Smart Logging System

DESCRIPTION

(Prototype)

Our setup uses a **load sensor** to measure weight by converting the physical force exerted by the load into an electrical signal. This signal, which is typically very weak, is amplified by the **HX711** load cell amplifier. The HX711 module provides precise Analog-to-Digital conversion, making it ideal for reading small variations in the load cell's output, which translates to accurate weight measurement.

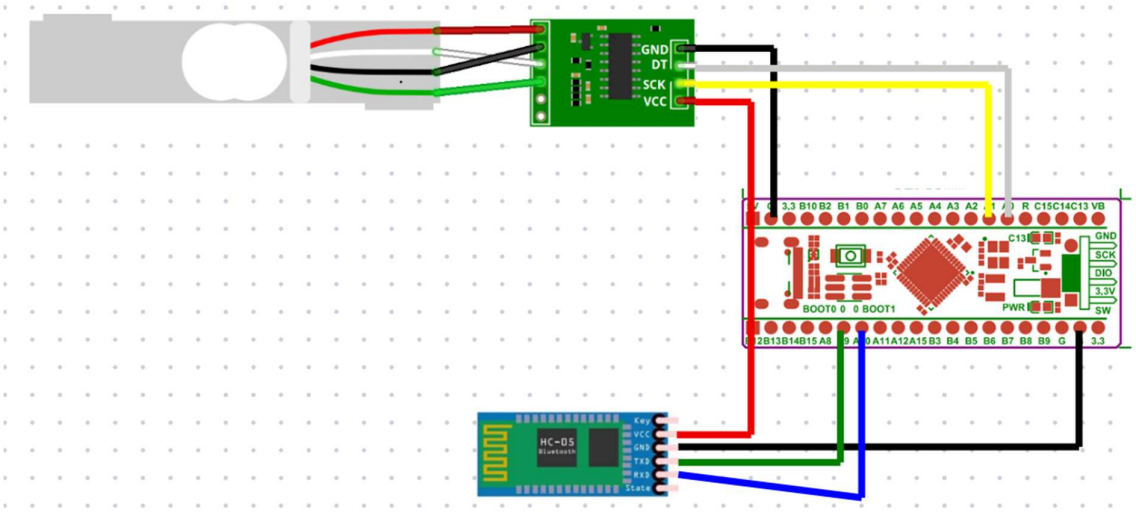
The **STM32F401CCU6** microcontroller processes the data from the HX711, allowing you to control the weight measurement system and handle the data processing. The weight readings are then sent wirelessly via the **HC-07 Bluetooth module**. This module provides reliable, short-range communication, enabling the weight data to be transmitted to a smartphone or computer, which can be monitored or recorded in real time.

PARTS REQUIRED
Here's a list of the parts required for this project:

- **STM32F401CCU6**
- **Load Cell with HX711 Amplifier**
- **Breadboard**
- **Jumper Wires**

Load Cell	HX711	HX711	STM32F401CCU6
Red (E+)	E+	GND	GND
Black (E-)	E-	DT	A0
White (A-)	A-	SCK	A1
Green (A+)	A+	VCC	5V

DIAGRAM



CODE

```
#include "stm32f4xx.h"
#include <stdio.h>

// Define pins for HX711
#define HX711_DATA_PIN    (1 << 1)  // PA1 (GPIOA Pin 1)
#define HX711_CLOCK_PIN   (1 << 0)  // PA0 (GPIOA Pin 0)

// Function prototypes
void HX711_Init(void);
long HX711_Read(void);
void USART1_Init(void);
void USART1_Write(char ch);
void USART1_WriteString(const char* str);
void printWeight(float weight);
float prevData = {0, 0, 0, 0, 0};
int i = 0;

void USART1_Init(void) {
    RCC->AHB1ENR |= (1U << 0);  // Enable GPIOA clock
    RCC->APB2ENR |= (1U << 4);  // Enable USART1 clock

    // Configure PA9 (TX) and PA10 (RX) for USART1 in alternate function mode
    GPIOA->MODER &= ~(3U << (9 * 2) | 3U << (10 * 2));
    GPIOA->MODER |= (2U << (9 * 2)) | (2U << (10 * 2)); // Set PA9, PA10 to AF mode
    GPIOA->AFR[1] |= (7U << (1 * 4)) | (7U << (2 * 4)); // Set AF7 for PA9 and PA10 (USART1)
```

```
    // Configure USART1 for 9600 baud rate
    USART1->BRR = 0x683; // Assuming 16 MHz clock, BRR for 9600 baud
    USART1->CR1 = USART_CR1_TE | USART_CR1_RE | USART_CR1_UE; // Enable TX, RX, and USART1
}

void USART1_Write(char ch) {
    while (!(USART1->SR & USART_SR_TXE)); // Wait until TX buffer is empty
    USART1->DR = ch;                      // Send character
    while (!(USART1->SR & USART_SR_TC));   // Wait until transmission is complete
}

void USART1_WriteString(const char* str) {
    while (*str) {                    // Loop until the null terminator
        USART1_Write(*str++);           // Send each character
    }
}

void printWeight(float weight) {
    char message[50];
    sprintf(message, "The weight is %.2f grams\n", weight); // Create message with weight
    USART1_WriteString(message);
}

void HX711_Init(void) {
    RCC->AHB1ENR |= RCC_AHB1ENR_GPIOAEN; // Enable GPIOA clock

    // Configure PA1 (Data pin) as input
    GPIOA->MODER &= ~HX711_DATA_PIN; // Clear mode bits for PA1
    GPIOA->PUPDR &= ~HX711_DATA_PIN; // No pull-up, pull-down

    // Configure PA0 (Clock pin) as output
    GPIOA->MODER |= (1 << (0 * 2));   // Set PA0 to output mode
    GPIOA->PUPDR &= ~HX711_CLOCK_PIN; // No pull-up, pull-down
}

long HX711_Read(void) {
    long count = 0;

    // Wait for the HX711 to be ready
    while ((GPIOA->IDR & HX711_DATA_PIN) != 0);

    // Read 24 bits from the HX711
    for (int i = 0; i < 24; i++) {
        GPIOA->ODR |= HX711_CLOCK_PIN;  // Set clock high
        count = count << 1;           // Shift left
        GPIOA->ODR &= ~HX711_CLOCK_PIN; // Set clock low
```

```
    // Read data
    if (GPIOA->IDR & HX711_DATA_PIN) {
      count++;                                  // If data bit is high, increment count
    }
  }

  GPIOA->ODR |= HX711_CLOCK_PIN;               // Set clock high for gain setting
  count ^= 0x800000;                           // Correct the count for two's complement
  GPIOA->ODR &= ~HX711_CLOCK_PIN;              // Set clock low

  return count;
}

int main(void) {
  SystemInit();
  HX711_Init();
  USART1_Init();

  while (1) {
    long rawData = HX711_Read();
    float weight = ((float)rawData / 1000.0) - 9018; // Adjust divisor based on calibration
        for (volatile int i = 0; i < 1000000; i++); // Simple delay
    printWeight(weight);for (volatile int i = 0; i < 100000; i++); // Simple delay
    for (volatile int i = 0; i < 100000; i++); // Simple delay
  }
}
```

OUTPUT

```
22:16:50.501 Connecting to HC-06 ...
22:16:54.358 Connected
22:17:22.131 Disconnected
22:17:22.804 Connecting to HC-06 ...
22:17:25.567 Connected
22:17:29.240 ?The weight is 1199.05 grams
22:17:29.334 The weight is 1199.04 grams
22:17:29.498 The weight is 1198.92 grams
22:17:29.633 The weight is 1198.94 grams
22:17:29.737 The weight is 1198.99 grams
22:17:29.896 The weight is 1198.96 grams
22:17:30.033 The weight is 1198.96 grams
22:17:30.139 The weight is 1198.97 grams
22:17:30.311 The weight is 1198.97 grams
22:17:30.442 The weight is 1198.90 grams
22:17:30.546 The weight is 1198.91 grams
22:17:30.712 The weight is 1199.00 grams
```