

Endliche Automaten & Rotary Encoder

Thema

Rotary Encoder (RE)

RE Aufbau

Schritte & Takte

Decoding Tabelle

RE Funktion

RE Bouncing

Wege zur Lösung

“Endlicher Automat” von Peter Csurgay

Exkurs Automaten

Tresor

Tresor Symbole

Tresor Tabelle

Mealey Automat

Lampe

Lampe Symbole & Funktionen

Anwendung auf RE

RE mit Übergängen

RE FLACY

RE Symbole

RE Tabelle

Polling oder Interrupt

Arduino Script

Sketch Peter Csurgay

Sketch EBW

Links

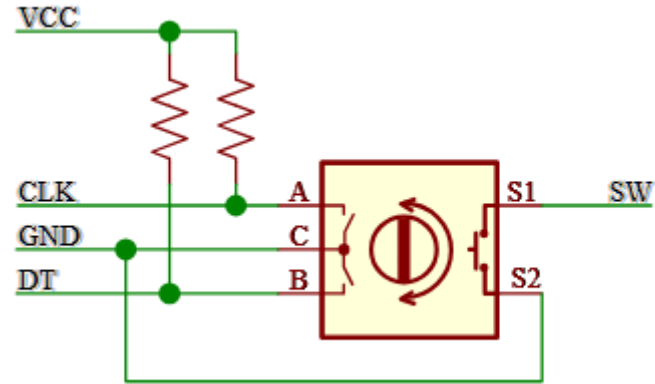
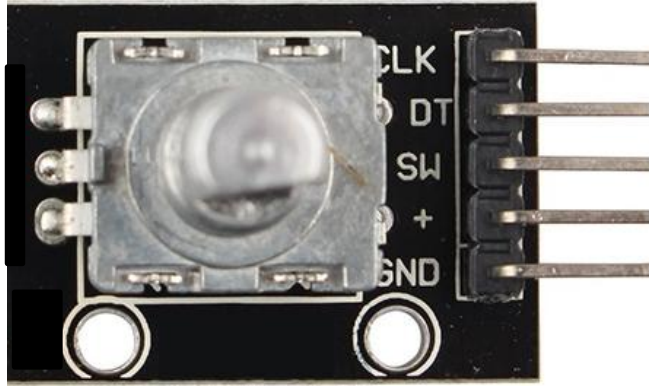
Kontakt & Dokumente

Thema

Anlass	<ul style="list-style-type: none">• Bei einer RGB-LED sollte der Farbwert im HSV-Farbraum genau eingestellt werden. Die Werte liegen zwischen 0 und 360.• Die Einstellung der Werte über ein Potentiometer erwies sich als zu ungenau.• Ein Rotary Encoder (RE) mit z.B. 20 Schritten pro Umdrehung als Lösung.
Problem	<ul style="list-style-type: none">• Das einfache Auslesen der beiden Datenleitungen zeigt einen Bouncing-Effekt, was zu Sprüngen beim Zählen führt.
Abhilfe	<ul style="list-style-type: none">• Geeigneten Arduino-C++-Code, der die Sprünge eliminiert.
Themen	<ul style="list-style-type: none">• Aufbau und Funktion des Rotary Encoders.• Aus dem Internet Erfahrungen mit Software Lösungen erkunden.• Theoretischer Ansatz „endlicher Automat“ verstehen und anwenden.• Rotary Encoder als Mealey-Automat darstellen.• Umsetzen des Encoder-Automaten in lesbaren, d.h. nachvollziehbaren, Arduino-C++-Code.• Optimieren hinsichtlich der Geschwindigkeit und der Effizienz.

Rotary Encoder (RE)

KY-040



Datenleitungen	CLK	(RE pin A)
	DT	(RE pin B)

Taster	SW
--------	----


Spannung 5 V	„+“	(VCC)
--------------	-----	-------

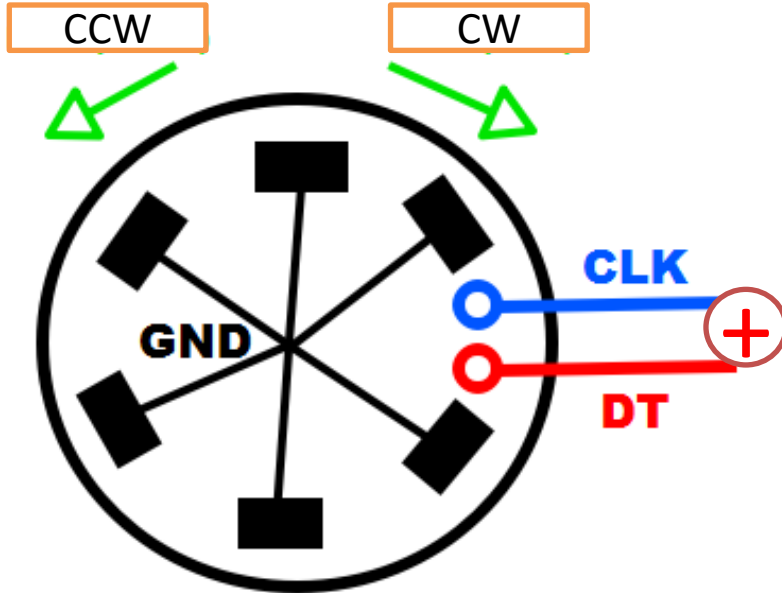
Masse	GND
-------	-----

Quelle:	https://docs.wokwi.com/de-DE/parts/wokwi-ky-040
---------	---

RE Aufbau

Rotary-Encoder
KY-040

Die Kontakte liegen auf positiver Spannung (**Pullup-Widerstände**). 
Die dunklen Kontakte sind mit Masse (**GND**) verbunden.



CW	
CLK	DT
1	1
0	1
0	0
1	0

CCW	
CLK	DT
1	1
1	0
0	0
1	0

Quelle:

<https://www.instructables.com/A-Complete-Arduino-Rotary-Solution/>

Schritte & Takte

Der KY-40 erzeugt 4 Takte für einen Schritt.
Fügt man CLK und DT binär zusammen, liegt ein gray-code vor.

Uhrzeigersinn CW	CLK	DT	Gegen Uhrzeigersinn CCW	CLK	DT
Start	1	1	Start	1	1
Takt 1	0	1	Takt 1	1	0
Takt 2	0	0	Takt 2	0	0
Takt 3	1	0	Takt 3	0	1
Takt 4	1	1	Takt 4	1	1

1 Umdrehung: Besteht aus 20 Schritten

jeder Schritt: Aus 4 Takten

Es sollen hier nur die Schritte gezählt werden.

Decoding Tabelle

Hier: Clockwise (CW)

CLK	1	0	0	1	1
		CLKfall		CLKrise	
DT	1	1	0	0	1
	DTKrise		DTfall		
status	(4)	1	2	3	4

	Takt nacher	1	2	3	4
Takt vorher		01	00	10	11
(4)	11 → CLKfall				
1	01		Dtfall		
2	00			CLKrise	
3	10				DTrise

CLK

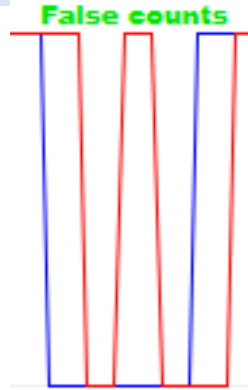
DT

RE Funktion

Ziel	Zählen der Schritte. Unterscheiden zwischen Uhrzeigersinn (CW) und gegen den Uhrzeigersinn (CCW).
Ausgangszustand?	CLK auf HIGH und DT auf HIGH.
Drehrichtung?	Bei CW folgt: auf CLK=HIGH und DT=HIGH: CLK=LOW und DT=HIGH Bei CCW folgt: auf CLK=HIGH und DT=HIGH: CLK=HIGH und DT=LOW
Algorithmus	Durch Vergleich des letzten mit dem aktuellen Zustand kann die Drehrichtung entschieden werden!
Prinzip	Ein vorheriger Zustand , gefolgt von einem Übergang , führt zu einem bestimmten nachfolgenden Zustand .
Endlicher Automat (Typ Mealey)	Damit sind die Voraussetzungen für einen „endlichen Automaten“ erfüllt.

RE Bouncing

Problem	Leider kommt es zum Bouncing-Effekt.
---------	--------------------------------------



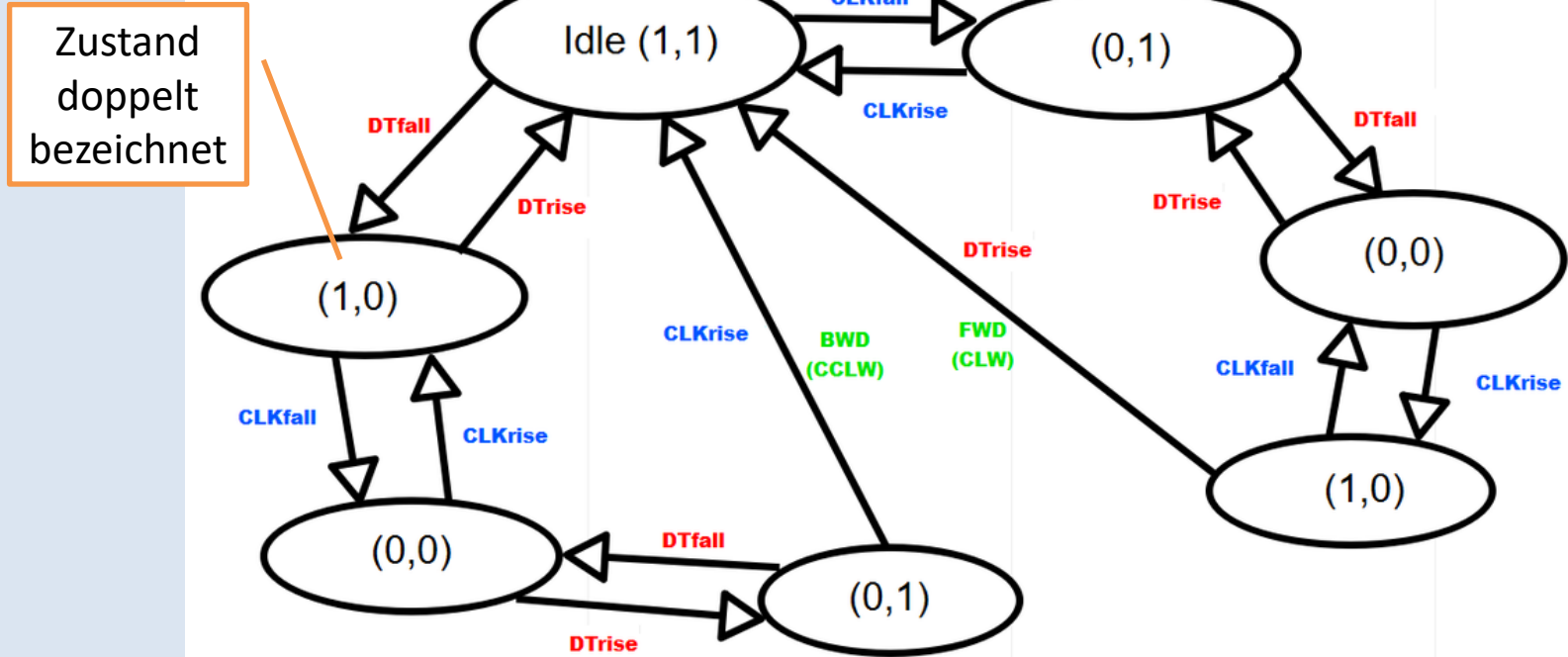
Ursache	Beim Übergang von einem Takt zum Folgetakt treten nicht erwünschte Sprünge im Potential der Datenleitungen CLK und DT auf.
---------	--

Folgen für Algorithmus	Ein Programm (Arduino-Sketch) sollte diesen Effekt erkennen, ansonsten treten beim Zählen der Schritte unerwünschte Sprünge auf.
------------------------	--

Wege zur Lösung

Ziel	Arduino-C++-Code entwickeln
Aufgaben	<ul style="list-style-type: none">• Zählen der Schritte im Uhrzeigersinn (CW) und gegen den Uhrzeigersinn (CCW).• Bouncing erkennen und sauberes Zählen erreichen.
Google ChatGPT	Zahlreiche sehr unterschiedliche Ansätze.
Bester Vorschlag	„A Complete Arduino Rotary Solution“ (FSM) von Peter Scurgay. Siehe: A Complete Arduino Rotary Solution : 5 Steps – Instructables
	Die Lösungsidee besteht darin, die Funktion des Rotary Encoders in einer „ Finite State Machine “ (FSM) abzubilden.
Englisch:	„Finite State Machine“ (FSM)
Deutsch:	„endlicher Automat“
Idee	Nur bei einem fehlerfreien Durchlauf der 4 Takte, soll ein Schritt gezählt werden.

“Endlicher Automat” von Peter Csurgay



Quelle: <https://www.instructables.com/A-Complete-Arduino-Rotary-Solution/>

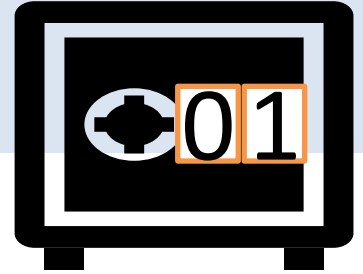
Exkurs Automaten

Automaten	Ein Thema der theoretischen Informatik.
Definition	<ul style="list-style-type: none">Automaten kann man sich als eine Art "Maschine" vorstellen, die einem festgelegtem Schema folgt.
	<ul style="list-style-type: none">Ein „endlicher Automat“ ist ein Modell eines Systems mit endlichen vielen Zuständen.
Kaffeemaschine Zustände	Eine Kaffeemaschine kann sich in den verschiedenen Zuständen befinden: <ul style="list-style-type: none">warten,Kaffee brühen,Kaffee warm halten.
Übergänge	Wenn der Brühvorgang ausgelöst wird, soll der Kaffee gebrüht werden. Wenn dieser beendet ist, soll der Kaffee warm halten werden.
Erfahrung	Der Automat setzt sich aus Zuständen und Übergängen zusammen.
Definition	Zu jedem Zeitpunkt befindet sich ein Automat in genau einem Zustand. Übergänge werden anhand einer Übergangsfunktion beschrieben.

Tresor

Aufbau

Ein Schlüsseltresor mit 2 Tasten.
Eine Taste für die Ziffer „0“.
Eine Taste für die Ziffer „1“.

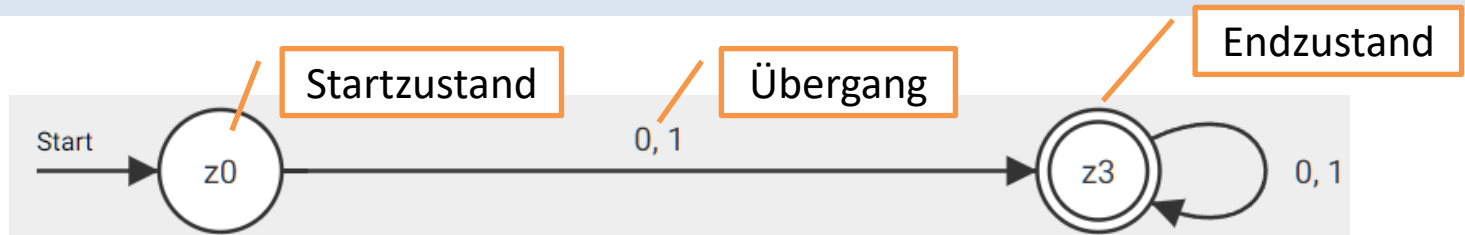


Bedingungen

1. Es sollen mindestens 3 Ziffern eingegeben werden.
2. Eine Pin mit 3 Nullen ist nicht erlaubt.

Versuch 1:

Zustände:
z0, z1



Bedingungen?

Es können beliebige Pin-Varianten eingegeben werden.
Nicht erfüllt.

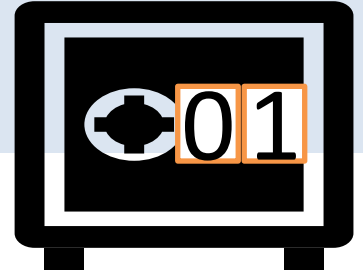
Hinweis:

Der Endzustand ist entweder wahr oder falsch!

Tresor

Aufbau

Ein Schlüsseltresor mit 2 Tasten.
Eine Taste für die Ziffer „0“.
Eine Taste für die Ziffer „1“.



Bedingungen

1. Es sollen mindestens 3 Ziffern eingegeben werden.
2. Eine Pin mit 3 Nullen ist nicht erlaubt.

Versuch 2:

Zustände:
z0 bis z3



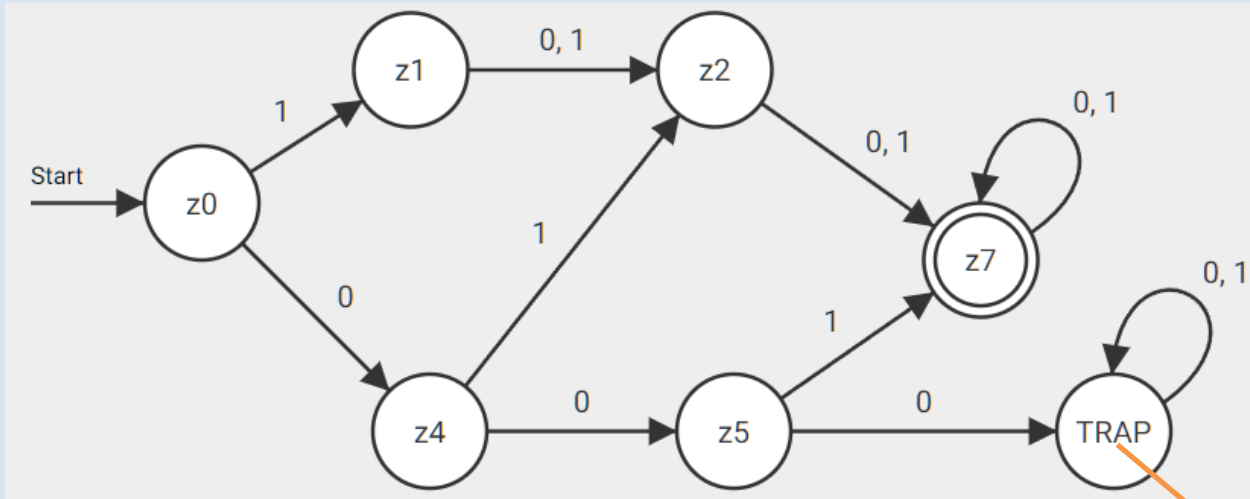
Bedingungen?

- Bedingung 1 erfüllt: Es müssen 3 Ziffern in beliebigen Pin-Varianten eingegeben werden.
- Bedingung 2 nicht erfüllt.

Z.B. führt das Wort „11“ zur Blockade.

Tresor

Versuch 3:



Erstellt mit
FLACI

Fehler
Zustand

Bedingungen: mindestens 3 Ziffern und keine Nuller-Pin erfüllt.

Tresor Symbole

Symbole & Bedeutungen

Typ:	Norm	Hier:
Akzeptor	Q	$z =$ endliche Menge von Zuständen
	Σ Sigma	$E =$ Eingabealphabet
	δ Delta	$u =$ Übergangsfunktion (Zustand)
	Q_0	$z_0 =$ Startzustand
	F	$z_E =$ Endzustand
	Σ^*	$E^* =$ Menge aller Wörter

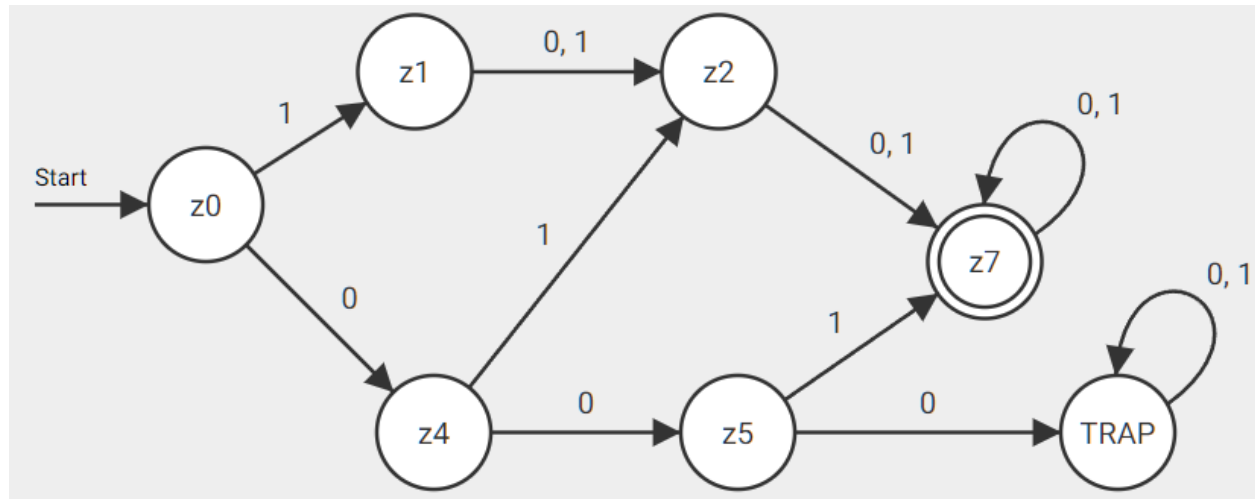
Zustände	$z = \{ z_0, z_1, z_2, z_3, z_4, z_5, z_6, z_7, \text{Trap} \}$
Eingabealphabet	$E = \{ 0, 1 \}$
Startzustand	$z_0 = z_0$
Endzustand	$z_E = z_7$
Wörter	$E^* = \{ 100, 110, 001, \dots \}$
Zustand-Funktion	$u : z \times E \rightarrow z$

Mengenlehre

Quelle: FLACI

Tresor Tabelle

Übergangstabelle Tresor



δ	0	1
TRAP	TRAP	TRAP
z0	z4	z1
z1	z2	z2
z2	z7	z7
z4	z5	z2
z5	TRAP	z7
z7	z7	z7

Quelle FLACI

Mealey Automat

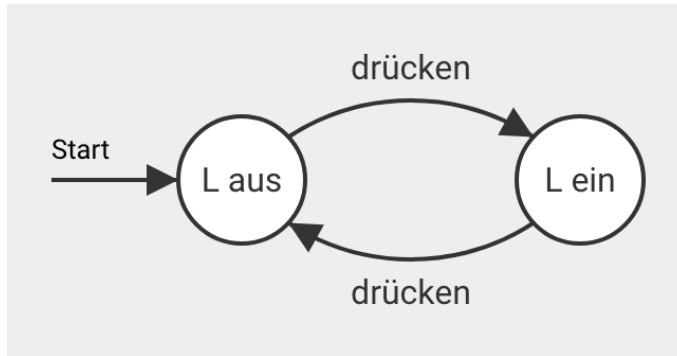
Bisher **Akzeptor**: Endergebnis **wahr oder falsch**.
Jetzt **Transduktor**: Ein Automat mit mehreren **Eingaben- und Ausgaben**.

Der Mealey-Automat hat keine Endzustände.

Beispiel
Lampe



Zustandsgraph
(nicht formal)



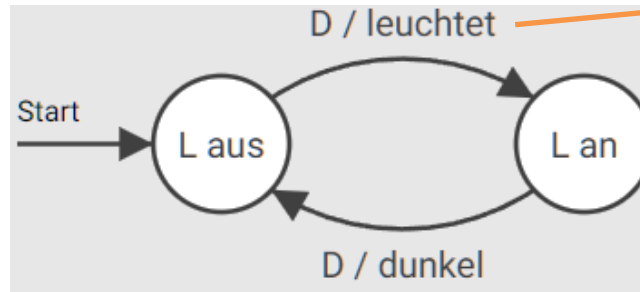
Interpretation:

Zu Beginn ist das Licht aus. Wenn ich den Schalter drücke, geht das Licht an. Wenn ich den Schalter nochmals drücke, geht es wieder aus.

Quelle: [Endliche Automaten | EF Informatik 2023](#)

Lampe

Zustandsgraph
Lampe



Eingabe / Ausgabe

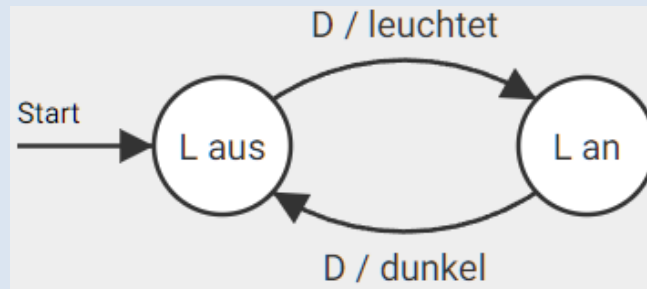
Achtung:
Der Übergangspfeil ist mit Angaben zur
Ein- und Ausgabe bezeichnet.

Zustandstabelle

	Eingabe von → alter Zustand ↓	D
	L aus	L ein
	L ein	L aus

Lampe Symbole & Funktionen

Zustände $z = \{ \text{L aus}, \text{L ein} \}$
 Eingabealphabet $E = \{ D \}$
 Ausgangsalphabet $A = \{ \text{leuchtet}, \text{dunkel} \}$
 Startzustand $z_0 = \text{„L aus“}$
 Menge aller Wörter $E^* = \{ D \}$
 Funktionen
Zustand: $u : z \times E \rightarrow z$
Ausgabe: $a : z \times E \rightarrow z$



Übergangs- & Ausgabefunktion

Alter Zustand		Eingabe		Neuer Zustand		Ausgabe
z		E		$u : z \times E \rightarrow z$		$a : z \times E \rightarrow z$
L aus	x	D	→	L ein		leuchtet
L ein	x	D	→	L aus		Dunkel

Anwendung auf RE

- Die Potentiale der Datenleitungen CLK und DT wechseln in einem festgelegten Rhythmus. Das ergibt einen stetigen Wechsel von **Zuständen**.
- Die Potentialänderung selbst kann als **Übergang** zwischen den Zuständen aufgefasst werden.


Im Uhrzeigersinn hat der RE die Zustände :	11	01	00	10
In C++ bezeichnet als:	IDLE_11	CW_01	CW_00	CW_10
In FLACI bezeichnet als:	ID11	CW01	CW00	CW10

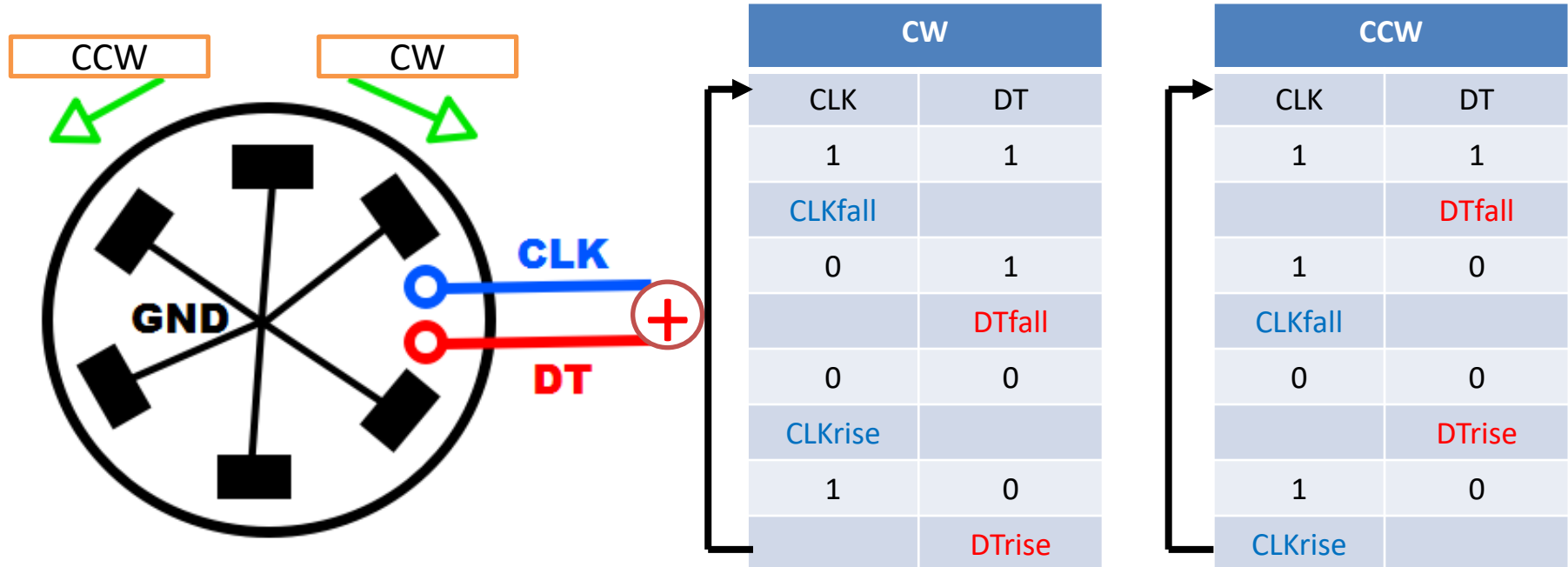
Gegen den Uhrzeigersinn die Zustände :	11	10	00	01
In C++ bezeichnet als:	IDLE_11	CCW_10	CCW_00	CCW_01
In FLACI bezeichnet als:	ID11	CCW10	CCW00	CCW01

Hiermit sollte sich der Zustandsgraph eines RE erstellen lassen.

RE mit Übergängen

Rotary-Encoder
KY-040

Die Kontakte liegen auf positiver Spannung **Pullup-Widerstände**. 
Die dunklen Kontakte sind mit Erde (**GND**) verbunden.

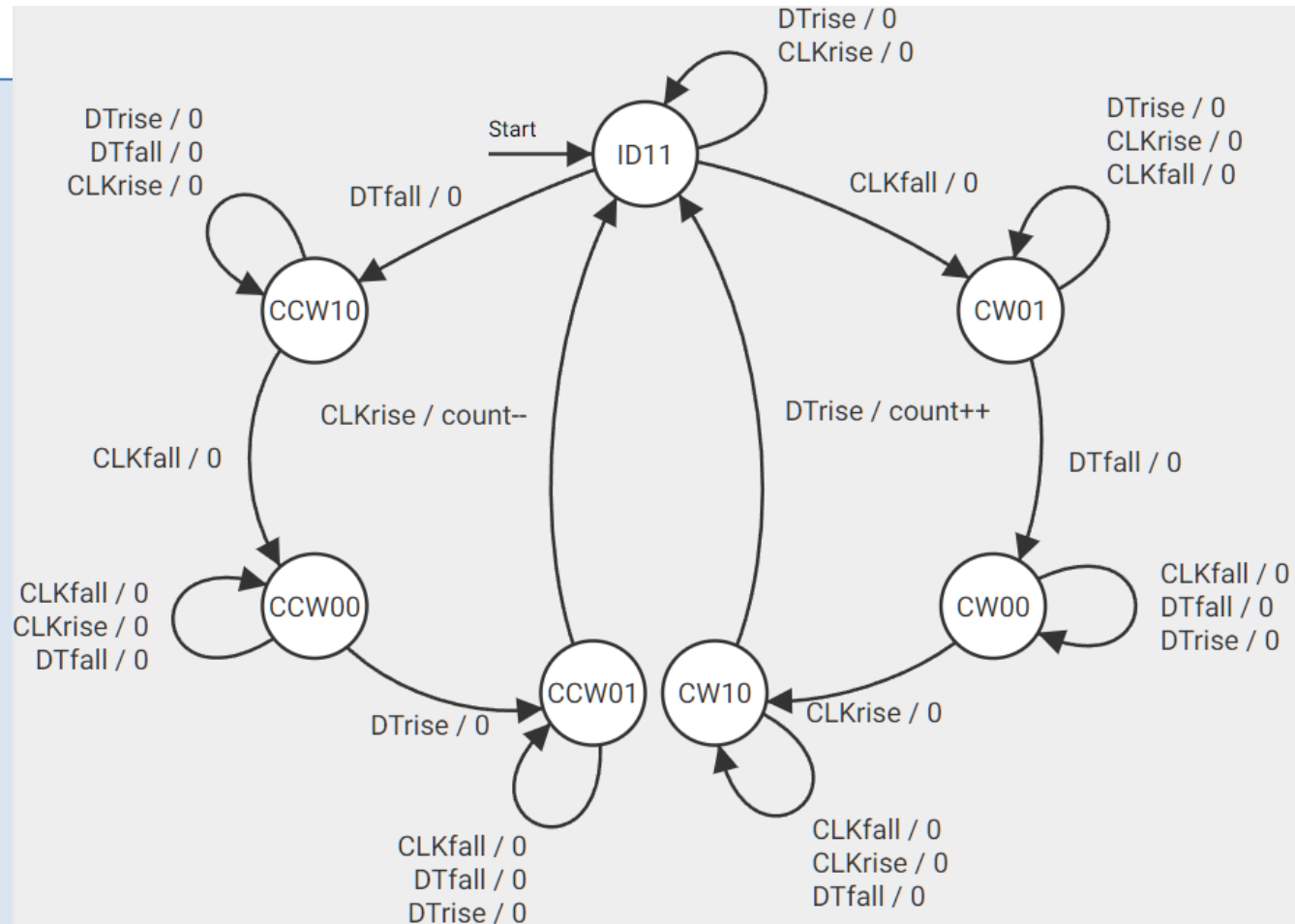


Quelle:

<https://www.instructables.com/A-Complete-Arduino-Rotary-Solution/>

RE FLACY

Zustandsgraph Typ: Mealey



Quelle: FLACY

RE Symbole

Typ: Mealey	z = endliche Menge von Zuständen E = Eingabealphabet A = Ausgabealphabet u = Übergangsfunktion a = Ausgabefunktion z ₀ = Startzustand E* = Menge aller Wörter							
Zustände	z = {	IDLE_11,	CW_01,	CW_00,	CW_10,	CCW_10	CCW_00	CCW_01 }
Eingabealphabet	E = {	CLKfall,	CLKrise,	DTfall,	DTrise }			
Ausgabealphabet	A = {	count++,	count++	, 0}				
Wörter	E* = {	„CLKfall DTfall CLKrise DTrise“, „CLKfall CLKfall DTfall CLKrise Dtrise“, ...}						
Startzustand	z ₀ = IDLE_00							
Funktionen	Zustand: u: z x E → z Ausgabe: a: z x E → z							

RE Tabelle

Übergangstabelle (C++ Bezeichner)					
Alter Zustand z		Eingabe E		Neuer Zustand $u : z \times E \rightarrow z$	Ausgabe $a : z \times E \rightarrow z$
IDLE_11	x	CLKfall	→	CW_01	λ
CW_01	x	DTfall	→	CW_00	λ
CW_00	x	CLKrise	→	CW_10	λ
CW_10	x	DTrise	→	IDLE_11	count++
IDLE_11	x	DTfall	→	CCW_10	λ
CCW_10	x	CLKfall	→	CCW_00	λ
CCW_00	x	DTrise	→	CCW_01	λ
CCW_10	x	CLKrise	→	IDLE_11	count--

λ : bedeutet keine Ausgabe

Polling oder Interrupt

Polling

```
void setup() {  
    // put your setup code here, to run once:  
  
}  
  
void loop() {  
    // code which shall be executed over and over  
    // in this case polling pins 3/CLK and 2/DT  
    CLK = digitalRead(CLKpin); // read Pin 3/CLK  
    DT  = digitalRead(DTpin);  // read Pin 2/DT  
    // ...  
}
```

Interrupt

```
void setup() {  
    // put your setup code here, to run once:  
    // ...  
    // Both CLK and DT will trigger interrupts  
    // for all level changes  
    attachInterrupt(digitalPinToInterrupt(CLKpin), ISRchange,  
                                                             CHANGE);  
    attachInterrupt(digitalPinToInterrupt(DTpin), ISRchange,  
                                                             CHANGE);  
}  
  
void loop() {  
}  
  
void ISRchange() {  
    oldState = state;  
    int CLK = digitalRead(CLKpin);  
    int DT  = digitalRead(DTpin);  
    //...  
}
```

Arduino Script

Aufgabe	Aus dem Zustandsgraph oder der Übergangstabelle einen lesbaren Arduino-C++-Code erstellen.
„else if“	<pre>if (condition1) { // block of code to be executed if condition1 is true } else if (condition2) { // block of code to be executed if the condition1 is false and condition2 is true }</pre>
Comparison operators	<code>==</code> (Equal to) - the comparison of two values leads to true (1) or false (0)
Logical operators	<code>&&</code> (AND) - all conditions must be true <code> </code> (OR) - at least one condition must be true <code>!</code> (NOT) - reverses a condition (true → false, false → true)

Arduino Script

„enum“	Ein enum ist ein spezieller Typ der eine Gruppe von Konstanten (nicht änderbar) definiert.
Syntax	<pre>enum Level { Apfel, Birne, Orange }; // Index 0 1 2</pre>
	<ul style="list-style-type: none">• In Zuweisungen verwendet man die Bezeichner, die in Wirklichkeit den Index als Werthaben.• In Bedingungen werden Indexe verglichen.
Beispiel	<pre>// Index 0 Index 1 enum State {IDLE_11 , CW_01, ...}; int state; // from previous function-call // Index 0 // e.g. state = IDLE_11 // 0 == 0 Index 1 If (state == IDLE_11) state = CW_01;</pre>

Sketch Peter Csurgay

```
// State Machine transitions for CLK level changes
void rotaryCLK() {
    if (digitalRead(CLK) == LOW) {
        if (state == IDLE_11) state = CW_01;
        else if (state == CCW_10) state = CCW_00;
    }
    else {
        if (state == CW_00) state = CW_10;
        else if (state == CCW_01) { state = IDLE_11; curVal--; }
    }
}

// State Machine transitions for DT level changes
void rotaryDT() {
    if (digitalRead(DT) == LOW) {
        if (state == IDLE_11) state = CCW_10;
        else if (state == CW_01) state = CW_00;
    }
    else {
        if (state == CCW_00) state = CCW_01;
        else if (state == CW_10) { state = IDLE_11; curVal++; }
    }
}
```

Hinweis: Am besten über den Zustandsgraph zu verstehen.

Sketch EBW

```
// transition table
// from IDLE_11 with CLKfall (LOW) to CW_01
// from CW_01 with DTfall (LOW) to CW_00
// from CW_00 with CLKrise (HIGH) to CW_10
// from CW_10 with DTrise (HIGH) to IDLE_11

// from IDLE_11 with DTfall (LOW) to CCW_10
// from CCW_10 with CLKfall (LOW) to CCW_00
// from CCW_00 with DTrise (HIGH) to CCW_01
// from CCW_01 with CLKrise (HIGH) to IDLE_11

void ISRchange() {
    // ...
    int CLK = digitalRead(CLKpin);
    int DT = digitalRead(DTpin);
    // CW
    if ( state==IDLE_11 && CLK==LOW ) state=CW_01;
    if ( state==CW_01 && DT ==LOW ) state=CW_00;
    if ( state==CW_00 && CLK==HIGH ) state=CW_10;
    if ( state==CW_10 && DT ==HIGH ) {state=IDLE_11; count++;}
    // CCW
    if ( state==IDLE_11 && DT ==LOW ) state=CCW_10;
    if ( state==CCW_10 && CLK==LOW ) state=CCW_00;
    if ( state==CCW_00 && DT ==HIGH ) state=CCW_01;
    if ( state==CCW_01 && CLK==HIGH ) {state=IDLE_11; count--;}
    // ...
}
```

Links

https://de.wikipedia.org/wiki/Endlicher_Automat

https://de.wikibooks.org/wiki/Theoretische_Informatik/_Das_Prinzip_des_Automaten

<https://informatik.mygymer.ch/ef2023/02-automaten/01-automaten.html>

https://lehrerfortbildung-bw.de/u_matnatech/informatik/gym/bp2016/fb2/05_automaten/1_hintergrund/1_infos/

https://bildungsserver.berlin-brandenburg.de/fileadmin/bbb/unterricht/faecher/naturwissenschaften/informatik/theoretische_informatik/automaten/automaten_und_sprachen.pdf

<https://www.instructables.com/A-Complete-Arduino-Rotary-Solution/>

https://www.pjrc.com/teensy/td_libs_Encoder.html?#

Kontakt & Dokumente

E-Mail: H39@email.de

GitHub: <https://github.com/EKlatt/Experiences>
Verzeichnis “Automaten”
