반도체 캡스톤설계 소스코드 분석

2017313107 이승태

1. bluetooth_naiming

Rx에 0번 Tx에 1번을 세팅해주고, Setup에서 bluetooth와 Serial통신을 시작한다. loop를 돌면서 bluetooth나 Serial이 통신가능할 때 시리얼 모니터에 bluetooth로 받은 데이터를 출력하고, bluetooth로 시리얼 모니터에서 보낸 데이터를 출력한다.

bluetooth_naming §

```
#include <SoftwareSerial.h>           //serial 통신 헤더파일 선언
SoftwareSerial bluetooth(0, 1);       //bluetooth pin 선언
void setup()
{
  Serial.begin(9600);                 //serial monitor 통신 시작
  bluetooth.begin(9600);              //bluetooth 통신 시작
}
void loop() {
  if (bluetooth.available())          //serial monitor에 bluetooth로
     Serial.write(bluetooth.read());  //받은 데이터 출력

  if (Serial.available())             //bluetooth로 serial monitor에서
     bluetooth.write(Serial.read());  //받은 데이터 출력
}
```

2. capstone_practice

(1) 세팅을 위해 RXD, TXD를 각각 0,1번으로 설정해주고, readPin0(IR 센서에 관여)를 아날로그 핀인 A0에, ledpinA(LEDA에 관여)를 디지털 핀인 12번에, ledpinB(LEDB에 관여)를 디지털핀인 13번에 할당해준다.

```
#define BT_RXD 0
#define BT_TXD 1
SoftwareSerial bluetooth(BT_RXD, BT_TXD); //bluetooth serial communication enable
int readPin0 = A0; // the number of IR sensor pin
int ledpinA = 12;  // the number of LEDA Pin
int ledpinB = 13;  // the number of LEDB pin
```

(2) setup함수에서 IR핀은 input으로 값을 받고, ledpin들은 빛을 내서 결과값을 보여주므로 output이 되고, ledpin을 모두 끈상태로 유지하며 통신속도는 9600보드라이트로 설정해준다.

```
void setup(){
  pinMode(readPin0,INPUT); // setting pin mode
  pinMode(ledpinA, OUTPUT);
  pinMode(ledpinB, OUTPUT);
  digitalWrite(ledpinA, LOW); // turn off led
  digitalWrite(ledpinB, LOW);
  Serial.begin(9600);     //serial monitor communication begin
  bluetooth.begin(9600);  //bluetooth commnunication begin
}
```

(3) loop안의 함수에서, 데이터를 읽고 데이터가 들어오지 않았다면 전의 값을 그대로 유지한다.

```
void loop(){
 if (bluetooth.available()) {

    data_in = bluetooth.read();   //saving data from bluetooth
 }
 else{
  data_in = data_in;
 }
```

(4) loop내의 switch문은 입력으로 받은 또는 이전의 데이터에 따라 동작을 달리한다.
- A, a가 들어오면 ledA를 각각 켜고 끈다.
- B, b가 들어오면 ledB를 각각 켜고 끈다.
- C, c가 들어오면 ir센서를 각각 enable, disable한다.
- T는 ledB에 대한 timer의 값을 토대로 ledB를 킨다.
- f, t는 각각 timer를 500, 1000으로 설정해준다.

```
switch(data_in){

   case 'A' :            // 'A' means that the BUTTON_ONA is touched.
   {
     digitalWrite(ledpinA, HIGH);
     break;
   }
   case 'a' :            // 'a' means that the BUTTON_OFFA is touched
   {
     digitalWrite(ledpinA, LOW);
     break;
   }
    case 'B' :           // 'B' means that the BUTTON_ONB is touch down.
   {
     digitalWrite(ledpinB, HIGH);
     break;
   }
    case 'b' :           // 'b' means that the BUTTON_ONB is touch up.
   {
     digitalWrite(ledpinB, LOW);
     break;
   }
   case 'C' :            // 'C' means that the BUTTON_ENABLE is touched.
   {
     ir_enable = 1;
     break;
   }
   case 'c' :            // 'c' means that the BUTTON_DISABLE is touched.
   {
     ir_enable = 0;
     break;
   }
   case 'T' :            // 'T' means that the BUTTON_ON_timer is touched.
   {
     t_enable = 1;
     data_in = 'n';      // to avoid lnfinite loop. 'n' means nothing.
     break;
   }
    case 'f' :           // 'f' means that state A is selected at spinner.
   {                     // (which does not means BUTTON_ONA is touched)
     timer = 500;        // setting five seconds
     break;
   }
    case 't' :           // 't' means that state B is selected at spinner.
   {                     // (which does not means BUTTON_ONB is touched)
     timer = 1000;       // setting ten seconds
     break;
   }
   default :
   {
     break;
   }
```

(5) 위에서 T값이 들어와 t_enable이 1이 되었을 때 동작이다.
timer의 값만큼 led의 불을 킨다. 먼저 if문에 들어올 때, ledB의 불을 켜주고, 나갈 때 t_enable을 0으로 바꾸어 주면서, ledB의 불을 꺼준다.
가운데의 코드는 for문을 timer만큼 돌면서 j가 1이 증가할 때 마다 delay(10) (약 10ms)을 발생시킨다.
j값이 100의 배수인 경우, if(mod == 0)에 들어오고, 남은시간(초)를 float -> char으로 바꾸어 bluetooth에 보내어 출력하게 된다.

```
//-------------------------LED timer loop-------------------------
  if(t_enable == 1){
    digitalWrite(ledpinB, HIGH);
    for(j = 0;j <= timer;j++)               // repeat counting
    {
       mod = j%100;
       if(mod == 0){
       left_time = (timer -j)/100;
       dtostrf(left_time , 1, 0, data_out); // conversion variable
                                            // (float -> char)
       bluetooth.write(data_out);           // sending left time to bluetooth
       }
       delay(10);
    }

    digitalWrite(ledpinB, LOW);
    t_enable = 0;
    j =0;
  }
```

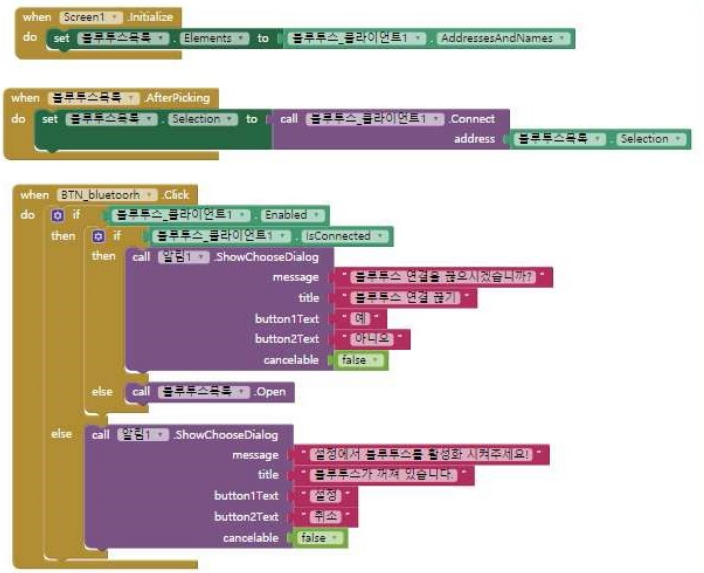
(6) C를 입력하여, ir_enable이 되었을 때 들어오는 if문이다.
간략하게 설명하면, num만큼 돌면서 거리를 측정하고 평균값을 내어 출력한다.
num은 200이고 for문을 돌면서 delay(10) (10ms)를 주기 때문에, 2초간 for문을 돈다.
total에 거리 di를 1번씩 더해주므로 2초간 200번 total에 더해진 값이 저장되는 것을 알 수 있다.
만약 중간에 값이 c로 들어온다면, ir_enable을 0으로 바꾸고 for문을 빠져나오는 것을 알 수 있다.
avaerage는 2초간 평균거리를 계산해주며, string으로 바꾸어주고, write를 통해(중간에 'c'를 입력해준 경우 출력되지 않음) 그 값을 써주게 된다.

```
//-------------------------IR sesor loop-------------------------
  if(ir_enable == 1){
    for(i = 0;i <= num;i++)                      // repeat sensing
    {
       float v = analogRead(readPin0)*(5.0/1023.0); // modifing sensing voltage
       float di = 60.495*pow(v,-1.1904);         // calculating distance
       total = total + di;
       delay(10);
       if (bluetooth.available())
       {
           if (bluetooth.read() == 'c'){
              data_in = 'c';
              ir_enable = 0;
              break;                             //disable IR sesing
           }
       }
    }

    average = (int)total/num;                    // averaging sensing value

    dtostrf(average , 5, 0, data_out);           // conversion variable
                                                 //  (float -> char)

    if(ir_enable == 1)
    {
      bluetooth.write(data_out);                 // Sending IR Sensor data to bluetooth
    }
    if( i >= num){
       i = 0;
       total = 0;
    }
  }
```

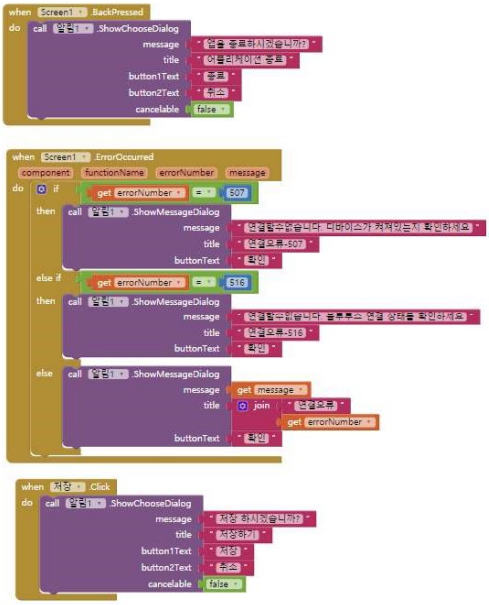
3. CAPSTONE_PRACTICE.aia
앱인벤터는 사용자의 편의를 위해 java나 kotlin, xml을 사용하는 대신 block coding을 이용하여 더 직관적인 코딩이 가능하게 만들어졌다.

(1) 블루투스를 연결하는 블락코딩부분이다. 먼저 화면이 켜질 때 블루투스 목록들을 받아와 블루투스 목록이라는 곳에 이름과 주소를 저장한다.
오른쪽 아래의 블루투스 버튼을 클릭하게 되면, 블루투스가 꺼져있는 경우 활성화 시키기 위

해 알림을 생성한다. 그렇지 않다면, 현재 연결된 장치가 있는지 확인하고, 연결되어있다면 끊을건지 물어보는 알람을, 그렇지 않다면 블루투스의 목록을 불러와서 어떤 장치에 연결할 건지 선택하게 한다.

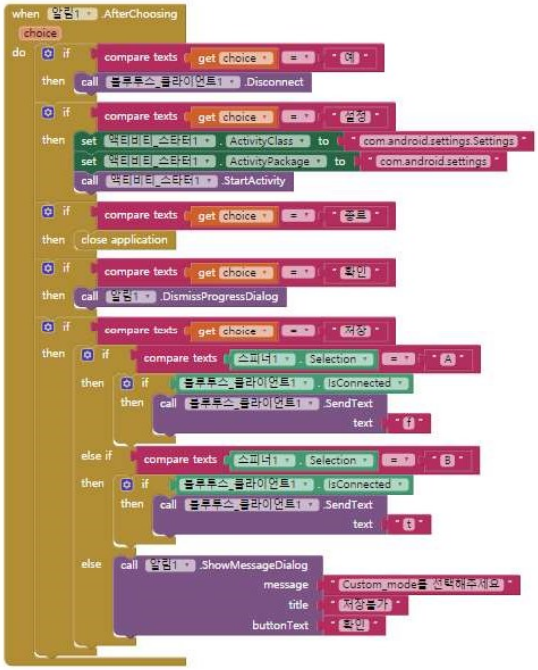


(2) 화면에서 back버튼을 눌렀을 때 알람이 뜨면서 종료할건지 물어본다.
가운데 블록코딩은 에러처리에 관한 내용으로, 디바이스의 현재 상태나 블루투스의 연결 등에 따라 연결되지 못했을 경우 에러 메시지를 띄워준다.
저장버튼을 누르게 되면 현재 상태를 저장할건지 물어보는 창이 뜬다.



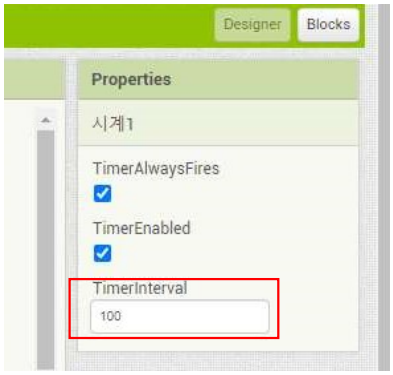
(3) 다음은 버튼에서 다음과 “예”, “설정”, “종료”, “확인”, “저장”이 선택되었을 때의 protocol을 나타냅니다. 각각 블루투스 끊기, android 설정창 열기, 앱종료하기, progress

dialog띄우기, 클라이언트가 연결되어있을 때, 스피너 값이 각각 A, B로 선택 되어있었다면, f,t값을 보내준다. A, B가 아닌 경우, 맨 아래와 같이 저장불가dialog를 띄우게 된다.



(4) timer에 관련된 코드이다.

timer는 일정시간마다 작동하는 코드이며, 코드에서는 100ms마다 작동하는 것을 알 수 있다.

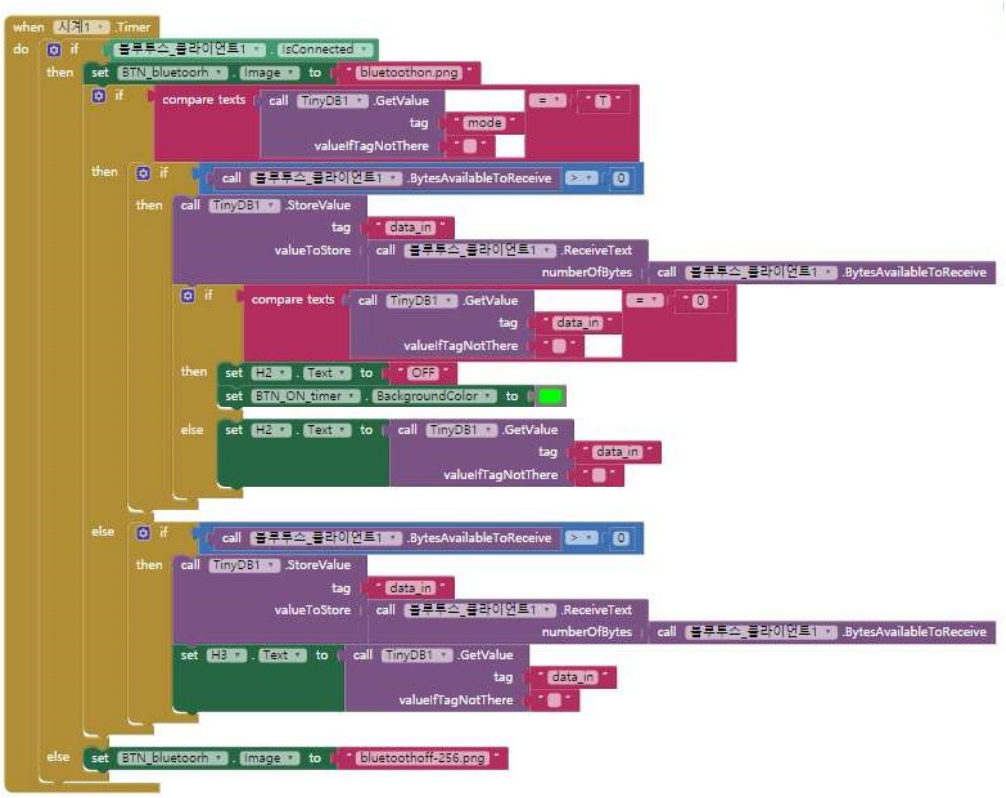


먼저 블루투스가 연결되었다면, 블루투스가 켜진 이미지(파란 블루투스 이미지)를 보여주고, 그렇지 않다면 꺼진 이미지(회색 블루투스 이미지)를 보여준다.

그 다음 내부 데이터에 mode T가 쓰여있는지 확인한다.

mode T가 쓰여있고, 데이터를 받은 것이 있다면, (LEDB를 켜고 있는 경우) 받은 값을 data_in에 저장하고, 만약 그 값이 0이라면, LEDB가 꺼진 경우를 나타내므로, ON_timer의 버튼색을 초록색으로 바꾸고, 그렇지 않다면, H2의 블록 값을 받은 data_in으로 표시하게 된다.

mode T가 쓰여있지 않고, 데이터를 받은 것이 있다면, 이 코드에서는 mode R이 있다는 이야기가 된다. 그러므로, 받은 값을 data_in에 저장한 후, 블록 H3에 값을 나타내준다.



(5) switch문과 비슷한 동작을 하는 부분으로 버튼을 눌렀을 때 블루투스 클라이언트에게 값을 보내주고, 버튼색이나 박스안의 값을 ON, OFF로 바꾸어준다.

- 버튼 ONA를 누르거나 OFFA를 눌렀을 경우, 블루투스가 연결되어있는지 확인하고, 각각 A, a를 클라이언트에게 보내고, H1의 박스 안의 텍스트를 ON, OFF로 설정해준다. 버튼색은 각각 노란색, 초록색이다.
- 버튼 B를 누르고 있거나 누르지 않는 경우, 블루투스가 연결되어있는지 확인하고, 각각 B, b를 클라이언트에게 보내고, H2의 박스 안의 텍스트를 ON, OFF로 설정해준다. 버튼색은 빨간색, 노란색으로 바뀌게 된다.
- 버튼 ENABLE, DISABLE을 각각 눌렀을 경우, 블루투스가 연결되어있는지 확인하고, 클라이언트에게 C, c를 보내고, 색깔은 각각 ENABLE되었을 경우 빨간색, 노란색으로 DISABLE되었을 경우 초록색 빨간색으로 설정해준다. 그 후 내부 데이터베이스에 mode R을 기록한다.
- 버튼 ON_timer가 눌리게 될 경우, 블루투스가 연결되어있는지 확인하고, T를 보낸다. 버튼을 빨간색으로 설정하고 내부 데이터베이스에 mode T를 기록한다. (위에서 쓰임)

스피너에 A, B가 선택되었을 때, 블루투스가 연결되어있는지 확인하고, 클라이언트에 f, t를 보내주는 코드이다.