Lampiran Program STM32F103C8 menggunakan Arduino-IDE

```
#include "dht.h"
#include "EmonLib.h"
#include <LiquidCrystal.h>
#define rms_current 200 //CT Spec 200A
#define i_measured (1.414 * rms_current) //root(2) * (rms_current)
#define nb_turns (200 / 5) //ratio CT 200/5A
#define i_sensor (i_measured / nb_turns) //(i_measured) / nb_turns
#define MaxAdcV 3.3
#define max_volt (MaxAdcV / 2)
#define r_burden 0.22 //(max_volt / i_sensor)
#define calib_val ((i_measured / i_sensor) / r_burden)
#define sensorSuhuPin PB9
#define ctT PA0
#define ctS PA1
#define ctR PA2
#define vT PA3
#define vS PA4
#define vR PA5
EnergyMonitor emonR;
EnergyMonitor emonS;
```

```
EnergyMonitor emonT;
dht sensorSuhu;
float Vresult[3], Aresult[3];
LiquidCrystal lcd(PA9, PA8, PB15, PB14, PB13, PB12);
#define DEBUG 1
char buffer[256];
const String WifiSSID = "bismillah";
const String WifiPass = "bismillah";
const String svrAddr = "https://api.thingspeak.com";
const String writeApiKey = "66ZADTYB6J3JE4BD";
unsigned long last_update_sensor, last_update_server;
void setup()
{
  lcd.begin(16,4);
  lcd.setCursor(0,0);
  lcd.print("Bismillah");
  lcd.setCursor(0,1);
  lcd.print("Bismillah");
  Serial.begin(9600);
  Serial3.begin(115200);
  emonR.voltage(vR, 468.52, 1.7);  // Voltage: input pin, calibration, phase_shift
```

```
  emonR.current(ctR, calib_val);
  emonS.voltage(vS, 468.52, 1.7);
  emonS.current(ctS, calib_val);
  emonT.voltage(vT, 468.52, 1.7);
  emonT.current(ctT, calib_val);
lcd.clear();
char wifi_status = '5';
  do
  {
   delay(5000);
   ESP_Send_Command("AT", "OK", 2000);
   ESP_Send_Command("AT+CWMODE=1", "OK", 2000);
   String cmdtmp = "AT+CWJAP=\"" + WifiSSID + "\",\"" + WifiPass + "\"";
   ESP_Send_Command(cmdtmp, "OK", 10000);
   ESP_Send_Command("AT+CIPSTATUS", "OK", 2000);
   wifi_status = buffer[23];
   Serial.print("WiFi Status: ");
   Serial.println(wifi_status);
  }
  while(wifi_status != '2');
}
```

```
void loop()
{
    emonR.calcVI(20,2000);
    emonS.calcVI(20,2000);
    emonT.calcVI(20,2000);
   Vresult[0] = (Vresult[0] * 0.70) + (emonR.Vrms * 0.30);
    Vresult[1] = (Vresult[1] * 0.70) + (emonS.Vrms * 0.30);
    Vresult[2] = (Vresult[2] * 0.70) + (emonT.Vrms * 0.30);
    Aresult[0] = (Aresult[0] * 0.70) + (emonR.Irms * 0.30);
    Aresult[1] = (Aresult[1] * 0.70) + (emonS.Irms * 0.30);
    Aresult[2] = (Aresult[2] * 0.70) + (emonT.Irms * 0.30);
if(millis() - last_update_sensor > 1000)
  {
    int ret = sensorSuhu.read11(sensorSuhuPin);
    Serial.println(sensorSuhu.temperature,1);
Serial.print(Aresult[0], 0);
    Serial.print("A\t");
    Serial.print(Vresult[0], 0);
    Serial.print("V\t");
    Serial.print(Aresult[1], 0);
```

```
    Serial.print("A\t");
    Serial.print(Vresult[1], 0);
    Serial.print("V\t");
    Serial.print(Aresult[2], 0);
    Serial.print("A\t");
    Serial.println(Vresult[2], 0);
lcd.setCursor(0,0);
    if(Aresult[0] < 2.5)
     lcd.print(0, 0);
    else
     lcd.print(Aresult[0], 0);
    lcd.print("A ");
    lcd.setCursor(6,0);
    if(Aresult[1] < 2.5)
     lcd.print(0, 0);
    else
     lcd.print(Aresult[1], 0);
    lcd.print("A ");
    lcd.setCursor(11,0);
    if(Aresult[2] < 2.5)
     lcd.print(0, 0);
```

```
    else
      lcd.print(Aresult[2], 0);
    lcd.print("A");
lcd.setCursor(0,1);
    if(Vresult[0] < 15)
      lcd.print(0, 0);
    else
      lcd.print(Vresult[0], 0);
    lcd.print("V ");
    lcd.setCursor(6,1);
    if(Vresult[1] < 15)
      lcd.print(0, 0);
    else
      lcd.print(Vresult[1], 0);
    lcd.print("V ");
    lcd.setCursor(11,1);
    if(Vresult[2] < 15)
      lcd.print(0, 0);
    else
      lcd.print(Vresult[2], 0);
    lcd.print("V");
```

```
  lcd.setCursor(0,2);
  lcd.print("Beban: ");
  if(Vresult[0] > 15 && Vresult[1] > 15 && Vresult[2] > 15 && Aresult[0] > 2.5
        && Aresult[1] > 2.5 && Aresult[2] > 2.5)
  {
    float beban = ((Vresult[0] * Aresult[0]) + (Vresult[1] * Aresult[1]) + (Vresult[2]
        * Aresult[2])) / 200000.0 * 100.0;
    lcd.print(beban, 2);
  }
  else
    lcd.print(0, 0);
  lcd.print(" %");

  lcd.setCursor(0,3);
  lcd.print("Suhu: ");
  lcd.print(sensorSuhu.temperature,1);
  lcd.print(" C");

  last_update_sensor = millis();
}
```

```
if(millis() - last_update_server > 15000)
  {
    ESP_Send_Command("AT+CIPMUX=0", "OK", 2000);
    ESP_Send_Command("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",80",
          "OK", 2000);
    String write_req = "GET /update?api_key=" + writeApiKey;
    write_req += "&field1=" + String(sensorSuhu.temperature,1);
    write_req += "&field5=" + String(emonR.Irms,2);
    write_req += "&field6=" + String(emonS.Irms,2);
    write_req += "&field7=" + String(emonT.Irms,2);
    write_req += "&field2=" + String(emonR.Vrms,0);
    write_req += "&field3=" + String(emonS.Vrms,0);
    write_req += "&field4=" + String(emonT.Vrms,0);

    String cmdtmp = "AT+CIPSEND=" + String(write_req.length() + 2);
    ESP_Send_Command(cmdtmp, ">", 5000);
    ESP_Send_Command(write_req, "OK", 5000);
    ESP_Send_Command("AT+CIPCLOSE", "OK", 2000);

    last_update_server = millis();
    //delay(10000);
```

```
  }
}
void ESP_Purge_Serial()
{
  while (Serial3.available())
    Serial3.read();
}

byte ESP_Send_Command(const String cmd, const char* expected, unsigned int
        timeout)
{
  if (cmd)
  {
    ESP_Purge_Serial();
#if DEBUG
    Serial.print('>');
    Serial.println(cmd);
#endif
    Serial3.println(cmd);
  }
  uint32_t t = millis();
```

```
 byte n = 0;
 do
 {
  if (Serial3.available())
  {
   char c = Serial3.read();
   if (n >= sizeof(buffer) - 1)
   {
    // buffer full, discard first half
    n = sizeof(buffer) / 2 - 1;
    memcpy(buffer, buffer + sizeof(buffer) / 2, n);
   }
   buffer[n++] = c;
   buffer[n] = 0;
   if (strstr(buffer, expected ? expected : "OK\r"))
   {
#if DEBUG
    Serial.print("[1]");
    Serial.println(buffer);
#endif
    return n;
```

```
      }
    }
  }
  while (millis() - t < timeout);
#if DEBUG
   Serial.print("[0]");
   Serial.println(buffer);
#endif
  return 0;
}
```

Lampiran Program ESP8266

```
#include <Adafruit_ESP8266.h>

#include <ESP8266WiFi.h>

#include <WiFiClient.h>

#include <ESP8266WebServer.h>

#include <ESP8266HTTPClient.h>

const char *ssid = "bismillah";

const char *password = "bismillah";

const char tail = '\n', header = '$';

String payload, url;

char cmd_idx;

const char *host = "127.0.0.1";

#define cmd_erase 'E'

#define cmd_write 'S'

#define cmd_read 'G'

#define cmd_con_wifi 'W'

#define cmd_req 'R'

#define cmd_status 'F'

#define cmd_host 'H'

#define cmd_read_con 'O'
```

```
void send_resp(bool statusFlag)
{
  if(statusFlag)
    Serial.println("$F,1");
  else
    Serial.println("$F,0");
}
bool write_to_server(String db, int len, String param, String val)
{
  HTTPClient http;    //Declare object of class HTTPClient

  String postData, getData;
  bool statusFlag = false;

  if(WiFi.status() == WL_CONNECTED)
  {
    postData = "dbtable=";
    postData += db;
    postData += "&length=";
    postData += String(len);
    postData += "&parameter=";
```

```
    postData += param;
    postData += "&value=";
    postData += val;
    http.begin("http://monitoring.simontraline.com/updatedatakwh.php");
    http.addHeader("Content-type", "application/x-www-form-urlencoded");
    int httpCode = http.POST(postData);
    String payload = http.getString();
  http.end();  //Close connection

    if(payload.indexOf("OK") > 0)
      statusFlag = true;
  }
  return statusFlag;
}

bool write_to_database()
{
  char separator = ',';
  byte start_idx, end_idx;
  String param = "", nilai = "";
```

```
start_idx = payload.indexOf(separator);
end_idx = payload.indexOf(separator, start_idx + 1);
String db = payload.substring(start_idx + 1, end_idx);
start_idx = end_idx;
end_idx = payload.indexOf(separator, start_idx + 1);
int len = payload.substring(start_idx + 1, end_idx).toInt();

for(int i=0; i<len; i++)
{
 start_idx = end_idx;
 end_idx = payload.indexOf(separator, start_idx + 1);
 param += payload.substring(start_idx + 1, end_idx);
 if(i + 1 < len)
 {
  separator = ',';
  param += ",";
 }
 else
  separator = tail;
 start_idx = end_idx;
 end_idx = payload.indexOf(separator, start_idx + 1);
```

```
    nilai += payload.substring(start_idx + 1, end_idx);
    if(i + 1 < len)
      nilai += ",";
  }

  if(!write_to_server(db, len, param, nilai))
    return false;

  return true;
}

bool wait_data_sensor(unsigned long timeout)
{
  unsigned long last_millis;
  bool header_flag;
  Serial.println("$R;");
  last_millis = millis();
  while(millis() - last_millis < timeout)
  {
    if(Serial.available())
    {
```

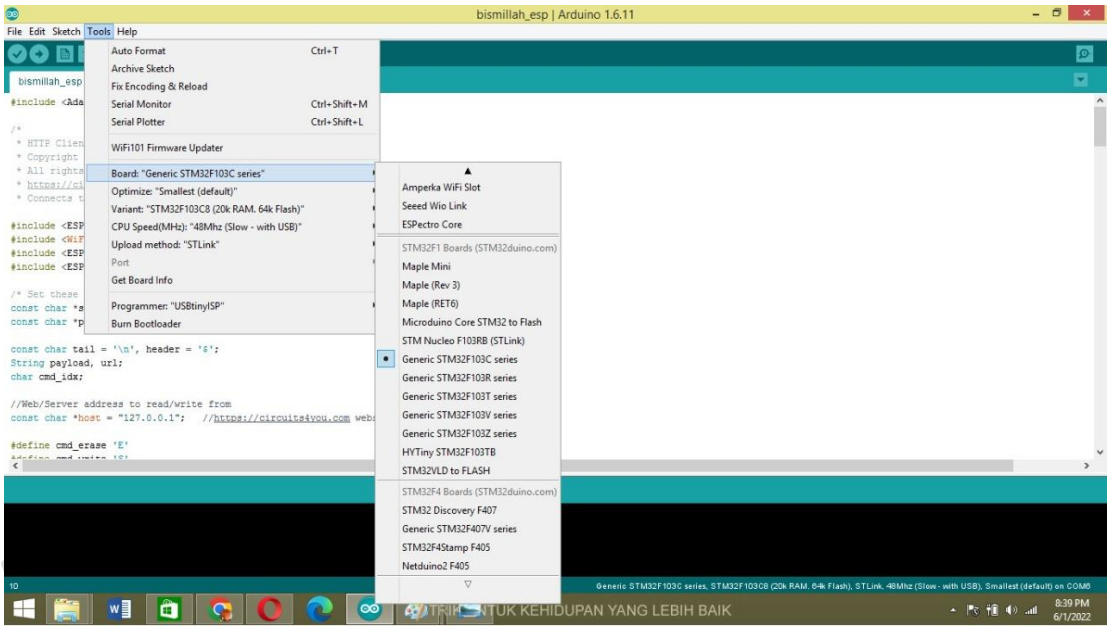
```
    char c = Serial.read();
    if(c == header)
    {
      last_millis = millis();
      header_flag = true;
      payload = "";
    }
    else if(c == tail && header_flag == true)
    {
      payload += c;
      cmd_idx = payload[0];
      return true;
    }
    else if(header_flag == true)
      payload += c;
   }
  }
  return false;
}
void setup() {
  delay(1000);
```

```
  Serial.begin(115200);
  WiFi.mode(WIFI_OFF);        //Prevents reconnection issue (taking too long to
        connect)
  delay(1000);
  WiFi.mode(WIFI_STA);        //This line hides the viewing of ESP as wifi hotspot
  WiFi.begin(ssid, password);     //Connect to your WiFi router
  Serial.println("");
Serial.print("Connecting");
  while (WiFi.status() != WL_CONNECTED)
  {
   delay(500);
   Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
}
void loop()
{
```

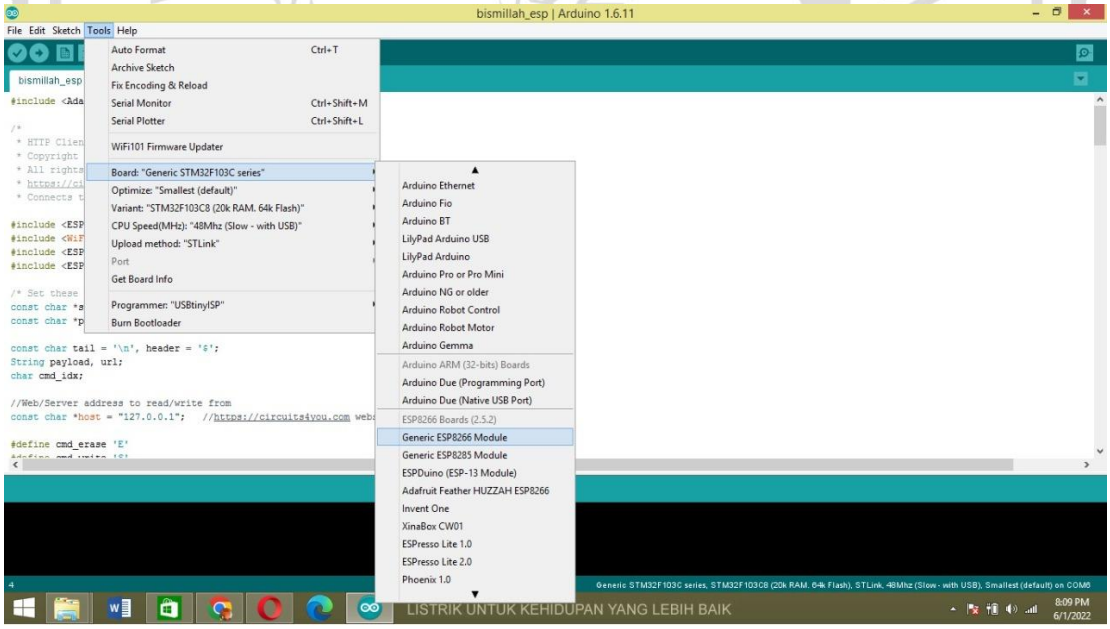
```
if(wait_data_sensor(1000))
{
 switch(cmd_idx)
 {
  case cmd_write:
   send_resp(write_to_database());
   break;
  case cmd_erase:
   //send_resp(erase_database());
   break;
  case cmd_con_wifi:
   //send_resp(wifi_config());
   //read_ssid_pass();
   break;
  case cmd_read:
   //read_database();
   break;
  case cmd_host:
   //send_resp(host_config());
   //read_host();
   break;
```

```
      case cmd_read_con:
        //read_ssid_pass();
        //read_host();
        //Serial.print("$O,");
        //Serial.print(host); Serial.print(",");
        //Serial.print(ssid); Serial.print(",");
        //Serial.println(pass);
        break;
      default:
        break;
    }
  }
}
```

Foto hasil dan proses pengerjaan



Proses inisiasi board STM32F103 pada Arduino ide



Proses inisiasi board esp8266 pada Arduino ide

Proses pemasangan alat pada Panel PHB-TR Trafo distribusi