PHỤ LỤC

# **I.Firmware(Raspberry Pi Pico RP2040):**

#include <SimpleKalmanFilter.h>

#include <stdio.h>

#include<string.h>

SimpleKalmanFilter filter(2, 2, 0.001);

double pulse; //biến đếm xung

double speed, Setpoint = 0; //tốc độ thực tế , tốc độ đặt

double E, E1, E2;//sai số e(k),e(k-1),e(k-2)

double alpha, gama, beta;

double Output, LastOutput; //tín hiệu điều khiển u(k),u(k-1)

const int IN1\_PIN =7; //IN1

const int IN2\_PIN =6; //IN2

const int SPEED\_PIN = 8; //ENA

int enco1 = 2; //Chân đọc Encoder kênh A

int enco2 = 3; //Chân đọc Encoder kênh B

int waittime = 100; // Thời gian lấy mẫu

double T = (waittime \* 1.0)/1000;

float Kp\_temp=0,Ki\_temp=0,Kd\_temp=0,Setpoint\_temp=0;

unsigned long counttime; //đếm thời gian

unsigned long present;

float Kp = 0, Kd = 0, Ki = 0; // Biến chứa thông số Ki Kp Kd từ C# Winform

String Direction;// biến chứa chiều quay

void setup()

{

pinMode(enco1, INPUT);//chân đọc encoder

pinMode(enco2, INPUT);

pinMode(SPEED\_PIN, OUTPUT);//chân PWM

pinMode(IN1\_PIN, OUTPUT);//chân DIR1

pinMode(IN2\_PIN, OUTPUT);//chân DIR2

speed=0;

E = 0 ;

E1 = 0;

E2 = 0;

Output=0;

LastOutput=0;

Serial.begin(9600);

attachInterrupt(digitalPinToInterrupt(enco1), PulseCount, RISING); //khi có cạnh lên //ở enco1, ngắt ngoài xảy ra sẽ thực hiện Chuong trình PulseCount

}

void PulseCount()

{

pulse++;

}

void loop(){

//nhận chuỗi dữ liệu từ C# Winform

String data ;

while (Serial.available() > 0) {

char c = Serial.read();

data += c;

delay(5);

}

//Loại bỏ kí tự " " ở cuối chuôĩ

data.trim();

if (data == NULL){

goto after\_get\_data;

}

else{

const int maxTokens = 10; // Số lượng token tối đa

char\* tokens[maxTokens];

int numTokens = 0;

char\* dataPtr = const\_cast<char\*>(data.c\_str()); // Chuyển đổi kiểu dữ liệu

char\* token = strtok(dataPtr, " ");

while (token != NULL && numTokens < maxTokens) {

tokens[numTokens] = token;

numTokens++;

token = strtok(NULL, " ");

}

Kp=atof(tokens[0]);

Ki=atof(tokens[1]);

Kd=atof(tokens[2]);

Setpoint=atof(tokens[3]);

Direction= tokens[4];

Output=0;

analogWrite(SPEED\_PIN,0);

digitalWrite(IN1\_PIN, HIGH);

digitalWrite(IN2\_PIN, LOW);

}

after\_get\_data:

counttime = millis();

if (counttime - present >= waittime)

{

present = counttime;

speed = (pulse/(21.3\*11))\*(1/T)\*60;

Serial.println(String (speed));

pulse=0;

// Tính toán giá trị ngõ ra PID rời rạc

E = Setpoint - speed;

alpha = 2\*T\*Kp + Ki\*T\*T + 2\*Kd;

beta = T\*T\*Ki -4\*Kd -2\*T\*Kp;

gama = 2\*Kd;

Output = (alpha\*E + beta\*E1 + gama\*E2 +2\*T\*LastOutput)/(2\*T);

if(Output>255){

Output=255;}

if(Output<0){

Output=0;}

LastOutput = Output;

E2=E1;

E1=E;

}

//điều chỉnh chiều quay

analogWrite(SPEED\_PIN, Output);

if(Direction =="Thuan"){

digitalWrite(IN1\_PIN, HIGH);

digitalWrite(IN2\_PIN, LOW);

}

if(Direction =="Nghich"){

digitalWrite(IN1\_PIN, LOW);

digitalWrite(IN2\_PIN, HIGH);

}

}

# **II.Sotfware(Visual Studio 2022):**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Deployment.Application;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO.Ports;

using ZedGraph;

namespace WindowsFormsApp1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

String[] Baudrate = {"1200","2400","4800","9600","19200","38400","57600","115200" };

String[] Direction = { "Thuan", "Nghich" };

comboBox1.Items.AddRange(Baudrate);

comboBox3.Items.AddRange(Direction);

Control.CheckForIllegalCrossThreadCalls = false;

}

private void Form1\_Load(object sender, EventArgs e)

{

textBox3.Text = "3.6";//"7.93867";

textBox4.Text = "24";// "6.61556";

textBox5.Text = "0.135";// "2.3816";

textBox6.Text = "170";

comboBox3.Text = "Thuan";

comboBox2.DataSource = SerialPort.GetPortNames();

comboBox1.Text = "9600";

GraphPane graph = zedGraphControl1.GraphPane;

graph.Title.Text = "graph";

graph.YAxis.Title.Text = "RPM";

graph.XAxis.Title.Text = "time";

RollingPointPairList list = new RollingPointPairList(500000);

LineItem line = graph.AddCurve("data", list, Color.Red, SymbolType.None);

graph.XAxis.Scale.Max = 10;

graph.XAxis.Scale.Min = 0;

graph.XAxis.Scale.MinorStep = 1;

graph.XAxis.Scale.MajorStep = 1;

graph.YAxis.Scale.Min = 0;

graph.YAxis.Scale.Max = 300;

graph.YAxis.Scale.MinorStep = 1;

graph.YAxis.Scale.MajorStep = 1;

zedGraphControl1.AxisChange();

graph.XAxis.ResetAutoScale(zedGraphControl1.GraphPane, CreateGraphics());

}

double tong = 0;

public void draw(double line) {

LineItem duongline = zedGraphControl1.GraphPane.CurveList[0] as LineItem;

if (duongline == null) {

return;

}

IPointListEdit list = duongline.Points as IPointListEdit;

if (list == null)

{

return;

}

list.Add(tong,line);

zedGraphControl1.AxisChange();

zedGraphControl1.Invalidate();

tong += 0.1;

}

private void button2\_Click(object sender, EventArgs e)

{

if (!serCOM.IsOpen)

{

MessageBox.Show("NOT CONNECTED YET!");

}

else

{

serCOM.Write("OFF");

}

}

private void button1\_Click(object sender, EventArgs e)

{

if (!serCOM.IsOpen) {

MessageBox.Show("NOT CONNECTED YET!");

}

else {

serCOM.Write("ON");

}

}

private void button3\_Click(object sender, EventArgs e)

{

if (!serCOM.IsOpen)

{

MessageBox.Show("CONNECTED!");

button3.Text = "DISCONNECT";

serCOM.PortName = comboBox2.Text;

serCOM.BaudRate = Convert.ToInt32(comboBox1.Text);

serCOM.Open();

serCOM.Write("0.1 0.1 0 0 Thuan");

}

else

{

MessageBox.Show("DISCONNECTED!");

button3.Text = "CONNECT";

serCOM.Write("0.1 0.1 0 0 Thuan");

serCOM.Close();

}

}

private void button4\_Click(object sender, EventArgs e)

{

if (!serCOM.IsOpen) {

serCOM.Open();

}

serCOM.Write("0.1 0.1 0 0 Thuan");

serCOM.Close();

Application.Exit();

}

private void button5\_Click(object sender, EventArgs e)

{

if (!serCOM.IsOpen)

{

MessageBox.Show("NOT CONNECTED YET!");

}

else {

String data = textBox1.Text;

serCOM.Write(text: data);

MessageBox.Show("UPDATED!");

}

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void comboBox2\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void serCOM\_DataReceived(object sender, SerialDataReceivedEventArgs e)

{

String data1 = "";

data1 = serCOM.ReadLine();

int len = data1.Length;

if( len > 0){

textBox2.Text = data1;

double data2;

if (double.TryParse(data1, out data2))

{

// Conversion successful, use parsedValue

Invoke(new MethodInvoker(() => draw(data2)));

}

else

{

MessageBox.Show("WRONG FORMAT!");

// Handle invalid input (e.g., show an error message)

}

// Invoke(new MethodInvoker(() => draw(Convert.ToDouble(data1))));

}

}

private void maskedTextBox2\_MaskInputRejected(object sender, MaskInputRejectedEventArgs e)

{

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

}

private void zedGraphControl1\_Load(object sender, EventArgs e)

{

}

private void groupBox1\_Enter(object sender, EventArgs e)

{

}

private void textBox3\_TextChanged(object sender, EventArgs e)

{

}

private void textBox4\_TextChanged(object sender, EventArgs e)

{

}

private void textBox5\_TextChanged(object sender, EventArgs e)

{

}

private void button6\_Click(object sender, EventArgs e)

{

if (!serCOM.IsOpen)

{

MessageBox.Show("NOT CONNECTED YET!");

}

else

{

String data2 = textBox3.Text+" "+textBox4.Text+" "+textBox5.Text+" "+textBox6.Text+" "+comboBox3.Text;

serCOM.Write(text: data2);

MessageBox.Show("UPDATED!");

}

}

private void textBox6\_TextChanged(object sender, EventArgs e)

{

}

private void comboBox3\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void label8\_Click(object sender, EventArgs e)

{

}

private void label1\_Click(object sender, EventArgs e)

{

}

}

}