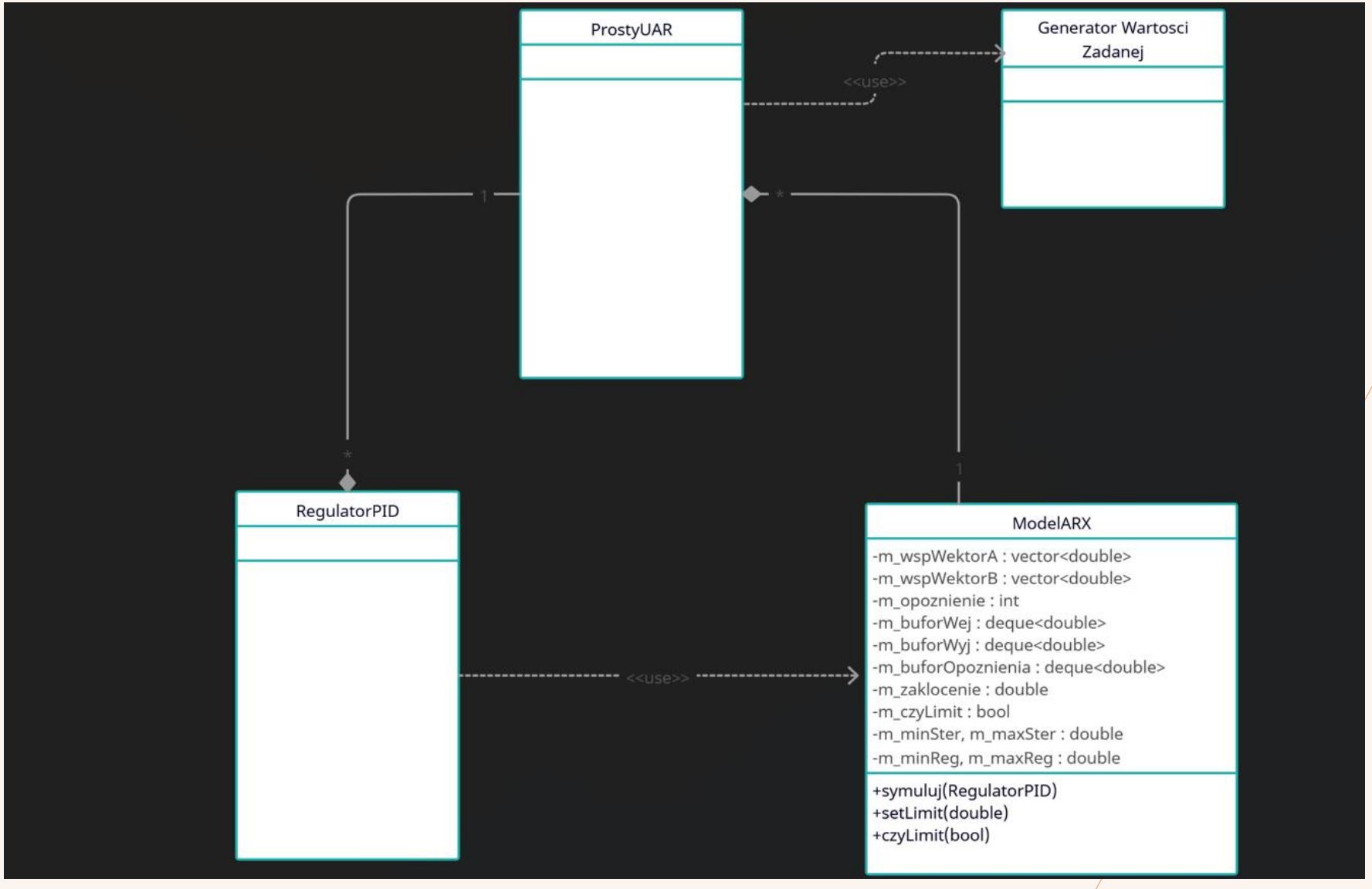
SYMULATOR UKŁADU AUTOMATYCZNEJ REGULACJI

Kacper Skowroński i Igor Juraszek

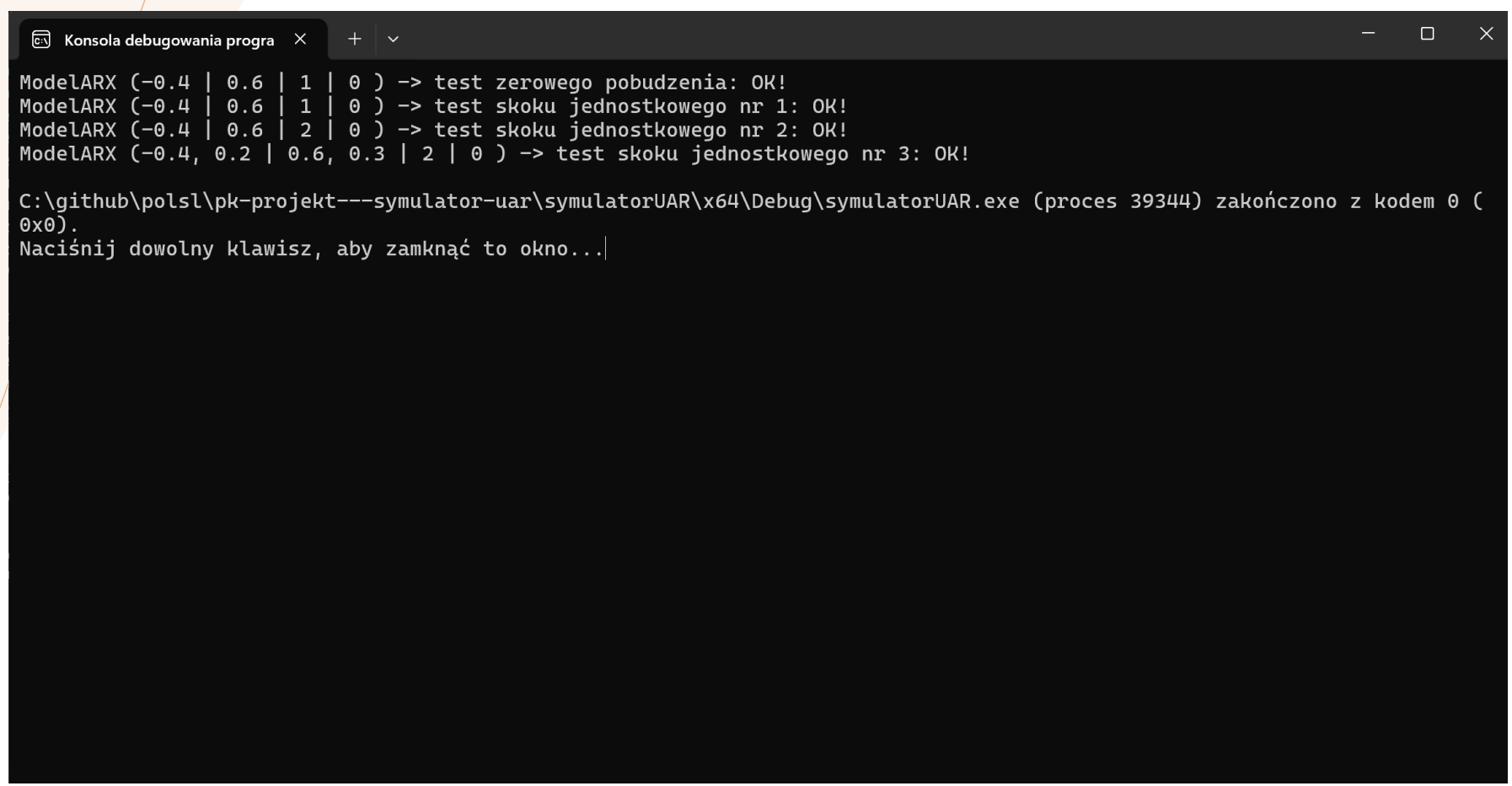
PODZIAŁ PRACY

Kacper Skowroński	Igor Juraszek
Tworzenie prezentacji Model ARX	GUI Regulator PID

WSTĘPNY SCHEMAT HIERARCHII KLAS



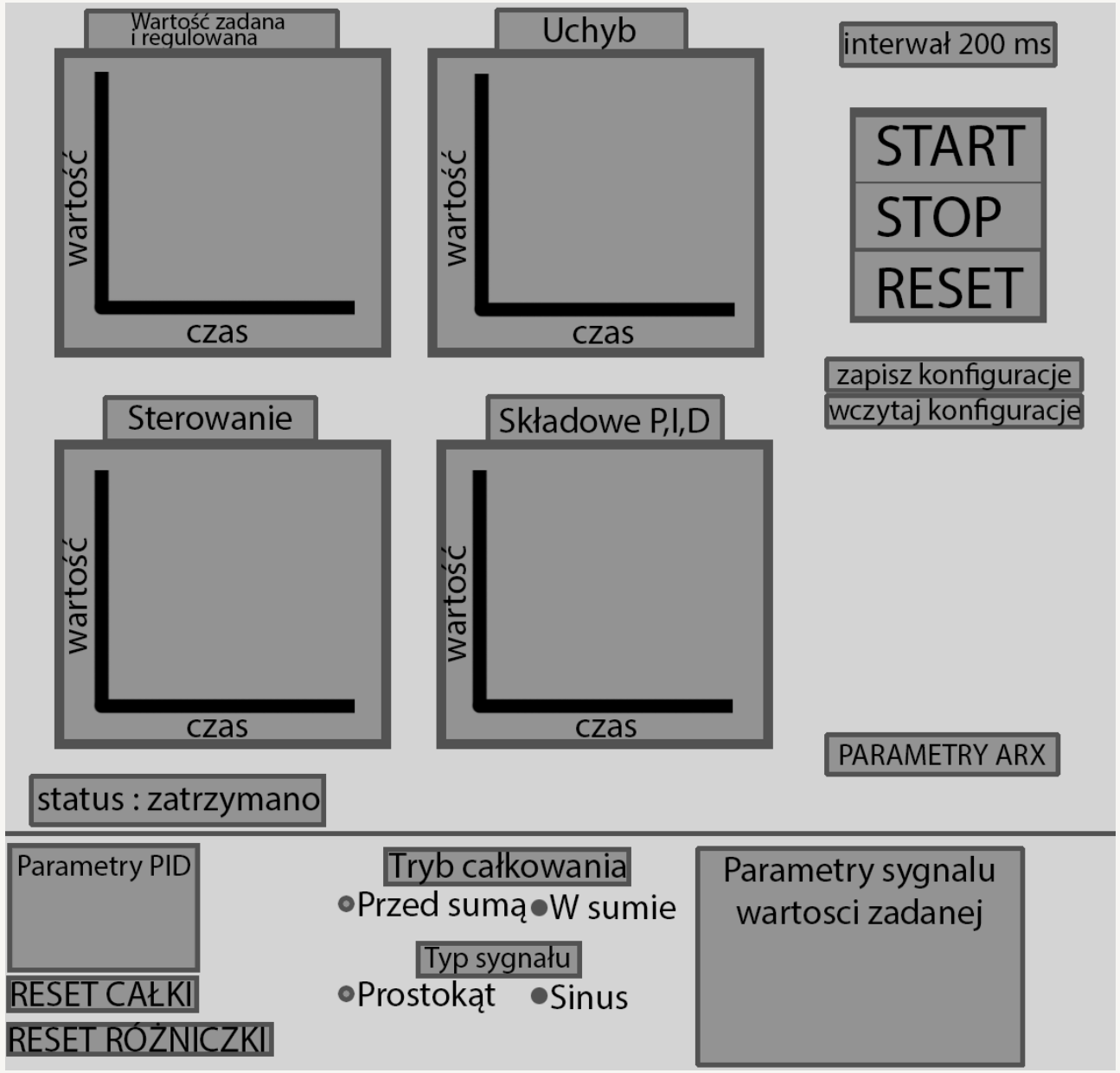
WYNIK TESTÓW MODELU ARX

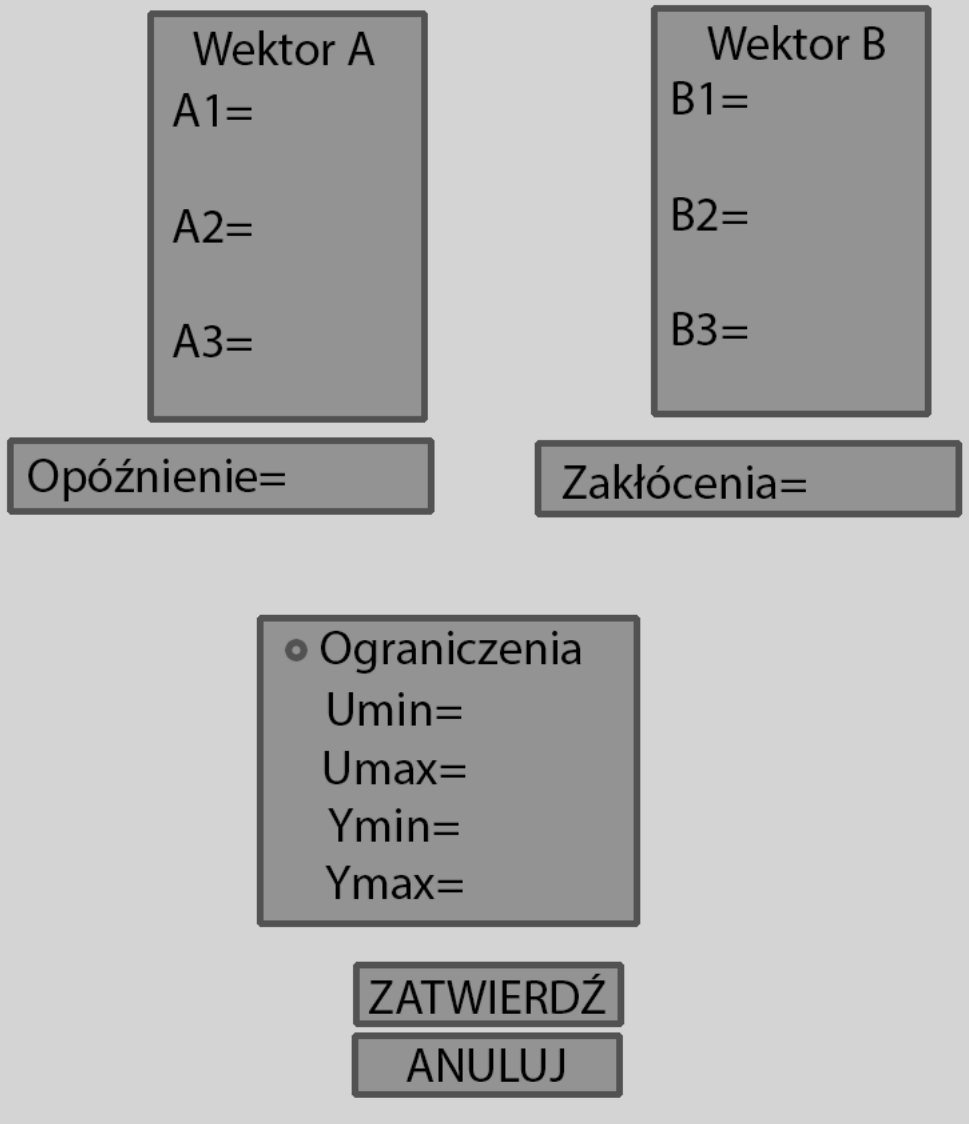


```cpp
class ModelARX
{
private:
    std::vector<double> m_wspWektorA;
    std::vector<double> m_wspWektorB;
    std::deque<double> m_kolejkaWej;
    std::deque<double> m_kolejkaWyj;
    std::deque<double> m_buforOpoznienia;
    double m_minSter, m_maxSter;
    double m_minReg, m_maxReg;
    int m_opoznienie;
    double m_zaklocenie;
    bool m_czyLimit;
public:
    ModelARX(const std::vector<double>& wektorA, const std::vector<double>& wektorB, int opoznienie, double zaklocenie) : m_wspWektorA(wektorA), m_wspWektorB(wektorB), m_opoznienie(opoznienie), m_zaklocenie(zaklocenie) {
        m_kolejkaWej = std::deque<double>(wektorB.size(), 0.0);
        m_kolejkaWyj = std::deque<double>(wektorA.size(), 0.0);
        m_buforOpoznienia = std::deque<double>(opoznienie, 0.0);
    }
    void setLimit(double minSter, double maxSter, double minReg, double maxReg);
    void czyLimit(bool czyLimit);
    bool getCzyLimit();
    double getMinSter();
    double getMaxSter();
    double getMinReg();
    double getMaxReg();
    double symuluj(double aktualnaWartSter);

};
```

```cpp
}
double ModelARX::symuluj(double aktualnaWartSter)
{
    if (!getCzyLimit()) {
        if (aktualnaWartSter > 10) aktualnaWartSter = 10;
        if (aktualnaWartSter < -10) aktualnaWartSter = -10;
    }
    else {
        if (aktualnaWartSter > getMaxSter()) aktualnaWartSter = getMaxSter();
        if (aktualnaWartSter < getMinSter()) aktualnaWartSter = getMinSter();
    }


    double aktBufor = 0.0;
    if (m_buforOpoznienia.size() >= m_opoznienie) {
        aktBufor = m_buforOpoznienia[m_opoznienie - 1];
    }

    m_buforOpoznienia.push_front(aktualnaWartSter);
    if (m_buforOpoznienia.size() > m_opoznienie) m_buforOpoznienia.pop_back();

    m_kolejkaWej.push_front(aktBufor);
    if (m_kolejkaWej.size() > m_wspWektorB.size()) m_kolejkaWej.pop_back();

    double sumaB = 0.0;
    for (int i = 0; i < m_wspWektorB.size(); i++) {
        if (i < m_kolejkaWej.size()) {
            sumaB += m_wspWektorB[i] * m_kolejkaWej[i];
        }
    }

    double sumaA = 0.0;
    for (int i = 0; i < m_wspWektorA.size(); i++) {
        if (i < m_kolejkaWyj.size()) {
            sumaA += m_wspWektorA[i] * m_kolejkaWyj[i];
        }
    }

    double wartReg = sumaB - sumaA;

    if (!getCzyLimit()) {
        if (wartReg > 10) wartReg = 10;
        if (wartReg < -10) wartReg = -10;
    }
    else {
        if (wartReg > getMaxReg()) wartReg = getMaxReg();
        if (wartReg < getMinReg()) wartReg = getMinReg();
    }

    m_kolejkaWyj.push_front(wartReg);
    if (m_kolejkaWyj.size() > m_wspWektorA.size()) m_kolejkaWyj.pop_back();

    return wartReg;
}
```

WSTĘPNY WYGLĄD GUI




Wektor A
A1=
A2=
A3=
Wektor B
B1=
B2=
B3=
Opóźnienie=
Zakłócenia=
Ograniczenia
Umin=
Umax=
Ymin=
Ymax=
ZATWIERDŹ
ANULUJ