Integrated Automotive Controller and Cluster Display System

Kyungpook National University

Microprocessor Capstone Design

Team 15 :: 유한솔, 이세인

Agenda

- 개발 과정
- 프로젝트 구조
  - Abstraction
  - Standardization
  - Implementation
  - Integration
- 논의사항 및 개선점
- 프로젝트 시연
- Q & A

개발 과정

개발 과정

Abstraction

추상화: 프로그램의 전체적인 구조는 어떻게 될 것인가?
어떤 기준으로 기능을 통합하고 분리할 것인가?

Standardization

표준화: 개별 기능 및 전체적인 workflow를 분석하여 표준화
프로젝트 코드의 신뢰성 향상을 위함

Implementation

구현: 개별 기능에 대한 개발 구상

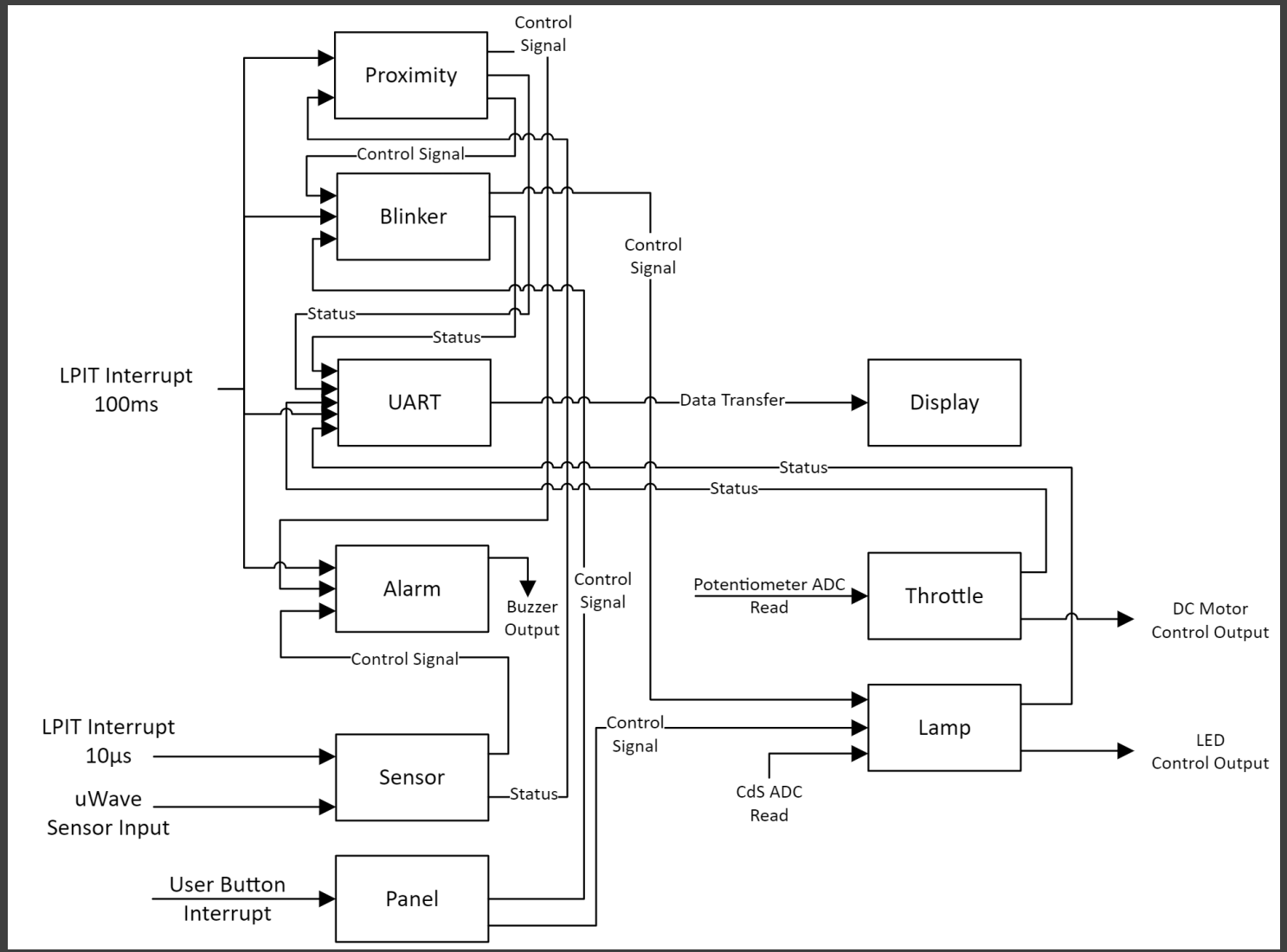
Integration

통합: 추상화 과정에서 구상한 내용과 표준화 과정에서
정의한 표준을 바탕으로 task flow를 단일화

Abstraction

추상화: 프로그램의 전체적인 구조는 어떻게 될 것인가?
어떤 기준으로 기능을 통합하고 분리할 것인가?

Abstraction (추상화)



- 9개의 Software Module로 구성
- 각 모듈은 왼쪽 그림과 같이 상호작용함
- 모듈 왼쪽으로 들어가는 화살표는 제어 및 데이터 입력,
  오른쪽으로 들어가는 화살표는 제어 및 데이터 출력을 의미함

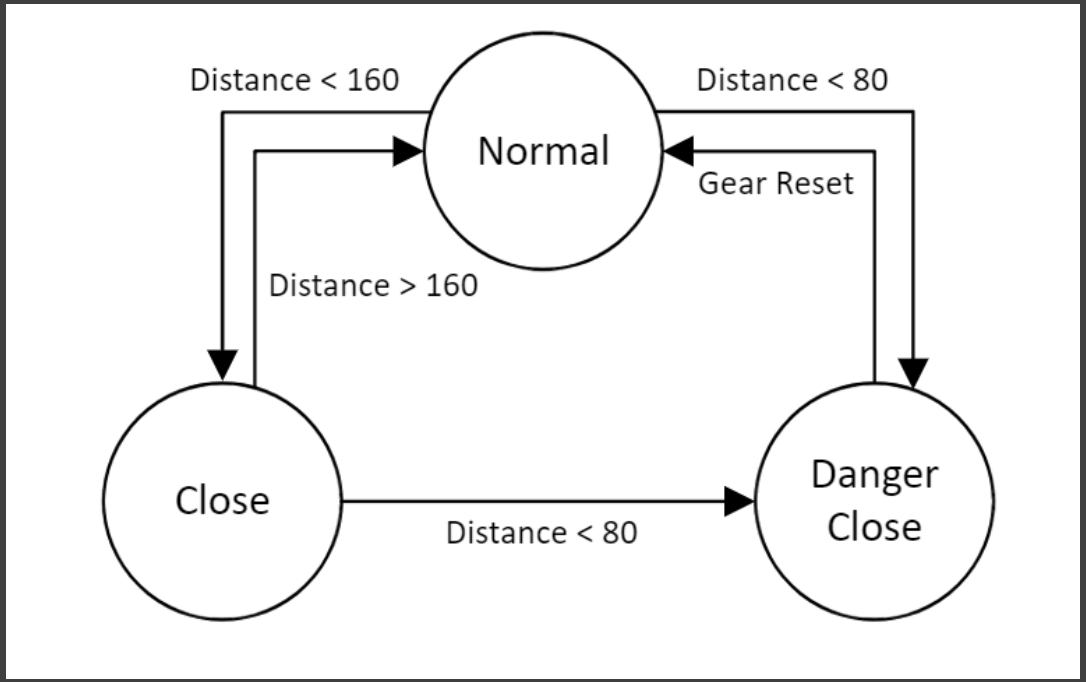
Software Modules (SWM)

- 제어기의 전체 기능을 세분화하여 구분

- 구분 기준:
  - 유사한 기능을 갖는 기능끼리 분류
  - 하나의 FSM으로 구성할 수 있는 기능
  - 유사한 기능을 수행하더라도 트리거 조건이 다르면 서로 다른 SWM으로 구분해야 함.

SWM Examples

- 유사한 기능끼리 분류: User Button Interrupt
- 하나의 FSM으로 표현 가능한 기능: 전방 근접 경보 (ProxWarning)



SWM Examples

- 트리거 조건에 따른 구분: Blinker Module과 Lamp Module
- 비슷한 기능을 수행하지만, 트리거 조건이 달라 서로 다른 모듈로 구분

- Lamp 모듈은 트리거 조건이 필요 없지만, Blinker 모듈은 0.5초마다 Trigger 되어 방향지시등을 On/Off 해 주어야 함
- Blinker 모듈에서 Control Signal을 생성하여 Lamp Module을 제어하는 방식으로 구현

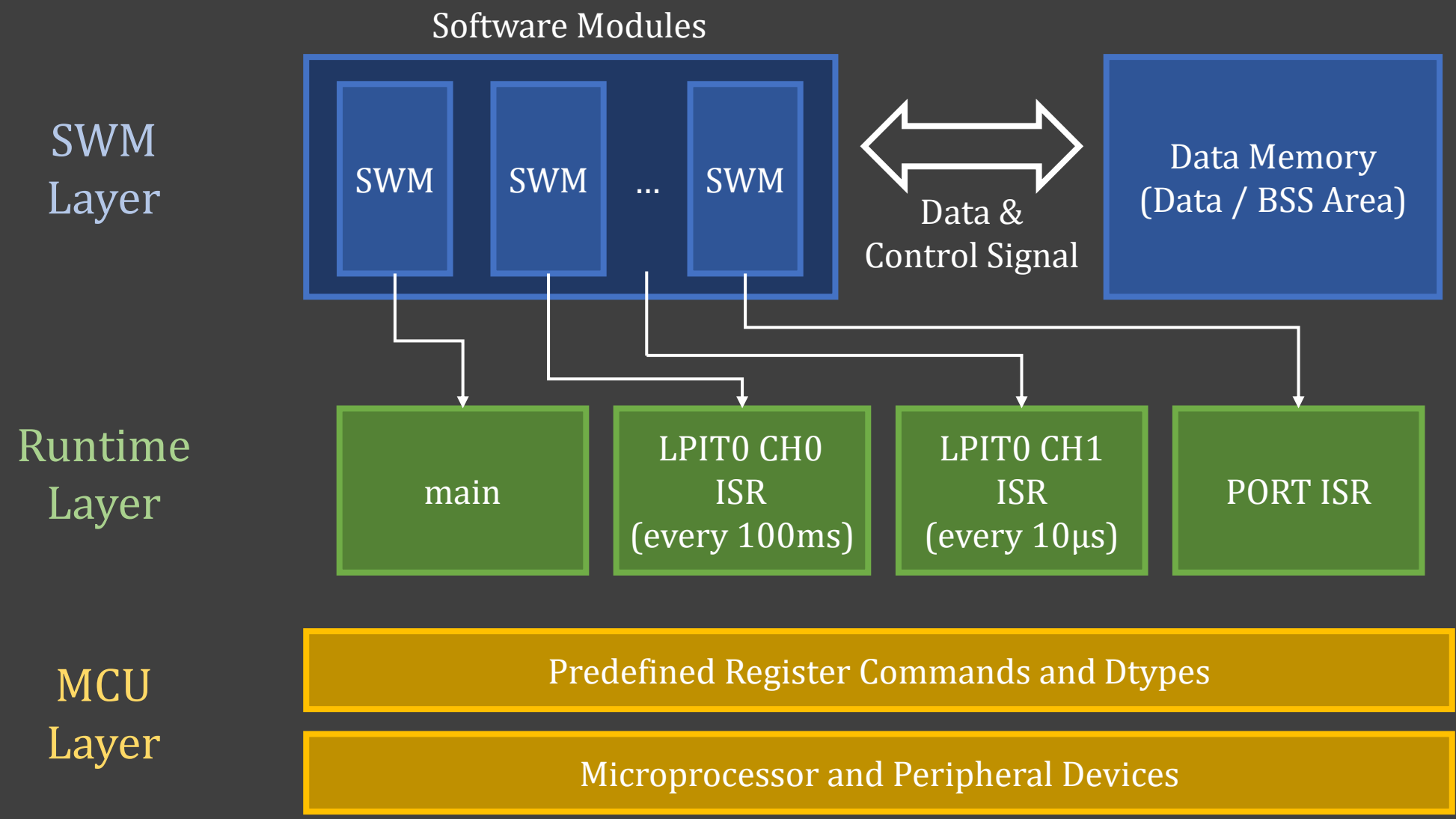
Standardization

표준화: 개별 기능 및 전체적인 workflow를 분석하여 표준화
프로젝트 코드의 신뢰성 향상을 위함

Reliability

- 소프트웨어의 신뢰성은 코드의 높은 가독성과 유지보수의 용이성에서 나옴

- 코드가 지나치게 복잡해지는 것을 막고, workflow를 쉽게 파악할 수 있도록 소프트웨어를 3개의 계층으로 구분함


Software Modules
SWM Layer
SWM
SWM
...
SWM
Data &
Control Signal
Data Memory
(Data / BSS Area)
Runtime Layer
main
LPIT0 CH0
ISR
(every 100ms)
LPIT0 CH1
ISR
(every 10µs)
PORT ISR
MCU Layer
Predefined Register Commands and Dtypes
Microprocessor and Peripheral Devices

SWM Layer

- 제어기의 개별 기능을 Software Module로 정의한 계층

- 프로그램의 구조와 Call Stack 단순화를 위해 각 모듈 간의 계층 구조는 정의하지 않음.
  즉, 모든 모듈은 동일한 계층에 위치함

- 각 모듈들은 단일 메모리 영역 (Data 및 BSS 영역)을 공유
  공유 메모리 영역을 통해 모듈 간 데이터 및 제어 신호 전송

Runtime Layer

- 소프트웨어 모듈이 호출되는 계층
- 소프트웨어 모듈이 실행되는 시점을 결정함
- 함수의 실행 시점 및 트리거 조건에 따라 4개의 영역으로 구분

  - main: 트리거 조건이 따로 없어 항상 실행되는 모듈
  - LPIT ISR: 일정 시간마다 실행되어야 하는 모듈
    실행 주기에 따라 10μs, 100ms ISR로 구분
  - PORT ISR: 사용자 입력에 반응하여 실행되어야 하는 모듈

- 각 Runtime 함수 내에서 SWM 함수를 호출하는 방식으로 SWM을 실행함.

MCU Layer

- 소스 코드 **컴파일 및 실제 실행**을 담당하는 계층
- SWM에서 사용할 **레지스터 설정 코드**와 Data memory에 저장될 변수의 **데이터 타입**을 정의함
- 사전 정의된 레지스터 설정 명령을 전처리 단계에서 실제 코드로 변환

```
#define GEAR_BUTTON_READ        (PORTC->ISFR & (1 << 8))
#define B_LEFT_BUTTON_READ      (PORTC->ISFR & (1 << 9))
#define B_RIGHT_BUTTON_READ     (PORTC->ISFR & (1 << 10))
#define LAMP_BUTTON_READ        (PORTC->ISFR & (1 << 11))
#define FTM0_CH2_PWM_OFF        FTM0->SC &= ~(FTM_SC_PWMEN4_MASK | FTM_SC_PWMEN5_MASK)
#define BLUE_LED_ON             PTD->PCOR |= 1 | (1 << 0)
#define BLUE_LED_OFF            PTD->PSOR |= 1 | (1 << 0)
#define RED_LED_ON              PTD->PCOR |= 1 | (1 << 15)
#define RED_LED_OFF             PTD->PSOR |= 1 | (1 << 15)
#define GREEN_LED_ON            PTD->PCOR |= 1 | (1 << 16)
#define GREEN_LED_OFF           PTD->PSOR |= 1 | (1 << 16)
#define B_LEFT_ON               PTD->PCOR |= 1 | (1 << 8)
#define B_LEFT_OFF              PTD->PSOR |= 1 | (1 << 8)
#define B_RIGHT_ON              PTD->PCOR |= 1 | (1 << 9)
#define B_RIGHT_OFF             PTD->PSOR |= 1 | (1 << 9)
#define BUZZER_ON               PTD->PSOR |= 1 | (1 << 7)
#define BUZZER_OFF              PTD->PCOR |= 1 | (1 << 7)
#define BUZZER_TOGGLE           PTD->PTOR |= 1 | (1 << 7)
#define PIEZO_ON                PTD->PSOR |= 1 | (1 << 10)
#define PIEZO_OFF               PTD->PCOR |= 1 | (1 << 10)
#define UWAVE_TRIG_SEND         PTD->PSOR |= (1 << 5)
#define UWAVE_TRIG_STOP         PTD->PCOR |= (1 << 5)
#define UWAVE_ECHO_READ         PTD->PDIR & (1 << 6)
```

```
/*
_UART_DATA_TYPE:
4byte union type
use .value to access to whole value
use .elements.blinker / gear / speed to access individual values
*/
typedef union
{
    uint32_t value;
    struct
    {
        uint32_t lamp : 2;
        uint32_t blinker : 2;
        uint32_t gear : 2;
        uint32_t speed : 10;
        uint32_t prox : 2;
    } elements;
} _UART_DATA_TYPE;
```

Implementation

구현: 개별 기능에 대한 개발

Implementation (구현)

- 개별 모듈의 동작은 소프트웨어 모듈 함수 내부에 구현

```
void SWM_Blinker(void);
void SWM_Lamp(void);
void SWM_Alarm(void);
void SWM_Sensor(void);
void SWM_UART(void);
void SWM_Throttle(void);
void SWM_Panel(void);
void SWM_ProxWarning(void);
```

SWM_Blinker

```
void SWM_Blinker(void)
{
    if (blinker_mode.value == 0)
    {
        blinker_counter = 0;
        blinker.value = 0;
    }
    else
    {
        if (blinker_counter < 5)
        {
            if (blinker_mode.elements.left)
                blinker.elements.left = 1;
            if (blinker_mode.elements.right)
                blinker.elements.right = 1;
        }
        else
            blinker.value = 0;
        if (++blinker_counter > 9)
            blinker_counter = 0;
    }
}
```

- 방향지시등 제어 신호 생성을 담당
- 버튼 입력에 따라 0.5초마다 방향지시등을 점멸시키는 제어 신호 생성

SWM_Lamp

```
void SWM_Lamp(void)
{
    // Read Cds
    if (lamp_mode == AUTO)
    {
        convertAdcChan(14);
        while (adc_complete() == 0)
            ;
        cdsResult = read_adc_chx();
    }

    // gear
    if (gear == P || proximity_warning > 1)
    {
        RED_LED_ON;
        GREEN_LED_OFF;
    }
    else if (gear == R)
    {
        RED_LED_ON;
        GREEN_LED_ON;
    }
    else
    {
        RED_LED_OFF;
        GREEN_LED_OFF;
    }
```

```
// lamp_mode
if (lamp_mode == HEAD_LAMP)
    lamp = HEAD_LAMP;
else if (lamp_mode == POS_LAMP)
    lamp = POS_LAMP;
else if (lamp_mode == AUTO)
    /*  Conducts software Schmitt trigger debouncing
        Centerd at: 1500, 2500
        Offset: 300 */
    if (lamp_auto_trig_high)
    {
        lamp = OFF;
        if (cdsResult < 2200) // 2500 - 300
            lamp_auto_trig_high = 0;
        if (cdsResult < 1200) // 1500 - 300
            lamp_auto_trig_low = 0;
    }
    else if (lamp_auto_trig_low)
    {
        lamp = POS_LAMP;
        if (cdsResult > 2800) // 2500 + 300
            lamp_auto_trig_high = 1;
        if (cdsResult < 1200)
            lamp_auto_trig_low = 0;
    }
    else
    {
        lamp = HEAD_LAMP;
        if (cdsResult > 2800)
            lamp_auto_trig_high = 1;
        if (cdsResult > 1800) // 1500 + 300
            lamp_auto_trig_low = 1;
    }
else
    lamp = OFF;
```

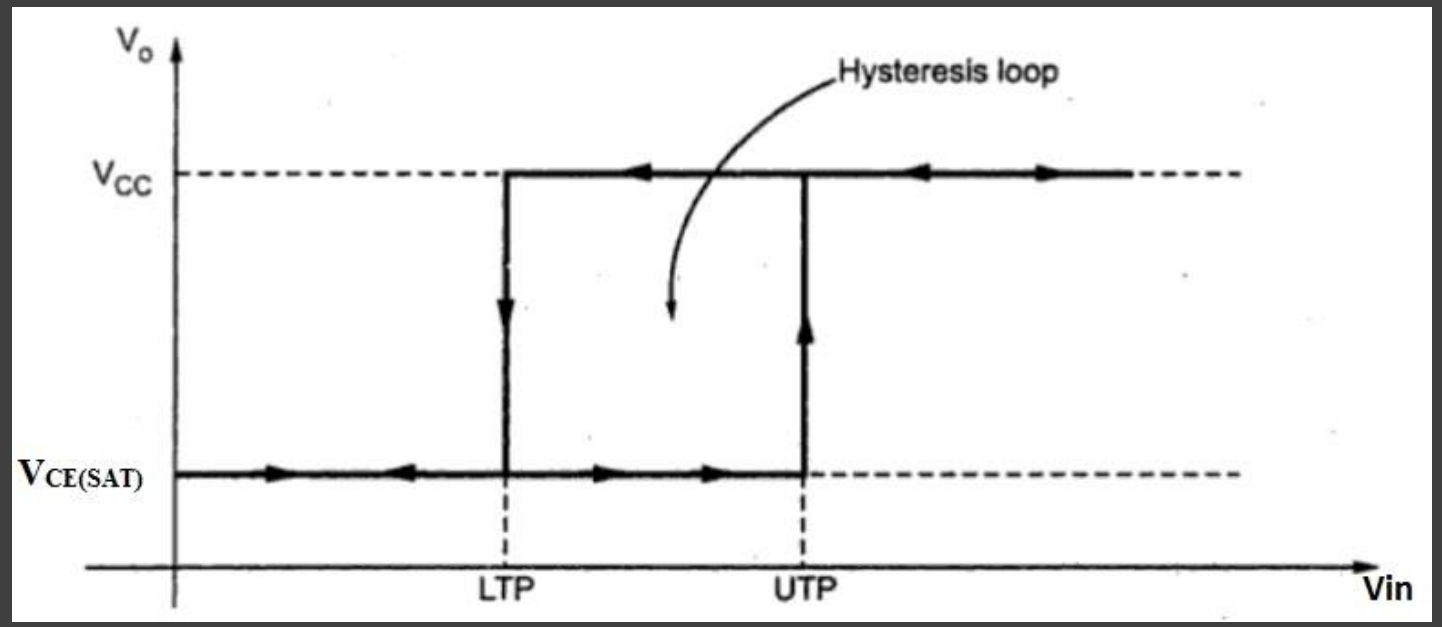
```
// lamp
switch (lamp)
{
case POS_LAMP:
    HEAD_LAMP_OFF;
    POS_LAMP_ON;
    break;
case HEAD_LAMP:
    HEAD_LAMP_ON;
    POS_LAMP_ON;
    break;
default:
    HEAD_LAMP_OFF;
    POS_LAMP_OFF;
    break;
}

// blinker
if (blinker.elements.left)
    B_LEFT_ON;
else
    B_LEFT_OFF;
if (blinker.elements.right)
    B_RIGHT_ON;
else
    B_RIGHT_OFF;
if (blinker.value != 0)
    PIEZO_ON;
else
    PIEZO_OFF;
```

- 전조등,
  방향지시등,
  후미등 제어 담당
- 전조등 자동 모드
  제어 코드 포함

SWM_Lamp

- CdS ADC read 과정에 Hysteresis 특성 추가
  경계 지점에서 on/off가 빠르게 전환하면서
  LED가 깜빡거리는 현상 방지




```
// lamp_mode
if (lamp_mode == HEAD_LAMP)
    lamp = HEAD_LAMP;
else if (lamp_mode == POS_LAMP)
    lamp = POS_LAMP;
else if (lamp_mode == AUTO)
    /*  Conducts software Schmitt trigger debouncing
        Centerd at: 1500, 2500
        Offset: 300 */
    if (lamp_auto_trig_high)
    {
        lamp = OFF;
        if (cdsResult < 2200) // 2500 - 300
            lamp_auto_trig_high = 0;
        if (cdsResult < 1200) // 1500 - 300
            lamp_auto_trig_low = 0;
    }
    else if (lamp_auto_trig_low)
    {
        lamp = POS_LAMP;
        if (cdsResult > 2800) // 2500 + 300
            lamp_auto_trig_high = 1;
        if (cdsResult < 1200)
            lamp_auto_trig_low = 0;
    }
    else
    {
        lamp = HEAD_LAMP;
        if (cdsResult > 2800)
            lamp_auto_trig_high = 1;
        if (cdsResult > 1800) // 1500 + 300
            lamp_auto_trig_low = 1;
    }
else
    lamp = OFF;
```

SWM_Alarm

```
void SWM_Alarm(void)
{
    if (gear == R)
    {
        if (uwave_distance < 80)
            BUZZER_ON;
        else if (uwave_distance < 160)
            BUZZER_TOGGLE;
        else if (uwave_distance < 240)
        {
            if (++buzzer_counter > 2)
            {
                BUZZER_TOGGLE;
                buzzer_counter = 0;
            }
        }
        else
            BUZZER_OFF;
    }
    else if (proximity_warning > 0)
    {
        if (buzzer_counter > 0)
        {
            BUZZER_TOGGLE;
            buzzer_counter--;
        }
        else
            BUZZER_OFF;
    }
    else
    {
        buzzer_counter = 0;
        BUZZER_OFF;
    }
}
```

- 버저 제어 담당
- Sensor 모듈이 읽어온 값을 바탕으로
  후진 시에는 후방 근접 경고, 주행 시에는 전방 근접 경고로
  작동

SWM_UART

```
void SWM_UART(void)
{
    d_send.elements.lamp = lamp;
    if (gear == P || proximity_warning == 2)
        d_send.elements.speed = 0;
    else
        d_send.elements.speed = vrResult >> 2; // squashing 12 bit data into 10 bit
    d_send.elements.gear = gear;
    d_send.elements.blinker = blinker.value;
    d_send.elements.prox = proximity_warning;
    LPUART1_transmit_word(d_send.value);
}
```

- UART 전송 신호 생성 및 통신 담당
- UART 전송 신호는 4바이트 (32비트)로 구성되어
  계기판 구동에 필요한 정보 (속도, 전조등, 방향지시등, 전방 근접 경고) 전송

SWM_Sensor

```
void SWM_Sensor(void)
{
    if (gear == R || gear == D)
    {
        if (uwave_counter == 0)
            UWAVE_TRIG_SEND;
        else if (uwave_counter == 2)
            UWAVE_TRIG_STOP;
        if (UWAVE_ECHO_READ)
            uwave_high++;

        uwave_counter++;
        if (uwave_counter >= 6000) // 10us * 6000 = 60ms
        {
            uwave_counter = 0;
            uwave_distance = uwave_high;
            uwave_high = 0;
        }
    }
}
```

- 초음파 센서 구동 담당

SWM_Throttle

```
void SWM_Throttle(void)
{
    // Read VR
    convertAdcChan(13);
    while (adc_complete() == 0)
        ;
    vrResult = read_adc_chx();

    // Generating PWM signal
    if (gear == P)
        FTM0_CH2_PWM_OFF;
    else if (gear == R)
        FTM0_CH2_PWM(4000 + (double)(vrResult) * 0.9768); // 0.9768 = 4000 / 4095
    else if (gear == D)
    {
        if (proximity_warning > 1)
            FTM0_CH2_PWM(4000);
        else
            FTM0_CH2_PWM(4000 - (double)(vrResult) * 0.9768); // 0.9768 = 4000 / 4095
    }
}
```

- 가변저항 값 읽기 및 DC 모터 구동 담당

SWM_Panel

```
void SWM_Panel(void)
{
    if (GEAR_BUTTON_READ != 0) // gear
    {
        if (++gear > 3)
            gear = P;
    }

    if (B_LEFT_BUTTON_READ != 0) // blinker left
        blinker_mode.elements.left ^= 1;

    if (B_RIGHT_BUTTON_READ != 0) // blinker right
        blinker_mode.elements.right ^= 1;

    if (LAMP_BUTTON_READ != 0)
    {
        if (++lamp_mode > 3)
            lamp_mode = OFF;
    }
}
```

- 버튼 입력 처리 담당

SWM_ProxWarning

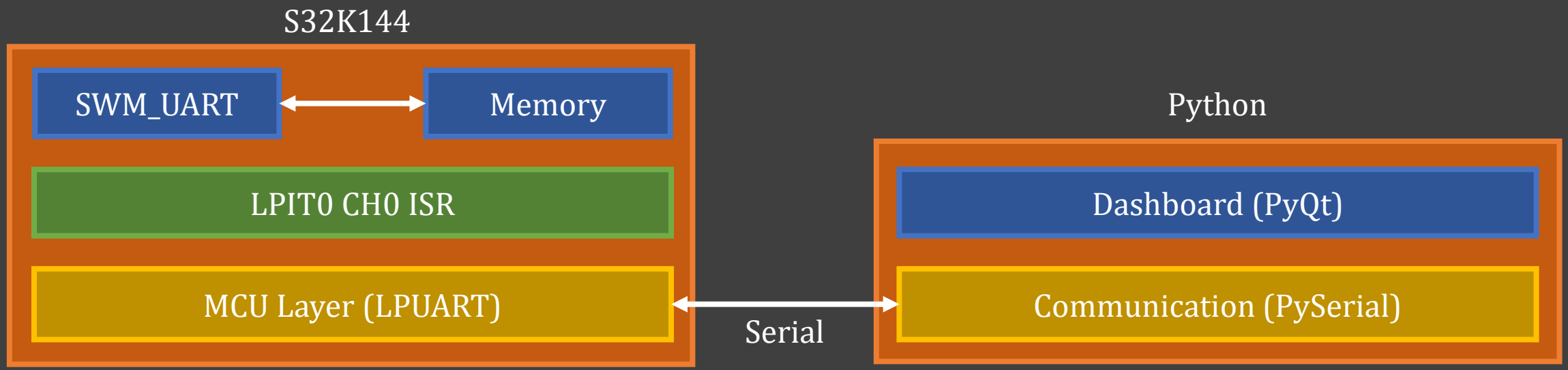
```
void SWM_ProxWarning(void)
{
    if (gear == D)
    {
        if (uwave_distance < 80) {
            if (proximity_counter > 5)
            {
                proximity_counter = 0;
                blinker_mode.value = 3;
                if (proximity_warning < 2)
                    buzzer_counter = 8;
                proximity_warning = 2;
            }
            else
                proximity_counter++;
        }
        else if (uwave_distance < 160 && proximity_warning < 2)
        {
            if (proximity_warning == 0)
                buzzer_counter = 4;
            proximity_warning = 1;
            proximity_counter = 0;
        }
        else if (proximity_warning < 2)
        {
            proximity_warning = 0;
            proximity_counter = 0;
        }
    }
    else
        proximity_warning = 0;
}
```

- 전방 근접 경고 신호 생성 담당
- 전방에 물체가 감지되면 경고음 2회 발생 및 계기판에 주의 표시
- 전방의 가까운 곳에 물체가 감지되면 경고음 4회 발생 및 모터 정지, 계기판에 경고 표시

Dashboard

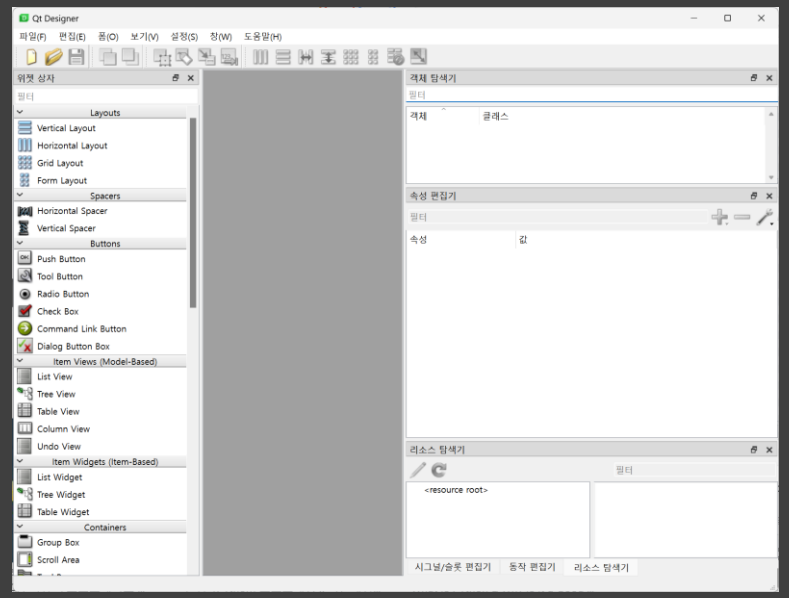
- PySerial
  Serial 통신을 담당하는 Python 패키지

- Qt
  크로스플랫폼 GUI 프레임워크
  Python으로 바인딩된 버전인 PyQt 사용


S32K144
SWM_UART
Memory
LPIT0 CH0 ISR
MCU Layer (LPUART)
Serial
Python
Dashboard (PyQt)
Communication (PySerial)

Dashboard

Qt 6 Desinger



dashboard.ui

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>DashBoard</class>
 <widget class="QMainWindow" name="DashBoard">
  <property name="enabled">
   <bool>true</bool>
  </property>
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>640</width>
    <height>481</height>
   </rect>
  </property>
  <property name="sizePolicy">
   <sizepolicy hsizetype="Fixed" vsizetype="Fixed">
    <horstretch>0</horstretch>
    <verstretch>0</verstretch>
   </sizepolicy>
  </property>
  <property name="windowTitle">
   <string>Dashboard</string>
  </property>
  <property name="windowOpacity">
   <double>1.000000000000000</double>
  </property>
  <property name="layoutDirection">
   <enum>Qt::LeftToRight</enum>
  </property>
  <property name="styleSheet">
   <string notr="true">background-color: rgb(50, 50, 50)</string>
  </property>
  <property name="dockNestingEnabled">
   <bool>false</bool>
```

pyuic

dashboard.py

```python
# Form implementation generated from reading ui file '.\dashboard.ui'
#
# Created by: PyQt6 UI code generator 6.6.0
#
# WARNING: Any manual changes made to this file will be lost when pyuic6 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt6 import QtCore, QtGui, QtWidgets


class Ui_DashBoard(object):
    def setupUi(self, DashBoard):
        DashBoard.setObjectName("DashBoard")
        DashBoard.setEnabled(True)
        DashBoard.resize(640, 481)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Policy.Fixed, QtWidgets.QSizePolicy.Policy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(DashBoard.sizePolicy().hasHeightForWidth())
        DashBoard.setSizePolicy(sizePolicy)
        DashBoard.setWindowOpacity(1.0)
        DashBoard.setLayoutDirection(QtCore.Qt.LayoutDirection.LeftToRight)
        DashBoard.setStyleSheet("background-color: rgb(50, 50, 50)")
        DashBoard.setDockNestingEnabled(False)
        self.centralwidget = QtWidgets.QWidget(parent=DashBoard)
        self.centralwidget.setObjectName("centralwidget")
        self.label_blinker_left = QtWidgets.QLabel(parent=self.centralwidget)
        self.label_blinker_left.setGeometry(QtCore.QRect(30, 400, 50, 50))
        font = QtGui.QFont()
        font.setPointSize(36)
        self.label_blinker_left.setFont(font)
        self.label_blinker_left.setStyleSheet("color: white")
        self.label_blinker_left.setText("")
        self.label_blinker_left.setPixmap(QtGui.QPixmap(":/resources/resources/left_arrow_off.png"))
        self.label_blinker_left.setWordWrap(False)
        self.label_blinker_left.setObjectName("label_blinker_left")
        self.label_gear = QtWidgets.QLabel(parent=self.centralwidget)
        self.label_gear.setGeometry(QtCore.QRect(260, 240, 121, 71))
        font = QtGui.QFont()
        font.setPointSize(24)
        self.label_gear.setFont(font)
        self.label_gear.setStyleSheet("background-color: rgba(0, 0, 0, 0%);\n"
"color: white;")
        self.label_gear.setText("None")
        self.label_gear.setAlignment(QtCore.Qt.AlignmentFlag.AlignCenter)
```

Dashboard

dashboard.py

import

display.py

```python
from serial import Serial, serialutil
from dashboard import Ui_DashBoard
from PyQt6.QtCore import QTimer, Qt
from PyQt6.QtWidgets import QApplication, QMainWindow
from PyQt6.QtGui import QPixmap, QTransform

# 여기에 포트 번호 입력
port = 'COM13'

try:
    s_s32k = Serial(port=port, timeout=0.15)
except serialutil.SerialException:
    print("Connection to port " + port + " failed.")
    exit(1)

class Dashboard(QMainWindow, Ui_DashBoard):

    def __init__(self):
        super().__init__()
        self.setupUi(self)

        # UART read Timer
        self.UART_timer = QTimer()
        self.UART_timer.setInterval(50)
        self.UART_timer.timeout.connect(self.UART_action)
        self.UART_timer.start()

        # Pixmap & Transform
        self.transform_speedometer_rotate = QTransform()
        self.pixmap_speedometer = QPixmap(".\\resources/speedometer.png")
        self.label_speedometer_analog.setPixmap(self.pixmap_speedometer)
        self.label_speedometer_bg.setPixmap(QPixmap(".\\resources/speedometer_bg.png"))

    def UART_action(self):
        try:
            (lamp, blinker, gear, speed, prox, errorNo) = self.UART_input_parser(int.from_bytes(s_s32k.read(size=4)))
        except serialutil.SerialException:
            (lamp, blinker, gear, speed, prox, errorNo) = (0, 0, 0, 0, 0, 4)

        self.transform_speedometer_rotate.reset()

        if errorNo == 1: # not connected
            self.label_gear.setText("E1")
            self.label_speedometer_digits.setText("NO SIGNAL")
```

Integration

통합: 추상화 과정에서 구상한 내용과 표준화 과정에서
정의한 표준을 바탕으로 task flow를 단일화

Integration (통합)

- Software Module은 동작 유형에 따라 알맞은 곳에서 호출됨
- 연속 실행: main 함수의 for(;;) 루프 내부

```
for (;;)
{
    // Conducting continuous SWMs
    SWM_Lamp();
    SWM_Throttle();
}
```

- 100ms 인터럽트: LPIT CH0 ISR 내부

```
void LPIT0_Ch0_IRQHandler(void)
{
    lpit0_ch0_flag_counter++;

    // Conducting 100ms synced SWMs
    SWM_Blinker();
    SWM_Alarm();
    SWM_UART();
    SWM_ProxWarning();
    LPIT0->MSR |= LPIT_MSR_TIF0_MASK; /* Clear LPIT0 timer flag 0 */
}
```

Integration (통합)

- 10μs 인터럽트: LPIT CH1 ISR 내부

```
void LPIT0_Ch1_IRQHandler(void)
{
    lpit0_ch1_flag_counter++;

    // Conducting 10us synced SWMs
    SWM_Sensor();
    LPIT0->MSR |= LPIT_MSR_TIF1_MASK;
}
```

- 버튼 인터럽트: GPIO PORTC ISR 내부

```
void PORTC_IRQHandler(void)
{
    // Conducting GPIO input interrupt connected SWMs
    SWM_Panel();

    PORTC->PCR[8] |= PORT_PCR_ISF_MASK;
    PORTC->PCR[9] |= PORT_PCR_ISF_MASK;
    PORTC->PCR[10] |= PORT_PCR_ISF_MASK;
    PORTC->PCR[11] |= PORT_PCR_ISF_MASK;
}
```

Integration (통합)

- 제어 신호 및 데이터는
  공통의 전역 변수에 읽고 쓰는 것으로
  구현함

```
int uwave_counter = 0;
int uwave_distance = 0;
int uwave_high = 0;
int buzzer_counter = 0;
int proximity_warning = 0;
int proximity_counter = 0;

_GEAR_TYPE gear = P;
_LAMP_TYPE lamp_mode = OFF; // which lamp mode is set
_LAMP_TYPE lamp = OFF;      // which lamps to turn on
int lamp_auto_trig_low, lamp_auto_trig_high;
_BLINKER_TYPE blinker_mode; // Is the blinker set or not
_BLINKER_TYPE blinker;       // Are the blinker lamps on or off
int blinker_counter;
_UART_DATA_TYPE d_send;

uint32_t vrResult, cdsResult;
```

논의사항 및 개선점

논의사항 및 개선점

1. 공유 메모리 영역에 대한 의존성 해결
   - 소프트웨어 모듈의 모든 데이터를 데이터 영역에 저장하고 공유하는 것은 편리하지만, 상황에 따라 위험할 수 있음
   - 한 소프트웨어 모듈의 제어 흐름이 다른 소프트웨어 모듈의 공유 메모리 접근에 의존한다면 소프트웨어의 구조가 복잡해짐

   - Runtime Layer를 새롭게 정의하여 현재 사용되지 않고 있는 개별 모듈의 스택 영역을 활용하는 방법이 있음.
   - 또는, 다른 소프트웨어 모듈에서는 보이지 않는, 소프트웨어 모듈 간 상호 작용을 위한 계층을 새로 정의할 수 있음

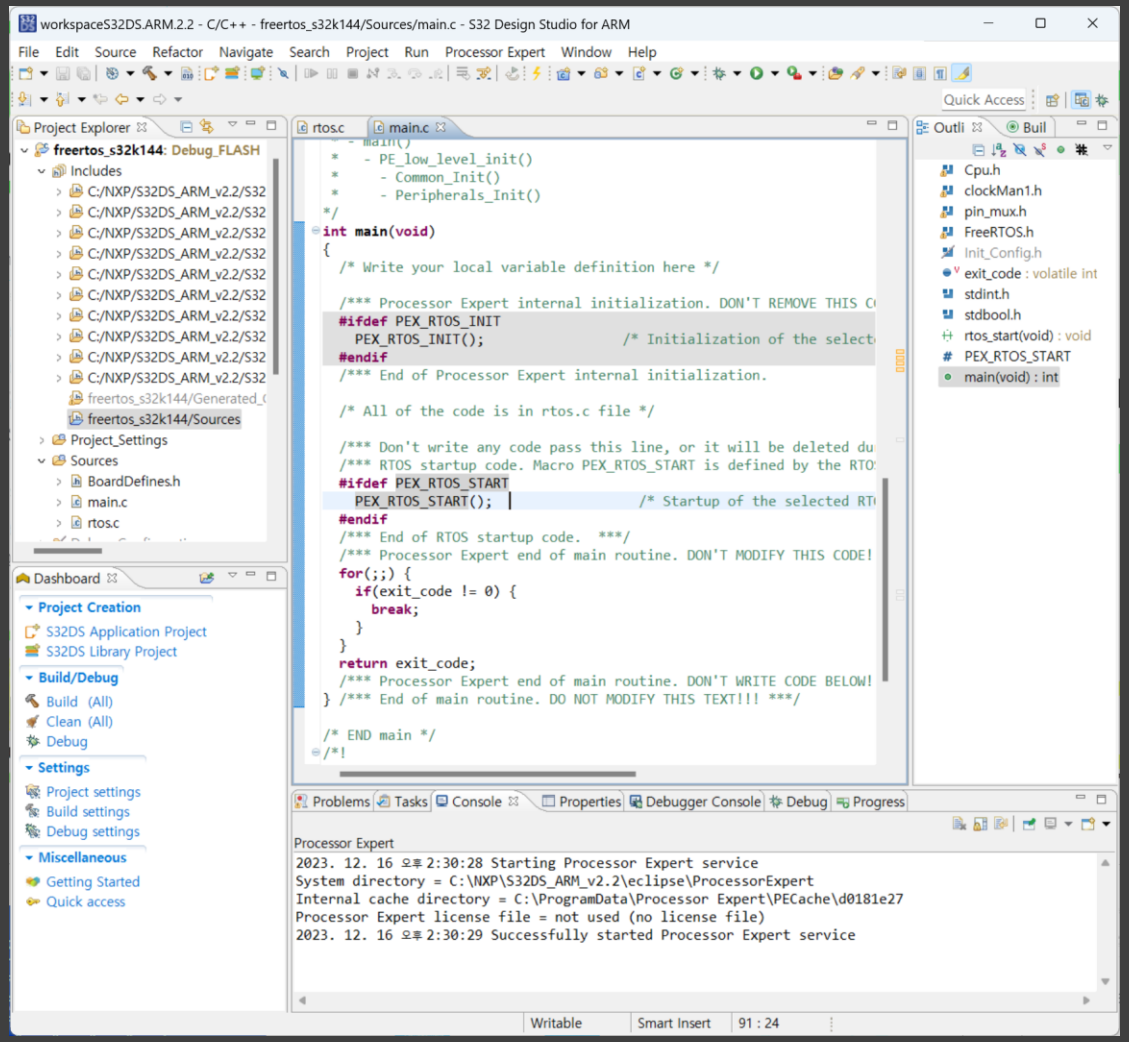
논의사항 및 개선점

2. 동적 멀티태스킹이 불가능함
   - 대부분의 RTOS에서는 Event 등의 개념과 Task의 Priority를 정의하여 알맞게 Scheduling이 발생하도록 설정되어 있음
   - Scheduling이 불가능하다면, Task 내에서 Preemption이 일어나야 하는 모든 상황에 대해 예외 처리 코드를 작성해 주어야 함

   - Preemption이 필요한 경우 수동으로 인터럽트를 invoke하고, ISR 내에서 PC와 LR의 주소를 변경하는 등의 방법으로 실현 가능할 것으로 보임

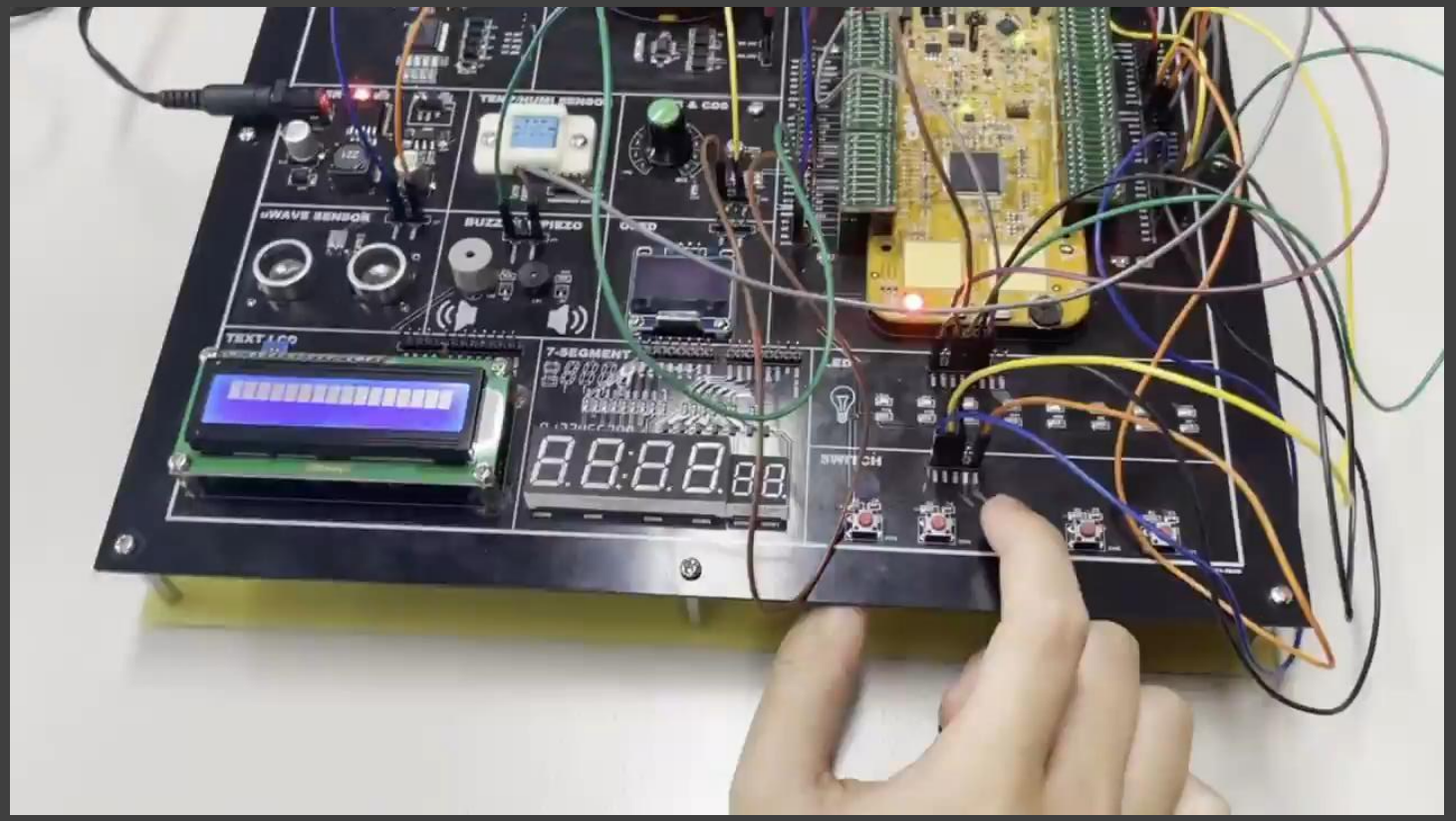
논의사항 및 개선점

- S32K144 보드에
  FreeRTOS를 비롯한 실시간 운영체제
  탑재 가능

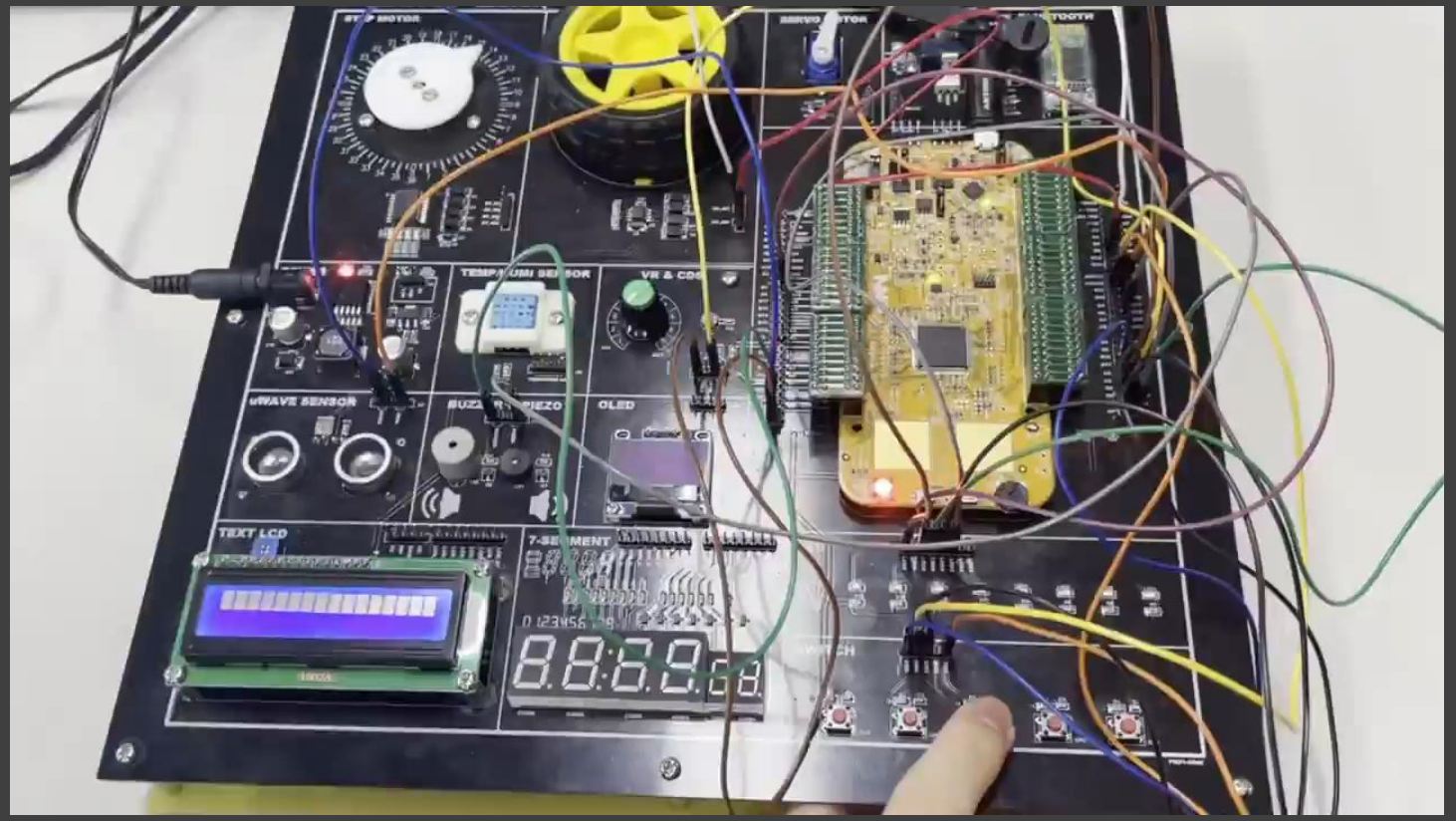


프로젝트 시연

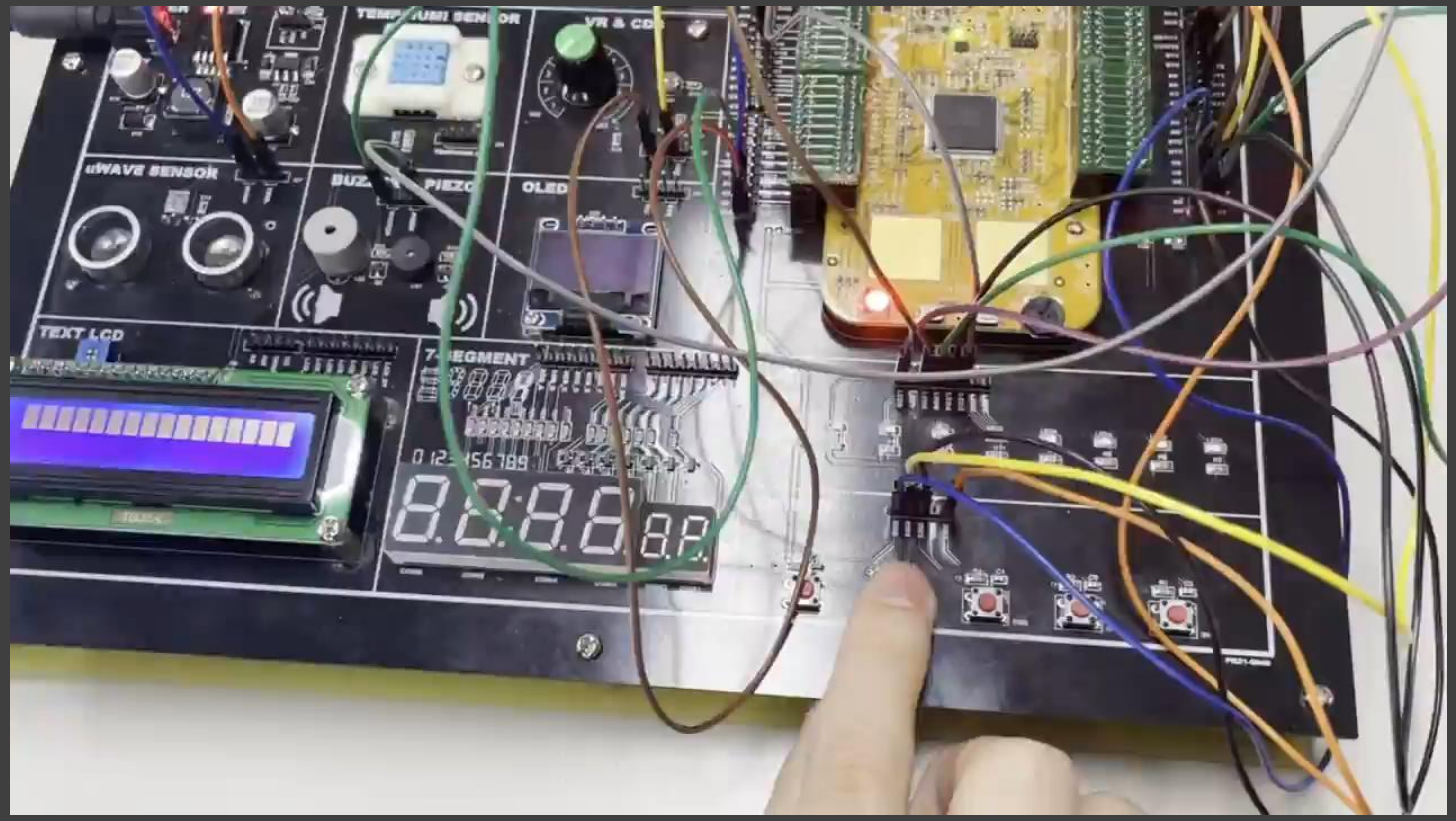
1. 기어 모드 변경



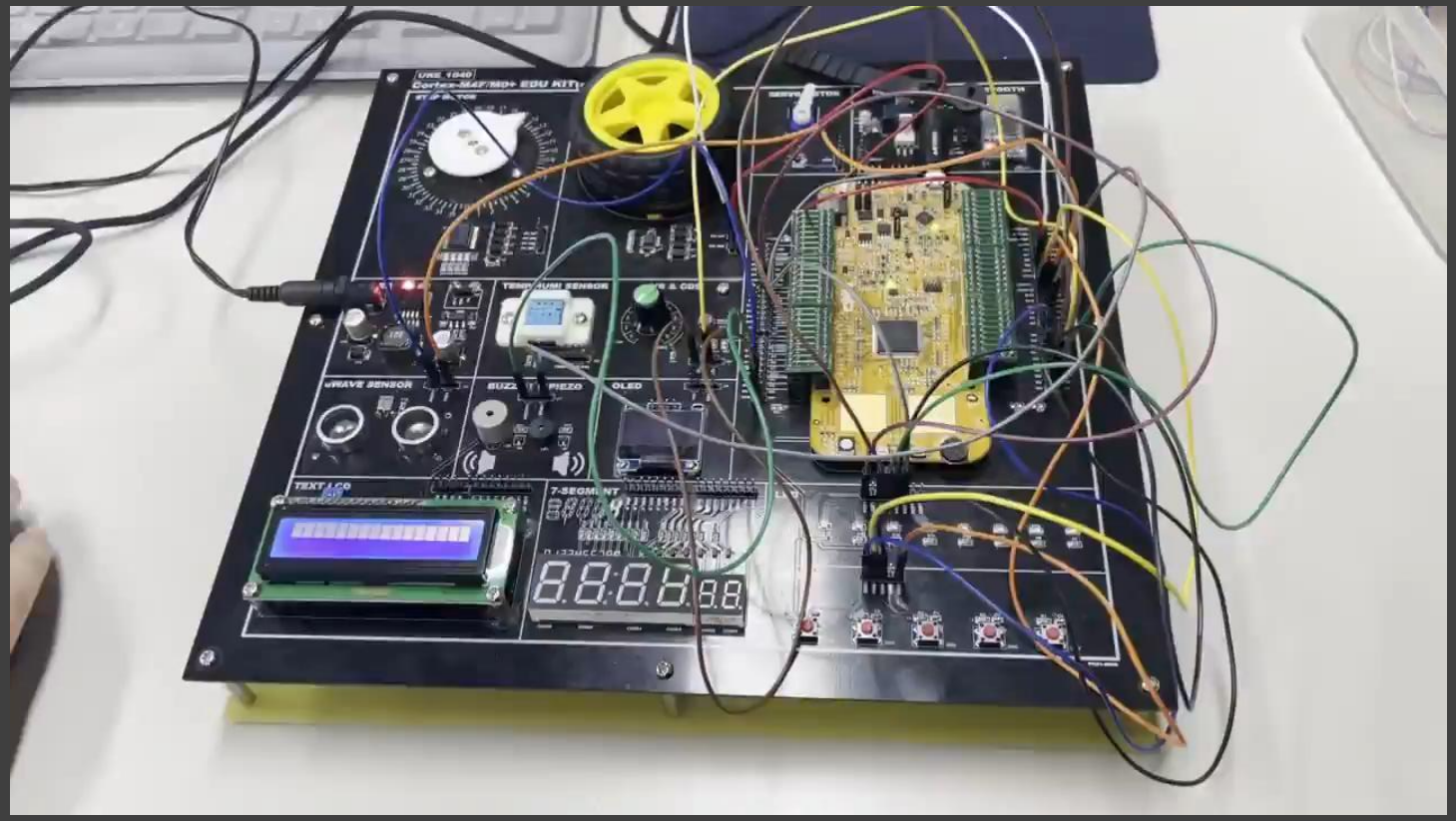
2. 전진 및 후진



3. 전조등 동작 확인



4. 전방 근접 경고



Q & A

Project Github Repository: https://github.com/lsin07/mcu_termproject