```c
#include "main.h"
#define NUM_ROWS 4
#define NUM_COLS 4
#define ROW_PORT (GPIO_IDR_ID0 | GPIO_IDR_ID1 | GPIO_IDR_ID2 | GPIO_IDR_ID3)
#define COL_PORT (GPIO_ODR_OD4 | GPIO_ODR_OD5 | GPIO_ODR_OD6 | GPIO_ODR_OD7)
#define LED_PORT (GPIO_ODR_OD4 | GPIO_ODR_OD5 | GPIO_ODR_OD6 | GPIO_ODR_OD7)
void SystemClock_Config(void);

//DEMO VIDEO: https://youtube.com/shorts/5qPlSSoLN4w?si=dc9cr0x66fUFotGn

// Look-up table for keypad values
static uint32_t keypad[4][4] = {
  {1, 2, 3, 12}, // 12 represents A
  {4, 5, 6, 13}, // 13 represents B
  {7, 8, 9, 14}, // 14 represents C
  {10, 0, 11, 15} // 10 = *, 11 = #, 15 = D
};


void keypad_init(void) {
  // Configure PC4-PC7 as outputs
  GPIOC->MODER &= ~(GPIO_MODER_MODE4 | GPIO_MODER_MODE5 | GPIO_MODER_MODE6 |
GPIO_MODER_MODE7);
  GPIOC->MODER |= (GPIO_MODER_MODE4_0 | GPIO_MODER_MODE5_0 | GPIO_MODER_MODE6_0 |
GPIO_MODER_MODE7_0);

  // Configure PC0-PC3 as inputs
  GPIOC->MODER &= ~(GPIO_MODER_MODE0 | GPIO_MODER_MODE1 | GPIO_MODER_MODE2 |
GPIO_MODER_MODE3);

  // Configure PC0-PC3 as pull-down resistors
  GPIOC->PUPDR &= ~(GPIO_PUPDR_PUPD0 | GPIO_PUPDR_PUPD1 | GPIO_PUPDR_PUPD2 |
GPIO_PUPDR_PUPD3);
  GPIOC->PUPDR |= (GPIO_PUPDR_PUPD0_1 | GPIO_PUPDR_PUPD1_1 | GPIO_PUPDR_PUPD2_1 |
GPIO_PUPDR_PUPD3_1);

  // Configure PC4-PC7 as push-pull outputs
  GPIOC->OTYPER &= ~(GPIO_OTYPER_OT4 | GPIO_OTYPER_OT5 | GPIO_OTYPER_OT6 | GPIO_OTYPER_OT7);

  //All of them are slow
  GPIOC->OSPEEDR &= ~(GPIO_OSPEEDR_OSPEED0 | GPIO_OSPEEDR_OSPEED1 | GPIO_OSPEEDR_OSPEED2
          | GPIO_OSPEEDR_OSPEED3 | GPIO_OSPEEDR_OSPEED4 | GPIO_OSPEEDR_OSPEED5
          | GPIO_OSPEEDR_OSPEED6 | GPIO_OSPEEDR_OSPEED7);

  // Set PC4-PC7 high initially, all COLS high
  GPIOC->ODR &= (COL_PORT);
}

uint32_t getKeypadResult(uint32_t arr) {
    switch (arr) {
        case 0x0001: return 0; // BIN 0001
        case 0x0002: return 1; // BIN 0010
        case 0x0004: return 2; // BIN 0100
        case 0x0008: return 3; // BIN 1000
        default:
            return -1; // Invalid or no key pressed
    }
```

```
 53 }
 54
 55 uint32_t getKeypadNumber(void){
 56
 57     //Detect if a button is pressed
 58     if((GPIOC->IDR & ROW_PORT) != 0){
 59         for(uint32_t col = 0; col < NUM_COLS; col++){
 60             // Clear all columns then set them high one by one
 61             GPIOC->ODR &= ~(COL_PORT);
 62             GPIOC->ODR |= (1 << (col + 4)); // +4 because GPIOC PC4 - PC7 is used
 63
 64             if((GPIOC->IDR & ROW_PORT) != 0){
 65                 int colRes = getKeypadResult(1 << col);
 66                 int rowRes = getKeypadResult(GPIOC->IDR & ROW_PORT);
 67                 //If pressing to at once or let go in the middle
 68                 if(colRes == -1 || rowRes == -1){
 69                     return -1;
 70                 }
 71                 // reset high for next press
 72                 GPIOC->ODR |= COL_PORT;
 73                 return keypad[rowRes][colRes];
 74             }
 75         }
 76     }
 77     // reset high for next press
 78     GPIOC->ODR |= (COL_PORT);
 79     return -1;
 80 }
 81
 82
 83
 84 int main(void)
 85 {
 86   // System init
 87   SystemClock_Config();
 88
 89   // Enable LED clock (GPIOA)
 90   RCC -> AHB2ENR |= (RCC_AHB2ENR_GPIOAEN);
 91
 92   // Enable KEYPAD clock (GPIOC)
 93   RCC -> AHB2ENR |= (RCC_AHB2ENR_GPIOCEN);
 94
 95   // LED outputs for GPIOA PA4 - PA7 (output, slow, no push/pull)
 96   GPIOA->MODER &= ~(GPIO_MODER_MODE4 | GPIO_MODER_MODE5 | GPIO_MODER_MODE6 |
    GPIO_MODER_MODE7);
 97   GPIOA->MODER |= (GPIO_MODER_MODE4_0 | GPIO_MODER_MODE5_0 | GPIO_MODER_MODE6_0 |
    GPIO_MODER_MODE7_0);
 98   GPIOA->OTYPER &= ~(GPIO_OTYPER_OT4 | GPIO_OTYPER_OT5 | GPIO_OTYPER_OT6 | GPIO_OTYPER_OT7);
 99   GPIOA->OSPEEDR &= ~(GPIO_OSPEEDR_OSPEED4 | GPIO_OSPEEDR_OSPEED5
100           | GPIO_OSPEEDR_OSPEED6 | GPIO_OSPEEDR_OSPEED7);
101   // Initialize keypad (uses GPIOC)
102   keypad_init();
103
104   while (1)
105   {
106
107     uint32_t number = getKeypadNumber();
```

```
108
109     // 12 is A, 13 is B, 14 is C, 15 is D, 10 is *, 11 is #
110     if(number != -1)  // If key pressed, show number on LEDs
111     {
112         GPIOA->ODR &= ~LED_PORT;
113         GPIOA->ODR |= (number << 4);
114
115
116     }
117   }
118 }
119
120 void SystemClock_Config(void)
121 {
122   RCC_OscInitTypeDef RCC_OscInitStruct = {0};
123   RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
124
125   /** Configure the main internal regulator output voltage
126   */
127   if (HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1) != HAL_OK)
128   {
129     Error_Handler();
130   }
131
132   /** Initializes the RCC Oscillators according to the specified parameters
133   * in the RCC_OscInitTypeDef structure.
134   */
135   RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_MSI;
136   RCC_OscInitStruct.MSIState = RCC_MSI_ON;
137   RCC_OscInitStruct.MSICalibrationValue = 0;
138   RCC_OscInitStruct.MSIClockRange = RCC_MSIRANGE_6;
139   RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
140   if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
141   {
142     Error_Handler();
143   }
144
145   /** Initializes the CPU, AHB and APB buses clocks
146   */
147   RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
148                                 |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
149   RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_MSI;
150   RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
151   RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
152   RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
153
154   if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK)
155   {
156     Error_Handler();
157   }
158 }
159
160 void Error_Handler(void)
161 {
162   __disable_irq();
163   while (1)
164   {
```

```
165       break;
166     }
167   }
168
169 #ifdef  USE_FULL_ASSERT
170 void assert_failed(uint8_t *file, uint32_t line)
171 {
172 }
173 #endif
174
```