

Contrôle des entrées numériques 2/2

Applications mobiles et objets connectés 420-W48-SF

Objectifs

- Entrées numériques
 - Bouton monté en pull-up
 - Phénomène de rebonds : solution logicielle
- Entrées sur le port série

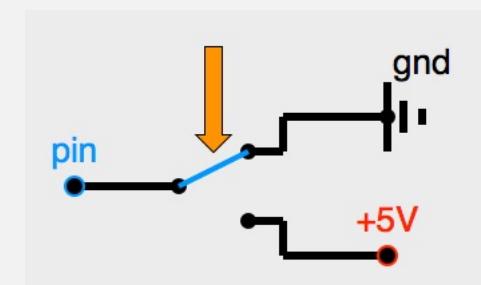
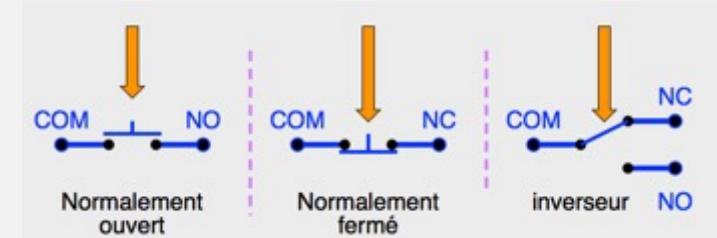
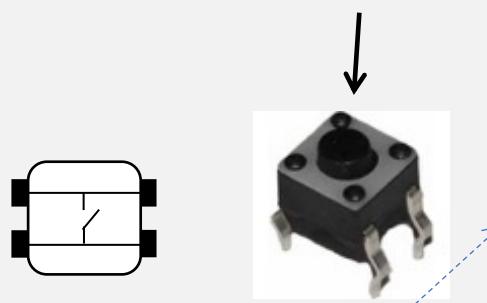
Entrée numérique

- Interrupteur

- Composant coupant ou laissant passer le courant
 - Modèles répondant à des besoins différents
 - Grande variété sur le marché

- Bouton poussoir est du type Normalement ouvert

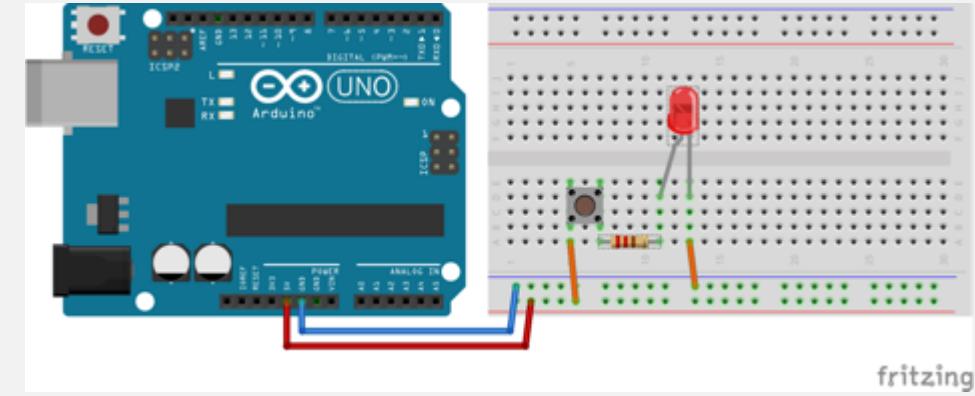
- La borne (pin) passe au niveau LOW lorsque reliée au ground
- La borne (pin) passe au niveau HIGH lorsque reliée au 5 V
- Tolérances aux écarts logiques en entrée pour Arduino



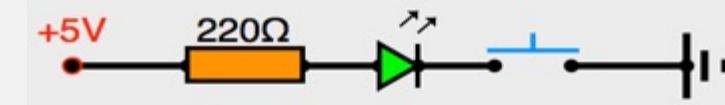
État	Logique	Électricité
LOW	0	0 V - 1,5 V
HIGH	1	3 V -- 5 V
Incertain	???	1,5 V - 3 V

Entrée numérique

- Contrôler l'éclairage de la DEL
 - Le bouton-poussoir « ferme » le circuit



- Bouton poussoir au laboratoire
 - 4 bornes sur 2 côtés
 - 2 bornes du même côté sont en contact si le bouton est enfoncé
 - Vérification possible avec le multimètre en position **Ω**



Entrée numérique

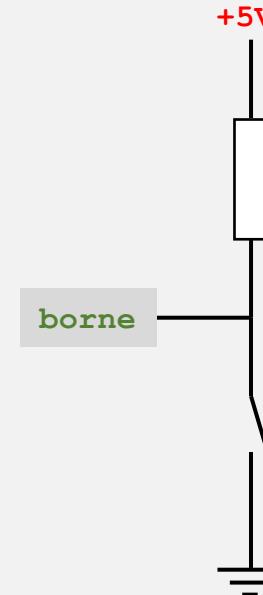
- Phénomène d'antenne parasite
 - Lorsque les bornes numériques sont utilisées en entrée, les fils agissent comme des antennes d'ondes électromagnétiques
 - Provoque de fausses interprétations:
 - Arduino peut lire des milliers de signaux par seconde, faisant croire à des mouvements rapides de l'interrupteur
 - L'oeil ne perçoit pas ce phénomène
 - Dépend de la qualité des matériaux de l'interrupteur
- Solution : usage d'une résistance Pull-up

```
void setup() {  
    pinMode(borneEntree, INPUT);
```

Entrée numérique

- Résistance Pull-up interne
 - Paramètre INPUT_PULLUP
 - Avantage : simplicité
 - Inconvénient : dépend de la valeur de la résistance interne, possibilité d'erreur de programmation (suppression du _PULLUP)
 - Utilisé durant les essais
- Résistance Pull-up externe
 - Avantage: contrôle absolu par le programmeur
 - Inconvénient: circuit supplémentaire à mettre en place
 - Utilisé sur un circuit permanent
- (Le format Pull_down existe aussi, mais non traité ici)

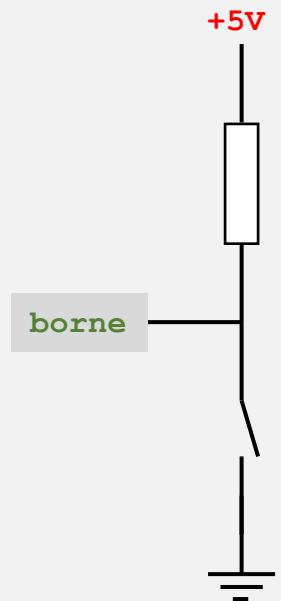
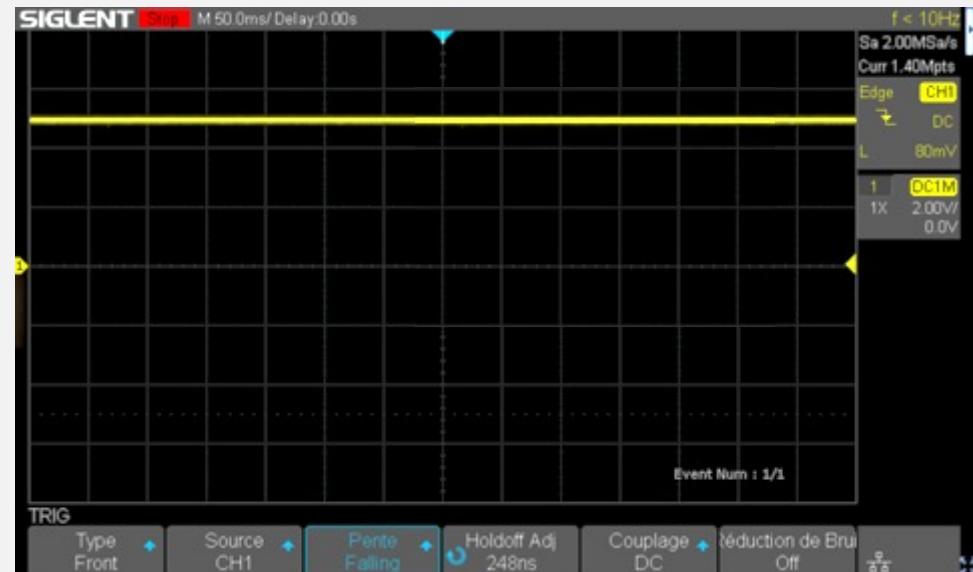
```
void setup() {  
    pinMode(borneEntree, INPUT_PULLUP);
```



Pull-up externe : principe de fonctionnement

Bouton en position **ouverte**

- L'électricité circule de la borne +5 V vers la borne d'entrée
- La borne détecte l'état HIGH

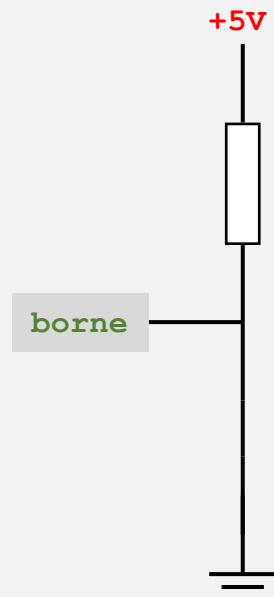
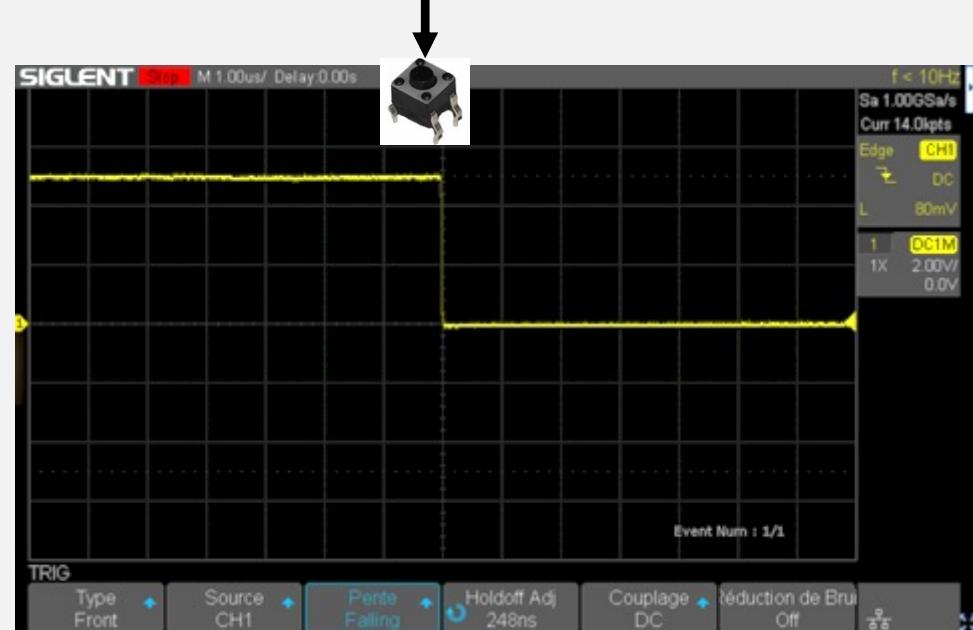


ATTENTION: la résistance Pull-up doit avoir une valeur élevée (5 KΩ ou plus)

Pull-up externe : principe de fonctionnement

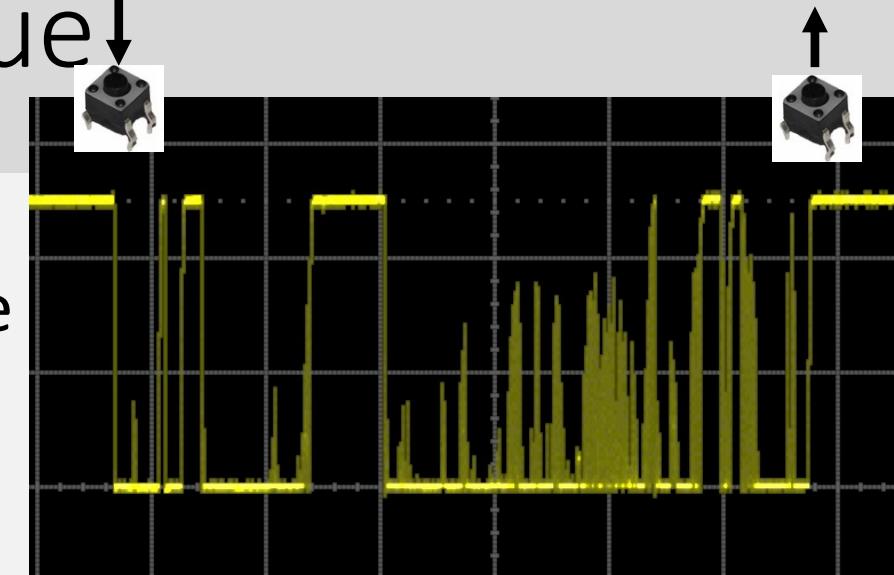
Bouton en position fermée

- Prend le chemin le plus facile
- La borne détecte l'état `LOW`
- **Attention sans résistance, il y aurait un court-circuit**

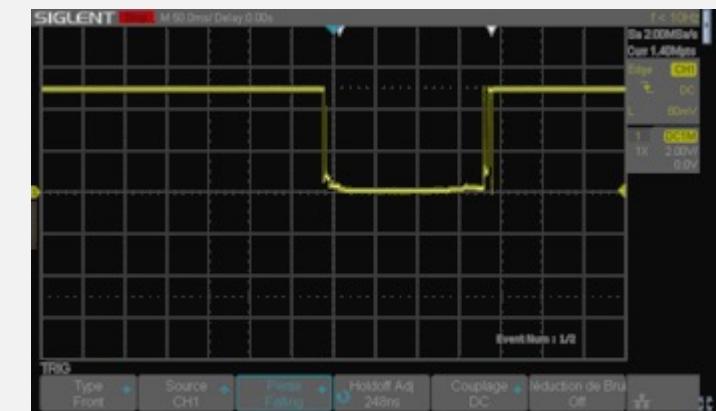
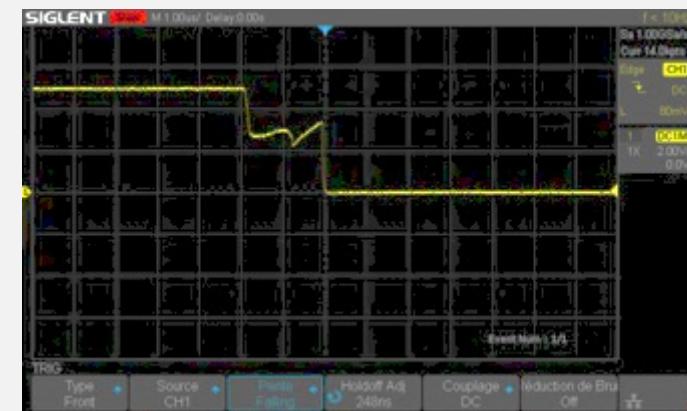
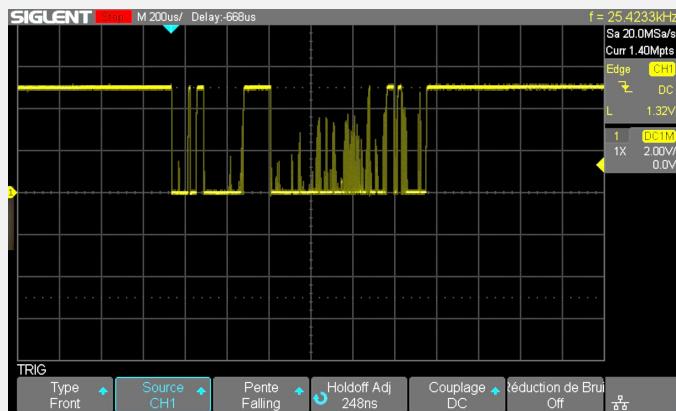


ATTENTION: la résistance Pull-up doit avoir une valeur élevée (5 K Ω ou plus)

Entrée numérique



- Rebond: émission d'impulsions de très courte durée
 - Causées par le passage rapide de 0 V à V_{max}
 - Provoque de fausses informations:
 - Arduino peut lire des milliers de signaux par seconde, faisant croire à des mouvements rapides de l'interrupteur
 - L'oeil ne perçoit pas ce phénomène

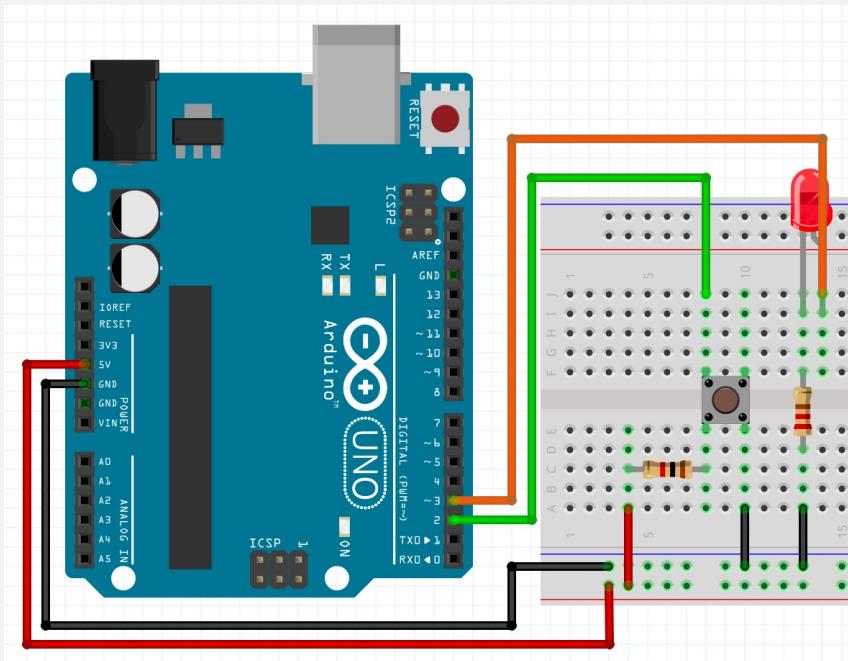


Entrée numérique

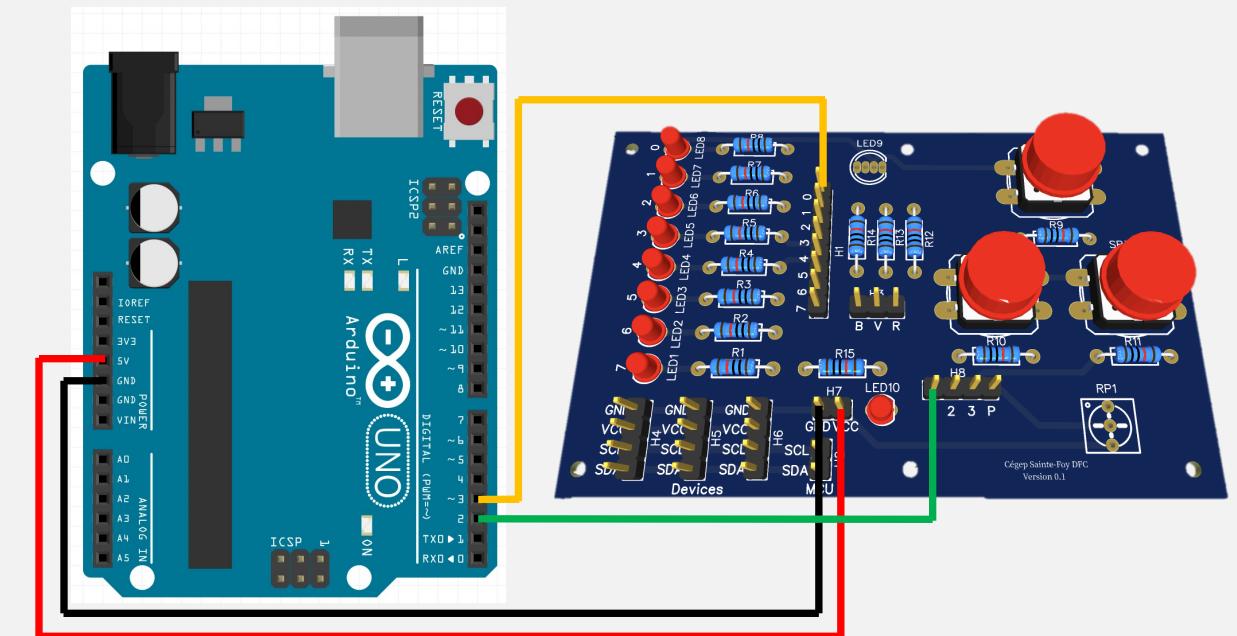
- Solutions
 - Matériel :
 - Interrupteurs certifiés anti-rebond (\$\$\$)
 - Utilisation d'un condensateur
 - Logiciel :
 - Attendre quelques millisecondes avant d'analyser l'état du bouton
 - Considérer que le bouton est appuyé si le délai mesuré comme appuyé dépasse une valeur prédéterminée
 - Etc.



Montage utilisé

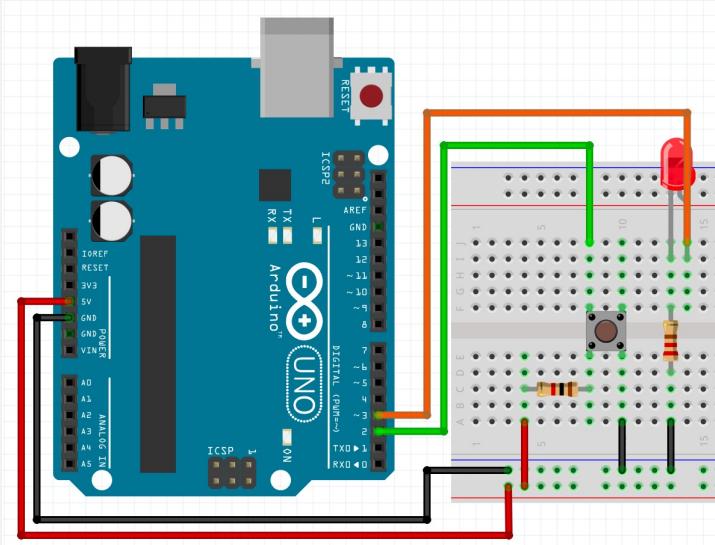


OU



Entrée numérique

- Programmation des bornes numériques en entrée
 - Déclaration du sens de l'usage : **pinMode (<NoBorne>, INPUT)**
 - Obtenir l'état de la borne : *variable = digitalRead (<NoBorne>)*
 - Retourne LOW ou HIGH



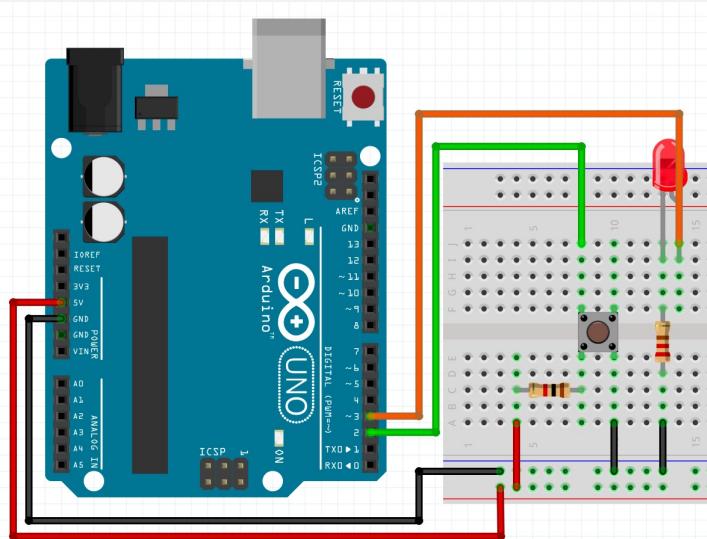
```
const int borneEntree = 2;
const int borneSortie = 3;
int valBouton;

void setup() {
    pinMode(borneEntree, INPUT);
    pinMode(borneSortie, OUTPUT);
}

void loop() {
    valBouton = digitalRead(borneEntree); // retourne LOW ou HIGH
    digitalWrite(borneSortie, !valBouton);
}
```

Entrée numérique – Solution Logicielle 1

- Attendre un certain temps avant de décider si le bouton est actionné
- Inconvénient : ralentissement du traitement général
- Utilisé seulement si le délai d'attente n'est pas critique au traitement



```
const int borneEntree = 2;
const int borneSortie = 3;
int valBouton;

void setup() {
    pinMode(borneEntree, INPUT);
    pinMode(borneSortie, OUTPUT);
}

void loop() {
    valBouton = digitalRead(borneEntree); // retourne LOW ou HIGH
    digitalWrite(borneSortie, !valBouton);
    delay(500);                         // s'assure de la stabilité du contact
}
```

Entrée numérique – Solution Logicielle 1 – Fonction delay

Users > pfleon > .platformio > packages > framework-arduino-avr > cores > arduino > C wiring.c

```
106 void delay(unsigned long ms)
107 {
108     uint32_t start = micros();
109
110     while (ms > 0) {
111         yield();
112         while ( ms > 0 && (micros() - start) >= 1000) {
113             ms--;
114             start += 1000;
115         }
116     }
117 }
```

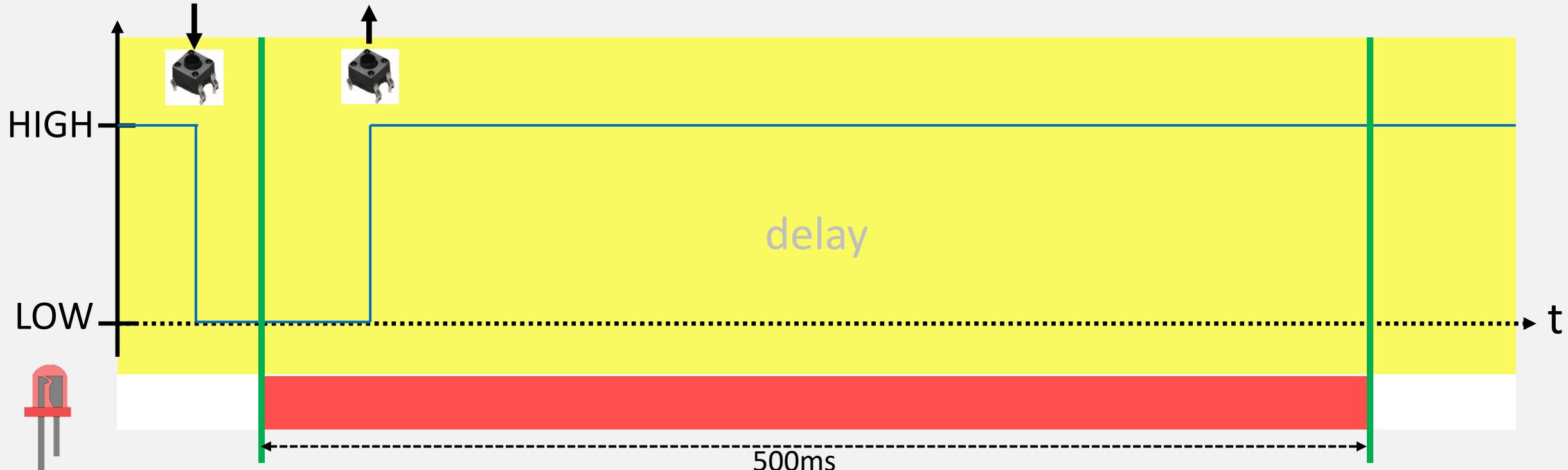
```
// Version simplifiée : ne fonctionne pas si
// dépassemens pour le calcul de fin
void monDelay(unsigned long p_ms) {
    unsigned long fin = micros() + p_ms * 1000;

    while (fin > micros()) {
        ; // yield();
    }
}
```

Users > pfleon > .platformio > packages > framework-arduino-avr > cores > arduino > C hooks.c

```
27
28 static void __empty() {
29     // Empty
30 }
31 void yield(void) __attribute__ ((weak, alias("__empty")));
32
```

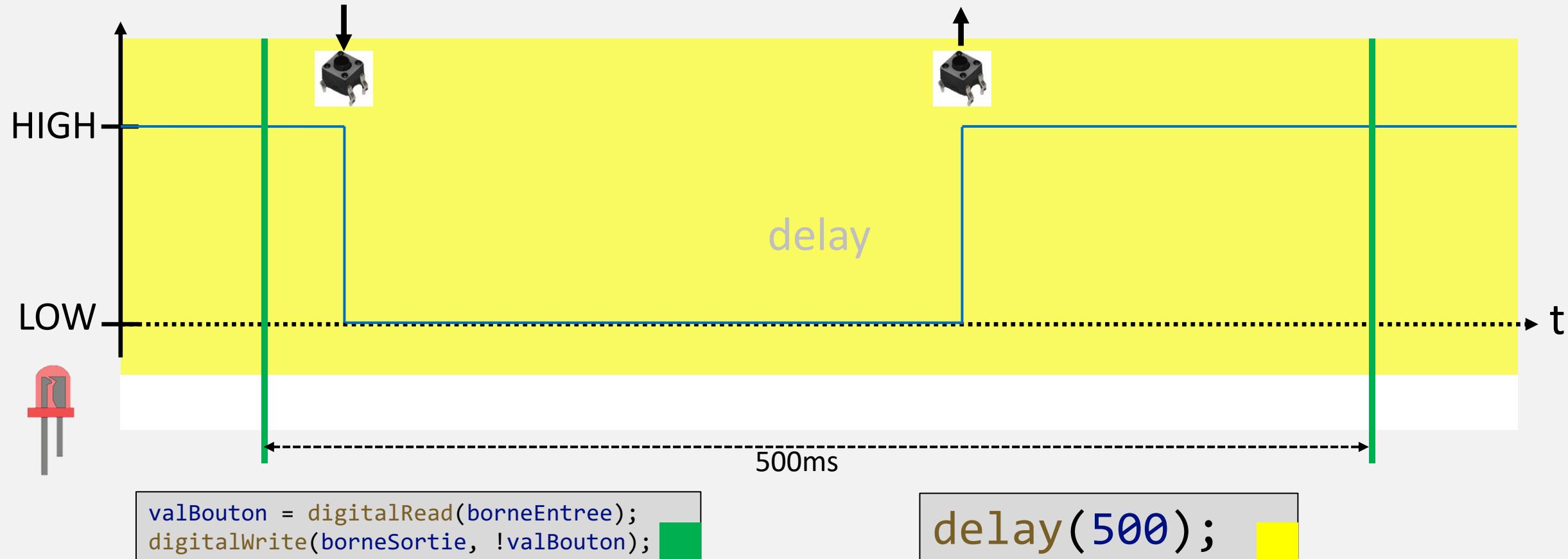
Entrée numérique – Solution Logicielle 1 – Problème



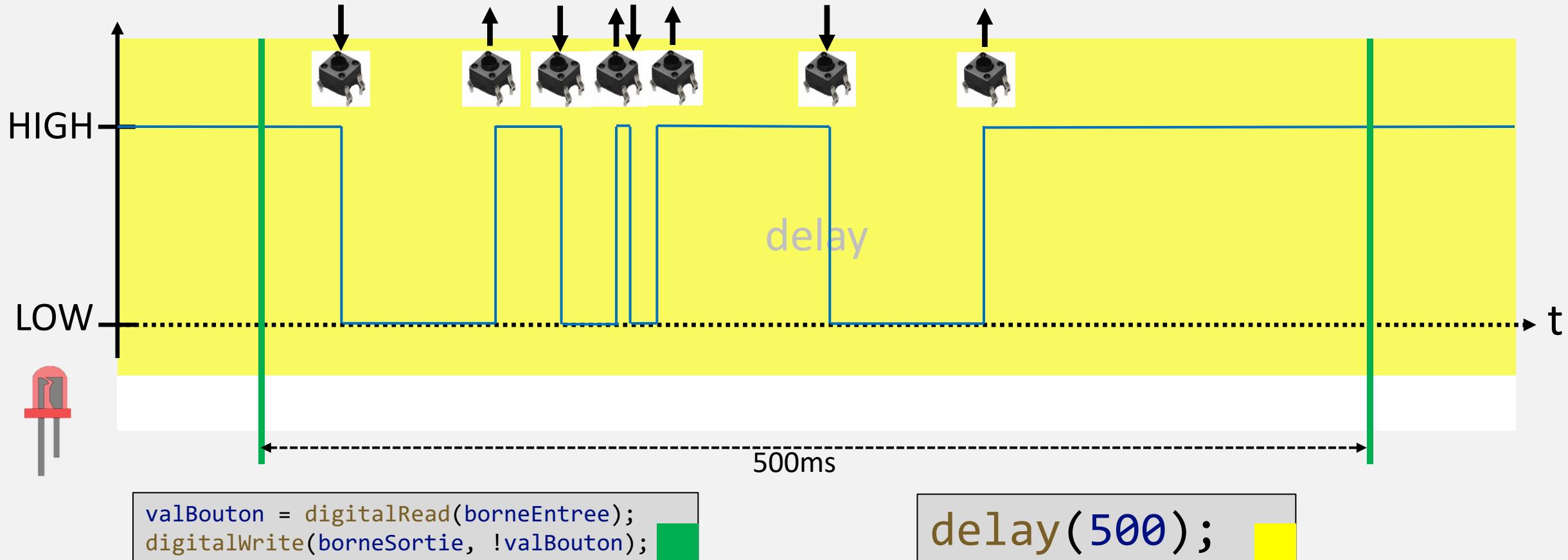
```
valBouton = digitalRead(borneEntree);  
digitalWrite(borneSortie, !valBouton);
```

```
delay(500);
```

Entrée numérique – Solution Logicielle 1 – Problème



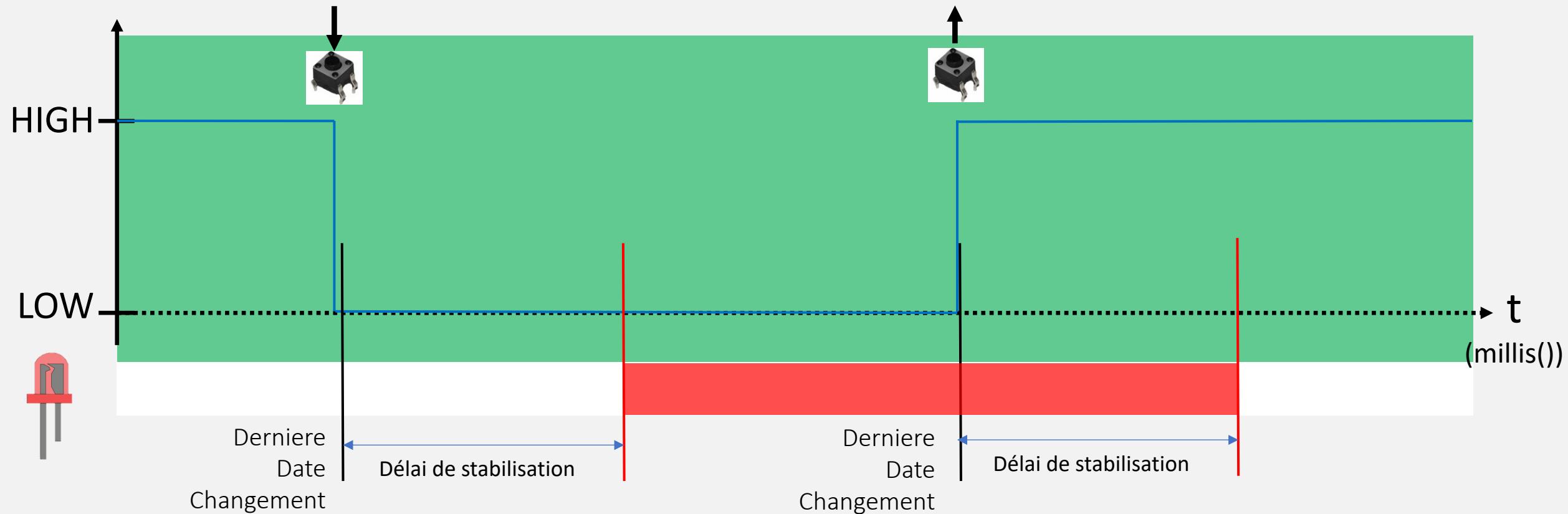
Entrée numérique – Solution Logicielle 1 – Problème



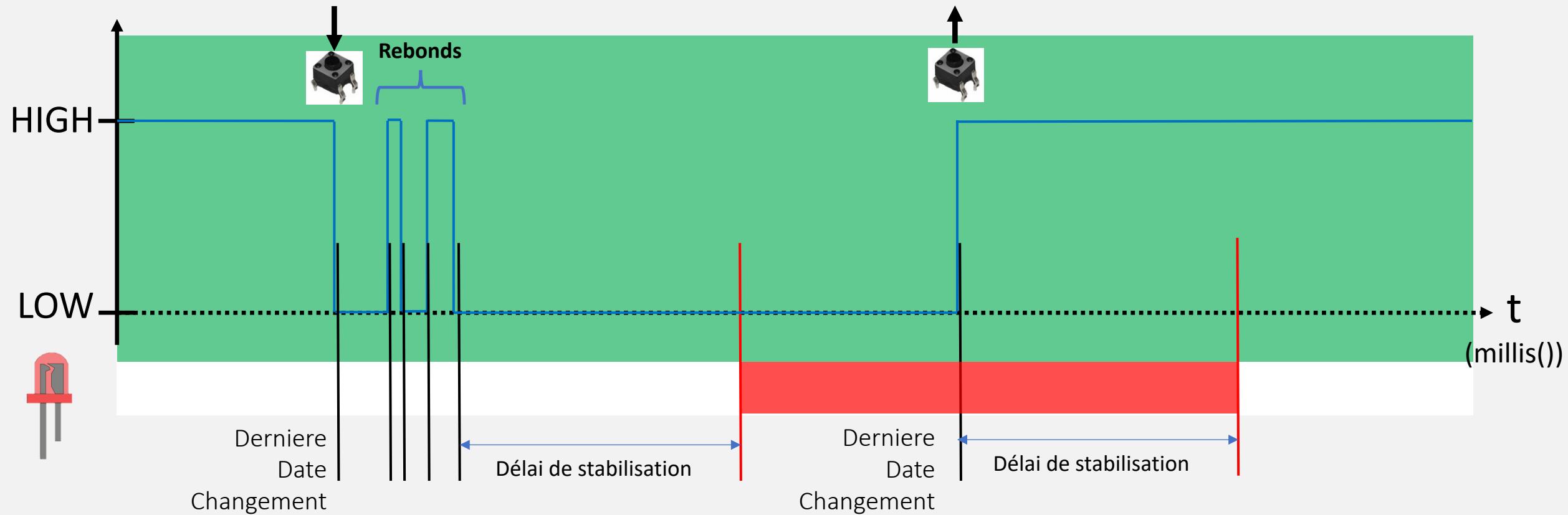
Entrée numérique – Solution Logicielle 2

- Mémorisation de l'état du bouton (LOW / HIGH) et de la date actuelle si changement par rapport au dernier état du bouton
- Si pas de changement d'état durant une certaine durée : durée de stabilisation (debounce)
 - Si détection d'un front descendant (`dernierEtatStableBouton == HIGH && etatBouton == LOW`)
 - Bouton en position considérée comme appuyée
 - Sinon si détection d'un front montant (`dernierEtatStableBouton == LOW && etatBouton == HIGH`)
 - Bouton en position considérée comme appuyée et relâchée
 - L'état du bouton actuel est le nouvel état stable du bouton (LOW / HIGH)

Entrée numérique – Solution Logicielle 2



Entrée numérique – Solution Logicielle 2

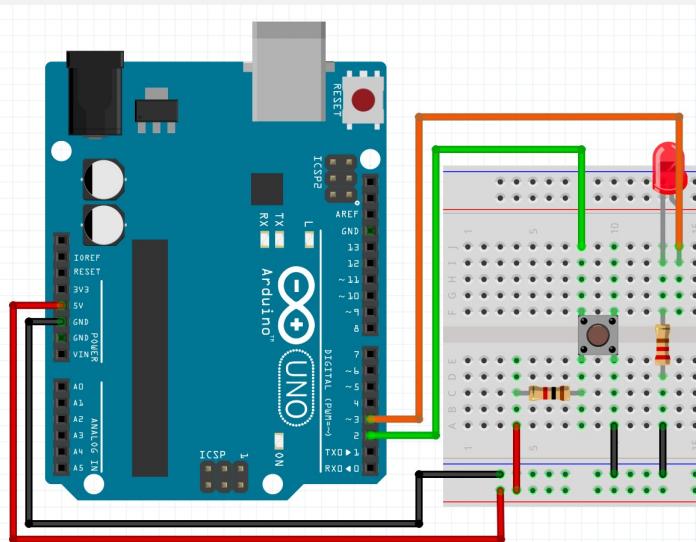


Entrée numérique – Solution logicielle 2

```
const int borneEntree = 2;
const int borneSortie = 3;

long derniereDateChangement = 0;
int dernierEtatBouton = HIGH;
int dernierEtatStableBouton = HIGH;
const int delaiMinPression = 25;

void setup(){
  pinMode(borneSortie, OUTPUT);
  pinMode(borneEntree, INPUT);
}
```



```
void loop(){
  int etatBouton = digitalRead(borneEntree);
  long dateActuelle = millis();
  if (etatBouton != dernierEtatBouton) {
    derniereDateChangement = dateActuelle;
    dernierEtatBouton = etatBouton;
  }

  if(dateActuelle - derniereDateChangement > delaiMinPression) {
    if (dernierEtatStableBouton == HIGH && etatBouton == LOW) {
      // bouton appuyé
      // Action à réaliser
      digitalWrite(borneSortie, HIGH);
    } else if (dernierEtatStableBouton == LOW && etatBouton == HIGH) {
      // bouton relaché
      // ... et donc comme click
      // Action à réaliser
      digitalWrite(borneSortie, LOW);
    }
    dernierEtatStableBouton = etatBouton;
  }
}
```

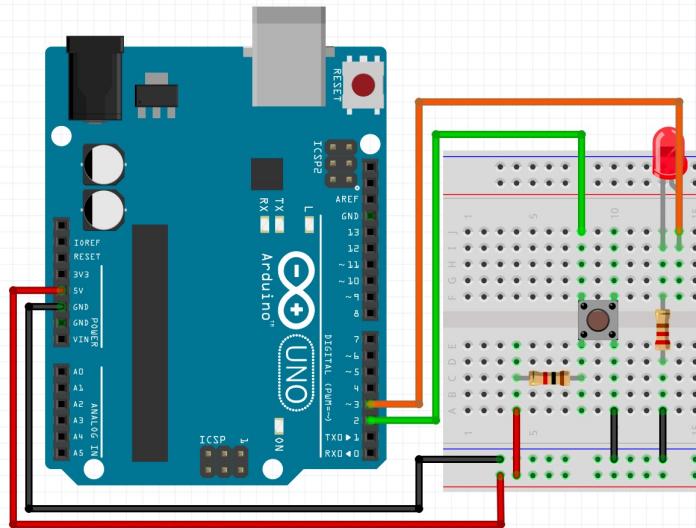
Entrée numérique – Solution logicielle 2 – Simulation d'un bouton on/off

```
const int borneEntree = 2;
const int borneSortie = 3;

int dernierEtatLed = LOW;

long derniereDateChangement = 0;
int dernierEtatBouton = HIGH;
int dernierEtatStableBouton = HIGH;
const int delaiMinPression = 25;

void setup(){
    pinMode(borneSortie, OUTPUT);
    pinMode(borneEntree, INPUT);
}
```



```
void loop(){
    int etatBouton = digitalRead(borneEntree);
    long dateActuelle = millis();
    if (etatBouton != dernierEtatBouton) {
        derniereDateChangement = dateActuelle;
        dernierEtatBouton = etatBouton;
    }

    if(dateActuelle - derniereDateChangement > delaiMinPression) {
        if (dernierEtatStableBouton == HIGH && etatBouton == LOW) {
            // bouton appuyé
            // Action à réaliser
        } else if (dernierEtatStableBouton == LOW && etatBouton == HIGH) {
            // bouton relaché
            // ... et donc comme click
            // Action à réaliser
            dernierEtatLed = !dernierEtatLed;
            digitalWrite(borneSortie, dernierEtatLed);
        }
        dernierEtatStableBouton = etatBouton;
    }
}
```

Références

- <https://www.arduino.cc/reference/en/>
- <https://openclassrooms.com/fr/courses/2778161-programmez-vos-premiers-montages-avec-arduino> : images prises dans ce cours