# Progetto ThereBoard

## Informatica Applicata al Suono

*De Filippis Matteo*
*Mandelli Giacomo*
*Pesando-Gamacchio Gabriele*

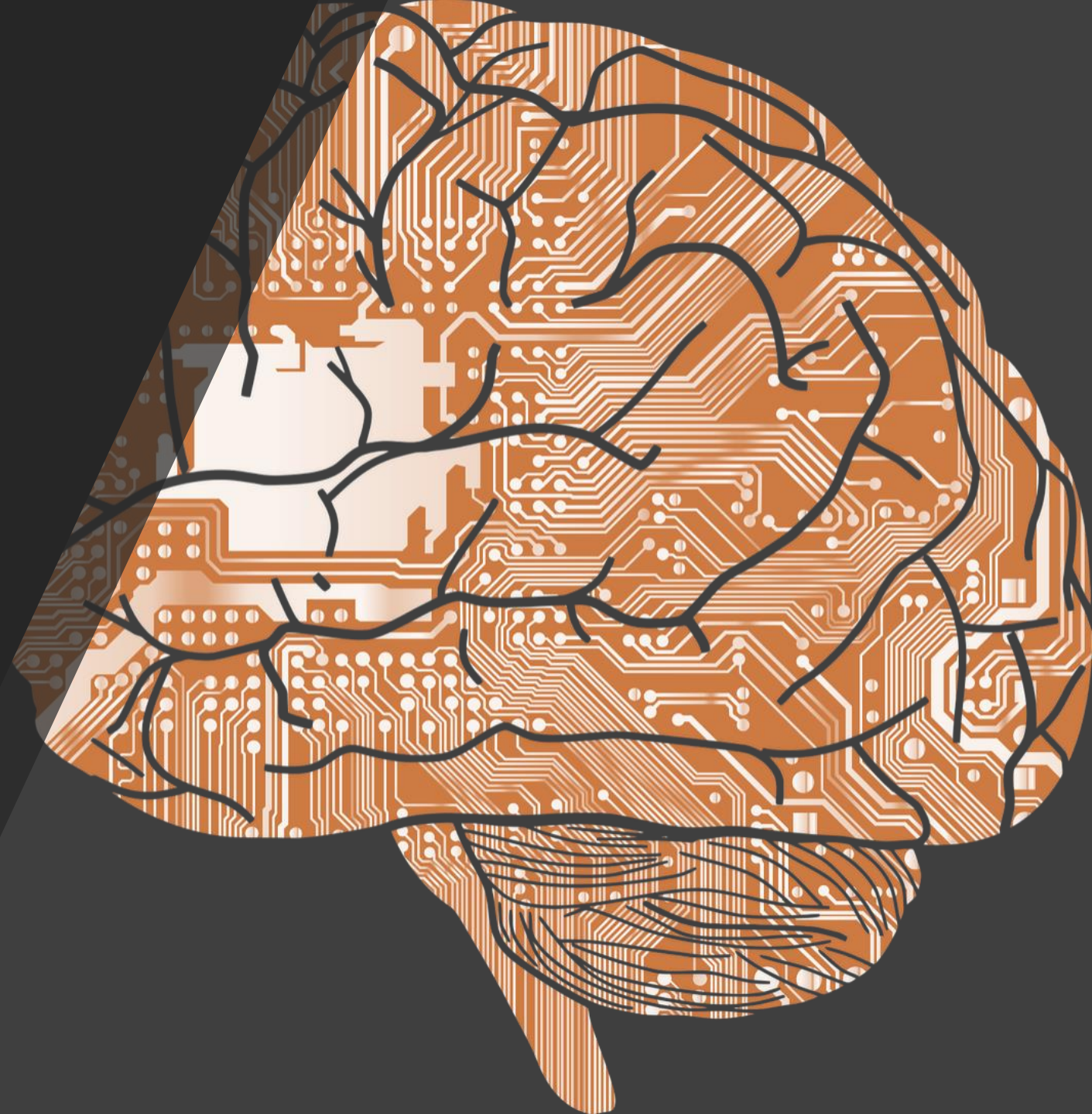
# Introduzione

- Cos’è il Theremin
- Come funziona il Theremin
- Le caratteristiche del Theremin

1. Il Theremin è uno strumento musicale elettronico.
   «*1919 – Lev Sergeevic Termen*»
2. Il Theremin si basa su oscillatori che lavorando in isofrequenza al di fuori dello spettro udibile, producono, per alterazioni delle loro caratteristiche a seguito della presenza delle mani del musicista nel campo d’onda, dei suoni sul principio fisico del battimento.
3. Il Theremin è composto da due antenne e un contenitore dove alloggia l’elettronica.

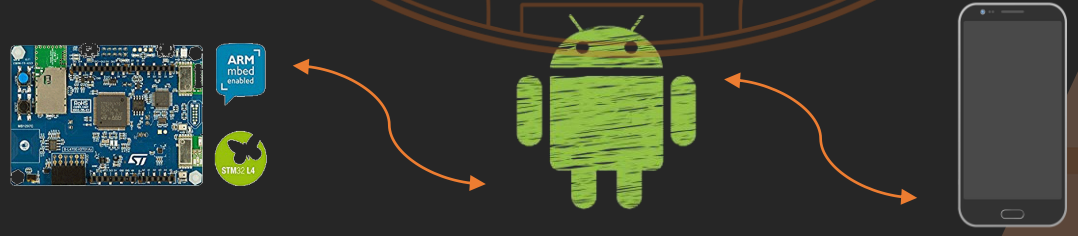
# L’idea Thereboard

- Scopo del progetto ThereBoard
- Introduzione al funzionamento
- Strumenti utilizzazti



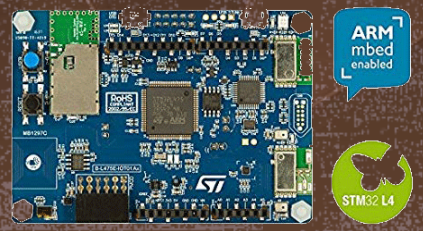
# Disco-l45vg-iot01a

- Sensore di prossimità
- Giroscopio del telefono
- Simulink & Matlab



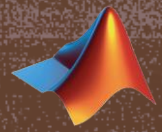
# Step visuali

1. La scheda legge i valori dai sensori.
2. I valori vengono ricevuti sul telefono collegato tramite bluetooth.
3. I valori vengono elaborati sul telefono attraverso opportune funzioni Matlab
4. Il suono del theremin viene emesso tramite l'altoparlante




1

2

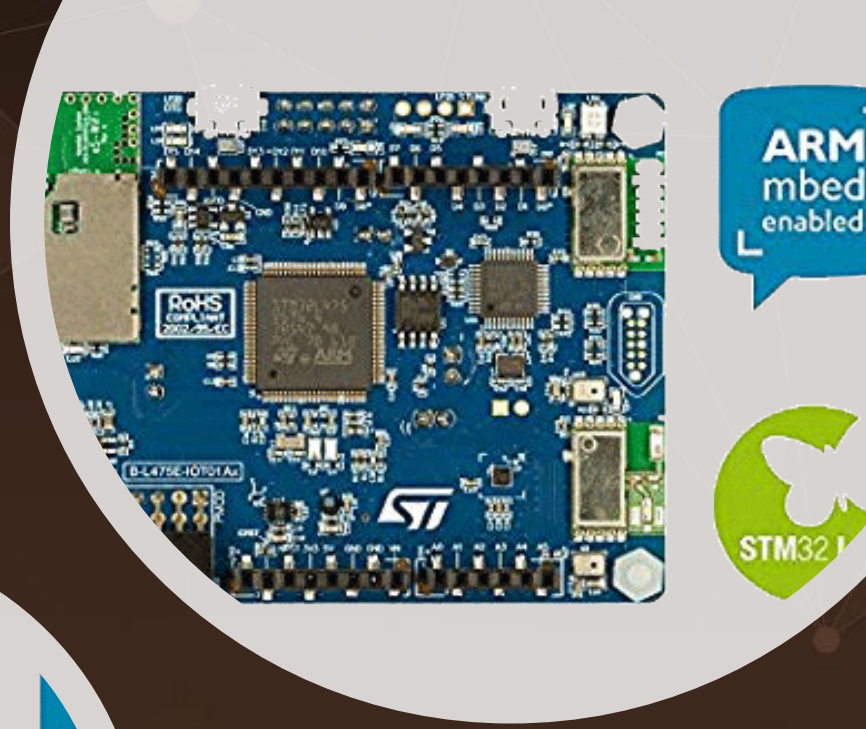


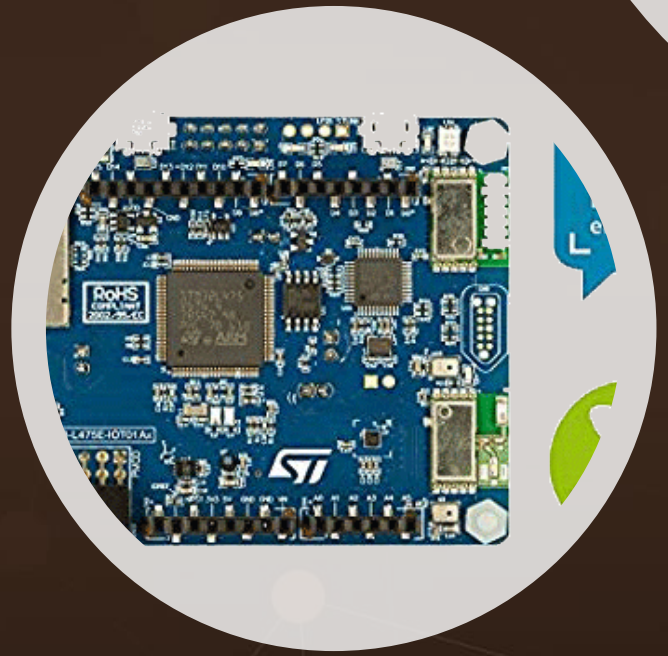
3

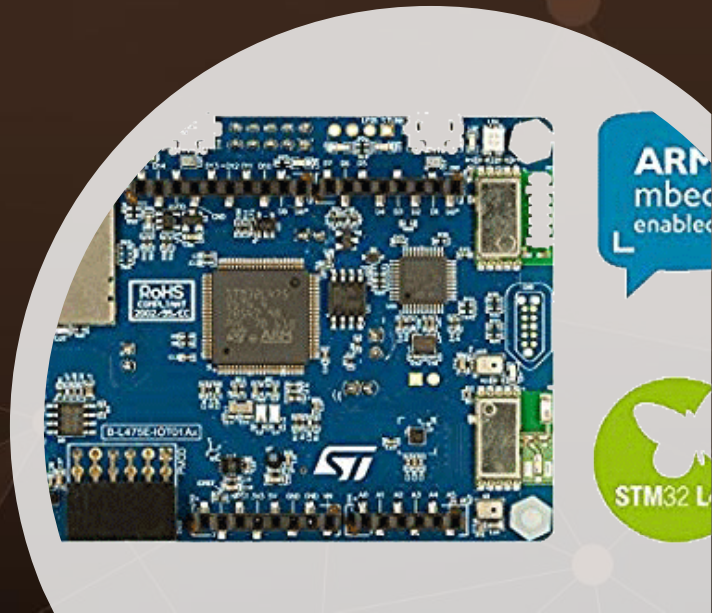
4

# Hardware
# Disco-l45vg-iot01a

- Processore STM32L475VGT6 MCU 80 MHz/100 DMIPS
- connettività wireless : BLE
- Sensore: prossimità
- Connettore: USB OTG



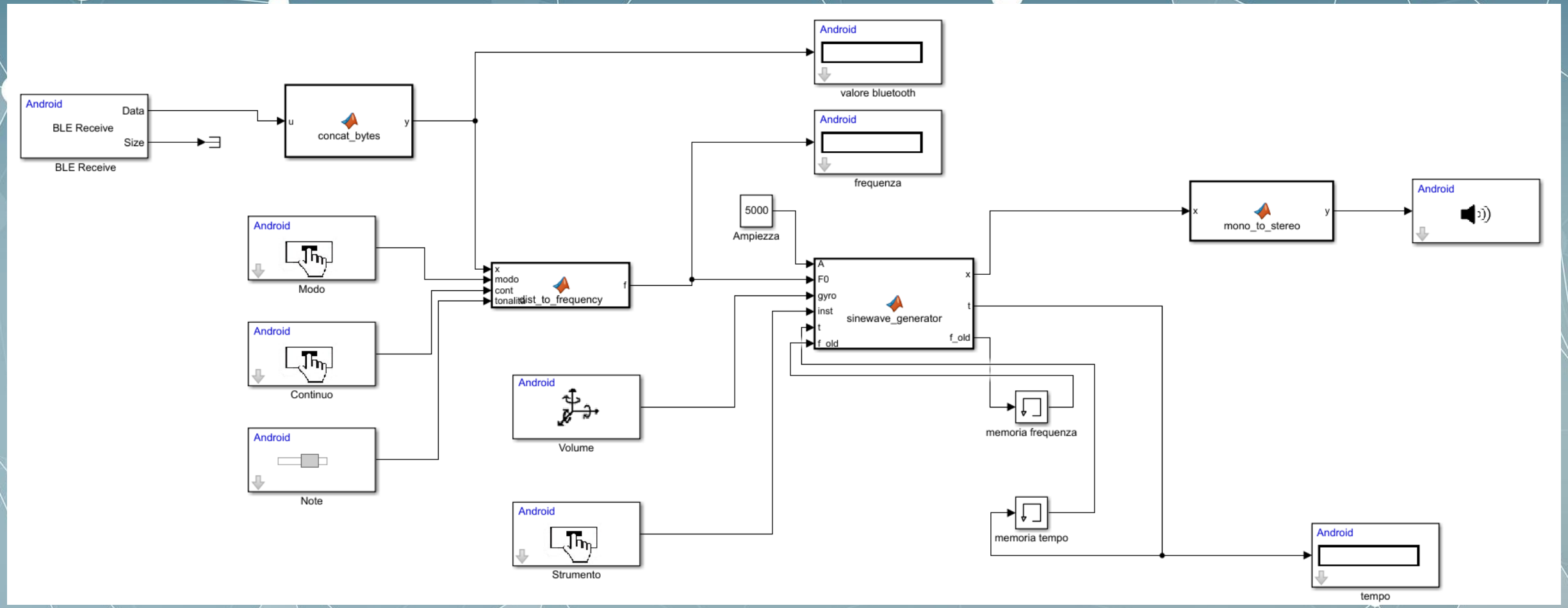




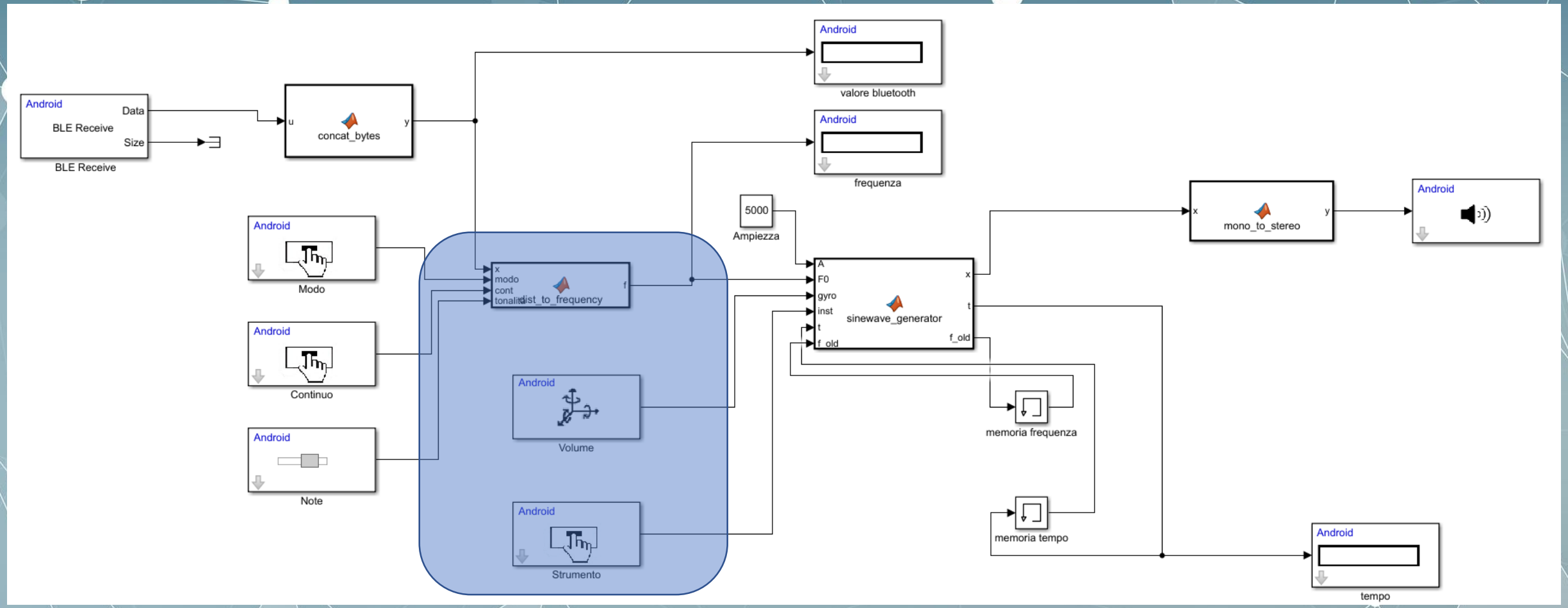
# Sviluppo Thereboard

## *Le innovazioni*

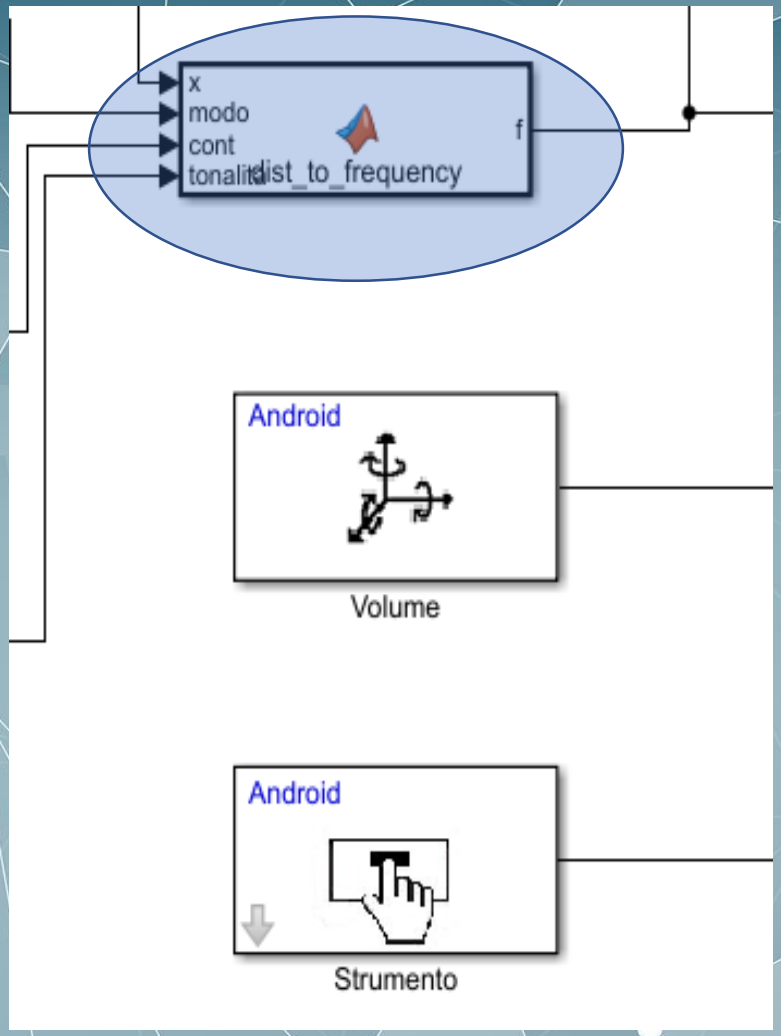
- Continuità approssimata
- Silenzio senza mano
- Scale musicali
- Cambiamento della timbrica
- Scale minori e maggiori
- Modulazione del volume




# Schema generale Simulink



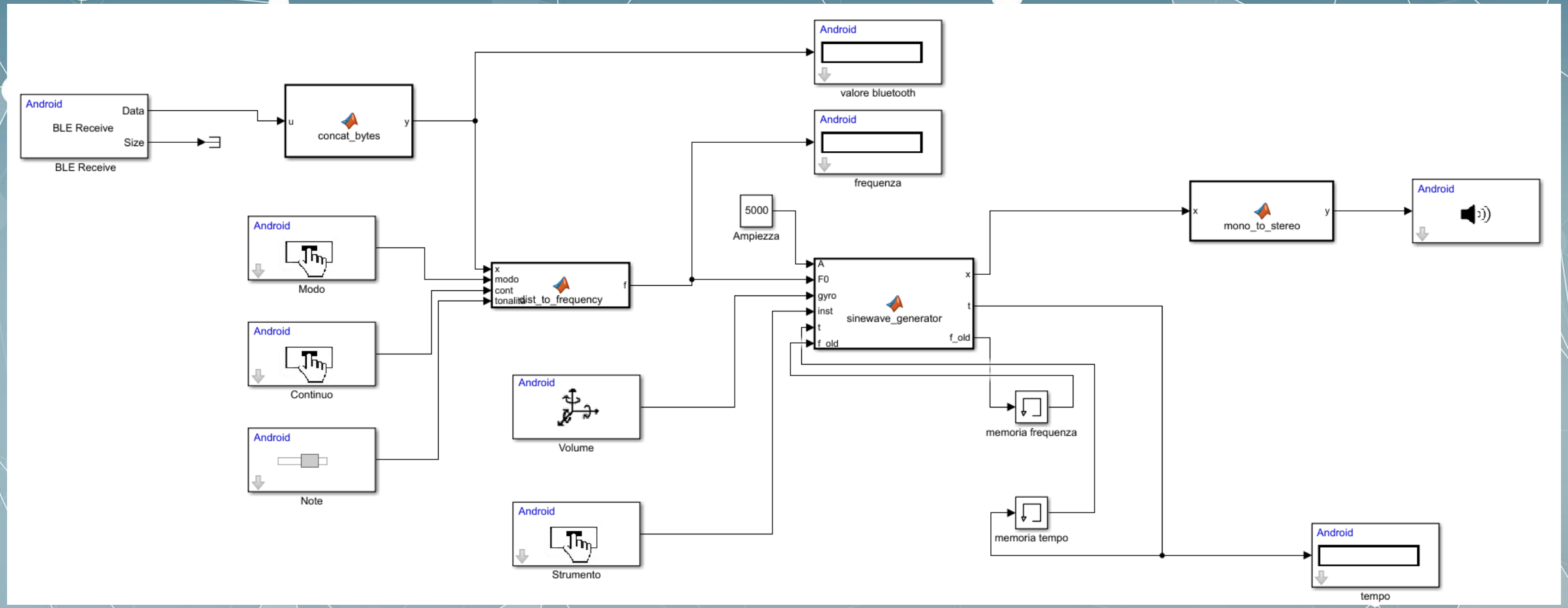

# Distance to Frequency



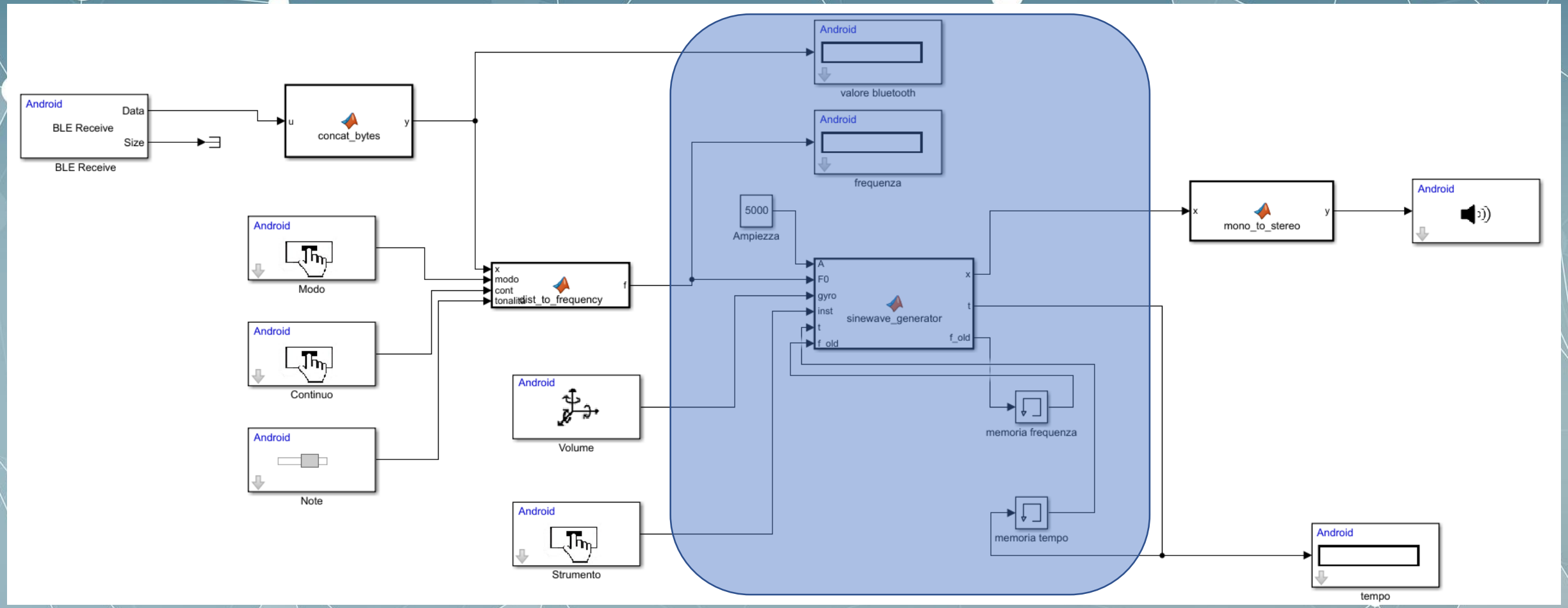

```
function f = dist_to_frequency(x, modo, cont, tonalita)

if cont == 0
    note = 0;
    switch tonalita
        case 1
            note = 261.63;
        case 2
            note = 277.18;
        case 3
            note = 293.66;
        case 4
            note = 311.13;
        case 5
            note = 329.63;
        case 6
            note = 349.23;
        case 7
            note = 369.99;
        case 8
            note = 392;
        case 9
            note = 415.30;
        case 10
            note = 440;
        case 11
            note = 466.16;
        case 12
            note = 493.88;
    end
```

```
    if modo == 0

        if x<=10
            f = double(90);
        elseif x>10 && x<60
            f = double(note/2^(1/12));
        elseif x>=60 && x<100
            f = double(note);
        elseif x>=100 && x<140
            f = double(note*2^(2/12));
        elseif x>=140 && x<180
            f = double(note*2^(4/12));
        elseif x>=180 && x<220
            f = double(note*2^(5/12));
        elseif x>=220 && x<260
            f = double(note*2^(7/12));
        elseif x>=260 && x<300
            f = double(note*2^(9/12));
        elseif x>=300 && x<340
            f = double(note*2^(11/12));
        else
            f = double(note*2);

        end
    else
        if x<=10
            f = double(90);
        elseif x>10 && x<60
            f = double(note/2^(2/12));
        elseif x>=60 && x<100
            f = double(note);
        elseif x>=100 && x<140
            f = double(note*2^(2/12));
        elseif x>=140 && x<180
            f = double(note*2^(3/12));
        elseif x>=180 && x<220
            f = double(note*2^(5/12));
        elseif x>=220 && x<260
            f = double(note*2^(7/12));
        elseif x>=260 && x<300
            f = double(note*2^(8/12));
        elseif x>=300 && x<340
            f = double(note*2^(10/12));
        else
            f = double(note*2);

        end
    end
else
    if x <= 10
        f = double(90);
    else
        f = double(233.08+x);
    end
end
```

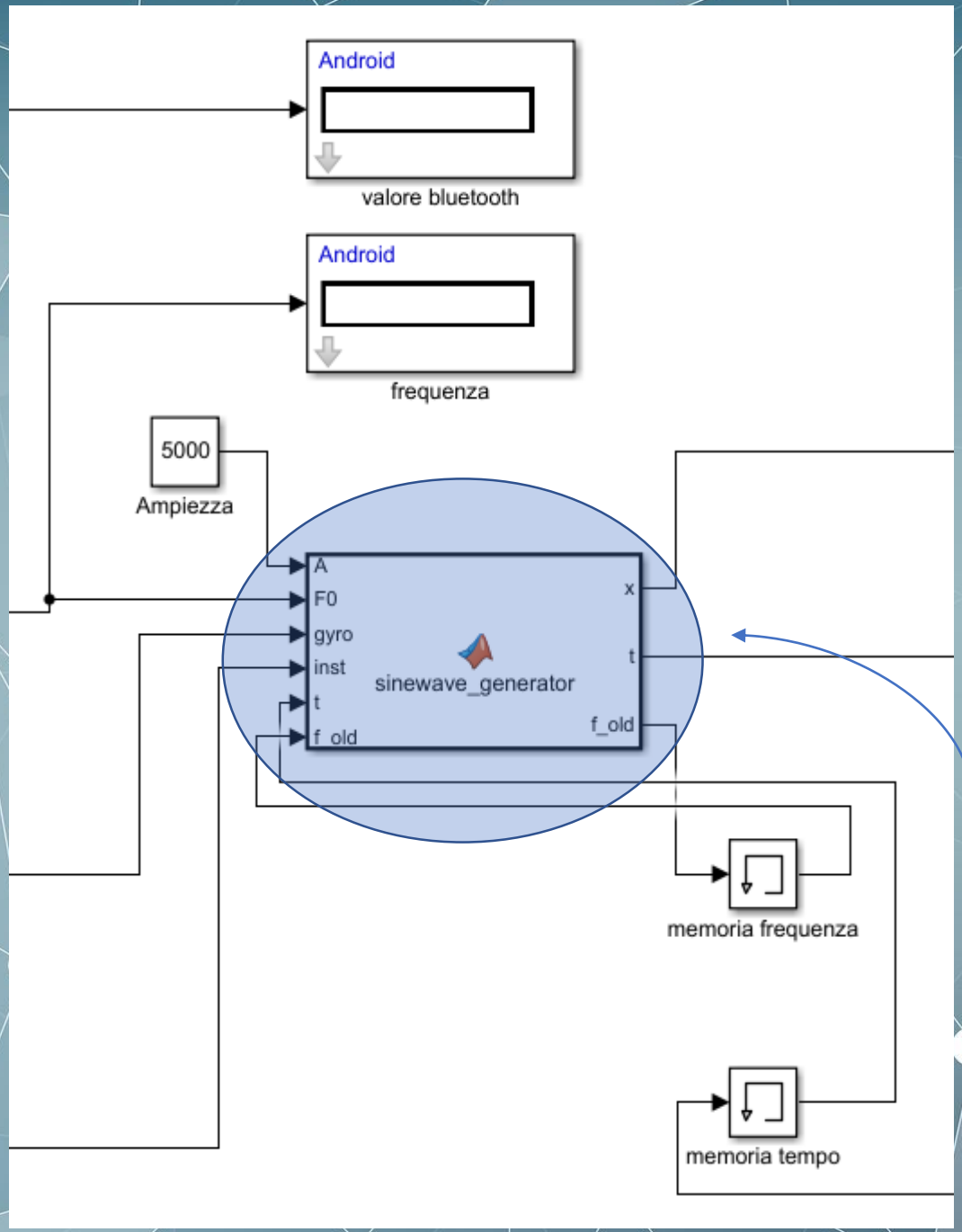
# Codice che implementa la funzione *dist_to_frequency*



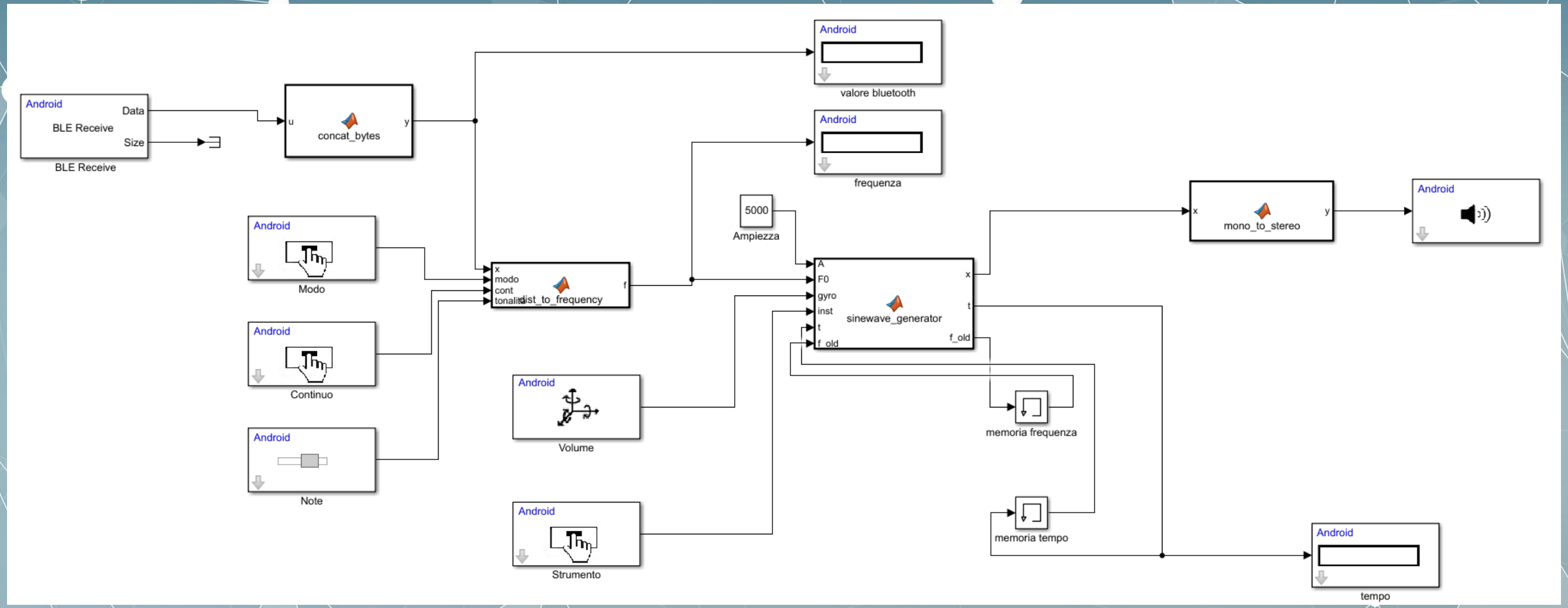

# Schema generale Simulink



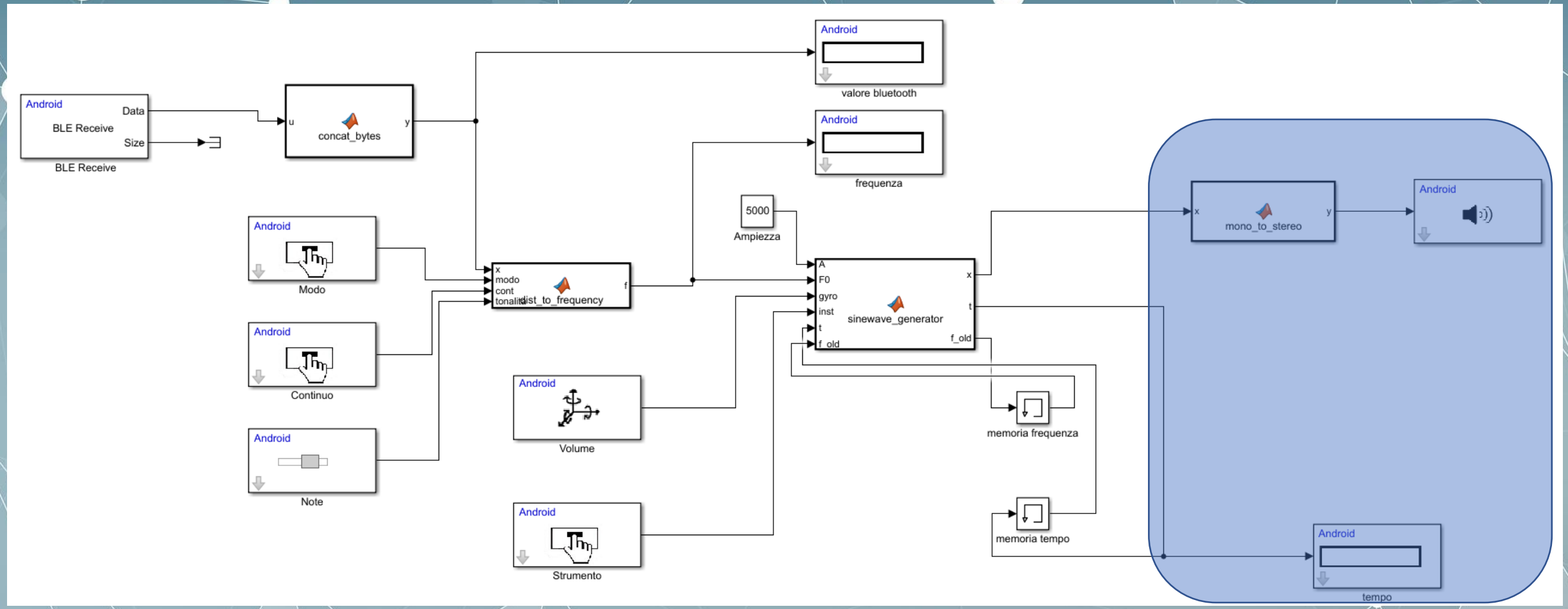

# Sinewave generator



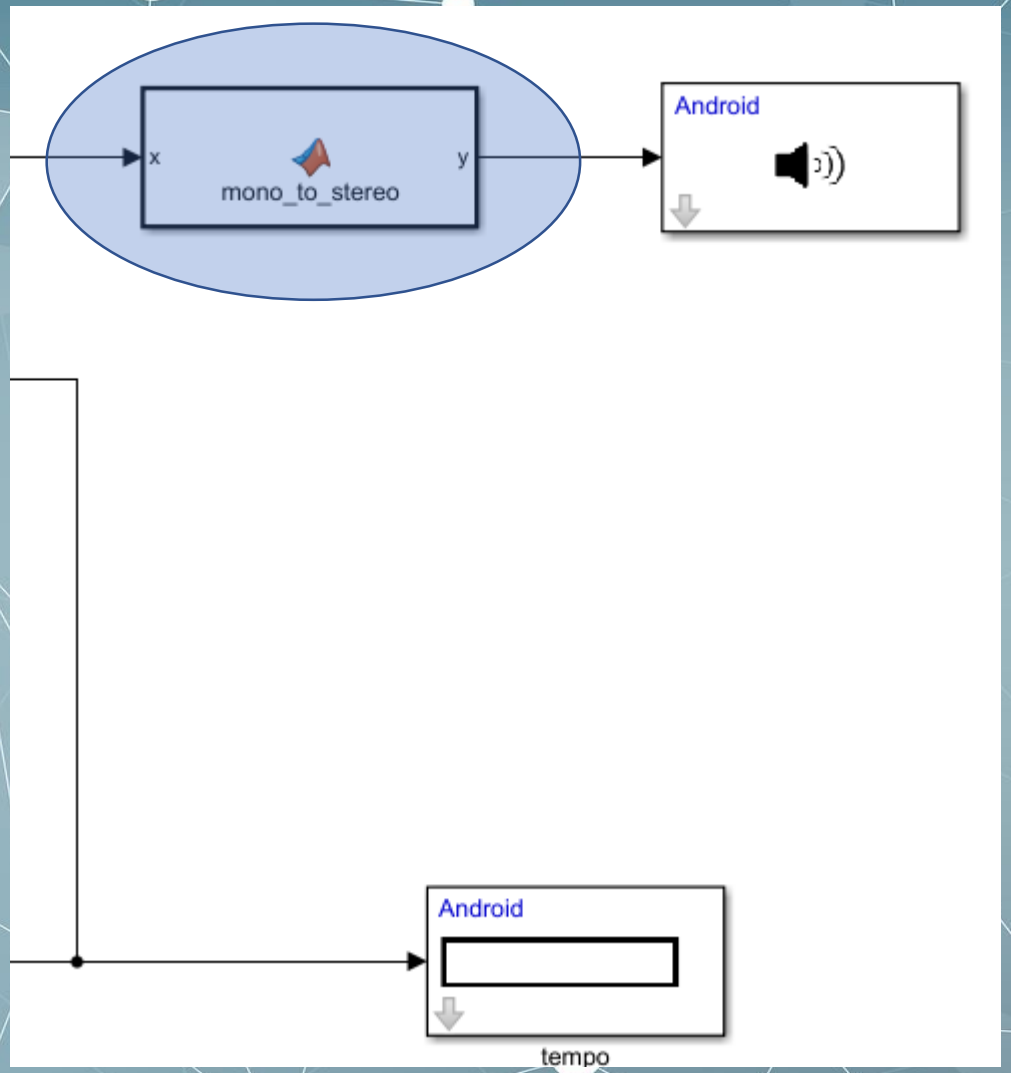

```
function [x, t, f_old] = sinewave_generator(A, F0, gyro, inst, t, f_old)
%{
Funzione che genera un'onda sinusoidale di
- frequenza F0 [Hz]
- ampiezza A [compresa tra 0 e 32737]

Il risultato è un array di N valori generati partendo dal tempo t calcolato
rispetto alla computazione precedente.
Al termine della computazione il tempo è ridotto di n periodi per limitarne
la crescita.
%}

Fs=8000; %[Hz]
Ts=1/Fs; %[s]
N=8;%[S]

x = int16(zeros(N,1));

if F0 > 90

    t = f_old * t / F0;
    f_old = F0;
    F1 = 2*F0;
    F2 = 3*F0;
    F3 = 4*F0;
    A1=A/2;
    A2=A/3;
    A3=A/4;

     if gyro(2) > 1
         volinc = double(gyro(2)/10);
     elseif gyro(2) <-1
         volinc = double(-gyro(2)/10);
     else
         volinc = 1;
     end

    for n=0:N-1
        if inst == 0
            x(n+1)=int16(volinc*(A*sin(2*pi*F0*t*Ts) + A1*sin(2*pi*F1*t*Ts) + A2*sin↙
(2*pi*F2*t*Ts) + A3*sin(2*pi*F3*t*Ts))/4);
        else
            x(n+1)=int16(volinc*A*sin(2*pi*F0*t*Ts));
        end
        t = t + 1;
    end

    T = Fs/F0;
    t = t - T*floor(t/T);

else
    t = f_old * t / F0;
    f_old = F0;

    for n=0:N-1
        x(n+1)=0;
        t = t + 1;
    end

    T = Fs/F0;
    t = t - T*floor(t/T);
end
```




# Schema generale Simulink




# Mono to Stereo




Editor - Block: progetto_final/MATLAB Function3

MATLAB Function3

```
1   function y = mono_to_stereo(x)
2
3 -     y = [x x];
4
```

# Un riepilogo delle funzionalità analizzate nel codice

1. Un bottone per scegliere se utilizzare la thereboard con suono continuo o con scale musicali
2. Un bottone per scegliere di utilizzare una scala maggiore o minore
3. Uno slider per scegliere le tonalità delle scale
4. Un bottone per scegliere se la timbrica di un tono puro oppure di un organetto
5. Un giroscopio per aumentare il volume
6. Riduzione dell'intervallo di callback nel cpp


*Dimostrazione live*

# GRAZIE PER L'ATTENZIONE