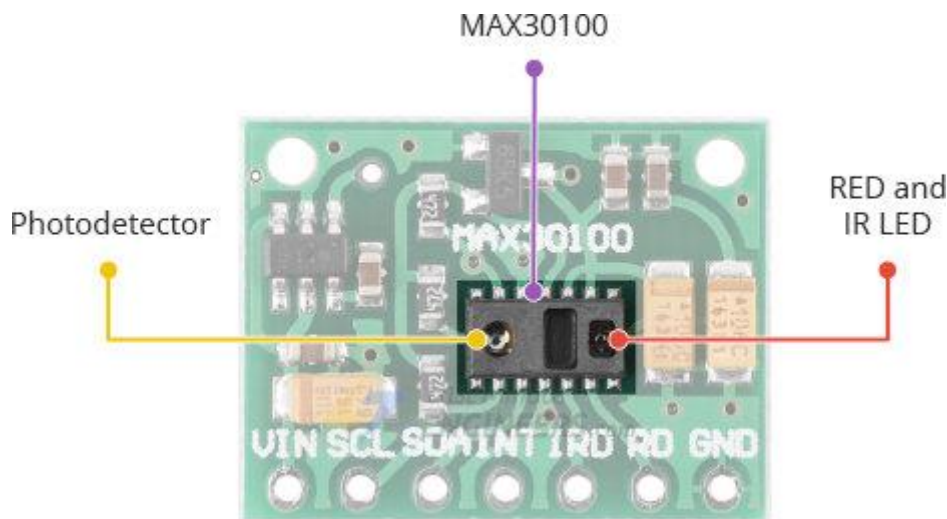


Interface entre l'oxymètre de pouls MAX30100 et le capteur de fréquence cardiaque avec Arduino

Le capteur d'oxymètre de pouls et de fréquence cardiaque MAX30100 est un capteur biométrique plug-and-play à faible consommation basé sur I2C. Il peut être utilisé par les étudiants, les amateurs, les ingénieurs, les fabricants et les développeurs de jeux et d'applications mobiles qui souhaitent intégrer des données de fréquence cardiaque en direct dans leurs projets.

Présentation du matériel du module MAX30100

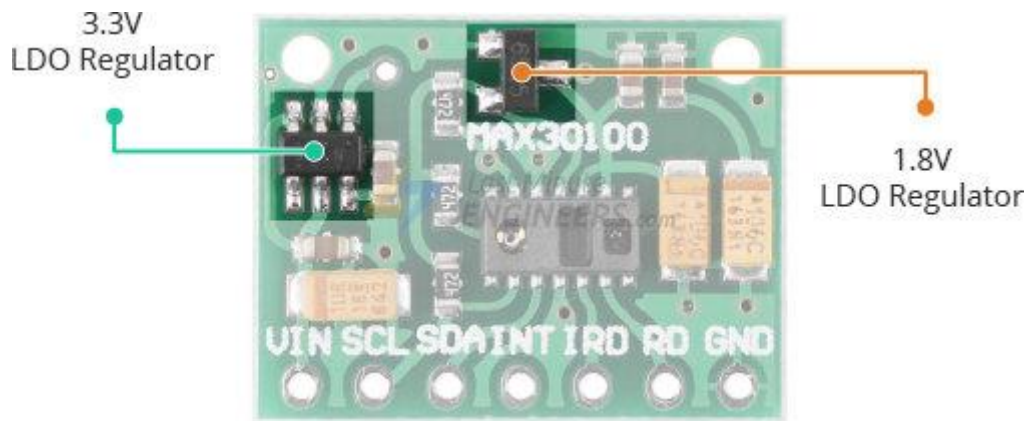
Le module est équipé du MAX30100, un circuit intégré moderne d'oxymètre de pouls et de capteur de fréquence cardiaque intégré d'Analog Devices. Il combine deux LED, un photodétecteur, une optique optimisée et un traitement de signal analogique à faible bruit pour détecter les signaux d'oxymétrie de pouls (SpO2) et de fréquence cardiaque (FC).



À droite, le MAX30100 est doté de deux LED : une LED ROUGE et une LED IR. À gauche, un photodétecteur très sensible. L'idée est de faire briller une seule LED à la fois, de détecter la quantité de lumière renvoyée vers le détecteur et, en fonction de la signature, de mesurer le taux d'oxygène dans le sang et la fréquence cardiaque.

Besoin en énergie

La puce MAX30100 nécessite deux tensions d'alimentation différentes : 1,8 V pour le circuit intégré et 3,3 V pour les LED ROUGES et IR. Le module est donc livré avec des régulateurs 3,3 V et 1,8 V. Cela vous permet de connecter le module à n'importe quel microcontrôleur avec des E/S de niveau 5 V, 3,3 V, voire 1,8 V.



L'une des caractéristiques les plus importantes du MAX30100 est sa faible consommation d'énergie : le MAX30100 consomme moins de 600 μA pendant la mesure. Il est également possible de mettre le MAX30100 en mode veille, où il ne consomme que 0,7 μA . Cette faible consommation d'énergie permet une mise en œuvre dans des appareils alimentés par batterie tels que des téléphones portables, des objets connectés ou des montres intelligentes.

Capteur de température sur puce

Le MAX30100 dispose d'un capteur de température intégré qui peut être utilisé pour compenser les changements de l'environnement et pour calibrer les mesures.

Il s'agit d'un capteur de température raisonnablement précis qui mesure la « température de la matrice » dans la plage de $-40\text{ }^{\circ}\text{C}$ à $+85\text{ }^{\circ}\text{C}$ avec une précision de $\pm 1\text{ }^{\circ}\text{C}$.

Interface I2C

Le module utilise une interface I2C simple à deux fils pour communiquer avec le microcontrôleur. Il possède une adresse I2C fixe : $0x\text{AE}_{\text{HEX}}$ (pour l'opération d'écriture) et $0x\text{AF}_{\text{HEX}}$ (pour l'opération de lecture).

Tampon FIFO

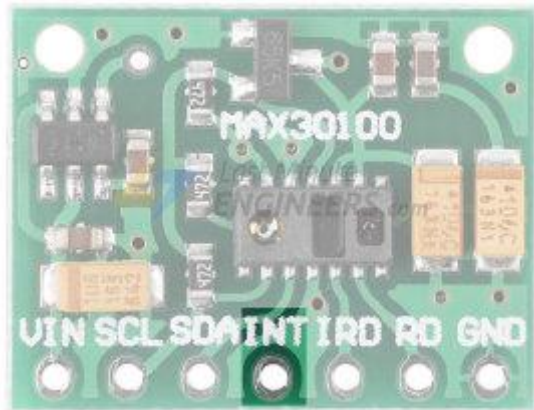
Le MAX30100 intègre un tampon FIFO pour stocker les échantillons de données. Le FIFO dispose d'une banque de mémoire de 16 échantillons, ce qui signifie qu'il peut contenir jusqu'à 16 échantillons de SpO_2 et de fréquence cardiaque. Le tampon FIFO peut décharger le microcontrôleur de la lecture de chaque nouvel échantillon de données du capteur, économisant ainsi l'énergie du système.

Interruptions

Le MAX30100 peut être programmé pour générer une interruption, permettant au microcontrôleur hôte d'effectuer d'autres tâches pendant que les données sont collectées par le capteur. L'interruption peut être activée pour 5 sources différentes :

- Power Ready : se déclenche à la mise sous tension ou après une panne de courant.
- Données SpO_2 prêtes : se déclenche après la collecte de chaque échantillon de données SpO_2 .
- Données de fréquence cardiaque prêtes : se déclenche après la collecte de chaque échantillon de données de fréquence cardiaque.

- Température prête : se déclenche lorsqu'une conversion de température de matrice interne est terminée.
- FIFO presque plein : se déclenche lorsque le FIFO devient plein et que les données futures sont sur le point d'être perdues.



La ligne INT est à drain ouvert, elle est donc tirée vers le haut par la résistance intégrée. Lorsqu'une interruption se produit, la broche INT passe au niveau bas et reste au niveau bas jusqu'à ce que l'interruption soit supprimée.

Spécifications techniques

Voici les spécifications techniques :

Alimentation électrique	3,3 V à 5,5 V
	~600µA (pendant les mesures)
Tirage actuel	~0,7 µA (en mode veille)
Longueur d'onde de la LED rouge	660 nm
Longueur d'onde de la LED IR	880 nm
Plage de température	-40 °C à +85 °C
Précision de la température	±1 °C

Vous trouverez des informations détaillées sur le capteur MAX30100 dans la fiche technique.

[Fiche technique du MAX30100](#)

Comparaison entre MAX30100 et MAX30102

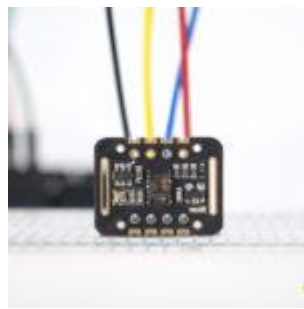
Analog Devices a franchi une nouvelle étape avec son nouveau capteur MAX30102. Le MAX30102 surpasse le MAX30100 dans les domaines suivants :

FIFO – Le MAX30102 dispose d'une banque de mémoire de 32 échantillons, ce qui signifie qu'il peut contenir jusqu'à 32 échantillons de SpO2 et de fréquence cardiaque. Alors que le MAX30100 ne peut conserver que 16 échantillons.

Résolution ADC – Le MAX30102 est capable de lire jusqu'à 18 bits ou des valeurs allant jusqu'à 262 144. Alors que le MAX30100 n'a qu'une résolution ADC de 16 bits. Grâce à cela, le MAX30102 peut détecter des mouvements extrêmement petits.

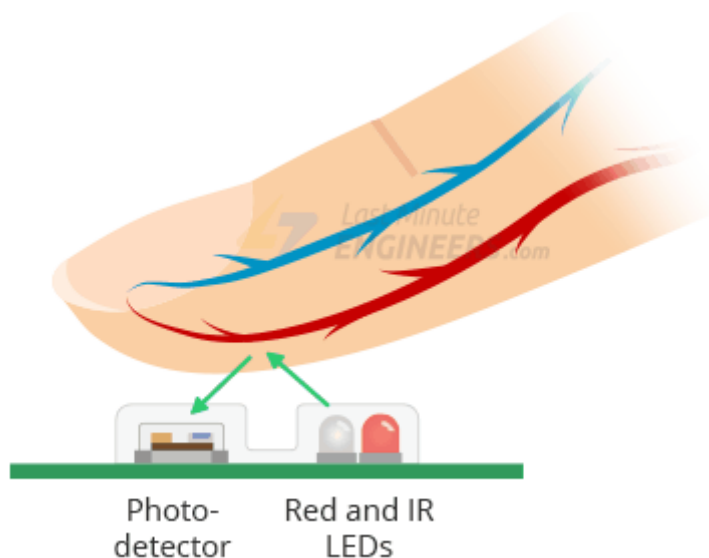
Largeur d'impulsion LED – Le MAX30102 a une largeur d'impulsion LED plus étroite que le MAX30100, ce qui entraîne une consommation d'énergie plus faible.

Pour commencer à utiliser le module MAX30102, vous pouvez vous référer au tutoriel ci-dessous



Comment fonctionnent l'oxymètre de pouls et le capteur de fréquence cardiaque MAX30100 ?

Le MAX30100, comme tout autre oxymètre de pouls optique et capteur de fréquence cardiaque, se compose d'une paire de LED haute intensité (ROUGE et IR, toutes deux de longueurs d'onde différentes) et d'un photodétecteur. Les longueurs d'onde de ces LED sont respectivement de 660 nm et 880 nm.



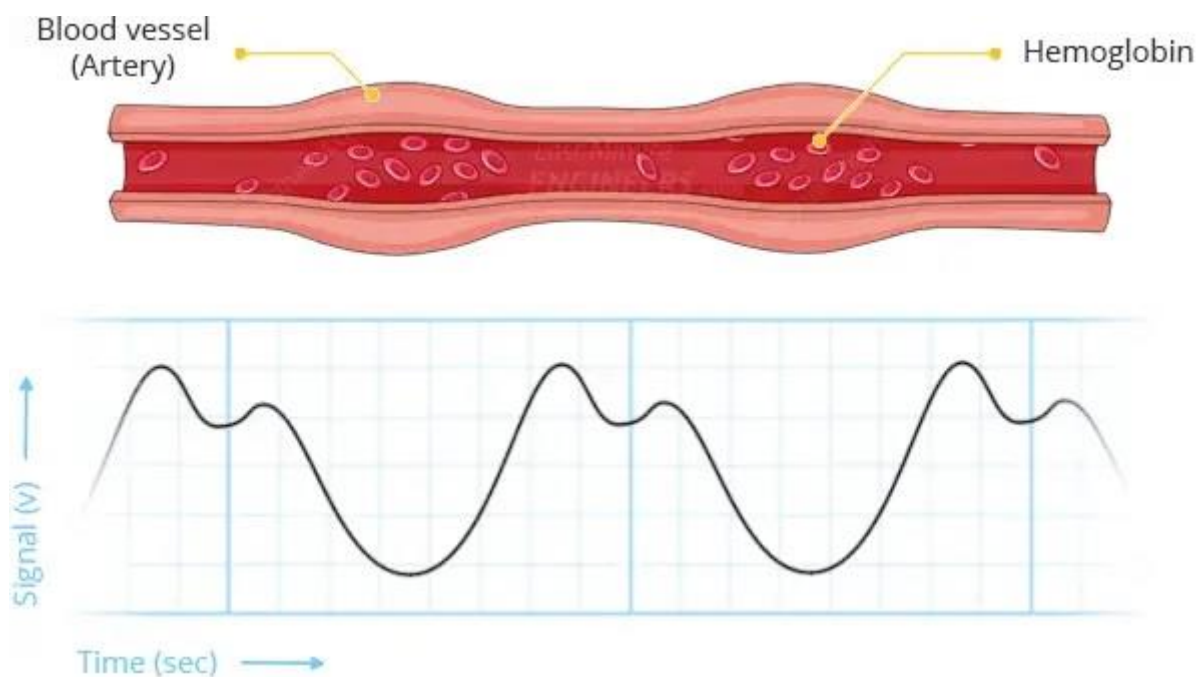
Le MAX30100 fonctionne en projetant les deux lumières sur le doigt ou le lobe de l'oreille (ou essentiellement n'importe où où la peau n'est pas trop épaisse, de sorte que les deux lumières peuvent facilement pénétrer le tissu) et en mesurant la quantité de lumière réfléchie à l'aide

d'un photodétecteur. Cette méthode de détection d'impulsions par la lumière est appelée [photopléthysmogramme](#).

Le fonctionnement du MAX30100 peut être divisé en deux parties : la mesure de la fréquence cardiaque et l'oxymétrie de pouls (mesure du niveau d'oxygène dans le sang).

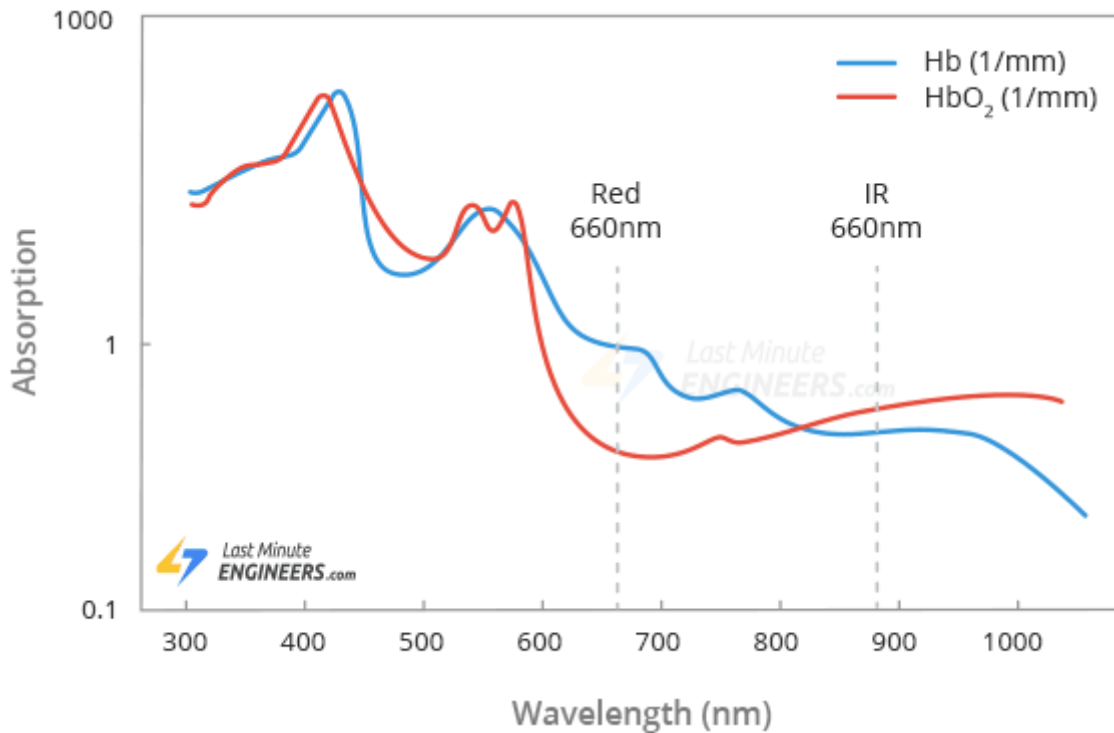
Mesure de la fréquence cardiaque

L'hémoglobine oxygénée (HbO_2) présente dans le sang artériel a la particularité d'absorber la lumière infrarouge. Plus le sang est rouge (plus le taux d'hémoglobine est élevé), plus la lumière infrarouge est absorbée. À mesure que le sang est pompé à travers le doigt à chaque battement de cœur, la quantité de lumière réfléchie change, créant une forme d'onde changeante à la sortie du photodétecteur. À mesure que vous continuez à faire briller la lumière et à prendre des mesures avec le photodétecteur, vous commencez rapidement à obtenir une mesure du pouls cardiaque (FC).



Oxymétrie de pouls

L'oxymétrie de pouls repose sur le principe selon lequel la quantité de lumière ROUGE et IR absorbée varie en fonction de la quantité d'oxygène dans votre sang. Le graphique suivant présente le spectre d'absorption de l'hémoglobine oxygénée (HbO_2) et de l'hémoglobine désoxygénée (Hb).



Comme vous pouvez le voir sur le graphique, le sang désoxygéné absorbe plus de lumière ROUGE (660 nm), tandis que le sang oxygéné absorbe plus de lumière IR (880 nm). En mesurant le rapport entre la lumière IR et la lumière ROUGE reçue par le photodétecteur, le niveau d'oxygène (SpO₂) dans le sang est calculé.

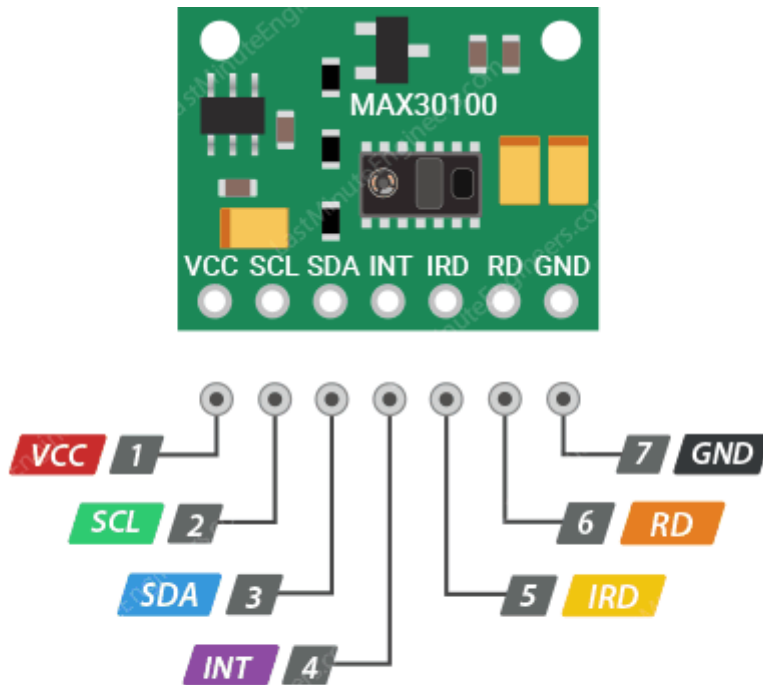
Saviez-vous?

La mesure de la saturation en oxygène de l'hémoglobine (HbO₂) en mesurant l'absorbance de la lumière rouge et IR a été introduite en 1935 par un médecin allemand, Karl Matthes.

Au début, il n'existait pas de bons photodétecteurs, c'est pourquoi on utilisait la bande verte du spectre lumineux au lieu de la bande infrarouge. Avec l'évolution de la technologie, des méthodes plus fiables ont été développées et la lumière verte a été remplacée par la lumière infrarouge.

Brochage du module MAX30100

Le module MAX30100 fait ressortir les connexions suivantes.



MAX30100 Module Pinout



VCC c'est la broche d'alimentation. Vous pouvez la connecter à la sortie 3,3 V ou 5 V de votre Arduino.

SCL c'est la broche d'horloge I2C, connectez-la à la ligne d'horloge I2C de votre Arduino.

SDA c'est la broche de données I2C, connectez-vous à la ligne de données I2C de votre Arduino.

INT Le MAX30100 peut être programmé pour générer une interruption à chaque impulsion. Cette ligne est à drain ouvert, elle est donc tirée vers le HAUT par la résistance intégrée. Lorsqu'une interruption se produit, la broche INT passe au BAS et reste BASSE jusqu'à ce que l'interruption soit supprimée.

IRD Le MAX30100 intègre un pilote LED pour piloter les impulsions LED pour les mesures SpO2 et HR. Utilisez-le si vous souhaitez piloter vous-même la LED IR, sinon laissez-la déconnectée.

RD La broche est similaire à la broche IRD, mais elle est utilisée pour piloter la LED rouge. Si vous ne souhaitez pas piloter la LED rouge vous-même, laissez-la déconnectée.

GND c'est le sol.

MAX30100 ne fonctionne pas – Problème et solutions pour le résoudre

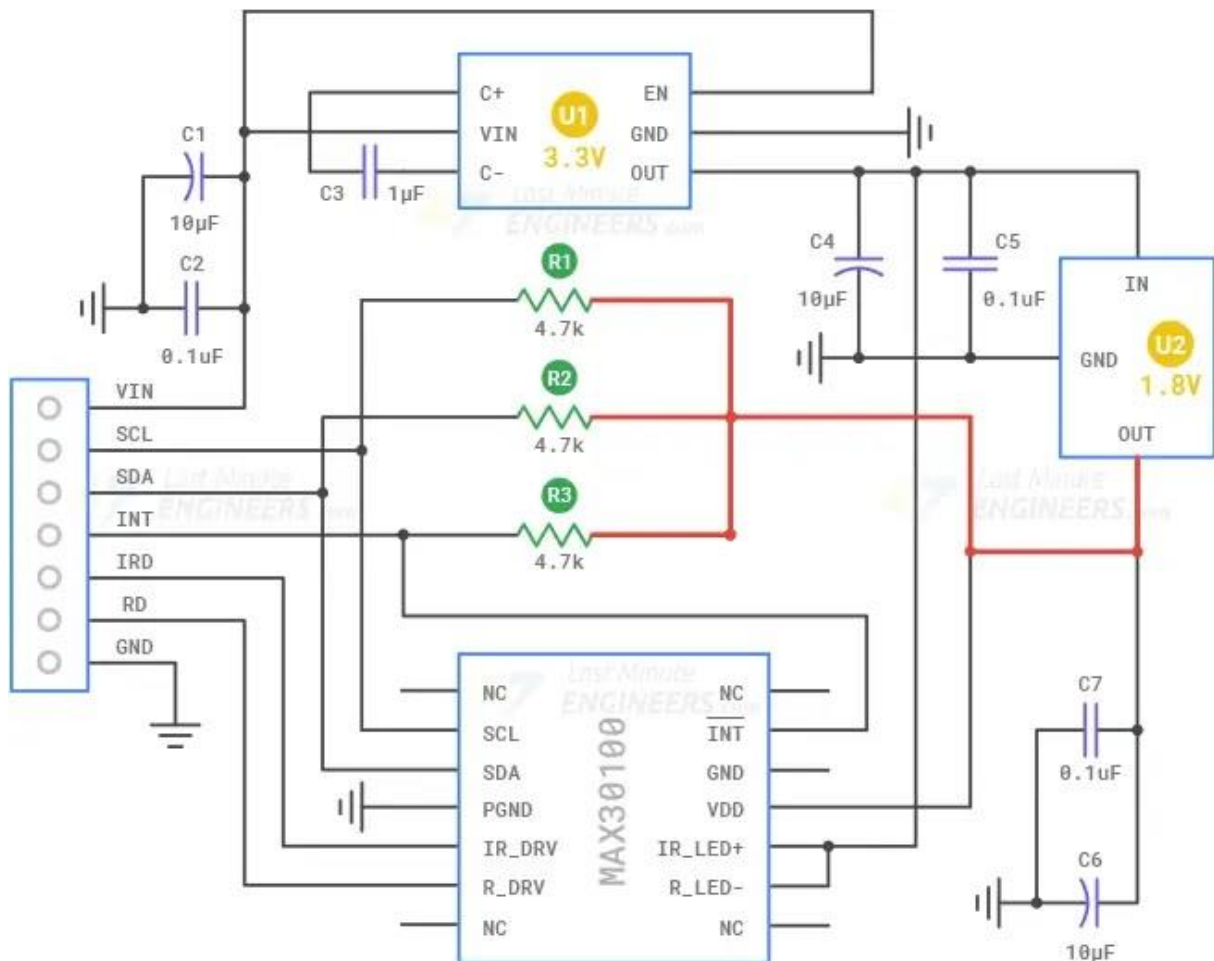
Il ne fait aucun doute que le module MAX30100 est bon marché et très populaire parmi les amateurs, mais malheureusement, le module (vendu par milliers) présente un sérieux problème de conception. Eh bien, réglons-le.

Le problème

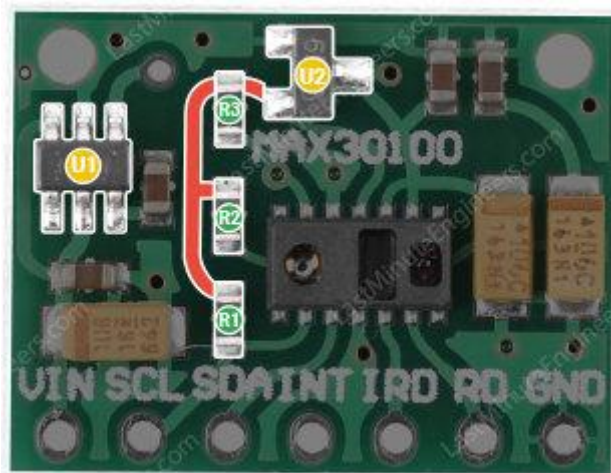
La puce MAX30100 nécessite deux tensions d'alimentation différentes : 1,8 V pour le circuit intégré et 3,3 V pour les LED ROUGES et IR. Le module est donc livré avec deux régulateurs de

tension linéaires – U1 et U2. Le premier génère de 5 V à 3,3 V. Le deuxième régulateur est connecté à la sortie du premier et génère 1,8 V.

Regardons maintenant de plus près les résistances de rappel de 4,7 k Ω (R1, R2 et R3) pour les lignes de signal SCL, SDA et INT. Elles sont connectées à l'alimentation 1,8 V (mises en évidence par une ligne rouge épaisse) !



Les deux régulateurs U1, U2 et la trace qui relie les résistances pull-up de 4,7 k Ω à l'alimentation de 1,8 V sont également mis en évidence sur le module.



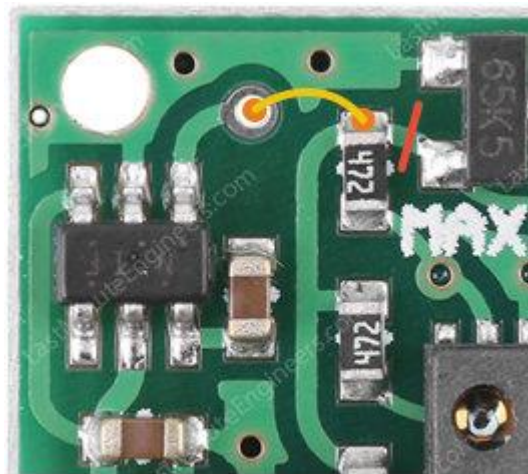
Si vous connectez un tel module à la logique 5 V de l'Arduino, il n'apparaîtra pas sur le bus I2C car le niveau logique est trop faible (Arduino signale HIGH si la tension dépasse 3,0 V pour les cartes 5 V et 2,0 V pour les cartes 3,3 V). Même avec une carte logique 3,3 V, cela ne fonctionnera pas.

Ne vous inquiétez pas ! Nous avons deux solutions pour résoudre ce problème.

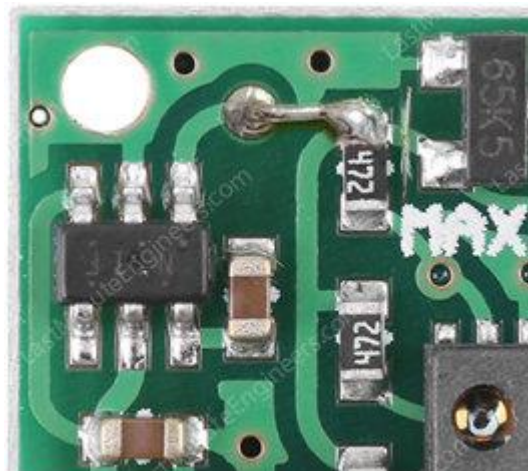
Solution 1 (testée) :

1. Coupez la trace à la place de la ligne rouge. Cela déconnectera toutes les résistances de rappel de 4,7 k Ω de la tension d'alimentation de 1,8 V.
2. Créez maintenant un cavalier comme indiqué par la ligne jaune avec un morceau de fil ou une goutte de soudure. Cela permettra de tirer toutes les résistances de 4,7 k Ω jusqu'à 3,3 V.

Voici la carte avec et sans les modifications :

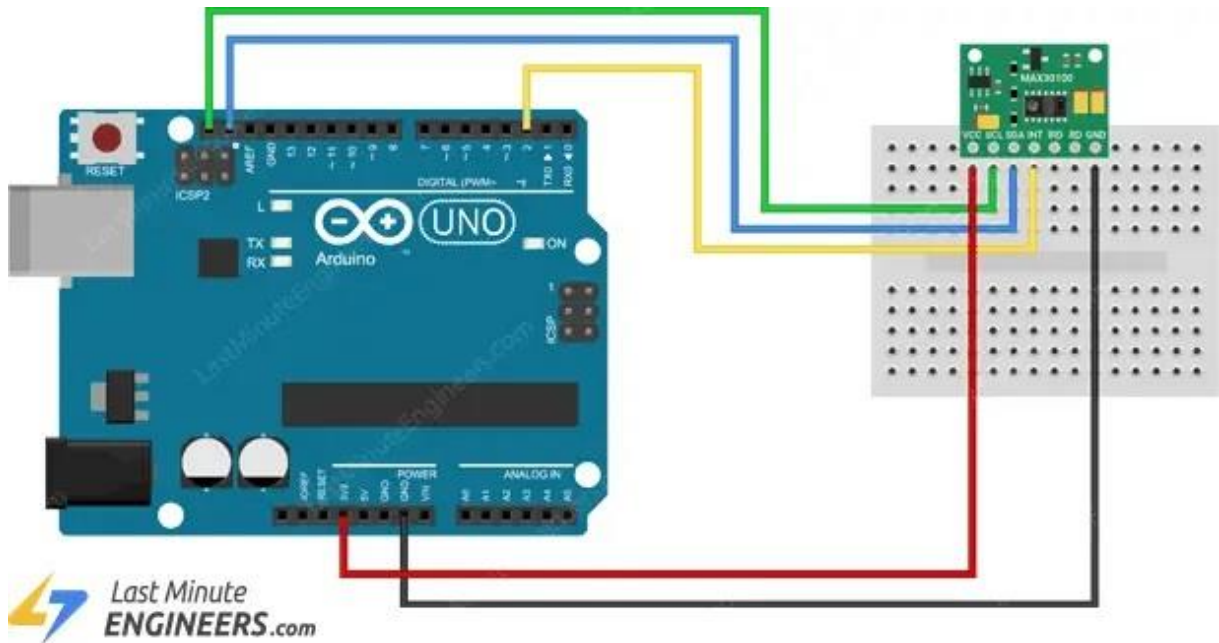


Before



After

Après avoir suivi ces étapes, connectez le module à l'Arduino UNO en utilisant le schéma de câblage ci-dessous :

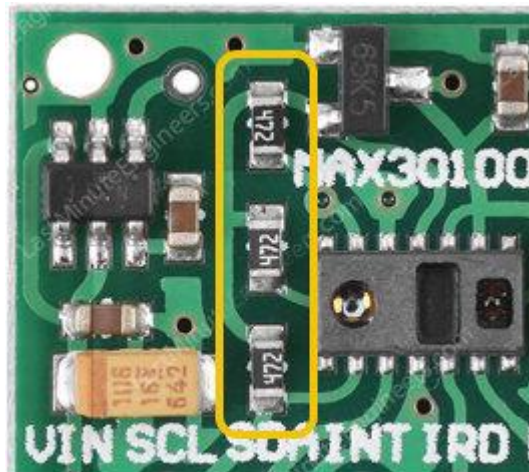


Le câblage est assez simple. Il suffit de connecter la broche VIN au 3,3 V de l'Arduino et GND à la terre. Connectez la broche SCL à la broche d'horloge I2C et la broche SDA à la broche de données I2C de votre Arduino. Enfin, connectez la broche INT à la broche numérique 2.

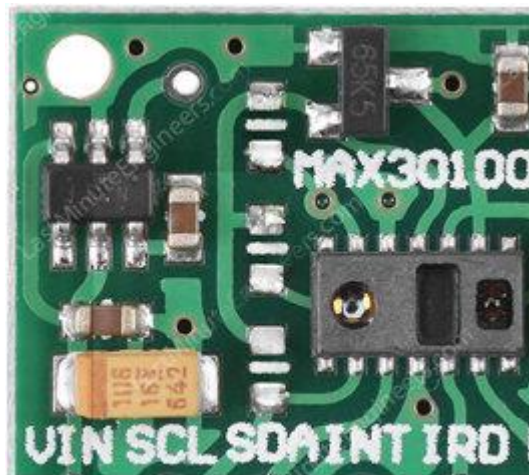
Solution 2 (testée) :

La deuxième solution consiste à retirer toutes les résistances de rappel de 4,7 kΩ de la carte. Nous utiliserons ultérieurement des résistances de rappel externes sur les lignes de signal SCL, SDA et INT.

Voici la carte avec et sans les résistances retirées :

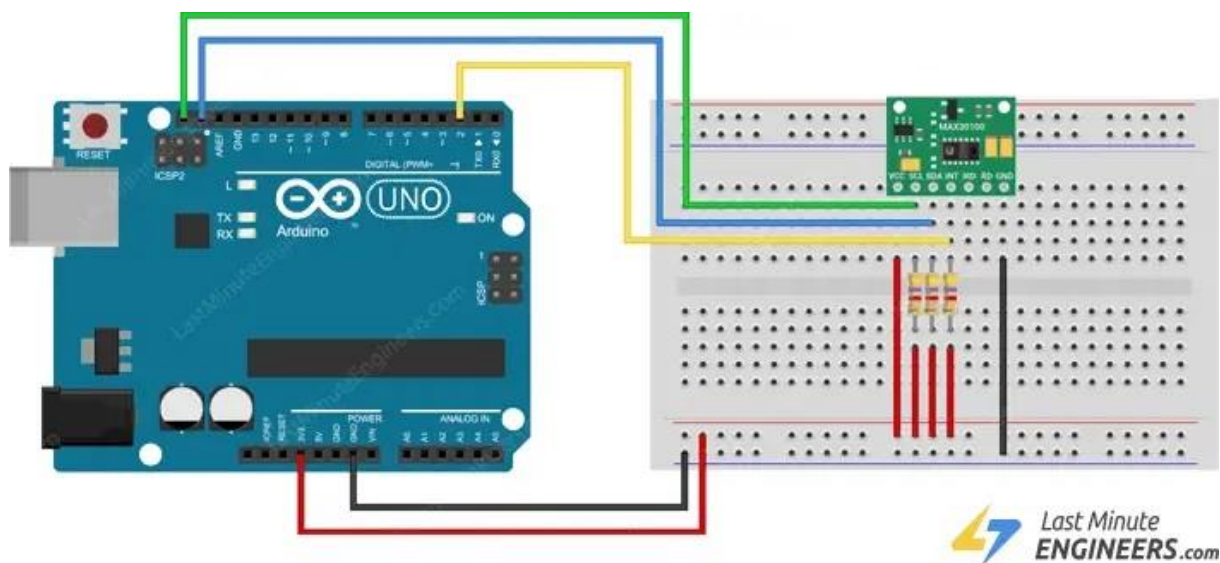


Before



After

Après avoir retiré les résistances, connectez le module à l'Arduino UNO en utilisant le schéma de câblage ci-dessous :



Le câblage est le même que le premier ; il suffit d'ajouter des résistances de rappel externes de 4,7 k Ω sur les lignes de signal SCL, SDA et INT.

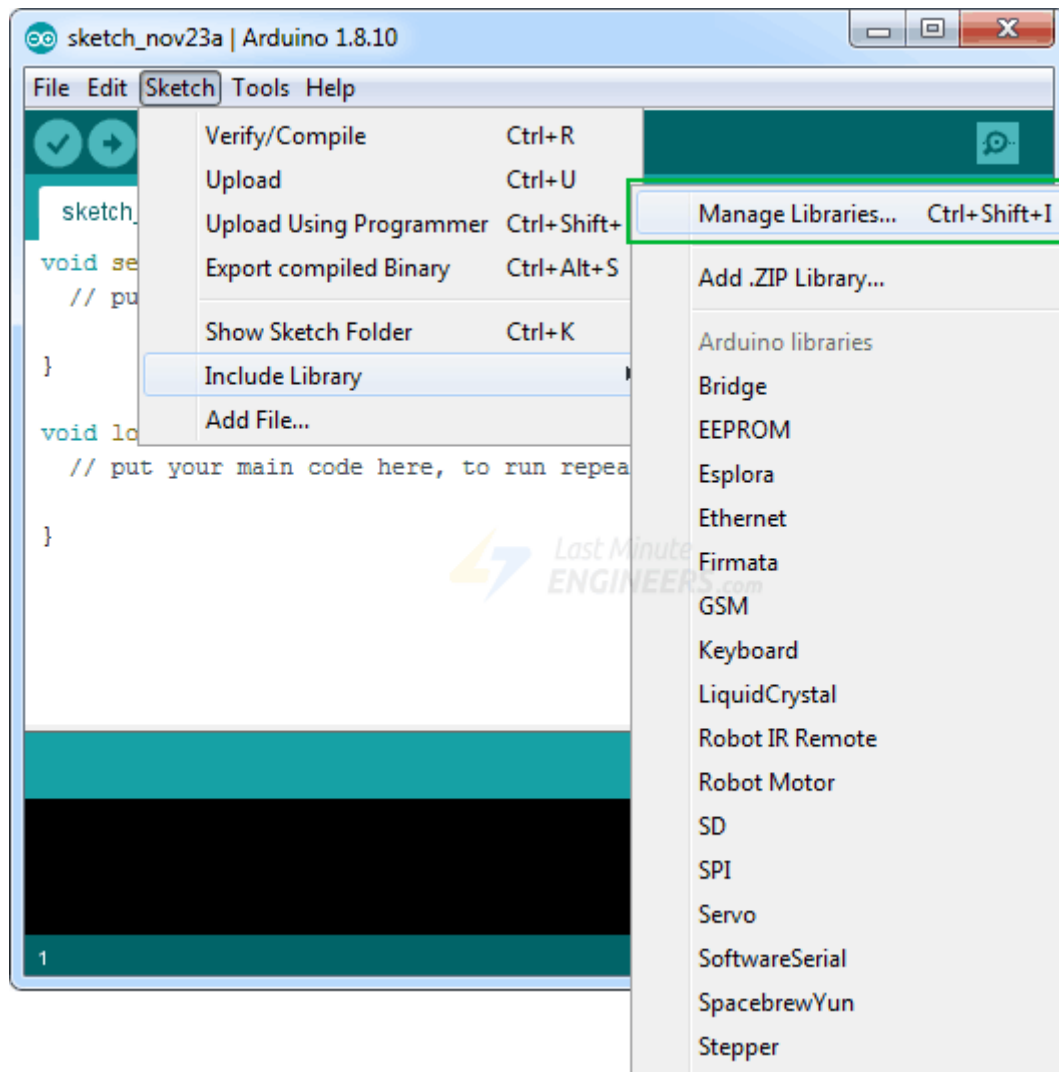
Notre évaluation :

Nous avons testé les deux solutions et elles fonctionnent parfaitement. Essayez donc celle qui vous semble la plus facile à mettre en œuvre.

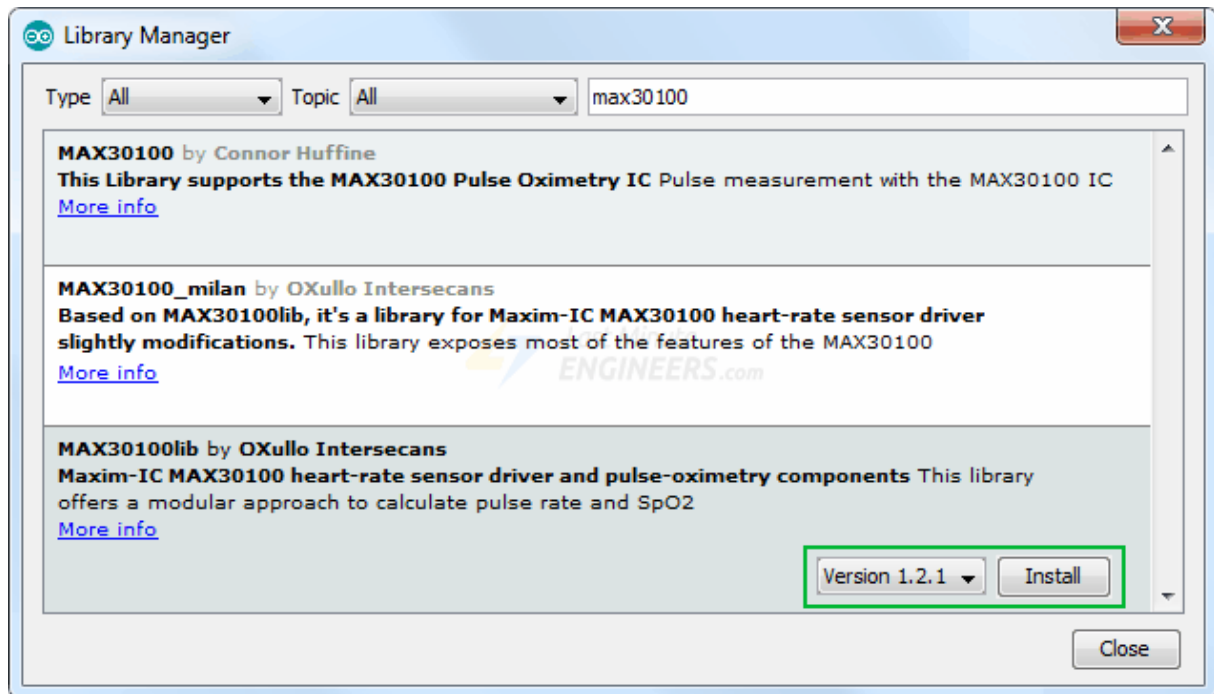
Installation de la bibliothèque

Il existe plusieurs bibliothèques disponibles pour le capteur MAX30100. Cependant, dans notre exemple, nous utilisons celle d' [OXullo Intersecans](#) . Cette bibliothèque expose la plupart des fonctionnalités du MAX30100 et offre des fonctions simples et faciles à utiliser pour calculer la fréquence du pouls et la SpO2. Vous pouvez télécharger cette bibliothèque à partir du gestionnaire de bibliothèques Arduino IDE.

Pour installer la bibliothèque, accédez à Sketch > Inclure la bibliothèque > Gérer les bibliothèques... Attendez que le gestionnaire de bibliothèque télécharge l'index des bibliothèques et mette à jour la liste des bibliothèques installées.



Filtrez votre recherche en tapant max30100 . Il devrait y avoir plusieurs entrées. Recherchez MAX30100lib Library by OXullo Intersecans . Cliquez sur l'entrée, puis sélectionnez Installer.



Exemple 1 – Mesure de la fréquence cardiaque (BPM) et de la saturation en oxygène (SpO2)

C'est ici que le plaisir commence vraiment ! Dans cet exemple, nous allons mesurer la fréquence cardiaque (battements par minute ou BPM) et le taux d'oxygène dans le sang (SpO2) de la personne que nous surveillons.

Avertissement:

Ce schéma détecte la fréquence cardiaque et la saturation en oxygène de manière optique. Cette méthode est délicate et peut donner des résultats erronés. Veuillez donc NE PAS l'utiliser pour un véritable diagnostic médical.

```
#include <Wire.h>
```

```
#include "MAX30100_PulseOximeter.h"
```

```
#define REPORTING_PERIOD_MS 1000
```

```
// Create a PulseOximeter object
```

```
PulseOximeter pox;
```

```
// Time at which the last beat occurred
```

```
uint32_t tsLastReport = 0;
```

```
// Callback routine is executed when a pulse is detected
```

```
void onBeatDetected() {  
    Serial.println("Beat!");  
}  
  
void setup() {  
    Serial.begin(9600);  
  
    Serial.print("Initializing pulse oximeter..");  
  
    // Initialize sensor  
    if (!pox.begin()) {  
        Serial.println("FAILED");  
        for(;;);  
    } else {  
        Serial.println("SUCCESS");  
    }  
  
    // Configure sensor to use 7.6mA for LED drive  
    pox.setIRLedCurrent(MAX30100_LED_CURR_7_6MA);  
  
    // Register a callback routine  
    pox.setOnBeatDetectedCallback(onBeatDetected);  
}  
  
void loop() {  
    // Read from the sensor  
    pox.update();  
  
    // Grab the updated heart rate and SpO2 levels  
    if (millis() - tsLastReport > REPORTING_PERIOD_MS) {  
        Serial.print("Heart rate:");  
    }  
}
```

```

Serial.print(pox.getHeartRate());

Serial.print("bpm / SpO2:");

Serial.print(pox.getSpO2());

Serial.println("%");

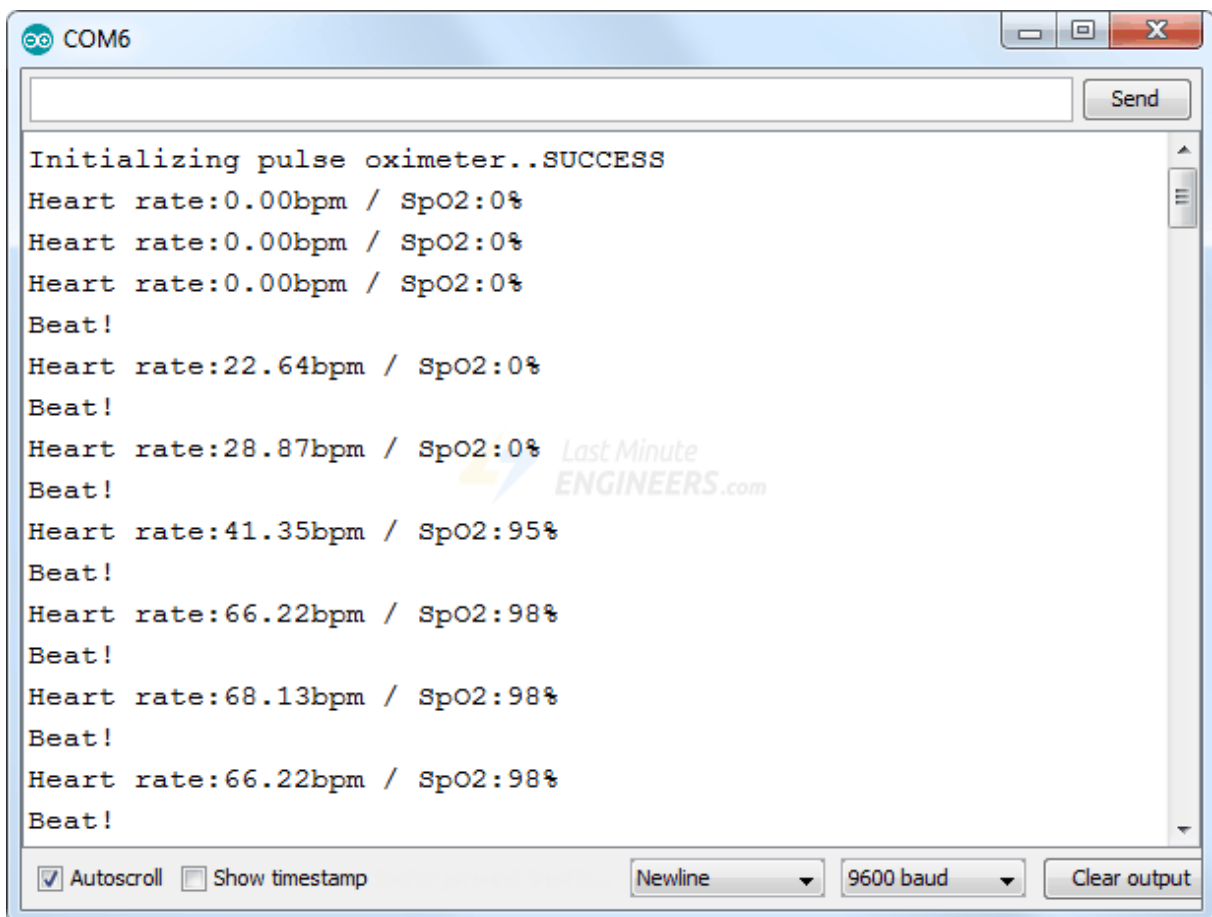
    tsLastReport = millis();

}

}

```

Après avoir téléchargé le croquis, maintenez votre doigt sur le capteur aussi stable que possible et attendez quelques secondes que les mesures aient un sens. Vous verrez un résultat comme celui-ci.



Explication du code :

Commençons par le haut de l'exemple. Deux bibliothèques, à savoir MAX30100_PulseOximeter.h et Wire.h, sont incluses, une constante appelée REPORTING_PERIOD_MS est définie pour retarder les mesures, PulseOximeter un objet est créé et une variable appelée tsLastReport est créée pour stocker l'heure à laquelle le dernier battement s'est produit.

```
#include <Wire.h>
```

```
#include "MAX30100_PulseOximeter.h"
```

```
#define REPORTING_PERIOD_MS 1000
```

```
PulseOximeter pox;
```

```
uint32_t tsLastReport = 0;
```

Ensuite, une routine de rappel est définie et exécutée lorsqu'une impulsion est détectée. Vous pouvez y écrire du code intéressant, par exemple pour regarder la LED intégrée clignoter au rythme de votre cœur !

```
void onBeatDetected() {  
    Serial.println("Beat!");  
}
```

Voyons maintenant la configuration. Il y a trois fonctions à souligner. Tout d'abord, l' `pox.begin()` appel de fonction garantit que nous pouvons communiquer avec le capteur. `begin()` La fonction renvoie 1 en cas de succès (ou 0 en cas d'échec). Elle configure également le MAX30100 pour commencer à collecter des données.

```
if (!pox.begin()) {  
    Serial.println("FAILED");  
    for(;;);  
} else {  
    Serial.println("SUCCESS");  
}
```

Deuxièmement, et tout aussi important, la `setIRLedCurrent()` fonction définit le courant à travers la LED IR. La ligne suivante définit le courant de 7,6 mA à travers la LED IR.

```
pox.setIRLedCurrent(MAX30100_LED_CURR_7_6MA);
```

Par défaut, la bibliothèque définit le courant de la LED IR sur 50 mA, ce qui peut parfois entraîner des problèmes tels que la LED qui ne s'allume pas ou l'affichage du message `Initializing pulse oxymeter.. FAILED` (Échec de l'initialisation de l'oxymètre de pouls) sur le moniteur série. Dans ce cas, définissez le courant de la LED sur une valeur inférieure. Voici les autres valeurs possibles que vous pouvez utiliser sur la `setIRLedCurrent()` fonction.

- `MAX30100_LED_COURANT_0MA`
- `MAX30100_LED_COURANT_4_4MA`
- `MAX30100_LED_COURANT_7_6MA`

- MAX30100_LED_COURANT_11MA
- MAX30100_LED_COURANT_14_2MA
- MAX30100_LED_COURANT_17_4MA
- MAX30100_LED_COURANT_20_8MA
- MAX30100_LED_COURANT_24MA
- MAX30100_LED_COURANT_27_1MA
- MAX30100_LED_COURANT_30_6MA
- MAX30100_LED_COURANT_33_8MA
- MAX30100_LED_COURANT_37MA
- MAX30100_LED_COURANT_40_2MA
- MAX30100_LED_COURANT_43_6MA
- MAX30100_LED_COURANT_46_8MA
- MAX30100_LED_COURANT_50MA

N'oubliez pas ! Plus le courant est élevé, plus la LED est lumineuse et plus elle pénètre profondément dans votre peau. Vous pouvez donc jouer avec elle pendant que vous la découvrez.

Enregistre enfin `setOnBeatDetectedCallback()` la fonction de rappel.

```
pox.setOnBeatDetectedCallback(onBeatDetected);
```

Dans la boucle principale, les données biométriques sont collectées à partir du capteur MAX30100 avec la `update()` fonction. En fait, il récupère les données du capteur FIFO.

```
pox.update();
```

Nous imprimons ensuite les données biométriques sur le moniteur série une fois par seconde (`REPORTING_PERIOD_MS` est défini sur 1000 ; modifiez-le selon vos besoins). Pour cela, nous utilisons `millis()` à la place de `delay()`. Étant donné que `delay()` l'ensemble du code est mis en pause, nous ne pouvons pas obtenir la mesure mise à jour.

```
if (millis() - tsLastReport > REPORTING_PERIOD_MS) {
```

```
    Serial.print("Heart rate:");
```

```
    Serial.print(pox.getHeartRate());
```

```
    Serial.print("bpm / SpO2:");
```

```
    Serial.print(pox.getSpO2());
```

```
    Serial.println("%");
```

```
    tsLastReport = millis();
```

```
}
```

Vous avez du mal à voir un battement de cœur ?

Si vous avez du mal à voir un battement de cœur, voici ce qu'il faut faire.

1. Si vous tenez le capteur trop fort, vous évacuerez tout le sang de vos doigts et il n'y aura aucun signe ! Si vous le tenez trop légèrement, vous serez exposé au bruit du mouvement et de la lumière ambiante. Une pression sur le point de transpiration (ni trop forte, ni trop douce) sur le capteur de pouls donnera un bon signal propre.
2. Une pression variable peut entraîner une circulation sanguine différente dans votre doigt, ce qui peut fausser les mesures du capteur. Essayez d'appliquer une pression constante en fixant le capteur à votre doigt à l'aide d'un élastique ou d'un autre dispositif de serrage.
3. Essayez le capteur sur différentes parties de votre corps qui ont du tissu capillaire (comme le lobe de l'oreille ou la lèvre inférieure).

Exemple 2 – Lecture du rouge et de l'IR

Le deuxième exemple génère les valeurs brutes (mesures IR et rouge) lues par le capteur. Chargez-le sur votre Arduino et ouvrez le terminal série pour voir les valeurs imprimées.

```
#include <Wire.h>
```

```
#include "MAX30100_PulseOximeter.h"
```

```
// Create a MAX30100 object
```

```
MAX30100 sensor;
```

```
void setup() {
```

```
    Serial.begin(115200);
```

```
    Serial.print("Initializing MAX30100..");
```

```
    // Initialize sensor
```

```
    if (!sensor.begin()) {
```

```
        Serial.println("FAILED");
```

```
        for(;;);
```

```
    } else {
```

```
        Serial.println("SUCCESS");
```

```
    }
```

```

    // Configure sensor
    configureMax30100();
}

void loop() {
    uint16_t ir, red;

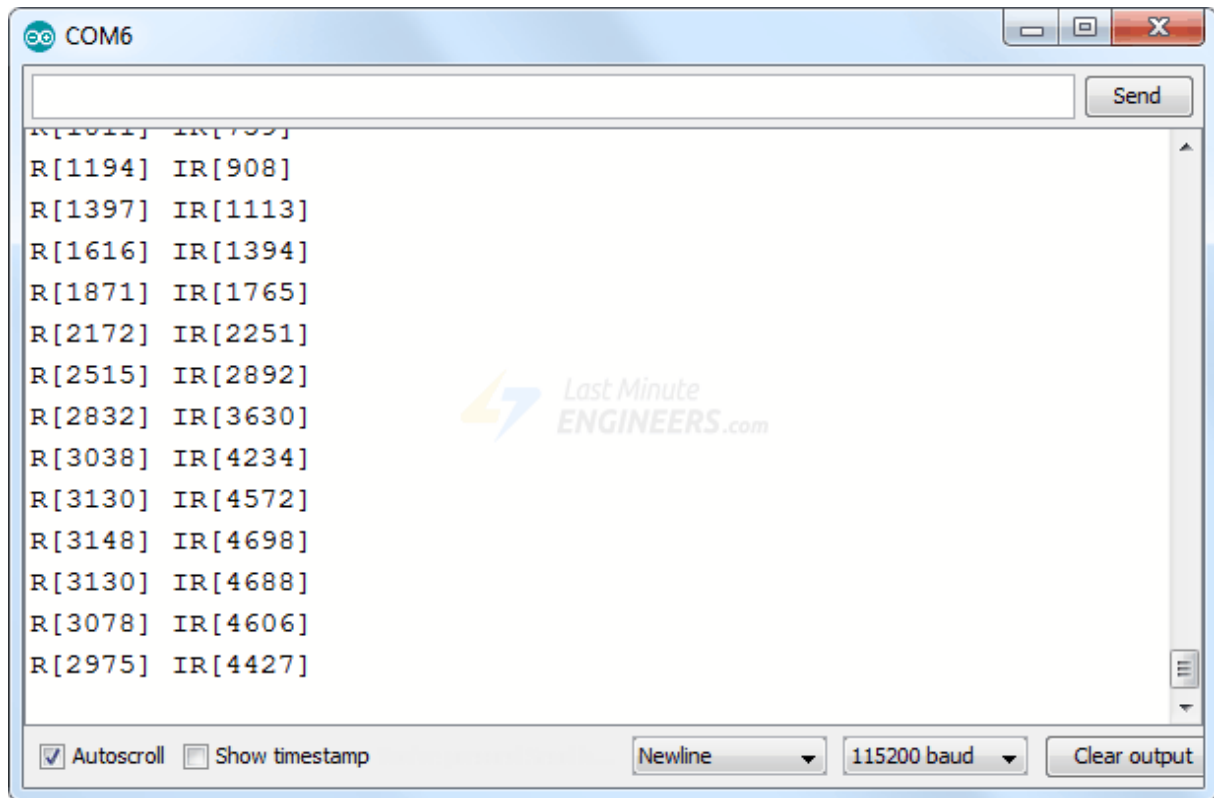
    sensor.update();

    while (sensor.getRawValues(&ir, &red)) {
        Serial.print("R[");
        Serial.print(red);
        Serial.print("] IR[");
        Serial.print(ir);
        Serial.println("]");
    }
}

void configureMax30100() {
    sensor.setMode(MAX30100_MODE_SPO2_HR);
    sensor.setLedsCurrent(MAX30100_LED_CURR_50MA, MAX30100_LED_CURR_27_1MA);
    sensor.setLedsPulseWidth(MAX30100_SPC_PW_1600US_16BITS);
    sensor.setSamplingRate(MAX30100_SAMPRATE_100HZ);
    sensor.setHighresModeEnabled(true);
}

```

Avec le capteur pointé vers le haut, passez votre main dessus. Vous devriez voir un changement dans les valeurs lorsque votre main réfléchit différentes quantités de lumière.

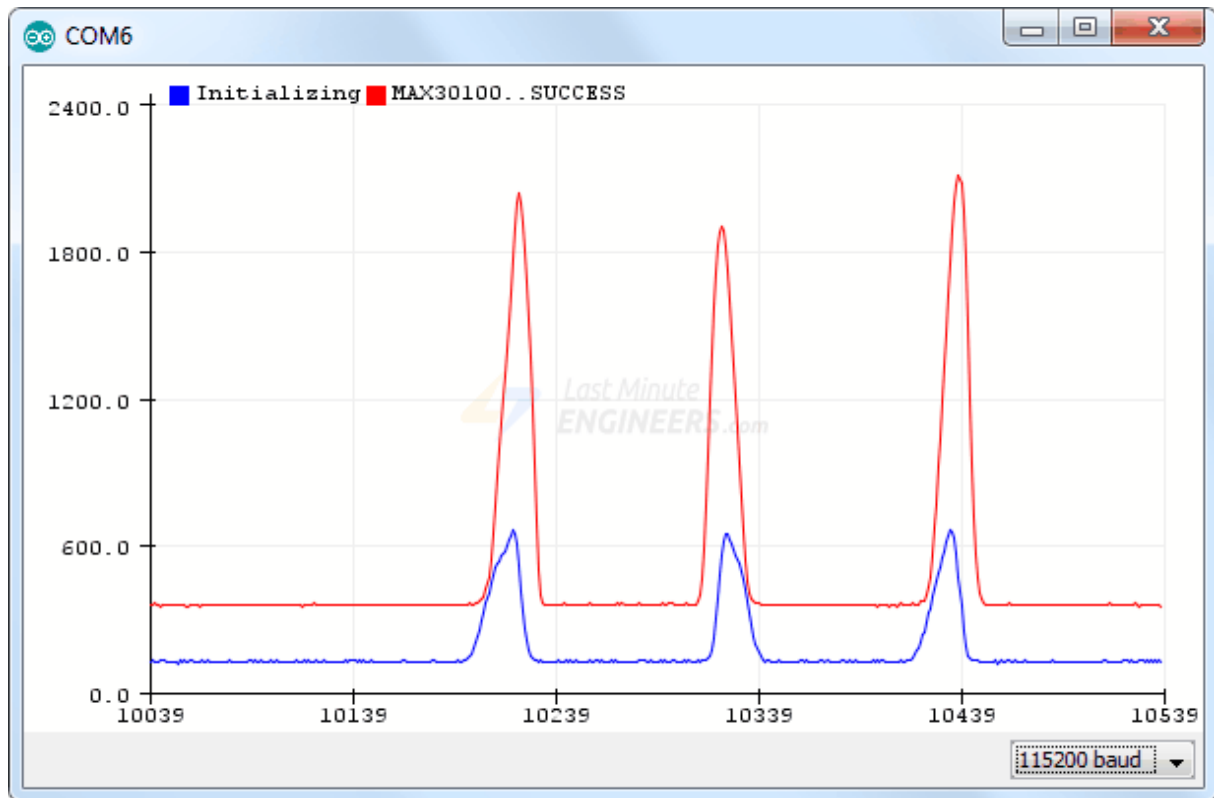


Les données série peuvent être difficiles à visualiser si vous ne regardez que les valeurs. Si vous utilisez l'IDE Arduino v1.6.6+, il existe une option permettant d'afficher les données sur un graphique à l'aide du traceur série Arduino .

Pour commencer, remplacez le while() dans le code ci-dessus par cet extrait de code :

```
while (sensor.getRawValues(&ir, &red)) {  
    Serial.print(red);  
    Serial.print(", ");  
    Serial.println(ir);  
}
```

Dans l'IDE Arduino, choisissez Outils > Traceur série. Vous devriez voir une onde semblable à l'image ci-dessous lorsque vous passez votre main sur le capteur.



Exemple 3 – Détection de présence

Notre prochaine expérience montre comment utiliser le capteur MAX30100 comme capteur de proximité ou de réflectance à usage général et peut servir de base à des expériences et projets plus pratiques.

Cet exemple fonctionne en prenant une poignée de mesures lors de la configuration et en faisant la moyenne de celles-ci. Il utilise ensuite cette moyenne comme base de référence. Si le capteur détecte un changement significatif par rapport à la moyenne, le message « Il y a quelque chose ! » s'affiche. Allez-y et essayez le croquis.

```
#include <Wire.h>
```

```
#include "MAX30100_PulseOximeter.h"
```

```
#define REPORTING_PERIOD_MS 1000
```

```
int lastOccurrence = LOW;
```

```
// Create a MAX30100 object
```

```
MAX30100 sensor;
```

```
uint16_t ir, red;
```

```
uint16_t avg_ir = 0, avg_red = 0;

void setup() {
    Serial.begin(115200);

    Serial.print("Initializing MAX30100..");

    // Initialize sensor
    if (!sensor.begin()) {
        Serial.println("FAILED");
        for(;;);
    } else {
        Serial.println("SUCCESS");
    }

    configureMax30100();

    takeSampleReadings();
}

void loop() {
    sensor.update();

    while (sensor.getRawValues(&ir, &red)) {
        if (ir > 10*avg_ir && red > 10*avg_red) {
            if (lastOccurrence == LOW) {
                Serial.println("Something is there!");
                lastOccurrence = HIGH;
            }
        }
    }
    else{
```

```

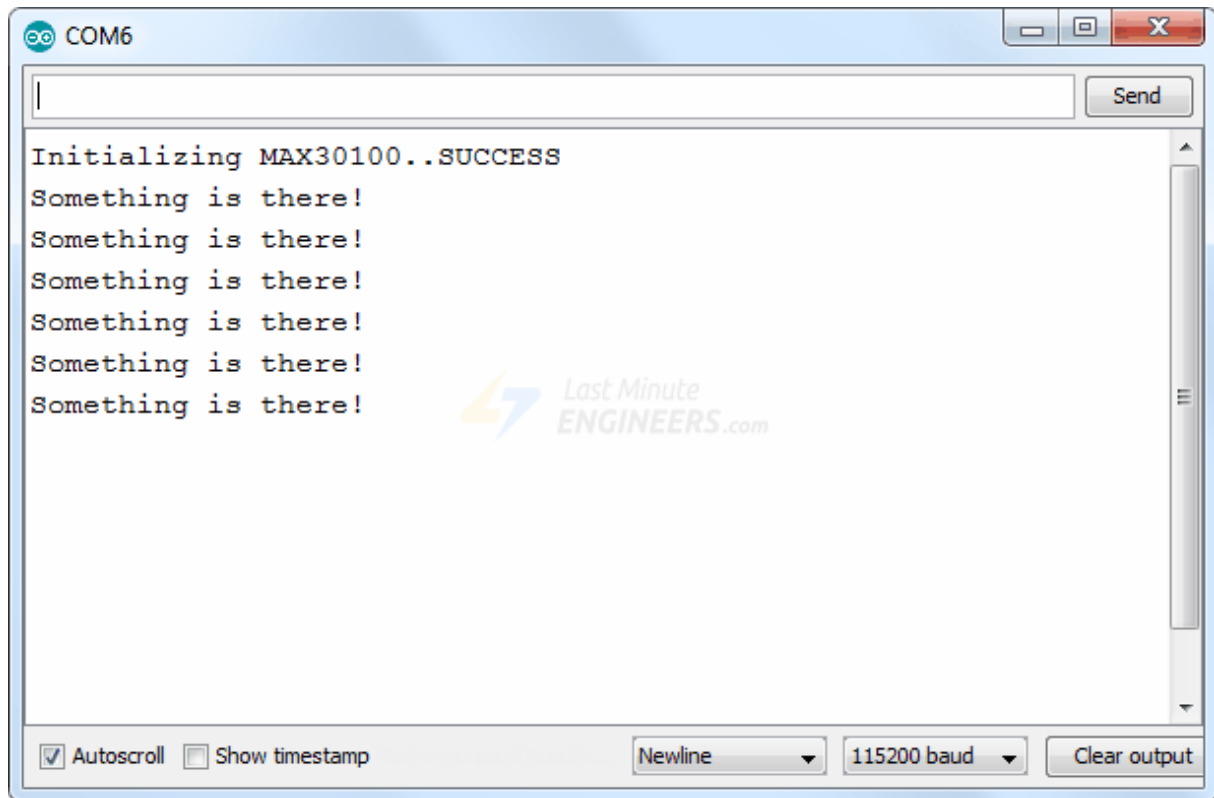
        lastOccurrence = LOW;
    }
}
}

void configureMax30100() {
    sensor.setMode(MAX30100_MODE_SPO2_HR);
    sensor.setLedsCurrent(MAX30100_LED_CURR_50MA, MAX30100_LED_CURR_27_1MA);
    sensor.setLedsPulseWidth(MAX30100_SPC_PW_1600US_16BITS);
    sensor.setSamplingRate(MAX30100_SAMPRATE_100HZ);
    sensor.setHighresModeEnabled(true);
}

void takeSampleReadings() {
    delay(50);
    for (int i = 0; i <= 9; i++) {
        sensor.update();
        sensor.getRawValues(&ir, &red);
        avg_ir += ir;
        avg_red += red;
        delay(50);
    }
    avg_ir /= 10;
    avg_red /= 10;
}

```

Passez à nouveau votre main sur le capteur et recherchez le message « Il y a quelque chose ! » imprimé sur le terminal série. Essayez de tester la portée à laquelle le capteur peut détecter quelque chose.



Notez que le MAX30100 est capable de lire jusqu'à 16 bits ou des valeurs jusqu'à 65 536. Un mouvement extrêmement petit peut être détecté !

Exemple 4 – Lecture de la température

Notre dernier exemple affiche les valeurs mesurées par le capteur de température intégré en degrés Celsius et Fahrenheit. Bien que la mesure de température doive être utilisée pour étalonner les mesures de FC et de SpO2, elle peut être utile si vous avez besoin d'un capteur de température sensible et à réponse rapide.

```
#include <Wire.h>
```

```
#include "MAX30100_PulseOximeter.h"
```

```
#define REPORTING_PERIOD_MS 1000
```

```
// Create a MAX30100 object
```

```
MAX30100 sensor;
```

```
// Time when the last reading was taken
```

```
uint32_t tsLastReading = 0;
```

```
void setup() {
```

```
Serial.begin(115200);

Serial.print("Initializing MAX30100..");

// Initialize sensor
if (!sensor.begin()) {
    Serial.println("FAILED");
    for(;;);
} else {
    Serial.println("SUCCESS");
}

// Configure sensor
configureMax30100();
}

void loop() {
    sensor.update();

    if (millis() - tsLastReading > REPORTING_PERIOD_MS) {
        sensor.startTemperatureSampling();
        if (sensor.isTemperatureReady()) {
            float temp = sensor.retrieveTemperature();
            Serial.print("Temperature = ");
            Serial.print(temp);
            Serial.print("*C | ");
            Serial.print((temp * 9.0) / 5.0 + 32.0); //print the temperature in Fahrenheit
            Serial.println("*F");
        }
        tsLastReading = millis();
    }
}
```

```
}
```

```
void configureMax30100(){  
    sensor.setMode(MAX30100_MODE_SPO2_HR);  
    sensor.setLedsCurrent(MAX30100_LED_CURR_50MA, MAX30100_LED_CURR_27_1MA);  
    sensor.setLedsPulseWidth(MAX30100_SPC_PW_1600US_16BITS);  
    sensor.setSamplingRate(MAX30100_SAMPRATE_100HZ);  
    sensor.setHighresModeEnabled(true);  
}
```

Essayez maintenant de chauffer le capteur avec votre doigt ou de souffler légèrement dessus. Vous devriez voir quelque chose comme le résultat ci-dessous.

