Experiment 9

Title: ADC and PWM

Aim: To integrate ADC with PWM on an STM32 Microcontroller

Tool used: Tool used for this assignment is **STM32CubeIDE**.

Procedure:

1. Create a new STM32 project with a suitable project name.

2. IOC UI will open in that configure desired pins as input/output.

3. Under system core select Serial-wire for Debug option.

4.Select channel 0 for ADC.

5.Set the parameter settings for ADC (resolution 12 bits).

6.In TIM2 mode and configuration select clock source as internal clock and channel 1 as PWM generation channel 1. Set prescaler 83 and period 255.

7.In PWM generation set pulse to 0.

8. Press Ctrl+S to generate the code.

9. In the main.c file add the desired code.

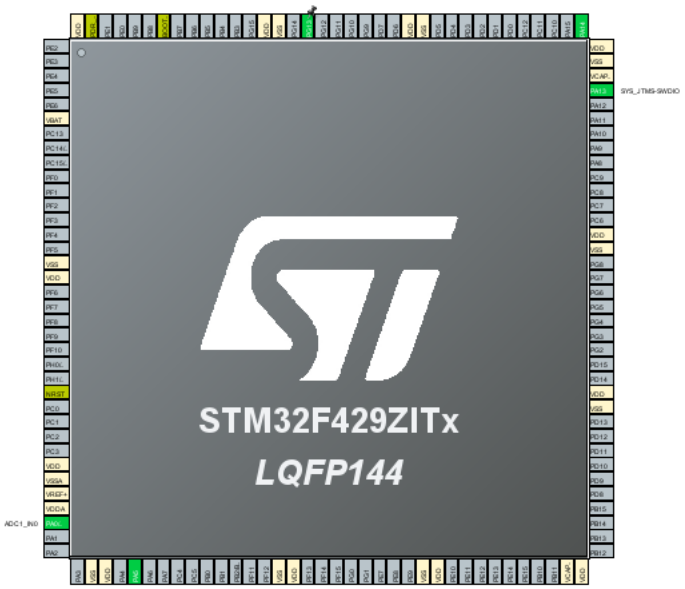
10. Go to Project-> Build Project

11. Connect anode of external leds to the port pins using breadboard and resistors.

12. Give gnd connection between the discovery board and cathode of all leds.

13. Connect the discovery Board and go to Run-> Run.

CubeMx pin diagram:



Code:

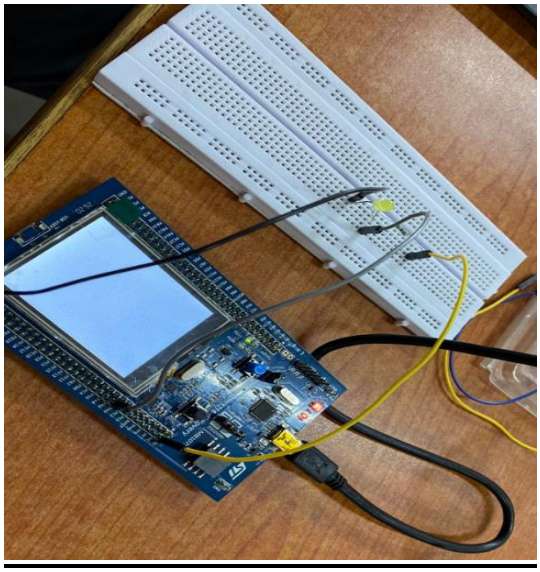
```
#include "stm32f4xx.h"
#include "stm32f4xx_hal.h"

ADC_HandleTypeDef hadc1;
TIM_HandleTypeDef htim2;

void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_ADC1_Init(void);
static void MX_TIM2_Init(void);

int main(void) {
  HAL_Init();
  SystemClock_Config();

  MX_GPIO_Init();
  MX_ADC1_Init();
  MX_TIM2_Init();
```

```
  HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
  HAL_ADC_Start(&hadc1);

  uint32_t adc_value;
  uint16_t pwm_value;

  while (1) {
    if (HAL_ADC_PollForConversion(&hadc1, 10) == HAL_OK) {
      adc_value = HAL_ADC_GetValue(&hadc1);
      pwm_value = (adc_value * 255) / 4095; // Scale ADC value to PWM range (0-255)
      __HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, pwm_value); // Set PWM duty cycle
    }
  }
}

void SystemClock_Config(void) {
  // Configure system clock (e.g., HCLK, PCLK1, PCLK2, etc.)
  // Refer to STM32 reference manual and datasheet for details.
}

static void MX_ADC1_Init(void) {
  ADC_ChannelConfTypeDef sConfig = {0};
  hadc1.Instance = ADC1;
  hadc1.Init.Resolution = ADC_RESOLUTION_12B;
  hadc1.Init.ScanConvMode = ADC_SCAN_DISABLE;
  hadc1.Init.ContinuousConvMode = DISABLE;
  hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
  hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc1.Init.NbrOfConversion = 1;
```

```
  hadc1.Init.DMAContinuousRequests = DISABLE;
  hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
  HAL_ADC_Init(&hadc1);
  sConfig.Channel = ADC_CHANNEL_0; // ADC input channel, adjust as needed
  sConfig.Rank = 1;
  sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK) {
    Error_Handler();
  }
}

static void MX_TIM2_Init(void) {
  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};
  TIM_OC_InitTypeDef sConfigOC = {0};
  htim2.Instance = TIM2;
  htim2.Init.Prescaler = 83; // Adjust prescaler as needed
  htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim2.Init.Period = 255; // PWM period
  htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  if (HAL_TIM_Base_Init(&htim2) != HAL_OK) {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig) != HAL_OK) {
    Error_Handler();
  }
  if (HAL_TIM_PWM_Init(&htim2) != HAL_OK) {
    Error_Handler();
  }
```

```
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL_OK) {
    Error_Handler();
  }
  sConfigOC.OCMode = TIM_OCMODE_PWM1;
  sConfigOC.Pulse = 0;
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  if (HAL_TIM_PWM_ConfigChannel(&htim2, &sConfigOC, TIM_CHANNEL_1) !=
HAL_OK) {
    Error_Handler();
  }
}
```

Output:



Functions used:

HAL_TIM_PWM_Start(&handle_variable_name, TIM_CHANNEL_1);

HAL_ADC_Start(&handle_variable_name);

HAL_ADC_PollForConversion(&handle_variable_name, Timeout);

HAL_ADC_GetValue(&handle_variable_name);

HAL_TIM_SET_COMPARE(&handle_variable_name, TIM_CHANNEL_1, pwm_value);

Result:

Basic STM32Cube project for converting analog voltages from potentiometer and using ADC to vary the brightness of the LED using PWM was built using STM32CubeIDE.