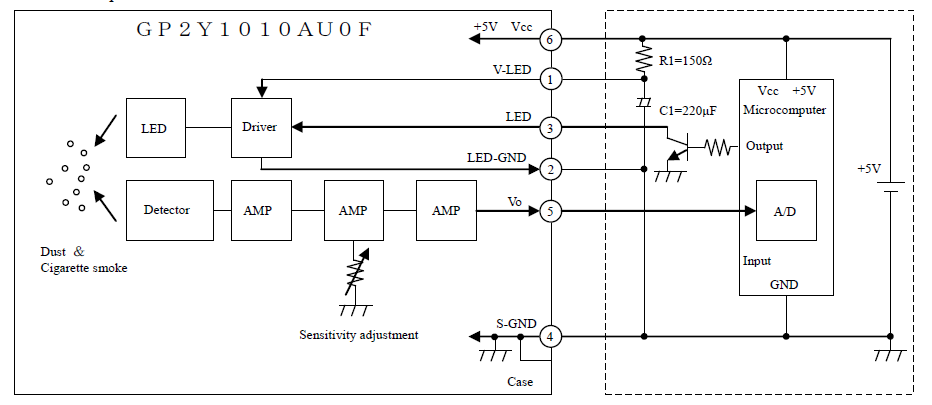
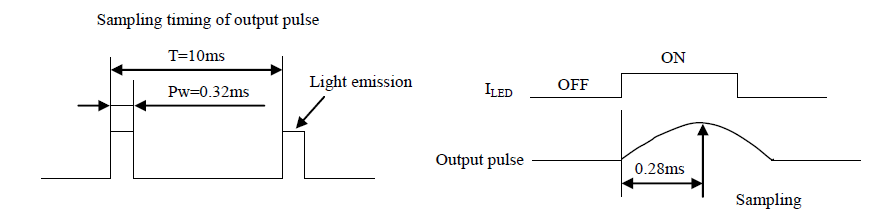
Mise en pratique du capteur de poussiere GP2Y1010AU0F

La documentation du **GP2Y1010AU0F** preconise **d’utiliser une capacite de 220uF et une resistance de 150 Ohms pour “driver” le pulse de la Led.**





**Il faut faire attention d’attendre le temps necessaire pour la recharge du condensateur entre 2 mesures, sinon la mesure sera faussé.**

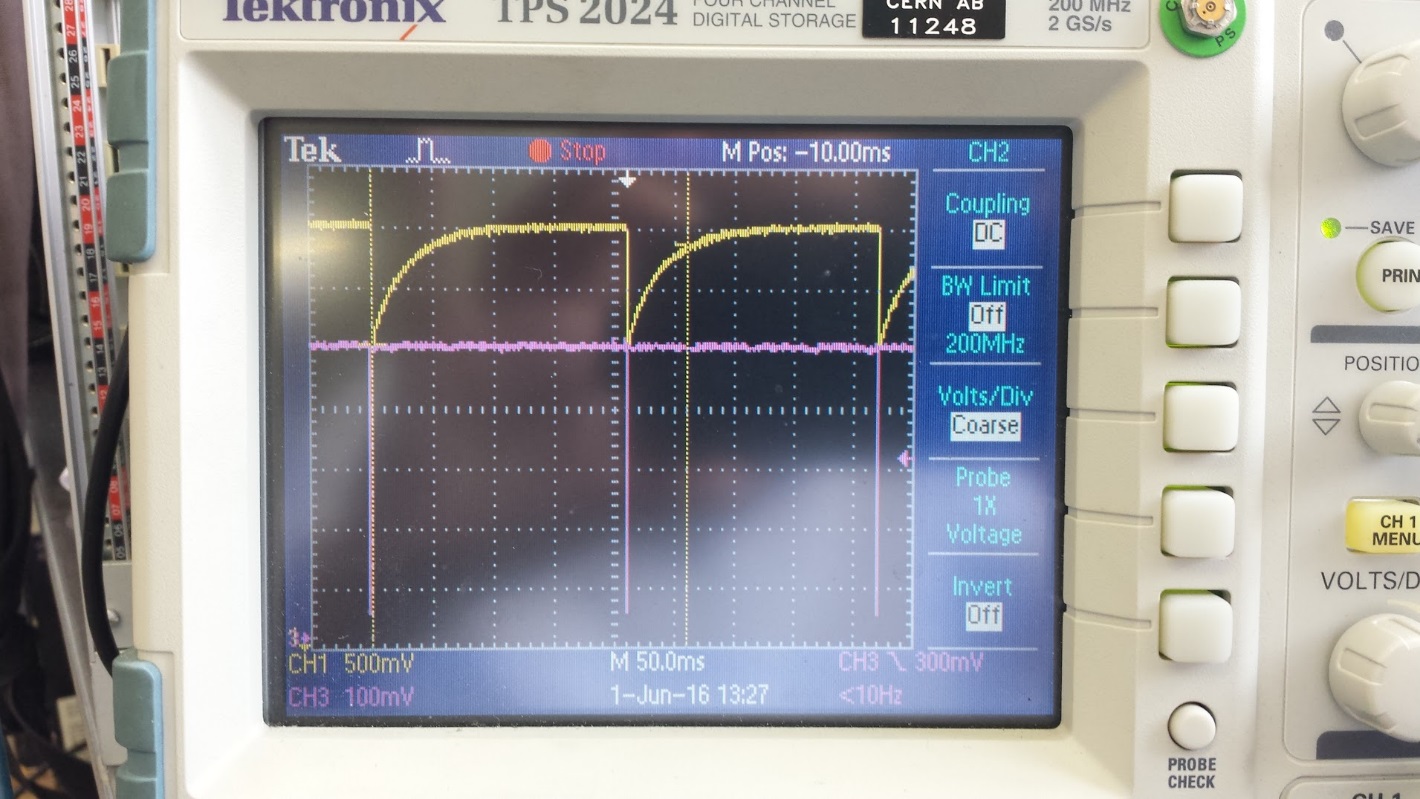
**Tau = 5\* R\*C**

**Tau = 5\*150\*220.e-6**

**Tau = 5\*0.033**

**Tau = 165 ms**

**J’ai pu constater q’une seule mesure permet d’obtenir une mesure fiable et si nous attendons le temps de recharge necessaire au condensateur la mesure répétée est stable.**

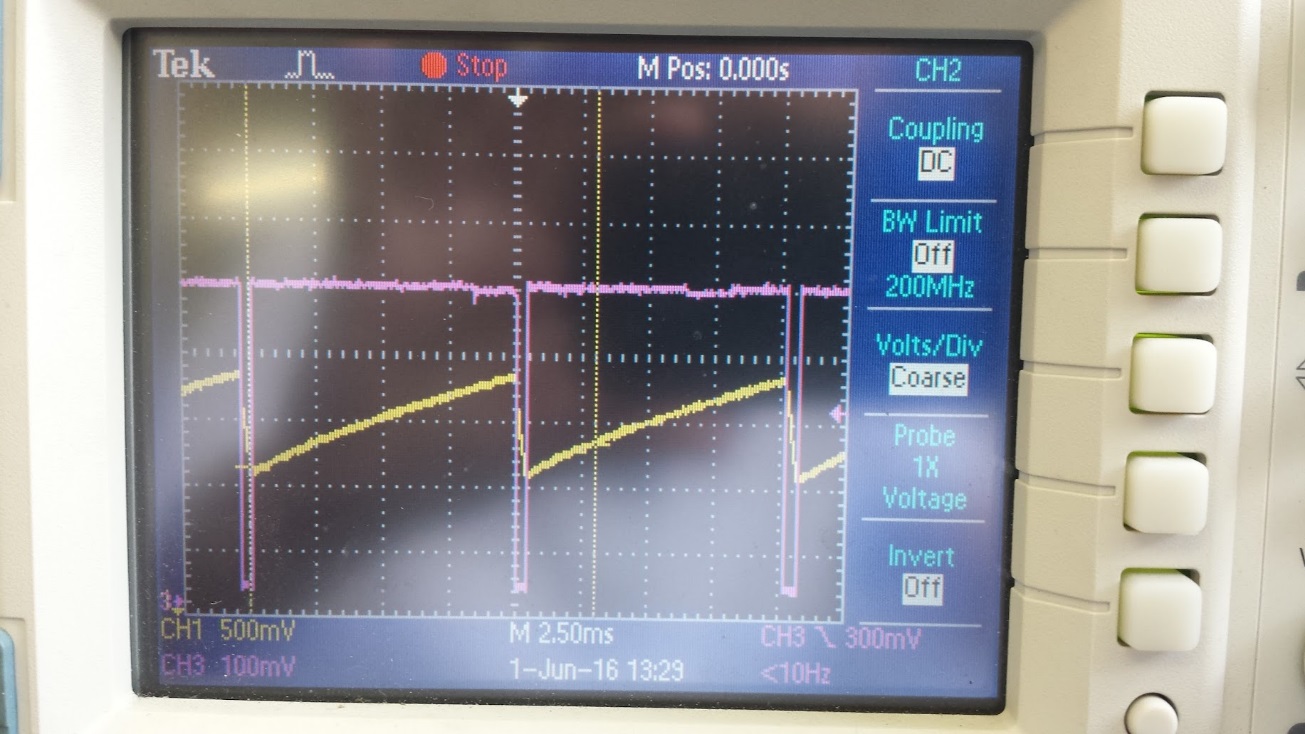
****

**Fig 1 : Charge du condensateur entre pulses (T = 210 ms).**

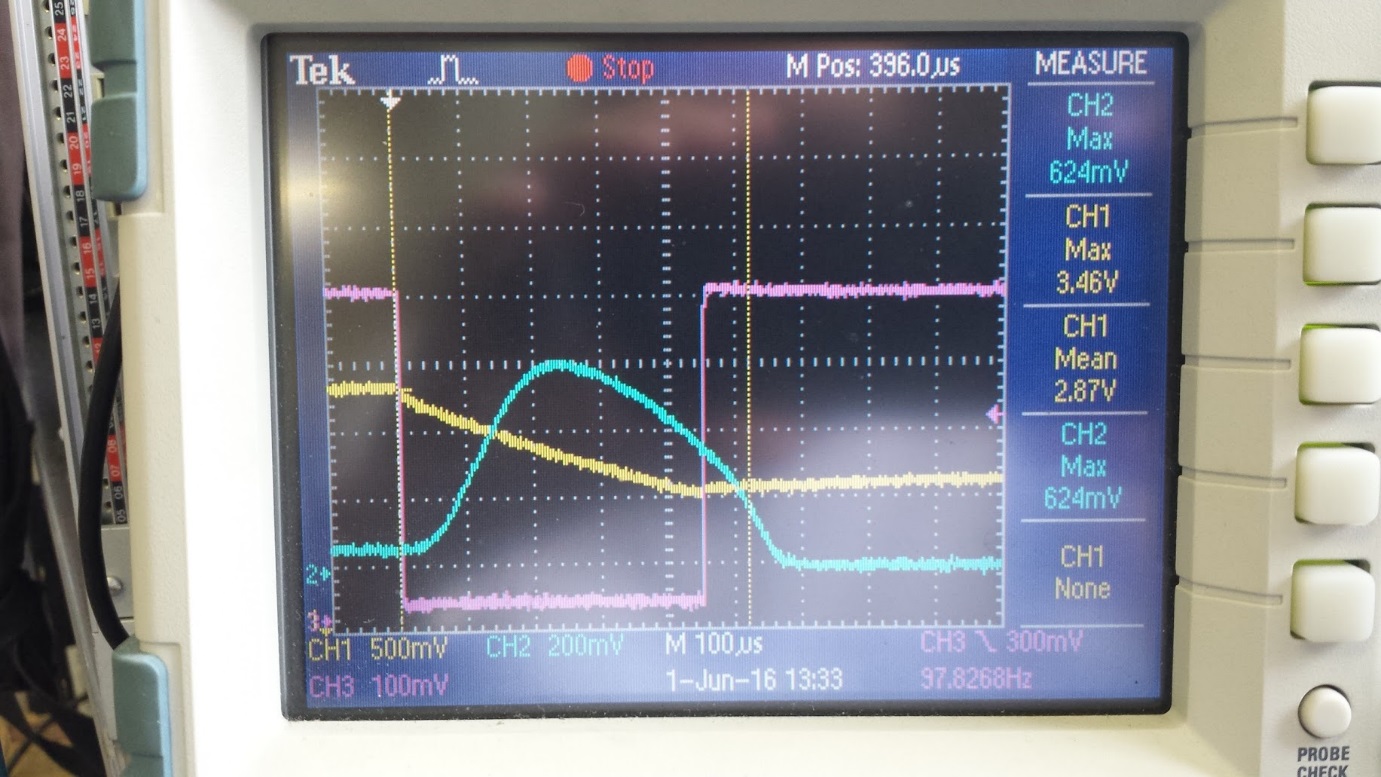
****

**Fig 2 : Mesure du signal de sortie Vo en bleu (max = 1.05v) , en jaune mesure de la tension du condensateur VLed, en violet pulse LED ON pendant T = 0.32 ms.**

**Exemple Pulse toute les 10ms :**

****

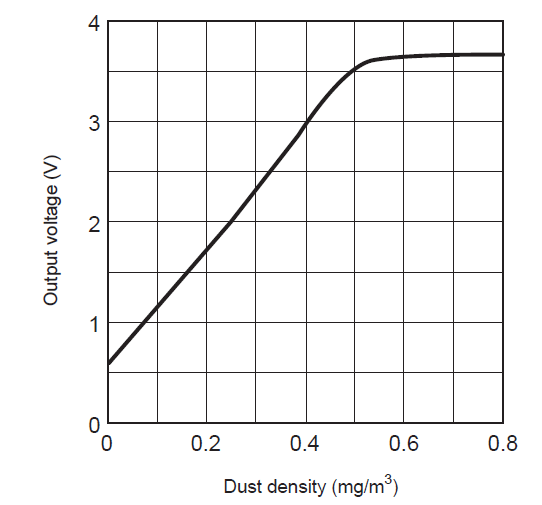
**Fig 3 : Toutes les 10ms un pulse LED, le condensateur n’a pas le temps de se recharger.**

****

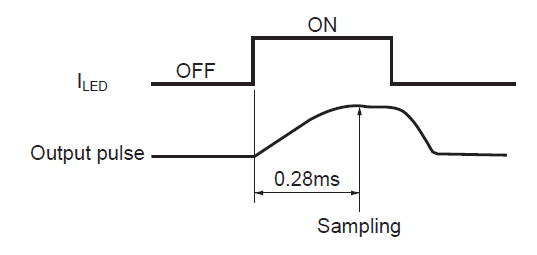
**Fig 4 : Mesure du signal de sortie Vo en bleu (max = 0.62v) , en jaune mesure de tension du condensateur VLed, en violet pulse LED.**

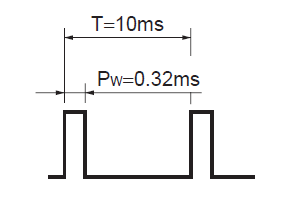
**Le condensateur n’a pas le temps de se recharger completement, la mesure de tension de sortie est diminuée.**

**La consequence est que si la tension de sortie est plus basse nous pouvons etre en dessous (ou a la limite) de la valeur minimale sur l’abaque .**



**Fig 5 : abaque du fabricant Vo du module et densitee (mg/m3)**





**#define SAMPLING\_TIME 280 microsecondes (soit 0.28ms)**

**#define DELTA\_TIME 40 microsecondes (soit ILed on pendant 0.28+0.04 = 0.32 ms)**

**#define SLEEP\_TIME 9680 microsecondes ( 0.32ms + 9.68ms = 10ms)**

**#define PAUSE 200 millisecondes (attendre la recharge de la capacitee )**

**#define NB\_MEAS 10 (faire plusieurs mesures)**

**Calcul de la pente dans la region lineaire :**

**(3.5-0.6)(0.5-0) = 5.8**

**courbe de la droite selon le constructeur y = 5.8 x + 0.6**

**avec x = en volts et y en mg/m3 soit x = (y - 0.6) / 5.8**

**Fichier SharpDust.cpp**

**#include "Arduino.h"**

**#include "SharpDust.h"**

**#include "pins\_arduino.h"**

**SharpDustClass::SharpDustClass()**

**{**

**}**

**void SharpDustClass::begin(int led, int mea , float rang )**

**{**

**ledPin = led;**

**measurePin = mea;**

**range = rang ;**

**pinMode(measurePin, INPUT);**

**pinMode(ledPin, OUTPUT);**

**digitalWrite(ledPin, HIGH);**

**}**

**float SharpDustClass::measure(void)**

**{**

**float calcVoltage = 0;**

**float average = 0;**

**unsigned long sum = 0 ;**

**unsigned int voMeasured[NB\_MEAS] ;**

**for(int i=0; i< NB\_MEAS; i++) {**

**digitalWrite(ledPin, LOW);**

**delayMicroseconds(SAMPLING\_TIME);**

**voMeasured[i] = analogRead(measurePin) ;**

**sum += voMeasured[i] ;**

**Serial.println(voMeasured[i]);**

**Serial.println(sum) ;**

**delayMicroseconds(DELTA\_TIME);**

**digitalWrite(ledPin, HIGH);**

**delayMicroseconds(SLEEP\_TIME);**

**delay (PAUSE) ;**

**}**

**average = sum / NB\_MEAS ;**

**Serial.print(" Average : ") ;**

**Serial.println(average) ;**

**calcVoltage = fmap(average, 0, 1023, 0.0, range );**

**// courbe de la droite selon le constructeur y = 5.8 x + 0.6**

**// avec x = en volts et y en mg/m3 soit x = (y - 0.6) / 5.8**

**if (calcVoltage < 0.6 ) calcVoltage = 0.6 ;**

**Serial.print(" calcVoltage : ") ;**

**Serial.println(calcVoltage) ;**

**calcVoltage = (calcVoltage - 0.6) / 5.8 ;**

**//return 0.17 \* calcVoltage - 0.1;**

**return calcVoltage\*1000 ;**

**}**

**float SharpDustClass::fmap(float x, float in\_min, float in\_max, float out\_min, float out\_max)**

**{**

**return (x - in\_min) \* (out\_max - out\_min) / (in\_max - in\_min) + out\_min;**

**}**

**SharpDustClass SharpDust;**

**Fichier : SharpDust.h**

**#ifndef \_\_SHARP\_DUST\_H**

**#define \_\_SHARP\_DUST\_H**

**#define SAMPLING\_TIME 280**

**#define DELTA\_TIME 40**

**#define SLEEP\_TIME 9680**

**#define PAUSE 200**

**#define NB\_MEAS 10**

**class SharpDustClass**

**{**

**private:**

**int ledPin; //Digital pin for LED power**

**int measurePin; //Analog pin for measurement**

**float range ; // external voltage ADC**

**public:**

**SharpDustClass();**

**void begin(int led, int mea , float rang );**

**float measure(void);**

**float fmap(float x, float in\_min, float in\_max, float out\_min, float out\_max);**

**};**

**extern SharpDustClass SharpDust;**

**#endif //\_\_SHARP\_DUST\_H**