LAMPIRAN

Lampiran 1. Program Arduino

*main

```
#include "hardware_config.h"
#include "sensor.h"
#include "menu.h"
#include "plan.h"
#include "running.h"

Ultrasonic ultrasonic(49,51); //Trig,Echo -- Deteksi Benda
Ultrasonic ultrasonic1(43,45); //Trig,Echo -- Deteksi Halangan

void setup() {
 init_button();
 init_sensor();
 lcd.begin(16, 2);
 lcd.clear();
 delay(10);
 lcd.setCursor(0, 0);
 lcd.print("   AGV LI-FO   ");
 lcd.setCursor(0, 1);
 lcd.print(" LUDHI PRASETYO ");
 delay(1500);
 lcd.clear();
 EEPROM.get(0, ee);
 plan_input();
 init_CPandSENSOR();
}

void loop() {
 displayHomeScreen();
```

```
 if(!button_MENU) {
   ee.setting.plan = 0;
   ayo();
 }
 if(!button_START) {
   ee.setting.plan = 1;
   ayo();
 }
 if (!button_UPL) {
   index_mainmenu = 0;
   delay(200);
   while (1) {
     main_menu();
     if (!button_START) {
       lcd.clear();
       delay(200);
       break;
     }
   }
 }
}

void ayo(){
 ulang:
 ulang2:
 if (ultrasonic.read(CM)<=10) {
   check_XOR();
   Running();
   goto ulang2;
   } else {
     goto ulang;
```

```
    }
}

void Running() {
 lcd.clear();
 lcd.print("GO...");
 delay(200);
 lcd.clear();
 ee.setting.cp[0] = 0;     //cp 0
 ee.setting.cp[1] = cp1;   //cp 1
 ee.setting.cp[2] = cp2;   //cp 2
 ee.setting.cp[3] = cp3;   //cp 3
 ee.setting.cp[4] = cp4;   //cp 4
 EEPROM.put(0, ee);
 index = ee.setting.cp[ee.setting.checkpoint];
 temp_timer = 0;

 while (1) {
  sensor_running = readSensor();
  if (temp_timer > 0) {
   temp_timer--;
   speed = ee.path.SA[ee.setting.plan][index];
  } else {
   speed = ee.setting.speed;
   if (index == ee.setting.stop_index) break;
   solve_path();
  }

  follow_line();
  if (ultrasonic1.read(CM)<=24) {
   setMotor(0,0);
```

```
      delay(2000);
    }
    if (!button_START) {
      break;
    }
    lcd.setCursor(0, 1);
    sprintf(buff, "%2d", index);
    lcd.print(buff);
  }
  setMotor(0,0);
  lcd.clear();
  lcd.print("Finish...");
  delay(1000);
  lcd.clear();
  EEPROM.get(0, ee);
  //2void
  plan_input();
}

void init_CPandSENSOR() {
  ee.setting.cp[0] = 0;
  ee.setting.cp[1] = cp1;
  ee.setting.cp[2] = cp2;
  ee.setting.cp[3] = cp3;
  ee.setting.cp[4] = cp4;

  ee.setting.count_mode[1]  =  XOR;
  ee.sensor.counter[1] =  s_lancip_kanan;
  ee.setting.count_mode[2] =  XOR;
  ee.sensor.counter[2] =  s_lancip_kiri;
  ee.setting.count_mode[3] =  XOR;
```

```
  ee.sensor.counter[3] =  s_perempatan;
  ee.setting.count_mode[4] =  XOR;
  ee.sensor.counter[4] =  s_pertigaan_kanan;
  ee.setting.count_mode[5] =  XOR;
  ee.sensor.counter[5] =  s_pertigaan_kiri;
  ee.setting.count_mode[6] =  sama_dengan;
  ee.sensor.counter[6] =  s_blok_hitam;
  ee.setting.count_mode[7] =  sama_dengan;
  ee.sensor.counter[7] =  s_blok_putih;
  ee.setting.count_mode[8] =  sama_dengan;
  ee.sensor.counter[8] =  s_semua_sensor;
  ee.setting.count_mode[8] =  OR;
}

*hardware_config.h
#include <EEPROM.h>
#include <Ultrasonic.h>
#define max_plan        2
#define max_index       50
#define max_modeSensor  25
#define max_sensor      12

/*--- Variable Menu ---*/
int index_mainmenu = 0;

/*--- Variable General ---*/
enum {forward, left, right, action_up, action_down, action_off};
enum {on, off};
enum {OR, sama_dengan, XOR};
int index = 0;
int temp_timer = 0;
```

```
int tick = 0;
int error = 0;
int lastError = 0;
int kp = 17;
int kd = 32;
int speed;
char buff[16];
unsigned int 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];

unsigned long lastTime = 0;
double lastErr = 0;

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];
};

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

```
struct EE_setting {
  unsigned int speed;
  int limit_value_sensor[max_sensor];
  int count_mode[max_modeSensor + 1];
  int plan;
  int cp[5];
  int checkpoint;
  int stop_index;
};

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 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] = 10;
  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;
}
```

```
/*----- HARDWARE CONFIG -----*/
#define pin_button_UPL      A15
#define pin_button_DOWNL    A14
#define pin_button_MENU     A13
#define pin_button_UPR      A12
#define pin_button_DOWNR    A11
#define pin_button_START    A10

#define pin_LCDRS           52
#define pin_LCDE            50
#define pin_LCDD4           48
#define pin_LCDD5           46
#define pin_LCDD6           44
#define pin_LCDD7           42

#define pin_MOTOR_DIRL      7
#define pin_MOTOR_PWML      6
#define pin_MOTOR_DIRR      5
#define pin_MOTOR_PWMR      4

#define pin_MOTOR_PWMU      2
#define pin_MOTOR_DIRU      3

#define pin_ENABLE_SENSORL  A1
#define pin_ENABLE_SENSORR  A0

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

#include <LiquidCrystal.h>;
```

```
LiquidCrystal lcd(pin_LCDRS, pin_LCDE, pin_LCDD4, pin_LCDD5,
pin_LCDD6, pin_LCDD7);

#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 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);
}


void reset_default() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Factory");
  lcd.setCursor(0, 1);
  lcd.print("Reset Data...");
  ee.setting.speed = 200;
  ee.setting.plan = 0;

  for (int a = 0; a <= 4; a++) {
    ee.setting.cp[a] = 0;
  }

  delay(1000);
  lcd.clear();
  EEPROM.put(0, ee);
  lcd.setCursor(0, 0);
  lcd.print("Completed.");
  delay(1000);
  lcd.clear();
}

*menu.h
void setMotorU(int L) {
  if (L < 0) {
    L = 255 + L;
```

```
    analogWrite(pin_MOTOR_PWMU, L);
    digitalWrite(pin_MOTOR_DIRU, 1);
  } else {
    analogWrite(pin_MOTOR_PWMU, L);
    digitalWrite(pin_MOTOR_DIRU, 0);
  }
}


void sensortest(){
 Ultrasonic ultrasonic(49,51); //Trig,Echo -- Deteksi Benda
 Ultrasonic ultrasonic1(43,45); //Trig,Echo -- Deteksi Halangan
 lcd.clear();
 lcd.setCursor(0, 0);
 lcd.print("Benda : ");
 lcd.setCursor(0, 1);
 lcd.print("Halangan : ");

 if (ultrasonic.read(CM)<=10) {
 lcd.setCursor(8, 0);
 lcd.print("Ada");
 }
 else {
 lcd.setCursor(8, 0);
 lcd.print("Tidak");
 }

 lcd.setCursor(11, 1);
 lcd.print(ultrasonic1.read(CM));
 delay(200);
```

```
}


void displayHomeScreen() {
 int i;
 readSensor();
 lcd.setCursor(11, 1);
 sprintf(buff, "V:%3d", ee.setting.speed);
 lcd.print(buff);
 lcd.setCursor(0, 1);
 lcd.print("MENU  A B");


 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);
 }
}

void main_menu() {
 if (!button_DOWNL) {
   if (++index_mainmenu > 5) index_mainmenu = 0;
   delay(200);
 }
```

```
 if (!button_UPL) {
  if (--index_mainmenu < 0) index_mainmenu = 5;
  delay(200);
 }

 switch (index_mainmenu) {
  /*
  case 0:
    lcd.setCursor(0, 0);
    sprintf(buff, ">Plan :%d       ", ee.setting.plan + 1);
    lcd.print(buff);
    lcd.setCursor(0, 1);
    sprintf(buff, " Stop Index :%2d ", ee.setting.stop_index);
    lcd.print(buff);

    if (!button_UPR) {
      if (++ee.setting.plan > 1) ee.setting.plan = 0;
      EEPROM.put(0, ee);
      delay(200);
    }
    if (!button_DOWNR) {
      if (--ee.setting.plan < 0) ee.setting.plan = 1;
      EEPROM.put(0, ee);
      delay(200);
    }
    break;

  case 1:
    lcd.setCursor(0, 0);
    sprintf(buff, " Plan :%d       ", ee.setting.plan + 1);
    lcd.print(buff);
```

```
    lcd.setCursor(0, 1);
    sprintf(buff, ">Stop Index :%2d", ee.setting.stop_index);
    lcd.print(buff);
    if (!button_UPR) {
      if (++ee.setting.stop_index > 99) ee.setting.stop_index = 0;
      EEPROM.put(0, ee);
      delay(200);
    }
    if (!button_DOWNR) {
      if (--ee.setting.stop_index < 0) ee.setting.stop_index = 99;
      EEPROM.put(0, ee);
      delay(200);
    }
    break;
*/

   case 0:
    lcd.setCursor(0, 0);
    lcd.print("______MENU______");
    lcd.setCursor(0, 1);
    sprintf(buff, "< S. Index :%2d >", ee.setting.stop_index);
    lcd.print(buff);
    if (!button_UPR) {
      if (++ee.setting.stop_index > 99) ee.setting.stop_index = 0;
      EEPROM.put(0, ee);
      delay(200);
    }
    if (!button_DOWNR) {
      if (--ee.setting.stop_index < 0) ee.setting.stop_index = 99;
      EEPROM.put(0, ee);
      delay(200);
```

```
  }
  break;

case 1:
  lcd.setCursor(0, 0);
  lcd.print("______MENU______");
  lcd.setCursor(0, 1);
  lcd.print("< Scan Sensor  >");
  if (!button_MENU) {
    lcd.clear();
    delay(200);
    calibrate_sensor();
  }
  break;

case 2:
  lcd.setCursor(0, 0);
  lcd.print("______MENU______");
  lcd.setCursor(0, 1);
  lcd.print("< Rest Default >");
  if (!button_MENU) {
    lcd.clear();
    delay(200);
    reset_default();
  }
  break;

case 3:
  lcd.setCursor(0, 0);
  lcd.print("______MENU______");
  lcd.setCursor(0, 1);
```

```
  lcd.print("< Kal. Naik    >");
  if (!button_MENU) {
    setMotorU(200);
    delay(100);
    setMotorU(0);
  }
  break;

case 4:
  lcd.setCursor(0, 0);
  lcd.print("______MENU______");
  lcd.setCursor(0, 1);
  lcd.print("< Kal. Turun   >");
  if (!button_MENU) {
    setMotorU(-200);
    delay(100);
    setMotorU(0);
  }
  break;

  case 5:
  lcd.setCursor(0, 0);
  lcd.print("______MENU______");
  lcd.setCursor(0, 1);
  lcd.print("< Sensor Test  >");
  if  (!button_MENU) {
      while (1){
      sensortest();
      if (!button_START) {
      lcd.clear();
      delay(200);
```

```
        break;
       }
      }
     }
    break;
  }
}
```

*plan.h

```
#define lft     left
#define fwd     forward
#define rgt     right
#define up      action_up
#define down    action_down
#define off     action_off

#define s_lancip_kanan      0b000011100011
#define s_lancip_kiri       0b110001110000
#define s_perempatan        0b111000000111
#define s_pertigaan_kanan   0b000111001110
#define s_pertigaan_kiri    0b011100111000
#define s_blok_hitam        0b111111111111
#define s_blok_putih        0b000000000000
#define s_semua_sensor      0b111111111111

#define lancip_kanan      1
#define lancip_kiri       2
#define perempatan        3
#define pertigaan_kanan   4
#define pertigaan_kiri    5
#define blok_hitam        6
```

```
#define blok_putih        7
#define semua_sensor      8

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

int cp1         = 1;
int cp2         = 10;
int cp3         = 15;
int cp4         = 20;

//2void
void plan_input() {
  //PLAN 1
  plan_set(1, 1, fwd, semua_sensor, 0, 0, 0, 0, 500);
  plan_set(1, 2, up, semua_sensor, 0, 0, 0, 0, 62);
  plan_set(1, 3, off, semua_sensor, 0, 0, 0, 0, 20);
  plan_set(1, 4, fwd, semua_sensor, 0, 200, 200, 200, 10);
  plan_set(1, 5, lft, pertigaan_kiri, 400, -150, 200, 200, 200);
  plan_set(1, 6, fwd, blok_putih, 0, 0, 0, 0, 50);
  plan_set(1, 7, down,blok_putih, 0, 0, 0, 0, 62);
  plan_set(1, 8, off, blok_putih, 0, 0, 0, 0, 2);
  plan_set(1, 9, fwd, blok_putih, 1400, -200, -200, 0, 0);
  plan_set(1, 10, lft, semua_sensor, 1470, -200, 200, 200, 15);
  plan_set(1, 11, rgt, pertigaan_kanan, 350, 200, -150, 200, 125);
  plan_set(1, 12, rgt, blok_putih, 1325, 200, -200, 200, 0);

  //PLAN 2
  plan_set(2, 1, fwd, semua_sensor, 0, 0, 0, 0, 500);
  plan_set(2, 2, up, semua_sensor, 0, 0, 0, 0, 62);
```

```
  plan_set(2, 3, off, semua_sensor, 0, 0, 0, 0, 20);
  plan_set(2, 4, fwd, semua_sensor, 0, 200, 200, 200, 10);
  plan_set(2, 5, rgt, pertigaan_kanan, 400, 200, -150, 200, 200);
  plan_set(2, 6, fwd, blok_putih, 0, 0, 0, 0, 50);
  plan_set(2, 7, down,blok_putih, 0, 0, 0, 0, 62);
  plan_set(2, 8, off, blok_putih, 0, 0, 0, 0, 2);
  plan_set(2, 9, fwd, blok_putih, 1400, -200, -200, 0, 0);
  plan_set(2, 10, rgt, semua_sensor, 1470, 200, -200, 200, 15);
  plan_set(2, 11, lft, pertigaan_kiri, 350, -150, 200, 200, 125);
  plan_set(2, 12, rgt, blok_putih, 1325, 200, -200, 200, 0);
}

*running.h
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 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 & 0b01111100));
  }
  else if (dir == right) {
    do {
      setMotor(pwmLeft * 0.3, pwmRight * 0.3);
      sensor_value = readSensor();
    }
    while (!(sensor_value & 0b00111110));
  }
}

void get_junction() {
  temp_timer = 0;
  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_up) {
      delay_action(ee.path.F[ee.setting.plan][index],
ee.path.R[ee.setting.plan][index]);
      setMotorU(200);
      delay(7000);
      lcd.clear();
      lcd.setCursor(5, 1);
      lcd.print("NAIK");
    }
    else if (ee.path.action[ee.setting.plan][index] == action_down) {
      delay_action(ee.path.F[ee.setting.plan][index],
ee.path.R[ee.setting.plan][index]);
      setMotorU(-200);
      delay(7000);
      lcd.clear();
      lcd.setCursor(5, 1);
      lcd.print("TURUN");
    }
    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]);
      setMotorU(0);
      delay(1000);
```

```
      lcd.clear();
      lcd.setCursor(5, 1);
      lcd.print("OFF");
    }
    temp_timer = ee.path.TA[ee.setting.plan][index];
    ee.path.TA[ee.setting.plan][index] = 0;
  }
}

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() {
 unsigned int sensor = sensor_running;

  switch (sensor) {
    case 0b000000000001: error = -11; break;
    case 0b000000000011: error = -10; break;
    case 0b000000000010: error = -9; break;
    case 0b000000000110: error = -8; 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 = 8; break;
    case 0b010000000000: error = 9; break;
    case 0b110000000000: error = 10; break;
```

```
  case 0b100000000000: 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 = -125;
 int maxSpeed = 255;
 moveLeft = constrain(moveLeft, minSpeed, maxSpeed);
 moveRight = constrain(moveRight, minSpeed, maxSpeed);
 setMotor(moveLeft, moveRight);
}

*sensor.h
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 < (max_sensor / 2); 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(1);
  enableSensor(0, 1);
  for (i = (max_sensor / 2); i < max_sensor; 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(1);
  enableSensor(0, 0);

  //garis putih
  //bitSensor &= 0b11111111111111;
  //bitSensor = 0b11111111111111 - 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 < (max_sensor / 2); 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 = (max_sensor / 2); i < max_sensor; 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() % 10 == 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] = minVal[i] + ((maxVal[i] - minVal[i]) / 4);
  }
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Saving...");
  EEPROM.put(0, ee);
  delay(1000);
  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);
  }
}
```