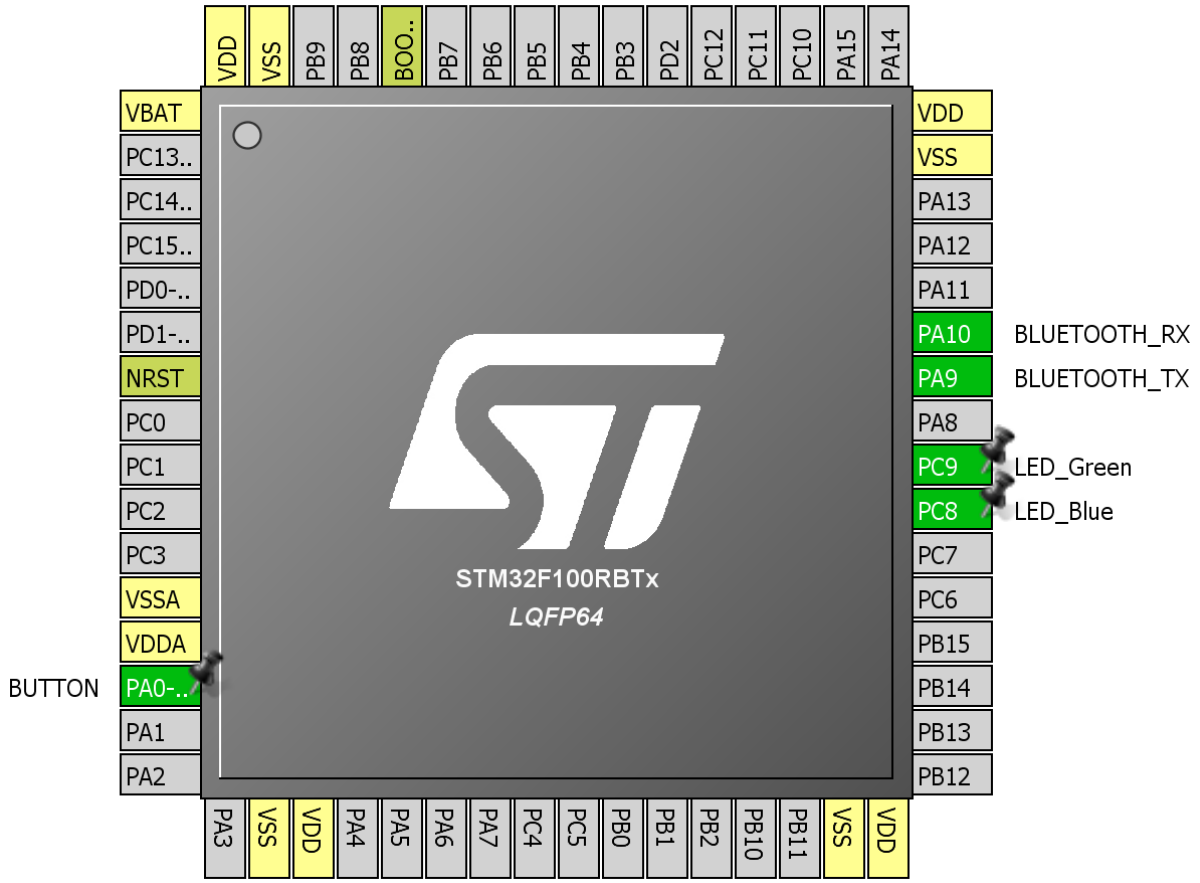
4. Dokumentacja

4.1. STM32

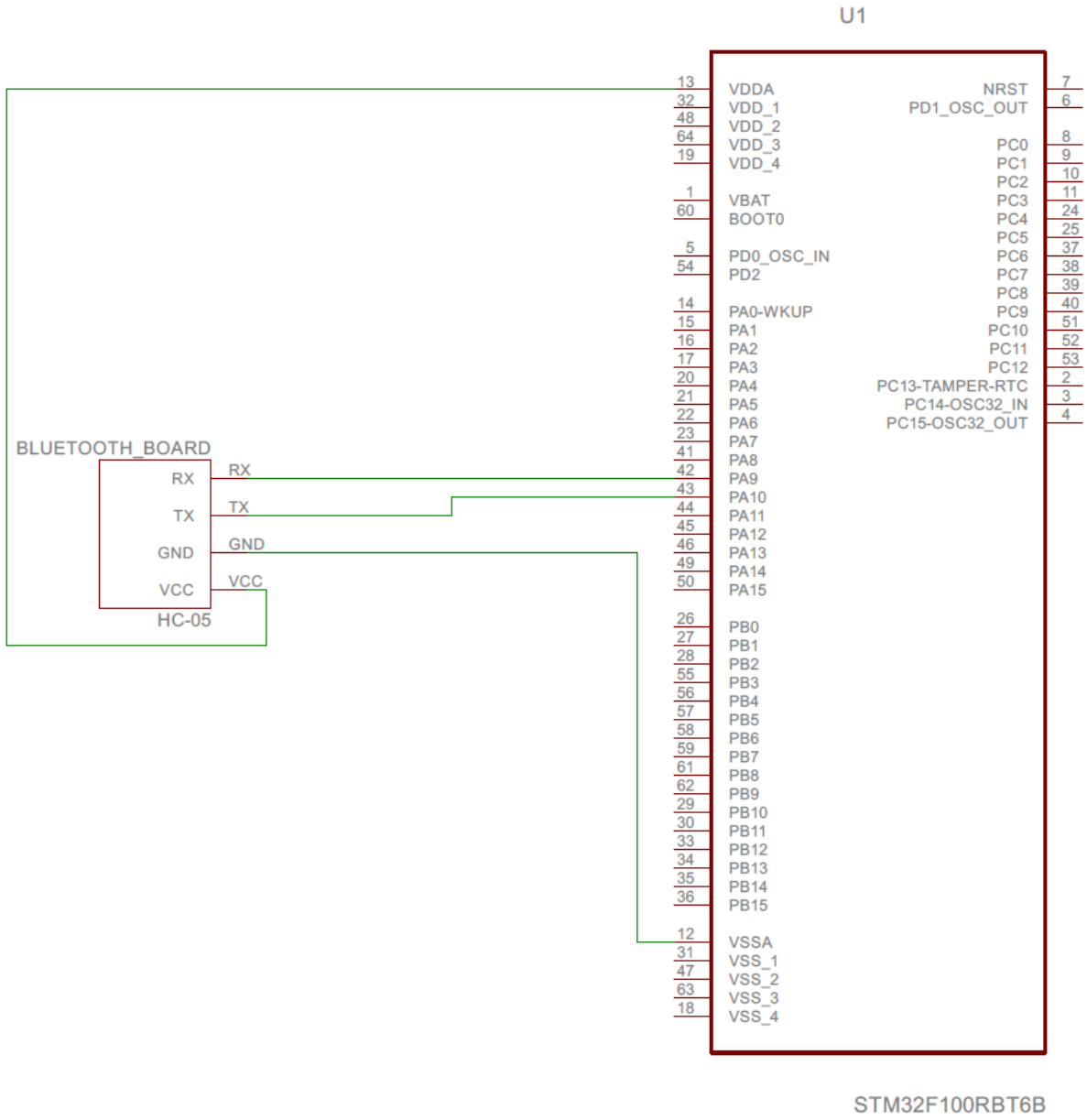
Program na płytkę STM32 F100RBT6B napisano przy użyciu oprogramowania STM32CubeMX. Na rysunku 4.1 przedstawiono zastosowane ustawienia płytki w tym właśnie programie.



Rys. 4.1. Schemat użytych wyprowadzeń procesora na płytce STM32

Wyprowadzenie PA9 oraz PA10 służy do obsługi interfejsu UART, co zostało wykorzystane do połączenia płytki z modułem Bluetooth HC-05v2. Dodatkowo w celu umożliwienia wizualizacji działania aplikacji desktopowej wraz z płytką wykorzystano wyprowadzenia PC8 (dioda niebieska), PC9 (dioda zielona) oraz PA0 (przycisk USER Button).

Schemat podłączenia płytki STM32 z modułem Bluetooth pokazano na rysunku 4.2.



Rys. 4.2. Schemat połączeniowy płytki STM32 z modułem HC-05v2

Na rysunkach 4.3 – 4.7 przedstawiono ustawienia z programu STM32CubeMX dotyczące portów GPIO, interfejsu UART, kontrolera DMA oraz kontrolera przerwań NVIC.

⊟ Basic Parameters	
Baud Rate	9600 Bits/s
Word Length	8 Bits (including Parity)
Parity	None
Stop Bits	1
⊟ Advanced Parameters	
Data Direction	Receive and Transmit
Over Sampling	16 Samples

Rys. 4.3. Ustawienia parametrów interfejsu UART

Interrupt Table	Enabled	Preemption Priority	Sub Priority
DMA1 channel4 global interrupt	☑	0	0
DMA1 channel5 global interrupt	☑	0	0
USART1 global interrupt	☑	0	0

Rys. 4.4. Ustawienia kontrolera przerwań NVIC

DMA Request	Channel	Direction	Priority
USART1_TX	DMA1 Channel 4	Memory To Peripheral	Low
USART1_RX	DMA1 Channel 5	Peripheral To Memory	Low

Rys. 4.5. Ustawienia kontrolera DMA

Pin Name	Signal on Pin	GPIO outpu...	GPIO mode	GPIO Pull-u...	Maximum o...	User Label	Modified
PA9	USART1_TX	n/a	Alternate Fu...	n/a	High	BLUETOOTH...	☑
PA10	USART1_RX	n/a	Input mode	No pull-up an...	n/a	BLUETOOTH...	☑

PA9 Configuration :

GPIO mode: Alternate Function Push Pull

Maximum output speed: High

User Label: BLUETOOTH_TX

Rys. 4.6. Ustawienia portu PA9 interfejsu UART

Pin Name	Signal on Pin	GPIO outpu...	GPIO mode	GPIO Pull-u...	Maximum o...	User Label	Modified
PA9	USART1_TX	n/a	Alternate Fu...	n/a	High	BLUETOOTH...	☑
PA10	USART1_RX	n/a	Input mode	No pull-up an...	n/a	BLUETOOTH...	☑

PA10 Configuration :

GPIO mode: Input mode

GPIO Pull-up/Pull-down: No pull-up and no pull-down

User Label: BLUETOOTH_RX

Rys. 4.7. Ustawienia portu PA10 interfejsu UART

4.1.1. Kod programu

Poniżej przedstawiono ważne fragmenty kodu źródłowego, który napisano z wykorzystaniem biblioteki HAL.

```
// - - - - - - - - - - - - - - - - ------------------ - - - - - - - - - - //
// - - - - - - - - - - - - - - - - | I N C L U D E S | - - - - - - - - - - //
// - - - - - - - - - - - - - - - - ------------------ - - - - - - - - - - //
// definicja bool
#include <stdbool.h>
#include <stdio.h>
#include <string.h>
// - - - - - - - - - - - - - - - - ------------------ - - - - - - - - - - //
// - - - - - - - - - - - - - - - - | I N C L U D E S | - - - - - - - - - - //
// - - - - - - - - - - - - - - - - ------------------ - - - - - - - - - - //
```

Listing 4.1. Definicja bibliotek

```
// - - - - - - - - - - - - - - - -------------------- - - - - - - - - - - //
// - - - - - - - - - - - - - - - | V A R I A B L E S | - - - - - - - - - - //
// - - - - - - - - - - - - - - - -------------------- - - - - - - - - - - //
//odebrany ciag
uint8_t receivedValue[5];

// wysylany ciag
uint8_t sentValue[2];

// oszukane wartosci akcelerometru (zakres od -1024 do 1024)
int16_t x, y, z;

// aktywnosc scian
uint8_t planeActive1, planeActive2, planeActive3, planeActive4,
planeActive5;

// poprzednia sciana na gorze (domyslnie sciana 0 [x = 0, y = 0, z = (-
1024-1024)])
uint8_t previousPlane;

// obecna sciana na gorze
uint8_t presentPlane;

// stan przycisku cofania
uint8_t buttonClicked;
// - - - - - - - - - - - - - - - -------------------- - - - - - - - - - - //
// - - - - - - - - - - - - - - - | V A R I A B L E S | - - - - - - - - - - //
// - - - - - - - - - - - - - - - -------------------- - - - - - - - - - - //
```

Listing 4.2. Definicja zmiennych

```
int main(void)
{

  /* USER CODE BEGIN 1 */

// - - - - - - - - - - - - - - - -------------------- - - - - - - - - - - //
// - - - - - - - - - - - - - - - | V A R I A B L E S | - - - - - - - - - - //
// - - - - - - - - - - - - - - - -------------------- - - - - - - - - - - //

      x = 0;
      y = 0;
      z = 0;

      planeActive1 = 0;
      planeActive2 = 0;
      planeActive3 = 0;
      planeActive4 = 0;
      planeActive5 = 0;

      previousPlane = 0;
      buttonClicked = 0;

// - - - - - - - - - - - - - - - -------------------- - - - - - - - - - - //
// - - - - - - - - - - - - - - - | V A R I A B L E S | - - - - - - - - - - //
// - - - - - - - - - - - - - - - -------------------- - - - - - - - - - - //

  /* USER CODE END 1 */
```

Listing 4.3. Inicjalizacja zmiennych w funkcji głównej programu

```
//- - - - - - - - - - - - - - - ------------------- - - - - - - - - - - //
//- - - - - - - - - - - - - - - | M A I N  L O O P | - - - - - - - - - - //
//- - - - - - - - - - - - - - - ------------------- - - - - - - - - - - //
while (1)
{
    //nasłuchuj transmisji Bluetooth
    if (HAL_UART_Receive_DMA(&huart1, receivedValue, 5)) {
// jesli pierwszy znak to "1" to zapal pierwsza sciane (ustaw jako aktywna)
        if (receivedValue[0] == 49) {
            HAL_GPIO_WritePin(LED_Blue_GPIO_Port, LED_Blue_Pin, SET);
            planeActive1 = 1;
        } else {
            HAL_GPIO_WritePin(LED_Blue_GPIO_Port, LED_Blue_Pin, RESET);
            planeActive1 = 0;
        }
// jesli drugi znak to "1" to zapal druga sciane (ustaw jako aktywna)
        if (receivedValue[1] == 49) {
            HAL_GPIO_WritePin(LED_Green_GPIO_Port, LED_Green_Pin, SET);
            planeActive2 = 1;
        } else {
            HAL_GPIO_WritePin(LED_Green_GPIO_Port, LED_Green_Pin, RESET);
            planeActive2 = 0;
        }
// gdyby bylo wiecej diod mozna wtedy zaprogramowac zapalanie wszystkich
scian
    }
```

Listing 4.4. Początek pętli głównej programu, obsługa transmisji Bluetooth wraz z aktywowaniem ścian (diod)

```
// wyznaczenie sciany znajdujacej sie na gorze na podstawie wartosci z
oszukanego akcelerometru
        // Sciana gorna
        if (x > -256 && x < 256) {
              if (y > -256 && y < 256) {
                    presentPlane = 0;
              }
        }

        // Sciana dolna
        if ((x >= -1024 && x < -768) || (x > 768 && x <= 1024)) {
              if (y > -256 && y < 256) {
                    if (planeActive1 == 1) {
                          presentPlane = 1;
                    }
              }
        }
        if (x > -256 && x < 256) {
              if ((y >= -1024 && y < -768) || (y > 768 && y <= 1024)) {
                    if (planeActive1 == 1) {
                          presentPlane = 1;
                    }
              }
        }

        // Sciana lewa
        if (x > 256 && x < 768) {
              if (z > -256 && z < 256) {
                    if (planeActive2 == 1) {
                          presentPlane = 2;
                    }
```

```
            }
        }
        if (x > -256 && x < 256) {
            if ((y > -768 && y < -256) || (y > 256 && y < 768)) {
                if (z > 256 && z < 768) {
                    if (planeActive2 == 1) {
                        presentPlane = 2;
                    }
                }
            }
        }

        // Sciana prawa
        if (x > -768 && x < -256) {
            if (z > -256 && z < 256) {
                if (planeActive3 == 1) {
                    presentPlane = 3;
                }
            }
        }
        if (x > -256 && x < 256) {
            if ((y > -768 && y < -256) || (y > 256 && y < 768)) {
                if (z > -768 && z < -256) {
                    if (planeActive3 == 1) {
                        presentPlane = 3;
                    }
                }
            }
        }

        // Sciana przednia
        if (y > 256 && y < 768) {
            if (z > -256 && z < 256) {
                if (planeActive4 == 1) {
                    presentPlane = 4;
                }
            }
        }

        // Sciana tylna
        if (y > -768 && y < -256) {
            if (z > -256 && z < 256) {
                if (planeActive5 == 1) {
                    presentPlane = 5;
                }
            }
        }
```

Listing 4.5. Fragment kodu realizujący wyznaczanie ściany będącej na górze kostki

```
// sprawdzenie stanu przycisku cofania
    if (HAL_GPIO_ReadPin(BUTTON_GPIO_Port, BUTTON_Pin) == GPIO_PIN_SET) {
        buttonClicked = 1;
    } else {
        buttonClicked = 0;
    }

    if (buttonClicked == 1) {
        // aktualizacja wysylanych danych
        sprintf(sentValue, "%d%d", presentPlane, buttonClicked);

        // wyslanie komunikatu do aplikacji
```

```
            HAL_UART_Transmit_DMA(&huart1, sentValue, 2);

            // prosty debounce
            HAL_Delay(500);
    }
```

Listing 4.6. Sprawdzenie stanu przycisku wraz z wysłaniem danych do PC

```
// jesli obecna sciana na gorze kostki jest rozna od poprzedniej
        if (previousPlane != presentPlane) {
              // aktualizacja wysylanych danych
              sprintf(sentValue, "%d%d", presentPlane, buttonClicked);

              // wyslanie komunikatu do aplikacji
              HAL_UART_Transmit_DMA(&huart1, sentValue, 2);

              // zapisanie aktualnej sciany jako poprzednia
              previousPlane = presentPlane;
        }
//- - - - - - - - - - - - - - - - -------------------- - - - - - - - - - - //
//- - - - - - - - - - - - - - - - | M A I N  L O O P | - - - - - - - - - - //
//- - - - - - - - - - - - - - - - -------------------- - - - - - - - - - - //
```

Listing 4.7. Wysłanie danych do PC w przypadku zmiany położenia kostki w przestrzeni i koniec pętli głównej

4.2. Aplikacja desktopowa

Po uruchomieniu programu na ekranie wyświetlają się kafelki przedstawiające poszczególne ściany kostki. Aktywną ścianę na urządzeniu symbolizują zapalające się diody. Po wybraniu interesującej nas ściany (zmiana położenia kostki w przestrzeni) uruchamiane jest odliczanie 3 sekundowe, które ogranicza możliwość przypadkowego wybrania określonej ściany. Jeżeli nie zmieniono ściany w ciągu tego czasu, następuje przejście do kolejnego poziomu menu aplikacji. Użytkownik ma możliwość powrotu do wcześniejszego ekranu za pomocą przycisku (przyciśnięcie dowolnej ściany). Aby komunikacja z urządzeniem była możliwa należy przed uruchomieniem aplikacji włączyć obsługę stosu Bluetooth na komputerze oraz uruchomić płytkę STM32. Wybranie ściany wiąże się ze zmianą ekranu aplikacji, a co za tym idzie zmianą programu głównego kostki (zmiana aktywnych ścian).

Poniżej przedstawiono najważniejsze fragmenty źródłowe kodu programu wraz z komentarzami.

```
public class Bluetooth {
    //region Variables

    private static StreamConnection streamConnection;
    private static OutputStream os;
    private static InputStream is;

    //endregion

    public static void connect() {
        // connect to device

        // device bluetooth address
        String deviceURL =
"btspp://98D33130854A:1;authenticate=false;encrypt=false;master=false";
```

```
        try {
            streamConnection = (StreamConnection)
Connector.open(deviceURL);
            os = streamConnection.openOutputStream();
            is = streamConnection.openInputStream();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static void disconnect() {
        // disconnect from device
        try {
            os.close();
            is.close();
            streamConnection.close();
        } catch (Exception e) {
           e.printStackTrace();
        }
    }

    public static void sendData(String activePlanes) {
        // send active planes configuration
        try {
            os.write(activePlanes.getBytes());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static void getData() {
        // get plane on top and button status
        try {
            DataInputStream din = new DataInputStream(is);
            byte[] buffer = new byte[2];
            din.readFully(buffer);
            Configuration.receivedData = buffer;
        } catch (Exception e) {
            e.printStackTrace();
        }

        UI.changeUIState();
    }
}
```

Listing 4.8. Klasa Bluetooth realizująca połączenie aplikacji z modułem HC-05v2 dołączonym do płytki STM32

```
public class UI {
    public static void changeUIState() {
        // check button status from device
        switch (Configuration.receivedData[1]) {
            case 49:
                // setting previous scene when button clicked (always main
scene)
                if (Configuration.window.getScene() == Scenes.mainScene) {
                    Bluetooth.disconnect();
                    System.exit(0);
                } else {
                    Tile.deselectPanel(Scenes.mainScene,
Configuration.activePanel);
                    Configuration.activePanel = MainScene.panel_0_tile;
```

```
                    Platform.runLater(() ->
Scenes.setScene(MainScene.panel_0_tile));
                }
                break;
            default:
                break;
        }
        // check top plane from device
        switch (Configuration.receivedData[0]) {
            case 49:
                if (Configuration.window.getScene() == Scenes.mainScene) {
                    if (Configuration.activePanel !=
MainScene.panel_1_tile) {
                        Tile.deselectPanel(Scenes.mainScene,
Configuration.activePanel);
                        Tile.selectPanel(MainScene.panel_1_tile,
Scenes.scene1);
                        Configuration.activePanel = MainScene.panel_1_tile;
                        TimerCounter.stopCounting();
                        TimerCounter.startCounting(MainScene.panel_1_tile);
                    }
                }
                // same for the rest scenes
                break;
            case 50:
                if (Configuration.window.getScene() == Scenes.mainScene) {
                    Tile.deselectPanel(Scenes.mainScene,
Configuration.activePanel);
                    Tile.selectPanel(MainScene.panel_2_tile,
Scenes.scene1);
                    Configuration.activePanel = MainScene.panel_2_tile;
                    TimerCounter.stopCounting();
                    TimerCounter.startCounting(MainScene.panel_2_tile);
                }
                // same for the rest scenes
                break;
            case 51:
                if (Configuration.window.getScene() == Scenes.mainScene) {
                    Tile.deselectPanel(Scenes.mainScene,
Configuration.activePanel);
                    Tile.selectPanel(MainScene.panel_3_tile,
Scenes.scene1);
                    Configuration.activePanel = MainScene.panel_3_tile;
                    TimerCounter.stopCounting();
                    TimerCounter.startCounting(MainScene.panel_3_tile);
                }
                // same for the rest scenes
                break;
            case 52:
                if (Configuration.window.getScene() == Scenes.mainScene) {
                    Tile.deselectPanel(Scenes.mainScene,
Configuration.activePanel);
                    Tile.selectPanel(MainScene.panel_4_tile,
Scenes.scene1);
                    Configuration.activePanel = MainScene.panel_4_tile;
                    TimerCounter.stopCounting();
                    TimerCounter.startCounting(MainScene.panel_4_tile);
                }
                // same for the rest scenes
                break;
            case 53:
```

```
                if (Configuration.window.getScene() == Scenes.mainScene) {
                    Tile.deselectPanel(Scenes.mainScene,
Configuration.activePanel);
                    Tile.selectPanel(MainScene.panel_5_tile,
Scenes.scene1);
                    Configuration.activePanel = MainScene.panel_5_tile;
                    TimerCounter.stopCounting();
                    TimerCounter.startCounting(MainScene.panel_5_tile);
                }
                // same for the rest scenes
                break;
            default:
                Tile.deselectPanel(Scenes.mainScene,
Configuration.activePanel);
                Configuration.activePanel = MainScene.panel_0_tile;
                TimerCounter.stopCounting();
                break;
        }
    }
}
```

Listing 4.9. Klasa UI realizująca zmiany scen ze względu na odebrane dane z urządzenia

```
public class TimerCounter {

    private static float i = 0.0f;
    private static Timer timer;

     public static void startCounting(Tile tile) {
        timer = new Timer();
        TimerTask timerTask = new TimerTask() {
            @Override
            public void run() {
                Configuration.isTimerRunning = true;
                i = i + 0.5f;
                if (i >= 3) {
                    stopCounting();
                    i = 0.0f;
                    Platform.runLater(() -> Scenes.setScene(tile));
                }
            }
        };
        timer.schedule(timerTask,500,500);
    }

    public static void stopCounting() {
        if (Configuration.isTimerRunning) {
            timer.cancel();
            timer.purge();
            i = 0.0f;
            Configuration.isTimerRunning = false;
        }
    }
}
```

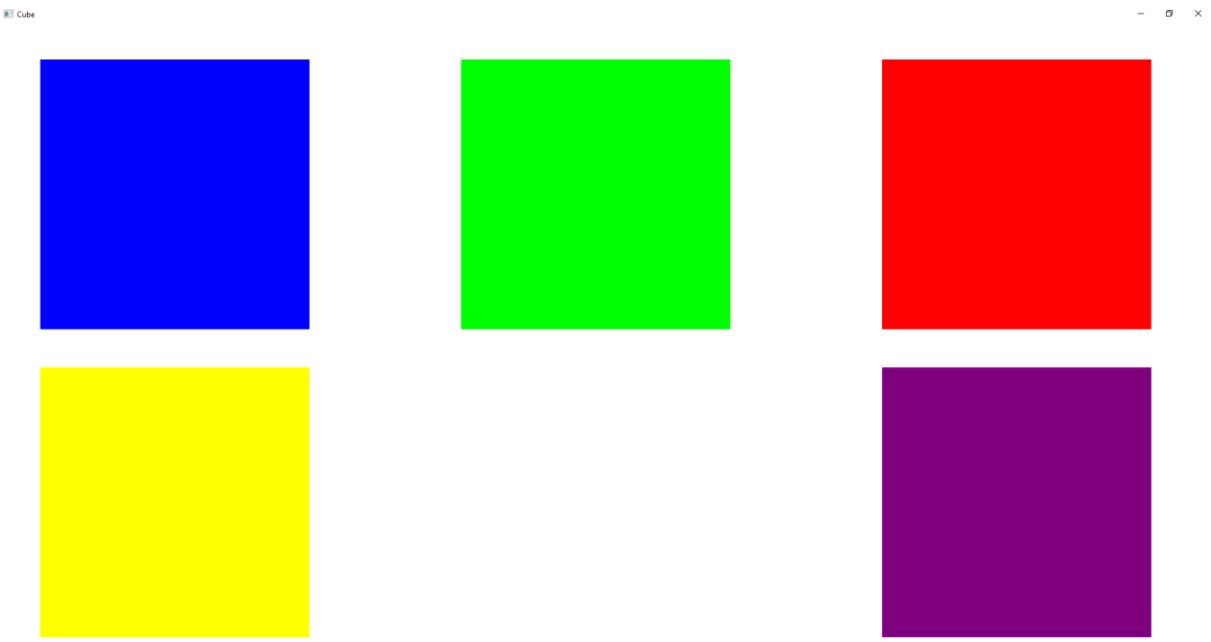
Listing 4.10. Klasa TimerCounter realizująca odliczanie po wybraniu ściany

```
public class Main extends Application {

    public static void main(String[] args) {
        // initialize configuration data
        new Configuration();
        // initialize all scenes data
```

```
        new MainScene();
        new Scene1();
        new Scene2();
        new Scene3();
        new Scene4();
        new Scene5();
        // initialize Bluetooth connection configuration
        new Bluetooth();

        // connect to Bluetooth device
        try {
            Bluetooth.connect();
        } catch (Exception e) {
            e.printStackTrace();
        }

        // send to Bluetooth device planes configuration (all planes
active)
        Bluetooth.sendData("11111");

        // initialize Bluetooth listening thread
        Configuration.bluetoothThread = new BluetoothThread();
        Configuration.bluetoothThread.start();

        // launch application
        launch(args);
    }

    @Override
    public void start(Stage primaryStage) throws Exception {
        // set primary stage
        Configuration.window = primaryStage;
        // set application title
        Configuration.window.setTitle("Cube");
        // set full screen to false
        Configuration.window.setFullScreen(false);
        // set window to maximized
        Configuration.window.setMaximized(true);

        // initialize scenes
        Scenes.mainScene = new Scene(MainScene.layout,
Configuration.screenSize.getWidth(), Configuration.screenSize.getHeight());
        Scenes.scene1 = new Scene(Scene1.layout,
Configuration.screenSize.getWidth(), Configuration.screenSize.getHeight());
        Scenes.scene2 = new Scene(Scene2.layout,
Configuration.screenSize.getWidth(), Configuration.screenSize.getHeight());
        Scenes.scene3 = new Scene(Scene3.layout,
Configuration.screenSize.getWidth(), Configuration.screenSize.getHeight());
        Scenes.scene4 = new Scene(Scene4.layout,
Configuration.screenSize.getWidth(), Configuration.screenSize.getHeight());
        Scenes.scene5 = new Scene(Scene5.layout,
Configuration.screenSize.getWidth(), Configuration.screenSize.getHeight());

        // set first scene
        Configuration.window.setScene(Scenes.mainScene);
        // show window
        Configuration.window.show();
    }
}
```

Listing 4.11. Klasa Main realizująca metodę główna aplikacji

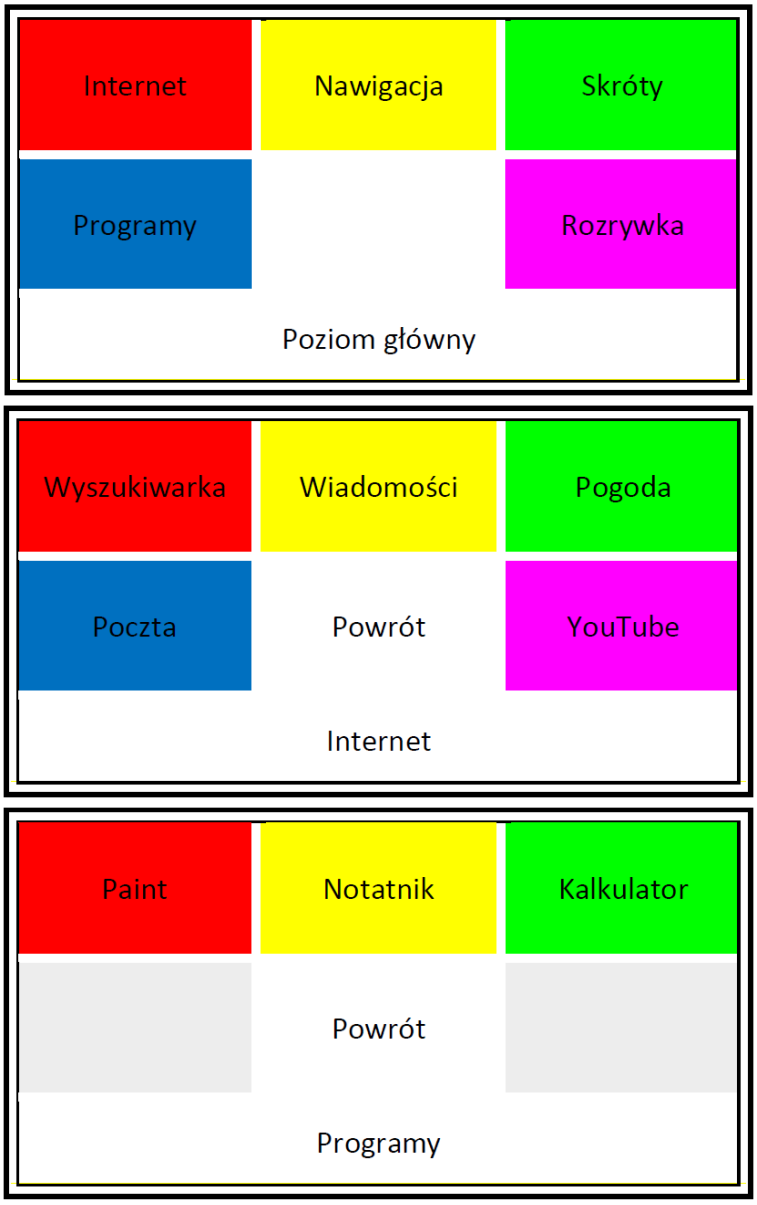
Na rysunku 4.8 przedstawiono wygląd aplikacji, a na rysunkach 4.9 i tabeli 4.1 możliwe funkcje aplikacji wykonywane po wybraniu odpowiednich ścian.



Rys. 4.8. Wygląd aplikacji

Tab. 4.1. Przykładowa możliwość realizacji funkcji po wybraniu odpowiedniej ściany

Poziom 1	Internet	Nawigacja w przeglądarce	Skróty	Programy	Rozrywka
Poziom2	Wyszukiwarka	Page UP	Znajdź [Ctrl+F]	Paint	Zdjęcia
	Wiadomości	Page Down	Drukuj [Ctrl+P]	Notatnik	Filmy
	Pogoda	Cofnij	Kopiuj [Ctrl+C]	Kalkulator	Muzyka
	Poczta	Dalej	Wklej [Ctrl+V]		Ipla
	YouTube	Odśwież	Zamknij [Alt+F4]		Gry



Rys. 4.9. Przykładowe możliwości realizacji funkcji po wybraniu odpowiedniej ściany

Tak wykonana aplikacja może posłużyć osobom z niedowładami kończyn górnych do obsługi komputera lub innego urządzenia elektronicznego (np. smartphone). Ma ona na względzie spowolnioną i ograniczoną ruchliwość docelowego użytkownika oraz posiada przejrzysty i czytelny interfejs graficzny odpowiedni dla osób w każdym wieku.

Ciekawą możliwością rozwoju mogłoby być rozszerzenie możliwości kostki do sterowania mieszkaniem (rozwiązania Smart Home). Poprzez sterowanie inteligentnym domem można rozumieć sterowanie oświetleniem, roletami okiennymi czy też ogrzewaniem.