『졸업 프로젝트 보고서』

작품 명 : 오리 자전거

걸어서 십분 팀

이진선
이유진

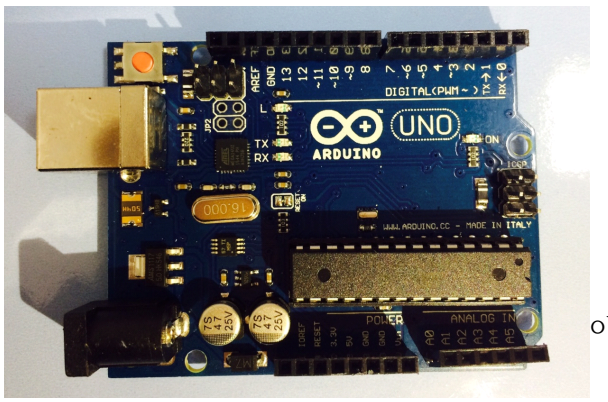
<목차>

1. 작품 개요

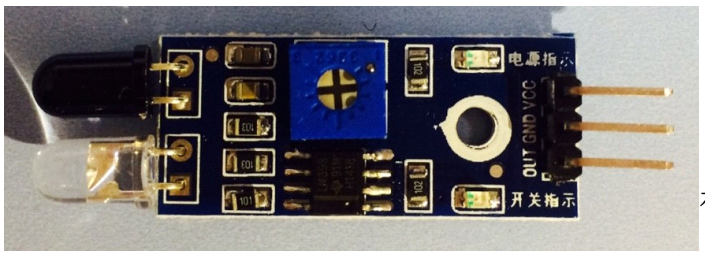
요즘의 바쁜 현대인들에게 필요한 것은 몸과 마음의 '힐링'입니다. 어떤 경험을 통해 힐링을 할 수 있을까라고 생각을 해봤을 때 바로 떠오른 것은 여행이었습니다. 그래서 실내에서 고정된 자전거를 타고 가상의 공간을 여행하는 프로그램을 만들면 좋을 것 같다는 생각을 하게 되었고 유니티, 아두이노, 센서 등을 활용하여 구현하였습니다. 저희 프로그램은 회원 관리 폼, 맵 선택, 게임 화면으로 구성되어 있습니다.

2. 작품 관련 사진

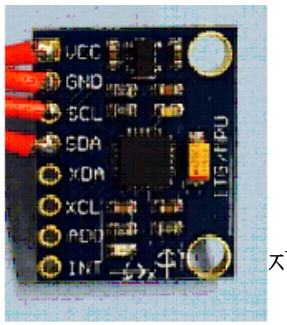
<재료 사진>



아두이노우노



적외선 감지 센서



자이로 센서

<전시 사진>

3. 구현 코드 및 설명

아두이노 스케치 코드부터 설명한 뒤에 유니티 소스코드를 설명하도록 하겠습니다. 소스코드 중 중요한 부분을 골라 작성했습니다. 먼저 적외선 감지 센서를 다루는 부분을 설명하도록 하겠습니다.

3.1 아두이노 스케치 코드

자전거의 실제 속력을 캐릭터에 적용하기 위해 아래의 코드를 작성하였으며, 저희는 속도 = 거리/시간 공식을 사용했습니다. 자전거 바퀴가 한 바퀴 도는 데 걸리는 시간으로 속도를 구했습니다. 그래서 거리 2πr을 구하기 위해 먼저 바퀴의 지름을 쟀고, 그 다음으로는 속도를 구하기 위해 적외선 감지 센서를 활용했습니다.

자전거 바퀴살에 먼저 적외선 감지 센서를 부착하고, 자전거의 고정된 부분에 한군데 돌출부위를 만듭니다. 그 돌출부위에 돌아가던 적외선 감지 센서가 감지되면 1이 출력 되도록 코드를 만들었고, 0에서 1로 바뀌는 사이의 시간을 재서 공식에 들어갈 시간을 구했습니다.

적외선 감지 센서 부분 코드
- loop 함수의 일부입니다.

```
if (digitalRead(IR) == LOW && Flag == false) {
        Serial.println(1);
        Flag = true;
} else if (digitalRead(IR) == HIGH && Flag == true) {
        Serial.println(0);
        Flag = false;
}
delay(50);
```

if 적외선 감지 센서에 물체가 감지 됨 -> println(1);
감지되지 않음 -> println(0);

자이로 센서 부분 코드

- loop 함수의 일부입니다. (팀원의 코드)

```
/* 앞선 적외선 감지 센서로 자전거의 속력을 구했다면,
이번에는 자이로 센서를 활용해 캐릭터의 방향(좌, 우)을 가져오기 위해 아래의 코드를 작성했습니다. */

error = MPU6050_read (MPU6050_ACCEL_XOUT_H, (uint8_t *) &accel_t_gyro, sizeof(accel_t_gyro));
센서로부터 gyro, accel 값을 가져옵니다.

    gx2 = angleInDegrees(lowX, highX, accel_t_gyro.value.x_gyro);
    gy2 = angleInDegrees(lowY, highY, accel_t_gyro.value.y_gyro);
    gz2 = angleInDegrees(lowZ, highZ, accel_t_gyro.value.z_gyro);

    predict(&angX, gx2, loopTime);
    predict(&angY, gy2, loopTime);
    predict(&angZ, gz2, loopTime);

    gx1 = update(&angX, accel_t_gyro.value.x_accel) / 10;
    gy1 = update(&angY, accel_t_gyro.value.y_accel) / 10;
    gz1 = update(&angZ, accel_t_gyro.value.z_accel) / 10;

//센서에서 읽어온 값에 칼만 필터를 적용합니다.

    if(initIndex < initSize) {
      xInit[initIndex] = gx1;
      yInit[initIndex] = gy1;
      zInit[initIndex] = gz1;
      if(initIndex == initSize - 1) {
        int sumX = 0; int sumY = 0; int sumZ = 0;
        for(int k=1; k <= initSize; k++) {
          sumX += xInit[k];
          sumY += yInit[k];
          sumZ += zInit[k];
        }

        xCal -= sumX/(initSize -1);
        yCal -= sumY/(initSize -1);
        zCal = (sumZ/(initSize -1) - zCal);
      }
      initIndex++;
    }
//최초 실행될 때 측정되는 n개의 값을 평균해서 저장하여 측정되는 값을 보정하는데 사용합니다.
```

이 코드를 통해 자전거에 붙인 자이로센서의 x, y, z 좌표를 얻어와 핸들의 움직임을 조정하는데 적용했습니다.

3.2 유니티 C# 코드

다음은 유니티 프로젝트에 쓰인 코드를 설명하도록 하겠습니다. 먼저 캐릭터의 움직임을 담당하는 소스코드입니다.

```
using UnityEngine;
using System.Collections;
using System.IO.Ports;
using System;

 [RequireComponent(typeof(NetworkView))]
public class PlayerAdvance : MonoBehaviour
{
    public static SerialPort sp = new SerialPort("COM6", 9600);
    float timer, speed, inter_timer;
    static float pi;
    static float radius;
    int sense;

    /* SerialPort형 변수 sp는 아두이노에서 읽어 들인
     * 적외선 감지 센서와 자이로 센서 값을 컴퓨터의
     * 시리얼 포트를 통해 읽어오고,
     * int형 변수 sense에 그 값을 저장합니다.
     * 나머지 변수 timer, inter_timer, pi, radius는
     * 자전의 속력을 구하기 위해 필요한 변수들이고
     * 아래의 코드에서 더 자세히 설명 드리겠습니다. */

    float Z = 0f;  //camera rotation value
    float Z0 = 100;
    float angle = 0.5f;

    /* 위의 변수들은 자이로 센서에서 받아온
     * 방향 값을 처리하기 위해 설정한 변수입니다.
     * Z0 값을 100으로 초기화 했는데
     * 이는 캐릭터가 정면을 보고 있을 때의 값으로,
     * 방향을 결정하는 기준 값으로 정했습니다. */

    int IR;
    /* 변수 IR은 아까 적외선 센서를 통해 구한
     * 0, 1 값을 담을 변수입니다. */

    NetworkView nView;

// Use this for initialization
void Start()
{
    nView = gameObject.GetComponent<NetworkView>();

    sp.Open();
    sp.ReadTimeout = 30;
    /* 포트를 열고 sp 변수의 ReadTimeout 속성을 정해줍니다.
     * ReadTimeout 속성은 읽기 작업을 마쳐야 하는 제한 시간(밀리초)을
     * 뜻하며 30 밀리초로 정해주었습니다. */


    timer = 2.0f;
    pi = 3.14f;

    radius = 0.3f;
    speed = 0.0f;

    /* 각 변수들을 초기화 시킵니다.
     * timer의 값은 후에 speed 변수의 분모가 될 것이므로
     * 0.0f로 초기화 하지 않았습니다.
     * radius는 직접 자전거 바퀴의 길이를 재서 나온 값(30cm)입니다. */

    IR = 3;
    /* IR 변수 값은 초기화할 때 캐릭터의 움직임과 상관없는 값으로 설정했습니다.
     * (0 혹은 1이 되면 플레이어는 전혀 조작하지 않았음에도 캐릭터가 움직이게 됩니다.) */
}
```

```
// Update is called once per frame
void Update()
{
    if (!nView.isMine)
        return;

    try
    {
        sense = int.Parse(sp.ReadLine());
        //아두이노에서 보내준 센서의 값을 한 줄씩 읽어옵니다.
    }
    catch (TimeoutException)
    {
    }

    if (sense == 0)
    {
        IR = 0;
    }

    if (sense == 1)
    {
        IR = 1;
    }
    /*
     * 만약 아두이노에서 읽어온 값이 0이면
     * (적외선 감지 센서에 아무것도 감지되지 않았음)
     * IR 변수의 값을 0으로 만들고 1이면
     * (적외선 감지 센서에 자전거 휠의 돌출된 부분이 감지 됨)
     * IR 변수의 값을 1로 만들어줍니다.
     * 원래 적외선 감지 센서 코드만 있었을 때에는 sense 값으로 모든 것을 해결했지만,
     * 자이로 센서와 코드를 합치면서 IR 변수를 만들게 되었습니다. */

    if (IR == 0)
    {
        timer += Time.deltaTime;
        /* 적외선 감지 센서에 아무것도 감지가 되지 않을 때,
         * Time.deltaTime을 계속 더해주어 한 바퀴 돌 때
         * 얼마큼의 시간이 걸렸는지를 측정합니다. */
    }


    if (IR == 1)
    {
        inter_timer = Time.deltaTime;
        timer = Mathf.Lerp(timer, inter_timer, 0.5f);

        IR = 0;
    }

    /*
     * 적외선 감지 센서에 물체가 감지 됐을 경우
     * timer의 값을 초기화 해주는데 위에서 말했던 것과 동일한 원리로
     * 분모에 0이 들어가면 안 되기 때문에 Time.deltaTime을 대신해서 넣어줍니다.
     * 그러나 실제로 실행을 했을 때 timer 변수에 time.deltaTime을 그대로 넣어주면
     * 캐릭터의 움직임이 부자연스럽게 갑자기 움직이는 듯한 현상이 나타나기 때문에
     * inter_timer라는 변수를 두었고 이전의 timer 값과 보간 함으로써
     * 부드러운 움직임을 표현하고자 했습니다.
     * 그리고 적외선 감지 센서에 물체가 감지되고 바로 IR 값을 0으로 만들어주어
     * 다음 바퀴가 도는 시간을 측정합니다.
     */

    speed = (2 * pi * radius) / (timer);
    /*
     * speed 변수는 각속도를 나타내는 변수입니다.
     * 방금 전에 구한 자전거가 한 바퀴 돌 때의 시간을 가지고
     * 자전거 바퀴의 실질적인 속도를 표현합니다.
     */

    if (IR == 0 || IR == 1)
    {
        transform.Translate(Vector3.forward * speed * Time.deltaTime * 7.5f);
    }
    /*
     * 이 부분은 시작과 동시에 캐릭터가 움직이는 것을 방지하기 위해
     * 적외선 감지 센서가 동작하고 있을 때만 캐릭터가 움직이도록 구현한 코드입니다.
     * (IR을 3으로 초기화 한 것도 마찬가지의 이유입니다.)
     */
```

```
    if (sense > 50) //변수 sense의 값이 500이 넘어가는 것은 자이로 센서의 영역입니다.
    {
        if (sense == Z0)
            transform.Rotate(0, 0, 0);
        //만약 자이로센서의 값이 Z0(정면을 보고 있다)라면 회전을 하지 않습니다.

        if (sense > Z0)
        {
            transform.Rotate(0, angle, 0);
        }

        else if (sense < Z0)
        {
            transform.Rotate(0, -angle, 0);
        }
        //Z0의 값이 100보다 작으면 왼쪽, 크면 오른쪽으로 회전하도록 작성한 코드입니다.
    }
}

void OnApplicationQuit()
{
    sp.Close();
}
//프로그램이 끝날 때 시리얼 포트를 닫아줍니다.
```

다음으로 회원관리 체계 소스코드입니다. 굳이 당장 필요한 기능은 아니었지만, 소스코드를 작성한 이유는 후에 프로그램이 더 다양한 기능을 갖췄을 때 회원 개개인의 데이터를 기반으로 확장 된 서비스를 제공 할 수 있는 여지가 있었기 때문입니다. 여기서 템플릿 메서드 패턴을 사용했습니다. 템플릿 메서드 패턴이란 부모 클래스에서 기능의 흐름을 정해놓고 자세한 구현은 자식 클래스에서 하는 것이기 때문에 부모 클래스에 작성한 내용만을 다루도록 하겠습니다.

```
using UnityEngine;
using UnityEngine.UI;
using System.Collections;

/*
* 템플릿 메소드 패턴 사용(부모 클래스)
*/
abstract public class SignUpField : MonoBehaviour {
    abstract protected string isEqualPwd();
    abstract protected void nullField();
    abstract protected void notMatch();
    abstract protected void error();
    abstract protected void afterWWW();
    abstract protected void onChangeName(string s);
    //회원가입&로그인 시 이름을 입력하는 Input Field에 바뀌는 내용 감지 및 변경
    abstract protected void onChangeID(string s);
    abstract protected void onChangePWD(string s);
    abstract protected void onChangeCFM(string s);
    //회원가입 체계에서 필요한 검사들을 하기 위한 메서드들의 정의입니다.

    protected WWW w;
    protected string url;

    // Use this for initialization (Empty)
    void Start() ...

    // Update is called once per frame (Empty)
    void Update()...

    /* 유니티 Scene에서
     * text field, input field, Button의
     * 객체를 찾아주는 역할을 합니다. */
    public T returnField<T>(T tmp, string fieldName)
    {
        tmp = GameObject.Find(fieldName).GetComponent<T>();

        return tmp;
    }
```

```
    protected string nullCheck(string[] arr, int NOF)
    {
        for (int i = 0; i < NOF; i++)
        {
            if (arr[i] == "" || arr[i] == null)
            {
                Debug.Log("null");
                return null;
            }
        }
        return "NOT NULL";
    }


    protected IEnumerator signUpAndLogIn(string[] jspVal, string[] unityVal, int nof)
    {
        /*
         * 템플릿 메서드입니다. 위에서 정의한 추상 메서드들로 처리의 흐름을 정해놓고
         * 실질적인 구현과 작동은 자식 클래스 SignUp.cs, LogIn.cs에서 당담하게 됩니다.
         * 회원가입과 로그인을 처리하는 과정이 비슷하다고 생각하여
         * 템플릿 메서드 패턴을 사용하게 되었습니다.
         * 리턴 형식은 IEnumerator로 로그인이나 회원가입 버튼을 누를 때 작동하게 됩니다.
         * 이 메서드 안에서 WWWform 객체를 통해 톰캣 서버와 통신을 하고
         * 통신 결과가 오류가 없는지 확인한 뒤 로그인 혹은 회원가입 처리를 하게 됩니다.
         */

        if (nullCheck(unityVal, nof) != null && isEqualPwd() == "match")
        {
            /* isEqualPwd 메서드의 경우
             * 회원가입에서 비밀번호를 두 번 입력하게 만든 뒤,
             * 비밀번호와 확인용 비밀번호의 입력이 다를 경우
             * 회원가입을 할 수 없도록 만든 것이기 때문에
             * 로그인 단계에선 불필요한 메서드입니다.
             * 따라서 회원가입 클래스(SignUp.cs)의 isEqulPwd()는
             * 검사를 시행하게 되지만 로그인 클래스(Login.cs)의 isEqualPwd()는
             * 무조건 "match"를 리턴하도록 만들었습니다. */

            WWWForm form = new WWWForm();
            // 새로운 WWWForm 객체 form을 만들어줍니다.

            for (int i = 0; i< nof; i++)
            {
                form.AddField(jspVal[i], unityVal[i]);
            }
            w = new WWW(url, form);

            yield return w;
            /* 위에서 null 체크 등을 마친 오류가 없는 string형 배열과
             * url 주소를 가지고 톰캣 서버와 통신을 합니다. */

            if (w.error != null)
            {
                Debug.Log(w.error);
                error();
            }
            /* 오류가 있다면 에러를 출력하게 되고 그렇지 않다면 afterWWW()를 통해
             * SignUp.cs, Login.cs 소스코드에서 각각 어떤 처리를 할 것인지 정하게 됩니다.
             * Login.cs의 경우에는 톰캣 서버를 통해 DB에 있는 ID와 Password가 같은지 검사를 하게 됩니다.
             * 그렇게 받은 결과 페이지의 HTML 태그를 다 떼고 html body의 내용을 보고
             * 로그인이 성공했는지 실패했는지 검사한 뒤,
             * 로그인에 성공했으면 다음 Scene으로 넘어가도록 되어있습니다.
             * SignUp.cs의 경우는 DB에 새로운 정보를 저장하는 것이기 때문에
             * 별도의 검사 없이 바로 처음 Scene으로 넘어가도록 만들었습니다.*/


            else
            {
                afterWWW();
            }
        }
        else if (nullCheck(unityVal, nof) == null)
        {
            nullField();
        }
        /* 위의 returnField 메서드를 통해 각 개체에 접근하여 얻어온 text 값이
         * null인지 아닌지 확인하는 메서드입니다.
         * 회원가입이나 로그인 시에 빈 칸이 없어야 하기 때문에 구현하게 됐습니다. */

        else if (isEqualPwd()== "" && nullCheck(unityVal, nof) == "NOT NULL")
        {
            notMatch();
        }
    }
}
```

이상으로 아두이노에서 센서들의 값을 어떻게 읽어오고, 그 값들로 자전거의 속력과 방향은 어떻게 정했으며 회원가입과 로그인 과정의 비슷한 처리 루틴을 템플릿 메서드 패턴으로 묶은 코드에 대한 설명이었습니다. 감사합니다.