1. Mettere sensitivity dei vari sensori nel main.c 🡪 OK
2. Controllare UART: stampa dei valori non è a frequenza costante 🡪 OK: Non avevo collegato la INTERRUPT dell’MPU9250!!!
3. Capire come e se cambia frequenza dell’update (sample period? Non sembra cambiare) 🡪 MPU9250\_SetSampleRateDivider OK
4. AHRS non funziona perché prima bisogna abilitare magnetometro
5. MaxSampleRate di ACC e GYRO: 1 KHz. MaxSampleRate di MAGN: 100 Hz -> plottare risultati ogni 0.01 s

**Perché non il magnetometro non veniva abilitato?**  
A fine lettura dei dati del magn contenuti tra 0x03 e 0x08, da Datasheet è necessario leggere registro ST2 (0x09) per concludere la fase di reading (leggere datasheet). In generale, corretta procedura è:

1. Leggere dispositivi connessi all’I2C bus (Presenti sia MPU9250 che AK8963). Se non presente AK8963, controllare se nello start (vedi dopo) è presente la funzione MPU9250\_EnableI2CBypass(), altrimenti aggiungerla
2. MPU9250\_Start(), al cui interno è necessario abilitare il magn MPU9250\_MAG\_Enable():

void MPU9250\_MAG\_Enable(void) {

MPU9250\_I2C\_Write(AK8963\_I2C\_ADDRESS, 0x0B,

(MPU9250\_I2C\_Read(AK8963\_I2C\_ADDRESS, 0x0B) | 0x01)); // (reset CNTL2 0x0B --> all reg reset

// For self-test:

//MPU9250\_I2C\_Write(AK8963\_I2C\_ADDRESS, 0x0C,

// (MPU9250\_I2C\_Read(AK8963\_I2C\_ADDRESS, 0x0C) | 0x40)); // (set ASTC bit6 high --> selftest

CyDelay(10);

MPU9250\_I2C\_Write(AK8963\_I2C\_ADDRESS, MPU9250\_MAG\_CNTL1\_REG, 0x0F); // Enter Fuse ROM access mode

CyDelay(10);

MPU9250\_I2C\_ReadMulti(AK8963\_I2C\_ADDRESS, 0x10, &raw\_data\_calib\_magn[0], 3); // Read the x-, y-, and z-axis calibration values

mag\_bias\_factory[0] = (float)(raw\_data\_calib\_magn[0] - 128) / 256. + 4.; // Return x-axis sensitivity adjustment values, etc.

mag\_bias\_factory[1] = (float)(raw\_data\_calib\_magn[1] - 128) / 256. + 4.;

mag\_bias\_factory[2] = (float)(raw\_data\_calib\_magn[2] - 128) / 256. + 4.;

MPU9250\_I2C\_Write(AK8963\_I2C\_ADDRESS, MPU9250\_MAG\_CNTL1\_REG, 0x00); // Power down magnetometer

CyDelay(10);

// setting the MPU9250\_MAG\_CNTL1\_REG:

// 0x00 = MAG off (default)

// 0x01 = MAG on (14-bit output)

// 0x02 = Continuous mode 1

// 0x11 = MAG on AND 16-bit output with SNGL\_MODE(0x11-> 00010001)

// 0x12 = MAG on AND 16-bit output with CNT\_MODE1, 8Hz (0x12-> 00010010)

// 0x16 = MAG on AND 16-bit output with CNT\_MODE2, 100Hz (0x16-> 00010110)

MPU9250\_I2C\_Write(AK8963\_I2C\_ADDRESS, MPU9250\_MAG\_CNTL1\_REG, 0x16); // MAG on AND 16-bit output

}

1. Dopo aver letto i dati del magnetometro (presenti nei registri 0x03 – 0x09), Leggere il registro ST2 per chiudere la lettura!!

MPU9250\_I2C\_ReadMulti(AK8963\_I2C\_ADDRESS, MPU9250\_MAG\_XOUT\_H\_REG, data + 12, 6);

end\_trasmission = MPU9250\_I2C\_Read(AK8963\_I2C\_ADDRESS, 0x09); //ST2 register (role as data reading end register)

 typedef *enum* {

        /\*\* ±2g accelerometer full scale range \*\*/

        MPU9250\_Acc\_FS\_2g,

        /\*\* ±4g accelerometer full scale range \*\*/

        MPU9250\_Acc\_FS\_4g,

        /\*\* ±8g accelerometer full scale range \*\*/

        MPU9250\_Acc\_FS\_8g,

        /\*\* ±16g accelerometer full scale range \*\*/

        MPU9250\_Acc\_FS\_16g

    } MPU9250\_Acc\_FS;

//in init:

MPU9250\_SetAccFS(MPU9250\_Acc\_FS\_2g);

*void* MPU9250\_SetAccFS(MPU9250\_Acc\_FS *fs*) {

    // Write the new full scale value in the acc conf register

    // We need to first read the current bits of the register

*uint8\_t* temp = MPU9250\_I2C\_Read(MPU9250\_I2C\_ADDRESS, MPU9250\_ACCEL\_CONFIG\_REG);

    // Then, we clear bits [4:3]

    temp &= ~MPU9250\_ACC\_FS\_MASK;

    // Lastly, we write the new byte to the register

    MPU9250\_I2C\_Write(MPU9250\_I2C\_ADDRESS, MPU9250\_ACCEL\_CONFIG\_REG, temp | ( *fs* << 3));

    // We also need to update the scaling factor

    switch(*fs*) {

    case MPU9250\_Acc\_FS\_2g:

        acc\_scale = MPU9250\_G \* 2.0f/pow(2,16)/2;

        break;

    case MPU9250\_Acc\_FS\_4g:

        acc\_scale = MPU9250\_G \* 4.0f/pow(2,16)/2;

        break;

    case MPU9250\_Acc\_FS\_8g:

        acc\_scale = MPU9250\_G \* 8.0f/pow(2,16)/2;

        break;

    case MPU9250\_Acc\_FS\_16g:

        acc\_scale = MPU9250\_G \* 16.0f/pow(2,16)/2;

        break;

    }

}