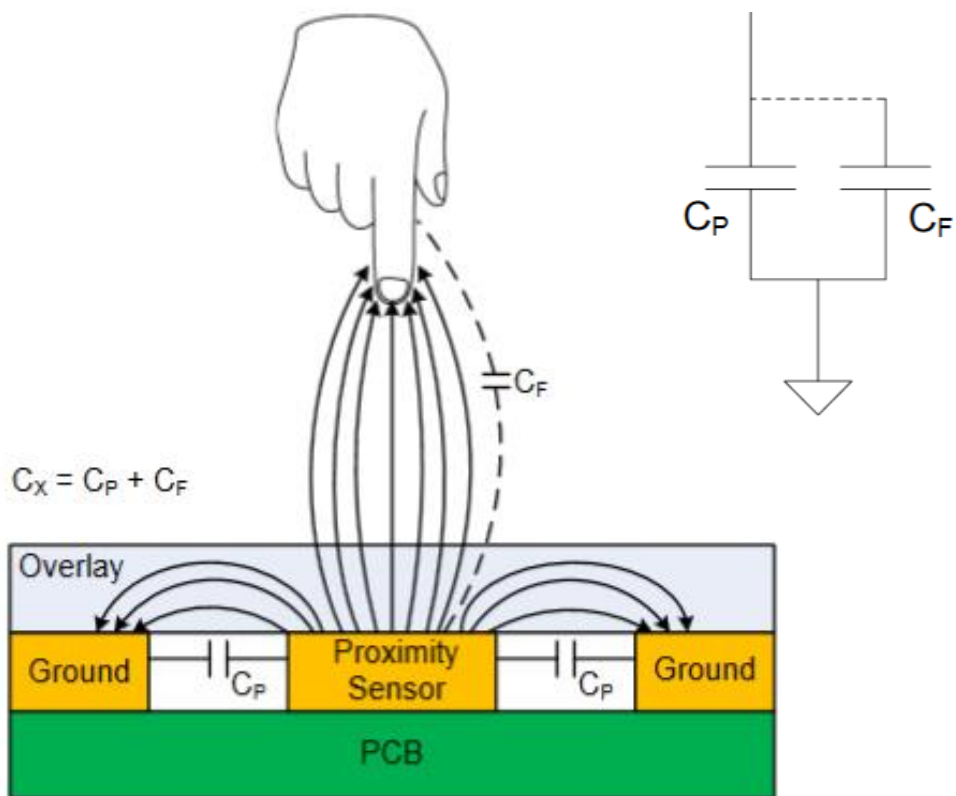


Détection de proximité

Exemples d'applications :



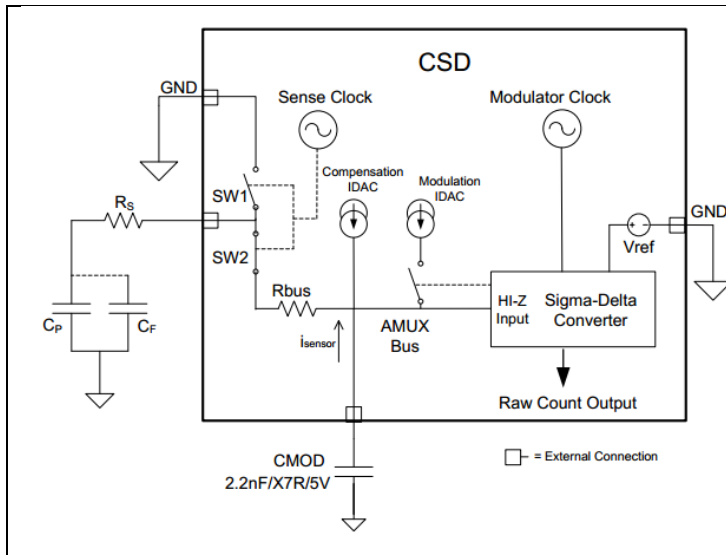
Principe physique :



Détection capacitive : lorsqu'un objet conducteur est placé à proximité des deux électrodes, la capacité totale C_x augmente et peut être détectée.

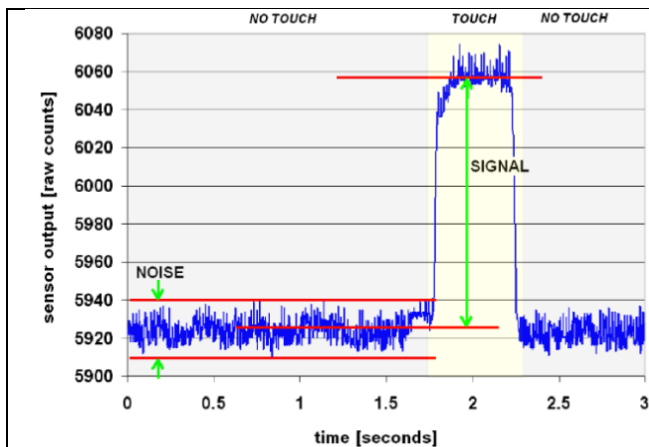
C_F : capacité ajoutée due à la proximité d'un objet

Traitement du signal :



La variation de tension analogique en sortie d'un circuit est l'image de la variation de capacité. Un modulateur génère alors un signal modulé. Un sous-système de comptage d'impulsions retourne finalement et périodiquement une valeur numérique brute (*raw count*) qui est l'image de la capacité modifiée par l'approche d'un objet à proximité du capteur.

Détection de proximité :



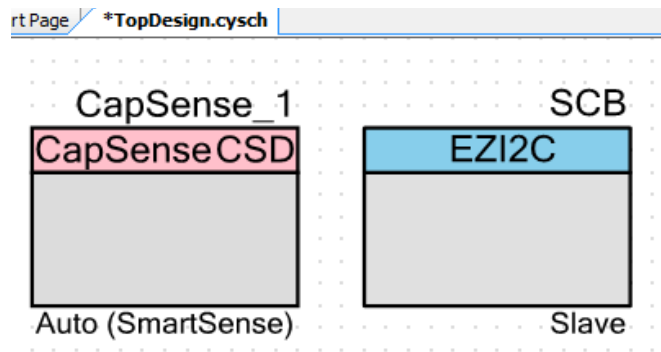
L'étude des variations de la valeur numérique *raw count* au cours du temps permet au firmware d'interpréter la proximité ou non d'un objet et d'évaluer sa distance.

Objectif de l'activité :

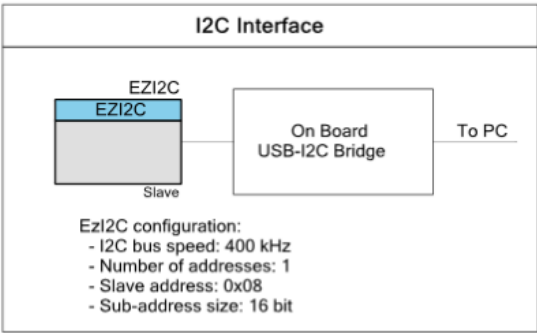
La détection de proximité d'un objet doit activer la LED verte :

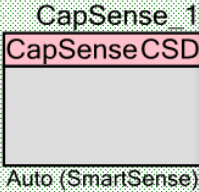
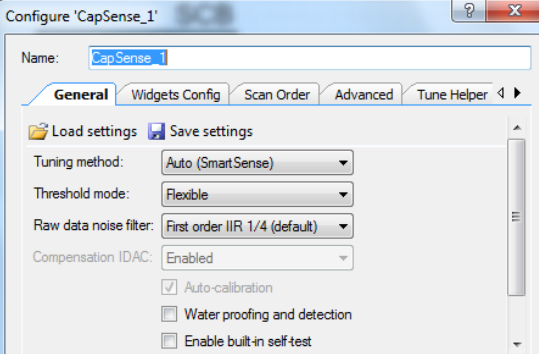
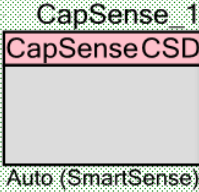
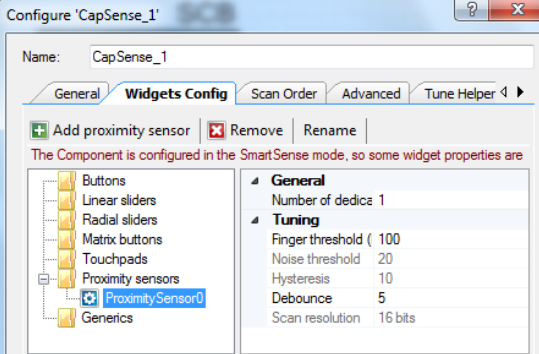
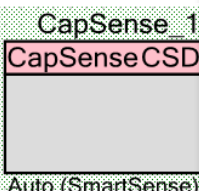
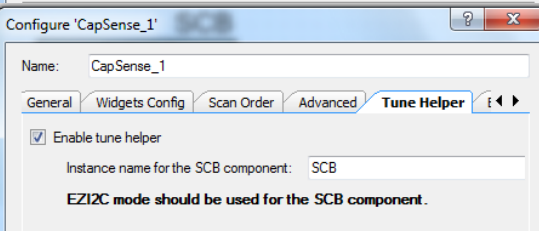


Design de départ :

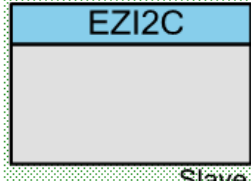


En mode *Tuning*, le PC via la liaison USB pourra requérir les données du capteur de proximité grâce à une communication I2C entre le contrôleur du capteur de proximité et le composant EZI2C en mode esclave :



 CapSense_1 CapSenseCSD Auto (SmartSense)	
 CapSense_1 CapSenseCSD Auto (SmartSense)	
 CapSense_1 CapSenseCSD Auto (SmartSense)	

- Onglet **Général** :
- Tuning method : Auto (SmartSense)
- Onglet **Widgets Config** :
- [+] Add proximity sensor
- Onglet **Tune Helper** :
- [x] Enable tune helper
- Instance name for the SCB component : SCB



SCB
EZI2C
Slave

Configure 'SCB'

Name: SCB

Configuration | EZI2C Basic | EZI2C Pins | Built-in

☐ Unconfigured SCB

☐ I2C

☒ EZI2C

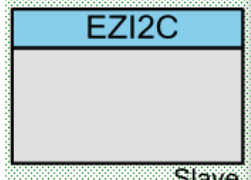
☐ SPI

☐ UART

Onglet Configuration :

Name : SCB (le même nom que dans l'onglet *TuneHelper* du composant *CapSenseCSD*)

Configuration : EZI2C



SCB
EZI2C
Slave

Configure 'SCB'

Name: SCB

Configuration | **EZI2C Basic** | EZI2C Pins | Built-in

Data rate (kbps): 400 Actual data rate (kbps): 400

☐ Clock from terminal

☒ Clock stretching

☐ Byte mode

Number of addresses: 1

Primary slave address (7-bits): 0x08

Secondary slave address (7-bits): 0x09

Sub-address size (bits): 16

☐ Enable wakeup from Deep Sleep Mode

Onglet EZI2C Basic :





Data rate (kbps) : 400

Number of addresses : 1

Primary slave address (7-bits) : 0x08

Sub-address size (bits) : 16

Brochage :

	Name	Port	Pin
	\CapSense_1:Cmod\ (Cmod)	P4 [2]	29
	\CapSense_1:Sns\ (ProximitySensor0_0__PROX)	P3 [7]	25
	\SCB:scl\	P4 [0]	27
	\SCB:sda\	P4 [1]	28

Le code initial du fichier *main.c* :

```
#include "project.h"

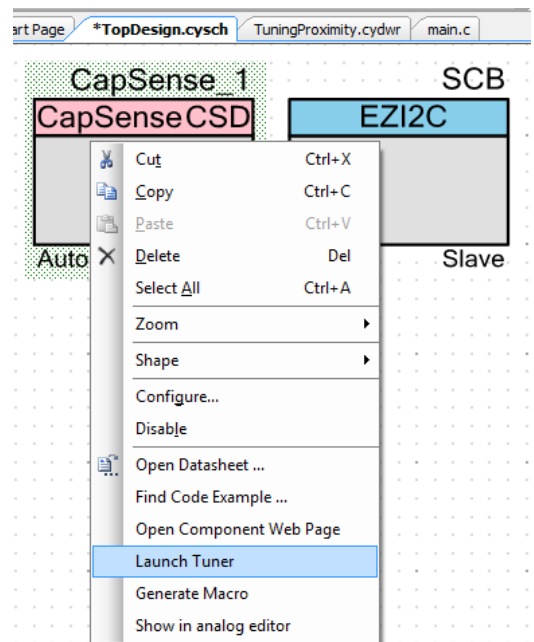
int main(void)
{
    CyGlobalIntEnable; /* Enable global interrupts. */

    CapSense_1_TunerStart();
    CapSense_1_EnableWidget(CapSense_1_PROXIMITYSENSOR0__PROX);
    /* All widgets are enabled by default except proximity widgets.
    * Proximity widgets must be manually enabled by calling
    * CapSense_1_EnableWidget() API, as their long scan time is
    * incompatible with the fast response required of other widget
    * types.
    */
    while(1)
    {
        CapSense_1_TunerComm();
    }
}

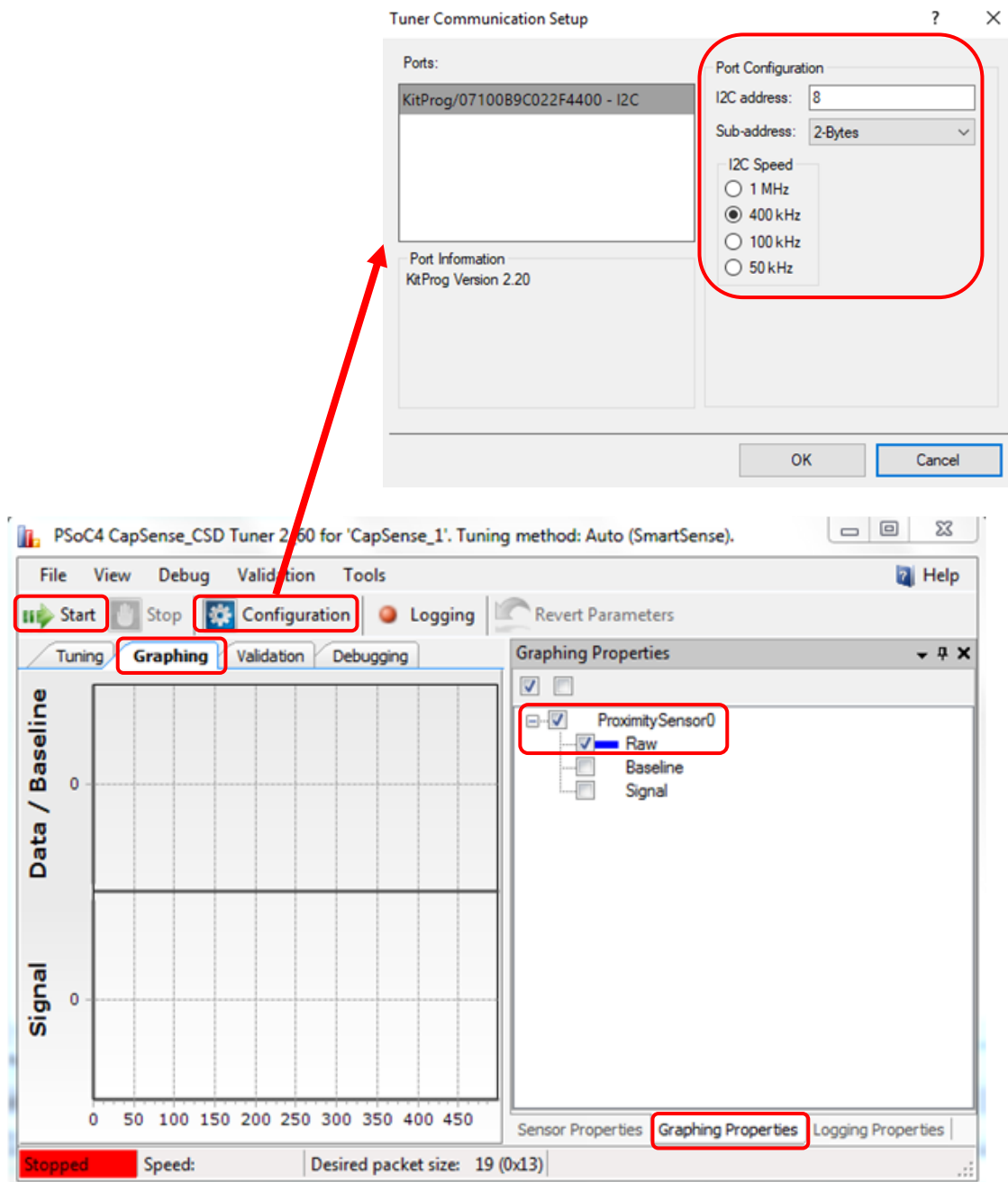
/* [] END OF FILE */
```

Premiers tests :

Après avoir transféré le programme dans la carte, activer le mode *Tuning*. Pour cela, dans la fenêtre *Design*, cliquer-droit sur le composant CapSenseCSD, puis *Launch Tuner*.

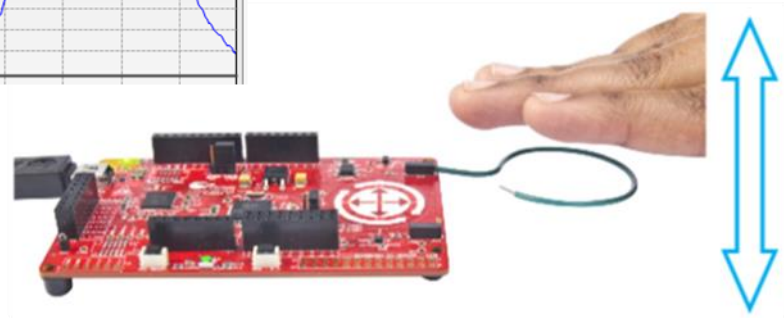
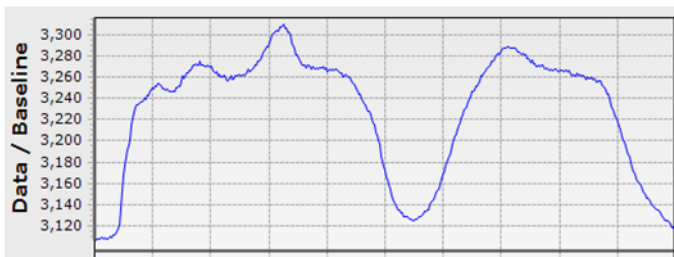


Une fenêtre s'ouvre, après configuration (bouton [Configuration]), cliquer sur l'icône **Start** :



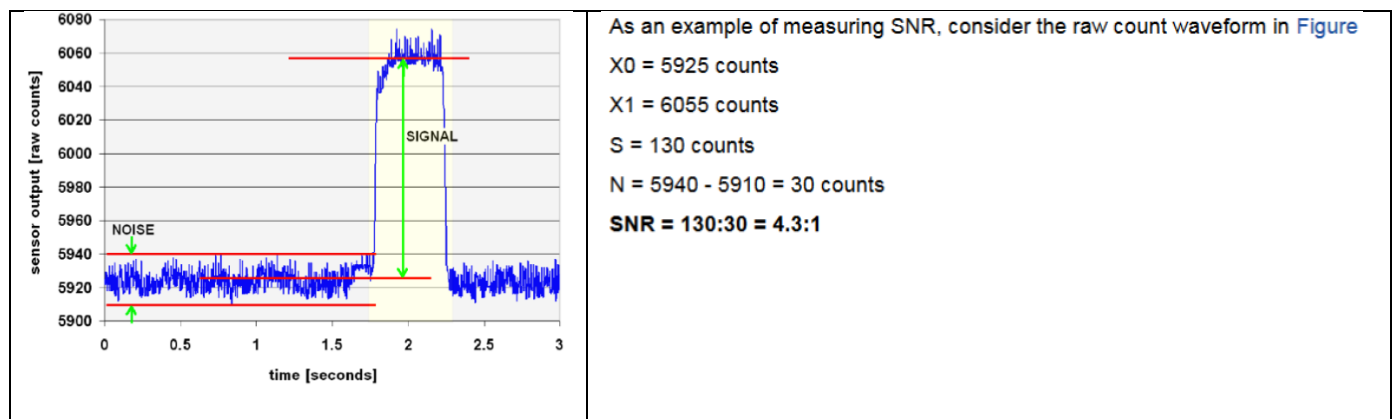
Retour des données de proximité sous forme de graphique

Le capteur de proximité est constitué d'un fil formant une boucle dont une extrémité est connectée au port *Proximity J10*. Le graphique du signal évolue en temps réel, lorsque vous approchez un obstacle...



Activité – Mesure du rapport Signal-Bruit (ou SNR pour *Signal Noise Ratio*)

Extrait documentation Cypress



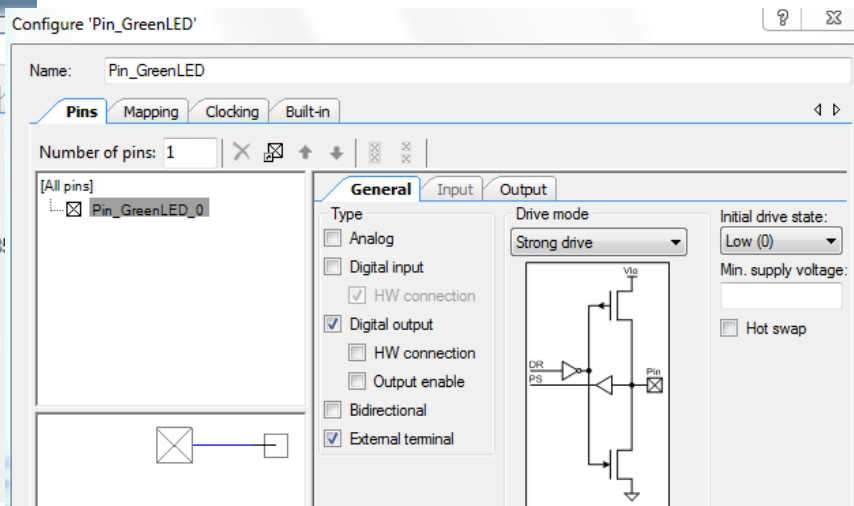
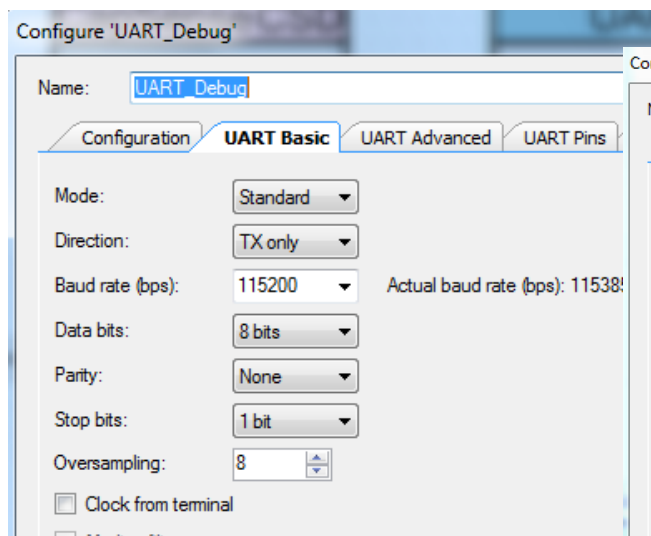
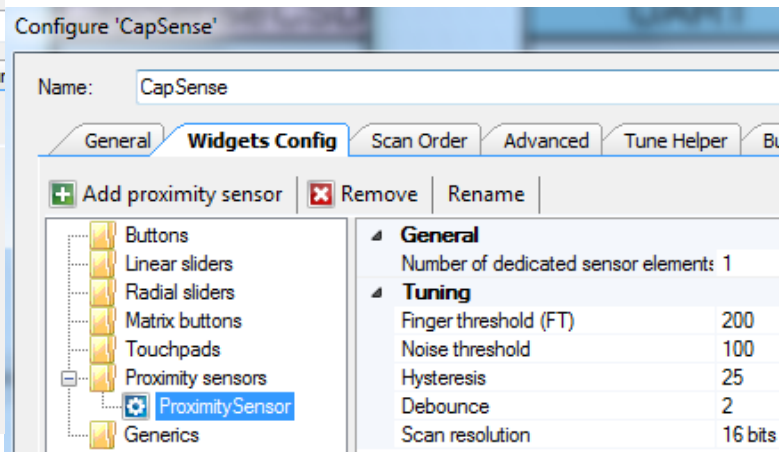
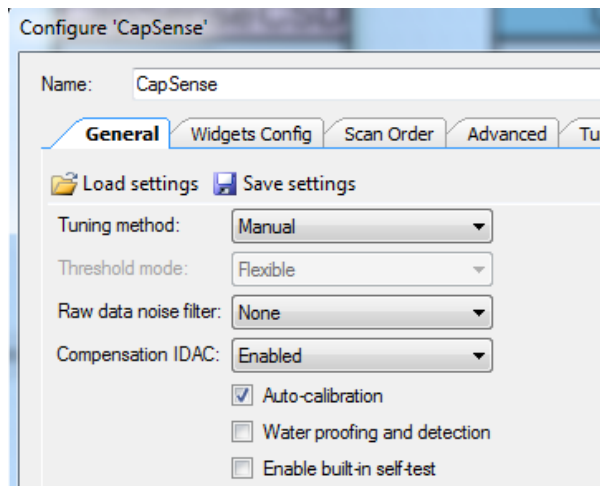
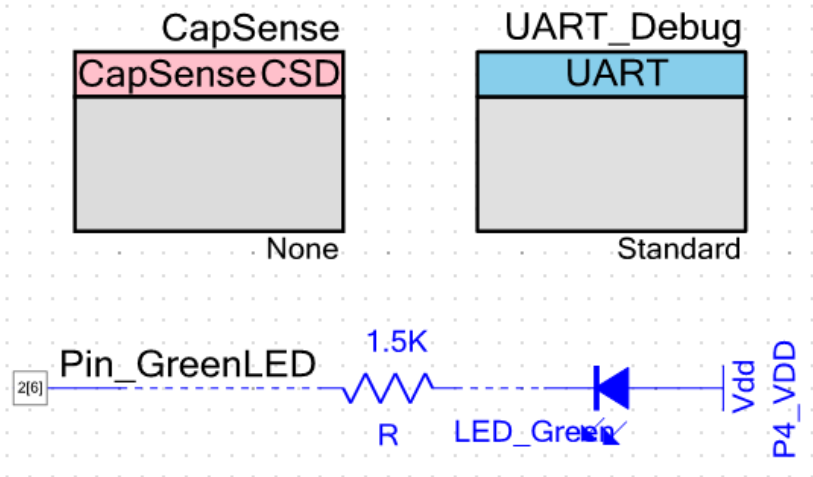
Afin d'assurer une bonne qualité de détection, le constructeur recommande un rapport Signal-Bruit (SNR) supérieur à 5:1.

A l'aide des graphiques en mode *Tuning*, déterminer le rapport Signal-Bruit (SNR) pour un obstacle (votre main tendue devant le capteur) à **5 cm** de distance. Conclusion ?

Application : La détection de proximité d'un objet doit activer la LED verte



On crée un nouveau projet avec le design ci-contre :



	Name	Port	Pin
	\CapSense:Cmod\ (Cmod)	P4 [2]	29
	\CapSense:Sns\ (ProximitySensor_0__PROX)	P3 [7]	25
	\UART_Debug:tx\	P7 [1]	38
	Pin_GreenLED	P2 [6]	8

Le code (à copier-coller) :

```
#include "project.h"
#include <stdio.h>

int main()
{
    /* Enable global interrupt. */
    CyGlobalIntEnable;
    /* Enable and start the CapSense block. */
    CapSense_Start();
    /* Initialize the baselines of the proximity sensor. */
    CapSense_InitializeSensorBaseline(CapSense_PROXIMITYSENSOR0__PROX);
    /* Update the baseline of the proximity sensor. */
    CapSense_UpdateSensorBaseline(CapSense_PROXIMITYSENSOR0__PROX);
    /* Scan the proximity sensor. */
    CapSense_ScanSensor(CapSense_PROXIMITYSENSOR0__PROX);

    UART_Debug_Start();

    for(;;)
    {
        if (!CapSense_IsBusy())
        {
            /* Check if proximity sensor is active. */
            uint8 proximity = CapSense_CheckIsSensorActive(CapSense_PROXIMITYSENSOR0__PROX);
            if (proximity)
            {
                Pin_GreenLED_Write(0); /* Switch on the LED if proximity is detected */
                /* Serial debug */
                char8 s[20];
                sprintf(s, "%u\r\n", CapSense_SensorRaw[0]);
                UART_Debug_UartPutString(s);
            }
            else /* Proximity sensor is inactive. */
            {
                /* Switch off the LED if proximity is not detected. */
                Pin_GreenLED_Write(1);
            }
            /* Update the baseline of the proximity sensor. */
            CapSense_UpdateSensorBaseline(CapSense_PROXIMITYSENSOR0__PROX);
            /* Scan the proximity sensor. */
            CapSense_ScanSensor(CapSense_PROXIMITYSENSOR0__PROX);
        }
    }
}

/* [] END OF FILE */
```

On demande :

- De tester l'application. Voir également ce qui se passe dans un terminal Série (*SerialTerm*).