LAMPIRAN A

Listing Program ESP32

```
#include <Arduino.h>
#include <ArduinoOTA.h>
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>
#include <SPI.h>
#include <HX711.h>
#include <MFRC522.h>
#include <LiquidCrystal_I2C.h>
#include <Keypad.h>
#include <string.h>
#include <Ticker.h>
#include <EEPROM.h>

// Definisi pinout untuk RFID
#define SCK_PIN 14  // Pin SCK untuk RFID
#define MOSI_PIN 13 // Pin MOSI untuk RFID
#define MISO_PIN 12 // Pin MISO untuk RFID
#define SS_PIN 15   // Pin SS/CS (SDA) untuk RFID
#define RST_PIN 4   // Pin RST untuk RFID

// Definisi pinout untuk load cell
#define LOADCELL_DOUT_PIN 16 // Pin data output load cell
#define LOADCELL_CLK_PIN 17  // Pin clock load cell

// Definisi pinout untuk tombol
#define START_PIN 5 // Pin tombol START
#define SAVE_PIN 19 // Pin tombol SAVE

// Definisi pinout untuk sensor ultrasonik
#define TRIGER_PIN 32 // Pin trigger untuk sensor ultrasonik
#define ECHO_PIN 33   // Pin echo untuk sensor ultrasonik

// Definisi pinout untuk LCD
#define SDA_PIN 21 // Pin SDA untuk I2C LCD
#define SCL_PIN 22 // Pin SCL untuk I2C LCD
```

```
// Membuat objek LCD dengan alamat I2C 0x27, 20 kolom dan 4 baris
LiquidCrystal_I2C lcd(0x27, 20, 4);

// Membuat objek RFID
MFRC522 rfid(SS_PIN, RST_PIN);

// Membuat objek load cell
HX711 scale;

// Membuat objek Ticker
Ticker ticker;


bool taskRfid = false; // Flag untuk tugas RFID

#define WIFI_SSID "XX"       // Nama SSID WiFi
#define WIFI_PASS "11111111" // Kata sandi WiFi
// #define WIFI_SSID "Tes"       // Nama SSID WiFi
// #define WIFI_PASS "katasandi" // Kata sandi WiFi

// Variabel untuk Google Spreadsheet
String GAS_ID =
"AKfycbxZswPeHSroSx0UVnBJtW3nxp3nivEeT7SJ1GsSuT9MhgJM
BGs8acc8qAEvy3CsU-NWFA"; // ID Google Apps Script
const char *host = "script.google.com";

#define MAX_CONN_FAIL 50          // Maksimal jumlah kegagalan
koneksi
#define MAX_LENGTH_WIFI_SSID 31 // Panjang maksimal SSID
WiFi
#define MAX_LENGTH_WIFI_PASS 63 // Panjang maksimal kata sandi
WiFi
#define PIN_LED LED_BUILTIN     // Pin LED bawaan

#define BOT_TOKEN "7252354301:AAHbMumwR6-
hxRNCGzoX_0BLz3jrchLHdi0" // Token bot Telegram
```

```
const unsigned long BOT_MTBS = 5000;                    // Mean Time Between Scans (waktu rata-rata antara pemindaian pesan)

WiFiClientSecure secured_client;                        // Klien WiFi dengan koneksi aman
UniversalTelegramBot bot(BOT_TOKEN, secured_client); // Membuat objek bot Telegram
unsigned long bot_lasttime;                             // Waktu terakhir pemindaian pesan
bool Start = false;                                     // Variabel untuk status awal
#define CHAT_ID "7162764022"                            // ID chat Telegram untuk testing
// #define CHAT_ID "1999837590"                         // ID chat Telegram untuk testing

struct UserData
{
   char uid[10];
   char name[50];
   char birthDate[20];
   char address[100];
   char parentName[50];
};

// Daftar UID yang diinginkan
UserData desiredUids[] = {
    {{0xF0, 0x97, 0xF7, 0x8B}, "Aleta Mikanandira", "03/06/2021", "RT01/06", "Istiyani"},
     {{0xA1, 0xB8, 0xF6, 0x8B}, "Arusha Kaiden", "10/10/2019", "RT02/06", "Tumiati"},
     {{0x14, 0x6F, 0xF6, 0x8B}, "Naura Kaniya", "21/08/2021", "RT03/06", "Dewi Laraswati"},
     {{0x94, 0xB8, 0xB4, 0x2C}, "Alfeisya Cayra", "23/10/2021", "RT03/06", "Aji Rahayu"},
      {{0x94, 0x82, 0xF7, 0x8B}, "Alfira Sahlan", "27/07/2019", "RT03/06", "Aji Rahayu"}};

const int EEPROM_SIZE = 512;
```

```
bool registerMode = false;
String cardName = "";
String birthDate = "";
String address = "";
String parentName = "";
int lastRegisteredAddress = -1;

// Prototipe fungsi
void initializeRFID();
void registerCard();
   rfid.PCD_Init();                    // Inisialisasi modul RFID

   // Inisialisasi LCD
   lcd.init();      // Inisialisasi LCD
   lcd.backlight(); // Nyalakan backlight LCD

   // Inisialisasi load cell
    scale.begin(LOADCELL_DOUT_PIN, LOADCELL_CLK_PIN); // Mulai load cell dengan pin data dan clock
   scale.set_scale(calibration_factor);          // Set faktor kalibrasi untuk load cell
   scale.tare();                         // Tare load cell

   // Inisialisasi sensor ultrasonik
   pinMode(trigPin, OUTPUT);        // Set pin trigger sebagai output
   pinMode(echoPin, INPUT);         // Set pin echo sebagai input
   pinMode(START_PIN, INPUT_PULLUP); // Set pin tombol START sebagai input dengan pull-up
    pinMode(SAVE_PIN, INPUT_PULLUP);  // Set pin tombol SAVE sebagai input dengan pull-up

   WiFi.disconnect(true);                                          // Memutuskan koneksi WiFi yang ada
                              WiFi.onEvent(WiFiStationConnected, WiFiEvent_t::ARDUINO_EVENT_WIFI_STA_CONNECTED);       // Menetapkan event untuk koneksi WiFi
```

```
                                        WiFi.onEvent(WiFiGotIP,
WiFiEvent_t::ARDUINO_EVENT_WIFI_STA_GOT_IP);                    //
Menetapkan event untuk mendapatkan IP
                                    WiFi.onEvent(WiFiStationDisconnected,
WiFiEvent_t::ARDUINO_EVENT_WIFI_STA_DISCONNECTED);  //
Menetapkan event untuk putus dari WiFi
  WiFi.mode(WIFI_STA);                                          //
Mengatur mode WiFi sebagai STATION
  WiFi.begin(WIFI_SSID, WIFI_PASS);

  while (WiFi.status() != WL_CONNECTED)
  {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  secured_client.setCACert(TELEGRAM_CERTIFICATE_ROOT);
  delay(3000);

  Serial.println("Connected to WiFi");
   EEPROM.begin(EEPROM_SIZE); // Memulai EEPROM dengan
ukuran yang ditentukan
  initializeEEPROM();
  bot_lasttime = millis();

  // Menampilkan pesan selamat datang di LCD
  lcd.setCursor(3, 0);        // Set posisi kursor LCD
  lcd.print("SELAMAT DATANG"); // Tampilkan pesan
  delay(2000);               // Tunggu 2 detik
  lcd.clear();               // Bersihkan LCD
}

void loop()
{
  if (millis() - bot_lasttime > BOT_MTBS)
  {
    int numNewMessages = bot.getUpdates(bot.last_message_received
+ 1);
    while (numNewMessages)
```

```
      {
        handleNewMessages(numNewMessages);
        numNewMessages = bot.getUpdates(bot.last_message_received +
1);
      }
      bot_lasttime = millis();
    }

    if (registerMode)
    {
      lcd.setCursor(0, 0);
      lcd.print(F(" MODE PENDAFTARAN ID"));
      lcd.setCursor(0, 1);
      lcd.print(F("IKUTI PESAN TELEGRAM"));
      registerCard();
    }
    else
    {
      checkCard();
    }
    }

    if (taskPengukuran)
    {
      readUltrasonik(); // Baca sensor ultrasonik
      readLoadCell();   // Baca load cell

      // Hitung IMT
      float imt = calculateIMT(tinggi, berat);
      String klasifikasi_imt = classifyIMT(imt);

      // Tampilkan hasil pengukuran di LCD

      lcd.setCursor(0, 0);
      lcd.print("Tinggi: ");
      lcd.print(tinggi);
      lcd.setCursor(16, 0);
      lcd.print(" cm");
```

```
        lcd.setCursor(0, 1);
        lcd.print("Berat : ");
        lcd.print(berat, 2);
        lcd.setCursor(15, 1);
        lcd.print(" Kg");

        lcd.setCursor(0, 2);
        lcd.print("IMT: ");
        lcd.print(imt, 2);

        lcd.setCursor(0, 3);
        lcd.print(klasifikasi_imt);
    }

    else if (!taskRfid && !registerMode)
    { // Jika tugas RFID tidak aktif
        // Tampilkan pesan di LCD untuk menempelkan kartu
        lcd.setCursor(2, 0);
        lcd.print(F("TEMPELKAN KARTU"));
        lcd.setCursor(5, 1);
        lcd.print(F("DEKAT RFID"));
        lcd.setCursor(4, 2);
        lcd.print(F("UNTUK MEMULAI"));
        lcd.setCursor(5, 3);
        lcd.print(F("PENGUKURAN"));
    }
}

// Fungsi event handler untuk diskoneksi WiFi
void WiFiStationDisconnected(WiFiEvent_t event, WiFiEventInfo_t
info)
{
    Serial.println("Disconnected from WiFi access point");
    Serial.print("WiFi lost connection. Reason: ");
    Serial.println(info.wifi_sta_disconnected.reason);
    Serial.println("Trying to Reconnect");
    WiFi.begin(WIFI_SSID, WIFI_PASS);
```

```
}

void readLoadCell()
{
   // Membaca berat dari load cell
   berat = scale.get_units();
   berat = round(berat * 1000) / 1000; // Bulatkan berat ke tiga desimal

   // Jika berat negatif, set menjadi 0
   if (berat <= 0.002)
   {
      berat = 0;
   }

   // Jika berat sama dengan berat sebelumnya
   if (berat == prevWeight)
   {
      sameReadingCount++; // Tambah jumlah pembacaan yang sama
   }
   else
   {
      sameReadingCount = 0; // Reset jumlah pembacaan yang sama
   }

   // Jika berat sama selama 5 kali pembacaan dan berat ditahan
   if (sameReadingCount >= 5 && !holdWeight)
   {
      digitalWrite(LED_BUILTIN, HIGH); // Nyalakan LED bawaan
      holdWeight = true;               // Tahan berat
   }
   else
   {
      digitalWrite(LED_BUILTIN, LOW); // Matikan LED bawaan
   }

   if (holdWeight && berat == 0)
   {
      holdWeight = false; // Lepaskan penahanan berat jika berat 0
```

```
    }
    if (holdWeight)
    {
      berat = prevWeight; // Set berat ke berat sebelumnya
    }
    prevWeight = berat; // Update berat sebelumnya
}

void readUltrasonik()
{
    if (millis() >= delayUltrasonik)
    {
      delayUltrasonik = millis() + 300; // Set delay 300ms
      // Bersihkan pin trigger
      digitalWrite(trigPin, LOW);
      delayMicroseconds(2);
      // Set pin trigger ke HIGH selama 10 microsecond
      digitalWrite(trigPin, HIGH);
      delayMicroseconds(10);
      digitalWrite(trigPin, LOW);

      // Membaca pin echo, mengembalikan waktu perjalanan gelombang
suara dalam microsecond
      long duration = pulseIn(echoPin, HIGH);
      float distanceCm = duration * SOUND_SPEED / 2; // Hitung jarak
      float height = tingggiSensor - distanceCm;     // hitung tinggi
      tinggi = height;                               // Update variabel tinggi
    }
}

float calculateIMT(float tinggi, float berat)
{
    // Menghitung IMT (Indeks Massa Tubuh)
    return berat / ((tinggi / 100) * (tinggi / 100));
}

String classifyIMT(float imt)
{
```

```
    // Klasifikasi IMT sesuai gambar
    if (imt < 17)
    {
      return "Kurus Berat";
    }
    else if (imt >= 17 && imt < 18.4)
    {
      return "Kurus Ringan";
    }
    else if (imt >= 18.5 && imt <= 25)
    {
      return "Normal";
    }
    else if (imt >= 25.1 && imt <= 27)
    {
      return "Gemuk Ringan";
    }
    else
    {
      return "Gemuk Berat";
    }
}

void initializeEEPROM()
{
    // Memeriksa apakah EEPROM sudah diinisialisasi
    if (EEPROM.read(0) != 0xAA)
    {
      Serial.println("Initializing EEPROM...");
      for (int i = 0; i < EEPROM_SIZE; i++)
      {
        EEPROM.write(i, 0xFF);
      }
        EEPROM.write(0, 0xAA); // Tanda bahwa EEPROM sudah
diinisialisasi
      EEPROM.commit();
    }
}
```

```
void resetEEPROM()
{
= getUID();

      // Cek apakah kartu sudah terdaftar
      UserData userData;
                        if      (isCardRegistered(uidString,      userData)      ||
isDesiredUID(rfid.uid.uidByte, rfid.uid.size, userData))
      {
         bot.sendMessage(CHAT_ID, "Card is already registered.");
         registerMode = false;
         return;
      }

      // Meminta informasi tambahan dari pengguna
     bot.sendMessage(CHAT_ID, "Enter the name using command /nama
<nama balita>");

      // Tunggu pengguna untuk memasukkan nama melalui Telegram
      while (cardName.length() == 0)
      {
         if (millis() - bot_lasttime > BOT_MTBS)
         {
                                                int      numNewMessages      =
bot.getUpdates(bot.last_message_received + 1);
            while (numNewMessages)
            {
               handleMessagesTele(numNewMessages);
                                                      numNewMessages      =
bot.getUpdates(bot.last_message_received + 1);
            }
            bot_lasttime = millis();
         }
         delay(100);
      }
         }
         delay(100);
```

```
    }

     bot.sendMessage(CHAT_ID, "Enter the address using command
/alamat <alamat>");
    while (address.length() == 0)
    {
      if (millis() - bot_lasttime > BOT_MTBS)
      {
                                          int     numNewMessages     =
bot.getUpdates(bot.last_message_received + 1);
        while (numNewMessages)
        {
          handleMessagesTele(numNewMessages);
                                                  numNewMessages     =
bot.getUpdates(bot.last_message_received + 1);
        }
        bot_lasttime = millis();
      }
      delay(100);
    }

      bot.sendMessage(CHAT_ID, "Enter the parent's name using
command /orang_tua <nama orang tua>");
    while (parentName.length() == 0)
    {
      if (millis() - bot_lasttime > BOT_MTBS)
      {
                                          int     numNewMessages     =
bot.getUpdates(bot.last_message_received + 1);
        while (numNewMessages)
        {
          handleMessagesTele(numNewMessages);
                                                  numNewMessages     =
bot.getUpdates(bot.last_message_received + 1);
        }
        bot_lasttime = millis();
      }
      delay(100);
```

```
    }

    name.trim();
    dob.trim();
    addr.trim();
    parent.trim();

    // Menyimpan data di EEPROM
      memset(&userData, 0, sizeof(UserData)); // Inisialisasi userData
dengan 0
    uidString.toCharArray(userData.uid, 10);
    name.toCharArray(userData.name, 50);
    dob.toCharArray(userData.birthDate, 20);
    addr.toCharArray(userData.address, 100);
    parent.toCharArray(userData.parentName, 50);

    int eepromAddress = findEmptySlot();
    if (eepromAddress != -1)
    {
       EEPROM.put(eepromAddress, userData);
       EEPROM.commit();
       lastRegisteredAddress = eepromAddress;
       bot.sendMessage(CHAT_ID, "Card registered successfully.");
       lcd.clear();
    }
    else
    {
       lcd.clear();
        bot.sendMessage(CHAT_ID, "EEPROM is full. Cannot register
more cards.");
    }

void checkCard()
{
```

```
  if (rfid.PICC_IsNewCardPresent() && rfid.PICC_ReadCardSerial())
  {
    String uidString = getUID();

      lcd.print("Nama: " + String(userData.name));
      lcd.setCursor(0, 1);
    else
    {
      Serial.println("Card is not registered.");
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("  MAAF KARTU SALAH  ");
      delay(5000);      // Tunggu 2 detik
      taskRfid = false; // Nonaktifkan tugas RFID
    }
  }
}

String getUID()
{
  String uidString = "";
  for (byte i = 0; i < rfid.uid.size; i++)
  {
    uidString += String(rfid.uid.uidByte[i] < 0x10 ? "0" : "");
    uidString += String(rfid.uid.uidByte[i], HEX);
  }
  uidString.toUpperCase();
  Serial.print("UID tag :");
  Serial.println(uidString);
  return uidString;
}
```

```
    }
    return false;
}

bool isDesiredUID(byte *uid, byte size, UserData &userData)
{
    for (const auto &desired : desiredUids)
    {
        if (compareUid(uid, (byte *)desired.uid, size))
        {
            userData = desired;
            return true;
        }
    }
    return false;
}

bool compareUid(byte *uid1, byte *uid2, byte size)
{
    for (int i = 0; i < size; i++)
    {
        if (uid1[i] != uid2[i])
        {
            return false; // UID berbeda
        }
    }
    return true; // UID cocok
}

void deleteLastRegisteredCard()
{
    if (lastRegisteredAddress != -1)
    {
        UserData emptyData;
          memset(&emptyData, 0xFF, sizeof(UserData)); // Mengisi data
kosong
        EEPROM.put(lastRegisteredAddress, emptyData);
```

```
      EEPROM.commit();
            bot.sendMessage(CHAT_ID, "Last registered card deleted
successfully.");
      lastRegisteredAddress = -1;
   }
   else
   {
      bot.sendMessage(CHAT_ID, "No card to delete.");
   }
}

void handleMessagesTele(int numNewMessages)
{
   Serial.print("handleNewMessages ");
   Serial.println(numNewMessages);

   for (int i = 0; i < numNewMessages; i++)
   {
      String chat_id = String(bot.messages[i].chat_id);
      String text = bot.messages[i].text;

      if (text.startsWith("/nama"))
      {
         cardName = text.substring(6);
         if (cardName.length() > 0)
         {
            bot.sendMessage(chat_id, "Name received: " + cardName);
         }
      }
      else if (text.startsWith("/tanggal_lahir"))
      {
         birthDate = text.substring(15);
         if (birthDate.length() > 0)
         {
            bot.sendMessage(chat_id, "Birth date received: " + birthDate);
         }
      }
      else if (text.startsWith("/alamat"))
```

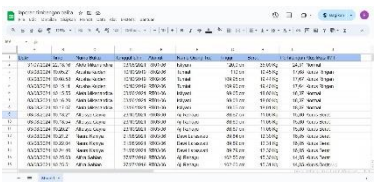
```
      {
        address = text.substring(8);
        if (address.length() > 0)
        {
          bot.sendMessage(chat_id, "Address received: " + address);
        }
      }
      else if (text.startsWith("/orang_tua"))
      {
        parentName = text.substring(11);
        if (parentName.length() > 0)
        {
              bot.sendMessage(chat_id, "Parent's name received: " +
parentName);
        }
      }
    }
}

void handleNewMessages(int numNewMessages)
{
  Serial.print("handleNewMessages ");
  Serial.println(numNewMessages);

  for (int i = 0; i < numNewMessages; i++)
  {
    String chat_id = String(bot.messages[i].chat_id);
    String text = bot.messages[i].text;

    if (text.startsWith("/daftar_kartu"))
    {
      lcd.clear();
      registerMode = true;
      bot.sendMessage(chat_id, "Scan the card to register...");
    }
    else if (text.startsWith("/hapus_id"))
    {
      deleteLastRegisteredCard();
```

```
    }
    else if (
{
  Serial.print("Connecting to ");
  Serial.println(host);

  WiFiClientSecure client;
  const int httpPort = 443;
  int retries = 5; // Jumlah percobaan koneksi

  client.setInsecure();

  while (!client.connect(host, httpPort))
  {
    Serial.print("Connection failed. Retries left: ");
    Serial.println(retries);

    if (--retries == 0)
    {
      Serial.println("Max retries exceeded.");
      return;
    }

    delay(2000); // Tunggu 2 detik sebelum mencoba lagi
  }

  Serial.println("Connected to server");

  // Encode parameter URL
  String encodedNama = urlencode(String(nama));
  String encodedTanggalLahir = urlencode(String(tanggalLahir));
  String encodedAlamat = urlencode(String(alamat));
  String encodedOrangTua = urlencode(String(orangTua));
  String encodedTinggi = urlencode(String(tinggi, 2));
  String encodedBerat = urlencode(String(berat, 2));

  String url = "/macros/s/" + GAS_ID + "/exec?nama=" + encodedNama
```

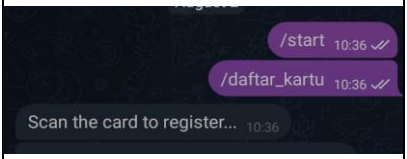
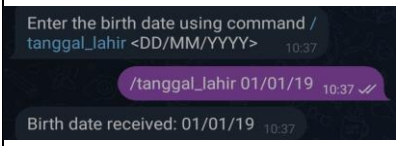
LAMPIRAN B
MANUAL BOOK

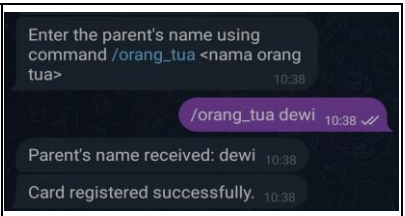
Cara Penggunaan Alat

NO	Langkah-langkah	Keterangan
1.	Pastikan alat sudah sudah mendapatkan tegangan listrik. Tekan *power on* pada box kontrol yang terpasang pada alat ukur tinggi dan berat badan otomatis.	
2.	Pastikan alat terkoneksi dengan jaringan internet agar alat bisa berfungsi dengan baik. Tampilan LCD "SELAMAT DATANG" menunjukan alat sudah siap digunakan.	
3.	Perintah selanjutnya adalah menempelkan kartu RFID pada RFID *reader*.	
4.	Jika kartu sudah terdaftar dan berhasil, maka LCD akan menampilkan keluaran identitas balita.	

5.	Tetapi jika kartu yang ditempelkan belum terdaftar, maka LCD akan menampilkan keluaran "MAAF KARTU SALAH"	
6.	Setelah menampilkan identitas balita, langkah selanjutnya adalah dengan menekan tombol *start* pada box kontrol.	
7.	Jika pengukuran tinggi dan berat sudah dilakukan, langkah selanjutnya adalah menekan tombol *save* untuk pengiriman data hasil pengukuran ke *spreadsheet.*	
8.	Tampilan data hasil pengukuran berhasil terkirim ke *spreadsheet.*	

Cara Pendaftaran Kartu untuk Balita Baru

NO	Langkah-langkah	Keterangan
1.	Untuk mendaftarkan kartu, pastikan pengguna sudah *download* aplikasi Telegram di *Smartphone*. Masukan perintah /start <enter> /daftar_kartu <enter> pada aplikasi Telegram.	
2.	Tempelkan kartu baru yang akan didaftarkan pada RFID *reader*. Maka sistem akan melakukan proses pendaftaran.	
3.	Masukan nama balita dengan perintah /nama <nama_balita>	
4.	Masukan tanggal lahir balita dengan perintah /tanggal_lahir <DD/MM/YYYY>	
5.	Masukan alamat dengan perintah /alamat <alamat>	

6.	Masukan nama orang tua dengan perintah /orang_tua <nama orang tua>	
7.	Kartu baru berhasil terdaftar	
8.	Jika ingin menghapus kartu untuk balita yang sudah tidak melakukan pengukuran, tempelkan kartu pada RFID *reader* dan masukan perintah /hapus_id pada Telegram. Kartu akan otomatis terhapus dari sistem.