LAMPIRAN-LAMPIRAN

Lampiran 1 Source Code ESP-32

```
// Start Code for ESP32 Devkit
#include <WiFi.h>
#include <DHT.h>
// #include <BH1750.h>
// #include <Wire.h>
#include <UniversalTelegramBot.h>
#include <WiFiClientSecure.h>
#include <PubSubClient.h>


// Set WiFi dan Telegram
#define WIFI_SSID "realme 3 Pro"
#define WIFI_PASSWORD "12345678"

#define BOT_TOKEN
"7771068553:AAGfW_TqTMVsa50tU8fyN9iExk_s2WzgZBg"

const char* mqtt_server = "192.168.115.96";
const char* mqtt_user = "admin";
const char* mqtt_pass = "admin123";

// Cloudflare DNS servers
// IPAddress primaryDNS(1, 1, 1, 1);
// IPAddress secondaryDNS(1, 0, 0, 1);

// Sensor dan relay pins
#define DHTPIN 18               // Pin untuk DHT22
#define MOISTURE_SENSOR_PIN 34  // Pin analog untuk sensor
kelembaban media jamur
#define DHTTYPE DHT22           // Tipe sensor DHT
#define LDR_PIN 32

#define RELAY_PIN_MINI PUMP 4      // Pin relay untuk solenoid
mini pump
#define RELAY_PIN_FAN 19        // Pin relay untuk kipas
#define RELAY_PIN_LAMP 23       // Pin relay untuk lampu

// Batas nilai sensor dan kontrol
#define FAN_TEMP_THRESHOLD 30.0      // Suhu ambang batas
```

```
// Inisialisasi DHT, BH1750, dan Telegram bot
WiFiClient espClient;
PubSubClient client(espClient);

DHT dht(DHTPIN, DHTTYPE);
// BH1750 lightMeter;
WiFiClientSecure clientTCP;
UniversalTelegramBot bot(BOT_TOKEN, clientTCP);

// Chat ID Telegram
String chat_id = "8152895238";

// Status perangkat
bool fanStatus = false;
bool lampStatus = false;
bool mini pumpStatus = false;
bool autoModeFan = true;
bool autoModeLamp = true;
bool autoModeMini pump = true;

void setup() {
  Serial.begin(115200);

  // Koneksi ke WiFi
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Menghubungkan ke WiFi...");
  }
  Serial.println("WiFi tersambung!");

  client.setServer(mqtt_server, 1883);
  client.connect("ESP32Dev", mqtt_user, mqtt_pass);

  if (!client.connected()) {
    Serial.println("MQTT connection failed");
  } else {
    Serial.println("Connected to MQTT broker");
  }
```

```
//fungsi telegram

void sendMQTT() {
  if (!client.connected()) {
    reconnectMQTT();  // Apabila disconnect MQTTnya akan
reconnect
  }

  if (client.connected()) {
    client.publish("mushroom/temp-alert", "high-temp");
    Serial.println("Mengirim peringatan ke MQTT");
  } else {
    Serial.println("MQTT connection failed");
  }
}

void reconnectMQTT() {
  if (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    if (client.connect("ESP32Dev", mqtt_user, mqtt_pass)) {
      Serial.println("connected to MQTT broker");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 10 seconds");
      delay(10000);  // Increase delay after failed connection
    }
  }
}

// Fungsi untuk mengirim status semua sensor
void sendStatus() {
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();
  // float lux = lightMeter.readLightLevel();
  int lightValue = analogRead(LDR_PIN);
  int soilMoistureValue = analogRead(MOISTURE_SENSOR_PIN);

  // Membuat pesan status
  String message = "Status Sistem:\n";
  message += "Suhu: " + String(temperature) + " °C\n";
  message += "Kelembaban: " + String(humidity) + " %\n";
  // message += "Cahaya: " + String(lux) + " lux\n";
```

```
message += "Cahaya: " + String(lightValue) + " (0-4095)\n";
  message += "Kelembaban Media jamur: " +
String(soilMoistureValue) + " (0-4095)\n";
  message += "Kipas: " + String(fanStatus ? "ON" : "OFF") + "\n";
  message += "Lampu: " + String(lampStatus ? "ON" : "OFF") +
"\n";
  message += "Pump: " + String(mini pumpStatus ? "ON" : "OFF") +
"\n";

  bot.sendMessage(chat_id, message, "");
}

void handleFanControl(String text) {
  if (text == "/fan_on") {
    digitalWrite(RELAY_PIN_FAN, LOW);
    bot.sendMessage(chat_id, "Kipas dinyalakan secara manual.",
"");
    fanStatus = true;
    autoModeFan = false;
  } else if (text == "/fan_off") {
    digitalWrite(RELAY_PIN_FAN, HIGH);
    bot.sendMessage(chat_id, "Kipas dimatikan secara manual.",
"");
    fanStatus = false;
    autoModeFan = false;
  } else if (text == "/fan_auto") {
    bot.sendMessage(chat_id, "Mode otomatis kipas diaktifkan.",
"");
    autoModeFan = true;
  }
}

void handleLampControl(String text) {
  if (text == "/lamp_on") {
    digitalWrite(RELAY_PIN_LAMP, LOW);
    bot.sendMessage(chat_id, "Lampu dinyalakan secara manual.",
"");
    lampStatus = true;
    autoModeLamp = false;
  } else if (text == "/lamp_off") {
    digitalWrite(RELAY_PIN_LAMP, HIGH);
    bot.sendMessage(chat_id, "Lampu dimatikan secara manual.",
"");
    lampStatus = false;
    autoModeLamp = false;
  } else if (text == "/lamp_auto") {
```

```
bot.sendMessage(chat_id, "Mode otomatis lampu diaktifkan.", "");
    autoModeLamp = true;
  }
}

void handleMini pumpControl(String text) {
  if (text == "/pump_on") {
    digitalWrite(RELAY_PIN_MINI PUMP, LOW);
    bot.sendMessage(chat_id, "Pump dinyalakan secara manual.",
"");
    mini pumpStatus = true;
    autoModeMini pump = false;
  } else if (text == "/pump_off") {
    digitalWrite(RELAY_PIN_MINI PUMP, HIGH);
    bot.sendMessage(chat_id, "Pump dimatikan secara manual.",
"");
    mini pumpStatus = false;
    autoModeMini pump = false;
  } else if (text == "/pump_auto") {
    bot.sendMessage(chat_id, "Mode otomatis pump diaktifkan.",
"");
    autoModeMini pump = true;
  }
}

void checkTemperatureAndControlFan() {
  if (autoModeFan) {
    float temperature = dht.readTemperature();
    if (isnan(temperature)) {
      Serial.println("Gagal membaca sensor DHT22");
      return;
    }

    Serial.print("Suhu: ");
    Serial.println(temperature);

    if (temperature >= FAN_TEMP_THRESHOLD) {
      if (!fanStatus) {
        digitalWrite(RELAY_PIN_FAN, LOW);
        bot.sendMessage(chat_id, "Suhu melebihi 30.9°C, kipas
dinyalakan!", "");

        fanStatus = true;
      }
    } else {
      if (fanStatus) {
```

```
 digitalWrite(RELAY_PIN_FAN, HIGH);
        bot.sendMessage(chat_id, "Suhu sudah turun, kipas
dimatikan!", "");
        fanStatus = false;
      }
    }
  }
}

void checkLightAndControlLamp() {
  if (autoModeLamp) {
    // float lux = lightMeter.readLightLevel();
    // if (lux == 0) {
    //   Serial.println("Gagal membaca sensor BH1750 atau
intensitas cahaya terlalu rendah.");
    //   return;
    // }

    int lightValue = analogRead(LDR_PIN);
    Serial.print("Cahaya (LDR analog): ");
    Serial.println(lightValue);

    // Serial.print("Cahaya: ");
    // Serial.println(lux);

    if (lightValue < LIGHT_THRESHOLD) {
      if (!lampStatus) {
        digitalWrite(RELAY_PIN_LAMP, LOW);
        bot.sendMessage(chat_id, "Intensitas cahaya rendah, lampu
dinyalakan!", "");
        lampStatus = true;
        sendMQTT();
      }
    } else {
      if (lampStatus) {
        digitalWrite(RELAY_PIN_LAMP, HIGH);
        bot.sendMessage(chat_id, "Cahaya cukup terang, lampu
dimatikan!", "");
        lampStatus = false;
      }
    }
  }
}

void checkSoilMoistureAndControlMini pump() {
  if (autoModeMini pump) {
```

```
int soilMoistureValue = analogRead(MOISTURE_SENSOR_PIN);
    Serial.print("Kelembaban media jamur: ");
    Serial.println(soilMoistureValue);

    if (soilMoistureValue > MOISTURE_THRESHOLD) {
      if (!mini pumpStatus) {
        digitalWrite(RELAY_PIN_MINI PUMP, LOW);
        bot.sendMessage(chat_id, "Kelembaban rendah, pump
dinyalakan!", "");
        mini pumpStatus = true;
        sendMQTT();
      }
    } else {
      if (mini pumpStatus) {
        digitalWrite(RELAY_PIN_MINI PUMP, HIGH);
        bot.sendMessage(chat_id, "Kelembaban media jamur cukup,
pump dimatikan!", "");
        mini pumpStatus = false;
      }
    }
  }
}

void loop() {

  int numNewMessages = bot.getUpdates(bot.last_message_received +
1);

  for (int i = 0; i < numNewMessages; i++) {
    if (bot.messages[i].chat_id != "") {  // Ensure the message
exists
      String chat_id_message = bot.messages[i].chat_id;
      String text = bot.messages[i].text;

      if (chat_id_message == chat_id) {
        if (text == "/status") {
          sendStatus();  // Kirim status semua sensor
        }
        else {
          handleFanControl(text);
          handleLampControl(text);
          handleMini pumpControl(text);
        }
      }
    }
  }
```

```
  checkTemperatureAndControlFan();
  checkLightAndControlLamp();
  checkSoilMoistureAndControlMini pump();


  delay(2000); // Jeda 2 detik sebelum pengecekan berikutnya
}

//end code esp32 devkit


  checkTemperatureAndControlFan();
  checkLightAndControlLamp();
  checkSoilMoistureAndControlMini pump();


  delay(2000); // Jeda 2 detik sebelum pengecekan berikutnya
}

//end code esp32 devkit
```

Lampiran 2 Source Code ESP-32 CAM

```
/ start code for esp32 cam
/*
  Rui Santos
  Complete project details at
https://RandomNerdTutorials.com/telegram-esp32-cam-photo-arduino/

  Permission is hereby granted, free of charge, to any person
obtaining a copy
  of this software and associated documentation files.

  The above copyright notice and this permission notice shall be
included in all
  copies or substantial portions of the Software.
*/

#include <Arduino.h>
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"
#include "esp_camera.h"
#include <UniversalTelegramBot.h>
#include <ArduinoJson.h>
#include <PubSubClient.h>

//end code esp32 devkit
        bot.sendMessage(chat_id, "Suhu sudah turun, kipas
dimatikan!", "");
        fanStatus = false;
      }
    }
  }
}
  message += "Kipas: " + String(fanStatus ? "ON" : "OFF") + "\n";
  message += "Lampu: " + String(lampStatus ? "ON" : "OFF") +
"\n";
  message += "Pump: " + String(mini pumpStatus ? "ON" : "OFF") +
"\n";

  bot.sendMessage(chat_id, message, "");
}

void handleFanControl(String text) {
  if (text == "/fan_on") {
```

```
const char* ssid = "realme 3 Pro";
const char* password = "12345678";

// const char* ssid = "Undiksha 2024";
// const char* password = "Aruhgung123";

const char* mqtt_server = "192.168.115.96";
const char* mqtt_user = "AdminMQTT";
const char* mqtt_pass = "pwd123";

// Initialize Telegram BOT
String BOTtoken =
"7771068553:AAGfW_TqTMVsa50tU8fyN9iExk_s2WzgZBg";  // your Bot
Token (Get from Botfather)

// Use @myidbot to find out the chat ID of an individual or a
group
// Also note that you need to click "start" on a bot before it
can
// message you
String CHAT_ID = "8152895238";

bool sendPhoto = false;

WiFiClientSecure clientTCP;
UniversalTelegramBot bot(BOTtoken, clientTCP);

WiFiClient espClient;
PubSubClient client(espClient);

#define FLASH_LED_PIN 4
bool flashState = LOW;

//Checks for new messages every 1 second.
int botRequestDelay = 1000;
unsigned long lastTimeBotRan;

//CAMERA_MODEL_AI_THINKER
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27

#define Y9_GPIO_NUM       35
```

```
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22


void configInitCamera(){
  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;
  config.grab_mode = CAMERA_GRAB_LATEST;

  //init with high specs to pre-allocate larger buffers
  if(psramFound()){
    config.frame_size = FRAMESIZE_UXGA;
    config.jpeg_quality = 10;  //0-63 lower number means higher
quality
    config.fb_count = 1;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;  //0-63 lower number means higher
quality
```

```
    if (text == "/start") {
      String welcome = "Welcome , " + from_name + "\n";

      bot.sendMessage(CHAT_ID, welcome, "");
    }
    if (text == "/flash") {
      flashState = !flashState;
      digitalWrite(FLASH_LED_PIN, flashState);
      Serial.println("Change flash LED state");
    }
    if (text == "/photo") {
      sendPhoto = true;
      Serial.println("New photo request");
    }
  }
}

String sendPhotoTelegram() {
  const char* myDomain = "api.telegram.org";
  String getAll = "";
  String getBody = "";

  //Dispose first picture because of bad quality
  camera_fb_t * fb = NULL;
  fb = esp_camera_fb_get();
  esp_camera_fb_return(fb); // dispose the buffered image

  // Take a new photo
  fb = NULL;
  fb = esp_camera_fb_get();
  if(!fb) {
    Serial.println("Camera capture failed");
    delay(1000);
    ESP.restart();
    return "Camera capture failed";
  }

  Serial.println("Connect to " + String(myDomain));


  if (clientTCP.connect(myDomain, 443)) {
    Serial.println("Connection successful");

    String head = "--RandomNerdTutorials\r\nContent-Disposition:
form-data; name=\"chat_id\"; \r\n\r\n" + CHAT_ID + "\r\n--
RandomNerdTutorials\r\nContent-Disposition: form-data;
```

```
name=\"chat_id\"; \r\n\r\n" + CHAT_ID + "\r\n--
RandomNerdTutorials\r\nContent-Disposition: form-data;
name=\"photo\"; filename=\"esp32-cam.jpg\"\r\nContent-Type:
image/jpeg\r\n\r\n";
    String tail = "\r\n--RandomNerdTutorials--\r\n";

    size_t imageLen = fb->len;
    size_t extraLen = head.length() + tail.length();
    size_t totalLen = imageLen + extraLen;

    clientTCP.println("POST /bot"+BOTtoken+"/sendPhoto
HTTP/1.1");
    clientTCP.println("Host: " + String(myDomain));
    clientTCP.println("Content-Length: " + String(totalLen));
    clientTCP.println("Content-Type: multipart/form-data;
boundary=RandomNerdTutorials");
    clientTCP.println();
    clientTCP.print(head);

    uint8_t *fbBuf = fb->buf;
    size_t fbLen = fb->len;
    for (size_t n=0;n<fbLen;n=n+1024) {
      if (n+1024<fbLen) {
        clientTCP.write(fbBuf, 1024);
        fbBuf += 1024;
      }
      else if (fbLen%1024>0) {
        size_t remainder = fbLen%1024;
        clientTCP.write(fbBuf, remainder);
      }
    }

    clientTCP.print(tail);

    esp_camera_fb_return(fb);

    int waitTime = 10000;   // timeout 10 seconds
    long startTimer = millis();
    boolean state = false;

    while ((startTimer + waitTime) > millis()){
      Serial.print(".");
      delay(100);
      while (clientTCP.available()) {
        char c = clientTCP.read();
```

```
        if (state==true) getBody += String(c);
        if (c == '\n') {
          if (getAll.length()==0) state=true;
          getAll = "";
        }
        else if (c != '\r')
          getAll += String(c);
        startTimer = millis();
      }
      if (getBody.length()>0) break;
    }
    clientTCP.stop();
    Serial.println(getBody);
  }
  else {
    getBody="Connected to api.telegram.org failed.";
    Serial.println("Connected to api.telegram.org failed.");
  }
  return getBody;
}

void setup(){
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);
  // Init Serial Monitor
  Serial.begin(115200);

  // Set LED Flash as output
  pinMode(FLASH_LED_PIN, OUTPUT);
  digitalWrite(FLASH_LED_PIN, flashState);

  // Config and init the camera
  configInitCamera();

  // Connect to Wi-Fi
  WiFi.mode(WIFI_STA);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  clientTCP.setCACert(TELEGRAM_CERTIFICATE_ROOT); // Add root
certificate for api.telegram.org
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println();
```

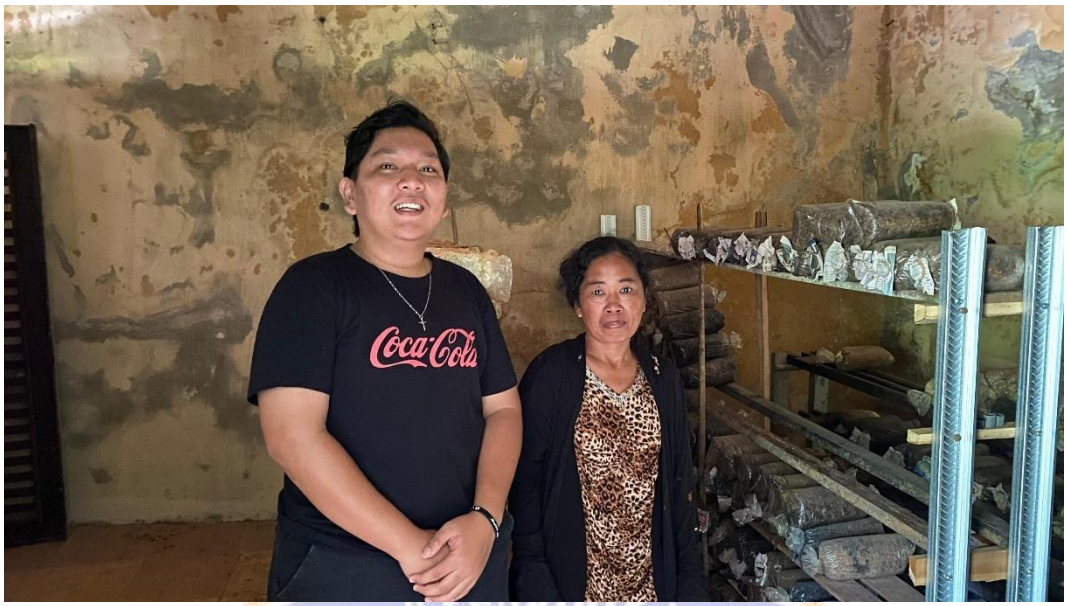
```
Serial.print("ESP32-CAM IP Address: ");
  Serial.println(WiFi.localIP());

  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
  reconnectMqtt();
}

void reconnectMqtt() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    if (client.connect("ESP32Cam", mqtt_user, mqtt_pass)) {
      Serial.println("connected");
      // Subscribe to the topic after successful connection
      client.subscribe("mushroom/temp-alert");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}

void loop() {
  // Check if still connected to MQTT, if not, reconnect
  if (!client.connected()) {
    reconnectMqtt();
  }
  client.loop();  // Maintain MQTT connection

  if (sendPhoto) {
    Serial.println("Preparing photo");
    sendPhotoTelegram();
    sendPhoto = false;

  }

  if (millis() > lastTimeBotRan + botRequestDelay)  {
    int numNewMessages = bot.getUpdates(bot.last_message_received +
1);
    while (numNewMessages) {
      Serial.println("Got response");
      handleNewMessages(numNewMessages);
      numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    }
    lastTimeBotRan = millis();
  }
}
```

Lampiran 3 Wawancara

MAHASISWA	NARASUMBER
Buk, biasanya gimana cara Babuk memantau suhu, kelembaban, dan cahaya di ruang budidaya jamur?	Selama ini saya lihat langsung aja, Dik. Masuk ke ruangan, cek bukai termometer sama alat seadanya. Kadang cuma kira-kira aja.
Ada kendala nggak Buk kalau harus pantau terus secara manual?	Ada, Dik. Kalau lagi keluar rumah atau sibuk, ya saya nggak bisa mantau. Pernah juga suhu naik, kelembaban turun, tapi saya baru tau pas udah terlambat.
Kalau suhu di ruangan terlalu tinggi, biasanya dambuknya apa ke jamurnya?	Jamurnya jadi layu, nggak tumbuh sempurna. Kadang bisa busuk juga. Kalau terlalu sering, ya bisa gagal panen.
Kalau soal penyiraman, Babuk siram media jamur tiap hari atau nunggu kering dulu?	Biasanya saya siram pagi sama sore, Dik. Tapi kalau lagi sibuk ya bisa lupa. Akhirnya medianya jadi kering.
Menurut ibuk, pencahayaan di ruang budidaya udah cukup belum?	Kalau siang dan cerah sih cukup, Dik. Tapi kalau mendung atau malam, ya gelap. Kasihan jamurnya kurang cahaya.

MAHASISWA	NARASUMBER
Kalau ada alat yang bisa pantau suhu, kelembaban, dan cahaya dari HP, kira-kira Babuk tertarik nggak?	Wah, tertarik banget, Dik. Bisa bantu banget. Jadi saya nggak perlu bolak-balik cek ke ruangan.
Gimana kalau alat itu juga bisa nyalain kipas kalau suhu panas, siram otomatis kalau kelembaban turun, dan nyalain lampu kalau cahaya kurang?	Wah, mantap itu, dik. Lengkap banget. Kalau gitu saya bisa lebih tenang, nggak khawatir ninggalin ruangan lama-lama.
Kalau begitu, boleh nggak Buk kalau nanti alat ini dicoba dulu di tempat Babuk? Sekalian kita uji coba alatnya.	Boleh banget, dik. Saya malah seneng bisa bantu dan lihat hasilnya langsung.

Lampiran 4 Dokumentasi Wawancara



RIWAYAT HIDUP

Andre Sebastian Purba, penulis skripsi berjudul **"Rancang Bangun Sistem Pengukur Suhu, Kelembaban, Dan Intensitas Cahaya Pada Ruangan Budidaya Jamur Tiram Berbasis *Internet Of Things* (IoT)",** lahir di Singaraja pada 11 Juli 2002. Ia merupakan putra dari pasangan Bapak Dr. (Cand.) Ir. Jhon Hardy Purba, M.P. dan Ibu Risma Saurida Sitorus Pane. Penulis beragama Kristen, dan saat ini orang tua penulis berdomisili di Desa Banyuning, Kecamatan Buleleng, Kabupaten Buleleng, Provinsi Bali. Pendidikan formalnya dimulai di SD 3 Banjar Jawa. Kemudian, penulis melanjutkan ke SMP Negri 2 Singaraja. Pendidikan menengah ke atas ditempuh di SMA Negeri 4 Singaraja. Saat pembuatan karya ini, penulis sedang menempuh pendidikan tinggi di Universitas Pendidikan Ganesha, pada Program Studi S1 Ilmu Komputer.