Signal and Image Processing Letters
Vol. 4., No. 2, 2022, pp. 16-27
ISSN 2714-6677
https://simple.ascee.org/

Sub controller design on KRSBI humanoid R-SCUAD robot sub controller design on KRSBI humanoid robot R-SCUAD

Bahrul Mizan [a,1], Nuryono Satya Widodo [a,2*]

[a] Department of Electrical Engineering, Universitas Ahmad Dahlan, Yogyakarta, Indonesia
[2] nuryono.sw@ee.uad.ac.id
* corresponding author

ARTICLE INFO

ABSTRACT

The purpose of this research is to design OpenCM9.04 controllers such as Arbotix (Pro) with MPU-9250 sensor as robot balance, as well as controlling the movement of DYNAMIXEL servo angles based on camera input on the robot, the Design of the OpenCM9.04 controller board on the robot consists of 2 main components namely OpenCM9.04 which works as a mini system and OpenCM 485 Expansion Board that works as a conference to the serial that provides interface to buttons and LEDs as well as a power supply circuit , where OpenCM9.04 as the main controller then sends data to OpenCM 485 which will process the MPU-9250 sensor as well as the DYNAMIXEL servo on the robot. The hardware system design consists of an MPU-9250 sensor to maintain balance in the robot so that the robot does not fall when walking or running and servo DYNAMIXEL to move the corners on the robot. The results obtained by the OpenCM9.04 controller have been successfully developed and tested on robots in the KRSBI-H racetrack and the robot has been able to maximize in the game well without any constraints on the microcontroller used.

Keywords
KRSBI-H
MPU-9250
OpenCM9.04
OpenCM485

This is an open access article under the CC–BY-SA license.

1. Introduction

Kontes Robot Indonesia (KRI) is one of the design and engineering competitions in the field of robotics organized annually by the Ministry of Research, Technology and Higher Education, Directorate General of Learning and Student Affairs. The Indonesian Robot Contest consists of 5 competition divisions, one of which is the Kontes Robot Sepak Bola Indonesia (KRSBI) division.

R-SCUAD (Robot Soccer UAD) is one of the humanoid soccer robot Research developed by lecturers and students of Ahmad Dahlan University. Currently, the R-SCUAD robot can mimic several abilities like humans playing soccer such as looking for the ball, chasing the ball, kicking the ball, getting up from falling and several other abilities. The problem faced in designing controllers is designing OpenCM9.04 controllers such as Arbotix (Pro) controllers with the ability to control the balance of the robot, as well as control the angular movements of DYNAMIXEL servos.

The control used to make the controller, added the MPU-9250 sensor as a robot balance controller and accessed the DYNAMIXEL servo to move the servo angles in order to move the robot. Minisystems in the R-SCUAD robot include Odroid mini system, OpenCM9.04 controller, OpenCM485 expansion, DYNAMIXEL servo, webcam camera, and battery. The R-SCUAD robot controller made must be able to integrate well with the controller used in order to obtain physical movements that match the original robot. OpenCM9.04 controller must be able to simulate some of the same movements as the R-SCUAD robot. Humanoid robots, especially ball player robots are built from a series of actuators in the form of servo motors and connectors between servos that connect them to each other [1].

2. Research Method

In this research, a Sub Controller Design system will be built on KRSBI Humanoid R-scuad Robot. This controller uses the main sensor, the MPU-9250 sensor to help maintain the balance of the robot so that it does not fall. This robot uses an OpenCM9.04 microcontroller as a Microprocessor Processing Unit. The OpenCM9.04 microcontroller will command the MPU-9250 sensor which functions to provide gyro and Accelerometer readings with the x y z axis on the balance of the robot so that it can be used as an input value that will be executed by OpenCM9.04 to maintain the balance of the robot by moving the DYNAMIXEL servo angles. The microcontroller designed can only receive data and the processed data from the microcontroller is used to move the actuator [2]. Here is the research method.

2.1. OpenCM9.04 Microcontroller

OpenCM9.04 is an Open Source, robotic microcontroller module including hardware and software. The controller also comes with a needle compatible with Robotis sensors and DYNAMIXEL servos. OpenCM9.04 has a similar appearance to the Arduino IDE. In addition, OpenCM9.04 is also equipped with a library to access Robotis DYNAMIXEL servos and sensors. In OpenCM9.04 there are 3 types, namely Type A, Type B and TypeC. The difference between these types is the availability of connectors. What you see in Fig. 1 is an image of the OpenCM9.04 board [3].

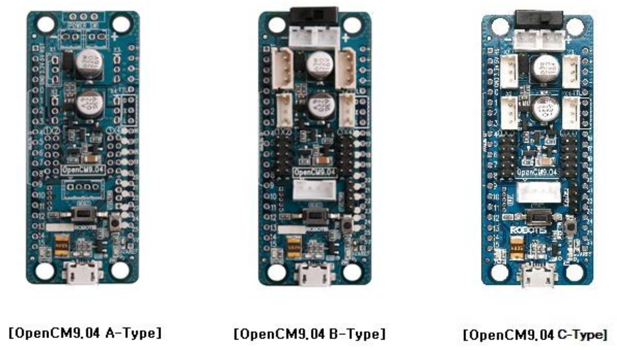


Fig. 1. Controller board

2.2. OpenCM 485

OpenCM is an open-source hardware and software controller. In hardware, an ARM Cortex-M3 microcontroller supports this controller. The controller also comes with a compatible needle model. OpenCM9.04, which can be used for C/C++ programming, can easily access the code via USB. The OpenCM 485 is an excellent choice for small and medium-sized robots, and can also be used as a DYNAMIXEL for intelligent controllers on large robots. Microcontrollers based on the ARM (Advanced RISC Machine) architecture, which is a 32-bit processor, are more reliable in data transmission and save power. STM32F103 is a series of STM32 ARM CortexM3 microcontrollers that belong to the mainstream type. STM32F103 has a 32-bit ARM Cortex-M3 processor performance with a clock frequency of 72 MHz [4].

2.3. MPU-9250

The MPU-9250 is a multi-chip module (MCM) consisting of two printed products integrated in a QFN package. The first mounts a 3-axis gyroscope and a 3-axis Accelerometer, and the second mounts a 3-axis magnetometer AK8963 from Asahi Kasei Microdevices Corporation. Therefore, the MPU-9250 is a 9-axis motion tracking device that combines a 3-axis gyroscope, 3-axis Accelerometer, 3-axis magnetometer, and Motion Digital Processor™ (DMP) together, and is provided in a small package of 3x3x1mm. compatible. Upgrade from MPU6515. Through a dedicated I2C sensor bus, the MPU-9250 can immediately provide a complete 9-axis MotionFusion™ output. MPU-9250 MotionTracking device with 9-axis integration, on-chip MotionFusion™ and runtime calibration firmware, with serial I2C and SPI interfaces, working range of 2.4V to 3.6V VDD and independent digital IO power supply, VDDIO from 1.71V to VDD [5].

2.4. UART and USART protocols

UART or Universal Asynchronous Receiver/Transmitter is a circuit that mediates and functions as a converter from serial to parallel on receiving data from peripheral devices and converting from parallel to serial on receiving data from the CPU. Can be seen in Fig. 2 is a picture of UART communication [6].

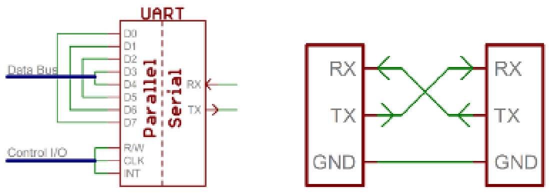


Fig. 2. Simple illustration of UART (Left) and UART wiring (Right)

Universal Synchronous Asynchronous Receiver Transmitter (USART) is a serial communication protocol. It usually uses two ports, one for transmitting data (TX port) and the other for receiving data (RX port). If necessary, it can use an additional port as a clock for synchronous communication. USART can be enabled through the USART register in the microcontroller. USART has three modes, asynchronous normal mode, asynchronous double speed and synchronous mode [7].

2.5. Full Duplex dan Half Duplex

Data transmission over transmission lines can be divided into full duplex or half duplex. When half-duplex transmission is used, one of the two stations on the end-to-end connection can only transmit in real time. The model also involves bidirectional transmission, where two stations must transmit alternately. This can be compared to a one-lane, two-bridge bridge. This form of transmission is typically used for terminal-to-computer interaction. When the user enters and sends data, the host computer will stop sending data to the terminal, as this might appear on the terminal screen and cause confusion. What you see in Fig. 3 is an image of full-duplex communication and half-duplex communication [8].

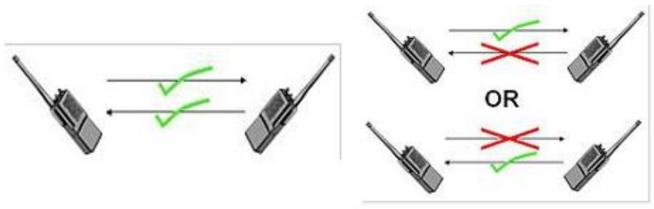


Fig. 3. Full-Duplex Communication (Left) and Half-Duplex Communication (Right)

2.6. Motor Servo (Actuator) DYNAMIXEL

Actuator is a mechanical device to drive or control a mechanism or system. The actuators used are DYNAMIXEL servo motors with the MX-64 and MX-28 types shown in Fig. 4. This type of actuator is used as joints on the robot that will support the movement of the robot and are interconnected to the OpenCM [9].

Fig. 4. DYNAMIXEL servo

2.7. SPI protocol

SPI uses two pins for data transmission, namely SDI (Din) and SDO (Dout). The SPI bus has SCLK (shift clock), which is used as a pulse provider line for data synchronization. Finally, SPI has one or more CE (chip enable) to select slaves to communicate with the master. On some devices, these four pins (SDI, SDO, SCLK and CE) are also called MOSI (master output and slave input), MISO (master input and slave output), SCK and SS (slave selection). The SPI architecture is shown in Fig. 5 [10].

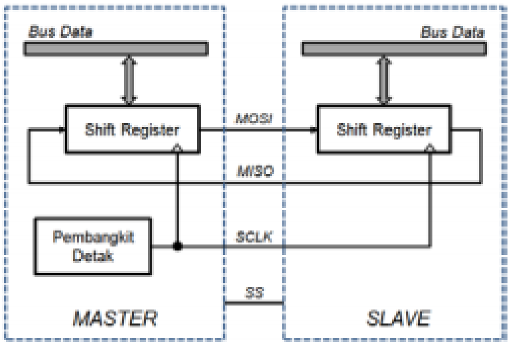


Fig. 5. SPI architecture

2.8. Measurement of Odroid System Mini Circuit and OpenCM9.04 Controller

Measurements of the Odroid mini system and OpenCM controller circuit are carried out by providing a voltage source from a 12 V battery using a 11.4 V/2600 mAh Lipo battery, to 5V for the Odroid mini system and 12 V for OpenCM. To get 5V voltage from the battery, a voltage regulator or UBEC 30A is used, this 5V voltage will be supplied to the odroid and compass sensor. While the 12V voltage will be directly supplied to OpenCM and DYNAMIXEL servo which requires 12V voltage. Can be seen in Fig. 6 is a picture of the controller board.

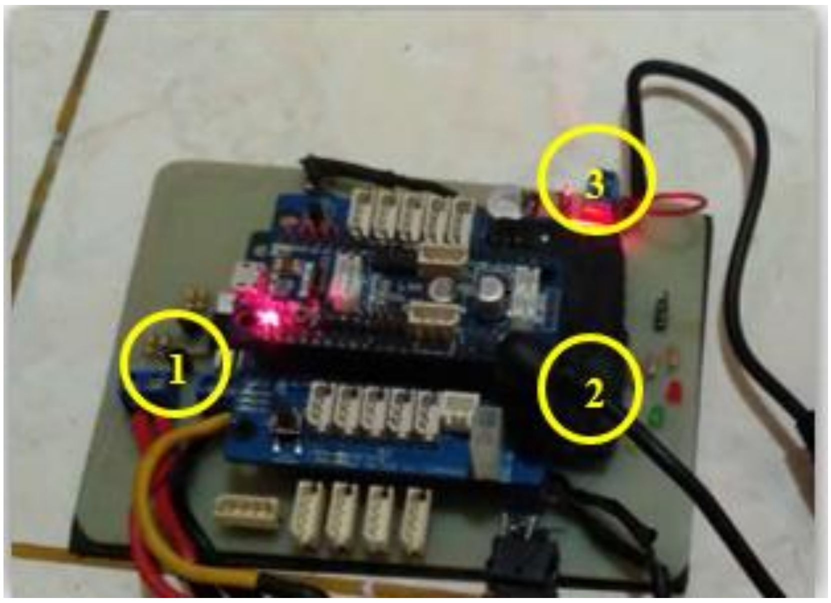


Fig. 6. Controller board

2.9. Robot System Block Diagram

In general, the system block diagram of the robot consists of several parts, namely the main controller, controller, actuator, camera, Wi-Fi module, and several other parts with the system block diagram design as shown in Fig. 7.

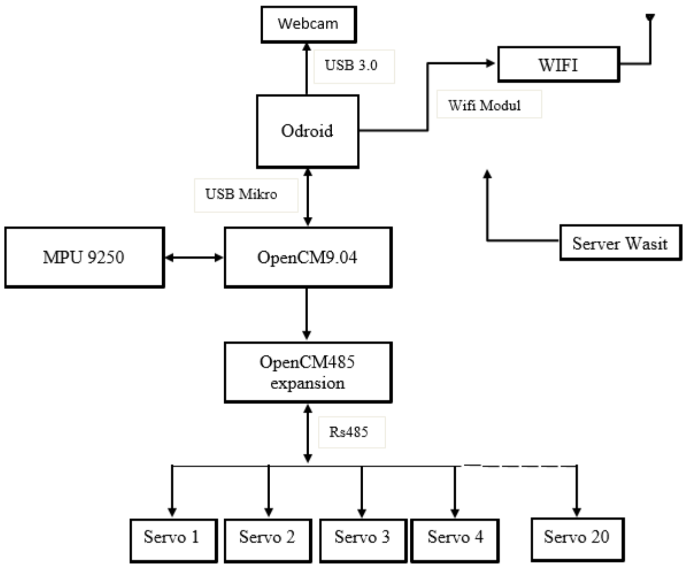


Fig. 7. Block diagram

The system design of this robot consists of a mini pc, Odroid, which functions as the main controller that processes the image captured through the webcam/camera, communication between Odroid and the camera using USB 3.0, the compass value is contained in the MPU-6050 sensor using I2C communication, the gyroscope and Accelerometermeter values for the balance of the robot are contained in the MPU-9250, communication between the MPU 9250 and OpenCM9.04 using SPI communication. Then, Odroid determines the movement that the robot should make by sending it to OpenCM9.04 which functions as a controller and OpenCM9.04 sends the received data to the actuator, namely the DYNAMIXEL servo. This MPU-9250 sensor will provide feedback to the main controller in the form of a value that will be used to balance the robot while walking or determine the condition of the robot falling forward or falling backward. Wi-Fi is used to process commands from the referee and processed by the main controller to determine the movement of the robot to be used.

2.10. Controller System Design

In the previous section, it has been described about the schematic design of the controller, so that the design can work, the controller system design is needed. In designing the controller system, there are several configurations in the main program used by the microcontroller, namely:

1. I/O configuration
2. Interrupt configuration
3. USART configuration
4. SPI configuration
5. Process Initialization
6. Gyroscope configuration
7. Accelerometer configuration
8. ADC configuration
9. Timer2 configuration
10. USB configuration of data packet process

To make the programming, a system flowchart of the microcontroller programming will first be made to explain the flow of the program that shows the sequence and process relationship of the program. Controller Flowchart can be seen in Fig. 8.

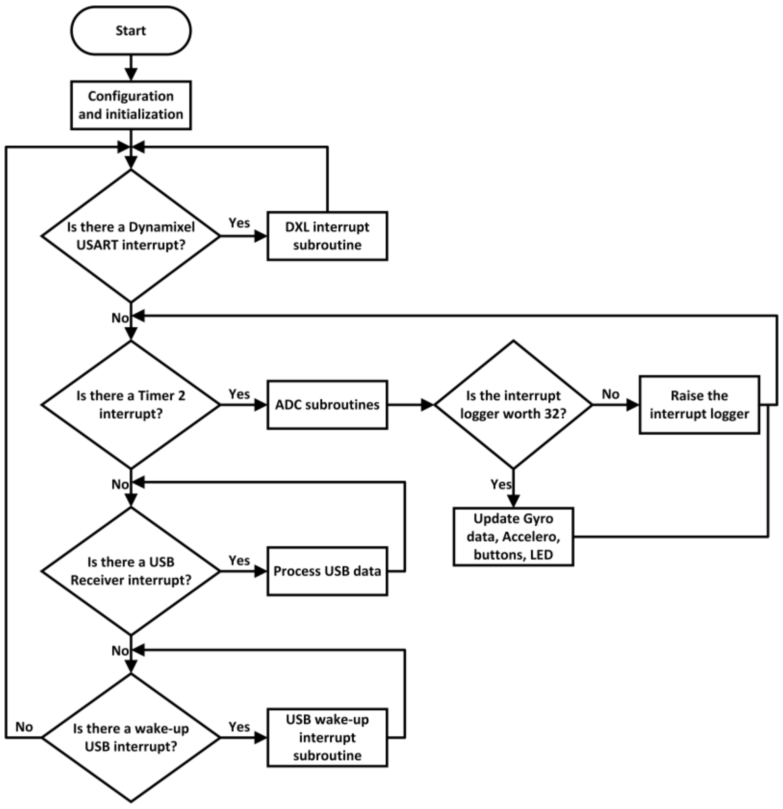

Start
Configuration and initialization
Is there a Dynamixel USART interrupt?
Yes
DXL interrupt subroutine
No
Is there a Timer 2 interrupt?
Yes
ADC subroutines
Is the interrupt logger worth 32?
No
Raise the interrupt logger
Yes
Update Gyro data, Accelero, buttons, LED
No
Is there a USB Receiver interrupt?
Yes
Process USB data
No
Is there a wake-up USB interrupt?
Yes
USB wake-up interrupt subroutine
No

Fig. 8. Controller Flowchart

The controller flowchart created is a microcontroller system process. In the controller system process that is running the program, if an interrupt is requested in one of the 4 interrupts above, the program will stop for a moment to serve the requested interrupt by running the program at the address indicated until the interrupt is complete. When it has finished running the interrupt, the microcontroller system process will resume the program that was stopped by the previous interrupt. Description of interrupts in the sub controller:

1. DYNAMIXEL USART interrupt: to send command data from Odroid via USB, then translated to DYNAMIXEL by the sub controller.
   (a) DXL interrupt subroutine: for translating serial data DYNAMIXEL protocol packets.
2. Interrupt Timer2: is a time base, setting to interrupt every 120 (microseconds)
   (a) ADC subroutine: for ADC communication line
   (b) 32 interrupt loggers: in the interrupt timer, if 32 interrupts have occurred, the MPU-9250 data will be accessed via SPI communication
3. USB receiver interrupt: if there is data from the Odroid to OpenCM9.04
4. USB wake-up interrupt: when connecting to Odroid for comport analysis

2.11. Sub Controller Tool Testing

The controller is used for communication between the controller and the DYNAMIXEL servo, the way it works is to send data to each other between the controller to the servo and from the servo to the controller, the results of this communication data will be seen through a logic analyzer that outputs digital wave results, where the digital wave will be compared with the communication results issued by the controller and DYNAMIXEL servo. In this test there are also 2 experiments, the first using USB2 DYNAMIXEL to send the data and the second through directly to the controller or more precisely through OpenCM9.04 to send the data. This tool consists of OpenCM9.04, USB2 DYNAMIXEL, Logic Analyzer, DYNAMIXEL servo and PC. Research on this controller can be seen in Fig. 9.

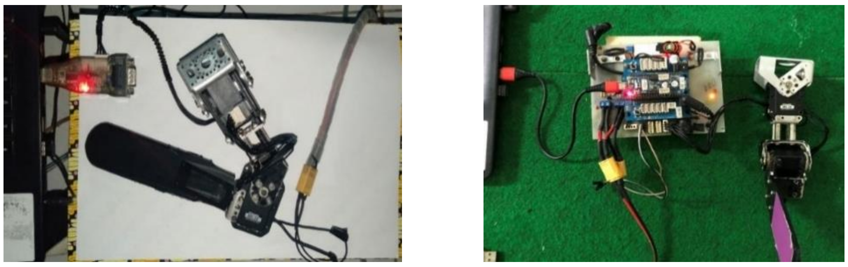


Fig. 9. Testing 1 USB2 DYNAMIXEL (Left) and Testing 2 OpenCM9.04 (Right)

3. Results and Discussion

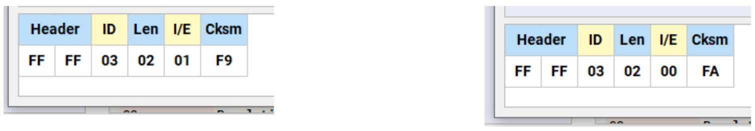
In this study, testing was carried out to ensure that the MPU-9250 sensor and DYNAMIXEL servo could take data properly. Where in this test the robot can ensure and can detect balance through the MPU-9250 sensor, so that the robot in a standing state can maintain its body balance. Next is testing the DYNAMIXEL servo, in this test ensuring that the DYNAMIXEL servo can be connected properly to the microcontroller. The test carried out on the controller is to see the communication data between the controller and the DYNAMIXEL servo, the goal is whether the data received and sent by OpenCM9.04 to the DYNAMIXEL servo is the same as that seen by the Logic Analyzer, because later the results issued by OpenCM9.04 will be compared with the results of the Logic Analyzer. While on USB2 DYNAMIXEL can only see communication data between the controller and DYNAMIXEL servo.

3.1. Data capture using USB DYNAMIXEL

On USB2 DYNAMIXEL there are 3 pins and 4 pins which both function for DYNAMIXEL servos, the pins that will be used on USB2 DYNAMIXEL for this test are 4 pins in the form of (D-), (D+), (+) and (-). This test is carried out by connecting the USB2 DYNAMIXEL to the PC after opening the DYNAMIXEL Wizard 2.0 software, for the 4 pins contained in the USB2 DYNAMIXEL are connected to the controller which has 4 pins, after which pin 4 contained in the controller is connected again to the DYNAMIXEL servo and then attach a 2600 mAh battery to turn on the controller found in Fig. 11 above. The test conducted in this study is to send data to the DYNAMIXEL servo that the data conveyed to the DYNAMIXEL servo is complete and as it should be. Tests carried out in sending data include several instructions, namely:

1. Ping Instruction

   Ping is an instruction that requests a status packet from a specific ID even if the status return level (16) is 0, so DYNAMIXEL returns status packets all the time for the ping instruction. The data sent by the ping instruction can be seen in Fig. 10.



Header		ID	Len	I/E	Cksm
FF	FF	03	02	01	F9

(a)

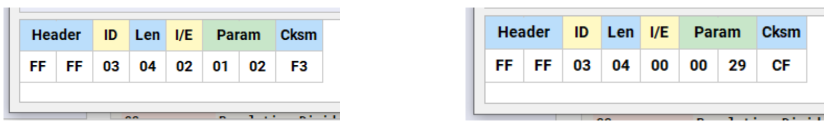
Header		ID	Len	I/E	Cksm
FF	FF	03	02	00	FA

(b)

Fig. 10. Data display (a) Tx communication, (b) Rx communication ping instructions

2. Read Instruction

   Read is an instruction to read the data of the starting address of the data and the length of the data to be read in the DYNAMIXEL control table. The data sent by the Read instruction can be seen in Fig. 11.



Header		ID	Len	I/E	Param		Cksm
FF	FF	03	04	02	01	02	F3

(a)

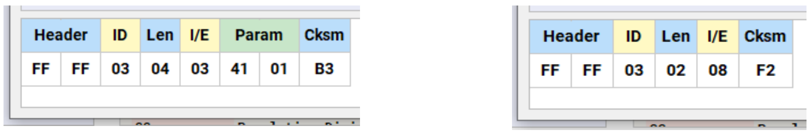
Header		ID	Len	I/E	Param		Cksm
FF	FF	03	04	00	00	29	CF

(b)

Fig. 11. Data display (a) Tx communication, (b) Rx communication read instruction

3. Write Instruction

   Write is an instruction to write data to the DYNAMIXEL control table, namely on Parameter 1, Parameter 2, Parameter 3, Parameter N+1. The data sent by the write instruction can be seen in Fig. 12.



(a)

Header		ID	Len	I/E	Param		Cksm
FF	FF	03	04	03	41	01	B3

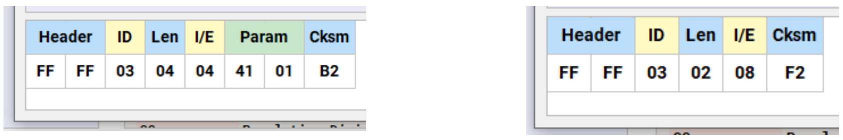
(b)

Header		ID	Len	I/E	Cksm
FF	FF	03	02	08	F2

Fig. 12. Data display (a) Tx communication, (b) Rx communication write instruction

4. Reg Write Instruction

   The Reg Write instruction is similar to the Write Instruction, but the Reg Write instruction has enhanced synchronization characteristics. The data sent by the reg write instruction can be seen in Fig. 13.



(a)

Header		ID	Len	I/E	Param		Cksm
FF	FF	03	04	04	41	01	B2

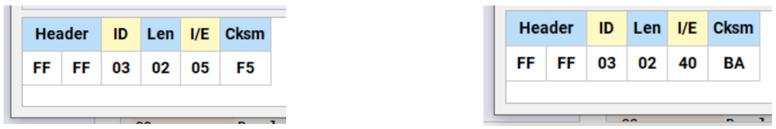
(b)

Header		ID	Len	I/E	Cksm
FF	FF	03	02	08	F2

Fig. 13. Data display (a) Tx communication, (b) Rx communication reg write instruction

5. Action Instruction

   This instruction is to execute the registered Reg Write instruction. The Action instruction is useful when multiple DYNAMIXELs are required to start moving at the same time. The data sent by the Action instruction can be seen in Fig. 14.



(a)

Header		ID	Len	I/E	Cksm
FF	FF	03	02	05	F5

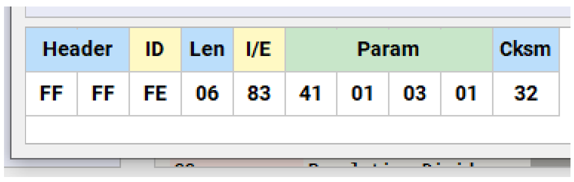
(b)

Header		ID	Len	I/E	Cksm
FF	FF	03	02	40	BA

Fig. 14. Data display (a) Tx communication, (b) Rx communication instruction action

6. Sync Write Instruction

   This instruction is used to control multiple DYNAMIXELs simultaneously with a single Instruction Packet transmission, however, the SYNC WRITE instruction can only be used to a single address with identical data length over connected DYNAMIXELs. The data sent by the ping instruction can be seen in Fig. 15.

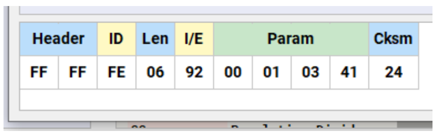


Header		ID	Len	I/E	Param				Cksm
FF	FF	FE	06	83	41	01	03	01	32

Fig. 15. Instruction Tx communication data display

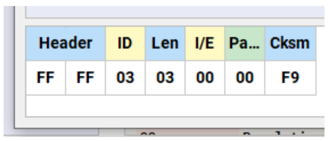
7. Bulk Read Instruction

   This instruction is used to read the value of multiple MX Series DYNAMIXELs simultaneously by sending a single Instruction Packet. The data sent by the bulk read instruction can be seen in Fig. 16.



Header		ID	Len	I/E	Param				Cksm
FF	FF	FE	06	92	00	01	03	41	24

(a)



Header		ID	Len	I/E	Pa...	Cksm
FF	FF	03	03	00	00	F9

(b)

Fig. 16. Data display (a) Tx communication, (b) Rx communication bulk read instruction

3.2. Data capture using OpenCM9.04

In this test, the way the data is collected is via Micro USB contained in OpenCM9.04. This test is carried out by connecting OpenCM9.04 to a PC after opening the DYNAMIXEL Wizard 2.0 software, after which pin 4 found on the controller is connected to the DYNAMIXEL servo and then attach a 2600 mAh battery to turn on the controller found in Fig. 14 above. In this study, the experiments carried out were the same as those using USB2 DYNAMIXEL, namely sending data to the DYNAMIXEL servo that the data conveyed to the DYNAMIXEL servo was complete and as it should be. The difference in this experiment is using the OpenCM9.04 controller. In this test, the data sent by Tx Rx will be recorded using a Logic Analyzer to prove that the data transmission has been conveyed to the DYNAMIXEL servo completely and as it should. The results of the comparison of the data transferred by the Tx Rx controller communication with the Logic Analyzer can be seen in the instruction tables. The results of the Tx Rx communication data comparison and the Logic Analyzer recorder, where the results of the Logic Analyzer display the same results as those displayed by the Tx OpenCM9.04 communication and the Rx communication on the Logic Analyzer also displays the same results as those displayed by the OpenCM904 Rx communication.

1. Ping Instruction

Table 1. Tx and Logic Analyzer communication comparison

OpenCM9.04											
Communication Tx Ping						Logic Analyzer					
Header		ID	Len	I/E	Cksm	Header		ID	Len	I/E	Cksm
FF	FF	05	02	01	F7	0×FF	0×FF	0×05	0×02	0×01	0×F7

Table 2. Comparison of Rx communication and Logic Analyzer

OpenCM9.04											
Communication Rx Ping						Logic Analyzer					
Header		ID	Len	I/E	Cksm	Header		ID	Len	I/E	Cksm
FF	FF	05	02	00	FB	0×FF	0×FF	0×05	0×02	0×00	0×FB

2. Read instruction

Table 3. Tx and Logic Analyzer communication comparison

OpenCM9.04															
Communication Tx Read								Logic Analyzer							
Header		ID	Len	I/E	Param		Cksm	Header		ID	Len	I/E	Param		Cksm
FF	FF	05	04	02	01	02	F1	0×FF	0×FF	0×05	0×04	0×01	0×01	0×02	0×F1

Table 4. Comparison of Rx and Logic Analyzer communication

OpenCM9.04															
Communication Tx Read								Logic Analyzer							
Header		ID	Len	I/E	Param		Cksm	Header		ID	Len	I/E	Param		Cksm
FF	FF	05	04	00	00	28	CE	0×FF	0×FF	0×05	0×04	0×00	0×00	0×28	0×CE

3. Write Instruction

Table 5. Tx and Logic Analyzer communication comparison

OpenCM9.04															
Communication Tx Write								**Logic Analyzer**							
Header		**ID**	**Len**	**I/E**	**Param**		**Cksm**	**Header**		**ID**	**Len**	**I/E**	**Param**		**Cksm**
FF	FF	05	04	03	01	01	F1	0×FF	0×FF	0×05	0×04	0×03	0×01	0×01	0×F1

Table 6. Comparison of Rx and Logic Analyzer communication

OpenCM9.04											
Communication Rx Write						**Logic Analyzer**					
Header		**ID**	**Len**	**I/E**	**Cksm**	**Header**		**ID**	**Len**	**I/E**	**Cksm**
FF	FF	05	02	08	F0	0×FF	0×FF	0×05	0×02	0×08	0×F0

4. Reg Write Instruction

Table 7. Tx and Logic Analyzer communication comparison

OpenCM9.04															
Communication Tx Reg Write								**Logic Analyzer**							
Header		**ID**	**Len**	**I/E**	**Param**		**Cksm**	**Header**		**ID**	**Len**	**I/E**	**Param**		**Cksm**
FF	FF	05	04	04	01	01	F0	0×FF	0×FF	0×05	0×04	0×04	0×01	0×01	0×F0

Table 8. Comparison of Rx and Logic Analyzer communication

OpenCM9.04											
Communication Rx Reg Write						**Logic Analyzer**					
Header		**ID**	**Len**	**I/E**	**Cksm**	**Header**		**ID**	**Len**	**I/E**	**Cksm**
FF	FF	05	02	08	F0	0×FF	0×FF	0×05	0×02	0×08	0×F0

5. Reg Write Instruction

Table 9. Tx and Logic Analyzer communication comparison

OpenCM9.04											
Communication Tx Action						**Logic Analyzer**					
Header		**ID**	**Len**	**I/E**	**Cksm**	**Header**		**ID**	**Len**	**I/E**	**Cksm**
FF	FF	05	02	05	F3	0×FF	0×FF	0×05	0×02	0×05	0×F3

Table 10. Comparison of Rx communication and Logic Analyzer

OpenCM9.04											
Communication Rx Action						**Logic Analyzer**					
Header		**ID**	**Len**	**I/E**	**Cksm**	**Header**		**ID**	**Len**	**I/E**	**Cksm**
FF	FF	05	02	40	B8	0×FF	0×FF	0×05	0×02	0×40	0×B8

6. Sync Write Instruction

Table 11. Tx and Logic Analyzer communication comparison

OpenCM9.04																			
Communication Tx Sync Write										**Logic Analyzer**									
Header		**ID**	**Len**	**I/E**	**Param**				**Cksm**	**Header**		**ID**	**Len**	**I/E**	**Param**				**Cksm**
FF	FF	FE	06	83	01	01	05	01	70	0×FF	0×FF	0×FE	0×06	0×83	0×01	0×01	0×05	0×01	0×70

7. Bulk Read Instruction

Table 12. Tx and Logic Analyzer communication comparison

OpenCM9.04																			
Communication Tx Bulk Read										**Logic Analyzer**									
Header		**ID**	**Len**	**I/E**	**Param**				**Cksm**	**Header**		**ID**	**Len**	**I/E**	**Param**				**Cksm**
FF	FF	FE	06	92	00	01	05	01	62	0×FF	0×FF	0×FE	0×06	0×92	0×00	0×01	0×05	0×01	0×62

Table 13. Comparison of Rx and Logic Analyzer communication

OpenCM9.04													
Communication Rx Bulk Read							**Logic Analyzer**						
Header		**ID**	**Len**	**I/E**	**Param**	**Cksm**	**Header**		**ID**	**Len**	**I/E**	**Param**	**Cksm**
FF	FF	05	03	00	00	F7	0×FF	0×FF	0×05	0×03	0×00	0×00	0×F7

3.3. Gyroscope and Accelerometermeter data capture

In this test, the way the data is collected is via Micro USB contained in OpenCM9.04. This test is carried out by connecting OpenCM9.04 to a PC after opening the RoboPlus software and then entering the DYNAMIXEL Wizard section. Then the Micro USB contained in the controller is connected to the DYNAMIXEL servo and then attach a 2600 mAh battery to turn on the controller.

1. Straight Position Data Capture of the Robot Facing Up

   In this test, the robot will be put to sleep on the scales with the robot's position facing up to test the Gyroscope and Accelerometer sensors contained in the robot, the test results obtained can be seen in table 14.

Table 14. Straight Position Data of the Robot Facing Up

Data Retrieval of Robot's Straight Position Facing Up					
Gyroscope			**Accelerometer**		
Z	**Y**	**X**	**X**	**Y**	**Z**
520	515	512	755	523	480

2. Straight Position Data Capture Robot Face down / Prone

   In this test, the robot will be put to sleep on the scales with the robot's position facing down / Prone to test the Gyroscope and Accelerometer sensors contained in the robot, the test results obtained can be seen in Table 15.

Table 15. Straight Position Data of Downward Facing / Prone Robot

Data Retrieval of Robot's Straight Position Facing Down / Prone					
Gyroscope			**Accelerometer**		
Z	**Y**	**X**	**X**	**Y**	**Z**
519	513	513	255	516	408

3. Data Collection of Robot Position when Standing

   In this test, the robot will be stood on a field track to test the Gyroscope and Accelerometer sensors contained in the robot, the test results obtained can be seen in Table 16.

Table 16. Data on Robot Position when Standing

Retrieval of Robot Position Data while Standing					
Gyroscope			**Accelerometer**		
Z	**Y**	**X**	**X**	**Y**	**Z**
520	514	513	375	512	700

From tables 14, 15 and 16 are the position of the robot when it is on its back, stomach and standing which produces values on the Accelerometer and Gyroscope, namely (X = pitch, Y = roll, Z =yaw). Where the Accelerometer shows the value to determine the position of the robot, while the gyroscope shows the value for the balance of the robot.

4. Conclusion

From the research that has been done, the conclusions are obtained: The OpenCM9.04 controller design can communicate with the MPU-9250 sensor and DYNAMIXEL servo. Gyroscope sensor to read the balance on the robot, Accelerometer shows the value to know the position on the robot while the DYNAMIXEL servo to move the servo angles based on the robot camera input and the designed controller can be applied directly to the robot properly.

References

[1] R. B. Hill, D. Six, A. Chriette, S. Briot and P. Martinet, "Crossing type 2 singularities of parallel robots without pre-planned trajectory with a virtual-constraint-based controller," *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6080-6085, 2017.

[2] K. Saurav, S. Sonkar, M. Raj and G. C. Nandi, "ZMP based feedback control of ankle joint," *2015 International Conference on Industrial Instrumentation and Control (ICIC)*, pp. 1032-1037, 2015.

[3] J. Kim, A. Alspach and K. Yamane, "3D printed soft skin for safe human-robot interaction," *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2419-2425, 2015.

[4] H. Ashari, “STM32 ARM Cortex-M as a microcontroller learning media,” *E-JPTE (Jurnal Elektronik Pendidikan Teknik Elektronika)*, vol. 8, no. 1, pp. 10-18, 2018.

[5] Liu Chunyang, Chen Fan, Sui Xin, Cheng Hongtao, Xu Junling and Xue Yujun, "Gesture detection and data fusion based on MPU9250 sensor," *2015 12th IEEE International Conference on Electronic Measurement & Instruments (ICEMI)*, pp. 1612-1615, 2015.

[6] F. A. Wahab, D. S. Asih, M. Surya, and A. K. Nugraha, “Embedded System Design in Humanoid Robot Soccer Team GLADIATOS UI,” pp. 27–32, 2017.

[7] N. Habibie, M. R. Alhamidi, J. Purnomo, D. Marhaendro, and M. F. Rachmadi, “Performance Comparison of USArt Communication Between Real TIME Operating System (Rtos) and Native Interrupt,” *Jurnal Ilmu Komputer dan Informasi*, vol. 9, no. 1, pp. 43-51, 2016.

[8] S. Wan, Z. Gu, and Q. Ni, “Cognitive computing and wireless communications on the edge for healthcare service robots,” *Computer Communications*, vol. 149, pp. 99-106, 2020.

[9] L. Pérez, I. Rodríguez, N. Rodríguez, R. Usamentiaga, and D. F. García, “Robot guidance using machine vision techniques in industrial environments: A comparative review,” *Sensors*, vol. 16, no. 3, p. 335, 2016.

[10] R. M. Woo-Garcia, U. H. Lomeli-Dorantes, F. López-Huerta, A. L. Herrera-May and J. Martínez-Castillo, "Design and implementation of a system access control by RFID," *2016 IEEE International Engineering Summit, II Cumbre Internacional de las Ingenierias (IE-Summit)*, pp. 1-4, 2016.