Rok. ak. 2021/2022
grupa dziekańska: 6AiSR2
termin realizowania laboratorium: czwartek 8:15

SPRAWOZDANIE Z LABORATORIUM

OPROGRAMOWANIA SYSTEMÓW MIKROPROCESOROWYCH

Autorzy sprawozdania:

1. Elena Gricjuta 228391

2. Arkadiusz Jóźwiak 228401

imię nazwisko nr indeksu

Data wykonania sprawozdania: 30.04.2022

Wersja szablonu 1.2.2

Spis treści

1. Symulator dyskretnego układu regulacji

1.1. Specyfikacja projektu

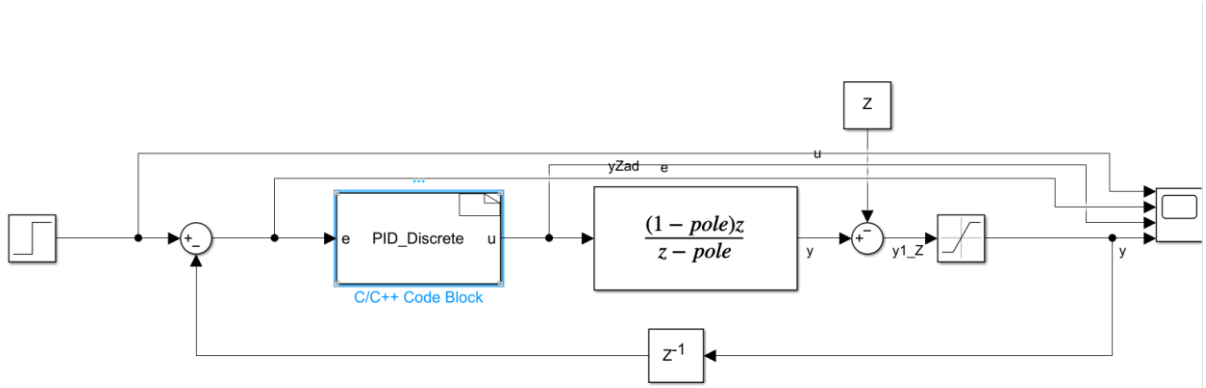
1.1.1 Założenia dotyczące modelu obiektu i regulatora.

Założyliśmy, że zrealizujemy układ regulacji z regulatorem PID. Głównym celem naszego układu jest regulacja poziomu wody w zbiorniku. W naszym założeniach zbiornika powinien posiadać dwa zawory. Przez ten pierwszy woda powinna się wlewać, a drugi otwiera się, gdy poziom wody jest wyższy niż wartość zadana.

1.1.2 Założenia związane z GUI

Na ekranie przede wszystkim powinien wyświetlać się zbiornik, w którym następuje regulacja poziomu wody. Wybór nastaw oraz wartości zadanej powinny być wybierane przez użytkownika. Także powinny być widoczne wykresy związane z regulacją, które są wykreślane w czasie rzeczywistym.

1.1.3 Schemat blokowy układu regulacji z symulowanym obiektem



Rysunek 1 - schemat blokowy

Dla:

%UAR

$Tp = 0.5;$
$yZad = 50;$
$HReal = 100;$

%PID

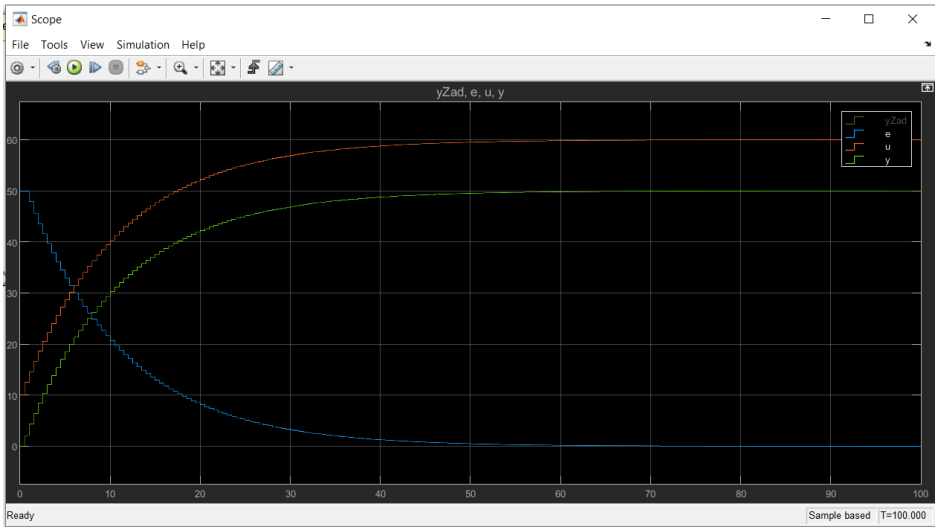
$kp = 0.1;$
$Ti = 1;$
$Td = 0.2;$
$Smin = 0;$
$Smax = HReal;$

%G

$pole = 0.1;$

%Z

$Z = HReal/10;$



Rysunek 2

Dla:

%UAR

$Tp = 0.5;$
$yZad = 100;$
$HReal = 100;$

%PID

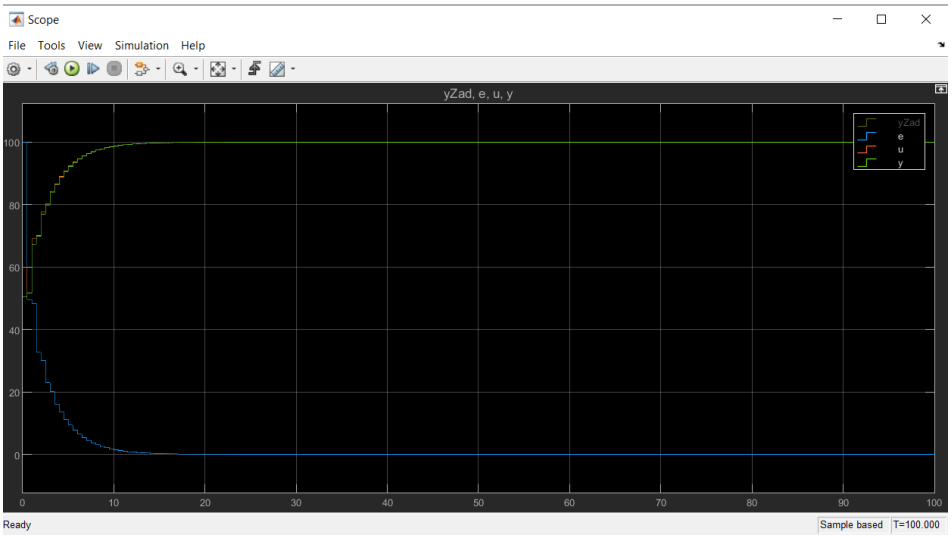
$kp = 0.2;$
$Ti = 1;$
$Td = 0.5;$
$Smin = 0;$
$Smax = HReal;$

%G

$pole = 0.1;$

%Z

$Z = 0;$



Rysunek 3

Dla:

%UAR

$Tp = 0.5;$
$yZad = 25;$
$HReal = 100;$

%PID

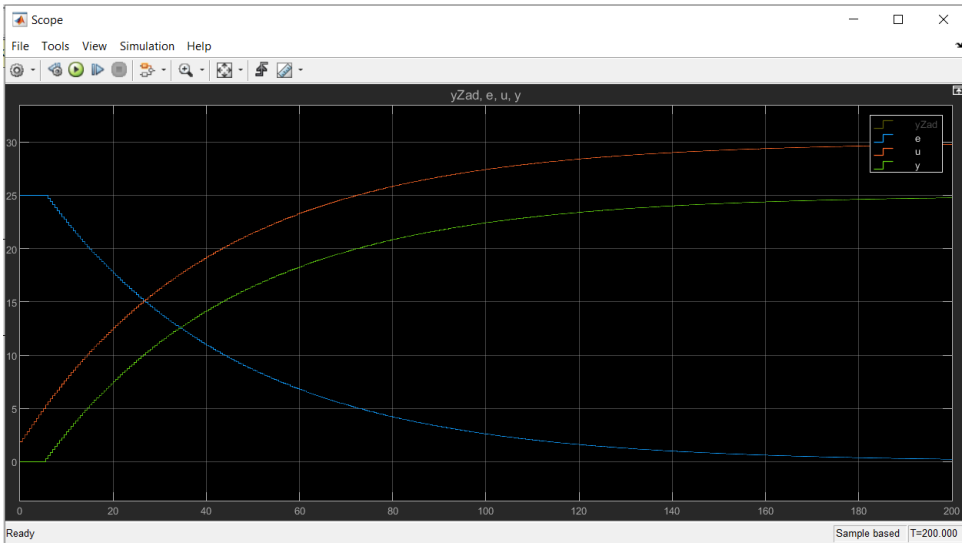
$kp = 0.05;$
$Ti = 2;$
$Td = 0.1;$
$Smin = 0;$
$Smax = HReal;$

%G

$pole = 0.1;$

%Z

$Z = Hreal/20;$



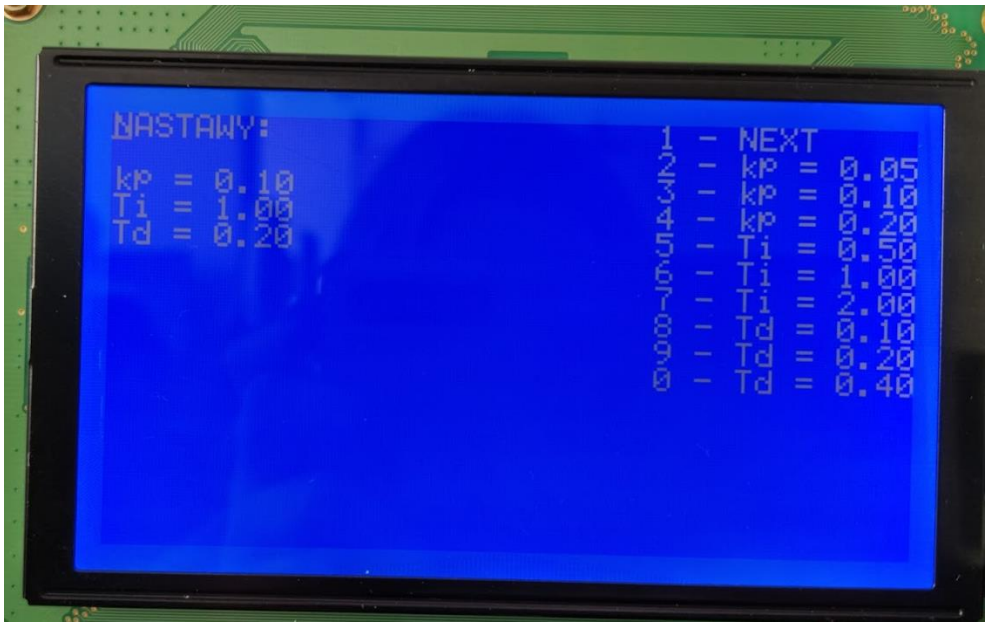
Rysunek 4

1.2. Oprogramowanie symulatora na platformę TMSMULTILAB

1.2.1 Charakterystyka oprogramowania

Od razu po uruchomieniu programu powinien być widoczny wybór nastaw oraz wartości zadanej dla danego obiektu. Na kolejnym ekranie powinien pojawić się zbiornik oraz realizacja regulatora PID dla wybranych nastaw oraz zadanej wartości. Ostatni ekran podzielony jest na cztery części. Na trzech wykreślają się wykresy, a czwarta duplikuje poprzedni ekran. Przełączanie się pomiędzy ekranami możliwe jest za pomocą przycisku „1” *(następny ekran)* oraz za pomocą przycisku „2” *(poprzedni ekran)*

Pierwszy ekran – wybór nastaw:

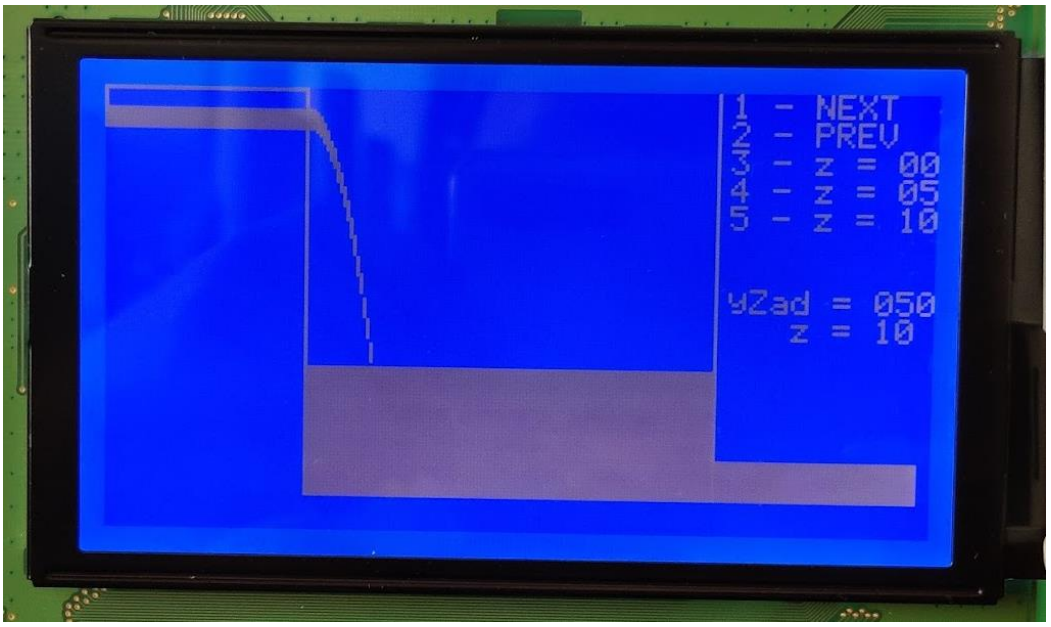


Rysunek 5 - pierwszy ekran

Drugi ekran – wybór wartości zadanej:

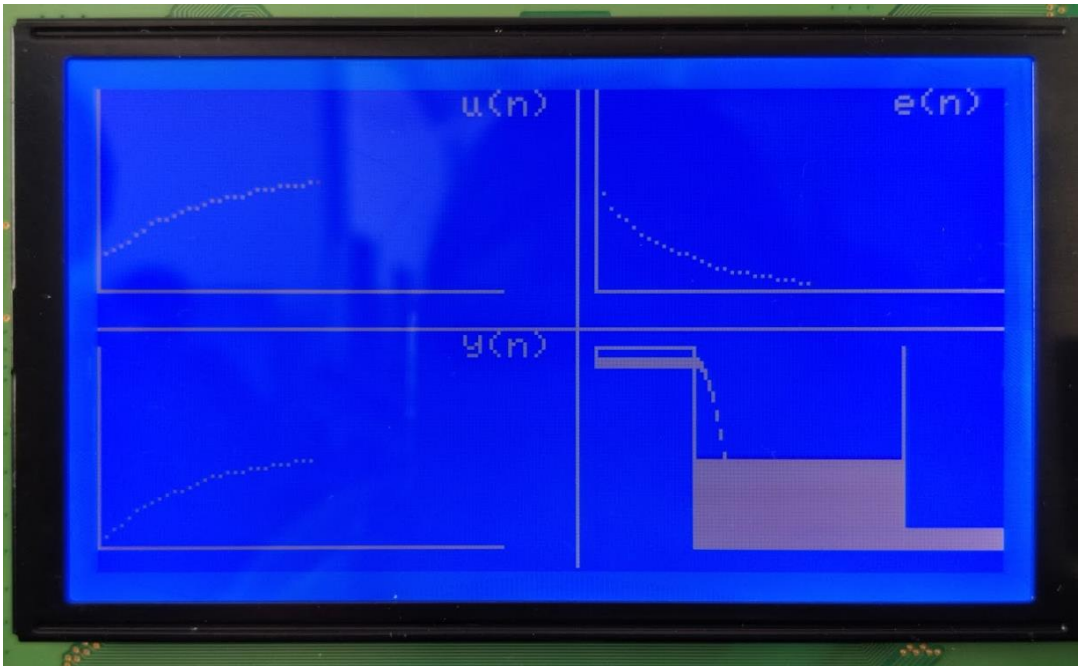
Rysunek 6 - drugi ekran

Trzeci ekran – wyświetlanie zbiornika:



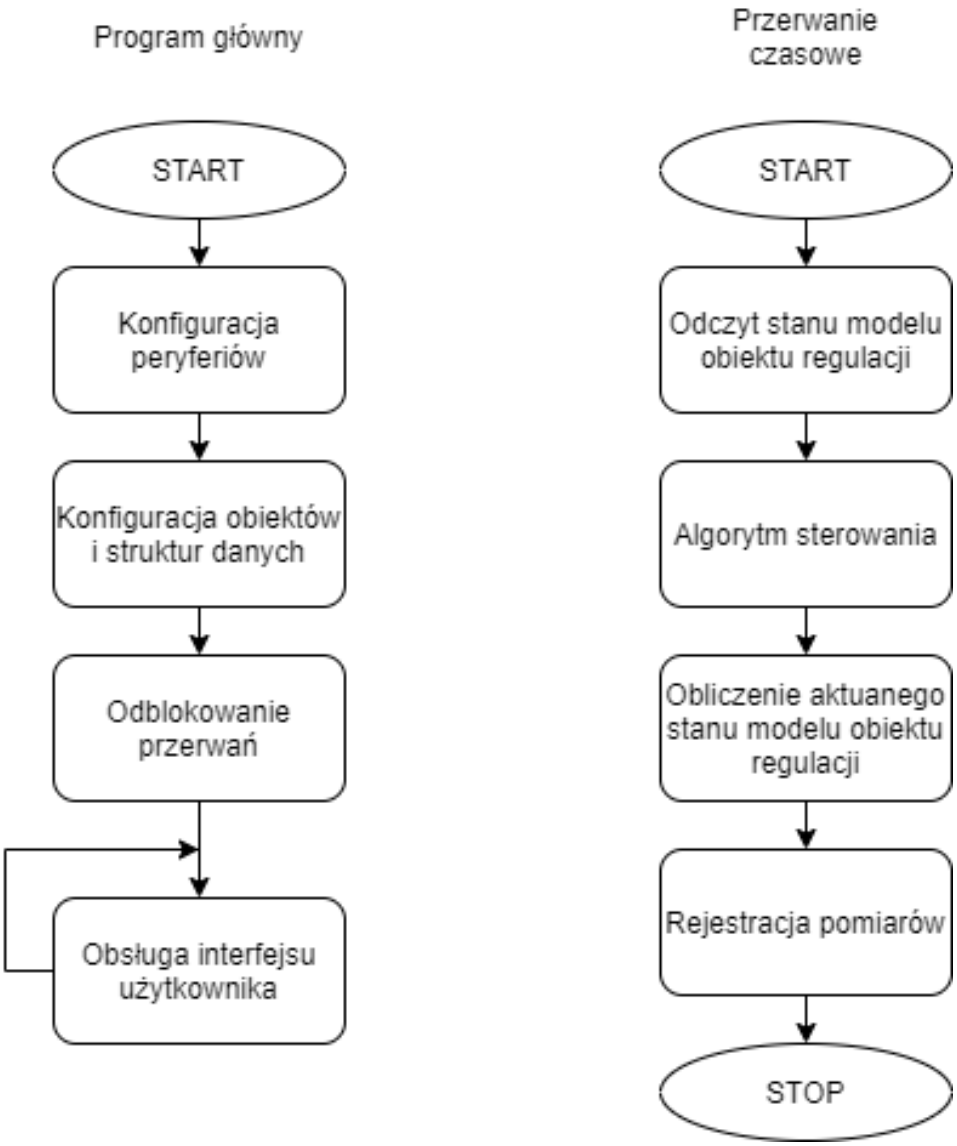
Rysunek 7 - trzeci ekran

Czwarty ekran:



Rysunek 8 - czwarty ekran

1.2.2 Schemat blokowe głównych algorytmów



Rysunek 9

1.2.3 Kod źródłowy

```
////////////////////////////////////////////////
#define BUFFERSYNC
#define WIN_PLOT
#define NazwaPlikuDanych "Data/TMSLABoutputData.cs"
#define CSV_NAGLOWEK "Wsp. x,Wsp. y1,Wsp. y2\n"
#define CSV_DANE "%i,%i,%i\n", Tim, y[0], y[1]
////////////////////////////////////////////////////

#include "main.h"
#include "vector.h"
#include "UAR.h"
#include "PID.h"
#include "InercModel.h"
#include <limits>
#include "stdio.h"

#ifdef TMSLAB_WIN
#include "stdio.h"
#endif

unsigned long *ekran;
#ifdef TMSLAB_C2000
unsigned char *textEkran;
#endif

#ifdef TMSLAB_WIN
unsigned short int *textEkran;
extern int (*PartialRefresh)();
char credits[43] = "-              DEMO DISC              -";
long Timer2IsrPeriod = 1;
#ifdef WIN_PLOT
FILE *outputCSV = 0;
#endif
#endif

int Tim = 0;
int TimPrev = 0;
unsigned int preScale = 0;
volatile char EnableRefresh = 0;

R_P_LCD_TMSLAB LCD;
R_P_KEYBOARD_TMSLAB KEYBOARD;
R_P_LEDBAR_TMSLAB LEDBAR;

// Tablice danych/obiektów graficznych

#define MaxObj 200
#ifdef CPP_EXAMPLE
square objects[MaxObj];
#else
int dx[MaxObj];
int dy[MaxObj];
int s[MaxObj];
int x[MaxObj];
int y[MaxObj];
#endif

// Screen

const int X = 255;
const int Y = 127;
const int a[] = {X / 2 - 20, X - 20};
const int H[] = {Y / 2 - 10, Y - 10};
const int x0[] = {0, X / 2 + 5};
const int y0[] = {0, Y / 2 + 5};
int screenNumber = 1;

// Tp

const unsigned int Tp_ms = 500;
const float Tp = (float)Tp_ms / 1000;

int main()

{
   SetUpPeripherials();
#ifdef TMSLAB_C2000
   LCD.LCD_Init(ekran, textEkran);
#endif

#ifdef TMSLAB_WIN
   LCD.LCD_Init(&ekran, &textEkran);
#ifdef WIN_PLOT
   outputCSV = fopen(NazwaPlikuDanych, „w+”);
   fprintf(outputCSV, CSV_NAGLOWEK);
#endif
#endif

   KEYBOARD.InitKB(100);

   LEDBAR.InitLedBar();

   InitData();

   EnableInterrupts();

   // UAR

   const float H_MAX = 100;
   float yZad = H_MAX / 2;

   // PID
   float kp = 0.1;
   float Ti = 1;
   float Td = 0.2;
   const float Smin = 0;
   const float Smax = H_MAX;
   PID pid = PID(kp, Ti, Td, Tp, Smin, Smax);
   UAR *C = &pid;

   // G
   float p = 0.1;
   InercModel im = InercModel(p);
```

```
   UAR *G = &im;

   // Signals

   float e = 0;
   float u = 0;
   float yWy = 0;
   float yWyPrev = 0;
   float z = 0;

   // v
   Vector v_u(1);
   Vector v_e(1);
   Vector v_yWy(1);

   // t

    const float t_yZad[] = {0, H_MAX * 0.25,
H_MAX * 0.5, H_MAX * 0.75, H_MAX * 1};
   const float t_kp[] = {0.5 * kp, kp, 2 * kp};
   const float t_Ti[] = {0.5 * Ti, Ti, 2 * Ti};
   const float t_Td[] = {0.5 * Td, Td, 2 * Td};
   const float t_z[] = {0, H_MAX / 20, H_MAX / 10};

   // ch

   unsigned char ch_next[] = "NEXT";
   unsigned char ch_prev[] = "PREV";
   unsigned char ch_u[] = "u(n)";
   unsigned char ch_e[] = "e(n)";
   unsigned char ch_y[] = "y(n)";
   unsigned char ch_nastawy[] = "NASTAWY:";
   unsigned char ch_kp[] = "kp =     ";
   unsigned char ch_Ti[] = "Ti =     ";
   unsigned char ch_Td[] = "Td =     ";
   unsigned char ch_wZad[] = "WARTOSC
ZADANA";
   unsigned char ch_yZad[] = "yZad =    ";
   unsigned char ch_z[] = "z =   ";
   unsigned char ch_temp[] = "   ";
   float temp;

   while (1)
   {
      // 1. CLEAR AND SYNCHRONIZE

      EnableRefresh = 1;
      LCD.Synchronize();
      EnableRefresh = 0;
      ClearScreen();
      ClearText();

      // 2. GET AND PROCESS AN INPUT; CLEAR
DATA IF SCREEN 3

      unsigned char Key = KEYBOARD.GetKey();

#ifdef TMSLAB_C2000
      switch (Key)
      {
         case 16:
            Key = 1;
            break;
         case 15:
            Key = 2;
            break;
         case 14:
            Key = 3;
            break;
         case 8:
            Key = 4;
            break;
         case 7:
            Key = 5;
            break;
         case 6:
            Key = 6;
            break;
         case 12:
            Key = 7;
            break;
         case 11:
            Key = 8;
            break;
         case 10:
            Key = 9;
            break;
         case 3:
            Key = 10;
            break;
      }
#endif

      if (ButtonClicked(Key))
      {
         if (Key == 1)
         {
            if (screenNumber == 4)
            {
               screenNumber = 0;
            }
            screenNumber++;
            if (screenNumber == 3)
            {
               yWy = 0;
               yWyPrev = 0;
               PID pid = PID(kp, Ti, Td, Tp, Smin,
Smax);
               C = &pid;
               InercModel im = InercModel(p);
               G = &im;
               v_u.clear();
               v_u.push_back(0);
               v_e.clear();
               v_e.push_back(0);
               v_yWy.clear();
               v_yWy.push_back(0);
            }
         }
      }
```

```
// 3. CHANGE SCREEN
switch (screenNumber)
{
// SCREEN 1 - NASTAWY
case 1:
   switch (Key)
   {
   case 2:
      kp = t_kp[0];
      break;
   case 3:
      kp = t_kp[1];
      break;
   case 4:
      kp = t_kp[2];
      break;
   case 5:
      Ti = t_Ti[0];
      break;
   case 6:
      Ti = t_Ti[1];
      break;
   case 7:
      Ti = t_Ti[2];
      break;
   case 8:
      Td = t_Td[0];
      break;
   case 9:
      Td = t_Td[1];
      break;
   case 10:
      Td = t_Td[2];
      break;
   }

   PrintText(textEkran, ch_nastawy, 8, 0, 0);
   for (int i = 1; i <= 10; i++)
   {
      if (i == 10)
      {
         ch_temp[0] = '0' + 0;
      }
      else
      {
         ch_temp[0] = '0' + i;
      }
      ch_temp[2] = '-';
      temp = i - 1;
      PrintText(textEkran,  ch_temp,  3,  27,
(int)temp);
   }
   PrintText(textEkran, ch_next, 4, 31, 0);

   for (int i = 0; i < 3; i++)
   {
      temp = t_kp[i];
      ch_kp[5] = '0' + temp;
      ch_kp[6] = '.';
      temp = (temp - (int)temp) * 10;
      ch_kp[7] = '0' + temp;
      temp = (temp - (int)temp) * 10;
      ch_kp[8] = '0' + temp;
      temp = i + 1;
      PrintText(textEkran, ch_kp, 9, 31, (int)temp);
   }
   for (int i = 0; i < 3; i++)
   {
      temp = t_Ti[i];
      ch_Ti[5] = '0' + temp;
      ch_Ti[6] = '.';
      temp = (temp - (int)temp) * 10;
      ch_Ti[7] = '0' + temp;
      temp = (temp - (int)temp) * 10;
      ch_Ti[8] = '0' + temp;
      temp = i + 4;
      PrintText(textEkran, ch_Ti, 9, 31, (int)temp);
   }
   for (int i = 0; i < 3; i++)
   {
      temp = t_Td[i];
      ch_Td[5] = '0' + temp;
      ch_Td[6] = '.';
      temp = (temp - (int)temp) * 10;
      ch_Td[7] = '0' + temp;
      temp = (temp - (int)temp) * 10;
      ch_Td[8] = '0' + temp;
      temp = i + 7;
      PrintText(textEkran, ch_Td, 9, 31, (int)temp);
   }

   temp = kp;
   ch_kp[5] = '0' + temp;
   ch_kp[6] = '.';
   temp = (temp - (int)temp) * 10;
   ch_kp[7] = '0' + temp;
   temp = (temp - (int)temp) * 10;
   ch_kp[8] = '0' + temp;
   PrintText(textEkran, ch_kp, 9, 0, 2);

   temp = Ti;
   ch_Ti[5] = '0' + temp;
   ch_Ti[6] = '.';
   temp = (temp - (int)temp) * 10;
   ch_Ti[7] = '0' + temp;
   temp = (temp - (int)temp) * 10;
   ch_Ti[8] = '0' + temp;
   PrintText(textEkran, ch_Ti, 9, 0, 3);

   temp = Td;
   ch_Td[5] = '0' + temp;
   ch_Td[6] = '.';
   temp = (temp - (int)temp) * 10;
   ch_Td[7] = '0' + temp;
   temp = (temp - (int)temp) * 10;
   ch_Td[8] = '0' + temp;
```

```
        PrintText(textEkran, ch_Td, 9, 0, 4);

        break;


    // SCREEN 2 - YZAD
    case 2:
        switch (Key)
        {
        case 2:
            yZad = t_yZad[0];
            break;
        case 3:
            yZad = t_yZad[1];
            break;
        case 4:
            yZad = t_yZad[2];
            break;
        case 5:
            yZad = t_yZad[3];
            break;
        case 6:
            yZad = t_yZad[4];
            break;
        }

        PrintText(textEkran, ch_wZad, 14, 0, 0);
        for (int i = 1; i <= 6; i++)
        {
            ch_temp[0] = '0' + i;
            ch_temp[2] = '-';
            temp = i - 1;
            PrintText(textEkran, ch_temp, 3, 26,
(int)temp);
        }
        PrintText(textEkran, ch_next, 4, 30, 0);
        for (int i = 0; i < 5; i++)
        {
            temp = t_yZad[i];
            ch_yZad[7] = '0' + (int)temp / 100;
            temp = temp - ((int)temp / 100) * 100;
            ch_yZad[8] = '0' + (int)temp / 10;
            temp = temp - ((int)temp / 10) * 10;
            ch_yZad[9] = '0' + (int)temp;
            temp = i + 1;
            PrintText(textEkran, ch_yZad, 10, 30,
(int)temp);
        }

        temp = yZad;
        ch_yZad[7] = '0' + (int)temp / 100;
        temp = temp - ((int)temp / 100) * 100;
        ch_yZad[8] = '0' + (int)temp / 10;
        temp = temp - ((int)temp / 10) * 10;
        ch_yZad[9] = '0' + (int)temp;

        PrintText(textEkran, ch_yZad, 10, 0, 2);

        break;


    // SCREEN 3 - WATERTANK + Z
    case 3:
        FunctionRealTime(a[1], H[1], x0[0], y0[0],
v_u, H_MAX, false);
        FunctionRealTime(a[1], H[1], x0[0], y0[0], v_e,
H_MAX, false);
        FunctionRealTime(a[1], H[1], x0[0], y0[0],
v_yWy, H_MAX, false);
        WaterTank_Pipes(a[1], H[1], x0[0], y0[0],
yWy, yWyPrev, H_MAX, z);

        switch (Key)
        {
        case 2:
            break;
        case 3:
            z = 0;
            break;
        case 4:
            z = t_z[1];
            break;
        case 5:
            z = t_z[2];
            break;
        }

        for (int i = 1; i <= 5; i++)
        {
            ch_temp[0] = '0' + i;
            ch_temp[2] = '-';
            temp = i - 1;
            PrintText(textEkran, ch_temp, 3, 30,
(int)temp);
        }
        PrintText(textEkran, ch_next, 4, 34, 0);
        PrintText(textEkran, ch_prev, 4, 34, 1);

        for (int i = 0; i < 3; i++)
        {
            temp = t_z[i];
            ch_z[4] = '0' + (int)temp / 10;
            temp = temp - ((int)temp / 10) * 10;
            ch_z[5] = '0' + (int)temp;
            temp = i + 2;
            PrintText(textEkran, ch_z, 6, 34, (int)temp);
        }
        temp = z;
        ch_z[4] = '0' + (int)temp / 10;
        temp = temp - ((int)temp / 10) * 10;
        ch_z[5] = '0' + (int)temp;

        PrintText(textEkran, ch_yZad, 10, 30, 7);
        PrintText(textEkran, ch_z, 6, 33, 8);
```

```
            break;




        // SCREEN 4 - (WATERTANK + FUNCTIONS)
+ Z
        case 4:
            PrintText(textEkran, ch_u, 4, 16, 0);
            PrintText(textEkran, ch_e, 4, 35, 0);
            PrintText(textEkran, ch_y, 4, 16, 8);

            FunctionRealTime(a[0], H[0], x0[0], y0[0],
v_u, H_MAX, true);
            FunctionRealTime(a[0], H[0], x0[1], y0[0], v_e,
H_MAX, true);
            FunctionRealTime(a[0], H[0], x0[0], y0[1],
v_yWy, H_MAX, true);
            WaterTank_Pipes(a[0], H[0], x0[1], y0[1],
yWy, yWyPrev, H_MAX, z);
            Cross();
            switch (Key)
            {
            case 2:
                screenNumber--;
                break;
            case 3:
                z = 0;
                break;
            case 4:
                z = t_z[1];
                break;
            case 5:
                z = t_z[2];
                break;
            }
            break;
        }

        // 4. ON INTERRUPT (T = TP) -> CALCULATE
        if (Tim != TimPrev && LastClickedKey(Key) !=
0)
        {
            e = yZad - yWy;
            C->setInput(e);
            C->Calculate();

            u = C->getOutput();
            G->setInput(u);
            G->Calculate();

            yWyPrev = yWy;

            yWy = G->getOutput() - z;
            if (yWy < 0)
            {
                yWy = 0;
            }

            v_u.push_back(u);
            v_e.push_back(e);
            v_yWy.push_back(yWy);

            TimPrev = Tim;
        }

#ifdef TMSLAB_WIN
        if (PartialRefresh())
            return 0;
#ifdef WIN_PLOT
        // Zapis danych do pliku
        fprintf(outputCSV, CSV_DANE);
        // printf("time %i \n", Tim);
        fflush(outputCSV);
        fflush(stdout);
#endif
#endif
    }
}

void ClearText()
{
    for (int y = 0; y < 15; y++)
    {
        for (int x = 0; x < 40; x++)
        {
            PrintText(textEkran, " ", 1, x, y);
        }
    }
}

bool ButtonClicked(int Key)
{
    static bool block = false;
    if (Key != 0 && block == false)
    {
        block = true;
        return true;
    }
    if (Key == 0)
    {
        block = false;
    }
    return false;
}

int LastClickedKey(int Key)
{
    static unsigned char KeyCurrent = 0;
    if (Key != 0)
    {
        KeyCurrent = Key;
    }
    return KeyCurrent;
}

void FunctionRealTime(int a, int H, int x0, int y0,
```

```
Vector &v, float H_MAX, bool visualize)
{

   if (visualize)
   {
      DrawAxes(a, H, x0, y0);
   }
   int n = 40;
   int dx = a / n;
   int x = x0;
   int y = 0;
   for (int i = 0; i < v.size(); i++)
   {
      y = v[i];
      y = y * H / H_MAX;
      y = H + y0 - y;
      if (visualize)
      {
         SetPixel(ekran, x, y);
      }
      if (x <= a + x0)
      {
         x = x + dx;
      }
      else
      {
         v.erase(v.begin());
      }
   }
}

void DrawAxes(int a, int H, int x0, int y0)
{
   for (int y = y0; y <= y0 + H; y++)
   {
      for (int x = x0; x <= x0 + a; x++)
      {
         if (y == y0 + H || x == x0)
         {
            SetPixel(ekran, x, y);
         }
      }
   }
}

void Cross()
{
   for (int y = 0; y < Y; y++)
   {
      for (int x = 0; x < X; x++)
      {
         if (y == Y / 2 || x == X / 2)
         {
            SetPixel(ekran, x, y);
         }
      }
   }
}

void WaterTank_Pipes(int a, int H, int x0, int y0, float
yWy, float yWyPrev, float H_MAX, float z)
{
   int a_P = a / 4;
   int a_WT = a - 2 * a_P;

   int H_WT = H;
   int H_P = H / 10;
   int H_WA = H - H_P;

   int x0_WT = x0 + a_P;
   int y0_WT = y0;
   int x0_P1 = x0;
   int y0_P1 = y0;
   int x0_P2 = x0 + a_P + a_WT;
   int y0_P2 = y0 + H - H_P;
   int x0_WA = x0 + a_P;
   int y0_WA = y0 + H_P;

   float h_WT = yWy * H / H_MAX;
   int h_P1 = 0;
   float h_P2 = z * H / H_MAX;

   float dh_WT = (float)(yWy - yWyPrev) * H /
H_MAX;

   WaterTank(a_WT, H_WT, x0_WT, y0_WT, h_WT);
   h_P1 = Pipe_In(a_P, H_P, x0_P1, y0_P1, dh_WT,
H_WT);
   WaterAnimation(a_WT, H_WA, x0_WA, y0_WA,
dh_WT, H_WT, h_P1);
   Pipe_Out(a_P, H_P, x0_P2, y0_P2, h_P2, h_WT,
H_MAX);
}

void WaterAnimation(int a, int H, int x0, int y0, float
dh_WT, int H_WT, int h_P1)
{
   const float vMax = (float)H / (a * a);
   float v;
   if (dh_WT > 0 && dh_WT <= H_WT / 100)
   {
      v = 50 * vMax;
   }
   if (dh_WT > H_WT / 100 && dh_WT <= H_WT /
90)
   {
      v = 45 * vMax;
   }
   if (dh_WT > H_WT / 90 && dh_WT <= H_WT /
80)
   {
      v = 40 * vMax;
   }
   if (dh_WT > H_WT / 80 && dh_WT <= H_WT /
70)
   {
      v = 35 * vMax;
   }
   if (dh_WT > H_WT / 70 && dh_WT <= H_WT /
60)
```

```
    {
        v = 30 * vMax;
    }
    if (dh_WT > H_WT / 60 && dh_WT <= H_WT /
50)
    {
        v = 25 * vMax;
    }
    if (dh_WT > H_WT / 50 && dh_WT <= H_WT /
40)
    {
        v = 20 * vMax;
    }
    if (dh_WT > H_WT / 40 && dh_WT <= H_WT /
30)
    {
        v = 15 * vMax;
    }
    if (dh_WT > H_WT / 30 && dh_WT <= H_WT /
20)
    {
        v = 10 * vMax;
    }
    if (dh_WT > H_WT / 20 && dh_WT <= H_WT /
10)
    {
        v = 5 * vMax;
    }
    if (dh_WT > H_WT / 10 && dh_WT <= H_WT)
    {
        v = 1 * vMax;
    }

    int y;
    if (dh_WT > 0)
    {
        for (int x = x0; x <= x0 + a; x++)
        {
            for (int i = 0; i <= h_P1; i++)
            {
                y = y0 + v * (x - x0) * (x - x0) - i;
                if (y < y0 + H && x < x0 + a)
                    SetPixel(ekran, x, y);
            }
        }
    }
}

void WaterTank(int a, int H, int x0, int y0, float h)
{
    for (int y = y0; y <= y0 + H; y++)
    {
        if (y == y0 + H)
        {
            for (int x = x0; x <= x0 + a; x++)
            {
                SetPixel(ekran, x, y);
            }
        }
        for (int x = x0; x <= x0 + a; x++)
        {
            if ((x == x0) || (x == x0 + a))
                SetPixel(ekran, x, y);
        }
    }

    for (int y = y0; y <= y0 + H; y++)
    {
        if (y <= y0 + H && y >= y0 + H - h)
        {
            for (int x = x0; x <= x0 + a; x++)
            {
                SetPixel(ekran, x, y);
            }
        }
    }
}

int Pipe_In(int a, int H, int x0, int y0, float dh_WT, int
H_WT)
{
    for (int y = y0; y <= y0 + H; y++)
    {
        if (y == y0 + H || y == y0)
        {
            for (int x = x0; x <= x0 + a; x++)
            {
                SetPixel(ekran, x, y);
            }
        }
        for (int x = x0; x <= x0 + a; x++)
        {
            if ((x == x0) || (x == x0 + a))
                SetPixel(ekran, x, y);
        }
    }

    float h;
    if (dh_WT > 0 && dh_WT <= H_WT / 100)
        h = 0.1 * H;
    if (dh_WT > H_WT / 100 && dh_WT <= H_WT /
90)
        h = 0.1 * H;
    if (dh_WT > H_WT / 90 && dh_WT <= H_WT /
80)
        h = 0.2 * H;
    if (dh_WT > H_WT / 80 && dh_WT <= H_WT /
70)
        h = 0.3 * H;
    if (dh_WT > H_WT / 70 && dh_WT <= H_WT /
60)
        h = 0.4 * H;
    if (dh_WT > H_WT / 60 && dh_WT <= H_WT /
50)
        h = 0.5 * H;
    if (dh_WT > H_WT / 50 && dh_WT <= H_WT /
40)
        h = 0.6 * H;
    if (dh_WT > H_WT / 40 && dh_WT <= H_WT /
30)
```

```
      h = 0.7 * H;
   if (dh_WT > H_WT / 30 && dh_WT <= H_WT /
20)
      h = 0.8 * H;
   if (dh_WT > H_WT / 20 && dh_WT <= H_WT /
10)
      h = 0.9 * H;
   if (dh_WT > H_WT / 10 && dh_WT <= H_WT)
      h = H;

   for (int y = y0; y <= y0 + H; y++)
   {
      if (y <= y0 + H && y >= y0 + H - h)
      {
         for (int x = x0; x <= x0 + a; x++)
         {
            SetPixel(ekran, x, y);
         }
      }
   }

   return h;
}

void Pipe_Out(int a, int H, int x0, int y0, float h, float
h_WT, float H_MAX)
{
   for (int y = y0; y <= y0 + H; y++)
   {
      if (y == y0 + H || y == y0)
      {
         for (int x = x0; x <= x0 + a; x++)
         {
            SetPixel(ekran, x, y);
         }
      }
      for (int x = x0; x <= x0 + a; x++)
      {
         if ((x == x0) || (x == x0 + a))
            SetPixel(ekran, x, y);
      }
   }
   float hSaturated;
   if (h_WT < h && h != 0)
   {
      hSaturated = h_WT;
   }
   else if (h != 0)
   {
      hSaturated = h;
   }

   for (int y = y0; y <= y0 + H; y++)
   {
      if (y <= y0 + H && y >= y0 + H - hSaturated)
      {
         for (int x = x0; x <= x0 + a; x++)
         {
            SetPixel(ekran, x, y);
         }
      }
   }
}

#ifdef TMSLAB_C2000

interrupt void Timer2Isr()
{

#ifdef BUFFERSYNC
   if (EnableRefresh)
      LCD.PartialRefresh();
#else
   LCD.PartialRefresh();
#endif

   KEYBOARD.PartialRefresh();

   if (++preScale == Tp_ms * 100)
   {
      preScale = 0;
      Tim++;
   }
}

unsigned long ADRFTECHED = 0;
interrupt void NoIsr()
{
   ADRFTECHED                                    =
PieCtrlRegs.PIECTRL.bit.PIEVECT;
   asm(" ESTOP0");
}

void EnableInterrupts()
{

   EALLOW;
   // Ustawienie wektorow przerwan
   unsigned long VECTBEG = (unsigned
long)&PieVectTable;
   unsigned long VECTLAST = (unsigned
long)&PieVectTable + sizeof(PieVectTable);
   while (VECTBEG >= VECTLAST)
      *(unsigned long *)VECTBEG++ = (unsigned
long)NoIsr;
   PieVectTable.TIMER2_INT = Timer2Isr;

   CpuTimer2Regs.TCR.bit.TIE = 1;
   CpuTimer2Regs.TCR.bit.TRB = 1;

   IER = IER_MASK; // Odblokuj przerwania
   asm(" push ier");
   asm(" pop dbgier");

   PieCtrlRegs.PIECTRL.bit.ENPIE = 1;
   PieCtrlRegs.PIEACK.all = 0xffff;
   EDIS;
   EINT;
}
```

```
void SetUpPeripherials()
{
   SetupCoreSystem();
   ekran = (unsigned long *)0x8000;     //[8*128*2]
   textEkran = (unsigned char *)0x8a00; //[40*16/2]
   EALLOW;
   // Okres licznika T2
   CpuTimer2Regs.PRD.all      =      System_Clk      *
Timer2ISR_Period;
   EDIS;
}
extern "C"
{
   int _system_pre_init()
   {
      EALLOW;
      WdRegs.WDWCR.all = 0x68;
      EDIS;
      return (1);
   }
}
#endif

#ifdef TMSLAB_WIN
void EnableInterrupts()
{
}
void SetUpPeripherials()
{
}
void Timer2Isr()
{
   if (++preScale == Tp_ms)
   {
      preScale = 0;
      Tim++;
   }
}
#endif

void InitData()
{
   for (int a = 0; a < (128 * 8); a++)
      ekran[a] = 0;
   for (int a = 0; a < (40 * 16); a++)
      textEkran[a] = ' ';
#ifndef CPP_EXAMPLE
   for (int a = 0; a < MaxObj; a++)
   {
      s[a] = (rand() & 0x1f) + 1;
      x[a] = rand() & 0x03ff;
      if (x[a] > 239 * 4 - s[a])
         x[a] -= 239 * 4 - s[a];
      y[a] = rand() & 0x1ff;
      if (y[a] > 127 * 4 - s[a])
         y[a] -= 127 * 4 - s[a];
      dx[a] = ((rand() & 0x1) ? 1 : -1) * (((rand() & 0x7)
+ 2) >> 1);
      dy[a] = ((rand() & 0x1) ? 1 : -1) * (((rand() & 0x7)
+ 2) >> 1);
   }
#endif
}
void ClearScreen()
{

   for (int a = 0; a < (128 * 8); a++)
      ekran[a] = 0;
}

void DrawPixels(int Key)
{
#ifdef CPP_EXAMPLE
   for (int c = 0; c < MaxObj; c++)
   {
      objects[c].move(Key, Key);
      objects[c].draw();
   }
#else
   for (int c = 0; c < MaxObj; c++)
   {
      y[c] += dy[c] + 6 - Key;
      x[c] += dx[c] + 6 - Key;
      if (x[c] < 0)
         x[c] += 239 * 4 - s[c];
      if (y[c] < 0)
         y[c] += 127 * 4 - s[c];
      if (x[c] > 239 * 4 - s[c])
         x[c] -= 239 * 4 - s[c];
      if (y[c] > 127 * 4 - s[c])
         y[c] -= 127 * 4 - s[c];
      long sdec = (((x[c] - 120 * 4L) * (x[c] - 120 * 4L))
>> 13) + (((y[c] - 58 * 4L) * (y[c] - 58 * 4L)) >> 9);
      long size = s[c] - sdec;
      if (size < 0)
         size = 0;
      size = s[c] - size;

      for (int b = y[c] >> 2; b < (y[c] + size) >> 2; b++)
         for (int a = x[c] >> 2; a < (x[c] + size) >> 2;
a++)
            SetPixel(ekran, a, b);
   }
#endif
}
```