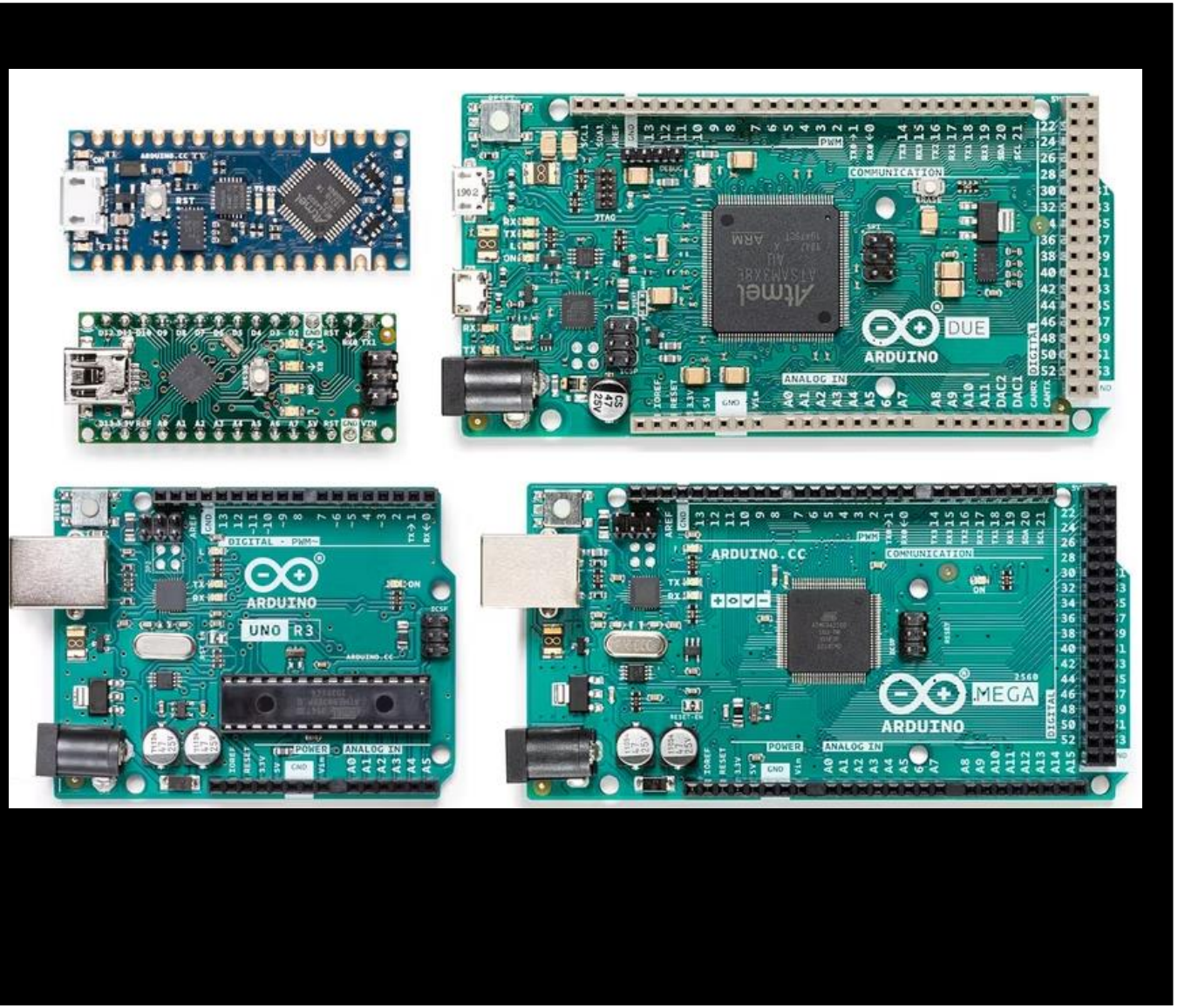
MIKROKONTROLLERIT JA NIIDEN HYÖDYNTÄMINEN

TIMO S / OH8CCS









ARDUINO POHJAISET LAITTEET

Uno, Nano, Mega, Leonardo, Due…

ATmega328P pohjaiset yleisiä.

Lukematon määrä eri variaatioita

Sisältää lukuisia Input ja Output liitäntöjä, joita voidaan ohjelmoida vapaasti

Lisäpalikoilla saadaan liitäntöjä kasvatettua lähes rajattomasti

https://www.arduino.cc/

5€ - 50€

ESP8266 JA ESP32 POHJAISET LAITTEET

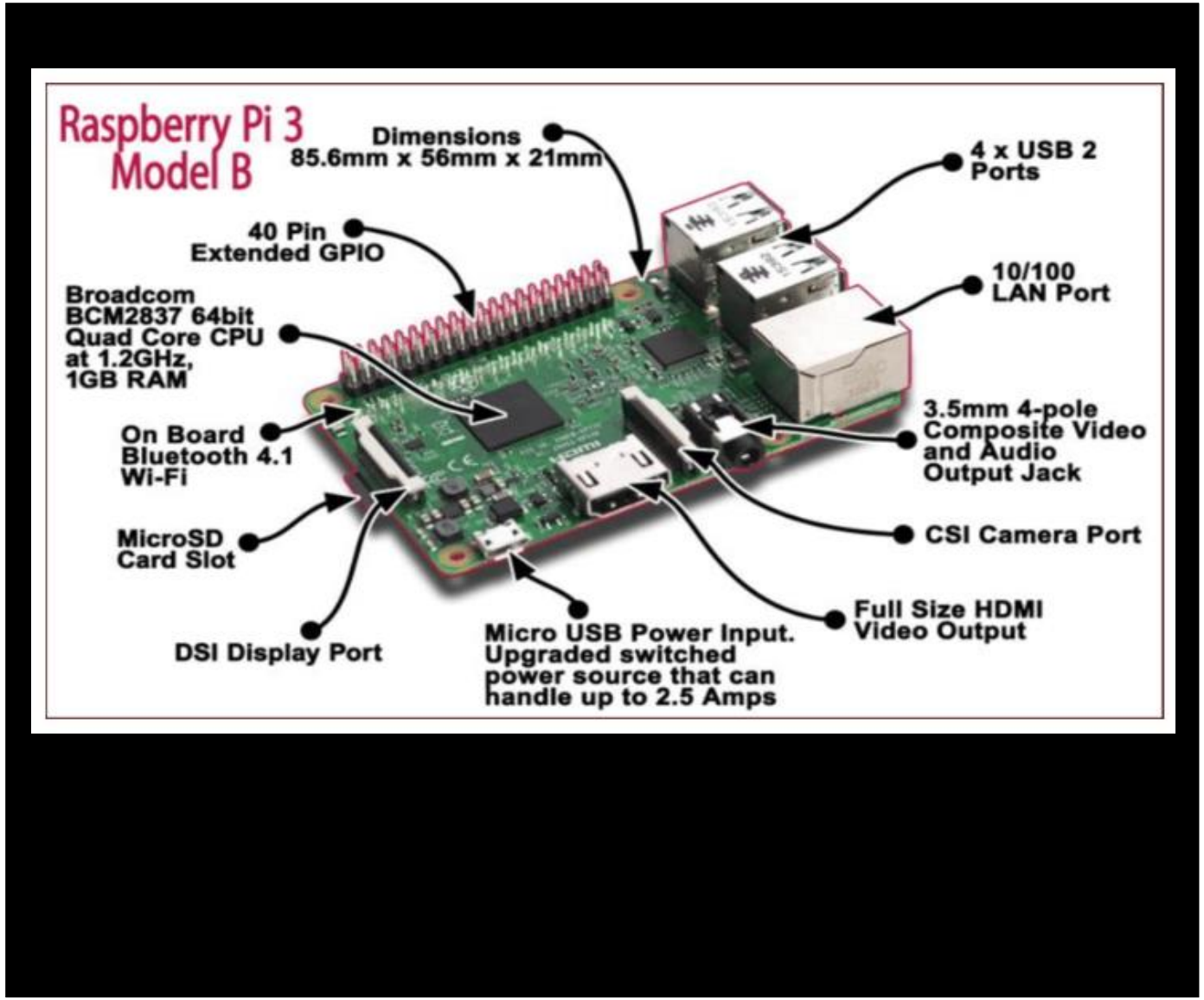
Prosessoriperheet ESP8266 ja ESP32

Ohjelmointi pohjautuu hyvin paljon Arduinoihin.

Sisältää oletuksena WiFi:n

https://esp32.com/

5€ - 50€



RASPBERRY PI

Minitietokone

Käyttöjärjestelmä SD kortilla

Broadcom BCM2837 64bit CPU tai uudempi.

Sisältää WiFi:n, USB:n, BL, LAN, HDMI portit.
Lisäksi lukuisia ohjelmoitavia GPIO liitäntöjä

Rpi 4 nykyään yleisin. Rpi 5 tulossa

https://www.raspberrypi.org/

N 70€ – 150€

SHELLY -MODUULIT

Ohjelmoitavia IoT -moduuleita.

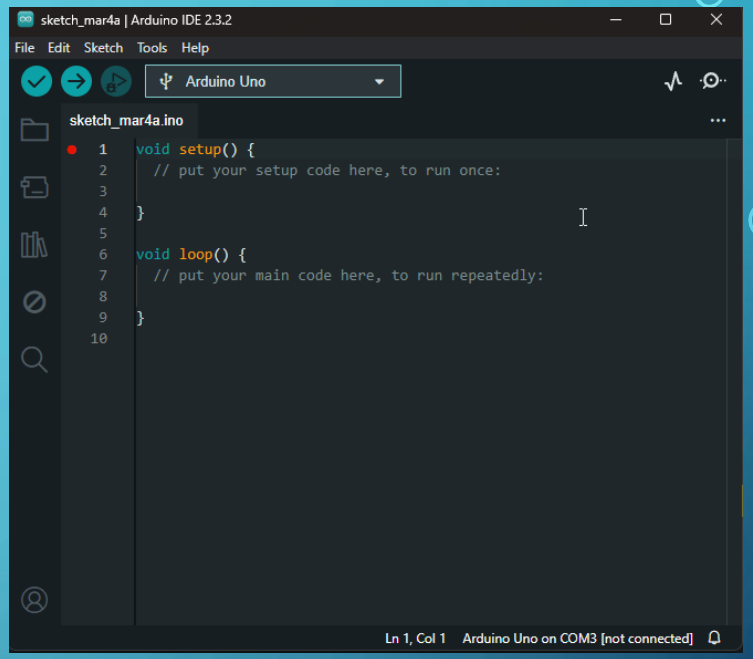
ESP32 pohjainen laite.

Langattomia, toimii WiFi tai BLE kautta

https://www.shelly.com/en-fi

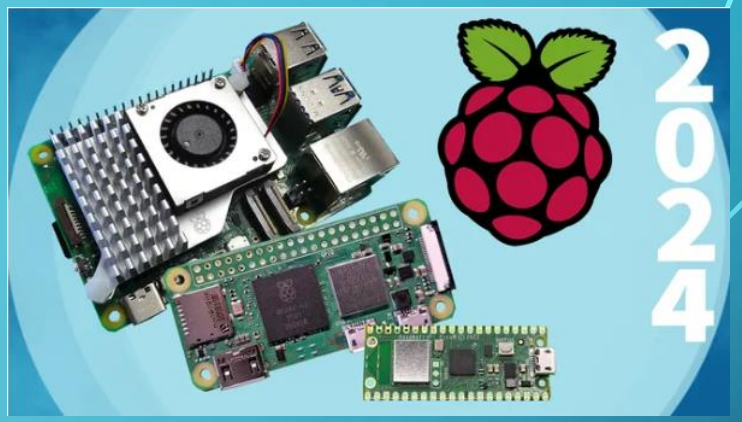
10€ – 200€

ARDUINO OHJELMOINTI

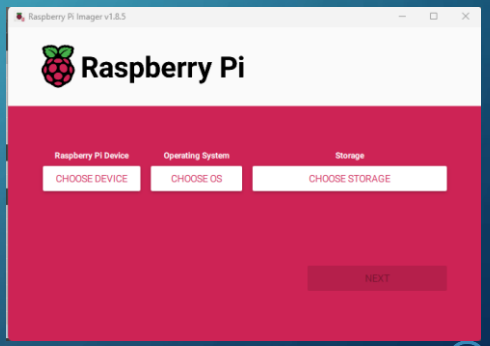


- Ohjelmoidaan omalla applikaatiolla (Arduino IDE)
- C/C++ tyyppinen -kieli käytössä yleensä
- Netistä löytyy valmista koodia moneen tarkoitukseen
- Arduino IDE applikaatiossa on valmiita esimerkkejä paljon
- Samalla Arduino IDE:llä ohjelmoidaan myös ESP8266 ja ESP32 –laitteet.

RASPBERRY OHJELMOINTI



- Raspberry Pi käyttöjärjestelmä on yleensä Linux –pohjainen
- Raspberry PI Imager asentaa helposti halutun käyttöjärjestelmän.
- Valmiita ohjelmia voidaan ladata lukemattomia
- Python on suosittu ohjelmointikieli
- Ohjataan joko komentoriviltä tai graafisesta käyttöliittymästä.

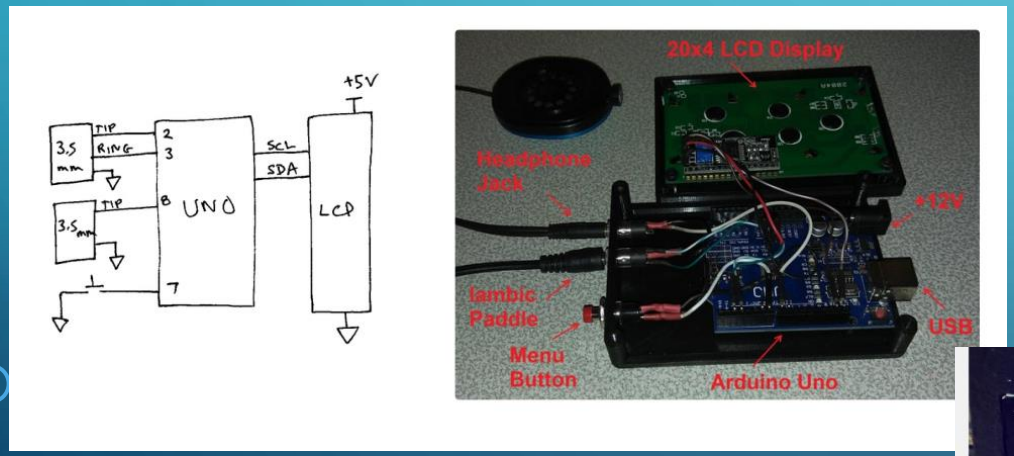


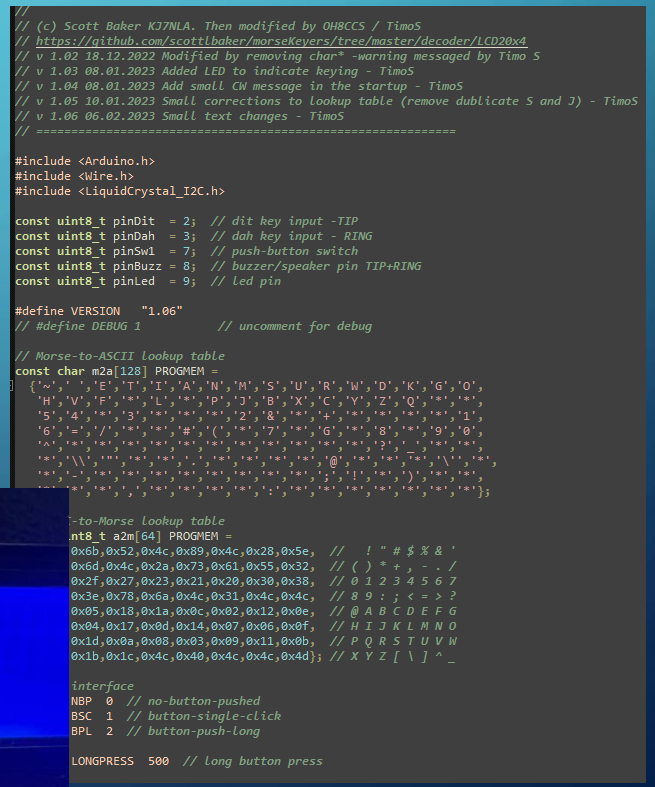
SHELLY OHJELMOINTI

- Shelly moduuli sisältää valmiina oman firmwaren (käyttöjärjestelmä)
- Konfigurointi kännykkäapplikaatio tai selaimella IP osoitteen mukaan
- Ohjelmoidaan yleensä skripteillä (Espruino, JS pohjainen kieli)
- Netistä löytyy paljon valmiita scriptejä ja verkkosivujen palikoita (API)
- Paljon käytössä taloautomaatiossa ja esim spottisähkön ohjauksessa.

KÄYTÄNNÖN TOTEUTUKSIA

- Arduinolla, näyttömoduulilla ja muutamalla lisäkomponentilla saadaan aikaiseksi ärsyttävän pedantti CW opetuslaite.

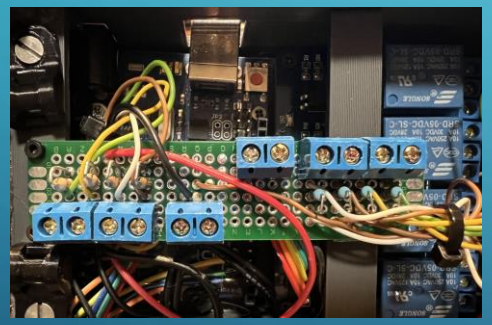




```
//
// (c) Scott Baker KJ7NLA. Then modified by OH8CCS / TimoS
// https://github.com/scottlbaker/morseKeyers/tree/master/decoder/LCD20x4
// v 1.02 18.12.2022 Modified by removing char* -warning messaged by Timo S
// v 1.03 08.01.2023 Added LED to indicate keying - TimoS
// v 1.04 08.01.2023 Add small CW message in the startup - TimoS
// v 1.05 10.01.2023 Small corrections to lookup table (remove dublicate S and J) - TimoS
// v 1.06 06.02.2023 Small text changes - TimoS
// ======================================================

#include <Arduino.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

const uint8_t pinDit  = 2;  // dit key input -TIP
const uint8_t pinDah  = 3;  // dah key input - RING
const uint8_t pinSw1  = 7;  // push-button switch
const uint8_t pinBuzz = 8;  // buzzer/speaker pin TIP+RING
const uint8_t pinLed  = 9;  // led pin

#define VERSION   "1.06"
// #define DEBUG 1           // uncomment for debug

// Morse-to-ASCII lookup table
const char m2a[128] PROGMEM =
  {'~',' ','E','T','I','A','N','M','S','U','R','W','D','K','G','O',
   'H','V','F','*','L','*','P','J','B','X','C','Y','Z','Q','*','*',
   '5','4','*','3','*','*','*','2','&','*','+','*','*','*','*','1',
   '6','=','/','*','*','#','(','*','7','*','G','*','8','*','9','0',
   '^','*','*','*','*','*','*','*','*','*','?','_','*','*',
   '*','\\','"','*','*','.','*','*','*','*','@','*','*','*','\'','*',
   '*','-','*','*','*','*','*','*','*',';','!','*',')','*','*',
   '*','*',',','*','*','*','*',':','*','*','*','*','*','*'};

// ASCII-to-Morse lookup table
uint8_t a2m[64] PROGMEM =
  {0x6b,0x52,0x4c,0x89,0x4c,0x28,0x5e,  //   ! " # $ % &
   0x6d,0x4c,0x2a,0x73,0x61,0x55,0x32,  // ( ) * + , - . /
   0x2f,0x27,0x23,0x21,0x20,0x30,0x38,  // 0 1 2 3 4 5 6 7
   0x3e,0x78,0x6a,0x4c,0x31,0x4c,0x4c,  // 8 9 : ; < = > ?
   0x05,0x18,0x1a,0x0c,0x02,0x12,0x0e,  // @ A B C D E F G
   0x04,0x17,0x0d,0x14,0x07,0x06,0x0f,  // H I J K L M N O
   0x1d,0x0a,0x08,0x03,0x09,0x11,0x0b,  // P Q R S T U V W
   0x1b,0x1c,0x4c,0x40,0x4c,0x4c,0x4d}; // X Y Z [ \ ] ^ _

// interface
NBP  0  // no-button-pushed
BSC  1  // button-single-click
BPL  2  // button-push-long

LONGPRESS  500  // long button press
```

KÄYTÄNNÖN TOTEUTUKSIA

- Arduinolla, relemoduulilla ja muutamalla lisäkomponentilla saadaan aikaiseksi logiikka antennien ohjaukseen.



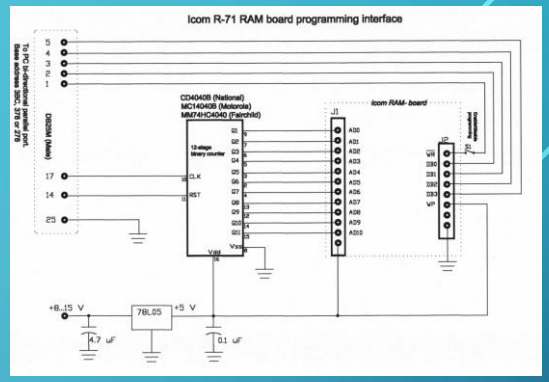

```
// test relays in the startup
digitalWrite(RELAY1_PIN, ON);
delay (1000);
digitalWrite(RELAY1_PIN, OFF);

digitalWrite(RELAY2_PIN, ON);
delay (1000);
digitalWrite(RELAY2_PIN, OFF);

digitalWrite(RELAY3_PIN, ON);
delay (1000);
digitalWrite(RELAY3_PIN, OFF);

digitalWrite(RELAY4_PIN, ON);
delay (1000);
digitalWrite(RELAY4_PIN, OFF);

}

// -------------------------------------
// main loop starts here
// -------------------------------------
void loop() {

Serial.print("A0: ");
Serial.print(analogRead(A0));
Serial.print(" - A1: ");
Serial.print(analogRead(A1));
Serial.print(" - A2: ");
Serial.print(analogRead(A2));
Serial.print(" - A3: ");
Serial.print(analogRead(A3));


// BCD type of values from input pins
bitValue = BinaryValues();
Serial.print(" ::: binary value: ");
Serial.print(bitValue);

// --------------------logic for relays selection -----------------
switch (bitValue) {

  case 0:  // 60m - 4
    digitalWrite(RELAY1_PIN, OFF);
    digitalWrite(RELAY2_PIN, OFF);
    digitalWrite(RELAY3_PIN, OFF);
    digitalWrite(RELAY4_PIN, ON);
    Serial.println(" :: Relay 4 ON");
    break;
  case 1:  // 160m - 4
    digitalWrite(RELAY1_PIN, OFF);
    digitalWrite(RELAY2_PIN, OFF);
    digitalWrite(RELAY3_PIN, OFF);
```

KÄYTÄNNÖN TOTEUTUKSIA

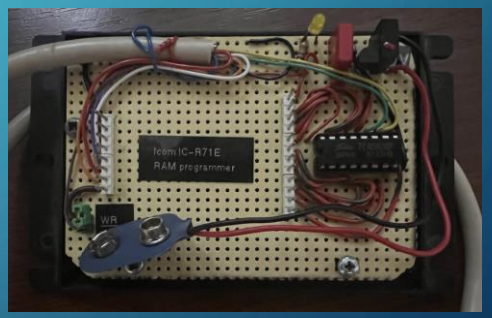
- Raspberry Pi ja Python
- eQSL kortit omalle koneelle Python koodin avulla.

```
#!/usr/bin/python
# coding=utf-8
# --------------------------
# modified for python 3.9
# Timo Savikoski 14.01.2023
# NOTE: When downloading QSL -cards. eQSL move them to Archived.
# --------------------------
import urllib
import urllib.request as urllib2
import re
import os
import configparser

def adifFixup(rec):
    if rec.__contains__('BAND') and not rec.__contains__('BAND_RX'):
        rec['BAND_RX'] = rec['BAND']
    if rec.__contains__('FREQ') and not rec.__contains__('FREQ_RX'):
        rec['FREQ_RX'] = rec['FREQ']

def adiParse(adif_data):
    raw = str(adif_data)
    # print (raw)

    # Find the EOH, in this simple example we are skipping
    # header parsing.
    pos = 0
    m = re.search('EOH', raw, re.IGNORECASE)
    if m != None:
        # Start parsing our ADIF file after the  marker
        pos = m.end()

    recs = []
    rec = dict()
    while 1:
        # Find our next field definition &lt;...&gt;
        pos = raw.find('<', pos)

        if pos == -1:
            return recs
        endPos = raw.find('>', pos)

        # Split to get individual field elements out
        fieldDef = raw[pos + 1:endPos].split(':')
        fieldName = fieldDef[0].lower()
```

KÄYTÄNNÖN TOTEUTUKSIA



- Icom IC-R71E, IC-735 ja IC-751 on paristovarmennettu RAM board
- Pariston tyhjennettyä koko RX tai RIG kuolee.
- Tällä laitteella saa RAM boardin ohjelmoitua uudestaan.
- Tarvitaan vanha PC & DOS, ohjelmointiohjelma ja tämä laite.
- Nykyään Arduinolle on samanlainen laite olemassa ja paljon helpommin hallittavissa



KÄYTÄNNÖN TOTEUTUKSIA

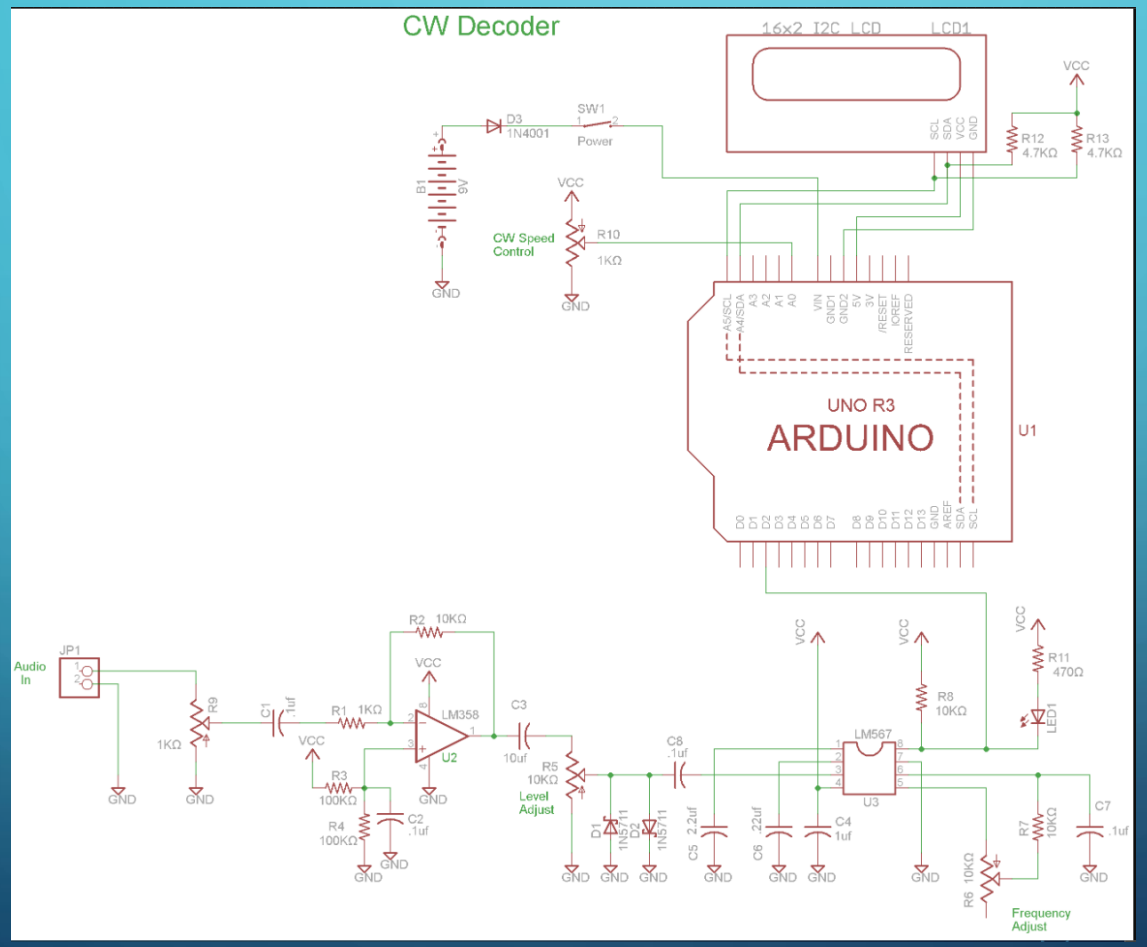
Do-it-yourself projects include:

- LCD shield
- Station timer
- General purpose panel meter
- Dummy load and watt meter
- CW automatic keyer
- Morse code decoder
- PS2 keyboard CW encoder
- Universal relay shield
- Flexible sequencer
- Rotator controller
- Directional watt and SWR meter
- Simple frequency counter
- DDS VFO
- Portable solar power source

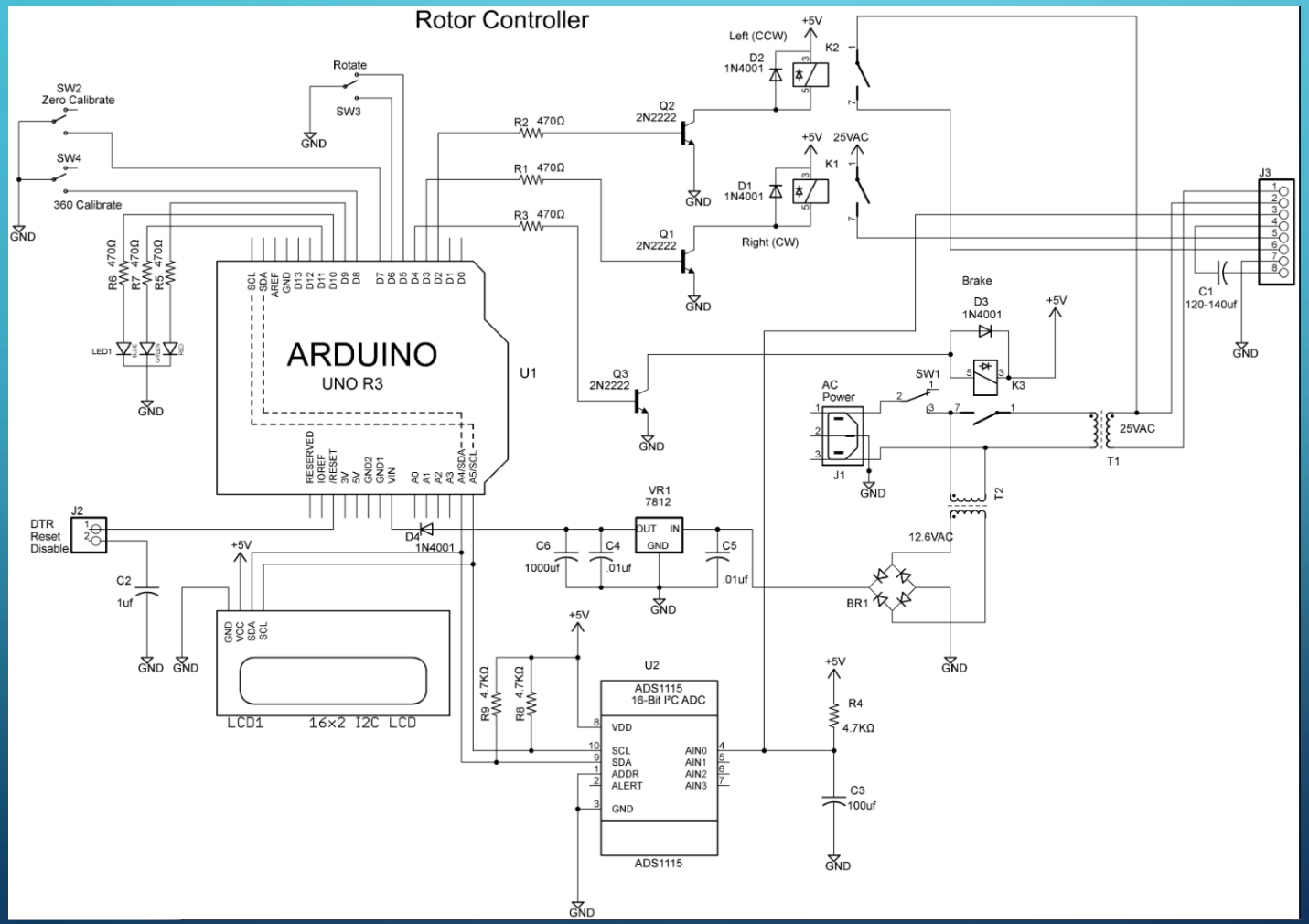
Piilota ^

Arduino antenna tuner

KÄYTÄNNÖN TOTEUTUKSIA

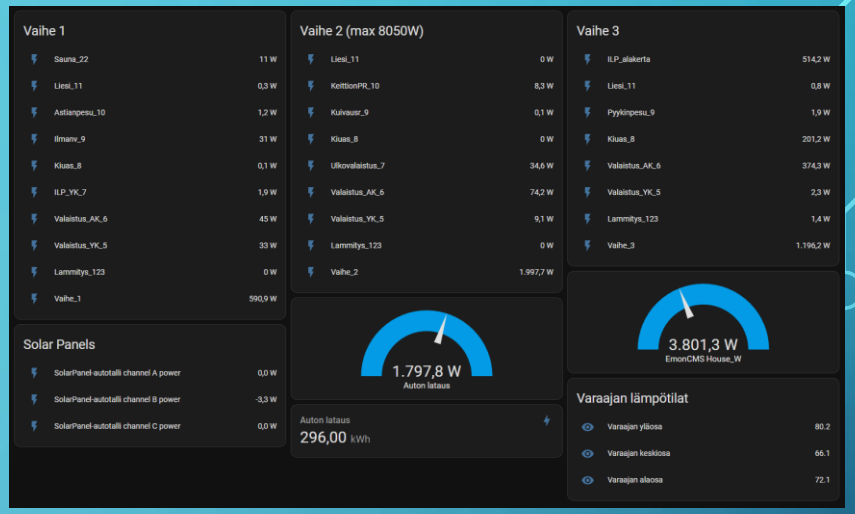


KÄYTÄNNÖN TOTEUTUKSIA

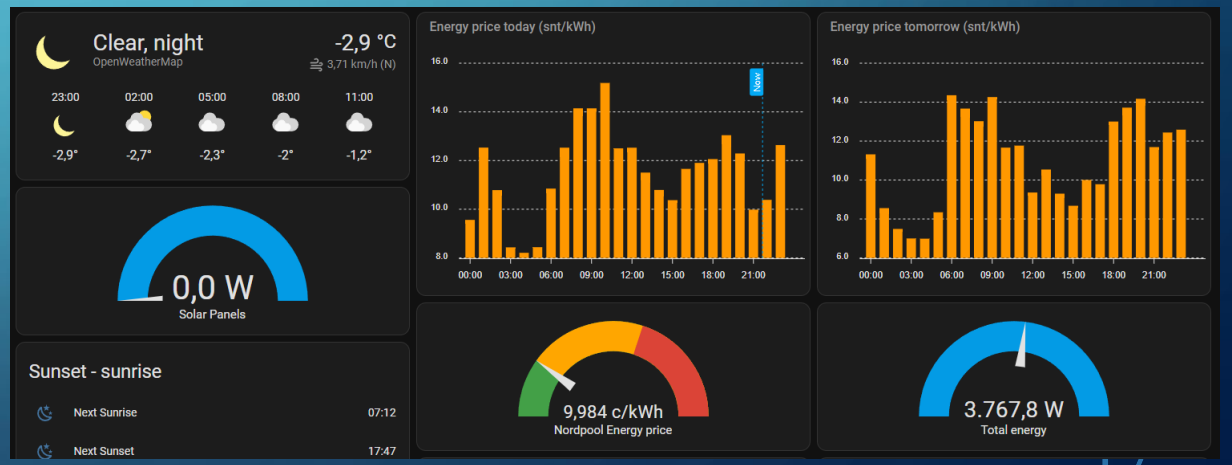


KÄYTÄNNÖN TOTEUTUKSIA KOTONA

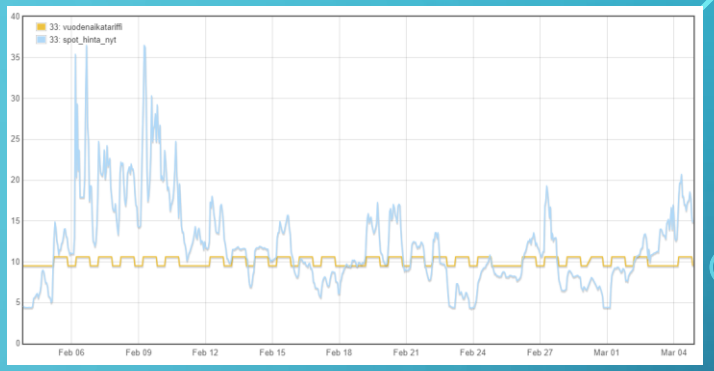
KÄYTÄNNÖN TOTEUTUKSIA



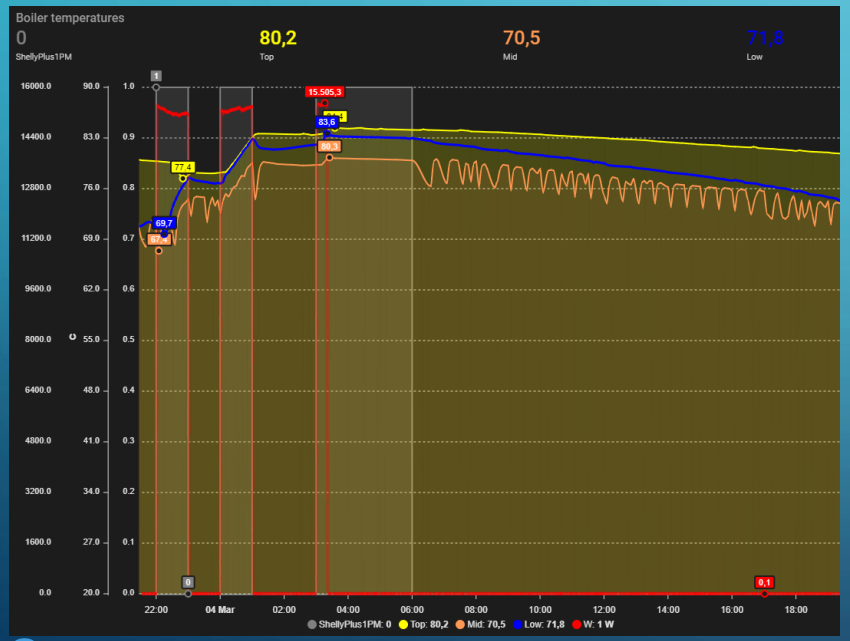
- HomeAssistant
- Erittäin suosittu linux –pohjainen käyttöjärjestelmä kodin ohjaukseen ja datan keräämiseen.
- Raspberry Pi tai PC asennus
- Oma ohjelmointikieli (YAML)
- Paljon graafisia palikoita valmiina

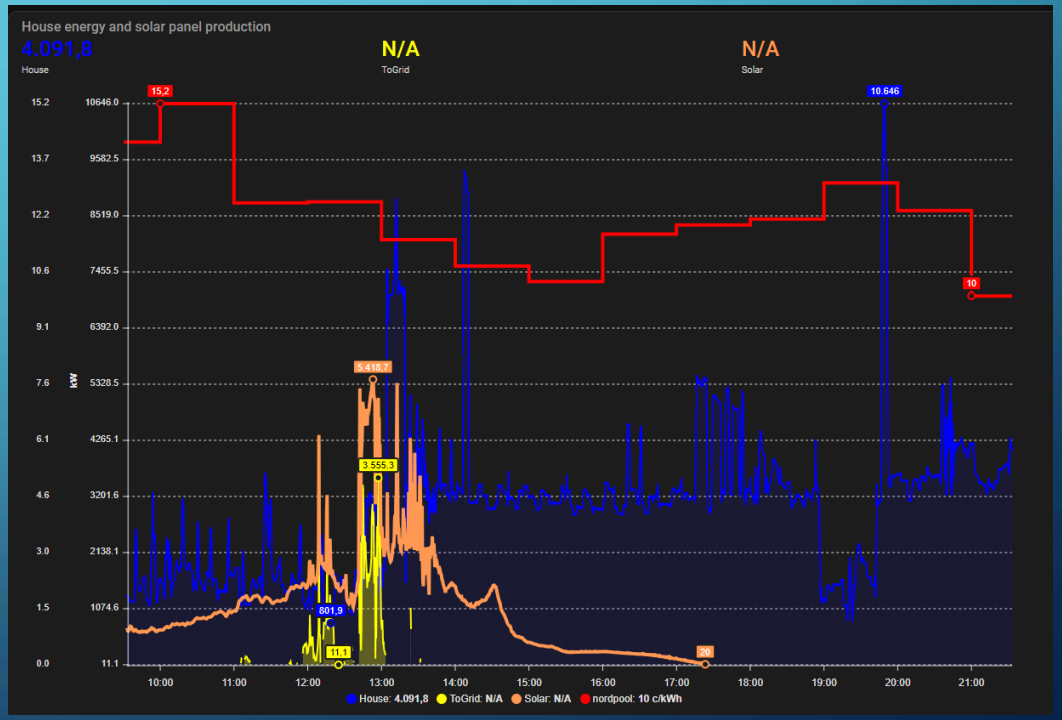


KÄYTÄNNÖN TOTEUTUKSIA



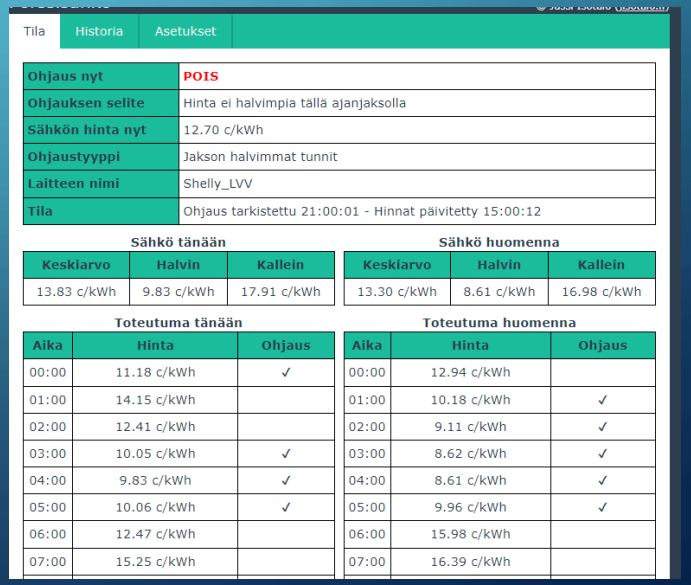
- Raspberry Pi voi ohjata sähkölaitteita ja kerätä mittadataa verkon yli





KÄYTÄNNÖN TOTEUTUKSIA

- Shellyn moduulit voi helposti ohjelmoida sammuttamaan valot määrättyyn aikaan, keräämään tietoa sähkönkulutuksesta, ohjaamaan spottisähkön mukaan vaikka LVV:a, katkaisemaan sähköt laitteilta kellonajan mukaan tai kännykällä jne.



Tila | Historia | Asetukset

Ohjaus nyt	POIS
Ohjauksen selite	Hinta ei halvimpia tällä ajanjaksolla
Sähkön hinta nyt	12.70 c/kWh
Ohjaustyyppi	Jakson halvimmat tunnit
Laitteen nimi	Shelly_LVV
Tila	Ohjaus tarkistettu 21:00:01 - Hinnat päivitetty 15:00:12

Sähkö tänään

Keskiarvo	Halvin	Kallein
13.83 c/kWh	9.83 c/kWh	17.91 c/kWh

Sähkö huomenna

Keskiarvo	Halvin	Kallein
13.30 c/kWh	8.61 c/kWh	16.98 c/kWh

Toteutuma tänään

Aika	Hinta	Ohjaus
00:00	11.18 c/kWh	✓
01:00	14.15 c/kWh	
02:00	12.41 c/kWh	
03:00	10.05 c/kWh	✓
04:00	9.83 c/kWh	✓
05:00	10.06 c/kWh	✓
06:00	12.47 c/kWh	
07:00	15.25 c/kWh	

Toteutuma huomenna

Aika	Hinta	Ohjaus
00:00	12.94 c/kWh	
01:00	10.18 c/kWh	✓
02:00	9.11 c/kWh	✓
03:00	8.62 c/kWh	✓
04:00	8.61 c/kWh	✓
05:00	9.96 c/kWh	✓
06:00	15.98 c/kWh	
07:00	16.39 c/kWh	