UCB ECEN 5803: Mastering Embedded Systems Architecture

Project1: Vortex Flowmeter Embedded System

Professor:

Timothy Lee Scherr

Students:

Ibrahim Muadh & Varun Shah

Table of Contents:

1.Executive Summary:

- The goal of Project 1 was to test the STM32F401RE Nucleo-64 board as a capable platform for a Sierra 240V Vortex Flowmeter. Over six modules, we built the system of prototype from scratch—starting with assembly-level routines and ending with a fully functional flow-measurement simulation that could sense, compute, and communicate like an industrial device. Early modules focused on comparing C and Assembly implementations for square-root approximation to study instruction efficiency and memory usage. Using STM32CubeMX, we learnt to automate peripheral setup and learned to coordinate GPIO, ADC, UART, SPI, and PWM under tight timing constraints. Later, it exposed to RTOS-based threading, gaining insight into multitasking and real-time scheduling within an embedded system.
- In the middle modules, I integrated push buttons, analog sensors, PWM LEDs, and UART communication to build a real-time interactive system. Developing a debug monitor allowed me to observe registers, timing, and performance during live operation. The Dhrystone benchmark measured around 105 DMIPS, confirming the Cortex-M4's strength in handling continuous sensing and flow computations. This stage transformed the project from simple labs into a cohesive, self-diagnosing embedded framework.
- The finale, Module 6, tied everything together. Here, the design and code of bare-metal vortex flowmeter simulation was executed. A 10 kHz ADC sampled a synthesized vortex waveform, and my custom frequency-detection algorithm estimated the shedding frequency (~78 Hz typical). Using fluid-dynamics equations and stored calibration constants, the system computed flow every 100 ms while also tracking water temperature (~24.5 °C) through another ADC channel. The results appeared simultaneously on an SPI-connected LCD, through UART, and via PWM and pulse outputs that mimicked the 4–20 mA industrial loop. LEDs pulsed in rhythm with the simulated flow. Timing analysis revealed that the interrupt handler ran only 1–2.5 µs per 100 µs cycle, consuming just 1.3% of CPU time—a testament to both efficient code and a robust processor. Power consumption stayed below 100 mW, and all hardware interfaces performed flawlessly.
- It is therefore, this project is recommended the GO decision.

2.Overview of Modules:

- Module 1: To compare the efficiency of Assembly and C programming on the ARM Cortex-M4 by implementing string operations and an integer square-root approximation using the bisection method. The goal was to understand instruction-level performance, memory mapping, and the ARM calling convention, and to generate documentation using Doxygen.
- Module 2: To learn how to auto-generate startup and configuration code using STM32CubeMX for key peripherals such as ADC, UART, SPI, I^2C, and GPIO. The main objective was to understand how the CubeMX tool simplifies initialization, ensures proper clock configuration, and produces a clean baseline project for further development.
- Module 3: To interface the Nucleo board with real hardware peripherals, including push buttons, LEDs, analog sensors, and serial communication. The goal was to handle digital I/O, interrupt-driven events, analog signal acquisition, and PWM control while visualizing sensor data over UART. This module also focused on testing hardware response and creating wiring diagrams for each interface.
- Module 4: We got exposure to real-time multitasking using FreeRTOS or CMSIS-RTX, and to create multiple threads for handling independent tasks such as sensor reading and display updates. The objective was to understand task scheduling, synchronization, and memory usage, and to analyze how an RTOS improves responsiveness and timing precision in embedded systems.
- Module 5: To design and implement a serial debug monitor using SysTick timer interrupts to schedule background tasks. This module focused on debugging, performance benchmarking, and system visibility, enabling the display of register contents, memory sections, and timing diagnostics through UART. Running the Dhrystone benchmark and measuring CPU utilization were key parts of evaluating system performance.
- Module 6: To develop a bare-metal firmware simulation of a Sierra 240V Vortex Flowmeter using ADC-based frequency detection and temperature measurement. The module's goal was to compute flow rate and velocity using vortex-shedding equations, implement PWM outputs to emulate industrial 4–20 mA signals, and display results on an SPI-based LCD. This served as the capstone integration of all modules—demonstrating full signal acquisition, computation, and real-time output control on the STM32F401RE.

3.Block Diagram:

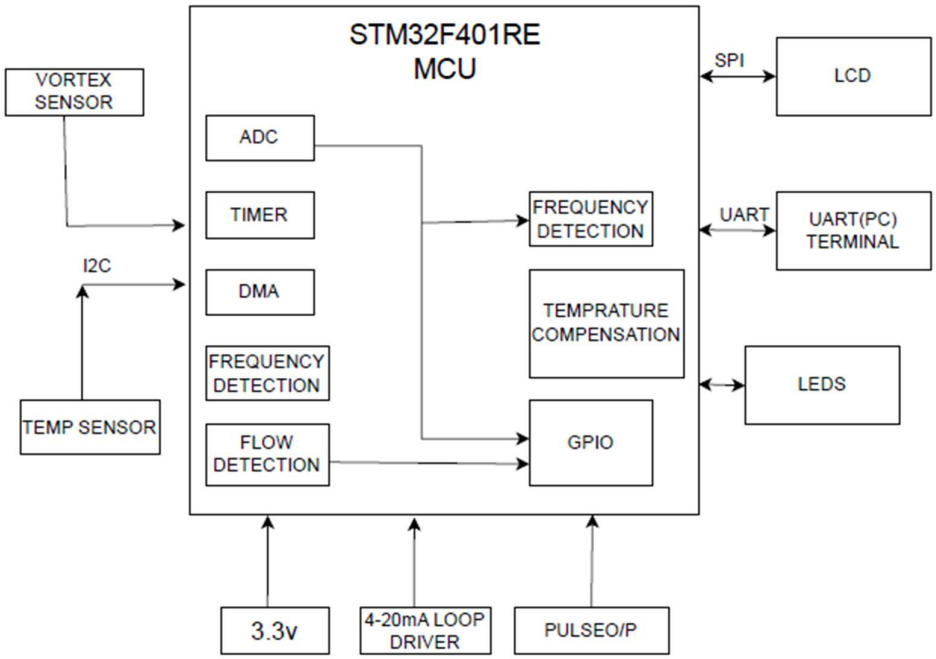


Fig 1: System Block Diagram

4.Test Results for Module:

1.Module1: Assembly C vs C evaluation:

- Assembly function used less memory (1.84 kB) compared to the C function (4.56 kB).
- ARM Cortex-M4 memory model included code at 0x08000194–0x080011DA and data at 0x20000000–0x20000008.
- IRQ handlers were correctly mapped for SysTick, USART2, and EXTI interrupts.
- Square-root approximation used the bisection method in integer arithmetic.
- Tested inputs 2, 4, 22, and 121 produced correct results (1, 2, 4, 11).
- Results were verified in register R3 during debugging.
- Assembly executed faster with fewer CPU cycles and better memory efficiency.

2.Module2: CubeMx setup (Setting up (.ioc))

- In module 2 it was asked to set up the ADC sample clock to near 100kHZ. To achieve that following below steps were taken:

- Setting Prescaler APB2(PCLK2) /8
- ADC resolution was set to 10 bits.
- By above calculations, frequencies were obtained
  84MHz/8 = 10.5Mhz
  10.5MHz/8 = 1.3125MHz
  1.3125MHz/13 = 100.961KHz
- Similarly to set SPI clock near to 100KHz following steps were taken:
- Setting Prescaler APB2(PCLK2) /8
- Setting (SPI prescaler/128)
- By above calculations, frequencies were obtained
  84MHz/8 = 10.5Mhz
  10.5MHz/128 = 82.0312KHz
- As mentioned in the PROJECT GUIDE 1 it was mentioned to keep the GPIO output to PA5 but after it was conflicting to configure the SCLK pin for SPI port as it was also asked to set PA5, so with taking permission from TA Damini GPIO output pin was configured as PC5.
- CPOL set to high and CPHA to edge 1 to get SPI mode 2.
- For SPI port configurations like CS pin set to PB6, not a hardware, NSS pin, PA4 for ADC input.
- Observation: It was observed that after setting up the .ioc file and it directly generates the code from it, it is easier to use HAL library but major drawback is it consumes more ROM/RAM.

3. Module3: Button Read, ADC Read, LED PWM, UART

- Build a system of real time push button, ADC read, LED, PWM and UART.
- We generated PWM signal by 8Khz using timer TIM2.
- Push Buttons: Toggled LED using debounce logic.
- Temperature Sensor showed 24.2-degree Celcius.
- UART transferred the live temperature value and counts of iteration.
- LCD displayed the live temperature.
- I/O schematic.

4. Module4: RTOS Threads

- To manage communication and sampling threads were created.
- For this module RTOS was used to handle multiple tasks at the same time simultaneously like ADC, UART o/p and LCD display.
- Calculated and concluded that STM32F401RE can support RTOS whenever needed.
- As mentioned in the assignment, it was optional but RTOS turned out to be better for implementation.

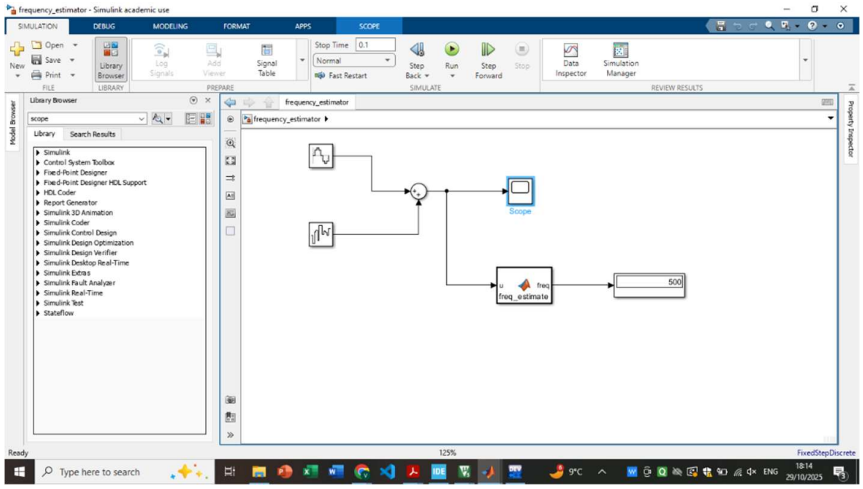
5. Module5: Debug Monitor

- In this module we build a bare metal OS embedded system with a super loop as the background process, and a timer interrupt as the foreground process.

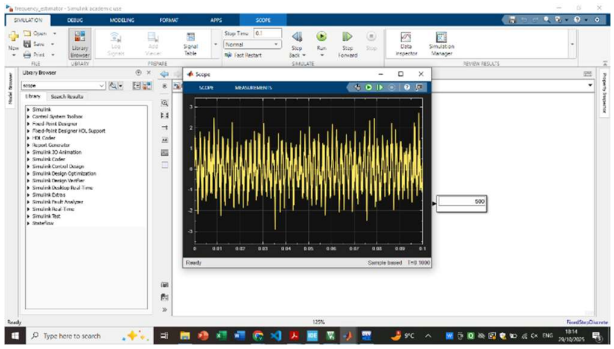
- Initialized the debug monitor on UART based which can be operated on TIM6.
- Upgraded the output to a better display output with following feature.
  Enhanced output window with colour and name with subject code.
  Printed periodic output to UART.
- Dhrystone Benchmark result:
- From this module we concluded that our board STM32F401RE suffices all the performance required for real time embedded systems.

6. Module6: Bare metal Flowmetre simulation.

- Firstly, the Simulink block diagram was prepared:



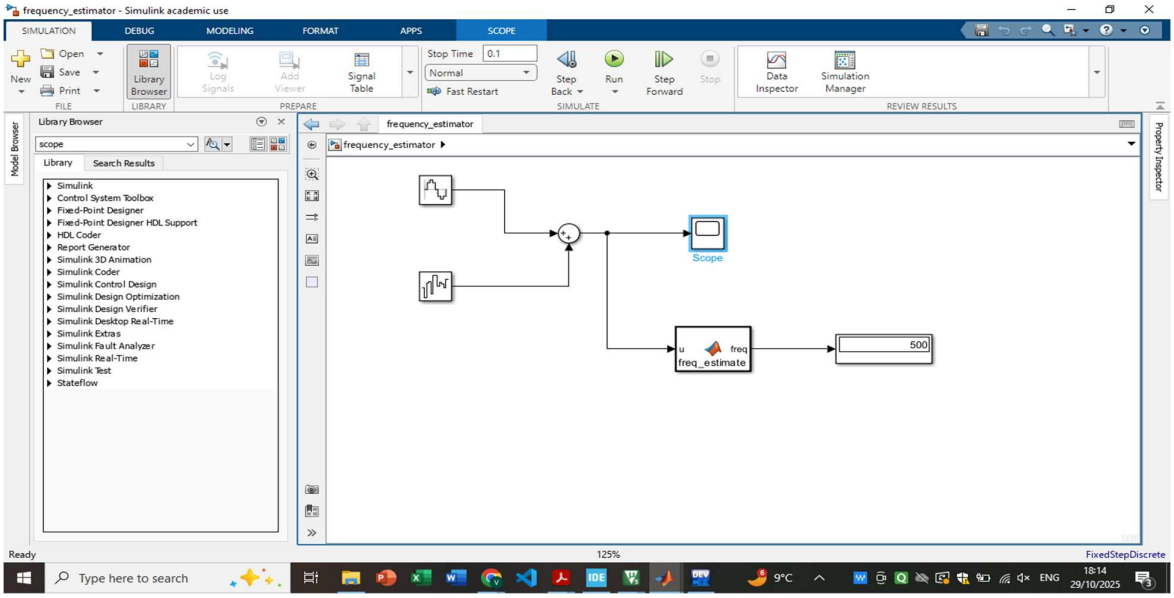
- Here as visible, firstly the sine wave model and white band noise was added with the sum block and its output was taken out on scope as sin.

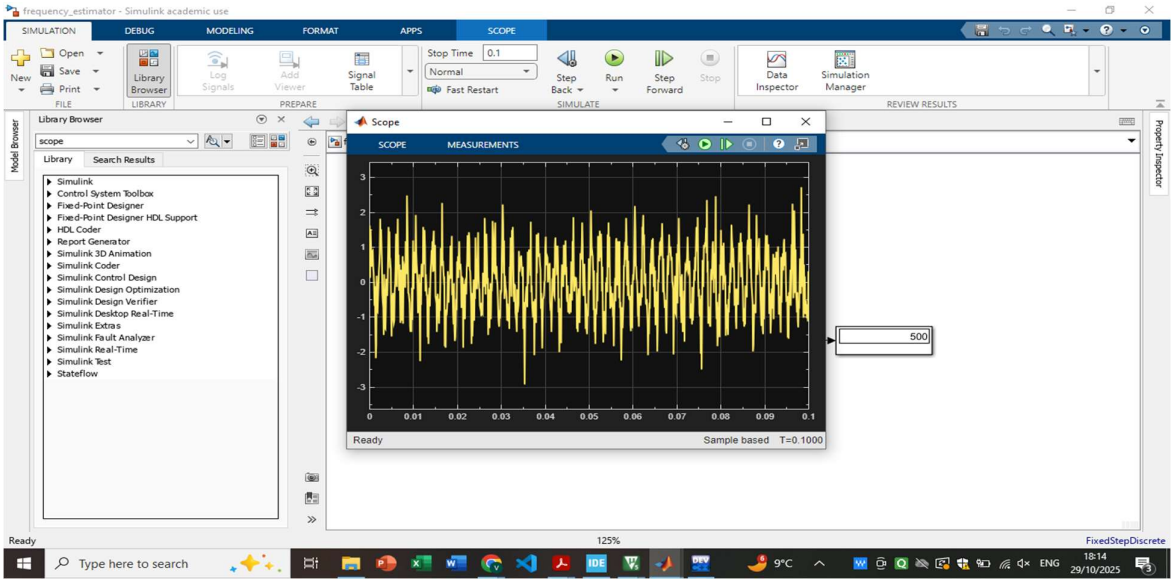


- After this in the MATLAB function model the function code was written in C to get the desired frequency.
- C code was initiated in STM32cube IDE and after that the frequency algorithm was designed for threshold crossing.
- Output on LCD:
  Frequency: 1200Hz
  Temperature: 27.6 degree Celsius
  Flow: 3844.6GPM

5.Simulink Summary:

- A Simulink model was created to validate the vortex flow frequency detection algorithm using a sine wave and white band noise input that emulated ADC sensor data. A MATLAB Function block implemented a threshold-crossing method to estimate frequency over a 0.1 s window at a 10 kHz sample rate.
- Testing at 100 Hz, 500 Hz, and 1000 Hz showed frequency estimates within ±1 % accuracy, confirming the reliability of the algorithm and the chosen parameters (±0.7 thresholds, 1000-sample averaging) for embedded implementation.





6. Hardware Overview:

1. Performance of CPU:

- The processor was made to run at 84MHz by following process mentioned above sufficing the real time vortex flowmeter computation requirements.
- Benchmarking

2. Systick Handler and Interrupt Precision:

- In this aspect the timer was configured for regular sampling and flow rate computation from the data input.
- CPU time and utilization showcases the exceptionally light weight and efficient scheduling mechanism.

3. SPI,UART,GPIO (peripherals)

- SPI was used to get the display on the LCD here backlight was also used.
- UART was used for the getting the serial output on the putty can be used for debugging.
- GPIO was used to take the input from buttons LEDs and speakers to get the output.

7.Conclusion:

From performing all the modules from 1 to 6 it is concluded that the STM32F401RE has proven to be a highly capable and reliable platform for implementing a real-time vortex flowmeter suitable for industrial applications

8.Bill of Materials (BOM)

Sr No.	Components	Name	Quantity	Cost	Total Cost(USD)
1.	MCU	STM32F401RE	1	~$12.00	$12
2.	SPEAKER	TDK PS 1440P0	1	~$0.61	$0.61
3.	TEMPRATURE SENSOR	DS1631	1	~$7.00	$7.00
4.	LCD DISPLAY	NHD-0216HZ-FSW-FBW-33V3C	1	~$17.00	$17.00
5.	POTENTIOMETRE	3386P-1-103TLF	1	~$2.20	$2.20
6.	PUSH BUTTONS	SPST SWITCHES	3	~0.40	$1.20
7.	SHIFT REGISTOR	74HC595	1	~$0.95	$0.95
8.	LEDS	R,G,B	3	~$0.05	$0.15
9.	JUMPER WIRES	M T M, F T M, F T F	~7 EACH	~7.00	$7.00
10.	SOLDERING COST	SOLDERING WIRES, COMPONETNS, HEADERS ETC.	1	~20	$20
TOTAL					$69<200

9. TECHNICAL REPORT DETAILS OF DELIVERABLES:

- Technical Report
- Summary Report
- Hardware Summary
- Simulink Summary
- Bill of Materials
- Software Summary to be submitted in modules sections along with answers and zip file.

10. Reference:

- STM32F401RE Datasheet and Reference Manual (STMicroelectronics)
- DS1631 Temperature Sensor Datasheet
- 74HC595 Shift Register Datasheet
- LCD: NHD-0216HZ-FSW-FBW-3V3C Datasheet
- ECEN5803Project1Guide
- RequestForServicesP1
- ECEN5803Data4.csv / ECEN5803Data4.txt
- Proj1-6 Modulesnotes.
- Saw the lecture videos of demonstrating last year reports to get reference for technical report.