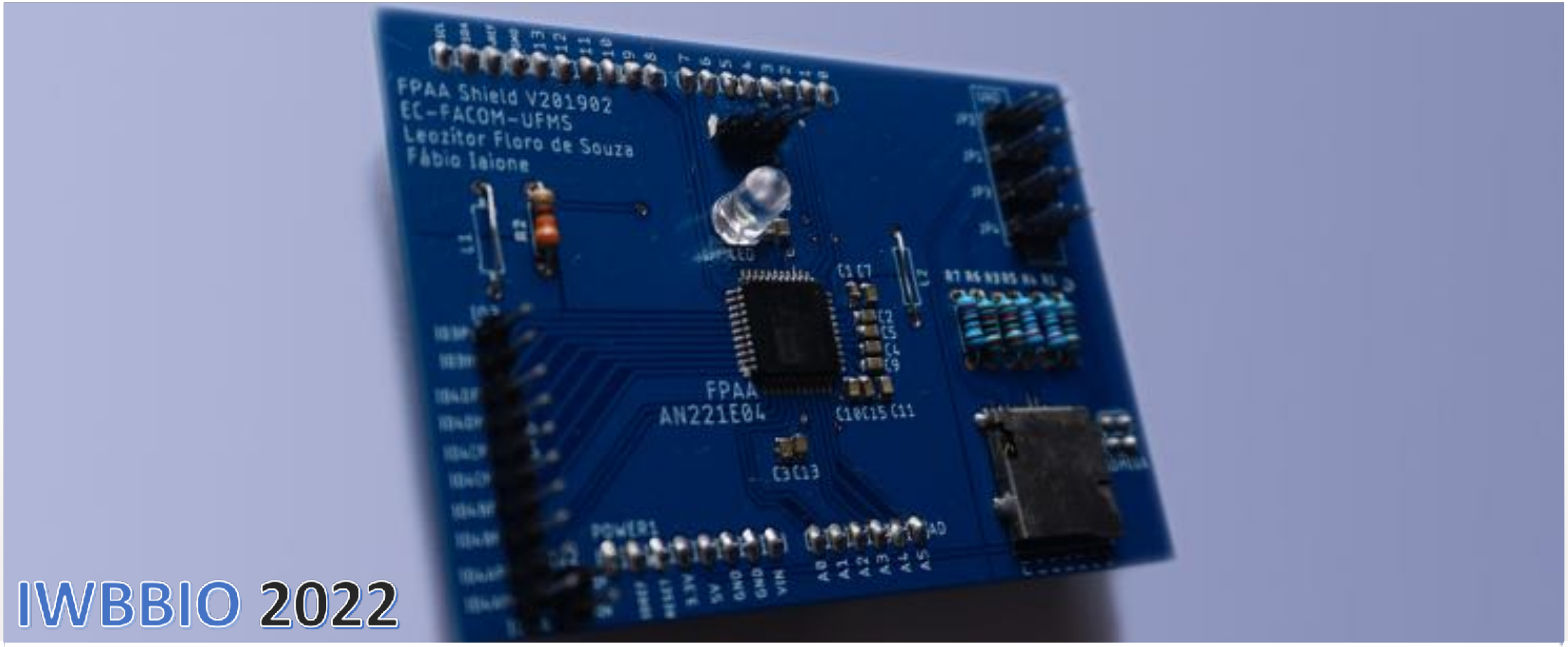


Reconfigurable Arduino Shield for Biosignal Acquisition

Leozitor Floro de Souza - Douglas College
Fabio Iaione
Shih Ting Ju

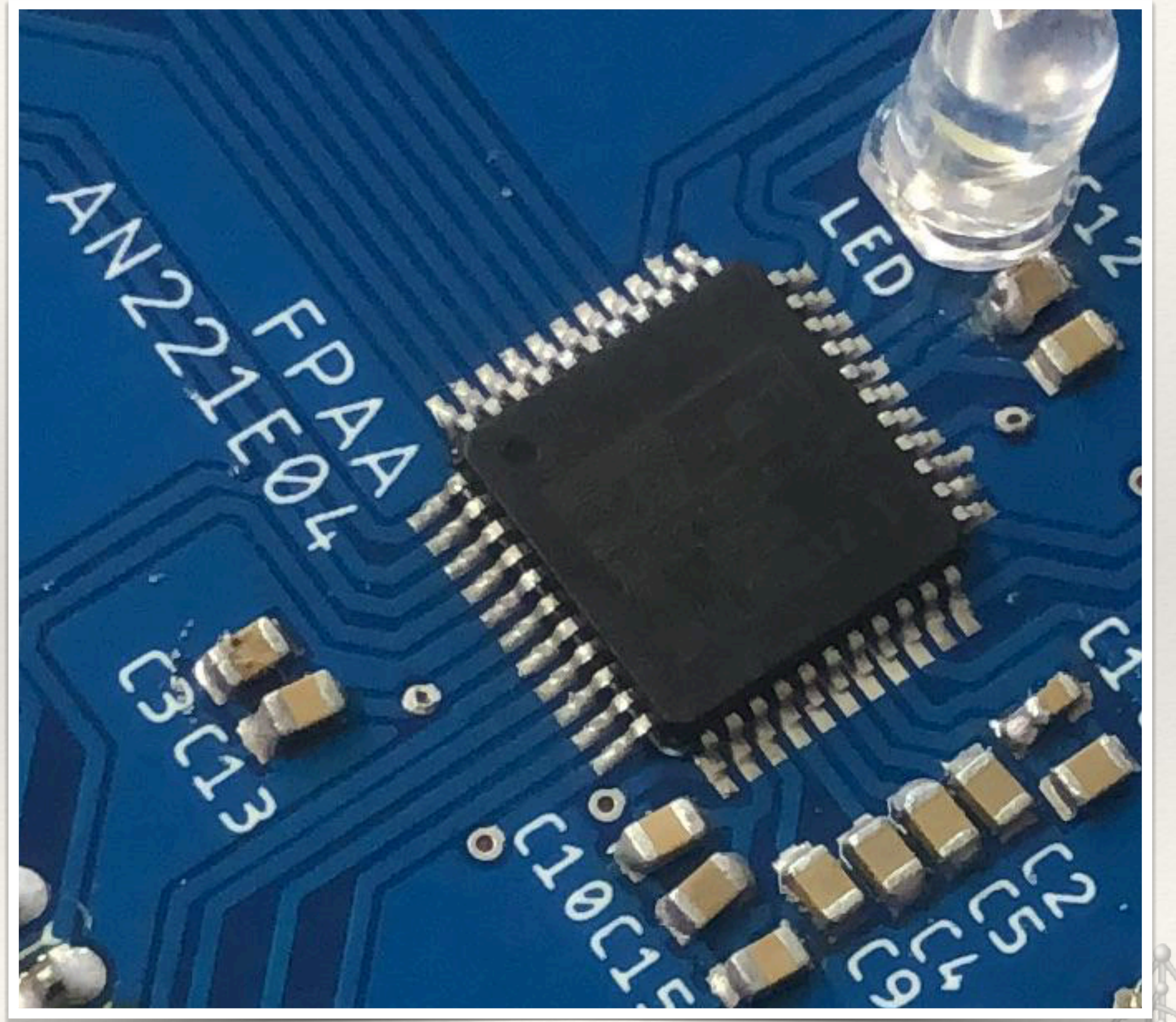
Federal University of Mato Grosso do Sul - UFMS
Faculty of Computing - FACOM

Introduction

- Specific Analog biosignal acquisition devices
  - ECG
  - EEG
  - PCG
  - EMG

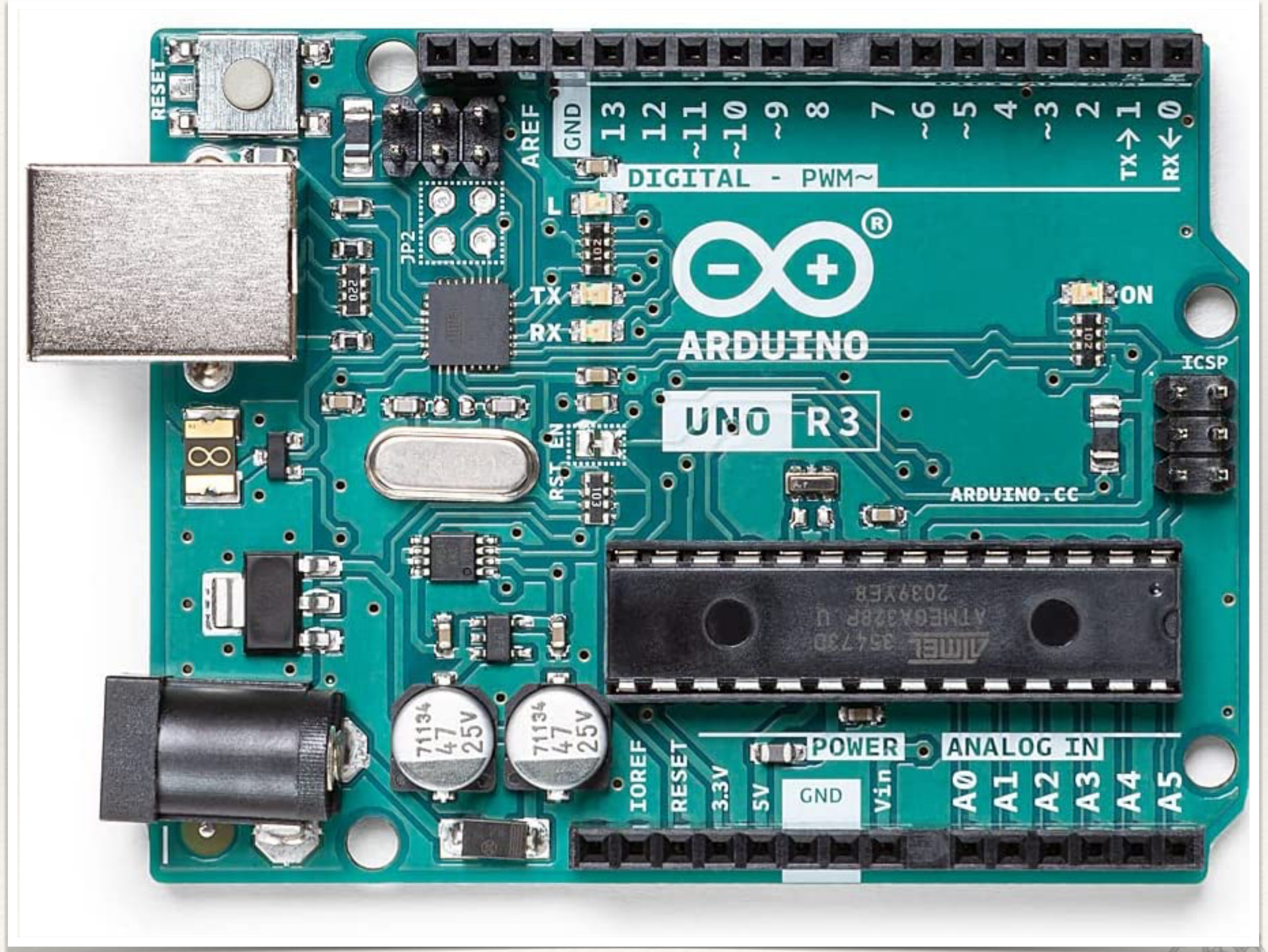
Introduction: FPAA

- Field Programmable Analog Array
- Integrated Circuit Device.
- Offers Field Programmability.
- Creates reconfigurable analog circuits.
- 2x2 Matrix of Configurable Analog Blocks.



Introduction: Arduino

- Open source microcontroller board.
- Based on microchip ATmega 328P.
- Equipped with digital/analog IOs, PWMs, SPI, I2C.
- Programmable via Arduino IDE.

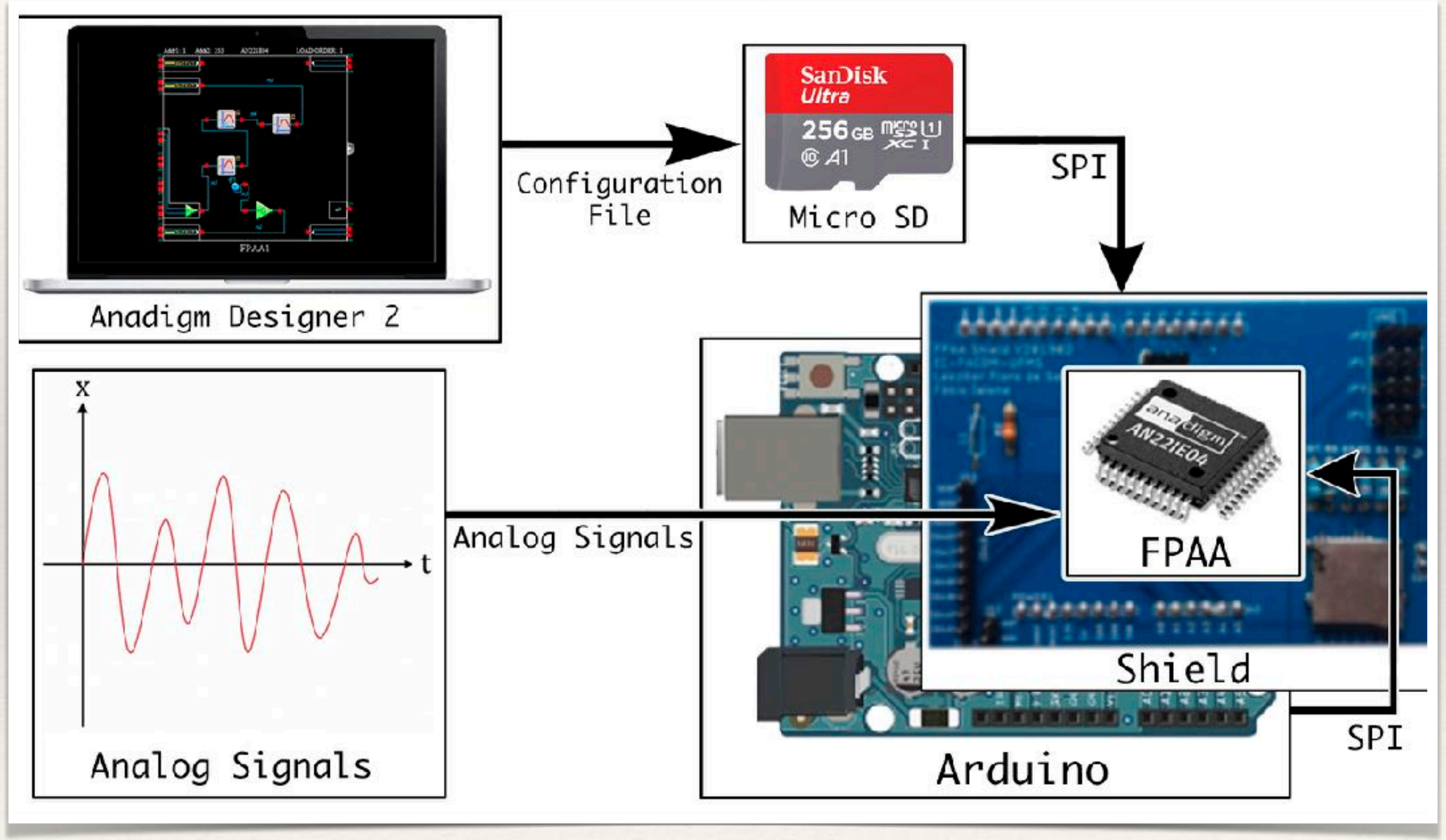


Introduction - Motivation

- Why not FPPA and Arduino.
- A search to following keywords was done:
  - Arduino
  - Biosigna
  - Bioelectrical
  - ECG or Electrocardiogram
  - EEG or Electroencephalogram
  - EMG or Electromyogram
  - EOG or Electrooculogram
  - ERG or Electroretinogram
  - EGG or Electrogastrogram
- Considering we did not find any similar scientific study, this work aims to create an electronic circuit (Arduino shield) and a software library to use and FPPA embedded in a Arduino Board

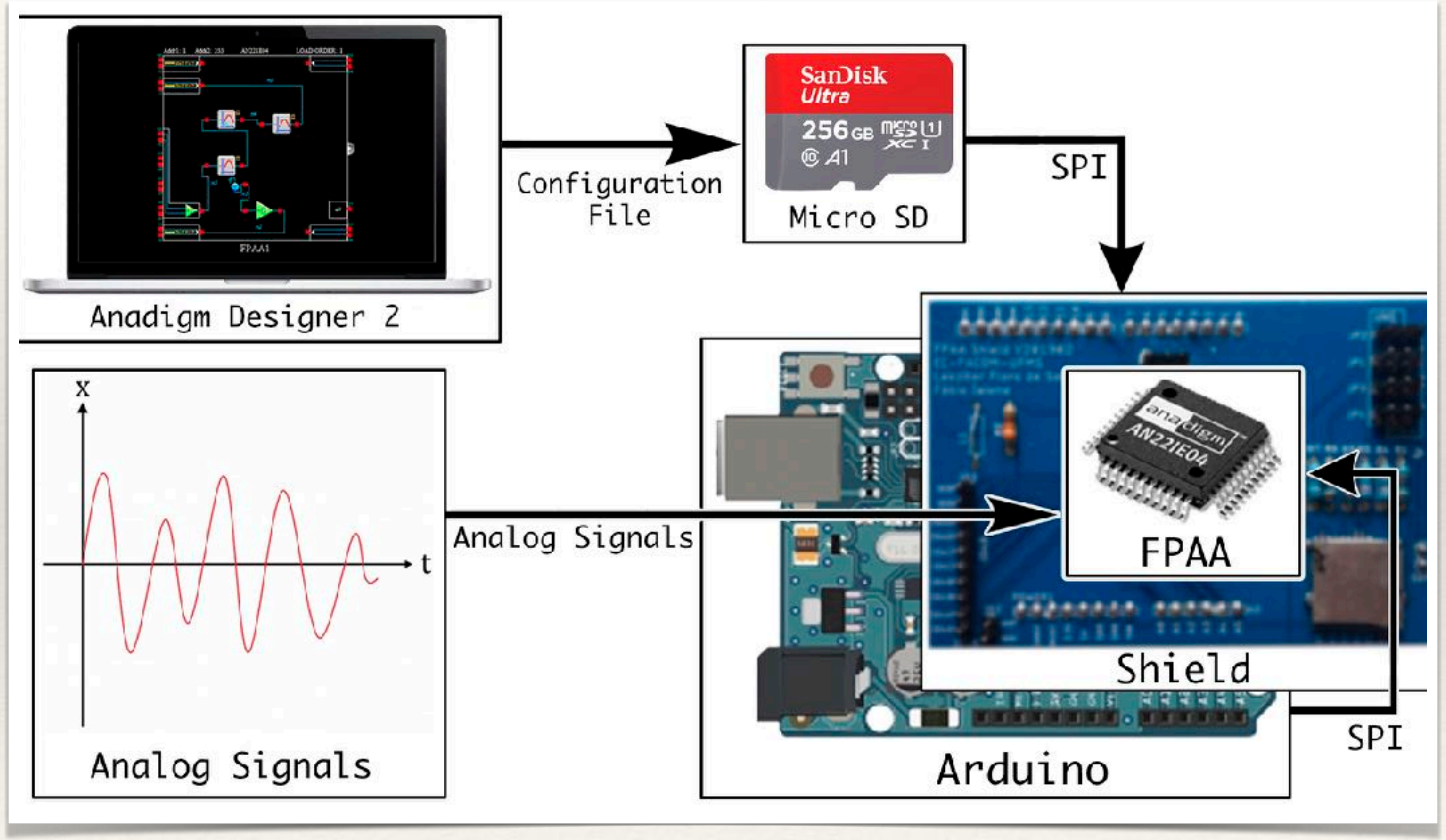
Materials and Methods: How the System Works

- Analog circuit built on Anadigm Designer 2.
- Configuration transferred to micro SD.
- Arduino Initializes FPAA, read configuration and sends to FPAA.
- The analog input passes through FPAA internal analog circuits and is read by Arduino (analog inputs).



Materials and Methods: How the System Works

- FPAA requires two clock signals:
  - DCLK for digital communication.
  - ACLK for analog circuits. (switched capacitor filters).
  - Arduino generated the DCLK by software (SPI bit-banging).
- Due to FPPA ACLK internal clock dividers limitation.
  - It is necessary configurable ACLK signal.
  - Is is necessary to generate the ACLK through Arduino as well.



Materials and Methods: Arduino Library

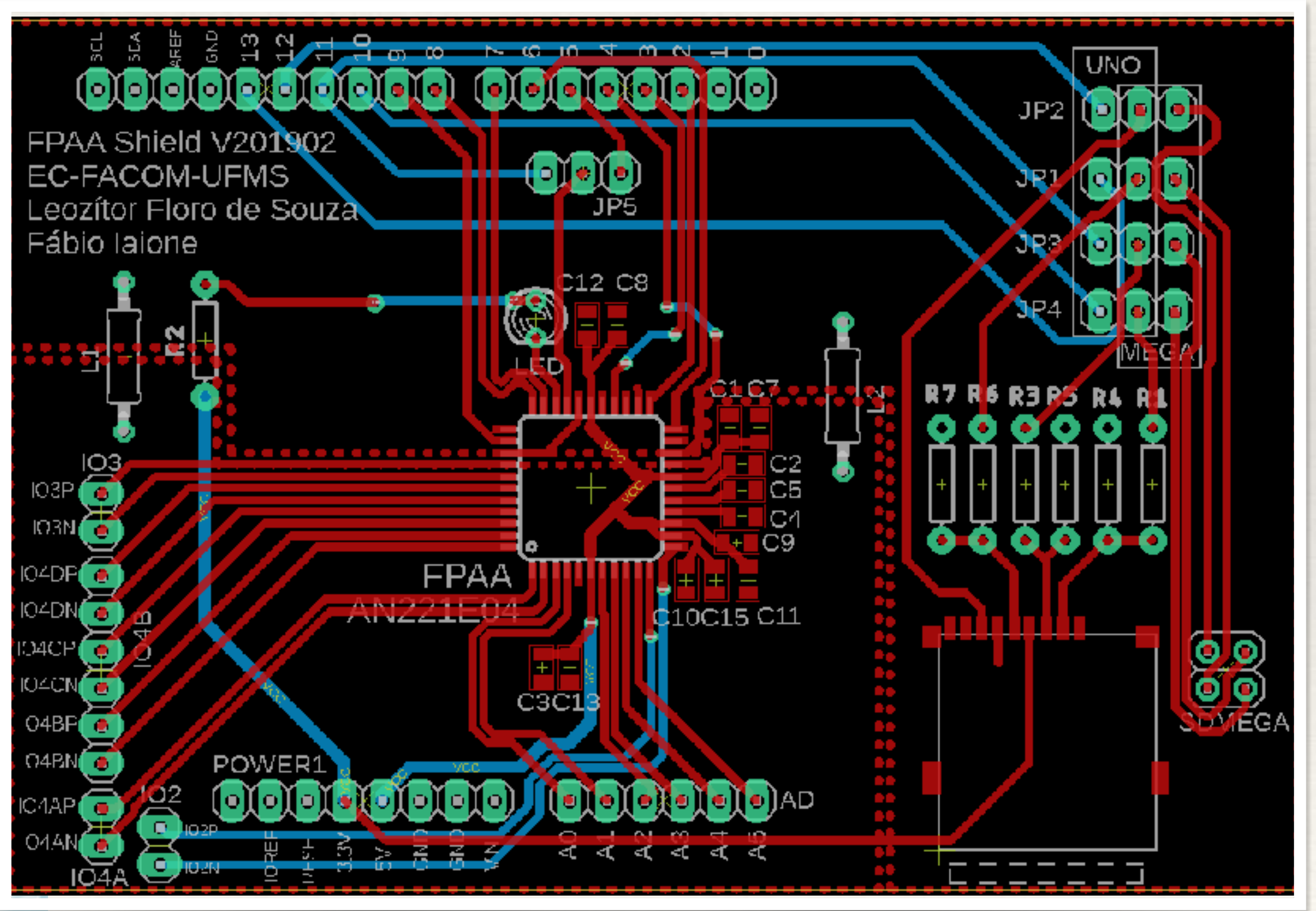
- Arduino built in **tone(pin, frequency)** function generated a maximum frequency of 65 kHz.
- A function **setClock(frequency)** was developed, manipulating registration of ATMega time allowing clock generation from 1.95 kHz up to 8 MHz

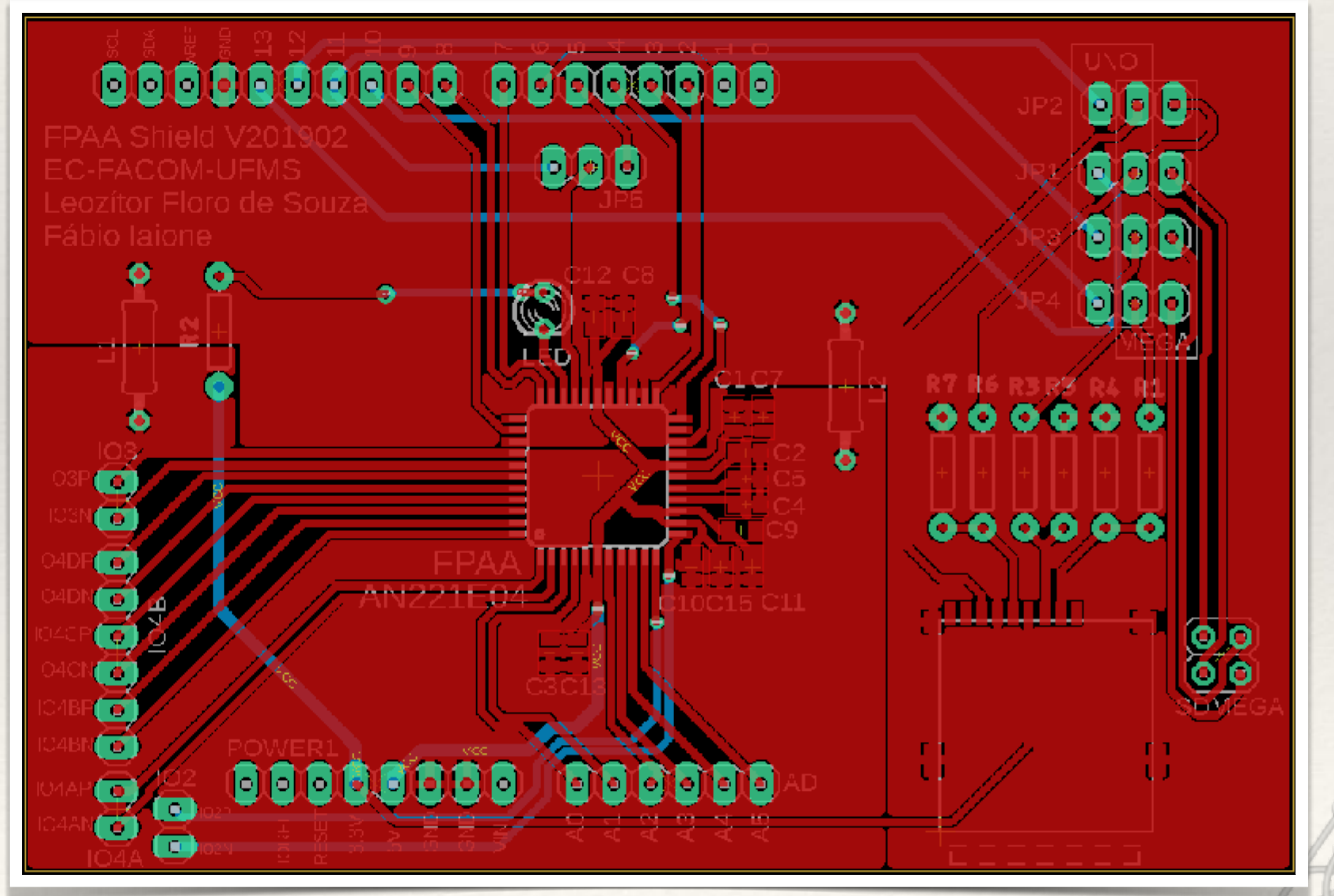
Materials and Methods: Arduino Library

- **FPPAA.begin(".ahf file")**
  - Initializes SPI Bus, SD card and FPAA.
- **FPAA.setClock (freq)**
  - Sets the frequency of the FPAA ACKL generated by Arduino allowing values of: F8M (8 MHz), F4M, F2 666M, F2M, F1 6M, F1 333M, F1M, F500K (500 kHz), F250K, F125K, F62 5K, F31 250K, F15 625K, F7 8125K, F3 90625K and F1 953125K;
- **FPAA.update (".ahf file")**
  - Sends reconfiguration data to the FPAA without the need for restart (on-the-fly).
- **FPAA.read (pinA, pinB, VREF)**
  - Returns the difference between voltages applied to pins A and B, considering the reference voltage of the Arduino ADC in Volts (VREF). This function returns the difference of the voltages because the FPAA has differential outputs.

Materials and Methods: PCB Design

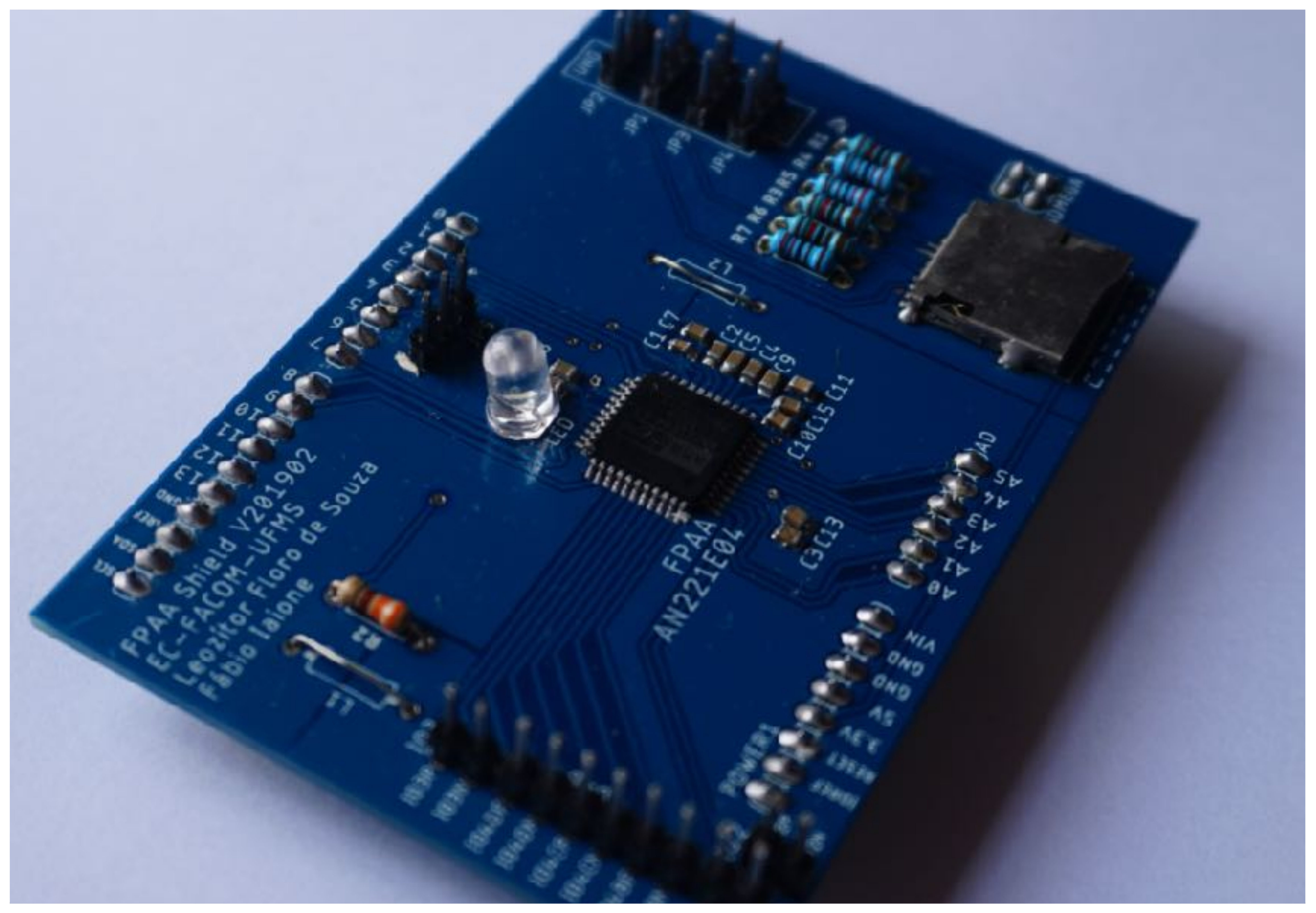
- PCB Designed on Auto Desk Eagle, following good practices of PCB design and recommendations of the FPAA manufacturer.

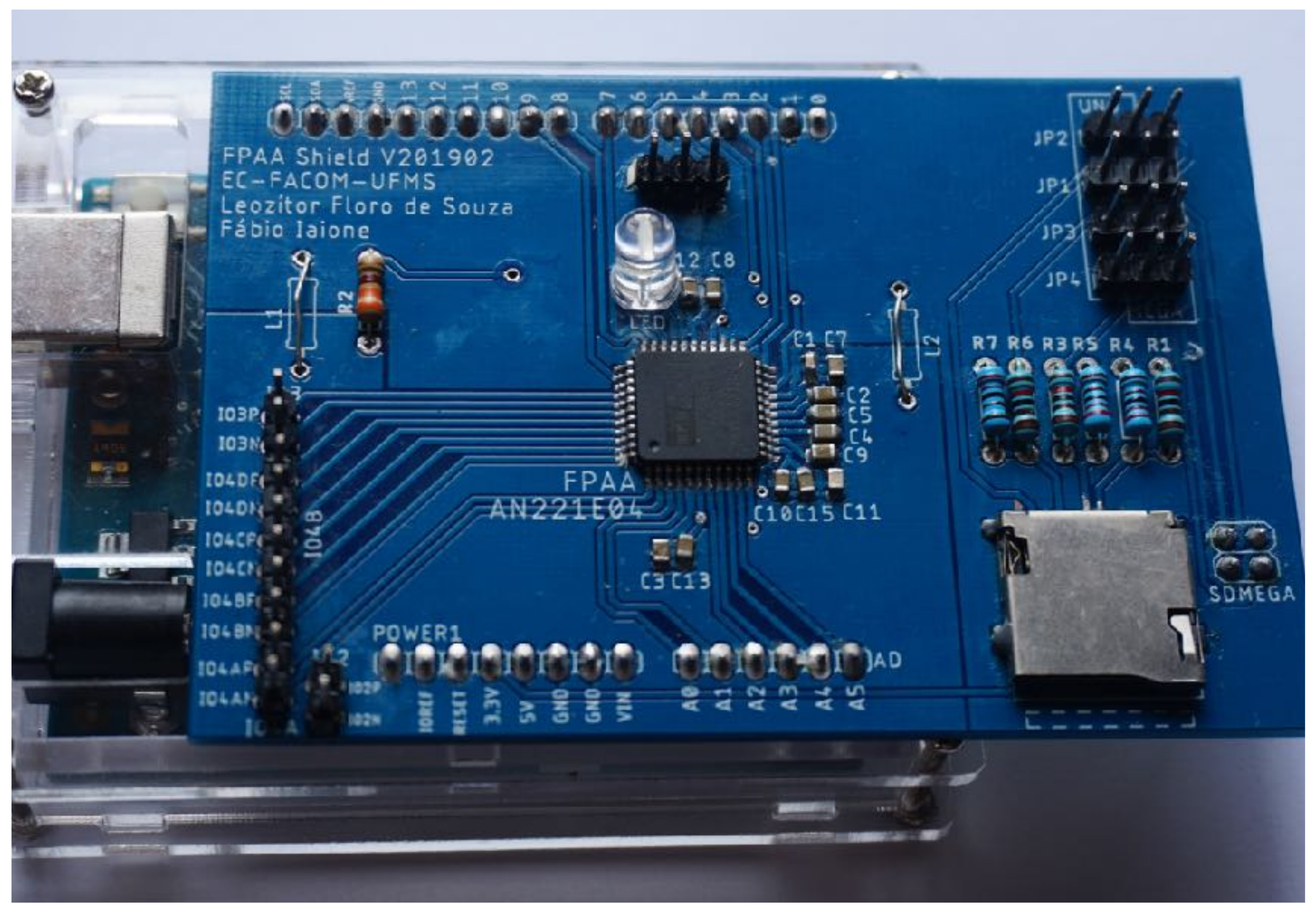




Results: Arduino Shield

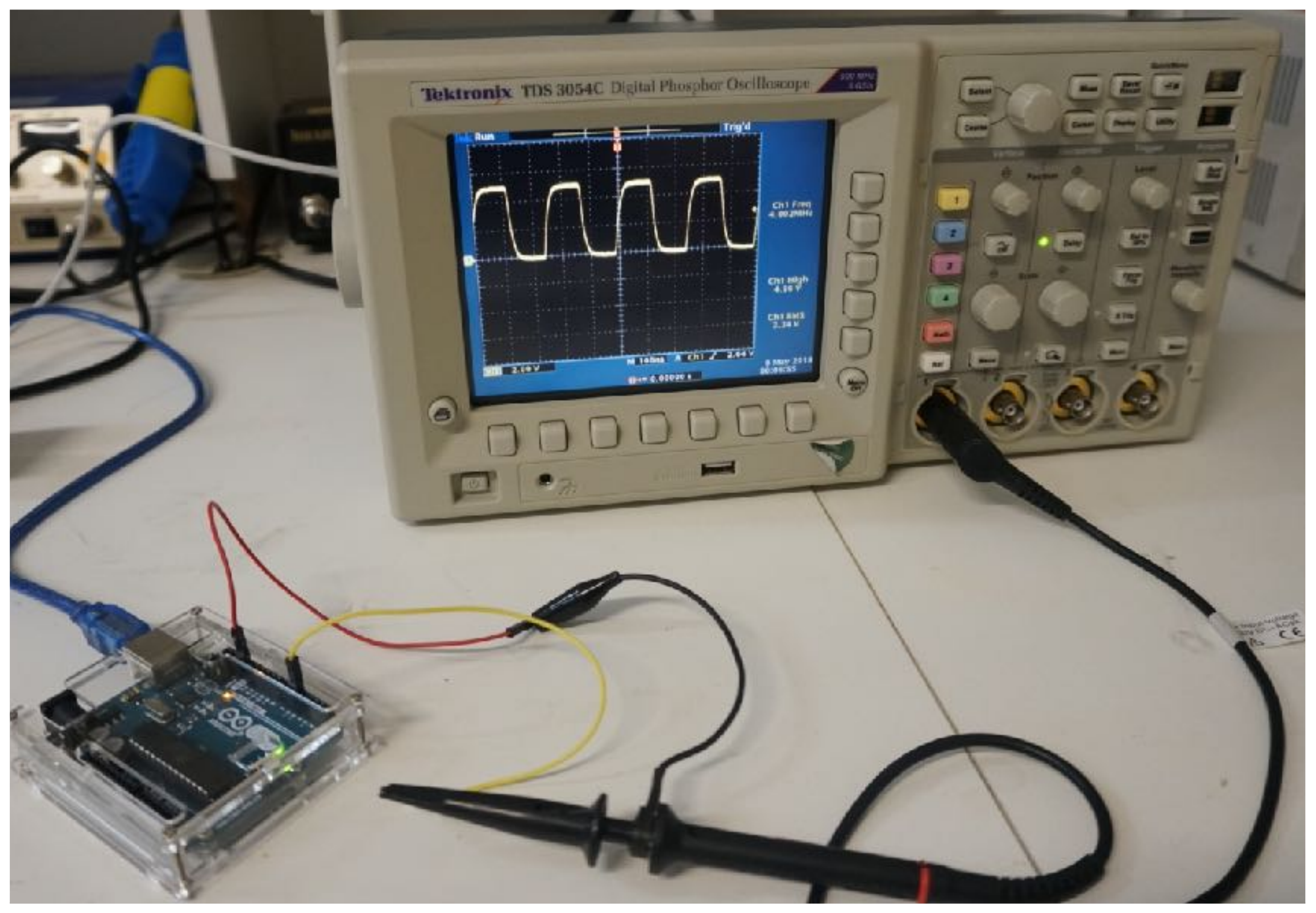
- Arduino Shield ready and attached to an Arduino Uno Board.

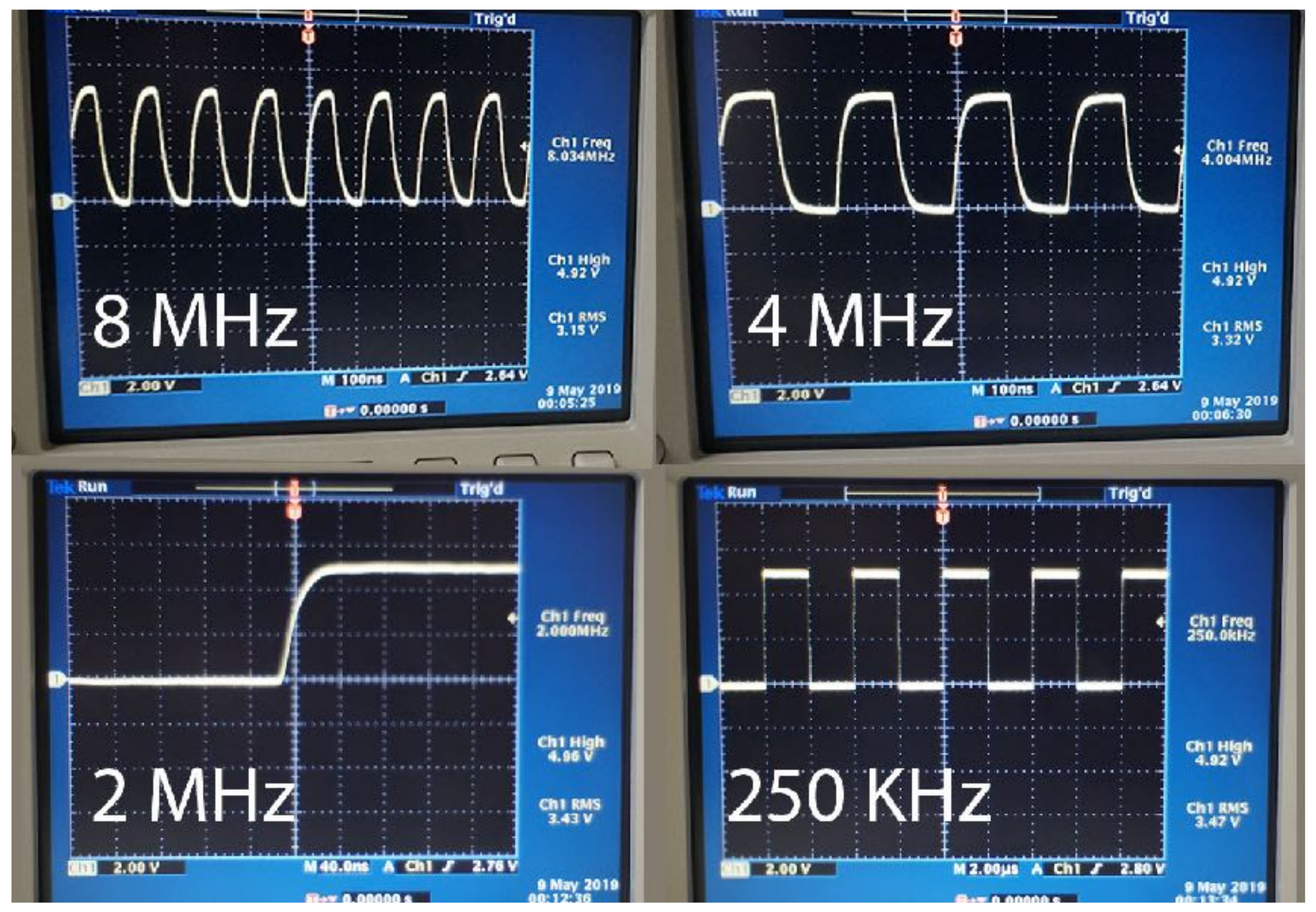




Results: Clock Test

- Digital Oscilloscope used to assure the correctness of Arduino Clock Generation, that is necessary for FPAA (ACLK).





Results: Clock Test

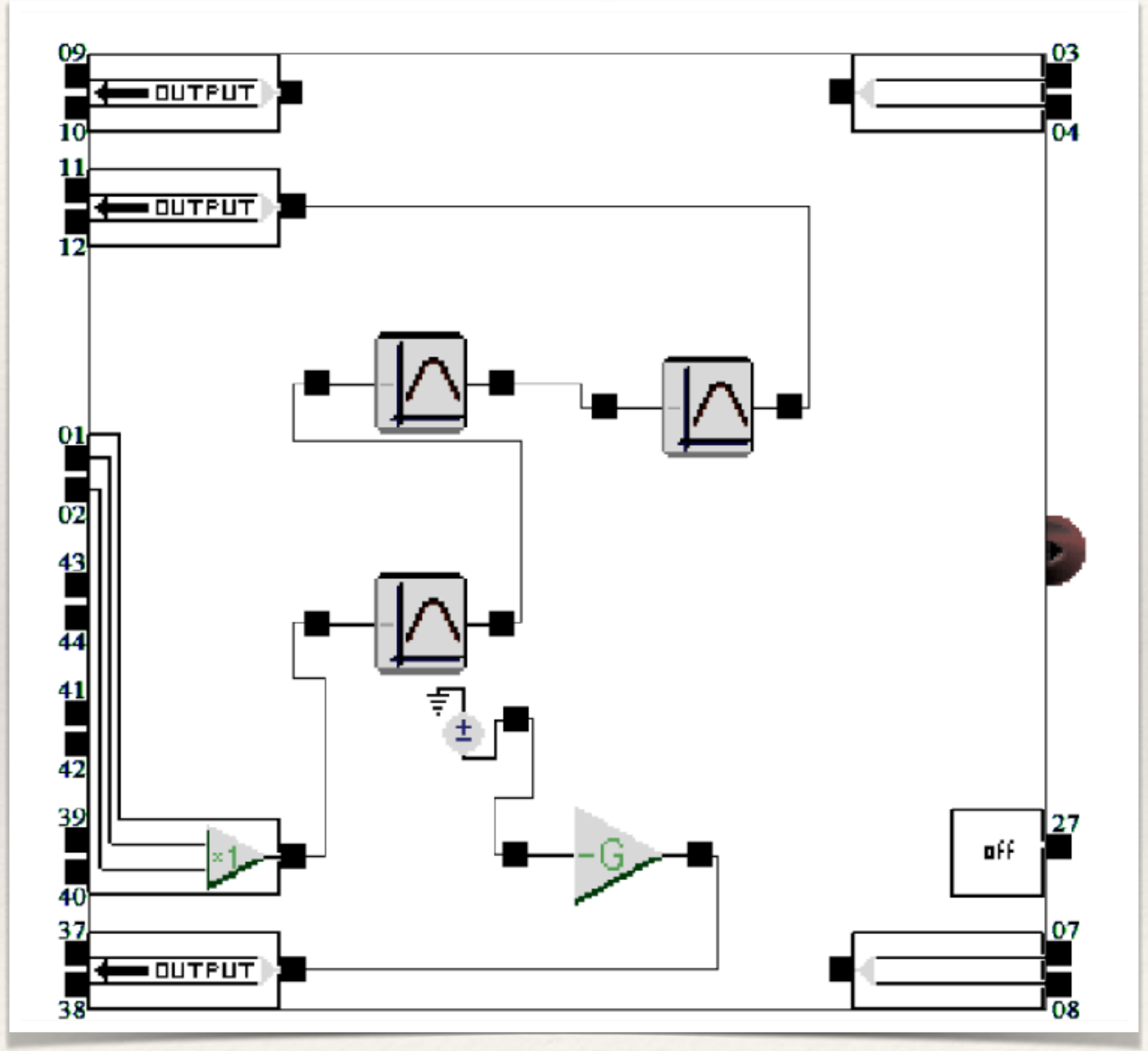
- Comparison between theoretical clock and real clock measure.
- Three measurements (M1, M2 and M3).
- Average Maximum Percentage Error of -0.45 %
- Standard deviation of 0.16%

F(Hz)	M1(Hz)	M2(Hz)	M3(Hz)	ME(Hz)	ME(%)
1953.125	1938	1942	1942	-15.1	-0.8
3906.25	3876	3873	3870	-36.3	-0.9
7812.5	7785	7777	7776	-36.5	-0.5
15625	15570	15570	15580	-55	-0.4
31250	31130	31130	31130	-120	-0.4
62500	62250	62260	62260	-250	-0.4
125000	124500	124500	124500	-500	-0.4
250000	249100	249100	249100	-900	-0.4
500000	498100	498100	498100	-1900	-0.4
1000000	996200	996200	996100	-3900	-0.4
1333000	1328000	1328000	1328000	-5000	-0.4
1600000	1594000	1594000	1594000	-6000	-0.4
2000000	1993000	1992000	1992000	-8000	-0.4
2666000	2657000	2657000	2658000	-9000	-0.3
4000000	3980000	3984000	3986000	-20000	-0.5
8000000	7974000	7968000	7968000	-32000	-0.4

Results: Filter Implementation

- Arbitrary band-pass filter implemented AnadigmFilter tool available in the AnadigmDesigner2.

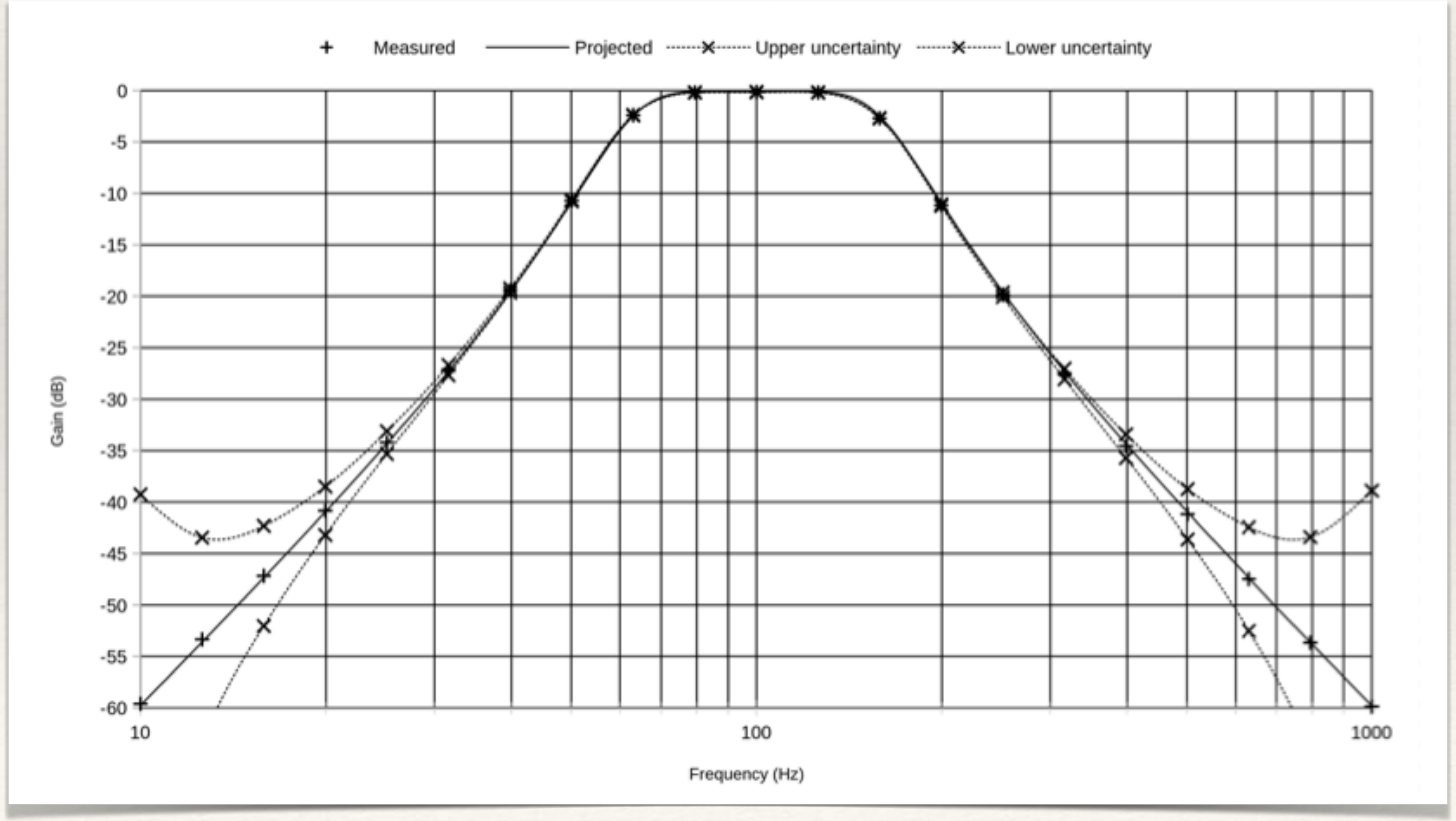
- Input Characteristics:
    - Passband ripple: 3 dB
    - Passband gain: 0 dB
    - Stop band attenuation: -30 dB
    - Center frequency: 100 Hz
    - Stop bandwidth: 500Hz



Block diagram of the circuit generated to implement the desired band-pass filter.

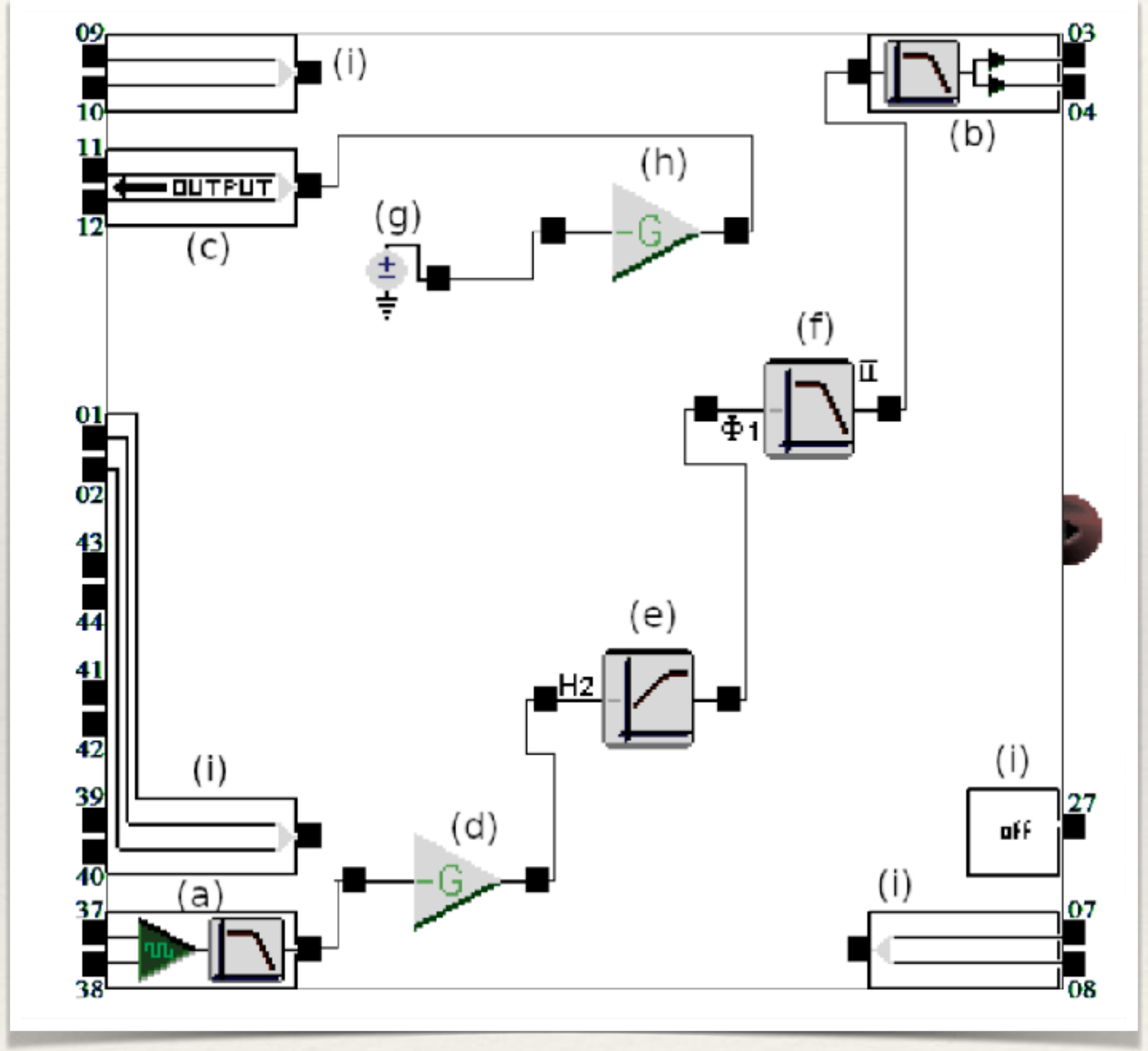
Results: Filter Implementation

- The graph shows the projected and measured frequency responses of the pass-band filter.
- Upper and lower limits considering the measurement uncertainty of used device.
- The average error was 0.027 dB (SD = 0.163 dB) and the maximum error was 0.265 dB.



Results: ECG Acquisition Circuit Test

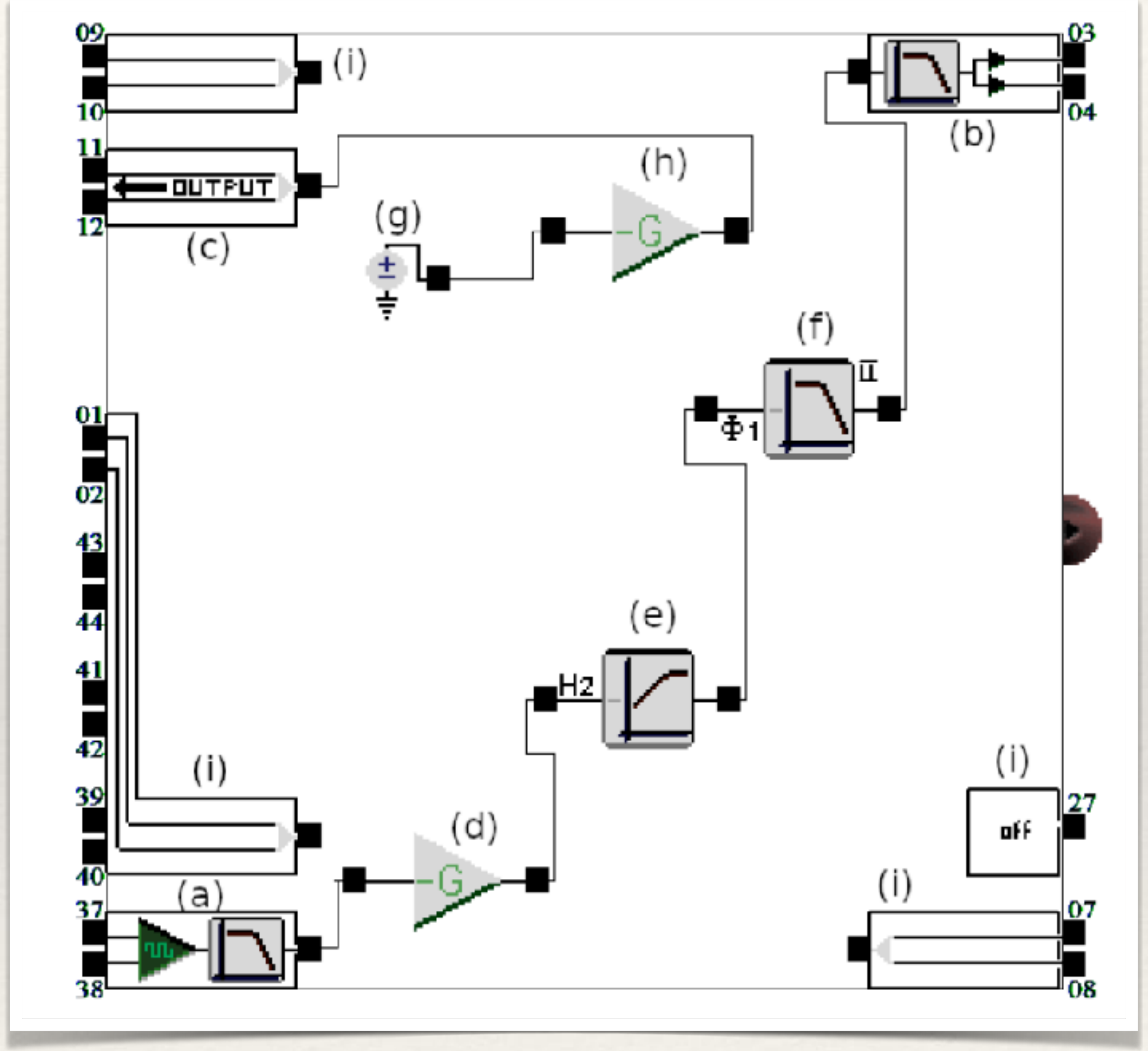
- a) Inputs in differential mode connected to the two ECG electrodes and a Low pass filter to eliminate radio interference (gain = 128).
- b) Low-pass filter.
- c) Output in connected to third ECG electrode working as a reference.
- d) Inverter amplifier to increase the gain of input signal (gain=4).



Circuit's block diagram for ECG signal acquisition. The blocks marked with (i) appear by default and are not used. The ACLK utilized was 125 kHz

Results: ECG Acquisition Circuit Test

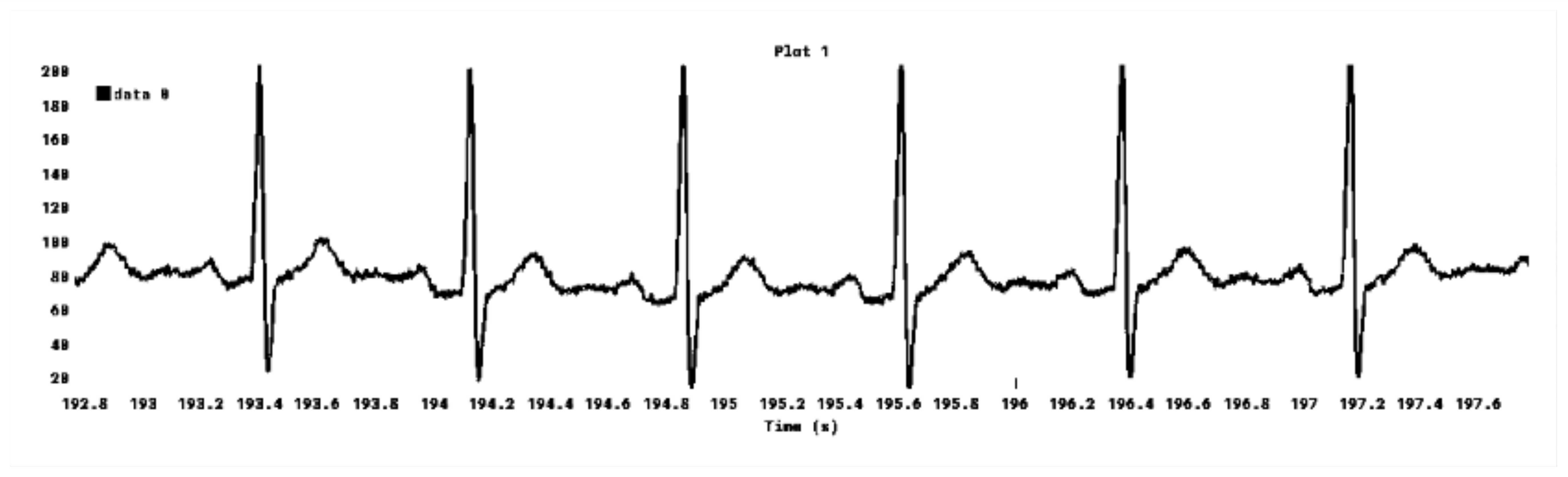
- e) Bilinear high-pass filter (Fc=0.504 Hz) to eliminate the DC and low frequency components from the signal.
- f) Biquadrátic low-pass filter (Fc=40 Hz, gain=6) to act as an anti-aliasing filter.
- g) Voltage source +3V.
- h) Inverter amplifier (gain=0.01) to transform the voltage approximately 0V.



Circuit's block diagram for ECG signal acquisition. The blocks marked with (i) appear by default and are not used. The ACLK utilized was 125 kHz

Results: ECG Acquisition Circuit Test

- Application image showing the ECG signal on computer.




```
#include "FPAA.h"
#include "TimerOne.h"

void setup(){
Serial.begin(57600);
while (!Serial);
if(FPAA.begin("ECG3.ahf")) Serial.println("FPAA_configuration_OK");
  else Serial.println("FPAA_configuration_Error");
FPAA.setClock(F125K);
Timer1.initialize(1000); // Ts=1ms -> Fs=1000 samples/s
Timer1.attachInterrupt(Timer1_ISR);
}

void Timer1_ISR(){
  int ECG_sample = analogRead(A0);
  Serial.println(ECG_sample>>2);
}

void loop(){
while(1);
}
```

Conclusions

- The Arduino Shield with FPAA allows implementation of most different analog circuits with a drag and drop desktop application.
- The flexibility makes possible the implementation of filters, linearizzers, rectifiers and other electronic circuits.
- The errors obtained in the tests were small and may be attributed due to the measurement uncertainties of the meters devices.
- The ECG signal acquisition circuit showed promising results, signal recorded with little noise, since was used unshielded 1.3m cables to connect to the electrodes.
- For future works the goal is to develop acquiring different bioelectrical signals such as EMG and EEG, for example, also a website to disseminate information and details about the shield.
- This work contributed to the biomedical engineering area and embedded system area. We didn't find any similar work.