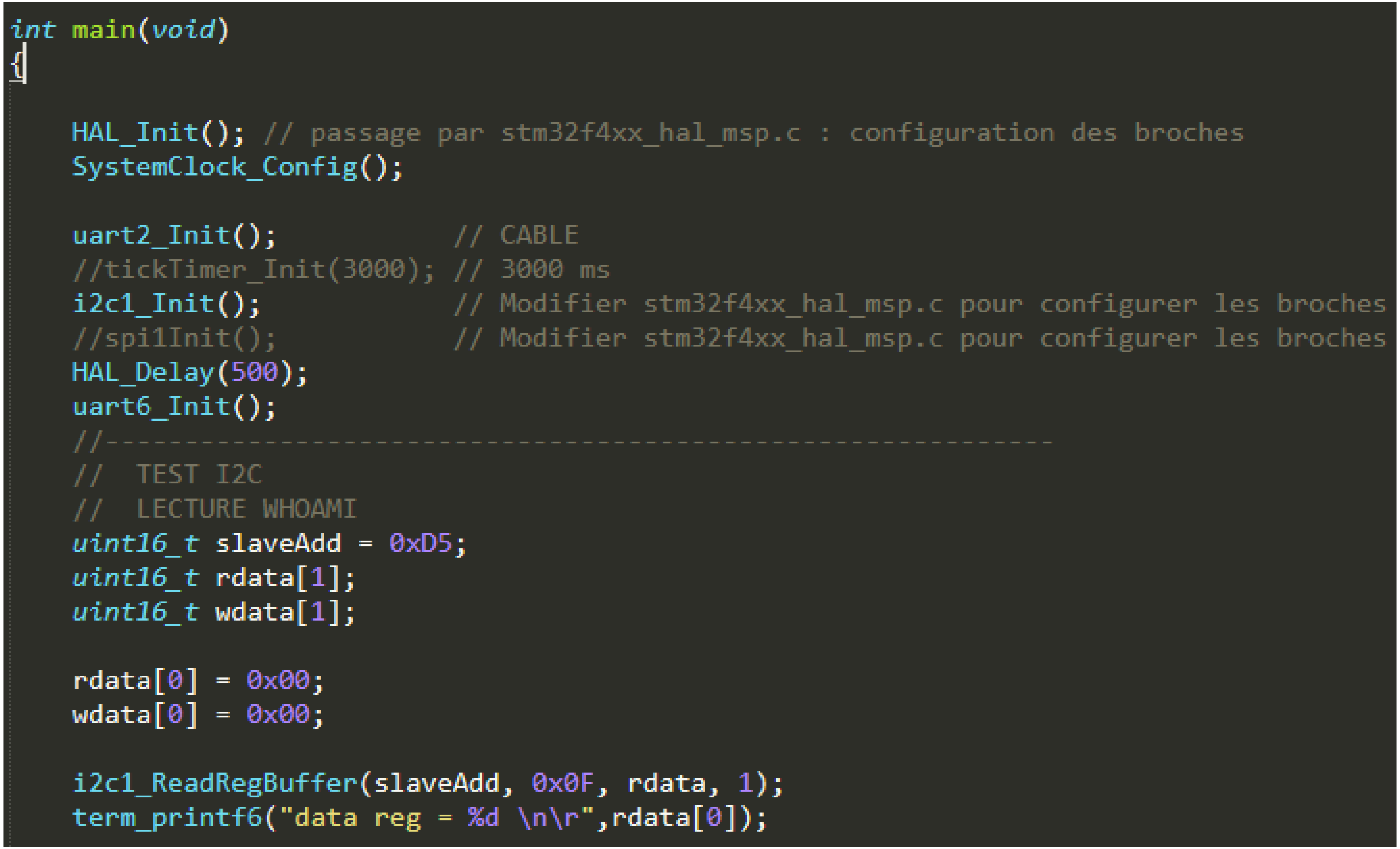


Table 11. SAD+Read/Write patterns



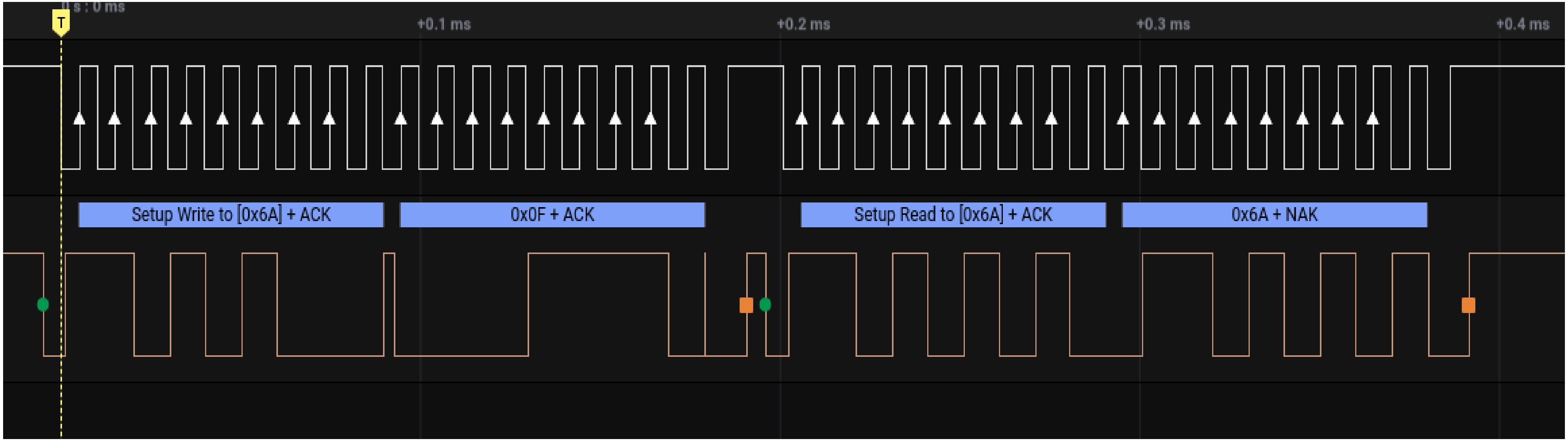
Command	SAD[6:1]	SAD[0] = SA0	R/W	SAD+R/W
Read	110101	0	1	11010101 (D5h)
Write	110101	0	0	11010100 (D4h)
Read	110101	1	1	11010111 (D7h)
Write	110101	1	0	11010110 (D6h)

Lecture d’un registre

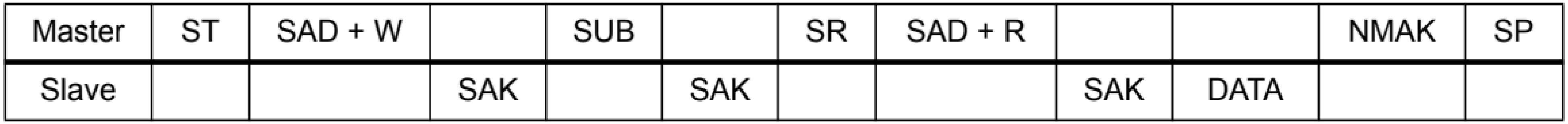


```
int main(void)
{

    HAL_Init(); // passage par stm32f4xx_hal_msp.c : configuration des broches
    SystemClock_Config();

    uart2_Init();           // CABLE
    //tickTimer_Init(3000); // 3000 ms
    i2c1_Init();            // Modifier stm32f4xx_hal_msp.c pour configurer les broches
    //spi1Init();           // Modifier stm32f4xx_hal_msp.c pour configurer les broches
    HAL_Delay(500);
    uart6_Init();
    //------------------------------------------------------------
    //  TEST I2C
    //  LECTURE WHOAMI
    uint16_t slaveAdd = 0xD5;
    uint16_t rdata[1];
    uint16_t wdata[1];

    rdata[0] = 0x00;
    wdata[0] = 0x00;

    i2c1_ReadRegBuffer(slaveAdd, 0x0F, rdata, 1);
    term_printf6("data reg = %d \n\r",rdata[0]);
```

Code C++ (1)



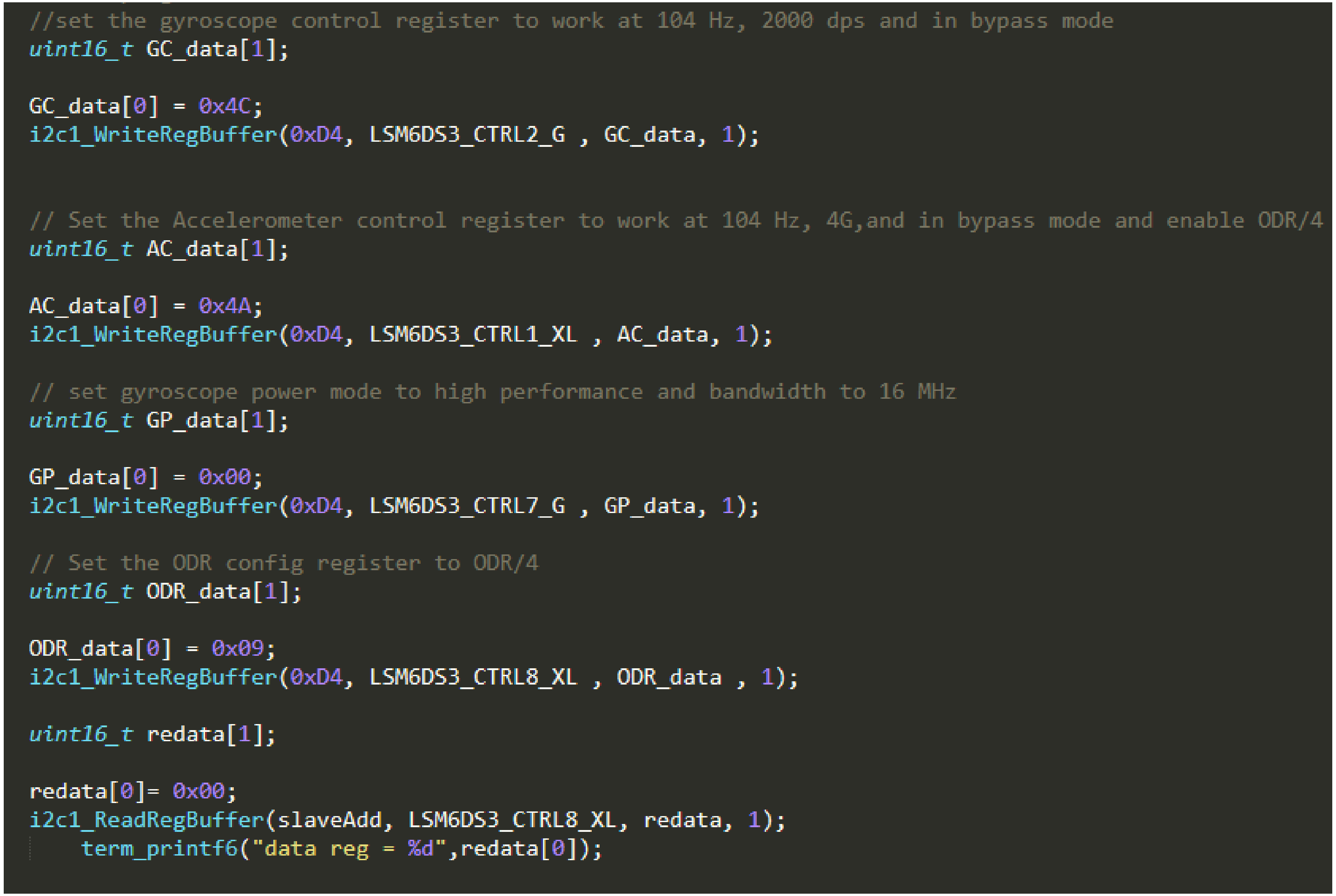
Capture d'une trame I2C (Read) avec un Analyseur logique et Logic



Master	ST	SAD + W		SUB		SR	SAD + R			NMAK	SP
Slave			SAK		SAK			SAK	DATA		

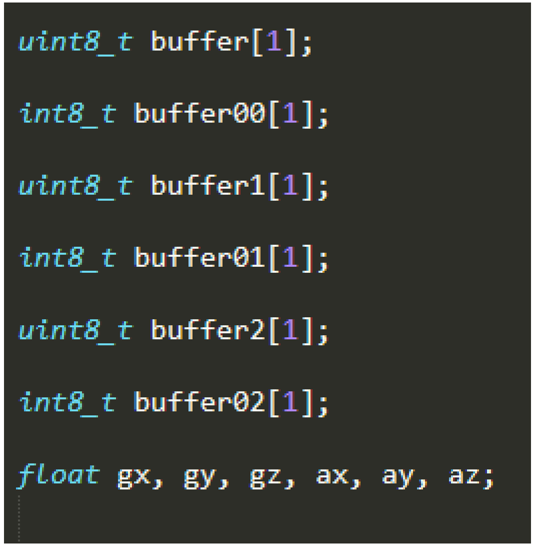
Communication I2C : LSMDS3 datasheet

Software



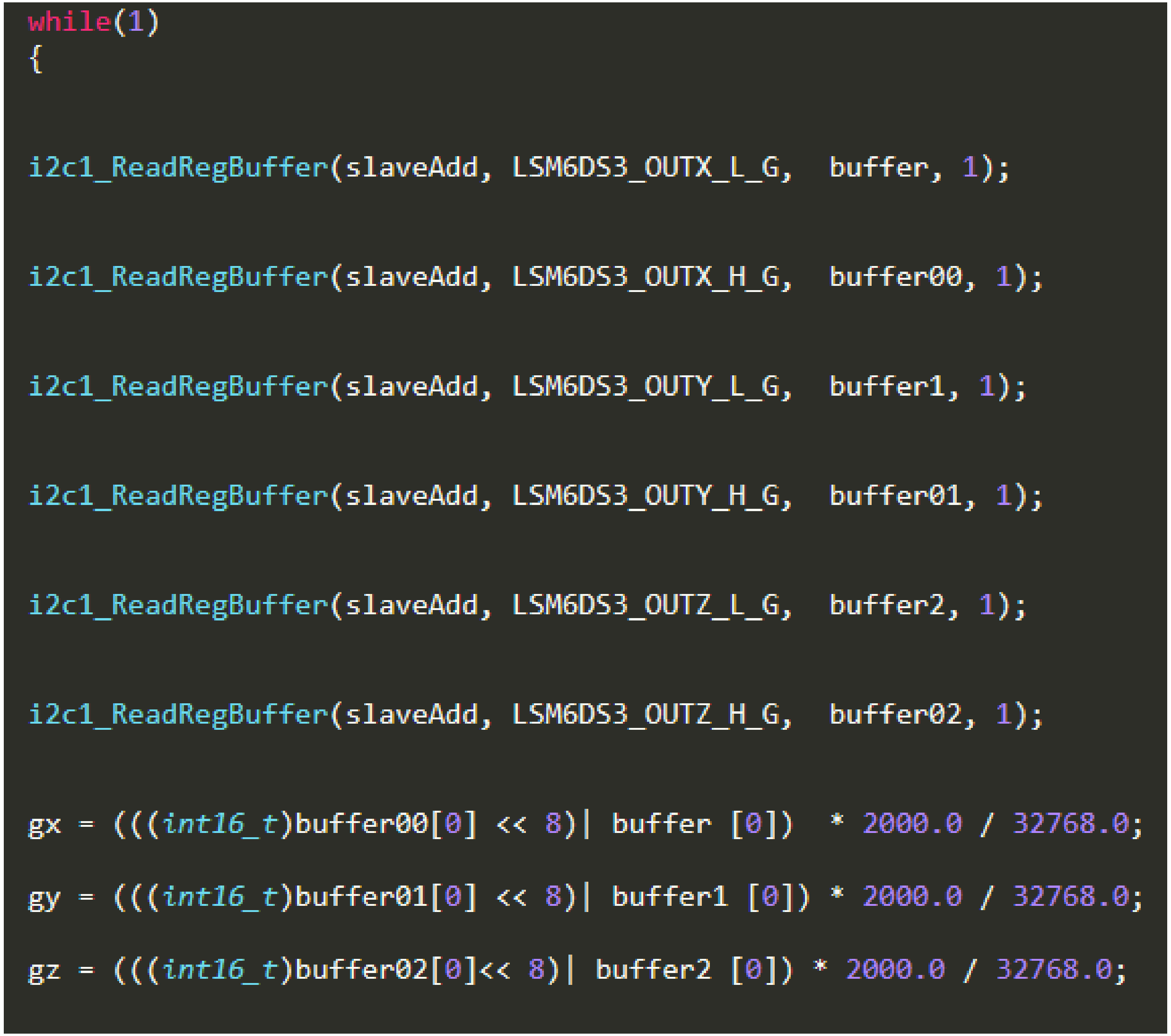
```
//set the gyroscope control register to work at 104 Hz, 2000 dps and in bypass mode
uint16_t GC_data[1];

GC_data[0] = 0x4C;
i2c1_WriteRegBuffer(0xD4, LSM6DS3_CTRL2_G , GC_data, 1);


// Set the Accelerometer control register to work at 104 Hz, 4G,and in bypass mode and enable ODR/4
uint16_t AC_data[1];

AC_data[0] = 0x4A;
i2c1_WriteRegBuffer(0xD4, LSM6DS3_CTRL1_XL , AC_data, 1);

// set gyroscope power mode to high performance and bandwidth to 16 MHz
uint16_t GP_data[1];

GP_data[0] = 0x00;
i2c1_WriteRegBuffer(0xD4, LSM6DS3_CTRL7_G , GP_data, 1);

// Set the ODR config register to ODR/4
uint16_t ODR_data[1];

ODR_data[0] = 0x09;
i2c1_WriteRegBuffer(0xD4, LSM6DS3_CTRL8_XL , ODR_data , 1);

uint16_t redata[1];

redata[0]= 0x00;
i2c1_ReadRegBuffer(slaveAdd, LSM6DS3_CTRL8_XL, redata, 1);
    term_printf6("data reg = %d",redata[0]);
```

Code C++ (2)



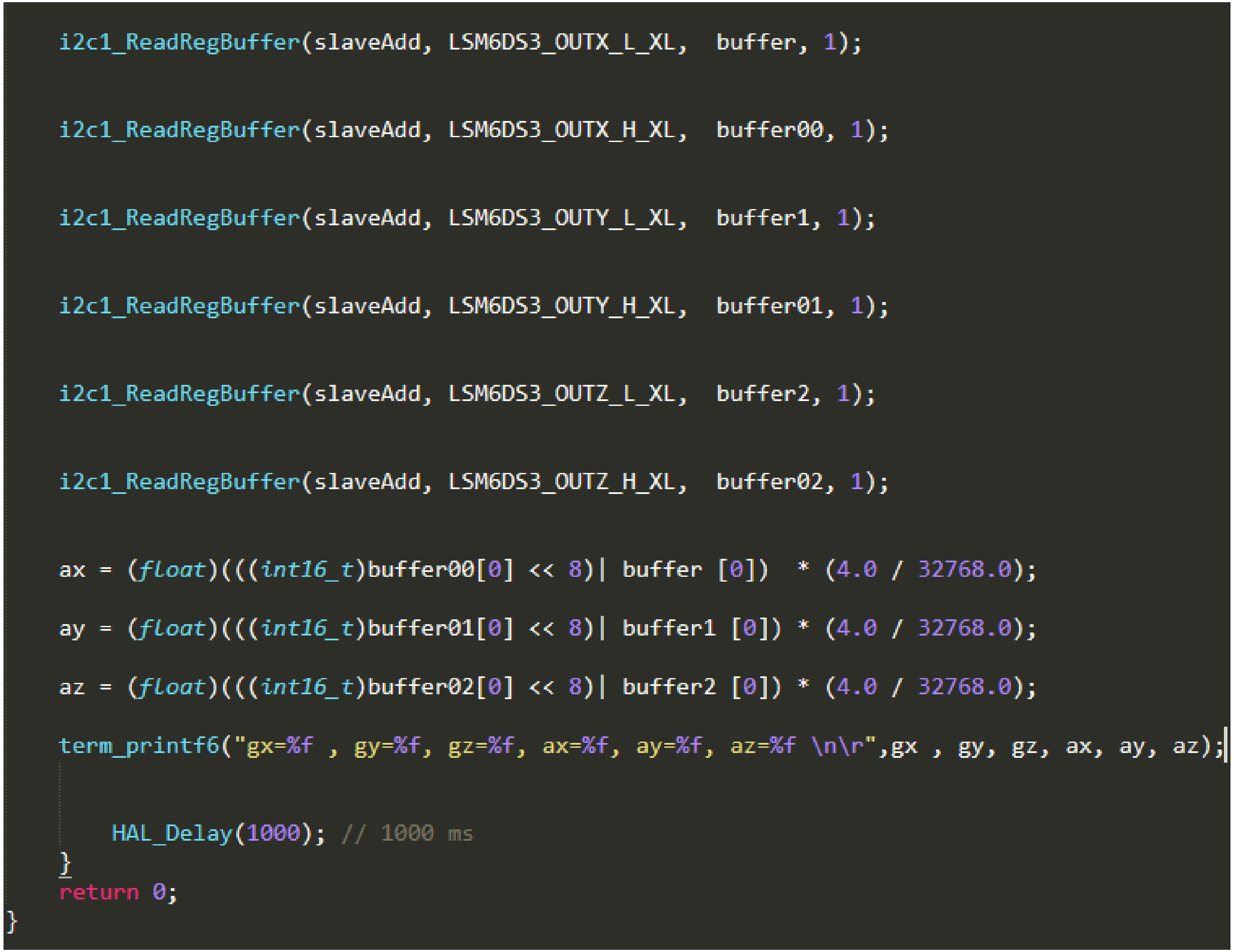
```
uint8_t buffer[1];

int8_t buffer00[1];

uint8_t buffer1[1];

int8_t buffer01[1];

uint8_t buffer2[1];

int8_t buffer02[1];

float gx, gy, gz, ax, ay, az;
```

Code C++ (3)



```
while(1)
{


i2c1_ReadRegBuffer(slaveAdd, LSM6DS3_OUTX_L_G,  buffer, 1);


i2c1_ReadRegBuffer(slaveAdd, LSM6DS3_OUTX_H_G,  buffer00, 1);


i2c1_ReadRegBuffer(slaveAdd, LSM6DS3_OUTY_L_G,  buffer1, 1);


i2c1_ReadRegBuffer(slaveAdd, LSM6DS3_OUTY_H_G,  buffer01, 1);


i2c1_ReadRegBuffer(slaveAdd, LSM6DS3_OUTZ_L_G,  buffer2, 1);


i2c1_ReadRegBuffer(slaveAdd, LSM6DS3_OUTZ_H_G,  buffer02, 1);


gx = (((int16_t)buffer00[0] << 8)| buffer [0])  * 2000.0 / 32768.0;

gy = (((int16_t)buffer01[0] << 8)| buffer1 [0]) * 2000.0 / 32768.0;

gz = (((int16_t)buffer02[0]<< 8)| buffer2 [0]) * 2000.0 / 32768.0;
```

Code C++ (4)

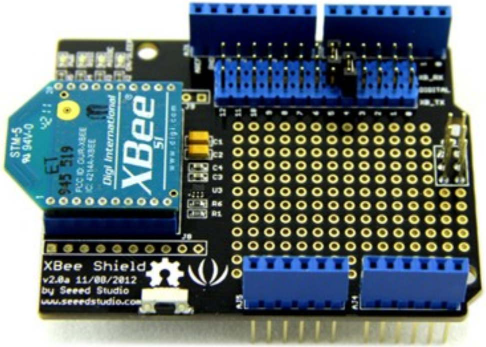


```
    i2c1_ReadRegBuffer(slaveAdd, LSM6DS3_OUTX_L_XL,  buffer, 1);

    i2c1_ReadRegBuffer(slaveAdd, LSM6DS3_OUTX_H_XL,  buffer00, 1);

    i2c1_ReadRegBuffer(slaveAdd, LSM6DS3_OUTY_L_XL,  buffer1, 1);

    i2c1_ReadRegBuffer(slaveAdd, LSM6DS3_OUTY_H_XL,  buffer01, 1);

    i2c1_ReadRegBuffer(slaveAdd, LSM6DS3_OUTZ_L_XL,  buffer2, 1);

    i2c1_ReadRegBuffer(slaveAdd, LSM6DS3_OUTZ_H_XL,  buffer02, 1);

    ax = (float)(((int16_t)buffer00[0] << 8)| buffer [0])  * (4.0 / 32768.0);

    ay = (float)(((int16_t)buffer01[0] << 8)| buffer1 [0]) * (4.0 / 32768.0);

    az = (float)(((int16_t)buffer02[0] << 8)| buffer2 [0]) * (4.0 / 32768.0);

    term_printf6("gx=%f , gy=%f, gz=%f, ax=%f, ay=%f, az=%f \n\r",gx , gy, gz, ax, ay, az);


        HAL_Delay(1000); // 1000 ms
    }
    return 0;
}
```

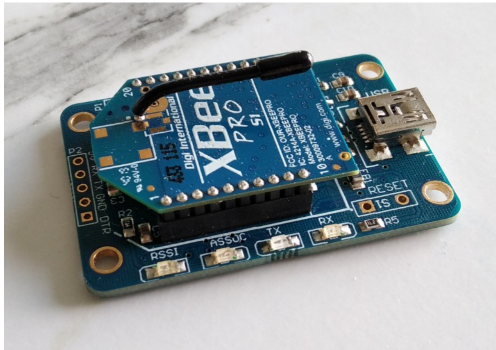
Code C++ (5)

LIAISON ZIGBEE

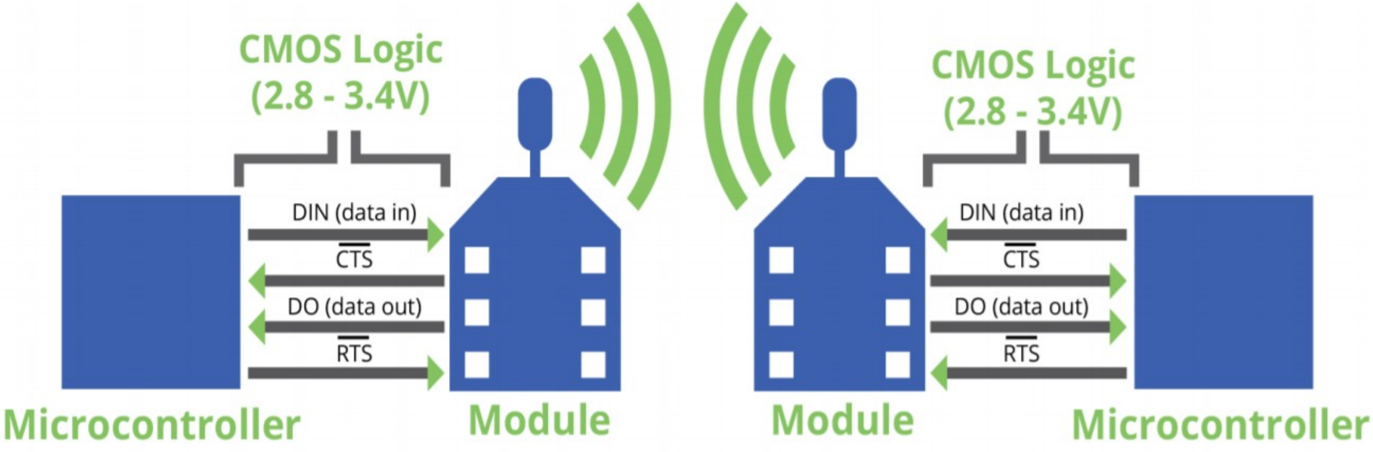
XBEE



XBEE SHIELD V2

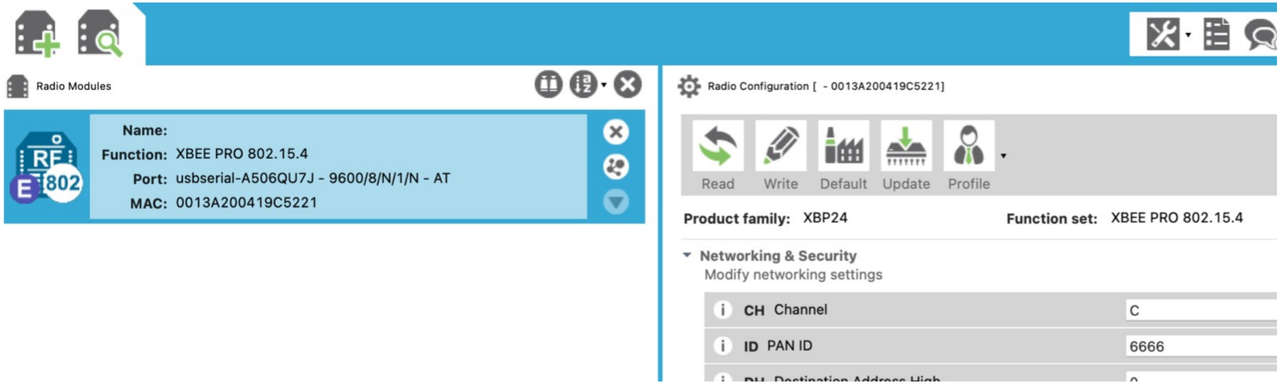


XBEE SHIELD USB



ZIGBEE datasheet

Paramétrage des identifiants



Name:
Function: XBEE PRO 802.15.4
Port: usbserial-A506QU7J - 9600/8/N/1/N - AT
MAC: 0013A200419C5221

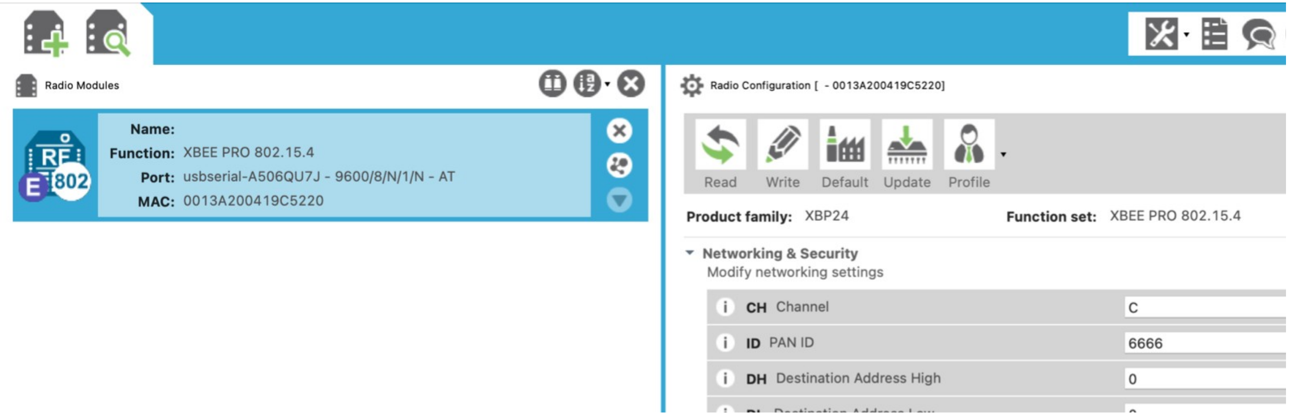
Radio Configuration [- 0013A200419C5221]

Product family: XBP24 Function set: XBEE PRO 802.15.4

Networking & Security
Modify networking settings

CH Channel C
ID PAN ID 6666
DH Destination Address High 0

Changement ID : XCTU (1)



Name:
Function: XBEE PRO 802.15.4
Port: usbserial-A506QU7J - 9600/8/N/1/N - AT
MAC: 0013A200419C5220

Radio Configuration [- 0013A200419C5220]

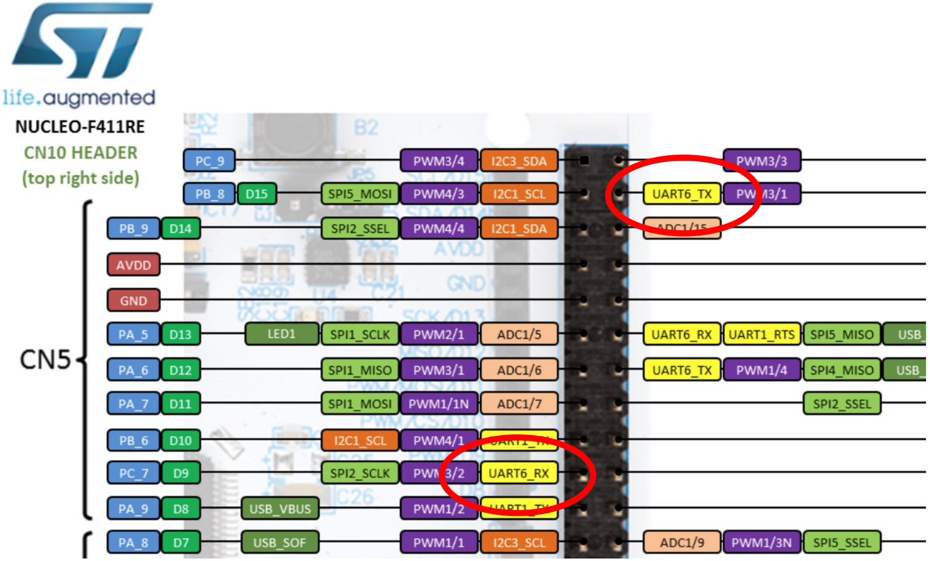
Product family: XBP24 Function set: XBEE PRO 802.15.4

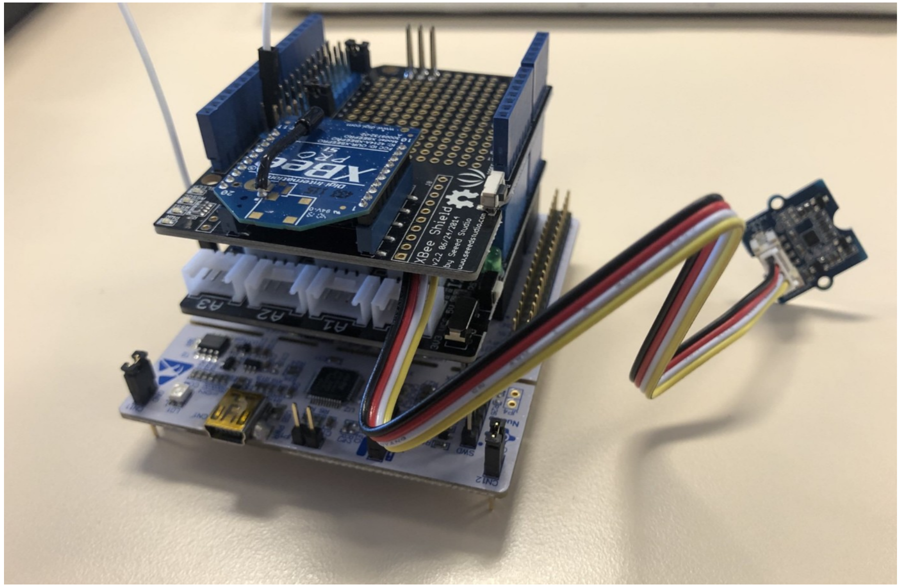
Networking & Security
Modify networking settings

CH Channel C
ID PAN ID 6666
DH Destination Address High 0

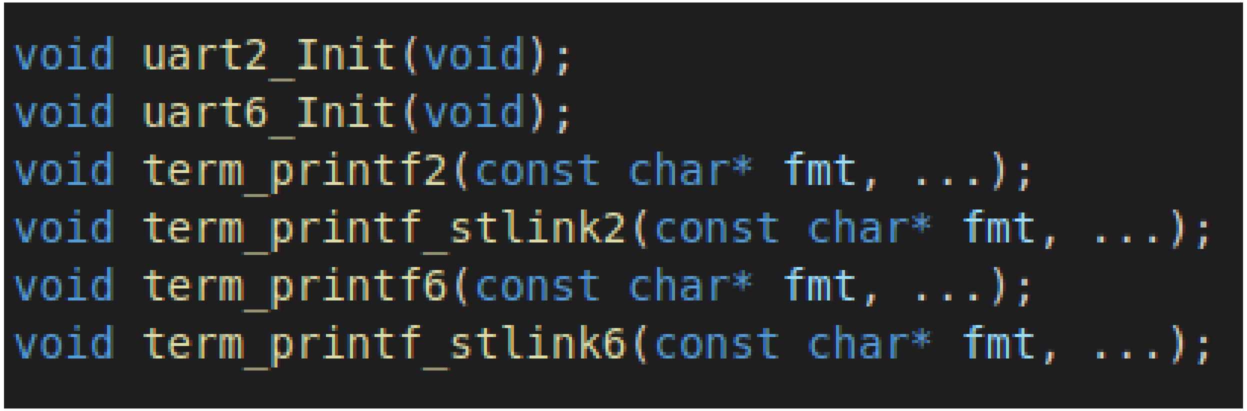
Changement ID : XCTU (2)

Hardware


life.augmented
NUCLEO-F411RE
CN10 HEADER
(top right side)
CN5
PC_9 PWM3/4 I2C3_SDA PWM3/3
PB_8 D15 SPI5_MOSI PWM4/3 I2C1_SCL UART6_TX PWM3/1
PB_9 D14 SPI2_SSEL PWM4/4 I2C1_SDA ADC1/15
AVDD
GND
PA_5 D13 LED1 SPI1_SCLK PWM2/1 ADC1/5 UART6_RX UART1_RTS SPI5_MISO USB_
PA_6 D12 SPI1_MISO PWM3/1 ADC1/6 UART6_TX PWM1/4 SPI4_MISO USB_
PA_7 D11 SPI1_MOSI PWM1/1N ADC1/7 SPI2_SSEL
PB_6 D10 I2C1_SCL PWM4/1 UART1_TX
PC_7 D9 SPI2_SCLK PWM3/2 UART6_RX
PA_9 D8 USB_VBUS PWM1/2 UART1_TX
PA_8 D7 USB_SOF PWM1/1 I2C3_SCL ADC1/9 PWM1/3N SPI5_SSEL



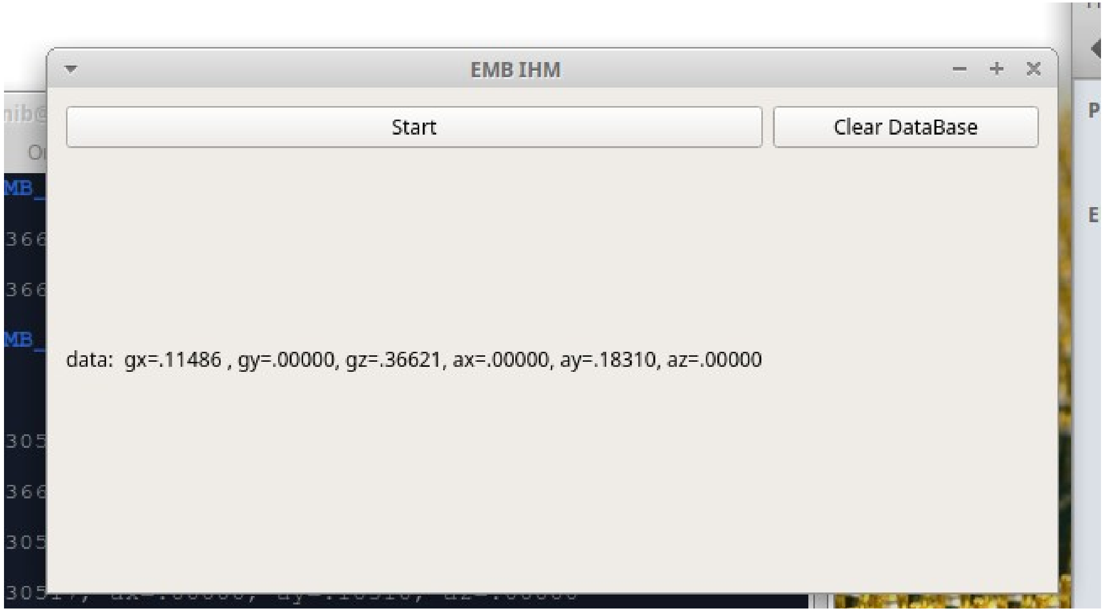
Initialisation de UART6



```c
void uart2_Init(void);
void uart6_Init(void);
void term_printf2(const char* fmt, ...);
void term_printf_stlink2(const char* fmt, ...);
void term_printf6(const char* fmt, ...);
void term_printf_stlink6(const char* fmt, ...);
```

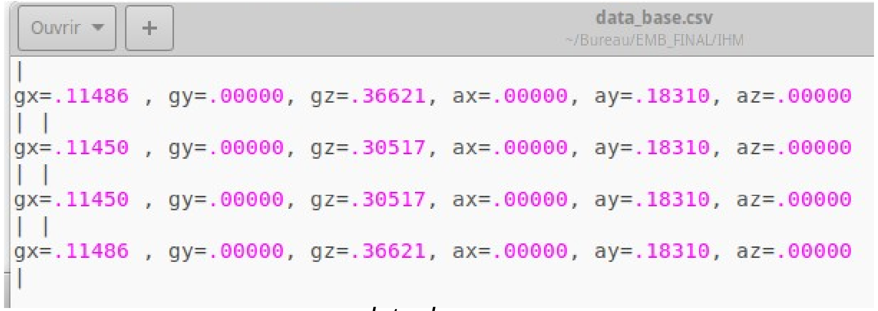
drv_uart.h

IHM



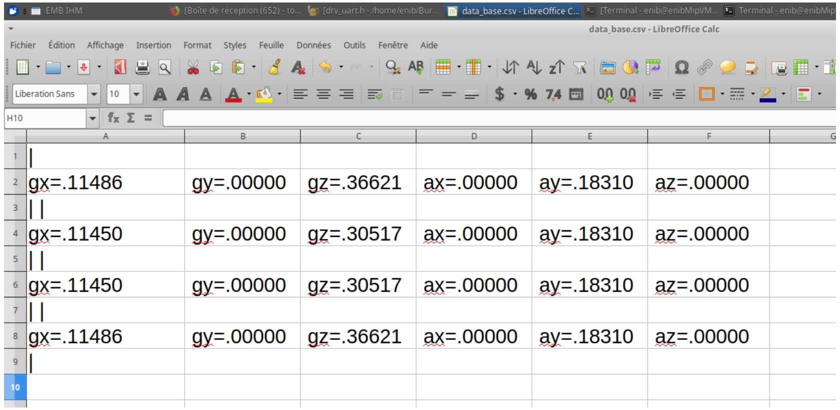
data: gx=.11486 , gy=.00000, gz=.36621, ax=.00000, ay=.18310, az=.00000

IHM.py



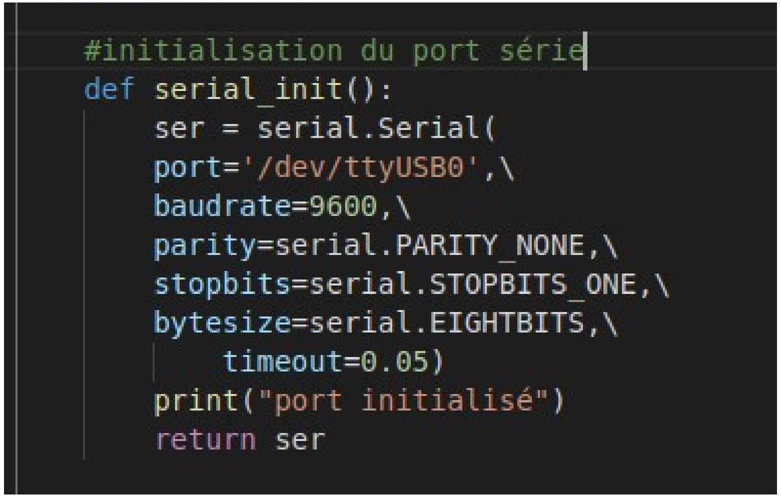
```
|
gx=.11486 , gy=.00000, gz=.36621, ax=.00000, ay=.18310, az=.00000
| |
gx=.11450 , gy=.00000, gz=.30517, ax=.00000, ay=.18310, az=.00000
| |
gx=.11450 , gy=.00000, gz=.30517, ax=.00000, ay=.18310, az=.00000
| |
gx=.11486 , gy=.00000, gz=.36621, ax=.00000, ay=.18310, az=.00000
|
```

data_base.csv

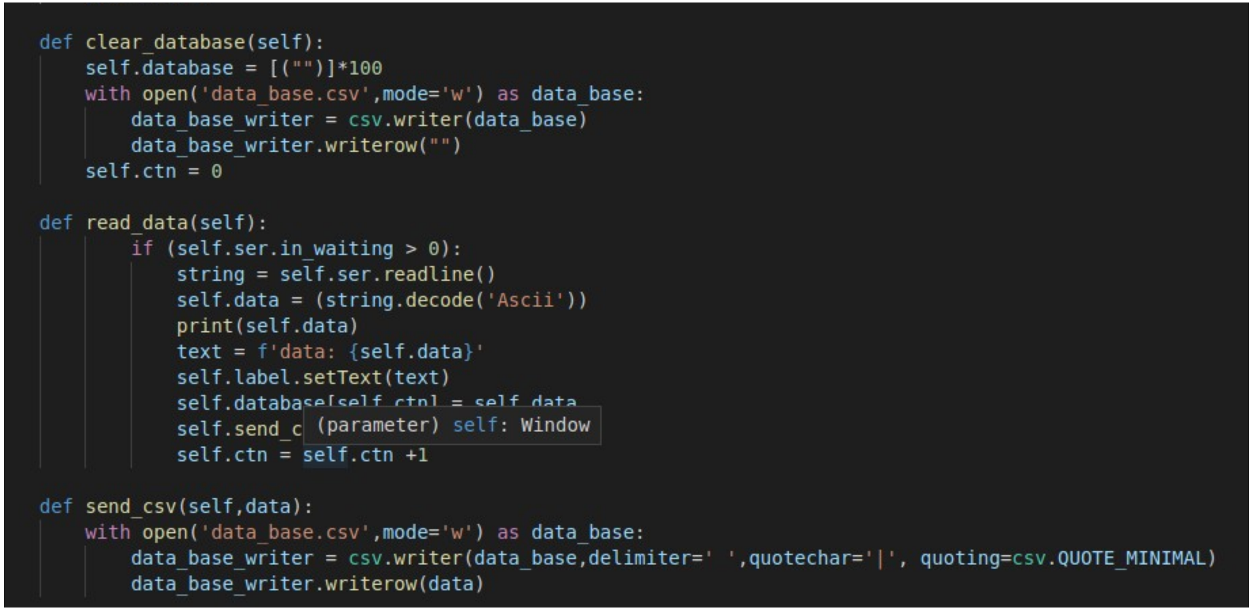


<table>
<tr><th></th><th>A</th><th>B</th><th>C</th><th>D</th><th>E</th><th>F</th></tr>
<tr><td>1</td><td>|</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>2</td><td>gx=.11486</td><td>gy=.00000</td><td>gz=.36621</td><td>ax=.00000</td><td>ay=.18310</td><td>az=.00000</td></tr>
<tr><td>3</td><td>| |</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>4</td><td>gx=.11450</td><td>gy=.00000</td><td>gz=.30517</td><td>ax=.00000</td><td>ay=.18310</td><td>az=.00000</td></tr>
<tr><td>5</td><td>| |</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>6</td><td>gx=.11450</td><td>gy=.00000</td><td>gz=.30517</td><td>ax=.00000</td><td>ay=.18310</td><td>az=.00000</td></tr>
<tr><td>7</td><td>| |</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>8</td><td>gx=.11486</td><td>gy=.00000</td><td>gz=.36621</td><td>ax=.00000</td><td>ay=.18310</td><td>az=.00000</td></tr>
<tr><td>9</td><td>|</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>10</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

data_base.csv



```
#initialisation du port série
def serial_init():
    ser = serial.Serial(
    port='/dev/ttyUSB0',\
    baudrate=9600,\
    parity=serial.PARITY_NONE,\
    stopbits=serial.STOPBITS_ONE,\
    bytesize=serial.EIGHTBITS,\
        timeout=0.05)
    print("port initialisé")
    return ser
```



```
def clear_database(self):
    self.database = [("")]*100
    with open('data_base.csv',mode='w') as data_base:
        data_base_writer = csv.writer(data_base)
        data_base_writer.writerow("")
    self.ctn = 0

def read_data(self):
        if (self.ser.in_waiting > 0):
            string = self.ser.readline()
            self.data = (string.decode('Ascii'))
            print(self.data)
            text = f'data: {self.data}'
            self.label.setText(text)
            self.database[self.ctn] = self.data
            self.send_c (parameter) self: Window
            self.ctn = self.ctn +1

def send_csv(self,data):
    with open('data_base.csv',mode='w') as data_base:
        data_base_writer = csv.writer(data_base,delimiter=' ',quotechar='|', quoting=csv.QUOTE_MINIMAL)
        data_base_writer.writerow(data)
```