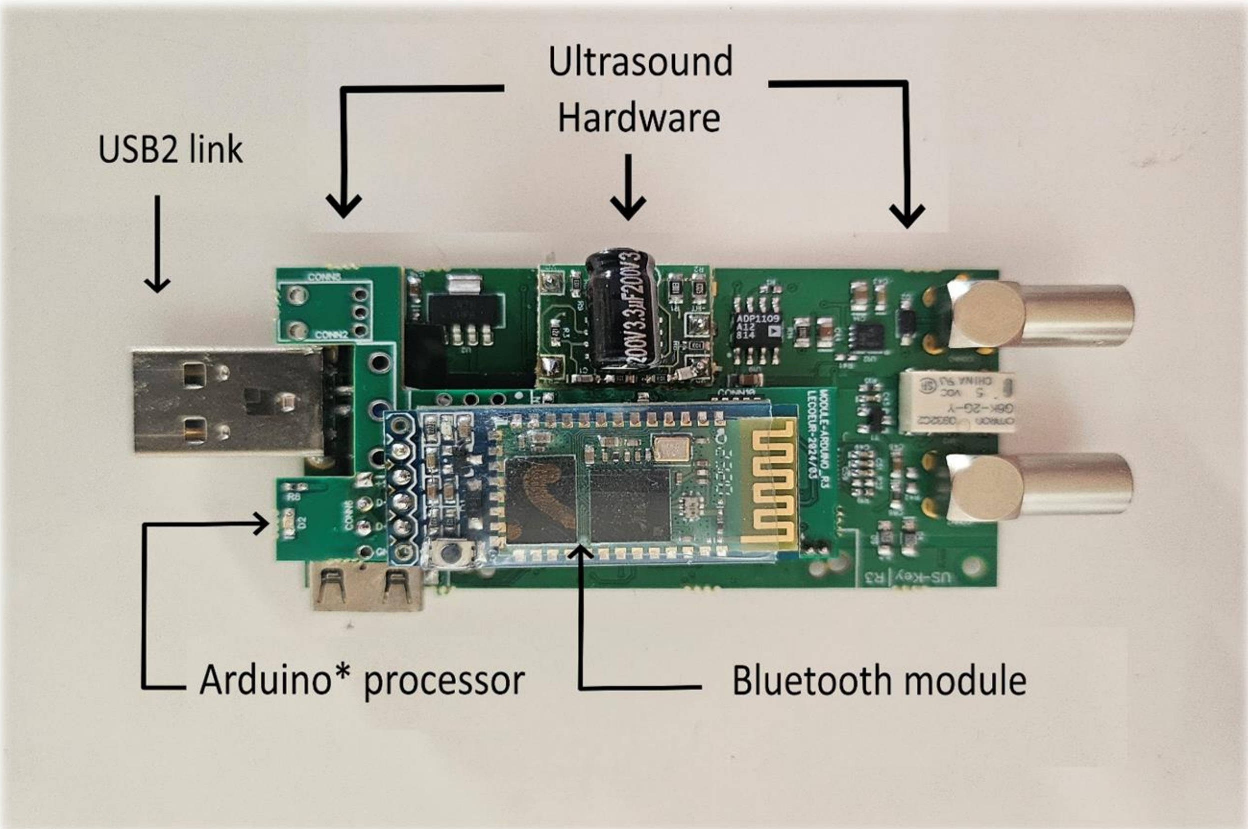
US-BUILDER

Ultrasound development Kit for designers



Create or adapt your own application

Ultrasonic hardware: Transmitter / receiver / digitizer
Control link: USB / Bluetooth
Operating system: Windows / Android / Linux
Programming language: C++/ python / Labview / Arduino

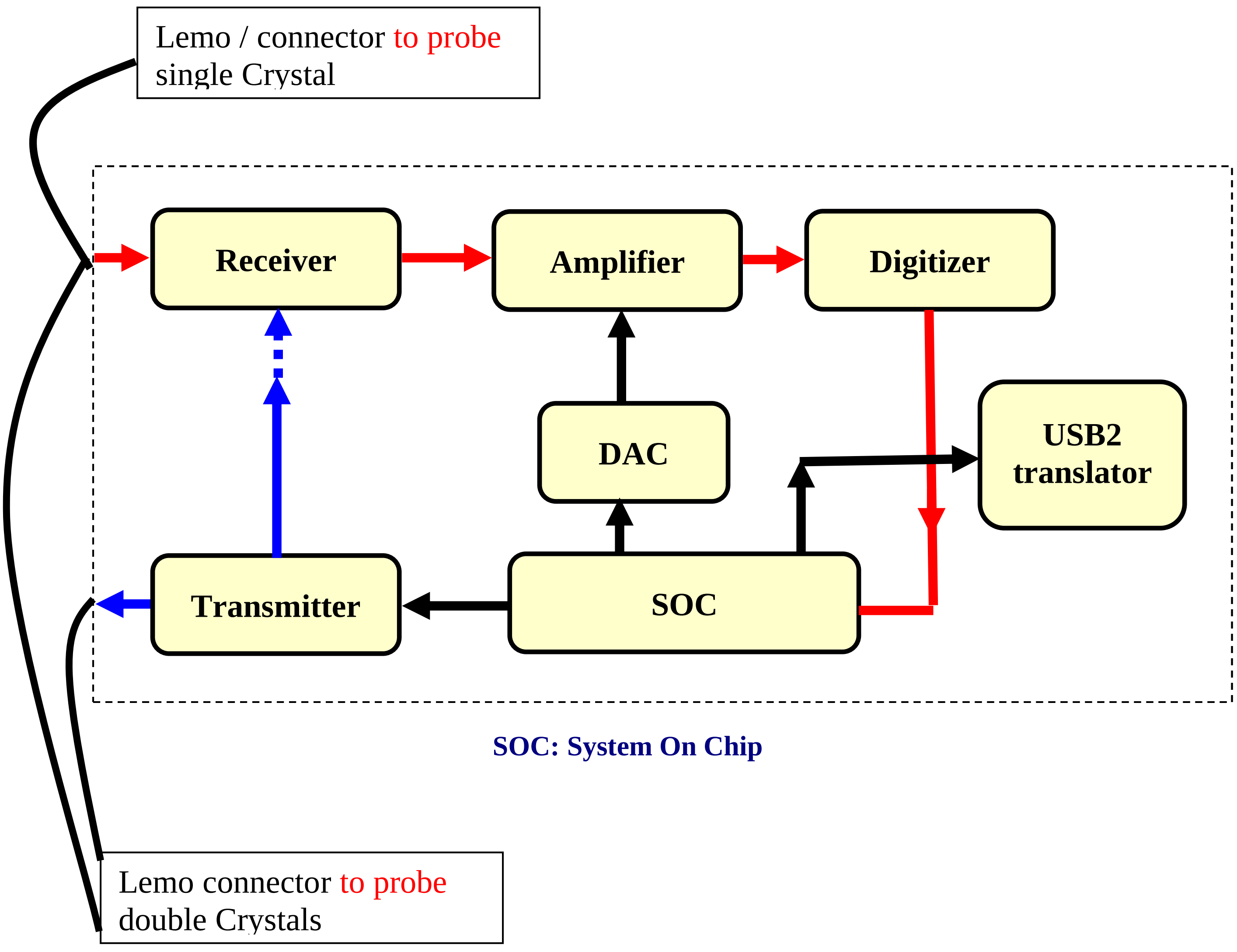
GENERAL DESCRIPTION

US-BUILDER is an OPEN-SOURCE platform dedicated to designers. It is delivered with many software examples that can be modified or rewritten to fit your needs. It is supported by main OS: Windows / Android / Linux. Due to the Arduino processor, that control ultrasound hardware, the user can also write real time application in Arduino language (.ino).

US-BUILDER is the more recent concept and technology in ultrasound control. All electronics are on the board. (Transmitter, Amplifier, Digitizer, Dac, USB2 and Bluetooth link).

The transmitter can generate pulses with a voltage level and a width programmed by the user. A low noise preamplifier combined to a VGA gives a gain range between 0 and 80 dB, a DAC curve is also available. A 12 bits analog digital converter with a sampling frequency of 80 MHz is used to digitize ultrasound signals. The USB2 or Bluetooth link transfers, in real time, ultrasonic waveforms to computer (lap-top, tablet, portable pc)

ULTRASOUND HARDWARE BLOCK DIAGRAM

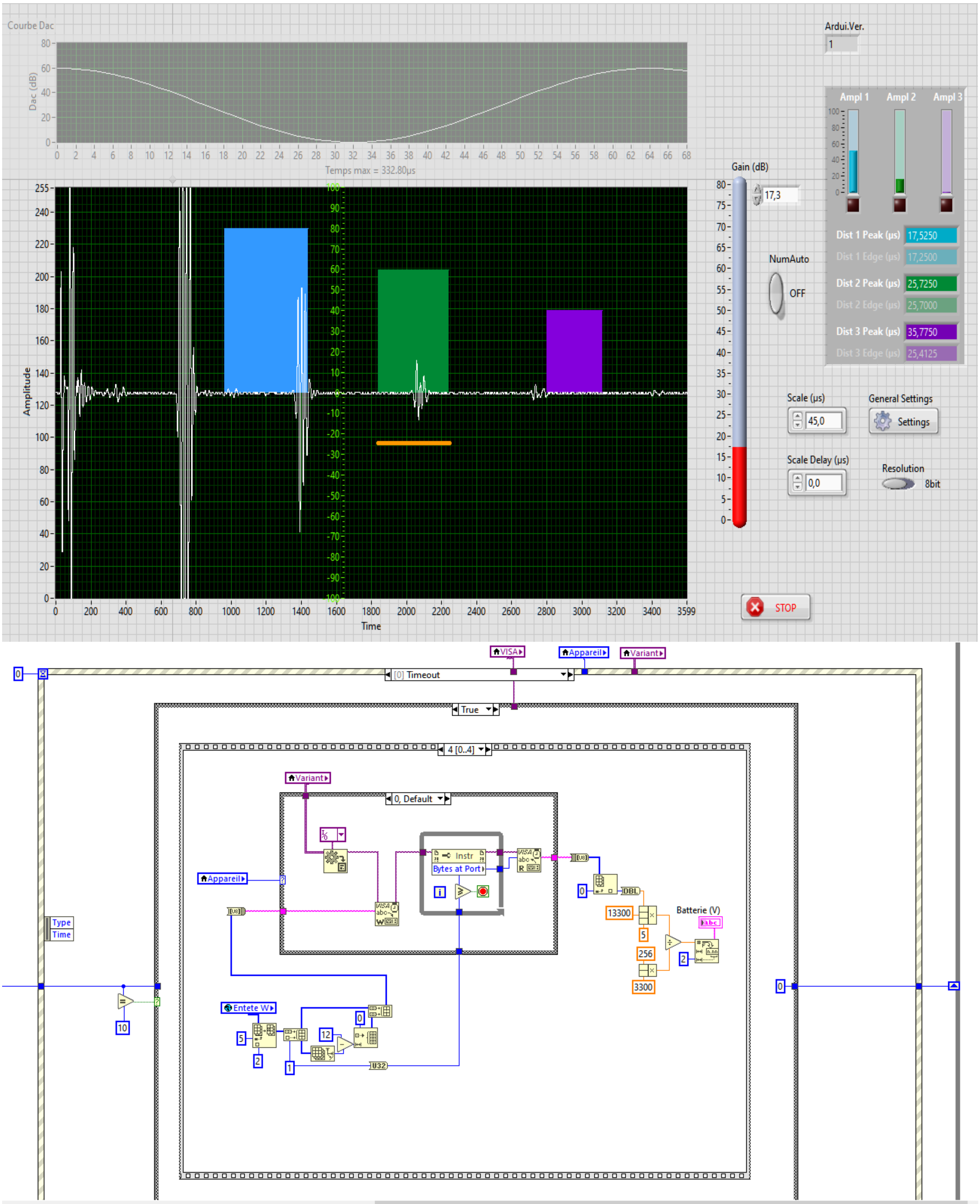


SOC: System On Chip

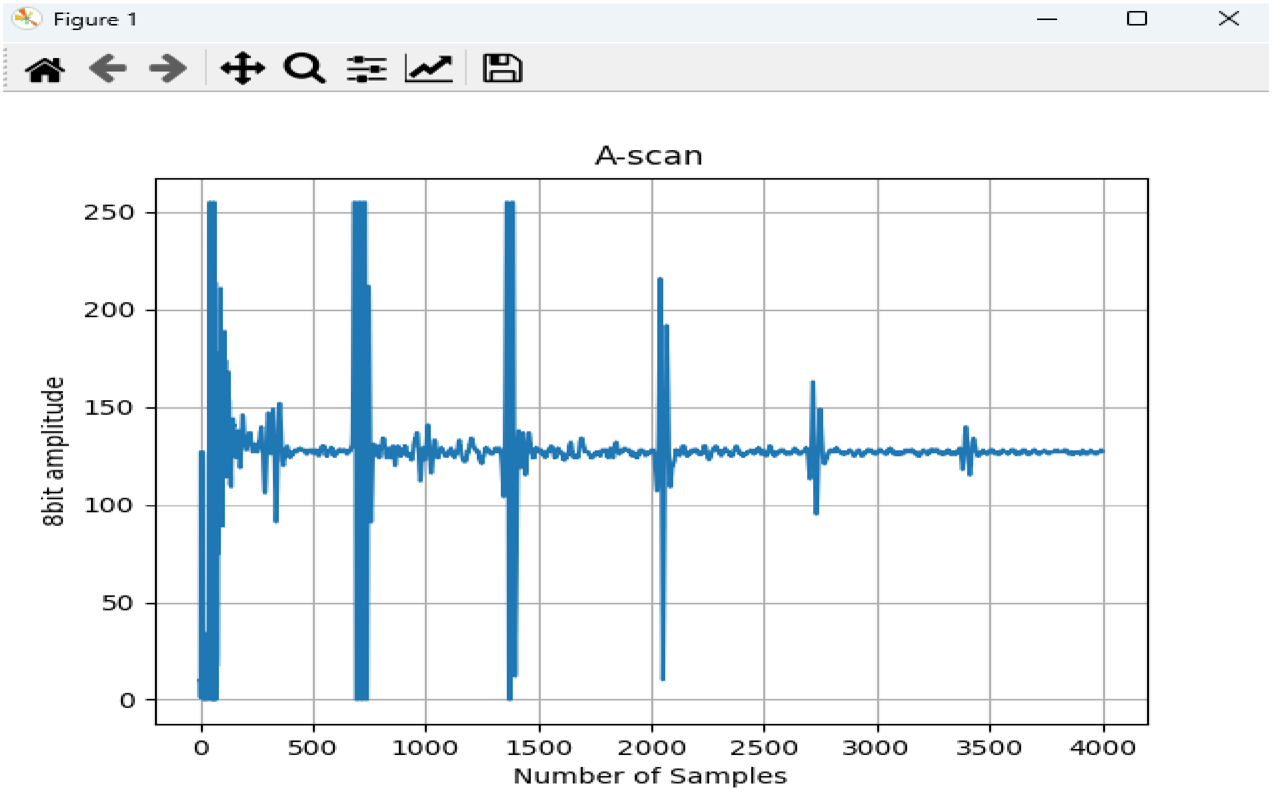
FEATURES

- Dual / single crystal probes from 1 to 25 MHz
- 160/80/40/20 MHz sampling frequency
- USB2 High Speed connection
- Bluetooth
- Arduino processor
- Ultralow noise preamplifier: 0.74 /√Hz
- 6 dB bandwidth: 540 KHz to 30 MHz
- High voltage transmitting pulses up to 250 Volts
- Single /dual crystal probes
- 50 Ω load drive
- Digitizer 12 bits at 80 MSPS
- Programmable gain : 0 to 80 dB

Labview code example (included)



Python code example (included)




```
with Serial(port="COM8", baudrate=115200, timeout=1, writeTimeout=1) as US:
    print ("Serial COM8 Open")
    print()

    Gain_Bin=int(65+(Gain_dB*10))
    GainLSB=ushort(Gain_Bin%256)
    GainMSB=ushort((Gain_Bin-GainLSB)/256)
    SerialFrame[5]=Write
    SerialFrame[6]=ushort(4) # NbByte to write for SPI values
    SerialFrame[7]=ushort(0) # TGC Address
    SerialFrame[8]=GainMSB
    SerialFrame[9]=GainLSB
    SerialFrame[10]=ushort(0)############################## Fct 0 = Gain
    US.write(bytes(SerialFrame[0:12]))
    print(SerialFrame[0:11])
    time.sleep(0.01)

    SerialFrame[5]=Write
    SerialFrame[6]=ushort(2) # NbByte to write for SPI values
    SerialFrame[7]=Compression
    SerialFrame[8]=ushort(3) ############################## Fct 3 = Compression
    US.write(bytes(SerialFrame[0:12]))
    print(SerialFrame[0:9])
    time.sleep(0.01)

    AutoSampLSB=ushort(AutoSampling%256)
    AutoSampMSB=ushort((AutoSampling-AutoSampLSB)/256)
    SerialFrame[5]=Write
    SerialFrame[6]=ushort(3) # NbByte to write for SPI values
    SerialFrame[7]=AutoSampMSB
    SerialFrame[8]=AutoSampLSB
    SerialFrame[9]=ushort(4) ############################## Fct 4 = AutoSampling
    US.write(bytes(SerialFrame[0:12]))
    print(SerialFrame[0:10])
    time.sleep(0.01)
```

Arduino code example (included)

```
#include <SPI.h>
#include "Variables_Functions.h"
#include "Functions.h"
#include "Setup_ArduinoMICRO.h"
#include "Setup_US.h"
#include "SPI_US.h"

byte versionFW=1;


void setup()
  {
    Setup_ArduinoMICRO();
    delay(1000);
    Setup_US();
  }



void loop()
  {
    MANAGE_SPI();
  }

```

```
void SEND_VOLTAGE_bin(byte Voltage_bin)
{
  digitalWrite(E_MOSI,0);
  SPI.transfer(Voltage_bin);
  SPI.transfer(ADDR_Voltage);
  digitalWrite(E_MOSI,1);
}

void SEND_VOLTAGE_Volt(double Voltage_V)
{
  byte Voltage_bin;

  Voltage_bin=(98*Voltage_V/180)+81.777;
  SEND_VOLTAGE_bin(Voltage_bin);
}

void SEND_WIDTH_bin(byte Width_bin)
{
  digitalWrite(E_MOSI,0);
  SPI.transfer(Width_bin);
  SPI.transfer(ADDR_Width);
  digitalWrite(E_MOSI,1);
}
void SEND_WIDTH_ns(double Width_ns)
{
  byte Width_bin;

  Width_bin=(Width_ns-18)/6.25;
  SEND_WIDTH_bin(Width_bin);
}

```