



대한상공회의소  
서울기술교육센터



# CPU Design and Verification

## : Vehicle Application

발표자: 김지환

팀원: 김태민, 박지수, 함영은

# Contents



01. Introduction

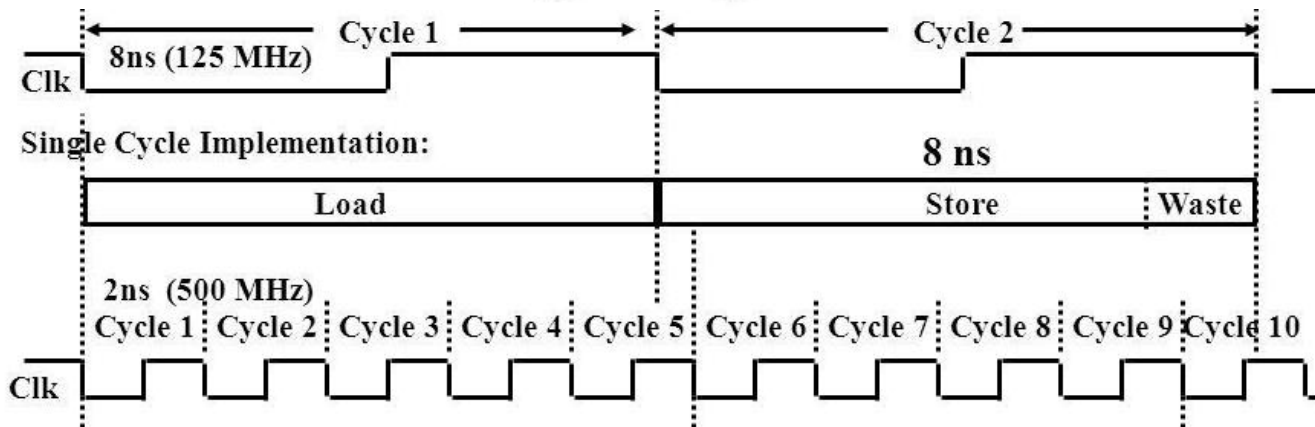
02. ABP BUS Specification

03. Peripherals

04. Conclusion

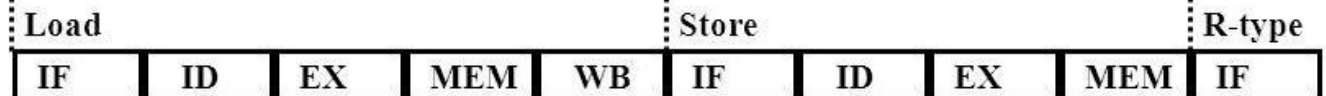
# 01. Introduction

## Single Cycle



## Multi-Cycle CPU

Multiple Cycle Implementation.



Single-Cycle CPU:

CPI = 1  $C = 8\text{ns}$   $f = 125\text{ MHz}$

One million instructions take =

$$I \times \text{CPI} \times C = 10^6 \times 1 \times 8 \times 10^{-9} = 8 \text{ msec}$$

Multi-Cycle CPU:

CPI = 3 to 5  $C = 2\text{ns}$   $f = 500\text{ MHz}$

One million instructions take from

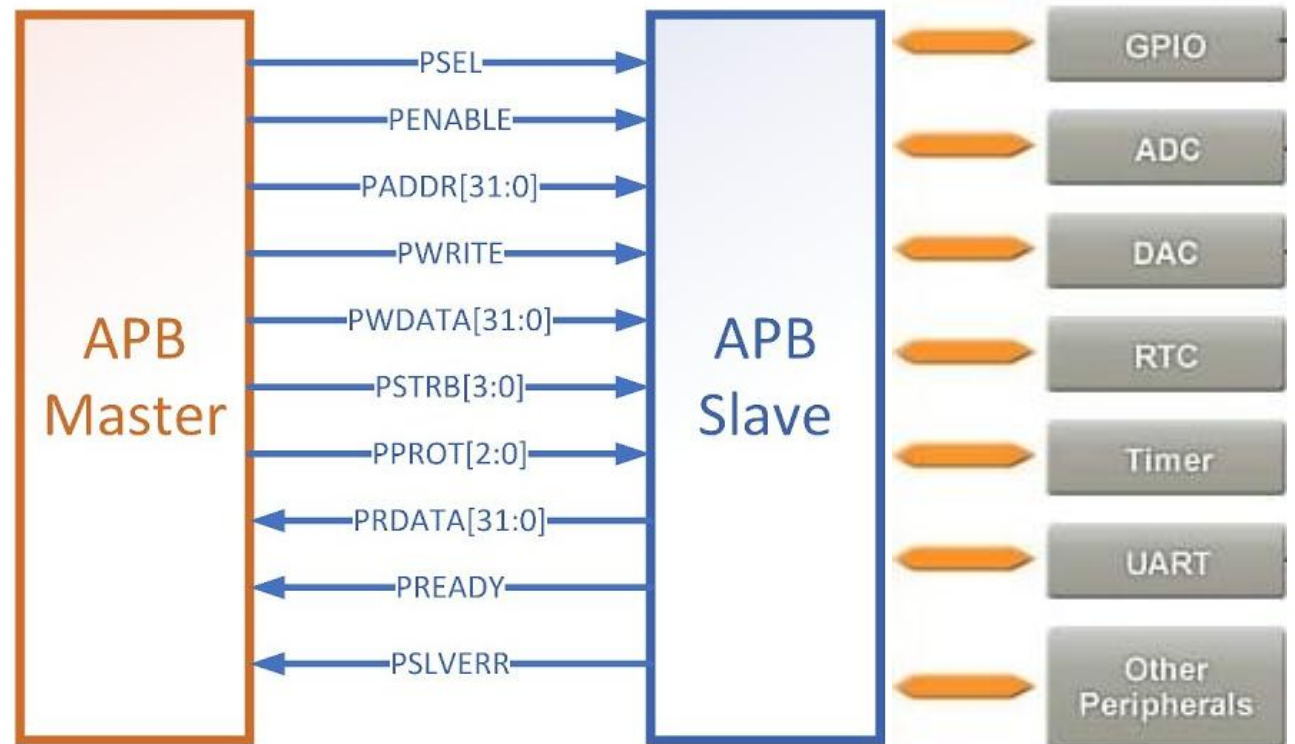
$$10^6 \times 3 \times 2 \times 10^{-9} = 6 \text{ msec}$$

to  $10^6 \times 5 \times 2 \times 10^{-9} = 10 \text{ msec}$   
depending on instruction mix used.

# 01. Introduction

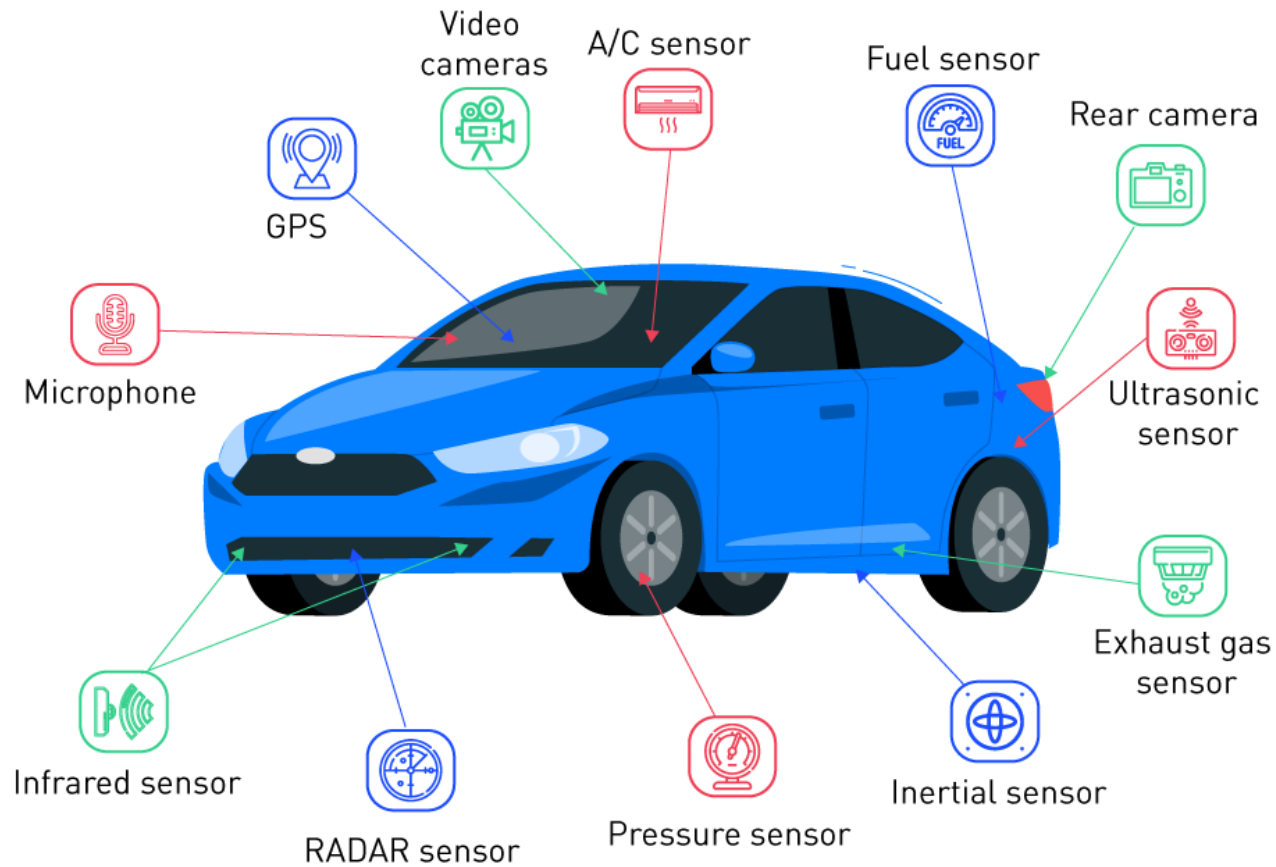
## Project Goal

**32-bit  
RISC-V CPU**



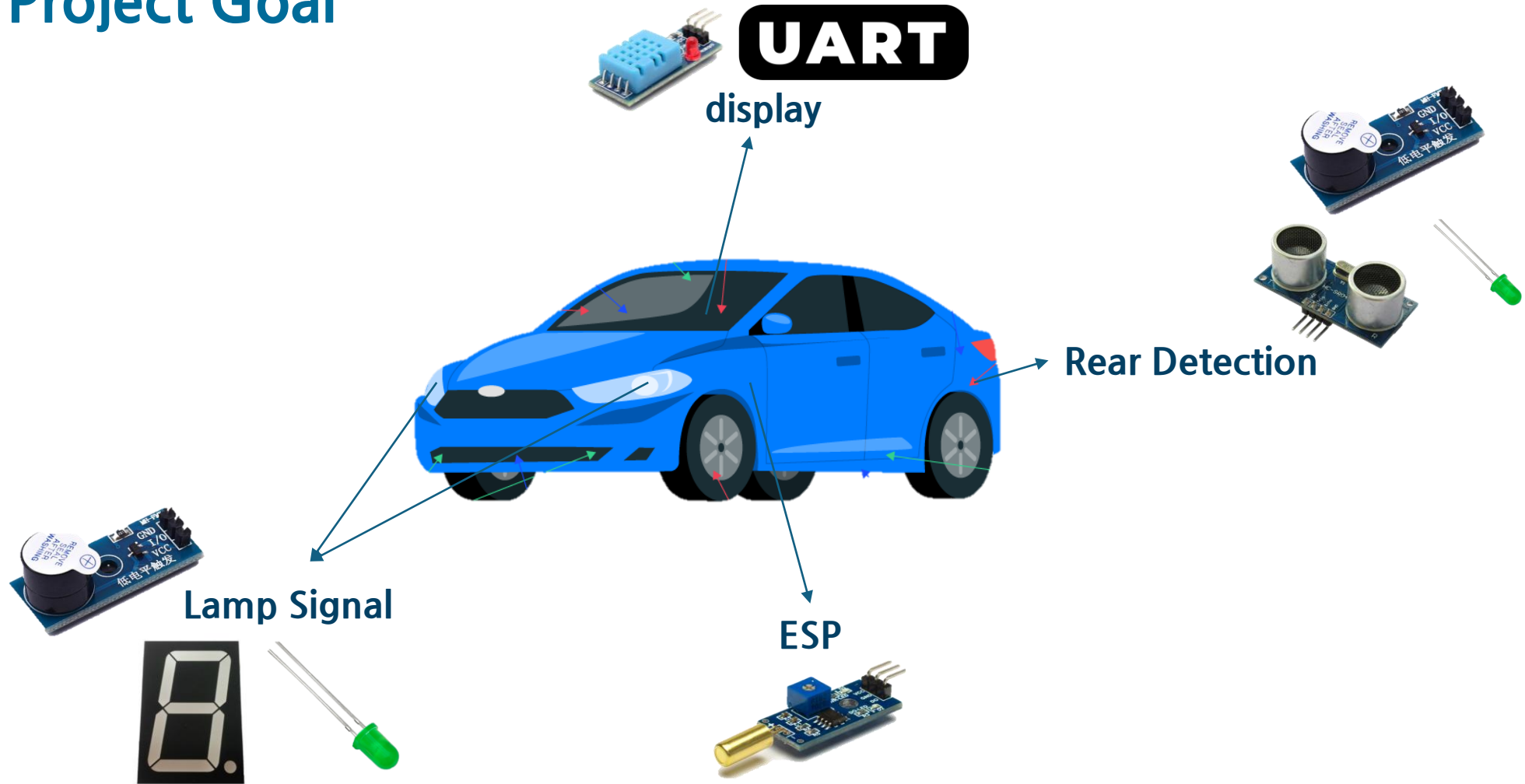
# 01. Introduction

## Project Goal



# 01. Introduction

## Project Goal



# 01. Introduction

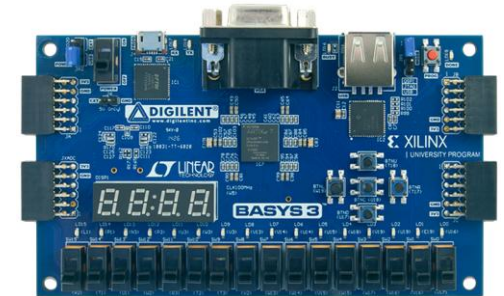
## Tool



## Language



## HW



# Advanced Peripheral Bus (APB)

**ARM<sup>®</sup>AMBA<sup>®</sup>**

Interconnect Standards



## 02. APB Bus Specification

# Advanced Peripheral Bus (APB)

### Key AMBA Specifications

AMBA generation:

AMBA 2

AMBA 3

AMBA 4

AMBA 5

**CHI**  
Coherent Hub Interface

CHI is a credited coherency protocol, layered architecture for scalability

**CHI**

**ACE**  
AXI coherency Extensions

ACE is superset of AXI – brings system-wide coherency across multicore clusters

**ACE**  
+Lite

**ACE5**  
+Lite

**AXI**  
Advanced eXtensible Interface

AXI supports separate A/D phases, bursts, multiple outstanding addresses, OoO responses

**AXI3**

**AXI4**  
+Lite, +Stream

**AXI5**

**AHB**  
Adv. High-performance Bus

AHB supports 64/128 bit, multi-master. AHB-Lite for single masters

**AHB**

**AHB**  
+Lite

**AHB5**  
+Lite

**APB**  
Advanced Peripheral Bus

System bus for low b/w peripherals

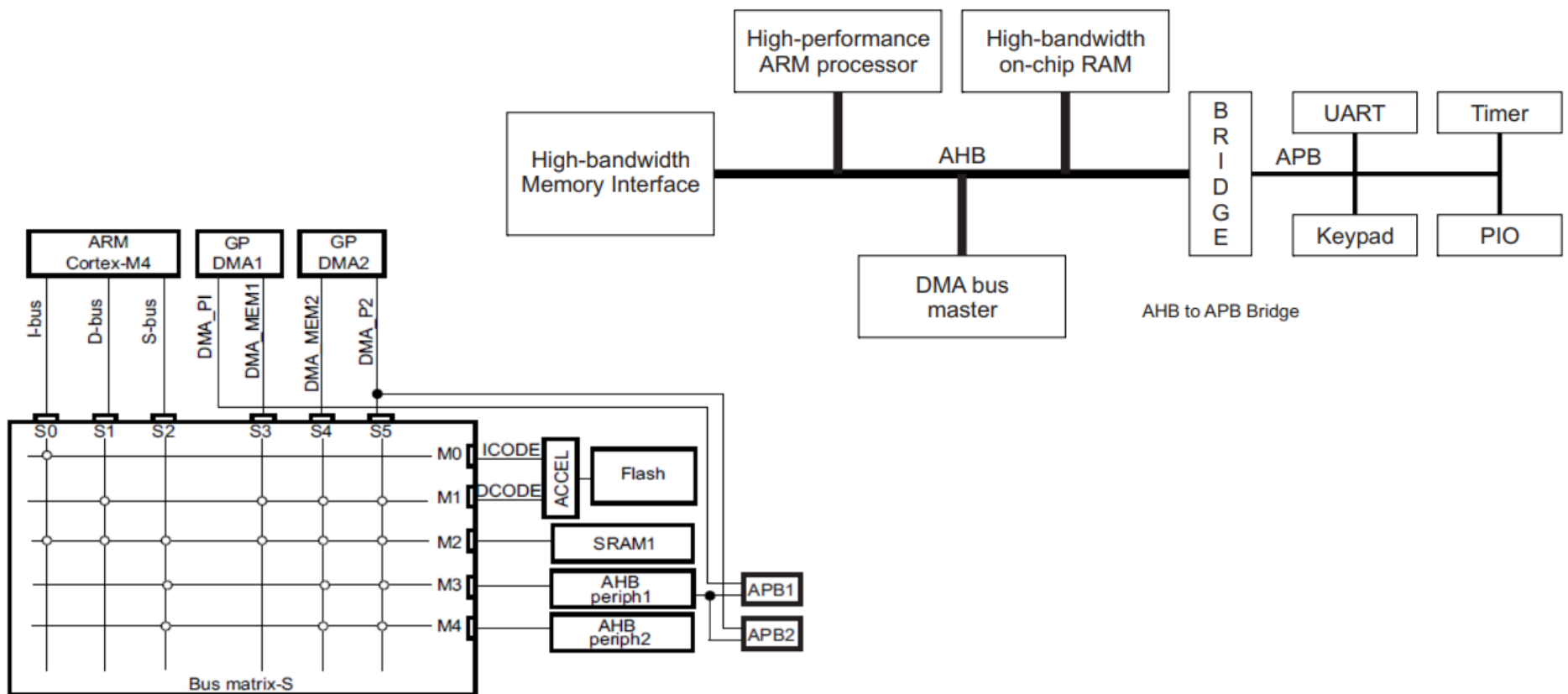
**APB2**

**APB3**

**APB4**

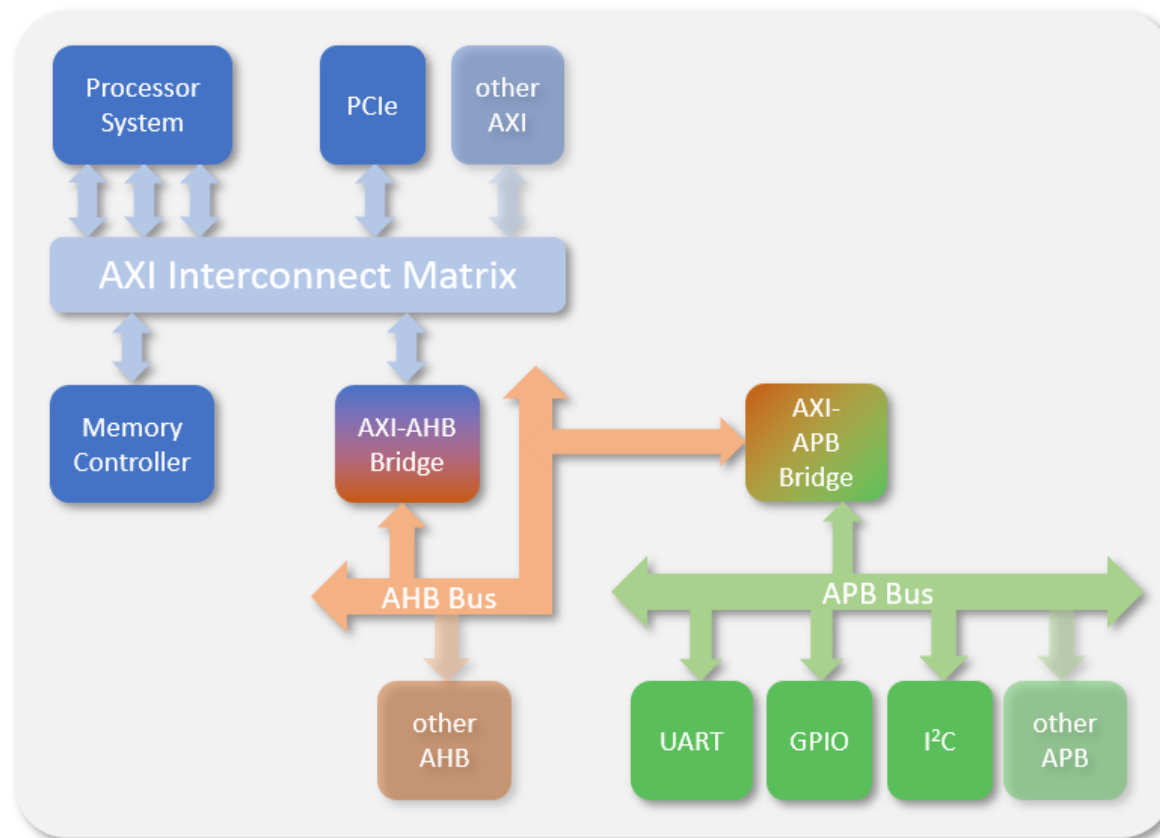
## 02. APB Bus Specification

# Advanced Peripheral Bus (APB)

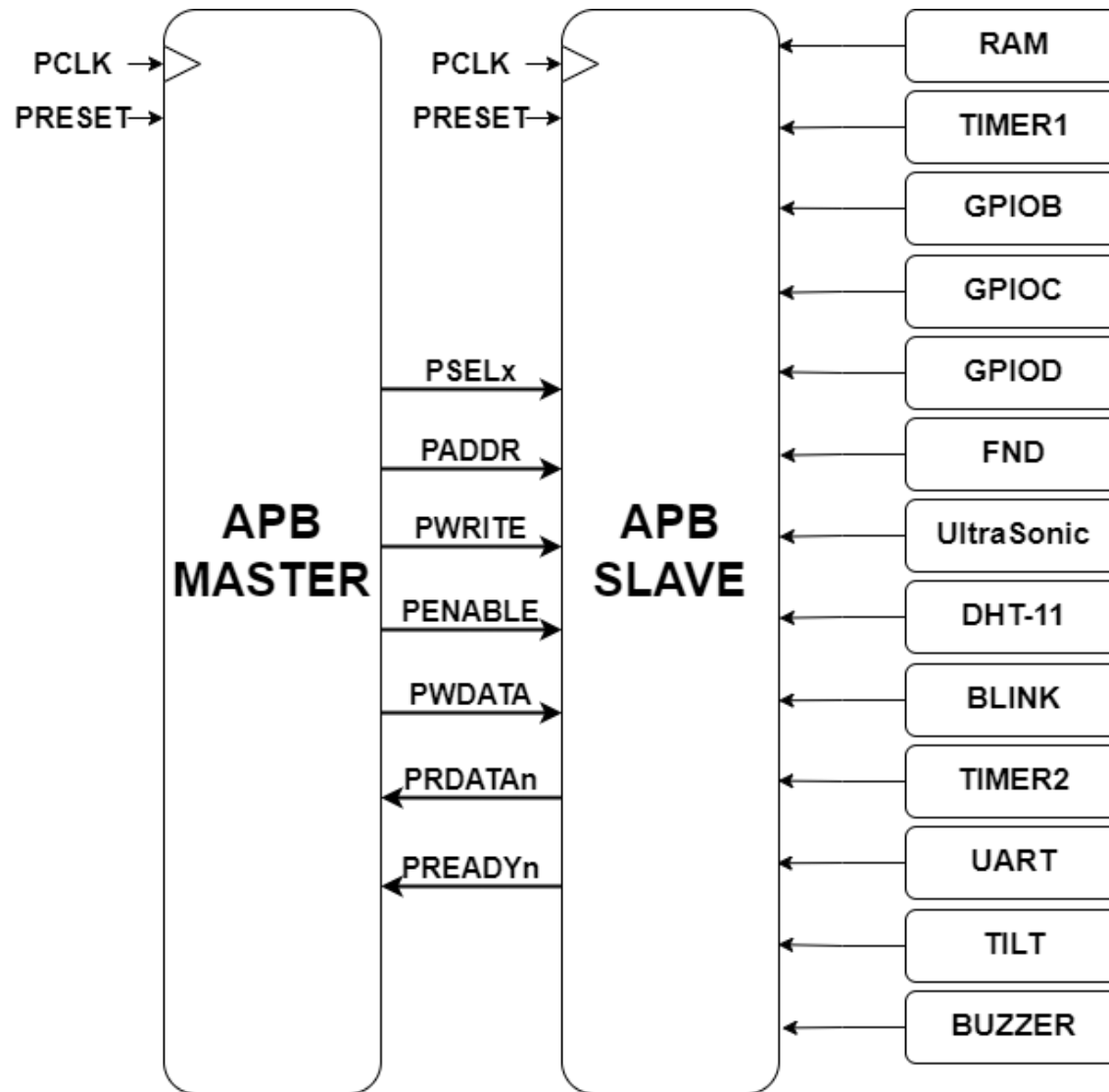


## 02. APB Bus Specification

# Advanced Peripheral Bus (APB)



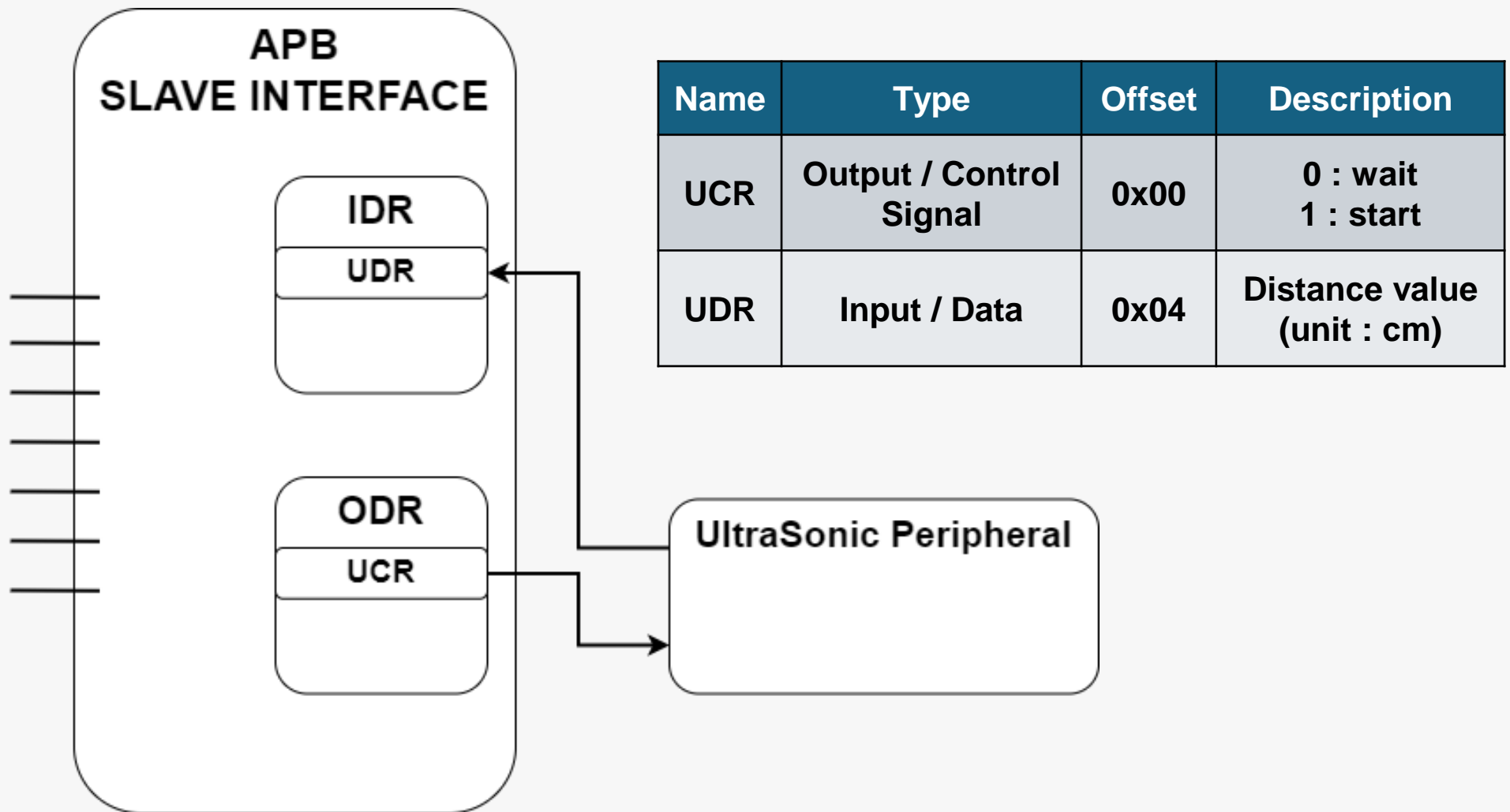
## 02. APB Bus Specification



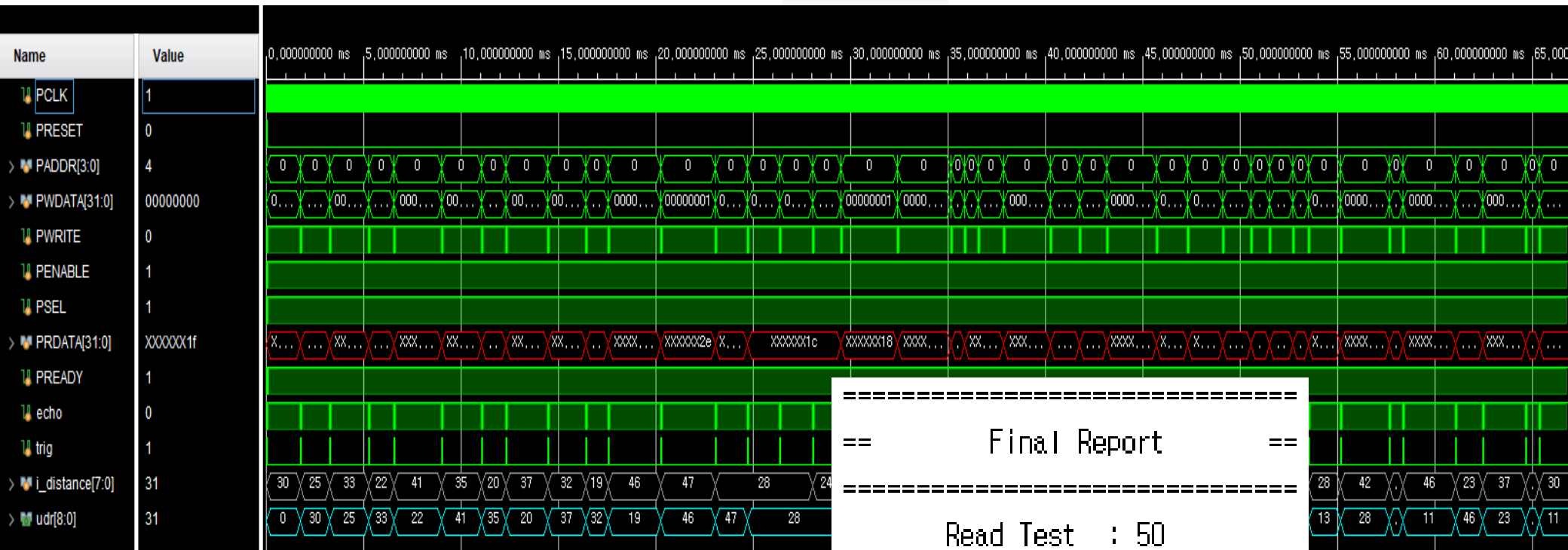
## 02. APB Bus Specification

Bus	Boundary address	Peripheral
APB	0x1000 4000 ~ 0x1000 43FF	BUZZER
	0x1000 3C00 ~ 0x1000 3FFF	NOTHING
	0x1000 3800 ~ 0x1000 3BFF	TILT
	0x1000 3400 ~ 0x1000 37FF	UART
	0x1000 3000 ~ 0x1000 33FF	TIMER2
	0x1000 2C00 ~ 0x1000 2FFF	BLINK
	0x1000 2800 ~ 0x1000 2BFF	DHT-11
	0x1000 2400 ~ 0x1000 27FF	ULTRASONIC
	0x1000 2000 ~ 0x1000 23FF	FND
	0x1000 1C00 ~ 0x1000 1FFF	GPIOC
	0x1000 1800 ~ 0x1000 1BFF	GPIOB
	0x1000 1400 ~ 0x1000 17FF	GPIOA
	0x1000 1000 ~ 0x1000 13FF	TIMER1
	0x1000 0000 ~ 0x1000 0FFF	RAM

# 03. Peripherals - UltraSonic



# 03. Peripherals - UltraSonic Simulation



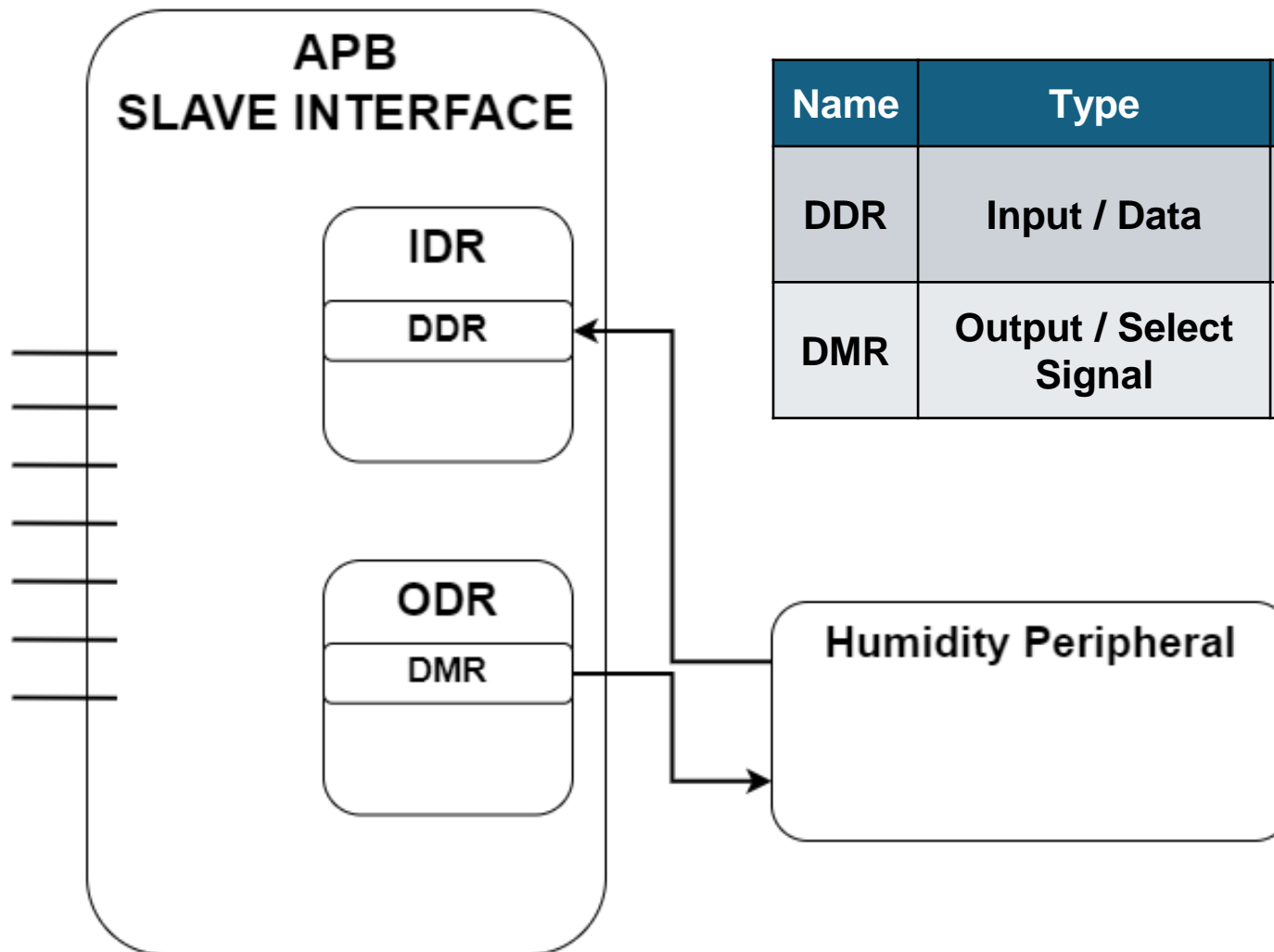
```
=====
==      Final Report      ==
=====

Read Test   : 50
(^-^)b PASS Test : 50
(;-)p FAIL Test  : 0
Total Test  : 50

=====
=====
```

Gray : Random Vaule  
Aqua : DUT UDR Value

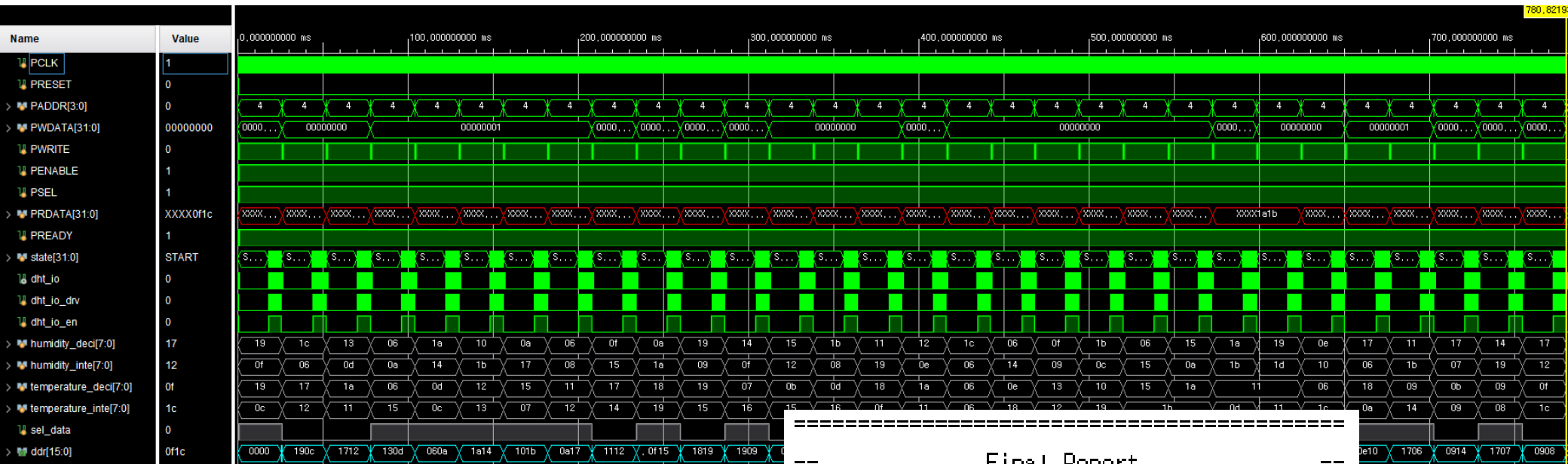
# 03. Peripherals – DHT-11



Name	Type	Offset	Description
DDR	Input / Data	0x00	Value
DMR	Output / Select Signal	0x04	0 : Temperature 1 : Humidity



# 03. Peripherals - Humidity Simulation



```
=====
Final Report
=====

Read Test : 30
(^-^)b PASS Test : HUMI = 13 / TEMP = 17
(:-;)p FAIL Test : HUMI = 0 / TEMP = 0
Total Test : 30
=====
```

Gray : Random Vaule  
Aqua : DUT DDR Value

# 03. Peripherals - C Code

## definition

```
typedef struct {  
    __IO uint32_t UCR; #define ULTRA_BASEADDR (APB_BASEADDR + 0x2400)  
    __IO uint32_t UDR; #define DHT_BASEADDR (APB_BASEADDR + 0x2800)  
} ULTRA_TypeDef; #define ULTRA ((ULTRA_TypeDef *) ULTRA_BASEADDR)  
#define DHT ((DHT_TypeDef *) DHT_BASEADDR)  
  
typedef struct {  
    __IO uint32_t DDR;  
    __IO uint32_t DMR;  
} DHT_TypeDef;
```

## function

```
void Ultra_init(ULTRA_TypeDef *ultra, uint32_t power){  
    ultra->UCR = power;  
}  
  
uint32_t Ultra_read(ULTRA_TypeDef *ultra){  
    return ultra->UDR;  
}  
  
void DHT_init(DHT_TypeDef *dht, uint32_t moder){  
    dht->DMR = moder;  
}  
  
uint32_t DHT_read(DHT_TypeDef *dht){  
    return dht->DDR;  
}
```

# 03. Peripherals - C Code

```
uint32_t sw = Switch_read(GPIOB);  
  
switch (sw) {
```

## UltraSonic run code

```
case (1 << 6):  
    delay(500);  
    Ultra_init(ULTRA, POWER_ON);  
    delay(10);  
    distance = Ultra_read(ULTRA);  
    Ultra_init(ULTRA, POWER_OFF);  
    FND_writeData(FND, distance);  
    BLINK_init(BLINK, distance);  
    BLINK_init(BUZZER, distance);  
    delay(100);  
    UART_Send_distance(UART, get_thousands_place(&distance),  
        get_hundreds_place(&distance), get_tens_place(&distance),  
        get_ones_place(&distance));  
    break;
```

## DHT-11 run code

```
case (1 << 5):  
    DHT_init(DHT, TEMPERATURE);  
    delay(1000);  
    temperature = DHT_read(DHT);  
    FND_writeData(FND, temperature);  
    delay(100);  
    UART_Send_Temp(UART, get_thousands_place(&temperature),  
        get_hundreds_place(&temperature), get_tens_place(&temperature),  
        get_ones_place(&temperature));  
    break;  
  
case (1 << 4):  
    DHT_init(DHT, HUMIDITY);  
    delay(1000);  
    humidity = DHT_read(DHT);  
    FND_writeData(FND, humidity);  
    delay(100);  
    UART_Send_Humi(UART, get_thousands_place(&humidity),  
        get_hundreds_place(&humidity), get_tens_place(&humidity),  
        get_ones_place(&humidity));  
    break;
```

# 03. Peripherals - C Code

## Car Blinker run code

```
case (1 << 3): {  
    LED_write(GPIOA, led_default);  
    FND_init(FND, POWER_OFF);  
    while(Switch_read(GPIOB) == (1<<3))  
    {  
        }  
  
    LED_write(GPIOA, 0);  
    FND_init(FND, POWER_ON);  
    break;  
}
```

Name	Input	Output
Left Signal Lamp	Switch[0]	led[8] FND
Right Signal Lamp	Switch[1]	led[9] FND
Hazard Light	Top Button	led[8], led[9] FND

# 03. Peripherals - C Code

## Car Blinker run code - State Determination

```
#define DEFAULT_STATE      0
#define HAZARD_BLINK_STATE 1
#define RIGHT_BLINK_STATE  2
#define LEFT_BLINK_STAT   3
```

Case	Description	State
(1<<0)	Switch[0]	LEFT
(1<<1)	Switch[1]	RIGHT
(1<<4)	Top Button	Hazard

```
switch(Switch_read(GPIOC))
{
    case (1<<0):
        blinker_state = LEFT_BLINK_STAT;
        break;

    case (1<<1):
        blinker_state = RIGHT_BLINK_STATE;
        break;

    case (1<<4):
        delay(10);
        if((Switch_read(GPIOC) == (1<<4)) && (btn_detect == 0))
        {
            btn_detect = 1;
            blinker_state ^= HAZARD_BLINK_STATE;
        }
        break;

    default:
        btn_detect = 0;
        if(!(blinker_state == 1))
        {
            blinker_state = DEFAULT_STATE;
            led_data = 0b11;
            LED_write(GPIOA, led_default);
            FND_init(FND,POWER_OFF);
            BLINK_init(BUZZER, 1);
        }
}
```

# 03. Peripherals - C Code

## Car Blinker run code - State-Driven Operation

```
case HAZARD_BLINK_STATE:
    ggambbak = 0b11;
    fnd_shape = HAZARD;
    if(Timer_read(TIMER2) == 0 && blink_flag == 0)
    {
        blink_flag = 1;
        led_data ^= ggambbak;
        fnd_blink = (led_data & ggambbak) == 0 ? POWER_OFF : POWER_ON;
    }
    else if(Timer_read(TIMER2) != 0) blink_flag = 0;

    delay(10);

    BLINK_init(BUZZER, 49);
    LED_write(GPIOA, led_data);
    FND_init(FND, fnd_blink);
    FND_writeData(FND, fnd_shape);
    break;
```

```
case DEFAULT_STATE: break;
```

```
case LEFT_BLINK_STAT:
    ggambbak = 0b10;
    fnd_shape = LEFT;
    if(Timer_read(TIMER2) == 0 && blink_flag == 0)
    {
        blink_flag = 1;
        led_data ^= ggambbak;
        fnd_blink = (led_data & ggambbak) == 0 ? POWER_OFF : POWER_ON;
    }
    else if(Timer_read(TIMER2) != 0) blink_flag = 0;

    delay(10);

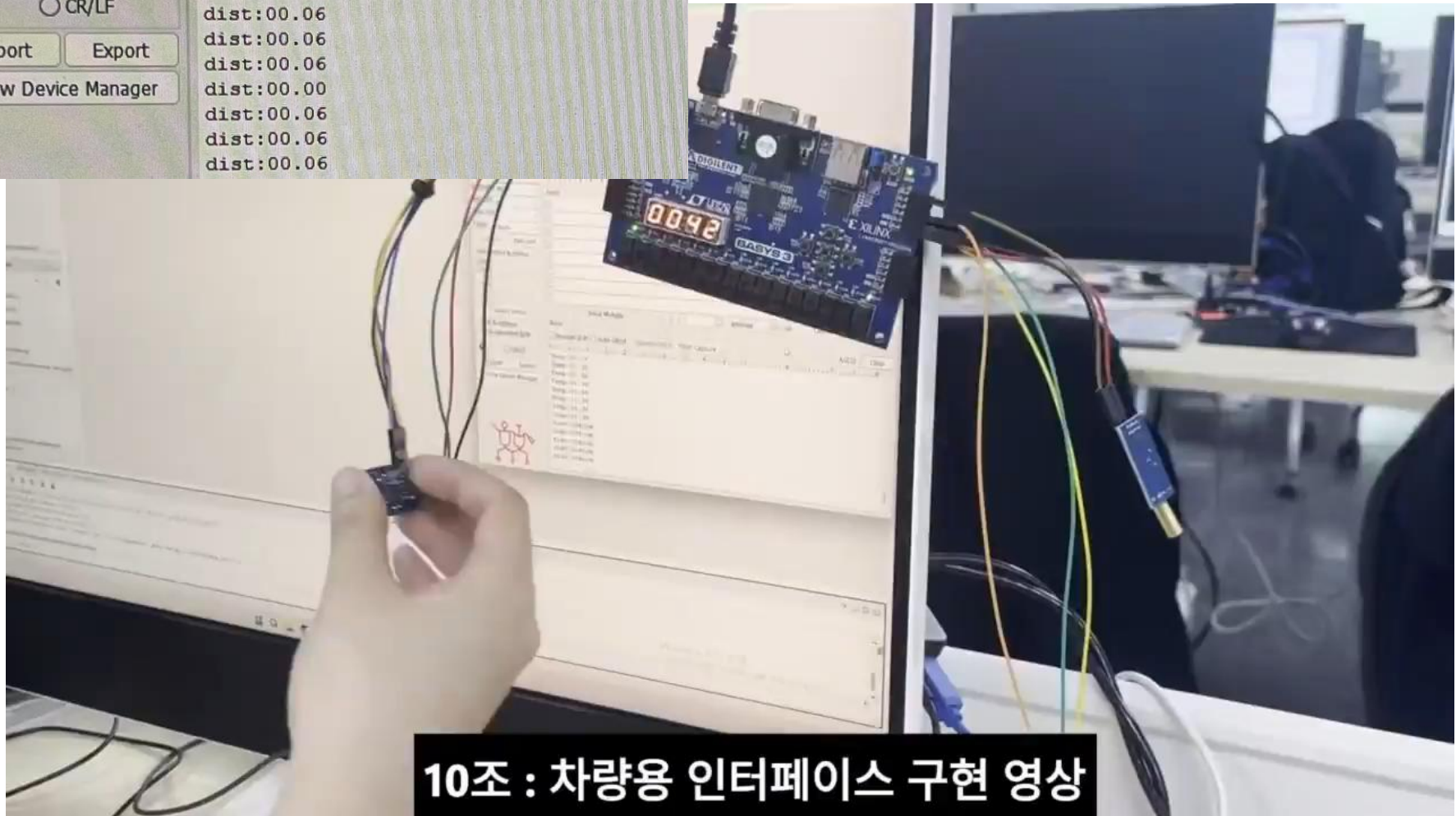
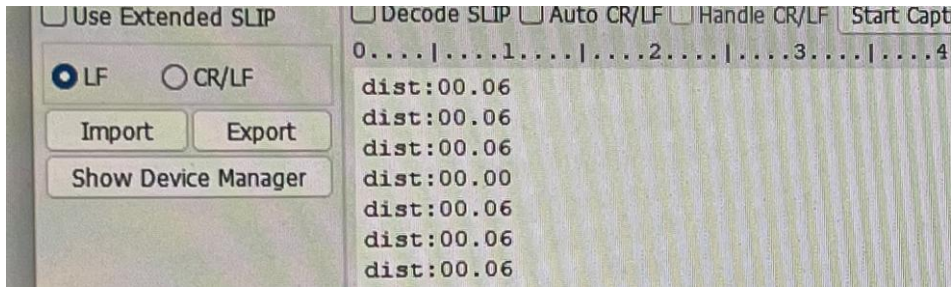
    BLINK_init(BUZZER, 49);
    LED_write(GPIOA, led_data);
    FND_init(FND, fnd_blink);
    FND_writeData(FND, fnd_shape);
    break;
```

```
case RIGHT_BLINK_STATE:
    ggambbak = 0b01;
    fnd_shape = RIGHT;
    if(Timer_read(TIMER2) == 0 && blink_flag == 0)
    {
        blink_flag = 1;
        led_data ^= ggambbak;
        fnd_blink = (led_data & ggambbak) == 0 ? POWER_OFF : POWER_ON;
    }
    else if(Timer_read(TIMER2) != 0) blink_flag = 0;

    delay(10);

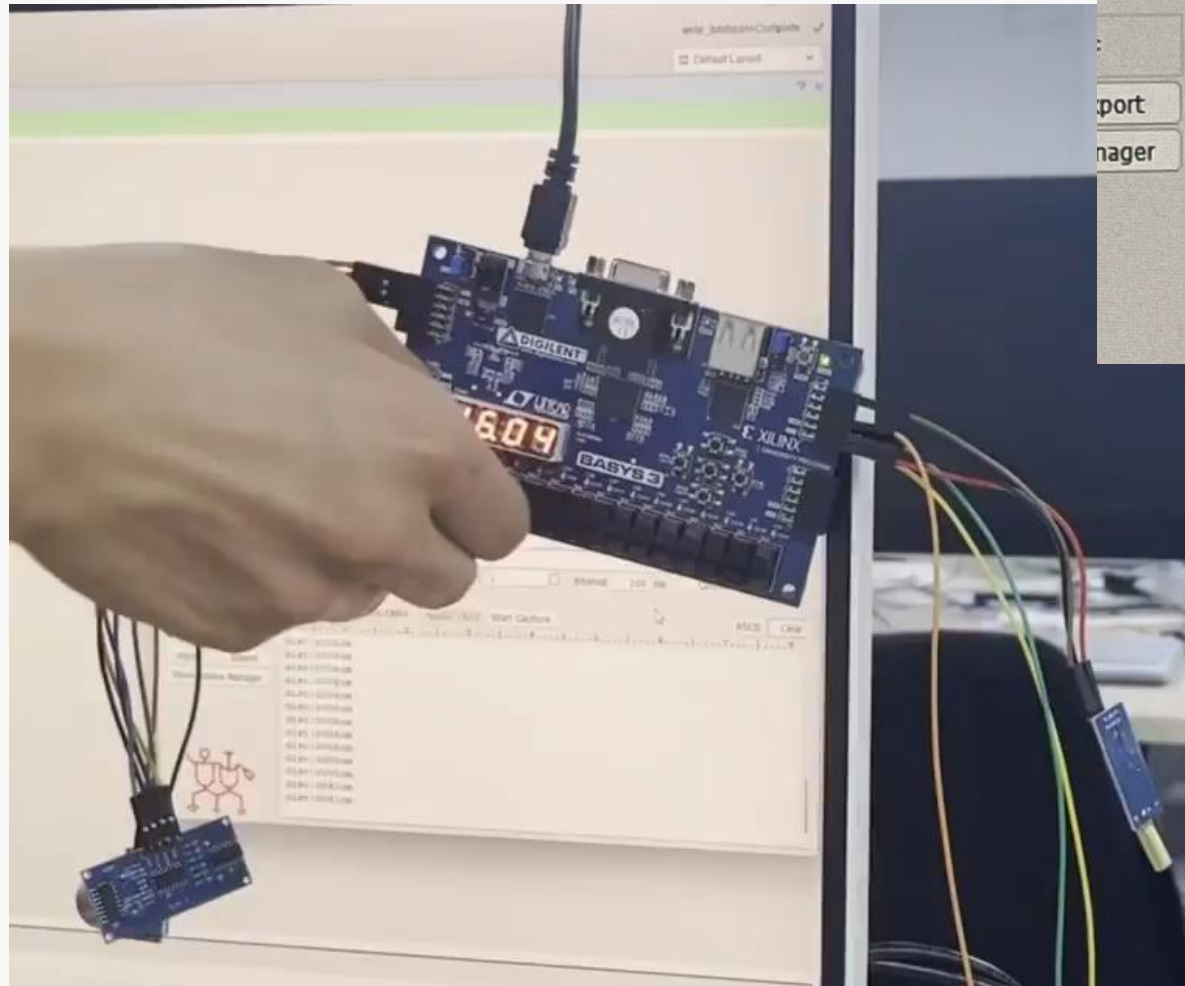
    BLINK_init(BUZZER, 49);
    LED_write(GPIOA, led_data);
    FND_init(FND, fnd_blink);
    FND_writeData(FND, fnd_shape);
    break;
```

# 03. Peripherals - Run Video



10조 : 차량용 인터페이스 구현 영상

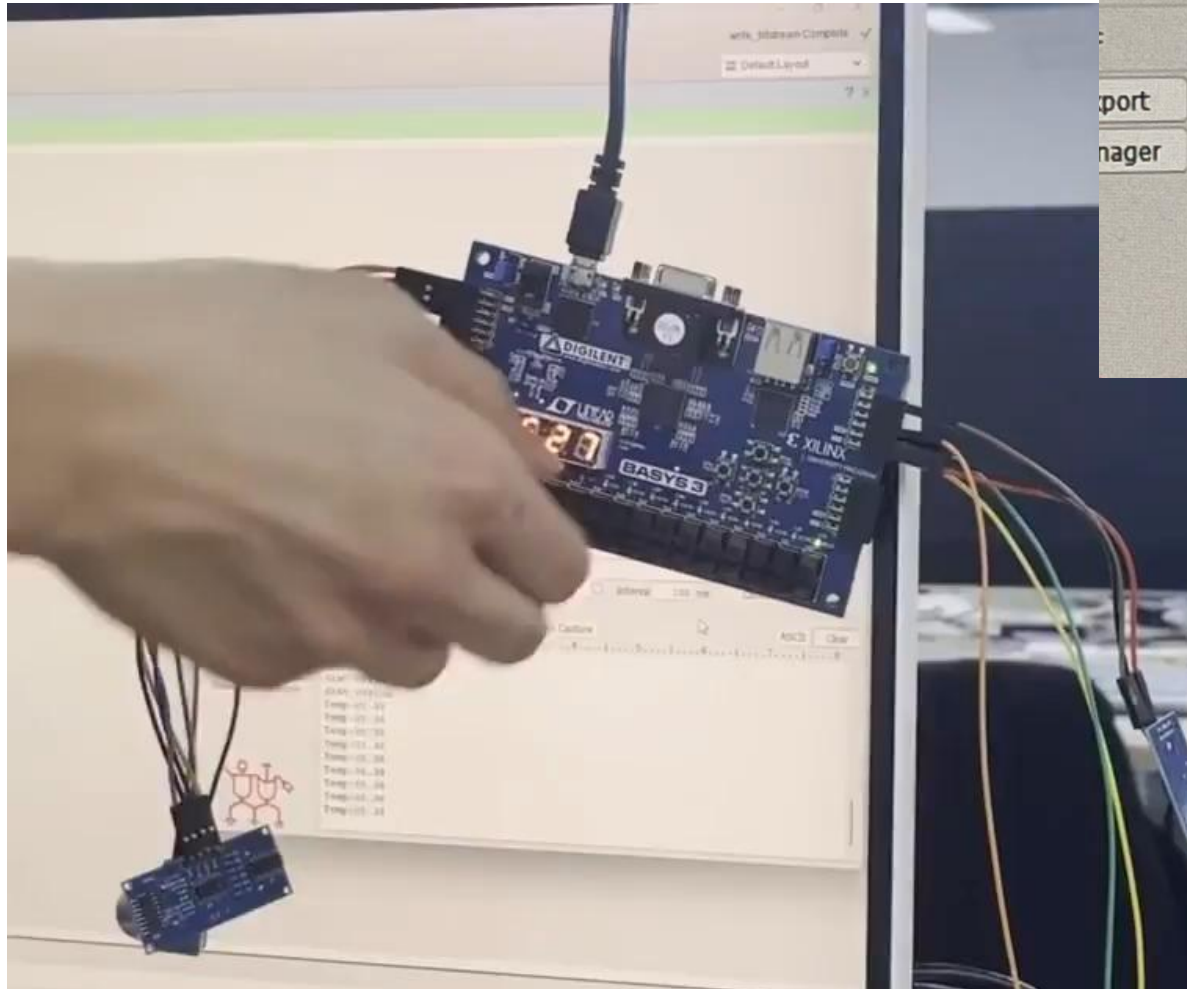
# 03. Peripherals - Run Video



Temp



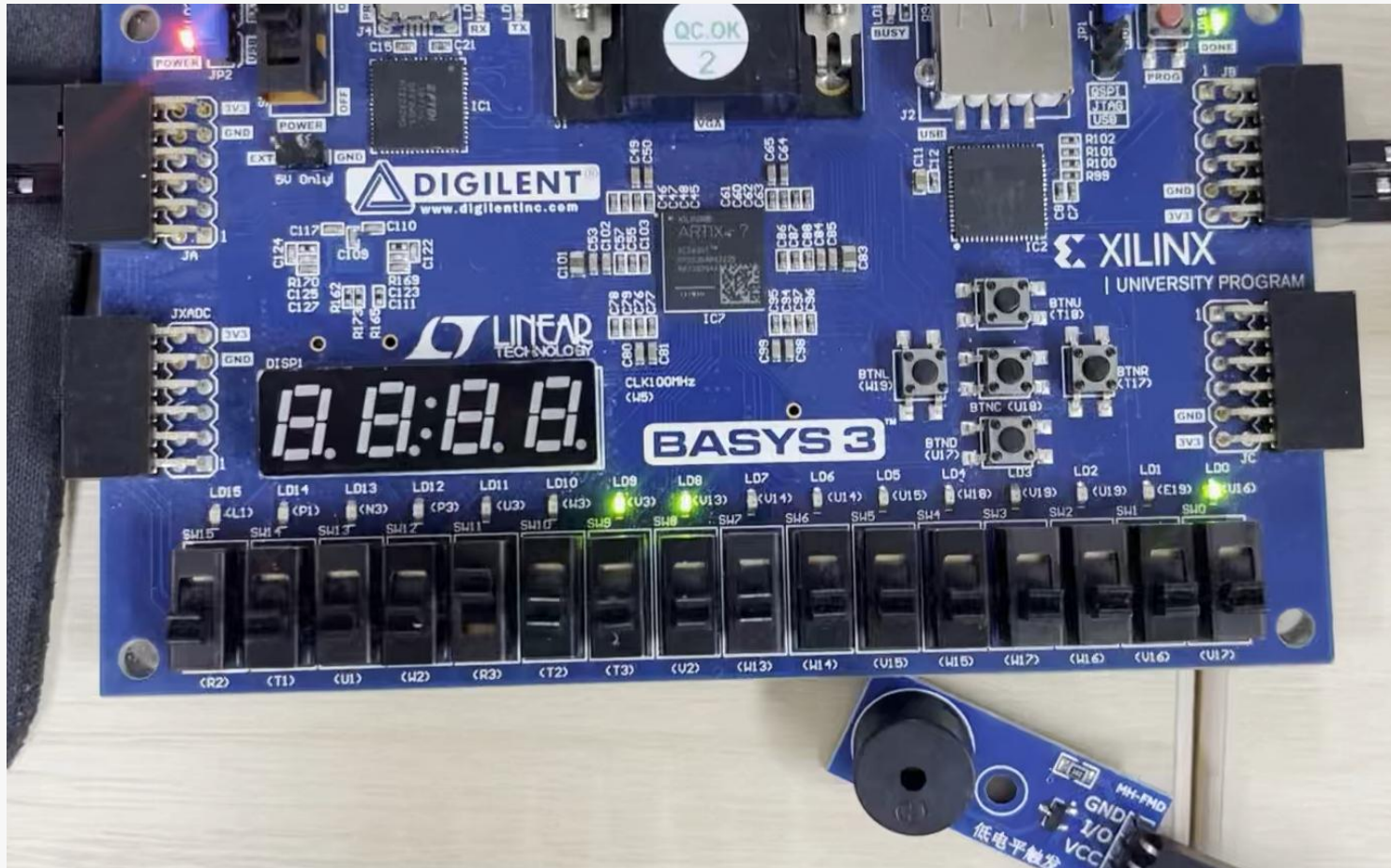
# 03. Peripherals - Run Video



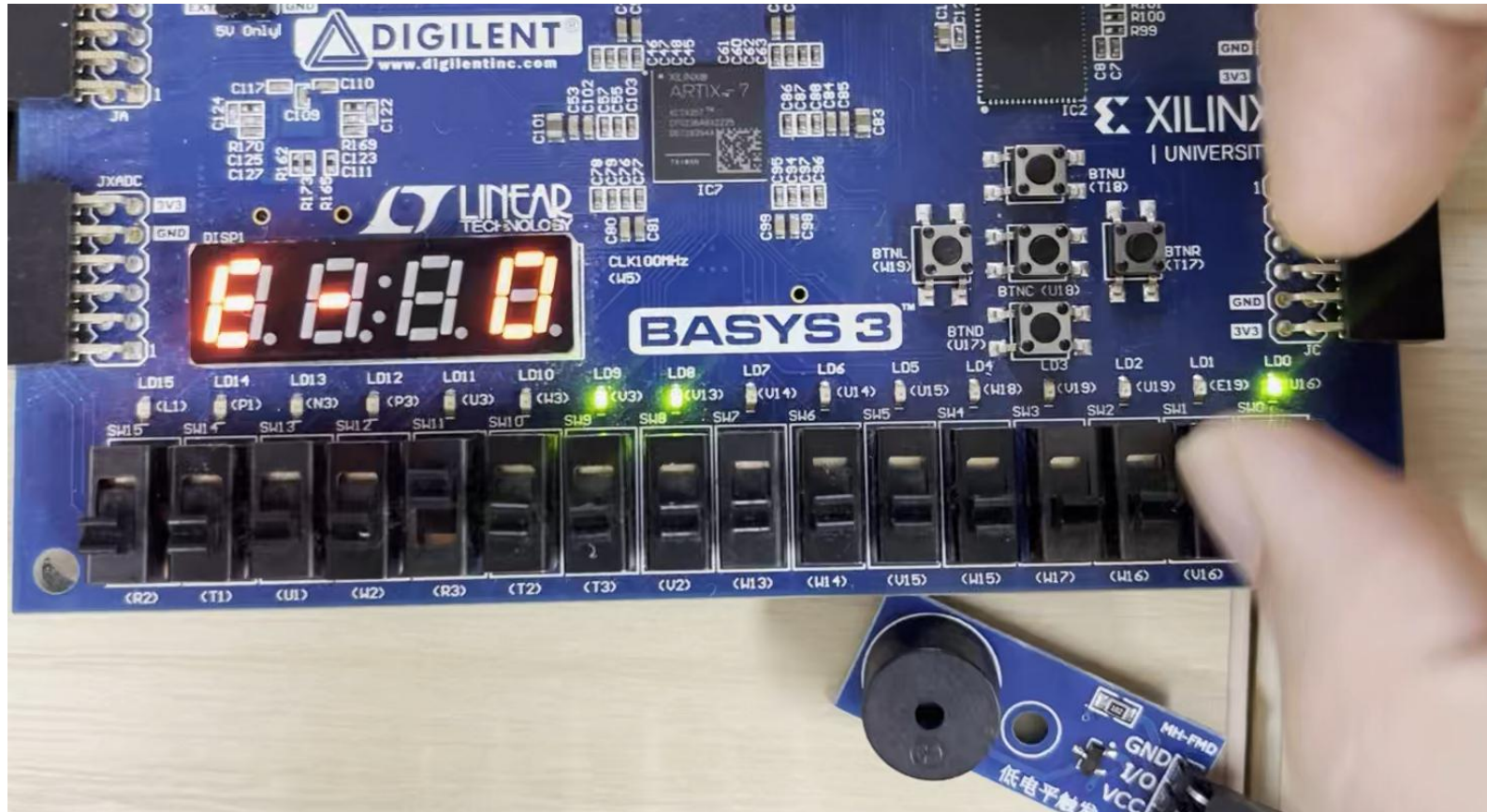
```
IP      ☐ Decode SLIP ☐ Auto CR/LF ☐ Har
0....|....1....|....2....|
Temp:18.18
Temp:18.18
Temp:18.18
Temp:18.18
Humidity:0039%
Humidity:0039%
Humidity:0039%
Humidity:0039%
```

Humidity

# 03. Peripherals - Run Video

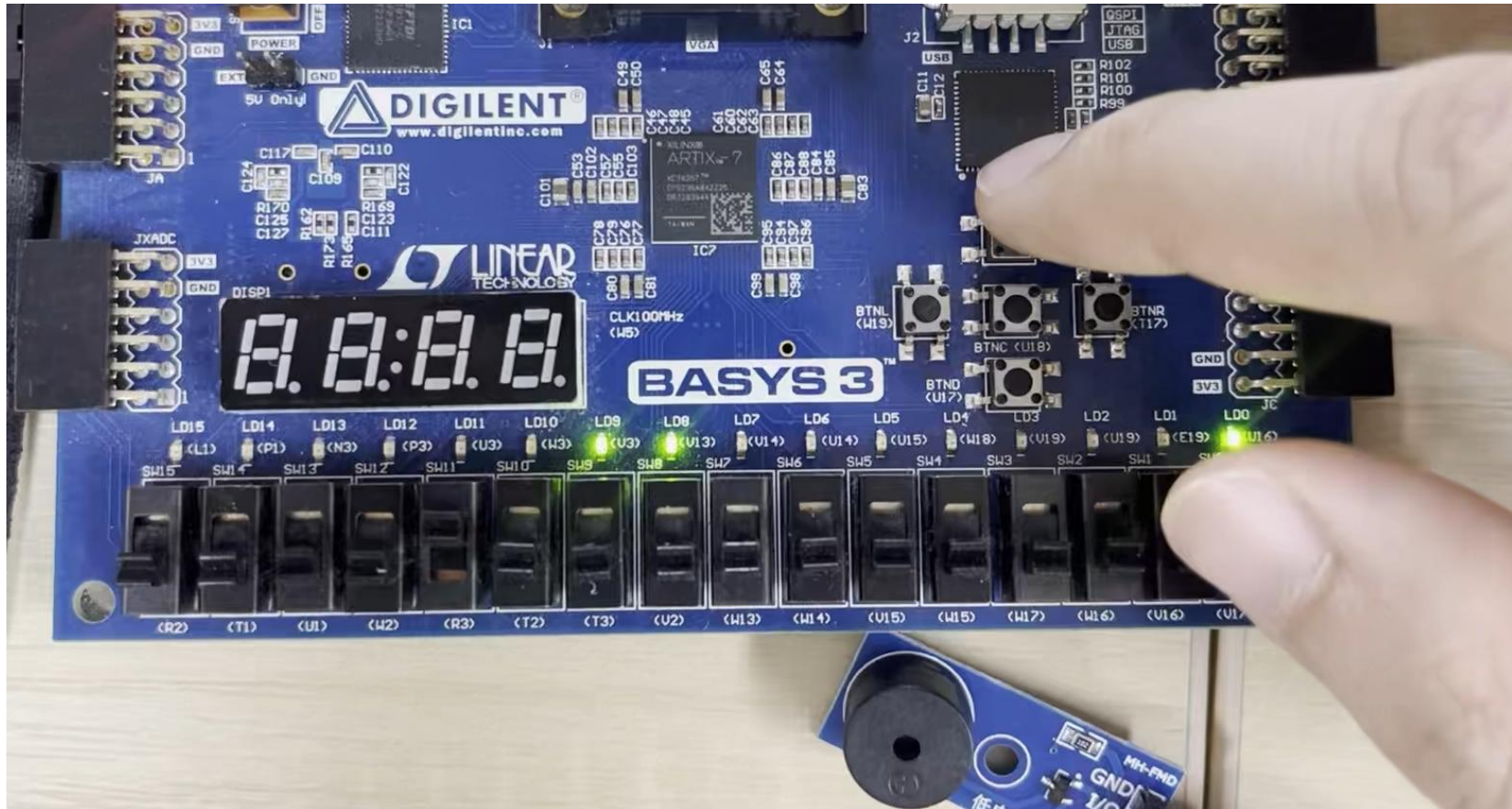


### 03. Peripherals - Run Video





# 03. Peripherals - Run Video



## 04. Conclusion - Trouble Shooting

### One Block

**State1**

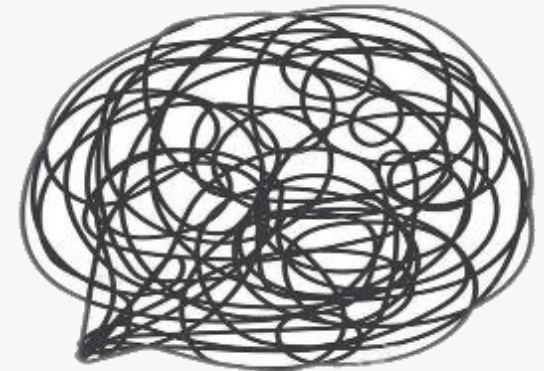
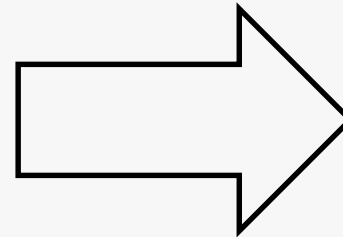
**State-Driven Operation**

**State2**

**State-Driven Operation**

**Stat3**

**State-Driven Operation**

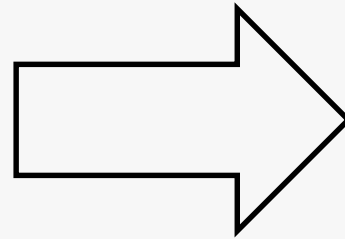


## 04. Conclusion - Trouble Shooting

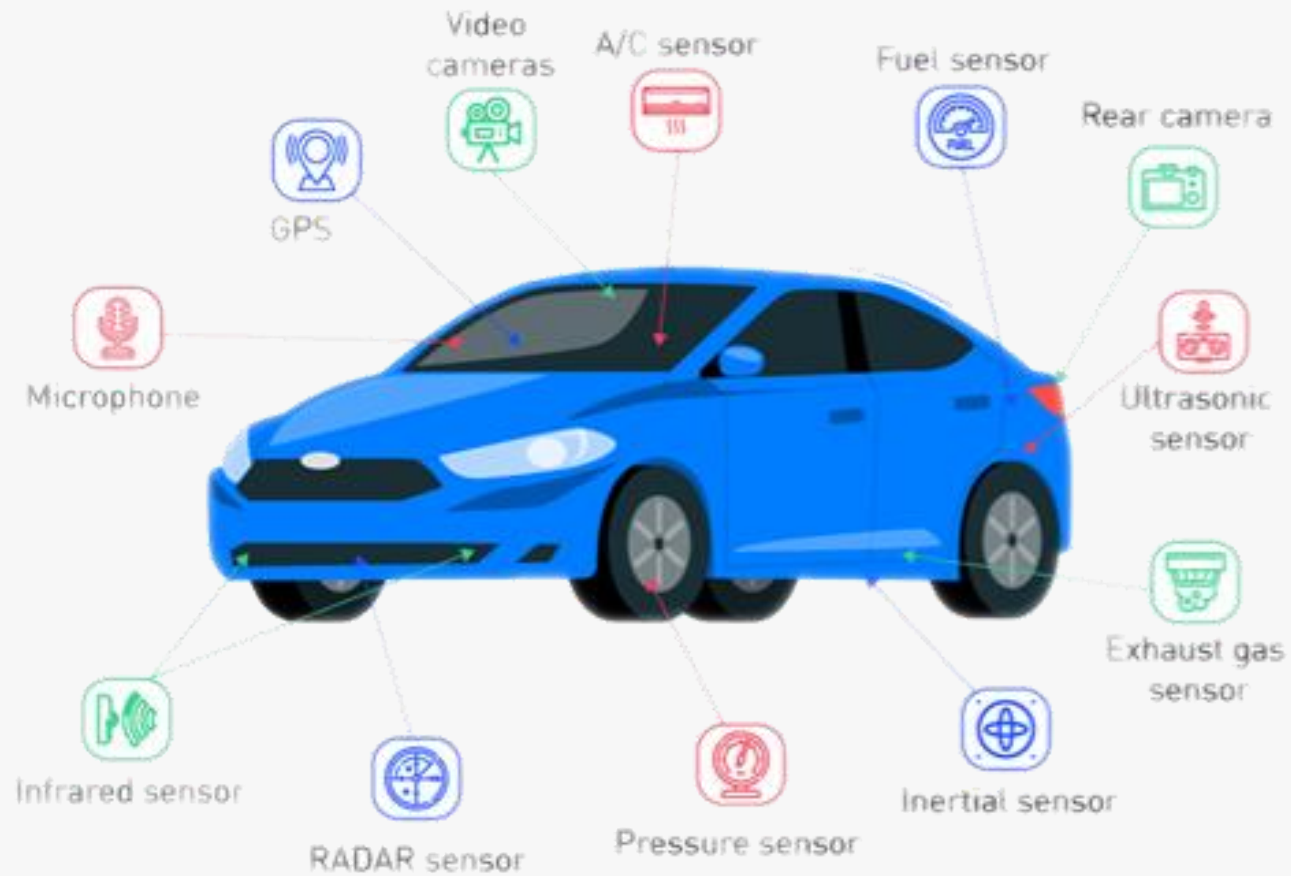
### Two Block

**State-Driven Operation**

**State Determination**



# 04. Conclusion



Thank you