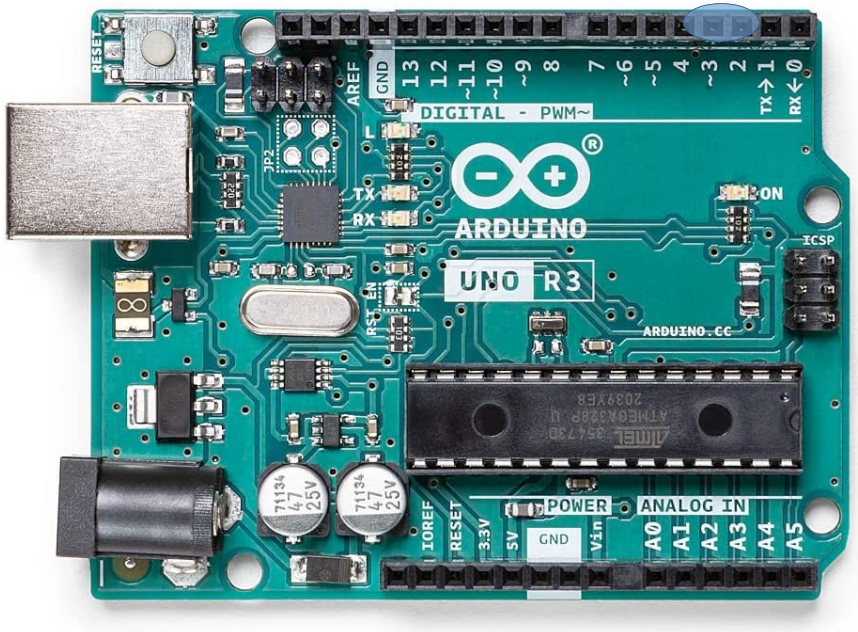
Problem:

Du möchtest in deinem Programm **jederzeit** auf ein Signal reagieren können.

```
attachInterrupt(PIN, ISR, mode)
```

Ausführen einer Funktion, wenn sich der Status eines der Pins 2 oder 3 ändert



Beispiel:

```
int rot = 12; int taster = 2;
void setup() {
  for (int i = 4; i < 13; i++) {
    pinMode(i, OUTPUT);
  }
  pinMode(2, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(taster),roteLed ,FALLING);
}
void loop() {
  for (int i =  4; i < 12; i++) {
    digitalWrite(i, HIGH);
    delay(500);
    digitalWrite(i, LOW);
  }
  digitalWrite(rot, LOW);
}
void roteLed(){
  digitalWrite(rot,HIGH);
}
```

Hier wird festgelegt, welches Unterprogramm aufgerufen wird, wenn der Interrupt auslöst.

Festlegung des Interrupt-Pins

Hier wird der Interrupt aktiviert

Hier wird festgelegt, auf welches Signal der Interrupt reagieren soll:
FALLING: das Signal wechselt von 1 auf 0.
RISING: das Signal wechselt von 0 auf 1.
LOW: das Signal ist 0.
CHANGE: das Signal wechselt.

Achtung! ISR-Regeln!

- ISR möglichst kurz - keine serielle Konsole in ISRs!
- ISR können keine Argumente bekommen oder Rückgabewerte zurückgeben. Ausnahmsweise globale Variablen benutzen diese müssen als **volatile** deklariert werden.
- Einige Funktionen verhalten sich in ISRs anders als gewohnt oder funktionieren gar nicht:
  - **millis()** verlässt sich zum Zählen auf Interrupts, wird also in einer Interrupt Service Routine niemals hochzählen.
  - **delay()** benutzt ebenfalls Interrupts und wird deshalb gar nicht in einer Interrupt Service Routine funktionieren - sollte dort aber aucch n iemals benutzt werden, weil, die ISR ja schnell abgearbeitet werden sollte!
  - **micros()** wird anfangs gut funktionieren, sich aber nach etwa 1 bis 2 ms unvorhersehbar verhalten - möglichst nicht benutzen.
  - **delayMicroseconds()** benutzt keine Zähler und wird deshalb normal funktionieren.

Beispiel 2

```
int taster = 2;
volatile int zustand = 0;

void setup() {
  for (int i = 4; i < 13; i++) {
    pinMode(i, OUTPUT);
  }
  pinMode(2, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(taster),meineISR ,FALLING);
}
void loop() {
  for (int i =  4; i < 12; i++) {
    digitalWrite(i, HIGH);
    delay(500);
    digitalWrite(i, LOW);

    if(zustand == 1){
       machIrgenwas();
       zustand = 0;  //zurücksetzen der Variablen zustand auf 0
    }
  }
}
void meineISR(){zustand = 1;}

void machIrgenwas(){…} // ist noch leer, ergänze!
```

Variablen im Interrupt müssen als volatile deklariert werden!