Fuzzy Control for Two-wheeled Self-balancing robot

Pham Dinh Duong

Assoc. Prof. Nguyen Hoai Nam

Hanoi University of Science and Technology

Outlines

1. Introduction (1)

Application

- Self-balancing two-wheeled vehicles are a type of smart personal vehicle, with a modern control system, currently commonly used in many developed countries.

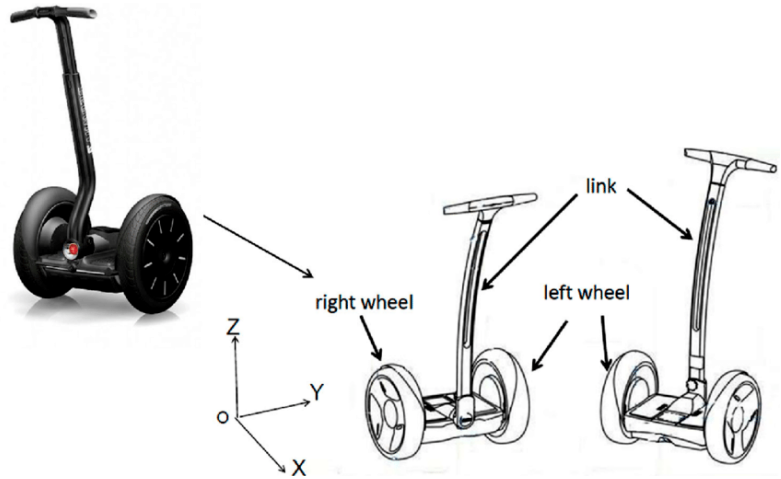


Figure: A Two-wheeled Self-balancing vehicle

1. Introduction (2)

The TWSR in the laboratory

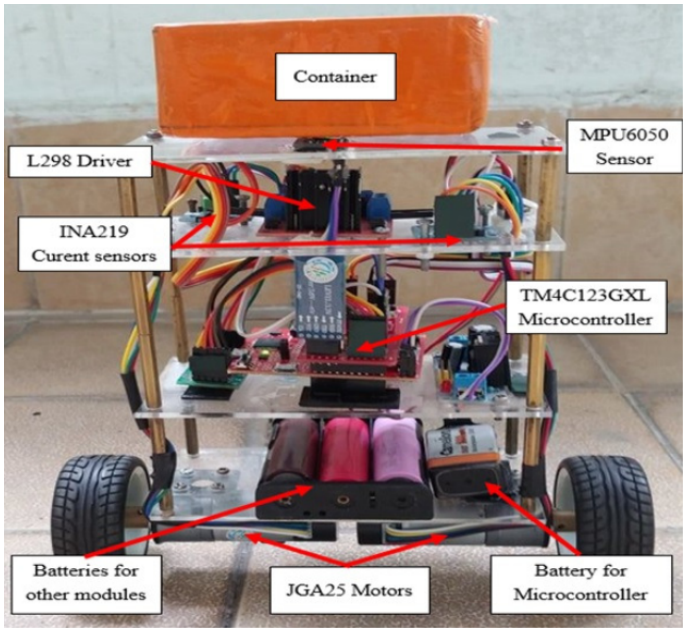


Figure: The fabricated TWSR in the LAB

2. Hardware Design(1)

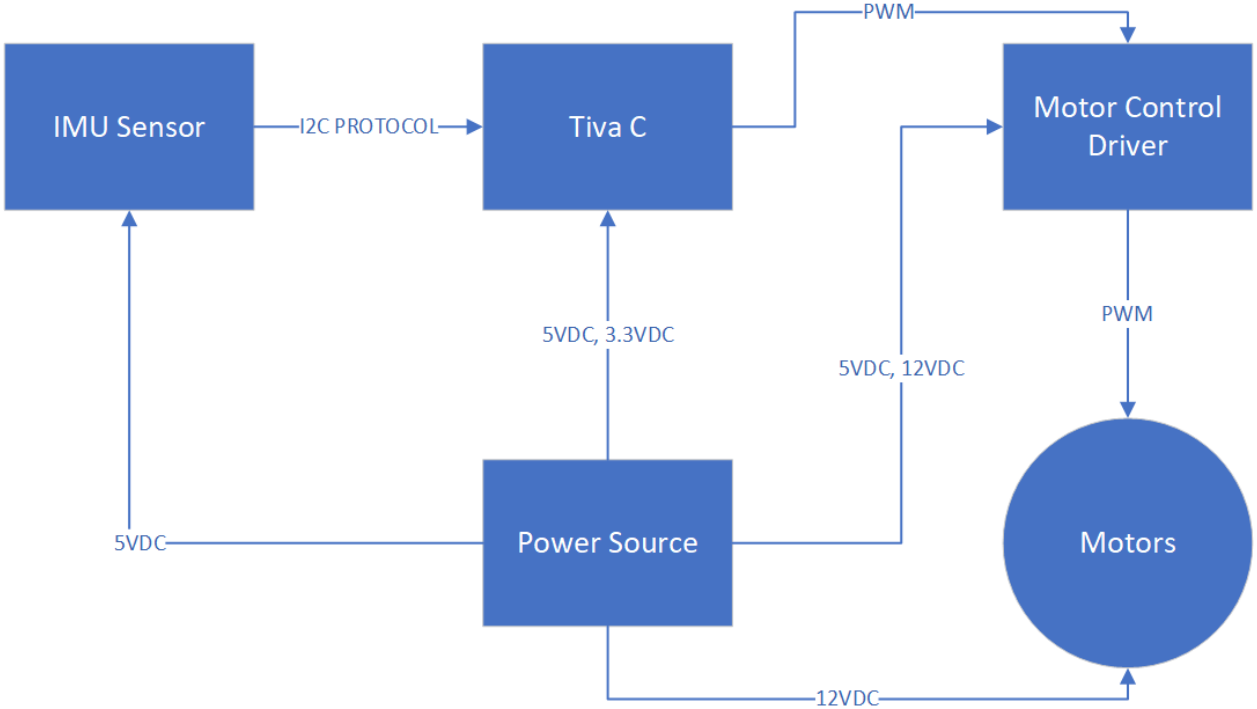


Figure: Hardware block diagram

2. Hardware Design(2)

TI C MCU

High Performance TM4C123GH6PM MCU:

- 80MHz 32-bit ARM Cortex-M4-based microcontrollers CPU
- 256KB Flash, 32KB SRAM, 2KB EEPROM
- Two Controller Area Network (CAN) modules (requires CAN transceivers)
- USB 2.0 Host/Device/OTG + PHY
- Dual 12-bit 2MSPS ADCs, motion control PWMs 8 UART, 6 I2C, 4 SPI
- On-board In-Circuit Debug Interface (ICDI)
- USB Micro-B plug to USB-A plug cable

2. Hardware Design(3)

TI C MCU

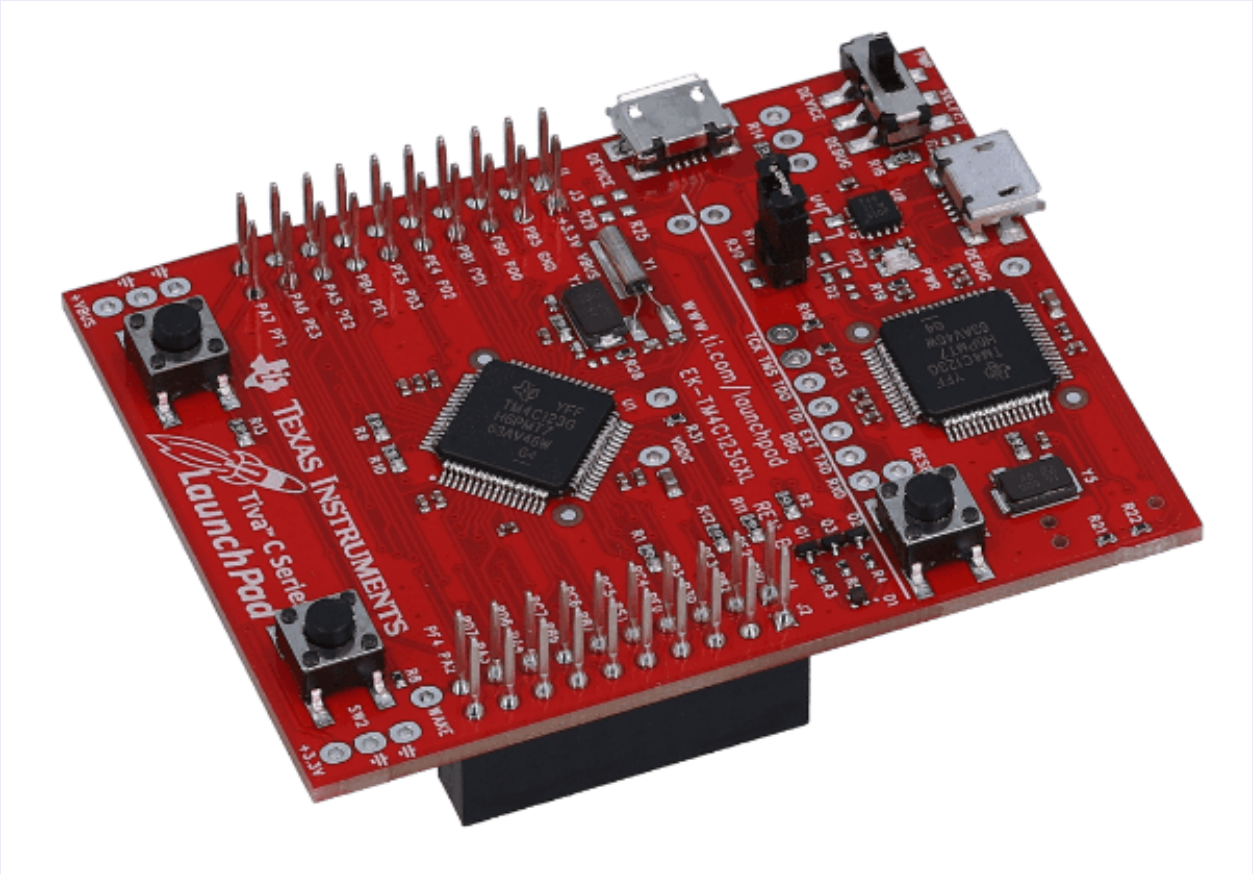


Figure: A TI CMU

2. Hardware Design(4)

Motor Driver L298N

- Driver: L298N integrates two H-bridge circuits
- Control voltage: +5 V - +35 V
- Maximum current for each H-bridge: 2A
- Control signal voltage: +5 V - +7 V
- Control signal current: 0 - 36mA

Figure: Motor driver L298N

2. Hardware Design(5)

IMU MPU 6050

- Digital-output X-, Y-, and Z-Axis angular rate sensors (gyroscopes) with a user-programmable fullscale range of ± 250, ± 500, ± 1000, and $\pm 2000^\circ/\text{sec}$
- Digital-output triple-axis accelerometer with a programmable full-scale range of $\pm 2g$, $\pm 4g$, $\pm 8g$ and $\pm 16g$
- Integrated 16-bit ADCs

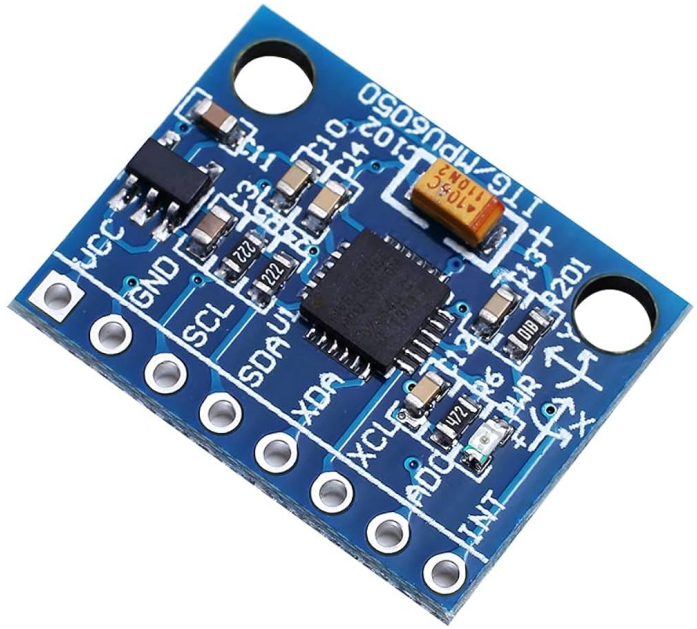


Figure: PMU 6050

2. Hardware Design(6)

Motor JGA25

- Gear ratio 21.3:1
- Maximum current under load: 1.3A
- No-load speed: 280RPM (280 rpm)
- Maximum speed under load: 220RPM (220 rounds per minute)
- Rated pulling force Moment: 0.5KG.CM
- Maximum Moment Force: 1.7KG.CM

Figure: Motor JGA 25

3. Simulation and Experimen
3.1. Mathematical model(1)

Notation

- x: Position of TWSR
- θ: Pitch angle of pendulum body
- ψ: Yaw angle of the TWSR
- d: Distance between the left and right wheels
- l: Length from the center of the pendulum to the wheel axis
- r: Radius of the wheel
- M: Mass of the pendulum (without wheels)
- m: Mass of the left (right) wheel

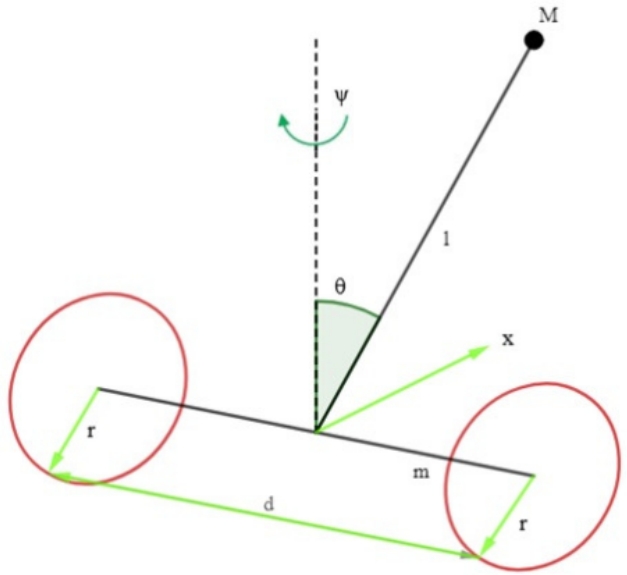


Figure: Schematic diagram

3.1. Mathematical model(2)

Notation

- J: Mass moment of inertia (MOI) of the wheel with respect to (w. r. t.) the wheel axis
- K: MOI of the wheel w. r. t. the vertical axis
- i_L, i_R: Motor's left (right) current
- C: Coefficient of viscous friction on the wheel axis
- I_1, I_2, I_3: MOI of the pendulum w. r. t. the frame at the center of the pendulum. of the pendulum

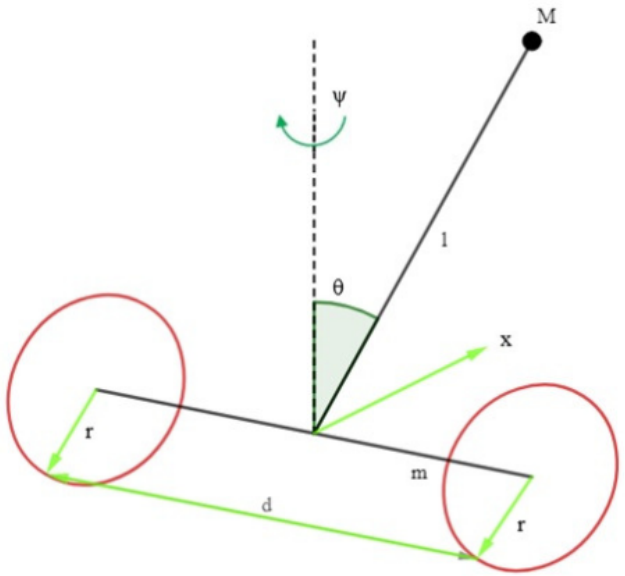


Figure: Schematic diagram

3.1. Mathematical model(3)

Motion equation:

1. $(M + 2m + 2J/r^2)\ddot{x} - Ml(\dot{\psi}^2 + \dot{\theta}^2)sin\theta + Mlcos\theta\ddot{\theta} + 2c_\alpha(\dot{x}/r - \dot{\theta})/r = K_m(i_L + i_R)/r$
2. $Mlcos\theta\ddot{x} + (Ml^2 + I_2)\ddot{\theta} - Ml^2\dot{\psi}^2 sin\theta cos\theta - Mglsin\theta - 2c_\alpha(\dot{x}/r - \dot{\theta}) = -K_m(i_L + i_R)$
3. $(I_3 + 2K + md^2/2 + Jd^2/(2r^2) + Ml^2sin^2\theta)\ddot{\psi} + Ml(\dot{x} + 2l\dot{\theta}cos\theta)\dot{\psi}sin\theta = (i_R - i_L)K_m d/(2r)$

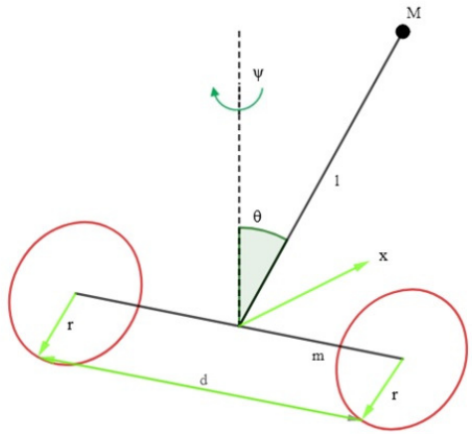


Figure: Schematic diagram

3.2. Model Matlab simulation(1)

From the mathematical model, the object model is simulated using a Matlab S-function (to limit the integration components in the simulation).

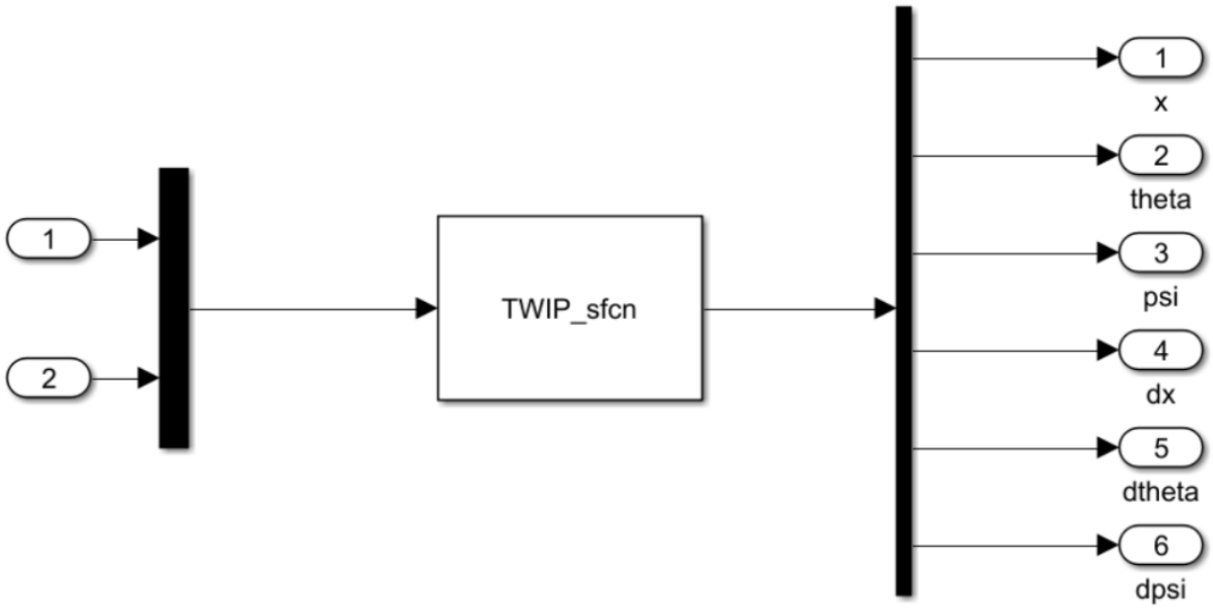


Figure: Model Matlab S-function

3.2. Model Matlab simulation(2)

Examine and Identify the system using a Step input.

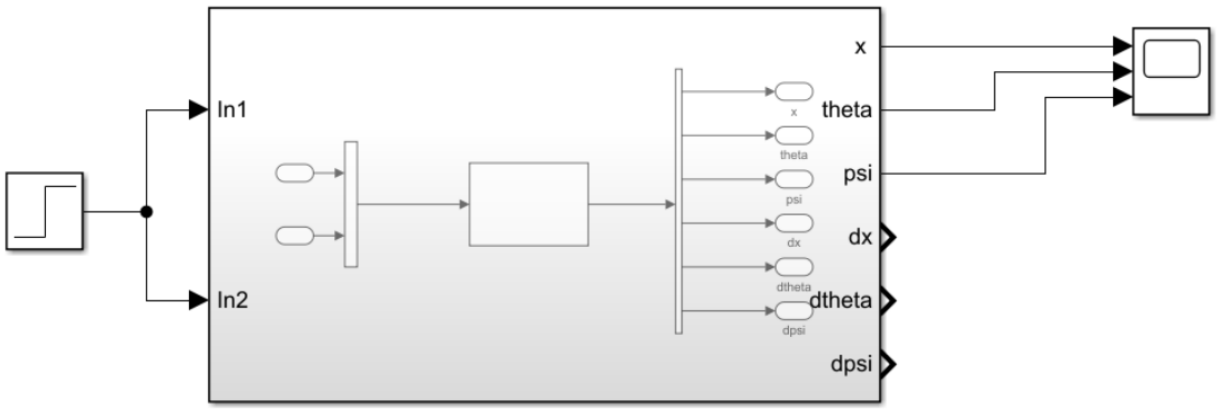


Figure: Identify System

3.2. Model Matlab simulation(3)

System responds to step input:

- Applying a current of 0.7 (A) to both wheels, position x increases and is approximated as a linear function, $\psi = 0$ so the robot doesn't change direction
- θ goes to $-\pi$ due to gravity

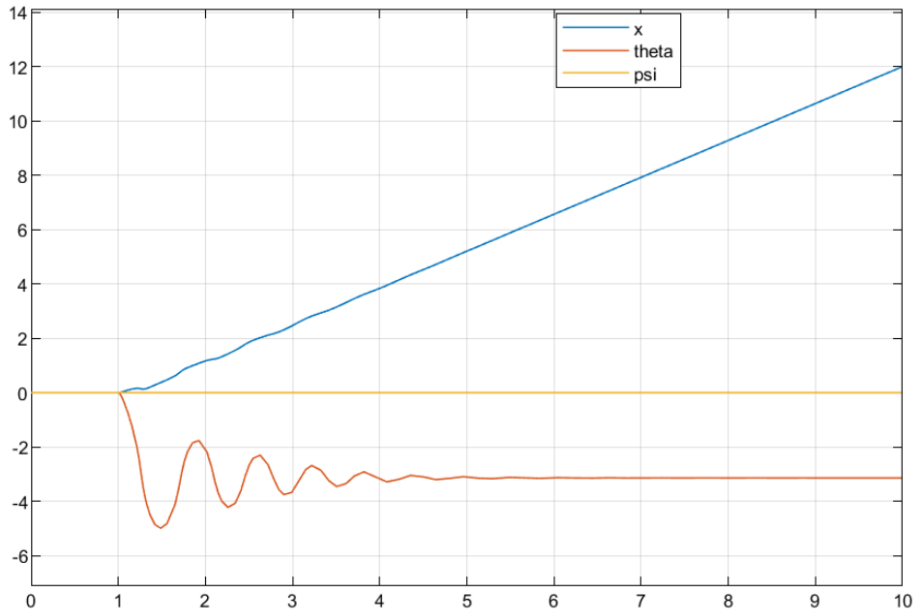


Figure: Schematic diagram

3.3. PD controller based on LQR solution (1)

Parameters of model

$M = 0.765(kg); m = 0.03(kg); l = 0.10(m); d = 0.195(m); r = 0.033(m); C = 0.005$

PD Controller based on LQR matrix

It is easy to obtain a LQR matrix from fully-model known (linearization model):

$$\begin{bmatrix} -0.8660 & -3.6762 & -0.7071 & -1.6965 & -0.5976 & -0.4891 \\ -0.8660 & -3.6762 & 0.7071 & -1.6965 & -0.5976 & 0.4891 \end{bmatrix}$$

Based on that, the baseline PD controllers are chosen as follows:

- Position-x controller: $K_P = -0.8660; K_D = -1.6965$
- θ angle controller: $K_P = -3.6762; K_D = -0.5976$
- ψ angle controller: $K_P = -0.7071; K_D = -0.4891$

3.3. PD controller based on LQR solution (2)

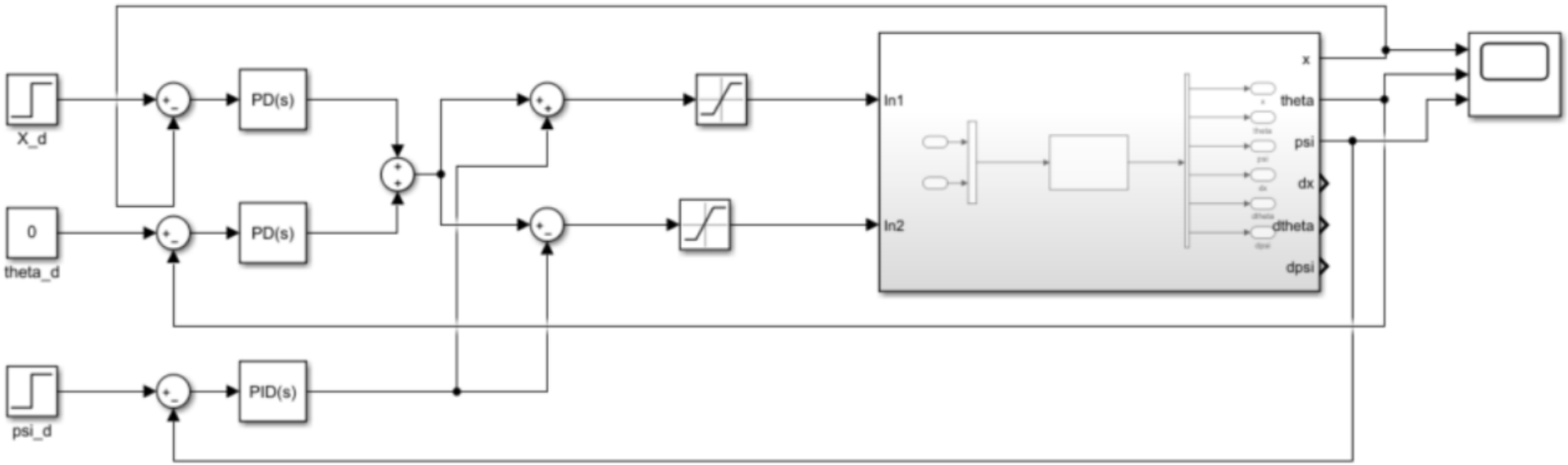


Figure: PD controller simulation schematic

3.3. PD controller based on LQR solution (3)

- In simulation, we set the setpoint of x: $x_d = 0.5(m)$, the settling time $= 5(s)$.
- In the experiment on the real robot, we set the setpoint of x: $x_d = 1(m)$, the settling time $= 15(s)$.

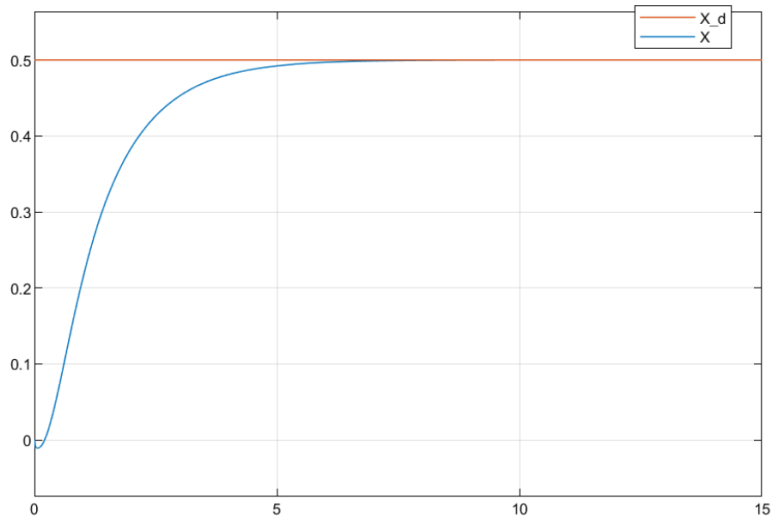


Figure: Tracking of position x in simulation

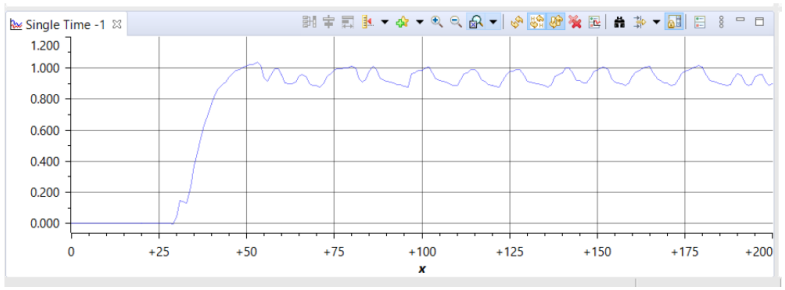


Figure: Tracking of position x in experiment

3.3. PD controller based on LQR solution (4)

- In simulation, we set the setpoint of ψ: $\psi_d = 0.8(rad)$, the settling time $= 5(s)$.
- In the experiment on the real robot, we set the setpoint of ψ: $\psi_d = 1(rad)$, the settling time $= 15(s)$.

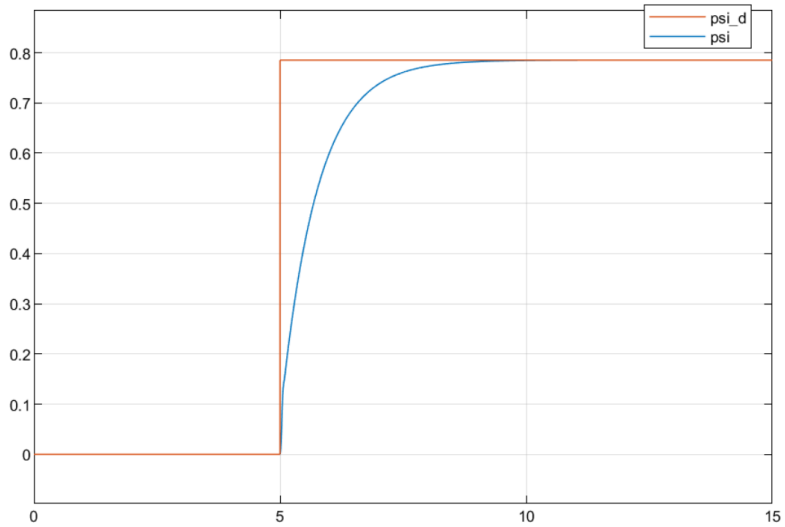


Figure: Tracking of angle ψ in simulation

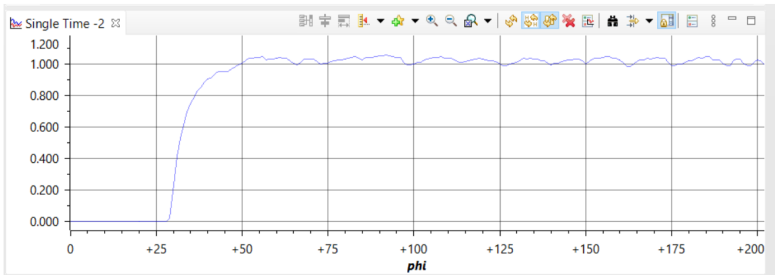


Figure: Tracking of angle ψ in experiment

3.3. PD controller based on LQR solution (5)

- In simulation, we set the setpoint of θ: $\theta_d = 0(rad)$ (maintain the robot standing vertically), the settling time $= 3$(s).
- In the experiment on the real robot, we set the setpoint of θ: $\theta_d = 0(rad)$ (keep the robot vertical), the settling time $= 10$(s).

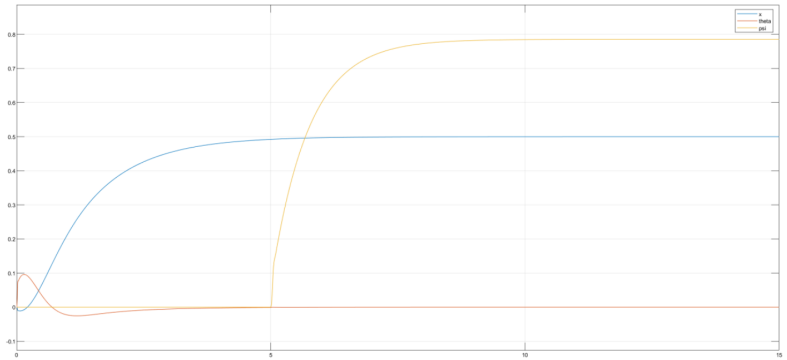


Figure: Tracking of angle θ in simulation

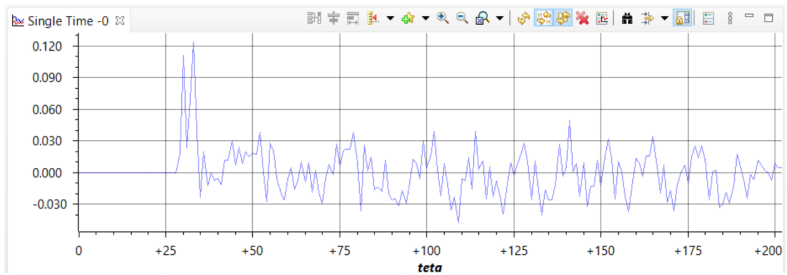


Figure: Tracking of angle θ in experiment

3.4. Fuzzy controller for TWSR (1)
3.4.1. Fuzzy Logic Controller introduction (1)

- Fuzzy logic control (FLC) is a heuristic approach that embeds the knowledge and key elements of human thinking in the design of nonlinear controllers
- Due to the good performance of the PD controller, we propose 6 inputs for FLC: $e_x, \dot{e}_x, e_\theta, \dot{e}_\theta, e_\psi, \dot{e}_\psi$ where: $e_x = x_d - x$; $e_\theta = \theta_d - \theta$; $e_\psi = \psi_d - \psi$

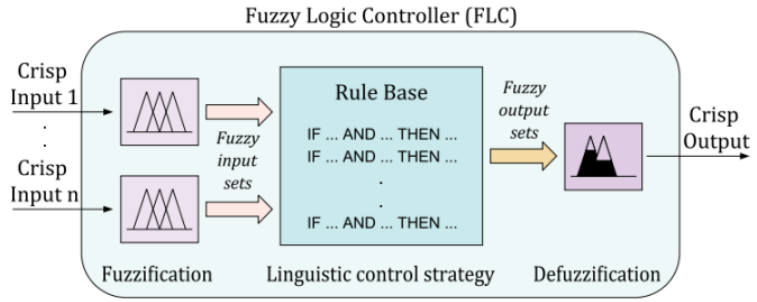


Figure: A Fuzzy Logic Controller structure

3.4.2. FLC simulation schematic (1)

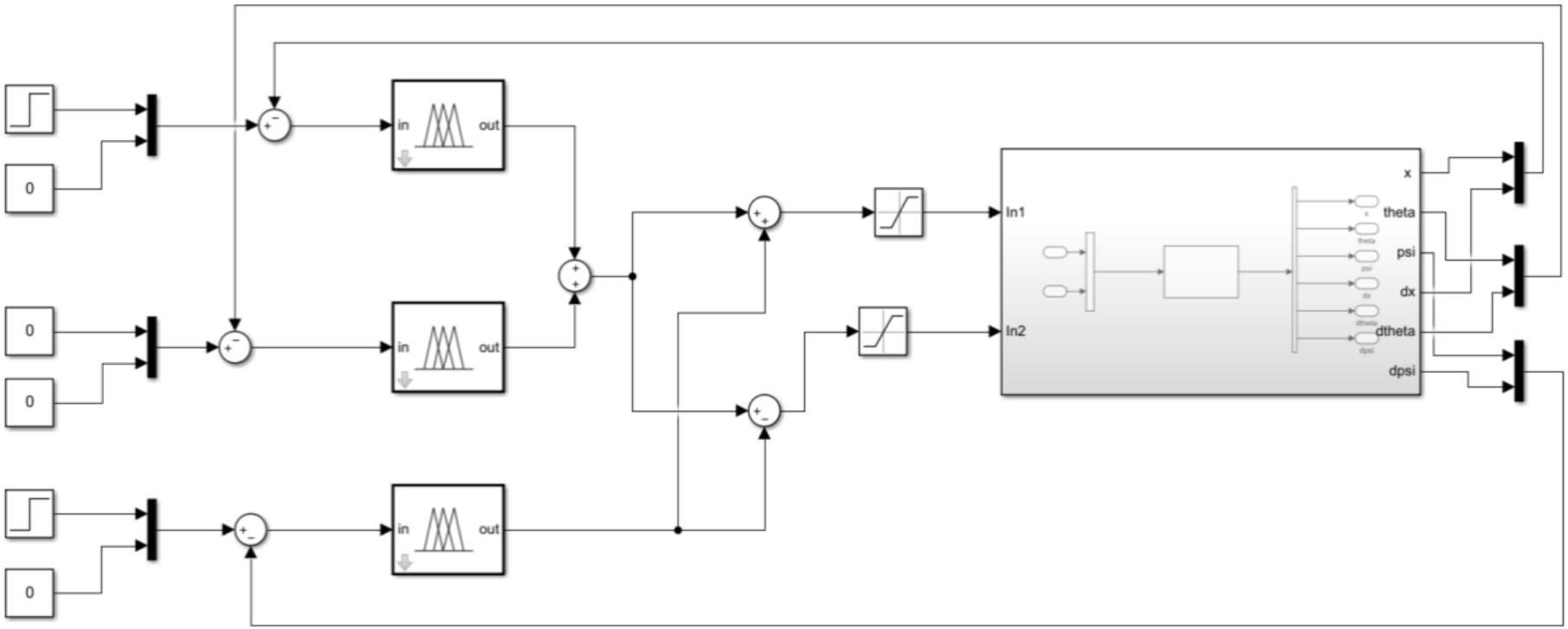
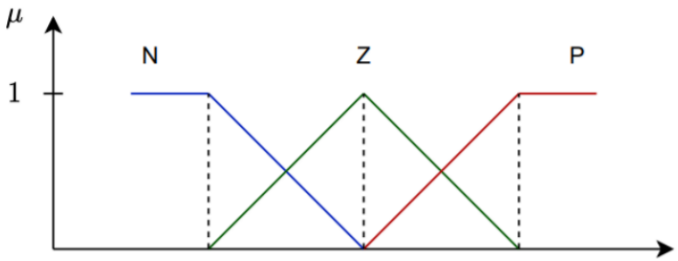


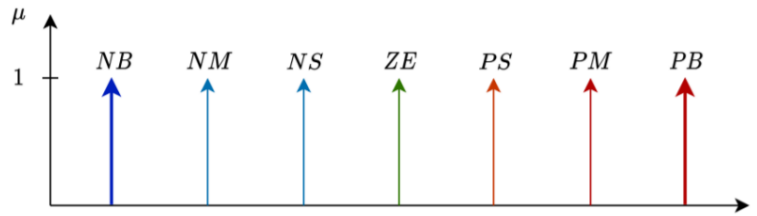
Figure: FLC simulation for TWSR

3.4.3. The proposed Sugeno-FLC parameters and functions for TWSR (1)



-2	0	2	$e_x(m)$	
-0.7	0	0.7	$e_{\dot{x}}(m/s)$	$FLCx$
-0.3	0	0.3	e_θ	
-2.8	0	2.8	$e_{\dot{\theta}}(1/s)$	$FLC\theta$
$-\pi/2$	0	$\pi/2$	e_ψ	
-1.2	0	1.2	$e_{\dot{\psi}}(1/s)$	$FLC\psi$

Figure: Membership functions of inputs



-2.9	-1.19	-0.54	0	0.54	1.19	2.9	$y_x(A)$	$FLCx$
-2.78	-1.67	-0.57	0	0.57	1.67	2.78	$y_\theta(A)$	$FLC\theta$
-1.70	-1.1	-0.52	0	0.52	1.1	1.70	$y_\psi(A)$	$FLC\psi$

Figure: Membership functions of outputs

z_dot\z	N	Z	P
N	NB	NS	NS
Z	NM	ZE	PM
P	PS	PS	PB

Figure: The fuzzy control rules ($z : x, \theta, \psi$)

3.4.4. The FLC simulation and experiment results (1)

- In simulation, we set the setpoint of x: $x_d = 0.68(m)$, the settling time $= 16(\text{s})$.
- In the experiment, we focus on keeping the robot standing upright, so for convenience, we set the setpoint of x: $x_d = 0(m)$, the settling time $= 10(\text{s})$. In fact, the proposed FLC could handle well when x_d is chosen arbitrarily.

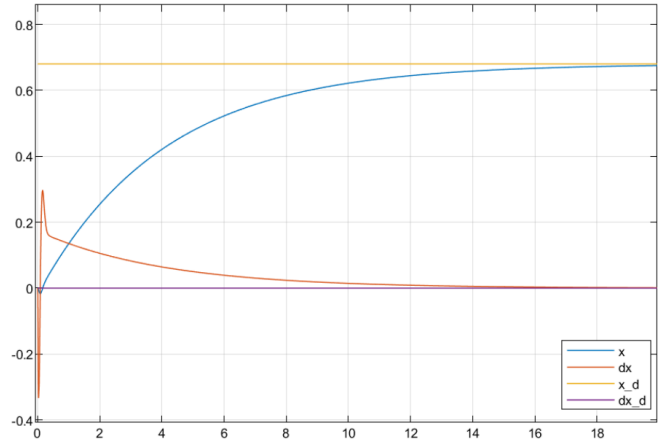


Figure: The tracking of position x in simulation

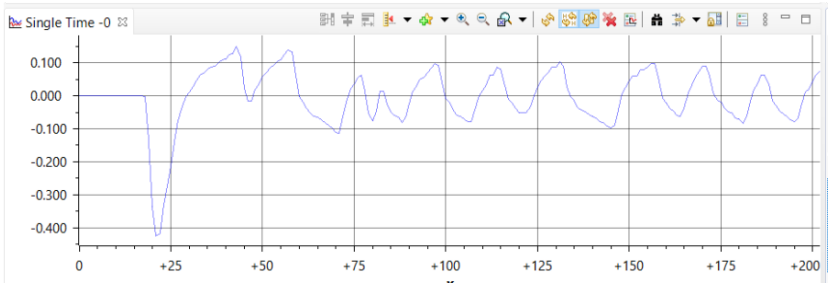


Figure: The tracking of position x in experiment

3.4.4. The FLC simulation and experiment results (2)

- In both simulation and experiment, we set the setpoint of θ: $\theta_d = 0(rad)$ for the robot to be kept vertically.
- The results illustrate that: in both cases, θ is stable at 0 (rad), indicating that the robot remains standing upright.

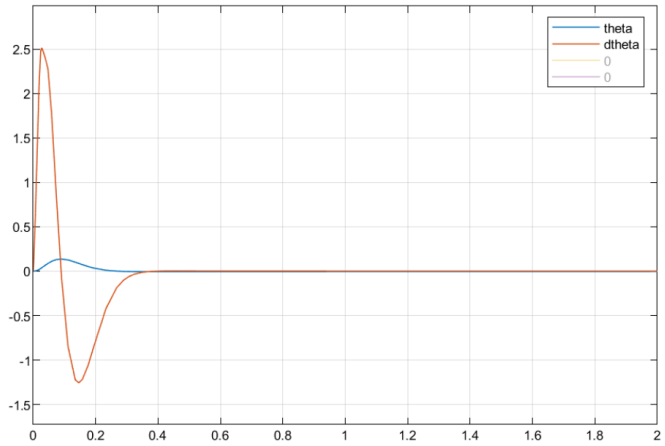


Figure: The tracking of angle θ in simulation

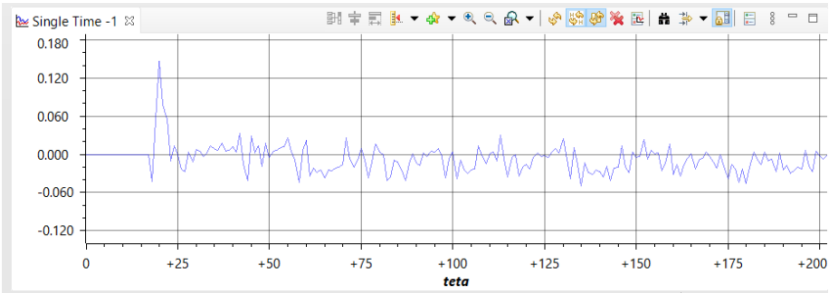


Figure: The tracking of angle θ in experiment

3.4.4. The FLC simulation and experiment results (3)

- In simulation, we set the setpoint of ψ: $\psi_d = \pi/2(rad)$, the settling time $= 2$(s).
- In the experiment, we focus on keeping the robot standing upright, so for convenience, we set the setpoint of ψ: $\psi_d = 0(rad)$, the settling time $=$ 20(s). In fact, the proposed FLC could handle well when ψ_d is chosen arbitrarily.

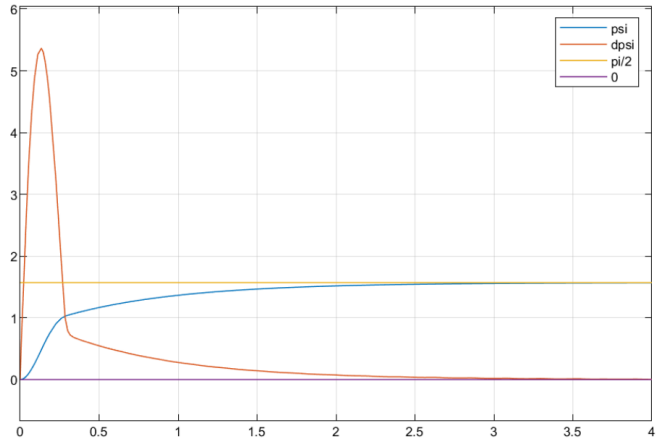


Figure: The tracking of angle ψ in simulation

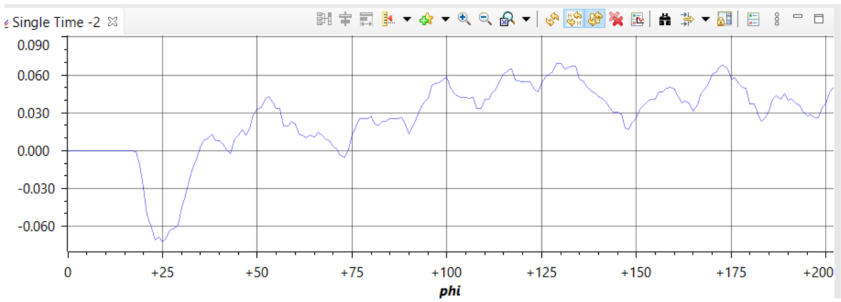


Figure: The tracking of angle ψ in experiment

3.4.4. The FLC simulation and experiment results (4)

The experiment video

We cried happily when the robot is stable and kept vertically!!! Check this link for our videos!!!

3.5. The embedded software design (1)
3.5.1. Program flowchart (1)

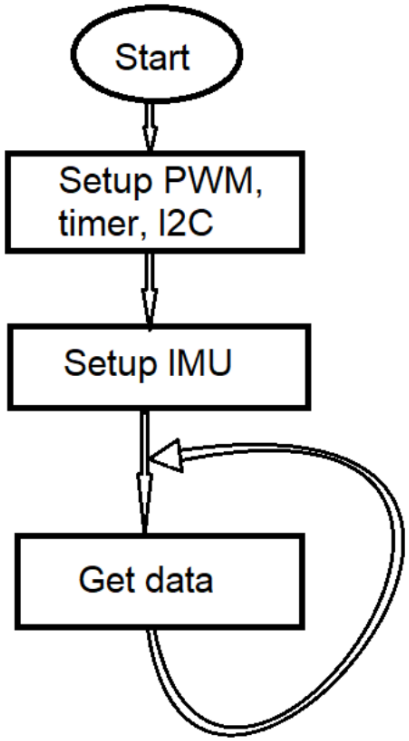


Figure: Boot up and Data-acquisition

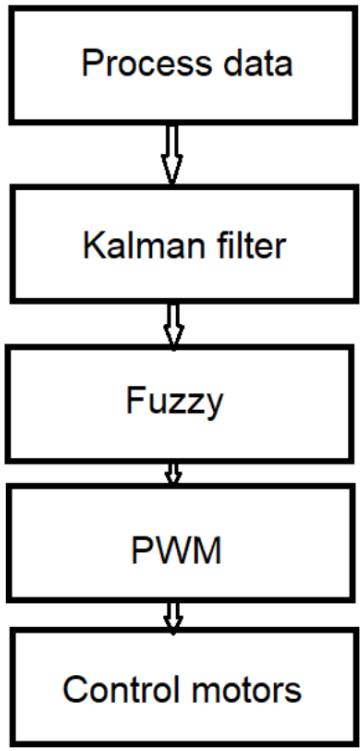


Figure: The processes in the interrupt service routine

3.5.2. Sources of FLC controller (1)

```
#ifdef fuzzy
 // fuzzy teta
 float teta_lim = 0.3;
 float dteta_lim = 2.5;
 a[1] = prod(trimf(-1000, -teta_lim, 0, teta), trimf(-1000, -dteta_lim, 0, teta_dot)); // teta < 0, teta_dot < 0
 a[2] = prod(trimf(-1000, -teta_lim, 0, teta), trimf(-dteta_lim, 0, dteta_lim, teta_dot)); // teta < 0, teta_dot = 0
 a[3] = prod(trimf(-1000, -teta_lim, 0, teta), trimf(0, dteta_lim, 1000, teta_dot)); // teta < 0, teta_dot > 0
 a[4] = prod(trimf(-teta_lim, 0, teta_lim, teta), trimf(-1000, -dteta_lim, 0, teta_dot)); // teta = 0, teta_dot < 0
 a[5] = prod(trimf(-teta_lim, 0, teta_lim, teta), trimf(-dteta_lim, 0, dteta_lim, teta_dot)); // teta = 0, teta_dot = 0
 a[6] = prod(trimf(-teta_lim, 0,teta_lim, teta), trimf(0, dteta_lim, 1000, teta_dot)); // teta = 0, teta_dot > 0
 a[7] = prod(trimf(0, teta_lim, 1000, teta), trimf(-1000, -dteta_lim, 0, teta_dot)); // teta > 0, teta_dot < 0
 a[8] = prod(trimf(0, teta_lim, 1000, teta), trimf(-dteta_lim, 0, dteta_lim, teta_dot)); // teta > 0, teta_dot = 0
 a[9] = prod(trimf(0, teta_lim, 1000, teta), trimf(0, dteta_lim, 1000, teta_dot)); // teta > 0, teta_dot > 0
 a[0] = 0;
for(t=1; t<10; t++) a[0] += a[t];
 y_teta = (a[1]*g_teta[1] + a[2]*g_teta[2] + a[3]*g_teta[3] + a[4]*g_teta[4] + a[5]*g_teta[5] + a[6]*g_teta[6] +
a[7]*g_teta[7] + a[8]*g_teta[8] + a[9]*g_teta[9])/a[0];
 // fuzzy X
 float x_lim = 2;
 float dx_lim = 0.7;
 a[1] = prod(trimf(-1000, -x_lim, 0, x), trimf(-1000, -dx_lim, 0, x_dot)); // teta < 0, teta_dot < 0
 a[2] = prod(trimf(-1000, -x_lim, 0, x), trimf(-dx_lim, 0, dx_lim, x_dot)); // teta < 0, teta_dot = 0
 a[3] = prod(trimf(-1000, -x_lim, 0, x), trimf(0, dx_lim, 1000, x_dot)); // teta < 0, teta_dot > 0
 a[4] = prod(trimf(-x_lim, 0, x_lim, x), trimf(-1000, -dx_lim, 0, x_dot)); // teta = 0, teta_dot < 0
 a[5] = prod(trimf(-x_lim, 0, x_lim, x), trimf(-dx_lim, 0, dx_lim, x_dot)); // teta = 0, teta_dot = 0
 a[6] = prod(trimf(-x_lim, 0,x_lim, x), trimf(0, dx_lim, 1000, x_dot)); // teta = 0, teta_dot > 0
 a[7] = prod(trimf(0, x_lim, 1000, x), trimf(-1000, -dx_lim, 0, x_dot)); // teta > 0, teta_dot < 0
 a[8] = prod(trimf(0, x_lim, 1000, x), trimf(-dx_lim, 0, dx_lim, x_dot)); // teta > 0, teta_dot = 0
 a[9] = prod(trimf(0, x_lim, 1000, x), trimf(0, dx_lim, 1000, x_dot)); // teta > 0, teta_dot > 0
 a[0] = 0;
 for(t=1; t<10; t++) a[0] += a[t];
 y_x = (a[1]*g_x[1] + a[2]*g_x[2] + a[3]*g_x[3] + a[4]*g_x[4] + a[5]*g_x[5] + a[6]*g_x[6] + a[7]*g_x[7] +
a[8]*g_x[8] + a[9]*g_x[9])/a[0];
```

Figure: Source code of FLC controller (1)

```
//fuzzy psi
 float psi_lim = 3.14/2;
 float dpsi_lim = 0.5;
 a[1] = prod(trimf(-1000, -psi_lim, 0, psi), trimf(-1000, -dpsi_lim, 0, psi_dot)); // teta < 0, teta_dot < 0
 a[2] = prod(trimf(-1000, -psi_lim, 0, psi), trimf(-dpsi_lim, 0, dpsi_lim, psi_dot)); // teta < 0, teta_dot = 0
 a[3] = prod(trimf(-1000, -psi_lim, 0, psi), trimf(0, dpsi_lim, 1000, psi_dot)); // teta < 0, teta_dot > 0
 a[4] = prod(trimf(-psi_lim, 0, psi_lim, psi), trimf(-1000, -dpsi_lim, 0, psi_dot)); // teta = 0, teta_dot < 0
 a[5] = prod(trimf(-psi_lim, 0, psi_lim, psi), trimf(-dpsi_lim, 0, dpsi_lim, psi_dot)); // teta = 0, teta_dot = 0
 a[6] = prod(trimf(-psi_lim, 0,psi_lim, psi), trimf(0, dpsi_lim, 1000, psi_dot)); // teta = 0, teta_dot > 0
 a[7] = prod(trimf(0, psi_lim, 1000, psi), trimf(-1000, -dpsi_lim, 0,
psi_dot)); // teta > 0, teta_dot < 0
 a[8] = prod(trimf(0, psi_lim, 1000, psi), trimf(-dpsi_lim, 0, dpsi_lim, psi_dot)); // teta > 0, teta_dot = 0
 a[9] = prod(trimf(0, psi_lim, 1000, psi), trimf(0, dpsi_lim, 1000, psi_dot)); // teta > 0, teta_dot > 0
 a[0] = 0;
 for(t=1; t<10; t++) a[0] += a[t];
 y_psi = (a[1]*g_psi[1] + a[2]*g_psi[2] + a[3]*g_psi[3] + a[4]*g_psi[4] + a[5]*g_psi[5] + a[6]*g_psi[6] +
a[7]*g_psi[7] + a[8]*g_psi[8] + a[9]*g_psi[9])/a[0];
 l_sp1 = y_x + y_teta - y_psi;
 l_sp2 = y_x + y_teta + y_psi;
#endif
```

Figure: Source code of FLC controller (2)

4. Conclusion

The results of the project

- The proposed FLC for TWSR is validated in both simulation and experiment to guarantee the robot standing vertically, the stability of the system which is comparable to the baseline PD controller.
- The project shows the advantages of Fuzzy Control, with easy implementation, utilizing the experience of the designer by reviewing previous control schemes or knowledge about the model and behavior of the system.

Development direction

- Develop another FLC based on the Mamdani model.
- Develop the FLC with online update rules for the adaptive control system.


Thank you