피지컬 컴퓨팅

Lec. 10. ADC, USART, UI (ECG example)

Heenam Yoon

Department of
Human-Centered Artificial Intelligence

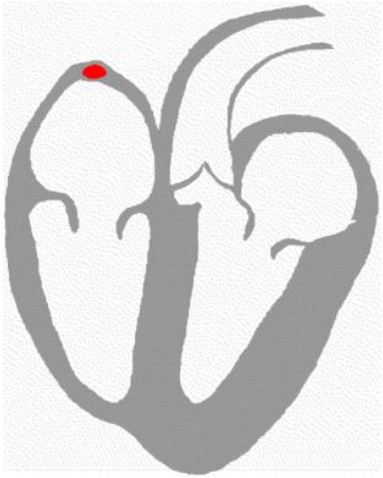
E-mail) h-yoon@smu.ac.kr
Room) 0112

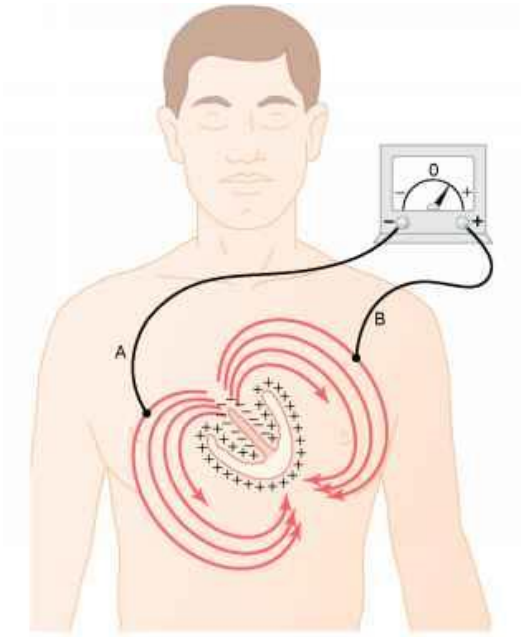
목표

- 심전도를 ADC한다 (백그라운드에서 계속 진행)
- ADC결과를 USART를 이용하여 PC로 전송한다
  - 초당 100개의 데이터를 수집하고 (100Hz=10ms=0.01초당 1번씩 데이터 수집)
  - 0.1초에 한번씩 USART로 데이터 전송
  - 이를 Timer를 통해 제어할 것임
- PC에서 수신한 데이터를 그래프로 그린다
- 심박수를 계산한다
- PC에서 보드로 어떤 결과를 전달한다

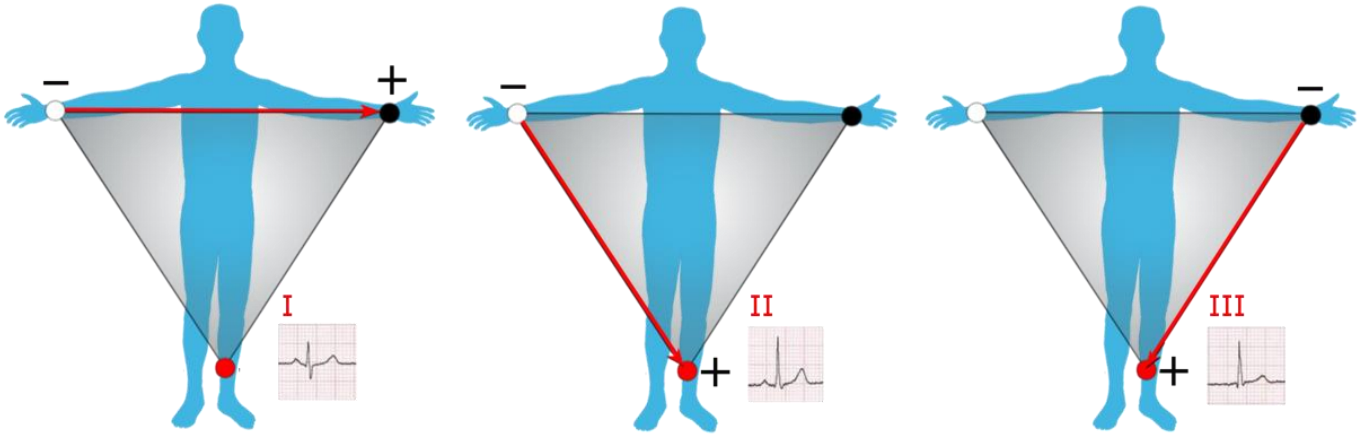
심전도

- 심장의 전기활동을 측정한 것
- 심근 수축 시 발생하는 전기적인 신호를 체표면에서 측정





심전도

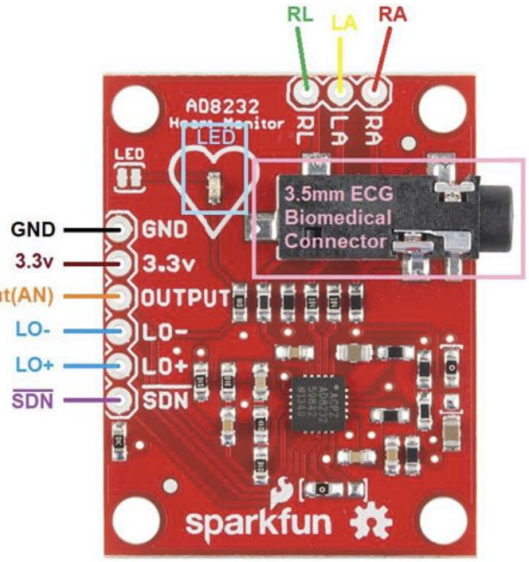


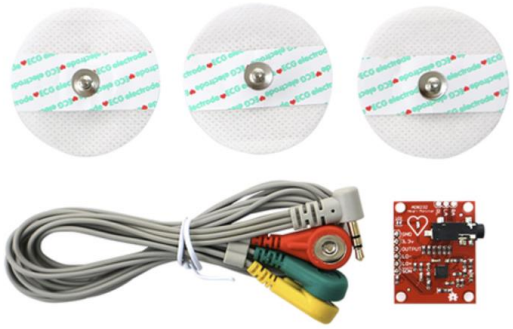
심전도 측정 회로

AD8232

- LO-, LO+: Leads Off Comparator Output. 몸에 전극이 잘 부착되었는지 감지해 주는 단자 (안 쓸 것임)
- SDN: Shutdown Control Input. Drive SDN low to enter the low power shutdown mode. (안 쓸 것임)

ADC 연결 ←

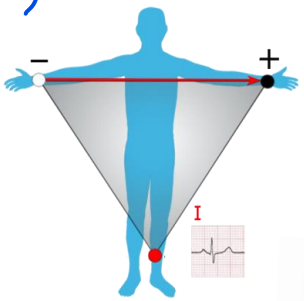


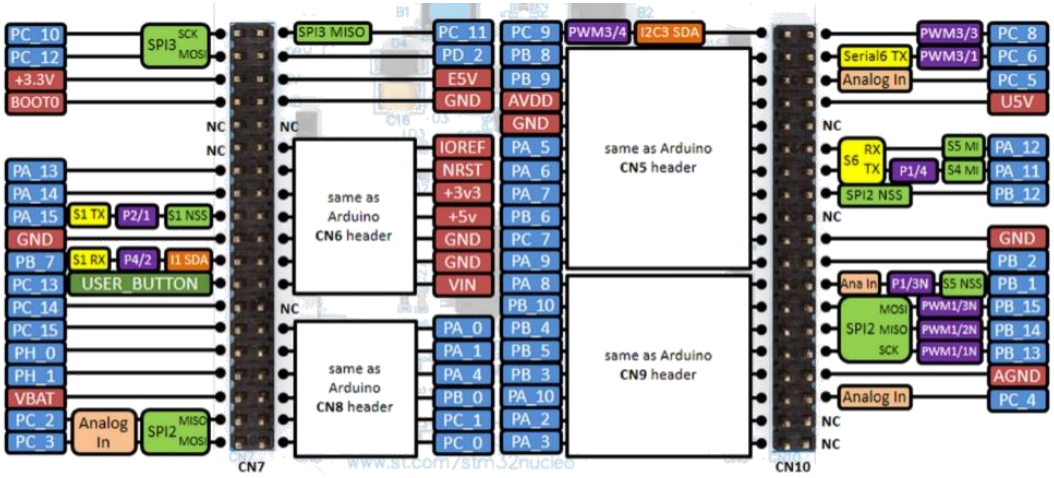


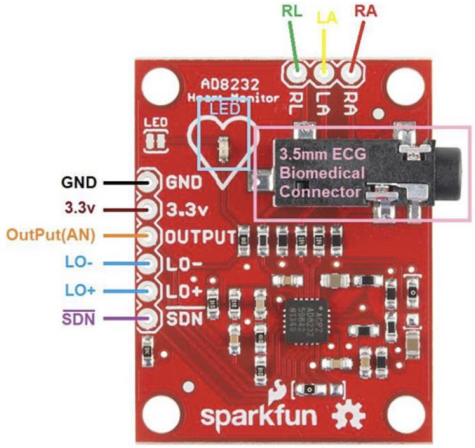
심전도 측정 회로

AD8232

- 심전도 측정 회로의 Output pin을 MCU ADC핀에 연결 (PC0)
- 심전도 측정 회로의 3.3v, GND를 MCU와 연결







실습: HW

- HW 구성: USB, 심전도 측정 회로
- FW 구성: 프로젝트 파일 추가
  - 폴더: STM32F4xx_DSP_StdPeriph_Lib_V1.8.0₩Libraries₩
    STM32F4xx_StdPeriph_Driver₩src
    파일: stm32f4xx_usart.c, stm32f4xx_adc.c , misc.c, stm32f4xx_tim.c,
    stm32f4xx_syscfg.c

실습: Firmware

- **Clock 변경 (→ 16MHz)**

  system_stm32f4xx.c 파일 수정

```
void SystemInit(void)
{
  /* FPU settings ------------------------------------------------------------*/
  #if (__FPU_PRESENT == 1) && (__FPU_USED == 1)
    SCB->CPACR |= ((3UL << 10*2)|(3UL << 11*2));  /* set CP10 and CP11 Full Access */
  #endif
  /* Reset the RCC clock configuration to the default reset state ------------*/
  /* Set HSION bit */
  RCC->CR |= (uint32_t)0x00000001;

  /* Reset CFGR register */
  RCC->CFGR = 0x00000000;

  /* Reset HSEON, CSSON and PLLON bits */
  RCC->CR &= (uint32_t)0xFEF6FFFF;

  /* Reset PLLCFGR register */
  RCC->PLLCFGR = 0x24003010;

  /* Reset HSEBYP bit */
  RCC->CR &= (uint32_t)0xFFFBFFFF;

  /* Disable all interrupts */
  RCC->CIR = 0x00000000;

#if defined (DATA_IN_ExtSRAM) || defined (DATA_IN_ExtSDRAM)
  SystemInit_ExtMemCtl();
#endif /* DATA_IN_ExtSRAM || DATA_IN_ExtSDRAM */

  /* Configure the System clock source, PLL Multiplier and Divider factors,
     AHB/APBx prescalers and Flash settings ----------------------------------*/
  SetSysClock();

  /* Configure the Vector Table location add offset address ------------------*/
#ifdef VECT_TAB_SRAM
  SCB->VTOR = SRAM_BASE | VECT_TAB_OFFSET; /* Vector Table Relocation in Internal SRAM */
#else
  SCB->VTOR = FLASH_BASE | VECT_TAB_OFFSET; /* Vector Table Relocation in Internal FLASH */
#endif
}
```

주석처리

실습: Firmware

```
#include "stm32f4xx.h"

uint16_t ADC1_data = 0;
uint16_t test_send = 0b0000111100001111;
uint8_t u16_2_8 = 0;
uint16_t send_count = 0;
uint8_t GetADC_Array[20];

void Delay(__IO uint32_t nCount)
{
  for(; nCount != 0; nCount--);
}

void LED_init(void)
{
   RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);
   GPIO_InitTypeDef GPIO_InitStructure;
   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
   GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
   GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
   GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
   GPIO_Init(GPIOA, &GPIO_InitStructure);
}
```

실습: Firmware

```
void Serial_Send(unsigned char t)
{
  while(USART_GetFlagStatus(USART2, USART_FLAG_TXE) == RESET);
  USART_SendData(USART2, t);
}

unsigned char Serial_Receive(void)
{
  unsigned char data;

  while(USART_GetFlagStatus(USART2, USART_FLAG_RXNE) == RESET);
  data = USART_ReceiveData(USART2);
  return data;
}
```

실습: Firmware

```
void TIM2_Configuration(int intervalms)
{
 RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);

 TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
 NVIC_InitTypeDef NVIC_InitStructure;

 TIM_TimeBaseStructure.TIM_Prescaler = 16000 - 1;
 TIM_TimeBaseStructure.TIM_Period = intervalms - 1;

 TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
 TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);


 TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
 TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE);
 TIM_Cmd(TIM2, ENABLE);

 NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);
 NVIC_InitStructure.NVIC_IRQChannel = TIM2_IRQn;
 NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
 NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
 NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
 NVIC_Init(&NVIC_InitStructure);
}
```

클럭 16MHz

실습: Firmware

```
void USART2_Configuration(void)
{
 RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);

 GPIO_InitTypeDef GPIO_InitStructure;
 GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
 GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
 GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
 GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2 | GPIO_Pin_3;
 GPIO_Init(GPIOA, &GPIO_InitStructure);

 GPIO_PinAFConfig(GPIOA, GPIO_PinSource2, GPIO_AF_USART2);
 GPIO_PinAFConfig(GPIOA, GPIO_PinSource3, GPIO_AF_USART2);

 RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART2, ENABLE);

 USART_InitTypeDef USART_InitStructure;
 USART_InitStructure.USART_BaudRate = 115200;
 USART_InitStructure.USART_WordLength = USART_WordLength_8b;
 USART_InitStructure.USART_StopBits = USART_StopBits_1;
 USART_InitStructure.USART_Parity = USART_Parity_No;
 USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
 USART_Init(USART2, &USART_InitStructure);
 USART_Cmd(USART2, ENABLE);
}
```

8bit

실습: Firmware

```
void ADC_Configuration(void)
{
  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC, ENABLE);

  GPIO_InitTypeDef GPIO_InitStructure;
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AN;
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
  GPIO_Init(GPIOC, &GPIO_InitStructure);

  NVIC_InitTypeDef NVIC_InitStructure;
  NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);

  NVIC_InitStructure.NVIC_IRQChannel = ADC_IRQn;
  NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x01;
  NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x02;
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStructure);
```

Interrupt

실습: Firmware

```
RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);

ADC_CommonInitTypeDef ADC_CommonInitStructure;

ADC_CommonInitStructure.ADC_Mode = ADC_Mode_Independent;
ADC_CommonInitStructure.ADC_DMAAccessMode = ADC_DMAAccessMode_Disabled;
ADC_CommonInitStructure.ADC_Prescaler = ADC_Prescaler_Div8; // 조정해서 쓰시오

ADC_CommonInit(&ADC_CommonInitStructure);

ADC_InitTypeDef ADC_InitStructure;
ADC_InitStructure.ADC_Resolution = ADC_Resolution_12b;
ADC_InitStructure.ADC_ScanConvMode = DISABLE;
ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
ADC_InitStructure.ADC_ExternalTrigConvEdge =   ADC_ExternalTrigConvEdge_None;
ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_T1_CC1;
ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
ADC_InitStructure.ADC_NbrOfConversion = 1;
ADC_Init(ADC1, &ADC_InitStructure);

ADC_RegularChannelConfig(ADC1, ADC_Channel_10, 1, ADC_SampleTime_3Cycles);
ADC_ITConfig(ADC1, ADC_IT_EOC, ENABLE);
ADC_Cmd(ADC1, ENABLE);
}
```

ADC 속도조절

12bit 받기

실습: Firmware

```
void ADC_IRQHandler(void)
{
  if (ADC_GetITStatus(ADC1, ADC_IT_EOC) != RESET)
  {
    ADC_ClearITPendingBit(ADC1, ADC_IT_EOC);
    ADC1_data = ADC_GetConversionValue(ADC1);
  }
}
```

ADC된 값을 읽어옴
ADC1_data는 uint16형임

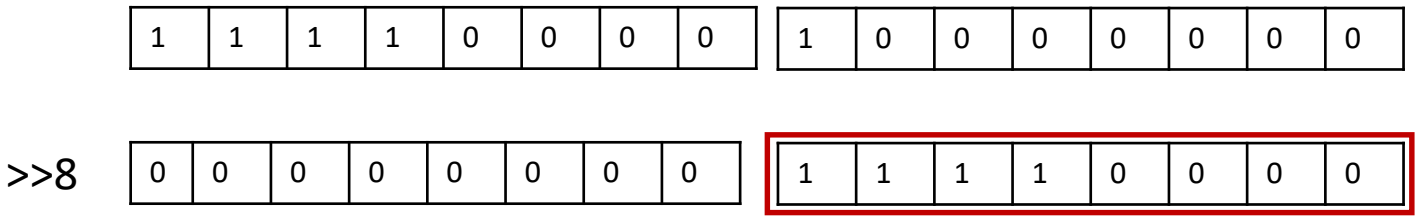
실습: Firmware

- ADC된 값을 USART를 통해 송신해야 함
- ADC는 12비트로 하였고, 16비트 변수에 저장되어 있음: ADC1_data
- USART는 8비트씩 전송함: Serial_Send(unsigned char t)
- 문제 1
  - Serial_Send(ADC1_data) 라고 구현하면 typecast되어 일부만 전송될 것
  - 2바이트로 나누어 보내야 함
- 문제 2
  - PC로 2바이트가 전달될 것임. 2바이트를 하나의 값으로 변환해야 함
  - 그런데, PC와 MCU가 커넥션되는 시점에 따라
    - 상위 바이트, 하위 바이트, 상위 바이트, 하위 바이트, … 가 넘어올 수도 있고
    - 하위 바이트, 상위 바이트, 하위 바이트, 상위 바이트, …가 넘어올 수도 있음

순서가능 ㄴㄴ

실습: Firmware

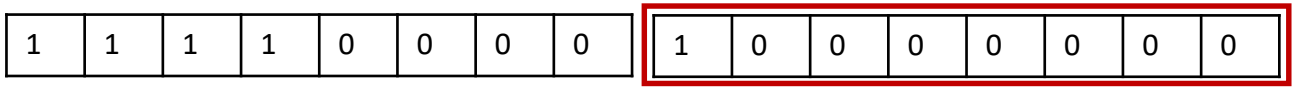
문제 1의 해결




```
u16_2_8 = (uint8_t) (ADC1_data>>8);
Serial_Send(u16_2_8);
```

16비트 변수를 8자리 shift시키고, 8비트로 typecasting

상위 8비트를 보냄




```
u16_2_8 = (uint8_t) (ADC1_data);
Serial_Send(u16_2_8);
```

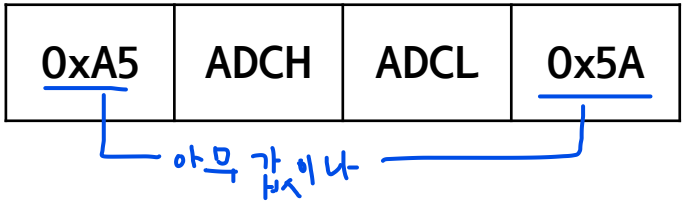
16비트 변수를 8비트로 typecasting → 하위 8비트만 남음

하위 8비트를 보냄

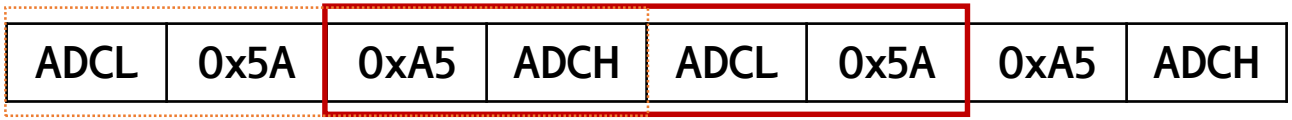
실습: Firmware

문제 2의 해결

- 2바이트 데이터를 패킷에 싸서 보냄
- 0xA5, , , 0x5A가 나올 가능성은 낮음



- PC에서는 stream으로 데이터가 들어오게 됨
- 1번째 값이 0xA5이고 4번째 값이 0x5A이면, 두 번째, 세 번째 값을 읽어 계산



$$Getdata = ADCH * 2^8 + ADCL;$$

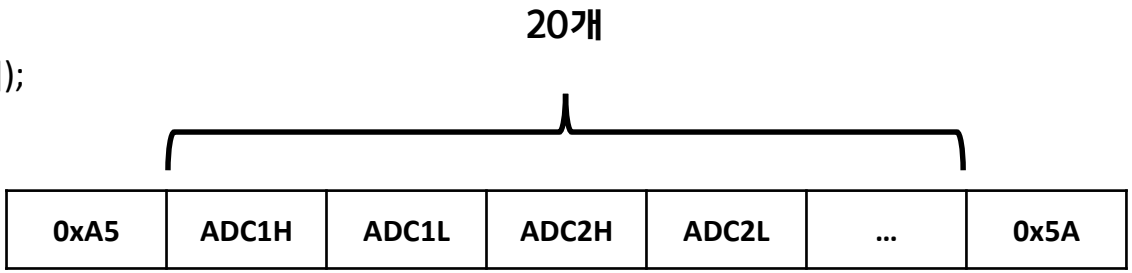
실습: Firmware

- ADC는 백그라운드에서 계속 수행하고 있음
- 초당 100개 (=100Hz=0.01초=10ms) 데이터를 수집해야 함
- 그리고 0.1초에 한번씩 데이터를 PC로 전송해야 함
  - 0.1초 동안 10개의 데이터가 수집될 것임
  - 0.1초에 한번씩 10개 데이터를 전송

- 0.01초 (=10ms) 마다 타이머 인터럽트를 발생시키고, 배열에 값을 저장해 둠
- 타이머 인터럽트가 10번 걸리면 (=0.1초), 데이터를 10번 전송

실습: Firmware

```
void TIM2_IRQHandler(void)
{
 if(TIM_GetITStatus(TIM2, TIM_IT_Update) != RESET)
 {
  GetADC_Array[send_count] = (uint8_t) (ADC1_data>>8);
  send_count++;
  GetADC_Array[send_count] = (uint8_t) (ADC1_data);
  send_count++;

  if (send_count == 20)
  {
   Serial_Send(0xA5);
   for(int i = 0; i < 20; i++)
   {
    Serial_Send(GetADC_Array[i]);
   }
   Serial_Send(0x5A);
   send_count = 0;
  }

// clear interrupt flag
  TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
 }
}
```



실습: Firmware

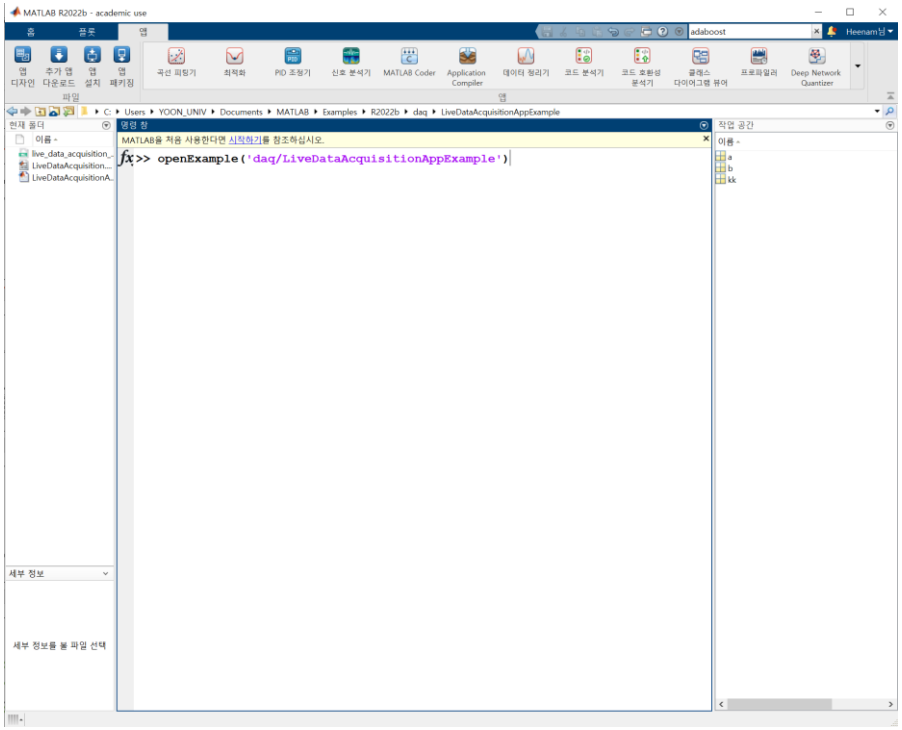
```
int main()
{
  LED_init();
  TIM2_Configuration(10);
  USART2_Configuration();

  ADC_Configuration();
  ADC_SoftwareStartConv(ADC1);

  unsigned char getdata;

  while(1)
  {
    getdata = Serial_Receive();

    if (getdata != 0)
    {
      GPIO_SetBits(GPIOA, GPIO_Pin_5);
      Delay(10000);
      GPIO_ResetBits(GPIOA, GPIO_Pin_5);
      Delay(10000);
    }
  }
  return 0;
}
```

실습: PC software

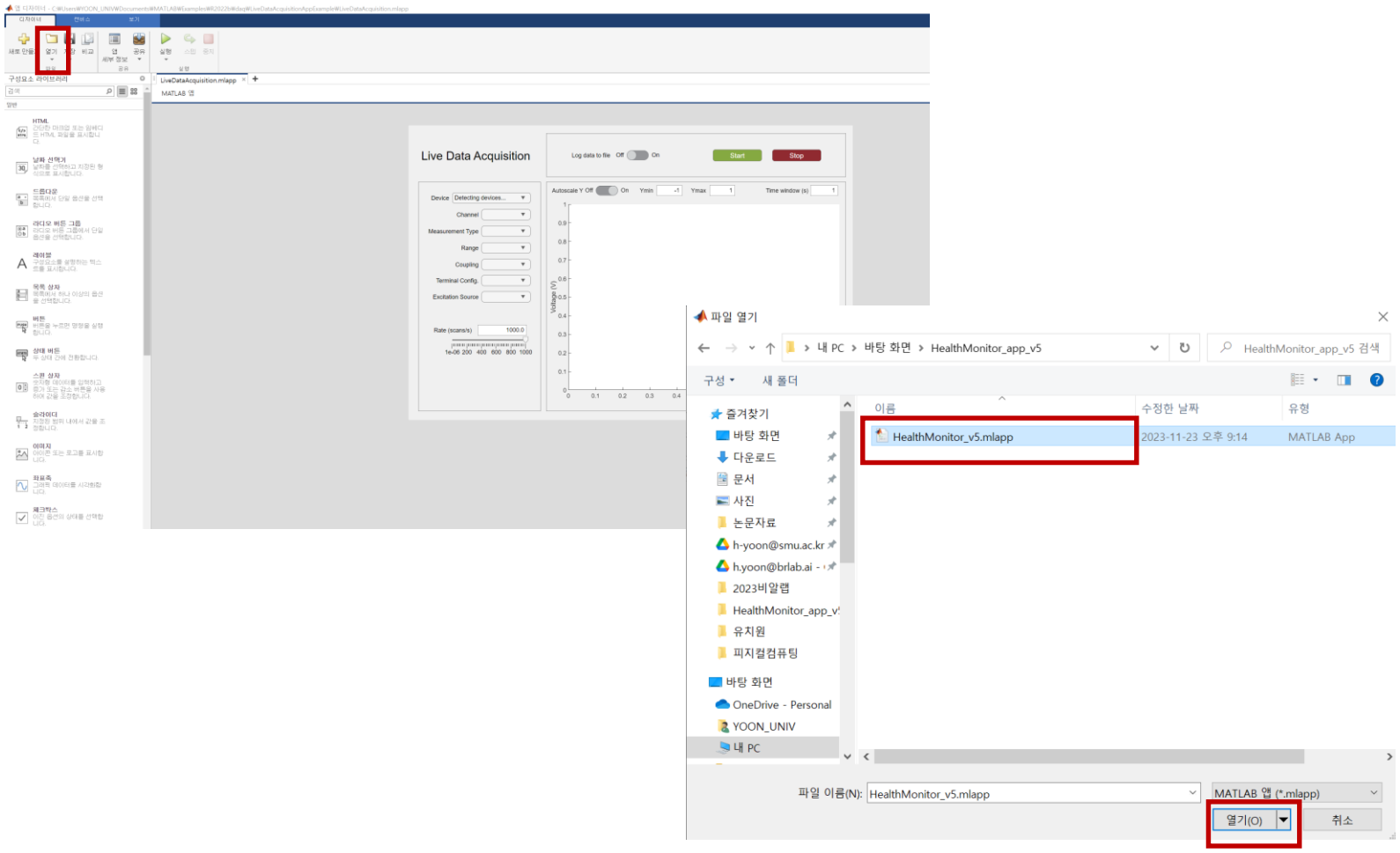
- 데이터를 수신하여, 그래프를 그리고, 심박수를 보여주는 프로그램
- 대부분 구현해 두었음

실습: PC software

- 매트랩 실행 후 아래와 같이 입력
- openExample('daq/LiveDataAcquisitionAppExample')

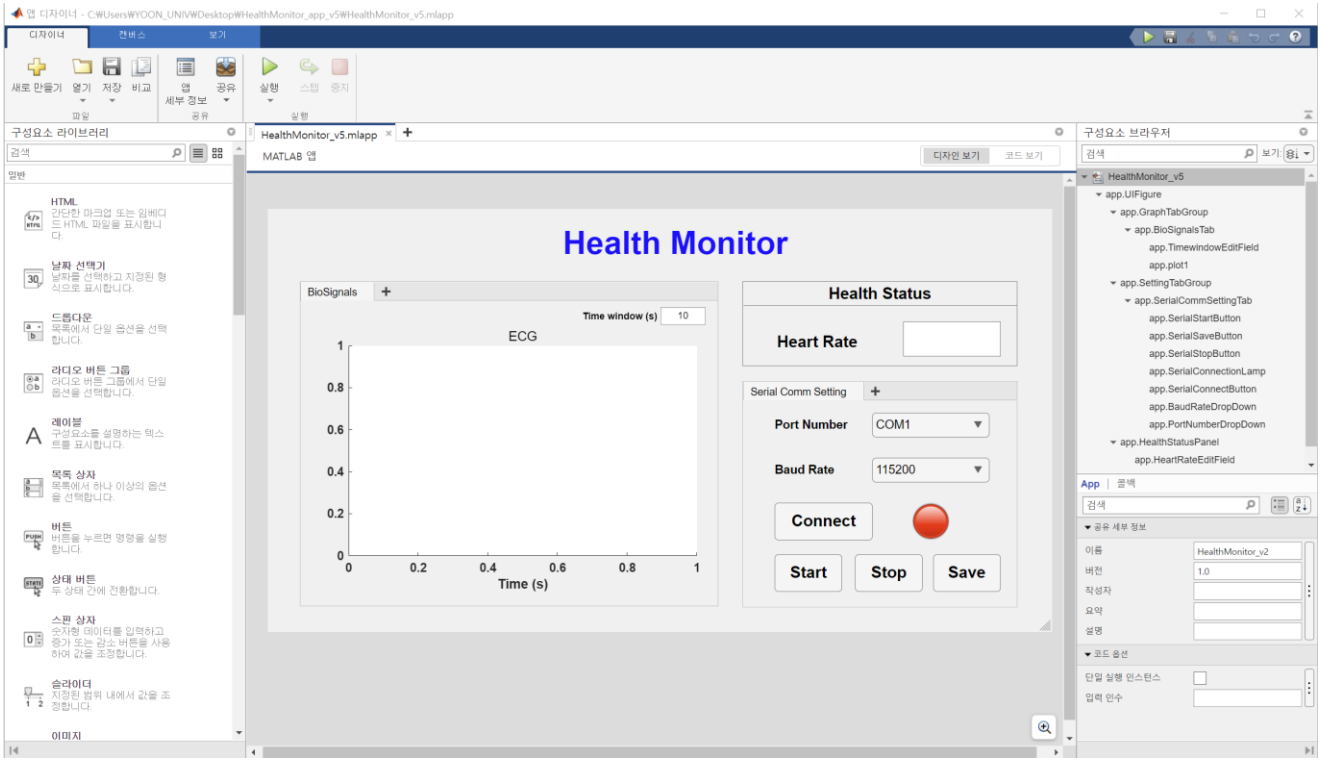


실습: PC software

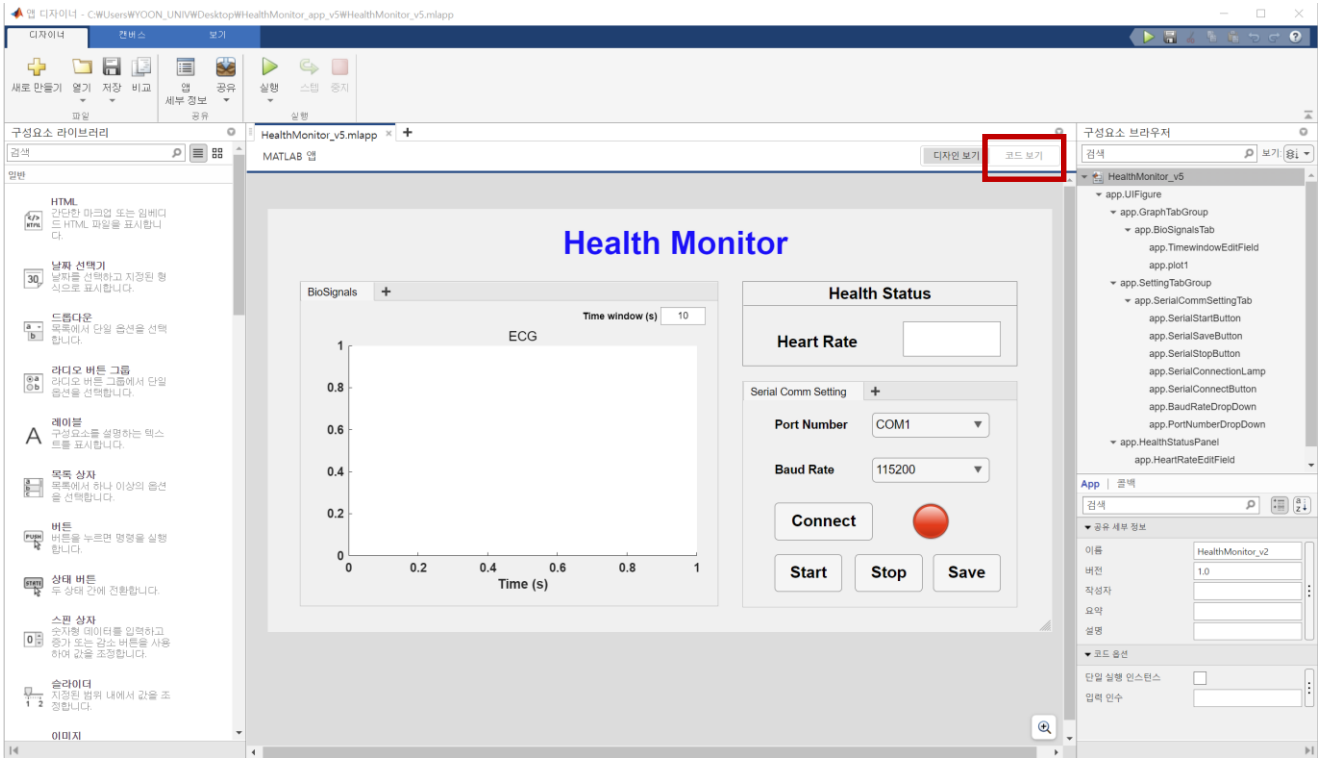


실습: PC software

- UI 메인 화면



실습: PC software



실습: PC software

- MCU에서 보낸 것을 처리

```
check_buffer = zeros(1, 22);

while(1)
    check_buffer(1:21) = check_buffer(2:22); % 시프트 시킴
    check_buffer(22)   = read(usb_nrf, 1, "uint8"); % 1비트씩 데이터를 읽어옴

    % 첫 값이 0xA5이고 22번째 값이 0x5A이면, 가운데 20개의 데이터를 가져와 계산
    if check_buffer(1) == 165 && check_buffer(22) == 90 % 165: 0xA5, 90: 0x5A
        outval1 = zeros(1, 10);
        cnt_t = 1;

        tmp = check_buffer(2:21);

        for kk=1:2:20
            t_high = tmp(kk);
            t_low = tmp(kk+1);
            t1 = int16(bitshift(int16(t_high), 8) + t_low);

            outval1(cnt_t) = t1;
            cnt_t = cnt_t + 1;
        end
        app.ECG = [app.ECG; outval1(:)];
        updateplot(app);
    end
end
```

실습: PC software

- **MCU로 값을 보냄**

```
%%%%%%%%%%%%%%%%%%%%%%%% HR extraction %%%%%%%%%%%%%%%%%%%%%%%%%%%
function f_value = CAL_HR_MERGED(app, data)
    f_value = zeros(app.Data.fs/10, 1);

    for kk=1:1:length(data)
        app.cnt = app.cnt + 1;

        [SMBCG, f_value(kk), app.dX_h_1, app.dY_h_1, app.dX_l_30, app.dY_l_30, app.Window1, app.Window2, app.Window3, app.pre_value] =  ...
            ECR_ECG_PREPROCESSING(data(kk), app.Window1, app.Window2, app.Window3, app.b_h_b, app.a_h_b, app.b_l_b, app.a_l_b, ...
            app.dX_h_1, app.dY_h_1, app.dX_l_30, app.dY_l_30, app.len1, app.len2, app.len3, app.pre_value);

        % 피크 검출
        app.PEAK_BUFF(1)   = app.PEAK_BUFF(2);
        app.PEAK_BUFF(2)   = app.PEAK_BUFF(3);
        app.PEAK_BUFF(3)   = SMBCG;

        % 피크가 검출되면,
        if app.cnt > 2*app.Data.fs
            if app.PEAK_BUFF(3) < app.PEAK_BUFF(2) && app.PEAK_BUFF(2) > app.PEAK_BUFF(1)
                app.PEAK_LOC        =  [app.PEAK_LOC (app.cnt-1)];
                app.PEAK_INTV       =  [app.PEAK_INTV (app.PEAK_LOC(end)-app.PEAK_LOC(end-1))/app.Data.fs];

                tmp = (app.PEAK_LOC(end)-app.PEAK_LOC(end-1))/app.Data.fs;
                tmp2 =    60./(mean(tmp));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                % MCU로 어떤 값을 보냄
                tmp3 = cast(tmp2, 'uint8');
                write(app.usb_nrf, tmp3, 'uint8');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            else
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                write(app.usb_nrf, 0, 'uint8');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

프로젝트 가이드

- MCU clock 주파수 16MHz
- Port, Pin 사용: 자유

	참고	배점
심전도 ADC, USART RX/TX, UI그래프 출력	ADC 100Hz	40
심박수 값을 수신하여 FND 출력	소수점 없어도 됨	10
평균 심박수 값의 3구간으로 나누어 RGB-LED 색깔 제어 (예. 60 ~ 80: G, 80~100: B, 100~120: R)	구간 자율	15
수신한 심박수 값만큼 서보모터 각도 제어	-	20
User switch 누르면 RGB-LED 밝기 조절	인터럽트로 구현	15
기타 아이디어 구현 (MAX 2개까지)	PC software도 가능	+15

프로젝트 가이드

- 방법: 동작여부 확인
- 발표 준비가 되면, 개별적으로 확인
- 12월 4일부터 발표 가능
- 마지막 발표는 12월 18일 15시까지
- 발표 기회는 2회
- 발표가 완료되면
  - 이캠퍼스에 main.c 업로드
  - 실습 재료 반납

- First come, first served! (+15 ~ 0, additional points)

Thank you.