Atte Pitkänen

Edullisten mikro– ohjainmodulien ominaisuuksien hyödyntäminen

Opinnäytetyö

Sähkö– ja automaatiotekniikan koulutus

2020

Kaakkois-Suomen ammattikorkeakoulu

Tekijä/Tekijät	Tutkintonimike	Aika
Atte Pitkänen	Insinööri (AMK)	Tammikuu 2020
Opinnäytetyön nimi Edullisten mikro– ohjainmodulien ominaisuuksien hyödyntäminen		48 sivua
Toimeksiantaja		
Ohjaaja Harri Kosonen		

Tiivistelmä

Opinnäytetyössä vertailtiin erilaisten mikro-ohjainten ominaisuuksia, sekä valmistajien tuotteita. Vertailtavia ominaisuuksia olivat laajennettavuus, ohjelmointi ja soveltuvat kohteet. Vertailuosuuden lisäksi työssä tutustuttiin Arduino– ympäristön käyttöön erilaisissa käytännön esimerkkisovelluksissa, kuten modulien ja antureiden käytössä.

Tutkielma on suunnattu erityisesti alan parissa työskenteleville sekä harrastajille toimimalla vertailuna erilaisille mikro-ohjaimille, sekä niiden ympärille rakennetuille ohjelmointiympäristöille.

Käytettyjä menetelmiä tutkimuksen aikana olivat valmistajien kotisivut, datalehdet, oma aikaisempi kokemus, sekä englanninkieliset artikkelit. Esimerkkisovelluksissa käytetty ympäristö oli elektroniikan laboratorio.

Ongelmia tutkimuksen aikana aiheutti tuotteiden valinnan rajaus, erilaiset termistöjen suomennokset, sekä datalehtien tulkinta.

Lopputuloksena on taulukkomainen vertailu ja pohdinta eri mikro–ohjaimista, sekä sulautetuista järjestelmistä, että niiden ominaisuuksista. Mikro– ohjainten ominaisuudet ovat vertailun perusteella parantuneet ja monipuolistuneet, vaikka itse komponentin koko on pysynyt lähestulkoon samana. Esimerkkisovelluksia on käsitelty vertailuosuuden jälkeen.

Asiasanat

tutkielmat, sulautettu tietotekniikka, mikro-ohjaimet

South-Eastern Finland University of Applied Sciences

Author (authors)	Degree	Time
Atte Pitkänen	Bachelor of Engineering	January 2020

Thesis title

Utilizing properties of low-cost micro-controller modules

48 pages

Commissioned by

Supervisor

Harri Kosonen

Abstract

Objective of thesis was to compare properties of different microcontrollers as well as different products of manufacturers. Compared properties were expendability, programming and appliable fields. Comparison section is followed by further introduction to Arduino–environment usage in different applications, like usage of modules and sensors.

Thesis is focused towards to people who work with embedded systems and hobbyists, acting as comparing sheet between different microcontrollers and embedded systems.

Used methods were manufacturer websites, datasheets, previous experience and different articles.

Troubles were built up around issues with choosing suitable devices to compare, different lingual terminology and datasheet reading in depth.

End result is comparison and pondering between different micro–controller and embedded systems of their various properties compilation made around in charts. Despite micro– controller's size has stayed relevantly same, properties of micro– controllers have improved as well as expanded in functionality. Application examples are discussed after comparison section.

Keywords

research, embedded systems, microcontrollers

SISÄLLYS

1 JOHDANTO

Mikro-ohjainten ominaisuudet ovat vuosien aikana kehittyneet huimaa vauhtia. Vaatimukset ovat nousseet niin suoritustehon, hinnan, kuin ohjelmoinnin osilta. Teollisuus on edellistä riippuvaisempaa automaatioon johtuen suurista tuotantokapasiteeteistä.

Tämän opinnäytetyön tavoitteena on tutustua ja vertailla tunnettujen mikro–ohjain valmistajien tuotteita keskenään.

Vertailun jälkeen tutustutaan tarkemmin Arduino–ympäristön käyttöön erilaisissa esimerkkisovelluksissa.

2 VALMISTAJIEN JA TUOTTEIDEN VERTAILU

Seuraavissa luvuissa tutustutaan mikro-ohjainten valmistajiin, sekä niiden valmistamiin oleellisimpiin tuotteisiin, sekä tuotteiden ominaisuuksiin. Tarkasteltavia ominaisuuksia olivat;

- Flash, EEPROM, RAM
- Kellotaajuus (MHz)
- Fyysiset kommunikointilinjat (UART, SPI, i2c, CAN)
- Käyttöjännite (V)
- I/O linjat
- Sisäiset ajastimet ja rekisterit
- Muut erityisominaisuudet (ADC, DAC, SD, DMA, RTC)

Seuraavaksi käsitteiden avausta;

1. Mikro-ohjaimen sisäiset muistit eroavat käyttötarkoituksen mukaan. Flash muistiin tallennetaan käytön aikana suorittuva koodi. Tätä koodia käytön aikana ei voi muuttaa. Muuttuvia muisteja käytön aikana ovat EEPROM (electrically erasable programmable read-only memory) ja RAM (Random Access Memory, keskusmuisti). EEPROM muistaa tilansa jännitteen poistuessa, mutta RAM:n tiedot häviävät. Tästä syystä esim. käyttäjätiedot numerolukoille tallennetaan EEPROM:iin, sillä muuten ne pitäisi tallentaa uudelleen sähkökatkon sattuessa. RAM muistiin tallennetaan muut käytön aikana tarvittavat tiedot, kuten lämpötila, mitä ei tarvitse pitää muistissa pitkiä aikoja.

2. Kellotaajuus (MHz) kertoo mikro-ohjaimen suoritusnopeuden. Kellotaajuus on jännitteestä riippuvainen; korkeampi käyttöjännite mahdollistaa nopeamman suuremman kellotaajuuden, mutta samalla virrankulutus kasvaa. Myöskin muistiväylän leveys vaikuttaa suoritusnopeuteen.

3. Kommunikointilinjoja käytetään mikro-ohjaimen, - päätelaitteen, - ja antureiden/sensoreiden väliseen kommunikointiin.

   3.1. UART ((Universal Asynchronous Receiver Transmitter) on sarjaväylä, missä on kaksi dataväylää; TX (transmit) ja RX (receive). Jos on tarkoitus vain joko lähettää, - tai lukea, voidaan toinen näistä linjoista jättää pois. UART:ssa määritetään TX ja RX päihin samat asetukset.

   3.2. SPI (Serial Peripheral Interface) on sarjaväylä, missä datalinjoja on neljä;

   -SCK (serial clock), eli kello, joka määrää siirtonopeuden.
   -MOSI (master out-slave in), väylä, minkä siirtosuunta on herralta orjalle, kuten komennot anturille.
   -MISO (master in-slave out), väylä, minkä siirtosuunta on orjalta herralle, kuten tiedot anturilta.
   -SS (slave select), väylä, millä aktivoidaan orja. SPI:ssä voi olla useampi orja rinnakkain.

   3.3. i2c (Inter-Integrated Circuit) on sarjaväylä. i2c:ssä on kaksi dataväylää;
   -DATA
   -CLK
   Kyseessä on osoitteellinen linja, eli laitteisiin viitataan osoitteen perusteella, ei fyysisellä aktivoinnilla. Tämän ansiosta, kuten SPI:ssä, i2c väylässä voi olla useampi laite yhtä aikaa liitettynä.

   3.4. CAN (controller area network) käytetään teollisuudessa ja ajoneuvoissa.

4. Erityisominaisuuksista merkittäviä olivat
   4.1. ADC (analog to digital converter), muuntaa analogisen tiedon digitaaliseen. Vaikuttavia tekijöitä muutokseen ovat referenssijännite sekä resoluutio.Esim. 5V referenssijännitteellä toimiva 10-bit ADC muuntaa jännitteen 0-5V digitaaliseen muotoon 0-1023. Suurempi resoluutio on hitaampi, mutta tarkempi, samalla myös häiriöalttius kasvaa.
   4.2. DAC (digital to analog converter) toimii kuten ADC, mutta mahdollistaa jännitteen muodostuksen digitaalisena. Nykyiset signaaligeneraattorit, sekä äänikortit käyttävät DAC: eja.
   4.3. SD (secure digital) kortille voidaan EEPROM:n tavoin tallentaa tietoja, kuten lokeja, esimerkiksi .CSV muodossa.
   4.4. RTC (real time clock) on päivyri, mikä ylläpitää ajankulkua käyttöjännitteen kadotessa. Yleensä RTC on patterivarmennettu, eli tarvitsee silti pienen patterin säilyttääkseen tiedot.
   4.5. DMA (direct memory access) mahdollistaa tiedon siirtämisen suoraan mikro-ohjaimen muistin ja ulkoisen laitteen välillä. Tästä on etua erityisesti suurten tietomäärien käsittelyssä, koska prosessori ei puutu tiedonsiirtoon. DMA: n toimiessa taustalla, prosessori voi tehdä suurempaa laskentatehoa vaativaa toimenpidettä.

2.1 Atmel (nykyinen microchip)

Atmel perustettiin vuonna 1984 Yhdysvalloissa. Sen tunnetuimmat tuotteet ovat AVR- sekä 8051 perheen mikrokontrollerit, kuten PIC.

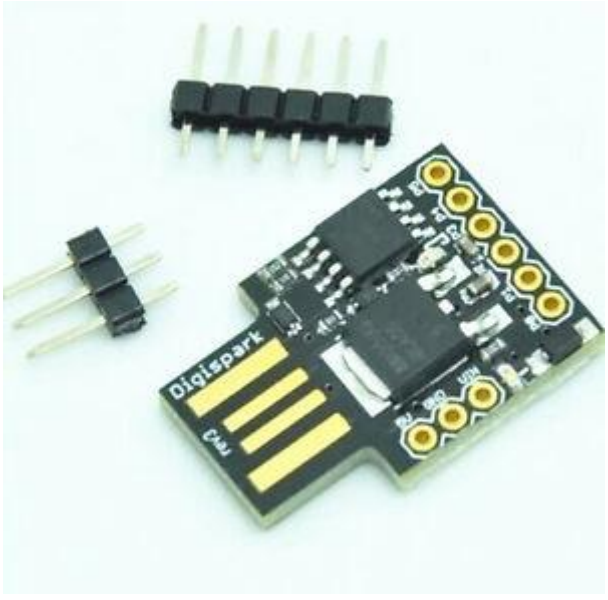
2.1.1 AVR

Luultavasti tunnetuin tuoteperhe Atmelilta, jossa tunnetuin alusta on Arduino. Arduino kehitettiin Wiring ympäristön pohjalta. Tarkoitus oli suunnitella helppokäyttöinen sekä edullinen ohjelmistoympäristö, mihin voitiin yhdistää erilaisia antureita, releitä, sekä muita komponentteja helposti. Kehitystyössä olivat mukana, Massimo Banzi, David Cuarielles sekä David Mellis. /1./

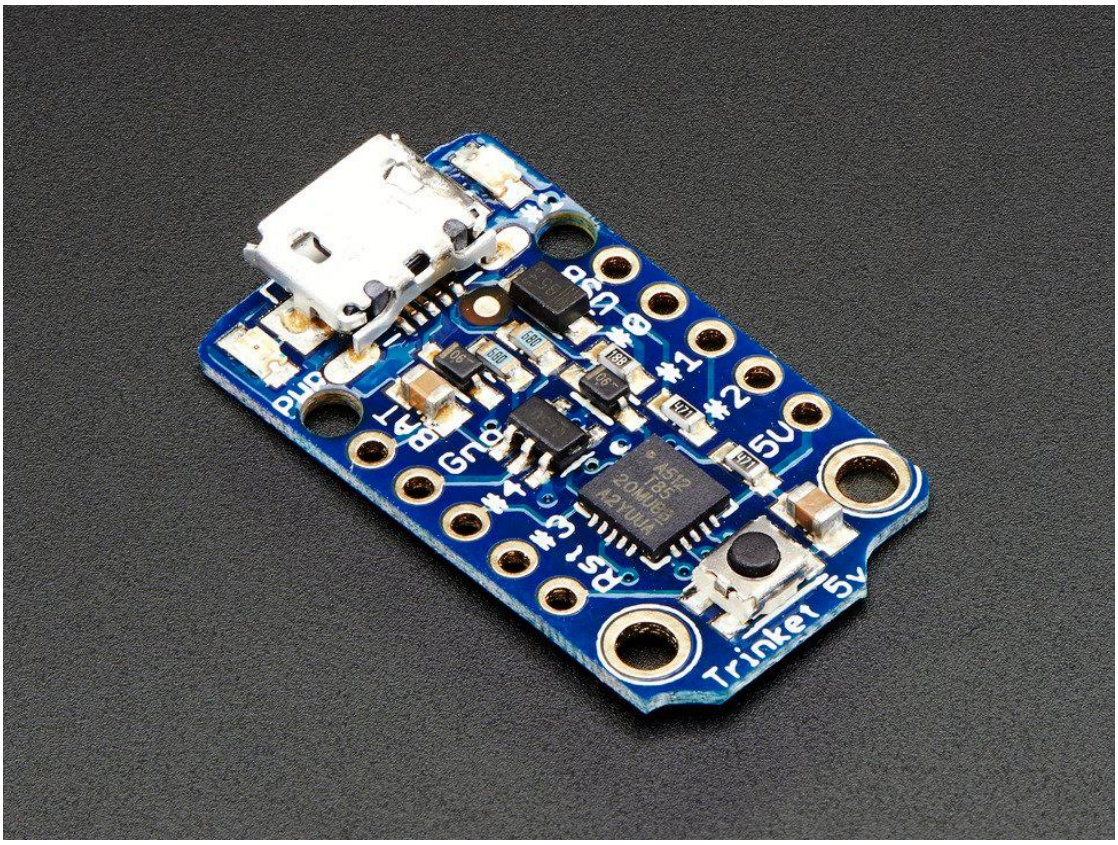
Arduinolla on suuri käyttäjäkunta ja käyttää avointa lähdekoodia. Ohjelmointiin käytetään pääsääntöisesti Arduino IDE ohjelmistoa.

2.1.2 AtTiny 25/45/85

AtTiny on pieni, 8-bittinen microchipin valmistama mikro-ohjain. joka käyttää AVR RISC arkkitehtuuria. AtTiny mikro-ohjaimia käytetään pienemmissä järjestelmissä, missä ei vaadita suurta laskentatehoa, tai suurta muistikapasiteettia. Esimerkkeinä alustoista on Digispark ja Adafruitin valmistama trinket-sarja (kuvat 1. ja 2).



Kuva 1 Digispark



Kuva 2 Trinket

Tärkeimmät ominaisuudet taulukossa 1.

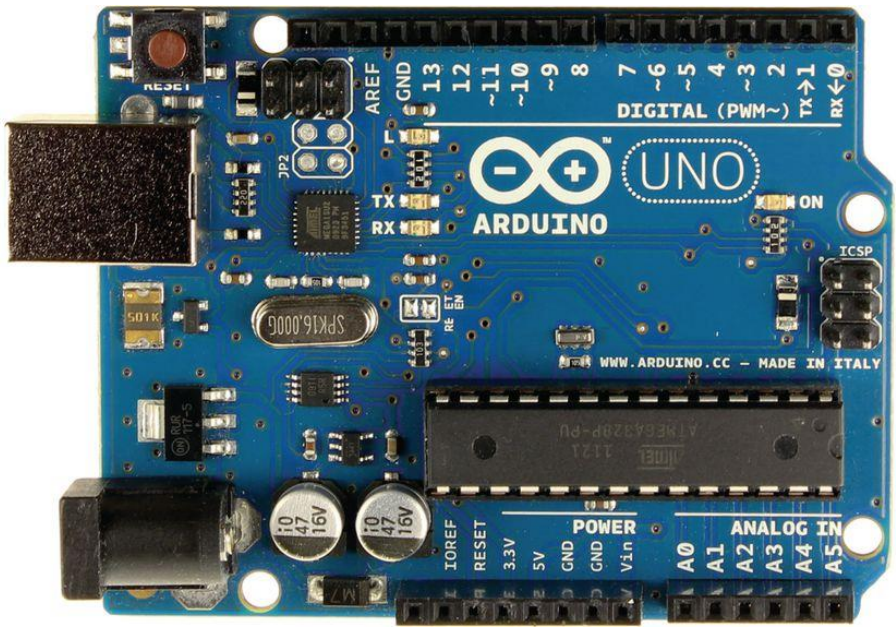
Taulukko 1 AtTiny25/45/85 ominaisuuksia. /2/.

	AtTiny25	AtTiny45	AtTiny85
Flash muisti (tavua)	2k	4k	8k
EEPROM (tavua)	128	256	512
SRAM (tavua)	128	256	512
Ajastin	8–bit kaksi 8–bittistä PWM		

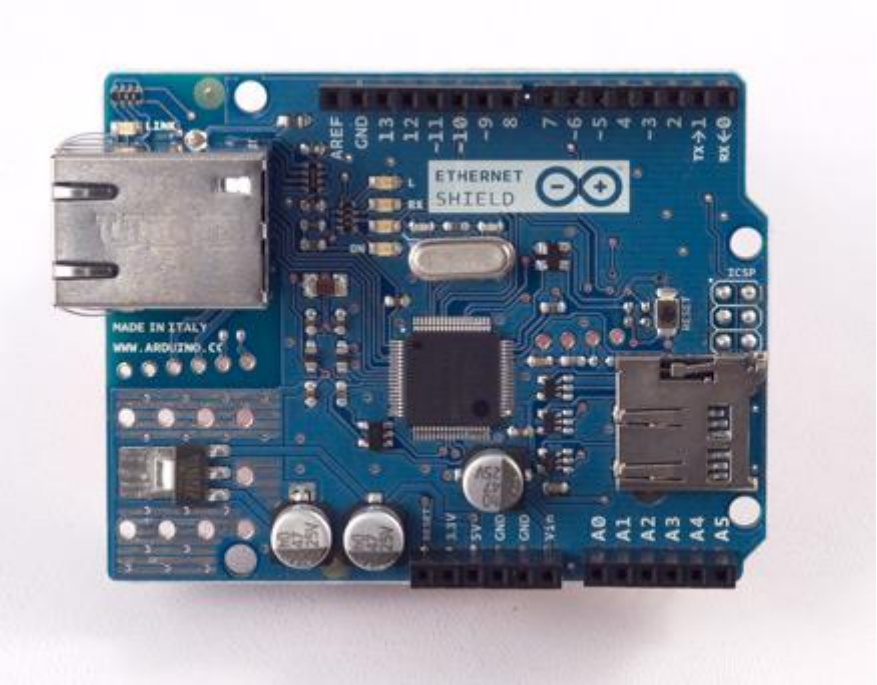
ADC	Neljä 10–bittistä
Käyttöjännite (V)	2.7-5.5
I/O linjat	6 ohjelmoitavaa
Ohjelmointi	ISP SPI
Kellotaajuus (MHz)	0-20 riippuen käyttöjännitteestä

2.1.3 AtMega 168/328

AtMega- mikro-ohjaimet ovat 8-bittisiä, mutta verrattuna aiemmin mainittuun AtTiny-sarjaan, AtMega on ominaisuuksiltaan parempi. Tunnettu alusta AtMega:lla on Arduino UNO (kuva 3**)**, joka on monelle harrastajalle ensimmäinen kosketus sulautettuihin järjestelmiin. Ulkomuodon ansiosta UNO:lla voidaan käyttää myös ulkoisia lisätarvikkeita, joita kutsutaan shieldeiksi. Shieldi kohdistetaan UNO:n päälle ja näitä voidaan laajentaa isommiksi kokonaisuuksiksi. Esimerkkejä shieldeistä on LCD ja ethernet-kortit (kuvat 4 ja 5).



Kuva 3 Arduino UNO



Kuva 4 Arduino ethernet shield

Kuva 5 Arduino LCD & keypad shield

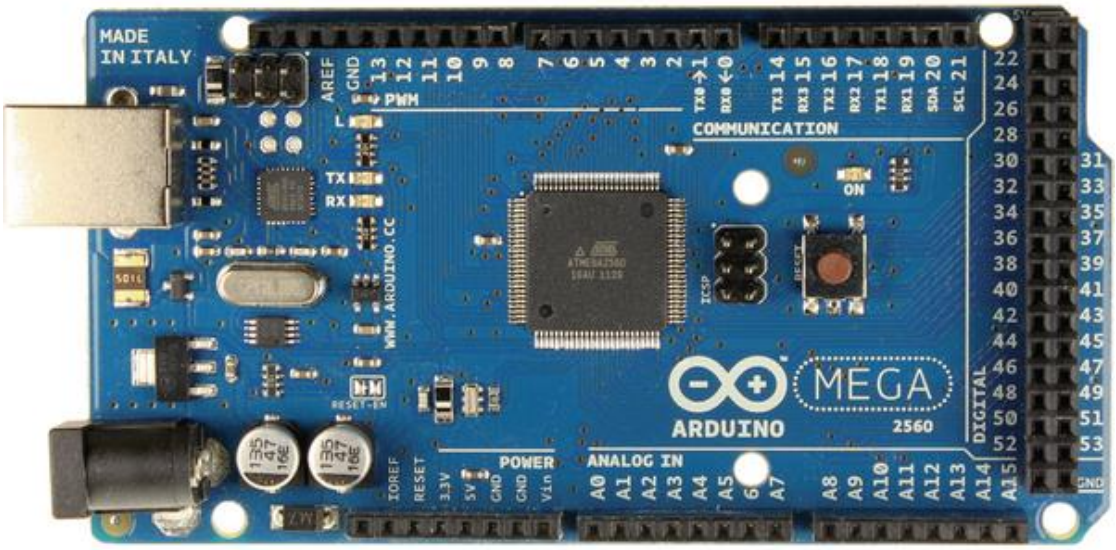
Taulukko 2 Atmega 168/328 ominaisuuksia. /3/.

	AtMega168	Atmega328
Flash muisti (tavua)	16k	32k
EEPROM (tavua)	512	1k
SRAM (tavua)	1k	2k
Ajastin	2 kpl 8- bittinen ja 1 kpl 16-bittinen. kuusi PWM lähtöä (4 kpl 8bit ja 2kpl 16bit)	
ADC	6- kanavainen 10 bittinen	
Käyttöjännite (V)	2.7-5.5	
I/O linjat	23 ohjelmoitavaa	
Ohjelmointi	ISP SPI	

Kellotaajuus (MHz)	0-20 riippuen käyttöjännitteestä
Muuta	-USART sarjaväylä -i2c sarjaväylä -SPI sarjaväylä -analoginen komparaattori -laitteistokeskeytykset

2.1.4 AtMega2560

AtMega2560 on vielä laajempi ja tehokkaampi verrattuna AtMega128/328 mikro-ohjaimiin. Tunnettu alusta AtMega2560: lle on Arduino Meag2560. At-Mega250 kykenee tehonsa ansiosta paremmin näyttösovelluksiin.



Kuva 6 Arduino Mega2560

Kun verrataan kuvia 3 ja 6, huomataan yhdennäköisyys muodossa. Arduino mega:n kanssa voidaan käyttää UNO:n suunniteltuja lisätarvikkeita. AtMega2560 ominaisuuksia taulukossa 3.

Taulukko 3 Atmega 2560 ominaisuuksia. /4/.

	AtMega2560
Flash muisti (tavua)	256k

EEPROM (tavua)	4k
SRAM (tavua)	8k
Ajastin	kaksi kpl 8- bittistä neljä 16- bittistä neljä 8- bittistä PWM 12 kpl 2-16 bittistä PWM lähtöä.
ADC	6- kanavainen 10-bittinen
Käyttöjännite (V)	2.7-5.5
I/O linjat	96 ohjelmoitavaa
Ohjelmointi	ISP SPI
Kellotaajuus (MHz)	0-16 riippuen käyttöjännitteestä
Muuta	-USART sarjaväylä -JTAG -i2c sarjaväylä -SPI sarjaväylä -analoginen komparaattori -laitteistokeskeytykset

Kun verrataan AtTiny, AtMega328, sekä AtMega2560, fyysisen koon kasvaessa myös ominaisuudet muuttuvat. Kuitenkin kaikki käyttävät samaa käyttöjännitettä toimien myös samalla kellotaajudella ja käyttäen samaa ohjelmointitekniikkaa. Kaikista malleista puuttuu DAC, mutta ulkoisia moduuleita on silti hyvin käytettävissä, sekä i2c, että SPI liitännällä. CAN väylää ei myöskään ole malleissa.
Myöskään murtolukujen laskennassa hyödyllistä FPU:ta ei ole, tai DMA:ta.

Muistien määrä vaihtelee oleellisesti myös, sekä I/O valmiudet. Esimerkiksi parhaimmillaan flash muistia on 256k ja 96 I/O liitännällä AtMega2560 tapauksessa, verrattuna AtTiny 2k flash muistiin 6: lla I/O linjalla. Etuna AtTiny:ssä on todella pieni koko.

2.1.5 Arduino IDE

Arduino IDE (integrated development environment), kuten edellä mainittiin, on Arduinon ohjelmointiin käytetty ohjelmointiympäristö. Ohjelmointikielenä käytetään c/c++ muunnosta. Arduino IDE tukee myös muiden mikrokontrollerien ohjelmointia, kuten STM ja ESP erilaisten liitännäisten avulla. Käytettävissä olevat kirjastot (libraries) mahdollistavat myös erinäisten komponenttien, kuten näyttöjen ja antureiden käytön sulavammin, antureiden käyttämän koodin ollessa piilossa.

IDE:een on olemassa tekstipohjaisen lisäksi graafisia laajennuksia, kuten arduBlock . ArduBlockin etuna tekstipohjaiseen verrattuna on koodin visuaalisen hahmottamisen helppous.

2.2 STM

Alun perin erilliset yhtiöt SGS Microelettronica ja Thomson Semiconducteurs, jotka yhdistyivät yhtiöksi nimeltään SGS-Thomson vuonna 1987. SGS-Thomson nimettiin myöhemmin nykyisin tunnetuksi STMicroelectronics 1998. Päämaja sijaitsee Sveitsissä. /5./

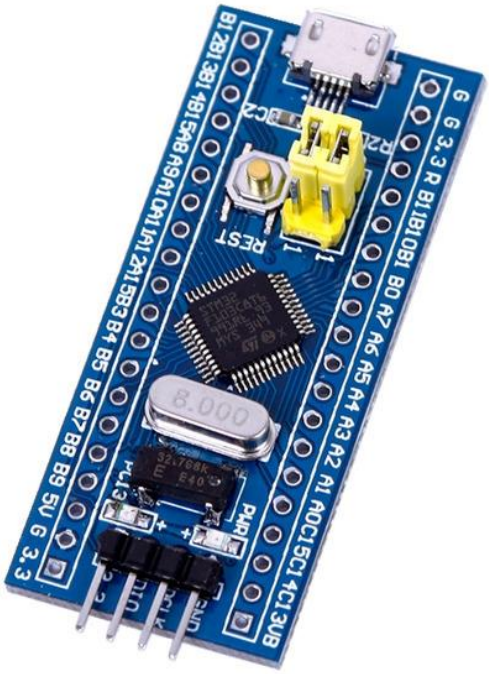
STM valmistaa puolijohteita teollisuuteen ja johtava avainkomponenttien valmistaja IoT:hen ja älykkäisiin ajoneuvoihin

Mikro-ohjaimia STM valmistaa eri tarkoituksiin 8–32 bittisinä versioina.

Opinnäytetyöhön valittiin tunnettuja STM ARM mikro-ohjaimia;

2.2.1 STM32

STM32 sarjassa tunnettu mikro-ohjain on STM32F103C8T6, jota käytetään mm. “blue pill” kehitysalustassa (kuva 7).



Kuva 7 STM32F

Ominaisuuksia taulukossa 4.

Taulukko 4 STM32F103C8T6 ominaisuuksia. /6/.

	STM32F103C8T6
Flash muisti (tavua)	64k
SRAM (tavua)	20k
Ajastin	3kpl 16-bit ajastin 16 bit PWM
ADC	10x12 bit
Käyttöjännite (V)	0-3.6
I/O linjat	32
Ohjelmointi	FTDI ja vaihtoehtoisesti USB
Kellotaajuus (MHz)	72
Muuta	• DMA (seitsemän kanavaa) • RTC • USART (kaksi kpl.) • SPI • i2c • USB

	• CAN

Verrattuna aikaisemmin käsiteltyihin Atmel:n tuotteisiin, STM32F103C8T6 täydentää puuttuvia ominaisuuksia hyvin. Esimerkiksi DMA, sekä CAN on käytettävissä, kuten myös suurempi kellotaajuus 72MHz ja integroitu RTC. ADC on myös tarkempi; jopa 12bit. I/O linjoja on vähemmän verrattuna AtMega2560:iin. Huomioitavaa on pienempi käyttöjännite.

2.3 NXP

NXP, (entinen Philips) on Alankomainen puolijohdevalmistaja, joka valmistaa sovelluksia sähkö–ajoneuvoihin, teollisuuteen sekä mobiiliteollisuuteen.

Opinnäytetyöhön valittiin mikro-ohjaimet;

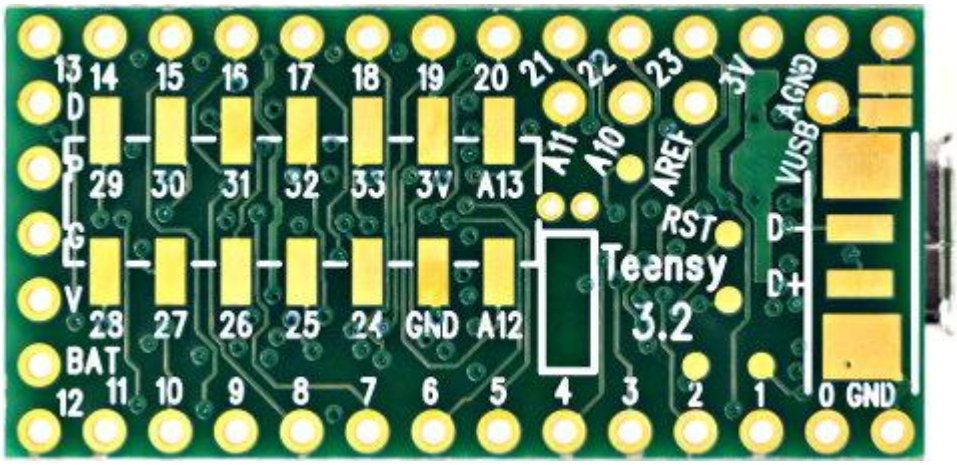
- MK20DX256VLH7 (Cortex M4 ydin)
- MK66FX1M0VMD18 (Cortex M4F ydin)

Yksi oleellinen ero 32 ja 32F välillä on murtolukujen käsittelyyn käytetty FPU, joka on noin 30 kertaa nopeampi verrattuna koodipohjaiseen laskemiseen FPU:sta on etua erityisesti matematiikan monimutkaisemmissa funktioissa, kuten FFT.

2.3.1 MK20DX256VLH7

MK20DX256VLH7 mikro-ohjainta käytetään mm. teensy 3.2 alustassa (kuvat 8 ja 9). I2S:n kanssa voidaan käyttää kuvassa 10 näkyvää audio shieldiä, mikä kiinnittyy aiemmin käsitellyn arduinon shieldien tavoin.

Kuva 8 Teensy 3.2 yläpuoli



Kuva 9 Teensy 3.2 alapuoli

Kuva 10 Teensy audio shield

MK20DX256VLH7 ominaisuuksia taulukossa 5.

Taulukko 5 MK20DX256VLH7 (teensy 3.2) ominaisuuksia. /7/.

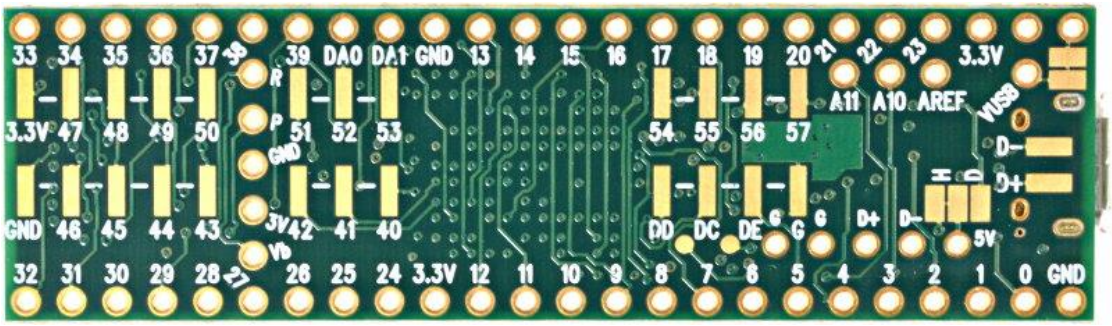
	MK20DX256VLH7
Flash muisti (tavua)	256k
SRAM (tavua)	64k
EEPROM (tavua)	2048
Ajastin	yhteensä 12kpl, kaikissa PWM valmius
ADC	21 x 16 bit.
Käyttöjännite (V)	1.71–3.6
I/O linjat	34
Ohjelmointi	USB
Kellotaajuus (MHz)	72
Muuta	• DMA (16 kanavaa) • RTC • USART (kolme kpl.) • SPI • i2c (kaksi kpl) • i2s (ääni i2c väylää pitkin) • USB • CAN • DAC (12 bit)

Kuten STM32F edellisessä luvussa, MK20DX256VLH7 on vähänlaisesti I/O linjoja. Liitännät ovat silti monipuoliset, kuten i2s, useampi USART ja integroitu DAC. ADC:n resoluutio on maksimissaan 16bit, mutta näin korkea resoluutio aiheuttaa jo haasteita piirilevyjen suunnittelussa. Myöskin DMA löytyy peräti 16 kanavaisena.

2.3.2 MK66FX1M0VMD18

MK66FX1M0VMD18 mikro-ohjainta käytetään mm. teensy 3.6 alustassa (kuvat 11 ja 12)

Kuva 11 Teensy 3.6



Kuva 12 Teensy 3.6

MK66FX1M0VMD18 ominaisuuksia taulukossa 6.

Taulukko 6 MK66FX1M0VMD18 (Teensy 3.6) ominaisuuksia. /8/.

	MK66FX1M0VMD18
Flash muisti (tavua)	1024k
SRAM (tavua)	256k
EEPROM (tavua)	4096
Ajastin	yhteensä 19kpl, kaikissa PWM valmius
ADC	25 x 16 bit.
Käyttöjännite (V)	1.71–3.6
I/O linjat	58
Ohjelmointi	USB
Kellotaajuus (MHz)	180, kellotettavissa 240 asti
Muuta	• DMA (16 kanavaa) • RTC • USART (6 kpl.) • SPI (3kpl.)

	• i2c (neljä kpl) • i2s (ääni i2c väylää pitkin) • USB (2 kpl.) • CAN (2 kpl.) • DAC (12 bit) • SD–korrtipaikka

Tähänastisista mikro-ohjaimista tehokkaimpana, MK66FX1M0VMD18 yltää jopa 240MHz kellotaajuuteen maksimissaan 1024k (1M) flash muistilla. I/O linjoja on vähemmän, kuin AtMega2560: ssa, mutta murtolukujen laskussa hyödyllinen FPU löytyy. Teensy 3.6 alustassa on myös valmis SD–korttipaikka.

2.4 Espressif

Espressif on tunnettu erityisesti IoT:ssa erilaisten langatonta yhteyttä käyttävien alustojen kautta, kuten Wi-Fi ja Bluetooth. Esimerkkeinä ESP8266 eri variaatiot, kuten ESP-01, sekä uudemmat ESP32. Esperssif on vuonna 2008 perustettu ja päämaja sijaitsee Shangaissa. /9/.

Opinnäytetyöhön valittiin ESP8266 ja ESP32.

2.4.1 ESP8266EX

ESP8266 on SoC (system on chip), jossa on integroitu Wi-Fi. Yleisimmät liitännät löytyvät, kuten USART, SPI ja i2c. ESP8266 voidaan asettaa toimimaan itsenäisesti ilman ulkopuolista isäntää. Ojelmointialustana esimerkkinä ESP-launcher (kuva 13).

Kuva 13 ESP-launcher

ESP8266 ominaisuuksia taulukossa 7.

Taulukko 7 ESP8266 ominaisuuksia. /10/.

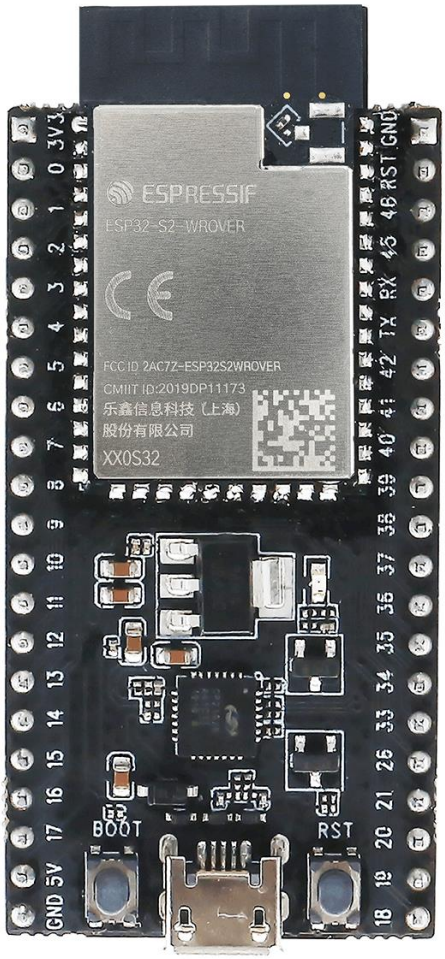
	ESP8266
Flash muisti (tavua)	2M
RAM (tavua)	50k
ADC	10–bit
Käyttöjännite (V)	2.5–3.6
I/O linjat	32
Ohjelmointi	UART
Kellotaajuus (MHz)	80–160
Muuta	• Wi-Fi protokolla 802.11 b/g/n/e/I • Taajuusalue 2.4–2.5 GHz • 2 kpl. UART • SPI • I2C • I2S

	• 4 kpl. PWM

ESP8266 yksi heikkous on sisäisen EEPROM:n puuttuminen. Myöskään DAC:ia ei ole.

2.4.2 ESP32

Kuten edellä käsitelty ESP8266, myös ESP32: sta löytyy integroitu Wi-Fi. ESP32 on kuitenkin tehokkaampi suorituskyvyltään verrattuna ESP8266: een. ESP32: sta käytetään mm ESP32-32-Saola alustassa (kuva 14).



Kuva 14 ESP32–32–Saola

ESP32 ominaisuuksia taulukossa 8.

Taulukko 8 ESP32 ominaisuuksia. /11/.

	ESP32
Flash muisti (tavua)	4M
RAM (tavua)	320k
ROM (tavua)	128k
ADC	2x12 bit (yhteensä 20 kanavaa)
DAC	2x8 bit
Käyttöjännite (V)	2.8–3.6
I/O linjat	43
Ohjelmointi	USB-UART
Kellotaajuus (MHz)	80–240
Muuta	• Wi-Fi protokolla 802.11 b/g/n • Taajuusalue 2.4–2.5 GHz • 2 UART • 4 SPI • 2 I2C • 1 I2S • 8 PWM • RTC

ESP32 on kaikilta ominaisuuksiltaan parempi verrattuna ESP8266:een. Jo muistia ja kellotaajuutta on yli kaksinkertainen määrä ja liitäntöjä enemmän ja laajemmin.

3 ESIMERKKISOVELLUKSIA ARDUINO–YMPÄRISTÖSSÄ

Seuraavassa kappaleessa tutustutaan tarkemmin Arduino–ympäristön käyttöön erilaisissa sovelluksissa. Arduino–ympäristöön päädyttiin edullisen hinnan, helppokäyttöisyyden sekä hyvän laajennettavuuden perusteella.

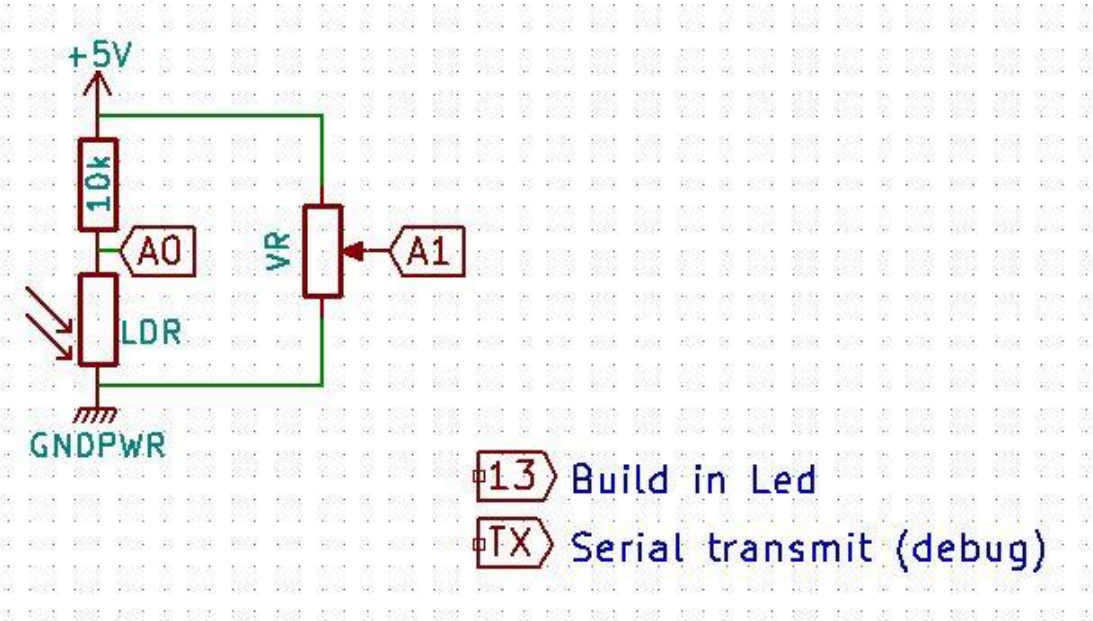
Sovelluksissa on esillä käytetyt varusteet, kytkentäkaaviot, kuvat kytkennöistä, sekä ohjelmistokoodit. Ohjelmistokoodin joukossa on myös kommentit tärkeistä koodin osista.

3.1 LDR–vastus

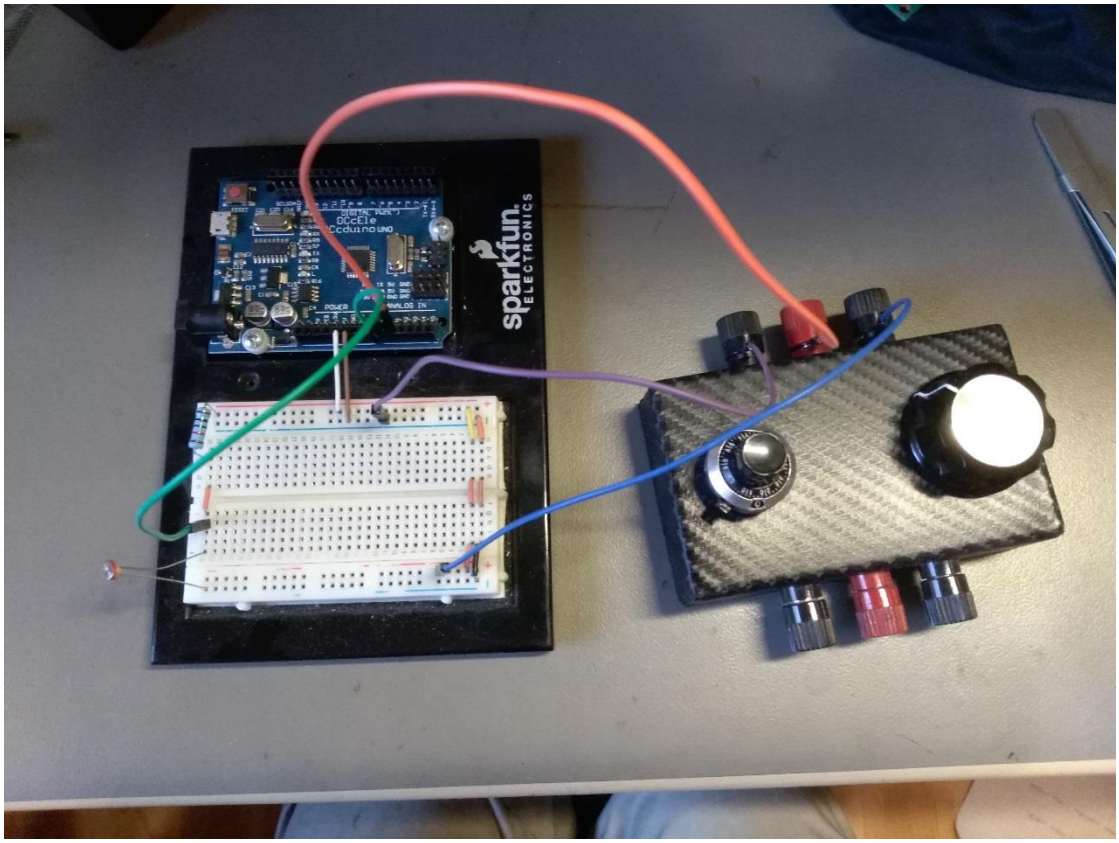
Työssä tutustutaan LDR–vastuksen käyttöön. Käytetyt komponentit olivat;

- CdS (kadniumsulfidi) LDR–vastus
- Potentiometri

Kytkentäkaavio kuvassa 15. Kuvassa 16 kuva kytkennästä.



Kuva 15 LDR kytkentäkaavio



Kuva 16 LDR kytkentä

Kytkennässä oli kaksi jännitteenjakoa; toisen muodosti LDR–vastus ja 10kΩ vastus. Tätä jännitteenjakajaa luettiin Arduinon A0 nastalla. Toinen muodostui potentiometristä, joka oli kytketty jännitteenjakajaksi ja jota luettiin Arduinon A1 nastasta. Ulostulona käytettiin Arduinon sisäänrakennettua LED:iä.

Kytkennän toiminta;

LDR jännitteenjakajan jännite muuttuu vallitsevan ympäristön valaistuksen mukaan. Potentiometrin jännitteenjakan jännite muuttuu käyttäjän säätäessä potentiometriä. Näitä kahta jännitettä verrataan koodissa ja hystereesin avustamana LED syttyy tai sammuu. Jännitteet muutetaan digitaaliseen muotoon ADC muuntimella, joka Arduinossa on sisäänrakennettu.

Sarjaväylää voidaan käyttää vianetsintään, kuten LDR:n toiminnan varmistamiseen.

Koodi on toteutettu pääasiassa käyttämällä if() funktiota, jota käytetään ADC:n tiedon ja hystereesin kanssa määrittämään LED:n tila.

Koska käytetty valovastus on resistiivisyydeltään epälineaarinen, kytkennän käyttö himmenninkäytössä on hankalaa.

Seuraavaksi ohjelmistokoodi;

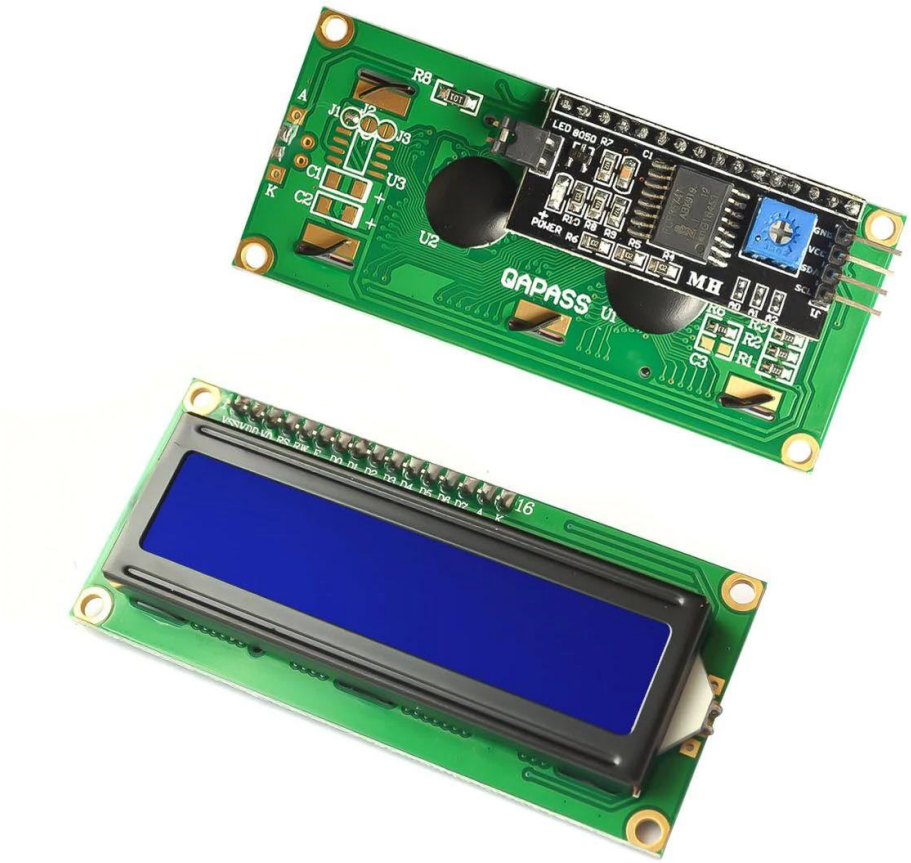
```
int LDRvalue, Controlvalue, triggeredvalue; //muuttuvat tiedot LDR:ltä ja poti-kalta

int Hysteresis = 50;       //Erilliset hystereesit LDR:lle ja säädölle
int Controlhysteresis = 10;

byte Led = 13;             //ohjattava lähtö, sisäänrakennettu Led
boolean debug = 0;      //vianetsintään koodissa, oletuksena false.

void setup()
{
  if (debug == true)
  {
    Serial.begin(9600); //debuggaukseen
  }
  pinMode(13, OUTPUT);
}

void loop()
{
  LDRvalue = analogRead(A0);      //pimeä=suuri impedanssi->arvo nousee. Valoisa=pieni impedanssi->arvo laskee
  Controlvalue = analogRead(A1);  //valoisa-pimeä

  if (debug = true) //Vianetsinnän rutiini
  {
    Serial.print("LDR=");
    Serial.print(LDRvalue);
    Serial.print('\t');
    Serial.print("Control=");
    Serial.println(Controlvalue);
```

```
  }

  if (LDRvalue > Controlvalue)                    //LDR:n alirutiini
  {
    digitalWrite(13, HIGH);
    triggeredvalue = LDRvalue;
  }

  if (LDRvalue - Hysteresis > triggeredvalue )      //Otetaan LDR:n hystereesi
mukaan
  {
    digitalWrite(13, LOW);
  }

  if (LDRvalue < Controlvalue - Controlhysteresis) //Säädön alirutiini hysteree-
sillä
  {
    digitalWrite(13, LOW);
  }
}
```

3.2 Ultraäänianturi+LCD

Työssä tutustutaan ultraäänianturiin, tyypiltään HC-SR04 (kuva 17), sekä LCD näyttöön (kuva 18).

Kuva 17 HC-SR04

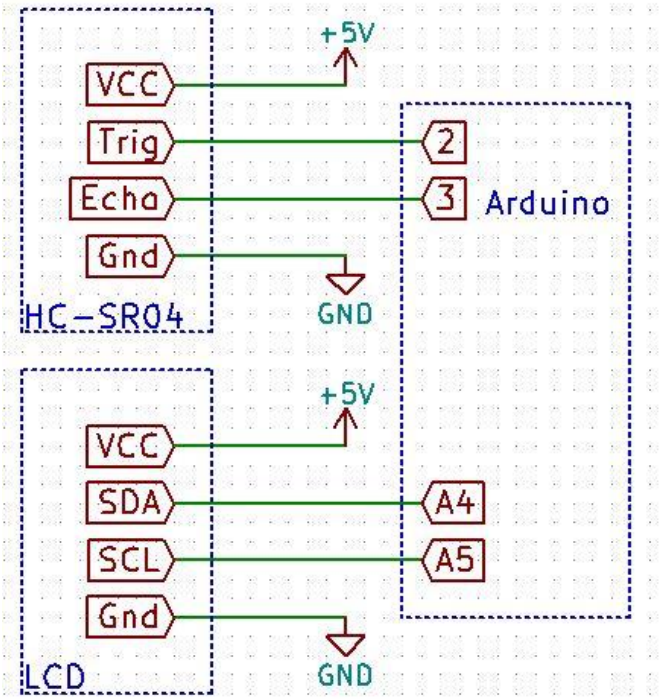


Kuva 18 LCD, jossa valmiina i2c laajennin

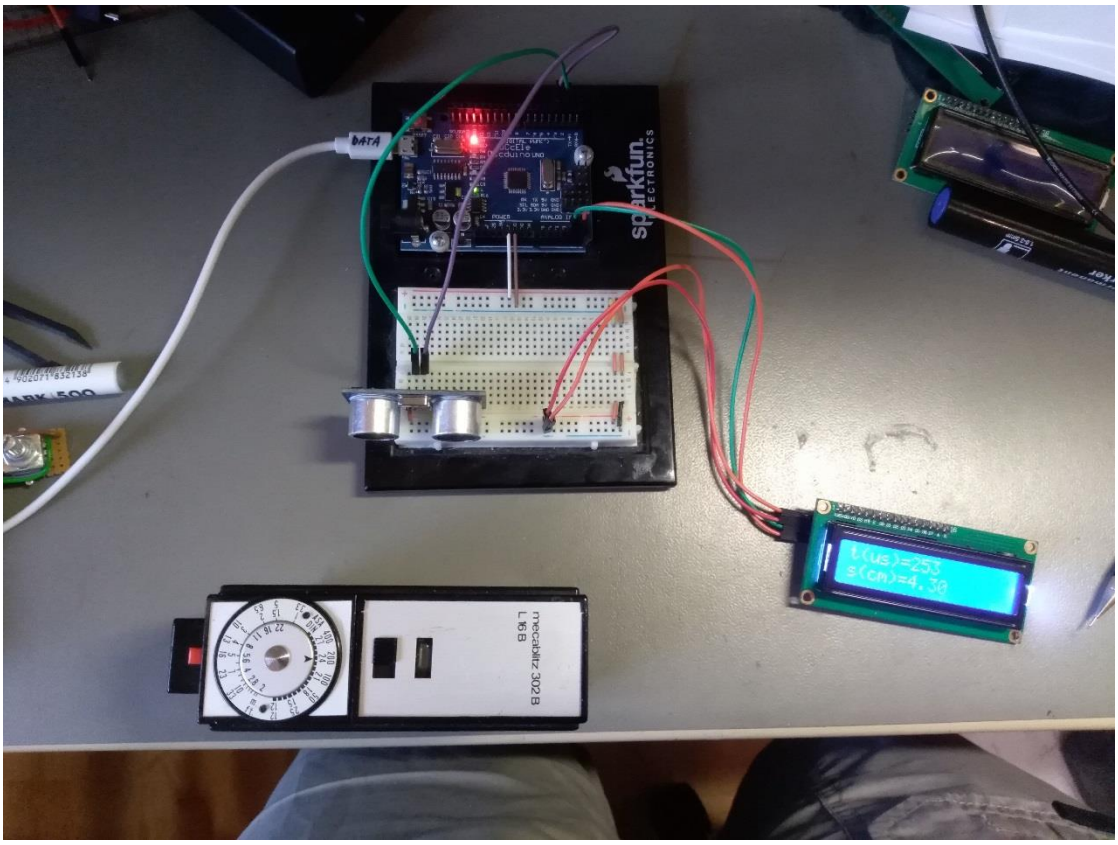
Normaalisti näyttöä ohjataan rinnakkaisväylällä, mutta tässä esimerkissä käytetään i2c–laajennin, joka on helpompi kytkeä ja käyttää rinnakkaisväylän sijasta i2c sarjaväylää. i2c näkyy valmiiksi asennettuna kuvassa 18.

Kytkennässä ultraäänianturilla lähetetään ensin äänipulssi, joka heijastuu takaisin esteestä. Ultraäänianturi lukee tämän heijastuksen ja Arduinolla lasketaan heijastukseen kuluneen ajan perusteella matka esteeseen. Tämä matka edelleen näytetään LCD:ssä.

Kytkentäkaavio kuvassa 19 ja kuva kytkennästä kuvassa 20.



Kuva 19 HC–SR04+LCD kytkentäkaavio



Kuva 20 HC–SR04+LCD kuva kytkennästä. Esteenä kameran salama

Ohjelmistokoodi:

```
/* Luetaan etäisyys käyttämällä HC-SR04 ultraäänianturia ja näytetään sekä
aika, että etäisyys LCD:llä */


#include <LiquidCrystal_I2C.h>

 /* kirjasto, mitä tässä käytetään. Se on ladattava Arduino IDE:ssä "Tools -
Manage Libraries"  alta hakuavaimella: "LiquidCrystal_I2C", ja siellä se on
ehkä 9:s vaihtoehto nimellä LiquidCrystal I2C, by Frank  de Brabander Ver-
sion 1.1.2  (jos perässä lukee installed, se on jo sinulle asennettu)

Kun vie hiiren kursorin sen kirjaston alueelle, tulee näkyviin painonappi ”In-
stall”, jota klikkaamalla kirjasto latautuu IDE:en */


LiquidCrystal_I2C lcd(0x27, 16, 2);  //määritetään lcd; annetaan sen osoite
                                              // sekä merkkien määrä (16x2),
                                              // joillain LCD-näytöillä osoite on 0x3F


byte trigger = 2; //liipaisu, millä laitetaan ultraääni etenemään

byte echo = 3;    //kaiku takaisin kohteelta anturille

long echoTime;  //mikrosekunnit tarvitsevat paljon tilaa, joten käytetään long
muistityyppiä

float distance;    //pitää olla float, koska kaikuaika kerrotaan desimaaliluvulla


void setup()

{

  lcd.init();

  lcd.backlight();

  pinMode(trigger, OUTPUT);

  pinMode(echo, INPUT);

  lcd.setCursor(0, 0);  //nämä eivät muutu, joten voidaan kirjoittaa näyttöön heti

  lcd.print("t(us)=");

  lcd.setCursor(0, 1);

  lcd.print("s(cm)=");

}

void loop()

{
```

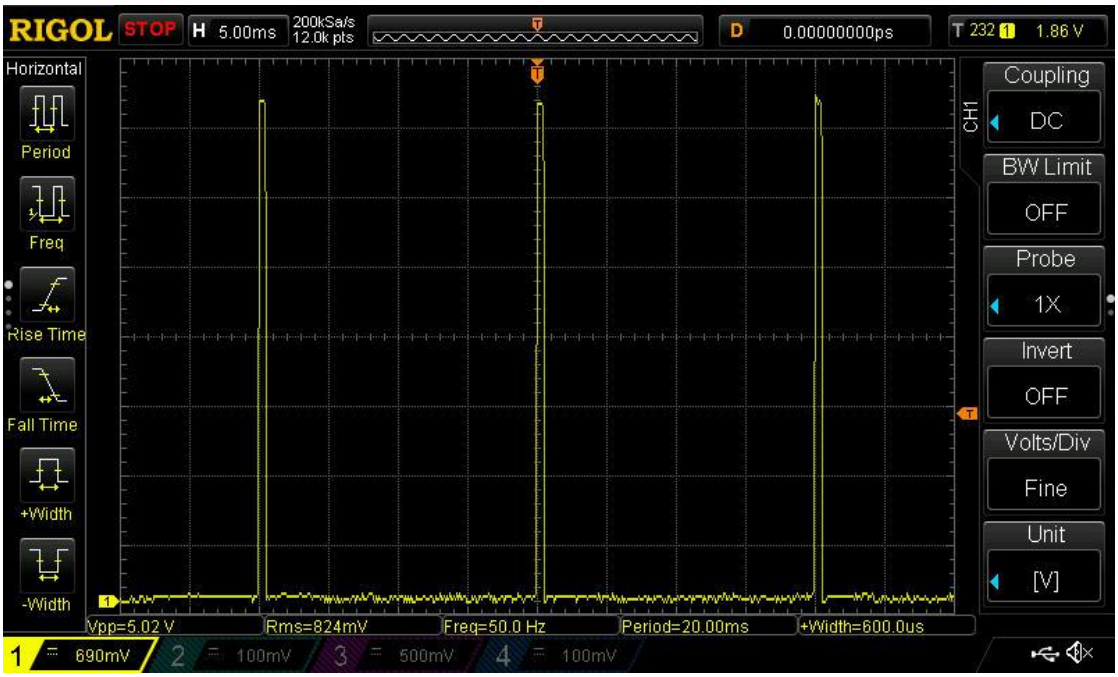
```
  digitalWrite(trigger, HIGH);
  delayMicroseconds(10);  //anturi vaatii 10us viiveen liipaisulle. delay() on liian
                                    //epätarkka, joten käytetään delayMicrose-
conds().
  digitalWrite(trigger, LOW);


  echoTime = pulseIn(echo, HIGH);
  distance = echoTime * 0.034 / 2;  // Tämän laskun takia etäisyys on float
tyyppiä


  /*
    anturin lähettämä ultraääni kulkee nopeudella 340m/s
   (riippuvainen lämpötilasta, mutta ei oteta huomioon demotapauksessa)
    echoTime on aika, mikä kuluu sekä heijastukseen, että kohteeseen osumi-
seen.
    nopeus on matka/aika, eli v=s/t
    Tunnetaan nopeus(v) ja aika (s), joten etäisyys saadaan kaavalla s=tv.
    Kaavaa saadaan näyttämään järkevämpiä lukemia käyttämällä 0.034cm/us,
    jolloin matka näkyy sentimetreinä metrien sijaan.
    Myös matka puolitetaan, sillä nyt se sisältää matkan molempiin suuntiin.
  */
  lcd.setCursor(6, 0); //näytetään lukemat
  lcd.print(echoTime);
  lcd.setCursor(6, 1);
  lcd.print(distance);


  delay(1000);  //odotetaan ja tyhjennetään näyttö muuttuvilta osin
  lcd.setCursor(6, 0);
  lcd.print("          ");
  lcd.setCursor(6, 1);
  lcd.print("          ");
}
```

3.3 Servon ohjaus PWM:llä

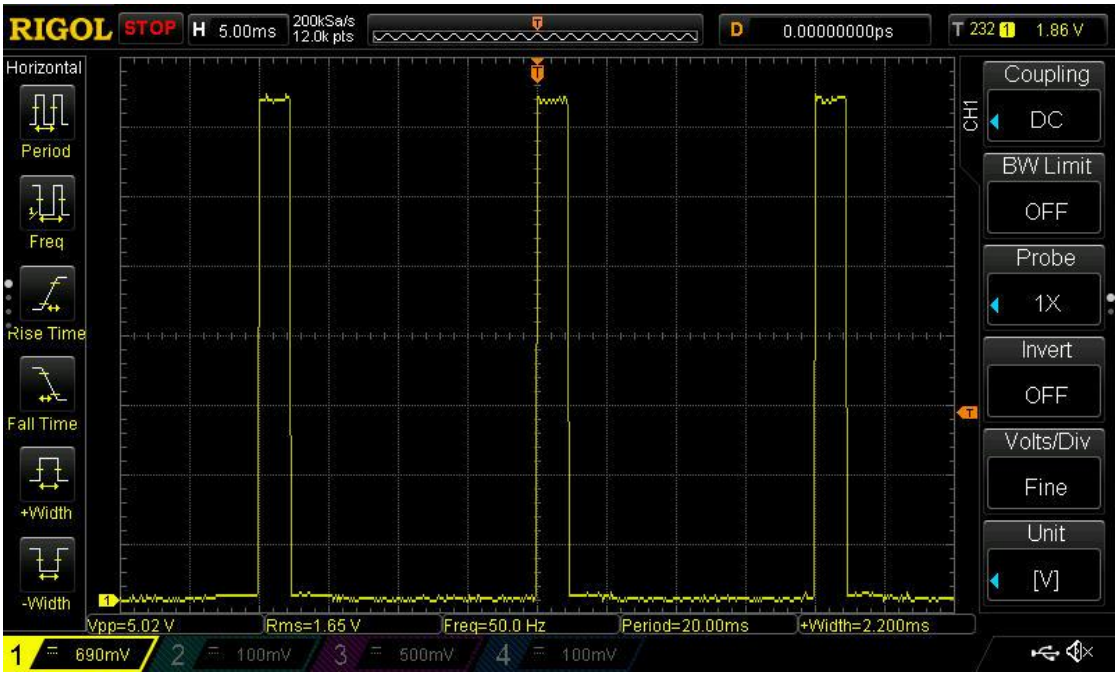
Työssä ohjataan servomoottorin kulmaa PWM:llä.

Valitun servon (SM-S2309S, kuva 21) kulmaa säädetään potentiometrillä, jota Arduino lukee ADC:llä ja muuntaa kirjaston avulla PWM:ksi, jonka taajuus on 50Hz. Kulma määräytyy edelleen pulssinleveyden kautta (kuvat 22 ja 23. Otettu käyttämällä Rigol DS1054Z oskilloskooppia).

Kuva 21 SM-S2309S servo



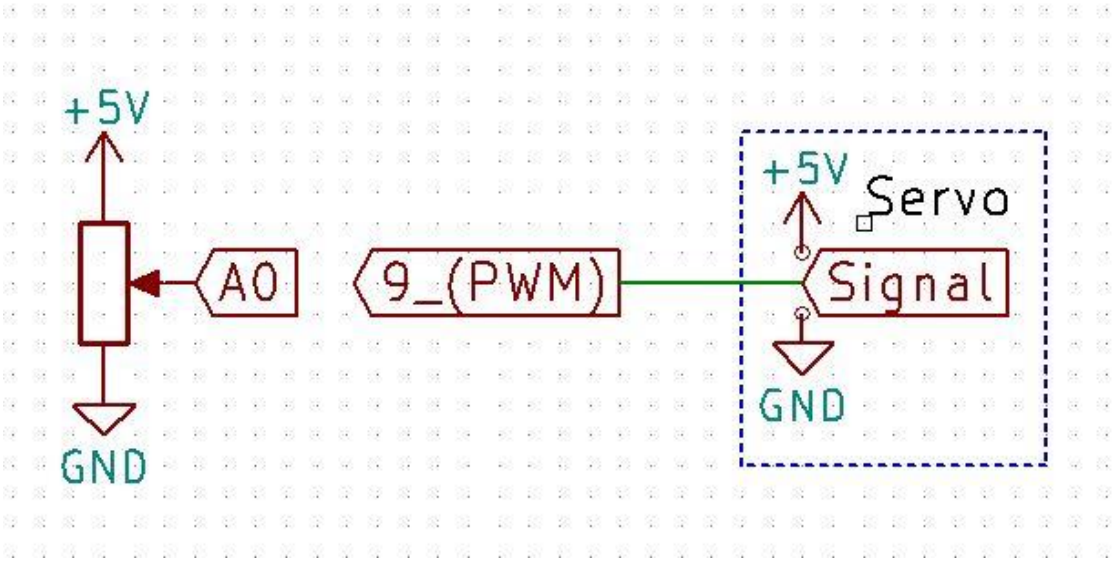
Kuva 22 PWM pulssinleveys minimissä (0,6ms)



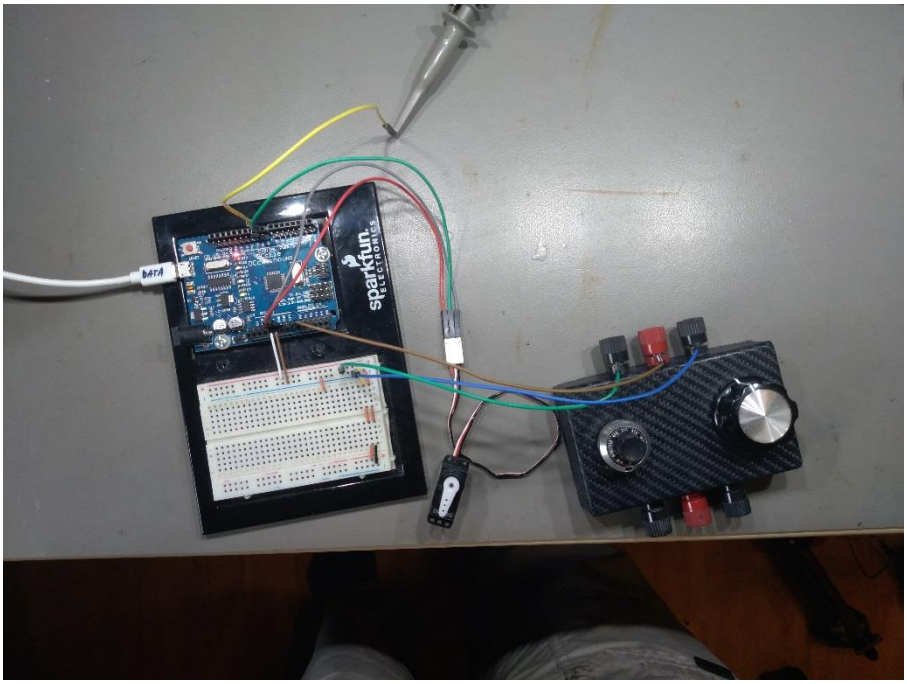
Kuva 23 PWM pulssinleveys maksimissa (2,2ms)

Kytkentäkaavio kuvassa 24 ja kuva kytkennästä kuvassa 25.

Kuten kuvista 24 ja 25 huomataan, että pulssinleveyden muutos kulmaan nähden on todella pieni (0,6ms–2,2ms=0–180 astetta)



Kuva 24 Servon kytkentäkaavio



Kuva 25 Servon kytkentäkuva

Ja ohjelmointikoodi kommentteineen:

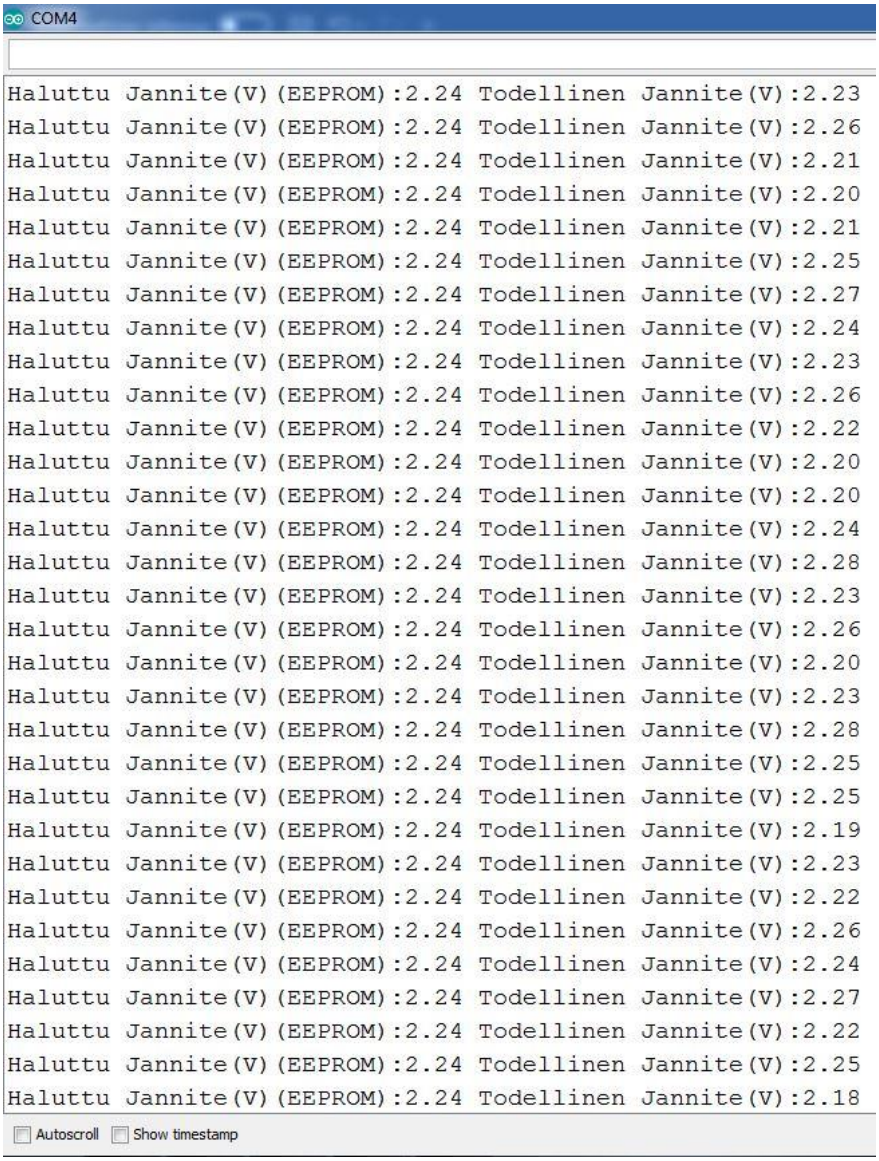
```
#include <Servo.h> //kirjasto, mikä hoitaa 50Hz taajuuden servon ohjaukseen

Servo smallServo;  //luodaan objekti ja nimetään se

int potentiometer = 0;  // nasta, mihin potentiometri kiinnittyy
int potentiometerValue;    //muuttuja potentiometrin arvoa varten

void setup() {
  smallServo.attach(9);  //valitaan PWM:ään kykenevä nasta arduinosta.
}

void loop()
{
  potentiometerValue = analogRead(potentiometer);           // luetaan poten-
tiometrin arvo
```

```
  potentiometerValue = map(potentiometerValue, 0, 1023, 0, 180);     // skaalataan arvo servolle käyttämällä map() komentoa.
  smallServo.write(potentiometerValue);                  // Anna käsky servolla
  delay(15);                          // koska ei ole takaisinkytkentää servolta Arduinolle, odotetaan pieni hetki
}
```

3.4 Jännitteen säätö PWM:n avulla

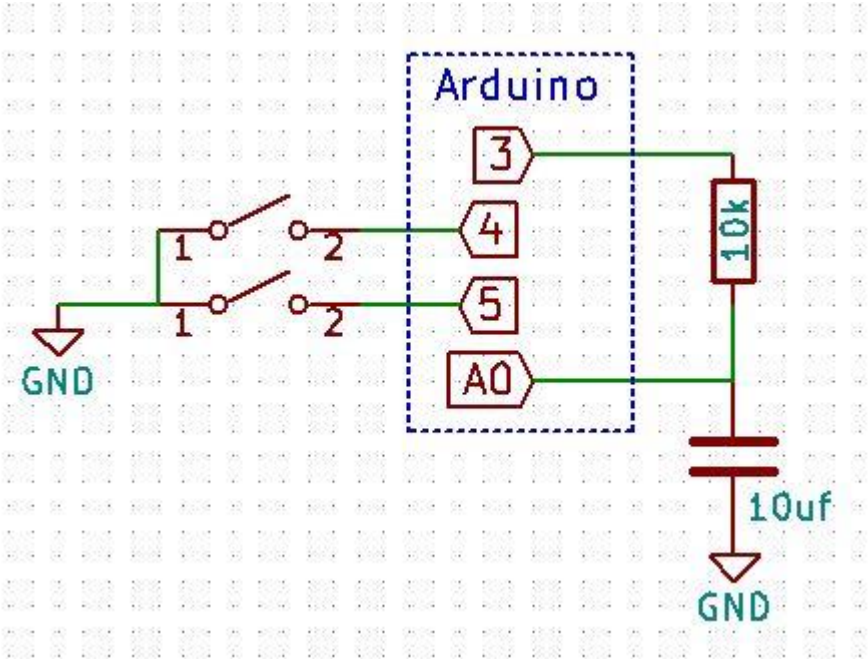
Työssä tutustutaan tiedon välitykseen sarjaväylää pitkin, tiedon muunnokseen, EEPROM:n käyttöön, sekä PWM:n ja ADC:n väliseen vertailuun takaisinkytkennällä.

Ensin valitaan kytkimellä, luetaanko haluttu jännitearvo sarjaväylää pitkin, vai muistista. Saraväylää pitkin jännite syötetään desimaalimuodossa, kuten 3.25 (kuva 26).

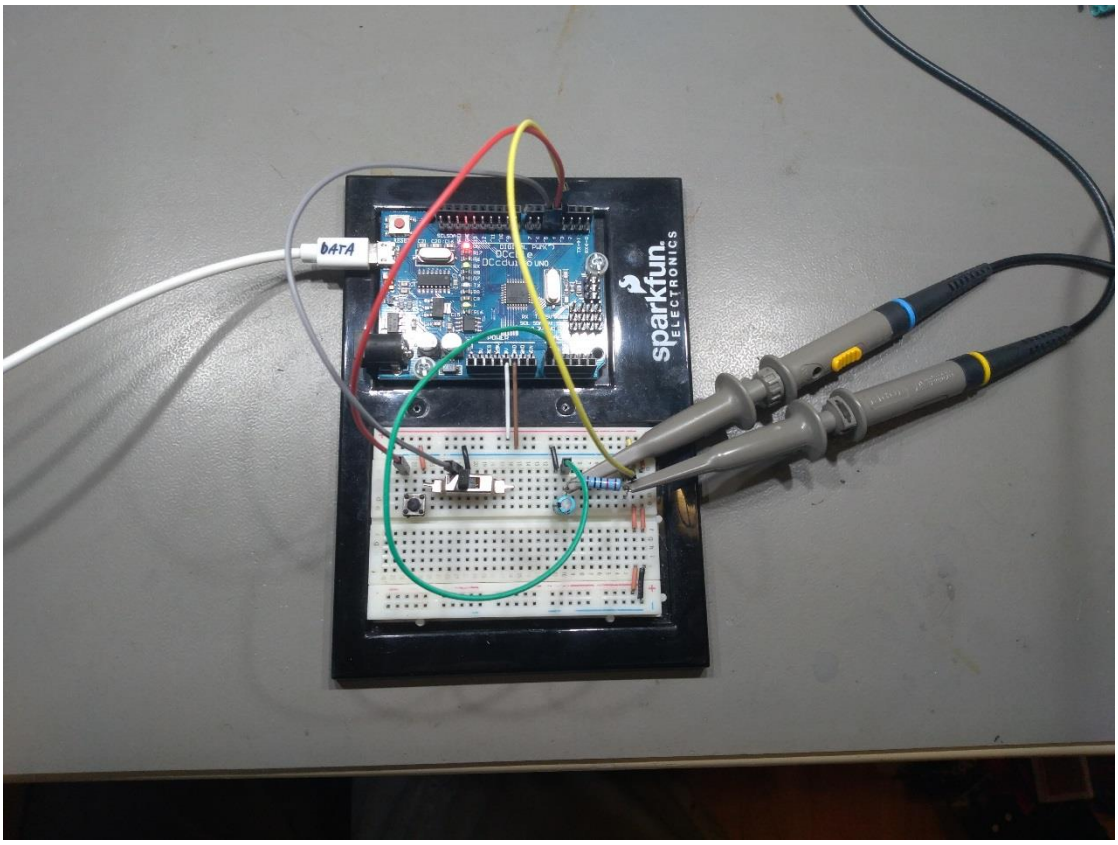


Kuva 26 Sarjaväylän näyttämä, kun haluttu jännite luetaan EEPROM:sta

Tätä haluttua jännitettä verrataan RC-suotimen kondensaattorin jännitteeseen, jonka jännite riippuu verrannollisesti PWM:n leveyden suhteessa. Kytkentä-kaavio kuvassa 27 ja kuva kytkennästä kuvassa 28.



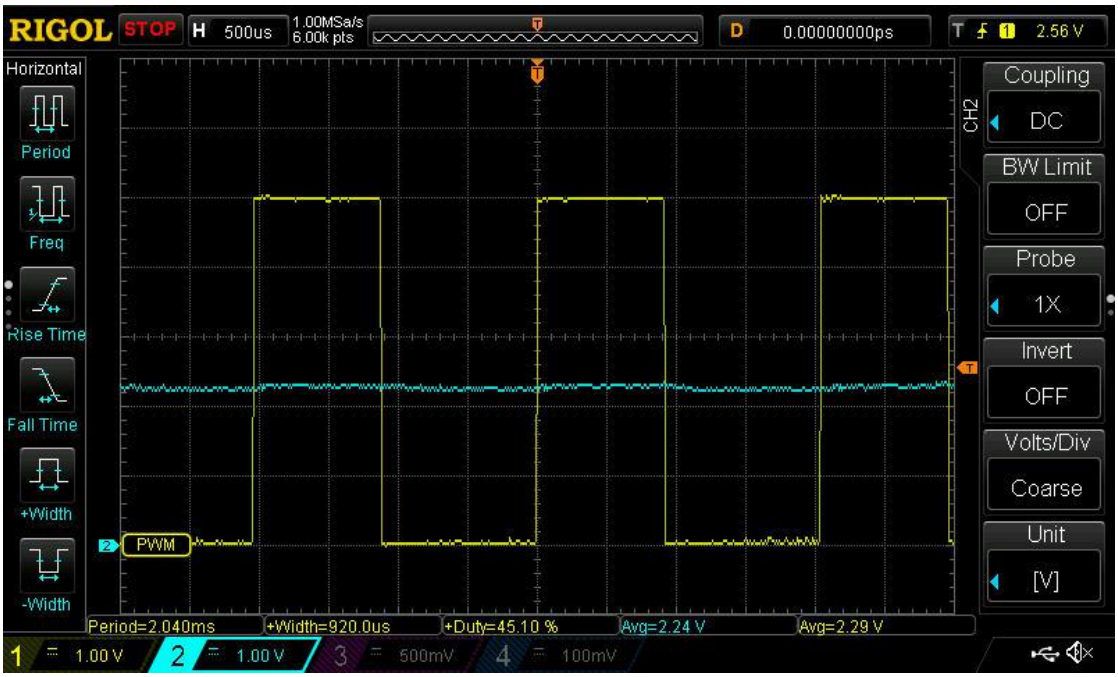
Kuva 27 Kytkentäkaavio jännitteensäädöstä PWM:n avulla



Kuva 28 Kuva kytkennästä

Jos vertailussa huomataan, että mitattu jännite on pienempi, kuin haluttu jännite, pulssinleveyttä kasvatetaan ja tarkistetaan jännite uudelleen hetken

päästä. Jos taas mitattu jännite on suurempi, kuin haluttu jännite, pulssinleveyttä lasketaan ja tarkistetaan jännite uudelleen hetken päästä. Pulssinleveys siis oskilloi hieman, kuten mitattu jännite, yrittäen pitää jännitteen vakiona. Kuvassa 29 nähdään PWM ja suodatettu jännite. Jos Kondensaattoria kuormitetaan, pulssinleveys kasvaa myös.



Kuva 29 Keltaisella PWM, sinisellä suodatettu jännite

Huomattavaa on, että PWM:n resoluutio on 8bit, kun taas ADC:n resoluutio on 10bit. Tästä johtuen säätötarkkuus on karkeampaa verrattuna lukutarkkuuteen.

Ohjelmistokoodi kommentteineen:

```
/*
  Lukee merkkijonon raakana datana sarjaväylää pitkin. (käytä pistettä desimaalina! Pilkku ei kelpaa)
  ->muuntaa muroluvuksi ja näyttää toivottuna jännitearvona. Syöttää PWM:llä RC siltaa, jonka jännite muuttuu aikavakion myötä.
  ->Laskee ADC:n kautta RC suotimen kondensaattorin jännitteen ja näyttää sen näytöllä.
  ->vertaa jänniteitä keskenään ja säätää PWM:n leveyden tarpeen mukaan, myös kuormituksen alaisena.
```

```
  ->voidaan tallentaa haluttu jännite EEPROM muistiin ja valita otetaanko ha-
luttu jännite sarjaväylää, vai EEPROM:n pitkin.
*/

#include <EEPROM.h>

char merkit[20];  //puskuri mihin tallennetaan tuleva data
char merkki; // erillinen muuttuja mihin tallennetaan välillä vertailtava merkki
byte laskuri = 0;
float merkitLukuina, luettuJannite, haluttuJannite;

float ylaRaja = 5.0;
float alaRaja = 0;

long previousMillis = 0;
int addr = 0;
int eepromArvo;
int PWMarvo;
void setup()
{
  Serial.begin(9600);
  pinMode(3, OUTPUT);       //PWM lähtö
  pinMode(4, INPUT_PULLUP); //EEPROM tallennus
  pinMode(5, INPUT_PULLUP); //EEPROM tai SERIAL valinta
}

void loop()
{
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis > 100)
  {
    previousMillis = currentMillis;
    if (digitalRead(5) == HIGH) //luetaanko sarjaväylää pitkin...
    {
      LueData(); //erillinen funktio, pitää itse loopin siistimpänä
      Serial.print("Haluttu Jannite (V)(SERIAL):");
```

```
    Serial.print(merkitLukuina);
    haluttuJannite=merkitLukuina;
  }
  if (digitalRead(5) == LOW) //vai muistista?
  {
    Serial.print("Haluttu Jannite(V)(EEPROM):");
    eepromArvo = EEPROM.read(addr);
    Serial.print(float(eepromArvo * 5.0 / 255.0)); //koska EEPROM ei säilytä NULL-terminaattoria, atof() funktio ei toimi
    haluttuJannite=eepromArvo * 5.0 / 255.0;
  }

  Serial.print('\t'); //tabulaattori erotin
  luettuJannite = float(analogRead(A0) * 5.0 / 1023.0); //luetaan kondensaattorin jännite ja lähetetään sarjaväylää pitkin
  Serial.print("Todellinen Jannite(V):");
  Serial.println(luettuJannite);

  if (luettuJannite <= haluttuJannite) //jos pienempi mitattu, kuin haluttu, nosta leveyttä
  {
    PWMarvo++;
  }

  if (luettuJannite > haluttuJannite) //jos suurempi mitattu, kuin haluttu, laske leveyttä
  {
    PWMarvo--;
  }

  analogWrite(3, constrain(PWMarvo, 0, 255)); //pidetään myös PWM rajojen sisällä constrain() funktiolla
 }
```

```
  if (digitalRead(4) == LOW && digitalRead(5) == HIGH) //alirutiini EEPROM tallennusta varten.
  {
    EEPROM.write(addr, merkitLukuina * 255 / 5);
    Serial.println("Tallennetaan...");
    for (byte i = 5; i > 0; i--)
    {
      delay(500);
      Serial.println(i);
    }
    Serial.println("Ok!");
    delay(1000);
  }



}

void LueData()
{
  if (Serial.available() > 0) //jos dataa on tulossa (luetaan RX-pinnistä)....
  {
    merkki = Serial.read(); //lue tuleva merkki
    if (merkki != '\n')   //jos merkki on JOKIN MUU kuin rivin lopetus....
    {
      merkit[laskuri++] = merkki;   //lisätään tuleva merkki puskuriin.
    }
    else //kun kerran lopetusmerkki saatiin, terminoidaan puskuri.
    {
      merkit[laskuri++] = '\0'; //terminointi hoidetaan lisäämällä null-merkki puskurin loppuun.
      laskuri = 0; //ja nollataan myös laskuri että uusi data alkaa puhtaalta pöydältä
      merkitLukuina = atof(merkit); //muutetaan merkkijono muuttujaksi
      if (merkitLukuina > ylaRaja) //tässä vaiheessa rajataan jännitearvot
      {
        merkitLukuina = ylaRaja;
```

```
    }
    if (merkitLukuina < alaRaja)
    {
      merkitLukuina = alaRaja;
    }
   }
  }

}
```

4 POHDINTA

Mikro-ohjaimia löytyy laajasti erilaisiin käyttötarkoituksiin suorituskyvystä ja fyysisistä ominaisuuksista riippuen. Pienemmät kykenevät hyvin helpommista tehtävistä, mutta muisti ja liitännät ovat puutteelliset. Uudemmat mallit kykenevät toimimaan jopa langattoman verkon yli. Uudemmat mallit myös käyttävät enemmän, yleistyvämpiä pienempiä käyttöjännitteitä kuten 3.3V. Myöskin langattomia järjestelmiä hyödynnetään enemmän uusissa järjestelmissä.

Esimerkkisovelluksista huomataan, että vaikkakin Arduino on ominaisuuksiltaan suppeampi verrattuna nykyaikaisempiin alustoihin, oppimisympäristönä on varteenotettava vaihtoehto.

LÄHTEET

1. Barragán, H. The untold history of Arduino. HTML–dokumentti. Saatavissa: https://arduinohistory.github.io/ [Viitattu 1.4.2020].

2. Atmel 8-bit AVR Microcontroller with 2/4/8K Bytes In-System Programmable Flash. PDF–dokumentti. Saatavissa:http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2586-AVR-8-bit-Microcontroller-ATtiny25-ATtiny45-ATtiny85_Datasheet.pdf [Viitattu 1.4.2020].

3. ATmega48A/PA/88A/PA/168A/PA/328/P. PDF–dokumentti. Saatavissa: http://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061A.pdf
[Viitattu 1.4.2020].

4. Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V. PDF–Dokumentti. Saatavissa: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf
[Viitattu 1.4.2020].

5. Who we are. HTML–dokumentti. Saatavissa: https://www.st.com/content/st_com/en/about/st_company_information/who-we-are.html
[Viitattu 1.4.2020].

6. PDF–dokumentti. STM32F103x8 produktion data. [Viitattu 1.4.2020].

7. K20 Sub-Family. PDF–dokumentti. Saatavissa: https://www.nxp.com/docs/en/data-sheet/K20P64M72SF1.pdf [Viitattu 1.4.2020].

8. Kinetis K66 Sub-Family. PDF–dokumentti. Saatavissa: https://www.nxp.com/docs/en/data-sheet/K66P144M180SF5V2.pdf [Viitattu 1.4.2020].

9. About Espressif. HTML–dokumentti. Saatavissa: https://www.espressif.com/en/company/about-us/who-we-are [Viitattu. 14.4.2020].

10. ESP8266 datasheet. PDF–dokumentti Saatavissa: https://www.itead.cc/wiki/images/8/8a/0a-esp8266ex_datasheet_en.pdf [Viitattu. 14.4.2020].

11. ESP32-S2 datasheet. PDF–dokumentti. Saatavissa: https://www.espressif.com/sites/default/files/documentation/esp32-s2_datasheet_en.pdf [Viitattu 14.4.2020].

Kuvaluettelo

TAULUKKOLUETTELO