DAFTAR PUSTAKA

ADS1293 Low-Power AFE for Biopotential Measurements. (2013). www.ti.com

Alexander Maier, Andrew Sharp, & Yuriy Vagapov. (2017). *Comparative analysis and practical implementation of the ESP32 microcontroller module for the Internet of Things*.

Anggraini, N. Y., Kep, S., Kep, M., & Leniwita, N. H. (2020). *MODUL KEPERAWATAN MEDIKAL BEDAH I* (Y. Anggraini & H. Leniwita, Eds.). http://repository.uki.ac.id/2744/1/MODULKEPERAWATANMEDIKALBEDAHIBuku1.pdf

Aulia, T. (2021, May). *Belajar Software Ubidots Untuk IoT Enthusiast!* kmtech.id/post/belajar-software-ubidots-untuk-iot-enthusiast

Blanc, Y., & Dimanico, U. (2014). Electrode Placement in Surface Electromyography (sEMG) ”Minimal Crosstalk Area“ (MCA). *The Open Rehabilitation Journal*, *3*(1), 110–126. https://doi.org/10.2174/1874943701003010110

By AllDataSheetCom, P. (n.d.). *ILI9488 a-Si TFT LCD Single Chip Driver 320(RGB) x 480 Resolution, 16.7M-color With Internal GRAM Specification*. http://www.ilitek.com

Cahyono, A. (2016). *RANCANG BANGUN MODUL EKG (ELEKTROKARDIOGRAM) MENGGUNAKAN RASPBERRY* [Teknologi Nasional Malang]. http://eprints.itn.ac.id/4257/1/SKRIPSI%20FULL%20%28%20BAB%20I%20-%20BAB%20V%20%29.pdf

Efendi, Y. (2018). *INTERNET OF THINGS* (IOT) SISTEM PENGENDALIAN LAMPU MENGGUNAKAN RASPBERRY PI BERBASIS MOBILE. *Jurnal Ilmiah Ilmu Komputer*, *4*(1). https://doi.org/https://doi.org/10.35329/jiik.v4i2.41

Fianti, S. (2017). *TRANSISTOR FILM TIPIS ORGANIK* [Universitas Negeri Semarang]. https://text-id.123dok.com/document/myj5nl56q-prinsip-kerja-monitor-lcd-tft-thin-film-transistor-transistor-film-tipis-organik.html

Goetzberger, A. (Adolf), & Hoffmann, V. U. (2005). *Photovoltaic solar energy generation*. Springer. http://fulviofrisone.com/attachments/article/455/Springer-Verlag%20Photovoltaic%20Solar%20Energy%20Generation.pdf

Halomoan, J. (2013, June 15). Analisa Sinyal EKG dengan Metoda HRV (Heart Rate Variability) pada Domain Waktu Aktivitas Berdiri dan Terlentang. *Seminar Nasional Aplikasi Teknologi Informasi (SNATI)*, 29–35. https://www.semanticscholar.org/paper/Analisa-Sinyal-EKG-dengan-Metoda-HRV-(Heart-Rate-Halomoan/8435ee80a5cb7bf317d3152f0f467f13bcfbebd2

Hariyanto, T., Laila Fajriah, U., & Rahayu, M. (2020). PERANCANGAN DAN REALISASI SISTEM TELEMONITORING AKTIVITAS BIOELEKTRIK JANTUNG DAN KADAR SATURASI OKSIGEN DALAM DARAH PADA PASIEN BERBASIS IOT. *Jurnal Pendidikan Teknologi Dan Kejuruan*, *17*(2).

Programmable Resolution1-Wire Digital Thermometer. (2019). https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

Putri, A. F., & Widiantoro, A. (2020). Monitoring Ekg (Elektrokardiograf) Berbasis Mikrokontroller Dan Pemrograman Delphi 7.0. *JURNAL TEKNIK ELEKTRO DAN KOMPUTER TRIAC*, *7*(1). https://doi.org/https://doi.org/10.21107/triac.v7i1.7196

Rachmat, H. H., & Ambaransari, D. R. (2018). Sistem Perekam Detak Jantung Berbasis Pulse Heart Rate Sensor pada Jari Tangan. *ELKOMIKA: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, & Teknik Elektronika*, *6*(3), 344. https://doi.org/10.26760/elkomika.v6i3.344

Rifali, M., & Irmawati, D. (2019). Sistem Cerdas Deteksi Sinyal Elektrokardiogram (EKG) untuk Klasifikasi Jantung Normal dan Abnormal Menggunakan Jaringan Syaraf Tiruan (JST). *Elinvo (Electronics, Informatics, and Vocational Education)*, *4*(1), 49–55. https://doi.org/10.21831/elinvo.v4i1.28242

Rifky, I. (2021, November 16). *Sejarah Mikrokontroler esp32*. https://raharja.ac.id/2021/11/16/mikrokontroler-esp32-2/

Sinauarduino. (2016, March 16). *Arduino IDE*.

https://www.sinauarduino.com/artikel/mengenal-arduino-software-ide/

Suci, D. F. A. (2018). *RANCANG BANGUN ALAT MONITORING DENYUT NADI DAN SUHU TUBUH DENGAN VISUALISASI LCD BERBASIS ARDUINO UNO* [Universtias Jember]. http://repository.unej.ac.id/handle/123456789/89394

Wijaksana Isma, T., Yuliza, M., Angraini, T., & Susanti, R. (2020). Efektifitas Sensor Elektrokardiograf (EKG) AD8232 Untuk Mendeteksi Kelelahan Pada Saat Penggunaan SMARTPHONE. *Elektron Jurnal Ilmiah*, *12*. https://jie.pnp.ac.id/index.php/jie/article/download/148/107/

Yuri, & Joel. (2018). *Open Hardware*. https://pulsesensor.com/pages/open-hardware

Zhang, J. X. J., & Hoshino, K. (2008). *Surface Electrode Implantable and wearable sensors*. https://www.sciencedirect.com/topics/engineering/surface-electrode/pdf

LAMPIRAN

Lampiran 1 Program Arduino untuk alat 1

```
#include <SPI.h>
#include <Wire.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include "BluetoothSerial.h"
#include <TFT_eSPI.h>
#define LED_BUILTIN 2 //pin with LED to turn on when BT connected

#define ONE_WIRE_BUS 5
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
DeviceAddress tempDeviceAddress;
#define Sensor 36
#define UpperThreshold 2120
#define LowerThreshold 1920



int  resolution = 12;
unsigned long lastTempRequest = 0;
int  delayInMillis = 0;
float temperature = 0.0;

TFT_eSPI tft = TFT_eSPI();
SPIClass hspi(HSPI);

boolean BT_cnx=false;
BluetoothSerial ESP_BT;
```

```
uint16_t LastTime=0;
uint16_t ThisTime;
bool BPMTiming=false;
bool BeatComplete=false;
int BPM=0;

const int gridsize = 90;
unsigned int lifeCount = 0;

int32_t c1;
int maxLife_1 = 0;
int minLife_1 = 10000000;
int maxLife_2 = 0;
int minLife_2 = 10000000;
int graphData_1[161];
int graphData_2[161];
float scaling_1 = 1;


void Callback(esp_spp_cb_event_t event, esp_spp_cb_param_t *param){
  if(event == ESP_SPP_SRV_OPEN_EVT){
    Serial.println("Client Connected");
    digitalWrite(LED_BUILTIN, HIGH);
    BT_cnx = true;
  }
  if(event == ESP_SPP_CLOSE_EVT ){
    Serial.println("Client disconnected");
```

```
    digitalWrite(LED_BUILTIN, LOW);
    BT_cnx = false;
  }
}

void setup() {

  Serial.begin(115200);
    tft.setRotation(1);     //Lanscape
    tft.begin();
    tft.fillScreen(TFT_BLACK);
    Home();
    tft.fillScreen(TFT_BLACK);
    tft.setTextColor(TFT_GREEN, TFT_BLACK);
    tft.fillRect(120,0,10,320,TFT_BLUE);
    tft.fillRect(0,275,120,5,TFT_BLUE);

  ESP_BT.register_callback(Callback);
  if(!ESP_BT.begin("WECG_1")){
    Serial.println("An error occurred initializing Bluetooth");
  }else{
    Serial.println("Bluetooth initialized... Bluetooth Device is Ready to Pair...");
  }
    sensors.begin();
    sensors.getAddress(tempDeviceAddress, 0);
    sensors.setResolution(tempDeviceAddress, resolution);
    sensors.setWaitForConversion(false);
    sensors.requestTemperatures();
```

```
    delayInMillis = 750 / (1 << (12 - resolution));
    lastTempRequest = millis();

  for(int i = 0; i < 176; i++) {
    graphData_1[i] = 0; }
  for(int i = 0; i < 176; i++) {
    graphData_2[i] = 0; }

  tft.setTextColor(TFT_BLACK);
}

void loop() {
  drawgraph_1();
  Layar();
  Suhu();
  bluetooth();
  tft.fillRect(130,0,5,320,TFT_BLACK);
}

void drawgraph_1(){
int value = analogRead(Sensor);
c1 = value;
graphData_1[160] = int(c1);///5.0);  //150 data points initially all 0.
Serial.println(graphData_1[160]);

  for(int i = 0; i <= 159; i++){
    tft.drawLine(130+i*2, 200+(minLife_1-graphData_1[i-1])/scaling_1,
130+i*2+2, 200+(minLife_1-graphData_1[i])/scaling_1,TFT_BLACK);
```

```
  }
   maxLife_1 = 0;
   minLife_1 = 10000000;
  for(int i = 0; i <= 159; i++){
      if(graphData_1[i] > maxLife_1)
      maxLife_1 = graphData_1[i];
    if(graphData_1[i] < minLife_1)
      minLife_1 = graphData_1[i];
  }
  for(int i = 0; i <= 159; i++){//shift graph
  graphData_1[i] = graphData_1[i+1];
  }
  int maxPixelsAllowed = 80; //will scale the graph if it's over this...
  if(  (maxLife_1 - minLife_1) >= maxPixelsAllowed){
    scaling_1 = (maxLife_1 - minLife_1)/(int(maxPixelsAllowed));
  }
    for(int i = 0; i <= 159; i++){
   tft.drawLine(130+i*2, 200+(minLife_1-graphData_1[i-1])/scaling_1,
130+i*2+2, 200+(minLife_1-graphData_1[i])/scaling_1,TFT_GREEN);
  }
}


void Layar() {
int value = analogRead(Sensor);
c1 = value;
graphData_2[160] = int(c1);///5.0);  //150 data points initially all 0.


for(int i = 0; i <= 159; i++){//shift graph
```

```
  graphData_2[i] = graphData_2[i+1];
  }

  maxLife_2 = 0;
  minLife_2 = 10000000;
  for(int i = 0; i <= 159; i++){
     if(graphData_2[i] > maxLife_2)
     maxLife_2 = graphData_2[i];
    if(graphData_2[i] < minLife_2)
     minLife_2 = graphData_2[i];
  }
ThisTime=millis();
  if(value > maxLife_2-80) {
    if(BeatComplete) {
    BPM= ThisTime-LastTime;
    BPM= int(60/(float(BPM)/1000));
    BPMTiming=false;
    BeatComplete=false;
}
  if(BPMTiming==false) {
    LastTime=millis();
    BPMTiming=true;
  }
}
if((value < maxLife_2-280)&(BPMTiming))
    BeatComplete=true;

tft.setTextColor(TFT_GREEN, TFT_BLACK);
```

```
int padding = tft.textWidth("9999", 6);
tft.setTextPadding(padding);
tft.setTextDatum(TC_DATUM);
tft.drawNumber(BPM,55,120,6);
int padding1 = tft.textWidth("999.99", 4);
tft.setTextPadding(padding1);
tft.setTextDatum(TR_DATUM);
tft.drawFloat(temperature,2,80,290,4);
tft.setTextPadding(0);
tft.drawString("o",90,285,2);
tft.drawString("C",110, 290, 4);

}

void bluetooth(void) {
   if(BT_cnx){
     ESP_BT.print('E');
     ESP_BT.print(c1);
   }
 }

void Home(void) {
 tft.setTextColor(TFT_GREEN, TFT_BLACK);
 tft.setTextPadding(0);
 tft.setTextDatum(TC_DATUM);
 tft.drawString("ECG", 240, 90, 4);
 tft.drawString("WIRELESS", 240, 130, 4);
 for (int16_t t=0; t<480; t += 1) {
```

```
  tft.fillRect(t,200,1,25,TFT_BLUE);
  delay (10);
  }
}

void Suhu(void) {
  // this part keeps the var temperature up to date as possible.
  if (millis() - lastTempRequest >= delayInMillis) // waited long enough??
  {
    temperature = sensors.getTempCByIndex(0);
    sensors.requestTemperatures();
    lastTempRequest = millis();
  }
}
```

Lampiran 2 Program Arduino untuk alat 2

```
#include <TFT_eSPI.h>
#include <SPI.h>
#include <Wire.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include "BluetoothSerial.h"
#define LED_BUILTIN 2 //pin with LED to turn on when BT connected

TFT_eSPI tft = TFT_eSPI();
SPIClass hspi(HSPI);

#define SENSOR 36
#define ONE_WIRE_BUS 5
#define LED_BUILTIN 2 //pin with LED to turn on when BT connected
#define UpperThreshold 2000
#define LowerThreshold 1970

OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
```

```
DeviceAddress tempDeviceAddress;

int  resolution = 12;
unsigned long lastTempRequest = 0;
int  delayInMillis = 0;
float temperature = 0.0;

uint16_t a=130;
uint16_t t1=131;
uint16_t lasta=130;
uint16_t lastb=40;
uint16_t LastTime=0;
uint16_t ThisTime;
bool BPMTiming=false;
bool BeatComplete=false;
uint16_t BPM = 0;
uint16_t lastBPM = 0;

boolean BT_cnx=false;
BluetoothSerial ESP_BT;

#define BLACK   0x0000
#define BLUE    0x001F
#define RED     0xF800
#define GREEN   0x07E0
#define CYAN    0x07FF
#define MAGENTA 0xF81F
#define YELLOW  0xFFE0
#define WHITE   0xFFFF

void Callback(esp_spp_cb_event_t event, esp_spp_cb_param_t *param){
  if(event == ESP_SPP_SRV_OPEN_EVT){
    Serial.println("Client Connected");
    BT_cnx = true;
  }
  if(event == ESP_SPP_CLOSE_EVT ){
    Serial.println("Client disconnected");
    digitalWrite(LED_BUILTIN, LOW);
    BT_cnx = false;
    //ESP.restart();
  }
}


void setup()
{
```

```
Serial.begin(115200);
tft.begin();
ESP_BT.register_callback(Callback);
if(!ESP_BT.begin("ESP32_ECG")){
Serial.println("An error occurred initializing Bluetooth");
}else{
Serial.println("Bluetooth initialized... Bluetooth Device is Ready to Pair...");
}
tft.setRotation(1);     //Portrait
tft.fillScreen(BLACK);
Home();
tft.fillScreen(BLACK);
tft.fillRect(120,0,10,320,BLUE);
tft.fillRect(0,60,120,5,BLUE);
tft.fillRect(0,275,120,5,BLUE);

sensors.begin();
sensors.getAddress(tempDeviceAddress, 0);
sensors.setResolution(tempDeviceAddress, resolution);
sensors.setWaitForConversion(false);
sensors.requestTemperatures();
delayInMillis = 750 / (1 << (12 - resolution));
lastTempRequest = millis();
pinMode(23, OUTPUT);
}

void Suhu(void) {
  // this part keeps the var temperature up to date as possible.
  if (millis() - lastTempRequest >= delayInMillis) // waited long enough??
  {

  digitalWrite(23, LOW);
  temperature = sensors.getTempCByIndex(0);
  sensors.requestTemperatures();
  lastTempRequest = millis();
  }
  digitalWrite(23, HIGH);
}

void loop()
{
Suhu();
Layar();
}
```

```
void Home(void) {
  tft.setTextSize(5);
  tft.setTextColor(GREEN, BLACK);
  tft.setCursor(195,90);
  tft.print("ECG");
  tft.setCursor(120,130);
  tft.print("WIRELESS");
  for (int16_t t=0; t<480; t += 1) {
  tft.fillRect(t,200,1,25,BLUE);
  delay (10);
  }
}

void Layar(void) {
tft.fillRect(90,305,10,8,BLACK);
tft.fillRect(0,100,110,100,BLACK);
int value = analogRead(SENSOR);
Serial.println(value);
if(a>480)
{
t1=131;
a=130;
lasta=a;
}
tft.drawRect(130,0,1,320,BLACK);
ThisTime=millis();
tft.setTextColor(WHITE, BLACK);
int b=1870-(value/1.1111); // Pulse Heart
if (b<40){
b = 40;}
if (b>280){
b = 278; }


tft.drawLine(lasta, lastb, a, b, GREEN);
tft.fillRect(t1,40,4,245,BLACK);

lastb=b;
lasta=a;

if(value>UpperThreshold)
{
if(BeatComplete)
{
BPM=ThisTime-LastTime;
BPM=int(60/(float(BPM)/1000));
```

```
BPMTiming=false;
BeatComplete=false;
}
if(BPMTiming==false)
{
LastTime=millis();
BPMTiming=true;
}
}
if((value<LowerThreshold)&(BPMTiming))
BeatComplete=true;
tft.setCursor(40,10); tft.setTextSize(2);
tft.print("ECG");
tft.setCursor(10,30); tft.setTextSize(2);
tft.print("WIRELESS");
tft.setCursor(5,135); tft.setTextSize(5);
if (BPM < 60)
BPM = lastBPM;
if (BPM > 140)
BPM = 0;
tft.print(BPM);
lastBPM = BPM;
if(BT_cnx){
ESP_BT.print('E'); //make the app Blutooth Graphics (https://play.google.com/store/apps/details?id=com.emrctn.BluetoothGraphics&hl=en_US) work (as specified by the app)
ESP_BT.println(value);
   }
tft.setCursor(0,290); tft.setTextSize(3);
tft.print(temperature);tft.setTextSize(2);
tft.print(char(247));tft.setTextSize(3);
tft.print("C");
a+=2;
t1+=2;
}
```

Lampiran 3 Program Arduino untuk alat 3

```
#include <SPI.h>
#include <Wire.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include "BluetoothSerial.h"
#include <TFT_eSPI.h>
#define LED_BUILTIN 2 //pin with LED to turn on when BT connected
```

```
#define ONE_WIRE_BUS 5
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
DeviceAddress tempDeviceAddress;

int  resolution = 12;
unsigned long lastTempRequest = 0;
int  delayInMillis = 0;
float temperature = 0.0;

TFT_eSPI tft = TFT_eSPI();
SPIClass hspi(HSPI);
static const int spiClk = 1000000; // 1 MHz

const int pin_DRDYB = 4;  // data ready
const int pin_ALARMB = 19; // alarm
const int pin_MISO = 12;  // MISO
const int pin_MOSI = 13;  // MOSI
const int pin_SCLK = 25;  // SCLK
const int pin_SS = 15;    // CSB
boolean BT_cnx=false;
BluetoothSerial ESP_BT;

uint16_t LastTime=0;
uint16_t ThisTime;
bool BPMTiming=false;
bool BeatComplete=false;
uint16_t BPM=0;

const int gridsize = 90;
unsigned int lifeCount = 0;

int32_t c1;
int32_t c2;
int32_t c3;

int maxLife_11 = 0;
int minLife_11 = 10000000;
int maxLife_1 = 0;
int minLife_1 = 10000000;
int maxLife_2 = 0;
int minLife_2 = 10000000;
int maxLife_3 = 0;
int minLife_3 = 10000000;

int graphData_11[90];
```

```
int graphData_1[90];
int graphData_2[90];
int graphData_3[90];
float scaling_1 = 1;
float scaling_2 = 1;
float scaling_3 = 1;

int32_t getValFromChannel(int channel)
{
  byte x1;
  byte x2;
  byte x3;

  switch (channel)
  {
  case 1:
    x1 = 0x37;
    x2 = 0x38;
    x3 = 0x39;
    break;
  case 2:
    x1 = 0x3A;
    x2 = 0x3B;
    x3 = 0x3C;
    break;
  case 3:
    x1 = 0x3D;
    x2 = 0x3E;
    x3 = 0x3F;
    break;
  }
  int32_t val;

  // 3 8-bit registers combination on a 24 bit number
  val = readRegister(x1);
  val = (val << 8) | readRegister(x2);
  val = (val << 8) | readRegister(x3);

  return val;
}

void setup_ECG_2_channel()
{
  writeRegister(0x01, 0x11);
  writeRegister(0x02, 0x19);
  writeRegister(0x0A, 0x07);
```

```
  writeRegister(0x0C, 0x04);
  writeRegister(0x12, 0x04);
  writeRegister(0x14, 0x24);
  writeRegister(0x21, 0x02);
  writeRegister(0x22, 0x02);
  writeRegister(0x23, 0x02);
  writeRegister(0x27, 0x08);
  writeRegister(0x2F, 0x30);
  writeRegister(0x00, 0x01);
}

void setup_ECG_3_channel()
{
  writeRegister(0x01, 0x11);
  writeRegister(0x02, 0x19);
  writeRegister(0x03, 0x2E); //diff
  writeRegister(0x0A, 0x07);
  writeRegister(0x0C, 0x04);
  writeRegister(0x0D, 0x01); //diff
  writeRegister(0x0E, 0x02); //diff
  writeRegister(0x0F, 0x03); //diff
  writeRegister(0x10, 0x01); //diff
  writeRegister(0x12, 0x04);
  writeRegister(0x21, 0x02);
  writeRegister(0x22, 0x02);
  writeRegister(0x23, 0x02);
  writeRegister(0x24, 0x02); //diff
  writeRegister(0x27, 0x08);
  writeRegister(0x2F, 0x70); //diff
  writeRegister(0x00, 0x01);
}

//===========SPECIALIZED SPI OPTION 1
byte readRegister(byte reg)
{
  byte data;
  reg |= 1 << 7;
  hspi.beginTransaction(SPISettings(spiClk, MSBFIRST, SPI_MODE0));
  digitalWrite(pin_SS, LOW);
  hspi.transfer(reg);
  data = hspi.transfer(0);
  digitalWrite(pin_SS, HIGH);
  hspi.endTransaction();
  return data;
}
```

```
void writeRegister(byte reg, byte data)
{
  reg &= ~(1 << 7);
  hspi.beginTransaction(SPISettings(spiClk, MSBFIRST, SPI_MODE0));
  digitalWrite(pin_SS, LOW);
  hspi.transfer(reg);
  hspi.transfer(data);
  digitalWrite(pin_SS, HIGH);
  hspi.endTransaction();
}
//===========SPECIALIZED SPI

void Callback(esp_spp_cb_event_t event, esp_spp_cb_param_t *param){
  if(event == ESP_SPP_SRV_OPEN_EVT){
    Serial.println("Client Connected");
    digitalWrite(LED_BUILTIN, HIGH);
    BT_cnx = true;
  }
  if(event == ESP_SPP_CLOSE_EVT ){
    Serial.println("Client disconnected");
    digitalWrite(LED_BUILTIN, LOW);
    BT_cnx = false;
    //ESP.restart();
  }
}

void setup() {

  pinMode(pin_DRDYB, INPUT);
  pinMode(pin_ALARMB, INPUT);
  pinMode(pin_SS, OUTPUT);

  Serial.begin(115200);
    tft.setRotation(1);     //Lanscape
    tft.begin();
    tft.fillScreen(TFT_BLACK);
    Home();
    tft.fillScreen(TFT_BLACK);
    tft.setTextColor(TFT_GREEN, TFT_BLACK);
    tft.fillRect(120,0,10,320,TFT_BLUE);
    tft.fillRect(0,275,120,5,TFT_BLUE);

  hspi.begin(pin_SCLK, pin_MISO, pin_MOSI, pin_SS);
  ESP_BT.register_callback(Callback);
  if(!ESP_BT.begin("ESP32_ECG")){
    Serial.println("An error occurred initializing Bluetooth");
```

```
  }else{
    Serial.println("Bluetooth initialized... Bluetooth Device is Ready to Pair...");
  }
  setup_ECG_3_channel();
    sensors.begin();
    sensors.getAddress(tempDeviceAddress, 0);
    sensors.setResolution(tempDeviceAddress, resolution);
    sensors.setWaitForConversion(false);
    sensors.requestTemperatures();
    delayInMillis = 750 / (1 << (12 - resolution));
    lastTempRequest = millis();

  for(int i = 0; i < 88; i++) {
    graphData_1[i] = 0; }
  for(int i = 0; i < 88; i++) {
    graphData_2[i] = 0; }
  for(int i = 0; i < 88; i++) {
    graphData_3[i] = 0; }

  tft.setTextColor(TFT_BLACK);
}

void loop() {
  drawgraph_1();
  Layar();
  Suhu();
  bluetooth();
  tft.fillRect(130,0,5,320,TFT_BLACK);
}

void drawgraph_1(){
c1 = getValFromChannel(1);
graphData_1[89] = int(c1); //150 data points initially all 0.
Serial.print(graphData_1[89]);
Serial.print(" ");

  for(int i = 0; i <= 88; i++){
    tft.drawLine(130+i*4, 95+(minLife_1-graphData_1[i-1])/scaling_1,
130+i*4+4, 95+(minLife_1-graphData_1[i])/scaling_1,TFT_BLACK);
  }
   maxLife_1 = 0;
   minLife_1 = 10000000;
  for(int i = 0; i <= 88; i++){
      if(graphData_1[i] > maxLife_1)
      maxLife_1 = graphData_1[i];
    if(graphData_1[i] < minLife_1)
```

```
    minLife_1 = graphData_1[i];
  }
  for(int i = 0; i <= 88; i++){//shift graph
  graphData_1[i] = graphData_1[i+1];
  }
  int maxPixelsAllowed = 90; //will scale the graph if it's over this...
  if(  (maxLife_1 - minLife_1) >= maxPixelsAllowed){
   scaling_1 = (maxLife_1 - minLife_1)/(int(maxPixelsAllowed));
  }
   for(int i = 0; i <= 88; i++){
   tft.drawLine(130+i*4, 95+(minLife_1-graphData_1[i-1])/scaling_1,
130+i*4+4, 95+(minLife_1-graphData_1[i])/scaling_1,TFT_GREEN);
  }

c2 = getValFromChannel(2);
graphData_2[89] = int(c2); //150 data points initially all 0.
Serial.print(graphData_2[89]);
Serial.print(" ");

  for(int i = 0; i <= 88; i++){
   tft.drawLine(130+i*4, 205+(minLife_2-graphData_2[i-1])/scaling_2,
130+i*4+4, 205+(minLife_2-graphData_2[i])/scaling_2,TFT_BLACK);
  }
   maxLife_2 = 0;
   minLife_2 = 10000000;
  for(int i = 0; i <= 88; i++){
     if(graphData_2[i] > maxLife_2)
     maxLife_2 = graphData_2[i];
   if(graphData_2[i] < minLife_2)
     minLife_2 = graphData_2[i];
  }
  for(int i = 0; i <= 88; i++){//shift graph
  graphData_2[i] = graphData_2[i+1];
  }

  if(  (maxLife_2 - minLife_2) >= maxPixelsAllowed){
   scaling_2 = (maxLife_2 - minLife_2)/(int(maxPixelsAllowed));
  }
  for(int i = 0; i <= 88; i++){
   tft.drawLine(130+i*4, 205+(minLife_2-graphData_2[i-1])/scaling_2,
130+i*4+4, 205+(minLife_2-graphData_2[i])/scaling_2,TFT_GREEN);
  }

c3 = getValFromChannel(3);
graphData_3[89] = int(c3); //150 data points initially all 0.
Serial.println(graphData_3[89]);
```

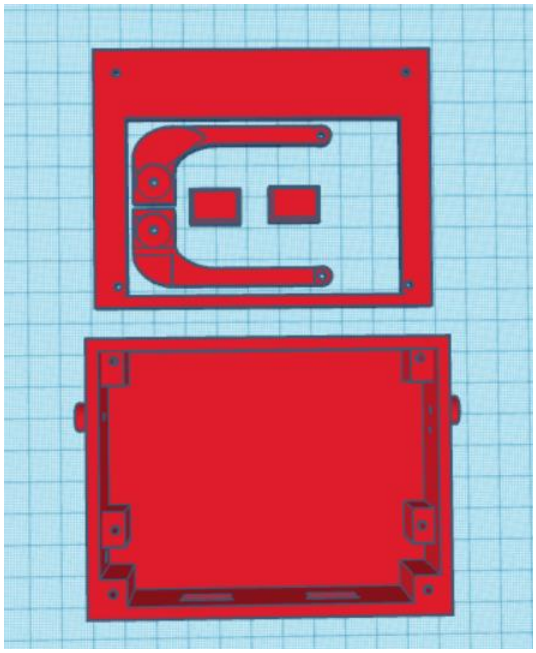
```
  for(int i = 0; i <= 88; i++){
    tft.drawLine(130+i*4, 320+(minLife_3-graphData_3[i-1])/scaling_3,
130+i*4+4, 320+(minLife_3-graphData_3[i])/scaling_3,TFT_BLACK);
  }
maxLife_3 = 0;
minLife_3 = 10000000;
  for(int i = 0; i <= 88; i++){
     if(graphData_3[i] > maxLife_3)
     maxLife_3 = graphData_3[i];
    if(graphData_3[i] < minLife_3)
     minLife_3 = graphData_3[i];
  }
  for(int i = 0; i <= 88; i++){//shift graph
  graphData_3[i] = graphData_3[i+1];
  }
  if(  (maxLife_3 - minLife_3) >= maxPixelsAllowed){
    scaling_3 = (maxLife_3 - minLife_3)/(int(maxPixelsAllowed));
  }
  for(int i = 0; i <= 88; i++){
    tft.drawLine(130+i*4, 320+(minLife_3-graphData_3[i-1])/scaling_3,
130+i*4+4, 320+(minLife_3-graphData_3[i])/scaling_3,TFT_GREEN);
  }
}
void Layar () {
graphData_11[89] = int(c1);

  if(c1>c1 - 2000) {
    if (BeatComplete) {
    BPM=ThisTime-LastTime;
    BPM=int (60/ (float (BPM)/1000));
    BPMTiming=false;
    BeatComplete=false;
}
  If (BPMTiming==false) {
    LastTime=millis();
    BPMTiming=true;
  }
}
If ((c1<c1 + 4000) & (BPMTiming))
    BeatComplete=true;
tft.setTextColor(TFT_GREEN, TFT_BLACK);
int padding = tft.textWidth("999", 6);
tft.setTextPadding(padding);
tft.setTextDatum(TC_DATUM);
tft.drawNumber(0,55,120,6);
```

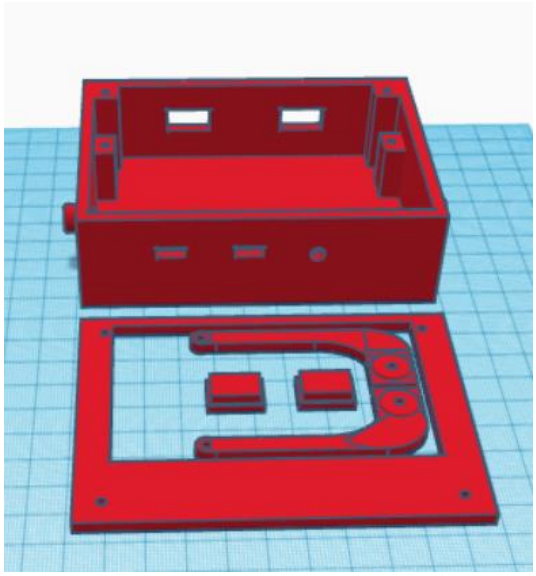
```
int padding1 = tft.textWidth("999.99", 4);
tft.setTextPadding(padding1);
tft.setTextDatum(TR_DATUM);
tft.drawFloat(temperature,2,80,290,4);
tft.setTextPadding(0);
tft.drawString("o",90,285,2);
tft.drawString("C",110, 290, 4);
}
void bluetooth(void) {
   // send the value of analog input 0 to serial:
   //Do the same for blutooth
   If (BT_cnx) {
     ESP_BT.print('E'); //make the app Blutooth Graphics
(https://play.google.com/store/apps/details?id=com.emrctn.BluetoothGraphics&hl
=en_US) work (as specified by the app)
     ESP_BT.print(c1);
     ESP_BT.print(",");
     ESP_BT.print(c2);
     ESP_BT.print(",");
     ESP_BT.println(c3);
   }
  }
void Home(void) {
  tft.setTextColor(TFT_GREEN, TFT_BLACK);
  tft.setTextPadding(0);
  tft.setTextDatum(TC_DATUM);
  tft.drawString("ECG", 240, 90, 4);
  tft.drawString("WIRELESS", 240, 130, 4);
  for (int16_t t=0; t<480; t += 1) {
  tft.fillRect(t,200,1,25,TFT_BLUE);
  delay (1);
  }
}
void Suhu(void) {
  // this part keeps the var temperature up to date as possible.
  if (millis() - lastTempRequest >= delayInMillis) // waited long enough??
  {
    temperature = sensors.getTempCByIndex(0);
    sensors.requestTemperatures();
    lastTempRequest = millis();
  }
}
```

Lampiran 4 Desain 3D *Cover* Alat Wireless ECG

- **Tampak Atas**



- **Tampak Samping**



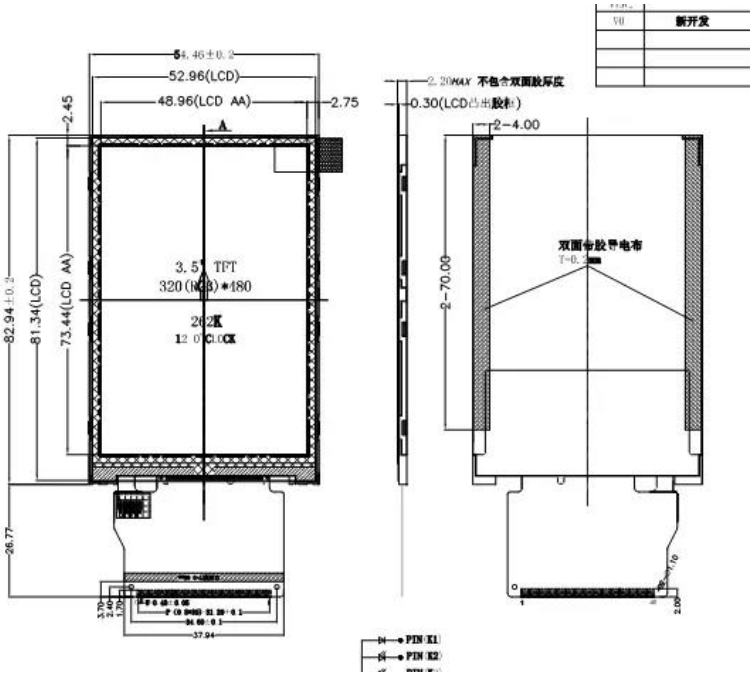
Lampiran 5 Ukuran Dimensi Desain *Cover* Alat

Panjang = 115,40 mm
Lebar = 81,0 mm
Tinggi = 30 mm

Lampiran 6 Spesifikasi ILI9488

Layar LCD TFT RGB 3,5 inci IC ILI9488

Nama	ENH-TV00035008	Dimensi Garis Besar	52.96 * 81.34 * 2.20mm
Resolusi	320 (RGB) * 480	Area Aktif	48.96 * 73.44mm
Mode tampilan	TFT / Transmisif	Pixel Pitch	0,153 * 0,153mm
Melihat Arah	12 O'jam	Lampu latar	3 LED Putih Chip Paralel
Drive IC	ILI9488	Warna	Putih (3.2V, 100mA)
Puncak	-10℃~+60℃	Tst	-20℃~+70℃



Lampiran 7 Hasil alat yang telah dibuat