+

**Etäohjattava robottiauto**

Kuva, joka sisältää kohteen elektroniikka, Tietokoneen osa, Sähkötekniikka, Tietokonelaitteisto

Kuvaus luotu automaattisesti

Näyttötyö: Sulautettujen järjestelmien toteutus ja käyttöönotto

Joona Lappalainen K3TVT22C

Savon ammattiopisto

7.3.2024

Sisällys

[Johdanto 4](#_Toc158364605)

[Tarvikeluettelo 4](#_Toc158364606)

[Ohjelmointityökalujen käyttöönotto 5](#_Toc158364607)

[Kytkentä ja johdotuskaavio 6](#_Toc158364608)

[Kuva valmiista testikytkennästä 7](#_Toc158364609)

[Ohjelmointi 7](#_Toc158364610)

[Koodi 8](#_Toc158364611)

[Johtopäätökset ja itsearviointi 9](#_Toc158364612)

Johdanto

Tässä työssä rakennettiin, ohjelmoitiin ja otettiin käyttöön etäohjattu robottiauto. Etäohjaus toteutettiin Blynkin ja joystickin avulla (jotenkuten).

Työ kasattiin valmiille alustalle ja käytettiin kahta DC-moottoria. Moottoreissa on vaihteistot valmiina ja renkaat kävivät suoraan akseleille.

…

…

Tarvikeluettelo

|  |  |
| --- | --- |
|  | kpl |
| Alustasarja | 1 |
| 9 V Paristo | 2 |
| Adafruit Huzzah ESP32 | 1 |
| Adafruit DC motor shield | 1 |

Ohjelmointityökalujen käyttöönotto **(Tämä on malli - Tee oma oikealle versiolle!)**

Asenna Arduino IDE v. 1.8.10

* https://www.arduino.cc/en/software

Asenna CP2014 USB ajuri

* https://learn.adafruit.com/adafruit-huzzah32-esp32-feather/using-with-arduino-ide

Asenna Adafruit Huzzah with ESP32 kirjastot Arduino Ideen

* Lisää tämä rivi Arduino Iden asetuksiin: https://dl.espressif.com/dl/package\_esp32\_index.json
* (https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package\_esp32\_index.json)

Asenna Boards Managerissa esp32 korttipaketti Arduino Ideen

Valitse Arduino Idessä kortti Adafruit ESP32 Feather

Asenna puhelimeesi Blynk-IoT. Sen jälkeen avaa tietokoneesi selaimella Blynk.cloud ja rekisteröi itsellesi sinne tunnukset. Seuraa sivuston tutoriaalin ohjeita ja luo itsellesi siellä uusi ESP32-laite ohjeiden mukaisesti. Muista syöttää oikeat SSID ja salasana jotta saat ao. kooditemplaten itsellesi oikeilla AuthTokeneilla ja TemplateID:llä käyttöösi.

Kytkentä ja johdotuskaavio

Johdotuskaavio tehty Fritzing ohjelmalla. Lisäkomponentteja joutui lataamaan valmistajan sivuilta ja tuomaamaan ne Fritzingiin. Esimerkiksi ESP32 –kuvaa ei löydy suoraan Fritzingistä.

Kytkentäkaavio on tehty multisim-ohjelmistolla

Laitteen johdotuskaavio (Joy Stick ohjaus ei toiminut)

Kuva, joka sisältää kohteen kuvakaappaus, teksti, diagrammi, muotoilu

Kuvaus luotu automaattisesti

Laitteen kytkentäkaavio (Tämä kuva on tehty MultiSimilla - Nyt tehdään Fritzingilla)

Kuva, joka sisältää kohteen diagrammi, Tekninen piirros, Suunnitelma, piirros

Kuvaus luotu automaattisesti

Kuva valmiista testikytkennästä

Kuva, joka sisältää kohteen elektroniikka, Sähkötekniikka, Tietokoneen osa, Tietokonelaitteisto

Kuvaus luotu automaattisesti

Ohjelmointi

Käytetyt kirjastot

Kuva, joka sisältää kohteen teksti, kuvakaappaus, numero, Fontti

Kuvaus luotu automaattisesti

Koodi(Kopioitavassa muodossa!)

/\* ---Robo auto blynkillä--- \*/

#define BLYNK\_TEMPLATE\_ID ""

#define BLYNK\_TEMPLATE\_NAME "ESP32"

#define BLYNK\_AUTH\_TOKEN ""

#include <WiFi.h>

#include <WebServer.h>

#include <Adafruit\_MotorShield.h>

#include <Wire.h>

#include <BlynkSimpleEsp32.h>

#include <NewPing.h>

// Moottorin liikesuunnat

#define FORWARD 1

#define BACKWARD 2

#define LEFT 3

#define RIGHT 4

#define STOP 5

// WiFi-yhteyden tiedot

const char\* ssid = "";

const char\* password = "";

// Blynk-tunnukset

char auth[] = "";

char server[] = "blynk-cloud.com";

#define BLYNK\_PRINT Serial

// Moottorisuojan määrittely

Adafruit\_MotorShield AFMS = Adafruit\_MotorShield();

Adafruit\_DCMotor\* myMotor1;

Adafruit\_DCMotor\* myMotor2;

// Moottorin nopeus

int speed = 255;

// Ultraäänisensorien pinnien määrittely

#define FRONT\_TRIGGER\_PIN 32

#define FRONT\_ECHO\_PIN 14

#define BACK\_TRIGGER\_PIN 33

#define BACK\_ECHO\_PIN 15

#define MAX\_DISTANCE 200

// Ultraäänisensorit

NewPing frontSonar(FRONT\_TRIGGER\_PIN, FRONT\_ECHO\_PIN, MAX\_DISTANCE);

NewPing backSonar(BACK\_TRIGGER\_PIN, BACK\_ECHO\_PIN, MAX\_DISTANCE);

unsigned long previousMillis = 0;

const long interval = 30000;

const long ultrasonicInterval = 100;

unsigned long previousUltrasonicMillis = 0;

void setup() {

Serial.begin(9600);

Blynk.begin(auth, ssid, password);

Serial.println("Setting up Motor Shield");

AFMS.begin();

myMotor1 = AFMS.getMotor(3);

myMotor2 = AFMS.getMotor(4);

myMotor1->setSpeed(speed);

myMotor2->setSpeed(speed);

}

void loop() {

Blynk.run();

unsigned long currentMillis = millis();

if (currentMillis - previousMillis >= interval) {

previousMillis = currentMillis;

}

if (currentMillis - previousUltrasonicMillis >= ultrasonicInterval) {

previousUltrasonicMillis = currentMillis;

unsigned int frontDistance = frontSonar.ping\_cm();

unsigned int backDistance = backSonar.ping\_cm();

if (frontDistance > 0 && frontDistance < 10) {

moveBackwardShort();

} else if (backDistance > 0 && backDistance < 10) {

moveForwardShort();

}

}

}

// Blynk-sovelluksesta saatu nopeus

BLYNK\_WRITE(V0) {

speed = param.asInt();

myMotor1->setSpeed(speed);

myMotor2->setSpeed(speed);

}

// Blynk-sovelluksesta saatu komento liikkua eteenpäin

BLYNK\_WRITE(V1) {

int pinValue = param.asInt();

if (pinValue == 1) {

Serial.println("Moving forward");

myMotor1->run(FORWARD);

myMotor2->run(FORWARD);

} else {

stopMotors();

}

}

// Blynk-sovelluksesta saatu komento liikkua taaksepäin

BLYNK\_WRITE(V2) {

int pinValue = param.asInt();

if (pinValue == 1) {

Serial.println("Moving backward");

myMotor1->run(BACKWARD);

myMotor2->run(BACKWARD);

} else {

stopMotors();

}

}

// Blynk-sovelluksesta saatu komento kääntyä vasemmalle

BLYNK\_WRITE(V3) {

int pinValue = param.asInt();

if (pinValue == 1) {

Serial.println("Turning left");

myMotor1->run(BACKWARD);

myMotor2->run(FORWARD);

} else {

stopMotors();

}

}

// Blynk-sovelluksesta saatu komento kääntyä oikealle

BLYNK\_WRITE(V4) {

int pinValue = param.asInt();

if (pinValue == 1) {

Serial.println("Turning right");

myMotor1->run(FORWARD);

myMotor2->run(BACKWARD);

} else {

stopMotors();

}

}

// Moottorien pysäyttäminen

void stopMotors() {

Serial.println("Stopping");

myMotor1->run(RELEASE);

myMotor2->run(RELEASE);

}

// Lyhyt liike taaksepäin esteen havaitsemisen jälkeen edessä

void moveBackwardShort() {

Serial.println("Obstacle detected in front! Moving backward briefly.");

myMotor1->run(BACKWARD);

myMotor2->run(BACKWARD);

delay(200);

stopMotors();

}

// Lyhyt liike eteenpäin esteen havaitsemisen jälkeen takana

void moveForwardShort() {

Serial.println("Obstacle detected in the back! Moving forward briefly.");

myMotor1->run(FORWARD);

myMotor2->run(FORWARD);

delay(200);

stopMotors();

}

**Rikkinäinen auto joystickillä!!!**

/\* ---Robo auto--- \*/

#define BLYNK\_TEMPLATE\_ID ""

#define BLYNK\_TEMPLATE\_NAME "ESP32"

#define BLYNK\_AUTH\_TOKEN ""

#include <WiFi.h>

#include <Adafruit\_MotorShield.h>

#include <Wire.h>

#include <BlynkSimpleEsp32.h>

#include <NewPing.h>

#include <esp\_now.h>

// Rakenne tietojen vastaanottamiseen

typedef struct struct\_message {

int xAxis;

int yAxis;

bool buttonPressed;

} struct\_message;

// Muuttuja saapuville tiedoille

struct\_message incomingData;

#define FORWARD 1

#define BACKWARD 2

#define LEFT 3

#define RIGHT 4

#define STOP 5

// WiFi-tiedot

const char\* ssid = "";

const char\* password = "";

// Blynk-tunnukset

char auth[] = "";

char server[] = "blynk-cloud.com";

#define BLYNK\_PRINT Serial

// Moottorisuoja ja moottorien määrittely

Adafruit\_MotorShield AFMS = Adafruit\_MotorShield();

Adafruit\_DCMotor\* myMotor1;

Adafruit\_DCMotor\* myMotor2;

int speed = 255;

// Ultraäänisensorien pinnit ja maksimietäisyys

#define FRONT\_TRIGGER\_PIN 32

#define FRONT\_ECHO\_PIN 14

#define BACK\_TRIGGER\_PIN 33

#define BACK\_ECHO\_PIN 15

#define MAX\_DISTANCE 200

NewPing frontSonar(FRONT\_TRIGGER\_PIN, FRONT\_ECHO\_PIN, MAX\_DISTANCE);

NewPing backSonar(BACK\_TRIGGER\_PIN, BACK\_ECHO\_PIN, MAX\_DISTANCE);

unsigned long previousUltrasonicMillis = 0;

const long ultrasonicInterval = 100;

// Funktio datan vastaanottamiseksi ESP-NOW:n kautta

void OnDataRecv(const esp\_now\_recv\_info \*recv\_info, const uint8\_t \*data, int len) {

memcpy(&incomingData, data, sizeof(incomingData));

Serial.print("Bytes received: ");

Serial.println(len);

Serial.print("xAxis: ");

Serial.println(incomingData.xAxis);

Serial.print("yAxis: ");

Serial.println(incomingData.yAxis);

Serial.print("Button pressed: ");

Serial.println(incomingData.buttonPressed);

}

void setup() {

Serial.begin(115200);

delay(1000);

// Blynk-yhteyden aloitus

Blynk.begin(auth, ssid, password);

// Moottorisuoja-asetukset

Serial.println("Setting up Motor Shield");

AFMS.begin();

myMotor1 = AFMS.getMotor(3);

myMotor2 = AFMS.getMotor(4);

myMotor1->setSpeed(speed);

myMotor2->setSpeed(speed);

// Moottorien pysäytys aluksi

stopMotors();

// WiFi-asetukset

WiFi.mode(WIFI\_STA);

// ESP-NOW:n alustus

if (esp\_now\_init() != ESP\_OK) {

Serial.println("Error initializing ESP-NOW");

return;

}

// Datan vastaanotto callback-funktion rekisteröinti

esp\_now\_register\_recv\_cb(OnDataRecv);

}

void loop() {

unsigned long currentMillis = millis();

// Ultraäänisensorin etäisyyden mittaus tietyin väliajoin

if (currentMillis - previousUltrasonicMillis >= ultrasonicInterval) {

previousUltrasonicMillis = currentMillis;

unsigned int frontDistance = frontSonar.ping\_cm();

unsigned int backDistance = backSonar.ping\_cm();

if (frontDistance > 0 && frontDistance < 10) {

moveBackwardShort();

} else if (backDistance > 0 && backDistance < 10) {

moveForwardShort();

}

}

// Käsitellään joystickin tiedot

processJoystickData();

}

// Blynk-widgetien käsittelyt

BLYNK\_WRITE(V0) {

speed = param.asInt();

myMotor1->setSpeed(speed);

myMotor2->setSpeed(speed);

}

BLYNK\_WRITE(V1) {

int pinValue = param.asInt();

if (pinValue == 1) {

Serial.println("Moving forward");

myMotor1->run(FORWARD);

myMotor2->run(FORWARD);

} else {

stopMotors();

}

}

BLYNK\_WRITE(V2) {

int pinValue = param.asInt();

if (pinValue == 1) {

Serial.println("Moving backward");

myMotor1->run(BACKWARD);

myMotor2->run(BACKWARD);

} else {

stopMotors();

}

}

BLYNK\_WRITE(V3) {

int pinValue = param.asInt();

if (pinValue == 1) {

Serial.println("Turning left");

myMotor1->run(BACKWARD);

myMotor2->run(FORWARD);

} else {

stopMotors();

}

}

BLYNK\_WRITE(V4) {

int pinValue = param.asInt();

if (pinValue == 1) {

Serial.println("Turning right");

myMotor1->run(FORWARD);

myMotor2->run(BACKWARD);

} else {

stopMotors();

}

}

// Moottorien pysäytys

void stopMotors() {

Serial.println("Stopping");

myMotor1->run(RELEASE);

myMotor2->run(RELEASE);

}

// Lyhyt liike taaksepäin esteen havaittaessa edessä

void moveBackwardShort() {

Serial.println("Obstacle detected in front! Moving backward briefly.");

myMotor1->run(BACKWARD);

myMotor2->run(BACKWARD);

delay(200);

stopMotors();

}

// Lyhyt liike eteenpäin esteen havaittaessa takana

void moveForwardShort() {

Serial.println("Obstacle detected in the back! Moving forward briefly.");

myMotor1->run(FORWARD);

myMotor2->run(FORWARD);

delay(200);

stopMotors();

}

// Joystick-tietojen käsittely

void processJoystickData() {

int x = incomingData.xAxis;

int y = incomingData.yAxis;

bool buttonPressed = incomingData.buttonPressed;

Serial.print("Joystick x: ");

Serial.print(x);

Serial.print(" y: ");

Serial.print(y);

Serial.print(" buttonPressed: ");

Serial.println(buttonPressed);

// Jos nappi painettu, pysäytä moottorit

if (buttonPressed) {

stopMotors();

return;

}

// Liikkuminen joystickin asentojen mukaan

if (x < 85) {

Serial.println("Turning left");

myMotor1->run(BACKWARD);

myMotor2->run(FORWARD);

} else if (x > 170) {

Serial.println("Turning right");

myMotor1->run(FORWARD);

myMotor2->run(BACKWARD);

} else if (y < 85) {

Serial.println("Moving backward");

myMotor1->run(BACKWARD);

myMotor2->run(BACKWARD);

} else if (y > 170) {

Serial.println("Moving forward");

myMotor1->run(FORWARD);

myMotor2->run(FORWARD);

} else {

stopMotors();

}

}  
  
/\* ---Ohjain--- \*/

#include <esp\_now.h>

#include <WiFi.h>

// Vastaanottimen MAC-osoite

uint8\_t broadcastAddress[] = {0x7C, 0x9E, 0xBD, 0xD8, 0x2A, 0x38};

// Rakenne viestin lähettämistä varten

typedef struct struct\_message {

int xAxis;

int yAxis;

bool buttonPressed;

} struct\_message;

struct\_message myData; // Muuttuja viestin tiedoille

esp\_now\_peer\_info\_t peerInfo; // ESP-NOW:n peer-tiedot

// Joystickin ja painikkeen pinnit

const int joyXPin = 33;

const int joyYPin = 32;

const int buttonPin = 14;

// Funktio joystickin arvojen muuntamiseen ja deadbandin käsittelyyn

int mapAndAdjustJoystickDeadBandValues(int value, bool reverse)

{

if (value >= 1910)

{

value = map(value, 1910, 4095, 127, 255); // Kartoitus yläpuolelle

}

else if (value <= 1906)

{

value = map(value, 1906, 0, 127, 0); // Kartoitus alapuolelle

}

else

{

value = 127; // Deadband-alue

}

if (reverse)

{

value = 254 - value; // Arvon kääntäminen, jos reverse on true

}

return value;

}

// Funktio datan lähettämisen tilan tarkistamiseksi

void OnDataSent(const uint8\_t \*mac\_addr, esp\_now\_send\_status\_t status) {

Serial.print("\r\nLast Packet Send Status:\t");

Serial.println(status == ESP\_NOW\_SEND\_SUCCESS ? "Delivery Success" : "Delivery Fail");

}

void setup() {

Serial.begin(115200); // Sarjaliikenteen aloitus

WiFi.mode(WIFI\_STA); // WiFi-tilan asetus

// ESP-NOW:n alustus

if (esp\_now\_init() != ESP\_OK) {

Serial.println("Error initializing ESP-NOW");

return;

}

esp\_now\_register\_send\_cb(OnDataSent); // Lähetyksen callback-funktion rekisteröinti

// Peer-tietojen asettaminen

memcpy(peerInfo.peer\_addr, broadcastAddress, 6);

peerInfo.channel = 0;

peerInfo.encrypt = false;

if (esp\_now\_add\_peer(&peerInfo) != ESP\_OK){

Serial.println("Failed to add peer");

return;

}

pinMode(buttonPin, INPUT\_PULLUP); // Painikkeen pinni inputiksi ja pullup-tilaan

}

void loop() {

// Joystickin ja painikkeen tilan lukeminen

myData.xAxis = mapAndAdjustJoystickDeadBandValues(analogRead(joyXPin), false);

myData.yAxis = mapAndAdjustJoystickDeadBandValues(analogRead(joyYPin), false);

myData.buttonPressed = digitalRead(buttonPin) == LOW;

// Datan lähetys ESP-NOW:n kautta

esp\_err\_t result = esp\_now\_send(broadcastAddress, (uint8\_t \*) &myData, sizeof(myData));

if (result == ESP\_OK) {

Serial.println("Sent with success");

}

else {

Serial.println("Error sending the data");

}

delay(500); // Viive 500 ms ennen seuraavaa lähetysyritystä

}

Johtopäätökset ja itsearviointi

Laitteen kasaus nyt ei ollut mitenkään kovin haastavaa mutta johtoja oli kyllä todella monta. Blynkin kanssa ei ollut haasteita sillä se oli todella simppeliä mutta kun yritti joystickiä lisätä esp-now avulla niin en saanut mitenkään autoa toimimaan. Koitin erilaisia koodeja katsoin eri lähteistä neuvoja ja käytin pelkän joystickin lisäämiseen viikon aikaa. Ainoa tapa mitenkä sain auton toimimaan joystickin kanssa on että siinä on viivettä ja blynkki ei sillon toimi. Autossa sai joystickin tai blynkin toimimaan mutta kun molempia koitti saada samaan aikaan niin se tuntui lähes mahottomalle.

Projekti kokonaisuudessaan oli opettava kokemus opin paljon lisää juttuja mikro kontrollereista kuten esp32 ja opin myös pikkusen lukemaan koodia paremmin koska kamppailin yli viikon sen joystickin kanssa enkä siltikkään saanut sitä toimimaan.  
  
  
**(uskon että joystickin saa tehtyä kun nään jonkun tekevän toimivan semmoisen)**