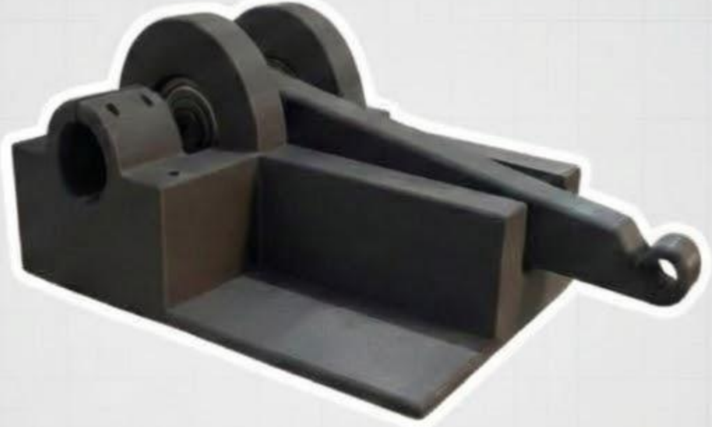
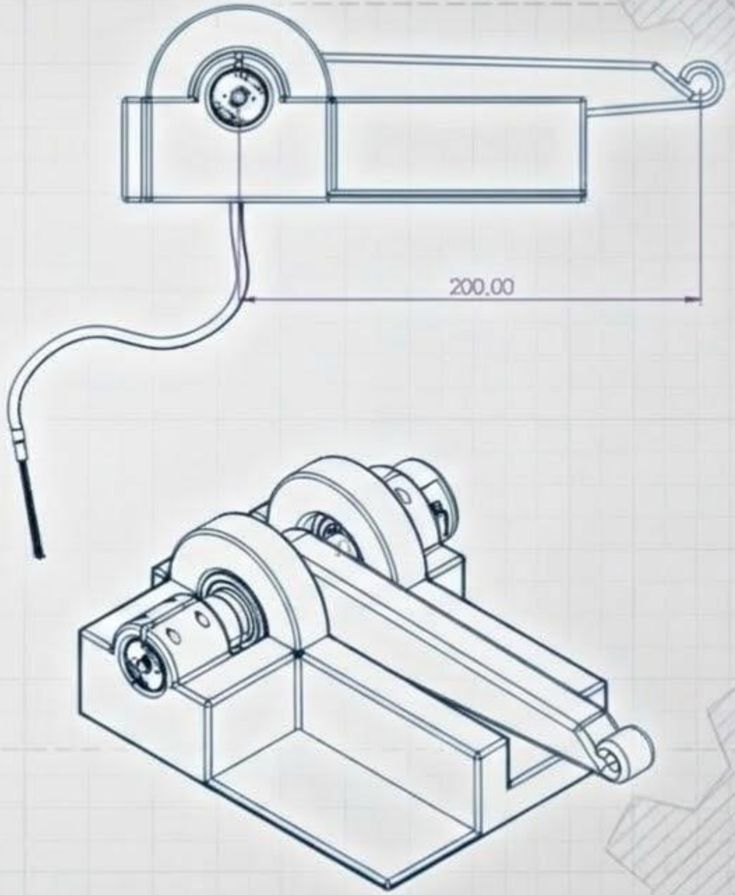
# ACTUATED ROBOT REVOLUTE JOINT(SINGLE AXIS)
# MEE311 MECHANISM DESIGN I – FALL 2025

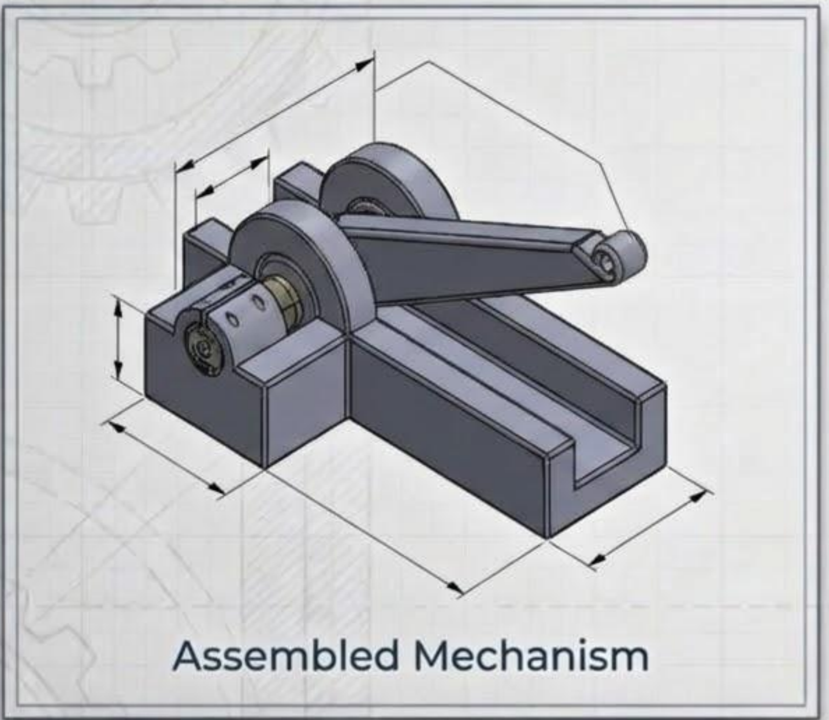


Efe Tuna Çalışkan 200412004
Nuri Kaan Gençtürk 210412019
İsmet Akalın 210412040
Deniz Tuna Yasemin 210412053
Beyza Demir 210412063

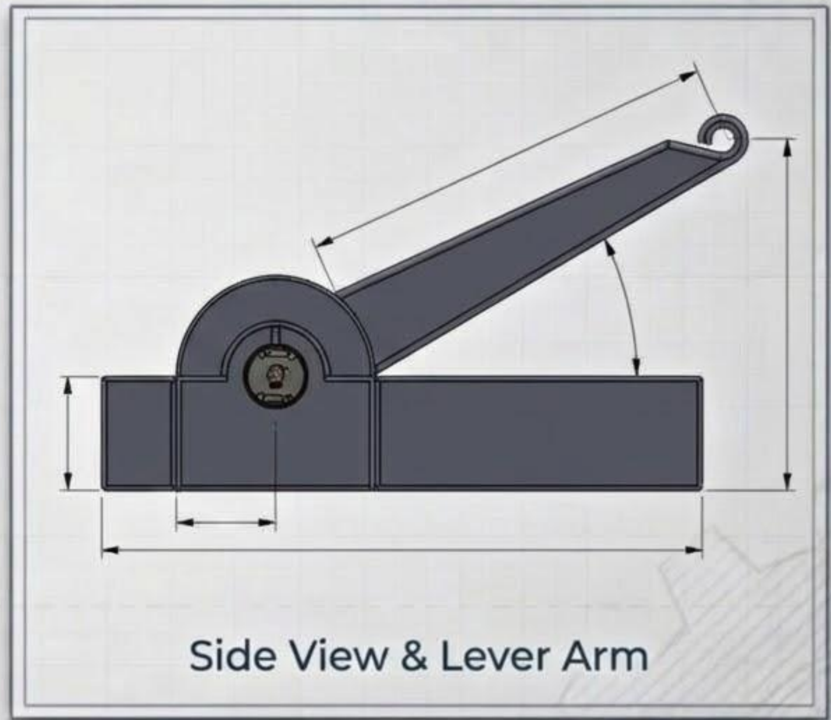
# DESIGN AND DEVELOPMENT MECHANISM



# PROJECT DEVELOPMENT

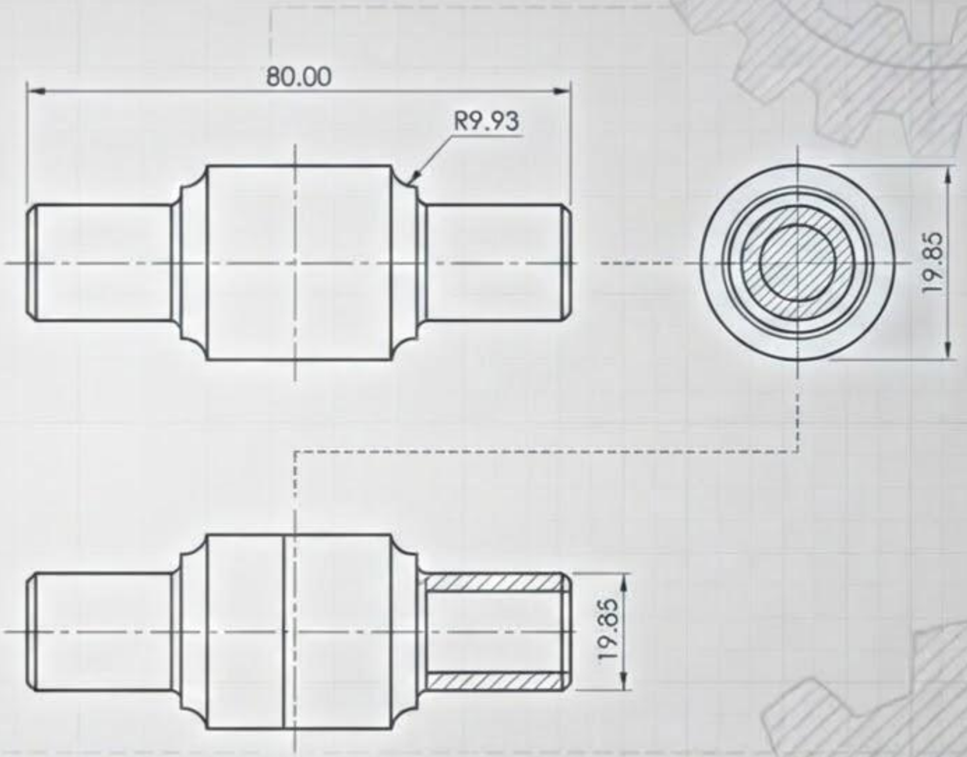


Assembled Mechanism



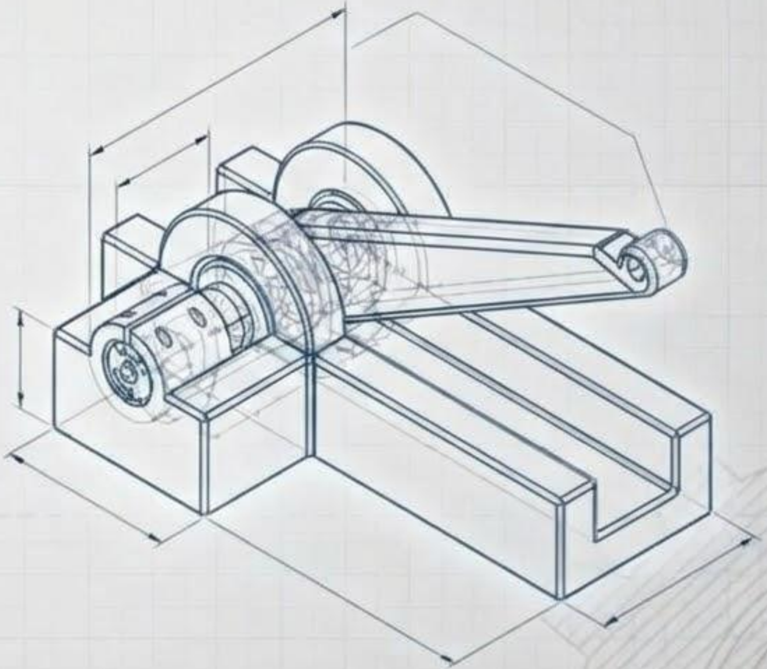
Side View & Lever Arm

# DESIGN AND DEVELOPMENT (SHAFT)

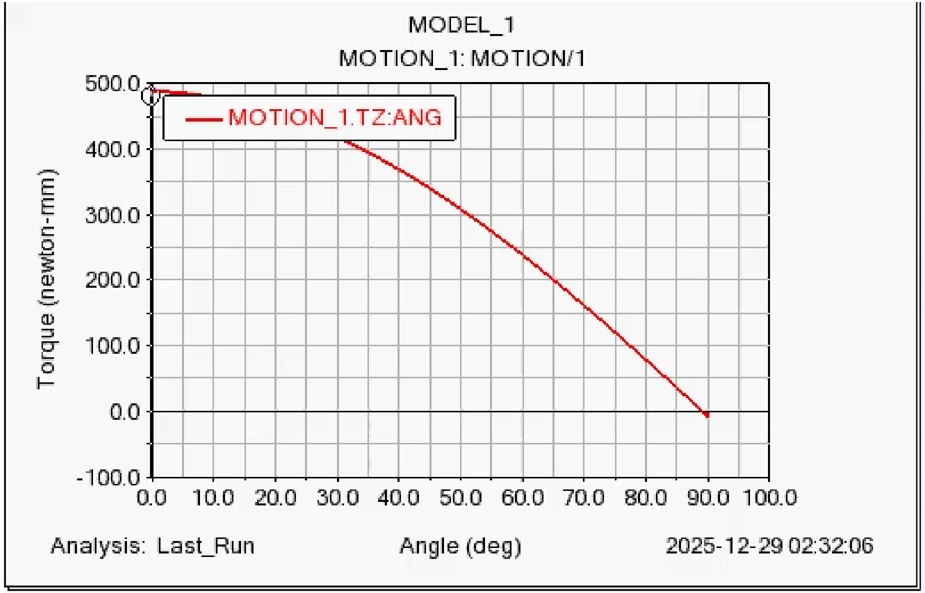


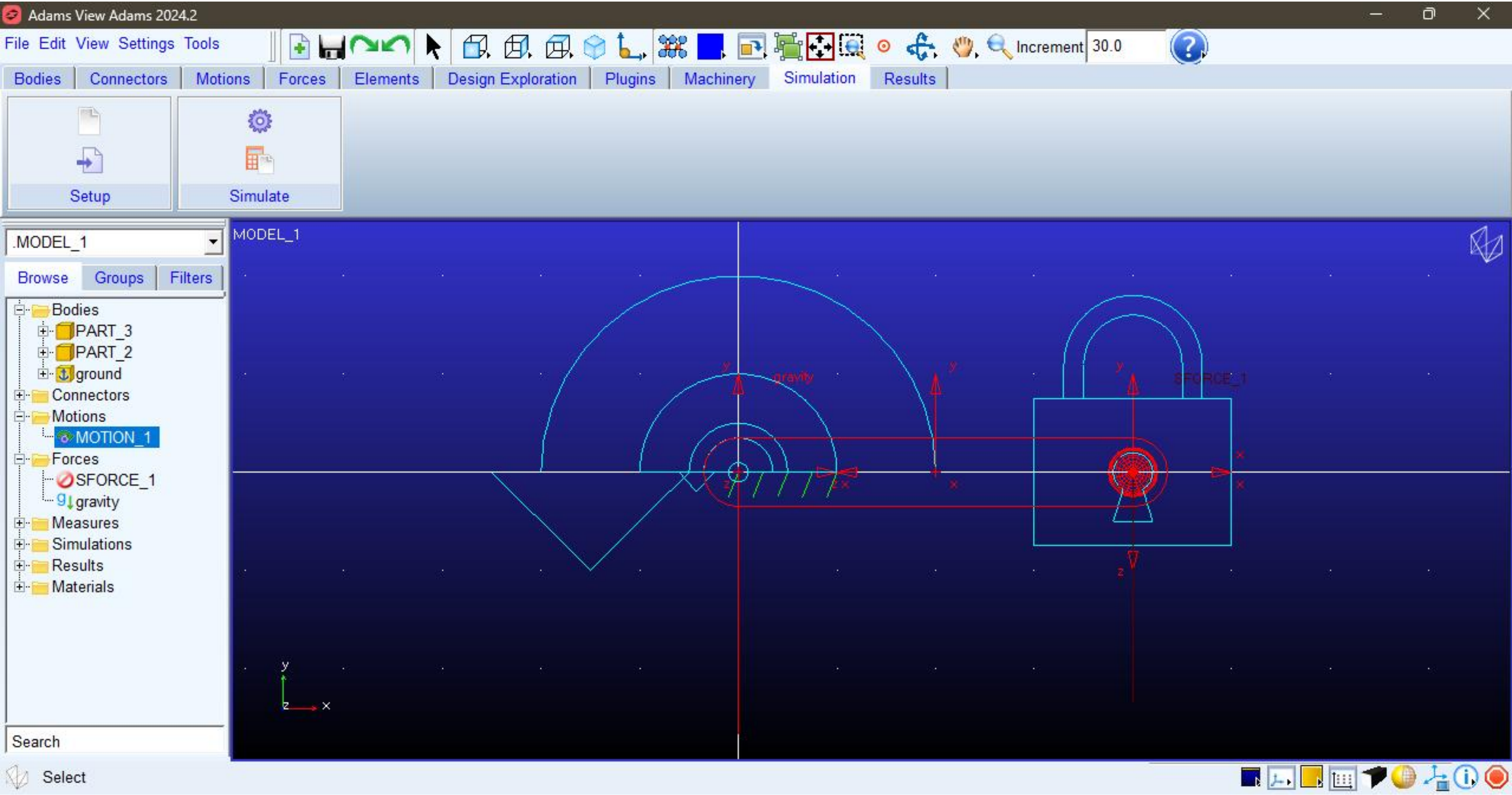
# TORQUE REQUIREMENT & MOTOR SELECTION

- $\tau_{\_req} = m*g*l \approx 4.0$ kg·cm
- Motor: 5 kg·cm

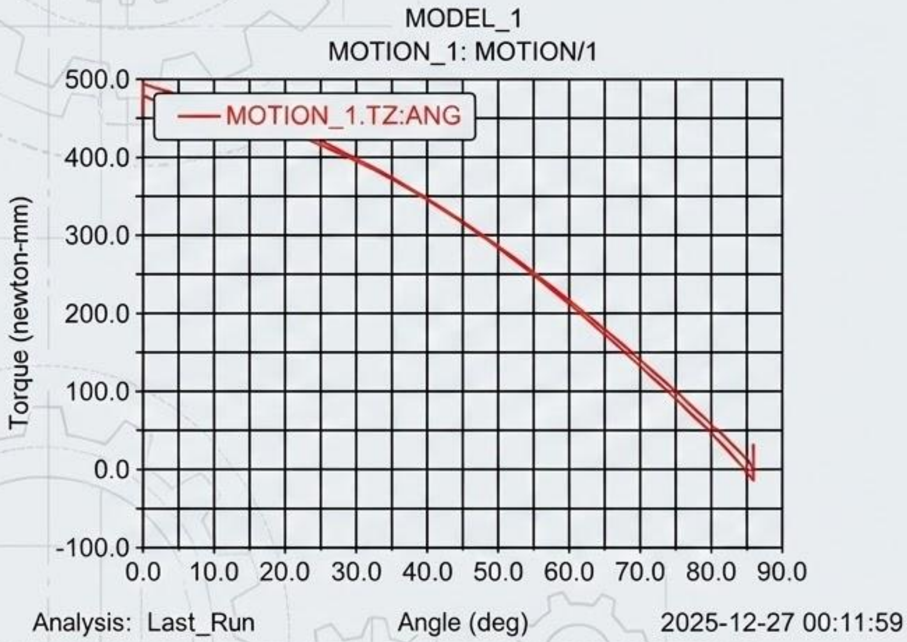


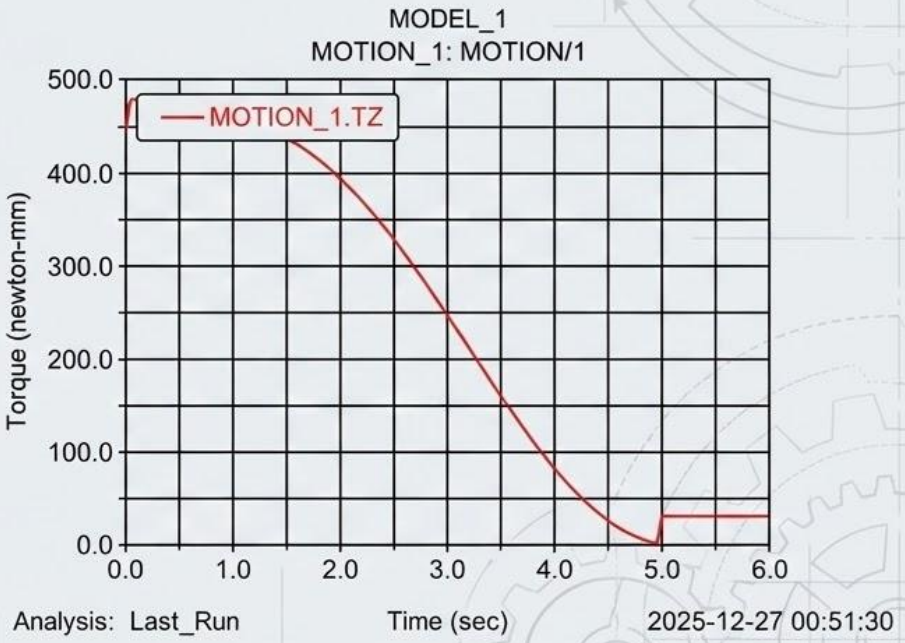
# Dynamic Simulation (Adams View)




Adams View Adams 2024.2
File
Edit
View
Settings
Tools
Increment
30.0
Bodies
Connectors
Motions
Forces
Elements
Design Exploration
Plugins
Machinery
Simulation
Results
Setup
Simulate
.MODEL_1
Browse
Groups
Filters
Bodies
PART_3
PART_2
ground
Connectors
Motions
MOTION_1
Forces
SFORCE_1
gravity
Measures
Simulations
Results
Materials
Search
Select
MODEL_1
gravity
SFORCE_1
y
x
z

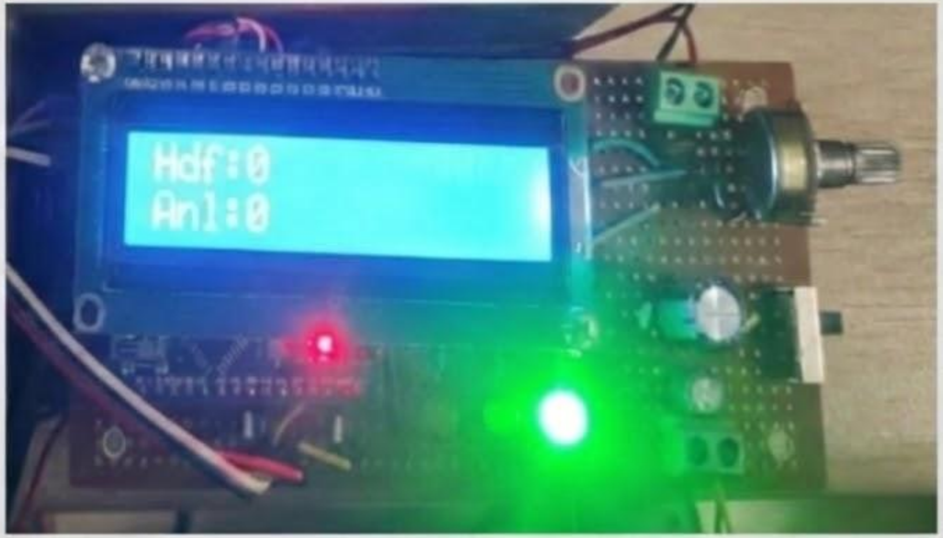
# Simulation Results – Torque Response





# ELECTRONIC PART

Assembled Electronic Circuit

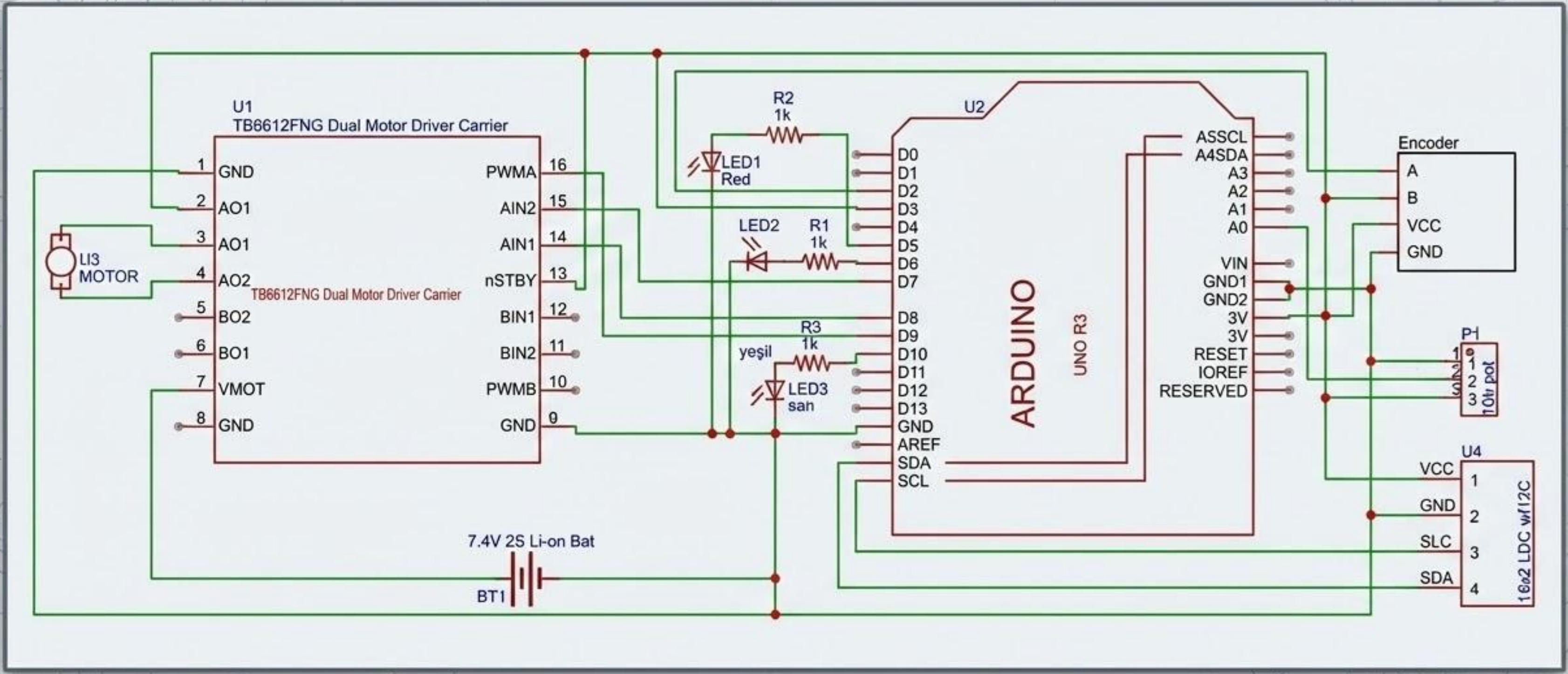


# COMPONENTS

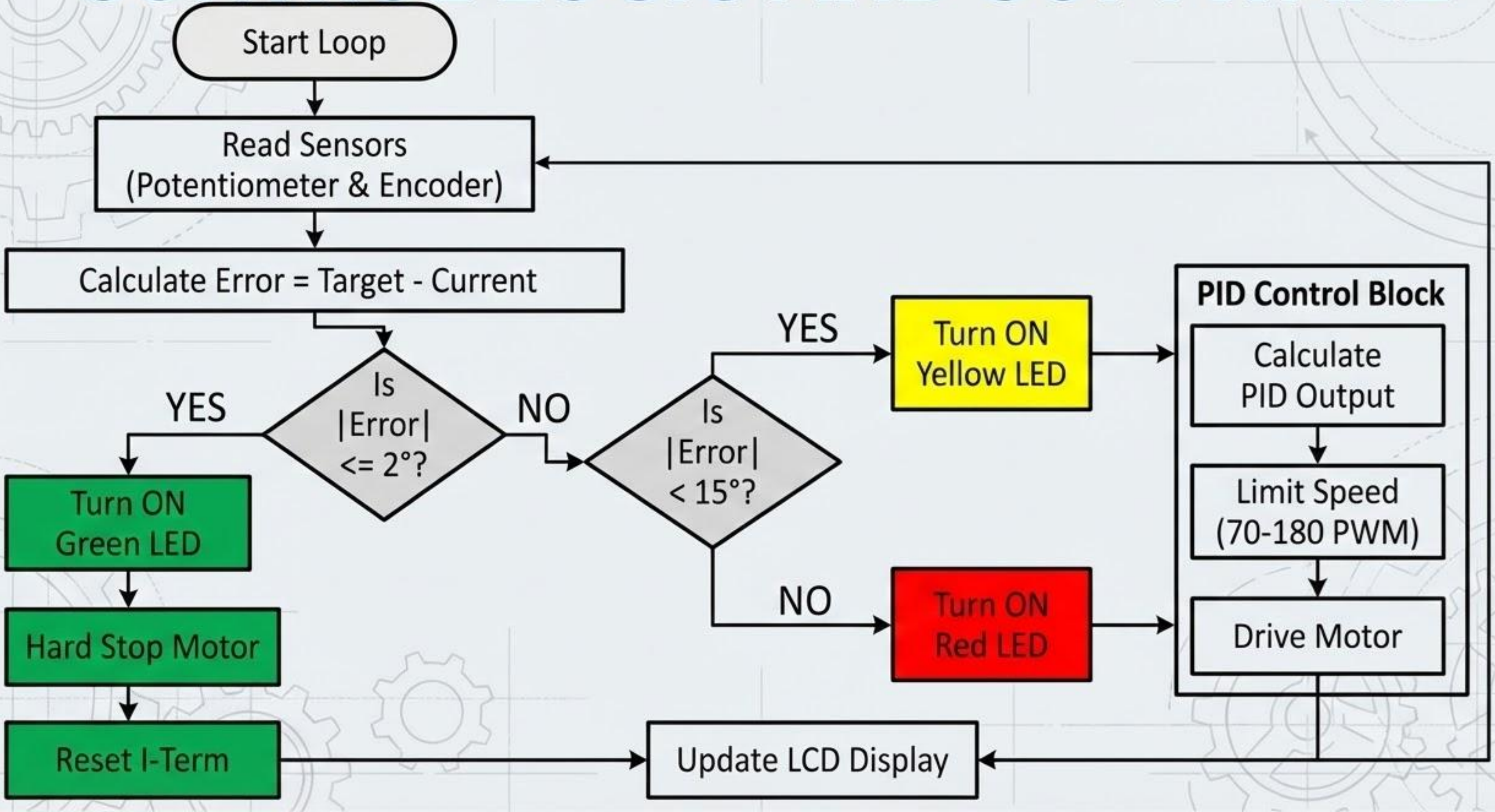
Incremental Optical Rotary Encoder

Geared DC Motor

# ELECTRONIC PART (SCHEMATIC)



# CONTROL LOGIC AND SOFTWARE



Start Loop

Read Sensors (Potentiometer & Encoder)

Calculate Error = Target - Current

Is |Error| <= 2°?

YES

Turn ON Green LED

Hard Stop Motor

Reset I-Term

NO

Is |Error| < 15°?

YES

Turn ON Yellow LED

NO

Turn ON Red LED

PID Control Block

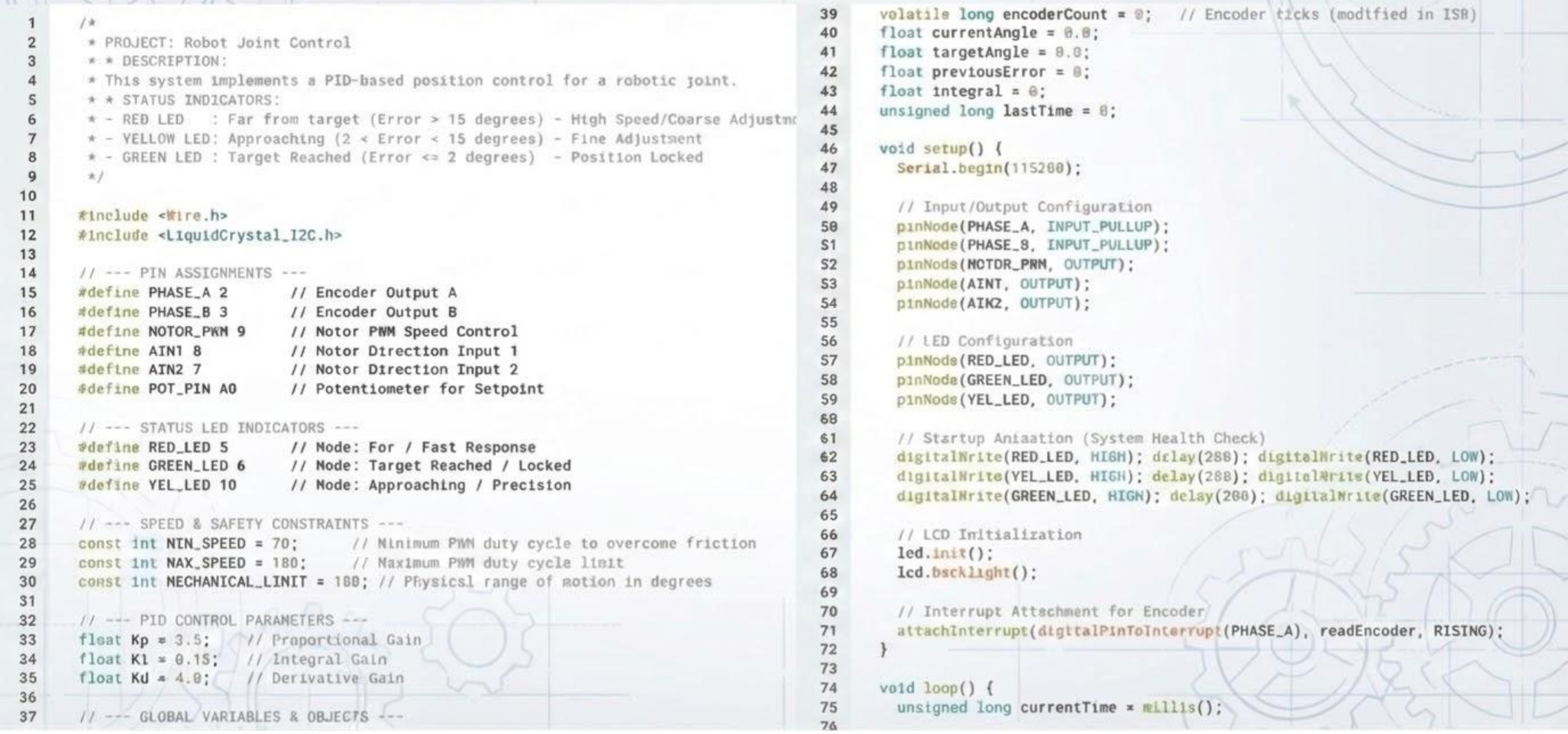
Calculate PID Output

Limit Speed (70-180 PWM)

Drive Motor

Update LCD Display

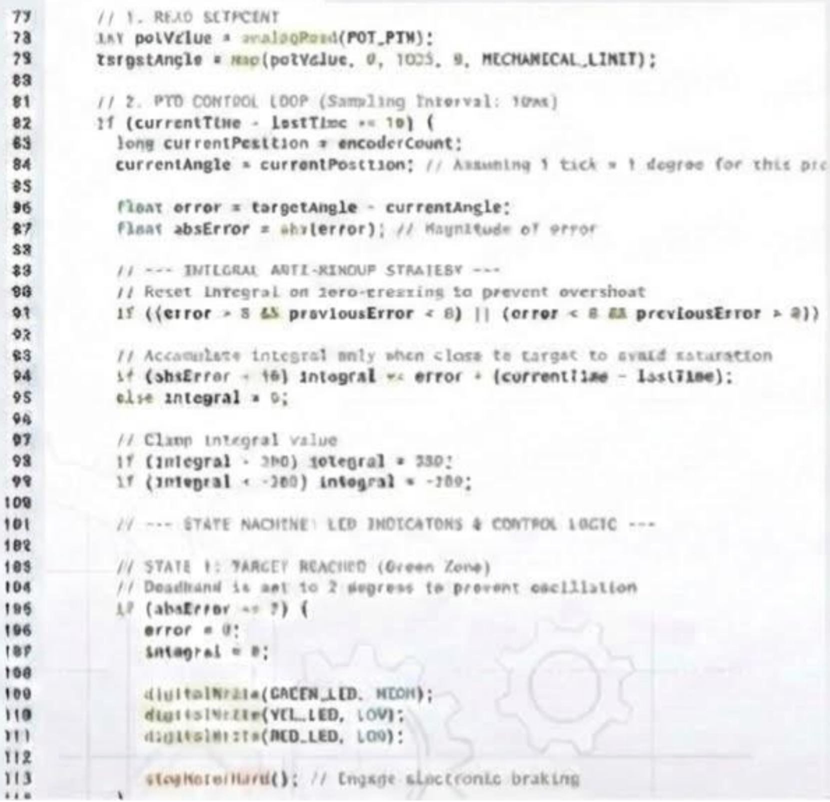
# MECHANISM CODE



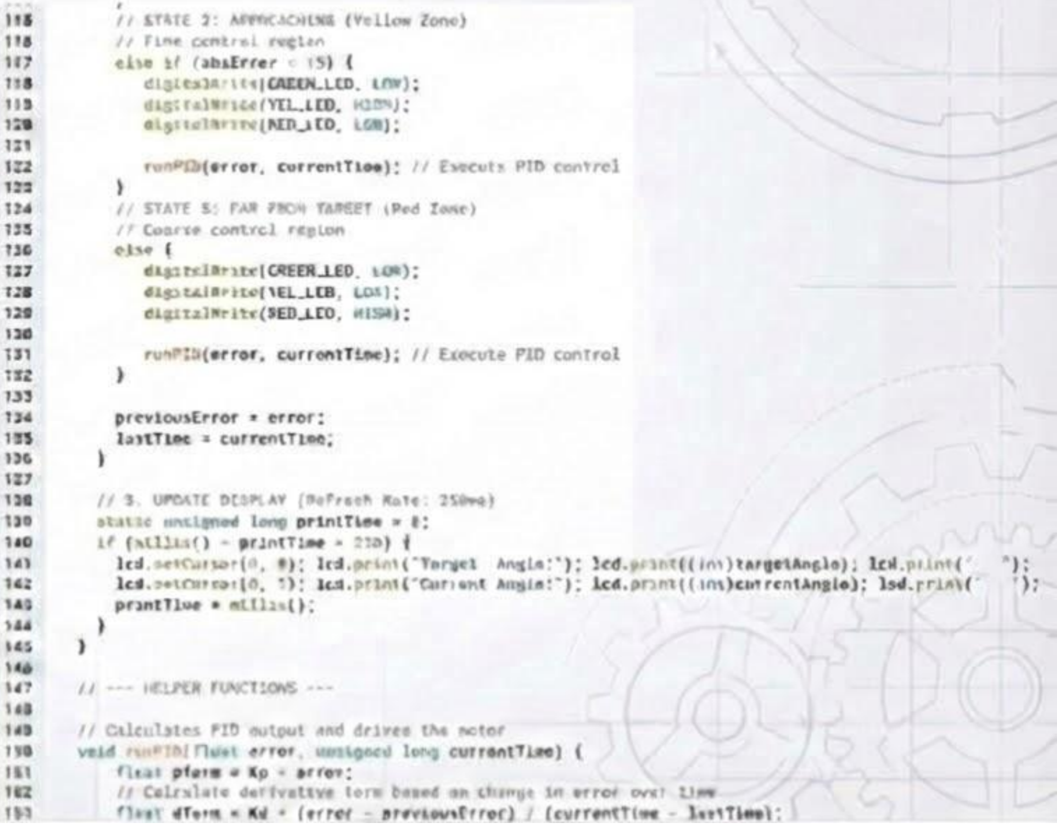
```
/*
 * PROJECT: Robot Joint Control
 * * DESCRIPTION:
 * This system implements a PID-based position control for a robotic joint.
 * * STATUS INDICATORS:
 * - RED LED   : Far from target (Error > 15 degrees) - High Speed/Coarse Adjustm
 * - YELLOW LED: Approaching (2 < Error < 15 degrees) - Fine Adjustment
 * - GREEN LED : Target Reached (Error <= 2 degrees)  - Position Locked
 */

#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// --- PIN ASSIGNMENTS ---
#define PHASE_A 2        // Encoder Output A
#define PHASE_B 3        // Encoder Output B
#define NOTOR_PKM 9      // Notor PWM Speed Control
#define AIN1 8           // Notor Direction Input 1
#define AIN2 7           // Notor Direction Input 2
#define POT_PIN A0       // Potentiometer for Setpoint

// --- STATUS LED INDICATORS ---
#define RED_LED 5        // Node: For / Fast Response
#define GREEN_LED 6      // Node: Target Reached / Locked
#define YEL_LED 10       // Node: Approaching / Precision

// --- SPEED & SAFETY CONSTRAINTS ---
const int NIN_SPEED = 70;      // Minimum PWM duty cycle to overcome friction
const int NAX_SPEED = 180;     // Maximum PWM duty cycle limit
const int NECHANICAL_LINIT = 180; // Physicsl range of motion in degrees

// --- PID CONTROL PARAMETERS ---
float Kp = 3.5;    // Proportional Gain
float Ki = 0.15;   // Integral Gain
float Kd = 4.0;    // Derivative Gain

// --- GLOBAL VARIABLES & OBJECTS ---
```

```
volatile long encoderCount = 0;   // Encoder ticks (modified in ISR)
float currentAngle = 0.0;
float targetAngle = 0.0;
float previousError = 0;
float integral = 0;
unsigned long lastTime = 0;

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

  // Input/Output Configuration
  pinNode(PHASE_A, INPUT_PULLUP);
  pinNode(PHASE_B, INPUT_PULLUP);
  pinNode(MOTOR_PRM, OUTPUT);
  pinNode(AINT, OUTPUT);
  pinNode(AIN2, OUTPUT);

  // LED Configuration
  pinNode(RED_LED, OUTPUT);
  pinNode(GREEN_LED, OUTPUT);
  pinNode(YEL_LED, OUTPUT);

  // Startup Aniaation (System Health Check)
  digitalNrite(RED_LED, HIGH); delay(200); digitalNrite(RED_LED, LOW);
  digitalNrite(YEL_LED, HIGH); delay(200); digitalNrite(YEL_LED, LOW);
  digitalNrite(GREEN_LED, HIGH); delay(200); digitalNrite(GREEN_LED, LOW);

  // LCD Initialization
  led.init();
  lcd.backlight();

  // Interrupt Attachment for Encoder
  attachInterrupt(digitalPinToInterrupt(PHASE_A), readEncoder, RISING);
}

void loop() {
  unsigned long currentTime = millis();
```

# MECHANISM CODE

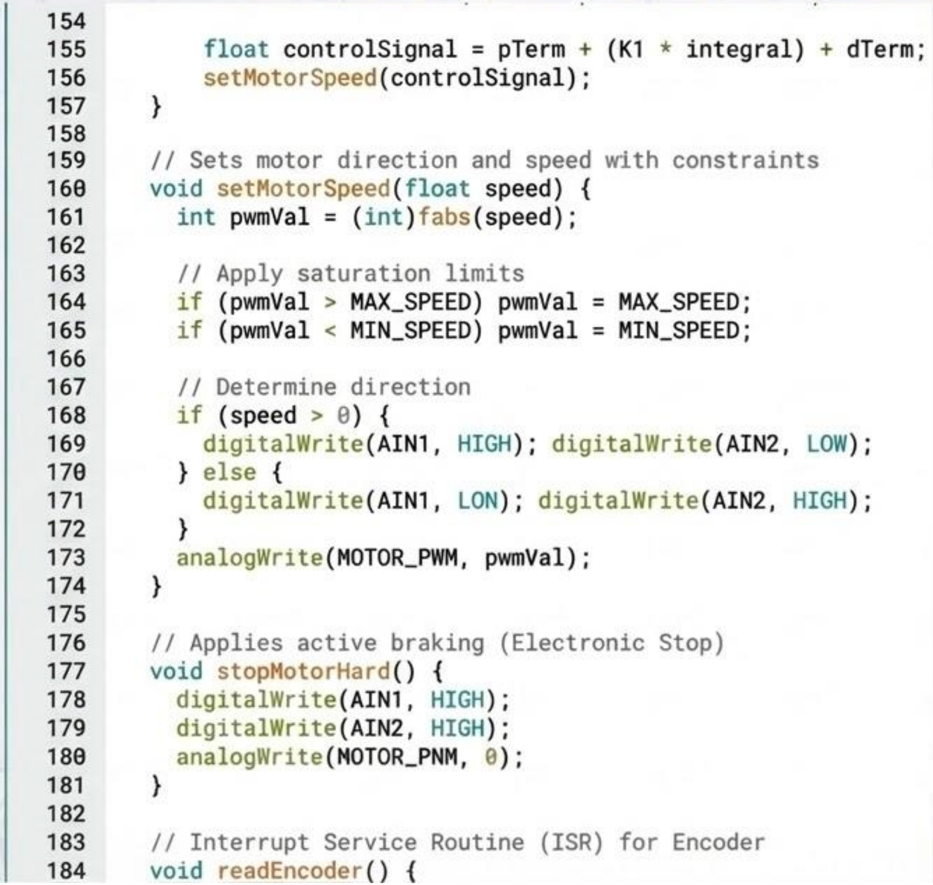


```
// 1. READ SETPCINT
iNY polVclue = analogRead(POT_PTN);
tsrgstAngle = map(polVclue, 0, 1025, 0, MECHANICAL_LINIT);

// 2. PTD CONTROL LOOP (Sampling Interval: 10ms)
if (currentTime - lastTime >= 10) (
  long currentPesition = encoderCount;
  currentAngle = currentPosition; // Assuming 1 tick = 1 degree for this prc

  float error = targetAngle - currentAngle;
  float absError = abs(error); // Magnitude of error

  // --- INTEGRAL ANTI-RINDUP STRATEBY ---
  // Reset integral on zero-creszing to prevent overshoat
  if ((error > 8 && praviousError < 8) || (orror < 8 && previousError > 8))

  // Accamulate integral only when close to target to avaid saturation
  if (absError < 16) integral += error * (currentTime - lastTime);
  else integral = 0;

  // Clamp integral value
  if (integral > 300) integral = 330;
  if (integral < -300) integral = -300;

  // --- STATE NACHINE: LED INDICATONS & CONTROL LOGIC ---

  // STATE 1: TARGET REACHED (Green Zone)
  // Deadband is set to 2 degress to prevent oscillation
  if (absError <= 2) {
    error = 0;
    integral = 0;

    digitalWrite(GRCEN_LED, HIGH);
    digitalWrite(YEL_LED, LOW);
    digitalWrite(RED_LED, LOW);

    stopMotorHard(); // Engage electronic braking
  }
```



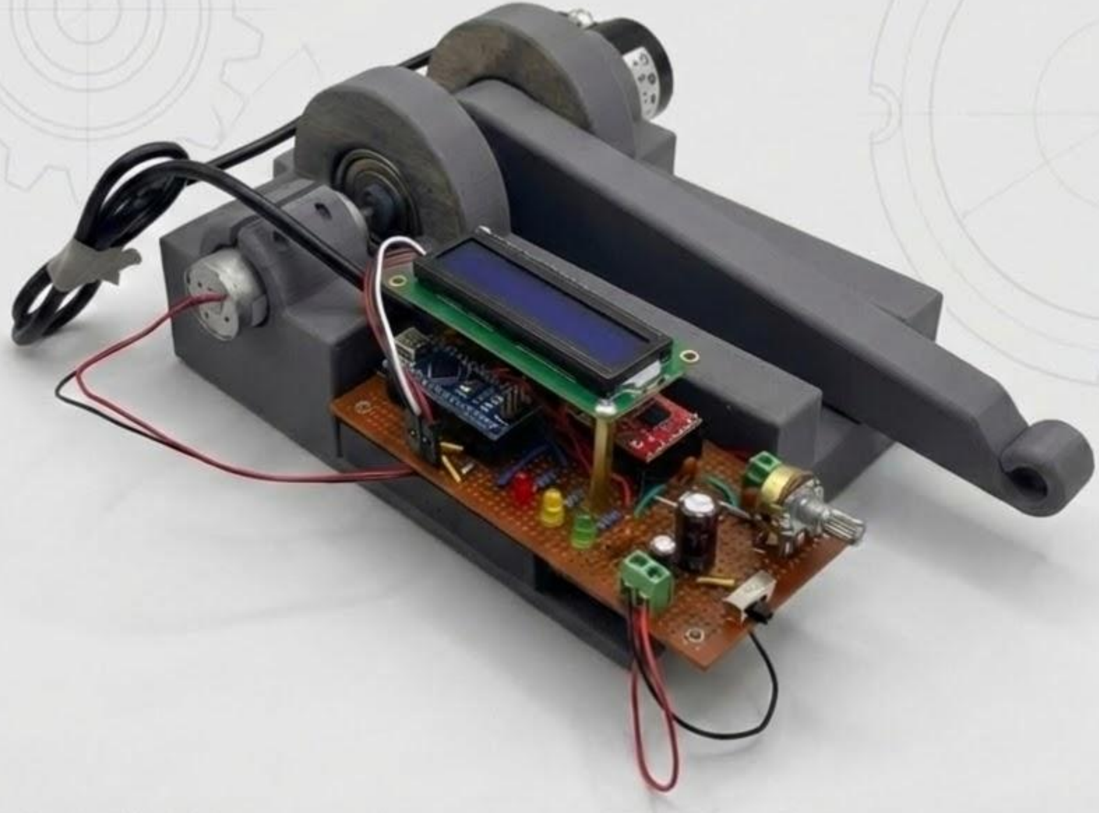
```
      // STATE 2: APPROACHING (Yellow Zone)
      // Fine control region
      else if (absErrer < 15) {
         digitalWrite(GREEN_LED, LOW);
         digitalWrite(YEL_LED, HIGH);
         digitalWrite(RED_LED, LOW);

         runPID(error, currentTime); // Executs PID control
      }
      // STATE 5: FAR FROM TARGET (Red Zone)
      // Coarse control region
      else {
         digitalWrite(GREER_LED, LOW);
         digitalWrite(YEL_LEB, LOW);
         digitalWrite(SED_LED, HIGH);

         runPID(error, currentTime); // Execute PID control
      }

      previousError = error;
      lastTime = currentTime;
    }

    // 3. UPDATE DISPLAY (Refresh Rate: 250ms)
    static unsigned long printTime = 0;
    if (millis() - printTime > 250) {
      lcd.setCursor(0, 0); lcd.print("Target  Angle:"); lcd.print((int)targetAngle); lcd.print("   ");
      lcd.setCursor(0, 1); lcd.print("Current Angle:"); lcd.print((int)currentAngle); lsd.print("   ");
      printTime = millis();
    }
  }

  // --- HELPER FUNCTIONS ---

  // Calculates PID output and drives the motor
  void runPID(float error, unsigned long currentTime) {
      float pTerm = Kp * error;
      // Calculate derivative term based on change in error over time
      float dTerm = Kd * (error - previousError) / (currentTime - lastTime);
```

# MECHANISM CODE



```
  float controlSignal = pTerm + (K1 * integral) + dTerm;
  setMotorSpeed(controlSignal);
}

// Sets motor direction and speed with constraints
void setMotorSpeed(float speed) {
 int pwmVal = (int)fabs(speed);

 // Apply saturation limits
 if (pwmVal > MAX_SPEED) pwmVal = MAX_SPEED;
 if (pwmVal < MIN_SPEED) pwmVal = MIN_SPEED;

 // Determine direction
 if (speed > 0) {
  digitalWrite(AIN1, HIGH); digitalWrite(AIN2, LOW);
 } else {
  digitalWrite(AIN1, LON); digitalWrite(AIN2, HIGH);
 }
 analogWrite(MOTOR_PWM, pwmVal);
}

// Applies active braking (Electronic Stop)
void stopMotorHard() {
 digitalWrite(AIN1, HIGH);
 digitalWrite(AIN2, HIGH);
 analogWrite(MOTOR_PNM, 0);
}

// Interrupt Service Routine (ISR) for Encoder
void readEncoder() {
```

# FINAL PROTOTYPE



# THANK YOU FOR LISTENING