Hanoi University of Science and Technology

School of Information and Communication Technology

IT3280E - Assembly Language and Computer Architecture Lab

Lab 13. Assembly Programming in ESP32-C3 – using Wokwi Simulation

Student Name: Nguyen Duc Anh
Student ID: 20235890

1 Assignment 1:

1.1 Source Code:

```
.global init

.eqv GPIO_ENABLE_REG, 0x60004020 # Register to config GPIO pins as I/O method
.eqv GPIO_OUT_W1TS_REG, 0x60004008 # Register to set GPIO pins

.text
init:
    li a1, GPIO_ENABLE_REG # Load register address to setup GPIO0
    li a2, 0x01 # Load the mask 0x01 for register GPIO_ENABLE_REG
    sw a2, 0(a1)

    li a1, GPIO_OUT_W1TS_REG # Load register to output GPIO0
    li a2, 0x01 # Load the mask 0x01 for register GPIO_OUT_W1TS_REG
    sw a2, 0(a1)
```

1.2 Explaination:

1.2.1 Register Configure

- `.eqv GPIO ENABLE REG, 0x60004020`: Register that enables output functionality for GPIO pins, Bits 0 to 21 correspond to GPIO0 through GPIO21, A bit value of 1 configures the corresponding GPIO pin as an output.
- `.eqv GPIO_OUT_W1TS_REG, 0x60004008`: Register for setting bits in the GPIO_ENABLE_REG register, a bit value of 1 sets the corresponding bit in GPIO_ENABLE_REG, leaving other bits unchanged.

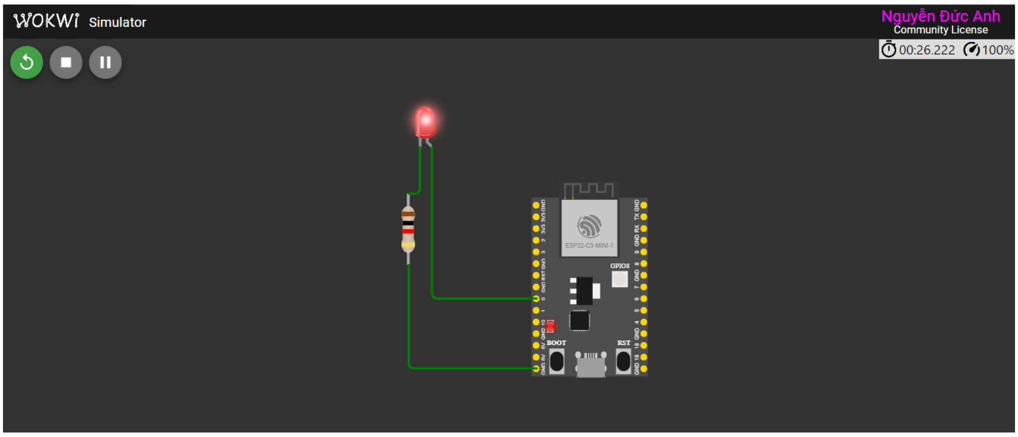
1.2.2 Enable GPIO0 pin for I/O method

- `li a1, GPIO_ENABLE_REG`: Load register address to setup GPIO0.
- `li a2, 0x01`: Load the mask 0x01 for register GPIO_ENABLE_REG.
- `sw a2, 0(a1)`: Store the bit value to the address.

1.2.3 Output signal for GPIO0 pin

- `li a1, GPIO_OUT_W1TS_REG`: Load register for output GPIO0.
- `li a2, 0x01`: Load the mask 0x01 for register GPIO_OUT W1TS_REG.
- `sw a2, 0(a1)`: Store the bit value to the address.

1.3 Result:



2 Assignment 2:

2.1 Source Code:

```
.global init

.eqv GPIO_ENABLE_REG, 0x60004020
.eqv GPIO_OUT_W1TS_REG, 0x60004008
.eqv GPIO_OUT_W1TC_REG, 0x6000400C

.text
init:
# Enable GPIO0 for output
    li a1, GPIO_ENABLE_REG
    li a2, 0x01
    sw a2, 0(a1)

main_loop:
    # Set GPIO0 to HIGH
    li a1, GPIO_OUT_W1TS_REG
    li a2, 0x01
    sw a2, 0(a1)
    call delay_asm  # Delay

    # Clear GPIO0 to LOW
    li a1, GPIO_OUT_W1TC_REG
    li a2, 0x01
    sw a2, 0(a1)
    call delay_asm  # Delay

    j main_loop # Loop

delay_asm:
    li a3, 0     # counter
    li a4, 5000000  # wait time (counting time)

loop_delay:
    addi a3, a3, 1
    blt a3, a4, loop_delay
    ret
```

2.2 Explaination:

2.2.1 Register Configure

- `.eqv GPIO_ENABLE_REG, 0x60004020`: Register that enables output functionality for GPIO pins, Bits 0 to 21 correspond to GPIO0 through GPIO21, A bit value of 1 configures the corresponding GPIO pin as an output.

- `.eqv GPIO_OUT_W1TS_REG, 0x60004008`: Register for setting bits in the GPIO_ENABLE_REG register, a bit value of 1 sets the corresponding bit in GPIO_ENABLE_REG, leaving other bits unchanged.

- `.eqv GPIO_OUT_W1TC_REG, 0x6000400C`: Register for clearing bits in the GPIO_ENABLE_REG register, a bit value of 1 clears the corresponding bit in GPIO_ENABLE_REG, leaving other bits unchanged.

2.2.2 Enable GPIO0 for output

```
    li a1, GPIO_ENABLE_REG
    li a2, 0x01
    sw a2, 0(a1)
```

- `li a1, GPIO_ENABLE_REG`: Load register address to setup GPIO0.
- `li a2, 0x01`: Load the mask 0x01 for register GPIO_ENABLE_REG.
- `sw a2, 0(a1)`: Store the bit value to the address.

2.2.3 Set GPIO0 pin to High

```
    li a1, GPIO_OUT_W1TS_REG
    li a2, 0x01
    sw a2, 0(a1)
    call delay_asm  # Delay
```

- `li a1, GPIO_OUT_W1TS_REG`: Load register for output GPIO0.
- `li a2, 0x01`: Load the mask 0x01 for register GPIO_OUT W1TS_REG.
- `sw a2, 0(a1)`: Store the bit value to the address.
- `call delay_asm`: Call for delaying.

2.2.4 Clear GPIO0 pin to Low

```
    li a1, GPIO_OUT_W1TC_REG
    li a2, 0x01
    sw a2, 0(a1)
    call delay_asm  # Delay
```

- `li a1, GPIO_OUT_W1TC_REG`: Load register for output GPIO0.
- `li a2, 0x01`: Load the mask 0x01 for register GPIO_OUT W1TS_REG.
- `sw a2, 0(a1)`: Store the bit value to the address.
- `call delay_asm`: Call for delaying.

2.2.5 Blink LED loop

- `j main_loop`: Jump back for looping.

2.2.6 Delay Function

```
delay_asm:
    li a3, 0     # counter
    li a4, 5000000  # wait time (counting time)
```

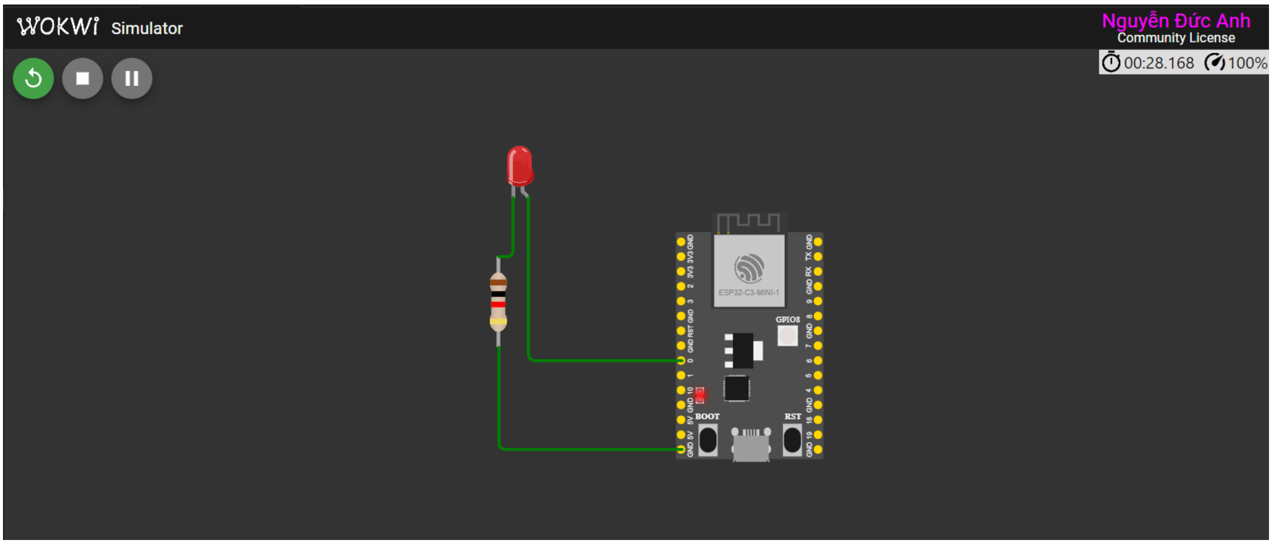
- `li a3, 0`: Initialize the index value.
- `li a4, 5000000`: Initialize the time for stop blinking.

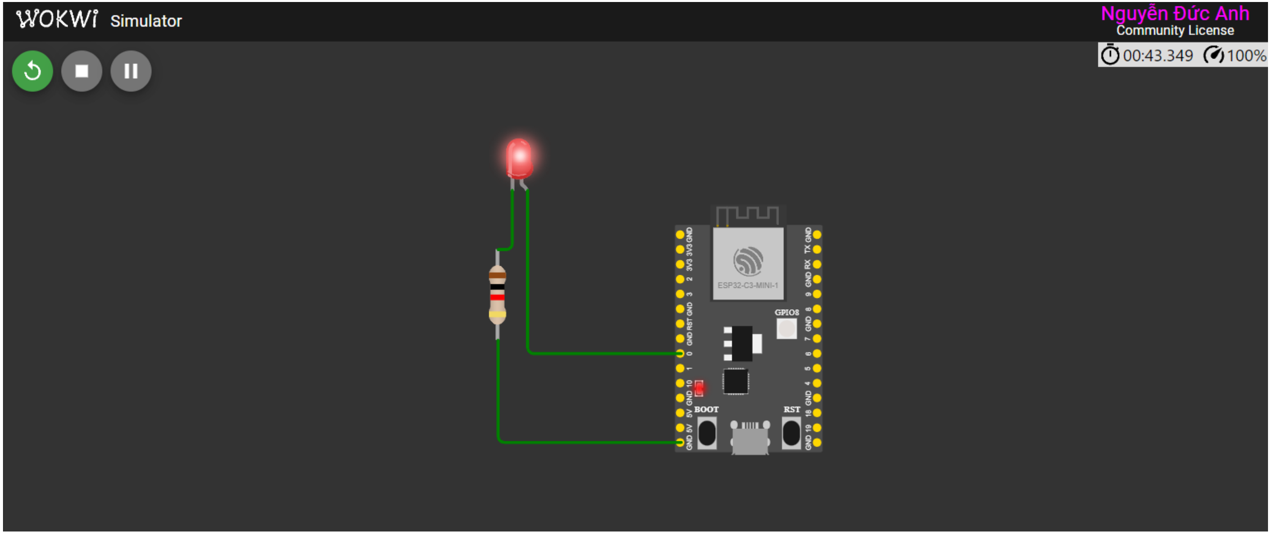
2.2.7 Delay Looping

```
loop_delay:
    addi a3, a3, 1
    blt a3, a4, loop_delay
    ret
```

- `addi a3, a3, 1`: Increase the index.
- `blt a3, a4, loop_delay`: Check if index smaller than time, continue looping.

2.3 Result:





3 Assignment 3:

3.1 Source Code:

```
.global init

.eqv GPIO_ENABLE_REG, 0x60004020 # Enable outpu GPIO
.eqv GPIO_OUT_REG, 0x60004004 # setup output

.eqv IO_MUX_GPIO4_REG, 0x60009014 # Setup function GPIO4
.eqv IO_MUX_GPIO5_REG, 0x60009018 # Setup function GPIO5
.eqv IO_MUX_GPIO6_REG, 0x6000901C # Setup function GPIO6
.eqv IO_MUX_GPIO7_REG, 0x60009020 # Setup function GPIO7

.text
init:
    li a1, GPIO_ENABLE_REG
    li a2, 0xFF # output from GPIO0 to GPIO7 (8 bits)
    sw a2, 0(a1) # setupt bits in GPIO_ENABLE_REG
    # setup function in GPIO4, GPIO5, GPIO6, GPIO7
    # in default, they are used for SPI function
    # we need to change to GPIO function

    li a2, 0x1000

    li a1, IO_MUX_GPIO4_REG
    sw a2, 0(a1)

    li a1, IO_MUX_GPIO5_REG
    sw a2, 0(a1)

    li a1, IO_MUX_GPIO6_REG
    sw a2, 0(a1)

    li a1, IO_MUX_GPIO7_REG
    sw a2, 0(a1)

    # a1 contains the address of state register GPIO
    li a1, GPIO_OUT_REG
    li a2, 0x40
    sw a2, 0(a1) # Output to GPIO
```

3.2 Explaination:

3.2.1 Register Configure

- `.eqv GPIO_ENABLE_REG, 0x60004020`: Register that enables output functionality for GPIO pins, Bits 0 to 21 correspond to GPIO0 through GPIO21, A bit value of 1 configures the corresponding GPIO pin as an output.

- `.eqv GPIO_OUT_REG, 0x60004004`: Register for configures output values for GPIO pins, bits 0 to 21 correspond to GPIO0 through GPIO21, a bit value of 1/0 sets the corresponding GPIO pin to a high/low logic level.

- `.eqv IO_MUX_GPIO4_REG, 0X60009014`: Configures the functionality of GPIO pins (GPIO0 to GPIO21), these registers are used to select functions and configure GPIO pin operations.

- `.eqv IO_MUX_GPIO5_REG, 0X60009018`: Configures the functionality of GPIO pins (GPIO0 to GPIO21), these registers are used to select functions and configure GPIO pin operations.

- `.eqv IO_MUX_GPIO6_REG, 0X6000901C`: Configures the functionality of GPIO pins (GPIO0 to GPIO21), these registers are used to select functions and configure GPIO pin operations.

- `.eqv IO_MUX_GPIO7_REG, 0X60009020`: Configures the functionality of GPIO pins (GPIO0 to GPIO21), these registers are used to select functions and configure GPIO pin operations.

3.2.2 Enable and Initialize the output for GPIO0 to GPIO7

```
    init:
        li a1, GPIO_ENABLE_REG
        li a2, 0xFF # output from GPIO0 to GPIO7 (8 bits)
        sw a2, 0(a1) # setupt bits in GPIO_ENABLE_REG
```

- `li a1, GPIO_ENABLE_REG`: Load register address to setup GPIO0.

- `li a2, 0xFF`: Load the mask 0xFF for register GPIO_ENABLE_REG.

- `sw a2, 0(a1)`: Store the mask value to the address.

3.2.3 Setup the value of IO_MUX_GPIOn_MCU_SEL on bit 12

```
    # setup function in GPIO4, GPIO5, GPIO6, GPIO7
    # in default, they are used for SPI function
    # we need to change to GPIO function

    li a2, 0x1000
```

- `li a2, 0x1000`: Load value 1 bit to bit 12.

3.2.4 Initialize the value of IO_MUX_GPIOn_MCU_SEL on bit 12 to GPIO4 to GPIO7 pin manually

```
    li a1, IO_MUX_GPIO4_REG
    sw a2, 0(a1)

    li a1, IO_MUX_GPIO5_REG
    sw a2, 0(a1)

    li a1, IO_MUX_GPIO6_REG
    sw a2, 0(a1)

    li a1, IO_MUX_GPIO7_REG
    sw a2, 0(a1)
```

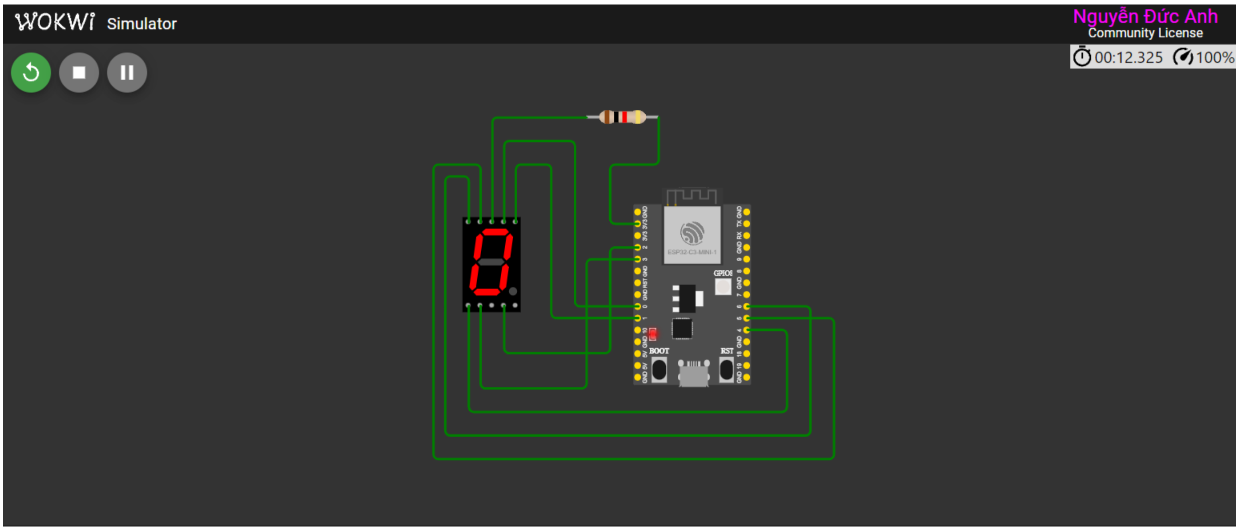
- `li a1, IO_MUX_GPIOn_REG # n = (4 -> 7)`: Load the address IO_MUX_GPIO_REG into a1.
- `sw a2, 0(a1)`: Store the bit value to the address.

3.2.5 Setup the output signal for 7-segment LED

```
    # a1 contains the address of state register GPIO
    li a1, GPIO_OUT_REG
    li a2, 0x40
    sw a2, 0(a1) # Output to GPIO
```

- `li a1, GPIO_OUT_REG`: Load the register for output GPIO.
- `li a2, 0x40`: Load the value to display number on 7-Seg Display (0x40 = 0).
- `sw a2, 0(a1)`: Store the value to the address for output.

3.3 Result:



4 Assignment 4:

4.1 Source Code:

```
.global init

.eqv GPIO_OUT_W1TS_REG, 0x60004008 # set register
.eqv GPIO_OUT_W1TC_REG, 0x6000400C # clear register
.eqv GPIO_ENABLE_REG, 0x60004020 # enable output register
.eqv GPIO_IN_REG, 0x6000403C # state register GPIO
.eqv IO_MUX_GPIO0_REG, 0x60009004 # function register GPIO0

.data

.text
init:
    li a1, GPIO_ENABLE_REG # Set GPIO1 as input
    li a2, 0x02
    sw a2, 0(a1)

    li a1, IO_MUX_GPIO0_REG # Enable GPIO0 as input
    lw a2, 0(a1)
    ori a2, a2, 0x200 # Set bit IO_MUX_GPIO0_FUN_IE
    sw a2, 0(a1)

loop:
    li a1, GPIO_IN_REG # Read status of GPIO
    lw a2, 0(a1)
    andi a3, a2, 0x01 # Check GPIO0
    beq a3, zero, clear # If GPIO0 = 0 => turn off LED

set:
    li a1, GPIO_OUT_W1TS_REG # turn on LED: Set GPIO1 = 1
    li a2, 0x02
    sw a2, 0(a1)
    j next

clear:
    li a1, GPIO_OUT_W1TC_REG # off LED: Clear GPIO1 = 0
    li a2, 0x02
    sw a2, 0(a1)

next:
    j loop # Loop
```

4.2 Explaination:

4.2.1 Register Configure

- `.eqv GPIO_ENABLE_REG, 0x60004020`: Register that enables output functionality for GPIO pins, Bits 0 to 21 correspond to GPIO0 through GPIO21, A bit value of 1 configures the corresponding GPIO pin as an output.

- `.eqv GPIO_OUT_W1TS_REG, 0x60004008`: Register for setting bits in the GPIO_ENABLE_REG register, a bit value of 1 sets the corresponding bit in GPIO_ENABLE_REG, leaving other bits unchanged.

- `.eqv GPIO_OUT_W1TC_REG, 0x6000400C`: Register for clearing bits in the GPIO_ENABLE_REG register, a bit value of 1 clears the corresponding bit in GPIO_ENABLE_REG, leaving other bits unchanged.

- `.eqv IO_MUX_GPIO0_REG, 0X60009004`: Configures the functionality of GPIO pins (GPIO0 to GPIO21), these registers are used to select functions and configure GPIO pin operations.

- `.eqv GPIO_IN_REG, 0x6000403C`: Register for reads the state of GPIO pins configured as input, bits 0 to 21 correspond to GPIO0 through GPIO21, a bit value of 1/0 indicates a high/low logic level on the corresponding GPIO pin.

4.2.2 Enable GPIO1 for input signal for the LED

```
init:
    li a1, GPIO_ENABLE_REG # Set GPIO1 as input
    li a2, 0x02
    sw a2, 0(a1)
```

- `li a1, GPIO_ENABLE_REG`: Load register address to setup GPIO1.
- `li a2, 0x02`: Load the mask 0x02 for register GPIO_ENABLE_REG.
- `sw a2, 0(a1)`: Store the bit value to the address.

4.2.3 Enable GPIO0 for input signal for the Switch

```
li a1, IO_MUX_GPIO0_REG # Enable GPIO0 as input
lw a2, 0(a1)
ori a2, a2, 0x200 # Set bit IO_MUX_GPIO0_FUN_IE
sw a2, 0(a1)
```

- `li a1, IO_MUX_GPIO0_REG`: Load register for input signal on GPIO0.
- `lw a2, 0(a1)`: Load the value from IO_MUX_GPIO0_REG for enable input.
- `ori a2, a2, 0x200`: Set bit IO_MUX_GPIO0_FUN_IE to 1 to enable input.
- `sw a2, 0(a1)`: Store the bit value to the address.

4.2.4 Loop function for reading status of GPIO

```
loop:
    li a1, GPIO_IN_REG # Read status of GPIO
    lw a2, 0(a1)
    andi a3, a2, 0x01 # Check GPIO0
    beq a3, zero, clear # If GPIO0 = 0 => turn off LED
```

- `li a1, GPIO_IN_REG`: Load the register address from GPIO pin.
- `lw a2, 0(a1)`: Read the value from the GPIO pin address.
- `andi a3, a2, 0x01`: Value for checking status of GPIO0.
- `beq a3, zero, clear`: If status value of GPIO0 = 0, jump to clear to turn off the LED.

4.2.5 Turn on the LED method

```
set:
    li a1, GPIO_OUT_W1TS_REG # turn on LED: Set GPIO1 = 1
    li a2, 0x02
    sw a2, 0(a1)
    j next
```

- `li a1, GPIO_OUT_W1TS_REG`: Load the register address from GPIO1 pin for set value.
- `li a2, 0x02`: Set the bit of GPIO1 when the Switch trigger.

- `sw a2, 0(a1)`: Store back the value to register as input signal.
- `j next`: Jump to next segment.

4.2.6 Turn off the LED method

```
clear:
    li a1, GPIO_OUT_W1TC_REG # off LED: Clear GPIO1 = 0
    li a2, 0x02
    sw a2, 0(a1)
```

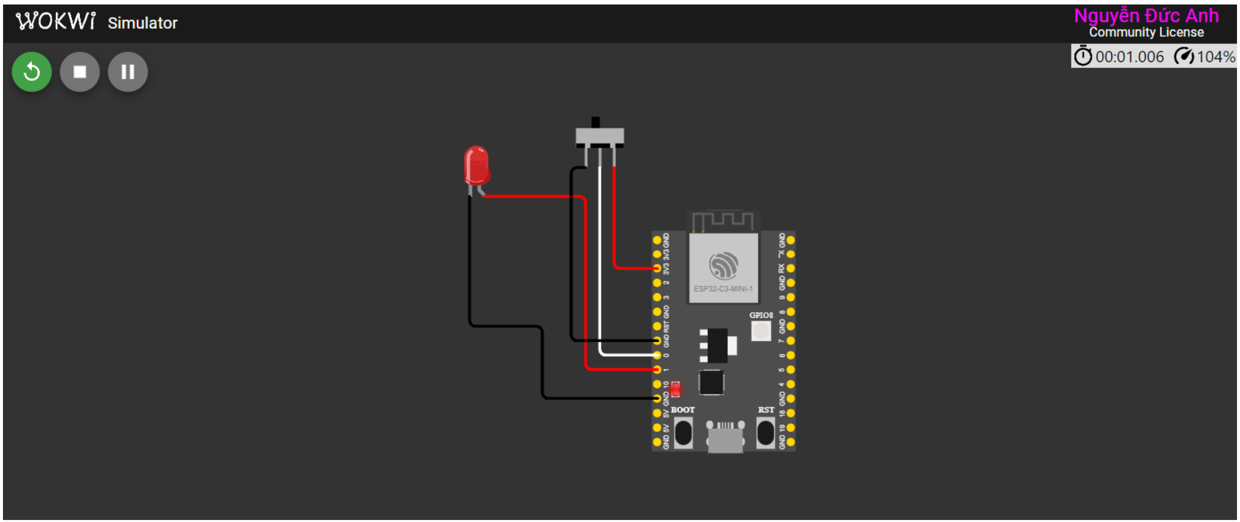
- `li a1, GPIO_OUT_W1TC_REG`: Load the register address from GPIO1 pin for clear value.
- `li a2, 0x02`: Clear the bit of GPIO1 when the Switch turn off.
- `sw a2, 0(a1)`: Store back the value to register.

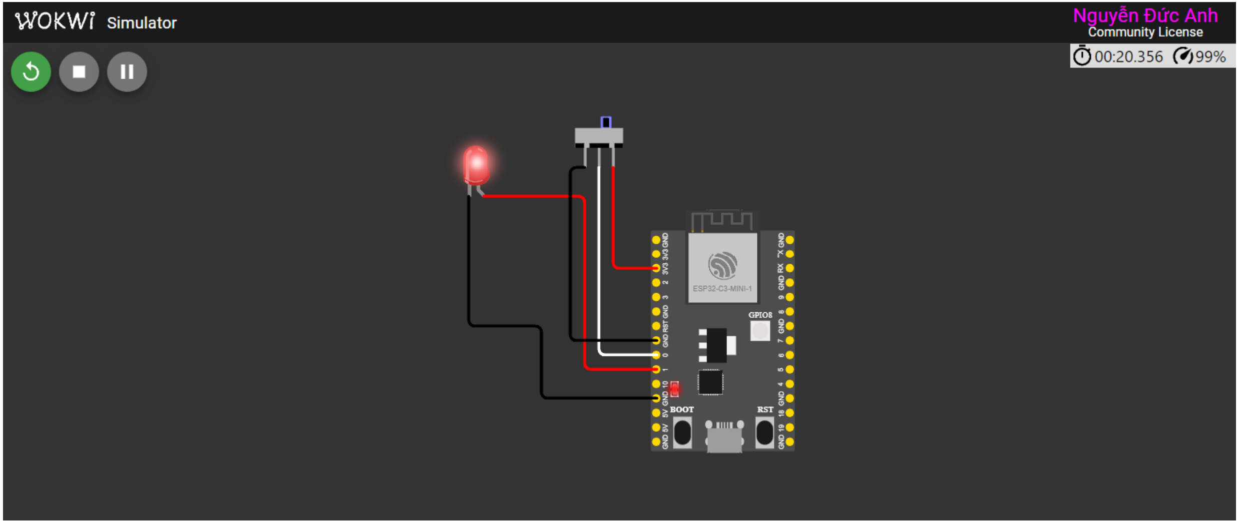
4.2.7 Looping

```
next:
    j loop # Loop
```

- `j loop`: Looping.

4.3 Result:





5 Assignment 5:

5.1 Source Code:

```
.global init

.eqv GPIO_ENABLE_REG, 0x60004020 # Enable output GPIO
.eqv GPIO_OUT_REG, 0x60004004 # Register for output

.eqv IO_MUX_GPIO4_REG, 0x60009014 # Setup function GPIO4
.eqv IO_MUX_GPIO5_REG, 0x60009018 # Setup function GPIO5
.eqv IO_MUX_GPIO6_REG, 0x6000901C # Setup function GPIO6
.eqv IO_MUX_GPIO7_REG, 0x60009020 # Setup function GPIO7

.data
    num_Seg: .word 0x40, 0x79, 0x24, 0x30, 0x19, 0x12, 0x02, 0x78, 0x00, 0x10 # Segment values for 0-9

.text
init:
    # Enable GPIO0-GPIO7 for output
    li a1, GPIO_ENABLE_REG
    li a2, 0xFF           # Enable GPIO0-GPIO7 (8 bits)
    sw a2, 0(a1)          # Write to GPIO_ENABLE_REG

    # Configure GPIO4-GPIO7 for GPIO function (not SPI)
    li a2, 0x1000         # GPIO function value

    li a1, IO_MUX_GPIO4_REG
    sw a2, 0(a1)

    li a1, IO_MUX_GPIO5_REG
    sw a2, 0(a1)

    li a1, IO_MUX_GPIO6_REG
    sw a2, 0(a1)

    li a1, IO_MUX_GPIO7_REG
    sw a2, 0(a1)

declare:
    li t1, 0              # Index
    li t2, 10             # Limit (count from 0 to 9)

loop:
    li a1, GPIO_OUT_REG   # Load the address of GPIO output register
    la t0, num_Seg        # Load base address of segment data (a[0])
    slli t1, t1, 2        # Multiply t1 by 4 to calculate offset
    add t0, t0, t1        # Add offset to base address
    lw a2, 0(t0)          # Load the value for the digit
    sw a2, 0(a1)          # Output to the display

    call delay            # Call for delay

next_num:
    srli t1, t1, 2        # Restore t1 for index counting
    addi t1, t1, 1        # Index = Index + 1
    blt t1, t2, loop      # Check if t1 < t2 -> continue looping

    j declare             # Else reset counting

delay:
    li a3, 0              # Counter
    li a4, 5000000        # Wait time (counting times)
```

```

loop_delay:
    addi a3, a3, 1
    blt a3, a4, loop_delay
    ret
```

5.2 Explaination:

5.2.1 Register Configure

- `.eqv GPIO_ENABLE_REG, 0x60004020`: Register that enables output functionality for GPIO pins, Bits 0 to 21 correspond to GPIO0 through GPIO21, A bit value of 1 configures the corresponding GPIO pin as an output.
- `.eqv GPIO_OUT_REG, 0x60004004`: Register for configures output values for GPIO pins, bits 0 to 21 correspond to GPIO0 through GPIO21, a bit value of 1/0 sets the corresponding GPIO pin to a high/low logic level.
- `.eqv IO_MUX_GPIO4_REG, 0X60009014`: Configures the functionality of GPIO pins (GPIO0 to GPIO21), these registers are used to select functions and configure GPIO pin operations.
- `.eqv IO_MUX_GPIO5_REG, 0X60009018`: Configures the functionality of GPIO pins (GPIO0 to GPIO21), these registers are used to select functions and configure GPIO pin operations.
- `.eqv IO_MUX_GPIO6_REG, 0X6000901C`: Configures the functionality of GPIO pins (GPIO0 to GPIO21), these registers are used to select functions and configure GPIO pin operations.
- `.eqv IO_MUX_GPIO7_REG, 0X60009020`: Configures the functionality of GPIO pins (GPIO0 to GPIO21), these registers are used to select functions and configure GPIO pin operations.

5.2.2 Data Segment

```
    .data
        num_Seg: .word 0x40, 0x79, 0x24, 0x30, 0x19, 0x12, 0x02, 0x78, 0x00, 0x10
```

- **num_Seg**: Segment values for 0-9.

5.2.3 Enable and Initialize the output for GPIO0 to GPIO7

```
    init:
        li a1, GPIO_ENABLE_REG
        li a2, 0xFF # output from GPIO0 to GPIO7 (8 bits)
        sw a2, 0(a1) # setupt bits in GPIO_ENABLE_REG
```

- `li a1, GPIO_ENABLE_REG`: Load register address to setup GPIO0.
- `li a2, 0xFF`: Load the mask 0xFF for register GPIO_ENABLE_REG.
- `sw a2, 0(a1)`: Store the mask value to the address.

5.2.4 Setup the value of IO_MUX_GPIOn_MCU_SEL on bit 12

```
    # Configure GPIO4-GPIO7 for GPIO function (not SPI)
    li a2, 0x1000 # GPIO function value
```

- `li a2, 0x1000`: Load value 1 bit to bit 12.

5.2.5 Initialize the value of IO_MUX_GPIOn_MCU_SEL on bit 12 to GPIO4 to GPIO7 pin manually

```
li a1, IO_MUX_GPIO4_REG
sw a2, 0(a1)

li a1, IO_MUX_GPIO5_REG
sw a2, 0(a1)

li a1, IO_MUX_GPIO6_REG
sw a2, 0(a1)

li a1, IO_MUX_GPIO7_REG
sw a2, 0(a1)
```

- `li a1, IO_MUX_GPIOn_REG # n = (4 -> 7)`: Load the address IO_MUX_GPIO_REG into a1.
- `sw a2, 0(a1)`: Store the bit value to the address.

5.2.6 Declare the index and limit

```
declare:
    li t1, 0 # Index
    li t2, 10 # Limit (count from 0 to 9)
```

- `li t1, 0`: Declare the index.
- `li t2, 10`: Declare the limit (10 digits).

5.2.7 Main Loop

```
loop:
    li a1, GPIO_OUT_REG    # Load the address of GPIO output register
    la t0, num_Seg         # Load base address of segment data (a[0])
    slli t1, t1, 2         # Multiply t1 by 4 to calculate offset
    add t0, t0, t1         # Add offset to base address
    lw a2, 0(t0)           # Load the value for the digit
    sw a2, 0(a1)           # Output to the display

    call delay             # Call for delay
```

- `li a1, GPIO_OUT_REG`: Load the register for output GPIO.
- `la t0, num_Seg`: Load base address of segment data (a[0]).
- `slli t1, t1, 2`: Shift left 2 to calculate offset.
- `add t0, t0, t1`: Add offset to base address.
- `lw a2, 0(t0)`: Load the value for the digit.
- `sw a2, 0(a1)`: Store the value to the address for output.
- `call delay`: Call for delay.

5.2.8 Increase the index

```
next_num:
    srli t1, t1, 2          # Restore t1 for index counting
    addi t1, t1, 1          # Index = Index + 1
    blt t1, t2, loop        # Check if t1 < t2 -> continue looping

j declare               # Else reset counting
```

- `srli t1, t1, 2`: Shift right 2 to restore t1 for index counting.
- `addi t1, t1, 1`: Increase the index.
- `blt t1, t2, loop`: Check if t1 smaller than t2, continue looping
- `j declare`: Else reset counting

5.2.9 Delay Function

```
delay_asm:
    li a3, 0     # counter
    li a4, 5000000  # wait time (counting time)
```

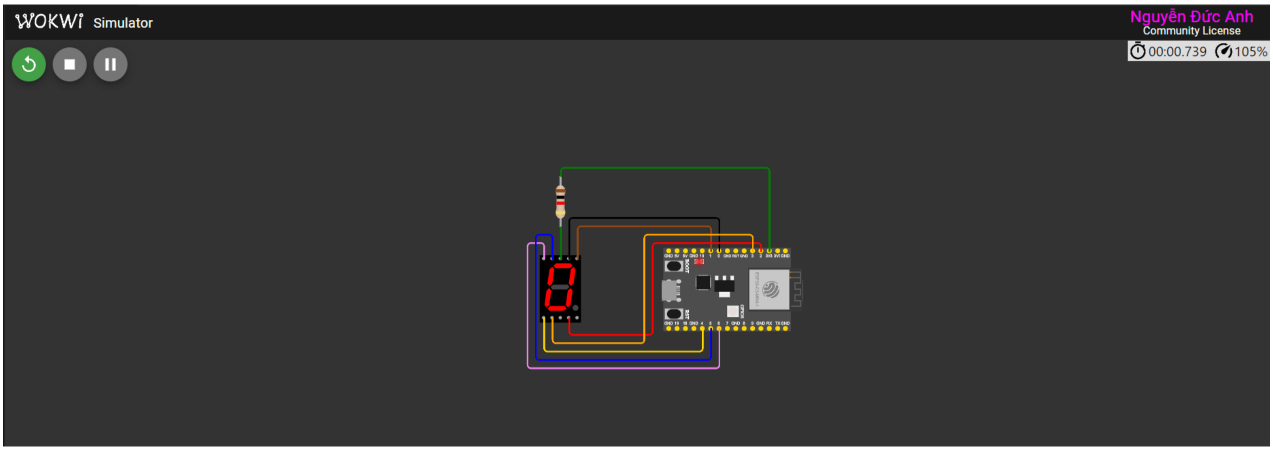
- `li a3, 0`: Initialize the index value.
- `li a4, 5000000`: Initialize the time for stop blinking.

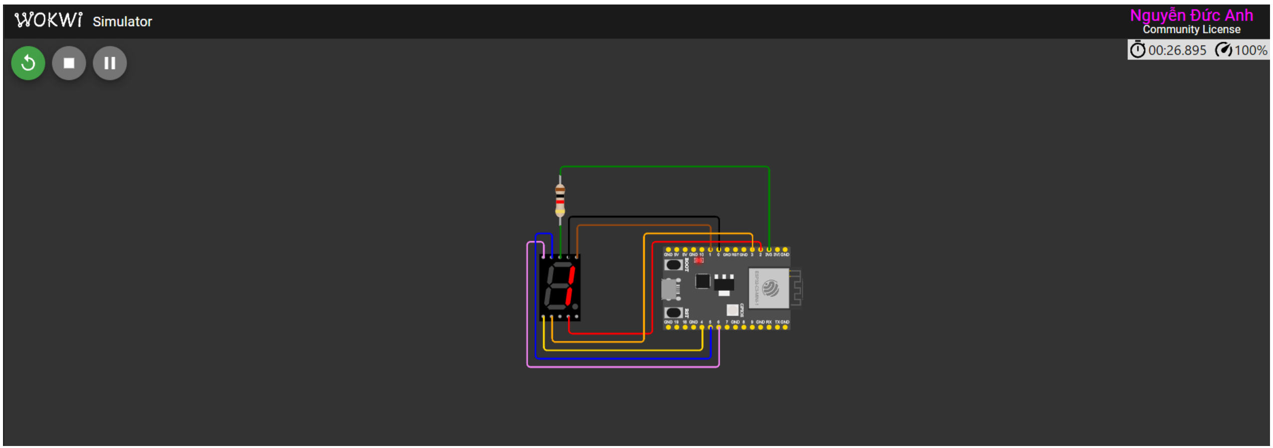
5.2.10 Delay Looping

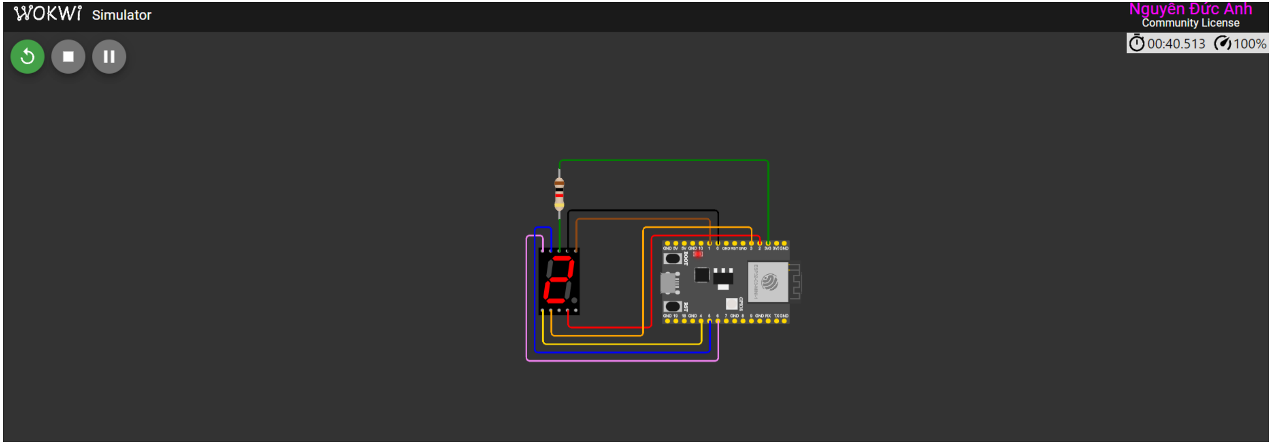
```
loop_delay:
    addi a3, a3, 1
    blt a3, a4, loop_delay
    ret
```

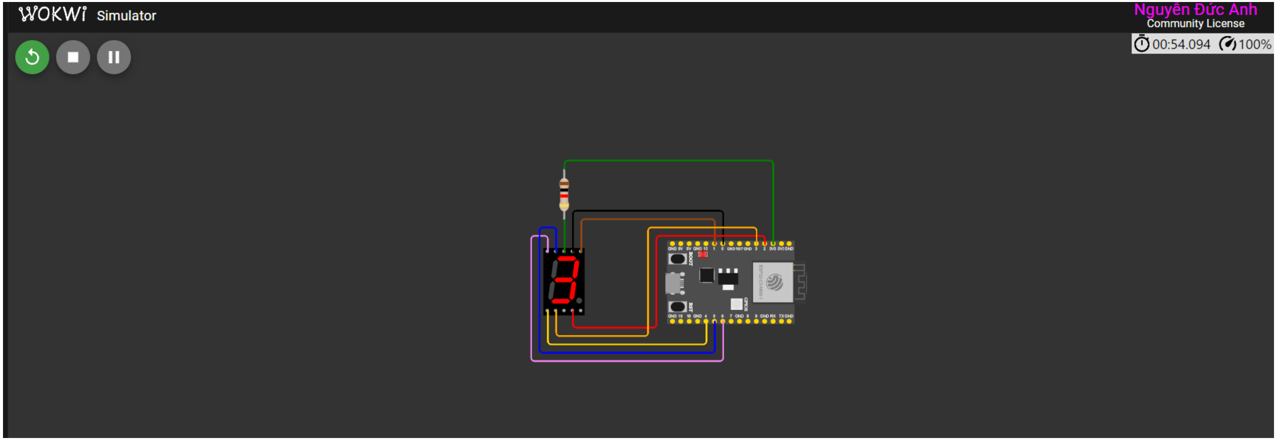
- `addi a3, a3, 1`: Increase the index.
- `blt a3, a4, loop_delay`: Check if index smaller than time, continue looping.

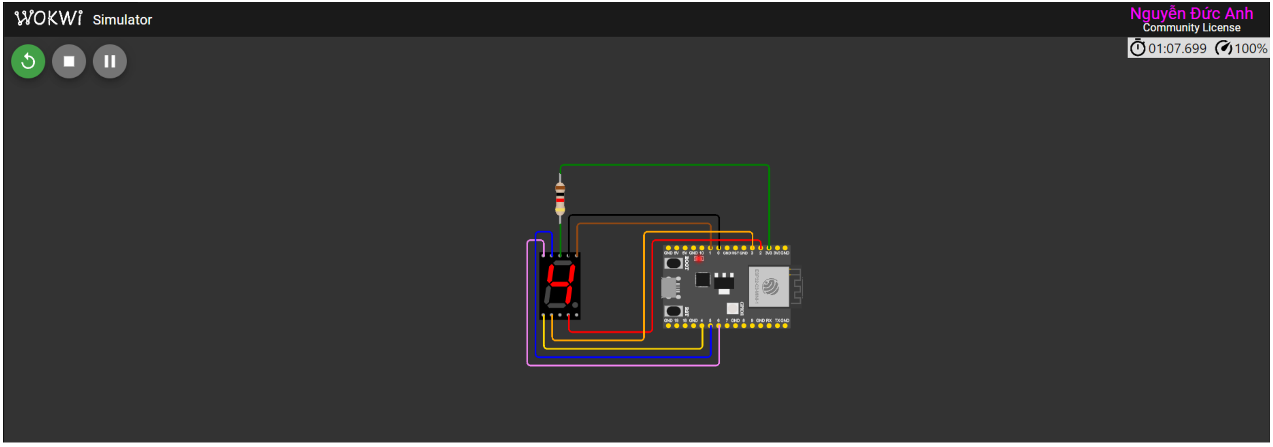
5.3 Result:

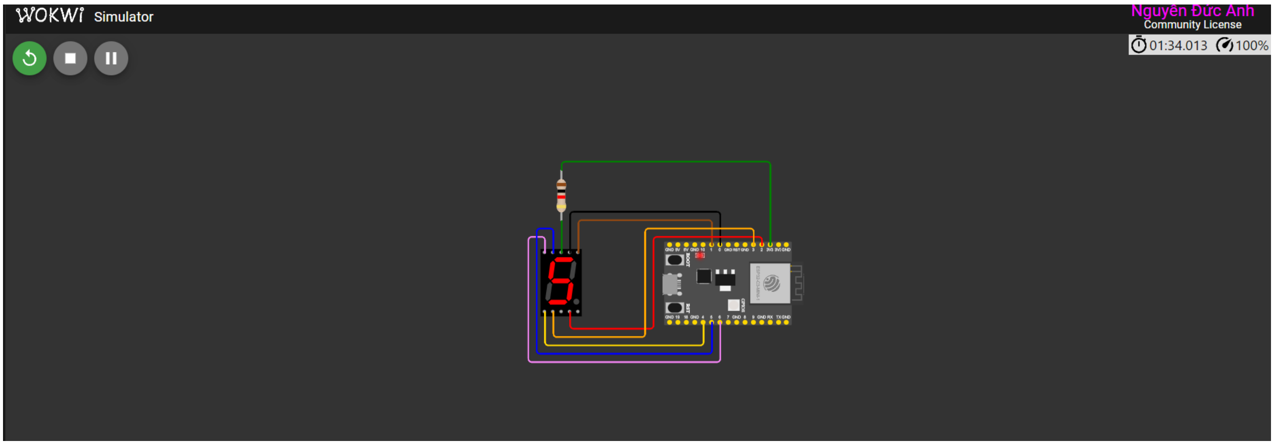


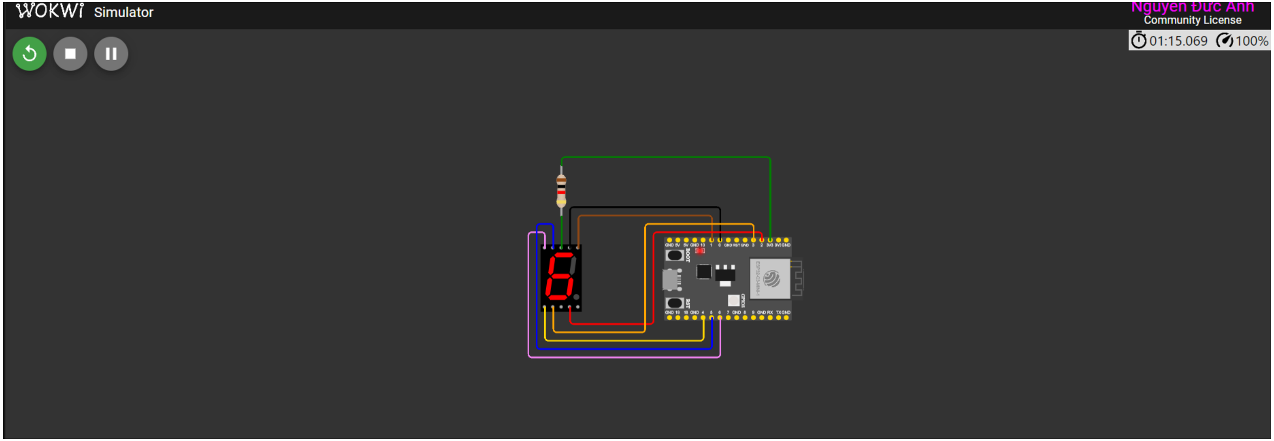

WOKWi Simulator
Nguyễn Đức Anh
Community License
00:26.895
100%

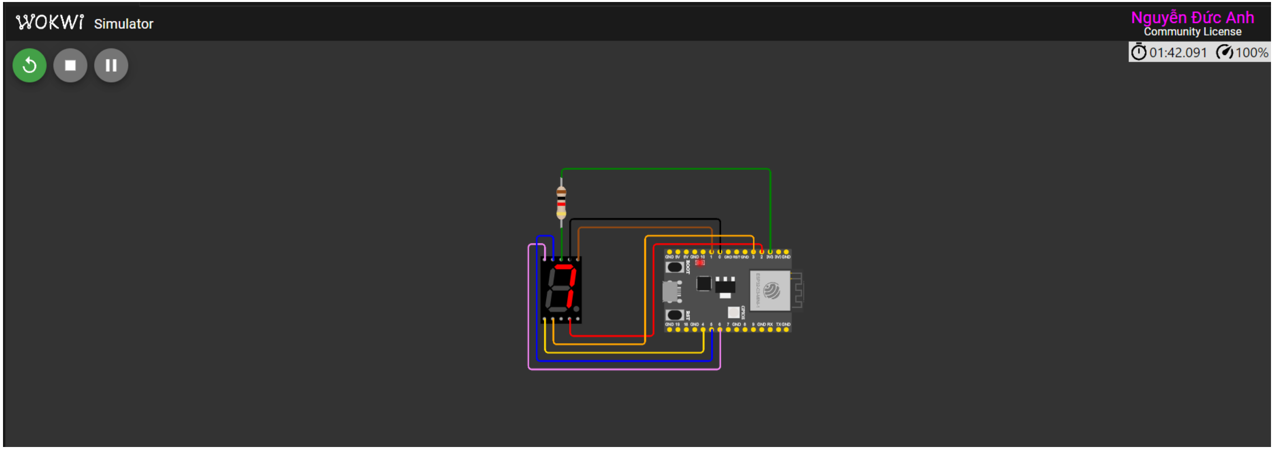

WOKWi Simulator
Nguyễn Đức Anh
Community License
00:40.513
100%

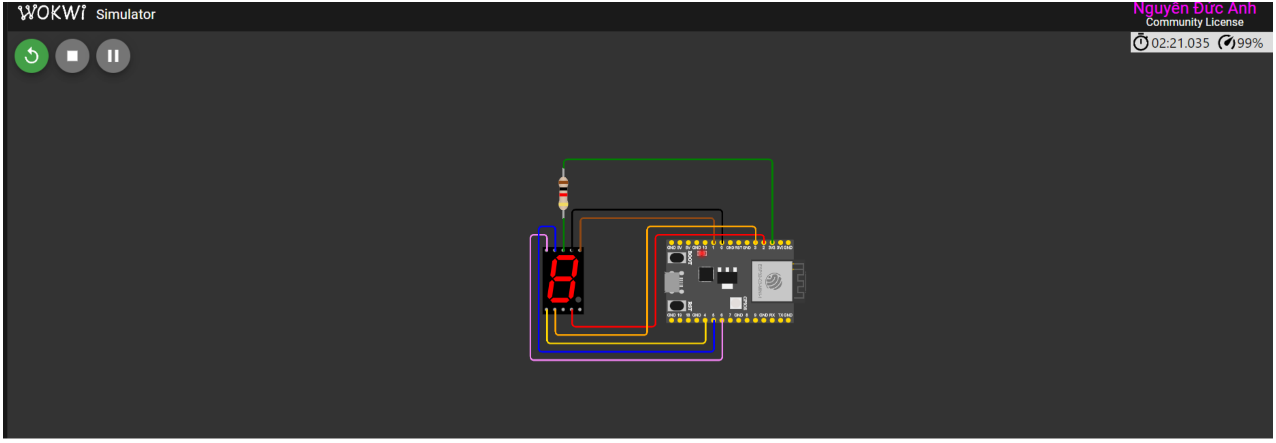

WOKWi Simulator
Nguyễn Đức Anh
Community License
00:54.094
100%

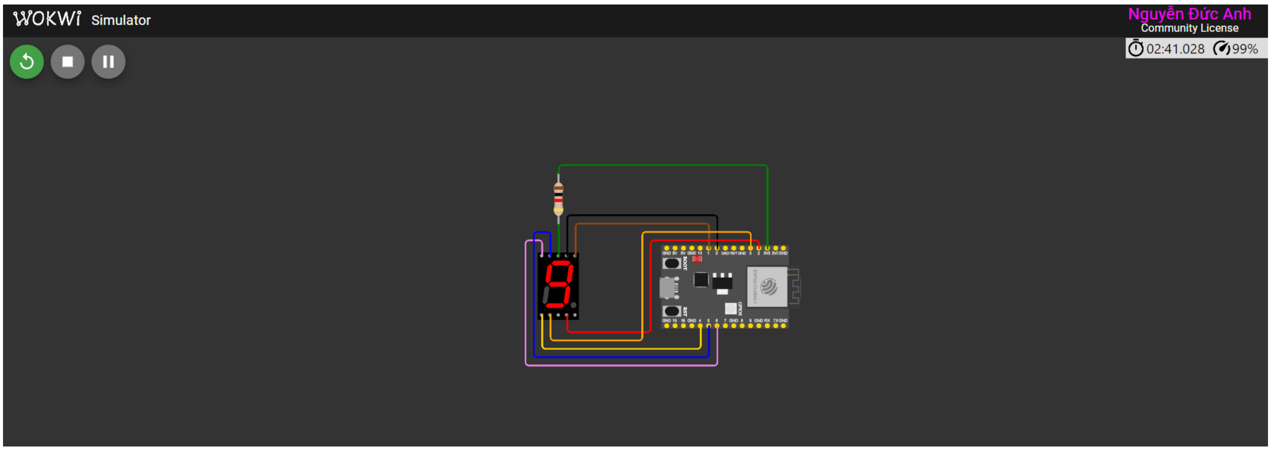

WOKWi Simulator
Nguyễn Đức Anh
Community License
01:07.699
100%


WOKWi Simulator
Nguyễn Đức Anh
Community License
01:34.013
100%


WOKWi Simulator
Nguyễn Đức Anh
Community License
01:15.069
100%


WOKWi Simulator
Nguyễn Đức Anh
Community License
01:42.091
100%


WOKWi Simulator
Nguyễn Đức Anh
Community License
02:21.035
99%


WOKWi Simulator
Nguyễn Đức Anh
Community License
02:41.028
99%