LAMPIRAN

Lampiran 1 *Source Code* Program

```
#include <HX711.h>
#include <ESP32Servo.h>
#include <MFRC522.h>
#include <Ultrasonic.h>
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>

// ============================== KONFIGURASI WIFI DAN BOT TELEGRAM
==============================
const char* ssid = "ZTE_2.4G_Z9eqkY";
const char* password = "C6z4ux6y";

const char* botToken =
"6286007752:AAEPw8FSbmicXJABHSjCwZzbgLt6fwyiQnU";
String CHAT_ID = "1661122449";  //Isi Initial ID telegram pemilik pakan
// ============================== END KONFIGURASI WIFI DAN BOT TELEGRAM
==============================



// ========================= KONFIGURASI STATE MANAGEMENT (STATE
MACHINE) =========================
#define STAND_BY 0
#define OPEN_SERVO 1
#define CLOSE_SERVO 2
#define SEND_START_MESSAGE 3
#define SEND_INFO_MESSAGE 4
#define SEND_STATUS_MESSAGE 5
#define SEND_EMPTY_STOCK_MESSAGE 6
int CURRENT_STATE = 0;
// ========================= END KONFIGURASI STATE MANAGEMENT (STATE
MACHINE) ====================



// ================================== KONFIGURASI PIN SENSOR
====================================
const int RFID_SS_PIN = 21;             //PIN RFID SS
const int RFID_RST_PIN = 22;            //PIN RFID RST
```

```
const int ULTRASONIC_ECHO_PIN = 12;     //PIN Sensor Jarak (ECHO)
const int ULTRASONIC_TRIGGER_PIN = 14;  //PIN Sensor Jarak (Trigger)
const int LED_RED = 33;                 //lampu LED untuk pakan Habis
const int LED_GREEN = 32;               //lampu LED untuk pakan masih
terisi
const int SERVO_PIN = 27;               //Pin Servo
const int LOADCELL_SCK_PIN = 26;        //PIN Sensor Berat
const int LOADCELL_DOUT_PIN = 25;       //PIN Sensor Berat
// ================================= END KONFIGURASI PIN SENSOR
================================


// ======================== KONFIGURASI CORONG PAKAN (Untuk Cari
Volume) ========================
const float pi = 3.14159;          // Nilai pi
float CONE_HEIGHT = 14.0;          // Tinggi Corong dari dasar coreong
ke permukaan (ujung sensor jarak)
float CONE_DIAMETER = 20.0;        // Diameter Corong
float CONE_R = CONE_DIAMETER / 2;  // Jari - Jari Corong
float CONE_VOLUME = 0;             // Penampung Nilai Volume Pakan
float CONE_HEIGHT_CUT = 0.0;       // Tinggi Corong yang di pangkas
(akibat pembacaan sensor jarak, untuk deteksi jumlah pakan)
// ===================== END KONFIGURASI CORONG PAKAN (Untuk Cari
Volume) ========================


// ==================== KONFIGURASI INITIAL VALUE / NILAI AWAL
==================================
const int SCALE_FACTOR_LOAD_CELL = 420.30;  // Configurasi Load Cell
untuk kalibrasi berat
bool SERVO_IS_OPEN = false;                 // FLAGING SERVO
const int SERVO_OPEN_ANGLE = 30;            // Nilai derajat servo
terbuka
const int SERVO_CLOSE_ANGLE = 0;            // Nilai derajat servo
tertutup
long jarakPenuh = 4;                        //setting jarak (dalam Cm)
pakan saat Penuh, Ujung sensor jarak ke pakan (batas limit dikatakan
corong pakan penuh)
long jarakKosong = 12;                      //setting batas jarak
(dalam cm) pakan dikatakan habis, Ujung sensor jarak ke dasar corong
pakan (batas limit dikatakan corong pakan habis)
```

```
float minimumLoadCellValue = 50;            //setting berat (dalam
Gram) dikatakan pakan kosong
bool isRfidDetected = false;                //flagging untuk pembacaan
RFID
float loadCellValue = 0;                    //Variable Penampung berat
pakan
long distance = 0;                          //Variable Penampung Sensor
Jarak
int numNewMessages = 0;                     //Variable Penampang jumlah
pesan masuk dari telegram
String statusPakan = "";                    //Variable Penampung Status
Pakan pada corong
bool isEmptyStock = false;
// ====================== END  KONFIGURASI INITIAL VALUE / NILAI AWAL
==============================

// ====================== KONFIGURASI NILAI MILLIS Pengganti Delay
================================
unsigned long previousLoadcellTime = 0;        //Variable penampung
nilai milis delay sensor berat sebelumnya
const unsigned long loadcellInterval = 100;    // Interval pembacaan
sensor loadcell (dalam milidetik)
unsigned long previousRfidTime = 0;            //Variable penampung
nilai milis delay sensor RFID sebelumnya
const unsigned long rfidInterval = 100;        // Interval pembacaan
sensor RFID (dalam milidetik)
unsigned long previousUltrasonicTime = 0;      //Variable penampung
nilai milis delay sensor Jarak sebelumnya
const unsigned long ultrasonicInterval = 100;  // Interval pembacaan
sensor ultrasonik (dalam milidetik)
unsigned long previousMoveTime = 0;            //Variable penampung
nilai milis delay Servo sebelumnya
const unsigned long moveDelay = 200;           //Interval Delay Servo
(Ubah nilai nya untuk mempercepat/ melambatkan servo kembali menutup)
// ====================== END KONFIGURASI NILAI MILLIS Pengganti Delay
============================


HX711 loadcell;
Servo servo;
MFRC522 rfid(RFID_SS_PIN, RFID_RST_PIN);
Ultrasonic ultrasonic(ULTRASONIC_TRIGGER_PIN, ULTRASONIC_ECHO_PIN);
WiFiClientSecure client;
```

```
UniversalTelegramBot bot(botToken, client);

void setup() {
  Serial.begin(9600);

  loadcell.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);
  loadcell.set_scale(SCALE_FACTOR_LOAD_CELL);
  loadcell.tare();

  SPI.begin();
  rfid.PCD_Init();

  pinMode(ULTRASONIC_TRIGGER_PIN, OUTPUT);
  pinMode(ULTRASONIC_ECHO_PIN, INPUT);
  pinMode(LED_RED, OUTPUT);
  pinMode(LED_GREEN, OUTPUT);
  digitalWrite(LED_RED, LOW);
  digitalWrite(LED_GREEN, LOW);

  servo.attach(SERVO_PIN);

#ifdef ESP32
  client.setCACert(TELEGRAM_CERTIFICATE_ROOT);  // Add root certificate
for api.telegram.org
#endif

  // Menghubungkan ke Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi!");
  CONE_VOLUME = coneVolume(distance);
}

void loop() {

  readAllSensor();
  readTelegramMessage();
  handleLedIndicator();
  handleStatusPakan();
  handleServo();
```

```
  Serial.println("CURRENT_STATE: " + String(CURRENT_STATE));

  switch (CURRENT_STATE) {
    case STAND_BY:
      if (isRfidDetected == true) {
        Serial.println("RFID card detected");
        // kirim pesan kucing terdeteksi
        bot.sendMessage(CHAT_ID, "Kucing Peliharaan Terdeteksi");
      }
      if (distance < jarakKosong && isRfidDetected == true &&
loadCellValue <= minimumLoadCellValue && isEmptyStock == false) {
        CURRENT_STATE = OPEN_SERVO;
      }
      if (distance >= jarakKosong && isEmptyStock == false) {
        isEmptyStock = true;
        CURRENT_STATE = SEND_EMPTY_STOCK_MESSAGE;
      }
      if (distance < jarakKosong){
        isEmptyStock = false;
      }
      break;
    case SEND_START_MESSAGE:
      // mengirim pesan siap terima perintah
      bot.sendMessage(CHAT_ID, "Halo! Bot siap menerima perintah.");
      CURRENT_STATE = STAND_BY;
      break;
    case SEND_INFO_MESSAGE:
      // mengirim jumlah pakan pada wadah pakan (sensor load cell)
      bot.sendMessage(CHAT_ID, "Berat Pakan Pada Mangkuk: " +
String(loadCellValue) + " gram.");
      CURRENT_STATE = STAND_BY;
      break;
    case SEND_STATUS_MESSAGE:
      // mengirim pesan jumlah pakan (sensor ultrasonik)
      bot.sendMessage(CHAT_ID, "Jumlah Pakan Tersedia : " +
String(CONE_VOLUME) + " cm²" + " (" + statusPakan + ")." + "\n" +
"Berat Pakan Pada Mangkuk: " + String(loadCellValue) + " gram.");
      CURRENT_STATE = STAND_BY;
      break;
    case SEND_EMPTY_STOCK_MESSAGE:
      // mengirim Pesan Stok Pakan Habis
```

```
      bot.sendMessage(CHAT_ID, "Pakan pada penampung habis, silahkan
isi ulang!.");
      CURRENT_STATE = STAND_BY;
      break;
    case OPEN_SERVO:
      SERVO_IS_OPEN = true;
      servo.write(SERVO_OPEN_ANGLE);
      previousMoveTime = millis();
      // mengirim pesan servo terbuka;
      bot.sendMessage(CHAT_ID, "Servo Terbuka");
      CURRENT_STATE = CLOSE_SERVO;
      break;
    case CLOSE_SERVO:
      if (millis() - previousMoveTime >= moveDelay) {
        servo.write(SERVO_CLOSE_ANGLE);  // Return to the initial
position
        previousMoveTime = millis();
        SERVO_IS_OPEN = false;
        bot.sendMessage(CHAT_ID, "Servo Tertutup");
        CURRENT_STATE = STAND_BY;
      } else {
        servo.write(SERVO_OPEN_ANGLE);
      }
      break;
    default:
      CURRENT_STATE = STAND_BY;
      break;
  }

  handleServo();

}

void readAllSensor() {
  // Baca sensor loadcell setiap loadcellInterval
  if (millis() - previousLoadcellTime >= loadcellInterval) {
    previousLoadcellTime = millis();
    loadCellValue = readLoadcell();
  }

  // Baca sensor RFID setiap rfidInterval
  if (millis() - previousRfidTime >= rfidInterval) {
    previousRfidTime = millis();
```

```
    isRfidDetected = readRfid();
  }

  // Baca sensor ultrasonik setiap ultrasonicInterval
  if (millis() - previousUltrasonicTime >= ultrasonicInterval) {
    previousUltrasonicTime = millis();
    distance = readUltrasonic();
  }
}

void readTelegramMessage() {
  // baca pesan perintah baru dari telegram
  numNewMessages = bot.getUpdates(bot.last_message_received + 1);
  while (numNewMessages) {
    handleNewMessages(numNewMessages);
    numNewMessages = bot.getUpdates(bot.last_message_received + 1);
  }
}

// Fungsi untuk menangani perintah dari Telegram
void handleNewMessages(int numNewMessages) {
  for (int i = 0; i < numNewMessages; i++) {
    String chatId = bot.messages[i].chat_id;
    CHAT_ID = chatId;
    String text = bot.messages[i].text;

    // Lakukan sesuatu berdasarkan perintah yang diterima
    if (text == "/start") {
      CURRENT_STATE = SEND_START_MESSAGE;
    } else if (text == "/info") {
      CURRENT_STATE = SEND_INFO_MESSAGE;
    } else if (text == "/status") {
      CURRENT_STATE = SEND_STATUS_MESSAGE;
    } else if (text == "/feed") {
      if(isEmptyStock == false) {
        CURRENT_STATE = OPEN_SERVO;
      }else{
        CURRENT_STATE = SEND_EMPTY_STOCK_MESSAGE;
      }

    } else {
      bot.sendMessage(CHAT_ID, "Perintah tidak dikenali.");
    }
```

```
  }
}


float readLoadcell() {
  float loadValue = loadcell.get_units();
  Serial.print("Load: ");
  Serial.print(loadValue);
  Serial.println(" g");
  return loadValue;
}

bool readRfid() {
  if (rfid.PICC_IsNewCardPresent() && rfid.PICC_ReadCardSerial()) {
    rfid.PICC_HaltA();
    rfid.PCD_StopCrypto1();
    return true;
  } else {
    return false;
  }
}


long readUltrasonic() {
  long duration, distance;

  digitalWrite(ULTRASONIC_TRIGGER_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(ULTRASONIC_TRIGGER_PIN, HIGH);

  delayMicroseconds(10);
  digitalWrite(ULTRASONIC_TRIGGER_PIN, LOW);
  duration = pulseIn(ULTRASONIC_ECHO_PIN, HIGH);
  distance = duration * 0.034 / 2;

  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  return distance;
}

void handleServo() {
```

```
  // Check if it's time to return to the initial position
  if (SERVO_IS_OPEN == true) {
    if (millis() - previousMoveTime >= moveDelay) {
      servo.write(SERVO_CLOSE_ANGLE);  // Return to the initial
position
      previousMoveTime = millis();
      SERVO_IS_OPEN = false;
      bot.sendMessage(CHAT_ID, "Servo Tertutup");
      CURRENT_STATE = STAND_BY;
    } else {
      servo.write(SERVO_OPEN_ANGLE);
    }
  } else {
    servo.write(SERVO_CLOSE_ANGLE);
  }
}

float coneVolume(float distanceValue) {
  CONE_HEIGHT_CUT = CONE_HEIGHT - distanceValue;
  // Hitung jari-jari dan tinggi awal
  CONE_R = CONE_DIAMETER / 2;

  // Hitung volume awal kerucut
  CONE_VOLUME = (1.0 / 3) * pi * pow(CONE_R, 2) * CONE_HEIGHT;

  // Hitung diameter, tinggi, dan volume setelah dipangkas
  CONE_DIAMETER = CONE_DIAMETER - 2 * CONE_HEIGHT_CUT;
  CONE_R = CONE_DIAMETER / 2;
  CONE_HEIGHT = CONE_HEIGHT - CONE_HEIGHT_CUT;

  // Hitung volume setelah dipangkas
  CONE_VOLUME = (1.0 / 3) * pi * pow(CONE_R, 2) * CONE_HEIGHT;

  return CONE_VOLUME;
}

void handleLedIndicator() {
  if (distance >= jarakKosong) {
    digitalWrite(LED_RED, HIGH);
    digitalWrite(LED_GREEN, LOW);
  } else {
    digitalWrite(LED_RED, LOW);
```

```
    digitalWrite(LED_GREEN, HIGH);
  }
}

void handleStatusPakan() {
  CONE_VOLUME = coneVolume(distance);
  if ((distance <= jarakPenuh) && (CONE_VOLUME > 0.0)) {
    statusPakan = "Penuh";
  } else if ((distance > jarakPenuh) && (distance < (jarakKosong - 2))
&& (CONE_VOLUME > 0.0)) {
    statusPakan = "Sedang";
  } else {
    statusPakan = "Kosong";
  }
}
```

Lampiran 2 Identitas Narasumber

A. Narasumber 1

Nama	Kadek Yota Ernanda Aryanto, S.Kom., M.T., Ph.D
Profesi	Dosen Ilmu Komputer S1-Teknik Informatika Undiksha
Email	#yota.ernanda@undiksha.ac.id

B. Narasumber 2

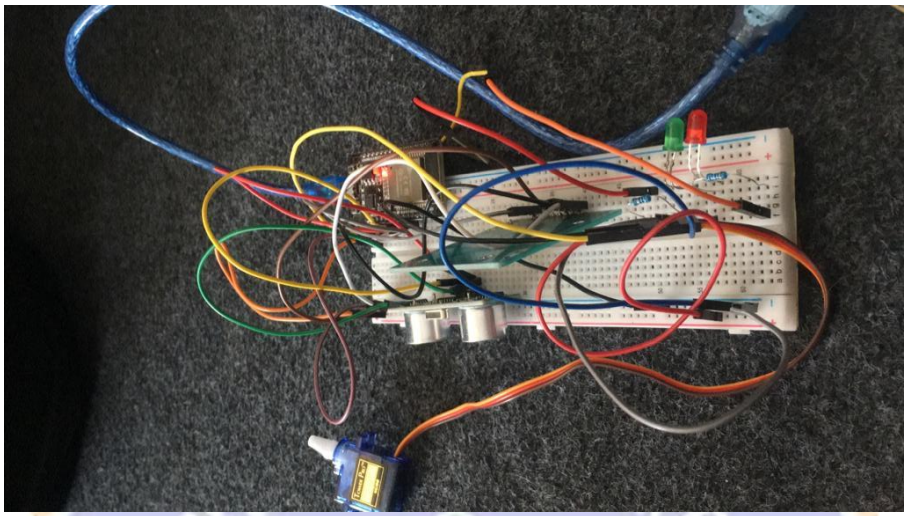
Nama	Kadek Utari Darma Putri
Profesi	Alumni Ilmu Komputer Undiksha
Email	Utaridp15@gmail.com

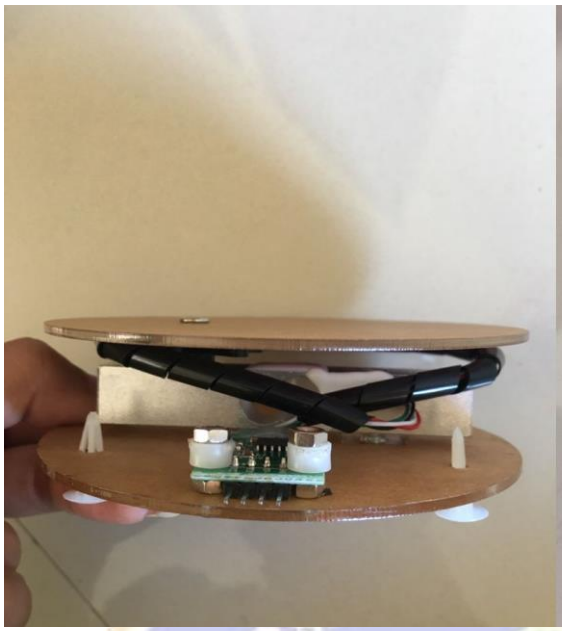
C. Narasumber 3

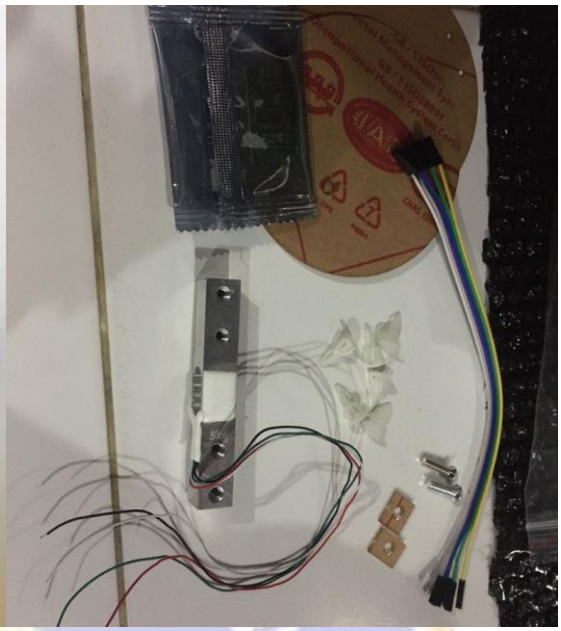
Nama	Dicka Febriansyah
Profesi	Pelajar
Email	dickafebriyansyah16@gmail.com

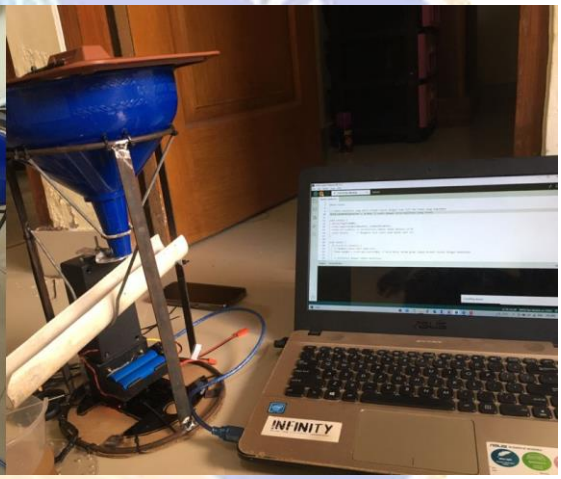
Lampiran 3 Tampilan Akhir Alat

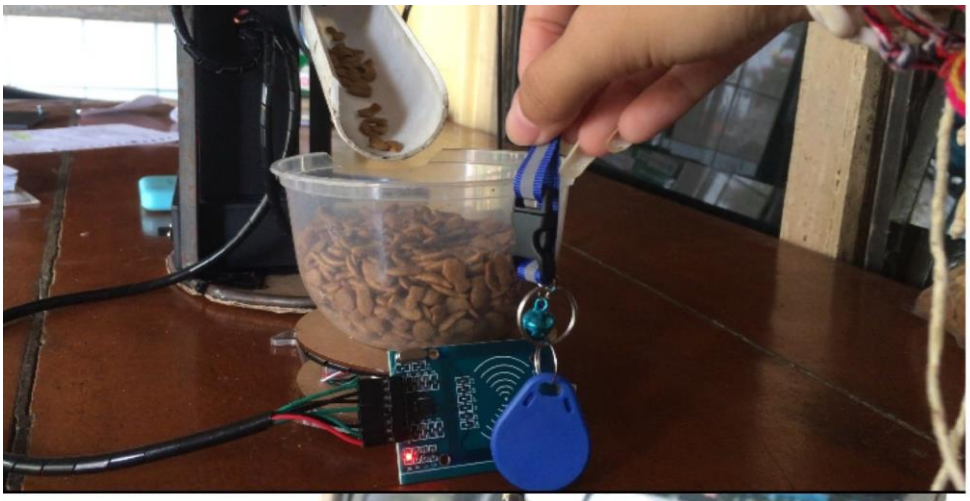
Lampiran 4 Proses Pembuatan Rangkaian Alat Pakan Kucing

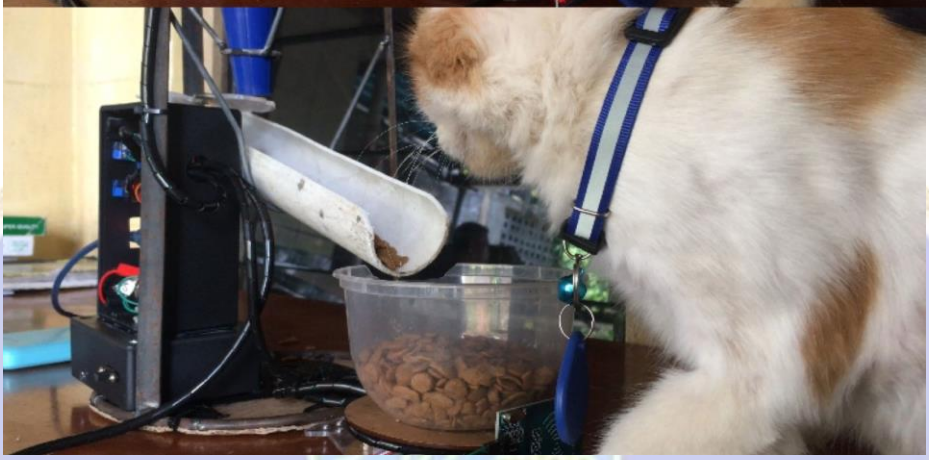







INFINITY





Lampiran 5 Lembar Persetujuan Hasil Perbaikan Skripsi

PERSETUJUAN HASIL PERBAIKAN SKRIPSI SETELAH UJIAN SKRIPSI			
No	**Nama**	**Tanda Tangan**	**Tanggal**
1	**I Ketut Resika Arthana, S.T., M.Kom.** (Pembimbing I)		
2	**Prof. Dr. Komang Setemen, S.Si., M.T.** (Pembimbing II)		
3	**Dr. Ni Ketut Kertiasih, S.Si., M.Pd.** (Penguji I)		
4	**Kadek Yota Ernanda Aryanto, S.Kom., M.T., Ph.D** (Penguji II)		

RIWAYAT HIDUP

Luh Putu Ayu Chandra Dewi lahir di Singaraja pada tanggal 23 September 2000. Penulis lahir dari pasangan suami istri Bapak Made Suyasa dan Ibu Komang Supareni. Penulis berkebangsaan Indonesia dan beragama Hindu. Kini penulis beralamat di Jl. Gajah Mada, Banjar Delod Peken G.IV/12 Kelurahan Kendran, Kabupaten Buleleng. Penulis menyelesaikan pendidikan dasar di SDN 1 Paket Agung dan lulus pada 2013. Kemudian penulis melanjutkan di SMP Negeri 6 Singaraja dan lulus pada tahun 2016. Pada tahun 2019, penulis lulus dari SMKN 3 Singaraja dengan jurusan TKJ. Penulis terdaftar sebagai mahasiswa Program Studi S1 Ilmu Komputer di Universitas Pendidikan Ganesha pada tahun 2019.