# **STM32-Based Weight Measurement System with UART Communication and LED Indicators**

**INTRODUCTION:**

This project demonstrates the implementation of a weight measurement system using an STM32F4 microcontroller. The system is designed to read data from a Force-Sensitive Resistor (FSR), convert the analog readings into a digital weight value using the Analog-to-Digital Converter (ADC), and then transmit the weight information via UART (Universal Asynchronous Receiver-Transmitter) communication. Additionally, the system uses two LED indicators to display stock status based on predefined weight thresholds.

The key objective of this project is to create an efficient and reliable weight-sensing system that can monitor inventory levels in real-time. For instance, it can be used to detect when a product is low in stock and trigger an alert when the inventory is depleted. The system features UART communication for remote monitoring, making it suitable for integration with a Bluetooth module or other serial communication devices.

Key features include:

* Real-time weight measurement using an FSR sensor.
* ADC for converting analog sensor output to a digital value.
* Calibration to convert ADC values into accurate weight readings.
* UART communication for transmitting data to a serial monitor or Bluetooth module.
* LED indicators to signal stock availability and alerts.

This system can be applied in inventory management, automated weight-based systems, or as a digital weighing scale for various use cases.

**SYSTEM COMPONENETS:**

* STM32F4 Microcontroller: The central processing unit responsible for reading sensor data, performing calculations, and controlling peripherals.
* Force-Sensitive Resistor (FSR): A sensor used to measure the applied force, which is then converted into a weight value.
* ADC (Analog-to-Digital Converter): Converts the analog voltage from the FSR to a digital value for processing by the microcontroller.
* UART Module: Facilitates serial communication to transmit weight data to a serial monitor or Bluetooth device.
* Bluetooth Module (Optional): Allows for wireless transmission of data via UART.
* LEDs:
  + Green LED (connected to PA5): Indicates sufficient stock or weight above the threshold.
  + Red LED (connected to PA6): Indicates low stock or weight below the threshold.
* Resistors: Used for current limiting with LEDs.
* Power Supply (5V or 3.3V): Provides the necessary operating voltage for the microcontroller and peripherals.
* USB to Serial Converter (for debugging): Used to connect the microcontroller’s UART d
* SysTick Timer: Used to implement software delays.
* STM32 Standard Peripheral Library (CMSIS): Provides low-level access to microcontroller features.

**SYSTEM DESIGN:**

1. ADC (Analog-to-Digital Converter) Subsystem

Purpose:

* + Converts the analog signal from the FSR to a digital value that the microcontroller can process.

Design:

* + The ADC is configured to use channel 1 (connected to pin PA1).
  + The function ADC\_Init() sets up the ADC for single conversion mode.
  + Flow:
    - ADC\_Read() function triggers the conversion.
    - Waits for the End of Conversion (EOC) flag.
    - Reads the digital value from the ADC data register

1. Weight Calculation Module:

Purpose:

* + Converts the raw ADC value into a meaningful weight measurement using a linear calibration formula.

Design:

* + Calibration Equation: weight = (M \* adc\_value) + B, where:
    - M: Slope determined from calibration.
    - B: Offset determined from calibration.
  + The function CalculateWeight() implements this conversion.

1. **Stock Monitoring and LED Indication:**

**Purpose**:

* + Provides visual alerts based on the weight measured, indicating stock availability.

**Design**:

* + Two GPIO pins control LEDs:
    - **Green LED (PA5)** for "In Stock" status.
    - **Red LED (PA6)** for "Low Stock" or "Out of Stock" status.
  + The main loop checks the measured weight:
    - If weight < 1 kg: Red LED is activated, and a low stock message is sent.
    - If weight >= 1 kg: Green LED is activated, and an in-stock message is sent.

1. UART Communication Subsystem:

Purpose:

* + Facilitates serial communication with external devices, like a Bluetooth module, for remote monitoring.

Design:

* + UART\_Init() configures UART for communication at a baud rate of 9600.
  + Uses PA9 for TX (Transmit) and PA10 for RX (Receive).
  + Enables UART interrupt to handle incoming data asynchronously.
  + Functions:
    - UART\_ SendChar() and UART\_Send\_String(): Send characters and strings over UART.
    - USART1\_IRQHandler(): Interrupt handler to process received data (e.g., 'O' command to check stock).

CODE FLOW:

* **System Initialization:**
* ADC, UART, and GPIO are initialized.
* UART interrupts are enabled to handle incoming commands.
* **UART Initialization (UART\_Init())**
* The UART interface is set up for serial communication with external devices (like a Bluetooth module) at a baud rate of 9600.
* Key steps:
  1. Enable clocks for USART1 and GPIOA.
  2. Set PA9 (TX) and PA10 (RX) to alternate function mode for UART communication.
  3. Configure the baud rate, enable the transmitter, receiver, and USART.
  4. Enable the RXNE (Receive Not Empty) interrupt for handling incoming UART data.
* **GPIO Configuration**
* Additional GPIO pins are configured for LED indicators:
  + PA5: Green LED (indicates "In Stock").
  + PA6: Red LED (indicates "Low Stock" or "Out of Stock").
* **Main Execution Loop (main())**

The core functionality of the system is executed within an infinite loop in the main() function:

**A. Reading FSR Sensor Data:**

1. Read ADC Value:
   * The function ADC\_Read() is called to start the ADC conversion.
   * The converted digital value (0-4095) is read from the ADC data register.
2. Calculate Weight:
   * The raw ADC value is converted into a weight measurement using the formula:
   * The calibration values M (slope) and B (offset) are used to ensure accurate weight calculation.
3. Weight Validation:
   * If the calculated weight is less than 0, it is set to 0 to avoid negative readings.
4. Convert Weight to String:
   * The Float\_To\_String() function converts the floating-point weight value to a string format with 3 decimal places.

**B. Stock Monitoring and LED Alerts:**

1. Check Stock Level:
   * If the measured weight is less than 1 kg:
     + Red LED (PA6) is turned on to indicate low stock.
     + Green LED (PA5) is turned off.
     + If the weight is 0, a message "Out of stock" is sent over UART.
     + If the weight is less than 1 kg but not zero, it sends "Only few in stock".
   * If the weight is 1 kg or more:
     + Green LED (PA5) is turned on to indicate sufficient stock.
     + Red LED (PA6) is turned off.
     + A message "In Stock" is sent over UART.
2. Send Data Over UART:
   * The current weight and stock status are transmitted via Bluetooth using the UART\_Send\_String() function.
3. Delay for Stabilization:
   * The system waits for 500 ms using the delay\_ms() function before the next reading to avoid rapid sensor polling.

**C. Newline Characters for UART Readability:**

* Each transmitted message is followed by \r\n to ensure clear line breaks in the Bluetooth terminal.

**UART Interrupt Handling**:

When a character is received over UART:

* The interrupt triggers, and the received character is read from USART1->DR.
* If the character is 'O', the system calculates the amount of weight needed to reach a target of **3 kg**.
* This difference is then converted to a string and sent over UART, followed by the message "To be ordered".

**PROBLEMS FACED AND SOLUTION PROPOSED:**

**1.Inaccurate Weight Measurements:**

The Force Sensitive Resistor (FSR) readings were inconsistent, leading to inaccurate weight calculations. The output varied due to factors like sensor noise and non-linear response of the FSR.

* **Solution:**

Calibration: Introduced calibration constants (M for slope and B for offset) to adjust the raw ADC values into more accurate weight measurements.

Formula Adjustment: Used the formula convert the ADC reading to a calibrated weight. Calibration was repeated multiple times to refine accuracy.

Smoothing Data: Implemented a delay between sensor readings to allow the sensor output to stabilize, reducing noise.

**2. Incorrect UART Communication:**

Data sent over the UART interface was often garbled or incomplete when displayed on the Bluetooth terminal. This issue affected both outgoing weight data and incoming commands.

* **Solution:**

Interrupt Handling: Enabled the RXNE (Receive Not Empty) interrupt for UART to efficiently process incoming data. This reduced data loss and improved command handling.

Baud Rate Configuration: Ensured the correct configuration of the UART baud rate using the formula

**3. LED Indicator Logic Errors:**

The LED indicators were not behaving as expected. For instance, both LEDs were turning on simultaneously or not at all, leading to confusion in stock status indication.

* **Solution:**

GPIO Configuration Review: Ensured proper configuration of GPIO pins for LED control (PA5 and PA6). Specifically corrected pin mode setup to output mode.

**4. Handling Negative Weight Values:**

In some cases, the calculated weight value was negative due to sensor noise or calibration errors.

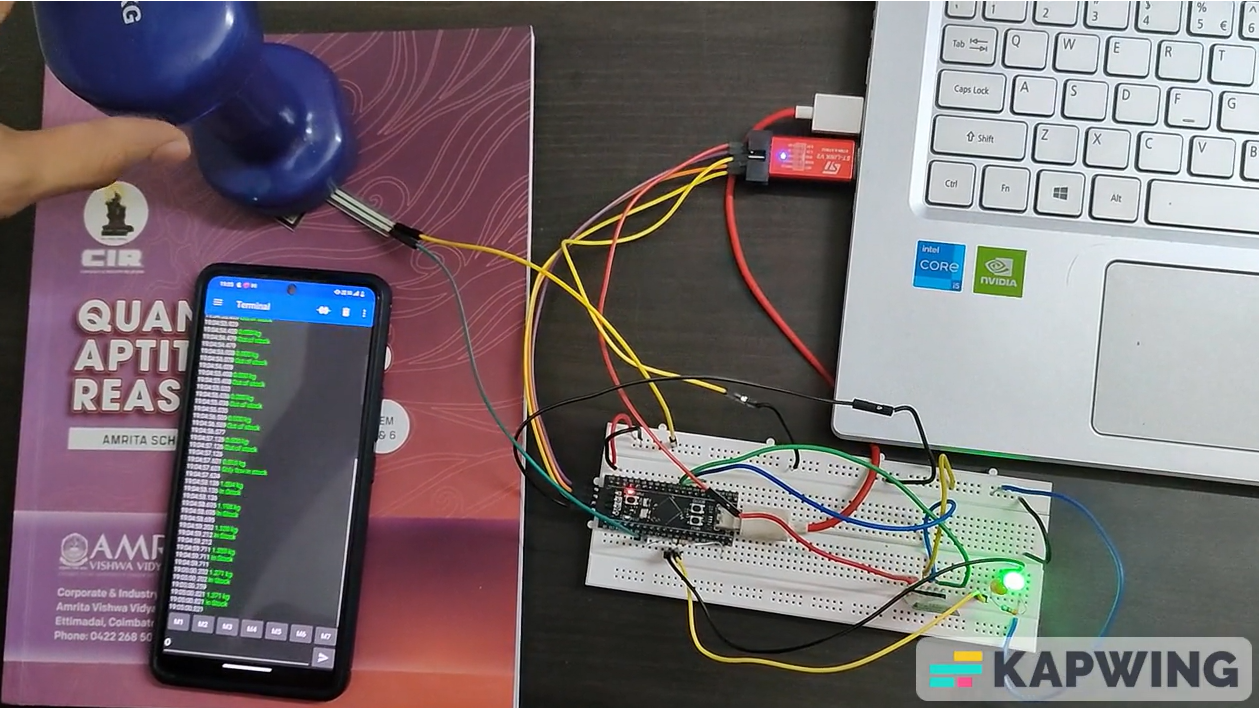
* **Solution:**

Weight Validation: Added a condition to set any negative weight to 0.0 to avoid misleading stock indications.

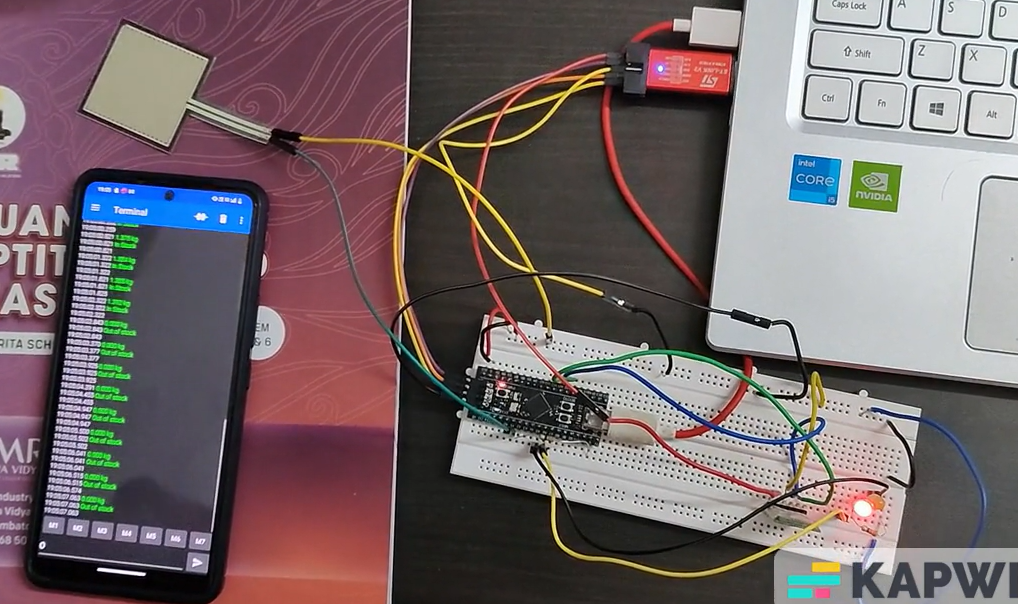
Calibration Adjustment: Fine-tuned the calibration constants to minimize negative readings.

**IMAGE DEMO:**

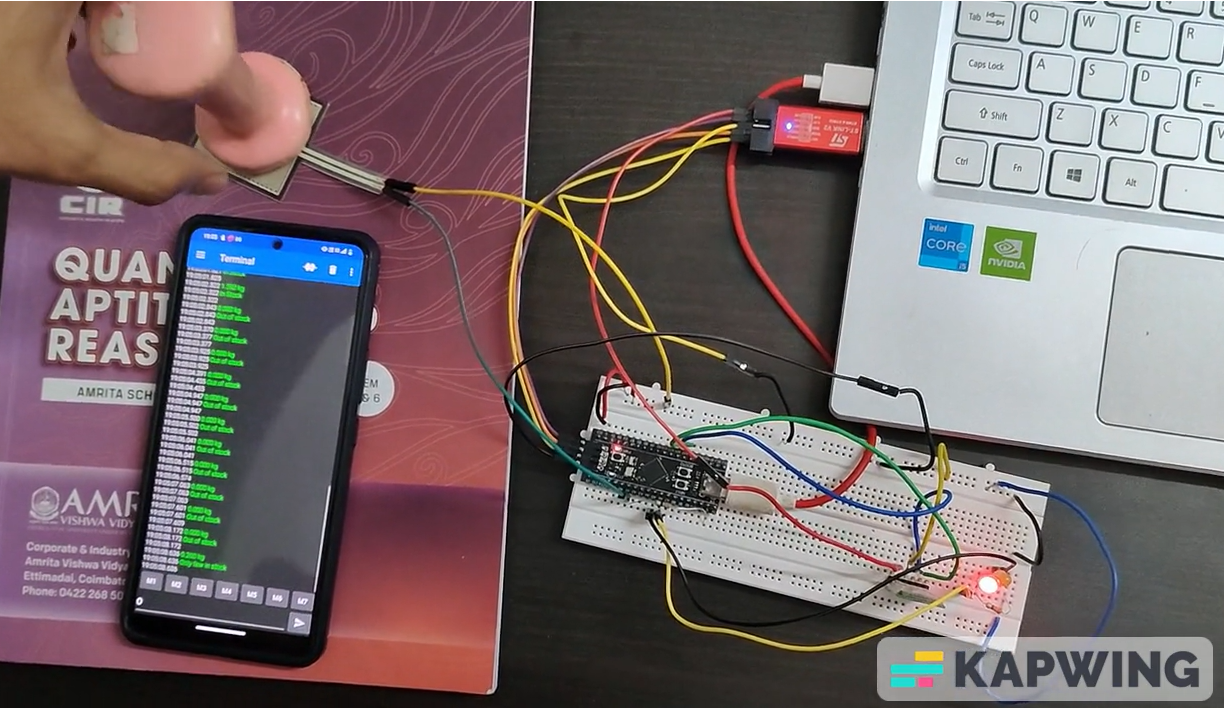
Weight above threshold value: Green

****

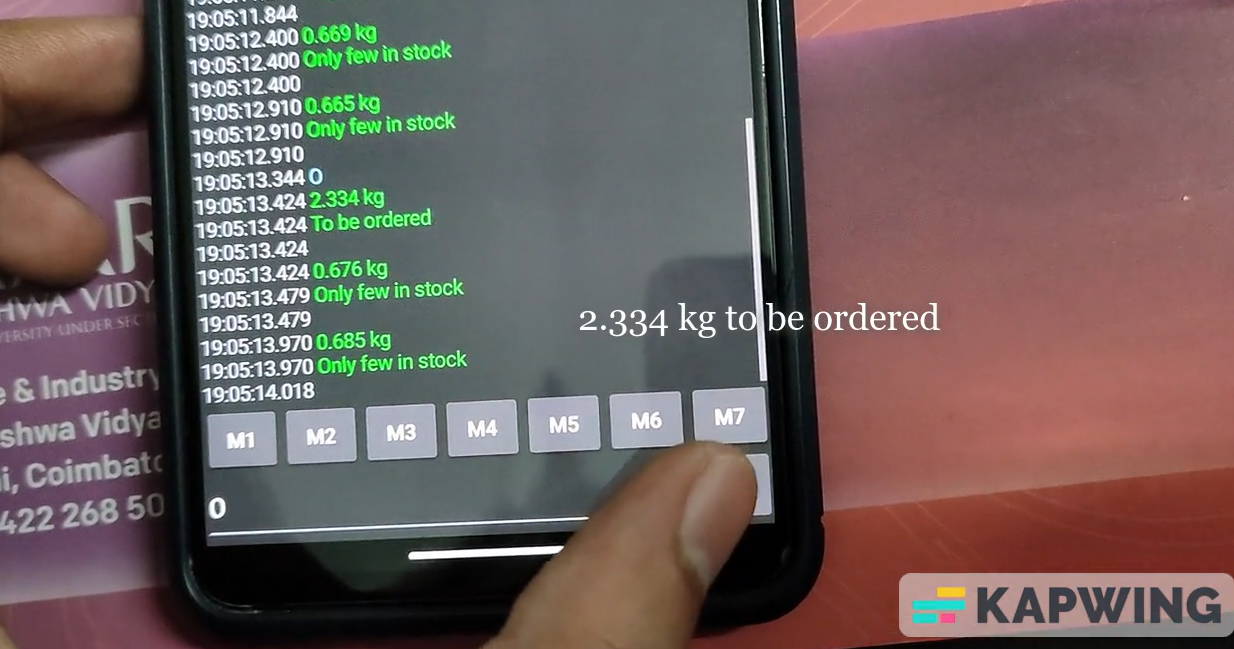
Weight below threshold value: Red

****

Weight below threshold value: Red

****

**O command:**

****

**DONE BY:**

CB.EN. U4CSE22506- Ayush Kumar

CB.EN. U4CSE22519-Kamala Krishnan U

CB.EN. U4CSE22550-Sudhakar K S

CB.EN. U4CSE22551-Suren Adhitya

CB.EN. U4CSE22563-Jaideep p