

# 전국과학전람회

## 탐구일지

줄줄연수원



# 탐구일지

탐구 주제	줄다리기의 이해	일자	2022.03.26
줄다리를 이기고 진다는 것은 무엇인가? -> 줄을 매개로 상대방을 끌어오는 것.			
탐구 내용 및 과정	<p>여기서 무거우면 유리하다 &gt; 왜? -&gt; 관성이 커지고, 수직항력이 커져 마찰력이 강해짐 만약 질량이 같다면? -&gt; 누우면 누울수록 유리해진다. &gt; 왜? -&gt; 자유물체도를 통해 알아보자</p>		

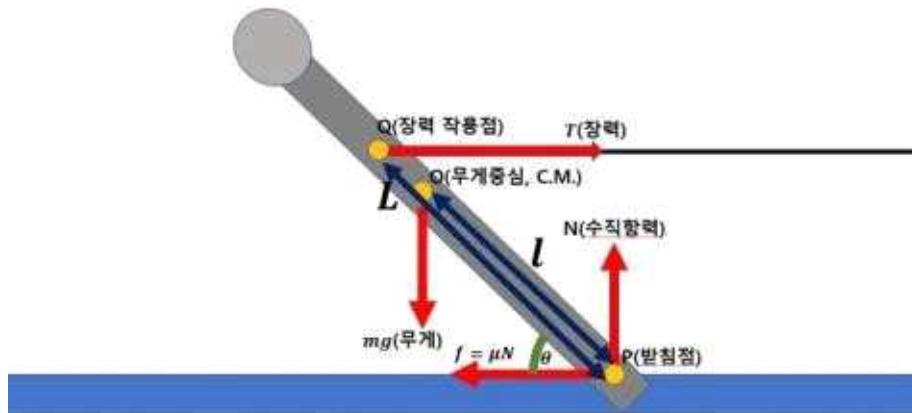
## 첨부 자료



# 탐구일지



줄을 당기는 사람에 대한 자유물체도(FBD)



그 이유는 돌림힘 평형임.

- 탐구 결과
- 1) T에 대한  $\sin\theta$  값이 작아지므로 돌림힘이 작아진다!
  - 2) 누우면 누울수록 지면에 작용하는 수직항력이 작아진다!

+ ) 작용 반작용과 힘의 평형에 의해 마찰력 싸움임을 알아내었다.

추후 계획

여기서 들어올려지기 때문에 지거나 최대 정지 마찰력이 모자라 미끄러여 진다.  
그러면 그 사이 최적의 각도가 있지 않을까?

# 탐구일지

탐구 주제	줄다리기 최적의 각도	일자	2022.03.28
탐구 내용 및 과정	<p>최적의 각도를 구하려면? 힘의 평형식과 돌림힘의 평형식을 연립!</p> <p>작용하는 장력을 <math>T</math>, 받침점으로부터 줄의 작용점까지의 거리를 <math>L</math>, 받침점으로부터 무게중심 까지의 거리를 <math>l</math>이라 하면 힘의 평형과 돌림힘의 평형을 고려하였을 때, 다음의 관계가 성립 한다.</p> <p><math>x</math>방향 힘의 평형(<math>\sum \vec{F}_x = 0</math>): <math>T - f = 0</math>, <math>\therefore T = f</math> .....①</p> <p><math>y</math>방향 힘의 평형(<math>\sum \vec{F}_y = 0</math>): <math>N - mg = 0</math>, <math>\therefore N = mg</math> .....②</p> <p>돌림힘의 평형(<math>\sum \vec{\tau}_{ext} = 0</math>): <math>(\vec{l} \times \vec{mg}) - (\vec{L} \times \vec{T}) = 0</math>, <math>\therefore TL\sin\theta = mgl\cos\theta</math> .....③</p> <p>최대 정지 마찰력: <math>F \leq \mu N</math> .....④</p> <p>③에서 <math>\tan\theta</math>에 대해 정리하면 <math>\tan\theta = \frac{mgl}{TL}</math> 이고 ①, ②를 대입하면, <math>\tan\theta = \frac{N}{f} \cdot \frac{l}{L}</math> 이다. ④</p> <p>식을 변형한 <math>\frac{N}{f} \geq \frac{1}{\mu}</math>를 대입하면 최종적으로 <math>\tan\theta \geq \frac{l}{\mu L}</math>이라는 식이 나온다. 이때 ④식에서 최대정지마찰력, 즉 최대의 마찰력이 작용하는 것은 등호를 이를 때이므로 <math>\tan\theta = \frac{l}{\mu L}</math> 일 때 마찰력 및 장력(①식 참고)이 가장 큰 값을 갖는다.</p>		

첨부 자료



## 탐구일지

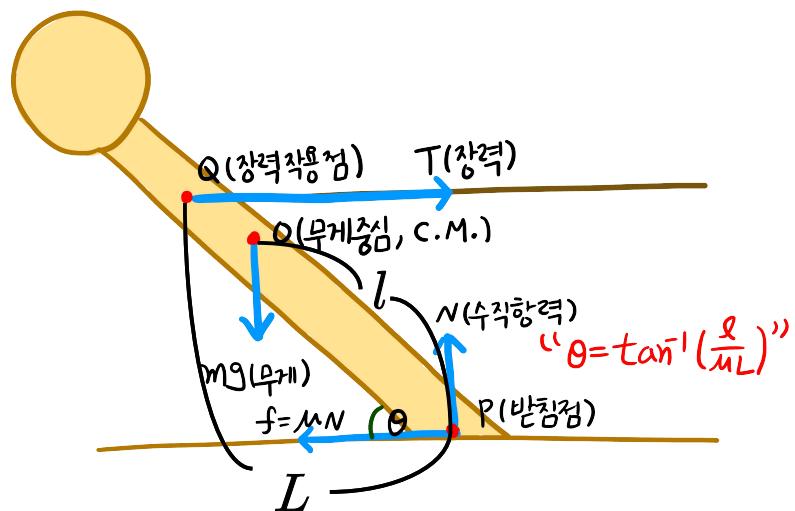


그림 6 줄을 당기며 버티고 있는 사람의 자유물체도

### 탐구 결과

줄다리기에서 평형을 유지하면서 가장 큰 마찰력을 만들어 내는 방법은 이 각도 ( $\theta = \tan^{-1} \frac{l}{\mu L}$ )를 유지하면서 버티는 것이다

### 추후 계획

같은 조건에서 최적의 각도를 동시에 유지한다면?

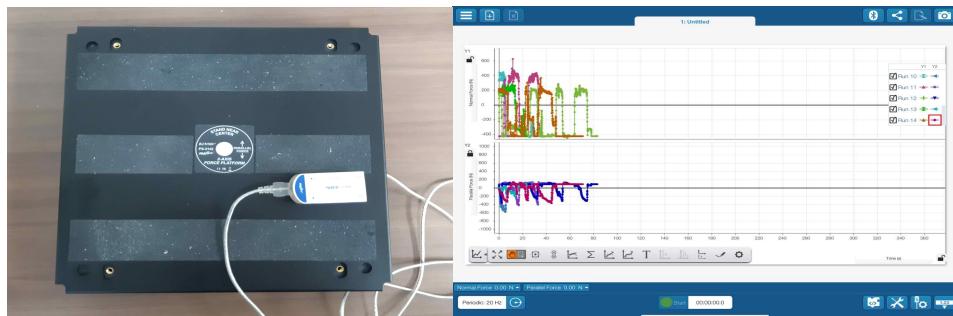
## 탐구일지

탐구 주제	서로 같은 조건에, 최적의 각도를 유지하면 영원히 승부가 나지 않을까?	일자	2022.04.06			
탐구 내용 및 과정	<p>단순히 정적인 상태에서는 알 수 없는 추가적인 변수가 있지 않을까? &gt; 선수들의 훈련영상을 분석함으로써 ‘반동’에 대한 힌트를 얻음. &gt; 상대를 끌고오려면 더 큰 최대정지마찰력이 필요함!   &gt; 무게 중심을 들어올리는 반동을 통해 일시적으로 더 큰 수직항력을 얻을 수 있음.</p>					
첨부 자료						
 <p>엉덩이를 내리면서 무게를 주면서 하는 스텝이에요</p>						
탐구 결과	무게 중심을 들어올리는 반동을 이용하여 순간적으로 더 큰 수직항력을 얻으면 상대방을 같은 조건에서도 이길 수 있을 것이다.					
추후 계획	정량적인 데이터의 수집					

# 탐구일지

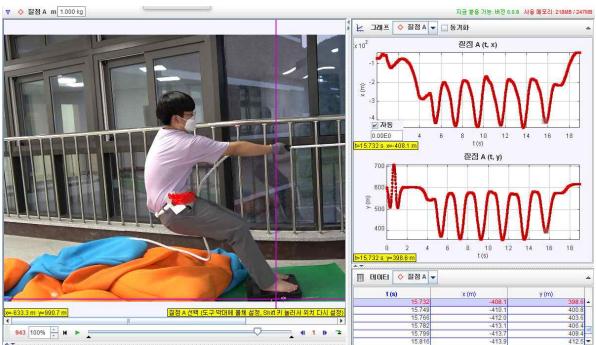
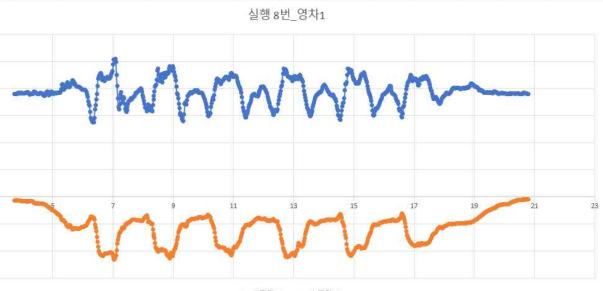
탐구 주제	정량적인 데이터의 수집	일자	2022.04.13
탐구 내용 및 과정	<p>이론적인 분석과 추론을 넘어서 정량적인 데이터 수집의 필요성을 느낌 최종적으로 가장 중요한 것은 (최대 정지) 마찰력, 그러나 그에 영향을 주는 수직항력 또한 중요하단은 것을 알게 됨. 이때 수직력과 수평력을 동시에 측정할 센서가 필요함. &gt; 2축 힘 센서를 이용해 측정하자</p> <p>무게 중심을 이용한 반동 역시 핵심 개념임을 알게 됨 &gt; 따라서 무게 중심의 이동을 tracking 한다면 유의미한 결과를 얻을 수 있지 않을까? 따라서 실제 출다리기를 진행하며 힘을 측정하고, 무게중심 이동을 측정해보자</p> <p>준비물 : 고무매트, 출, 장갑, 2축 힘 센서, 트래킹 용 질점, 충격완화 매트</p>		

## 첨부 자료



탐구 결과	다음 실험을 위한 준비 완료
추후 계획	준비물을 이용해 무게 중심의 반동을 확인해보자.

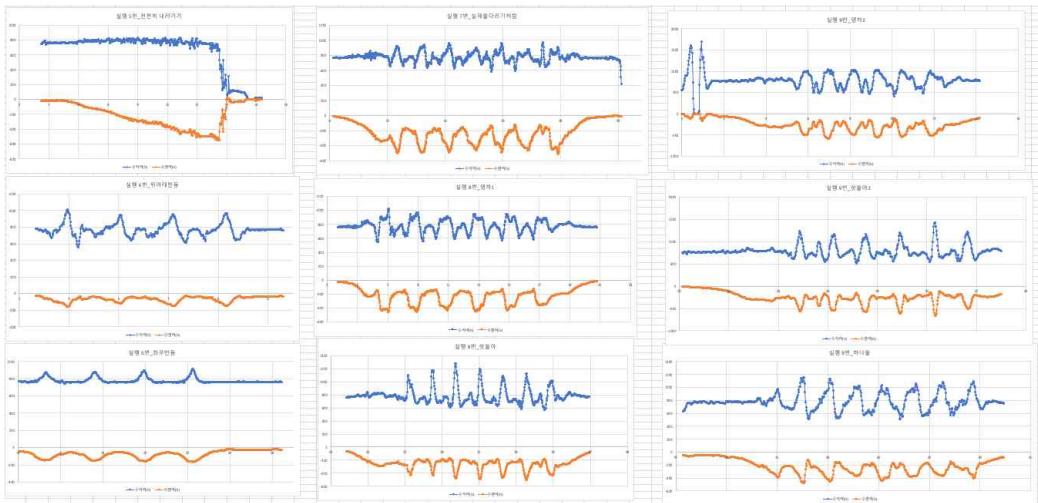
# 탐구일지

탐구 주제	정량적인 데이터의 수집	일자	2022.04.17																																																												
탐구 내용 및 과정	<p>기둥에 줄을 묶고, 1차 실험 진행</p> <p>1) 각도에 따른 수직/수평력의 관계          2) 좌우/상하 반동에 따른 수직/수평력의 관계          3) 실제 줄다리기처럼 반동을 줄 때의 양상</p>																																																														
첨부 자료																																																															
탐구 결과	 <table border="1"> <thead> <tr> <th>Time (s)</th> <th>Vertical Force (N)</th> <th>Horizontal Force (N)</th> </tr> </thead> <tbody> <tr><td>0.00</td><td>0.00</td><td>400.00</td></tr> <tr><td>1.00</td><td>100.00</td><td>400.00</td></tr> <tr><td>2.00</td><td>200.00</td><td>400.00</td></tr> <tr><td>3.00</td><td>300.00</td><td>400.00</td></tr> <tr><td>4.00</td><td>400.00</td><td>400.00</td></tr> <tr><td>5.00</td><td>500.00</td><td>400.00</td></tr> <tr><td>6.00</td><td>600.00</td><td>400.00</td></tr> <tr><td>7.00</td><td>700.00</td><td>400.00</td></tr> <tr><td>8.00</td><td>800.00</td><td>400.00</td></tr> <tr><td>9.00</td><td>900.00</td><td>400.00</td></tr> <tr><td>10.00</td><td>1000.00</td><td>400.00</td></tr> <tr><td>11.00</td><td>100.00</td><td>400.00</td></tr> <tr><td>12.00</td><td>200.00</td><td>400.00</td></tr> <tr><td>13.00</td><td>300.00</td><td>400.00</td></tr> <tr><td>14.00</td><td>400.00</td><td>400.00</td></tr> <tr><td>15.00</td><td>500.00</td><td>400.00</td></tr> <tr><td>16.00</td><td>600.00</td><td>400.00</td></tr> <tr><td>17.00</td><td>700.00</td><td>400.00</td></tr> <tr><td>18.00</td><td>800.00</td><td>400.00</td></tr> </tbody> </table> 			Time (s)	Vertical Force (N)	Horizontal Force (N)	0.00	0.00	400.00	1.00	100.00	400.00	2.00	200.00	400.00	3.00	300.00	400.00	4.00	400.00	400.00	5.00	500.00	400.00	6.00	600.00	400.00	7.00	700.00	400.00	8.00	800.00	400.00	9.00	900.00	400.00	10.00	1000.00	400.00	11.00	100.00	400.00	12.00	200.00	400.00	13.00	300.00	400.00	14.00	400.00	400.00	15.00	500.00	400.00	16.00	600.00	400.00	17.00	700.00	400.00	18.00	800.00	400.00
Time (s)	Vertical Force (N)	Horizontal Force (N)																																																													
0.00	0.00	400.00																																																													
1.00	100.00	400.00																																																													
2.00	200.00	400.00																																																													
3.00	300.00	400.00																																																													
4.00	400.00	400.00																																																													
5.00	500.00	400.00																																																													
6.00	600.00	400.00																																																													
7.00	700.00	400.00																																																													
8.00	800.00	400.00																																																													
9.00	900.00	400.00																																																													
10.00	1000.00	400.00																																																													
11.00	100.00	400.00																																																													
12.00	200.00	400.00																																																													
13.00	300.00	400.00																																																													
14.00	400.00	400.00																																																													
15.00	500.00	400.00																																																													
16.00	600.00	400.00																																																													
17.00	700.00	400.00																																																													
18.00	800.00	400.00																																																													
추후 계획	<p>손해와 이익이 동시에 존재하는 개념인 반동을 어떻게 해야 이기는 전략으로 사용 가능할까?          우리가 사용했던 구호와도 관련 있지 않을까?</p>																																																														

# 탐구일지

탐구 주제	4월 17일 실험의 결과 정리	일자	2022.04.26
탐구 내용 및 과정	<p>1) 얻은 영상에 대해 Tracking 실시      2) 얻은 수직/수평력 데이터를 엑셀을 이용해 가공      3) 시간 축을 일치 시켜 두 데이터 비교</p>		

## 첨부 자료



탐구 결과	<p>고무판에서 진행하면서 얻은 결론</p> <ol style="list-style-type: none"> <li>최적의 각도를 찾으면서 진행을 해보니 고무판에서는 10~20도가 넘어지지 직전인 각도이다.</li> <li>따라서 넘어지기 전의 각도를 키워야 하기 때문에 마찰을 줄이자.</li> <li>따라서 신발의 마찰을 줄이고, 반동을 주자</li> </ol> <p>생각의 발전</p> <ol style="list-style-type: none"> <li>줄다리기를 이기려면 어떻게 해야하나?</li> <li>장력은 서로 같다, 그렇다면 마찰력싸움!</li> <li>미끄러지지 않고 안 끌리는 최적의 각도가 존재</li> <li>서로 최적의 각도면 어떻게 승패가 갈리지?</li> </ol>
-------	--

## 탐구일지

	<ol style="list-style-type: none"><li>5. 반동을 줌으로써 순간적으로 더 큰 수직항력 얻음</li><li>6. 시간의 이득? 힘의 이득? 운동량? 충격량?</li><li>7. 무게중심이 내려가는 상황에서는 손해인데?</li><li>8. 어떤 전략이 필요한거지?</li><li>9. 로봇으로 구현</li><li>10. 가장 단순한 모델은 뭐지?</li></ol>
추후 계획	

## 탐구일지

탐구 주제	줄다리기 모델 제안	일자	2022.05.15
탐구 내용 및 과정	<p>줄다리기 로봇을 만들기 위해, 줄다리기의 가장 간단한 모형을 제안할 필요성을 느낌 그 핵심이 최적의 각도를 유지하기 위해 외부의 충격에 대한 변화가 있을 때 앞 뒤로 주춤거리며 다시 최적의 각도를 유지하는 것이라 판단</p> <ul style="list-style-type: none"><li>▶ 1인 이동수단인 세그웨이, 즉 셀프 밸런싱 휠과 흡사하다고 생각함</li><li>▶ 그러나 세그웨이는 지면에 수직, 우리가 원하는 각도는 약 45도</li><li>▶ 유사하나 분명 차이점이 존재</li><li>▶ 이에 경사면에서의 셀프 밸런싱 휠을 생각하며 자유물체와 힘의 분해를 통해 분석</li><li>▶ 해당 경사를 다시 지면과 일치하도록 돌리면, 경사각만큼의 각도를 유지하는 휠</li><li>▶ 대신 수평 방향으로 당겨주는 힘 필요 -&gt; 장력 T!</li></ul> <p>가장 간단한 줄다리기 모형을 제안함!</p>		

### 첨부 자료



탐구 결과	줄다리기의 첫 핵심 메커니즘, ‘최적의 각도를 유지하다 각도에 변화가 생기면 앞뒤로 주춤거려 다시 각도를 맞춘다.’에 해당하는 모델 제안의 실마리를 얻었다.
추후 계획	단, 발과 바퀴의 차이를 과연 고려해야하는 가에 대해서 더 논의가 필요함.

## 탐구일지

탐구 주제	로봇 제작	일자	2022.05.20
탐구 내용 및 과정	아두이노 벨런싱로봇 프로그램(참조 및 출처 : circuitdigest.com)을 참고하여 MPU6050 센서를 다루어봄. (줄줄이 1호)		
첨부 자료			

## 탐구일지

```
// 아두이노 블런싱로봇 프로그램 (참조 및 출처 : circuitdigest.com)

#include "I2Cdev.h"
#include <PID_v1.h>
#include "MPU6050_6Axis_MotionApps20.h"
MPU6050 mpu;

// MPU control/status vars
bool dmpReady = false; // set true if DMP init was successful
uint8_t mpuIntStatus; // holds actual interrupt status byte from MPU
uint8_t devStatus; // return status after each device operation (0 =
success, !0 = error)
uint16_t packetSize; // expected DMP packet size (default is 42 bytes)
uint16_t fifoCount; // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer

// orientation/motion vars
Quaternion q; // [w, x, y, z] quaternion container
VectorFloat gravity; // [x, y, z] gravity vector
float ypr[3]; // [yaw, pitch, roll] yaw/pitch/roll container and
gravity vector

// 아래의 4개의 값을 본인의 로봇에 맞게 튜닝합니다.
//*****파라메터 튜닝 시작*****
double setpoint= 220; //로봇이 지면에서 평형을 유지하는 상태의 값입니다.
//다음은 PID 제어기의 Kp, Ki, Kd 파라메터를 설정합니다. 아래의 순서대로 설정합니다.
double Kp = 21;
double Kd = 0.8;
double Ki = 10;
//*****파라메터 튜닝 끝*****

double input, output;
PID pid(&input, &output, &setpoint, Kp, Ki, Kd, DIRECT);

volatile bool mpuInterrupt = false; // MPU6050의 인터럽트 발생유무 확인
void dmpDataReady()
{
    mpuInterrupt = true;
}
```

# 탐구일지

```
void setup() {
    Serial.begin(115200);

    // MPU6050 초기화
    Serial.println(F("Initializing I2C devices..."));
    mpu.initialize();

    // MPU6050 통신확인
    Serial.println(F("Testing device connections..."));
    Serial.println(mpu.testConnection() ? F("MPU6050 connection successful") :
F("MPU6050 connection failed"));

    // DMP 초기화
    devStatus = mpu.dmpInitialize();

    // 기본 음센сор 설정
    mpu.setXGyroOffset(220); // 수평일때의 기본 x축 자이로값
    mpu.setYGyroOffset(76);
    mpu.setZGyroOffset(-85);
    mpu.setZAccelOffset(1688);

    // 정상동작하는 경우
    if (devStatus == 0)
    {
        // DMP 가동
        Serial.println(F("Enabling DMP..."));
        mpu.setDMPEnabled(true);

        // 아두이노 인터럽트 설정
        Serial.println(F("Enabling interrupt detection (Arduino external
interrupt 0)..."));
        attachInterrupt(0, dmpDataReady, RISING);
        mpuIntStatus = mpu.getIntStatus();

        // DMP 사용가능 상태 설정
        Serial.println(F("DMP ready! Waiting for first interrupt..."));
        dmpReady = true;

        // 패킷사이즈 가져오기
        packetSize = mpu.dmpGetFIFOPacketSize();

        //PID 설정
        pid.SetMode(AUTOMATIC);
        pid.SetSampleTime(10);
        pid.SetOutputLimits(-255, 255);
    }
    else
    {
        // 오류시
        // 1 = 초기 메모리 에러
        // 2 = DMP 설정 오류
        Serial.print(F("DMP Initialization failed (code "));
        Serial.print(devStatus);
        Serial.println(F("")));
    }

    //모터 출력핀 초기화
    pinMode (6, OUTPUT);
    pinMode (9, OUTPUT);
    pinMode (10, OUTPUT);
    pinMode (11, OUTPUT);

    //모터 동작 OFF
    analogWrite(6,LOW);
    analogWrite(9,LOW);
    analogWrite(10,LOW);
    analogWrite(11,LOW);
}
```

## 탐구일지

```
void loop() {
    // 오류시 작업중지
    if (!dmpReady) return;

    // MPU 인터럽트나 패킷 대기
    while (!mpuInterrupt && fifoCount < packetSize)
    {
        //MPU6050 데이터가 없는 경우 PID 계산
        pid.Compute();

        //シリ얼 모니터로 현재 상태 출력
        Serial.print(input); Serial.print(" =>"); Serial.println(output);

        if (input>150 && input<250){//로봇이 기울어지는 경우(각도 범위내에서만)

            if (output>0) //앞으로 기울어지는 경우
                Forward(); //전진
            else if (output<0) //뒤로 기울어지는 경우
                Reverse(); //후진
            }
            else //로봇이 기울어지지 않은 경우
                Stop(); //모터 정지
        }

        // reset interrupt flag and get INT_STATUS byte
        mpuInterrupt = false;
        mpuIntStatus = mpu.getIntStatus();

        // get current FIFO count
        fifoCount = mpu.getFIFOCount();

        // check for overflow (this should never happen unless our code is too inefficient)
        if ((mpuIntStatus & 0x10) || fifoCount == 1024)
        {
            // reset so we can continue cleanly
            mpu.resetFIFO();
            Serial.println(F("FIFO overflow!"));

            // otherwise, check for DMP data ready interrupt (this should happen frequently)
        }
        else if (mpuIntStatus & 0x02)
        {
            // wait for correct available data length, should be a VERY short wait
            while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();

            // read a packet from FIFO
            mpu.getFIFOBytes(fifoBuffer, packetSize);

            // track FIFO count here in case there is > 1 packet available
            // (this lets us immediately read more without waiting for an interrupt)
            fifoCount -= packetSize;

            mpu.dmpGetQuaternion(&q, fifoBuffer); //get value for q
            mpu.dmpGetGravity(&gravity, &q); //get value for gravity
            mpu.dmpGetYawPitchRoll(ypr, &q, &gravity); //get value for ypr

            input = ypr[1] * 180/M_PI + 180;
        }
    }
}
```

## 탐구일지

```
void Forward() //전진
{
    analogWrite(6,output);
    analogWrite(9,0);
    analogWrite(10,output);
    analogWrite(11,0);
    Serial.print("F");
}

void Reverse() //후진
{
    analogWrite(6,0);
    analogWrite(9,output*-1);
    analogWrite(10,0);
    analogWrite(11,output*-1);
    Serial.print("R");
}

void Stop() //정지
{
    analogWrite(6,0);
    analogWrite(9,0);
    analogWrite(10,0);
    analogWrite(11,0);
    Serial.print("S");
}
```

탐구 결과

추후 계획

## 탐구일지

탐구 주제	IMU 갑의 조정	일자	2022.05.23
탐구 내용 및 과정	MPU6050의 센서값을 받아서, 어떤 방식으로 가공을 진행해야 하는지 고민함.		
첨부 자료			

## 탐구일지



```
#include<Wire.h>
const int MPU=0x68; //MPU 6050 의 I2C 기본 주소
int16_t AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;

void setup(){
    Wire.begin(); //Wire 라이브러리 초기화
    Wire.beginTransmission(MPU); //MPU로 데이터 전송 시작
    Wire.write(0x6B); // PWR_MGMT_1 register
    Wire.write(0); //MPU-6050 시작 모드로
    Wire.endTransmission(true);
    Serial.begin(115200);
    Serial.println("CLEARDATA");
    Serial.println("LABEL, AcZ, GyX, GyY"); }

void loop(){
    Wire.beginTransmission(MPU); //데이터 전송시작
    Wire.write(0x3B); // register 0x3B (ACCEL_XOUT_H), 큐에 데이터 기록
    Wire.endTransmission(false); //연결유지
    Wire.requestFrom(MPU,14,true); //MPU에 데이터 요청
    //데이터 한 바이트 씩 읽어서 반환
    AcX=Wire.read()<<8|Wire.read();
    AcY=Wire.read()<<8|Wire.read();
    AcZ=Wire.read()<<8|Wire.read();
    Tmp=Wire.read()<<8|Wire.read();
    GyX=Wire.read()<<8|Wire.read();
    GyY=Wire.read()<<8|Wire.read();
    GyZ=Wire.read()<<8|Wire.read();

    //시리얼 모니터에 출력
    Serial.print("DATA, ");
    Serial.print(AcZ);
    Serial.print(",");
    Serial.print(GyX);
    Serial.print(",");
    Serial.print(GyY);
    Serial.println();
    delay(333);
}
```

탐구 결과 각도에 영향을 주는 요소를 찾음

추후 계획 이 요소를 고려하여 각도를 유지하는 코드를 작성하고자 함.

# 탐구일지

탐구 주제	핵심 요소 알고리즘 제작			일자	2022.06.01				
탐구 내용 및 과정	이러한 값을 엑셀에 기록하여 어떤 방식으로 알고리즘을 짜야 할지 고민함.								
첨부 자료									
<b>AcZ</b>		<b>GyX</b>		<b>GyY</b>					
				상태					
22424	734	53		0					
22440	598	46		0					
22442	600	45		0					
22358	585	43		0					
22384	643	48		0					
22408	616	46		0					
22314	645	48		0					
22384	641	48		0					
22344	630	47		0					
22376	608	46		0					
21178	-2150	-87		-1					
16156	411	37		-1					
16044	595	47		-1					
15988	595	47		-1					
15990	603	49		-1					
16000	604	44		-1					
15920	612	47		-1					
15874	619	46		-1					
15902	614	44		-1					
15862	636	47		-1					
19388	4519	249		0					
22412	320	33		0					
22410	542	43		0					
22390	570	46		0					
22380	642	48		0					
22406	612	47		0					
22494	598	44		0					
22432	620	47		0					
22490	598	46		0					
22444	572	43		0					
20804	3592	163		1					
13292	485	135		1					
13968	591	44		1					
13944	612	45		1					
13936	613	45		1					
13992	611	41		1					
14052	610	45		1					
13926	615	48		1					
13818	467	56		1					
17794	-976	94		1					
22464	767	56		0					
22448	266	32		0					
22386	646	48		0					
22440	660	50		0					
탐구 결과									
Serial을 통해 AcZ, GyX, GyY값을 받고 값의 범위를 찾고자 함.									
추후 계획									
값의 범위를 통한 모터 조절을 진행하고자 함.									

# 탐구일지

탐구 주제	모터의 움직임 확인 코드 작성	일자	2022.06.02			
탐구 내용 및 과정	DC 모터가 움직이는지에 대해 확인함.					
첨부 자료						
						
	<pre>#define output 100 // 기본적인 움직임으로 가정한다. void setup() { //모터 출력핀 초기화     pinMode (6, OUTPUT);     pinMode (9, OUTPUT);     pinMode (10, OUTPUT);     pinMode (11, OUTPUT);      //모터 동작 OFF     analogWrite(6,LOW);     analogWrite(9,LOW);     analogWrite(10,LOW);     analogWrite(11,LOW); }  void loop() {     Forward(); //전진     delay(1000);     Reverse(); //후진     delay(1000);     Stop(); //모터 정지     delay(1000); }  void Forward() //전진 {     analogWrite(6,output);     analogWrite(9,0);     analogWrite(10,output);     analogWrite(11,0); }  void Reverse() //후진 {     analogWrite(6,0);     analogWrite(9,output*-1);     analogWrite(10,0);     analogWrite(11,output*-1); }  void Stop() //정지 {     analogWrite(6,0);     analogWrite(9,0);     analogWrite(10,0);     analogWrite(11,0); }</pre>					
탐구 결과	함수를 만들어 사용할 준비를 함.					
추후 계획	위 알고리즘과 이 코드를 합쳐 알고리즘을 짜게 됨.					

## 탐구일지

탐구 주제	코드 안정화	일자	2022.06.12
탐구 내용 및 과정	알고리즘을 짜서 구현해보았지만, 매우 불안정하며, 값이 매우 뒤는 현상을 발견해 코드를 수정함		
첨부 자료			

# 탐구일지



```
/*
    arduino ==> 셀프밸런싱로봇을 만들어 실험해본 코드
*/
//-----
// PID 제어, 모터 제어, MPU6050센서 값을 받기 위한 선언문
#include <PID_v1.h>
#include <LMotorController.h>
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"

#if I2CDEV_IMPLEMENTATION == I2CDEV_ADUINO_WIRE
    #include "Wire.h"
#endif

//-----
#define OUTPUT_TEAPOT 1                      // Processing을 통해 MPU6050 센서를
Visualize 하고 싶은 경우 1, 아니면 0으로 선언합니다
#define MIN_ABS_SPEED 30                     // 모터의 최저속도를 설정합니다. 0 ~ 255 값
중 선택
#define OUTPUT_READABLE_YAWPITCHROLL          // Yaw, Pitch, Roll 값을 얻기 위해 선언합니다
#define INTERRUPT_PIN 2                      // MPU6050 센서의 INT 핀이 꽂혀있는 번호를 설정합니다. 보통 2번
#define LED_PIN 13                           // Arduino Uno의 13번핀 LED를 동작 중에 반
짝거리게 하려고 선언합니다

//-----
//MPU 객체를 선언합니다
MPU6050 mpu;
// MPU control/status vars
bool blinkState = false;           // LED를 반짝거리게 하기 위한 변수
bool dmpReady = false;             // set true if DMP init was successful
uint8_t mpuIntStatus;              // holds actual interrupt status byte from MPU
uint8_t devStatus;                // return status after each device operation (0 =
success, !0 = error)
uint16_t packetSize;               // expected DMP packet size (default is 42 bytes)
uint16_t fifoCount;                // count of all bytes currently in FIFO
uint8_t fifoBuffer[64];            // FIFO storage buffer
```

## 탐구일지



```
//-----
// MPU6050 센서를 통해 쿼터니언과 오일러각, Yaw, Pitch, Roll 값을 얻기 위해 선언합니다
// orientation/motion vars
Quaternion q;           // [w, x, y, z]      quaternion container
VectorInt16 aa;          // [x, y, z]      accel sensor measurements
VectorInt16 aaReal;       // [x, y, z]      gravity-free accel sensor
                         measurements
VectorInt16 aaWorld;     // [x, y, z]      world-frame accel sensor
                         measurements
VectorFloat gravity;     // [x, y, z]      gravity vector
float euler[3];          // [psi, theta, phi] Euler angle container
float ypr[3];            // [yaw, pitch, roll] yaw/pitch/roll container and
                         gravity vector

//-----
// Processing으로 MPU6050 센서를 Visualize 하기 위한 변수
// packet structure for InvenSense teapot demo
uint8_t teapotPacket[14] = { '$', 0x02, 0,0, 0,0, 0,0, 0,0, 0x00, 0x00, '\r', '\n' };
};

//-----
// PID 제어용 변수 선언
double kp = 2;
double ki = 0.8;
double kd = 10;

// 기울임 각도 선택
// 제가 만든 뱀파싱로봇에는 184.0도가 가장 최적의 평형각도었습니다
// 각도가 180도를 기준으로 +-를 설정해주시면 됩니다
double originalSetpoint = 270;
double setpoint = originalSetpoint;
double movingAngleOffset = 0.3;

// PID 제어용 input, output 변수를 선언합니다
double input, output;

// PID값을 설정해준다
PID pid(&input, &output, &setpoint, kp, ki, kd, DIRECT);
```

# 탐구일지



```
// 모터 제어용 변수 선언
// EnA, EnB는 속도제어용(pwm), IN1,2,3,4는 방향제어용 핀입니다
int ENA = 10;
int IN1 = 12;
int IN2 = 6;
int IN3 = 9;
int IN4 = 8;
int ENB = 11;

// motorController 객체 생성, 맨 끝 파라미터 1,1은 각각 좌측, 우측모터의 최대속도(%) 입니다.
LMotorController motorController(ENA, IN1, IN2, ENB, IN3, IN4, 1, 1);

// =====
// ==           INTERRUPT DETECTION ROUTINE           ==
// =====

volatile bool mpuInterrupt = false;      // indicates whether MPU interrupt pin has
gone high

void dmpDataReady() {
    mpuInterrupt = true;
}

void setup(){
    // join I2C bus (I2Cdev library doesn't do this automatically)
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
    Wire.begin();
    Wire.setClock(400000); // 400kHz I2C clock. Comment this line if having
compilation difficulties
#elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
    Fastwire::setup(400, true);
#endif

    Serial.begin(115200);
    while (!Serial); // wait for Leonardo enumeration, others continue immediately

    // initialize device
    Serial.println(F("Initializing I2C devices..."));
    mpu.initialize();
    pinMode(INTERRUPT_PIN, INPUT);
    // verify connection
    Serial.println(F("Testing device connections..."));
    Serial.println(mpu.testConnection() ? F("MPU6050 connection successful") :
F("MPU6050 connection failed"));
}
```

# 탐구일지



```
// load and configure the DMP
Serial.println(F("Initializing DMP..."));
devStatus = mpu.dmpInitialize();

// supply your own gyro offsets here, scaled for min sensitivity
mpu.setXGyroOffset(220);
mpu.setYGyroOffset(76);
mpu.setZGyroOffset(-85);
mpu.setZAccelOffset(1788); // 1688 factory default for my test chip

// devStatus 값이 0 이면 정상작동, 0이 아니면 오작동입니다
// make sure it worked (returns 0 if so)
if (devStatus == 0){
    // turn on the DMP, now that it's ready
    Serial.println(F("Enabling DMP..."));
    mpu.setDMPEnabled(true);

    // enable Arduino interrupt detection
    Serial.println(F("Enabling interrupt detection (Arduino external interrupt
0)..."));
    attachInterrupt(digitalPinToInterrupt(INTERRUPT_PIN), dmpDataReady,
RISING);
    mpuIntStatus = mpu.getIntStatus();

    // set our DMP Ready flag so the main loop() function knows it's okay to
use it
    Serial.println(F("DMP ready! Waiting for first interrupt..."));
    dmpReady = true;

    // get expected DMP packet size for later comparison
    packetSize = mpu.dmpGetFIFOPacketSize();

    // MPU6050 센서가 정상작동하면 PID 제어용 코드를 초기화합니다
    pid.SetMode(AUTOMATIC);
    pid.SetSampleTime(10);
    pid.SetOutputLimits(-255, 255);
}
else{ // MPU6050 센서가 오작동한 경우
    // ERROR!
    // 1 = initial memory load failed
    // 2 = DMP configuration updates failed
    // (if it's going to break, usually the code will be 1)
    Serial.print(F("DMP Initialization failed (code "));
    Serial.print(devStatus);
    Serial.println(F("")));
}

// 로봇이 작동 중 13번 LED를 깜빡거리기 위해 OUTPUT으로 초기화합니다
pinMode(LED_PIN, OUTPUT);
}
```

# 탐구일지



```
void loop(){
    // if programming failed, don't try to do anything
    if (!dmpReady) return;

    // wait for MPU interrupt or extra packet(s) available
    while (!mpuInterrupt && fifoCount < packetSize){
        //no mpu data - performing PID calculations and output to motors

        pid.Compute(); // 루프를 돌면서 pid 값을 업데이트합니다
        motorController.move(output, MIN_ABS_SPEED); // pid 연산으로 나온 output 값을
        motorController로 전송합니다. (모터제어)
    }

    // reset interrupt flag and get INT_STATUS byte
    mpuInterrupt = false;
    mpuIntStatus = mpu.getIntStatus();

    // get current FIFO count
    fifoCount = mpu.getFIFOCount();

    // MPU6050 센서가 정상작동하는 경우에만 PID제어를 해야하므로 아래와 같이 if-else문을 작성합니다
    // check for overflow (this should never happen unless our code is too
    inefficient)
    if ((mpuIntStatus & 0x10) || fifoCount == 1024){
        // reset so we can continue cleanly
        mpu.resetFIFO();
        Serial.println(F("FIFO overflow!"));
    }
    // MPU6050 센서가 정상작동하는 경우
    else if (mpuIntStatus & 0x02){
        // wait for correct available data length, should be a VERY short wait
        while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();

        // read a packet from FIFO
        mpu.getFIFOBytes(fifoBuffer, packetSize);

        // track FIFO count here in case there is > 1 packet available
        // (this lets us immediately read more without waiting for an interrupt)
        fifoCount -= packetSize;

        // display Euler angles in degrees
        mpu.dmpGetQuaternion(&q, fifoBuffer);
        mpu.dmpGetGravity(&gravity, &q);
        mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
    }
}
```

# 탐구일지



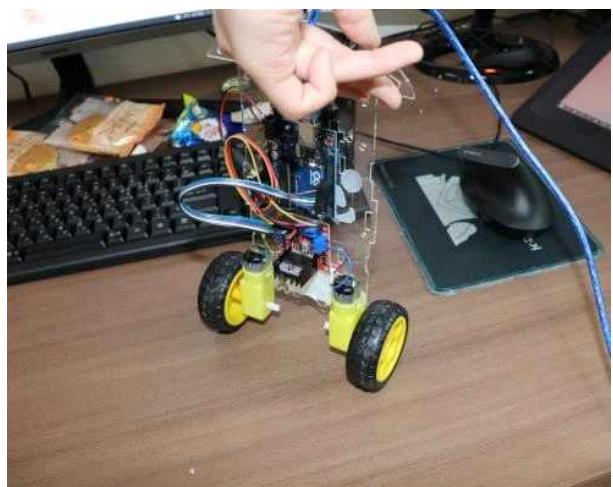
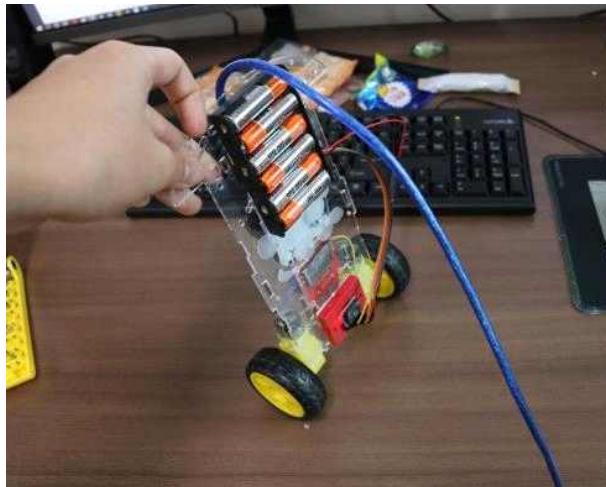
```
#ifdef OUTPUT_READABLE_YAWPITCHROLL // 센서를 통해 구한 Yaw, Pitch, Roll 값을 Serial  
Monitor에 표시합니다  
    Serial.print("ypr\t");  
    Serial.print(ypr[0] * 180/M_PI);  
    Serial.print("\t");  
    Serial.print(ypr[1] * 180/M_PI);  
    Serial.print("\t");  
    Serial.println(ypr[2] * 180/M_PI);  
#endif  
// PID 제어를 하기 위해 input 변수에 Pitch 값을 넣습니다  
input = ypr[1] * 180/M_PI + 180;  
  
#ifdef OUTPUT_TEAPOT // Processing으로 MPU6050센서의 움직임을 Visualize 하기 위한 코드  
// display quaternion values in InvenSense Teapot demo format:  
    teapotPacket[2] = fifoBuffer[0];  
    teapotPacket[3] = fifoBuffer[1];  
    teapotPacket[4] = fifoBuffer[4];  
    teapotPacket[5] = fifoBuffer[5];  
    teapotPacket[6] = fifoBuffer[8];  
    teapotPacket[7] = fifoBuffer[9];  
    teapotPacket[8] = fifoBuffer[12];  
    teapotPacket[9] = fifoBuffer[13];  
    Serial.write(teapotPacket, 14);  
    teapotPacket[11]++; // packetCount, loops at 0xFF on purpose  
#endif  
}  
}
```

탐구 결과	전의 코드를 합쳐 진행하였지만 잘 되지 않는 관계로 kalman filter와 PID를 이용함.
추후 계획	로봇에 코드를 적용하여 정말로 균형을 잘 잡는지 확인하고자 함.

# 탐구일지

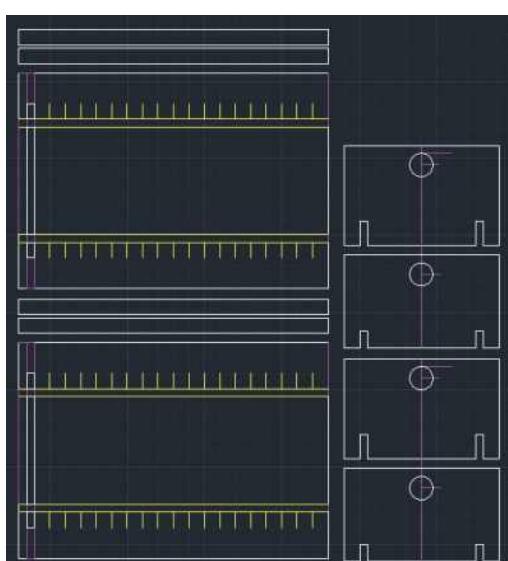
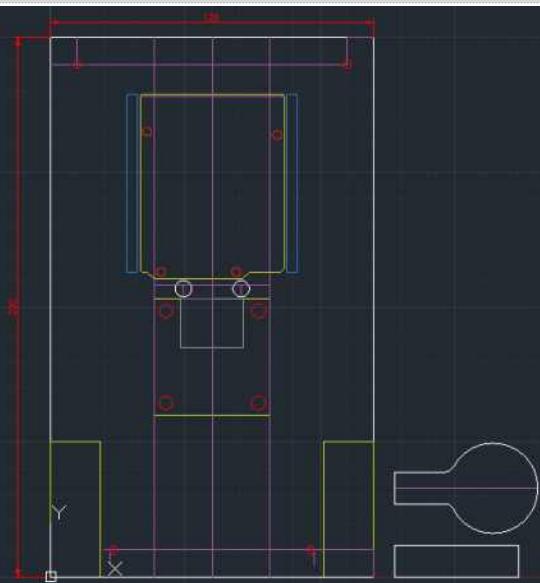
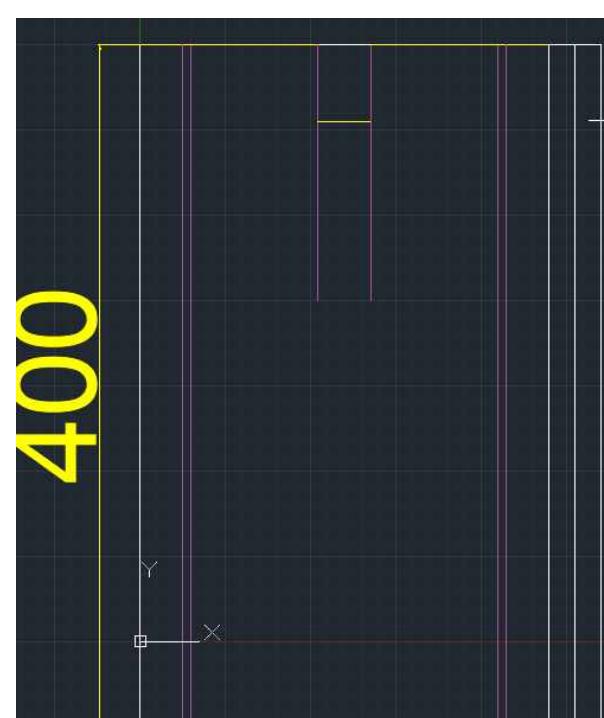
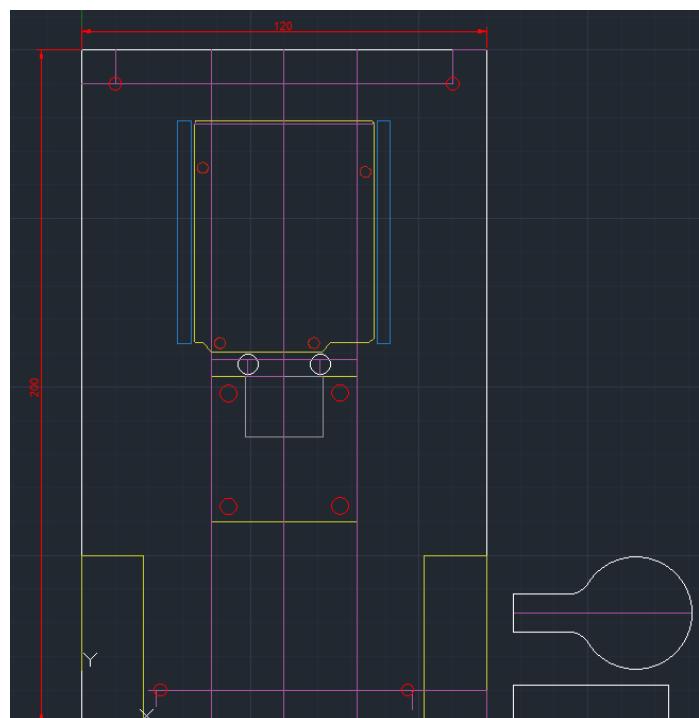
탐구 주제	6월 12일의 코드 바탕 실험	일자	2022.06.14
탐구 내용 및 과정	6월 12일의 코드를 바탕으로 조금의 PID값 조정 이후 테스트 진행		

## 첨부 자료

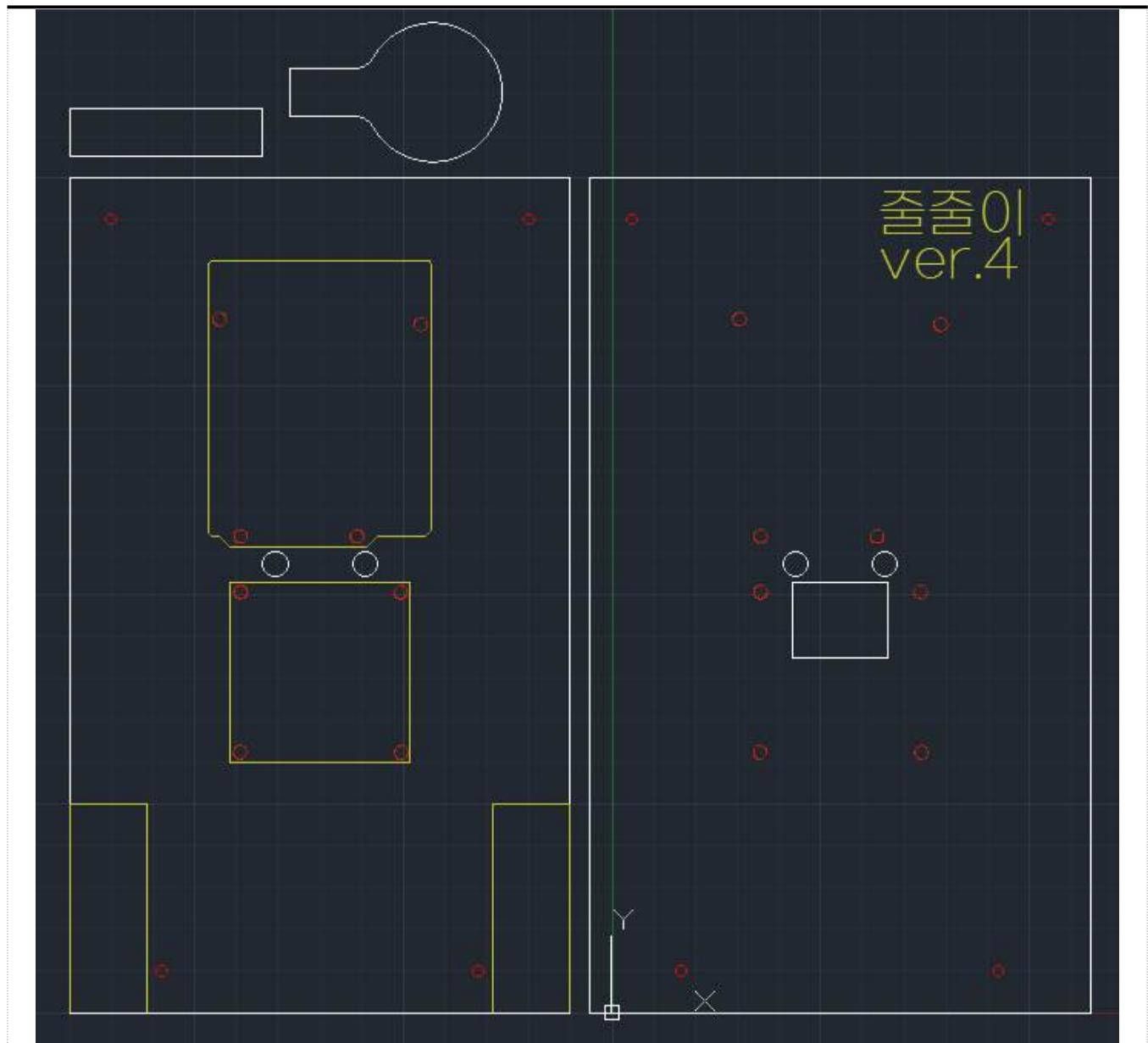


탐구 결과	코드의 적용을 통해 균형을 잡을 수 있다는 실마리를 잡음.
추후 계획	현재 로봇의 불안정한 부분을 개선하고 안정적으로 만들고자 함.

# 탐구일지

탐구 주제	코드의 안정화	일자	2022.06.15. ~ 2022.07.17			
탐구 내용 및 과정	코드를 바탕으로 설계를 진행하고, 이후 설계와 코드 속 PID값을 조정함.					
첨부 자료						
						
						
						
						

## 탐구일지



탐구 결과 아크릴을 통해서 로봇을 만들고 PID 값을 조정하여 균형을 잘 잡도록 함.

추후 계획 조금의 로봇 업그레이드를 통해 줄다리기 실험을 진행하고자 함.

# 탐구일지

탐구 주제	PID 값 조정	일자	2022.07.17
탐구 내용 및 과정	PID 값을 안정화 시키고자 함.		
첨부 자료			
			
	<pre>double kp = 40; double ki = 100; double kd = 0.8;</pre>		
			
탐구 결과	각도를 벼티는 것에 대해서 안정감 있어져, 20도까지 벼틸 수 있었다.		
추후 계획	로봇의 안정화와 모터와 바퀴의 변화를 진행하고자 함.		

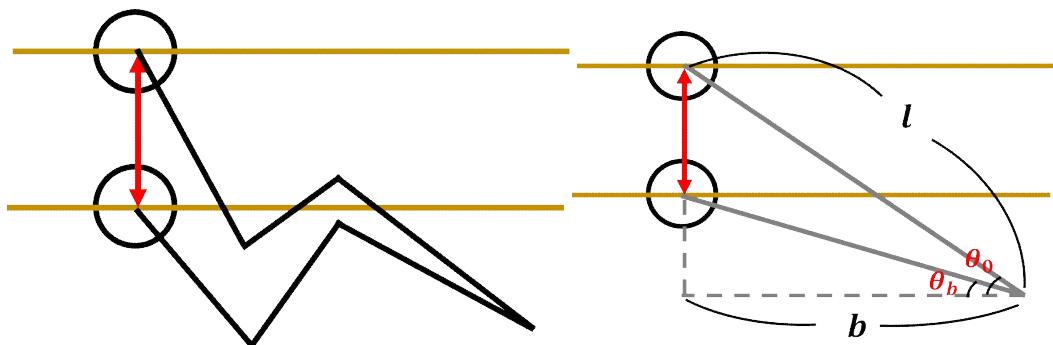
# 탐구일지

탐구 주제	반동 모델에 더 필요한 가정은?	일자	2022.08.02
탐구 내용 및 과정	<p>1. 동역학적 모델 제안을 시도하였으나, 현재의 ‘각도 유지 및 발을 앞 뒤로 주춤거림’ 만으로는 수학적 모델링에 한계를 느낀다 2. 선수들의 영상을 재분석하여 새로운 핵심 메커니즘을 발견함</p>		
첨부 자료			
			
탐구 결과	선수들의 반동 중 대부분은 발이 고정된 체 상하방향으로만 반동을 줌을 확인하였다. 즉, 반동 과정에서의 $\Delta x$ 가 0이라는 새로운 핵심 메커니즘을 발견하였다.		
추후 계획	새로운 메커니즘을 적용하여 모델을 설정하고, 수학적 모델을 유도한다.		

# 탐구일지

탐구 주제	동역학적 줄다리기 상황의 기하학적 모델 제안	일자	2022.08.04
탐구 내용 및 과정	<p>1. 앞서 얻은 핵심 메커니즘, ‘발이 고정된 상태에서 상하방향의 반동을 준다.’를 적용하여 아래 그림과 같은 기하학적 모델을 설정하였다.</p> <p>2. 설정한 모델에 대하여 <math>l, \theta</math> 값이 변하고 회전 동역학 방정식을 세울 기본 식들을 고려하였다.</p>		

첨부 자료



탐구 결과	회전동역학 상황에서의 줄다리기를 간단한 기하학적 모델로 표현하였다.
추후 계획	회전 동역학 방정식을 풀어하여 장력 T에 대한 수식을 얻는다.

## 탐구일지

탐구 주제	$\theta$ 의 변화량에 따른 T의 방정식 유도	일자	2022.08.08
탐구 내용 및 과정	<p>1. 8월 4일 설정한 기하학적 모델에 대해 회전동역학 운동 방정식을 세웠다.</p> <p>2. 아래와 같은 유도 과정을 통해 theta의 변화량에 따른 장력 T의 방정식을 얻었다.</p> <p>3. 수식을 이루는 각 항들이 가지는 의미를 고찰하여 논의하였다.</p>		

### 첨부 자료

$$\tau = \frac{dL}{dt} = \frac{d(I\omega)}{dt} = I \frac{d^2\theta}{dt^2} + \omega \frac{dI}{dt} \quad \text{수식 7}$$

$$Tl\sin\theta - mgl\cos\theta = ml^2 \frac{d^2\theta}{dt^2} + 2ml \frac{d\theta}{dt} \quad \text{수식 8}$$

$$\begin{aligned} T\sin\theta &= ml \frac{d^2\theta}{dt^2} + 2m \frac{d\theta}{dt} + mg\cos\theta \\ &= m(l \frac{d^2\theta}{dt^2} + 2 \frac{d\theta}{dt} + g\cos\theta) \end{aligned} \quad \text{수식 9}$$

$$T = m\csc\theta(l \frac{d^2\theta}{dt^2} + 2 \frac{d\theta}{dt} + g\cos\theta) \quad \text{수식 10}$$

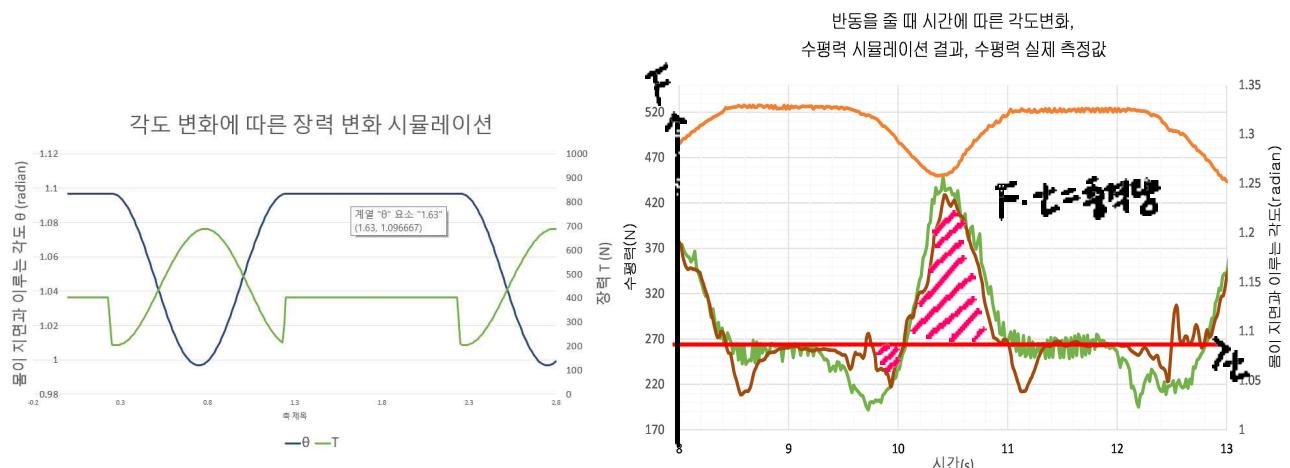
$$T = m\csc\theta(b\sec\theta \frac{d^2\theta}{dt^2} + 2 \frac{d\theta}{dt} + g\cos\theta) \quad \text{수식 11}$$

탐구 결과	반동을 주는 줄다리기 상황에서 $\theta_{(t)}$ 를 이용하여 $T_{(\theta)}$ 를 나타내는 수식을 얻었다. 전 세계 최초로 동역학 상황에서의 줄다리기 모델의 수학적 모델링에 성공하였다.
추후 계획	

# 탐구일지

탐구 주제	엑셀을 이용한 수치적해석을 통해 모델을 검증해보자	일자	2022.08.09
탐구 내용 및 과정	<ol style="list-style-type: none"> <li>유도한 수학적 모델이 타당한지 검증하기 위해 엑셀을 통한 수치적 해석을 진행함</li> <li>임의의 시간에 따른 각도 함수를 적용하여 장력 변화 시뮬레이션을 진행함</li> <li>실제 측정한 각도와 장력에 대해 예측값과 실측값을 비교분석함</li> </ol>		

## 첨부 자료

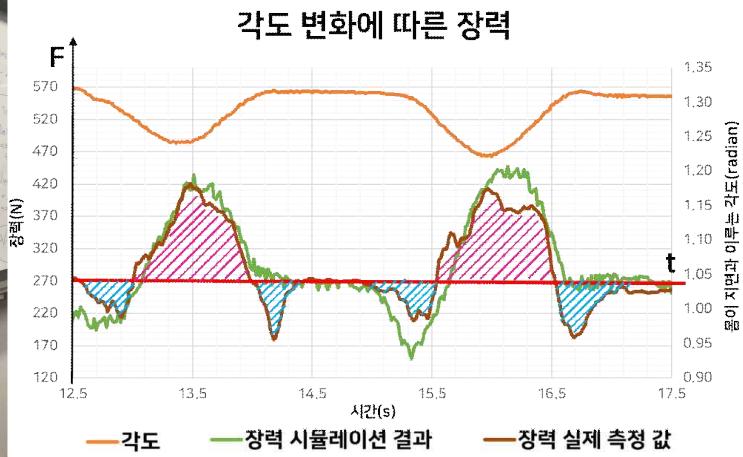
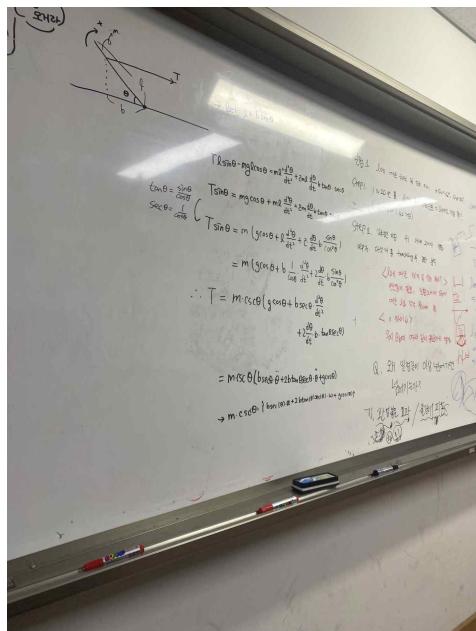


탐구 결과	버티다가 반동을 주는 상황에 대하여, 시간축을 일치시켜 각도 변화에 따라 장력 변화를 시각화 하였다. 반동의 시작과 종료 구간에서 손해를, 중간 구간에서 이득을 취하는 것을 볼 수 있다. 각도에 따른 장력의 예측값과 실측값이 거의 유사한 것을 보아 우리가 제안한 수학적 모델이 타당함을 검증하였다. 또한 장력에 대한 면적 값의 합이 양수인 것을 보아 반동이 상대를 끌어와 이길 수 있는 요인이라고 주장하는 근거를 얻을 수 있었다.
추후 계획	

# 탐구일지

탐구 주제	수학적 모델 오류 수정 및 재검토	일자	2022.08.09
탐구 내용 및 과정	<p>1. 앞서 유도한 식에서 간과한 부분을 발견하여 해당 내용을 다시 적용하여 식을 아래와 같이 재유도 하였다.</p> <p>2. 뒤이어 10월 16일과 같은 방법을 이용해 수학적 모델의 타당성 재검토를 진행하였다.</p>		

## 첨부 자료



## 탐구일지



### 탐구 결과

식의 형태는 일부 복잡해졌으나 기존의 식보다 실측값에 더욱 유사한 형태의 예측을 했음을  
위의 시각화를 통해 관찰할 수 있다.

### 추후 계획

## 탐구일지

탐구 주제	각도에 따른 PID 제어의 도입	일자	2022.08.10
탐구 내용 및 과정	PID를 다시 한번 조절하여 안정감 있게 만들고, 각도에 따른 PID를 분할하였다.		

### 첨부 자료

```
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
#include "Wire.h"
#include <PID_v1.h>
#include <LMotorController.h>

MPU6050 mpu;

#define DEBUG
#ifdef DEBUG
//#define DPRINT(args...) Serial.print(args)           //OR use the following syntax:
#define DPRINTSTIMER(t)   for (static unsigned long SpamTimer; (unsigned long)(millis() - SpamTimer)
>= (t); SpamTimer = millis())
#define DPRINTSFN(StrSize,Name,...) {char
S[StrSize];Serial.print("\t");Serial.print(Name);Serial.print(" ");
Serial.print(dtostrf((float)_VA_ARGS__ ,S));}//StringSize,Name,Variable,Spaces,Percision
#define DPRINTLN(...)      Serial.println(_VA_ARGS__)
#else
#define DPRINTSTIMER(t)   if(false)
#define DPRINTSFN(...)    //blank line
#define DPRINTLN(...)     //blank line
#endif

#define LED_PIN 13 //

// supply your own gyro offsets here, scaled for min sensitivity use MPU6050_calibration.ino
// -4232 -706 1729 173 -94 37
//          XA      YA      ZA      XG      YG      ZG
int MPUOffsets[6] = { -4232, -706, 1729, 173, -94, 37};
```

# 탐구일지

```
// =====
// ===           i2c SETUP Items           ===
// =====
void i2cSetup() {
    // join I2C bus (I2Cdev library doesn't do this automatically)
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
    Wire.begin();
    TWBR = 24; // 400kHz I2C clock (200kHz if CPU is 8MHz)
#elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
    Fastwire::setup(400, true);
#endif
}

// =====
// ===           INTERRUPT DETECTION ROUTINE           ===
// =====
volatile bool mpuInterrupt = false;      // indicates whether MPU interrupt pin has gone high
void dmpDataReady() {
    mpuInterrupt = true;
}

// =====
// ===           MPU DMP SETUP           ===
// =====
int FifoAlive = 0; // tests if the interrupt is triggering
int IsAlive = -20; // counts interrupt start at -20 to get 20+ good values before assuming connected
// MPU control/status vars
uint8_t _mpuIntStatus; // holds actual interrupt status byte from MPU
uint8_t devStatus; // return status after each device operation (0 = success, !0 = error)
uint16_t packetSize; // expected DMP packet size (default is 42 bytes)
uint16_t fifoCount; // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer

// orientation/motion vars
Quaternion q;          // [w, x, y, z]        quaternion container
VectorInt16 aa;         // [x, y, z]        accel sensor measurements
VectorInt16 aaReal;     // [x, y, z]        gravity-free accel sensor measurements
VectorInt16 aaWorld;    // [x, y, z]        world-frame accel sensor measurements
VectorFloat gravity;    // [x, y, z]        gravity vector
float euler[3];         // [psi, theta, phi] Euler angle container
float ypr[3];           // [yaw, pitch, roll] yaw/pitch/roll container and gravity vector
byte StartUP = 100; // lets get 100 readings from the MPU before we start trusting them (Bot is not trying to balance at this point it
is just starting up.)

void MPU6050Connect() {
    static int MPUMainCntr = 0;
    // initialize device
    mpu.initialize(); // same
    // load and configure the DMP
    devStatus = mpu.dmpInitialize(); // same

    if (devStatus != 0) {
        // ERROR!
        // 1 = initial memory load failed
        // 2 = DMP configuration updates failed
        // (if it's going to break, usually the code will be 1)

        char * StatStr[5] { "No Error", "initial memory load failed", "DMP configuration updates failed", "3", "4" };

        MPUMainCntr++;

        Serial.print(F("MPU connection Try #"));
        Serial.println(MPUMainCntr);
        Serial.print(F("DMP Initialization failed (code "));
        Serial.print(StatStr[devStatus]);
        Serial.println(F("")));
    }

    if (MPUMainCntr >= 10) return; //only try 10 times
    delay(1000);
    MPU6050Connect(); // Lets try again
    return;
}

mpu.setXAccelOffset(MPUOffsets[0]);
mpu.setYAccelOffset(MPUOffsets[1]);
mpu.setZAccelOffset(MPUOffsets[2]);
mpu.setXGyroOffset(MPUOffsets[3]);
mpu.setYGyroOffset(MPUOffsets[4]);
mpu.setZGyroOffset(MPUOffsets[5]);

Serial.println(F("Enabling DMP..."));
mpu.setDMPEnabled(true);
// enable Arduino interrupt detection
Serial.println(F("Enabling interrupt detection (Arduino external interrupt pin 2 on the Uno)..."));
Serial.print("mpu.getInterruptDrive= "); Serial.println(mpu.getInterruptDrive());
attachInterrupt(0, dmpDataReady, RISING); //pin 2 on the Uno
mpuIntStatus = mpu.getIntStatus(); // Same
// get expected DMP packet size for later comparison
packetSize = mpu.dmpGetFIFOPacketSize();
delay(1000); // Let it Stabilize
mpu.resetFIFO(); // Clear fifo buffer
mpu.getIntStatus();
mpuInterrupt = false; // wait for next interrupt
}
```

# 탐구일지

# 탐구일지

```
// =====
// === Motor Setup ===
// =====
#define MIN_ABS_SPEED 30 // 모터의 최저속도를 설정합니다. 0 ~ 255 값 중 선택

// 모터 제어용 변수 선언
// EnA, EnB는 속도제어용(pwm), IN1,2,3,4는 방향제어용 핀입니다
int ENA = 10;
int IN1 = 12;
int IN2 = 6;
int IN3 = 9;
int IN4 = 8;
int ENB = 11;

LMotorController motorController(ENA, IN2, IN1, ENB, IN4, IN3, 1, 1);

// =====
// === Setup ===
// =====
void setup() {
    Serial.begin(115200); //115200
    while (!Serial);
    Serial.println("i2cSetup");
    i2cSetup();
    Serial.println("MPU6050Connect");
    MPU6050Connect();
    Serial.println("Setup complete");
    pinMode(LED_PIN, OUTPUT);

    // Input(initial is original Pitch)
    Input = Pitch;

    // PID on
    myPID.SetMode(AUTOMATIC);

}
// =====
// === Loop ===
// =====
void loop() {
    if (mpuInterrupt) { // wait for MPU interrupt or extra packet(s) available
        GetDMP();
    }

    Input = Pitch;

    double gap = Input - Setpoint; // Pitch 값을 back으로 갈 수록 값이 커짐. (수직 : -15)
    // 앞으로 가면 gap은 음수, 앞으로가면 gap은 음수의 값을 가짐.
    if(gap>0)
    { //we're close to setpoint, use conservative tuning parameters
        myPID.SetTunings(bKp, bKi, bKd);
    }
    else
    {
        //we're far from setpoint, use aggressive(적극적인) tuning parameters
        myPID.SetTunings(fKp, fKi, fKd);
    }

    myPID.Compute();
    motorController.move(Output, MIN_ABS_SPEED); // pid 연산으로 나온 output 값을 motorController로 전송합니다. (모터제어)
}


```

탐구 결과

추후 계획

# 탐구일지

탐구 주제	질문의 구체화	일자
		2022.08.13
탐구 내용 및 과정	1. 반동을 주는 코드와 실제 반동 사이에 얼마나 차이나는 것일까? 2. 기존 모델과 줄줄이의 차이점은? 3. 줄줄이 성능 테스트 방법은?	
<p>유지구간 반동구간 각도 세팅값 반동</p> <p>기존 세팅값 베이스</p> <p>기존 세팅값 베이스로 초기화 기존 세팅값으로 초기화</p>		
<p>첨부 자료</p> <p>기존 세팅값 베이스로 초기화 기존 세팅값으로 초기화</p> <p><math>x^2 + y^2 = L^2</math></p> <p><math>L \sin \theta = y</math></p> <p><math>L \cos \theta = x</math></p> <p><math>T - mg \cos \theta = 0</math>, <math>T \cos \theta = \frac{dy}{dt}</math></p> <p><math>N \sin \theta - mg = m a_y</math>, <math>N \sin \theta = \frac{dx}{dt}</math></p> <p><math>T = \cancel{mg \cos \theta}</math>, <math>\tan \theta = \frac{m(a_y + g)}{T}</math></p> <p><math>T = \frac{m(a_y + g)}{\tan \theta} = \frac{m(g - L \sin \theta)}{\tan \theta}</math></p> <p>코딩으로 반동 theta</p> <p>무게 중심이 움직인다?</p> <p>각도가 커졌을 때, overshoot 크기</p> <p>각도가 작아졌을 때, " " " " </p>		
탐구 결과	<ol style="list-style-type: none"> <li>반동 코드에 의한 사각파를 주더라도 PID 제어적 성질에 의해 출렁거림이 발생할 것</li> <li>발이 고정되었던 모델과는 다르게 움직이며 L이 안 변하지만 거의 유사할 것</li> <li>도르래를 이용하여 손해를 볼 때 끌려가기도 하는 상황 세팅</li> </ol>	
추후 계획		

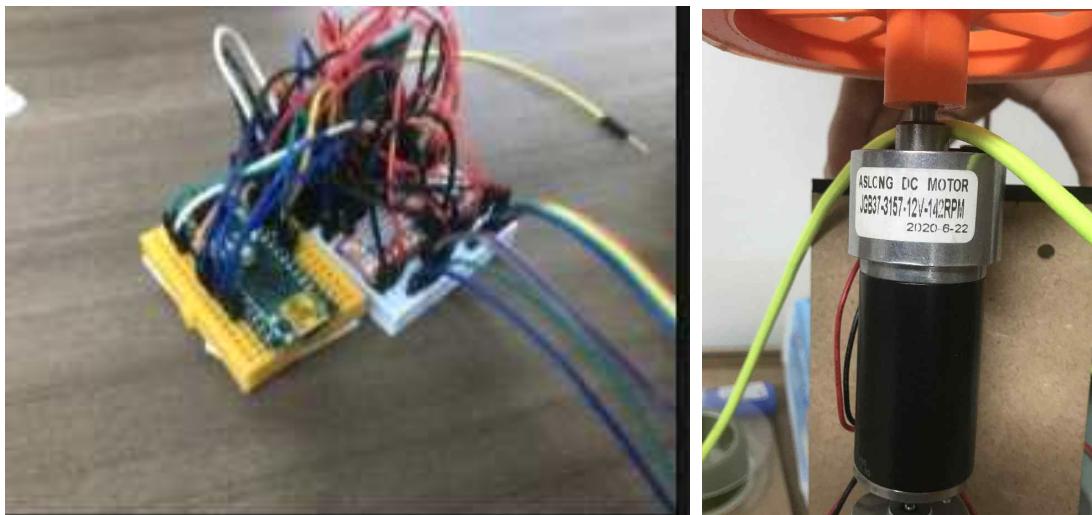
# 탐구일지

탐구 주제	바퀴 재 설계	일자	2022.08.15
탐구 내용 및 과정	바퀴의 크기, 재질에 따른 마찰력 차이가 생길 것이라고 예상하여 다양한 종류의 바퀴를 이용해 실험을 진행함.		
첨부 자료			
			
탐구 결과	100파이의 지름을 지닌 3d 출력물 바퀴를 이용하고자 함.		
추후 계획			

# 탐구일지

탐구 주제	모터 교체(Stepper motor)	일자	2022.08.19
탐구 내용 및 과정	Stepper motor로의 변화를 진행하였지만, 힘을 버티지 못해 보다 더 좋은 dc 모터를 이용하였다.		

## 첨부 자료



탐구 결과	
추후 계획	

## 탐구일지

# 탐구일지

탐구 주제	반동주기	일자	2022.08.26			
탐구 내용 및 과정	전체적인 제어를 완성하고, 성능 테스트를 위하여 상자에 대한 실험과 도르래에 대한 실험 세트에 줄줄이를 세팅함. 반동 코드를 업로드하여 하이퍼 파라미터의 범위를 실험적으로 파악함.					
첨부 자료						
						
						
탐구 결과	고무판 상황에서 줄줄이는 약 40도 까지도 꽈나 안정적으로 베티는 것을 관찰할 수 있었음 그러나 반동을 주는 각도(2~10도)를 달리함에 따라 반동을 줄 수 있는 시간 역시 달라졌음. (미끄러지지 않음~180ms~ 100ms)					
추후 계획						

# 탐구일지

탐구 주제	줄줄이 채 설계 및 90도 유지 모델 제작	일자	2022.10.10
탐구 내용 및 과정	python을 이용하여 serial으로부터 set된 theta값과 실질적인 theta 값을 받아들여 이를 그래프로 표현해 PID 값을 조정함.		

## 첨부 자료



```
// --- PID Setup ---  
//  
double originalSetpoint;  
double currentSetpoint = originalSetpoint;  
double movingAngleOffset = 0.1;  
double kp = 75, fd = 200, id = 0.8; //Front Kpid  
double bkp = 75, bkd = 200, bid = 0.6; //Back Kpid
```

## 탐구 결과

## 추후 계획

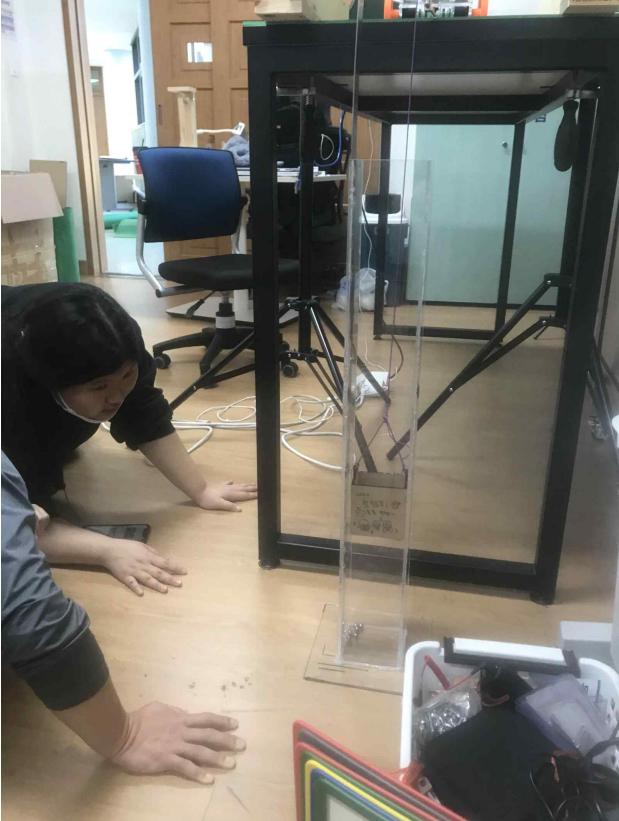
# 탐구일지

탐구 주제	박스 끌어오기 작업 및 실험 설계	일자	2022.10.12
탐구 내용 및 과정	새로 제작한 줄줄이를 통해 먼저 안정 평형을 이루기 쉬운 박스 실험 세트를 진행하였다. 세타_zero를 45도로 설정한 상태에서 박스가 끌리지 않는 한계 무게까지 박스 질량을 크게 하였다. 세타_b를 37도, t_b를 150ms로 설정하여 총 10회 반복실험하였다.		
첨부 자료			
탐구 결과	10회 반복 실험 중 못 끌어옴 1회, 잘 끌어옴 7회, 도중에 미끄러져 넘어짐 2회를 기록함. 해당 결과를 관찰한 뒤, 세타_zero와 세타_b를 각각 3도 씩 줄여 실험한 결과 못 끌어옴 0회, 잘 끌어옴 4회, 미끄러짐 6회를 기록하였다.		
추후 계획			

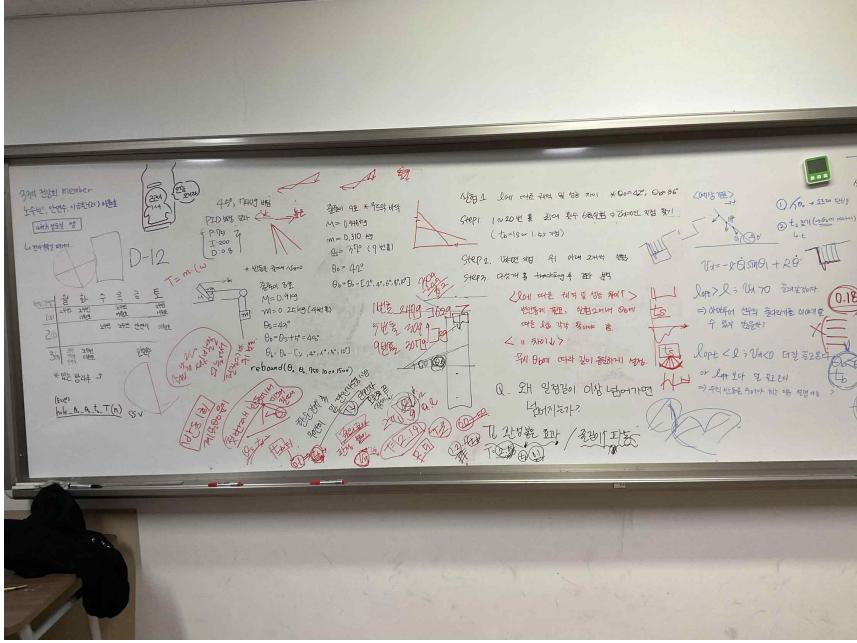
## 탐구일지

탐구 주제	도르래에서의 줄줄이 적용 및 실험 구체화	일자	2022.10.13			
탐구 내용 및 과정	<p>도르래 상황에 대해서는 자칫 불안정 평형 상황을 야기할 수 있기 때문에, 상자의 질량을 더욱 간편하고 미세하게 조정할 필요성이 있다. 이를 뚜껑 없는 상자를 실에 매달고 다양한 크기의 쇠구슬을 이용하기로 하였다. 도르래를 줄줄이의 줄이 묶인 부분까지 옮겨주기 위하여 도르래용 나무 단을 제작하기로 하였다.</p>					
첨부 자료						
						
탐구 결과	<p>높은 스탠딩 책상을 실험실에 설치하고, 규격에 맞게 고무매트를 재단하여 옮겨놓게 되었다. MDF를 이용하여 도르래 무게 추 상자를 제작하여 끈에 매달았다. 소량의 목재를 이용하여 도르래 나무 단을 제작후 스탠딩 책상에 클램프로 고정하였다.</p>					
추후 계획						

# 탐구일지

탐구 주제	1차 도르래 실험 진행	일자	2022.10.14
탐구 내용 및 과정	<p>앞서 제작한 도르래 실험세트를 이용하여, 본 실험 전 예비 실험의 양상으로 진행하였다. 각도를 유지하는 코드를 통해 미끄러지는 각도(세타_s)를 찾는다. 각각의 각도에 대해 평형을 유지하는 무게를 찾는다. 반동을 주어 각 각도별 미끄러져 넘어지는 반동 시간(t_s)를 찾는다.</p>		
첨부 자료			
			
탐구 결과	<p>세타_s = 37도 ts (10ms 간격) 32: 130ms 34: 130ms 36: 160ms 38: 180ms 40: 미끄러지지 않지만 시간을 늘릴수록 오히려 앞으로 끌려감 new!30: 110ms</p>		
추후 계획			

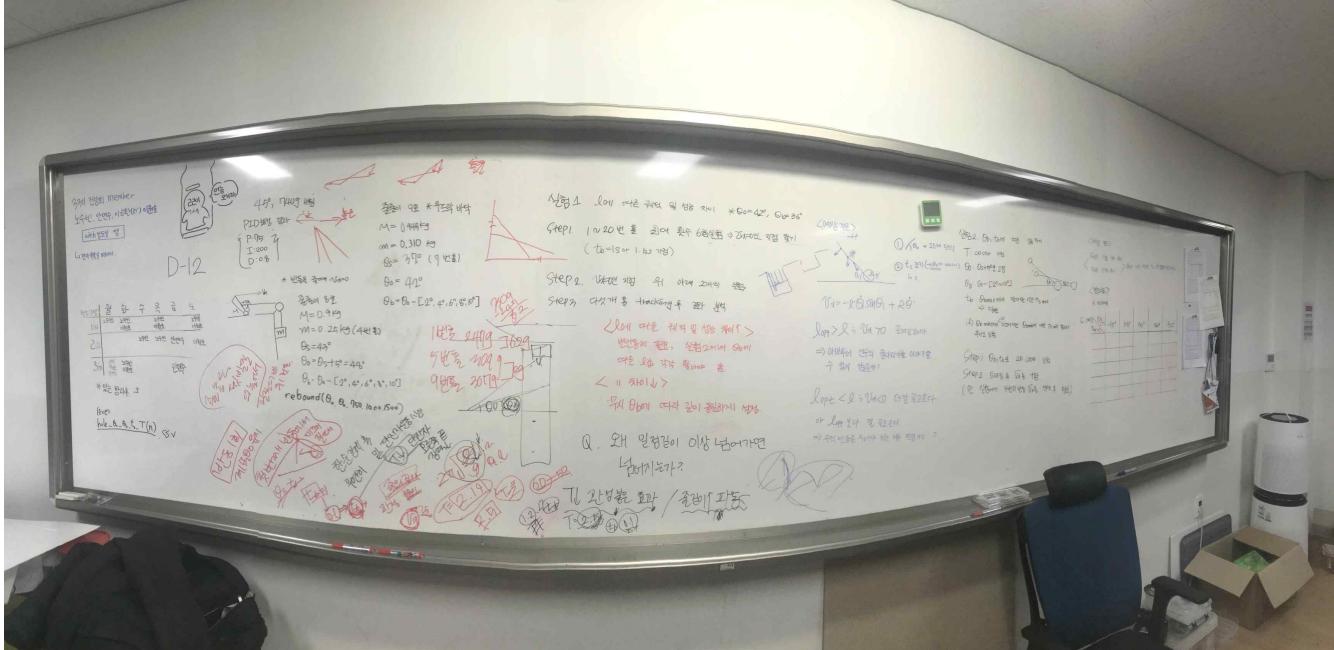
# 탐구일지

탐구 주제	도르래 실험의 진행	일자	2022.10.15
탐구 내용 및 과정	<p>(실험 1) 1에 따른 궤적 및 성능 차이</p> <p>1) 1번, 5번, 9번 홀에서 각각 3번씩 실험하여 <math>v_x</math>가 0인 지점을 찾는다.</p> <p>2) 이때의 상태는 <math>\theta_s = 37^\circ</math>, <math>\theta_b = 36^\circ</math>, <math>\theta_0 = 42^\circ</math>, <math>t_b = 250(ms)</math>, <math>T = 10500(ms)</math>으로 한다.</p> <p>3) 이후 실험을 영상으로 촬영하여 tracking을 진행하여 v-t, x-y 그래프를 얻어 수직으로 줄을 내리는 홀을 찾아 1에 따른 궤적 및 성능 차이를 확인한다.</p>		
첨부 자료			
			
탐구 결과	<p>x방향의 변위가 0인 최적의 홀이 존재하고, 그 위 아래로 다른 궤적의 모습이 나타날 것이라고 생각하였지만, 1번 홀에 가까울수록 수직에 가까운 움직임을 보이고, 모든 홀이 비슷한 이동 양상을 보였다. 따라서 줄줄이의 홀 위치는 1번 홀로 정하고자 하였다.</p>		
추후 계획			

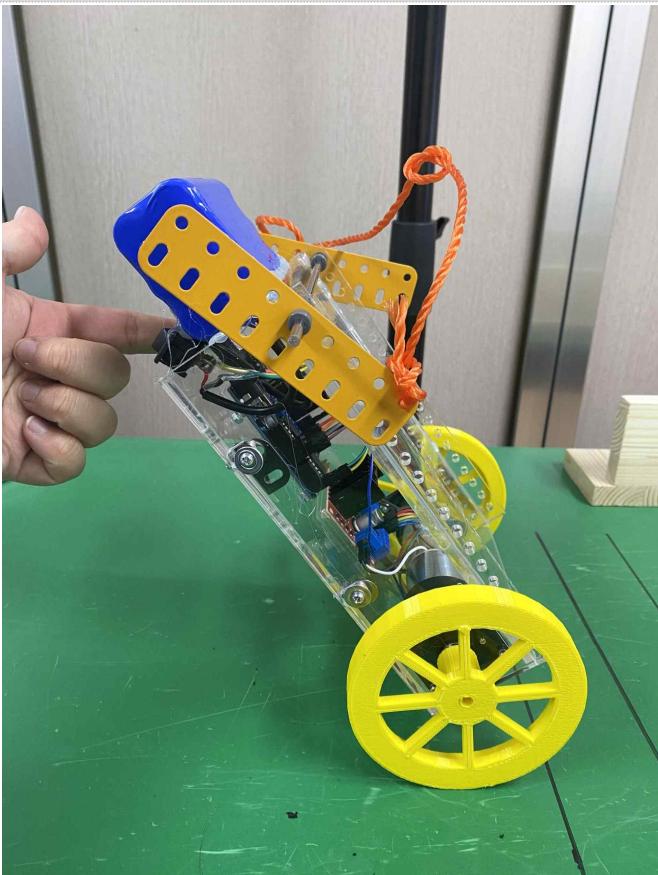
# 탐구일지

탐구 주제	일자																																			
	2022.10.18																																			
<p>(실험 2) <math>\theta_b</math>, <math>t_b</math>에 따른 성능 차이</p> <ol style="list-style-type: none"> <li>1) <math>T = 10500(ms)</math>, <math>\theta_0 = 42^\circ</math>, <math>\theta_b = 40^\circ, 38^\circ, 36^\circ, 34^\circ, 32^\circ</math>, <math>t_b : \theta_{b,\max}</math>에서 미끄러지는 시간인 <math>t_{s,\max}</math>을 5등분한다고 하자</li> <li>2) <math>\theta_b</math>에 따른 <math>t_s</math>를 찾는다.</li> <li>3) 2)에 과정에서 serial을 통해 각도 값에 대한 csv 파일을 받아 분석을 진행한다.</li> <li>4) 구한 <math>t_s</math>를 5등분하여 <math>\theta_b</math>와 <math>t_b</math>값에 따른 반동을 3번 주었을 때의 <math>\bar{v}_x</math>값을 3번 구해 평균을 낸다.</li> <li>5) 4)의 데이터를 정리해 분석을 진행한다.</li> </ol>																																				
첨부 자료																																				
<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th><math>\theta_b</math></th> <th><math>T</math></th> <th><math>t_s</math></th> <th><math>v_x</math></th> <th><math>X</math></th> </tr> </thead> <tbody> <tr> <td>38</td> <td>10500</td> <td>~</td> <td>~</td> <td>~</td> </tr> <tr> <td>36</td> <td>10500</td> <td>~</td> <td>~</td> <td>~</td> </tr> <tr> <td>34</td> <td>10500</td> <td>~</td> <td>~</td> <td>~</td> </tr> <tr> <td>32</td> <td>10500</td> <td>~</td> <td>~</td> <td>~</td> </tr> <tr> <td>30</td> <td>10500</td> <td>~</td> <td>~</td> <td>~</td> </tr> <tr> <td>28</td> <td>10500</td> <td>~</td> <td>~</td> <td>~</td> </tr> </tbody> </table>		$\theta_b$	$T$	$t_s$	$v_x$	$X$	38	10500	~	~	~	36	10500	~	~	~	34	10500	~	~	~	32	10500	~	~	~	30	10500	~	~	~	28	10500	~	~	~
$\theta_b$	$T$	$t_s$	$v_x$	$X$																																
38	10500	~	~	~																																
36	10500	~	~	~																																
34	10500	~	~	~																																
32	10500	~	~	~																																
30	10500	~	~	~																																
28	10500	~	~	~																																
탐구 결과																																				
추후 계획																																				

# 탐구일지

탐구 주제	도르래 반동전략 실험	일자	2022.10.19			
탐구 내용 및 과정	<p>1. 세타_zero = 42, 세타_b = 38~28 (급간 2), t_b = 36~180(급간 36)</p> <p>2. 42도에 대한 안정적인 질량을 맞추어 도르래에 견다.</p> <p>3. 두 가지 변수 값에 대하여 5*6의 영상을 촬영한다. (한 케이스당 3회 반복)</p> <p>4. 저배속 상황에서 한 번 반동 시 이동한 거리(단위 0.1mm)를 기록한다.</p> <p>5. 반동 주기 5000ms로 나누어 속도를 구하여 나타낸다.</p>					
첨부 자료						
						
탐구 결과	<p>시간이 클수록, 각도가 클수록 더 많이 끌고 온다는 일관성은 확인 할 수 있으나, 각 자료마다 편차가 데이터의 <math>\pm 50\%</math> 수준으로 신뢰하기 어려운 수준의 데이터가 산출되었다.</p>					
추후 계획						

# 탐구일지

탐구 주제	줄의 영향을 줄인 재실험	일자	2022.10.20
탐구 내용 및 과정	실험 도중 전원선의 영향이 생각보다 편차를 크게 만든다고 생각하여, 프레임에 전원 선을 고정하고, 조절하여 재실험을 진행하였다. 또한 한 번 반동에 대한 속도에서, 25cm를 주파하는 데 걸리는 시간을 측정하여 속도를 구하고자 하였다.		
첨부 자료			
			
탐구 결과	각도가 일정한 상황에서는 미끄러지지 않는 수준에서 가능한 반동시간이 길수록 효과적이었고, 반동시간이 일정한 경우, 늙는 각도가 높을수록 높아지다가 일정 각도 이후에서부터는 다시 감소하였다. 즉, 효과가 가장 좋게 나타나는 특정각도가 있을 것이라는 결론을 내렸다.		
추후 계획			