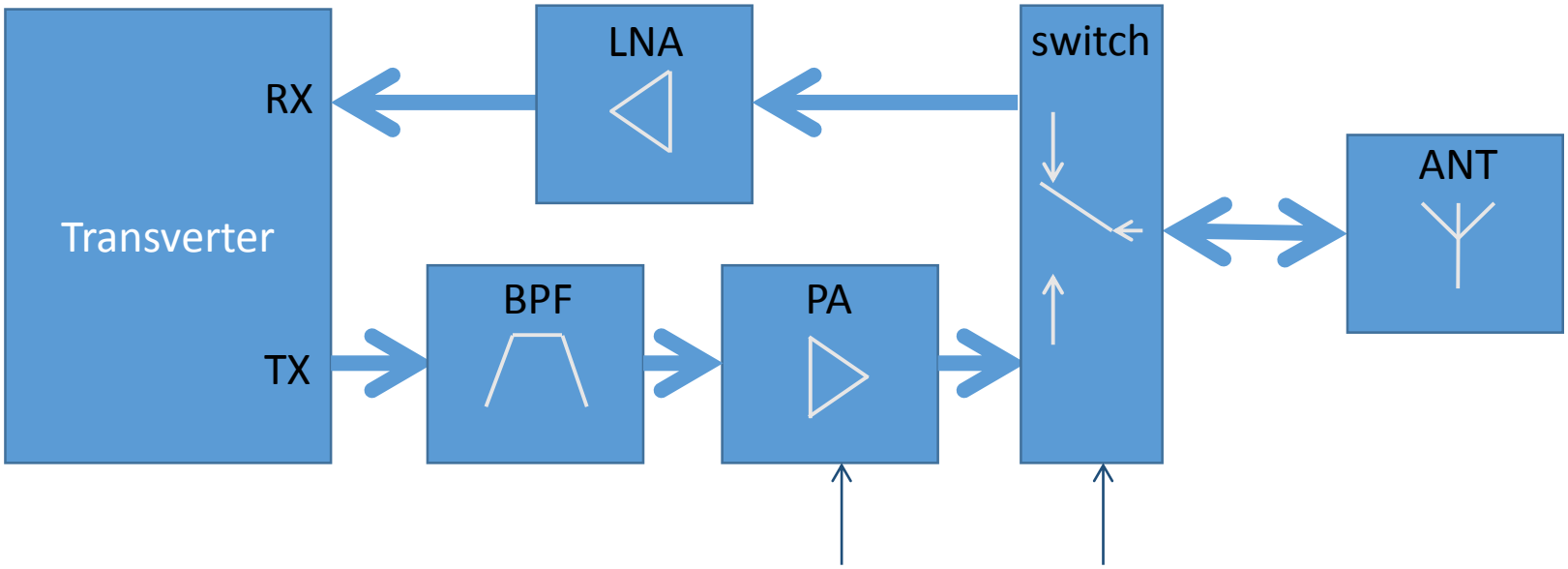
Ein „Helferlein“ für alle Fälle

Dezember 2018, OE8WOZ

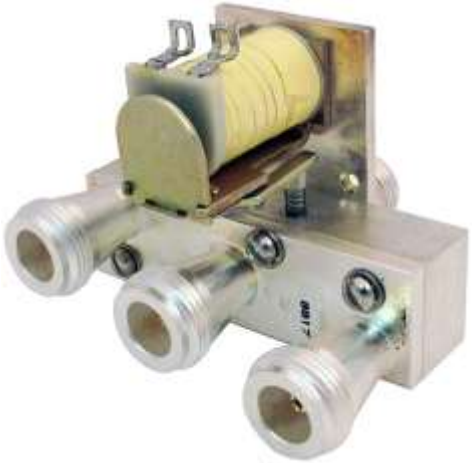
Typische AFU Anwendung

- High-End / High-Power Setup – „Timing“ ist alles!



Es geht nicht ohne sie….

- HF/RF- Schalter

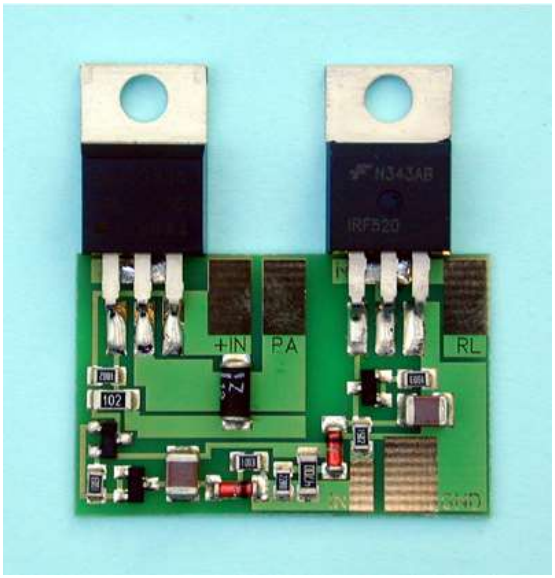


Generelle Unterschiede

- Monostabile Schalter („Failsafe“):
  - Ein Grundzustand (mechanische Feder)
  - Ein Zustand elektrisch erregt
- Selten „Gute“ im Surplus-Handel zu finden
  - Teuer, kosten Strom, leider recht oft 24V Typen
  - Einfach zu verwenden (wie „normale“ Relais)

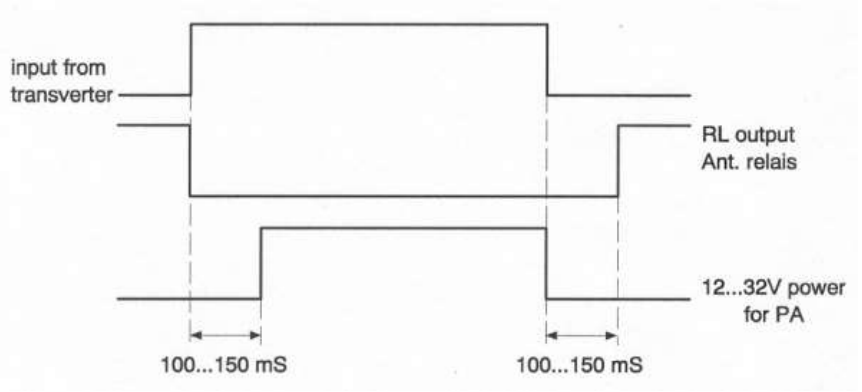
- Bistabile (Multistabile) Schalter („Latching“):
  - Jeder Zustand wird elektrisch eingestellt
  - Verbleibt auch im stromlosen Zustand an der gewählten Position
- Leichter im Surplus-Handel zu finden
  - Günstig, sparsam, 12V Typen recht häufig
  - Vor allem „Multischalter“
  - (angeblich) schwieriger anzusteuern … ?

Der SEQ4 von Kuhne (DB6NT)



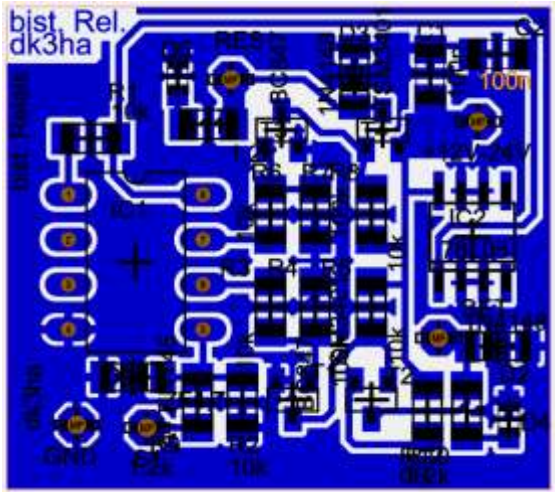
- 1x Leistungsausgang
- 1x Schaltausgang
- 1x Schaltgeschwindigkeit
- Genauigkeit ca.: +/-20%
- Schwer(er) nachzubauen (SMT)

Erhältlich im Kuhne-Shop (Nachfolger des SEQ2)



- Der bekannte „Klassiker“
- Failsafe-Relais-Applikationen
- PA-Ansteuerung

Steuerung bist. Relais (DK3HA)



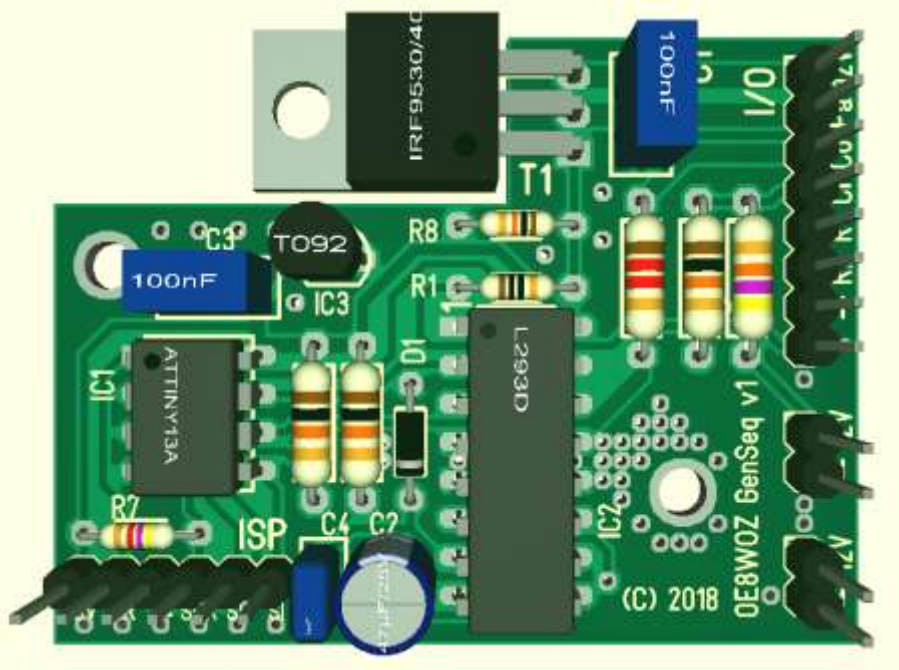
- kein Leistungsausgang
- 2x Schaltausgang C+
  (C- mit alternativer Platine)
- 1x Schaltgeschwindigkeit
- Genauigkeit ca.: +/-5%
- Mischaufbau (SMT+THT)
- Programmierbar

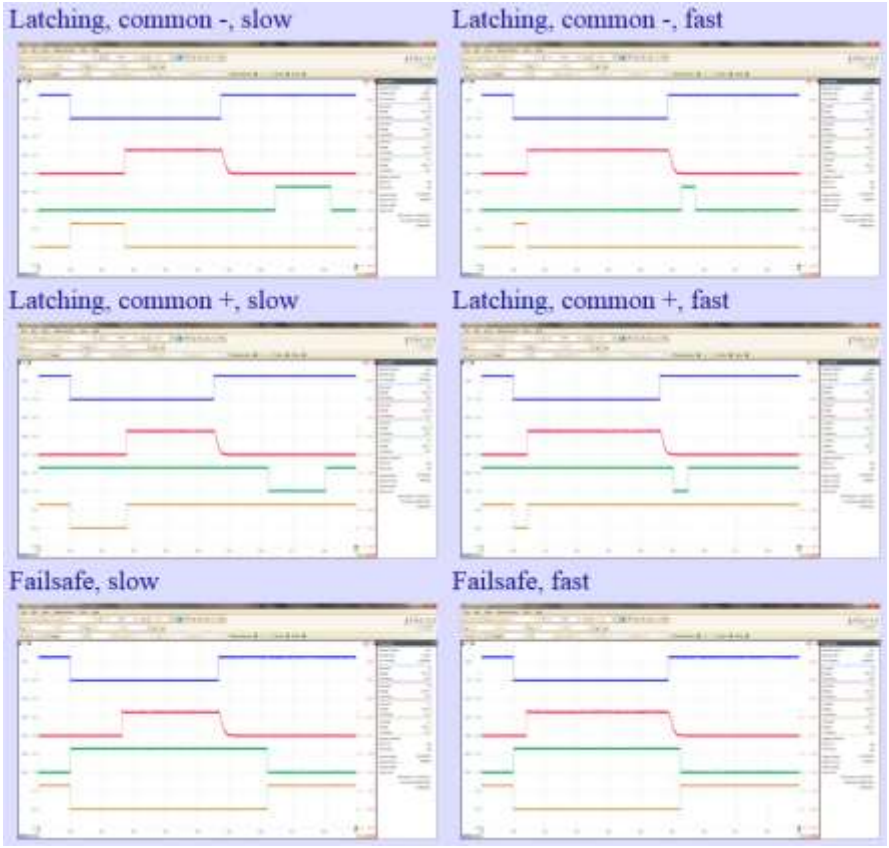
Vorgestellt 2015 in der GHz-Tagung in Dorsten / Deutschland

→durch Zufall im Web gefunden…

- Latching-Relais-Applikationen
- Keine PA Ansteuerung
- Leider kein Plattform-Design („all in one“)

Der GenSeq V1





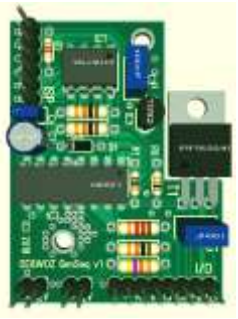
- 1x Leistungsausgang
- 3x Schaltausgänge
- 1x optionaler Ein/Ausgang
- ca. +/-5% Genauigkeit
- Einfach nachzubauen (THT)
- Beliebig programmierbar/nutzbar

- Mit der Basis-Software
  - Failsafe oder Latching C+ / C-
  - 2 Schaltgeschwindigkeiten
- Kann ohne Programmiergerät umgestellt werden
- **Plattform-Konzept**

Kostenvergleich (sehr grob, bei Selbstbau)

- SEQ 4

- GenSeqV1



- bist. SW. Umsch.

DB6NT	Stück	Preis/Stück	Summe €
Widerstände	8	0,10	0,80
Dioden	2	0,20	0,40
Kapazitäten	2	0,20	0,40
Elko	1	1,00	1,00
Transistoren	3	0,50	1,50
Leistungstr.	2	1,00	2,00
Gesamt			6,10

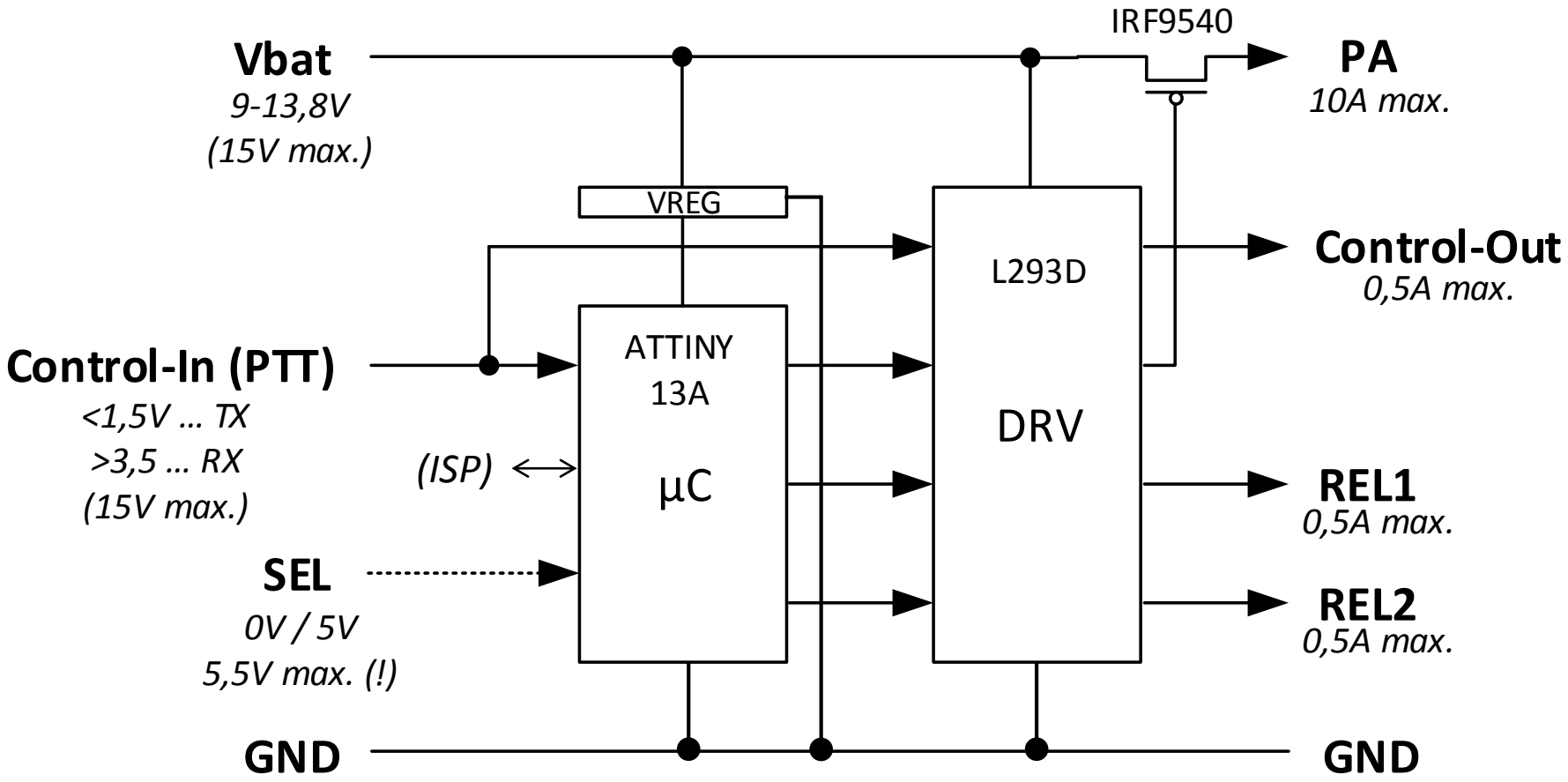
OE8WOZ	Stück	Preis/Stück	Summe €
Widerstände	6	0,10	0,60
Dioden	1	0,20	0,20
Kapazitäten	3	0,20	0,60
Elko	1	1,00	1,00
IC	3	1,30	3,90
Leistungstr.	1	1,00	1,00
Gesamt			7,30

DK3HA	Stück	Preis/Stück	Summe €
Widerstände	12	0,10	1,20
Dioden	5	0,20	1,00
Kapazitäten	2	0,20	0,40
IC	2	1,00	2,00
Transistoren	2	0,50	1,00
Leistungstr.	2	1,00	2,00
Gesamt			7,60

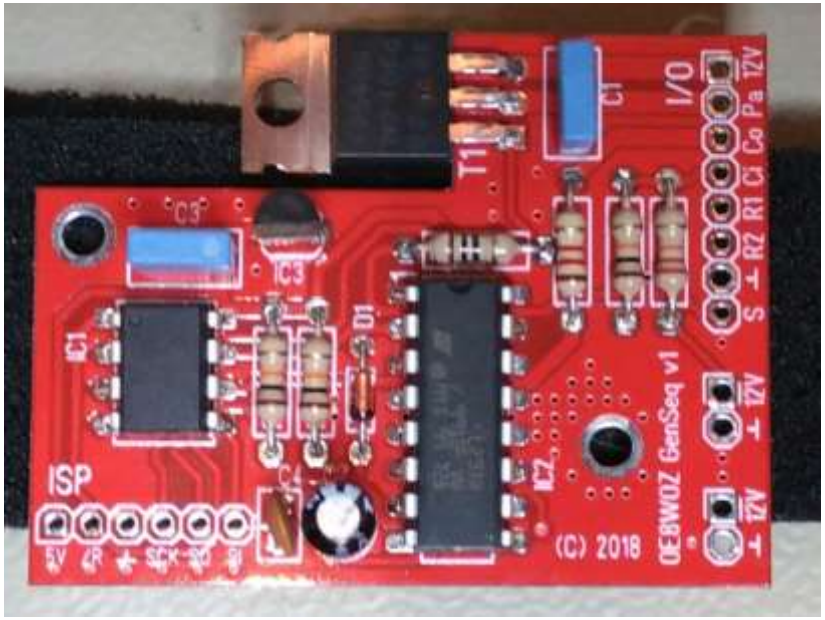
Stückzahlen und Tagespreise/Angebote können den Preis stark beeinflussen – daher eher „qualitativ“ zu sehen.
PCB-Kosten, ggf. Kühlung, … in beiden Fällen sehr ähnlich (außer DK3HA ohne PA Ausgang), deshalb nicht mit eingerechnet.
Stiftleisten sind im GenSeqV1 möglich (auch THT zur einfachen Verwendung), bei DK3HA/DB6NT nicht vorgesehen daher nicht verglichen.

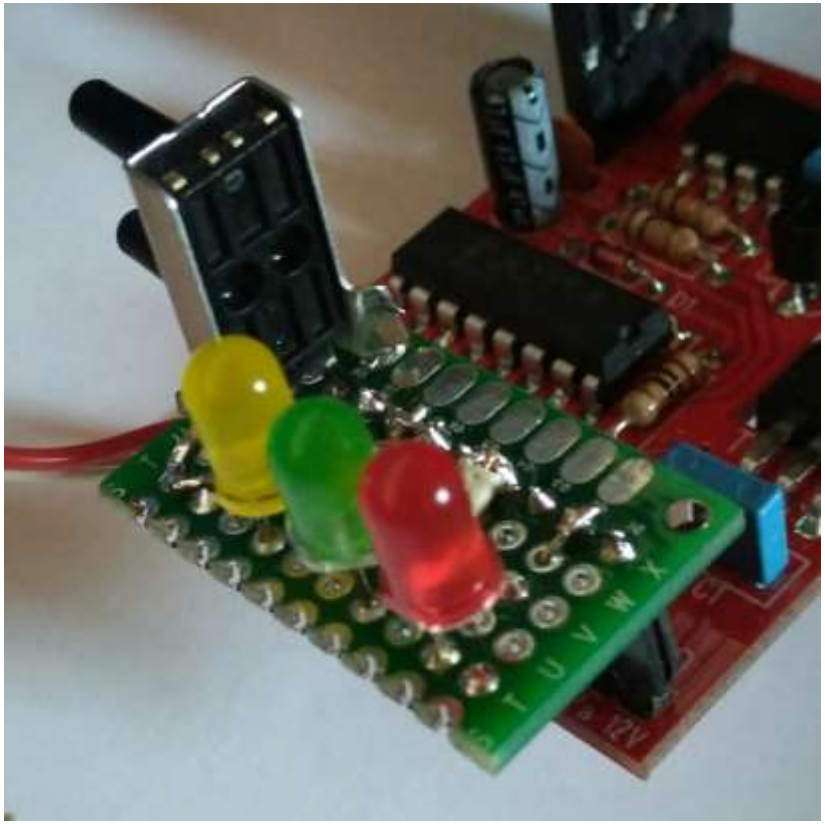
Generelle Aussage: „Digitalisierung“ bringt Flexibilität ohne echte Kosten-Nachteile !

Prinzip-Schaltung GenSeqV1



Aufbau und Konfiguration



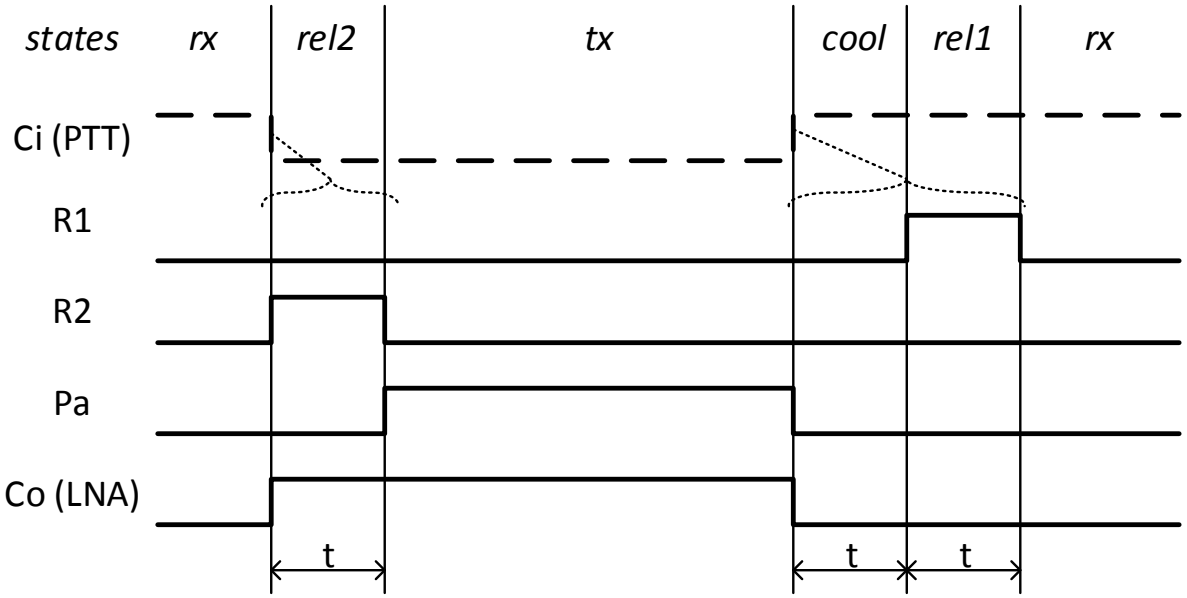


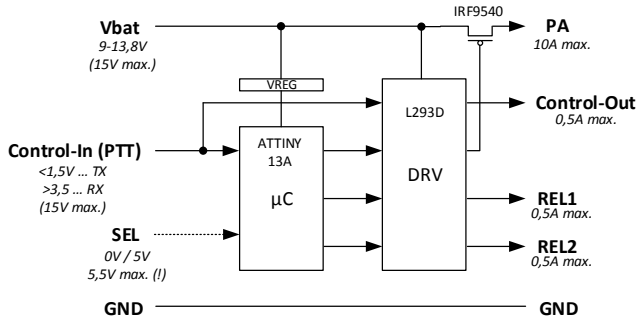
Vorprogrammierte ATTINY13A + PCB
sind verfügbar, zur Konfiguration selbst ist kein Programmiergerät nötig!

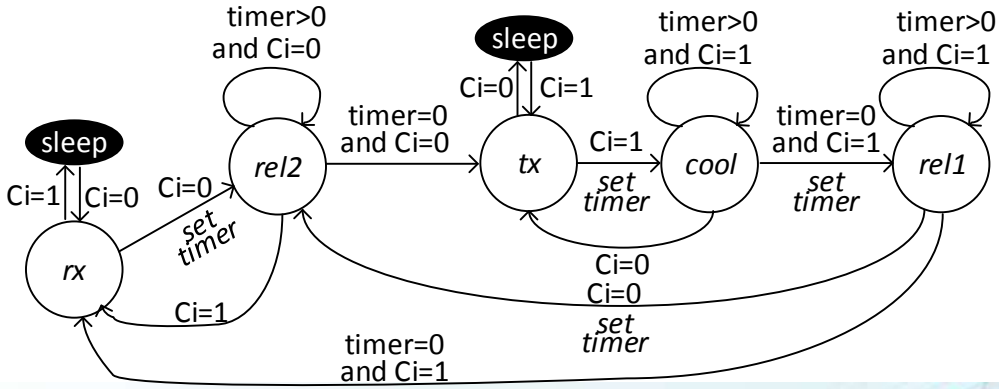
Soweit, so gut – für Anwender ….

… wenn da nicht die Möglichkeit wäre,
das Teil selbst zu programmieren!

Software – Zustandsmaschine







Software Implementierung

- https://www.microchip.com/mplab/avr-support/atmel-studio-7

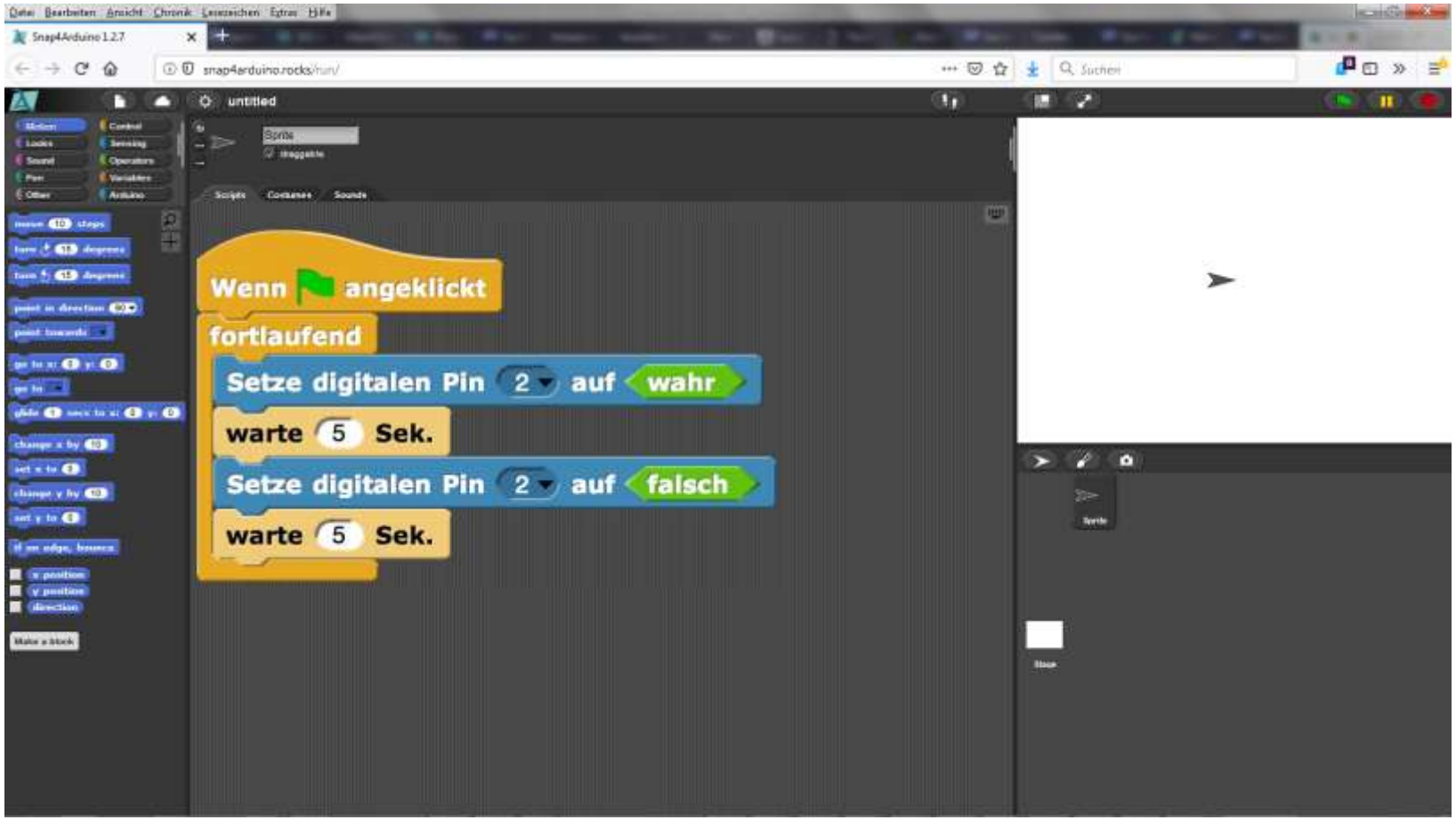
```
/* setup I/O */
PORTB = 0b00000000;   // set all pins to LOW
PORTB |= _BV(P_PA_N); // set PA pin HIGH (disable)
PORTB |= _BV(P_SEL);  // set SEL pin via pull-up
PORTB |= _BV(P_CTRL); // set CTRL pin via pull-up
invert = eeprom_read_byte(&eep_invert);
failsafe = eeprom_read_byte(&eep_failsafe);
if (failsafe) {
      PORTB |= _BV(P_REL2);  // set REL2 pin HIGH
} else if (invert) {
      PORTB |= _BV(P_REL2);  // set REL2 pin HIGH
      PORTB |= _BV(P_REL1);  // set REL1 pin HIGH
}
DDRB = 0b00000000;    // set all pins as INPUT
DDRB |= _BV(P_REL2);  // set REL2 pin as OUTPUT
DDRB |= _BV(P_REL1);  // set REL1 pin as OUTPUT
DDRB |= _BV(P_PA_N);  // set PA pin as OUTPUT

/* setup delay time */
base_time = 5000 >> (eeprom_read_byte(&eep_time)<<1);

/* initially set latching switch */
if (!failsafe) {
      /* disable PA and set RF switch to RX */
      if (invert)   SET_REL1i; else SET_REL1;
      for(uint16_t i = 0; i<base_time; ++i) _delay_us(1);
      if (invert)   CLRi; else CLR;
      seqstate=rx;
}
```

```
/* this interrupt will react on CTRL (PTT) changes */
PCMSK |= (1<<P_CTRL); // pin change interrupt mask
GIMSK |= _BV(PCIE);   // enable PCINTx
sei(); // enable global interrupts

/* done, now we wait on changes */
set_sleep_mode(SLEEP_MODE_PWR_DOWN);

/* main loop, just enter sleep if we wake up by an interrupt */
while (1)
{
      sleep_mode();   // go to sleep and wait for interrupt on CTRL (PTT)...
}
```

```
// sequencer variables
typedef enum { rx, rel2, tx, cool, rel1 } seqstate_t;
seqstate_t seqstate = rx;
uint8_t invert = 0;
uint8_t failsafe = 0;
uint16_t base_time = 5000;

// helper macros (incomplete)
#define CI_STATE (PINB >> P_CTRL) & 1
#define SEL_STATE (PINB >> P_SEL) & 1

#define SET_REL1 PORTB = (PORTB & (~_BV(P_REL2))) | _BV(P_PA_N) | _BV(P_REL1)
#define SET_REL2 PORTB = (PORTB & (~_BV(P_REL1))) | _BV(P_PA_N) | _BV(P_REL2)
#define SET_PA   PORTB = (PORTB & (~_BV(P_PA_N)) & (~_BV(P_REL1)) & (~_BV(P_REL2)))
#define CLR      PORTB = (PORTB & (~_BV(P_REL1)) & (~_BV(P_REL2))) | _BV(P_PA_N)

// PIN CHANGE INTERRPUT
ISR(PCINT0_vect)
{     uint8_t pinstat = CI_STATE; // 0...TX, 1...RX
      uint16_t timer=0;     // us

      // state machine handling PA, REL1 and REL2 outputs
      // we loop here until we reach a stable TX or RX state
      while(1) {
            switch (seqstate) {
                  case rx:
                        if (failsafe)
                              CLRf;
                        else if (invert) CLRi; else CLR;
                        if (pinstat) {
                              return;
                        } else { // (!pinstat)
                              timer=base_time;
                              seqstate=rel2;
                        }
                        break;
                  case rel2:
                        // (incomplete)
                        break;
                  case tx:
                        // (incomplete)
                        break;
                  case cool:
                        // (incomplete)
                        break;
                  case rel1:
                        if (failsafe)
                              CLRf;
                        else if (invert) SET_REL1i; else SET_REL1;
                        if (pinstat) {
                              if (timer) {
                                    --timer;
                              } else { // (!timer)
                                    seqstate=rx;
                              }
                        } else { // (!pinstat)
                              timer=base_time;
                              seqstate=rel2;
                        }
                        break;
            }
            pinstat = CI_STATE;
      }
}
```

Alternative für „Nicht-Programmierer“

http://snap4arduino.rocks/

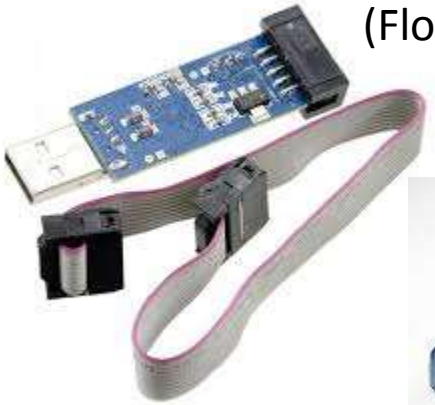


https://www.cs.uni-potsdam.de/~mprz/uploads/Anleitung_Snap4Arduino_MyIG_und_Grove.pdf

µC-Programmierung

von 5€	…	…	…	bis 500 €

ISP „Klon“
(Flohmarkt, Amazon, Aliexpress, …)



ISP Original

ICE Klon/Original

generisch
(PROFI-Tools)

Für Sparsame: der Klon reicht völlig aus und ist leicht zu beschaffen.
Man kann man auch einen Arduino verwenden (einfach mal „googeln“).

Danke für Eure Aufmerksamkeit!

- Weitere Informationen im nächsten **DUBUS 4/2018**

Kleines “Helferlein” für vielfältige Schaltaufgaben

von Wolfgang Scherr, OE8WOZ – wscherr@inode.at

Jeder Funkamateur ist vermutlich beim Zusammenbau eines Transverters über die Problematik

A little helper for various control tasks

by Wolfgang Scherr, OE8WOZ – wscherr@inode.at

Every radio amateur has probably come across the problem of controlling modules in a specific