임베디드 시스템 설계 및 실험
화요일 8조 텀 프로젝트 보고서

달리는 알람시계

개요 2

기능 2

사용 센서 3

시나리오 4

프로젝트 구현 5

결론 및 토의 9

8조

201602181 정진성

201924660 한병정

201824523 안혜준

202055516 김명서

201824483 박진영

1. 개요

기상알람시계의 목적은 사용자가 알람을 듣고 잠에서 깨어나게 하는 것이다. 하지만 알람을 통해 한 번에 일어나지 못하고 여러 번의 알람을 통해 일어나는 사람들이 있다. 우리는 이에 대한 원인이 침대에서의 손쉬운 알람 끄기라고 보았다. 이를 해결하기 위해 우리는 알람을 끄기 어렵게 하도록 알람시계가 움직이도록 하였다. 이로써 사용자는 알람을 끄기 위해 침대에서 벗어나게 될 것이다. 알람시계는 사람을 감지하는 순간 사람에게서 도망간다. 침대에서 벗어난 사용자는 알람을 끄기 위해 도망가는 알람을 따라잡아야 한다.

알람은 켜져 있으면 짜증나고, 또한, 끄기가 귀찮아야 한다. 사람들은 짜증과 귀찮음 사이에서 고민하다 침대에서 벗어나게 될 것이다.

2. 기능

사람을 감지함과 동시에 달리기 시작하는 알람 시계를 개발한다. 알람 시계는 벽에 부딪히지 않아야 한다. bluetooth를 통해 알람을 울릴 시간을 제어한다.

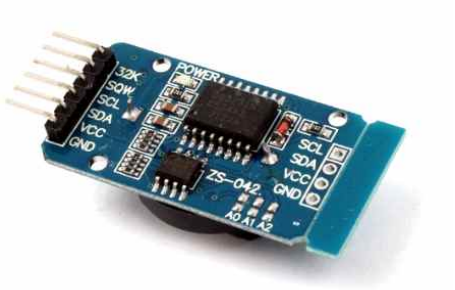
2-1) bluetooth 통신

- 사용자는 블루투스 통신을 통해 기기로 알람이 울리는 시점을 설정할 수 있다.

2-2) 기기의 작동

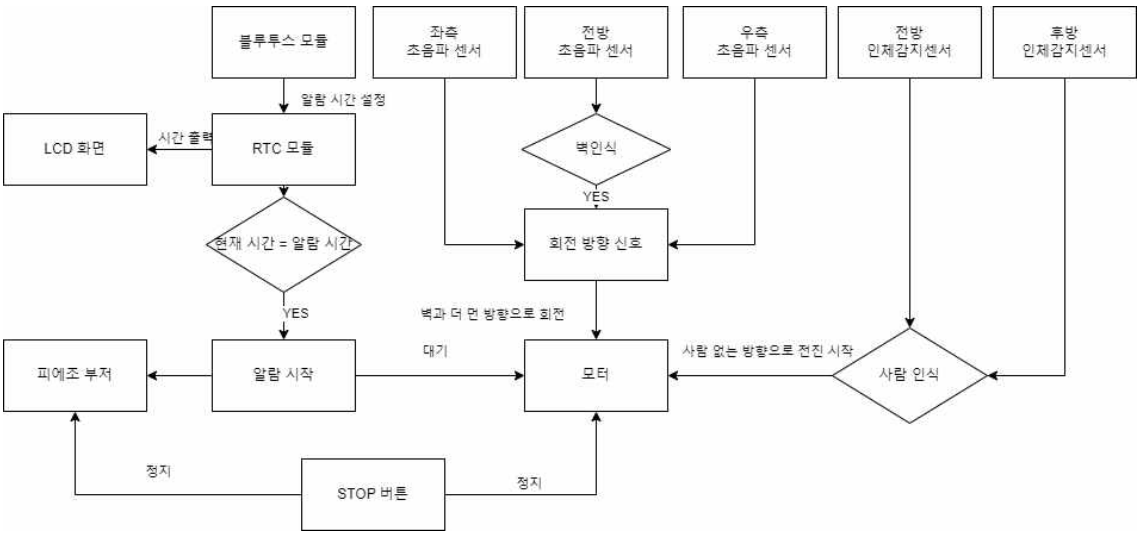
- 신호를 받은 기기는 부저를 울린다.
- 기기가 벽에 닿으려 할 시, 초음파 센서를 통해 방향을 전환하여 벽과 닿지 않도록 한다.
- 기기의 버튼을 누르면 알람이 종료된다.
- LCD에는 RTC모듈을 이용해 현재 시간을 보여준다.

3. 사용센서 및 모듈

초음파 거리센서 모듈 HC-SR04 - 초음파가 되돌아오는 시간을 통해 장애물까지의 거리를 감지 - 진행 방향의 장애물을 확인함 - 좌우 방향 장애물을 확인 후 더 장애물이 없는 곳으로 회피한다	
인체감지센서모듈 HC-SR501 [SZH-EK052] - 적외선을 통해 인체 감지 - 진행 방향의 사람을 확인함 - 사람을 감지하면 뒤로 돌아 도망간다	
3-24V 알람 경보용 피에조 부저 [FQ-050] - 알람을 울린다	
DS3231 RTC 고정밀 리얼타임 클럭 모듈 [SZH-EK047] - 시간 측정을 위해 사용 - I2C 통신 사용	
스마트 RC카 DV1802-KIT - 모터와 바퀴로 알람을 움직이게한다. - 모터 방향 조정을 통해 직진, 회전, 후진을 구현한다.	

4. 시나리오

흐름도



1) 시간 표시

- RTC모듈을 이용하여 현재 시간을 확인한 후 LCD에 보여준다.

2) 알람 설정

- 사용자는 스마트폰의 블루투스를 통해 원하는 시간에 알람을 설정한다.

3) 알람 시작

- 알람 시각에 도달하면 부저를 울린다.

4) 알람의 이동

- 근처에 사람이 감지되면 모터가 작동되어 도망가기 시작한다.
  (인체감지 센서를 통한 interrupt 신호)
- 전방 센서에 감지되었을 시 180도 회전하여 도망간다.

5) 방향 바꾸기

- 앞으로 나아가는 기기의 전방 초음파 센서가 미리 설정한 거리 안에 벽이 있다는 신호를 보내면 좌, 우의 초음파 센서값을 읽어 벽과의 거리가 더 먼 곳으로 회전한다. (timer로 음파 돌아오는 시간을 측정하여 거리 계산)

6) 알람 종료

- 사용자가 기기를 잡아 버튼을 누르면 알람이 종료된다
- 기기의 이동과 부저가 모두 정지한다.

5. 프로젝트 구현

1) 센서 및 모듈

1-1) 초음파 센서

```
void EnableHCSR04PeriphClock() {
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM5, ENABLE);
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);
 }
```

초음파 센서는 TIMER를 이용하여 초음파가 물체에 부딪히고 돌아오는데 걸리는 시간을 측정한다. 이를 통하여 물체까지의 거리를 측정한다. 총 3개의 초음파센서에 TIM2, TIM3, TIM5를 부여하였다.

```
int32_t HCSR04GetDistance() {
   (US_TIMER)->CNT = 0;
   TIM_Cmd(US_TIMER, ENABLE);
   while(!TIM_GetFlagStatus(US_TIMER, TIM_FLAG_Update));
   TIM_Cmd(US_TIMER, DISABLE);
   TIM_ClearFlag(US_TIMER, TIM_FLAG_Update);
   return (TIM_GetCapture2(US_TIMER)-TIM_GetCapture1(US_TIMER))*165/1000;
}
```

거리를 측정하는 함수이다. 이를 main의 while문에서 call하여 장애물 여부를 확인한다.

1-2) 블루투스 모듈

블루투스 모듈과 보드의 통신을 UART4로 하였다. USART2와 USART3의 포트가 초음파의 TIMER 사용으로 겹치는 문제로 UART4를 선택하였다.

```
void UART4_IRQHandler() {
   uint16_t word;
    if(USART_GetITStatus(UART4,USART_IT_RXNE)!=RESET){
       // the most recent received data by the USART1 peripheral
       word = USART_ReceiveData(UART4);
       printf("receive4 : %c\n", word);

       if (start_offset == 1) {
         if (word == 0x3b) { //SetAlarm 7-19:20 [0x53, 0x65, 0x74, 0x41, 0x6c, 0x61, 0x72
           printf("start partition\n");
           string_partition();
           start_offset = 0;
         }
         else {
           receive_string[string_count] = word;
           string_count++;

           // clear 'Read data register not empty' flag
           USART_ClearITPendingBit(UART4,USART_IT_RXNE);
         }
       }

       if (word == 0x40) {
         start_offset = 1;
       }
    }
}
```

UART4_IRQHandler()로 명령어에 대한 처리를 하였다. 0x40(@)은 명령어 시작을 의미한다. 그 후 명령어를 입력 받은 후 0x3b(;)을 입력 받으면 명령어 입력을 종료하고 명령어 해석과정을 거쳐 사용자의 명령을 처리한다.

시간 선택 명령어는 다음과 같다.

@SetAlarm DD-HH:MM;

위의 명령어로 기기의 알람을 설정할 수 있다.

1-3) RTC(Real Time Clock)

현재 시간을 저장하고 있고, 알람 설정이 가능하다. RTC의 시간 구조체를 사용하여 블루투스 모듈로부터 받은 알람시간을 저장한다. 해당 저장값과 현재 시간이 같으면 알람 Flag를 Set하여 기기를 구동한다.

1-4) LCD

현재 시간과 설정된 알람 시간을 출력한다.

1-5) 피에조 부저

SetBit와 ResetBit를 통하여 알람이 동작 중일 때 켜고 중지되었을 때 끈다.

1-6) 인체감지 센서

각각 PD0와 PD1에 pin mapping하여 EXTI0와 EXTI1으로 인터럽트를 부여하였다. 알람이 울리던 중 인체감지를 하게 되면 움직이기 시작한다. 전방에서 사람이 감지되면 뒤로 180도 회전한 후 전진한다.

1-7) 모터 드라이버

하나의 드라이버에 총 2개의 모터를 부착한다. 각 모터와 연결된 pin에 대해서 한쪽을 SetBit 한쪽을 ResetBit를 하여 모터를 구동 시킬 수 있다. 반대로 reset/set을 하게 되면 역방향으로 회전한다.

1-8) stop button

PC3에 연결하여 EXTI3로 사용한다. 버튼이 눌렸을 때 interrupt를 발생시켜 알람 동작을 즉시 멈춘다.

2) 프로그램 동작

main 함수 내의 while 문에서 알람시간과 현재시간을 읽어 같은지 반복해서 확인한다.

```
ds3231_read_time(&ds_time_default);
ds3231_read_alarm1(&alarm1_default);
alarm_check(&ds_time_default, &alarm1_default);
```

Alarm_ONOFF가 1이 되어 알람이 울려야 하면 내부의 while문에 진입하여 알람을 동작시킨다.

```
while(Alarm_ONOFF){
  directionFlag = 1;
  setDirection(directionFlag);
```

setDirection을 통해 기기를 전진을 하도록 한다.(directionFlag = 1은 전진)

```
//get distance
int32_t dist = HCSR04GetDistance();
int32_t dist2 = HCSR04GetDistance2();//left
int32_t dist3 = HCSR04GetDistance3();
//avoid wall
int isBiggerLeft = (dist2 >= dist3);
int isBiggerRight = (dist2 < dist3);
int timeD = delayByTim2()-timeNow;

if(dist <=400 && timeD>10){
  timeNow = delayByTim2();
  if(isBiggerLeft){
    while(1){//go left
      setDirection(4);
      int endCondition =(HCSR04GetDistance()>=150);
      delay(1000000);
      if(endCondition) break;
    }
  }
  else if(isBiggerRight){
    while(1){//go right
      setDirection(3);
      int endCondition =( HCSR04GetDistance()>=150);
      delay(1000000);
      if(endCondition) break;
    }
  }
}
```

dist, dist1, dist2의 값은 각 초음파 센서 1, 2, 3의 값을 받아온다. 이후 전방의 초음파센서 거리가 400이하이며 - 이전에 이 조건을 통과한 시간이 10초 이상일 때 - 좌, 우의 초음파 값을 비교해(isBiggerLeft or isBiggerRight) 나아갈 방향을 결정한다. 나아갈 방향이 정해지면 기기는 한 방향으로 회전하며, 전방의 센서 값이 150이상이 되면 회전을 종료한다.

이 동작에서 timeD의 역할은 이상치의 제거이다. 이를테면 초음파센서는 각종 상태에 노출된 결과로 정상적이지 않은 값을 출력할 수 있다. 이 경우 원치 않게 방향을 전환하는 기능이 반복적 수행될 수 있는데, “timeD > 10”라는 조건은 이를 미리 예방해준다. while문 내의 delay함수도 이와 같은 기능을 한다.

사전에 기기가 전진하는 조건에는 인체감지 센서의 값이 1인 것(즉, 인체가 감지된 것)이 존재하였다. 다만, 구현 과정에서 각종 센서 및 모듈의 연결로 전력이 부족하여 인체감지 센서의 정상적인 값의 출력을 기대하기 어려워졌다.

예를 들자면 2개의 외부전력을 사용하였음에도, 인체감지 센서 2개를 추가 연결 시 각 센서가 정상적으로 동작하지 않는 문제가 발생하였다.

그 결과 구현에서 후순위에 있다고 판단된 인체감지센서의 사용을 중지하였다.

6. 결론 및 토의

사람을 깨운다는 알람시계의 목적을 효율적으로 달성할 수 있는 움직이는 알람시계를 제작하였다.

이를 위해 3개의 초음파 센서, 블루투스 모듈, 모터 드라이버 모듈 및 모터 2개, 피에조 부저, Real Time Clock 모듈, 인체감지센서 2개를 이용하였다. 많은 모듈을 사용하는 만큼, 구현에 어려운 점이 존재했다. 그 예로 개별적으로는 원할하게 동작하던 모듈들이 통합하는 과정에서 문제가 생겼다.

STM32보드는 핀별로 각기 다른 기능을 하도록 매핑되어 있는데, 사용하고자 하는 핀의 각 기능이 한 핀에 몰려있는 경우 새로운 핀을 사용하기 위해 기존의 코드를 수정해야 하는 일이 빈번했다. 예를 들자면 Timer는 각 번호별(Timer1, Timer2... 등)로 부여받은 역할이 달랐고, 또한 그것을 사용을 위해서는 각기 다른 번호의 핀을 사용할 필요가 있었다. 잘 동작하던 코드가 핀을 바꾼 뒤 그렇지 않게 된 경우도 비일비재했다.

문제는 핀 부족뿐만이 아니었다. STM32 스키마를 보아도 명확하지 않은 회로의 문제도 있었다. 결국 각 모듈에 대한 내부 스키마를 모두 보고서야 온전히 회로도를 작성할 수 있었다.

많은 센서의 수는 회로를 복잡하게 만들었다. 설계 시 미리 작성해놓은 핀 번호 및 회로도를 보지 않고 새로운 동작을 추가할 수 없게 되었고, 모듈을 수행하며 확인하는 동적 테스팅을 수행하기 힘들게 되었다. 실행될 수 있는 테스트 방법은 모든 모듈을 종합하기 전까지는 다만 개별적인 모듈, 그리고 코드 레벨에서 뿐이었다.

끝으로 모든 단계를 마치고 기기를 작동시켰을 때, 전력이 부족하여 모든 센서를 작동시킬 수 없다는 문제가 있다는 것을 알 수 있었다. 이를 해결하기 위한 방법으로 1. 외부 전력 추가 연결하기 2. 다양한 최적의 전력값 찾기 3.

코드에서 각 센서의 기능을 죽이고 시험하기, 를 수행하였다. 그 결과 외부전력 2개를 사용하고, 우선순위가 낮은 인체감지 센서 2개를 사용하지 않는 것이 우리의 해결책이었다.

이외에도 각 모듈 단계에서는 정상 동작하던 기능들이 최종 테스트 단계에서 수정될 필요성이 생겼다.

이로써 우리의 코드 내에는 2개의 인터럽트로 구현된 인체감지센서가 stm32 보드에서는 사용하지 않게 되었다. 결론적으로 구현한 기능은 1. rtc모듈로 현재 시간 갱신 및 lcd에 출력 2. 블루투스 모듈로 명령어를 받아 알람을 설정하는 기능 3. 알람이 발생하면 달리며 전방의 장애물을 인식 및 회피하는 기능, 이외에 알람을 울리고 끄는 기본적 기능을 구현하게 되었다.

단순히 한 줄로 요약되는 이 기능들을 구현하기 위해 많은 시간이 소요되었다. 하지만 오히려 들인 시간만큼 stm32를 포함한 임베디드시스템 설계에 대한 이해력이 느는 계기가 되었다.

개선할 점

이번 실험을 통해 느낀 사실은, 개발에 앞서 설계 역시 큰 비중을 두고 살펴야 한다는 것이다. 앞서 언급한 문제점들은 개발 이전 설계 단계에 더욱 집중했다면 사전에 차단될 수 있는 문제였다. 그 결과 핀 부족 및 전력 부족이라는 예상치 못한 사태를 맞게 되었다.

더욱이 구현에 대한 아쉬운 점도 다수 있다. 말하자면 알람을 설정하는 커맨드를 더욱 다양하게 구현할 여지가 있었으며, 기기의 움직임을 더욱 다채롭게 할 가능성이 있었다. 하지만 결함을 해결하는 데에 많은 시간이 들어 그것을 실현하지 못한 점이 아쉽다.

다음번에 기기를 설계, 기획 및 제작하게 될 기회가 생긴다면 이번 임베디드 시스템 설계 및 실험 과목의 프로젝트에서 느낀 점을 계기로 더욱 나은 결과물을 만들 수 있을 것이다.