TUGAS BESAR MIKROPROSESOR DAN ANTARMUKA

Reaction Game Berbasis STM32F103

Disusun Oleh

Maulana Hafiz	1103223105
Dhanendra Nivadirokhman	1103220113
Marcellinus Geofani Sihaloho	1103223171
Raffalino Djira Ibrahim	1103223021

Teknik Komputer
Fakultas Teknik Elektro
Universitas Telkom
2024

A. Latar Belakang

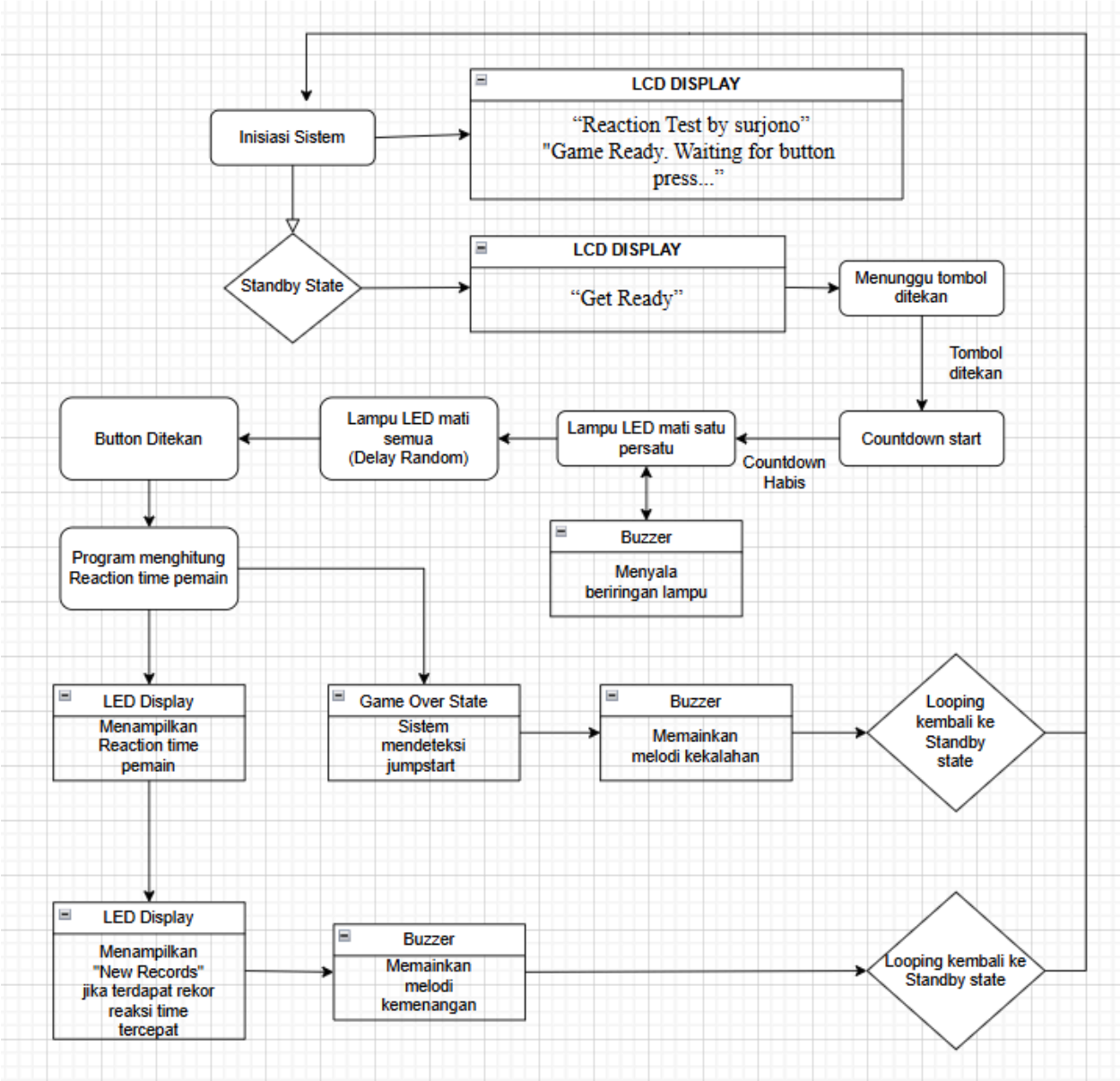
Dalam era sekarang banyak teknologi *mikroprosessor* yang memainkan peran penting dalam menciptakan solusi inovatif untuk berbagai kebutuhan, salah satunya di bidang hiburan interaktif. Salah satu perangkat yang memiliki potensi besar dalam pengembangan teknologi berbasis mikroprosesor adalah Blue Pill, sebuah mikrokontroler berbasis STM32 yang dikenal karena efisiensi, performa tinggi, serta harganya yang terjangkau.Game interaktif yang menguji kemampuan reaksi pengguna telah menjadi salah satu bentuk hiburan yang populer. Reaction game tidak hanya memberikan hiburan, tetapi juga memiliki manfaat dalam meningkatkan refleks, koordinasi tangan-mata, serta kemampuan kognitif pengguna.

Proyek ini bertujuan untuk merancang dan mengimplementasikan sebuah *reaction game* berbasis mikrokontroler Blue Pill. Game ini dirancang untuk memberikan pengalaman interaktif dengan memanfaatkan LED dan push button sebagai antarmuka utama. Melalui proyek ini, mahasiswa dapat mempraktikkan teori yang telah dipelajari, termasuk pemrograman mikrokontroler, pengelolaan sinyal digital, serta integrasi perangkat keras dan perangkat lunak.

B. Alat dan Bahan

1. STM32F103 Blue Pill
2. Breadboard
3. LCD I2c 16x2
4. LED
5. Buzzer Passive
6. Kabel Jumper
7. Kabel Connector
8. Switch Button

C. Cara Kerja



1. Inisiasi sistem
    - Saat rangkaian dinyalakan, mikrokontroller Bluepill akan melakukan inisiasi yaitu konfigurasi pin LED, Push Button, dan LCD display
    - Sistem juga akan menampilkan “Reaction Test by surjono” dan "Game Ready. Waiting for button press...” untuk memberi tahu pengguna bahwa game siap dimainkan

2. Mode Standby
    - Sistem akan berada di mode stanby menunggu pengguna menekan tombol untuk memulai permainan
    - Setelah tombol ditekan maka game akan mulai countdown yang bisa dilihat dari LED Display
    - Setelah LED Display menunjukan angka 0, lampu LED mati satu persatu LED Display akan menampilkan “Get Ready” agar player bersiap menekan tombol saat semua lampu LED mati
    - Buzzer akan menyala beriringan dengan LED mati satu persatu

3. Reaction Time
    - Saat lampu LED mati semua pengguna harus menekan tombol secepat mungkin yang delay matinya semua LED diatur secara random
    - LED Display akan menampilkan Reaction time yang didapat player Setelah menekan tombol, Jika player mencatat rekor reaksi tercepat LED Display juga akan menampilkan “New Record !” dan buzzer memainkan melody kemenangan
    - Beriringan dengan tombol ditekan buzzer juga akan berbunyi 5 kali
    - Game akan menglooping dari awal kembali

4. Game over state
    - Apabila pengguna menekan tombol sebelum lampu LED mati semua maka akan ada output display yang memberi tahu bahwa pengguna terlalu awal menekan button dan Buzzer akan memainkan melody game over
    - Game akan menglooping dari awal kembali

D. Hasil Implementasi

Source Code:

```
#include "stm32f1xx.h"
#include "stm32f1xx_hal.h"

// Pin and configuration constants
#define BUTTON_PIN GPIO_PIN_12          // Button connected to pin PB12
#define NUM_LEDS 5                      // Number of LEDs
const uint16_t ledPins[NUM_LEDS] = {GPIO_PIN_0, GPIO_PIN_1, GPIO_PIN_3, GPIO_PIN_0,
GPIO_PIN_1}; // Array of LED pins (PA0, PA1, PA3, PB0, PB1)

#define BUZZER_PIN GPIO_PIN_14          // Buzzer connected to pin PC14
#define LED_DELAY 1000                  // Delay between LED transitions (ms)
```

```
#define MIN_WAIT_DELAY 500            // Minimum random wait delay (ms)
#define MAX_WAIT_DELAY 5000           // Maximum random wait delay (ms)

// Variables to track button press and timing
volatile uint8_t btnPressed = 0;      // Set by the ISR when the button is pressed
uint32_t btnPressedTime = 0;
uint32_t reactionTime = 0;
uint32_t ledsOffTime = 0;
uint32_t record = 5000;               // Initial record time in ms

// Interrupt Service Routine for button press
void EXTI15_10_IRQHandler(void) {
  if (__HAL_GPIO_EXTI_GET_IT(BUTTON_PIN) != RESET) {
    __HAL_GPIO_EXTI_CLEAR_IT(BUTTON_PIN); // Clear the interrupt flag
    btnPressed = 1;                   // Set button pressed flag
    btnPressedTime = HAL_GetTick();       // Record the time of the button press
  }
}

// Function to initialize GPIO pins
void GPIO_Init(void) {
  __HAL_RCC_GPIOB_CLK_ENABLE(); // Enable GPIOB clock
  __HAL_RCC_GPIOC_CLK_ENABLE(); // Enable GPIOC clock
  __HAL_RCC_GPIOA_CLK_ENABLE(); // Enable GPIOA clock

  GPIO_InitTypeDef GPIO_InitStruct = {0};

  // Initialize button pin (PB12)
  GPIO_InitStruct.Pin = BUTTON_PIN;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;  // Interrupt on falling edge
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  // Initialize LED pins
  for (int i = 0; i < NUM_LEDS; i++) {
    GPIO_InitStruct.Pin = ledPins[i];
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;  // Push-pull output mode
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
    HAL_GPIO_WritePin(GPIOA, ledPins[i], GPIO_PIN_RESET);  // Ensure LEDs are off
  }

  // Initialize buzzer pin (PC14)
  GPIO_InitStruct.Pin = BUZZER_PIN;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
  HAL_GPIO_WritePin(GPIOC, BUZZER_PIN, GPIO_PIN_RESET);  // Ensure buzzer is off
}
```

```
// Function to initialize the Timer for delays (if needed for buzzer or timing)
void Timer_Init(void) {
  HAL_SYSTICK_Config(HAL_RCC_GetHCLKFreq() / 1000);  // Configure SysTick to generate 1ms
ticks
}

// Function to play melody
void playMelody(uint16_t *melody, uint16_t *durations, uint8_t length) {
  for (int i = 0; i < length; i++) {
    HAL_GPIO_WritePin(GPIOC, BUZZER_PIN, GPIO_PIN_SET);  // Turn on buzzer
    HAL_Delay(durations[i]);                             // Wait for the duration
    HAL_GPIO_WritePin(GPIOC, BUZZER_PIN, GPIO_PIN_RESET); // Turn off buzzer
    HAL_Delay(50); // Short pause between notes
  }
}

// Main game loop
int main(void) {
  HAL_Init();
  GPIO_Init();
  Timer_Init();

  // Configure external interrupt for the button (PB12)
  HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);  // Enable EXTI interrupt line for PB12

  while (1) {
    // Wait for button press to start the game
    if (!btnPressed) {
      // Display waiting message here (use USART or another method to show it on a screen)
      continue;  // Stay in this loop until button is pressed
    }

    // Reset button press flag
    btnPressed = 0;

    // Countdown before the game starts (Display this on an LCD or USART)
    HAL_Delay(3000);  // Simulated countdown

    // Sequentially turn on LEDs and beep
    for (int i = 0; i < NUM_LEDS; i++) {
      if (btnPressed) {
        // Button pressed too early, play losing melody
        uint16_t loseMelody[] = {220, 196, 174, 164, 147};
        uint16_t loseDurations[] = {300, 300, 300, 300, 300};
        playMelody(loseMelody, loseDurations, 5);

        // Turn off all LEDs and stop the game
        for (int k = 0; k < NUM_LEDS; k++) {
          HAL_GPIO_WritePin(GPIOA, ledPins[k], GPIO_PIN_RESET);
        }
```

```
      HAL_Delay(3000);
      continue;  // Restart the game loop
    }

    HAL_GPIO_WritePin(GPIOA, ledPins[i], GPIO_PIN_SET);  // Turn on LED
    // Play beep sound (this can be done using a timer interrupt or delay)
    HAL_GPIO_WritePin(GPIOC, BUZZER_PIN, GPIO_PIN_SET);
    HAL_Delay(200);  // Beep duration
    HAL_GPIO_WritePin(GPIOC, BUZZER_PIN, GPIO_PIN_RESET);
    HAL_Delay(LED_DELAY - 200);  // Remaining time with silence
   }

   // Random delay before turning LEDs off
   uint32_t waitDelay = MIN_WAIT_DELAY + (rand() % (MAX_WAIT_DELAY -
MIN_WAIT_DELAY + 1));
   HAL_Delay(waitDelay);

   // Turn off LEDs and measure reaction time
   ledsOffTime = HAL_GetTick();
   for (int i = 0; i < NUM_LEDS; i++) {
    HAL_GPIO_WritePin(GPIOA, ledPins[i], GPIO_PIN_RESET);
   }

   // Start buzzer sound without blocking
   HAL_GPIO_WritePin(GPIOC, BUZZER_PIN, GPIO_PIN_SET);  // Higher frequency
   btnPressed = 0;

   while (!btnPressed) {
    // Wait for ISR to set btnPressed
   }

   // Stop buzzer sound
   HAL_GPIO_WritePin(GPIOC, BUZZER_PIN, GPIO_PIN_RESET);

   // Calculate reaction time
   reactionTime = HAL_GetTick() - ledsOffTime;

   // Check for new record and play winning sound
   if (reactionTime < record) {
    record = reactionTime;
    // Play winning melody
    uint16_t winMelody[] = {262, 294, 330, 349, 392, 440};
    uint16_t winDurations[] = {300, 300, 300, 300, 300, 300};
    playMelody(winMelody, winDurations, 6);
   }

   HAL_Delay(5000);  // Wait before restarting the game
  }
}
```

E. Kesimpulan

Sistem ini adalah permainan reaksi berbasis mikrokontroler STM32F103 yang menguji kecepatan reaksi pemain dengan urutan LED yang menyala dan mati, diiringi suara buzzer sebagai petunjuk. Pemain diminta untuk menekan tombol sesegera mungkin setelah LED terakhir mati, dan waktu reaksi diukur. Sistem ini menggunakan LED untuk menampilkan urutan, buzzer untuk memberikan umpan balik suara, dan tombol untuk merekam respons pemain. Setelah setiap permainan, waktu reaksi pemain dibandingkan dengan rekor sebelumnya, dan melodi kemenangan atau kekalahan dimainkan berdasarkan hasil reaksi. Sistem ini bertujuan untuk memberikan tantangan reaksi cepat dan mencatat waktu terbaik pemain, dengan komponen hardware yang sederhana namun efektif.