Rejoignez GitHub aujourd'hui

GitHub regroupe plus de 36 millions de développeurs travaillant ensemble pour héberger et réviser du code, gérer des projets et créer des logiciels.

S'inscrire

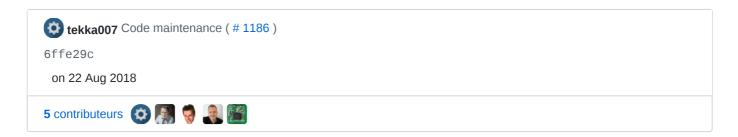
Rejeter

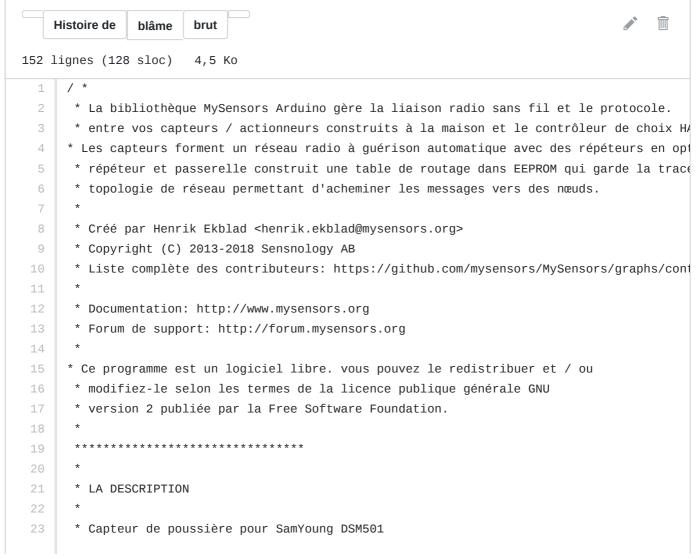
Branche: **maître** ▼

Trouver un fichier

Copier le chemin

MySensors / examples / DustSensorDSM / DustSensorDSM.ino





```
24
      * connectez le capteur comme suit:
25
      * Broche 2 du capteur de poussière PM1 -> Digital 3 (PMW)
      * Broche 3 du capteur de poussière -> + 5V
26
27
      * Broche 4 du capteur de poussière PM2.5 -> Digital 6 (PWM)
      * Broche 5 du capteur de poussière -> Terre
28
      * Fiche technique: http://www.samyoungsnc.com/products/3-1%20Specification%20DSM501.pd
29
     * Contributeur: epierre
     * * /
31
     // Activer les impressions de débogage
34
    # définir MY_DEBUG
35
36
    // Activer et sélectionner le type de radio attaché
37
    # define MY_RADIO_RF24
    // #define MY_RADIO_NRF5_ESB
38
39
    // #define MY_RADIO_RFM69
40
    // #define MY_RADIO_RFM95
41
    # include < MySensors.h >
42
43
    # define CHILD_ID_DUST_PM10
44
    # define CHILD_ID_DUST_PM25
45
46
    # define DUST_SENSOR_DIGITAL_PIN_PM10
    # define DUST_SENSOR_DIGITAL_PIN_PM25
47
48
    uint32_t SLEEP_TIME = 30 * 1000 ; // Temps de sommeil entre les lectures (en millisecor
49
    // VARIABLES
50
    int val = 0;
                            // variable pour stocker la valeur provenant du capteur
51
52
    float valDUSTPM25 = 0,0 ;
    float lastDUSTPM25 = 0.0 ;
53
54
    float valDUSTPM10 = 0,0 ;
    float lastDUSTPM10 = 0.0 ;
55
56
    uint32_t duration;
57
    uint32_t starttime;
58
    uint32_t endtime;
    uint32_t sampletime_ms = 30000 ;
60
    uint32_t lowpulseoccupancy = 0 ;
61
    rapport de flottement = 0 ;
     longue concentrationPM25 = 0 ;
63
    longue concentrationPM10 = 0 ;
64
     MyMessage dustMsgPM10 (CHILD_ID_DUST_PM10, V_LEVEL);
     MyMessage msgPM10 (CHILD_ID_DUST_PM10, V_UNIT_PREFIX);
66
     MyMessage dustMsgPM25 (CHILD_ID_DUST_PM25, V_LEVEL);
67
     MyMessage msgPM25 (CHILD_ID_DUST_PM25, V_UNIT_PREFIX);
68
69
70
    void setup ()
71
72
       pinMode (DUST_SENSOR_DIGITAL_PIN_PM10, INPUT);
73
       pinMode (DUST_SENSOR_DIGITAL_PIN_PM25, INPUT);
74
     }
```

```
présentation vide ()
 76
 77
      {
        // Envoyer les informations de version de l'esquisse à la passerelle et au contrôleur
 78
        sendSketchInfo ( " Capteur de poussière DSM501 " , " 1.4 " );
 79
 80
        // Enregistre tous les capteurs sur la passerelle (ils seront créés en tant que péri
 81
        present (CHILD_ID_DUST_PM10, S_DUST);
 82
        send (msgPM10. set ( " ppm " ));
 83
        present (CHILD_ID_DUST_PM25, S_DUST);
        send (msgPM25. set ( " ppm " ));
 86
      }
 87
       boucle vide ()
 88
 89
      {
        // obtenir une densité de particules PM 2,5 supérieure à 2,5 µm.
 91
        concentrationPM25 = ( long ) getPM (DUST_SENSOR_DIGITAL_PIN_PM25);
        En série. imprimer ( " PM25: " );
        En série. impression (concentration MP25 );
        En série. print ( " \ n " );
        if ((concentrationPM25! = lastDUSTPM25) && (concentrationPM25> 0 )) {
 97
          send (dustMsgPM25. set (( int32_t ) ceil (concentrationPM25))));
          lastDUSTPM25 = ceil (concentrationPM25);
        }
100
101
        // PM 1.0 - densité de particules supérieure à 1 \mu m.
        concentrationPM10 = getPM (DUST_SENSOR_DIGITAL_PIN_PM10);
103
        En série. imprimer ( " PM10: " );
104
        En série. impression (concentrationPM10);
105
        En série. print ( " \ n " );
106
        // ppmv = mg / m3 * (0.08205 * Tmp) / Molecular_mass
        // 0,08205 = Constante universelle des gaz en atm \cdot m3 / (kmol \cdot K)
108
109
        int temp = 20 ; // température externe, si vous pouvez remplacer ceci par un DHT11 ou
        longue ppmv = (concentrationPM10 * 0,0283168 / 100 / 1000 ) * ( 0,08205 temp *) / 0,0
111
        if (( ceil (concentrationPM10)! = lastDUSTPM10) && (( long ) concentrationPM10> 0 ))
112
          send (dustMsgPM10. set (( int32_t ) ppmv));
          lastDUSTPM10 = ceil (concentrationPM10);
114
        }
115
        // dormir pour économiser à la radio
117
        dormir (SLEEP_TIME);
118
119
120
      }
121
122
       getPM long ( int DUST_SENSOR_DIGITAL_PIN)
123
124
125
126
        heure de début = millis ();
```

```
128
        tandis que (1) {
129
130
          duration = pulseIn (DUST_SENSOR_DIGITAL_PIN, LOW);
131
          faible occupation + = durée;
132
          heure de fin = millis ();
133
          if ((endtime-starttime)> sampletime_ms) {
134
            ratio = (lowpulseoccupancy-endtime + starttime) / (sampletime_ms * 10.0 ); // Po
135
136
            concentration longue = 1,1 * pow (ratio, 3 ) - 3,8 * pow (ratio, 2 ) + 520 * ration
            // Serial.print ("lowpulseoccupancy:");
137
            // Serial.print (lowpulseoccupancy);
138
            // Serial.print ("\ n");
139
            // Serial.print ("ratio:");
140
            // Serial.print (ratio);
141
            // Serial.print ("\ n");
142
            // Serial.print ("DSM501A:");
143
            // Serial.println (concentration);
144
            // Serial.print ("\ n");
145
146
            faible occupation du vide = 0;
147
            retour (concentration);
148
149
          }
150
        }
      }
151
```