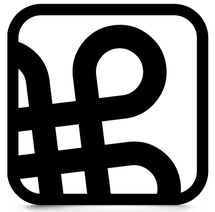


Macoun

Handerkennung mit einem gepulsten Stift

Maik Borkenstein
Peter Kämpf

Ablauf

- Problem
- Lösung
- Was wir auf dem Weg dorthin gelernt haben

Problem

Handschrift auf dem iPad

- Handschrift hilft beim Lernen
- Das iPad ist für Gesten gedacht, nicht für Handschrift
- Wenn man beim Schreiben nicht die Hand auflegen kann, kann man nicht vernünftig schreiben
- Das iPad braucht einen Stift, der ein Handauflegen erlaubt

Analyse

- Das Tastfeld misst 60 mal pro Sekunde die lokale Kapazität
- Minimale Kapazitätsänderung zur Erkennung: 1µF
- Minimale Kontaktfläche zur Erkennung braucht 6mm Nupsi

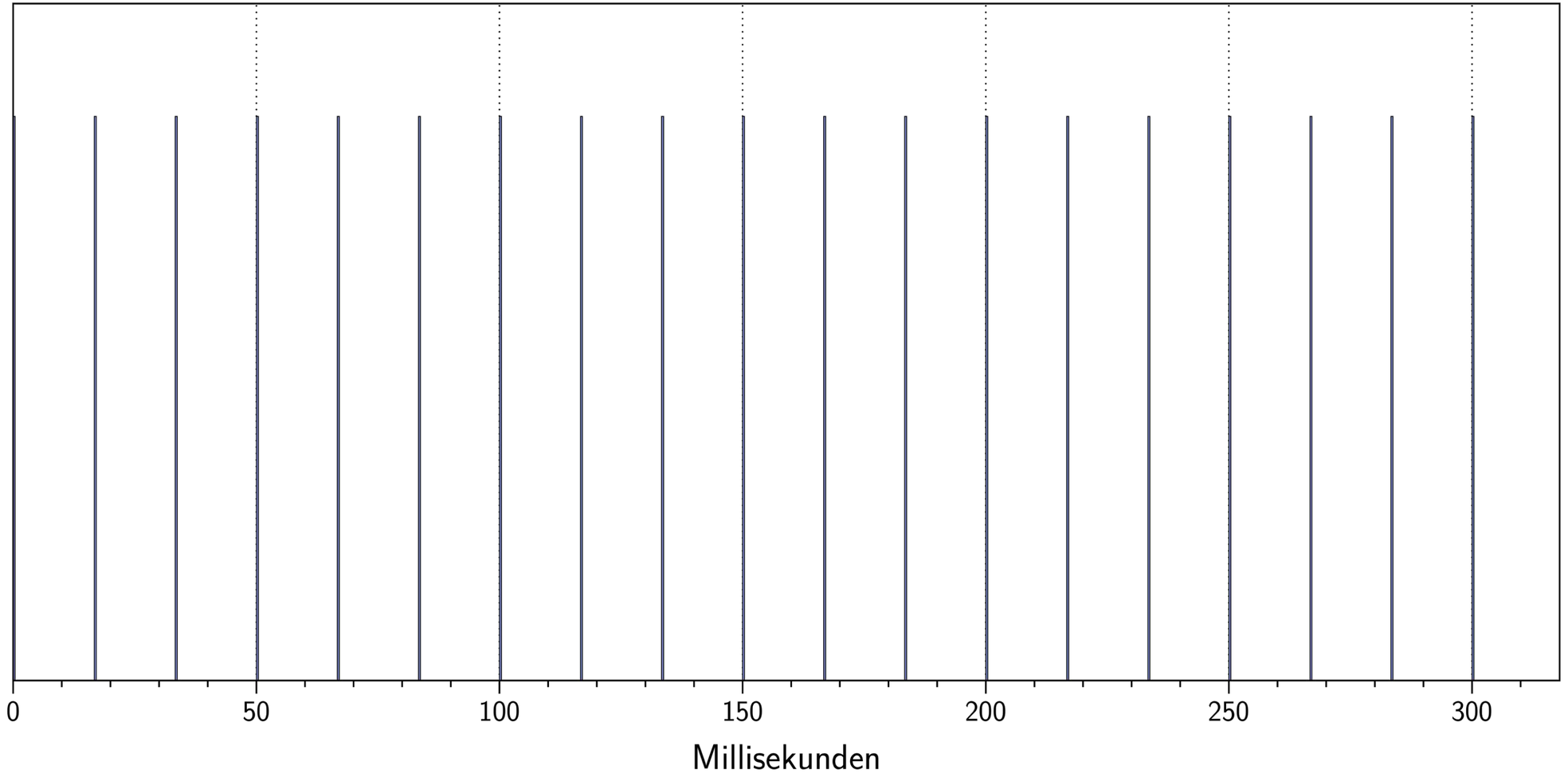
Lösung

Lösungsansatz

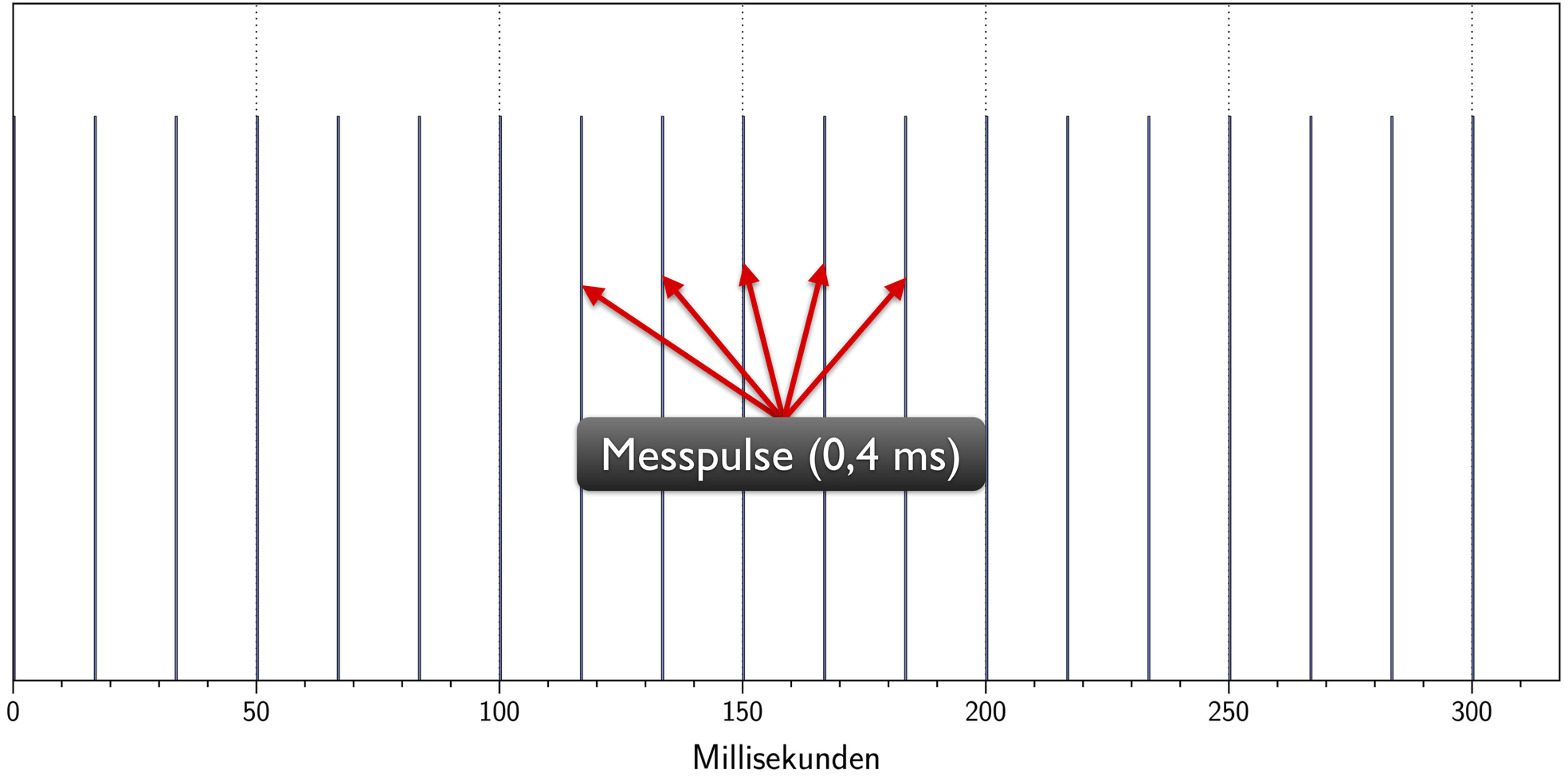
- Wir spielen „jetzt siehst Du mich — jetzt nicht“ mit dem Tastfeld
- Umschalten der Leitfähigkeit der Spitze eines Stiftes
- Maximale Frequenz < halbe Tastfeld-Frequenz

Unser aktiver Stift

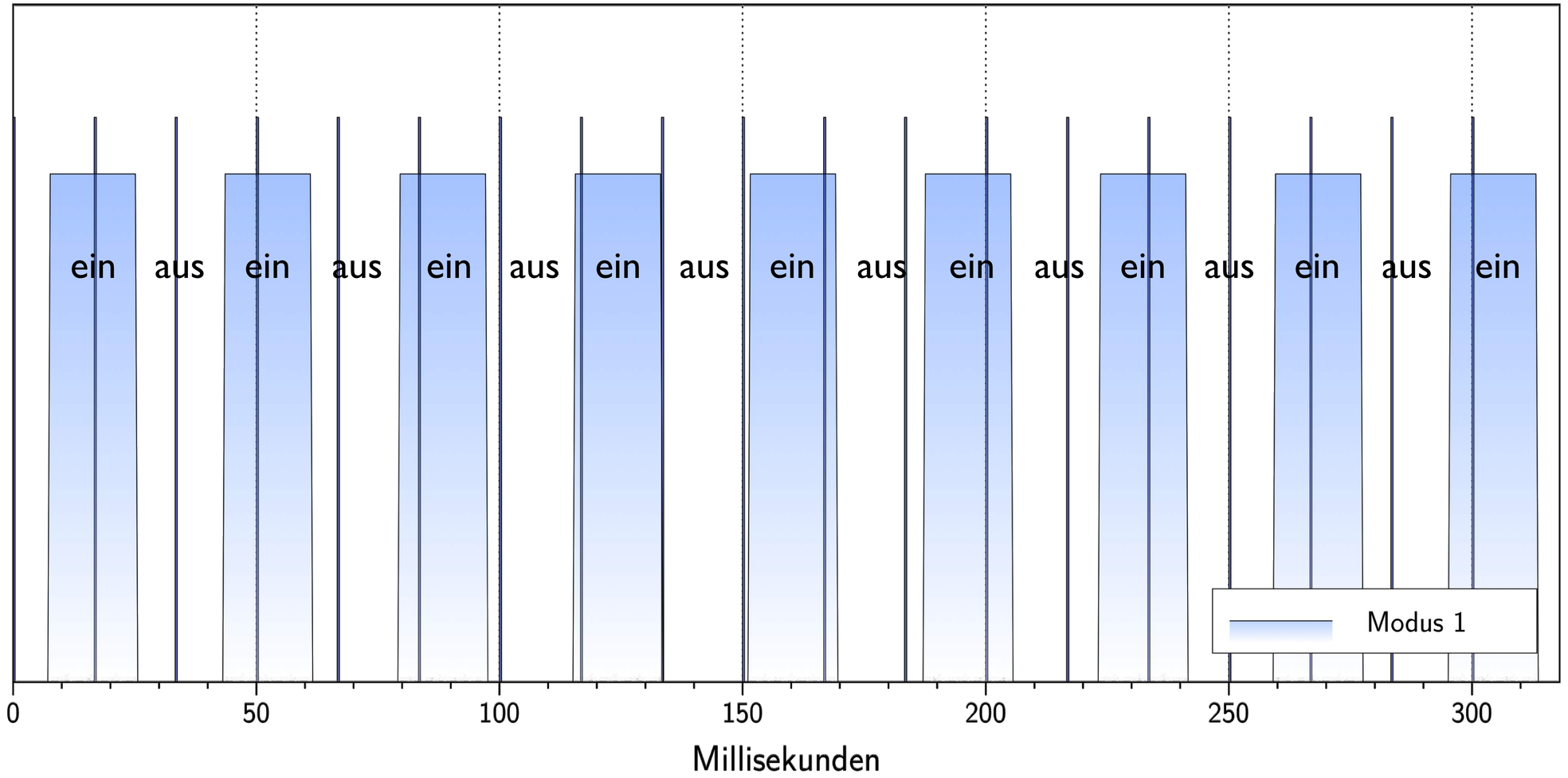
Schaltmuster



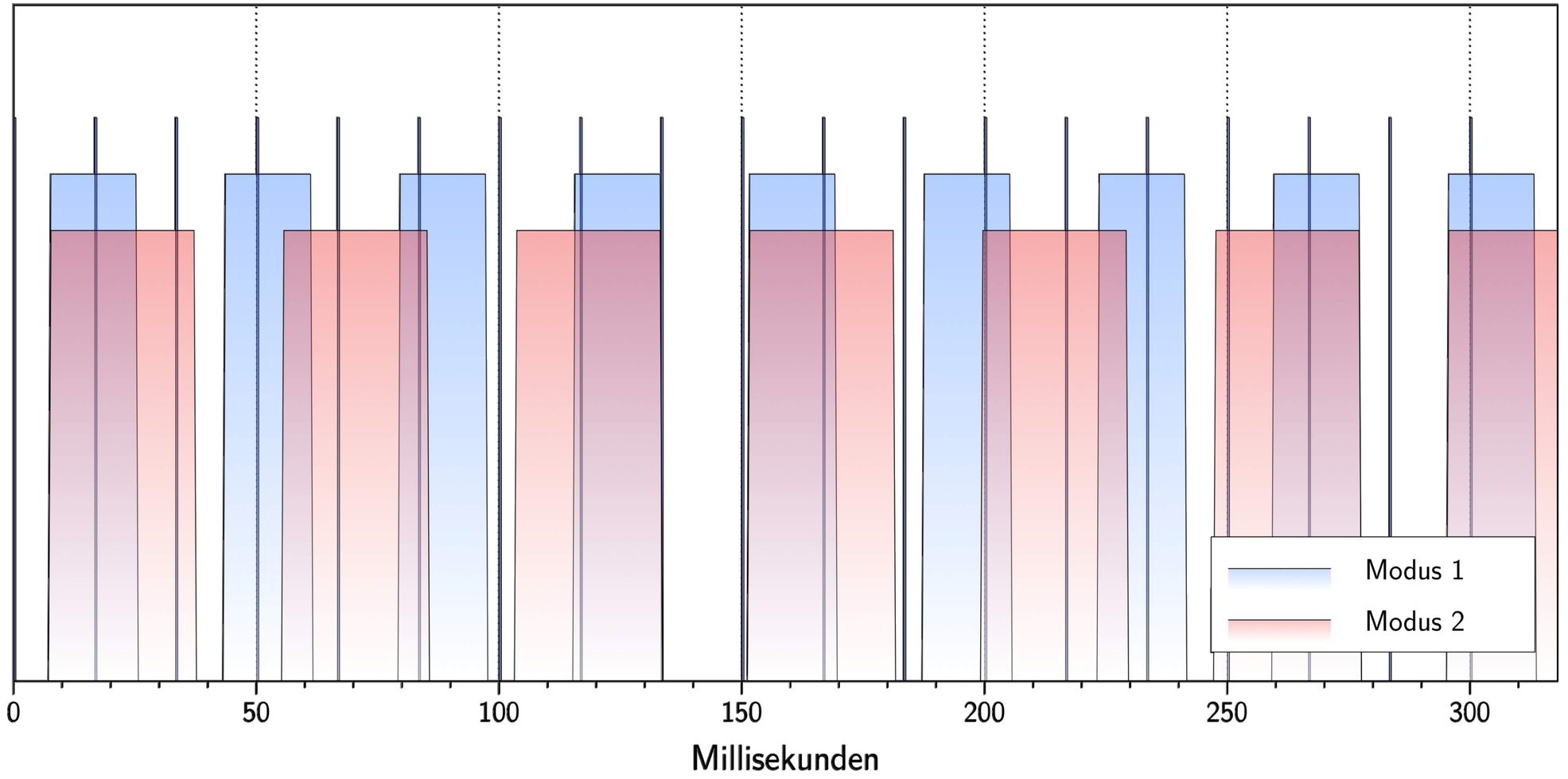
Schaltmuster



Schaltmuster



Schaltmuster



pulsedTouchRecognizer

- Zwei Möglichkeiten zur Einbettung im ViewController

```
- (void)viewDidLoad {

    if (self.touchAnalyzer) {
        self.tAn  = [[SID_PulsedTouchAnalyzer alloc] initWithDelegate:self];

    } else {
        self.tRec = [[SID_PulsedTouchRecognizer alloc] initWithTarget:self
                                                action:@selector(handleTouchEnded:)];
        [self.view addGestureRecognizer:self.tRec];
    }
}
```

Neues Objekt: SID_Touch

```
@interface SID_Touch : NSObject

@property (assign, nonatomic) CGPoint point;
@property (assign, nonatomic) NSTimeInterval timestamp;
@property (assign, nonatomic) CGPoint velocity;
@property (strong, nonatomic) NSString *lineID;
@property (assign, nonatomic) NSInteger classification;
@property (assign, nonatomic) NSInteger state;

@end
```

- `classification`: Unbestimmt, Stift, Finger oder Hand

Rückgabewerte

- touchAnalyzer: Events weiterleiten, Rückgabe per NSDictionary

```
- (void) touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event {
    [self.tAn SID_touchesBegan:touches withEvent:(UIEvent *)event];
}
…
- (void) SID_linesChangedClass:(NSDictionary *)touchesDict {
   for (NSString *key in touchesDict) {
        NSArray *touchArray   = touchesDict[key];
        SID_Touch *firstTouch = [touchArray firstObject];
        switch (firstTouch.classification) {
        }
    }
}
```

Rückgabewerte

- touchRecognizer: Rückgabe eines NSDictionary

```
- (void) handleTouchEnded:(SID_PulsedTouchRecognizer *)tRec {

   NSDictionary *newIncrements = [self.tRec get_SID_LineIncrement];

   for (NSString *key in newIncrements) {
        NSArray *touchArray   = newIncrements[key];
        SID_Touch *firstTouch = [touchArray firstObject];
        switch (firstTouch.classification) {
        }
    }
}
```

Und das funktioniert?

Demo

Was wir gelernt haben

Erkennung verschiedener Muster

- Minimaler Kontakt von ca. 200 ms
- Kompromiss aus Musteranzahl, Genauigkeit und Geschwindigkeit
- Vermeidung von Frequenznachbarschaft
- 2 verschiedenartige Ansätze

Erkennung verschiedener Muster

Stift 130 mit 4 Zyklen

	4 aus	5 aus	6 aus	7 aus	8 aus	9 aus
3 an		1	1	1, 3	3	
4 an	1	1	1, 3	3	3	
5 an	1	1, 3	1, 3	3		
6 an	1, 2	1, 3	3	3		4
7 an	1, 2	2, 3	2	3, 4	4	
8 an	2	2	2, 4	4	4	
9 an	2	2, 4	4	4		
10 an	2, 4	4	4			
11 an	4	4				
12 an	4					

Erkennung verschiedener Muster

	M1-1	M1-2
an	*	16
aus	*	16
an	33	16
aus	16	33
an	16	16
aus	16	16
an	33	16

- Suche nach Anomalien innerhalb des Musters
- Zulassen von Fehlern je nach Länge der Anomalie
- maximal 3,5 Perioden Erkennungszeit
- bei langen Perioden sinkt Erkennungszeit auf 1,5 bis 2,5 Perioden

Extrapolation

- Vergleich des neuen Punktes mit Extrapolation der Linie(n)
- Nutzung zur Fortführung der angezeigten Linie
- Kompromiß zwischen Unruhe und Abstand Spitze-Linie

Demo

Extrapolation

```
static inline CGPoint extrapolationFromLine(SID_Line *line) {
    NSUInteger length       = [line.touches count];
    SID_Touch *t2           = line.touches[length-3];
    SID_Touch *t1           = line.touches[length-2];
    SID_Touch *t0           = line.touches[length-1];
    NSTimeInterval timeInt1 = 60 * (t0.timestamp - t1.timestamp);
    NSTimeInterval timeInt2 = 60 * (t1.timestamp - t2.timestamp);

    CGFloat d01x = (t0.point.x - t1.point.x) / timeInt1;
    CGFloat d01y = (t0.point.y - t1.point.y) / timeInt1;
    CGFloat d12x = (t1.point.x - t2.point.x) / timeInt2;
    CGFloat d12y = (t1.point.y - t2.point.y) / timeInt2;
    return extrapolatedPoint = CGPointMake(t0.point.x + 2*d01x - d12x,
                                           t0.point.y + 2*d01y - d12y);
}
```

Weitere gestureRecognizer

- Im Prinzip möglich
- pulsedTouchRecognizer sollte Vorrang haben
- Brauchen längeren Fingerkontakt zur Aktivierung

Demo

Fragen?

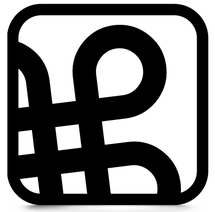
Antworten (und Source) gibt es hier:

www.stabilodigital.com

User: appdeveloper
PW: letmein

(Tut uns leid, aber die Firma läßt es nicht anders zu)

Vielen Dank



Macoun