LAMPIRAN

LAMPIRAN

Lampiran 1

```
#define TRIGGER_PIN_FRONT A0
#define ECHO_PIN_FRONT A1
#define TRIGGER_PIN_LEFT A2
#define ECHO_PIN_LEFT A3
#define TRIGGER_PIN_RIGHT A4
#define ECHO_PIN_RIGHT A5
#define MAX_DISTANCE  800
#define SONAR_NUM 3
#define MAX_SPEED 200
#define MAX_PID_OUT 1023
#define DIST_START_AVOID 17
#define DIST_MIN 5
#define BACKWARD_DLY 50
int test_speed = 0;

NewPing sonar[SONAR_NUM] = { // Sensor object array.
  NewPing(TRIGGER_PIN_FRONT, ECHO_PIN_FRONT, MAX_DISTANCE), //
Each sensor's trigger pin, echo pin, and max distance to ping.
  NewPing(TRIGGER_PIN_RIGHT, ECHO_PIN_RIGHT, MAX_DISTANCE),
  NewPing(TRIGGER_PIN_LEFT, ECHO_PIN_LEFT, MAX_DISTANCE)

};

//NewPing sonar_front(TRIGGER_PIN_FRONT, ECHO_PIN_FRONT,
MAX_DISTANCE);
//NewPing sonar_left(TRIGGER_PIN_LEFT, ECHO_PIN_LEFT,
MAX_DISTANCE);
//NewPing sonar_right(TRIGGER_PIN_RIGHT, ECHO_PIN_RIGHT,
MAX_DISTANCE);

int motor1Pin1 = 3; //pwm pin
int motor1Pin2 = 4;
int motor2Pin1 = 6; //pwm pin
int motor2Pin2 = 7;
```

```
double Kp = 6.0;
double Ki = 0.2;
double Kd = 2.0;

// Tambahkan konstanta untuk setpoint dan batas integral
const int setpoint = DIST_MIN;
const int integral_limit = 50;

int error_front = 0;
int lastError_front = 0;
int integral_front = 0;
int derivative_front = 0;
int output_front = 0;

int error_side = 0;
int lastError_side = 0;
int integral_side = 0;
int derivative_side = 0;
int output_side = 0;

// Variabel untuk rata-rata bergerak
const int numReadings = 5;
int readings_front[numReadings];
int readings_left[numReadings];
int readings_right[numReadings];
int readIndex = 0;
int total_front = 0;
int total_left = 0;
int total_right = 0;
int average_front = 0;
int average_left = 0;
int average_right = 0;
int distance[SONAR_NUM];
//func init
void turn_right_pid(void);
void turn_left_pid(void);
void forward_pid_reset(void);
```

```
void setup() {
  pinMode(motor1Pin1, OUTPUT);
  pinMode(motor1Pin2, OUTPUT);
  pinMode(motor2Pin1, OUTPUT);
  pinMode(motor2Pin2, OUTPUT);
  Serial.begin(9600);

  // Inisialisasi array untuk rata-rata bergerak
  for (int i = 0; i < numReadings; i++) {
  readings_front[i] = 0;
    readings_left[i] = 0;
    readings_right[i] = 0;
  }
  delay(5000);

}

void loop() {
  for (uint8_t i = 0; i < SONAR_NUM; i++) { // Loop through each sensor and
display results.
    delay(30); // Wait 50ms between pings (about 20 pings/sec). 29ms should be the
shortest delay between pings.
    distance[i] = sonar[i].ping_cm();

  }

  int distance_front = distance[0];
  int distance_left = distance[1];
  int distance_right = distance[2];

  // Rata-rata bergerak
  total_front -= readings_front[readIndex];
  total_left -= readings_left[readIndex];
  total_right -= readings_right[readIndex];
  readings_front[readIndex] = distance_front;
  readings_left[readIndex] = distance_left;
  readings_right[readIndex] = distance_right;
```

```
  total_front += readings_front[readIndex];
  total_left += readings_left[readIndex];
  total_right += readings_right[readIndex];
  readIndex = (readIndex + 1) % numReadings;
  average_front = total_front / numReadings;
  average_left = total_left / numReadings;
  average_right = total_right / numReadings;

  Serial.print("Average Distance Front: ");
  Serial.print(average_front);
  Serial.print(" cm, ");
  Serial.print("Average Distance Left: ");
  Serial.print(distance_left);
  Serial.print(" cm, ");
  Serial.print("Average Distance Right: ");
  Serial.print(average_right);
  Serial.println(" cm");
//
  // Logika untuk menghindari halangan depan, kiri, dan kanan
  //

  //  if (average_front > 3 && average_front < DIST_START_AVOID) {
  if (average_left > DIST_MIN && average_left < DIST_START_AVOID) {
    turn_right_pid();
    forward_pid_reset();
  }
  else if (average_right > DIST_MIN && average_right < DIST_START_AVOID)
  {
    turn_left_pid();
    forward_pid_reset();
  }

  // }
  else if (average_front > 0 && average_front <= DIST_MIN)
  {
    if (average_left <= average_right) //mundur serong kiri sdikit
    {
      m_right_run(0, 60);
```

```
  m_left_run(0, 40);

 }
 else if (average_right < average_left)//mundur serong kanan sdikit
 {
 m_right_run(0, 40);
 m_left_run(0, 60);
 }
 else
 {
  m_run_backward(40);
  turn_pid_reset();
 }
  Serial.println("BACKWARD FRONT");
  delay(BACKWARD_DLY);

  forward_pid_reset();
}
 else if (average_left > 0 && average_left <= DIST_MIN)
 {
    if(average_front==0 ||average_front>150) //mundur serong  kanan sdikit
  {
    m_right_run(0, 30);
   m_left_run(0, 60);

    } else
    {
      m_run_backward(40);
    }
  Serial.println("BACKWARD LEFT");
  delay(BACKWARD_DLY);
  if (average_right > 0 && average_right < DIST_START_AVOID)
  {
   turn_left_pid();

  }
  forward_pid_reset();
 }
```

```
  else  if (average_right > 0 && average_right <= DIST_MIN)
  {

    if(average_front==0 ||average_front>150) //mundur serong kiri sdikit
    {
    m_right_run(0, 60);
    m_left_run(0, 30);

      } else
      {
        m_run_backward(40);
      }
    Serial.println("BACKWARD RIGHT");
    delay(BACKWARD_DLY);
    if (average_left > 0 && average_left < DIST_START_AVOID) {
     turn_right_pid();
    }
    else if (average_right > 0 && average_right < DIST_START_AVOID)
    {
     turn_left_pid();
    }
    forward_pid_reset();
  }
  else
  {
    m_run_forward_pid();
    turn_pid_reset();
    //m_run_forward_distance();
    // m_stop();

  }
void m_run_forward_distance(void)
 {

  if (average_front > 3 && average_front < 100 ) {

    int speed_now = map(average_front, 0, 100, 10, 100);
    m_run_forward(speed_now);
```

```
  Serial.print ("FORWARD =");
  Serial.print(speed_now);
  Serial.println("%");
 }
 else if (average_front < 3)
 {
  m_stop();
 }
 else
 {
  m_run_forward(100);
 }


}


void m_run_forward_pid(void)
{

 //  if (average_front > 0 && average_front < 150)
 int input_disctance=average_front;
 input_disctance=min(input_disctance,average_left);
  input_disctance=min(input_disctance,average_right);

 if (input_disctance > 0 && input_disctance < 200 ) //jarak mulai turun kecepatan
 {
  error_front = input_disctance - setpoint;
  integral_front = integral_front + error_front;

  // Batasi integral windup
  if (integral_front > integral_limit) integral_front = integral_limit;
  else if (integral_front < -integral_limit) integral_front = -integral_limit;

  derivative_front = error_front - lastError_front;
  output_front = Kp * error_front + Ki * integral_front + Kd * derivative_front;
  lastError_front = error_front;
```

```
   output_front = constrain(output_front, 0, 512);
   Serial.print ("OUT PID =");
   Serial.println(output_front);
   output_front = map(output_front, 0, 512, 20, 100);
   int speed_now = output_front;
   m_run_forward(speed_now);
   Serial.print ("FORWARD =");
   Serial.print(speed_now);
   Serial.println("%");

  }
  else
  {
   m_run_forward(100);
   Serial.println ("FORWARD = FULL");

  }
}

void forward_pid_reset(void)
{
  integral_front = 0;
  derivative_front = 0;
  lastError_front = 0;
}
void m_run_forward(int m_speed)
{
  m_speed = constrain(m_speed, 0, 100);
  m_speed = map(m_speed, 0, 100, 0, MAX_SPEED);
  analogWrite(motor1Pin1, m_speed);
  digitalWrite(motor1Pin2 , LOW);

  analogWrite(motor2Pin1, m_speed);
  digitalWrite(motor2Pin2, LOW);

}

void m_run_backward(int m_speed)
```

```
{
  m_speed = constrain(m_speed, 0, 100);
  m_speed = map(m_speed, 0, 100, MAX_SPEED, 0);
  analogWrite(motor1Pin1, m_speed);
  digitalWrite(motor1Pin2 , HIGH);

  analogWrite(motor2Pin1, m_speed);
  digitalWrite(motor2Pin2, HIGH);


}
void m_left_run(int dir, int m_speed)
{
  if (dir == 1)
  {
    m_speed = constrain(m_speed, 0, 100);
    m_speed = map(m_speed, 0, 100, 0, MAX_SPEED);
    analogWrite(motor1Pin1, m_speed);
    digitalWrite(motor1Pin2 , LOW);
  }
  else if (dir == 0)
  {
    m_speed = constrain(m_speed, 0, 100);
    m_speed = map(m_speed, 0, 100, MAX_SPEED, 0);
    analogWrite(motor1Pin1, m_speed);
    digitalWrite(motor1Pin2 , HIGH);

  }

}


void m_right_run(int dir, int m_speed)
{
  if (dir == 1)
  {
    m_speed = constrain(m_speed, 0, 100);
    m_speed = map(m_speed, 0, 100, 0, MAX_SPEED);
```

```
   analogWrite(motor2Pin1, m_speed);
   digitalWrite(motor2Pin2 , LOW);
  }
  else if (dir == 0)
  {
   m_speed = constrain(m_speed, 0, 100);
   m_speed = map(m_speed, 0, 100, MAX_SPEED, 0);
   analogWrite(motor2Pin1, m_speed);
   digitalWrite(motor2Pin2 , HIGH);

  }

}
void m_stop(void)
{

 digitalWrite(motor1Pin1 , LOW);
 digitalWrite(motor1Pin2 , LOW);
 analogWrite(motor2Pin1,  LOW);
 digitalWrite(motor2Pin2, LOW);
}
void turn_pid_reset(void)
{
   integral_side = 0;
 derivative_side = 0;
 lastError_side = 0;

 }
void turn_right_pid(void)
{
 error_side = average_left - setpoint;
 integral_side = integral_side + error_side;

 // Batasi integral windup
 if (integral_side > integral_limit) integral_side = integral_limit;
 else if (integral_side < -integral_limit) integral_side = -integral_limit;

 derivative_side = error_side - lastError_side;
```

```
  output_side = Kp * error_side + Ki * integral_side + Kd * derivative_side;
  lastError_side = error_side;
  output_side =  constrain(output_side, 0, 512);
  output_side = map(output_side, 0, 512, 10, 50);
  m_right_run(0, output_side);
  m_left_run(1, 50);
  Serial.print("TURN RIGHT = ");
  Serial.println(output_side);
  // analogWrite(motor1Pin1, HIGH);
  //digitalWrite(motor1Pin2, 0);
  //analogWrite(motor2Pin1, constrain(output_side, 0, 255));
  //digitalWrite(motor2Pin2, HIGH);
}

void turn_left_pid(void)
{
  error_side = average_left - setpoint;
  integral_side = integral_side + error_side;

  // Batasi integral windup
  if (integral_side > integral_limit) integral_side = integral_limit;
  else if (integral_side < -integral_limit) integral_side = -integral_limit;

  derivative_side = error_side - lastError_side;
  output_side = Kp * error_side + Ki * integral_side + Kd * derivative_side;
  lastError_side = error_side;
  output_side =  constrain(output_side, 0, 512);
  output_side = map(output_side, 0, 512, 10, 50);
  m_left_run(0, output_side);
  m_right_run(1, 50);
  Serial.print("TURN LEFT = ");
  Serial.println(output_side);
}
```

Lampiran 2