Kode Pemrograman Arduino

```
#include <ESP8266WiFi.h>
#include <HX711.h>
#include <Servo.h>
#include <Firebase_ESP_Client.h>
#include <addons/TokenHelper.h>
#include <addons/RTDBHelper.h>
#include <NTPClient.h>
#include <WiFiUdp.h>
#define WIFI_SSID "annisa rahma"
#define WIFI_PASSWORD "123456789"
// Config Firebase Realtime Database
#define API_KEY "AIzaSyAsv2Ni8OhjWgaf0vBwtoU0XJfU1gnL2qs"
#define USER_EMAIL "anisarahmasari200@gmail.com"
#define USER_PASSWORD "bisalogin"
#define DATABASE_URL "https://pakan-ikan-koi-default-rtdb.asia-
southeast1.firebasedatabase.app/"
// Define Firebase objects
FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;
FirebaseJson notif, sensor;
#define SOUND_VELOCITY 0.034
// Pin Config
#define turbidity A0// input
#define trigPin D1 //ULTRASONIK
#define echoPin D2 // ULTRASONIK
#define relay1 D3
#define relay2 D4
#define servo D6 // input
#define dt D7 // input
#define sck D8 //
```

```
HX711 scale;
Servo motorServo;
String uid, sensorPath, controlPath, jadwalPath, notifPath;
boolean airKeluar = false, airMasuk = false;
unsigned int timerPakan = 0;
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "pool.ntp.org", 25200);
void setup() {
  Serial.begin(115200);
  timeClient.begin();
  // Load Cell
  scale.begin(dt, sck);
  scale.set_scale(1100.74);
  scale.tare();
  // Ultrasonic
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  // Turbidity
  pinMode(turbidity, INPUT);
  // Relay setup
  pinMode(relay1, OUTPUT);  // Pump
  pinMode(relay2, OUTPUT);  // Valve
  digitalWrite(relay1, HIGH);
  digitalWrite(relay2, HIGH);
  // Servo Setup
  motorServo.attach(servo);
  motorServo.write(0);
  // Connecting wifi
  const char *ssid = WIFI_SSID;
  const char *pass = WIFI_PASSWORD;
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED) {
    delay(200);
  }
```

```
  // Firebase
  config.api_key = API_KEY;
  auth.user.email = USER_EMAIL;
  auth.user.password = USER_PASSWORD;
  config.database_url = DATABASE_URL;
  Firebase.reconnectWiFi(true);
  fbdo.setResponseSize(4096);
  config.token_status_callback = tokenStatusCallback;
  config.max_token_generation_retry = 5;
  Firebase.begin(&config, &auth);
  Serial.print("Getting User UID");
  while ((auth.token.uid) == "") {
    delay(200);
  }
  uid = auth.token.uid.c_str();
  sensorPath = "/" + uid + "/dataSensor/";
  notifPath = "/" + uid + "/notification/";
  controlPath = "/" + uid + "/control/";
  jadwalPath = "/" + uid + "/jadwal/";
  delay(1000);
}
void loop() {
  mengirimData();
  setPakan();
  kurasAir();
  delay(1000);
}
// Mengirim data ke firebase
void mengirimData() {
  sensor.set("loadcell", loadcell());
  sensor.set("waterLevel", ultrasonic());
  sensor.set("waterQuality", waterQuality());
```

```
  String res = Firebase.RTDB.setJSON(&fbdo, sensorPath.c_str(),
&sensor) ? "Sukses" : fbdo.errorReason().c_str();
  Serial.println("Mengirim Data Sensor => " + res);
}
// Send Notification
void sendNotif(String title, String desc) {
  // memberi pakan, pakan mau habis (100gram), air keruh, ganti air
(59%)
  Serial.println("Send Notification Title => " + title);
  Serial.println("DESC : " + desc);
  timeClient.update();
  String notification = notifPath + timeClient.getEpochTime();
  notif.set("/title", title);
  notif.set("/desc", desc);
  String res = Firebase.RTDB.setJSON(&fbdo, notification.c_str(),
&notif) ? "Sukses" : fbdo.errorReason().c_str();
  Serial.println("Status Kirim Notifikasi => " + res);
}
void kurasAir() {
  String readKuras = readControl();
  int presentase = ultrasonic().toInt();
  if (readKuras.indexOf("\"kuras\":\"on\"") > 0 && airKeluar == false &&
airMasuk == false) {
    sendNotif("Sistem Kuras", "Air keruh, akan dilakukan penggantian air
aquarium.");
    airKeluar = true;
  }
  if (readKuras.indexOf("\"kuras\":\"on\"") > 0 && airKeluar == true) {
    digitalWrite(relay1, LOW);
    digitalWrite(relay2, HIGH);
    if (presentase < 25 && airMasuk == false) {
      sendNotif("Sistem Kuras", "Air keruh telah di keluarkan, pengisian
ulang air    berjalan.");
      airKeluar = false;
      airMasuk = true;
```

```
    }
  }

  if (readKuras.indexOf("\"kuras\":\"on\"") > 0 && airMasuk == true) {
    digitalWrite(relay1, HIGH);
    digitalWrite(relay2, LOW);

    if (presentase >= 100) {
      sendNotif("Sistem Kuras", "Sistem kuras air aquarium selesai.");
      String turnOff = controlPath + "kuras";
      Firebase.RTDB.setString(&fbdo, turnOff.c_str(), "off");
      digitalWrite(relay1, HIGH);
      digitalWrite(relay2, HIGH);
      airKeluar = false;
      airMasuk = false;
    }
  }
}
// Membaca Control
String readControl() {
  String resData = Firebase.RTDB.getJSON(&fbdo, controlPath) ?
fbdo.stringData() : fbdo.errorReason();
  return resData;
}
// Cek berat pakan
String loadcell() {
  scale.power_up();
  Serial.print("one reading:\t");
  Serial.println(scale.get_units(10), 1);
  int val = scale.get_units(10);
  scale.power_down();
  return (String)val;
}
```

```
// Cek ketinggian Air
String ultrasonic() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  long duration = pulseIn(echoPin, HIGH);
  float distanceCm = duration * SOUND_VELOCITY / 2;
  Serial.print("Distance : ");
  Serial.print(distanceCm);
  Serial.println("cm");
  int val = map(distanceCm, 21, 4, 0, 100);
  return (String)val;
}
// Cek kekeruhan air
String waterQuality() {
  int val = map(analogRead(turbidity), 425, 450, 0, 100);
  Serial.print("Water Quality : ");
  Serial.println(analogRead(val));
  if (val < 0) val = 0;
  if (val > 100) val = 100;
  String cekKuras = controlPath + "auto";
  String modeKuras = Firebase.RTDB.getString(&fbdo, cekKuras) ?
fbdo.stringData() : fbdo.errorReason();
  Serial.println("Mode Kuras => " + modeKuras);
  if (val < 60 && modeKuras == "on") {
    String turnOff = controlPath + "kuras";
    Firebase.RTDB.setString(&fbdo, turnOff.c_str(), "on");
  }
  return (String)val;
  }
```

```
// Memberi pakan ikan berdasarkan jadwal pada firebase
void setPakan() {
  timeClient.update();
  String jam = (timeClient.getHours() < 10) ? "0" +
(String)timeClient.getHours() : (String)timeClient.getHours();
  String menit = (timeClient.getMinutes() < 10) ? "0" +
(String)timeClient.getMinutes() : (String)timeClient.getMinutes();
  String currentTime = jam + ":" + menit;
  String jadwal = Firebase.RTDB.getString(&fbdo, jadwalPath.c_str()) ?
fbdo.stringData() : fbdo.errorReason();
  if ((jadwal.indexOf(currentTime) > 0) && (timerPakan == 0)) {
    sendNotif("Pakan Otomatis", "Pakan pada jam " + currentTime + "
sudah diberikan.");
    for (int i = 0; i < 25; i++) {
      motorServo.attach(servo);
      motorServo.write(180);
      delay(1000);
      motorServo.write(0);
      delay(1000);
    }
    timerPakan++;
  }
  if (!(jadwal.indexOf(currentTime) > 0)) timerPakan = 0;
}
```

Kode Program Main Activity Android

```
package com.ica.pakanikan
import android.app.NotificationChannel
import android.app.NotificationManager
import android.content.Context
import android.os.Build
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import androidx.core.app.NotificationCompat
import androidx.core.app.NotificationManagerCompat
import androidx.viewpager2.widget.ViewPager2
import com.google.android.material.tabs.TabLayout
import com.google.android.material.tabs.TabLayoutMediator
import com.google.firebase.auth.ktx.auth
import com.google.firebase.database.*
import com.google.firebase.ktx.Firebase
import com.ica.pakanikan.adapters.PageAdapter
import com.ica.pakanikan.fragments.HomeFragment
import com.ica.pakanikan.fragments.SettingFragment
class MainActivity : AppCompatActivity() {
    private lateinit var fbdb: DatabaseReference
    private lateinit var tabLayout: TabLayout
    private lateinit var viewPager: ViewPager2
    private lateinit var adapter: PageAdapter
    private var stateApp: Boolean = false
    private val channelId = "channel_pakan_ikan"
    private val notificaitonId = 33233
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        tabLayout = findViewById(R.id.tabLayout)
        viewPager = findViewById(R.id.viewPager2)
        val auth = Firebase.auth
```

```
        val currentUser = auth.currentUser
        fbdb = FirebaseDatabase.getInstance().getReference(currentUser!!.uid)
        tabInit()
        createNotification()
        onChangeData()
    }
    private fun tabInit() {
        adapter = PageAdapter(this)
        adapter.addFragment(HomeFragment(), "Home",
R.drawable.ic_twotone_home_24)
        adapter.addFragment(SettingFragment(), "Setting",
R.drawable.ic_twotone_settings_24)
        viewPager.adapter = adapter
        TabLayoutMediator(tabLayout, viewPager) { tab, index ->
            tab.text = adapter.setTitle(index)
            tab.setIcon(adapter.setIcon(index))
        }.attach()
    }
    private fun onChangeData() {
        val notificationPath = fbdb.child("notification")
        val query: Query = notificationPath.orderByValue().limitToLast(1)
        query.addChildEventListener(object : ChildEventListener {
            override fun onChildAdded(snapshot: DataSnapshot,
previousChildName: String?) {
                val title = snapshot.child("title").value.toString()
                val desc = snapshot.child("desc").value.toString()
                if (stateApp) setNotification(title, desc)
            }
            override fun onChildChanged(snapshot: DataSnapshot,
previousChildName: String?) { }
            override fun onChildRemoved(snapshot: DataSnapshot) { }
            override fun onChildMoved(snapshot: DataSnapshot,
previousChildName: String?) { }
```

```
override fun onCancelled(error: DatabaseError) { }
        })
    }
    private fun createNotification() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            val nameNotif = "Pakan dan Sistem Kuras"
            val desNotif = "Notifikasi jadwal pakan dan sistem kuras air
aquarium."
            val importance = NotificationManager.IMPORTANCE_DEFAULT
            val channel = NotificationChannel(channelId, nameNotif,
importance).apply {
                description = desNotif
            }
            val notifMan: NotificationManager =
getSystemService(Context.NOTIFICATION_SERVICE) as
NotificationManager
            notifMan.createNotificationChannel(channel)
        }
    }
    private fun setNotification(title: String, text: String) {
        val builder = NotificationCompat.Builder(this, channelId)
            .setSmallIcon(R.drawable.ic_launcher_foreground)
            .setContentTitle(title)
            .setContentText(text)
            .setPriority(NotificationCompat.PRIORITY_DEFAULT)
        with(NotificationManagerCompat.from(this)) {
            notify(notificaitonId, builder.build())
        }
    }
    override fun onStart() {
        super.onStart()
        stateApp = false
    }
    override fun onResume() {
        super.onResume()
```

```
    stateApp = false
}
override fun onPause() {
    super.onPause()
    stateApp = true
}
override fun onBackPressed() {
//super.onBackPressed()
stateApp = true
moveTaskToBack(true)
 }
 }
```

Kode Program Login Activity

```
package com.ica.pakanikan
import android.content.Intent
import android.os.Bundle
import android.widget.Button
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.FirebaseUser
import com.google.firebase.auth.ktx.auth
import com.google.firebase.ktx.Firebase
class LoginActivity : AppCompatActivity() {
    private lateinit var auth: FirebaseAuth
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_login)
        auth = Firebase.auth
        val currentUser = auth.currentUser
        if (currentUser != null) {
            startActivity(Intent(this, MainActivity::class.java))
            this.finishAffinity()
        }
        val email: TextView = findViewById(R.id.emailField)
        val password: TextView = findViewById(R.id.passField)
        val loginBtn: Button = findViewById(R.id.loginBtn)
        loginBtn.setOnClickListener {
            val getEmail = email.text.toString().trim()
            val getPass = password.text.toString()
            if (getEmail.isEmpty()) {
                email.error = "Email is Required!"
                email.requestFocus()
                return@setOnClickListener
            }
```

```
            if
(!android.util.Patterns.EMAIL_ADDRESS.matcher(getEmail).matches()) {
                email.error = "Email is Not Valid!"
                email.requestFocus()
                return@setOnClickListener
            }
            if (getPass.isEmpty()) {
                password.error = "Password is Required"
                password.requestFocus()
                return@setOnClickListener
            }
            auth.signInWithEmailAndPassword(getEmail,
getPass).addOnCompleteListener(this) { task ->
                if (task.isSuccessful) {
                    val user = auth.currentUser
                    updateUI(user)
                } else {
                    updateUI(null)
                }
            }
        }

    private fun updateUI(user: FirebaseUser?) {
        if (user != null) {
            startActivity(Intent(this, MainActivity::class.java))
            this.finishAffinity()
        } else {
            Toast.makeText(baseContext, "Auth Failed!",
Toast.LENGTH_LONG).show()
        }
    }
}
```

Kode Program Home Fragment

```
package com.ica.pakanikan.fragments
import android.annotation.SuppressLint
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ImageView
import android.widget.TextView
import androidx.fragment.app.Fragment
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.ktx.auth
import com.google.firebase.database.*
import com.google.firebase.ktx.Firebase
import com.ica.pakanikan.R
import java.text.SimpleDateFormat
import java.util.*
class HomeFragment : Fragment() {
    private lateinit var auth: FirebaseAuth
    private lateinit var database: DatabaseReference
    private lateinit var block1: ImageView
    private lateinit var block2: ImageView
    private lateinit var block3: ImageView
    private lateinit var block4: ImageView
    private lateinit var block5: ImageView
    private lateinit var textWeight: TextView
    private lateinit var time1: TextView
    private lateinit var time2: TextView
    private lateinit var time3: TextView
    private lateinit var time4: TextView
    private lateinit var waterQuality: ImageView
    private lateinit var waterVolume: TextView
    private lateinit var waterClear: TextView
    private lateinit var tanggalText: TextView
```

```
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        return inflater.inflate(R.layout.fragment_home, container, false)
    }
    override fun onViewCreated(view: View, savedInstanceState: Bundle?)
{
        super.onViewCreated(view, savedInstanceState)
        auth = Firebase.auth
        val currentUser = auth.currentUser
        database =
FirebaseDatabase.getInstance().getReference(currentUser!!.uid)
        textWeight = view.findViewById(R.id.textWeight)
        tanggalText = view.findViewById(R.id.tanggalText)
        block1 = view.findViewById(R.id.block1)
        block2 = view.findViewById(R.id.block2)
        block3 = view.findViewById(R.id.block3)
        block4 = view.findViewById(R.id.block4)
        block5 = view.findViewById(R.id.block5)
        time1 = view.findViewById(R.id.time1)
        time2 = view.findViewById(R.id.time2)
        time3 = view.findViewById(R.id.time3)
        time4 = view.findViewById(R.id.time4)
        waterQuality = view.findViewById(R.id.imageQuality)
        waterVolume = view.findViewById(R.id.textVolume)
        waterClear = view.findViewById(R.id.textClear)
        loadData()
        setTanggal()
    }
    @SuppressLint("SimpleDateFormat")
    private fun setTanggal() {
        val today: Date = Calendar.getInstance().time
```

```
        val formatter = SimpleDateFormat("EEEE, dd MMMM yyyy")
        val date: String = formatter.format(today)



        tanggalText.text = date
    }
    private fun loadData() {
        database.addValueEventListener(object : ValueEventListener {
            @SuppressLint("SetTextI18n")
            override fun onDataChange(snapshot: DataSnapshot) {
                val weight =
snapshot.child("dataSensor").child("loadcell").value.toString()
                textWeight.text = weight
                if (weight.toInt() > 400) block5.visibility = View.VISIBLE else
block5.visibility = View.INVISIBLE
                if (weight.toInt() > 300) block4.visibility = View.VISIBLE else
block4.visibility = View.INVISIBLE
                if (weight.toInt() > 200) block3.visibility = View.VISIBLE else
block3.visibility = View.INVISIBLE
                if (weight.toInt() > 100) block2.visibility = View.VISIBLE else
block2.visibility = View.INVISIBLE
                if (weight.toInt() > 10) block1.visibility = View.VISIBLE else
block1.visibility = View.INVISIBLE
                waterVolume.text =
snapshot.child("dataSensor").child("waterLevel").value?.toString() + "%"


                val waterStat =
snapshot.child("dataSensor").child("waterQuality").value?.toString() ?: "0"
                waterClear.text = "$waterStat%"
                if (waterStat == "null")
waterQuality.setImageResource(R.drawable.rounded)
                else if (waterStat.toInt() in 75..100)
waterQuality.setImageResource(R.drawable.ic_baseline_water_drop_goo
d)
                else if (waterStat.toInt() in 60..74)
```

```
waterQuality.setImageResource(R.drawable.ic_baseline_water_drop_normal)
            else if  (waterStat.toInt() <= 59)
waterQuality.setImageResource(R.drawable.ic_baseline_water_drop_bad)
            time1.text = snapshot.child("jadwal").child("1").value.toString()
            time2.text = snapshot.child("jadwal").child("2").value.toString()
            time3.text = snapshot.child("jadwal").child("3").value.toString()
            time4.text = snapshot.child("jadwal").child("4").value.toString()
          }
          override fun onCancelled(error: DatabaseError) {}
        })
    }
}
```

Kode Program Setting Fragment

```
package com.ica.pakanikan.fragments
import android.content.Intent
import android.os.Build
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.LinearLayout
import android.widget.TextView
import android.widget.TimePicker
import androidx.annotation.RequiresApi
import androidx.appcompat.widget.SwitchCompat
import androidx.fragment.app.Fragment
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.ktx.auth
import com.google.firebase.database.*
import com.google.firebase.ktx.Firebase
import com.ica.pakanikan.LoginActivity
import com.ica.pakanikan.R
class SettingFragment : Fragment() {
    private lateinit var auth: FirebaseAuth
    private lateinit var database: DatabaseReference
    private lateinit var timePickerButton: LinearLayout
    private lateinit var timePicker: TimePicker
    private lateinit var time1: TextView
    private lateinit var time2: TextView
    private lateinit var time3: TextView
    private lateinit var time4: TextView
```

```
        private lateinit var saveBtn: Button
        private lateinit var cancelBtn: Button



    private lateinit var kurasBtn: Button
    private lateinit var kurasMode: SwitchCompat
    private lateinit var logoutBtn: Button
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        return inflater.inflate(R.layout.fragment_setting, container, false)
    }
    @RequiresApi(Build.VERSION_CODES.M)
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        timePickerButton = view.findViewById(R.id.timePickerButton)
        timePicker = view.findViewById(R.id.timePicker)
        timePicker.setIs24HourView(true)
        time1 = view.findViewById(R.id.time1)
        time2 = view.findViewById(R.id.time2)
        time3 = view.findViewById(R.id.time3)
        time4 = view.findViewById(R.id.time4)
        saveBtn = view.findViewById(R.id.saveBtn)
        cancelBtn = view.findViewById(R.id.cancelBtn)
        logoutBtn = view.findViewById(R.id.logoutBtn)
        kurasMode = view.findViewById(R.id.kurasMode)
        kurasBtn = view.findViewById(R.id.kurasBtn)
        auth = Firebase.auth
        val currentUser = auth.currentUser
        database =
FirebaseDatabase.getInstance().getReference(currentUser!!.uid)
        kurasMode.setOnCheckedChangeListener { _, isChecked ->
```

```
    if (isChecked) {
        database.child("control").child("auto").setValue("on")
        kurasBtn.visibility = View.GONE
    } else {

            database.child("control").child("auto").setValue("off")
            kurasBtn.visibility = View.VISIBLE
        }
    }
    loadData()
    time1.setOnClickListener {
        setTime("1")
    }
    time2.setOnClickListener {
        setTime("2")
    }
    time3.setOnClickListener {
        setTime("3")
    }
    time4.setOnClickListener {
        setTime("4")
    }
    cancelBtn.setOnClickListener {
        timePicker.visibility = View.GONE
        timePickerButton.visibility = View.GONE
    }
    logoutBtn.setOnClickListener {
        Firebase.auth.signOut()
        startActivity(Intent(this.activity, LoginActivity::class.java))
        this.activity?.finishAffinity()
    }
    kurasBtn.setOnClickListener {
        database.child("control").child("kuras").setValue("on")
    }
}
```

```
    private fun loadData() {
        database.addValueEventListener(object : ValueEventListener {
            override fun onDataChange(snapshot: DataSnapshot) {

                time1.text = snapshot.child("jadwal").child("1").value.toString()
                time2.text = snapshot.child("jadwal").child("2").value.toString()
                time3.text = snapshot.child("jadwal").child("3").value.toString()
                time4.text = snapshot.child("jadwal").child("4").value.toString()
                val autoMode =
snapshot.child("control").child("auto").value.toString()
                if (autoMode == "on") {
                    kurasMode.isChecked = true
                    kurasBtn.visibility = View.GONE
                } else if (autoMode == "off") {
                    kurasMode.isChecked = false
                    kurasBtn.visibility = View.VISIBLE
                }
            }
            override fun onCancelled(error: DatabaseError) {}
        })
    }
    @RequiresApi(Build.VERSION_CODES.M)
    private fun setTime(index: String) {
        timePicker.visibility = View.VISIBLE
        timePickerButton.visibility = View.VISIBLE
        saveBtn.setOnClickListener {
            val hour = if (timePicker.hour < 10)
"0"+timePicker.hour.toString() else timePicker.hour.toString()
            val minute = if (timePicker.minute < 10)
"0"+timePicker.minute.toString() else timePicker.minute.toString()
            database.child("jadwal").child(index).setValue("$hour:$minute")
            timePicker.visibility = View.GONE
            timePickerButton.visibility = View.GONE
        }
```

```
    }
}
```