**Introduction:**

The integrated Arduino based embedded system performs three functions by sampling analog signals and triggering hardware interrupts with MC74HC595A shift register responses and executing parallel communication with an LCD. The analog sampling system includes a potentiometer and an RC low-pass filter which helps aliasing behaviors by maintaining 250 samples in a circular storage space. An external push button activates the hardware interrupt, which initiates the 8-bit data word shift into the MC74HC595A shift register to enable outputs, after which the full data loading is completed. The system's communication interface operates through parallel communication to transfer data with an LCD display.

**RC Filter Design and Calculations**

The analog signal needs conditioning to eliminate all frequency content that exceeds Fs/2 (Nyquist limit) so aliasing does not occur during sampling.

Consider, Fs=1000 Hz

Thus, the frequency would be:

​​ = 500Hz

To safely remove the unwanted high frequency, I chose the filter’s cutoff to be lower than the Nyquist frequency, which is 70%

To condition the potentiometer output and prevent aliasing, a RC low-pass filter is designed.

The cutoff frequency fc​ of RC filter is determined by the formula:

Now, solving RC as follows

Standard resistor value of R = 4.7 kΩ, then the required capacitance would be

Therefore, a standard value of **100 nF** for capacitor is acceptable.

**Hardware Interrupt and Shift Register Design**

An external push button signal that activates an interrupt on Arduino digital pin 2 will execute the Interrupt Service Routine (ISR). The MC74HC595A shift register accepts an 8-bit data word sequence during its operation within the Interrupt Service Routine of the output system. During the shifting process, the design maintains a high Output Enable (OE) to block the outputs from the shift register. After clocking in the 8-bit word, the latch (ST\_CP) receives a pulse, and OE becomes low to enable the outputs. This mechanism ensures that unwanted information fails to exhibit at the output ports.

**I2C Communication:**

*Using a Parallel LCD Instead of I2C LCD on Arduino:*

Since my kit does not include an **I2C LCD**, I am using a **Parallel LCD (16x2)** instead. Unlike I2C communication, which requires only two pins **(A4 - SDA, A5 - SCL**), a**parallel LCD requires multiple digital pins** for proper operation. The **LCD module operates at +5V** and shares a **common ground** with the microcontroller. Instead of communicating via **I2C**, the LCD is connected in **4-bit parallel mode**, requiring connections to **RS, E, D4, D5, D6, and D7** pins on the Arduino.

**Include screenshot of circuit and LCD Display**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A close-up of wires on a circuit board

AI-generated content may be incorrect.**

**A circuit board with wires and a display

AI-generated content may be incorrect.**

**Conclusion**

The system fulfills its mission by taking **analog signals** through a **properly engineered RC filter,** then detects **hardware interrupts** to transfer an **8-bit word** into an **MC74HC595A shift register**before displaying messages on an **LCD screen using parallel communication** instead of I2C.

The designed **RC filter** produces an appropriate **cutoff frequency (~350 Hz)** using a **4.7 kΩ resistor** and a **100 nF capacitor**, based on hand calculations, to prevent **aliasing** when sampling at **1000 Hz.** This project demonstrates how **analog signal conditioning** interacts with **digital data handling through interrupts** and showcases **parallel LCD communication** as an alternative to I2C protocol.