**DOKUMEN DESAIN PERANGKAT LUNAK**

**BLUELIGHT**

SOURCE CODE

*LSKK-SW-SC-01-BL*



**PT. LANGGENG SEJAHTERA KREASI KOMPUTASI**

**©2019**

Dokumen Desain Perangkat Lunak, BLUELIGHT

Source Code Sistem Bluelight

PT. Langgeng Sejahtera Kreasi Komputasi ©2019

# Deskripsi Aplikasi

Sistem dirancang sebagai bagian dari penelitian yang terkait dengan pola penanggulangan keadaan darurat,yang bersifat individual dan memerlukan layanan “*bluelight*” seperti ambulan, polisi dan pemadam kebakaran, maupun yang sifatnya lebih luas seperti banjir, gempa dll. Sistem ini secara arsitektur mirip dengan sistem *homeauto*, namun selain menggunakan sensor-sensor yang khusus terkait dengan bencana tersebut, juga mengeksploitasi pola sosial dalam mitigasinya.

# *Source Code*

Berikut *source code* untuk sistem perangkat lunak *microcontroller* system bluelight:

|  |
| --- |
| /\*  Develop by : Nurman Hariyanto  Email : nurman.hariyanto13@gmail.com  Project : Homeautomation  Version : 3.0  \*/  //library  #include <FS.h> //this needs to be first, or it all crashes and burns...  #include <Ticker.h>  #include <ESP8266WiFi.h>  #include <DNSServer.h>  #include <ESP8266WebServer.h>  #include <WiFiManager.h>  #include <ArduinoJson.h>  //#include <SPI.h>  //#include "MFRC522.h"  #include <PubSubClient.h>  #include "config.h"  //var temp untuk menampung data json dari payload  // constants won't change. They're used here to set pin numbers:  const int LED = D2; // the number of the pushbutton pin  const int sensorGas = D0; // the number of the pushbutton pin  const int buzzer = D3;  // variables will change:  int sensorGasState = 0;  /\*  \* Setup pin and load config file  \*/  void setup() {  pinMode(sensorGas,INPUT);  pinMode(LED, OUTPUT);  pinMode(buzzer,OUTPUT);  digitalWrite(buzzer,LOW);    Serial.begin(115200);  Serial.println(F("Booting...."));  //read config wifi,mqtt dan yang lain  ReadConfigFile();  setup\_wifi();  SaveConfigFile();  client.setServer(mqtt\_server, mqtt\_port);  client.setCallback(callback);  }  //setup wifi  void setup\_wifi() {  WiFi.macAddress(MAC\_array);  for (int i = 0; i < sizeof(MAC\_array) - 1; ++i) {  sprintf(MAC\_char, "%s%02x:", MAC\_char, MAC\_array[i]);  }  sprintf(MAC\_char, "%s%02x", MAC\_char, MAC\_array[sizeof(MAC\_array) - 1]);  WiFiManagerParameter custom\_mqtt\_server("server", "mqtt server", mqtt\_server, 40);  String(mqtt\_port).toCharArray(smqtt\_port, 5);  WiFiManagerParameter custom\_mqtt\_port("port", "mqtt port", smqtt\_port, 5);  WiFiManagerParameter custom\_mqtt\_user("user", "mqtt user", mqtt\_user, 40);  WiFiManagerParameter custom\_mqtt\_password("password", "mqtt password", mqtt\_password, 40);  WiFiManagerParameter custom\_mqtt\_keywords1("keyword1", "mqtt keyword1", mqtt\_keywords1, 40);  ///WiFiManagerParameter custom\_mqtt\_keywords2("keyword2", "mqtt keyword2", mqtt\_keywords2, 40);  WiFiManager wifiManager;  wifiManager.setSaveConfigCallback(saveConfigCallback);  wifiManager.addParameter( & custom\_mqtt\_server);  wifiManager.addParameter( & custom\_mqtt\_port);  wifiManager.addParameter( & custom\_mqtt\_user);  wifiManager.addParameter( & custom\_mqtt\_password);  wifiManager.addParameter(&custom\_mqtt\_keywords1);  //wifiManager.addParameter( & custom\_mqtt\_keywords2);  //fetches ssid and pass and tries to connect  //if it does not connect it starts an access point with the specified name  //here "AutoConnectAP"  //and goes into a blocking loop awaiting configuration  if (!wifiManager.autoConnect(MAC\_char, "password")) {  Serial.println("failed to connect and hit timeout");  delay(2000);  //reset and try again, or maybe put it to deep sleep  ESP.reset();  delay(2000);  }  Serial.println("");  Serial.println("WiFi connected");  Serial.println("IP address: ");  Serial.println(WiFi.localIP());  }  //calback  void callback(char \* topic, byte \* payload, unsigned int length) {  Serial.print("Message arrived [");  Serial.print(topic);  Serial.print("] ");  Serial.println();      }  void reconnect() {  // Loop until we're reconnected  while (!client.connected()) {  Serial.print("Attempting MQTT connection...%s");  Serial.println(mqtt\_server);  // Attempt to connect  if (client.connect(MAC\_char, mqtt\_user, mqtt\_password)) {  Serial.println("connected");  // Once connected, publish an announcement...  //client.publish(mqtt\_keywords1, "halo smarthome");  } else {  Serial.print("failed, rc=");  Serial.print(client.state());  if (client.state() == 4) ESP.restart();  else {  Serial.println(" try again in 5 seconds");  // Wait 5 seconds before retrying  //delay(10000);  }  }  }  }  //loop publish  //loop publish  void loop() {    if (!client.connected()) {  reconnect();  }  client.loop();  sensorGasState = digitalRead(sensorGas);  Serial.println(sensorGasState);    // check if the pushbutton is pressed. If it is, the buttonState is HIGH:  String jenis\_iot = "Input-Sensor-Asap";  String macAdd = MAC\_char;  String data1 = "detected";  int data2 = 0;  int data3 = 0;  String kode\_device = "D92232B5FF";    if (sensorGasState== LOW) {  digitalWrite(buzzer,HIGH);  String pubmsg = "#" + jenis\_iot + "#" + macAdd + "#" + data1 + "#" + data2 + "#" + data3 + "#" + kode\_device;  char datapub[200];  pubmsg.toCharArray(datapub,sizeof(datapub));  client.publish(mqtt\_keywords1,datapub);  Serial.println(datapub);  Serial.println("Ada Gas");  delay(5000);  }  else if (sensorGasState == HIGH) {  digitalWrite(buzzer,LOW);  Serial.println("Aman");  }    } |

Source 1. *Source code*  *microcontroller* sensor asap main

|  |
| --- |
| #ifndef \_\_CONFIGFILE\_\_  #define \_\_CONFIGFILE\_\_  //flag for saving data  bool shouldSaveConfig = false;  char wifi\_ssid[34] = "RuangIOT";  char wifi\_password[34] = "lskkiot2019";  char mqtt\_server[40] = "rmqbluelight.pptik.id";  int mqtt\_port = 1883;  char smqtt\_port[5] = "1883";  char mqtt\_user[40] = "/bluelight:bluelight";  char mqtt\_password[40] = "bluelight123!";  char mqtt\_keywords1[40] = "bluelightin"; //gas  const long interval = 2000;  uint8\_t MAC\_array[6];  char MAC\_char[18];  WiFiClient espClient;  PubSubClient client(espClient);  void SaveConfigCallback ();  void ReadConfigFile();  #endif |

Source 2. *Source code*  *microcontroller* sensor asap config

|  |
| --- |
| #include "config.h"  //callback notifying us of the need to save config  void saveConfigCallback () {  Serial.println("Should save config");  shouldSaveConfig = true;  }  void ReadConfigFile()  {  //read configuration from FS json  Serial.println("mounting FS...");  if (SPIFFS.begin()) {  Serial.println("mounted file system");  if (SPIFFS.exists("/config.json")) {  //SPIFFS.remove("/config.json");  //file exists, reading and loading  Serial.println("reading config file");  File configFile = SPIFFS.open("/config.json", "r");  if (configFile) {  Serial.println("opened config file");  size\_t size = configFile.size();  // Allocate a buffer to store contents of the file.  std::unique\_ptr<char[]> buf(new char[size]);  configFile.readBytes(buf.get(), size);  DynamicJsonBuffer jsonBuffer;  JsonObject& json = jsonBuffer.parseObject(buf.get());  json.printTo(Serial);  if (json.success()) {  Serial.println("\nparsed json");  strcpy(mqtt\_server, json["mqtt\_server"]);  String port = json["mqtt\_port"];  mqtt\_port = port.toInt();  strcpy(mqtt\_user, json["mqtt\_user"]);  strcpy(mqtt\_password, json["mqtt\_password"]);  strcpy(mqtt\_keywords1, json["mqtt\_keywords1"]);  //strcpy(mqtt\_keywords2, json["mqtt\_keywords2"]);  } else {  Serial.println("failed to load json config");  }  }  }  } else {  Serial.println("failed to mount FS");  }  //end read  }  String IpAddress2String(const IPAddress& ipAddress)  {  return String(ipAddress[0]) + String(".") +\  String(ipAddress[1]) + String(".") +\  String(ipAddress[2]) + String(".") +\  String(ipAddress[3]) ;  }  void SaveConfigFile()  {    //save the custom parameters to FS  if (shouldSaveConfig) {  Serial.println("saving config");  DynamicJsonBuffer jsonBuffer;  JsonObject& json = jsonBuffer.createObject();  json["mqtt\_server"] = mqtt\_server;  json["mqtt\_port"] = String(mqtt\_port);  json["mqtt\_user"] = mqtt\_user;  json["mqtt\_password"] = mqtt\_password;  json["mqtt\_keywords1"] = mqtt\_keywords1;  //json["mqtt\_keywords2"] = mqtt\_keywords2;    File configFile = SPIFFS.open("/config.json", "w");  if (!configFile) {  Serial.println("failed to open config file for writing");  }  json.printTo(Serial);  json.printTo(configFile);  configFile.close();  //end save  }  }  void ResetSettings()  {  WiFiManager wifiManager;  wifiManager.resetSettings();  delay(2000);  ESP.restart();  } |

Source 3. *Source code*  *microcontroller* sensor asap configfile

|  |
| --- |
| /\*  Develop by : Nurman Hariyanto  Email : nurman.hariyanto13@gmail.com  Project : Homeautomation  Version : 3.0  \*/  //library  #include <FS.h> //this needs to be first, or it all crashes and burns...  #include <Ticker.h>  #include <ESP8266WiFi.h>  #include <DNSServer.h>  #include <ESP8266WebServer.h>  #include <WiFiManager.h>  #include <ArduinoJson.h>  //#include <SPI.h>  //#include "MFRC522.h"  #include <PubSubClient.h>  #include "config.h"  int Status = D4; // Digital pin D6  int sensor = D2; // Digital pin D7  int sensorState = 0;  int prevState = 0;  void setup() {  Serial.begin(115200);  pinMode(sensor, INPUT\_PULLUP); // declare sensor as input  pinMode(Status, OUTPUT); // declare LED as output  Serial.println(F("Booting...."));  //read config wifi,mqtt dan yang lain  ReadConfigFile();  setup\_wifi();  SaveConfigFile();  client.setServer(mqtt\_server, mqtt\_port);  client.setCallback(callback);  }  //setup wifi  void setup\_wifi() {  WiFi.macAddress(MAC\_array);  for (int i = 0; i < sizeof(MAC\_array) - 1; ++i) {  sprintf(MAC\_char, "%s%02x:", MAC\_char, MAC\_array[i]);  }  sprintf(MAC\_char, "%s%02x", MAC\_char, MAC\_array[sizeof(MAC\_array) - 1]);  WiFiManagerParameter custom\_mqtt\_server("server", "mqtt server", mqtt\_server, 40);  String(mqtt\_port).toCharArray(smqtt\_port, 5);  WiFiManagerParameter custom\_mqtt\_port("port", "mqtt port", smqtt\_port, 5);  WiFiManagerParameter custom\_mqtt\_user("user", "mqtt user", mqtt\_user, 40);  WiFiManagerParameter custom\_mqtt\_password("password", "mqtt password", mqtt\_password, 40);  WiFiManagerParameter custom\_mqtt\_keywords1("keyword1", "mqtt keyword1", mqtt\_keywords1, 40);  ///WiFiManagerParameter custom\_mqtt\_keywords2("keyword2", "mqtt keyword2", mqtt\_keywords2, 40);  WiFiManager wifiManager;  wifiManager.setSaveConfigCallback(saveConfigCallback);  wifiManager.addParameter( & custom\_mqtt\_server);  wifiManager.addParameter( & custom\_mqtt\_port);  wifiManager.addParameter( & custom\_mqtt\_user);  wifiManager.addParameter( & custom\_mqtt\_password);  wifiManager.addParameter(&custom\_mqtt\_keywords1);  //wifiManager.addParameter( & custom\_mqtt\_keywords2);  //fetches ssid and pass and tries to connect  //if it does not connect it starts an access point with the specified name  //here "AutoConnectAP"  //and goes into a blocking loop awaiting configuration  if (!wifiManager.autoConnect(MAC\_char, "password")) {  Serial.println("failed to connect and hit timeout");  delay(2000);  //reset and try again, or maybe put it to deep sleep  ESP.reset();  delay(2000);  }  Serial.println("");  Serial.println("WiFi connected");  Serial.println("IP address: ");  Serial.println(WiFi.localIP());  }  //calback  void callback(char \* topic, byte \* payload, unsigned int length) {  Serial.print("Message arrived [");  Serial.print(topic);  Serial.print("] ");  Serial.println();  }  void reconnect() {  // Loop until we're reconnected  while (!client.connected()) {  Serial.print("Attempting MQTT connection...%s");  Serial.println(mqtt\_server);  // Attempt to connect  if (client.connect(MAC\_char, mqtt\_user, mqtt\_password)) {  Serial.println("connected");  // Once connected, publish an announcement...  //client.publish(mqtt\_keywords1, "halo smarthome");  } else {  Serial.print("failed, rc=");  Serial.print(client.state());  if (client.state() == 4) ESP.restart();  else {  Serial.println(" try again in 5 seconds");  // Wait 5 seconds before retrying  //delay(10000);  }  }  }  }  //loop publish  void loop() {  if (!client.connected()) {  reconnect();  }  client.loop();  String jenis\_iot = "Sensor\_Pir";  String macAdd = MAC\_char;  String data1 = ("detected");  int data2 = 0;  int data3 = 0;  String kode\_device = "D92232B5FF";  sensorState = digitalRead(sensor);  if (sensorState == HIGH) {  if (prevState != sensorState) {  digitalWrite (Status, HIGH);  Serial.println("Motion detected!");  String pubmsg = "#" + jenis\_iot + "#" + macAdd + "#" + data1 + "#" + data2 + "#" + data3 + "#" + kode\_device;  char datapub[200];  pubmsg.toCharArray(datapub, sizeof(datapub));  client.publish(mqtt\_keywords1, datapub);  Serial.println("Success send to RMQ");  delay(1000);  }  }  else {  digitalWrite (Status, LOW);  Serial.println("Motion absent!");  delay(1000);  }  } |

Source 4. *Source code*  *microcontroller* sensor pir main

|  |
| --- |
| #ifndef \_\_CONFIGFILE\_\_  #define \_\_CONFIGFILE\_\_  //flag for saving data  bool shouldSaveConfig = false;  char wifi\_ssid[34] = "RuangIOT";  char wifi\_password[34] = "lskkiot2019";  char mqtt\_server[40] = "rmqbluelight.pptik.id";  int mqtt\_port = 1883;  char smqtt\_port[5] = "1883";  char mqtt\_user[40] = "/bluelight:bluelight";  char mqtt\_password[40] = "bluelight123!";  char mqtt\_keywords1[40] = "bluelightin"; //gas  const long interval = 2000;  uint8\_t MAC\_array[6];  char MAC\_char[18];  WiFiClient espClient;  PubSubClient client(espClient);  void SaveConfigCallback ();  void ReadConfigFile();  #endif |

Source 5. *Source code*  *microcontroller* sensor pir config

|  |
| --- |
| #include "config.h"  //callback notifying us of the need to save config  void saveConfigCallback () {  Serial.println("Should save config");  shouldSaveConfig = true;  }  void ReadConfigFile()  {  //read configuration from FS json  Serial.println("mounting FS...");  if (SPIFFS.begin()) {  Serial.println("mounted file system");  if (SPIFFS.exists("/config.json")) {  //SPIFFS.remove("/config.json");  //file exists, reading and loading  Serial.println("reading config file");  File configFile = SPIFFS.open("/config.json", "r");  if (configFile) {  Serial.println("opened config file");  size\_t size = configFile.size();  // Allocate a buffer to store contents of the file.  std::unique\_ptr<char[]> buf(new char[size]);  configFile.readBytes(buf.get(), size);  DynamicJsonBuffer jsonBuffer;  JsonObject& json = jsonBuffer.parseObject(buf.get());  json.printTo(Serial);  if (json.success()) {  Serial.println("\nparsed json");  strcpy(mqtt\_server, json["mqtt\_server"]);  String port = json["mqtt\_port"];  mqtt\_port = port.toInt();  strcpy(mqtt\_user, json["mqtt\_user"]);  strcpy(mqtt\_password, json["mqtt\_password"]);  strcpy(mqtt\_keywords1, json["mqtt\_keywords1"]);  //strcpy(mqtt\_keywords2, json["mqtt\_keywords2"]);  } else {  Serial.println("failed to load json config");  }  }  }  } else {  Serial.println("failed to mount FS");  }  //end read  }  String IpAddress2String(const IPAddress& ipAddress)  {  return String(ipAddress[0]) + String(".") +\  String(ipAddress[1]) + String(".") +\  String(ipAddress[2]) + String(".") +\  String(ipAddress[3]) ;  }  void SaveConfigFile()  {    //save the custom parameters to FS  if (shouldSaveConfig) {  Serial.println("saving config");  DynamicJsonBuffer jsonBuffer;  JsonObject& json = jsonBuffer.createObject();  json["mqtt\_server"] = mqtt\_server;  json["mqtt\_port"] = String(mqtt\_port);  json["mqtt\_user"] = mqtt\_user;  json["mqtt\_password"] = mqtt\_password;  json["mqtt\_keywords1"] = mqtt\_keywords1;  //json["mqtt\_keywords2"] = mqtt\_keywords2;    File configFile = SPIFFS.open("/config.json", "w");  if (!configFile) {  Serial.println("failed to open config file for writing");  }  json.printTo(Serial);  json.printTo(configFile);  configFile.close();  //end save  }  }  void ResetSettings()  {  WiFiManager wifiManager;  wifiManager.resetSettings();  delay(2000);  ESP.restart();  } |

Source 6. *Source code*  *microcontroller* sensor pir configfile

|  |
| --- |
| #include <PubSubClient.h>  #include <ESP8266WiFi.h>  #include <WiFiUdp.h>  // Update these with values suitable for your network.  const char\* ssid = "RuangIOT2";  const char\* password = "lskkiot2019";  const char\* mqtt\_server = "rmqbluelight.pptik.id";  const char\* mqtt\_user = "/bluelight:bluelight";  const char\* mqtt\_pass = "bluelight123!";  const char\* CL = "bl-02";  const char\* mqtt\_pub\_topic = "bluelightin";  char msg[100];  WiFiClient espClient;  PubSubClient client(espClient);  byte mac[6];  String MACAddress;  const int echoPin = D1;  const int trigPin = D2;  const int LED = D3;  const int LEDPin = 0;  void setup\_wifi() {  delay(10);  // We start by connecting to a WiFi network  Serial.println();  Serial.print("Connecting to ");  Serial.println(ssid);  WiFi.begin(ssid, password);  while (WiFi.status() != WL\_CONNECTED) {  delay(500);  Serial.print(".");  }  Serial.println("");  Serial.println("WiFi connected");  Serial.println("IP address: ");  Serial.println(WiFi.localIP());  }  String mac2String(byte ar[]) {  String s;  for (byte i = 0; i < 6; ++i)  {  char buf[3];  sprintf(buf, "%2X", ar[i]);  s += buf;  if (i < 5) s += ':';  }  return s;  }  void printMACAddress() {  WiFi.macAddress(mac);  MACAddress = mac2String(mac);  Serial.println(MACAddress);  }  void callback(char\* topic, byte\* payload, unsigned int length) {  Serial.print("Message arrived [");  Serial.print(topic);  Serial.print("] ");  for (int i = 0; i < length; i++) {  Serial.print((char)payload[i]);  }  Serial.println();  // Switch on the LED if an 1 was received as first character  if ((char)payload[0] == '1') {  digitalWrite(LED, HIGH); // Turn the LED on (Note that HIGH is the voltage level  } else {  digitalWrite(LED, LOW); // Turn the LED off by making the voltage LOW  }  }  void reconnect() {  // Loop until we're reconnected  printMACAddress();  const char\* CL;  CL = MACAddress.c\_str();  Serial.println(CL);  while (!client.connected()) {  Serial.print("Attempting MQTT connection...");  // Attempt to connect  if (client.connect(CL, mqtt\_user, mqtt\_pass)) {  Serial.println("connected");  digitalWrite(LED, HIGH);  } else {  digitalWrite(LED, LOW);  Serial.print("failed, rc=");  Serial.print(client.state());  Serial.println(" try again in 5 seconds");  ESP.restart();  delay(5000);  }  }  }  void setup()  {  Serial.begin(115200);  setup\_wifi();  printMACAddress();  client.setServer(mqtt\_server, 1883);  client.setCallback(callback);  pinMode(LED, OUTPUT);  pinMode(trigPin, OUTPUT);  pinMode(echoPin, INPUT);  pinMode(LEDPin, OUTPUT);  }  void loop() {  String pubmsg1 = "";  String pubmsg2 = "";  String pubmsg3 = "";  String pubmsg4 = "";  /\* Berikut siklus trigPin atau echo pin yang digunakan  untuk menentukan jarak objek terdekat dengan memantulkan  gelombang suara dari itu. \*/  int maximumRange = 200; //kebutuhan akan maksimal range  int minimumRange = 00; //kebutuhan akan minimal range  long duration, distance; //waktu untuk kalkulasi jarak  digitalWrite(trigPin, LOW); delayMicroseconds(2);  digitalWrite(trigPin, HIGH); delayMicroseconds(10);  digitalWrite(trigPin, LOW);  duration = pulseIn(echoPin, HIGH);  distance = duration / 58.2;  String jenis\_iot = "Input-Sensor-Banjir";  String mac = MACAddress;  String data\_1 = ("detected");  int data\_2 = (distance);  String siaga1 = ("Siaga 1");  String siaga2 = ("Siaga 2");  String siaga3 = ("Siaga 3");  String siaga4 = ("Banjir");  String kode\_device = "D92232B5FF";  Serial.print("Publish message: ");  //perhitungan untuk dijadikan jarak    if (!client.connected()) {  reconnect();  }  client.loop();  delay(100);  if (distance >= maximumRange || distance <= minimumRange)  {  Serial.println("-1"); digitalWrite(LEDPin, HIGH);  }  if (distance >= 200)  {  Serial.println("Aman");  }  else if (distance >= 180)  {  Serial.println("Siaga 1");  String pubmsg1 = "#" + jenis\_iot + "#" + mac + "#" + data\_1 + "#" + data\_2 + "#" + siaga1 + "#" + kode\_device;  //harus di convert dulu ke charbyte  char dataStatus1[200];  pubmsg1.toCharArray(dataStatus1, sizeof(dataStatus1));  client.publish(mqtt\_pub\_topic, dataStatus1);  Serial.println(dataStatus1);  }  else if (distance >= 120)  {  Serial.println("Siaga 2");  String pubmsg2 = "#" + jenis\_iot + "#" + mac + "#" + data\_1 + "#" + data\_2 + "#" + siaga2 + "#" + kode\_device;  //harus di convert dulu ke charbyte  char dataStatus2[200];  pubmsg2.toCharArray(dataStatus2, sizeof(dataStatus2));  client.publish(mqtt\_pub\_topic, dataStatus2);  Serial.println(dataStatus2);  }  else if (distance >= 60)  {  Serial.println("Siaga 3");  String pubmsg3 = "#" + jenis\_iot + "#" + mac + "#" + data\_1 + "#" + data\_2 + "#" + siaga3 + "#" + kode\_device;  //harus di convert dulu ke charbyte  char dataStatus3[200];  pubmsg3.toCharArray(dataStatus3, sizeof(dataStatus3));  client.publish(mqtt\_pub\_topic, dataStatus3);  Serial.println(dataStatus3);  }  else if (distance < 60)  {  Serial.println("Banjir");  String pubmsg4 = "#" + jenis\_iot + "#" + mac + "#" + data\_1 + "#" + data\_2 + "#" + siaga4 + "#" + kode\_device;  //harus di convert dulu ke charbyte  char dataStatus4[200];  pubmsg4.toCharArray(dataStatus4, sizeof(dataStatus4));  client.publish(mqtt\_pub\_topic, dataStatus4);  Serial.println(dataStatus4);  }  /\*Kirim jarak ke komputer menggunakan Serial protokol, dan  menghidupkan LED OFF untuk menunjukkan membaca sukses. \*/  Serial.println(distance);  digitalWrite(LEDPin, LOW);  //waktu tunda 50mS  delay(50);  delay(1000);  } |

Source 7. *Source code*  *microcontroller* sensor banjir

# *Release Notes*

Berikut *release notes* aplikasi untuk sistem bluelight.

### *BL-SC, b1912, v1.0*

* Versi rilis pertama.

# *Approval*

Dibuat oleh Tim *S/W* *Engineer* pada 13 Desember 2019.

Dicek dan disetujui pada 13 Desember, oleh:

Mochamad Vicky Ghani Aziz