ДОДАТОК А
КОД ПРОГРАМИ

```
#include "interpolation.h"
#include "config.h"
#include "queue.h"
#include "logger.h"

Interpolation::Interpolation(){
  pos_offset.xmm = 0.0;
  pos_offset.ymm = 0.0;
  pos_offset.zmm = 0.0;
  pos_offset.emm = 0.0;
}

//G92 POSITION OFFSET FUNCTIONS
void Interpolation::setPosOffset(float new_x, float new_y, float new_z, float new_e) {
  pos_offset.xmm = xPosmm - new_x;
  pos_offset.ymm = yPosmm - new_y;
  pos_offset.zmm = zPosmm - new_z;
  pos_offset.emm = ePosmm - new_e;
  Logger::logINFO("POSITION OFFSET: [X" + String(pos_offset.xmm) + " Y:" +
String(pos_offset.ymm) + " Z:" + String(pos_offset.zmm) + " E:" + String(pos_offset.emm) +
"]");
  Logger::logINFO("CURRENT POSITION: [X:"+String(new_x)+" Y:"+String(new_y)+"
Z:"+String(new_z)+" E:"+String(new_e)+"]");
}

void Interpolation::resetPosOffset(){
  pos_offset.xmm = 0.0;
  pos_offset.ymm = 0.0;
  pos_offset.zmm = 0.0;
  pos_offset.emm = 0.0;
}
```

```
Point Interpolation::getPosOffset() const {
  return pos_offset;
}


void Interpolation::setCurrentPos(float px, float py, float pz, float pe) {
  Point p;
  p.xmm = px;
  p.ymm = py;
  p.zmm = pz;
  p.emm = pe;
  setCurrentPos(p);
}

void Interpolation::setInterpolation(float px, float py, float pz, float pe, float v) {
  Point p;
  p.xmm = px;
  p.ymm = py;
  p.zmm = pz;
  p.emm = pe;
  setInterpolation(p, v);
}

void Interpolation::setInterpolation(float p1x, float p1y, float p1z, float p1e, float p2x, float p2y,
float p2z, float p2e, float v) {
  Point p1;
  Point p2;
  p1.xmm = p1x;
  p1.ymm = p1y;
  p1.zmm = p1z;
  p1.emm = p1e;
```

```
  p2.xmm = p2x;
  p2.ymm = p2y;
  p2.zmm = p2z;
  p2.emm = p2e;
  setInterpolation(p1, p2, v);
}

void Interpolation::setInterpolation(Point p1, float v) {
  Point p0;
  p0.xmm = xStartmm + xDelta;
  p0.ymm = yStartmm + yDelta;
  p0.zmm = zStartmm + zDelta;
  p0.emm = eStartmm + eDelta;
  setInterpolation(p0, p1, v);
}

void Interpolation::setInterpolation(Point p0, Point p1, float av) {
  v = av; //mm/s

  float a = (p1.xmm - p0.xmm);
  float b = (p1.ymm - p0.ymm);
  float c = (p1.zmm - p0.zmm);
  float e = abs(p1.emm - p0.emm);
  float dist = sqrt(a*a + b*b + c*c);

  if (dist < e) {
    dist = e;
  }

  if (v < 5) { //includes 0 = default value
    v = sqrt(dist) * 10; //set a good value for v
```

```
  }
  if (v < 5) {
    v = 5;
  }

  tmul = v / dist;

  xStartmm = p0.xmm;
  yStartmm = p0.ymm;
  zStartmm = p0.zmm;
  eStartmm = p0.emm;

  xDelta = (p1.xmm - p0.xmm);
  yDelta = (p1.ymm - p0.ymm);
  zDelta = (p1.zmm - p0.zmm);
  eDelta = (p1.emm - p0.emm);

  state = 0;

  startTime = micros();
}

void Interpolation::setCurrentPos(Point p) {
  xStartmm = p.xmm;
  yStartmm = p.ymm;
  zStartmm = p.zmm;
  eStartmm = p.emm;
  xDelta = 0;
  yDelta = 0;
  zDelta = 0;
  eDelta = 0;
```

```
}

void Interpolation::updateActualPosition() {
  if (state != 0) {
    return;
  }
  long microsek = micros();
  float t = (microsek - startTime) / 1000000.0;
  float progress;
  switch (SPEED_PROFILE){
    // FLAT SPEED CURVE
    case 0:
      progress = t * tmul;
      if (progress >= 1.0){
        progress = 1.0;
        state = 1;
      }
      break;
    // ARCTAN APPROX
    case 1:
      progress = atan((PI * t * tmul) - (PI * 0.5)) * 0.5 + 0.5;
      if (progress >= 1.0) {
        progress = 1.0;
        state = 1;
      }
      break;
    // COSIN APPROX
    case 2:
      progress = -cos(t * tmul * PI) * 0.5 + 0.5;
      if ((t * tmul) >= 1.0) {
        progress = 1.0;
```

```
      state = 1;
    }
    break;
}
pos_tracker[X_AXIS] = xStartmm + progress * xDelta;
pos_tracker[Y_AXIS] = yStartmm + progress * yDelta;
pos_tracker[Z_AXIS] = zStartmm + progress * zDelta;
pos_tracker[E_AXIS] = eStartmm + progress * eDelta;

if(isAllowedPosition(pos_tracker)){
  xPosmm = pos_tracker[X_AXIS];
  yPosmm = pos_tracker[Y_AXIS];
  zPosmm = pos_tracker[Z_AXIS];
  ePosmm = pos_tracker[E_AXIS];
} else {
  pos_tracker[X_AXIS] = xPosmm;
  pos_tracker[Y_AXIS] = yPosmm;
  pos_tracker[Z_AXIS] = zPosmm;
  pos_tracker[E_AXIS] = ePosmm;
  state = 1;
  progress = 1.0;
  xStartmm = xPosmm;
  yStartmm = yPosmm;
  zStartmm = zPosmm;
  eStartmm = ePosmm;
  xDelta = 0;
  yDelta = 0;
  zDelta = 0;
  eDelta = 0;
}
//FOR DECIPHERING SPEED CURVE
```

```
  //Serial.print("xPosmm:");
  //Serial.print(xPosmm);
  //Serial.print(" yPosmm:");
  //Serial.print(yPosmm);
  //Serial.print(" zPosmm:");
  //Serial.println(zPosmm);
}

bool Interpolation::isFinished() const {
  return state != 0;
}

float Interpolation::getXPosmm() const {
  return xPosmm;
}

float Interpolation::getYPosmm() const {
  return yPosmm;
}

float Interpolation::getZPosmm() const {
  return zPosmm;
}

float Interpolation::getEPosmm() const {
  return ePosmm;
}

Point Interpolation::getPosmm() const {
  Point p;
  p.xmm = xPosmm;
```

```
  p.ymm = yPosmm;
  p.zmm = zPosmm;
  p.emm = ePosmm;
  return p;
}

bool Interpolation::isAllowedPosition(float pos_tracker[4]) {
  float rrot_ee = hypot(pos_tracker[X_AXIS], pos_tracker[Y_AXIS]);
  float rrot = rrot_ee - END_EFFECTOR_OFFSET;
  float rrot_x = rrot * (pos_tracker[Y_AXIS] / rrot_ee);
  float rrot_y = rrot * (pos_tracker[X_AXIS] / rrot_ee);
  float squaredPositionModule = sq(rrot_x) + sq(rrot_y) + sq(pos_tracker[Z_AXIS]);


     bool retVal = (
        squaredPositionModule <= sq(R_MAX)
        && squaredPositionModule >= sq(R_MIN)
        && pos_tracker[Z_AXIS] >= Z_MIN
        && pos_tracker[Z_AXIS] <= Z_MAX
        && pos_tracker[E_AXIS] <= RAIL_LENGTH
     );
  if(!retVal) {
    Logger::logERROR("LIMIT REACHED: [X:" + String(pos_tracker[X_AXIS]) + " Y:" +
String(pos_tracker[Y_AXIS]) + " Z:" + String(pos_tracker[Z_AXIS]) + " E:" +
String(pos_tracker[E_AXIS]) + "]");
  }
  return retVal;
}
```

ДОДАТОК Б
ДЕМОНСТРАЦІЙНИЙ МАТЕРІАЛ

№	Формат	Позначення	Найменування	Кіл.	Прим.
			Документація текстова		
1	*А4*	*ГЮІК.416616.003ПЗ*	*Пояснювальна записка*	*73с*	
			Інші документи		
2	*А4*		*Код програми*	*8с*	
3	*А4*		*Демонстраційний матеріал*	*12с*	*у форматі *.ppt*

Змі	Арк	№ докум.	Підпис	Дата	ГЮІК. 416616.003ВД		
Розроб		Слупський Є.Ю.			*Розроблення модуля автоматичного керування промисловим маніпулятором*	Літера	Аркуш 1 / Аркушів
Перевірив		Новоселов С.П.					
Н. Контр.		Стародубцев М. Г.				ХНУРЕ кафедра КІТАР	
Затвер.		Невлюдов І.. Ш.					