# Eingaben

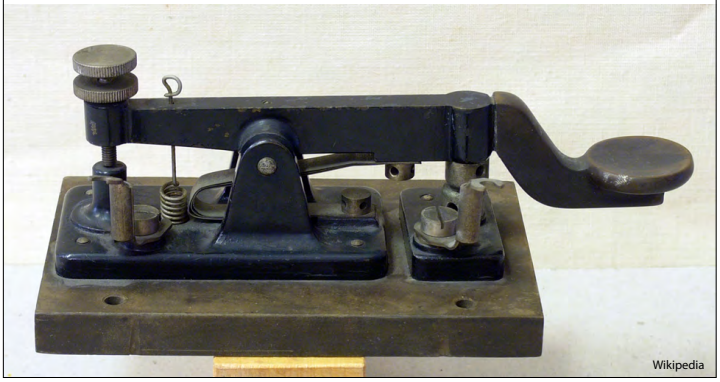
Programmieren für Ingenieure
Sommer 2015

Andreas Zeller, Universität des Saarlandes

# Themen heute

- Eingaben
- Zuweisungen
- Zeitmessung

# Taster



Wikipedia

# Ziel

Wenn Taste gedrückt,
soll LED leuchten

# Sensor abfragen

- Wir haben bereits digitalWrite() kennengelernt, die Daten digital ausgibt
- Neu: digitalRead() liest Daten digital *ein*

```
digitalRead(pin_number)
```

  hat den Wert HIGH, wenn + am Pin anliegt; und LOW, wenn nicht.

# Sensor abfragen

- Wenn digitalRead() = HIGH, soll die LED leuchten
- Wenn digitalRead() = LOW, soll die LED ausgehen
- … und das ganze immer wieder

# Sensor abfragen

```
int ledPin = 13;   // Die LED
int buttonPin = 8; // Der Taster

void setup() {
    pinMode(ledPin, OUTPUT);
    pinMode(buttonPin, INPUT);
}
```

# Sensor abfragen

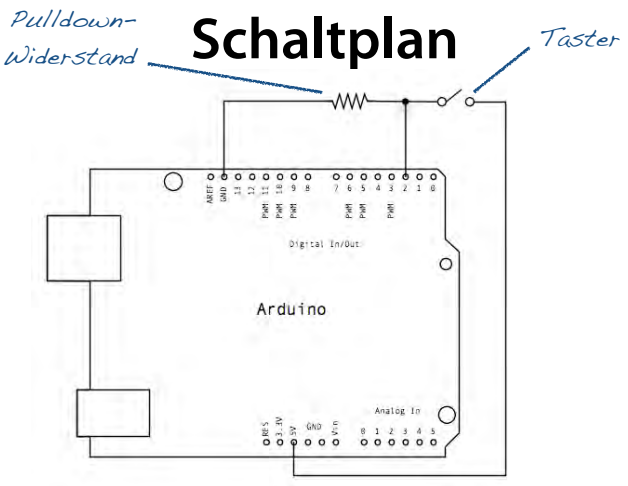
```
int ledPin = 13;   // Die LED
int buttonPin = 8; // Der Taster

void setup() {
    pinMode(ledPin, OUTPUT);
    pinMode(buttonPin, INPUT);
}

void loop() {
    if (digitalRead(buttonPin) == HIGH) {
        digitalWrite(ledPin, HIGH);
    }
    if (digitalRead(buttonPin) == LOW) {
        digitalWrite(ledPin, LOW);
    }
}
```

*prüfe Sensor*

# Schaltplan

*Pulldown-Widerstand*

*Taster*

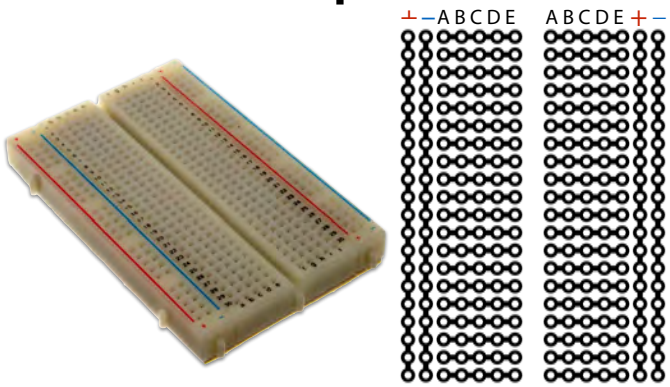



Quelle: arduino.cc

# Pullup und Pulldown

- Ist an einem digitalen Eingang nichts angeschlossen (= weder + noch –), ist der Eingangswert *undefiniert*
- Ein Pullup- bzw. Pulldown-Widerstand (Arduino: 10kΩ, Galileo: 1kΩ) definiert den Pegel, wenn sonst nichts anliegt
- Wird von Taster kurzgeschlossen

# Steckplatine






# Sensor abfragen

```
int ledPin = 13;   // Die LED
int buttonPin = 8; // Der Taster

void setup() {
    pinMode(ledPin, OUTPUT);
    pinMode(buttonPin, INPUT);
}

void loop() {
    if (digitalRead(buttonPin) == HIGH) {
        digitalWrite(ledPin, HIGH);
    }
    if (digitalRead(buttonPin) == LOW) {
        digitalWrite(ledPin, LOW);
    }
}
```

# Demo

- LED und Taster anschließen; blinken lassen

## Werte speichern

- In unserem Programm wird der Sensor 2x hintereinander abgefragt, obwohl 1x reichen würde
- Wir müssen das Ergebnis *speichern*
- Wir können das Ergebnis einer Variablen *zuweisen*.

## Zuweisung

- Die Anweisung

```
name = wert
```

  bewirkt, dass die Variable *name* den neuen Wert *wert* hat.
- Im weiteren Programmablauf liefert jeder spätere Zugriff auf die Variable den Wert *wert (*bis zur nächsten Zuweisung)

# Sensor abfragen

```
int ledPin = 13;      // Die LED
int buttonPin = 8;    // Der Taster


void loop() {
    if (digitalRead(buttonPin) == HIGH) {
        digitalWrite(ledPin, HIGH);
    }
    if (digitalRead(buttonPin) == LOW) {
        digitalWrite(ledPin, LOW);
    }
}
```

prüfe Sensor

# Sensor abfragen

```
int ledPin = 13;      // Die LED
int buttonPin = 8;    // Der Taster
int tasterStatus;     // Der Tasterzustand


void loop() {
    tasterStatus = digitalRead(buttonPin);
    if (tasterStatus = HIGH) {
        digitalWrite(ledPin, HIGH);
    }
    if (tasterStatus = LOW) {
        digitalWrite(ledPin, LOW);
    }
}
```

Was ist hier falsch?

# Sensor abfragen

```
int ledPin = 13;      // Die LED
int buttonPin = 8;    // Der Taster
int tasterStatus;     // Der Tasterzustand


void loop() {
    tasterStatus = digitalRead(buttonPin);
    if (tasterStatus == HIGH) {
        digitalWrite(ledPin, HIGH);
    }
    if (tasterStatus == LOW) {
        digitalWrite(ledPin, LOW);
    }
}
```

==, nicht = !

# if und Klammern

- Folgt auf die if-Bedingung nur *eine* Anweisung, können die {…} weggelassen werden:

```
if (tasterStatus == HIGH) {
    digitalWrite(ledPin, HIGH);
}
```

ist dasselbe wie

```
if (tasterStatus == HIGH)
    digitalWrite(ledPin, HIGH);
```

# if und Klammern

- Beim Weglassen von {…} kann es zu subtilen Fehlern kommen:

```
if (tasterStatus == HIGH)
    digitalWrite(ledPin, HIGH);
    Serial.println("HIGH");

Serial.println(tasterStatus);
```

*Was ist hier falsch?*

# if und Klammern

- Beim Weglassen von {…} kann es zu subtilen Fehlern kommen:

```
if (tasterStatus == HIGH) {
    digitalWrite(ledPin, HIGH);
    Serial.println("HIGH");
}
Serial.println(tasterStatus);
```

- Merke: *Einrückung* ist für Menschen, *Klammern* sind für den Computer.

# Wenn dann sonst

- Mit if … else definiert man Anweisungen,, die ausgeführt werden, wenn die Bedingung *nicht* erfüllt ist

```
if (Bedingung) {
    Anweisungen...
}
if (!Bedingung) {
    Anweisungen...
}
```

⇔

```
if (Bedingung) {
    Anweisungen...
}
else {
    Anweisungen...
}
```

# Sonst wenn

- if … else lässt sich auch *verketten:*

```
if (Bedingung) {
    Anweisungen...
}
else if (Bedingung) {
    Anweisungen...
}
else {
    Anweisungen...
}
```

# Sensor abfragen

```
int ledPin = 13;      // Die LED
int buttonPin = 8;    // Der Taster
int tasterStatus;     // Der Tasterzustand


void loop() {
    tasterStatus = digitalRead(buttonPin);
    if (tasterStatus == HIGH) {
        digitalWrite(ledPin, HIGH);
    }
    if (tasterStatus == LOW) {
        digitalWrite(ledPin, LOW);
    }
}
```

# Sensor abfragen

```
int ledPin = 13;      // Die LED
int buttonPin = 8;    // Der Taster
int tasterStatus;     // Der Tasterzustand


void loop() {
    tasterStatus = digitalRead(buttonPin);
    if (tasterStatus == HIGH) {
        digitalWrite(ledPin, HIGH);
    }
    else {
       digitalWrite(ledPin, LOW);
    }
}
```

---

# Sensor abfragen

```
int ledPin = 13;      // Die LED
int buttonPin = 8;    // Der Taster
int tasterStatus;     // Der Tasterzustand


void loop() {
    tasterStatus = digitalRead(buttonPin);
    if (tasterStatus == HIGH)
        digitalWrite(ledPin, HIGH);
    else
        digitalWrite(ledPin, LOW);
}
```

---

# Sensor abfragen

```
int ledPin = 13;      // Die LED
int buttonPin = 8;    // Der Taster


void loop() {
    digitalWrite(ledPin, digitalRead(buttonPin));
}
```

*Tut dasselbe, ist aber nicht mehr gut lesbar*

# Ziel

Wenn Taste gedrückt,
soll LED *an/aus gehen*

# Ansatz

- Wir führen eine Variable *ledStatus* ein, die den aktuellen LED-Zustand darstellt und vom Taster umgeschaltet wird

# Zustand wechseln

```
int ledPin = 13;          // Pin LED
int buttonPin = 8;        // Pin Taster
int ledStatus = HIGH;     // Zustand LED

void setup() { ... }

void loop() {
    if (digitalRead(buttonPin) == HIGH) {
        if (ledStatus == HIGH)
            ledStatus = LOW;
        else
            ledStatus = HIGH;

        digitalWrite(ledPin, ledStatus);
        delay(200);
    }
}
```

# Zustand wechseln

```
int ledPin = 13;          // Pin LED
int buttonPin = 8;        // Pin Taster
int ledStatus = HIGH;     // Zustand LED

void setup() { ... }

void loop() {
    if (digitalRead(buttonPin) == HIGH) {
        ledStatus = !ledStatus;  // Kurzform
        digitalWrite(ledPin, ledStatus);
        delay(200);
    }
}
```

# Negation

- Wahrheitswerte in C:
  null (falsch) und nicht-null (wahr)
- ! ist die *Negation* (¬):
  - !0 == 1
  - !1 == 0
- HIGH und LOW haben die Werte 1 und 0

# Demo

- LED und Taster anschließen; blinken lassen.
- Problem: Wenn ich den Taster gedrückt halte, fängt es an zu blinken.

# Problem

Wenn Taste gedrückt,
blinkt LED

# Ansatz

- Die Variable *pushed* ist gesetzt, solange der Taster gedrückt ist
- Die Variable *ledStatus* wird nur geändert, wenn der Taster seinen Zustand ändert

```
int ledPin = 13;        // Pin LED
int buttonPin = 8;      // Pin Taster
int ledStatus = HIGH;   // Zustand LED
int pushed = 0;         // Zustand Taster

void setup() { ... }

void loop() {
    if (!pushed && digitalRead(buttonPin) == HIGH) {
        ledStatus = !ledStatus;
        pushed = 1;

        digitalWrite(ledPin, ledStatus);
        delay(200);
    }
    if (pushed && digitalRead(buttonPin) == LOW)
        pushed = 0;

}
```

# Logische Operatoren

- && ist ein logisches *und* (∧)

| && | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

- || ist ein logisches *oder* (∨)

| \|\| | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 1 |

---

# *Demo*

- LED und Taster anschließen; ein-/ausschalten

---

# Ziel

**Durch Tastendruck *Blinken* ein-/ausschalten**

# Blinken auf Wunsch

```
void loop() {
    if (!pushed && digitalRead(buttonPin) == HIGH) {
        ledStatus = !ledStatus;
        pushed = 1;
    }
    else if (pushed && digitalRead(buttonPin) == LOW)
        pushed = 0;

    if (ledStatus) {
        digitalWrite(ledPin, HIGH);
        delay(200);
        digitalWrite(ledPin, LOW);
        delay(200);
    }
}
```

---

*Demo*

- LED und Taster anschließen
- Problem: Nichts passiert!

---

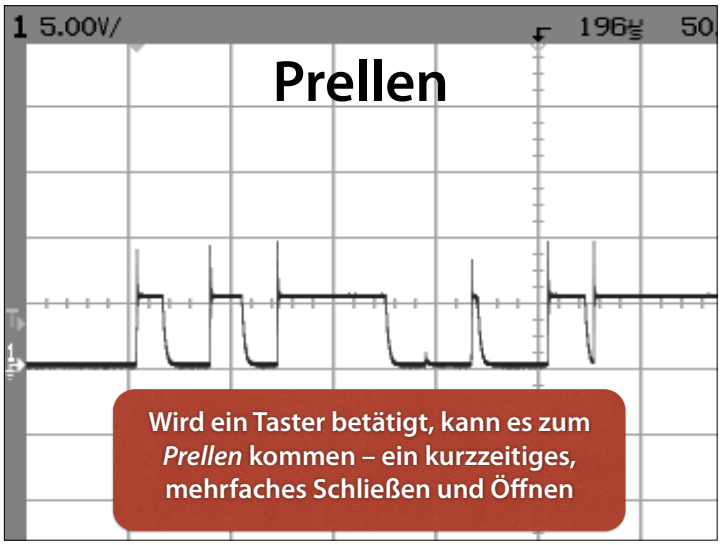
# Problem

Tastendrücke werden ignoriert

# Blinken auf Wunsch

```
void loop() {
    if (!pushed && digitalRead(buttonPin) == HIGH) {
        ledStatus = !ledStatus;
        pushed = 1;
    }
    else if (pushed && digitalRead(buttonPin) == LOW)
        pushed = 0;

    if (ledStatus) {
        digitalWrite(ledPin, HIGH);
        delay(200);
        digitalWrite(ledPin, LOW);
        delay(200);
    }
}
```

---




---

# Blinken auf Wunsch

```
void loop() {
    if (!pushed && digitalRead(buttonPin) == HIGH) {
        ledStatus = !ledStatus;
        pushed = 1;
        delay(20);  // warten bis Prellen zu Ende
    }
    else if (pushed && digitalRead(buttonPin) == LOW)
        pushed = 0;

    if (ledStatus) {
        digitalWrite(ledPin, HIGH);
        delay(200);
        digitalWrite(ledPin, LOW);
        delay(200);
    }
}
```

# Demo

- LED und Taster anschließen
- Problem: Tastendrücke werden ignoriert, wenn sie in die Blinkphase fallen

---

# Problem

Tastendrücke werden *manchmal* ignoriert

---

# Ziel

Taste *ununterbrochen* abfragen

# Zeit messen

- Während eines delay() werden Eingaben ignoriert
- Die Funktion millis() gibt die Millisekunden seit Programmstart zurück
- Wir können millis() nutzen, um die Zeit zu messen

# Blinken mit Millis

```
int ledPin = 13;      // Pin LED
int buttonPin = 8;    // Pin Taster

void setup() {
    pinMode(ledPin, OUTPUT);
    pinMode(buttonPin, INPUT);
}

void loop() {
    int ms = millis() % 1000;
    if (ms < 500)
        digitalWrite(ledPin, LOW);
    if (ms > 500)
        digitalWrite(ledPin, HIGH);
}
```

*Demo*

- LED und Taster anschließen
- Problem: Tastendrücke werden ignoriert, wenn sie in die Blinkphase fallen

# Taste mit Millis

```
void loop() {
    if (!pushed && digitalRead(buttonPin) == HIGH) {
        ledStatus = !ledStatus;
        pushed = 1;
        delay(20);
    }
    else if (pushed && digitalRead(buttonPin) == LOW)
        pushed = 0;

    ms = millis() % 1000;
    if (ms < 500)
        digitalWrite(ledPin, LOW);
    if (ms > 500)
        digitalWrite(ledPin, HIGH);
}
```

---

*Demo*

- Jetzt läuft alles gut

---

# Taste mit Millis

```
void loop() {
    if (!pushed && digitalRead(buttonPin) == HIGH) {
        ledStatus = !ledStatus;
        pushed = 1;
        delay(20);  Immer noch ein delay
    }
    else if (pushed && digitalRead(buttonPin) == LOW)
        pushed = 0;

    ms = millis() % 1000;
    if (ms < 500)
        digitalWrite(ledPin, LOW);
    if (ms > 500)
        digitalWrite(ledPin, HIGH);
}
```

# Entprellen mit Millis

```
int previousPush = 0;

void loop() {
  if (millis() - previousPush >= 20) {
    if (digitalRead(buttonPin) == HIGH) {
      previousPush = millis();
      ledStatus = !ledStatus;
      pushed = 1;
    }
    else if (pushed && digitalRead(buttonPin) == LOW)
      pushed = 0;
  }

  // Blinken...
}
```

# Datentypen in C

- long – mindestens 32 bits, [$-2^{31}$…$2^{31}$-1]
- int – 16 bis (meist) 32 bits, [$-2^{31}$…$2^{31}$–1]
- short – mindestens 16 bits, [$-2^{15}$…$2^{15}$–1]
- char – mind. 8 bits, meist [$-2^{7}$…$2^{7}$–1]
- Jeweils auch "unsigned", dann [0…$2^{bits}$–1]

*(long long >) long > int > short > char*

# Datentypen

- millis() hat den Typ "unsigned long" – ganzzahlige Werte im Bereich [0…$2^{32}$–1]
- Gewöhnliche ganze Zahlen ("int") haben einen Bereich [$-2^{n-1}$…$2^{n-1}$–1]
- *n* ist hierbei (je nach Gerät) 16 oder 32
- $2^{15}$ Millisekunden = 32767 ms = 32,7 s
  $2^{32}$ Millisekunden = 1193 h = 49,7 Tage

# Überlauf

- Versucht man, einen zu großen Wert in einem zu kleinen Datentyp zu speichern, kommt es zum *Überlauf*.
- Nur die letzten (Binär)Ziffern werden gespeichert
- Hat beliebige Werte zur Folge

# Entprellen mit Millis

```
int previousPush = 0;

void loop() {
  if (millis() - previousPush >= 20) {
    if (digitalRead(buttonPin) == HIGH) {
      previousPush = millis();
      ledStatus = !ledStatus;
      pushed = 1;
    }
    else if (pushed && digitalRead(buttonPin) == LOW)
      pushed = 0;
  }

  // Blinken...
}
```

# Entprellen mit Millis

```
unsigned long previousPush = 0;
```

Jetzt passt es

```
void loop() {
  if (millis() - previousPush >= 20) {
    if (digitalRead(buttonPin) == HIGH) {
      previousPush = millis();
      ledStatus = !ledStatus;
      pushed = 1;
    }
    else if (pushed && digitalRead(buttonPin) == LOW)
      pushed = 0;
  }

  // Blinken...
}
```

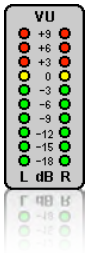
# Demo

- Und jetzt ist es perfekt. Wir sind fertig.

---

# Vorschau

- Felder
- Schleifen
- Pegelmessung




---

## Sensor abfragen

```
int ledPin = 13;   // Die LED
int buttonPin = 8; // Der Taster

void setup() {
    pinMode(ledPin, OUTPUT);
    pinMode(buttonPin, INPUT);
}

void loop() {
    if (digitalRead(buttonPin) == HIGH) {
        digitalWrite(ledPin, HIGH);
    }
    if (digitalRead(buttonPin) == LOW) {
        digitalWrite(ledPin, LOW);
    }
}
```

## Zuweisung

- Die Anweisung

  *name = wert*

  bewirkt, dass die Variable *name* den neuen Wert *wert* hat.
- Im weiteren Programmablauf liefert jeder spätere Zugriff auf die Variable den Wert *wert* (bis zur nächsten Zuweisung)

## Blinken mit Millis

```
    int ledPin = 13;     // Pin LED
    int buttonPin = 8;   // Pin Taster
    int ms = 0;          // Zeit

    void setup() {
        pinMode(ledPin, OUTPUT);
        pinMode(buttonPin, INPUT);
    }

    void loop() {
        int ms = millis() % 1000;
        if (ms < 500)
            digitalWrite(ledPin, LOW);
        if (ms > 500)
            digitalWrite(ledPin, HIGH);
    }
```

## Entprellen mit Millis

```
unsigned long previousPush = 0;

void loop() {
  if (millis() - previousPush >= 20) {
    if (digitalRead(buttonPin) == HIGH) {
      previousPush = millis();
      ledStatus = !ledStatus;
      pushed = 1;
    }
    else if (pushed && digitalRead(buttonPin) == LOW)
      pushed = 0;
  }

  // Blinken...
}
```

# Handouts

## Sensor abfragen

- Wir haben bereits digitalWrite() kennengelernt, die Daten digital ausgibt
- Neu: digitalRead() liest Daten digital *ein*

```
digitalRead(pin_number)
```

hat den Wert HIGH, wenn + am Pin anliegt; und LOW, wenn nicht.

## Sensor abfragen

```
int ledPin = 13;   // Die LED
int buttonPin = 8; // Der Taster

void setup() {
    pinMode(ledPin, OUTPUT);
    pinMode(buttonPin, INPUT);
}

void loop() {
    if (digitalRead(buttonPin) == HIGH) {
        digitalWrite(ledPin, HIGH);
    }
    if (digitalRead(buttonPin) == LOW) {
        digitalWrite(ledPin, LOW);
    }
}
```

# Zuweisung

- Die Anweisung

  *name = wert*

  bewirkt, dass die Variable *name* den neuen Wert *wert* hat.

- Im weiteren Programmablauf liefert jeder spätere Zugriff auf die Variable den Wert *wert (*bis zur nächsten Zuweisung)

# Zustand wechseln

```
int ledPin = 13;          // Pin LED
int buttonPin = 8;        // Pin Taster
int ledStatus = HIGH;   // Zustand LED

void setup() { ... }

void loop() {
    if (digitalRead(buttonPin) == HIGH) {
        if (ledStatus == HIGH)
            ledStatus = LOW;
        else
            ledStatus = HIGH;

        digitalWrite(ledPin, ledStatus);
        delay(200);
    }
}
```

# Blinken auf Wunsch

```
void loop() {
    if (!pushed && digitalRead(buttonPin) == HIGH) {
        ledStatus = !ledStatus;
        pushed = 1;
        delay(20);  // warten bis Prellen zu Ende
    }
    else if (pushed && digitalRead(buttonPin) == LOW)
        pushed = 0;

    if (ledStatus) {
        digitalWrite(ledPin, HIGH);
        delay(200);
        digitalWrite(ledPin, LOW);
        delay(200);
    }
}
```

# Blinken mit Millis

```
int ledPin = 13;      // Pin LED
int buttonPin = 8;    // Pin Taster

void setup() {
    pinMode(ledPin, OUTPUT);
    pinMode(buttonPin, INPUT);
}

void loop() {
    int ms = millis() % 1000;
    if (ms < 500)
        digitalWrite(ledPin, LOW);
    if (ms > 500)
        digitalWrite(ledPin, HIGH);
}
```

# Datentypen in C

- long – mindestens 32 bits, [$-2^{31}\ldots2^{31}-1$]
- int – 16 bis (meist) 32 bits, [$-2^{31}\ldots2^{31}-1$]
- short – mindestens 16 bits, [$-2^{15}\ldots2^{15}-1$]
- char – mindestens 8 bits, [$-2^{7}\ldots2^{7}-1$]
- Jeweils auch "unsigned", dann [$0\ldots2^{bits}-1$]

*(long long >) long > int > short > char*

# Entprellen mit Millis

```
unsigned long previousPush = 0;

void loop() {
  if (millis() - previousPush >= 20) {
    if (digitalRead(buttonPin) == HIGH) {
      previousPush = millis();
      ledStatus = !ledStatus;
      pushed = 1;
    }
    else if (pushed && digitalRead(buttonPin) == LOW)
      pushed = 0;
  }

  // Blinken...
}
```