TECHNISCHE HOCHSCHULE
OSTWESTFALEN-LIPPE
UNIVERSITY OF
APPLIED SCIENCES
AND ARTS

DDS Signalgenerator

Vertiefungspraktikum

10.12.21 | Hannes Krause und Tim Fischer

Überblick

- **Aufgabenstellung & Herausforderungen**
- **Hintergrundwissen**
- **Hardware**
- **Aufbau**
- **Arduino Programm**
- **Remote Programm**
- **Vorführung**
- **Messung und Fehleranalyse**
- **Ausblick**
- **Quellen**

Aufgabenstellung & Herausforderungen

Aufgabenstellung

- **Softwareentwurf DDS-Generator**
- **Festfrequenz**
- **Lineares & logarithmisches Wobbeln**
- **Einstellung aller Frequenzen in 10Hz-Schritten**
- **Menü**
- **Fernsteuerung über PC**

Herausforderungen

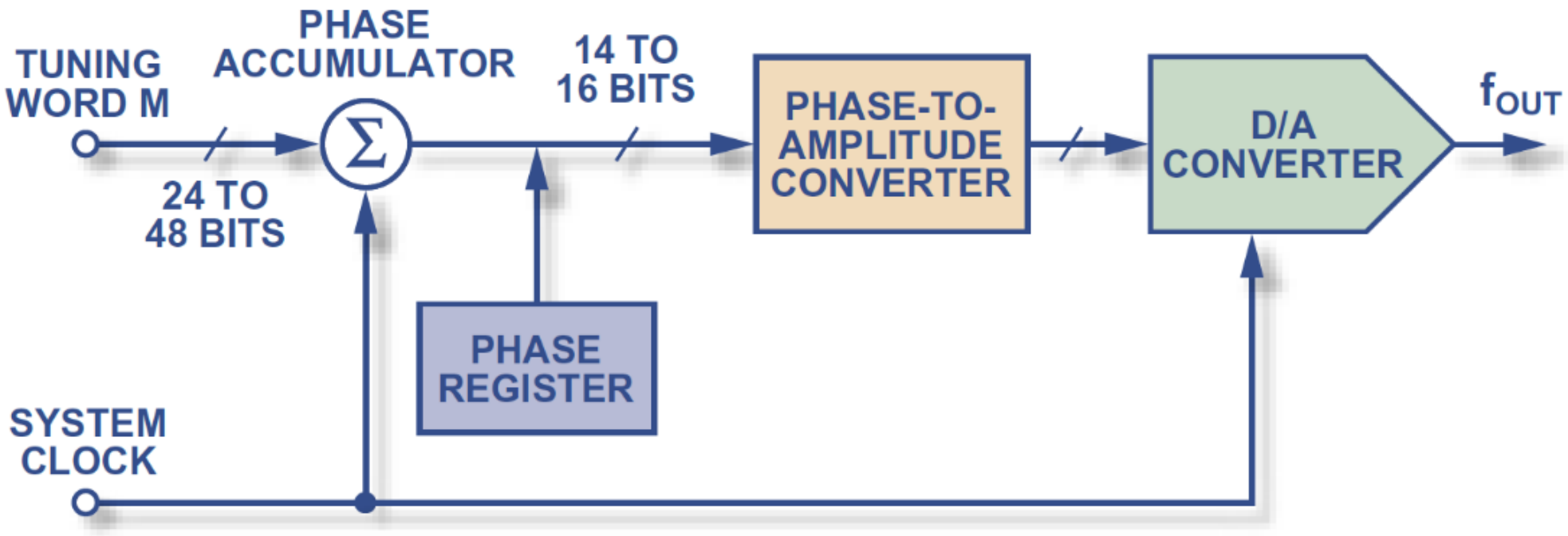
- **Display mit zwei Zeilen stellt hohe Anforderungen an Menü**
- **Begrenzte Eingabemöglichkeit (Rotary Encoder + 3 Taster)**
- **Anforderung von linearem und logarithmischem Frequenzlauf**

Hintergrundwissen

Was ist ein DDS-Generator?

- **DDS = Direct Digital Synthesis/Direct Digital Synthesizer**
- **Verfahren zur Erzeugung analoger Wellenformen**
- **Digital erzeugtes, zeitveränderliches Signal mittels D/A-Wandler in analoge Ausgangsgröße umgewandelt**
- **DDS & D/A-Wandler auf einem Chip: Complete DDS**
- **Referenztakt & Steuerwort benötigt**

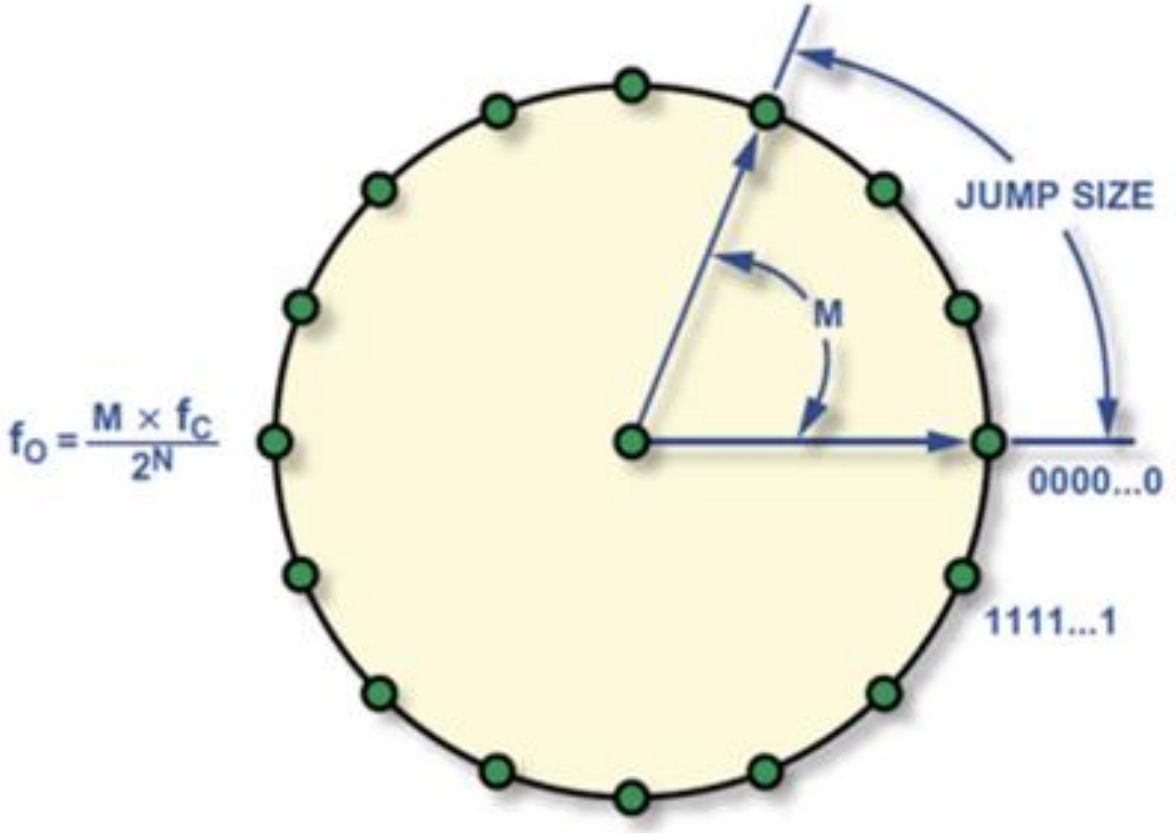
Was ist ein DDS-Generator?



Was ist ein DDS-Generator?

- **Akkumulator addiert pro Taktzyklus Wert des Steuerworts auf aktuellen Akkumulator-Wert**
- **Bei Überlauf Rücksetzen auf Null**
- **→sägezahnförmiger Verlauf des Wertes**
- **→kann als diskretes Maß für Phasenwinkel eines Sinus betrachtet werden**

Was ist ein DDS-Generator?

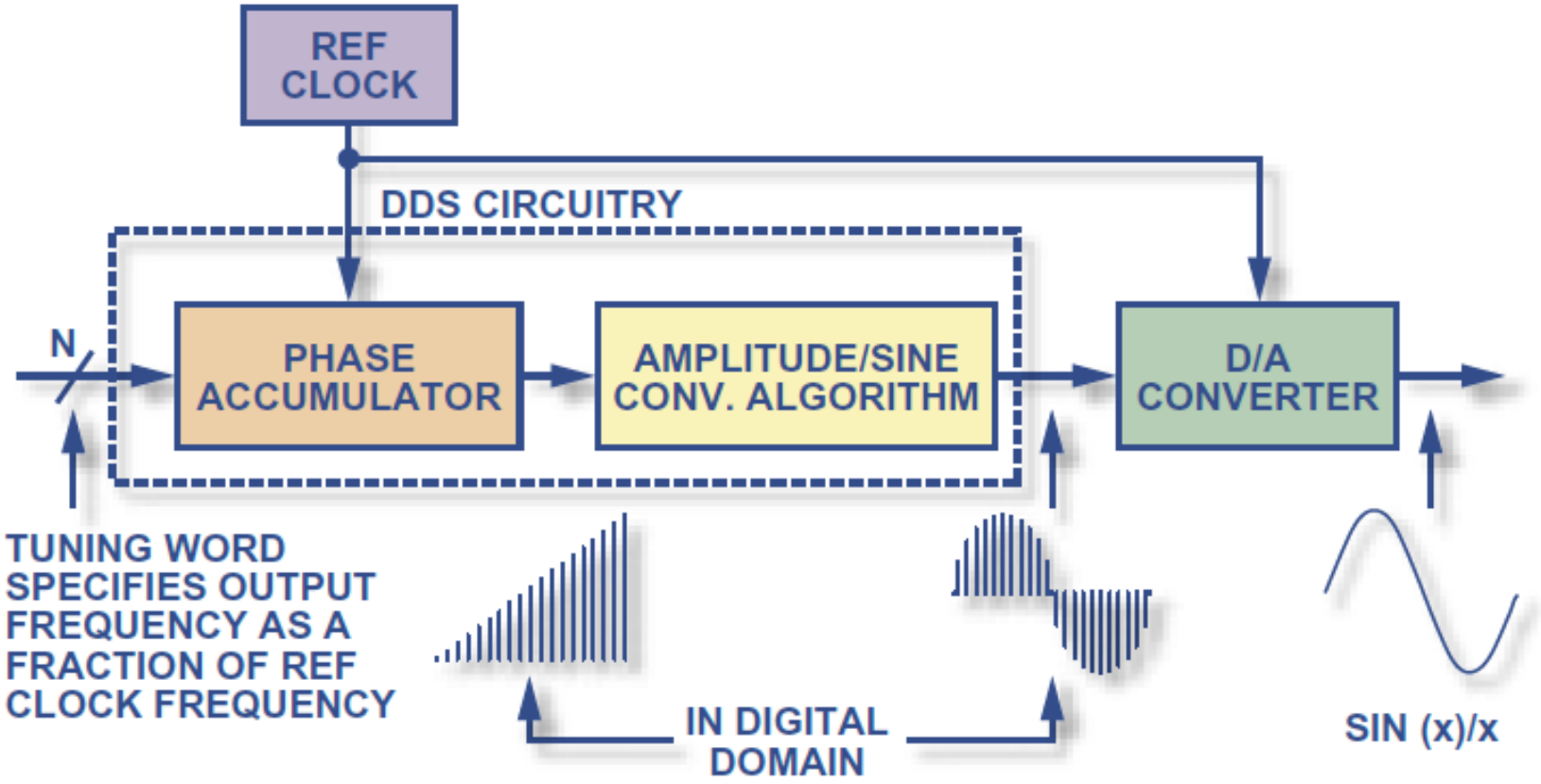


n	NUMBER OF POINTS
8	256
12	4096
16	65535
20	1048576
24	16777216
28	268435456
32	4294967296
48	281474976710656

Was ist ein DDS-Generator?

- $f_{out} = \frac{M * f_c}{2^n}$
- **Nach Nyquist-Shannon gilt:** $f_{out,max} = \frac{f_c}{2}$
- **tatsächliche Maximalfrequenz deutlich kleiner, da dann bessere Signalgüte & Weiterverarbeitung möglich**
- **Zuordnung von Funktionswerten zu Phasenwerten über Lookup-Table oder Algorithmen (z.B. CORDIC)**
- **Erzeugen des quasi-zeitkontinuierlichen , sinusförmigen Ausgangssignals durch den D/A-Wandler**

Was ist ein DDS-Generator?



Was ist ein DDS-Generator?

- **Kompakte & kostengünstige Herstellung**
- **Vollst. digital programmierbar → simple Hardware**
- **Frequenzauflösung im Microhertz-Bereich**
- **Extrem schnelle, exakte Frequenzänderung**
- **Qualitativ hochwertiges Ausgangssignal mit großem Frequenzbereich**
- **Geringer Leistungsbedarf**

Frequenzlauf

- **Auch Sweep, Chirp oder Wobbel genannt**
- **Wechselsignal konstanter Amplitude**
- **Frequenz ändert sich zeitlich periodisch zwischen Start- und Endwert**
- **Änderung in unterschiedlicher Form möglich**

Linearer Frequenzlauf

- **Lineare Frequenzänderung vom Start- zum Endwert:**

$$f(t) = ct + f_0$$

- **Mit Frequenzänderung c:**

$$c = \frac{f_1 - f_0}{T}$$

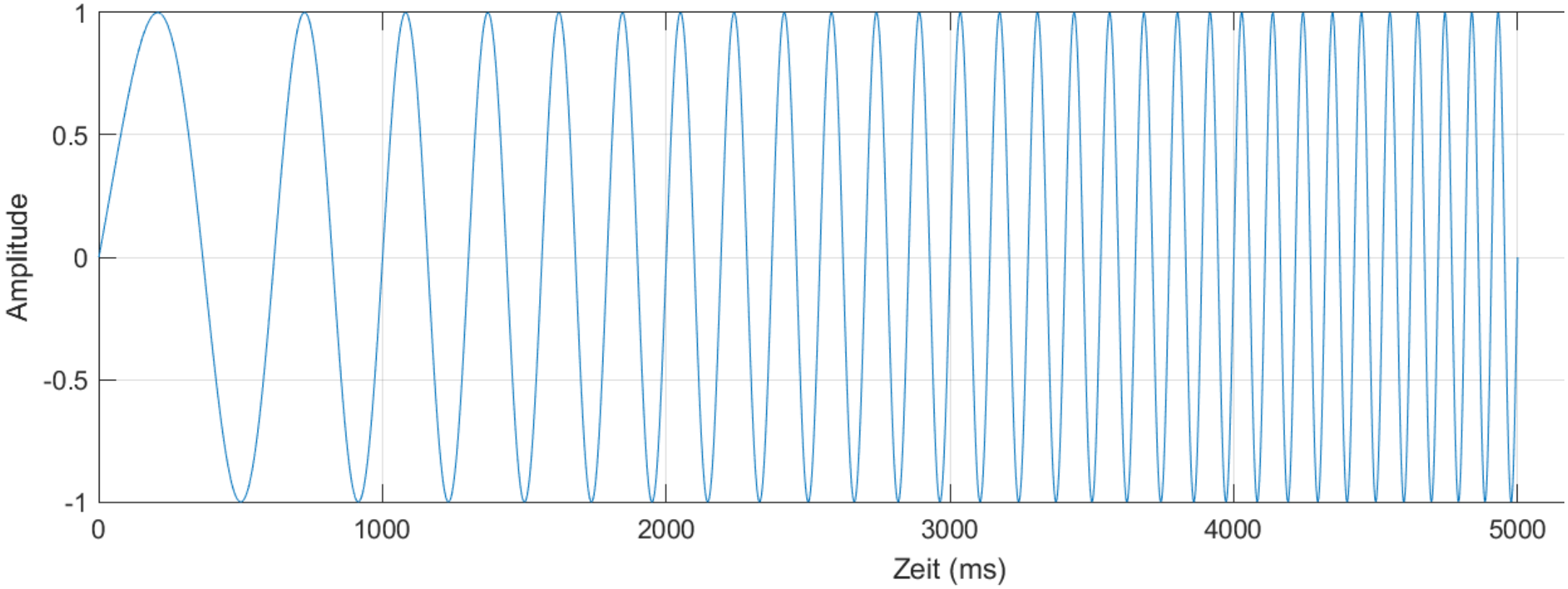
- **Integration der Frequenz liefert Phase:**

$$\varphi(t) = \varphi_0 + 2\pi \int_0^t f(\tau)\, d\tau$$

- **Zeitfunktion des linear gesweepten Signals:**

$$x(t) = \sin[\varphi(t)\,] = \sin\left[\varphi_0 + 2\pi\left(\frac{c}{2} * t^2 + f_0 t\right)\right]$$

Linearer Frequenzlauf



Logarithmischer Frequenzlauf

- **Lineare Frequenzänderung vom Start- zum Endwert:**

$$f(t) = f_0 k^t$$

- **Mit Basis k:**

$$k = \left(\frac{f_1}{f_0}\right)^{\frac{1}{T}}$$

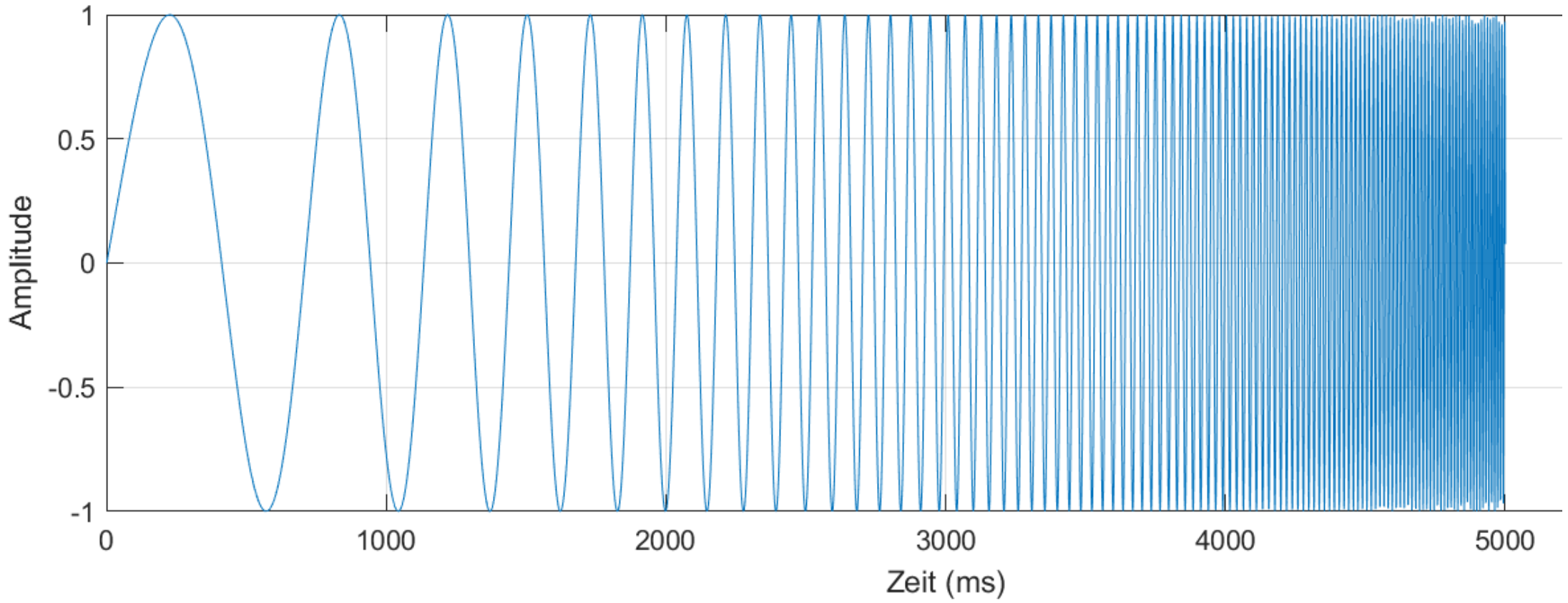
- **Integration der Frequenz liefert Phase:**

$$\varphi(t) = \varphi_0 + 2\pi f_0 \left(\frac{k^t - 1}{\ln(k)}\right)$$

- **Zeitfunktion des logarithmisch gesweepten Signals:**

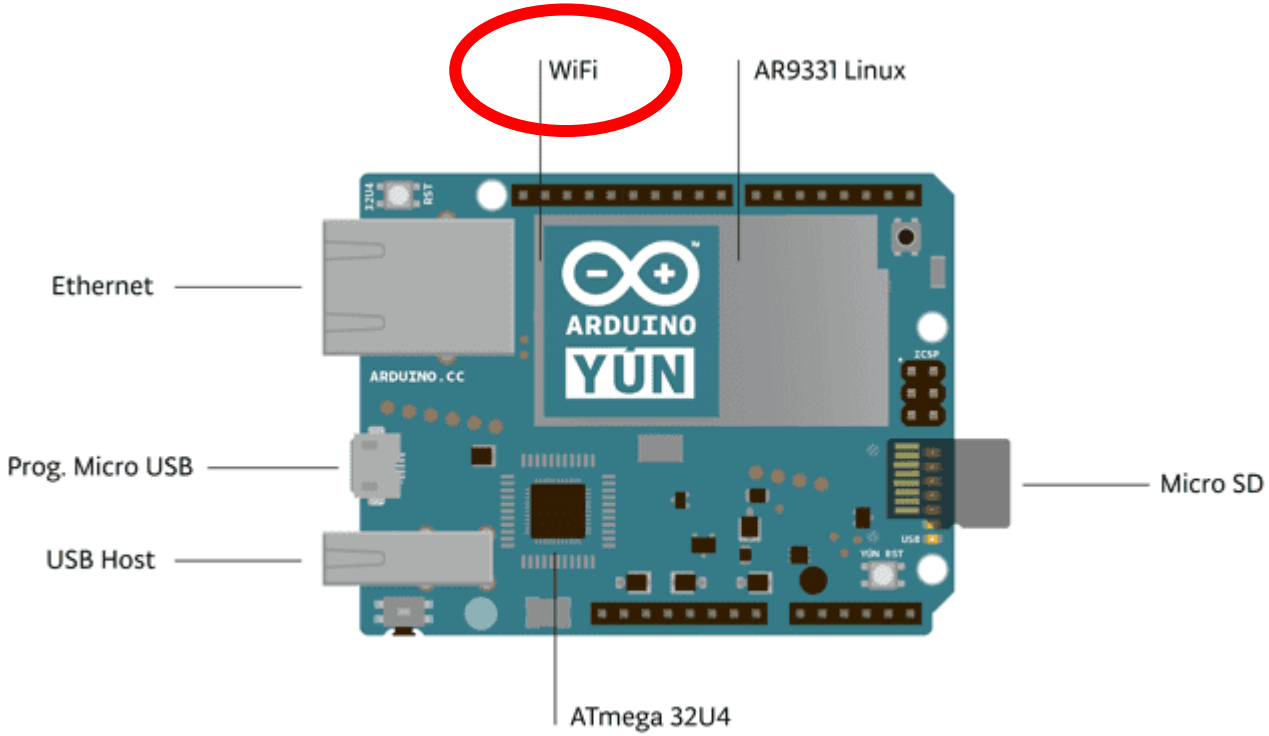
$$x(t) = \sin[\varphi(t)\,] = \sin\left[\varphi_0 + 2\pi f_0 \left(\frac{k^t - 1}{\ln(k)}\right)\right]$$

Logarithmischer Frequenzlauf

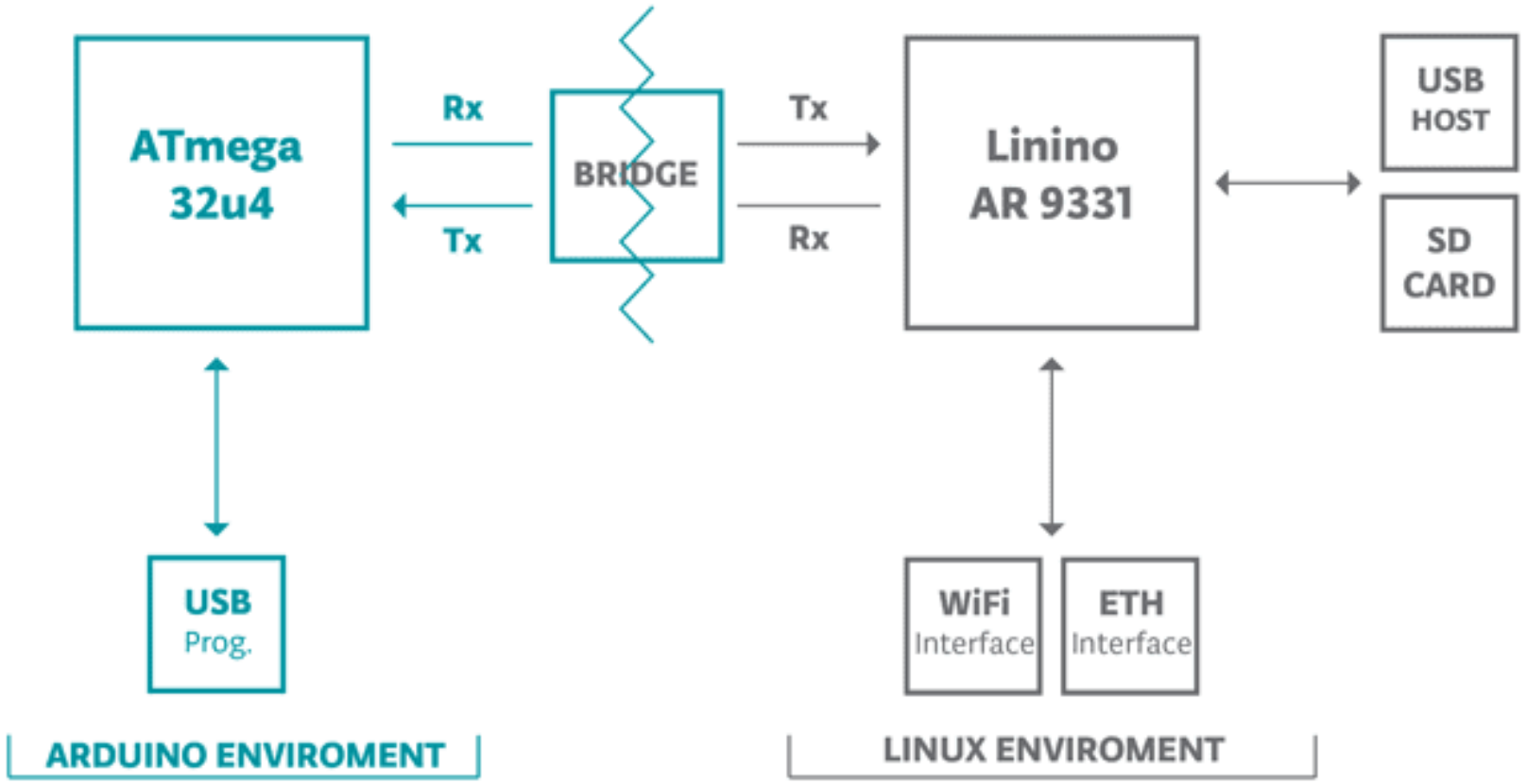


Verwendete Hardware

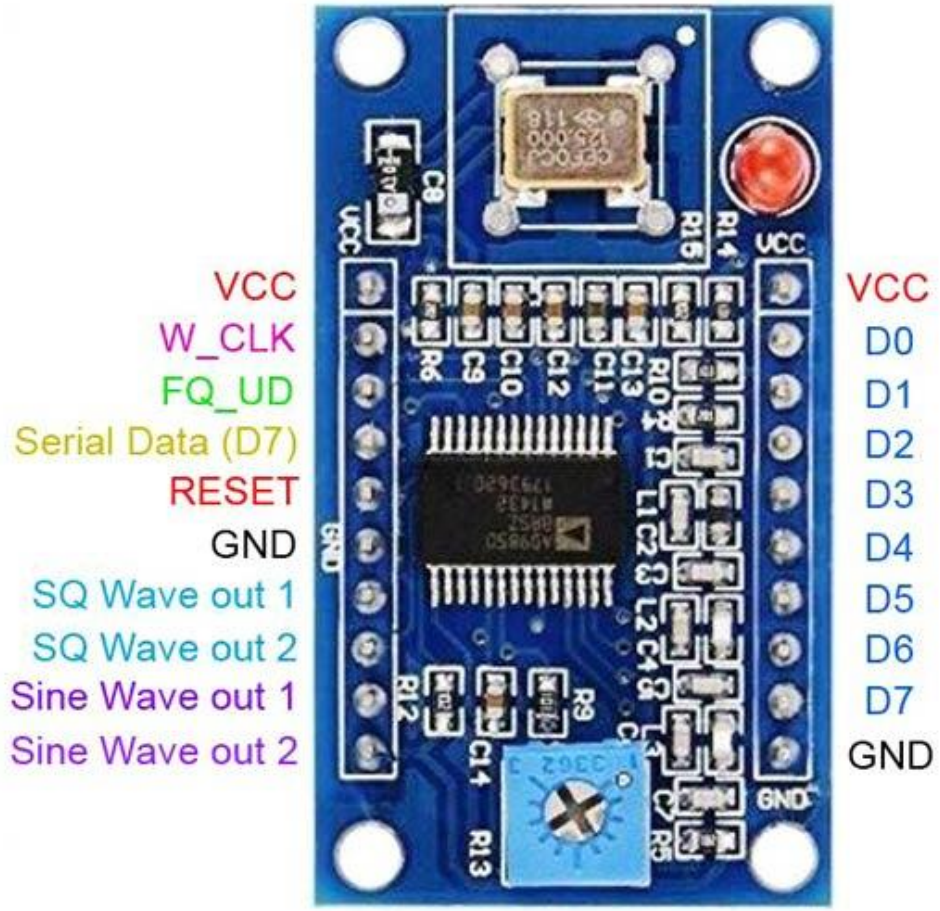
Arduino Yun


WiFi
AR9331 Linux
Ethernet
Prog. Micro USB
USB Host
Micro SD
ATmega 32U4
ARDUINO
YÚN
ARDUINO.CC

Arduino Yun


ATmega
32u4
Rx
Tx
BRIDGE
Tx
Rx
Linino
AR 9331
USB
HOST
SD
CARD
USB
Prog.
WiFi
Interface
ETH
Interface
ARDUINO ENVIROMENT
LINUX ENVIROMENT

ADS9850 (HC-SR08)



ADS9850 (HC-SR08)

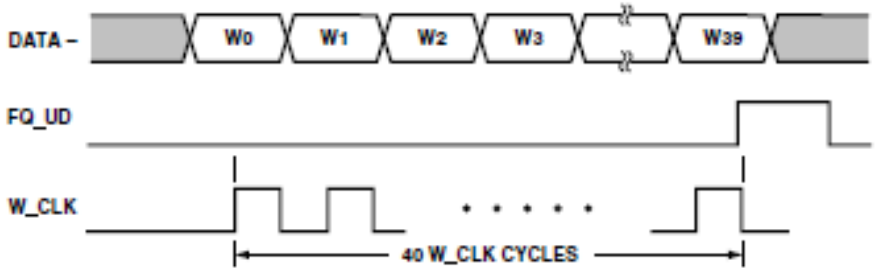
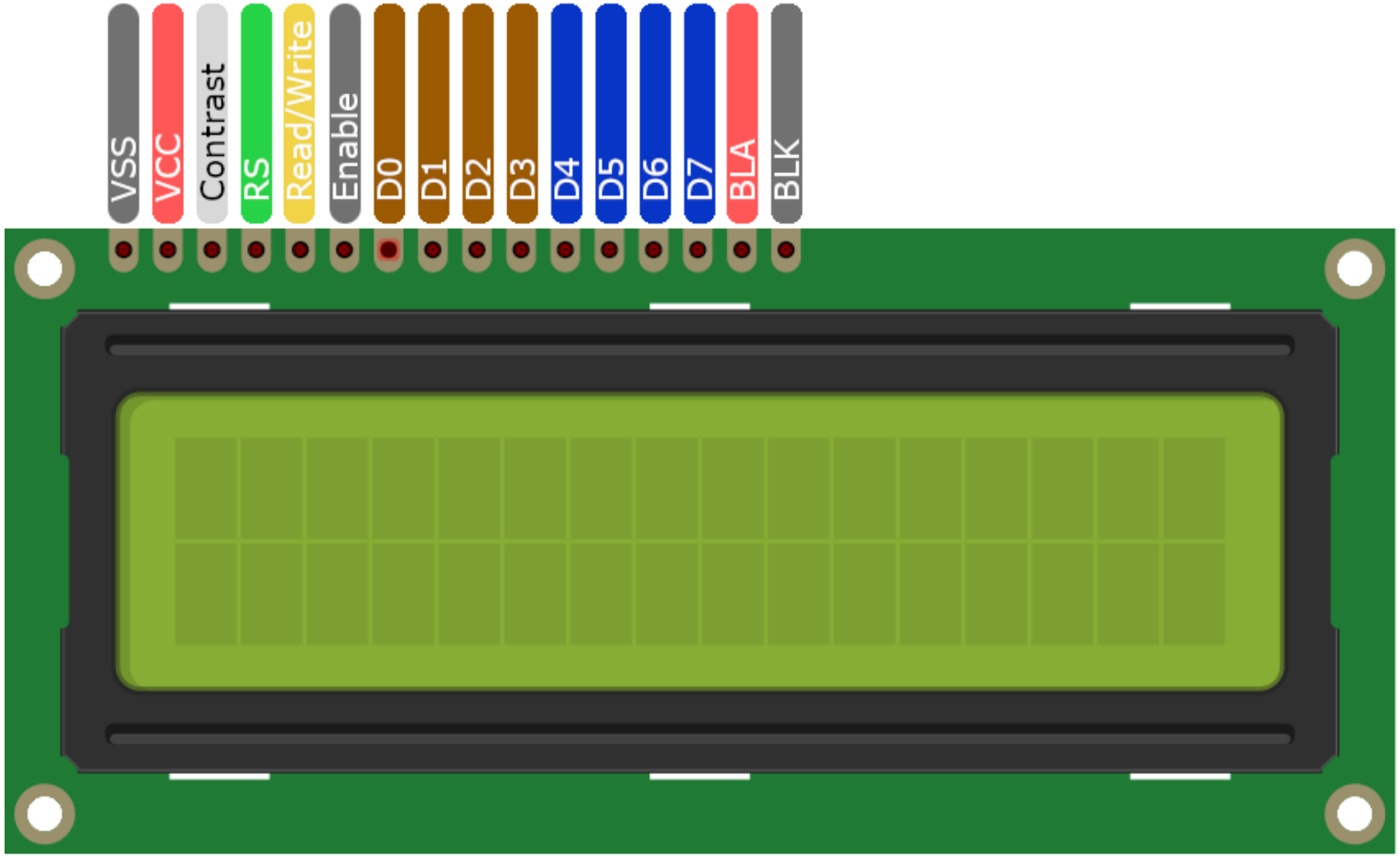


Figure 12. Serial Load Frequency/Phase Update Sequence

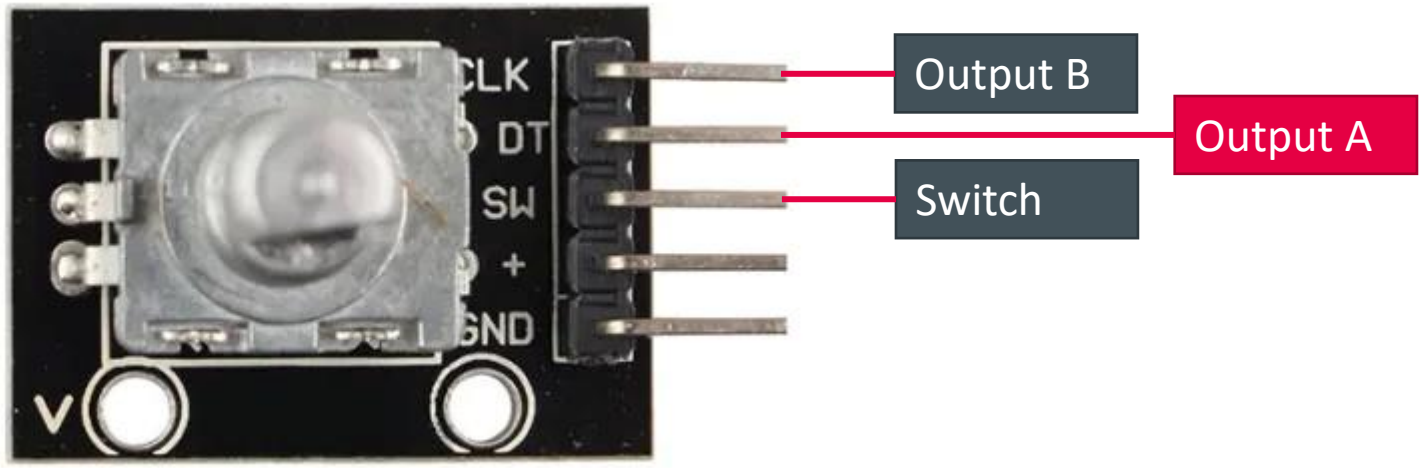
Table IV. 40-Bit Serial Load Word Function Assignment

W0	Freq-b0 (LSB)	W14	Freq-b14	W28	Freq-b28
W1	Freq-b1	W15	Freq-b15	W29	Freq-b29
W2	Freq-b2	W16	Freq-b16	W30	Freq-b30
W3	Freq-b3	W17	Freq-b17	W31	Freq-b31 (MSB)
W4	Freq-b4	W18	Freq-b18	W32	Control
W5	Freq-b5	W19	Freq-b19	W33	Control
W6	Freq-b6	W20	Freq-b20	W34	Power-Down
W7	Freq-b7	W21	Freq-b21	W35	Phase-b0 (LSB)
W8	Freq-b8	W22	Freq-b22	W36	Phase-b1
W9	Freq-b9	W23	Freq-b23	W37	Phase-b2
W10	Freq-b10	W24	Freq-b24	W38	Phase-b3
W11	Freq-b11	W25	Freq-b25	W39	Phase-b4 (MSB)
W12	Freq-b12	W26	Freq-b26		
W13	Freq-b13	W27	Freq-b27		

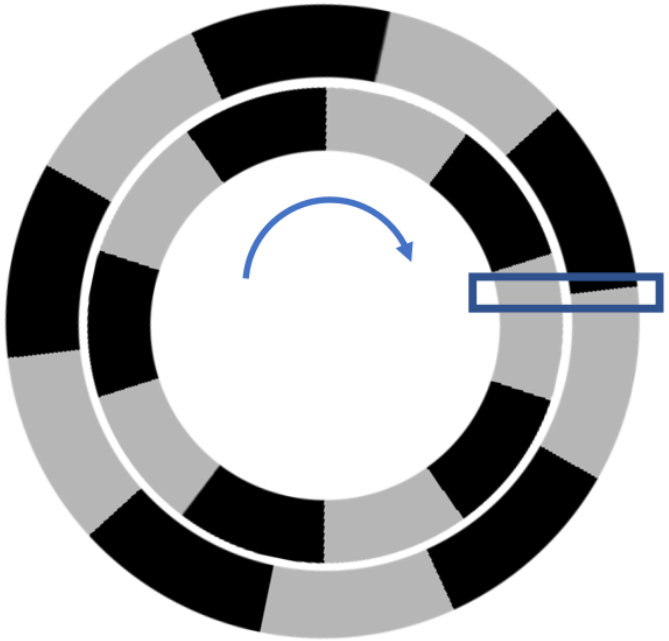
LCD mit HD44780-Controller



Drehinkrementalgeber

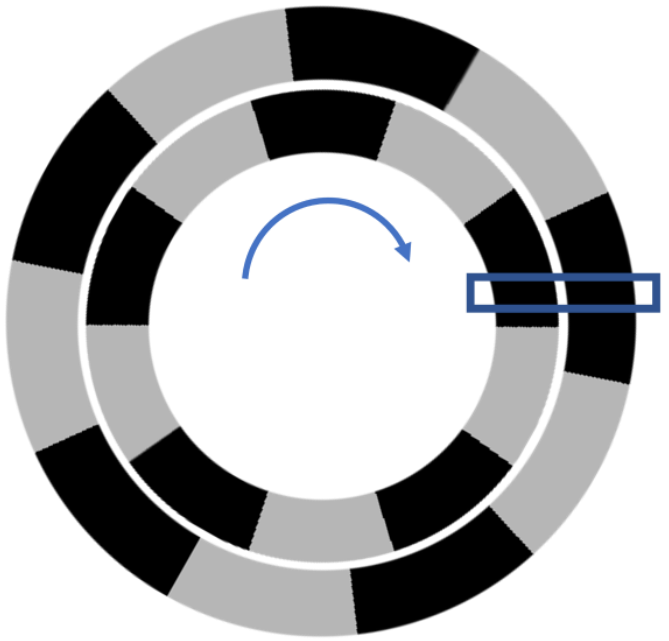


Drehinkrementalgeber



Innen: A
Außen: B

A	B
0	1

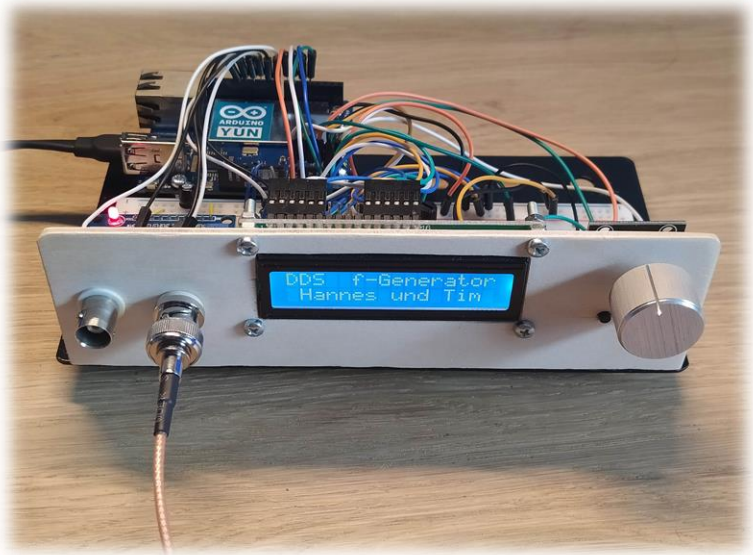


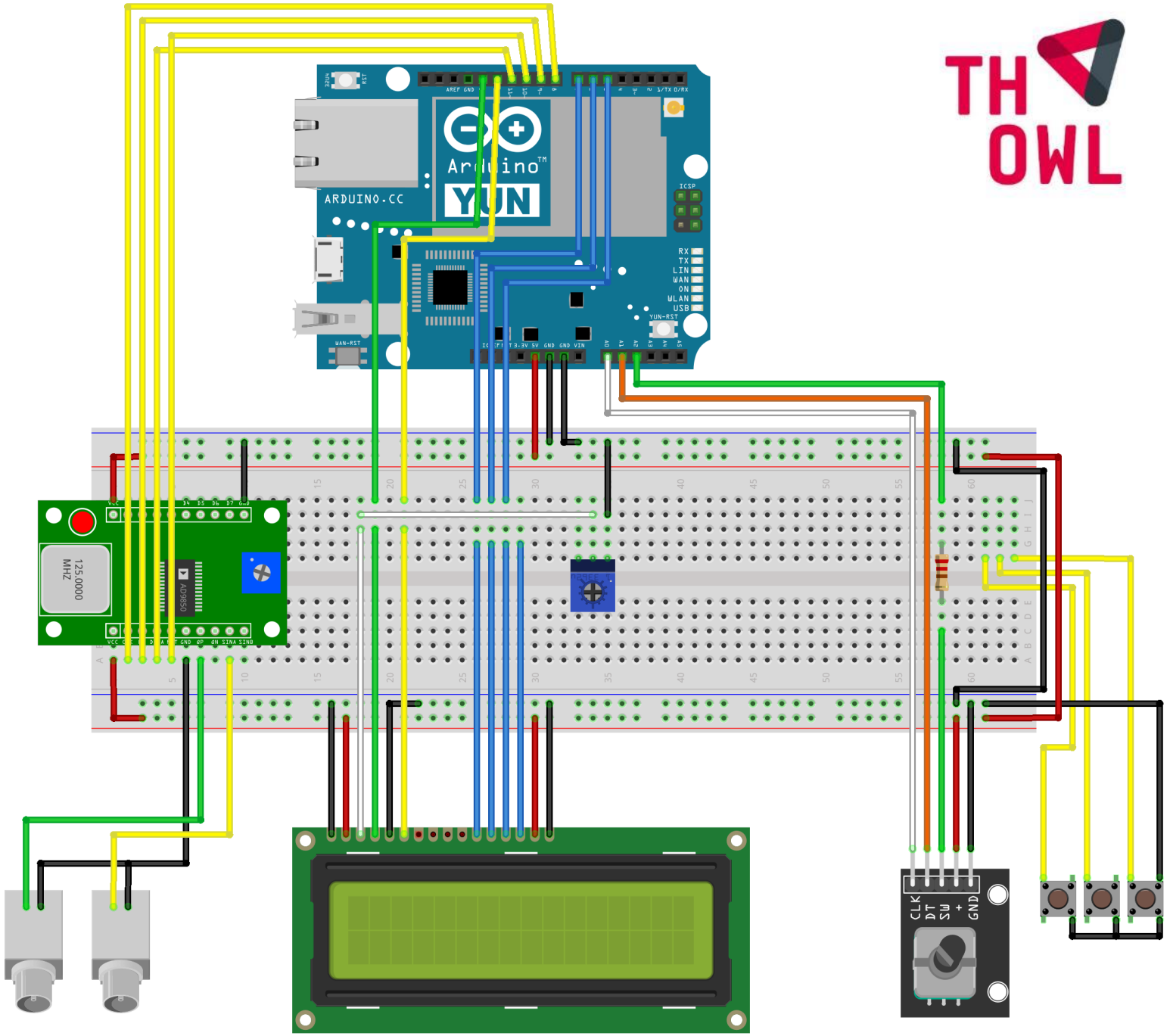
Innen: A
Außen: B

A	B
1	1

Der Aufbau

Der Aufbau





Das Arduino-Programm

Menüstruktur

- **Nutzen der „LCD-Simple Menu Library“**
- **Strukturierung durch verschiedene Menüebenen / Untermenüs**
- **Unterschiedlich strukturierte Menüpunkte für unterschiedliche Aufgaben:**

```
SimpleMenu(String _name, void (*_CallBack)()); //function menu
```

```
SimpleMenu(String _name,int *_value, int _min, int _max); //Value menu with min and max
```

Menüstruktur

```
153 SimpleMenu Menu[4] = {
154   SimpleMenu("Statische Freq.", 4, MenuSubStatic),
155   SimpleMenu("Lin. Freqlauf", 6, MenuSubLin),
156   SimpleMenu("Log. Freqlauf", 6, MenuSubLog),
157   SimpleMenu("Info", info)
158 };
```

Menüstruktur

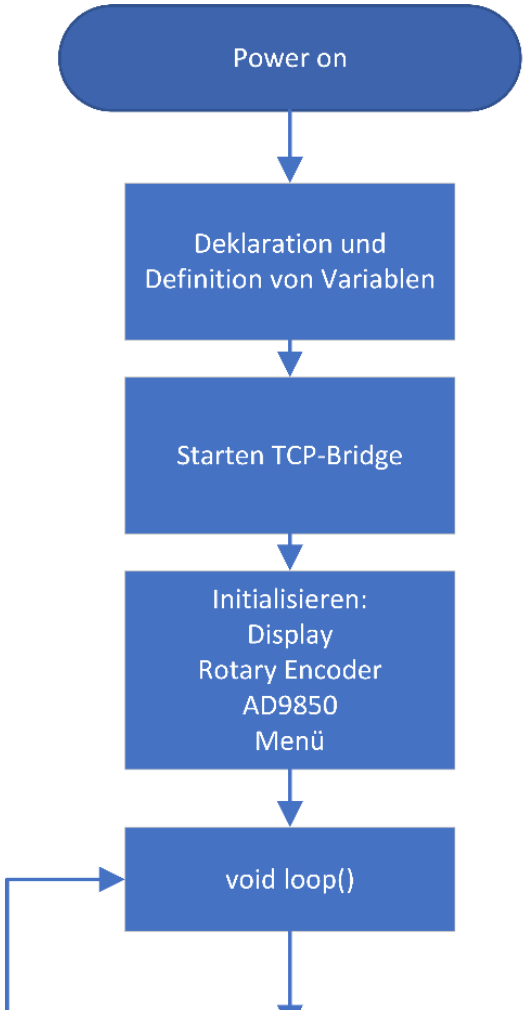
```
144 SimpleMenu MenuSubLog[6] = {
145   SimpleMenu("Startfrequenz", setLogStartFreq),
146   SimpleMenu("Anzahl Dekaden", &logDecades, 0, 7),
147   SimpleMenu("Anzahl Schritte", &logSteps, 50, 500),
148   SimpleMenu("Sweepzeit", setLogSweeptime),
149   SimpleMenu("Starten", logFreqStart),
150   SimpleMenu("Zurueck", goBack)
151 };
```

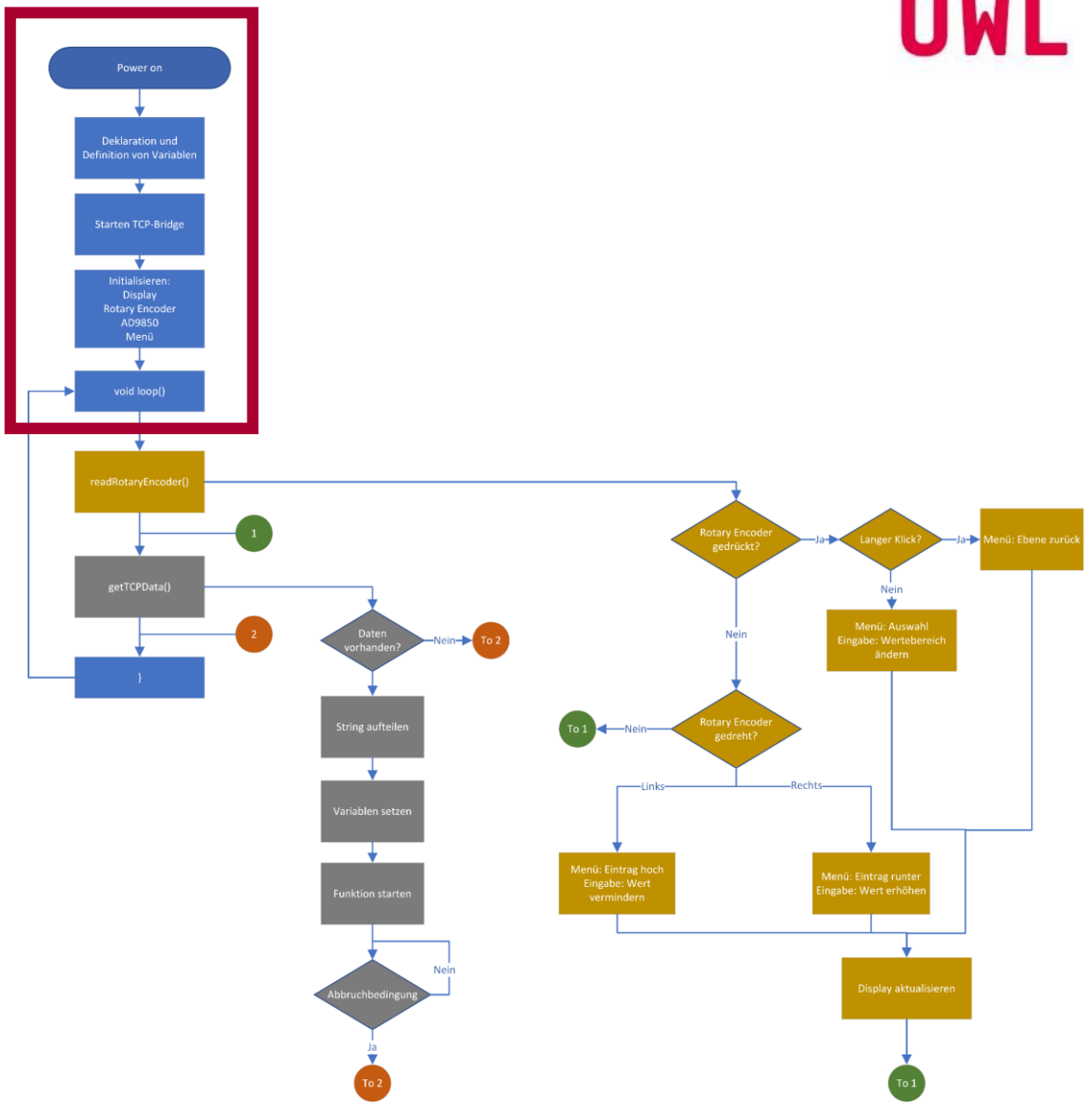
Menüstruktur

```
32          void home();
33          void select();
34          void back();
35          void returned();
36          void up();
37          void down();
```

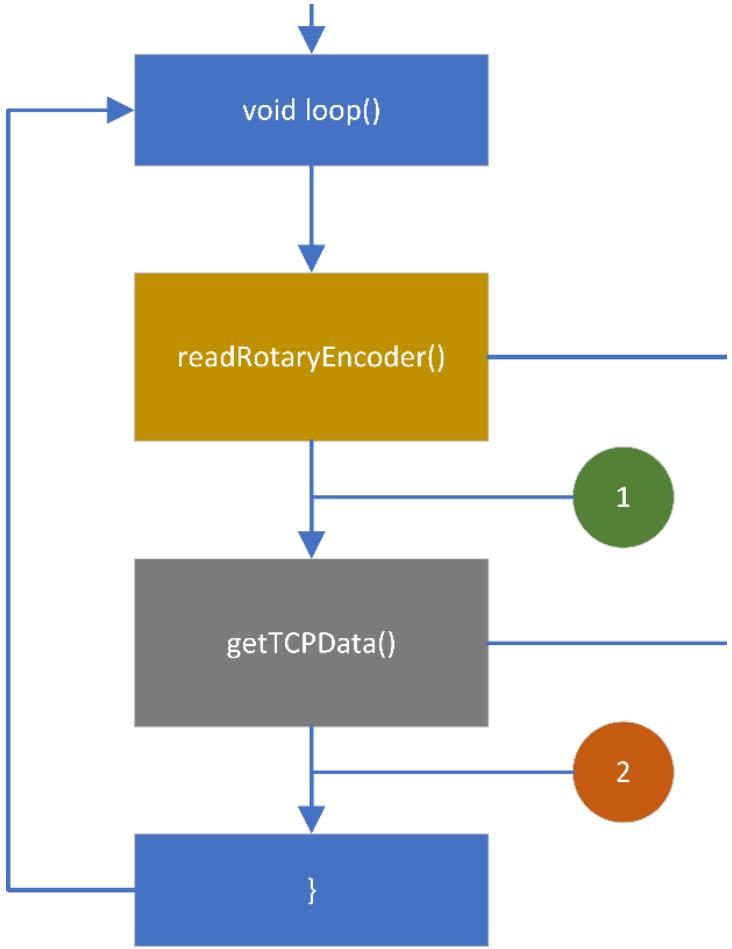
```
40          SimpleMenu *next();
41          SimpleMenu *next(int index);
42          int getValue();
43          bool hasValue();
44          int getIndex();
45          void print();
```

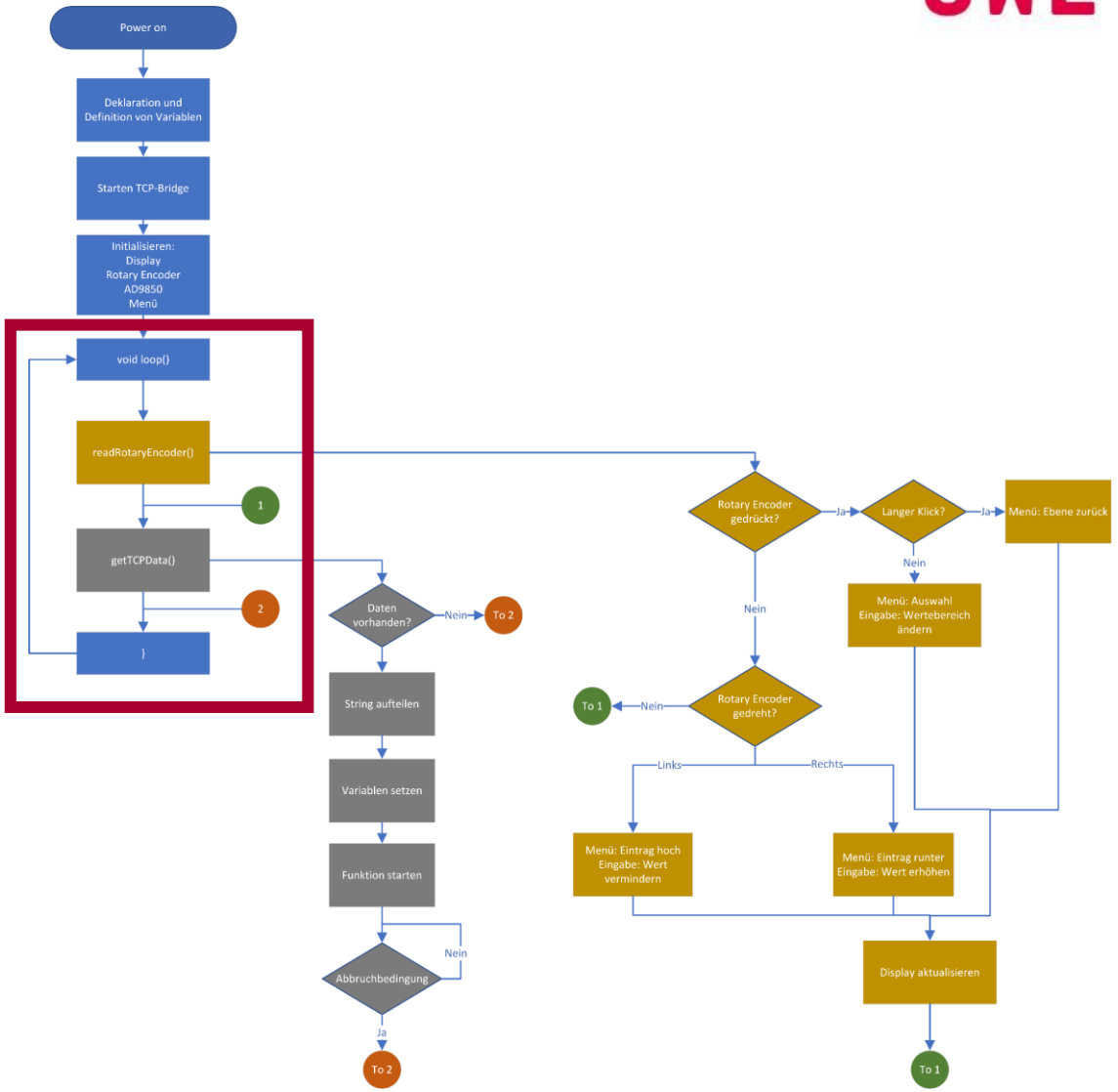
Programmablauf



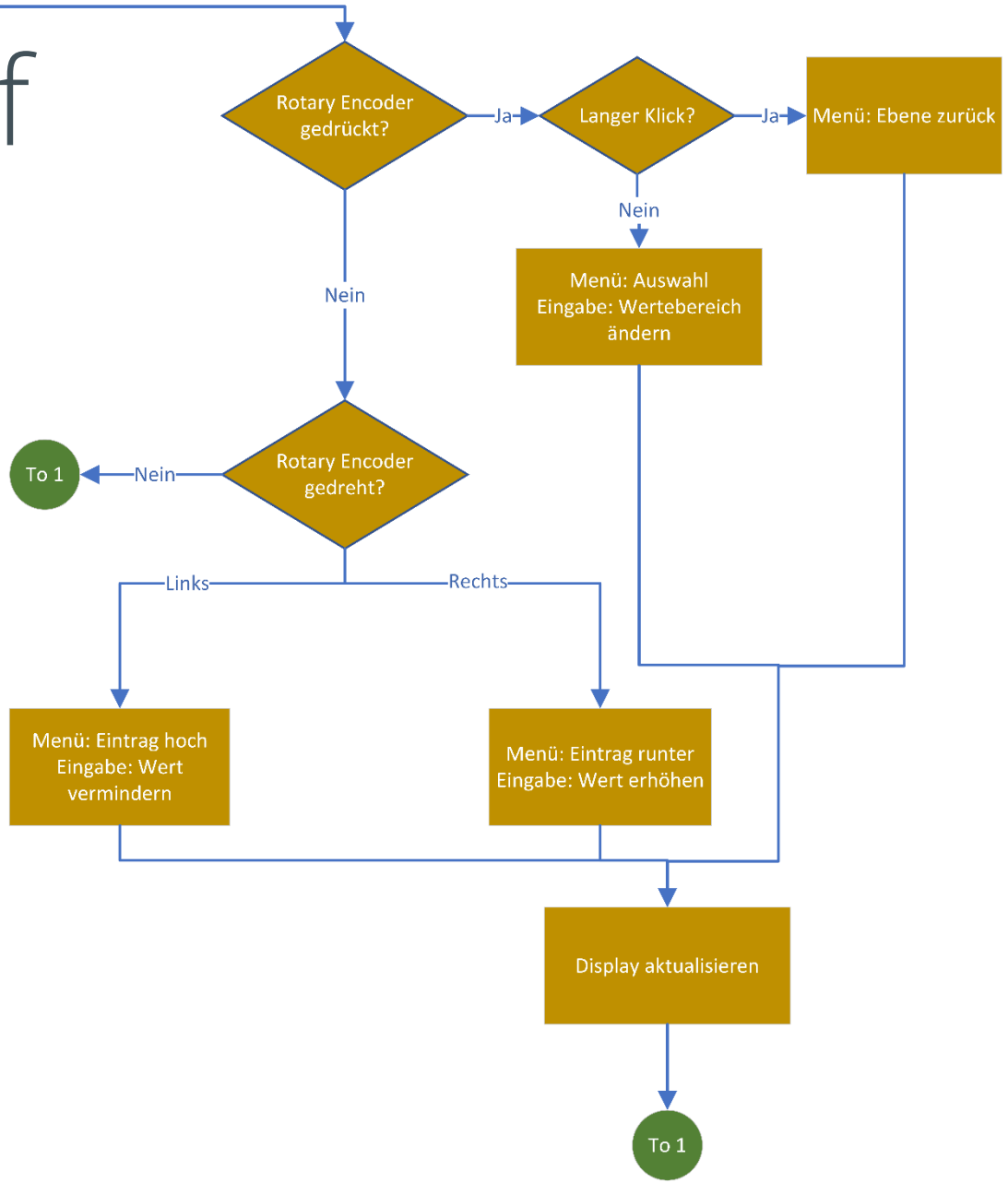


Programmablauf

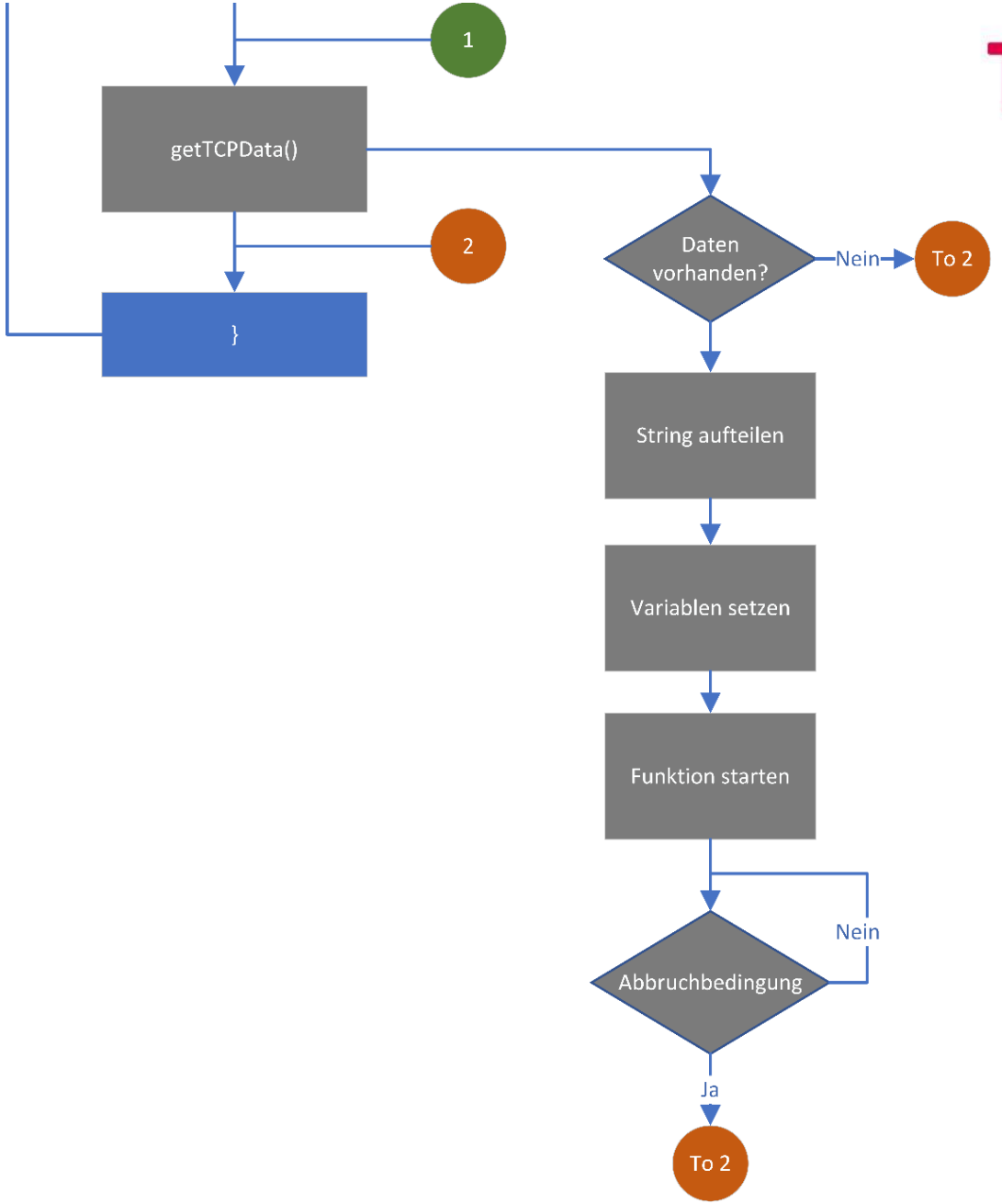




Programmablauf



Programmablauf



Ausgewählte Funktionen

```
113 //general function heads
114 void setHighValue(long int *value, int mode);
115 void info();
116 void goBack();
117
118 //function heads for TCP connection
119 void startBridge();
120 void getTCPData();
121
122 //funtion heads for frequency output
123 void setStaticFreq();
124 void setLinStartFreq();
125 void setLogStartFreq();
126 void setLinEndFreq();
127 void staticFreqStart();
128 void setLinSweeptime();
129 void setLogSweeptime();
130 void linFreqStart();
131 void logFreqStart();
132 void linSweep(long int freqStart, long int freqStop, int nstep, long int tms);
133 void logSweep(long int freqStart, int nDec, int nstep, int tms);
```

setHighValue

```
493 void setHighValue(long int *value, int mode) {
494   valueMode = mode;
495   stepMode = 0;
```

- **Zur Einstellung sämtlicher Variablenwerte**
- **Adressübergabe der zu ändernden Variable als Pointer**
- **Übergabe Eingabeformat als „mode“**

setHighValue

```
496   while (1) {
497   //leave loop, if RotaryEncoder is held long
498     readRotaryEncoder();
499     if (doBreak == 1) {
500       doBreak = 0;
501       break;
502     }
```

Funktion verlassen

Schrittweite wählen

```
503 //increment step mode by clicking the encoder
504     if (doSelect == 1) {
505       doSelect = 0;
506       stepMode++;
507       if (stepMode > 3) {
508         stepMode = 0;
509       }
510     }
```

setHighValue

- **Unterscheidung, ob Frequenz oder Zeit**
- **Inkrementieren bzw. Dekrementieren des Variablenwertes**
- **Festlegen der Grenzwerte**

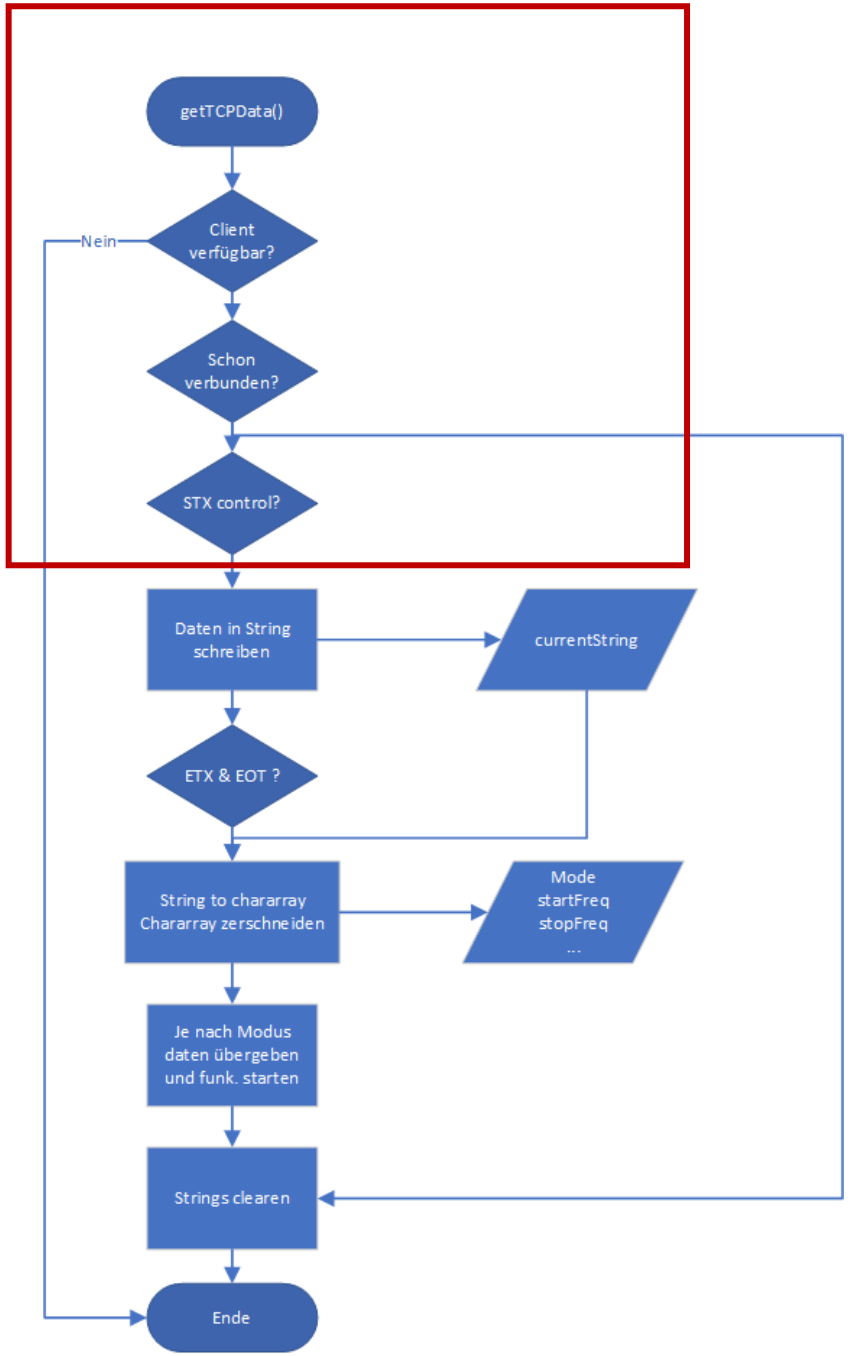
```
512   switch (valueMode) {
513     case 1:
514       //if Encoder is turned right:
515       if (turnRight == 1) {
516         turnRight = 0;
517         *value += changeValue;
518         if (*value > 40000000) {
519           *value = lastValue;
520         }
521       }
522       //if Encoder is turned left:
523       else if (turnLeft == 1) {
524         turnLeft = 0;
525         *value -= changeValue;
526         if (*value < 10) {
527           *value = lastValue;
528         }
529       }
```

setHighValue

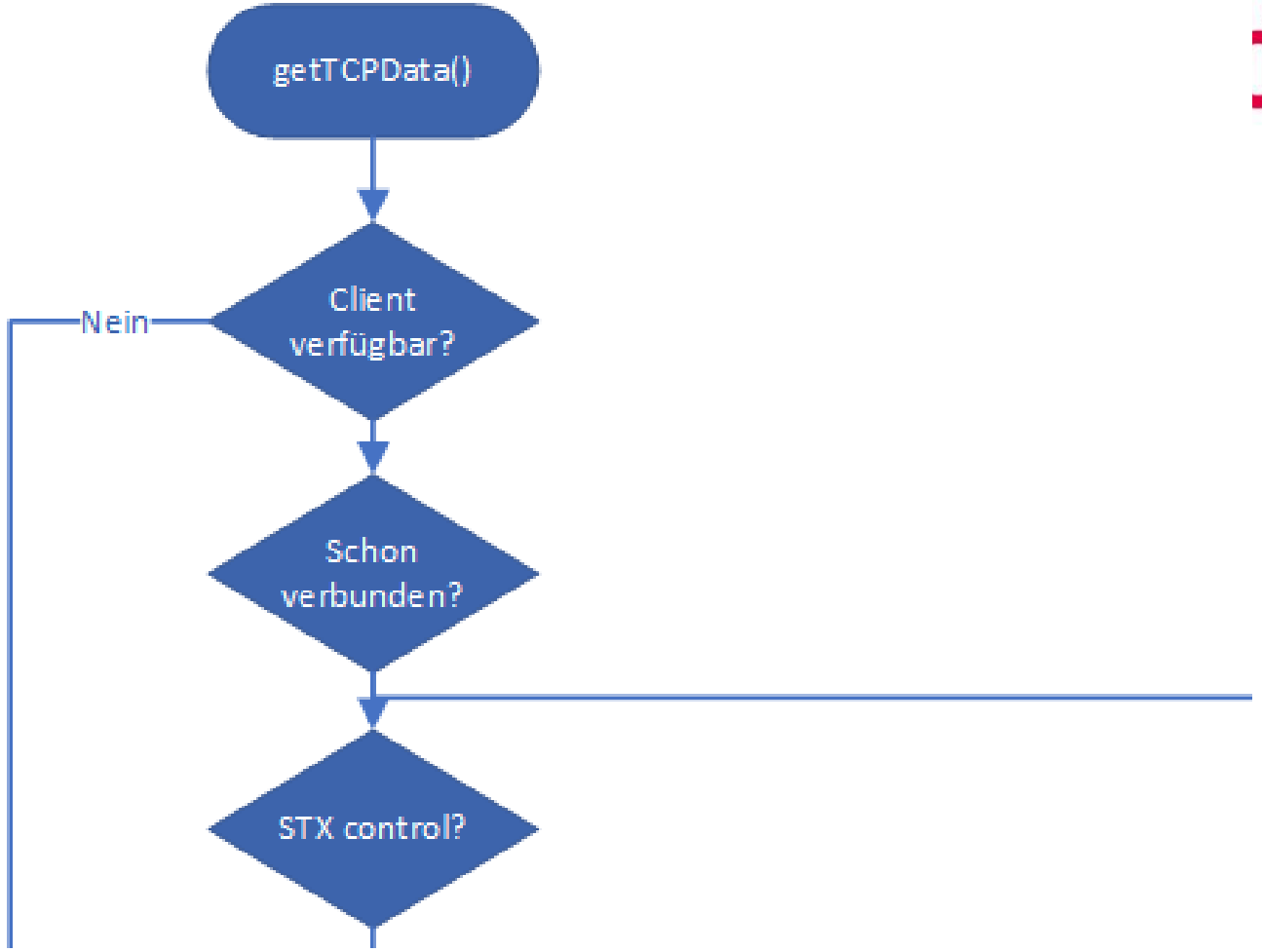
- **Festlegen der Schrittweite**
- **10Hz, 100Hz, 10kHz, 1MHz**
- **10ms, 100ms, 1s, 10s**

```
532       //definig different step sizes
533       switch (stepMode) {
534         //1 step = 10Hz
535         case 0:
536           changeValue = 10;
537           lcd.setCursor(0, 1);
538           lcd.print("Step:  10Hz");
539           break;
540         case 1:
541           changeValue = 100;
542           lcd.setCursor(0, 1);
543           lcd.print("Step: 100Hz");
544           break;
545         case 2:
```

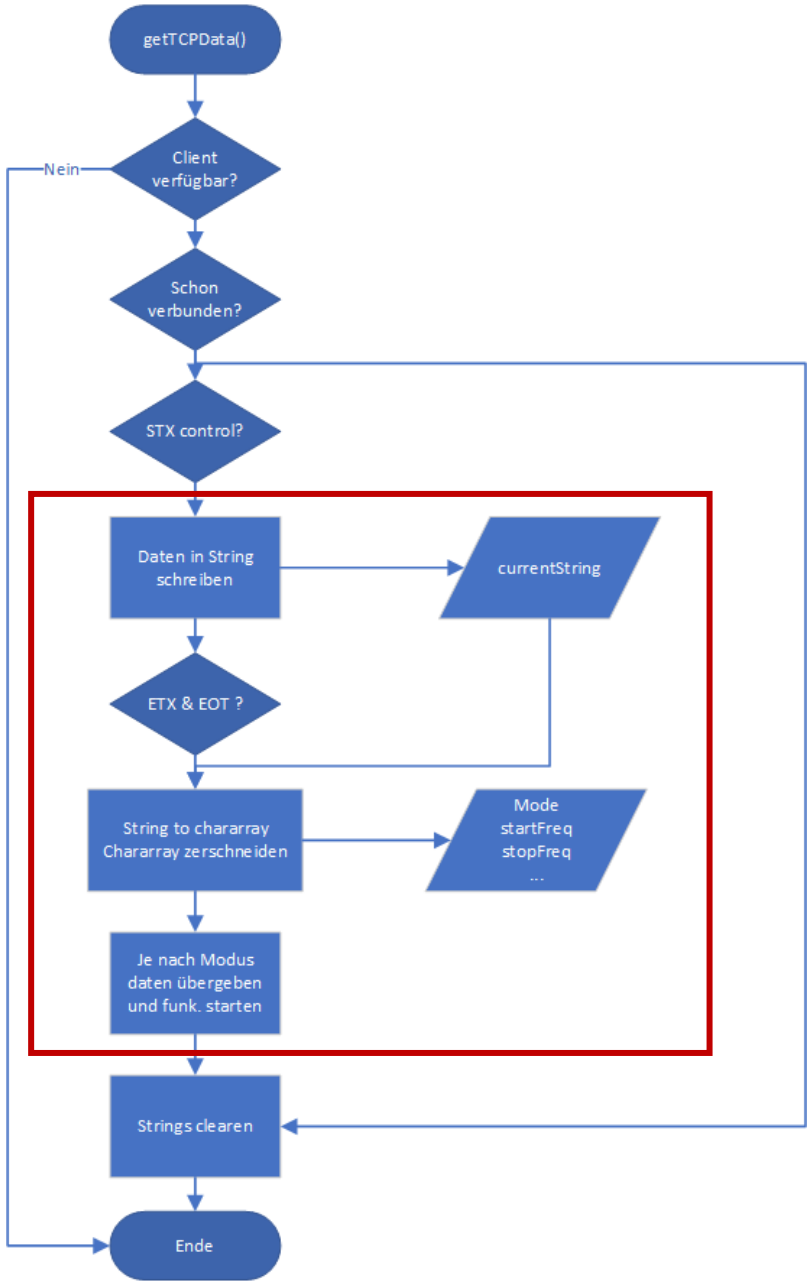
getTCPData()


getTCPData()
Client verfügbar?
Nein
Schon verbunden?
STX control?
Daten in String schreiben
currentString
ETX & EOT ?
String to chararray Chararray zerschneiden
Mode
startFreq
stopFreq
...
Je nach Modus daten übergeben und funk. starten
Strings clearen
Ende

getTCPData()

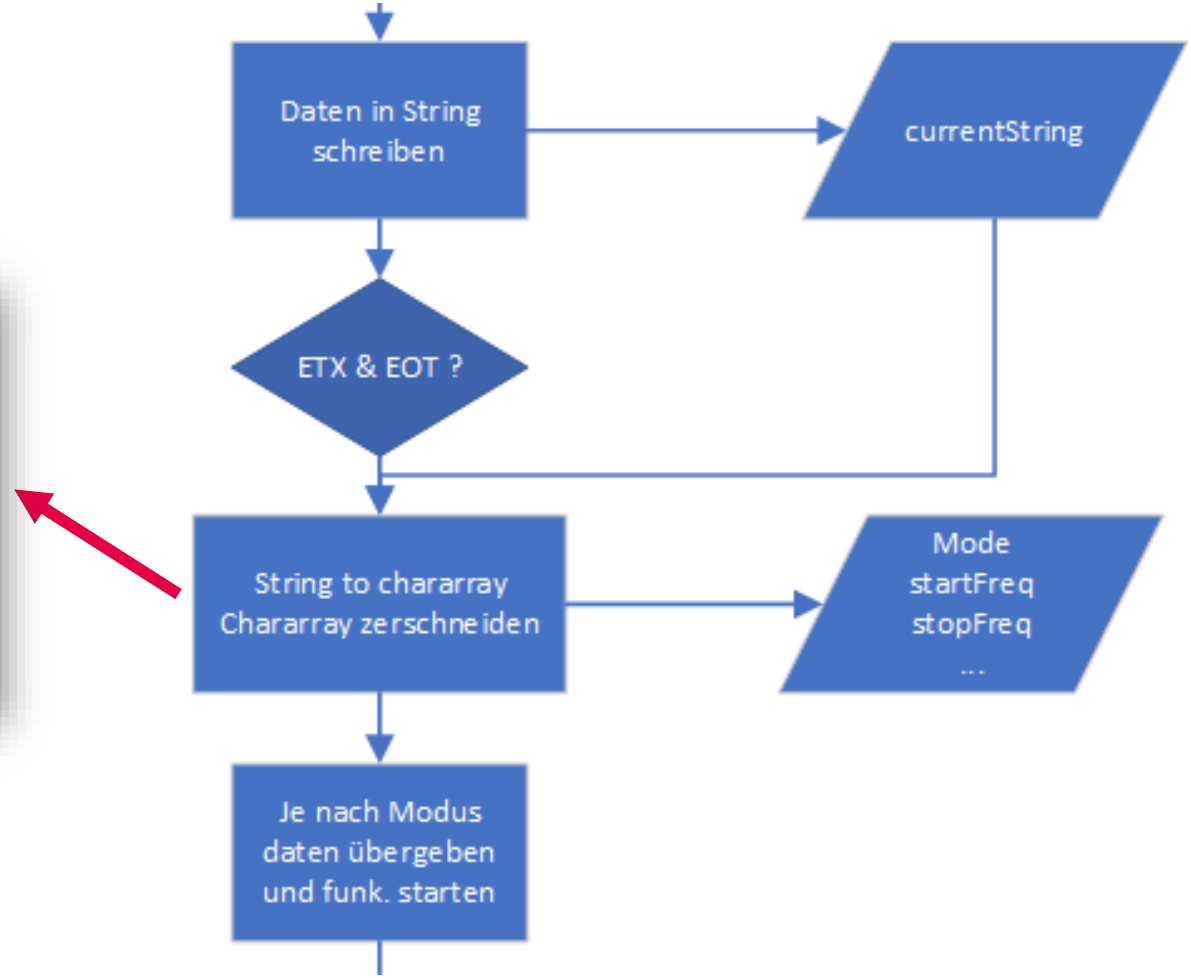

getTCPData()
Client verfügbar?
Nein
Schon verbunden?
STX control?

getTCPData()

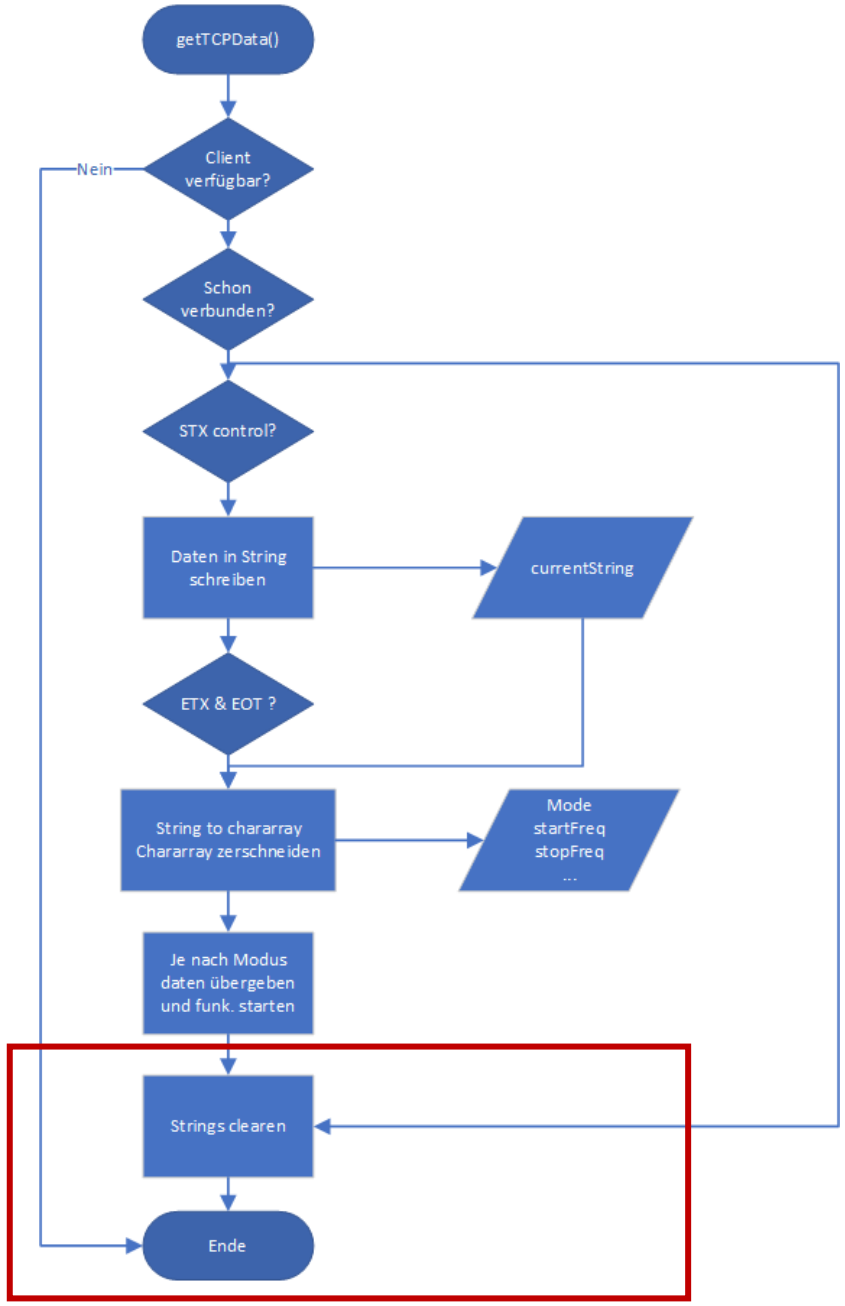

getTCPData()
Client verfügbar?
Nein
Schon verbunden?
STX control?
Daten in String schreiben
currentString
ETX & EOT ?
String to chararray Chararray zerschneiden
Mode
startFreq
stopFreq
...
Je nach Modus daten übergeben und funk. starten
Strings clearen
Ende

getTCPData()

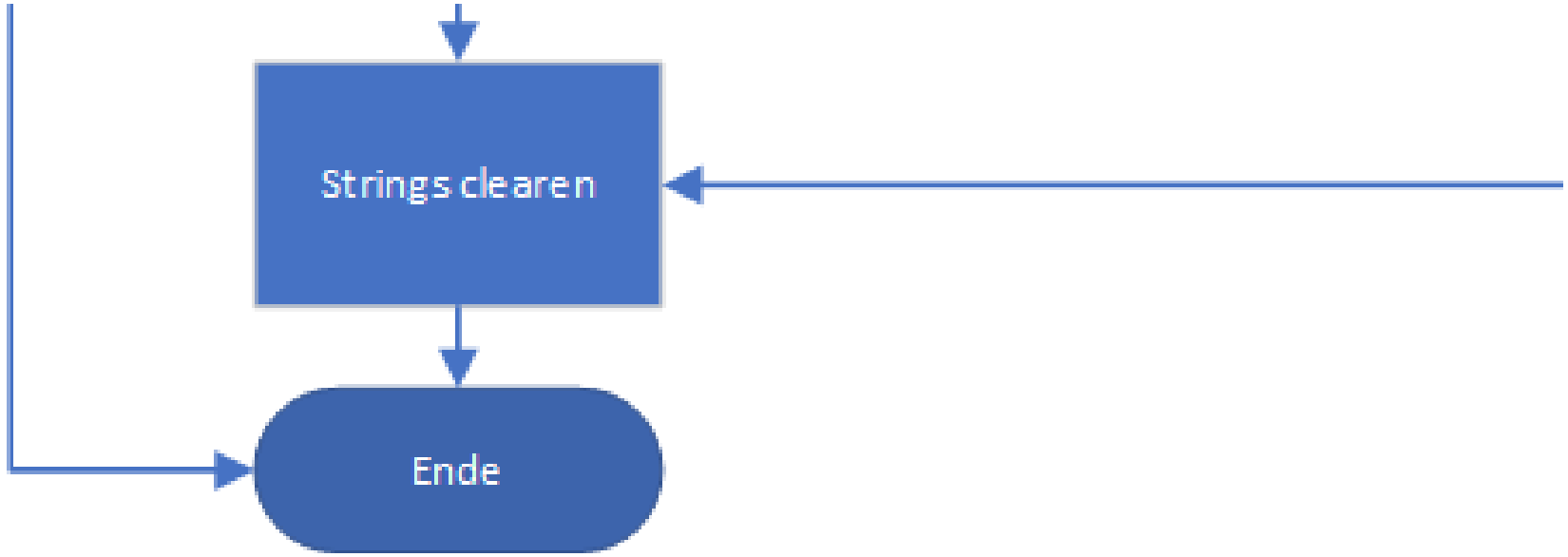
```
//Convert String to char-Array (atoi only
currentString.toCharArray(workString, 30)

//cut String into the single parameters
mode = atoi(strtok(workString, ";"));
startFreq = atol(strtok(NULL, ";"));
stopFreq = atol(strtok(NULL, ";"));
nDec = atoi(strtok(NULL, ";"));
nStep = atoi(strtok(NULL, ";"));
tms = atoi(strtok(NULL, ";"));
```



getTCPData()


getTCPData()
Client verfügbar?
Nein
Schon verbunden?
STX control?
Daten in String schreiben
currentString
ETX & EOT ?
String to chararray Chararray zerschneiden
Mode
startFreq
stopFreq
...
Je nach Modus daten übergeben und funk. starten
Strings clearen
Ende

getTCPData()



logSweep

```
695 void logSweep(long int freqStart, int nDec, int nstep, int tms) {
696 // freqStart: start frequency
697 // nDec: number of decades
698 // nstep: number of sweep-steps
699 // tms: sweep time in ms
```

- **Berechnung eines exponentiellen Frequenzverlaufs, Übergabe an den DDS**
- **Ausgabe aller wichtigen Parameter auf dem LCD**

logSweep

- **Berechnung Endfrequenz & Zeitspanne pro einzelnem Schritt**

```
732   freqStop = freqStart * pow(10, nDec);
733   //calculate max. duration of one step
734   dt = tms / (nstep - 1);
```

- **Schleife mit Abbruchbedingung, jedem Schritt eigene Frequenz zugeordnet**

```
737   for (i = 0; i < nstep; i++) {
738     readRotaryEncoder();
739     //leave loop, if Encoder is held long
740     if (doBreak == 1) {
741       break;
742     }
```

logSweep

- **Math. Funktion zur Modellierung eines exponentiellen Anstiegs mit bestimmter Schrittanzahl und Wertebereich:**

$$y(i) = 2^{log_2(res)*\frac{i+1}{nstep}} - 1$$

- **res = Wertebereich von y → wenn res = 65536 (16 Bit), dann kann y alle Werte zwischen 0 und 65535 annehmen**
- **i = Zählervariable, pro Schleifendurchlauf inkrementiert**
- **An den DDS übergebene Frequenz in Abhängigkeit von i:**

$$freq(i) = freqStart + \frac{freqStop - freqStart}{res - 1} * y(i)$$

logSweep

- **Umsetzung im Quellcode:**

```
777     y = pow(2, (log(res) / log(2)) * (i + 1) / nstep) - 1;
778     freq = freqStart + ((freqStop - freqStart) / (res - 1.0)) * y;
779     DDS.setfreq(freq, phase);
```

- **Warteschleife, bis Zeit für einen Schritt verstrichen ist:**

```
775     t1 = millis();
```

```
782     //wait until next step can be executed
783     while ((millis() - t1) < dt) {}
```

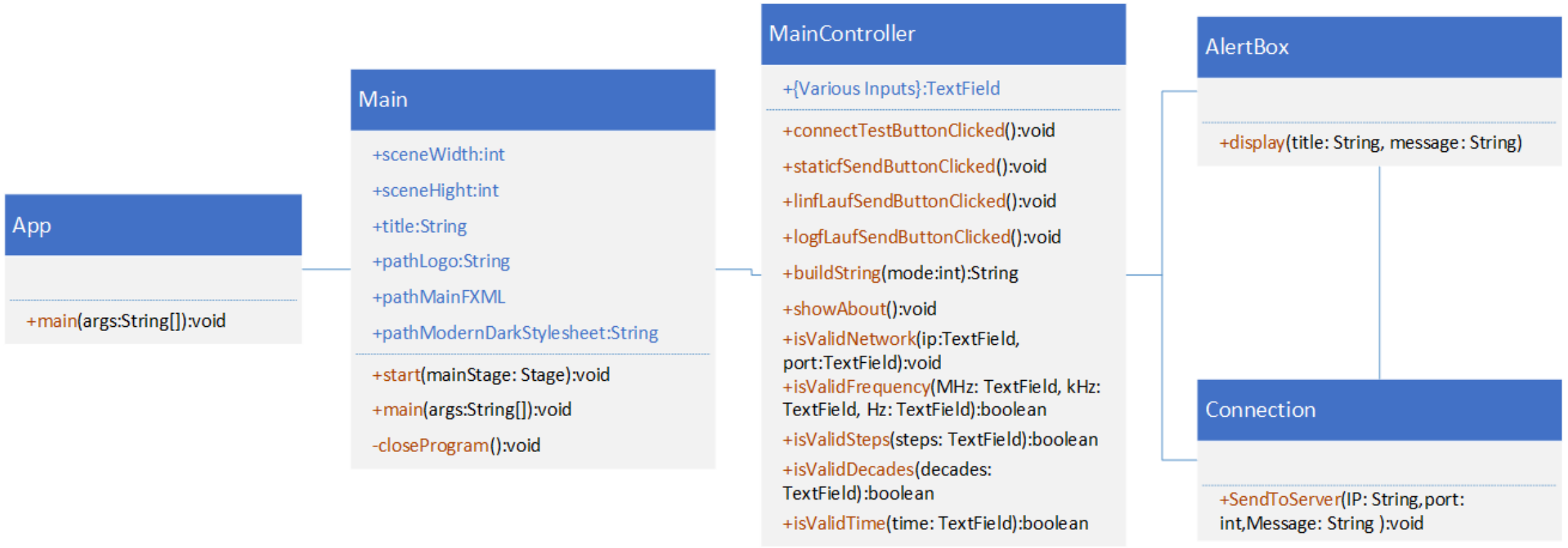
logSweep

- **Wechselnde Displayausgabe:**

```
744     if ((millis() - previousMillis) > interval) {
745       previousMillis = millis();
746       lcd.clear();
747       if (page == 0) {
748         page = 1;
749       }
750       else {
751         page = 0;
752       }
753     }
```

Das remote Programm

Klassen


App
+main(args:String[]):void
Main
+sceneWidth:int
+sceneHight:int
+title:String
+pathLogo:String
+pathMainFXML
+pathModernDarkStylesheet:String
+start(mainStage: Stage):void
+main(args:String[]):void
-closeProgram():void
MainController
+{Various Inputs}:TextField
+connectTestButtonClicked():void
+staticfSendButtonClicked():void
+linfLaufSendButtonClicked():void
+logfLaufSendButtonClicked():void
+buildString(mode:int):String
+showAbout():void
+isValidNetwork(ip:TextField, port:TextField):void
+isValidFrequency(MHz: TextField, kHz: TextField, Hz: TextField):boolean
+isValidSteps(steps: TextField):boolean
+isValidDecades(decades: TextField):boolean
+isValidTime(time: TextField):boolean
AlertBox
+display(title: String, message: String)
Connection
+SendToServer(IP: String, port: int, Message: String):void

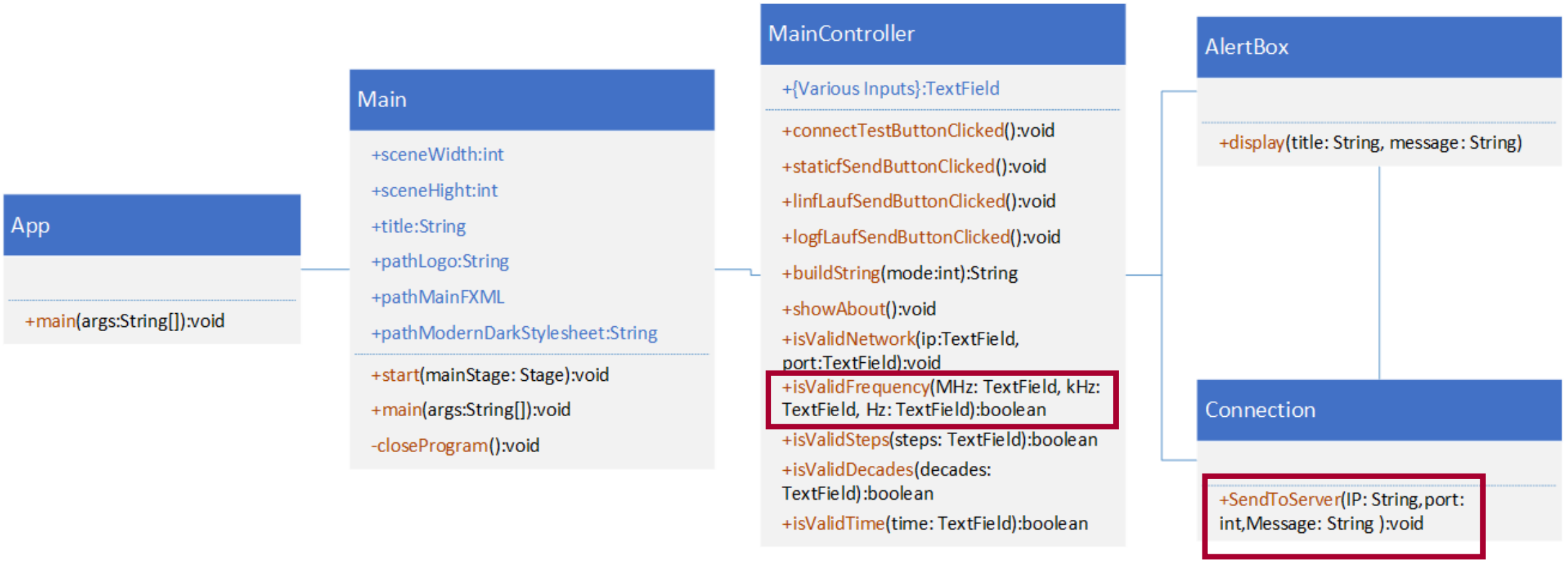
JavaFX -> FXML

```
<AnchorPane>
   <children>
      <Label fx:id="ipLabel" layoutX="41.0" layoutY="13.0" text="IP:">
         <font>
            <Font size="16.0" />
         </font>
      </Label>
      <TextField fx:id="ipInput" layoutX="75.0" layoutY="13.0" onAction="#connectTestButtonClicked" promptText="127.0.0.1" text="192.168.0.168" />
      <Label fx:id="portLabel" layoutX="267.0" layoutY="13.0" text="Port:">
         <font>
            <Font size="16.0" />
         </font>
      </Label>
```

JavaFX -> FXML

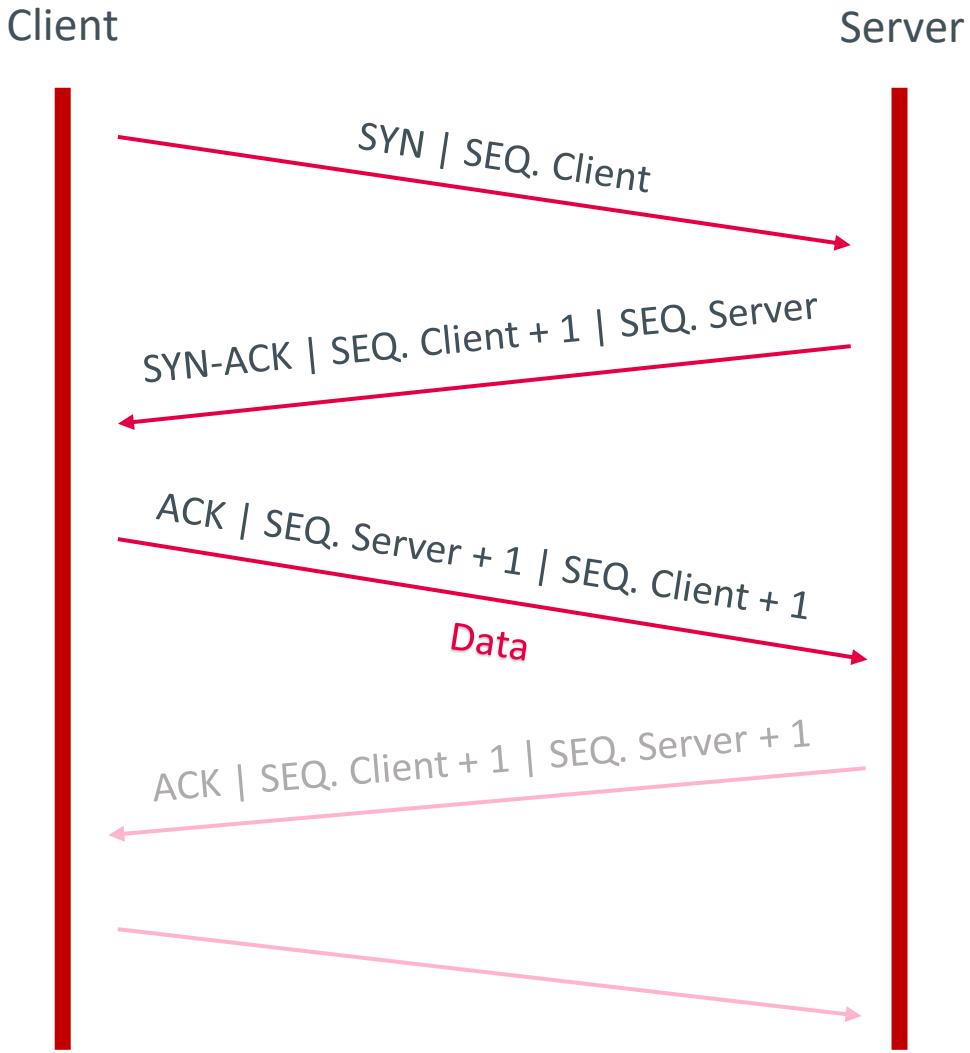
```java
/** Define Textfield-Input for connection (IP and Port of the DDS) (link to fxml) */
@FXML
public TextField ipInput, portInput;
```

Ausgewählte Funktionen



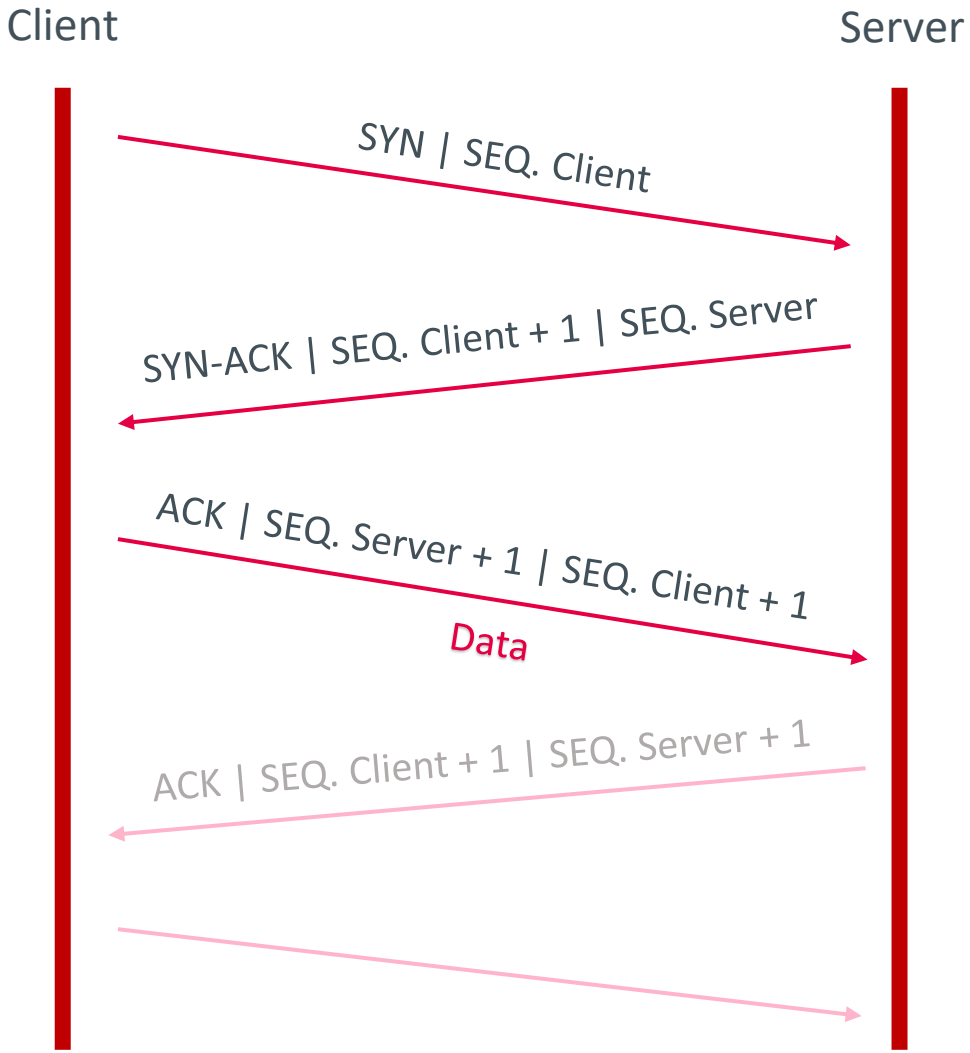
sendToServer(...)

```java
public static void sendToServer(String IP, int port, String Message) {
    //Try to establish a connection
    try (Socket socket = new Socket(IP, port)) {
        //Send data
        OutputStream output = socket.getOutputStream();
        PrintWriter writer = new PrintWriter(output, autoFlush: true);
        writer.println(Message);
        System.out.println("Message send: " + Message);
        System.out.println("To: " + IP + ":" + port);

    } catch (UnknownHostException ex) {
        //Build exception String
        String unknownHostExceptionString = "Server not found: ".concat(ex.getMessage());
        //Give out error message
        System.out.println(unknownHostExceptionString);
        AlertBox.display( title: "Übermittlungsfehler", unknownHostExceptionString);

    } catch (IOException ex) {
        //Build exception String
        String unknownIOExceptionString = "IO Error: ".concat(ex.getMessage());
        //Give out error message
        System.out.println(unknownIOExceptionString);
        AlertBox.display( title: "Übermittlungsfehler", unknownIOExceptionString);
    }
}
```



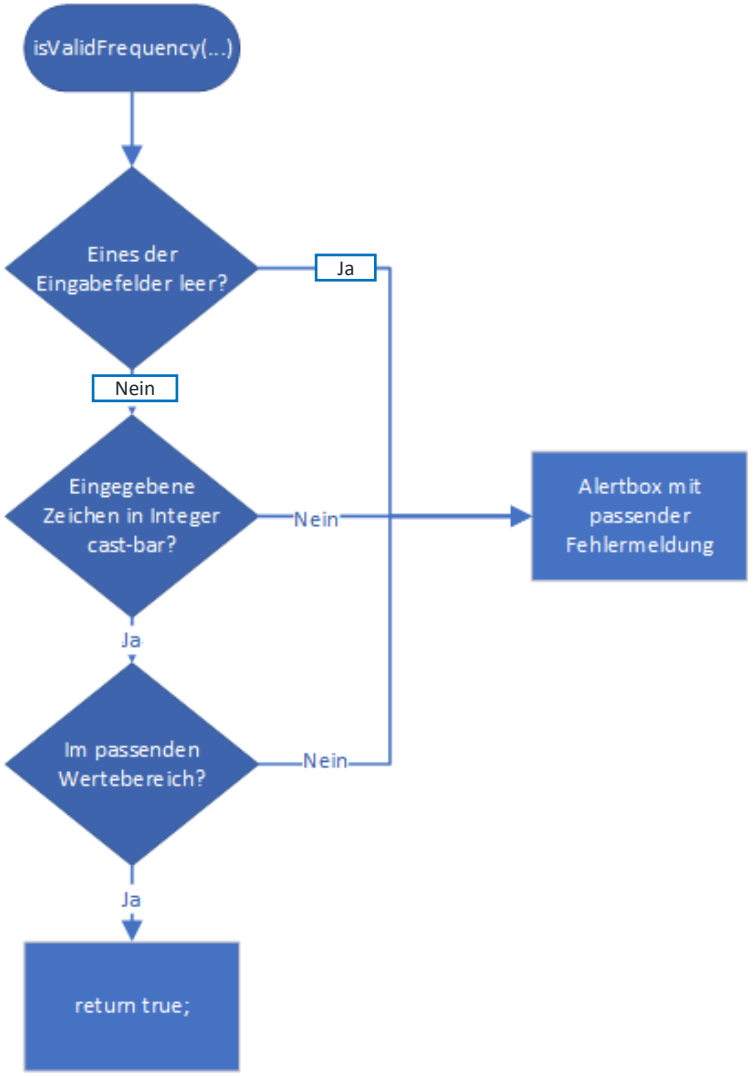
sendToServer(...)

```java
public static void sendToServer(String IP, int port, String Message) {
    //Try to establish a connection
    try (Socket socket = new Socket(IP, port)) {
        //Send data
        OutputStream output = socket.getOutputStream();
        PrintWriter writer = new PrintWriter(output, autoFlush: true);
        writer.println(Message);
        System.out.println("Message send: " + Message);
        System.out.println("To: " + IP + ":" + port);

    } catch (UnknownHostException ex) {
        //Build exception String
        String unknownHostExceptionString = "Server not found: ".concat(ex.getMessage());
        //Give out error message
        System.out.println(unknownHostExceptionString);
        AlertBox.display( title: "Übermittlungsfehler", unknownHostExceptionString);

    } catch (IOException ex) {
        //Build exception String
        String unknownIOExceptionString = "IO Error: ".concat(ex.getMessage());
        //Give out error message
        System.out.println(unknownIOExceptionString);
        AlertBox.display( title: "Übermittlungsfehler", unknownIOExceptionString);
    }
}
```



isValidFrequency(...)

```java
public boolean isValidFrequency(TextField MHz, TextField kHz, TextField Hz){
    if ((!MHz.getText().isEmpty()) && (!kHz.getText().isEmpty()) && (!Hz.getText().isEmpty())){
        try{
            int MHzInput = Integer.parseInt(MHz.getText());
            int kHzInput = Integer.parseInt(kHz.getText());
            int HzInput = Integer.parseInt(Hz.getText());
            int freqInput = (MHzInput*1000000) + (kHzInput*1000) + HzInput;
            System.out.println(freqInput);
            if((freqInput <= 40000000) && (freqInput >= 10)){
                return true;
            }
            else{...}
        }
        catch(NumberFormatException e){...}
    }
    else{...}
}
```

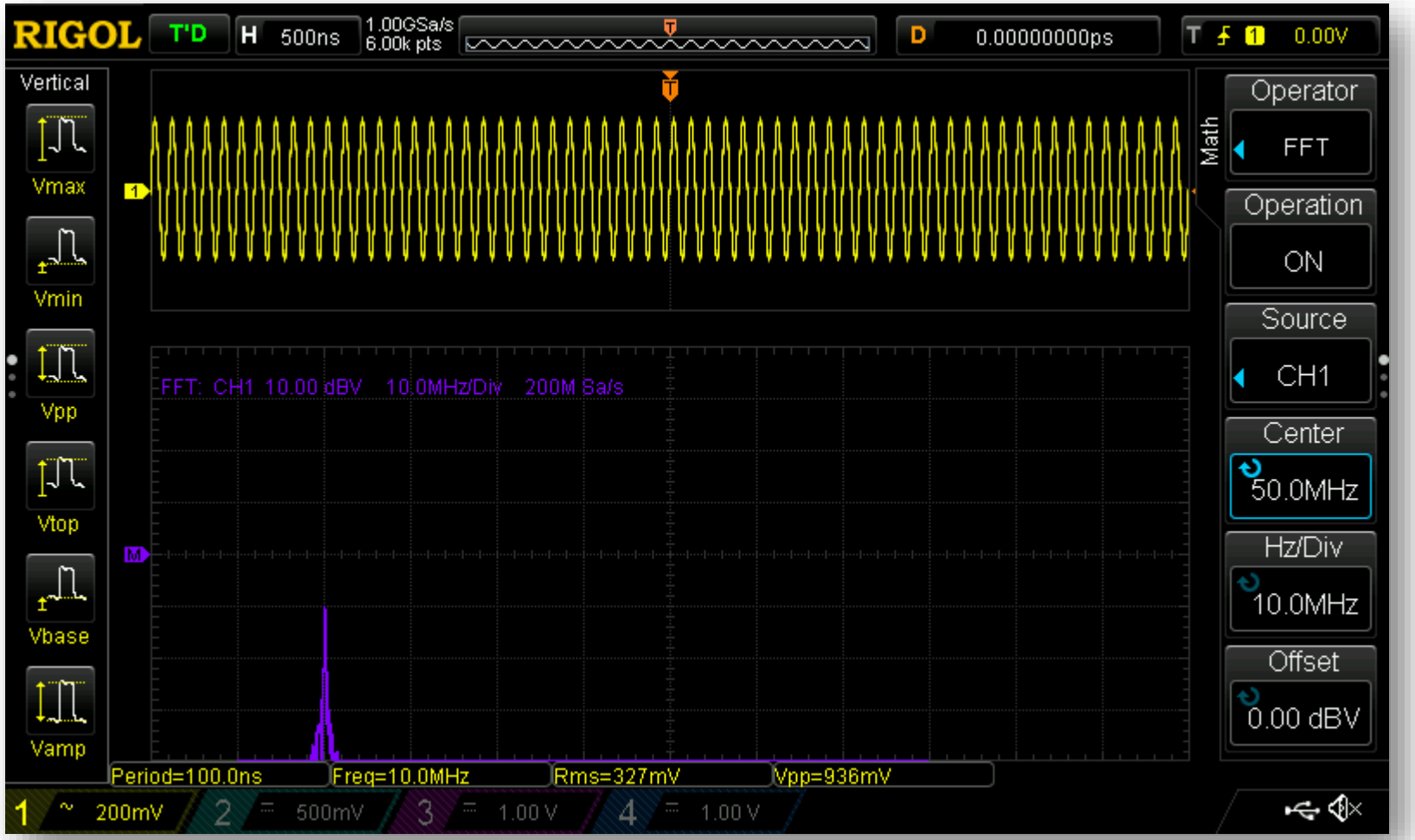


Vorführung

Messung und Fehleranalyse

Frequenzmessung

FFT


RIGOL
T'D
H 500ns
1.00GSa/s
6.00k pts
D 0.00000000ps
T 1 0.00V
Vertical
Vmax
Vmin
Vpp
Vtop
Vbase
Vamp
Math
FFT: CH1 10.00 dBV 10.0MHz/Div 200M Sa/s
Period=100.0ns
Freq=10.0MHz
Rms=327mV
Vpp=936mV
1 200mV
2 500mV
3 1.00 V
4 1.00 V
Operator
FFT
Operation
ON
Source
CH1
Center
50.0MHz
Hz/Div
10.0MHz
Offset
0.00 dBV

Ausblick

Ausblick

Quellen

Quellen

- **Siehe Dokumentation**

TH OWL

TECHNISCHE HOCHSCHULE
OSTWESTFALEN-LIPPE
UNIVERSITY OF
APPLIED SCIENCES
AND ARTS

Vielen Dank für Eure Aufmerksamkeit!