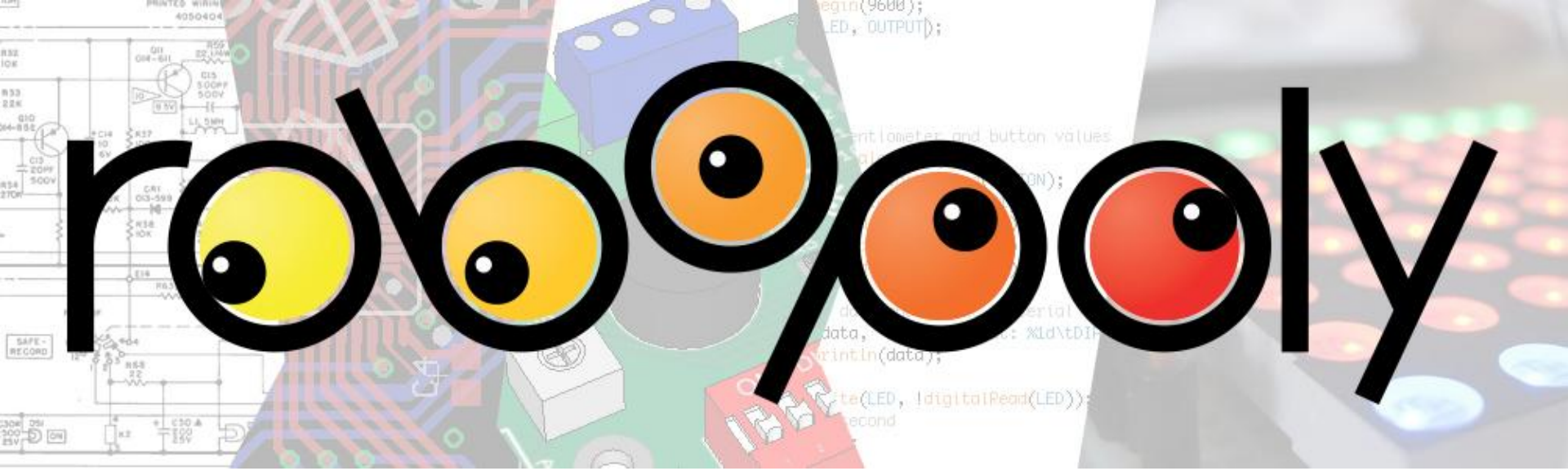


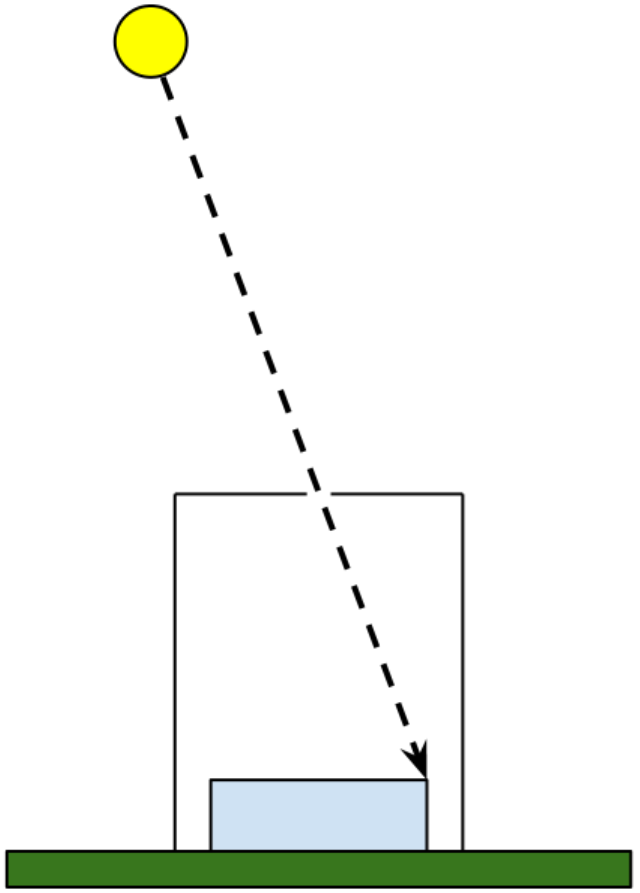
CAMERA LINEAIRE

I2C

Ca sert à quoi?

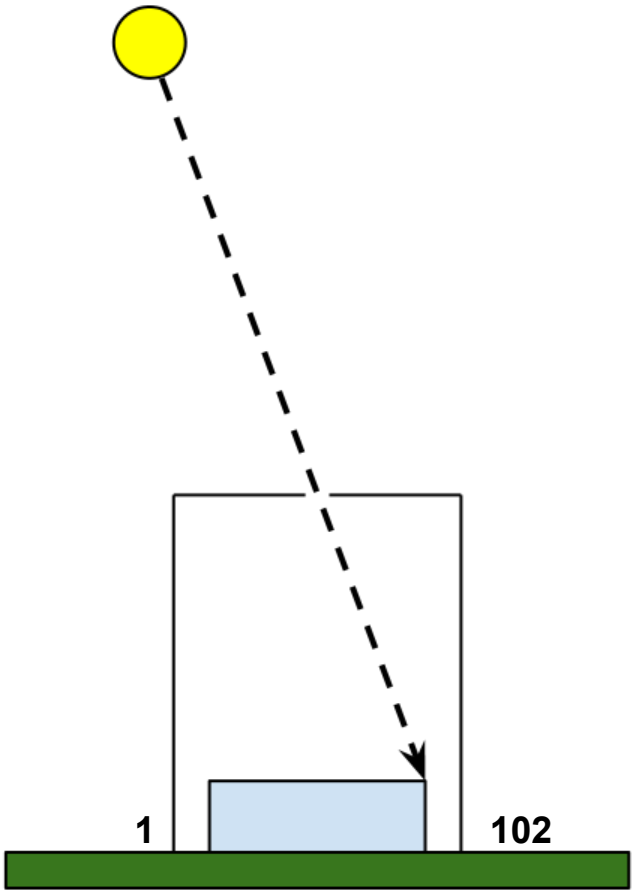
- Détecter la **position d'une source lumineuse**

- Potentiellement
  - **Lecture de code barre**
  - **Flux optique**
  - **Detection de ligne**
- Nécessite du travail
  - Trouver l'optique adaptée
  - Utilisation d'une illumination externe?



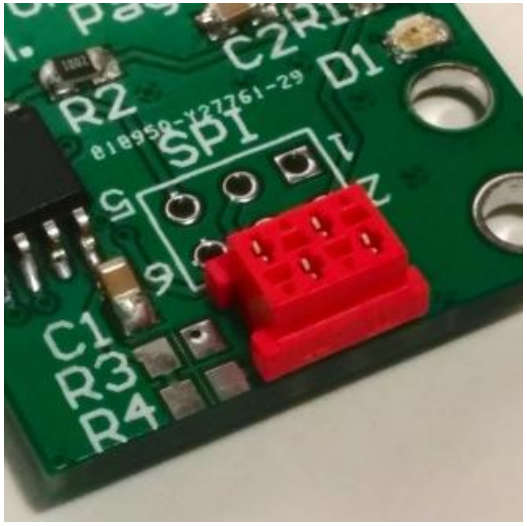
Principe de fonctionnement

- La lumière arrive par la fente sur les 102 recepteurs (pixels)
- A dependance du temps d'exposition et de l'intensité de la lumière, la valeur du recepteur monte de 0 à 255
- Attention à la saturation. Si le temps d' exposition est trop long et/ou la lumière est trop forte, tous les pixels vont à 255



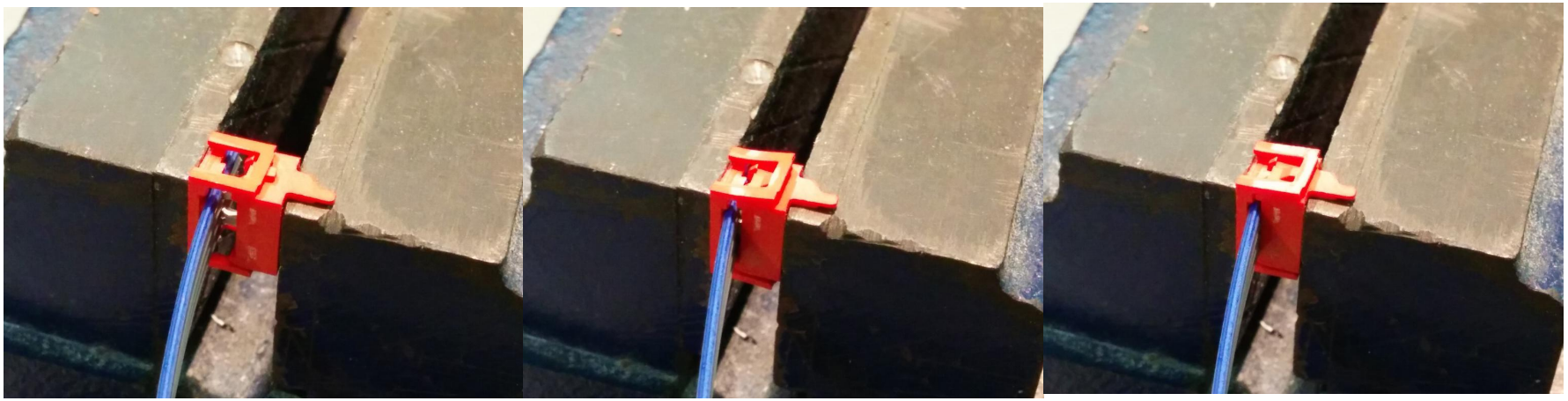
Connection

- Connecteur de type “microMatch”
- Nappe à 4 fils



Comment attacher le connecteur mâle à la nappe

- le fil bleu toujours du côté du detrompeur (ou l’envers)



Librairie

NE PAS OUBLIER DE TELECHARGER LE FIRMWARE

Initialisation

```
LinearCamera lc = LinearCamera(5);
```

Définir le temps d'exposition

```
void setExpTime(uint16_t);
```

Prendre une image et acquerir les pixels (pointeur sur le premier pixel)

```
lc.getPixels();
```

Récupérer le pic

```
lcPeak = lc.getPeak();
```

https://github.com/Robopoly/prismino-library/tree/master/libraries/LinearCamera

Lire les pixels

```
#include <prismino.h>
#include <LinearCamera.h>

// new instance of the camera, it works over i2c and the default address is "5"
LinearCamera lc = LinearCamera(5);

uint8_t *lcDataPtr;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  // must be called to fetch data from the linear camera
  lcDataPtr = lc.getPixels();

  // send all 102 pixels to serial
  for(uint8_t i = 0; i < LinearCamera::PIXELS; i++)
  {
    Serial.print(*(lcDataPtr + i));
  }
  Serial.print("\n");

  delay(500);
}
```

https://github.com/Robopoly/prismino-library/blob/master/libraries/LinearCamera/examples/GetPixels/GetPixels.ino

Trouver le pic

```
#include <prismino.h>
#include <LinearCamera.h>

// new instance of the camera, it works over i2c and the default address is "5"
LinearCamera lc = LinearCamera(5);

int8_t lcPeak = 0;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  // must be called to fetch data from the linear camera
  lc.getPixels();

  // this will process the local linear camera data and find the peak position between 0 and 101
  // if the value is 51 it means the peak is straight ahead
  lcPeak = lc.getPeak();
  Serial.println(lcPeak);

  delay(100);
}
```

https://github.com/Robopoly/prismino-library/blob/master/libraries/LinearCamera/examples/FindHighest/FindHighest.ino

Temps d’exposition, conseils

- La variable est donnée en us
  - 150us est un bon valeur de début (est la valeur par défaut)
- Peut être reglé dynamiquement
  - Trop de pixels saturés → diminuer le temps

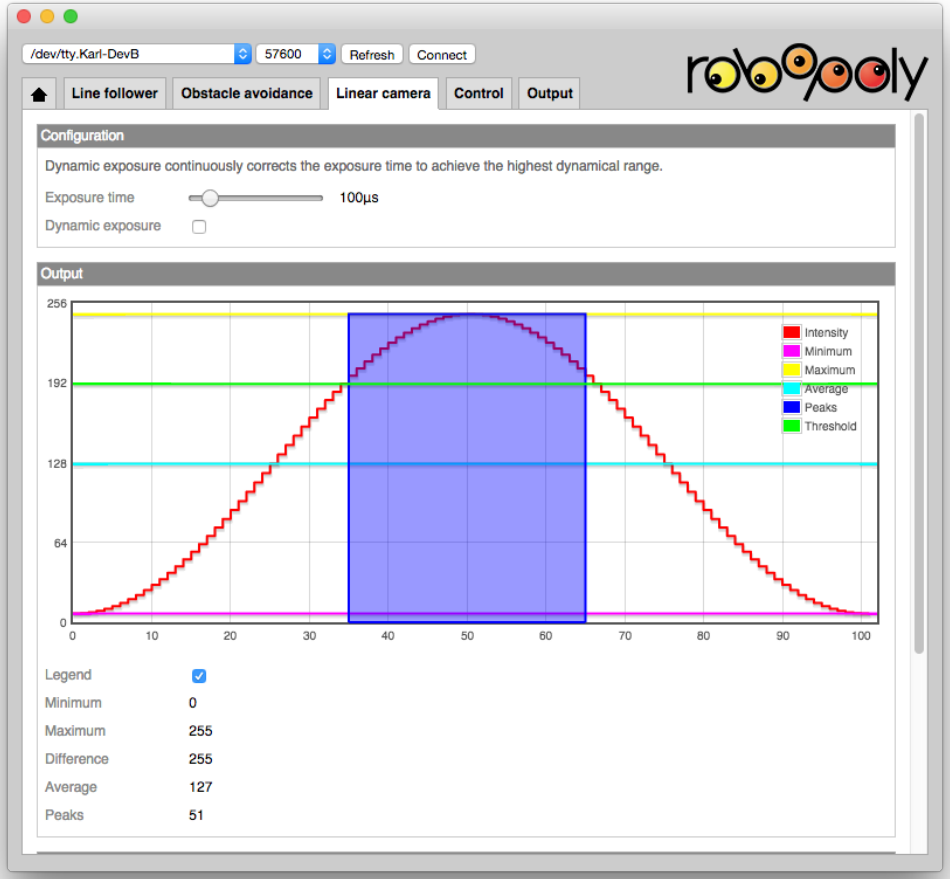
Quelque challenge - En ordre de difficulté

- Aller vers une lumière
- Ajuster dynamiquement le temps d'exposition
- Trouver la position de plusieurs pics à la fois
- Utiliser la caméra pour suivre une ligne
- Utiliser plusieurs caméras en même temps
- Pour les motivés, implementation du "optic flow"

http://en.wikipedia.org/wiki/Optical_flow

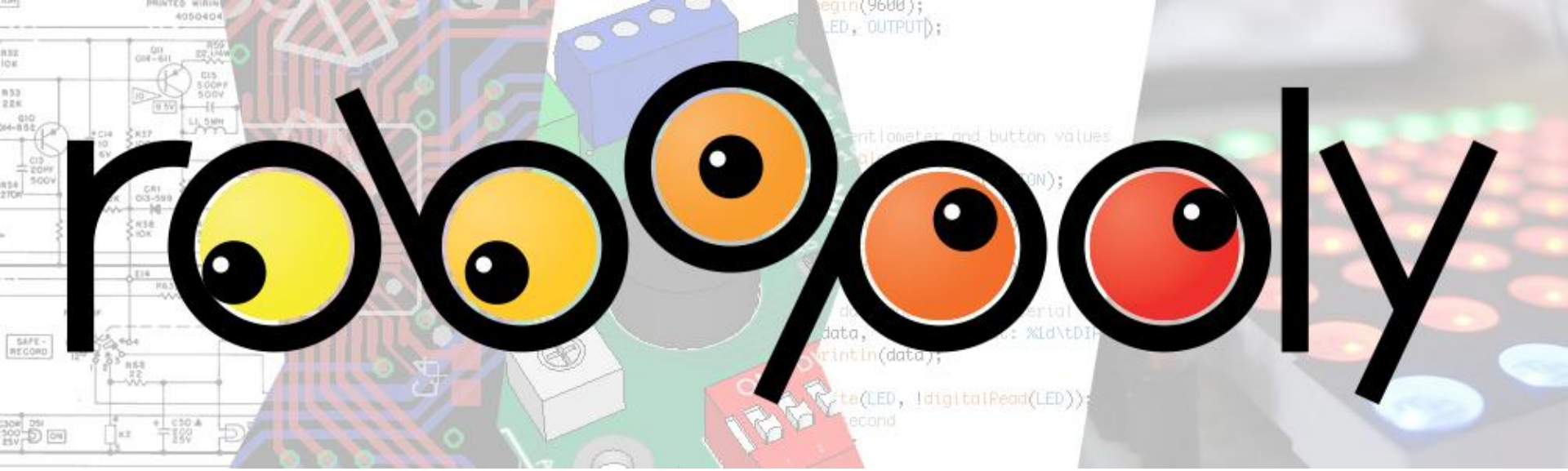
DEMO

Démonstration



Robopoly Control Center
A installer sur Chrome, fonctionne avec tous les OS

https://chrome.google.com/webstore/detail/robopoly-control-center/kimabfnmacaikklmibbbomomjffkfalo

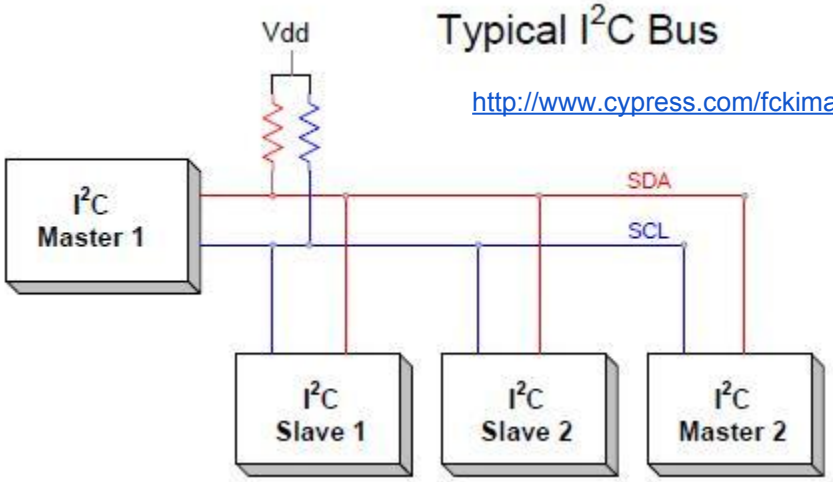


CAMERA LINEAIRE

I2C

Que est ce que c'est

- Protocole de communication série
  - SDA (Data)
  - SCL (Clock)
- Master - slave
- Open-Drain Data

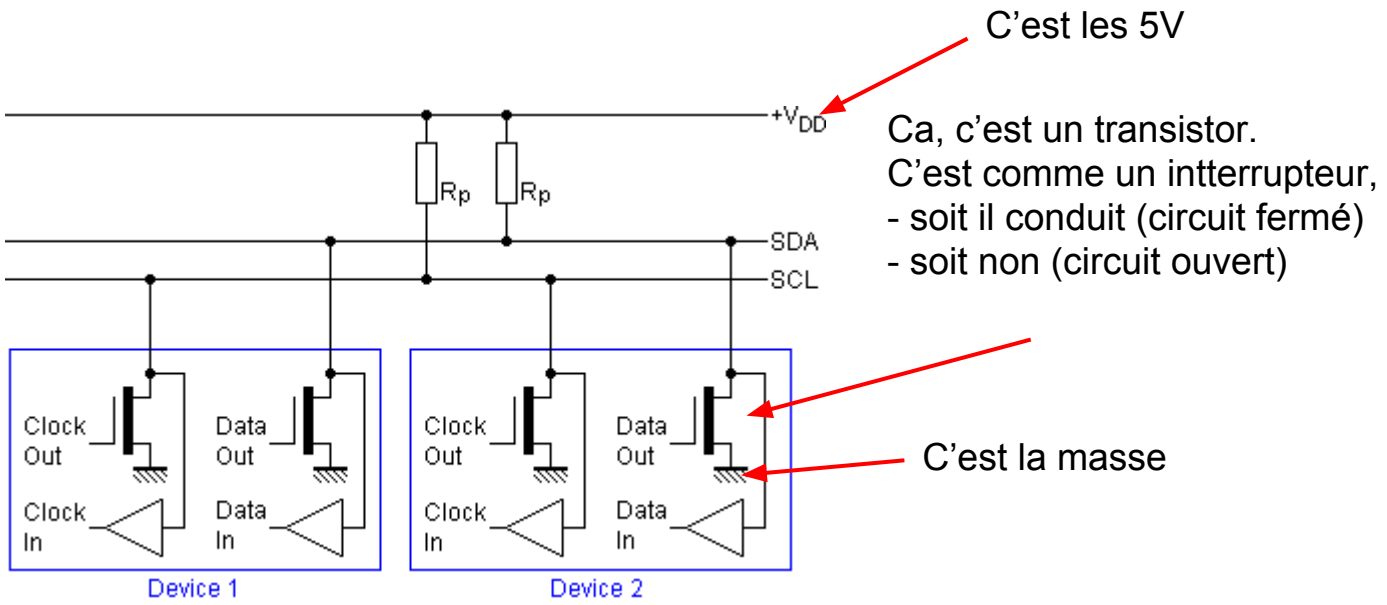


Comment?

- **Physical layer**
  - Soit, les composantes physiques du bus
- **Logic layer**
  - Ou, comment les signaux éléctriques sont interprétes par la logique du uC
- **Application layer**
  - Comme on l'utilise au niveau du code pour communiquer avec la caméra linéaire

Comment? - Physical layer

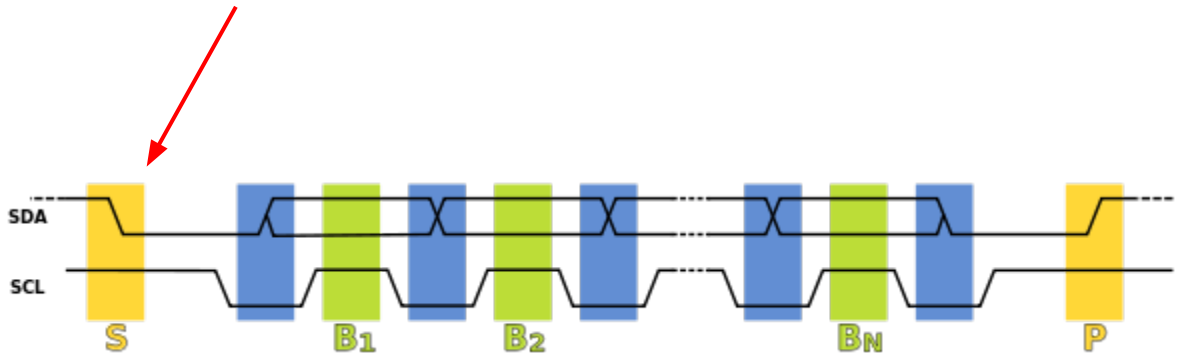
- La ligne est tirée à “1” par la resistance de “pull-up” **Rp**
- La ligne est tirée à “0” par le **Master** quand il met son transistor dans l’état de conduire



http://www.lammertbies.nl/picture/i2c_interface.png

Comment? - Logic layer

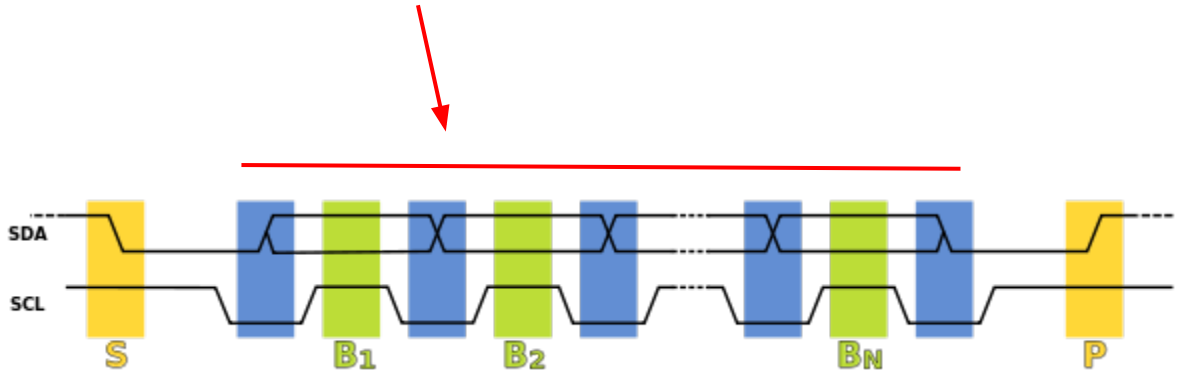
- Le **Master** “pull-down” (tire à “0”) la ligne SDA



http://en.wikipedia.org/wiki/I²C

Comment? - Logic layer

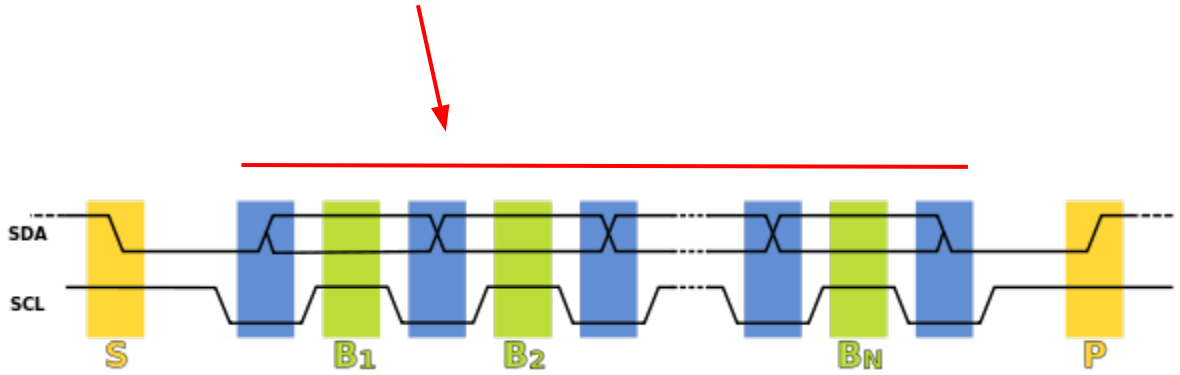
- Le **Master** “pull-down” (tire à “0”) la ligne SDA
- Le **Master** envoye un adresse de 7 bit. Le 8ème bit decide le mode d’écriture où lecture
  - Le **Master** ecrit le bit (zone bleue)
  - Au même temps, les **Master** pulse le clock



http://en.wikipedia.org/wiki/I²C

Comment? - Logic layer

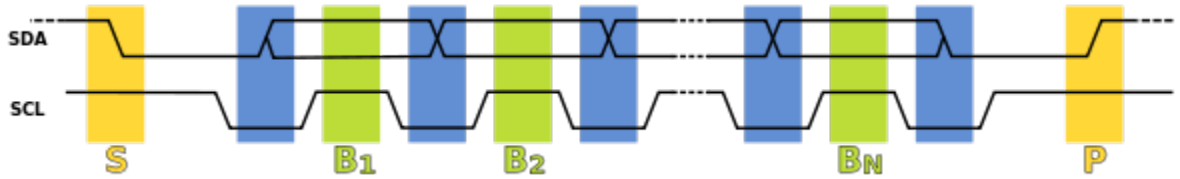
- Le **Master** “pull-down” (tire à “0”) la ligne SDA
- Le **Master** envoye un adresse de 7 bit. Le 8ème bit decide le mode d’écriture où lecture
  - Le **Master** ecrit le bit (zone bleue)
  - Au même temps, les **Master** pulse le clock
- Le **Slave** lit le bit **B1 .. Bn** quand le clock est à “1” (zone verte)



http://en.wikipedia.org/wiki/I²C

Comment? - Logic layer

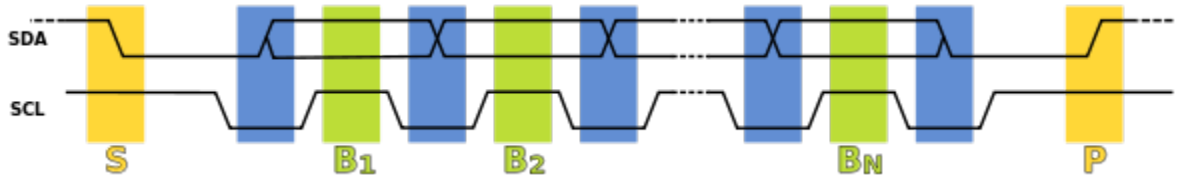
- Le **Master** “pull-down” (tire à “0”) la ligne SDA
- Le **Master** envoye un adresse de 7 bit. Le 8ème bit decide le mode d’écriture où lecture
  - Le **Master** ecrit le bit (zone bleue)
  - Au même temps, les **Master** pulse le clock
- Le **Slave** lit le bit **B1 .. Bn** quand le clock est à “1” (zone verte)
- Le **Slave** avec la bonne adresse “acknowledge” le debut de la transmission (pas montré dans l’image)



http://en.wikipedia.org/wiki/I²C

Comment? - Logic layer

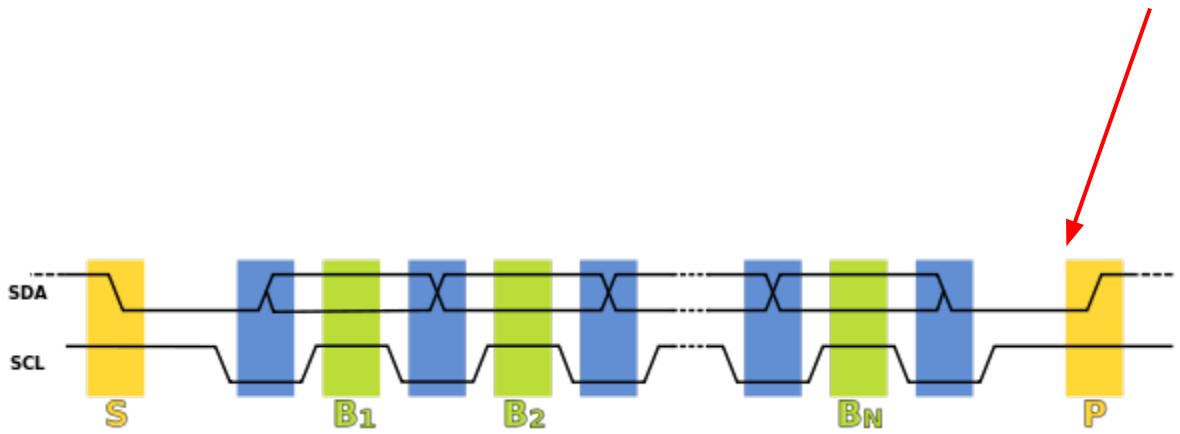
- Par la suite, le **Master** envoye le data (de même façon que l' adresse)
  - Que le **Slave** avec le bon adresse lit le data. Il "acknowledge" la transmission après chaque byte



http://en.wikipedia.org/wiki/I²C

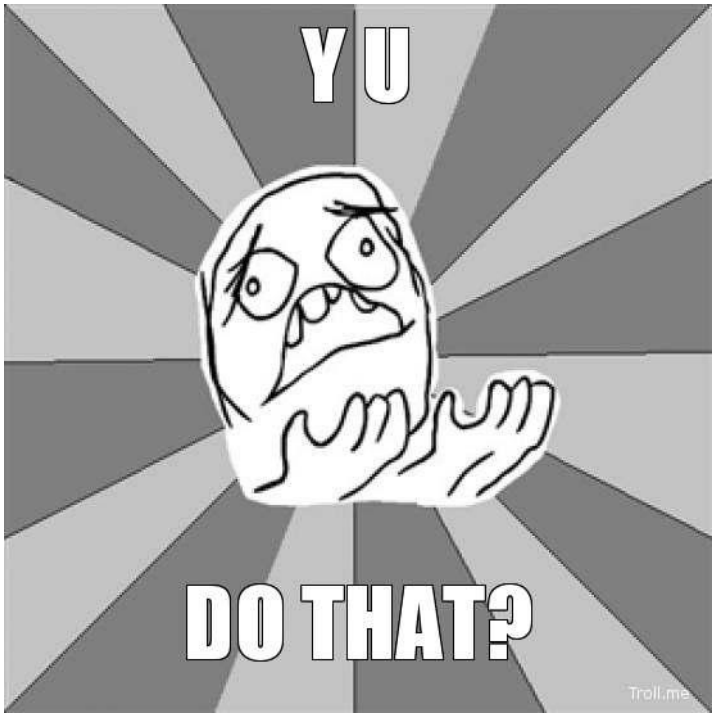
Comment? - Logic layer

- Par la suite, le **Master** envoye le data (de même façon que l'adresse)
  - Que le **Slave** avec le bon adresse lit le data. Il "acknowledge" la transmission après chaque byte
- La communication se termine quand le **Master** arrête de pulser le clock et relache la ligne de data



http://en.wikipedia.org/wiki/I²C

Pourquoi tout ça?



- Que 2 câbles entre peripheriques ou 2 traces sur un PCB au lieu 8 pour un byte plus 1 pour le clock.
- Que 2 pattes de la periphérique utilisées plutôt que 9 voir plus
- Permet aussi d'utiliser que deux connections pour faire communiquer jusqu'à 127 devices!

Comment? - Application layer - Write

- Le **Master** (Prismino) écrit l'adresse du registre du **Slave** (Caméra linéaire) qu'il veut écrire
  - Par example, temps d'intégration (sur 2 octets)
    - $0x06 byte de poids faible
    - $0x07 byte de poids fort
- La transmission est terminée
- Le **Master** envoye la valeur à écrire dans cet adresse
- La transmission est terminée

Comment? - Application layer - Read

- Le **Master** (Prismino) écrit l'adresse du registre du **Slave** (Caméra linéaire) qu'il veut lire
  - Par example
    - $0x00 pixel n°1
    - ...
    - $0x101 pixel n°102
- La transmission est terminée
- Le **Master** démarre une transmission en mode lecture
- Chaque byte est lu de façon consecutive
- La transmission est terminée quand on le veut

Des questions?

Hésitez pas passer poser des question

Next events!

- Lundi 8 decembre: **Presentation règles du grand concours et annonce de la date**