An Obstacle Avoidance Two-Wheeled Self-Balancing Robot

Ryuichi Tsutada, Trong-Thuc Hoang, and Cong-Kha Pham
Graduate School of Informatics and Engineering, University of Electro-Communications, Tokyo, Japan
e-mail: tsutada@vlsilab.ee.uec.ac.jp, thuc@vlsilab.ee.uec.ac.jp, phamck@uec.ac.jp

***Abstract*—This paper introduces a Two-Wheeled Self-Balancing Robot (TWSBR) which is controlled to avoid obstacles. The TWSBR is a type of the inverted pendulum and is treated as an inherently unstable nonlinear system. Therefore, a continuous appropriate control is required to maintain the inverted state. The TWSBR consists of two DC motors with encoders and 6-axis sensor (accelerometer and gyroscope). All peripherals are connected to a 32-bit RISC-V soft microprocessor implemented on an FPGA, and all control circuits for the peripherals are also implemented on the same FPGA. An attitude control system of the TWSBR is provided through 3 Proportional-Integral- Differential (PID) controllers with a sensor fusion-based on a Kalman Filter, which is implemented on the 32-bit RISC-V soft microprocessor. The obstacle avoidance system of the TWSBR is based on a fuzzy control using multiple ultrasonic sensors. The 32-bit RISC-V soft microprocessor includes a 32-bit fixed-point (Q16.16) arithmetic instructions of addition, subtraction, multiplication, maximum and minimum as a custom instruction set architecture (ISA) extensions for calculation of a speed improvement. The software program is written in C language and compiled by the GNU GCC cross-compiler for the RISC-V ISA.**

***Index Terms*—Two-Wheeled Self-Balancing Robot (TWSBR), MPU-6500, Kalman Filter, PID controller, fuzzy controller, obstacle avoidance, RISC-V, fixed-point arithmetic**

I. INTRODUCTION

An inverted pendulum is an inherently unstable system and cannot be maintained in a balancing state without control. There are several types of inverted pendulums, such as truck-type inverted pendulums, wheel-type inverted pendulums and rotary-type inverted pendulums. In this paper, we focus on a two-wheeled self-balancing robot (TWSBR) based on the wheel-type inverted pendulum.

TWSBR can be controlled by many different methods. The most common method to control TWSBR is to use PID controller [1]. In [2], state feedback control by Linear Quadratic Regulator (LQR) is proposed. It is common that microprocessors such as ARM Cortex-M4 [3], STM32 [4] and AVR [5], [6] have been used to control TWSBR. Also, obstacle avoidance of robots has been researched for many years and many methods are adopted fuzzy inference. In [7], obstacle avoidance by multiple ultrasonic sensors for TWSBR is proposed.

RISC-V is an open-source instruction set architecture (ISA) and free to use, which is different from those microprocessor's ISA. In addition, RISC-V processors are used for controlling robots [8], [9]. In this paper, we propose a control and obstacle avoidance system of the TWSBR based on a 32-bit RISC-V soft microprocessor.

The rest of the paper is organized as follows: Section II pro- vides a system overview of the TWSBR. Section III describes the control system of the TWSBR. Section IV discusses the obstacle avoidance system of the TWSBR. Section V proposes the method of the fixed-point arithmetic. Section VI explains the implementation of the software program. Finally, Section VII and VIII shows the results and conclusion.

II. SYSTEM OVERVIEW

A. Self-Balancing Robot

Terasic's Self-Balancing Robot (Fig. 1) is a type of the TWSBR, which includes accelerometer, gyroscope, motor driver and encoders like other robotic kits [10]. However, it uses a FPGA board DE10-NANO with Cyclone V 5CSEBA6U2317 as a control board. Therefore, it has a very high configurability and ideal for implementing and experimenting with a new system. Fig. 2 shows a block diagram of the control system of the robot.

Figure 1. Terasic's self-balancing robot [10].

The Terasic's Self-Balancing Robot equips a Cyclone SoC FPGA, which means that the ARM processor is embedded in the FPGA. Therefore, there are two processor options available to control the Robot. One is to use the ARM processor and the other is to implement a soft NIOS II processor [11] in the FPGA. If user want to select ARM to control the robot, the FPGA will boot from the Micro SD

Manuscript received July 21, 2021; revised November 1, 2021.

doi: 10.18178/ijmerr.11.1.1-7

card and run the Linux by ARM processor to control the robot. If user want to select NIOS II processor to control the robot, the FPGA will boot from the configuration device (EPCS). Then, after FPGA is configured, the NIOS II processor will control the robot. Instead of using the ARM processor and soft NIOS II processor, we proposed a control system using a 32-bit RISC-V soft CPU as a microprocessor.

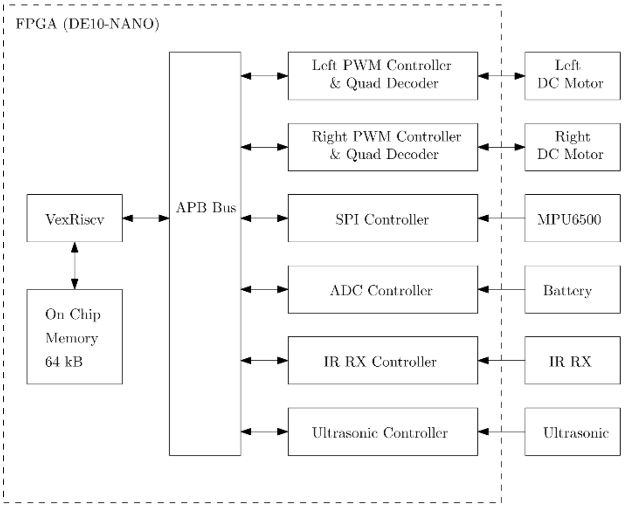

Figure 2. Block diagram of the control board.

B. Microprocessor

VexRiscv [12] is a 32-bit RISC-V soft microprocessor written in SpinalHDL [13] and its architecture is RV32IM. SpinalHDL is Scala [14] based hardware description language (HDL). Therefore, it can be implemented on FPGA. VexRiscv can also add a custom instruction as a plugin by SpinalHDL. Since VexRiscv does not have a floating-point unit (FPU), We implemented custom instructions of signed 32-bit fixed-point addition, subtraction, multiplication, minimum and maximum, where the sign part is 1-bit, the integer part is 15-bit, and the fractional part is 16-bit (Q16.16). In this work, the clock frequency of the whole system is set to 50 MHz, which is the internal clock of the FPGA board.

C. Peripherals

The following peripherals are used in the robot:

1) Motors: The robot has two DC geared motors (AS-LONG JGB37-520B). This motor has a speed reducer which can reduce the rotation speed and increase the torque. It is controlled by the motor driver device TB6612FNG.

2) Encoders: The encoders on two DC motors is capable of measuring the rotation speed of wheels.

3) MPU-6500 (Accelerometer and Gyroscope): The MPU- 6500 is an inertial measurement unit (IMU) equipped with an accelerometer and a gyroscope. It can acquire x, y and z-axis acceleration (a_x, a_y, a_z) and angular velocity $(\omega_x, \omega_y, \omega_z)$ as signed 16-bit integers. Then, the tilt angle of the robot ψ, its angular velocity $\dot{\psi}$ and its yaw angular velocity $\dot{\phi}$ is expressed as follows:

$$\psi = \tan^{-1}\left(\frac{a_x}{a_z}\right);\ \dot{\psi} = \omega_y; \dot{\phi} = \omega_z \quad (1)$$

4) Battery and A/D Converter: This robot has a 12 V lithium battery package. Since a certain level of the voltage is required for proper motor control of the robot, a 12-bit A/D Converter (LTC2308) is used to obtain the whole system voltage.

5) IR Receiver: The IR receiver is used to receive and process signals which are sent from the IR remote controller. This allows to give the commands to the robot to run, rotate, and stop.

6) Ultrasonic Sensor: The ultrasonic sensor module (HC- SR04) is used to detect the distance of the obstacle in front of the robot. For obstacle avoidance, 3 ultrasonic sensor modules were installed in front of the robot and at 45 degrees to the left and right.

7) UART: The UART is a type of serial communication circuit. It is used to transmit and receive data between the PC and the robot.

III. Control System

This section describes how to control the robot.

A. PID Controller

The PID controller is the most classical and common method. We consider to design 3 PID controllers to control balance, speed, and turn of the robot [15]. These controllers can output PWM values from −100 to 100. Fig. 3 shows the block diagram of 3 PID controllers. The PWM values for left and right motors $(\mathrm{PWM_{left}}, \mathrm{PWM_{right}})$ are calculated as follows:

$$\mathrm{PWM_{left}} = -\mathrm{PWM_{balance}} - \mathrm{PWM_{speed}} + \mathrm{PWM_{turn}} \quad (2)$$
$$\mathrm{PWM_{right}} = -\mathrm{PWM_{balance}} - \mathrm{PWM_{speed}} - \mathrm{PWM_{turn}} \quad (3)$$

1) Balance Controller (PD): The balance controller is expressed as follows:

$$\mathrm{PWM_{balance}} = K_p\psi + K_d\omega_y \quad (4)$$

where, ψ is the tilt angle and ω_y is the angular velocity of y-axis component.

2) Speed Controller (PI): The speed controller is expressed as follows:

$$\mathrm{PWM_{speed}} = K_pE_t + K_i\left(\left(\sum E_t\right) + v\right) \quad (5)$$
$$E_t = 0.8E_{t-1} + 0.2(C_{\mathrm{right}} - C_{\mathrm{left}}) \quad (6)$$

where, v is the target speed and $C_{\mathrm{right}}, C_{\mathrm{left}}$ are the encoder values at the right and left motor. (6) means first-order low pass filter. Note that it is necessary to implement a saturation process since $\sum E_t$ can diverge in practice.

3) Turn Controller (PD): The turn controller is expressed as follows:

$$\mathrm{PWM_{turn}} = K_p\left(C_{\mathrm{left}} + C_{\mathrm{right}} + u\right) - K_d\omega_z \tag{7}$$

where, u is the target turn speed and ω_z is the angular velocity of z-axis component.

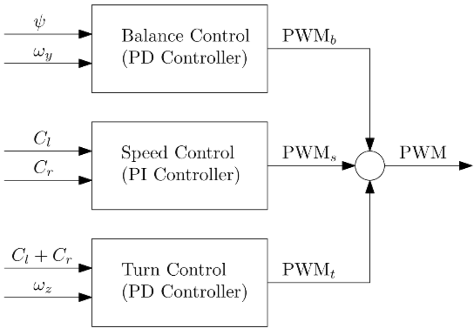

Figure 3. The block diagram of 3 PID controllers.

B. Kalman Filter

The sensor values obtained from the MPU-6500 contain a lot of noise, and it is not possible to control the system using those values. Therefore, we need to correct the sensor values. Following equations are a definition of discrete-time Kalman Filter.

$$\bar{x}_k = A\hat{x}_{k-1} + Bu_k \tag{8}$$

$$\bar{P}_k = AP_{k-1}A^{\mathsf{T}} + Q \tag{9}$$

$$K_k = \bar{P}_k H^{\mathsf{T}}\left(H\bar{P}_k H^{\mathsf{T}} + R\right)^{-1} \tag{10}$$

$$\hat{x}_k = \bar{x}_k + K_k(z_k - H\bar{x}_k) \tag{11}$$

$$P_k = (I - K_k H)\bar{P}_k \tag{12}$$

where,
$\bar{x}_k$: A priori state estimate; $\hat{x}_k$: A posteriori state estimate;
$\bar{P}_k$: A priori covariance matrix;
P_k: A posteriori covariance matrix;
Q , R: Covariance matrices; K_k: Kalman gain;
z_k: Observed value; I: Identity matrix; $k = 1, 2, 3, \ldots$.

In particular, gyroscope has a certain amount of error called as bias. We consider a state-space model with the tilt angle ψ and the bias of the gyroscope y-component $\omega_{y,\mathrm{bias}}$ and apply Kalman filter [16].

$$\bar{x}_k = \begin{bmatrix} 1 & -t_s \\ 0 & 1 \end{bmatrix} \hat{x}_{k-1} + \begin{bmatrix} t_s \\ 0 \end{bmatrix} \omega_y^{(k)} \tag{13}$$

$$\bar{P}_k = P_{k-1} - \begin{bmatrix} p_{01}^{(k-1)} + p_{10}^{(k-1)} & p_{11}^{(k-1)} \\ p_{11}^{(k-1)} & 0 \end{bmatrix} t_s + Q \tag{14}$$

$$K_k = \frac{1}{\bar{p}_{00}^{(k)} + R} \begin{bmatrix} \bar{p}_{00}^{(k)} \\ \bar{p}_{10}^{(k)} \end{bmatrix} \tag{15}$$

$$\hat{x}_k = \bar{x}_k + K_k\left(\psi_{\mathrm{obs}}^{(k)} - \bar{\psi}_k\right) \tag{16}$$

$$P_k = \bar{P}_k - K_k\begin{bmatrix} \bar{p}_{00}^{(k)} & \bar{p}_{01}^{(k)} \end{bmatrix} \tag{17}$$

where,

$$\hat{x}_k = \begin{bmatrix} \hat{\psi}_k \\ \hat{\omega}_{y,\mathrm{bias}}^{(k)} \end{bmatrix}; \bar{x}_k = \begin{bmatrix} \bar{\psi}_k \\ \bar{\omega}_{y,\mathrm{bias}}^{(k)} \end{bmatrix}; P_k = \begin{bmatrix} p_{00}^{(k)} & p_{01}^{(k)} \\ p_{10}^{(k)} & p_{11}^{(k)} \end{bmatrix}. \tag{18}$$

ψ_{obs} is derived by Eq 1 and used for correcting values. t_s is the sampling time. The remaining parameters are set as follows: $t_s = 10$ ms; $Q_k = \mathrm{diag}(0.00003, 0.00001)$; $R_k = 0.5$; $\hat{x}_0 = [0, 0]^{\mathsf{T}}$; $P_0 = \mathrm{diag}(1, 1)$.

IV. Obstacle Avoidance System

This section describes the method of obstacle avoidance of the robot based on Mamdani's fuzzy inference system [17].

A. Fuzzy Logic Controller

The fuzzy logic controller (FLC) receives the distance data between the robot and obstacles obtained from the left, front, and right ultrasonic sensors (d_l, d_d, d_r), then outputs the azimuth angle of the robot ϕ. Fig. 4 shows the block diagram of the FLC.

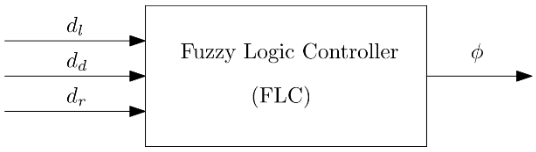

Figure 4. Fuzzy Logic Controller (FLC).

B. Fuzzy Membership Functions

The range of inputs (d_l, d_d, d_r) is limited from 0 m to 1 m and divided into linguistic variables {"Near", "Far"}. The range of output (ϕ) is limited from −90° to 90° and divided into linguistic variables {"Left", "Front", "Right"}. Fig. 5, 6 shows the fuzzy membership functions of inputs and output.

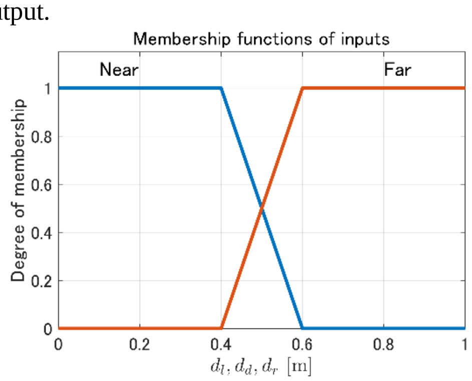

Figure 5. Fuzzy membership functions of inputs $(\boldsymbol{d_l}, \boldsymbol{d_d}, \boldsymbol{d_r})$.

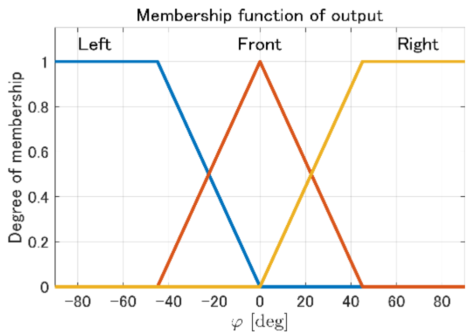

Figure 6. Fuzzy membership functions of output (ϕ).

C. Fuzzy Rules

Table I shows the fuzzy rules of the controller. For example, the rule No.1 means that if the inputs d_l is "Near", d_d is "Near", and d_r is "Far", then the output ϕ is "Right".

TABLE I. FUZZY RULES

Rule No.	d_l	d_d	d_r	ϕ
1	Near	Near	Far	Right
2	Near	Far	Far	Right
3	Far	Near	Near	Left
4	Far	Far	Near	Left
5	Far	Far	Far	Front
6	Far	Near	Far	Right

D. Defuzzification

There are many defuzzification methods: center of gravity (CoG), first of maximum (FoM), mean of maximum (MoM), last of maximum (LoM) and so on. In this paper, we adopt MoM method expressed as following equation.

$$\mathrm{MoM} = \frac{\mathrm{FoM} + \mathrm{LoM}}{2} \tag{19}$$

E. Simulation Results

Fig. 7 shows the simulation results of the fuzzy controller by MATLAB Fuzzy Logic Toolbox.

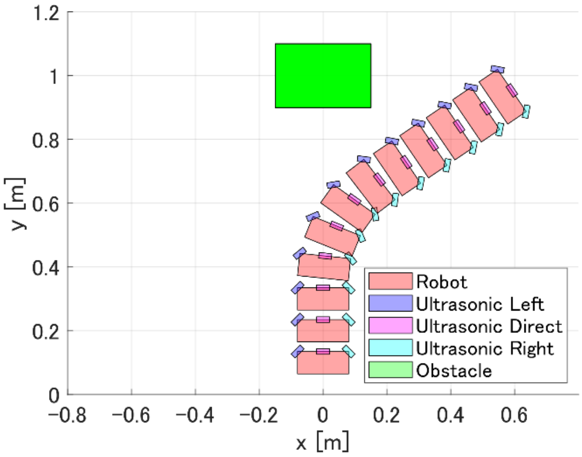

Figure 7. The simulation results of the controller.

V. FIXED-POINT ARITHMETIC

Fixed-point arithmetic is used for calculating Kalman filter, PID controller and Fuzzy logic controller.

A. Custom Instructions

We implemented custom instructions of Addition, subtraction, multiplication, minimum and maximum for fixed-point arithmetic. Table II shows encodings of the instructions. Since these instructions are binary operation, encodings are expressed as R-Format.

TABLE II. R-FORMAT ENCODING

	funct7	rs2	rs1	funct3	rd	opcode
Addition	0000010	rs2	rs1	000	rd	0001011
Subtraction	0000010	rs2	rs1	001	rd	0001011
Multiplication	0000010	rs2	rs1	010	rd	0001011
Minimum	0000010	rs2	rs1	100	rd	0001011
Maximum	0000010	rs2	rs1	101	rd	0001011

1) *Addition and Subtraction:* The calculation diagrams of addition ($C = A + B$) and subsection ($C = A - B$) are shown in Fig. 8 (a). $clip_{add}$ and $clip_{sub}$ are clipping (saturation) functions which can be expressed below. Note that these numerical values are 2's complement.

$$clip_{\mathrm{add}} = \begin{cases} \mathrm{0x80000000} & (a = 0, of_{\mathrm{add}} = 1) \\ res & (of_{\mathrm{add}} = 0) \\ \mathrm{0x7FFFFFFF} & (a = 1, of_{\mathrm{add}} = 1) \end{cases} \tag{20}$$

$$clip_{\mathrm{sub}} = \begin{cases} \mathrm{0x80000000} & (a = 0, of_{\mathrm{sub}} = 1) \\ res & (of_{\mathrm{sub}} = 0) \\ \mathrm{0x7FFFFFFF} & (a = 1, of_{\mathrm{sub}} = 1) \end{cases} \tag{21}$$

where, of_{add} and of_{sub} are the combinational logic for overflow detection:

$$of_{\mathrm{add}} = \neg(a \oplus b) \wedge (a \oplus s_+) \tag{22}$$
$$of_{\mathrm{sub}} = (a \oplus b) \wedge (a \oplus s_-) \tag{23}$$

where, a, b and $s_\pm$ are the most significant bit (MSB) of $A, B, A \pm B$. Symbols of $\neg$, $\wedge$ and $\oplus$ are the operator of NOT, AND, and exclusive OR, respectively.

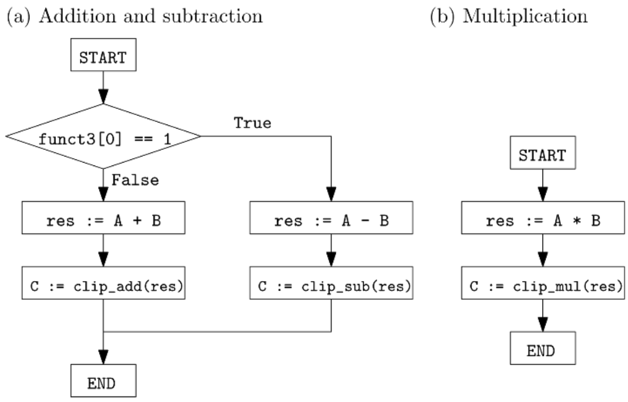

Figure 8. (a) The calculation diagrams of addition and subtraction, (b) The calculation diagrams of multiplication.

2) *Multiplication:* The calculation diagram of multiplication is shown in Fig. 8 (b). $clip_{mul}$ is a clipping function which can be expressed below. Note that res is 64-bit data and the output result C is a 32-bit fixed-point.

$$clip_{\mathrm{mul}} = \begin{cases} \mathrm{0x80000000} & (a \oplus b = 1, res[63{:}48] \neq 0) \\ res[47{:}16] + res[15] & (a \oplus b = 0) \\ \mathrm{0x7FFFFFFF} & (a \oplus b = 0, res[63{:}48] \neq 0) \end{cases} \tag{24}$$

3) *Minimum and Maximum:* The maximum and minimum instructions are intended to get rid of conditional branches, which are expressed as follows:

$$min(A, B) = \begin{cases} B & (A > B) \\ A & (A < B) \end{cases} \tag{25}$$

$$max(A, B) = \begin{cases} A & (A > B) \\ B & (A < B) \end{cases} \tag{26}$$

This is effective in pipelined processors.

B. *Divison*

In this work, division was calculated by libfixmath [18], which is a 32-bit fixed-point arithmetic library for C language.

C. *Trigonometric Arithmetic (Arctan)*

It is common to use a standard library such as “math.h” to calculate trigonometric functions in C language. However, it is implemented using a floating-point arithmetic. We implement a fixed-point version of arctan function based on COordinate Rotation DIgital Computer (CORDIC) [19] since only arctan is used for trigonometric calculations in this work.

VI. SOFTWARE IMPLEMENTATION

Fig. 9 shows the flow chart of the software program. This main loop program is implemented by a timer interrupt at 10 ms cycles. The control program is written in C language and compiled by GNU GCC cross-compiler for RISC-V ISA [20]. In this work, the compiler optimization option (-O3) was used. In addition to that, we used the “insn” pseudo-instruction via inline assembly when calling custom instructions in C language.

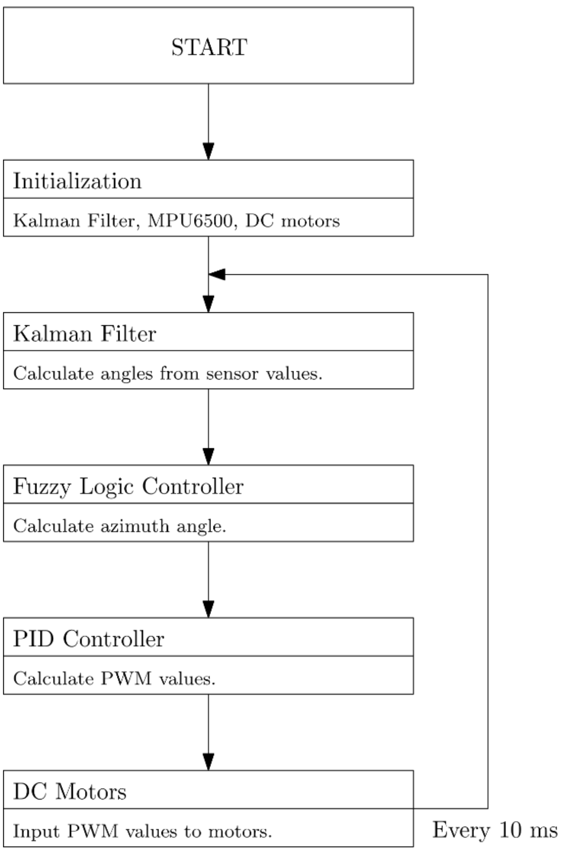

Figure 9. Flow chart of the software program.

VII. EXPERIMENT AND RESULTS

A. *Systhesis Results*

Table III shows the synthesis results of the FPGA board (DE10-NANO, Cyclone V 5CSEBA6U2317) by Quartus Prime.

TABLE III. SYNTHESIS RESULTS

Logic utilization (in ALMs)	2,521
Total registers	4,672
Total block memory bits	661,760
Total DSP Blocks	3
Maximum frequency [MHz]	63.22

B. *Calculation Speed*

Table IV shows the comparison of calculation speed of addition, subtraction and multiplication between libfixmath and our custom instructions. In this work, all types of our custom instructions are faster than software implementation by libfixmath. All custom instructions are same calculation time since their calculation part are processed by a clock cycle. Table V shows the comparison of the number of the instructions of PID Controller, Kalman Filter and Fuzzy Controller. Table VI shows the comparison of the calculation speed of PID Controller, Kalman Filter and Fuzzy Controller between libfixmath and our custom instructions.

TABLE IV. CALCULATION SPEED COMPARISON: ADDITION, SUBTRACTION, MULTIPLICATION, MINIMUM, MAXIMUM

	libfixmath [µs]	Custom Instructions [µs]
Addition	1.37	0.83
Subtraction	1.41	0.83
Multiplication	2.96	0.83
Minimum	0.95	0.83
Maximum	0.95	0.83

TABLE V. NUMBER OF FIXED-POINT INSTRUCTIONS

	Add	Sub	Mul	Min	Max
PID Controller	5	4	9	0	0
Kalman Filter	8	8	11	0	0
FLC	1	18	19	141	164

TABLE VI. CALCULATION SPEED COMPARISON: PID CONTROLLER, KALMAN FILTER AND FLC

	libfixmath [µs]	Custom Instructions [µs]
PID Controller	32	13
Kalman Filter	58	31
FLC	1,815	1,028

C. *Obstacle Avoidance*

We have an experiment for an obstacle avoidance of the robot. In this experiment, a box was placed as an obstacle in front of the robot. Fig. 10 shows the results of the obstacle avoidance experiment. Fig. 10 (a) and (b) confirm that the robot turns right according to the fuzzy rule No.6 (Table I) since d_l is “Far”, d_d is “Near” and d_r is “Far”. Fig. 10 (c) and (d) demonstrate that the robot keeps the direction according to the fuzzy rule No. 2 and No. 4.

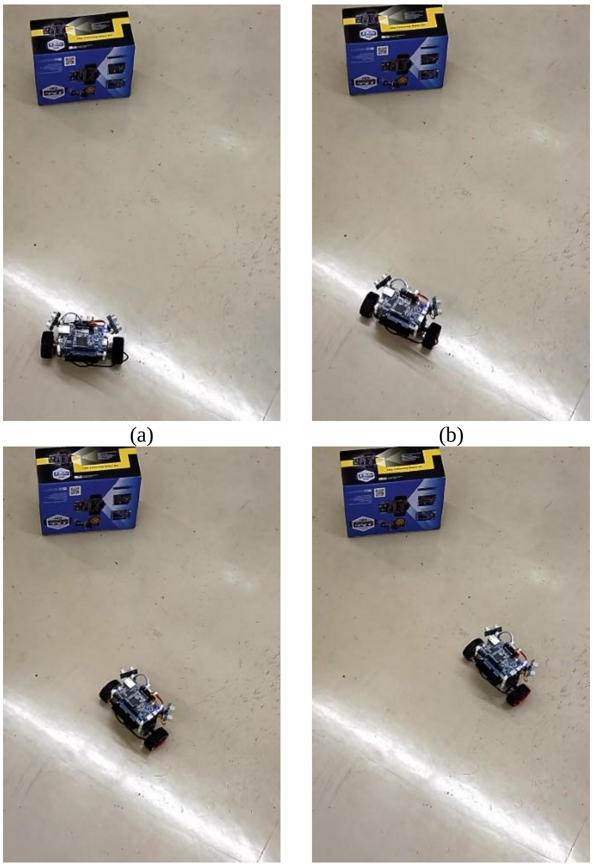
(a) (b) (c) (d)

Figure 10. Experiment results of obstacle avoidance.

VIII. Conclusion

In this paper, we introduced the design of a controller with obstacle avoidance function using an accelerometer, a gyroscope, motor encoders, and ultrasonic sensors. The control and obstacle avoidance programs were executed on VexRiscv, a 32-bit RISC-V soft microprocessor with custom instructions of 32-bit fixed-point operations. As a result, we have managed to construct control and obstacle avoidance system without FPU.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

T. T. Hoang and C. K. Pham supervised the research and revised the manuscript. R. Tsutada carried out the experiment and wrote the paper.

References

[1] K. Liu, M. Bai, and Y. Ni, "Two-wheel self-balanced car based on Kalman filtering and PID algorithm," in *Proc. 2011 IEEE 18th International Conference on Industrial Engineering and Engineering Management*, Changchun, pp. 281–285, 2011.

[2] C. Xu, M. Li, and F. Pan, "The system design and LQR control of a two-wheels self-balancing mobile robot," in *Proc. 2011 International Conference on Electrical and Control Engineering*, Yichang, pp. 2786–2789, 2011.

[3] M. Engin, "Embedded LQR controller design for self-balancing robot," in *Proc. 2018 7th Mediterranean Conference on Embedded Computing (MECO)*, Budva, pp. 1–4, 2018.

[4] C. Iwendi, M. A. Alqarni, J. H. Anajemba, A. S. Alfakeeh, Z. Zhang, and A. K. Bashir, "Robust navigational control of a two-wheeled self-balancing robot in a sensed environment," *IEEE Access*, vol. 7, pp. 82337–82348, 2019.

[5] H. Juang and K. Lurrr, "Design and control of a two-wheel self-balancing robot using the arduino microcontroller board," in *Proc. 2013 10th IEEE International Conference on Control and Automation (ICCA)*, Hangzhou, pp. 634–639, 2013.

[6] B. Zeng, J. Zhang, L. Chen, and Y. Wang, "Self-balancing car based on ARDUINO UNO R3," in *Proc. 2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, Chongqing, pp. 1939–1943, 2018.

[7] X. Ruan and W. Li, "Ultrasonic sensor based two-wheeled self-balancing robot obstacle avoidance control system," in *Proc. 2014 IEEE International Conference on Mechatronics and Automation*, Tianjin, China, pp. 896– 900, 2014.

[8] A. Ruospo, R. Cantoro, E. Sanchez, P. D. Schiavone, A. Garofalo, and L. Benini, "On-line testing for autonomous systems driven by RISC-V processor design verification," in *Proc. 2019 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, Noordwijk, Netherlands, pp. 1–6, 2019.

[9] J. Lee, H. Chen, J. Young, and H. Kim, "RISC-V FPGA Platform Toward ROS-Based Robotics Application," in *Proc. 2020 30th International Conference on Field-Programmable Logic and Applications (FPL)*, Gothenburg, Sweden, pp. 370–370, 2020.

[10] Terasic, Robotic Kits - Self-Balancing Robot. [Online]. Available: https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=&No=1096.

[11] Intel Nios II processor. [Online]. Available: https://www.intel.co.jp/content/www/jp/ja/products/programmable/processor/nios-ii.html.

[12] VexRiscv. [Online]. Available: https://github.com/SpinalHDL/VexRiscv.

[13] SpinalHDL, [Online]. Available: https://github.com/SpinalHDL/SpinalHDL.

[14] The Scala Programming Language, [Online]. Available: https://www.scala-lang.org/.

[15] T. Liu, X. Wang, H. Zhou, X. Che, H. Liu, and Q. Wang, "Design and control of a two-wheeled self-balancing robot made in 3D printing," in *Proc. 2018 Chinese Automation Congress (CAC)*, Xi'an, China, pp. 1211–1216, 2018.

[16] J. Juan Rincón Pasaye, J. Alberto Bonales Valencia, and F. Jiménez Pérez, "Tilt measurement based on an Accelerometer, a Gyro and a Kalman Filter to control a self-balancing vehicle," in *Proc. 2013 IEEE International Autumn Meeting on Power Electronics and Computing (ROPEC)*, Mexico City, pp. 1–5, 2013.

[17] Mamdani, "Application of fuzzy logic to approximate reasoning using linguistic synthesis," *IEEE Transactions on Computers*, vol. C-26, no. 12, pp. 1182–1191, Dec. 1977.

[18] Libfixmath. [Online]. Available: https://code.google.com/archive/p/libfixmath/

[19] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Transactions on Electronic Computers*, vol. EC-8, no. 3, pp. 330– 334, Sept. 1959.

[20] RISC-V GNU Compiler Toolchain, [Online]. Available: https://github.com/riscv/riscv-gnu-toolchain

Ryuichi Tsutada is received the B.E. degree in information and communication engineering from the University of Electro-Communications, Tokyo, Japan in 2020. He is currently a master student in information and network engineering with the Department of Information and Network Engineering, the University of Electro-Communications, Tokyo.

Trong-Thuc Hoang received the B.Sc. degree in electronics and telecommunications and the M.S. degree in microelectronics from the University of Science, Vietnam National University, Ho Chi Minh City, Vietnam, in 2012 and 2017, respectively. He is currently pursuing the Ph.D. degree in information and network engineering with the University of Electro-Communications, Tokyo, Japan. He is also a Research Assistant with the National Institute of Advanced Industrial Science and Technology, Tokyo.

Cong-Kha Pham received the B.S., M.S., and Ph.D. degrees in electronics engineering from Sophia University, Tokyo, Japan in 1989, 1990 and 1992, respectively. He is currently a Professor with the Department of Information and Network Engineering, University of Electro-Communications, Tokyo. His research interests include the design of analog and digital systems using integrated circuits.