

Ljubljana, Vegova 4

# APLIKACIJE IN INFORMACIJSKI SISTEMI

PAMETNA HIŠA – IMPLEMENTACIJA SISTEMA

Mentor: Marko Kastelic

Avtor: Jakob Hostnik, G 4. A

Ljubljana, april 2017

Ljubljana, Vegova 4

## Zahvala

Zahvaljujem se mentorju Marku Kastelicu za pomoč in svetovanje. Zahvaljujem pa se tudi Anžetu Žaberlu za pomoč pri izdelavi vezij in možnost uporabe CNC stroja.

## Povzetek in ključne besede

Seminarska naloga temelji na implementaciji pametne hiše. Predstavljene so tehnologije s katerimi sem se srečeval, ovire, ki sem jih moral premagati, opisani so okvirni postopki izdelave, analiza potreb in zmožnosti implementacije. Postavil sem strežnik, implementiral potrebne protokole, oblikoval aplikacijo in izdelal nekaj testnih pripomočkov - elementov pametne hiše.

Ključne besede: pametna hiša, mikrokrmilnik, protokol MQTT, mikrokrmilnik ESP8266, arduino, posrednik, aplikacija OpenHAB, mikroračunalnik Raspberry Pi

## Summary and keywords

This seminar work is based around the implementation of the smart house. In it, are the representations of all the technologies with whom I met while I was designing it, obstacles that I have overcome, descriptions of the expected manufacturing processes and analysis of the needs in the capabilities of the implementation. I set up the server and implemented all the necessary protocols, configured application and made multiple widgets, essentially the elements of the smart house

Keywords: smart house, microcontroller, MQTT protocol, ESP8266 microcontroller, arduino, broker, OpenHAB application, microcomputer Raspberry PI

## Seznam kratic in okrajšav

Ang. – Angleško

3D – Tridimenzionalno (Ang. Three Dimensional)

AT – Ukazne zahteve.

ESP – Espressif (Okrajšava za podjetje, ki proizvaja mikrokrmilnike)

GPIO – Splošne vhodno izhodno-izhodni pini (Ang. General-purpose input / output)

HTTP – Protokol za prenos hiperpovezav - spletnih strani (Ang. Hyper Text Transfer Protocol)

HTTPS – Varni protokol za prenos hiperpovezav - spletnih strani (Ang. Secure Hyper Text Transfer Protocol)

IDE – Integrirano razvojno okolje (Ang. Integrated Development Environment)

KNX – Okrajšava za Konnex

MQTT – Telemetričen promet v vrsti čakajočih sporočil (Ang. Message Queuing Telemetry Transport)

OASIS – Organizacija za napredek strukturiranih informacijskih standardov (Ang. Organization for the Advancement of Structured Information Standards)

SD – Spominska kartica majhnega formata (Ang. Secure Digital)

USB – Univerzalno serijsko vodilo (Ang. Universal Serial Bus)

WiFi – Brežžiče povezave (Ang. Wireless Fidelity)

## Kazalo vsebine

<b>1</b>	<b>UVOD</b>	<b>9</b>
<b>2</b>	<b>UPORABLJENE TEHNOLOGIJE IN SREDSTVA</b>	<b>10</b>
<b>2.1</b>	<b>PROTOKOL MQTT</b>	<b>10</b>
2.1.1	OPIS	10
2.1.2	UPORABA	10
<b>2.2</b>	<b>PROTOKOL KNX</b>	<b>12</b>
2.2.1	OPIS	12
<b>2.3</b>	<b>ARDUINO</b>	<b>12</b>
2.3.1	OPIS	12
2.3.2	UPORABA	13
<b>2.4</b>	<b>ESP8266</b>	<b>13</b>
2.4.1	OPIS	13
2.4.2	UPORABA	13
2.4.3	ESP – 01	14
2.4.4	ESP – 12F	16
<b>2.5</b>	<b>RASPBERRY PI 3</b>	<b>17</b>
2.5.1	OPIS	17
<b>2.6</b>	<b>OPERACIJSKI SISTEMI PODPRTI NA RASPBERRY STROJNI PLATFORMI</b>	<b>18</b>
2.6.1	RASPBIAN	18
2.6.2	WINDOWS 10 IOT CORE	18
2.6.3	ARCH LINUX	18
<b>2.7</b>	<b>OPENHAB 2</b>	<b>18</b>
2.7.1	OPIS	18
2.7.2	DELOVANJE	19
<b>3</b>	<b>REALIZACIJA KOMPONENT IN INTEGRACIJA V ZAKLJUČEN SISTEM</b>	<b>22</b>
<b>3.1</b>	<b>ANALIZA ZMOŽNOSTI IN OPREDELITEV ODLOČITEV</b>	<b>22</b>
3.1.1	KNX IN MQTT	22

**Ljubljana, Vegova 4**

3.1.2	ARDUINO IN ESP8266	22
3.1.3	OPERACIJSKI SISTEM NA RASPBERRY PI	22
<b>3.2</b>	<b>REALIZACIJA POSAMEZNIH KOMPONENT SISTEMA</b>	<b>23</b>
3.2.1	STREŽNIK – POSREDNIK (ANG. BROKER)	23
3.2.2	STIKALO NA DALJAVO	24
3.2.3	ALARMNI SISTEM	27
<b>4</b>	<b>SKLEPNE UGOTOVITVE</b>	<b>29</b>
<b>5</b>	<b>LITERATURA IN VIRI</b>	<b>30</b>
<b>6</b>	<b>PRILOGE</b>	<b>32</b>
<b>7</b>	<b>IZJAVA O AVTORSTVU</b>	<b>41</b>

## Kazalo slik

<i>Slika 1: Prikaz uporabe MQTT protokola</i> .....	11
<i>Slika 2: Mikrokrmilnik ESP8266 - ESP01 (Vir: <a href="http://www.ebay.com/cln/kenneyrsw-lk/NodeMcu-Lua/336792467017">http://www.ebay.com/cln/kenneyrsw-lk/NodeMcu-Lua/336792467017</a>)</i> .....	14
<i>Slika 3: Napetostni pretvornik signala (Vir: <a href="https://www.amazon.com/Esp8266-Wireless-Transceiver-Arduino-IFANCY-TECH/dp/B01J94UWNC">https://www.amazon.com/Esp8266-Wireless-Transceiver-Arduino-IFANCY-TECH/dp/B01J94UWNC</a>)</i> .....	14
<i>Slika 4: ESP8266 - ESP01 v pretvorniku v USB signal</i> .....	14
<i>Slika 5: Razporeditev pinov (Vir: <a href="http://stafferier.com/?p=107">http://stafferier.com/?p=107</a>)</i> .....	15
<i>Slika 6: ESP8266 - ESP01 v Pretvorniku v USB signal z stikalom za preklon v način za nalaganje programa</i> .....	15
<i>Slika 7: Izбира nastavitve za nalaganje programa</i> .....	15
<i>Slika 8: NodeMCU Lua ploščica z ESP - 12F (Vir: <a href="http://www.ebay.com/itm/NodeMCU-Lua-Wireless-Entwicklungs-Board-ESP8266-WiFi-Modul-ESP-12F-USB-/272282652517?hash=item3f654f8f65:g:CzgAAOSwGIRXaswq">http://www.ebay.com/itm/NodeMCU-Lua-Wireless-Entwicklungs-Board-ESP8266-WiFi-Modul-ESP-12F-USB-/272282652517?hash=item3f654f8f65:g:CzgAAOSwGIRXaswq</a>)</i> .....	16
<i>Slika 9: Razporeditev pinov na ploščici NODEMCU, ki nam omogoča lažjo izdelavo prototipnih izdelkov z uporabo ESP – 12F krmilnika (Vir: <a href="https://gist.github.com/eriknomitch/a904ad1d806c349970f553af3e93a0e3">https://gist.github.com/eriknomitch/a904ad1d806c349970f553af3e93a0e3</a>)</i> .....	17
<i>Slika 10: Raspberry Pi 3 (vir: <a href="https://shop.pimoroni.com/products/raspberry-pi-3">https://shop.pimoroni.com/products/raspberry-pi-3</a>)</i> .....	17
<i>Slika 11: OpenHAB - predstaviteni model (Vir: <a href="https://mcuoneclipse.com/2015/12/23/installing-openhab-home-automation-on-raspberry-pi/">https://mcuoneclipse.com/2015/12/23/installing-openhab-home-automation-on-raspberry-pi/</a>)</i> .....	19
<i>Slika 12: Zgled datoteke kazala</i> .....	20
<i>Slika 13: Zgled datoteke predmetov</i> .....	20
<i>Slika 14: OpenHAB aplikacija - prikaz skupine</i> .....	24
<i>Slika 15: OpenHAB aplikacija - stanja senzorja gibanja</i> .....	24
<i>Slika 16: Stikalo na daljavo - pripravljena slika vezja za CNC stroj</i> .....	25
<i>Slika 17: Stikalo na daljavo - 3D slika vezja v programu Target 3000 (spodnja stran)</i> .....	25
<i>Slika 18: Testiranje vezja</i> .....	26
<i>Slika 19: Stikalo na daljavo - 3D slika vezja v programu Target 3000 (spodnja stran)</i> .....	26
<i>Slika 20: Stikalo na daljavo - shema vezja</i> .....	27
<i>Slika 21: Prikaz delovanja alarmnega sistema</i> .....	27
<i>Slika 22: Senzor gibanja - delujoči prototip izdelka</i> .....	28
<i>Slika 23: Senzor gibanja (Vir: <a href="http://www.ebay.com/itm/HC-SR501-Infrared-PIR-Motion-Sensor-Module-for-Arduino-Raspberry-pi-new-/272600006597?hash=item3f7839ffc5">http://www.ebay.com/itm/HC-SR501-Infrared-PIR-Motion-Sensor-Module-for-Arduino-Raspberry-pi-new-/272600006597?hash=item3f7839ffc5</a>)</i> .....	28
<i>Slika 24: Alarm med gibanjem</i> .....	29

Ljubljana, Vegova 4

## Kazalo prilog

<i>Priloga 1: Koda alarma .....</i>	<i>32</i>
<i>Priloga 2: Koda senzorja gibanja .....</i>	<i>34</i>
<i>Priloga 3: ESP8266 – ESP01 koda za stikalo na daljavo .....</i>	<i>36</i>
<i>Priloga 4: Arduino koda za stikalo na daljavo .....</i>	<i>38</i>
<i>Priloga 5: Tabela primerjave mikrokontrolerov Arduino in ESP8266 .....</i>	<i>39</i>
<i>Priloga 6: Primerjava Arduino mikrokontrolerov (Vir: <a href="https://learn.adafruit.com/assets/6122">https://learn.adafruit.com/assets/6122</a>) .....</i>	<i>40</i>



## 1 Uvod

Pametne hiše so zadnjih nekaj let pravi hit in vsak si jo želi imeti. Pametne hiše se gradi na dva načina. Prvi način je iz temeljev navzgor, kar pomeni da se zgradi celotno hišo z električno in vodovodno napeljavo prilagojeno pametnemu upravljanju. Ker je tak način gradnje je običajno zelo drag, in ker si vsakdo ne želi graditi nove hiše ali pa bi raje živel v bloku, se čedalje pogosteje pojavlja alternativa, ki prilagodi naprave za vgradnjo v obstoječe hiše in ne obratno.

### Cilji

Cilj naloge je izdelati različne pripomočke in dodatke, vzpostaviti sistem za njihovo upravljanje in povezovanje, analizirati strojne platforme za izdelavo pametnih naprav (omejitev na meni cenovno dostopne) in preučiti protokole za izmenjavo podatkov. Uporabljene tehnologije in posamezni cilji so opredeljeni v naslednjih poglavjih.

Postavil sem strežnik, ki mi omogoča izmenjavo podatkov med napravami, izdelal pametno stikalo, ki mi doma prižiga in ugaša luč in alarmni sistem, ki je sestavljen iz dveh delov (zvočnika in senzorja gibanja). Vse skupaj pa lahko nadziram iz lokalne spletne strani ali preko mobilne aplikacije. Za upravljanje in nadzor sem uporabil že obstoječi sistem OpenHAB 2, ki vključuje nadzor pametne hiše, mobilno aplikacijo in spletno stran.

## 2 Uporabljene tehnologije in sredstva

V tem poglavju predstavim tehnologije in sredstva med katerimi sem se odločal za realizacijo. Opisana sta protokola za povezovanje naprav MQTT in KNX, strojne platforme Arduino ESP8266, Raspberry PI, operacijski sistemi za Raspberry PI in aplikacija za upravljanje pametne hiše OpenHAB.

### 2.1 Protokol MQTT

#### 2.1.1 Opis

MQTT je protokol namenjen komunikaciji med pametnimi napravami (Ang. machine-to-machine). Uporablja se za povezovanje naprav preko strežnika T.I. posrednika (Ang. broker). Od 7. novembra 2014 je tudi OASIS<sup>1</sup> standard. Med drugimi ga uporablja tudi podjetje Facebook, Inc. v svoji aplikaciji Messenger.

#### 2.1.2 Uporaba

Naprave povezane v MQTT omrežje lahko zahtevajo prejem podatkov iz določene kategorije lahko pa jih tudi pošiljajo v kategorijo. Vsi podatki se pošiljajo preko strežnika, tako imenovanega posrednika.

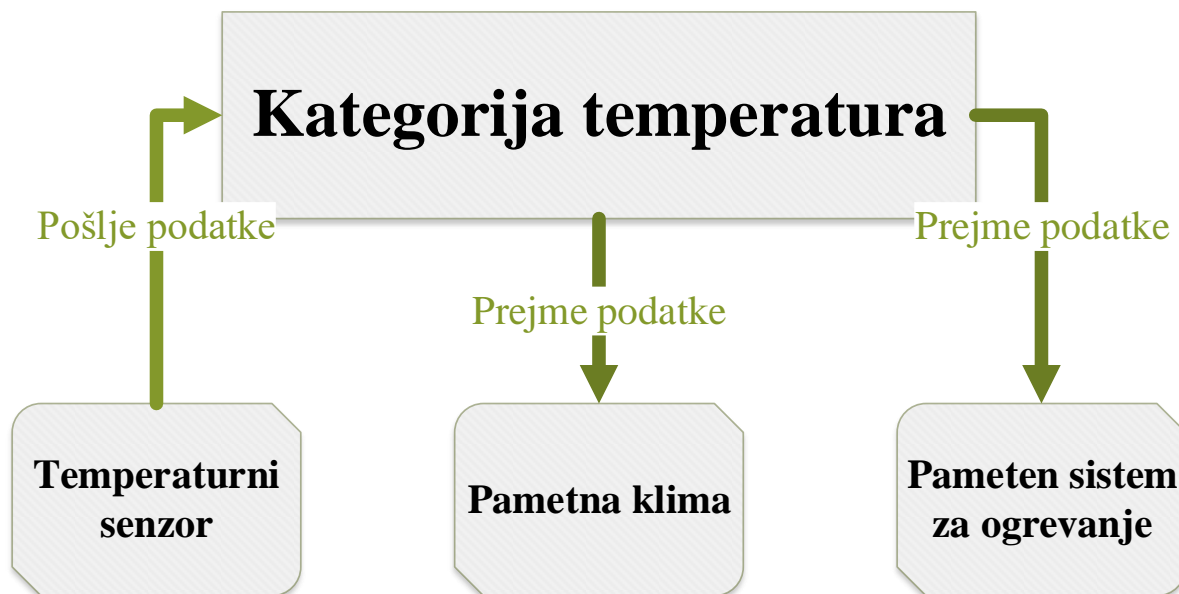
Na primer: Senzor za temperaturo pošilja izmerjeno temperaturo v skupino »Temperatura«. Namerjene podatke pa potem zahteva naša klima.

---

<sup>1</sup> Organizacija za napredek strukturiranih informacijskih standardov (Ang. Organization for the Advancement of Structured Information Standards)

Ljubljana, Vegova 4

SLIKA 1: PRIKAZ UPORABE MQTT PROTOKOLA



Strežniška (posredniška) aplikacija, ki omogoča MQTT v linux operacijskem sistemu se imenuje Mosquitto in jo na namestimo z ukazom `apt-get install mosquitto`. Da pa potem zaženemo mqtt strežnik pa zaženemo ukaz `mosquitto`.

Primeri pošiljanja podatkov v kategorijo s programom Mosquitto:

```
mosquitto_pub -d -t kategorija -m "naše sporočilo"
```

Primer naročanja na podatke iz kategorije:

```
mosquitto_sub -d -t kategorija
```

*Prejeti podatki se bodo izpisali v terminalu.*

Sam sem za testiranje delovanja strežnika uporabil tudi brezplačno Android aplikacijo MyMQTT.

Ljubljana, Vegova 4

## 2.2 Protokol KNX

### 2.2.1 Opis

Protokol KNX je standard za prenos podatkov, ki temelji na OSI modelu<sup>2</sup>. KNX opredeljuje več fizičnih komunikacijskih prenosov podatkov:

- Sukana parica
- Radijo
- Infrardeči vmesnik
- Ethernet
- Powerline

Protokol KNX je zasnovan tako, da je neodvisen od katerekoli strojne platforme.

KNX naprava ima implementirane 3 načine.

**A-način** ali avtomatični način zahteva od naprave da se namesti samodejno.

**E-način** ali preprost način od nas zahteva nekaj preprostega znanja da namestimo napravo.

**S-način** ali sistemski način pa je uporabljen pri avtomatizaciji stavb po meri. Vedenje naprave je potrebno programirati.

## 2.3 Arduino

### 2.3.1 Opis

Arduino je odprtokodna platforma, ki je sestavljena iz programskega in strojnega okolja, ki nam omogoča izdelavo interaktivnih projektov.

Strojno okolje Arduino je Arduino plošča, mikrokontroler ki nam omogoča izvajanje ukazov in upravljanja elektronskih komponent. Obstaja več različnih Arduino plošč, ki jih izdeluje podjetje Arduino AG. Ker pa je projekt odprt za uporabo in izboljšave vsem, se z izdelavo mikrokontrolerov ukvarja tudi mnogo manjših podjetij. Arduino programsko okolje pa je orodje za programiranje, nalaganje kode in upravljanje Arduino plošče.

---

<sup>2</sup> OSI model – Odprt model sistema za medsebojno povezavo (Ang. Open Systems Interconnection model)

Ljubljana, Vegova 4

### 2.3.2 Uporaba

Arduino se uporablja za upravljanje strojnih komponent. Omogoča serijsko komunikacijo z računalnikom ali drugim mikrokrmilnikom. V našem primeru preko mikrokrmilnika Arduino vklapljam in izklapljam luč z servomotorjem, ki fizično pritisne na stikalo.

## 2.4 ESP8266

### 2.4.1 Opis

ESP8266 je mikrokrmilnik, ki ima že vgrajeno WiFi anteno. Programiramo ga lahko v okolju Arduino IDE s predhodno namestitvijo ploščice.

#### 2.4.1.1 Priprava okolja Arduino IDE za programiranje ESP8266 (Vir: <https://github.com/esp8266/Arduino>)

1. Namestimo program Arduino IDE iz [Arduino strani](#).
2. Zaženemo Arduino IDE in odpremo Nastavitve/Preferences (ctrl + vejica)
3. Vnesemo `http://arduino.esp8266.com/stable/package_esp8266com_index.json` v *Additional Board Manager URLs* polje.
4. Odpremo Board Manager (Orodja->plošča->Boards Manager)
5. Iščemo ESP8266 in namestimo edino ponujeno ploščico.

### 2.4.2 Uporaba

ESP8266 moduli se uporabljajo predvsem v internetu stvari (IOT-Internet Of Things), pametnih hišah in centralnih nadzornih sistemov. Saj ne potrebujemo nobenega dodatka za brezžično povezovanje mikrokrmilnikov med seboj, kot na primer pri Arduinu, kjer potrebujemo WiFi oklop (Ang. shield).

Ljubljana, Vegova 4

### 2.4.3 ESP – 01

SLIKA 2: MIKROKRMLNIK ESP8266 - ESP01 (VIR: [HTTP://WWW.EBAY.COM/CLN/KENNEYRSW-LK/NODEMCU-LUA/336792467017](http://www.ebay.com/CLN/KENNEYRSW-LK/NODEMCU-LUA/336792467017))



ESP – 01 se večinoma uporablja v kombinaciji z Arduino ploščico kot Arduino WiFi modul s serijsko komunikacijo, za kar pa potrebujemo pretvornik iz 5 voltnega signala v 3,3 voltni signal. Saj Arduino mikrokrmilnik deluje na napetosti 5 voltov, medtem ko ESP8266 deluje na 3,3 voltih.

SLIKA 3: NAPETOSTNI PRETVORNIK SIGNALA (VIR: [HTTPS://WWW.AMAZON.COM/Esp8266-Wireless-Transceiver-Arduino-IFANCY-TECH/dp/B01J94UWNC](https://www.amazon.com/Esp8266-Wireless-Transceiver-Arduino-IFANCY-TECH/dp/B01J94UWNC))



Za komunikacijo med ESP8266 in računalnikom, vključno z nalaganjem programa potrebujemo pretvornik iz serijskega v USB signal. Ti dve ploščici nam omogočata serijsko povezavo med računalnikom in mikrokrmilnikom, ki jo lahko kar direktno preizkusimo, tako da preko Arduino serijskega ekrana pošiljamo AT ukazi.

SLIKA 4: ESP8266 - ESP01 V PRETVORNIKU V USB SIGNAL

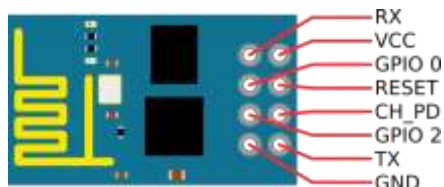


Program pa naložimo tako, da ESP8266 preklapimo v način za nalaganje programa (Ang. Flash mode). Zato moramo povezati pina GPIO 0 in GND. Vendar moramo paziti, da lahko pina po

## Ljubljana, Vegova 4

nalaganju programa zopet razklenemo. Jaz sem prispajkal stikalo. Nato pa vstavimo USB pretvornik z ESP8266 krmilnikom v računalnik.

SLIKA 5: RAZPOREDITEV PINOV (VIR: [HTTP://STAFFERIER.COM/?P=107](http://STAFFERIER.COM/?P=107))



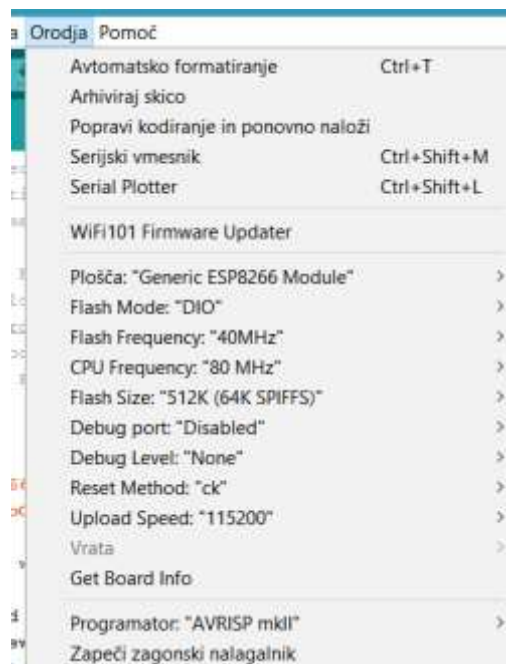
SLIKA 6: ESP8266 - ESP01 V PRETVORNIKU V USB SIGNAL Z STIKALOM ZA PREKLOP V NAČIN ZA NALAGANJE PROGRAMA



V večini primerov je na računalnik potrebno namestiti gonilnik dostopen na strani: <http://sparks.gogo.co.nz/ch340.html>

Odpremo Arduino IDE izberemo port in nastavimo ploščico na Generic ESP8266 module.

SLIKA 7: IZBIRA NASTAVITEV ZA NALAGANJE PROGRAM



Nato lahko med primeri namestimo testni program. (Examples->ESP8266->Blink).

## Ljubljana, Vegova 4

Če želimo pisati program, ki bo uporabljal MQTT protokol, moramo namestiti knjižnico PubSubClient (Dostopno na: <https://github.com/knolleary/pubsubclient>), ki ima tudi nekaj primerov za programiranje krmilnikov ESP8266.

#### 2.4.4 ESP – 12F

ESP – 12F je od ploščic Arduino veliko zmogljivejši, cene pa so primerljive. Zato lahko kupimo samo ESP – 12F, ki ga bomo uporabljali pri dejanskih izdelkih. Za izdelavo prototipov pa je bolje uporabiti celo ploščico s pini in USB priklopom. Nato pa jo uporabljamo na enak način kot ploščico Arduino, le v Arduino IDE nastavimo ploščico na ESPino (ESP-12 module).

*SLIKA 8: NODEMCU LUA PLOŠČICA Z ESP - 12F (VIR: [HTTP://WWW.EBAY.COM/ITM/NODEMCU-LUA-WIRELESS-ENTWICKLUNGS-BOARD-ESP8266-WIFI-MODUL-ESP-12F-USB-/272282652517?hash=item3f654f8f65:g:CZGAAOSwGIRXaswQ](http://www.ebay.com/itm/NODEMCU-LUA-WIRELESS-ENTWICKLUNGS-BOARD-ESP8266-WIFI-MODUL-ESP-12F-USB-/272282652517?hash=item3f654f8f65:g:CZGAAOSwGIRXaswQ))*

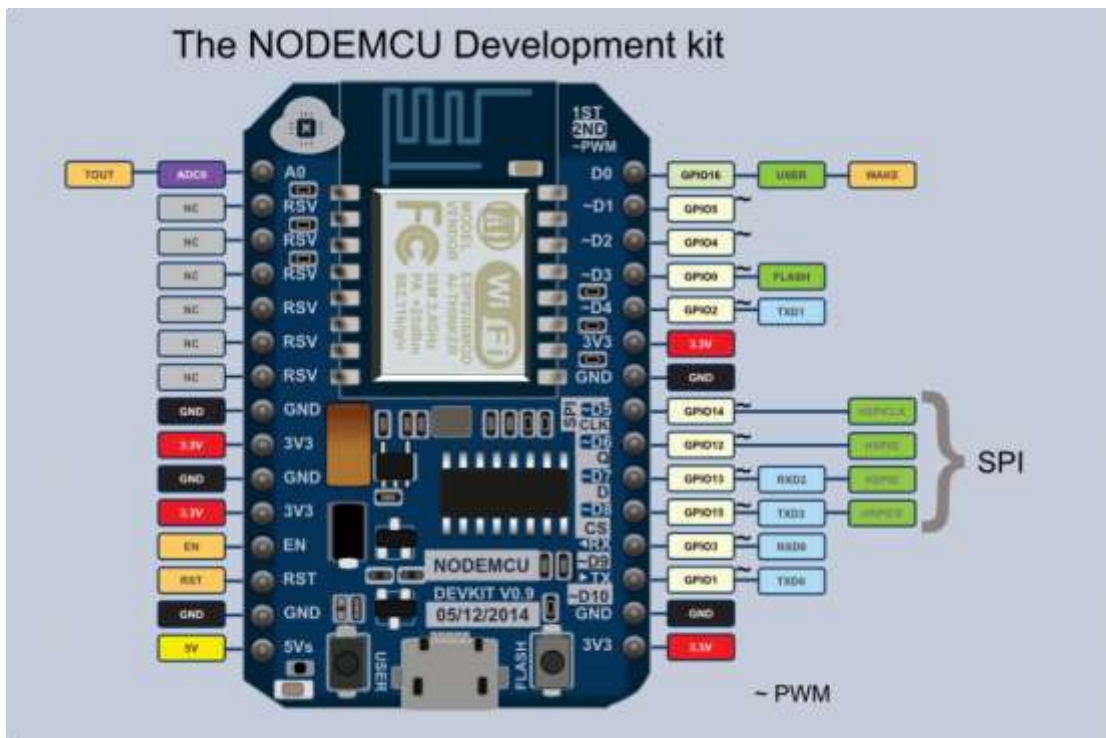


Pred nalaganjem programa pa moramo paziti, da na ploščici tiščimo flash gumb in jo hkrati resetiramo – pritisnemo reset gumb. Nato lahko namestimo program.



## Ljubljana, Vegova 4

SLIKA 9: RAZPOREDITEV PINOV NA PLOŠČICI NODEMCU, KI NAM OMOGOČA LAŽJO IZDELAVO PROTOTIPNIH IZDELKOV Z UPORABO ESP – 12F KRMILNIKA (VIR: [HTTPS://GIST.GITHUB.COM/ERIKNOMITCH/A904AD1D806C349970F553AF3E93A0E3](https://gist.github.com/eriknomitch/a904ad1d806c349970f553af3e93a0e3))



## 2.5 Raspberry Pi 3

### 2.5.1 Opis

Raspberry Pi je mikroračunalnik, ki je bil izdelan z namenom spodbujanja učenja osnov računalništva. Pogosto se ga uporablja v manjše testne strežniške namene, nam pa z dodatnimi pini omogoča tudi krmiljenje elektronskih komponent (motorjev, lučk ...). Raspberry Pi 3 se od ostalih modelov razlikuje po tem, da je zmogljivejši in ima že vgrajeno Bluetooth in WiFi anteno.

SLIKA 10: RASPBERRY PI 3 (VIR: [HTTPS://SHOP.PIMORONI.COM/PRODUCTS/RASPBERRY-PI-3](https://shop.pimoroni.com/products/raspberry-pi-3))



## 2.6 Operacijski sistemi podprti na Raspberry strojni platformi

Raspberry ploščica podpira več operacijskih sistemov. Operacijski sistem se zapeče na SD kartico, ki se jo potem vstavi v Raspberry PI. Najpogostejši so:

- Raspbian
- Windows 10 IOT Core
- Arch linux

### 2.6.1 Raspbian

Je uradno podprt sistem na Raspberry PI platformi. Temelji na najpopularnejši linux distribuciji debian. Raspbian že ima prednameščeno javansko platformo in nekaj drugih uporabnih orodij. Ker je raspbian tako kot debian zelo popularna distribucija je odpravljanje težav toliko lažje, saj zagotovo obstaja nekdo, ki je že imel podoben problem.

### 2.6.2 Windows 10 IOT Core

Windows 10 IOT Core je verzija sistema Windows 10, ki je optimizirana za manjše naprave z ali brez ekrana. Omogoča lažjo povezavo naprav z sistemom Windows 10 in upravljanje pinov.

### 2.6.3 Arch linux

Je linux sistem, ki pride z minimalno prednameščenimi programi. Namestitev sistema je ročna v terminalu. Zaradi majhnega števila prednameščenih programov in programov v ozadju je sistem zelo odziven in dobro deluje tudi na nekoliko manj zmogljivih računalnikih, kot je Raspberry Pi.

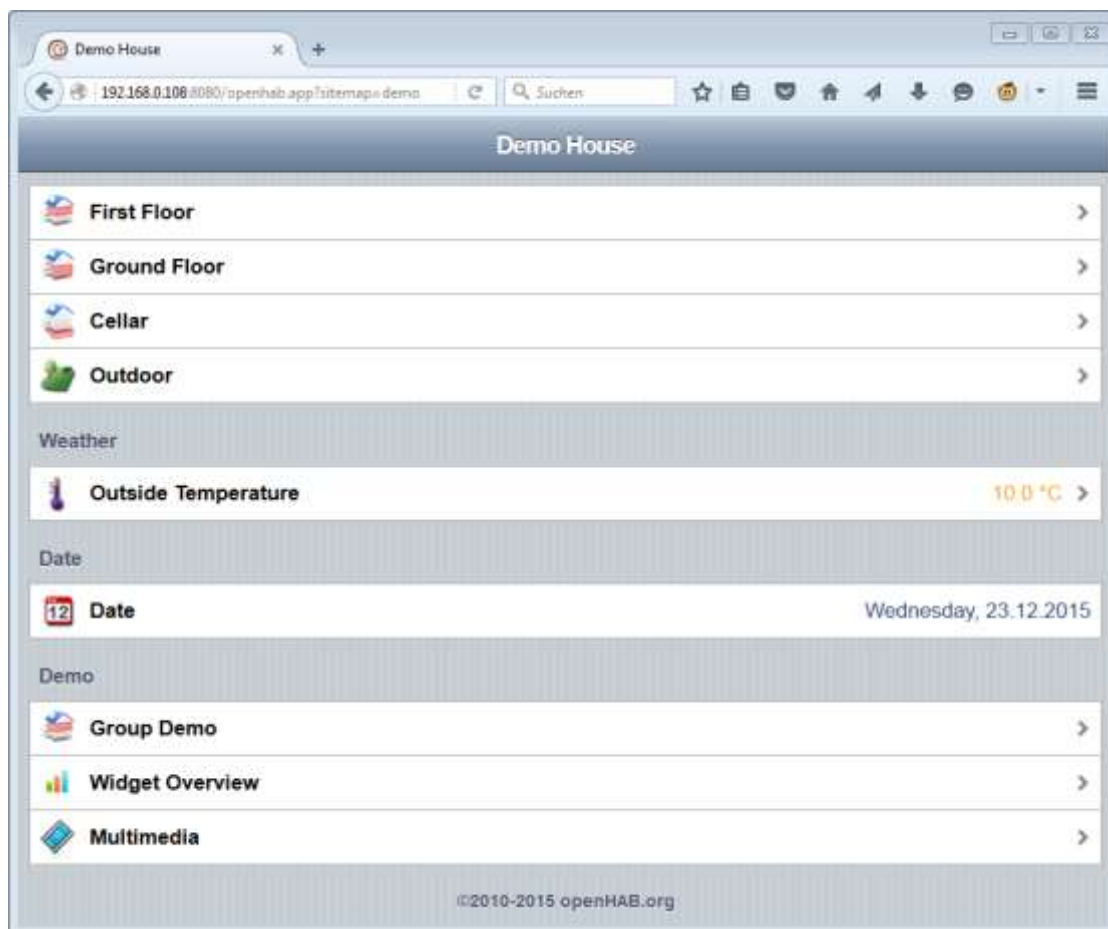
## 2.7 OpenHAB 2

### 2.7.1 Opis

OpenHAB je odprtokodna avtomatizacijska programska rešitev. Razvita je v javanskem programskem jeziku z uporabo Eclipse SmartHome programskega ogrodja. Omogoča nam nadzor nad napravami pametne hiše ali drugega sistema.

## Ljubljana, Vegova 4

SLIKA 11: OPENHAB - PREDSTAVITVENI MODEL (Vir: [HTTPS://MCUONECLIPSE.COM/2015/12/23/INSTALLING-OPENHAB-HOME-AUTOMATION-ON-RASPBERRY-PI/](https://mcuoneclipse.com/2015/12/23/installing-openhab-home-automation-on-raspberry-pi/))



### 2.7.2 Delovanje

Da lahko namestimo OpenHAB na strežnik moramo najprej poznati nekaj osnov o samem programu. Znanje črpamo iz dokumentacije dostopne na <http://docs.openhab.org/>. Vsak dom ima drugačno zgradbo, drugačno razporeditev naprav in drugačno pomembnost posamezne naprave. Zgradbo našega sistema določamo v kazalu (Ang. sitemap), datoteki s končnico .sitemap, definicijo naprav pa v datoteki z končnico .items. Da lahko namestimo aplikacijo, moramo imeti na strežniku nameščeno platformo java, potem pa prekopiramo datoteke aplikacije in jo zaženemo.

#### 2.7.2.1 Kazala (Ang. Sitemaps)

V datoteki s končnico .sitemap napišemo kazalo, ki ustreza naši zgradbi in strukturi. Začnemo z definicijo kazala, saj imamo lahko več različnih kazal, z različnimi vsebinami. In potem

## Ljubljana, Vegova 4

vsakič med zavite oklepaje pišemo vsebino. Elementi kot so frame (okvir) in group (skupina) pa imata lahko tudi podelemente znotraj svojih zavitih oklepajev.

SLIKA 12: ZGLED DATOTEKE KAZALA

```
sitemap demo label="My home automation" {
  Frame label="Date" {
    Text item=Date
  }
  Frame label="Demo" {
    Group item=Heating
    Switch item=Lights icon="big_bulb" mappings=[OFF="All Off"]
    Text item=Temperature valuecolor=[>25="orange",>15="green",<=15="blue"]
    Text item=Multimedia_Summary label="Multimedia" icon="video" {
      Selection item=TV_Channel mappings=[0="off", 1="DasErste", 2="BBC One", 3="Cartoon Network"]
      Slider item=Volume
    }
  }
}
```

### 2.7.2.2 Predmeti(Ang. Items)

Predmeti v aplikaciji OpenHAB so predmeti, ki se bodo prikazali na uporabniškem vmesniku. Se pravi stikala, oznake in drugi elementi, ki nam omogočajo upravljanje pametnega doma preko spletne strani ali mobilne aplikacije. Predmete vpisujemo v datoteko s končnico .items po naslednji sintaksi:

```
vrsta_predmeta ime_predmeta["prikazano besedilo"]
[<ime_prikazne_datoteke>] [(skupina1, skupina2, ...)]
[["oznaka1", "oznaka2", ...]] [{konfiguracija vezi}]
```

SLIKA 13: ZGLED DATOTEKE PREDMETOV

Switch	Phone_Mobile	"My Mobile Phone"	{ channel="network:device:devicename:online" }
Number	Netatmo_Indoor_CO2	"CO2"	{ channel="netatmo:RAA/main:home:inside:co2" }
Number	Azimuth	"Azimuth"	{ channel="astro:sun:home:position#azimuth" }
Contact	Garage	"Garage is [MAP(en.map):%s]"	{ channel="Izave:Zl:command=sensor_binary,respond_to_basic=true" }

## Ljubljana, Vegova 4

### 2.7.2.3 *Vezi*(Ang. *Bindings*)

Vezi nam omogočajo pridobivanje podatkov in upravljanje naprav preko različnih protokolov (MQTT, HTTP, KNX...). V našem primeru nam omogočajo povezavo med programoma OpenHAB in mosquitto. Najprej konfiguriramo povezavo po sintaksi:

```
mqtt:<ime_posrednika>.url=<url_naslov_posrednika>
```

Potem pa po naslednji sintaksi v .items datoteki določamo konfiguracijo posameznim predmetom, ki lahko prikazujejo naše podatke ali pa podatke predmetom pošiljajo:

```
Vrsta_predmeta ime_predmeta {mqtt="<smer>[< ime_ posrednika  
>:<kategorija:mqtt>:<vrsta_povezave>:<preslikava>], <smer>[<  
ime_ posrednika  
>:<kategorija:mqtt>:<vrsta_povezave>:<preslikava>],..."} }
```

### 3 Realizacija komponent in integracija v zaključen sistem

V tem delu opisujem kako sem realiziral sistem v praksi, pojasnim svojo izbiro protokolov in strojne opreme. Za integracijo je bil postavljen strežnik, na katerega sem najprej namestil aplikacijo Mosquitto, potem pa še sistem za upravljanje in nadzor pametne hiše OpenHAB. Izdelal sem napravo, s katero lahko preko Android aplikacije prižigam luč. Naredil sem tudi alarmni sistem, ki začne piskati, če je bilo zaznano gibanje.

#### 3.1 Analiza zmožnosti in opredelitev odločitev

Glede izbire programskih in strojnih rešitev so mi bile prioritete meni dostopna cena, popularnost in preprostost implementacije in primernost moji seminarski nalogi.

##### 3.1.1 KNX in MQTT

Protokola KNX in MQTT sta oba namenjena izmenjavi podatkov med napravami, vendar KNX ne podpira brezžične WiFi povezave. V mojem primeru je bilo preprosteje in ceneje uporabiti vgrajeno WiFi anteno s protokolom MQTT v Raspberry Pi 3, ki sem si ga lastil že pred izdelavo seminarske naloge.

##### 3.1.2 Arduino in ESP8266

Arduino in ESP8266 ploščice so priročni in uporabni pripomočki, kadar gre za izdelavo manjših naprav. V realizaciji pametne hiše se nisem odločil za eno ploščico, ampak sem kombiniral obe. V primeru stikala na daljavo sem serijsko povezal obe ploščici, pri senzorju gibanja sem uporabil samo ESP8266 – ESP01, saj nisem potreboval zmogljivejšega mikrokontrolerja, ta pa je v tem primeru cenovno najbolj ugoden. Pri alarmu pa sem uporabil ESP8266 – ESP12F, saj sem potreboval večje število pinov za zvočnike in luči.

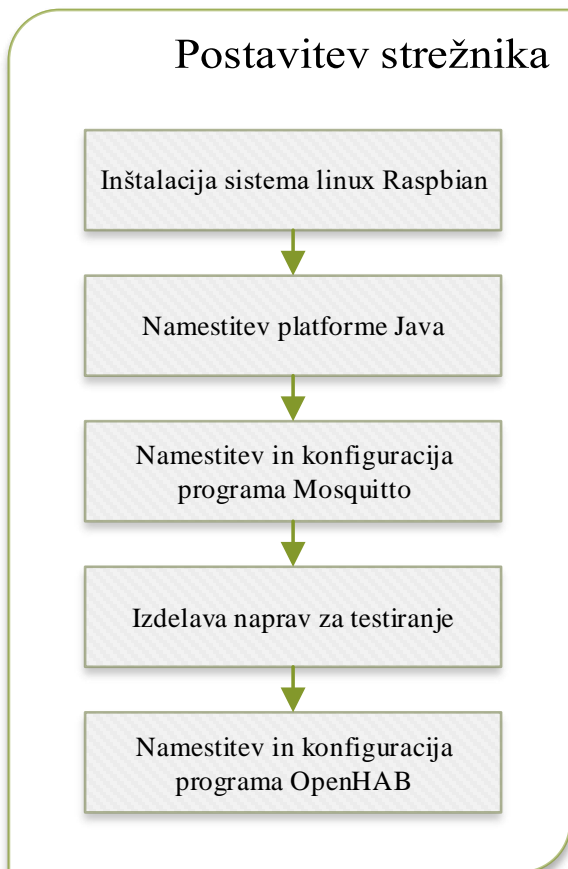
##### 3.1.3 Operacijski sistem na Raspberry PI

Odločil sem se, da za operacijski sistem na Raspberry PI uporabim Raspbian, saj je na internetu veliko več vodičev, in pomoči za ta sistem. Nisem potreboval velike odzivnosti sistema, zato se mi ni splačalo ukvarjati z ročno inštalacijo sistema Arch linux. Sistema Windows 10 IOT Core pa nisem uporabil zato, ker je znano, da so linux sistemi varnejši in stabilnejši od sistemov windows za strežniške namene. Prav tako pa mi Raspbian omogoča več možnosti v primeru širitve sistema.

## 3.2 Realizacija posameznih komponent sistema

### 3.2.1 Strežnik – Posrednik (Ang. Broker)

Za strežnik sem uporabil Raspberry PI 3. Nanj sem namestil operacijski sistem Raspbian in programa Mosquitto in OpenHAB 2, ki tečeta na javanski platformi.



Na Raspberry Pi 3 sem najprej namestil sistem linux Raspbian ga povezal v omrežje, na njem namestil platformo Java in programa Mosquitto in OpenHAB in jih zagnal. Potem sem izdelal naprave, in jih med seboj povezoval. Ko sem končal z izdelavo alarmnega sistema in stikalom na daljavo, sem nastavil aplikacijo OpenHAB, da sem naprave lahko tudi nadzoroval. Najprej sem v datoteki .sitemap naredil kazalo s strukturo, ki je najbolj ustrezala. Potem pa sem moje naprave dodal v datoteko .items. Po uspešni postavitvi sistema sem končno lahko spremljal stanje senzorja gibanja in upravljal stanje luči v moji sobi preko mobilne aplikacije.



## Ljubljana, Vegova 4

SLIKA 14: OPENHAB APLIKACIJA - PRIKAZ SKUPINE



SLIKA 15: OPENHAB APLIKACIJA - STANJA SENZORJA GIBANJA

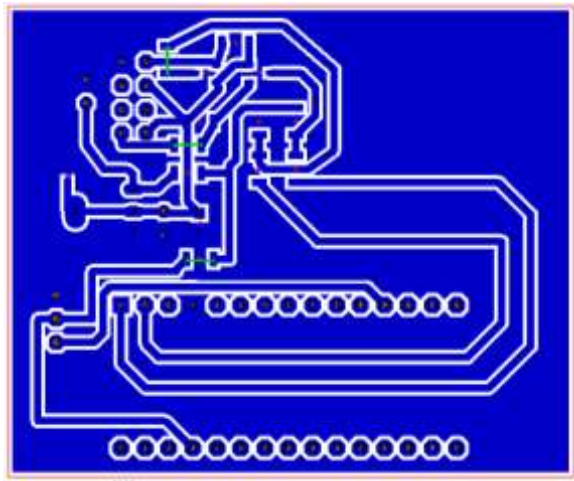
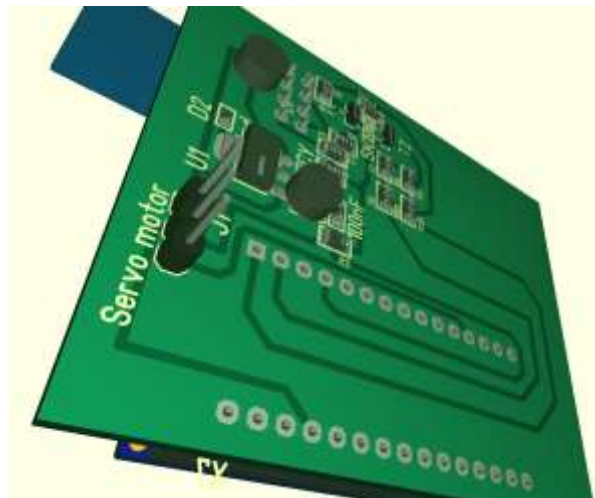


### 3.2.2 Stikalo na daljavo



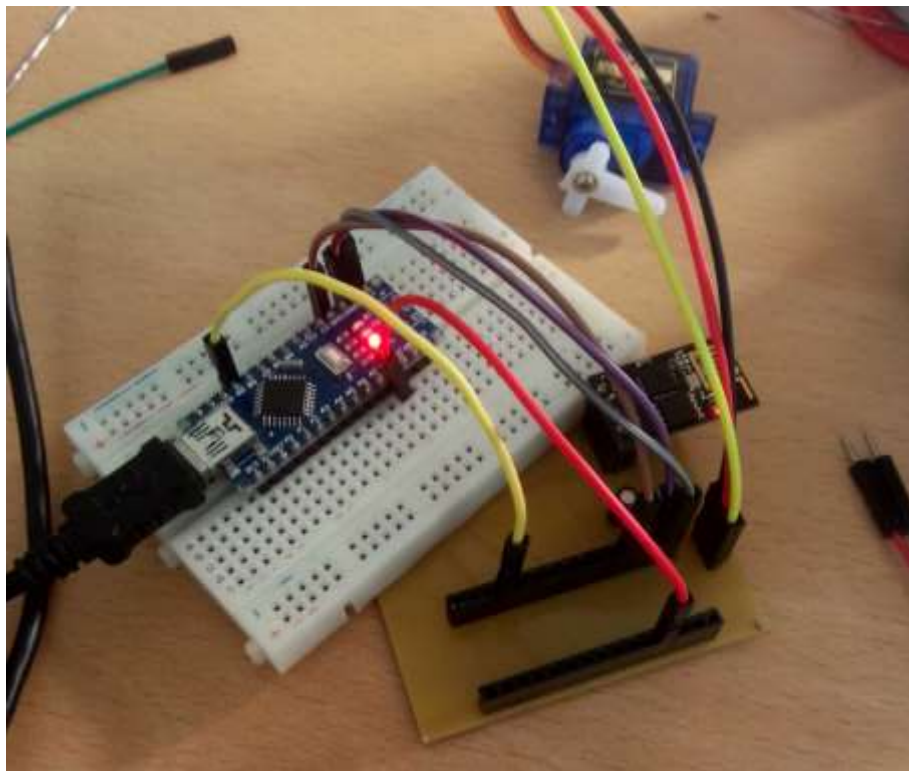
Vsa vezja sem izdelal v programu Traget 3000, ki ima že vgrajenih veliko komponent. Med njimi tudi ESP8266 in Arduino mikrokrmilnike in mi s tem zelo olajšal delo. Omogoča pa tudi 3D pogled končnega vezja.



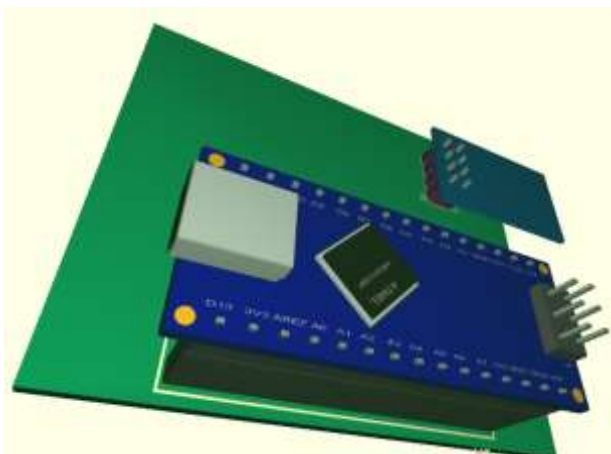
**Ljubljana, Vegova 4***SLIKA 16: STIKALO NA DALJAVO - PRIPRAVLJENA SLIKA VEZJA ZA CNC STROJ**SLIKA 17: STIKALO NA DALJAVO - 3D SLIKA VEZJA V PROGRAMU TARGET 3000 (SPODNJA STRAN)*

Za upravljanje stikala sem uporabil Arduino in ESP8266 ESP – 01 kot WiFi modul, saj Arduino nima vgrajene WiFi antene. Zato podatke na posrednik pošilja in jih z njega pridobiva ESP8266 in potem po serijski povezavi pošlje podatke Arduino, ki obrne motor in prižge stikalo.

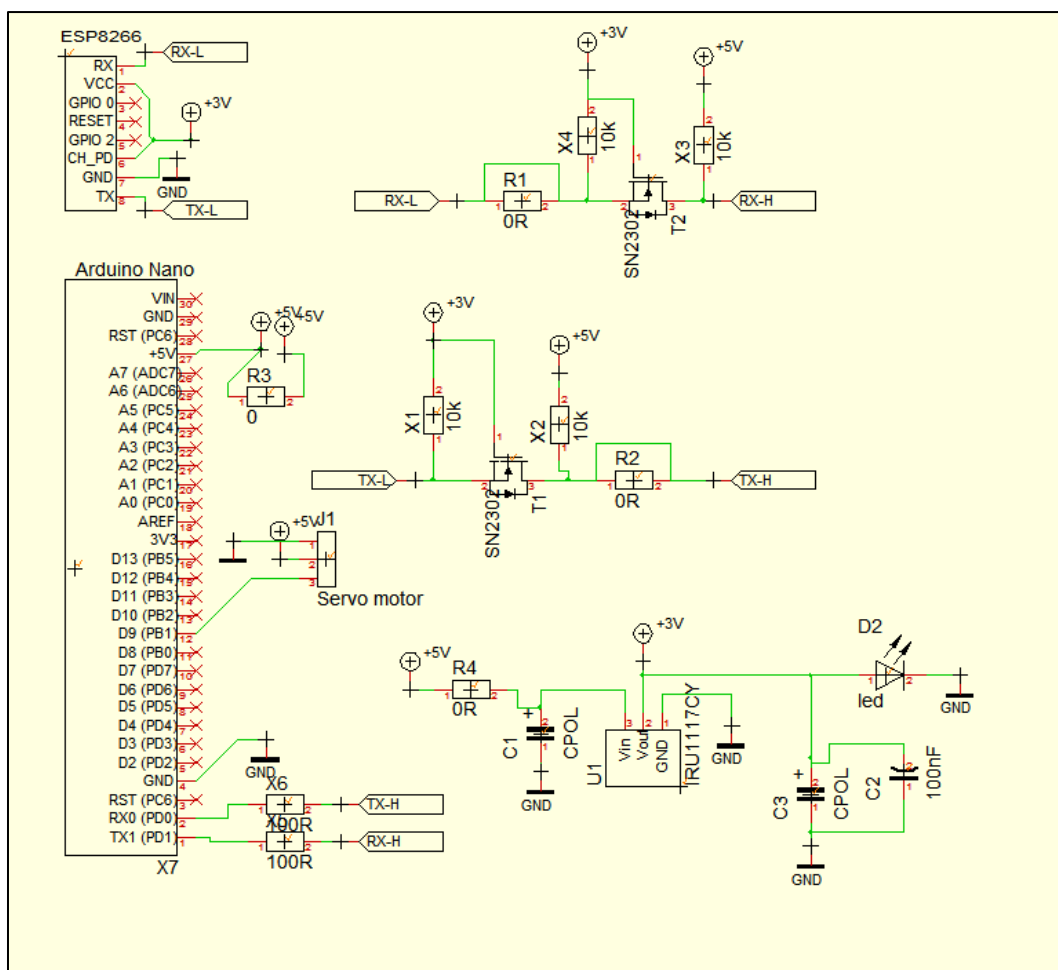
Oba krmilnika sem sprogramiral v programu Arduino IDE. Kodo sem pisal v programskem jeziku C++. Omembe vredna je knjižnica PubSubClient, ki je omogočala uporabo protokola MQTT na ESP8266 krmilniku vključno z naročanjem na podatke, njihovo prejemanje in pošiljanje. Ukaze za vklop ali izklop pridobi preko OpenHAB aplikacije iz kategorije gTurn/sobaHostnik/mainLight.

**Ljubljana, Vegova 4***SLIKA 18: TESTIRANJE VEZJA*

Največ težav sem imel z serijskim povezovanjem med mikrokrmilnikoma zaradi različne operativne napetosti, a sem ob pomoči prijatelja uspel naredi vezje, ki mi je pravilno in brez napak pošiljalo podatke iz ESP8266 na Arduino.

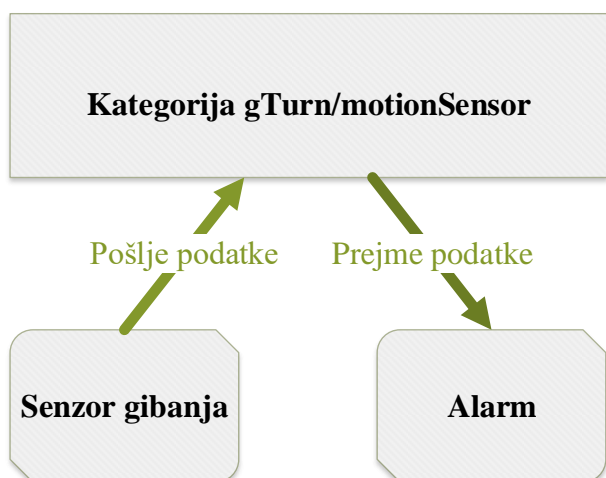
*SLIKA 19: STIKALO NA DALJAVO - 3D SLIKA VEZJA V PROGRAMU TARGET 3000 (SPODNJA STRAN)*

*SLIKA 20: STIKALO NA DALJAVO - SHEMA VEZJA*



### 3.2.3 Alarmni sistem

SLIKA 21: PRIKAZ DELOVANJA ALARMNEGA SISTEMA



Alarmni sistem je zgrajen iz dveh posameznih pripomočkov, ki med seboj komunicirata. Senzor gibanja vsake pol sekunde pošlje podatke o gibanju v kategorijo gTurn/motionSensor in s tem omogoča risanje grafa aplikaciji OpenHAB. Na drugi strani pa pripomoček alarm, ki pridobiva podatke o gibanju iz posrednika, s pomočjo zvočnika, začne piskati kadar je senzor v kategorijo poslal stanje 1, kar pomeni da je zaznal gibanje.

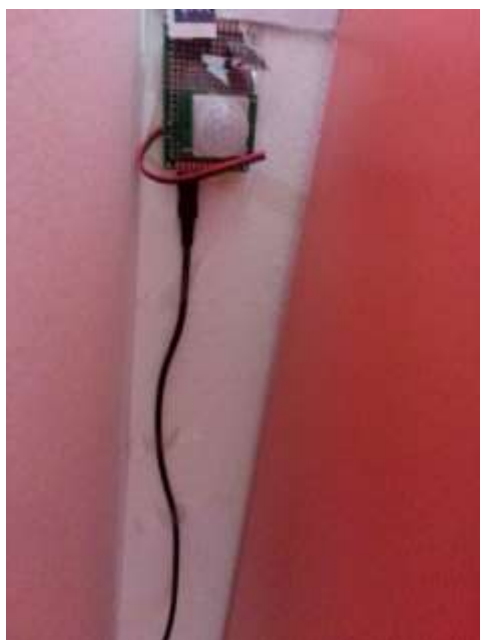
Ljubljana, Vegova 4

### 3.2.3.1 Senzor gibanja

Za senzor gibanja uporabljam samo ESP8266 ESP – 01, saj za branje podatkov iz dejanskega senzorja gibanja in pošiljanje na strežnik potrebuje zgolj 1 digitalni pin.

Nekaj težav sem imel, ker senzor gibanja ni pravilno poslal podatkov. Kasneje pa sem ugotovil, da je bila napaka v dokumentaciji, kajti pisalo je, da je potrebno senzor napajati z napetostjo 3,3 volta. Dejansko pa senzor potrebuje napajanje petih voltov.

SLIKA 22: SENZOR GIBANJA - DELUJOČI PROTOTIP IZDELKA



SLIKA 23: SENZOR GIBANJA (VIR: [HTTP://WWW.EBAY.COM/ITM/HC-SR501-InfraRed-PIR-Motion-Sensor-Module-for-Arduino-Raspberry-Pi-New-/272600006597?hash=item3f7839ffc5](http://www.ebay.com/itm/HC-SR501-InfraRed-PIR-Motion-Sensor-Module-for-Arduino-Raspberry-Pi-New-/272600006597?hash=item3f7839ffc5))



Ljubljana, Vegova 4

### 3.2.3.2 Alarm

Alarm pa je narejen z krmilnikom ESP8266 ESP-12F, kar zadovoljuje, da ne potrebujem dodatnih krmilnikov. Z njim pridobivamo podatke iz posrednika, predvajamo piskanje in prižigamo svetleče diode. Prav tako pa ga programiramo z okoljem Arduino IDE in uporabo knjižnice PubSubClient.

SLIKA 24: ALARM MED GIBANJEM



## 4 Sklepne ugotovitve

Z implementacijo sistema pametne hiše sem se po pričakovanjih veliko naučil. Uporabil sem veliko načinov in tehnologij. Mikrokontroler ESP8266 sem uporabil kot samostojni krmilnik (ESP01 in ESP12F), prav tako pa sem ESP01 uporabil kot WiFi modul za Arduino. Naučil sem se nastaviti aplikacijo OpenHAB in uporabiti protokol MQTT. Če bi se moral ponovno odločil, bi se enake seminarske naloge lotil ponovno. Sistem sem pol leta preizkušal v dijaškem domu v svoji sobi. Po nekaj začetnih napakah je sistem deloval popolno. Bil je v polni uporabi. Glavno težavo so mi predstavljala pravila dijaškega doma, zato nisem mogel uporabljati alarmnega sistema. Prav tako bi lahko tudi stikalo na daljavo naredil na elegantnejši način z uporabo električnega stikala (Ang. relay), vendar bi moral razstaviti stikalo, zato sem omejen s pravili raje uporabil servomotor.

## 5 Literatura in viri

Kovačič, Matija. 2014. Centralni nadzorni sistem daljinskega ogrevanja športnega centra. Ljubljana : [M. Kovačič], 2014

Krainer, Aleš. 1996. Naslov projekta Pametna hiša - Termodinamični in optični del. V Ljubljani : FAGG, Oddelek za gradbeništvo in geodezijo, 1996

Krašna, Gregor. Kaj to je pametna hiša. Nova Gorica : [G. Krašna], 2009

Menhart, Boštjan. 2007. Centralni nadzorni sistem trgovskega objekta. Maribor : [B. Menhart], 2007

Pogačar, Robert. 2011. Nadgradnja sistemov ogrevanja, hlajenja in prezračevanja na centralni nadzorni sistem. Kranj : [R. Pogačar], 2011

Simončič, Mirjam. Pametna hiša. Maribor : [M. Simončič], 2016

Stražišar, Rok. 2008. Sistem za vodenje in nadzor pametne hiše. Ljubljana : [R. Stražišar], 2008

Adafruit Learning System, pridobljeno z: <https://learn.adafruit.com/assets/6122> dne 26. 3. 2017

Amazon, pridobljeno z: <https://www.amazon.com/Esp8266-Wireless-Transceiver-Arduino-IFANCY-TECH/dp/B01J94UWNC> dne 1. 4. 2017

Arch linux, pridobljeno z: <https://www.archlinux.org/> dne 14. 04. 2017

CH340 Drivers for Windows, Mac and Linux, pridobljeno z: <http://sparks.gogo.co.nz/ch340.html> dne 2. 3. 2017

Download Raspbian for Raspberry Pi, pridobljeno z: <https://www.raspberrypi.org/downloads/raspbian> dne 20. 3. 2017

EBay: NodeMcu Lua collection on eBay!, pridobljeno z: <http://www.ebay.com/cln/kenneyrsw-lk/NodeMcu-Lua/336792467017> dne 24. 2. 2017

ESP-12 VS Arduino Uno R3 VS Arduino Nano - Everything ESP8266, pridobljeno z: <http://www.esp8266.com/viewtopic.php?F=6&t=4348> dne 26. 1. 2017

ESP8266 (nodemcu ESP-12E) Quickstart, pridobljeno z: <https://gist.github.com/eriknomitch/a904ad1d806c349970f553af3e93a0e3> dne 2. 11. 2016



## Ljubljana, Vegova 4

Github - Esp8266/Arduino: ESP8266 core for Arduino, pridobljeno z: <https://github.com/esp8266/Arduino> dne 12. 2. 2017

Github - Knolleary/pubsubclient: A client library for the Arduino Ethernet Shield that provides support for MQTT, pridobljeno z: <https://github.com/knolleary/pubsubclient> dne 24. 1. 2017

HC-SR501 Infrared PIR Motion Sensor Module for Arduino Raspberry pi new | ebay, pridobljeno z: <http://www.ebay.com/itm/HC-SR501-Infrared-PIR-Motion-Sensor-Module-for-Arduino-Raspberry-pi-new-/272600006597?Hash=item3f7839ffc5> dne 25. 3. 2017

Installing openhab Home Automation on Raspberry Pi | MCU on Eclipse, pridobljeno z: <https://mcuoneclipse.com/2015/12/23/installing-openhab-home-automation-on-raspberry-pi> dne 26. 3. 2017

Internet Of Things – Remote Temperature Sensor Project | Stafferier, pridobljeno z: <http://stafferier.com/?P=107> dne 26. 3. 2017

KNX (standard), pridobljeno z [https://en.wikipedia.org/wiki/KNX\\_\(standard\)](https://en.wikipedia.org/wiki/KNX_(standard)) dne 14. 4. 2017

Learn about Windows 10 IoT Core, pridobljeno z <https://developer.microsoft.com/en-us/windows/iot/explore/iotcore> dne, 14. 4. 2017

MQTT, pridobljeno z: <http://mqtt.org> dne 26. 3. 2017

Openhab 2 Documentation - openhab 2 - Empowering the Smart Home, pridobljeno z: <http://docs.openhab.org> dne 25. 3. 2017

Openhab, pridobljeno z: <https://www.openhab.org> dne 25. 3. 2017

Raspberry Pi 3 - Pimoroni, pridobljeno z: <https://shop.pimoroni.com/products/raspberry-pi-3> dne 26. 3. 2017

Ljubljana, Vegova 4

## 6 Priloge

PRILOGA 1: KODA ALARMA

### Koda alarma

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#define TONE_PIN 5
#define ALARM_LED_1 14
#define ALARM_LED_2 12
#define NOTE_G3 196
#define NOTE_C3 131
// Update these with values suitable for your network.

const char* ssid = "ImeOmrezja";
const char* password = "gesloOmrezja";
const char* mqtt_server = "192.168.1.250"; // IP naslov posrednika

WiFiClient espClient;
PubSubClient client(espClient);

void setup() {
  pinMode(ALARM_LED_1, OUTPUT);
  pinMode(ALARM_LED_2, OUTPUT);
  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

void setup_wifi() {

  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
```



## Ljubljana, Vegova 4

```
    Serial.print((char)payload[i]);
}
Serial.println();

// Switch on the LED if an 1 was received as first character
if ((char)payload[0] == '1') {
    alarm();
}
}

void reconnect() {
    // Loop until we're reconnected
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        // Attempt to connect
        if (client.connect("Alarm")) {
            Serial.println("connected");
            client.subscribe("gTurn/motionSensor");
        } else {
            Serial.print("failed, rc=");
            Serial.print(client.state());
            Serial.println(" try again in 5 seconds");
            // Wait 5 seconds before retrying
            delay(5000);
        }
    }
}

void alarm() {
    for (int i = 0; i < 8; i++) {
        digitalWrite(ALARM_LED_1, (boolean)i%2);
        digitalWrite(ALARM_LED_2, (boolean)(i+1)%2);
        tone(TONE_PIN, NOTE_G3, 250);
        delay(250 * 1.30);
        noTone(8);

        tone(TONE_PIN, NOTE_C3, 250);
        delay(250 * 1.30);
        noTone(8);
    }
    digitalWrite(ALARM_LED_1, LOW);
    digitalWrite(ALARM_LED_2, LOW);
}

void loop() {

    if (!client.connected()) {
        reconnect();
    }
    client.loop();
}
```

## Ljubljana, Vegova 4

PRILOGA 2: KODA SENZORJA GIBANJA

## Koda senzorja gibanja

```
#define GPIO2 2

#include <ESP8266WiFi.h>
#include <PubSubClient.h>

// Update these with values suitable for your network.

const char* ssid = "ImeOmrezja";
const char* password = "gesloOmrezja";
const char* mqtt_server = "192.168.1.250"; // IP naslov posrednika
unsigned int lastMsg = 0;
WiFiClient espClient;
PubSubClient client(espClient);
void setup() {
    pinMode(GPIO2, INPUT); // Initialize the BUILTIN_LED pin as an output
    setup_wifi();
    client.setServer(mqtt_server, 1883);
    client.setCallback(callback);
}

void setup_wifi() {

    delay(10);

    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
    }
}

void callback(char* topic, byte* payload, unsigned int length) {

}

void reconnect() {
    while (!client.connected()) {
        if (client.connect("MotionSensor")) {
            client.publish("gTurn/motionSensor", "Connected");
        } else {
            delay(5000);
        }
    }
}

boolean previousState = false;
int minutesCounter = 0;
void loop() {

    if (!client.connected()) {
        reconnect();
    }
    client.loop();
    long now = millis();
    if (now - lastMsg > 100) {
```

**Ljubljana, Vegova 4**

```
boolean currentState = digitalRead(GPIO2);
minutesCounter++;

//on every 0.1s - send to alarm
if (currentState) {
    client.publish("gTurn/motionSensor", "1");
} else {
    client.publish("gTurn/motionSensor", "0");
}
lastMsg = now;

//on change || on 30 seconds - send to openhab
if ((previousState ^ currentState) || (minutesCounter > 300)) {
    if (currentState) {
        client.publish("gTurn/motionSensor/openHAB", "1");
    } else {
        client.publish("gTurn/motionSensor/openHAB", "0");
    }
    previousState = currentState;
    minutesCounter=0;
}
}
```

## Ljubljana, Vegova 4

PRILOGA 3: ESP8266 – ESP01 KODA ZA STIKALO NA DALJAVO

## ESP8266 – ESP01 koda za stikalo na daljavo

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

const char* ssid = "ImeOmrezja";
const char* password = "gesloOmrezja";
const char* mqtt_server = "192.168.1.250"; // IP naslov posrednika

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int value = 0;

void setup() {
    pinMode(BUILTIN_LED, OUTPUT);
    Serial.begin(115200);
    setup_wifi();
    client.setServer(mqtt_server, 1883);
    client.setCallback(callback);
}

void setup_wifi() {

    delay(10);
    // We start by connecting to a WiFi network
    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
    }
}

void callback(char* topic, byte* payload, unsigned int length) {
    int c = 0;
    String dataString;
    String data = "";
    for (int i = 0; i < length; i++) {
        data += (char)payload[i];
    }
    Serial.println();
    if (data == "ON")
        Serial.print((char)1);
    else if (data == "OFF")
        Serial.print((char)2);
}

void reconnect() {
    // Loop until we're reconnected
    while (!client.connected()) {
        if (client.connect("ESP8266Client")) {
            client.publish("gTurn/sobaHostnik/mainLight", "OFF");
            client.subscribe("gTurn/sobaHostnik/mainLight");
        } else {
            delay(5000);
        }
    }
}
```

**Ljubljana, Vegova 4**

```
    }  
  }  
}  
void loop() {  
  if (!client.connected()) {  
    reconnect();  
  }  
  client.loop();  
  long now = millis();  
  if (now - lastMsg > 2000) {  
  }  
}
```

## Ljubljana, Vegova 4

PRILOGA 4: ARDUINO KODA ZA STIKALO NA DALJAVO

## Arduino koda za stikalo na daljavo





```
#include <SoftwareSerial.h>
#include <Servo.h>
Servo pusher;
SoftwareSerial esp(2, 3); //RX TX
void setup() {
    esp.begin(115200);
    Serial.begin(115200);
    //pusher.attach(9);
}

void loop() {
    if (esp.available() && esp.read() == '\n') {
        byte c = esp.read();
        Serial.println(c);
        if (c == 1) {
            turnOn();
        } else if (c == 2) {
            turnOff();
        }
    }
}

void turnOn() {
    pusher.attach(9);
    pusher.write(180);
    delay(1000);
    pusher.write(0);
    delay(1000);
    pusher.detach();
}

void turnOff() {
    pusher.attach(9);
    pusher.write(180);
    delay(1000);
    pusher.write(0);
    delay(1000);
    pusher.detach();
}
```

**Ljubljana, Vegova 4**
*PRILOGA 5: TABELA PRIMERJAVE MIKROKRMILNIKOV ARDUINO IN ESP8266*

	Arduino Uno	Arduino Nano	ESP-12	ESP-12E
Izgled				
Največji tok	40mA	40mA	12mA	12mA
Urna frekvenca	16MHz	16MHz	80-165 MHz	80-165 MHz
Čip	ATmega328	ATmega328	ESP8266	ESP8266
Bliskovni pomnilnik	32KB	16 KB / 32KB	4 MB	4 MB
Statični RAM	2 KB	1KB / 2 KB	64 KB	64 KB
Električno izbrisljivi programirljivi bralni pomnilnik	1 KB	512 B / 1 KB	Ne	Ne
USB	Da	Da	Ne	Ne
Priklop za zunanje napajanje	Da	Ne	Ne	Ne
Operativna napetost	5V	5V	3,3V	3,3V
Število pinov	32	30	16	22
Programsko upravljaljivi pini	14	14	13 (9 Digital I/O)	16 (11 Digital I/O)
Analogni vhodi/izhodi	6	8	1	1
WiFi	Ne	Ne	Da	Da
Temperaturni senzor	Ne	Ne	Da	Da
PWM	6	6	9	9
Gumb za resetiranje	Da	Da	Ne	Ne
Okvirna cena	3.7 €	2.9 €	3,50 €	2.9 €

## Ljubljana, Vegova 4

PRILOGA 6: PRIMERJAVA ARDUINO MIKROKRMILNIKOV (Vir: <https://learn.adafruit.com/assets/6122>)

	Processor	Processor Voltage	Supply Voltage	Flash	SRAM	Digital I/O Pins	PWM Pins	Analog Inputs	Hardware Serial Ports	Dimensions	Shield Compatibility	Notes and Special Features
Uno	16MHz Atmega 328	5v	7-12v	32Kb	2Kb	14	6	6	1	2.1"x2.7" 53x75mm	Excellent (most will work)	
Uno Ethernet	16MHz Atmega 328	5v	7-12v	32Kb	2Kb	14	6	6	1	2.1"x2.7" 53x75mm	Very Good (some pin conflicts)	Has Ethernet Port. Requires FTDI cable to program.
Mega	16MHz Atmega 2560	5v	7-12v	256Kb	8Kb	54	14	16	4	2.1"x4" 53x102mm	Good (some pinout differences)	
Mega ADK	16MHz Atmega 2560	5v	7-12v	256Kb	8Kb	54	14	16	4	2.1"x4" 53x102mm	Good (some pinout differences)	Works with Android Development Kit.
Leonardo	16MHz Atmega 32U4	5v	7-12v	32Kb	2.5Kb	20*	7	12*	1	2.1"x2.7" 53x75mm	Fair (many Pinout Differences)	Native USB capabilities. USB Micro B programming port.
Due	84MHz ARM SAM3X8E	3.3v	7-12v	512Kb	96Kb	54	12	12	4	2.1"x4" 53x102mm	Poor (voltage and pinout differences)	Fastest processor. Most memory. 2-channel DAC. USB micro B programming port. Native micro AB port.
Micro	16MHz Atmega 32U4	5v	5v	32Kb	2.5Kb	20*	7	12*	1	0.7"x1.9" 18x49mm	N/A	Smallest board size. Native USB capabilities
Flora	8MHz Atmega 32U4	3.3v	3.5-16v	32Kb	2.5Kb	8*	4	4*	1	1.75" dia 44.5mm dia	N/A	Sewable Pads. Fabric-friendly design. Native USB Capabilities
DC Boarduino	16MHz Atmega 328	5v	7-12v	32Kb	2Kb	14	6	6	1	0.8"x3" 20.5x76mm	N/A	Can build without headers or sockets for smaller size. Requires FTDI cable for programming
USB Boarduino	16MHz Atmega 328	5v	5v (USB)	32Kb	2Kb	14	6	6	1	0.8"x3" 20.5x76mm	N/A	Can build without headers or sockets for smaller size. USB Mini B programming port.
Menta	16MHz Atmega 328	5v	7-12v	32Kb	2Kb	14	6	6	1	0.8"x3" 20.5x76mm	Excellent (most will work)	Mint-Tin Size and Prototyping Area. Requires FTDI cable for programming.



Ljubljana, Vegova 4

## 7 Izjava o avtorstvu

Izjavljam, da je seminarska naloga v celoti moje avtorsko delo, ki sem ga izdelal samostojno s pomočjo navedene literature in pod vodstvom mentorja.

Ljubljana, 29. 04. 2017

Jakob Hostnik

