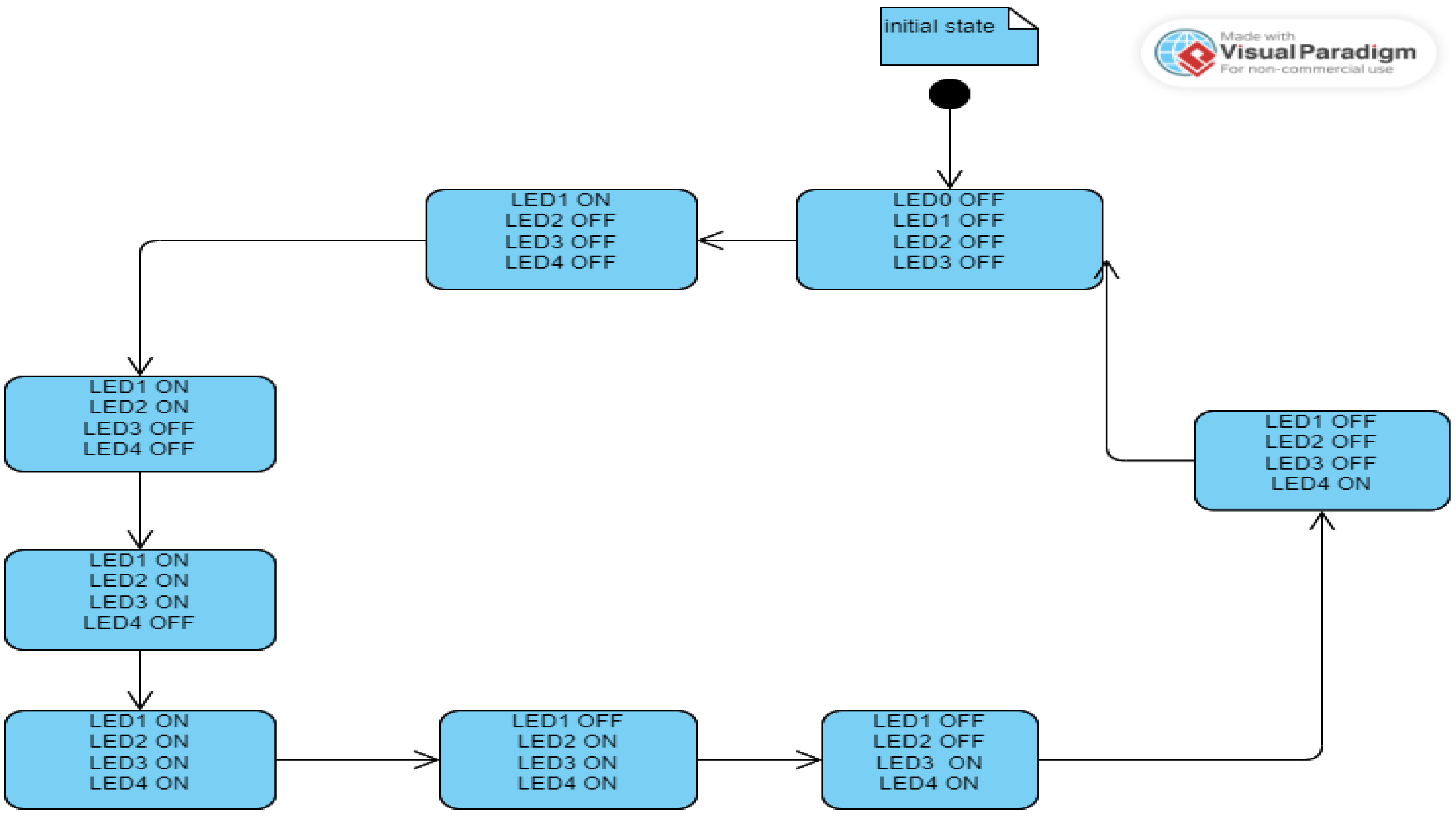
Project Title: Led Sequence V 1.0

Name: Basel Nagy

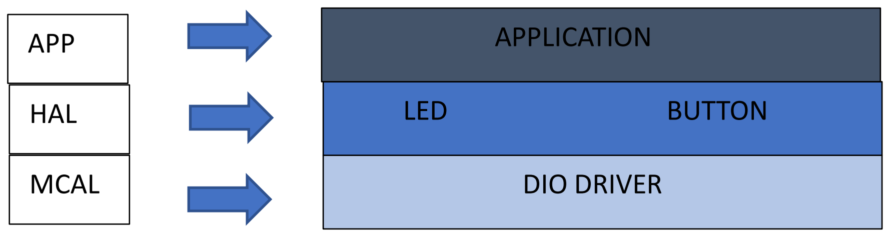
Description:

1. ***Hardware Requirements***
   1. *Four LEDs (LED0, LED1, LED2, LED3)*
   2. One button (BUTTON0)
2. ***Software Requirements***
   1. Initially, all LEDs are OFF
   2. Once BUTTON0 is pressed, LED0 will be ON
   3. Each press further will make another LED is ON
   4. At the fifth press, LED0 will changed to be OFF
   5. Each press further will make only one LED is OFF
   6. This will be repeated forever
   7. The sequence is described below
      1. Initially (OFF, OFF, OFF, OFF)
      2. Press 1 (ON, OFF, OFF, OFF)
      3. Press 2 (ON, ON, OFF, OFF)
      4. Press 3 (ON, ON, ON, OFF)
      5. Press 4 (ON, ON, ON, ON)
      6. Press 5 (OFF, ON, ON, ON)
      7. Press 6 (OFF, OFF, ON, ON)
      8. Press 7 (OFF, OFF, OFF, ON)
      9. Press 8 (OFF, OFF, OFF, OFF)
      10. Press 9 (ON, OFF, OFF, OFF)

State machine:


initial state
LED0 OFF LED1 OFF LED2 OFF LED3 OFF
LED1 ON LED2 OFF LED3 OFF LED4 OFF
LED1 ON LED2 ON LED3 OFF LED4 OFF
LED1 ON LED2 ON LED3 ON LED4 OFF
LED1 ON LED2 ON LED3 ON LED4 ON
LED1 OFF LED2 ON LED3 ON LED4 ON
LED1 OFF LED2 OFF LED3 ON LED4 ON
LED1 OFF LED2 OFF LED3 OFF LED4 ON
Made with Visual Paradigm For non-commercial use

Layered architecture:



Project Modules APIs:

DIO DRIVER:

```
/*typedef*/
typedef enum DIO_PORTS
{
        porta, portb, portc, portd
} DIO_PORTS;
typedef enum DIO_PINS
{
        pin0, pin1, pin2, pin3, pin4, pin5, pin6, pin7
} DIO_PINS;
typedef enum PIN_DIRECTION
{
        INPUT,
        OUTPUT
} PIN_DIRECTION;

typedef enum PIN_STATE
{
        LOW,
        HIGH
 } PIN_STATE;

/*****************************************    APIs PROTOTYPES    **************************************************************************/
STD_return DIO_INIT (DIO_PORTS port, DIO_PINS pin, PIN_DIRECTION direction);
STD_return DIO_WRITE_PIN (DIO_PORTS port, DIO_PINS pin, PIN_STATE state);
STD_return DIO_READ_PIN (DIO_PORTS port, DIO_PINS pin, uint8_t* vale);
```

LED APIs:

```
typedef struct LED
{
  DIO_PORTS port;
DIO_PINS pin;
} LED;
```

```
/****************************************    APIs PROTOTYPES     *****************************************************************/
STD_return LED_INIT (LED* led);
STD_return LED_ON (DIO_PORTS, DIO_PINS);
STD_return LED_OFF (DIO_PORTS,DIO_PINS);
```

BUTTON APIs:

```
typedef  enum EN_BUTTON_STATE_t {PUSHED,RELEASED} EN_BUTTON_STATE_t;
typedef struct BUTTON
{
DIO_PORTS port;
DIO_PINS pin;
} BUTTON;
/****************************************    APIs PROTOTYPES     ************************************************************ **********/
STD_return BUTTON_INIT (BUTTON*  button);
STD_return BUTTON_READ (DIO_PORTS port, DIO_PINS pin, EN_BUTTON_STATE_t* value);
```