Elena Skrebneva
KA45T18S

IoT-Projektin toteutus
Projekti 4: IoT II toteutus
Älykäs valaistusjärjestelmä
Ohjaaja: Tommi Kokko

30.04.2020

Sisältö

Idea:

Projektin ideana on älykäs valaistusjärjestelmä. Älykkään sen tekee moni asia. Sen älykkyyden ydin on kyvyssään tunnistaa valoisuutta ja itse päättää, milloin valoa kannattaa sytyttää ja milloin sammuttaa. Sen lisäki, järjestelmä antaa myös ihmiselle mahdollisuuden säädellä valaistusta: Ihminen voi etäohjauksella sytyttää tai sammuttaa valon ja myös voi säädellä valon voimakkuutta kulma-anturin avulla. Järjestelmä ”muistaa” menneitä tapahtumia: se lähettääviestin ”light on!”, kun valo syttyy, ja tallentaa sekä viestin että aikaleiman tietokantaan. Kun valo syttyy, se myös ottaa valokuvan Picameralla ja sekä tallentaa sen paikallisesti tietokoneelle että vie pilvitietokantaan. Tämä on sitä varten, että voi kontrolloida, mitä tapahtui huoneessa silloin, kun valo syttyi.

Kalusto:

Raspberry Pi 3b (raspi),

ESP32 (model DevKitC_V4),

valosensori ”Grove Light Sensor”,

kulma-anturi ”Grove Rotary Angle Sensor”,

vihreä LED-valo (värillä ei ole väliä),

koekytkentälevy (5V sähköpiiri),

resistori (10kOhm) LED-valoa varten,

johtoja: HDMI-johto raspin ja television välillä, USB-johto raspin ja ESP32:n välillä, kuparijohtoja sähköpiiriä varten,

televisio,

oma tietokone: raspiin otetaan etäyhteyttä VNC-ohjelman kautta.

Ohjelmointiympäristöt, ohjelmointikielet, kirjastot ja protokollat

Sensoreiden koodi on C++ kielellä, koodattu Arduino Ide koodausympäristössä. Tarvittavat kirjastot:

- Wifi.h – Wifi yhteys,
- PubSubClient.h – mqtt aiheesta viestien julkaisu ja tilaaminen,
- WiFiClientSecure.h, MQTT.h, CloudIoTCore.h ja CloudIoTCoreMqtt.h – yhteyden

ottaminen Google Cloud pilveen etäohjausta varten

- FirebaseESP32.h – yhteyden ottaminen Firebase pilvitietokantaan,
- time.h – aikaleiman ottaminen.

PiCameran koodi on toteutettu Python kielellä Thony Python-ohjelmassa. Tarvittavat kirjastot:

- picamera (luokka PiCamera()) - kuvan ottaminen PiCameralla,
- time ja datetime – nykyhetken ottaminen,
- paho.mqtt.client – mqtt aiheesta viestien tilaaminen,
- pyrebase – valokuvien tallentaminen Firebase Storage varastoon.

Etäohjausta varten käytetään Google Cloud IoT Core palveluita: https://console.cloud.google.com/iot .

Tietokantana käytetään Google:n Firebase-analysointityökalua: https://console.firebase.google.com/ .

Tieto ”sensorit”-ohjelman ja ”kamera”-ohjelman välillä kulkee mqtt protokollan avulla, joka mahdollistaa datan julkaisun yhdessä ohjelmassa ja viestien tilaamisen toisessa ohjelmassa. Sensoreiden koodi on julkaisija ja Picameran koodi on tilaaja. Mqtt protokolla käytetään myös ”sensorit”-ohjelman ja Google Cloud IoT Core:n välillä. Tässä tapauksessa ”sensorit”-ohjelma on tilaaja ja Google Cloud on julkaisija.

Koodi

Rapsilla pyörii samaan aikaan kaksi koodia: Arduino Ide:ssa ”sensori”-niminen ohjelma ja Thony Python ohjelmassa python-kielellä ”kamera”-nimnen ohjelma. Katsotaan näiden koodien logiikka:

”sensorit”-ohjelman koodin logiikka

Tässä selityksessä jätän väliin joko täysin tai osittain Wifi:n, mqtt:n, Cloud IoT Core ja Firebase cientien toimintaa, sillä ne ovat avustavia objekteja, jotka auttavat toteutumaan valaistusjärjestelmän toiminnallisuutta. Tämä selitys koskee sensoreiden toimintaa ja mqtt:n viestittelyn logiikkaa.

Lähtotilanne:

LEDOff = true – valo on alustavasti sammuttettu

pinMode(ROTARY, INPUT) – kulma-anturilta luetaan dataa

pinMode(LIGHT_SENSOR, INPUT) – valosensorista luetaan dataa

pinMode(LED, OUTPUT) – LED-valoon dataa lähetetään

Loop funktio:

Luetaan valosensorin dataa: onko se alle 20 (mittakaavassa 0 (täysin pimeää) – 511 (täysin valoisaa))?

1. Jos on, luetaan kulma-anturin dataa (mittakaavassa 0 – 511) ja välitetään se LED-valolle – eli laitetaan valo päälle ja annetaan mahdollisuus säädellä sen voimakkuutta

- if (LEDOff == true) – Seuraavaksi katsotaan, oliko valo ennen sammutettu?
    1. Jos oli:
        1. julkaistaan mqtt aiheeseen VALO_TOPIC viesti ”light on!”;
        2. otetaan aikaleima
        3. lähetetään pilvitietokantaan objekti: {aikaleima: viesti}
        4. LEDOff = false – valo ei ole enää sammutettu
    2. (ei ollut sammutettu – ei tarvitse tehdä mitään.)
- Odotetaan 0,3sec ennen seuraavaa loopia, eli seuraavaa mittausta

2. Ei ole alle 20:

- lähetetään LED-valolle 0-arvo – sammutetaan valo
- LEDOff = true – nyt LED-valo on sammutettu

Muutama koodiin liittyvä selitys:

Numero 20 on saatu yksinkertaisesti kokeilemalla. Se tarkoitti, että minun huoneen tavallinen valoisuus ei aiheuttanut LED-valon syttymistä, mutta reagoi, kun valosensorin peitti kädellä. Se voisi olla mikä tahansa numero, joka tuntuu riittävän pimeältä.

Katsotaan tarkemmin muuttujan LEOff tarkoitusta. Koodissa halutaan tietää, milloin valo menee päälle ja milloin sammuu, ja sen perusteella lähetetään mqtt aiheeseen viestejä. Pelkkä valosensorin datan lukeminen ei riitä. Kuvitellaan tilanne, että alkoi olla pimeää, valo meni päälle ja jatkaa palamista. Joka 0,3sek tapahtuu uusi loop ja koodi mittaa valosensorin tietoa. Jos valo jatkaa palamista, se tarkoittaa , että joka 0,3 sekuntia koodi lähettää viestin ”light on!” mqtt topiciin, mikä ei ole tämän järjestelmän idea. Me halutaan tietää muutoksesta. Pelkkä valoisuuden mittaaminen ei riitä, vaan meidän pitää tietää myös LED-valon tilan muutoksesta. Tätä varten on muuttuja LEDOff.

"kamera"-ohjelman koodin logiikka

Lähtötilanne:

mqtt client on tilaajana aiheessa VALO_TOPIC.

Kun aiheesta saapuu viesti:

Tarkistetaan, onko tämä viesti ”light on!”?

1. Jos on:
    1. otetaan aikaleima
    2. otetaan kuva ja tallennetaan se uniikki nimellä, jossa on aikaleima
    3. läheteään kuva pilvivarastoon (samaan projektiin, kuin ”sensorit”-ohjelmassa).
2. (Jos ei ole: ei tarvitse tehdä mitään.)

Täydet ohjelmien koodit löytyvät tämän työn lopussa liitteestä. Tähän työhön liittyy myös video, jossa näytän, miten järjestelmä toimii.

Pohdinta

Tämän projektin tarkoitus oli olla yksinkertainen, mutta järkevä. Halusin tehdä jotakin jolla olisi potentiaalista käytännöllistä arvoa. Uskon, että tämä minulla onnistui. Projekti voisi oll monimutkaisempi ja mielenkiintoisempi. Jos minulla olisi enemmän aikaa, ei olisi muita viittä kurssia ja minä saisin tästä työstä palkkaa, voisin miettiä vielä parantamisia projektille. Esimerkiksi, tässä työssä voisi sisällyttää koodiin virran säästöä. Voisi lisätä vielä kasvojen tunnistusta, jotta kun kuva otetaan, tieto tunnistetuista kasvoista lähtee kuvan mukaan tietokantaan. Voisi myös parantaa tiedon rakennetta tietokannassa. - Nyt minulla tieto valon päälle menosta tallentuu Realtime Database tietokantaan ja valokuvat tallentuvat Storage varastoon, ja voisi olla kaikki samassa paikassa ja vielä paremmin järejstetty kansioihin

Mutta todellisuus on se, että työtä edes niin pienen projektin kanssa yksin on yllättävän paljon. Valtava osa aikaa meni erilaisten ongelmien ratkaisuun. Ongelmat aina tulivat arvaamattomasti enkä koskaan tiennyt paljonko aikaa menee niiden ratkaisuun. Jotkut ongemat veivät tunteja ja jotkut päiviä, minkä takia lopuksi päädyin projektin yksinkertaisempaan versioon.

Työ oli kuitenkin erittäin mielenkiinoinen. Tässä projektissa jouduin palauttaa mieleen ohjelmoinnin perusteita, kuten eroja eri muuttuja-tyyppien välillä (esim. char array ja String). Jouduin kovasti miettimään koodin logiikkaa alusta asti. Opin rakentamaan

sähköpiiriä, käyttämään eri ohjelmintikieliä (C++ ja python), kirjastoja, Raspberry Pi:ta, Linuxin käyttöjärjestelmää, PiCameraa, antureita, koekytkentälevyä, jne.

Olen tyytyväinen projektin tulokseen.

Lähteet

Arduino CC. Forum. Char array to String. Osoitteessa: https://forum.arduino.cc/t/char-array-to-string/531812. Viitattu: 30.04.2020.

RandomNerdsTutorials. Analog output with Arduino Ide. Osoitteessa: https://randomnerdtutorials.com/esp32-pwm-arduino-ide/. Viitattu: 30.04.2020.

Raspberry Pi. Getting started with the Camera Module. Osoitteessa: https://projects.raspberrypi.org/en/projects/getting-started-with-picamera. Viitattu: 30.04.2020.

Techtutorialsx. Python: Subscribing to MQTT topic. Osoitteessa: https://techtutorialsx.com/2017/04/23/python-subscribing-to-mqtt-topic/. Viitattu: 30.04.2020.

Techtutorialsx. Esp32: Subscribing to MQTT topic. Osoitteessa: https://techtutorialsx.com/2017/04/24/esp32-subscribing-to-mqtt-topic/. Viitattu: 30.04.2020.

Timestamp to Date Converter. Osoitteessa: https://timestamp.online/article/how-to-convert-timestamp-to-datetime-in-python. Viitattu: 30.04.2020.

Työssä käytin myös esimerkkejä kirjastoista CloudIoTCore.h ja time.h Arduino Ide:ssä.

Liitteet

”sensorit”-ohjelma

```
// Wifi network details.
const char *ssid = "XXXXXXXXX";
const char *password = "XXXXXXXXXX";

// Configuration for NTP
const char* ntp_primary = "pool.ntp.org";
const char* ntp_secondary = "time.nist.gov";

//mqtt muuttujat
const char* mqtt_server = "XXX.XXX.XXX.XXX";
const char* mqtt_user = "XXX";
const char* mqtt_password = "XXX";
const int mqtt_port = 1883;
const char* clientID = "XXXX";

// Cloud iot details.
const char *project_id = "cool-coral-303015";
const char *location = "europe-west1";
const char *registry_id = "ElenaSkrebneva";
const char *device_id = "ElenaSkrebnevaRasp";
const char *private_key_str =
    "XXXXXXXX";

// Time (seconds) to expire token += 20 minutes for drift
const int jwt_exp_secs = 60*20; // Maximum 24H (3600*24)

//Google certificate
const char *root_cert = "XXXXXXXXXX";

// Firebase details
#define HOST "https://cool-coral-303015-default-rtdb.europe-west1.firebasedatabase.app/"
#define AUTH "XXXXXXXXX"

// Tarvittavat kirjastot
#include <Client.h>
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <PubSubClient.h>
#include <MQTT.h>
#include <CloudIoTCore.h>
#include <CloudIoTCoreMqtt.h>
#include <FirebaseESP32.h>
#include "time.h"

//sensoreiden muuttujat
#define ROTARY 35
#define LED 32
#define LIGHT_SENSOR 34
int freq = 5000;
```

```
int ledChannel = 0;
int resolution = 9;
boolean LEDOff;

// Google Could IoT Core muuttujat
Client *netClient;
CloudIoTCoreDevice *device;
CloudIoTCoreMqtt *mqtt;
MQTTClient *mqttClient;
unsigned long iat = 0;
String jwt;

//Firebase muuttujat
FirebaseData firebaseData;
FirebaseJson json;

//wifi client
WiFiClient wifiClient;

//PubSub client
PubSubClient client(mqtt_server, mqtt_port, wifiClient);

// mqtt topic
#define VALO_TOPIC "valo"
//mqtt viesti
char msg[10] = "light on!";

// time muuttujat
const char* ntpServer = "pool.ntp.org";
const long  gmtOffset_sec = 2; //time zone
const int   daylightOffset_sec = 3600;

// The MQTT callback function for Core IoT
void messageReceived(String &topic, String &payload) {
  Serial.println("incoming: " + topic + " - " + payload);
  if (payload.equals("ON")) {
    Serial.println("LED on!");
    if (LEDOff == true) {
      client.publish(VALO_TOPIC, msg);
      LEDOff = false;
    }
    unsigned long starttime = millis();
    unsigned long endtime = starttime;
    while ((endtime - starttime) < 14000) {
      int brightness = analogRead(ROTARY);
      ledcWrite(ledChannel, brightness);
      String info = "valo paalle";
      String path = "/" + getTimestring();
      Serial.println(path);
      json.set(path, info);
      Firebase.updateNode(firebaseData, "/data", json);
      endtime = millis();
    }
  }
}
```

```
String getJwt(){
  iat = time(nullptr);
  Serial.println("Refreshing JWT");
  jwt = device->createJWT(iat, jwt_exp_secs);
  return jwt;
}

// setup WiFi
void setupWifi() {
  Serial.println("Starting Wifi");
  WiFi.mode(WIFI_STA);
  Serial.print("Conencting to wifi");

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
    }
  configTime(0, 0, ntp_primary, ntp_secondary);
  Serial.println("Waiting on time sync...");
  while (time(nullptr) < 1510644967){
    delay(10);
  }
}

// tarkista WiFi
void checkWifi(){
  Serial.print("checking wifi...");
  while (WiFi.status() != WL_CONNECTED){
    Serial.print(".");
    delay(1000);
  }
  Serial.print("WiFi connected to ");
  Serial.println(WiFi.localIP());
}

// ota yhteys PubSub mqtt clientillä
void mqttconnect() {
  while (!client.connected()) {
    Serial.print("MQTT connecting...");

    if (client.connect("clientID", mqtt_user, mqtt_password)) {
      Serial.println("connected");
      Serial.println("subscribed");
      }
     else {
      Serial.print("failed to connect");
      Serial.println("try again in 5 sec");
      delay(5000);
      }
    }
  }
```

```
// setup CloudIoTCore
void setupCloudIoT(){
  device = new CloudIoTCoreDevice(
    project_id, location, registry_id, device_id,
    private_key_str);
  setupWifi();
  netClient = new WiFiClientSecure();
  mqttClient = new MQTTClient(512);
  mqttClient->setOptions(180, true, 1000);
  mqtt = new CloudIoTCoreMqtt(mqttClient, netClient, device);
  mqtt->setUseLts(true);
  mqtt->startMQTT();
}

// setup Firebase
void setupFirebase() {
   Firebase.begin(HOST, AUTH);
   Firebase.reconnectWiFi(true);
   Firebase.setReadTimeout(firebaseData, 1000*60);
   Firebase.setwriteSizeLimit(firebaseData, "tiny");
   Serial.println("Connected to FIrebase");
}

//saa aikaleimasta String
String getTimestring() {
  struct tm timeinfo;
  if(!getLocalTime(&timeinfo)){
    Serial.println("Failed to obtain time");
    return "notime";
  }
  char buff[30];
  strftime(buff,30, "%Y_%B_%d_%H:%M:%S", &timeinfo);
  byte at = 0;
  String str1 = "";
  char *p = buff;
  while (*p++) {
    str1.concat(buff[at++]);
  }
  return str1;
}

//lähtötilanne
void setup() {
  //Serial Monitor päälle
  Serial.begin(115200);
  //sensoreiden modit
  pinMode(ROTARY, INPUT);
  pinMode(LIGHT_SENSOR, INPUT);
  pinMode(LED, OUTPUT);

 // LED-valo analogiseen modeen
  analogSetWidth(9);
  ledcSetup(ledChannel, freq, resolution);
  ledcAttachPin(LED, ledChannel);
  ledcWrite(ledChannel, 0);
```

```
  //muita Setup funktioita
  setupCloudIoT();
  setupFirebase();
  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
  client.setServer(mqtt_server, mqtt_port);
  // valo on pois päältä
  LEDOff = true;
  //PubSub Client pyörimään
  client.loop();
  }

void loop() {
  //tarkista, että PubSub Client on yhdistetty
  if (!client.connected()) {
    checkWifi();
    mqttconnect();
    }
  //ClouIoTCore mqtt client pyörimään
  mqtt->loop();
  delay(10);
  //tarkista CloudIoTCore mqtt client on yhdistetty
  if (!mqttClient->connected()) {
    checkWifi();
    mqtt->mqttConnect();
    }
  //mittaa valoisuus
  int lightness = analogRead(LIGHT_SENSOR);
  //jos liian pimeää
  if (lightness < 20) {
    //luetaan kulma-anturin dataa
    int brightness = analogRead(ROTARY);
    //ja lähetetään se analogisen kanavan kautta LED-valolle
    ledcWrite(ledChannel, brightness);
   //oliko valo sammutettu ennen?
    if (LEDOff == true) {
      //jos oli, tehdään seuraavat jutut:
      client.publish(VALO_TOPIC, msg);
      String info = "valo paalle";
      String path = "/" + getTimestring();
      Serial.println(path);
      json.set(path, info);
      Firebase.updateNode(firebaseData, "/data", json);
      LEDOff = false;
      }
    delay(300);
    }
  // jos ei ole liian pimeää
  else {
    //sammutetaan valo
    ledcWrite(ledChannel, 0);
    LEDOff = true;
  }
}
```

”kamera”-ohjelma

```
from picamera import PiCamera
import time
import datetime
import paho.mqtt.client as mqttClient
from time import sleep
import pyrebase


firebaseConfig = {
    'apiKey': "XXXXXX",
    'authDomain': "cool-coral-303015.firebaseapp.com",
    'databaseURL': "https://cool-coral-303015-default-rtdb.europe-west1.firebasedatabase.app",
    'projectId': "XXXXX",
    'storageBucket': "XXXXX",
    'messagingSenderId': "XXXXXXX",
    'appId': "XXXXXXXX",
    'measurementId': "XXXXXXXX"
  }
firebase = pyrebase.initialize_app(firebaseConfig)
storage = firebase.storage()
camera = PiCamera()
camera.framerate = 15

// callback funktio: tilaa viestejä mqtt aiheesta
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        client.subscribe("valo")
        print("Connected to broker")
    else:
        print("Connection failed")

// callback funktio: ota kuva, tallenna se ja lähetä pilvivarastoon, kun aiheesta saapuu oikea viesti.
def on_message(client, userdata, msg):
    print(msg.topic + ": " + str(msg.payload))
    if str(msg.payload) == "b'light on!'":
        camera.rotation = 180
        camera.start_preview()
        sleep(5)
        // minne kuva tallennetaan tällä tietokoneella
        path = '/home/pi/projekti4/test'
        //ota aikaleima
        time_now = datetime.datetime.now().strftime("%d-%m-%Y_%H:%M:%S")
        // polku kuvaan tällä tietokoneella
        pic = path + time_now + '.jpg'
        // kuvan nimi pilvivarastoa varten
        pic_name = time_now + '.jpg'
        camera.annotate_text = (time_now)
        //ota kuva
        camera.capture(pic)
        camera.stop_preview()
```

```
        //lähetä kuva pilvivarastoon
        storage.child(pic_name).put(pic)

broker = "XXX.XXX.XXX.XXX"
port = 1883
user = 'XXXXX'
pwd = 'XXXXX'

client = mqttClient.Client("XXXXXX")
client.username_pw_set(user, password=pwd)
//mitä tehdään kun mqtt client saa yhteyden
client.on_connect = on_connect
// mitä tehdään kun saapuu viesti
client.on_message = on_message
//yhdistä mqtt palvelimeen (brokeriin)
client.connect(broker,port=port)
//mqtt client pyörimään
client.loop_forever()
```