

LAMPIRAN

Lampiran 1. 1 Program ESP32

```
#pragma region Kosong

#include <Arduino.h>

#include <LiquidCrystal_I2C.h>

#include <max6675.h>

#include <Wire.h>

#include <ESP_Google_Sheet_Client.h>

#include <Adafruit_INA219.h>

#include <WiFi.h>

#include <I2CKeyPad.h>


#pragma region WiFi

//Wifi

const char* ssid    = "*****";

const char* password = "*****";

int rssi = 0;

char wipistat = 'X';

bool WifiStat = false;

unsigned long Wtm = 0;

const unsigned long Wwtm = 500;

bool terhubung = LOW;

#pragma endregion

#pragma region Declaration

#define RXD2 35

#define TXD2 32

//MAX6675

#define thermoDO  34

#define thermoCS1 19

#define thermoCS2 18

#define thermoCS3 17

#define thermoCS4 16

#define thermoCS5 33

#define thermoCLK 23
```

```

MAX6675 thermocouple1(thermoCLK, thermoCS1, thermoDO);
MAX6675 thermocouple2(thermoCLK, thermoCS2, thermoDO);
MAX6675 thermocouple3(thermoCLK, thermoCS3, thermoDO);
MAX6675 thermocouple4(thermoCLK, thermoCS4, thermoDO);
MAX6675 thermocouple5(thermoCLK, thermoCS5, thermoDO);

float C1 = 0;
float C2 = 0;
float C3 = 0;
float C4 = 0;
float C5 = 0;
float Avg_Bat = 0;
float Avg_Ov = 0;
float Delta_Bat = 0;
float DeltaOldBat = 0;
float Delta_Ov = 0;
float DeltaOldOv = 0;
bool FirstDataBat = true;
bool FirstDataOv = true;

//I2C
I2CKeyPad keyPad(0x20);
char Och = 'N';
char ch = 'N';
bool Keyin = false;
unsigned long lastPressTime = 0;
const unsigned long debounceDelay = 50;
char keymap[19] = "123A456B789C*0#DNF";

/*
D # 0 *
C 9 8 7
B 6 5 4
A 3 2 1
*/

```

```
LiquidCrystal_I2C lcd(0x27,20,4);
```

```
Adafruit_INA219 ina219;
```

```
float Solar_Radiation = 0;
```

```
//UART
```

```
float voltage = 0;
```

```
float current = 0;
```

```
float Kelembapan_DHT_Oven = 0;
```

```
float Suhu_DHT_Oven = 0;
```

```
float Kelembapan_DHT_Battery = 0;
```

```
float Suhu_DHT_Battery = 0;
```

```
//INA219
```

```
float BusVoltage = 0;
```

```
float Current_mA = 0;
```

```
float DayaIna = 0;
```

```
//Epoch Time
```

```
const char* ntpServer = "pool.ntp.org";
```

```
String waktuStr = "";
```

```
String getFormattedTime() {
```

```
    struct tm timeinfo;
```

```
    if (!getLocalTime(&timeinfo)) {
```

```
        return "Time Error";
```

```
    }
```

```
    char buf[25];
```

```
    strftime(buf, sizeof(buf), "%d %b %Y, %H:%M", &timeinfo);
```

```
    return String(buf);
```

```
}
```

```
//LCD Display
```

```
enum Layar {Home, Changing, Suhu_Battery, Suhu_Oven, DHT21_OVEN, DHT21_BATTERY,  
Panel_Utama, Radiasi, Oven_Run, Oven_Stop, Oven_Running, Oven_Stopping, Restart, LCD_Restart, zzz};
```

```
Layar layar = Home;
```

```

Layar LastLayar = Home;

unsigned long lastLcdSwitch = 0;

const unsigned long lcdInterval = 3000;

int lcdPage = 0;

const int totalLcdPages = 6;

int LcdReset = -1;


//Run Oven

bool OvenMenu = false;

bool OvenState = false;

bool OvenRunning = false;

unsigned long OvnPup = 0;

const unsigned long OvnPupInv = 2000;


#pragma endregion

#pragma region SpreadSheet

//Google SpreadSheet

#define PROJECT_ID "*****"

#define CLIENT_EMAIL "*****"

const char PRIVATE_KEY[] PROGMEM = "-----BEGIN PRIVATE KEY-----
\nMIIEvGIBADANBgkqhkiG9w0BAQEFAASCBAgEAAoIBAQDIVXxYy1I/g+Ui\nHcNcxUKk3
iBtoWrXXvrk3AQBk6sCpsXxgWndtVr0WRTNudgUL3Fslg8I2dMO19Nb\n+oax6g9UR0/J2f1pLyfDwyOpc
P2SpvM427EvEXy6z0kFQrzGuWH2QgOicEiqc3YN\nWBKXZwbbgV3pSkSABDhGy/+HLXQvgFRuY7C
1OzCKd9O/PwqraAy2cAlfiKEzfpVz\n4AQH+OXi0Zen9HE/GrUIVSNGuZhLzH9ibJXBN1j0rTUVMG3I7kl
gGjXUU0tHVQQy\nD6Vyft5xkff/n4mk2+xDD7/y8cKymI Q2lkPdAc2zIXJ2fjRxKcZZ0QCW+MA2Hutm\nc
FkOr5nhAgMBAAECggEAD7DwwXYLJ4bf/YdWol6Uf9Y17F0Gw9Epw/vCIhR8jIF6\nL4XDXKonshUf7
VqO9wSZqLcUtMOTTg5E1c6B+vgw5SXoRunteDiuf8BhNzBLXUZ4\ng6D+fMy1bCJk1IwQ1rJWSk8700
MGVHPyysNw/t8g/ZGv/1diRqBjSa7JX0ekIzVu\nBc/DF33XLccn0DrVF+XDRggs+z6drwE+9HosWgJLgue
q8qthaaQe60GqxhihAufh\ny93LsfZb4VWtHYtZLU3QdtSsY0vNBcVfWrozD7ONVYQob/a3dJU3tapwZ0pjz
KUO\nEo5e++mJTfuZTcccfJdJSgThx Cov29gXt5FIlhTJQKBgQD5Q6P5V0jG0gfmD1Ug\nd/3HnJWNw4
3YUjZ4JrZxEaUBB2aSYnTYlwxj mj/3T+eDmOVjCCernL0lVWcrJtTX\ntiWiunVpGoYg/qIs6ledbzgFBr1X
QnH59dXHnuCDNrGn6BeLtZe6agS6dpm7iqpN\nU+OmxCLVmgWzyDsHZm/3lWCSBQKBgQDrh/ixl0G
1fMvZV0C8fcPdPGfi8/cdrnha\njoASWZh0QgE5tYyIPxLSDV5Z1frcgVXCn/2paRoo+5JrbftuN5udYqKTM
ILRyC8B\nAQIAF552b9wazAW2M/DJ6poxZPwPwGPsONz+Zw9obxNGlUEdznHrhG0gTwyz2PPC\nnOfD
H1aZjLQKBgDBD03epTfEl13qABxhhr9AHRc1X4DHU4dS2TjfqIIxulaxZhipw\nnRGsfBsJJsrYJ+JgagPf/V
YPk6jupMSsZoHvOiff2wAeu4HCuyTU22zu+bwZ7Ufoa\nnBPzSiuVqu1XE5qyo82OsJna3hGsrJxgR6t30uH
2RgQG9HBB3fUoKI96xAoGBAMio\n7O2P6cFO8vTVayerFeU0A2XNZ2j4LT0pTl9TyZiyawZaULiSYx/
KztqInEeFwxa\nnecQzeLAGXvZFW5PHdq2H7+6gplrLPB0BVPcXrQuNJz32J3kHm5H2tneKQjaZcbo\nno
w5yiyP0Db4Chpl4OoHmLnnpqHm5SIUmRxlY4lp5AoGBAljMXNNSFcJfLceYnPUa\n/nz/Kg128HzTcPx/9F
7KdyMOU8vDh32x/NdrWfB8QvNl bj+b9TDDwb0t431gQVey6K\ncKqv7jKZQS+Cwl6dYD4HgJ5NNK4E
+jCVZven8Bwu69JwgZ9PY3TPwBkD8e92HbE\na+oWT3Y1oWQO1CgQbm85A9ns\n-----END
PRIVATE KEY-----\n";

const char spreadsheetId_Battery[] = "1bFpMzBMYbp22Cxi0gPL_6lIPWc3sAsvJRL0l1lhjOdA";

const char spreadsheetId_Oven[] = "1NHFF7cSrDvKOj7Pulk3y_ySFXngMCFbdraTplyGHH7Y";

```

```

bool SheetStat = false;
int interval = 300000;
unsigned long tmnw = 0;
int ReadInterval = 200;
unsigned long Rdtm = 0;
int WriteInterval = 500;
unsigned long Wrtm = 0;
int Reconnect_Interval = 500;
unsigned long Rctm = 0;
//600000 = 10 menit
#pragma endregion
#pragma region Prototype
void Gsheet_Boot();
void Gsheet_Battery();
void Gsheet_Oven();
void Thermo_Battery();
void Thermo_Oven();
void wifi();
void Receive();
void Transmit();
void INA219();
void Rumus();
void Keypad();
void Interface();
void SerialMonitor();
#pragma endregion

void setup() {
    Serial.begin(115200);
    Serial2.begin(9600, SERIAL_8N1, RXD2, TXD2);
    Serial2.setTimeout(20);
    pinMode(LED_BUILTIN, OUTPUT);

    Wire.begin();

```

```

if (keyPad.begin() == false)
{
    Serial.println("\nERROR: cannot communicate to keypad.\nPlease reboot.\n");
    while (1);
}

keyPad.loadKeyMap(keymap);
delay(500);

lcd.init();
lcd.backlight();
lcd.setCursor(0,0);
lcd.print("Sand Battery");
lcd.setCursor(0,1);
lcd.print("Dryer");
lcd.setCursor(0,3);
lcd.print("0=Skip D=Restart");
delay(500);

WiFi.begin(ssid, password);
Serial.print("starting..");

while (WiFi.status() != WL_CONNECTED && WifiStat == false) {
    ch = keyPad.getChar();
    if(millis()-Wtm >= Wwtm){
        Serial.print(".");
        lcd.setCursor(0, 2);
        lcd.print("Wifi..");
        Wtm = millis();
    }
    if(ch == '3'){
        WifiStat = true;
    }
}

```

```

else if(ch == '1'){
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Restarting...");
    delay(500);
    ESP.restart();
}
delay(50);
}

Serial.println("\nWiFi connected!");
Serial.print("IP: ");
Serial.println(WiFi.localIP());

// if (! ina219.begin()) {
//   Serial.println("Failed to find INA219 chip");
//   lcd.clear();
//   lcd.setCursor(0,0);
//   lcd.print("Failed to find ");
//   lcd.setCursor(0,1);
//   lcd.print("INA219 chip");
//   while (1) { delay(10); }
// }

configTime(7 * 3600,0 , ntpServer);

//spread Sheet
if(WiFi.status() == WL_CONNECTED)Gsheet_Boot();
lcd.clear();

}

void loop() {
    wifi();
    Receive();
    Interface();

```

```

//Read sens
if(millis() - Rdtm >= ReadInterval ){
    Thermo_Battery();
    Thermo_Oven();
    INA219();
    //SerialMonitor();
    Rdtm = millis();
}

//Gsheet Upload
if(WiFi.status() == WL_CONNECTED && millis() - tmnw >= interval){
    Rumus();
    Gsheet_Battery();
    if(OvenState) Gsheet_Oven();
    tmnw = millis();
}

//Tx to Arduino
if(millis() - Wrtm >= WriteInterval ){
    Transmit();
    Wrtm = millis();
}
}

#pragma region Void
void Gsheet_Boot(){
    GSheet.printf("ESP Google Sheet Client v%s\n\n", ESP_GOOGLE_SHEET_CLIENT_VERSION);
    GSheet.setPrerefreshSeconds(10 * 60);
    GSheet.begin(CLIENT_EMAIL, PROJECT_ID, PRIVATE_KEY);

    while (!GSheet.ready() && SheetStat == false) {
        ch = keyPad.getChar();
        if(millis()-Wtm >= Wwtm){
            Serial.print(".");

```

```

lcd.setCursor(0, 2);
lcd.print("GSheet..");
Wtm = millis();}

if(ch == '3'){
    SheetStat = true;
}
else if(ch == '1'){
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Restarting...");
    delay(2000);
    ESP.restart();
}
delay(50);
}

Serial.println("\nToken ready!");

FirebaseJson response1;
Serial.println("\nUpdate spreadsheet values...");
Serial.println("-----");
FirebaseJson valueRange1;
valueRange1.add("majorDimension", "ROWS");
valueRange1.set("values/[0]/[0]", "Jam"); // row 1/ column 1
valueRange1.set("values/[0]/[1]", "Suhu 1"); // row 2/ column 1
valueRange1.set("values/[0]/[2]", "Suhu 2");
valueRange1.set("values/[0]/[3]", "Suhu rerata");
valueRange1.set("values/[0]/[4]", "Delta T");
valueRange1.set("values/[0]/[5]", "Tegangan");
valueRange1.set("values/[0]/[6]", "Arus");
valueRange1.set("values/[0]/[7]", "Daya");
valueRange1.set("values/[0]/[8]", "Radiasi");
valueRange1.set("values/[0]/[9]", "V INA");
valueRange1.set("values/[0]/[10]", "A INA");
valueRange1.set("values/[0]/[11]", "P INA");
valueRange1.set("values/[0]/[12]", "Suhu DHT");

```

```

    valueRange1.set("values/[0]/[13]", "Kelembapan ");

    bool success1 = GSheet.values.append(&response1 /* returned response */, spreadsheetId_Battery /*
spreadsheet Id to update */, "Sheet1!A1" /* range to update */, &valueRange1 /* data to update */);

    if (success1) {
        Serial.println("Upload Battery berhasil!");
        response1.toString(Serial, true);
    } else {
        Serial.println("Upload Battery Gagal!");
        Serial.println(GSheet.errorReason());
    }

    FirebaseJson response2;

    Serial.println("\nUpdate spreadsheet values...");
    Serial.println("-----");
    FirebaseJson valueRange2;

    valueRange2.add("majorDimension", "ROWS");
    valueRange2.set("values/[0]/[0]", "Jam"); // row 1/ column 1
    valueRange2.set("values/[0]/[1]", "Suhu 1"); // row 2/ column 1
    valueRange2.set("values/[0]/[2]", "Suhu 2");
    valueRange2.set("values/[0]/[3]", "Suhu 3");
    valueRange2.set("values/[0]/[4]", "Suhu Rerata");
    valueRange2.set("values/[0]/[5]", "Delta T");
    valueRange2.set("values/[0]/[6]", "Suhu DHT");
    valueRange2.set("values/[0]/[7]", "Kelembapan");
    //valueRange2.set("values/[0]/[6]", "Daya");

    bool success2 = GSheet.values.append(&response2 /* returned response */, spreadsheetId_Oven /*
spreadsheet Id to update */, "Sheet1!A1" /* range to update */, &valueRange2 /* data to update */);

    if (success2) {
        Serial.println("Upload Oven berhasil!");
        response2.toString(Serial, true);
    } else {
        Serial.println("Upload Oven Gagal!");
    }

```

```

    Serial.println(GSheet.errorReason());
}

delay(2000);

Thermo_Battery();
Thermo_Oven();
INA219();
Receive();
Rumus();
Gsheet_Battery();
Gsheet_Oven();
}

void Gsheet_Battery(){
    waktuStr = getFormattedTime();
    Serial.println("Uploading ke Battery!");

    FirebaseJson response1;
    Serial.println("\nUpdate spreadsheet values...");
    Serial.println("-----");
    FirebaseJson valueRange1;
    valueRange1.add("majorDimension", "ROWS");
    valueRange1.set("values/[0]/[0]", waktuStr); // row 1/ column 1
    valueRange1.set("values/[0]/[1]", C1); // row 2/ column 1
    valueRange1.set("values/[0]/[2]", C2);
    valueRange1.set("values/[0]/[3]", Avg_Bat);
    valueRange1.set("values/[0]/[4]", Delta_Bat);
    valueRange1.set("values/[0]/[5]", voltage);
    valueRange1.set("values/[0]/[6]", current);
    valueRange1.set("values/[0]/[7]", (voltage*current));
    valueRange1.set("values/[0]/[8]", Solar_Radiation);
    valueRange1.set("values/[0]/[9]", BusVoltage);
    valueRange1.set("values/[0]/[10]", Current_mA);
    valueRange1.set("values/[0]/[11]", DayaIna);
    valueRange1.set("values/[0]/[12]", Suhu_DHT_Battery);

```

```

valueRange1.set("values/[0]/[13]", Kelembapan_DHT_Battery);

bool success1 = GSheet.values.append(&response1 /* returned response */, spreadsheetId_Battery /*
spreadsheet Id to update */, "Sheet1!A1" /* range to update */, &valueRange1 /* data to update */);

if (success1) {
    Serial.println("Upload Battery berhasil!");
    response1.toString(Serial, true);
} else {
    Serial.println("Upload Battery Gagal!");
    Serial.println(GSheet.errorReason());
}
}

void Gsheet_Oven(){
    waktuStr = getFormattedTime();
    Serial.println("Uploading ke Oven!");

    FirebaseJson response2;
    Serial.println("\nUpdate spreadsheet values...");
    Serial.println("-----");
    FirebaseJson valueRange2;
    valueRange2.add("majorDimension", "ROWS");
    valueRange2.set("values/[0]/[0]", waktuStr); // row 1/ column 1
    valueRange2.set("values/[0]/[1]", C3); // row 2/ column 1
    valueRange2.set("values/[0]/[2]", C4);
    valueRange2.set("values/[0]/[3]", C5);
    valueRange2.set("values/[0]/[4]", Avg_Ov);
    valueRange2.set("values/[0]/[5]", Delta_Ov);
    valueRange2.set("values/[0]/[6]", Suhu_DHT_Oven);
    valueRange2.set("values/[0]/[7]", Kelembapan_DHT_Oven);
    //valueRange2.set("values/[0]/[6]", "Daya");

    bool success2 = GSheet.values.append(&response2 /* returned response */, spreadsheetId_Oven /*
spreadsheet Id to update */, "Sheet1!A1" /* range to update */, &valueRange2 /* data to update */);

    if (success2) {

```

```

    Serial.println("Upload Oven berhasil!");
    response2.toString(Serial, true);
} else {
    Serial.println("Upload Oven Gagal!");
    Serial.println(GSheet.errorReason());
}}

void Thermo_Battery(){
    C1 = thermocouple5.readCelsius();
    C2 = thermocouple4.readCelsius();
    Avg_Bat = ((C1+C2)/2);
}

void Thermo_Oven(){
    C3 = thermocouple3.readCelsius();
    C4 = thermocouple2.readCelsius();
    C5 = thermocouple1.readCelsius();
    Avg_Ov = ((C3+C4+C5)/3);
}

void wifi(){
    rssi = WiFi.RSSI();

    if(WiFi.status() == WL_CONNECTED && !terhubung ){
        configTime(7 * 3600,0 , ntpServer);
        Serial.println("Connected!");
        wipistat = 'O';

        digitalWrite(2, HIGH);

        terhubung = HIGH;
    }

    if(WiFi.status() != WL_CONNECTED){
        if(millis() - Rctm > Reconnect_Interval){
            Serial.println(".");
            wipistat = 'X';
            WiFi.disconnect();

```

```

    WiFi.begin(ssid, password);

    digitalWrite(2, LOW);

    terhubung = LOW;

    Rctm = millis();
  }
}

void Receive(){
  if (Serial2.available()) {
    digitalWrite(LED_BUILTIN, HIGH);

    String data = Serial2.readStringUntil('\n');
    data.trim();

    int c1 = data.indexOf('/');
    int c2 = data.indexOf('/', c1 + 1);
    int c3 = data.indexOf('/', c2 + 1);
    int c4 = data.indexOf('/', c3 + 1);
    int c5 = data.indexOf('/', c4 + 1);

    if (c1 > 0 && c2 > c1 && c3 > c2 && c4 > c3 && c5 > c4) {
      voltage      = data.substring(0, c1).toFloat();
      current      = data.substring(c1 + 1, c2).toFloat();
      Kelembapan_DHT_Oven = data.substring(c2 + 1, c3).toFloat();
      Suhu_DHT_Oven      = data.substring(c3 + 1, c4).toFloat();
      Kelembapan_DHT_Battery = data.substring(c4 + 1, c5).toFloat();
      Suhu_DHT_Battery      = data.substring(c5 + 1).toFloat();
    } else {
      Serial.print("Data format tidak valid: ");
      Serial.println(data);
    }
  }

  else{digitalWrite(LED_BUILTIN, LOW);}
}

void Transmit(){
  //Transmit

  Serial2.print(C3);

  Serial2.print("");

  Serial2.print(C4);

```

```

Serial2.print("/");
Serial2.print(C5);
Serial2.print("/");
Serial2.println(OvenState);
}
void INA219(){
  // BusVoltage = ina219.getBusVoltage_V();
  // Current_mA = ina219.getCurrent_mA()/1000;
  // DayaIna = (ina219.getPower_mW()/1000);
}
void Rumus(){
  //Battery
  if(FirstDataBat){
    DeltaOldBat = Avg_Bat;
    FirstDataBat = false;
  }
  else{
    Delta_Bat = ((Avg_Bat - DeltaOldBat)/(interval/60000));
  }
  DeltaOldBat = Avg_Bat;

  //Oven
  if(FirstDataOv){
    DeltaOldOv = Avg_Ov;
    FirstDataOv = false;
  }
  else{
    Delta_Ov = ((Avg_Ov - DeltaOldOv)/(interval/60000));
  }
  DeltaOldOv = Avg_Ov;

  //Radiasi Matahari
  Solar_Radiation = (DayaIna/0.061425)*17.46;
}
void Interface(){

```

```

char Nch = keyPad.getChar();

if(Nch != Och) {
    if(Nch != 'N' && Nch != 'F') { ch = Nch; }
    else { Keyin = false; ch = 'N'; }
    Och = Nch;
} else {
    ch = 'N';
}

if(ch != 'N') {
    Serial.print("Key Pressed : ");
    Serial.println(ch);
}

if(layar != zzz){
    static unsigned long Moment = 0;
    if(millis()-Moment>=500){
        Moment = millis();
        Serial.print("layar : ");
        Serial.println(layar);
        Serial.print("Bool : ");
        Serial.print(OvenMenu);
        Serial.print("/");
        Serial.print(OvenState);
        Serial.print("/");
        Serial.println(OvenRunning);
    }
}

if(ch == '3' && !OvenMenu) { layar = Home; }
else if(ch == '3' && OvenMenu) { layar = LastLayar; }
else if(ch == '*') { layar = Changing; }
else if(ch == 'D') { layar = Suhu_Battery; }

```

```

else if(ch == '#') { layar = Suhu_Oven; }
else if(ch == '0') { layar = DHT21_OVEN; }
else if(ch == 'C') { layar = DHT21_BATTERY; }
else if(ch == '9') { layar = Panel_Utama; }
else if(ch == '8') { layar = Radiasi; }
else if(ch == '7' && !OvenMenu && !OvenState) { LastLayar = layar; layar = Oven_Run; }
else if(ch == '7' && !OvenMenu && OvenState) { LastLayar = layar; layar = Oven_Stop; }
else if(ch == '1') { layar = Restart; }
else if(ch == 'A' && layar != zzz) { LastLayar = layar; layar = zzz; }
else if(ch == 'A' && layar == zzz) { layar = LastLayar; }
else if(ch == '2'){LastLayar = layar; layar = LCD_Restart;}

if(layar == Oven_Run || layar == Oven_Stop) { OvenMenu = true; }
else { OvenMenu = false; }

if(OvenRunning && OvenMenu && ch == '4') { layar = Oven_Running; OvnPup = millis();}
if(!OvenRunning && OvenMenu && ch == '4') { layar = Oven_Stopping; OvnPup = millis();}

if(layar == Oven_Running){OvenState = true;}
else if(layar == Oven_Stopping){OvenState = false;}

if((layar == Oven_Running || layar == Oven_Stopping) && millis() - OvnPup >= OvnPupInv) {
    layar = LastLayar;
}

static unsigned long lastLcdDraw = 0;
if(millis() - lastLcdDraw >= 250 || layar != LcdReset) {
    lastLcdDraw = millis();

    if(layar != LcdReset) { lcd.clear(); }
    if(layar == zzz) { lcd.noBacklight(); }
    else if(layar != zzz) { lcd.backlight(); }

    switch(layar) {
        case Home:

```

```

lcd.setCursor(0,0); lcd.print("Home");

lcd.setCursor(19,0); lcd.print(wipistat);

lcd.setCursor(0, 1); lcd.print("Pilih Menu");

lcd.setCursor(0, 2); lcd.print("Melalui Keypad! ");

break;

case Changing:
#pragma region changing

if (millis() - lastLcdSwitch >= lcdInterval || layar != LcdReset) {

    lastLcdSwitch = millis();

    lcd.clear();

    switch (lcdPage) {

        case 0:

            lcd.setCursor(0,0); lcd.print("Battery"); lcd.setCursor(19,0); lcd.print(wipistat);

            lcd.setCursor(0, 1); lcd.print("Suhu 1: "); lcd.print(C1); lcd.print(" C");

            lcd.setCursor(0, 2); lcd.print("Suhu 2: "); lcd.print(C2); lcd.print(" C");

            break;

        case 1:

            lcd.setCursor(0,0); lcd.print("Oven"); lcd.setCursor(19,0); lcd.print(wipistat);

            if(OvenState){

                lcd.setCursor(0, 1); lcd.print("Suhu 1: "); lcd.print(C3); lcd.print(" C");

                lcd.setCursor(0, 2); lcd.print("Suhu 2: "); lcd.print(C4); lcd.print(" C");

                lcd.setCursor(0, 3); lcd.print("Suhu 3: "); lcd.print(C5); lcd.print(" C");

            } else {lcd.setCursor(0, 1); lcd.print("Oven isn't Running!");}

            break;

        case 2:

            lcd.setCursor(0, 0); lcd.print("DHT21_OVEN "); lcd.setCursor(19,0); lcd.print(wipistat);

            if(OvenState){

                lcd.setCursor(0, 1); lcd.print("Suhu DHT : "); lcd.print(Suhu_DHT_Oven);

                lcd.setCursor(0, 2); lcd.print("Humid DHT : "); lcd.print(Kelembapan_DHT_Oven);

            } else {lcd.setCursor(0, 1); lcd.print("Oven isn't Running!");}

            break;

        case 3:

            lcd.setCursor(0, 0); lcd.print("DHT21_BATTERY "); lcd.setCursor(19,0); lcd.print(wipistat);

```

```

    lcd.setCursor(0, 1); lcd.print("Suhu DHT : "); lcd.print(Suhu_DHT_Battery);
    lcd.setCursor(0, 2); lcd.print("Humid DHT : "); lcd.print(Kelembapan_DHT_Battery);
    break;
case 4:
    lcd.setCursor(0, 0); lcd.print("Panel Utama"); lcd.setCursor(19,0); lcd.print(wipistat);
    lcd.setCursor(0, 1); lcd.print("Volt : "); lcd.print(voltage); lcd.print(" V ");
    lcd.setCursor(0, 2); lcd.print("Current : "); lcd.print(current); lcd.print(" A ");
    lcd.setCursor(0, 2); lcd.print("Power : "); lcd.print(current); lcd.print(" w ");
    break;
case 5:
    lcd.setCursor(0, 0); lcd.print("Radiasi"); lcd.setCursor(19,0); lcd.print(wipistat);
    lcd.setCursor(0,1); lcd.print("Daya : "); lcd.print(DayaIna); lcd.print(" W ");
    lcd.setCursor(0,2); lcd.print("Bus Voltage"); lcd.print("V"); lcd.print(Current_mA); lcd.print(" A ");
    lcd.setCursor(0,3); lcd.print("P/A : "); lcd.print(Solar_Radiation); lcd.print(" W/m^2");
    break;
}
lcdPage++;
if (lcdPage >= totalLcdPages) { lcdPage = 0; }
}
#pragma endregion
break;

case Suhu_Battery:
    lcd.setCursor(0,0); lcd.print("Battery"); lcd.setCursor(19,0); lcd.print(wipistat);
    lcd.setCursor(0, 1); lcd.print("Suhu 1: "); lcd.print(C1); lcd.print(" C");
    lcd.setCursor(0, 2); lcd.print("Suhu 2: "); lcd.print(C2); lcd.print(" C");
    break;

case Suhu_Oven:
    lcd.setCursor(0,0); lcd.print("Oven"); lcd.setCursor(19,0); lcd.print(wipistat);
    if(OvenState){
        lcd.setCursor(0, 1); lcd.print("Suhu 1: "); lcd.print(C3); lcd.print(" C");
        lcd.setCursor(0, 2); lcd.print("Suhu 2: "); lcd.print(C4); lcd.print(" C");
        lcd.setCursor(0, 3); lcd.print("Suhu 3: "); lcd.print(C5); lcd.print(" C");
    }

```

```

    }else{lcd.setCursor(0, 1); lcd.print("Oven isn't Running!");}
    break;

```

```

case DHT21_OVEN:

```

```

    lcd.setCursor(0, 0); lcd.print("DHT21_OVEN "); lcd.setCursor(19,0); lcd.print(wipistat);
    if(OvenState){
        lcd.setCursor(0, 1); lcd.print("Suhu DHT : "); lcd.print(Suhu_DHT_Oven);
        lcd.setCursor(0, 2); lcd.print("Humid DHT : "); lcd.print(Kelembapan_DHT_Oven);
    }else{lcd.setCursor(0, 1); lcd.print("Oven isn't Running!");}
    break;

```

```

case DHT21_BATTERY:

```

```

    lcd.setCursor(0, 0); lcd.print("DHT21_BATTERY "); lcd.setCursor(19,0); lcd.print(wipistat);
    lcd.setCursor(0, 1); lcd.print("Suhu DHT : "); lcd.print(Suhu_DHT_Battery);
    lcd.setCursor(0, 2); lcd.print("Humid DHT : "); lcd.print(Kelembapan_DHT_Battery);
    break;

```

```

case Panel_Utama:

```

```

    lcd.setCursor(0, 0); lcd.print("Panel Utama"); lcd.setCursor(19,0); lcd.print(wipistat);
    lcd.setCursor(0, 1); lcd.print("Volt : "); lcd.print(voltage); lcd.print(" V ");
    lcd.setCursor(0, 2); lcd.print("Current : "); lcd.print(current); lcd.print(" A ");
    break;

```

```

case Radiasi:

```

```

    lcd.setCursor(0, 0); lcd.print("Radiasi"); lcd.setCursor(19,0); lcd.print(wipistat);
    lcd.setCursor(0,1); lcd.print("Daya : "); lcd.print(DayaIna); lcd.print(" W ");
    lcd.setCursor(0,2); lcd.print("Bus Voltage"); lcd.print("V|"); lcd.print(Current_mA); lcd.print(" A ");
    lcd.setCursor(0,3); lcd.print("P/A : "); lcd.print(Solar_Radiation); lcd.print(" W/m^2");
    break;

```

```

case Oven_Run:

```

```

    lcd.setCursor(0,0); lcd.print("Run Oven?");
    lcd.setCursor(0,1); lcd.print("C to Run");
    lcd.setCursor(0,2); lcd.print("0 Back");
    OvenRunning = true;

```

```

        break;

    case Oven_Stop:
        lcd.setCursor(0,0); lcd.print("Stop Oven?");
        lcd.setCursor(0,1); lcd.print("C to Stop");
        lcd.setCursor(0,2); lcd.print("0 Back");
        OvenRunning = false;
        break;

    case Oven_Running:
        lcd.setCursor(0,0); lcd.print("Oven is Running!");
        break;

    case Oven_Stopping:
        lcd.setCursor(0,0); lcd.print("Oven Stop Running!");
        break;

    case Restart:
        lcd.setCursor(0,0); lcd.print("Restarting..");
        delay(1000);
        ESP.restart();
        break;

    case LCD_Restart:
        lcd.init();
        lcd.backlight();
        lcd.clear();
        layar = LastLayar;
        break;

    case zzz:
        break;
}
LcdReset = layar;
}

```

```

}

void SerialMonitor(){
  Serial.print("Suhu Battery = ");
  Serial.print(C1);
  Serial.print("/");
  Serial.print(C2);
  Serial.print("/ Avg = ");
  Serial.println(Avg_Bat);
  Serial.print("Suhu Oven = ");
  Serial.print(C3);
  Serial.print("/");
  Serial.print(C4);
  Serial.print("/");
  Serial.print(C5);
  Serial.print("/ Avg = ");
  Serial.println(Avg_Ov);
  Serial.print("From Arduino // Voltage: ");
  Serial.print(voltage, 2);
  Serial.print(" V | Current: ");
  Serial.print(current, 2);
  Serial.print(" A | Humid : ");
  Serial.print(Kelembapan_DHT_Oven);
  Serial.print(" | Suhu DHT : ");
  Serial.print(Suhu_DHT_Oven);
  Serial.println(" C");
  Serial.print("RSSI : ");
  Serial.println(rssi);
}

```

```
#pragma endregion
```

```
#pragma endregion
```

Lampiran 1. 2 Program Arduino nano

```
#include <Arduino.h>

#include <ACS712.h>
#include <DHT.h>
#include <SoftwareSerial.h>

#pragma region Declaration
#define Vin A0

#define BDC 5
#define BAC 6
#define RX 8
#define TX 7

#define ACS_Pin A1
ACS712 ACS(ACS_Pin, 5.0, 1023, 100);

SoftwareSerial Serial2(RX, TX);

#define DHTPIN1 A2
#define DHTPIN2 A3
#define DHTTYPE DHT21

DHT dht(DHTPIN1, DHTTYPE);
DHT dht1(DHTPIN2, DHTTYPE);

//UART

float C1 = 0;
float C2 = 0;
float C3 = 0;

//Read Sens

int WriteInterval = 500;
unsigned long Wrtm = 0;
```

```

int ReadInterval = 200;
unsigned long Rdtm = 0;

//Read avg Voltage
int Voltage_Interval = 2;
unsigned long Vltm = 0;
long voltTotal = 0;
int voltSampleCount = 0;
const int VoltSamples = 100; //

//Read avg Current
int Current_Interval = 1;
unsigned long Cltm = 0;
long CurrentTotal = 0;
int CurrentSampleCount = 0;
const int CurrentSamples = 200; //

float mA = 0;
float h = 0;
float t = 0;
float h1 = 0;
float t1 = 0;
float avgAdc = 0;
double Vp = 0;
double Vreal = 0;

//Kontrol
float SetPoint = 45;
float Hysteresis = 2;
int ControlInterval = 500;
unsigned long Cttn = 0;
bool BlowerAC = false;
float HighHyst;
float LowHyst;
int OvenSerial = 0;

```

```

bool OvenState = 0;

//BLOWER

unsigned long Idop = 60000; // 1 menit = 60000
float TimerIdop = 3600000; //1 jam = 3600000
unsigned long LagiAktip = 0;
unsigned long LagiAktipJugo = 0;
bool BoolBombong = true;
bool Pakum = false;

#pragma endregion

#pragma region prototype
void Read_DHT();
void Read_Current();
void Read_Voltage();
void Transmit();
void Receive();
void Blower_DC();
void Blower_AC();
void Vakum();
void Serialprint();
#pragma endregion

void setup() {
  Serial.begin(9600);
  Serial2.begin(9600);
  pinMode(Vin, INPUT);
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(BAC, OUTPUT);
  pinMode(BDC, OUTPUT);
  digitalWrite(BDC, LOW);
  digitalWrite(BAC, LOW);
  Serial.println("Mulai");

  ACS.setMidPoint(510);

```

```

dht.begin();
dht1.begin();

HighHyst = SetPoint + Hysteresis;
LowHyst = SetPoint - Hysteresis;

digitalWrite(BDC, LOW);
digitalWrite(BAC, LOW);
}

void loop() {
  Read_Voltage();
  Read_Current();
  Receive();
  Blower_AC();
  Blower_DC();

  //Oven Debug
  static unsigned long opentest = 0;
  if(millis()-opentest>=1000){
    opentest = millis();
    Serial.print("Oven Serial : ");
    Serial.println(OvenSerial);
    Serial.print("Oven State: ");
    Serial.println(OvenState);
    Serial.print("BAC: ");
    Serial.println(BAC);
    Serial.print("BDC: ");
    Serial.println(BDC);
  }

  if(millis() - Rdtm >= ReadInterval){
    //Serial.println("Read Sensor");
    Read_DHT();
    //Serialprint();
    Rdtm = millis();
  }
}

```

```

    }

    if(millis() - Wrtm >= WriteInterval){
        //Serial.println("Write to Esp");
        Transmit();
        Wrtm = millis();
    }
}

#pragma region Void
void Read_DHT(){
    h = dht.readHumidity();
    t = dht.readTemperature();
    h1 = dht1.readHumidity();
    t1 = dht1.readTemperature();
    // Serial.println(mA);
}

void Read_Current(){
    if (millis() - Cltm >= Current_Interval) {
        CurrentTotal += ACS.mA_DC();
        CurrentSampleCount++;
        Cltm = millis();

        if (CurrentSampleCount >= CurrentSamples) {
            mA = ((CurrentTotal / (float)CurrentSamples)/1000)*0.975;
            //Serial.println("Current jadi");
            // reset for the next batch
            CurrentTotal = 0;
            CurrentSampleCount = 0;
        }
    }
}

void Read_Voltage(){
    if (millis() - Vltm >= Voltage_Interval) {
        voltTotal += analogRead(Vin);
        voltSampleCount++;
    }
}

```

```

Vltm = millis();

if (voltSampleCount >= VoltSamples) {
    avgAdc = voltTotal / (float)VoltSamples;
    Vp = (avgAdc * 4.98) / 1024.0;
    Vreal = Vp / 0.0484;

    // reset for the next batch
    voltTotal = 0;
    voltSampleCount = 0;
}}}

void Transmit(){
    // Serial.print("Vp/Vreal/I : ");
    // Serial.print(Vp);
    // Serial.print("/");
    Serial2.print(Vreal);
    Serial2.print("/");
    Serial2.print(mA);
    Serial2.print("/");
    Serial2.print(h);
    Serial2.print("/");
    Serial2.print(t);
    Serial2.print("/");
    Serial2.print(h1);
    Serial2.print("/");
    Serial2.println(t1);
}

void Receive(){
    if (Serial2.available()) {
        digitalWrite(LED_BUILTIN, HIGH);
        Serial.println("Mashooookk");
        String data = Serial2.readStringUntil('\n');
        data.trim();

        int c1 = data.indexOf('/');

```

```

int c2 = data.indexOf('/', c1 + 1);
int c3 = data.indexOf('/', c2 + 1);

if (c1 > 0 && c2 > c1 && c3 > c2) {
    Serial.println("Jadeee");
    C1    = data.substring(0, c1).toFloat();
    C2    = data.substring(c1 + 1, c2).toFloat();
    C3    = data.substring(c2 + 1, c3).toFloat();
    OvenSerial = data.substring(c3 + 1).toInt();
    // Serial.print(C1);
    // Serial.print("/");
    // Serial.print(C2);
    // Serial.print("/");
    // Serial.print(C3);

    } }
else{digitalWrite(LED_BUILTIN, LOW);}
if(OvenSerial == 0){OvenState = false;}
else{OvenState = true;}
}

void Blower_DC(){

}

void Blower_AC(){
    static bool Weuhuu = false;
    static unsigned long Wuh = 0;
    static bool bebek = false;
    static bool wasOvenState = false;

    if(!wasOvenState && OvenState){
        Wuh = millis();
    }
    else if(!OvenState){
        Weuhuu = false;
    }
}

```

```

wasOvenState = OvenState;

if(OvenState && millis()-Wuh>=5000){
    Weuhuu = true;
}

if (Weuhuu && t < LowHyst) {
    bebek = true;
}
else if (Weuhuu && t >= HighHyst) {
    bebek = false;
}

if(!Weuhuu){bebek = false;}

digitalWrite(BAC, bebek ? HIGH : LOW);
digitalWrite(BDC, bebek ? HIGH : LOW);

}

void Vakum(){
    if(BoolBombong && millis()-LagiAktip >= TimerIdop){
        BoolBombong = false;
        Pakum = true;
        LagiAktip = millis();
        LagiAktipJugo = millis();
    }
    if(!BoolBombong && millis()-LagiAktipJugo >= Idop){
        Pakum = false;
        BoolBombong = true;
    }

    digitalWrite(BAC, Pakum ? HIGH : LOW);
    // if(millis() - Cttm >= ControlInterval){
    //  if(!BlowerAC && C2 > HighHyst){

```

```

// BlowerAC = true;
// }

// else if (BlowerAC && C2 <= LowHyst){
//   BlowerAC = false;
// }

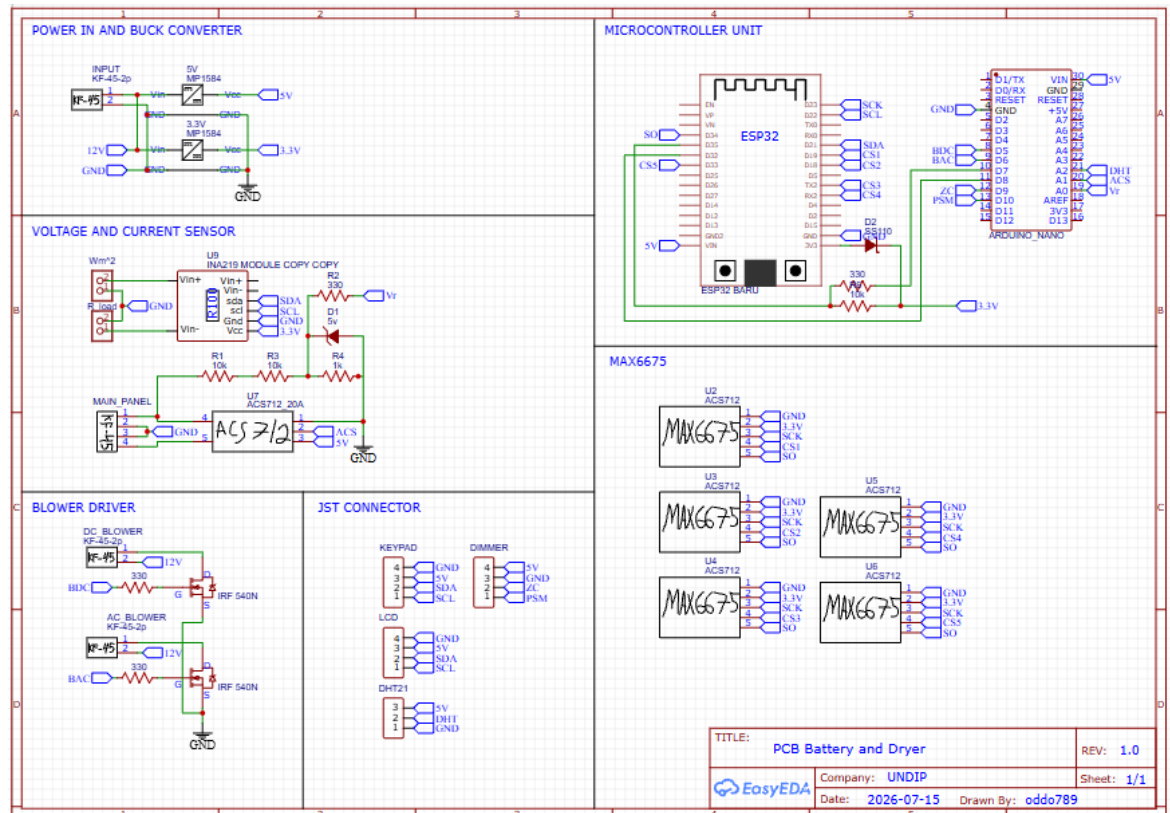
// digitalWrite(BAC, BlowerAC ? HIGH : LOW);
// Cttm = millis();
// }
}

void Serialprint(){
  Serial.print("Arus/Tegangan : ");
  Serial.print(mA);
  Serial.print("/");
  Serial.print(analogRead(A1));
  Serial.print("/");
  Serial.print(Vreal);
  Serial.print("/");
  Serial.println(Vp);
  Serial.print("DHT : ");
  Serial.print(t);
  Serial.print("/");
  Serial.print(h);
  Serial.print("/");
  Serial.print(t1);
  Serial.print("/");
  Serial.println(h1);
}

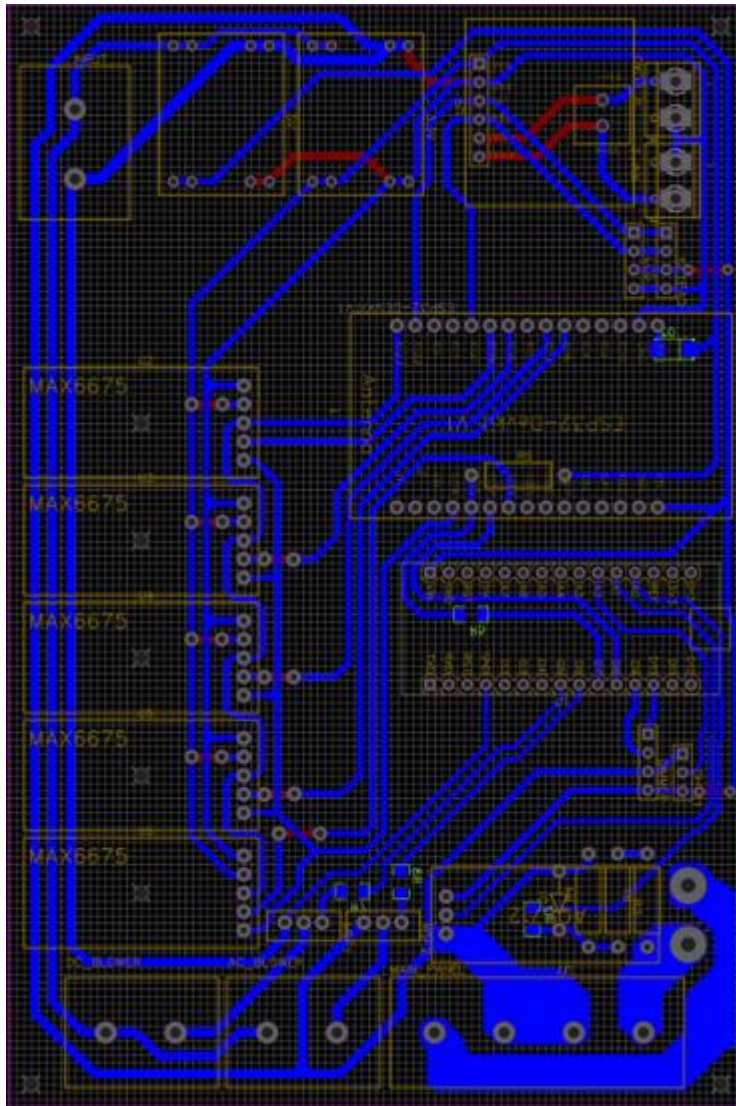
#pragma endregion

```

Lampiran 1. 3 Skematik PCB



Lampiran 1. 4 Desain Layout PCB



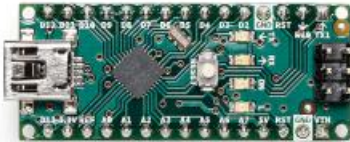
Lampiran 1. 5 Data Penelitian

Lampiran 1. 6 Datasheet Arduino Nano



Arduino® Nano

User Manual
SKU: A000005



Description

Arduino® Nano is an intelligent development board designed for building faster prototypes with the smallest dimension. Arduino Nano being the oldest member of the Nano family, provides enough interfaces for your breadboard-friendly applications. At the heart of the board is **ATmega328 microcontroller** clocked at a frequency of 16 MHz featuring more or less the same functionalities as the Arduino® Duemilanove. The board offers 20 digital input/output pins, 8 analog pins, and a mini-USB port.

Target Areas

Maker, Security, Environmental, Robotics and Control Systems



Features

- **ATmega328** Microcontroller
 - High-performance low-power 8-bit processor
 - Achieve up to 16 MIPS for 16 MHz clock frequency
 - 32 kB of which 2 KB used by bootloader
 - 2 kB internal SRAM
 - 1 kB EEPROM
 - 32 x 8 General Purpose Working Registers
 - Real Time Counter with Separate Oscillator
 - Six PWM Channels
 - Programmable Serial USART
 - Master/Slave SPI Serial Interface
- **Power**
 - Mini-B USB connection
 - 7-15V unregulated external power supply (pin 30)
 - 5V regulated external power supply (pin 27)
- **Sleep Modes**
 - Idle
 - ADC Noise Reduction
 - Power-save
 - Power-down
 - Standby
 - Extended Standby
- **I/O**
 - 20 Digital
 - 8 Analog
 - 6 PWM Output



2 Ratings

2.1 Recommended Operating Conditions

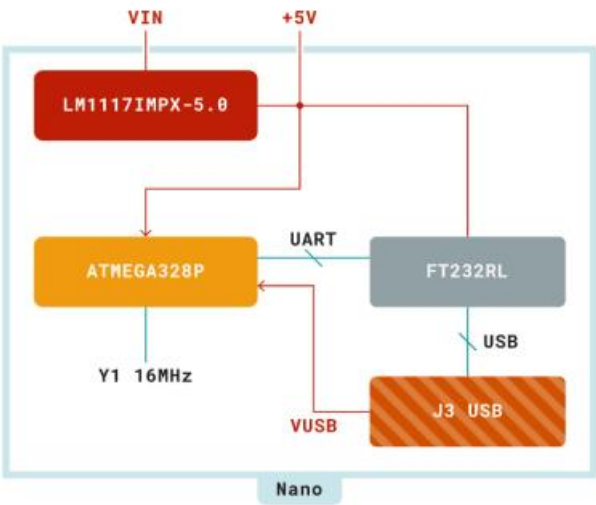
Symbol	Description	Min	Max
	Conservative thermal limits for the whole board:	-40 °C	85 °C

2.2 Power Consumption

Symbol	Description	Min	Typ	Max	Unit
USB VCC	Input supply from USB		TBC		mW
VIN	Input from VIN pad		TBC		mW

3 Functional Overview

3.1 Block Diagram



- | | |
|----------------|--------------------|
| Power | Microcontroller |
| LED | Data Communication |
| Internal Parts | Connectors |

Block Diagram of Arduino Nano



5.1 Analog

Pin	Function	Type	Description
1	+3V3	Power	5V USB Power
2	A0	Analog	Analog input 0 /GPIO
3	A1	Analog	Analog input 1 /GPIO
4	A2	Analog	Analog input 2 /GPIO
5	A3	Analog	Analog input 3 /GPIO
6	A4	Analog	Analog input 4 /GPIO
7	A5	Analog	Analog input 5 /GPIO
8	A6	Analog	Analog input 6 /GPIO
9	A7	Analog	Analog input 7 /GPIO
10	+5V	Power	+5V Power Rail
11	Reset	Reset	Reset
12	GND	Power	Ground
12	VIN	Power	Voltage Input

5.2 Digital

Pin	Function	Type	Description
1	D1/TX1	Digital	Digital Input 1 /GPIO
2	D0/RX0	Digital	Digital Input 0 /GPIO
3	D2	Digital	Digital Input 2 /GPIO
4	D3	Digital	Digital Input 3 /GPIO
5	D4	Digital	Digital Input 4 /GPIO
6	D5	Digital	Digital Input 5 /GPIO
7	D6	Digital	Digital Input 6 /GPIO
8	D7	Digital	Digital Input 7 /GPIO
9	D8	Digital	Digital Input 8 /GPIO
10	D9	Digital	Digital Input 9 /GPIO
11	D10	Digital	Digital Input 10 /GPIO
12	D11	Digital	Digital Input 11 /GPIO
13	D12	Digital	Digital Input 12 /GPIO
14	D13	Digital	Digital Input 13 /GPIO
15	Reset	Reset	Reset
16	GND	Power	Ground

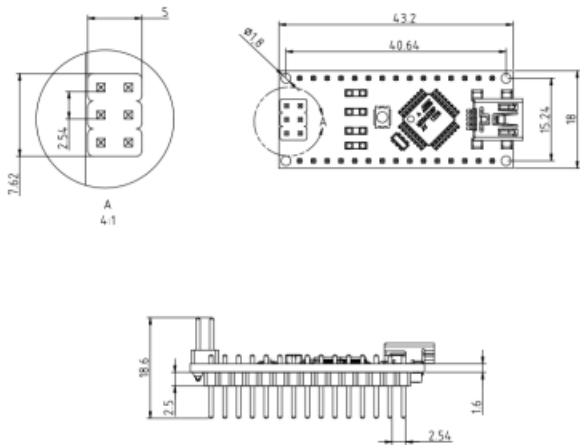


5.3 ATmega328

Pin	Function	Type	Description
1	PB0	Internal	Serial Wire Debug
2	PB1	Internal	Serial Wire Debug
3	PB2	Internal	Serial Wire Debug
4	PB3	Internal	Serial Wire Debug
5	PB4	Internal	Serial Wire Debug
6	PB5	Internal	Serial Wire Debug

6 Mechanical Information

ARDUINO
NANO
Size



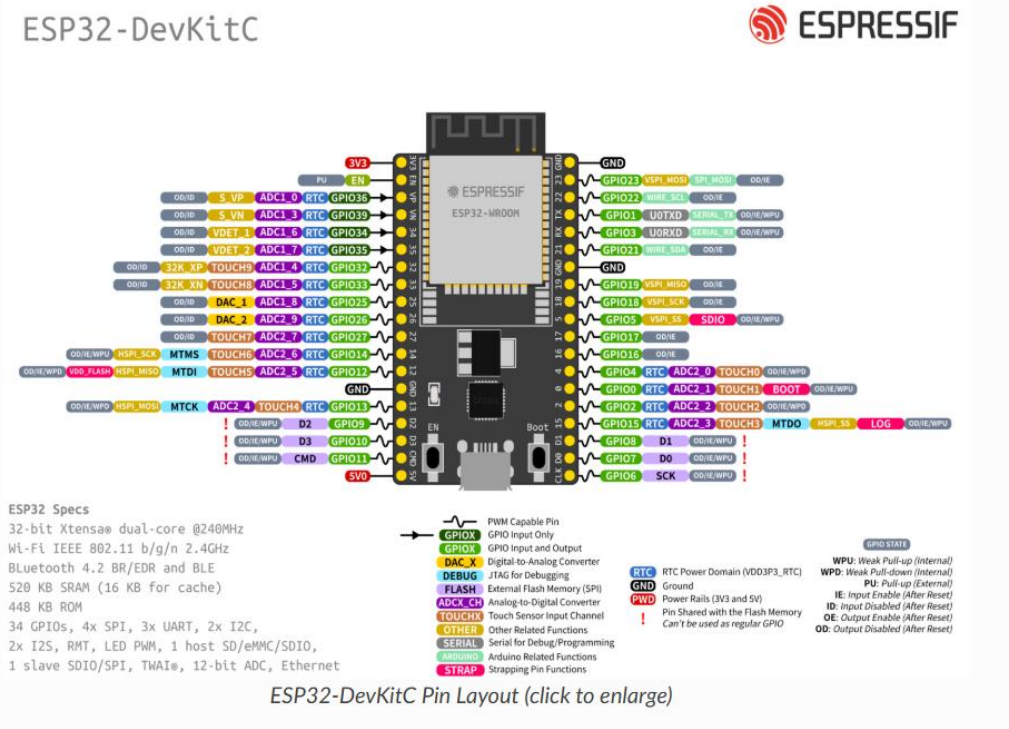
Mechanical dimensions of Arduino Nano

2020/11/19

Lampiran 1. 7 Datasheet ESP32

Key Component	Description
ESP32-WROOM-32	A module with ESP32 at its core. For more information, see ESP32-WROOM-32 Datasheet .
EN	Reset button.
Boot	Download button. Holding down Boot and then pressing EN initiates Firmware Download mode for downloading firmware through the serial port.
USB-to-UART Bridge	Single USB-UART bridge chip provides transfer rates of up to 3 Mbps.
Micro USB Port	USB interface. Power supply for the board as well as the communication interface between a computer and the ESP32-WROOM-32 module.
5V Power On LED	Turns on when the USB or an external 5V power supply is connected to the board. For details see the schematics in Related Documents .
I/O	Most of the pins on the ESP module are broken out to the pin headers on the board. You can program ESP32 to enable multiple functions such as PWM, ADC, DAC, I2C, I2S, SPI, etc.

Pin Layout



Lampiran 1. 8 Datasheet ACS712



ACS712

Fully Integrated, Hall Effect-Based Linear Current Sensor
with 2.1 kVRMS Voltage Isolation and a Low-Resistance Current Conductor

Features and Benefits

- Low-noise analog signal path
- Device bandwidth is set via the new FILTER pin
- 5 μ s output rise time in response to step input current
- 50 kHz bandwidth
- Total output error 1.5% at $T_A = 25^\circ\text{C}$, and 4% at -40°C to 85°C
- Small footprint, low-profile SOIC package
- 1.2 m Ω internal conductor resistance
- 2.1 kV_{RMS} minimum isolation voltage from pins 1-4 to pins 5-8
- 5.0 V, single supply operation
- 66 to 185 mV/A output sensitivity
- Output voltage proportional to AC or DC currents
- Factory-trimmed for accuracy
- Extremely stable output offset voltage
- Nearly zero magnetic hysteresis
- Ratiometric output from supply voltage

Package: 8 pin SOIC (suffix LC)



Approximate Scale 1:1

Description

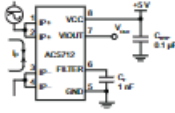
The Allegro[®] ACS712 provides economical and precise solutions for AC or DC current sensing in industrial, automotive, commercial, and communications systems. The device package allows for easy implementation by the customer. Typical applications include motor control, load detection and management, switched-mode power supplies, and overcurrent fault protection.

The device consists of a precise, low-offset, linear Hall sensor circuit with a copper conduction path located near the surface of the die. Applied current flowing through this copper conduction path generates a magnetic field which is sensed by the integrated Hall IC and converted into a proportional voltage. Device accuracy is optimized through the close proximity of the magnetic signal to the Hall transducer. A precise, proportional voltage is provided by the low-offset, chopper-stabilized BiCMOS Hall IC, which is programmed for accuracy after packaging.

The output of the device has a positive slope ($>V_{IOU(TO)}$) when an increasing current flows through the primary copper conduction path (from pins 1 and 2, to pins 3 and 4), which is the path used for current sensing. The internal resistance of this conductive path is 1.2 m Ω typical, providing low power

Continued on the next page...

Typical Application



Application 1. The ACS712 outputs an analog signal, V_{OUT} , that varies linearly with the uni- or bi-directional AC or DC primary sensed current, I_P , within the range specified. C_F is recommended for noise management, with values that depend on the application.

ACS712-08

Lampiran 1. 9 Datasheet INA210



INA210

INA211

INA212

INA213

INA214

www.ti.com

SDOS437 – MAY 2008

Voltage Output, High or Low Side Measurement, Bi-Directional Zero-Drift Series CURRENT SHUNT MONITOR

FEATURES

- WIDE COMMON-MODE RANGE: -0.3V to 26V
- OFFSET VOLTAGE: $\pm 35\mu\text{V}$ (Max, INA210) (Enables shunt drops of 10mV full-scale)
- ACCURACY
 - $\pm 1\%$ Gain Error (Max over temperature)
 - $0.5\mu\text{V}/^\circ\text{C}$ Offset Drift (Max)
 - $10\text{ppm}/^\circ\text{C}$ Gain Drift (Max)
- CHOICE OF GAINS:
 - INA210: 200V/V
 - INA211: 500V/V
 - INA212: 1000V/V
 - INA213: 50V/V
 - INA214: 100V/V
- QUIESCENT CURRENT: 100 μA (max)
- SC70 package

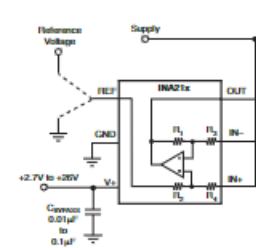
APPLICATIONS

- NOTEBOOK COMPUTERS
- CELL PHONES
- TELECOM EQUIPMENT
- POWER MANAGEMENT
- BATTERY CHARGERS
- WELDING EQUIPMENT

DESCRIPTION

The INA210, INA211, INA212, INA213, and INA214 are voltage output current shunt monitors that can sense drops across shunts at common-mode voltages from -0.3V to 26V, independent of the supply voltage. Five fixed gains are available: 50V/V, 100V/V, 200V/V, 500V/V, or 1000V/V. The low offset of the Zero-Drift architecture enables current sensing with maximum drops across the shunt as low as 10mV full-scale.

These devices operate from a single +2.7V to +26V power supply, drawing a maximum of 100 μA of supply current. All versions are specified over the extended operating temperature range (-40 $^\circ\text{C}$ to +125 $^\circ\text{C}$), and offered in an SC70 package.



PRODUCT	GAIN	R_s and R_L	R_1 and R_2
INA210	200	36 Ω	1M Ω
INA211	500	36 Ω	1M Ω
INA212	1000	16 Ω	1M Ω
INA213	50	20k Ω	1M Ω
INA214	100	10k Ω	1M Ω



Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

All trademarks are the property of their respective owners.

PRODUCTION DATA information is current as of publication date. Products conform to specifications per the terms of the Texas Instruments standard warranty. Production processing does not necessarily include testing of all parameters.

Copyright © 2008, Texas Instruments Incorporated

Lampiran 1. 10 Datasheet MAX6675

EVALUATION KIT AVAILABLE

[Click here to ask about the production status of specific part numbers.](#)

MAX6675

Cold-Junction-Compensated K-Thermocouple-to-Digital Converter (0°C to +1024°C)

General Description

The MAX6675 performs cold-junction compensation and digitizes the signal from a type-K thermocouple. The data is output in a 12-bit resolution, SPI-compatible, read-only format.

This converter resolves temperatures to 0.25°C, allows readings as high as +1024°C, and exhibits thermocouple accuracy of 8 LSHs for temperatures ranging from 0°C to +700°C.

The MAX6675 is available in a small, 8-pin SO package.

Applications

- Industrial
- Appliances
- HVAC

Features

- Direct Digital Conversion of Type -K Thermocouple Output
- Cold-Junction Compensation
- Simple SPI-Compatible Serial Interface
- 12-Bit, 0.25°C Resolution
- Open Thermocouple Detection

Ordering Information

PART	TEMP RANGE	PIN-PACKAGE
MAX6675ISA	-20°C to +85°C	8 SO

Pin Configuration

Typical Application Circuit

Lampiran 1. 11 Datasheet Buck Converter MP1584EN



MP1584
3A, 1.5MHz, 28V
Step-Down Converter

DESCRIPTION

The MP1584 is a high frequency step-down switching regulator with an integrated internal high-side high voltage power MOSFET. It provides 3A output with current mode control for fast loop response and easy compensation.

The wide 4.5V to 28V input range accommodates a variety of step-down applications, including those in an automotive input environment. A 100µA operational quiescent current allows use in battery-powered applications.

High power conversion efficiency over a wide load range is achieved by scaling down the switching frequency at light load condition to reduce the switching and gate driving losses.

The frequency foldback helps prevent inductor current runaway during startup and thermal shutdown provides reliable, fault tolerant operation.

By switching at 1.5MHz, the MP1584 is able to prevent EMI (Electromagnetic Interference) noise problems, such as those found in AM radio and ADSL applications.

The MP1584 is available in a thermally enhanced SOIC3E package.

FEATURES

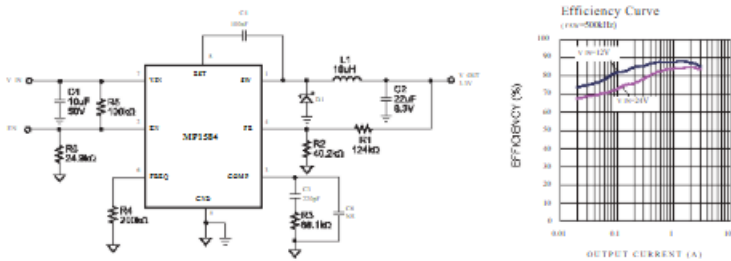
- Wide 4.5V to 28V Operating Input Range
- Programmable Switching Frequency from 100kHz to 1.5MHz
- High-Efficiency Pulse Skipping Mode for Light Load
- Ceramic Capacitor Stable
- Internal Soft-Start
- Internally Set Current Limit without a Current Sensing Resistor
- Available in SOIC3E Package.

APPLICATIONS

- High Voltage Power Conversion
- Automotive Systems
- Industrial Power Systems
- Distributed Power Systems
- Battery Powered Systems

*MPS® and "The Future of Analog IC Technology" are Registered Trademarks of Monolithic Power Systems, Inc.

TYPICAL APPLICATION



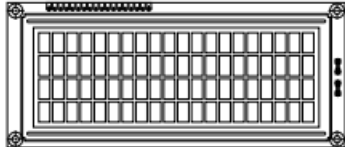
Lampiran 1. 12 Datasheet LCD 20x4



VISHAY
www.vishay.com

LCD-020N004L
Vishay

20 x 4 Character LCD



FEATURES

- Type: Character
- Display format: 20 x 4 characters
- Built-in controller: ST 7066 (or equivalent)
- Duty cycle: 1/16
- 5 x 8 dots includes cursor
- + 5 V power supply (also available for + 3 V)
- LED can be driven by pin 1, pin 2, pin 15, pin 16 or A and K
- N.V. optional for + 3 V power supply
- Material categorization: For definitions of compliance please see www.vishay.com/doc799912



MECHANICAL DATA		
ITEM	STANDARD VALUE	UNIT
Module Dimension	146.0 x 62.5	mm
Viewing Area	123.5 x 43.0	
Dot Size	0.92 x 1.10	
Dot Pitch	0.98 x 1.16	
Mounting Hole	139.0 x 55.5	
Character Size	4.84 x 9.22	

ABSOLUTE MAXIMUM RATINGS						
ITEM	SYMBOL	STANDARD VALUE	MIN.	TYP.	MAX.	UNIT
Power Supply	V _{DD} to V _{SS}	- 0.3	-	-	7.0	V
Input Voltage	V _I	- 0.3	-	-	V _{DD}	V

Note

- V_{SS} = 0 V, V_{DD} = 5.0 V

ELECTRICAL CHARACTERISTICS						
ITEM	SYMBOL	CONDITION	STANDARD VALUE			UNIT
			MIN.	TYP.	MAX.	
Input Voltage	V _{DD}	V _{DD} = + 5 V	4.7	5.0	5.3	V
		V _{DD} = + 3 V	2.7	3.0	5.3	
Supply Current	I _{DD}	V _{DD} = + 5 V	-	8.0	10.0	mA
		+ 20 °C	5.0	5.1	5.7	V
Recommended LC Driving Voltage for Normal Temperature Version Module	V _{DD} to V _O	0 °C	4.6	4.8	5.2	
		25 °C	4.1	4.5	4.7	
		50 °C	3.9	4.2	4.5	
		70 °C	3.7	3.9	4.3	
LED Forward Voltage	V _F	25 °C	-	4.2	4.6	V
LED Forward Current	I _F	25 °C	-	540	1080	mA
EL Power Supply Current	I _{EL}	V _{EL} = 110 V _{AC} , 400 Hz	-	-	5.0	mA

OPTIONS									
PROCESS COLOR						BACKLIGHT			
TN	STN Gray	STN Yellow	STN Blue	FSTN B&W	STN Color	None	LED	EL	CCFL
x	x	x	x	x		x	x	x	

For detailed information, please see the "Product Numbering System" document.

DISPLAY CHARACTER ADDRESS CODE																												
Display Position																												
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20								
DD RAM Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13								
DD RAM Address	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53								
DD RAM Address	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27								
DD RAM Address	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	61	62	63	64	65	66	67								

Revision: 09-Oct-12

1

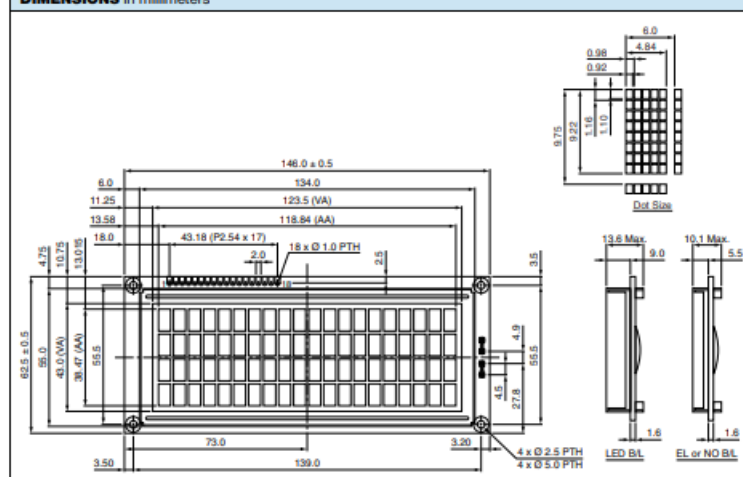
Document Number: 37314

For technical questions, contact: displays@vishay.com

THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE. THE PRODUCTS DESCRIBED HEREIN AND THIS DOCUMENT ARE SUBJECT TO SPECIFIC DISCLAIMERS, SET FORTH AT www.vishay.com/doc791000

INTERFACE PIN FUNCTION		
PIN NO.	SYMBOL	FUNCTION
1	V _{SS}	Ground
2	V _{DD}	+ 3 V or + 5 V
3	V ₀	Contrast adjustment
4	RS	H/L register select signal
5	R/W	H/L read/write signal
6	E	H → L enable signal
7	DB0	H/L data bus line
8	DB1	H/L data bus line
9	DB2	H/L data bus line
10	DB3	H/L data bus line
11	DB4	H/L data bus line
12	DB5	H/L data bus line
13	DB6	H/L data bus line
14	DB7	H/L data bus line
15	A	Power supply for LED (4.2 V)
16	K	Power supply for B/L (0 V)
17	NC/V _{EE}	NC or negative voltage output
18	NC	NC connection

DIMENSIONS in millimeters



Lampiran 1. 13 Datasheet KEYPAD 4x4



Web Site: www.parallax.com
 Forums: forums.parallax.com
 Sales: sales@parallax.com
 Technical: support@parallax.com

Office: (916) 624-8333
 Fax: (916) 624-8003
 Sales: (888) 512-1024
 Tech Support: (888) 997-8267

4x4 Matrix Membrane Keypad (#27899)

This 16-button keypad provides a useful human interface component for microcontroller projects. Convenient adhesive backing provides a simple way to mount the keypad in a variety of applications.

Features

- Ultra-thin design
- Adhesive backing
- Excellent price/performance ratio
- Easy interface to any microcontroller
- Example programs provided for the BASIC Stamp 2 and Propeller P8X32A microcontrollers

Key Specifications

- Maximum Rating: 24 VDC, 30 mA
- Interface: 8-pin access to 4x4 matrix
- Operating temperature: 32 to 122 °F (0 to 50°C)
- Dimensions:
Keypad: 2.7 x 3.0 in (6.9 x 7.6 cm)
Cable: 0.78 x 3.5 in (2.0 x 8.8 cm)

Application Ideas

- Security systems
- Menu selection
- Data entry for embedded systems

