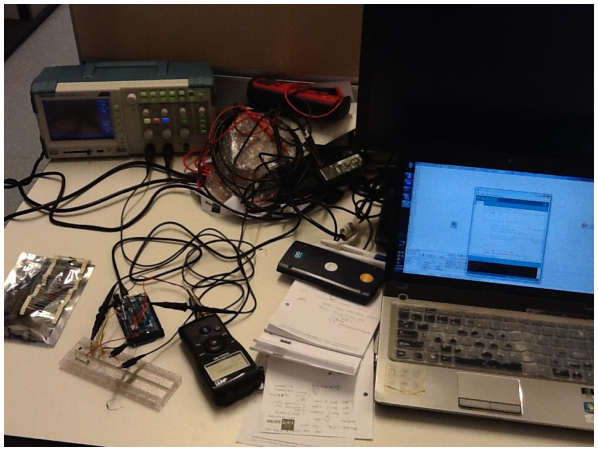
User Manual (v2.0)

Tao Zeng

Csicsvari Group
Institute of Science and Technology Austria
Am Campus 1
Klosterneuburg 3400

2013.09.04



Contents

Chapter 1

Introduction

This user manual includes two systems based on Arduino due. They are kinds of embedded systems, so basically they need both hardware and software support. The open source characteristic for both hardware and software of Arduino satisfied all of the requirements and specifications. And the powerful functions of the 32-bit microprocessor "Arduino Due" is chosen as the core of these two systems.

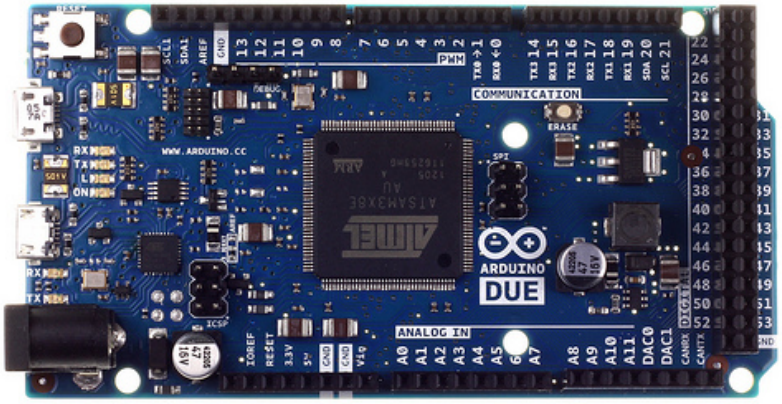


Fig 1. Arduino Due Front View

1.1 Introduction to Arduino Due

The Arduino Due is a microcontroller board based on the Atmel SAM3X8E ARM Cortex-M3 CPU (datasheet). It is the first Arduino board based on a 32-bit ARM core microcontroller. It has 54 digital input/output pins (of which 12 can be used as PWM outputs), 12 analog inputs, 4 UARTs (hardware serial ports), a 84 MHz clock, an USB OTG capable connection, 2 DAC (digital to analog), 2 TWI, a power jack, an SPI header, a JTAG header, a reset button and an erase button.
The board contains everything needed to support the microcontroller; simply connect it to a computer with a micro-USB cable or power it with a AC-to-DC adapter or battery to get started. The Due is compatible with all Arduino shields that work at 3.3V and are compliant with the 1.0 Arduino pinout.

Programming

The Arduino Due can be programmed with the Arduino software (IDE)
Uploading sketches to the SAM3X is different than the AVR microcontrollers found in other Arduino boards because the flash memory needs to be erased before being

re-programmed. Upload to the chip is managed by ROM on the SAM3X, which is run only when the chip's flash memory is empty.

Either of the USB ports can be used for programming the board, though it is recommended to use the Programming port due to the way the erasing of the chip is handled:

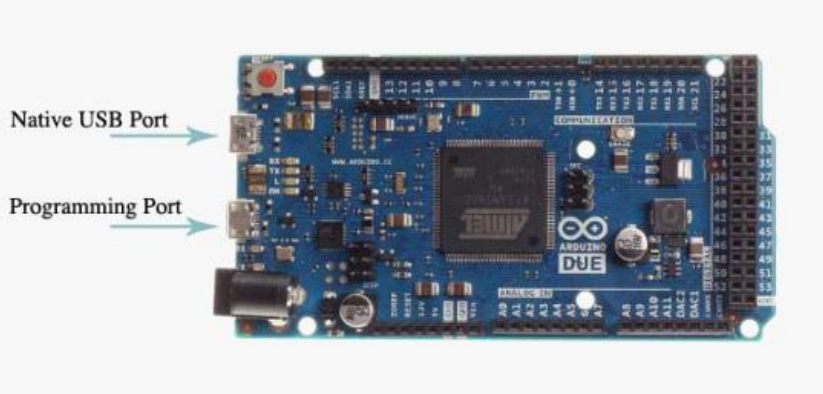


Fig 1.1.1

Programming port: To use this port, select "Arduino Due (Programming Port)" as your board in the Arduino IDE. Connect the Due's programming port (the one closest to the DC power jack) to your computer. The programming port uses the 16U2 as a USB-to-serial chip connected to the first UART of the SAM3X (RX0 and TX0). The 16U2 has two pins connected to the Reset and Erase pins of the SAM3X. Opening and closing the Programming port connected at 1200bps triggers a hard erase procedure of the SAM3X chip, activating the Erase and Reset pins on the SAM3X before communicating with the UART. This is the recommended port for programming the Due. It is more reliable than the "soft erase" that occurs on the Native port, and it should work even if the main MCU has crashed.

Native port: To use this port, select "Arduino Due (Native USB Port)" as your board in the Arduino IDE. The Native USB port is connected directly to the SAM3X. Connect the Due's Native USB port (the one closest to the reset button) to your computer. Opening and closing the Native port at 1200bps triggers a 'soft erase' procedure: the flash memory is erased and the board is restarted with the bootloader. If the MCU crashed for some reason it is likely that the soft erase procedure won't work as this procedure happens entirely in software on the SAM3X. Opening and closing the native port at a different baudrate will not reset the SAM3X.

1.2 Setup Arduino IDE

Windows System:
(1) Go to *http://arduino.cc/en/Main/Software* to download suitable version (Now the

latest version for Arduino due is 1.5.3)
(2) Connect the Due to your computer with a USB cable via the Programming port.
(3) Windows should initiate its driver installation process once the board is plugged in, but it won't be able to find the driver on its own. You'll have to tell it where the driver is.
(4) Click on the Start Menu and open the Control Panel.
(5) Navigate to "System and Securit". Click on System, and open the Device Manager.
(6) Look for the listing named (COM & LPT). You should see an open port named "Arduino Due Prog. Port".
(7) Select the Browse my computer for Driver software option.

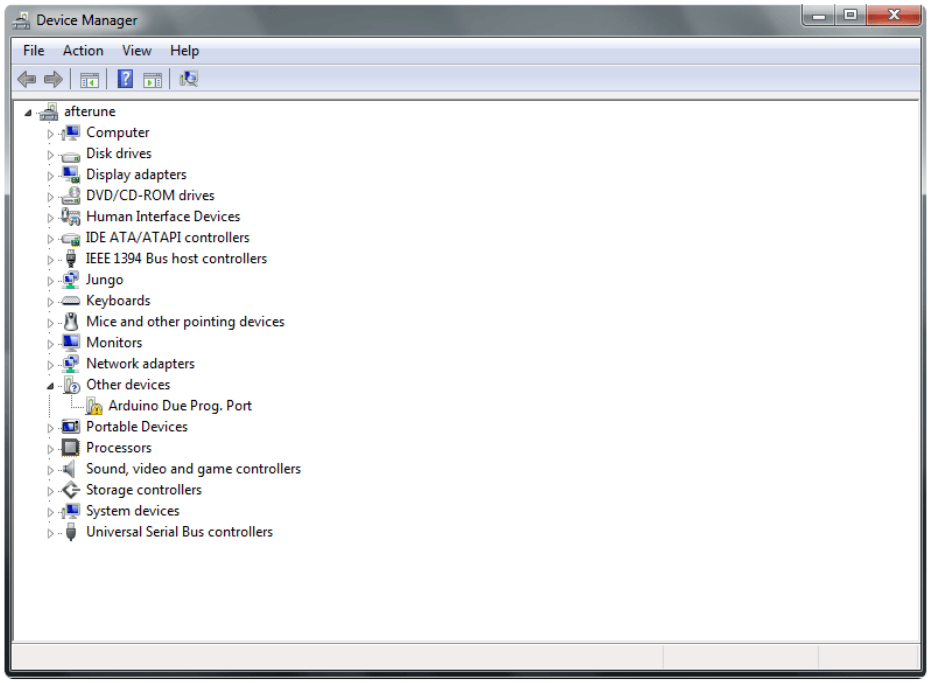


Fig 1.2.1

(8) Right click on the "Arduino Due Prog. Port" and choose "Update Driver Software". Descriptive text.

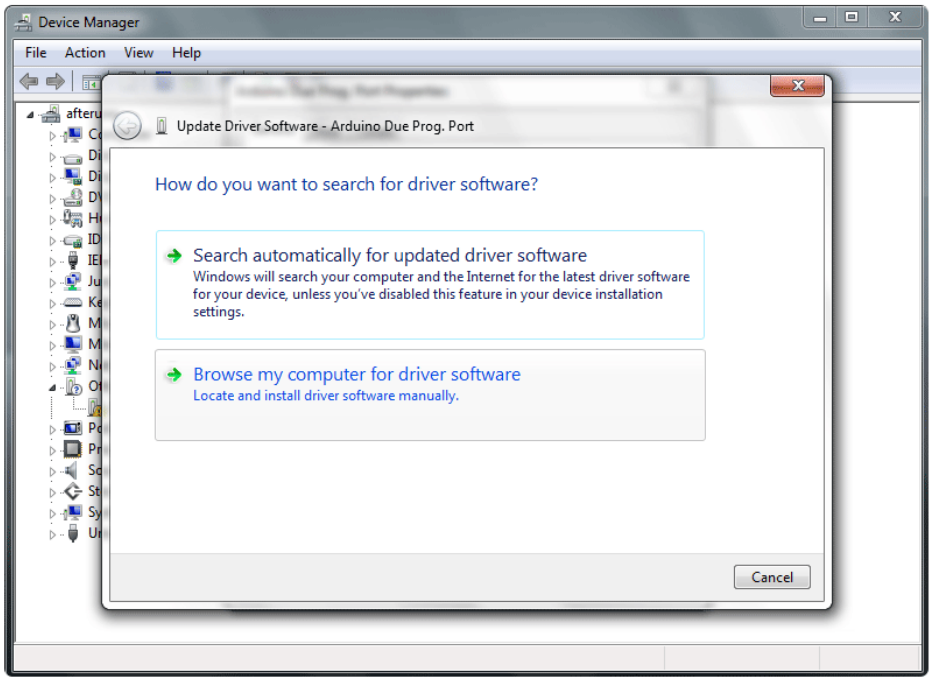


Fig 1.2.2

(9) Navigate to the folder with the Arduino IDE you downloaded and unzipped earlier. Locate and select the "Drivers" folder in the main Arduino folder (not the "FTDI USB Drivers" sub-directory). Press "OK" and "Nex" to proceed.
(10) If you are prompted with a warning dialog about not passing Windows Logo testing, click "Continue Anyway".
(11) Windows now will take over the driver installation.

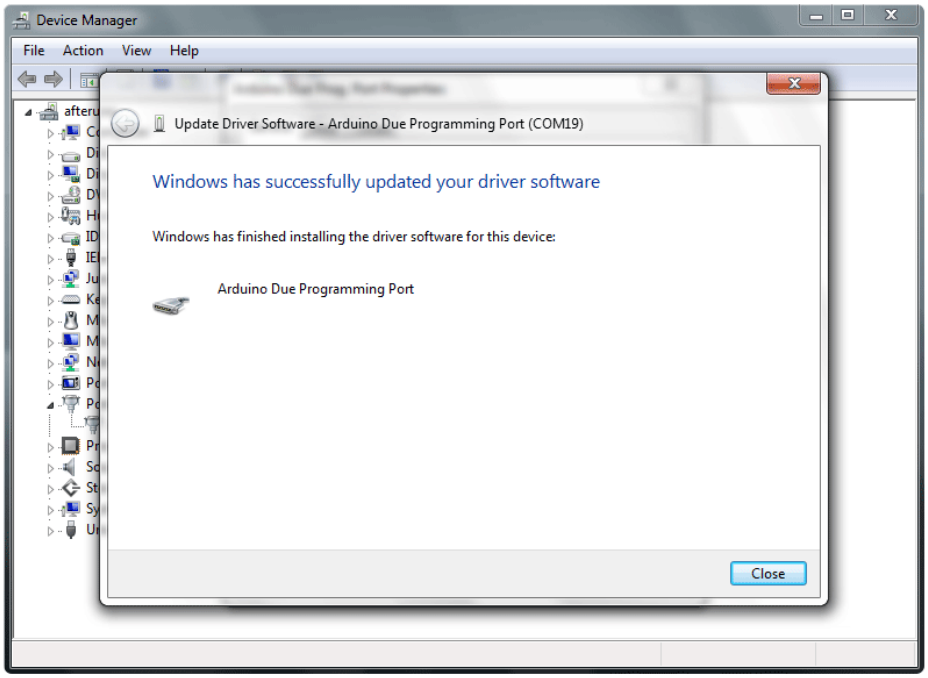


Fig 1.2.3

(12) You have installed the driver on your computer. In the Device Manager, you should now see a port listing similar to "Arduino Due Programming Port (COM4)".

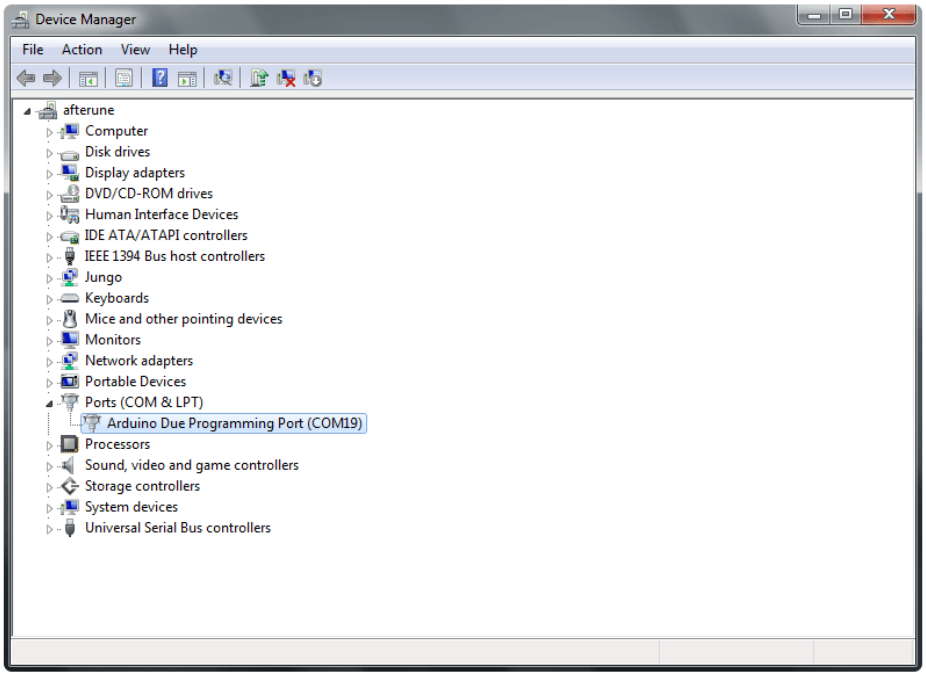


Fig 1.2.4

Linux System (Fedora 17)

Before starting, please ensure you installed all the latest updates with:
$ sudo yum update
Arduino packages are now available directly from Fedora repos, and the install is as minimal as:
$ sudo yum install arduino
If you run the Arduino IDE as a normal user (you should do that of course), you may get this error message : **check_group_uucp()** : error testing lock file creation Error details: **Permission denied**
This is probably because you don't have write permissions on /run/lock directory. Try this to see if this solves your problem : **sudo chmod 777 /run/lock/run/lock** is created on boot by /usr/lib/tmpfiles.d/legacy.conf (both on systemd and initscripts). So if you want to keep the permissions you set, simply copy the file:
/usr/lib/tmpfiles.d/legacy.conf to /etc/tmpfiles.d/
and edit it there (set 0777 permissions).

1.3 How to Use Arduino due IDE

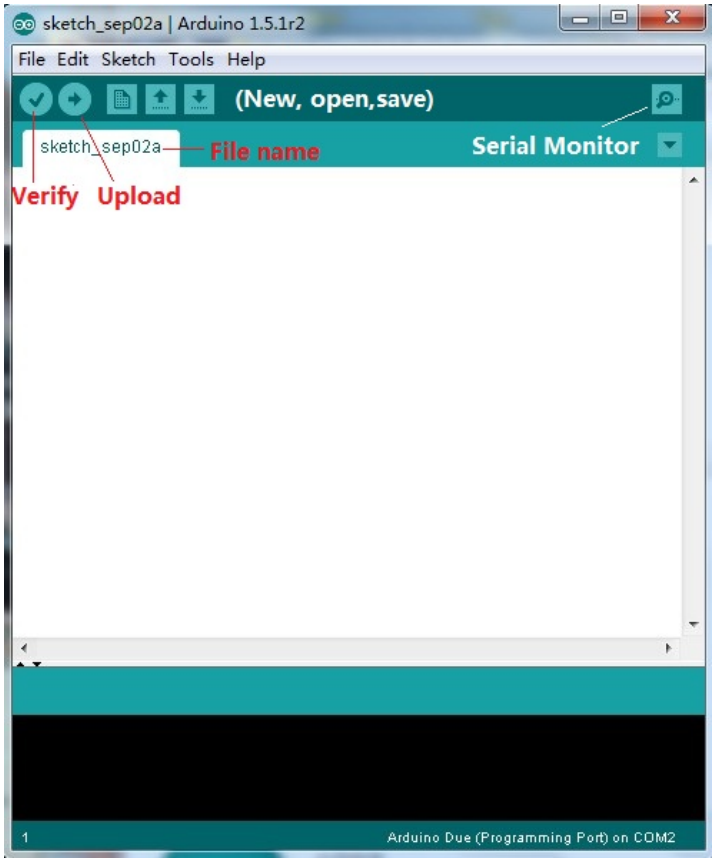


Fig 1.3.1

Verify: Similar to “complie”, to verify if it is any bugs in your program.
Upload: Upload=Verify+Upload your program to Arduino. Don’t disconnect the USB wire when it is uploading.
New: Creat a new file, format:“‘date+sequence’.ino”.
Open: Open an “.ino” file.
Save: Save current files.

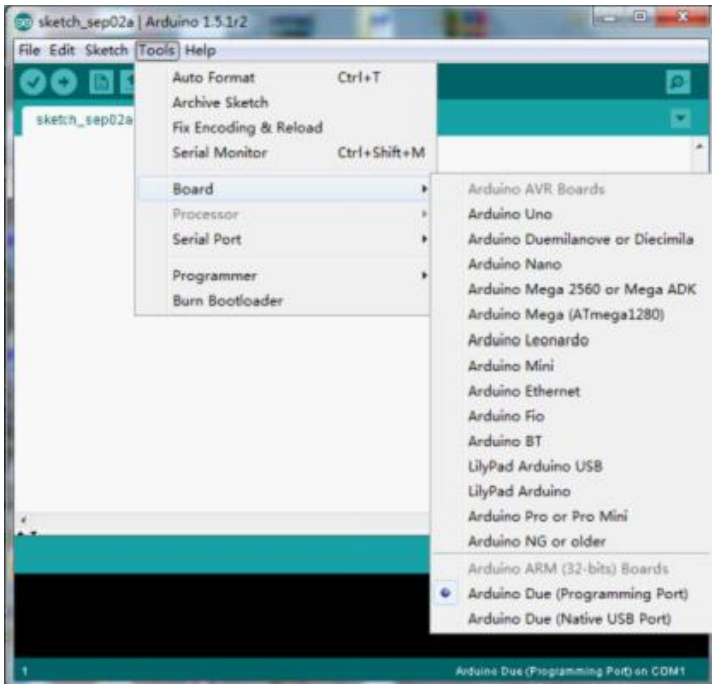


Fig 1.3.2

When you open Arduino IDE, you should make sure that which board you are using now. These two systems both use **Arduino Due** and they are supposed to use **programming port** to upload programs.

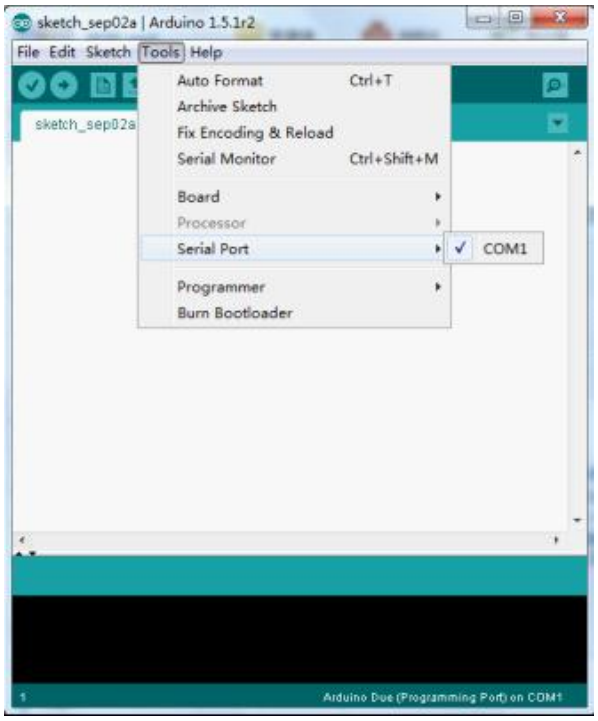


Fig 1.3.3

Choose the correct serial port you are connecting to.

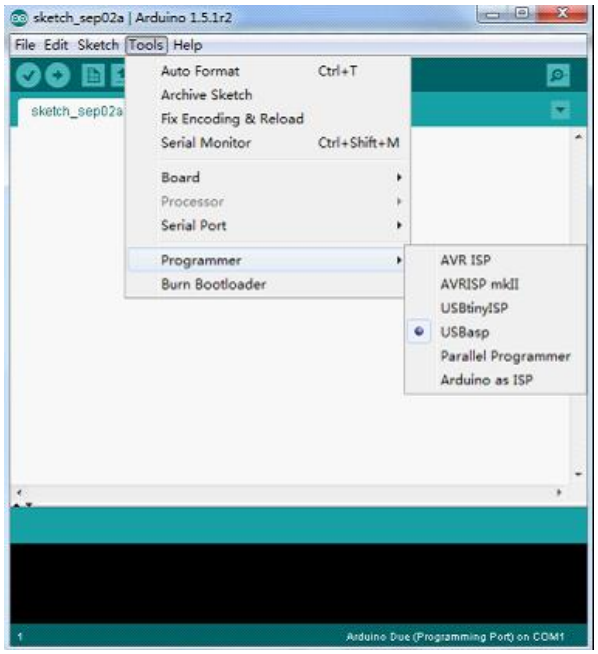


Fig 1.3.4

Because we use mini USB wire to upload programs, so we use USBasp as the programmer.

Chapter 2

Flexible Sharp Wave Ripple Detector

The basic idea of this system is to build a band pass filter, keep the signal with the certain frequency interval and cancel the signal with frequency out of the interval.
This system use window functions to design the band pass filter and after getting the output, user can choose a window with flexible length to calculate the signal power inside this window. Then detect the SWR through a threshold value.

2.1 Introduction to Band Pass Filter

FIR Band Pass Filter is applied in this system. It has a linear phase, which means the time delay is linearly related with the frequency component of signals and this property make the output signal without distortion.

2.1.1 Window functions

There are totally four kinds of window functions provided:
(a) Rectangular window: Small main lobe, low side lobe attenuation, large power leakage.(Not suitable for SWR detection, just for testing)
(b) Hanning window: Larger main lobe, high and fast side lobe attenuation, small power leakage.
(c) Hamming window: Improvement of Hanning window, smallest power leakage.(Better performance if signal passband is fixed)
(d) Kaiser window: Most complicated one, but it is also the most accurate, flexible one. It can adjust the main lobe width and side lobe attenuation at the same time. (Recommended)
The detail source code are named **"Rectangular.c", "Hanning.c", "Hamming.c"** and **"Kaiser.c".**

2.1.2 Input and Output

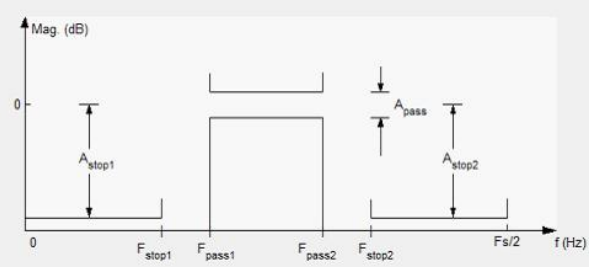


Fig 2.1.2.1 Kaiser window design model

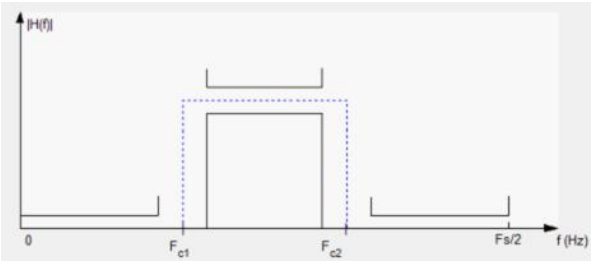


Fig 2.1.2.2 Retangular, Hanning, Hamming window design model

–	Input Parameters	Output Parameters
Rectangular	Fs Fc1 Fc2 Main lobe width	Filter length & Filter coefficients
Hanning	Fs Fc1 Fc2 Main lobe width	Filter length & Filter coefficients
Hamming	Fs Fc1 Fc2 Main lobe width	Filter length & Filter coefficients
Kaiser	Side lobe attenuation Fs fs1 fp1 fp2 fs2	Filter length & Filter coefficients

Table 2.1.2.1

A file named **"data.txt"** will be created automatically, There are the design results (window length and window coefficients) inside.
Copy the window coefficients to the **"BandPassFilter.c"** (replace the static array) and window length to the **"BandPassFilter.h"** (replace the window length).

2.1.3 Core Algorithm

Convolution is the core algorithm of the digital band pass filter. "BandPassFilter.c" and "BandPassFilter.h" are the source code and head file to realize this algorithm.
For "BandPassFilter.h" the only flexible part is:

"#define BANDPASSFILTER_TAP_NUM"

The number after it should be referred to the window length calculated by computer.
For "BandPassFilter.h" the only flexible part is:

"static int filter_taps"

The coefficient inside should be referred to the window coefficients calculated by computer as well.
These two files are concluded in the main funtion of Arduino IDE project, so all the algorthm will be compiled by Arduino IDE.

2.1.4 Main Function

Main functions are the programs running on Arduino IDE with the filename format "XXX.ino".

–	**Description**
Calibration.ino	To adjust the ADC sampling rate to Certain sampling frequency(0-7000Hz).
DACTEST.ino	To store the test signal sequence and output the analog signal to Second board.
FilteredResult.ino	To get the filtered result directly from the DAC (with both P/N value).
FourChannels.ino	To get the power when there are four channels analog input (Aperopdic).
PowerAperiodic.ino	To get the power through a window with specific length, detecting it by pulse.
PowePeriodic.ino	If the length of signal is known, use this program to replace the signal (save space).

Table 2.1.2.1

2.2 Circuit Schematic

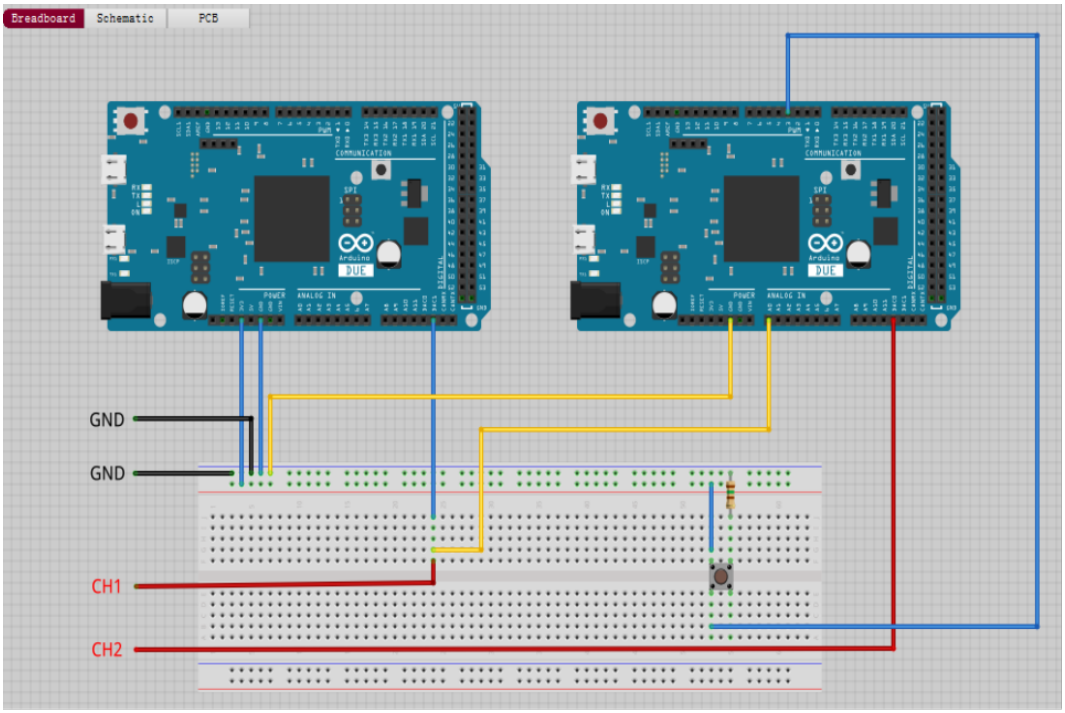


Fig 2.2.1

Warning: Unlike other Arduino boards, the Arduino Due board runs at 3.3V. The maximum voltage that the I/O pins can tolerate is 3.3V. Providing higher voltages, like 5V to an I/O pin could damage the board.
For the test signal, we need to do some modification to make it in the range 0 3.3V:

For data sequence from axona(off-line): Try to use map(value, min, max, 0, 4095) to make the scale fit to the requirement (12-bit ADC has the resolution 4096)
For the real signal(on-line): It can be connect to the analog input of Board II directly. But some signal conditioning circuits should be made to transfer the voltage level of real signal to 0 3.3V as mentioned.

2.3 Setting Procedure

1. Connect the circuit.
2. Choose a window, use a C compiler to run the program. Type the input parameters, get the window length and window coefficients.
3. After getting the window length and window coefficients. Copy it to **"BandPassFilter.c"** and **"BandPassFilter.h"** and save the files.
4. Open Arduino IDE.
5. Open **"DACTEST.ino"** copy the data sequence to the static array, then Upload to Board I. (For real signal, it is not necessary, just skip this step.)
6. Open ''Calibration.ino", adjust the value of **"ADC_MR"** until it close to the user required sampling frequency.(5000Hz is recommanded)
7. Open **"PowerAperiodic.ino"**(for signal with unknown length) or **"Powerperiodic.ino"**(for signal with known length), upload it to Arduino Due.
8. After uploading, you can use the oscilloscope to detect if the system works well (CH1 shows the original test signal CH1 shows the filtered result).
9. Open the Serial Monitor.
10. If it is working. Press **RESET button** on Board II as the starting point of signal recording and Press the **button on the bread board** as the ending point of signal recording.
11. The results are showed in serial monitor, users can copy the data for further analysis.

2.4 Trouble Shooting

2.4.1 Resource Used in The System

Hardware	Two Arduino boards Board I (3.3V, GND, DAC1) Board II (GND, A0, PWM3, DAC0) 1 push button 1 10kilo-ohm resistor Wires
Software	Arduino IDE Rectangular.c Hanning.c Hamming.c Kaiser.c BandPassFilter.h BandPassFilter.c Calibration.ino DACTEST.ino FilteredResult.ino Fourchannels.ino PowerAperiodic.ino PowerPeriodic.ino

Table 2.4.1.1

2.4.2 Common Q & A

Q1: I cannot upload the program with the error "No device found on COMX(ACMttyX)", what can I do?
Answer:
a. Press "Erase" for severals seconds, hold on.
b. Then press "Reset"
c. Release "Erase"
d. Release "Reset" finally.

Q2: I want to write some libraries and include them in my ".ino" files, what can I do?
Answer: Put them in arduino-1.5.2-windows\arduino-1.5.2\libraries, Arduino IDE will search this directory when it is uploading program.

Q3: I uploaded the program successfully, but it doesn't work, what's the reason?
Answer: Notice what serial port you are connected before you upload program.(right down side of Arduino IDE)

Chapter 3

Simple Function Generator

With push buttons, you will be able to choose a waveform shape (sine, triangular, sawtooth, or square) on both DAC channels (DAC0 and DAC1) and change the frequency and amplitude of the generated signal.

3.1 Function & Specifications

Function:

	Mode 1	**Mode 2**	**Mode 3**	**Mode 4**
DAC0	Sine	Triangular	Sawtooth	Square
DAC1	Sine	Triangular	Sawtooth	Square

Table 3.1.1

Specification:

	Vpp	**Frequency**	**Offset**
Sine	0-3.3V	0-350Hz	1.65V
Triangular	0-3.3V	0-350Hz	1.65V
Sawtooth	0-3.3V	0-350Hz	1.65V
Square	0-3.3V	0-350Hz	1.65V

Table 3.1.2

For further improvement, using hardware circuit (Operation Amplifier) to adjust the offset value. Then it can be applicable for testing or calibrating signal to animals.

3.2 Circuit Schematic

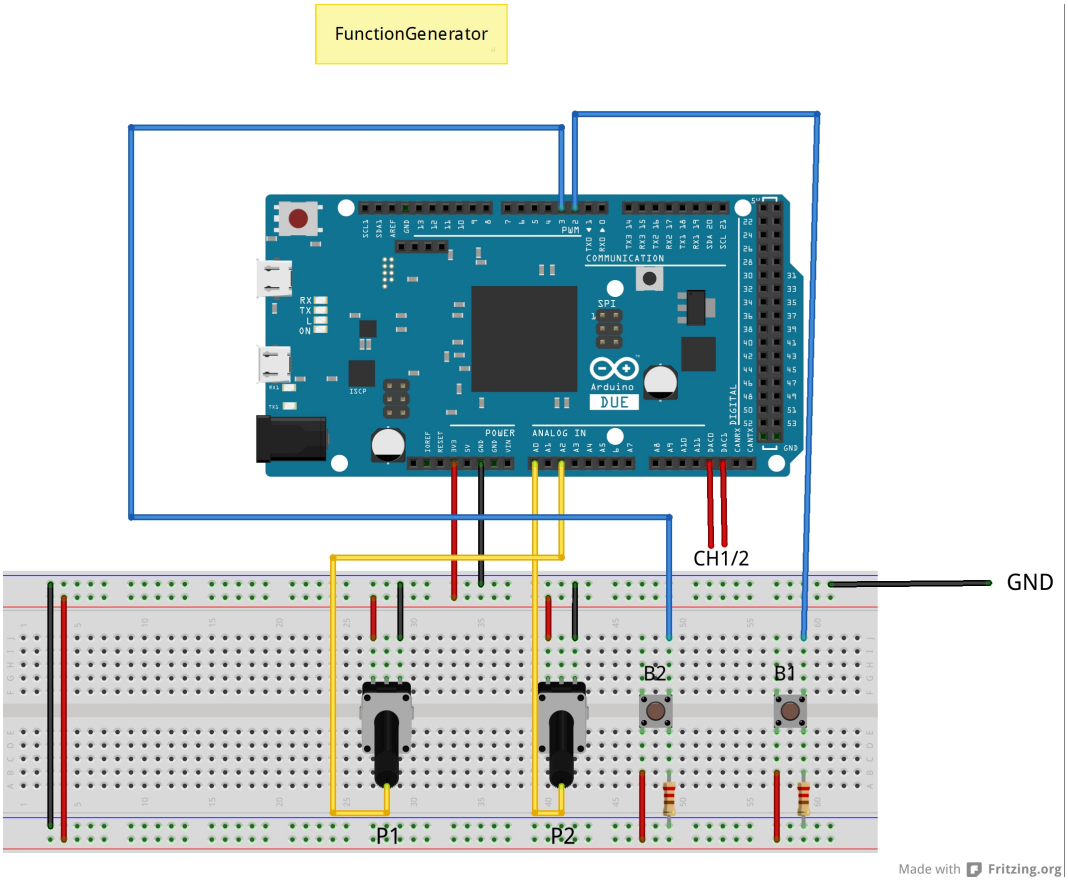


Fig 3.2.1

Hardware Required:

- One Arduino Due Board
- 10 kilo-ohm potentiometer
- 2 push buttons
- 2 10 kilo-ohm resistors
- Jumper Wiresp

P1: This potentiometer controls the amplitude of the waveforms.
P2: This potentiometer controls the frequency of the waveforms.
B1: This button is the switch to change the waveform mode of DAC0
B2: This button is the switch to change the waveform mode of DAC1

Notice: Users can change the **maximum frequency** by setting the value of **ADC_MR**
Higher maximum frequency provides larger range but worse resolution.
Lower maximum frequency provides smaller range but better resolution. So it is a trade off.

3.3 Setting Procedure

1. Connect the circuit.
2. Open Arduino IDE, upload the program "FunctionGenerator.ino".
3. Connect the wire to oscilloscope, detect the waveform.
4. Press the button to transfer the waveforms and adjust the potentiometer to change the amplitude and frequency of signals to a suitable level.

3.4 Trouble Shooting

3.4.1 Resource Used in The System

Hardware	one Arduino board (3.3V, GND, A0, A2 DAC0, DAC1 PWM2, PWM3) 2 10k potentiometers 2 push buttons 2 10 kilo-ohm resistors Wires
Software	Arduino IDE FunctionGenerator.ino Waveforms.h

Table 3.4.1.1

3.4.2 Common Q & A

Q1: How to get low frequency (Such as 10Hz 50Hz)?

Answer: Adjust the register **"ADC_MR"** to realize a lower sampling frequency, then the maximum value should decrease, the resolution in low frequency domain should increase.

Chapter 4

Reference

More Information:

All of the source code can be found at:
1. https://github.com/diegozeng/Filter-Design
2. https://github.com/diegozeng/FunctionGenerator

More Reference:

Official Website of Arduino:
http://arduino.cc/en/