DAFTAR PUSTAKA

Ahmad, N., Ghazilla, R. A. R., Khairi, N. M., & Kasi, V. (2013). Reviews on Various Inertial Measurement Unit. International Journal of Signal Processing Systems, 1(2), 256-262.

AL-Rousan, M., & Assaleh, K. (2011). A wavelet- and neural network-based voice system for a smart wheelchair control. Journal of the Franklin Institute, 348(1), 90-100.

Anshar, M. (2009). Embedded Robotics Implementation. In Proceedings of the First International Workshop on Modern Research Methods in Electrical Engineering 2009 (IWoRMEE'09) (pp. 1-5). Makassar City, Indonesia: Electrical Engineering Department, Hasanuddin University.

Automotive Power. (2014). BTS 7960 High Current PN Half Bridge NovalithIC TM.http://www.robotpower.com/downloads/BTS7960_v1.1_2004-12-07.pdf.

Carailmu. (2021, Juni 18). Perbedaan Sistem Kendali Loop Terbuka dan Loop Tertutup. https://www.carailmu.com/2021/06/open-loop-close-loop.html

CEVA, Inc. (2023). BNO08X Datasheet Revision 1.16.

CEVA Technologies, Inc. (2019). BNO080/BNO085 Sensor Calibration Procedure (Document Revision 1.3).

Fitzgerald, A. E. (1984). Basic Electrical Engineering. Jakarta: Erlangga.

Fujitsu. 2010. Capacitive Touch Sensors Application Fields, technology overview and implementation example. https://www.fujitsu.com/downloads/MICRO/fme/articles/fujitsuwhitepaper-capacitive-touchsensors.pdf.

Handson Technology. (2016). BTS7960 High Current 43A H-Bridge Motor Driver. https://www.handsontec.com/dataspecs/module/BTS7960%20Motor%20Driver.pdf.

KBBI. (2018). Kamus Besar Bahasa Indonesia (KBBI). https://kbbi.kemdikbud.go.id/.

Machmud, M. T. (2019). Sistem Navigasi pada Prototipe Robot Kursi Beroda untuk Penyandang Disabilitas. Bachelor's Thesis, Department of Electrical Engineering, Faculty of Engineering, Hasanuddin University, Makassar.

Muntasir, N. F., Raif, M. R., Hermawan, R., & Anshar, M. (2023). Feasibility Analysis of Smart Wheelchairs Based on Voice Recognition for People with Disability. Enthusiastic: International Journal of Applied Statistics and Data Science, 3(1), 85-96.

Nurdin, K. (2021, Desember 14). Penjelasan & Contoh Sistem Kendali Loop Tertutup & Terbuka. https://www.kakangnurdin.com/2021/12/penjelasan-contoh-sistem-kendali-loop-tertutup-terbuka.html.

Paturungi, A. E. H. (2019). Sistem Komunikasi Wireless pada Multi Robot. Bachelor's Thesis, Department of Electrical Engineering, Faculty of Engineering, Hasanuddin University, Makassar.

Prayudha, R. (2023). Rancang Bangun Robot Kursi Beroda untuk Penyandang Disabilitas. Bachelor's Thesis, Department of Electrical Engineering, Faculty of Engineering, Hasanuddin University, Makassar.

Prawira N, B. (2018). Aplikasi Sensor Inertia Measurement Unit (IMU) untuk Memperbaiki Gerak Berjalan Lurus pada Robot Quadruped. Final Project, Diploma III Program of Mechanical Engineering, Department of Industrial Mechanical Engineering, Vocational Faculty, Sepuluh Nopember Institute of Technology, Surabaya.

Purwadani, B. A., Sofyan, F. I., Mallisa, F. P. P., Ghaisyani, O., & Pramono, N. A.* (2022). Prototype Kursi Roda Cerdas Berbasis Raspberry Pi dan Sistem Kendali Android, Speech Recognition, Touch, dan Gesture Control. Jurnal MIPA, 2(4), 276-281. DOI: 10.17977/um067v2i3p276-281.

Rachmat, M. N. (2022). Rancang Bangun Robot Pendulum Terbalik Beroda Dua dengan Pengendali PID. Bachelor's Thesis, Department of Electrical Engineering, Faculty of Engineering, Hasanuddin University, Makassar.

Ramadhan, A. F. (2021). Analisis Cara Kerja Sensor Ultrasonik Menggunakan Mikrokontroler Arduino Uno Untuk Merancang Alat Deteksi Banjir Secara Otomatis.https://www.semanticscholar.org/paper/0bc897fd57d87a48f71af67889624121865745b7.

Shah, S. N. M., Zakaria, M. N. B., Haron, N., Mahmood, A. K. B., & Naono, K. (2012). Design and evaluation of agent based prioritized dynamic round robin scheduling algorithm on computational grids. AASRI Procedia, 1, 531-543.

Singh, R., Raut, R., & Adhikari, S. (2070). Smart Wheel Chair. Project Report, Department of Electronics and Communication Engineering, Kathmandu Engineering College, Tribhuvan University, Kathmandu, Nepal.

Sibuea, T. P. J., Poekoel, V. C., & Kambey, F. D. (2018). Penerapan Sistem Kontrol Optimal Pada Kursi Roda. Electrical Engineering, Sam Ratulangi University Manado, Journal of Electrical and Computer Engineering, 7(3), 355-359.

Suwanda, Anggi. (2014). Motor DC sebagai penggerak pintu putar otomatis berbasis mikrokontroler ATMega8535. Faculty of Computer Science and Information Technology, Gunadarma University. https://www.semanticscholar.org/paper/7e297d7e9f23996558c72da5e471db3c9d38870d.

van der Zalm, G. M. (2004). Tuning of PID-type controllers: literature overview. DCT Reports (Vol. 2004.054). Eindhoven University of Technology.

Visioli, A. (2006). Practical PID Control. Springer, London. pp. 1-2. ISBN 978-1-84628-585-1. https://link.springer.com/chapter/10.1007/1-84628-586-0_1.

WHO. (2022). Global Report on Health Equity for Persons with Disability. https://www.who.int/publications/i/item/9789240063600.

Wajiansyah, A., Purwadi, H., Astagani, A., & Supriadi, S. (2018). Implementation of master slave method on multiprocessor based embedded system: case study on mobile robot. International Journal of Engineering & Technology, 7(2.2), 53-56. Retrieved from http://www.sciencepubco.com/index.php/IJET

Wajiansyah, A., Supriadi, Noval, Ramadhan, N., Sandria, R., & Pratama, L. M. D. (2020). Implementasi Master-slave Pada Embedded System Menggunakan Komunikasi RS-485. ELKHA, 12(1), 26-31.

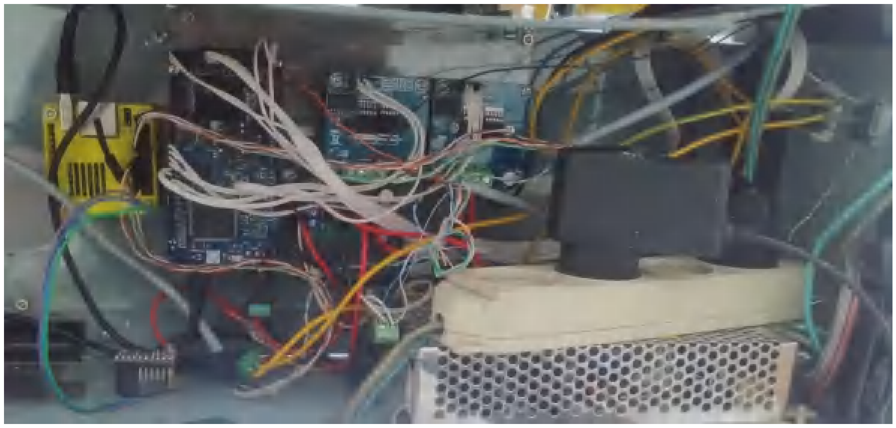
Zhao, H., & Wang, Z. (2012). Motion Measurement Using Inertial Sensors, Ultrasonic Sensors, and Magnetometers With Extended Kalman Filter for Data Fusion. IEEE Sensors Journal, 12(5), 943-953.

LAMPIRAN

Lampiran 1 Tampak depan robot kursi beroda

Lampiran 2 Tampak belakang robot kursi beroda

Lampiran 3 Rangkaian elektronik



Lampiran 4 Program Orange Pi (master)

```
import OPi.GPIO as GPIO
import time
import serial

# Inisialisasi GPIO
GPIO.setmode(GPIO.BOARD)

# Inisialisasi serial untuk komunikasi dengan Arduino
ser = serial.Serial('/dev/ttyUSB0', 115200, timeout=1)
time.sleep(2)  # Tunggu beberapa saat untuk memastikan koneksi serial siap

# Definisikan pin untuk setiap sensor
sensors = {
    "Sensor 1": {"trigger": "PA12", "echo": "PA11"},
    "Sensor 2": {"trigger": "PL0", "echo": "PL1"},
    "Sensor 3": {"trigger": "PA19", "echo": "PA18"},
    "Sensor 4": {"trigger": "PA15", "echo": "PA16"},
    "Sensor 5": {"trigger": "PA14", "echo": "PA13"}
}

# Definisikan pin untuk setiap tombol
buttons = {
    "Button 1": "PA3",
    "Button 2": "PA18",
    "Button 3": "PA19",
    "Button 4": "PA2"
}

# Konfigurasi pin GPIO untuk sensor dan tombol
for sensor, pins in sensors.items():
    GPIO.setup(pins["trigger"], GPIO.OUT)
    GPIO.setup(pins["echo"], GPIO.IN)
```

```
for button, pin in buttons.items():
    GPIO.setup(pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)

def distance(sensor):
    # Kirim sinyal trigger
    GPIO.output(sensors[sensor]["trigger"], GPIO.HIGH)
    time.sleep(0.00001)
    GPIO.output(sensors[sensor]["trigger"], GPIO.LOW)

    # Waktu awal dan akhir
    pulse_start = time.time()
    pulse_end = time.time()

    # Tanggapi sinyal echo
    while GPIO.input(sensors[sensor]["echo"]) == GPIO.LOW:
        pulse_start = time.time()

    while GPIO.input(sensors[sensor]["echo"]) == GPIO.HIGH:
        pulse_end = time.time()

    # Hitung waktu perjalanan suara
    pulse_duration = pulse_end - pulse_start

    # Hitung jarak
    distance = pulse_duration * 17150
    distance = round(distance, 2)

    return distance

def read_serial():
    if ser.in_waiting > 0:
        return ser.readline().decode('utf-8').strip()
    return None

try:
    robot_state = "diam"

    while True:
        # Membaca jarak dari semua sensor
        distances = {sensor: distance(sensor) for sensor in sensors}

        for sensor, dist in distances.items():
            print(f"{sensor}: {dist} cm")

        # Cek status tombol
        button_pressed = None
        for button, pin in buttons.items():
            if GPIO.input(pin) == GPIO.LOW:
                button_pressed = button
```

```
                break

        if button_pressed:
            if button_pressed == "Button 1":
                if distances["Sensor 1"] >= 60 and distances["Sensor 2"] >= 60:
                    print("Maju")
                    ser.write(b'1')  # Kirim kode 1 ke Arduino
                else:
                    print("Kondisi tidak sesuai")
            elif button_pressed == "Button 2":
                if distances["Sensor 5"] >= 60:
                    print("Mundur")
                    ser.write(b'2')  # Kirim kode 2 ke Arduino
                else:
                    print("Kondisi tidak sesuai")
            else:
                print("Kondisi tidak sesuai")

        # Baca data dari Arduino
        arduino_code = read_serial()
        if arduino_code == '1':
            robot_state = "maju"
        elif arduino_code is None:
            robot_state = "diam"

        # Tindakan berdasarkan kondisi robot
        if robot_state == "maju":
            print("Robot maju")
            if distances["Sensor 1"] < 60 or distances["Sensor 2"] < 60:
                ser.write(b'3')  # Kirim kode 3 ke Arduino

        time.sleep(1)

except KeyboardInterrupt:
    GPIO.cleanup()
    ser.close()
```

Lampiran 5 Program Arduino mode rintangan

```
#include <Arduino.h>
#include <Adafruit_BNO08x.h>
#include <Wire.h>

#define TURN_LIMIT 150
#define PWM_FORWARD 30
#define PWM_TURN 30
#define TURN_TIME 1500
#define STRAIGHT_TIME 1000
#define FORWARD2_TIME 2000
#define TURN_LEFT2_TIME 1300
#define STRAIGHT2_TIME 1000
#define TURN_RIGHT2_TIME 1800
#define STOP_TIME 1000

#define BNO08X_CS 10
#define BNO08X_INT 9
#define BNO08X_RESET -1

#define M1_RPWM 2
#define M1_LPWM 3
#define M1_L_EN 4
#define M1_R_EN 5

#define M2_RPWM 6
#define M2_LPWM 7
#define M2_L_EN 8
#define M2_R_EN 9

enum class State { FORWARD, TURN_RIGHT, STRAIGHT, TURN_LEFT, FORWARD2,
TURN_LEFT2, STRAIGHT2, TURN_RIGHT2, END };
State state = State::FORWARD;
unsigned long stateMillis;
unsigned long previousPingMillis[5] = {0, 0, 0, 0, 0}; // untuk setiap sensor
const long pingInterval = 50;  // interval di mana Anda ingin melakukan ping (50 milidetik
dalam kasus ini)

struct euler_t {
  float yaw;
  float pitch;
  float roll;
} ypr, offset;

Adafruit_BNO08x  bno08x(BNO08X_RESET);
sh2_SensorValue_t sensorValue;
sh2_SensorId_t reportType = SH2_ARVR_STABILIZED_RV;
long reportIntervalUs = 5000;
bool calibrated = false;
```

```
float pwm_kiri = 35.0, pwm_kanan = 35.0;
float target_yaw = 0.0;
float KP = 10.0, KI = 3.0, KD = 4.0;
float integral_error = 0.0, previous_error = 0.0;
float dt = 0.1; // Waktu antara sampel dalam detik. Ganti dengan waktu aktual antara
loop() jika diperlukan

void setReports(sh2_SensorId_t reportType, long report_interval) {
  if (! bno08x.enableReport(reportType, report_interval)) {
    Serial.println("Could not enable report");
  }
}
void setupMotor();
float PID_Controller(float target, float actual);
void motor(float pwmki, float pwmka);
void quaternionToEuler(float qr, float qi, float qj, float qk, euler_t* ypr, bool degrees =
false) {
    float sqr = sq(qr);
    float sqi = sq(qi);
    float sqj = sq(qj);
    float sqk = sq(qk);

    ypr->yaw = atan2(2.0 * (qi * qj + qk * qr), (sqi - sqj - sqk + sqr));
    ypr->pitch = asin(-2.0 * (qi * qk - qj * qr) / (sqi + sqj + sqk + sqr));
    ypr->roll = atan2(2.0 * (qj * qk + qi * qr), (-sqi - sqj + sqk + sqr));

    if (degrees) {
      ypr->yaw *= RAD_TO_DEG;
      ypr->pitch *= RAD_TO_DEG;
      ypr->roll *= RAD_TO_DEG;
    }
    if (ypr->yaw < 0) ypr->yaw += 360;
    if (ypr->pitch < 0) ypr->pitch += 360;
    if (ypr->roll < 0) ypr->roll += 360;
}
void quaternionToEulerRV(sh2_RotationVectorWAcc_t* rotational_vector, euler_t* ypr,
bool degrees = false) {
    quaternionToEuler(rotational_vector->real, rotational_vector->i, rotational_vector->j,
rotational_vector->k, ypr, degrees);
}
void setupMotor() {
  // Setup Motor Control Pins
  int motorPins[] = {2, 3, 4, 5, 6, 7, 8, 9};
  for (int i = 0; i < 8; i++) {
    pinMode(motorPins[i], OUTPUT);
  }
  // Enable motor
  digitalWrite(M1_L_EN, HIGH);
  digitalWrite(M1_R_EN, HIGH);
  digitalWrite(M2_L_EN, HIGH);
```

```
  digitalWrite(M2_R_EN, HIGH);
}

void motor(float pwmki, float pwmka) {
  pwmki = constrain(pwmki, -255, 255);
  pwmka = constrain(pwmka, -255, 255);
  if (pwmki <= -1 && pwmki > -255) {
    analogWrite(M2_RPWM, (int)(-pwmki));
    analogWrite(M2_LPWM, 0);
  }
  if (pwmki >= 1 && pwmki < 255) {
    analogWrite(M2_RPWM, 0);
    analogWrite(M2_LPWM, (int)pwmki);
  }
  if (pwmka >= 1 && pwmka < 255) {
    analogWrite(M1_RPWM, (int)pwmka);
    analogWrite(M1_LPWM, 0);
  }
  if (pwmka <= -1 && pwmka > -255) {
    analogWrite(M1_RPWM, 0);
    analogWrite(M1_LPWM, (int)(-pwmka));
  }
  if (pwmka > -1 && pwmka < 1) {
    analogWrite(M1_RPWM, 0);
    analogWrite(M1_LPWM, 0);
  }
  if (pwmki > -1 && pwmki < 1) {
    analogWrite(M2_RPWM, 0);
    analogWrite(M2_LPWM, 0);
  }
}
float PID_Controller(float target, float actual) {
  float error = target - actual;
  integral_error += error * dt;
  float derivative_error = (error - previous_error) / dt;
  previous_error = error;

  return KP * error + KI * integral_error + KD * derivative_error;
}

void setup() {
  Serial.begin(115200);
  if (!bno08x.begin_I2C()) {
    Serial.println("Failed to find BNO08x chip");
    while (1) { delay(10); }
  }
  setReports(reportType, reportIntervalUs);

  setupMotor();
}
```

```
void loop() {
  if (Serial.available() > 0) {
    char instruksi = Serial.read();
    if (instruksi == '1') {
      Serial.write('3'); // Kirim kode 3 ke Orange Pi
        while (true) {
          if (bno08x.wasReset()) {
            setReports(reportType, reportIntervalUs);
          }

          if (bno08x.getSensorEvent(&sensorValue)) {
            if (sensorValue.sensorId == reportType) {
              quaternionToEulerRV(&sensorValue.un.arvrStabilizedRV, &ypr, true);
              if (!calibrated) {
                offset = ypr;  // save the first reading as the offset
                calibrated = true;
              }
              ypr.yaw -= offset.yaw;
              ypr.pitch -= offset.pitch;
              ypr.roll -= offset.roll;

              // Normalize to 0-360
              if (ypr.yaw < 0) ypr.yaw += 360;
              if (ypr.pitch < 0) ypr.pitch += 360;
              if (ypr.roll < 0) ypr.roll += 360;

              Serial.println();
              Serial.print("Yaw: "); Serial.print(ypr.yaw); Serial.print("\t");

              switch (state) {
                case State::FORWARD:
                if (Serial.available() > 0 && Serial.read() == '3') {
                    state = State::TURN_RIGHT;
                    stateMillis = millis();
                    Serial.println("Obstacle detected. Turning right.");
                    motor(30, -30);
                  } else {
                    Serial.print("Moving forward.");
                    if (calibrated) {
                      float error = target_yaw - ypr.yaw;
                      float adjust = PID_Controller(target_yaw, ypr.yaw);

                      if (ypr.yaw >= 0.5 && ypr.yaw <= 180) {
                        pwm_kiri = constrain(pwm_kiri - adjust, 30.0, 55.0);
                        pwm_kanan = constrain(pwm_kanan + adjust, 30.0, 55.0);
                      } else if (ypr.yaw > 180 && ypr.yaw <= 359.9) {
                        pwm_kiri = constrain(pwm_kiri + adjust, 30.0, 55.0);
                        pwm_kanan = constrain(pwm_kanan - adjust, 30.0, 55.0);
                      }
                      motor(pwm_kiri, pwm_kanan);
```

```
    Serial.print("\tPWM Kiri: "); Serial.print(pwm_kiri);
    Serial.print("\tPWM Kanan: "); Serial.println(pwm_kanan);
   }
  }
  break;

 case State::TURN_RIGHT:
  if (millis() - stateMillis >= TURN_TIME) {
   state = State::STRAIGHT;
   stateMillis = millis();
   Serial.println("Turn right completed. Moving straight.");
   motor(30, 40);
  } else {
   Serial.print("Turning right.");
   motor(30, -30);
  }
  break;

 case State::STRAIGHT:
  if (millis() - stateMillis >= STRAIGHT_TIME) {
   state = State::TURN_LEFT;
   stateMillis = millis();
   Serial.println("Straight completed. Turning left.");
   motor(0, 0);
  } else {
   Serial.print("Moving straight.");
   motor(30, 35);
  }
  break;

 case State::TURN_LEFT:
  if (millis() - stateMillis >= TURN_TIME) {
   state = State::FORWARD2;
   stateMillis = millis();
   Serial.println("Turn left completed. Moving forward2.");
   motor(30, 35);
  } else {
   Serial.print("Turning left.");
   motor(-30, 30);
  }
  break;

 case State::FORWARD2:
  if (millis() - stateMillis >= FORWARD2_TIME) {
   state = State::TURN_LEFT2;
   stateMillis = millis();
   Serial.println("Forward2 completed. Turning left2.");
   motor(0, 0);
  } else {
   Serial.print("Moving forward2.");
```

```
      motor(30, 35);
    }
    break;

  case State::TURN_LEFT2:
    if (millis() - stateMillis >= TURN_LEFT2_TIME) {
      state = State::STRAIGHT2;
      stateMillis = millis();
      Serial.println("Turn left2 completed. Moving straight2.");
      motor(30, 35);
    } else {
      Serial.print("Turning left2.");
      motor(-30, 30);
    }
    break;

  case State::STRAIGHT2:
    if (millis() - stateMillis >= STRAIGHT2_TIME) {
      state = State::TURN_RIGHT2;
      stateMillis = millis();
      Serial.println("Straight2 completed. Turning right2.");
      motor(30, -30);
    } else {
      Serial.print("Moving straight2.");
      motor(30, 35);
    }
    break;

  case State::TURN_RIGHT2:
    if (millis() - stateMillis >= TURN_RIGHT2_TIME) {
      state = State::END;
      stateMillis = millis();
      Serial.println("Turn right2 completed. Ending.");
      motor(0, 0);
    } else {
      Serial.print("Turning right2.");
      motor(30, -30);
    }
    break;

  case State::END:
    if (millis() - stateMillis >= STOP_TIME) {
      state = State::FORWARD;
      stateMillis = millis();
      Serial.println("1 second elapsed. Returning to forward state.");
    }
    break;
  }
 }
}
```

```
      }
    }
  }
 else {
   Serial.println("Tunggu Tombol Ditekan");
   delay(1000);
 }
}
```

Lampiran 6 Program Arduino mode tanpa rintangan

```
#include <Arduino.h>
#include <Adafruit_BNO08x.h>

#define M1_RPWM 2
#define M1_LPWM 3
#define M1_L_EN 4
#define M1_R_EN 5
#define M2_RPWM 6
#define M2_LPWM 7
#define M2_L_EN 8
#define M2_R_EN 9

#define BNO08X_CS 10
#define BNO08X_INT 9
#define BNO08X_RESET -1

struct euler_t {
  float yaw;
  float pitch;
  float roll;
} ypr, offset;

Adafruit_BNO08x  bno08x(BNO08X_RESET);
sh2_SensorValue_t sensorValue;
sh2_SensorId_t reportType = SH2_ARVR_STABILIZED_RV;
long reportIntervalUs = 5000;

bool calibrated = false;

float pwm_kiri = 35.0, pwm_kanan = 35.0;
float target_yaw = 0.0;
float KP = 10.0, KI = 3.0, KD = 4.0;
float integral_error = 0.0, previous_error = 0.0;
float dt = 0.1;
void setReports(sh2_SensorId_t reportType, long report_interval) {
  if (! bno08x.enableReport(reportType, report_interval)) {
    Serial.println("Could not enable report");
  }
```

```
} void setupMotor();
//void adjustPWM();
float PID_Controller(float target, float actual);
void motor(float pwmki, float pwmka);

void setup() {
  Serial.begin(115200);
  if (!bno08x.begin_I2C()) {
    Serial.println("Failed to find BNO08x chip");
    while (1) { delay(10); }
  }
  setReports(reportType, reportIntervalUs);

  setupMotor();
}

void quaternionToEuler(float qr, float qi, float qj, float qk, euler_t* ypr, bool degrees =
false) {
    float sqr = sq(qr);
    float sqi = sq(qi);
    float sqj = sq(qj);
    float sqk = sq(qk);

    ypr->yaw = atan2(2.0 * (qi * qj + qk * qr), (sqi - sqj - sqk + sqr));
    ypr->pitch = asin(-2.0 * (qi * qk - qj * qr) / (sqi + sqj + sqk + sqr));
    ypr->roll = atan2(2.0 * (qj * qk + qi * qr), (-sqi - sqj + sqk + sqr));

    if (degrees) {
      ypr->yaw *= RAD_TO_DEG;
      ypr->pitch *= RAD_TO_DEG;
      ypr->roll *= RAD_TO_DEG;
    }

    if (ypr->yaw < 0) ypr->yaw += 360;
    if (ypr->pitch < 0) ypr->pitch += 360;
    if (ypr->roll < 0) ypr->roll += 360;
}

void quaternionToEulerRV(sh2_RotationVectorWAcc_t* rotational_vector, euler_t* ypr,
bool degrees = false) {
    quaternionToEuler(rotational_vector->real, rotational_vector->i, rotational_vector->j,
rotational_vector->k, ypr, degrees);
}

void loop() {
  if (Serial.available() > 0) {
    char instruksi = Serial.read();
    if (instruksi == '1') {
    while (true) {
      if (bno08x.wasReset()) {
```

```
      setReports(reportType, reportIntervalUs);
    }

    if (bno08x.getSensorEvent(&sensorValue)) {
      if (sensorValue.sensorId == reportType) {
        quaternionToEulerRV(&sensorValue.un.arvrStabilizedRV, &ypr, true);
        if(!calibrated) {
          offset = ypr;  // save the first reading as the offset
          calibrated = true;
        }
        ypr.yaw -= offset.yaw;
        ypr.pitch -= offset.pitch;
        ypr.roll -= offset.roll;

        // Normalize to 0-360
        if (ypr.yaw < 0) ypr.yaw += 360;
        if (ypr.pitch < 0) ypr.pitch += 360;
        if (ypr.roll < 0) ypr.roll += 360;
    if (calibrated) {
      float error = target_yaw - ypr.yaw;
      float adjust = PID_Controller(target_yaw, ypr.yaw);

      if (ypr.yaw >= 0.5 && ypr.yaw <= 180) {
        pwm_kiri = constrain(pwm_kiri - adjust, 35.0, 50.0);
        pwm_kanan = constrain(pwm_kanan + adjust, 35.0, 50.0);
      } else if (ypr.yaw > 180 && ypr.yaw <= 359.9) {
        pwm_kiri = constrain(pwm_kiri + adjust, 35.0, 50.0);
        pwm_kanan = constrain(pwm_kanan - adjust, 35.0, 50.0);
      }

      motor(pwm_kiri, pwm_kanan);

      Serial.print("Yaw: "); Serial.print(ypr.yaw);
      Serial.print("\tPWM Kiri: "); Serial.print(pwm_kiri);
      Serial.print("\tPWM Kanan: "); Serial.println(pwm_kanan);
    }
  }
    }
  }
}
else if (instruksi == '2') {

  while (true) {
    if (bno08x.wasReset()) {
      setReports(reportType, reportIntervalUs);
    }

    if (bno08x.getSensorEvent(&sensorValue)) {
      if (sensorValue.sensorId == reportType) {
        quaternionToEulerRV(&sensorValue.un.arvrStabilizedRV, &ypr, true);
```

```
        if(!calibrated) {
          offset = ypr;  // save the first reading as the offset
          calibrated = true;
        }
        ypr.yaw -= offset.yaw;
        ypr.pitch -= offset.pitch;
        ypr.roll -= offset.roll;

        // Normalize to 0-360
        if (ypr.yaw < 0) ypr.yaw += 360;
        if (ypr.pitch < 0) ypr.pitch += 360;
        if (ypr.roll < 0) ypr.roll += 360;
      if (calibrated) {
        float error = target_yaw - ypr.yaw;
        float adjust = PID_Controller(target_yaw, ypr.yaw);

        if (ypr.yaw > 1 && ypr.yaw < 180) {
          pwm_kiri = constrain(pwm_kiri + adjust, 35.0, 50.0);
          pwm_kanan = constrain(pwm_kanan - adjust, 35.0, 50.0);
        } else if (ypr.yaw > 180 && ypr.yaw < 359.9) {
          pwm_kiri = constrain(pwm_kiri - adjust, 35.0, 50.0);
          pwm_kanan = constrain(pwm_kanan + adjust, 35.0, 50.0);
        }

        motor(-pwm_kiri, -pwm_kanan);

        Serial.print("Yaw: "); Serial.print(ypr.yaw);
        Serial.print("\tPWM Kiri: "); Serial.print(pwm_kiri);
        Serial.print("\tPWM Kanan: "); Serial.println(pwm_kanan);
      }
    }
      }
    }
  }
  }
  else {
    Serial.println("Tunggu Tombol Ditekan");
    delay(1000);
  }
}

void setupMotor() {
  // Setup Motor Control Pins
  int motorPins[] = {2, 3, 4, 5, 6, 7, 8, 9};
  for (int i = 0; i < 8; i++) {
    pinMode(motorPins[i], OUTPUT);
  }

  // Enable motor
  digitalWrite(M1_L_EN, HIGH);
```

```
  digitalWrite(M1_R_EN, HIGH);
  digitalWrite(M2_L_EN, HIGH);
  digitalWrite(M2_R_EN, HIGH);
}

float PID_Controller(float target, float actual) {
  float error = target - actual;
  integral_error += error * dt;
  float derivative_error = (error - previous_error) / dt;
  previous_error = error;

  return KP * error + KI * integral_error + KD * derivative_error;
}

void motor(float pwmki, float pwmka) {
  pwmki = constrain(pwmki, -255, 255);
  pwmka = constrain(pwmka, -255, 255);

  if (pwmki <= -15 && pwmki > -255) {
    analogWrite(M2_RPWM, (int)(-pwmki));
    analogWrite(M2_LPWM, 0);
  }
  if (pwmki >= 15 && pwmki < 255) {
    analogWrite(M2_RPWM, 0);
    analogWrite(M2_LPWM, (int)pwmki);
  }
  if (pwmka >= 15 && pwmka < 255) {
    analogWrite(M1_RPWM, (int)pwmka);
    analogWrite(M1_LPWM, 0);
  }
  if (pwmka <= -15 && pwmka > -255) {
    analogWrite(M1_RPWM, 0);
    analogWrite(M1_LPWM, (int)(-pwmka));
  }
  if (pwmka > -15 && pwmka < 15) {
    analogWrite(M1_RPWM, 10);
    analogWrite(M1_LPWM, 10);
  }
  if (pwmki > -15 && pwmki < 15) {
    analogWrite(M2_RPWM, 10);
    analogWrite(M2_LPWM, 10);
  }
}
```