18-12-2020

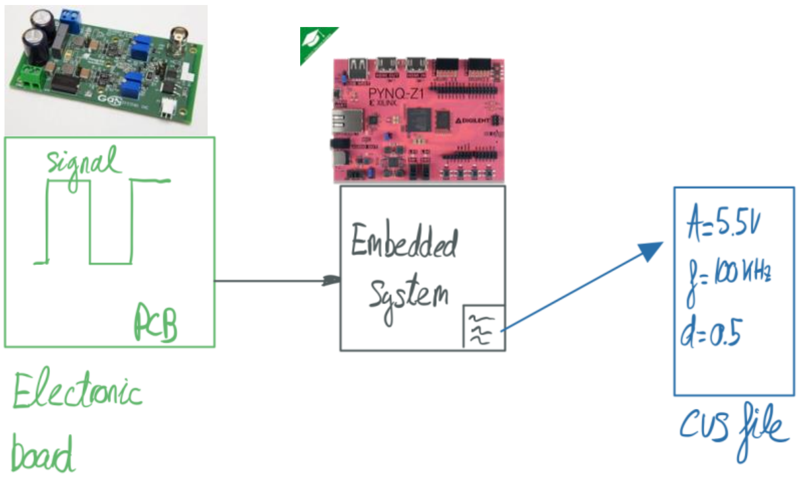
Signal Management

Programming Project

JAVIER GALINDOS VICENTE

PROGRAMMING FOR FACTORY ENGINEERS

Project purpose



The purpose of this project is the following. First of all, there is an electronic setup where signals are generated. We are going to focus within a square wave. This analog signal will be digitalized using an Analog to Digital Converter (ADC) in an Embedded System. The embedded system will save the main characteristics of the signal in a csv file, for further analysis. In the case of a square wave, the characteristics saved will be the amplitude, the frequency and the duty cycle. With these three parameters it is possible to reconstruct a square wave.

This C++ project reads the data from the csv file and storages within a dynamic matrix. Afterwards, the data will be saved within the class Signal. There will be as many objects of the class as there are signals. Finally, the program will “reconstruct” the original signals, plotting them. In the future, the data of the signals could be used to implement Artificial Intelligence techniques to analyze the data and draw conclusions.

A User Interface (UI) is created to interact with the program. The user can:

- Read data from a file.
- Print the data of the signals.
- Add new signal typing the main characteristics.
- Plot the signals.
- Save the data of the signal in a csv file.

Software requirements

The project has been developed within macOS environment. However, it could be compiled and executed in any Unix environment (Linux and macOS).
To execute the project the following requirements are needed: (see README.txt):

- GNU compiler
- Python
- Matplotlib

Programming Elements

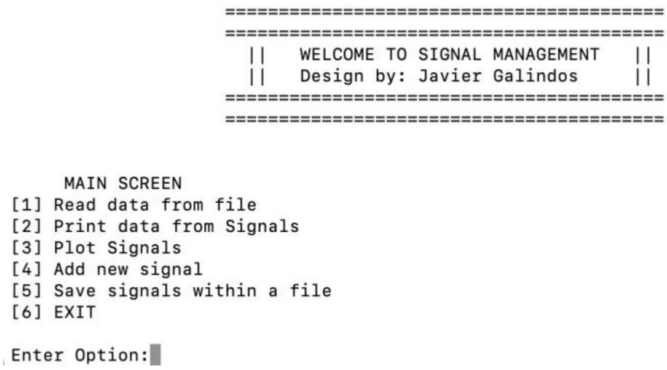
Some of the programming elements used are:

- Organize the project with multiple header and source files:

    - Main.cpp
    - FileManagement.hpp
    - FileManagement.cpp
    - SquareWave.hpp
    - SquareWave.cpp
    - UI.hpp
    - UI.cpp
- Use of files:
    - Read from a csv file
    - Write a csv file
- Use of dynamic data:
    - Dynamic Matrix
    - Vector of dynamic Objects (vector<Signals>)
- Efficient use of memory:
    - The parameters of the functions are passed by reference (use of contest whether do not modify the parameter)
    - Deletion of Matrix/classes when are not necessary anymore.
    - Fflush(stdin) to avoid problems with standard input
- OOP (class SquareWave):
    - Encapsulation (attributes: private, methods: public)
    - Good use of constructor, destructor, getters and setters.
- Use of excpetions (provide a way to react to exceptional circumstances (like runtime errors).
    - Try() catch()
- Use of standard library <vector> and <fstream>
- Use of comercial libraries:
    - Eigen
    - Matplotlib-cpp (Python wrapper)
- Use of makefile
    - Makes easier the compiling and linking process.
- Easy to read/understand source code.

Example of use

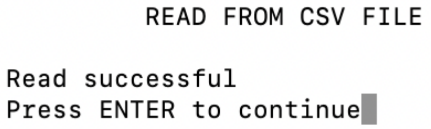
Init screen and main screen.



```
=========================================
=========================================
  ||   WELCOME TO SIGNAL MANAGEMENT   ||
  ||   Design by: Javier Galindos     ||
=========================================
=========================================


     MAIN SCREEN
[1] Read data from file
[2] Print data from Signals
[3] Plot Signals
[4] Add new signal
[5] Save signals within a file
[6] EXIT

Enter Option:
```

Read data from file

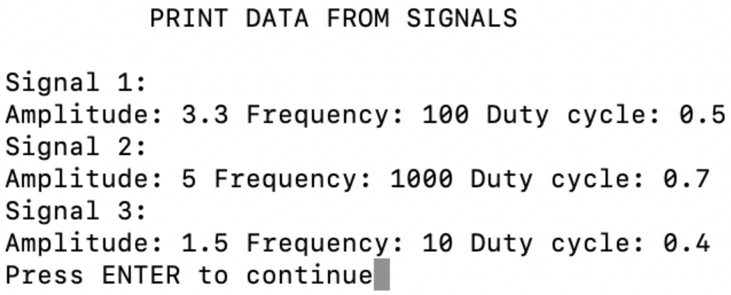
- Reads from csv file
- Storage the date in a dynamic matrix
- Save the data within vector of Objects (Signals)
- Delete dynamic Matrix



```
          READ FROM CSV FILE

Read successful
Press ENTER to continue
```

Print data from Signals.

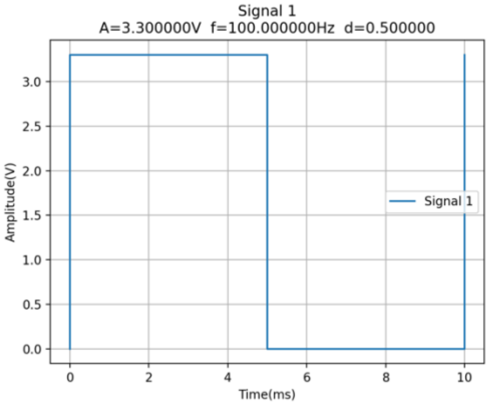
- Print the data of the Signals running through the vector.

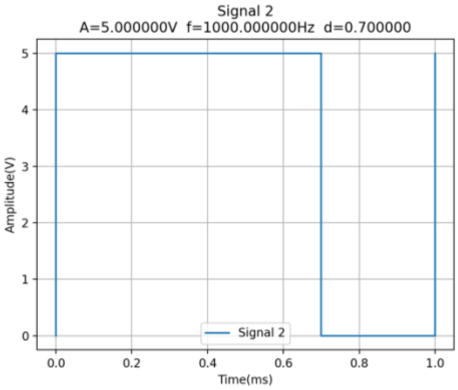


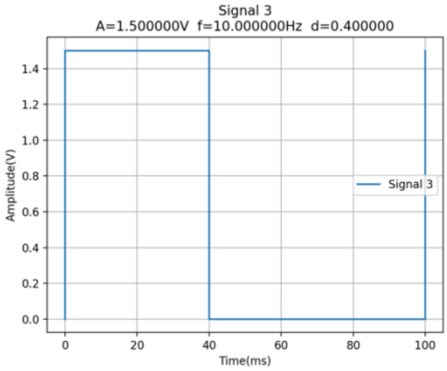
```
         PRINT DATA FROM SIGNALS

Signal 1:
Amplitude: 3.3 Frequency: 100 Duty cycle: 0.5
Signal 2:
Amplitude: 5 Frequency: 1000 Duty cycle: 0.7
Signal 3:
Amplitude: 1.5 Frequency: 10 Duty cycle: 0.4
Press ENTER to continue
```

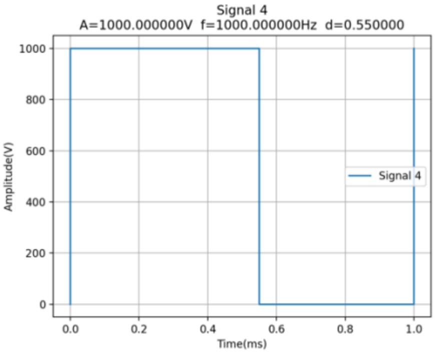
Plots Signals

- Reconstruct the signals and plot them
- The user can save the picture of the signals.



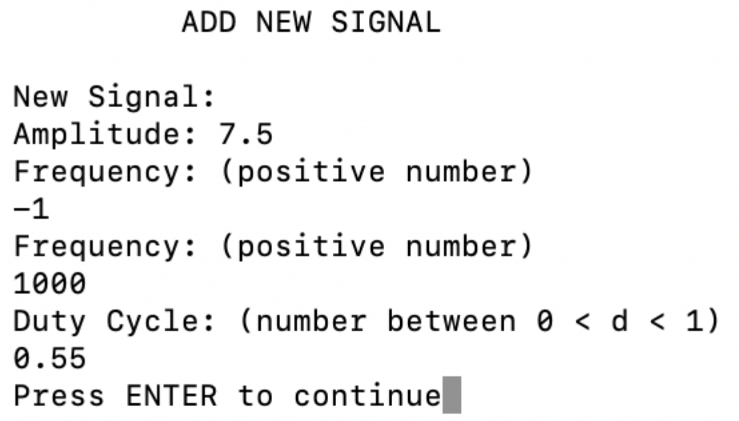






Add Signal

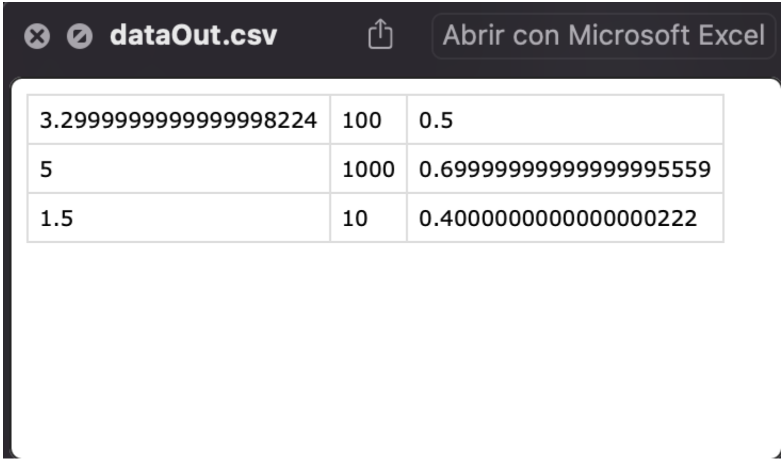
- The user can add a new signal.
- Checks whether the values typed by the user are correct



```
                  ADD NEW SIGNAL

New Signal:
Amplitude: 7.5
Frequency: (positive number)
-1
Frequency: (positive number)
1000
Duty Cycle: (number between 0 < d < 1)
0.55
Press ENTER to continue
```

Save data

- Save the data of the Signals in a csv file
- Deletes the vector of objects



3.2999999999999998224	100	0.5
5	1000	0.69999999999999995559
1.5	10	0.40000000000000000222