Tommi Kainulainen

Capillary Networksin toimintaa esittävä prototyyppi

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Automaatiotekniikka

Insinöörityö

30.10.2015

Tekijä Otsikko Sivumäärä Aika	Tommi Kainulainen Capillary Networksin toimintaa esittävä Prototyyppi 22 sivua + 2 liitettä 30.10.2015
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Automaatiotekniikka
Suuntautumisvaihtoehto	Kappaletavara-automaatio
Ohjaaja(t)	Lehtori Raisa Vartia Guest Researcher Nicklas Beijar

Insinöörityön tavoitteena on tutustua Ericsson Capillary Networksin esittelyssä ja havainnollistamisessa käytettävään demoon sekä kehittää sen ohjaus ja kommunikaatio kykyjä.

Alkuperäinen järjestelmä koostui kahden LEGO- paketin sisällöstä muodostaen kauko-ohjattavan junan ja junaradan. Juna kiskoi perässään vaunuja, joiden sisällä oli mikrokontrollereita, joilla oltiin yhteydessä tietokoneeseen.

Tarkoituksena oli suunnitella ja asentaa junaan moottoriohjaus ja kyky kommunikoida tietokoneelle, josta sitä voidaan komentaa.

Lopputuloksena oli prototyyppi, jossa junan sisällä oleva mikrokontrolleri pystyy langattoman verkon kautta olemaan yhteydessä tietokoneeseen, josta pystytään hallitsemaan junan uuden tasavirtamoottorin pyörimisnopeutta ja suuntaa reaaliajassa.

Avainsanat	Capillary Networks, prototyyppi, moottorinohjaus, kommunikointi

Author Title Number of Pages Date	Tommi Kainulainen Prototype demonstrating Capillary Networks 22 pages + 2 appendices 30.10.2015
Degree	Bachelor of Engineering
Degree Programme	Automation engineering
Specialisation option	Manufacturing automation
Instructor(s)	Lecturer Raisa Vartia Guest researcher Nicklas Beijar

The purpose of this bachelors thesis is to explore the prototype that is used to demonstrate and visualize the Ericsson Capillary Networks and to develop its control and communication capabilities.

The original system consisted of the contents of two different LEGO- package, forming remote -controlled motorized train and a train track. The train was pulling two carts that each had a microcontroller inside, which were used to communicate with a computer.

The goal of the project was to design and install a motor control for the train and to develop an ability to communicate with a computer, from where it could be controlled.

The end product was a prototype where a microcontroller located inside the train is used to connect to a computer trough wireless network where you can control the speed and direction of the new DC motor in real time.

Keywords	Capillary Networks, prototype, motorcontrol, communication

Sisällys

Lyhenteet

TCP/IP Transmission Control Protocol / Internet Protocol. Kaksi tärkeintä internet protokollaa, jotka varmistavat, että tietoa siirrettäessä se saapuu oikeaan paikkaan ilman virheitä.

REST Representational State Transfer. Ohjelmointiarkkitehtuuri, jonka tarkoituksena on parantaa tietokoneiden kommunikoinnin tehokkuutta.

PWM Pulse Width Modulation. Tapa lähettää viestejä pulssisignaaleina sähköisille laitteille.

GPIO General Purpose Input/Output. Mikrokontrollereista löytyvä pinni, joka voidaan ohjelmoida signaalin joko outputiksi tai inputiksi

HTTP Hypertext Transfer Protocol. Internetkommunikaatioprotokolla, jota käytetään tiedonsiirtoon.

IDE Integrated Development Environment. Ohjelma jonka tarkoitus on helpottaa uusien ohjelmien luomista, esimerkiksi lähdekoodin kirjoitus tekstieditorilla.

IoT Internet of Things. Esineiden internet, jolla tarkoitetaan laitteita, jotka pystyvät olemaan yhteydessä muihin laitteisiin, jotka vuorostaan reagoivat vastaanotettavaan tietoon älykkäästi ja samalla lähettäen tietoa eteenpäin.

1 Johdanto

Insinöörityö tehdään Ericssonin tutkimus- ja tuotekehitysyksikkö NOMADICLABille, ja tavoitteena on perehtyä vanhaan Capillary Netwoksin toimintaa esittämään luotuun demoon. Työssä suunnitellaan sekä luodaan uusi prototyyppi, jossa tulee olla mikrokontrolleri, joka kykenee olemaan etäyhteydessä tietokoneeseen. Tietokoneesta sille lähetetään komentoja junan liikkeen suhteen, ja sen tulee pystyä välittämään nämä komennot moottorille, joka liikuttaa sitä.

Alkuperäinen prototyyppi koostuu kahdesta eri LEGO-paketin yhdistelmästä, muodostaen moottoroidun kauko-ohjattavan junan, kaksi vedettävää vaunua sekä junaradan.

Molemmissa perässä tulevissa vaunuissa on joko sinisellä tai punaisella merkitty kontti, mikä mahdollistaa konttien havainnoinnin ja erottelun radan päällä sijaitsevalla kameralla. Molempien konttien sisällä on mikrokontrolleri, jolla pystytään muodostamaan yhteys tietokoneeseen.

Prototyyppiä varten tulee myös tehdä tai löytää sopiva rajapinta, jolta käsin mikrokontrolleriin saadaan yhteys tietokoneelta ja josta voidaan ohjata junan liikkumista radalla. Lisäksi pitää rakentaa moottorinohjausjärjestelmä, jolla moottori saadaan kiinni edellämainittuun mikrokontrolleriin sekä kirjoittaa koodi, jolla kaikki tämä saadaan yhdistettyä toimivaksi kokonaisuudeksi.

2 Oy L M Ericsson Ab

Ericsson on maailman johtava telekommunikaatiolaitteiden valmistaja ja niihin liittyvien palveluiden tarjoaja 35 %:n (2012) markkinaosuudellaan, joka kehittää mm. laajakaista-, multimedia- ja radioteknologiaa sekä langatontaverkkoa ja mobiiliverkkoa. Ericsson tarjoaa alan johtavaa verkkolaitteiden ja -ohjelmistojen sekä palveluiden verkko- ja liiketoimintaa. Se tekee tuotteita kaapeli-, mobiilialusta - ja tehomoduulin markkinoille.

Yhtiö aloitti toimintansa Ruotsissa vuonna 1876 ja laajensi toimintaansa Suomeen vuonna 1918 työllistäen nykyään 950 henkilöä Suomessa ja 117 655 maailmanlaajuisesti (2014).

Suomen yksikkö on keskittynyt mobiiliviestintään ja laitteiden väliseen kommunikaatioon, ja etenkin IoT tulee olemaan tulevaisuudessa tärkeä.

Capillary Networks koostuu sekä fyysisesti verkossa kiinni olevista sekä lyhyen kantaman radioteknologiaa käyttävistä laitteista, jotka mahdollistavat niiden keskenäisen paikallisen yhteyden luomisen lähettyvillä oleviin laitteisiin, jotka voidaan silloin yhdistää matkapuhelinverkkoon Capillary Gatewayllä. [6.]

3 Suunnittelu

Uuden prototyypin suunnittelu lähti käyntiin pohtimalla, miten pystytään hoitamaan moottorinohjaus sekä kommunikaatio käyttäjän kanssa. Kokemuksesta tiedettiin jo ennestään, että Arduino–mikrokontrollerit toimivat hyvin moottorinohjauksessa ja että niillä on mahdollista kommunikoida hyvin monipuolisesti lukuisten eri laitteiden kanssa. Lisäksi alustalle löytyy paljon valmista koodia, ohjelmisto -kirjastoja sekä projektiohjeita, joten Arduinon käytön tutkimisella oli hyvä aloittaa. [5.]

Projektin ensimmäinen vaihe oli tilata Arduino Starter kit, ja sen mukana tulleet Arduino Uno, 9V tasavirtamoottori, koekytkentälauta, hyppylangat sekä L293DNE- moottorinohjauspiiri. Nämä mahdollistivat aikaisemmin opittujen asioiden muistiin palauttamisen sekä moottoriohjauksen toteuttamisen suunnittelun, koska lopullisessa prototyypissä tulee olemaan samankaltaiset välineet mukana.

Ensimmäinen realistinen suunnitelma oli käyttää Arduino Unoa junan moottorin hallinnassa ja olla sarjakytkettynä kommunikointiin suunnitellun STM32W-BRD mikrokontrollerin kanssa rx- ja tx- pinnien kautta. [12.]

Mikrokontrollerit

Alun perin kommunikaatioon tietokoneen kanssa suunniteltiin käytettäväksi ST-Microelectroniksin STM32W-BRD-mikrokontrolleria, joita oli käytetty jo alkuperäistä esittämistä varten, mutta pian ilmeni ongelmia komponentin integroimisessa Arduinon kanssa. Arduino ja STM-sarjan tuotteet vaativat hyppylankoja toistensa tx- ja rx-pinnien välille, jotta ne saataisiin keskusteluyhteyteen keskenään. STM32W:ssä ei kuitenkaan ole suoraan näitä pinnejä vaan ne pitäisi itse ohjelmoida GPIO-pinneihin. [8.]

Lopulta mikrokontrollerista päätettiin luopua, kun kävi selväksi, että uusien kappaleiden ja varaosien saamisesta tulee hankalaa tai mahdotonta, koska kyseisen komponentin tuotanto on lopetettu.

Korvaava vaihtoehto STM32-BRD:n tilalle oli TMOTE-Sky, joka vastaa monella tavalla edeltäjäänsä toiminnaltaan. Tästä luovuttiin kuitenkin hyvin pian, koska Tmoten integroiminen muuhun laitteistoon koodipuolella olisi ollut hyvin vaikeaa ja työlästä. [11.]

Seuraava ja lopulliseksi jäänyt kokeiltu mikrokontrolleri oli Libeliumin Waspmote PRO, joita löytyi varastosta ja minkä testaamista helpottivat mukana tulleet yhteensopiva 3,7 V:n ja 6600mAh akku sekä Roving Networksin RN171XV wifi -Module, jolla kommunikaatio ylöspäin voitaisiin mahdollisesti hoitaa.

Waspmoten ohjelmoiminen tehdään Waspmote PRO IDE -ohjelmalla, mikä osoittautui hieman kaksijakoiseksi toiminnaltaan. Ohjelma näyttää ja toimii lähes täysin samalla tavalla kuin Arduinon ohjelmoinnissa käytetty Arduino IDE, joka on helppokäyttöinen ja tuttu. Toisaalta Arduinoa ohjelmoidaan joko C- tai C++ -kielellä, mutta Waspmotea kielellä nimeltä Wiring, mikä on sekoitus molempia edellä mainittuja. Ne eivät tue toistensa kirjastoja, joten laitteiden integroiminen koodipuolella osoittautui haastavaksi. Waspmotelle löytyi kuitenkin vanha toimiva koodi TCP-yhteyden luomiseen tietokoneelle, mikä helpotti päätöstä siirtyä sen käyttöön. [1; 2; 3; 9.]

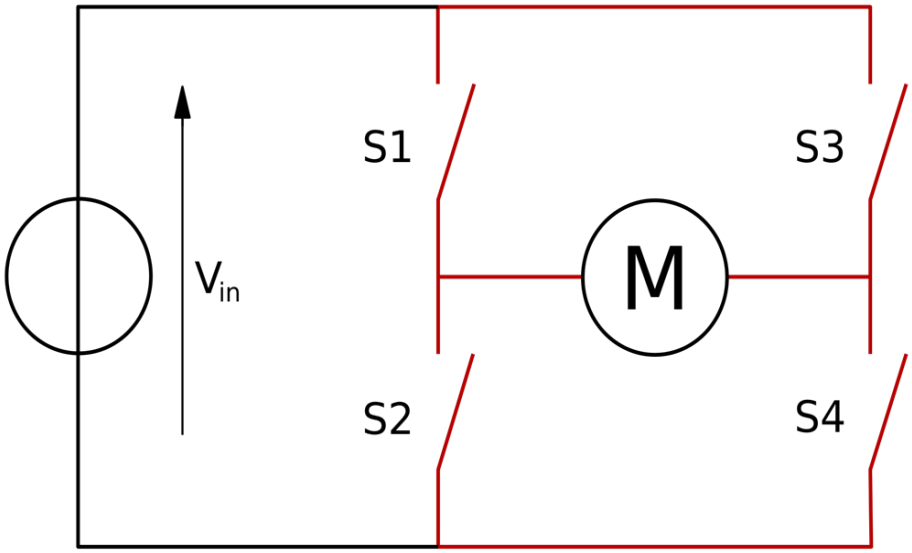
4 Toteutus

Kun saatiin päätettyä miten kommunikointi suoritetaan, aloitettiin moottorinohjauksen rakentaminen, ja toimivan koodin kirjoittaminen samanaikaisesti. Ensimmäinen tavoite oli saada molemmat sellaiseen vaiheeseen, että moottoriin saadaan virta, ja sen jälkeen aletaan pikkuhiljaa lisäämään ominaisuuksia kuten nopeuden ja pyörimissuunnan muuttaminen. Lopulta kasassa olisi kokonaisuus, jolla prototyyppi saa suoritettua vaaditut tehtävänsä.

4.1 Hardware

4.1.1 L293DNE-moottorinohjain

L293DNE-moottorinohjain on Texas Instrumentsin valmistama H-silta, joka sallii jännitteen kaksisuuntaisen kulun mahdollistaen tasavirtamoottorin pyörimisen sekä eteen- että taaksepäin mikropiirissä olevien neljän kytkimen asentoja vaihtamalla ohjelmoinnin mukaan. [10; 17.]



Kuva 1 H-sillan rakenne

Taulukko 1 Kytkinten asennon vaikutus ohjattavaan moottoriin

S1	S2	S3	S4	Vaikutus moottoriin
1	0	0	1	Pyörii oikealle
0	1	1	0	Pyörii vasemmalle
0	0	0	0	Ei vaikutusta
0	1	0	1	Ei vaikutusta
1	0	1	0	Ei vaikutusta
1	1	0	0	Ei vaikutusta
0	0	1	1	Ei vaikutusta
1	1	1	1	Ei vaikutusta

4.1.2 Waspmote

Waspmote on Libeliumin valmistama matalavirtainen mikrokontrolleri, joka käyttää aktiivisena 15 mA, ja joka on ensisijaisesti suunniteltu erilaisten anturien käyttöön ja kykenee viestimään viidellätoista eri tekniikalla jakautuen lyhyen, keskipitkän ja pitkän matkan kommunikointiteknologiohin. [2,3.]
Waspmote suorittaa prototyypissä kriittistä roolia moottorinohjauksessa vastaanottaen käskyjä tietokoneelta RN-XV:n kautta.

4.1.3 RN-XV

RN-XV on Waspmoteen suunniteltu ja helposti integroitava Wi-Fi-laite, joka vaatii yhteyden virta-, RX-, TX- sekä maa-pinniin muodostaakseen langattoman yhteyden. Se käyttää virtaa 38 mA aktiivisena ja 4uA unitilassa. Sisältää 802.11 b/g radion, 32 bitin prosessorin, TCP/IP -stackin, reaaliaikaisen kellon, joiden avulla yhteyden luominen ja ajastimen käyttö onnistuvat hyvin**.** [4.]

4.1.4 Juna

Koska junan kyytiin oli tulossa alkuperäistä enemmän kalustoa, todettiin, että junaa pitää laajentaa, jotta kaikki saadaan mahtumaan sen sisään. Suurimmat sovitusongelmat syntyivät RN171XV:n antennin ja Waspmoten akun kanssa. Ensimmäinen oli liian pitkä ja esti katon paikalleen panemisen, toinen liian leveä ja pitkä, jotta sen saisi muiden laitteiden kanssa sisälle 26 cm pitkään, 4,8 cm leveään ja 7,1 cm korkeaan junaan. Leveyden laajennus tapahtui kiinnittämällä junan pohjan päälle Lego -levyjä, jotka tekivät keskiosasta 13,5 cm pitkän ja 8 cm leveän, minkä lisäksi sen molemmilla puolilla on 3 cm pitkä 6,4 cm:n levennys. Junaa ei lopulta tarvinnut korottaa antennin takia, vaan seinille laitettiin ikkunat, joista yhdestä antenni tulee ulos junan sivulle.

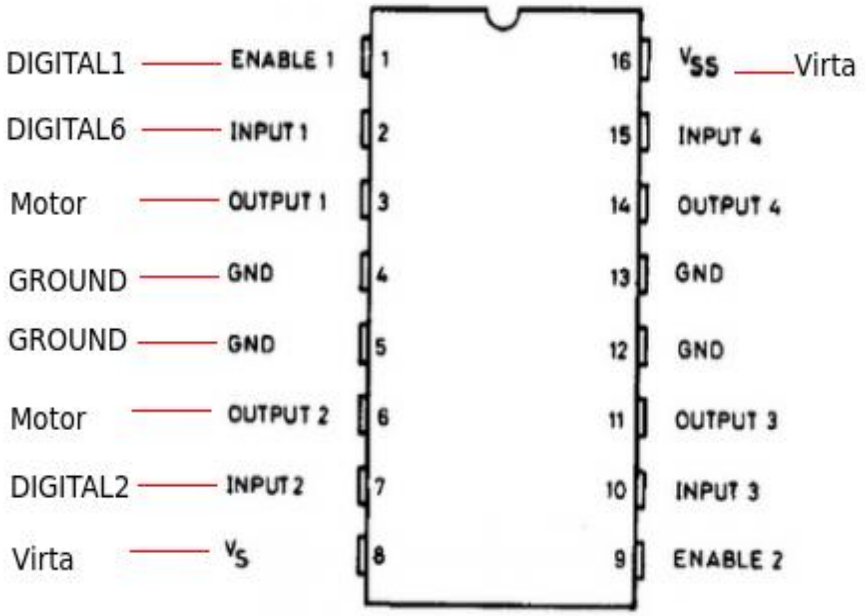
4.1.5 Koekytkentälevy ja hyppylangat

Koekytkentälevy on kaksipuolinen levy, jossa toisella puolella on kuparia ja toisella sähköä johtamatonta materiaalia. Levy on täynnä reikiä, jotka on eristetty toisistaan riveittäin, jolloin voidaan säädellä, mitkä levyyn kytketyt komponentit saavat sähköä ja mistä. [15.]
Virtaa näihin riveihin johdetaan metallisilla hyppylangoilla, joita liitetään koekytkentälevyyn eri komponenttien välille, mikä mahdollistaa virran kulun paristolta moottorinohjaimen kautta moottoriin.

4.2 Moottorinohjaus

16-pinninen L293DNE-moottorinohjain kolvattiin kiinni 3 cm * 2,4 cm kokoiseksi leikatun pitkittäin kuparoidun koekytkentälevyn keskelle, ja sen jalkojen 1-8 sekä 16 kytköksissä oleville riveille jokaiselle kolvattiin oma hyppylanka, joilla pystytään sallimaan virrankulku moottorinohjausta hallitsevalle Waspmotelle. [7.]

Yhteys Waspmoteen, moottoriin sekä ulkoiseen virtalähteeseen luodaan kuvan mukaisella tavalla.



Kuva 2 Kytkennät moottorinohjaimen jaloista eri lähteisiin

DIGITAL1 toimii Waspmotessa PWM-pinninä, jolla määrätään moottorin pyörimisnopeutta säätelemällä läpikulkevaa virtaa. Moottori pyörii maksiminopeudella (HIGH) ,kun läpi päästetään 5 V ja kun läpi ei päästetä ollenkaan virtaa, moottori pysähtyy (LOW). Arvo voi myös olla jotain tuolta väliltä.

Input-pinnit 1 ja 2 kytketään Waspmoten digitaalipinneihin 6 ja 2, joilla määritetään moottorin pyörimissuuntaa. Kun DIGITAL2-pinnin tulo on HIGH ja DIGITAL6:n LOW, pyörii moottori myötäpäivään, ja jos DIGITAL2 on LOW ja DIGITAL6 HIGH, moottori käy vastapäivään. Mikäli molemmat pinnit ovat samaan aikaan joko LOW tai HIGH, niin moottori pysähtyy.

OUTPUT-pinnit 1 ja 2 on kiinnitetty moottoriin, ja niitä pitkin kulkeva virta määrittelee moottorin toiminnan aikaisemmin mainittujen pinnien saamien käskyjen mukaan. GND-pinnit on kytketty yhteiseen maahan Waspmoten kanssa, jolloin estetään mahdollisia vikatiloja kuten sähköiskujen aiheutumisia laitteille tai käyttäjille.

Virtapinnit on kytketty ulkoiseen virtalähteeseen, josta moottori saa virtansa. Tässä tapauksessa virtalähteenä toimii 9 V:n paristo.

4.3 Ohjelmointi

Mikrokontrollereiden ohjelmoimisessa käytettiin valmistajien tekemiä ohjelmia, Waspmotea ohjelmoitiin Waspmote PRO IDE -ohjelmalla ja ohjelmointikielenä toimii Wiring, mikä on yhdistelmä C - ja C++ -kieliä. Arduino UNOn ohjelmointi toteutettiin Arduino IDE:llä, ja ohjelma tunnistaa sekä C- että C++ -ohjelmointikielet. Tässä tapauksessa käytettiin C:tä.

Tuotettu koodi koostuu kolmesta osasta, jossa ensin ohjelmaan sisälletään tarvittavat kirjastot, joista mikrokontrolleri löytää ohjelmassa tarvittavia komentoja sekä määritellään käytettävän langattoman verkon nimi ja salasana. Itse koodin runko koostuu kahdesta osasta, setup()-osa ajetaan yhden kerran, kun ohjelma aloitetaan. ja siinä esitetään koodissa käytettävät tulot, lähdöt ja muuttujat. Loop() osaa ajaa nimensä mukaan läpi koko ajan uudestaan, ja tähän osaan sijoitetaan koodin toiminnallinen osa kuten moottorin ohjaus. Ohjelmat tallennetaan.pde-muodossa.

Koska Arduino on yksi käytetyimmistä mikrokontrollereista etenkin moottoriohjauksessa, on siihen tehty valmiita ohjelmia paljon. Internetistä löytyikin valmis koodi, jolla pystyi säätämään moottorin nopeutta ja pyörimissuuntaa. Tämä ei kuitenkaan ollut suoraan kopioitavissa Waspmotelle, koska koodissa käytetyt kirjastot eivät olleet yhteensopivia ohjelmien kesken. Se johtaa siihen että Waspmote ei ymmärtänyt komentoja. Koodin muokkaamisen lisäksi se piti integroida wi-fi-koodiin, jolla mikrokontrollerilla otetaan yhteys sitä komentavaan tietokoneeseen ja sen langattomaan verkkoon. [5.]

Waspmote Pro oli aluksi hankalakäyttöinen, ja suurimman ongelman tuotti virran saaminen sen eri alueille, joilla laite on tarkoitus yhdistää moottorinohjauspiiriin. Virta ei kulkenut, vaikka Waspmoten maa- ja 5 V-virtapinniin johti virtaa suoraan hyppylangoilla. Vaati tarkempaa tutustumista manuaaleihin. Ennen kävi ilmi, että virran kulku pitää sallia koodissa. [1.]

4.3.1 Koodi

Alustus

Alustus on ensimmäinen osa koodia, ja siinä siihen sisällytetään WIFI–kirjasto. Kirjasto sisältää selityksiä käskyille, jonka avulla koodissa voidaan käyttää uusia toimintoja niin, että Waspmote ymmärtää ne. Näin pystytään myöhemmin muodostamaan WIFI -yhteys.

Seuraavaksi valitaan ja aktivoidaan alue socket0 Waspmotesta. Tämä on osa mikro-kontrolleria, johon kaikki moottorinohjaukseen liittyvät hyppylangat on kiinnitetty ja josta moottorin ohjaukseen liittyvät käskyt kulkevat. Alueen aktivointi sallii virran kulkemisen alueelle.

Viimeiseksi määritetään käytettävän langattoman verkon nimi ja salasana, johon RN-XV WiFi-laitteen halutaan ottavan yhteyttä, kun koodia aletaan suorittaa, sekä TCP-portti kyseiselle laitteelle. Näillä tiedoilla pystytään luomaan etäyhteys tietokoneen ja mikrokontrollerin välille.

```
// Include WIFI library
 #include <WaspWIFI.h>
// choose socket (SELECT USER'S SOCKET)
 uint8_t socket=SOCKET0;
// WiFi AP settings (CHANGE TO USER'S AP)
#define ESSID "NOMADICLAB"
#define AUTHKEY "xxxxxxxxxxxxxxx"
// TCP server settings
#define LOCAL_PORT 2000 //TCP port that your sensor is listening
at
char direction[10];
char speed[10];
char timer[100];
char url[513];
int speed_int;
int delaylenght;
```

Koodin osan 1 rivit, jotka alkavat //-merkein, ovat kommentteja, joilla on tarkoitus selventää koodia.

Setup

Tässä osassa on määritelty asetukset ja edellytykset WiFi:n toiminnalle, kuinka muodostaa yhteys, tallentaa ja unohtaa tietoja sekä ilmoittaa, kun yhteys on joko luotu tai sitä ei ole pystytty muodostamaan.
Tämän lisäksi lopussa on määritelty Digitaali-pinnit 1,2 ja 6 Outputeiksi, jolloin niiden kautta pystytään johtamaan virtaa eteenpäin. Nämä pinnit ovat hyppylangoilla kiinni moottorinohjauksessa, ja niiden läpi kulkevan virran määrä vaikuttaa moottorinohjaukseen.

```
void setup() {
   //USB.ON();
   delay(100);
  if( WIFI.ON(socket) == 1 )
  {
    USB.println(F("WiFi switched ON"));
        // reset to avoid previous configuration stored in the
module
    //WIFI.resetValues();
    // 1. Configure the transport protocol (UDP, TCP, FTP,
HTTP...)
    WIFI.setConnectionOptions(HTTP|CLIENT_SERVER);
    // 2. Configure the way the modules will resolve the IP
address.
    WIFI.setDHCPoptions(DHCP_ON);
    // 3. Configure how to connect the AP
    WIFI.setJoinMode(MANUAL);
    // 4. Set the AP authentication key
    WIFI.setAuthKey(WPA1,AUTHKEY);
    // 5. Save Data to module's memory
    WIFI.storeData();
        // If it is manual, call join giving the name of the AP
    if( WIFI.join(ESSID) )
    {
      USB.println(F("Joined"));
          USB.println(F("-----------------------"));
      USB.println(F("get IP"));
```

```
      USB.println(F("------------------------"));
      WIFI.getIP();
      USB.println();
         WIFI.cleanRemoteMessage();
// Call the function to create a TCP connection on port 2000
      if (WIFI.setTCPserver(LOCAL_PORT))
      {
        USB.println(F("TCP server set"));
      }
      else
      {
        USB.println(F("TCP server NOT set"));
      }
    }
    else
    {
      USB.println(F("NOT joined to AP"));
    }
  }
  else
  {
    USB.println(F("WiFi did not initialize correctly"));
  }
      //Set up the variables
  // Sets the 5V switch ON
  PWR.setSensorPower(SENS_5V, SENS_ON);
  delay(100);
  pinMode(DIGITAL6, OUTPUT);
  pinMode(DIGITAL2, OUTPUT);
  pinMode(DIGITAL1, OUTPUT); //ENABLEPIN -1
  analogWrite(DIGITAL1, LOW);
}
```

Setup osa koodista.

Loop

Loop on viimeinen osa koodia, jota luetaan koko ohjelman käynnissäoloajan. Se mahdollistaa arvojen muutoksen ohjauksessa. Kohdassa määritetään moottorin ohjaamista varten tarvittavat arvot, joita voidaan syöttää REST Easy -ohjelmaan, kun etäyhteys on muodostettu. Moottorin pyörimissuunta voi saada arvot 0,1 tai 2 ja tätä ohjataan säätelemällä virran kulkua aikaisemmin määritellyistä OUTPUT:eista. Pyörimisnopeudeksi voidaan laittaa mikä tahansa arvo väliltä 0-255, missä 255 on maksimivauhti ja 0 on pysähtynyt. Moottoriin voidaan pistää myös ajastin, että se kulkee vain halutun ajan mittaisen ajan verran.

```
void loop() {
   // Reads from the TCP connection
   WIFI.read(NOBLO);
      if(WIFI.length>0)
   {
      strcpy(url, (char*)WIFI.answer);
            char chars_array[strlen(strtok(url, "\r\n"))+1];
            strcpy(chars_array, strtok(url, "\r\n"));
            if (chars_array!=NULL){
                     ///Parsing the information GET /X/Y/Z  (X)
= 0,1,2 DIRECTION - (Y)= 0-255 SPEED - (Z) DELAY          if
(strstr(chars_array, "GET")!=NULL){
              USB.println(chars_array);
              strtok(chars_array, " ");
              char tmp[20];
              strcpy(tmp, strtok(NULL, " "));
              //char sentence[100];
              //sprintf(sentence,"tmp %s\0", tmp);
              //USB.println(sentence);
              strcpy(direction, strtok(tmp, "/"));
              char sentence[100];
              sprintf(sentence,"direction %s\0", direction);
              USB.println(sentence);
              strcpy(speed, strtok(NULL, "/"));
              sprintf(sentence,"speed %s\0", speed);
              USB.println(sentence);
```

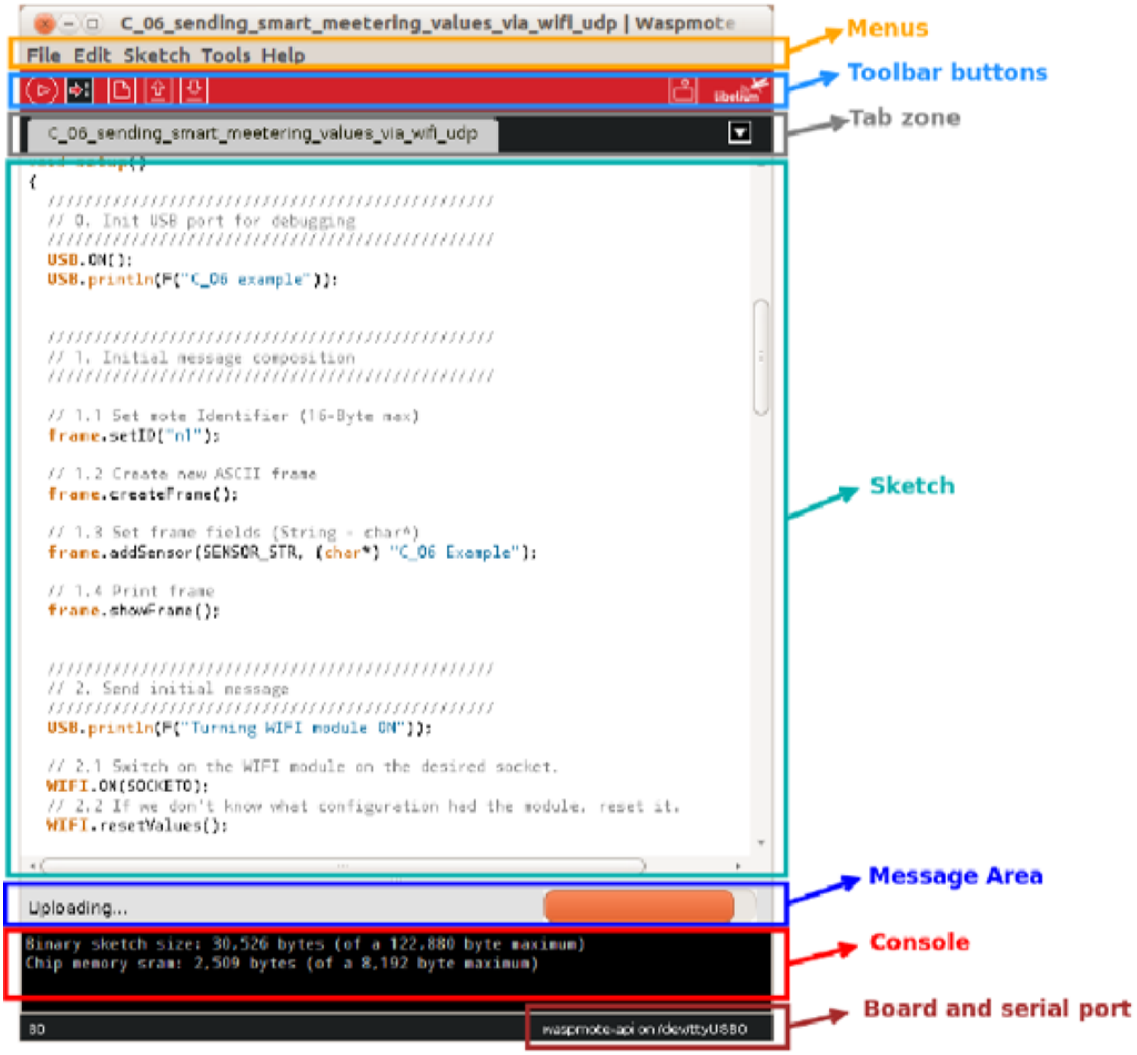
```
            speed_int = atoi( speed );
            analogWrite(DIGITAL1, speed_int);
            strcpy(timer, strtok(NULL, "/"));
            sprintf(sentence,"timer %s\0", timer);
            USB.println(sentence);
            delaylenght = atoi(timer);
          //Stop the engine
          if (strcmp(direction,"0") == 0){
            digitalWrite(DIGITAL6, LOW);
            digitalWrite(DIGITAL2, LOW);
          }
         //forward
          else if(strcmp(direction,"1") == 0) {
            digitalWrite(DIGITAL6, HIGH);
            digitalWrite(DIGITAL2, LOW);
            delay (delaylenght*1000);

            if (delaylenght!=0){
              digitalWrite(DIGITAL6, LOW);
              digitalWrite(DIGITAL2, LOW);
            }
          }
          //back
          else if(strcmp(direction,"2") == 0) {
            digitalWrite(DIGITAL6, LOW);
            digitalWrite(DIGITAL2, HIGH);
            delay (delaylenght*1000);
             if (delaylenght!=0){
              digitalWrite(DIGITAL6, LOW);
              digitalWrite(DIGITAL2, LOW);
           }
          }
       }
      ////////////////////////////////////////////
      // Send the HTTP get/post query (specifying the WEB server
so DNS is used)
      ////////////////////////////////////////////
```

```
        WIFI.flush();
        WIFI.send("HTTP\/1.1 200 OK\r\nContent-Type:
text\/plain\r\nContent-Length: 0\r\n\r\n\r\n");
    }
}
```

Loop osa koodista

4.3.2 Waspmote Pro IDE



Kuva 3 Waspmote Pro IDEn käyttöjärjestelmä

Waspmote PRO:n ohjelmoinnissa käytettävä ohjelma, jossa koodi ja muistiinpanot kirjoitetaan kohtaan Sketch, ja mahdollistaa virheiden ja tunnistamattomien komentojen tarkistuksen. Se tapahtuu painamalla Compile-kohtaa. Huomautukset virheistä ilmestyvät Console-kohtaan, josta pystyy lukemaan virheen luonteen, ja rivin, missä kohtaa se sijaitsee koodia.
Board and serial port -kohdasta näkee, onko ohjelmalle asetettu mitään kohdetta, mihin koodin voi ladata.

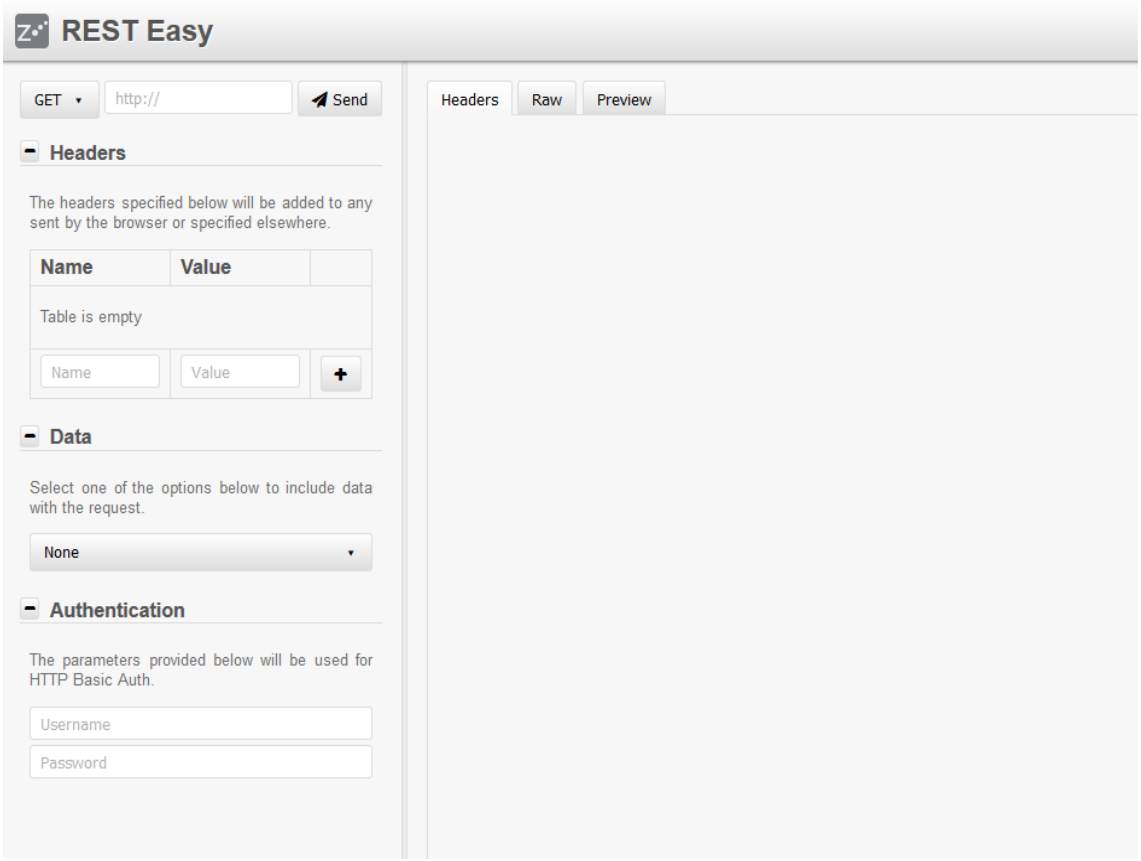
4.2.2 REST Easy Firefox

REST Easy -ohjelma on kehitetty laitteiden kommunikoinnin tehostamista varten. Siinä käytetään REST-arkkitehtuuria, jossa tiedon välittäminen tapahtuu viittauksilla kyseiseen tietoon täydellisen kopion lähettämisen sijaan. Esimerkkinä arkkitehtuurin toiminnasta ovat monet internetissä toimivat yritykset kuten Netflix, jossa katsojalle ei saa kopiota alkuperäisestä videosta vaan se lähetetään suoratoistona.
Prototyypissä sovellusta käytetään yhteyden muodostamiseen mikrokontrollerin ja tietokoneen välille, josta käskyjen antaminen tehdään. [16.]

5 Prototyypin käyttäminen, testaus ja mittaukset

5.1 Prototyypin käyttö

Waspmote kiinnitetään USB-johdolla tietokoneeseen, josta ladataan koodi Waspmote PRO IDE:n kautta mikrokontrolleriin. Tämän jälkeen ajetaan ladattu koodi ja käynnistetään sarjamonitori, joka luo TCP-yhteyden koneen ja mikrokontrollerin välille. Varmistetaan, että akku ja paristo ovat kiinni.
Seuraava vaihe on REST easy -ohjelman käynnistys Firefox -verkkoselaimessa. Valitaan ylhäällä olevasta palkista GET ja syötetään viereiselle riville mikrokontrollerin IP-osoite, gateway sekä halutut arvot junan liikkumiselle kuvan mukaisesti. Lopuksi lähetetään komento SEND -nappulasta.



Kuva 4 Rest Easyn pääkymä

Kuva 5 Palkki johon kirjoitetaan IP-osoite, TCP-portti sekä halutut arvot liikkeen suhteen

Liikkuminen toimii periaatteella X/Y/Z, jossa X on kulkusuunta. Se määritellään arvoilla 0-2: seis, eteenpäin ja taaksepäin numerojärjestyksen mukaan. Y on kulkunopeus ja vaihtelee väliltä 0-255 teoriassa. Käytännössä junalla on niin paljon massaa, että arvon tulee olla vähintään 100, jotta juna lähtee liikkeelle ollessaan radalla. Z on ajastin, millä juna voidaan laittaa kulkemaan halutun ajan valittuun suuntaan ja halutulla vauhdilla, minkä jälkeen se pysähtyy. Kohtaan laitettu arvo vastaa samaa määrää sekunteina. Pysähtymisen voi toteuttaa laittamalla arvon nolla joko kohtaan X tai Y.

5.2 Virrankulutus

Mittaus toteutettiin ajamalla junalla 15 minuuttia. Samalla mitattiin moottorin sekä Waspmoten virtalähteiden jännite ennen ja jälkeen testin.

Junan moottorin virtalähde on 9 V paristo, jossa oli testin alussa 8,78 V. Junaa ajettiin 15 minuuttia vaihdellen suuntaa ja nopeutta. Kokeen jälkeen virtaa oli jäljellä 7,87 V. 0,91 voltin kulutus 15 minuutissa tekee 3,64 V tunnissa, mikä tarkoittaa, että täydellä 9 V paristolla pystyy teoriassa ajamaan reilun tunnin ennen kuin lataus tippuu alle 5 V:n jolloin moottori ei enää saa tarpeeksi vääntöä junan liikuttamiseen. Virrankulutus riippuu tietenkin paljolti siitä, kuinka nopeasti junalla kuljetaan, suunnanvaihteluiden tiheydestä sekä pysähdysten ja uudestaan liikkeelle lähtemisten määrästä.

Waspmotessa oleva 3,7 V:n ladattava akku, jossa oli yleismittarin ja patterin latauksen mittauskoodin mukaan ~4,1355V ja oli 92 % täynnä, menetti noin 20 minuutin ajamisen jälkeen lataustaan ~ 0,0065V, ollen ~4,1290v ja 91 % täynnä. Akun tulisi täydellä latauksella kestää pidempiäkin demosessioita. Myöhemmin todettiin, että virranmittauskoodin mukaan akku on 1 % täynnä, kun akussa on 3,309 V virtaa, eikä se pystynyt liikuttamaan edes pelkkää moottoria ilman junan painoa mukana.

5.3 Reagointiaika

Ensimmäisessä versiossa koodia uuden junan liikkumista koskevan komennon lähettämisen ja mikrokontrollerin siihen vastaanottamiseen ja reagoimiseen meni noin neljä sekuntia. Tätä saatiin kuitenkin parannettua WiFi-kirjastoa optimoimalla, jolla saatiin poistettua viive lähes olemattomaksi. Nyt eniten aikaa kuluu haluttujen arvojen syöttämiseen REST Easy -rajapintaan.

6 Parannusehdotuksia

Juna kuluttaa turhan nopeasti 9 V paristonsa, ja moottori menettää tehoaan, joten isompi paristo tai akku moottorin virtalähteeksi pidempiä esityksiä varten olisi tarpeellinen parannus. Myös jonkinlaisen koodin luominen virtalähteen tilan seuraamiseksi olisi tutkimisen arvoista, koska tällä hetkellä pariston varauksen mittaaminen pitää tehdä yleismittarilla, mikä on työlästä, koska juna pitää purkaa osittain pariston irrottamista varten.

Vaihtoehtoisesti junan ja sen sisällä olevien komponenttien pienentäminen toisi lisää elinaikaa virtalähteelle, kun liikutettavan massan määrää vähennetään.

REST easy on toimiva, joskin hieman kömpelö ja hidas rajapinta junan ohjaamista varten, ja sen korvaaminen toisella ohjelmalla tai uuden kehittäminen olisi hyvä idea tulevaisuudessa.

6 Yhteenveto

Opinnäytetyön tarkoituksena oli perehtyä LMF Ericssonin tutkimus ja tuotekehitys -yksikön NOMADICLABin Capillary Networksin toiminnan esittämiseksi käytettävään demoon. Pyrittiin kehittämään tilalle uusi korvaava prototyyppi, jolle tulee luoda moottorinohjauspiiri, joka yhdistetään mikrokontrolleriin, mille pitää luoda kyky olla langattomassa yhteydessä tietokoneeseen, josta sille voidaan antaa komentoja prototyypin liikkumista koskien.

Suunnittelutyö tehtiin osittain ryhmässä, jossa päätettiin prototyypin halutut ominaisuudet, minkä jälkeen tavoitteisiin pääsemiseen annettiin melko vapaat kädet. Ohjelmointi ja moottoriohjauksen teko suoritettiin itsenäisesti etsimällä ja soveltamalla verkosta löytyviä projekteja, tai kysymällä apua muilta ryhmän jäseniltä.

Hankaluuksia tuotti erityisesti sopivan mikrokontrollerin löytäminen, jonka pystyisi integroimaan valmiiksi olemassa olevaan kalustoon ja mikä pystyisi toteuttamaan kaiken prototyypiltä vaaditun samaan aikaan.

Eniten aikaa vaati toimivan koodin luominen, joka toteutettiin kirjoittamalla erikseen moottorinohjauskoodi, joka yhdistettiin koodiin, joka mahdollisti mikrokontrollerin kommunikoinnin langattomasti tietokoneen kanssa, muodostaen näin yhtenäisen kokonaisuuden, jolla kaikki halutut toiminnot pystyttiin toteuttamaan helposti ja nopeasti.

Lähteet

1 Waspmote programming guide. 2014. Verkkodokumentti. Libelium. <http://www.libelium.com/downloads/documentation/waspmote_programming_guide.pdf>. Luettu 30.10.2015.

2 Waspmote technical guide. 2014. Verkkodokumentti. Libelium. <http://www.libelium.com/downloads/documentation/waspmote_technical_guide.pdf>. Luettu 30.10.2015.

3 WiFi Module getworking guide. 2014. Verkkodokumentti. Libelium. <http://www.libelium.com/downloads/documentation/wifi_networking_guide.pdf>. Luettu 30.10.2015.

4 RN-XV data sheet. 2011. Verkkodokumentti. Roving Networks. <http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Wireless/WiFi/WiFly-RN-XV-DS.pdf>. Luettu 30.10.2015.

5 Arduino DC motor code. 2012. Verkkodokumentti. Adafruit. <https://learn.adafruit.com/adafruit-arduino-lesson-13-dc-motors/arduino-code>. Luettu 30.10.2015.

6 Sachs, Joachim; Beijar, Nicklas; Elmdahl, Per; Melen Jan; Militano, Francesco; Salmela, Patrik. 9/2014. Capillary networks - a smart way to get things connected. Verkkodokumentti.Ericsson.<http://www.ericsson.com/res/thecompany/docs/publications/ericsson_review/2014/er-capillary-networks.pdf>. Luettu 30.10.2015.

7 Arduino Controlled L293D Robot. 2013. Verkkodokumentti. Instructables. http://www.instructables.com/id/Arduino-and-L293D-Robot-Part-1-/?ALLSTEPS>. Luettu 30.10.2015.

8 UM0894 User manual. 8/2012. Verkkodokumentti. ST Microelectronics. < http://www.stmicroelectronics.com.cn/st-web-ui/static/active/en/resource/technical/document/user_manual/CD00262415.pdf>. Luettu 30.10.2015.

9 Waspmote IDE User Guide. 1/2014. Verkkodokumentti. Libelium. < http://www.libelium.com/downloads/documentation/waspmote_ide_user_guide.pdf >. Luettu 20.10.2015.

10 L293, L293D Quadruple half-H drivers. 11/2004. Texas Instruments. < http://www.ti.com/lit/ds/symlink/l293d.pdf>. Luettu 30.10.2015.

11 Tmote Sky. 2006. Verkkodokumentti. Moteiv. < http://www.eecs.harvard.edu/ ~konrad/projects/shimmer/references/tmote-sky-datasheet.pdf>. Luettu 30.10.2015.

12 Serial. 2015. Verkkodokumentti. Arduino. < http://arduino.cc/en/reference/serial>. Luettu 30.10.2015.

13 Serial Communication. 12/2012. Verkkodokumentti. Sparkfun. <https://learn.sparkfun.com/tutorials/serial-communication>. Luettu 30.10.2015.

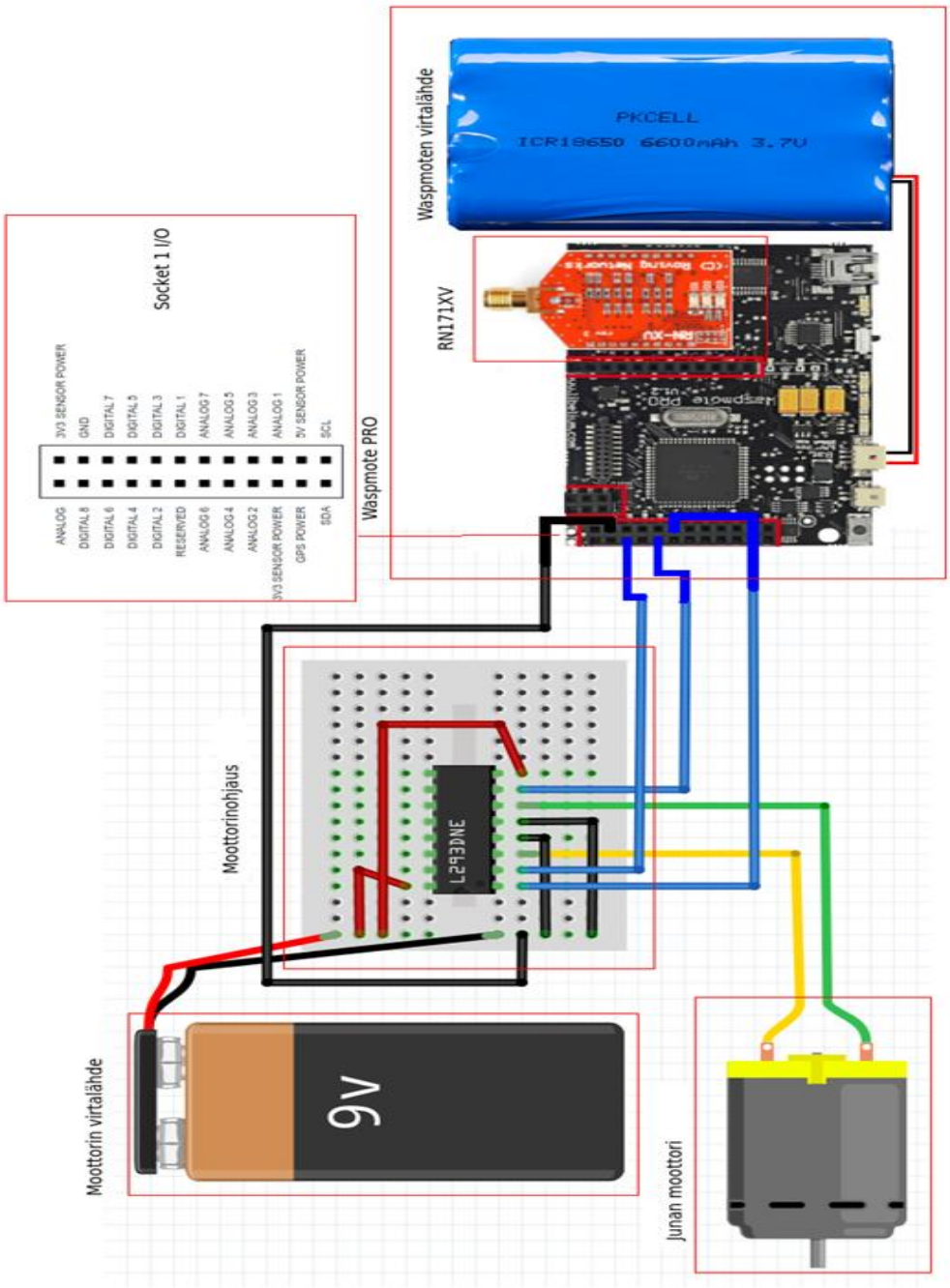
14 Technical overview. Verkkodokumentti. Libelium. < http://www.libelium.com/ products/waspmote/hardware/>. Luettu 30.10.2015.

15 Stripboard. Verkkodokumentti. Electronics Club. <http://electronicsclub.info/ stripboard.htm>. Luettu 30.10.2015.

16 Representational state transfer. Verkkodokumentti. Wikipedia. <https://simple.wikipedia.org/wiki/Representational_state_transfer>.Luettu 30.10.2015.

17 H bridge. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/H_bridge>. Luettu 30.10.2015.

Liitteet



Kuva 1 Moottorinohjauksen ulkonäkö

Prototyypin-koodi

```
 // Include WIFI library
 #include <WaspWIFI.h>
// choose socket (SELECT USER'S SOCKET
 ///////////////////////////////////////
 uint8_t socket=SOCKET0;
 // WiFi AP settings (CHANGE TO USER'S AP)
////////////////////////////////
#define ESSID "NOMADICLAB"
#define AUTHKEY "xxxxxxxxxxxxxxxx"
////////////////////////////////
// TCP server settings
////////////////////////////////
#define LOCAL_PORT 2000 //TCP port that your sensor is listening at
////////////////////////////////
char direction[10];
char speed[10];
char timer[100];
char url[513];
int speed_int;
int delaylenght;
void setup() {
  //USB.ON();
  delay(100);
 if( WIFI.ON(socket) == 1 )
 {
   USB.println(F("WiFi switched ON"));
      // reset to avoid previous configuration stored in the module
   //WIFI.resetValues();
   // 1. Configure the transport protocol (UDP, TCP, FTP, HTTP...)
   WIFI.setConnectionOptions(HTTP|CLIENT_SERVER);
   // 2. Configure the way the modules will resolve the IP address.
   WIFI.setDHCPoptions(DHCP_ON);
   // 3. Configure how to connect the AP
   WIFI.setJoinMode(MANUAL);
```

```
 // 4. Set the AP authentication key
 WIFI.setAuthKey(WPA1,AUTHKEY);
 // 5. Save Data to module's memory
 WIFI.storeData();
    // If it is manual, call join giving the name of the AP
 if( WIFI.join(ESSID) )
 {
  USB.println(F("Joined"));
      USB.println(F("-----------------------"));
  USB.println(F("get IP"));
  USB.println(F("-----------------------"));
  WIFI.getIP();
  USB.println();
     WIFI.cleanRemoteMessage();
      // Call the function to create a TCP connection on port 2000
  if (WIFI.setTCPserver(LOCAL_PORT))
  {
   USB.println(F("TCP server set"));
  }
  else
  {
   USB.println(F("TCP server NOT set"));
  }
 }
 else
 {
  USB.println(F("NOT joined to AP"));
 }
}
else
{
 USB.println(F("WiFi did not initialize correctly"));
}
   //Set up the variables
//          Sets          the          5V          switch          ON
```

```
  PWR.setSensorPower(SENS_5V, SENS_ON);
  delay(100);
  pinMode(DIGITAL6, OUTPUT);
  pinMode(DIGITAL2, OUTPUT);
  pinMode(DIGITAL1, OUTPUT); //ENABLEPIN -1
  analogWrite(DIGITAL1, LOW);
}
void loop() {
   // Reads from the TCP connection
   WIFI.read(NOBLO);
     if(WIFI.length>0)
   {
     strcpy(url, (char*)WIFI.answer);
          char chars_array[strlen(strtok(url, "\r\n"))+1];
          strcpy(chars_array, strtok(url, "\r\n"));
          if (chars_array!=NULL){
                ///Parsing the information GET /X/Y/Z  (X) = 0,1,2 DIRECTION - (Y)= 0-
255 SPEED - (Z) DELAY          if (strstr(chars_array, "GET")!=NULL){
            USB.println(chars_array);
            strtok(chars_array, " ");

            char tmp[20];
            strcpy(tmp, strtok(NULL, " "));
            //char sentence[100];
            //sprintf(sentence,"tmp %s\0", tmp);
            //USB.println(sentence);
            strcpy(direction, strtok(tmp, "/"));
            char sentence[100];
            sprintf(sentence,"direction %s\0", direction);
            USB.println(sentence);
            strcpy(speed, strtok(NULL, "/"));
            sprintf(sentence,"speed %s\0", speed);
            USB.println(sentence);
            speed_int = atoi( speed );
            analogWrite(DIGITAL1,                                    speed_int);
```

```
            strcpy(timer, strtok(NULL, "/"));
            sprintf(sentence,"timer %s\0", timer);
            USB.println(sentence);
            delaylenght = atoi(timer);
          //Stop the engine
          if (strcmp(direction,"0") == 0){
            digitalWrite(DIGITAL6, LOW);
            digitalWrite(DIGITAL2, LOW);
          }
         //forward
          else if(strcmp(direction,"1") == 0) {
            digitalWrite(DIGITAL6, HIGH);
            digitalWrite(DIGITAL2, LOW);
            delay (delaylenght*1000);

            if (delaylenght!=0){
              digitalWrite(DIGITAL6, LOW);
              digitalWrite(DIGITAL2, LOW);
            }
          }
          //back
          else if(strcmp(direction,"2") == 0) {
            digitalWrite(DIGITAL6, LOW);
            digitalWrite(DIGITAL2, HIGH);
            delay (delaylenght*1000);
             if (delaylenght!=0){
              digitalWrite(DIGITAL6, LOW);
              digitalWrite(DIGITAL2, LOW);
             }
          }
        }
        ////////////////////////////////////////////
        // Send the HTTP get/post query (specifying the WEB server so DNS is used)
        ////////////////////////////////////////      WIFI.flush();
        WIFI.send("HTTP\/1.1 200 OK\r\nContent-Type: text\/plain\r\nContent-Length:
0\r\n\r\n\r\n");
```

```
    }
}
```