Project Proposal: STM32-Based Real-Time Energy Monitoring System (EE304)

Course: Embedded Systems (EE304) By Mehmet Bozdal
Duration: 5 Weeks
Development Environment: Keil MDK-ARM (Assembly Language)
Team members: Mohammad Sayeduzzaman Ohe & Abdullah Rihawi

1. Introduction & Motivation

The growing global demand for electricity necessitates efficient control and real-time monitoring of energy consumption. Many residential and laboratory settings lack immediate feedback systems to visualize power usage patterns, making it challenging to identify and correct inefficient practices.

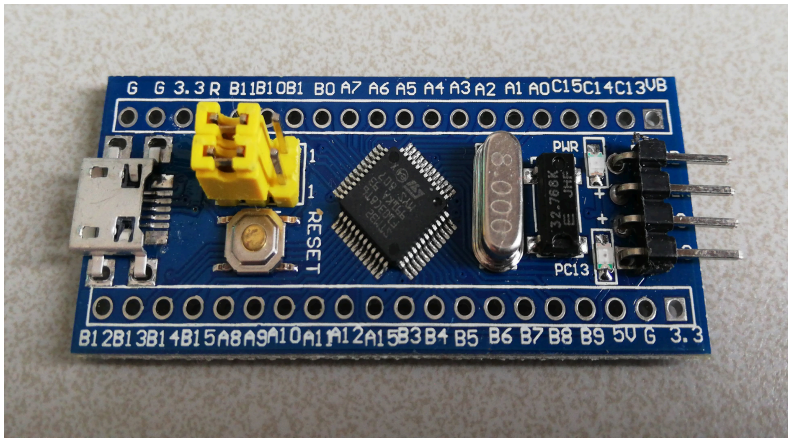
This project proposes the design and construction of a real-time energy monitoring system utilizing the **STM32 (e.g., STM32F103 Series) microcontroller**. The system will accurately measure and calculate key electrical parameters—voltage, current, power, and accumulated energy consumption—and display the results instantaneously on a 16x2 LCD. Crucially, all system drivers, peripheral configuration, and mathematical calculations will be implemented directly in **ARM Assembly Language**. This approach ensures maximum code efficiency and demonstrates a deep understanding of the Cortex-M architecture. Furthermore, a PWM-controlled LED will provide a dynamic, visual signal corresponding to the current power consumption level, enhancing the system's interactivity and awareness function.

2. Project Objectives & Requirement Compliance

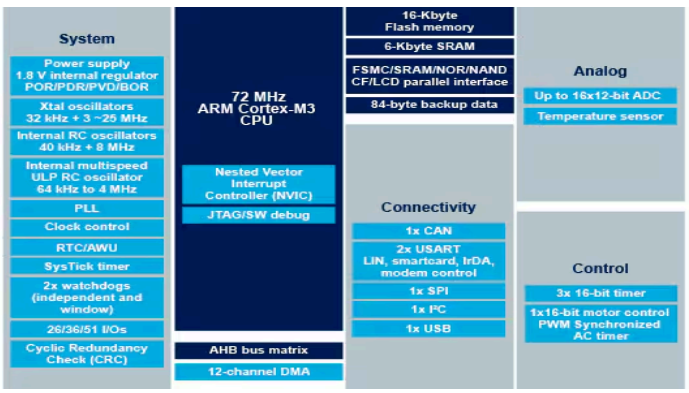
The key aims of this 5-week project are to:

- Develop a robust **STM32-based** real-time energy monitoring appliance addressing a practical application.
- Accurately measure and identify RMS voltage, RMS current, real power (P), and energy consumption (kWh) parameters.
- **Implement all core logic, drivers, and mathematical algorithms using ARM Assembly Language** for optimal performance and register-level control, addressing the high-difficulty challenge.
- Utilize the STM32's **high-resolution internal ADC** for precision signal acquisition.
- Implement I^2C communication, PWM control, and interrupt handlers entirely in Assembly.

- Employ an **interrupt-based control structure** (Timer and External Interrupts) to ensure system responsiveness.



STM32F103



Circuit Diagram

3. Core Technical Requirements Check

The proposed system design explicitly meets all technical requirements of the EE304 project:

Requirement	Component / Feature	Compliance
Communication Protocol	I^2C Communication	Implemented in Assembly for the 16x2 LCD.
Analog/Digital Conversion	ADC Module Utilization	Used to digitize signals from the ZMPT101B and ACS712 sensors.
PWM Signal Generation	Timer PWM Output	Used to dynamically modulate the brightness of the Visual Feedback LED.
Input Device	Push Button (External	Used by the user to reset

	Interrupt)	the accumulated Energy (kWh) value.
Output Device	16x2 LCD Display & PWM LED	Provides real-time numerical and visual feedback.
Power Consideration	Stable Power Design	The final design will include stable power regulation circuitry (LDO/switching regulator) for the STM32 and sensors.
Software Design	Interrupt-Based Control	Timer Interrupt for high-speed sampling; External Interrupt for user input handling.

4. Hardware Components

Component	Function
Microcontroller	**STM32 (e.g., STM32F103 Series)**: Central unit for data acquisition, computation, and control logic.
Voltage Sensor	ZMPT101B: Measures the AC mains voltage and conditions the signal for the ADC.
Current Sensor	ACS712: Measures the current flow through the load via the Hall effect.
Input Device	Push Button: Triggers an External Interrupt to reset accumulated energy (kWh).
Output Device 1	16x2 LCD (I^2C interface): Displays real-time parameters (V, I, P, E).
Output Device 2	PWM-Controlled LED: Provides a dynamic, visual representation of current power consumption.

5. Software & Tools

Tool/Technology	Description
IDE/Framework	**Keil MDK-ARM (Microcontroller Development Kit)**: Used for writing, compiling, debugging, and flashing the embedded code.
Coding Language	**ARM Assembly Language**: Low-level implementation of all drivers, peripherals, and computation logic. This is critical for meeting performance and academic objectives.
Code/Libraries	Minimal use of high-level libraries; reliance on direct register manipulation based on the STM32 Reference Manual.
Version Control	GitHub: progress tracking, documentation, and source code management.

6. Project Progression Outline (5 Weeks)

This plan allocates significant time to the complex tasks of register configuration and mathematical implementation in Assembly.

Week 1: Toolchain Setup and Low-Level Configuration

Milestone	Task Description	Deliverable
1.1 Toolchain Setup & Project Structure	Install and configure Keil MDK-ARM. Create the initial project structure, including assembly startup file and memory map definition.	Keil Project ready for Assembly development.
1.2 Clock & GPIO Initialization (Assembly)	Configure the STM32 system clock and initialize basic GPIO pins (e.g., for	Simple Assembly code that successfully initializes clock and toggles an LED.

	the status LED) directly by writing to the RCC and GPIO registers in Assembly.	
1.3 Version Control	Initialize the GitHub repository and upload initial Assembly files.	GitHub repository initialized with Week 1 code.
Focus	Establish the core software framework and mastery of register access via Assembly.	

Week 2: ADC and Interrupt Service Routine (Assembly)

Milestone	Task Description	Deliverable
2.1 ADC Configuration (Assembly)	Configure the ADC registers (sampling rate, channels, conversion mode) in Assembly to prepare for simultaneous acquisition of Voltage and Current sensor inputs.	Assembly code successfully samples a known DC input and stores the raw value in memory.
2.2 Timer & Interrupt Handler (Assembly)	Configure a Timer peripheral to generate periodic interrupts (e.g., 100µs for fast sampling). Write the complete Interrupt Service Routine (ISR) in Assembly, ensuring correct register saving/restoring (Context Switching).	ADC sampling triggered accurately by a Timer Interrupt, confirming stable ISR execution.
2.3 Basic Communication (GPIO)	Implement a basic, bit-banged delay function and low-level GPIO control routines for eventual I^2C/PWM use.	Stable Assembly-implemented delay functions.

Focus	Mastering interrupt handling and high-speed data acquisition in Assembly.	

Week 3: Core Calculations and Math Routines (Assembly)

Milestone	Task Description	Deliverable
3.1 Digital Filtering & RMS Math Routines	Write Assembly routines for floating-point/fixed-point arithmetic needed for RMS calculation (V_{RMS} and I_{RMS}).	Assembly function that takes an array of raw ADC samples and returns a calculated RMS value.
3.2 I^2C Master Initialization (Assembly)	Configure I^2C peripheral registers in Assembly to initialize the STM32 as a Master. Implement Assembly routines for START, STOP, ACK, and NACK generation.	I^2C Bus is initialized and ready for communication.
3.3 Sensor Calibration	Initial calibration of sensor constants to refine the Assembly RMS calculation routines.	Calibrated Voltage and Current Assembly functions.
Focus	Implementing complex math and low-level communication drivers entirely in Assembly.	

Week 4: Power Integration, Input, and Dynamic Control (Assembly)

Milestone	Task Description	Deliverable
4.1 Real Power Calculation (Assembly)	Implement instantaneous power calculation ($P_{inst} = V \times I$) and subsequent	Assembly routine calculates and stores P (Watts).

	integration/averaging in Assembly to determine **Real Power (P)**.	
4.2 LCD Display Driver (Assembly)	Integrate the I^2C master routines with the 16x2 LCD communication sequence (sending commands/data), all in Assembly. Display V, I, and P results.	LCD displays accurate real-time V, I, and P values.
4.3 PWM Control & Input (Assembly)	Configure Timer peripheral registers for PWM output (for LED). Configure an External Interrupt pin for the **Push Button Input** and write the corresponding Assembly ISR to reset kWh.	LED dynamically changes brightness; Push Button input successfully resets a counter via an interrupt.
Focus	Full system integration, dynamic control mechanisms, and implementing the required Input Device (Button).	

Week 5: Final Integration, Optimization, and Documentation

Milestone	Task Description	Deliverable
5.1 Energy (kWh) Calculation & Optimization	Implement the long-term energy accumulation (kWh) in Assembly. Refactor and optimize all critical math and I/O routines for execution speed and stability.	LCD displays Energy in kWh. Code is stable and highly efficient.
5.2 Final Testing & Quality Check	Run the system for an extended period to check for drift, register overflow, and calculation consistency. Verify all	Comprehensive test log confirming system stability and accuracy.

	project objectives are met.	
5.3 Final Report & Presentation Preparation	Complete the final project report, emphasizing the complexity and efficiency gains achieved by using Assembly. Prepare the demonstration.	Final Report Document and Presentation Slides.
5.4 Project Submission	Complete the final GitHub commit, documentation, and deliver the final prototype and presentation.	Project delivered successfully.
Focus	Professional documentation and final optimization of the Assembly code.	

7. Expected Outcomes & Evaluation Alignment

Rubric Category	Expected Deliverable	Description
System Design & Functionality	Fully working **STM32-based** system	The system performs as intended, utilizing **ADC, PWM, I^2C, and Interrupts** correctly for stable, accurate energy monitoring.
Engineering Implementation	Reliable, optimized **Assembly Code**	Demonstrates rigorous register-level design, robust **Assembly Language** implementation, highly efficient code, and professional interrupt handling. **(High Score Expectation)**
Creativity & Purpose	Real-time energy monitor + **Assembly**	Addresses a practical application (energy

	Implementation	awareness) and exhibits high engineering creativity through the mandatory use of **ARM Assembly Language**. **(High Score Expectation)**
Communication & Documentation	Organized GitHub repo + Detailed Report	Exhibits well-organized documentation, circuit schematics, clear version control discipline, and technical explanations of design in the report.
Presentation & Demonstration	Functional prototype & confident explanation	Effectively conveys project outcomes and demonstrates a strong understanding of the underlying engineering principles, particularly the low-level Assembly implementation.

8. Conclusion

The **STM32-Based** Real-Time Energy Monitoring System is a highly rigorous project that satisfies all core requirements while maximizing technical rigor through its implementation in **ARM Assembly Language**. The comprehensive 5-week plan provides a clear roadmap to successfully deliver a functional and efficient embedded system.