Real-Time Operating System (Day 3 Lab)

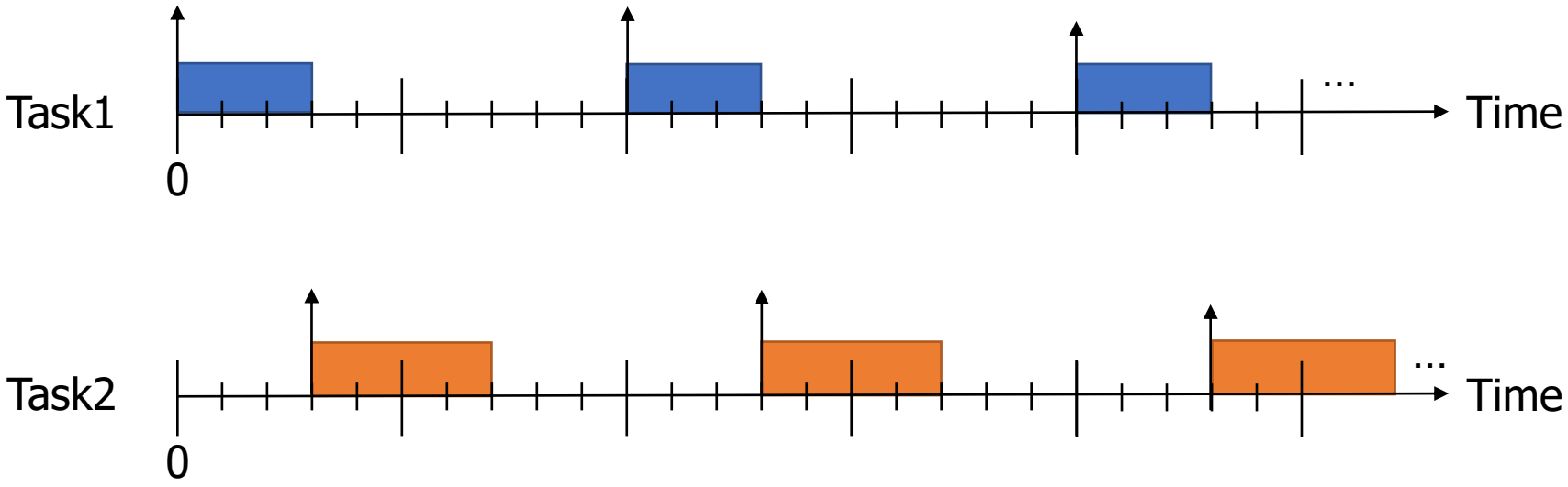
Jong-Chan Kim

Graduate School of Automotive Engineering

KMU 국민대학교 KOOKMIN UNIVERSITY

(Review) 06-3. Periodic Tasks

- [예제] 아래 조건의 Task들을 구현해보기
  - Task1: 높은 우선순위, 실행 시간 3초
  - Task2: 낮은 우선순위, 실행 시간 4초
  - AUTOSTART = False로 수정하고 동일하게 구현
  - 타이머 인터럽트를 활용하여 10초마다 주기적으로 실행



(Review) 06-3. Periodic Tasks

```
ISR2(TimerISR)
{
    static long c = -4;
    osEE_tc_stm_set_sr0_next_match(1000000U);
    if (c == 0)
        ActivateTask(Task1);
    printfSerial("\n%4ld: ", c++);
}

TASK(Task1)
{
    printfSerial("Task1 Begins...");
    mdelay(3000);
    ActivateTask(Task2);
    printfSerial("Task1 Finishes...");
    TerminateTask();
}
```

```
ISR2(TimerISR)
{
    static long c =
    osEE_tc_stm_set_sr0_next_match(1000000U);
    if (c % 10 == 0)
        ActivateTask(Task1);
    printfSerial("\n%4ld: ", c++);
}

TASK(Task1)
{
    printfSerial("Task1 Begins...");
    mdelay(3000);
    printfSerial("Task1 Finishes...");
    ChainTask(Thask2);
}
```

10초마다 주기적으로
Task1 실행

이후 동작 없으니,
ChainTask를 사용해도됨

(Review) 09. ButtonISR

- ButtonISR 추가 (Button 입력에 반응하여 Task Activate)

```
ISR2(ButtonISR)
{
    unsigned int a0;
    DisableAllInterrupts();
    osEE_tc_delay(5000);
    a0 = readADCValue(3);
    if (a0 < 500) {          /* TOP */
        printfSerial("<BUTTON:T>");
        ActivateTask(Task1);
    } else if (a0 < 1200) { /* DOWN */
        printfSerial("<BUTTON:D>");
        ActivateTask(Task2);
    } else if (a0 < 1600) { /* LEFT */
        printfSerial("<BUTTON:L>");
    } else if (a0 < 2200) { /* RIGHT */
        printfSerial("<BUTTON:R>");
    } else {
        printfSerial("<BUTTON:?>");
    }
    osEE_tc_delay(3000);
    EnableAllInterrupts();
}
```

```
ISR ButtonISR {
    CATEGORY = 2;
    SOURCE = "SCUERU0";
    PRIORITY = 10;
};
```

(Review) 09. ButtonISR

```
..............
...OS Starts...
..............

 -4:
 -3:
 -2: <BUTTON:T>Task1 Begins...
 -1:
  0:
  1: Task1 Finishes...
  2:
  3:
  4: <BUTTON:D>Task2 Begins...
  5:
  6:
  7: Task2 Finishes...
  8:
  9:
 10:
 11: <BUTTON:L><BUTTON:L>
 12:
 13:
 14: <BUTTON:R>
 15:
 16:
```

Top 버튼 신호에 따라
Task1 실행

Down 버튼 신호에 따라
Task2 실행

(Review) 09. ButtonISR

- ButtonISR에서 30초 mdelay 실행하면?

```
..............
...OS Starts...
..............

  -4:
  -3:
  -2:
  -1:
   0: Task1 Begins...
   1:
   2:
   3: Task1 Finishes...
   4:
   5:
   6: <BUTTON:?>
   7: <BUTTON:?>
   8:
   9:
  10:
```

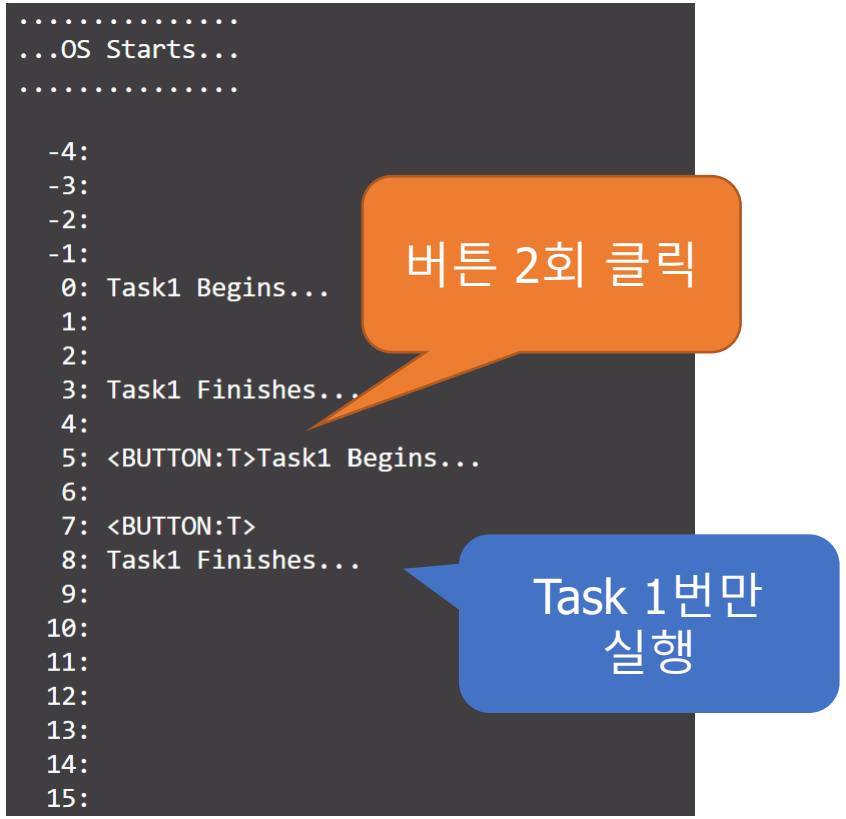
mdelay(30000);

ISR2는 OS와 상호작용, ISR 안에서
오래 걸리는 동작을 하면 Task도 멈추고 버튼도 멈춘다.
시스템 전체가 정지 상태에 빠진다.

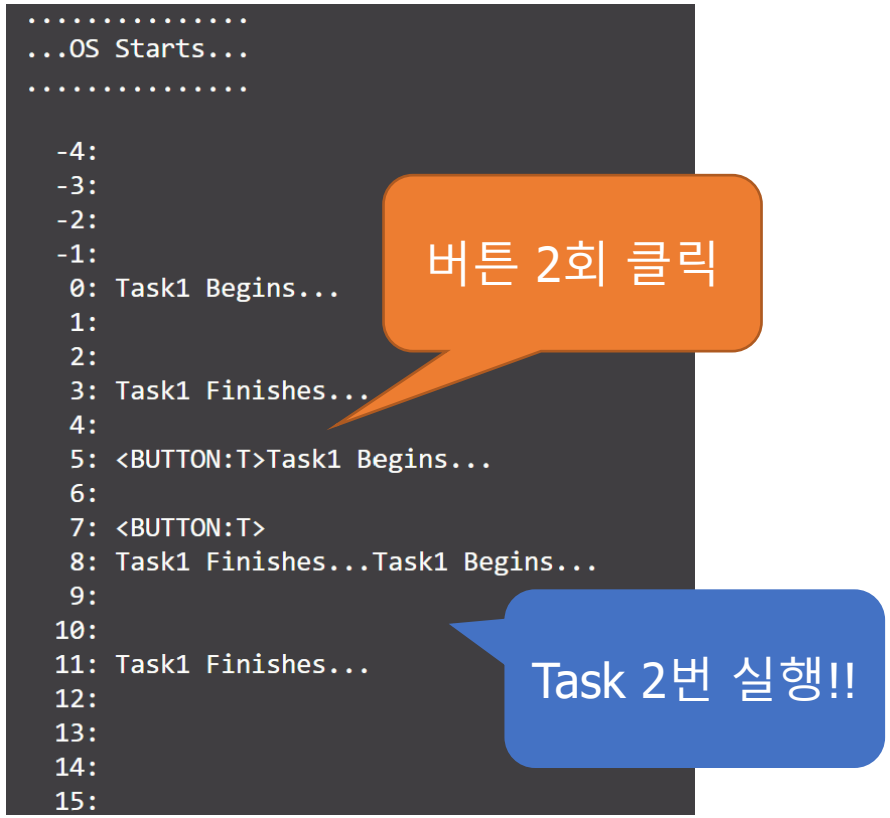
(Review) 09. ButtonISR

- 중복 Activation 하려면?
  - ACTIVATION = 1 → 2로 수정 필요

- ACTIVATION = 1



- ACTIVATION = 2



(Review) 10-1. Alarm

- OIL에 COUNTER와 ALARM 추가

```
COUNTER mycounter {
    MINCYCLE = 1;
    MAXALLOWEDVALUE = 127;
    TICKSPERBASE = 1;
};

ALARM alarm1 {
    COUNTER = mycounter;
    ACTION = ACTIVATETASK {
        TASK = Task1;
    };
    AUTOSTART = TRUE {
        ALARMTIME = 5;
        CYCLETIME = 10;
    };
};
```

```
ALARM alarm2 {
    COUNTER = mycounter;
    ACTION = ACTIVATETASK {
        TASK = Task2;
    };
    AUTOSTART = TRUE {
        ALARMTIME = 5;
        CYCLETIME = 20;
    };
};
```

(Review) 10-1. Alarm

- TimerISR에서
  - ActivateTask(Task1) 삭제
  - mycounter 증가

```
ISR2(TimerISR)
{
    static long c = -4;
    osEE_tc_stm_set_sr0_next_match(1000000U);
    if (c == 0)
        ActivateTask(Task1);
    IncrementCounter(mycounter);
    printfSerial("\n%4ld: ", c++);
}
```

(Review) 10-1. Alarm

- Alarm1: Task1 실행
  - OS 시작 후 5ms 뒤 첫 실행, 이후 10ms 주기
- Alarm2: Task2 실행
  - OS 시작 후 5ms 뒤 첫 실행, 이후 20ms 주기

우선순위에 따라,
Task2 먼저 실행 후
Task1 실행

```
..............
...OS Starts...
..............

 -4:
 -3:
 -2:
 -1:
  0: Task2 Begins...
  1:
  2:
  3: Task2 Finishes...Task1 Begins...
  4:
  5:
  6: Task1 Finishes...
  7:
  8:
  9:
 10: Task1 Begins...
 11:
 12:
 13: Task1 Finishes...
 14:
 15:
 16:
 17:
 18:
 19:
 20: Task2 Begins...
 21:
 22:
 23: Task2 Finishes...Task1 Begins...
 24:
 25:
 26: Task1 Finishes...
 27:
 28:
 29:
```

(Review) 11. Alarm Callback

- 콜백 함수 등록

```
ALARMCALLBACK(MyCallback)
{
    printfSerial("<MyCallback>");
}
```

```
ALARM alarm3 {
    COUNTER = mycounter;
    ACTION = ALARMCALLBACK {
        ALARMCALLBACKNAME = "MyCallback";
    };
    AUTOSTART = TRUE {
        ALARMTIME = 5;
        CYCLETIME = 15;
    };
};
```

(Review) 12. Event

```
ISR2(ButtonISR)
{
    unsigned int a0;
    DisableAllInterrupts();
    osEE_tc_delay(5000);
    a0 = readADCValue(3);
    if (a0 < 500) {
        printfSerial("<BUTTON:T>");
        SetEvent(Task2, Event1);
    } else if (a0 < 1200) {
        printfSerial("<BUTTON:D>");
        SetEvent(Task2, Event2);
    } else if (a0 < 1600) {
        printfSerial("<BUTTON:L>");
    } else if (a0 < 2200) {
        printfSerial("<BUTTON:R>");
    } else {
        printfSerial("<BUTTON:?>");
    }
    osEE_tc_delay(3000);
    EnableAllInterrupts();
}
```

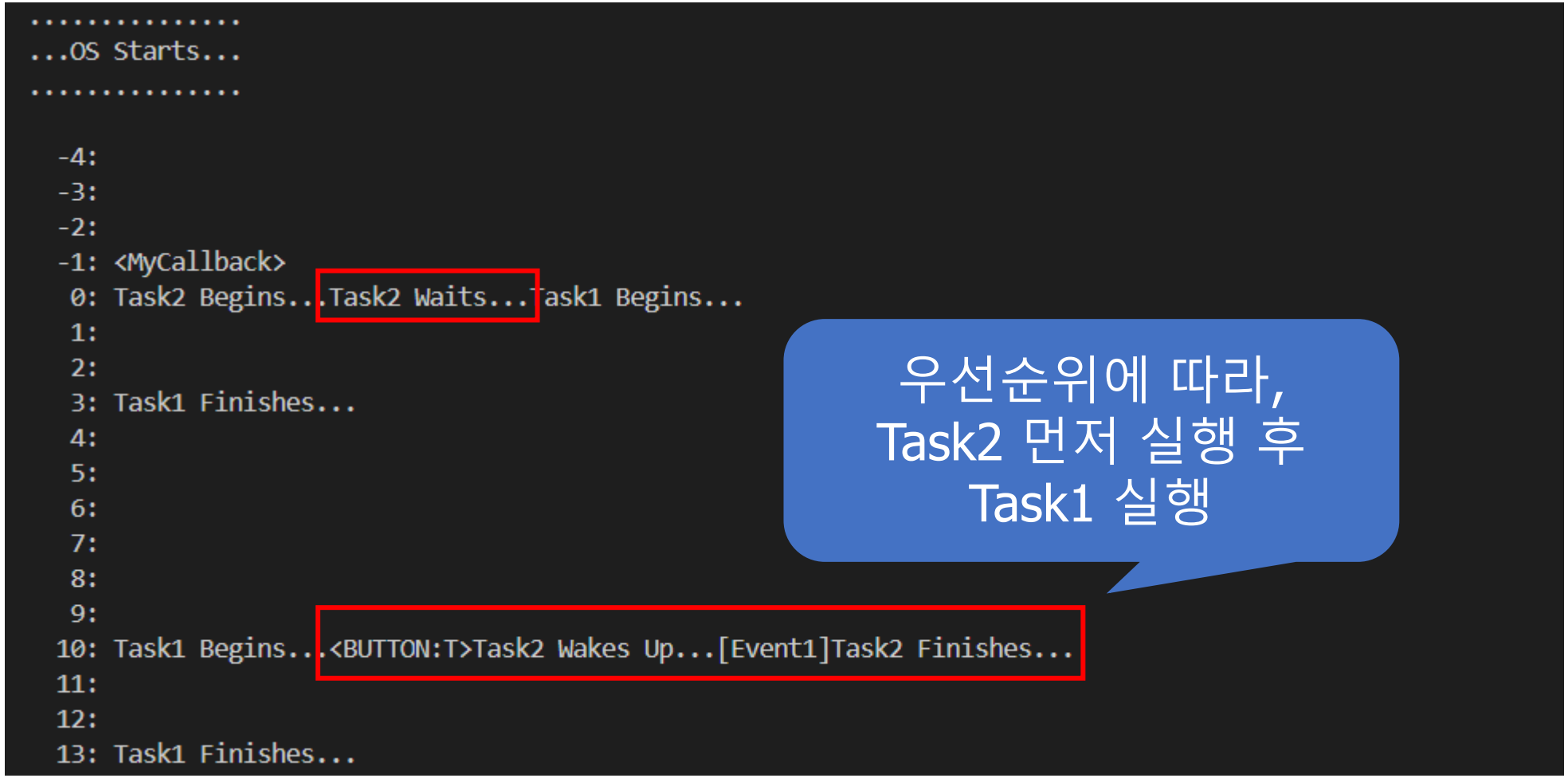
```
CPU_DATA = TRICORE {
    ID = 0x0;
    CPU_CLOCK = 200.0
    MULTI_STACK = TRUE;
};
…
EVENT Event1 { MASK = AUTO; };
EVENT Event2 { MASK = AUTO; };
…
TASK Task2 {
    PRIORITY = 2;
    STACK = PRIVATE {
        SIZE = 1024;
    };
    SCHEDULE = FULL;
…
    EVENT = Event1;
    EVENT = Event2;
…
}
```

(Review) 12. Event

```
TASK(Task2)
{
    EventMaskType mask;
    printfSerial("Task2 Begins...");
    printfSerial("Task2 Waits...");
    WaitEvent(Event1 | Event2);
    printfSerial("Task2 Wakes Up...");
    GetEvent(Task2, &mask);
    if (mask & Event1) {
        printfSerial("[Event1]");
        ClearEvent(Event1);
    }
    if (mask & Event2) {
        printfSerial("[Event2]");
        ClearEvent(Event2);
    }
    printfSerial("Task2 Finishes...");
    TerminateTask();
}
```

(Review) 12. Event

- OS 시작 시 알람 콜백과 Task2가 먼저 실행 → Task2는 이벤트 대기 상태로 진입
- 버튼 입력으로 Event1이 발생하면 Task2가 깨어나 이벤트를 처리하고 종료됨



(Review) 12. Event

- 우선순위 반대의 경우 스케줄링

```
..............
...OS Starts...
..............

  -4:
  -3:
  -2:
  -1: <MyCallback>
   0: Task1 Begins...
   1:
   2:
   3: Task1 Finishes...Task2 Begins...Task2 Waits...
   4:
   5:
   6:
   7:
   8:
   9:
  10: Task1 Begins...<BUTTON:T>
  11:
  12:
  13: Task1 Finishes...Task2 Wakes Up...[Event1]Task2 Finishes...
  14: <MyCallback>
  15:
  16:
```

우선순위에 따라,
Task1 먼저 실행 후
Task2 실행

(Review) 13. Alarm SetEvent

- Alarm을 이용한 주기적인 SetEvent Action

```
ALARM alarm3 {
    COUNTER = mycounter;
    ACTION = SETEVENT {
        TASK = Task2;
        EVENT = Event1;
    };
    AUTOSTART = TRUE {
        ALARMTIME = 7;
        CYCLETIME = 20;
    };
};
```

(Review) 13. Alarm SetEvent

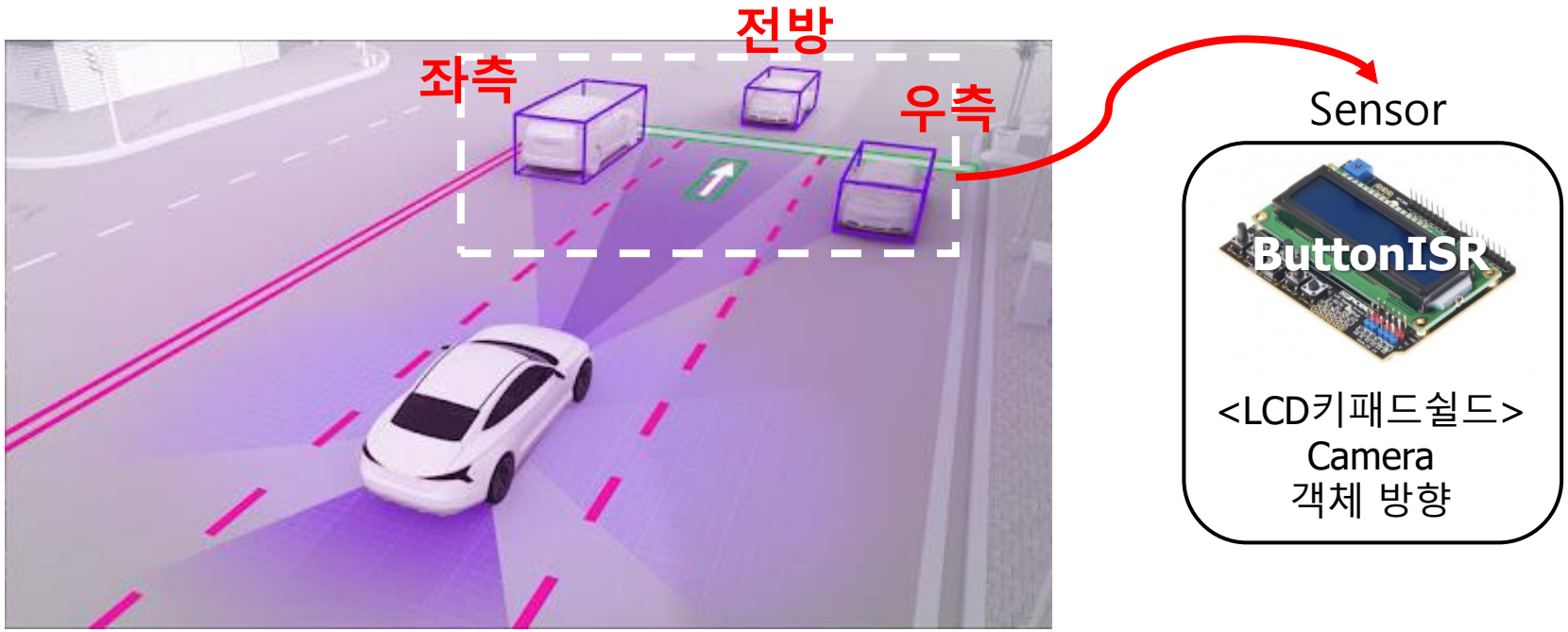
- Alarm을 이용한 주기적인 SetEvent Action

```
.............
...OS Starts...
.............

 -4:
 -3:
 -2:
 -1:
  0: Task2 Begins...Task2 Waits...Task1 Begins...
  1:
  2: Task2 Wakes Up...[Event1]Task2 Finishes...
  3: Task1 Finishes...
  4:
  5:
  6:
  7:
  8:
  9:
 10: Task1 Begins...
 11:
 12:
 13: Task1 Finishes...
 14:
 15:
 16:
 17:
 18:
 19:
 20: Task2 Begins...Task2 Waits...Task1 Begins...
 21:
 22: Task2 Wakes Up...[Event1]Task2 Finishes...
 23: Task1 Finishes...
 24:
```

Alarm에 의해 주기적으로 Event가 발생하여 Task2 깨어남

(Review) Team Project

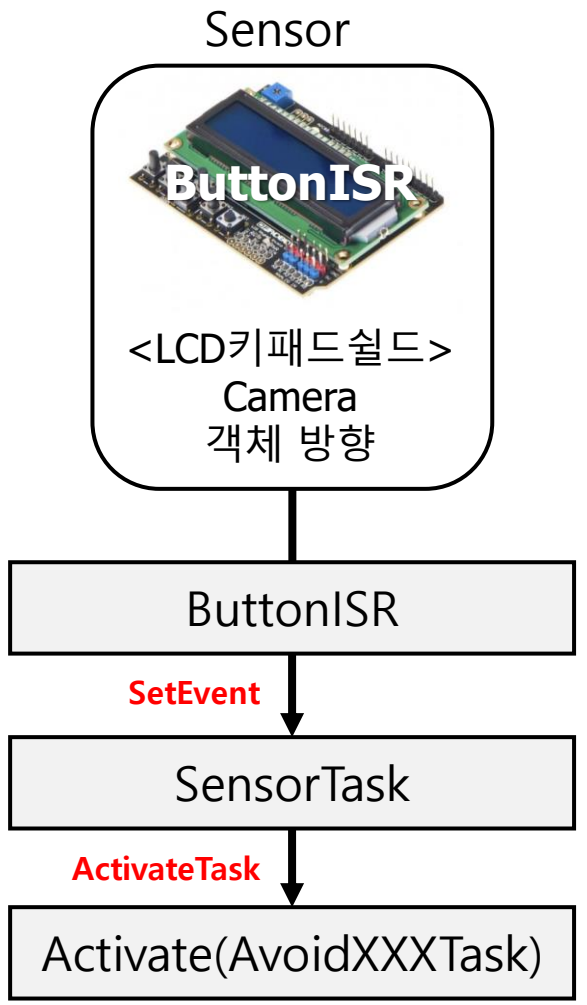
- RTOS 기반 자율주행 장애물 감지 및 회피 시스템
  - ButtonISR로 전방 객체의 위치를 파악한 후 적절한 Task를 실행
  - ✓ RTOS에서 ISR, Task, Event 구조 이해
  - ✓ 자율주행 시스템처럼 Event 기반 판단 및 회피 전략 구성



(Review) Team Project

- 시스템 시나리오
  1. 자율주행 차량이 주행 중
  2. 전방 / 좌측 / 우측에 장애물이 감지
  3. 감지 Event에 따라 적절한 Task 실행

Task 이름	우선순위	기능 설명
SensorTask	3	방향 판단, AvoidTask 결정
AvoidFrontTask	4	전방 회피
AvoidLeftTask	2	좌측 회피
AvoidRightTask	2	우측 회피



17-1. Loss (공유 자원 data loss 확인)

- Task1: Low priority, AUTOSTART
- Task2: High priority, 1 ms 주기 실행
- 공유 전역 변수 (`volatile unsigned long shared`)
  - Task1의 루프에서 `shared++`
  - Task2 주기적으로 반복 실행하며 `shared ++`
- 양 쪽에서 더한 숫자가 모두 유지되는지 확인
- Task와 ISR 사이에서도 같은 문제가 발생하는지 확인

17-1. Loss

bsw.c

```
int main(void)
{
    osEE_tc_stm_set_clockpersec();
    osEE_tc_stm_set_sr0(1000U, 1U);
```

```
#include "bsw.h"
volatile unsigned long shared = 0;
```

```
TASK(Task1)
{
    unsigned long i;
    printfSerial("Task1 Begins...\n");
    for (i = 0; i < 20000000; i++) {
        shared++;
    }
    printfSerial("Added 20000000 to shared\n");
    printfSerial("counter = %lu\n", shared);
    printfSerial("Task1 Finishes...\n");
    TerminateTask();
}
```

```
TASK(Task2)
{
    static unsigned long i = 0;
    if (i < 500) {
        shared++;
    } else if (i == 500) {
        printfSerial("Added 500 to shared\n");
    }
    i++;
    TerminateTask();
}
```

17-1. Loss

- Resource를 이용해서 Integrity Loss 문제 해결 필요

```
..............
...OS Starts...
..............
Task1 Begins...
Added 500 to shared
Added 20000000 to shared
counter = 20000256
Task1 Finishes...
```

20000000 + 500 = 20000256(!!!)

17-2. No Loss

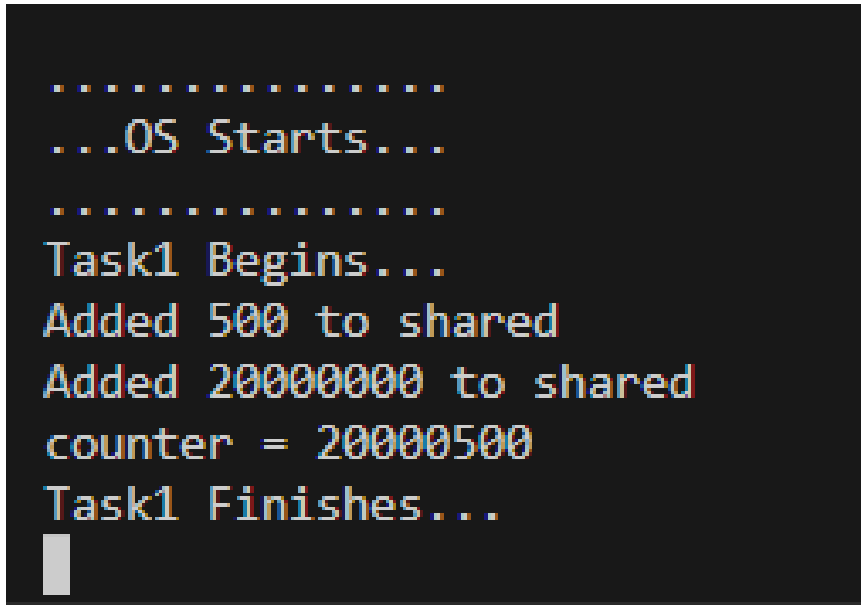
- OSEK의 RESOURCE 기능을 이용하여 Data Loss 문제 해결

```
GetResource(S1);
shared++;
ReleaseResource(S1);
```

```
RESOURCE S1 {
    RESOURCEPROPERTY = STANDARD;
};

…

TASK Task1 {
    PRIORITY = 1;
    STACK = SHARED;
    SCHEDULE = FULL;
    AUTOSTART = TRUE;
    ACTIVATION = 1;
    RESOURCE = S1;
};
```

17-2. No Loss

- Data Integrity 문제 해결

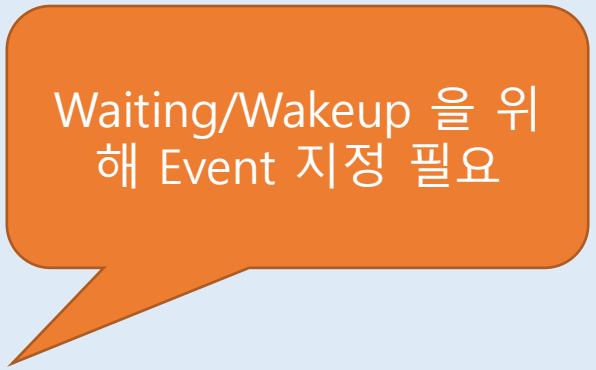


20000000 + 500 = 20000500(!!!)

18. Mutex

- mutex.h

```
#ifndef MUTEX_H_
#define MUTEX_H_

#define LOCKED   1
#define UNLOCKED  0

typedef struct _MutexType {
    int flag;
    EventMaskType event;
    TaskType waiting_task;
} MutexType;

void InitMutex(MutexType *mutex, EventMaskType event);

void GetMutex(MutexType *mutex);

void ReleaseMutex(MutexType *mutex);


#endif /* MUTEX_H_ */
```



PCP 없이 Mutex 사용할 경우 문제점을 확인하기 위한 Dummy 구현

18. Mutex

- mutex.c : 다음 설명을 참고하여 Mutex함수를 구현해보기

```
#include "ee.h"
#include "bsw.h"
#include "mutex.h"

void InitMutex(MutexType *mutex, EventMaskType event)
{
    1
}

void GetMutex(MutexType *mutex)
{
    2
}

void ReleaseMutex(MutexType *mutex)
{
    3
}
```

1. InitMutex
   - Mutex를 초기화
   - 초기상태: flag = UNLOCKED, waiting_task = 0, event = event
2. GetMutex
   - 만약 Mutex가 해제되지 않았다면
     - Block 메세지 출력
     - 현재 실행중인 Task의 ID를 저장 (GetTaskID)
     - 해당 이벤트를 기다린다 (WaitEvent)
   - Mutex를 획득하면 flag를 LOCKED로 설정
3. ReleaseMutex
   - 만약 Mutex가 해제되지 않았다면
     - flag를 UNLOCKED로 설정
     - 만약 Mutex를 기다리는 Task가 있다면 해당 Task에 이벤트를 보냄 (SetEvent)

18. Mutex

- Mutex 선언
- Timer ISR 이용
  - Mutex 초기화
  - Task Activation
- Mutex의 동작 확인

```
MutexType s1;


ISR2(TimerISR)
{
    osEE_tc_stm_set_sr0_next_match(1000000U);
    static long c = -5;
    printfSerial("\n%4ld: ", ++c);
    if(c == -4) {
        InitMutex(&s1, Event1);
    } else if (c == 0) {
        ActivateTask(TaskL);
    } else if (c == 5) {
        ActivateTask(TaskH);
    }
}
```

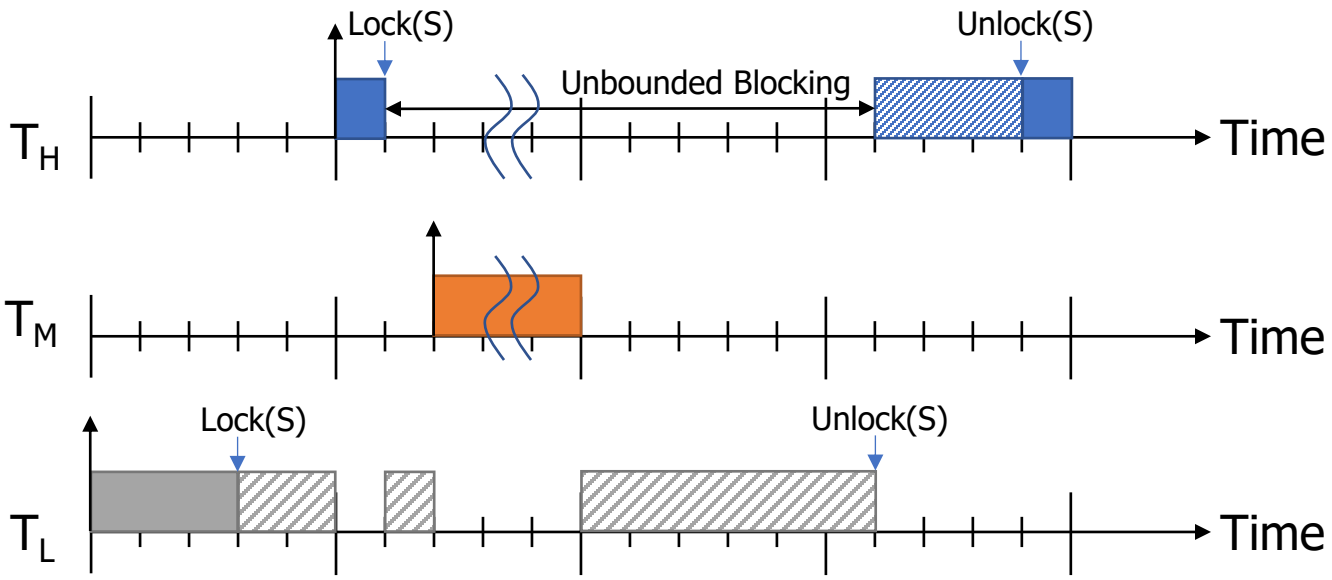
18. Mutex

- Mutex.c 추가

```
 APPDATA tricore_mc {
        APP_SRC="illd/src/IfxAsclin_Asc.c";
        APP_SRC="illd/src/IfxStm.c";
        APP_SRC="illd/src/IfxStm_cfg.c";
        …

        APP_SRC="illd/src/IfxScuEru.c";
        APP_SRC="illd/src/IfxVadc_Adc.c";
        APP_SRC="illd/Libraries/iLLD/TC27D/Tricore/_Impl/IfxVadc_cfg.c";
        APP_SRC="illd/Libraries/iLLD/TC27D/Tricore/Vadc/std/IfxVadc.c";

        APP_SRC="mutex.c";
        APP_SRC="bsw.c";
        APP_SRC="asw.c";
    };
```

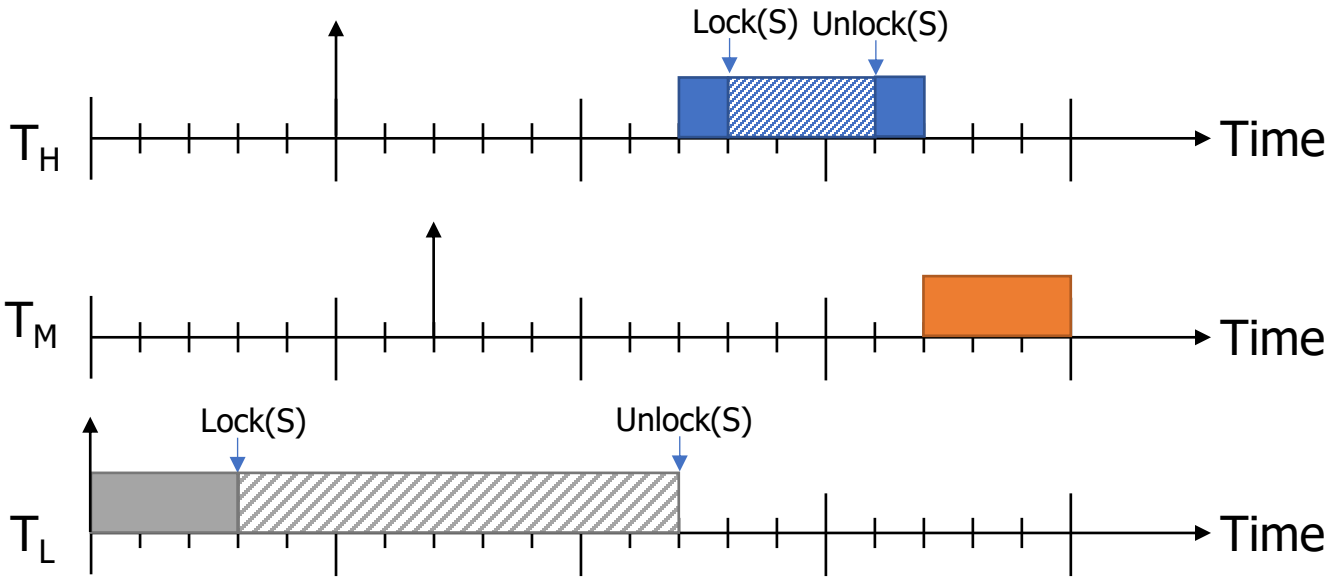
19-1. Priority Inversion

- 아래 스케쥴 재현하기



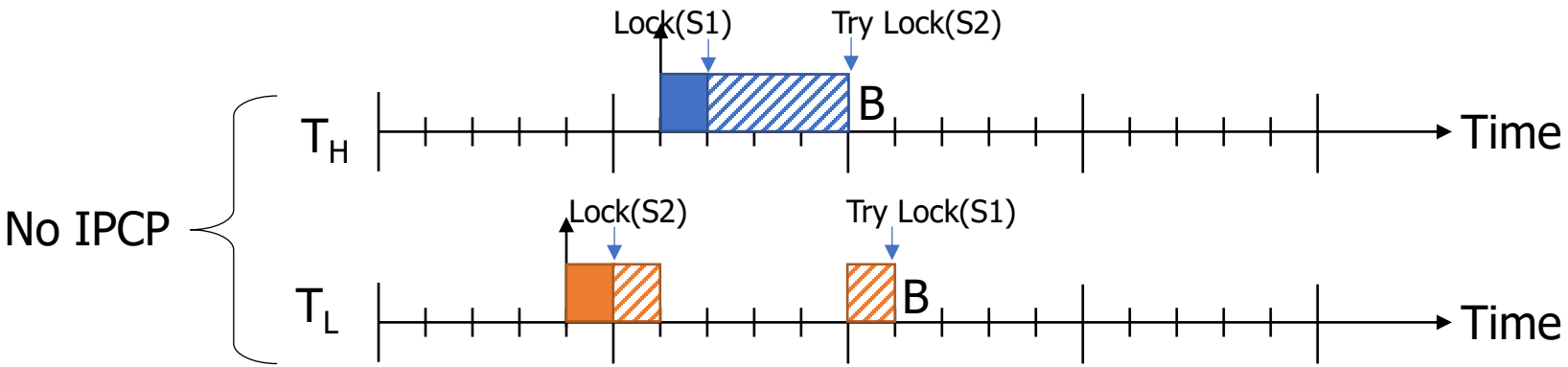
19-2. No Priority Inversion

- 아래 스케쥴 재현하기
- IPCP가 적용된 Resource를 이용하여 스케쥴 변화 확인



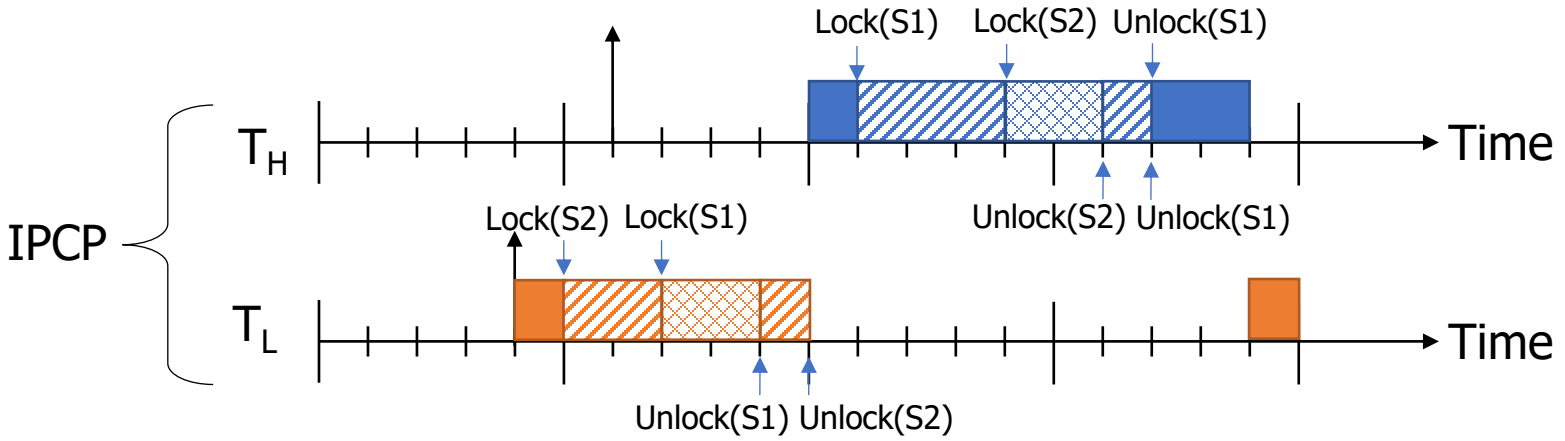
20-1. Deadlock

- Deadlock 재현하기



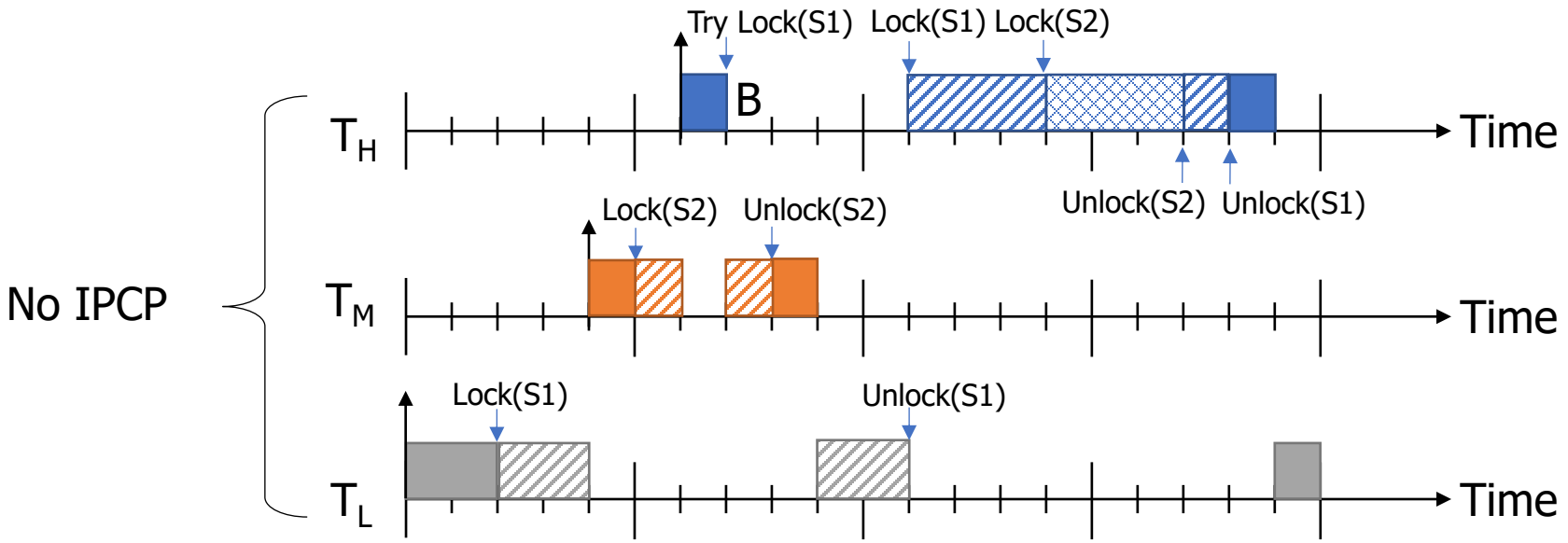
20-2. No Deadlock

- Deadlock 재현하기
- IPCP가 적용된 Resource를 이용하여 Deadlock 해결



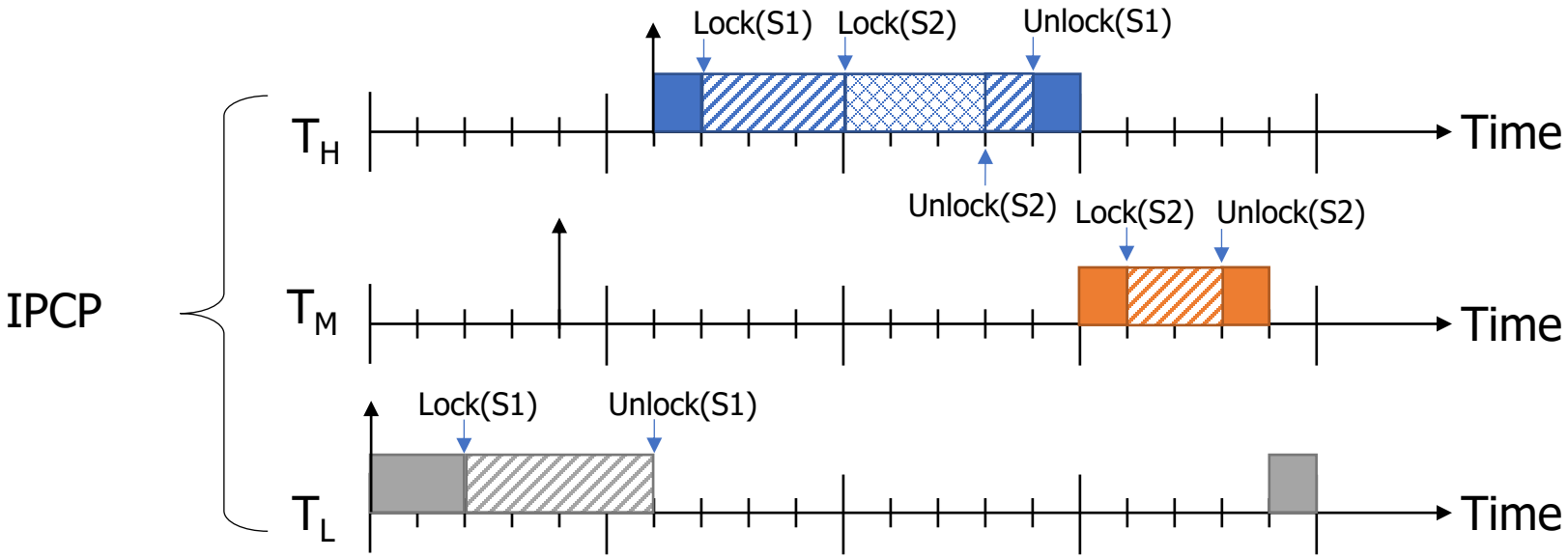
21-1. Before IPCP

- 아래 스케쥴 재현하고 IPCP 동작 확인



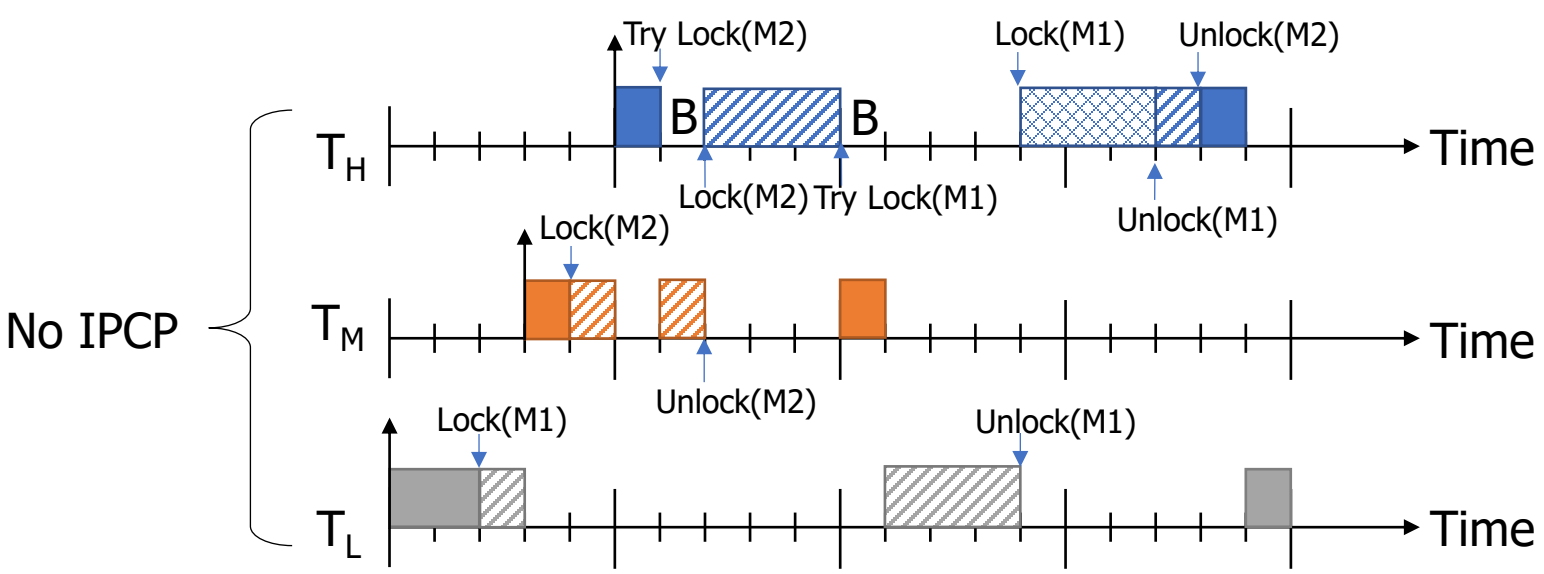
21-2. After IPCP

- 아래 스케쥴 재현하고 IPCP 동작 확인



21-3. Before/After IPCP

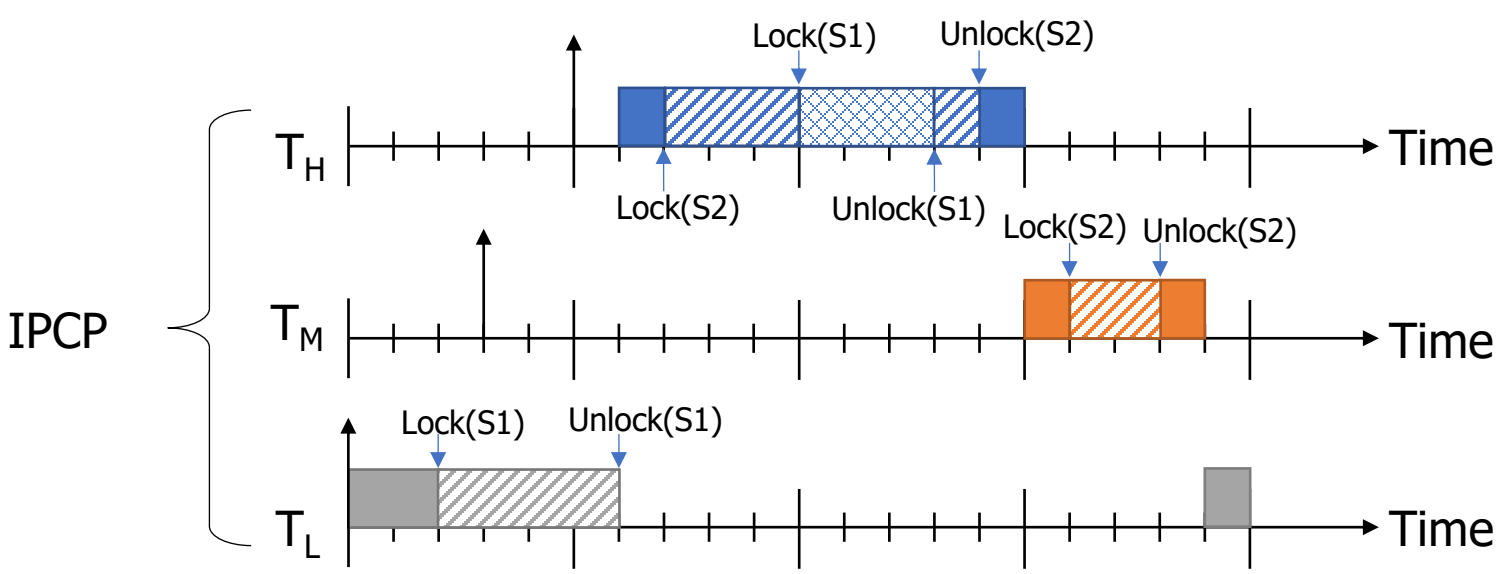
- Mutex로 아래 시스템 구현




```
 0: <TaskL begins.>
 1:
 2: TaskL : Try Lock(S1). TaskL : Get Lock(S1).
 3: <TaskM begins.>
 4: TaskM : Try Lock(S2). TaskM : Get Lock(S2).
 5: <TaskH begins.>
 6: TaskH : Try Lock(S2).   -->  BLock
 7: TaskM : Release Lock(S2). TaskH : Get Lock(S2).
 8:
 9:
10: TaskH : Try Lock(S1).   -->  BLock
11: <TaskM ends.>
12:
13:
14: TaskL : Release Lock(S1). TaskH : Get Lock(S1).
15:
16:
17: TaskH : Release Lock(S1).
18: TaskH : Release Lock(S2).
19: <TaskH ends.>
20: <TaskL ends.>
```

21-3. Before/After IPCP

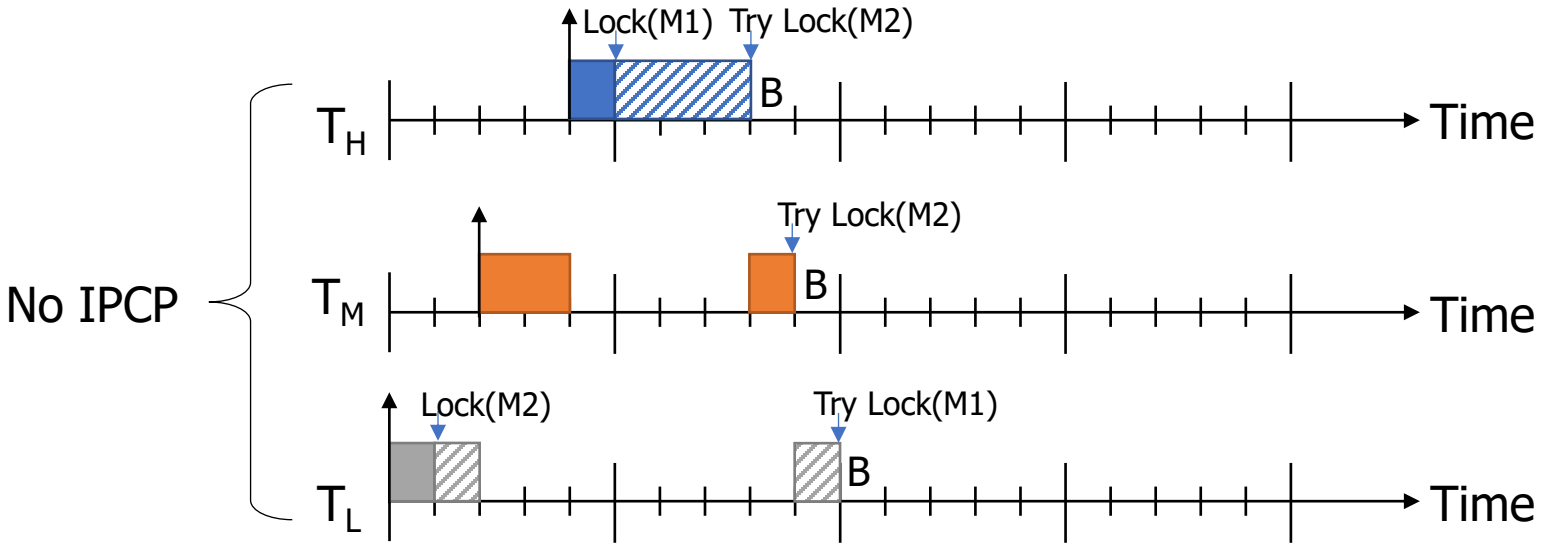
- Mutex를 Resource로 교체, 차이 비교




```
 0: <TaskL begins.>
 1:
 2: TaskL : Try Lock(S1). TaskL : Get Lock(S1).
 3:
 4:
 5:
 6: TaskL : Release Lock(S1). <TaskH begins.>
 7: TaskH : Try Lock(S2). TaskH : Get Lock(S2).
 8:
 9:
10: TaskH : Try Lock(S1). TaskH : Get Lock(S1).
11:
12:
13: TaskH : Release Lock(S1).
14: TaskH : Release Lock(S2).
15: <TaskH ends.> <TaskM begins.>
16: TaskM : Try Lock(S2). TaskM : Get Lock(S2).
17:
18: TaskM : Release Lock(S2).
19: <TaskM ends.>
20: <TaskL ends.>
```

21-4. Before/After IPCP

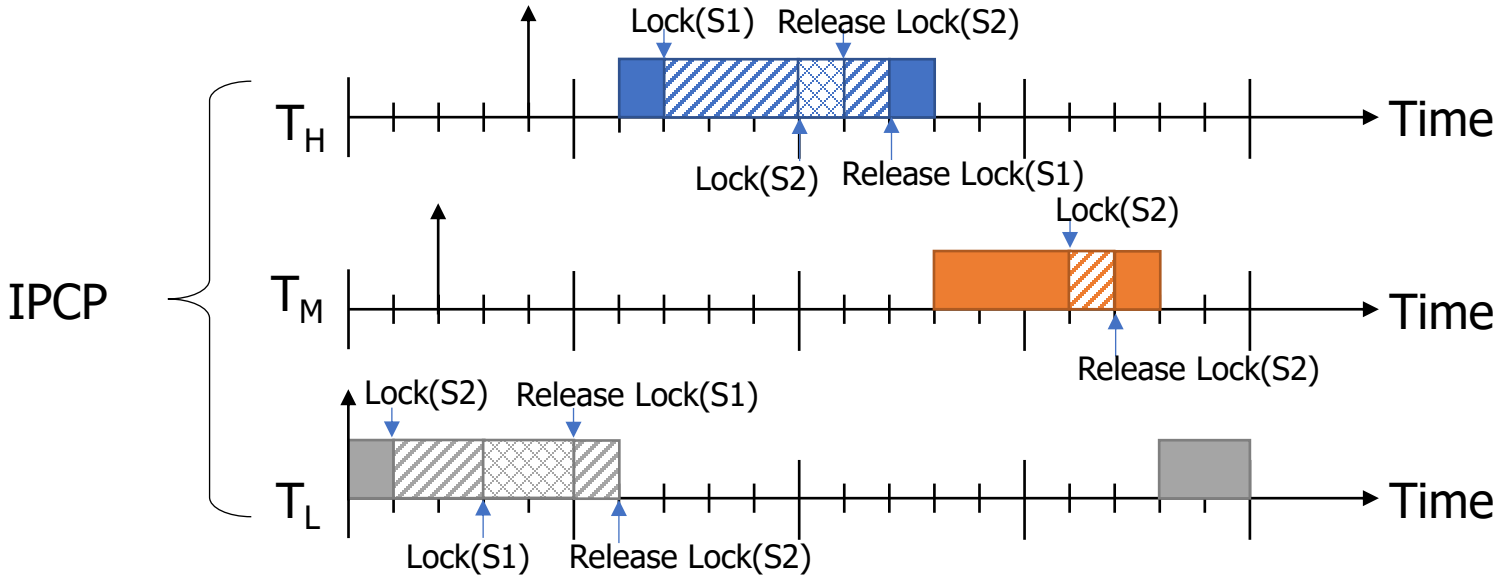
- Mutex로 아래 시스템 구현




```
 0: <TaskL begins.>
 1: TaskL : Try Lock(S2). TaskL : Get Lock(S2).
 2: <TaskM Begins.>
 3:
 4: <TaskH begins.>
 5: TaskH : Try Lock(S1). TaskH : Get Lock(S1).
 6:
 7:
 8: TaskH : Try Lock(S2).    -->  BLock
 9: TaskM : Try Lock(S2),    -->  BLock
10: TaskL : Try Lock(S1).    -->  BLock
11:
12:
13:
14:
15:
16:
17:
18:
19:
20:
```

21-4. Before/After IPCP

- Mutex를 Resource로 교체, 차이 비교

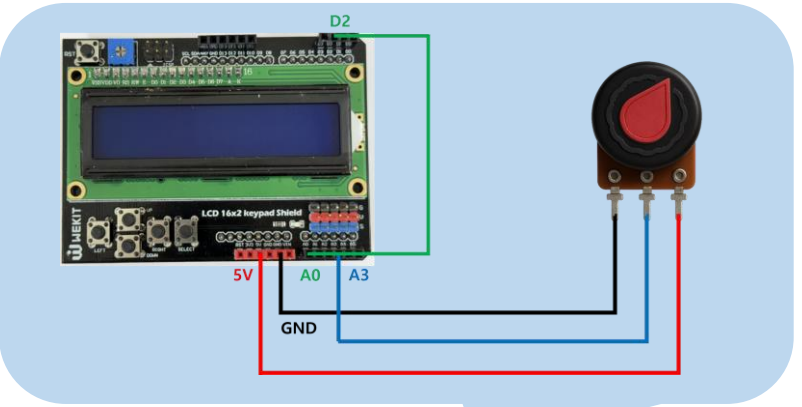


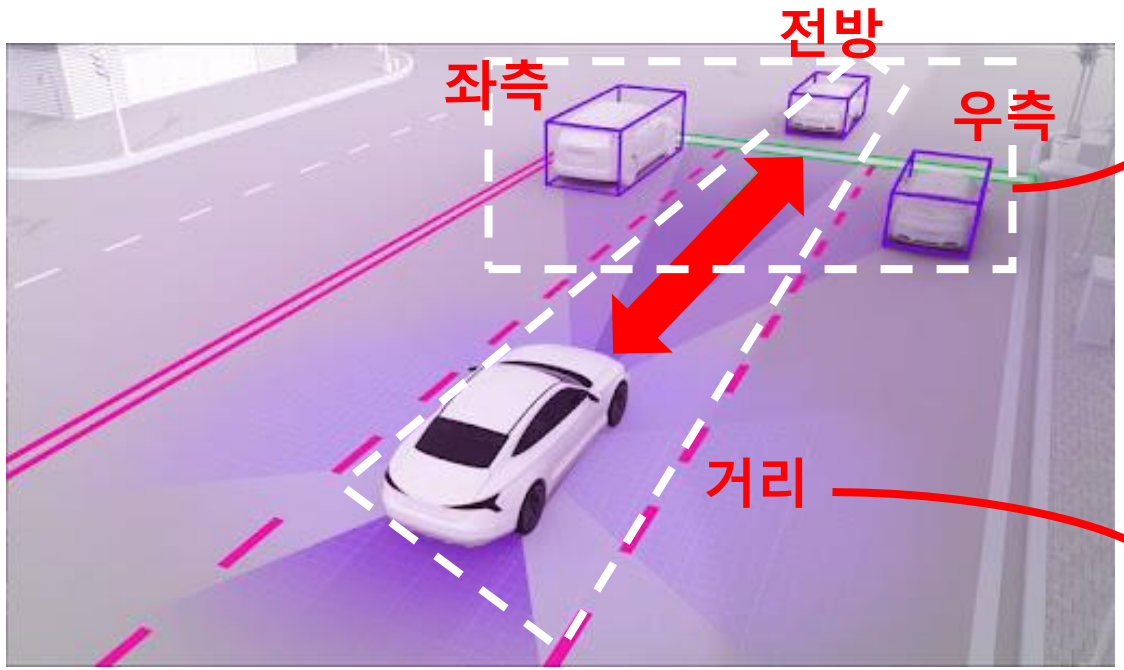

```
 0: <TaskL begins.>
 1: TaskL : Try Lock(S2). TaskL : Get Lock(S2).
 2:
 3: TaskL : Try Lock(S1). TaskL : Get Lock(S1).
 4:
 5: TaskL : Release Lock(S1).
 6: TaskL : Release Lock(S2). <TaskH begins.>
 7: TaskH : Try Lock(S1). TaskH : Get Lock(S1).
 8:
 9:
10: TaskH : Try Lock(S2). TaskH : Get Lock(S2).
11: TaskH : Release Lock(S2).
12: TaskH : Release Lock(S1).
13: <TaskH ends.> <TaskM Begins.>
14:
15:
16: TaskM : Try Lock(S2), TaskM : Get Lock(S2).
17: TaskL : Release Lock(S2).
18: <TaskM ends.>
19:
20: <TaskL ends.>
```

Team Project

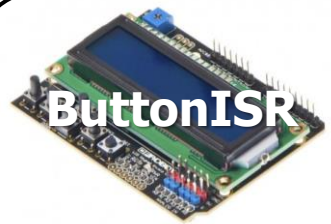
- RTOS 기반 자율주행 장애물 감지 및 회피 시스템
- 목표:
  - ✓ RTOS에서 ISR, Task, Event, Resource 구조 이해
  - ✓ 자율주행 시스템처럼 Event 기반 판단 및 회피 전략 구성
  - ✓ 실제 센서(Button/가변저항)와 연동된 Task 스케줄링 실습

가변저항 연결 (점퍼선 3개)





Sensor 1



<LCD키패드쉴드>
Camera
객체 방향

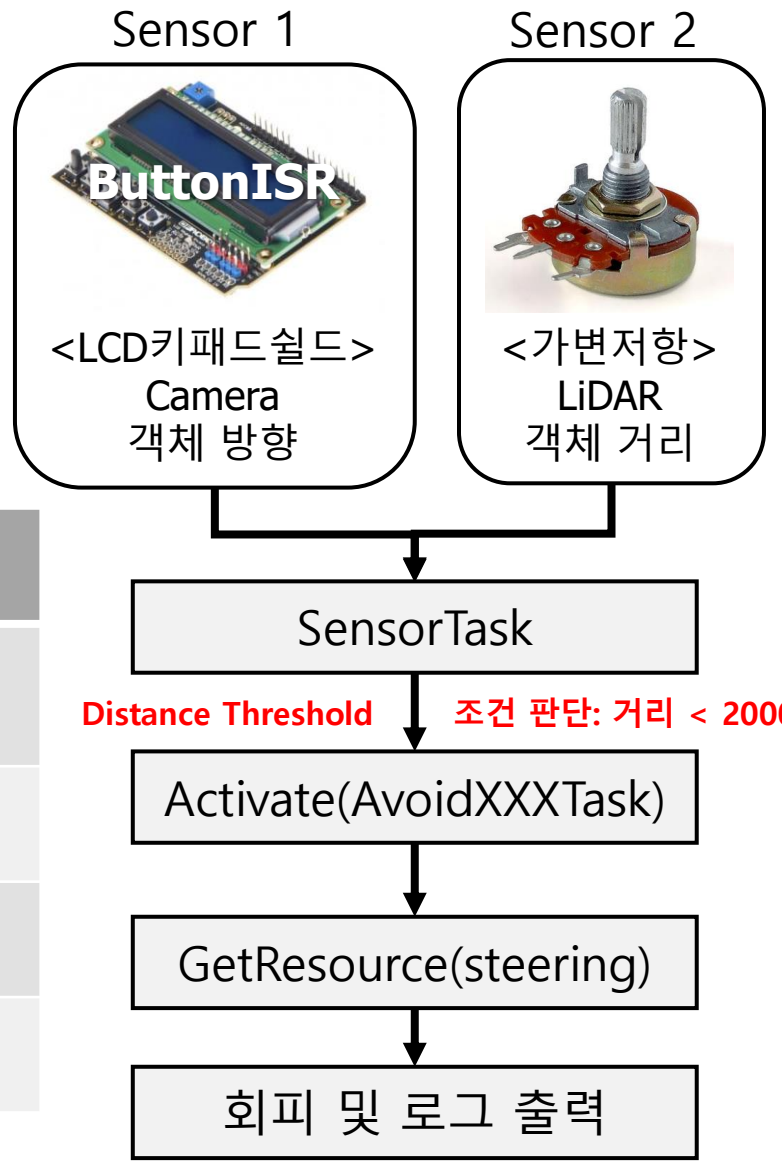
Sensor 2

<가변저항>
LiDAR
객체 거리

Team Project

- 시스템 시나리오
  1. 자율주행 차량이 주행 중
  2. 전방 / 좌측 / 우측에 장애물이 감지되면
     - → 거리가 일정 값 미만이면 회피
     - → 그렇지 않으면 무시
  3. 회피 동작은 반드시 자원을 점유하고 단독 실행

Task 이름	우선순위	기능 설명	공유 자원
SensorTask	3	방향 및 거리 판단, AvoidTask 결정	없음
AvoidFrontTask	4	전방 회피	steering_control
AvoidLeftTask	2	좌측 회피	steering_control
AvoidRightTask	2	우측 회피	steering_control



Questions