HITACHI
Inspire the Next

EMC設計対策コンテスト

日立Astemo 株式会社
第1インバータ設計部
河村　倖助
第2インバータ設計部
竹本　寛晴

2023/11/9

Astemo

HITACHI
Inspire the Next

1.アプリケーションについて

2.測定仕様

3.プレ測定〔社内環境〕

4.近傍磁界測定

5.対策内容

6.対策結果

7.結論

Astemo

1.アプリケーションについて〔概要〕

製作物：温湿度・気圧計

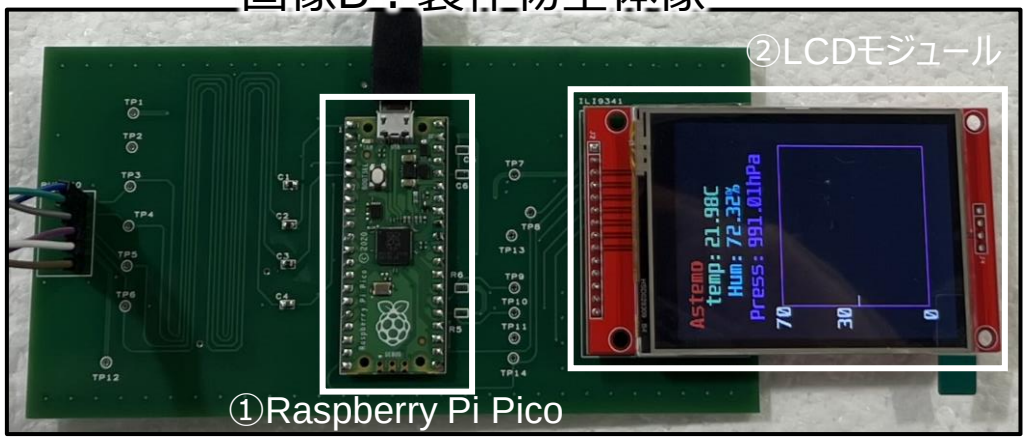
・アプリケーションについて
　使用部材
　　①Raspberry Pi Pico：マイコン動作周波数 125MHz
　　　水晶体動作周波数　12MHz
　　②LCDモジュール(ディスプレイ) ：SPI通信速度 50MHz
　　③温度センサ： SPI通信速度 1MHz
・動作について
　センサー部が温湿度と気圧を検知。その情報をRaspberry Pi Pico
　が逐次、読み取りに行きその情報をLCDモジュールへ表示される。
　表示内容として画面上部にリアルタイムの数値データを表示させ、温度に
　関しては、その変化を感覚的に認識できるようプロットで表示させている。

製作理由

近年、特に夏の記録的猛暑で熱中症・脱水症の増加しており、注意を促されているが、
自分の**現在地の温湿度**は余り意識する機会はない…
身近にあるものではスマホのアプリ等の情報のため詳細なレベルだと基本的には市町村
レベルのものが主。
そのため、なるべく手軽な部材できる個人用途の温湿度計を作製しました。

温湿度・気圧センサー(③)

画像B：製作物全体像



1.アプリケーションについて〔ソフト〕

HITACHI
Inspire the Next

ソフト〔言語：MicroPython〕

main.py

```python
from machine import Pin, SPI
from xglcd_font import XglcdFont
import ili9341
import time
from bme280_spi import BME280
import math

times = 0

# SPI設定(ILI9341ディスプレイ用)
spi_display = SPI(0, baudrate=51200000, sck=Pin(18), mosi=Pin(19))
display = ili9341.Display(spi_display, dc=Pin(16), cs=Pin(17), rst=Pin(14))
font = XglcdFont("Unispace12x24.py", 12, 24)

# SPI設定(BME280センサー用)
spi_sensor = SPI(0, baudrate=1000000, sck=Pin(2), mosi=Pin(3), miso=Pin(0))
cs = Pin(1, Pin.OUT, value=1)

# ディスプレイ表記内容
display.draw_text(10, 10, "Astemo", font, ili9341.color565(255, 0, 0))
display.draw_text(35, 30, "temp:", font, ili9341.color565(96, 255, 128))
display.draw_text(46, 55, "Hum:", font, ili9341.color565(80, 208, 255))
display.draw_text(20, 80, "Press:", font, ili9341.color565(160, 32, 255))
```

main.py

```python
display.draw_text(35, 30, "temp:", font, ili9341.color565(96, 255, 128))
display.draw_text(46, 55, "Hum:", font, ili9341.color565(80, 208, 255))
display.draw_text(20, 80, "Press:", font, ili9341.color565(160, 32, 255))
display.draw_rectangle(35, 120, 180, 170, ili9341.color565(255,0,255))
display.draw_text(10, 110, "70", font, ili9341.color565(255, 255, 255))
display.draw_text(10, 180, "30", font, ili9341.color565(255, 255, 255))
display.draw_text(20, 280, "0", font, ili9341.color565(255, 255, 255))

while True:
    # BME280からデータを読み取る
    bme = BME280(spi=spi_sensor, cs=cs)
    temp, press, hum = bme.values

    # BME280からデータを読み取る(単位抜きデータ)
    temp_N = bme.read_compensated_data()[0]/100
    press_N = bme.read_compensated_data()[1]/25600
    hum_N = bme.read_compensated_data()[2]/1024

    # 背景色(黒)
    background_color = ili9341.color565(0, 0, 0)

    # 温度、湿度、気圧をILI9341ディスプレイに表示
    display.draw_text(100, 30, temp, font, ili9341.color565(96, 255, 128))
    display.draw_text(100, 55, hum, font, ili9341.color565(80, 208, 255))
```

1~6行目：モジュールのインポート
10~13行目：ディスプレイとRaspberyPiPico間のSPIインスタンスの生成です。
周波数(baudrate)、クロックピン(sck)、出力ピン(mosi) を指定。
Pin番号はRaspberry Pi PicoのGPIO番号 (GPnn) に合わせる。
12行目でili9341のDisplay関数を呼び出してモジュールを表示。
15~17行目：” 10~13行目”と同じ要領で周波数の指定とPinの指定。
19~27行目：ディスプレイに表示させるテキストとフォントとその位置をここで指定、加えて色をcolor565(R,G,B)関数で16bit カラーで指定

Astemo

1.アプリケーションについて〔ソフト〕

HITACHI
Inspire the Next

ソフト〔言語：MicroPython〕

main.py

```
21 display.draw_text(35, 30, "temp:", font, ili9341.color565(96, 255, 128))
22 display.draw_text(46, 55, "Hum:", font, ili9341.color565(80, 208, 255))
23 display.draw_text(20, 80, "Press:", font, ili9341.color565(160, 32, 255))
24 display.draw_rectangle(35, 120, 180, 170, ili9341.color565(255,0,255))
25 display.draw_text(10, 110, "70", font, ili9341.color565(255, 255, 255))
26 display.draw_text(10, 180, "30", font, ili9341.color565(255, 255, 255))
27 display.draw_text(20, 280, "0", font, ili9341.color565(255, 255, 255))
28
29 while True:
30     # BME280からデータを読み取る
31     bme = BME280(spi=spi_sensor, cs=cs)
32     temp, press, hum = bme.values
33
34     # BME280からデータを読み取る(単位抜きデータ)
35     temp_N = bme.read_compensated_data()[0]/100
36     press_N = bme.read_compensated_data()[1]/25600
37     hum_N = bme.read_compensated_data()[2]/1024
38
39     # 背景色(黒)
40     background_color = ili9341.color565(0, 0, 0)
41
42     # 温度、湿度、気圧をILI9341ディスプレイに表示
43     display.draw_text(100, 30, temp, font, ili9341.color565(96, 255, 128))
44     display.draw_text(100, 55, hum, font, ili9341.color565(80, 208, 255))
```

main.py

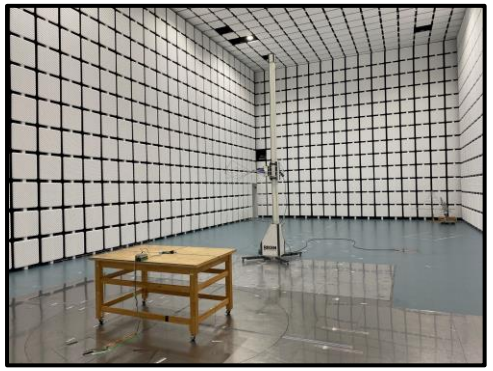
```
32     temp, press, hum = bme.values
33
34     # BME280からデータを読み取る(単位抜きデータ)
35     temp_N = bme.read_compensated_data()[0]/100
36     press_N = bme.read_compensated_data()[1]/25600
37     hum_N = bme.read_compensated_data()[2]/1024
38
39     # 背景色(黒)
40     background_color = ili9341.color565(0, 0, 0)
41
42     # 温度、湿度、気圧をILI9341ディスプレイに表示
43     display.draw_text(100, 30, temp, font, ili9341.color565(96, 255, 128))
44     display.draw_text(100, 55, hum, font, ili9341.color565(80, 208, 255))
45     display.draw_text(100, 80, press, font, ili9341.color565(160, 32, 255))
46
47     # 温度プロットをILI9341ディスプレイに表示
48     display.draw_rectangle(times+36,121,1,168, ili9341.color565(0, 0, 0))
49     display.draw_hline(times+35,math.floor(temp_N)*-2 + 250, 5, ili9341.color565(255, 255, 255))
50     if times < 174:
51         times = times + 1
52     else:
53         times = 0
54     time.sleep(1)
```

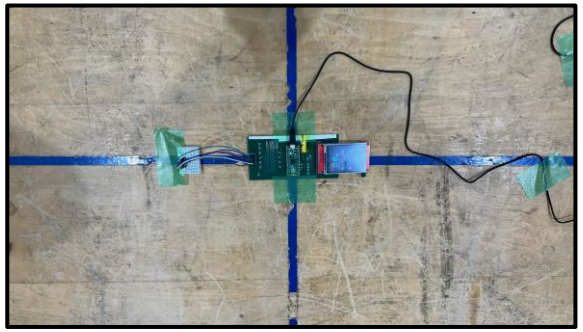
29~37行目：以降のプログラムを無限ループでプログラムを実行します。温湿度センサーからのデータを取得
39~45行目：背景色の指定をし、“温度”、“湿度”、“気圧”の読み取った値のフォントとその位置をここで指定、
加えて色をcolor565(R,G,B)関数で16bit カラーで指定
47~54行目：温度のリアルタイムプロットプロットを実行します。現在の温度値をプロットしていき174secでループし、古い物プロットを消去される。
また上記のプロットを1sec間隔で実施し以降は“While True”から無限ループになる。

Astemo

2.測定仕様

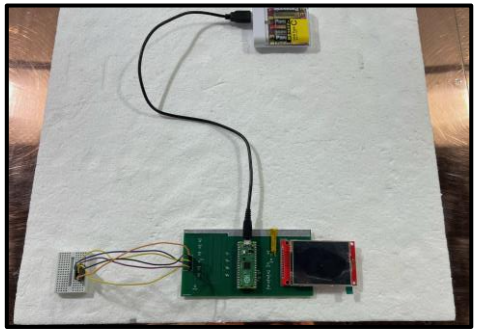
ADOX福岡様測定環境





ターンテーブルなし

社内プレ測定環境



測定仕様

・アンテナ高さ：2m
・測定周波数：30～1000MHz
・測定距離：3m
・測定：EUTを回転させまた受信アンテナを 2m/3 m
　　の範囲で昇降させて、受信レベルが最大となるようにする
・電源：設備電源を使用

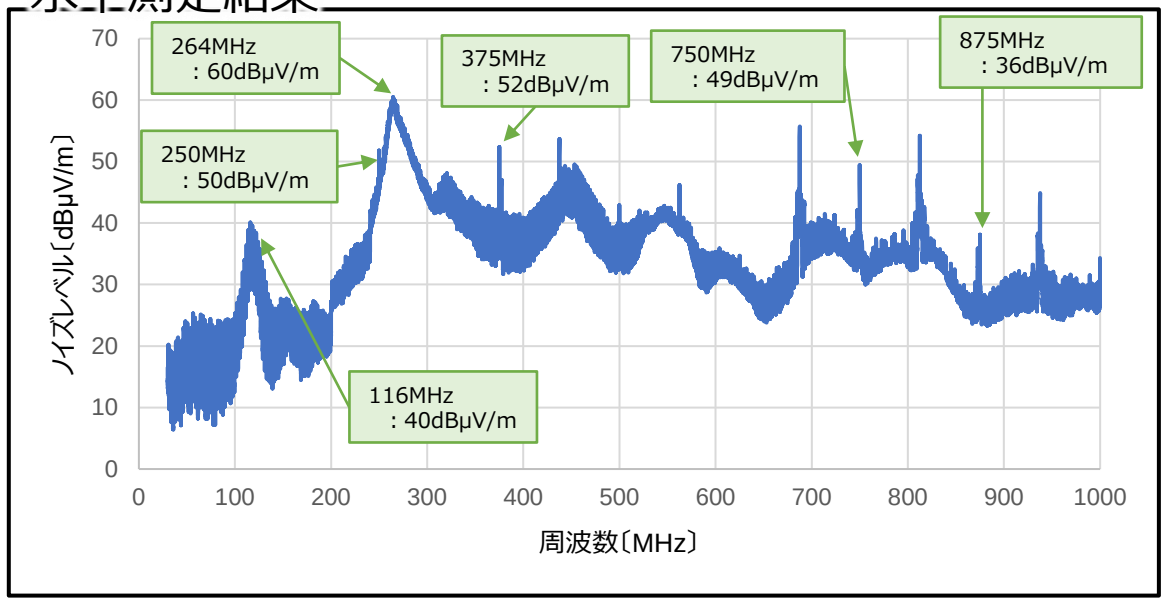
測定仕様

・アンテナ高さ：2m
・測定周波数：30～200・200～1000MHz
・測定距離：1m
・測定：EUTを机上に置きグランドテーブルから発砲スチロールで
　　浮かせ、受信アンテナを1mの高さで固定
・電源：市販の電池BOXを使用

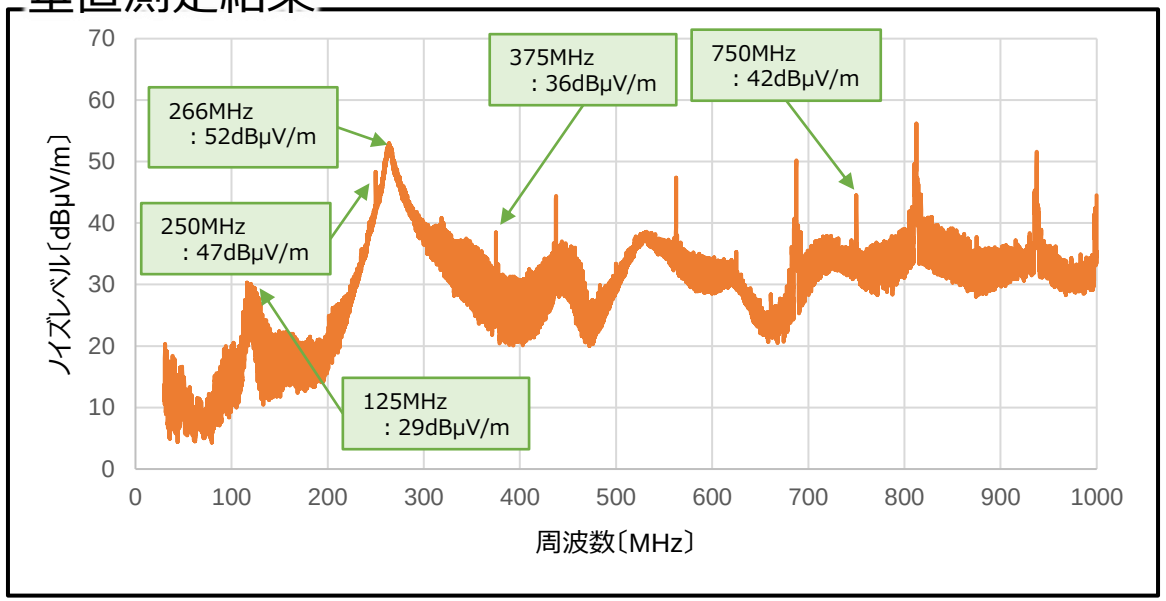
3.プレ測定〔社内環境〕

HITACHI
Inspire the Next

水平測定結果



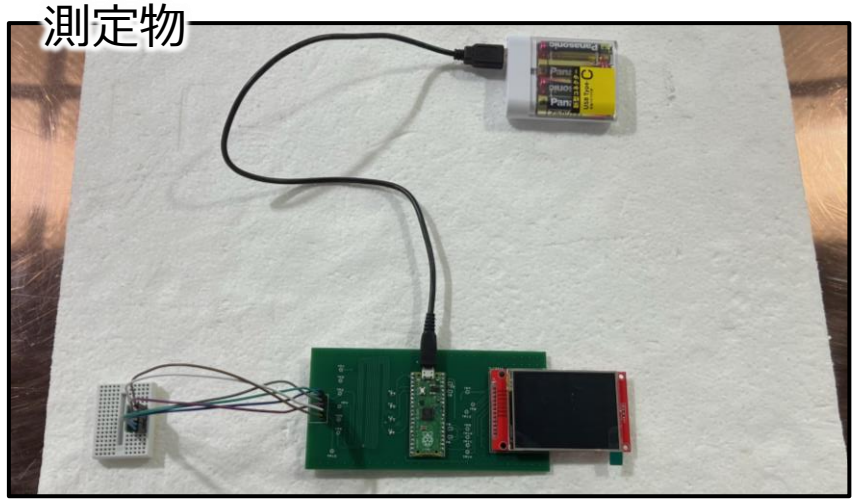
垂直測定結果



ノイズ課題

・120MHz近傍で40dB(水平)のノイズピークが発生
・270MHz近傍で60dB(水平)ノイズピークが発生
・125MHzの整数倍の周波数でノイズピークが発生
⇒上記３つの項目を近傍磁界で測定し原因解析を実施

測定物

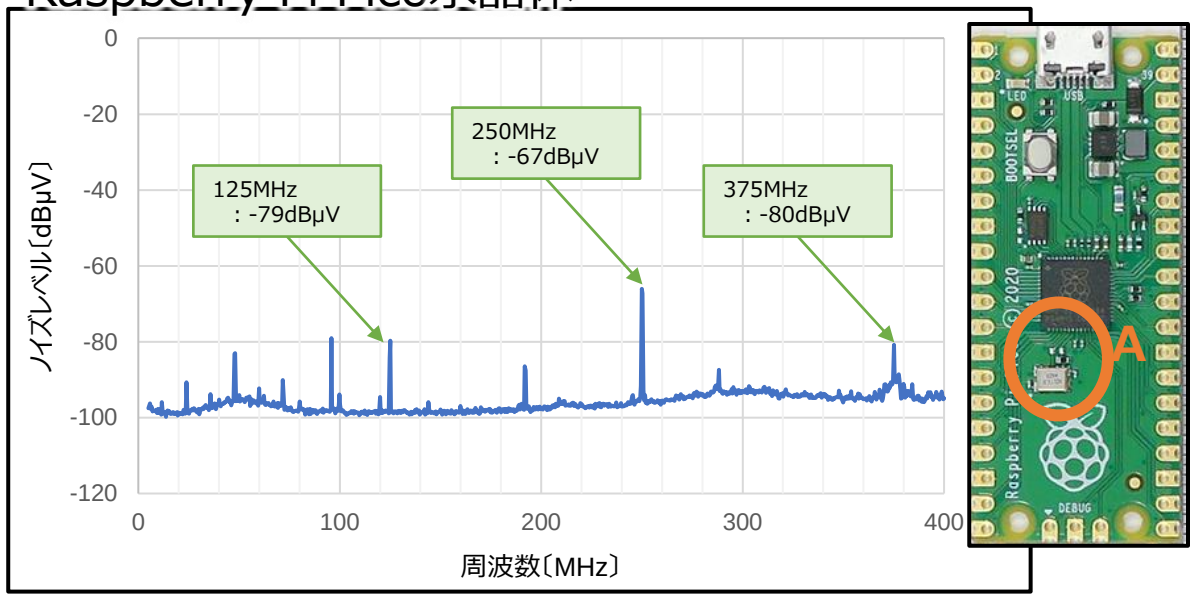


Astemo

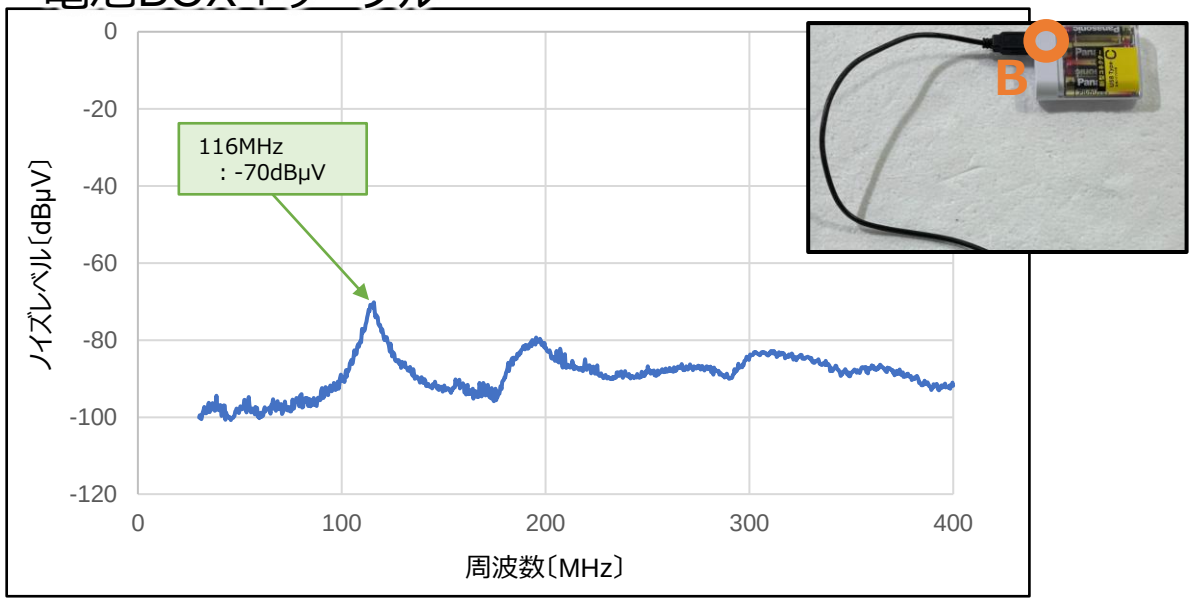
4.近傍磁界測定

HITACHI
Inspire the Next

Raspberry_Pi_Pico水晶体



電池BOX＋ケーブル



近傍磁界結果

・125MHzの整数倍の周波数でノイズピークが発生
⇒ Raspberry Pi Picoのマイコン動作周波数が125MHzであることから測定点Aを確認。Picoのみ動作させた状態で125MHzの整数倍の周波数でノイズピークを確認できたためノイズ要因と判断

・120MHz近傍で40dB(水平)のノイズピークが発生
⇒電池BOX+ケーブルで確認した際に120MHz近傍でピークを確認できたためノイズ要因と判断

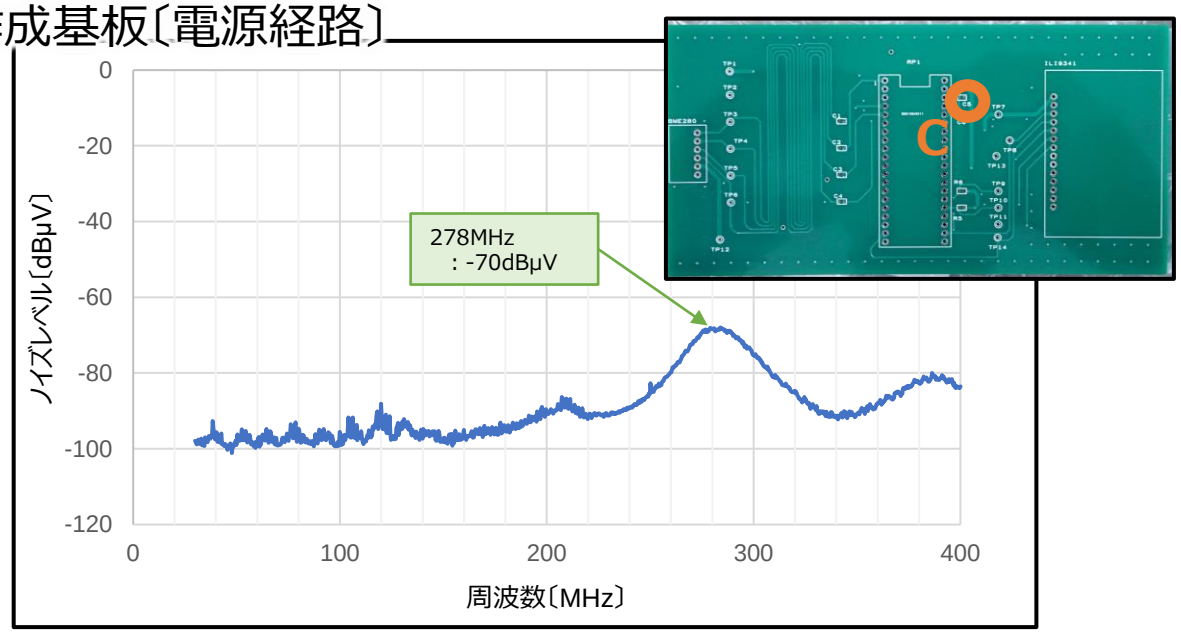
119-4146-00 近接界プローブ

GSP-818 スペクトラムアナライザ

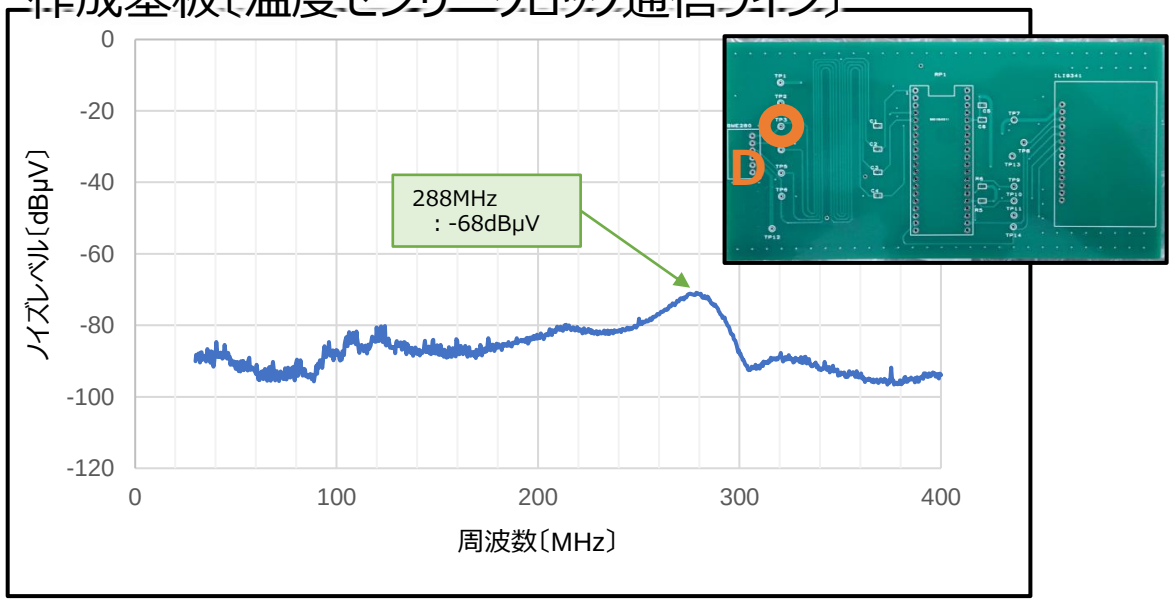
Astemo

4.近傍磁界測定

作成基板〔電源経路〕



作成基板〔温度センサークロック通信ライン〕



近傍磁界結果

・270MHz近傍で60dB(水平)ノイズピークが発生

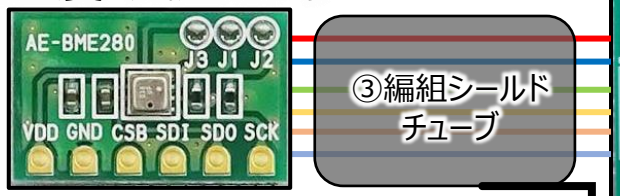
⇒ Raspberry Pi Picoの電源ノイズが実装パターンに乗りピークが発生していると仮定。Picoのみを基板に搭載した状態で測定点Cを確認。270MHz近傍でノイズピークを確認できたためノイズ要因の一つと判断。また温度センサーとのクロック通信線でも同一のノイズが乗りピークが発生していることを確認

119-4146-00 近接界プローブ

GSP-818　スペクトラムアナライザ

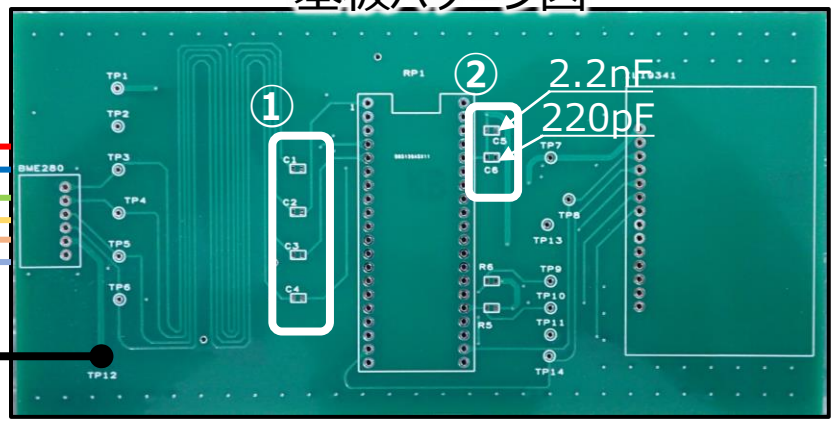
5.対策内容

温湿度・気圧センサー

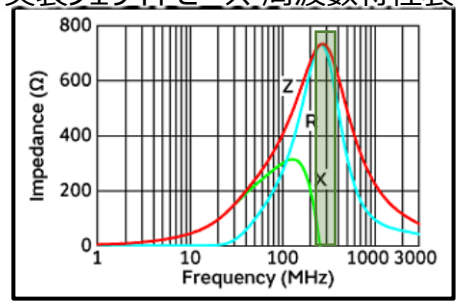


③編組シールドチューブ

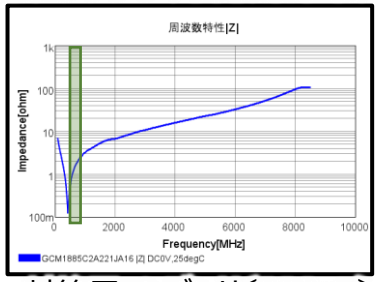
基板パターン図



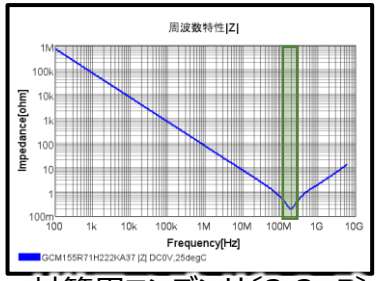
実装フェライトビーズ 周波数特性表



実装コンデンサ 周波数特性表



対策用コンデンサ〔220pF〕



対策用コンデンサ〔2.2nF〕

対策内容

①：RaspberyPiPico - 温湿度・気圧センサーへ”フェライトビーズ”を実装
- 水晶体動作周波数を起因とした実装パターンからのノイズを懸念し、”フェライトビーズ“で抑制するために実装

②：Vout - GND間へバイパスコンデンサ(220pFと2.2nF)を実装
- 電源ループの最小化によるノイズ放出を抑制するために実装

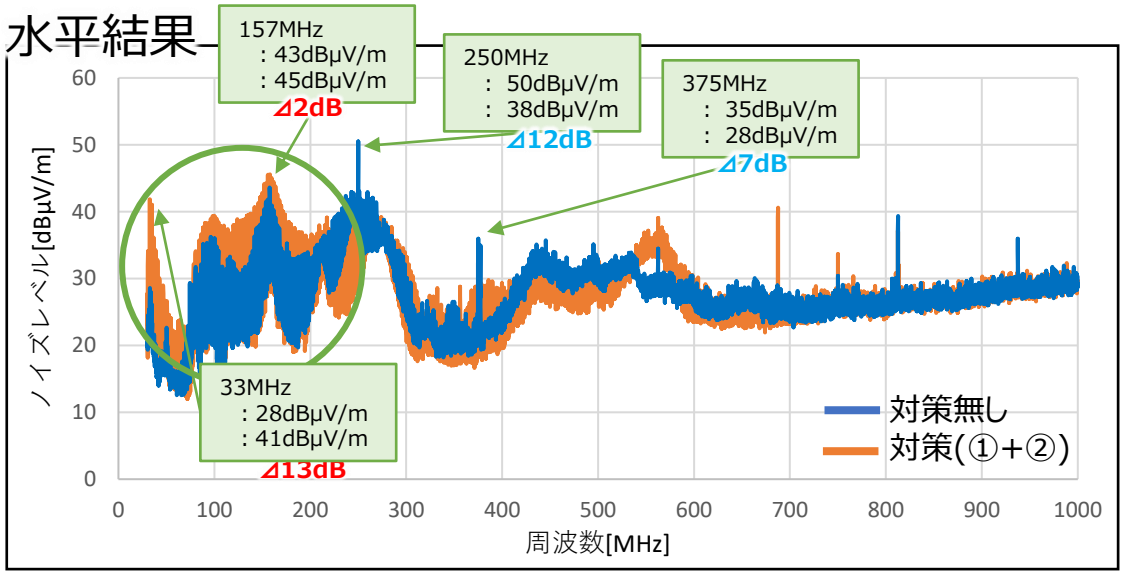
③：RaspberyPiPico – 温湿度・気圧センサー間に編組シールドチューブ追加
-通信線からの放射ノイズを抑制するため実装

対策コスト

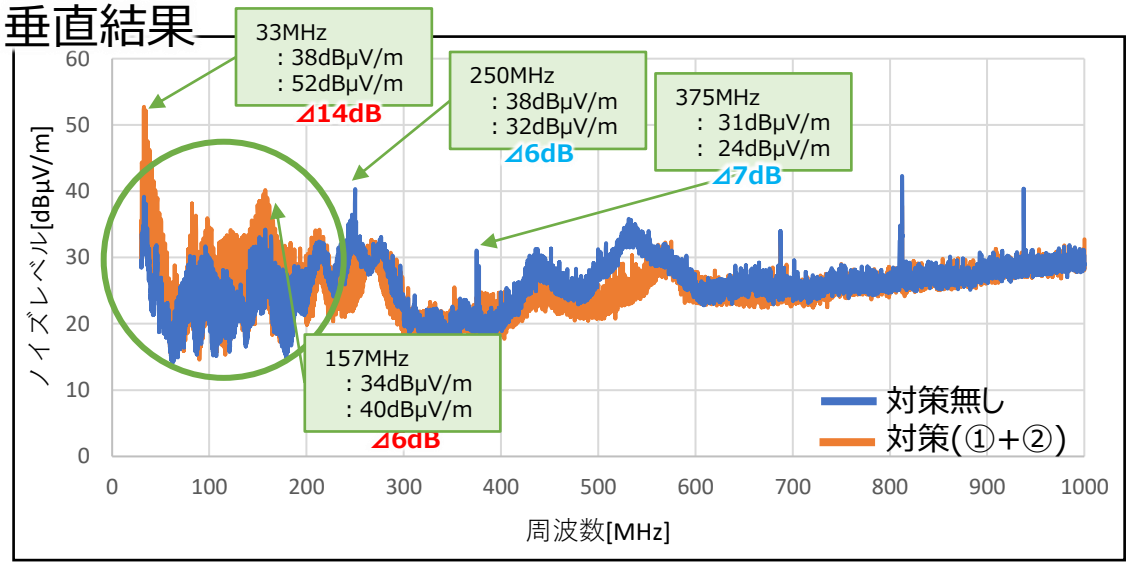
対策項目	価格
PCB設計	￥0
バイパスコンデンサ	￥16
フェライトビーズ	￥53
編組シールドチューブ	￥125
合計	￥141

5.対策結果〔①＋②〕

水平結果



垂直結果

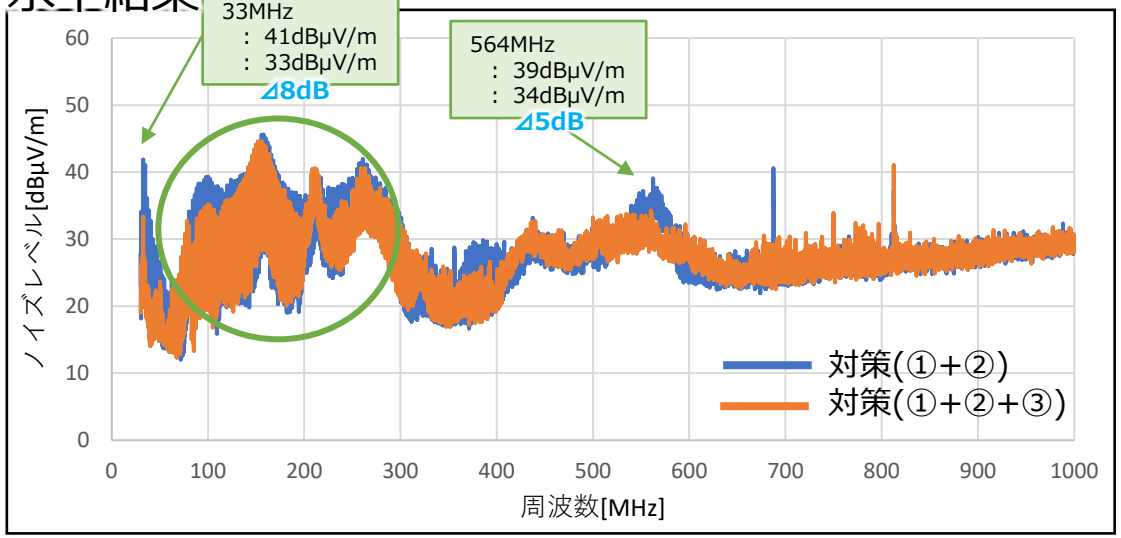


対策結果

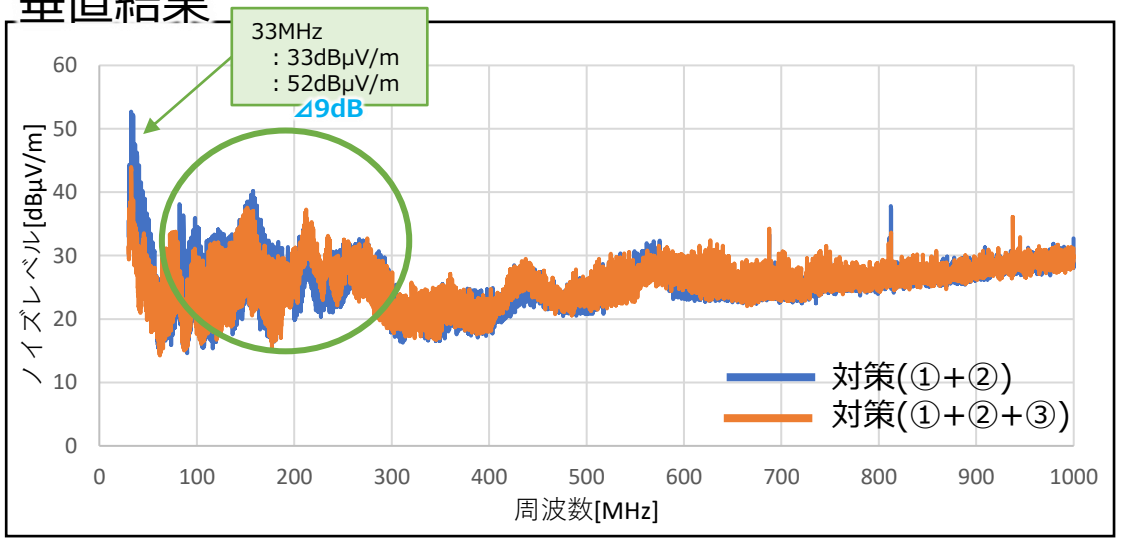
①：RaspberyPiPico - 温湿度・気圧センサーへ”フェライトビーズ”を実装
②：Vout - GND間へバイパスコンデンサ(220pFと2.2nF)を実装
⇒目標としていた250MHzとそれ以降125MHz間隔で発生しているピークが低減
低周波領域(30~200NHz)でノイズが増大してしまった。このノイズ源としてPicoの水晶体動作ノイズの伝導経路が変わったためと思われる。

5.対策結果〔①＋②＋③〕

水平結果



垂直結果



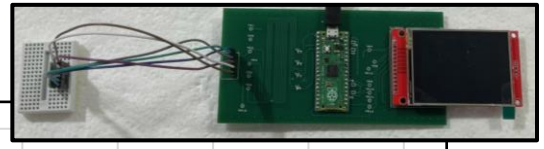
対策結果

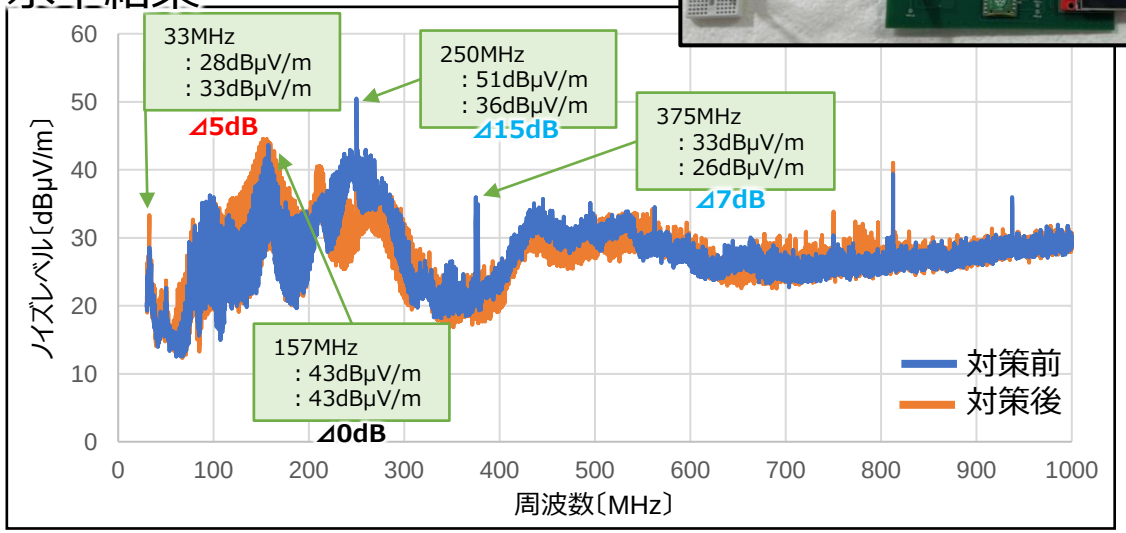
③：RaspberyPiPico – 温湿度・気圧センサー間に編組シールドチューブ追加。
⇒低周波領域でのノイズの主要因と考えていた、ジャンパー線の電源経路からの放射を抑制することで、ノイズのピークである、33MHzでの、水平8dB、垂直9dBのノイズ低減を確認する事ができている。また、低周波領域での全体的なノイズレベルの低減を確認する事ができる。

6.対策結果〔①＋②＋③〕

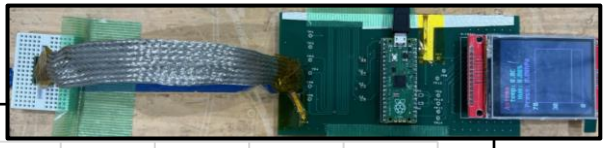
HITACHI
Inspire the Next

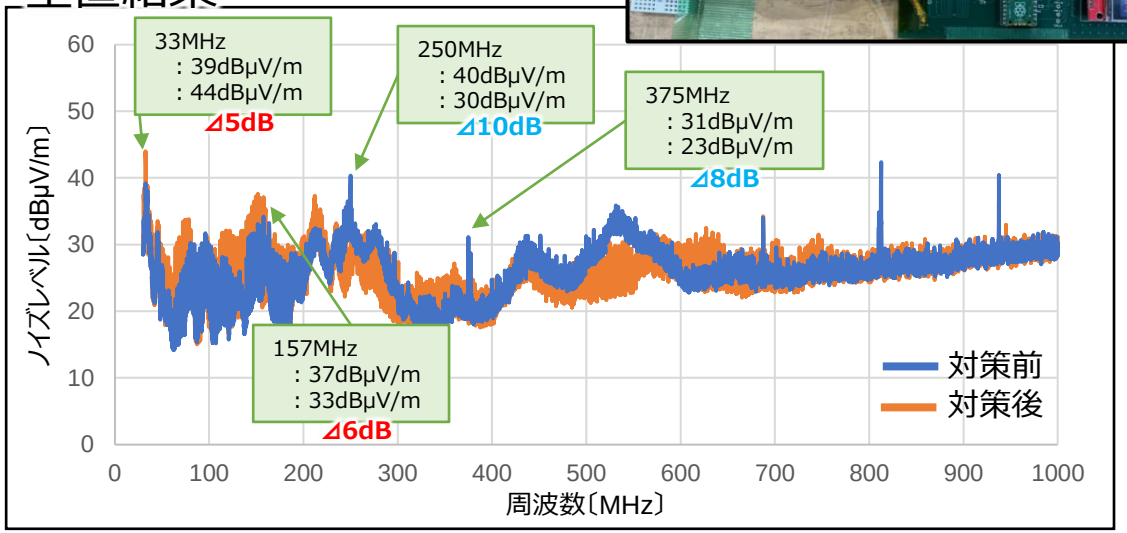
水平結果





垂直結果





結論

・今回の対策効果として、ノイズのピークである250MHzの低減を確認することができた。ノイズの主要因である、Picoからの影響低減のために、フェライトビーズとシールドチューブでの温度センサ信号線への対策及び電源線への対策としてパスコンを実装することで、ノイズの低減効果を確認する事ができた。また、低周波領域のノイズへの追加対策として、フェライトコアを実装することで全体的なノイズ低減を確認したが対策コストを考えた際に、許容できないため本対策では不採用する。以上のことから、コストを踏まえた上で期待通りの対策を行うことができたと考える。
課題として、シールドチューブを対処療法的な対策として採用してしまった。基板の信号線ラインへRCフィルタやLCフィルタを実装できるように設計する事でより安価にノイズ抑制を行うことができるためこれを今後考慮していきたい。

Astemo

HITACHI

Inspire the Next