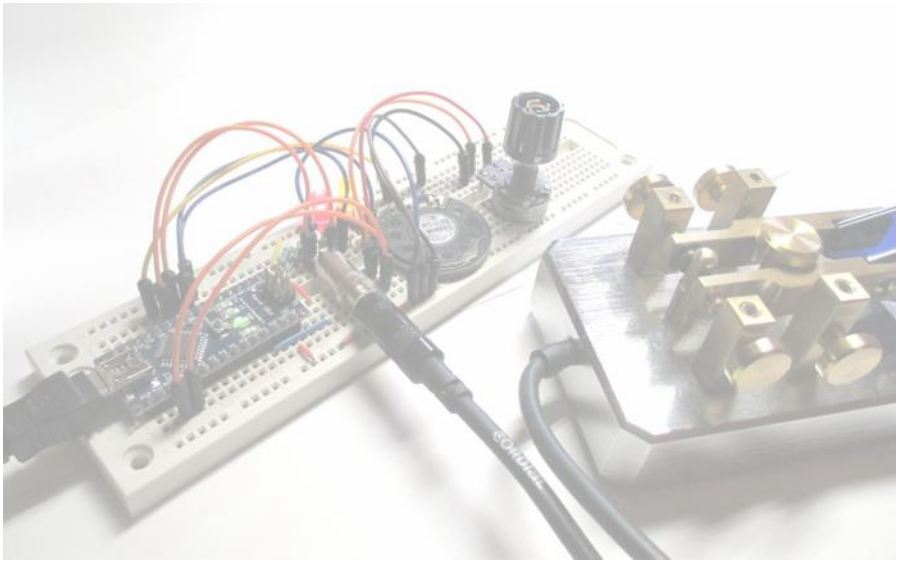
Elektronische Morsetaste mit ARDUINO

Eine Projektstudie



DK2JK 1/10

Motivation:

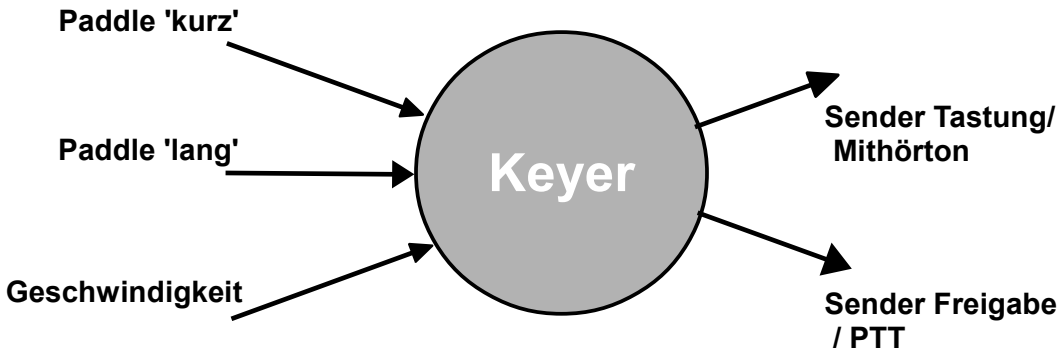
Darstellung einer *Ansatzes*, wie eine elektronische Morsetaste programmiert werden *könnte . Ein Experiment!*
Dies soll keine Bauanleitung sein.

Vorbilder:

- Diverse Keyer mit CMOS Technik [2][3]
- Veröffendlichungen in SPRAT „Occams Microcontroller“ [5]
- Tentec Rebel 506 [6]

Elektronische Morsetaste mit ARDUINO

Übersicht

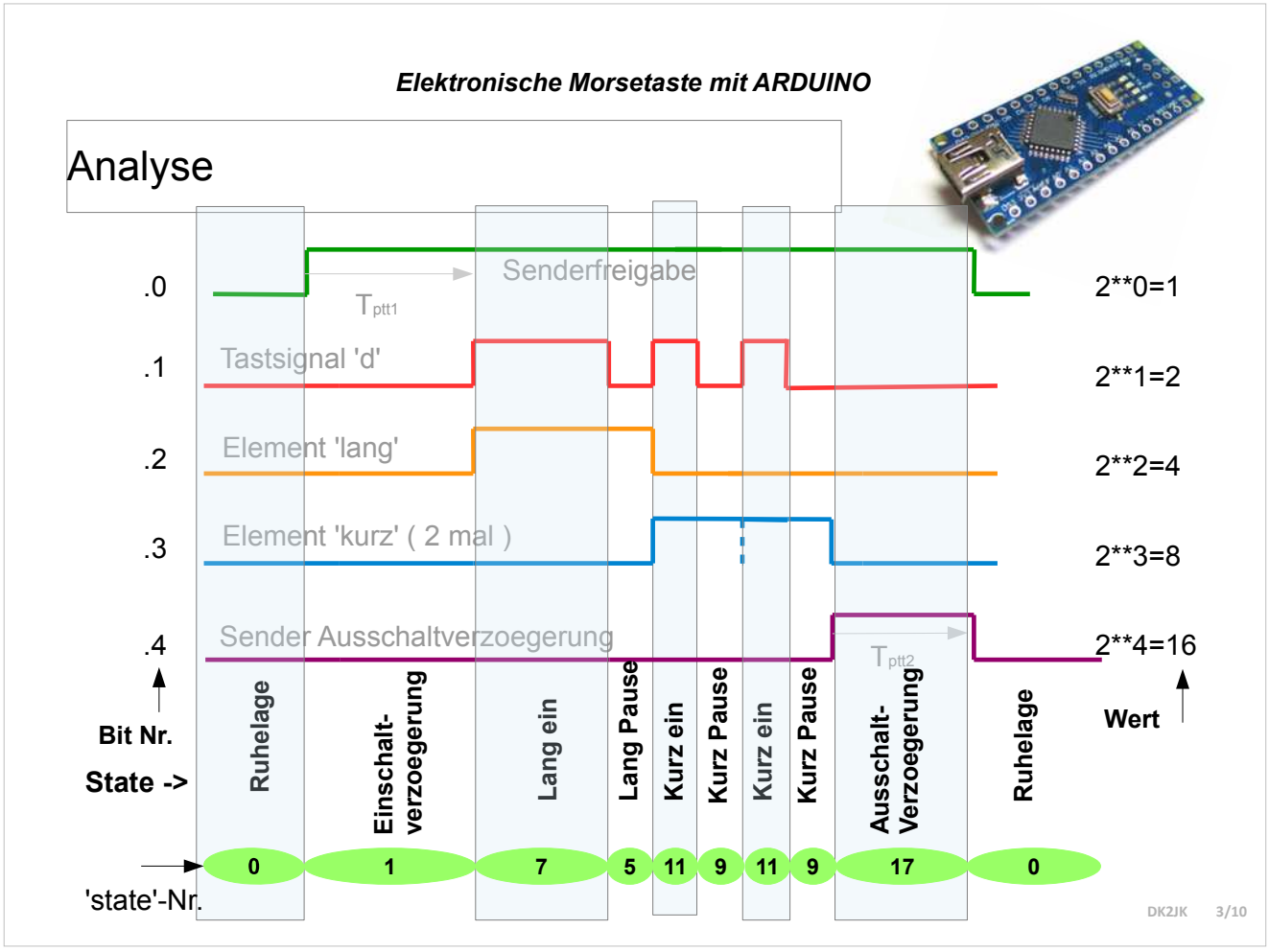


DK2JK 2/10

Hier ist die elektronische Morsetaste (kurz 'Keyer') als 'Blackbox' dargestellt. Das Ziel ist, möglichst viele Funktionen durch einen Prozessor zu erledigen (Hier ATMEL / Arduino).

Es sollen nur die Grundfunktionen des Keyers mit Squeezetechnik realisiert werden; also keine Speichertexte etc.

Im Folgenden wird gezeigt, wie die Software des Keyers entworfen wurde.



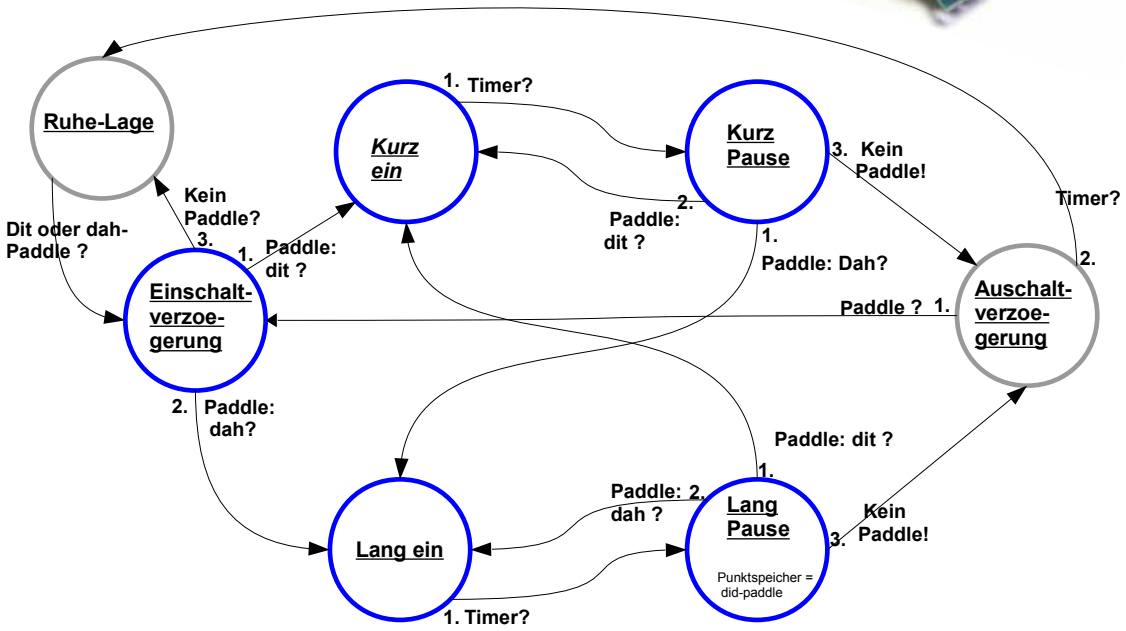
Die Programmierung beginnt mit der Analyse.
Das Morsezeichen (rot) wird in Elemente 'Lang' und 'Kurz' eingeteilt: Lang (orange) dauert 4 Zeiteinheiten (3 Ein + 1 Pause); Kurz(blau) dauert 2 Zeiteinheiten (1 Ein + 1 Pause).
Die Sendefreigabe (grün) besteht aus Verzögerung am Anfang, dem oder den Morsezeichen und einer Verzögerung am Ende.
Die Zeitabschnitte werden mit Namen versehen('Zustände', engl. 'state'):

Ruhelage
Einschaltverzögerung
Lang_Ein
Lang_Pause
Kurz_Ein
Kurz_Pause
Ausschaltverzögerung

Um die Zustände einfacher dekodieren zu können, wird für jedes Signal bzw. Element ein Bit in einem Byte verwendet; die Summe der Bitwerte ergibt die Zustandsnummer.

Elektronische Morsetaste mit ARDUINO

Zustandsdiagramm



DK2JK 4/10

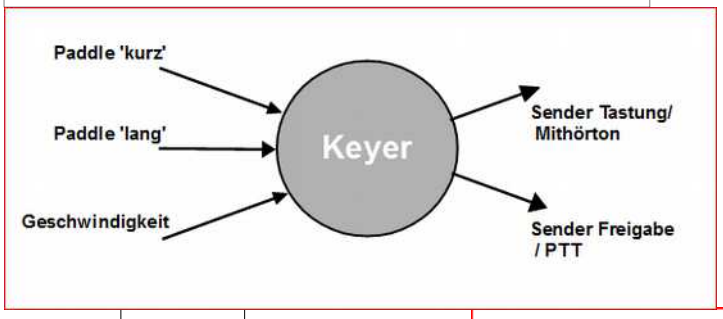
Mit den definierten Zuständen wird das Zustandsdiagramm ('statechart') gezeichnet.
Dabei gelten die Regeln:

1. Jeder Zustand wird durch einen Kreis dargestellt.
2. Die Pfeile zwischen den Zuständen stellen die Ereignisse ('Events') dar.
3. Ein Zustand wir nur bei Auftreten eines Ereignisses verlassen. Hier speziell wird immer erst auf das Timer- Ereignis gewartet. Solange der Timer in einem Zustand noch nicht abgelaufen ist, kann der Prozessor was anderes tun.
4. Der Folgezustand wird durch Abfragen bestimmt; die Abfragen erfolgen in der nummerierten Reihenfolge.

Bei LangEin und KurzEin wird der Folgezustand nur durch den Timer bestimmt; in den anderen Fällen auch durch die Stellung des Paddles.

Elektronische Morsetaste mit ARDUINO

Programm (1)



Entwurf

Keyer.h (Headerdatei)

```
public:
  Keyer(  unsigned char wpm = 18,
          unsigned char delay_start = 50,
          unsigned char delay_end = 500);
  void run();
  void update_wpm(unsigned char wpm);
  bool txEin();
  bool txFreigabe();
```

```
inline static bool dit_paddle() ;
inline static bool dah_paddle();
```

DK2JK 5/10

Die Programmiersprache ist C++ (=Objekt orientiert)

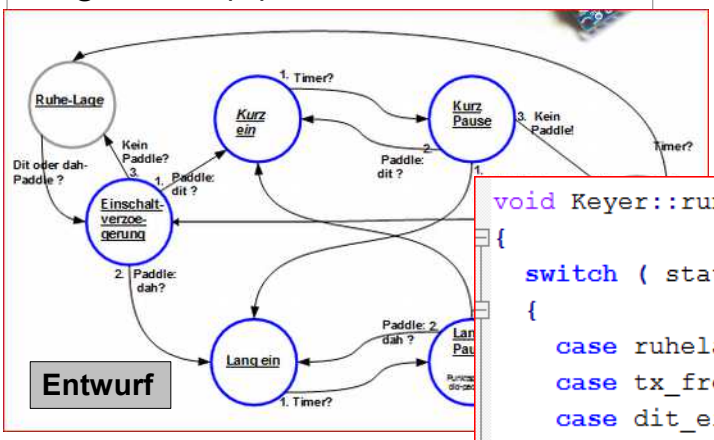
Die Schnittstellen des Objekts 'Keyer'
stehen in der Headerdatei 'keyer.h' unter 'public:'.

Die Initialisierung des Objekts 'Keyer' stellt standardmäßig 18 WpM, 50ms Anfangsverzögerung und 500ms Verzögerung am Ende ein. Hier gehört streng genommen noch die Übergabe der Pinnummern für die Paddle hin; dies wird hier durch zwei Inline-Funktionen erledigt; das ist kürzer.

Die Methode Keyer.run() muss zyklisch aufgerufen werden. Mit Keyer.update.wpm(..) wird die Geschwindigkeit verstellt. Keyer.txEin() und Keyer.txFreigabe() fragen die Ausgänge des Objekts Keyer ab.

Elektronische Morsetaste mit ARDUINO

Programm (2)




```
void Keyer::run()
{
  switch ( state)
  {
    case ruhelage:          {...}    break;
    case tx_freigabe:       {...}    break;
    case dit_ein:           {...}    break;
    case dit_zwischenraum:  {...}    break;
    case dah_ein:           {...}    break;
    case dah_zwischenraum:  {...}    break;
    case tx_delay:          {...}    break;
  }
}
```

Function Keyer.run()

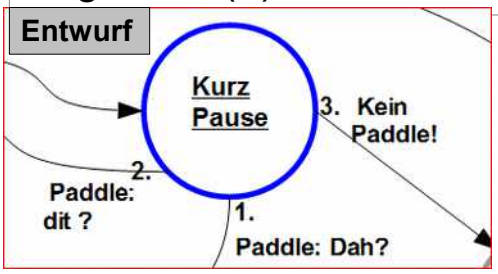
DK2JK 6/10

Das Zustandsdiagramm wird im Programm durch einen Switch-Block abgebildet. Jedem Zustand im Zustandsdiagramm ist im Programm ein 'Case' zugeordnet. Es wird beim Aufruf nur ein Case ausgeführt. In den Programmzeilen in {…} werden die Abfragen durchgeführt, die im Zustandsdiagramm als Pfeile dargestellt sind.

Elektronische Morsetaste mit ARDUINO

Programm (3)

Entwurf



Programm Fragment

```
case dit_zwischenraum:
  /* sender ist aus fuer eine zeiteinheit
   * dadurch, dass jetzt dah-taste zuerst abgefragt wird,
   * kommen beim druecken beider tasten wechselnd dits und dahs heraus
   */
  if ( timer.event() )
  {
    if (dah_paddle()) {       // ereignis 1: dah-taste gedrueckt
      timer.interval( 3 * t_dit );
      state = dah_ein;
    }
    else  if (dit_paddle()) { // ereignis 2: dit-taste gedrueckt
      timer.interval(t_dit);
      state = dit_ein;
    }
    else {                    // ereignis 3: keine taste gedrückt
      timer.interval(t_tx_delay_end);
      state = tx_delay;
    }
  }
  break;
```

DK2JK 7/10

Als Beispiel wird der Inhalt des 'switch-case' 'dit_zwischenraum' gezeigt, welches unserem Zustand 'kurz Pause' entspricht:

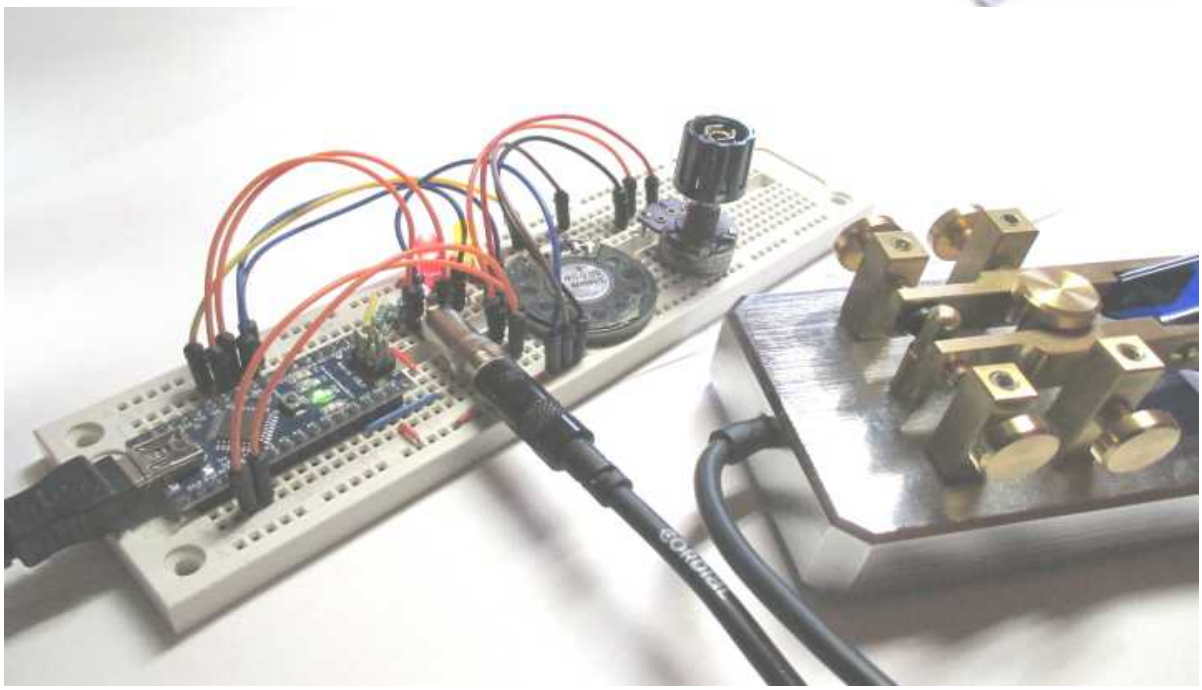
Nur wenn der Timer- Event kommt (*Abrage 'If(timer.event() '*) - nämlich nach Ablauf der Zeit Tdit - , hat das Programm in diesem Zustand etwas zu tun.

Bei Ablauf des Timers gibt folgende Möglichkeiten:

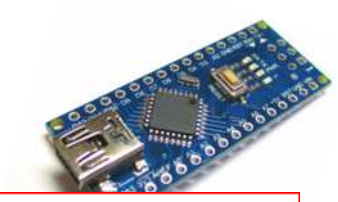
1. 'dah_paddle() gedrückt
2. 'dit_paddle() gedrückt
3. kein Paddle gedrückt

Je nach Ereignis wird der Timer mit einem neuen Wert gestartet und der nächste Zustand eingestellt. Im nächsten Durchlauf von Keyer.run() geht es dann in einem anderen 'switch-case' weiter.

Testaufbau



Elektronische Morsetaste mit ARDUINO



Hauptprogramm

Hauptprogramm:

```
void loop() {
  ////////////// keyer /////////////////////
 keyer.run();                                  //  keyer aktionen
  digitalWrite(PIN_TX_ENABLE, keyer.txFreigabe()); //  tx freigabe schalten
  digitalWrite(PIN_TX_KEY, keyer.txEin());        //  tx schalten
  if(  keyer.txEin() )   {     tone(PIN_SIDETONE, 650) ;   }
  else                   {     noTone(PIN_SIDETONE);       }// mithoerton
}
```

Verstellung der Geschwindigkeit:

```
int x = analogRead( A0);
int wpm = map  ( x, 0,1024, 10,30);
keyer.update( wpm);
```

Option Decoder:

```
// decoder auf serielle schnittstelle
    decoder.decode(keyer.txEin());  // decoder arbeiten lassen
    if ( decoder.available() )      // hat er was erkannt ?
    { char x = decoder.read();      // dann lesen
      Serial.print(x);              // und auf UART ausgeben
    }
```

DK2JK 9/10

Jetzt wird das entworfene Programm benutzt. Im Hauptprogramm ('loop') wird 'keyer.run()' zyklisch aufgerufen. Weiterhin werden die Ausgangsdaten 'keyer.txFreigabe()' und 'keyer.txEin' ausgewertet. Im Beispiel wird zusätzlich zur Tastung ein Mithörton erzeugt, z.B. für Transceiver mit nur einem DDS- Oszillator.

Die Geschwindigkeit wird am einfachsten über ein Poti am Analog- Digital-Converter des Prozessors eingestellt (im Beispiel weggelassen):

```
int x = analogRead( A0);
int wpm = map  ( x, 0,1024, 10,30);
keyer.update( wpm);
```

Als Option kann ein Decoder nachgeschaltet werden , der die gegebenen Zeichen anzeigt.

Referenzen

[1] http://www.qsl.net/dk5ke/squeezen.html
„ Die vollautomatische Squeeze-Taste“

[2] Ted Theroux, N9BQ : „A Digital CMOS IAMBIC Keyer“ Jun 82 QST
(Hier wird ein Hardware- State- Counter verwendet)

[3] http://dk2jk.darc.de/arduino/keyer/oldies/
„CMOS Keyer 1982“

[4] http://m0xpd.blogspot.co.uk/2013/02/keyerduino.html
„KEYERduino“

[5] http://m0xpd.blogspot.co.uk/p/occams-microcontroller.html
„Occams Microcontroller“ SPRAT 156, Autumn 2013

[6] http://www.rkrdesignsllc.com/products/transceivers-receivers/506-rebel-open-source-qrp-transceiver/
„506 Rebel - Open Source Arduino-based CW QRP Transceiver“

[7] http://dk2jk.darc.de/arduino/keyer/
„*Elektronische Morsetaste mit ARDUINO“ (dieses Dokument)*