시그모이드 함수의 형태를 보면 완만한 기울기로 시작해서 기울기가 증가했다가 다시 감소하며 완만하게 끝난다. 서보 모터의 회전 동작이 부드럽게 시작해서 끝나게 하기 위해서 시그모이드 함수 형태로 제어를 하는 것 같다.

차가 올 때 빠르게 차단기를 올려주지만 부드럽게 동작하는 것 처럼 보이게 하는 것이 나을 것 같아 시작 가속도는 높게 시작했다가 멈출 때 부드럽게 멈추는 함수를 찾아보았다. Ease-out Quantic이라는 수식이 나왔는데 Ease-out은 처음에는 빨리 갈수록 서서히 멈추는 움직임을 말한다. Ease-out 수식에는 Quantic만 있는 것이 아니라 시그모이드를 포함해 대표적으로 3가지가 있다고 한다. 끝점에서 속도 가속도가 0이 되는 Quantic 끝에서 속도는 0이지만 가속도는 0이 아니라 Quantic 보다는 조금 거칠어 보일 수 있는 Cubic 그리고 시작점과 끝점에서 다 부드럽게 진행한다.

설명을 다읽어보면 내 의도와 제일 부합하는 것이 역시 Quantic이었기 때문에 시그모이드와 Quantic으로 구현을 해 보았다.

코드 구조는 이전에 거리 측정할 때 사용했던 코드를 그대로 사용하고 조건에 맞춰서 차단기를 올리고 내리는 방식으로 짰다.

```
if (!car && mid <= TH_NEAR) {
  car = true;
  sigmoid(true);
}
if (car && (mid >= TH_FAR)) {
  car = false;
  sigmoid(false);
}
```

위의 거리 측정 부분은 이전에 사용했던 중위값 필터 코드를 그대로 사용하였다. Car 조건은 현재 거리 측정 값이 차가 있을 때 측정값인지 아닌지를 판단하는 조건이고 차가 없었던 상황에서 측정 거리가 일정 이상 줄어들면 차가 온 것으로 판단하고 차단기를 올리는 함수를 호출한다. 차가 있었던 상황에서 측정 거리가 최대값 이상으로 튀면 차가 지나간 것으로 판단하고 차단기를 내리는 함수를 호출한다.

```
void sigmoid(bool up) {
  // 시작/목표 각도 설정 (현재 위치에서 시작 → 점프 방지)
  int fromAngle = myServo.read();
  int toAngle   = up ? STOP_ANGLE : START_ANGLE;

  // 이미 목표면 바로 종료
```

```
  if (fromAngle == toAngle) {
    myServo.write(toAngle);
    return;
  }
```

시그모이드 함수이다. Up이 true이면 올라가고 false이면 내려간다. STOP_ANGLE과 START_ANGLE은 원하는 서보의 각도로 미리 설정해준다.

```
  while (true) {
    unsigned long elapsed = millis() - t0;
    if (elapsed >= dur) {
      myServo.write(toAngle);     // 최종 위치 고정
      break;
    }

    // 0~1 정규화된 시간과 시그모이드
    float x = (float)elapsed / (float)dur;                          // 0..1
    float s = 1.0f / (1.0f + expf(-(x - 0.5f) * 2.0f * SIGMOID_K)); // 0→1

    // 보간 후 출력
    int angle = (int)lroundf(fromAngle + (toAngle - fromAngle) * s);
    myServo.write(constrain(angle, 0, 180));

    delay(5);   // 업데이트 간격(더 부드럽게 보이면 3~10ms 사이로 조절)
  }
```

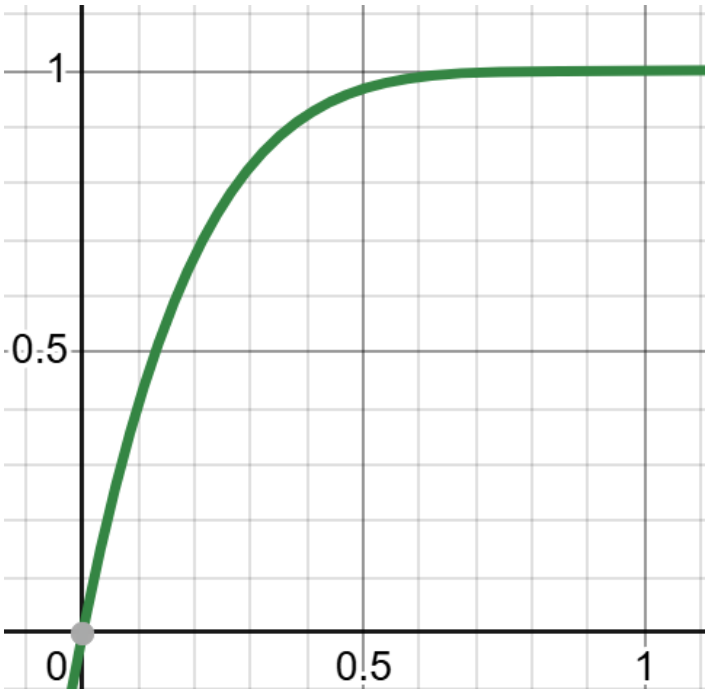
서보의 동작을 시그모이드 형태로 제어하는 부분이다.

s는 진행 시간에 따른 시그모이드 함수에서 위치이고 그 s값으로 각도를 보정해서 부드럽게 움직일 수 있도록 한다.

```
float s = (float)t / (float)T;    // 0..1
    float u = 1.0f - s;
    float p = 1.0f - u*u*u*u*u;

    int angle = (int)(fromAngle + (toAngle - fromAngle) * p + 0.5f);
    myServo.write(constrain(angle, 0, 180));
```

Quantic으로 제어되도록 하려면 위 코드에서 s값을 정하는 부분과 각도를 보정하는 부분만 그에 맞게 수정해주면 된다.



Quantic 함수의 그래프 형태는 다음과 같고 0부터 1까지 진행하므로 끝점에서 부드럽게 멈출 수 있다.