#include "main.h"

#include <stdio.h>

#include <string.h>

// Function Prototypes

void SystemClock\_Config(void);

void UART\_Transmit(char \*msg);

void Read\_DHT22(void);

void Read\_pH\_Sensor(void);

void Send\_Alert(char \*alert);

// Peripheral Handlers

UART\_HandleTypeDef huart1; // RFID & 433MHz Transmitter

UART\_HandleTypeDef huart2; // Barcode Scanner

ADC\_HandleTypeDef hadc1; // pH Sensor

// DHT22 Data Pin

#define DHT22\_PORT GPIOB

#define DHT22\_PIN GPIO\_PIN\_5

// Buzzer & LED

#define BUZZER\_PORT GPIOB

#define BUZZER\_PIN GPIO\_PIN\_12

#define LED\_PORT GPIOB

#define LED\_PIN GPIO\_PIN\_13

void HAL\_UART\_MspInit(UART\_HandleTypeDef \*huart) {

// Configure UART GPIOs here

}

void HAL\_ADC\_MspInit(ADC\_HandleTypeDef \*hadc) {

// Configure ADC GPIOs here

}

int main(void) {

HAL\_Init();

SystemClock\_Config();

// Initialize UART

huart1.Instance = USART1;

huart1.Init.BaudRate = 9600;

huart1.Init.WordLength = UART\_WORDLENGTH\_8B;

huart1.Init.StopBits = UART\_STOPBITS\_1;

huart1.Init.Parity = UART\_PARITY\_NONE;

huart1.Init.Mode = UART\_MODE\_TX\_RX;

huart1.Init.HwFlowCtl = UART\_HWCONTROL\_NONE;

HAL\_UART\_Init(&huart1);

// Initialize ADC for pH Sensor

hadc1.Instance = ADC1;

hadc1.Init.ScanConvMode = DISABLE;

hadc1.Init.ContinuousConvMode = ENABLE;

HAL\_ADC\_Init(&hadc1);

// Initialize GPIO for Buzzer & LED

\_\_HAL\_RCC\_GPIOB\_CLK\_ENABLE();

GPIO\_InitTypeDef GPIO\_InitStruct = {0};

GPIO\_InitStruct.Pin = BUZZER\_PIN | LED\_PIN;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(BUZZER\_PORT, &GPIO\_InitStruct);

while (1) {

Read\_DHT22(); // Read Temperature & Humidity

Read\_pH\_Sensor(); // Read pH Level

HAL\_Delay(2000);

}

}

// Function to Transmit Data over UART

void UART\_Transmit(char \*msg) {

HAL\_UART\_Transmit(&huart1, (uint8\_t\*)msg, strlen(msg), 100);

}

// Function to Read DHT22 Sensor (Temperature & Humidity)

void Read\_DHT22(void) {

int temperature = 25; // Mock Data, replace with actual reading logic

int humidity = 60; // Mock Data

char buffer[50];

sprintf(buffer, "Temp: %d C, Humidity: %d%%\r\n", temperature, humidity);

UART\_Transmit(buffer);

if (temperature > 40) {

Send\_Alert("High Temp Alert! Heatwave Detected.");

}

}

// Function to Read pH Sensor Value (Analog)

void Read\_pH\_Sensor(void) {

HAL\_ADC\_Start(&hadc1);

HAL\_ADC\_PollForConversion(&hadc1, 100);

uint32\_t adc\_value = HAL\_ADC\_GetValue(&hadc1);

float pH\_value = (adc\_value \* 14.0) / 4095.0; // Convert ADC value to pH Scale (0-14)

char buffer[30];

sprintf(buffer, "pH Level: %.2f\r\n", pH\_value);

UART\_Transmit(buffer);

if (pH\_value < 5.0 || pH\_value > 9.0) {

Send\_Alert("pH Alert! Unusual Water Acidity.");

}

}

// Function to Send Alerts via 433MHz TX

void Send\_Alert(char \*alert) {

UART\_Transmit("ALERT: ");

UART\_Transmit(alert);

UART\_Transmit("\r\n");

// Activate Warning Buzzer & LED

HAL\_GPIO\_WritePin(BUZZER\_PORT, BUZZER\_PIN, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(LED\_PORT, LED\_PIN, GPIO\_PIN\_SET);

HAL\_Delay(3000);

HAL\_GPIO\_WritePin(BUZZER\_PORT, BUZZER\_PIN, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(LED\_PORT, LED\_PIN, GPIO\_PIN\_RESET);

}

// Clock Configuration (Default)

void SystemClock\_Config(void) {

RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};

RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSE;

RCC\_OscInitStruct.HSEState = RCC\_HSE\_ON;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;

RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_HSE;

RCC\_OscInitStruct.PLL.PLLMUL = RCC\_PLL\_MUL9;

HAL\_RCC\_OscConfig(&RCC\_OscInitStruct);

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK | RCC\_CLOCKTYPE\_SYSCLK |

RCC\_CLOCKTYPE\_PCLK1 | RCC\_CLOCKTYPE\_PCLK2;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_PLLCLK;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV2;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;

HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_2);

}