**Data Logger for Solar System using STM32F401RE Microcontroller**

**Project Overview**

Project Duration\*\*: January 2023 to March 2023

Project Supervisor/Advisor\*\*: Prf. Ahmed Reja Alai

**Project Description**

The project aims to develop a data logger system for a solar power system using the STM32F401RE microcontroller. The system will monitor various environmental parameters, including light intensity (using an LDR), temperature (using an LM35 sensor), current (using a Hall Effect sensor), and humidity.

**Objectives**

- Design and build a data logger system for a solar power system.

- Utilize the STM32F401RE microcontroller for data acquisition and storage.

- Implement sensors to measure light intensity (LDR), temperature (LM35), current (Hall Effect sensor), and humidity.

- Develop a user-friendly interface for data visualization and retrieval.

- Ensure real-time data monitoring and logging.

**Components and Sensors**

The project will use the following components and sensors:

- \*\*STM32F401RE Microcontroller\*\*: The main controller for data acquisition and storage.

- \*\*Light Dependent Resistor (LDR)\*\*: To measure light intensity and monitor day-night variations.

- \*\*LM35 Temperature Sensor\*\*: To monitor temperature conditions in the solar system.

- \*\*Hall Effect Sensor\*\*: To measure current in the solar system.

- \*\*Humidity Sensor\*\*: To measure humidity levels in the environment.

**Sensor Interfacing**

**1. LDR Interface:**

- To interface the Light Dependent Resistor (LDR) with the STM32F401RE microcontroller and read its value, you'll need to set up a basic circuit and write some code. Here's a step-by-step guide along with sample code in C using the HAL (Hardware Abstraction Layer) library for STM32.

Hardware Setup

Materials Required:

1. STM32F401RE Nucleo or Discovery board.

2. LDR (Light Dependent Resistor).

3. Resistor (around 10k ohms) for making a voltage divider.

4. Jumper wires.

5. Breadboard (optional).

Circuit Connections:

1. Connect one terminal of the LDR to a 3.3V pin on the STM32 board.

2. Connect the other terminal of the LDR to one end of the resistor.

3. Connect the other end of the resistor to one of the analog input pins (e.g., PA0) on the STM32 board.

4. Connect the ground (GND) pin of the STM32 board to the common junction of the LDR and resistor.

**Code :**

**#include "stm32f4xx\_hal.h"**

**ADC\_HandleTypeDef hadc;**

**void SystemClock\_Config(void);**

**static void MX\_GPIO\_Init(void);**

**static void MX\_ADC1\_Init(void);**

**int main(void) {**

**HAL\_Init();**

**SystemClock\_Config();**

**MX\_GPIO\_Init();**

**MX\_ADC1\_Init();**

**HAL\_ADC\_Start(&hadc);**

**while (1) {**

**HAL\_ADC\_PollForConversion(&hadc, HAL\_MAX\_DELAY);**

**uint16\_t adcValue = HAL\_ADC\_GetValue(&hadc);**

**// You can further process or use the adcValue here.**

**}**

**}**

**void SystemClock\_Config(void) {**

**RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};**

**RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};**

**\_\_HAL\_RCC\_PWR\_CLK\_ENABLE();**

**\_\_HAL\_PWR\_VOLTAGESCALING\_CONFIG(PWR\_REGULATOR\_VOLTAGE\_SCALE1);**

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

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

**RCC\_OscInitStruct.HSEPredivValue = RCC\_CKD\_DIV1;**

**RCC\_OscInitStruct.PLL.PLLM = 25;**

**RCC\_OscInitStruct.PLL.PLLN = 360;**

**RCC\_OscInitStruct.PLL.PLLP = RCC\_PLLP\_DIV2;**

**RCC\_OscInitStruct.PLL.PLLQ = 7;**

**RCC\_OscInitStruct.PLL.PLLR = 2;**

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

**RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_SYSCLK | RCC\_CLOCKTYPE\_HCLK | RCC\_CLOCKTYPE\_PCLK1 | RCC\_CLOCKTYPE\_PCLK2;**

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

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

**RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV4;**

**RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV2;**

**HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_5);**

**}**

**static void MX\_ADC1\_Init(void) {**

**\_\_HAL\_RCC\_ADC1\_CLK\_ENABLE();**

**ADC\_ChannelConfTypeDef sConfig = {0};**

**hadc.Instance = ADC1;**

**hadc.Init.ClockPrescaler = ADC\_CLOCK\_SYNC\_PCLK\_DIV4;**

**hadc.Init.Resolution = ADC\_RESOLUTION\_12B;**

**hadc.Init.ScanConvMode = DISABLE;**

**hadc.Init.ContinuousConvMode = ENABLE;**

**hadc.Init.DiscontinuousConvMode = DISABLE;**

**hadc.Init.NbrOfDiscConversion = 1;**

**hadc.Init.ExternalTrigConvEdge = ADC\_EXTERNALTRIGCONVEDGE\_NONE;**

**hadc.Init.ExternalTrigConv = ADC\_SOFTWARE\_START;**

**hadc.Init.DataAlign = ADC\_DATAALIGN\_RIGHT;**

**hadc.Init.NbrOfConversion = 1;**

**hadc.Init.DMAContinuousRequests = DISABLE;**

**hadc.Init.EOCSelection = ADC\_EOC\_SINGLE\_CONV;**

**HAL\_ADC\_Init(&hadc);**

**sConfig.Channel = ADC\_CHANNEL\_0;**

**sConfig.Rank = 1;**

**sConfig.SamplingTime = ADC\_SAMPLETIME\_84CYCLES;**

**HAL\_ADC\_ConfigChannel(&hadc, &sConfig);**

**}**