

YOZGAT BOZOK ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

MİKROİŞLEMCİ SİSTEMLER VE UYGULAMALARI DERSİ

TEKNİK RAPOR

“STM32 Mikrodenetleyici ve Mobil Uygulama ile Parmak İzi Kontrollü Kilit Sistemi”

HAZIRLAYANLAR

ALİ İHSAN BALAMUR

2022

1. **GİRİŞ**

**AMAÇ**

Günümüzde teknolojinin hızla gelişmesiyle birlikte, günlük yaşamımızda kullanılan cihazlar ve sistemler de giderek daha akıllı ve etkileşimli hale gelmektedir. Bu bağlamda, güvenlik sistemleri özellikle önemli bir alan olarak öne çıkmaktadır. Bizim projemiz, bu teknolojik ilerlemenin bir parçası olarak, günlük yaşamda kullanılabilecek yenilikçi bir kilit sistemi geliştirmeyi amaçlamaktadır.

**KULLANILAN TEKNOLOJİLER VE BİLEŞENLER**

Projemiz, STM32F103C8T6 mikrodenetleyici, HC-06 Bluetooth modülü ve SG-90 9G servo motor kullanarak tasarlanmış bir kilit mekanizması üzerine kuruludur. Bu kilit sistemi, servo motorun kontrolü ile etkin bir şekilde çalışmaktadır.

**Mobil Uygulama**

Projemizin bir diğer önemli bileşeni ise Flutter kullanılarak geliştirilen bir mobil uygulamadır. Bu uygulama, kullanıcı dostu bir arayüz sunarak, kilit sisteminin uzaktan kontrolünü mümkün kılmaktadır. Uygulama, Bluetooth üzerinden HC-06 modülüne bağlanarak, STM32 mikrodenetleyiciye komutlar gönderir. Bu komutlar sayesinde, servo motorun hareketi kontrol edilerek kilit açılıp kapatılabilmektedir.

**Geliştirme Ortamı**

Projenin geliştirme sürecinde, STM32 Cube IDE kullanılarak mikrodenetleyicinin pin ve saat konfigürasyonları yapılmış, STM32 Utility ile de STM32 için yazılan kod hex dosyası biçiminde cihaza yüklenmiştir. Ayrıca, projenin fiziksel yapısını oluşturmak için 3 boyutlu bir yazıcı kullanılarak, tough PLA filamenti ile bir kutu üretilmiştir. Bu kutu, kilit sisteminin uygulamalı olarak gösterilmesi için kullanılmıştır.

Bu rapor, söz konusu kilit sisteminin tasarımı, geliştirilmesi ve işlevselliği hakkında detaylı bilgiler sunmayı amaçlamaktadır. Projemiz, teknolojinin günlük yaşamdaki uygulamalarına yenilikçi bir bakış açısı getirerek, güvenlik sistemlerinin nasıl daha etkileşimli ve kullanıcı dostu hale getirilebileceğini göstermektedir.

1. **GELİŞME**

Donanım Tasarımı ve Kurulumu: Projemizin temelini oluşturan donanım, STM32F103C8T6 mikrodenetleyici, HC-06 Bluetooth modülü ve SG-90 9G servo motor içermektedir. Bu bileşenler, bir kilit mekanizmasını kontrol etmek üzere entegre edilmiştir.

* STM32F103C8T6 Mikrodenetleyici: Bu mikrodenetleyici, projemizin beyni olarak görev yapmakta ve tüm kontrol sinyallerini işlemektedir.
* HC-06 Bluetooth Modülü: Bu modül, mikrodenetleyici ile mobil uygulama arasında kablosuz iletişim kurmamızı sağlar.
* SG-90 9G Servo Motor: Kilit mekanizmasının fiziksel hareketini gerçekleştiren bu motor, mikrodenetleyiciden gelen komutlarla yönetilmektedir.

Bu bileşenlerin birleştirilmesi ve bağlantıları, [Proje Şeması] bölümünde detaylı olarak ifade edilmiştir.

Yazılım Geliştirme: Projemizin yazılım kısmı, iki ana bölümden oluşmaktadır: mikrodenetleyici için yazılan kod ve Flutter ile geliştirilen mobil uygulama.

* Mikrodenetleyici Yazılımı: STM32 Cube IDE kullanılarak geliştirilen bu yazılım, mikrodenetleyicinin pin ve saat konfigürasyonlarını yönetir ve Bluetooth modülü üzerinden gelen komutlara yanıt verir. Kodun önemli bölümleri [Kod Parçaları] bölümünde sunulmuştur.
* Mobil Uygulama: Flutter kullanılarak geliştirilen bu uygulama, kullanıcı dostu bir arayüz sunar ve Bluetooth üzerinden HC-06 modülüne bağlanarak kilit mekanizmasını kontrol eder. Uygulamanın ekran görüntüleri [Uygulama Ekran Görüntüleri] bölümünde yer almaktadır.

**Test ve Uygulama:** Projemizin her bir bileşeni, test edilmiştir. Bu testler, sistemimizin güvenilirliğini ve etkinliğini doğrulamak için yapılmıştır. Test süreçleri [Test Süreçleri] bölümünde ayrıntılı olarak ifade edilmiştir.

1. **SONUÇ**

Projemiz, STM32F103C8T6 mikrodenetleyici, HC-06 Bluetooth modülü ve SG-90 9G servo motor kullanarak geliştirilen yenilikçi bir kilit sisteminin başarılı bir şekilde tasarımını ve uygulamasını göstermiştir. Flutter ile geliştirilen mobil uygulama aracılığıyla uzaktan kontrol edilebilen bu sistem, güvenlik teknolojilerinde kullanıcı dostu ve etkileşimli çözümler sunma potansiyeline sahiptir. Test süreçleri projenin pratik uygulamalarda etkili bir şekilde kullanılabileceğini göstermiştir.

Gelecekte, bu sistemin daha da geliştirilerek, çeşitli uygulama alanlarında kullanılması hedeflenmektedir.

1. **REFERANSLAR**

**[1]** [**https://veyselgokdemir.com/2018/05/01/vn38-stm32f4-hc06-bluetooth-module/**](https://veyselgokdemir.com/2018/05/01/vn38-stm32f4-hc06-bluetooth-module/)

**[2]** [**https://www.alldatasheet.com/view\_datasheet.jsp?Searchword=HC-06**](https://www.alldatasheet.com/view_datasheet.jsp?Searchword=HC-06)

**[3]** [**https://www.alldatasheet.com/view\_datasheet.jsp?Searchword=STM32F103C8T6**](https://www.alldatasheet.com/view_datasheet.jsp?Searchword=STM32F103C8T6)

**[4]** [**https://components101.com/microcontrollers/stm32f103c8t8-blue-pill-development-board**](https://components101.com/microcontrollers/stm32f103c8t8-blue-pill-development-board)

**[5]** [**https://www.youtube.com/watch?v=TAAaE7cHdMQ**](https://www.youtube.com/watch?v=TAAaE7cHdMQ)

**[6]** [**https://www.st.com/resource/en/user\_manual/um2609-stm32cubeide-user-guide-stmicroelectronics.pdf**](https://www.st.com/resource/en/user_manual/um2609-stm32cubeide-user-guide-stmicroelectronics.pdf)

**[7]** [**https://www.youtube.com/watch?v=q3w70u9qmKQ**](https://www.youtube.com/watch?v=q3w70u9qmKQ)

**[8]** [**https://www.youtube.com/watch?v=PWOPuTOXzHs&t=329s**](https://www.youtube.com/watch?v=PWOPuTOXzHs&t=329s)

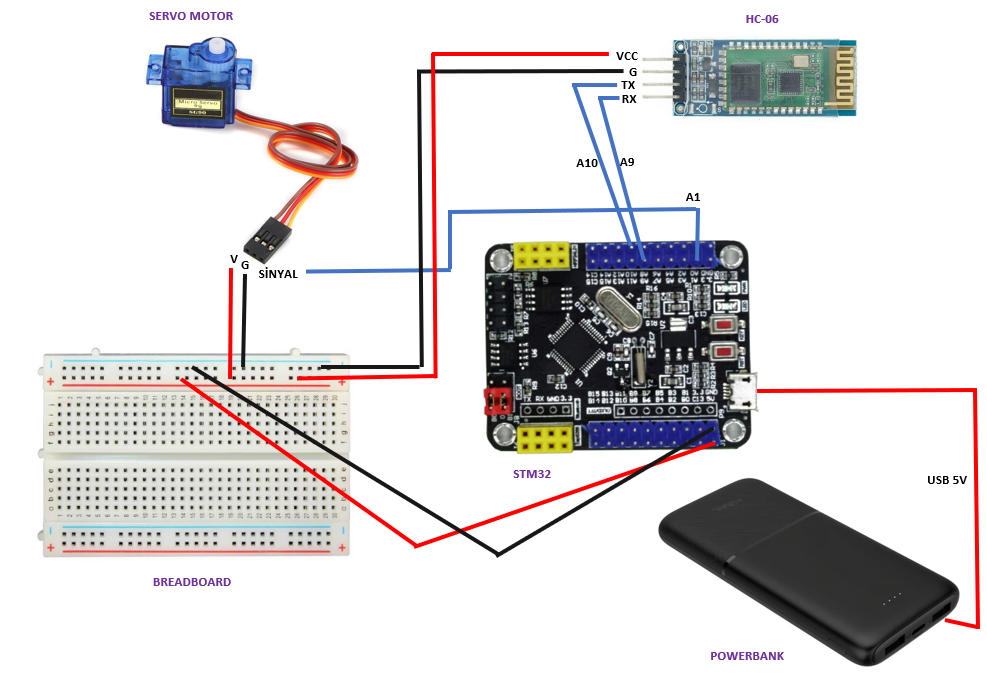
**[9]** [**https://www.youtube.com/watch?v=cnv8bpl67uw**](https://www.youtube.com/watch?v=cnv8bpl67uw)

**[10]** [**https://www.youtube.com/watch?v=R5nNMRF86Dc&t=338s**](https://www.youtube.com/watch?v=R5nNMRF86Dc&t=338s)

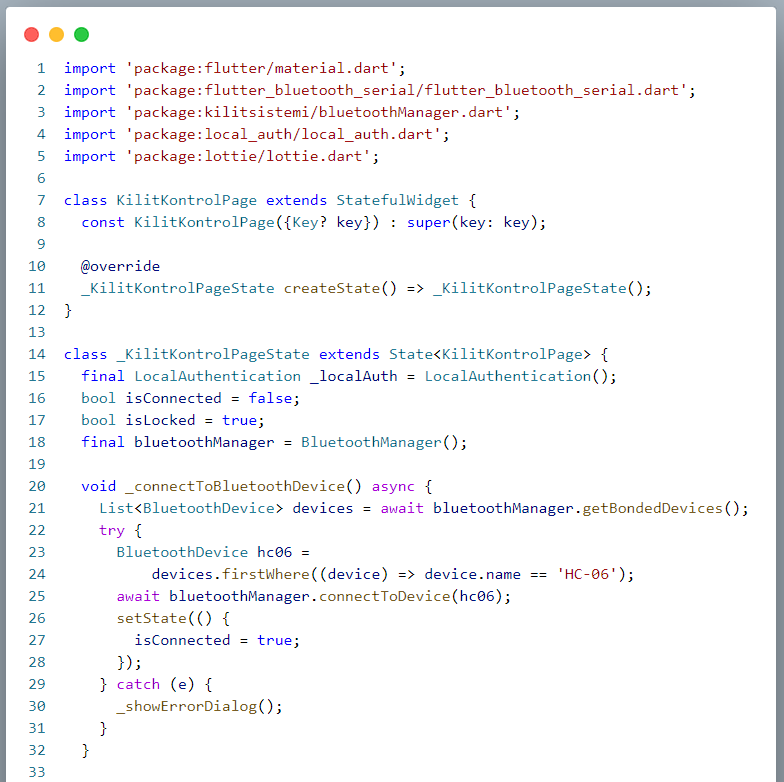
**[11]** [**https://www.alldatasheet.com/view\_datasheet.jsp?Searchword=SG90**](https://www.alldatasheet.com/view_datasheet.jsp?Searchword=SG90)

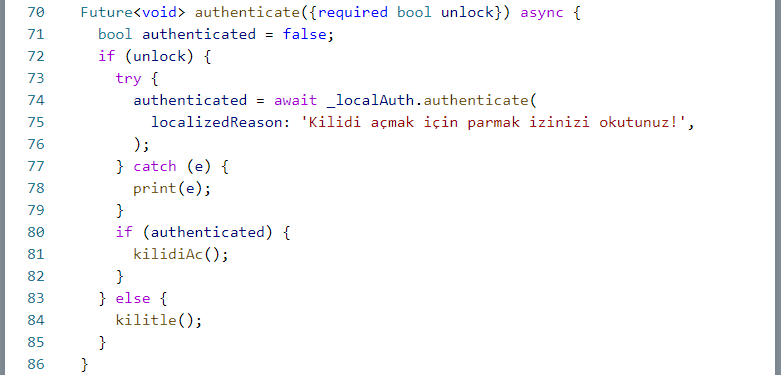
1. **EKLER**

* **Proje Şeması**

****

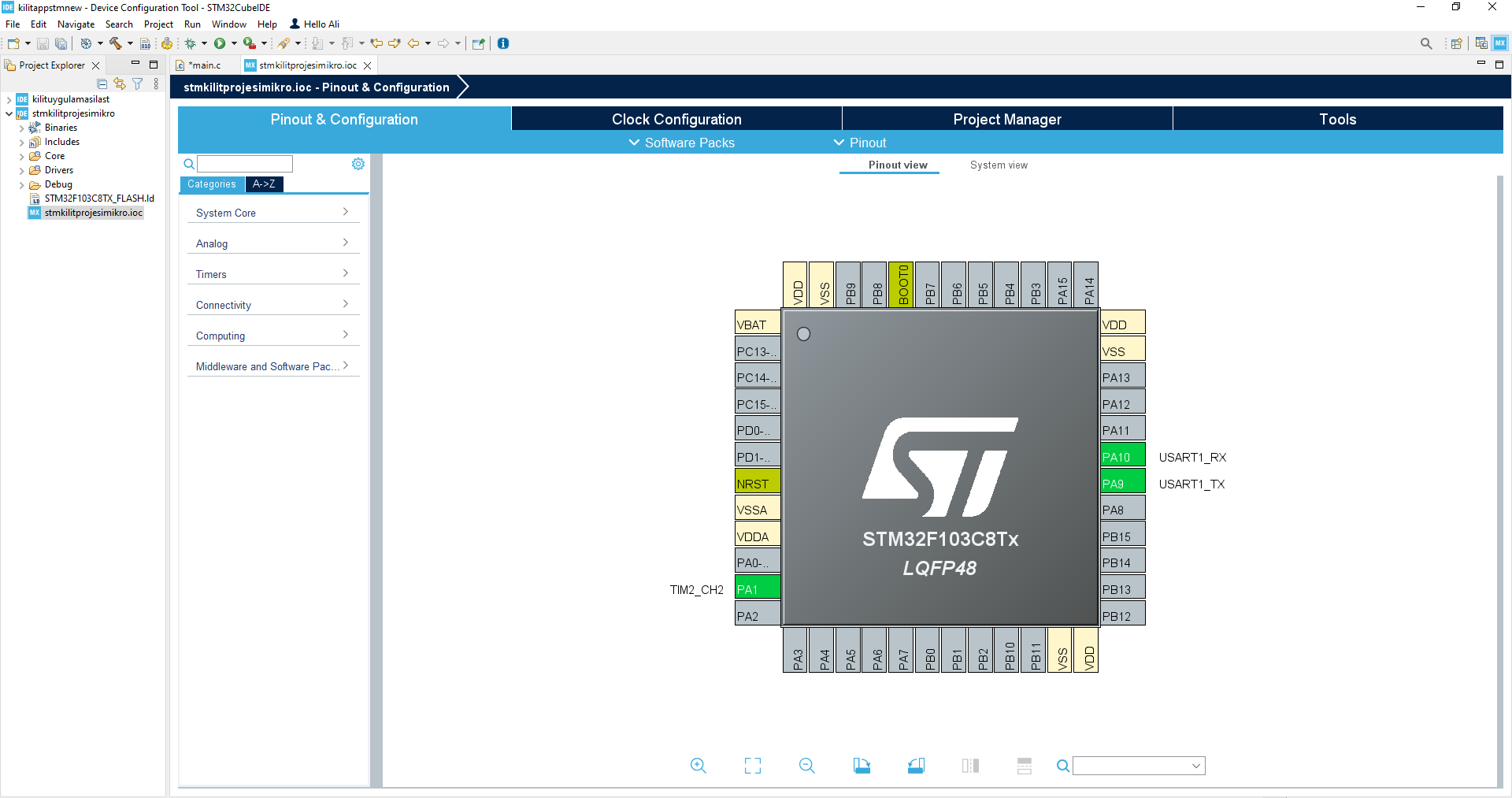
* **Kod Parçaları Flutter ve STM**

****









/\* USER CODE BEGIN Header \*/

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : main.c

\* @brief : Main program body

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @attention

\*

\* Copyright (c) 2023 STMicroelectronics.

\* All rights reserved.

\*

\* This software is licensed under terms that can be found in the LICENSE file

\* in the root directory of this software component.

\* If no LICENSE file comes with this software, it is provided AS-IS.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

/\* USER CODE END Header \*/

/\* Includes ------------------------------------------------------------------\*/

**#include** "main.h"

/\* Private includes ----------------------------------------------------------\*/

/\* USER CODE BEGIN Includes \*/

/\* USER CODE END Includes \*/

/\* Private typedef -----------------------------------------------------------\*/

/\* USER CODE BEGIN PTD \*/

/\* USER CODE END PTD \*/

/\* Private define ------------------------------------------------------------\*/

/\* USER CODE BEGIN PD \*/

/\* USER CODE END PD \*/

/\* Private macro -------------------------------------------------------------\*/

/\* USER CODE BEGIN PM \*/

/\* USER CODE END PM \*/

/\* Private variables ---------------------------------------------------------\*/

TIM\_HandleTypeDef htim2;

UART\_HandleTypeDef huart1;

/\* USER CODE BEGIN PV \*/

uint8\_t rxData;

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

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

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

**static** **void** **MX\_TIM2\_Init**(**void**);

**static** **void** **MX\_USART1\_UART\_Init**(**void**);

/\* USER CODE BEGIN PFP \*/

/\* USER CODE END PFP \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

/\* USER CODE END 0 \*/

/\*\*

\* @brief The application entry point.

\* @retval int

\*/

**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();

/\* USER CODE BEGIN Init \*/

/\* USER CODE END Init \*/

/\* Configure the system clock \*/

SystemClock\_Config();

/\* USER CODE BEGIN SysInit \*/

/\* USER CODE END SysInit \*/

/\* Initialize all configured peripherals \*/

MX\_GPIO\_Init();

MX\_TIM2\_Init();

MX\_USART1\_UART\_Init();

/\* USER CODE BEGIN 2 \*/

HAL\_TIM\_PWM\_Start(&htim2, TIM\_CHANNEL\_2); // Servo motor için timer PWM başlatma

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

**while** (1)

{

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

HAL\_UART\_Receive(&huart1, &rxData, **sizeof**(rxData), HAL\_MAX\_DELAY); //Bluetooth veri alımı

**if** (rxData == '2') { // Eğer 2 datası gelirse kilidi aç

\_\_HAL\_TIM\_SET\_COMPARE(&htim2,TIM\_CHANNEL\_2, 750); // SERVO Motor 90 derece döndür

//htim2.Instance->CCR1 = 100;

HAL\_Delay(500);

} **else** **if** (rxData == '3') { // Eğer 3 datası gelirse kilitle

\_\_HAL\_TIM\_SET\_COMPARE(&htim2,TIM\_CHANNEL\_2, 250); // SERVO Motor 0 dereceye geri döndür

//htim2.Instance->CCR1 = 25;

HAL\_Delay(500);

}

}

/\* USER CODE END 3 \*/

}

/\*\*

\* @brief System Clock Configuration

\* @retval None

\*/

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

{

RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};

RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};

/\*\* Initializes the RCC Oscillators according to the specified parameters

\* in the RCC\_OscInitTypeDef structure.

\*/

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSI;

RCC\_OscInitStruct.HSIState = RCC\_HSI\_ON;

RCC\_OscInitStruct.HSICalibrationValue = RCC\_HSICALIBRATION\_DEFAULT;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_NONE;

**if** (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != *HAL\_OK*)

{

Error\_Handler();

}

/\*\* Initializes the CPU, AHB and APB buses clocks

\*/

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK|RCC\_CLOCKTYPE\_SYSCLK

|RCC\_CLOCKTYPE\_PCLK1|RCC\_CLOCKTYPE\_PCLK2;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_HSI;

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

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV1;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;

**if** (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_0) != *HAL\_OK*)

{

Error\_Handler();

}

}

/\*\*

\* @brief TIM2 Initialization Function

\* @param None

\* @retval None

\*/

**static** **void** **MX\_TIM2\_Init**(**void**)

{

/\* USER CODE BEGIN TIM2\_Init 0 \*/

/\* USER CODE END TIM2\_Init 0 \*/

TIM\_ClockConfigTypeDef sClockSourceConfig = {0};

TIM\_MasterConfigTypeDef sMasterConfig = {0};

TIM\_OC\_InitTypeDef sConfigOC = {0};

/\* USER CODE BEGIN TIM2\_Init 1 \*/

/\* USER CODE END TIM2\_Init 1 \*/

htim2.Instance = TIM2;

htim2.Init.Prescaler = 15;

htim2.Init.CounterMode = TIM\_COUNTERMODE\_UP;

htim2.Init.Period = 9999;

htim2.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;

htim2.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE;

**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\_2) != *HAL\_OK*)

{

Error\_Handler();

}

/\* USER CODE BEGIN TIM2\_Init 2 \*/

/\* USER CODE END TIM2\_Init 2 \*/

HAL\_TIM\_MspPostInit(&htim2);

}

/\*\*

\* @brief USART1 Initialization Function

\* @param None

\* @retval None

\*/

**static** **void** **MX\_USART1\_UART\_Init**(**void**)

{

/\* USER CODE BEGIN USART1\_Init 0 \*/

/\* USER CODE END USART1\_Init 0 \*/

/\* USER CODE BEGIN USART1\_Init 1 \*/

/\* USER CODE END USART1\_Init 1 \*/

huart1.Instance = USART1;

huart1.Init.BaudRate = 9600;

huart1.Init.WordLength = UART\_WORDLENGTH\_8B;

huart1.Init.StopBits = UART\_STOPBITS\_1;

huart1.Init.Parity = UART\_PARITY\_NONE;

huart1.Init.Mode = UART\_MODE\_TX\_RX;

huart1.Init.HwFlowCtl = UART\_HWCONTROL\_NONE;

huart1.Init.OverSampling = UART\_OVERSAMPLING\_16;

**if** (HAL\_UART\_Init(&huart1) != *HAL\_OK*)

{

Error\_Handler();

}

/\* USER CODE BEGIN USART1\_Init 2 \*/

/\* USER CODE END USART1\_Init 2 \*/

}

/\*\*

\* @brief GPIO Initialization Function

\* @param None

\* @retval None

\*/

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

{

/\* GPIO Ports Clock Enable \*/

\_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();

}

/\* USER CODE BEGIN 4 \*/

/\* USER CODE END 4 \*/

/\*\*

\* @brief This function is executed in case of error occurrence.

\* @retval None

\*/

**void** **Error\_Handler**(**void**)

{

/\* USER CODE BEGIN Error\_Handler\_Debug \*/

/\* User can add his own implementation to report the HAL error return state \*/

\_\_disable\_irq();

**while** (1)

{

}

/\* USER CODE END Error\_Handler\_Debug \*/

}

**#ifdef** USE\_FULL\_ASSERT

/\*\*

\* @brief Reports the name of the source file and the source line number

\* where the assert\_param error has occurred.

\* @param file: pointer to the source file name

\* @param line: assert\_param error line source number

\* @retval None

\*/

**void** assert\_failed(uint8\_t \*file, uint32\_t line)

{

/\* USER CODE BEGIN 6 \*/

/\* User can add his own implementation to report the file name and line number,

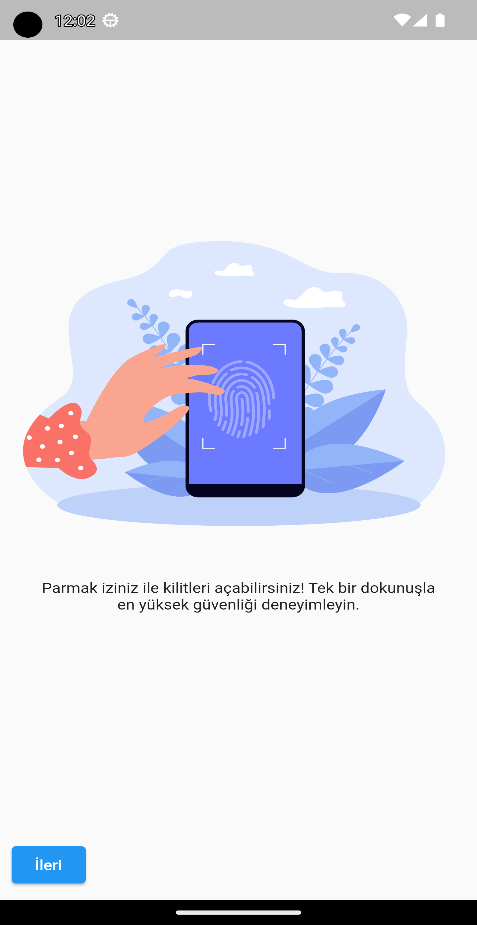
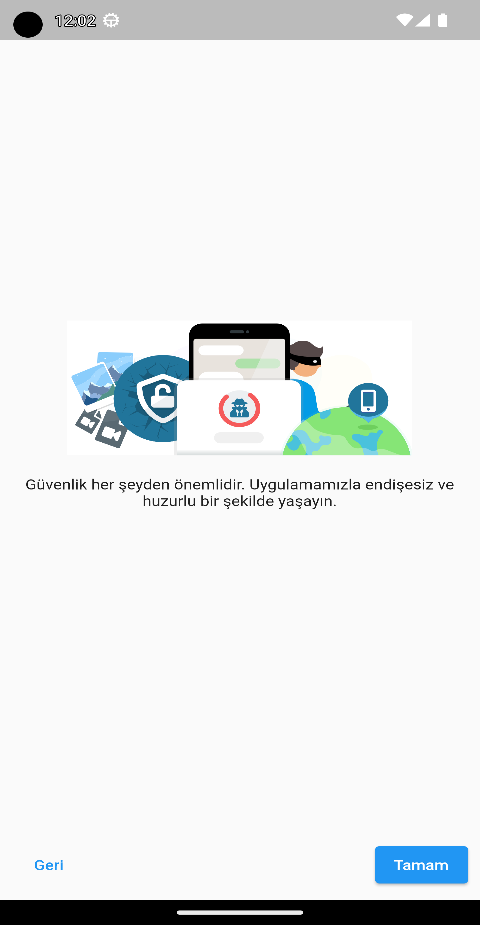
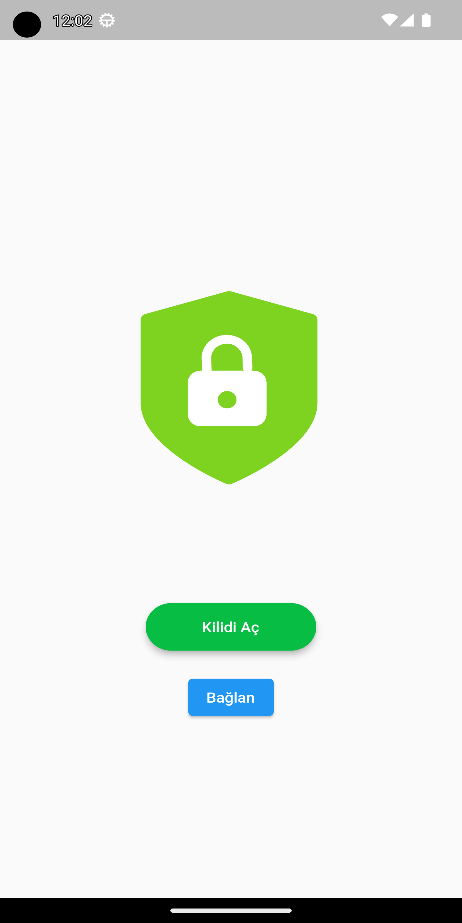
ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) \*/

/\* USER CODE END 6 \*/

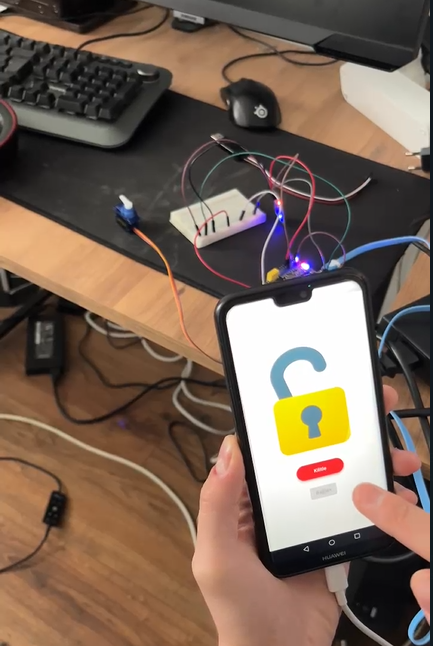
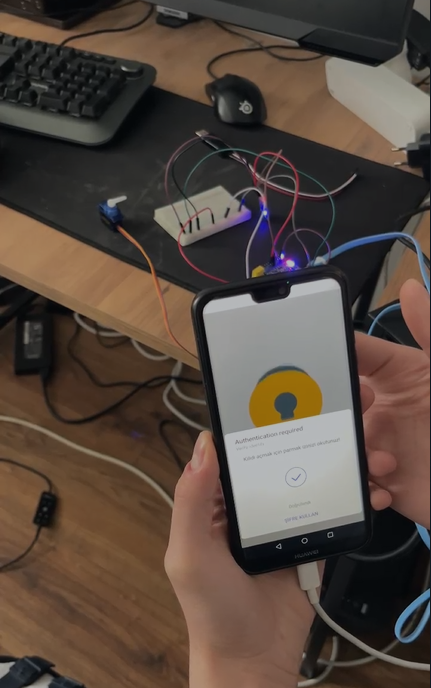
}

**#endif** /\* USE\_FULL\_ASSERT \*/

* **Uygulama Ekran Görüntüleri**



* **Test Süreçleri**

****