## Opći spoj (za sve zadatke)

- **LED**: anoda → otpornik 220–330 Ω → **D8 (ili neki drugi izvod po želji)**, katoda → **GND**
- **Tipkalo**: jedan kraj → **D2 (ili neki drugi izvod po želji)**, drugi kraj → **GND**

U kodu koristimo: `pinMode(D2, INPUT_PULLUP);` (tipkalo aktivno na LOW, kada se pritisne)

## Zadatak 1: LED svijetli dok držimo tipkalo (osnovno)

**Opis:**

LED je ugašena. Kad pritisnemo tipkalo, LED svijetli. Kad pustimo, LED se gasi.

**Spoj:**

- LED + otpornik 220–330 Ω na **D8** (anoda na D8 preko otpornika, katoda na GND)
- Tipkalo na **D2** i **GND** (koristimo `INPUT_PULLUP`)

**Rješenje (kod)**

```
const int LED_PIN = 8;
const int BTN_PIN = 2;

void setup() {
  pinMode(LED_PIN, OUTPUT);
  pinMode(BTN_PIN, INPUT_PULLUP); // tipkalo na GND
}
void loop() {
  bool pressed = (digitalRead(BTN_PIN) == LOW); // aktivno na LOW
  digitalWrite(LED_PIN, pressed ? HIGH : LOW);
}
```

---

**Što je bouncing (odbijanje kontakta), zašto dolazi do toga i kako se rješava?**

Mehaničko tipkalo **nije idealni prekidač**.

Kad ga pritisnemo ili pustimo, njegovi metalni kontakti **ne spoje se trenutno i čisto**, nego:

- kratko **poskakuju** i dodiruju jedan drugog
- tijekom nekoliko **milisekundi** naprave niz vrlo brzih **uključenja i isključenja**

Mikrokontroler radi **puno brže** od mehanike tipkala i te kratke promjene vidi kao:

**više uzastopnih pritisaka**, iako je tipkalo pritisnuto samo jednom.

➔ To se naziva **contact bounce** (*odbijanje kontakta*).

## Kako se problem rješava? (debouncing)

Postoje **dva osnovna pristupa**:

### 1. Programsko rješenje (najčešće u nastavi)

Nakon promjene stanja tipkala:

- pričeka se **kratko vrijeme** (npr. 20–50 ms)
- tek ako je stanje i dalje isto, smatra se **valjanim pritiskom**

➔ Na taj se način ignoriraju brze, neželjene promjene.

**Prednosti:**

- nema dodatnih komponenti
- lako razumljivo
- idealno za učenje

### 2. Hardversko rješenje (rjeđe u osnovnoj nastavi)

- koristi se **RC sklop** (otpornik + kondenzator)
- ili **Schmitt-trigger** ulaz

➔ Električki "izravna" signal prije nego dođe do mikrokontrolera.

**Nedostatak:**

- dodatne komponente
- složenije za početnike

# Jedna rečenica za učenike

Tipkalo ne mijenja stanje trenutno, nego kratko "poskakuje", a mikrokontroler je toliko brz da to vidi kao više pritisaka, pa zato koristimo **debouncing**.

---

## Zadatak 2: Tipkalo pali/gasi LED (toggle) – s debounce

**Opis:**

Svaki pritisak tipkala promijeni stanje LED (ON→OFF, OFF→ON). Treba spriječiti "dvostruko okidanje" zbog odbijanja kontakta (debounce).

**Spoj:** isto kao Zadatak 1.

**Rješenje (kod)**

```
const int LED_PIN = 8;
const int BTN_PIN = 2;
bool ledState = false;
bool lastBtn = HIGH;
unsigned long lastDebounceTime = 0;
const unsigned long debounceMs = 30;

void setup() {
  pinMode(LED_PIN, OUTPUT);
  pinMode(BTN_PIN, INPUT_PULLUP);
  digitalWrite(LED_PIN, LOW);
}
void loop() {
  bool btn = digitalRead(BTN_PIN);
  // detekcija promjene
  if (btn != lastBtn) {
    lastDebounceTime = millis();
    lastBtn = btn;
  }
  // stabilno stanje nakon debounce vremena
  if ((millis() - lastDebounceTime) > debounceMs) {
    static bool lastStable = HIGH;
    if (btn != lastStable) {
      lastStable = btn;
      // okidaj na pritisak (HIGH -> LOW)
      if (lastStable == LOW) {
        ledState = !ledState;
        digitalWrite(LED_PIN, ledState ? HIGH : LOW);
      }
    }
  }
}
```

## Zadatak 3: Kratki pritisak = toggle, dugi pritisak (≥1 s) = LED blink (2 Hz)

**Opis:**

- Kratki pritisak: LED se pali/gasi (toggle)
- Dugi pritisak: LED prelazi u režim treptanja 2 Hz (svakih 250 ms mijenja stanje)
- Ponovni kratki pritisak izlazi iz blink moda i vraća toggle režim

**Spoj:** isto.

**Rješenje (kod)**

```
const int LED_PIN = 8;
const int BTN_PIN = 2;

bool ledState = false;
bool blinkMode = false;

bool lastBtn = HIGH;
unsigned long pressStart = 0;

unsigned long lastBlink = 0;
const unsigned long blinkInterval = 250; // 2 Hz (ON/OFF svakih 250ms)

void setup() {
  pinMode(LED_PIN, OUTPUT);
  pinMode(BTN_PIN, INPUT_PULLUP);
  digitalWrite(LED_PIN, LOW);
}

void loop() {
  bool btn = digitalRead(BTN_PIN);

  // pritisak detektiran
  if (lastBtn == HIGH && btn == LOW) {
    pressStart = millis();
  }

  // puštanje detektirano
  if (lastBtn == LOW && btn == HIGH) {
    unsigned long pressDur = millis() - pressStart;
```

```
    if (pressDur >= 1000) {
      blinkMode = true; // dugi pritisak -> blink
      lastBlink = millis();
    } else {
      // kratki pritisak
      if (blinkMode) {
        blinkMode = false;           // izlaz iz blink moda
        ledState = false;            // po želji reset stanja
        digitalWrite(LED_PIN, LOW);
      } else {
        ledState = !ledState;        // toggle
        digitalWrite(LED_PIN, ledState ? HIGH : LOW);
      }
    }
  }

  lastBtn = btn;

  // blink režim bez delay-a
  if (blinkMode) {
    if (millis() - lastBlink >= blinkInterval) {
      lastBlink = millis();
      ledState = !ledState;
      digitalWrite(LED_PIN, ledState ? HIGH : LOW);
    }
  }
}
```

---

## ZADATAK 3: Brojač pritisaka (kratki pritisak), ispis na Serial

**Opis:**

Svaki kratki pritisak tipkala povećava brojač. LED se upali na 200 ms kao potvrda. Broj se ispiše na Serial.

**Rješenje (kod)**

```
const int LED_PIN = 8;
const int BTN_PIN = 2;

unsigned long lastChange = 0;
const unsigned long debounceMs = 30;

bool lastRead = HIGH;
bool stableBtn = HIGH;

unsigned long ledUntil = 0;
unsigned long countPress = 0;

void setup() {
  pinMode(LED_PIN, OUTPUT);
  pinMode(BTN_PIN, INPUT_PULLUP);
  Serial.begin(9600);
}

void loop() {
  bool reading = digitalRead(BTN_PIN);

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

  if (millis() - lastChange > debounceMs) {
    if (reading != stableBtn) {
      stableBtn = reading;

      // okidaj na pritisak
      if (stableBtn == LOW) {
        countPress++;
        Serial.print("Broj pritisaka: ");
        Serial.println(countPress);
```

```
        ledUntil = millis() + 200;
      }
    }
  }

  digitalWrite(LED_PIN, (millis() < ledUntil) ? HIGH : LOW);
}
```

---

## ZADATAK 4: Dvostruki klik (double-click) = promjena režima

**Opis:**

- Jedan klik: LED toggle (ON/OFF)
- Dvostruki klik unutar 350 ms: LED ide u “blink” režim (2 Hz)
- Sljedeći jedan klik: izlaz iz blink režima i toggle

**Rješenje (kod)**

```
const int LED_PIN = 8;
const int BTN_PIN = 2;
const unsigned long debounceMs = 30;
const unsigned long doubleClickMs = 350;
const unsigned long blinkInterval = 250; // 2 Hz
bool lastRead = HIGH, stableBtn = HIGH;
unsigned long lastChange = 0;
bool ledState = false;
bool blinkMode = false;
unsigned long lastBlink = 0;
int clickCount = 0;
unsigned long firstClickTime = 0;

void setup() {
  pinMode(LED_PIN, OUTPUT);
  pinMode(BTN_PIN, INPUT_PULLUP);
}

void handleSingleClick() {
  if (blinkMode) {
    blinkMode = false;
    ledState = false;
    digitalWrite(LED_PIN, LOW);
  } else {
    ledState = !ledState;
    digitalWrite(LED_PIN, ledState ? HIGH : LOW);
  }
}

void handleDoubleClick() {
  blinkMode = true;
  lastBlink = millis();
```

```
}

void loop() {
  // debounce + edge
  bool reading = digitalRead(BTN_PIN);
  if (reading != lastRead) {
    lastRead = reading;
    lastChange = millis();
  }

  if (millis() - lastChange > debounceMs) {
    if (reading != stableBtn) {
      stableBtn = reading;

      if (stableBtn == LOW) { // click
        clickCount++;
        if (clickCount == 1) firstClickTime = millis();
      }
    }
  }

  // obrada single/double klika
  if (clickCount == 1 && (millis() - firstClickTime) > doubleClickMs) {
    handleSingleClick();
    clickCount = 0;
  } else if (clickCount >= 2) {
    handleDoubleClick();
    clickCount = 0;
  }

  // blink mode
  if (blinkMode) {
    if (millis() - lastBlink >= blinkInterval) {
      lastBlink = millis();
      ledState = !ledState;
      digitalWrite(LED_PIN, ledState ? HIGH : LOW);
    }
  }
}
```

---

## ZADATAK 5: Morse s tipkalom (kratko = točka, dugo = crta), LED pokazuje unos

**Opis:**

- Držiš tipkalo: mjeri trajanje
- Pustiš tipkalo:
  - < 300 ms = **točka**
  - ≥ 300 ms = **crta**
- LED potvrđuje:
  - točka: kratko bljesne 100 ms
  - crta: duže bljesne 300 ms
- Ispiši "." ili "-" na Serial

**Rješenje (kod)**

```
const int LED_PIN = 8;
const int BTN_PIN = 2;

const unsigned long debounceMs = 30;
const unsigned long dotThreshold = 300;

bool lastRead = HIGH, stableBtn = HIGH;
unsigned long lastChange = 0;

unsigned long pressStart = 0;
unsigned long ledUntil = 0;

void setup() {
  pinMode(LED_PIN, OUTPUT);
  pinMode(BTN_PIN, INPUT_PULLUP);
  Serial.begin(9600);
}

void loop() {
  bool reading = digitalRead(BTN_PIN);

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

  if (millis() - lastChange > debounceMs) {
```

```
    if (reading != stableBtn) {
      stableBtn = reading;

      if (stableBtn == LOW) {
        pressStart = millis();
      } else { // released
        unsigned long dur = millis() - pressStart;

        if (dur < dotThreshold) {
          Serial.print(".");
          ledUntil = millis() + 100;
        } else {
          Serial.print("-");
          ledUntil = millis() + 300;
        }
      }
    }
  }

  digitalWrite(LED_PIN, (millis() < ledUntil) ? HIGH : LOW);
}
```

---

## ZADATAK 6: “Sigurnosni start” – drži tipkalo 2 s da se LED uopće aktivira

### Opis:

Dok ne držiš tipkalo **2 sekunde u komadu**, LED ignorira klikove. Nakon “otključavanja” radi normalni toggle.

### Rješenje (kod)

```
const int LED_PIN = 8;
const int BTN_PIN = 2;

const unsigned long debounceMs = 30;
const unsigned long unlockMs = 2000;

bool lastRead = HIGH, stableBtn = HIGH;
unsigned long lastChange = 0;

bool unlocked = false;
unsigned long pressStart = 0;

bool ledState = false;

void setup() {
  pinMode(LED_PIN, OUTPUT);
  pinMode(BTN_PIN, INPUT_PULLUP);
}

void loop() {
  bool reading = digitalRead(BTN_PIN);

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

  if (millis() - lastChange > debounceMs) {
    if (reading != stableBtn) {
      stableBtn = reading;

      if (stableBtn == LOW) {
        pressStart = millis();
      } else { // release
```

```
        if (!unlocked) {
          // ništa
        } else {
          ledState = !ledState;
          digitalWrite(LED_PIN, ledState ? HIGH : LOW);
        }
      }
    }
  }

  // unlock check (dok je tipkalo stisnuto)
  if (!unlocked && stableBtn == LOW) {
    if (millis() - pressStart >= unlockMs) {
      unlocked = true;
      // signal: tri brza bljeska
      // (bez delay-a: jednostavno kratko upali pa ugasi nakon 150ms, ponovi u idućoj verziji ako želiš)
      ledState = true;
      digitalWrite(LED_PIN, HIGH);
    }
  }

  // nakon unlock-a vrati LED u OFF (da se vidi da sad radi toggle)
  if (unlocked && ledState == true && stableBtn == HIGH) {
    ledState = false;
    digitalWrite(LED_PIN, LOW);
  }
}
```

---