PROGRAM ANDROID

```
Sub Process_Globals
	Dim serial1 As Serial
	Dim Timer1 As Timer
	Dim connected As Boolean
	Dim data As TextWriter
	'Dim TextReader1 As TextReader
End Sub

Sub Globals
	Private Button6 As Button
End Sub

Sub Activity_Create(FirstTime As Boolean)
	serial1.Initialize("serial1")
	Timer1.Initialize("timer1",100)
	Activity.LoadLayout("layout")
End Sub

Sub Activity_Resume
If serial1.IsEnabled=False Then
	Msgbox("Please turn on your bluetooth","ERROR")
Else
	serial1.Listen
End If
End Sub

Sub mnuConnect_click
	Dim PairedDevices As Map
	PairedDevices = serial1.GetPairedDevices
	Dim l As List
	l.Initialize
	For i = 0 To PairedDevices.Size - 1
		l.Add(PairedDevices.GetKeyAt(i))
	Next
	Dim res As Int
	res = InputList(l, "Choose device", -1) 'show list with
paired devices
	If res <> DialogResponse.CANCEL Then
		serial1.Connect(PairedDevices.Get(l.Get(res)))
'convert the name to mac address
		ProgressDialogShow("Connecting...")
	End If
End Sub

Sub serial1_connected(Success As Boolean)
ProgressDialogHide
If Success Then
	ToastMessageShow("Robot is Connected",False)
	data.Initialize(serial1.OutputStream)
	'TextReader1.initialized(serial1.InputStream)
	Timer1.Enabled=True
	connected=True
	Dim icon As Bitmap
```

```
      icon.Initialize(File.DirAssets,"bluetooth_icon1.png")
      Button6.SetBackgroundImage(icon)
Else
      connected=False
      Timer1.Enabled=False
      Msgbox("Robot Connected Unsuccesfully","ERROR")
End If
End Sub

Sub Timer1_Tick

End Sub

Sub mnuDisconnect_click
ProgressDialogShow("Disconnecting...")
      serial1.Disconnect
      serial1.StopListening
      connected=False
      ToastMessageShow("Robot is Disconnected",False)
      Dim icon As Bitmap
      icon.Initialize(File.DirAssets,"bluetooth_icon.png")
      Button6.SetBackgroundImage(icon)
ProgressDialogHide
End Sub

Sub atas_Down
If connected Then
      data.Write("atasPress")
      data.Flush
End If
End Sub

Sub atas_Up
If connected Then
      data.Write("atasRelease")
      data.Flush
End If
End Sub

Sub kiri_Down
If connected Then
      data.Write("kiriPress")
      data.Flush
End If
End Sub

Sub kiri_Up
If connected Then
      data.Write("kiriRelease")
      data.Flush
End If
End Sub

Sub kanan_Down
If connected Then
      data.Write("kananPress")
```

```
      data.Flush
End If
End Sub

Sub kanan_Up
If connected Then
      data.Write("kananRelease")
      data.Flush
End If
End Sub

Sub Bawah_Down
If connected Then
      data.Write("bawahPress")
      data.Flush
End If
End Sub

Sub Bawah_Up
If connected Then
      data.Write("bawahRelease")
      data.Flush
End If
End Sub

Sub puter_Down
If connected Then
      data.Write("puterPress")
      data.Flush
End If
End Sub

Sub puter_Up
If connected Then
      data.Write("puterRelease")
      data.Flush
End If
End Sub

Sub Button6_Click
If serial1.IsEnabled=False Then
      Msgbox("Please turn on your bluetooth","ERROR")
End If
If connected=False AND serial1.IsEnabled=True Then
      mnuConnect_click
End If
If connected=True Then
      mnuDisconnect_click
End If
End Sub
```

PROGRAM MIKROKONTROLER

```
#include <LiquidCrystal.h>;
LiquidCrystal lcd(0, 2, 4, 5, 6, 7);

const int pin_LCDRW = 1; //Read/Write LCD
const int pin_LCDLED = 3; //Backlight
String data;
#include <EEPROM.h>
struct EEPROMDATA {
 int SPEED;
 int LIMIT_VALUE_SENSOR[14];
 int MAZE_MODE;
 int INDEXDATA[100];
 char NAME[16];
};
EEPROMDATA EE;

#define MAZE_LEFT 0
#define MAZE_RIGHT 1
#define MAZE_RUN 2
#define FWD 0
#define LEFT 1
#define RIGHT 2
#define STOP 3
```

```
#define pin_MOTOR_DIRL 11
#define pin_MOTOR_PWML 12
#define pin_MOTOR_DIRR 14
#define pin_MOTOR_PWMR 13

#define pin_button_UPL 16
#define pin_button_DOWNL 20
#define pin_button_UPR 17
#define pin_button_DOWNR 21
#define pin_button_START 18
#define pin_button_MENU 19

#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)

#define pin_ENABLE_SENSORL 23
#define pin_ENABLE_SENSORR 22

const int pin_ADC_SENSOR[14] = {A7, A6, A5, A4, A3, A2, A1, A1, A2, A3, A4, A5, A6, A7};
void setup() {
```

```
// put your setup code here, to run once:
  Serial.begin(9600);
init_button();
init_sensor();
pinMode(pin_LCDRW, OUTPUT);
pinMode(pin_LCDLED, OUTPUT);
digitalWrite(pin_LCDRW, LOW);
digitalWrite(pin_LCDLED, HIGH);
lcd.begin(16, 2);
lcd.clear();
delay(100);
lcd.setCursor(0, 0);
lcd.print("Muhammad Alfian");
lcd.setCursor(0, 1);
lcd.print("6TD");
delay(1000);
lcd.clear();

EEPROM.get(0, EE);
if (strcmp(EE.NAME, "ichibot") != 0 || !button_DOWNL) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Init EEPROM data");
  memset(EE.LIMIT_VALUE_SENSOR, 0,
sizeof(EE.LIMIT_VALUE_SENSOR));
```

```
  EE.SPEED = 80;
  EE.MAZE_MODE = MAZE_RUN;
  memset(EE.INDEXDATA, FWD, sizeof(EE.INDEXDATA));
  strcpy(EE.NAME, "ichibot");
  EEPROM.put(0, EE);
  lcd.setCursor(0, 1);
  lcd.print("Completed.");
  delay(1000);
  lcd.clear();
 }
}
void loop() {
 // put your main code here, to run repeatedly:
 displayHomeScreen();
 if (!button_START) {
  switch (EE.MAZE_MODE) {
  case MAZE_LEFT:
    run_maze_left();
  break;
  case MAZE_RIGHT:
    run_maze_right();
  break;
  case MAZE_RUN:
    goooooooo();
  break;
```

```
  }
 }
 if (!button_MENU) {
  setting();
 }
}
void displayHomeScreen() {
 int i;
 int valSensor = readSensor();
 lcd.setCursor(1, 0);
 for (i = 0; i < 14; i++) {
  if (valSensor & (0b10000000000000 >> i)) {
  lcd.write(0xff);
 } else {
  lcd.write('_');
 }
 }
 lcd.setCursor(11, 1);
 char buff[16];
 sprintf(buff, "V:%3d", EE.SPEED);
 lcd.print(buff);
 if (!button_UPR) {
  if (++EE.SPEED > 255) EE.SPEED = 0;
  EEPROM.put(0, EE);
  delay(200);
```

```
}
if (!button_DOWNR) {
  if (--EE.SPEED < 0) EE.SPEED = 255;
  EEPROM.put(0, EE);
  delay(200);
}
lcd.setCursor(0, 1);
switch (EE.MAZE_MODE) {
case MAZE_LEFT:
  lcd.print("MAZE LEFT ");
break;
case MAZE_RIGHT:
  lcd.print("BLUETOOTH");
break;
case MAZE_RUN:
  lcd.print("MAZE RUN ");
break;
}
if (!button_UPL) {
  if (++EE.MAZE_MODE > 2) EE.MAZE_MODE = 0;
  EEPROM.put(0, EE);
  delay(200);
}
if (!button_DOWNL) {
  if (--EE.MAZE_MODE < 0) EE.MAZE_MODE = 2;
```

```
  EEPROM.put(0, EE);
  delay(200);
 }
}
int indexVal = 0;
int stopIndex = 99;
void goooooooo() {
 lcd.clear();
 lcd.print("GOOOO....");
 delay(200);
 digitalWrite(pin_LCDLED, LOW);
 indexVal = 0;
 while (1) {
  followLine();
  if (solve_maze() == 0) {
   break;
  }
  if (!button_START) {
   break;
  }
 }
 setMotor(0, 0);
 digitalWrite(pin_LCDLED, HIGH);
 lcd.clear();
 lcd.print("Completed....");
```

```
 delay(2000);
 lcd.clear();
}
int error = 0;
int lastError = 0;
int kp = 15;
int kd = 100;

void followLine() {
 int sensor = readSensor();
 switch (sensor) {
 case 0b00000000000001: error = -13; break;
 case 0b00000000000011: error = -12; break;
 case 0b00000000000010: error = -11; break;
 case 0b00000000000110: error = -10; break;
 case 0b00000000000100: error = -9; break;
 case 0b00000000001100: error = -8; break;
 case 0b00000000001000: error = -7; break;
 case 0b00000000011000: error = -6; break;
 case 0b00000000010000: error = -5; break;
 case 0b00000000110000: error = -4; break;
 case 0b00000000100000: error = -3; break;
 case 0b00000001100000: error = -2; break;
 case 0b00000001000000: error = -1; break;
 case 0b00000011000000: error = 0; break;
```

```
case 0b00000010000000: error = 1; break;
case 0b00000110000000: error = 2; break;
case 0b00000100000000: error = 3; break;
case 0b00001100000000: error = 4; break;
case 0b00001000000000: error = 5; break;
case 0b00011000000000: error = 6; break;
case 0b00010000000000: error = 7; break;
case 0b00110000000000: error = 8; break;
case 0b00100000000000: error = 9; break;
case 0b01100000000000: error = 10; break;
case 0b01000000000000: error = 11; break;
case 0b11000000000000: error = 12; break;
case 0b10000000000000: error = 13; break;
}
int rateError = error - lastError;
lastError = error;
int moveVal = (int) (error * kp) + (rateError * kd);
int moveLeft = EE.SPEED - moveVal;
int moveRight = EE.SPEED + moveVal;
int minSpeed = -125;
int maxSpeed = 255;
moveLeft = constrain(moveLeft, minSpeed, maxSpeed);
moveRight = constrain(moveRight, minSpeed, maxSpeed);
setMotor(moveLeft, moveRight);
}
```

```
int solve_maze() {
 int sensor = readSensor();
 int sensorMid = sensor & 0b00000111100000;
 int sensorLeft = sensor & 0b00000000001111;
 int sensorRight = sensor & 0b11110000000000;
 int get_index = 0;
 if (sensorMid && sensorLeft ) {
  get_index = 1;
 }else if (sensorMid && sensorRight ) {
  get_index = 1;
 }else if (sensorRight && sensorLeft ) {
  get_index = 1;
 };
 if (get_index == 1) {
  digitalWrite(pin_LCDLED, HIGH);
  indexVal ++;
  if ( indexVal > 99 ) {
   return 0;
  }
  if (EE.INDEXDATA[indexVal] == STOP) {
   return 0;
  }else if (EE.INDEXDATA[indexVal] == LEFT) {
    setMotor(-125, 200);
    delay(100);
```

```
  } else if (EE.INDEXDATA[indexVal] == RIGHT) {
    setMotor(200, -125);
    delay(100);
  } else if (EE.INDEXDATA[indexVal] == FWD) {
  setMotor(EE.SPEED, EE.SPEED);
  unsigned long startMillis = millis();
  unsigned long interVal = 150;
  while (millis() - startMillis < interVal) {
   followLine();
  }
}
lcd.setCursor(0, 1);
lcd.print("LastIndex:");
lcd.print(indexVal);
lcd.print(" ");
if (EE.INDEXDATA[indexVal] == LEFT) {
  lcd.print("LEFT");
} else if (EE.INDEXDATA[indexVal] == RIGHT) {
  lcd.print("RGHT");
} else if (EE.INDEXDATA[indexVal] == FWD) {
  lcd.print("FWD ");
}
  digitalWrite(pin_LCDLED, LOW);
}
return 1;
```

```
}
void setting() {
 lcd.clear();
 delay(200);
 int menu = 0;
 while (1) {
  if (!button_UPL) {
    if (++menu > 1) menu = 0;
    delay(200);
  }
  if (!button_DOWNL) {
    if (--menu < 0) menu = 1;
    delay(200);
  }
  switch (menu) {
  case 0:
    lcd.setCursor(0, 0);
    lcd.print("> View DataIndex");
    lcd.setCursor(0, 1);
    lcd.print(" Scan Sensor ");
    if (!button_MENU)viewDataIndex() ;
  break;
  case 1:
    lcd.setCursor(0, 0);
    lcd.print(" View DataIndex");
```

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

void viewDataIndex() {
 lcd.clear();
 delay(200);
 int i = 1;
 while (1) {
  lcd.setCursor(0, 0);
  lcd.print("index: ");
  lcd.print(i);
  lcd.print("  ");
  if (!button_UPL) {
    if (++i > 99) i = 1;
    delay(200);
```

```
}
if (!button_DOWNL) {
  if (--i < 1) i = 99;
  delay(200);
}
lcd.setCursor(0, 1);
if (EE.INDEXDATA[i] == LEFT) {
  lcd.print("Value: LEFT ");
} else if (EE.INDEXDATA[i] == RIGHT) {
  lcd.print("Value: RIGHT ");
} else if (EE.INDEXDATA[i] == FWD) {
  lcd.print("Value: FWD ");
} else if (EE.INDEXDATA[i] == STOP) {
  lcd.print("Value: STOP ");
} else {
  lcd.print("Value: ERROR");
}
if (!button_UPR) {
if (++EE.INDEXDATA[i] > STOP) EE.INDEXDATA[i] = FWD;
  EEPROM.put(0, EE);
 delay(200);
}
if (!button_DOWNR ) {
if (--EE.INDEXDATA[i] < FWD) EE.INDEXDATA[i] = STOP;
 EEPROM.put(0, EE);
```

```
    delay(200);
   }
   if (!button_START) {
     break;
   }
 }

 lcd.clear();
 delay(200);
}
void run_maze_left() {
 lcd.clear();
 delay(200);
 //lcd.setCursor(0, 0);
 //lcd.print("Kamu Bisa Bikin");
 //lcd.setCursor(0, 1);
 //lcd.print("programnya?");
 //while (button_START);
 //delay(200);
 //lcd.clear();
}
void run_maze_right() {
 lcd.clear();
 delay(200);
bluetooth();
```

```
// lcd.setCursor(0, 0);
// lcd.print("Kamu Bisa Bikin");
// lcd.setCursor(0, 1);
// lcd.print("programnya?");
// while (button_START);
// delay(200);
// lcd.clear();
}
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 enableSensor(int L, int R) {
 digitalWrite(pin_ENABLE_SENSORL, L);
 digitalWrite(pin_ENABLE_SENSORR, R);
 delay(1);
}
void init_sensor() {
 int i = 0;
 for (i = 0; i < 14; 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;
 enableSensor(1, 0);
```

```
 for (i = 0; i < 7; i++) {
   if (analogRead(pin_ADC_SENSOR[i]) > EE.LIMIT_VALUE_SENSOR[i])
{
     bitSensor = bitSensor | (0b10000000000000 >> i);
   }
 }
 enableSensor(0, 1);
 for (i = 7; i < 14; i++) {
   if (analogRead(pin_ADC_SENSOR[i]) > EE.LIMIT_VALUE_SENSOR[i])
{
     bitSensor = bitSensor | ( 0b10000000000000 >> i);
   }
 }
 enableSensor(0, 0);
 return bitSensor;
}
int calibrate_sensor() {
 int i, valSensor, xCursor = 0;
 int minVal[14], maxVal[14];
 lcd.clear();
 for (i = 0; i < 14; i++) {
   minVal[i] = 1023;
   maxVal[i] = 0;
 }
 while (1) {
 enableSensor(1, 0);
```

```
for (i = 0; i < 7; i++) {
  valSensor = analogRead(pin_ADC_SENSOR[i]);
  if (valSensor > maxVal[i]) {
    maxVal[i] = valSensor;
  }
  if (valSensor < minVal[i]) {
    minVal[i] = valSensor;
  }
}
enableSensor(0, 1);
for (i = 7; i < 14; i++) {
  valSensor = analogRead(pin_ADC_SENSOR[i]);
  if (valSensor > maxVal[i]) {
    maxVal[i] = valSensor;
  }
  if (valSensor < minVal[i]) {
    minVal[i] = valSensor;
  }
}
enableSensor(0, 0);
if (!button_START) {
  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 < 14; i++) {
  EE.LIMIT_VALUE_SENSOR[i] = ((maxVal[i] - minVal[i]) / 2) +
minVal[i];
 }
 lcd.clear();
 lcd.setCursor(0, 0);
 lcd.print("Saving...");
 EEPROM.put(0, EE);
 delay(500);
 lcd.clear();
}
void bluetooth() {
  // put your main code here, to run repeatedly:
  lcd.clear();
  while(1){
```

```
  lcd.setCursor(0,0);
  lcd.print("MODE BLUETOOTH");
  if (Serial.available()) {
  delay(100);
  data = "";
  while (Serial.available()) {
    char c = Serial.read();
    data += c;
  }
}
if (data == "atasPress") {
  setMotor(200, 200);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("maju");
} else if (data == "bawahPress") {
  setMotor(-200, -200);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("mundur");
} else if (data == "kananPress") {
  setMotor(-190, 200);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("kanan");
```

```
  } else if (data == "kiriPress") {
    setMotor(190, -200);
    lcd.print("kiri");
    lcd.clear();
    lcd.setCursor(0, 0);
  } else if (data == "puterPress") {
    setMotor(-200, 200);
    lcd.print("puter");
    lcd.clear();
    lcd.setCursor(0, 0);
  } else {
    setMotor(0, 0);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("stop");
  }
  }
}
```