UNIVERSIDADE FEDERAL DO CEARÁ
DEPARTAMENTO DE ENGENHARIA DE TELEINFORMÁTICA
SISTEMAS MICROPROCESSADOS

Atividade Prática 02 - GPIOs

Professor(a): Ricardo Jardel Nunes Silveira

Introdução

O microprocessador STM32F446VE possui 81 pinos de entrada e saída para propósitos gerais. Estes pinos são chamados de GPIOs (General-purpose input/outputs) e podem ser utilizados para diversas funções como acionamento de cargas, leitura de sensores, comandar leds e etc. A Figura 1 destaca em vermelho esses pinos no microprocessador.

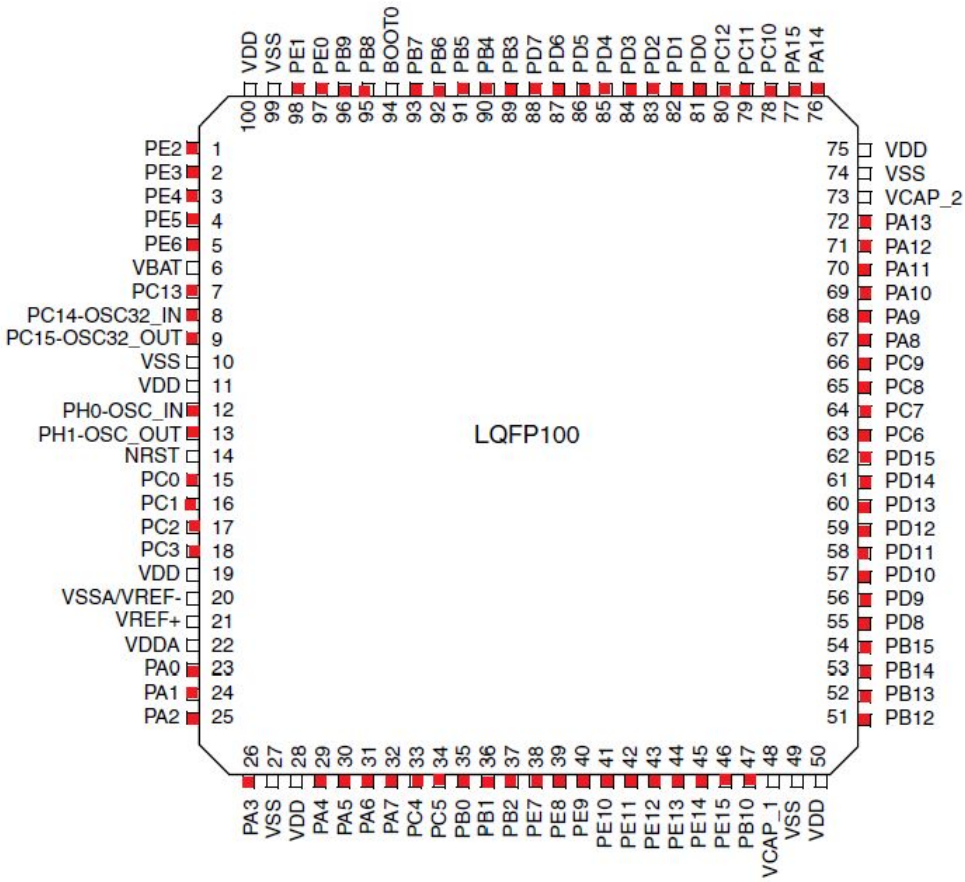

Figura 1. STM32F446VE LQFP100

Estes pinos são distribuídos em 8 portas que variam entre 8 e 16 pinos. Eles são nomeados como PXY, onde X é uma letra variando de A-H que representa a porta, e Y um número entre 0-15 que representa o pino da porta.

Nesta prática você irá aprender a configurar esses pinos como saída para acender um LED e como entrada para ler um botão.

Atividade Prática

1. **Acionamento de LED**

Pelo esquemático, observamos que o LED Vermelho está conectado ao pino PE13. Agora precisamos configurar no STM32Cube este pino como saída.

Na aba Pinout, selecione o pino PE13 e marque a opção GPIO_Output como mostrado na Figura 2. Configure o Clock como você aprendeu na Prática 1.

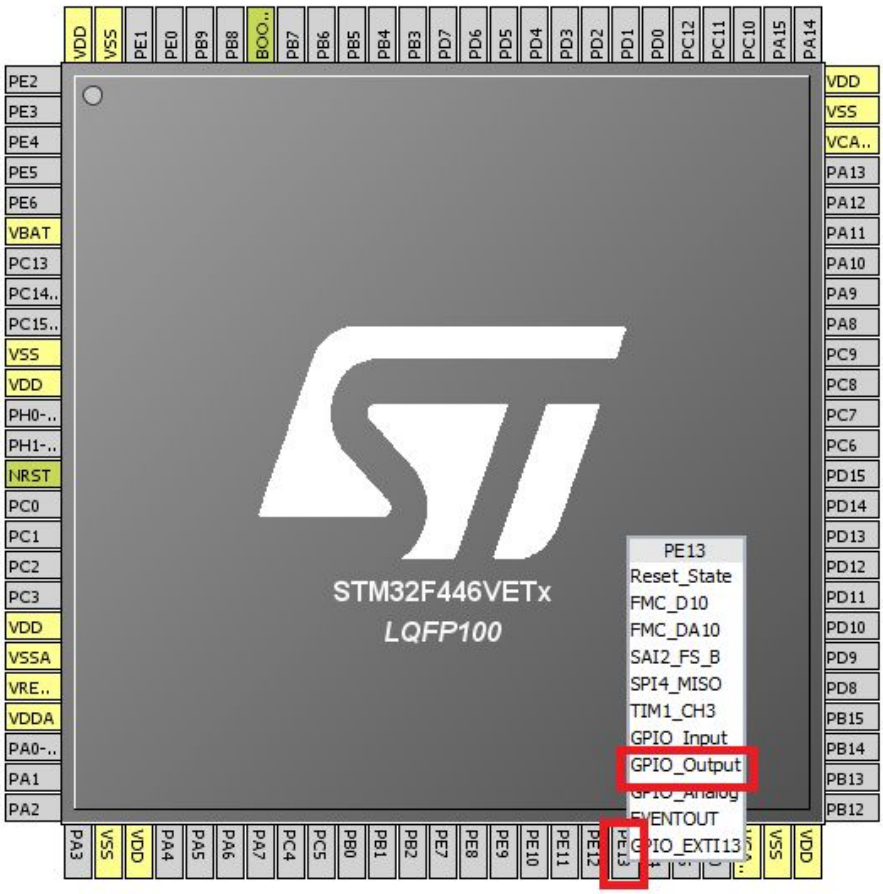

Figura 2. Habilitando pino PE13

O próximo passo é configurar o pino PE13. Clique na aba Configuration e depois no botão GPIO, como ilustrado na Figura 3. Clique no pino PE13, você vai ver a janela mostrada na Figura 4. GPIO output level configura qual o valor inicial do pino, se você quer High ou Low. Deixaremos Low, pois queremos que o LED já comece aceso, pelo

esquemático observamos que o LED está ligado no VCC. Nesta aba, alteraremos apenas o User Label como sinalizado na Figura4 . Por fim, basta gerar o código.

Figura 3. Configurando os GPIOs

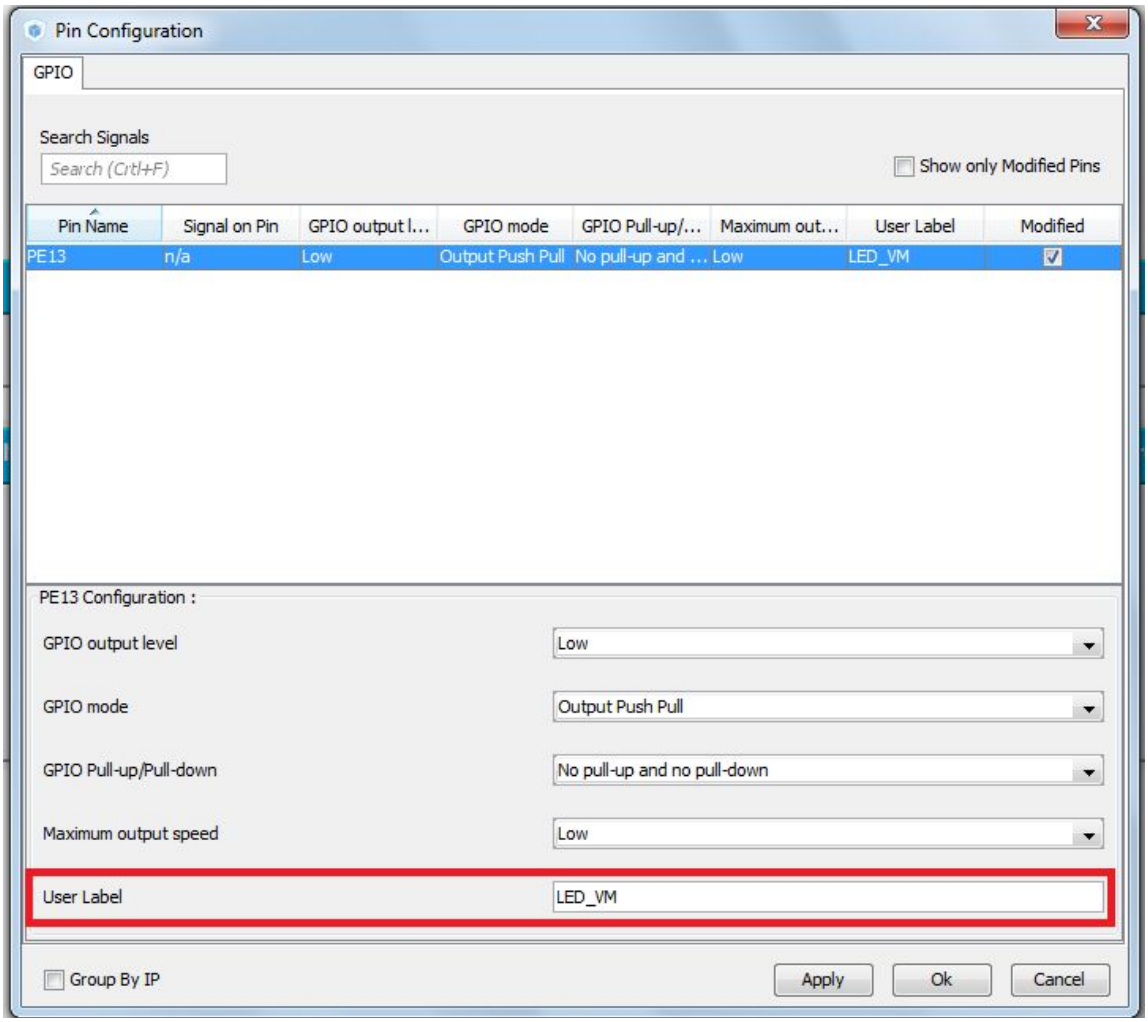

Figura 4. Configurando pino PE13

Agora vamos analisar o código gerado pelo STM32Cube. O código relacionado com o GPIO é mostrado na Figura 5. A função MX_GPIO_Init() que é chamada dentro da função main() é responsável pela configuração do pino PE13. Ela utiliza funções da bilblioteca “stm32f4xx_hal_gpio.c”. É importante destacar as seguintes estruturas e funções.

- **GPIO_InitTypeDef:** É responsável por armazenar as informações de configuração do pino.
- **__HAL_RCC_GPIOE_CLK_ENABLE():** Esta função habilita o clock da Porta E.
- **GPIOE:** Variável que armazena o endereço dos registradores relacionado à Porta E.
- **GPIO_PIN_13:** Variável que armazena o valor do pino 13.
- **GPIO_PIN_RESET:** Enumeração que representa o valor 0
- **HAL_GPIO_WritePin(PORTA, PINO, VALOR LÓGICO)**: Esta função é responsável pela escrita em GPIOs. Você passa a porta (A-H), o pino (0-15) e o valor lógico que você quer esrever na porta (Alto ou Baixo).
- **HAL_GPIO_Init():** Função para inicializar o GPIO.

```
136 void MX_GPIO_Init(void)
137 {
138
139   GPIO_InitTypeDef GPIO_InitStruct;
140
141   /* GPIO Ports Clock Enable */
142   __HAL_RCC_GPIOE_CLK_ENABLE();
143
144   /*Configure GPIO pin Output Level */
145   HAL_GPIO_WritePin(GPIOE, GPIO_PIN_13, GPIO_PIN_RESET);
146
147   /*Configure GPIO pin : PE13 */
148   GPIO_InitStruct.Pin = GPIO_PIN_13;
149   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
150   GPIO_InitStruct.Pull = GPIO_NOPULL;
151   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
152   HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);
153
154 }
```

Figura 5. Função do GPIO

2. **Pisca LED**

   Adicione o protótipo da função PiscaLed() como ilustrado na Figura 6.

```
53 /* USER CODE BEGIN PFP */
54 /* Private function prototypes ----------
55 void PiscaLed(void);
```

Figura 6. Protótipo da função PiscaLed()

Escreva o código da função PiscaLed() como mostrado na Figura 7.

```
/* USER CODE BEGIN 4 */
void PiscaLed(void){

    HAL_GPIO_WritePin(GPIOE,GPIO_PIN_13,  GPIO_PIN_SET);

    HAL_Delay(500);

    HAL_GPIO_WritePin(GPIOE,GPIO_PIN_13,  GPIO_PIN_RESET);

    HAL_Delay(500);
}

```

Figura 7. Implementação da função PiscaLed()

Por fim, chame a função PiscaLED() dentro do laço While da main() e grave o código na placa.

```
int main(void)
{

    /* USER CODE BEGIN 1 */

    /* USER CODE END 1 */

    /* MCU Configuration--------------------------------------------------------*/

    /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
    HAL_Init();

    /* Configure the system clock */
    SystemClock_Config();

    /* Initialize all configured peripherals */
    MX_GPIO_Init();

    /* USER CODE BEGIN 2 */

    /* USER CODE END 2 */

    /* Infinite loop */
    /* USER CODE BEGIN WHILE */
    while (1)
    {
        /* USER CODE BEGIN 3 */
        PiscaLed();

        /* USER CODE END 3 */
```

Figura 8. Chamada da função PiscaLed()

3. Botão e LED

Nas atividades anteriores aprendemos como configurar os GPIOs com saída e controlá-los utilizando sinais lógicos de ALTO e BAIXO. Nesta atividade vamos aprender como configurar entradas e ler os sinais. O objetivo desta atividade é alternar entre os LEDs Vermelho e Azul quando o botão SM400 da nossa placa for pressionado.

Primeiro, repita os passos da Atividade 1 e configure o LED Azul. Pelo esquemático, vemos que o botão SM400 está ligado ao pino PD4. Configure esse pino como mostrado na Figura 9. Na aba Pinout, clique em PD4 e marque a opção GPIO_Input.

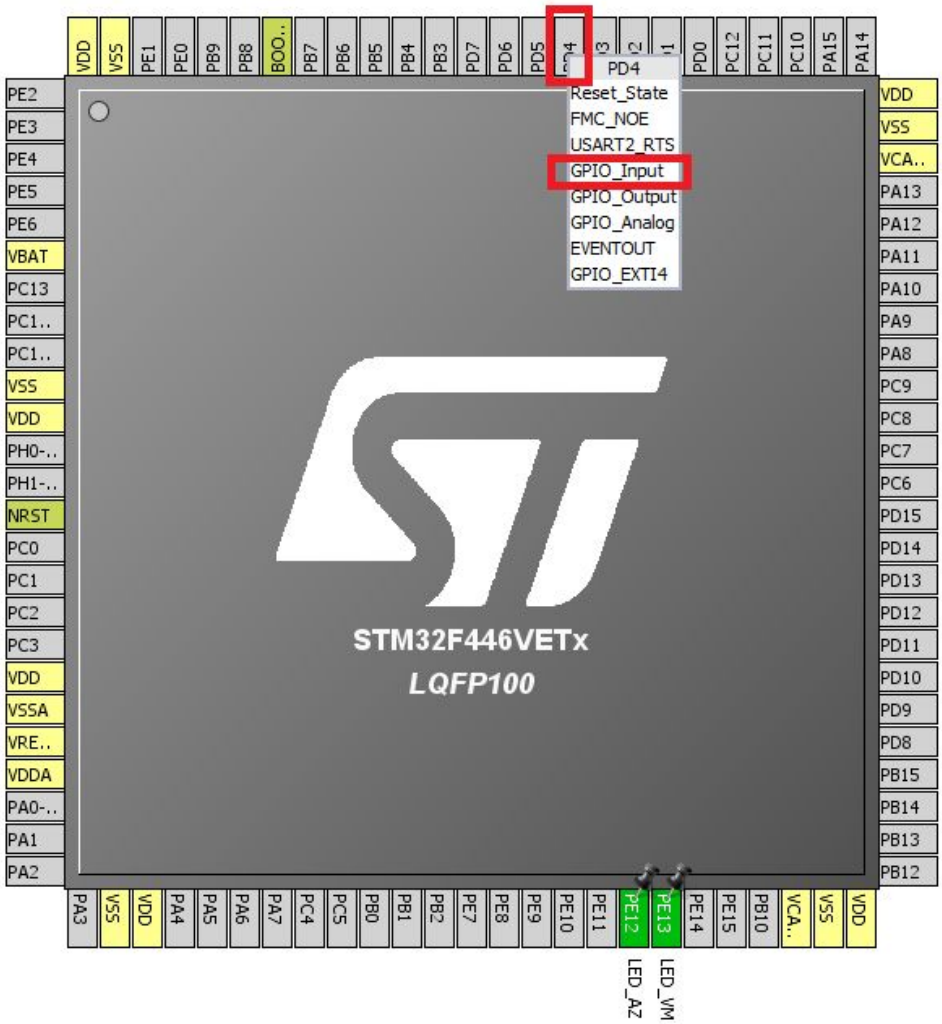

Figura 9. Habilitando o pino PD4

Clique na aba Configuration e depois clique no botão GPIO como mostrado na Figura 3. Então selecione o PD4 e altere o GPIO Pull-up/Pull-down para Pull-up, pois esta é a configuração na placa. Também mude o User Label para Botão. As configurações devem estar iguais a da Figura 10.

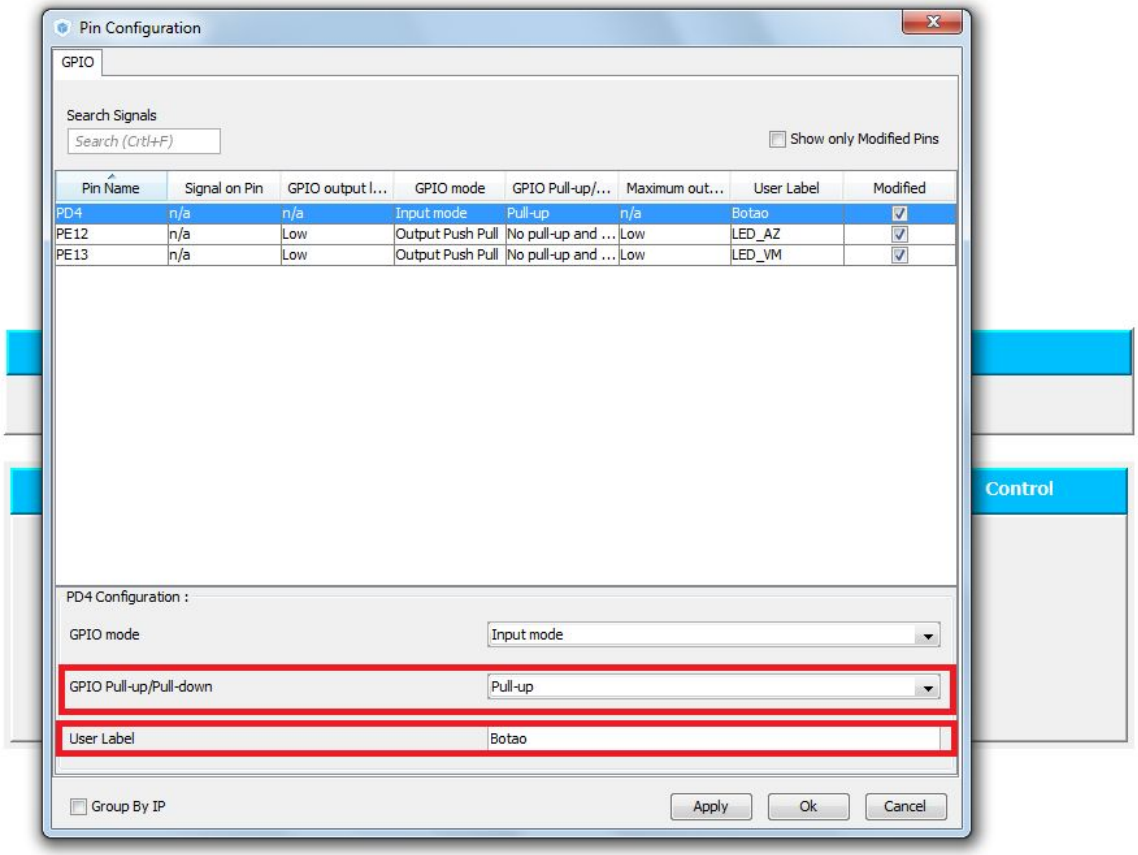

Figura 10. Configurando o pino PD4

Pronto. Só gerar o código.

Adicione o protótipo da função BotaoLed() e a função BotaoLed() ao código.

```
51 /* USER CODE BEGIN PFP */
52 /* Private function prototypes -----------------------
53 void Delay(__IO uint32_t nCount);
54 void PiscaLed(void);
55 void BotaoLed(void);
```

Figura 11. Adicionando o protótipo da função BotaoLed()

```
186 void BotaoLed(void){
187
188     if (HAL_GPIO_ReadPin(GPIOD, GPIO_PIN_4) == GPIO_PIN_RESET){
189         HAL_GPIO_WritePin(GPIOE,GPIO_PIN_13,  GPIO_PIN_SET);
190         HAL_GPIO_WritePin(GPIOE,GPIO_PIN_12,  GPIO_PIN_RESET);
191     }
192     else {
193         HAL_GPIO_WritePin(GPIOE,GPIO_PIN_13,  GPIO_PIN_RESET);
194         HAL_GPIO_WritePin(GPIOE,GPIO_PIN_12,  GPIO_PIN_SET);
195     }
196 }
197
```

Figura 12. Implementação da função BotaoLed()

Substitua onde você adicionou PiscaLed() por BotaoLed() na Figura 8 e grave o código na placa.

Exercícios

1. Reescreva a função PiscaLed() utilizando a função HAL_GPIO_TogglePin().
2. Reescreva a função BotaoLed() para que o Led Vermelho mude de estado a cada vez que o botão for pressionado.

Referências

[1] Esquema elétrico do kit de desenvolvimento
[2] Documentação das bibliotecas do STM32F446VE