Setup for form measurements with chromatic confocal sensor

Andrea Giura

Setup for form measurements with chromatic confocal sensor

T.R. 23/2022 June 2022

I.N.RI.M. TECHNICAL REPORT

Abstract

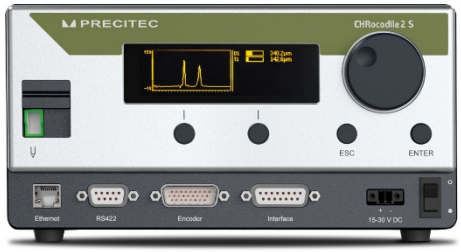
In the following RT is described the activity carried out in order to set up the 3 axis MOORE measuring machine, situated in INRIM's laboratory, to measure samples with the confocal chromatic sensor CHRocodile.

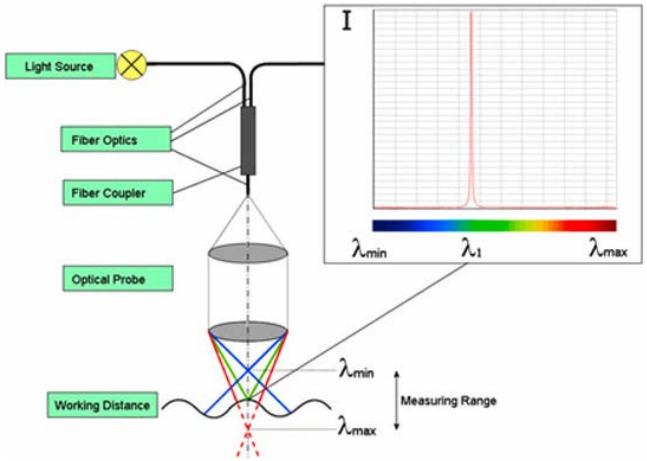
Riassunto

Nel presente RT si descrive l'attività svolta per predisporre la macchina di misura a tre assi MOORE, presente nel laboratorio dell'INRIM, per effettuare misure di forma con il sensore confocale cromatico CHRocodile.

Index

1 CHRocodile chromatic confocal sensor





1.1 Chromatic confocal technology

Incident white light is imaged through a chromatic lens to emit monochromatic light along the z-axis, when an object is present in this colour field, a single wavelength is fixed to its surface and then reflected back to the optical system. The backscattered beam passes through a filtering pinhole and is then acquired by a spectrometer. The beam's specific wavelength is calculated to precisely determine the position of the surface in the measurement field.

The device consists of a control / interface system and a chromatic confocal probe for scanning the sample.

The control system interfaces with the PC via Ethernet connection, it is therefore possible, with a library provided by the manufacturer, to implement the instrument reading in C ++ in order to save and process the data coming from the instrument.

The library includes the main functions for managing the instrument, starting from data recovery, up to the choice of the probe and the ambient parameters.

2 Movement of the measuring machine

To move the sample, it was decided to use the MOORE three-axis measuring machine, normally used for measuring diametrical samples and optical scales. The axes of the machine are moved by three stepper motors, controlled by three drivers that generate the waveforms necessary for the operation of the motors. The control of the drivers is described in the next paragraph.

On the X axis is mounted a laser interferometer which provides, through the GPIB protocol, the absolute position of the axis.

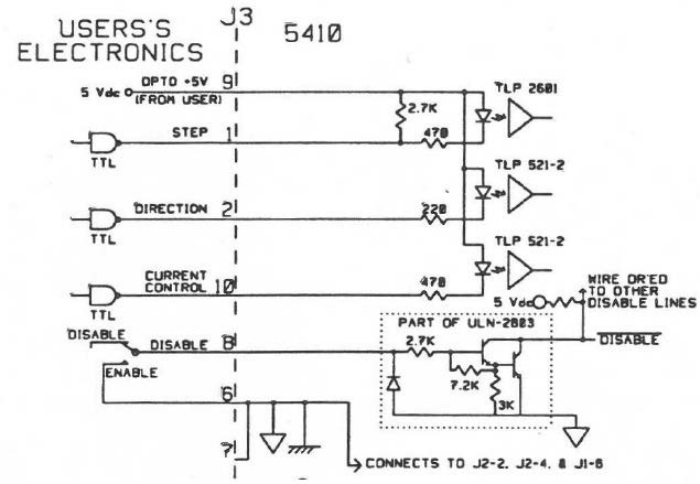
The reading of the position of the Z axis is relative to the sample being measured and is given by the confocal instrument. The Z axis movement is only used to keep the probe in its range of use during the measurement.

2.1 STM 32

A microcontroller of the STM32 family, NUCLEO-F401 RE board, has been used to move the axes of the measuring machine. The device has the function of creating the waveforms for the stepper motor driver:

- Enable / disable
- Direction back / fore
- Variable frequency square wave.

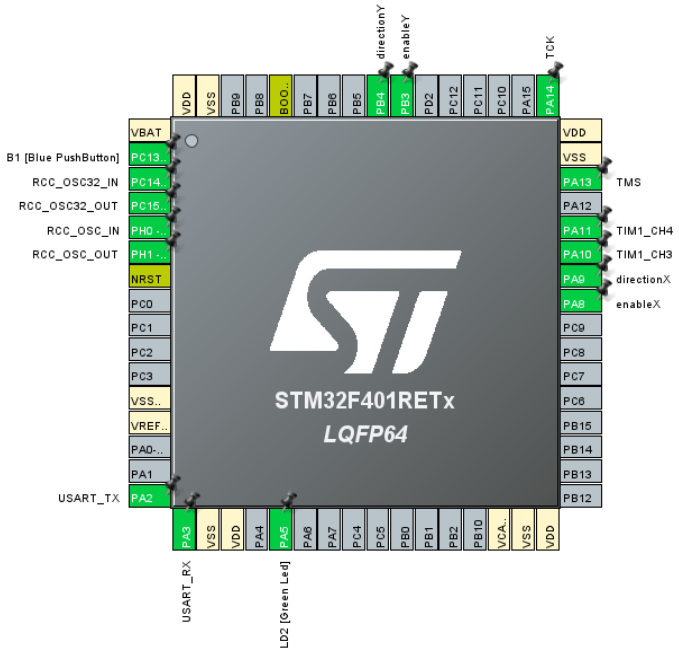
In the following schematic is reported the input configuration of the motors' drivers.



The driver accepts TTL signals at its input, the microcontroller has outputs in CMOS logic (3.3V) 5V tolerant, the enable and direction pins have therefore been configured as open-collector outputs, inserting a pull-up resistor to move the output characteristic between 0 V and 5 V.

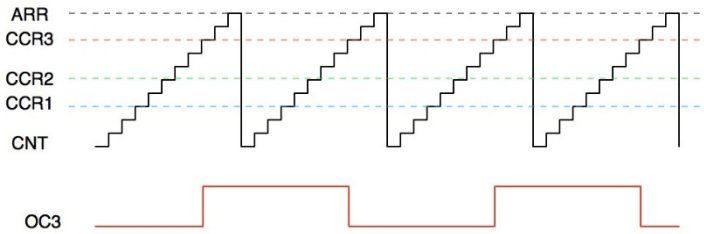
As for the square wave, the push-pull configuration was used to ensure a reasonable switching speed of the GPIO, to optimize the project it would be possible to use a CMOS - TTL converter to make the most of the push-pull outputs of the GPIO.

The pin configuration of the microcontroller is shown below:



2.1.1 Square wave generation

To generate the variable frequency square wave, the general pourpose timer (TIM1) of the microcontroller was used in output compare mode, configuring the output pin of the TIM channel in toggle alternate function mode, so that every time the counter reaches the *CCR* value (capture compare register), the output that controls the step motor switches.



This allows to have a fixed frequency square wave output, which can be calculated based on the clock frequency, the prescaler and the *ARR* (auto reload register) value.

$$f = \frac{f_{clk}}{ARR \times PSC} \; \forall \; CCR$$

To generate a variable frequency wave it is possible to dynamically act on the *CCR* value by increasing it each time by a parameter *T* (instantaneous wave period); to do this, every time the counter reaches the *CCR* value, an interrupt is generated and the relative ISR, which updates the *CCR* value to the next toggle value, is called.

$$CCR_i = CCR_{i-1} + T_i$$

CCR is a 16-bit register, therefore also the variable *T* must be of type uint16_t, this allows to have compatibility in the result of the sum even in case of overflow, in fact:

$$CCR_i > 2^{16} \rightarrow CCR_i = CCR_{i-1} + T_i - 2^{16}$$

This means that, by setting the *ARR* value equal to 2^{16}, there will be a perfect match between the new *CCR* value and the counter value, in this way the frequency will depend on the instantaneous value of the *CCR* and therefore on the parameter *T*:

$$f = \frac{f_{clk}}{PSC \times T}$$

2.1.2 Control of the T parameter

Motor speed must be limited between a maximum and a minimum value, it is therefore appropriate to define:

$$T_{max} = \frac{1}{f_{min}} \rightarrow v_{min}$$

$$T_{min} = \frac{1}{f_{max}} \rightarrow v_{max}$$

The microcontroller receives commands via serial interface, the microcontroller USART has been configured to receive data on 8 bits and generate an interrupt every time a byte is received.

The motor speed has been mapped to values between 1 and 255 (0 used for another function), where 1 indicates its minimum speed and 255 its maximum, it is possible to associate the corresponding value to T with the following equation:

$$f = \frac{1}{T} = (x-1)\frac{(f_{max} - f_{min})}{255-1} + f_{min} = (x-1)\frac{\left(\frac{1}{T_{min}} - \frac{1}{T_{max}}\right)}{255-1} + \frac{1}{T_{max}}, \qquad x \in [1, 255]$$

It must be taken into account that *T* is represented as an integer on 16 bits, since the TIM1 *CCR* is itself a 16-bit register, this means that the formula must be modified to ensure that there are no data losses in the divisions.

To do this it is sufficient to scale the numerator and denominator by a common factor, the optimal solution would be scaling by ($T_{max} * T_{min}$) but this value cannot be represented in a 16 bit variable so the final equation is only scaled by (T_{max}):

$$T = \frac{1}{f} = \frac{1}{(x-1)\frac{\left(\frac{1}{T_{min}} - \frac{1}{T_{max}}\right)}{255-1} + \frac{1}{T_{max}}} * \frac{T_{max}}{T_{max}} = \frac{T_{max}}{(x-1)\frac{\left(\frac{T_{max}}{T_{min}} - 1\right)}{255-1} + 1}$$

2.2 Microcontroller – PC interface

For the movement of the axes it is necessary that the PC coordinates the motor control with the reading of the position instrument present on the relative axis, to do this was used the RS232 protocol between the micro and the PC.

The configuration of the STM32 USART is shown below:

Basic Parameters	
Baud Rate	57600 Bits/s
Word Length	8 Bits (including Parity)
Parity	None
Stop Bits	1
Advanced Parameters	
Data Direction	Receive and Transmit
Over Sampling	16 Samples

The micro waits for each byte and generates an interrupt to manage the received command. The commands for controlling the motor are listed below:

- L (lock): Upon receipt of the character, the micro blocks the motor
- U (unlock): Upon receipt of the character, the micro unlocks the motor
- F (forward): Upon receipt of the character, the micro sets the direction of the motor forward
- B (backward): Upon receipt of the character, the micro sets the direction of the motor backwards
- V (speed): Upon receipt of the character, the micro waits for the speed values until it receives a 0.

To dynamically set the motor speed it is therefore sufficient to send:

$$'V' \rightarrow v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_n \rightarrow 0$$

Sending zero does not block the motor, if the motor is not blocked, by sending the 'L' character after zero, it will continue to move at minimum speed since TIM1 never ends the count.

2.2.1 Axis selection

The measurement with the confocal sensor involves the movement of the three axes of the MOORE machine, the Y axis is controlled manually by the operator, the X and Z axes, on the other hand, are both managed by the microcontroller. This means that the STM32 must be able to know whether the PC intends to move the first or the second axis. To do this, it was decided to implement a potentially scalable solution to N axes. Before each command, a character that identifies the axis that the micro must control is sent to the USART (this does not apply to speed control).

For example for the movement of the X axis:

$$'XU' \rightarrow 'XF' \rightarrow 'XV' \rightarrow v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_n \rightarrow 0 \rightarrow 'XL'$$

2.2.2 Acceleration and deceleration ramps

To guarantee the correct movement of the motors it is necessary that the axis speed follows a linear trend, both in acceleration and in deceleration, to do this the PC must calculate the instantaneous speed and send it to the microcontroller.

This is done by the setVelocity function, which calculates the instantaneous speed based on the distance travelled by the axis and the parameters set by the user.

$$\begin{cases} v = a * travel, & se\ trav < \dfrac{dist}{2} \\ v = a * (dist - travel), & se\ trav > \dfrac{dist}{2} \end{cases}$$

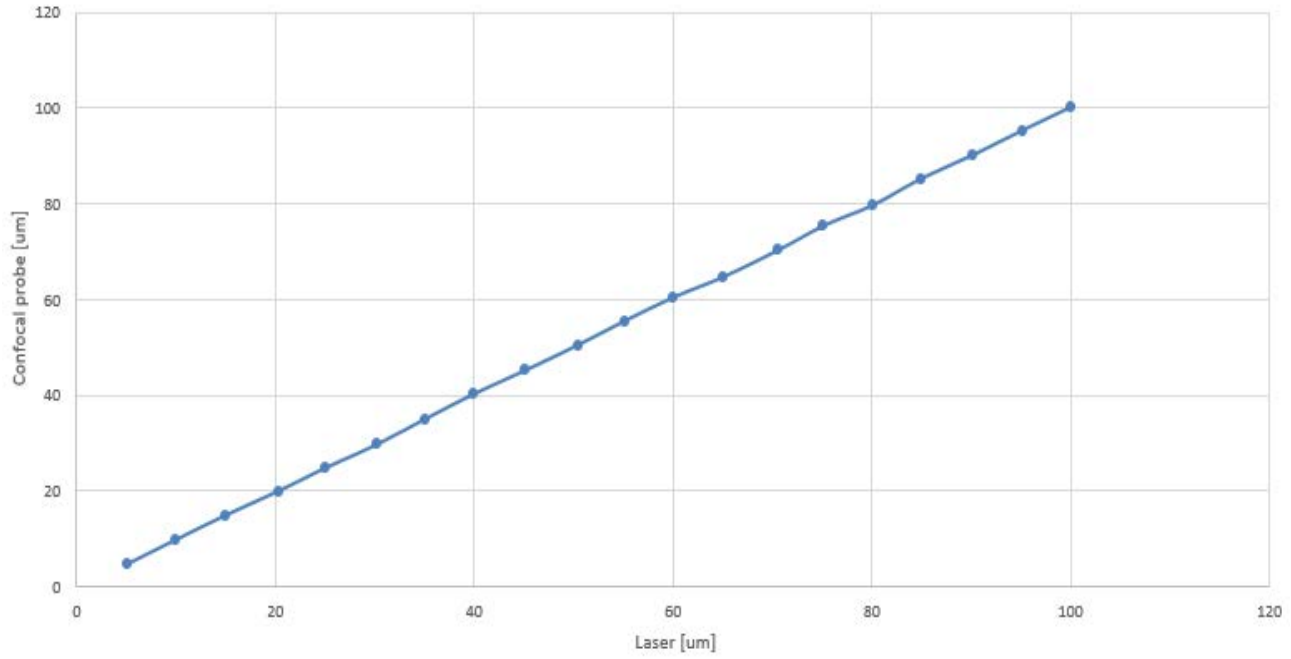
After calculating the speed, the function limits the result to the maximum and minimum values to be sent to the microcontroller (1,255), in this way the following speed curve is obtained.

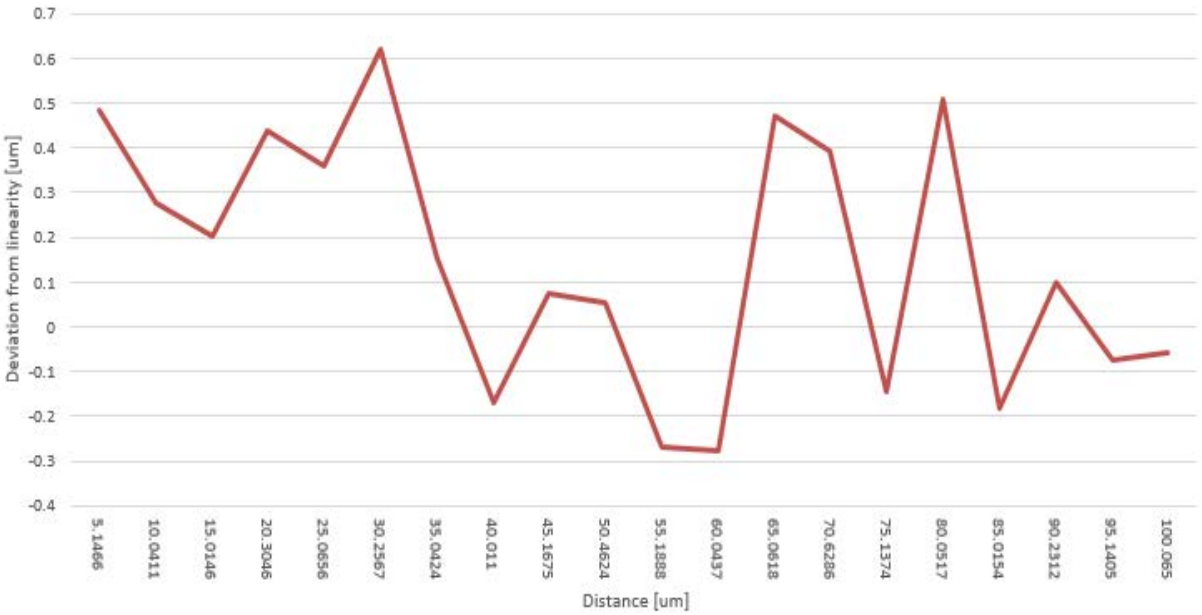
In the figure *a*=2, *dist*=10, v_{max}=4 e *travel*=x;

3 Confocal probe

3.1 Calibration

For the calibration of the probe connected to the CHRocodile device, the reading of the sensor was compared with the value of the X axis laser, mounting the probe parallel to the X axis. The range of the probe is 100 µm, it was decided to carry out the calibration on 20 points equally spaced within the limits of the CHRocodile probe range:

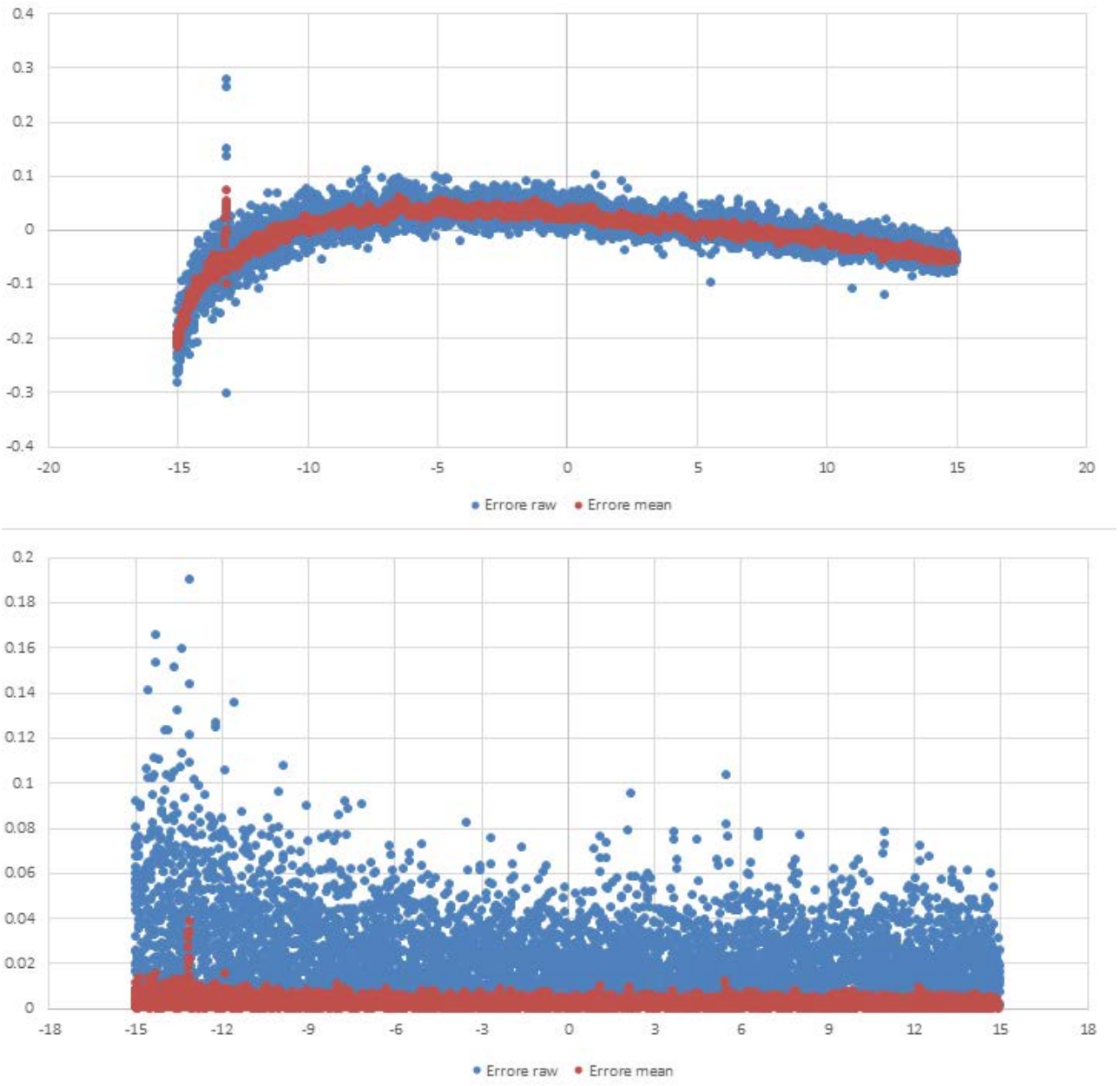




3.2 Noise characterization

By measuring a mirror, it is possible to characterize the noise of the confocal probe, the measurement was carried out on 30 mm to collect a reasonable set of samples.

A moving mean of 10 values was also applied to the samples to see the difference of the fluctuations between the averaged values and the raw samples, the figure shows the deviation from linearity in both cases:



DEVIATION FROM LINEARITY AND CONSECUTIVE SAMPLE DIFFERENCE

4 Stitching

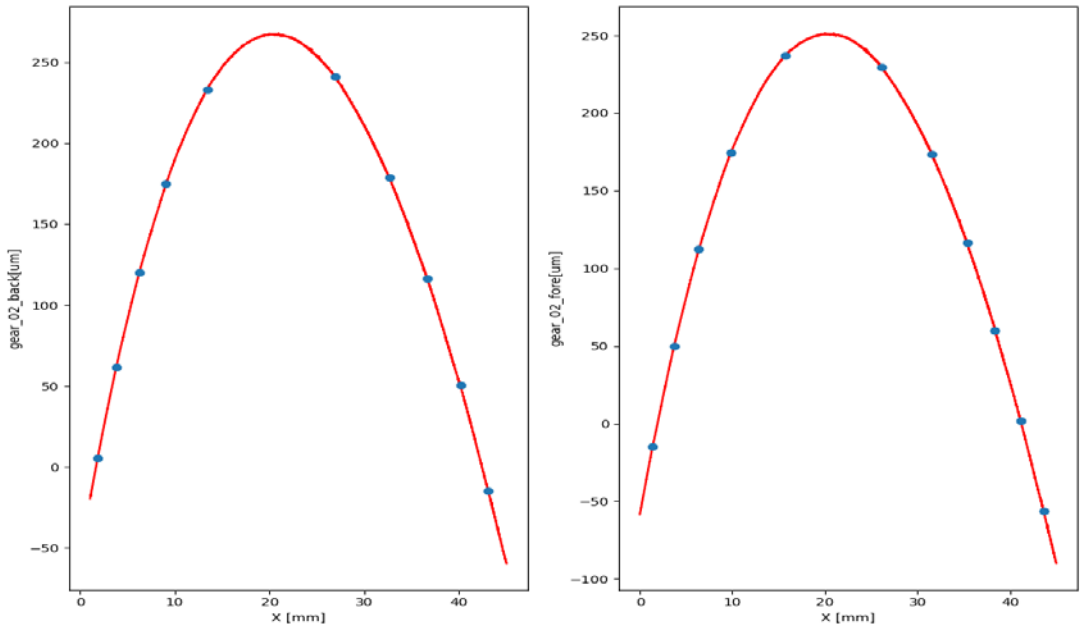
The setup must be designed for shape measurements of samples with height greater than 100 µm, this means that the confocal sensor would work outside its nominal range. To solve the problem, the Z axis is programmed to bring the probe back into its working range whenever the upper or lower limit is reached.

Whenever this is done, it is necessary to correct the data coming from the sensor in order to take into account the Z axis displacement, the measurement routine implemented takes care of all the calculations.

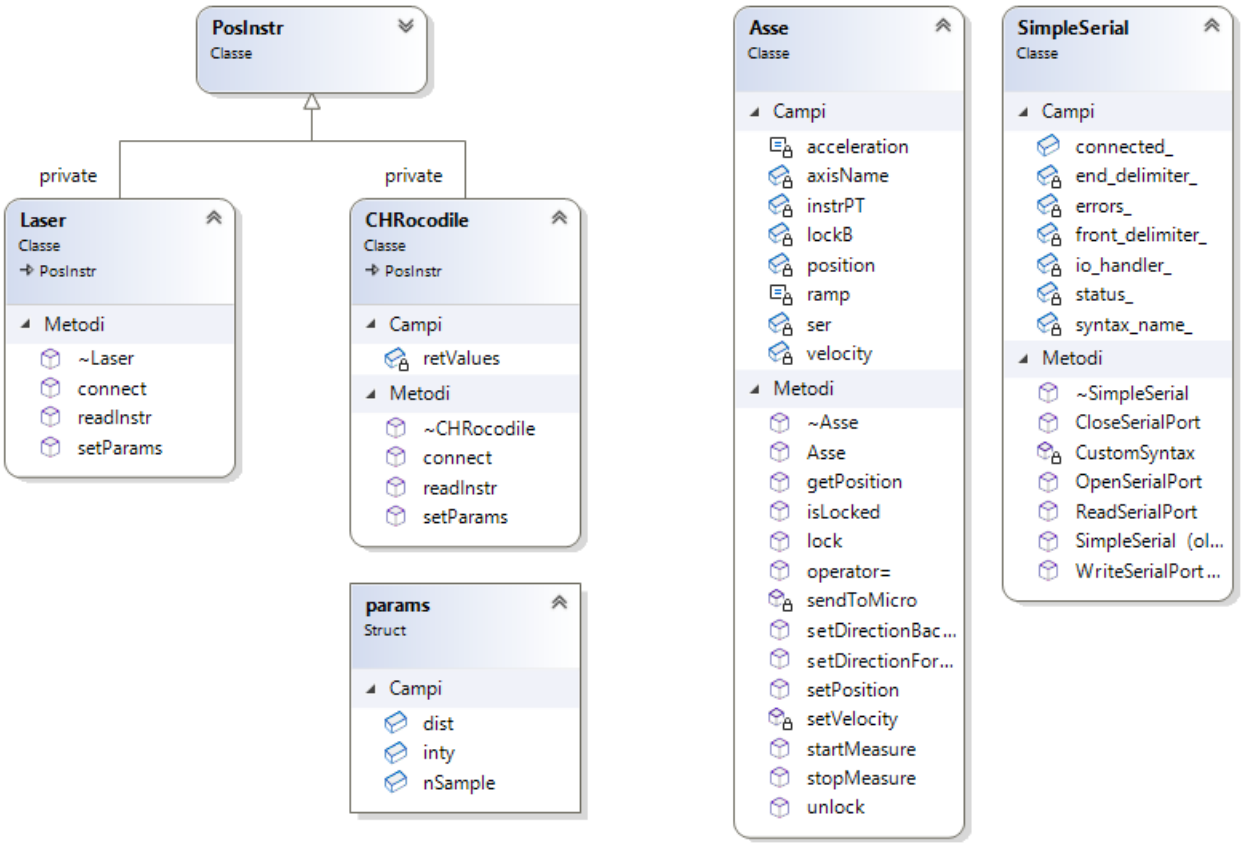
```
while (dis < 85 && dis > 15)
{
	pos = las.readInstr();
	dis = CHR.readInstr();
	std::cout << "ok " << dis << std::endl;
	ost << pos << '\t' << dis + totalDisplacement << std::endl;
	Sleep(100); // a point every 100 ms

	if (GetKeyState('S') & 0x8000) // press S to end measure
	{
		end = true;
		break;
	}
} // while in range
asseX.stopMeasure(); // stop
if (dis > 85)
{
	asseY.setPosition(25);
}
else if (dis < 15)
{
	asseY.setPosition(75);
}
if (!end) {
	Sleep(8000); // wait for y axis to slow down
	totalDisplacement += dis - CHR.readInstr();
}
```

In the program the limit values are 15 µm and 85 µm, the return values are respectively 75 µm and 25 µm.

The figure shows how the sensor, using the Z axis stitching method, is able to measure the form of a gear tooth (height difference > 250 µm), stitching points are represented in blue:



5 C++ implementation



Example: axis movement

```
int main()
{
	try
	{
		SimpleSerial ser;
		connectMicro(ser); // connessione per entrambi gli assi

		CHRocodile CHR;
		Laser las;

		Asse asseY((PosInstr*) &CHR, ser, 'Y');
		Asse asseX((PosInstr*) &las, ser, 'X');

		std::cout << asseX.getPosition() << "\t" << asseY.getPosition();
		asseY.setRamp(2.0f, 180, 200, 5, inv_mov);

		float to;
		std::cin >> to;
		asseY = to;

		std::cout << asseX.getPosition() << "\t" << asseY.getPosition();
	}
	catch (std::runtime_error & _e)
	{
		std::cout << "Terminating: " << _e.what() << std::endl;
	}
}
```

References

[0] Project: https://github.com/andeledea/TestSerial

[1] *Simple serial library*, dmicha16, https://github.com/dmicha16/simple_serial_port

[2] *CHRocodileLibAPI reference,* Precitec

[3] STM32 *User manual*

[4] *Chromatic confocal technology,* Polytec