Taster, INPUT_PULLUP und Entprellen

Einen Taster korrekt stecken, als Ereignis erkennen und ohne blockierende Pause entprellen.

MERKEN

Mit INPUT_PULLUP wird der Eingang intern nach 3,3 V gezogen. Der Taster verbindet ihn beim Druecken mit GND: nicht gedrueckt = HIGH, gedrueckt = LOW.

SCHALTUNG

```
GPIO21 ---- TASTER ---- GND

INPUT_PULLUP aktiv:

offen       HIGH (1)
gedrueckt   LOW  (0)

Vierbeiniger Taster:
Beine einer Seite sind intern
bereits miteinander verbunden.
```

ZUSTAND LESEN

```
const int BUTTON_PIN = 21;

void setup() {
  Serial.begin(115200);
  pinMode(BUTTON_PIN, INPUT_PULLUP);
}

void loop() {
  int state = digitalRead(BUTTON_PIN);
  Serial.println(state);
  delay(100);
}
```

ZUSTAND

- Ist der Taster jetzt gedrueckt?
- if (state == LOW)
- Aktion bleibt wahr, solange gehalten

FLANKE

- Hat er gerade gewechselt?
- HIGH -> LOW = Druck
- Ideal fuer Zaehler und Umschalten

EREIGNIS

- Einmal pro Druck reagieren
- Vorherigen Zustand speichern
- Nach Entprellung auswerten

Fehler	Ursache	Loesung
Immer LOW	Taster/Steckbrett falsch verbunden	Durchgang messen; Taster ueber Mittelsteg setzen
Immer HIGH	GND-Verbindung fehlt	Taster muss GPIO beim Druecken mit GND verbinden
Zufallswerte	Offener Eingang	INPUT_PULLUP oder externer Pull-Widerstand
Mehrfachzaehlung	Kontakt prellt	Flanke erkennen und zeitlich entprellen

PRAXIS

Vor dem Code den Taster mit dem Multimeter pruefen. Erst stabile 1/0-Werte im Serial Monitor, dann LED, Zaehler oder Menue anbinden.

Taster, INPUT_PULLUP und Entprellen

Einen Taster korrekt stecken, als Ereignis erkennen und ohne blockierende Pause entprellen.

NICHT-BLOCKIEREND ENTPRELLEN MIT millis()

```
const int BUTTON_PIN = 21;
const uint32_t DEBOUNCE_MS = 30;

int lastReading = HIGH;
int stableState = HIGH;
uint32_t lastChange = 0;

void setup() {
  Serial.begin(115200);
  pinMode(BUTTON_PIN, INPUT_PULLUP);
}

void loop() {
  int reading = digitalRead(BUTTON_PIN);

  if (reading != lastReading) {
    lastChange = millis();
  }

  if (millis() - lastChange >= DEBOUNCE_MS &&
      reading != stableState) {
    stableState = reading;
    if (stableState == LOW) {
      Serial.println("Tastendruck");
    }
  }

  lastReading = reading;
}
```

Die Differenzbildung mit uint32_t/unsigned long bleibt auch beim Ueberlauf von millis() korrekt.

Variable	Speichert	Warum?
reading	aktuellen Rohpegel	Kontakt darf waehrend des Prellens wechseln
lastReading	letzten Rohpegel	Start einer neuen Ruhezeit erkennen
stableState	bestaetigten Zustand	Nur stabile Aenderungen auswerten
lastChange	Zeit des letzten Wechsels	Nach 30 ms ohne Wechsel gilt der Pegel als stabil

KURZ

- Druckzeit speichern
- Beim Loslassen auswerten

LANG

- millis()-start
- Schwellwert z.B. 800 ms

TOGGLE

- Nur bei HIGH -> LOW
- bool state = !state

ACHTUNG

delay(50) kann fuer einen ersten Versuch reichen, blockiert aber in dieser Zeit Sensoren, Animationen, Kommunikation und weitere Taster.