임베디드 시스템 설계 및 실험
텀 프로젝트 최종보고서
- Smart Parking System -

화요일 7조
201424470 서민영
201424421 김시은
201424533 정종진
201424532 정재광

목차

1. 설계목표

- 실험시간에 배운 여러 센서 및 보드의 기능을 이용하여 하드웨어를 개발한다.
- bluetooth 및 통신 관련 기능을 이용하여 하드웨어를 개발한다.
- 실생활에서의 불편한 점을 임베디드 설계를 통해 해소한다.
- 주차장에서 주차 공간을 찾을 때 시야 확보를 방해하여 안전성에 문제가 있고, 차량이 많을 때는 주차 공간을 쉽게 찾을 수 없는 문제를 해결하여 준다.

2. 설계 시스템의 블록 다이어그램

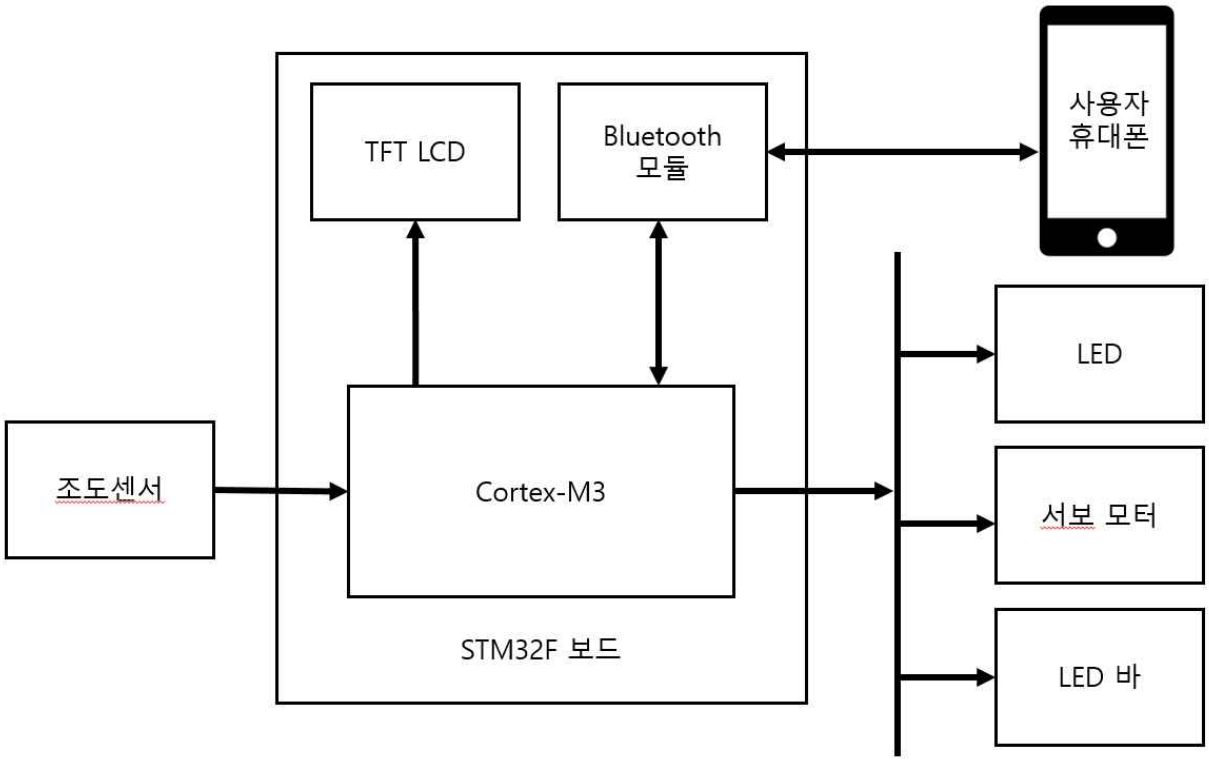


그림 1 : Smart Parking System의 블록 다이어그램

3. 설계 시스템의 흐름도

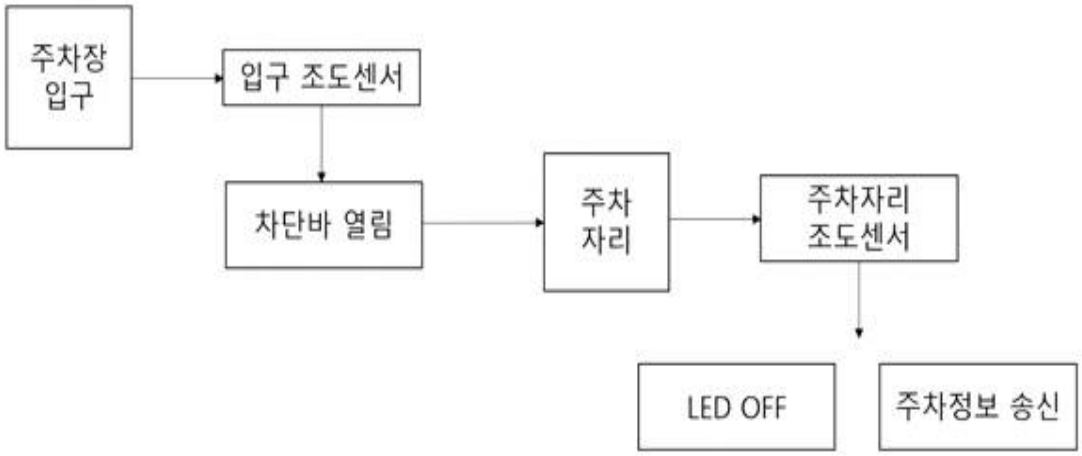


그림 2 : 입차 시 시스템 흐름도

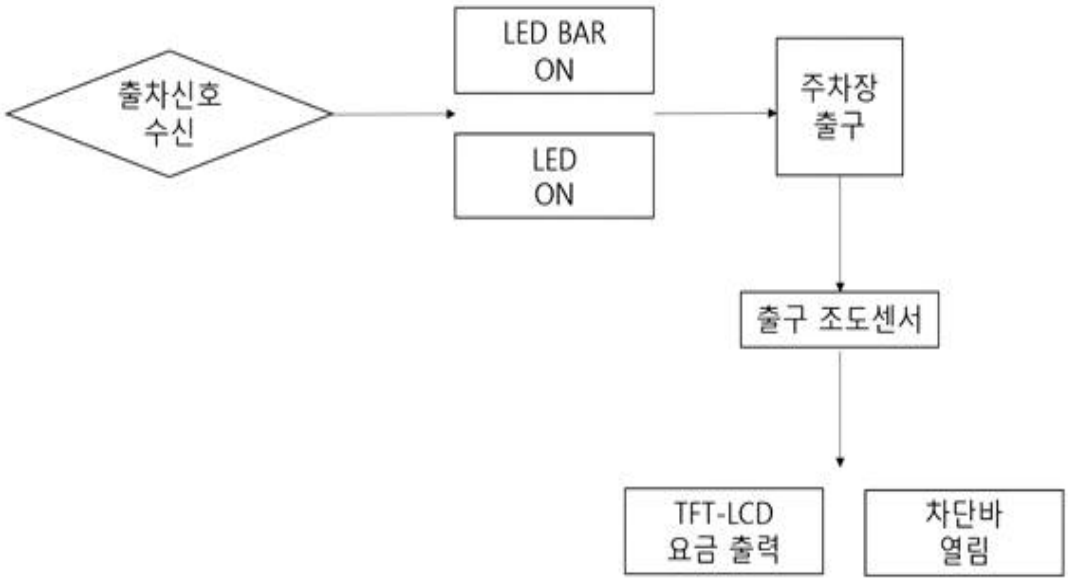
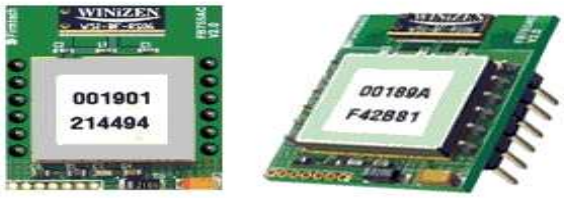


그림 3 : 출차 시 시스템 흐름도

4. 설계 시스템에 사용된 부품 설명

표 1 : 설계 시스템에 사용한 부품 소개

<table>
<tr><td>블루투스 모듈 : FB755AC (1EA)
- 블루투스 프로파일 : GAP, SPP
- 블루투스 버전 : V2.0
- 통신거리 : 100m(Class1)
- 제품크기 : 27.7 x 20.6 mm
- 인터페이스 : UART(TTL)
- 입력전원 : 3.3VDC +/- 0.2V</td><td>  </td></tr>
<tr><td colspan="2">사용목적 : 주차장 사용자가 출차 및 입차 시 보드와 통신하기 위하여 사용. 입차 시 블루투스 모듈에서 사용자 휴대폰으로 주차 정보를 보내며 출차 시 사용자 휴대폰에서 블루투스 모듈로 출차 신호를 보냄.</td></tr>
<tr><td>적색 LED (2EA)
- 발광 색상 : 적색
- LED 크기 : 5mm
- 렌즈 모양 : 원형
- 파장 : 620 ~ 630nm
- 광도 : 8000 ~ 10000 mcd
- 시야각 : 30°
- 순방향 전압 : 1.9 ~ 2.1V
- 순방향 전류 : 20mA
- 최대 작동 온도 : 100°C
- 최소 작동 온도 : - 20°C</td><td>  </td></tr>
<tr><td colspan="2">사용목적 : 주차장 자리에 주차가 되어있는지 나타내는 신호로 사용. 주차가 되어있으면 LED가 꺼지고 주차가 되어 있지 않으면 LED가 켜진다.</td></tr>
<tr><td>LED BAR (1EA)
고휘도 3528칩을 사용한 플렉시블 타입의 슬림 LED로 뒷면에는 양면테이프, 끝부분에는 전선처리가 되어있음
- 전압 : DC 5V
- 사이즈 : 폭 8mm / 전선 90mm
- 전체너비 130mm
- 색상 : 화이트</td><td>  </td></tr>
<tr><td colspan="2">사용목적 : 주차장 사용자가 출차 시에 출차 신호를 보내면 LED BAR가 작동하여 출구로 가는 가이드 라인을 표시한다.</td></tr>
</table>

조도센서 (4EA)

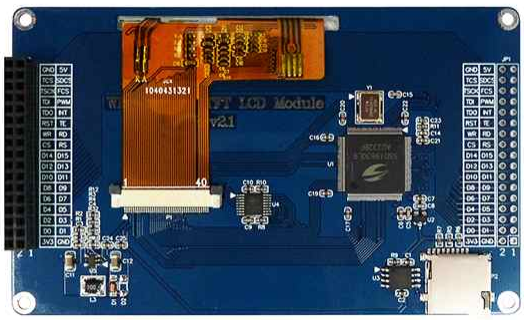
- Type : GL5528
- Max voltage : 150
- Max power : 100
- Environmental temp : -30 ~ 70
- Spectrum peak value : 540
- Light resistance : 10 ~ 20
- Dark resistance : 1

사용목적 : 주차장 입구 및 출구에 각각 하나씩, 주차장 자리 1, 2에 각각 하나씩 설치되어 사용자 자동차의 움직임을 관찰한다.

서보모터 (1EA)

- Control system : Pulse width control 1.5m/sec neutral
- Operating voltage range : 4.8[V] to 6.0[V]
- Operating temperature range : -20℃ to 60℃
- Operating speed : 0.19sec at 4.8V / 0.15sec at 6.0V
- Stall torque : 3.5kg.f.cm at 4.8V / 4.3kg.f.cm at 6.0V
- Operating angle : 60˚at one side pulse traveling 0.6m/sec
- Direction : C/W Pulse traveling 1.5m/sec to 2.1m/sec
- Idle current : 3[mA]
- Running current : 180[mA]
- Dead band width : 3μs
- Connector wire length : 300mm
- Dimensions : 40*20*36.6mm
- Weight : 40g

사용목적 : 주차장 입구에 있어 차단바로 사용되는 센서이다. 차단바 양옆으로 조도센서가 설치되어 이 조도센서를 동작시키면 차단바가 올라가거나 내려간다.

TFT-LCD (1EA) 와이드 4.3 인치 480 x 272 해상도, 터치판넬형TFTLCD 모듈. SSD1963을 내장하여 AVR, PIC, C51, ARM 등의 CPU에서 손쉽게 제어할 수 있음. 터치 콘트롤러 TSC2046, LCD 콘트롤러 SSD1963, mini-SDcard용 소켓, 시리얼 16M Bit 플래쉬 메모리(25Q16)등이 장착되어 있음. - SSD1963 디스플레이 컨트롤러 칩 - 24비트 RGB 인터페이스(표준 40핀 인터페이스) - 표시색상 : 16만 - 화면크기 : 4.3" (16:9) - 화면해상도 : 480×272 픽셀 - Mini-SD 메모리 카드 지원 - 16Mbit Serial Flash(25Q16) 메모리 탑재 - 터치스크린, 터치 컨트롤칩(TSC2046) 내장 - STM32F103ZE, STM32F103VC, STM32F107VC 개발보드와 pip to pin 호완 - 낮은 전력소비	 
사용목적 : 주차장 상태를 표시하고 주차장 출차 시 사용자가 지불해야 되는 요금을 표시한다. 추가로 현재 시간을 나타낸다.	

5. Application 동작 설명 및 세부사항

주차관리 시스템은 무인 주차관리 시스템으로 조도센서를 사용하여 자동차의 위치를 감지하여 차단기와 LED, 가이드라인이 동작하고 주차요금을 계산하며 TFT LCD에 주차장의 남은자리와 현재시간, 주차요금을 출력하는 시스템이다.

세부적으로 입차 동작 시 입구 조도센서에 자동차가 감지될 경우 서보모터가 동작하여 차단기가 올라가며 주차요금 계산을 위해 입차시간을 저장한다. 주차장 각 자리의 LED를 통해 주차가능한 자리를 확인할 수 있으며, 주차칸에 주차하면 블루투스를 통해 사용자에게 주차자리와 입차시간을 전송하므로 사용자가 주차위치를 쉽게 찾을 수 있다. 주차칸의 조도센서에 자동차가 감지되면 해당 주차자리의 LED가 꺼지고 TFT LCD의 주차장 그림에서 해당 주차칸에 X표시가 되고 현재 주차장의 주차가능 차량대수 정보를 출력한다.

출차 동작 시 사용자가 스마트폰을 통해 주차자리를 블루투스로 전송하면 바닥의 가이드라인을 통해 출구까지 안내하여 출구를 쉽게 찾을 수 있다. 출구 조도센서에 자동차가 감지될 경우 가이드라인이 꺼지고 TFT LCD에 주차요금을 출력한다.

6. 구현 결과

- 조도 센서를 통해 차량의 출/입 여부와, 주차 여부를 판단
- 서보 모터를 사용하여 주차장 출입문 개폐 동작
- 블루투스 모듈을 사용하여 주차장 사용자에게 입차 시간과 주차자리를 전송
- 플렉시블 타입의 고휘도 슬림 LED를 사용하여 주차장 출구를 쉽게 찾을 수 있도록 가이드 라인 구현

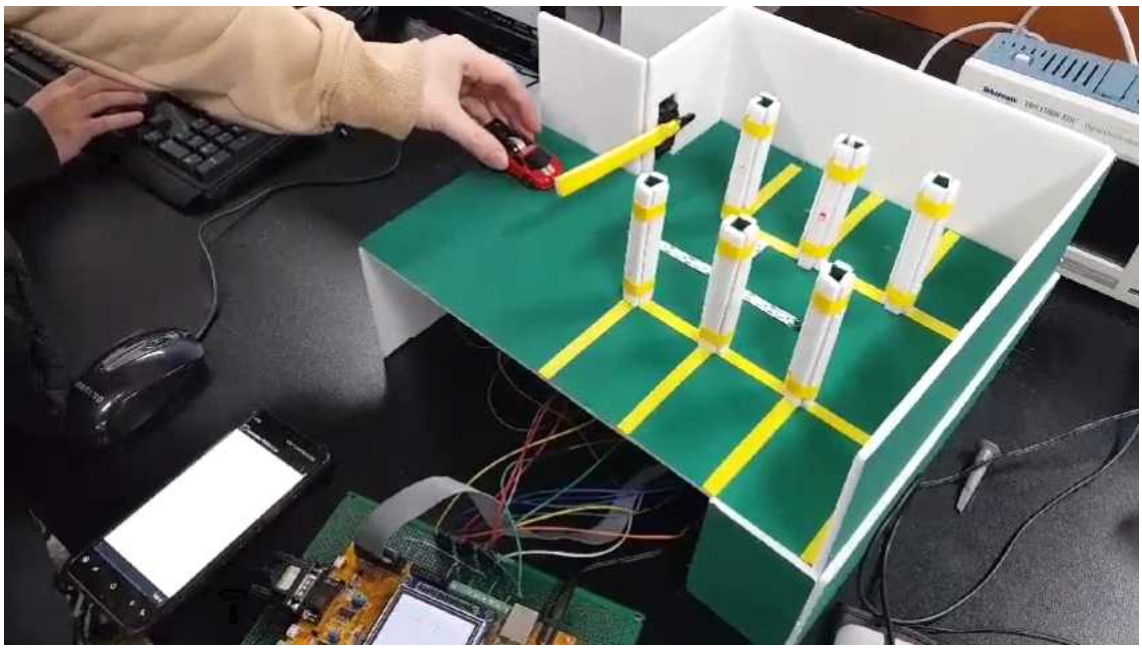


그림 4 : 출/입 여부 판단 및 서브 모터를 통한 출입문 개폐(1)

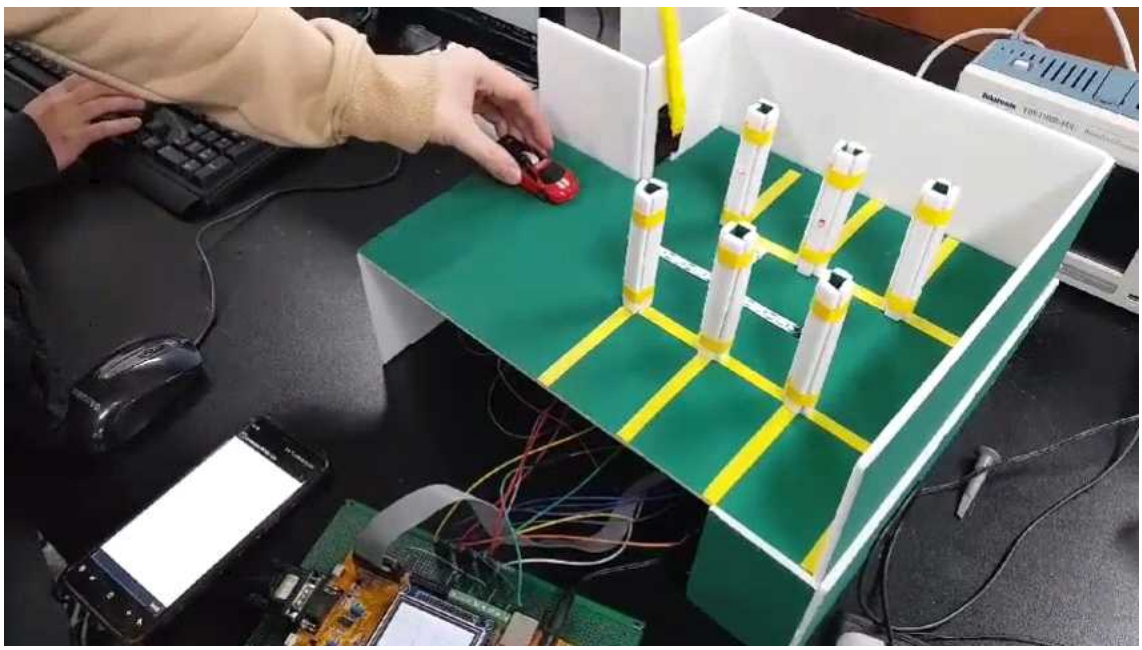


그림 5 : 출/입 여부 판단 및 서브 모터를 통한 출입문 개폐(1)

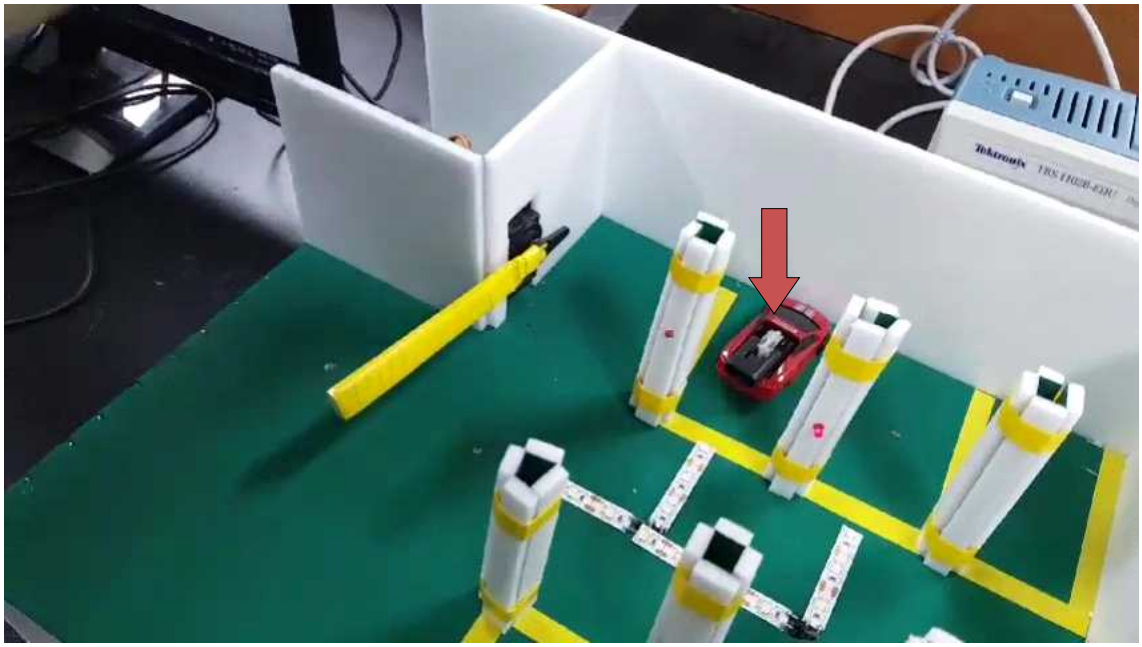


그림 6 : 조도 센서를 통한 각 자리 주차 여부 판단

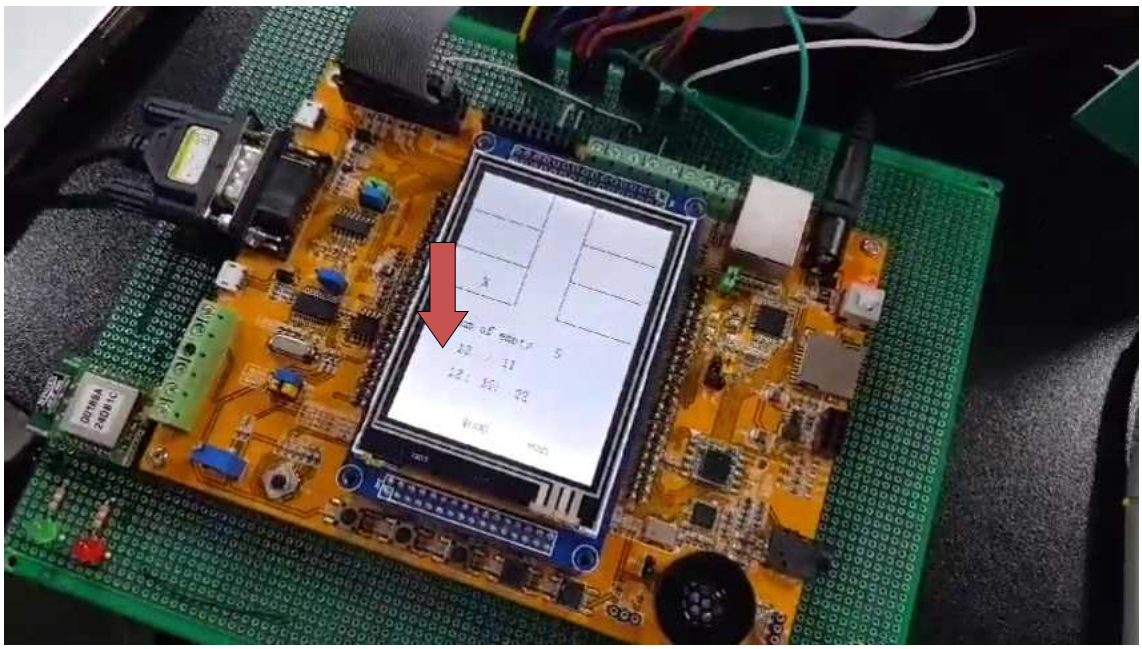


그림 7 : TFT LCD에 각 자리 주차 여부 표시

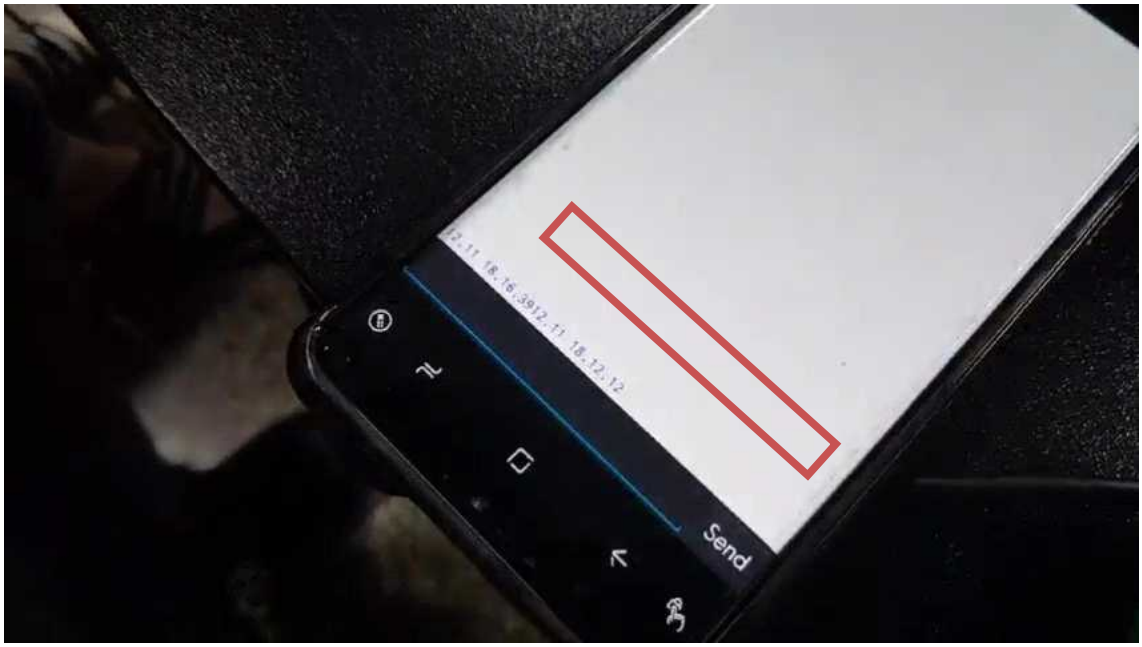


그림 9 : 블루투스 모듈을 통한 입차시간과 주차자리 전송

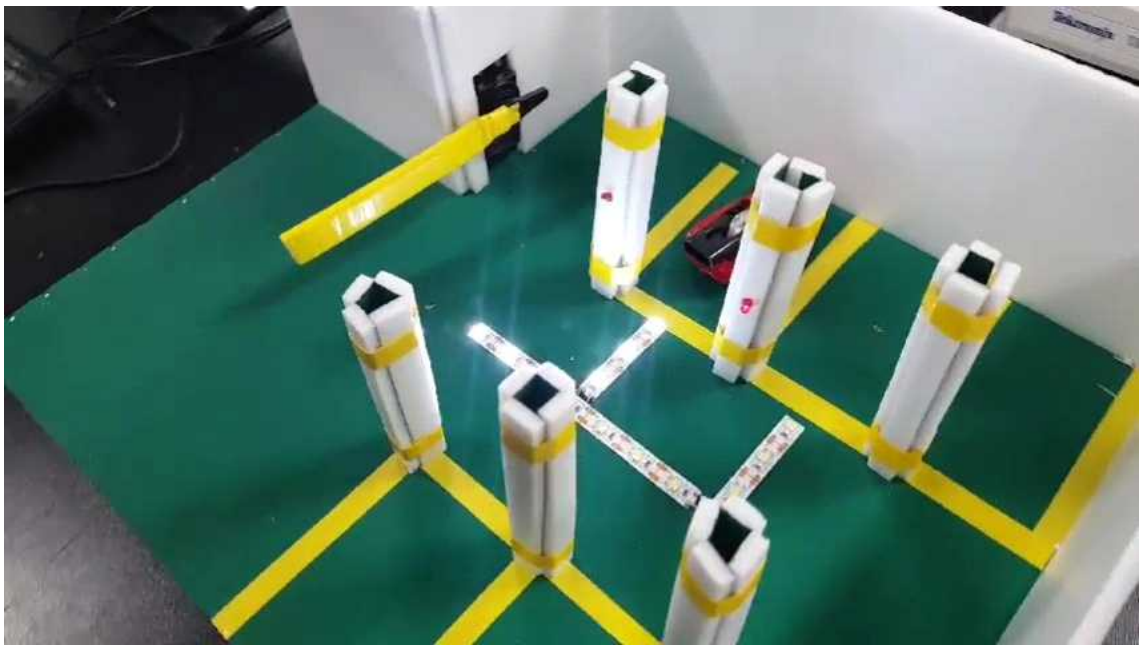


그림 10 : 출차 시 LED 바를 통한 출구 안내

7. 소스코드

main.c

```
#include "stm32f10x.h"
#include "stm32f10x_gpio.h"
#include "stm32f10x_rcc.h"
#include "stm32f10x_usart.h"
#include "stm32f10x_adc.h"
#include "stm32f10x_exti.h"
#include "stm32f10x_dma.h"
#include "stm32f10x_tim.h"
#include "misc.h"
#include "core_cm3.h"
#include "stdio.h"
#include "stdlib.h"
#include "lcd.h"
#include "touch.h"

int flag = 0, out_flag1 = 0, out_flag2 = 0;
int send_flag = 0;
int t_msec = 0;
int time_save1[2] = {0,};
int time_save2[2] = {0,};
//int color[12]=
//{WHITE,CYAN, BLUE,RED,MAGENTA,LGRAY,GREEN,YELLOW,BROWN,BRRED,GRAY};

vu32 ADC_VAL[] = {0, 0, 0, 0};
int pCount = 2;
int t_sec = 0, t_min = 11, t_hour = 18;
int d_day = 11, d_month = 12;

char ch_time[] = "12.11 18.11.00";

void delay() {
   int i=0;
   for(i=0; i<1000000; i++)
      ;
}

// Bluetooth 구현 부분 1
void EXTI11_Config(void)
{
   GPIO_InitTypeDef GPIO_InitStructure;
   EXTI_InitTypeDef EXTI_InitStructure;
   NVIC_InitTypeDef NVIC_InitStructure;
   RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD, ENABLE);


   GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11;
   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPD;
   GPIO_Init(GPIOD, &GPIO_InitStructure);
   RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);

   GPIO_EXTILineConfig(GPIO_PortSourceGPIOD, GPIO_PinSource11);
```

```
    EXTI_InitStructure.EXTI_Line = EXTI_Line11;
    EXTI_InitStructure.EXTI_LineCmd = ENABLE;
    EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
    EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Rising_Falling;
    EXTI_Init(&EXTI_InitStructure);

    NVIC_InitStructure.NVIC_IRQChannel = EXTI15_10_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x00;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x01;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
}

// Blutooth 구현 부분 2
void send_com(char buf[]) {
    char *s = buf;
    while (*s) {
        while (USART_GetFlagStatus(USART1, USART_FLAG_TXE) == RESET)
            ;
        USART_SendData(USART1, *s++);
    }
}

// Blutooth 구현 부분 3
void send_phone(char buf[]) {
    char *s = buf;

    while (*s) {
        while (USART_GetFlagStatus(USART2, USART_FLAG_TXE) == RESET)
            ;
        USART_SendData(USART2, *s++);
    }
}

// Blutooth 구현 부분 4
void USART1_Init(void)
{
    USART_InitTypeDef usart1_init_struct;
    GPIO_InitTypeDef gpioa_init_struct;
    NVIC_InitTypeDef NVIC_InitStructure;

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1 | RCC_APB2Periph_AFIO |
    RCC_APB2Periph_GPIOA, ENABLE);

    gpioa_init_struct.GPIO_Pin = GPIO_Pin_9;
    gpioa_init_struct.GPIO_Speed = GPIO_Speed_50MHz;
    gpioa_init_struct.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_Init(GPIOA, &gpioa_init_struct);
    gpioa_init_struct.GPIO_Pin = GPIO_Pin_10;
    gpioa_init_struct.GPIO_Speed = GPIO_Speed_50MHz;
    gpioa_init_struct.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    GPIO_Init(GPIOA, &gpioa_init_struct);

    usart1_init_struct.USART_BaudRate = 9600;
    usart1_init_struct.USART_WordLength = USART_WordLength_8b;
    usart1_init_struct.USART_StopBits = USART_StopBits_1;
```

```
    usart1_init_struct.USART_Parity = USART_Parity_No;
    usart1_init_struct.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
    usart1_init_struct.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
    USART_Init(USART1, &usart1_init_struct);
    USART_Cmd(USART1, ENABLE);
    USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);

    NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x01;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x01;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
}

// Blutooth 구현 부분 5
void USART2_Init(void)
{
    USART_InitTypeDef usart2_init_struct;
    GPIO_InitTypeDef gpioa_init_struct;
    NVIC_InitTypeDef NVIC_InitStructure;

    RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART2, ENABLE);

    // tx, rx 설정
    gpioa_init_struct.GPIO_Pin = GPIO_Pin_2;
    gpioa_init_struct.GPIO_Speed = GPIO_Speed_50MHz;
    gpioa_init_struct.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_Init(GPIOA, &gpioa_init_struct);

    gpioa_init_struct.GPIO_Pin = GPIO_Pin_3;
    gpioa_init_struct.GPIO_Speed = GPIO_Speed_50MHz;
    gpioa_init_struct.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    GPIO_Init(GPIOA, &gpioa_init_struct);

    usart2_init_struct.USART_BaudRate = 9600;
    usart2_init_struct.USART_WordLength = USART_WordLength_8b;
    usart2_init_struct.USART_StopBits = USART_StopBits_1;
    usart2_init_struct.USART_Parity = USART_Parity_No;
    usart2_init_struct.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
    usart2_init_struct.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
    USART_Init(USART2, &usart2_init_struct);
    USART_Cmd(USART2, ENABLE);
    USART_ITConfig(USART2, USART_IT_RXNE, ENABLE);

    NVIC_InitStructure.NVIC_IRQChannel = USART2_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x01;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x01;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
}

// Blutooth 구현 부분 6
void USART1_IRQHandler(void) {
    unsigned char d;
    while (USART_GetFlagStatus(USART1, USART_FLAG_TXE) == RESET)
        ;
```

```
    d = (unsigned char) USART_ReceiveData(USART1);
    USART_SendData(USART2, d);
    USART_ClearITPendingBit(USART1, USART_IT_RXNE);
}

// Blutooth 구현 부분 7
void USART2_IRQHandler(void) {
    unsigned char d;
    while (USART_GetFlagStatus(USART2, USART_FLAG_TXE) == RESET)
        ;

    d = (unsigned char) USART_ReceiveData(USART2);
    USART_SendData(USART1, d);

    if(d == '1') {
        out_flag1 = 1;
    } else if(d == '2') {
        out_flag2 = 1;
    }

    USART_ClearITPendingBit(USART2, USART_IT_RXNE);

}

// Servo Motor 구현 부분 1
TIM_OCInitTypeDef TIM_OCInitStructure;
TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;

void RCC_Configure(void){
     RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD, ENABLE);
     RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);
     RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);
     RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);
     RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1, ENABLE);
     RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
     RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);
}

// System Init
void SysInit(void) {
    /* Set HSION bit */
    /* Internal Clock Enable */
    RCC->CR |= (uint32_t)0x00000001; //HSION
                                /* Reset SW, HPRE, PPRE1, PPRE2, ADCPRE and MCO bits */
    RCC->CFGR &= (uint32_t)0xF0FF0000;
    /* Reset HSEON, CSSON and PLLON bits */
    RCC->CR &= (uint32_t)0xFEF6FFFF;
    /* Reset HSEBYP bit */
    RCC->CR &= (uint32_t)0xFFFBFFFF;
    /* Reset PLLSRC, PLLXTPRE, PLLMUL and USBPRE/OTGFSPRE bits */
    RCC->CFGR &= (uint32_t)0xFF80FFFF;
    /* Reset PLL2ON and PLL3ON bits */
    RCC->CR &= (uint32_t)0xEBFFFFFF;
    /* Disable all interrupts and clear pending bits  */
    RCC->CIR = 0x00FF0000;
    /* Reset CFGR2 register */
    RCC->CFGR2 = 0x00000000;
```

```
}

// System Clock
void SetSysClock(void)
{
   volatile uint32_t StartUpCounter = 0, HSEStatus = 0;
   /* SYSCLK, HCLK, PCLK2 and PCLK1 configuration ----------------------------*/
   /* Enable HSE */
   RCC->CR |= ((uint32_t)RCC_CR_HSEON);

   /* Wait till HSE is ready and if Time out is reached exit */
   do
   {
      HSEStatus = RCC->CR & RCC_CR_HSERDY;
      StartUpCounter++;
   } while ((HSEStatus == 0) && (StartUpCounter != HSE_STARTUP_TIMEOUT));

   if ((RCC->CR & RCC_CR_HSERDY) != RESET)
   {
      HSEStatus = (uint32_t)0x01;
   }
   else
   {
      HSEStatus = (uint32_t)0x00;
   }
   if (HSEStatus == (uint32_t)0x01)
   {
      /* Enable Prefetch Buffer */
      FLASH->ACR |= FLASH_ACR_PRFTBE;
      /* Flash 0 wait state */
      FLASH->ACR &= (uint32_t)((uint32_t)~FLASH_ACR_LATENCY);
      FLASH->ACR |= (uint32_t)FLASH_ACR_LATENCY_0;
      /* HCLK = SYSCLK */
      RCC->CFGR |= (uint32_t)RCC_CFGR_HPRE_DIV1;
      /* PCLK2 = HCLK */
      /*@TODO*/
      RCC->CFGR|= (uint32_t)RCC_CFGR_PPRE2_DIV1;
      /* PCLK1 = HCLK */
      RCC->CFGR |= (uint32_t)RCC_CFGR_PPRE1_DIV2;
      /* Configure PLLs ------------------------------------------------------*/
      /* PLL configuration: PLLCLK = ???? */
      /*@TODO*/
      RCC->CFGR &= (uint32_t)~(RCC_CFGR_PLLXTPRE | RCC_CFGR_PLLSRC | RCC_CFGR_PLLMULL);
      RCC->CFGR |= (uint32_t)(RCC_CFGR_PLLXTPRE_PREDIV1 | RCC_CFGR_PLLSRC_PREDIV1 |
       RCC_CFGR_PLLMULL4);
      /* PLL2 configuration: PLL2CLK = ???? */
      /* PREDIV1 configuration: PREDIV1CLK = ???? */
      /*@TODO*/
      RCC->CFGR2 &= (uint32_t)~(RCC_CFGR2_PREDIV2 | RCC_CFGR2_PLL2MUL |
       RCC_CFGR2_PREDIV1 | RCC_CFGR2_PREDIV1SRC);
      RCC->CFGR2 |= (uint32_t)(RCC_CFGR2_PREDIV2_DIV5 | RCC_CFGR2_PLL2MUL10 |
       RCC_CFGR2_PREDIV1SRC_PLL2 | RCC_CFGR2_PREDIV1_DIV10);
      /* Enable PLL2*/
      RCC->CR |= RCC_CR_PLL2ON;
      /* Wait till PLL2 is ready */
      while ((RCC->CR & RCC_CR_PLL2RDY) == 0)
      {
```

```
      }
      /* Enable PLL */
      RCC->CR |= RCC_CR_PLLON;
      /* Wait till PLL is ready */
      while ((RCC->CR & RCC_CR_PLLRDY) == 0)
      {
      }
      /* Select PLL as system clock source */
      RCC->CFGR &= (uint32_t)((uint32_t)~(RCC_CFGR_SW));
      RCC->CFGR |= (uint32_t)RCC_CFGR_SW_PLL;
      /* Wait till PLL is used as system clock source */
      while ((RCC->CFGR & (uint32_t)RCC_CFGR_SWS) != (uint32_t)0x08)
      {
      }
      /* Select System Clock as output of MCO */
      //@TODO
      RCC->CFGR &= (uint32_t)((uint32_t)~(RCC_CFGR_MCO));
      RCC->CFGR |= (uint32_t)RCC_CFGR_MCO_SYSCLK;
   }
   else
   { /* If HSE fails to start-up, the application will have wrong clock
     configuration. User can add here some code to deal with this error */
   }
}

// Servo Motor 구현 부분 2
void TIME_Configuration_MOTOR(void) {
   TIM_TimeBaseStructure.TIM_Prescaler = 1000/36;
   TIM_TimeBaseStructure.TIM_Period = 14399;
   TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;
   TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
   TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);
   TIM_Cmd(TIM2, ENABLE);
}

// TIMER 구현 부분 1
void TIME_Configuration_TIMER(void) {
   TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
   NVIC_InitTypeDef NVIC_InitStructure;

   // TIM3 Clock Enable
   RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);

   // Enbale TIM21 Global Interrupt
   NVIC_InitStructure.NVIC_IRQChannel = TIM3_IRQn;
   NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
   NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
   NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
   NVIC_Init(&NVIC_InitStructure);

   // TIM3 Initialize
   // 1/40MHz * prescaler * period
   TIM_TimeBaseStructure.TIM_Period = 1000000-1;
   TIM_TimeBaseStructure.TIM_Prescaler = 40-1; /* check*/

   TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;
   TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
```

```
    TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);

    // TIM3 Enable
    // TIM_ARRPreloadConfig(TIM2, ENABLE);
    TIM_Cmd(TIM3, ENABLE);
    TIM_ITConfig(TIM3, TIM_IT_Update, ENABLE); // interrupt enable
}

// TIMER 구현 부분 2
void TIM3_IRQHandler(void) {

    if ((TIM_GetITStatus(TIM3, TIM_IT_Update) != RESET)){
        t_msec++;
        if(flag!=0 && send_flag==0) {
            send_phone(ch_time);
            time_save1[0] = t_sec;
            time_save1[1] = t_min;
            time_save2[0] = t_sec;
            time_save2[1] = t_min;
            send_flag = 1;
            //send_phone(flag+'0');
    }

        TIM_ClearITPendingBit(TIM3, TIM_IT_Update);
    }

}

void GPIO_Configuration(void) {
    GPIO_InitTypeDef GPIO_InitStructure;
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2 | GPIO_Pin_3 | GPIO_Pin_4 | GPIO_Pin_7;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_10MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_Init(GPIOD, &GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_10MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
    GPIO_Init(GPIOC, &GPIO_InitStructure);
    GPIOD->CRL = GPIO_CRL_MODE2_0 ;
}


void LED_Configure(void){
     GPIO_InitTypeDef GPIO_InitStructure;
     // LED 1
     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
     GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
     GPIO_Init(GPIOC, &GPIO_InitStructure);
     // LED 2
```

```
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_Init(GPIOC, &GPIO_InitStructure);
    // GUIDE_1
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_7;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_Init(GPIOC, &GPIO_InitStructure);
    // GUIDE_2
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_Init(GPIOC, &GPIO_InitStructure);
    // GUIDE_3
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_Init(GPIOC, &GPIO_InitStructure);
    // GUIDE_4
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_Init(GPIOC, &GPIO_InitStructure);
}

// Servo Motor 구현 부분 3
void PWM1_Configuration(void) {
   TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
   TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
   TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
   TIM_OCInitStructure.TIM_Pulse = 0;
   TIM_OC1Init(TIM2, &TIM_OCInitStructure);
   TIM_OC1PreloadConfig(TIM2, TIM_OCPreload_Enable);
   TIM_ARRPreloadConfig(TIM2, ENABLE);
}

// 조도 센서
void LSensor_Configure(void){
    GPIO_InitTypeDef GPIO_InitStructure;
    // PC1 - In
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_10MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
    GPIO_Init(GPIOC, &GPIO_InitStructure);
    // PC2 - Out
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_10MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
    GPIO_Init(GPIOC, &GPIO_InitStructure);
    // PC3 - Parking 1
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_10MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
    GPIO_Init(GPIOC, &GPIO_InitStructure);
    // PC4 - Parking 2
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4;
```

```
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_10MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
    GPIO_Init(GPIOC, &GPIO_InitStructure);
}

// Analog to Digital
void ADC_Configure(void){
    ADC_InitTypeDef ADC_InitStructure;
    ADC_DeInit(ADC1);
    ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
    ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
    ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
    ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
    ADC_InitStructure.ADC_NbrOfChannel = 4;
    ADC_InitStructure.ADC_ScanConvMode = ENABLE;
    ADC_ITConfig(ADC1, ADC_IT_EOC, ENABLE);

    ADC_Init(ADC1,&ADC_InitStructure);
    ADC_RegularChannelConfig(ADC1, ADC_Channel_11, 1, ADC_SampleTime_239Cycles5);
    ADC_RegularChannelConfig(ADC1, ADC_Channel_12, 2, ADC_SampleTime_239Cycles5);
    ADC_RegularChannelConfig(ADC1, ADC_Channel_13, 3, ADC_SampleTime_239Cycles5);
    ADC_RegularChannelConfig(ADC1, ADC_Channel_14, 4, ADC_SampleTime_239Cycles5);

    ADC_Cmd(ADC1, ENABLE);
    ADC_DMACmd(ADC1,ENABLE);
    ADC_ResetCalibration(ADC1);
    while(ADC_GetResetCalibrationStatus(ADC1)!=RESET);
    ADC_StartCalibration(ADC1);
    while(ADC_GetCalibrationStatus(ADC1)!=RESET);
}

// DMA
void DMA_Cofigure(void){
    DMA_InitTypeDef DMA_InitStructure;
    DMA_DeInit(DMA1_Channel1);
    DMA_InitStructure.DMA_BufferSize = 4;
    DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;
    DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
    DMA_InitStructure.DMA_MemoryBaseAddr = (uint32_t)ADC_VAL;
    DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_Word;
    DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
    DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
    DMA_InitStructure.DMA_PeripheralBaseAddr = (uint32_t)&ADC1->DR;
    DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_Word;
    DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
    DMA_InitStructure.DMA_Priority = DMA_Priority_High;
    DMA_Init(DMA1_Channel1, &DMA_InitStructure);
    DMA_Cmd(DMA1_Channel1,ENABLE);
}

// LED Bar
void GuideLineSet(int On){
    if(On == 1){
        GPIO_SetBits(GPIOC, GPIO_Pin_7);
        GPIO_SetBits(GPIOC, GPIO_Pin_8);
    }
    else if(On == 2){
```

```
        GPIO_SetBits(GPIOC, GPIO_Pin_7);
        GPIO_SetBits(GPIOC, GPIO_Pin_9);
        GPIO_SetBits(GPIOC, GPIO_Pin_10);
    }
    else if(On == 0){
        GPIO_ResetBits(GPIOC, GPIO_Pin_7);
        GPIO_ResetBits(GPIOC, GPIO_Pin_8);
        GPIO_ResetBits(GPIOC, GPIO_Pin_9);
        GPIO_ResetBits(GPIOC, GPIO_Pin_10);
    }
}

int day[] = {31,28,31,30,31,30,31,31,30,31,30,31};

int main(void){
   uint16_t r = 90, c = 50;
   uint16_t half_c = 25, gap = 52;
   char parkStr[] = "Num of empty ";
   int empty_num, park1, park2;
    int i = 0, flag_in = 0, flag_out = 0;
    int servoflag = 0;
    char adc_1[10], adc_2[10], adc_3[10], adc_4[10];


    SysInit();
    SetSysClock();
    EXTI11_Config();
    LCD_Init();
    Touch_Configuration();
    Touch_Adjust();
    LCD_Clear(WHITE);

    RCC_Configure();
    GPIO_Configuration();
    LSensor_Configure();
    LED_Configure();

    TIME_Configuration_MOTOR();
    PWM1_Configuration();
    TIME_Configuration_TIMER();


    USART1_Init();
    USART2_Init();

    ADC_Configure();
    DMA_Cofigure();

   /* for drawing parking line */
   for(i=1;i<=3;++i) {
     LCD_DrawLine(0,c*i,r,c*i);
     LCD_DrawLine(gap+r,c*i,gap+2*r,c*i);
   }

   LCD_DrawLine(r,0,r,3*c);
   LCD_DrawLine(gap+r,0,gap+r,3*c);
```

```
GuideLineSet(0);

 while(1){
    /* for drawing 'x' mark */
   for(i=0;i<2;++i) {
      if((int)ADC_VAL[3-i] > 4000)
         LCD_ShowChar(r/2, (3/2+i)*c+half_c, 'X', 16, BLACK, WHITE);

      else
         LCD_ShowChar(r/2, (3/2+i)*c+half_c, ' ', 16, BLACK, WHITE);
   }
   if(t_msec >= 10) {
       t_msec = 0;
       t_sec++;
   }
   if(t_sec >= 60) {
      t_sec = 0;
      t_min++;
   }

   if(t_min >= 60) {
      t_min = 0;
      t_hour++;
   }

   if(t_hour >= 24) {
      t_hour = 0;
      d_day++;
   }
   if(d_day >= day[d_month-1]) {
      d_day = 1;
      d_month++;

      d_month = d_month%12;
   }

   ch_time[0]=d_month/10 + '0';
   ch_time[1]=d_month%10 + '0';
   ch_time[3]=d_day/10 + '0';
   ch_time[4]=d_day%10 + '0';
   ch_time[6]=t_hour/10 + '0';
   ch_time[7]=t_hour%10 + '0';
   ch_time[9]=t_min/10 + '0';
   ch_time[10]=t_min%10 + '0';
   ch_time[12]=t_sec/10 + '0';
   ch_time[13]=t_sec%10 + '0';


   /* for drawing text */
   park1 = (int)ADC_VAL[2] > 4000 ? 1 : 0;
   park2 = (int)ADC_VAL[3] > 4000 ? 1 : 0;
   empty_num = 6 - park1 - park2;

   LCD_ShowString(30, 180, parkStr, BLACK, WHITE);
   LCD_ShowNum(130, 180, empty_num, 4, BLACK, WHITE);

   LCD_ShowNum(30, 210, d_month, 4, BLACK, WHITE);
```

```
        LCD_ShowString(80, 210, "/", BLACK, WHITE);
        LCD_ShowNum(85, 210, d_day, 4, BLACK, WHITE);

        LCD_ShowNum(30, 240, t_hour, 4, BLACK, WHITE);
        LCD_ShowString(65, 240, ":", BLACK, WHITE);
        LCD_ShowNum(70, 240, t_min, 4, BLACK, WHITE);
        LCD_ShowString(100, 240, ":", BLACK, WHITE);
        LCD_ShowNum(110, 240, t_sec, 4, BLACK, WHITE);

          ADC_SoftwareStartConvCmd(ADC1, ENABLE);
          while(ADC_GetFlagStatus(ADC1, SET)!=RESET);

          // 주차장 안 OR 밖 조도센서
          if((int)ADC_VAL[0] > 4000 || (int)ADC_VAL[1] > 4000) {
              TIM2->CCR1 = 520; // open

              if(out_flag1 == 1 && (int)ADC_VAL[1] > 4000){
                   int prev = time_save1[1]*60 + time_save1[0];
                   int now = t_min*60 + t_sec;
                   int collapse = now - prev;
                   int money = collapse/10 * 1000;

                   LCD_ShowNum(80,290, money, 4, BLACK, WHITE);
                   LCD_ShowString(160, 290, "won", BLACK, WHITE);

                 GuideLineSet(0);
                 out_flag1 = 0;
              }
              if(out_flag2 == 1 && (int)ADC_VAL[1] > 4000){
                   int prev = time_save2[1]*60 + time_save2[0];
                   int now = t_min*60 + t_sec;
                   int collapse = now - prev;
                   int money = collapse/10 * 1000;

                   LCD_ShowNum(80,290, money, 4, BLACK, WHITE);
                   LCD_ShowString(160, 290, "won", BLACK, WHITE);
                 GuideLineSet(0);
                 out_flag2 = 0;
              }

              delay();
          }

        // 주차장 안 AND 밖 조도센서
        if((int)ADC_VAL[0] < 4000 && (int)ADC_VAL[1] < 4000) {
            TIM2->CCR1 = 1300; // close
            delay();
        }


        // Parking 1
        if((int)ADC_VAL[2] > 4000) {
            flag = 1;
            // 주차장 구역 LED
            GPIO_ResetBits(GPIOC, GPIO_Pin_5);
            // 가이드라인
            if(out_flag1 == 1) {
```

```
                GuideLineSet(1);
            }
        }
        else {
          if(flag == 1) {
            send_flag = 0;
            flag = 0;
          }
           GPIO_SetBits(GPIOC, GPIO_Pin_5);
        }

        // Parking 2
        if((int)ADC_VAL[3] > 4000) {
            flag = 2;
            // 주차장 구역 LED
            GPIO_ResetBits(GPIOC, GPIO_Pin_11);
            // 가이드라인
            if(out_flag2 == 1) {
                GuideLineSet(2);
            }
        }
        else {
          if(flag == 2) {
              send_flag = 0;
              flag = 0;
          }
           GPIO_SetBits(GPIOC, GPIO_Pin_11);
        }

    }
    return 0;
}
```