QO-100 Empfängerbau

Der Satellit

Der Satellit

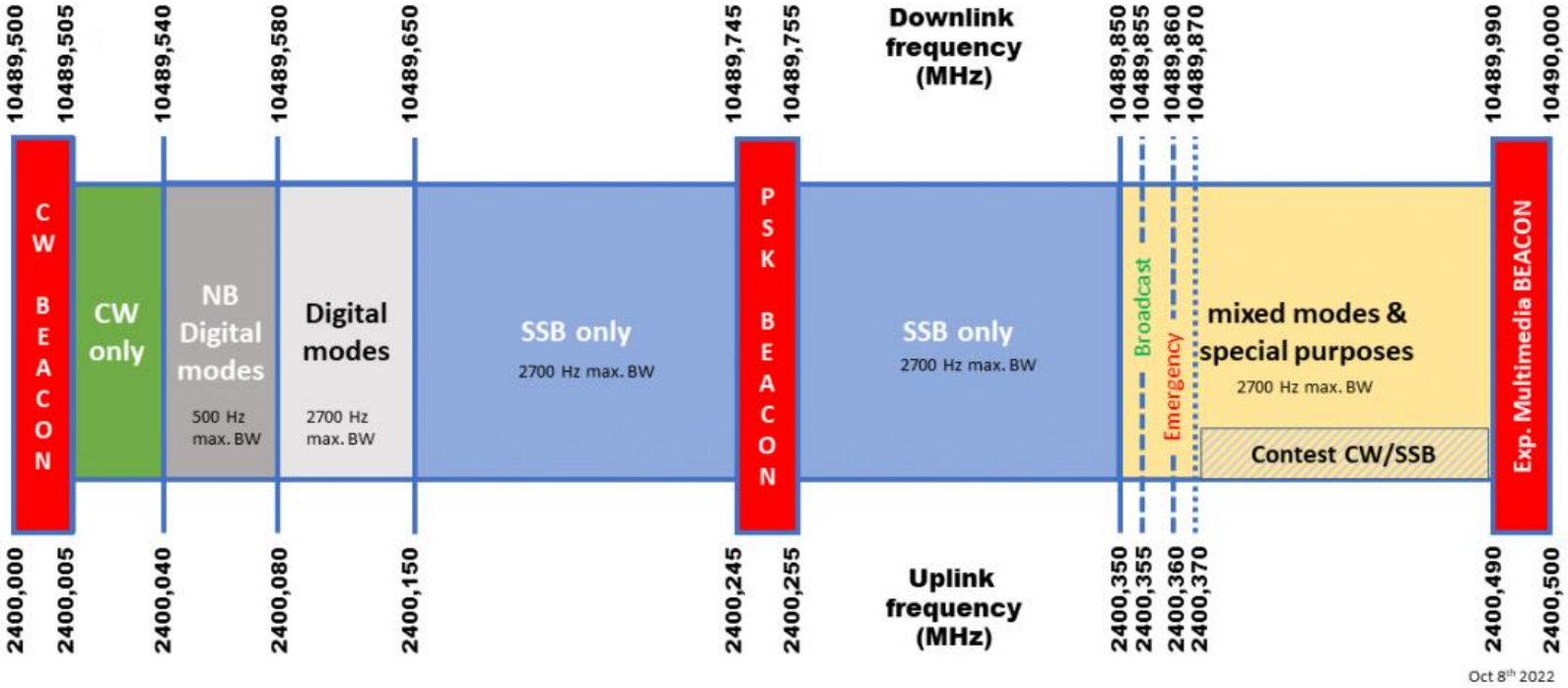
- Geostationärer Satellit 25,5° Ost
- Umlaufbahn etwa 35.700km
  - Die ISS hat eine Bahn Höhe von etwa 400 km
- Empfänger Projekt adressiert Schmalband (CW/Phonie)

QO-100 NB Transponder Bandplan

AMSAT QO-100 / P4A NB Transponder Bandplan

AMSAT-DL Satelliten für Kommunikation, Wissenschaft und Bildung
Satellites for Communication, Science and Education

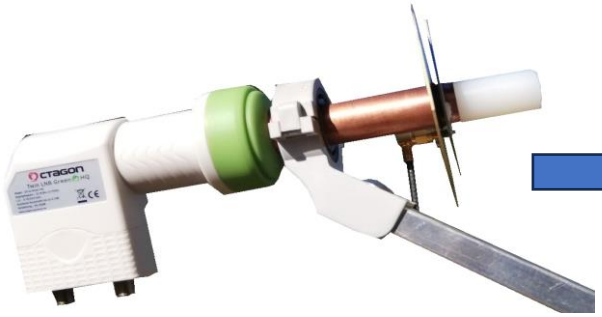
Es'hailSat سهيل سات



Arbeiten im NB Bereich

- Up 2,400 000 GHz – 2,400 495 GHz
- Down10,489 500 GHz – 10,489 995 GHz
- Das Selbstbauprojekt soll Empfang von 10GHz hörbar machen
  - SSB Empfänger für 70cm Band nötig
    - IC 9700, IC 705, IC-910H (mit 70 cm Modul), FT 817/817, TS-2000, FT-991A, FT-847, diverse SDR Lösungen
  - Runtermischen auf 6m Band für KW Geräte

Empfang

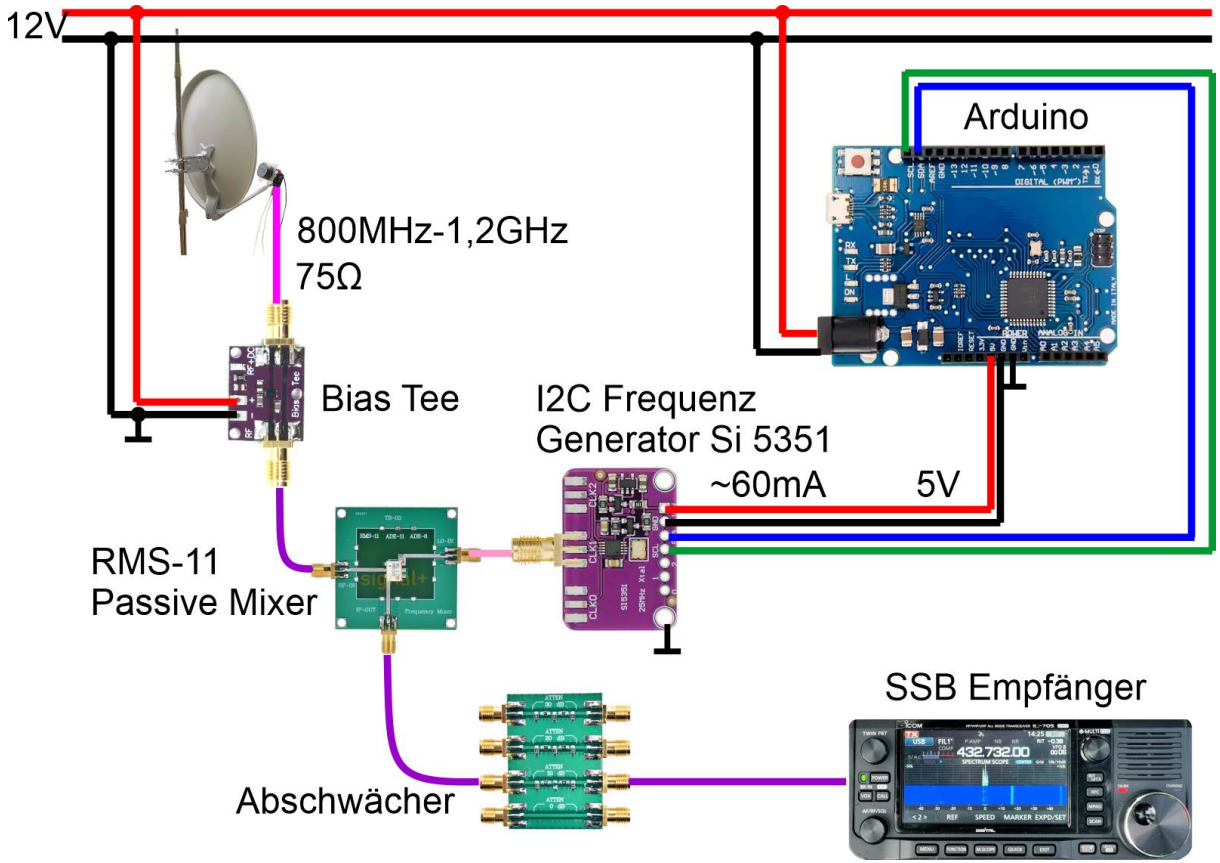


LNB 10 GHz -> ~1GHz

Down Converter 70cm bis 10m

SSB Empfänger

Lernprojekt



Lernprojekt

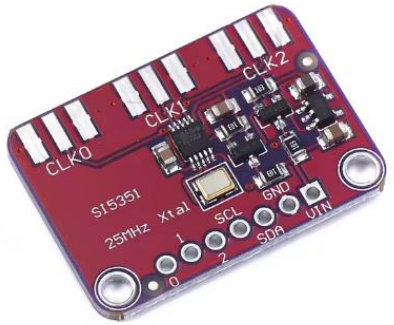
- Nur Empfang, kein Senden
- Keine Frequenzstabilisierung
  - Handelsüblicher TV LNB ohne Stabilisierung
  - Bei Mischen vervielfachen sich Frequenzfehler -> instabil
  - Daher eher hohe Mischer Frequenz am Si 5351 um Fehler gering zu halten

Empfänger Selbstbau

- Blechgehäuse zur Abschirmung
  - Es gibt mehrere HF Quellen, die sollen nicht stören
- Einzelkomponenten werden auf Distanzhaltern montiert
- Sperrholzplatte als Isolator und Träger
- Durchbrüche im Gehäuse für Stecker

Empfänger Selbstbau

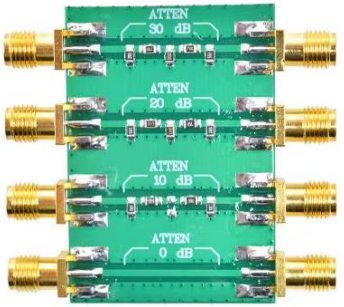
- Arduino Leonardo
  - Hat eigenes USB-Interface zum Programmieren
  - Reicht für Stromversorgung des Moduls selbst
  - Steuert das Si 5351 Modul an
  - Erweiterung zur Ansteuerung eines Displays mög



- 12 Versorgung und 5V Versorgung
  - Der Arduino hat einen 5V Regler an Bord versorgt damit die anderen 5V Verbraucher
  - Via Bias-T Modul Speisung des LNB

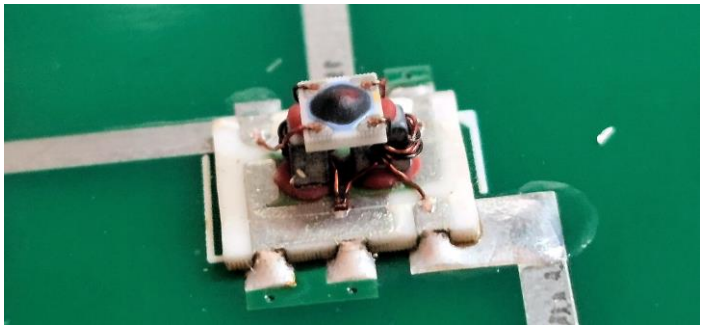
Empfänger Selbstbau

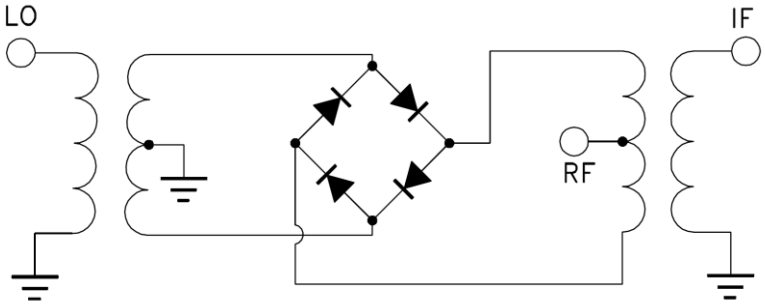
- Mischer RMS-11 zentrales Bauteil des Projekts
  - Signal vom LNB wird mit dem des Si 5351 „verbunden“
  - Daraus entsteht ein Gemisch von Signalen
  - Sowohl über als auch unter der Einzelsignale
  - Daraus wird üblicherweise ein Signal herausgefiltert und weiter genutzt dieses Projekt verzichtet auf den Filter
  - Signal wird gedämpft, um den Empfänger nicht zu übersteuern

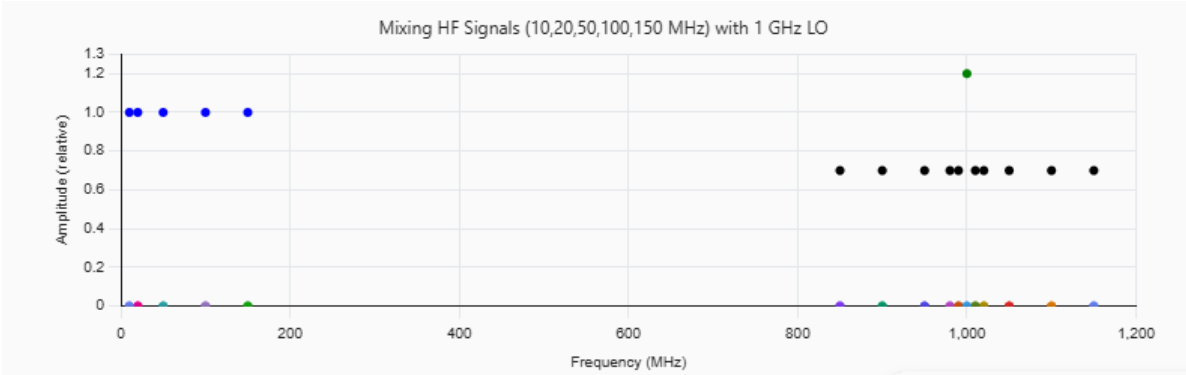


Der Mischer

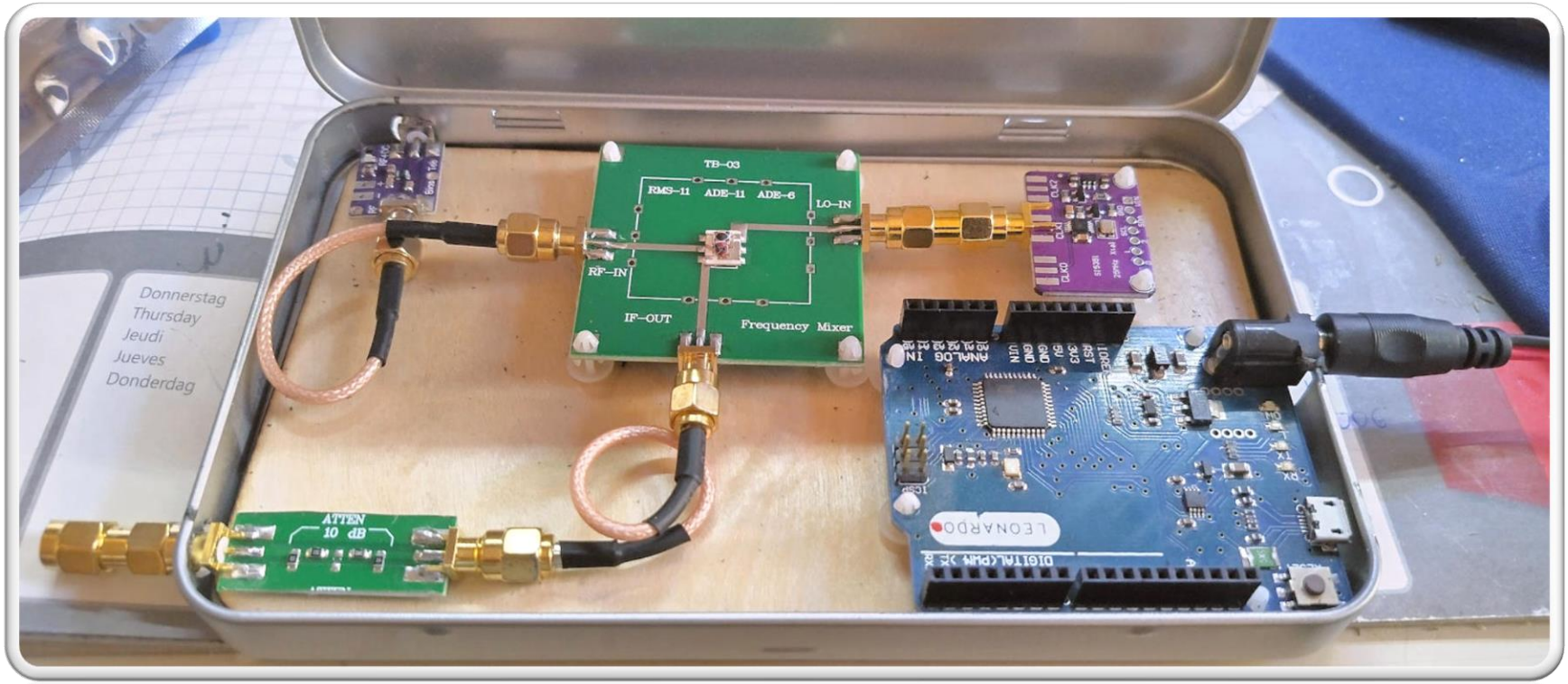
- Mischer RMS-11 zentrales Bauteil des Projekts





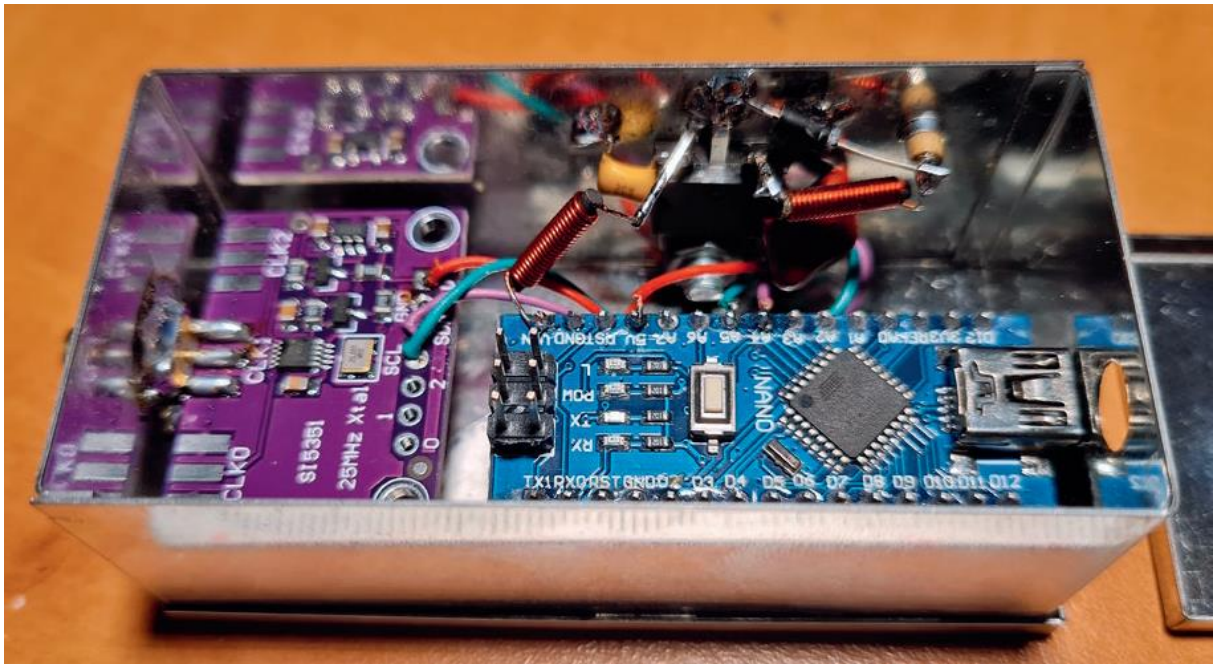


Einbau Vorschlag



Aufbau CQ-QSO 05-06 2024

Quelle *Frits ON4CCO* in der belgischen Clubzeitschrift (flämisch / französisch)

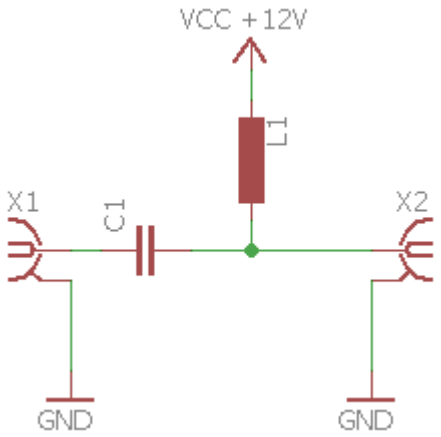


LNB ausmessen

- TV LNBs transferieren das Signal vom 10GHz herunter in den 900MHz Bereich
- Im LNB arbeitet ein Mischer der etwa 25MHz nutzt und daher etwa Faktor 390 die Signale wandelt.
  - Der die Frequenz ist instabil, bei Profi-Sets wird der Generator via externer über GPS gewonnener Signala stabilisiert
  - Unser Empfänger wird daher laufend korrigiert werden müssen

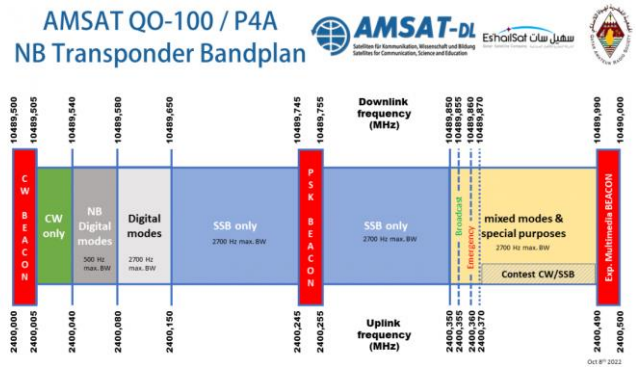
LNB ausmessen SETUP

- LNB benötigt Stromversorgung
  - Wird ins HF Kabel eingespeist über ein Bias-T
- Im Spektrum am Analysator sieht man die Rauschglocke die der LNB anliefert



Schüssel Montage

- Position mit Sat APP am Streichelphone suchen z.B. SATFINDER
  - 167,9° / 34° LNB -8,1°
- An der Position ist ein TV Satellit Es‘hail 2
- Auf 10.489.500kHz (~10GHz) ist eine Zweiton CW Bake zu hören
- SSB QSOs siehe Bandplan

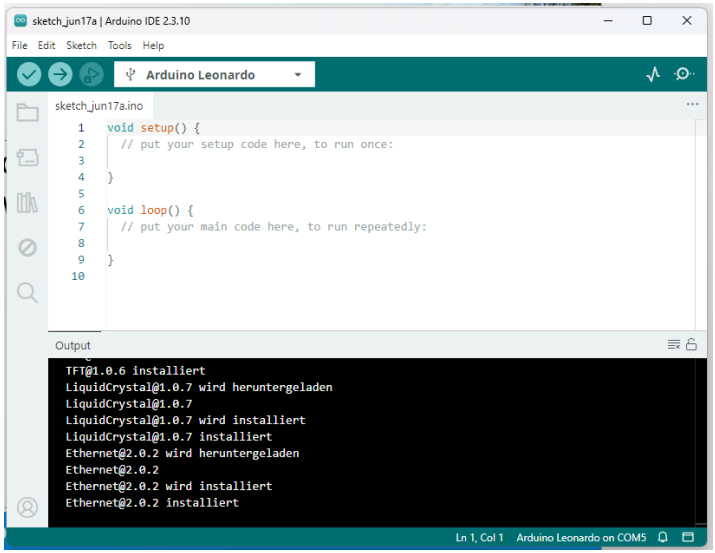


Arduino Software

- Es gibt mehrere Plattformen für Programm Entwicklung. IDE's am PC oder WEB Services
  - Wir werden die PC-IDE nutzen gibt's für PC, Linux, MAC
  - Aufpassen wegen DeviceTreiber, Leonardo sollte überall automatisch konfiguriert werden ohne Ärger
- Programm muß nur die Frequenz am Si5351 einstellen sonst nix
  - Source gibts am WEB zum Download
  - Das Programm kann auch via Copilot erstellt werden durch simple Prompts

Arduino IDE

- IDE Download von https://www.arduino.cc/en/software/
- Board wird automatisch erkannt, Bibliotheken installiert



Das Programm

Bibliotheken einbinden

Starten

Ist Si 5351 überhaupt da?

Frequenz setzen

Über Status berichten

```
20  #include "si5351.h"
21  #include "Wire.h"
22  Si5351 si5351;
23  void setup()
24  {
25    bool i2c_found;
26    // Start serial and initialize the Si5351
27    Serial.begin(57600);
28    i2c_found = si5351.init(SI5351_CRYSTAL_LOAD_8PF, 0, 0);
29    if(!i2c_found)
30    {
31      Serial.println("Device not found on I2C bus!");
32    }
33    // Set CLK1 to output 103 MHz
34    si5351.set_freq(9841330000ULL, SI5351_CLK1);
35    // Set CLK0 to output 119 MHz
36    si5351.set_freq(10980180000ULL, SI5351_CLK0);
37    // Query a status update and wait a bit to let the Si5351 populate the
38    // status flags correctly.
39    si5351.update_status();
40    delay(500);
41  }
42
43  void loop()
44  {
45    // Read the Status Register and print it every 10 seconds
46    si5351.update_status();
47    Serial.print("SYS_INIT: ");
48    Serial.print(si5351.dev_status.SYS_INIT);
49    Serial.print("  LOL_A: ");
50    Serial.print(si5351.dev_status.LOL_A);
51    Serial.print("  LOL_B: ");
52    Serial.print(si5351.dev_status.LOL_B);
53    Serial.print("  LOS: ");
54    Serial.print(si5351.dev_status.LOS);
55    Serial.print("  REVID: ");
56    Serial.println(si5351.dev_status.REVID);
57    delay(10000);
58  }
```

KI das Programm schreiben lassen

- Prompt für Copilot:
  `schreibe ein Programm für Arduino Leonardo das einen Si5351 Frequenzgenerator auf 100MHz einstellt`
- Ergebnis:
  - Simples Programm das ausreichend viel macht

```
#include <Wire.h>
#include <si5351.h>

Si5351 si5351;

void setup() {
  Wire.begin();

  // Initialisierung
  si5351.init(SI5351_CRYSTAL_LOAD_8PF, 0);

  // Ausgang aktivieren
  si5351.output_enable(SI5351_CLK0, 1);

  // Frequenz setzen (in Hz * 100!)
  // 100 MHz = 100 000 000 Hz
  si5351.set_freq(10000000000ULL, SI5351_CLK0);
}

void loop() {
  // nichts zu tun
}
```

Verbesserung <UP> <DOWN>

```ino
#include <Wire.h>
#include <si5351.h>

Si5351 si5351;

// Pins
#define BTN_UP   4
#define BTN_DOWN 5

// Frequenz (in Hz)
long long freq = 100000000;  // Start: 100 MHz

// Schrittweite
const long step = 10000; // 10 kHz

// Entprellen
unsigned long lastPressTime = 0;
const int debounceDelay = 200;

void setFrequency() {
  si5351.set_freq(freq * 100ULL, SI5351_CLK0);
}

void setup() {
  Wire.begin();

  pinMode(BTN_UP, INPUT_PULLUP);
  pinMode(BTN_DOWN, INPUT_PULLUP);

  si5351.init(SI5351_CRYSTAL_LOAD_8PF, 0);
  si5351.drive_strength(SI5351_CLK0, SI5351_DRIVE_8MA);
  si5351.output_enable(SI5351_CLK0, 1);

  setFrequency();
}
```

```ino
void loop() {
  // Taster UP
  if (digitalRead(BTN_UP) == LOW) {
    if (millis() - lastPressTime > debounceDelay) {
      freq += step;
      setFrequency();
      lastPressTime = millis();
    }
  }

  // Taster DOWN
  if (digitalRead(BTN_DOWN) == LOW) {
    if (millis() - lastPressTime > debounceDelay) {
      freq -= step;
      setFrequency();
      lastPressTime = millis();
    }
  }
}
```

Zusätzlich OLED Display via I²C

```ino
#include <Wire.h>
#include <si5351.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
Si5351 si5351;

// Taster
#define BTN_UP   4
#define BTN_DOWN 5

// Frequenz (Hz)
long long freq = 100000000; // 100 MHz
const long step = 10000;    // 10 kHz

// Entprellen
unsigned long lastPressTime = 0;
const int debounceDelay = 200;

// ========================
// Frequenz setzen
// ========================
void setFrequency() {
  si5351.set_freq(freq * 100ULL, SI5351_CLK0);
}

// Anzeige aktualisieren
// ========================
void updateDisplay() {
  display.clearDisplay();

  display.setTextSize(2);
  display.setTextColor(SSD1306_WHITE);

  // Frequenz schön formatieren (MHz.kHz)
  long mhz = freq / 1000000;
  long khz = (freq % 1000000) / 1000;

  char buffer[20];
  sprintf(buffer, "%ld.%03ld MHz", mhz, khz);

  display.setCursor(0, 20);
  display.println(buffer);

  display.display();
}

void setup() {
  Wire.begin();

  pinMode(BTN_UP, INPUT_PULLUP);
  pinMode(BTN_DOWN, INPUT_PULLUP);

  // Si5351
  si5351.init(SI5351_CRYSTAL_LOAD_8PF, 0);
  si5351.drive_strength(SI5351_CLK0, SI5351_DRIVE_8MA);
  si5351.output_enable(SI5351_CLK0, 1);

  // OLED starten
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    while (true); // Fehler
  }

  setFrequency();
  updateDisplay();
}

void loop() {
  // UP
  if (digitalRead(BTN_UP) == LOW) {
    if (millis() - lastPressTime > debounceDelay) {
      freq += step;
      if (freq > 160000000) freq = 160000000;

      setFrequency();
      updateDisplay();
      lastPressTime = millis();
    }
  }

  // DOWN
  if (digitalRead(BTN_DOWN) == LOW) {
    if (millis() - lastPressTime > debounceDelay) {
      freq -= step;
      if (freq < 1000000) freq = 1000000;

      setFrequency();
      updateDisplay();
      lastPressTime = millis();
    }
  }
}
```