DAFTAR PUSTAKA

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

Ang, K. H., Chong, G., & Li, Y. (2015). PID Control System Analysis, Design, PID Control System Analysis, Design,. *IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY, 13*(4), 559-576.

Atmel Corporation. (2015). ATmega328P. San Jose: Atmel press.

Ayenew, E., & Sukhavasi, D. S. (2017). Design and Implementation of Linear Algebraic Controller for Broom Balancing. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, 6*(1), 401-414.

David Rapos, C. M. (2016). Dynamic Sensor Calibration: A Comparative Study of a Hall Effect Sensor and an Incremental Encoder for Measuring Shaft Rotational Position. *2016 IEEE International Conference on Prognostics and Health Management (ICPHM),* , 1-5.

Fahmi, H. Z., Maulana, R., & Kurniawan, W. (2017). Implementasi Complementary Filter Menggunakan Sensor Accelerometer. *urnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 1*(11), 1376-1385.

Johnson, M. A., & Moradi, M. H. (2005). *PID Control New Identification and Design Methods.* London: Springer-Verlag.

Kung, F. (2019). A Tutorial on Modelling and Control of Two-Wheeled Self-Balancing Robot with Stepper Motor. *APPLICATIONS OF MODELLING AND SIMULATION, 3*(2), 64-74.

Li, Z., Yang, C., & Fan, L. (2013). *Advanced Control of Wheeled Inverted Pendulum Systems.* New York: Springer.

Lima, J. L., Goncalves, J. C., Costa, P. G., & Moreira, A. P. (2006). *INVERTED PENDULUM VIRTUAL CONTROL.* Porto: University of Porto Press.

Lundberg, K. H., & Barton, T. W. (2010). History of Inverted-Pendulum Systems. *IFAC Proceedings Volumes, I*(24), 131-135.

Nusantoro, G. D., Muslim, M. A., , P., & C, R. I. (2012). Rancang Bangun Rotary Inverted Pendulum (RIP) Dengan Menggunakan Kontrol PID. *EECCIS, 6*(2), 161-171.

Pathak, K., Franch, J., & Agrawal, S. K. (2005). Velocity and Position Control of a Wheeled Inverted. *IEEE TRANSACTIONS ON ROBOTICS, 21*(3), 505-513.

Pititeeraphab, Y. J. (2016). The Effect of Average Filter for Complementary Filter and Kalman Filter Based on Measurement Angle. *The 2016 Biomedical Engineering International Conference (BMEiCON-2016)*, 105-109.

Rorabaugh , C. B. (1996). *Digital Filter Designer's Handbook.* Franklin: McGraw Hill.

Tim, W. (2000). Embedded System Programming. Dalam *PID without a PhD* (hal. 86-108). Oregon: Elsevier.

Vahid, F., & Givargis, T. (1999). *Embedded System Design: A Unified Hardware/Software Approach.* California: Department of Computer Science and Engineering press.

Van der Zalm, G. M. (2004). *Tuning of PID-type controllers: Literature overview.* Eindhoven: DCT rapporten Eindhoven: Technische Universiteit.

Virgala, I., Kelemen, M., Gmiterko, A., & Lipták, T. (2015). Control of Stepper Motor by Microcontroller. *Journal of Automation and Control, 3*(3), 131-134.

Visioli, A. (2006). *Practical PID Control.* London: Springer.

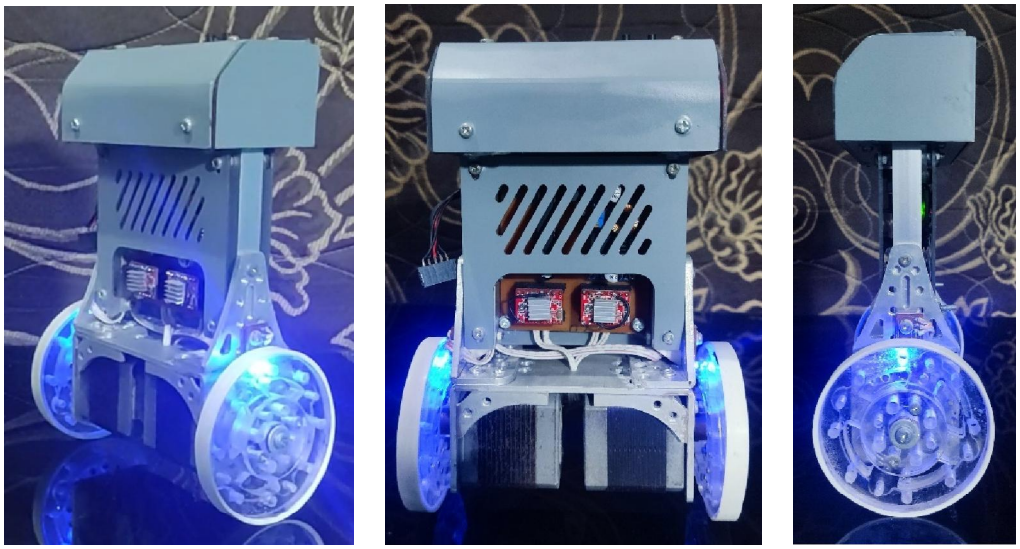
Wu, S. T., Chen, J. Y., & Wu, S. H. (2014). A Rotary Encoder With an Eccentrically Mounted Ring Magnet. *IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT, 63*(8), 1907-1915.

Zhang, D., Wang, J., Qian, L., & Yi, J. (2018). Stepper motor open-loop control system modeling and control strategy optimization. *Archives of electrical engineering , 68*(1), 63-75.

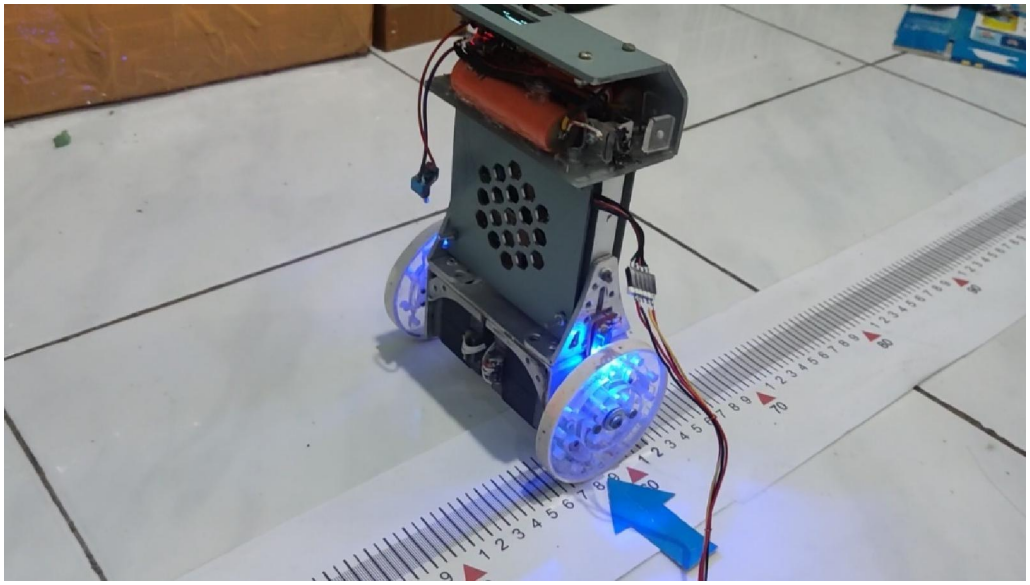
Zhao, H., & Wang, Z. (2012). Motion Measurement Using Inertial Sensors,Ultrasonic Sensors, and Magnetometers With Extended Kalman Filter for Data Fusion. *IEEE SENSORS JOURNAL, 12*(5), 943-953.

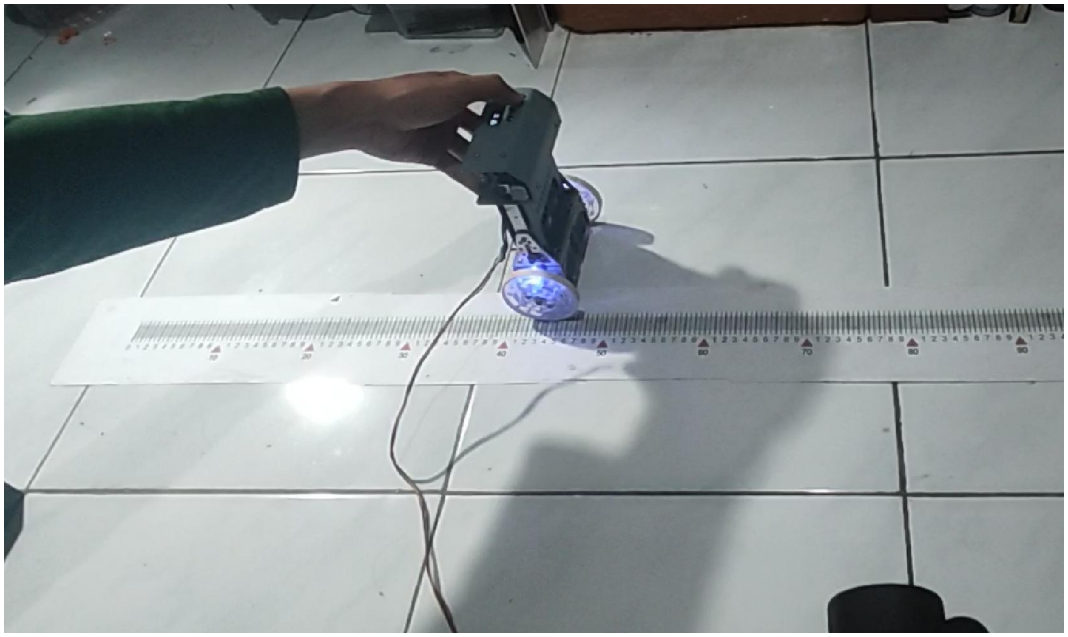
LAMPIRAN

Prototype robot pendulum terbalik beroda dua yang dibangun



Pengujian unjuk kerja robot





Program kendali PID yang diimplementasikan pada Atmega328 dalam bahasa C dengan firmware Arduino

```
#include <EEPROM.h>
#include <Wire.h>
volatile byte command = 0;

volatile byte pos;

bool tes;
float buff9[1];
volatile int posisicm, selektor;
////////////////////////////
volatile int flagdirkan = 1, enckan;
volatile int flagdirkir = 1, enckir, statekir;
volatile int countkan, statekan, countkir;
int pulsa;
////////////////////////////
byte menu;
byte buzz;
volatile byte enckiri, enckanan;
int valkiri, valkanan;
int selectrun, volt;
long skalibrasi;
volatile int valeeprom, selisih;

bool bol = false, tmp = false;
int kon = 0, flag2 = 0, flag3 = 0;


int gyro_address = 0x68;
int acc_calibration_value = 1000;
//settings
float pid_p_gain = 9;
float pid_i_gain = 1.3;
float pid_d_gain = 10;
float turning_speed = 30;
float max_target_speed = 150;
float kpp = 0.4, kip = 0, kdp = 0.15; //kdp = 0.15, kpp 0.25   ///0.4 0.15
////////////////////////////////////////////////////////////////////////////////////////////////
//variabel
////////////////////////////////////////////////////////////////////////////////////////////////
float epos, lastepos, outpos = 0;
volatile int setposisi = 0;
byte start, received_byte, low_bat = 0;

volatile int left_motor, throttle_counter_left_motor, throttle_left_motor_memory;
volatile int throttle_left_motor;
volatile int right_motor, throttle_counter_right_motor, throttle_right_motor_memory;
volatile int throttle_right_motor;
int battery_voltage, printaja = 0;
int gyro_pitch_data_raw, gyro_yaw_data_raw, accelerometer_data_raw;

long gyro_yaw_calibration_value, gyro_pitch_calibration_value;

unsigned long loop_timer;

volatile float angle_gyro, angle_acc, angle, self_balance_pid_setpoint ;
float pid_error_temp, pid_i_mem, pid_setpoint, gyro_input, pid_output,
pid_last_d_error;
float pid_output_left, pid_output_right;

void setup () {
  Serial.begin(2000000);
  DDRD |= 0b11110000; //STEPPER PD4 PD5 PD6 PD7
  DDRB |= 0b00000001; //EN MOTOR
  PORTB |= 0b00000001;
  pinMode(MISO, OUTPUT);
  pinMode(A0, INPUT);
  pinMode(A2, INPUT);
```

```
  DDRC |= 0b00001000;
  PORTC |= 0b00001000;
  delay(50);
  PORTC &= 0b11110111;
  SPCR |= _BV(SPE);
  SPCR |= _BV(SPIE);
  attachInterrupt (0, ss_falling, FALLING);
  Wire.begin();
  TWBR = 12;

  TCCR2A = 0;
  TCCR2B = 0;
  TIMSK2 |= (1 << OCIE2A);
  TCCR2B |= (0 << CS21);
  OCR2A = 39;
  TCCR2A |= (1 << WGM21);

  ////////////////TIMER1 AMBIL DATA
  //      TCCR1A = 0;
  //  TCCR1B = 0;
  //  TIMSK1 |= (1 << OCIE1A);
  //  OCR1A = 780;
  //  TCCR1B |= (1 << WGM12);

  kalibrasimpu();

  //  TCCR1B |= (1 << CS12);
  //  TCCR1B |= (1 << CS10);
}

ISR (SPI_STC_vect) {
  byte c = SPDR;

  switch (command)
  {
    // no command? then this is the command
    case 0:
      command = c;
      SPDR = 0;
      break;

    //untuk nilai menu utama
    case 4:
      menu = c;
      break;

    // untuk kirim terima 8 bit
    case 5:
      SPDR = EEPROM.read(2);
      break;

    // untuk buzzing
    case 6:
      buzz = c;
      break;

    // untuk kirim terima 8 bit
    case 7:
      SPDR = EEPROM.read(1);
      break;

    case 8:
      selectrun = c;
      break;
  }
}

ISR(TIMER2_COMPA_vect) {
  //Left motor pulse calculations
```

```
136     throttle_counter_left_motor ++;
137     if (throttle_counter_left_motor > throttle_left_motor_memory) {
138       throttle_counter_left_motor = 0;
139       throttle_left_motor_memory = throttle_left_motor;
140       if (throttle_left_motor_memory < 0) {
141         PORTD |= 0b01000000;
142         flagdirkir = 1;
143         throttle_left_motor_memory *= -1;
144       }
145       else  {
146         PORTD &= 0b10111111;
147         flagdirkir = 0;
148       }
149     }
150     else if (throttle_counter_left_motor == 1) {
151       PORTD |= 0b10000000;
152     }
153     else if (throttle_counter_left_motor == 2)PORTD &= 0b01111111;
154
155     //right motor pulse calculations
156     throttle_counter_right_motor ++;
157     if (throttle_counter_right_motor > throttle_right_motor_memory) {
158       throttle_counter_right_motor = 0;
159       throttle_right_motor_memory = throttle_right_motor;
160       if (throttle_right_motor_memory < 0) {
161         PORTD |= 0b00100000;
162         throttle_right_motor_memory *= -1;
163       }
164       else {
165         PORTD &= 0b11011111;
166         //           flagdirkan=0;
167       }
168     }
169     else if (throttle_counter_right_motor == 1) {
170       PORTD |= 0b00010000;
171       enc();
172     }
173     else if (throttle_counter_right_motor == 2)PORTD &= 0b11101111;
174
175   }
176
177   void loop () {
178     buzzer();
179     PORTB |= 0b00000001;
180
181     if (selectrun == 1) { ////////////RUN 1
182       delay(100);
183       resetEnc();
184       selisih = 0;
185       kalibrasimpu();
186       TCCR2B |= (1 << CS21);
187       while (1) {
188         buzzer();
189         pidSudut();
190         while (loop_timer > micros());
191         loop_timer += 4000;
192         if (selectrun == 0) {
193           TCCR2B |= (0 << CS21);
194           throttle_counter_left_motor = 0;
195           throttle_counter_right_motor = 0;
196           throttle_left_motor_memory = 0;
197           throttle_right_motor_memory = 0;
198           pid_error_temp = 0;
199           angle_gyro = 0;
200           self_balance_pid_setpoint = 0;
201           pid_setpoint = 0;
202           break;
203         }
204       }
```

```
205     }
206     if (selectrun == 2) { ///////////RUN 2
207       //kalibrasi
208       delay(200);
209       buzzer();
210       resetEnc();
211       selisih = 1;
212       kalibrasimpu();
213       TCCR2B |= (1 << CS21);
214       while (1) {
215         buzzer();
216         pidposisi();
217         pidSudut();
218         while (loop_timer > micros());
219         loop_timer += 4000;
220         if (selectrun == 0) {
221           TCCR2B |= (0 << CS21);
222           throttle_counter_left_motor = 0;
223           throttle_counter_right_motor = 0;
224           throttle_left_motor_memory = 0;
225           throttle_right_motor_memory = 0;
226           pid_error_temp = 0;
227           angle_gyro = 0;
228           self_balance_pid_setpoint = 0;
229           pid_setpoint = 0;
230           setposisi = 0;
231           countkan = 0;
232           countkir = 0;
233           break;
234         }
235       }
236     }
237     if (selectrun == 3) { //////////////RUN 3
238       bol = false;
239       PORTC |= 0b00001000;
240       delay(50);
241       PORTC &= 0b11110111;
242       delay(50);
243     }
244     if (selectrun == 4) { ////////////kalibrasi
245       bol = false;
246       kalibrasi();
247     }
248     if (selectrun == 5) { ////////////kalibrasi
249     }
250   }
251
252   void buzzer () {
253     if (buzz == 1)PORTC |= 0b00001000;
254     else PORTC &= 0b11110111;
255   }
256
257   void ss_falling () {
258     command = 0;
259     pos = 0;
260   }
261
262   void kalibrasi() {
263     PORTB &= 0b11111110;
264     PORTD |= 0b01100000;  //SET DIR STEPPER HIGH  D5,D6
265     PORTD &= 0b10011111;
266     ///////////KANAN
267     for (int i = 0; i < 200; i++) {
268
269       PORTC &= (0 << 3);
270       PORTD |= 0b10010000;
271       delayMicroseconds(1200);
272       PORTD &= 0b01101111;
273       delayMicroseconds(1200);
```

```
      PORTD |= 0b10010000;
      delayMicroseconds(1200);
      PORTD &= 0b01101111;
      delayMicroseconds(1200);
      PORTD |= 0b10010000;
      delayMicroseconds(1200);
      PORTD &= 0b01101111;
      delayMicroseconds(1200);
      PORTD |= 0b10010000;
      delayMicroseconds(1200);
      PORTD &= 0b01101111;
      delayMicroseconds(1200);


    }
    //eeprom(1,countkan);
    //eeprom(2,countkir);
    //Serial.println(countkir);
    //Serial.println(countkan);
    delay(100);
    while (1) {
      PORTB |= 0b00000001;
      buzzer();
      if (selectrun == 0)break;
    }
  }

  void eeprom(int i, int nilai) {

    if (i == 1) {
      EEPROM.write(1, nilai); //KANAN 8
    }
    else if (i == 2) {
      EEPROM.write(2, nilai); //KIRI
    }
    else if (i == 3) {
      valeeprom = EEPROM.read(1);  //KANAN
      valeeprom = valeeprom * 4;
    }
    else if (i == 4) {
      valeeprom = EEPROM.read(2);  //KIRI
      valeeprom = valeeprom * 4;
    }
  }

  void pidSudut() {

    Wire.beginTransmission(gyro_address);
    Wire.write(0x3F);
    Wire.endTransmission();
    Wire.requestFrom(gyro_address, 2);
    accelerometer_data_raw = Wire.read() << 8 | Wire.read();
    accelerometer_data_raw += acc_calibration_value;
    if (accelerometer_data_raw > 8200)accelerometer_data_raw = 8200;
    if (accelerometer_data_raw < -8200)accelerometer_data_raw = -8200;

    angle_acc = asin((float)accelerometer_data_raw / 8200.0) * 57.296;

    if (start == 0 && angle_acc > -0.5 && angle_acc < 0.5) {
      angle_gyro = angle_acc;
      start = 1;
    }

    Wire.beginTransmission(gyro_address);
    Wire.write(0x43);
    Wire.endTransmission();
    Wire.requestFrom(gyro_address, 4);
    gyro_yaw_data_raw = Wire.read() << 8 | Wire.read();
    gyro_pitch_data_raw = Wire.read() << 8 | Wire.read();
```

```
343
344     gyro_pitch_data_raw -= gyro_pitch_calibration_value;
345     angle_gyro += gyro_pitch_data_raw * 0.000031;
346
347
348     gyro_yaw_data_raw -= gyro_yaw_calibration_value;
349     //Uncomment the following line to make the compensation active
350     //angle_gyro -= gyro_yaw_data_raw * -0.0000003;
351
352     angle_gyro = angle_gyro * 0.996 + angle_acc * 0.004;
353     /////////percobaan menghilangkan
354     pid_error_temp = angle_gyro - self_balance_pid_setpoint - pid_setpoint + 1.8;
355     if (pid_output > 5 || pid_output < -5)pid_error_temp += pid_output * 0.015 ;
356
357     pid_i_mem += pid_i_gain * pid_error_temp;
358     if (pid_i_mem > 400)pid_i_mem = 400;
359     else if (pid_i_mem < -400)pid_i_mem = -400;
360     //Calculate the PID output value
361     pid_output = pid_p_gain * pid_error_temp + pid_i_mem + pid_d_gain *
        (pid_error_temp - pid_last_d_error);
362     if (pid_output > 400)pid_output = 400;
363     else if (pid_output < -400)pid_output = -400;
364
365     pid_last_d_error = pid_error_temp;
366
367     if (pid_output < 10 && pid_output > -10)pid_output = 0;
368
369     if (angle_gyro > 30 || angle_gyro < -30 || start == 0 || low_bat == 1) {
370       pid_output = 0;
371       pid_i_mem = 0;
372       start = 0;
373       self_balance_pid_setpoint = 0;
374     }
375
376
377     ////////////////////////////////////////////////////////////////////////////////////
        ////////////////////////////////////////////////
378     //Control calculations
379
        ////////////////////////////////////////////////////////////////////////////////////
        ////////////////////////////////////////////////
380     pid_output_left = pid_output;
381     pid_output_right = pid_output;
382
383
384     if (selisih == 0) {
385       if (pid_output < 0)self_balance_pid_setpoint += 0.0015;
386       if (pid_output > 0)self_balance_pid_setpoint -= 0.0015;
387       //terindikasi adjust
388     }
389     //else self_balance_pid_setpoint = 0;
390
        ////////////////////////////////////////////////////////////////////////////////////
        ////////////////////////////////////////////////
391     //Motor pulse calculations
392
        ////////////////////////////////////////////////////////////////////////////////////
        ////////////////////////////////////////////////
393     //To compensate for the non-linear behaviour of the stepper motors the folowing
        calculations are needed to get a linear speed behaviour.
394     if (pid_output_left > 0)pid_output_left = 405 - (1 / (pid_output_left + 9)) * 5500;
395     else if (pid_output_left < 0)pid_output_left = -405 - (1 / (pid_output_left - 9))
        * 5500;
396
397     if (pid_output_right > 0)pid_output_right = 405 - (1 / (pid_output_right + 9)) *
        5500;
398     else if (pid_output_right < 0)pid_output_right = -405 - (1 / (pid_output_right -
        9)) * 5500;
```

```
399
400     //Calculate the needed pulse time for the left and right stepper motor controllers
401     if (pid_output_left > 0)left_motor = 400 - pid_output_left;
402     else if (pid_output_left < 0)left_motor = -400 - pid_output_left;
403     else left_motor = 0;
404
405     if (pid_output_right > 0)right_motor = 400 - pid_output_right;
406     else if (pid_output_right < 0)right_motor = -400 - pid_output_right;
407     else right_motor = 0;
408
409     //Copy the pulse time to the throttle variables so the interrupt subroutine can
        use them
410     throttle_left_motor = left_motor;
411     throttle_right_motor = right_motor;
412   }
413
414   void kalibrasimpu() {
415     Wire.beginTransmission(gyro_address);
416     Wire.write(0x6B);
417     Wire.write(0x00);
418     Wire.endTransmission();
419
420     Wire.beginTransmission(gyro_address);
421     Wire.write(0x1B);
422     Wire.write(0x00);
423     Wire.endTransmission();
424
425     Wire.beginTransmission(gyro_address);
426     Wire.write(0x1C);
427     Wire.write(0x08);
428     Wire.endTransmission();
429
430     Wire.beginTransmission(gyro_address);
431     Wire.write(0x1A);
432     Wire.write(0x03);
433     Wire.endTransmission();
434     for (int receive_counter = 0; receive_counter < 1100; receive_counter++) {
435       Wire.beginTransmission(gyro_address);
436       Wire.write(0x43);
437       Wire.endTransmission();
438       Wire.requestFrom(gyro_address, 4);
439       gyro_yaw_calibration_value += Wire.read() << 8 | Wire.read();
440       gyro_pitch_calibration_value += Wire.read() << 8 | Wire.read();
441       delayMicroseconds(3700);
442     }
443     gyro_pitch_calibration_value /= 500;
444     gyro_yaw_calibration_value /= 500;
445
446     PORTC |= 0b00001000;
447     delay(50);
448     PORTC &= 0b11110111;
449     PORTB &= 0b11111110;
450     loop_timer = micros() + 4000;
451   }
452
453   void enc() {
454     enckir = (PINC >> PC0 & B00000001 >> PC0);
455     enckan = (PINC >> PC2 & B00000100 >> PC2);
456     //kiri maju
457     if ((enckir == 1) && (statekir == 0) && (flagdirkir == 1)) {
458       countkir++;
459       statekir = 1;
460     }
461     if ((enckir == 0) && (statekir == 1) && (flagdirkir == 1)) {
462       countkir++;
463       statekir = 0;
464     }
465     //kiri mundur
466     if ((enckir == 1) && (statekir == 0) && (flagdirkir == 0)) {
```

```
467       countkir--;
468       statekir = 1;
469     }
470     if ((enckir == 0) && (statekir == 1) && (flagdirkir == 0)) {
471       countkir--;
472       statekir = 0;
473     }
474   }
475
476   void resetEnc() {
477     while (1) {
478       PORTB &= 0b11111110;
479       enc();
480       PORTD |= 0b00010000;
481       delayMicroseconds(1200);
482       PORTD &= 0b11101111;
483       delayMicroseconds(1200);
484       if (enckir == 0) {
485         countkir = 0;
486         break;
487       }
488     }
489     while (1) {
490       PORTB &= 0b11111110;
491       enc();
492       PORTD |= 0b10000000;
493       delayMicroseconds(1200);
494       PORTD &= 0b01111111;
495       delayMicroseconds(1200);
496       if (enckan == 0) {
497         countkan = 0;
498         break;
499       }
500     }
501   }
502
503   void pidposisi() {
504
505     epos = setposisi + countkir; //setPoint=jarak yg d inginkan dan jarkir adalah
      jarak yg d baca sensor skrng
506     pid_setpoint = (kpp * epos) + (kip * (epos + lastepos)) + (kdp * (epos -
      lastepos)); //rumus pid
507     lastepos = epos;  // mendaptkan error sebelumnya
508     if (pid_setpoint >= 3)pid_setpoint = 3;
509     if (pid_setpoint <= -3)pid_setpoint = -3;
510   }
511
512   void mpu() {
513     Wire.beginTransmission(gyro_address);
514     Wire.write(0x3F);
515     Wire.endTransmission();
516     Wire.requestFrom(gyro_address, 2);
517     accelerometer_data_raw = Wire.read() << 8 | Wire.read();
518     accelerometer_data_raw += acc_calibration_value;
519     if (accelerometer_data_raw > 8200)accelerometer_data_raw = 8200;
520     if (accelerometer_data_raw < -8200)accelerometer_data_raw = -8200;
521
522     angle_acc = asin((float)accelerometer_data_raw / 8200.0) * 57.296;
523
524     Wire.beginTransmission(gyro_address);
525     Wire.write(0x43);
526     Wire.endTransmission();
527     Wire.requestFrom(gyro_address, 4);
528     gyro_yaw_data_raw = Wire.read() << 8 | Wire.read();
529     gyro_pitch_data_raw = Wire.read() << 8 | Wire.read();
530
531     gyro_pitch_data_raw -= gyro_pitch_calibration_value;
532     angle_gyro += gyro_pitch_data_raw * 0.000031;
533
```

```
534
535     gyro_yaw_data_raw -= gyro_yaw_calibration_value;
536
537     angle_gyro = angle_gyro * 0.996 + angle_acc * 0.004;
538   }
539
```