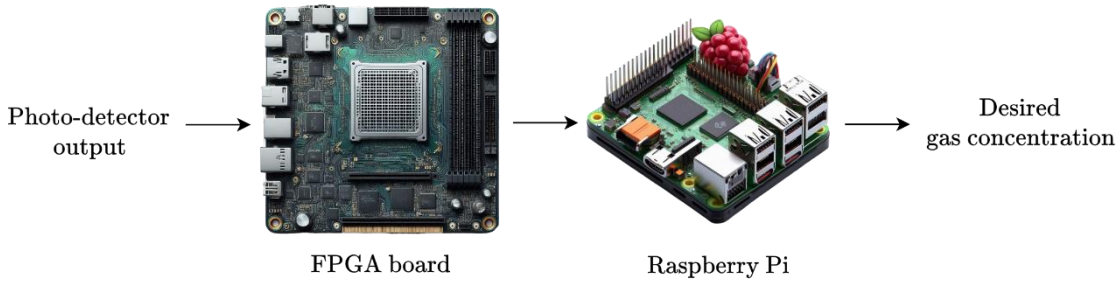
SRIP 2024 Report

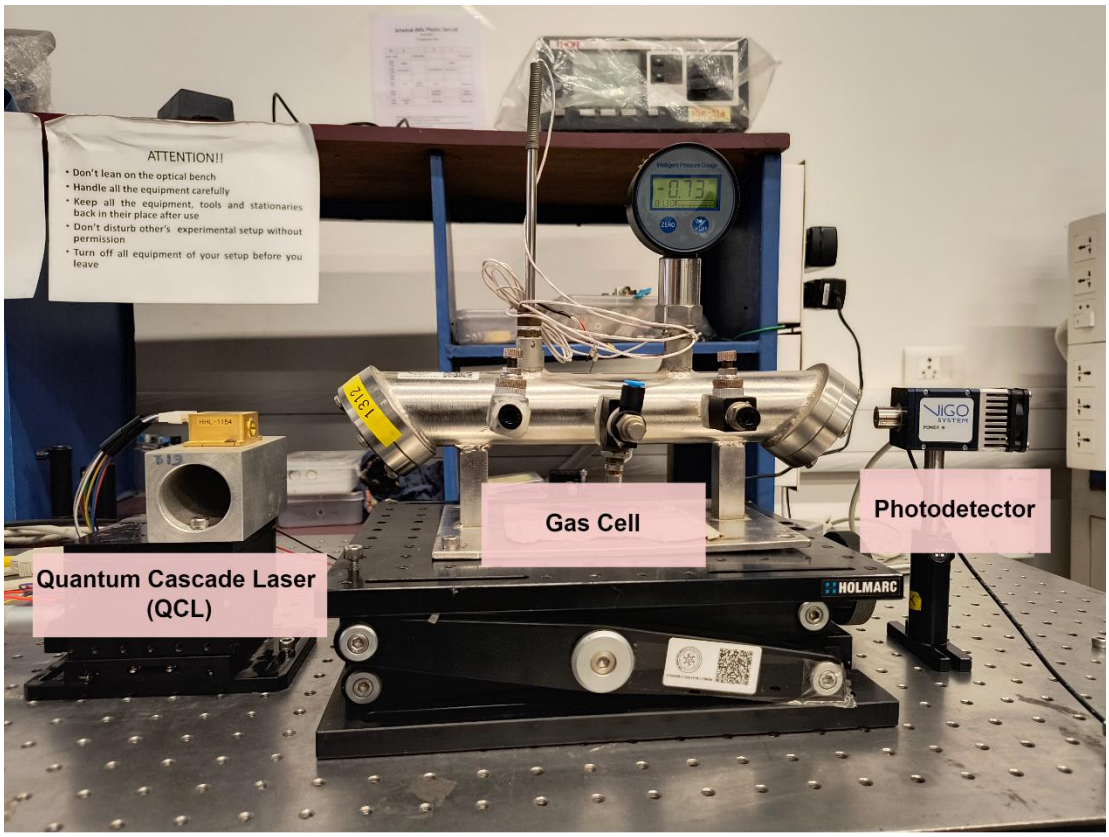
Parag Sarvoday Sahu
Electrical Engineering, IIT Gandhinagar

Aim: To implement a lock-in amplifier on an FPGA board and enable high-speed data transfer (for lock-in detected data) to Raspberry Pi through the Serial Peripheral Interface (SPI) for further processing.

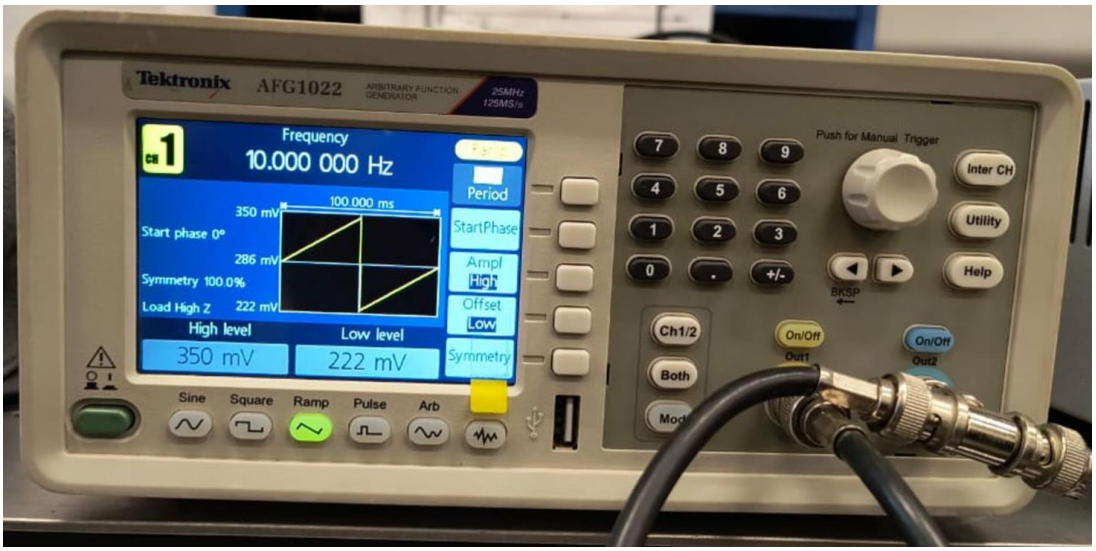


A diagrammatic representation of the aim

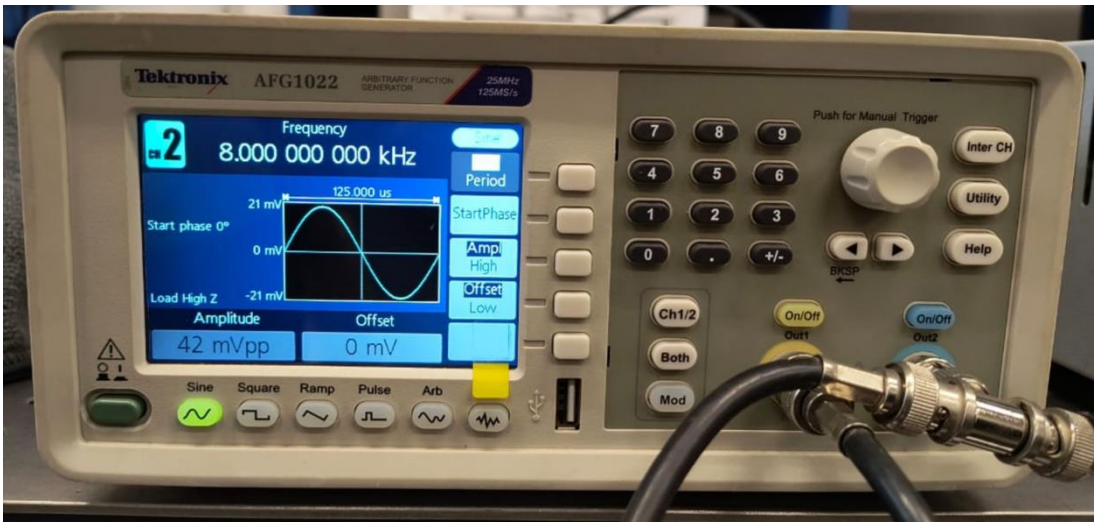
Experimental setup



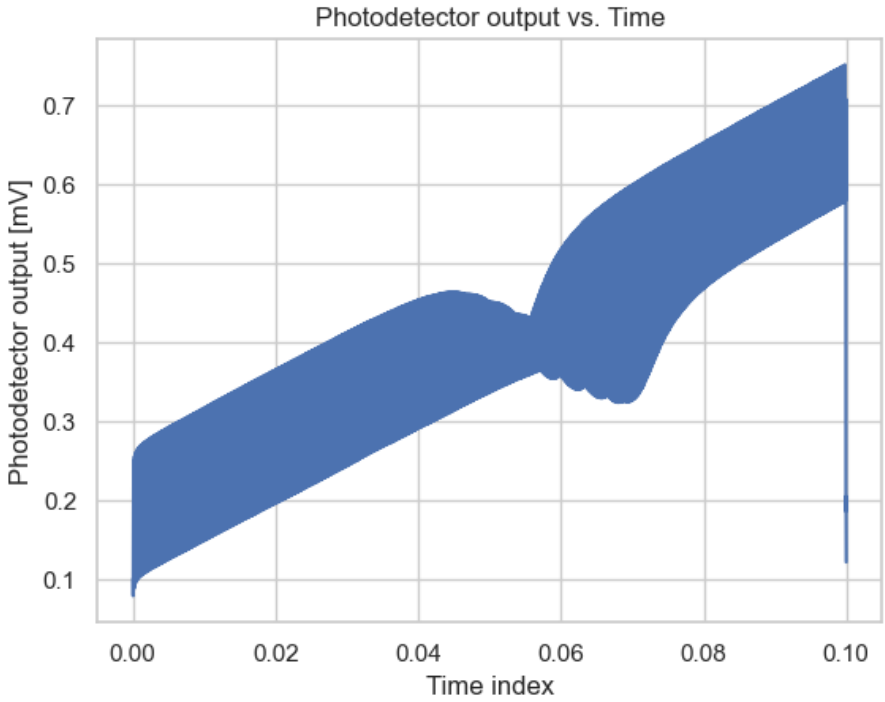
We are performing Wavelength Modulation Spectroscopy (WMS) using the experimental setup as depicted in the picture above. Here, we modulate the laser light with the summation of the following two signals (these signals are for the detection of Nitric Oxide):



A 10 Hz (low frequency) ramp signal



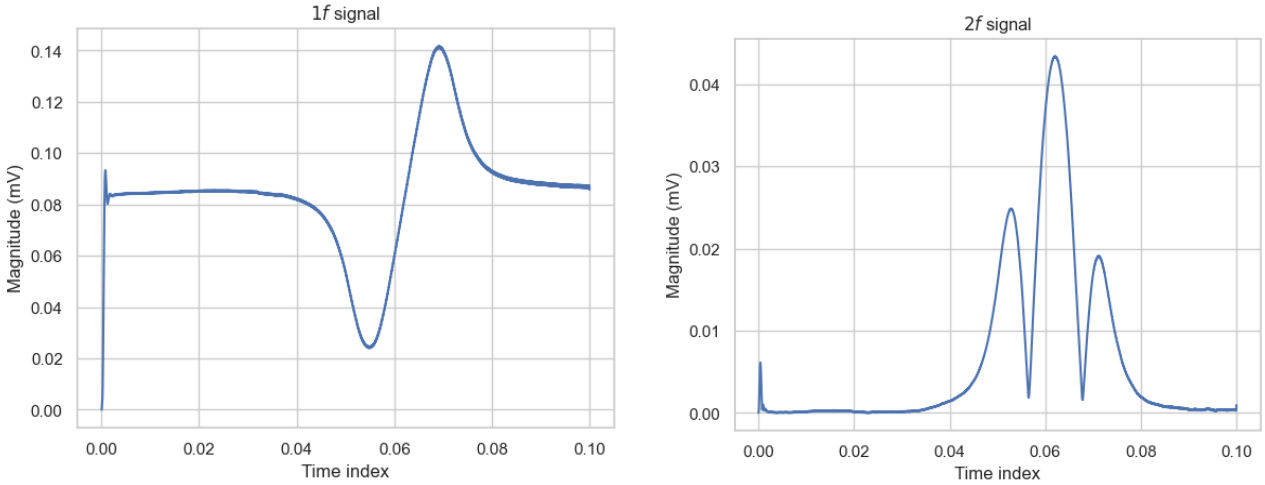
An 8 kHz (high frequency) sine wave with a peak-to-peak amplitude of 42 mV



Raw output from the photodetector after the gas has interacted with the laser light (the dip in the otherwise ramp-like shape of the above curve is due to the absorption of light by the gas molecules)

The gas acts as a non-linear system and introduces some additional harmonics. A lock-in amplifier is used to extract the first-harmonic (1f) and second-harmonic (2f) signals from the photodetector output.

After that, we fit a simulated curve in the 1f and/or 2f signal by minimizing the square of the difference between the two curves. After the curve fitting procedure, we obtain the two unknown parameters in the simulated curve's equation: gas concentration and pressure.



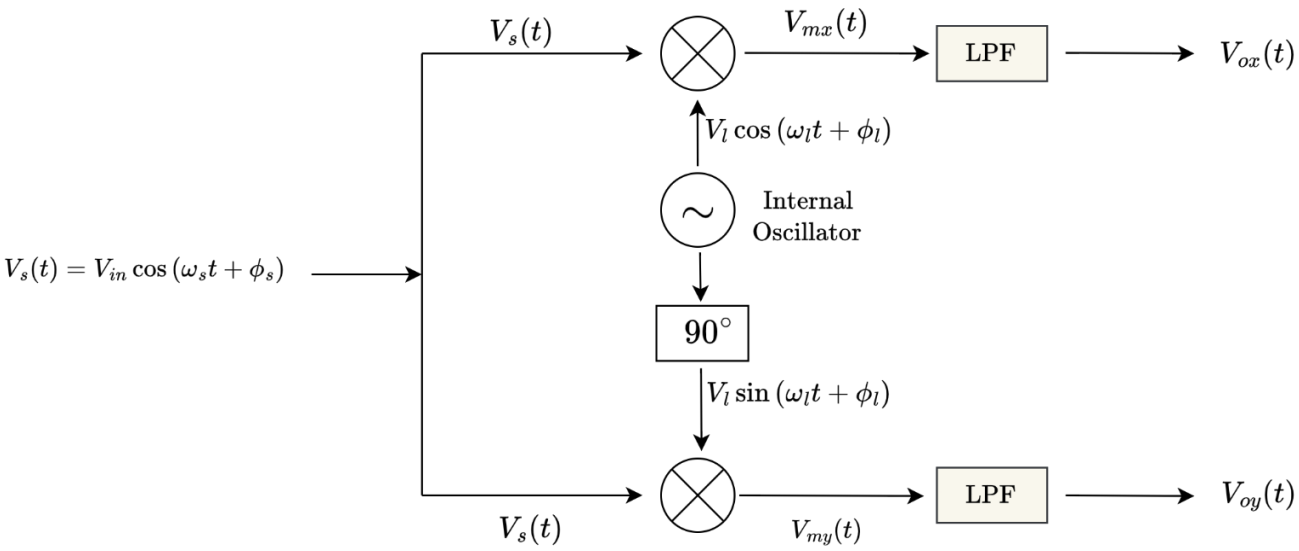
Illustrative 1f and 2f signals after lock-in detection

What is a lock-in amplifier?

A lock-in amplifier is a very narrow bandwidth amplifier, i.e. from the signal given to it as input; it only extracts a signal of a specified frequency (chosen by us). However, some neighbouring frequencies also get extracted as no system behaves ideally. This amplifier can give us the amplitude and phase of the signal at the specified frequency.

We will focus on the extraction of the amplitude of the signal, which is significant to the applications at the Photonic Sensors Lab.

It works in the following way:



The input signal is multiplied with sine and cosine waves of the frequency we want to 'lock on' to, and thereafter, the two resulting signals are passed through low-pass filters. The two resulting signals after low-pass filtering are called the x-component and y-component of lock-in detection.

Some mathematical details of the operations taking place:

$$V_{my}(t) = \frac{V_{in}V_l}{2}\left[\sin\Big((\omega_s+\omega_l)t+(\phi_s+\phi_l)\Big)+\sin\Big((\omega_s-\omega_l)t+(\phi_s+\phi_l)\Big)\right]$$

If $\omega_s = \omega_l$ then after low pass filtering:

$$V_{my}(t) = \frac{V_{in}V_l}{2}\left[\sin\Big(\cancelto{0}{(\omega_s+\omega_l)t}+(\phi_s+\phi_l)\Big)+\sin\Big(\cancelto{0}{(\omega_s-\omega_l)t}+(\phi_s+\phi_l)\Big)\right]$$

$$V_{oy}(t) = \frac{V_{in}V_l}{2}\left[\sin(\phi_s-\phi_l)\right]$$

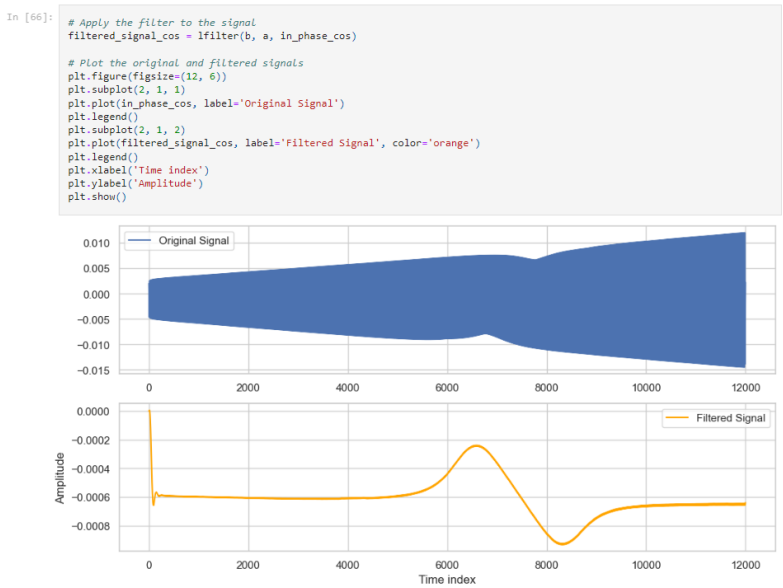
Similarly,

$$V_{ox}(t) = \frac{V_{in}V_l}{2}\left[\cos(\phi_s-\phi_l)\right]$$

$$|V_o| = \sqrt{V_{ox}^2+V_{oy}^2} = \frac{V_{in}V_l}{2}$$

$$\boxed{V_{in} = \frac{2|V_o|}{V_l}}$$

I also created a Jupyter notebook to illustrate lock-in detection for our use case. The notebook can be accessed at this link.



A glimpse of the Jupyter Notebook illustrating lock-in detection

Working of the FPGA-based lock-in amplifier

I continued the work from where a SRIP intern left off the previous year. He had completed the implementation of a lock-in amplifier on FPGA. He was using an R-2R ladder as a digital-to-analog converter (DAC) to view the output signal from the FPGA on an oscilloscope. I worked on figuring out a way of visualising the output from FPGA on our computer screen, which meant implementing real-time data transfer from FPGA. I chose to transfer the data from FPGA to a Raspberry Pi using the Serial Peripheral Interface (SPI) protocol.

Acquiring analog signal from photodetector: The FPGA board that is being used here is Nexys 4 DDR; it has a built-in 12-bit 1 MSPS analog-to-digital converter (ADC). The data from the ADC can be accessed by the XADC module from the IP sources available in Vivado 2022.2.

The operations are performed on the data acquired from the ADC using the IP sources such multipliers, adder and 'cordic' module for implementing the square root operation. The implemented low pass filter has a cut-off frequency of 80 Hz.

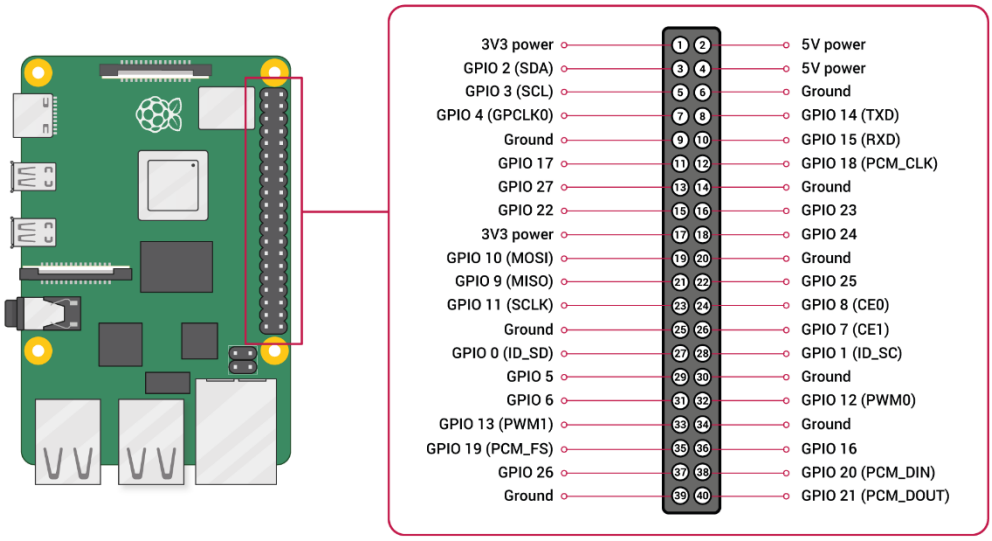
Generation of sine and cosine waves: The sine and cosine waves used for multiplication with the input signal are generated using a look-up table, i.e. we pre-calculate their values beforehand instead of evaluating them in real-time to save up on processing overhead.

SPI-based data transfer from FPGA to Raspberry Pi 5

An excellent resource is this Wikipedia page to understand the workings of the SPI protocol. The SPI configuration that I used in my case is 'MODE 3' where 'CPHA' and 'CPOL' are 1 (this will make more sense after a thorough reading of the Wikipedia page).

SPI protocol uses four wires for data transmission, i.e. SPI clock (SCLK), SELECT, Master-In-Slave-Out (MISO) and Master-Out-Slave-In (MOSI). In our setup, Raspberry Pi acts as a master, and the FPGA board acts as a Slave. Although I had connected the MOSI cable, it is not required in our case as we only need to transfer data from FPGA (Slave) to Raspberry Pi (Master).

The layout of the GPIO pins on Raspberry Pi 5 is as follows:



The pins used on Raspberry Pi 5 are as follows:

MOSI – GPIO 10

MISO – GPIO 9

SCLK – GPIO 11

SELECT – GPIO 8

I used the 'spidev' library to enable SPI communication on the Raspberry Pi. An important insight is that spidev library only supports making the Raspberry Pi a master and NOT a slave. I was trying out this configuration early on and wasn't getting reliable results, so I had to change the configuration later.

On the Nexys 4 DDR board, I used the PMOD ports for SPI connections. Due to a lack of resources in this area, using the PMOD connectors for SPI was not obvious as Nexys 4 DDR does not have dedicated SPI pins. I had to read a lot of articles on the web, devour all the relevant forums, consult a few of my seniors and then I reached the conclusion of using the PMOD ports.

The pins used on the FPGA board are as follows:

MOSI – H2

MISO – F3

SCLK – H4

SELECT – G3

It is also important to have a **common ground between the FPGA board and Raspberry Pi**. I did not have a common ground for a long time in my setup and couldn't establish reliable data transfer. Only after I made the ground common did the communication work reliably.

To implement SPI on the FPGA's end, I took help from this excellent blog written on SPI communication between FPGA and Arduino by Cort Vonk. The blog also helped me strengthen my understanding of how to work with SPI protocol.

I could only achieve an SPI clock speed of 7 MHz with reliable data transfer. However, we require at least 12 MHz SPI clock frequency to transmit data in real-time. Now, the only plausible solution is to first store data on the FPGA board and then transfer it to Raspberry Pi. It will indeed induce some lag in the live feed of the output signal, but it should be manageable.

The files used to conduct all the tasks described in this report can be found in this GitHub repository.

Conclusion

I learned about the wavelength modulation spectroscopy (WMS) method, which detects and measures gases present in trace amounts. I learnt the principle of lock-in detection and made a Jupyter notebook to illustrate the working of a lock-in amplifier as it is used in the Photonic Sensors Lab. I worked with the implementation of a lock-in amplifier on an FPGA board done by an earlier SRIP intern. I also worked on facilitating high-speed data transfer between the FPGA board and the Raspberry Pi using SPI protocol, which is still a work in progress.