# 캡스톤 디자인

진행상황

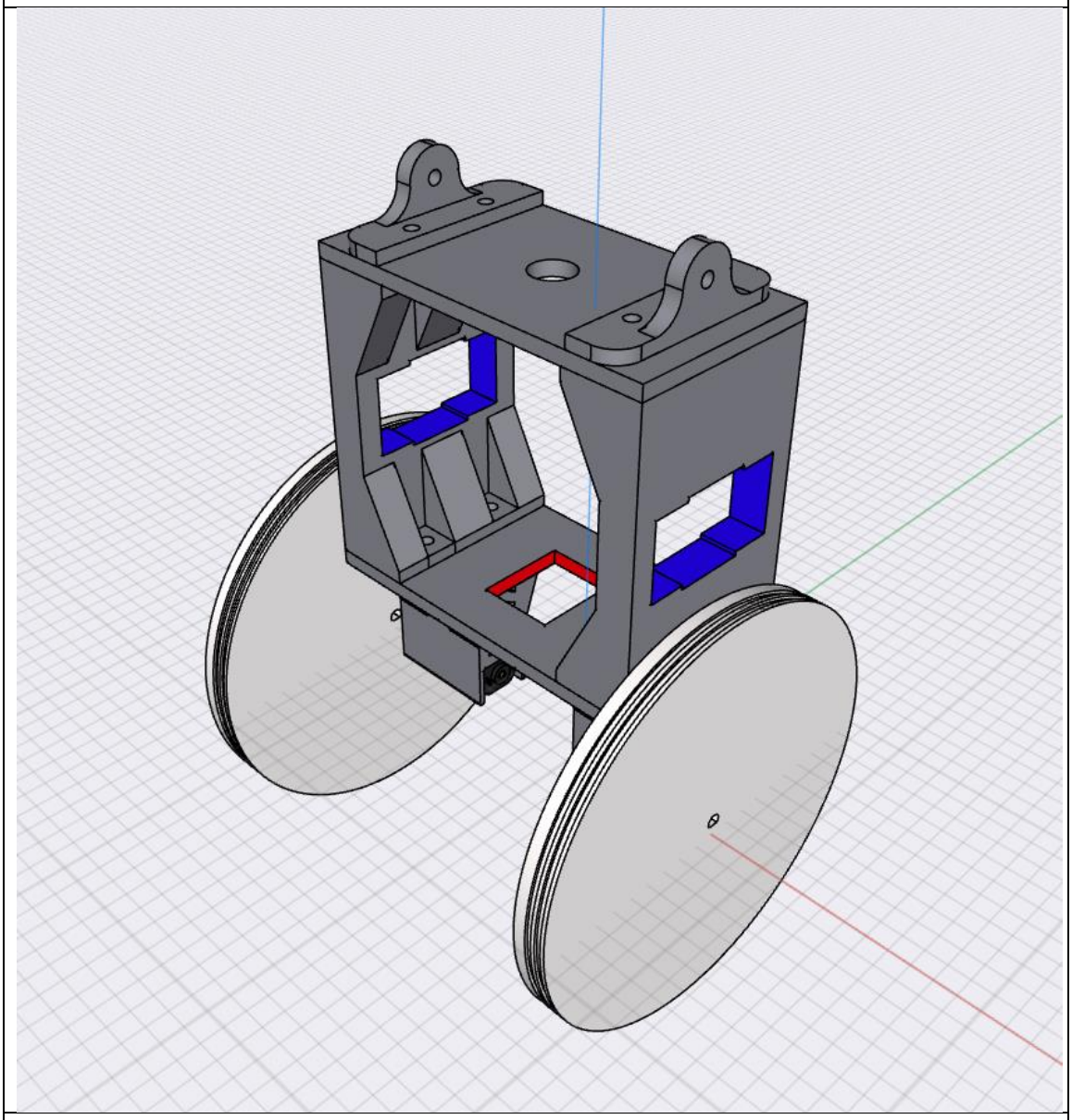
팀장: 박광렬

팀원: 유안

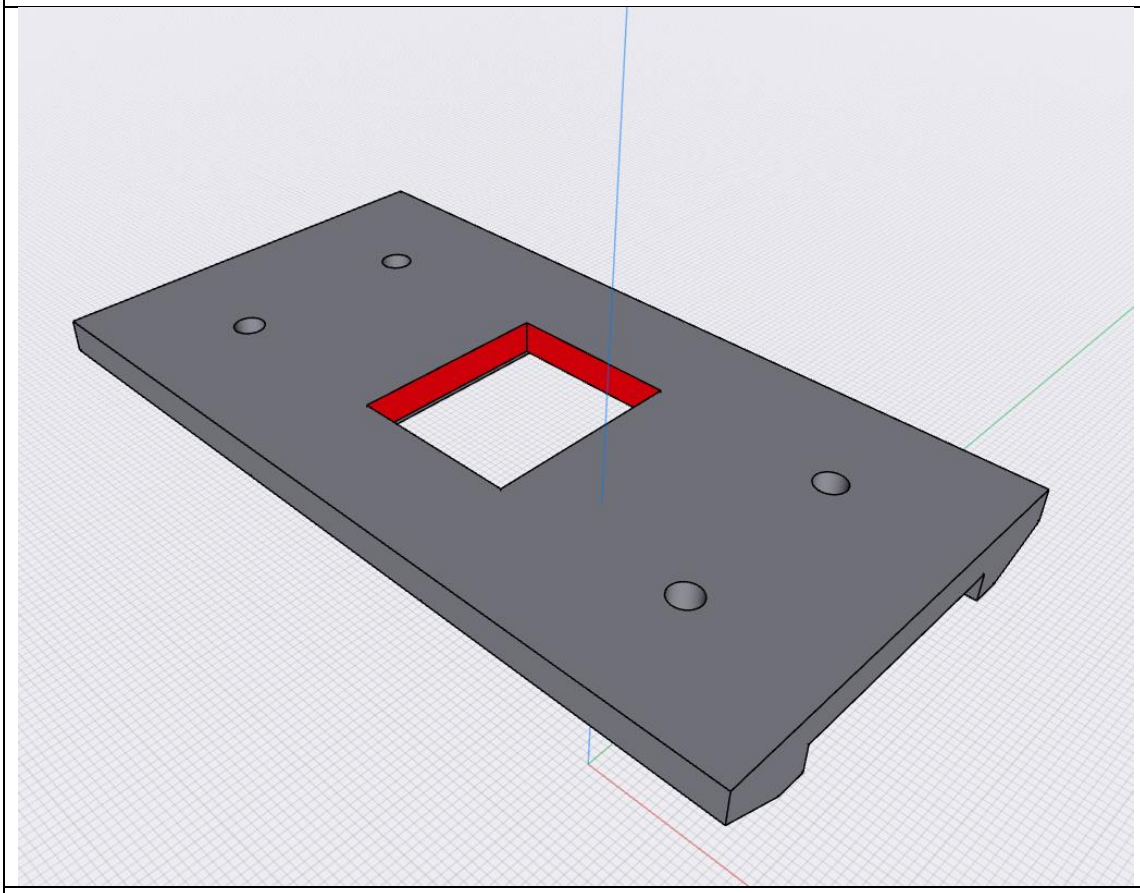
1. 진행 상황

1.1. 하드웨어

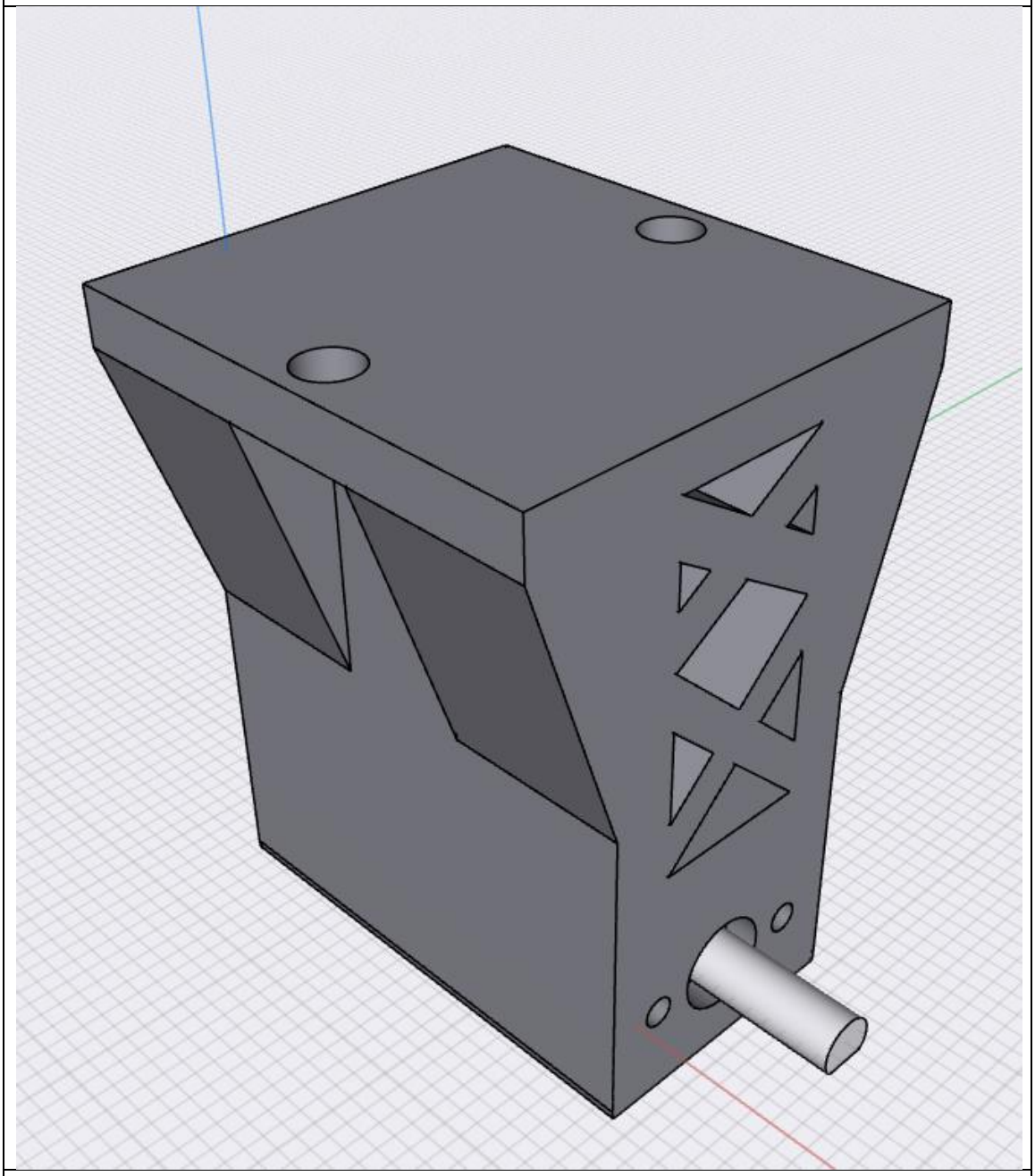
1.1.1. 기구물

| 전체 완성도 |
| --- |
|  |
| Shapr 3D로 모델링 했으며, 벨런싱 로봇의 균형을 위해 대칭 구조를 이루고 있다.<br>높이는 19CM, 너비 11CM이며, 하드웨어적 안정성을 위해 상단부에 무게 중심이 위치하도록 베터리와 추가의 무게 추를 부착한다. |

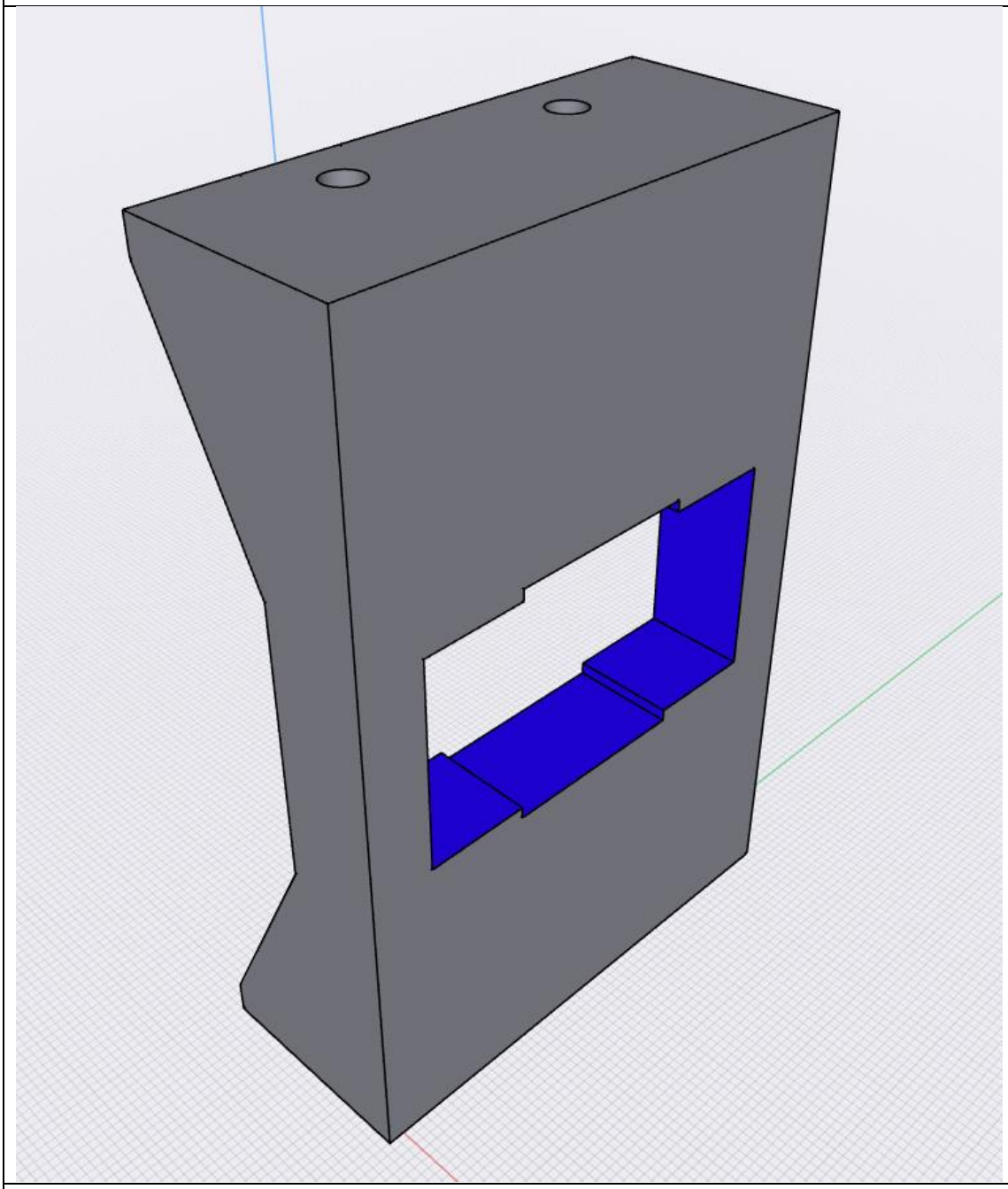
하단 받침대



모터의 지지대와 몸통을 연결하는 부품이며, 로봇의 하단에 위치한다.
붉은 색 부분에는 DRV8833 드라이버를 장착할 수 있도록 설계돼 있다.

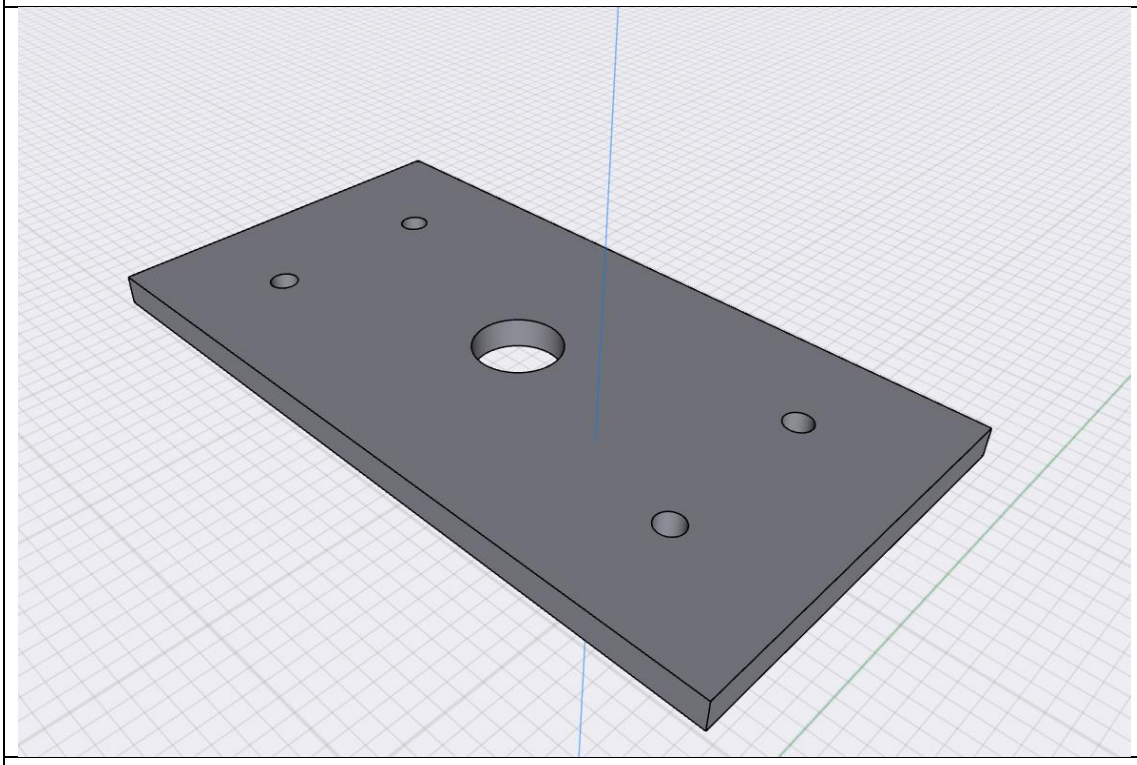
| 모터 지지대 |
| --- |
|  |
| 모터와 하단 받침대를 연결하는 지지대이며, 내구성을 위해 트러스 구조로 설계되어 있다. |

몸통



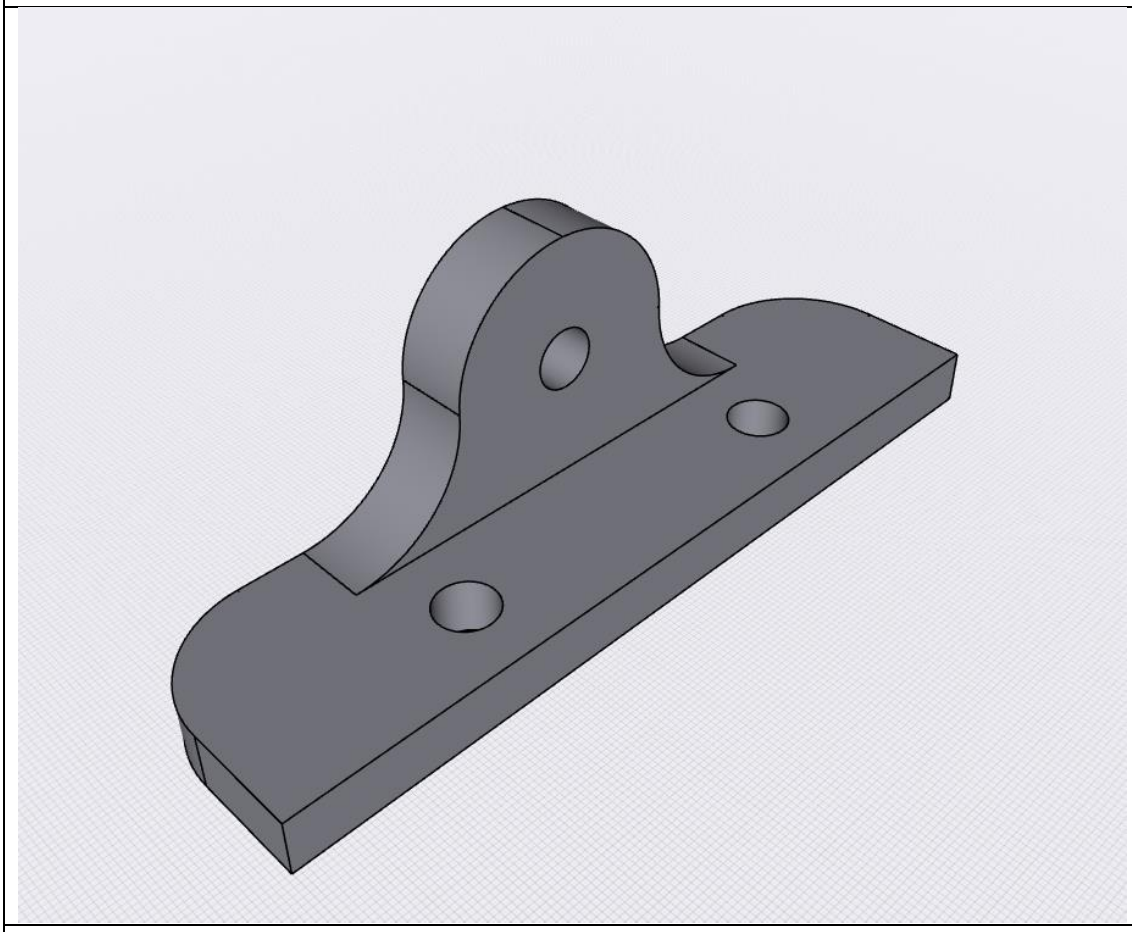
하단 받침대와 상단 받침대를 연결하는 구조물이며, 푸른색 영역에 Arduino Pro mini 3.3v이 위치하게 된다.

상단 받침대



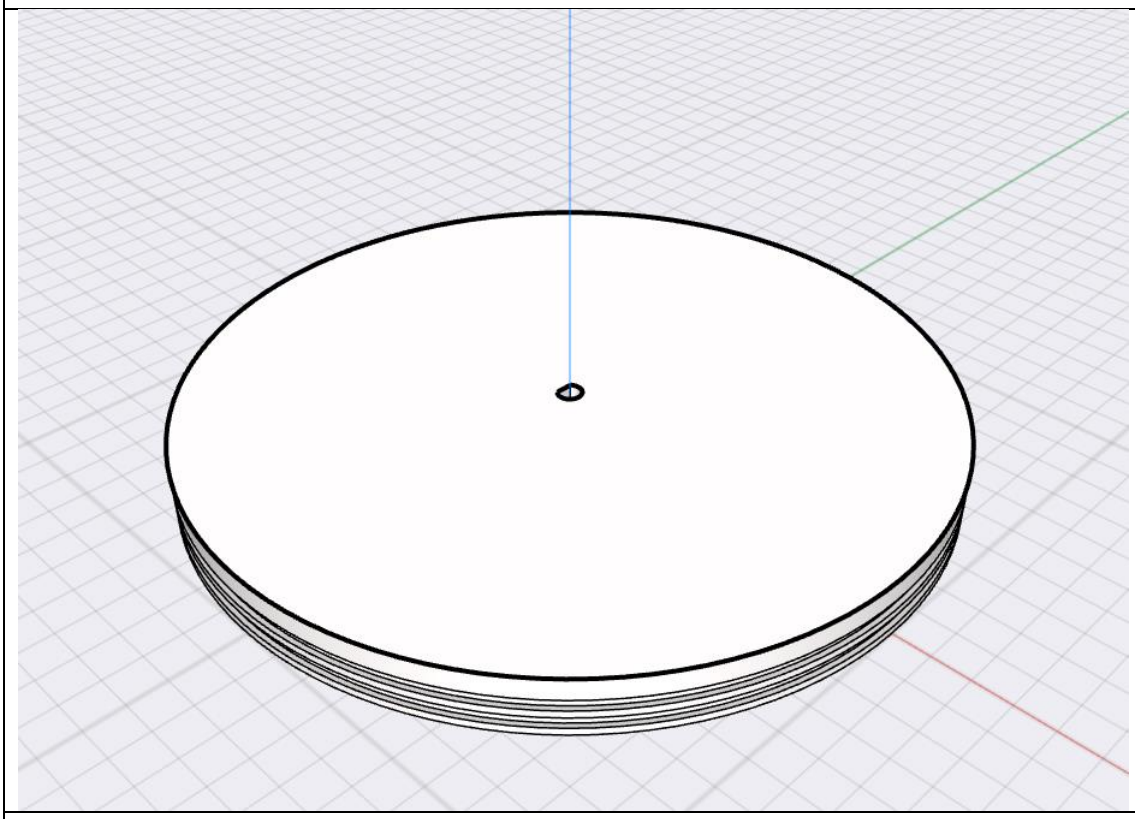
몸통 위에 위치한 상단 받침대이며, MPU6050과 베터리 그리고 무게 추가 위치하는 공간이다. MPU6050과 베터리를 Arduino Pro mini 3.3v에 선으로 연결하기 중앙에 적절한 크기의 구멍을 뚫어 놓았다.

상단 지지대



벨런싱 로봇의 offset 각도를 측정하기 위한 상단 지지대이다.
상단 받침대 양쪽에 설치되며, 중앙에 뚫려 있는 구멍에 미리 준비돼 있는 축을 끼워 넣어 로봇의 균형점을 찾을 수 있도록 도와주는 장치다.

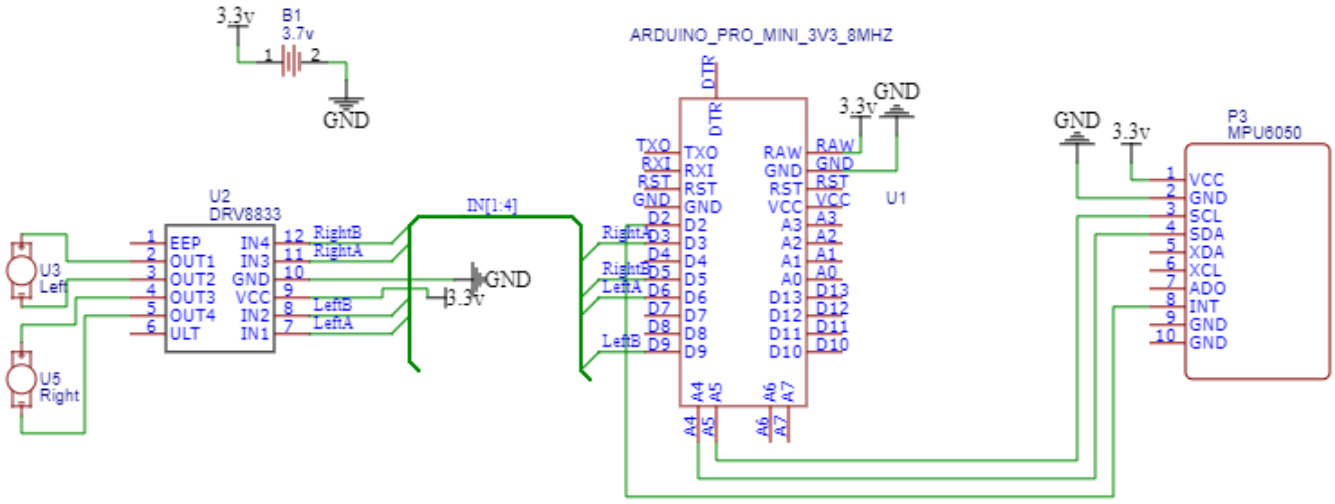
바퀴



양쪽 모터에 장착되는 바퀴로, 직경은 100MM이다.
700MM, 500MM 등의 작은 직경을 갖는 바퀴로 실험을 진행해 보았으나, 균형을 잡을 수 있는 각속도가 나오지 않았다. 또한 100MM보다 큰 직경의 경우, 적절한 토크가 나오지 않아 정상적인 동작을 방해했다. 따라서 100MM 직경을 갖는 바퀴를 채택하게 되었다.

### 1.1.2. 전자파트

회로도




연결 상태

| Arduino | DRV8833 | MPU6050 | Battery |
|---|---|---|---|
| D2 (Interrupt) | | INT | |
| D3 (PWM) | IN3 (Right motor +) | | |
| D5 (PWM) | IN4 (Right motor -) | | |
| D6 (PWM) | IN1 (Left mtor +) | | |
| D9 (PWM) | IN2 (Left motor -) | | |
| A4 (SDA) | | SDA | |
| A5 (SCL) | | SCL | |
| RAW (3.3v) | VCC | VCC | 3.7v |
| GND (ref) | GND | GND | GND |

| Arduino <-> DRV8833 |
|---|
| DRV8833은 모터 드라이버이며, 모터는 PWM 신호를 기반으로 속도를 제어한다.<br>IN1과 IN2는 로봇의 왼쪽 모터의 각각 +와 -에 연결되어 있으며, 이 IN1,2를 PWM 제어하기 위해 Arduino의 PWM 핀인 D3과 D5를 연결했다.<br>IN3와 IN4도 동일하다. |

| Arduino <-> MPU6050 |
|---|
| MPU6050은 I2C 통신을 기반으로 다른 장치와 통신하기에 Arduino의 I2C 통신 선인 A4(SDA), A5(SCL)에 연결했다. 또 MPU6050은 FIFO 구조를 가지고 있으며, 새로운 데이터가 발생할 시 INT핀이 활성화된다. 이를 즉각적으로 Arduino에서 발견하기 위해 Arduino Pro mini 3.3v의 Pin Interrupt를 제공하는 D2 핀에 MPU6050의 INT 핀을 연결했다. |

2. 소프트웨어

Main

```
#include <Wire.h>
#include <Kalman.h>
#include <CMPU6050.h>
#include "Motor.h"
#include "PID.h"

#define MOTORL_A   6
#define MOTORL_B   9
#define MOTORR_A   3
#define MOTORR_B   5

Motor L_Motor(MOTORL_A, MOTORL_B);
Motor R_Motor(MOTORR_A, MOTORR_B);

CMpu6050Manager g_Mpu6050;

void setup() {
  Serial.begin(19200);

  g_Mpu6050.Init(); //setup에서 반드시 한번 호출해 줍니다.
  pinMode(13, OUTPUT);
  digitalWrite(13, 1);



  pinMode(4, OUTPUT);    //   Motor standby
  digitalWrite(4, 1);
  motor(0, 0);
}

PID pid;

uint32_t timer = 0;
uint32_t previous = 0;
PID_TYPE gap = 0;
PID_TYPE target = -3.15;   //   -2.9

//#define GET_OFFSET_ANGLE

void loop() {

  g_Mpu6050.Update();
```

```
    float kalmanangley = g_Mpu6050.GetKalAngleY();
    float kalmananglex = g_Mpu6050.GetKalAngleX();
    timer = millis();
    if (abs(kalmanangley) > 20 || abs(kalmananglex) > 20) {
      motor(0, 0);
      digitalWrite(4, LOW);
      digitalWrite(13, 0);
      while (1);
    }
  #ifndef GET_OFFSET_ANGLE
    PID_TYPE error = kalmanangley - target;
    PID_TYPE val;
    gap = PID_TYPE (timer - previous);
    if (gap == 0) {
      gap = 0.1;
    }
    gap *= 0.001;
    Serial.println(gap, 5);

    pid.PID_cal(error, &val, gap);

    val = constrain(val, -255, 255);
    motor(val, val);
    previous = timer;

    PID_TYPE p, i, d;
    pid.PID_vals(&p, &i, &d);
  #else
    Serial.print(kalmanangley);
    Serial.print("₩n");
  #endif
  }

  void motor(int lspd, int rspd) {
    L_Motor.wheel(lspd);
    R_Motor.wheel(rspd);
}
```

상단 지지대를 통해 얻은 offset 각도를 target 변수로 넣어 안정된 각도를 목표 각도로 설정한다.
Pid 계산 라이브러리에 interval, error 값을 전달하여 Proportional, Intergrate, Derivate 연산을 취한다.

계산된 결과 값은 모터의 양쪽 신호 값으로 들어가 모터의 속도를 조절한다.

PID

```
#ifndef _PID_H
#define _PID_H

typedef double PID_TYPE;

#define Kc   90.0
#define Tc   0.45
/*
const PID_TYPE P_gain = 90 * 0.6;
const PID_TYPE I_gain = P_gain * 2 / Tc;//0.9;
const PID_TYPE D_gain = P_gain * Tc * 0.125;//43;
*/

const PID_TYPE P_gain = 50;
const PID_TYPE I_gain = 280;//0.9;
const PID_TYPE D_gain = 4;//43;

class PID {
  public:
    PID();
    void PID_cal(PID_TYPE error, PID_TYPE* pid, PID_TYPE gap);
    void PID_vals(PID_TYPE* p, PID_TYPE* i, PID_TYPE* d);

  private:
    PID_TYPE   previous_error;
    PID_TYPE   I_temp;
    PID_TYPE   P_control, I_control, D_control;
};

PID::PID() {
  previous_error = 0;
  I_temp = 0;
};

void PID::PID_cal(PID_TYPE error, PID_TYPE* pid, PID_TYPE gap) {
  I_temp += (error * gap);
  *pid = (P_control = P_gain * error) + (I_control = I_gain * (I_temp)) + (D_control = D_gain *
((error - previous_error) / gap));
  previous_error = error;
```

```
};

void PID::PID_vals(PID_TYPE* p, PID_TYPE* i, PID_TYPE* d) {
  *p = P_control;
  *i = I_control;
  *d = D_control;
}



#endif
```

실험적 결과로 얻은 PID gain 값을 각각 P_gain, I_gain, D_gain에 저장한 후, 계산 시 이를 ghkfyddgks 다.
PID_cal 함수에서는 파라미터로 받은 error, pid 주소, gap(time interval)을 활용하여 PID 값을 계산한다.
Error는 목표값과 현재 값 사이의 오차 값이다.
Pid 주소는 계산된 결과가 저장될 포인터 변수이다.
Gap은 Integration, Derivative 연산의 dt를 위한 변수이다.

3. 문제점 및 해결 방안

문제

어느정도 안정적인 자세를 취하지만, 로봇이 흔들리는 현상이 계속되며, 급작스러운 외란이 있을 시, 로봇의 균형이 무너진다.

해결 방안

PID 제어의 특징을 고려하여 각각의 gain 값을 조절하여 안정도를 확보한다.