

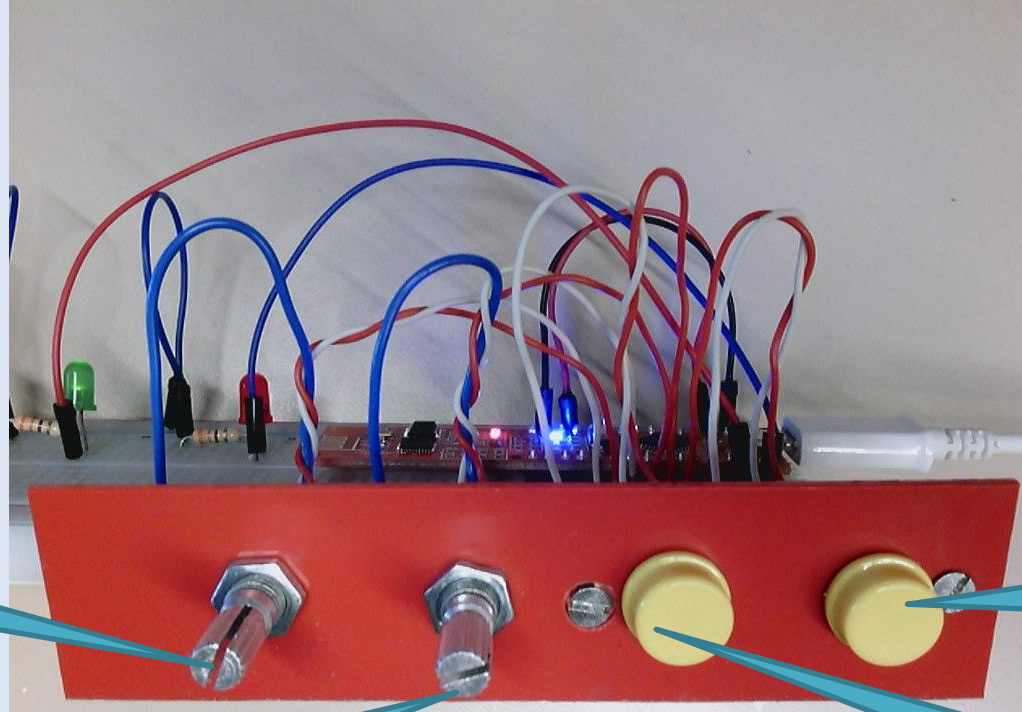
Rotary Encoder (RE)

- Rotary Encoder & Taster
- Kontakte & Takte
- Takt 1-4
- Schritt & Takte
- Schaltplan
- Takte lesen (Polling)
- Polling oder Interrupt
- Takte lesen (Interrupt)
- Interrupt?
- Decoding Tabelle
- Algorithmus
- Takte lesen & zählen I & II
- Schritte zählen
- Ben Baxton state-machine
- Library Interrupt
- Sketch 2 RE & 2 Taster
- Download

Rotary Encoder & Taster

(Sketch am Ende der Präsentation)

Die Variablen „countleft“ und „countright“ zählen die Schritte.



Die Variablen „switchleft“ und „switchright“ sind „HIGH“ beim Betätigen der Taster.

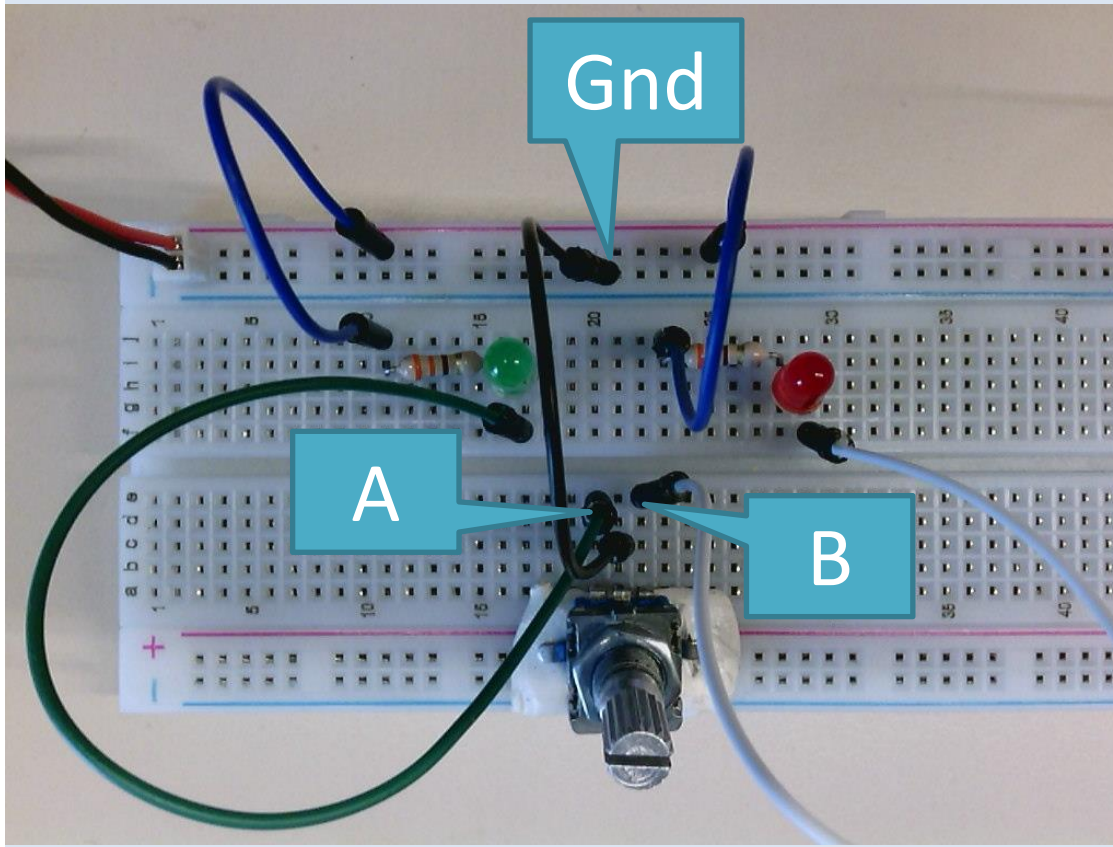
countleft

countright

switchleft

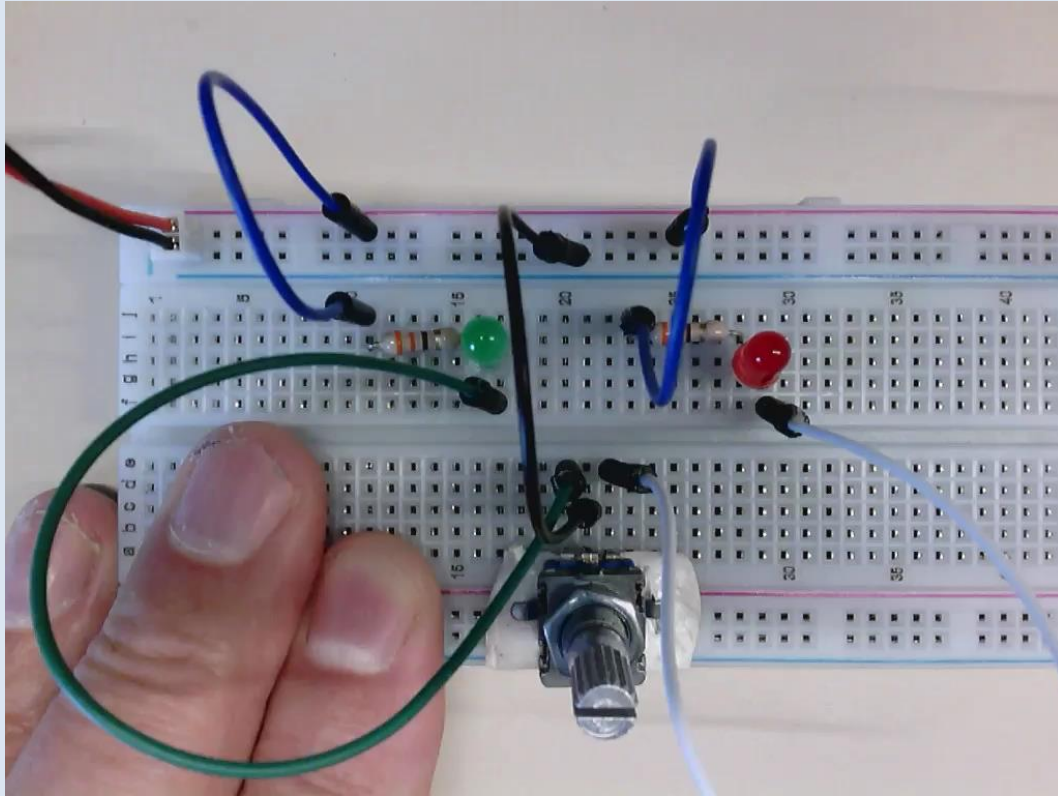
switchright

Kontakte & Takte



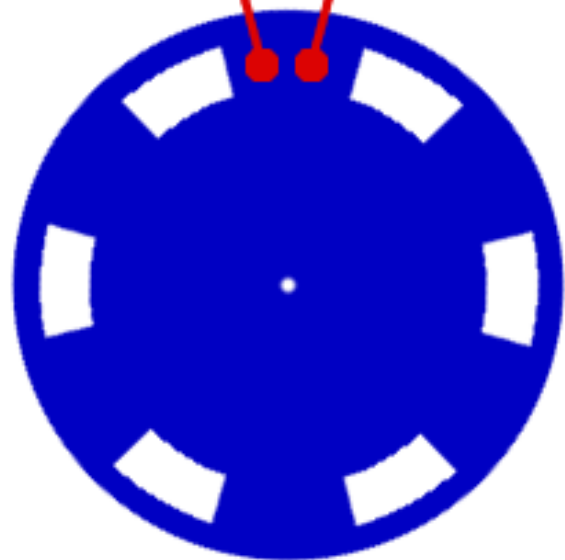
Der Rotary Encoder (RE) hat 20 Schritte für eine Umdrehung. Eine Umdrehung besteht aus 4 Takten. Mit welchem Takt beginnt ein Schritt?

Takte Video



Takt 1

Pin1 Pin2



Pin1



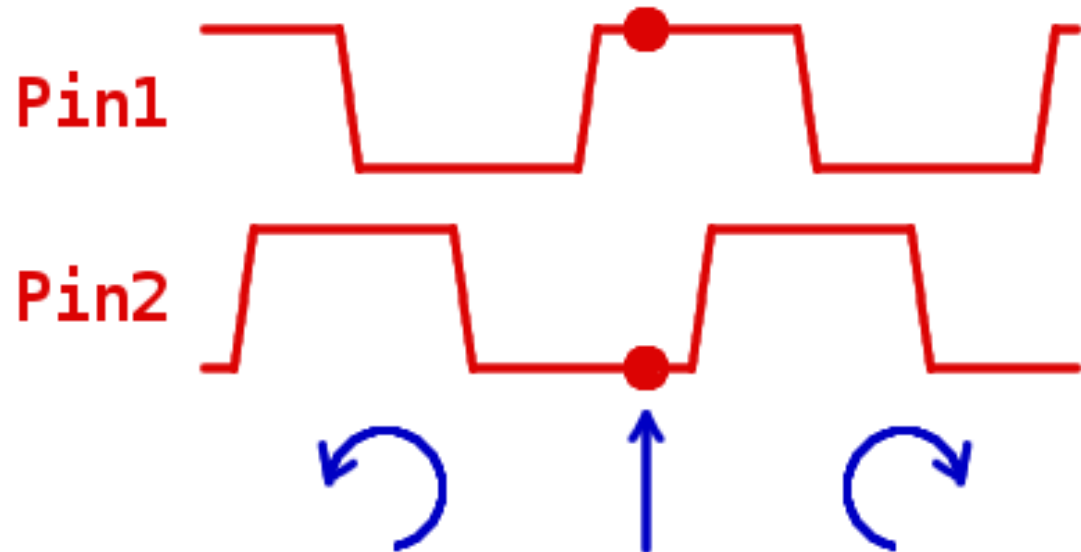
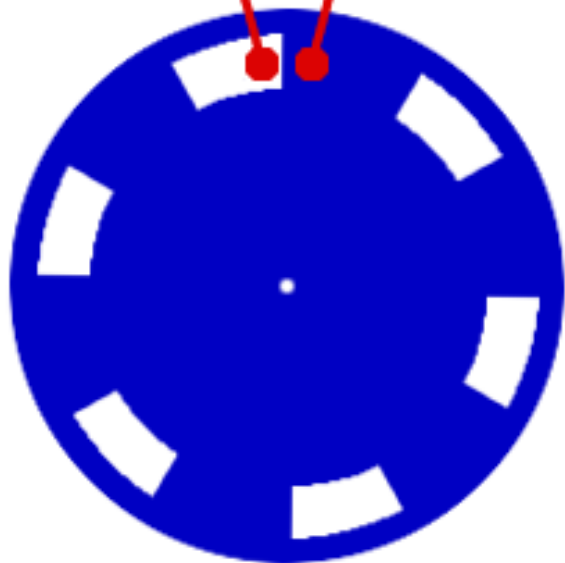
Pin2



<-- Counterclockwise 0 Clockwise -->

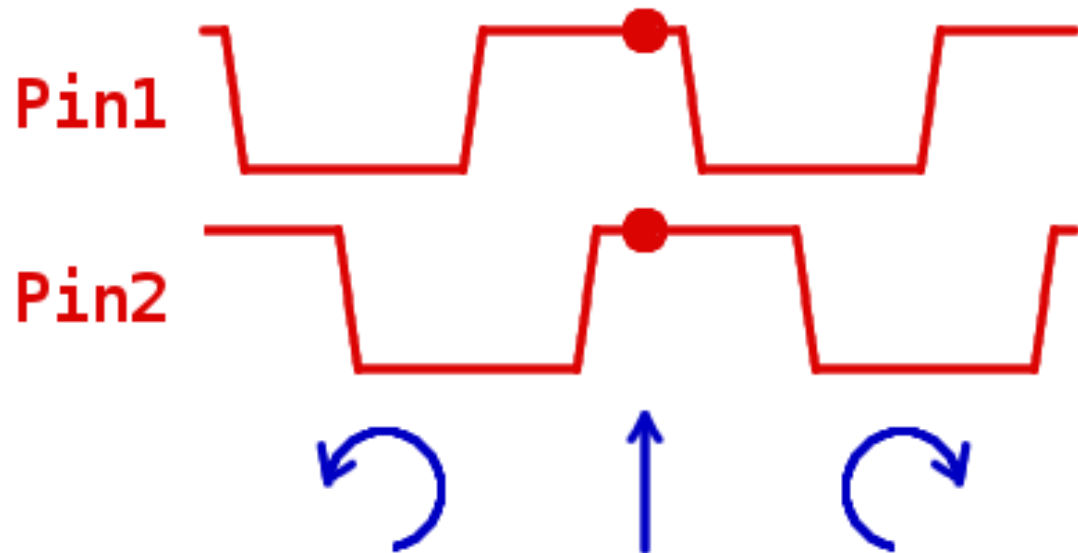
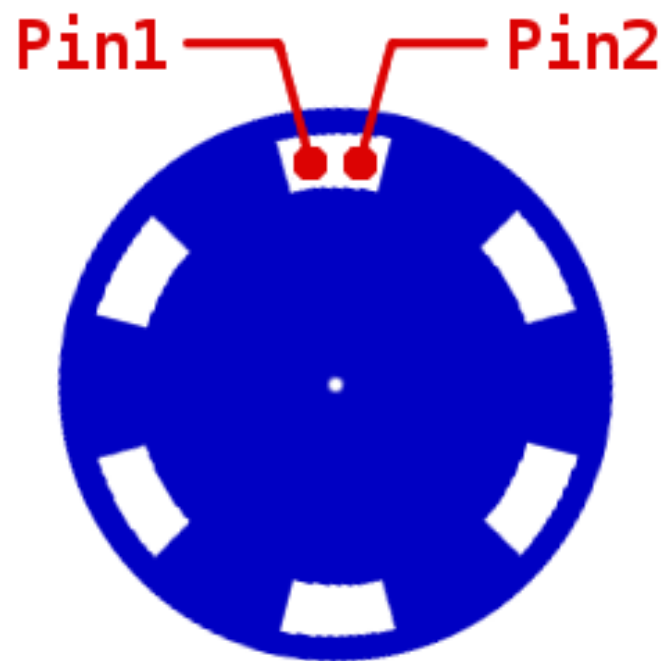
Takt 2

Pin1 Pin2



<-- Counterclockwise 1 Clockwise -->

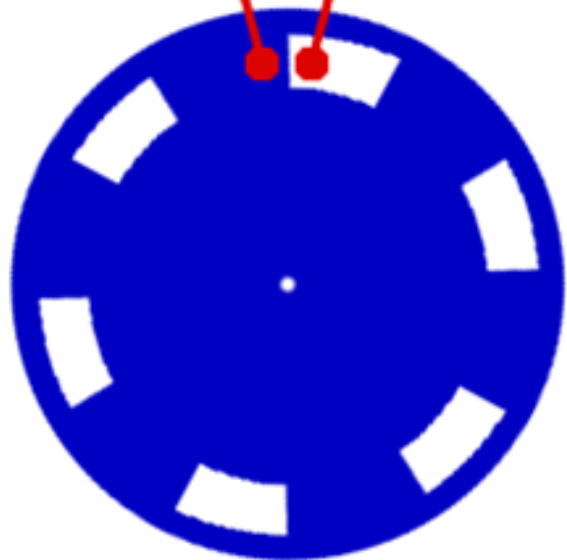
Takt 3



<-- Counterclockwise 2 Clockwise -->

Takt 4

Pin1 Pin2



Pin1



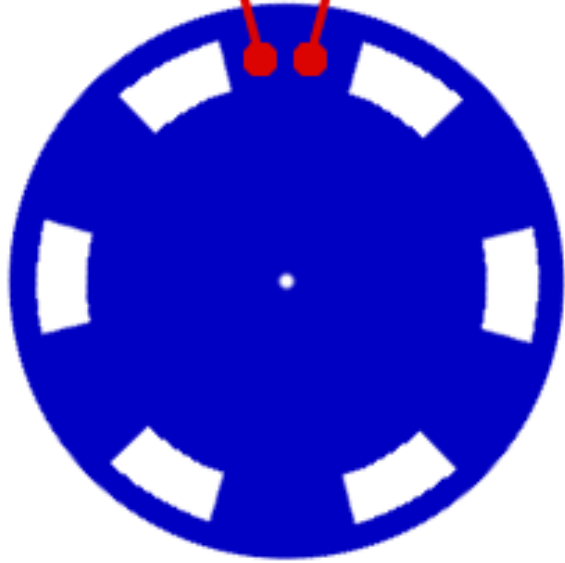
Pin2



<-- Counterclockwise 3 Clockwise -->

Takt 1

Pin1 Pin2



Pin1



Pin2



<-- Counterclockwise 4 Clockwise -->

Schritt & Takte


Ein einzelner Schritt erzeugt 4 Takte (die man zählen könnte).
Es sollen aber nur die Schritte gezählt werden!

Uhrzeigersinn	Grün A	Rot B	Gegen Uhrzeigersinn	Grün A	Rot B
Takt 1	0	0	Takt 1	0	0
Takt 2	1	0	Takt 4	0	1
Takt 3	1	1	Takt 3	1	1
Takt 4	0	1	Takt 2	1	0
Takt 1	0	0	Takt 1	0	0

1 Umdrehung Besteht aus 20 Schritten

1 Schritt 4 Takte

Schaltplan Arduino UNO

Hardware Pin	2		3	11	12
Rotary Encoder	A		B	LED grün	LED rot
Drehrichtung				CW	CCW
Hardware Interrupt	INT0		INT1		
	Masse (Gnd) auf mittleren Kontakt.				
	Im Uhrzeigersinn:	CW	(clockwise)		
	Gegen Uhrzeigersinn:	CCW	(counterclockwise)		
Hardware Pin 2 & 3:	Eingerichtet mit Pullup-Widerständen (20 kΩ), daher Status HIGH!				
	Kontakte 2 & 3 ziehen den Status auf LOW!				
Sketch	Zur Übereinstimmung mit Takt-Logik:		A=!A; B=!B		

Takte lesen (Polling)

```
/* Hands on Arduino Incremental Rotary Encoders
 * by EBW Enno Klatt 02/2019
 * Primary sketch: Determining the sequence
 * EBW_Enno_00.ino
 * Based on Robert Paz
 * https://www.youtube.com/watch?v=S1JJc8YAJqQ
 */
#define intPin0 2 // Arduino UNO Hardware-Interrupt INT0
#define intPin1 3 // Arduino UNO Hardware-Interrupt INT1

unsigned long count = 0;
byte A, B; // value of Pin 2 and Pin 3 to be read
byte Ap=B00, Bp=B00; // initialize previous state

void setup() {
  Serial.begin(57600);
  Serial.println("EBW_Enno_00.ino");
  pinMode(11, OUTPUT); // show the movement of the Rotary Encoder
  pinMode(12, OUTPUT);

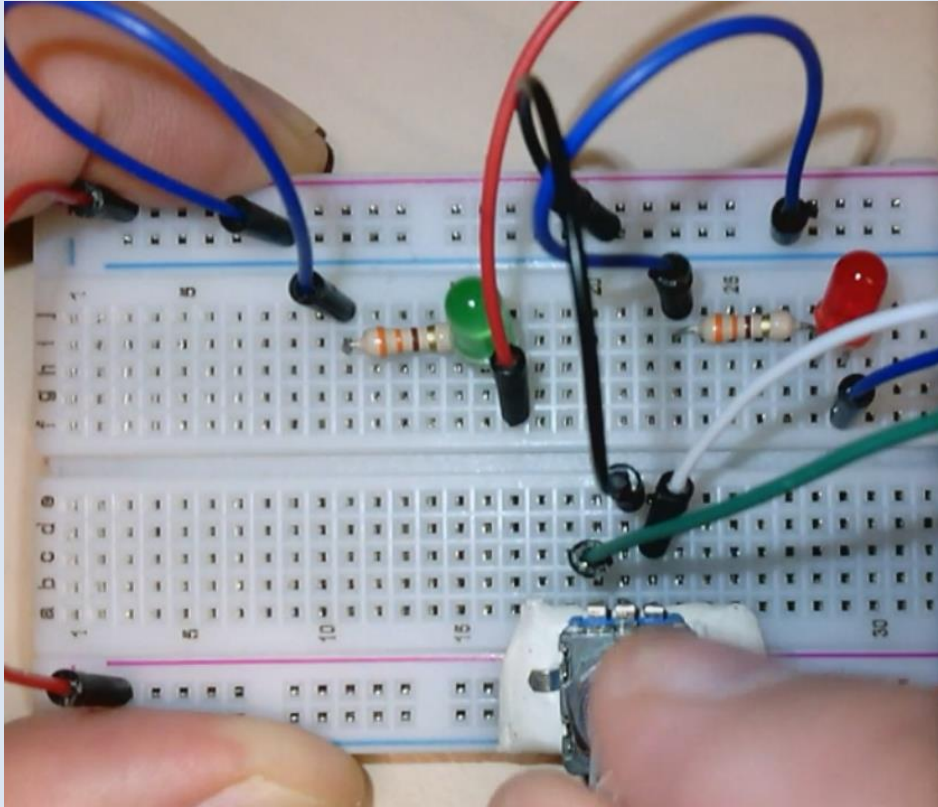
  pinMode(intPin0, INPUT_PULLUP);
  pinMode(intPin1, INPUT_PULLUP);
}

void loop() {
  // code which shall be executed over and over
  // in this case polling Pins 2 and 3
  A = digitalRead(intPin0); // reading Pin 2
  B = digitalRead(intPin1); // reading Pin 3
  A=!A; // we get LOW, but we want HIGH and vice versa
  B=!B; // we get LOW, but we want HIGH and vice versa
```

```
// catch changing of state A or state B
if (A^Ap || B^Bp) { // exclusive or "^"
  // *****
  // code to be executed after state has changed
  count++;
  Serial.print("count=");Serial.print(count);
  Serial.print(" A=");Serial.print(A);
  Serial.print(" B=");Serial.print(B);
  Serial.print(" AB=");
  // a little bit of mystic code to show state A and B together
  binDisplay(2, A<<1 | B);
  // end code to be executed after state has changed
  // *****
}
Ap = A; Bp = B;
// debugging LEDs
if (A) digitalWrite(11,HIGH);
else digitalWrite(11,LOW);
if (B) digitalWrite(12,HIGH);
else digitalWrite(12,LOW);
}

// displaying binary number in Monitor for debug
void binDisplay(int size, unsigned long int value){
  for (int i = size-1; i >= 0; i--) {
    Serial.print((value >> i) & 1);}
  Serial.println();
}
```

Polling Video



```
COM3  
  
count=1 A=1 B=0 AB=10  
count=2 A=1 B=1 AB=11  
count=3 A=0 B=1 AB=01  
count=4 A=0 B=0 AB=00  
count=5 A=1 B=0 AB=10  
count=6 A=1 B=1 AB=11  
count=7 A=0 B=1 AB=01  
count=8 A=0 B=0 AB=00  
count=9 A=0 B=1 AB=01  
count=10 A=0 B=0 AB=00
```

Taktfolge i.O.

Bouncing?

Polling oder Interrupt

Polling	Zustand von A und B in der Funktion loop() abfragen.
Sketch	Durch die stetige Abfrage der Zustände von A und B kann es theoretisch vorkommen, dass die Abfrage nicht zum tatsächlichen Zeitpunkt der Betätigung erfolgt. Die Anzahl der Programmschritte beeinflusst diesen Effekt.
Nachteile	Es erfolgt eine Abfrage, auch wenn sich die Zustände nicht geändert haben. Es erfolgt keine Abfrage, auch wenn sich die Zustände geändert haben.
Vorteile	Fehlereffekte?
Abhilfe	Anwenden der Interrupt-Technik, um eine Zustandsänderung sofort zu erkennen und eine Aktion auszulösen

Takte lesen (Interrupt)

```
/* Hands on Arduino Incremental Rotary Encoders
 * by EBW Enno Klatt 02/2019
 * First sketch: Example for Interrupt
 * EBW_Enno_01.ino
 * Based on Robert Paz
 * https://www.youtube.com/watch?v=S1JJc8YAJqQ
 */
#define intPin0 2 // Arduino UNO Hardware-Interrupt INT0
#define intPin1 3 // Arduino UNO Hardware-Interrupt INT1

long count=0, countSave; // countner
byte A, B;                // value of intPin0 / INT0
                          // value of intPin1 / INT1

boolean intA = false; // initialize interrupt-state channel A
boolean intB = false; // initialize interrupt-state channel B
byte Ap=B00, Bp=B00;  // initialize previous state

void setup() {
  Serial.begin(57600);
  Serial.println("EBW_Enno_01.ino");
  // LEDs for debugging
  pinMode(11, OUTPUT); // show the movement of the Rotary Encoder
  pinMode(12, OUTPUT);

  pinMode(intPin0, INPUT_PULLUP);
  pinMode(intPin1, INPUT_PULLUP);

  // attach Interrupt-Capability
  attachInterrupt(0, Achange, CHANGE);
  attachInterrupt(1, Bchange, CHANGE);
}
```

```
void loop() {
  // *****
  // code to be executed after an Interrupt
  if (intA || intB){
    Serial.print("count=");Serial.print(count);
    Serial.print(" A=");Serial.print(A);
    Serial.print(" B=");Serial.print(B);
    Serial.print(" AB=");
    intA = false;
    intB = false;
    // a little bit of mystic code
    binDisplay(2, A<<1 | B);
  // end of code to be executed after an Interrupt
  // *****
  }
  // debugging LEDs
  if (A)  digitalWrite(11,HIGH);
  else   digitalWrite(11,LOW);
  if (B)  digitalWrite(12,HIGH);
  else   digitalWrite(12,LOW);
}
```

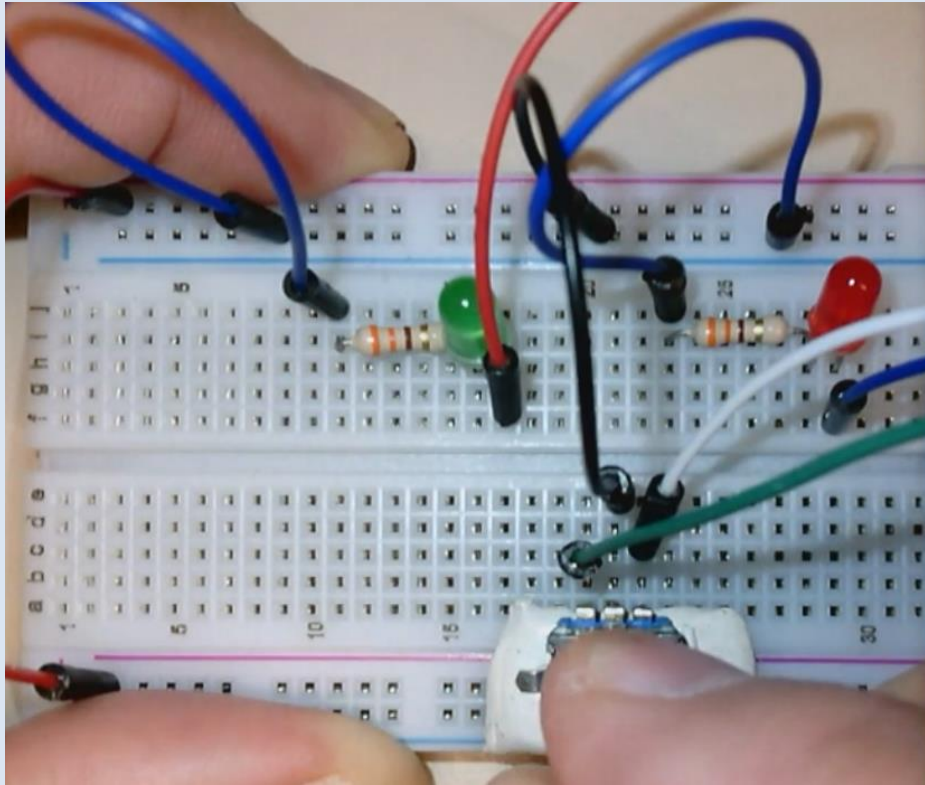
Takte lesen (Interrupt)

```
// Interrupt Service Routine Channel A
void Achange() {
  A = digitalRead(intPin0); // reading Pin 2
  B = digitalRead(intPin1); // reading Pin 3
  A=!A;
  B=!B;
  if (A^Ap || B^Bp) count++;
  Ap = A; Bp = B;
  intA = true;
}
```

```
// Interrupt Service Routine Channel B
void Bchange() {
  A = digitalRead(intPin0); // reading Pin 2
  B = digitalRead(intPin1); // reading Pin 3
  A=!A;
  B=!B;
  if (A^Ap || B^Bp) count++;
  Ap = A; Bp = B;
  intB = true;
}

// displaying binary number in monitor for debug
void binDisplay(int size, unsigned long int value){
  for (int i = size-1; i >= 0; i--) {
    Serial.print((value >> i) & 1);
  }
  Serial.println();
}
```


Video Interrupt



```
COM3  
|  
count=1 A=1 B=0 AB=10  
count=2 A=1 B=1 AB=11  
count=3 A=0 B=1 AB=01  
count=4 A=0 B=0 AB=00  
count=5 A=1 B=0 AB=10  
count=6 A=1 B=1 AB=11  
count=6 A=0 B=1 AB=01  
count=7 A=0 B=1 AB=01  
count=13 A=0 B=0 AB=00  
count=18 A=0 B=0 AB=10  
count=27 A=1 B=0 AB=10  
count=28 A=1 B=1 AB=11  
count=29 A=0 B=1 AB=01  
count=30 A=0 B=0 AB=00
```

Taktfolge i.O.

Bouncing!

Interrupt?

Vorteile

Alle Takte werden erkannt.

Nachteile

Auch fehlerhafte Takte, Bouncing-Effekte, werden erkannt.

Abhilfe

Hardwarelösung: Kondensatoren

Ist eine Softwarelösung die bessere Alternative?

Abhilfe?

Bei welcher Taktfolge soll der Schritt gezählt werden?

Kann die Abfolge der Takte kontrolliert werden?

Decoding Tabelle

	status vorher	1	2	3	4										
status aktuell	A & B	00	10	11	01		A	0	1	1	0	0	1	1	0
1	00		<i>dec</i>		<i>inc</i>										
2	10	<i>inc</i>		<i>dec</i>			B	0	0	1	1	0	0	1	1
3	11		<i>inc</i>		<i>dec</i>										
4	01	<i>dec</i>		<i>inc</i>			status	1	2	3	4	1	2	3	4

Algorithmus

Ziel: Zählen der Schritte.
Unterscheiden zwischen Uhrzeigersinn (CW) und gegen Uhrzeigersinn (CCW).

Ausgangszustand? A auf LOW und B auf LOW im Sketch realisieren.

Drehrichtung?
Bei CW folgt auf A=LOW und B=LOW: A=HIGH und B=LOW
Bei CCW folgt auf A=LOW und B=LOW: A=LOW und B=HIGH
(weitere Kombinationen sind möglich)

Algorithmus „Letzten Zustand“ von A und B merken: hier A=LOW und B=LOW

Folgt darauf A=HIGH und B=LOW liegt CW vor.

oder

Folgt darauf A=LOW und B=HIGH liegt CCW vor.

Durch Vergleich des letzten mit dem aktuellen Zustand kann die Drehrichtung entschieden werden.

Takte lesen & zählen I

```
#define intPin0 2 // Arduino UNO Hardware-Interrupt INT0 / A
#define intPin1 3 // Arduino UNO Hardware-Interrupt INT1 / B

long count=0, countSave; // countner
byte A, B;                // value of intPin0 / INT0
                          // value of intPin1 / INT1
byte state, statep;      // state: possible 4 actual state(s)
                          // statep: previous state

void setup() {
  Serial.begin(57600);
  Serial.println("EBW_Enno_02.ino");
  // LEDs for debugging
  pinMode(11, OUTPUT);    // show the movement of the Rotary Encoder
  pinMode(12, OUTPUT);

  pinMode(intPin0, INPUT_PULLUP);
  pinMode(intPin1, INPUT_PULLUP);

  // attach Interrupt-Capability
  attachInterrupt(0, Achange, CHANGE);
  attachInterrupt(1, Bchange, CHANGE);

  // read the initial value of A & B
  A = digitalRead(intPin0);A=!A;
  B = digitalRead(intPin1);B=!B;

  // set initial state value of A & B
  if ((A==LOW)  && (B==LOW)) statep = 1;
  if ((A==HIGH) && (B==LOW)) statep = 2;
  if ((A==HIGH) && (B==HIGH)) statep = 3;
  if ((A==LOW)  && (B==HIGH)) statep = 4;
}
```

```
void loop() {
  // show count only when changed
  if (countSave != count){
    countSave = count;
    Serial.print("count=");Serial.print(count);
    Serial.print(" A=");Serial.print(A);
    Serial.print(" B=");Serial.print(B);
    Serial.print(" AB=");

    // a little bit of mystic code
    binDisplay(2, A<<1 | B);
    // debugging LEDs
    if (A) digitalWrite(11,HIGH);
    else  digitalWrite(11,LOW);
    if (B) digitalWrite(12,HIGH);
    else  digitalWrite(12,LOW);
  }
  // end of show count only when changed
}

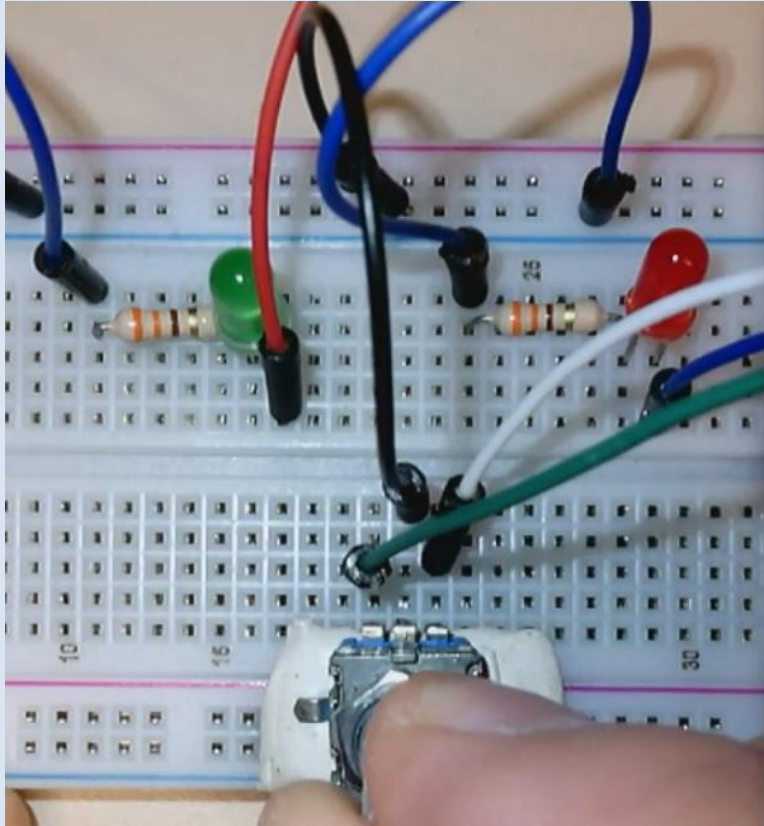
// displaying binary number in monitor for debug
void binDisplay(int size, unsigned long int value){
  for (int i = size-1; i >= 0; i--) {
    Serial.print((value >> i) & 1);
  }
  Serial.println();
}
```

Takte lesen & zählen II

```
void Achange() {
  A = digitalRead(intPin0);
  B = digitalRead(intPin1);
  A=!A;
  B=!B;
  // determine state value
  if ((A==LOW) && (B==LOW)) state = 1;
  if ((A==HIGH) && (B==LOW)) state = 2;
  if ((A==HIGH) && (B==HIGH)) state = 3;
  if ((A==LOW) && (B==HIGH)) state = 4;
  switch (state){
    case 1: {
      if (statep == 2) count--;
      if (statep == 4) count++;
      break;
    }
    case 2: {
      if (statep == 1) count++;
      if (statep == 3) count--;
      break;
    }
    case 3: {
      if (statep == 2) count++;
      if (statep == 4) count--;
      break;
    }
    case 4: {
      if (statep == 1) count--;
      if (statep == 3) count++;
      break;
    }
  }
  statep = state;
}
```

```
void Bchange() {
  A = digitalRead(intPin0);
  B = digitalRead(intPin1);
  A=!A;
  B=!B;
  // determine state value
  if ((A==LOW) && (B==LOW)) state = 1;
  if ((A==HIGH) && (B==LOW)) state = 2;
  if ((A==HIGH) && (B==HIGH)) state = 3;
  if ((A==LOW) && (B==HIGH)) state = 4;
  switch (state){
    case 1: {
      if (statep == 2) count--;
      if (statep == 4) count++;
      break;
    }
    case 2: {
      if (statep == 1) count++;
      if (statep == 3) count--;
      break;
    }
    case 3: {
      if (statep == 2) count++;
      if (statep == 4) count--;
      break;
    }
    case 4: {
      if (statep == 1) count--;
      if (statep == 3) count++;
      break;
    }
  }
  statep = state;
}
```

Video Takte zählen



```
COM3  
  
count=1 A=1 B=0 AB=10  
count=2 A=1 B=1 AB=11  
count=3 A=0 B=1 AB=01  
count=4 A=0 B=0 AB=00  
count=5 A=1 B=0 AB=10  
count=6 A=1 B=1 AB=11  
count=7 A=0 B=1 AB=01  
count=8 A=0 B=0 AB=00  
count=9 A=1 B=0 AB=10  
count=8 A=0 B=0 AB=00  
count=9 A=0 B=0 AB=00  
count=8 A=1 B=0 AB=10  
count=9 A=1 B=0 AB=10  
count=10 A=1 B=1 AB=11  
count=11 A=0 B=1 AB=01  
count=12 A=0 B=0 AB=00
```

4 Takte werden erkannt

Bouncing

Schritte zählen

Ziel:	Zählen der Schritte. Unterscheiden zwischen Uhrzeigersinn (CW) und gegen Uhrzeigersinn (CCW).
Welcher Takt soll zählen?	„Letzten Zustand“ von A und B merken: hier A=LOW und B=LOW Folgt darauf A=HIGH und B=LOW liegt CW vor. oder Folgt darauf A=LOW und B=HIGH liegt CCW vor. Durch Vergleich des letzten mit dem aktuellen Zustand kann die Drehrichtung entschieden werden. Gezählt wird nur o.g. Zustandsänderung, so dass nicht die Takte, sondern die Schritte gezählt werden.

Erfahrungen

Bouncing

Infolge des Bouncing bringt die Interrupt-Technik hier keine Vorteile.

Der Rotary Encoder ist hier nicht zeitkritisch.

Durch die Interrupt-Technik gehen keine Schritte verloren.

Andererseits fällt das Bouncing schwerer in Gewicht.

Polling

Zeigt hier keine Vorteile.

Interrupt

Zeigt hier eher Nachteile.

Algorithmus

Es muss ein Algorithmus gefunden werden, der die Bouncing-Effekte minimiert!

Entscheidung

Entscheidung für Interrupt-Technik weil prinzipiell besserer Ansatz!

Library Ben Baxton

Einführung

<http://www.buxtronix.net/2011/10/rotary-encoders-done-properly.html>

Ben Baxton hat beim Rotary Encoder die Ausgänge A und B vertauscht!
Die benutzte Library ist veraltet (seine Webseite).

Library von „brianlow“

Aktuelle Library von:

<https://github.com/brianlow/Rotary>

Library entpacken und in den Ordner „libraries“ der Arduino-IDE kopieren.

Pin-Anpassung Korrekturen in Rotary.cpp Zeile 92

```
unsigned char pinstate = (digitalRead(pin1) << 1) | digitalRead(pin2);
```

Algorithmus

Die Library enthält eine sogenannte „state-machine“, die basierend auf
aktuelle und vergangene Stellungen des RE zuverlässig zählt.

Ben Baxton state-machine

ergibt Index Spalte

A	B	pinstate=
0	0	0
1	0	2
1	1	3
0	1	1
0	0	0

count	A	B	AB	Index Reihe alter Status	Index Spalte aktuell	Index neue Reihe
count=1						
count=2	A=0	B=0	AB=00	indexold=1	pinstate=0	R_CW_NEXT 0x3
count=2	A=1	B=0	AB=10	indexold=3	pinstate=2	R_CW_FINAL 0x1
count=3	A=1	B=1	AB=11	indexold=1	pinstate=3	R_START 0x0
count=3	A=0	B=1	AB=01	indexold=0	pinstate=1	R_CW_BEGIN 0x2
count=3	A=0	B=0	AB=00	indexold=2	pinstate=0	R_CW_NEXT 0x3

ergibt Index Reihe

R_START 0x0
R_CW_FINAL 0x1
R_CW_BEGIN 0x2
R_CW_NEXT 0x3
R_CCW_BEGIN 0x4
R_CCW_FINAL 0x5
R_CCW_NEXT 0x6

Ben Baxton state-machine

		Neuer Status = Tabelle[Index Alter Status] [Index Aktueller Status]						
		0	1	2	3			
Neuer Status								
Alter Status								
ergibt Index Reihe						ergibt Index Spalte		
R_START 0x0	0	{R_START	R_CW_BEGIN	R_CCW_BEGIN	R_START}	A	B	pinstate=
R_CW_FINAL 0x1		// R_CW_FINAL				0	0	0
R_CW_BEGIN 0x2	1	{R_CW_NEXT	R_START	R_CW_FINAL	R_START DIR_CW}	1	0	2
R_CW_NEXT 0x3		// R_CW_BEGIN				1	1	3
R_CCW_BEGIN 0x4	2	{R_CW_NEXT	R_CW_BEGIN	R_START	R_START}	0	1	1
R_CCW_FINAL 0x5	3	{R_CW_NEXT	R_CW_BEGIN	R_CW_FINAL	R_START}	0	0	0
R_CCW_NEXT 0x6		// R_CCW_BEGIN						
	4	{R_CCW_NEXT	R_START	R_CCW_BEGIN	R_START}			
		// R_CCW_FINAL						
	5	{R_CCW_NEXT	R_CCW_FINAL	R_START	R_START DIR_CCW}			
		// R_CCW_NEXT						
	6	{R_CCW_NEXT	R_CCW_FINAL	R_CCW_BEGIN	R_START}			

Library Interrupt

```
/* Hands on Arduino Incremental Rotary Encoders
 * by EBW Enno Klatt 02/2019
 * Fifth sketch: Library from Ben Buxton's
 * EBW_Enno_05.ino
 * http://www.buxtronix.net/2011/10/rotary-encoders-done-
properly.html
 * Library von: https://github.com/brianlow/Rotary
 */
#include <Rotary.h> // define the class Rotary

#define intPin0 2 // Arduino UNO Hardware-Interrupt INT0
#define intPin1 3 // Arduino UNO Hardware-Interrupt INT1

// creating object "re" from class Rotary
Rotary re = Rotary(intPin0, intPin1);

long int count=0;
long int lastcount=0;
byte A, B;           // value of intPin0 / INT0
                    // value of intPin1 / INT1

void setup() {
  Serial.begin(57600);
  Serial.println("EBW_Enno_05.ino");
  // LEDs for debugging
  pinMode(11, OUTPUT); // show the movement of the Rotary Encoder
  pinMode(12, OUTPUT);

  re.begin(true); // initialize Rotary Encoder
  initInterrupt(); // initialize Interrupt
}
```

```
void loop() {
  // show count only when changed
  if (count != lastcount){
    Serial.print("count=");Serial.println(count);
  }
  // end of show count only when changed
  lastcount=count;

  // debugging LEDs
  A = digitalRead(intPin0);
  B = digitalRead(intPin1);
  A=!A;B=!B;
  if (A)  digitalWrite(11,HIGH);
  else   digitalWrite(11,LOW);
  if (B)  digitalWrite(12,HIGH);
  else   digitalWrite(12,LOW);
}

// calling the InterruptServiceRoutine for group
PCINT2
ISR(PCINT2_vect) {
  unsigned char result = re.process();
  if (result == DIR_CW) count++;
  if (result == DIR_CCW) count--;
}
```

Sketch Interrupt (Datei muss im Ordner des Sketches gespeichert sein)

```
/* Hands on Arduino Incremental Rotary Encoders
 * by EBW Enno Klatt 02/2019
 * Fifth sketch: Library from Ben Buxton's
 * http://www.buxtronix.net/2011/10/rotary-encoders-done-properly.html
 * Library von: https://github.com/brianlow/Rotary
 * http://m.arduino-projekte.webnode.at/registerprogrammierung/pinchangeinterrupt/
 *
 * The circuit:
 * encoder pin A to Arduino pin 2
 * encoder pin B to Arduino pin 3
 * encoder ground pin to ground (GND)
 */
```

```
void initInterrupt(){
```

```
  /* PCICR PinChangeInterruptControlRegister
   * enables in this case PCIE2 a group for DPin 0 to DPin 7
   * PCMSK2 PinChange2Mask (of group PCIE2)
   * enables specifig Pins
   *
   * Physical Pin | Port Pin | Hardware Interrupt Pin | PinChange Interrupt Pin |
   * 2 | PD2 | INT0 | PCINT18 |
   * 3 | PD3 | INT1 | PCINT19 |
   * 4 | PD4 | | PCINT20 |
   * 5 | PD5 | | PCINT21 |
   * 6 | PD6 | | PCINT22 |
   * 7 | PD7 | | PCINT23 |
   * sei()-function: sets the Global Enable Interrupt Bit
   *
   * ISR(PCINT2_vect): is the InterruptServiceRoutine for group PCIE2
   */
```

```
  PCICR |= (1 << PCIE2);
  PCMSK2 |= (1 << PCINT18) | (1 << PCINT19);
  sei();
```

```
}
```

Sketch 2 RE & 2 Taster I

```
/* Hands on Arduino Incremental Rotary Encoders
 * by EBW Enno Klatt 02/2019
 * Sketch 6: Library from Ben Buxton's
 * EBW_Enno_06.ino
 * http://www.buxtronix.net/2011/10/rotary-encoders-done-properly.html
 * Library von: https://github.com/brianlow/Rotary
 */
#include <Rotary.h> // define the class Rotary
                    // (modified by EBW Enno Klatt)

// creating objects from class Rotary for Rotary Encoders
Rotary left = Rotary(2, 3);
Rotary right = Rotary(4, 5);

long int countLeft=0, lastcountLeft=0;
long int countRight=0, lastcountRight=0;
byte A, B; // value of Pin 2
           // value of Pin 3
boolean switchLeft, switchRight; //state of switches
```

```
void setup() {
  Serial.begin(57600);
  Serial.println("EBW_Enno_06.ino");

  // pullup Pins 2, 3, 4, 5, 6, 7 (HIGH)
  // pinMode(2, INPUT_PULLUP); // done by Rotary-class
  // pinMode(3, INPUT_PULLUP); // done by Rotary-class
  // pinMode(4, INPUT_PULLUP); // done by Rotary-class
  // pinMode(5, INPUT_PULLUP); // done by Rotary-class
  pinMode(6, INPUT_PULLUP); // switch 1 to HIGH
  pinMode(7, INPUT_PULLUP); // switch 2 to HIGH

  // LEDs for debugging
  pinMode(11, OUTPUT); // show the movement of A and B
  pinMode(12, OUTPUT);

  // sets input with pullup to HIGH
  left.begin(true); // pullup Pins 2, 3 (HIGH)
  right.begin(true); // pullup Pins 4, 5 (HIGH)
  initInterrupt(); // initialize Interrupt-Capability
}
```

Sketch 2 RE & 2 Taster II

```
void loop() {
  // show counts and switches only when changed
  if (countLeft != lastcountLeft){
    Serial.print("countLeft=");Serial.println(countLeft);
    lastcountLeft=countLeft;
  }
  if (countRight != lastcountRight){
    Serial.print("countRight=");Serial.println(countRight);
    lastcountRight=countRight;
  }
  if (switchRight) {
    Serial.print("switchRight="); Serial.println(switchRight);
    switchRight=!switchRight;
  }
  if (switchLeft) {
    Serial.print("switchLeft=");Serial.println(switchLeft);
    switchLeft=!switchLeft;
  }
  // end of show counts only when changed

  // debugging LEDs
  A = digitalRead(2);
  B = digitalRead(3);
  A=!A;B=!B;
  if (A)  digitalWrite(11,HIGH);
  else   digitalWrite(11,LOW);
  if (B)  digitalWrite(12,HIGH);
  else   digitalWrite(12,LOW);
}
```

```
// calling the InterruptServiceRoutine for group PCINT2
ISR(PCINT2_vect) {
  unsigned char result_left = left.process();
  if (result_left == DIR_CW) countLeft++;
  if (result_left == DIR_CCW) countLeft--;

  unsigned char result_right = right.process();
  if (result_right == DIR_CW) countRight++;
  if (result_right == DIR_CCW) countRight--;

  // Port D holds the state of digital Pins 6 and 7
  switchLeft = !((PIND & B01000000) >> 6);
  switchRight = !((PIND & B10000000) >> 7);
}
```


Sketch Interrupt (Datei muss im Ordner des Sketches gespeichert sein)

```
/* Hands on Arduino Incremental Rotary Encoders
 * by EBW Enno Klatt 02/2019
 * Fifth sketch: Library from Ben Buxton's
 * Interrupt.ino
 * http://www.buxtronix.net/2011/10/rotary-encoders-done-properly.html
 * Library von: https://github.com/brianlow/Rotary
 * http://m.arduino-projekte.webnode.at/registerprogrammierung/pinchangeinterrupt/
 */

* The circuit:
* first encoder pin A to Arduino pin 2
* first encoder pin B to Arduino pin 3
* second encoder pin A to Arduino pin 4
* second encoder pin B to Arduino pin 5
* encoder ground pin to ground (GND)
* button 1 to Arduino pin 6
* button 2 to Arduino pin 7
*/

void initInterrupt(){
  /* PCICR PinChangeInterruptControlRegister
   * enables in this case PCIE2 a group for DPin 0 to DPin 7
   * PCMSK2 PinChange2Mask (of group PCIE2)
   * enables specifig Pins
   *
   * | | Hardware | PinChange |
   * Physical Pin | Port Pin | Interrupt Pin | Interrupt Pin |
   * 2 | PD2 | INT0 | PCINT18 |
   * 3 | PD3 | INT1 | PCINT19 |
   * 4 | PD4 | | PCINT20 |
   * 5 | PD5 | | PCINT21 |
   * 6 | PD6 | | PCINT22 |
   * 7 | PD7 | | PCINT23 |
   * sei()-function: sets the Global Enable Interrupt Bit
   *
   * ISR(PCINT2_vect): is the InterruptServiceRoutine for group PCIE2
   */
}
```

```
PCICR |= (1 << PCIE2);
PCMSK2 |= (1 << PCINT18) | (1 << PCINT19);
PCMSK2 |= (1 << PCINT20) | (1 << PCINT21);
PCMSK2 |= (1 << PCINT22) | (1 << PCINT23);
sei();
}
```

Download

Sketche und Bibliothek herunterladen von:

https://c.web.de/@334322739298962515/PSIFSmrtS_O5HBXcSfoNnA
