LAMPIRAN

LAMPIRAN 1. Program Robot

1, Mazesolving

```
#include "hardware_config.h"
#include "button_and_motor.h"
#include "variable.h"
#include "sensor.h"
#include "panel.h"
#include "plan.h"
#include "running.h"

Ultrasonic ultrasonic(20,21);

void setup() {
  init_button();
  init_sensor();
  pinMode(pin_LCDRW, OUTPUT);
  pinMode(pin_LCDLED, OUTPUT);
  digitalWrite(pin_LCDRW, LOW);
  digitalWrite(pin_LCDLED, HIGH);
  pinMode(pin_OUTPUT_EXT, OUTPUT);
  digitalWrite(pin_OUTPUT_EXT, HIGH);
  lcd.begin(16, 2);
  lcd.clear();
  delay(100);
  lcd.setCursor(0, 0);
  lcd.print("Ryan Saputra");
  lcd.setCursor(0, 1);
  lcd.print("UMU");
  delay(1000);
  lcd.clear();

  EEPROM.get(0, ee);

  reset_default();
  plan_input();

}

void loop() {
  displayHomeScreen();
  if (!button_START) {
    check_XOR();
    Running();
  }
  if (!button_MENU) {
    calibrate_sensor();
  }
}

void Running() {
  lcd.clear();
  lcd.print("GOOOO....");
  delay(200);
  lcd.clear();
  digitalWrite(pin_LCDLED, LOW);
  index = ee.path.cp[ee.path.index_cp];
  temp_timer = 0;
  tick = 0;

  setMotor(speed_delay_start, speed_delay_start);
  delay(delay_start);

  while (1) {
    sensor_running = readSensor();
    if (temp_timer > 0) {
      lcd_backlightOn;
      tick++;
      if (tick > 4) {
        temp_timer--;
        tick = 0;
      }
      speed = ee.path.SA[ee.setting.plan][index];
    }
    else {
      lcd_backlightOff;
      temp_timer = ee.path.TA[ee.setting.plan][index];
      ee.path.TA[ee.setting.plan][index] = 0;
      tick = 0;
      speed = ee.setting.speed;

      if (index == stop_index) break;
      solve_path();
    }

    lcd.setCursor(0, 1);
    lcd.print(index);
    follow_line();
    if ( ultrasonic.Ranging(CM)<=20) {
      setMotor(0,0);
      delay(2000);
    }
    if (!button_START) {
      break;
    }
  }
  setMotor(0, 0);
  digitalWrite(pin_LCDLED, HIGH);
  lcd.clear();
  lcd.print("Completed....");
  delay(2000);
  lcd.clear();
  EEPROM.get(0, ee);
  reset_default();
  plan_input();
}
```

2. Button and Motor

```
#define button_UPL digitalRead(pin_button_UPL)
#define button_DOWNL digitalRead(pin_button_DOWNL)
#define button_UPR digitalRead(pin_button_UPR)
#define button_DOWNR digitalRead(pin_button_DOWNR)
#define button_START digitalRead(pin_button_START)
#define button_MENU digitalRead(pin_button_MENU)

void init_button() {
  pinMode(pin_button_UPL, INPUT_PULLUP);
  pinMode(pin_button_DOWNL, INPUT_PULLUP);
  pinMode(pin_button_UPR, INPUT_PULLUP);
  pinMode(pin_button_DOWNR, INPUT_PULLUP);
  pinMode(pin_button_START, INPUT_PULLUP);
  pinMode(pin_button_MENU, INPUT_PULLUP);
}

void setMotor(int L, int R) {
  if (L > 0) {
    digitalWrite(pin_MOTOR_DIRL, LOW);
  } else {
    digitalWrite(pin_MOTOR_DIRL, HIGH);
    L = 255 + L;
  }
  analogWrite(pin_MOTOR_PWML, L);
  if (R > 0) {
    digitalWrite(pin_MOTOR_DIRR, LOW);
  } else {
    digitalWrite(pin_MOTOR_DIRR, HIGH);
    R = 255 + R;
  }
  analogWrite(pin_MOTOR_PWMR, R);
}

void setMotorT(int L, int R) {
  if (L < 0) {
    L = 255 + L;
    analogWrite(PWM_naik, L);
    digitalWrite(dir_naik, 1);
  } else {
    analogWrite(PWM_naik, L);
    digitalWrite(dir_naik, 0);
  }
  if (R < 0) {
    R = 255 + R;
    analogWrite(PWM_maju, R);
    digitalWrite(dir_maju, 1);
  } else {
    analogWrite(PWM_maju, R);
    digitalWrite(dir_maju, 0);
  }
}
```

3. Hardware Config

```
//const int pin_ADC_SENSOR[14] = {A0, A1, A2, A3, A4, A5, A6, A6, A5, A4,
A3, A2, A1, A0};

// config for ichibot ultimate arduino
#define PWM_naik              45
#define dir_naik              47
#define PWM_maju              46
#define dir_maju              48

#define pin_button_UPL        A8
#define pin_button_DOWNL       8
#define pin_button_UPR        11
#define pin_button_DOWNR      12
#define pin_button_START      10
#define pin_button_MENU        9

#define pin_LCDRS             34
#define pin_LCDRW             111
#define pin_LCDE              32
#define pin_LCDD4             30
#define pin_LCDD5             28
#define pin_LCDD6             26
#define pin_LCDD7             24
#define pin_LCDLED            22

#define lcd_backlightOn    digitalWrite(pin_LCDLED, HIGH)
#define lcd_backlightOff   digitalWrite(pin_LCDLED, LOW)
#define pin_MOTOR_DIRL           7
#define pin_MOTOR_PWML           6
#define pin_MOTOR_DIRR           5
#define pin_MOTOR_PWMR           4

#define pin_ENABLE_SENSORL  A1
#define pin_ENABLE_SENSORR  A0

#define pin_OUTPUT_EXT     15

#define max_sensor        12

const int pin_ADC_SENSOR[max_sensor] = {A2, A3, A4, A5, A6, A7, A7, A6,
A5, A4, A3, A2};
```

```
#include <Ultrasonic.h>
#include <LiquidCrystal.h>;
LiquidCrystal lcd(pin_LCDRS, pin_LCDE, pin_LCDD4, pin_LCDD5,
pin_LCDD6, pin_LCDD7);
```

4. Panel

```
void displayHomeScreen() {
  int i;
  char buff[16];
  readSensor();
  lcd.setCursor(11, 1);
  sprintf(buff, "V:%3d", ee.setting.speed);
  lcd.print(buff);
  lcd.setCursor(0, 1);
  sprintf(buff, "CP%d", ee.path.index_cp);
  lcd.print(buff);

  if (!button_UPL) {
    if (++ee.path.index_cp >= 6)ee.path.index_cp = 0;
    EEPROM.put(0, ee);
    delay(200);
  }
  if (!button_DOWNL) {
    if (--ee.path.index_cp < 0)ee.path.index_cp = 5;
    EEPROM.put(0, ee);
    delay(200);
  }

  if (!button_UPR) {
    if (++ee.setting.speed > 255) ee.setting.speed = 0;
    EEPROM.put(0, ee);
    delay(200);
  }
  if (!button_DOWNR) {
    if (--ee.setting.speed < 0) ee.setting.speed = 255;
    EEPROM.put(0, ee);
    delay(200);
  }
  lcd.setCursor(0, 1);
}
```

5. Plan Set

```
#define lft   left
#define fwd   forward
#define rgt   right
#define on    action_on
#define off   action_off

//plan_set (plan, index, action, mode sensor, brake, delay, pwm L, pwm R, SA,
TA)

int speed_delay_start = 80;
int delay_start = 200;

/*Mode Sensor
  pertigaan T/perempatan        mode 3     XOR 111000000111
                                mode 4     XOR 110000000011
  pertigaan Kiri                mode 5     XOR 110011110000
                                mode 7     XOR 100011110000
  pertigaan Kanan               mode 6     XOR 000011110011
                                mode 8     XOR 000011110001
  SISI KIRI => untuk mendeteksi tikungan di kiri, misal siku
                                mode 11     OR  100000000000
                                mode 13     OR  110000000000
  SISI KANAN => untuk mendeteksi tikungan di kanan, misal siku
                               mode 12     OR  000000000001
                               mode 14     OR  000000000011
  anywhere                     mode 17     OR  111111111111
  kosong                       mode 18     =   000000000000
*/
//plan_set (plan, index, action, mode sensor, brake, delay, pwm L, pwm R, SA,
TA)

int stop_index = 5;

void plan_input() {
  //setting kecepatan default robot
  ee.setting.speed = 170;

  //PID
  kp = 7;
  kd = 55;

  //setting checkpoint
  ee.path.cp[0] = 0; ee.path.cp[1] = 10; ee.path.cp[2] = 20; ee.path.cp[3] = 30;
ee.path.cp[4] = 40; ee.path.cp[5] = 50;

  //setting plan
  plan_set(1, 0, fwd, 0, 0, 0, 0, 0, 80, 5); //start
  plan_set(1, 1, on, 3, 50, 50, 0, 0, 0, 10); //muat
  plan_set(1, 2, fwd, 17, 10, 100, 0, 0, 150, 15); // maju setelah muat
  plan_set(1, 3, off, 3, 50, 50, 0, 0, 0, 10); //bongkar
  plan_set(1, 4, fwd, 17, 10, 100, 0, 0, 150, 20); // maju setelah bongkar
  plan_set(1, 5, fwd, 3, 0, 0, 0, 0, 80, 5);//maju finish
}
```

6. Planning

```
void delay_action(int pwmLeft, int pwmRight) {
  setMotor(-speed, -speed);
  delay(ee.path.B[ee.setting.plan][index]);
  setMotor(pwmLeft, pwmRight);
  delay(ee.path.D[ee.setting.plan][index]);
}

void turn(int dir, int pwmLeft, int pwmRight) {
  unsigned int sensor_value = 0;
  delay_action(pwmLeft, pwmRight);
  if (dir == left) {
    do {
      setMotor(pwmLeft * 0.3, pwmRight * 0.3);
      sensor_value = readSensor();
    }
    while (!(sensor_value & 0b011111111000));
  }
  else if (dir == right) {
    do {
      setMotor(pwmLeft * 0.3, pwmRight * 0.3);
      sensor_value = readSensor();
    }
    while (!(sensor_value & 0b000111111110));
  }
}

void get_junction() {
  temp_timer = 0;
  lcd_backlightOn;
  if (index < max_index) {
    index++;
    if (ee.path.action[ee.setting.plan][index] == forward ||
ee.path.action[ee.setting.plan][index] == left ||
ee.path.action[ee.setting.plan][index] == right) {
      turn(forward, ee.path.F[ee.setting.plan][index],
ee.path.R[ee.setting.plan][index]);
      lcd.setCursor(5, 1);
      lcd.print("ACT");
    }
    else if (ee.path.action[ee.setting.plan][index] == action_on) {
      delay_action(ee.path.F[ee.setting.plan][index],
ee.path.R[ee.setting.plan][index]);
```

```
      setMotorT(250, 0);
      delay(4000);
      setMotorT(0, 100);
      delay(1500);
      lcd.setCursor(5, 1);
      lcd.print("ON");
    }
    else if (ee.path.action[ee.setting.plan][index] == action_off) {
      delay_action(ee.path.F[ee.setting.plan][index],
ee.path.R[ee.setting.plan][index]);
      setMotorT(0, -100);
      delay(1500);
      setMotorT(-250, 0);
      delay(3800);
      setMotorT(0, 0);
      delay(0);
      lcd.setCursor(5, 1);
      lcd.print("OFF");
    }
  }
  temp_timer = ee.path.TA[ee.setting.plan][index];
}

void solve_path() {
  if (ee.setting.count_mode[ee.path.mode_sensor[ee.setting.plan][index + 1]] ==
OR) {
    if (sensor_running &
ee.sensor.counter[ee.path.mode_sensor[ee.setting.plan][index + 1]]) {
      get_junction();
    }
  }
  else if (ee.setting.count_mode[ee.path.mode_sensor[ee.setting.plan][index + 1]]
== sama_dengan) {
    if (sensor_running ==
ee.sensor.counter[ee.path.mode_sensor[ee.setting.plan][index + 1]]) {
      get_junction();
    }
  }
  else if (ee.setting.count_mode[ee.path.mode_sensor[ee.setting.plan][index + 1]]
== XOR) {
    if (sensor_running &
sensor_counterXOR[ee.path.mode_sensor[ee.setting.plan][index + 1]]) {
      if (sensor_running &
sensor_counterXOR1[ee.path.mode_sensor[ee.setting.plan][index + 1]]) {
        get_junction();
      }
    }
  }
}

void follow_line() {
  int sensor = sensor_running;
  switch (sensor) {
    case 0b000000000001: error = -18; break;
    case 0b000000000011: error = -14; break;
    case 0b000000000010: error = -11; break;
    case 0b000000000110: error = -9; break;
    case 0b000000000100: error = -7; break;
    case 0b000000001100: error = -6; break;
    case 0b000000001000: error = -5; break;
    case 0b000000011000: error = -4; break;
    case 0b000000010000: error = -3; break;
    case 0b000000110000: error = -2; break;
    case 0b000000100000: error = -1; break;
    case 0b000001100000: error = 0; break;
    case 0b000001000000: error = 1; break;
    case 0b000011000000: error = 2; break;
    case 0b000010000000: error = 3; break;
    case 0b000110000000: error = 4; break;
    case 0b000100000000: error = 5; break;
    case 0b001100000000: error = 6; break;
    case 0b001000000000: error = 7; break;
    case 0b011000000000: error = 9; break;
    case 0b010000000000: error = 11; break;
    case 0b110000000000: error = 14; break;
    case 0b100000000000: error = 18; break;

    case 0b000000000111: error = -11; break;
    case 0b000000001111: error = -9; break;
    case 0b000000001110: error = -7; break;
    case 0b000000011110: error = -6; break;
    case 0b000000011100: error = -5; break;
    case 0b000000111100: error = -4; break;
    case 0b000000111000: error = -3; break;
    case 0b000001111000: error = -2; break;
    case 0b000001110000: error = -1; break;

    case 0b000011110000: error = 0; break;

    case 0b000011100000: error = 1; break;
    case 0b000111100000: error = 2; break;
    case 0b000111000000: error = 3; break;
    case 0b001111000000: error = 4; break;
    case 0b001110000000: error = 5; break;
    case 0b011110000000: error = 6; break;
    case 0b011100000000: error = 7; break;
    case 0b111100000000: error = 9; break;
    case 0b111000000000: error = 11; break;
   }

   int rateError = error - lastError;
   lastError = error;
   int moveVal = (int) (error * kp) + (rateError * kd);
   int moveLeft = speed - moveVal;
   int moveRight = speed + moveVal;
   int minSpeed = -150;
   int maxSpeed = 230;
   moveLeft = constrain(moveLeft, minSpeed, maxSpeed);
   moveRight = constrain(moveRight, minSpeed, maxSpeed);
   setMotor(moveLeft, moveRight);
```

7. Sensor

```
void enableSensor(int L, int R) {
digitalWrite(pin_ENABLE_SENSORL, L);
digitalWrite(pin_ENABLE_SENSORR, R);
delayMicroseconds(50);
}

void init_sensor() {
int i = 0;
for (i = 0; i < max_sensor; i++) {
pinMode(pin_ADC_SENSOR[i], INPUT);
}
pinMode(pin_ENABLE_SENSORL, OUTPUT);
pinMode(pin_ENABLE_SENSORR, OUTPUT);
enableSensor(0, 0);
}

int readSensor() {
int bitSensor = 0;
int i;
int xpos[12] = {2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13};
enableSensor(1, 0);
for (i = 0; i < 6; i++) {
if (analogRead(pin_ADC_SENSOR[i]) > ee.setting.limit_value_sensor[i]) {
bitSensor = bitSensor | (0b100000000000 >> i);
lcd.setCursor(xpos[i], 0);
lcd.write(255);
}
else {
lcd.setCursor(xpos[i], 0);
lcd.print("_");
}
}
delayMicroseconds(150);
enableSensor(0, 1);
for (i = 6; i < 12; i++) {
if (analogRead(pin_ADC_SENSOR[i]) > ee.setting.limit_value_sensor[i]) {
bitSensor = bitSensor | ( 0b100000000000 >> i);
lcd.setCursor(xpos[i], 0);
lcd.write(255);
}
else {
lcd.setCursor(xpos[i], 0);
lcd.print("_");
}
}
delayMicroseconds(150);
enableSensor(0, 0);

//invert
//bitSensor &= 0b111111111111;
//bitSensor = 0b111111111111 - bitSensor;

return bitSensor;
}

int calibrate_sensor() {
int i, valSensor, xCursor = 0;
int minVal[max_sensor], maxVal[max_sensor];
delay(300);
lcd.clear();
for (i = 0; i < max_sensor; i++) {
minVal[i] = 1023;
maxVal[i] = 0;
}
while (1) {
enableSensor(1, 0);
for (i = 0; i < 6; i++) {
valSensor = analogRead(pin_ADC_SENSOR[i]);
if (valSensor > maxVal[i]) {
maxVal[i]  = valSensor;
}
if (valSensor < minVal[i]) {
minVal[i]  = valSensor;
}
}
delay(1);
enableSensor(0, 1);
for (i = 6; i < 12; i++) {
valSensor = analogRead(pin_ADC_SENSOR[i]);
if (valSensor > maxVal[i]) {
maxVal[i]  = valSensor;
}
if (valSensor < minVal[i]) {
minVal[i]  = valSensor;
}
```

```
}
delay(1);
enableSensor(0, 0);
if (!button_MENU) {
break;
}
lcd.setCursor(0, 0);
lcd.print("Scanning Sensor");

if (millis() % 25 == 0) {
lcd.setCursor(xCursor, 1);
lcd.write(0xff);
if (++xCursor > 15) {
xCursor = 0;
lcd.clear();
}
}
}
for (i = 0; i < max_sensor; i++) {
ee.setting.limit_value_sensor[i] = ((maxVal[i] - minVal[i]) * 25 / 100) +
minVal[i];
}
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Saving...");
EEPROM.put(0, ee);
delay(500);
lcd.clear();
}
void check_XOR() {
unsigned int i, x;
for (i = 0; i < max_modeSensor; i++) {
unsigned int temp_sensor = ee.sensor.counter[i];
sensor_counterXOR[i] = 0;
sensor_counterXOR1[i] = 0;
x = 0;
do {
data_bit = (temp_sensor & (1 << x));
sensor_counterXOR[i] |= data_bit;
x++;
}
while (data_bit == 0 && x < max_sensor);
do {
```

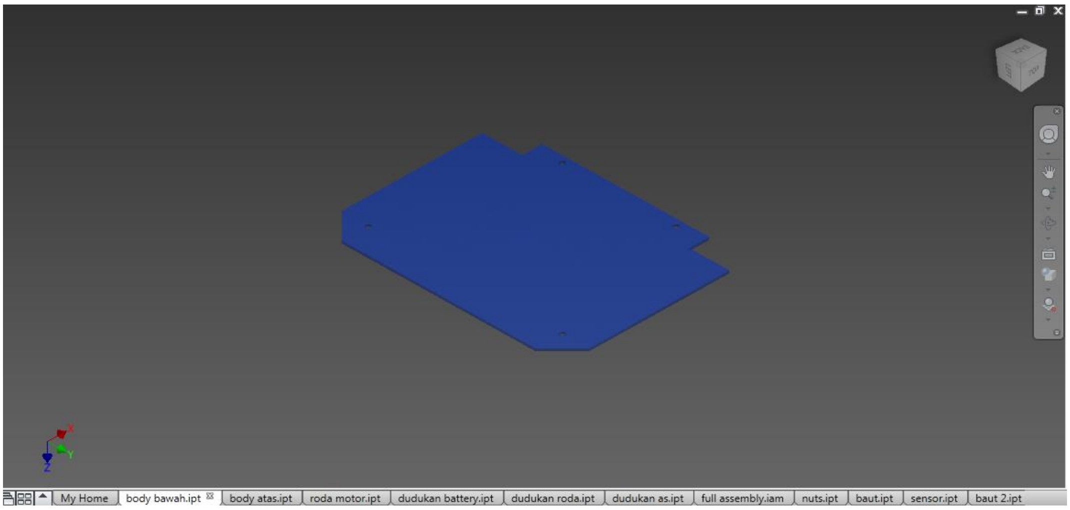
```
data_bit = (temp_sensor & (1 << x));
sensor_counterXOR[i] |= data_bit;
x++;
}
while (data_bit > 0 && x < max_sensor);
do {
data_bit = (temp_sensor & (1 << x));
sensor_counterXOR1[i] |= data_bit;
x++;
}
while (x < max_sensor);
```

8. Variable

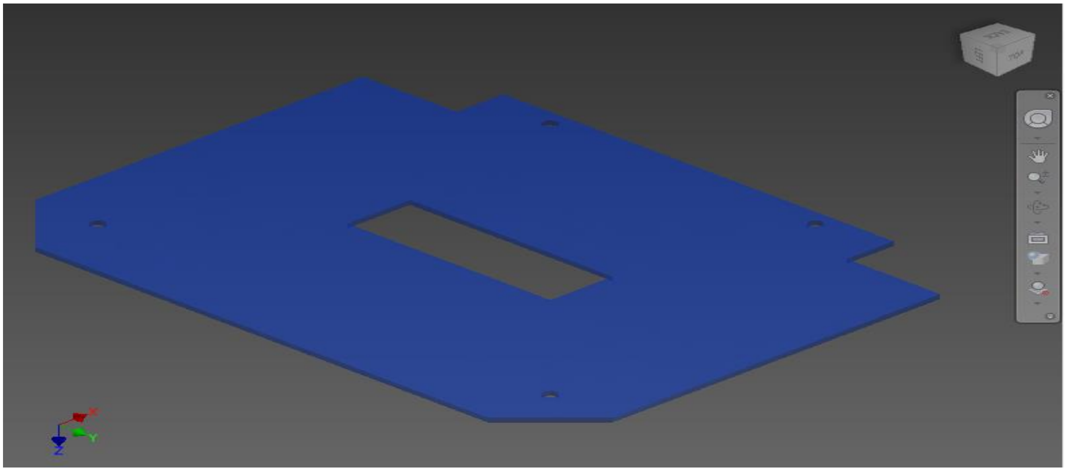
```
#include <EEPROM.h>

#define max_plan        2
#define max_index       99
#define max_modeSensor  25

enum {forward, left, right, action_on, action_off};
enum {on, off};
enum {OR, sama_dengan, XOR};
int index;
int temp_timer = 0;
int tick = 0;
int error = 0;

int lastError = 0;
int kp;
int kd;
int speed;
unsigned long sensor_running;

unsigned int data_bit;
unsigned int sensor_counterXOR[max_modeSensor + 1];
unsigned int sensor_counterXOR1[max_modeSensor + 1];
int temp_limit_value_sensor[max_sensor];

struct EE_path {
  int action[max_plan][max_index];
  int mode_sensor[max_plan][max_index];
  int B[max_plan][max_index];
  int D[max_plan][max_index];
  int F[max_plan][max_index];
  int R[max_plan][max_index];
  int SA[max_plan][max_index];
  int TA[max_plan][max_index];
  int cp[6];
  int index_cp;
};

struct EE_sensor {
  unsigned int counter[max_modeSensor + 1];
};
```

```
struct EE_setting {
  int speed;
  int limit_value_sensor[max_sensor];
  int count_mode[max_modeSensor + 1];
  int plan;
};

struct eeprom_data {
  EE_path path;
  EE_setting setting;
  EE_sensor sensor;
};
eeprom_data ee;
void plan_set(int Plan, int Index, int action, int mode_sensor, int brake, int delay,
int forward, int reverse, int speed_a, int timer_a) {
  ee.path.action[Plan - 1][Index] = action;
  ee.path.mode_sensor[Plan - 1][Index] = mode_sensor;
  ee.path.B[Plan - 1][Index] = brake;
  ee.path.D[Plan - 1][Index] = delay;
  ee.path.F[Plan - 1][Index] = forward;
  ee.path.R[Plan - 1][Index] = reverse;
  ee.path.SA[Plan - 1][Index] = speed_a;
  ee.path.TA[Plan - 1][Index] = timer_a;
}

void reset_default() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Init EEPROM data");
  ee.setting.plan = 0;

  ee.setting.count_mode[0]  =  sama_dengan;
  ee.sensor.counter[0] =  0b001100001100;
  ee.setting.count_mode[1]  =  XOR;
  ee.sensor.counter[1] =  0b111110011111;
  ee.setting.count_mode[2] =  XOR;
  ee.sensor.counter[2] =  0b111100001111;
  ee.setting.count_mode[3] =  XOR;
  ee.sensor.counter[3] =  0b111000000111;
  ee.setting.count_mode[4] =  XOR;
  ee.sensor.counter[4] =  0b110000000011;
  ee.setting.count_mode[5] =  XOR;
  ee.sensor.counter[5] =  0b110011110000;
  ee.setting.count_mode[6] =  XOR;
  ee.sensor.counter[6] =  0b000011110011;
  ee.setting.count_mode[7] =  XOR;
  ee.sensor.counter[7] =  0b100011110000;
  ee.setting.count_mode[8] =  XOR;
  ee.sensor.counter[8] =  0b000011110001;
  ee.setting.count_mode[9] =  XOR;
  ee.sensor.counter[9] =  0b111001111000;
  ee.setting.count_mode[10] =  XOR;
  ee.sensor.counter[10] =  0b000111100111;
  ee.setting.count_mode[11] =  OR;
  ee.sensor.counter[11] =  0b100000000000;
  ee.setting.count_mode[12] =  OR;
  ee.sensor.counter[12] =  0b000000000001;
  ee.setting.count_mode[13] =  OR;
  ee.sensor.counter[13] =  0b110000000000;
  ee.setting.count_mode[14] =  OR;
  ee.sensor.counter[14] =  0b000000000011;
  ee.setting.count_mode[15] =  OR;
  ee.sensor.counter[15] =  0b111000000000;
  ee.setting.count_mode[16] =  OR;
  ee.sensor.counter[16] =  0b000000000111;
  ee.setting.count_mode[17] =  OR;
  ee.sensor.counter[17] =  0b111111111111;
  ee.setting.count_mode[18] =  sama_dengan;
  ee.sensor.counter[18] =  0b000000000000;
  EEPROM.put(0, ee);
  lcd.setCursor(0, 1);
  lcd.print("Completed");
  delay(1000);
  lcd.clear();
}
```

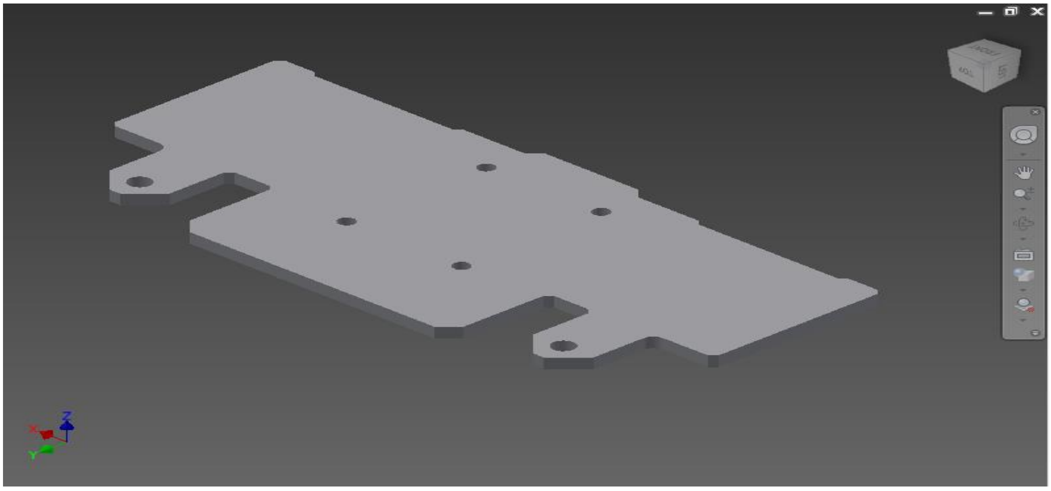
LAMPIRAN 2. DESAIN



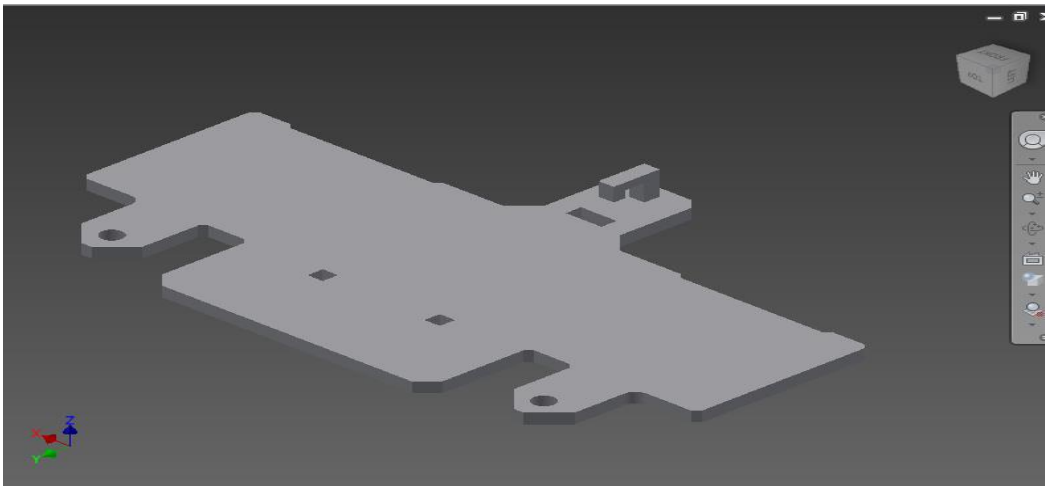
Gambar 1. Desain Bodi Robot Bagian Bawah.



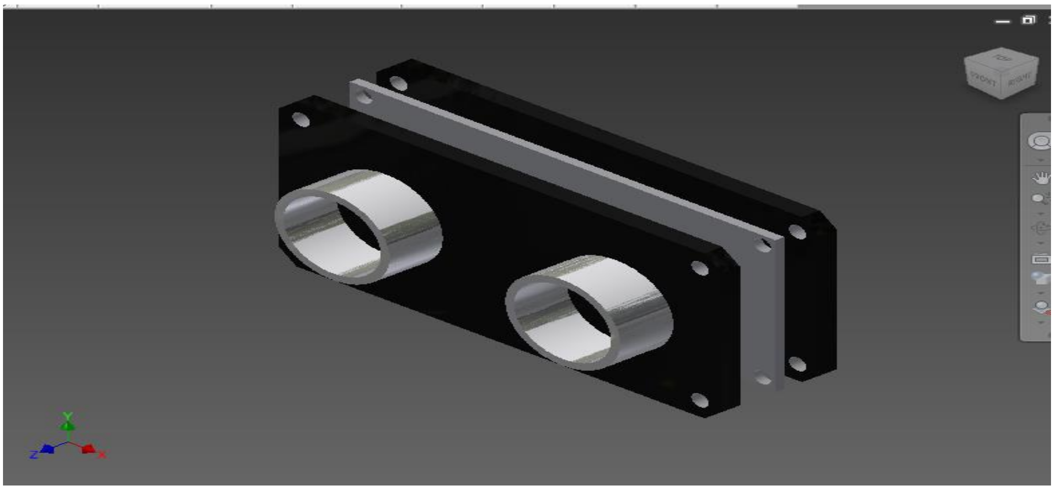
Gambar 2. Desain Bodi Robot Bagian Atas.



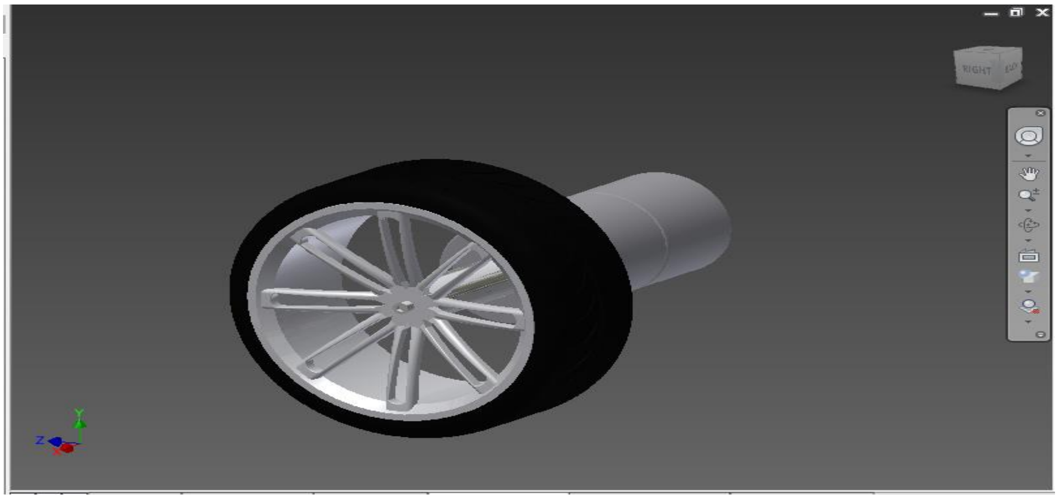
Gambar 3. Desain Dudukan Baterai.



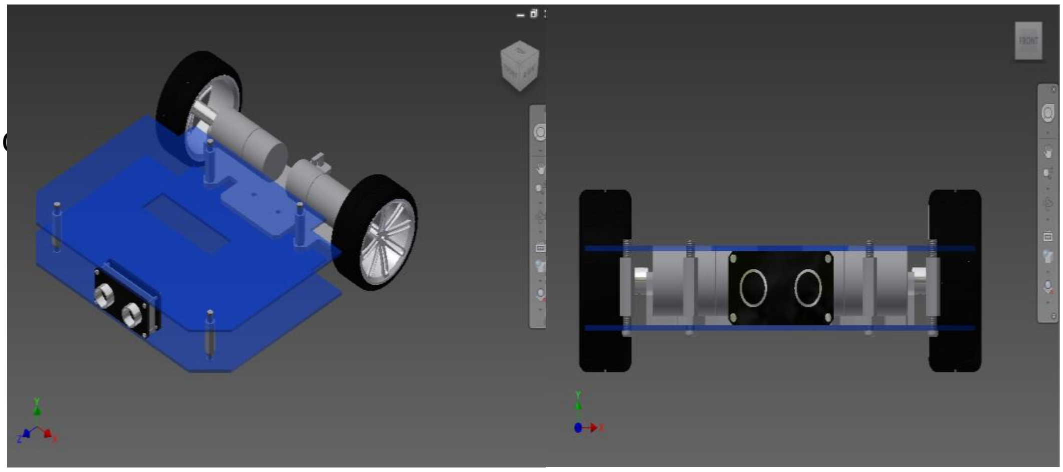
Gambar 4. Desain Dudukan Motor dan Motor *Driver*.



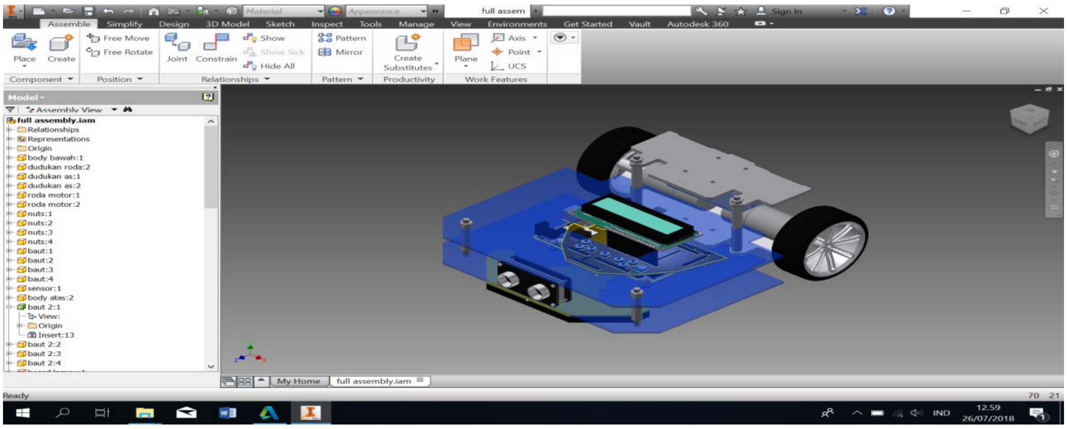
Gambar 5. Desain Dudukan Sensor Ultrasonik HC SR-04.



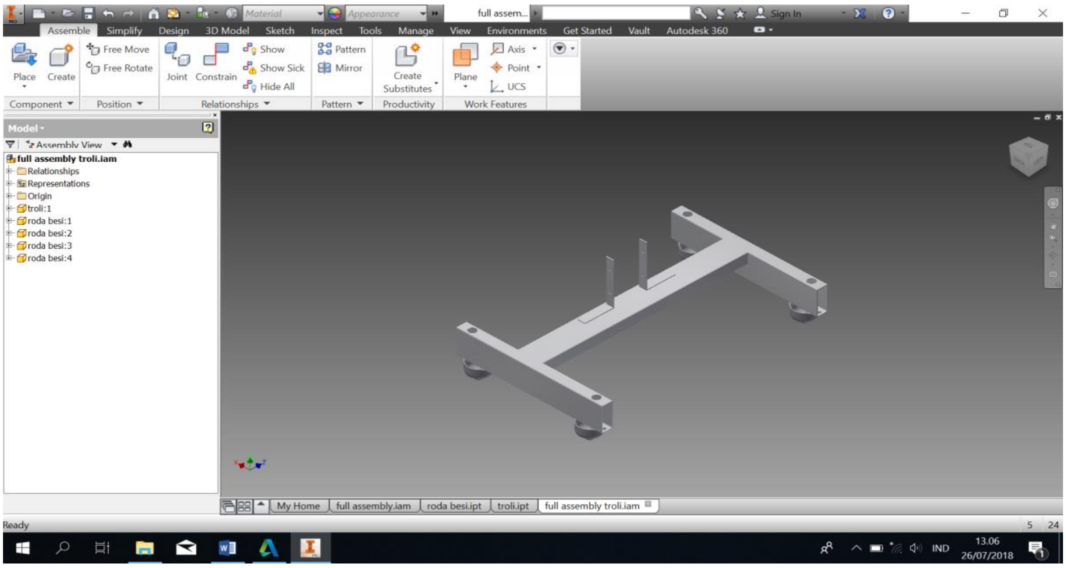
Gambar 6. Desain Motor DC dan Roda.



Gambar 8. *Assembly* Komponen.



Gambar 9. Hasil *Assembly*.



Gambar 9. Desain Bagian *Trolley*.